



**UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS VII GOVERNADOR ANTÔNIO MARIZ
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

JOSÉ ISAAC BARREIRO CAMPOS

**AGRAR CONNECT: UMA PROPOSTA DE DESENVOLVIMENTO DE SOFTWARE
PARA A GESTÃO AGRÍCOLA DE PEQUENOS E MÉDIOS AGRICULTORES**

**PATOS
2024**

JOSÉ ISAAC BARREIRO CAMPOS

**AGRAR CONNECT: UMA PROPOSTA DE DESENVOLVIMENTO DE SOFTWARE
PARA A GESTÃO AGRÍCOLA DE PEQUENOS E MÉDIOS AGRICULTORES**

Trabalho de Conclusão de Curso em
Ciência da Computação da Universidade
Estadual da Paraíba.

Área de concentração: Desenvolvimento
de software.

Orientador: Prof. Dr. Rodrigo Alves Costa

**PATOS
2024**

É expressamente proibido a comercialização deste documento, tanto na forma impressa como eletrônica. Sua reprodução total ou parcial é permitida exclusivamente para fins acadêmicos e científicos, desde que na reprodução figure a identificação do autor, título, instituição e ano do trabalho.

C198a Campos, Jose Isaac Barreiro.
Agrar Connect [manuscrito] : uma proposta de desenvolvimento de software para a gestão agrícola de pequenos e médios agricultores / Jose Isaac Barreiro Campos. -2024.
75 p. : il. colorido.
Digitado.
Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências Exatas e Sociais Aplicadas, 2024.
"Orientação : Prof. Dr. Rodrigo Alves Costa, Coordenação do Curso de Computação - CCEA."
1. Computação em nuvem. 2. Kotlin Multiplataforma. 3. Gestão agrícola. I. Título

21. ed. CDD 004

JOSÉ ISAAC BARREIRO CAMPOS

**AGRAR CONNECT: UMA PROPOSTA DE DESENVOLVIMENTO DE SOFTWARE
PARA A GESTÃO AGRÍCOLA DE PEQUENOS E MÉDIOS AGRICULTORES**

Trabalho de Conclusão de Curso apresentado ao
Curso de Bacharelado em Ciência da
Computação da Universidade Estadual da
Paraíba — Campus VII, em cumprimento à
exigência para obtenção do grau de Bacharel em
Ciência da Computação.

Aprovado em 26/06/2024

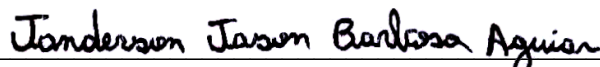
BANCA EXAMINADORA



Prof. Dr. Rodrigo Alves Costa
(Orientador)



Profa. Dra. Mikaelle Oliveira Santos Gomes
(Examinadora)



Prof. Dr. Janderson Jason Barbosa Aguiar
(Examinador)

À minha mãe, que mesmo diante das dificuldades, sempre continuou apoiando o meu sonho.

AGRADECIMENTOS

Lembro-me de, na adolescência, ouvir uma frase de Steve Jobs que dizia: "Concentre-se naquilo que você é bom, delegue todo o resto." Isso ficou gravado em minha mente e passei grande parte do meu ensino médio buscando descobrir em que eu era bom. Quem me conhece sabe o quanto sou apaixonado por música; boa parte das minhas lembranças têm alguma melodia que marca o momento. No ensino médio, comecei a explorar essa paixão: escrevi rap, cantei rap, tentei ser beat maker. No entanto, em cada uma dessas tentativas, sentia que faltava algo, um ingrediente essencial.

Alguns podem pensar que era talento, e talvez seja verdade. Mas hoje, ao olhar para trás, entendo que o ingrediente que faltava era o fascínio, aquele brilho no olho e a curiosidade por mais conhecimento a respeito de um tema. Não se trata apenas de ser o melhor, mas de genuinamente querer entender mais e mais sobre uma área.

Comigo, essa descoberta aconteceu com a computação. Na minha busca por encontrar algo que despertasse em mim uma curiosidade genuína, resolvi buscar entender como os jogos eram hackeados, para isso, precisava começar pela lógica de programação. De início, cometi muitos erros, algo que logo percebi que sempre seria parte da jornada, mas, de alguma forma, isso me deixava mais intrigado em descobrir a resposta. A sensação de evoluir o pensamento em busca de uma solução para um problema me lembrava de quando criança, montando um quebra-cabeça e encaixando as peças.

Hoje, estou convencido de que foi essa busca incansável pelo entendimento e pela descoberta que me levou a encontrar minha verdadeira paixão pela computação. A cada nova linha de código, sinto a mesma emoção que sentia ao tentar encaixar as peças de um quebra-cabeça quando criança.

Agora, olhando para o futuro, estou entusiasmado com as infinitas possibilidades que a computação oferece. Ansioso para continuar aprendendo e crescendo, sempre buscando novas maneiras de aplicar meus conhecimentos e fazer a diferença.

Dessa forma, gostaria de deixar meus sinceros agradecimentos a todos que contribuíram direta e indiretamente para minha jornada. Aos meus

professores, em especial: Rodrigo Alves Costa, Ingrid Morgane Medeiros de Lucena e Janderson Jason Barbosa Aguiar, pela dedicação em compartilhar seu conhecimento e me inspirarem a seguir adiante.

Aos meus amigos e colegas, em especial, Luanderson Bruno Martins Silva, Sthéfani Gomes Medeiros de Araújo e Felipe Dantas do Nascimento, pela parceria e apoio nos momentos de dúvidas.

À minha família, especialmente à minha mãe, Marinês Campos da Silva, por sempre acreditar em mim, e à minha irmã, Ana Alice Barreiro Campos, por estar sempre presente.

Gostaria também de expressar minha gratidão aos artistas de música hip-hop que tanto me inspiraram ao longo da minha trajetória, especialmente Eminem, Racionais MCs e Rashid. Suas letras, ritmos e histórias me acompanharam e motivaram nos momentos mais desafiadores, proporcionando uma trilha sonora que alimentou minha determinação e criatividade.

“Eu acredito que às vezes são as pessoas que ninguém espera nada que fazem as coisas que ninguém consegue imaginar.”

Alan Turing

RESUMO

O desenvolvimento tecnológico no agronegócio é um campo em constante evolução, enfrentando o desafio crucial da integração entre o campo e o meio digital. Com o surgimento de novas soluções tecnológicas, surge a questão de como lidar com a crescente quantidade de dados gerados e integrá-los de maneira eficaz. Esse desafio é ainda mais complexo devido às diferentes habilidades técnicas dos agricultores e à variedade de soluções disponíveis. Este trabalho propõe o desenvolvimento de um sistema integrado para pequenos e médios produtores rurais, visando melhorar a eficiência da gestão em suas cadeias produtivas. A motivação para esse projeto surge das diversas dificuldades enfrentadas pelos agricultores, que vão desde restrições financeiras e baixa taxa de alfabetização até problemas socioeconômicos como infraestrutura subdesenvolvida em áreas rurais. A metodologia adotada envolve o uso de Scrum, com organização do time, backlog, sprint e reuniões. Este trabalho é de natureza aplicada e tem como objetivo explorar o caso de uso proposto, buscando identificar soluções práticas para os desafios enfrentados pelos produtores rurais. A elaboração da arquitetura proposta envolveu a análise de trabalhos publicados e a identificação dos principais desafios enfrentados pelos produtores. Embora ainda existam limitações quanto à abrangência dos requisitos, foi possível chegar a um modelo de arquitetura escalável que incorpora tecnologias modernas e o uso da computação em nuvem, ao mesmo tempo em que considera as características de acesso e conectividade nas áreas rurais. Além disso, destacamos os desafios enfrentados durante a elaboração do projeto e apresentamos novos horizontes de melhorias futuras. Entre as melhorias possíveis estão o aprimoramento da interface do usuário para torná-la mais intuitiva e acessível, a expansão das funcionalidades para incluir recursos que utilizem IA e *Big Data* para auxiliar os produtores na gestão, a aplicação dos requisitos em diferentes contextos, e uma análise mais detalhada dos custos.

Palavras-Chave: Agro 5.0; Computação em Nuvem; Kotlin Multiplataforma; Gestão Agrícola.

ABSTRACT

Technological development in agribusiness is a field in constant evolution, facing the crucial challenge of integration between the field and the digital environment. With the emergence of new technological solutions, the question arises of how to deal with the growing amount of data generated and integrate it effectively. This challenge is even more complex due to the different technical skills of farmers and the variety of solutions available. This work proposes the development of an integrated system for small and medium-sized rural producers, aiming to improve management efficiency in their production chains. The motivation for this project arises from the various difficulties faced by farmers, ranging from financial constraints and low literacy rates to socioeconomic problems such as underdeveloped infrastructure in rural areas. The methodology adopted involves the use of Scrum, with team organization, backlog, sprint and meetings. This work is of an applied nature and aims to explore the proposed use case, seeking to identify practical solutions to the challenges faced by rural producers. The development of the proposed architecture involved the analysis of published works and the identification of the main challenges faced by producers. Although there are still limitations regarding the scope of requirements, it was possible to arrive at a scalable architectural model that incorporates modern technologies and the use of cloud computing, while also considering the characteristics of access and connectivity in rural areas. Furthermore, we highlight the challenges faced during the preparation of the project and present new horizons for future improvements. Among the possible improvements are the improvement of the user interface to make it more intuitive and accessible, the expansion of functionalities to include resources that use AI and Big Data to assist producers in management, the application of requirements in different contexts, and more detailed cost analysis.

Keywords: Agro 5.0; Cloud computing; Kotlin Multiplatform; Agricultural Management.

LISTA DE FIGURAS

Figura 1 - Opções de compartilhamento de código no KMP.....	29
Figura 2 - Plataformas de destino.....	30
Figura 3 - Principais tipos de monólitos.....	32
Figura 4 - Produtividade versus complexidade entre monólitos e microsserviços....	33
Figura 5 - Comunicação entre os microsserviços.....	34
Figura 6 - Exemplo de agregação de API.....	37
Figura 7 - Visão geral da arquitetura.....	42
Figura 8 - Visão geral da arquitetura, usando API Gateway ebff.....	44
Figura 9 - Diagrama de caso de uso: visão alto nível do sistema.....	50
Figura 10 - Diagrama de caso de uso: módulo financeiro.....	51
Figura 11 - Diagrama de caso de uso: módulo financeiro - integrações.....	52
Figura 12 - Diagrama de caso de uso: módulo de Associações e Cooperativas.....	54
Figura 13 - Diagrama de classe: Autenticação e controle de acesso multimódulo...	56
Figura 14 - Diagrama de classe: Cadastro de contas a pagar e receber.....	58
Figura 15 - Diagrama de classe: Fluxo de caixa futuro.....	59
Figura 16 - Diagrama de classe: Cadastro de membros em uma associação.....	60
Figura 17 - Diagrama de classe: Registro de atas.....	62
Figura 18 - Diagrama de classe: Sistema de votação online.....	63
Figura 19 - Protótipo: tela de login.....	64
Figura 20 - Protótipo: tela inicial do sistema.....	65
Figura 21 - Protótipo: menu e submenus.....	66
Figura 22 - Protótipo: modelo formulário com poucos campos.....	67
Figura 23 - Protótipo: modelo formulário com muitos campos a serem preenchidos	67

LISTA DE QUADROS

Quadro 1 - Práticas de Extreme Programming.....	25
Quadro 2 - Módulo Financeiro: requisitos funcionais.....	45
Quadro 3 - Módulo Financeiro: requisitos não funcionais.....	46
Quadro 4 - Módulo de Associações e Cooperativas: requisitos funcionais.....	48
Quadro 5 - Módulo de Associações e Cooperativas: requisitos não funcionais.....	49

SUMÁRIO

1	INTRODUÇÃO	13
1.1	Contextualização do tema	13
1.2	Problemática	14
1.3	Justificativa	15
1.4	Objetivos	17
1.5	Organização do trabalho	18
2	FUNDAMENTAÇÃO TEÓRICA	20
2.1	Contextualizando sobre a evolução do agro 1.0 para o agro 4.0	20
2.2	Agro 5.0 - comunicação em tempo real e decisões apoiada em dados . 21	
2.3	Metodologias de desenvolvimento de software	22
2.3.1	Métodos ágeis	23
2.3.1.1	<i>Extreme Programming (XP)</i>	24
2.3.1.2	<i>Scrum</i>	26
2.4	Desenvolvimento multiplataforma	27
2.4.1	Kotlin multiplataforma	27
2.5	Comunicação assíncrona e microsserviços vs monólitos	29
2.5.1	Monólitos	30
2.5.2	Microsserviços	32
2.5.3	Comunicação assíncrona	34
2.6	API gateway	35
3	METODOLOGIA	38
4	RESULTADOS	40
4.1	Definição do software	40
4.1.1	Arquitetura do software	40
4.1.2	Funcionalidades identificadas para o sistema	44
4.1.2.1	<i>Módulo financeiro</i>	44
4.1.2.2	<i>Modulo de associações e cooperativas</i>	46

4.2	Modelagem dos requisitos	48
4.2.1	Diagrama de caso de uso	48
4.2.1.1	<i>Diagrama de caso de uso: Módulo financeiro</i>	50
4.2.1.2	<i>Diagrama de caso de uso: Módulo de associações e cooperativas</i>	52
4.2.2	Diagrama de classes	54
4.2.2.1	<i>Diagrama de classes: Cadastro de contas a pagar e receber</i>	56
4.2.2.2	<i>Diagrama de classes: Fluxo de caixa futuro</i>	58
4.2.2.3	<i>Diagrama de classes: Cadastro de membros em uma associação</i>	59
4.2.2.4	<i>Diagrama de classes: Registro de atas</i>	60
4.2.2.5	<i>Diagrama de classes: Sistema de votação online</i>	62
4.3	Prototipagem	63
5	CONCLUSÃO	68
	REFERÊNCIAS	71

1 INTRODUÇÃO

Gerenciar um pequeno e médio empreendimento convencional é uma tarefa que exige uma quantidade de esforço considerável, especialmente diante dos desafios na gestão de recursos limitados, alta competitividade e constante mudança no mercado. Na agricultura familiar, esses desafios se intensificam devido à natureza complexa da atividade, que envolve produção, comercialização e gestão em um contexto familiar. Uma administração eficiente e bem estruturada simplifica o controle dos custos de produção, permitindo tomar decisões mais precisas e gerenciar a propriedade da melhor forma possível (Batista *et al.* 2023, p. 12).

Assim como destaca Figueiredo *et al.* (2023, p. 192), "Os processos produtivos e comerciais de excedentes das cooperativas de agricultores familiares exigem o mínimo de organização para tomada de decisões assertivas". No entanto, a falta de conhecimento técnico e a dificuldade de investimentos em novas tecnologias intensificam a carência de processos de gestão na tomada de decisões.

A relevância do agronegócio na economia brasileira é notável e em um cenário global extremamente competitivo, o setor vem se renovando a cada ano e investindo cada vez mais em soluções tecnológicas para apoiar tal crescimento (EMBRAPA, 2018). Com efeito, tal crescimento demanda um alto investimento no desenvolvimento de novas soluções que sejam aplicadas no contexto de médios e grandes produtores rurais.

1.1 Contextualização do tema

O agronegócio contribuiu com cerca de 24,5% do PIB (Produto Interno Bruto) brasileiro no primeiro trimestre de 2023 (CEPEA; CNA, 2023). Tamanha participação tem gerado e trazido grandes avanços para o setor, com alto foco na utilização e no desenvolvimento de novas soluções tecnológicas que se utilizam de tecnologias de ponta, tais como: Internet das Coisas (em inglês, *Internet of Things* – IoT), Inteligência Artificial (IA), Visão Computacional e *Big Data*.

Essas tecnologias, quando aplicadas ao agronegócio, têm o potencial de otimizar processos e evitar desperdícios de recursos. Segundo Cócáro e Jesus (2008), a aplicação de tecnologia da informação possibilita que os produtos agropecuários brasileiros se tornem mais competitivos em preço e qualidade,

conseguindo assim um melhor posicionamento no mercado mundial. Em função das peculiaridades do nosso sistema de produção, faz-se necessário o desenvolvimento de tecnologias nacionais que atendam às nossas condições de produção (Cócaro; Jesus, 2008).

Segundo Dias *et al.* (2012), soluções baseadas em computação em nuvem, modelo que possibilita o acesso remoto e sob demanda de recursos computacionais pela internet e que tem aberto novas oportunidades para o setor de agronegócio, utilizam-se de uma demanda variável de recursos computacionais sem necessariamente ter que investir em uma ampla infraestrutura de TI. Essa abordagem tem se mostrado especialmente vantajosa para o agronegócio, pois o setor frequentemente enfrenta variações sazonais nas demandas, tornando inviável manter uma infraestrutura estática e superdimensionada durante todo o ano.

De acordo com a Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA, 2018, p. 152), “haverá uma progressiva necessidade de otimização e aperfeiçoamento do desempenho dos sistemas de produção, (...). Análises integradas e prospectivas deverão apoiar a tomada de decisão e incrementar a capacidade dos produtores”. Tais necessidades demandam o uso de novas tecnologias ao setor, novas formas de aplicação dessas tecnologias e integração do meio físico ao digital, melhorando processos e otimizando as tomadas de decisões com informações cada vez mais precisas.

1.2 Problemática

Nos tempos atuais, a otimização e a automatização de processos tornaram-se evidentes na nossa sociedade. Cada vez mais, testemunhamos aplicações de tecnologia em nosso cotidiano, resultando em melhorias significativas de tempo e recursos em diversas áreas. O agronegócio não é exceção, estando todo o setor em constante evolução e em busca de melhorias.

Exemplos como a utilização de drones e sensores que podem ser utilizados para auxiliar a aplicação de fertilizantes nas plantações, capazes de avaliar a distribuição de nutrientes da cultura, com o objetivo de ajustar a quantidade de fertilizantes a serem aplicados antecipadamente. Outra aplicação é o monitoramento da plantação, que possibilita a realização de atividades tais como: monitorar seu

crescimento, avaliar o estresse hídrico, verificar se a plantação foi afetada por alguma praga e se é necessária a aplicação de fertilizantes (Pino, 2019).

Além de aplicações diretas no campo, existem soluções que utilizam software que buscam auxiliar a gestão do campo, como o ClimAPI, que é uma API (em inglês, *Application Programming Interface*) desenvolvida pela Embrapa capaz de fornecer dados meteorológicos que ajudam a antecipar tendências do clima, fundamental para a tomada de decisão no campo (Galinari, 2022). Todas as informações fornecidas pela API são atualizadas a cada seis horas e podem ser utilizadas por desenvolvedores e empresas na criação de novas soluções que permitam apoiar o produtor no campo.

Essas inovações possuem um grande potencial para otimizar a produtividade, reduzir custos e minimizar o impacto ambiental. No entanto, mesmo com todo o avanço do setor nos últimos anos, é importante reconhecer que uma parcela considerável de pequenos produtores ainda não utilizam recursos de Tecnologia da Informação e Comunicação (TIC) na cadeia produtiva.

Vários obstáculos dificultam o acesso e a implementação das TIC no campo, incluindo restrições financeiras, baixas taxas de alfabetização e infraestrutura subdesenvolvida em áreas rurais, como apontado por Lucas (2023). Além disso, a falta de qualificação técnica também é um desafio significativo. Embora existam soluções de diferentes níveis técnicos disponíveis no mercado, os pequenos e médios produtores, que muitas vezes estão na ponta da cadeia produtiva, nem sempre possuem o conhecimento necessário para escolher a melhor solução que atenda às suas necessidades específicas. Isso é evidenciado pela falta de uma plataforma centralizada que possa reunir essas soluções e permitir que elas trabalhem de maneira integrada.

1.3 Justificativa

Quando falamos em desenvolvimento tecnológico no agronegócio, o setor tem sido um campo de constante evolução. Um dos desafios cruciais que o setor enfrenta é a integração entre o campo e o meio digital. Com novas soluções tecnológicas sendo desenvolvidas, surge o questionamento sobre como lidar com a crescente quantidade de dados gerados e integrar esses dados de maneira eficaz.

Isso se torna ainda mais complexo devido às diferentes habilidades técnicas dos agricultores e à variedade de soluções disponíveis.

Uma abordagem centralizada é fundamental para gerenciar as várias soluções e os dados heterogêneos gerados. Isso requer não apenas a agregação dos dados, mas também a apresentação das informações de forma clara e objetiva.

De acordo com Bolfe *et al.* (2020, p. 391), a maximização da utilidade das tecnologias disponíveis para os produtores rurais requer a implementação de mecanismos e sistemas capazes de agregar e processar a grande quantidade de dados gerados por essas tecnologias. Além disso, enfatizam a importância de que tais ferramentas sejam capazes de fornecer informações que possam ser prontamente utilizadas na tomada de decisões.

No mercado atual, há uma ampla variedade de aplicativos móveis projetados para auxiliar agricultores na gestão de suas atividades no campo. Muitos desses aplicativos oferecem soluções informativas em áreas específicas, como o Agrobase¹, que simplifica a identificação de ervas daninhas, doenças, insetos e pragas em campos agrícolas, além de sugerir produtos de proteção de culturas para resolver problemas específicos.

