INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIAS E TECNOLOGIA DE RONDÔNIA – CAMPUS JI-PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

SISTEMA WEB PARA GERENCIAMENTO DE DÍZIMO E CONTRIBUIÇÕES NAS COMUNIDADES RELIGIOSAS

AMANDA TEODORO CUNHA

Ji-Paraná – RO
2026

AMANDA TEODORO CUNHA

SISTEMA WEB PARA GERENCIAMENTO DE DÍZIMO E CONTRIBUIÇÕES NAS COMUNIDADES RELIGIOSAS

Monografia apresentada ao Instituto Federal de Rondônia - Campus Ji-Paraná, como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Área de Concentração: Ciências Exatas e da Terra.

Orientador (a): Prof. Me. Reinaldo Lima Pereira

Ji-Paraná – RO

2026

FICHA CATALOGRÁFICA

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

Cunha, Amanda Teodoro.
Sistema web para gerenciamento de dízimo e contribuições nas comunidades religiosas / Amanda Teodoro Cunha. - Ji-Paraná, 2026.
99 f. : il.

Orientador(a): Prof. Me. Reinaldo Lima Pereira.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Ji-Paraná, 2026.

1. Aplicação Web. 2. Gestão Financeira. 3. Dízimos. 4. Comunidades Religiosas. I. Pereira, Reinaldo Lima (orient.). II. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. III. Título.

Bibliotecário(a) Responsável: Cleuza Diogo Antunes, CRB-11/864

BANCA EXAMINADORA

gov.br
Documento assinado digitalmente
JEFFERSON ANTONIO DOS SANTOS
Data: 07/08/2026 15:35:27-0300
Verifique em https://validar.iti.gov.br

Esp. Jefferson Antonio dos Santos

Avaliador nº 1

gov.br
Documento assinado digitalmente
KARAN LUCIANO SILVA DI PERNIS
Data: 07/08/2026 21:56:51-0300
Verifique em https://validar.iti.gov.br

Esp. Karan Luciano Silva di Pernis

Avaliador nº 2

SerproID Serpro
Reinaldo Lima Pereira
CPF: ***.509.772-**
07/08/2026
ICP Brasil

Me. Reinaldo Lima Pereira

Presidente da Banca Examinadora

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema web para gerenciamento de dízimos, contribuições, fiéis e despesas em comunidades religiosas. A proposta surge a partir da realidade observada em uma comunidade religiosa, na qual o controle financeiro ainda é realizado de forma manual, por meio de registros físicos, o que pode ocasionar dificuldade na organização dos dados, geração de relatórios confiáveis e a transparência na prestação de contas. A pesquisa adotou uma abordagem qualitativa, com levantamento bibliográfico sobre gestão financeira em instituições religiosas, informatização de processos administrativos e tecnologias de desenvolvimento web. Para o desenvolvimento do software, foram realizadas etapas de levantamento de requisitos, prototipação de interfaces, modelagem do sistema, modelagem do banco de dados e planejamento das funcionalidades com apoio da metodologia ágil Scrum. O sistema proposto contempla módulos para cadastro e gerenciamento de contribuintes, registro de dízimos e contribuições, controle de despesas e geração de relatórios financeiros. A aplicação utiliza Vue.js para o front-end, Nest.js e TypeScript para a construção da API, além do MySQL para armazenamento dos dados. Como resultado, espera-se que a solução facilite o acompanhamento das contribuições e despesas, e auxilie os responsáveis pela administração financeira na consulta e geração de relatórios.

Palavras-chave: Aplicação Web. Gestão Financeira. Dízimos. Comunidades Religiosas.

ABSTRACT

This work presents the development of a web system for managing tithes, contributions, members, and expenses in religious communities. The proposal arises from the reality observed in a religious community where financial control is still carried out manually through physical records, which may cause difficulties in organizing data, generating reliable reports, and ensuring transparency in accountability. The research adopted a qualitative approach, with a bibliographic review on financial management in religious institutions, informatization of administrative processes, and web development technologies. For the software development, the stages included requirements gathering, interface prototyping, system modeling, database modeling, and functionality planning supported by the agile Scrum methodology. The proposed system includes modules for registering and managing contributors, recording tithes and contributions, controlling expenses, and generating financial reports. The application uses Vue.js for the front-end, Nest.js and TypeScript for building the API, and MySQL for data storage. As a result, the solution is expected to facilitate the monitoring of contributions and expenses and support those responsible for financial administration in consulting information and generating reports.

Keywords: Web Application. Financial Management. Tithes. Religious Communities.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API - Application Programming Interface

CSS - Cascading Style Sheets

HTML – HyperText Markup Language

HTTP - Hypertext Transfer Protocol

IBGE - Instituto Brasileiro de Geografia e Estatística

LGPD - Lei Geral de Proteção de Dados Pessoais

SGBD - Sistema Gerenciador de Banco de Dados

SQL - Structured Query Language

UML – Unified Modeling Language

SUMÁRIO

1. INTRODUÇÃO

A gestão financeira em comunidades religiosas é essencial para a organização administrativa e para a manutenção das ações desenvolvidas por essas instituições. O controle de dízimos, contribuições, fiéis e despesas está diretamente relacionado ao acompanhamento dos recursos arrecadados, à realização de atividades religiosas e sociais e à prestação de contas. No entanto, em algumas comunidades, esses registros ainda são realizados de forma manual, por meio de fichas ou outros registros físicos, o que pode dificultar a consulta das informações, o acompanhamento das movimentações financeiras e a geração de relatórios.

Nesse cenário, a tecnologia surge como uma alternativa importante para apoiar processos administrativos e financeiros, oferecendo recursos capazes de tornar o registro, a organização e a consulta de informações mais eficientes. Os sistemas web, por serem acessíveis por meio de navegadores e possibilitarem o gerenciamento de dados em ambiente digital, podem contribuir para a melhoria do controle financeiro nas comunidades religiosas.

Dessa forma, o trabalho tem como objetivo principal desenvolver uma aplicação que auxilia no controle das informações, facilita o acompanhamento das movimentações financeiras e apoia a tomada de decisões pelos responsáveis pela administração da comunidade.

1.1. Problemática

De que forma a tecnologia pode se apresentar como uma opção viável para contribuir com a otimização do gerenciamento de dízimos e contribuições em comunidades religiosas, tornando os processos financeiros e o controle de informações dos contribuintes mais organizados, seguros e eficientes?

1.2. Justificativa

O controle de informações nas comunidades religiosas é essencial para garantir maior organização administrativa, transparência na gestão de dízimos e contribuições e apoio à prestação de contas. Nesse contexto, o desenvolvimento de um sistema web voltado ao gerenciamento dessas informações torna-se relevante, pois pode tornar os registros mais organizados, facilitar o acesso aos dados e diminuir dificuldades comuns em processos realizados manualmente.

1.3. Objetivos

1.3.1. Objetivo Geral

O objetivo geral desse projeto é desenvolver uma aplicação Web voltada para comunidades religiosas, com a finalidade de gerenciar os contribuintes e controlar os dízimos e contribuições, reduzindo falhas de registro, melhorando a organização das informações e facilitando a geração de relatórios.

1.3.2. Objetivos Específicos

I. Desenvolver módulos de cadastros e consultas, permitindo o controle organizado das informações de todos os contribuintes cadastrados no sistema.
II. Implementar a funcionalidade de emissão de relatórios personalizados, com a finalidade de facilitar na coleta de dados quando necessário.
III. Desenvolver módulos de controle de contribuições e gastos, para gestão financeira.

1.4. Estrutura do Trabalho

Este trabalho está organizado em cinco capítulos. O primeiro capítulo apresenta a introdução, abordando o contexto do tema, a problemática, a justificativa e os objetivos da pesquisa. O segundo capítulo contém o referencial teórico, reunindo conceitos relacionados às comunidades religiosas, à gestão financeira, à informatização de processos administrativos e às ferramentas utilizadas no desenvolvimento do sistema. O terceiro capítulo descreve as metodologias adotadas para o desenvolvimento do trabalho. O quarto capítulo apresenta os resultados e discussões, destacando as funcionalidades desenvolvidas e as telas do sistema. O quinto capítulo apresenta as conclusões do trabalho, retomando os objetivos propostos e indicando possíveis melhorias futuras para o sistema. Por fim, no Apêndice A, é apresentado o projeto de software, contendo a documentação elaborada para apoiar o desenvolvimento da aplicação.

2. REFERENCIAL TEÓRICO

2.1. Comunidades Religiosas no Brasil

O Brasil é um país reconhecido por sua forte tradição religiosa, tendo em vista o elevado número de pessoas que se declaram pertencentes a alguma crença. De acordo com dados do Censo Demográfico de 2022, realizado pelo Instituto Brasileiro de Geografia e Estatística (IBGE), o catolicismo permanece como a religião com maior número de adeptos no país, representando mais da metade da população em todas as regiões, embora tenha apresentado uma redução significativa nas últimas décadas. Em contrapartida, observa-se o crescimento expressivo das religiões evangélicas e também do grupo de pessoas que se declaram sem religião.

Segundo Carvalho (2017), além do aspecto espiritual, as instituições religiosas exercem um importante papel social, promovendo ações comunitárias, atividades assistenciais e apoio a famílias em situações de vulnerabilidade. Dessa forma, as igrejas prestam um papel de convivência e apoio, desempenhando funções que vão além da dimensão exclusivamente religiosa.

Diante dessa situação, considerando a quantidade de fiéis que participam das comunidades religiosas, observa-se uma necessidade de organização administrativa eficiente, tanto para o controle dos membros e participantes das atividades quanto para a gestão dos recursos financeiros provenientes de dízimos e contribuições voluntárias, recursos que são destinados à manutenção das instituições e ao desenvolvimento de ações sociais. Nesse contexto, torna-se fundamental a adoção de mecanismos que facilitem o registro, o controle e a transparência dessas informações, garantindo maior eficiência administrativa e confiabilidade na prestação de contas à comunidade.

2.2. Gestão financeira nas Igrejas

As Igrejas possuem organização interna, necessidade de controle financeiro, registro de dízimos e ofertas, prestação de contas à comunidade. Como essas instituições gerenciam seus recursos? Quais desafios existem no controle manual?

De acordo com Santos (2018), “ao lado do elemento fé, nasce a doutrina; ao lado da comunhão, a instituição dos sacramentos; ao lado do serviço, os cargos e funções”. Dessa forma, as igrejas, assim como qualquer outra instituição, apresentam a necessidade de planejamento e controle administrativo. Para cada área específica como administração, tesouraria e coordenação de atividades sociais, é fundamental que haja uma liderança

responsável por suas respectivas funções, a fim de garantir uma gestão estruturada que assegure o adequado funcionamento das atividades desenvolvidas.

Como consequência de sua atividade principal, as igrejas desenvolvem diversas ações voltadas à comunhão entre os membros, ao ensino bíblico e à participação ativa na sociedade por meio de serviços comunitários. Para que essa estrutura funcione de maneira adequada, torna-se indispensável a existência de recursos financeiros. Diferentemente das organizações empresariais, cuja arrecadação está vinculada à oferta de produtos ou serviços, as igrejas dependem fundamentalmente da conscientização dos fiéis quanto à importância das contribuições para a continuidade de suas atividades. Nesse contexto, a entrega de dízimos e ofertas é compreendida como prática voluntária fundamentada em princípios bíblicos, destinada à manutenção da instituição e ao desenvolvimento de ações religiosas e sociais, sendo igualmente necessária a demonstração de que tais recursos estão sendo aplicados de forma responsável e transparente (GABY; HIRUMA, 2021).

Entretanto, como acontece em muitas comunidades, o controle dessas informações ocorre de forma manual, o que dificulta o fator organização e transparência. Sendo assim, torna-se evidente a necessidade de modernização de processos administrativos por meio de adoção tecnológicas, que possibilitem maior precisão e fortalecimento da transparência na gestão dos recursos financeiros.

2.3. Informatização das Igrejas

A evolução da tecnologia da informação tem promovido mudanças significativas na forma como as organizações estruturam seus processos administrativos. No contexto religioso, a informatização surge como alternativa para otimizar o armazenamento, a organização e a consulta de informações, especialmente em secretarias paroquiais. De acordo com Mello e Oliveira (2011), a informatização dos serviços paroquiais possibilita maior agilidade, redução de custos e melhor gerenciamento dos dados, substituindo arquivos físicos e controles manuais por sistemas digitais integrados.

Os autores destacam que, em muitas paróquias, os processos administrativos ainda são realizados de maneira manuscrita, incluindo registros financeiros, controle de dízimos, emissão de documentos e organização de cadastros. Essa prática, além de demandar maior tempo operacional, aumenta o risco de inconsistências e perda de informações. A pesquisa realizada pelos autores com paróquias do Distrito Federal evidenciou que grande parte das instituições ainda utiliza planilhas básicas ou registros físicos para controle financeiro e

administrativo, demonstrando a necessidade de modernização desses processos (MELLO; OLIVEIRA, 2011).

Nesse sentido, a informatização não se limita apenas à digitalização de documentos, mas envolve a implementação de sistemas de informação capazes de integrar diferentes setores da instituição, promover padronização de procedimentos e facilitar a comunicação interna e externa. Conforme ressalta Mello e Oliveira (2011), a adoção de um sistema único pode contribuir significativamente para a melhoria da gestão, proporcionando maior controle, transparência e eficiência administrativa.

No entanto, a informatização também exige atenção ao tratamento adequado dos dados pessoais envolvidos. A Lei Geral de Proteção de Dados Pessoais (LGPD - Lei n° 13.709/2018), estabelece normas para o tratamento de dados pessoais, inclusive em meios digitais, com o objetivo de proteger direitos fundamentais como a liberdade e a privacidade (BRASIL, 2018). Em sistemas voltados a comunidades religiosas, essa atenção torna-se ainda mais relevante, pois informações como nome, telefone, endereço, data de nascimento e registros financeiros dos fiéis precisam ser armazenadas e utilizadas de forma responsável. Além disso, por estarem vinculadas a uma comunidade religiosa, tais informações revelam aspectos relacionados à convicção religiosa ou à participação em organização religiosa, elementos tratados pela lei como dados pessoais sensíveis. Dessa forma, o tratamento dessas informações deve observar as hipóteses legais previstas na LGPD, como o consentimento do titular ou outras bases aplicáveis, além dos princípios de finalidade, necessidade, segurança e transparência (BRASIL, 2018).

2.4. Ferramentas de desenvolvimento

2.4.1. Astah UML

A UML (Linguagem de Modelagem Unificada) é uma linguagem visual composta por um conjunto de diagramas utilizados na modelagem de sistemas com o objetivo de visualizar, especificar, construir e documentar esses softwares. A existência de um modelo visual de um sistema facilita a comunicação entre membros de equipes de desenvolvimento, além de auxiliar na identificação e definição de possíveis alterações nas funcionalidades ainda nas fases iniciais do projeto, antes mesmo da entrega do sistema já finalizado (GOULART, 2013).

Essa linguagem possui um conjunto de treze diagramas que permitem representar diferentes perspectivas de um sistema, sendo organizados em dois grupos principais: Diagramas Estruturais e Diagramas Comportamentais, sendo que os comportamentais

possuem uma subdivisão chamada de Diagramas de Interação (DA SILVA; DINIZ; MARTINS, 2017). Dentre os mais utilizados destacam-se: diagramas de casos de uso, o diagrama de classes e o diagrama de sequência (FERREIRA, 2010). Sendo o Diagrama de Caso de Uso utilizado para especificar um conjunto de funcionalidades do sistema, deixando bem claro quais são e quem irá fazer o uso delas. Já o Diagrama de Classes, como o nome já diz, define as classes do sistema, bem como seus atributos, métodos, interfaces, heranças e também seus relacionamentos. Por fim, o Diagrama de Sequência demonstra, de forma visual, a interação entre diferentes objetos ao longo do tempo, evidenciando a troca de mensagens que ocorre durante a execução das funcionalidades do sistema (DA SILVA; DINIZ; MARTINS, 2017). Esses diagramas facilitam tanto o entendimento da lógica do sistema quanto a sua documentação técnica, servindo como guias para as etapas posteriores de desenvolvimento.

Dentro desse contexto o Astah se destaca como uma ferramenta de modelagem de diagramas UML, sendo amplamente utilizada no mercado, destacando-se por sua popularidade e contínua evolução. A disponibilidade de uma versão gratuita, denominada Astah UML, atrai a atenção de muitos usuários, o que por consequência desperta um grande número de sugestões de melhorias na ferramenta, fator considerado de grande importância para melhorias constantes. Além disso, o Astah conta com sua versão Profissional, na qual se transforma em uma ferramenta completa, disponibilizando todos os tipos de diagramas UML (NETO, 2017). Para a realização deste projeto, será utilizado o Astah UML com o objetivo de modelar o diagrama de casos de uso, possibilitando a representação clara das funcionalidades do sistema e da interação entre os usuários e a aplicação proposta.

2.4.2. Figma

O Figma é uma ferramenta gratuita de design usada principalmente para a criação de interfaces e protótipos interativos de um certo produto a ser feito, com foco em usabilidade e experiência do usuário. Por ser uma ferramenta que permite a colaboração em tempo real e oferece recursos como testes, sistemas de design e uma biblioteca de componentes reutilizáveis, o Figma é amplamente utilizado tanto por profissionais e equipes de design que buscam otimizar seu fluxo de trabalho quanto por estudantes da área que procuram ferramentas mais intuitivas (FIGMA, 2025).

Por ser uma ferramenta de criação de interfaces gráficas e protótipos sem a necessidade de codificação, o Figma permite que seja realizado a validação prévia de fluxos de navegação e a identificação de problemas de usabilidade dentro das aplicações, ainda nas

etapas iniciais do desenvolvimento. Dessa forma, ele auxilia na construção de soluções mais alinhadas a necessidade dos usuários.

2.4.3. Javascript

Segundo Copes (2023), o JavaScript é considerado uma das linguagens de programação mais populares do mundo, usado principalmente para sites e aplicações web. Criado em 1995 por Brendan Eich, a linguagem surgiu inicialmente para atender à necessidade de validação de formulários HTML nos navegadores desenvolvidos pela empresa Netscape. Ao longo dos anos, o JavaScript evoluiu significativamente, ampliando suas funcionalidades e consolidando-se como uma linguagem executada no lado do cliente (client-side). Dessa forma, o JavaScript é responsável por controlar elementos visuais e interações das páginas web, permitindo que ações realizadas pelo usuário sejam processadas diretamente no navegador, contribuindo para interfaces mais interativas e responsivas (REBELLO, 2022).

As principais vantagens da utilização do JavaScript estão relacionadas à sua simplicidade, uma vez que possui uma estrutura acessível e de fácil compreensão e implementação. Além disso, destaca-se pela velocidade, pois executa scripts diretamente no navegador do usuário, sem a necessidade de conexão prévia com o servidor ou uso de compiladores. A versatilidade também é um ponto relevante, já que o JavaScript é compatível e pode ser integrado com outras linguagens, como PHP e Java (REBELLO, 2022). Destaca-se também a sua ampla popularidade, que resulta em uma vasta quantidade de bibliotecas, frameworks, fóruns e materiais de apoio, atendendo tanto iniciantes quanto desenvolvedores mais experientes. Por fim, a linguagem contribui para a redução da carga do servidor, visto que validações e algumas solicitações podem ser realizadas diretamente no lado do cliente (REBELLO, 2022).

2.4.3.1. Chart.js

O Chart.js é uma biblioteca JavaScript de código aberto utilizada para a criação de gráficos em aplicações web. A ferramenta permite representar dados de forma visual por meio de diferentes tipos de gráficos, como barras, linhas, pizza e área. Além disso, possui configuração simples e opções de personalização, facilitando sua utilização em sistemas que precisam apresentar informações de maneira mais clara e dinâmica (CHART.JS, 2025).

De acordo com a documentação oficial, o Chart.js é compatível com os principais frameworks JavaScript, incluindo Vue.js, React e Angular, além de possuir suporte ao

TypeScript. A biblioteca utiliza o elemento Canvas, do HTML5, para renderização dos gráficos, o que contribui para um bom desempenho na apresentação dos dados. Também oferece recursos como animações, legendas, escalas e plugins, permitindo adaptar os gráficos às necessidades da aplicação (CHART.JS, 2025).

Neste trabalho, o Chart.js será utilizado no dashboard do sistema para apresentar, por meio de gráficos, informações relacionadas a dízimos, contribuições e despesas. Essa visualização permitirá acompanhar os dados financeiros registrados na aplicação de maneira mais clara e objetiva.

2.4.4. TypeScript

O TypeScript é uma linguagem de programação desenvolvida pela Microsoft que funciona como um superset do JavaScript, ou seja, uma extensão da linguagem que adiciona novos recursos sem deixar de ser compatível com o JavaScript tradicional. Entre as principais funcionalidades adicionadas estão a tipagem de dados e melhores mecanismos para utilização da programação orientada a objetos, o que contribui para a construção de aplicações com uma arquitetura mais organizada e estruturada (DEVMEDIA, 2016).

> Diferentemente do que ocorria alguns anos atrás, onde JavaScript era utilizada apenas no lado cliente das aplicações, atuando na validação de formulários e composição de elementos da interface, atualmente temos iniciativas que levam essa linguagem a representar a parte principal de aplicações mobile (em frameworks de desenvolvimento híbrido como Apache Cordova) e web (no lado servidor com Node.js). Nestes cenários, os projetos não se resumem mais a pequenos arquivos com algumas linhas de código, muitas vezes apenas utilizando outros frameworks front-end. Agora, passa a ser necessária a possibilidade de construir arquiteturas mais sólidas, onde se possa organizar adequadamente o código e aplicar as melhores práticas e técnicas de programação, como a Orientação a Objetos e, dentro desta, o uso de interfaces. (DEVMEDIA, 2016)

Uma característica importante do TypeScript é que todo código escrito na linguagem é posteriormente compilado para JavaScript, garantindo compatibilidade com navegadores e ambientes de execução já existentes. Além disso, por ser um superconjunto do JavaScript, qualquer código nesta linguagem pode ser utilizado diretamente em arquivos TypeScript, permitindo que projetos existentes sejam adaptados gradualmente, sem a necessidade de reescrever todo o código (DEVMEDIA, 2016).

2.4.5. Framework

Os Frameworks são estruturas pré-definidas de código que servem como base para o desenvolvimento de aplicações. Sua principal função é fornecer um conjunto de ferramentas e bibliotecas que ajudam desenvolvedores a criar sistemas de maneira mais rápida, organizada e eficiente (MIRANDA, 2025).

Na prática, os frameworks funcionam como esqueletos da aplicação. Um framework já vem, em sua configuração padrão, com partes essenciais da estrutura que um software necessita para funcionar corretamente, como funcionamento de rotas, conexão com banco de dados, autenticação de usuários e manipulação de requisitos (MIRANDA, 2025). Ao invés de começar do zero, tudo que o programador precisa fazer é preencher o código com a ideia principal do projeto, ou seja, as suas funcionalidades específicas.

Uma característica importante de um framework é que ele define como o código da aplicação deve ser organizado e quando cada trecho será executado (TICOOP BRASIL, [s.d.]). Em outras palavras, é o próprio framework que define o fluxo principal da aplicação. Essa funcionalidade ajuda a padronizar o desenvolvimento e facilitar manutenções futuras, o que geralmente ajuda a melhorar a produtividade.

Os frameworks podem ser classificados de acordo com a sua finalidade, linguagem de programação e área de aplicação (LENCINA, 2023), sendo utilizados no desenvolvimento de diferentes tipos de sistemas, como aplicações móveis, aplicações web, jogos e sistemas empresariais. Tendo em vista o desenvolvimento web, os frameworks são divididos em duas partes, front-end e back-end, onde no front-end, os frameworks atuam no lado do cliente e são responsáveis pela construção da interface e da interatividade das aplicações, utilizando tecnologias como HTML, CSS e JavaScript, tendo como exemplo o Angular e o Vue.js. Já no back-end, os frameworks utilizados são executados no lado do servidor e auxiliam no processamento de dados, controle de requisições HTTP e integração com banco de dados, como ocorre com o Django, desenvolvido em Python, e o Spring, baseado na linguagem Java (LENCINA, 2023). Dessa forma, a utilização dos frameworks traz a possibilidade de maior padronização e organização nos códigos, logo auxilia no aumento da produtividade na construção de sistemas.

2.4.6. Vue.js

O Vue.js é um framework JavaScript voltado para o desenvolvimento de interfaces, sendo amplamente utilizado na construção de aplicações web modernas. O framework possui uma abordagem progressiva, o que permite que seja incorporado gradualmente a projetos que

já existem ou utilizados do zero de forma completa na criação de aplicações (CALDAS, 2024). O Vue.js se baseia no uso de tecnologias padrão da web, como HTML, CSS e JavaScript, oferecendo um modelo de programação com componentes, o que contribui para a organização e reutilização de código.

De acordo com Caldas (2024), uma de suas principais características é a sua reatividade, na qual permite a atualização automática da interface sempre que ocorra alguma alteração no código. Esse mecanismo facilita o gerenciamento do estado da aplicação, tornando o desenvolvimento mais eficiente e menos propensos a erros.

O Vue.js com todas as suas características pode também ser utilizado para o desenvolvimento de Single Page Applications, ou aplicações de página única, nas quais o conteúdo é atualizado sem a necessidade de recarregar toda a página. Nesse modelo de desenvolvimento entra o uso de seus componentes, onde são criados conjuntos de estruturas (HTML), estilo (CSS) e comportamento (JavaScript) em um único elemento. Além disso, os componentes podem interagir entre si por meio de propriedades e eventos tornando o desenvolvimento mais dinâmico. Dessa forma, é possível dividir a aplicação em partes menores e independentes, o que, como mencionado anteriormente, facilita a organização do código e a manutenção do sistema (GEEKHUNTER, 2020).

2.4.6.1. Lucide

O Lucide é uma biblioteca de ícones de código aberto que disponibiliza arquivos no formato SVG para uso em sistemas digitais. A biblioteca possui um conjunto amplo de ícones e busca facilitar a utilização desses elementos por designers e desenvolvedores, mantendo padrões visuais.(LUCIDE, s.d).

