CENTRO PAULA SOUZA
FACULDADE DE TECNOLOGIA DE FRANCA
“Dr. THOMAZ NOVELINO”

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LUCAS D’ELIA MIRANDA DE ANDRADE
PEDRO HENRIQUE FERNANDES

SISTENDA, ESTUDO DE CASO EM UMA COMUNIDADE CATÓLICA DE FRANCA/SP

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Franca - “Dr. Thomaz Novelino”, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Carlos Alberto Lucas

FRANCA/SP
2023

SISTENDA, ESTUDO DE CASO EM UMA COMUNIDADE CATÓLICA DE FRANCA/SP

LUCAS D'ELIA MIRANDA DE ANDRADE[1]

PEDRO HENRIQUE FERNANDES[2]

Resumo

O controle e gerenciamento do processo de comunicação de dados e tramitação de documentos é considerado como problema em algumas empresas, e não é diferente em um grupo ou comunidade. Habitualmente, estes grupos ou comunidades crescem de forma rápida e às vezes acabam se perdendo no controle dos próprios meios de comunicação, principalmente quando adotam mecanismos que não sejam a utilização da tecnologia. Tal ação, eventualmente, pode provocar a perda de documentos e a dificuldade na compreensão dos processos ocorridos anteriormente. Prova disto foi o resultado de uma entrevista realizada com alguns líderes e pessoas da Comunidade Tenda que relataram e apresentaram que muitas vezes ocorrem falhas na comunicação, gerando dados incompletos para criação das escalas de serviços, indicação de casais para apresentação dos ensinos bíblicos e a participação das equipes nos acampamentos, o que prejudica o fluxo dos processos da comunidade, ocasionando uma repetição de várias solicitações para que um mesmo processo seja realizado. Para enfrentar esse desafio, a metodologia adotada inclui a implementação de um *monorepo* que faz uso de tecnologias modernas, como Docker, Next JS., Nest JS. e Turbo Repo, entre outras. Essas tecnologias desempenham um papel fundamental para sustentar a agilidade e eficácia do desenvolvimento e operação do sistema. Além disso, a centralização de dados e comunicação é crucial para prevenir falhas e redundâncias, o que, por sua vez, resulta em melhorias significativas no desempenho e na qualidade dos serviços oferecidos pela comunidade.

Palavras-chave: Comunidade. Docker. Escalas. Monorepo. Nest JS. Next JS.

Abstract

The control and management of the data communication and document process is considered a problem in some companies, and it is no different in a group or community. These groups or communities usually grow quickly and sometimes end up losing control of their own media, especially when they adopt mechanisms other than the use of technology. Such action may eventually cause the loss of documents and difficulty in understanding the processes that occurred previously. Thus, the result

[1] Graduando em Análise e Desenvolvimento de Sistemas pela Fatec Dr. Thomaz Novelino – Franca/SP. Endereço eletrônico: lucas.andrade43@fatec.sp.gov.br

[2] Graduando em Análise e Desenvolvimento de Sistemas pela Fatec Dr. Thomaz Novelino – Franca/SP. Endereço eletrônico: pedro.fernandes8@fatec.sp.gov.br

of an interview carried out with some leaders and people from the Tenda Community who reported and presented that communication failures often occur, generating incomplete data for the creation of service schedules, indication of couples to present the biblical teachings and the participation of teams in the camps, which harms the flow of community processes, causing multiple requests to be repeated for the same process to be carried out. To address this challenge, the adopted methodology includes the implementation of a monorepo that uses modern technologies, such as Docker, Next, Nest, and Turbo Repo, among others. These technologies play a fundamental role in supporting the agility and effectiveness of the system development and operation. Furthermore, data and communication centralization are crucial for preventing failures and redundancies, ultimately resulting in significant improvements in the performance and quality of services offered by the community.

Keywords: Community. Docker. Monorepo. Nest JS. Next JS. Scales.

1 Introdução

A Comunidade Tenda é um grupo religioso fundado em 1991, que tem como objetivo oferecer aos seus casais uma formação integral nas dimensões humana, espiritual e apostólica. Atualmente, a comunidade conta com mais de mil casais, que participam de diversos eventos, retiros e encontros promovidos pela organização.

Devido ao grande número de casais e atividades realizadas pela comunidade, a gestão de escalas de serviços destes se tornou uma tarefa complexa e desafiadora. Para garantir uma organização eficiente e transparente, é necessário documentar de forma clara e organizada todas as atividades e tarefas realizadas.

Nesse contexto, a implementação de um sistema para gestão da Comunidade Tenda é essencial para facilitar as escalas e organizar as informações. Esse sistema permitirá uma gestão mais eficiente das atividades, possibilitando um melhor controle de escalas e um acompanhamento mais preciso.

Além disso, um sistema de gestão permitirá a realização de análises mais precisas sobre as atividades da comunidade, permitindo identificar problemas e oportunidades de melhoria. Com a documentação clara e organizada das atividades e tarefas realizadas pelos casais, será possível tomar decisões mais estratégicas e orientadas a dados.

Dessa forma, a implementação de um sistema para a Comunidade Tenda é uma iniciativa relevante e necessária para garantir uma gestão mais eficiente e organizada das atividades realizadas pela organização, e para atender às necessidades de seus casais.

Utilizou-se neste projeto a pesquisa de campo. O procedimento envolveu a coleta de dados por meio de entrevistas com alguns líderes da Comunidade Tenda para compreender suas necessidades específicas de gestão e, posteriormente, a implementação do sistema utilizando a abordagem do framework ágil Scrum e as tecnologias Next JS e Nest JS.

Neste trabalho, o conteúdo está estruturado em seis capítulos distintos. O primeiro capítulo, intitulado Introdução, servirá como o ponto de partida, proporcionando uma visão geral do projeto e de sua importância. No segundo capítulo, será abordada a Viabilidade do Projeto, explorando conceitos como o *Business Model Canvas* (BMC), a Matriz SWOT e o Plano de Ação 5W2H do Projeto.

O terceiro capítulo, Levantamento de Requisitos, é um componente essencial dedicado à elicitação e especificação de requisitos. Este capítulo incluirá tópicos como *Business Process Model and Notation* (BPMN), Requisitos Funcionais e Não Funcionais, Regras de Negócio, Casos de Uso, Diagramas, Matriz de Rastreabilidade e o Modelo Entidade-Relacionamento.

No quarto capítulo, Ferramentas e Métodos, serão apresentadas as ferramentas e metodologias utilizadas ao longo do projeto. O capítulo cinco, denominado desenvolvimento, explorará o processo de construção do sistema, enquanto o capítulo seis, resultados e discussão, se concentrará na análise dos resultados obtidos durante a realização do projeto.

1.1 Termo da Abertura do Projeto (TAP)

A fase de iniciação em gestão de projetos é importante para definir e autorizar um novo projeto ou uma fase adicional de um projeto já existente.

Durante essa etapa, realizou-se processos essenciais, como a definição do escopo inicial do projeto e o comprometimento dos recursos financeiros iniciais. Além disso, são identificadas as partes interessadas internas e externas que terão impacto no projeto, e a seleção do gerente de projetos quando ainda não designado. Esses detalhes cruciais são registrados no termo de abertura do projeto e no registro das partes interessadas.

A aprovação do termo de Abertura do Projeto representa a autorização oficial para o início do projeto, sendo importante ressaltar que a aprovação e o financiamento do projeto ocorrem fora dos limites do próprio projeto.

No contexto de projetos grandes ou complexos, a fase de iniciação pode ser dividida em fases subsequentes, nas quais os processos de iniciação são reexecutados para validar as decisões tomadas anteriormente, como as relacionadas ao termo de abertura do projeto e à identificação das partes interessadas.

Essa abordagem ajuda a manter o foco do projeto na necessidade empresarial original e permite verificar os critérios de sucesso, analisar as influências e objetivos das partes interessadas e tomar decisões sobre a continuidade, adiamento ou interrupção do projeto. É também ressaltado que o envolvimento ativo dos clientes e outras partes interessadas durante a fase de iniciação aumenta a probabilidade de propriedade compartilhada, aceitação da entrega e satisfação geral dos clientes e partes interessadas (PROJECT MANAGEMENT INSTITUTE, p. 144 - 2012).

Por fim, é mencionado que os processos de iniciação podem ser conduzidos por processos organizacionais, de programas ou de portfólios externos ao escopo de controle do projeto, e que o gerente de projetos obtém a autoridade para alocar recursos organizacionais às atividades subsequentes do projeto como parte desse processo de iniciação.

A estrutura delineada neste quadro representa o contexto do projeto corporativo intitulado "Sistenda", cuja demanda e patrocínio são provenientes da Comunidade Tenda. O responsável pela gestão deste projeto é Lucas D’Elia Miranda de Andrade, atuando como gerente, enquanto a equipe de desenvolvimento é composta por Lucas D’Elia Miranda de Andrade e Pedro Henrique Fernandes. Esta configuração reflete a colaboração direta entre a Comunidade Tenda e a equipe responsável pelo desenvolvimento do projeto, evidenciando um alinhamento estreito entre as partes interessadas e os executores das atividades, visando à realização bem-sucedida do empreendimento denominado Sistenda.

Quadro 1 - Identificação do Projeto

Projeto Corporativo: Sistenda
Unidade Idealizadora (demandante): Comunidade Tenda
Patrocinador: Comunidade Tenda

Gerente do projeto: Lucas D’Elia Miranda de Andrade
Desenvolvedores: Lucas D’Elia Miranda de Andrade // Pedro Henrique Fernandes

Fonte: Os autores

A visão geral deste projeto é a implementação de um sistema de gestão para a Comunidade Tenda. O projeto busca atender à necessidade premente de otimizar a gestão de escalas e organizar as informações, tornando-a eficiente e transparente.

Através da aplicação da do framework ágil Scrum e das tecnologias Nest JS e Next JS, o sistema permitirá o controle preciso das atividades e tarefas realizadas pelos casais, facilitando as análises estratégicas e orientadas a dados.

Com uma abordagem focada na colaboração com as partes interessadas internas e externas, o projeto visa aprimorar a administração das atividades da Comunidade Tenda, contribuindo para uma gestão mais eficiente e organizada, além de proporcionar uma experiência satisfatória aos seus líderes.

A justificativa deste projeto reside na necessidade de aprimorar a gestão das atividades da Comunidade Tenda, dada sua complexidade e o grande número de pessoas envolvidas. A implementação de um sistema de gestão se justifica pela demanda por maior eficiência na organização das escalas e na documentação das atividades dos casais.

2 Viabilidade do Projeto

O estudo de viabilidade representa um processo na avaliação e na tomada de decisão sobre a realização de um projeto proposto. É uma análise minuciosa que busca determinar se os recursos disponíveis são adequados para a execução do plano e se os resultados esperados justificam o investimento necessário. Ele não apenas examina a viabilidade técnica e operacional, mas também avalia a viabilidade financeira, respondendo às questões fundamentais sobre a capacidade da equipe em reunir os recursos necessários e se o retorno esperado é proporcional ao investimento. Projetos que envolvem consideráveis investimentos financeiros ou aqueles que têm potencial de impacto significativo no mercado geralmente demandam um estudo de viabilidade para garantir uma tomada de decisão embasada e prudente.

Para gestores de projeto, compreender o estudo de viabilidade é essencial, mesmo que não estejam diretamente encarregados de sua condução. Conhecer os elementos que constituem essa análise permite oferecer um suporte mais eficaz à

equipe responsável, assegurando, assim, um resultado mais assertivo para o projeto. Ao entender a importância de cada componente avaliado no estudo de viabilidade, os gestores podem contribuir com informações valiosas e tomar decisões mais embasadas durante o processo de planejamento e execução do projeto, visando alcançar os melhores resultados possíveis (ASANA, [s.d.], *online*).

2.1 *Business Model Canvas (*BMC)

O BMC, uma ferramenta de planejamento estratégico, foi concebido para desmembrar uma ideia de negócio em diversos componentes, permitindo uma visão completa do modelo de negócios.

A abordagem visual do Canvas, influenciada pelo *design thinking*, o torna intuitivo e fácil de compreender à primeira vista. Inicialmente proposto por Alex Osterwalder em sua tese de doutorado, o Canvas evoluiu por meio da colaboração de mais de 200 consultores globais.

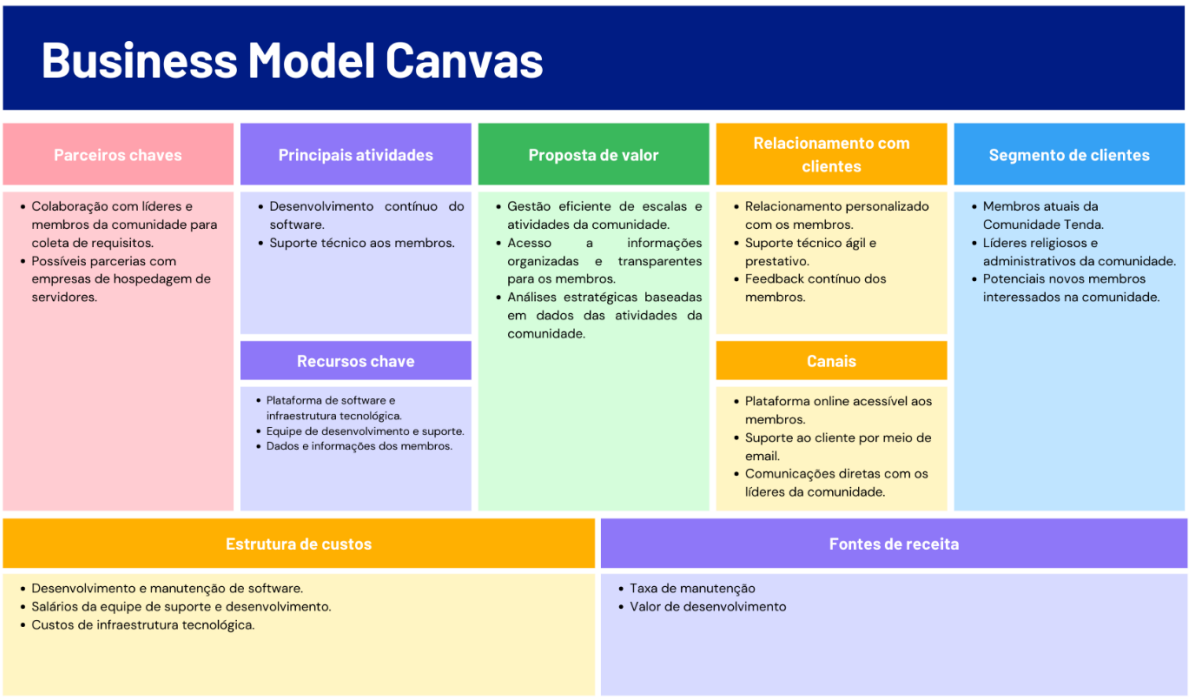
Seus nove campos representam os principais pilares de um negócio, preenchidos com anotações que proporcionam uma visão holística e flexível do modelo, revelando diferenciais competitivos e possíveis melhorias para aumentar a eficiência.

É importante destacar a diferença entre um modelo de negócios e um plano de negócios. O modelo de negócios concentra-se na criação, entrega e captura de valor no mercado, oferecendo uma visão estruturada e simplificada do funcionamento da empresa.

Por outro lado, o plano de negócios é um documento mais abrangente, reunindo todas as informações da organização, obtendo investimentos, formalizar a empresa e executar um planejamento detalhado (“O que é Business Model Canvas e como aplicá-lo no seu negócio?”, 2023).

Na sequência, apresentamos o BPMN correspondente ao sistema em análise.

Figura 1 – Business Model Canvas



Fonte: Os autores, 2023

O bloco de identifica os principais grupos de pessoas que se relacionarão com o sistema de gestão de casais da Comunidade Tenda. Os atuais casais, líderes e potenciais novos membros são os principais *stakeholders*.

A proposta de valor descreve os benefícios que o sistema oferecerá aos envolvidos. No caso, são melhorias na gestão, acesso a informações e análises estratégicas.

Os canais descrevem como a comunicação e a entrega da proposta de valor ocorrerão. Neste caso, inclui uma plataforma online, suporte aos casais através de e-mail e comunicação direta com os líderes.

O relacionamento descreve como a comunidade se envolverá com os casais. Aqui, é enfatizada a importância de um relacionamento personalizado, suporte técnico eficaz e *feedback* constante.

O bloco de fontes de receita identifica como o projeto gera receita, principalmente por meio do valor de desenvolvimento e das taxas de manutenções.

Recursos-chave são os elementos essenciais para a operação do projeto, como a plataforma de *software*, equipe de suporte e dados dos casais.

O bloco de atividades chave enumera as atividades-chave necessárias para entregar a proposta de valor e manter o sistema funcionando, incluindo

desenvolvimento, suporte e análise de dados.

Parcerias-chave destacam as colaborações essenciais para o projeto, incluindo a colaboração com a comunidade para coletar requisitos e possíveis parcerias de hospedagem.

A estrutura de custos lista os principais gastos do projeto, como desenvolvimento de *software*, salários da equipe e custos de infraestrutura tecnológica.

2.2 Matriz SWOT

A matriz *Strengths Weaknesses Opportunities Threats* (SWOT) é uma ferramenta de análise estratégica utilizada para avaliar as forças, fraquezas, oportunidades e ameaças de uma empresa ou organização. Essa análise é feita através da identificação de fatores internos (forças e fraquezas) e externos (oportunidades e ameaças) que podem afetar o desempenho da empresa (PRADA, 2020, *online*).

A importância da matriz SWOT reside na sua capacidade de fornecer uma visão geral das condições internas e externas que afetam uma empresa ou organização. Isso ajuda a empresa a identificar seus pontos fortes e fracos, bem como as oportunidades e ameaças que enfrenta (PRADA, 2020, *online*).

Com base nessas informações, a empresa pode desenvolver estratégias para aproveitar suas forças, superar suas fraquezas, explorar oportunidades e minimizar ameaças. A matriz SWOT é, portanto, uma ferramenta útil para a tomada de decisões estratégicas e para o planejamento empresarial (PRADA, 2020, *online*).

Na sequência, apresentamos a Matriz SWOT correspondente ao sistema em análise.

Figura 2 - Matriz SWOT

	Fatores Positivos	Fatores Negativos
Fatores internos	Comunidade engajada Comunidade em crescimento Bem estruturada	Gestão de membros Falta de confiabilidade dos dados Dificuldade na integração dos dados Baixa disponibilidade dos dados Falta de atomicidade nos dados
Fatores externos	Alto potencial de crescimento Parcerias	Diminuição de eventos Dificuldade da captação de investimentos Baixa visibilidade fora do meio religioso

Fonte: Os autores, 2023

Analisando a matriz SWOT, percebemos que a criação de um sistema de gestão é uma iniciativa altamente benéfica para a Comunidade Tenda.

O sistema oferece uma solução para várias fraquezas identificadas, como a complexidade da gestão de casais, a falta de confiabilidade dos dados e a dificuldade na integração de informações. Isso permitirá à equipe responsável pela gestão de se concentrar em atividades estratégicas, como planejamento de eventos e desenvolvimento de conteúdo.

Além disso, a disponibilidade de dados e a confiabilidade das informações melhorarão significativamente, possibilitando a geração de relatórios personalizados para avaliação e ajuste de atividades.

Dessa forma, o sistema atua como uma ferramenta para otimizar as operações da comunidade e promover um crescimento mais eficiente, mesmo em um contexto de diminuição de eventos e dificuldade na captação de investimentos.

A necessidade de expandir a visibilidade da Comunidade Tenda além do meio religioso também é abordada, já que o sistema pode auxiliar na coleta de dados estratégicos para a tomada de decisões informadas e na criação de estratégias de promoções mais eficazes.

Portanto, o desenvolvimento do sistema não apenas aborda as fraquezas identificadas, mas também abre portas para novas oportunidades e aprimoramentos em uma perspectiva de longo prazo. Com a base sólida estabelecida, o projeto está preparado para buscar soluções inovadoras e superar os desafios futuros com confiança.

2.3 Plano de Ação 5W2H do Projeto

O plano de ação *What Why Who When Where How How Much* (5W2H), também conhecida como plano de ação, é uma ferramenta versátil que se destina a qualquer pessoa que precise colocar um plano em ação. Sua utilidade reside na capacidade de responder às sete perguntas fundamentais: o que, por que, quem, onde, quando, como e quanto, que guiam a implementação do plano.

Ela pode ser aplicada de várias formas, desde situações simples, como adquirir um novo equipamento, até planos de negócios táticos e operacionais. O 5W2H ajuda a tomar decisões informadas sobre a execução das ações, fornecendo clareza sobre as responsabilidades, custos e processos envolvidos.

Além disso, é particularmente eficaz quando combinada com outras ferramentas analíticas, como SWOT, *Boston Consulting Group* (BCG) ou análises de forças de mercado, para garantir que as estratégias e planos sejam efetivamente implementados, transformando oportunidades em resultados tangíveis.

Em cenários complexos, como a gestão de projetos ou a elaboração de planos de negócios abrangentes, o 5W2H desempenha é utilizado na orientação da implementação das decisões contidas nos documentos. Ela ajuda a traduzir os *insights* estratégicos em ações concretas, destacando quem é responsável por cada etapa, onde e quando essas ações serão executadas, como serão conduzidas e qual será o investimento necessário.

Portanto, o 5W2H é uma ferramenta essencial para garantir que os planos sejam concretizados com sucesso, tornando-se uma aliada valiosa para transformar ideias em realidade (NAKAGAWA, [s.d.], *online*).

Na sequência, apresentamos a Matriz 5W2H correspondente ao sistema em análise.

Figura 3 - Matriz 5W2H

Ação	O quê? (What?)	Por quê? (Why?)	Onde? (Where?)	Quem? (Who?)	Quando? (When?)	Como? (How?)	Quanto? (How much?)
1	Criar software para gerenciamento de pessoas.	Melhorar gestão de escala dos membros.	Em qualquer lugar com conexão a internet	Membros Fundadores e guardiões.	24 horas por dia, 7 dias na semana.	Desenvolvimento de software com base na elicitação de requisitos	Valor do projeto
2	Desenvolver software seguindo padrões de segurança de dados(LGPD).	Garantir que a gestão de escalas siga as regras de negocios.	Em qualquer lugar com conexão a internet	Membros Fundadores e guardiões.	24 horas por dia, 7 dias na semana.	Desenvolvimento de software com base na elicitação de requisitos e ISO 27001	Valor do projeto
3	Gerenciar integração e disponibilidade de dados	Para garantir a integridade e disponibilidade dos dados.	Em qualquer lugar com conexão a internet	Todos os membros da comunidade.	24 horas por dia, 7 dias na semana.	Desenvolvimento de software com base na elicitação de requisitos.	Valor do projeto

Fonte: Os autores, 2023

A matriz mostrada apresenta três pontos relacionados à criação de um software para gerenciamento de pessoas, onde o objetivo é melhorar a gestão de escala dos casais, garantir a segurança dos dados e disponibilizar o *software* para todas as paróquias. Esses pontos são:

- O primeiro ponto destaca a criação do *software* com base na elicitação de requisitos, indicando que o objetivo é melhorar a gestão de escala, que provavelmente está sendo feita de forma manual ou em sistemas não tão eficientes.
- O segundo ponto destaca a importância de desenvolver o *software* seguindo padrões de segurança de dados (LGPD) e ISO 27002, para garantir a segurança das informações e proteger os dados dos casais.

- O terceiro ponto apresenta a intenção de disponibilizar o *software* para todas as paróquias, com o objetivo de garantir a integridade e disponibilidade dos dados para todos os casais da comunidade.

Em resumo, a matriz 5W2H destaca a importância do *software* para melhorar a gestão de escala, garantir a segurança dos dados e disponibilizá-lo para todas as paróquias.

3 Levantamento de Requisitos

O levantamento de requisitos representa a fase inicial no processo de desenvolvimento de software, focado na compreensão e identificação das necessidades do cliente que o sistema a ser desenvolvido deve atender. Essa etapa visa definir as funcionalidades essenciais e delimitar o escopo do projeto, sendo conduzida pelo Analista de Requisitos.

Um dos maiores desafios enfrentados durante esse processo é a comunicação efetiva entre o cliente e o analista. Muitas vezes, o cliente não é o usuário direto do sistema, o que pode gerar uma lacuna na compreensão das necessidades reais. Isso pode resultar em uma descrição imprecisa dos requisitos, baseada em observações individuais que podem não representar fielmente as perspectivas dos usuários finais.

Para evitar omissões ou interpretações inadequadas, o Analista de Requisitos deve adotar uma abordagem abrangente, considerando todos os possíveis cenários de uso do sistema. Ao fazer isso, busca-se garantir uma compreensão mais completa das necessidades dos usuários e reduzir a probabilidade de erros durante a implementação das funcionalidades. Essa compreensão detalhada é essencial para o desenvolvimento de um software que atenda efetivamente às expectativas e necessidades do cliente (PRECIOSO, 2018, *online*).

3.1 Elicitação e especificação dos Requisitos

No processo de elicitação de requisitos para o projeto da Comunidade Tenda, foi empregada a técnica de questionário aplicado aos líderes e casais mais antigos da comunidade. Essa técnica de elicitação é valiosa para obter uma compreensão aprofundada das dores, necessidades e expectativas, especialmente daqueles que têm uma perspectiva histórica e ampla sobre a comunidade. A utilização da entrevista permite coletar informações de forma estruturada e documentada, facilitando a análise posterior e a identificação de padrões nas respostas.

A aplicação aos casais mais antigos da comunidade oferece a vantagem de capturar a experiência acumulada ao longo do tempo, bem como as nuances das atividades e demandas específicas que podem não ser evidentes para casais mais recentes.

Além disso, essa técnica de elicitação ajuda a garantir que as necessidades da comunidade sejam consideradas de maneira abrangente e que o sistema de gestão seja desenvolvido de acordo com as expectativas e requisitos dos envolvidos. Portanto, o uso da entrevista como parte da elicitação de requisitos é uma abordagem eficaz para reunir informações valiosas que orientarão o desenvolvimento bem-sucedido do projeto.

A elicitação de requisitos desempenha um papel de extrema importância em projetos de software, pois é a fase inicial onde se busca entender e documentar de forma precisa os objetivos a serem alcançados. Muitos dos fracassos em projetos de desenvolvimento podem ser atribuídos a falhas na análise e especificação dos requisitos, devido à comunicação inadequada entre as partes envolvidas.

Além disso, requisitos mal levantados podem resultar em impactos significativos, atrasos e retrabalho. Portanto, investir em processos bem definidos, documentação detalhada, equipe qualificada e ferramentas apropriadas para a elicitação de requisitos pode contribuir para o sucesso dos projetos de desenvolvimento de software, garantindo uma compreensão clara das necessidades dos clientes e, consequentemente, resultados satisfatórios no final das atividades propostas (DEVMEDIA – Engenharia de Requisitos, [s.d.], *online*).