Outra solução notável é o Aegro², que se destaca no mercado por suas diversas funcionalidades. Entre elas, destacam-se a capacidade de registrar atividades diretamente do campo, mesmo em áreas sem conectividade à internet. Além disso, o Aegro permite mapear e medir as áreas dos talhões, planejar e controlar atividades da fazenda, gerenciar abastecimentos e manutenções de máquinas, registrar leituras de pluviômetros e acompanhar a quantidade acumulada de precipitação.

Muitas das soluções existentes enfrentam desafios significativos. Elas frequentemente não fornecem uma apresentação simples e concisa dos dados, o que limita sua utilidade para diferentes níveis de necessidades e conhecimento. Além disso, é importante notar que muitas dessas soluções são projetadas principalmente para atender às demandas dos médios e grandes produtores.

No entanto, quando consideramos a realidade dos pequenos agricultores e a instabilidade do acesso à Internet em áreas rurais, é fundamental abordar a

¹ Agrobase está disponível na loja de apps do Android em https://play.google.com/store/apps/details?id=lt.farmis.apps.farmiscatalog&hl=pt_BR&gl=US

² Aegro: <https://aegro.com.br/plataforma>

possibilidade de permitir o cadastro de dados mesmo quando os dispositivos estão desconectados. Dessa forma, os pequenos agricultores podem inserir informações no sistema mesmo em locais sem acesso à Internet, podendo esses dados ser sincronizados e armazenados em nuvem à *posteriori*, quando uma conexão estiver disponível.

O atual trabalho busca abordar desafios como esses, priorizando funcionalidades essenciais para um sistema de gestão eficaz no campo. Nesse sentido, torna-se necessário o desenvolvimento de módulos para gestão que abranjam áreas críticas, como o módulo financeiro, módulo de associações e cooperativas, bem como um módulo dedicado ao plantio e colheita.

Uma característica fundamental deste projeto é a adoção de tecnologias de computação em nuvem, que proporcionam flexibilidade e escalabilidade. A computação em nuvem permitirá que a solução se adapte dinamicamente às necessidades em constante mudança dos agricultores e do setor agrícola como um todo. Isso garantirá que tanto pequenos quanto médios produtores possam usufruir dos benefícios da tecnologia sem comprometer recursos financeiros escassos.

Dessa forma, almejamos contribuir para a superação dos desafios enfrentados pelos pequenos produtores rurais e promover o uso eficiente da tecnologia no agronegócio, o que, por sua vez, pode impactar positivamente a produtividade e o desenvolvimento das comunidades rurais. Ao facilitar o acesso a recursos tecnológicos e proporcionar ferramentas de gerenciamento robustas, acreditamos que podemos capacitar os agricultores a enfrentar os desafios da modernização agrícola, contribuindo, assim, para a prosperidade das comunidades rurais e para o crescimento sustentável do setor agrícola.

1.4 Objetivos

Com base na problematização apresentada anteriormente, propomos o desenvolvimento de um sistema integrado de gestão (*Enterprise Resource Planning* – ERP). Denominaremos tal sistema como AgrarConnect³, que tem como principal objetivo oferecer soluções integradas de gestão, permitindo que pequenos e médios

³ AgrarConnect é a união das palavras "agrário" e "connect", representando a conexão do campo com o meio digital.

produtores rurais tenham acesso a benefícios, tais como: melhoria na tomada de decisões, aumento da produtividade e redução de riscos.

O principal diferencial desta aplicação é que ela está baseada em uma arquitetura em nuvem, mas provendo disponibilidade de funcionalidades básicas mesmo sem acesso à internet. A adoção de uma arquitetura em nuvem garantirá a escalabilidade do sistema, permitindo que os agricultores acessem suas informações de qualquer lugar e a qualquer momento, além de assegurar a segurança e a integridade dos dados.

A disponibilidade de funcionalidades básicas sem acesso a internet garante que o AgrarConnect esteja alinhado com as necessidades de uso dos agricultores, possibilitando que possam utilizar todo o potencial da computação em nuvem, mas não o restringindo quando estiverem em locais com oscilações na conectividade.

Assim, os objetivos específicos deste trabalho são:

- Contextualizar agricultura historicamente, e compreender as possibilidades propostas e desafios ainda presentes no Agro 5.0;
- Pesquisar soluções existentes no mercado para resolução de problemas semelhantes;
- Elaboração de projeto de software, com modelagem de requisitos e elaboração de diagramas de arquitetura e de fluxo de trabalho;
- Pesquisa sobre métodos de desenvolvimento para propor um sistema escalável e multiplataforma;
- Desenvolvimento de protótipo de baixa fidelidade do sistema proposto.

1.5 Organização do trabalho

No Capítulo 2, iniciaremos com uma revisão bibliográfica que introduzirá conceitos fundamentais sobre as diferentes fases de evolução do setor agrícola, abrangendo desde o Agro 1.0 até a mais recente, Agro 5.0. Exploraremos as características distintivas de cada fase e, em seguida, abordaremos duas das principais metodologias de desenvolvimento de software em uso atualmente, assim como as tecnologias aplicadas ao contexto do AgrarConnect.

No Capítulo 3, detalharemos a metodologia adotada nesta pesquisa, juntamente com seus objetivos. Em seguida, no Capítulo 4, apresentaremos os resultados obtidos, destacando tanto os principais requisitos funcionais quanto os

não funcionais dos diversos módulos. Além disso, forneceremos uma visão detalhada da arquitetura desenvolvida para o projeto, incluindo os mecanismos de comunicação entre os serviços e módulos do sistema.

Por fim, no Capítulo 5, apresentaremos as conclusões, as quais incluirão sugestões para possíveis trabalhos futuros e os desafios que foram encontrados ao longo do processo de pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

O setor agrícola, ao longo dos anos, passou por significativas transformações impulsionadas pela integração de tecnologias avançadas e inovações nos processos de produção. A evolução do agronegócio, conhecida como Agronegócio 3.0 e 4.0, representa uma revolução na forma como as atividades agrícolas são conduzidas, incorporando sistemas inteligentes, automação e conectividade em toda a cadeia produtiva.

É fundamental compreender a jornada evolutiva que a agricultura tem percorrido ao longo dos anos, até chegar ao Agro 5.0. Desde os primórdios da agricultura, marcados pela domesticação de plantas e animais (Agro 1.0), até as revoluções industriais e a automação de processos agrícolas (Agro 4.0), cada fase representou um salto significativo em termos de eficiência e produtividade.

No entanto, é com o advento do Agro 5.0 que testemunhamos a convergência de tecnologias disruptivas que não apenas prometem otimizar a produção, mas também revolucionar a própria essência da agricultura, tornando-a mais inteligente, adaptável e sustentável do que nunca.

2.1 Contextualizando sobre a evolução do agro 1.0 para o agro 4.0

O agro 1.0 tem sua origem por volta do início do século 20 em que a força de trabalho era provida pela mão de obra das famílias, utilizando instrumentos manuais, apoiados pela tração animal. É conhecido pela sua baixa produção agrícola, tendo como foco principal o consumo próprio.

Durante as décadas de 1950 e 1960, a agricultura enfrentava desafios expressivos devido à sua estrutura rudimentar. O predomínio do trabalho manual nas atividades agropecuárias delineava esse panorama, enquanto a carência de tecnologia e informações traduzia-se em baixos rendimentos por hectare e produção inadequada. Nesse contexto, a necessidade premente de ampliar a produção agrícola para atender à crescente demanda interna, em meio a um cenário de industrialização intensa e escassez alimentar, desencadeou a Revolução Verde.

Essa revolução introduziu uma série de inovações tecnológicas, incluindo modificações genéticas em sementes, novas técnicas de fertilização do solo e o uso intensivo de máquinas movidas a motor a combustão. A mecanização no campo

tornou-se uma tendência, substituindo gradualmente a tração manual. Essas transformações culminaram na implantação da Agricultura 2.0, marcando um período de transição fundamental na modernização e aumento da eficiência do setor agrícola brasileiro (Massruhá *et al.* 2020, p. 26).

Segundo Sordi e Vaz (2020), a agricultura 3.0 refere-se à intensificação ocorrida no século 20 na utilização de programas de computador e técnicas robóticas em máquinas agrícolas que passaram a operar com mais precisão e eficiência. Para Massruhá *et al.* (2020, p. 27), a evolução das tecnologias desde então foi notável, superando as expectativas da época. Máquinas e implementos surgiram para aprimorar a eficiência das atividades agrícolas, dando origem à agricultura de precisão, marco que caracteriza o advento da agricultura 3.0.

A evolução para o Agro 4.0 incorpora uma integração ainda maior de tecnologias digitais, como Internet das Coisas (IoT), *Big Data*⁴ e automação avançada, possibilitando uma gestão agrícola mais precisa e automatizada. Esse avanço desperta um interesse crescente nos âmbitos político, econômico e ambiental. Tecnologias como edição genômica de culturas, monitoramento meteorológico por satélite, softwares de gestão agrícola, sensores para controle de pesticidas e irrigação, mapeamento digital de índices de fertilidade, umidade, temperatura e condições físico-químicas do solo são características da agricultura 4.0 (Viola; Mendes, 2022).

2.2 Agro 5.0 - comunicação em tempo real e decisões apoiada em dados

É inegável que nossa interação crescente com a tecnologia resulta em uma vasta quantidade de dados sobre nossos hábitos e preferências a cada ano. Há anos, as gigantes da tecnologia, conhecidas como *big tech*⁵, têm explorado esses dados para diversos fins. Espera-se que outras áreas também comecem a se beneficiar cada vez mais dessa riqueza de informações. Com a convergência de tecnologias como Internet das Coisas, inteligência artificial e outras, surge uma oportunidade sem precedentes para explorar e aproveitar todo o potencial desses dados.

⁴ Big data é a área do conhecimento que estuda como tratar, analisar e obter informações a partir de conjuntos de dados muito grandes.

⁵ O termo *big tech* se refere a grandes empresas que exercem domínio no mercado de tecnologia e inovação. Destacam-se nomes como Apple, Google, Amazon, Microsoft e OpenAI.

O Agro 5.0 representa a mais recente inovação nos modelos de produção do agronegócio. Ao unir a coleta de dados em tempo real e em larga escala com as tecnologias de inteligência artificial (IA) e análise de dados, possibilita aos produtores rurais uma gestão mais dinâmica e sustentável das fazendas. Após a introdução do conceito de agricultura digital, ficou claro que sua essência reside na integração de tecnologias digitais em todas as etapas da cadeia de valor agrícola, visando a obtenção de vantagens competitivas e benefícios socioambientais.

Esse processo fundamenta-se no aproveitamento do vasto volume de dados gerados ao longo de todas as fases da produção agrícola, desde a pré-produção até a pós-produção, evidenciando a importância da análise e do processamento desses dados para impulsionar a eficiência e a sustentabilidade do setor (Massruhá *et al.* 2020, p. 29).

A computação em nuvem desempenha um papel fundamental ao fornecer a infraestrutura necessária para armazenar, processar e compartilhar dados de forma ágil e acessível. Sua capacidade de escalar recursos conforme a demanda e permitir acesso remoto de diversos dispositivos oferece aos produtores agrícolas maior flexibilidade e eficiência na gestão e análise dos dados.

Shepherd *et al.* (2018) argumentam que a capacidade de empregar tecnologias digitais para transformar dados precisos em conhecimento é fundamental para sustentar e impulsionar a tomada de decisões complexas na fazenda e ao longo da cadeia de valor. Isso possibilitará a transição da agricultura de precisão para a agricultura de decisão.

2.3 Metodologias de desenvolvimento de software

Conforme destaca Sommerville (2018), a engenharia de software tem o propósito de apoiar o desenvolvimento de software, abrangendo técnicas que envolvem desde a especificação até o projeto e a evolução do programa. Nesse contexto, surgem dois modelos de processo amplamente reconhecidos: o modelo em cascata e o modelo incremental.

O modelo em cascata, também conhecido como ciclo de vida do software, é caracterizado por processos que enfatizam o planejamento e a sequencialidade. Em cada fase do processo, documentos são produzidos, revisados e aprovados. Embora seja frequentemente associado a uma abordagem linear, é importante

observar que há flexibilidade dentro do modelo, permitindo a ocorrência de feedbacks entre as fases. Isso significa que documentos podem ser modificados para refletir alterações realizadas ao longo do processo.

No entanto, o modelo em cascata não é particularmente flexível em relação a mudanças bruscas nos requisitos. Ele é mais adequado quando os requisitos estão bem definidos e não se espera uma mudança significativa durante a fase de desenvolvimento. Portanto, o sucesso do uso desse modelo depende da clareza e da estabilidade dos requisitos iniciais.

Por outro lado, o modelo incremental é amplamente preferido no cenário atual de desenvolvimento de software, especialmente quando se adotam abordagens ágeis. A premissa fundamental desse modelo é a exposição antecipada de incrementos ou versões do sistema desde o início do projeto. Isso permite que cada incremento entregue seja avaliado pelo usuário. Em caso de inconformidades nos requisitos, é possível avaliar e priorizar ajustes ou mudanças sem incorrer em custos significativos ou interrupções substanciais no desenvolvimento.

Comparado ao modelo em cascata, o modelo incremental oferece uma abordagem mais flexível e interativa, tornando-se uma escolha valiosa em cenários no qual os requisitos podem evoluir ao longo do projeto e a colaboração contínua com os usuários é essencial para o sucesso.

2.3.1 Métodos ágeis

Vivemos em um mundo cada vez mais conectado e em constante mudança. Com o advento da internet, soluções de software passaram a ser acessíveis mediante alguns cliques. Por consequência, um ambiente muito mais dinâmico e acessível para a busca e o desenvolvimento de novas soluções foi gerado, possibilitando que elas sejam requisitadas de forma mais rápida e constante.

Nas décadas de 80 e 90 já era perceptível a necessidade de um processo de desenvolvimento que se adequasse aos cenários de entregas de software mais ágeis. O movimento ganhou mais força na final da década de 90 com o desenvolvimento da noção de abordagens ágeis, como Metodologia de Desenvolvimento de Sistemas Dinâmicos (DSDM, do inglês *Dynamic Systems Development Method*) (Stapleton, 1997), *Scrum* (Schwaber; Beedle, 2001) e *Extreme Programming* (Beck, 1999).

De acordo com Sommerville (2018), o desenvolvimento ágil é caracterizado por um processo incremental de produção de software, com incrementos pequenos, e novas versões do sistema disponibilizadas aos clientes em ciclos de desenvolvimento com duração de duas a quatro semanas. Tais métodos incluem a participação dos clientes no processo de desenvolvimento para obter feedback rápido sobre a evolução dos requisitos. Como resultado, a ênfase recai mais sobre a comunicação efetiva do que sobre reuniões formais e documentação extensa. Dentre alguns dos métodos existentes, focaremos em dois principais paradigmas: *Extreme Programming (XP)* e *Scrum*.

2.3.1.1 *Extreme Programming (XP)*

O termo *Extreme Programming* foi cunhado por Beck (1999) em seu livro "*Extreme Programming Explained: Embrace Change*" e é talvez um dos mais conhecidos e utilizados dos métodos ágeis. O XP define um conjunto de práticas que refletem os princípios dos métodos ágeis, como destaca Soares (2004, p. 3): "a maioria das regras do XP causa polêmica à primeira vista e muitas não fazem sentido se aplicadas isoladamente. É a sinergia de seu conjunto que sustenta o sucesso de XP (...)".

Quadro 1 - Práticas de Extreme Programming

(continua)

PRINCÍPIO OU PRÁTICA	DESCRIÇÃO
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema de modo que não se desenvolvem 'ilhas de conhecimento', e todos os desenvolvedores assumem a responsabilidade por todo o código. Qualquer um pode mudar qualquer coisa.
Integração contínua	Assim que o trabalho em uma tarefa é concluído, ele é integrado ao sistema completo. Após qualquer integração desse tipo, todos os testes de unidade no sistema devem passar.
Planejamento incremental	Os requisitos são registrados em 'cartões de história', e as histórias a serem incluídas em um lançamento são determinadas de acordo com o tempo disponível e com sua prioridade relativa. Os desenvolvedores decompõem essas histórias em 'tarefas' de desenvolvimento (Figuras 3.5 e 3.6).

(conclusão)

PRINCÍPIO OU PRÁTICA	DESCRIÇÃO
Representante do cliente	Um representante do usuário final do sistema (o cliente) deve estar disponível em tempo integral para o time de programação. Em um processo como esse, o cliente é um membro do time de desenvolvimento, sendo responsável por levar os requisitos do sistema ao time, visando sua implementação.
Programação em pares	Os desenvolvedores trabalham em pares, conferindo o trabalho um do outro e oferecendo o apoio necessário para que o resultado final seja sempre satisfatório.
Refatoração	Todos os desenvolvedores devem refatorar o código continuamente logo que sejam encontradas possíveis melhorias para ele. Isso mantém o código simples e de fácil manutenção.
Projeto (design) simples	Deve ser feito o suficiente de projeto (design) para satisfazer os requisitos atuais, e nada mais.
Lançamentos pequenos	O mínimo conjunto útil de funcionalidade que agregue valor ao negócio é desenvolvido em primeiro lugar. Os lançamentos do sistema são frequentes e acrescentam funcionalidade à primeira versão de uma maneira incremental.
Ritmo sustentável	Grandes quantidades de horas extras não são consideradas aceitáveis, já que o efeito líquido muitas vezes é a diminuição da qualidade do código e da produtividade no médio prazo.
Desenvolvimento com testes a priori (test-first)	Um framework automatizado de teste de unidade é utilizado para escrever os testes de um novo pedaço de funcionalidade antes que ela própria seja implementada.

Fonte: Sommerville (2018, p. 63)

Muitas das práticas e princípios que o XP definiu são amplamente utilizados pelo mercado atualmente (ver Quadro 1). Santos (2013, p. 16) destaca que o XP é amplamente utilizado durante o desenvolvimento do sistema quando temos equipes pequenas e médias e os requisitos costumam ser incertos ou mudam com frequência. O principal foco está em tornar rápido o desenvolvimento do sistema por meio de entregas rápidas e frequentes, envolvendo o cliente de forma ativa no processo de desenvolvimento e na priorização das tarefas a serem desenvolvidas.

No contexto do XP, o sistema é desenvolvido em ciclos de releases, nos quais cada entrega representa um pequeno incremento de valor. Os requisitos são coletados por meio de cenários ou histórias de usuários, descrições simples feitas pelos próprios usuários, que posteriormente, são refinadas pela equipe de desenvolvimento em tarefas e, posteriormente, software.

Conforme destaca Sommerville (2018), diferentemente da engenharia de software tradicional, que busca montar um planejamento do sistema que aceite

mudanças, o XP reconhece que mudanças acontecem e que o software será ajustado se necessário. O software é então ajustado conforme as mudanças ocorrem efetivamente, proporcionando assim mais dinamismo ao ciclo de desenvolvimento.

2.3.1.2 Scrum

O termo "*Scrum*" deriva de um estudo seminal conduzido por Hirotaka Takeuchi e Ikujiro Nonaka em 1986, publicado na *Harvard Business Review*, no qual compararam equipes de *Rugby* de alto desempenho com equipes tradicionais, e extrapolam esta comparação para o desenvolvimento de produtos por equipes ágeis com aquelas que adotavam o método de multitarefas (Silva; Guimarães Filho, 2023). Essa analogia inspirou a criação do Scrum como um framework ágil para gestão e desenvolvimento de produtos complexos, fundamentado no empirismo.

O estudo de Takeuchi e Nonaka (1986) destacou a eficácia das equipes auto-organizadas e multidisciplinares, que compartilham objetivos claros e trabalham de forma colaborativa para alcançá-los, uma abordagem fundamental no Scrum. A partir desta pesquisa, o Scrum foi concebido como um processo iterativo e incremental que permite a entrega rápida e adaptativa de produtos de alta qualidade.

Na abordagem do Scrum, a equipe de desenvolvimento trabalha de forma colaborativa e coesa, com o objetivo de entregar software funcional de alta qualidade dentro de ciclos curtos e bem definidos, conhecidos como *sprints*, geralmente com duração de duas a quatro semanas (Huff, 2023). Durante cada *sprint*, as tarefas a serem realizadas são definidas e o progresso é acompanhado de perto por meio de reuniões diárias, chamadas de *Daily Scrum*. Ao final de cada ciclo, uma reunião de retrospectiva é realizada para identificar os pontos de melhoria e ajustes necessários para os próximos sprints.

Além disso, o Scrum também enfatiza a importância da transparência, inspeção e adaptação contínua. Por exemplo, o *Product Backlog* é uma lista dinâmica de requisitos priorizados pelo cliente, que é constantemente atualizada para refletir as necessidades do negócio. O *Scrum Master* é responsável por facilitar o processo e remover quaisquer obstáculos que possam impedir o progresso da equipe.

Esse processo iterativo e incremental promove uma adaptação contínua às mudanças e demandas do projeto, resultando em uma entrega mais rápida e eficiente do produto final, além de estimular a colaboração e a transparência dentro da equipe.

2.4 Desenvolvimento multiplataforma

O desenvolvimento multiplataforma é uma abordagem para a criação de software que permite que o mesmo código seja executado em diferentes sistemas operacionais e dispositivos. Isso significa que o código escrito pode ser compilado para diferentes plataformas, por exemplo, mobile, desktop e web.

