INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS GUARABIRA
DIRETORIA DE DESENVOLVIMENTO DE ENSINO
COORDENAÇÃO DO CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

José Filipe Oliveira Pereira

RELATÓRIO TÉCNICO

Projeto Integrador em Sistemas para Internet (PISI): Domus

GUARABIRA – PB
2026

José Filipe Oliveira Pereira

RELATÓRIO TÉCNICO
Projeto Integrador em Sistemas para Internet (PISI): Domus

Trabalho de Conclusão de Curso apresentado ao Curso de Tecnologia em Sistemas para Internet, do Instituto Federal da Paraíba – Campus Guarabira, em cumprimento às exigências parciais para a obtenção do título Tecnólogo em Sistemas para Internet.

Orientador(a): Prof. Dr. Rodrigo Leone Alves
Coorientador(a): Prof. Me. Lucas Vieira de Souza

GUARABIRA – PB
2026

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IFPB - GUARABIRA

P436r Pereira, José Filipe Oliveira
Relatório técnico: Projeto Integrador em Sistemas para Internet (PISI): Domus / José Filipe Oliveira Pereira.- Guarabira, 2026.
48f.; il.; color.

Trabalho de Conclusão de Curso (Tecnólogo em Sistemas para Internet). – Instituto Federal da Paraíba, Campus Guarabira. 2026.

"Orientação: Prof. Dr. Rodrigo Leone Alves
Coorientação: Prof. Me. Lucas Vieira de Souza"

Referências.

1.Sistema para internet. 2. Gestão eclesiástica. 3. SaaS. 4. Spring Boot. 5. Isolamento de dados. I. Título.

CDU 004.4(0.067)

Elaborada por Ana Carine da Costa Gonçalves - CRB 000676

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS GUARABIRA

ATA 14/2026 - CCSTSI/DDE/DG/GB/REITORIA/IFPB

ATA DE APRESENTAÇÃO E DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

CCS de Tecnologia em Sistemas para Internet

Aos 06 de agosto de 2026, às 10:00, no Laboratório de Informática, reuniram-se os membros da banca avaliadora, Rodrigo Leone Alves (Orientador), Lucas Vieira de Souza (Coorientador), Tannissa Luanna Cardoso de Araujo (Examinador Interno), Daniel Marques Targino Guimarães (Examinador Interno), para avaliarem a apresentação do *Trabalho de Conclusão do Curso Superior de Tecnologia em Sistemas para Internet* (Relatório Final do Projeto Integrador em Sistemas para Internet - PISI) do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB), *campus* Guarabira, desenvolvido pela aluno(a) **Jose Filipe Oliveira Pereira, matrícula de nº 202323810024**, intitulado "*Domus* ", protocolado para apresentação de acordo com os requisitos expostos no Projeto Pedagógico de Curso de Tecnologia em Sistemas para Internet. Após a apresentação, a banca apresentou, por unanimidade, pareceres a favor da aprovação do trabalho. Desta forma, o Trabalho de Conclusão de Curso foi **aprovado** e definiu-se a **nota final 93 (Noventa e Três)**.

Nada mais havendo a tratar, às 16:45, encerraram-se os trabalhos, determinando a lavratura desta ata, que, após lida e considerada conforme, será assinada pelos presentes. Eu, Rodrigo Leone Alves, lavrei a presente ata.

Guarabira/PB, em 10 de agosto de 2026.

Documento assinado eletronicamente por:

- **Rodrigo Leone Alves**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/08/2026 11:52:18.
- **Tannissa Luanna Cardoso de Araujo**, **PEDAGOGO-AREA**, em 10/08/2026 12:47:42.
- **Lucas Vieira de Souza**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/08/2026 14:21:56.
- **Daniel Marques Targino Guimaraes**, **PROF ENS BAS TEC TECNOLOGICO-SUBSTITUTO**, em 10/08/2026 22:47:01.

Este documento foi emitido pelo SUAP em 10/08/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 919877
Código de Autenticação: 96c73fe943

A Deus, que me sustentou com graça e misericórdia todos os dias. À minha namorada, por todo amor, suporte e ouvidos. Aos meus pais e irmãs, pelo amor e sustento em cada etapa. Dedico.

AGRADECIMENTOS

A Deus, pela graça, pela força e pela direção ao longo de toda esta jornada.

Ao Prof. Dr. Rodrigo Leone Alves, orientador deste trabalho, pela paciência, pela confiança e por cada orientação que transformou dúvidas em direção. Ao Prof. Me. Lucas Vieira de Souza, coorientador, pelas contribuições técnicas precisas e pelo incentivo constante ao longo do desenvolvimento.

Aos professores do Curso de Tecnologia em Sistemas para Internet do IFPB Campus Guarabira, que durante todo o curso compartilharam conhecimento, desafiaram o pensamento crítico e ajudaram a construir a base técnica que tornou este projeto possível.

Ao IFPB Campus Guarabira, pela estrutura, pelo acolhimento e por ter sido, ao longo desses anos, um ambiente real de aprendizado e crescimento.

À Igreja Batista Central de Piancó, que generosamente abriu as portas e atuou como organização-piloto deste trabalho, permitindo que o Domus fosse pensado e validado a partir de necessidades concretas e reais.

Aos colegas de curso, pela parcerias, pelas trocas de conhecimento e pelo apoio mútuo que tornou a caminhada mais leve.

Aos meus pais, pelo amor incondicional e pelo sustento em cada etapa, sem os quais nada disso seria possível. Às minhas irmãs, pela presença constante e pelo carinho. Ao grande amor da minha vida, por cada palavra de incentivo, por cada ouvido paciente e por acreditar em mim mesmo nos dias mais difíceis.

Aos amigos, que tornaram esse tempo mais fácil.

A todos que, direta ou indiretamente, contribuíram para a realização deste trabalho, meu sincero agradecimento.

"Pela graça de Deus eu sou o que sou; e se eu for algo mais, será por Sua graça."

Charles Spurgeon

RESUMO

A gestão administrativa de igrejas de pequeno e médio porte enfrenta desafios recorrentes pela ausência de ferramentas especializadas, levando essas organizações a dependerem de planilhas eletrônicas, mensagens informais e registros manuais para processos essenciais como o cadastro de membros, a organização de eventos e o controle financeiro. Esse cenário gera inconsistências cadastrais, perda de histórico financeiro e dificuldades na prestação de contas, além de expor as organizações a riscos de conformidade com a Lei Geral de Proteção de Dados. Diante desse problema, o presente trabalho apresenta o desenvolvimento do Domus, um sistema web do tipo Software as a Service (SaaS) para a gestão administrativa de múltiplas igrejas. O objetivo geral consistiu em desenvolver uma plataforma centralizada que cobrisse os módulos de membros, eventos e financeiro, com isolamento lógico de dados por igreja e controle de acesso baseado em perfis de usuário. O isolamento lógico dos dados foi garantido pela aplicação consistente do identificador da igreja em todas as operações de leitura e escrita, inclusive nas consultas ao índice de busca, com extração segura a partir do token de autenticação. Como resultado, obteve-se um sistema funcional contendo os principais módulos administrativos, com interface responsiva, busca avançada, medidas de adequação aos princípios da Lei Geral de Proteção de Dados e arquitetura preparada para evoluções futuras. Conclui-se que a aplicação dos conceitos estudados durante o curso permitiu construir uma solução tecnicamente sólida e socialmente relevante, contribuindo para a digitalização de organizações que historicamente operam com recursos limitados.