As perguntas e respostas relacionadas ao projeto foram detalhadas e estão disponíveis no apêndice. Esse recurso oferece uma maneira organizada e acessível de acessar informações adicionais e complementares que podem ser úteis para uma compreensão mais aprofundada do projeto e de suas partes integrantes. Portanto, para mais informações específicas sobre as perguntas e respostas relacionadas ao projeto, é recomendável consultar o apêndice, onde esses detalhes foram registrados de forma clara e concisa.

3.2 BPMN

A BPMN atua na gestão de processos de negócios, oferecendo uma abordagem visual e detalhada para representar as etapas de um processo de ponta a ponta. Essa notação descreve com clareza as atividades de negócios e os fluxos de

informação necessários para a conclusão bem-sucedida de um processo específico.

Além de sua função primordial na documentação e representação de processos, o BPMN tem evoluído ao longo dos anos, passando por novas padronizações e permanecendo uma ferramenta fundamental para modelar formas de aprimorar a eficiência operacional, adaptar-se a novas circunstâncias de negócios e ganhar vantagem competitiva. Vale ressaltar que o BPMN não se limita ao mapeamento de processos atuais; ele também se estende à modelagem de processos futuros e à tomada de decisões estratégicas para otimização de operações.

Enquanto o BPMN se destaca como uma ferramenta versátil de gestão de processos de negócios, ele difere do mapeamento de processos atuais, que tem um foco mais restrito em padronização, treinamento, qualidade e conformidade.

Além disso, o BPMN possui uma relação estreita com a *Unified Modelling Language* (UML), que é amplamente utilizada no design de software. Essa conexão entre BPMN e UML permite uma integração eficaz entre a modelagem de processos de negócios e o design de sistemas de software, facilitando a comunicação e a colaboração entre as equipes de negócios e tecnologia.

O BPMN, do alto nível ao detalhado, desempenha um papel fundamental na melhoria da compreensão e comunicação dos processos de negócios. Ela atende tanto às partes interessadas que buscam uma representação visual prática das etapas quanto àqueles envolvidos na implementação precisa desses processos.

A sua utilidade abrange uma ampla gama de profissionais, incluindo analistas de negócios, membros da equipe de processos, gerentes, desenvolvedores técnicos e consultores externos, proporcionando uma linguagem comum que transcende barreiras técnicas.

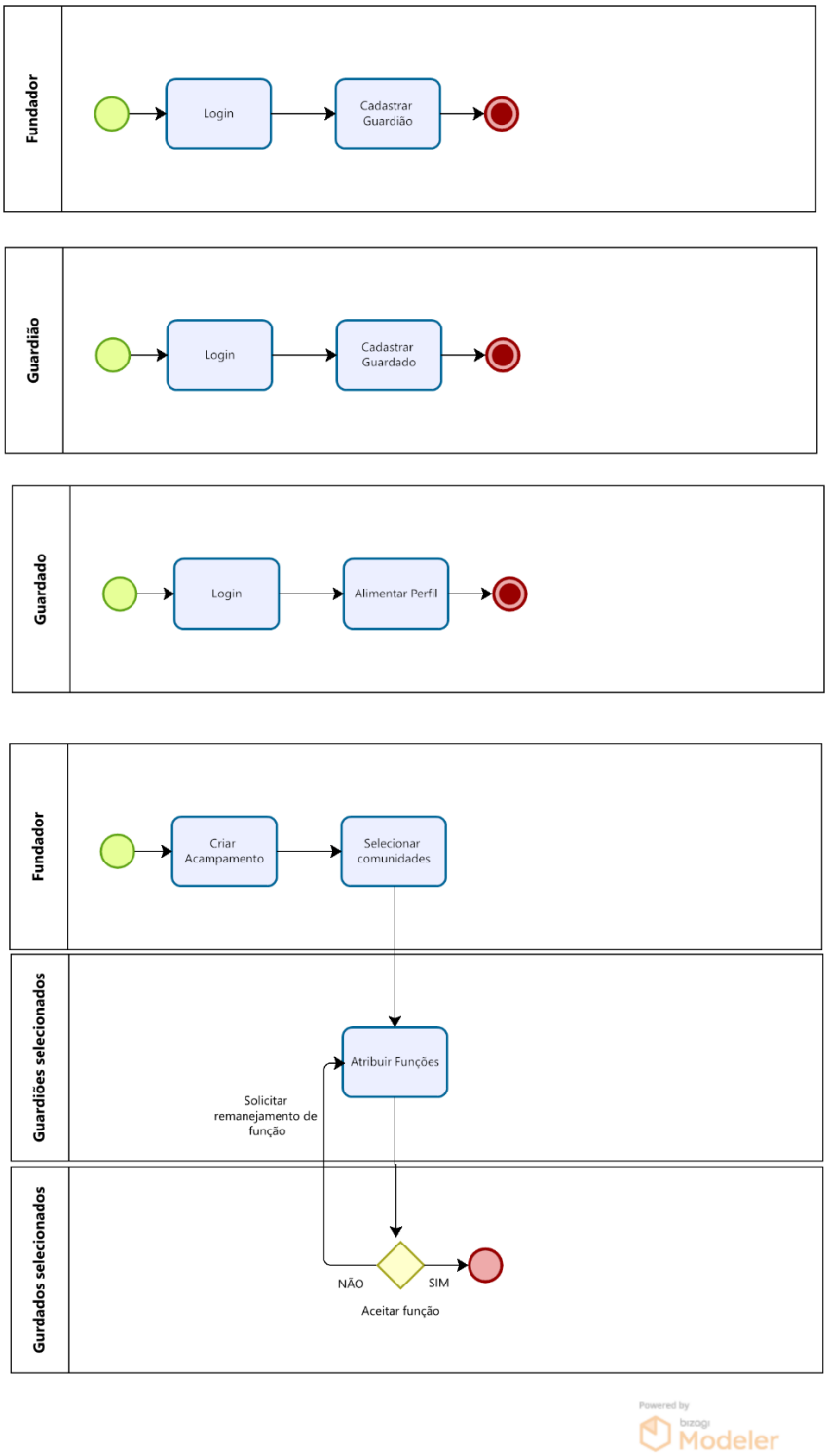
Ao preencher a lacuna entre a concepção e a implementação dos processos, o BPMN oferece detalhes claros sobre a sequência de atividades empresariais, tornando a comunicação mais eficaz e colaborativa.

Além disso, essa notação facilita a integração com documentos XML, essenciais para a execução de diversos processos, incluindo aqueles definidos em padrões como o *Business Process Execution Language* (BPEL), que atua na orquestração de serviços da web (LUCIDCHART - BPMN, [s.d.], *online*).

Portanto, o BPMN é uma ferramenta poderosa para aprimorar a eficiência e qualidade dos processos de negócios.

Na sequência, apresentamos o BPMN correspondente ao sistema em análise.

Figura 4 - BPMN



Fonte: Os autores, 2023

O BPMN apresentado na figura acima demonstra os quatro fluxos principais para o *software*. Esses pontos são:

- O primeiro ponto mostra o processo para cadastrar um novo membro

guardião, que é executado por um membro fundador.

- O segundo ponto mostra o processo para cadastrar um novo membro guardado, que é executado por um membro guardião.
- O terceiro ponto mostra o processo para o guardado alimentar o perfil com seus dados pessoais que o guardião não inseriu.
- O quarto ponto apresenta o processo de atribuir funções aos membros que é iniciado por um membro fundador selecionando as comunidades que participarão do acampamento, em seguida um membro guardião atribui funções as funções aos seus guardados e por fim os membros guardados aceitam ou recusam suas funções.

3.3 Requisitos Funcionais

Requisitos funcionais são elementos essenciais na definição de sistemas de software, delineando as funções específicas que o sistema ou seus componentes devem executar. Em termos simples, eles descrevem o comportamento esperado do sistema, destacando os processos ou transformações que os componentes de software ou hardware realizarão nas entradas para produzir as saídas desejadas.

Esses requisitos funcionais têm um enfoque no usuário, capturando as funcionalidades que serão percebidas e utilizadas por aqueles que interagem com o sistema. Portanto, eles servem como uma base para o desenvolvimento e teste de sistemas de software, assegurando que as expectativas dos usuários sejam atendidas de forma eficaz e precisa (DEVMEDIA – Engenharia de Requisitos, [s.d.], *online*).

Na sequência, apresentamos o quadro de Requisitos Funcionais correspondente ao sistema em análise.

Quadro 2 – Requisitos Funcionais

ID: RF001	**Nome:** Acessar página acampamento.
Categoria: Evidente	**Prioridades:** Altíssima.
Descrição:	O sistema deve permitir acessar a página acampamento de acordo com o tipo de usuário.
Informações:	Nome do acampamento, data do acampamento, função.
Regras de Negócio:	**RN007** – Selecionar comunidades. Só serão permitidos membros Fundadores criarem acampamentos.

Regras de Negócio:	**RN003** – Cadastro de membros Guardados. Só serão permitidos os usuários Guardiões cadastrar membros Guardados.
ID: RF002	**Nome:** Alterar dados pessoais.
Categoria: Evidente	**Prioridades:** Altíssima.
Descrição:	O sistema deve permitir alterar os dados pessoais, caso o usuário já esteja cadastrado no sistema.
Informações:	Número de celular, Senha, Nome, Casal, Idade, se possui problema de saúde.
Regras de Negócio:	**RN003** – Cadastro de membros Guardados. Só serão permitidos os usuários Guardiões cadastrar membros Guardados.
ID: RF003	**Nome:** Atribuir função e acampamento.
Categoria: Evidente	**Prioridades:** Altíssima.
Descrição:	O sistema deve permitir apenas os membros Fundadores criar acampamentos e atribuir funções aos guardados nesses acampamentos.
Informações:	Nome do acampamento, data, comunidades que farão parte, funções.
Regras de Negócio:	**RN006** - Só serão permitidos membros fundadores atribuírem as funções dos guardados nos acampamentos e os guardados aceitar ou recusar essas funções. Sendo desejável que o guardado alterne entre as funções.
ID: RF004	**Nome:** Aceitar função**.**
Categoria: Oculta	**Prioridades:** Altíssima.
Descrição:	O sistema deve permitir que os membros guardados escolham entre aceitar ou recusar a função atribuída.
Informações:	Função atribuída, data do acampamento, aceite ou recusa.
Regras de Negócio:	**RN006** - Só serão permitidos membros fundadores atribuírem as funções dos guardados nos acampamentos e os guardados aceitar ou recusar essas funções. Sendo desejável que o guardado alterne entre as funções.
ID: RF005	**Nome:** Retornar resposta de recusa ao fundador.

Categoria: Oculta	**Prioridades:** Alta.
Descrição:	No caso de o guardado recusar a função atribuída a ele no acampamento o sistema deve retornar ao fundador responsável o nome do guardado pedindo reajuste e motivo.
Informações:	Nome do guardado e motivo da recusa, nova função.
Regras de Negócio:	**RN006** - Só serão permitidos membros fundadores atribuírem as funções dos guardados nos acampamentos e os guardados aceitar ou recusar essas funções. Sendo desejável que o guardado alterne entre as funções
ID: RF006	**Nome:** Notificar guardado de acampamento e função atribuída.
Categoria: Evidente	**Prioridades:** Alta.
Descrição:	O sistema deve informar ao guardado quando ele for atribuído a alguma função e a data do acampamento.
Informações:	Função, data do acampamento nome do acampamento.
Regras de Negócio:	**RN006** - Só serão permitidos membros fundadores atribuírem as funções dos guardados nos acampamentos e os guardados aceitar ou recusar essas funções. Sendo desejável que o guardado alterne entre as funções.
ID: RF007	**Nome:** Efetuar Login.
Categoria: Evidente	**Prioridades:** Altíssima.
Descrição:	O sistema deve permitir efetuar o login após o cadastro ter sido feito.
Informações:	Número celular, senha.
Regras de Negócio:	**RN009** - Só serão permitidos acessar o sistema os usuários com cadastro, informando o número do celular pessoal e a senha.
ID: RF008	**Nome:** Acessar a Home.
Categoria: Evidente	**Prioridades:** Altíssima.
Descrição:	O sistema deve permitir acessar a home após efetuar o login.
Informações:	Login response, acampamentos que já participou, funções, datas.

Regras de Negócio:	**RN009** - Só serão permitidos acessar o sistema os usuários com cadastro, informando o número do celular pessoal e a senha.
ID: RF009	**Nome:** Acessar tela de cadastro de novos membros.
Categoria: Evidente	**Prioridades:** Altíssima.
Descrição:	O sistema deve permitir os membros admin, fundadores e guardiões acessem a tela cadastrar novos membros.
Informações:	CPF, Celular pessoal e nome, comunidade, nível de acesso, nome do novo membro e seu cônjuge, celular, CPF.
Regras de Negócio:	**RN001** - Só serão permitidos os usuários administradores cadastrarem membros Fundadores. **RN002 -** Só serão permitidos os usuários fundadores cadastrarem membros **Guardiões**, Cadastro de membros Guardados. **RN003 -** Só serão permitidos os usuários Guardiões cadastrar membros Guardados.
ID: RF010	**Nome:** Ver acampamentos.
Categoria: Evidente	**Prioridades:** Altíssima.
Descrição:	O sistema deve permitir que os usuários vejam os acampamentos que já participaram, que estão atribuídos, e suas funções em cada um.
Informações:	Nome acampamento, data, função.
Regras de Negócio:	**RN006** - Só serão permitidos membros fundadores atribuírem as funções dos guardados nos acampamentos e os guardados aceitar ou recusar essas funções. Sendo desejável que o guardado alterne entre as funções.
ID: RF011	**Nome:** Ver comunidade.
Categoria: Evidente	**Prioridades:** Alta.
Descrição:	O sistema deve permitir que o usuário veja os dados da comunidade que faz parte.
Informações:	Nome da comunidade, membros, celular dos membros.

Regras de Negócio:	**RN001** - Só serão permitidos os usuários administradores cadastrarem membros Fundadores. **RN002 -** Só serão permitidos os usuários fundadores cadastrarem membros **Guardiões**, Cadastro de membros Guardados. **RN003 -** Só serão permitidos os usuários Guardiões cadastrar membros Guardados.

Fonte: Os autores, 2023

3.4 Requisitos Não Funcionais

Na engenharia de sistemas de software, os requisitos não funcionais desempenham um papel fundamental ao descreverem não apenas o que o sistema fará, mas principalmente como ele o fará. Isso inclui requisitos relacionados ao desempenho, à interface externa, às restrições de projeto e aos atributos de qualidade do sistema. A avaliação desses requisitos não funcionais abrange tanto testes práticos quanto avaliações subjetivas.

Esses requisitos são igualmente cruciais no desenvolvimento de um sistema de software, desempenhando um papel significativo na seleção e na composição da arquitetura de software, influenciando as decisões de projeto à medida que os sistemas se tornam mais complexos. É amplamente reconhecido pela comunidade de arquitetura de software que o suporte a requisitos não funcionais depende cada vez mais de decisões cuidadosas tomadas durante a fase de projeto da arquitetura, sendo essa uma perspectiva compartilhada por profissionais da área.

Dessa forma, os requisitos não funcionais, também conhecidos como atributos de qualidade, são elementos cruciais para garantir que um sistema de software atenda não apenas ao que os usuários esperam em termos de funcionalidade, mas também a critérios essenciais relacionados à eficiência, confiabilidade, usabilidade e outros aspectos que impactam diretamente na experiência do usuário e na qualidade global do sistema (DEVMEDIA – Engenharia de Requisitos não Funcionais, [s.d.], *online*).

Na sequência, apresentamos o quadro de Requisitos Não Funcionais correspondente ao sistema em análise.

Quadro 3 - Requisitos não funcionais

ID: RNF001	**Nome:** Plataforma web
Tipo: Plataforma	**Prioridades:** Obrigatório
Descrição:	O sistema deve ser executável em plataforma web.
Regras de Negócio:	
ID: RNF002	**Nome:** Segurança do Acesso
Tipo: Segurança	**Prioridades:** Obrigatório
Descrição:	O sistema deve garantir a autenticação e autorização adequadas, permitindo apenas que administradores cadastrem membros fundadores (RN001). O acesso à funcionalidade de cadastro de membros guardiões devem ser restrito aos usuários fundadores (RN002) e membros guardiões (RN003).
Regras de Negócio:	RN001 - Só serão permitidos os usuários administradores cadastrarem membros Fundadores. RN002 - Só serão permitidos os usuários fundadores cadastrarem membros Guardiões, Cadastro de membros Guardados. RN003 - Só serão permitidos os usuários Guardiões cadastrar membros Guardados.
ID: RNF003	**Nome:** LGPD
Tipo: Segurança	**Prioridades:** Obrigatório
Descrição:	Devido ao acesso a dados sensíveis o sistema deve ser construído de acordo com as normas LGPD e boas práticas de segurança de dados seguindo a ISO 27002
Regras de Negócio:	
ID: RNF004	**Nome:** Disponibilidade
Tipo: Disponibilidade	**Prioridades:** Obrigatória
Descrição:	O sistema deve estar disponível 24h por dia.
Regras de Negócio:	
ID: RNF005	**Nome:** Performance e Escalabilidade
Tipo: Desempenho	**Prioridades:** Obrigatória

Descrição:	O sistema deve ser capaz de lidar com muitos membros e casais, garantindo um desempenho eficiente na criação e manutenção de registros de membros e casais (RN004).
Regras de Negócio:	RN004 – Ao cadastrar membros o sistema criará a entidade casal, que será a entidade que relacionará o casal a sua etapa, suas funções e, se houver, filhos.
ID: RNF006	**Nome:** Gestão de Acampamentos
Tipo: Funcional	**Prioridades:** Obrigatória
Descrição:	Somente membros fundadores devem ter permissão para criar acampamentos (RN005).
Regras de Negócio:	RN005 - Só serão permitidos membros fundadores criarem acampamentos
ID: RNF007	**Nome:** Atribuição de Funções
Tipo: Funcional	**Prioridades:** Obrigatória
Descrição:	O sistema deve fornecer uma interface eficaz para atribuição de funções nos acampamentos e para que os guardados aceitem ou recusem essas funções (RN006). Deve ser possível alternar entre funções de guardados nos acampamentos.
Regras de Negócio:	RN006 - Só serão permitidos membros fundadores atribuírem as funções dos guardados nos acampamentos e os guardados aceitar ou recusar essas funções. Sendo desejável que o guardado alterne entre as funções.
ID: RNF004	**Nome:** Privacidade e Segurança de Dados
Tipo: Segurança	**Prioridades:** Obrigatória
Descrição:	Os membros devem ter a capacidade de alterar seus próprios dados pessoais e os dados de seus filhos, garantindo a privacidade e segurança das informações (RN008). O sistema deve implementar medidas de segurança adequadas para proteger informações sensíveis dos membros.
Regras de Negócio:	RN008 - Só serão permitidos os membros alterarem os próprios dados ou de seus filhos

Fonte: Os autores, 2023

3.5 Regras de Negócio

Regras de negócios são elementos cruciais para o funcionamento eficaz de uma organização, representando decisões sobre como conduzir e executar as atividades comerciais.

Elas fornecem diretrizes claras e diretas que definem o comportamento e a estrutura de processos específicos dentro do contexto de um negócio. Cada regra de negócio abrange situações particulares e estabelece o resultado esperado para ações ou decisões tomadas em cenários específicos.

Assim, as regras de negócios desempenham um papel fundamental no delineamento das operações de uma empresa, garantindo consistência, conformidade e eficiência em suas práticas, e são essenciais para o sucesso de projetos de desenvolvimento de sistemas de informação.

A importância das regras de negócios transcende a simples organização de processos; elas atuam na otimização e no controle das operações de uma empresa.

A padronização promovida pelas regras de negócios contribui significativamente para a fluidez e a automação dos processos, eliminando etapas repetitivas e reduzindo o desperdício de tempo e recursos.

Essa eficiência resultante não apenas agiliza as operações, mas também melhora a qualidade dos produtos ou serviços oferecidos, ao mesmo tempo em que reduz custos operacionais.

Além disso, as regras de negócios desempenham um papel crítico no controle e na detecção precoce de falhas nos processos, permitindo correções ágeis quando algo foge das diretrizes estabelecidas. Elas também oferecem suporte à tomada de decisões estratégicas e à execução de estratégias pré-definidas, contribuindo para o alinhamento dos objetivos da empresa.

No contexto do BPM (*Business Process Management*), as regras de negócios, quando aplicadas de forma adequada e mensuradas, têm o potencial de impulsionar o desempenho organizacional, reduzindo custos e agregando maior valor aos clientes.

No entanto, para colher esses benefícios, a clareza na comunicação interna e a transparência nos fluxos de atividades são essenciais, garantindo que as regras sejam aplicadas corretamente ao longo de todo o processo, evitando falhas que possam prejudicar a entrega final ao cliente (DALLAVALLE; CAZARINI, 2000).

Na sequência, apresentamos o quadro das Regras de Negócio correspondente ao sistema em análise.

Quadro 3- Regras de negócio

RN001 – Cadastro de membros Fundadores
Descrição: Só serão permitidos os usuários administradores cadastrarem membros fundadores.
RN002 - Cadastro de membros Guardiões
Descrição: Só serão permitidos os usuários fundadores cadastrarem membros guardiões.
RN003 - Cadastro de membros Guardados
Descrição: Só serão permitidos os usuários fundadores cadastrarem membros guardiões.

RN004 - Relacionamento Casal
Descrição: Ao cadastrar membros o sistema criará a entidade casal, que será a entidade que relacionará o casal a sua etapa, suas funções e, se houver, filhos.
RN005 - Criação acampamento
Descrição: Só serão permitidos membros fundadores criarem acampamentos
RN006 - Atribuir funções
Descrição: Só serão permitidos membros fundadores atribuírem as funções dos guardados nos acampamentos e os guardados aceitar ou recusar essas funções. Sendo desejável que o guardado alterne entre as funções.
RN007 - Selecionar comunidades
Descrição: Só serão permitidos membros fundadores selecionarem as comunidades que participarão do acampamento
RN008 - Alterar dados pessoais
Descrição: Só serão permitidos os membros alterarem os próprios dados ou de seus filhos

Fonte: Os autores, 2023

3.6 Casos de Uso

Na UML, um diagrama de caso de uso é uma representação que condensa as informações relativas aos usuários de um sistema, também chamados de atores, e suas interações com o sistema em questão.

Através de símbolos e conectores especializados, esse diagrama permite a visualização e discussão de cenários nos quais o sistema interage com pessoas, organizações ou sistemas externos. Além disso, ele auxilia na compreensão das metas que o sistema ajuda os atores a alcançarem e na definição clara do escopo do sistema, fornecendo um meio eficaz de documentar e comunicar as funcionalidades e interações envolvidas no projeto.

O uso de diagramas de caso de uso na UML é uma prática valiosa para o desenvolvimento de sistemas, contribuindo para uma melhor compreensão e planejamento das interações entre o sistema e seus usuários.

A importância do caso de uso reside na sua capacidade de representar de forma abrangente e organizada as interações entre os diferentes tipos de usuários e o sistema, fornecendo uma visão clara das metas dos usuários e do comportamento necessário do sistema para atingir essas metas.

Ele atua como um unificador ao longo de todo o ciclo de desenvolvimento do sistema, desempenhando papéis cruciais em várias etapas. Primeiramente, o caso de uso serve como a principal especificação dos requisitos funcionais do sistema,

proporcionando uma base sólida para a análise e o design.

Além disso, ele é atua no planejamento de iterações e na definição de casos de teste, garantindo que o sistema atenda às expectativas dos usuários (LUCIDCHART – Caso de uso UML, [s.d.], *online*).

Por último, mas não menos importante, o caso de uso também desempenha um papel importante na documentação dos usuários, tornando mais fácil para os usuários finais compreenderem como o sistema funciona e como podem interagir com ele de maneira eficaz.

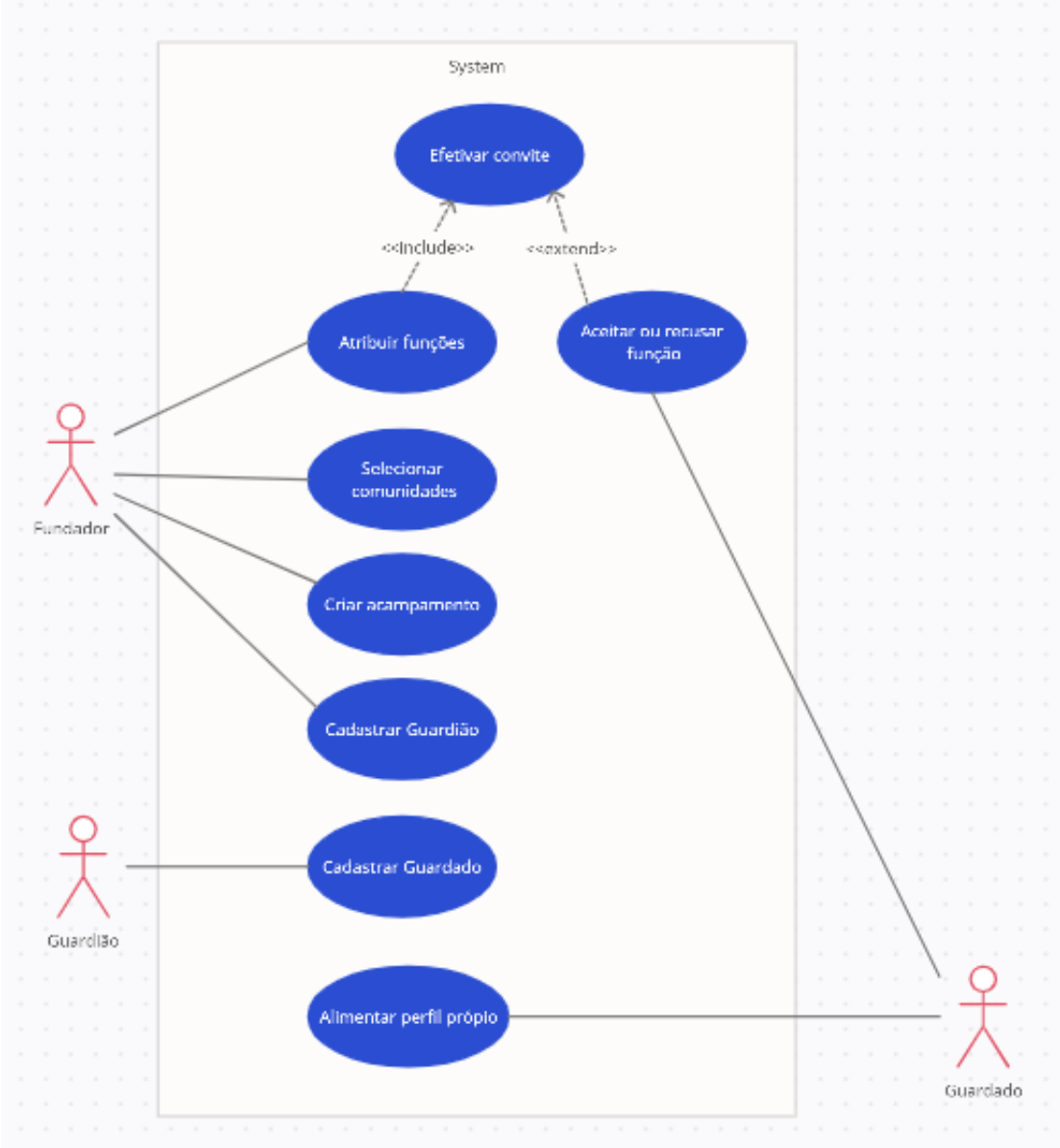
Além disso, a capacidade de estruturar o modelo de caso de uso em pacotes simplifica a análise, a comunicação, a navegação, o desenvolvimento, a manutenção e o planejamento do sistema.