Além da variedade de ícones, o Lucide oferece pacotes oficiais para diferentes tecnologias, como React, Vue, Angular, Svelte, React Native, entre outras. No caso do Vue.js, a biblioteca disponibiliza ícones como componentes independentes, permitindo sua importação e utilização diretamente nas telas da aplicação. Esses componentes podem ser personalizados por meio de propriedades como tamanho, cor e espessura, facilitando sua adaptação ao estilo visual do sistema (LUCIDE, s.d).

Neste trabalho, o Lucide será utilizado na interface do sistema web para representar visualmente ações, menus e funcionalidades da aplicação. Dessa forma, os ícones contribuem para tornar a navegação mais intuitiva e auxiliar os usuários na identificação dos recursos disponíveis no sistema.

2.4.7. Nest.js

O Nest.js é um framework utilizado para o desenvolvimento de aplicações do lado do servidor em Node.js, projetado para permitir a criação de sistemas eficientes e escaláveis. Ele utiliza JavaScript progressivo e oferece suporte completo à linguagem TypeScript, embora também permita o desenvolvimento utilizando apenas o JavaScript. O framework combina diferentes elementos de programação, como programação orientada a objetos, programação funcional e programação reativa funcional, com o objetivo de fornecer uma estrutura moderna e organizada para o desenvolvimento de aplicações back-end (NESTJS, 2017).

> O Nest.js fornece uma arquitetura de aplicativos pronta para uso que permite que desenvolvedores e equipes criem aplicativos altamente testáveis, escalonáveis e de fácil manutenção. A arquitetura é fortemente inspirada em Angular. (NESTJS, 2017)

Entre as principais vantagens do Nest.js está sua arquitetura limpa e modular, que organiza a aplicação em diferentes partes com responsabilidades bem definidas, facilitando a manutenção e organização do código. Além disso, o Nest.js utiliza o padrão de injeção de dependência, o que facilita o desenvolvimento e os testes da aplicação. Outro ponto importante é sua escalabilidade, permitindo a criação de aplicações robustas, como APIs, microsserviços e sistemas em tempo real. O framework também é compatível com outras tecnologias do ecossistema Node.js, como o Express e bibliotecas de banco de dados, ampliando suas possibilidades de uso no desenvolvimento de aplicações back-end (SOARES, 2023).

2.4.8. Sistema de Gerenciamento de Banco de Dados

Segundo Caldeira (2023), um Banco de Dados é um dispositivo de armazenamento para conjunto de informações que precisam ser armazenadas e controladas por alguém ou alguma instituição. Para esses dados serem gerenciados é necessário o uso de Sistemas de Gerenciamento de Banco de Dados, ou SGBD, os quais possibilitam a criação e o gerenciamento de usuários, bem como a consulta, alteração e exclusão de registros, entre outras funcionalidades (CALDEIRA, 2023).

> Os SGBDs facilitam a organização de dados, permitindo que informações sejam armazenadas de forma estruturada e acessadas de maneira eficiente. Eles são essenciais para garantir a integridade e a segurança dos dados, além de permitir a recuperação rápida de informações relevantes (OLIVEIRA, 2024).

Os SGBDs podem ser classificados de acordo com a forma como os dados são armazenados nos bancos de dados. Entre os modelos mais utilizados, destacam-se os bancos de dados relacionais e os não relacionais (NoSQL). Segundo Andrade (2021), os SGBDs

relacionais são aqueles que organizam os dados em tabelas que podem se relacionar entre si, sendo compostas por atributos com diferentes tipos de dados. Andrade (2021) ainda aponta que os modelos Não-Relacionais, ou NoSql, caracterizam-se pelo alto desempenho e, geralmente, não utilizam a linguagem SQL como principal meio de consulta aos dados.

A utilização de Sistemas de Gerenciamento de Banco de Dados é fundamental no desenvolvimento e na manutenção de sistemas, especialmente dentro de contextos onde há um grande volume de dados sendo gerados constantemente (OLIVEIRA, 2024). Sendo assim, os SGBDs possibilitam a organização estruturada das informações, reduzindo problemas como redundância e inconsistência dos dados, além de oferecerem acesso rápido às informações armazenadas (ANDRADE, 2021). Dessa forma os SGBDs contribuem para a redução do esforço humano, otimizando processos e rotinas de desenvolvimento, tornando-se indispensáveis para aplicações modernas.

2.4.9. MySQL Workbench

De acordo com a Oracle (2020), um banco de dados consiste em uma coleção organizada de informações estruturadas, armazenadas eletronicamente em um sistema computacional e controladas por um Sistema de Gerenciamento de Banco de Dados (SGBD). Nos modelos mais comuns atualmente, os dados são organizados em tabelas compostas por linhas e colunas, o que torna o processamento e a consulta das informações mais eficientes. Nesse contexto, a linguagem SQL (Structured Query Language) é amplamente utilizada para consultar, manipular e definir dados, além de possibilitar o controle de acesso às informações. Assim, os bancos de dados relacionais, como o MySQL, utilizam o SQL como principal meio de interação com os dados, sendo apoiados por ferramentas que facilitam sua modelagem, gerenciamento e administração (ORACLE, 2020). Para o desenvolvimento do banco de dados da aplicação proposta, foi adotado o MySQL Workbench como ferramenta de modelagem e gerenciamento, tendo em vista as suas funcionalidades integradas e interface gráfica intuitiva.

O MySQL Workbench é uma ferramenta visual desenvolvida pela Oracle que abrange modelagem de dados, desenvolvimento SQL e administração de banco de dados (DNC, 2025). Entre suas principais funcionalidades, destaca-se a possibilidade de criação e visualização de modelos de banco de dados por meio de interface gráfica, facilitando o planejamento da estrutura das tabelas e seus relacionamentos antes da implementação no sistema. Além disso, a ferramenta proporciona um ambiente amigável para a execução de comandos SQL, permitindo a criação, consulta, alteração e exclusão de dados de forma mais intuitiva (DNC, 2025).

O MySQL Workbench também reúne recursos essenciais para o gerenciamento do servidor MySQL, incluindo controle de conexões, gerenciamento de usuários e permissões, bem como suporte à manutenção do banco de dados (DNC, 2025). Dessa forma, a ferramenta contribui para maior organização, segurança e eficiência no desenvolvimento e administração de bancos de dados relacionais, sendo adequada para aplicações acadêmicas e profissionais.

2.4.10. Scrum

A metodologia Scrum é um framework de gerenciamento de projetos baseado em princípios ágeis, desenvolvido com o objetivo de organizar o trabalho das equipes e permitir a entrega contínua de valor ao cliente. Essa abordagem surgiu no contexto das mudanças constantes nos processos de desenvolvimento de software e em outros tipos de projetos, que passaram a exigir maior flexibilidade, adaptação e rapidez nas entregas. Segundo a CNN (2023), o Scrum foi criado em 1993 e posteriormente teve seus princípios consolidados, tornando-se uma das metodologias ágeis mais utilizadas para organizar e conduzir projetos de forma colaborativa e eficiente.

O funcionamento do Scrum baseia-se na divisão do projeto em ciclos curtos chamados de sprints, que normalmente possuem duração entre duas e quatro semanas. Durante esses ciclos, a equipe trabalha em um conjunto de tarefas previamente definidas, buscando entregar incrementos funcionais do produto ao final de cada etapa (CNN, 2023). O processo envolve quatro eventos formais, concedidos dentro de cada sprint, são eles o planejamento da sprint, reuniões diárias, revisão da sprint e retrospectiva. Cada um desses eventos propõe acompanhar o progresso das atividades, identificar dificuldades e promover melhorias contínuas no desenvolvimento do projeto (SCHWABER; SUTHERLAND, 2020).

Além disso, o Scrum é estruturado com base em papéis e artefatos que auxiliam na organização das atividades e na gestão do trabalho. Segundo Schwaber e Sutherland (2020), dentro de uma Scrum Team não existem sub-times ou hierarquias, somente uma unidade coesa de profissionais com seus papeis divididos, focados em um objetivo de cada vez. São definidos três responsabilidades específicas dentro das equipes: os Developers, responsáveis pelo desenvolvimento das funcionalidades e pela adaptação do plano de trabalho ao longo do ciclo, o Product Owner, que define e prioriza as funcionalidades do produto de acordo com as necessidades dos usuários e das partes interessadas, e o Scrum Master, que é responsável por auxiliar a equipe na aplicação correta das práticas do Scrum e remover possíveis impedimentos ao progresso do trabalho (SCHWABER; SUTHERLAND, 2020). A metodologia também utiliza artefatos como o Product Backlog, que reúne as funcionalidades

do produto, o Sprint Backlog, que define as tarefas de cada sprint, e o incremento, que representa a entrega funcional resultante de cada ciclo de desenvolvimento (CNN, 2023).

2.4.11. API RESTful

Uma API, ou Interface de Programação de Aplicações, é um recurso que permite a comunicação entre sistemas diferentes. Por meio delas, uma aplicação consegue solicitar, enviar e receber informações de outra aplicação de forma padronizada. Em sistemas Web, as APIs são muito utilizadas para conectar o front-end, que é a parte visual acessada pelo usuário, com o back-end, onde ficam as regras de negócio, o processamento dos dados e a comunicação com o banco de dados.

Um dos tipos mais utilizados é a API RESTful, que segue os princípios da arquitetura REST, sigla para Representational State Transfer. De acordo com a Amazon Web Services (s.d.), a REST é uma arquitetura de software que define algumas condições sobre como uma API deve funcionar, sendo usada para facilitar a comunicação em redes complexas, como a internet. Nesse modelo, a comunicação entre cliente e servidor acontece por meio de recursos, que são identificados por URLs e acessados através de métodos HTTP, como GET, POST, PATCH e DELETE. O método GET é usado para buscar informações, o POST para cadastrar novos dados, o PATCH para atualizar informações existentes e o DELETE para remover registros.

Outra característica importante das APIs RESTful é a separação entre cliente e servidor. Essa separação facilita o desenvolvimento e a manutenção do sistema, pois o front-end pode ser alterado sem que seja necessário modificar toda a estrutura do back-end, desde que a forma de comunicação entre eles continue a mesma. Além disso, as APIs RESTful trabalham com requisições independentes, ou seja, cada solicitação enviada ao servidor precisa conter as informações necessárias para ser entendida e processada (AWS, s.d.).

No sistema desenvolvido neste trabalho, a API RESTful é responsável por fazer a comunicação entre o front-end e o back-end. Assim, quando o usuário realiza alguma ação, como cadastrar, consultar, editar ou excluir dados, o front-end envia uma requisição para a API. Em seguida, o back-end processa essa requisição, aplica as regras necessárias e retorna uma resposta para a interface do sistema. Esse modelo torna o sistema mais organizado, facilita futuras melhorias e contribui para uma comunicação mais clara entre as partes da aplicação.

2.4.11.1. Swagger

O Swagger é uma ferramenta utilizada para documentar APIs REST de forma mais clara e organizada. Ele permite apresentar os recursos disponíveis na API, os métodos que podem ser utilizados, os parâmetros necessários e as respostas que podem ser retornadas pelo sistema (IBM, 2022).

No desenvolvimento do sistema, o Swagger foi utilizado para facilitar a visualização e o teste dos endpoints da API. Por meio do Swagger UI, é possível acessar a documentação pelo navegador, consultar as rotas disponíveis e testar as requisições diretamente pela interface (IBM, 2022). Isso contribui para o desenvolvimento e a validação das funcionalidades da aplicação.

2.4.12. TypeORM

O mapeamento objeto-relacional, conhecido como ORM (Object-Relational Mapping), é uma técnica de programação utilizada para facilitar a interação entre sistemas orientados a objetos e bancos de dados relacionais. Segundo Santos (2025), o ORM permite que os dados sejam manipulados em forma de objetos na aplicação, enquanto a própria ferramenta realiza o mapeamento desses objetos para as tabelas do banco de dados. Dessa forma, essa técnica reduz a necessidade de escrever consultas SQL diretamente no código, contribuindo para maior produtividade, legibilidade e manutenção da aplicação.

Nesse contexto, o TypeORM foi utilizado neste projeto como ferramenta de mapeamento objeto-relacional, responsável por realizar a comunicação entre a API desenvolvida em Nest.js e o banco de dados MySQL. De acordo com Santos (2024), o TypeORM é um ORM escrito em TypeScript e voltado ao ambiente Node.js, permitindo que os desenvolvedores trabalhem com bancos de dados relacionais a partir de uma abordagem orientada a objetos. A ferramenta possibilita a definição de entidades diretamente no código TypeScript e oferece suporte a diferentes tipos de relacionamentos, como um-para-um, um-para-muitos e muitos-para-muitos, o que contribui para a integração entre a lógica da aplicação e a estrutura do banco de dados.

3. METODOLOGIA

3.1. Metodologia de Pesquisa

A metodologia adotada neste trabalho possui abordagem qualitativa, tendo como objetivo compreender como a tecnologia pode auxiliar na organização e no gerenciamento de dízimos e contribuições nas comunidades religiosas por meio do desenvolvimento de um sistema web. Inicialmente foi realizada uma pesquisa bibliográfica, baseada na análise de artigos científicos, livros e materiais acadêmicos disponíveis online, com o objetivo de compreender conceitos relacionados a sistemas de informação, desenvolvimento web e utilização de tecnologias no apoio à gestão administrativa.

3.2. Metodologia de Desenvolvimento de Software

Para alcançar os objetivos propostos, inicialmente buscou-se compreender as principais necessidades relacionadas ao controle de contribuições financeiras em ambientes religiosos, identificando dificuldades presentes em processos realizados de forma manual, como o registro de dízimos, o controle de contribuições e contribuintes, bem como a geração de relatórios financeiros. A partir dessa análise, foi realizado o levantamento dos requisitos do sistema, possibilitando a definição das principais funcionalidades que irão compor a aplicação.

Após a etapa de levantamento de requisitos, foi desenvolvido um protótipo das interfaces do sistema por meio da plataforma Figma, com o objetivo de proporcionar uma melhor visualização da aplicação. Nesse protótipo foram representadas as principais telas e módulos do sistema, incluindo telas de cadastro, consultas e emissão de relatórios.

Com base nos requisitos identificados, foram elaborados os diagramas de modelagem do sistema. Para essa etapa foi utilizada a ferramenta Astah, com a finalidade de desenvolver o diagrama de caso de uso, permitindo a visualização dos usuários do sistema e das interações disponíveis para cada um deles. Posteriormente, foi realizada a elaboração dos Casos de Uso Expandidos, com o objetivo de descrever de forma detalhada as funcionalidades da aplicação.

Em seguida, foi realizada a modelagem do banco de dados, responsável por

armazenar as informações dos módulos do sistema. Para isso, foi utilizado o MySQL como Sistema Gerenciador de Banco de Dados (SGBD), e o MySQL Workbench para a construção do modelo lógico.

Por fim, foi realizada a implementação do sistema. Durante o desenvolvimento, o back-end foi executado em um servidor local, utilizando o ambiente Node.js e o framework NestJS. Desenvolvido em TypeScript, o back-end é responsável pelo processamento das regras de negócio, pela API e pela comunicação com o banco de dados MySQL por meio do TypeORM, ferramenta utilizada para mapear as entidades da aplicação e facilitar as operações realizadas no banco de dados. O front-end foi desenvolvido em JavaScript com o framework Vue.js e realiza a comunicação com a API por meio de requisições HTTP. Para documentar e auxiliar na validação dos endpoints disponibilizados pela API, utilizou-se o Swagger.

Para o gerenciamento e acompanhamento do desenvolvimento do sistema foi utilizada a metodologia ágil Scrum. Entre seus artefatos, foi empregado o Product Backlog, responsável por reunir todas as funcionalidades e tarefas do projeto, e o Sprint Backlog, utilizado para organizar e definir as atividades que serão realizadas em cada ciclo de desenvolvimento.

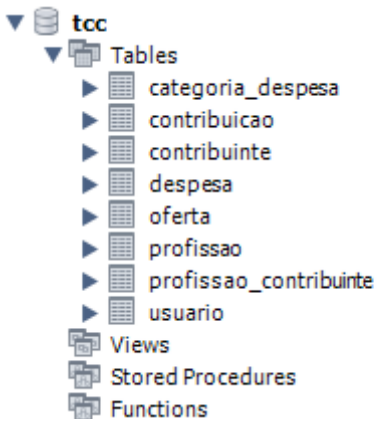
4. RESULTADOS E DISCUSSÕES

O presente projeto teve como resultado um sistema web funcional, estruturado para atender às necessidades específicas da comunidade alvo. O sistema conta com módulos destinados ao cadastro e gerenciamento de fiéis, bem como ao controle financeiro das entradas e saídas relacionadas aos dízimos, contribuições recebidas e despesas registradas, além da possibilidade de gerar relatórios de maneira mais fácil. Dessa forma, a aplicação busca substituir o controle financeiro realizado manualmente por um ambiente digital mais organizado, facilitando o registro, a consulta e o acompanhamento das informações. A seguir, será apresentada a estrutura do back-end, front-end e as telas do sistema, acompanhadas de descrições sobre suas respectivas funcionalidades.

4.1. Back-end

4.1.1. Banco de Dados

Figura 1: Estrutura do Banco de Dados



Fonte: Autoria própria (2026)

O banco de dados do sistema foi desenvolvido utilizando o MySQL, tendo como base o modelo lógico elaborado no projeto de software. Sua estrutura foi organizada para armazenar as informações necessárias ao funcionamento da aplicação, contemplando dados relacionados aos usuários, contribuintes, contribuições, ofertas, despesas, categorias de despesas e profissões.

A comunicação entre o back-end e o banco de dados ocorre por meio do TypeORM, responsável por realizar o mapeamento entre as entidades da aplicação e as tabelas correspondentes no banco. Dessa forma, as operações realizadas no sistema, como cadastros, consultas, atualizações e exclusões, são executadas a partir da API e refletidas no banco de dados de maneira estruturada. Além disso, os dados armazenados são utilizados em diferentes

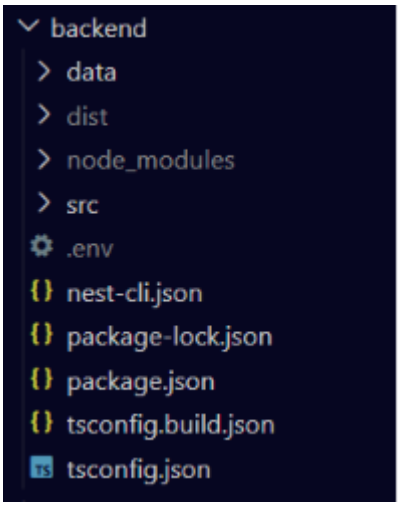
funcionalidades da aplicação, como o gerenciamento de contribuintes, o registro das entradas e saídas financeiras, a exibição de informações no dashboard e a geração de relatórios.

4.1.2. API Restful

4.1.2.1. Estrutura do projeto

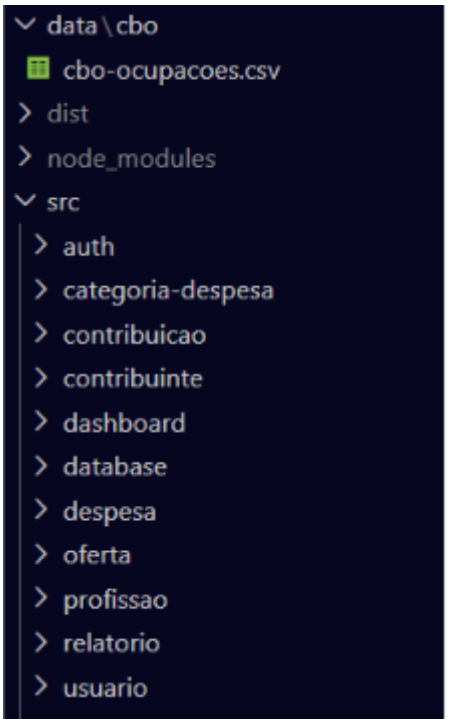
A API se encontra dividida nas seguintes pastas:

Figura 2: Estrutura do Back-end



Fonte: Autoria própria (2026)

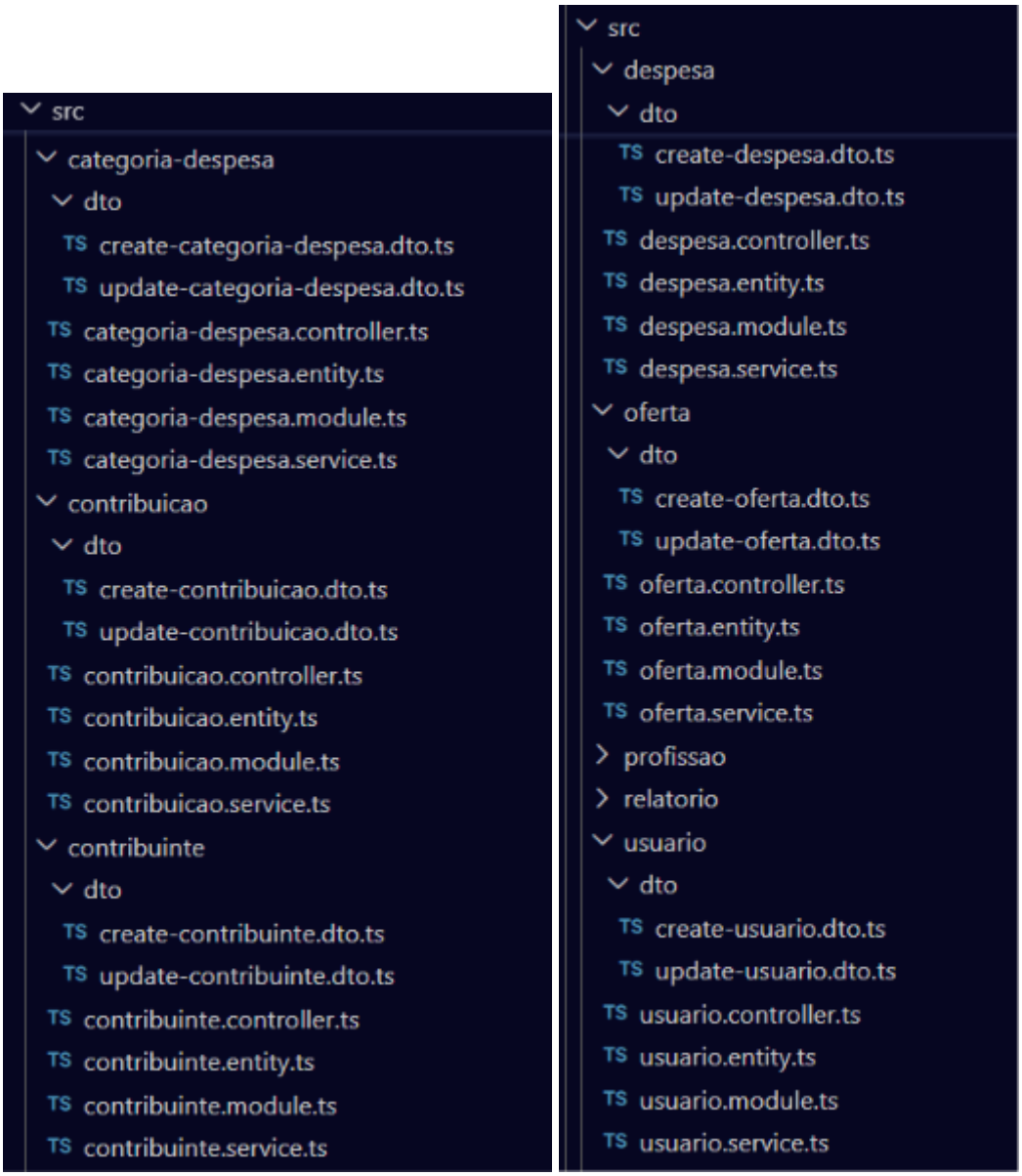
Figura 3: Estrutura dos diretórios do Back-end



Fonte: Autoria própria (2026)

Na pasta data encontra-se o arquivo utilizado para a importação das profissões exibidas no cadastro de contribuintes. O núcleo da API está localizado na pasta “src”, onde se encontram os principais módulos do sistema. Além disso, o projeto conta com o arquivo .env, responsável por armazenar configurações que a aplicação precisa para funcionar, mas que não devem ficar “fixas” diretamente no código, como porta de execução, host, nome do banco de dados, usuário e senha de acesso.

Figura 4: Estrutura do diretório src



Fonte: Autoria própria (2026)

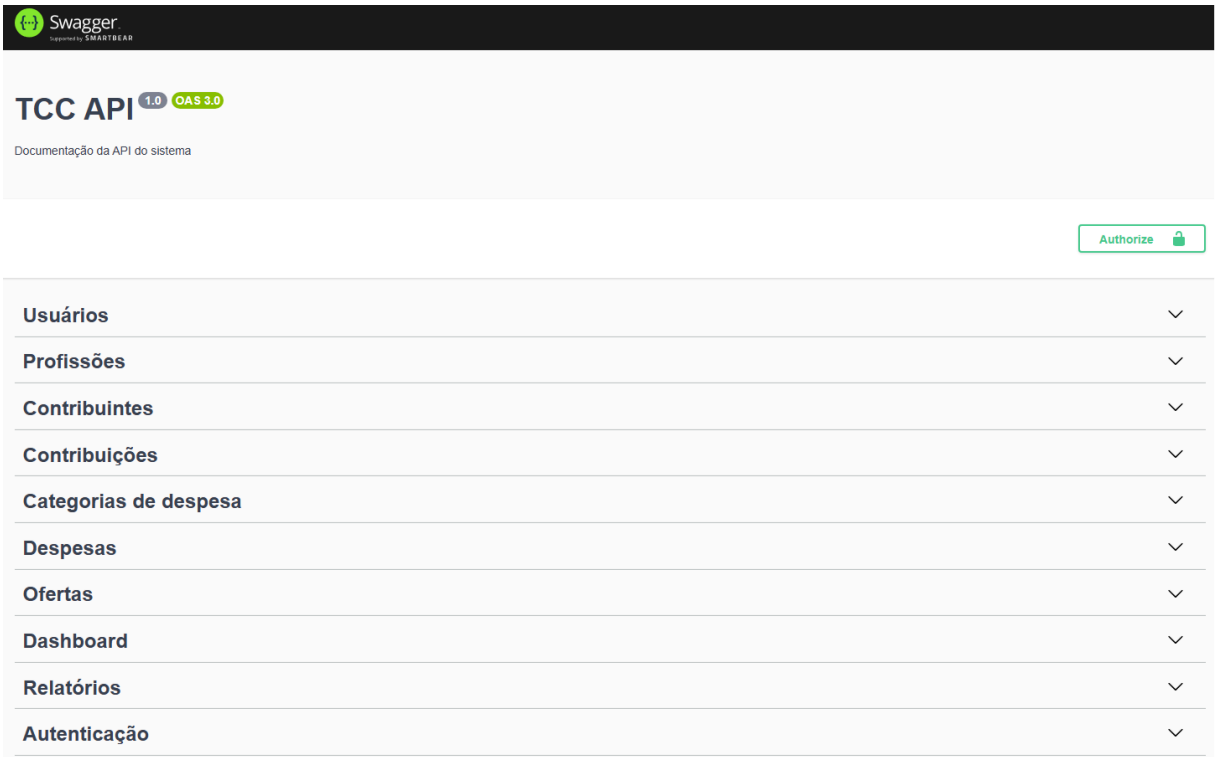
Mais especificamente dentro da pasta src, as funcionalidades da API são organizadas em pastas próprias e seguem uma estrutura padronizada. Em dto estão reunidos os arquivos

responsáveis por definir o formato dos dados recebidos nas requisições, tanto para criação quanto para atualização de registros. O arquivo controller.ts define as rotas da API e recebe as requisições enviadas pelo front-end, como cadastro, consulta, atualização e exclusão de registros, encaminhando-as ao service.ts, que concentra as regras de negócio e executa as operações necessárias para cada funcionalidade. O arquivo entity.ts representa a estrutura da entidade no banco de dados, sendo utilizado pelo TypeORM para realizar o mapeamento com as tabelas correspondentes. Por fim, o arquivo module.ts é responsável por organizar e integrar os elementos dessa estrutura dentro da aplicação, permitindo que controllers, services e demais dependências funcionem de forma integrada.