Palavras-chave: Gestão eclesiástica; SaaS; Isolamento de dados; Elasticsearch; Spring Boot.

ABSTRACT

The administrative management of small and medium-sized churches faces recurring challenges due to the lack of specialized tools, leading these organizations to rely on spreadsheets, informal messaging, and manual records for essential processes such as member registration, event organization, and financial control. This scenario generates registration inconsistencies, loss of financial history, and accountability difficulties, while also exposing organizations to compliance risks regarding the General Data Protection Law. In view of this problem, this work presents the development of Domus, a Software as a Service (SaaS) web system for the administrative management of multiple churches. The general objective was to develop a centralized platform covering the modules of members, events, and finance, with logical data isolation per church and access control based on user profiles. Logical data isolation was ensured by the consistent application of the church identifier across all read and write operations, including search index queries, with secure extraction from the authentication token. As a result, a functional system was obtained containing the main administrative modules, with a responsive interface, advanced search, measures aligned with the principles of the General Data Protection Law, and an architecture prepared for future evolutions. It is concluded that the application of the concepts studied during the course allowed the construction of a technically sound and socially relevant solution, contributing to the digitalization of organizations that historically operate with limited resources.

Keywords: Church management; SaaS; Data isolation; Elasticsearch; Spring Boot.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ANPD	Autoridade Nacional de Proteção de Dados
API	Application Programming Interface
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheets
DTO	Data Transfer Object
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IFPB	Instituto Federal da Paraíba
JPA	Jakarta Persistence API
JSON	JavaScript Object Notation
JWT	JSON Web Token
LGPD	Lei Geral de Proteção de Dados
MVP	Minimum Viable Product
ODS	Objetivos de Desenvolvimento Sustentável
ONU	Organização das Nações Unidas
PISI	Projeto Integrador em Sistemas para Internet
REST	Representational State Transfer
SaaS	Software as a Service
SQL	Structured Query Language
UUID	Universally Unique Identifier

SUMÁRIO

1 INTRODUÇÃO

A transformação digital tem alcançado setores cada vez mais diversos da sociedade, oferecendo soluções tecnológicas capazes de otimizar processos administrativos, garantir maior segurança no tratamento de informações e ampliar o acesso a serviços essenciais. No entanto, organizações religiosas — em especial igrejas de pequeno e médio porte — permanecem frequentemente à margem desse processo, recorrendo a planilhas eletrônicas, mensagens informais e registros manuais para conduzir atividades operacionais que poderiam ser sistematizadas por meio de software especializado.

O presente trabalho propõe a construção do Domus, um sistema web para gestão administrativa de múltiplas igrejas, desenvolvido como Projeto Integrador em Sistemas para Internet (PISI) do curso de Tecnologia em Sistemas para Internet do Instituto Federal da Paraíba — Campus Guarabira. O Domus reúne, em uma plataforma única, os processos centrais da administração eclesiástica: gestão de membros, organização de eventos e controle financeiro.

1.1 Dados Gerais

O projeto Domus foi concebido no âmbito do Trabalho de Conclusão de Curso do autor José Filipe Oliveira Pereira, discente do curso de Tecnologia em Sistemas para Internet do Instituto Federal da Paraíba (IFPB) — Campus Guarabira. A orientação acadêmica foi conduzida pelo Prof. Dr. Rodrigo Leone Alves, com coorientação do Prof. Lucas Vieira de Souza, ambos vinculados ao IFPB Campus Guarabira.

O desenvolvimento ocorreu durante o sexto semestre do curso, entre os meses de fevereiro e agosto de 2026, contemplando as etapas de levantamento de requisitos, projeto, desenvolvimento e validação da solução proposta.

1.2 Tema

O tema deste trabalho está centrado no desenvolvimento de uma plataforma Software as a Service (SaaS) destinada à gestão administrativa de múltiplas organizações religiosas, abrangendo os módulos de gestão de membros, organização de eventos e controle financeiro. O trabalho explora como princípios consolidados de engenharia de software — como isolamento de dados multilocatário, autenticação baseada em tokens, controle de acesso por perfis de usuário e desenvolvimento orientado a User Stories (COHN, 2004) — podem ser aplicados para atender às necessidades específicas de organizações religiosas de pequeno e médio porte, que historicamente operam com recursos limitados e dependem de processos manuais para sua administração cotidiana. Ao propor uma solução centralizada, segura e de baixo custo

operacional, o Domus busca democratizar o acesso a ferramentas profissionais de gestão e contribuir para a digitalização de um segmento com baixa adoção de software especializado.

1.3 Delimitação do Problema

O escopo da versão inicial (Minimum Viable Product — MVP) do Domus contempla os seguintes módulos funcionais: autenticação e recuperação de senha; gestão de usuários internos da igreja; cadastro, consulta, edição e remoção de membros; gestão de eventos; lançamento e controle de movimentações financeiras com categorização e relatórios consolidados por período; e busca global unificada sobre as entidades do sistema.

Estão fora do escopo deste trabalho funcionalidades que demandariam ciclos adicionais de desenvolvimento ou integrações externas, entre elas: processamento de pagamentos online, controle de contas a pagar e a receber (gestão de compromissos financeiros futuros), controle de presença em eventos, aplicativo mobile nativo, sistema de notificações automáticas, exportação de relatórios em formatos PDF, Excel e CSV, integração com serviços de terceiros e perfil de Tesoureiro com permissões granulares. Tais funcionalidades estão previstas no roadmap de evolução do produto e serão tratadas como trabalhos futuros.

1.4 Justificativa

A escolha pelo desenvolvimento de uma plataforma de gestão eclesiástica fundamenta-se em três dimensões complementares: técnica, social e acadêmica.

Do ponto de vista técnico, o projeto representa uma oportunidade de aplicar, de forma integrada, conceitos consolidados de engenharia de software contemporânea — isolamento lógico de dados por igreja, segurança baseada em tokens, controle de acesso por papéis, persistência relacional com migrações versionadas e desenvolvimento frontend baseado em componentes — em um cenário realista que demanda soluções robustas e escaláveis.

Sob a perspectiva social, observou-se, no contexto da organização-piloto deste trabalho, a operação com recursos limitados, dispondo de orçamento reduzido para aquisição de softwares comerciais e equipe técnica restrita para customização de soluções genéricas. A ausência de uma plataforma especializada de baixo custo perpetua a dependência de processos manuais sujeitos a falhas, comprometendo a transparência da gestão financeira e a comunicação com a comunidade. O Domus busca democratizar o acesso a ferramentas profissionais de administração, contribuindo para a digitalização de organizações com baixa adoção de software de gestão especializado.