Essa estrutura organizacional facilita a compreensão e a gestão do sistema, permitindo que os diferentes componentes sejam abordados de forma mais eficiente.

Portanto, o caso de uso não apenas desempenha um papel central na definição e na representação dos requisitos do sistema, mas também contribui significativamente para o sucesso global do projeto de desenvolvimento, garantindo a clareza, a coesão e a consistência ao longo de todo o processo (“Conceito: Modelo de Caso de Uso”, [s.d.], *online*).

Na sequência, apresentamos o Diagrama de Caso de Uso correspondente ao sistema em análise.

Figura 5 - Diagrama de Caso de Uso



Fonte: Os autores, 2023

Sua documentação também é essencial para demostrar o cenário no qual vai ser usado, juntamente com as regras em que o sistema deverá seguir com outras opções por meio da escolha do usuário.

Serão descritos os eventos necessários para efetuar as ações no cenário principal e alternativo, considerando pré-condições ou aquelas que forem posteriores a sua execução.

Novamente são destacados os atores, parâmetros a que devem ser fornecidos, restrições e validações.

Na sequência, apresentamos o quadro dos Casos de Uso correspondente ao sistema em análise.

Quadro 4 - Caso de uso: Realizar cadastro do guardião

Caso de Uso – Realizar Cadastro Guardião	
ID	UC 001
Descrição	Este caso de uso tem por objetivo que membros fundadores possam cadastrar membros guardiões
Ator Primário	Usuário do sistema (Membro fundador)
Pré-condição	Membro fundador estar cadastrado
Cenário Principal	1. O use case inicia quando o usuário seleciona a opção cadastro de guardiões. 2. O sistema carrega o formulário de cadastro de usuários 3. O usuário insere os dados no formulário. 4. O usuário confirma os dados. 5. O sistema recebe e valida os dados do usuário 6. O sistema confirma o cadastramento do usuário 7. O sistema encerra a operação
Pós-condição	O usuário guardião confirma os dados
Cenário Alternativo	3ª – O usuário informa dados incorretos. 3ª.1 – O sistema mostra mensagem alertando dados incorretos. 3ª.2 – O sistema retorna ao passo 2 do fluxo principal. 4ª – Cliente já cadastrado. 4ª.1 - O sistema alerta que o usuário já está cadastrado. 4ª.2 – O sistema volta ao passo 2 do fluxo principal.
Caso de Uso – Realizar Cadastro Campista/Guardado	
ID	UC 002
Descrição	Este caso de uso tem por objetivo os membros Guardiões cadastrarem Campistas/Guardados
Ator Primário	Usuário do sistema (Membro Guardião)
Pré-condição	Membro Guardião estar cadastrado
Cenário Principal	1. O use case inicia quando o usuário seleciona a opção cadastro de campista. 2. O sistema carrega o formulário de cadastro de usuários. 3. O usuário insere os dados no formulário. 4. O usuário confirma os dados. 5. O sistema recebe e valida os dados do usuário. 6. O sistema confirma o cadastro do usuário. 7. O sistema encerra a operação.

Pós-condição	. O usuário Campista confirma os dados
Cenário Alternativo	3ª – O usuário informa dados incorretos. 3ª.1 – O sistema mostra mensagem alertando dados incorretos. 3ª.2 – O sistema retorna ao passo 2 do fluxo principal. 4ª – Cliente já cadastrado. 4ª - O sistema alerta que o usuário já está cadastrado. 4ª.2 – O sistema volta ao passo 2 do fluxo principal.
Caso de Uso – Alimentar perfil	
ID	UC 003
Descrição	Este caso de uso tem como objetivo o guardado alimentar seu perfil
Ator Primário	Usuário do sistema (Membro guardado)
Pré-condição	Ele estar inserido no sistema
Cenário Principal	1. O use case inicia quando o usuário seleciona a opção editar perfil. 2. O sistema carrega todos já cadastrados do usuário. 3. O usuário seleciona os dados que deseja inserir. 4. O usuário insere os dados conforme as validações de inserção. 5. O usuário salva as alterações. 6. O sistema encerra a operação.
Pós-condição	Não possui
Cenário Alternativo	5ª. O usuário inseriu alguma informação invalida. 5ª.1 O sistema informa campo invalido. 5ª.2 O sistema retoma ao passo 4 do fluxo principal.
Caso de Uso – Criar acampamento	
ID	UC 004
Descrição	Este caso de uso tem por objetivo o membro fundador criar um acampamento
Ator Primário	Usuário do sistema (Fundador)
Pré-condição	Não possui
Cenário Principal	1. O use case inicia quando o membro fundador clicar em criar acampamento. 2. O sistema direciona para a tela de criação de acampamento. 3. O usuário insere os dados conforme as validações de inserção. 4. Usuário salva e confirma criação do acampamento. 5. O sistema encerra a operação.
Pós-condição	Usuário atribuir funções
Cenário Alternativo	2ª – O usuário cancelou a criação do acampamento. 2ª.1 – retorna a tela inicial.

Caso de Uso – Adicionar funções	
ID	UC 005
Descrição	Este caso de uso tem por objetivo o membro fundador adicionar funções ao acampamento
Ator Primário	Usuário do sistema (Fundador)
Pré-condição	Usuário fundador ter criado o acampamento
Cenário Principal	1. O use case inicia quando o usuário entrar no acampamento criado e clicar em atribuir funções. 2. O sistema abre campo em formato listagem de atribuição de funções. 3. O usuário atribui cada função criada a outro usuário. 4. O usuário clica em salvar e depois em confirmar escolhas. 5. O sistema encerra o caso de uso.
Pós-condição	Cada usuário que teve função atribuída confirma sua função
Cenário Alternativo	2ª O usuário com função atribuída recusa a função. 2ª.1 O sistema retoma o passo 3 do fluxo principal.
Caso de Uso – Responder função	
ID	UC 006
Descrição	Este caso de uso tem por objetivo o membro guardado aceitar ou recusar função atribuída
Ator Primário	Usuário (Guardado)
Pré-condição	Membro fundador ter atribuído alguma função ao guardado
Cenário Principal	1. O use case inicia quando o usuário recebe alguma função no acampamento. 2. O usuário aceita a função. 3. O sistema encerra o caso de uso.
Pós-condição	O Sistema informa o fundador responsável da resposta do guardado
Cenário Alternativo	2ª. 1 O usuário recusa a atribuição. 2ª.2 O sistema informa o fundador da recusa. 2ª. 3 O sistema retoma o UC 005 passo 3.
Caso de Uso – Preencher Formulário	
ID	UC 007
Descrição	Este caso de uso tem por objetivo o membro fundador selecionar comunidades
Ator Primário	Usuário (Fundador)
Pré-condição	Nenhuma

Cenário Principal	1. O use case inicia quando o fundador clica em selecionar comunidades 2. O sistema direciona para a tela de visualização de comunidades 3. O fundador seleciona uma das comunidades que ele tem acesso 4. O sistema direciona para a tela da comunidade selecionada 5. O fundador visualiza os membros da comunidade, acampamentos que participaram e dados que tem acesso 6. O usuário confirma o envio das informações 7. O sistema encerra o caso de uso
Pós-condição	Não possui
Cenário Alternativo	Não possui

Fonte: Os autores, 2023

3.7 Diagrama de Atividades

Um diagrama de atividades é uma representação gráfica de um processo, algoritmo, ou conjunto de ações dentro de um sistema. Pertencendo à UML, esses diagramas são uma ferramenta essencial para a visualização e comunicação de como um sistema funciona.

A importância dos diagramas de atividades reside em sua capacidade de unir as partes interessadas, como equipes de negócios e desenvolvimento, para garantir que todos tenham um entendimento comum dos processos e comportamentos envolvidos em um sistema.

Eles oferecem clareza e concisão na representação de lógica de algoritmos, casos de uso UML, processos de negócios, e até mesmo elementos de arquitetura de software, tornando-os valiosos para simplificar processos complexos e aprimorar a compreensão de qualquer sistema.

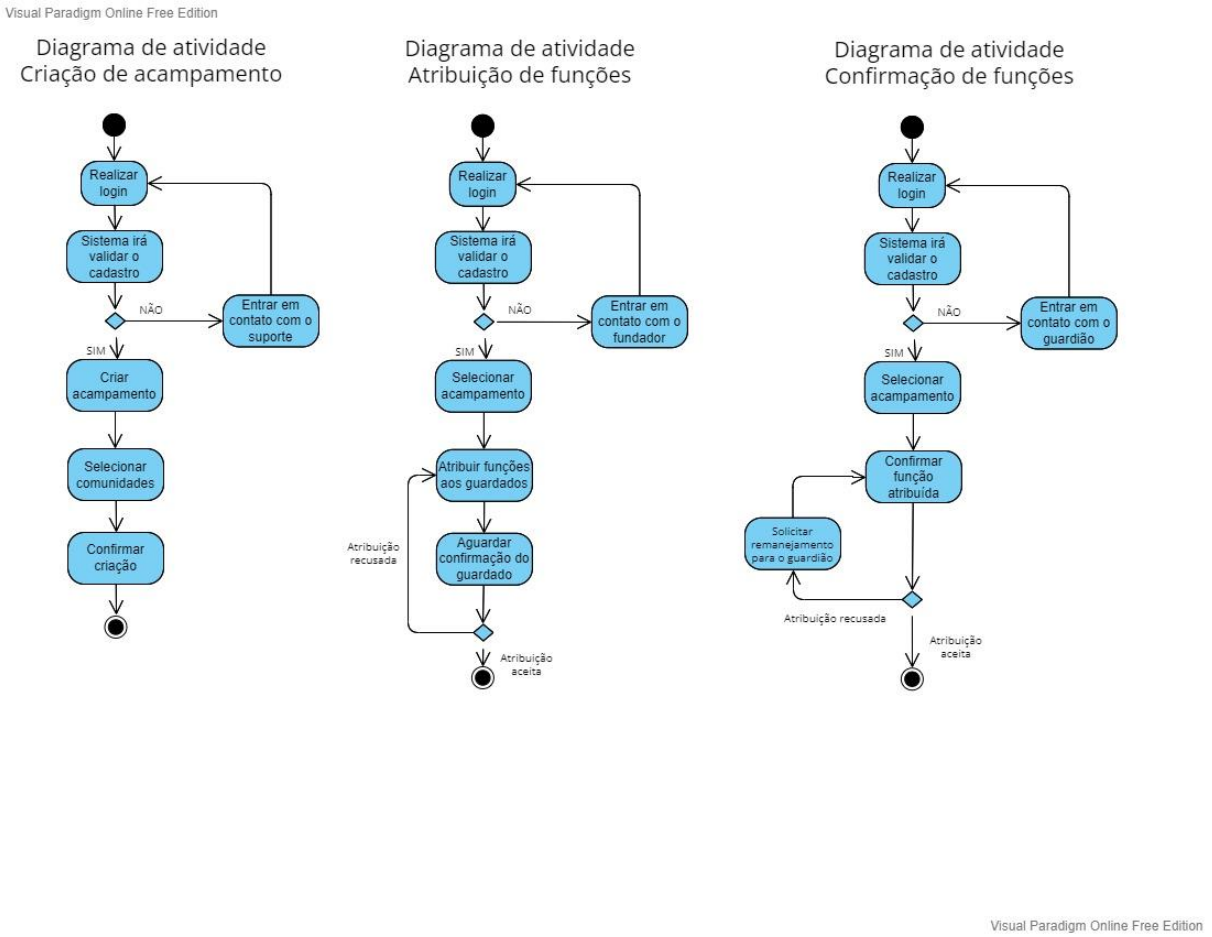
Os benefícios dos diagramas de atividades são diversos. Eles são cruciais para demonstrar a lógica por trás de algoritmos, descrever as etapas em casos de uso UML e ilustrar processos de negócios ou fluxos de trabalho em sistemas.

Além disso, simplificam a representação de casos de uso complicados, tornando-os mais acessíveis. Também podem ser usados para modelar elementos de arquitetura de software, como métodos, funções e operações, sendo utilizado para o desenvolvimento e a documentação de sistemas complexos.

Em resumo, os diagramas de atividades auxiliam na compreensão e comunicação eficaz de como os sistemas funcionam, colaborando para o sucesso de projetos de desenvolvimento de software e processos de negócios (LUCIDCHART – Diagrama de Atividade, [s.d.], *online*).

Na sequência, apresentamos o Diagrama de Atividade correspondente ao sistema em análise.

Figura 6 - Diagrama de Atividade



Fonte: Os autores, 2023

3.8 Diagrama de Estados

Os diagramas de estado são uma ferramenta de UML para modelar e representar os diversos estados que um objeto pode assumir ao longo de seu ciclo de vida. Essa representação visual, conhecida como diagrama de estados, é amplamente utilizada em áreas como eletrônica digital e linguagens formais devido à sua eficiência e clareza na descrição dos possíveis estados de um sistema e nas transições entre eles.

Esses diagramas mostram, de maneira organizada, os estados de um objeto e os eventos ou ações que levam a transições de um estado para outro, proporcionando uma compreensão detalhada do comportamento do objeto em questão.

Vale ressaltar que os diagramas de estados desempenham um papel complementar aos diagramas de casos de uso, sendo frequentemente empregados na fase de projeto do sistema, concentrando-se na identificação dos valores que os atributos de uma classe podem assumir e nas interações que resultam na alteração desses valores.

Enquanto os diagramas de casos de uso abordam tarefas de maneira mais geral e podem envolver diversos objetos para executar uma atividade específica, os diagramas de estados focam especificamente nos estados e nas transições de um objeto individual.

Embora seja possível criar diagramas de estados que abranjam mais de um objeto, a prática recomendada é modelar diagramas de estados individuais para cada objeto, recorrendo a outros tipos de diagramas, como diagramas de colaboração ou de sequência, para ilustrar as interações entre objetos durante a execução do sistema.

Assim, os diagramas de estados são uma valiosa ferramenta na UML para analisar e projetar o comportamento detalhado dos objetos em um sistema, garantindo uma representação precisa e compreensível de seu ciclo de vida e transições de estado.

A importância do diagrama de estados na UML é significativa, pois essa ferramenta atua na modelagem detalhada do comportamento dos objetos ao longo de seu ciclo de vida. Ao representar visualmente os estados possíveis e as transições entre eles, os diagramas de estados oferecem uma compreensão clara e organizada do funcionamento de um sistema.

Isso auxilia para o desenvolvimento de software eficiente, pois permite identificar e analisar os estados de um objeto e os eventos que levam a mudanças de estado. Além disso, os diagramas de estados são particularmente valiosos na fase de projeto do sistema, onde a ênfase recai sobre a definição precisa dos valores que os atributos de uma classe podem assumir.

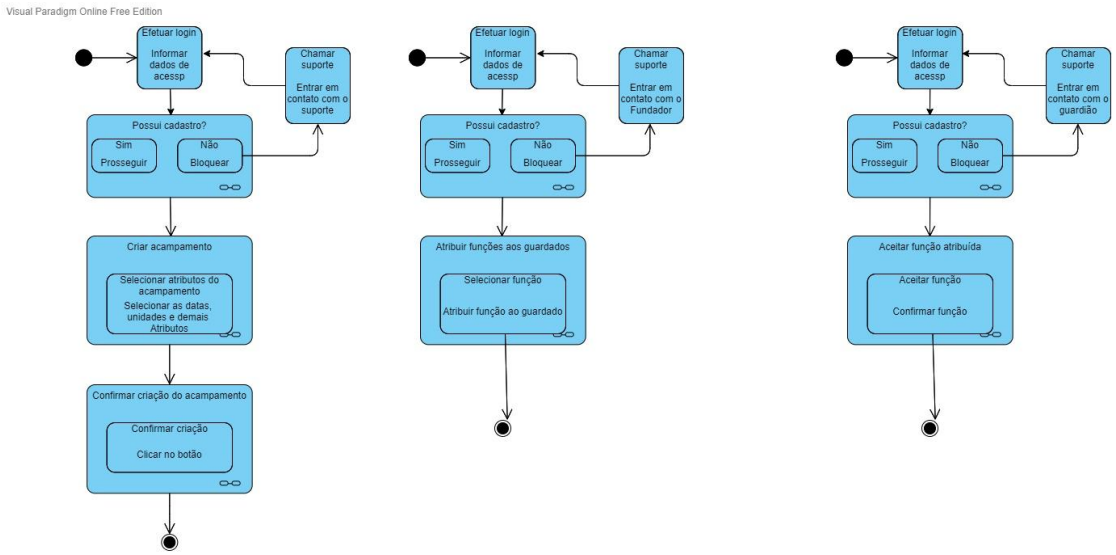
Ao complementar os diagramas de casos de uso, que abordam tarefas de maneira mais geral, os diagramas de estados garantem uma representação detalhada e específica do comportamento dos objetos, tornando mais fácil o entendimento dos processos internos do sistema.

Isso é essencial para que os desenvolvedores possam implementar de forma precisa e eficiente as funcionalidades desejadas. Em resumo, os diagramas de estados são uma ferramenta valiosa para a modelagem e a análise de sistemas,

contribuindo para a qualidade do software e para a compreensão dos aspectos dinâmicos e comportamentais dos objetos envolvidos (LUCIDCHART – Diagrama de Máquina de Estados UML, [s.d.], *online*).

Na sequência, apresentamos o Diagrama de Máquina de Estado correspondente ao sistema em análise.

Figura 7 - Diagrama de Máquina de Estado



Fonte: Os autores, 2023

3.9 Diagrama de Sequência

Um diagrama de sequência na UML é uma representação gráfica que descreve a interação e a ordem de trabalho de um grupo de objetos em um sistema.

Esses diagramas desempenham auxiliam no desenvolvimento de software e na documentação de processos de negócios. Eles são ferramentas valiosas para desenvolvedores e profissionais de negócios, permitindo que compreendam como os objetos colaboram para alcançar um objetivo e, assim, compreendam as necessidades do sistema.

Além disso, os diagramas de sequência UML são essenciais para representar detalhadamente casos de uso, modelar a lógica de processos complexos e visualizar a interação entre objetos, contribuindo para o planejamento e a compreensão de funcionalidades detalhadas de cenários existentes ou futuros.

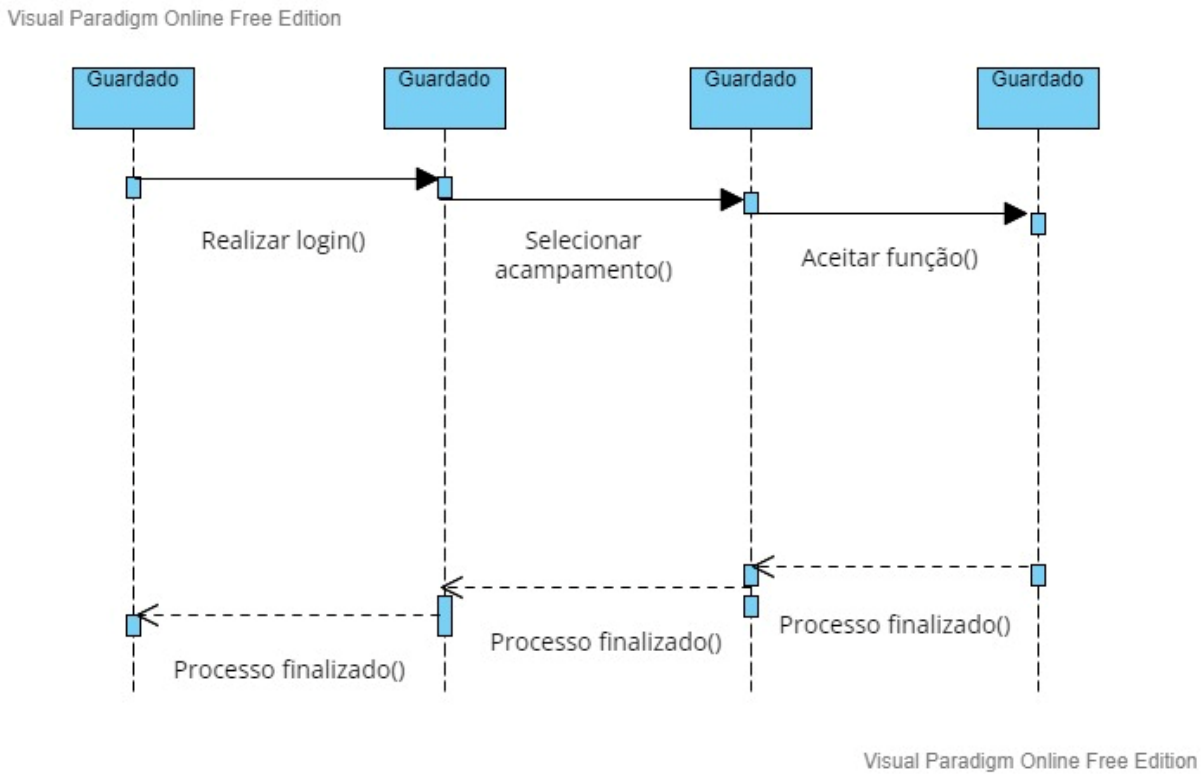
A importância dos diagramas de sequência reside em sua capacidade de oferecer uma visão clara e estruturada das interações entre objetos, tornando mais fácil para as empresas e organizações entenderem a dinâmica de seus sistemas.

Esses diagramas servem como referências valiosas para garantir uma compreensão completa da lógica por trás de casos de uso, funções complexas e operações, auxiliando no desenvolvimento de sistemas mais eficientes e na otimização de processos.

Portanto, os diagramas de sequência desempenham um papel essencial na modelagem e na análise de sistemas, permitindo que profissionais capturem e comuniquem de forma eficaz as interações entre objetos em um contexto específico (LUCIDCHART – Diagrama de Sequência UML, [s.d.], *online*).

Na sequência, apresentamos o Diagrama de Sequência correspondente ao sistema em análise.

Figura 8 - Diagrama de Sequência



Fonte: Os autores, 2023

3.10 Matriz de Rastreabilidade

Uma matriz de rastreabilidade de requisitos é uma ferramenta para qualquer projeto, pois permite a vinculação direta e organizada dos requisitos com diversos componentes do empreendimento (ESPINHA, 2020, online).

Isso não apenas facilita o acompanhamento da origem dos requisitos, mas também estabelece conexões claras com outros elementos do projeto, possibilitando a identificação de inconsistências e o alinhamento de diferentes níveis do projeto com os mesmos requisitos (ESPINHA, 2020, online).

Com essa ferramenta, é possível evitar que os requisitos se percam ou se desvinculem ao longo do desenvolvimento, tornando as mudanças de escopo mais controladas e garantindo que o projeto continue a atender às necessidades iniciais (ESPINHA, 2020, online).

Existem vários tipos de matrizes de rastreabilidade, incluindo aquelas que focam em dependências, funcionalidades, fontes de requisitos, subsistemas e interfaces (ESPINHA, 2020, online).

Em essência, essas matrizes oferecem um método sistemático e organizado para gerenciar os requisitos do projeto, possibilitando um desenvolvimento mais eficiente e uma maior qualidade do produto (ESPINHA, 2020, online).

Portanto, a matriz de rastreabilidade desempenha um papel crucial na engenharia de requisitos, assegurando que os objetivos iniciais sejam atendidos e que todas as partes interessadas estejam alinhadas ao longo do ciclo de vida do projeto (ESPINHA, 2020, online).

Assim, a matriz de rastreabilidade tem uma função essencial na engenharia de requisitos, garantindo o cumprimento dos objetivos iniciais e o alinhamento contínuo de todas as partes interessadas ao longo do ciclo de vida do projeto (ESPINHA, 2020, online).

A seguir, são apresentados os quadros correspondentes às relações entre as Regras de Negócio e os Requisitos Funcionais, bem como entre os Requisitos Funcionais e os Casos de Uso. Esses quadros delineiam de forma concisa a interligação entre as diretrizes operacionais do negócio, os requisitos específicos do sistema e a correlação direta com as funcionalidades a serem implementadas, proporcionando uma visão clara e organizada das exigências de negócio e suas interações com os aspectos funcionais e de uso do sistema em desenvolvimento.

Quadro 5 – Regras de Negócio x Requisitos Funcionais

RN001	Cadastro de membros Fundadores	RF009
RN002	Cadastro de membros Guardiões	RF009
RN002	Cadastro de membros guardados	RF009
RN004	Relacionamento Casal	Nenhum RF relacionado diretamente
RN005	Criação acampamento	RF003
RN006	Atribuir funções	RF003, RF004, RF005, RF006
RN007	Selecionar comunidades	RF001
RN008	Alterar dados pessoais	RF002
RN009	Efetuar Login	RF007, RF008
RN010	Acessar a Home	RF008

RN011	Acessar tela de cadastro de novos membros	RF009
RN011	Acessar tela de cadastro de novos membros	RF010
RN011	Acessar tela de cadastro de novos membros	RF011

Fonte: Os autores, 2023

Quadro 6 - Requisitos Funcionais x Caso de Uso

ID Caso de Uso	**Descrição Caso de Uso**	**RF Relacionados**
UC 001	Realizar Cadastro Guardião	RF009
UC 002	Realizar Cadastro Campista/Guardado	RF009
UC 002	Realizar Cadastro Campista/Guardado	RF011
UC 003	Alimentar perfil	RF002
UC 004	Criar acampamento	RF001

UC 005	Adicionar funções	RF003
UC 006	Responder função	RF004, RF005
UC 007	Preencher Formulário	RF011

Fonte: Os autores, 2023

3.11 Modelo Entidade-Relacionamento

O Modelo Entidade-Relacionamento (MER) é uma ferramenta conceitual essencial na Engenharia de Software, desempenhando o papel de representar de forma abstrata os elementos envolvidos em um domínio de negócios (JOEL, 2014, *online*).

Esse modelo descreve as entidades, seus atributos distintivos e as relações que interligam essas entidades. Ao fazê-lo, proporciona uma visão clara da estrutura que será incorporada ao banco de dados da aplicação, antecipando a organização dos dados e seu relacionamento, o que é crucial para o desenvolvimento eficiente de sistemas (JOEL, 2014, *online*).