4.1.2.2. Documentação da API

A documentação das rotas da API foi realizada por meio do Swagger, sendo organizada separadamente de acordo com os módulos presentes na aplicação. A seguir, serão apresentadas as telas da documentação, ilustrando os módulos e suas respectivas funcionalidades.

Figura 5: Tela inicial da documentação da API

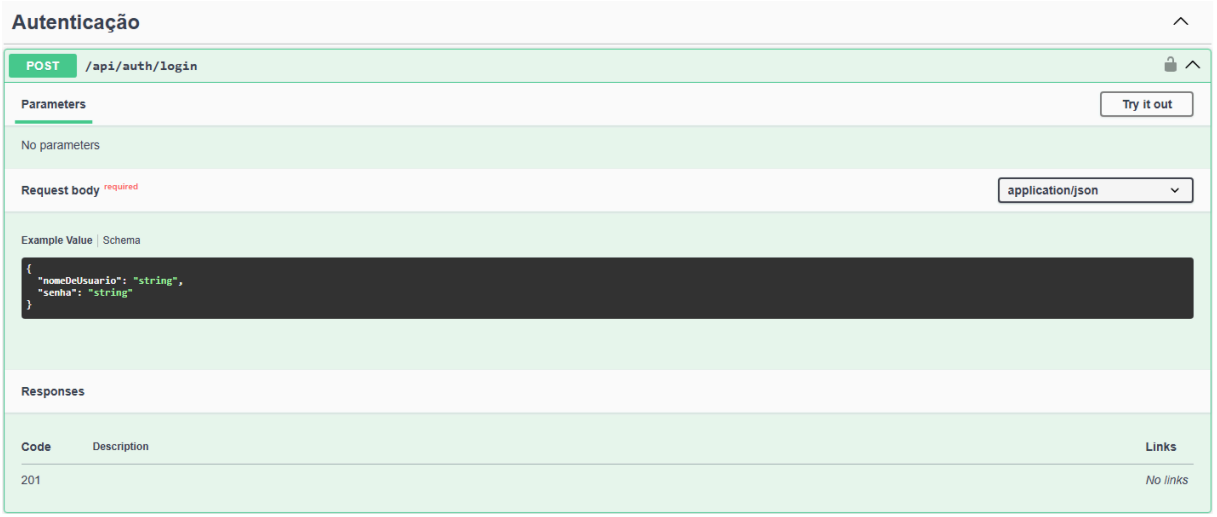


Fonte: Autoria própria (2026)

A Figura 5 apresenta a lista de rotas presentes no sistema, as quais serão explicadas individualmente a seguir. Essas rotas utilizam métodos do protocolo HTTP para indicar a

operação que será realizada sobre determinado recurso, onde o método GET é empregado para consultar dados; POST, para cadastrar novos registros; PATCH, para atualizar registros existentes; e DELETE, para realizar sua exclusão.

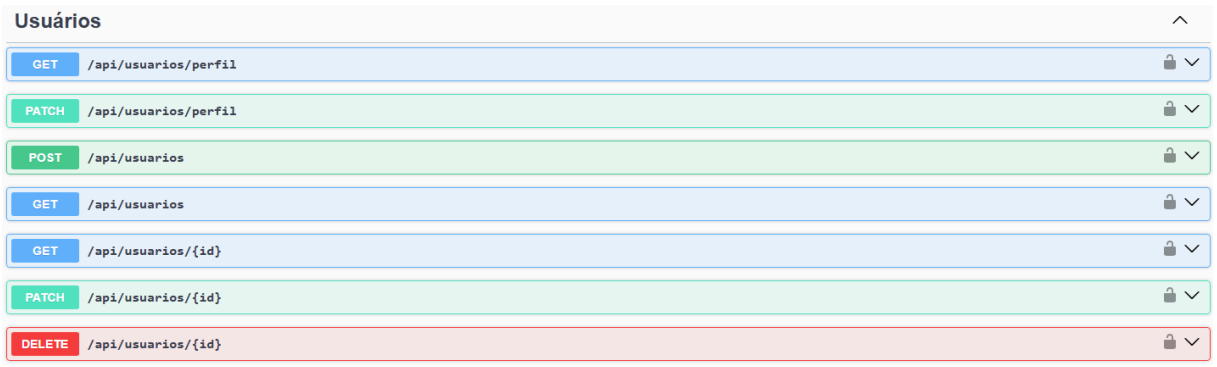
Figura 6: Rota de autenticação



Fonte: Autoria própria (2026)

A Figura 6 demonstra a rota responsável pela autenticação e pela emissão do token de acesso. Para que a requisição seja realizada corretamente, é necessário informar o nome de usuário e a senha cadastrados no sistema. Após a validação dessas informações, a API retorna um token de acesso, utilizado para identificar o usuário autenticado e controlar quais funcionalidades ele poderá acessar. Dessa forma, o sistema permite restringir determinadas ações conforme o tipo de usuário.

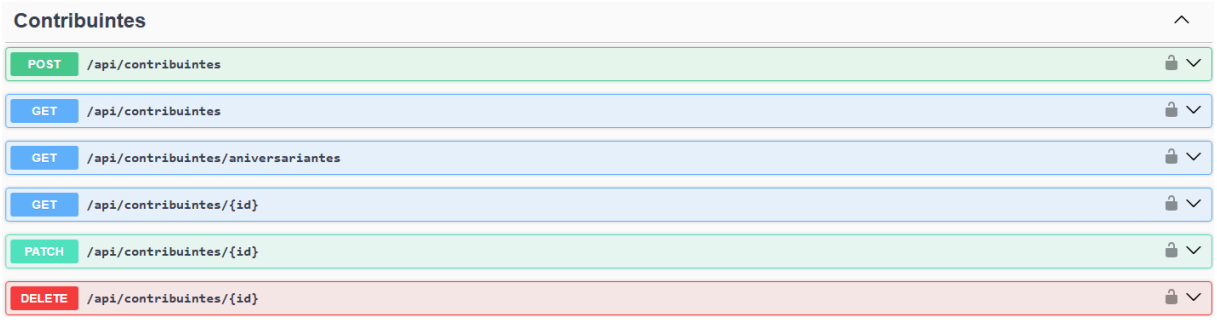
Figura 7: Rota de usuário



Fonte: Autoria própria (2026)

A Figura 7 apresenta os endpoints relacionados ao gerenciamento de usuários. As operações GET /api/usuarios/perfil e PATCH /api/usuarios/perfil permitem, respectivamente, consultar e atualizar os dados do usuário autenticado. O endpoint POST /api/usuarios permite cadastrar um novo usuário, enquanto GET /api/usuarios retorna a lista de usuários cadastrados. Por fim, as rotas que recebem o parâmetro {id} possibilitam consultar, atualizar ou excluir um usuário específico. Essas operações são restritas aos usuários com nível de acesso de administrador.

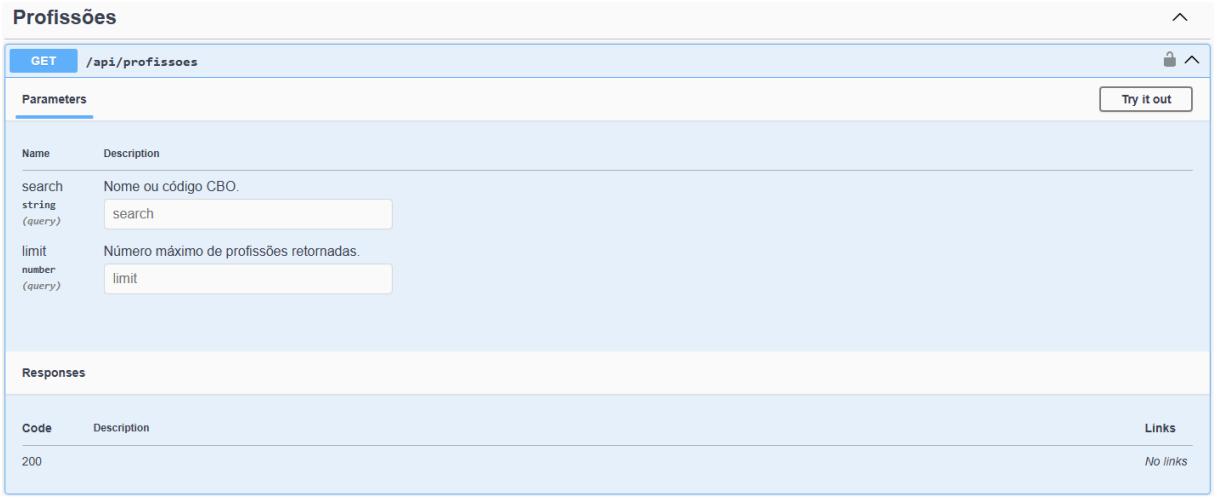
Figura 8: Rota de contribuintes



Fonte: Autoria própria (2026)

A Figura 8 exibe os endpoints responsáveis pelo gerenciamento dos contribuintes. O endpoint POST /api/contribuintes permite cadastrar um novo contribuinte, enquanto GET /api/contribuintes retorna os contribuintes cadastrados e possibilita a pesquisa por nome por meio do parâmetro opcional search. O endpoint GET /api/contribuintes/aniversariantes retorna os aniversariantes do mês informado no parâmetro mes e caso esse parâmetro não seja fornecido, o sistema utiliza o mês atual. Por fim, as rotas que recebem o parâmetro {id} permitem consultar, atualizar ou excluir um contribuinte específico por meio dos métodos GET, PATCH e DELETE, respectivamente.

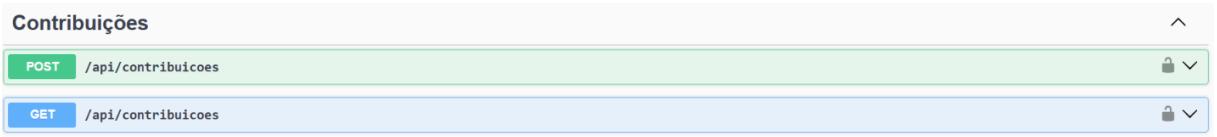
Figura 9: Rota de profissões



Fonte: Autoria própria (2026)

A Figura 9 detalha o endpoint GET /api/profissoes, responsável pela consulta das profissões disponíveis no sistema. A pesquisa pode ser feita pelo nome ou pelo código da Classificação Brasileira de Ocupações (CBO) por meio do parâmetro search, enquanto o parâmetro limit define a quantidade máxima de resultados retornados. Esse endpoint auxilia no preenchimento e na associação das profissões durante o cadastro dos contribuintes.

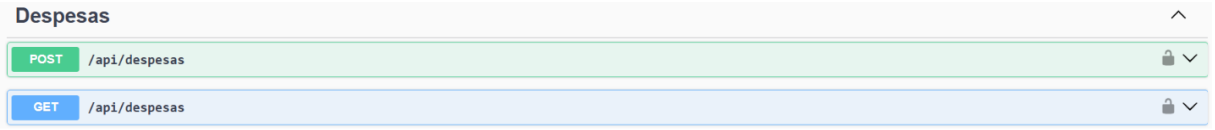
Figura 10: Rota de contribuições



Fonte: Autoria própria (2026)

A Figura 10 demonstra os endpoints responsáveis pelo gerenciamento das contribuições registradas no sistema. O método POST /api/contribuicoes permite cadastrar uma nova contribuição, onde é associada a um contribuinte e ao usuário responsável pelo registro. Já em GET /api/contribuicoes é possível consultar as contribuições cadastradas no sistema.

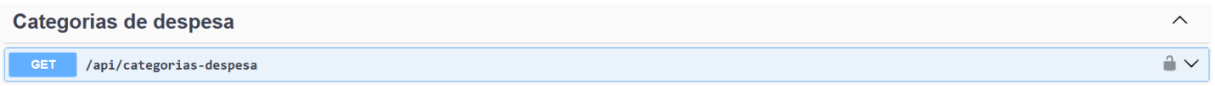
Figura 11: Rota de despesas



Fonte: Autoria própria (2026)

A Figura 11 exibe os endpoints responsáveis pelo gerenciamento das despesas registradas. O método POST /api/despesas permite cadastrar uma nova despesa, sendo associada ao usuário responsável pelo registro. Já o método GET /api/despesas possibilita consultar as despesas cadastradas no sistema.

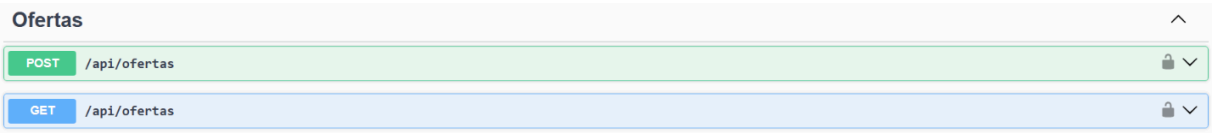
Figura 12: Rota de categoria de despesa



Fonte: Autoria própria (2026)

O endpoint GET /api/categorias-despesa disponibiliza as categorias existentes para preenchimento da combobox. Caso o usuário informe uma nova categoria no campo do formulário, seu cadastro ocorre internamente por meio da requisição POST /api/despesas, dispensando um endpoint específico para criação de novas categorias.

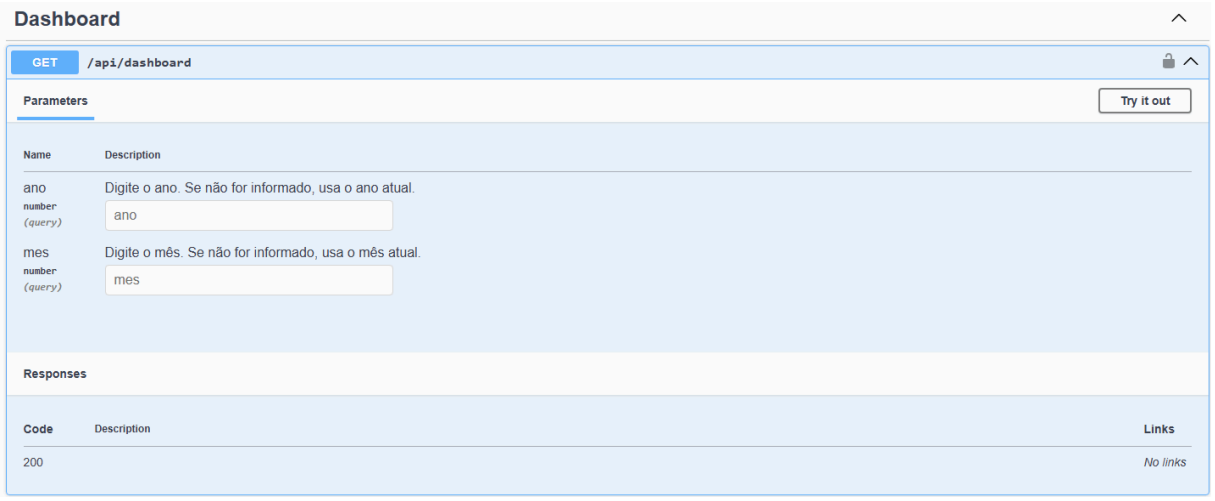
Figura 13: Rota de ofertas



Fonte: Autoria própria (2026)

A Figura 13 ilustra os endpoints responsáveis pelo gerenciamento das ofertas registradas no sistema. O método POST /api/ofertas permite cadastrar uma nova oferta. Já o método GET /api/ofertas possibilita consultar as ofertas cadastradas e possui os parâmetros opcionais inicio e fim, utilizados para delimitar o período da consulta.

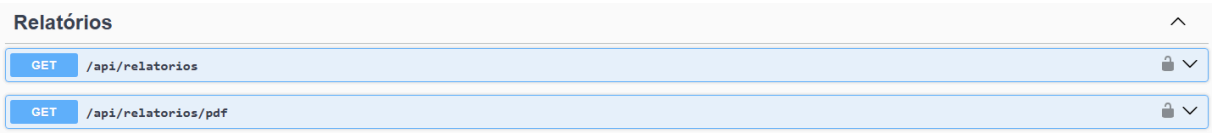
Figura 14: Rota do dashboard



Fonte: Autoria própria (2026)

A Figura 14 apresenta o endpoint responsável pelo fornecimento das informações exibidas no dashboard do sistema. O método GET /api/dashboard retorna um resumo das principais informações do sistema, como a quantidade arrecadada no mês, o valor total das despesas, o saldo no mês, entre outros. A consulta possui os parâmetros opcionais ano e mes, que permitem selecionar o período desejado. Caso não sejam informados, o sistema utiliza o mês e o ano atuais.

Figura 15: Rota de relatórios



Fonte: Autoria própria (2026)

A Figura 15 demonstra os endpoints destinados à geração de relatórios. O método GET /api/relatorios retorna os dados conforme o tipo selecionado, enquanto o método GET /api/relatorios/pdf gera o mesmo conteúdo em formato PDF. As operações recebem o parâmetro obrigatório tipo e os parâmetros opcionais inicio e fim, utilizados para definir o período da consulta ou da emissão do relatório em PDF.

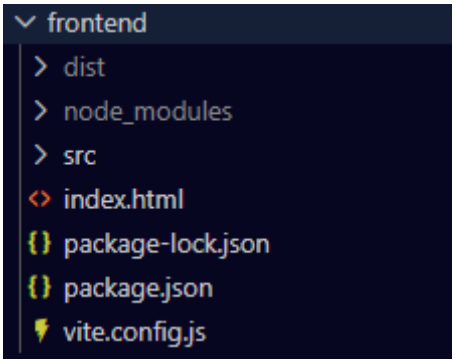
4.2.Front-end

4.2.1. Aplicação Web

4.2.1.1. Estrutura do projeto

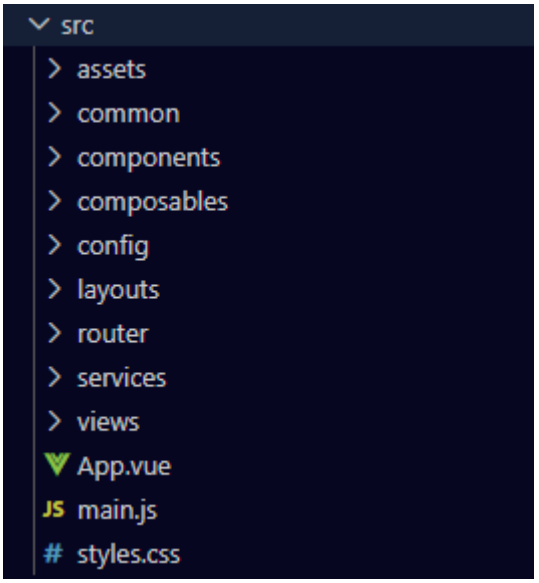
A pasta da aplicação web do projeto se encontra dividida da seguinte forma:

Figura 16: Estrutura do Front-end



Fonte: Autoria própria (2026)

Figura 17: Estrutura do diretório src

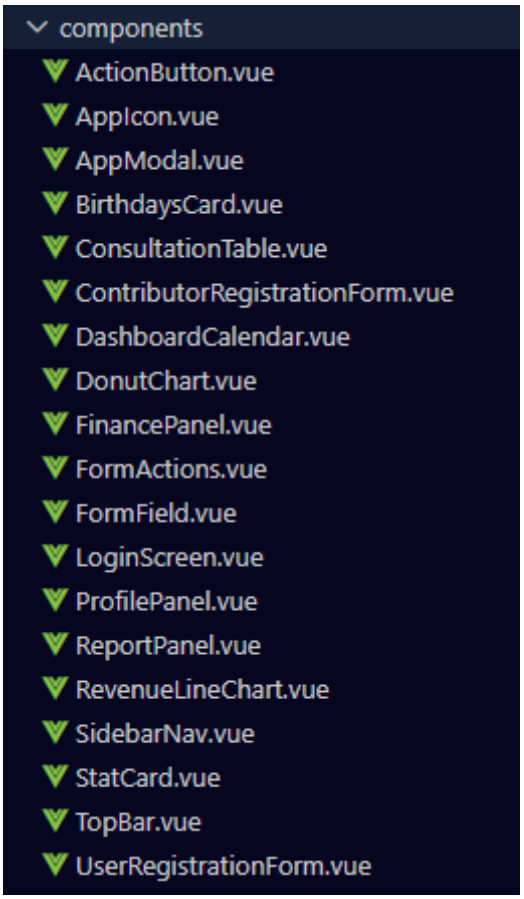


Fonte: Autoria própria (2026)

O arquivo index.html, localizado na raiz do projeto, é responsável por fornecer a estrutura inicial da página web e definir o ponto em que a aplicação Vue.js será renderizada no navegador. Já a pasta src concentra o núcleo da aplicação, reunindo os arquivos responsáveis pela construção e funcionamento da interface. Esses arquivos são organizados em diretórios específicos, separando componentes, páginas, rotas, serviços, layouts e demais recursos utilizados na montagem da aplicação. O arquivo main.js é o ponto de partida da

aplicação, responsável por importar o componente principal App.vue e as rotas definidas no projeto, carregar os estilos globais por meio da importação do arquivo styles.css e executar as demais inicializações necessárias ao funcionamento da aplicação. O arquivo App.vue, por sua vez, utiliza o componente RouterView do Vue.js, permitindo que o conteúdo exibido na tela seja alterado de acordo com a rota acessada, como login, dashboard, cadastros, consultas e demais páginas do sistema. Já o styles.css reúne os estilos globais da aplicação, incluindo as variáveis de cores, fontes, dimensões e demais definições visuais utilizadas para manter a interface padronizada.

Figura 18: Estrutura do diretório components

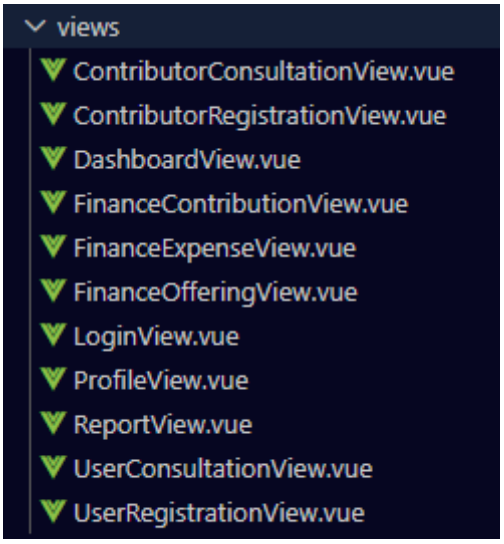


Fonte: Autoria própria (2026)

Dentro da pasta components, encontram-se os componentes reutilizáveis da aplicação. Segundo aponta a documentação oficial do Vue.js, os componentes permitem dividir a interface em partes independentes e reutilizáveis, facilitando a organização da aplicação. Dessa forma, elementos visuais utilizados em diferentes telas podem ser criados separadamente e reaproveitados conforme a necessidade do sistema (VUE.JS, 2026).

No projeto, a pasta components reúne elementos reutilizados em diferentes telas, como botões, ícones, modais, campos de formulário, tabelas, cards, gráficos, menu lateral e barra superior. Também inclui componentes específicos para funcionalidades do sistema, como formulários de cadastro, painel financeiro, painel de relatórios e painel de perfil. Essa organização contribui para reduzir a repetição de código e facilitar a manutenção da interface.

Figura 19: Estrutura do diretório views

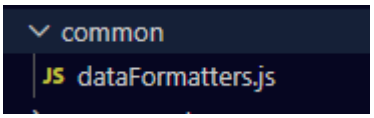


Fonte: Autoria própria (2026)

A pasta views reúne os arquivos responsáveis pelas principais telas da aplicação. Cada arquivo representa uma página acessada pelo usuário por meio das rotas definidas no sistema, como login, dashboard, perfil, relatórios, cadastros e consultas, além das telas de controle financeiro. Essas views organizam a estrutura de cada página e utilizam os componentes reutilizáveis da pasta components para compor a interface e executar as funcionalidades necessárias.

Os demais arquivos utilizados na configuração e no funcionamento da interface foram organizados em diretórios específicos, de acordo com suas responsabilidades dentro da aplicação. Essa divisão permite separar diferentes funções do front-end, contribuindo para a organização do código, facilitando a manutenção da aplicação e possibilitando futuras melhorias.

Figura 20: Estrutura do diretório common



Fonte: Autoria própria (2026)

A pasta common contém arquivos com funções auxiliares utilizadas em diferentes partes da aplicação. No projeto, o arquivo dataFormatters.js reúne funções responsáveis pela formatação de dados exibidos na interface, como valores monetários e datas. Além disso, também possui funções para trabalhar com datas no formato utilizado pelos campos do sistema e para verificar se uma data informada é futura.