De acordo com Gromov e Chernyshev (2021), na indústria de desenvolvimento de software a programação multiplataforma está se tornando cada vez mais popular. Isso ocorre porque ela permite a reutilização do código-fonte escrito em aplicativos compilados para diferentes sistemas operacionais, o que pode acelerar o desenvolvimento do produto e reduzir os custos de produção.

2.4.1 Kotlin multiplataforma

Kotlin é uma linguagem de programação moderna e concisa desenvolvida pela JetBrains⁶. Desde o Google I/O 2019⁷, é a linguagem de programação oficial para o desenvolvimento de aplicativos móveis no sistema operacional Android⁸. Kotlin fornece recursos que tornam os programas escritos 20% mais estáveis, além de ser totalmente interoperável com o Java (Google, Android's Kotlin-first approach). Isso significa que programas escritos em Kotlin são compilados para o mesmo *bytecode* que o Java, permitindo que projetos existentes em Java utilizem Kotlin. Assim, códigos em ambas as linguagens podem ser executados na mesma JVM (*Java Virtual Machine*).

⁶ JetBrains é uma empresa tcheca de desenvolvimento de software de capital fechado que produz ferramentas para desenvolvedores de software e gerenciamento de projetos.

⁷ O Google I/O é um evento de tecnologia realizado anualmente pelo Google, o evento demonstra diversas novidades tecnológicas que a gigante da tecnologia tem trabalhado nos últimos tempos.

⁸ Android é um sistema operacional (SO) baseado no núcleo Linux, projetado principalmente para dispositivos eletrônicos móveis (como smartphones e tablets) com tela sensível ao toque ou interface de usuário baseada na manipulação direta.;

Nos últimos anos, a JetBrains tem trabalhado em uma nova abordagem para o desenvolvimento multiplataforma com Kotlin, conhecida como KMP (Kotlin Multiplataforma)⁹. Um diferencial é que essa tecnologia não segue totalmente o princípio *"Write once, run everywhere"*, dessa forma o código que é compartilhado entre as plataformas fica em um módulo específico conhecido como *common module*, e o código que depende da plataforma permanece no módulo da plataforma (Gromov; Chernyshev, 2021, p. 1).

Figura 1 - Opções de compartilhamento de código no KMP

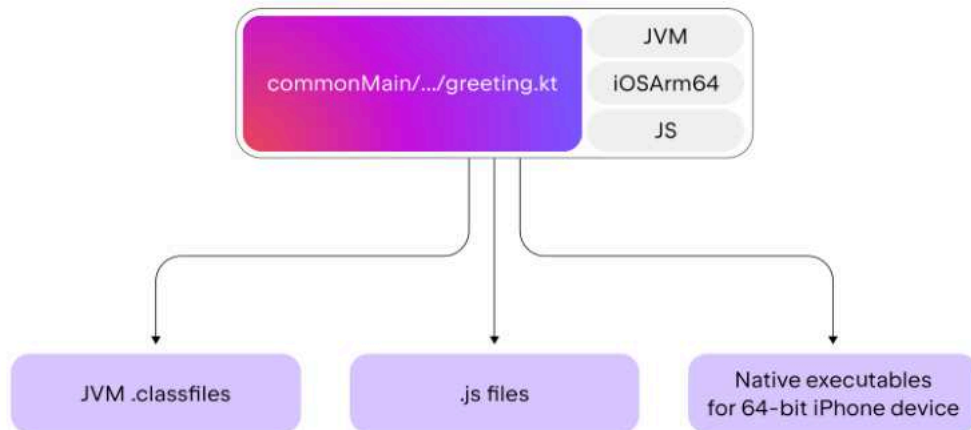


Fonte: JetBrains (2023).

Tal abordagem oferece flexibilidade, como demonstrado na Figura 1, permitindo escolher entre compartilhar partes específicas da aplicação e manter a UI nativa para cada plataforma, ou compartilhar toda a interface gráfica (UI) e lógica, ou uma variação entre essas duas opções. Isso torna a adoção do KMP em projetos existentes mais fluida, permitindo a convivência do código existente com o uso do código multiplataforma. Toda essa flexibilidade é alcançada por meio dos compiladores da linguagem: *Kotlin/JVM*, *Kotlin/Native* e *Kotlin/JS* como apresentado na Figura 2.

⁹ Em primeiro de novembro de 2023 o Kotlin Multiplataforma foi anunciado oficialmente como estável para uso em produção. <https://blog.jetbrains.com/kotlin/2023/11/kotlin-multiplatform-stable/>

Figura 2 - Plataformas de destino



Fonte: JetBrains (2024).

Dellaluce (2021) enfatiza que esta arquitetura é suportada pelo recurso *expected/actual* introduzido na linguagem. Por meio desse recurso, o código no módulo comum (*common module*) define as funcionalidades que espera dos módulos externos usando o comando *expected*. Na lógica de negócios, essas funcionalidades são utilizadas conforme definidas. Cada módulo externo implementa essas funcionalidades usando o comando *actual*. O compilador então verifica se todos os módulos externos implementam corretamente as funcionalidades esperadas. Essencialmente, o código comum assume que uma implementação real será fornecida. Assim, o KMP garante que o código específico de cada plataforma seja devidamente implementado, fornecendo as implementações reais (*actual*) necessárias para cada plataforma.

2.5 Comunicação assíncrona e microserviços vs monólitos

No desenvolvimento de software, a comunicação entre componentes é primordial para a funcionalidade e eficiência do sistema. Em arquiteturas monolíticas, os componentes costumam se comunicar de maneira síncrona, no qual uma operação aguarda a resposta de outra para continuar, criando uma interdependência que pode levar a gargalos e dificuldades na escalabilidade.

Em contraste, a arquitetura de microsserviços adota frequentemente a comunicação assíncrona, no qual serviços independentes se comunicam mediante mensagens ou filas, permitindo que cada serviço opere de maneira autônoma e responsiva. Esta abordagem não apenas melhora a escalabilidade e resiliência do sistema, mas também facilita a manutenção e a implementação de novas funcionalidades, promovendo uma estrutura mais flexível e adaptável às mudanças rápidas do ambiente de negócios.

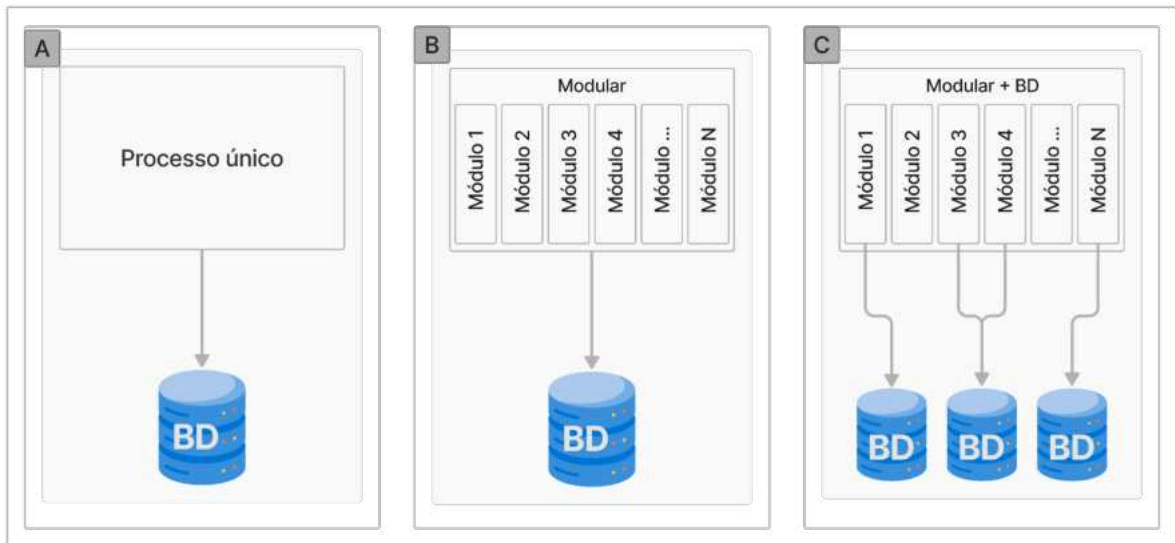
2.5.1 Monólitos

Quando falamos em monólitos, nos referimos a sistemas cuja arquitetura é projetada para que toda a lógica de negócios esteja contida em uma única aplicação, resultando em um único fluxo de implantação do sistema. Isso implica que todas as funcionalidades são desenvolvidas, testadas e implantadas como uma unidade coesa, facilitando o gerenciamento e a coordenação do desenvolvimento, especialmente em sistemas menos complexos.

Além disso, dentro do conceito de monólitos, podemos identificar três tipos principais: monólitos de processo único, monólitos modulares e monólitos modulares com segregação de banco de dados. Segundo Storone (2023), eles se caracterizam da seguinte forma:

- Monólitos de processo único: Toda a aplicação é executada como um único processo, no qual todas as funcionalidades estão intimamente acopladas, dificultando a escalabilidade e a manutenção (Figura 3A).
- Monólitos modulares: A aplicação é organizada em módulos distintos, que dividem as funcionalidades internamente, mas ainda são implantados como um único artefato, permitindo uma melhor organização do código (Figura 3B).
- Monólitos modulares com segregação de banco de dados: Além da modularização interna, cada módulo pode ter sua própria base de dados, promovendo uma separação de dados que facilita a migração para uma arquitetura de microsserviços no futuro (Figura 3C).

Figura 3 - Principais tipos de monólitos



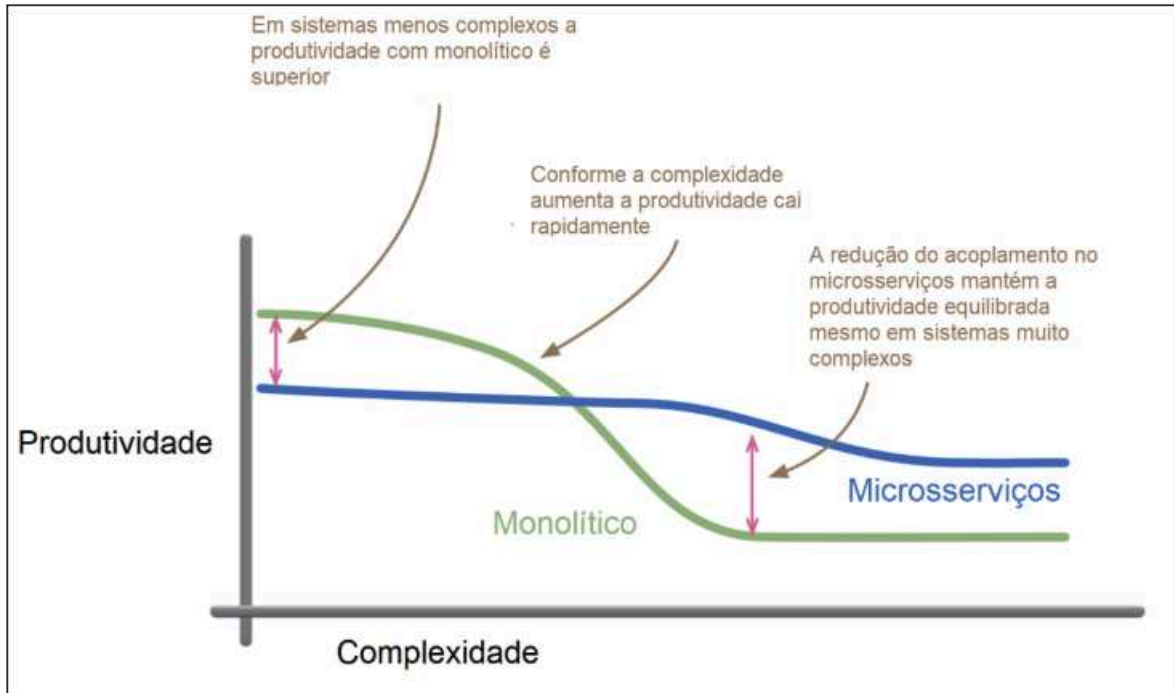
Fonte: Adaptado de Storone (2023).

Uma vantagem significativa de uma arquitetura monolítica é sua simplicidade. Segundo Fowler (2015), a natureza coesa de um monolito facilita o desenvolvimento, o teste e a implantação do software, pois todas as partes do sistema estão em um único código base, reduzindo a complexidade de gerenciamento.

Ao adotar uma arquitetura monolítica no desenvolvimento de sistemas, é importante tomar certos cuidados, especialmente à medida que o sistema cresce em complexidade e número de usuários. A centralização de todas as funcionalidades em um único projeto pode dificultar a escalabilidade e a manutenção conforme argumenta Souza (2023, p.36).

Podemos observar na Figura 4 que à medida que a complexidade de uma arquitetura monolítica aumenta, a produtividade no desenvolvimento e na manutenção tende a diminuir. Isso ocorre porque a centralização de todas as funcionalidades em uma única aplicação pode levar a dificuldades na gestão do código e na implementação de novas funcionalidades.

Figura 4 - Produtividade versus complexidade entre monólitos e microsserviços.



Fonte: Souza (2023, p. 36).

No entanto, quando a complexidade do sistema é baixa, a arquitetura monolítica apresenta uma vantagem significativa em termos de produtividade em comparação com a arquitetura de microsserviços. Isso se deve à sua simplicidade inerente, que facilita o desenvolvimento inicial e a implantação, sem a necessidade de gerenciar a comunicação entre serviços independentes. Portanto, para sistemas menos complexos, a escolha de uma arquitetura monolítica pode ser mais eficiente e produtiva.

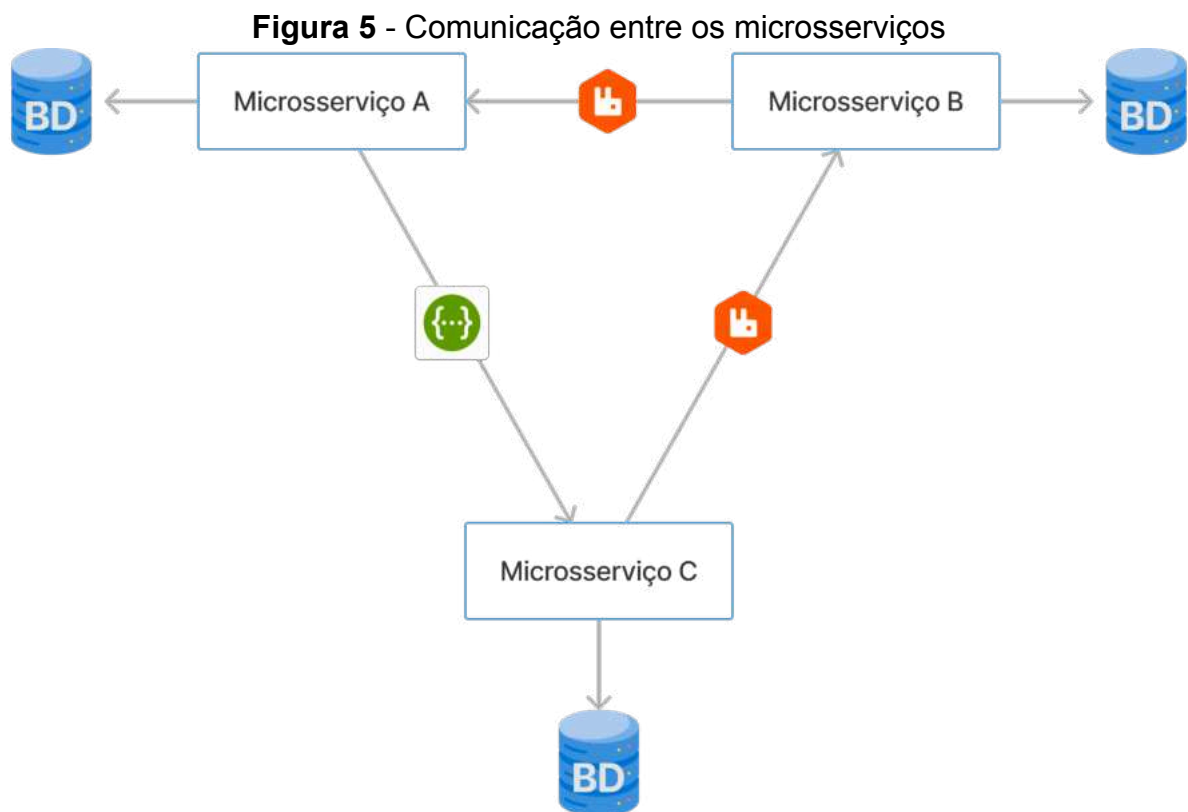
2.5.2 Microsserviços

A arquitetura de microsserviços surgiu como uma alternativa moderna aos monólitos, permitindo que aplicações complexas sejam divididas em pequenos serviços independentes. Storone (2023) argumenta que essa divisão é geralmente baseada no domínio de negócio e nos contextos delimitados, o que traz benefícios como a facilidade de escalabilidade, manutenção e implementação contínua.

Souza (2023) destaca que uma outra característica que diferencia as duas abordagens é a dependência de bancos de dados. Enquanto os monólitos utilizam um único tipo de banco de dados, as arquiteturas distribuídas permitem que cada

serviço utilize o banco de dados mais adequado ao seu contexto. Isso facilita a adoção de soluções como bancos de dados não relacionais, como o NoSQL.

Entretanto, Fowler (2015) ressalta que a adoção de microsserviços traz consigo uma maior complexidade. Optar por uma arquitetura distribuída requer atenção especial à implementação contínua, monitoramento e tratamento de falhas, além de garantir que as partes do sistema possam continuar evoluindo de forma independente.



Fonte: Autoria Própria (2024).

Na Figura 5, é possível observar um exemplo genérico de comunicação entre microsserviços, que pode ocorrer de forma síncrona, representada pelo ícone em verde, ou de forma assíncrona, representada pelo ícone em laranja. Cada microsserviço opera de maneira independente, o que contribui para uma maior resiliência do sistema. No entanto, essa independência também adiciona mais complexidade ao processo, especialmente em relação aos testes de ponta a ponta (E2E), que validam todo o fluxo da solução. Comparados com uma arquitetura monolítica, esses testes são mais complexos de serem realizados.

2.5.3 Comunicação assíncrona

A comunicação assíncrona desempenha um papel importante na arquitetura de microsserviços. Ao permitir que os serviços troquem informações sem depender de respostas imediatas, essa comunicação aumenta a resiliência e a escalabilidade do sistema. Em um ambiente assíncrono, os microsserviços enviam mensagens para filas ou tópicos, em que outros serviços podem processá-las conforme sua disponibilidade.

O desacoplamento entre produtores e consumidores permite que os produtores enviem mensagens sem conhecer as aplicações consumidoras, interagindo apenas com o serviço responsável pela fila. Essa abordagem assíncrona permite que a produção de mensagens ocorra independentemente da velocidade de consumo, garantindo que os produtores possam continuar enviando mensagens mesmo que nenhum consumidor esteja conectado (Magalhães, 2023, p.17).

Esse padrão é conhecido como *Publish/Subscribe*, no qual um serviço publica mensagens e outros se inscrevem para recebê-las. Isso desacopla os serviços, permitindo que cada um funcione e seja atualizado de forma independente, sem interromper os demais. Além disso, essa abordagem melhora a tolerância a falhas, já que a falha de um serviço não impede o funcionamento dos outros. No entanto, implementar comunicações assíncronas requer um planejamento cuidadoso para garantir a consistência dos dados e a correta orquestração entre os serviços.

Eugster *et al.* (2003) destacam que o modelo *Publish/Subscribe* possui três níveis de desacoplamento: temporal, espacial e de sincronização. O desacoplamento temporal significa que produtores (publicadores) e consumidores (assinantes) não precisam estar ativos simultaneamente para que a comunicação ocorra. As mensagens são enviadas para uma fila ou tópico, que ficam armazenadas até que os consumidores estejam prontos para processá-las. Essa abordagem permite uma comunicação assíncrona eficiente, aumentando a resiliência e flexibilidade do sistema.

Além do desacoplamento temporal, o modelo *Publish/Subscribe* também facilita o desacoplamento espacial. Produtores e consumidores não precisam conhecer as localizações uns dos outros. Os produtores publicam mensagens em filas ou tópicos centralizados, e os consumidores se inscrevem nesses pontos para receber as mensagens. Essa separação elimina a necessidade de conhecimento

mútuo entre os serviços, simplificando a escalabilidade e manutenção da arquitetura, e permitindo que os serviços evoluam de forma independente.

Por fim, o desacoplamento de sincronização é outro aspecto importante que o autor destaca. Aqui, os produtores e consumidores operam de maneira independente em relação ao ritmo de produção e consumo de mensagens. Os produtores podem continuar enviando mensagens sem esperar que os consumidores as processem imediatamente. Isso garante que a produção de mensagens não seja limitada pela velocidade de consumo, permitindo que os serviços funcionem de forma mais eficiente e sem bloqueios, melhorando a capacidade do sistema de lidar com variações na carga de trabalho.

2.6 API gateway

Em uma arquitetura de microsserviços, temos um conjunto de pequenos serviços independentes que se comunicam de forma assíncrona ou por intermédio de APIs leves. Essa abordagem proporciona flexibilidade, escalabilidade e rapidez no desenvolvimento do sistema. No entanto, quando consideramos os clientes que consomem esses serviços, especialmente os clientes front-end externos, surgem alguns problemas.