Academicamente, o desenvolvimento do Domus permite consolidar competências adquiridas ao longo do curso, incluindo programação orientada a objetos, modelagem de dados, segurança em aplicações web, design de interfaces e gerenciamento de projetos. Além disso, a documentação rigorosa do processo de desenvolvimento, alinhada às práticas profissionais da

indústria de software, fortalece o perfil técnico do autor e gera artefatos que poderão servir de referência para projetos similares.

1.5 Objetivo do Projeto

1.5.1 Objetivo Geral

Desenvolver o Domus, sistema web do tipo SaaS para gestão administrativa de múltiplas igrejas, contemplando os módulos de gestão de membros, organização de eventos e controle financeiro, com isolamento lógico de dados por igreja, controle de acesso baseado em perfis de usuário e interface adequada ao contexto eclesiástico.

1.5.2 Objetivos Específicos

Para alcançar o objetivo geral, foram estabelecidos os seguintes objetivos específicos:

a) projetar o banco de dados relacional (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2024) com isolamento lógico por igreja, garantindo que cada organização acesse exclusivamente seus próprios dados;

b) desenvolver a aplicação back-end em Java com o framework Spring Boot (VMWARE, 2024), implementando os módulos de autenticação, membros, eventos, financeiro e busca;

c) desenvolver a aplicação front-end com interface responsiva, destinada à interação dos usuários com os módulos do sistema.

1.6 Método de Trabalho

O desenvolvimento do Domus foi conduzido com ciclo iterativo e incremental orientado a User Stories. As atividades foram organizadas em etapas sequenciais, com sobreposição parcial, contemplando: levantamento e formalização de requisitos; modelagem do domínio e da arquitetura; desenvolvimento do backend; desenvolvimento do frontend; integração das camadas; testes manuais e validação; e documentação final.

A escolha pelo desenvolvimento iterativo justifica-se pela natureza do projeto, em que decisões de design técnico precisam ser validadas progressivamente à medida que novos módulos são integrados. A cada ciclo, um conjunto de User Stories é selecionado para desenvolvimento, seguido de testes de validação e refinamento da documentação correspondente.

A organização do código e da documentação adotou práticas reconhecidas pela indústria: versionamento com Git, fluxo de branches por funcionalidade, mensagens de commit no padrão Conventional Commits, e centralização da documentação técnica em ambiente colaborativo. Tais práticas, embora não sejam objeto direto da avaliação acadêmica, garantem rastreabilidade ao

processo e aproximam a experiência educacional do cotidiano profissional do desenvolvedor de software.

1.7 Organização do Trabalho

O presente relatório está estruturado em cinco capítulos, complementados pela seção de referências.

O Capítulo 1, Introdução, apresenta o contexto do projeto, seus objetivos, a delimitação do problema, a justificativa, o método de trabalho adotado e a relação do projeto com os Objetivos de Desenvolvimento Sustentável e com a Lei Geral de Proteção de Dados.

O Capítulo 2, Descrição Geral do Problema, detalha o problema endereçado pelo sistema, descreve os perfis de usuário previstos e formaliza os requisitos funcionais e não funcionais (SOMMERVILLE, 2019).

O Capítulo 3, Análise e Design, apresenta os artefatos de modelagem do sistema, incluindo o diagrama de classes, o modelo lógico da base de dados, o dicionário de dados e a descrição do ambiente de desenvolvimento.

O Capítulo 4, Resultados Obtidos, documenta a arquitetura efetivamente construída, as tecnologias adotadas, as decisões de desenvolvimento relevantes e as telas resultantes do sistema.

O Capítulo 5, Conclusão, apresenta as considerações finais, retoma os objetivos propostos avaliando seu cumprimento e discute possibilidades de evolução futura do Domus.

1.8 Sobre os Objetivos de Desenvolvimento Sustentável

O projeto Domus alinha-se a três Objetivos de Desenvolvimento Sustentável (ODS) estabelecidos pela Organização das Nações Unidas (ONU) na Agenda 2030 (ORGANIZAÇÃO DAS NAÇÕES UNIDAS, 2015).

O ODS 9 — Indústria, Inovação e Infraestrutura — é contemplado por meio do desenvolvimento de uma solução tecnológica inovadora aplicada a um segmento com oferta reduzida de software especializado. A construção de uma plataforma SaaS escalável demonstra o potencial da inovação tecnológica para resolver problemas administrativos concretos enfrentados por organizações da sociedade civil.

O ODS 10 — Redução das Desigualdades — relaciona-se com a proposta do Domus de democratizar o acesso a ferramentas profissionais de gestão. Ao oferecer uma alternativa de baixo custo operacional, o sistema reduz a barreira tecnológica enfrentada por igrejas com orçamento limitado, contribuindo para diminuir a desigualdade no acesso a tecnologias administrativas modernas entre organizações de diferentes portes.

O ODS 16 — Paz, Justiça e Instituições Eficazes — é apoiado pelos mecanismos de

transparência e rastreabilidade incorporados ao sistema. O controle estruturado das movimentações financeiras, a manutenção de histórico de operações e a segregação de responsabilidades por perfil de usuário fortalecem a prestação de contas das igrejas perante suas comunidades, contribuindo para a construção de instituições mais transparentes e responsáveis.

1.9 Sobre a Lei Geral de Proteção de Dados

A Lei nº 13.709, de 14 de agosto de 2018 (BRASIL, 2018), conhecida como Lei Geral de Proteção de Dados (LGPD), estabelece regras sobre o tratamento de dados pessoais por pessoas físicas ou jurídicas, com o objetivo de proteger os direitos fundamentais de liberdade, privacidade e livre desenvolvimento da personalidade da pessoa natural.

O Domus, ao manipular dados pessoais de membros de igrejas — incluindo nome, endereço, telefone, data de nascimento e e-mail — caracteriza-se como agente de tratamento de dados na qualidade de operador, conforme definido no artigo 5º, inciso VII, da LGPD. A igreja contratante, por sua vez, atua como controladora, sendo responsável pelas decisões referentes ao tratamento dos dados de seus membros. A base legal para o tratamento é a execução de contrato firmado entre o Domus e a igreja contratante, no qual são estabelecidas as obrigações de cada parte quanto à proteção dos dados pessoais envolvidos.

O princípio da finalidade é observado mediante a coleta de dados estritamente vinculados às operações administrativas da igreja, sem armazenamento de informações que não tenham propósito legítimo definido. O princípio da minimização orienta o cadastro de membros, em que campos como data de nascimento e endereço são opcionais, permitindo que cada igreja determine quais informações são efetivamente necessárias para sua gestão.

A segurança das informações é assegurada por meio de múltiplas camadas técnicas: senhas armazenadas com algoritmo de hash bcrypt (PROVOS; MAZIÈRES, 1999); comunicação criptografada via HTTPS em todas as requisições; autenticação baseada em tokens JSON Web Token (JWT) com tempo de expiração configurado; proteção contra ataques de força bruta por meio de mecanismo de bloqueio temporário de contas após sucessivas tentativas de login malsucedidas; e isolamento lógico de dados entre as diferentes igrejas, garantido pela aplicação consistente do identificador da igreja em todas as operações de leitura e escrita executadas pelo sistema.