Figura 21: Estrutura do diretório composable

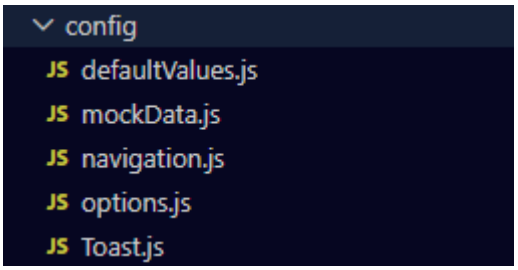
Fonte: Autoria própria (2026)

De acordo com a documentação oficial do Vue.js , composables são funções utilizadas para encapsular e reutilizar lógicas com estado dentro de uma aplicação Vue. Esse tipo de lógica envolve informações que podem mudar ao longo do tempo, como dados de autenticação, mensagens exibidas na interface ou outros comportamentos utilizados em diferentes partes do sistema (VUE.JS, 2026).

Para a criação dos composables, utiliza-se a Composition API, que, segundo a documentação oficial do Vue.js, é um conjunto de APIs que permite construir componentes a partir de funções importadas, como ref, reactive, computed e onMounted. Esses recursos possibilitam controlar estados reativos, valores computados e ações relacionadas ao ciclo de vida dos componentes (VUE.JS, 2024).

No projeto, o arquivo useAuth.js utiliza recursos da Composition API, como ref e computed, para controlar a autenticação no front-end. A variável currentUser, criada com ref, armazena as informações do usuário autenticado, enquanto a variável isAuthenticated, criada com computed, verifica se há um usuário carregado e um token de acesso disponível. Já o arquivo useToast.js possui a função showToast, responsável por exibir mensagens ao usuário, como avisos de sucesso, erro ou confirmação durante as operações realizadas na aplicação.

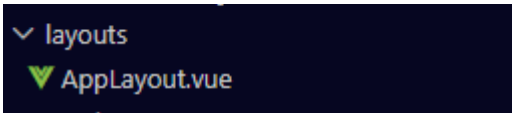
Figura 22: Estrutura do diretório config



Fonte: Autoria própria (2026)

Na pasta config, encontram-se arquivos responsáveis por armazenar configurações utilizadas em diferentes partes da interface. Em defaultValues.js está as definições dos valores padrões utilizados em todos os formulários, que no caso são os campos vazios. O arquivo navigation.js define os itens do menu lateral, os títulos exibidos nas telas e os botões de acesso rápido presentes no dashboard. O arquivo options.js reúne opções utilizadas em campos de combobox mais simples do sistema, como tipos de contribuição, tipos de celebração, entre outros. Já o arquivo Toast.js contém o código da configuração visual e comportamental das mensagens de confirmação exibidas ao usuário, presentes na biblioteca do Toast disponível online (ADERINOKUN, 2016).

Figura 23: Estrutura do diretório layouts



Fonte: Autoria própria (2026)

Já a pasta layouts, contém o arquivo AppLayout.vue, responsável por definir a estrutura principal das telas do sistema. Esse layout organiza a interface em menu lateral, barra superior e área central de conteúdo, onde as páginas são exibidas conforme a rota acessada pelo usuário. Além disso, o arquivo controla os itens de navegação disponíveis de acordo com o nível de acesso do usuário autenticado, exibindo as opções de cadastro e consulta de usuários apenas para administradores.

Figura 24: Estrutura do diretório router

Fonte: Autoria própria (2026)

A pasta router contém o arquivo index.js, responsável pela configuração das rotas da aplicação e pelo controle das regras de navegação. Nesse arquivo, são definidas as restrições de acesso com base na autenticação do usuário, verificando se ele está logado no sistema, e também no tipo de usuário, diferenciando as permissões entre usuários padrão e administradores.

Figura 25: Estrutura do diretório services

Fonte: Autoria própria (2026)

A pasta services contém o arquivo api.js, responsável por concentrar a comunicação entre o front-end e a API do sistema. Nesse arquivo, é definida a URL base da API e são organizadas as funções utilizadas para realizar requisições HTTP, como GET, POST, PATCH e DELETE. Além disso, o arquivo gerencia o token de autenticação armazenado no navegador. Quando o usuário está autenticado, esse token é adicionado ao cabeçalho das requisições, permitindo que a API identifique e autorize o acesso às funcionalidades protegidas do sistema.

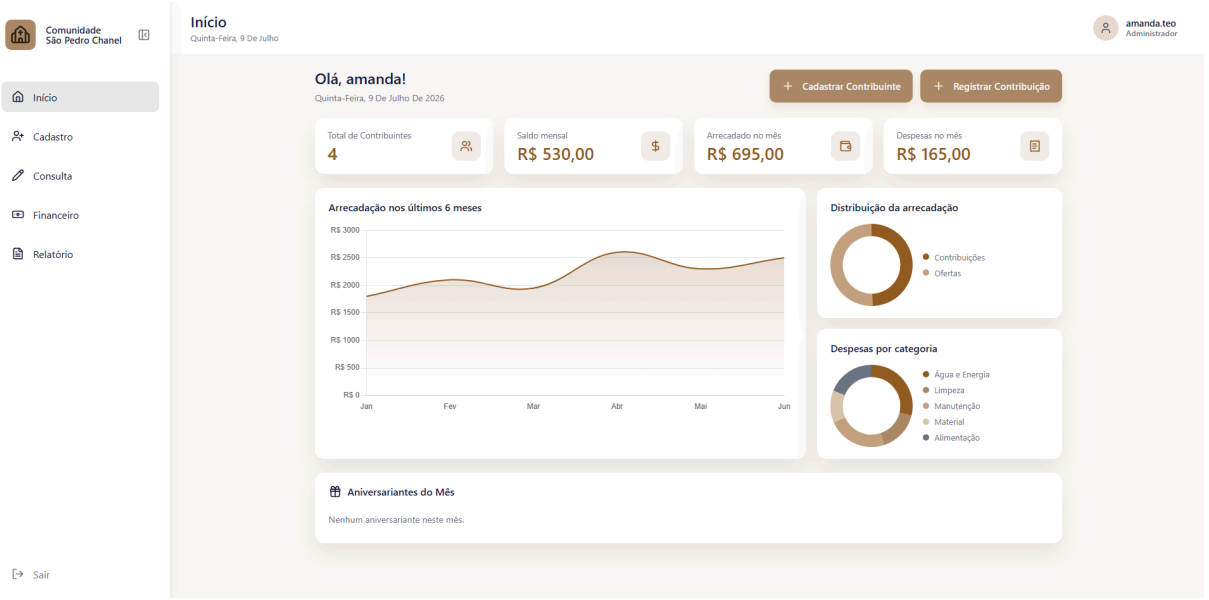
4.2.1.2. Interface e funcionalidades

Figura 26: Tela de login

Fonte: Autoria própria (2026)

Ao acessar a aplicação, o usuário se depara com a tela de login, que possui campos para o preenchimento do nome de usuário e da senha previamente cadastrados por um usuário administrador. Após a validação das credenciais, o sistema libera o acesso à aplicação de acordo com o nível de permissão do usuário.

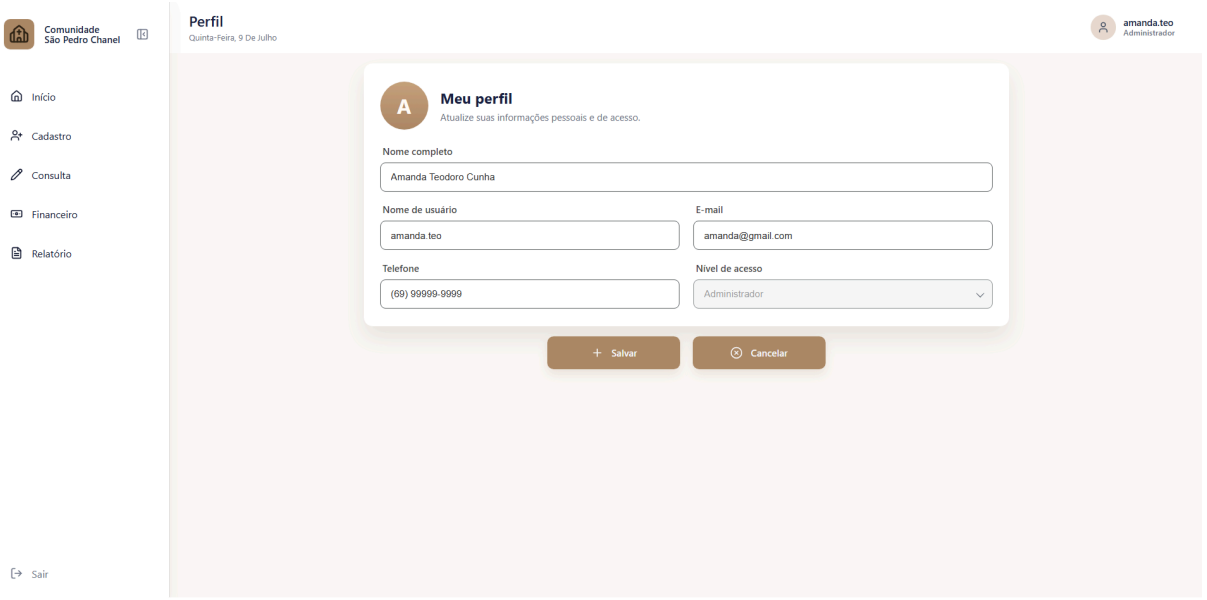
Figura 27: Dashboard



Fonte: Autoria própria (2026)

Após a autenticação, o usuário é direcionado à tela principal do sistema, onde inicialmente é apresentado o dashboard com as principais informações da aplicação. Nessa tela, há dois botões de acesso rápido para o cadastro de novos contribuintes e para o registro de uma nova contribuição, por serem funcionalidades diretamente relacionadas às principais necessidades do sistema. Logo abaixo, são exibidos cards informativos com dados como a quantidade de contribuintes cadastrados, o saldo mensal, o total de entradas e o total de saídas registradas. Em seguida, são apresentados gráficos que permitem a visualização da movimentação financeira da comunidade de forma mais clara. Por fim, há um card destinado à exibição dos aniversariantes do mês atual. A tela também conta com o menu lateral, que permite a navegação entre os módulos do sistema, como cadastros, consultas, entre outros.

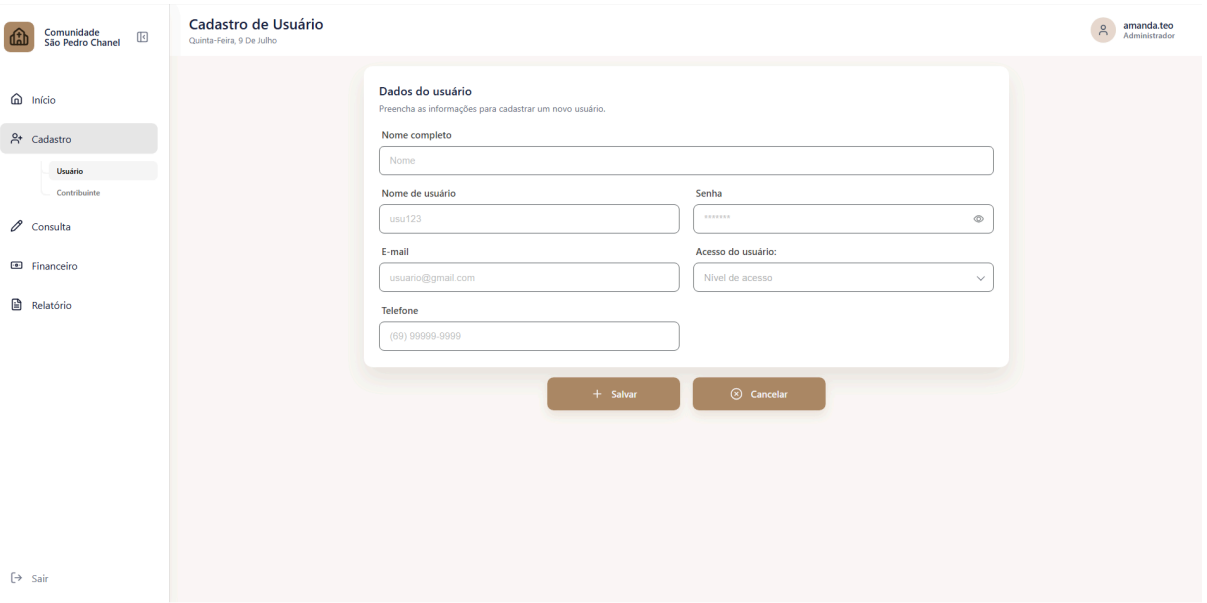
Figura 28: Tela de perfil do usuário



Fonte: Autoria própria (2026)

No canto superior direito, encontra-se a seção de acesso ao perfil do usuário, que permite a atualização de informações pessoais, como nome de usuário, senha, telefone e e-mail. O campo referente ao nível de acesso fica bloqueado para edição, sendo exibido apenas com a finalidade de informar o nível de permissão do usuário autenticado no sistema. A alteração desse nível é realizada somente na edição de usuários, funcionalidade exclusiva dos administradores, conforme será apresentado nas próximas seções.

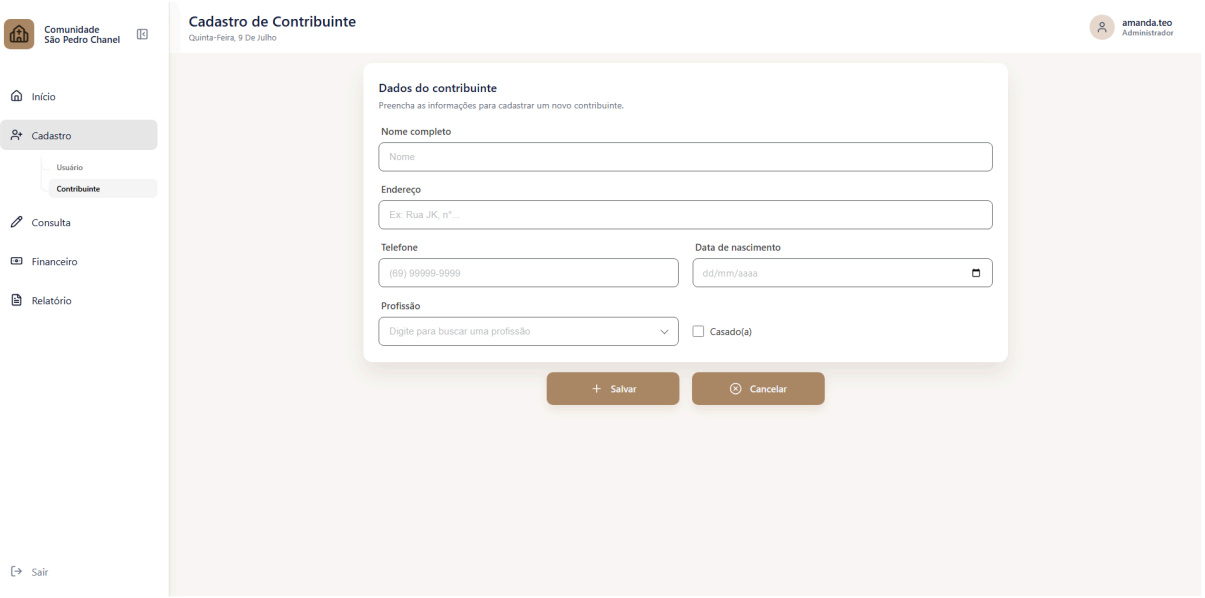
Figura 29: Tela de cadastro de usuário



Fonte: Autoria própria (2026)

Partindo para o menu lateral, a primeira sessão logo abaixo de Início, se encontra os formulários de cadastros de contribuintes, para usuários padrão, e adicional do cadastro de novos usuários para administradores. O cadastro de usuários conta com um formulário com campos para o nome completo, nome de usuário, senha, email, telefone e nível de acesso. Ao clicar em salvar, as informações preenchidas são enviadas para a API, onde são processadas e posteriormente armazenadas no banco de dados.

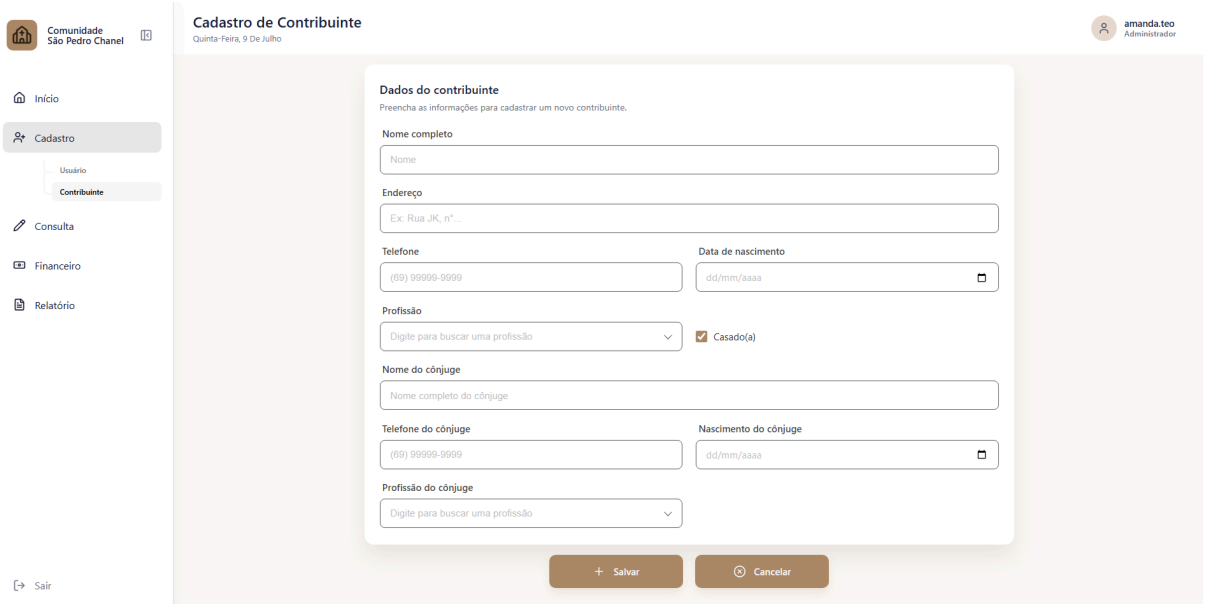
Figura 30: Tela de cadastro de contribuinte



Fonte: Autoria própria (2026)

Já o cadastro de contribuintes conta com um formulário inicial que solicita informações como nome completo, endereço, telefone, data de nascimento e profissão, além de uma caixa de marcação para indicar se o contribuinte é casado ou não.

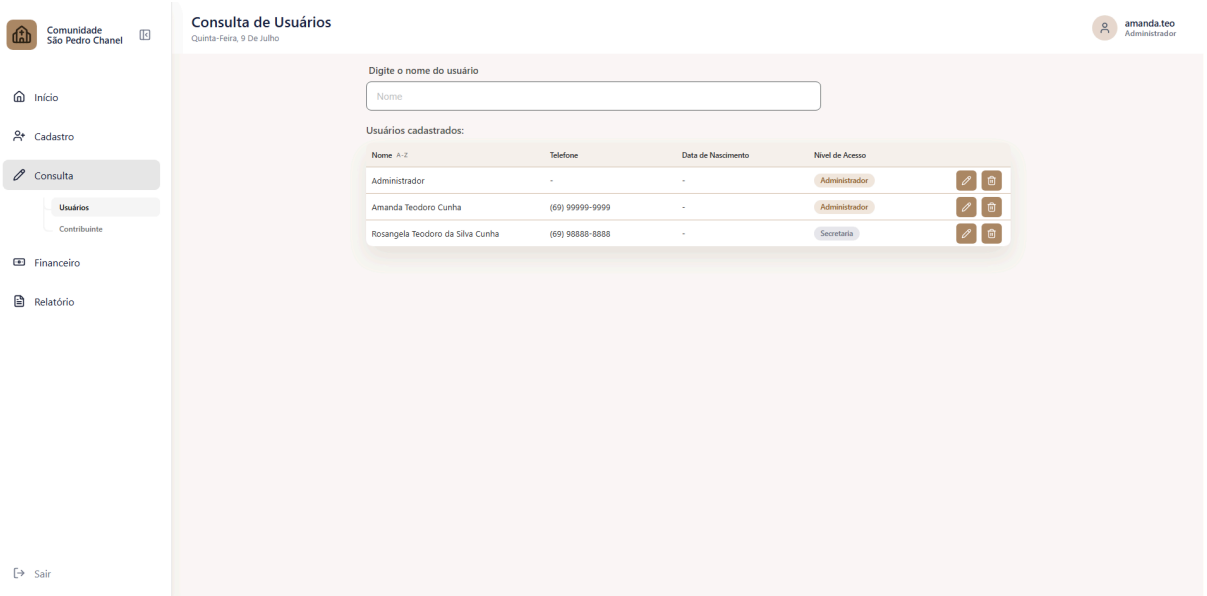
Figura 31: Tela de cadastro de contribuinte com cônjuge



Fonte: Autoria própria (2026)

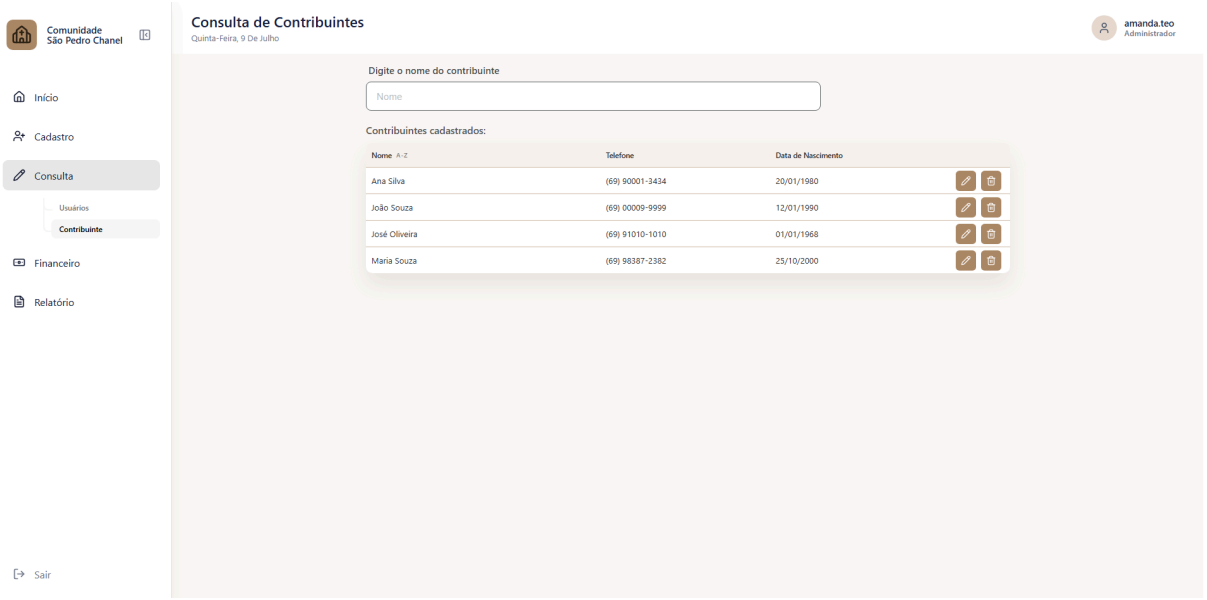
Caso a opção Casado(a) seja marcada, um novo formulário é exibido logo abaixo para o preenchimento das informações do cônjuge. Nesse caso, o endereço informado anteriormente é reaproveitado no registro, e o cônjuge também é salvo como contribuinte no banco de dados.

Figura 32: Tela de consulta de usuários



Fonte: Autoria própria (2026)

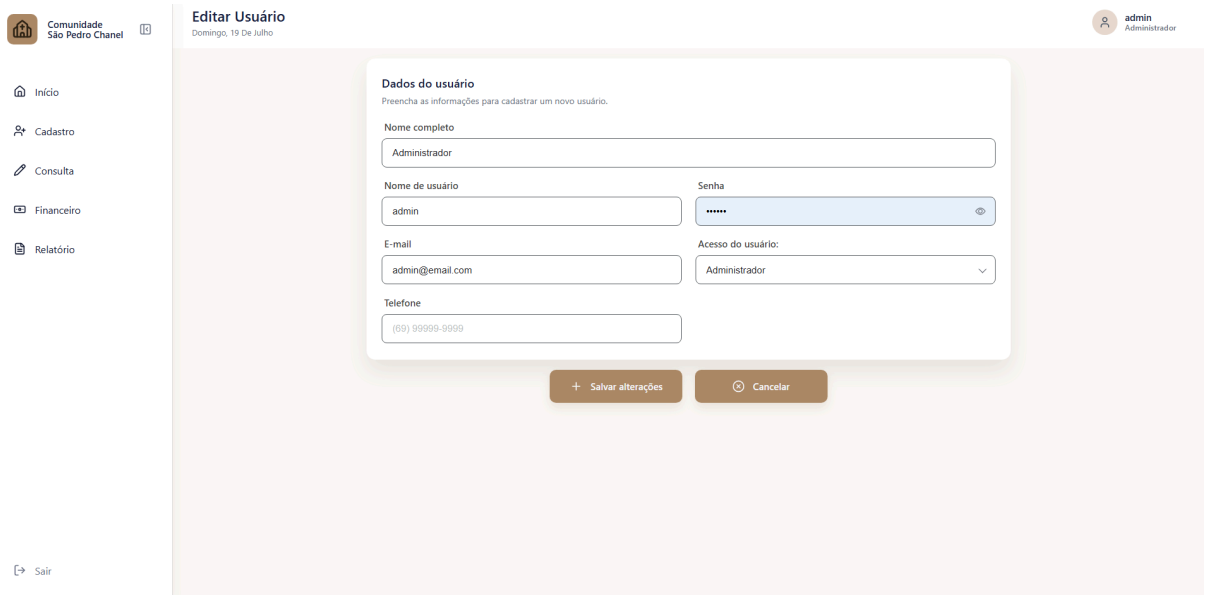
Figura 33: Tela de consulta de contribuintes



Fonte: Autoria própria (2026)

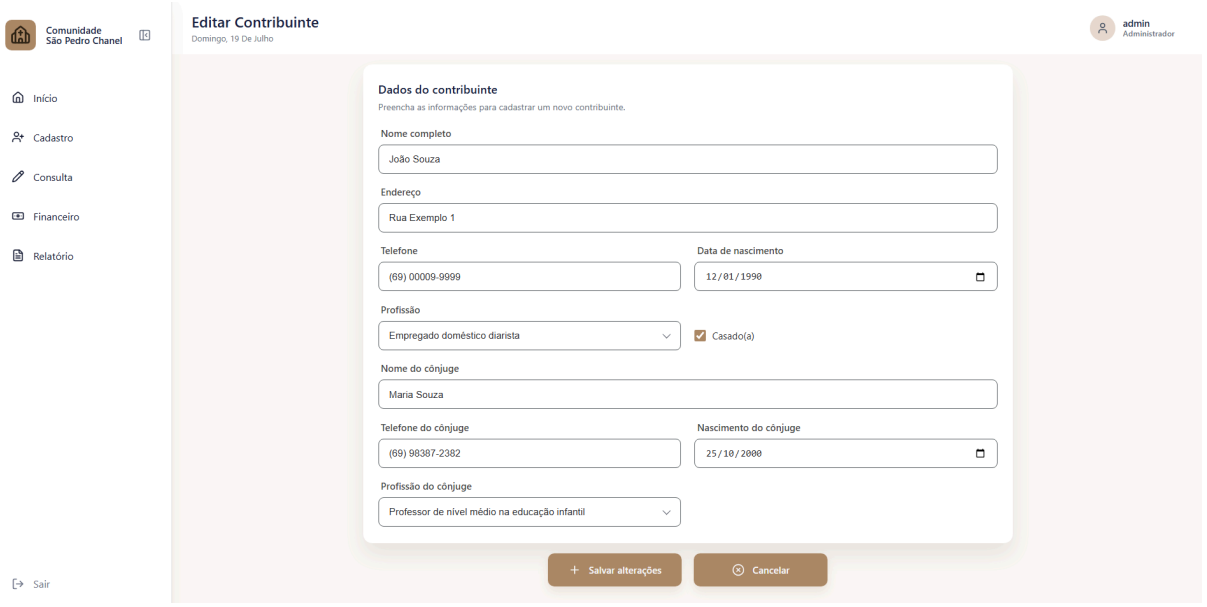
Abaixo da seção Cadastro, no menu lateral, encontram-se as páginas de consulta, seguindo o mesmo padrão de nível de acesso do módulo de cadastro. Nessas telas, é exibida uma lista com os usuários ou contribuintes cadastrados no sistema, além de uma barra de pesquisa acima da lista para facilitar a identificação de registros específicos. Em cada item listado, são apresentadas as principais informações do registro, acompanhadas de botões para edição e exclusão.