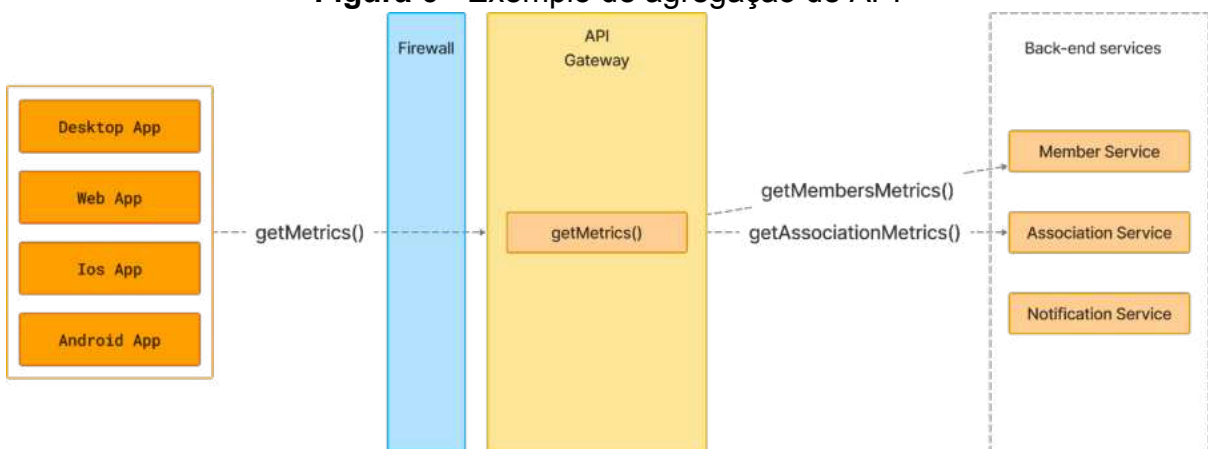
Segundo Silva (2023) alguns dos principais desafios incluem:

- Excesso de Requisições: O grande número de requisições pode impactar negativamente a experiência do usuário, tornando o sistema mais lento.
- Falta de Encapsulamento: Como os clientes conhecem diretamente as APIs dos microsserviços, mudanças internas no sistema se tornam mais difíceis e arriscadas.
- Diferenças nos Protocolos de Comunicação: Os protocolos usados internamente pelos microsserviços podem não ser os mesmos que os esperados pelos clientes, criando incompatibilidades.
- Variedade de banda larga: A largura de banda e a capacidade de processamento variam entre diferentes tipos de dispositivos, dificultando a otimização das respostas de acordo com o dispositivo utilizado.

Para gerenciar a complexidade da comunicação entre uma multiplicidade de microsserviços, surge o conceito de *API Gateway*. Um *API Gateway* atua como um ponto único de entrada para todas as requisições que chegam aos microsserviços gerenciando as requisições, seja roteando-as diretamente para o serviço apropriado ou distribuindo-as entre vários serviços. Além disso, em vez de oferecer um único estilo de API, o *API Gateway* pode expor diferentes APIs personalizadas para cada tipo de cliente (Silva, 2023, p. 72).

Alguns benefícios do uso de um *API Gateway* incluem simplificação do roteamento de requisições, centralizando a gestão de tráfego e encaminhando as requisições para os serviços apropriados. Além disso, melhora a segurança ao gerenciar autenticação, autorização e proteção contra ataques. Com a capacidade de agregar dados de múltiplos serviços em uma única resposta, o *API Gateway* reduz a complexidade para os clientes e facilita o monitoramento e a depuração através de um registro de *log* (histórico de transações) centralizado.

Figura 6 - Exemplo de agregação de API



Fonte: Autoria Própria (2024).

Como exemplificado na Figura 6, quando o front-end faz uma requisição *getMetrics()*, o *API Gateway* processa essa solicitação chamando dois serviços distintos: *member* e *association*. Em vez de o cliente realizar múltiplas requisições e consolidar as respostas, o *API Gateway* realiza essas operações internamente e retorna uma resposta unificada. Isso não apenas melhora a eficiência e a velocidade da comunicação, mas também reduz a complexidade e a carga de processamento no lado do cliente.

Apesar de suas vantagens, um *API Gateway* também apresenta desvantagens. Ele pode se tornar um ponto único de falha, pois todas as requisições passam por ele, e se ficar indisponível, pode afetar todo o sistema. Além disso, a implementação e manutenção de um *API Gateway* adiciona uma camada extra de complexidade à arquitetura, exigindo habilidades especializadas e mais esforço de desenvolvimento. Outro problema é a latência adicional, já que o *Gateway* introduz uma etapa extra no processamento das requisições, o que pode impactar o desempenho geral do sistema. No entanto, essa latência geralmente não é um problema significativo para a maioria dos casos de uso em aplicações.

3 METODOLOGIA

Esta pesquisa é de natureza aplicada, especialmente direcionada para a geração de conhecimentos com aplicabilidade prática, visando resolver problemas específicos, conforme explicado por Prodanov e Freitas (2013). No contexto deste estudo, o foco está na integração de soluções tecnológicas de gestão na produção agrícola, com atenção especial voltada para pequenos e médios produtores. Os principais públicos interessados na aplicação dessas soluções incluem produtores rurais, associações e cooperativas ligadas ao setor agrícola, bem como prefeituras e governos estaduais.

Em relação aos objetivos, esta pesquisa adota uma abordagem tanto exploratória quanto descritiva. A pesquisa exploratória busca fornecer informações adicionais sobre o tema em investigação, permitindo sua definição e delineamento (Prodanov; Freitas, 2013, p.51). Por sua vez, a pesquisa descritiva tem como objetivo principal descrever as características de uma determinada população ou fenômeno (Prodanov; Freitas, 2013, p.52). Neste contexto, é necessário explorar e descrever o objeto de estudo, que envolve a modelagem de um sistema de gestão integrado multiplataforma. Por meio desta abordagem, busca-se impactar positivamente na gestão eficiente dos recursos e no aumento da produtividade e eficiência operacional dos agricultores.

Em relação aos procedimentos adotados, esta pesquisa se concentra principalmente em fontes bibliográficas e documentais. Optamos por tal abordagem devido à necessidade de explorar tópicos que ainda não são amplamente conhecidos e que envolvem a interseção entre gestão, tecnologias novas e disruptivas, e o contexto específico da produção agrícola. Dessa maneira, buscamos garantir uma base científica sólida para o trabalho, abordando de forma abrangente e fundamentada os elementos que influenciam esse campo.

Ao final da fundamentação teórica sobre o tema, foi elaborada uma proposta de arquitetura para a aplicação com a modelagem inicial do software. Esse processo envolveu o levantamento dos requisitos identificados durante a pesquisa, além da construção dos diagramas de caso de uso e diagramas de classe. Iniciamos o protótipo inicial do sistema com a criação de seu padrão visual. Em conclusão, a metodologia adotada até agora permitiu uma compreensão mais elaborada dos

requisitos iniciais da solução, um delineamento claro da arquitetura e dos componentes principais do sistema e o desenvolvimento de protótipo das telas.

4 RESULTADOS

O AgrarConnect visa auxiliar na gestão das atividades agrícolas para pequenos e médios agricultores e associações agrícolas, otimizando processos e aumentando a produtividade, de forma que cada módulo passa ter integração de funcionalidades entre si, visando assim uma experiência de uso mais unificada do sistema. No que tange ao objetivo deste trabalho, o software será composto por dois módulos principais: financeiro e o de associações e cooperativas.

4.1 Definição do software

O módulo financeiro é o responsável por lidar com a gestão de entrada e saída de caixa, o que envolve gerenciar o fluxo de caixa, incluindo lançamento de receitas e despesas, despesas recorrentes, controle de contas bancárias e geração de relatórios financeiros. De modo que essas funcionalidades sejam expostas para que tanto um produtor rural independente como uma associação de agricultores possam utilizar as funcionalidades disponibilizadas pelo módulo.

O módulo de associações e cooperativas é o responsável pelo gerenciamento de atividades desse tipo de organizações, gerenciando o cadastro de membros, incluindo informações pessoais, contato, propriedades, cotas e participação em atividades. Esse módulo permite a gestão de cotas, incluindo pagamento, atualização de valores e histórico de transações, além de organizar a realização de assembleias, com votação eletrônica e registro de atas e decisões.

4.1.1 Arquitetura do software

No contexto da crescente demanda por aplicações multiplataforma, este projeto propõe a utilização do KMP como solução para otimizar o desenvolvimento e a manutenção do código das aplicações (clientes). O KMP permite escrever um único código em Kotlin e compilá-lo para diferentes plataformas, como Android, iOS e Web, centralizando o código comum em um módulo e reutilizando-o nos módulos específicos de cada plataforma.

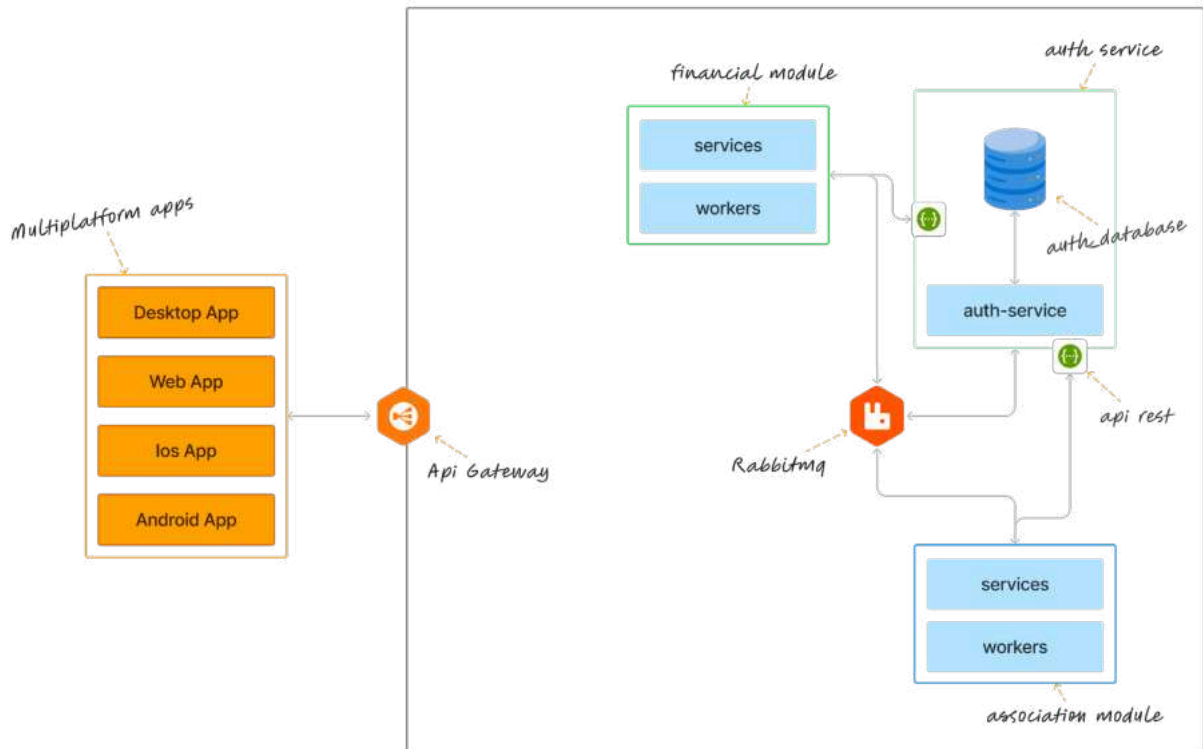
Para o back-end adotaremos o Kotlin com Spring Boot em uma arquitetura de microsserviços. Como destaca Kurniawan *et al.* (2024, p. 4, tradução nossa), "um microsserviço deve ser projetado para executar bem uma única tarefa ou

funcionalidade em uma escala menor em comparação com um aplicativo monolítico". Essa abordagem traz como principal benefício a capacidade de ser ter partes menores em que a junção delas compõem um sistema e que essas partes menores sejam facilmente gerenciadas, atualizadas e melhoradas de forma independente.

4.1.1.1 Proposta da arquitetura

O diagrama de alto nível apresentado na Figura 7 ilustra a arquitetura geral do sistema, que é composto por quatro módulos principais. À esquerda, em laranja, temos o módulo responsável pela criação dos aplicativos multiplataforma. No canto superior direito, em verde claro, está o módulo responsável pela gestão da autenticação e autorização multimódulo dos usuários. O terceiro módulo, localizado no centro, é responsável pelas funcionalidades financeiras. Por fim, no canto inferior direito, em azul, encontra-se o módulo dedicado às regras das associações e cooperativas.

Figura 7 - Visão geral da arquitetura
agrar connect back-end system



Fonte: Autoria Própria (2024).

Os módulos financeiro e de associações são compostos por *workers* e *services*. Um "*worker*" é um componente que executa tarefas assíncronas ou de segundo plano, como processamento de transações ou envio de notificações, garantindo que operações intensivas não afetem o desempenho geral do sistema. Já um "*service*" é responsável por fornecer funcionalidades específicas de negócios, como cálculos financeiros, gestão de membros ou geração de relatórios, disponibilizando interfaces e operações que podem ser usadas por outros componentes do sistema. A combinação de *workers* e *services* nesses módulos permite uma arquitetura escalável e eficiente, no qual tarefas complexas são divididas e gerenciadas de forma otimizada.

A comunicação interna entre os *workes* e *services* pode ocorrer de forma assíncrona ou síncrona. Na comunicação assíncrona, é utilizado *RabbitMQ*¹⁰, um sistema de mensageria que permite que os módulos enviem e recebam mensagens de forma desacoplada e eficiente, facilitando a execução de tarefas em background e a coordenação de processos complexos.

Por outro lado, a comunicação síncrona é realizada por meio de *APIs REST*, no qual os módulos interagem diretamente, enviando e recebendo requisições *HTTP* em tempo real. Esta abordagem é adequada para operações que requerem respostas imediatas, como consultas de dados ou validações instantâneas. A flexibilidade em utilizar ambas as formas de comunicação permite que o sistema seja tanto responsivo quanto robusto, adaptando-se às necessidades específicas de cada operação.

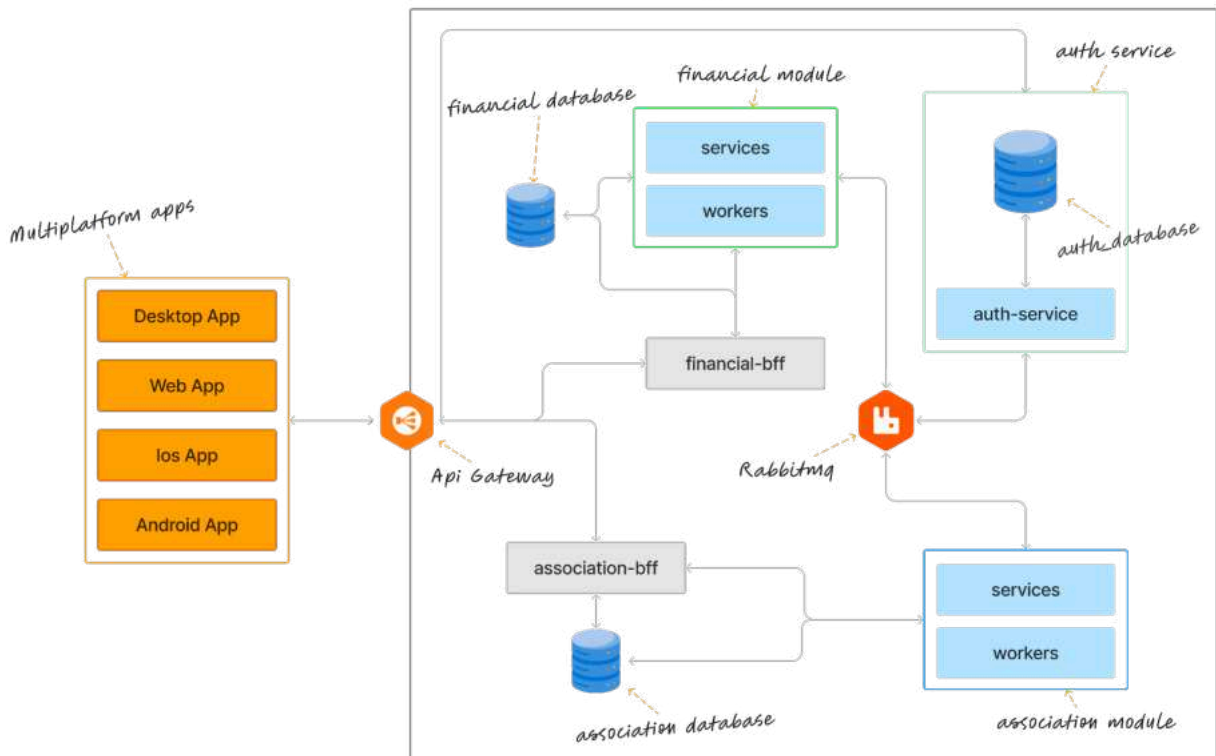
Entretanto, ao considerarmos a comunicação dos clientes externos com o backend ela ocorre através de um *API Gateway*, atuando como um único ponto de entrada para o sistema, dessa forma abstraindo a complexidade de comunicação com os serviços back-end além de fornecer recursos tais como: balanceamento de carga, métricas, logs e cache.

Conforme é possível observar na Figura 8, será utilizada uma camada intermediária na comunicação do front-end com o back-end, conhecida como BFF (*Back-End for Front-End*). O BFF atua como uma camada intermediária que serve para otimizar e personalizar a comunicação entre o front-end e os diversos serviços

¹⁰ O RabbitMQ é um software de mensagens com código aberto, que implementou o protocolo "Advanced Message Queuing Protocol" (AMQP), que foi estendido com uma arquitetura de plug-in para suportar o protocolo "Streaming Text Oriented Messaging Protocol" (STOMP), o MQTT entre outros protocolos.

no *back-end*. Esta abordagem permite que o BFF agregue dados de múltiplos serviços, aplique lógica específica de negócios para diferentes tipos de clientes (por exemplo, web, mobile), e entregue respostas mais eficientes e adaptadas às necessidades do front-end. Usando o BFF, podemos minimizar a quantidade de chamadas diretas do front-end aos serviços *back-end*, reduzindo a complexidade no cliente e melhorando a performance geral do sistema.

Figura 8 - Visão geral da arquitetura, usando API Gateway e bff agrar connect back-end system



Fonte: Autoria Própria (2024).

Dessa forma o *API Gateway* fica responsável por ser um ponto de entrada único para todas as requisições, proporcionando roteamento, balanceamento de carga, cache e outras políticas de segurança e redireciona as requisições do front-end para o BFF apropriado, que então processa e agrega as respostas dos diversos serviços back-end. Esta combinação permite uma arquitetura mais modular e escalável, em que o *API Gateway* cuida da infraestrutura e segurança, enquanto o *BFF* é responsável por agregar valor ao front-end com lógica específica e otimização de dados.

4.1.2 Funcionalidades identificadas para o sistema

Para atender às necessidades dos usuários do AgrarConnect, o foco estará na implementação de funcionalidades essenciais e na integração perfeita entre elas, proporcionando uma experiência de uso mais eficiente e intuitiva.

4.1.2.1 Módulo financeiro

Conforme Lopes (2019) destaca em sua pesquisa, as principais funcionalidades utilizadas para gestão financeira por pequenas e médias empresas envolvem: capital de giro, fluxo de caixa, controle de contas a pagar e receber, controle de estoque, balanço patrimonial e demonstração de resultado e exercício.

Com base nos requisitos apresentados e considerando os aspectos da problematização apresentada no trabalho, foram elaborados os requisitos funcionais detalhados no Quadro 2 e os não funcionais, que estão no Quadro 3.

Quadro 2 - Módulo Financeiro: requisitos funcionais

(continua)

NÚMERO DO REQUISITO	REQUISITO FUNCIONAL	DESCRIÇÃO
RF001	Autenticação do sistema	O sistema deve fornecer um mecanismo de autenticação robusto para garantir que apenas usuários autorizados possam acessar o sistema e suas funcionalidades. A autenticação deve ser baseada em um sistema de login com nome de usuário e senha, complementado por autenticação multifator (MFA) para perfis administrativos e outras funções críticas.
RF002	Gestão de níveis de acesso	O sistema deve permitir a criação, atribuição e gerenciamento de perfis de acesso que determinem as permissões dos usuários em relação às funcionalidades dos módulos do sistema (Financeiro e Associações e Cooperativas). Além disso, deve haver perfis de acesso gerais aplicáveis a todo o sistema.
RF003	Cadastro de contas a pagar e receber	O sistema deve permitir o cadastro, consulta, edição e exclusão de contas a pagar e a receber, garantindo um gerenciamento eficaz das obrigações financeiras da organização. As contas devem ser categorizadas, com detalhes sobre vencimento, valor, descrição, e status de pagamento. Além disso, o sistema deve suportar o cadastro de despesas recorrentes.