Quanto ao exercício dos direitos previstos no artigo 18 da LGPD pelos titulares — confirmação da existência de tratamento, acesso, correção, anonimização, eliminação, portabilidade e revogação de consentimento —, o sistema prevê dois caminhos. Os membros que possuem conta de acesso (perfil MEMBRO) podem visualizar parte de seus próprios dados diretamente na plataforma. Para os demais direitos e para os membros que não possuem acesso ao sistema, o exercício ocorre por intermédio da igreja, em sua condição de controladora, que dispõe das ferramentas administrativas no módulo de gestão de membros para atender às solicitações dos

titulares.

Como instrumentos formais de transparência e responsabilização, o Domus disponibilizará Política de Privacidade própria, descrevendo seu papel como operador e os procedimentos de tratamento adotados, bem como Termos de Uso direcionados às igrejas contratantes, formalizando suas responsabilidades como controladoras dos dados de seus membros. Adicionalmente, recomenda-se às igrejas usuárias a elaboração de política de privacidade interna que informe seus membros sobre o tratamento de dados realizado por intermédio do sistema, garantindo o cumprimento do princípio da transparência estabelecido pela legislação.

2 DESCRIÇÃO GERAL DO PROBLEMA

Este capítulo apresenta uma visão detalhada do problema que motivou o desenvolvimento do Domus. A Seção 2.1 caracteriza as dificuldades enfrentadas na gestão administrativa de igrejas de pequeno e médio porte; a Seção 2.2 descreve os perfis de usuário previstos para o sistema; e a Seção 2.3 formaliza os requisitos funcionais e não funcionais que orientam a construção da solução.

2.1 Descrição do Problema

Igrejas de pequeno e médio porte desempenham, além de suas atividades religiosas, um relevante papel social e filantrópico em suas comunidades, promovendo ações de assistência, acolhimento e apoio a famílias em situação de vulnerabilidade. Esse trabalho demanda organização administrativa: cadastro e acompanhamento de membros e famílias, planejamento de eventos comunitários e controle das finanças internas. Contudo, essas organizações geralmente operam com orçamento limitado — proveniente majoritariamente de doações de seus frequentadores —, o que restringe sua capacidade de investir em soluções tecnológicas especializadas para a gestão de suas atividades.

A partir da observação empírica da rotina administrativa de uma igreja parceira, que atuou como organização-piloto do projeto, evidenciou-se que, na ausência de ferramentas especializadas, essas organizações recorrem a planilhas eletrônicas, grupos de mensagens instantâneas e registros manuais para conduzir processos essenciais, como o cadastro e o acompanhamento de membros, a organização da agenda de eventos e o controle das finanças.

Essa abordagem dispersa e não estruturada produz consequências diretas: inconsistência nos dados cadastrais dos membros, perda de histórico financeiro, ausência de rastreabilidade nas movimentações de entrada e saída, dificuldade na prestação de contas às lideranças e à congregação e retrabalho constante na consolidação manual de informações provenientes de fontes distintas.

Sob a perspectiva técnica, a ausência de isolamento de dados representa um risco adicional quando sistemas genéricos são compartilhados entre múltiplas organizações ou quando líderes e membros acessam informações sem um controle de acesso adequado por perfil. Nesses cenários, não há garantia de confidencialidade nem de integridade de informações sensíveis, como os dados pessoais dos membros e os registros financeiros da instituição.

Do ponto de vista legal, a Lei Geral de Proteção de Dados (Lei nº 13.709/2018) impõe às organizações que tratam dados pessoais a obrigação de adotar mecanismos adequados de coleta, armazenamento e proteção dessas informações. Igrejas que mantêm cadastros de membros com dados como data de nascimento, endereço e telefone estão sujeitas a essas exigências e, sem uma

solução apropriada, ficam expostas a riscos de conformidade.

Diante desse cenário, identifica-se a necessidade de uma solução centralizada, segura, de baixo custo e de fácil utilização, capaz de substituir os processos manuais por fluxos digitais estruturados, garantindo rastreabilidade das operações, controle de acesso por perfil e isolamento lógico dos dados entre as organizações usuárias da plataforma.

2.2 Perfis de Usuários

O Domus prevê três perfis de usuário, cada um com um conjunto distinto de permissões, definidas de acordo com as responsabilidades exercidas no contexto administrativo da igreja. O controle de acesso é aplicado em nível de módulo e de operação, garantindo que cada usuário acesse apenas os recursos compatíveis com o seu perfil.

Administrador da Igreja (`ADMIN_IGREJA`)**:** perfil com acesso total a todos os módulos da igreja. É responsável pelo cadastro e pela gestão dos demais usuários, pela administração de membros e eventos e pelo controle financeiro. No escopo da versão inicial, é o único perfil com acesso ao módulo financeiro.

Líder (`LIDER`)**:** perfil destinado a líderes de ministérios e auxiliares administrativos. Possui acesso de leitura ao cadastro de membros e acesso de leitura e escrita ao módulo de eventos, podendo cadastrar e gerenciar a agenda da igreja. Não possui acesso ao módulo financeiro.

Membro (`MEMBRO`)**:** perfil destinado aos membros da comunidade. Possui acesso de leitura às informações que lhe são pertinentes, em especial à agenda de eventos e ao cadastro de membros da igreja, incluindo a busca por essas entidades. A inscrição em eventos por meio do sistema está prevista como evolução futura, não integrando o escopo desta versão.

A busca global respeita a matriz de permissões: cada perfil obtém resultados apenas das entidades que lhe são acessíveis. Assim, dados financeiros e de usuários, restritos ao Administrador da Igreja, não são retornados nas buscas realizadas pelos perfis Líder e Membro.

Cabe destacar que, no contexto da Lei Geral de Proteção de Dados, a igreja atua como controladora dos dados de seus membros, sendo responsável pelas decisões relativas ao tratamento dessas informações. Dessa forma, ainda que o perfil Membro disponha de acesso de leitura ao sistema, as decisões sobre o tratamento de seus dados permanecem sob responsabilidade da igreja.

2.3 Requisitos

Os requisitos do sistema foram levantados de forma iterativa, a partir da análise do domínio da gestão eclesiástica e da definição de *User Stories*, e estão organizados em requisitos

funcionais (Seção 2.3.1) e não funcionais (Seção 2.3.2). Cada requisito é identificado por um código único e classificado quanto à prioridade, conforme as seguintes definições:

Essencial: requisito imprescindível, sem o qual o sistema não cumpre seu propósito; deve ser implementado impreterivelmente.

Importante: requisito cuja ausência não impede o funcionamento do sistema, mas compromete parte de sua utilidade; deve ser implementado sempre que viável.

Desejável: requisito que não compromete as funcionalidades básicas, podendo ser postergado para versões posteriores.