Figura 34: Tela de edição de usuário



Fonte: Autoria própria (2026)

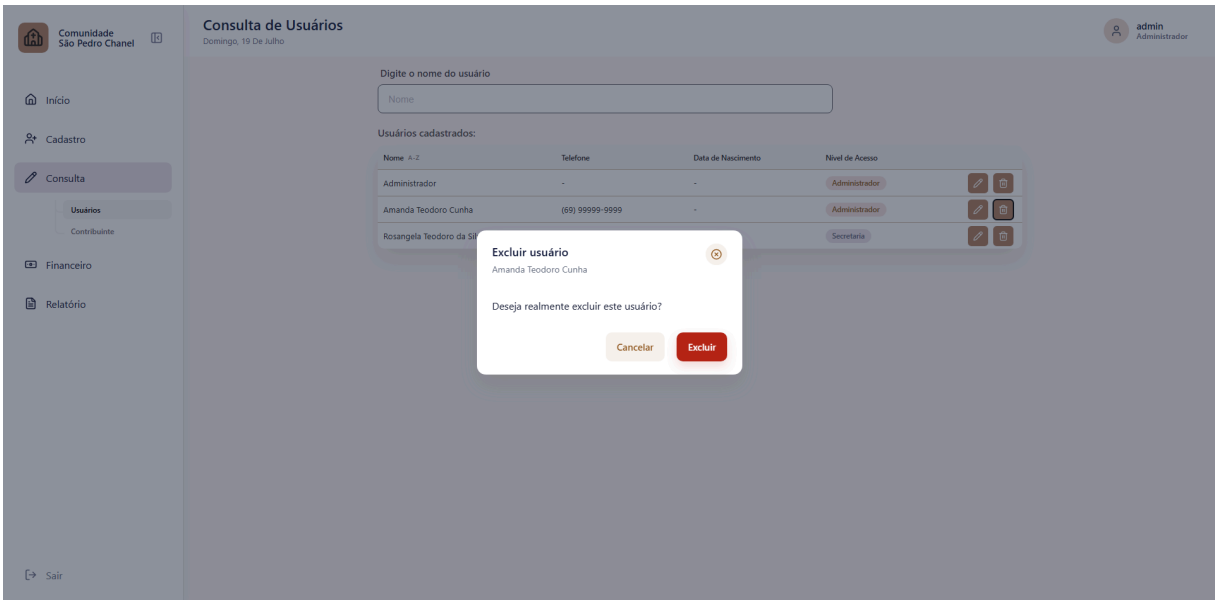
Figura 35: Tela de edição de contribuinte



Fonte: Autoria própria (2026)

Caso o usuário clique no botão de edição, ele é direcionado ao formulário correspondente, já preenchido com as informações do usuário ou contribuinte selecionado, permitindo a realização das alterações necessárias. Após concluir as modificações, ao clicar em salvar, os dados são enviados para a API e atualizados no banco de dados. Caso selecione a opção cancelar, o usuário retorna para a tela de consulta sem realizar alterações.

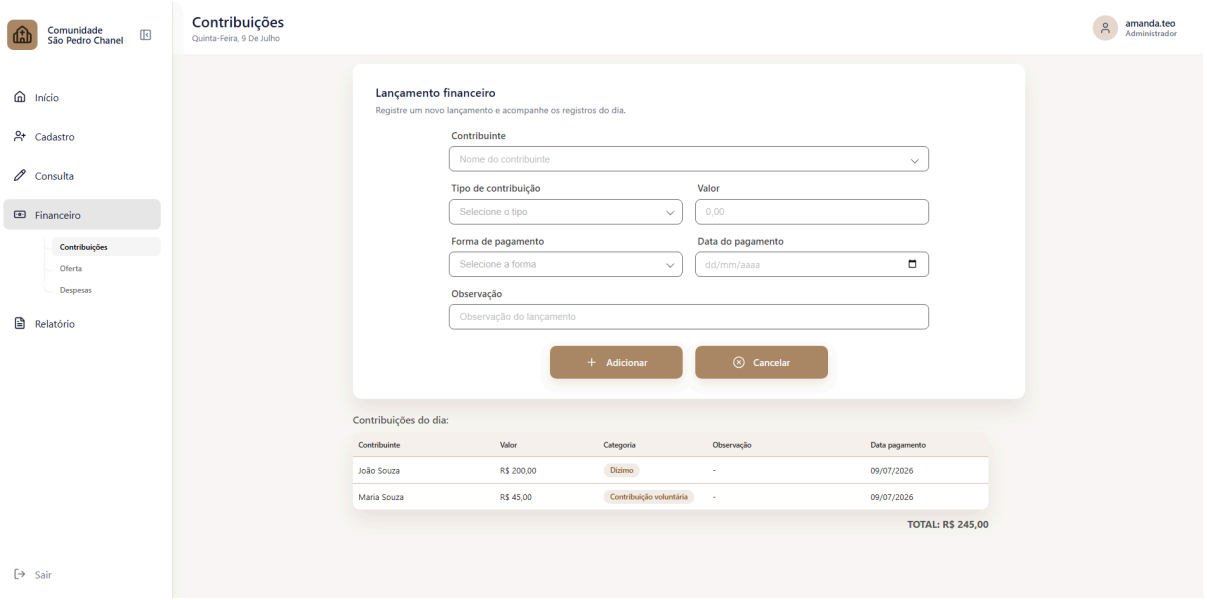
Figura 36: Mensagem de confirmação de exclusão



Fonte: Autoria própria (2026)

Caso o usuário clique no ícone de lixeira, disponível na tela de consulta, o sistema exibe uma mensagem de aviso solicitando a confirmação da ação. Se o usuário confirmar a exclusão, o registro é removido do banco de dados. Caso clique em cancelar, a ação é interrompida e o usuário retorna à tela de consulta sem que nenhuma alteração seja realizada.

Figura 37: Tela de registro de contribuições

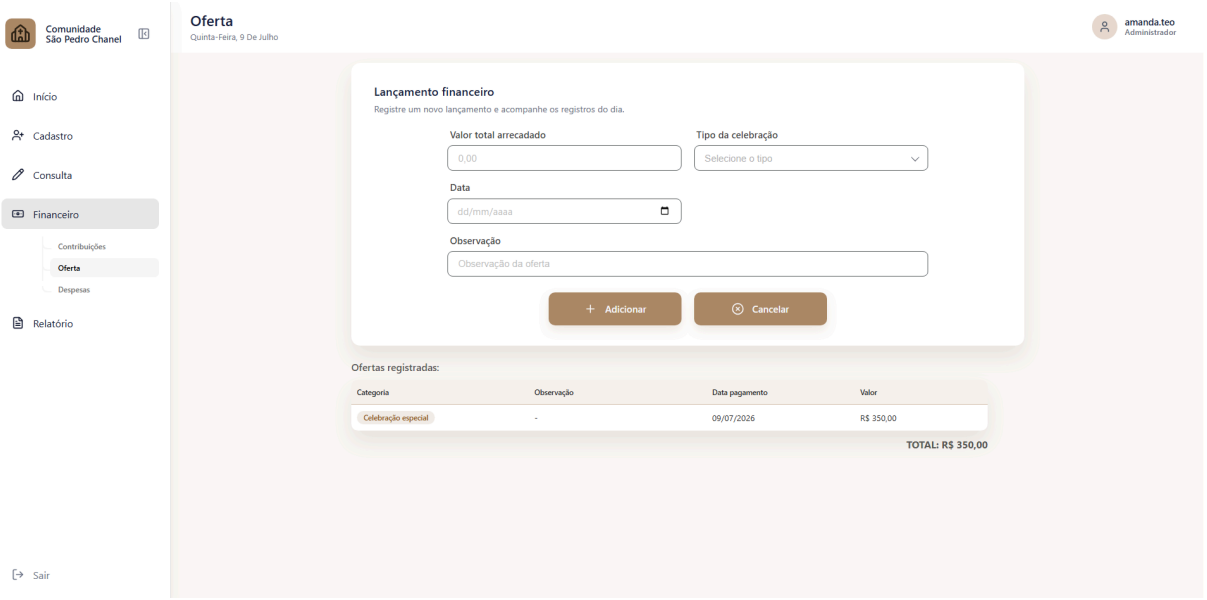


Fonte: Autoria própria (2026)

Na seção Financeiro do sistema, encontram-se os módulos destinados ao registro das contribuições, como dízimos e contribuições voluntárias, das ofertas recebidas durante as celebrações e das despesas da comunidade. Ao clicar em Contribuições, o usuário é direcionado a um formulário que solicita informações como o nome do contribuinte, selecionado por meio de uma combobox com os contribuintes já cadastrados no sistema, o tipo da contribuição, o valor, a forma de pagamento, a data do pagamento e um campo opcional de observação.

Ao clicar no botão salvar, o registro da contribuição é enviado para a API e armazenado no banco de dados. As principais informações cadastradas são exibidas logo abaixo do formulário, em formato de lista, permitindo ao usuário acompanhar os lançamentos realizados. Essa listagem é atualizada diariamente, exibindo apenas os registros correspondentes ao dia em questão. Caso o usuário clique em Cancelar, o formulário é limpo, possibilitando a inserção de novas informações.

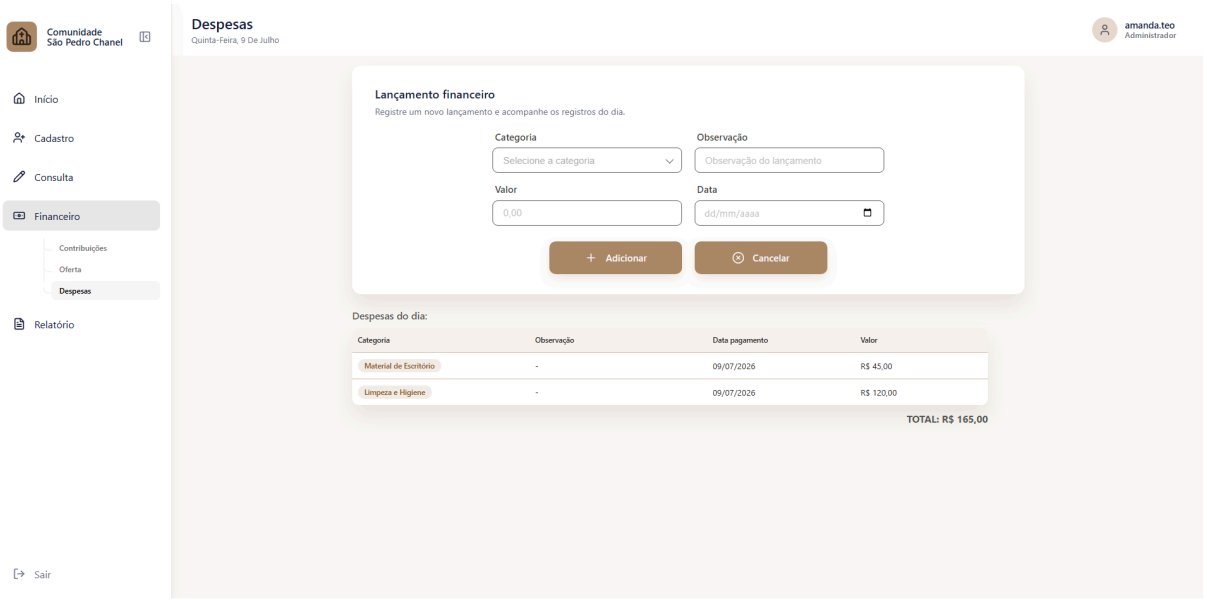
Figura 38: Tela de registro de ofertas



Fonte: Autoria própria (2026)

O formulário de registro de ofertas solicita informações como o valor total arrecadado, o tipo da celebração, a data e um campo opcional de observação. Da mesma maneira que ocorre no módulo de Contribuições, ao clicar no botão salvar, o registro da oferta é armazenado no banco de dados. Em seguida, as principais informações cadastradas são exibidas logo abaixo do formulário, em formato de lista. Ao clicar em Cancelar, os campos preenchidos são limpos.

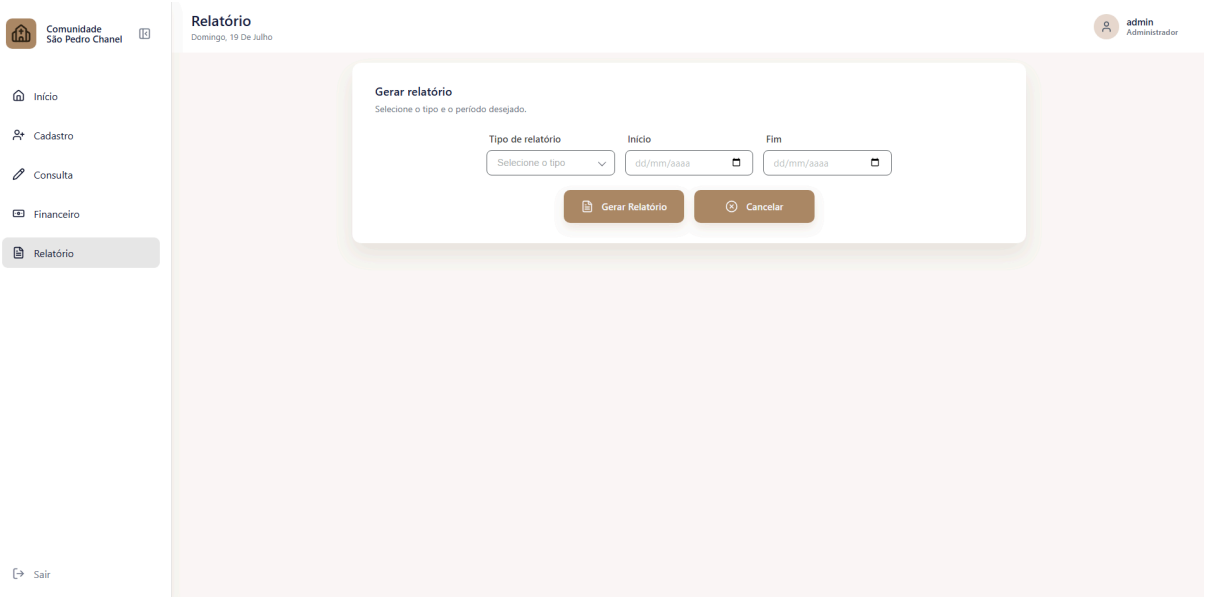
Figura 39: Tela de registro de despesas



Fonte: Autoria própria (2026)

Por fim, no formulário de registro das despesas da comunidade, são solicitadas informações como a categoria da despesa, selecionada por meio de uma combobox, o valor, a data e um campo opcional de observação. Seguindo o mesmo padrão dos formulários anteriores, ao clicar no botão salvar, o registro da despesa é armazenado no banco de dados e suas principais informações são exibidas logo abaixo do formulário, em formato de lista. Caso o usuário clique em cancelar, os campos preenchidos são limpos.

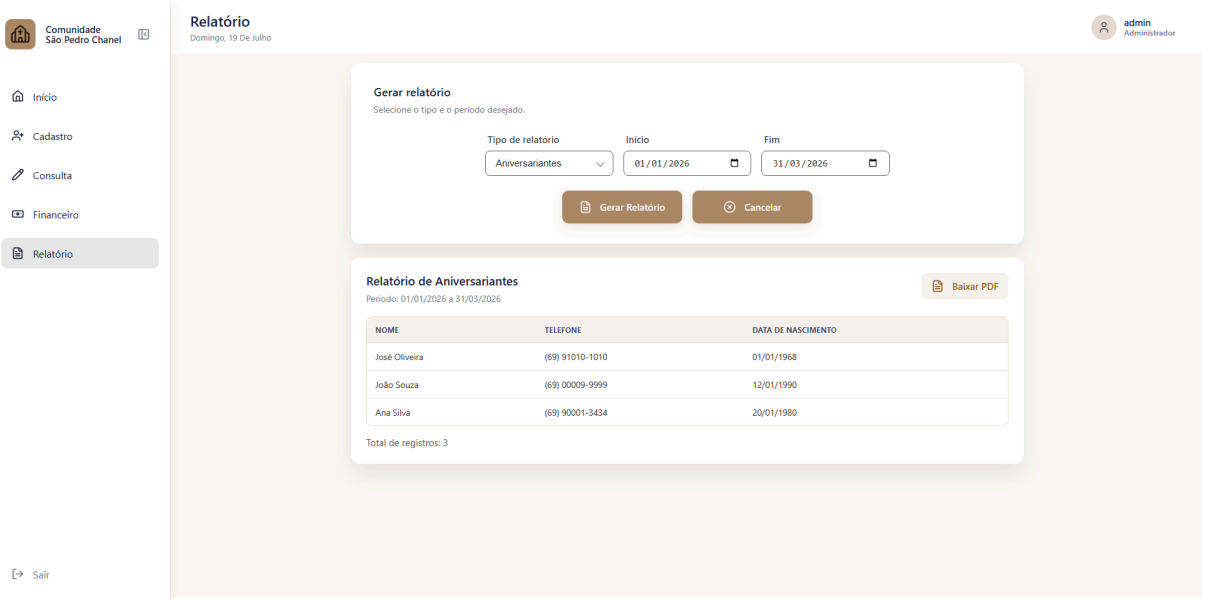
Figura 40: Tela de emissão de relatórios



Fonte: Autoria própria (2026)

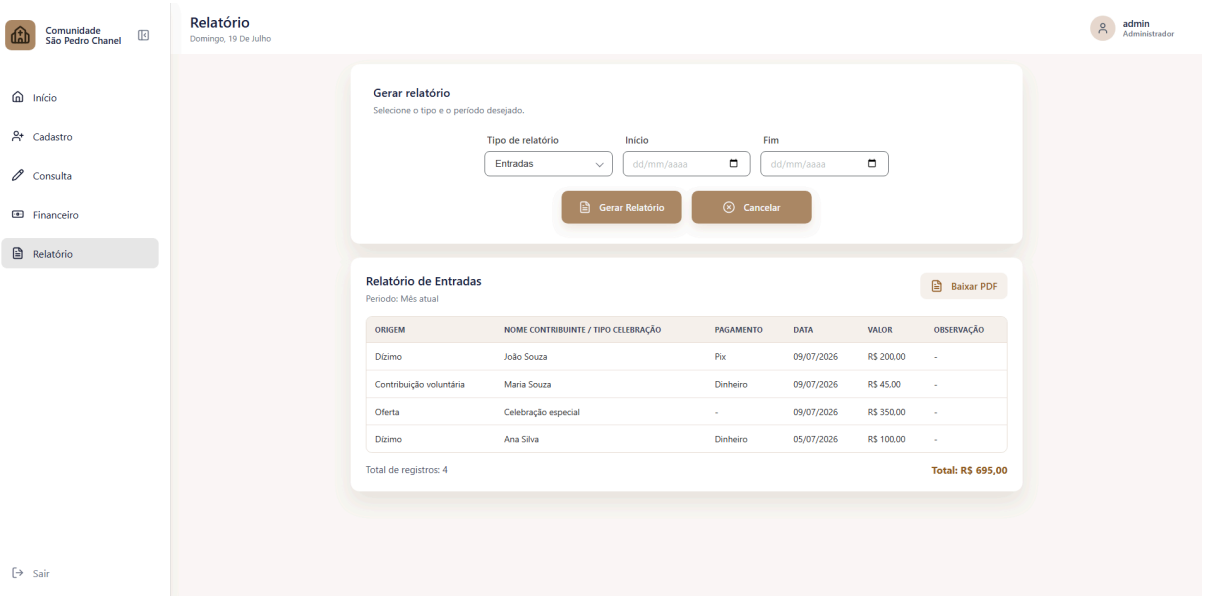
No módulo de Relatório, há um pequeno formulário destinado à definição dos filtros utilizados na consulta das informações. Nele, o usuário pode selecionar o tipo de dado que deseja visualizar, como entradas, saídas, saldo ou aniversariantes, além de informar um período específico para a busca, por meio dos campos de data inicial e data final.

Figura 41: Exibição do relatório com filtros de data gerado na tela



Fonte: Autoria própria (2026)

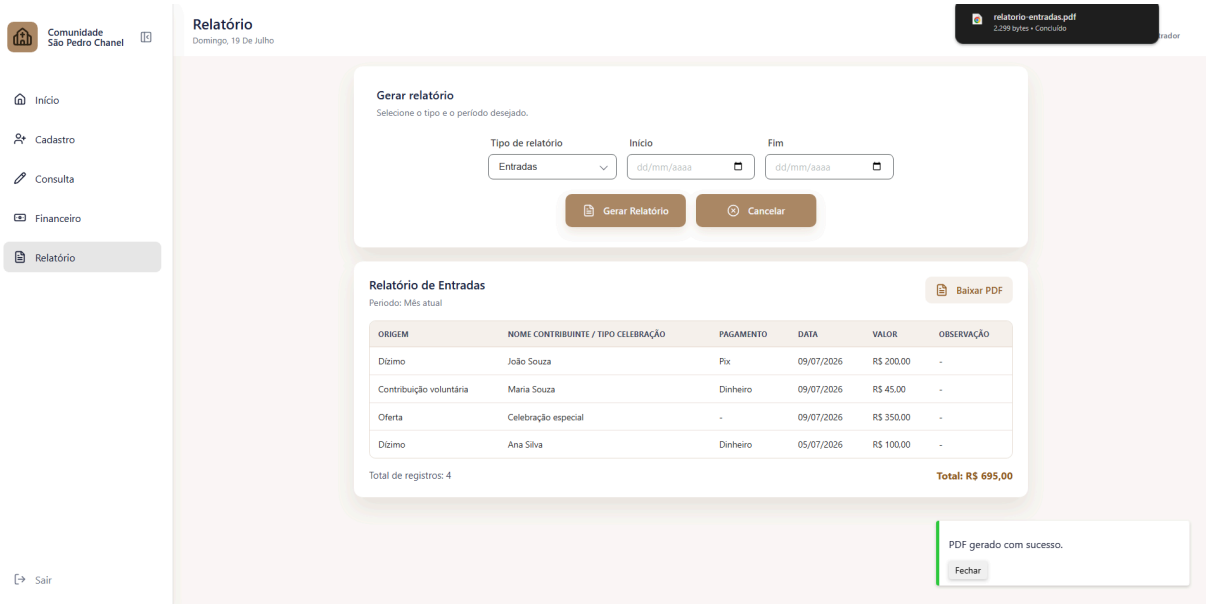
Figura 42: Exibição do relatório sem filtros de data gerado na tela



Fonte: Autoria própria (2026)

O campo referente ao tipo do relatório é obrigatório, pois define quais dados serão consultados, enquanto os campos de data são opcionais. Caso o período não seja informado, o sistema retorna automaticamente as informações correspondentes ao mês atual.

Figura 43: Mensagem de confirmação de download do relatório em PDF



Fonte: Autoria própria (2026)

Figura 44: Arquivo PDF do relatório

Fonte: Autoria própria (2026)

Por fim, após a geração do relatório em tela, o sistema disponibiliza o botão Baixar PDF. Ao clicar nessa opção, o front-end envia para a API os filtros selecionados no formulário, como tipo de relatório, data inicial e data final. A API valida essas informações, consulta os dados correspondentes no banco de dados e gera um arquivo PDF contendo cabeçalho, período consultado, tabela de registros e resumo com total de itens e valores,

quando aplicável. Após a geração do arquivo, é retornado o PDF para o navegador, que realiza automaticamente o download do documento.

5. CONCLUSÃO

A realização deste trabalho teve como objetivo principal o desenvolvimento de um sistema web voltado para a gestão de fiéis e dízimos, atendendo especificamente às necessidades da comunidade alvo. Durante o desenvolvimento da proposta, foram adicionadas outras funcionalidades com o intuito de ampliar sua utilização para diferentes áreas da gestão financeira da comunidade, como o registro de ofertas recebidas em celebrações e o controle de despesas. Dessa forma, o sistema cumpre o objetivo proposto ao oferecer uma ferramenta que contribui para maior organização, segurança e produtividade nas atividades administrativas da comunidade.