(conclusão)

NÚMERO DO REQUISITO	REQUISITO FUNCIONAL	DESCRIÇÃO
RF004	Fluxo de caixa	O sistema deve permitir a geração de um relatório de fluxo de caixa futuro, que possibilite a visualização projetada das receitas e despesas recorrentes para um período de tempo especificado pelo usuário. Este relatório deve calcular o total de receitas esperadas, as despesas previstas e a diferença entre esses valores, fornecendo uma visão clara da saúde financeira futura
RF005	Importação de dados	O sistema deve permitir a importação massiva de dados de despesas e receitas a partir de arquivos CSV. Esta funcionalidade deve facilitar a inserção rápida e eficiente de grandes volumes de dados, garantindo a integridade e consistência das informações importadas.
RF006	Relatório de balanço patrimonial	Permite uma visão clara da situação financeira e patrimonial do agricultor, auxiliando na avaliação do desempenho e na identificação de oportunidades de melhoria. Depende do requisito R003.

Fonte: Autoria Própria (2024).

Quadro 3 - Módulo Financeiro: requisitos não funcionais

REQUISITO NÃO FUNCIONAL	DESCRIÇÃO
Consistência visual	Garantir que a interface siga um padrão coerente de design como: cores, tipografia, espaçamento e layout. Proporcionando uma experiência visual harmoniosa e reconhecível para os usuários em toda a aplicação.
Facilidade de utilização	A aplicação deve ser intuitiva e fácil de usar para usuários de todos os níveis de habilidade. Contém feedback claros e concisos.
Funcionalidades offline	O sistema precisa ter funcionalidades que funcionem sem internet.
Multiplataforma	A capacidade de funcionar em diferentes plataformas é uma característica desejável do sistema.
Níveis de acesso	Implementar diferentes níveis de permissão de acesso para usuários, garantindo que apenas aqueles autorizados possam acessar determinadas funcionalidades ou informações da aplicação
Responsividade	O sistema precisa ser adaptar a diferentes tamanhos de telas
Segurança das informações	Proteção dos dados sensíveis e confidenciais armazenados ou transmitidos pela aplicação contra acessos não autorizados, modificações indevidas ou vazamentos.
Sincronização de dados	O sistema deve sincronizar dados automaticamente quando a conexão com a internet for estabelecida.

Fonte: Autoria Própria (2024).

Dessa forma, torna-se possível abordar funcionalidades que são primordiais para o sistema, adicionando alguns requisitos que caracterizam o diferencial da solução proposta para o desenvolvimento do AgrarConnect como a capacidade de ter funcionalidades desconectadas da Internet. Com efeito, quando o acesso à Internet estiver disponível, haverá a sincronização dos dados. Além disso, para atender às necessidades de diferentes públicos de produtores e associações agrícolas, a intenção deste projeto é desenvolver interfaces multi-plataformas, que sejam acessíveis em diferentes dispositivos, tais como computadores, tablets e smartphones.

4.1.2.2 Módulo de associações e cooperativas

A gestão de pequenos produtores, associações e cooperativas rurais exige ferramentas que atendam às suas necessidades específicas (Figueiredo *et al.*, 2023, p. 205). A gestão eficiente dos recursos humanos, por exemplo, é fundamental para otimizar o trabalho e aumentar a produtividade. Ferramentas que facilitam a comunicação e o trabalho colaborativo também são essenciais para fortalecer o trabalho em equipe e alcançar melhores resultados tendo em vista a natureza da gestão colaborativa das cooperativas.

Quadro 4 - Módulo de Associações e Cooperativas: requisitos funcionais

(continua)

NÚMERO DO REQUISITO	REQUISITO FUNCIONAL	DESCRIÇÃO
RF001	Custo com mão de obra externa	O sistema deve permitir o cadastro, visualização e gestão dos custos com mão de obra externa. Os membros podem realizar o cadastro, mas apenas usuários com o perfil de Tesoureiro ou presidente podem aprovar o custo, todos os membros podem visualizar os custos cadastrados e pendentes.
RF002	Sistemas de votação	O sistema deve permitir a criação, gestão e visualização de votações online no módulo de associações e cooperativas. Apenas usuários com perfil de secretário ou presidente podem criar e registrar novas votações no sistema. Todos os membros da associação têm acesso para visualizar os resultados das votações.
RF003	Compartilhamento de informações	Portal centralizado com acesso a documentos, relatórios, notícias e outras informações relevantes para os membros da cooperativa.

(conclusão)

NÚMERO DO REQUISITO	REQUISITO FUNCIONAL	DESCRIÇÃO
RF004	Registro de atas	O sistema de associações e cooperativas deve permitir o registro de atas de reuniões, que são documentos oficiais e importantes para a gestão e transparência das atividades da associação. Apenas usuários com os perfis de secretário e presidente têm a permissão para registrar atas. Todos os membros da associação têm o direito de visualizar as atas registradas. O sistema deve garantir que as atas sejam armazenadas de forma segura e acessível, mantendo um histórico completo de todas as reuniões e decisões tomadas.
RF005	Cadastro de membros	O sistema deve permitir o cadastro de novos membros em associações e cooperativas. Para se tornar um membro, o indivíduo deve receber um convite caso ainda não esteja associado a nenhuma associação ou cooperativa no sistema. O convite deve ser enviado por um administrador da associação ou cooperativa e, uma vez aceito, permitirá ao novo membro acessar as funcionalidades e benefícios da associação ou cooperativa.
RF006	Alocação de cooperados em tarefas	Permitir o gerenciamento e alocação de cooperados em tarefas.
RF007	Solicitação de transferência entre associações ou cooperativas.	O sistema deve permitir que os membros possam solicitar transferência de uma associação para outra e para que essas solicitações sejam aprovadas ou rejeitadas. Os membros podem iniciar uma solicitação de transferência, que ficará pendente até a aprovação de um administrador.

Fonte: Autoria Própria (2024).

Quadro 5 - Módulo de Associações e Cooperativas: requisitos não funcionais

(continua)

REQUISITO NÃO FUNCIONAL	DESCRIÇÃO
Consistência visual	Garantir que a interface siga um padrão coerente de design como: cores, tipografia, espaçamento e layout. Proporcionando uma experiência visual harmoniosa e reconhecível para os usuários em toda a aplicação.
Facilidade de utilização	A aplicação deve ser intuitiva e fácil de usar para usuários de todos os níveis de habilidade. Contendo feedback claros e concisos.
Funcionalidades offline	O sistema precisa ter funcionalidades que funcionem sem internet.
Multiplataforma	A capacidade de funcionar em diferentes plataformas é uma característica desejável do sistema.
Níveis de acesso	Implementar diferentes níveis de permissão de acesso para usuários, garantindo que apenas aqueles autorizados possam acessar determinadas funcionalidades ou informações da aplicação

(conclusão)

REQUISITO NÃO FUNCIONAL	DESCRIÇÃO
Responsividade	O sistema precisa ser adaptar a diferentes tamanhos de telas
Segurança das informações	Proteção dos dados sensíveis e confidenciais armazenados ou transmitidos pela aplicação contra acessos não autorizados, modificações indevidas ou vazamentos.
Sincronização de dados	O sistema deve sincronizar dados automaticamente quando a conexão com a internet for estabelecida.

Fonte: Autoria Própria (2024).

O contexto da agricultura familiar exige soluções inovadoras que promovam a colaboração e o empoderamento dos agricultores na gestão e na tomada de decisões. O segundo módulo do software se propõe a atender tal demanda, oferecendo funcionalidades que facilitam a gestão de recursos humanos e estimulam a participação colaborativa nas tomadas de decisões, conforme os requisitos funcionais detalhados no Quadro 4, e não funcionais no Quadro 5.

4.2 Modelagem dos requisitos

Com base nos requisitos fundamentais apresentados, este tópico abordará a modelagem de *software* necessária para atender essas necessidades, explorando as interações entre os usuários e o sistema por meio de ações e respostas. Assim, é possível fornecer uma visão detalhada e clara das funcionalidades mapeadas, garantindo uma compreensão abrangente dos requisitos do sistema.

4.2.1 Diagrama de caso de uso

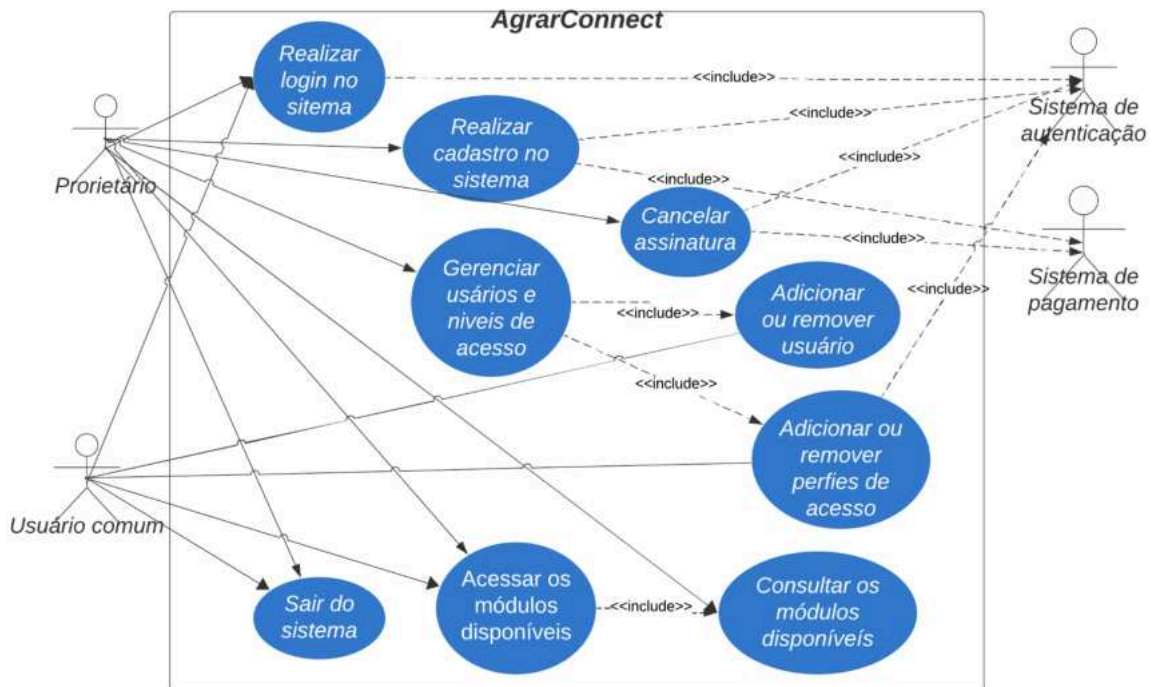
Em uma abordagem de alto nível dos diagramas, inicialmente observamos a interação primordial com o sistema. Neste contexto, identificamos dois atores principais: o proprietário, que possui acesso administrativo, e o usuário comum. O proprietário detém privilégios administrativos, enquanto o usuário comum, em sua interação inicial, tem acesso apenas à leitura de informações e funcionalidades básicas do sistema.

Devido à natureza multi modular do sistema, é necessário estabelecer dois tipos de perfis de acesso: os de contexto global e os de contexto local. Os perfis de

contexto global definem regras de acesso válidas para todos os módulos aos quais os usuários têm acesso, enquanto os perfis de contexto local são específicos para cada módulo.

Por exemplo, um usuário comum, com acesso básico ao sistema, pode ter permissão para acessar o módulo financeiro e cadastrar suas despesas pessoais. Por outro lado, um usuário que faz parte de uma associação pode cadastrar despesas para a associação, despesas essas que precisam ser aprovadas pelo responsável financeiro ou por um administrador com permissões mais amplas.

Figura 9 - Diagrama de caso de uso: visão alto nível do sistema



Fonte: Autoria Própria (2024).

Conforme destacado na Figura 9, identificamos mais dois atores essenciais: os sistemas de autenticação e de pagamento. Ambos são atores externos que são acionados toda vez que os usuários realizam ações relacionadas à autenticação, autorização ou pagamento, respectivamente.

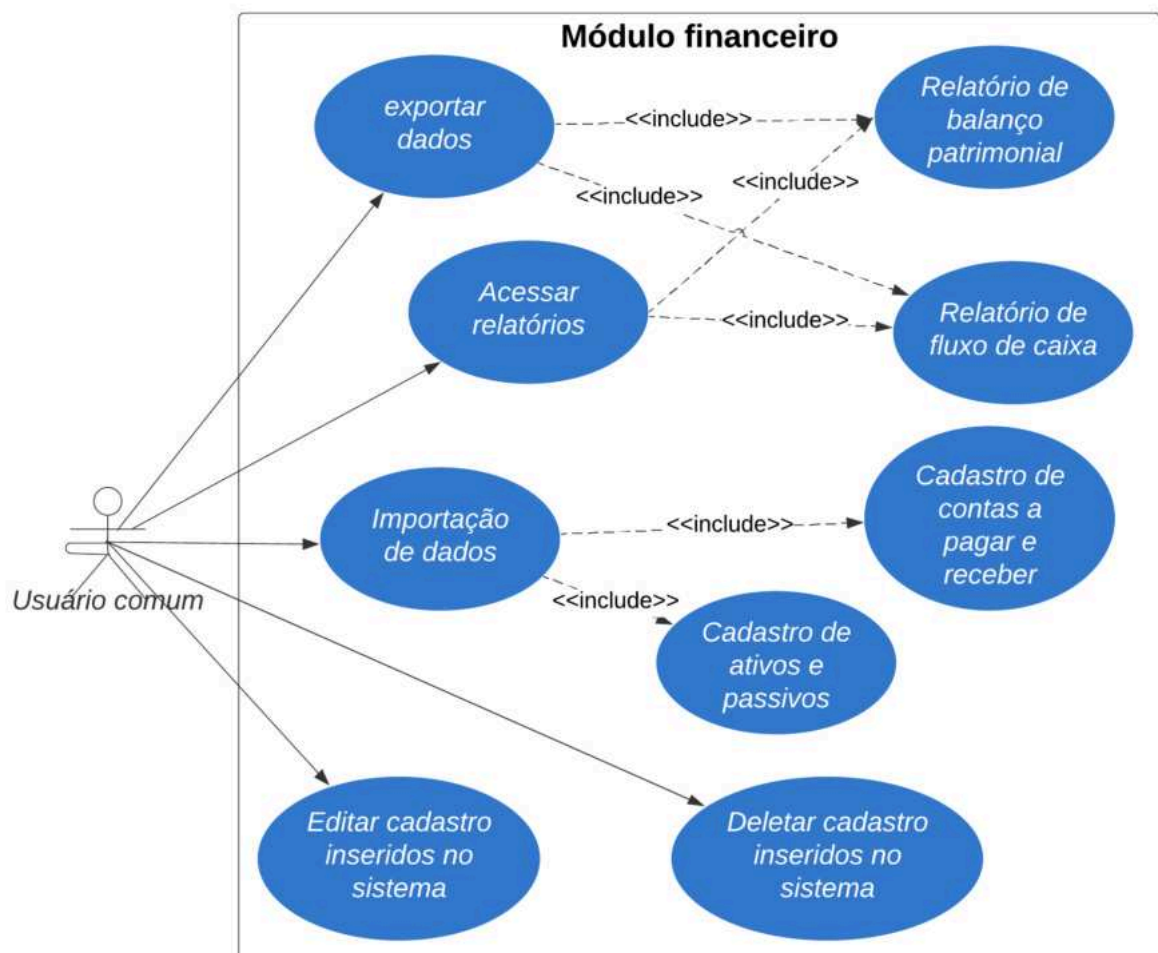
Os principais casos de uso mapeados nesta interação em alto nível incluem: realizar login no sistema, realizar cadastro no sistema, cancelar assinatura, gerenciar usuários e níveis de acesso. Além disso, há funcionalidades para consultar ou acessar os módulos disponíveis e sair do sistema. Entre essas funcionalidades, o usuário comum só pode realizar o login após ser adicionado por um administrador

ou usuário com permissões adequadas. Ele também pode consultar ou acessar os módulos disponíveis para o seu nível de acesso ou sair do sistema. As demais funcionalidades mencionadas podem ser realizadas apenas pelo administrador, conforme demonstrado na Figura 9.

4.2.1.1 Diagrama de caso de uso: Módulo financeiro

Para compreender o módulo financeiro, é necessário analisar o diagrama de casos de uso do módulo, assim como os casos de uso de integração com outros módulos. Neste trabalho, o foco será na integração entre o módulo financeiro e o módulo de associações e cooperativas.

Figura 10 - Diagrama de caso de uso: módulo financeiro



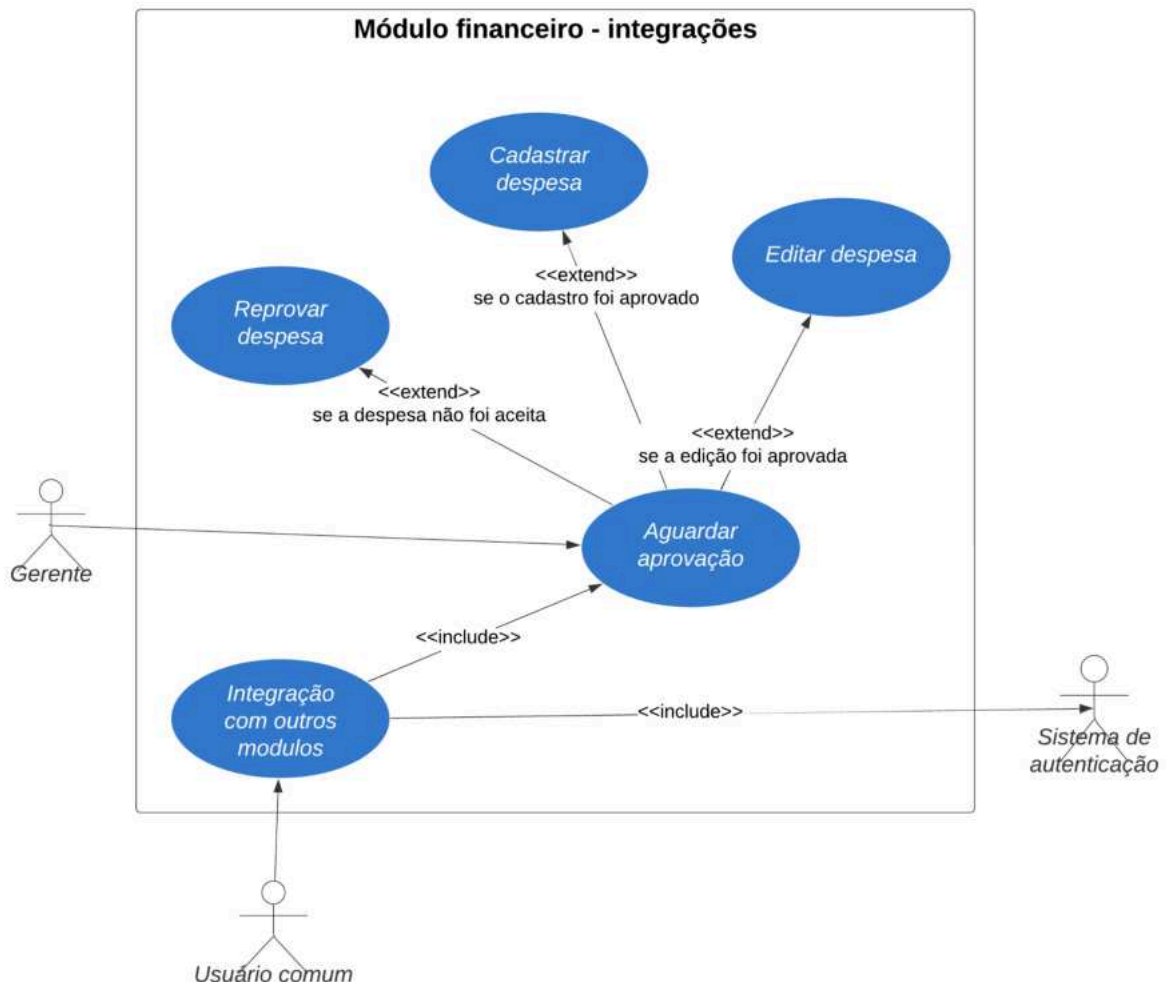
Fonte: Autoria Própria (2024).

O módulo financeiro oferece uma série de funcionalidades essenciais para o gerenciamento financeiro. Entre essas funcionalidades, conforme apresentado na

Figura 10, destacam-se o cadastro, edição ou exclusão de contas a pagar e receber. Além disso, os usuários têm a capacidade de cadastrar ativos e passivos, fornecendo uma visão completa da situação financeira.

É importante ressaltar que todo usuário com acesso ao módulo financeiro tem permissão para utilizar as funcionalidades destinadas ao controle financeiro pessoal. Essas informações são estritamente privadas e exclusivas do usuário logado, garantindo que apenas ele tenha acesso a elas. Isso significa que nenhum outro usuário tem permissão para visualizar ou manipular esses dados financeiros, assegurando a confidencialidade e a privacidade das informações pessoais de cada usuário.

Figura 11 - Diagrama de caso de uso: módulo financeiro - integrações



Fonte: Autoria Própria (2024).

Uma característica importante é a capacidade de importar e exportar dados, possibilitando a integração com outras plataformas e a geração de relatórios personalizados. Essas ferramentas permitem aos usuários manter um controle detalhado de suas finanças e facilitam o processo de tomada de decisões financeiras.

Essas funcionalidades também estão disponíveis para uso em outros módulos por meio de integrações com o módulo financeiro. O processo de utilização difere ligeiramente do contexto apresentado na Figura 10: na integração, as solicitações de outros módulos são recebidas e validadas para garantir que o usuário que realiza a ação tenha acesso ao nível de usuário comum no módulo financeiro.