Vale ressaltar que a abordagem na criação de modelos pode variar de acordo com a escala do projeto, permitindo a criação de modelos específicos para partes individuais do sistema, tornando a modelagem mais gerenciável e contextualizada, especialmente em sistemas abrangentes, como ERP’s, onde a modularização se faz necessária (JOEL, 2014, *online*).

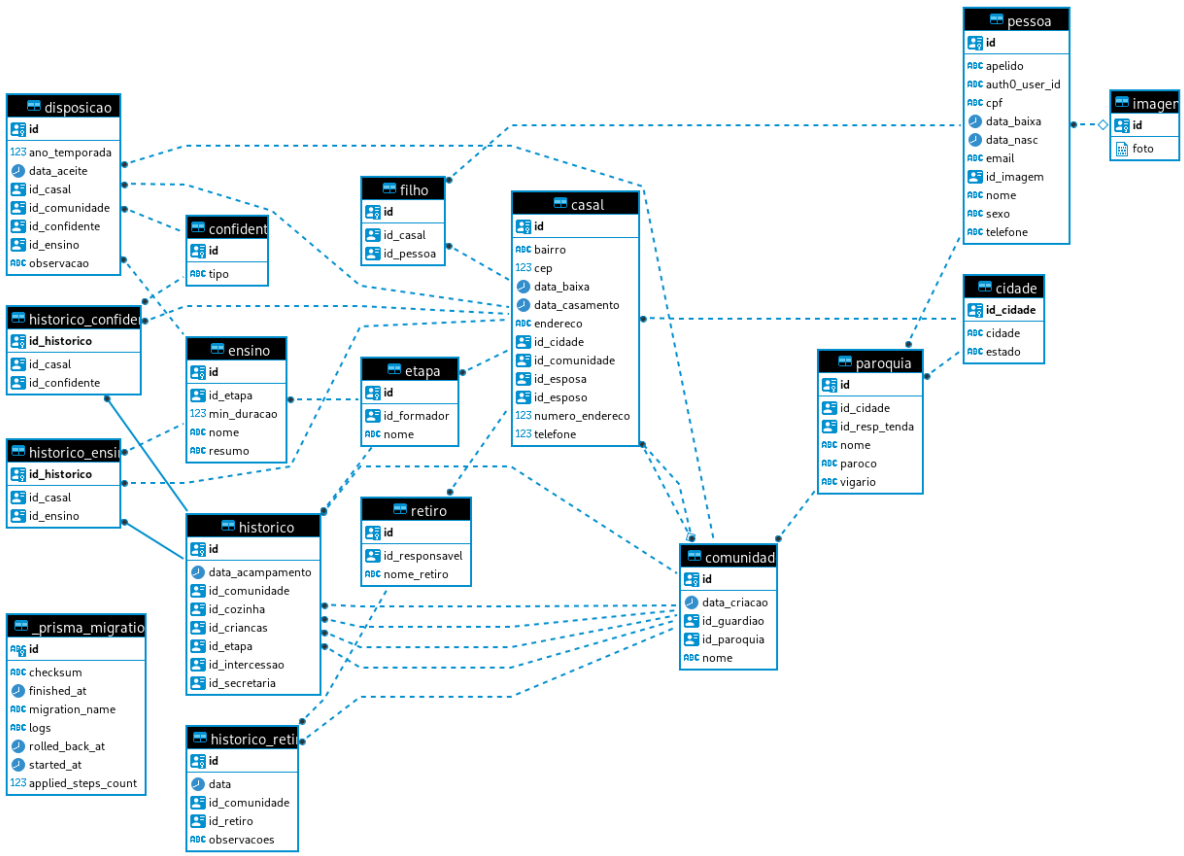
A importância do Modelo Entidade-Relacionamento reside na sua capacidade de fornecer um panorama claro da estrutura do banco de dados de um sistema, tornando-se uma ferramenta valiosa para desenvolvedores, arquitetos e equipes de projeto (JOEL, 2014, *online*).

Com a representação das entidades e de como elas se relacionam, a complexidade do sistema é gerenciada de maneira eficaz, o que permite a identificação de requisitos, a criação de bases de dados sólidas e a otimização do desenvolvimento de software (JOEL, 2014, *online*).

Além disso, a flexibilidade desse modelo, que permite a divisão em partes menores, oferece adaptabilidade em projetos de diferentes magnitudes, garantindo que a modelagem atenda às necessidades específicas do sistema, desde módulos isolados até sistemas corporativos abrangentes, impulsionando a eficiência e a clareza em todas as etapas do processo de desenvolvimento de software (JOEL, 2014, *online*).

A representação gráfica do nosso Modelo Entidade-Relacionamento (MER) é demonstrada na figura subsequente. Este modelo oferece uma visão esquemática e estruturada das entidades envolvidas no sistema, bem como dos relacionamentos estabelecidos entre elas, proporcionando uma compreensão visual clara da organização e interação dos dados dentro do contexto do projeto ou sistema em análise.

Figura 9 - Modelo Entidade-Relacionamento



Fonte: Os autores, 2023

4 Ferramentas e Métodos

Para o desenvolvimento do *front-end*, optou-se por utilizar Next.js 13, TypeScript 5, React 18, SaaS UI e Chakra UI. A escolha dessas tecnologias foi baseada em uma série de fatores que visam garantir uma experiência de usuário superior e uma base sólida para o desenvolvimento.

Next.js, conhecido por suas otimizações incorporadas, oferece melhorias significativas na experiência do usuário e atende aos *Core Web Vitals*, otimizando automaticamente imagens, fontes e scripts. Além disso, sua capacidade de *streaming* de HTML dinâmico, integrada ao roteador de aplicativos e ao React Suspense, proporciona um recarregamento instantânea do servidor.

A integração de React Server Components permite a adição de componentes sem a necessidade de enviar JavaScript adicional para o cliente, aproveitando os recursos mais recentes do React. Next.js também oferece suporte robusto para recuperação de dados, tanto do lado do servidor quanto do lado do cliente, bem como para estilização com ferramentas populares, como CSS Modules, Sass, Tailwind CSS e styled-jsx.

Sua flexibilidade na renderização e opções de cache, incluindo Regeneração Estática Incremental (ISR) por página, permite que seja ajustado o desempenho conforme necessário. Além disso, a compatibilidade com Node.js e as opções de execução de Edge RunTimes possibilitam a criação de soluções escaláveis, incluindo funções sem servidor para fornecer conteúdo dinâmico e personalizado na borda.

Além disso, a Next.js oferece suporte para roteamento avançado e layouts aninhados, simplificando a criação de rotas usando a estrutura de arquivos (“Next.js by Vercel - The React Framework”, [s.d.], *online*).

Em suma, a escolha do Next.js e TypeScript se baseia na busca por uma solução robusta e otimizada que atenda aos requisitos de performance, escalabilidade e flexibilidade, enquanto TypeScript, como uma extensão do JavaScript, proporciona um ambiente mais seguro e produtivo para o desenvolvimento, identificando erros precocemente no processo de codificação e oferecendo confiança na interoperabilidade com diversos ambientes de execução JavaScript, seja no navegador, Node.js ou Deno (“TypeScript - JavaScript that scales.”, [s.d.], *online*).

A escolha do React como parte fundamental do nosso front-end é altamente justificada por várias razões. Primeiro, React permite que sejam criadas interfaces de usuário de forma modular, usando componentes individuais. Essa abordagem é

altamente vantajosa, pois nos permite desenvolver componentes personalizados, como miniaturas, botões de curtir e vídeos, e, em seguida, combiná-los para criar telas, páginas e aplicativos completos. Isso facilita a reutilização de código e a manutenção, tornando o desenvolvimento mais eficiente e escalável.

E, React é uma escolha excelente porque promove a escrita de componentes usando código e marcação. Os componentes React são essencialmente funções JavaScript, o que significa ser possível usar estruturas de controle como declarações condicionais e mapeamento de *arrays* para tornar nossos componentes dinâmicos e interativos.

A sintaxe JSX é uma extensão JavaScript popularizada pelo React e permite que a marcação esteja intimamente ligada à lógica de renderização relacionada, tornando os componentes React fáceis de criar, manter e depurar (“React”, [s.d.], *online*).

Outra vantagem importante é a capacidade de adicionar interatividade onde for necessário. Os componentes React podem receber dados e retornar o que deve ser exibido na tela. Isso permite atualizar a interface em resposta a interações do usuário, como entrada de texto, sem a necessidade de recarregar a página.

Além disso, React não exige que todo o aplicativo seja desenvolvido usando essa biblioteca; ele pode ser facilmente adicionado a uma página HTML existente, permitindo a renderização de componentes interativos em qualquer lugar.

Além disso, React é uma biblioteca altamente flexível e agnóstica em relação à arquitetura, não prescrevendo como fazer roteamento e busca de dados.

Por último, mas não menos importante, React permite que o aproveitamento do melhor de cada plataforma. Sendo possível criar aplicativos da web e aplicativos nativos usando as mesmas habilidades e componentes, tirando proveito das vantagens únicas de cada plataforma para oferecer a melhor experiência possível aos usuários.

Essa abordagem nos permite ser desenvolvedores tanto de aplicativos da web quanto de aplicativos nativos, atendendo a diversas plataformas sem comprometer a qualidade da experiência do usuário. Em resumo, React é uma escolha sólida que nos permite desenvolver interfaces de usuário eficientes, escaláveis e altamente interativas para uma variedade de plataformas (“React”, [s.d.], *online*).

Para a construção do design das páginas, escolheu-se utilizar as bibliotecas Chakra UI e SaaS UI. O Chakra UI oferece uma abordagem acessível, estritamente

seguindo os padrões WAI-ARIA para todos os componentes, garantindo uma experiência inclusiva para todos os usuários. Sua capacidade de personalização é notável, permitindo a adaptação de qualquer parte dos componentes para atender às necessidades de design específicas do projeto.

Além disso, o Chakra UI é altamente componível, facilitando a criação de novos componentes de maneira eficiente. Sua capacidade de alternar entre modos de cores claras e escuras otimiza a experiência para os usuários em diferentes contextos.

Com uma comunidade ativa e um foco na experiência do desenvolvedor, o Chakra UI promete aumentar a produtividade no desenvolvimento de aplicativos ou sites, reduzindo a quantidade de código de interface do usuário necessário e permitindo mais tempo para a construção de uma experiência excepcional para os clientes (“Chakra UI - A simple, modular and accessible component library that gives you the building blocks you need to build your React applications.”, [s.d.], *online*).

Por outro lado, o SaaS UI é uma biblioteca de componentes React e um starter kit que agiliza o processo de construção de produtos SaaS intuitivos e eficientes. Ele se destaca pela redução do *boilerplate code*, fornecendo uma ampla gama de componentes fundamentais e avançados que cobrem diversas necessidades, desde telas de autenticação até tabelas de dados funcionais.

Com mais de 40 componentes de código aberto, incluindo integrações com bibliotecas como Clerk, Supabase e Magic para autenticação, e React Hook Form para formulários, o SaaS UI oferece uma base sólida para o desenvolvimento de aplicativos. Além disso, o SaaS UI se baseia em ferramentas líderes do setor, como Chakra UI, React Hook Form, React Table e outras, proporcionando um ambiente robusto para a construção de produtos.

Esta biblioteca foi projetada para atender tanto a desenvolvedores quanto a designers, fornecendo componentes bem elaborados e uma base sólida para personalização. O SaaS UI reduz consideravelmente o tempo na construção de funcionalidades essenciais, permitindo foco na validação de novas ideias, na busca pelo ajuste perfeito com o mercado e na criação de funcionalidades que tornam nosso produto único (“Saas UI”, [s.d.], *online*).

Para o desenvolvimento do *back-end*, a escolha recaiu sobre o Nest.js, juntamente com o TypeScript, devido à sua extensibilidade e arquitetura modularizada, que oferecem uma verdadeira flexibilidade no uso de outras bibliotecas. O Nest.js é uma estrutura Node.js progressiva, projetada para a construção de

aplicativos eficientes, escaláveis e confiáveis no lado do servidor. Além disso, ele adota as mais recentes características do JavaScript, trazendo padrões de design e soluções maduras para o mundo do Node.js.

O Nest.js proporciona uma abstração acima de estruturas HTTP robustas como o Express (a configuração padrão) e opcionalmente pode ser configurado para usar o Fastify.

Isso oferece aos desenvolvedores a liberdade de utilizar uma infinidade de módulos de terceiros disponíveis para a plataforma subjacente. A filosofia do Nest.js alinha-se com a necessidade de uma arquitetura sólida para aplicativos no lado do servidor, proporcionando uma estrutura que permite a criação de aplicativos altamente testáveis, escaláveis, desacoplados e facilmente mantíveis, inspirada fortemente pelo sucesso do Angular no mundo front-end (“Documentation | NestJS - A progressive Node.js framework”, 2023, *online*).

Além disso, o Prisma foi adotado na versão 9 para facilitar a integração com qualquer banco de dados. O Prisma é uma ORM de próxima geração, composta por várias partes, incluindo o Prisma Client, que é um gerador de consultas seguro e tipado para Node.js e TypeScript (“What is Prisma? (Overview)”, [s.d.], *online*).

Ele oferece suporte para diversas aplicações no *back-end*, como REST APIs, GraphQL APIs e até mesmo aplicações *serverless* e *microservices*. A adição do Prisma ao *stack* de tecnologias contribui para uma eficiente manipulação de dados e simplifica o acesso e a manipulação de informações no banco de dados, resultando em um desenvolvimento mais ágil e eficaz do *back-end* do sistema (“What is Prisma? (Overview)”, [s.d.], *online*).

Utilizou-se o poder do TanStack Query para simplificar nossa gestão assíncrona de estado dentro de um ambiente TypeScript e React. Essa robusta biblioteca nos permitiu descartar as complexidades da gestão manual de estado e da busca de dados trabalhosa, resultando em um código mais limpo e eficiente.

O TanStack Query adota uma abordagem declarativa, gerenciando com facilidade consultas e mutações enquanto garantimos que nossos dados estejam sempre atualizados. Essa melhoria na experiência do desenvolvedor se traduziu em uma experiência aprimorada para o usuário, graças ao armazenamento em cache embutido, atualizações em segundo plano e tratamento sem configuração prévia de dados obsoletos.

Ademais, a simplicidade e familiaridade do TanStack Query, baseadas em conceitos como promessas e *async/await*, se integraram perfeitamente ao nosso projeto, eliminando a necessidade de uma complexa gestão global de estado ou configurações pesadas.

Ele oferece extensibilidade até mesmo em cada instância observadora, fornecendo opções e ferramentas personalizáveis para uma ampla variedade de casos de uso. Sem dependências externas, o TanStack Query se revelou uma solução leve, mas rica em recursos, atendendo a projetos pessoais e aplicativos de nível empresarial.

Sua versatilidade, natureza agnóstica em relação ao backend e suporte a uma ampla variedade de recursos, como *devtools* dedicados, armazenamento em cache automático, paginação, API de mutações e suporte offline, o tornaram um ativo valioso em nossa busca por uma gestão eficiente e flexível de dados (“TanStack Query | React Query, Solid Query, Svelte Query, Vue Query”, [s.d.], *online*).

Além disso, foi utilizado o isomorphic-unfetch que é uma biblioteca JavaScript extremamente leve, com cerca de 500 bytes após ser comprimida, projetada para simplificar e padronizar as solicitações HTTP, especialmente ao utilizar a API *fetch* em ambientes tanto do lado do servidor quanto do lado do cliente.

Ela oferece funcionalidades mínimas, fornecendo suporte apenas para solicitações *fetch* com cabeçalhos e respostas em texto ou JSON. Uma das características essenciais do isomorphic-unfetch é sua compatibilidade com navegadores mais antigos, incluindo o Internet Explorer 8, desde que a Promise seja devidamente emulada.

No projeto, a escolha do isomorphic-unfetch é justificada pela sua simplicidade e tamanho mínimo, o que ajuda a manter o aplicativo leve e eficiente em termos de recursos. Ele fornece uma interface consistente e fácil de usar para fazer solicitações HTTP, seja no servidor ou no cliente, facilitando a integração de serviços externos e o carregamento de dados dinâmicos.

Ademais, sua compatibilidade com navegadores mais antigos é vantajosa, pois garante que o aplicativo funcione de maneira consistente em uma variedade de ambientes de usuário, sem a necessidade de lógica adicional para lidar com diferenças entre navegadores.

Portanto, o isomorphic-unfetch é uma escolha sólida para a realização de solicitações HTTP no projeto, contribuindo para uma experiência de usuário mais

confiável e eficaz (MILLER, 2023, *online*).

Quanto a autenticação do usuário, o Auth0 é uma solução flexível e completa para adicionar serviços de autenticação e autorização às nossas aplicações, descomplicando todo o processo.

No projeto, utilizou-se o Auth0 pelas diversas vantagens que oferece. Com ele, conseguimos implementar facilmente a autenticação e autorização dos usuários, seja por meio de autenticação tradicional com nome de usuário/senha ou por contas de redes sociais.

Além disso, o Auth0 nos permite personalizar a interface do usuário e aplicar políticas de autorização específicas após o login, garantindo um alto nível de controle sobre o acesso aos recursos da aplicação. Também é uma escolha estratégica, pois nos ajuda a atender a diversos requisitos de conformidade, como *Service Organization Control 2* (SOC2), Regulamento Geral de Proteção de Dados (GDPR), *Payment Card Industry Data Security Standard* (PCI DSS) e *Health Insurance Portability and Accountability Ac*t (HIPAA), o que é essencial para garantir a segurança e a privacidade dos dados dos nossos usuários.

Com o Auth0, é possível focar no desenvolvimento do nosso aplicativo, deixando toda a complexidade de autenticação e autorização nas mãos de uma solução confiável e robusta.

E, o Auth0 se alinha perfeitamente com as necessidades do nosso projeto, como a segurança de uma API com OAuth 2.0, a implementação de Single Sign-on (SSO) em várias aplicações, a autenticação baseada em *Security Assertion Markup Language* (SAML) e a capacidade de notificar e proteger os usuários em caso de comprometimento de suas informações de login.

Também nos permite adotar *Multi-Factor Authentication* (MFA) quando necessário, o que é importante para o acesso a dados sensíveis. Com a capacidade de monitorar e analisar o comportamento dos usuários em nosso aplicativo, o Auth0 oferece uma visão valiosa para melhorar continuamente a experiência do usuário e otimizar nossos processos de inscrição.

No geral, a escolha do Auth0 simplifica nossa implementação de autenticação e autorização, garantindo a segurança, o controle e a conformidade necessários para o sucesso do nosso projeto (AUTH0, 2023, *online*).

E, para validar o token do usuário e criar rotas autenticadas, utilizou-se o Passport, que é uma ferramenta de middleware de autenticação para Node.js, que

oferece autenticação de forma simples e flexível. Sua abordagem modular permite que seja facilmente integrado em qualquer aplicativo web baseado no framework Express. Com um conjunto abrangente de estratégias de autenticação, o Passport suporta diversos métodos de autenticação, incluindo usuário e senha, Facebook, Twitter, entre outros.

A escolha do Passport para nosso projeto se justifica pela sua facilidade de implementação e pela ampla documentação disponível. Seja para criar a primeira página de login ou para lidar com questões avançadas de autenticação e autorização, o Passport oferece suporte necessário.

Além disso, sua modularidade permite que a adaptação e personalização de acordo com as estratégias de autenticação de acordo com as necessidades específicas do nosso aplicativo, tornando-o uma escolha sólida para garantir a segurança e autenticidade de nossos usuários.

O Passport trabalha em conjunto com a Passport-jwt e RxJS para fornecer autenticação segura usando JSON Web Tokens (JWT). A Passport-jwt é uma estratégia do Passport voltada para autenticação por meio de JWT, o que é particularmente útil para proteger endpoints RESTful sem a necessidade de sessões. Isso torna nossa aplicação mais eficiente e adequada para um ambiente onde a autenticação precisa ser rápida e sem sobrecarregar o servidor com a gestão de sessões (“Documentation”, 2023, *online*).

A integração com a biblioteca RxJS também é vantajosa, pois permite a programação reativa com Observables, facilitando a composição de código assíncrono de maneira mais legível e eficiente.

Essa abordagem permite melhorar o desempenho e a modularidade do nosso aplicativo, reduzindo o tamanho geral e melhorando a consistência das APIs. Portanto, a utilização do Passport, em conjunto com a Passport-jwt e RxJS, fortalece nossa estratégia de autenticação e proporciona uma experiência segura e eficaz para nossos usuários (“RxJS”, [s.d.], *online*).

Para garantir uma experiência de desenvolvimento eficiente e flexível, foi adotado um conjunto de tecnologias que simplificam o ambiente de trabalho para novos membros da equipe. O Traefik, uma plataforma de roteamento de borda de código aberto, foi escolhido para facilitar a publicação de serviços, tornando o processo divertido e descomplicado.

O Traefik é nativamente compatível com várias tecnologias de cluster, incluindo Kubernetes, Docker, Docker Swarm, AWS, Mesos, Marathon e muito mais, tornando-o uma escolha versátil. O que o diferencia é sua capacidade de descobrir automaticamente a configuração certa para os serviços, eliminando a necessidade de manter um arquivo de configuração separado.

Com o Traefik, tudo acontece automaticamente, em tempo real, sem reinicializações ou interrupções de conexão, permitindo que a equipe se concentre no desenvolvimento e implantação de novos recursos, em vez de configurar e manter o ambiente de trabalho (“Traefik Proxy Documentation - Traefik”, [s.d.], *online*)

E, utilizou-se também o NextAuth.js, que é uma solução completa de autenticação de código aberto projetada especialmente para aplicativos Next.js. Sua utilização em nosso projeto se justifica pela facilidade de implementação que oferece, juntamente com a flexibilidade para se adaptar às necessidades específicas da nossa aplicação.

Com suporte embutido para diversos serviços populares de autenticação, como Google, Facebook, Auth0 e Apple, além de autenticação por e-mail, senha e links mágicos, o NextAuth.js simplifica a autenticação de usuários de maneira segura.

Além disso, sua capacidade de suportar qualquer sistema de armazenamento de nomes de usuário/senhas, bem como serviços OAuth 1.0 e 2.0, torna-o uma escolha versátil.

O NextAuth.js também se destaca por sua segurança. Ele utiliza cookies assinados e prefixados do lado do servidor, validação de token CSRF e oferece suporte a *JSON Web Token (*JWT) com *JSON Web Signature* (JWS) / *JSON Web Encryption* (JWE) / *JSON Web Key* (JWK) para garantir a integridade dos dados e a autenticação segura.

Outro fator, não depende de JavaScript do lado do cliente, garantindo que a autenticação seja resistente a ataques e vulnerabilidades comuns. Sua capacidade de ser executado em ambientes *Serverless*, como AWS Lambda, bem como a flexibilidade para escolher entre sessões de banco de dados ou JWT, tornam-no uma escolha robusta para nosso projeto.

Com o NextAuth.js, é possível implementar a autenticação de forma fácil, segura e altamente personalizável, mantendo o controle sobre nossos dados e promovendo as melhores práticas de segurança (“INTRODUCTION | NextAuth.js”, [s.d.], *online*).

O uso de contêineres Docker faz parte dessa abordagem para simplificar o desenvolvimento. O Docker é uma plataforma aberta que permite separar aplicativos da infraestrutura, facilitando a entrega rápida de software. Ele permite o gerenciamento da infraestrutura da mesma forma que se gerencia aplicativos, aproveitando suas metodologias para envio, teste e implantação de código de maneira eficiente.

Ao adotar o Docker, reduzimos significativamente o tempo entre a escrita de código e sua execução em produção, promovendo a eficiência no desenvolvimento (“Get Docker”, 2022, *online*).

O Turbo Repo, por sua vez, reimagina técnicas de sistemas de compilação utilizadas por empresas como Facebook e Google para eliminar a carga de manutenção e overhead. Ele oferece builds incrementais, *content-aware hashing*, execução paralela e *caching* remoto, o que resulta em compilações mais rápidas e redução de trabalho duplicado. O Turbo Repo se ajusta às necessidades da equipe, simplificando o processo de desenvolvimento e melhorando a velocidade do CI, tornando as tarefas de desenvolvimento e implantação mais eficazes (“Turborepo Quickstart – Turborepo”, [s.d.], *online*).

Para facilitar a integração de novos desenvolvedores, implementou-se o Dev Container com arquivos make e yml. O Dev Container é uma extensão do Visual Studio Code que permite usar um contêiner como ambiente de desenvolvimento completo.

Isso significa que os desenvolvedores podem abrir qualquer pasta dentro de um contêiner e aproveitar todo o conjunto de recursos do Visual Studio Code, independentemente de onde suas ferramentas ou código estão localizados. Isso elimina a necessidade de configurações complicadas no ambiente de desenvolvimento local e permite uma transição suave para novos membros da equipe, que precisam apenas do Docker e Docker Compose para começar a trabalhar (“Developing inside a Container using Visual Studio Code Remote Development”, [s.d.], *online*).

Com essa abordagem, garantimos que os desenvolvedores tenham uma experiência de desenvolvimento de alta qualidade, incluindo *IntelliSense*, navegação de código e depuração, independentemente de onde suas ferramentas ou código estejam localizados (“Developing inside a Container using Visual Studio Code Remote Development”, [s.d.], *online*).

Também utilizou-se o PM2, que é uma poderosa ferramenta de gerenciamento de processos para aplicativos Node.js, projetada para otimizar o ambiente de produção e garantir que os aplicativos permaneçam online de forma confiável 24/7.

Sua ampla gama de recursos, incluindo configuração de comportamento, suporte a mapas de origem, integração de contêineres, monitoramento, gerenciamento de logs e muito mais, fazem dele uma escolha essencial para ambientes de produção.

Com sua capacidade de carregamento a quente, recarregamento automático e recursos de *clustering*, o PM2 torna o gerenciamento de aplicativos Node.js mais eficiente e robusto, auxiliando na estabilidade e o desempenho em cenários de produção (“PM2 - Home”, [s.d.], *online*).

Além disso, o suporte a integrações com outras ferramentas e a compatibilidade com diversas plataformas como serviços de nuvem (PaaS) e o monitoramento oferecido pelo *Keymetrics* tornam o PM2 uma escolha altamente vantajosa para o desenvolvimento e o gerenciamento de aplicativos Node.js em ambientes de produção (“PM2 - Home”, [s.d.], *online*).

Também, o uso de make files simplifica ainda mais o processo de desenvolvimento e implantação, permitindo que os desenvolvedores executem tarefas específicas com facilidade e sem a necessidade de comandos complicados.

Combinado com um arquivo yml bem definido, esse sistema oferece uma experiência de desenvolvimento consistente e eficiente para toda a equipe, independentemente de sua experiência anterior ou configuração de máquina local.

Essa abordagem garante que novos desenvolvedores possam começar a trabalhar rapidamente, sem a necessidade de configurações complicadas, resultando em uma equipe mais produtiva e eficaz (“Tutorial on writing makefiles”, [s.d.], *online*).