2.3.1 Requisitos Funcionais

Os requisitos funcionais descrevem as capacidades que o sistema deve oferecer. O Quadro 1 apresenta os requisitos funcionais do Domus, organizados por módulo. O detalhamento dos atributos de cada entidade é apresentado no dicionário de dados, no Capítulo 3.

Quadro 1 – Requisitos funcionais do Domus.

Código	**Descrição (o sistema deve permitir)**	**Prioridade**
Autenticação e Usuários		
RF-01	Cadastrar uma igreja com a criação do usuário administrador inicial.	Essencial
RF-02	Autenticar usuários por e-mail e senha.	Essencial
RF-03	Encerrar a sessão do usuário (logout).	Essencial
RF-04	Cadastrar, listar, editar e remover usuários da igreja, incluindo a definição de seu perfil de acesso.	Importante
Membros		
RF-05	Cadastrar, listar, editar e remover membros.	Essencial
Eventos		
RF-06	Cadastrar, editar e remover eventos.	Importante
RF-07	Visualizar os eventos em formato de cards.	Importante
Financeiro		
RF-08	Cadastrar, listar, editar e remover categorias financeiras.	Importante
RF-09	Cadastrar, listar, editar e remover movimentações financeiras de entrada e saída.	Essencial
RF-10	Filtrar as movimentações financeiras por tipo, categoria e período.	Importante

continua na próxima página

Quadro 1 — continuação

Código	**Descrição (o sistema deve permitir)**	**Prioridade**
RF-11	Gerar relatório financeiro por período, com total de entradas, saídas e saldo.	Importante
RF-12	Gerar relatório financeiro por categoria.	Importante
Busca		
RF-13	Pesquisar por texto nas listagens de membros, eventos, usuários e categorias.	Importante
RF-14	Realizar busca global unificada e tolerante a erros de digitação sobre as entidades do sistema, respeitando o perfil de acesso do usuário.	Importante

Fonte: Elaborado pelo autor (2026).

2.3.2 Requisitos Não Funcionais

Os requisitos não funcionais descrevem atributos de qualidade e restrições que a solução deve atender. O Quadro 2 apresenta os requisitos não funcionais considerados no escopo deste trabalho, organizados por categoria. São listados apenas os requisitos verificáveis no contexto do desenvolvimento e da validação realizados; metas de qualidade que dependem de operação em ambiente de produção são tratadas separadamente, ao final desta seção.

Quadro 2 – Requisitos não funcionais do Domus.

Código	**Descrição**	**Prioridade**
Segurança		
RNF-01	As senhas devem ser armazenadas com algoritmo de hash bcrypt.	Essencial
RNF-02	A autenticação deve utilizar tokens JWT com tempo de expiração configurável.	Essencial
RNF-03	O sistema deve implementar proteção contra força bruta no login, com bloqueio temporário após tentativas sucessivas malsucedidas.	Essencial
RNF-04	O sistema deve garantir o isolamento lógico dos dados entre as igrejas, validando o identificador da igreja em todas as operações de leitura e escrita na camada de serviço, inclusive nas consultas ao índice de busca.	Essencial
Desempenho		
RNF-05	As listagens devem ser paginadas.	Importante

continua na próxima página

Quadro 2 — continuação

Código	**Descrição**	**Prioridade**
RNF-06	O sistema deve utilizar cache em memória para reduzir a latência de leituras frequentes, com invalidação automática mediante alterações nos dados.	Importante
Usabilidade		
RNF-07	A interface deve ser responsiva, adaptando-se a desktops e dispositivos móveis.	Importante
RNF-08	O sistema deve apresentar mensagens de validação e erro claras e consistentes.	Importante
Confiabilidade e Manutenibilidade		
RNF-09	O sistema deve registrar logs estruturados das operações para apoio à depuração.	Desejável
Busca		
RNF-10	A busca global deve refletir as alterações nos dados de forma assíncrona, garantindo consistência eventual entre a base transacional e o índice de busca.	Importante
RNF-11	A busca deve ser tolerante a variações e erros de digitação, retornando resultados aproximados.	Desejável
Conformidade (LGPD)		
RNF-12	O sistema deve coletar apenas os dados necessários à sua finalidade.	Essencial

Fonte: Elaborado pelo autor (2026).

Além dos requisitos listados, foram identificadas metas de qualidade que, por dependerem de operação contínua em ambiente de produção, não integram o escopo verificável deste trabalho, ficando registradas como direção para evoluções futuras: disponibilidade monitorada, com meta de 99,5% e alertas de indisponibilidade; rotina formal de *backup* automático com teste periódico de restauração; e validação da escalabilidade horizontal por meio de testes de carga. A arquitetura do sistema foi projetada de modo a viabilizar essas metas — por exemplo, mantendo a aplicação sem estado (*stateless*) —, ainda que sua comprovação esteja prevista para etapas posteriores.