O desenvolvimento do sistema possibilitou a aplicação prática das tecnologias estudadas ao longo do trabalho. O front-end foi construído com Vue.js, enquanto o back-end foi desenvolvido com NestJS e TypeScript, utilizando o TypeORM para a comunicação com o banco de dados MySQL. Essa organização permitiu separar as responsabilidades da aplicação, mantendo a interface, as regras de negócio e o armazenamento dos dados em camadas distintas.

Durante o desenvolvimento, a API exerceu a função de conectar as ações realizadas na interface com os dados armazenados no banco. Por meio dela, foram disponibilizadas funcionalidades como autenticação de usuários, cadastro e consulta de contribuintes, registro de contribuições, ofertas e despesas, além da emissão de relatórios. A documentação das rotas foi realizada com o Swagger, facilitando a visualização e a validação dos endpoints implementados.

A partir dos resultados obtidos, observa-se que os objetivos definidos para o trabalho foram alcançados. O sistema contempla módulos de cadastro e consulta, funcionalidades voltadas ao controle financeiro e emissão de relatórios personalizados, atendendo às principais necessidades identificadas na comunidade alvo. Com isso, a aplicação contribui para tornar o registro e o acompanhamento das informações mais organizados, reduzindo a dependência de controles manuais.

Como trabalhos futuros, sugere-se a implementação da recuperação de senha por meio do e-mail cadastrado, caso o usuário esqueça suas credenciais de acesso. Recomenda-se também o aperfeiçoamento do gerenciamento das movimentações financeiras, permitindo a edição e a exclusão de contribuições, ofertas e despesas registradas incorretamente. Além disso, podem ser acrescentadas melhorias nos relatórios, novas formas de exportação e recursos adicionais no dashboard, ampliando a visualização e o acompanhamento das movimentações financeiras.

REFERÊNCIAS

ADERINOKUN, Ire. **Toast.js.** 2016. Disponível em: https://ireade.github.io/Toast.js. Acesso em 4 jul. 2026.

ANDRADE, Ana Paula de. **O que é um SGBD?**. 2021. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-um-sgbd. Acesso em: 8 fev. 2026.

AWS. **O que é uma API RESTful?**. Disponível em: https://aws.amazon.com/pt/what-is/restful-api/. Acesso em: 7 jul. 2026.

BRASIL. Instituto Brasileiro de Geografia e Estatística (IBGE). **Religiões: resultados preliminares da amostra.** 2025. Disponível em: https://agenciadenoticias.ibge.gov.br/media/com_mediaibge/arquivos/3f1708b5d315aca50d5a7d8764469c45.pdf. Acesso em: 23 fev. 2026.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. **Lei Geral de Proteção de Dados Pessoais (LGPD)**. Brasília, DF. 2018.

BRASIL. Ministério do Trabalho e Emprego. **Classificação Brasileira de Ocupações: downloads.** Brasília, DF: Ministério do Trabalho e Emprego, 2002. Disponível em: https://www.gov.br/trabalho-e-emprego/pt-br/assuntos/cbo/servicos/downloads. Acesso em: 11 jun. 2026.

CALDAS, Angela. V**ue.js: o que é, como funciona e como começar a usar esse framework JS.** 16 mai. 2024. Disponível em: https://www.alura.com.br/artigos/vue-js. Acesso em: 27 jan. 2026.

CALDEIRA, Raphael P. Banco de dados, database, **SGBD: você sabe o que é isso?**. 5 mai. 2023. Disponível em: https://academy.indicium.tech/blog/banco-de-dados-database-sgbd-voce-sabe-o-que-e-isso. Acesso em: 8 fev. 2026.

CARVALHO, Eveline BS. **The Role and Rationality of Religion: The Case of Brazil.**

Sociology International Journal, v. 1, n. 4, 3 nov. 2017. Disponível em: https://medcraveonline.com/SIJ/SIJ-01-00022.pdf. Acesso em: 23 fev. 2026.

CHART.JS. **Documentação**. 2025. Disponível em: https://www.chartjs.org/docs/latest/. Acesso em: 7 jul. 2026.

CNN. **Entenda como funciona e quando usar a metodologia Scrum**. 28 ago. 2023. Disponível em: https://www.cnnbrasil.com.br/tecnologia/scrum/. Acesso em: 09 mar. 2026.

COPES, Flavio. **The JavaScript Beginner's Handbook**. 1 mar. 2020. Disponível em: https://www.freecodecamp.org/news/the-complete-javascript-handbook-f26b2c71719c/. Acesso em: 26 jan. 2026.

DA SILVA, Rogério Oliveira; MARTINS, Bonny Rodrigues; DINIZ, Walisson Gama. **A complexibilidade da UML e seus diagramas**. Tecnologias em Projeção, [S. l.], v. 8, n. 1, p. 86–99, 2017. Disponível em: https://www.projecaociencia.com.br/index.php/Projecao4/article/view/825/727. Acesso em: 26 jan. 2026.

DEVMEDIA. **Introdução ao TypeScript**. 2016. Disponível em: https://www.devmedia.com.br/introducao-ao-typescript/36729#TypeScript. Acesso em: 09 mar. 2026.

DNC. **MySQL Workbench: entenda o que é e como funciona**. 14 abr. 2025. Disponível em: https://www.escoladnc.com.br/blog/introducao-ao-mysql-workbench. Acesso em: 8 fev. 2026.

FERREIRA, Ana Cristina Martins. **Refinamento de diagramas de classes: análise e verificação**. FCT, 2010.

FIGMA. **O que é o Figma?**. 2025. Disponível em: https://help.figma.com/hc/pt-br/articles/14563969806359-O-que-%C3%A9-o-Figma. Acesso em: 26 jan. 2026.

GEEKHUNTER. **Vue JS: o que é, como funciona e vantagens.** 13 abr. 2020. Disponível em: https://blog.geekhunter.com.br/vue-js-so-vejo-vantagens-e-voce/. Acesso em: 27 jan. 2026.

GOULART, Reane Franco. **UML e a Ferramenta Astah**. 2013. Disponível em: https://profareane.wordpress.com/wp-content/uploads/2013/09/aula-3-uml-e-astah.pdf. Acesso em: 25 jan. 2026.

HIRUMA, Elton Keishi; GABY, Eliel dos Santos. **Gestão financeira no ambiente eclesiástico.** Revista Teologia e Espiritualidade, v. 8, n. 16, Curitiba, p. 74, 2021.

LENCINA, Walter. **O que é um framework e para que serve?**. 26 ago. 2023. Disponível em: https://ebaconline.com.br/blog/framework-seo. Acesso em: 27 jan. 2026.

LUCIDE. **What is Lucide?**. Disponível em: https://lucide.dev/guide/. Acesso em: 7 jul. 2026.

MIRANDA, Luiz. **Framework: o que é, para o que serve e exemplos**. 31 jan. 2025. Disponível em: https://querobolsa.com.br/revista/framework-o-que-e. Acesso em: 27 jan. 2026.

NESTJS. **Documentação**. 2017. Disponível em: https://docs.nestjs.com/. Acesso em: 09 mar. 2026.

NETO, Moacyr Franco. **Tutorial da ferramenta de modelagem ASTAH (Versão resumida)**. 2017. Disponível em: http://codilon.qlix.com.br/bd/bdaula09.pdf. Acesso em: 25 jan. 2026.

OLIVEIRA, Emilene Lima de; MELLO, Paulo Augusto M. S. **Informatização dos serviços paroquiais arquidiocesanos**. Revista Tecnologias em Projeção, v. 2, n. 2, p. 13–17, dez. 2011.

OLIVEIRA, Natanael. **Sistemas de Gerenciamento de Banco de Dados**. 7 set. 2024. Disponível em: https://www.dio.me/articles/sistemas-de-gerenciamento-de-banco-de-dados. Acesso em: 8 fev. 2026.

ORACLE. **O que é um banco de dados?**. 24 nov. 2020. Disponível em: https://www.oracle.com/br/database/what-is-database/. Acesso em: 8 fev. 2026.

PM3. **Figma: o que é, por que usar e principais funcionalidades**. 19 fev. 2024. Disponível em: https://pm3.com.br/blog/figma. Acesso em: 26 jan. 2026.

REBELLO, Mariana. **Javascript: o que é, como surgiu e onde utilizar**. 10 jul. 2022. Disponível em: https://www.resilia.com.br/blog/javascript-o-que-e-como-surgiu-e-onde-utilizar/. Acesso em: 26 jan. 2026.

SANTOS, Adriel Faria dos. **Documentação de Software para usuários desenvolvedores: uma avaliação de bibliotecas ORM populares na plataforma Node.** 2025. Trabalho de Conclusão de Curso - Universidade Federal do Rio Grande do Norte, 2025.

SANTOS, Nilson Alves dos. **Desenvolvimento e validação de software de auxílio à gestão de instituições evangélicas: uma prática à luz da Igreja Salvação em Cristo de Porto Nacional - TO**. 2018. Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia do Tocantins, Campus Porto Nacional, 2018.

SANTOS, Thiago do. **TypeORM: O ORM que Você Precisa Conhecer para Trabalhar com Node.js e TypeScript.** 11 jun. 2024. Disponível em: https://dev.to/iamthiago/typeorm-o-orm-que-voce-precisa-conhecer-para-trabalhar-com-nodejs-e-typescript-21m2. Acesso em: 7 jul. 2026.

SCHWABER, Ken; SUTHERLAND, Jeff. **O guia do Scrum**. nov. 2020. Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf. Acesso em: 09 mar. 2026.

SOARES, Kelvin Diego da Silva. **Vamos falar um pouco do poderoso framework Nestjs**. 25 abr. 2023. Disponível em: https://pt.linkedin.com/pulse/vamos-falar-um-pouco-do-poderoso-framework-nestjs-da-silva-soares. Acesso em: 09 mar. 2026.

TICOOP BRASIL. **O que é Framework?**. Disponível em: https://ticoopbrasil.coop.br/dicionario-ti/o-que-e-framework/. Acesso em: 27 jan. 2026.

VUE.JS. **Components Basics**. 2026. Disponível em: https://vuejs.org/guide/essentials/component-basics. Acesso em: 18 jul. 2026.

VUE.JS. **Composables**. 2026. Disponível em: https://vuejs.org/guide/reusability/composables. Acesso em: 18 jul. 2026.

VUE.JS. **Composition API FAQ**. 2024. Disponível em: https://vuejs.org/guide/extras/composition-api-faq.html#what-is-composition-api. Acesso em: 18 jul. 2026.

APÊNDICE A: Projeto do Software

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA D RONDÔNIA – IFRO CAMPUS JI-PARANÁ

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

AMANDA TEODORO CUNHA

SISTEMA WEB PARA GERENCIAMENTO DE DÍZIMO E CONTRIBUIÇÕES NAS COMUNIDADES RELIGIOSAS

Ji-Paraná, 2026

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. DOCUMENTO DE ESPECIFICAÇÃO DO PROJETO

1.1. Produto

Trata-se do desenvolvimento de um sistema web para o gerenciamento de dizimistas e contribuições nas comunidades religiosas, com o objetivo de otimizar o controle das contribuições e informações dos membros. O sistema permite realizar o cadastro de dizimistas, registro e acompanhamento de doações, além da emissão de relatórios para auxiliar na administração financeira da comunidade. Sendo desenvolvido com as tecnologias Vue.js e Nest.js, o sistema é acessível por meio de qualquer navegador, necessitando apenas de conexão à internet, o que garante praticidade e facilidade de uso tanto para os administradores quanto para os usuários autorizados.

1.2. Aspectos Técnicos do Produto

1.2.1. Tecnologias Front End

O desenvolvimento do front-end da aplicação foi realizado utilizando a linguagem de programação JavaScript em conjunto com o framework Vue.js. Segundo Copes (2023), o JavaScript é considerado uma das linguagens de programação mais populares do mundo, usado principalmente para sites e aplicações web. Ao longo dos anos, o JavaScript evoluiu significativamente, ampliando suas funcionalidades e consolidando-se como uma linguagem executada no lado do cliente (client-side) (REBELLO, 2022). As principais vantagens da utilização do JavaScript estão relacionadas à sua simplicidade, uma vez que possui uma estrutura acessível e de fácil compreensão e implementação. Além disso, destaca-se pela velocidade, pois executa scripts diretamente no navegador do usuário, sem a necessidade de conexão prévia com o servidor ou uso de compiladores. A versatilidade também é um ponto relevante, já que o JavaScript é compatível e pode ser integrado com outras linguagens, como PHP e Java (REBELLO, 2022). Destaca-se também a sua ampla popularidade, que resulta em uma vasta quantidade de bibliotecas, frameworks, fóruns e materiais de apoio, atendendo tanto iniciantes quanto desenvolvedores mais experientes. Por fim, a linguagem contribui para a redução da carga do servidor, visto que validações e algumas solicitações podem ser realizadas diretamente no lado do cliente (REBELLO, 2022).

O Vue.js é um framework JavaScript voltado para o desenvolvimento de interfaces, sendo amplamente utilizado na construção de aplicações web modernas. O framework possui uma abordagem progressiva, o que permite que seja incorporado gradualmente a projetos que

já existem ou utilizados do zero de forma completa na criação de aplicações (CALDAS, 2024). O Vue.js se baseia no uso de tecnologias padrão da web, como HTML, CSS e JavaScript, oferecendo um modelo de programação com componentes, o que contribui para a organização e reutilização de código. De acordo com Caldas (2024), uma de suas principais características é a sua reatividade, na qual permite a atualização automática da interface sempre que ocorra alguma alteração no código. Esse mecanismo facilita o gerenciamento do estado da aplicação, tornando o desenvolvimento mais eficiente e menos propensos a erros.

1.2.2. Tecnologias Back End

O desenvolvimento do back-end foi realizado utilizando o framework Nest.js juntamente com a linguagem de programação TypeScript. O TypeScript é uma linguagem de programação desenvolvida pela Microsoft que funciona como um superset do JavaScript, ou seja, uma extensão da linguagem que adiciona novos recursos sem deixar de ser compatível com o JavaScript tradicional. Entre as principais funcionalidades adicionadas estão a tipagem de dados e melhores mecanismos para utilização da programação orientada a objetos, o que contribui para a construção de aplicações com uma arquitetura mais organizada e estruturada (DEVMEDIA, 2016). O Nest.js é um framework utilizado para o desenvolvimento de aplicações do lado do servidor em Node.js, projetado para permitir a criação de sistemas eficientes e escaláveis. Ele utiliza JavaScript progressivo e oferece suporte completo à linguagem TypeScript, embora também permita o desenvolvimento utilizando apenas o JavaScript (NESTJS, 2017).

Para o armazenamento e gerenciamento dos dados da aplicação foi utilizado o MySQL, um SGBD relacional que permite organizar as informações em tabelas estruturadas e relacionadas entre si. Segundo Caldeira (2023), um Banco de Dados é um dispositivo de armazenamento para conjunto de informações que precisam ser armazenadas e controladas por alguém ou alguma instituição. Para esses dados serem gerenciados é necessário o uso de SGBDs, os quais possibilitam a criação e o gerenciamento de usuários, bem como a consulta, alteração e exclusão de registros, entre outras funcionalidades (CALDEIRA, 2023).

Para auxiliar no processo de modelagem e gerenciamento do banco de dados foi utilizado o MySQL Workbench, uma ferramenta visual que permite a criação de modelos de banco de dados por meio de interface gráfica, facilitando o planejamento da estrutura das tabelas e seus relacionamentos antes da implementação no sistema.

1.3. Equipe de Desenvolvimento

A equipe de desenvolvimento é composta pelos seguintes papéis e atribuições, de acordo com a metodologia ágil Scrum estabelecida para este projeto:

Tabela 1 - Equipe de desenvolvimento

ID	Nome	Função	Atribuições
01	Rosangela Teodoro da Silva	Product Owner	Representar as necessidades dos usuários e orientar o desenvolvimento do produto.
02	Reinaldo Lima Pereira	Scrum Master	Definir o que é preciso ser entregue e o que é prioridade.
03	Amanda Teodoro Cunha	Desenvolvedora	Desenvolver o sistema.

Fonte: Autoria própria (2026).

1.4. Metodologia de Desenvolvimento

Para alcançar os objetivos propostos, inicialmente buscou-se compreender as principais necessidades relacionadas ao controle de contribuições financeiras em ambientes religiosos, identificando dificuldades presentes em processos realizados de forma manual, como o registro de dízimos, o controle de contribuições e contribuintes, bem como a geração de relatórios financeiros. A partir dessa análise, foi realizado o levantamento dos requisitos do sistema, possibilitando a definição das principais funcionalidades que irão compor a aplicação.

Após a etapa de levantamento de requisitos, foi desenvolvido um protótipo das interfaces do sistema por meio da plataforma Figma, com o objetivo de proporcionar uma melhor visualização da aplicação. Nesse protótipo foram representadas as principais telas e módulos do sistema, incluindo telas de cadastro, consultas e emissão de relatórios.

Com base nos requisitos identificados, foram elaborados os diagramas de modelagem do sistema. Para essa etapa foi utilizada a ferramenta Astah, com a finalidade de desenvolver

o diagrama de caso de uso, permitindo a visualização dos usuários do sistema e das interações disponíveis para cada um deles. Posteriormente, foi realizada a elaboração dos Casos de Uso Expandidos, com o objetivo de descrever de forma detalhada as funcionalidades da aplicação.

Em seguida, foi realizada a modelagem do banco de dados, responsável por armazenar as informações dos módulos do sistema. Para isso, foi utilizado o MySQL como Sistema Gerenciador de Banco de Dados (SGBD), e o MySQL Workbench para a construção do modelo lógico.

Por fim, foi realizada a implementação do sistema. Durante o desenvolvimento, o back-end foi executado em um servidor local, utilizando o ambiente Node.js e o framework NestJS. Desenvolvido em TypeScript, o back-end é responsável pelo processamento das regras de negócio, pela API e pela comunicação com o banco de dados MySQL por meio do TypeORM, ferramenta utilizada para mapear as entidades da aplicação e facilitar as operações realizadas no banco de dados. O front-end foi desenvolvido em JavaScript com o framework Vue.js e realiza a comunicação com a API por meio de requisições HTTP. Para documentar e auxiliar na validação dos endpoints disponibilizados pela API, utilizou-se o Swagger.

Para o gerenciamento e acompanhamento do desenvolvimento do sistema foi utilizada a metodologia ágil Scrum. Entre seus artefatos, foi empregado o Product Backlog, responsável por reunir todas as funcionalidades e tarefas do projeto, e o Sprint Backlog, utilizado para organizar e definir as atividades que serão realizadas em cada ciclo de desenvolvimento.

1.4.1. Product Backlog

Neste tópico serão documentados todos os itens do Product Backlog criados ao longo do desenvolvimento do projeto.

Tabela 2 - Product Backlog

Item	Descrição	Estimativa de Esforço	Prioridade
1	Realizar o levantamento de informações e análise inicial do sistema	5h	100
2	Elaborar esboço do sistema e suas funcionalidade	5h	100
3	Elaborar o Documento de Requisitos	48h	100
4	Criar o design do protótipo do sistema	64h	90
5	Modelar o Diagrama de Caso de Uso e escrever os Casos de Uso Expandidos	45h	80
6	Modelar o Banco de Dados Lógico	20h	80
7	Revisar e Organizar o Projeto de Software	15h	70
8	Configurar ambiente de desenvolvimento	4h	100
9	Desenvolver tela inicial	15h	100
10	Desenvolver tela de login	10h	100
11	Desenvolver tela de perfil do usuário	10h	80
12	Desenvolver telas de cadastro	40h	100
13	Desenvolver telas de consulta	40h	100
14	Desenvolver telas do financeiro	60h	100
15	Desenvolver tela de relatório	30h	100
16	Revisar e Organizar a interface geral do sistema	30h	100

17	Revisar e Organizar o projeto de Software	20h	70

Fonte: Autoria própria (2026).

1.4.2. Sprint Backlog

Neste tópico serão documentados todos os itens das Sprints Backlog criados ao longo do desenvolvimento do projeto.

Tabela 3 - Sprint Backlog

Item do Product Backlog	**Nº Sprint e Período**	**Descrição da Tarefa**	**Responsável**	**Esforço**
1 e 2	Sprint 1: 08/05/2025 a 08/06/2025	1.1 Realizar o levantamento de informações, identificando o problema que o sistema pretende resolver e documentar	Amanda Teodoro	5h
		1.2 Criar esboço das funcionalidades do sistema	Amanda Teodoro	5h
3	Sprint 2: 09/06/2025 a 07/07/2025	2.1. Documento de Requisitos: Fazer Login	Amanda Teodoro	6h
		2.2. Documento de Requisitos: Cadastrar Usuário	Amanda Teodoro	6h
		2.3. Documento de Requisitos: Consultar Usuário	Amanda Teodoro	6h
		2.4. Documento de Requisitos: Cadastrar Contribuinte	Amanda Teodoro	6h
		2.5. Documento de Requisitos: Consultar Contribuinte	Amanda Teodoro	6h
		2.6. Documento de Requisitos: Registrar Contribuições	Amanda Teodoro	6h

		2.7. Documento de Requisitos: Registrar Despesas	Amanda Teodoro	6h
		2.8. Documento de Requisitos: Emitir Relatórios	Amanda Teodoro	6h
4, 5 e 6	Sprint 3: 07/07/2025 a 10/08/2025	3.1. Protótipo do sistema: Fazer Login	Amanda Teodoro	8h
		3.2. Protótipo do sistema: Cadastrar Usuário	Amanda Teodoro	8h
		3.3. Protótipo do sistema: Consultar Usuário	Amanda Teodoro	8h
		3.4. Protótipo do sistema: Cadastrar Contribuinte	Amanda Teodoro	8h
		3.5. Protótipo do sistema: Consultar Contribuinte	Amanda Teodoro	8h
		3.6. Protótipo do sistema: Registrar Contribuições	Amanda Teodoro	8h
		3.7. Protótipo do sistema: Registrar Despesas	Amanda Teodoro	8h
		3.8. Protótipo do sistema: Emitir Relatórios	Amanda Teodoro	8h
		3.9 Modelar Diagrama de Caso de Uso	Amanda Teodoro	5h
		3.10 Caso de Uso Expandido: Fazer Login	Amanda Teodoro	5h

		3.11 Caso de Uso Expandido: Cadastrar Usuário	Amanda Teodoro	5h
		3.12 Caso de Uso Expandido: Consultar Usuário	Amanda Teodoro	5h
		3.13 Caso de Uso Expandido: Cadastrar Contribuinte	Amanda Teodoro	5h
		3.14 Caso de Uso Expandido: Consultar Contribuinte	Amanda Teodoro	5h
		3.15 Caso de Uso Expandido: Registrar Contribuições	Amanda Teodoro	5h
		3.16 Caso de Uso Expandido: Registrar Despesas	Amanda Teodoro	5h
		3.17 Caso de Uso Expandido: Emitir Relatórios	Amanda Teodoro	5h
		3.18 Modelo Lógico do Banco de Dados	Amanda Teodoro	20h
7	Sprint 4: 20/01/2026 a 10/03/2026	4.1 Revisão e organização do Projeto de Software	Amanda Teodoro	15h
8	Sprint 5: 20/04/2026 a 26/04/2026	5.1 Criação das pastas para o back-end e o front-end	Amanda Teodoro	1h
		5.2 Instalação e configuração das dependências necessárias para o desenvolvimento, como o Vue.js e o NestJS e demais bibliotecas	Amanda Teodoro	3h

9, 10 e 11	Sprint 6: 27/04/2026 a 04/05/2026	6.1 Desenvolver a tela de início e dashboard	Amanda Teodoro	15h
		6.2 Desenvolver a tela de login	Amanda Teodoro	10h
		6.2 Desenvolver tela de perfil do usuário	Amanda Teodoro	10h
12	Sprint 7: 05/05/2026 a 24/05/2026	7.1 Desenvolver a tela de cadastro de usuário	Amanda Teodoro	20h
		7.2 Desenvolver a tela de cadastro de contribuinte	Amanda Teodoro	20h
13	Sprint 8: 25/05/2026 a 12/06/2026	8.1 Desenvolver tela de consulta de usuário	Amanda Teodoro	20h
		8.2 Desenvolver tela de consulta de contribuinte	Amanda Teodoro	20h
14	Sprint 9: 13/06/2026 a 02/07/2026	9.1 Desenvolver tela de registro de contribuições	Amanda Teodoro	20h
		9.1 Desenvolver tela de registro de oferta	Amanda Teodoro	20h
		9.1 Desenvolver tela de registro de despesas	Amanda Teodoro	20h
15	Sprint 10: 03/07/2026 a 10/07/2026	10.1 Desenvolver tela de emissão de relatórios	Amanda Teodoro	15h
		10.2 Implementar geração de relatórios em formato PDF	Amanda Teodoro	15h

16 e 17	Sprint 11: 11/07/2026 a 22/07/2026	11.1 Revisar e organizar o código do back-end e banco de dados	Amanda Teodoro	15h
		11.2 Revisar e organizar o código do front-end	Amanda Teodoro	15h
		11.3 Revisar e organizar o Projeto de Software	Amanda Teodoro	20h

Fonte: Autoria própria (2026).

2. DOCUMENTO DE REQUISITOS

Após a fase de levantamento de requisitos, foram identificados e coletados os seguintes requisitos para o software.