É importante ressaltar que toda criação ou alteração de dados necessita de aprovação, exceto quando o próprio usuário que realiza a ação é responsável pela aprovação, caso em que a aprovação é automaticamente liberada. A Figura 11 ilustra um caso de uso simplificado que demonstra o funcionamento desse processo de integração.

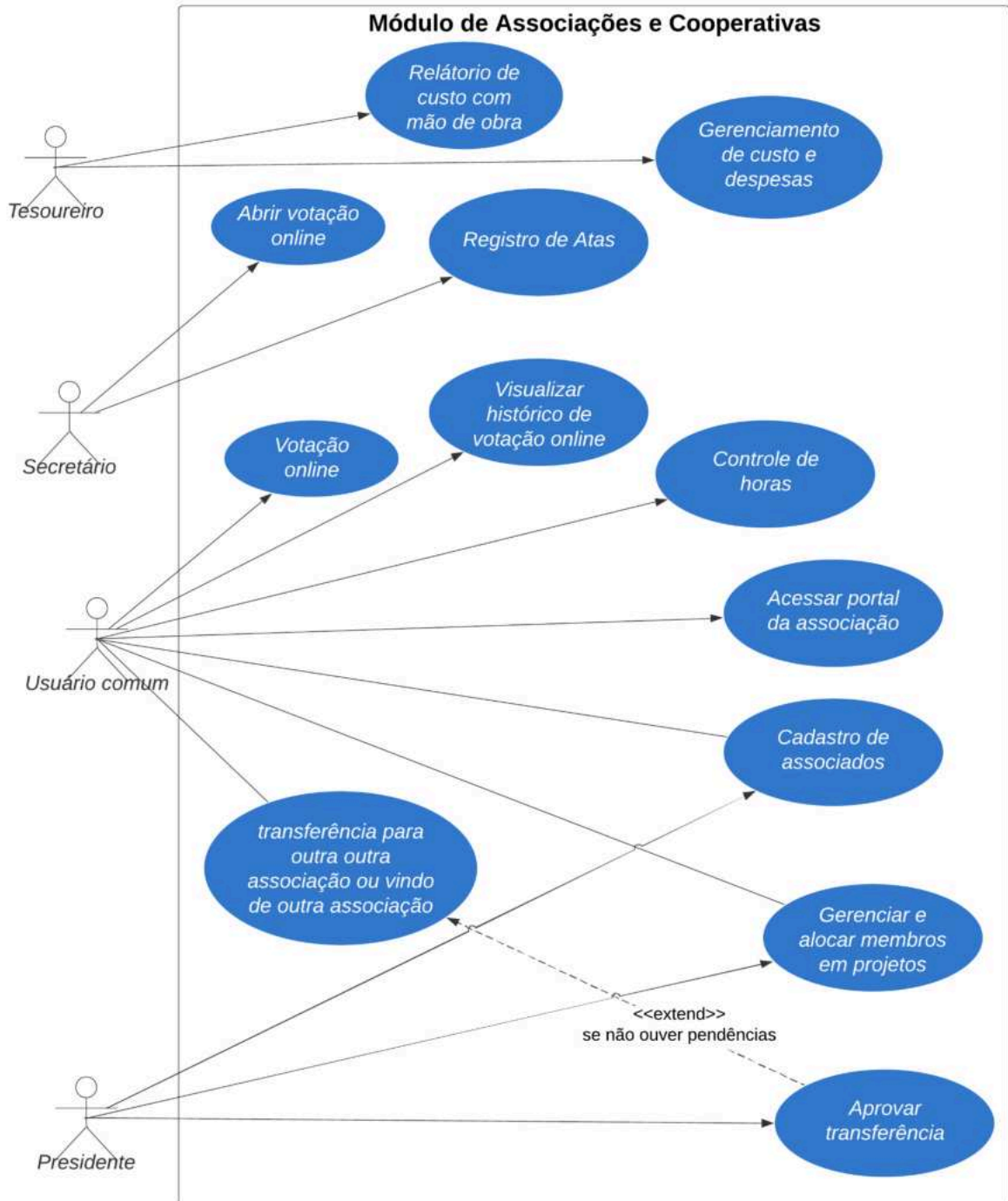
4.2.1.2 Diagrama de caso de uso: Módulo de associações e cooperativas

No módulo de associações e cooperativas, observamos uma estrutura diferente, com quatro atores principais: o presidente, secretário, tesoureiro e o usuário comum, que representa o nível de acesso básico em cada módulo. O tesoureiro é responsável por gerar relatórios de custos e gerenciar despesas, incluindo o cadastro, atualização e aprovação de despesas registradas por outros membros. O secretário é encarregado de criar enquetes de votação online e registrar atas. Já o presidente tem a responsabilidade de revisar e aprovar transferências de membros da associação, registrar novos membros e alocar membros em projetos, quando necessário.

O usuário comum possui a capacidade de realizar diversas ações, como participar de votações online e acessar o histórico de votações anteriores. Além disso, ele pode realizar o controle de horas trabalhadas, uma funcionalidade que depende da estratégia de gestão adotada por cada associação ou cooperativa. Essa funcionalidade pode ser habilitada posteriormente, de acordo com as necessidades

específicas da organização. Além disso, o usuário comum também pode acessar o portal da associação para obter informações relevantes e recursos disponíveis.

Figura 12 - Diagrama de caso de uso: módulo de Associações e Cooperativas



Fonte: Autoria Própria (2024).

O portal é um espaço centralizado para o compartilhamento de informações da comunidade. Nele, os membros podem acessar registros de atas passadas,

notícias publicadas e avisos sobre reuniões ou eventos futuros que estão programados para acontecer. Essa plataforma serve como uma fonte essencial de comunicação e colaboração para os membros da associação ou cooperativa, facilitando o acesso a informações relevantes e atualizadas sobre as atividades e decisões da comunidade.

É fundamental ressaltar que todas as ações disponíveis para o usuário comum também podem ser realizadas pelo presidente, tesoureiro e secretário (ver Figura 12). Essas funcionalidades estão disponíveis no contexto do usuário, independentemente do cargo atribuído. Dessa forma, todos os membros têm acesso às mesmas capacidades e recursos, promovendo uma participação igualitária e colaborativa na comunidade.

4.2.2 Diagrama de classes

O diagrama de classes para o sistema de autenticação e controle de acesso multi módulo foi desenvolvido para garantir uma gestão eficiente e segura dos perfis de usuários e suas permissões em diferentes módulos do sistema. No centro do diagrama está a classe *User*, que representa os usuários do sistema. Cada instância de usuário possui atributos como *id*, *email*, senha, nível de acesso, e métodos essenciais para a autenticação e consulta de permissões, como *login*, *resetPassword*, *lockAccount* (ver Figura 13).

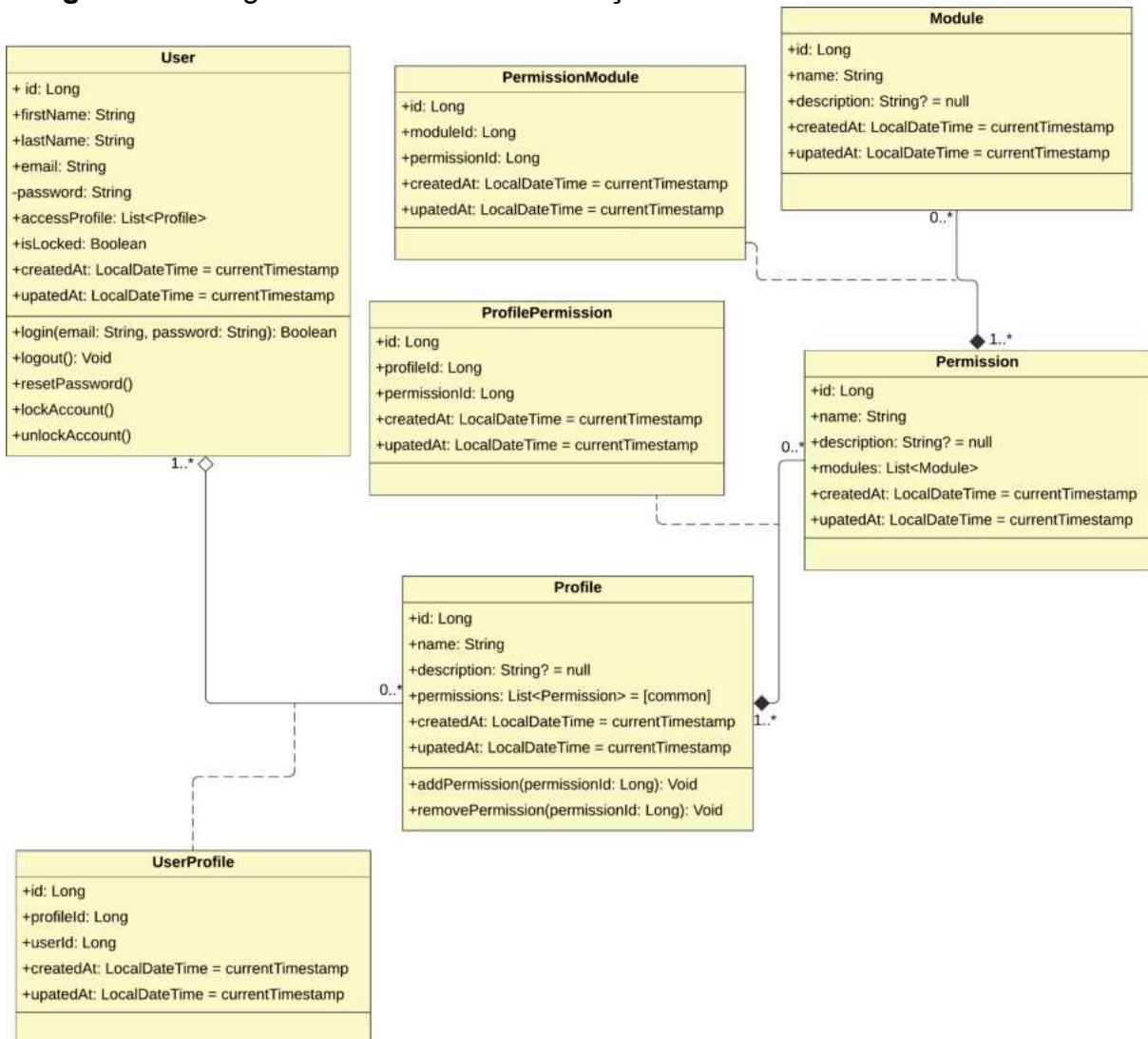
Para gerenciar os níveis de acesso, a classe *permission* é associada a *profile* e a diferentes módulos do sistema. A *Permission* define o escopo das ações que um usuário pode realizar em cada módulo, e essa relação é fundamental para garantir que apenas usuários autorizados possam acessar e manipular dados em módulos específicos, como o módulo financeiro ou o módulo de associações e cooperativas.

Além disso, o diagrama inclui a classe *Profile*, que agrupa várias permissões, facilitando a atribuição de conjuntos de permissões a usuários com funções similares. A *profile* contém atributos como *id* e nome, e métodos para gerenciar as permissões associadas. Esta estrutura modular permite que o sistema seja extensível e facilmente adaptável a novas necessidades de segurança e controle de acesso.

O diagrama também incorpora a classe *Module*, que representa os diferentes módulos do sistema, como o financeiro e o de associações e cooperativas. Cada

módulo possui um identificador único e um nome, e está associado a múltiplas permissões. Essa relação assegura que cada módulo pode ser configurado com diferentes níveis de acesso, conforme as necessidades específicas de segurança e operação.

Figura 13 - Diagrama de classe: Autenticação e controle de acesso multimódulo



Fonte: Autoria Própria (2024).

Por fim, a relação entre *User*, *Profile*, *Permission* e *Module* é configurada de forma a permitir uma flexibilidade máxima, no qual um usuário pode ter múltiplos papéis (*profiles*), e cada papel pode conferir várias permissões em diferentes módulos. Esse design modular e escalável do diagrama de classes proporciona uma base sólida para a implementação de um sistema de autenticação e controle de acesso eficiente, seguro e adaptável às necessidades dinâmicas do sistema.

4.2.2.1 Diagrama de classes: Cadastro de contas a pagar e receber

O diagrama de classes para o cadastro de contas a pagar e receber foi elaborado para proporcionar um gerenciamento abrangente e eficiente das finanças dentro de um sistema empresarial. No coração deste diagrama estão as classes *Expense* e *Income* (ver Figura 14), que representam, respectivamente, as despesas e receitas do sistema. Cada instância da classe *Expense* possui atributos como *id*, descrição, quantia, data de vencimento, a categoria que pertence, o status que se encontra no sistema, o método de pagamento, além de campos que tratam de recorrência, como o intervalo da recorrência e a data final da recorrência.

Analogamente, a classe *Income* possui atributos similares. Ambas as classes, *Expense* e *Income*, estão associadas à classe *Category*, que permite a categorização flexível de despesas e receitas. Uma associação importante é feita com a classe *BankAccount*, que representa as contas bancárias às quais as despesas e receitas podem ser associadas. Cada *BankAccount* tem atributos como: *id*, *bank* e *balance*. Essa relação é importante pois permite que cada transação financeira seja atribuída à conta bancária correta, garantindo um rastreamento preciso e organizado dos fluxos de caixa.

Além disso, o sistema inclui a classe *CreditCard*, que permite a inclusão de despesas e receitas associadas a cartões de crédito. Essa relação possibilita que as transações realizadas via cartão de crédito sejam integradas ao gerenciamento financeiro do sistema. Para tratar de despesas e receitas recorrentes, o diagrama inclui a classe *recurrence*. Cada recorrência está associada a uma instância de *Expense* ou *Income*. A composição entre *Expense/Income* e *Recurrence* garante que a recorrência esteja estritamente ligada à vida útil da transação financeira.

Figura 14 - Diagrama de classe: Cadastro de contas a pagar e receber



Fonte: Autoria Própria (2024).

Por fim, o diagrama apresenta a classe *Notification*, que é responsável por gerenciar alertas e notificações relacionadas às despesas e receitas e classe *Report* que desempenha um papel essencial ao fornecer funcionalidades de geração e exportação de relatórios financeiros. A integração dessas classes no diagrama de cadastro de contas a pagar e receber proporciona um sistema robusto, capaz de

gerenciar de forma eficaz as finanças de uma organização, oferecendo funcionalidades abrangentes para o controle de despesas, receitas, categorias, contas bancárias, cartões de crédito, recorrências e notificações.

4.2.2.2 Diagrama de classes: Fluxo de caixa futuro

O diagrama de classes do fluxo de caixa futuro foi atualizado para incluir os novos campos necessários na classe *Report*, sem alterações significativas na estrutura global do sistema. A classe *Report* agora possui novos atributos específicos para lidar com a geração de relatórios de fluxo de caixa futuro, sem afetar as outras funcionalidades existentes do sistema.

Os novos campos adicionados à classe *Report* incluem o valor total de despesas e receitas, que são preenchidos com base no filtro do relatório, e também um novo para representar o valor em caixa, que são usados para armazenar os valores calculados das receitas totais, despesas totais e diferença líquida entre elas, respectivamente.

Figura 15 - Diagrama de classe: Fluxo de caixa futuro



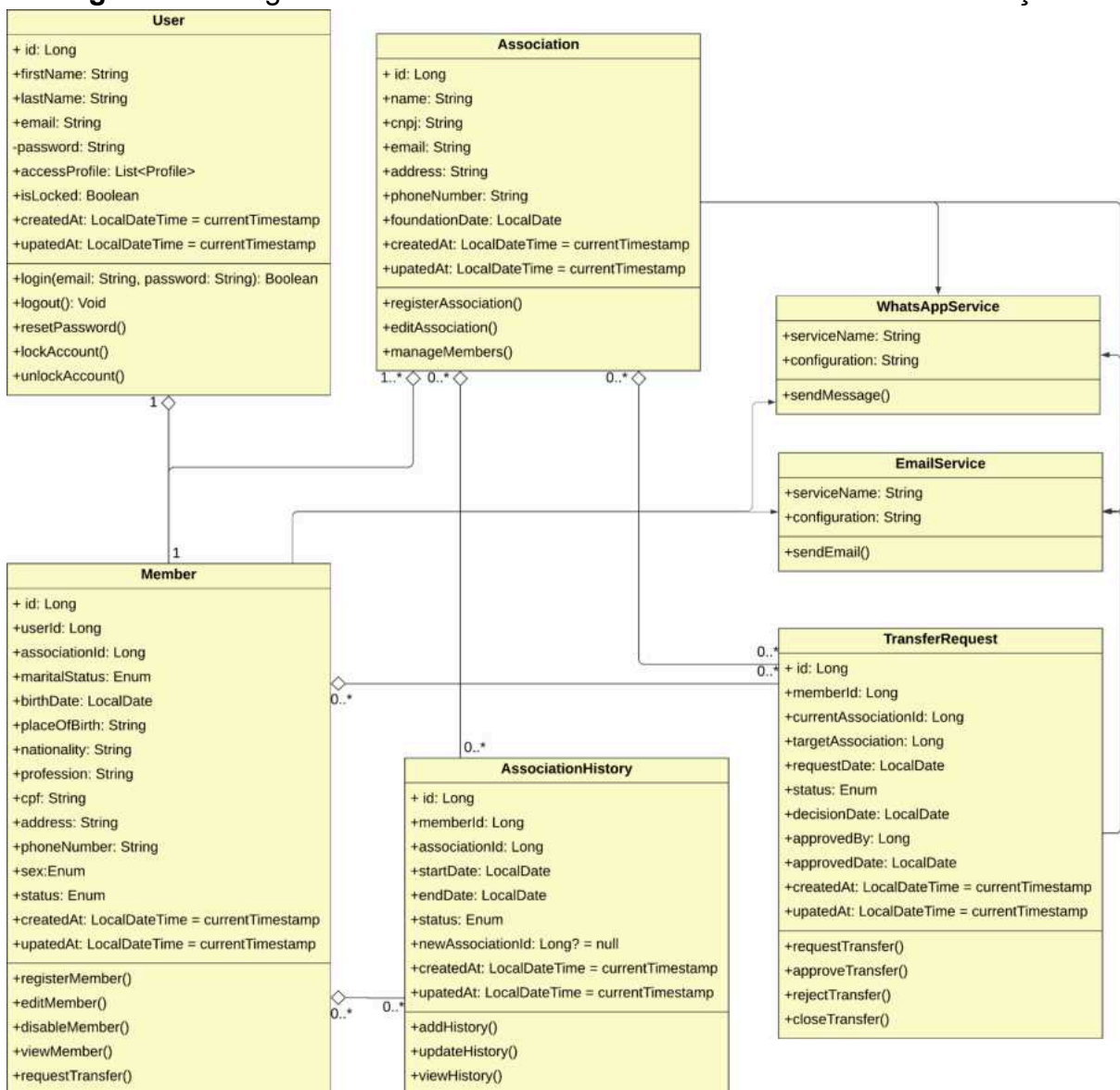
Fonte: Autoria Própria (2024).

Elas são cruciais para fornecer uma visão abrangente do fluxo de caixa futuro e ajudam os usuários a entenderem melhor a sua situação financeira projetada.

4.2.2.3 Diagrama de classes: Cadastro de membros em uma associação

O diagrama de classes para o cadastro de membros em um sistema de associações e cooperativas mostra como os diferentes componentes interagem para gerenciar as informações dos membros e das associações. No centro do diagrama está a classe *Member*, que representa um membro individual. Essa classe contém informações pessoais e associativas essenciais, como nome, estado civil, data de nascimento, naturalidade, nacionalidade, profissão, CPF, endereço e sexo. O membro também possui um status que indica sua situação atual na associação (ver Figura 16).

Figura 16 - Diagrama de classe: Cadastro de membros em uma associação



Fonte: Autoria Própria (2024).

A classe *Association* é responsável por representar uma associação ou cooperativa. Ela contém atributos como nome, CNPJ e endereço. Cada associação pode ter múltiplos membros e cada membro apenas uma associação por vez. Para gerenciar o histórico de um membro dentro de diferentes associações, temos a classe *AssociationHistory*. Esta classe mantém um histórico das associações às quais um membro pertenceu, registrando as datas de início e término e o status da associação.