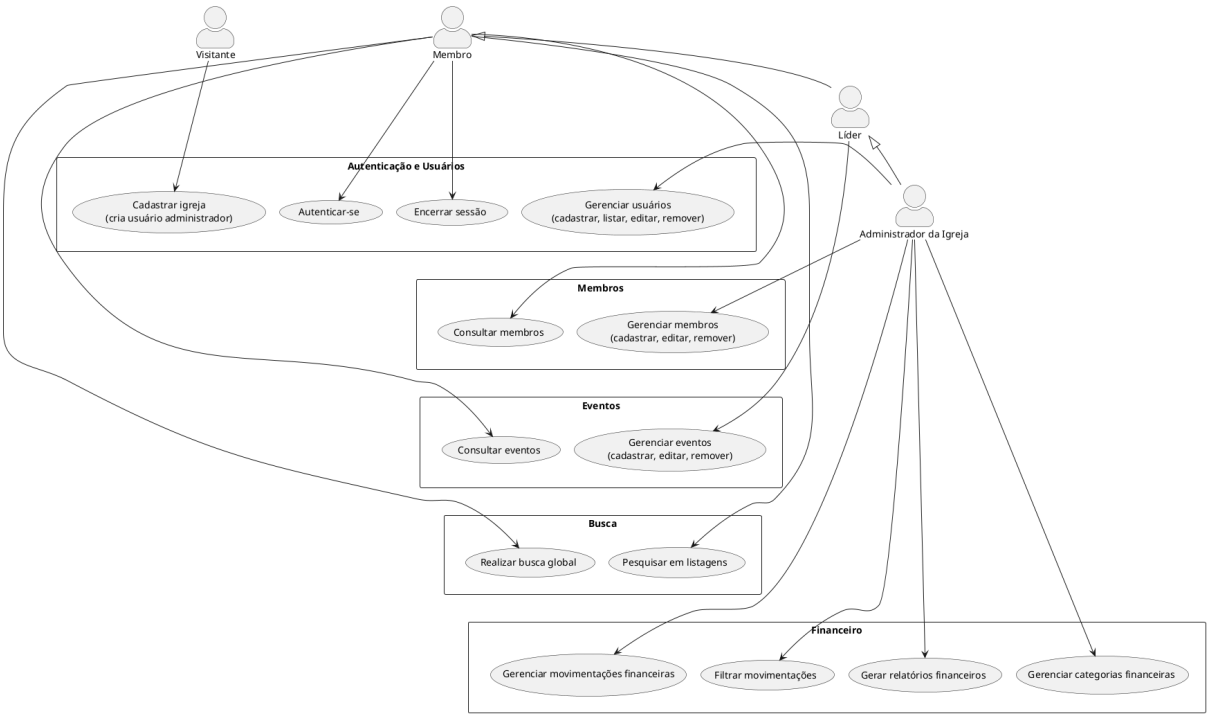
3 ANÁLISE E DESIGN

Este capítulo apresenta os artefatos de modelagem que fundamentaram a construção do Domus. A Seção 3.2 descreve o diagrama de classes do domínio; a Seção 3.3 apresenta o modelo lógico da base de dados, sua criação física e o dicionário de dados; e a Seção 3.4 descreve o ambiente de desenvolvimento utilizado.

3.1 Diagrama de Casos de Uso

O diagrama de casos de uso representa as funcionalidades oferecidas pelo Domus sob a perspectiva dos perfis de usuário descritos na Seção 2.2. A Figura 1 apresenta o modelo, no qual os atores `Líder` e `Administrador da Igreja` são representados por meio de generalização (herança de ator): `Líder` herda todos os casos de uso de `Membro` e acrescenta o gerenciamento de eventos; `Administrador da Igreja` herda todos os casos de uso de `Líder` e acrescenta o gerenciamento de usuários, de membros e do módulo financeiro. Essa notação reflete diretamente a matriz de permissões implementada na camada de segurança do backend.

Figura 1 – Diagrama de casos de uso do Domus.



Fonte: Elaborado pelo autor (2026).

O ator `Visitante` representa uma pessoa sem vínculo prévio com o sistema, responsável exclusivamente pelo caso de uso de cadastro de uma nova igreja (RF-01), fluxo que cria simulta-

neamente a igreja e seu usuário administrador inicial. Os demais casos de uso pressupõem um usuário já autenticado, motivo pelo qual a autenticação é atribuída ao ator `Membro` — perfil de base da hierarquia — e herdada pelos demais.

3.2 Diagrama de Classes

O diagrama de classes representa as entidades do domínio do Domus, seus atributos e os relacionamentos entre elas. A Figura 2 apresenta o modelo, no qual se destaca a entidade `Igreja` como elemento central da arquitetura multilocatária: todas as demais entidades do domínio referenciam a igreja à qual pertencem, o que materializa o isolamento lógico de dados descrito no Capítulo 4.

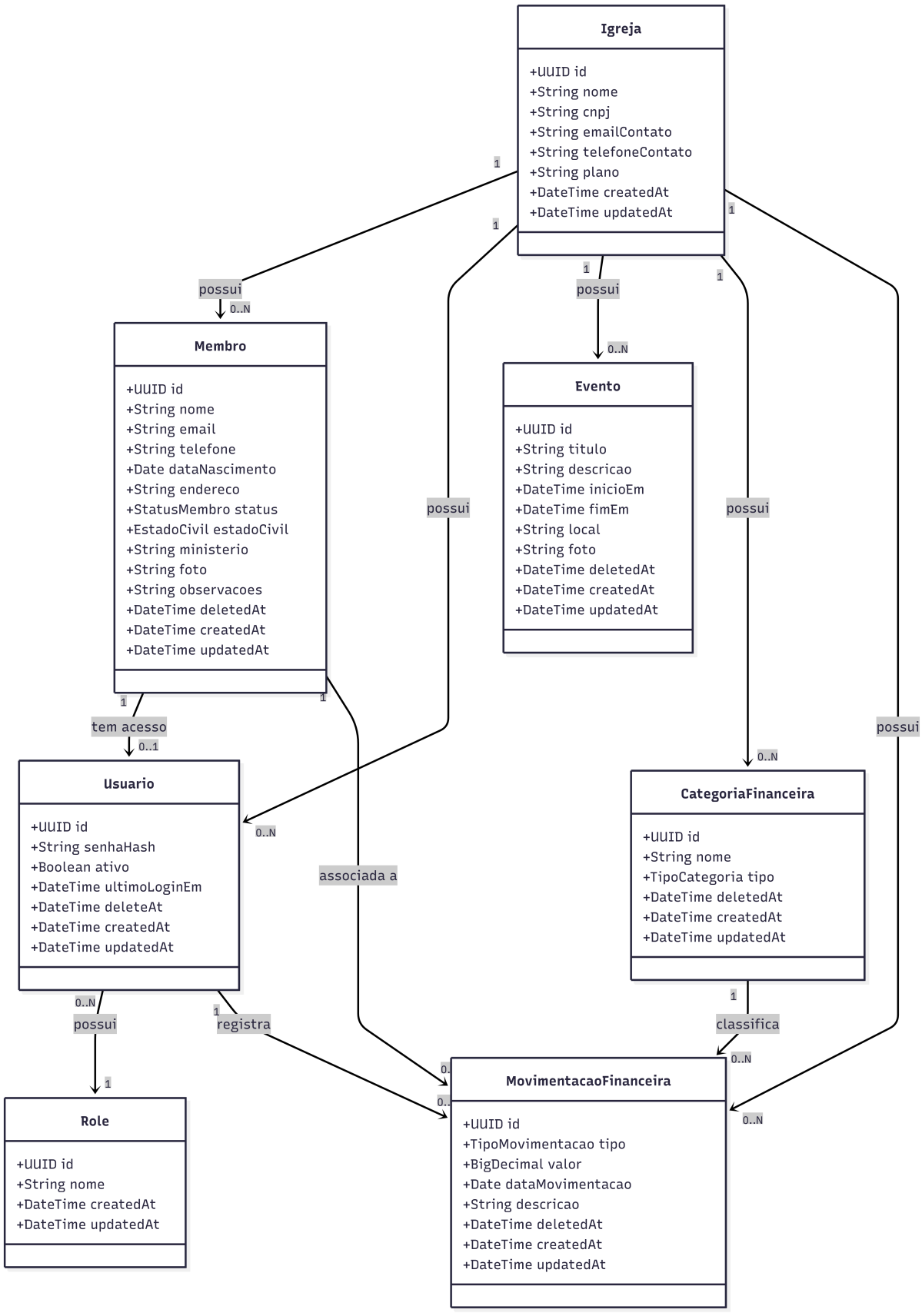
O diagrama concentra-se nos atributos das entidades e em seus relacionamentos. Essa escolha reflete a arquitetura adotada, na qual as entidades atuam como representações do domínio e a lógica de negócio reside na camada de serviços, conforme descrito no Capítulo 4.

A entidade `Igreja` mantém um relacionamento de um-para-muitos com `Membro`, `Usuario`, `Evento`, `CategoriaFinanceira` e `MovimentacaoFinanceira`, refletindo o fato de que cada organização possui seu próprio conjunto de dados, isolado das demais.