E, utilizou-se também o lefthook, que é um gerenciador de hooks Git versátil e de alto desempenho que simplifica e aprimora o controle de qualidade de código em projetos de desenvolvimento de software modernos. Ele oferece uma solução simplificada para o gerenciamento eficiente de tarefas de pré-commit, como linting e *type-checking*, permitindo que equipes garantam que nenhum código de baixa qualidade chegue à produção.

Com a facilidade de instalação do Lefthook e a capacidade de cobrir ambientes tanto de *frontend* quanto de *backend*, ele se apresenta como uma ferramenta inestimável para grandes bases de código com múltiplos contribuidores.

Seu suporte à execução de scripts concorrentes, dependências mínimas e configuração direta o torna uma escolha ideal para manter a qualidade do código sem impor sobrecarga significativa às equipes de desenvolvimento.

Como uma solução poliglota, o Lefthook simplifica o processo de revisão de código e capacita os desenvolvedores com uma única e flexível ferramenta que unifica fluxos de trabalho e garante o uso de código de alta qualidade.

No cenário atual de desenvolvimento colaborativo, onde projetos frequentemente envolvem equipes diversas e inúmeros contribuidores, manter a qualidade do código é um desafio.

A abordagem tradicional de edição manual de *hooks* Git é trabalhosa e distrai os desenvolvedores de suas tarefas principais. O Lefthook oferece uma solução perfeita, tornando as verificações de qualidade de código uma tarefa simples, ao mesmo tempo em que oferece suporte a diferentes ambientes de programação.

Sua configuração direta e execução extremamente rápida permitem que os desenvolvedores se concentrem na escrita de código, em vez de lidar com configurações complexas. Ao reduzir a sobrecarga e garantir a qualidade consistente do código, o Lefthook simplifica o processo de desenvolvimento, tornando-se uma ferramenta indispensável para qualquer projeto, independentemente do tamanho ou complexidade.

Com o Lefthook, a equipe de desenvolvimento inteira pode trabalhar de forma eficiente, sabendo que a base de código permanece limpa e confiável (ABROSKIN; BARNOV, 2019, *online*).

Quanto ao banco de dados, optou-se por utilizar PostgreSQL pois ele é um sistema avançado de gerenciamento de banco de dados relacional de código aberto, adequado para empresas. Ele oferece suporte tanto para consultas SQL (relacionais) quanto para consultas JSON (não relacionais).

O PostgreSQL é conhecido por sua extrema estabilidade e integridade, com mais de 20 anos de desenvolvimento comunitário contribuindo para sua resiliência e correção.

Amplamente utilizado como armazenamento de dados primário ou *data warehouse* em diversas aplicações, como web, mobile, geoespaciais e análises, o PostgreSQL também se destaca por suportar tipos de dados avançados e otimização de desempenho comparável a sistemas de banco de dados comerciais, como Oracle e SQL Server.

Optou-se pelo PostgreSQL em vez do MySQL devido à sua robustez e suporte a recursos avançados de banco de dados. O PostgreSQL oferece uma ampla gama de recursos, incluindo visões materializadas, gatilhos INSTEAD OF e procedimentos armazenados em várias linguagens. Além disso, o PostgreSQL é conhecido por sua conformidade consistente com ACID (Atomicidade, Consistência, Isolamento, Durabilidade) independente do mecanismo de armazenamento, garantindo a integridade e a confiabilidade dos dados.

Embora o MySQL seja mais simples para iniciantes, o PostgreSQL se destaca em cenários que requerem desempenho aprimorado em operações de gravação de alta frequência e suporte a tipos de dados avançados, como endereços de rede geométricos e matrizes (“What is PostgreSQL? – Amazon Web Services”, [s.d.], *online*).

Essa escolha foi fundamentada na necessidade de um sistema de gerenciamento de banco de dados robusto e altamente funcional para o nosso projeto (“PostgreSQL vs. MySQL | Diferença entre sistemas de gerenciamento de bancos de dados relacionais (RDBMS) | AWS”, [s.d.], *online*).

As ferramentas escolhidas para o projeto foram selecionadas com base em sua eficiência, escalabilidade e suporte à comunidade. Além disso, essas ferramentas têm documentação abrangente, tutoriais e recursos disponíveis na comunidade de desenvolvedores, o que torna mais fácil para os desenvolvedores aprenderem e implementarem as soluções.

A escolha dessas ferramentas também foi influenciada pela preferência pessoal da equipe de desenvolvimento e experiência prévia no uso delas. As licenças das ferramentas são de código aberto, o que significa que são gratuitas e podem ser usadas para fins comerciais e pessoais. Os repositórios oficiais para cada um dos artefatos gerados são:

Next JS:

Versão: 13.4.12

Licença: MIT

Site oficial: https://nextjs.org

Nest JS:

Versão: 10.0.0

Licença: MIT

Site oficial: https://nestjs.com

Typescript:

Versão: 5.2.2

Licença: Apache-2.0

Site oficial: https://www.typescriptlang.org

React:

Versão: 18.2.0

Licença: Apache-2.0

Site oficial: https://react.dev

Prisma:

Versão: 5.2.0

Licença: Apache-2.0

Site oficial: https://www.prisma.io

Traefik:

Versão: 3.0

Licença: MIT

Site oficial: https://traefik.io

Docker:

Versão: 24.0.6

Licença: Apache-2.0

Site oficial: https://www.docker.com

Docker Compose:

Versão: 2.5.1

Licença: Apache-2.0

Site oficial: https://www.docker.com

Turbo Repo:

Versão: 1.7.4

Licença: MPL-2.0

Site oficial: https://turbo.build/repo

Dev Container:

Versão: 0.245.2

Licença: MIT

Site oficial: https://github.com/microsoft/vscode-dev-containers/tree/main

Make files:

Versão: 4.4.1

Licença: GNU Free Documentation License

Site oficial: https://www.gnu.org/software/make/manual/make.html

Lefthook:

Versão: 1.5.0

Licença: MIT

Site oficial: https://github.com/evilmartians/lefthook

PM2:

Versão: 5.3.0

Licença: GNU-AGPL-3.0

Site oficial: https://pm2.io

PostgreSQL:

Versão: 15

Licença: PostgreSQL Licence

Site oficial: https://www.postgresql.org

TanStack Query:

Versão: 5

Licença: MIT License

Site oficial: https://tanstack.com

Auth 0:

Versão: 4.0.1

Licença: MIT License

Site oficial: https://auth0.com

Next Auth:

Versão:4

Licença: ISC License

Site oficial: https://next-auth.js.org

Passport:

Versão: 0.6.0

Licença: MIT License

Site oficial: https://www.passportjs.org

Passport-jwt:

Versão: 4.0.1

Licença: MIT License

Site oficial: https://github.com/mikenicholson/passport-jwt

RxJS:

Versão: 7.8.1

Licença: Apache-2.0

Site oficial: https://rxjs.dev

Isomorphic Unfetch:

Versão: 4.0.2

Licença: MIT License

Site oficial: https://github.com/developit/unfetch

5 Desenvolvimento

A decisão de usar um repositório monolítico (*monorepo*) tem implicações significativas no ciclo de desenvolvimento. Um *monorepo* é um repositório de código versionado que abriga muitos projetos. Embora esses projetos possam estar relacionados, geralmente são logicamente independentes e gerenciados por equipes diferentes (FERNANDEZ, 2021, *online*).

A escolha do *monorepo* para este projeto oferece várias vantagens. Primeiramente, ele proporciona maior visibilidade, uma vez que todos podem acessar o código uns dos outros. Isso fomenta a colaboração e contribuições interdepartamentais, possibilitando que desenvolvedores de diferentes equipes resolvam problemas que podem não ter conhecimento anterior (FERNANDEZ, 2021, *online*).

Além disso, a gestão de dependências se torna mais simples, pois não há necessidade de um gerenciador de pacotes separado, já que todos os módulos estão no mesmo repositório. A existência de uma única fonte de verdade elimina conflitos de versão e problemas de dependência (FERNANDEZ, 2021, *online*).

A consistência também é uma vantagem, pois é mais fácil impor padrões de qualidade de código e um estilo unificado quando todo o código está em um só lugar. A escolha do *monorepo* permite que todas as equipes acompanhem as alterações em APIs ou bibliotecas compartilhadas imediatamente, incentivando a comunicação e a colaboração (FERNANDEZ, 2021, *online*).

Ademais, a capacidade de efetuar *commits* atômicos torna as refatorações em larga escala mais simples, pois um desenvolvedor pode atualizar vários pacotes ou projetos em um único *commit*. A integração contínua (CI) é garantida por padrão, uma

vez que todo o código já está unificado em um só lugar (FERNANDEZ, 2021, *online*).

Além disso, o processo de desenvolvimento *Continuous Integration/ Continuous Delivery* (CI/CD) é unificado, podendo ser aplicado da mesma forma a todos os projetos no repositório. O processo de construção também é simplificado, pois é possível usar um processo de construção compartilhado para todas as aplicações no repositório (FERNANDEZ, 2021, *online*).

No entanto, a escolha do *monorepo* também traz desafios. À medida que o repositório cresce, é possível atingir limites de design nas ferramentas de controle de versão, sistemas de compilação e pipelines de integração contínua. Problemas de desempenho podem surgir, incluindo lentidão em ferramentas de controle de versão e IDE's, bem como testes em todo o repositório a cada *commit*, o que pode se tornar inviável (FERNANDEZ, 2021, *online*).

Um repositório que engloba muitos projetos pode ser mais difícil de gerenciar para novos desenvolvedores, uma vez que a curva de aprendizado é mais íngreme. Outro desafio é a gestão de volumes de dados significativos, que podem se tornar difíceis de lidar (FERNANDEZ, 2021, *online*).

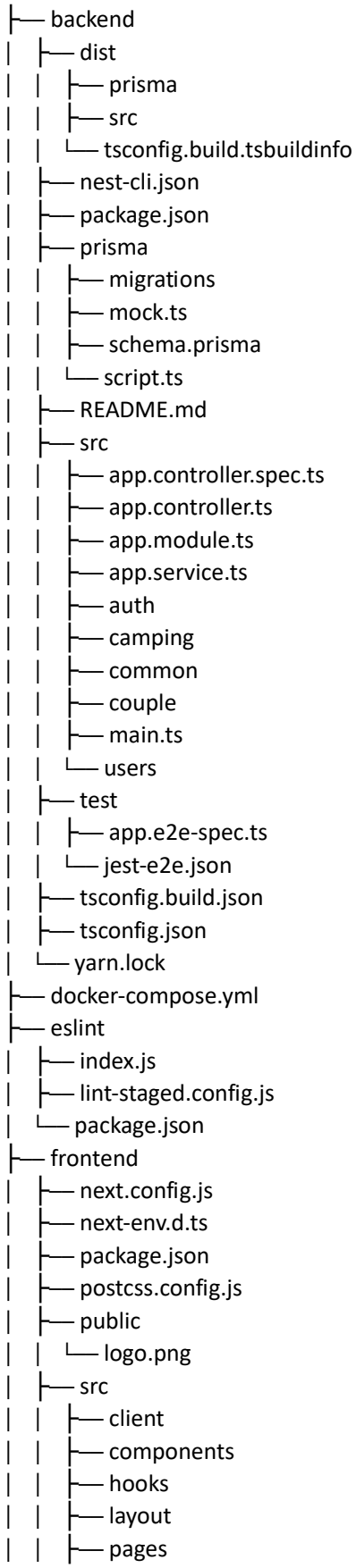
A propriedade de arquivos é mais complexa, pois sistemas como Git ou Mercurial não possuem permissões de diretório integradas. As revisões de código também podem ser ruidosas, especialmente em casos de muitos *pull requests* e revisões de código (FERNANDEZ, 2021, *online*).

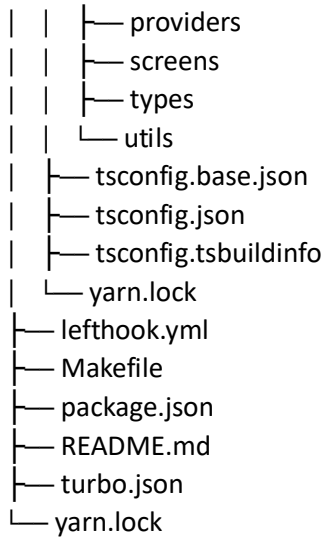
Em resumo, a escolha de adotar um *monorepo* para o desenvolvimento deste projeto foi fundamentada nas vantagens que ele oferece, como maior visibilidade, simplicidade na gestão de dependências, consistência, comunicação eficaz e *commits* atômicos.

No entanto, também é importante estar ciente dos desafios associados, incluindo problemas de desempenho, gestão de dados e revisões de código. A equipe está comprometida em enfrentar esses desafios, investindo em ferramentas e soluções personalizadas para otimizar o desenvolvimento contínuo do projeto.

Ao construir nosso *monorepo*, optamos por uma estrutura que reflete a modularidade e escalabilidade necessárias para o sucesso a longo prazo. O *monorepo* é composto por várias pastas e subpastas, cada uma com seu próprio propósito e conjunto de recursos. A árvore ao final do projeto ficou da seguinte forma:

Figura 10 - Árvore do projeto





Fonte: Os autores, 2023

Iniciamos a codificação pela criação de um ambiente de desenvolvimento consistente e isolado usando um arquivo Dockerfile localizado no diretório ".devcontainer".

Esse arquivo especifica a imagem base a ser utilizada, que é baseada no Ubuntu 22.04, garantindo um ambiente padronizado.

Figura 11 - Dockerfile do dev container

```
FROM mcr.microsoft.com/vscode/devcontainers/base:ubuntu-22.04

RUN apt-get update \
  && apt-get install -y build-essential git dnsutils net-tools iputils-
ping netcat libcurl4-openssl-dev libssl-dev socat

RUN curl -fsSL https://deb.nodesource.com/setup_lts.x | bash - \
  && apt-get install -y nodejs \
  && corepack enable

RUN curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg \
  && echo "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null \
  && apt-get update \
  && apt-get install -y docker-ce

RUN curl -SL
https://github.com/docker/compose/releases/download/v2.5.1/docker-
compose-linux-x86_64 -o /usr/bin/docker-compose \
  && chmod +x /usr/bin/docker-compose \
  && usermod -aG docker vscode

RUN curl -1sLf
'https://dl.cloudsmith.io/public/evilmartians/lefthook/setup.deb.sh' |
sudo -E bash \
  && sudo apt install lefthook

RUN yarn global add pm2
RUN yarn global add @nestjs/cli

RUN chown -R vscode:vscode /home/vscode
```

Fonte: Os autores, 2023

O Dockerfile contém uma série de comandos "RUN" que são executados durante a criação da imagem do contêiner

- O primeiro RUN atualiza o sistema operacional Ubuntu e instala várias ferramentas e utilitários de desenvolvimento, incluindo *build-essential, git, dnsutils, net-tools, iputils-ping, netcat, libcurl4-openssl-dev, libssl-dev,* e *socat.*

- O segundo RUN instala o Node.js e habilita o *corepack*, que é uma ferramenta para gerenciamento de pacotes.

- O terceiro RUN configura a instalação do Docker, incluindo a importação de chaves de assinatura e a adição de repositórios Docker. Em seguida, instala o Docker Community Edition (docker-ce).

- O quarto RUN baixa e instala o Docker Compose, uma ferramenta para definir e executar aplicativos Docker multi-container.

- O quinto RUN instala o LeftHook, que é um gerenciador de ganchos Git usado para automatizar ações durante o ciclo de vida de um projeto.

- Os comandos subsequentes instalam globalmente o PM2, que é um gerenciador de processos Node.js, e a CLI do NestJS, que é um framework para desenvolvimento de aplicativos Node.js.

- O último RUN define as permissões corretas para os arquivos no diretório /home/vscode, garantindo que o usuário vscode tenha acesso adequado aos arquivos no ambiente de desenvolvimento.

Além disso, configuramos o ambiente do Visual Studio Code dentro do contêiner usando o arquivo de configuração "devcontainer.json". Isso permite que os desenvolvedores utilizem o VS Code de forma otimizada no ambiente de desenvolvimento.

Figura 12 - Configurações do VSCode

```
{
    "name": "Workbench",
    "remoteUser": "vscode",
    "dockerComposeFile": ["../docker-compose.yml"],
    "service": "devcontainer",
    "workspaceFolder": "/sistenda",
    "customizations": {
      "vscode": {
        "extensions": [
      "ms-azuretools.vscode-docker",
      "dbaeumer.vscode-eslint",
      "esbenp.prettier-vscode",
      "dsznajder.es7-react-js-snippets",
      "bierner.markdown-mermaid",
      "ms-vsliveshare.vsliveshare",
      "ms-vscode.makefile-tools",
      "DavidAnson.vscode-markdownlint",
      "Prisma.prisma",
      "pflannery.vscode-versionlens",
      "cweijan.vscode-postgresql-client2"
    ],
        "settings": {
          "files.eol": "\n",
          "search.exclude": {
            "**/.next": true
          },
          "[javascript]": {
            "editor.defaultFormatter": "esbenp.prettier-vscode",
            "editor.formatOnSave": true
          },
          "typescript.tsdk": "/sis-tenda/node_modules/typescript/lib/",
          "[typescript]": {
            "editor.defaultFormatter": "esbenp.prettier-vscode",
            "editor.formatOnSave": true
          },
          "[typescriptreact]": {
            "editor.defaultFormatter": "esbenp.prettier-vscode",
            "editor.formatOnSave": true
          },
          "editor.codeActionsOnSave": {
            "source.fixAll.eslint": true,
            "source.fixAll": true,
            "source.removeUnusedImports": true
          },
          "eslint.format.enable": true,
          "eslint.validate": [
            "javascript",
            "typescript",
            "typescriptreact"
```

```
        ],
        "eslint.workingDirectories": [
          "./frontend",
          "./backend"
      ]
      }
    }
  }
}
```

Fonte: Os autores, 2023

Para gerenciar os processos da aplicação, utilizamos o PM2 e configuramos dois aplicativos ("apps") no arquivo "pm2.json", garantindo que a aplicação seja executada de forma robusta e resiliente.

Figura 13 - Aplicações do PM2

```
{
  "apps" : [
    {
      "name": "frontend",
      "script": "set -a; source .env; cd ./frontend && yarn dev --port
3333",
      "watch": ".env",
      "ignore_watch": []
    },
    {
      "name": "backend",
      "script": "set -a; source .env; cd ./backend && yarn start:dev --
port 9001",
      "watch": ".env",
      "ignore_watch": []
    }
  ]
}
```

Fonte: Os autores, 2023

O arquivo "docker-compose.yml", localizado na raiz do projeto, declara e configura os serviços necessários para o projeto, incluindo o Traefik para roteamento, o dev container para ambiente de desenvolvimento e o banco de dados.

Figura 14 - Docker compose do dev container

```yaml
version: '3.2'

services:
  traefik:
    image: traefik:v3.0
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./.devcontainer/traefik/:/etc/traefik/
    environment:
      - TRAEFIK_LOG_LEVEL=DEBUG
      - TRAEFIK_ENTRYPOINTS_WEB_ADDRESS=:80
      - TRAEFIK_ENTRYPOINTS_WEBSECURE_ADDRESS=:443
      - TRAEFIK_ENTRYPOINTS_WEBSECURE_HTTP_TLS=true
      - TRAEFIK_PROVIDERS_DOCKER=true
      - TRAEFIK_PROVIDERS_FILE=true
      - TRAEFIK_PROVIDERS_FILE_DIRECTORY=/etc/traefik/
    depends_on:
      - devcontainer

  devcontainer:
    build:
      context: ./
      dockerfile: .devcontainer/Dockerfile
    container_name: devcontainer_sistenda
    env_file: .env
    image: sistenda/devcontainer_sistenda:latest
    working_dir: /sistenda
    tty: true
    user: vscode:vscode
    shm_size: "4gb"
    volumes:
      - ./:/sistenda/
      - /var/run/docker.sock:/var/run/docker.sock
      - ~/.ssh:/home/vscode/.ssh
    depends_on:
      - database
    labels:
      - "traefik.enable=true"
      - "traefik.http.middlewares.redirect.redirectscheme.scheme=https"
      - "traefik.http.middlewares.redirect.redirectscheme.permanent=true"

      - "traefik.http.routers.app.entrypoints=web"
      - "traefik.http.routers.apps.entrypoints=websecure"
      - "traefik.http.routers.app.rule=Host(`dev.sistenda.com`)"
      - "traefik.http.routers.apps.rule=Host(`dev.sistenda.com`)"
      - "traefik.http.routers.app.middlewares=redirect"
```

```
      - "traefik.http.routers.apps.tls=true"
      - "traefik.http.routers.app.service=www"
      - "traefik.http.routers.apps.service=www"
      - "traefik.http.services.www.loadbalancer.server.port=3333"

  database:
    container_name: database_sistenda
    image: postgres:15
    env_file: .env
```

Fonte: Os autores, 2023

Também configuramos o LeftHook, garantindo que o "*pre-commit*" seja executado para verificar e garantir que o código enviado para o repositório não contenha erros de lint e TypeScript.

Figura 15 - Configuração do pre-commit

```
pre-commit:
  parallel: true

  commands:

    lint_frontend:
      root: frontend
      glob: "frontend/**/*.{js,ts,jsx,tsx}"
      run: yarn eslint {staged_files}
      stage_fixed: true

    lint_backend:
      root: backend
      glob: "backend/**/*.{js,ts,jsx,tsx}"
      run: yarn eslint {staged_files}
      stage_fixed: true

    typecheck_frontend:
      root: frontend
      glob: "*.{js,ts,jsx,tsx}"
      run: echo {staged_files} > /dev/null ; yarn typecheck;

    typecheck_backend:
      root: backend
      glob: "*.{js,ts,jsx,tsx}"
      run: echo {staged_files} > /dev/null ; yarn typecheck;
```

Fonte: Os autores, 2023

Por fim, o arquivo "Makefile" oferece comandos essenciais para o desenvolvimento, como o "*bootstrap*" para baixar as dependências do projeto, instalar o LeftHook, realizar migrações no banco de dados e popular o banco.

O comando "update" atualiza as dependências do projeto, e o comando "server" inicia os serviços do projeto conforme descrito no arquivo "pm2.json". Essa configuração torna o ambiente de desenvolvimento eficiente e facilita o desenvolvimento e o gerenciamento do projeto.

Figura 16 - Configuração do Make file

```
help:
    @echo
    @echo "  888888 888888 88b 88 8888b.     db
    @echo "    88   88__   88Yb88  8I  Yb   dPYb
    @echo "    88   88""   88 Y88  8I  dY  dP__Yb
    @echo "    88   888888 88  Y8 8888Y"  dP""""Yb
    @echo
    @echo "  bootstrap      Install dependencies"
    @echo "  update         Update dependencies"
    @echo "  start          Start all services"
    @echo


bootstrap:
    @yarn install --non-interactive
    @lefthook install
    @npx prisma migrate dev --name init --
schema=backend/prisma/schema.prisma
    @npx ts-node backend/prisma/script.ts

update:
    @yarn install --non-interactive

server:
    @pm2 start .devcontainer/pm2.json
```

Fonte: Os autores, 2023

Após a configuração bem-sucedida do ambiente de desenvolvimento, direcionamos nossos esforços para o desenvolvimento do *backend* do projeto. Inicialmente, garantimos que todas as dependências e devDependencies necessárias fossem devidamente instaladas, conforme especificado em nosso arquivo *package.json* do *backend*.

Figura 17 - Dependências do back end

```
"dependencies": {
  "@nestjs/common": "^10.0.0",
  "@nestjs/core": "^10.0.0",
  "@nestjs/platform-express": "^10.0.0",
  "@prisma/client": "^5.2.0",
  "reflect-metadata": "^0.1.13",
  "rxjs": "^7.8.1",
  "ts-node": "^10.9.1"
},
"devDependencies": {
  "@nestjs/cli": "^10.0.0",
  "@nestjs/schematics": "^10.0.0",
  "@nestjs/testing": "^10.0.0",
  "@types/express": "^4.17.17",
  "@types/jest": "^29.5.2",
  "@types/node": "^20.5.9",
  "@types/supertest": "^2.0.12",
  "@typescript-eslint/eslint-plugin": "^6.0.0",
  "@typescript-eslint/parser": "^6.0.0",
  "eslint": "^8.42.0",
  "eslint-config-prettier": "^9.0.0",
  "eslint-plugin-prettier": "^5.0.0",
  "jest": "^29.5.0",
  "prettier": "^3.0.0",
  "prisma": "^5.2.0",
  "source-map-support": "^0.5.21",
  "supertest": "^6.3.3",
  "ts-jest": "^29.1.0",
  "ts-loader": "^9.4.3",
  "ts-node": "^10.9.1",
  "tsconfig-paths": "^4.2.0",
  "typescript": "^5.2.2"
},
```

Fonte: Os autores, 2023

Em seguida, configuramos o *schema* do banco de dados utilizando o Prisma, permitindo-nos definir a estrutura e as relações das tabelas de forma eficiente.