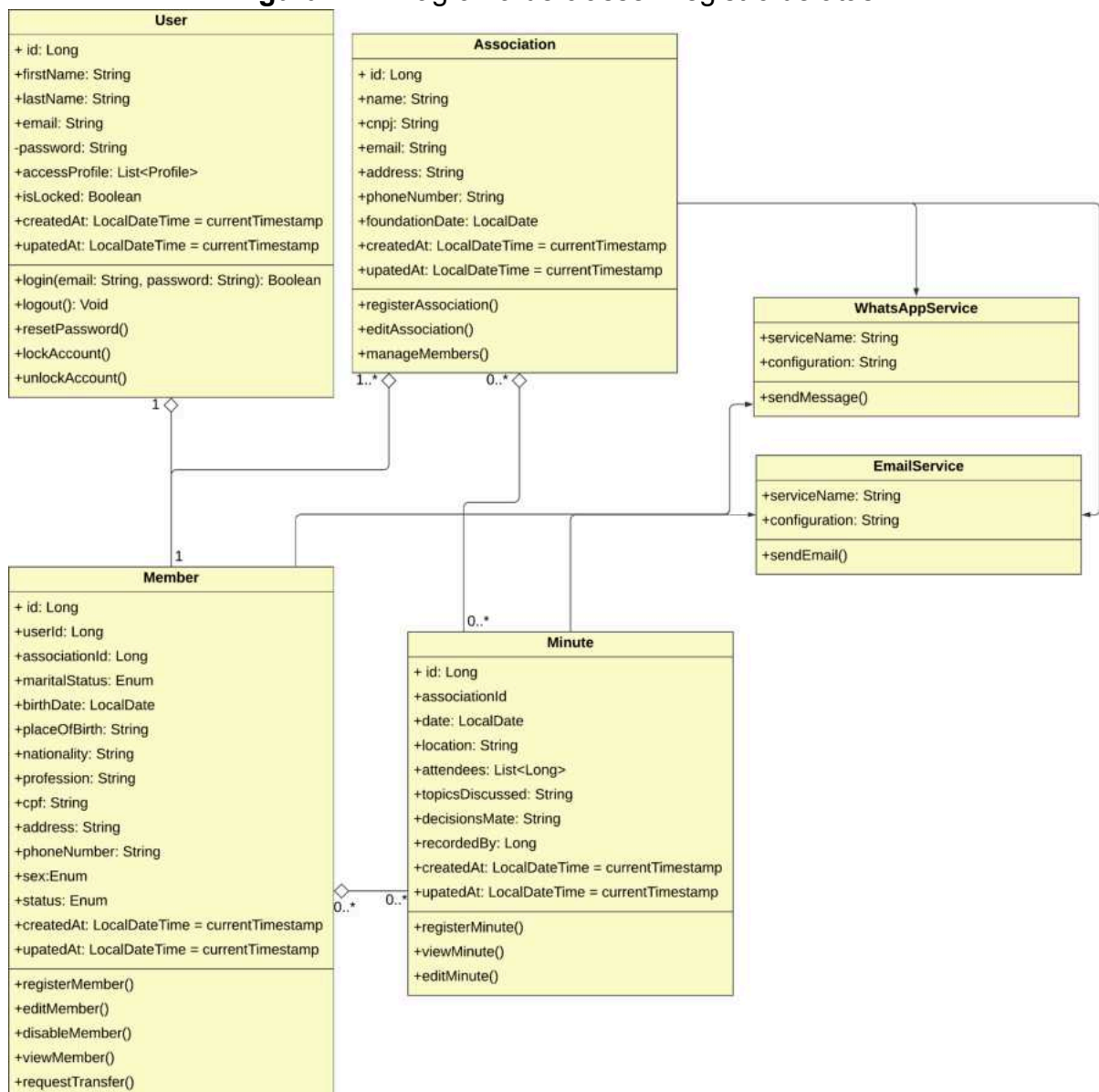
Quando um membro deseja mudar de uma associação para outra, a classe *TransferRequest* entra em ação. Esta classe gerencia as solicitações de transferência, contendo informações sobre a associação atual, a associação alvo, a data da solicitação, o status da solicitação e a data e a pessoa que aprovou a transferência, caso aprovada.

Para facilitar a comunicação e a notificação dos membros e administradores sobre eventos importantes, como a solicitação de transferência, são utilizados os serviços *EmailService* e *WhatsAppService*. Estes serviços permitem enviar notificações por *email* e *WhatsApp*, respectivamente, garantindo que todos os envolvidos sejam informados prontamente sobre as mudanças e solicitações.

4.2.2.4 Diagrama de classes: Registro de atas

O diagrama de classes para o cadastro de atas ilustra como diferentes componentes interagem para permitir o registro e a visualização de atas de reuniões, a classe central é *Minute* (ver Figura 17), que representa uma ata de reunião. Ela contém informações como a data da reunião, local, lista de presentes, tópicos discutidos e decisões tomadas. Apenas usuários com perfil de secretário ou presidente podem registrar novas atas, enquanto todos os membros podem visualizar as atas registradas.

Figura 17 - Diagrama de classe: Registro de atas



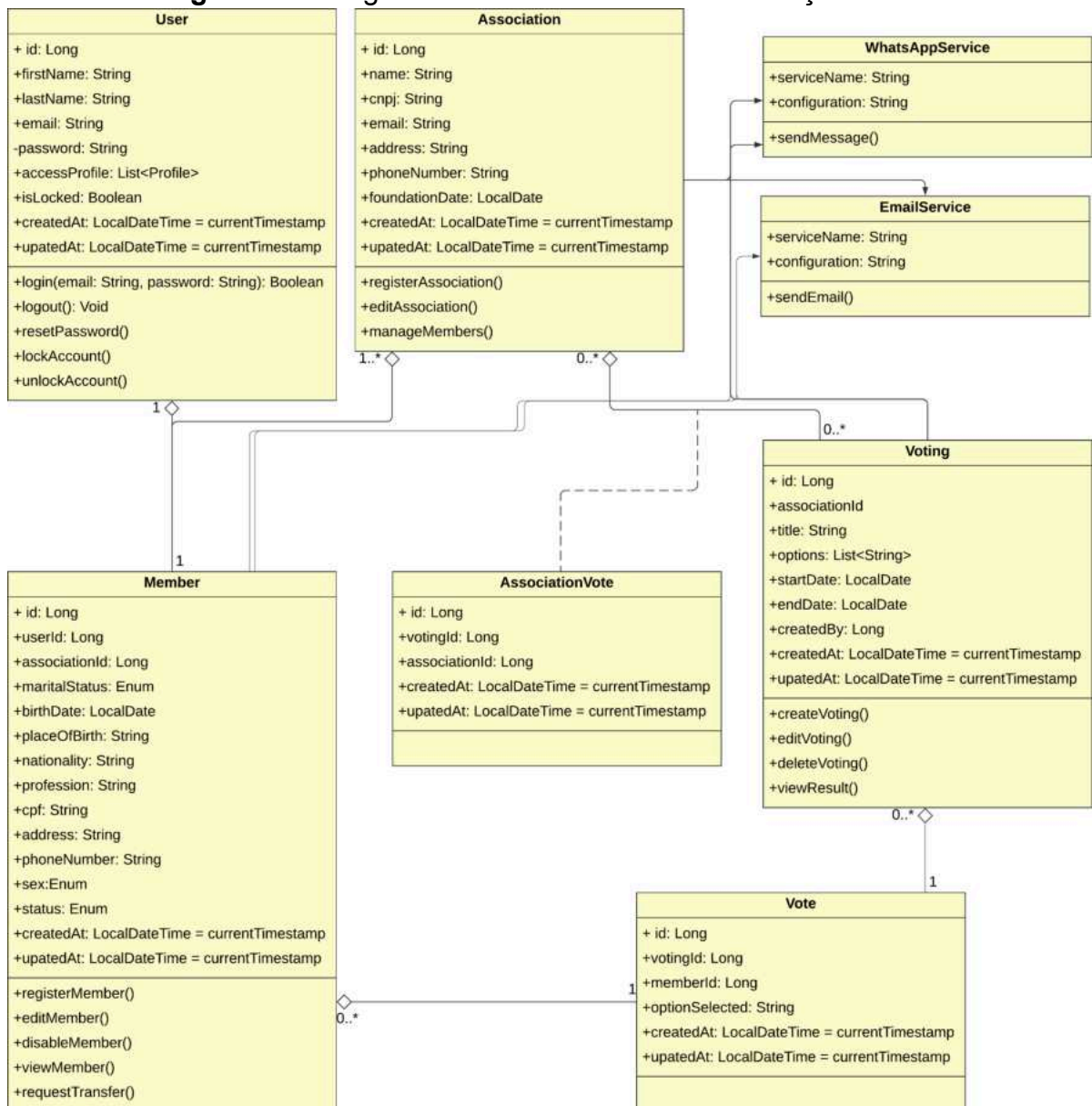
Fonte: Autoria Própria (2024).

Adicionalmente, são utilizados os serviços *EmailService* e *WhatsAppService*, que permitem notificar todos os membros sobre o cadastro de uma nova ata. Esses serviços garantem que todos os membros sejam informados sobre as decisões tomadas e as mudanças ocorridas, promovendo transparência e comunicação eficiente dentro da associação.

4.2.2.5 Diagrama de classes: Sistema de votação online

No diagrama de classes para o sistema de votação online, a classe *Voting* representa uma votação específica, contendo atributos como título, descrição, opções de voto, data de início e término, e quem criou a votação (ver Figura 18). Apenas usuários com perfil de secretário ou presidente podem criar e registrar novas votações.

Figura 18 - Diagrama de classe: Sistema de votação online



Fonte: Autoria Própria (2024).

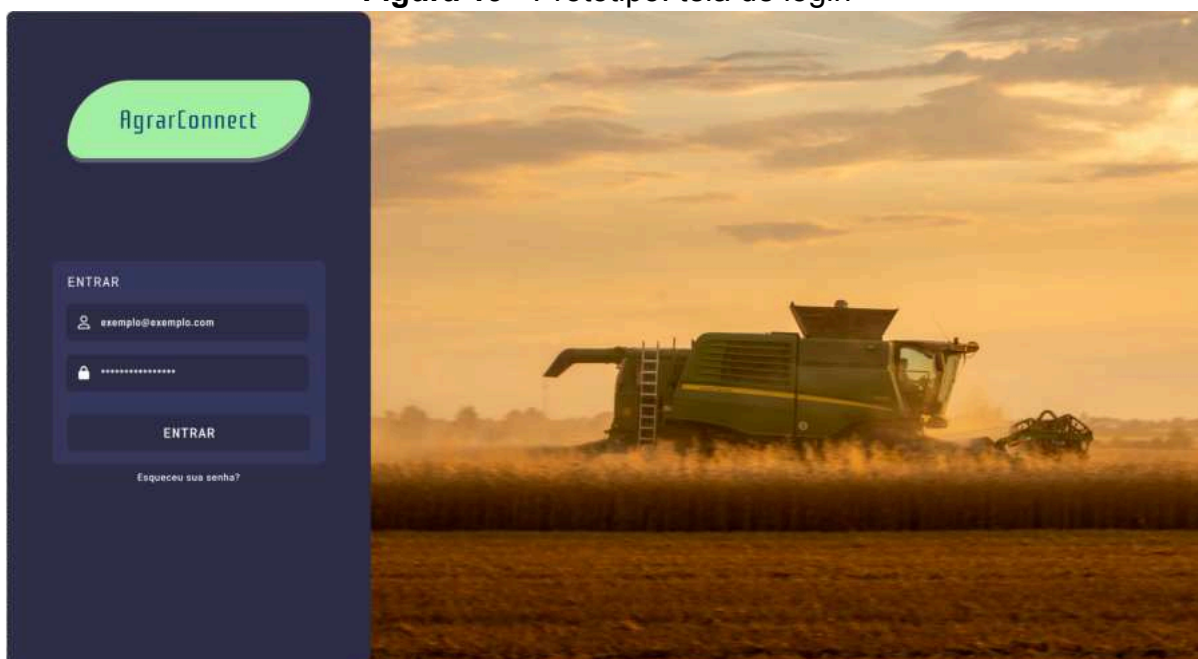
A classe *Vote* representa um voto individual, registrando qual membro votou em qual opção de qual votação. Isso garante que cada membro possa votar apenas uma vez por votação e que o sistema possa rastrear e contabilizar os votos corretamente.

Adicionalmente, são utilizados os serviços *EmailService* e *WhatsAppService*, que permitem notificar todos os membros sobre a criação de uma nova votação e os resultados das votações. Esses serviços garantem que todos os membros sejam informados de maneira eficiente e oportuna, promovendo transparência e participação ativa.

4.3 Prototipagem

A prototipagem das telas concentrou-se principalmente na elaboração de um padrão visual para o sistema e na melhoria da rapidez de troca entre as funcionalidades de cada módulo, mantendo a simplicidade de uso. Para isso, utilizamos o *Figma*, uma ferramenta de design colaborativo baseada na nuvem.

Figura 19 - Protótipo: tela de login



Fonte: Autoria Própria (2024).

O Figma permite criar interfaces interativas, facilitando a visualização de como o usuário irá interagir com o sistema. Além disso, ele possibilita a colaboração

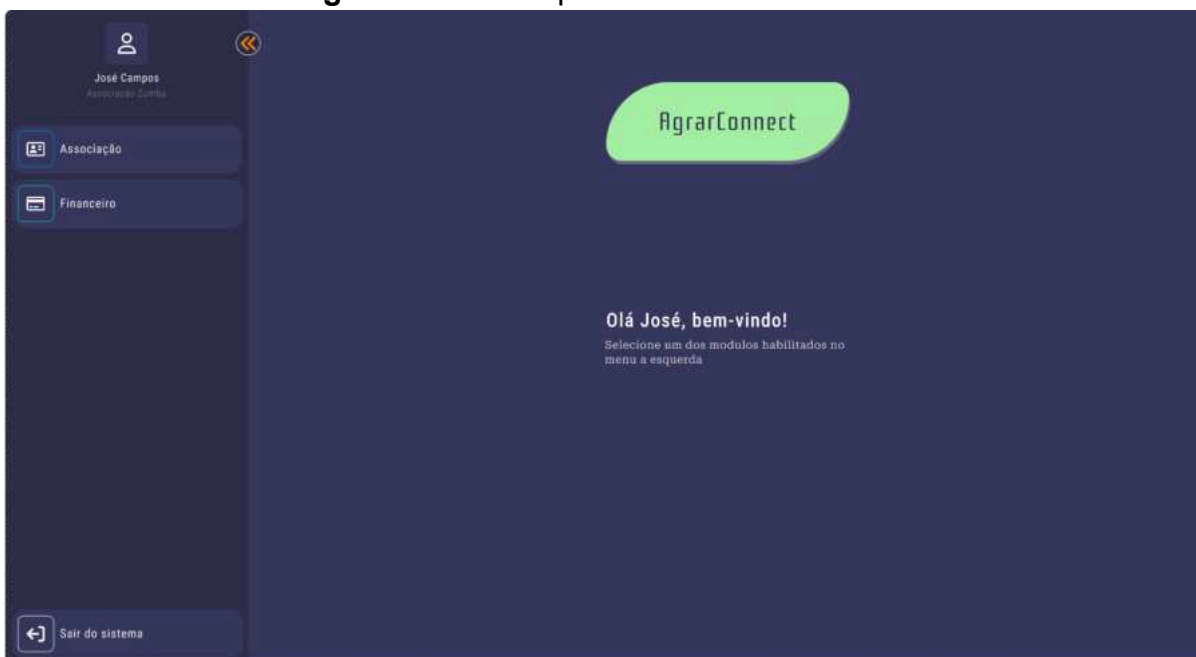
em tempo real, permitindo que diferentes membros da equipe de *design* e desenvolvimento trabalhem juntos de forma eficiente.

A Figura 19 representa a tela de login do sistema. Ela segue o padrão de mercado, apresentando um formulário no qual o usuário pode inserir seus dados de acesso (*email* e senha) para logar no sistema. Acima do formulário, é exibida a logo com o nome AgrarConnect, situada sobre um retângulo arredondado que representa abstratamente uma folha.

Após realizar o login no sistema, o usuário se depara com a tela inicial (ver Figura 20), em que, ao centro, é exibida uma mensagem de boas-vindas e um texto orientando-o a escolher um dos módulos disponíveis no menu à esquerda. Na parte superior do menu, o usuário pode acessar a página de perfil, no qual é possível atualizar ou adicionar informações complementares.

Na página de perfil, também é possível ver mais detalhes sobre os acessos atribuídos ao seu usuário. Abaixo do perfil, encontra-se a lista de módulos disponíveis para acesso, e na parte inferior do menu, há a opção de se desconectar do sistema.

Figura 20 - Protótipo: tela inicial do sistema

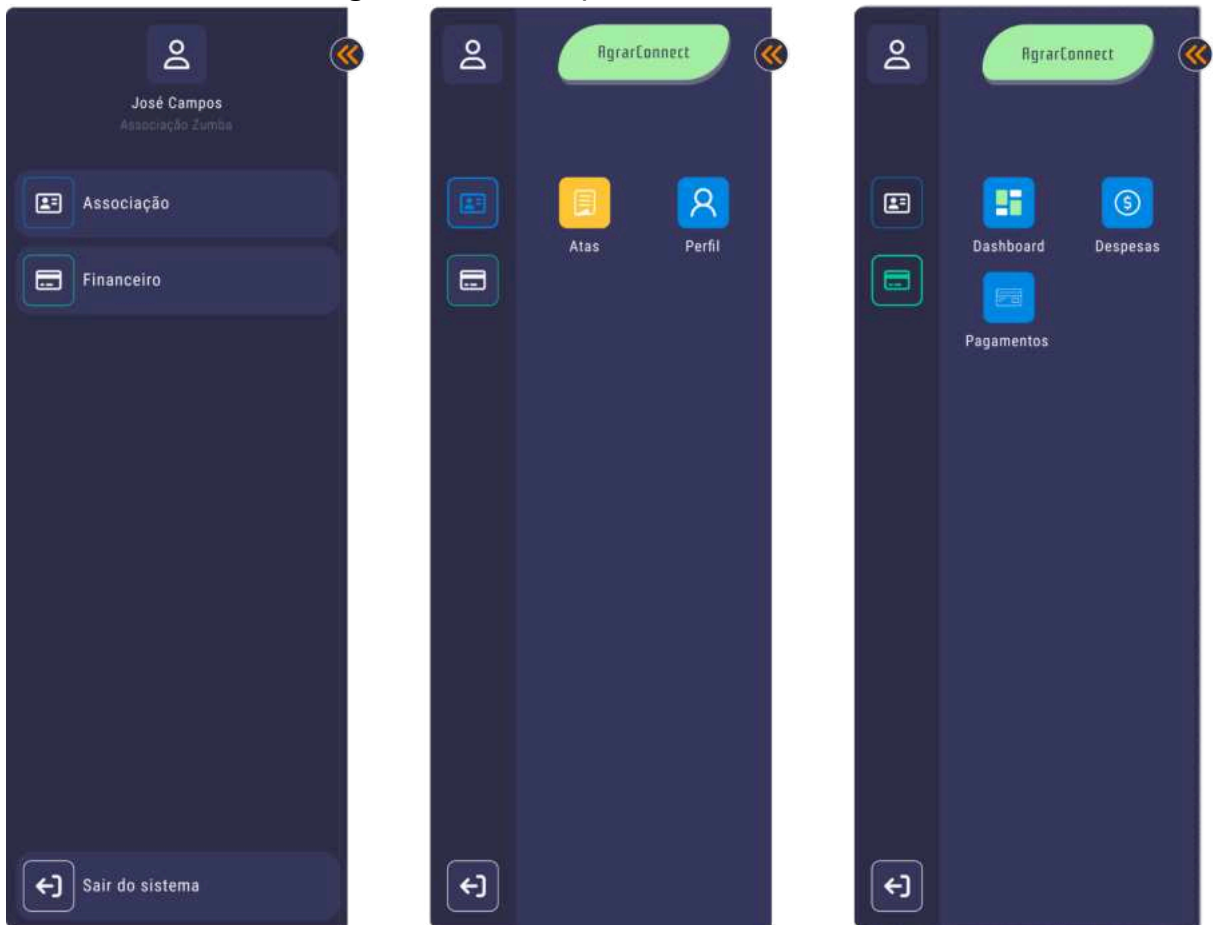


Fonte: Autoria Própria (2024).

Cada módulo do sistema pode ter várias funcionalidades, e listar todas elas em um único menu resultaria em uma interface muito extensa e confusa. Uma

solução lógica para essa situação é agrupar as funcionalidades e listá-las em tópicos, como é comum em muitos sistemas. Durante a elaboração do menu, identificamos a possibilidade de adicionar um submenu dentro de cada módulo. Isso torna a navegação mais intuitiva, facilitando a troca entre os módulos e o acesso às funcionalidades específicas de cada um.

Figura 21 - Protótipo: menu e submenus



Fonte: Autoria Própria (2024).

A Figura 21 apresenta as trocas entre os menus, demonstrando como a navegação no sistema foi organizada para ser intuitiva e eficiente. Nessa figura, é possível observar como os módulos principais são exibidos no menu à esquerda, enquanto os submenus relacionados a cada módulo são apresentados após um módulo ser selecionado. Isso permite ao usuário acessar rapidamente as funcionalidades específicas de cada módulo sem sobrecarregar a interface. A estrutura de menus e submenus facilita a localização e o uso das diversas funcionalidades, garantindo uma experiência de usuário mais fluida e agradável.

Figura 22 - Protótipo: modelo formulário com poucos campos

The figure displays three sequential screens of a mobile application for recording expenses. Each screen is titled 'CADASTRANDO NOVA DESPESA' and features a close button (X) in the top right corner. The first screen shows the initial form with fields for 'Dados da despesa', 'Valor' (0,00), 'Data de criação' (30/03/2024), 'Descrição' (with a placeholder 'Insira um texto descrevendo a despesa'), 'Pagamento', 'Método de pagamento' (dropdown), 'Categoria' (dropdown), 'Parcelado' (checkbox), 'Recorrência', and 'Recorrente' (checkbox). The second screen shows the 'Parcelado' checkbox checked. The third screen shows the 'Recorrente' checkbox checked, with additional fields for 'Data de vencimento' (30/03/2024) and 'Tipo de recorrência' (dropdown). All screens have a 'CRIAR' button at the bottom.