Destaca-se a relação entre `Membro` e `Usuario`. Um membro representa uma pessoa vinculada à igreja, ao passo que um usuário representa uma credencial de acesso ao sistema. Conforme a decisão de modelagem discutida no capítulo anterior, todo usuário está obrigatoriamente associado a um membro, do qual derivam informações como nome e e-mail — por isso a entidade `Usuario` expõe os métodos `getNome()` e `getEmail()`, que recuperam esses dados do membro correspondente. O relacionamento é de um-para-no-máximo-um: cada membro pode ter, no máximo, um usuário associado, e nem todo membro possui acesso ao sistema. Cada usuário, por sua vez, está vinculado a exatamente um perfil de acesso (`Role`), que determina suas permissões.

A entidade `MovimentacaoFinanceira` concentra os relacionamentos do módulo financeiro. Cada movimentação é classificada por uma `CategoriaFinanceira` e registrada por um `Usuario` (o autor do lançamento). Opcionalmente, uma movimentação pode estar associada a um `Membro`, nos casos em que o lançamento se refere a um contribuinte ou beneficiário específico. A movimentação também mantém referência ao usuário responsável por sua última atualização, permitindo a rastreabilidade das alterações.

Figura 2 – Diagrama de classes do Domus.



Fonte: Elaborado pelo autor (2026).

3.3 Modelo de Dados

3.3.1 Modelo Lógico da Base de Dados

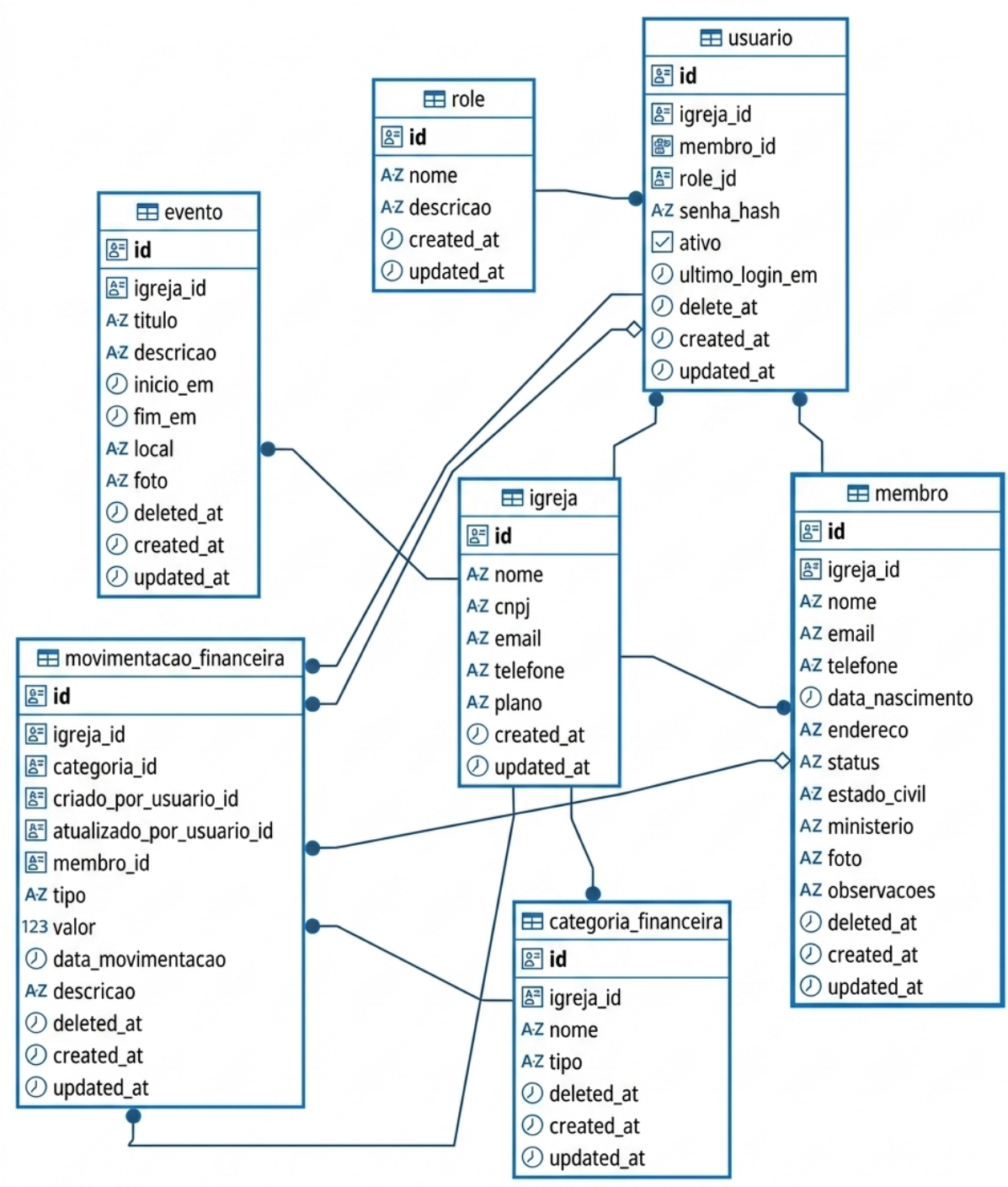
O modelo lógico traduz o domínio para uma estrutura relacional, definindo tabelas, colunas, chaves primárias e estrangeiras (ELMASRI; NAVATHE, 2018). A base de dados do Domus foi implementada no sistema gerenciador de banco de dados PostgreSQL, com o esquema versionado por meio da ferramenta Flyway, que permite a evolução controlada e reproduzível da estrutura entre ambientes.

A Figura 3 apresenta o Diagrama Entidade-Relacionamento (DER) da base de dados, obtido por engenharia reversa do esquema físico implementado. O diagrama contempla as sete tabelas de domínio (`igreja`, `role`, `membro`, `usuario`, `evento`, `categoria_financeira` e `movimentacao_financeira`); a tabela técnica `outbox`, por não representar uma entidade do domínio, foi omitida do diagrama pelo mesmo critério já adotado no diagrama de classes (Seção 3.2).

Além das entidades de domínio apresentadas no diagrama de classes, o modelo físico inclui a tabela `outbox`, de natureza técnica, responsável por registrar os eventos de alteração das entidades para posterior sincronização com o índice de busca, conforme o padrão *transactional outbox* descrito no Capítulo 4. Por não representar um conceito do domínio, essa tabela não figura no diagrama de classes, integrando apenas o modelo físico.

Todas as chaves primárias adotam o tipo `UUID` (*Universally Unique Identifier*), gerado pelo próprio banco. As tabelas do domínio incorporam uma coluna `igreja_id`, que referencia a tabela `igreja` e viabiliza o isolamento lógico multilocatário. As entidades sujeitas a exclusão lógica (DATE, 2004) possuem uma coluna de data de exclusão (`deleted_at` ou `delete_at`), que, quando preenchida, indica que o registro foi arquivado.