Figura 18 - Schema do banco de dados

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model Cidade {
  id String @id @default(uuid()) @db.Uuid

  cidade String
  estado String

  paroquia Paroquia[]
  casal    Casal[]

  @@map("cidade")
}

model Imagem {
  id   String @id @default(uuid()) @db.Uuid
  foto Bytes

  pessoa Pessoa[]

  @@map("imagem")
}
```

Fonte: Os autores, 2023

Para automatizar o processo de população do banco de dados, desenvolvemos um script customizado que é executado sempre que executamos o comando *"make bootstrap".*

Figura 19 - Comando "make bootstrap"

```
bootstrap:
    @yarn install --non-interactive
    @lefthook install
    @npx prisma migrate dev --name init --
schema=backend/prisma/schema.prisma
    @npx ts-node backend/prisma/script.ts
```

Fonte: Os autores, 2023

Figura 20 - Script para popular o banco de dados

```
import { PrismaClient } from "@prisma/client";
import { citiesMock, confidenteMock, peopleMock } from "./mock";

const prisma = new PrismaClient();

async function main() {
  const havaCities = await prisma.cidade.findMany();
  if (havaCities.length === 0) {
    await prisma.cidade.createMany({
      data: citiesMock,
    });
  }
  const cities = await prisma.cidade.findMany();
}

main()
  .then(async () => {
    await prisma.$disconnect();
  })
  .catch(async (e) => {
    // eslint-disable-next-line no-console
    console.error(e);
    await prisma.$disconnect();
    process.exit(1);
  });
```

Fonte: Os autores, 2023

Com a base de dados pronta, prosseguimos para o desenvolvimento da validação do Json Web Token. Utilizamos o framework Passport, Passporte-jwt e RxJS para construir a estratégia do JWT.

Figura 21 - Código para validação do Json Web Token

```
import { Injectable } from "@nestjs/common";
import { PassportStrategy } from "@nestjs/passport";
import * as dotenv from "dotenv";
import { passportJwtSecret } from "jwks-rsa";
import { ExtractJwt, Strategy } from "passport-jwt";

dotenv.config();

@Injectable()
export class JwtStrategy extends PassportStrategy(Strategy) {
  constructor() {
    super({
      secretOrKeyProvider: passportJwtSecret({
        cache: true,
        rateLimit: true,
        jwksRequestsPerMinute: 5,
        jwksUri: `https://sistenda.us.auth0.com/.well-known/jwks.json`,
      }),

      jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
      audience: "https://sistenda.us.auth0.com/api/v2/",
      issuerBaseURL: "https://sistenda.us.auth0.com/",
      tokenSigningAlg: "RS256",
    });
  }

  validate(payload: unknown): unknown {
    return payload;
  }
}
```

Fonte: Os autores, 2023

Este código configura uma estratégia de autenticação JWT que verifica tokens JWT nas solicitações HTTP, usando as chaves JWKS de um provedor externo (Auth0, no caso). Quando um token é validado com sucesso, as informações contidas no token são retornadas para serem utilizadas no aplicativo, como identificação do usuário ou outras informações relevantes.

Com a autenticação pronta, prosseguimos para o desenvolvimento das APIs. Utilizamos o framework Nest JS, que segue a arquitetura MVC (Model-View-Controller), para criar nossos *controllers, modules* e *services*, tornando a organização do código mais eficiente e modular.

Figura 22 - Controller para o CRUD de usuários

```
import {
  Body,
  Controller,
  Delete,
  Get,
  Param,
  Patch,
  Post,
  Res,
  UseGuards,
} from "@nestjs/common";
import { AuthGuard } from "@nestjs/passport";
import { Pessoa } from "@prisma/client";
import { Response } from "express";
import { JSendService } from "src/common/http";
import { UsersService } from "./users.service";

@Controller()
export class UsersController {
  constructor(
    private readonly prismaService: UsersService,
    private readonly jSendService: JSendService
  ) {}

  @UseGuards(AuthGuard("jwt"))
  @Get()
  async getAllUsers(@Res() res: Response) {
    const result = await this.prismaService.pessoa.findMany();
    if (result) return this.jSendService.sendResponse(res, result);
    return this.jSendService.sendResponse(res, null, 404);
  }

  @UseGuards(AuthGuard("jwt"))
  @Get("/me/auth0/:auth0id")
  async getUserByAuth0Id(@Param("auth0id") id: string, @Res() res:
Response) {
    const result = await this.prismaService.pessoa.findUnique({
      where: {
        id: id,
      },
    });
    if (result) return this.jSendService.sendResponse(res, result);
    return this.jSendService.sendResponse(res, null, 404);
  }

  @UseGuards(AuthGuard("jwt"))
  @Get("/me/email/:email")
  async getUserByEmail(@Param("email") email: string, @Res() res:
Response) {
```

Fonte: Os autores, 2023

Figura 23 - Módulo para a api dos usuários

```
import { Module } from "@nestjs/common";
import { JSendService } from "src/common/http";
import { UsersController } from "./users.controller";
import { UsersService } from "./users.service";

@Module({
  imports: [],
  controllers: [UsersController],
  providers: [UsersService, JSendService],
})
export class UsersModule {}
```

Fonte: Os autores, 2023

Figura 24 - Serviços para a api de usuários

```
import { Injectable, OnModuleInit } from "@nestjs/common";
import { PrismaClient } from "@prisma/client";

@Injectable()
export class UsersService extends PrismaClient implements OnModuleInit {
  async onModuleInit() {
    await this.$connect();
  }
}
```

Fonte: Os autores, 2023

Seguimos a mesma lógica e estrutura de código ao criar os outros CRUD's do *backend*, garantindo uma abordagem consistente e organizada no desenvolvimento das APIs.

Neste projeto, o desenvolvimento do pipeline no Turbo Repo desempenha um papel fundamental no ciclo de desenvolvimento. O pipeline é projetado com foco na eficiência e automação, abrangendo diversas etapas essenciais. Isso inclui a compilação, testes, análise de lint, construção de artefatos e implantação.

O pipeline segue uma lógica bem definida, garantindo que as etapas sejam executadas em ordem, com dependências claras. Além disso, a configuração do cache e a persistência são otimizadas para melhorar a velocidade e a confiabilidade do processo de desenvolvimento. Dessa forma, o pipeline Turbo Repo se torna uma peça-chave no desenvolvimento contínuo e na implantação bem-sucedida do projeto,

simplificando o fluxo de trabalho e melhorando a qualidade do código resultante.

Figura 25 - Pipeline

```
{
    "$schema": "https://turbo.build/schema.json",
    "pipeline": {
      "build": {
        "dependsOn": ["^build"],
        "outputs": [".next/**", "!.next/cache/**"]
      },
      "deploy": {
        "dependsOn": ["build", "test", "lint"]
      },
      "test": {
        "dependsOn": ["build"],
        "inputs": ["src/**/*.tsx", "src/**/*.ts", "test/**/*.ts",
"test/**/*.tsx"]
      },
      "lint": {},
      "dev": {
        "cache": false,
        "persistent": true
      }
    }
  }
```

Fonte: Os autores, 2023

No desenvolvimento do *frontend*, começamos pela criação da estrutura e do fluxo de trabalho do projeto. Optamos por usar criadores de páginas personalizados no Next.js para garantir maior flexibilidade ao escolher entre o *server side* ou *client side rendering*, dependendo dos requisitos de cada página. Esta abordagem nos permite otimizar o desempenho da aplicação de acordo com a necessidade.

Figura 26 - Hook para criação de páginas

```
import Head from "next/head";
import { useRouter } from "next/router";
import { NextPage } from "next/types";

export interface LayoutProps {
  layout?: React.ReactNode;
  isPublic?: boolean;
}

export interface CreatePageProps<TPageProps extends object>
  extends LayoutProps {
  title?: string;
  renderComponent: React.FC<PageProps & TPageProps>;
}

export type NextPageWithLayout<P = object, IP = P> = NextPage<P, IP> &
  LayoutProps;

export interface NavigateOptions {
  replace?: boolean;
}

export interface PageProps {
  /* eslint-disable-next-line */
  query: Record<string, any>;
  locale?: string;
}

/**
 * @todo implement features/roles/auth/layouts/data loading?
 *
 * Inspired by
 * https://blog.rstankov.com/structuring-next-js-application/
 */
export const createPage = <TPageProps extends object>(
  props: CreatePageProps<TPageProps>,
) => {
  const { title, layout, isPublic, renderComponent: PageComponent } =
props;

  const Page: NextPageWithLayout<TPageProps> = (innerProps) => {
    const router = useRouter();

    return (
      <>
        <Head>
          <title>{title}</title>
        </Head>
```

Fonte: Os autores, 2023

O código começa importando vários módulos do Next.js, como Head, useRouter, NextPage, e outras dependências necessárias. Em seguida, são definidos interfaces e tipos TypeScript que são usados para tipar os componentes e propriedades utilizadas no projeto.

A função *createPage* é uma função que recebe um conjunto de props, incluindo título da página, layout, se a página é pública ou não, e um componente de renderização (*renderComponent*). Essa função é responsável por criar uma página no Next.js com as configurações fornecidas.

O componente Page é uma página do Next.js que incorpora o layout e outras configurações especificadas na função *createPage*. Ele recebe as propriedades padrão do Next.js, como *query* e *locale*, e passa essas informações para o componente de renderização (*PageComponent*).

O código também atribui o layout e a visibilidade pública da página às propriedades do componente Page, permitindo que essas configurações sejam definidas ao criar uma página usando a função *createPage*.

Figura 27 - Criação de página utilizando o hook customizado

```
import { createPage } from "@/components/create-page";
import { Profile } from "@/screens/profile/view";

export default createPage({
  title: "Perfil",
  layout: "fullscreen",
  renderComponent: () => <Profile />,
});
```

Fonte: Os autores, 2023

O código final demonstra como usar a função *createPage* para criar uma página chamada "Perfil" com um layout em tela cheia ("*fullscreen*") e um componente de renderização chamado Profile. Isso ilustra como a função *createPage* é usada para configurar rapidamente páginas com diferentes layouts e componentes de renderização.

Adicionalmente, implementamos nossos próprios *hooks* personalizados para lidar com as requisições HTTP. Isso nos permitiu manter um código de alta qualidade, garantir a segurança adequada e fornecer suporte para escopos e tokens JWT (*JSON*

Web Token). Também criamos um *hook* de inicialização do projeto, pois, ao registrar e fazer login com o Auth0, precisamos manter os dados do usuário em nosso próprio banco de dados. Dessa forma, quando um usuário se registra ou faz login, o sistema verifica se já possui um perfil desse usuário em nosso banco e, se não, cria um.

Figura 28 - Hook para realização de requests

```
import fetch from "isomorphic-unfetch";

type RequestMethod = "GET" | "POST" | "PATCH" | "DELETE";

interface HTTPRequest {
  body?: string;
  method: RequestMethod;
  headers: { [key: string]: string };
}

interface HTTPResponse<T> {
  status?: "success" | "fail" | "error";
  message?: string;
  data?: T;
}

interface RequestBody {
  [key: string]: string | number | boolean | RequestBody;
}

interface RequestOptions {
  jwt?: string;
  body?: RequestBody;
}

const wrappedFetch = (endpoint: string, request: HTTPRequest) => {
  return fetch(`http://localhost:9001${endpoint}`, request);
};

class RequestError extends Error {
  constructor(
    public response: globalThis.Response,
    message?: string,
  ) {
    super(message);
  }
}

const buildRequest = (
  method: RequestMethod,
  body: RequestBody,
  token: string | undefined,
): HTTPRequest => {
  const request: HTTPRequest = {
    method,
    headers: {
      "Content-Type": "application/json",
    },
  };
```

Fonte: Os autores, 2023

O código apresentado define um conjunto de funções e interfaces para realizar solicitações HTTP em um aplicativo. Ele utiliza a biblioteca isomorphic-unfetch para efetuar as chamadas de rede de forma assíncrona. As funções exportadas, como get, post e patch, são métodos convenientes para realizar solicitações HTTP comuns, como GET, POST e PATCH, juntamente com opções personalizadas, como um token JWT e um corpo de solicitação.

O código também trata respostas de servidor, lançando exceções *RequestError* em caso de erro na resposta. Essa estrutura permite que o desenvolvedor realize chamadas de API de maneira simples e eficaz, abstraindo muitos detalhes da implementação HTTP e lidando com possíveis erros de maneira consistente.

Em seguida, implementamos as rotas de autenticação do usuário, que redirecionam o usuário para o Auth0, juntamente com a configuração do NextAuth.js. Isso garantiu que usuários não autenticados fossem redirecionados para o Auth0, onde realizavam a autenticação. Após o login bem-sucedido, persistíamos o token de acesso do Auth0 de forma segura nos *cookies*, garantindo sua criptografia.

Figura 29 - API para autenticação

```
import NextAuth, { Session } from "next-auth";
import Auth0Provider from "next-auth/providers/auth0";

export default NextAuth({
  cookies: {
    sessionToken: {
      name: "__Secure-next-auth.session-token",
      options: {
        httpOnly: true,
        sameSite: "lax",
        path: "/",
        secure: true,
        domain: "localhost",
      },
    },
    callbackUrl: {
      name: "__Secure-next-auth.callback-url",
      options: {
        sameSite: "lax",
        path: "/",
        secure: true,
        domain: "localhost",
      },
    },
    pkceCodeVerifier: {
      name: "__Secure-next-auth.pkce.code_verifier",
      options: {
        httpOnly: true,
        sameSite: "lax",
        path: "/",
        secure: true,
        domain: "localhost",
      },
    },
    state: {
      name: "__Secure-next-auth.state",
      options: {
        httpOnly: true,
        sameSite: "lax",
        path: "/",
        secure: true,
        domain: "localhost",
      },
    },
  },
  providers: [
    Auth0Provider({
      clientId: process.env.AUTH0_CLIENT_ID as string,
      clientSecret: process.env.AUTH0_CLIENT_SECRET as string,
```

Fonte: Os autores, 2023

Posteriormente, construímos várias telas da aplicação utilizando componentes do Chakra UI e SaaS UI para garantir uma interface de usuário atraente e consistente.

Figura 30 - Código da página de Perfil

```
import { HStack, Heading, StackDivider, Text } from "@chakra-ui/react";
import MyProfileForm from "./components/form";

const Profile = () => {
  return (
    <HStack display="flex" flexDir="column" alignItems="flex-start"
w="65%">
      <Heading as="h1" size="lg">
        Perfil
      </Heading>
      <Text>Gerencie seu perfil</Text>
      <StackDivider backgroundColor="gray.100" h="1px" mt="2" mb="2" />
      <MyProfileForm />;
    </HStack>
  );
};

export { Profile };
```

Fonte: Os autores, 2023

Por fim, realizamos a integração entre o *frontend* e o *backend* usando nossos próprios *hooks* personalizados. Isso nos permitiu buscar e exibir dados do *backend* de forma eficiente e coesa.

Figura 31 - Código do formulário da tela de perfil

```
export default function MyProfileForm() {
  const session = useSession();
  const { data } = session;

  const onSubmit = (params: any) => {
    updateUserMutation.mutate({ ...params, cpf, phone });
  };

  const currentUser = useCustomQuery<Pessoa>({
    queryKey: ["user", session?.data?.user?.email as string],
    custom: getCurrentUserQuery(session.data?.user?.email as string),
    enabled: !!session?.data?.user?.email,
  });

  const updateUserMutation = useMutation({
    mutationKey: ["updateUser"],
    mutationFn: ({ cpf, name, phone }: SchemaFormData) =>
      updateUser(
        currentUser.data?.id || "",
        {
          nome: name,
          cpf,
          telefone: phone,
        },
        data?.accessToken || "",
      ),
  });