Fonte: Autoria Própria (2024).

Figura 23 - Protótipo: modelo formulário com muitos campos a serem preenchidos

The figure displays six sequential screens of a mobile application for recording expenses, showing a more complex form with many fields. Each screen is titled 'CADASTRANDO NOVA DESPESA' and features a close button (X) in the top right corner. The first screen shows the initial form with fields for 'Dados da despesa', 'Valor' (0,00), 'Data de criação' (30/03/2024), 'Descrição' (with a placeholder 'Insira um texto descrevendo a despesa'), 'Pagamento', 'Método de pagamento' (dropdown), 'Categoria' (dropdown), 'Parcelado' (checkbox), 'Recorrência', and 'Recorrente' (checkbox). The second screen shows the 'Parcelado' checkbox checked. The third screen shows the 'Recorrente' checkbox checked. The fourth screen shows the 'Recorrente' checkbox checked, with additional fields for 'Data de vencimento' (30/03/2024) and 'Tipo de recorrência' (dropdown). The fifth screen shows the 'Recorrente' checkbox checked, with additional fields for 'Data de vencimento' (30/03/2024) and 'Tipo de recorrência' (dropdown). The sixth screen shows the 'Recorrente' checkbox checked, with additional fields for 'Data de vencimento' (30/03/2024) and 'Tipo de recorrência' (dropdown). The first three screens have an 'AVANÇAR' button at the bottom, and the last three have a 'SALVAR' button at the bottom.

Fonte: Autoria Própria (2024).

Por fim, também iniciamos a definição de um padrão para os formulários de cadastro, optando por dois modelos distintos. Para formulários com poucos campos, ficou definido o modelo apresentado na Figura 22. Esse modelo foi projetado para ser simples e direto, facilitando o preenchimento rápido e eficiente pelo usuário.

Para formulários mais extensos, adotamos um *layout* de agrupamento, organizando as informações em tópicos, passando mais clareza para o usuário sobre os dados que estão sendo preenchidos no momento e quantos tópicos ainda faltam para concluir o cadastro. Esse método facilita a navegação e o entendimento do formulário, garantindo uma experiência mais organizada e eficiente, conforme apresentado na Figura 23.

Este trabalho teve como objetivo principal explorar o caso de uso proposto, focando na identificação de soluções práticas para os desafios enfrentados pelos produtores rurais, especialmente na integração entre o campo e o meio digital. Através de uma análise do referencial teórico, foi possível correlacionar aspectos de gestão, tecnologia e produção agrícola a fim de identificar os principais desafios enfrentados pelos agricultores na adoção de novas tecnologias e, dessa forma, apresentar estratégias viáveis para o desenvolvimento de um sistema integrado de gestão.

Concluimos que a adoção de tecnologias digitais e a adaptação eficiente de tecnologias existentes ao contexto de produção dos pequenos e médios produtores rurais, aliada a uma gestão eficaz e à capacitação dos produtores, têm potencial de promover um melhor desenvolvimento tecnológico sustentável no setor agrícola, atendendo ao contexto de produção desses produtores.

5 CONCLUSÃO

Este trabalho teve como objetivo apresentar um sistema integrado para pequenos e médios produtores rurais, visando melhorar a eficiência da gestão em suas cadeias produtivas. Por meio da análise de trabalhos publicados sobre gestão agrícola e integração com tecnologia, identificamos os principais desafios enfrentados pelos produtores. Desenvolvemos um modelo de arquitetura escalável que permite a adição de novas funcionalidades ao sistema, utilizando tecnologias de ponta. Esse sistema pode melhorar significativamente a gestão dos pequenos produtores rurais, atendendo às características específicas de suas produções e aumentando a eficiência da cadeia produtiva.

A análise de trabalhos publicados foi primordial para identificar os desafios enfrentados pelos produtores rurais, proporcionando uma base sólida para o desenvolvimento de soluções efetivas. Entender as dificuldades comuns, como a falta de acesso a tecnologias avançadas, problemas de conectividade, a complexidade na gestão das operações agrícolas e a utilização de diferentes ferramentas separadas, permite que as soluções propostas sejam mais direcionadas e eficazes. Esta abordagem garante que o sistema desenvolvido atenda às necessidades reais dos produtores, aumentando a probabilidade de adoção e sucesso na prática.

A identificação dos problemas comuns enfrentados pelos produtores levou diretamente ao desenvolvimento da arquitetura escalável do sistema. Esta arquitetura é importante porque oferece flexibilidade e adaptabilidade, permitindo que o sistema evolua com as necessidades dos produtores e as inovações tecnológicas. Além disso, a escalabilidade facilita a inclusão de novas funcionalidades sem a necessidade de grandes reestruturações, garantindo que o sistema permaneça relevante e útil a longo prazo. A adoção de tecnologias de ponta assegura que os produtores possam beneficiar-se das últimas inovações, melhorando a eficiência e competitividade de suas cadeias produtivas.

Embora os resultados sejam promissores, encontramos algumas limitações. Os requisitos levantados não foram amplamente explorados em diferentes contextos de produção, públicos e regiões. Além disso, é necessária uma análise mais detalhada dos custos do sistema para determinar seu valor de comercialização. Pesquisas futuras podem expandir a amostra de requisitos e incluir diferentes

setores de produção, a fim de validar as especificidades de cada setor e aplicar essas características de forma integrada e eficiente. Também é possível adicionar mais módulos ao sistema, como um para comercialização da cadeia produtiva, e aplicar IA e *Big Data* para aumentar a eficiência por meio da análise e aprendizagem dos dados coletados.

A implementação do sistema exige uma equipe multidisciplinar composta por desenvolvedores, gestores de projetos, especialistas em DevOps, QA (controle de qualidade), designers de UX e UI, além de especialistas em segurança da informação. Todo o desenvolvimento seguirá a metodologia Scrum, promovendo uma gestão ágil e eficiente do projeto. Adicionalmente, é fundamental garantir uma fonte de financiamento para o desenvolvimento do software. Isso pode ser viabilizado por intermédio de investidores que financiem o projeto com o intuito de comercializá-lo como um serviço SaaS (*Software as a Service*).

Em termos de desenvolvimento contínuo, os próximos passos incluem a expansão do sistema para incorporar módulos adicionais que atenderão a aspectos específicos da produção agrícola, como a comercialização e a logística, além da elaboração de artefatos como o modelo de Entendi-Relacionamento e o diagrama de fluxo de evento.

A integração de inteligência artificial (IA) e *Big Data* será um diferencial importante da solução à medida que mais usuários utilizem o sistema, otimizando a tomada de decisões através da análise de grandes volumes de dados e a identificação de padrões que possam melhorar a eficiência operacional, contribuindo para a produtividade dos agricultores. Além disso, o desenvolvimento de uma interface de usuário mais intuitiva e acessível está nos planos, para garantir que o sistema seja fácil de usar por produtores com diferentes níveis de familiaridade com a tecnologia.

Em resumo, este trabalho apresentou um sistema integrado para pequenos e médios produtores rurais, com o objetivo de melhorar a eficiência da gestão em suas cadeias produtivas. A análise de trabalhos publicados e a identificação dos principais desafios enfrentados pelos produtores permitiram o desenvolvimento de uma arquitetura escalável, utilizando tecnologias de ponta. Apesar das limitações encontradas, como a necessidade de explorar requisitos em diferentes contextos e uma análise mais detalhada dos custos, os resultados são promissores.

Pesquisas futuras podem expandir e validar o sistema em diversos setores, além de adicionar novos módulos e tecnologias como IA e *Big Data* e o aprimoramento da interface do usuário para torná-la mais intuitiva e acessível. Este trabalho reforça a importância de soluções integradas e específicas para melhorar a eficiência na gestão agrícola, contribuindo significativamente para o desenvolvimento de software na área.

REFERÊNCIAS

- [Farmis]. Agrobase - infestante,doença. Android. Disponível em: https://play.google.com/store/apps/details?id=lt.farmis.apps.farmiscatalog&hl=pt_BR&gl=US. Acesso em: 16 nov. 2023.
- Aegro. Disponível em: <https://aegro.com.br/>. Acesso em: 16 nov. 2023.
- BATISTA, Paulo Sillas Bandeira; BARROS, Giovana Lyssa Toledo de; DELFINO, Lara; ZUCENTE, Guilherme Ricardo; KETTENE, Lucas Guilherme. Administração rural: conceitos e desafios, 2023. Trabalho de conclusão de curso (Ensino Médio com habilitação Profissional de Técnico em Administração) - Escola Técnica Estadual Prof. Ídio Zucchi - Bebedouro, SP, 2023. Disponível em: <https://ric.cps.sp.gov.br/handle/123456789/16629>. Acesso em: 28 jun. 2024
- BECK, Kent. Extreme Programming Explained: Embrace Change. Addison Wesley, 1999. 224 p. ISBN-13: 978-0201616415.
- BOLFE, Édson Luis *et al.* Desafios, tendências e oportunidades em agricultura digital no Brasil. Agricultura Digital: Pesquisa, desenvolvimento e inovação nas cadeias produtivas, 1ª edição, p. 380-406, 2020. Disponível em: <https://ainfo.cnptia.embrapa.br/digital/bitstream/item/218131/1/LV-Agricultura-digital-2020.pdf>. Acesso em: 01 out. 2023.
- CEPEA - Centro de Estudos Avançados em Economia Aplicada. Sustentado por safra recorde no campo, pib do agronegócio tem alta modesta no primeiro trimestre. Piracicaba (SP), Cepea; 2023. Disponível em: <https://cepea.esalq.usp.br/upload/kceditor/files/PIB-DO-AGRO-27JUN2023.pdf>. Acesso em: 25 jul. 2023.
- CÓCARO, H.; JESUS, J. C. S. A agroinformática em empresas rurais: algumas tendências. In: Congresso Da Sociedade Brasileira De Economia, Administração E Sociologia Rural, 46., 2008, Rio Branco, AC. Anais do Congresso... Lavras, MG, 2008. Disponível em: <https://ageconsearch.umn.edu/record/102898/>. Acesso em: 29 jul. 2023.
- DELLALUCE, Jason. Enhancing Symbolic AI Ecosystems with Probabilistic Logic Programming: a Kotlin Multi-Platform Case Study. [Tese de Mestrado]. Università di Bologna, Bologna, 2021. Disponível em: <https://amslaurea.unibo.it/23856/1/%5B2021-07-14%5D%20Jason%20Dellaluce%20-%20Master%20Thesis.pdf>. Acesso em: 22 mar. 2024.
- DIAS, J. M. F.; RODRIGUES, R. C. M. C.; PIRES, D. F. (2012). A segurança de dados na computação em nuvens nas pequenas e médias empresas. RESIGeT - Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica, 2(1), 56-69. Disponível em: <https://www.academia.edu/download/86606634/278.pdf>. Acesso em: 06 ago. 2023.
- EMBRAPA – Empresa Brasileira De Pesquisa Agropecuária. Visão 2030: o futuro da agricultura brasileira, Brasília, DF, 2018. 212p. il. color. Disponível

em:<https://www.embrapa.br/documents/10180/9543845/Vis%C3%A3o+2030+-+o+futuro+da+agricultura+brasileira/2a9a0f27-0ead-991a-8cbf-af8e89d62829?version=1.1>. Acesso em: 07 ago. 2023.

EUGSTER, P. T. *et al.* The many faces of publish/subscribe. ACM computing surveys (CSUR), ACM New York, NY, USA, v. 35, n. 2, p. 114–131, 2003. DOI: <https://doi.org/10.1145/857076.857078>. Acesso em: 26 maio 2024.

FIGUEIREDO, P. R. R.; EVANGELISTA, I. M. A.; MONTEIRO, F. D.; EID, F. Planejamento estratégico participativo em cooperativa recém formada de agricultores familiares: o caso da Cooperativa dos Agricultores Familiares Guamaenses (COAFAG). Tecnologia e Sociedade, Curitiba, v. 19, n. 58, p. 191-211, out./dez. 2023. DOI: [10.3895/rts.v19n58](https://doi.org/10.3895/rts.v19n58). Acesso em: 16 mar. 2024.

FOWLER, Martin. Microservice Premium. **Martin Fowler**, 2015. Disponível em: <https://martinfowler.com/bliki/MicroservicePremium.html>. Acesso em: 25 maio 2024.

GALINARI, Graziella. Embrapa oferece serviço de acesso a dados agrometeorológicos para uso em soluções digitais. Embrapa, 2022. Disponível em: <https://www.embrapa.br/busca-de-noticias/-/noticia/72771790/embrapa-oferece-servico-de-acesso-a-dados-agrometeorologicos-para-uso-em-solucoes-digitais>. Acesso em: 17 set. 2023.

GOOGLE. Android Developers. Android's Kotlin-first approach. Disponível em: <https://developer.android.com/kotlin/first>. Acesso em: 17 mar. 2024.

GROMOV, Pavel; CHERNYSHEV, Yaroslav. Integration of Kotlin Multiplatform Projects with Swift Package Manager Dependencies. In: 29th Conference of Open Innovations Association FRUCT, 2021, Tampere, Finland. Anais/Proceedings. Tampere: Open Innovations Association FRUCT, 2021. Disponível em: <https://fruct.org/conferences/29/program/>. Acesso em: 19 mar. 2024.

HUFF, Guilherme Feier. Scrum: um estudo de caso aplicado ao desenvolvimento da ferramenta LStyle. 2023. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Tecnológica Federal do Paraná, Santa Helena, 2023. Disponível em: <http://repositorio.utfpr.edu.br/jspui/handle/1/32130>. Acesso em: 04 maio. 2024.

JETBRAINS. Opções de compartilhamento de código no KMP. 2023. [Imagem]. Disponível em: <https://blog.jetbrains.com/pt-br/kotlin/2023/11/kotlin-multiplatform-estavel-e-pronto-para-producao/>. Acesso em: 19 mar. 2024.

JETBRAINS. Plataformas de destino. 2024. [Image]. Disponível em: <https://kotlinlang.org/docs/multiplatform-discover-project.html#targets>. Acesso em: 22 mar. 2024.

KURNIAWAN, D. E.; DZIKRI, A. (2024). Development of a Prototype for Microservices Architecture and API Gateway Integration in an eLearning Platform. ICAE [International Conference on Applied Engineering / EAI [Research Meets Innovation]], DOI: <https://dx.doi.org/10.4108/eai.7-11-2023.2342930>. Acesso em: 1 abr.

2024.

LOPES, B. F. *et al.* Ferramentas de gestão financeira: uma pesquisa sobre o seu papel nas micro e pequenas empresas. *LIBERTAS: Rev. Ciênci. Soc. Apl.*, Belo Horizonte, v. 9, n. 1, p. 51-77, jan./jul. 2019.

LUCAS, Paloma Reis. O agricultor familiar e os aplicativos móveis: fatores que afetam o uso da tecnologia no campo. 2023. 139 f. Dissertação (Programa Stricto Sensu em Governança, Tecnologia e Inovação) - Universidade Católica de Brasília, Brasília, 2023. Disponível em: <https://bdtd.ucb.br:8443/jspui/handle/tede/3199>. Acesso em 21 set. 2023.

MAGALHÃES, Pedro Felipe Santos. Uma biblioteca para testes determinísticos em sistemas distribuídos com comunicação assíncrona. 2023. 67 f. Dissertação (Mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2023. Disponível em: <https://www.maxwell.vrac.puc-rio.br/62872/62872.PDF>. Acesso em: 26 maio 2024.

MASSRUHÁ, S. M. F. S.; LEITE, M. A. de A.; LUCHIARI JUNIOR, A.; EVANGELISTA, S. R. M. A transformação digital no campo rumo à agricultura sustentável e inteligente. In: MASSRUHÁ, S. M. F. S.; LEITE, M. A. de A.; OLIVEIRA, S. R. de M.; MEIRA, C. A. A.; LUCHIARI JUNIOR, A.; BOLFE, E. L. (Ed.). *Agricultura digital: pesquisa, desenvolvimento e inovação nas cadeias produtivas*. Brasília, DF: Embrapa, 2020. cap. 1, p. 20-45. Disponível em: <https://ainfo.cnptia.embrapa.br/digital/bitstream/item/217698/1/LV-Agricultura-digital-2020-cap1.pdf>. Acesso em: 24 abri. 2024.

PINO V., Edwin. Drones a tool for efficient agriculture: a high-tech future. *Idesia*, Arica , v. 37, n. 1, p. 75-84, março de 2019 . Disponível em: http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-34292019000100075&lng=es&nrm=iso. Acesso em: 09 set. 2023.

PRODANOV, C. C.; FREITAS, E. C. *Metodologia do Trabalho Científico: métodos e técnicas da pesquisa e do trabalho acadêmico*. 2ª ed. Novo Hamburgo, RS: Feevale, 2013. Disponível em: https://drive.google.com/file/d/1lp5RRyTrt6X8UPoq2jJ8gO3UEfM_JJd/view. Acesso em: 15 maio. 2024.

SANTOS, Glaucia Schnoeller dos. *Extreme programming: metodologia ágil*, 2013. Trabalho de conclusão de curso (Curso de Tecnologia em Análise e Desenvolvimento de Sistemas) - Faculdade de Tecnologia de Americana, Americana, 2013. Disponível em: <https://ric.cps.sp.gov.br/handle/123456789/604>. Acesso em: 11 nov. 2023.

SCHWABER, Ken; BEEDLE, Mike. *Agile Software Development with Scrum*. Illustrated ed. ELT Importado Pearson, 2001. 158 p. ISBN-13: 978-0130676344.

SENDUK, F. X.; NAJOAN, X. B. N. N.; SOMPIE, S. R. U. A. Development of Microservices Architecture with RESTful API Gateway using Backend-for-frontend Pattern in Higher Education Academic Portal. *Jurnal Teknik Informatika*, v.18, n.1, p.

315-324 mar. 2023. DOI: <https://doi.org/10.35793/jti.v18i1.50402>. Acesso em: 1 abr. 2024.

SHEPHERD, M.; TURNER, J. A.; SMALL, B.; WHEELER, D. Priorities for science to overcome hurdles thwarting the full promise of the 'digital agriculture' revolution. *Journal of the Science of Food and Agriculture*, v. 100, n. 14, Sept 2018. DOI: [10.1002/jsfa.9346](https://doi.org/10.1002/jsfa.9346). Acesso em: 01 maio. 2024. SILVA, D.; Guimarães Filho, L. P. (2023). Aplicando as instruções do SCRUM em um projeto de implementação de sistemas. *Revista Sociedade Científica*, v. 6, n. 1. DOI: <https://doi.org/10.61411/rsc29022> Acesso em: 04 maio. 2024.

SILVA, Roni da Cruz. (2021). Aplicando Arquitetura de Microsserviços no Desenvolvimento de Software. *Revista Ubiquidade*, v. 4, n. 2, p. 58-84. Disponível em: <https://revistas.anchieta.br/index.php/RevistaUbiquidade/article/view/1859>. Acesso: 27 maio 2024.

SOARES, M. d. S. Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software. *Revista Eletrônica de Sistemas de Informação (RESI)*, v. 3, n. 1, 2004. Disponível em: <https://periodicosibepes.org.br/index.php/reinfo/article/view/146/0>. Acesso em: 11 nov. 2023.

SOMMERVILLE, Ian. *Engenharia de Software*. 10. ed. São Paulo: Pearson Education do Brasil, 2018. 1 vol. (Título original: *Software Engineering*). ISBN: 978-85-430-2497-4.

SORDI, V. F.; VAZ, S. C. M. Os Principais Desafios para a Popularização de Práticas Inovadoras de Agricultura Inteligente. *Desenvolvimento em Questão*, [S. l.], v. 19, n. 54, p. 204–217, 2021. DOI: 10.21527/2237-6453.2021.54.204-217. Disponível em: <https://www.revistas.unijui.edu.br/index.php/desenvolvimentoemquestao/article/view/10891>. Acesso em: 25 abr. 2024.

SOUZA, Wilians Douglas Conde. *Guia de orientações na migração de sistemas de informação do ambiente monolítico para ambiente de microsserviços na computação em nuvem em microempresas de software*. 2023. 239 f. Dissertação (Programa de Pós-Graduação em Informática e Gestão do Conhecimento) - Universidade Nove de Julho, São Paulo.

STAPLETON, Jennifer. *DSDM: A Framework for Business Centered Development*. 1ª ed. Addison Wesley, 1997. 192 p. ISBN-13: 978-0201178890.

TAKEUCHI, H.; NONAKA, I. (1986). The New New Product Development Game. *Harvard Business Review*. Recuperado de <https://hbr.org/1986/01/the-new-new-product-development-game> Acesso em: 04 maio. 2024.

VIOLA, Eduardo.; MENDES, V. *Agricultura 4.0 e mudanças climáticas no Brasil*. *Ambiente & Sociedade*, São Paulo, v. 25, 2022. DOI: <https://doi.org/10.1590/1809-4422asoc20200246r2vu2022L3AO>. Disponível em: <https://www.scielo.br/j/asoc/a/Bwg7NVTs5kcrK6WRxbqh4LS/?lang=pt>. Acesso em:

25 abr. 2024.