2.1. Fazer Login

Requisito Funcional		
Nome:	Fazer Login	**Código:**RF01
Descrição:	O sistema deve permitir que o usuário realize autenticação por meio do fornecimento do nome de usuário e senha válidos. Após a validação das credenciais, o usuário deve ser direcionado à área restrita do sistema.	
Estimativa de Esforço: 10h		**Prioridade:** Alta

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
RNF01	O sistema deve validar os dados de entrada (nome de usuário e senha)	Segurança
RNF02	Senhas serão criptografadas.	Segurança
RNF03	O sistema deve restringir o acesso a dados com base em funções de usuário.	Segurança
RNF04	A autenticação deve ocorrer em até 3 segundos após o envio das credenciais.	Desempenho
RNF05	O Sistema terá a opção de mostrar a senha para o usuário ver o que digitou.	Usabilidade
RNF06	O sistema deve ser fácil de usar e entender.	Usabilidade
RNF07	A função apresentará os campos de usuário e senha, botões de acesso e visualizar senha.	Especificação

2.2. Cadastrar Usuário

Requisito Funcional		
Nome:	Cadastrar Usuário	**Código:**RF02
Descrição:	O sistema deve permitir o cadastro de novos usuários, preenchendo informações como nome completo, nome de usuário, senha, e-mail, telefone e nível de acesso. Os dados devem ser validados antes do salvamento e armazenados no banco de dados.	
Estimativa de Esforço: 20h		**Prioridade:** Alta

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
RNF01	O sistema deve ser capaz de lidar com cadastros simultâneos sem perda de desempenho.	Desempenho
RNF02	O sistema deve cadastrar um novo usuário em um tempo aceitável, de pelo menos alguns segundos.	Desempenho
RNF03	A função deve registrar os seguintes dados: nome completo, nome de usuário, senha, e-mail, telefone e nível de acesso.	Especificação
RNF04	Informações sensíveis do usuário devem ser protegidas contra acessos não autorizados.	Segurança
RNF05	Senhas serão criptografadas.	Segurança
RNF06	O sistema deve garantir que apenas usuários autenticados e autorizados possam cadastrar novos usuários.	Segurança
RNF07	A interface para cadastrar um novo usuário deve ser intuitiva e fácil de usar, mesmo para usuários leigos.	Usabilidade
RNF08	O sistema deve fornecer feedback claro ao usuário durante o processo de cadastro, informando se a operação foi bem sucedida ou se ocorreu algum erro.	Usabilidade
RNF09	O sistema deve estar disponível para cadastrar novos usuários a qualquer momento, desde que o dispositivo esteja conectado à internet.	Disponibilidade
RNF10	O sistema irá validar os tipos de dados inseridos.	Confiabilidade.

2.3. Consultar Usuário

Requisito Funcional		
Nome:	Consultar Usuário	**Código:**RF03
Descrição:	O sistema deve permitir ao usuário visualizar uma lista de usuários cadastrados, mostrando informações principais para fins de identificação como nome completo, telefone, data de nascimento e o nível de acesso. A função terá opção de busca por nome. A partir dessa listagem, o sistema deve permitir a edição e a exclusão dos dados do usuário selecionado.	
Estimativa de Esforço: 20h		**Prioridade:** Alta

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
RNF01	O sistema deve retornar os dados da consulta em até 5 segundos.	Desempenho
RNF02	A função terá uma barra de pesquisa para o usuário encontrar o usuário desejado, o sistema deve trazer os resultados em no máximo 5 segundos.	Desempenho
RNF03	A interface da tela de consulta deve ser intuitiva, com campos de busca e filtros acessíveis.	Interface
RNF04	O sistema deve ordenar os resultados por nome, telefone, data de nascimento e nível de acesso.	Interface
RNF05	Apenas usuários autenticados e com permissão adequada poderão acessar a funcionalidade de consulta de usuários.	Segurança
RNF06	Dados como, senha, nome de usuário e e-mail não serão exibidos na consulta	Segurança
RNF07	Em caso de falha na consulta, uma mensagem deve ser exibida ao usuário, sugerindo tentar novamente mais tarde.	Confiabilidade
RNF08	A função terá botões para editar, que irá levar o usuário ao formulário com os dados do usuário selecionado para caso haja uma informação que deseja alterar, e para excluir.	Usabilidade

2.4. Cadastrar Contribuinte

Requisito Funcional		
Nome:	Cadastrar Contribuinte	**Código:**RF04
Descrição:	O sistema deve permitir o cadastro de novos contribuintes, preenchendo informações como nome completo, endereço, telefone, data de nascimento, e profissão. O formulário terá um campo a ser marcado caso o contribuinte seja casado, caso sim, o sistema irá apresentar outro formulário abaixo requisitando informações sobre o cônjuge como nome completo, data de nascimento e telefone. Os dados devem ser validados antes do salvamento e armazenados no banco de dados.	
Estimativa de Esforço: 15h		**Prioridade:** Alta

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
RNF01	O sistema deve ser capaz de lidar com cadastros simultâneos sem perda de desempenho.	Desempenho
RNF02	O sistema deve cadastrar um novo contribuinte em um tempo aceitável, de pelo menos alguns segundos.	Desempenho
RNF03	A função deve registrar os seguintes dados do contribuinte: nome completo, endereço, telefone, data de nascimento e profissão	Especificação
RNF04	O sistema deve garantir que apenas usuários autenticados e autorizados possam cadastrar novos contribuintes.	Segurança
RNF05	A interface para cadastrar um novo contribuinte deve ser intuitiva e fácil de usar, mesmo para usuários leigos.	Usabilidade
RNF06	O sistema deve fornecer feedback claro ao usuário durante o processo de cadastro, informando se a operação foi bem sucedida ou se ocorreu algum erro.	Usabilidade
RNF07	Se o campo “Casado(a)” for marcado, o formulário do cônjuge deve aparecer automaticamente de forma clara e acessível.	Usabilidade
RNF08	A função deve registrar os seguintes dados do cônjuge: nome completo, data de nascimento e telefone.	Especificação

RNF09	Informações sensíveis do contribuinte e cônjuge devem ser protegidas contra acessos não autorizados.	Segurança

2.5. Consultar Contribuinte

Requisito Funcional		
Nome:	Consultar Contribuinte	**Código:**RF05
Descrição:	O sistema deve permitir ao usuário visualizar uma lista de contribuintes cadastrados, mostrando informações principais para fins de identificação como o nome completo, telefone e a data de nascimento. A função terá opção de busca por nome. A partir dessa listagem, o sistema deve permitir a edição e a exclusão dos dados do contribuinte selecionado.	
Estimativa de Esforço: 15h		**Prioridade:** Alta

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
RNF01	O sistema deve retornar os dados da consulta em até 5 segundos.	Desempenho
RNF02	A função terá uma barra de pesquisa para o usuário encontrar o contribuinte desejado, o sistema deve trazer os resultados em no máximo 5 segundos.	Desempenho
RNF03	A interface da tela de consulta deve ser intuitiva, com campos de busca e filtros acessíveis.	Interface
RNF04	O sistema deve ordenar os resultados por nome, telefone e data de nascimento.	Interface
RNF05	Apenas usuários autenticados e com permissão adequada poderão acessar a funcionalidade de consulta de contribuintes.	Segurança
RNF06	Em caso de falha na consulta, uma mensagem deve ser exibida ao usuário, sugerindo tentar novamente mais tarde.	Confiabilidade
RNF07	A função terá botões para editar, que irá levar o usuário ao formulário com os dados do contribuinte selecionado para caso haja uma informação que deseja alterar, e para excluir.	Usabilidade

2.6. Registrar Contribuições

Requisito Funcional		
Nome:	Registrar Contribuições	**Código:**RF06
Descrição:	O sistema deve permitir que o usuário registre contribuições financeiras realizadas por contribuintes, informando o tipo, o valor, a forma de pagamento, a data, e associando o lançamento ao contribuinte correspondente. Assim que a contribuição for registrada, suas informações serão mostradas em uma tabela logo abaixo do formulário.	
Estimativa de Esforço: 20h		**Prioridade:** Alta

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
RNF01	O sistema deve ser capaz de lidar com cadastros simultâneos sem perda de desempenho.	Desempenho
RNF02	O sistema deve registrar a contribuição em um tempo aceitável, de pelo menos alguns segundos.	Desempenho
RNF03	A função deve registrar os seguintes dados: Contribuinte, tipo de contribuição, valor, forma de pagamento e a data do pagamento.	Especificação
RNF04	A função terá os botões de Salvar e Cancelar, para registrar as informações no Banco de Dados ou cancelar a ação caso seja necessário.	Especificação
RNF05	O sistema deve fornecer feedback claro ao usuário durante o processo de cadastro, informando se a operação foi bem sucedida ou se ocorreu algum erro.	Usabilidade
RNF06	A interface para registrar uma nova contribuição deve ser intuitiva e fácil de usar, mesmo para usuários leigos.	Usabilidade
RNF07	O sistema deve garantir que apenas usuários autenticados e autorizados possam registrar novas contribuições.	Segurança
RNF08	O sistema deve estar disponível para registrar novas contribuições a qualquer momento, desde que o dispositivo esteja conectado à internet.	Disponibilidade

2.7. Registrar Oferta

Requisito Funcional		
Nome:	Registrar Oferta	**Código:**RF07
Descrição:	O sistema deve permitir que o usuário registre as ofertas realizadas nas celebrações da comunidade, informando o tipo da celebração, o valor e a data. Assim que a oferta for registrada, suas informações serão mostradas em uma tabela logo abaixo do formulário.	
Estimativa de Esforço: 15h		**Prioridade:** Alta

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
RNF01	O sistema deve ser capaz de lidar com cadastros simultâneos sem perda de desempenho.	Desempenho
RNF02	O sistema deve registrar a oferta em um tempo aceitável, de pelo menos alguns segundos.	Desempenho
RNF03	A função deve registrar os seguintes dados: tipo de celebração, valor e data.	Especificação
RNF04	A função terá os botões de Salvar e Cancelar, para registrar as informações no Banco de Dados ou cancelar a ação caso seja necessário.	Especificação
RNF05	O sistema deve fornecer feedback claro ao usuário durante o processo de cadastro, informando se a operação foi bem sucedida ou se ocorreu algum erro.	Usabilidade
RNF06	A interface para registrar uma nova contribuição deve ser intuitiva e fácil de usar, mesmo para usuários leigos.	Usabilidade
RNF07	O sistema deve garantir que apenas usuários autenticados e autorizados possam registrar novas ofertas.	Segurança
RNF08	O sistema deve estar disponível para registrar novas ofertas a qualquer momento, desde que o dispositivo esteja conectado à internet.	Disponibilidade

2.8. Registrar Despesas

Requisito Funcional		
Nome:	Registrar Despesas	**Código:**RF08
Descrição:	O sistema deve permitir que o usuário registre despesas financeiras informando a categoria, uma descrição, o valor, e a data da despesa, associando-as ao período correspondente. Assim que a despesa for registrada, suas informações serão mostradas em uma tabela logo abaixo do formulário.	
Estimativa de Esforço: 15h		**Prioridade:** Alta

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
RNF01	O sistema deve ser capaz de lidar com cadastros simultâneos sem perda de desempenho.	Desempenho
RNF02	O sistema deve registrar a despesa em um tempo aceitável, de pelo menos alguns segundos.	Desempenho
RNF03	A função deve registrar os seguintes dados: categoria, descrição, valor e data.	Especificação
RNF04	A função terá os botões de Salvar e Cancelar, para registrar as informações no Banco de Dados ou cancelar a ação caso seja necessário.	Especificação
RNF05	O sistema deve fornecer feedback claro ao usuário durante o processo de cadastro, informando se a operação foi bem sucedida ou se ocorreu algum erro.	Usabilidade
RNF06	A interface para registrar uma nova despesa deve ser intuitiva e fácil de usar, mesmo para usuários leigos.	Usabilidade
RNF07	O sistema deve garantir que apenas usuários autenticados e autorizados possam registrar novas despesas.	Segurança
RNF08	O sistema deve estar disponível para registrar novas despesas a qualquer momento, desde que o dispositivo esteja conectado à internet.	Disponibilidade

2.9. Emitir Relatórios

Requisito Funcional		
Nome:	Emitir Relatórios	**Código:**RF09
Descrição:	O sistema deve permitir que o usuário gere relatórios financeiros sobre as entradas, saídas e saldo total, além de relatórios de aniversariantes. Os relatórios poderão ser filtrados por período, utilizando data inicial e data final, sendo possível visualizar os dados em tela ou exportar em formato de PDF.	
Estimativa de Esforço: 20h		**Prioridade:** Alta

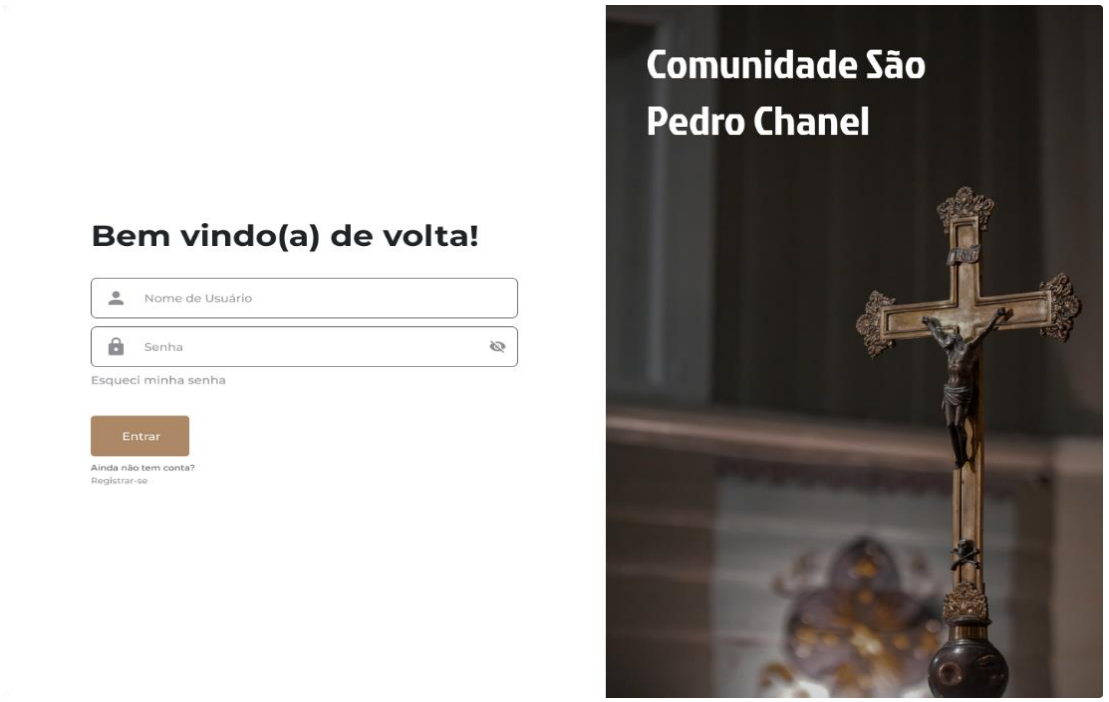
Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
RNF01	O sistema só será capaz de emitir relatórios de dados que estão ativos no sistema.	Confiabilidade
RNF02	A interface de emissão de relatórios deve ser intuitiva, contendo uma combo box para seleção do tipo de relatório e campos de data.	Usabilidade
RNF03	Ao clicar no botão Emitir Relatório, o sistema deve executar a função em até 5 segundos.	Desempenho
RNF04	O sistema deve exibir mensagens claras ao usuário em caso de ausência de registros para o período selecionado.	Usabilidade
RNF05	O sistema deve permitir a visualização dos relatórios diretamente em tela ou emissão em pdf, dependendo do que for necessário.	Usabilidade
RNF06	O acesso à funcionalidade de geração de relatórios deve estar disponível apenas a usuários autenticados e autorizados.	Segurança

3. PROTÓTIPO DO SISTEMA

O protótipo da aplicação foi desenvolvido utilizando a ferramenta Figma. A seguir, são apresentadas as imagens do protótipo de cada requisito funcional na mesma ordem do Documento de Requisito para documentação no projeto.

3.1. RF 1 – Fazer Login

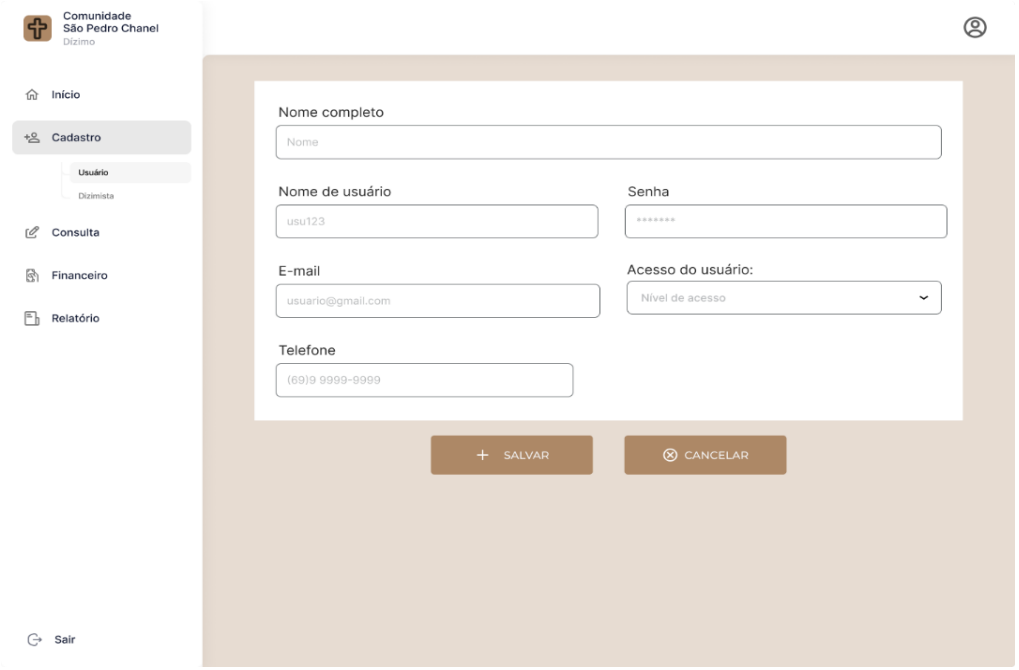
Figura 1 - Tela de login



Fonte: Autoria própria (2025).

3.2. RF 2 – Cadastrar Usuário

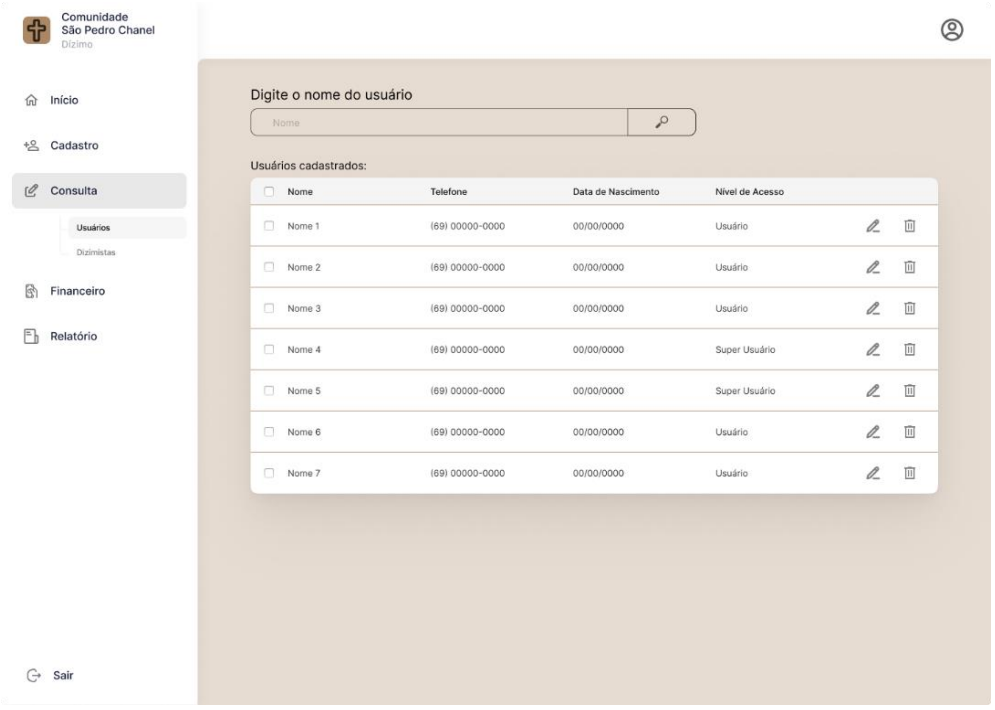
Figura 2 - Tela de cadastro de usuários



Fonte: Autoria própria (2025).

3.3. RF 3 – Consultar Usuário

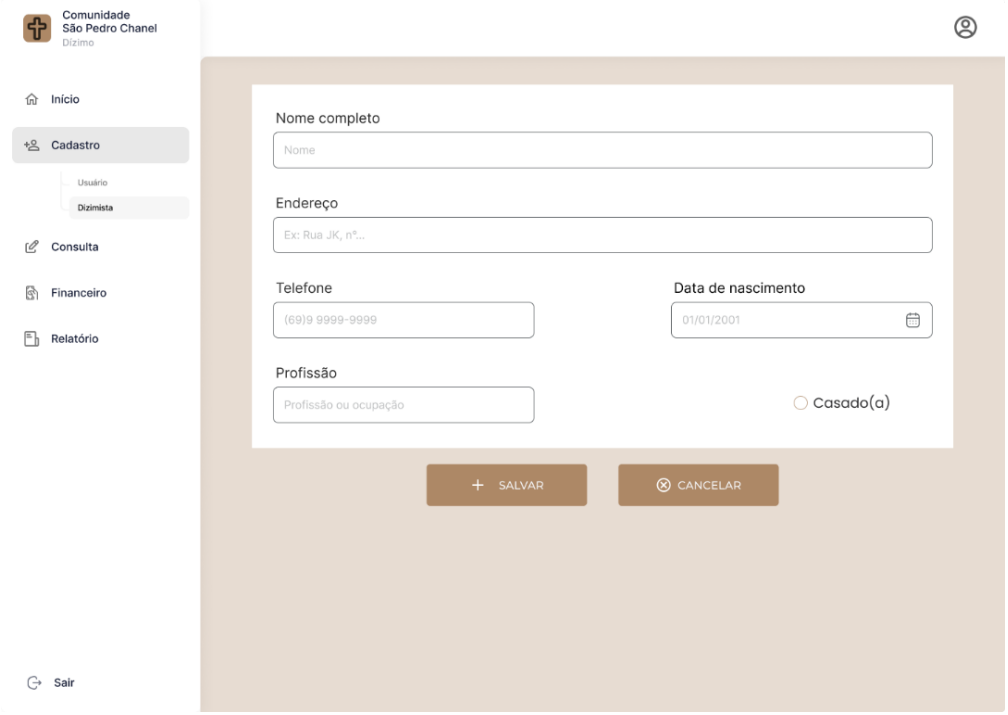
Figura 3 - Tela de consulta de usuários cadastrados



Fonte: Autoria própria (2025).

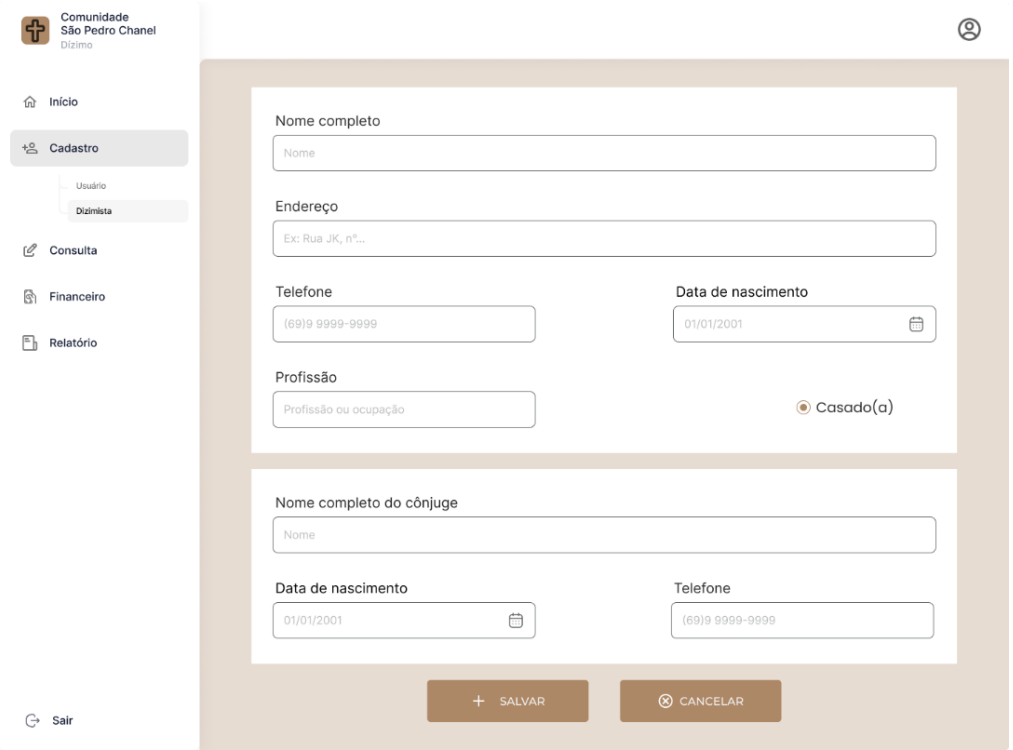
3.4. RF 4 – Cadastrar Contribuinte

Figura 4 - Tela de cadastro de contribuintes



Fonte: Autoria própria (2025).

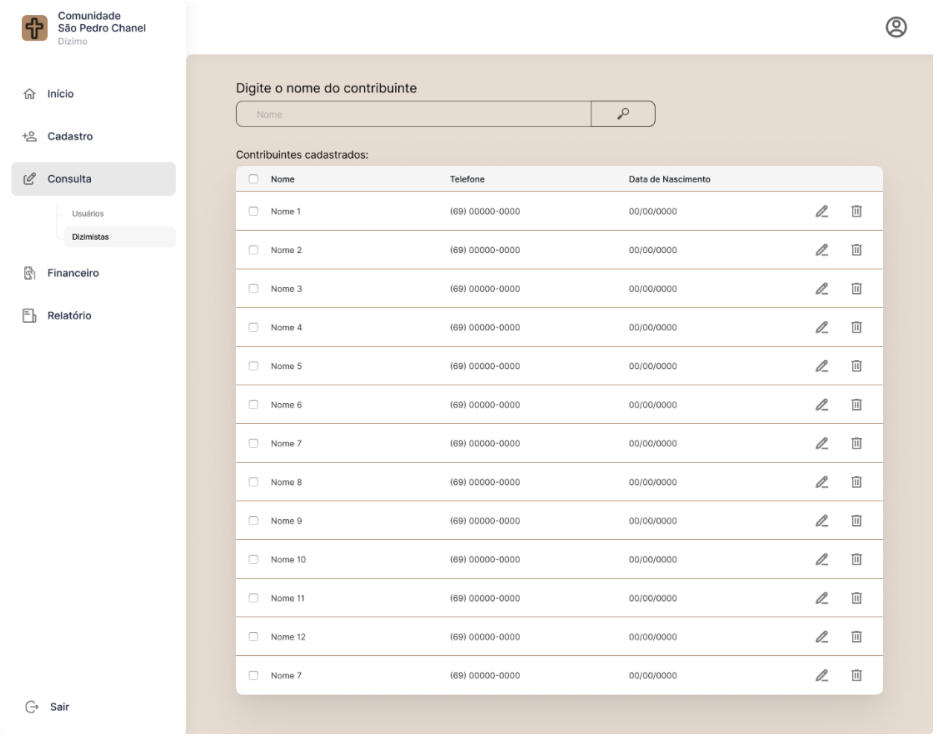
Figura 5 - Tela de cadastro de contribuintes com opção de cônjuge selecionado



Fonte: Autoria própria (2025).