  return (
    <Form onSubmit={onSubmit} w="100%" schema={schema}>
      {({ Field }) => (
        <FormLayout>
        //restante do código
```

Fonte: Os autores, 2023

Com essas etapas concluídas, estabelecemos a base sólida necessária para desenvolver as demais telas e funcionalidades da aplicação.

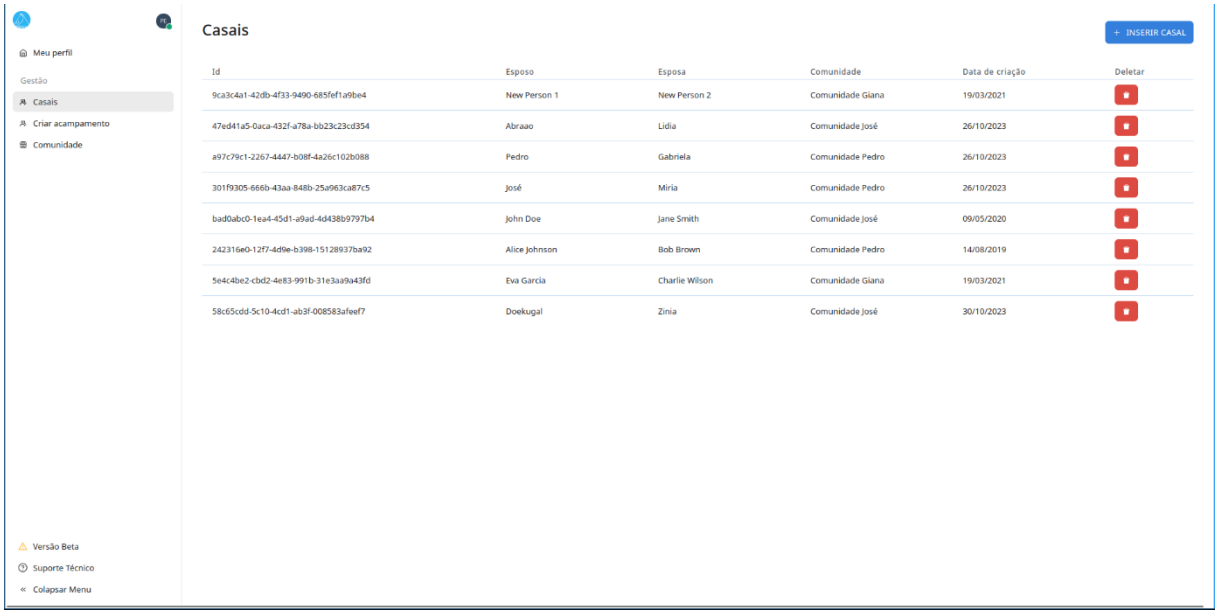
A seguir, apresentaremos uma visão geral das telas que compõem o sistema, destacando seu design, funcionalidade e usabilidade. Cada tela foi desenvolvida com foco na experiência do usuário, e isso é evidenciado em cada aspecto do layout. Vamos explorar as principais telas do sistema:

Cadastrar Casal

Nesta seção da aplicação, os usuários têm acesso a uma lista de todos os casais registrados no sistema. A tela oferece uma visão geral dos casais, destacando seus nomes e informações básicas. Essa funcionalidade permite que os usuários naveguem facilmente pela lista de casais existentes, facilitando a busca de informações específicas.

Para simplificar o processo de registro de novos casais, os usuários podem utilizar o botão "Criar Casal" localizado na tela de visualização. Ao clicar neste botão, um modal interativo é aberto, orientando os usuários por meio de cinco passos distintos. Esses passos incluem fornecer informações sobre o esposo, a esposa, o casal, dados de endereço e, por fim, uma etapa de confirmação.

Figura 32 – Visualizar casais

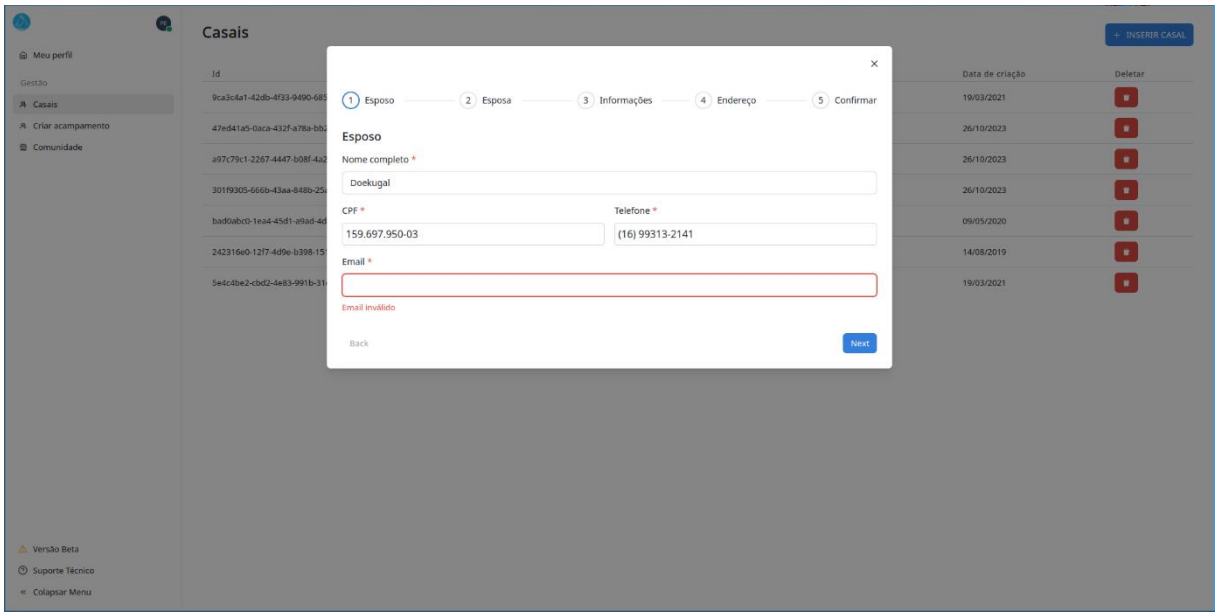


Fonte: Os autores, 2023

O modal de cinco passos foi projetado para orientar os usuários durante o processo de registro de novos casais. Ele divide a tarefa em etapas gerenciáveis, como a inclusão de informações do esposo, da esposa, do casal, dos detalhes de endereço e uma etapa final de confirmação. Essa abordagem garante que todos os dados essenciais sejam fornecidos antes da criação do casal no sistema.

Na etapa 1, o usuário insere os detalhes do esposo, fornecendo informações pessoais relevantes, como nome completo, data de nascimento e outros detalhes necessários para o registro do esposo no sistema.

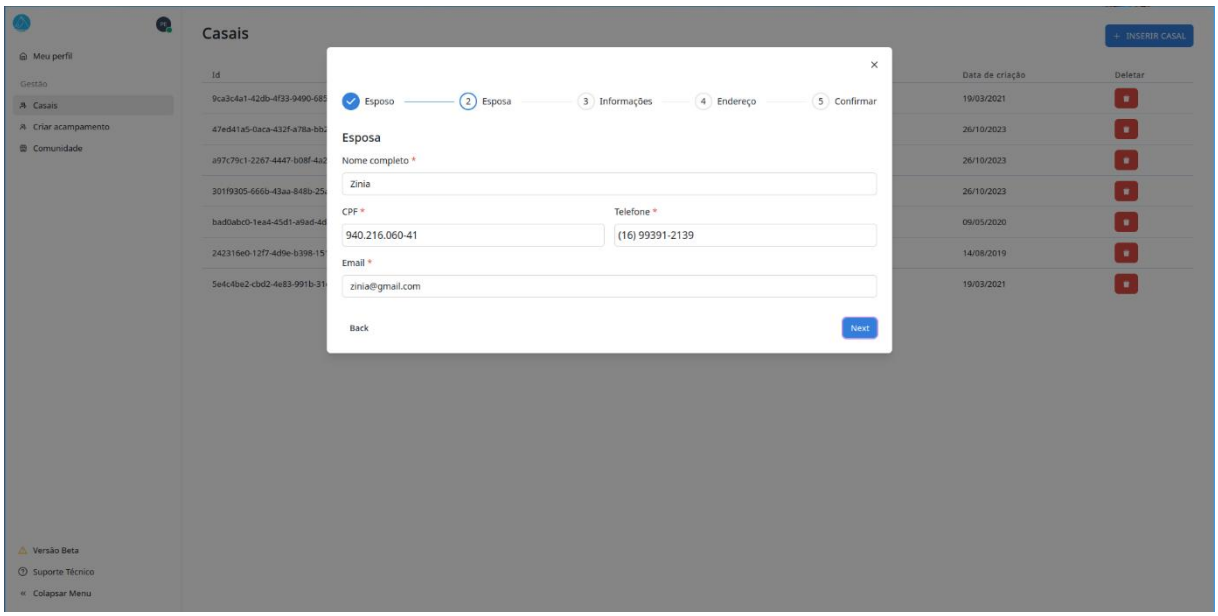
Figura 33 - Informações do esposo



Fonte: Os autores, 2023

No segundo passo, é a vez de registrar informações da esposa. O usuário fornece dados como nome completo, data de nascimento e outras informações essenciais para criar o perfil da esposa no sistema.

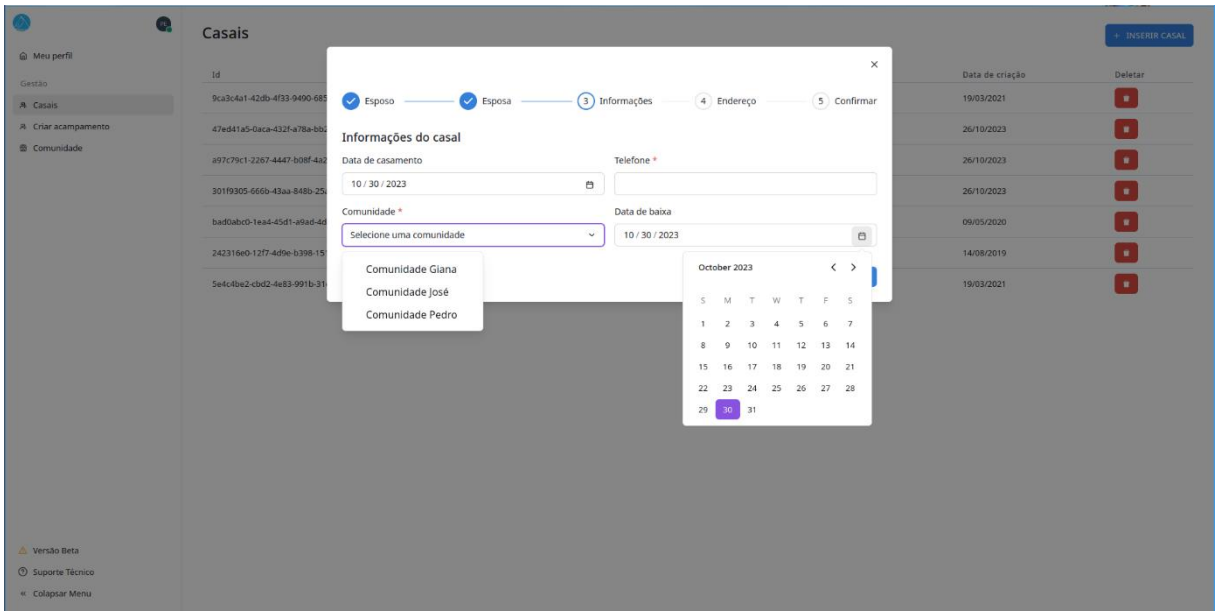
Figura 34 - Informações da esposa



Fonte: Os autores, 2023

Na terceira etapa, os usuários combinam as informações previamente fornecidas para criar um perfil de casal. Detalhes como data do casamento, aniversários do casal e outras informações relacionadas são registrados aqui.

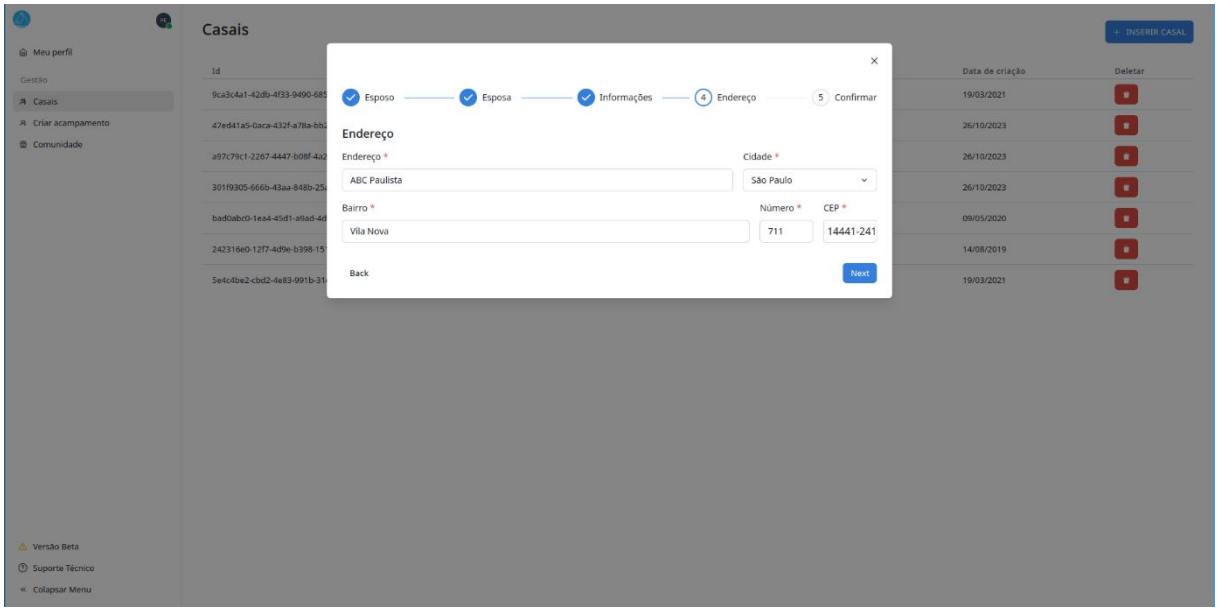
Figura 35 - Informações do casal



Fonte: Os autores, 2023

No quarto passo, é o momento de incluir informações de endereço do casal. Os usuários podem fornecer o endereço residencial, informações de contato e outras informações relevantes sobre a localização do casal.

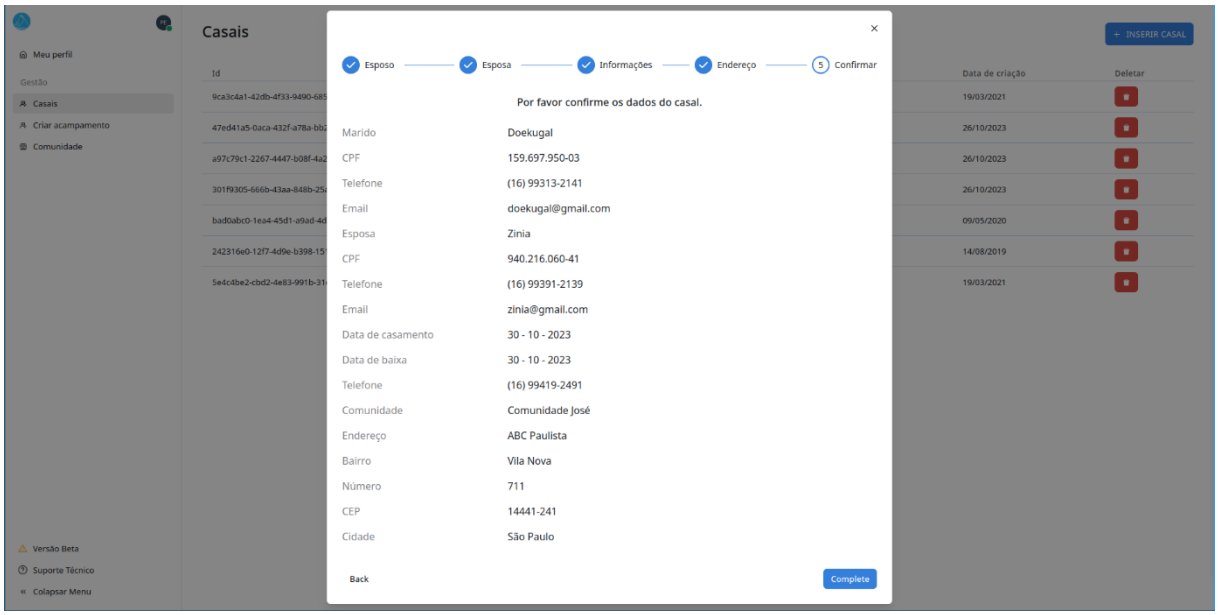
Figura 36 - Endereço



Fonte: Os autores, 2023

Nesta etapa final, os usuários revisam todas as informações inseridas nos passos anteriores. Eles podem confirmar a precisão dos dados antes de prosseguir para criar o casal no sistema.

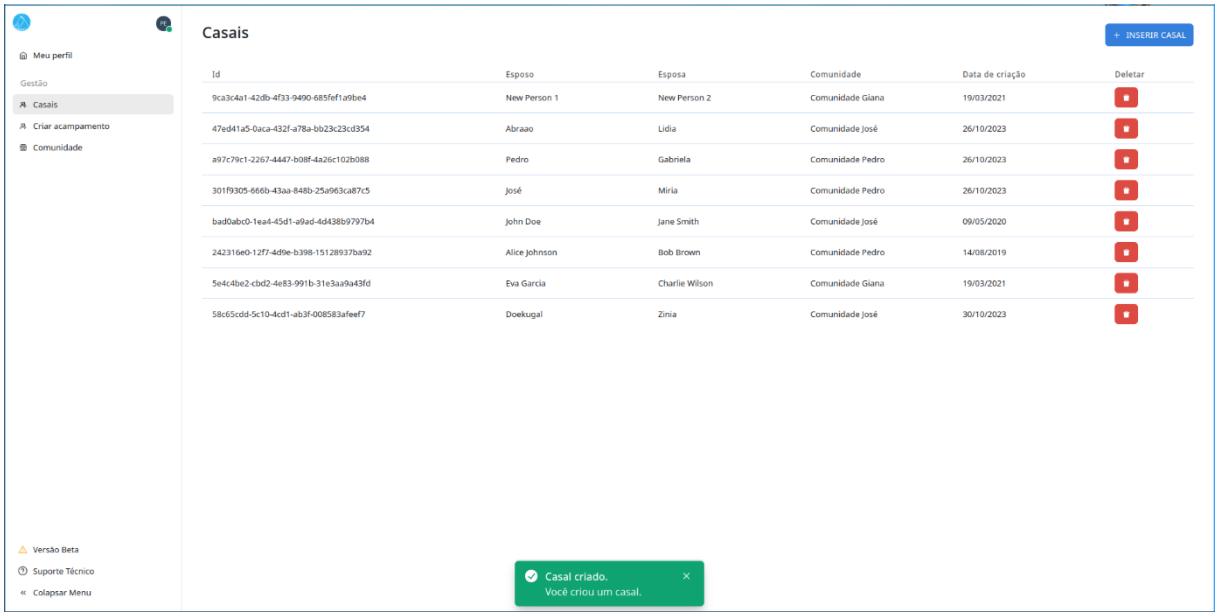
Figura 37 - Confirmar dados



Fonte: Os autores, 2023

Após a conclusão bem-sucedida dos cinco passos, os usuários receberão um feedback confirmando a criação do casal. Isso garante que o casal tenha sido registrado com sucesso no sistema e está pronto para ser listado na tela principal de casais.

Figura 38 - Feedback ao criar casal

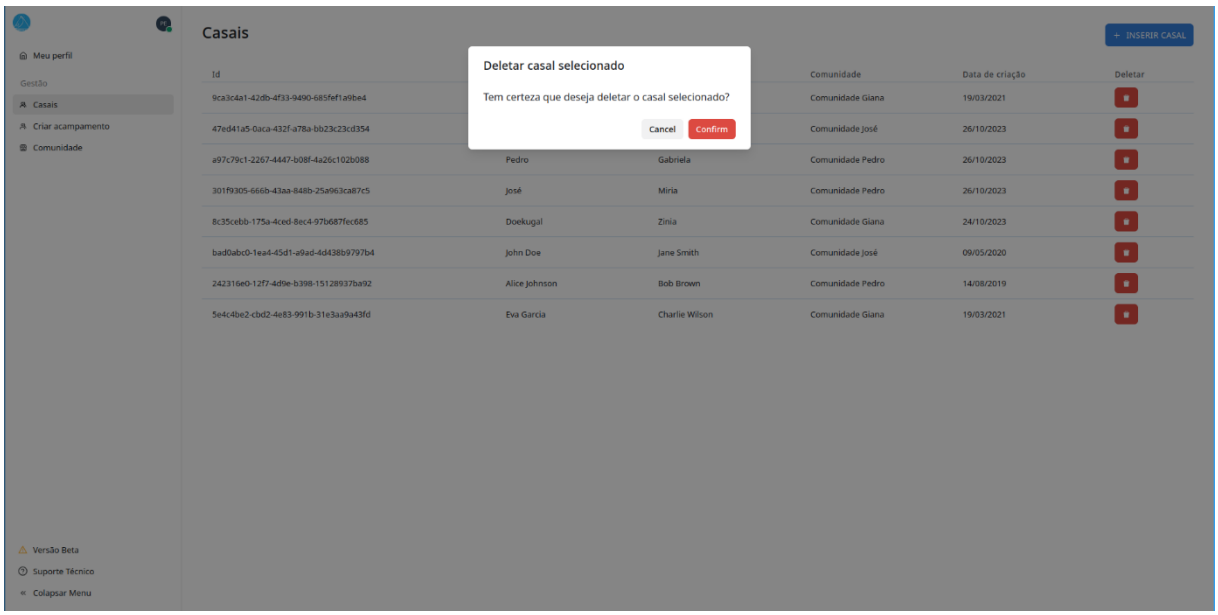


Fonte: Os autores, 2023

Durante o processo de cadastro de casais, foi implementada uma funcionalidade para a segurança e usabilidade do sistema. Ao clicar no botão de exclusão de um casal, os usuários se deparam com um modal de confirmação. Esse modal serve como uma camada adicional de proteção contra exclusões acidentais, permitindo que o usuário reveja sua ação e tome uma decisão consciente.

A confirmação por meio do modal de exclusão oferece uma experiência mais controlada, minimizando erros e garantindo que a ação de exclusão seja intencional.

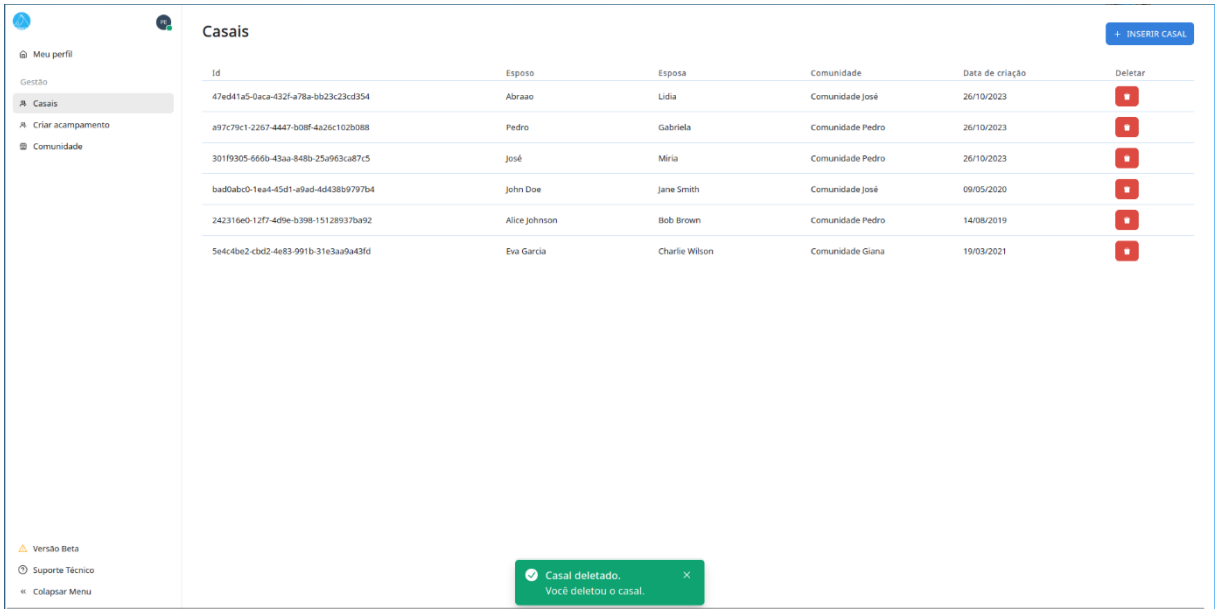
Figura 39 - Modal para deletar casal



Fonte: Os autores, 2023

Após a conclusão bem-sucedida da ação de exclusão, os usuários recebem um feedback visual que confirma a remoção do casal. Esse feedback é essencial para garantir que o usuário tenha a certeza de que o casal foi excluído com sucesso do sistema. O uso de feedback visual é uma prática recomendada em design de interface do usuário, proporcionando uma sensação de realização e transparência na interação com o sistema. Isso garante que os usuários possam confiar na precisão das operações realizadas no sistema e contribui para uma experiência de usuário eficaz e satisfatória.

Figura 40 - Feedback confirmando que o casal foi deletado



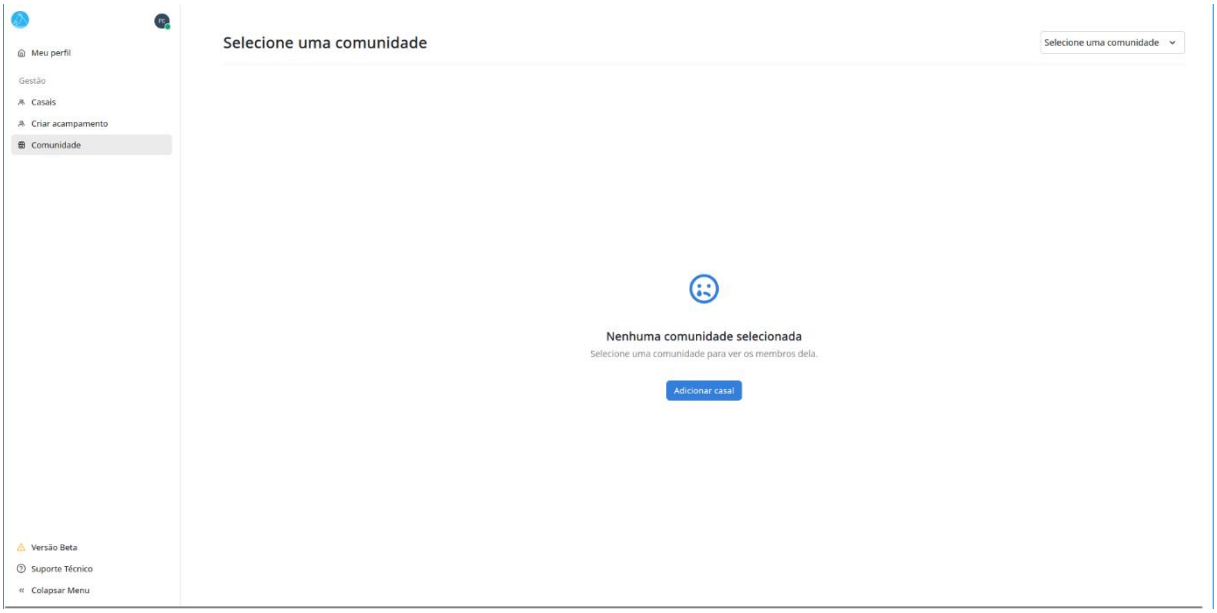
Fonte: Os autores, 2023

Visualizar Comunidade

Nesta seção da aplicação, os usuários têm acesso à visualização de membros pertencentes a diferentes comunidades. Essa funcionalidade permite que os usuários identifiquem e gerenciem os membros das comunidades com facilidade. A tela de visualização é projetada de forma intuitiva, com dois estados distintos, dependendo da seleção da comunidade.

Quando nenhum membro é selecionado em um primeiro momento, a tela exibe uma mensagem visual para informar ao usuário que nenhuma comunidade foi escolhida. Isso ajuda a evitar confusões e possibilita uma experiência clara e compreensível.

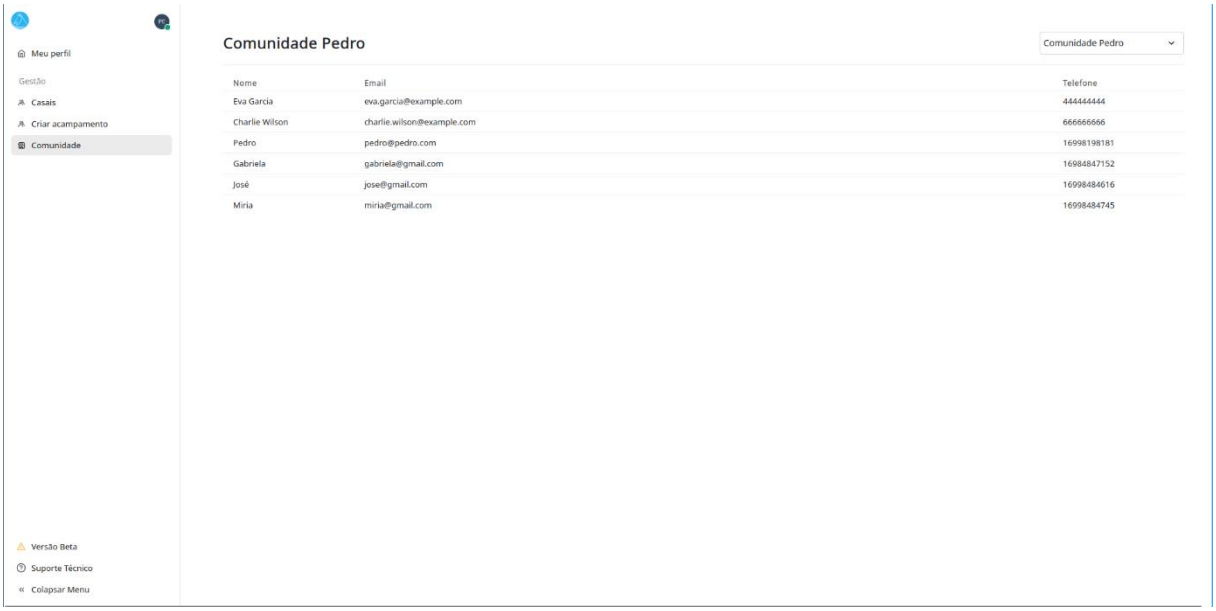
Figura 41 – Nenhuma comunidade selecionada



Fonte: Os autores, 2023

Quando uma comunidade específica é escolhida pelo usuário a partir do menu suspenso "Selecionar Comunidade," a tela passa a exibir os membros associados a essa comunidade. Nesse estado, os membros são apresentados de forma organizada, permitindo a fácil identificação de seus nomes e informações relevantes.

Figura 42 - Tela de comunidades ao selecionar comunidade

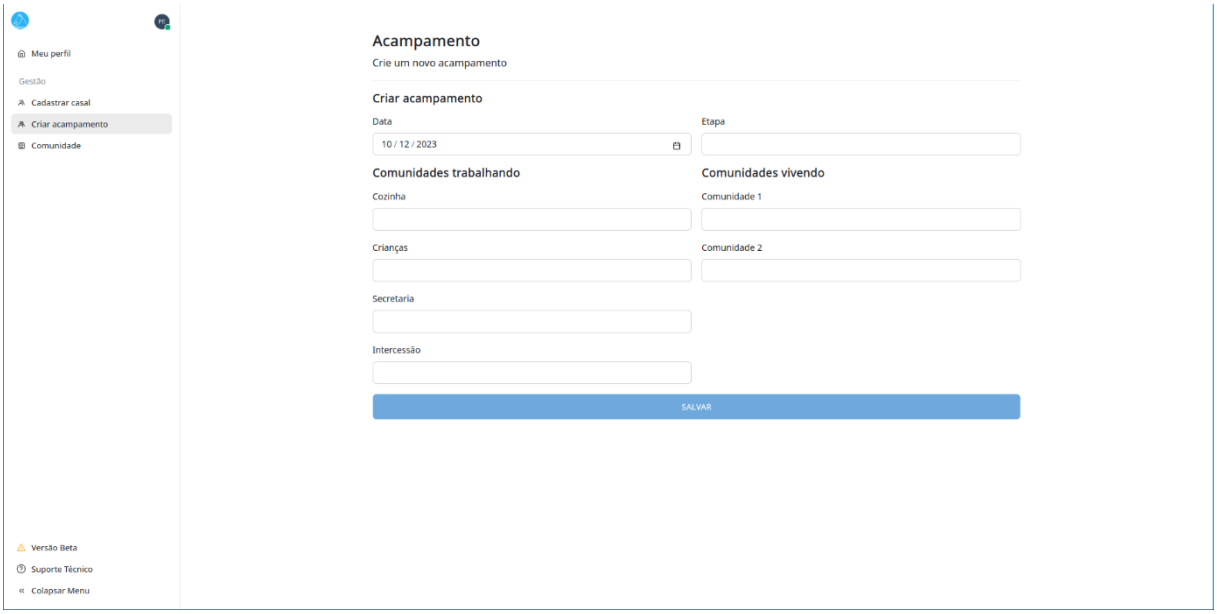


Fonte: Os autores, 2023

Acampamento

Essa tela é essencial para o gerenciamento de acampamentos, na qual permite que seja criado acampamentos.

Figura 43 - tela para criação de acampamento

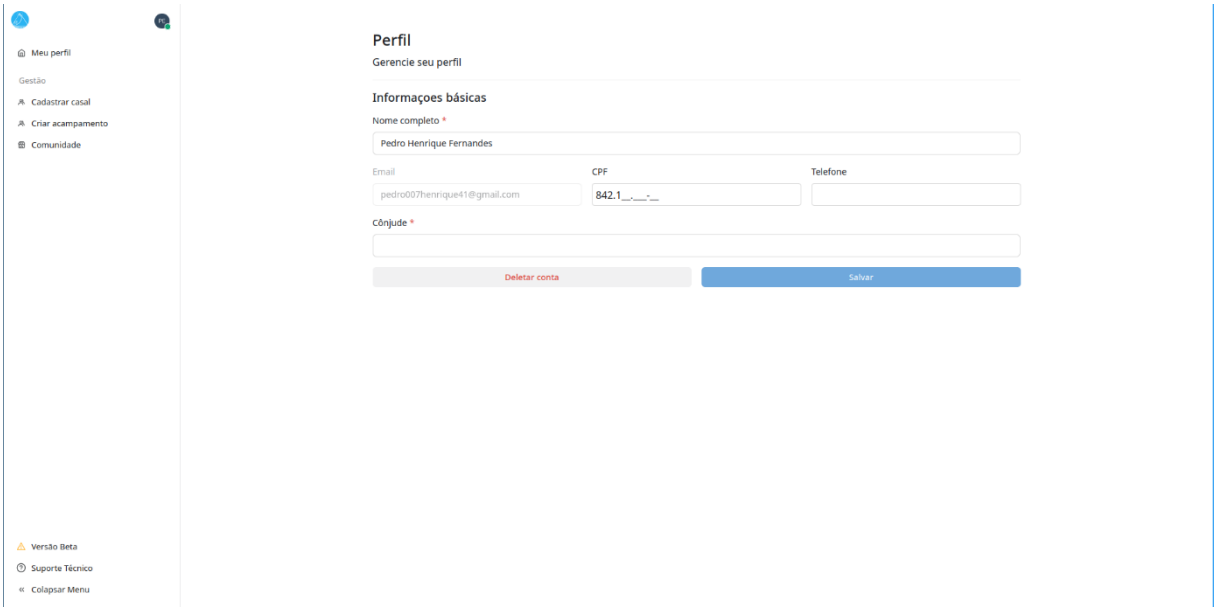


Fonte: Os autores, 2023

Meu perfil

Nesta tela, os usuários podem visualizar e editar informações pessoais, como nome e dados de contato. Essa tela fornece uma visão abrangente do perfil do usuário e permite uma fácil personalização.

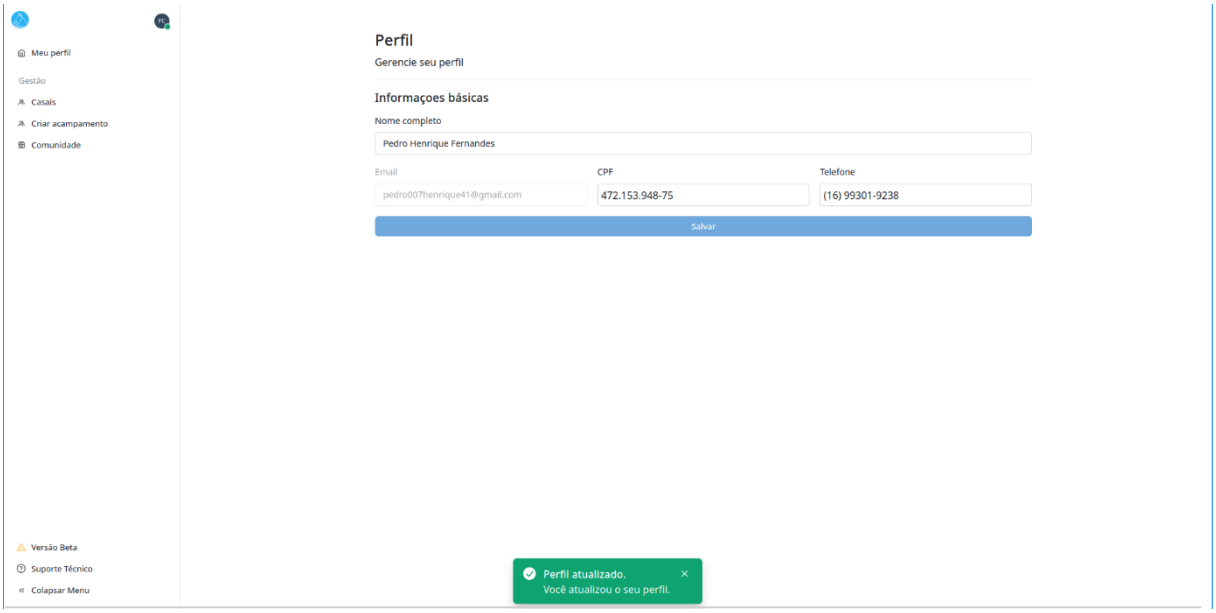
Figura 44 - Tela de meu perfil



Fonte: Os autores, 2023

Além disso, um mecanismo semelhante de feedback visual é aplicado quando os usuários fazem alterações em seus perfis, garantindo que qualquer modificação seja confirmada e refletida de forma clara no sistema.

Figura 45 - Feedback visual ao atualizar perfil

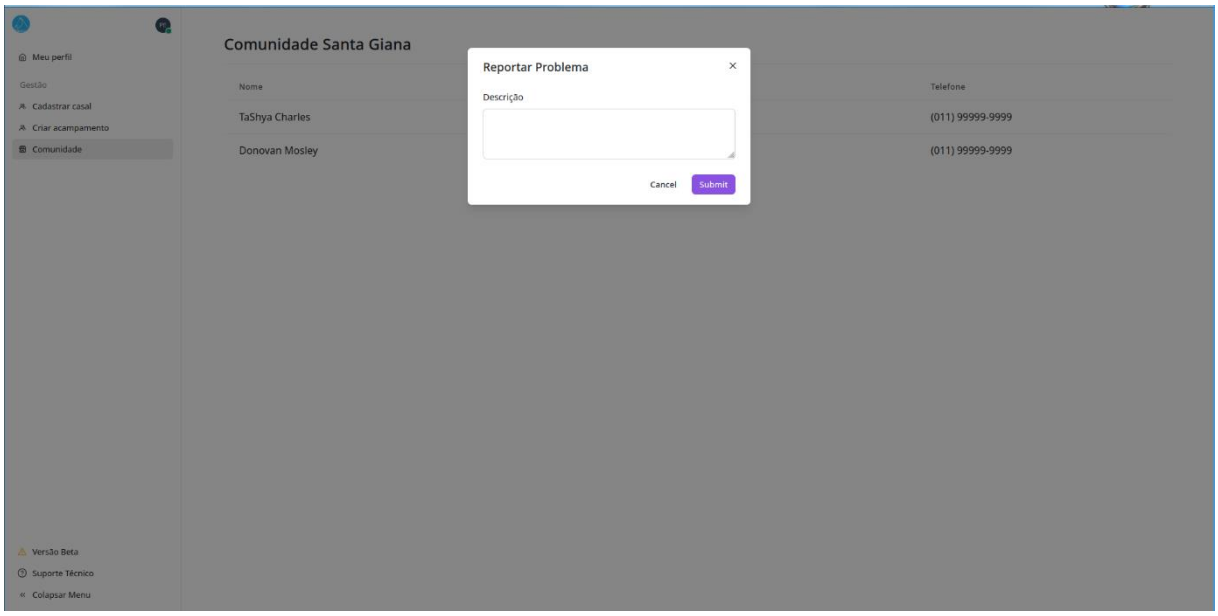


Fonte: Os autores, 2023

Modal para Reportar Problema

Acessível pela barra lateral, esse modal permite que os usuários comuniquem problemas, bugs ou preocupações à equipe de suporte. É um canal importante para manter a qualidade da aplicação.

Figura 46 - Tela para reportar problema

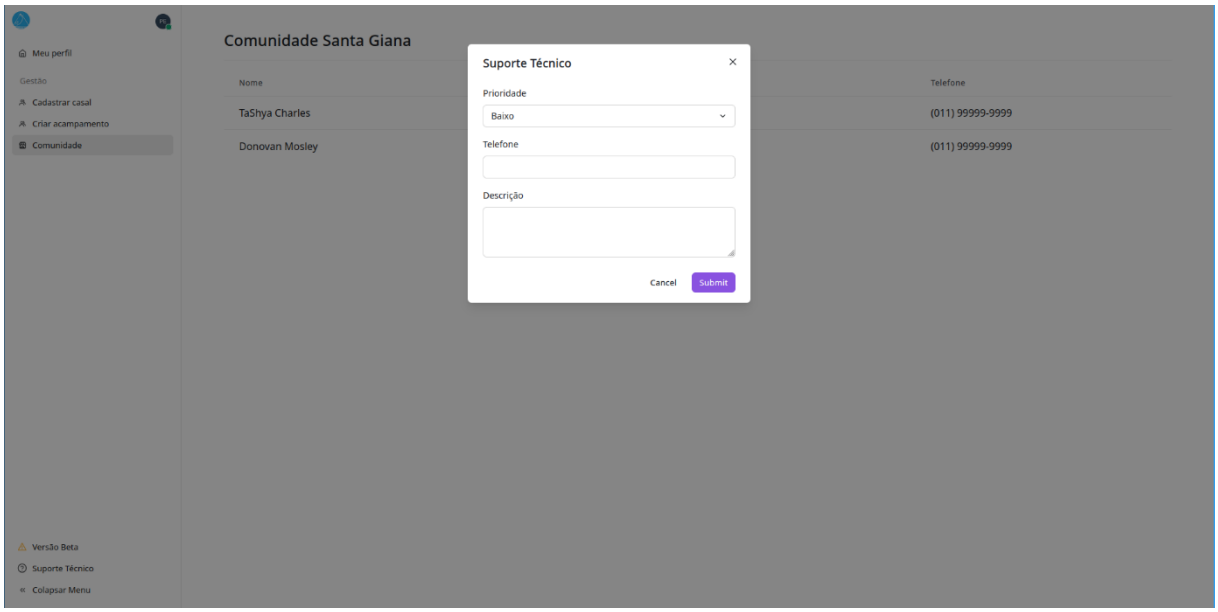


Fonte: Os autores, 2023

Modal para Contatar Suporte Técnico:

Outro recurso acessível pela barra lateral, esse modal é voltado para a comunicação direta com a equipe de suporte técnico. Isso proporciona aos usuários uma maneira eficaz de buscar ajuda quando necessário.

Figura 47 - Tela de suporte técnico

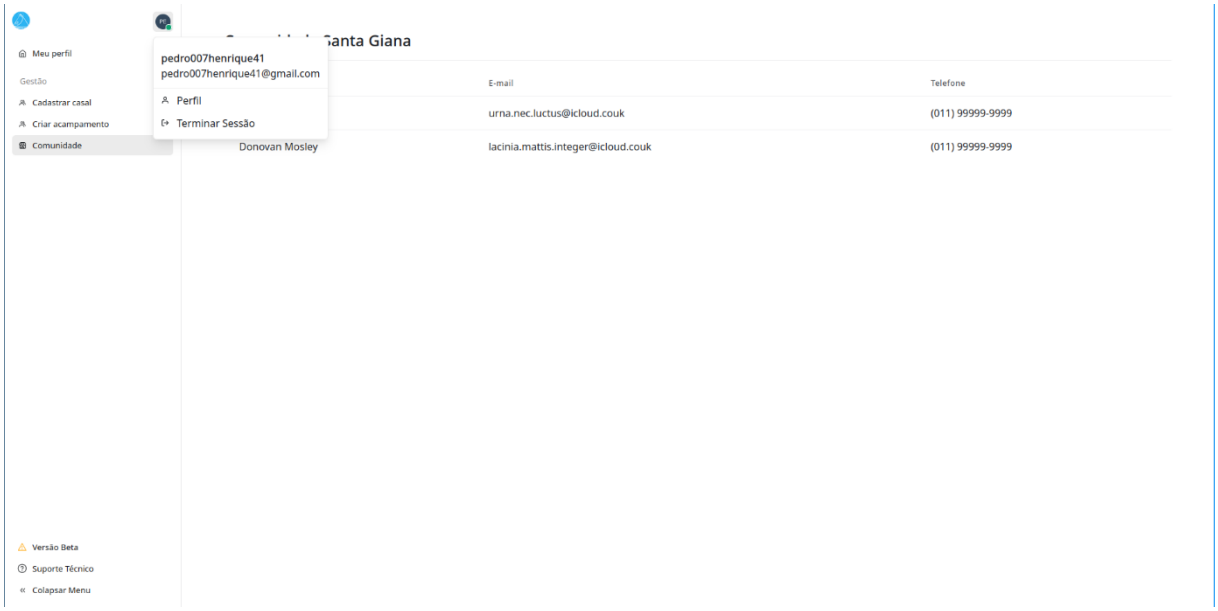


Fonte: Os autores, 2023

Logout

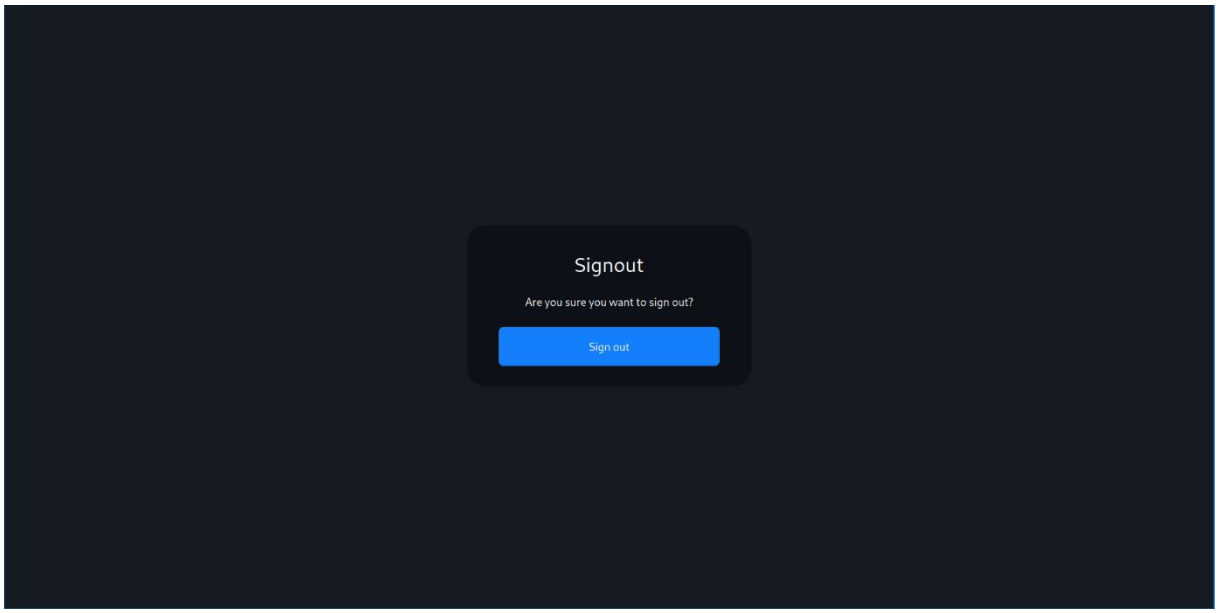
Ao clicar na foto de perfil na barra lateral, os usuários podem realizar logout de forma rápida e segura, encerrando sua sessão e protegendo suas informações pessoais.

Figura 48 - Modal para logout



Fonte: Os autores, 2023

Figura 49 - Tela para logout



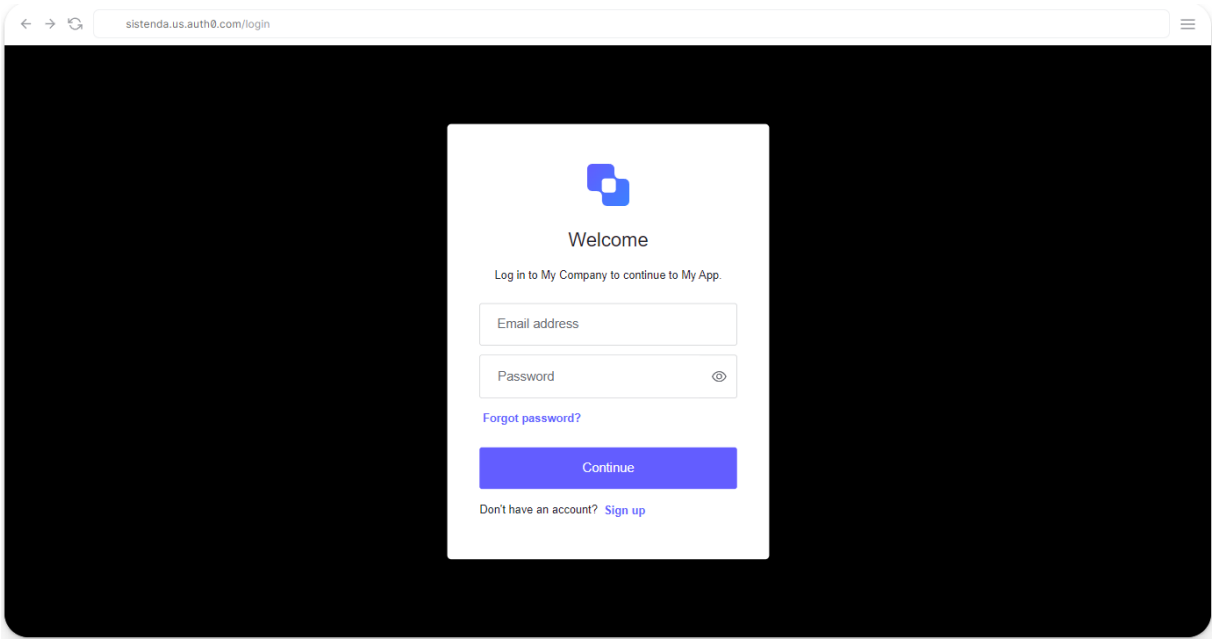
Fonte: Os autores, 2023

Tela de Login e Registro da Auth0

A tela de login e registro, alimentada pelo serviço Auth0, oferece uma entrada segura e eficiente aos usuários. A autenticação e autorização são fundamentais para a proteção de dados e informações do usuário. Além disso, também é permitido que

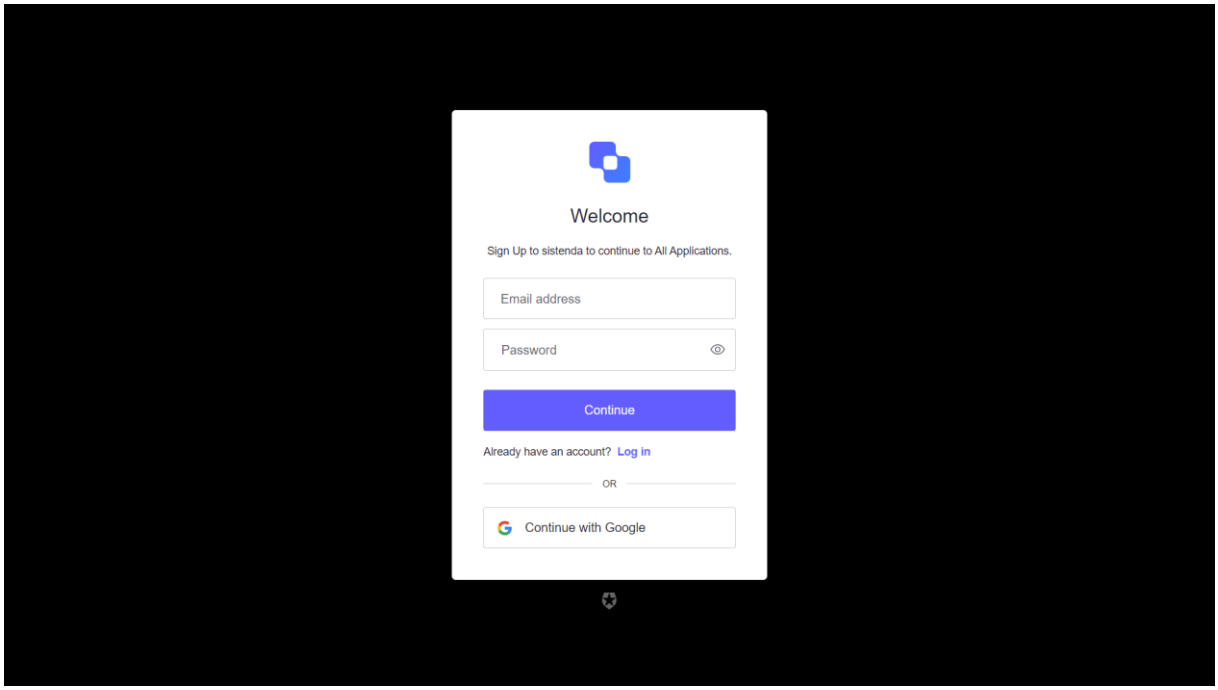
o usuário continue seu registro com uma conta Google.

Figura 50 - Tela de login do auth0



Fonte: Os autores, 2023

Figura 51 - Tela de registro do usuário



Fonte: Os autores, 2023

6 Resultados e Discussão

Vivemos em uma era em constante evolução, onde a tecnologia desempenha um papel crucial na vida de todos. As organizações e comunidades religiosas não estão isentas desse impacto.

No contexto atual, onde a Comunidade Tenda cresceu consideravelmente e abraça uma série de atividades em casais, a necessidade de uma solução tecnológica se tornou evidente. Este projeto desenvolveu um software que aborda de maneira eficaz o desafio da gestão de escalas da Comunidade Tenda, proporcionando uma solução transparente e organizada.

Nossa solução proposta foi o desenvolvimento de um software de gestão personalizado, projetado especificamente para atender às necessidades da Comunidade Tenda.

Esse sistema é uma ferramenta abrangente que permite a organização ter eficiência nas atividades, tarefas e escalas. Ele oferece funcionalidades para registrar e acessar informações dos casais, gerenciar eventos, retiros e encontros, além de proporcionar uma visão clara e organizada das atividades da comunidade.

Este software foi desenvolvido sob medida, considerando os requisitos específicos da Comunidade Tenda, proporcionando uma solução única e eficaz.

Considerações finais

Ao longo deste processo, buscamos atender aos objetivos iniciais estabelecidos para o desenvolvimento do sistema de gestão dos membros da Comunidade Tenda.

O principal desafio foi a complexidade das escalas e a organização das informações em um ambiente com mais de mil casais e uma variedade de eventos e retiros. Embora atualmente tenhamos implementado algumas telas, como "Meu Perfil", "Criar Acampamento", "Cadastrar Casal" e "Comunidade".

É importante ressaltar que construímos a base sólida do sistema toda a infraestrutura de desenvolvimento foi implementada, e estamos prontos para realizar o *deploy* e continuar o desenvolvimento das funcionalidades restantes. As tecnologias escolhidas, como Next.js e Nest.js, provaram ser sólidas e eficazes. Essas tecnologias são amplamente reconhecidas pela comunidade e oferecem uma série de recursos e

facilidades para a construção de aplicativos web modernos e escaláveis.

Projetos futuros incluem a finalização do desenvolvimento das funcionalidades pendentes, a adição de recursos de gerenciamento de escalas e a continuação da otimização do sistema. Com a estrutura estabelecida, estamos prontos para continuar aprimorando e personalizando o sistema para atender às necessidades específicas da Comunidade Tenda.

Nossa parceria resultará em um sistema de gestão eficiente, que permitirá o acompanhamento preciso das atividades, análises estratégicas e uma gestão organizada das operações da comunidade. Estamos comprometidos em continuar trabalhando juntos para alcançar os objetivos deste projeto e garantir o sucesso da Comunidade Tenda.

Referências

ABROSKIN, A., BARNOV, A. **Lefthook: knock your team's code back into shape**. Disponível em: <https://evilmartians.com/chronicles/lefthook-knock-your-teams-code-back-into-shape?utm_source=lefthook>. Acesso em: 10 out. 2023.

ASANA. **Como usar um estudo de viabilidade na gestão de projetos • Asana**. Disponível em: <https://asana.com/pt/resources/feasibility-study>. Acesso em: 10 out. 2023.

AUTH0. **Auth0 Overview**. Disponível em: <https://auth0.com/docs/get-started/auth0-overview>. Acesso em: 10 out. 2023.

Chakra UI - A simple, modular and accessible component library that gives you the building blocks you need to build your React applications. Disponível em: <https://chakra-ui.com>. Acesso em: 10 out. 2023.

DALLAVALLE, Silvia Inês; CAZARINI, Edson Walmir. **Regras do Negócio, um fator chave de sucesso no processo de desenvolvimento de Sistemas de Informação**. In: Anais do XX ENEGEP-Encontro Nacional de Engenharia de Produção. São Paulo, 2000.

Developing inside a Container using Visual Studio Code Remote Development. Disponível em: <https://code.visualstudio.com/docs/devcontainers/containers>. Acesso em: 10 out. 2023.

DEVMEDIA – Engenharia de Requisitos. **A importância da Engenharia de Requisitos - Engenharia de Software 33**. Disponível em: <https://www.devmedia.com.br/a-importancia-da-engenharia-de-requisitos-engenharia-de-software-33/19305>. Acesso em: 10 out. 2023.

DEVMEDIA – Engenharia de Requisitos não Funcionais. **Artigo Engenharia de Software 3 - Requisitos Não Funcionais**. Disponível em: <https://www.devmedia.com.br/artigo-engenharia-de-software-3-requisitos-nao-funcionais/9525>. Acesso em: 10 out. 2023.

Documentation. Disponível em: <https://www.passportjs.org/docs/>. Acesso em: 10 out. 2023.

Documentation | NestJS - A progressive Node.js framework. Disponível em: <https://docs.nestjs.com>. Acesso em: 10 out. 2023.

ESPINHA, R. G. **Matriz de Rastreabilidade de Requisitos: saiba como gerenciar as mudanças no escopo**. Disponível em: <https://artia.com/blog/matriz-de-rastreabilidade>. Acesso em: 4 nov. 2023.

FERNANDEZ, T. **What is monorepo? (and should you use it?),** 2021. Disponível em: <https://semaphoreci.com/blog/what-is-monorepo>. Acesso em: 10 out. 2023.

Get Docker. Disponível em: <https://docs.docker.com/get-docker/>. Acesso em: 10 out. 2023.

Introduction | NextAuth.js. Disponível em: <https://next-auth.js.org/getting-started/introduction>. Acesso em: 10 out. 2023.

JOEL. **MER e DER: Modelagem de Bancos de Dados**. Disponível em: <https://www.devmedia.com.br/mer-e-der-modelagem-de-bancos-de-dados/14332>. Acesso em: 10 out. 2023

LUCIDCHART – BPMN. **O que é BPMN?** Disponível em: <https://www.lucidchart.com/pages/pt/o-que-e-bpmn>. Acesso em: 10 out. 2023.

LUCIDCHART – Caso de uso UML. **Diagrama de caso de uso UML: O que é, como fazer e exemplos**. Disponível em: <https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml>. Acesso em: 10 out. 2023.

LUCIDCHART – Diagrama de Atividade. **Diagrama de Atividade.** Disponível em: <https://www.lucidchart.com/pages/pt/modelos/diagrama-de-atividade>. Acesso em: 10 out. 2023.

LUCIDCHART – Diagrama de Máquina de Estados UML. **O que é um diagrama de máquina de estados?.** Disponível em: <https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-maquina-de-estados-uml>. Acesso em: 10 out. 2023.

LUCIDCHART – Diagrama de Sequência UML. **O que é um diagrama de sequência UML?** Disponível em: <https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-sequencia-uml>. Acesso em: 10 out. 2023.

MINAS, S. **O QUE É BUSINESS MODEL CANVAS E COMO APLICÁ-LO NO SEU NEGÓCIO?** Disponível em: <https://inovacaosebraeminas.com.br/o-que-e-business-model-canvas-e-como-aplica-lo-no-seu-negocio/>. Acesso em: 10 out. 2023.

Nakagawa, M. **5W2H – PLANO DE AÇÃO PARA EMPREENDEDORES**. Disponível em: <https://sebrae.com.br/Sebrae/Portal%20Sebrae/Anexos/5W2H.pdf>. Acesso em: 10 out. 2023.

Next.js by Vercel - The React Framework. Disponível em: <https://nextjs.org>. Acesso em: 10 out. 2023.

O que é Business Model Canvas e como aplicá-lo no seu negócio? Disponível em: <https://inovacaosebraeminas.com.br/o-que-e-business-model-canvas-e-como-aplica-lo-no-seu-negocio>. Acesso em: 12 dez. 2023.

PM2 - Home. Disponível em: <https://pm2.keymetrics.io>. Acesso em: 4 nov. 2023.

PRADA, C. **TUDO sobre MATRIZ SWOT: o que é, passo a passo e dicas para aplicar.** Postado em 25/03/2020. Disponível em: <https://www.euax.com.br/2020/03/matriz-swot/>. Acesso em: 10 out. 2023.

PRECIOSO, V. **Como realizar o Levantamento de Requisitos no desenvolvimento de software.** Disponível em: <https://www.cedrotech.com/blog/levantamento-de-requisitos-e-desenvolvimento-de-softwares/>. Acesso em: 10 out. 2023.

PROJECT MANAGEMENT INSTITUTE. **Um Guia Do Conhecimento Em Gerenciamento De Projetos**, 4° Edição. Saraiva, 2012, p. 144.

RxJS. Disponível em: <https://rxjs.dev/guide/overview>. Acesso em: 4 nov. 2023.

Saas UI. 2023. Disponível em: <https://saas-ui.dev>. Acesso em: 10 out. 2023.

Turborepo Quickstart – Turborepo. Disponível em: <https://turbo.build/repo/docs>. Acesso em: 4 nov. 2023.

Tutorial on writing makefiles. Disponível em: <https://www.math.colostate.edu/~yzhou/computer/writemakefile.html>. Acesso em: 4 nov. 2023.

Apêndice 01 – Entrevista: Principais atividades realizadas pelo Tenda:

Avaliação do problema

1) Qual é o problema? (tente explicá-lo resumindo-o em 2 linhas)

2) Por que este problema existe/o que gerou ele?

3) Como é possível solucioná-lo agora/como ele é solucionado hoje?

4) Qual seria a melhor forma de solucioná-lo?

5) Oque tornaria essa solução a melhor para você?

6) Como você avaliaria uma solução eficaz?

7) Compreensão do ambiente do usuário

8) Quem serão os usuários?

9) Qual a formação educacional desses?

De 1 a 5 qual o nível de conhecimento de informática deles? (1- precisa de ajuda; 2 - tem dificuldade; 3- familiarizado com informática; 4- utiliza com facilidade; 5- possui conhecimentos de informática avançada)

1) Os usuários estão familiarizados com o tipo de aplicativo a ser implantado?

2) Que plataformas (hardware, SO, etc.) estão sendo usadas? Quais são seus planos para plataformas futuras?

3) Quais são os sistemas ou aplicativos adicionais que precisam ser integrados à solução? Quais os sistemas que irão necessitar das informações fornecidas pela solução?

4) Quem busca informações no sistema?

5) Quais as limitações para quem as busca? (se houverem níveis de acesso especifique-os aqui)

6) Quem provê informações para o sistema? (se houverem níveis de acesso especifique-os aqui)

7) Que tipos de documentação impressa e/ou online são necessárias?

8) Quais as expectativas em relação a facilidade de uso do produto?

9) Quais as expectativas em relação ao tempo de treinamento?

10) Os usuários finais já foram considerados/entrevistados?

11) Outros requerimentos

12) Se houver, quais são os requisitos legais, portarias, regulamentos, normas utilizadas pelo Cliente ou ainda requisitos ambientais que devem ser suportados?

13) Quais padrões devem ser suportados?

14) Algum outro requisito extra que deva ser mencionado?

CONFIABILIDADE, DO DESEMPENHO E DAS NECESSIDADES DE SUPORTE

1) Quais são as suas expectativas em relação à confiabilidade?
2) Quais são as suas expectativas em relação ao desempenho?
3) Você dará suporte ao produto ou isso será feito por outras pessoas?
4) Você tem necessidades especiais de suporte e manutenção?
5) Quais são os requisitos de segurança?
6) Quais são os requisitos especiais de licenciamento?
7) Como o software será distribuído?

Finalização

1) Qual é a configuração padrão de tela?
2) Haverá arquivos de ajuda online (dentro do sistema) e qual padrão desse help?
3) Para cada tela definir qual tipo de dado esperado pelos campos?
4) O que deve ser entregue ao final do projeto?
5) Existe outra questão ou fato que eu deveria atentar?

Anexo 01 – Missão, visão e valores da Comunidade Tenda

Missão: Ser Família em Comunhão com Deus e com o Próximo, que no sentido amplo do termo, significa dom do céu.

Visão: A comunidade Tenda tem como visão a perspectiva doutrinária da igreja católica apostólica romana.

Valores: de acordo com a igreja católica apostólica romana, objetivando uma comunhão entre os membros que vivem um matrimônio.