3.5. RF 5 – Consultar Contribuinte

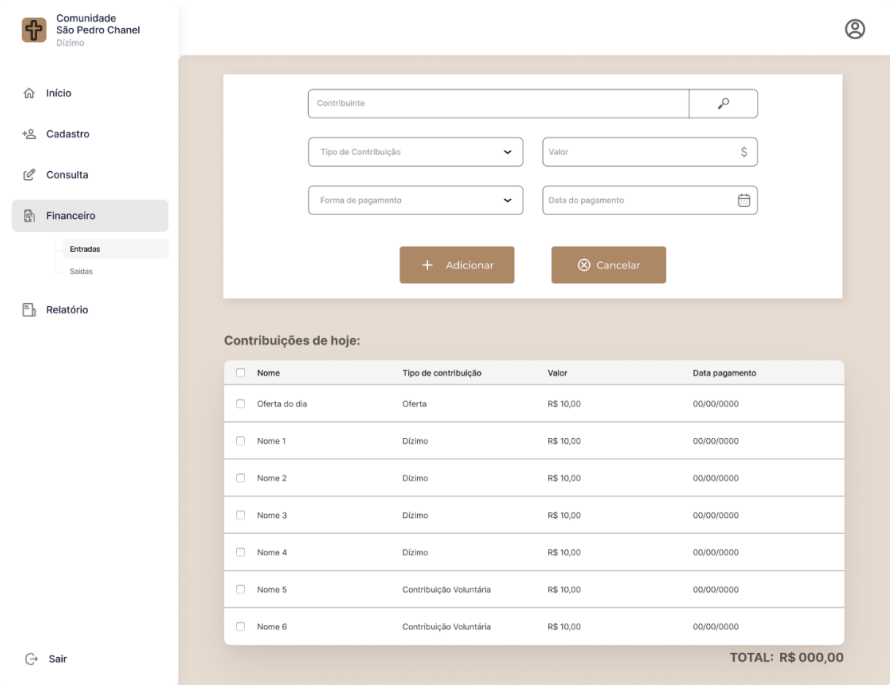
Figura 6 - Tela de consulta de contribuintes cadastrados



Fonte: Autoria própria (2025).

3.6. RF 6 – Registrar Contribuições

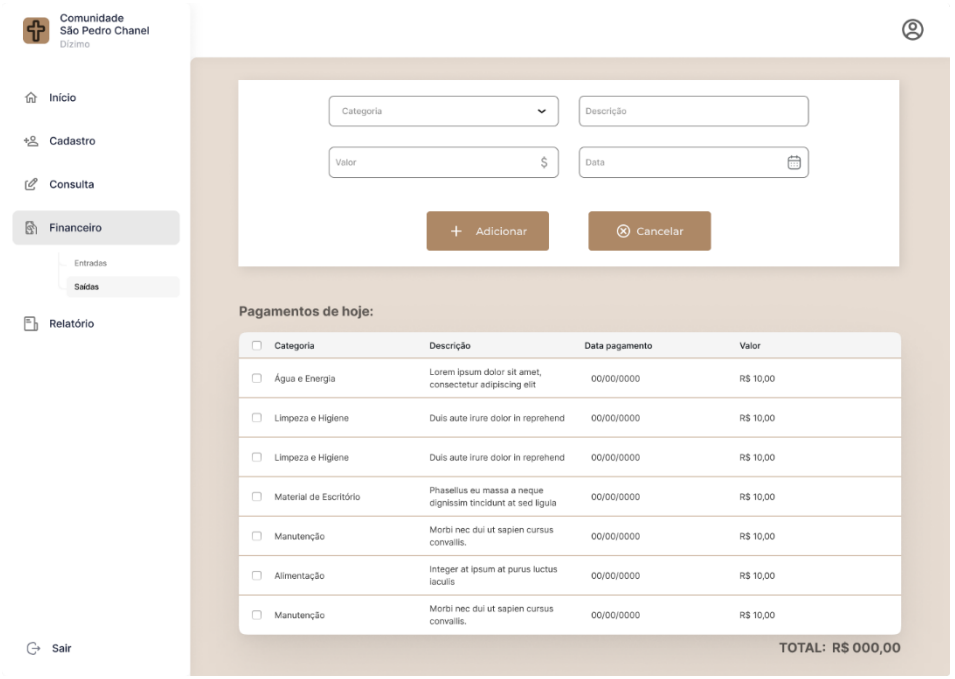
Figura 7 - Tela de registro de contribuições recebidas



Fonte: Autoria própria (2025).

3.7. RF 7 – Registrar Despesas

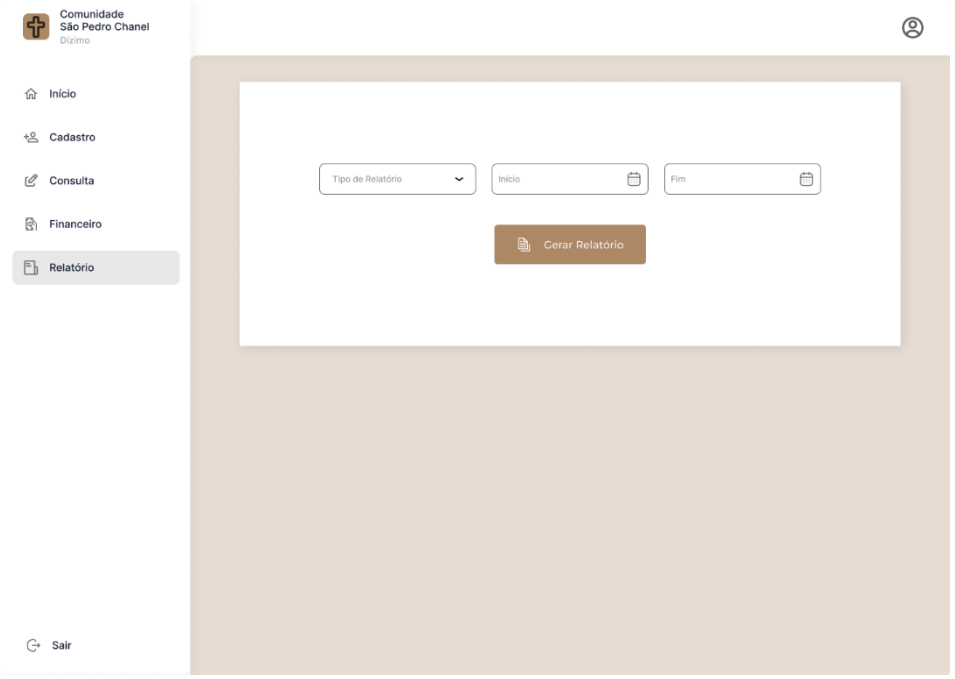
Figura 8 - Tela de registro de despesas da comunidade



Fonte: Autoria própria (2025).

3.8. RF 8 – Emitir Relatórios

Figura 9 - Tela de emissão de relatórios

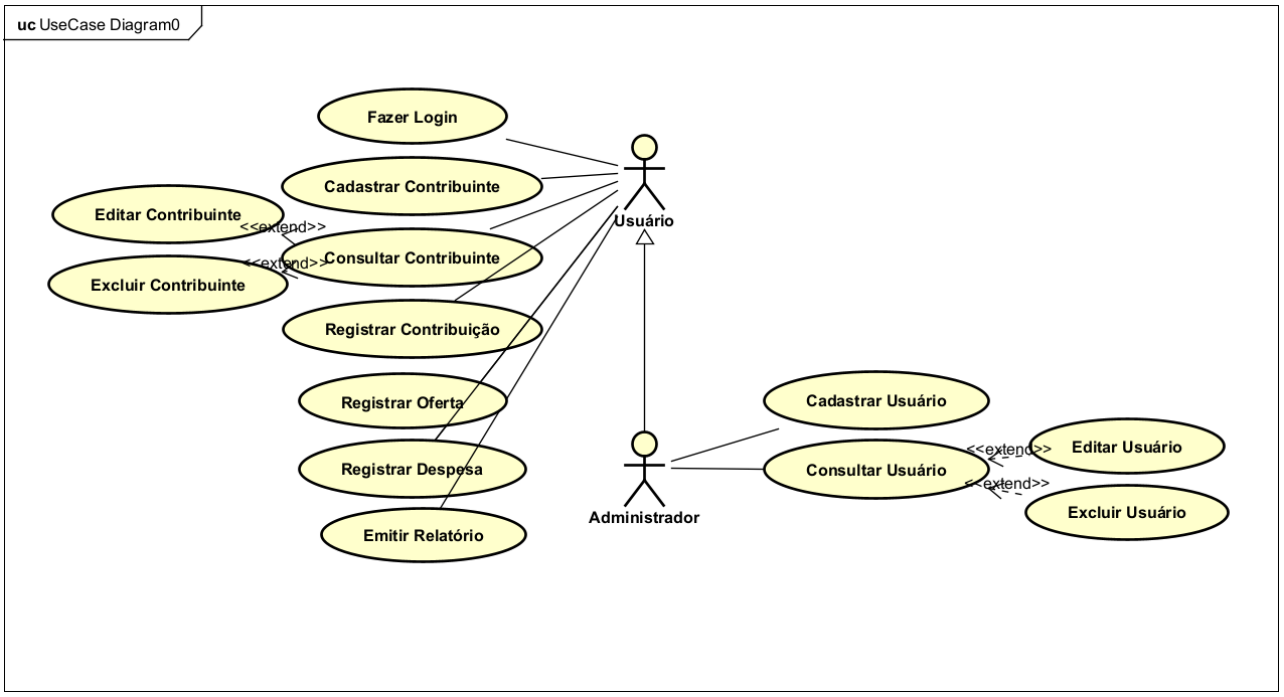


Fonte: Autoria própria (2025).

4. MODELO DE CASO DE USO

4.1. Diagrama de Caso de Uso

Figura 10 - Diagrama de Caso de Uso



Fonte: Autoria própria (2026).

4.2. Caso de Uso Expandido

4.2.1. Caso de Uso – Fazer Login

Caso de Uso: Fazer Login (CSU01)

Descrição: O usuário faz acesso ao sistema.

Ator Primário: Secretaria, Pastoral do Dízimo.

Atores Secundários: Administrador.

Precondições: Estar cadastrado no sistema com o nome de usuário e a senha.

Fluxo Principal

1. Na tela de login, o usuário informa nome de usuário e senha e clica no botão Entrar.
2. O sistema verifica se os dados informados são válidos.
3. Se os dados informados forem válidos o sistema libera o acesso do usuário ao sistema e o redireciona para a interface inicial. Se os dados forem inválidos, o sistema informa onde está o erro e se mantém na tela de login até que os dados informados forem válidos.

Fluxo Alternativo 1: Dados inválidos

1. Se os dados informados forem inválidos o sistema informa e aponta o erro.
2. O fluxo retorna para o passo 1 até que os dados informados sejam válidos.

4.2.2. Caso de Uso – Cadastrar Usuário

Caso de Uso: Cadastrar Usuário (CSU02)

Descrição: O usuário cadastra um novo usuário no sistema.

Ator Primário: Administrador

Atores Secundários: Nenhum.

Precondições: O usuário deve estar autenticado no sistema e ter o nível de acesso necessário para utilizar a função.

Fluxo Principal

1. O usuário acessa a funcionalidade "Usuário" na aba "Cadastro".
2. O sistema exibe o formulário de cadastro.
3. O usuário preenche os dados obrigatórios: nome completo, nome de usuário, email, senha, nível de acesso e telefone.
4. O usuário confirma o cadastro.
5. O sistema valida os dados informados.
6. O sistema salva o novo usuário no banco de dados.
7. O sistema exibe uma mensagem de sucesso.

Fluxo Alternativo 1: Dados obrigatórios não preenchidos

1. O sistema exibe uma mensagem de erro indicando os campos obrigatórios.
2. O fluxo retorna para o passo 3 do fluxo principal.

4.2.3. Caso de Uso – Consultar Usuário

Caso de Uso: Consultar Usuário (CSU03)

Descrição: O usuário visualiza uma tabela com uma lista dos usuários cadastrados no sistema e suas informações.

Ator Primário: Administrador.

Atores Secundários: Nenhum.

Precondições: O usuário deve estar autenticado no sistema e ter o nível de acesso necessário para utilizar a função.

Fluxo Principal

1. O usuário acessa a funcionalidade “Usuário” na aba “Consulta”.
2. O sistema exibe uma lista com todos os usuários cadastrados.
3. O usuário pode optar por pesquisar um registro específico através da barra de pesquisa.
4. O usuário pode optar por editar ou excluir um item da lista.

Fluxo Alternativo 1: Nenhum registro corresponde à pesquisa

1. Conforme o usuário escreve o nome do registro que deseja encontrar, o sistema mostra na lista os nomes que correspondem a pesquisa.
2. Caso não haja nenhum nome correspondente com a pesquisa o sistema mostra a lista vazia.

Fluxo Alternativo 2: Usuário opta por editar um registro

1. O usuário clica no botão de Editar.
2. O sistema exibe um formulário preenchido com os dados do usuário selecionado.
3. O usuário altera os dados e confirma.
4. O sistema valida e atualiza as informações.
5. O fluxo retorna ao passo 2 do fluxo principal.

Fluxo Alternativo 3: Usuário opta por excluir um registro

1. O usuário clica no botão de Excluir.
2. O sistema exibe uma mensagem solicitando confirmação de exclusão dos dados.
3. O usuário confirma.
4. O sistema remove o usuário selecionado do banco de dados.
5. O fluxo retorna ao passo 2 do fluxo principal.

4.2.4. Caso de Uso – Cadastrar Contribuinte

Caso de Uso: Cadastrar Contribuinte (CSU04)

Descrição: O usuário cadastrar um novo contribuinte no sistema.

Ator Primário: Secretaria.

Atores Secundários: Administrador.

Precondições: O usuário deve estar autenticado no sistema e ter o nível de acesso necessário para utilizar a função.

Fluxo Principal

1. O usuário acessa a funcionalidade “Contribuinte” na aba “Cadastro”.
2. O sistema exibe o formulário de cadastro.

3. O usuário preenche os dados obrigatórios: nome completo, endereço, telefone, data de nascimento e profissão.
4. O usuário confirma o cadastro.
5. O sistema valida os dados informados.
6. O sistema salva o novo contribuinte no banco de dados.
7. O sistema exibe uma mensagem de sucesso.

Fluxo Alternativo 1: Dados obrigatórios não preenchidos

1. O sistema exibe uma mensagem de erro indicando os campos obrigatórios.
2. O fluxo retorna para o passo 3 do fluxo principal.

Fluxo Alternativo 2: Estado civil "Casado"

1. Se caso o estado civil do contribuinte for "Casado", o usuário irá selecionar a opção no formulário.
2. O sistema exibe logo abaixo o formulário para preenchimento dos dados obrigatório do cônjuge: nome completo, data de nascimento, telefone e profissão.
3. O fluxo retorna para o passo 4 do fluxo principal.

4.2.5. Caso de Uso – Consultar Contribuinte

Caso de Uso: Consultar Contribuinte (CSU05)

Descrição: O usuário visualiza uma tabela com uma lista dos contribuintes cadastrados no sistema e suas informações.

Ator Primário: Secretaria.

Atores Secundários: Administrador.

Precondições: O usuário deve estar autenticado no sistema e ter o nível de acesso necessário para utilizar a função.

Fluxo Principal

1. O usuário acessa a funcionalidade "Contribuinte" na aba "Consulta".
2. O sistema exibe uma lista com todos os contribuintes cadastrados.
3. O usuário pode optar por pesquisar um registro específico através da barra de pesquisa.
4. O usuário pode optar por editar ou excluir um item da lista.

Fluxo Alternativo 1: Nenhum registro corresponde à pesquisa

1. Conforme o usuário escreve o nome do registro que deseja encontrar, o sistema mostra na lista os nomes que correspondem a pesquisa.

2. Caso não haja nenhum nome correspondente com a pesquisa o sistema mostra a lista vazia.

Fluxo Alternativo 2: Usuário opta por editar um registro

1. O usuário clica no botão Editar.
2. O sistema exibe um formulário preenchido com os dados do contribuinte selecionado.
3. O usuário altera os dados e confirma.
4. O sistema valida e atualiza as informações.
5. O fluxo retorna ao passo 2 do fluxo principal.

Fluxo Alternativo 3: Usuário opta por excluir um registro

1. O usuário clica no botão Excluir.
2. O sistema exibe uma mensagem solicitando confirmação de exclusão dos dados.
3. O usuário confirma.
4. O sistema remove o contribuinte selecionado do banco de dados.
5. O fluxo retorna ao passo 2 do fluxo principal.

4.2.6. Caso de Uso – Registrar Contribuições

Caso de Uso: Registrar Contribuições (CSU06)

Descrição: O usuário registra uma nova contribuição feita por um contribuinte.

Ator Primário: Secretaria, Pastoral do Dízimo.

Atores Secundários: Administrador.

Precondições: O usuário deve estar autenticado no sistema e ter o nível de acesso necessário para utilizar a função.

Fluxo Principal

1. O usuário acessa a função “Contribuições” na aba “Financeiro”.
2. O sistema exibe um formulário para seleção do contribuinte e preenchimento dos dados obrigatórios referente a contribuição: tipo de contribuição, valor, forma de pagamento, data do pagamento e um campo opcional de observação.
3. O usuário confirma o registro.
4. O sistema valida os dados informados.
5. O sistema salva a contribuição no banco de dados, vinculada ao contribuinte selecionado.
6. O sistema exibe uma mensagem de sucesso.

7. Os dados da contribuição, caso forem registradas na data atual, são exibidos em uma tabela logo abaixo do formulário.

Fluxo Alternativo 1: Dados obrigatórios não preenchidos

1. O sistema exibe uma mensagem de erro indicando os campos obrigatórios.
2. O fluxo retorna para o passo 2 do fluxo principal.

4.2.7. Caso de Uso – Registrar Oferta

Caso de Uso: Registrar Oferta (CSU07)

Descrição: O usuário registra uma nova oferta do dia no sistema.

Ator Primário: Secretaria, Pastoral do Dízimo.

Atores Secundários: Administrador.

Precondições: O usuário deve estar autenticado no sistema e ter o nível de acesso necessário para utilizar a função.

Fluxo Principal

1. O usuário acessa a função “Oferta” na aba “Financeiro”.
2. O sistema exibe um formulário para preenchimento dos dados obrigatórios referente a oferta: valor arrecadado, tipo da celebração, data e um campo opcional de observação.
3. O usuário confirma o registro.
4. O sistema valida os dados informados.
5. O sistema salva a oferta no banco de dados.
6. O sistema exibe uma mensagem de sucesso.
7. Os dados da oferta, caso forem registradas na data atual, são exibidos em uma tabela logo abaixo do formulário.

Fluxo Alternativo 1: Dados obrigatórios não preenchidos

1. O sistema exibe uma mensagem de erro indicando os campos obrigatórios.
2. O fluxo retorna para o passo 2 do fluxo principal.

4.2.8. Caso de Uso – Registrar Despesas

Caso de Uso: Registrar Despesas (CSU08)

Descrição: O usuário registra uma nova despesa no sistema.

Ator Primário: Secretaria, Pastoral do Dízimo.

Atores Secundários: Administrador.

Precondições: O usuário deve estar autenticado no sistema e ter o nível de acesso necessário para utilizar a função.

Fluxo Principal

1. O usuário acessa a função “Despesas” na aba “Financeiro”.
2. O sistema exibe um formulário para preenchimento dos dados obrigatórios referente a despesa: categoria, valor, data e um campo opcional de observação.
3. O usuário confirma o registro.
4. O sistema valida os dados informados.
5. O sistema salva a despesa no banco de dados.
6. O sistema exibe uma mensagem de sucesso.
7. Os dados da despesa, caso forem registradas na data atual, são exibidos em uma tabela logo abaixo do formulário.

Fluxo Alternativo 1: Dados obrigatórios não preenchidos

3. O sistema exibe uma mensagem de erro indicando os campos obrigatórios.
4. O fluxo retorna para o passo 2 do fluxo principal.

4.2.9. Caso de Uso – Emitir Relatórios

Caso de Uso: Emitir Relatórios (CSU09)

Descrição: O usuário gera relatórios de informações desejadas com base nas datas selecionadas pelo próprio.

Ator Primário: Secretaria, Pastoral do Dízimo.

Atores Secundários: Administrador.

Precondições: O usuário deve estar autenticado no sistema e ter o nível de acesso necessário para utilizar a função.

Fluxo Principal

1. O usuário acessa a funcionalidade “Relatório”.
2. O sistema exibe um formulário para seleção do tipo de relatório, a data inicial e a final desejada. Caso o usuário não selecionar nenhuma data, o sitema entende como o período padrão o mês atual.
3. O usuário confirma a emissão.
4. O sistema valida os filtros.
5. O sistema consulta os dados no banco com base nos filtros aplicados.

6. O sistema gera e exibe o relatório com as informações solicitadas em uma listagem detalhada.
7. O usuário pode somente visualizar, como também baixar o relatório em PDF.

Fluxo Alternativo 1: Período inválido

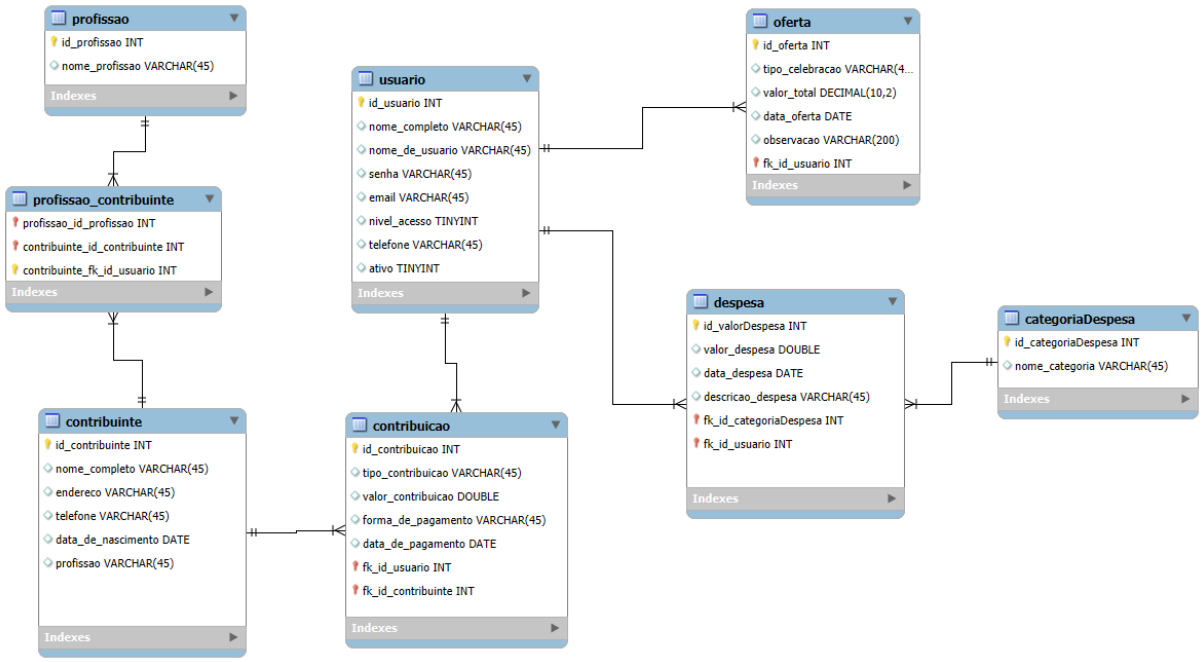
1. O sistema exibe uma mensagem de erro.
2. O fluxo retorna ao passo 2 do fluxo principal.

Fluxo Alternativo 2: Nenhum dado encontrado para os filtros selecionados

1. O sistema exibe a mensagem “Nenhum registro encontrado para os filtros informados”.
2. O fluxo retorna ao passo 2 do fluxo principal.

5. BANCO DE DADOS

Figura 11 - Diagrama Lógico Entidade-Relacionamento.



Fonte: Autoria própria (2026).

REFERÊNCIAS

COPES, Flavio. The JavaScript Beginner's Handbook. 1 mar. 2020. Disponível em: https://www.freecodecamp.org/news/the-complete-javascript-handbook-f26b2c71719c/. Acesso em: 26 jan. 2026.

CALDAS, Angela. Vue.js: o que é, como funciona e como começar a usar esse framework JS. 16 mai. 2024. Disponível em: https://www.alura.com.br/artigos/vue-js. Acesso em: 27 jan. 2026.

CALDEIRA, Raphael P. Banco de dados, database, SGBD: você sabe o que é isso?. 5 mai. 2023. Disponível em: https://academy.indicium.tech/blog/banco-de-dados-database-sgbd-voce-sabe-o-que-e-isso. Acesso em: 8 fev. 2026.

DEVMEDIA. Introdução ao TypeScript. 2016. Disponível em: https://www.devmedia.com.br/introducao-ao-typescript/36729#TypeScript. Acesso em: 09 mar. 2026.

REBELLO, Mariana. Javascript: o que é, como surgiu e onde utilizar. 10 jul. 2022. Disponível em: https://www.resilia.com.br/blog/javascript-o-que-e-como-surgiu-e-onde-utilizar/. Acesso em: 26 jan. 2026.

NESTJS. Documentação. 2017. Disponível em: https://docs.nestjs.com/. Acesso em: 09 mar. 2026.