Figura 3 – Diagrama Entidade-Relacionamento (DER) da base de dados do Domus.



Fonte: Elaborado pelo autor (2026), a partir de engenharia reversa do esquema físico via DBeaver.

3.3.2 Criação Física do Modelo de Dados

A criação física das tabelas foi realizada por meio de *scripts* de migração SQL executados pelo Flyway. O Quadro 3 relaciona as tabelas que compõem a base de dados e sua respectiva finalidade.

Quadro 3 – Tabelas da base de dados do Domus.

Tabela	**Finalidade**
`igreja`	Armazena as organizações (locatárias) atendidas pela plataforma.
`role`	Define os perfis de acesso ao sistema (Administrador, Líder e Membro).
`membro`	Armazena as pessoas vinculadas a cada igreja.
`usuario`	Armazena as credenciais de acesso, vinculadas a um membro e a um perfil.
`evento`	Registra os eventos da agenda de cada igreja.
`categoria_financeira`	Classifica as movimentações financeiras por tipo (entrada, saída ou ambos).
`movimentacao_financeira`	Registra as movimentações financeiras de entrada e saída.
`outbox`	Registra os eventos de alteração para sincronização com o índice de busca.

Fonte: Elaborado pelo autor (2026).

Para assegurar a integridade dos dados, o modelo emprega restrições de diversos tipos: chaves estrangeiras que garantem a consistência referencial entre as tabelas; restrições de unicidade — como a que impede o cadastro de dois membros com o mesmo e-mail, ou de duas categorias com o mesmo nome em uma mesma igreja; e restrições de verificação (*check*), que validam, por exemplo, se o tipo de uma movimentação é `ENTRADA` ou `SAIDA` e se seu valor é positivo. Adicionalmente, foram criados índices sobre as colunas mais utilizadas em consultas — notadamente as que combinam o identificador da igreja com outros critérios de filtragem —, com o objetivo de otimizar o desempenho das leituras.

3.3.3 Dicionário de Dados

O dicionário de dados detalha os atributos das entidades do domínio. As colunas de controle comuns a todas as tabelas — `created_at` e `updated_at`, que registram os instantes de criação e de última atualização — foram omitidas dos quadros individuais para evitar repetição. A tabela técnica `outbox`, por não representar uma entidade do domínio, não é detalhada nesta seção.

O Quadro 4 descreve a tabela `igreja`.

Quadro 4 – Dicionário de dados — tabela `igreja`.

Campo	**Tipo**	**Descrição**
`id`	UUID	Identificador único da igreja (chave primária).
`nome`	VARCHAR(255)	Nome da igreja. Obrigatório.
`cnpj`	VARCHAR(18)	Cadastro Nacional da Pessoa Jurídica. Opcional.
`email`	VARCHAR(255)	E-mail de contato da igreja. Obrigatório.
`telefone`	VARCHAR(20)	Telefone de contato. Opcional.
`plano`	VARCHAR(50)	Plano de assinatura da igreja. Opcional.

Fonte: Elaborado pelo autor (2026).

O Quadro 5 descreve a tabela `membro`.

Quadro 5 – Dicionário de dados — tabela `membro`.

Campo	**Tipo**	**Descrição**
`id`	UUID	Identificador único do membro (chave primária).
`igreja_id`	UUID	Referência à igreja proprietária (chave estrangeira).
`nome`	VARCHAR(255)	Nome do membro. Obrigatório.
`email`	VARCHAR(255)	E-mail do membro. Opcional; único, quando informado.
`telefone`	VARCHAR(20)	Telefone do membro. Opcional.
`data_nascimento`	DATE	Data de nascimento. Opcional.
`endereco`	VARCHAR(500)	Endereço do membro. Opcional.
`status`	VARCHAR(20)	Situação do membro (ativo, inativo ou visitante).
`estado_civil`	VARCHAR(20)	Estado civil do membro. Opcional.
`ministerio`	VARCHAR(255)	Ministério ao qual o membro está vinculado. Opcional.
`foto`	VARCHAR(500)	Caminho para a foto do membro. Opcional.
`observacoes`	TEXT	Observações gerais. Opcional.
`deleted_at`	TIMESTAMP	Data de arquivamento (exclusão lógica).

Fonte: Elaborado pelo autor (2026).

O Quadro 6 descreve a tabela `usuario`.

Quadro 6 – Dicionário de dados — tabela `usuario`.

Campo	**Tipo**	**Descrição**
`id`	UUID	Identificador único do usuário (chave primária).
`igreja_id`	UUID	Referência à igreja proprietária (chave estrangeira).
`membro_id`	UUID	Referência ao membro correspondente (chave estrangeira, única).
`role_id`	UUID	Referência ao perfil de acesso (chave estrangeira).
`senha_hash`	VARCHAR(255)	Senha do usuário, armazenada com *hash* bcrypt.
`ativo`	BOOLEAN	Indica se o acesso está ativo.
`ultimo_login_em`	TIMESTAMP	Data e hora do último acesso.
`delete_at`	TIMESTAMP	Data de arquivamento (exclusão lógica).

Fonte: Elaborado pelo autor (2026).

O Quadro 7 descreve a tabela `role`.

Quadro 7 – Dicionário de dados — tabela `role`.

Campo	**Tipo**	**Descrição**
`id`	UUID	Identificador único do perfil (chave primária).
`nome`	VARCHAR(50)	Nome do perfil de acesso. Único.
`descricao`	VARCHAR(255)	Descrição do perfil. Opcional.

Fonte: Elaborado pelo autor (2026).

O Quadro 8 descreve a tabela `evento`.

Quadro 8 – Dicionário de dados — tabela `evento`.

Campo	**Tipo**	**Descrição**
`id`	UUID	Identificador único do evento (chave primária).
`igreja_id`	UUID	Referência à igreja proprietária (chave estrangeira).
`titulo`	VARCHAR(255)	Título do evento. Obrigatório.
`descricao`	TEXT	Descrição do evento. Opcional.

Campo	Tipo	Descrição
`inicio_em`	TIMESTAMP	Data e hora de início. Obrigatório.
`fim_em`	TIMESTAMP	Data e hora de término. Opcional.
`local`	VARCHAR(255)	Local de realização. Opcional.
`foto`	VARCHAR(500)	Caminho para a imagem do evento. Opcional.
`deleted_at`	TIMESTAMP	Data de arquivamento (exclusão lógica).

Fonte: Elaborado pelo autor (2026).

O Quadro 9 descreve a tabela `categoria_financeira`.

Quadro 9 – Dicionário de dados — tabela `categoria_financeira`.

Campo	Tipo	Descrição
`id`	UUID	Identificador único da categoria (chave primária).
`igreja_id`	UUID	Referência à igreja proprietária (chave estrangeira).
`nome`	VARCHAR(255)	Nome da categoria. Único por igreja.
`tipo`	VARCHAR(20)	Tipo permitido: entrada, saída ou ambos.
`deleted_at`	TIMESTAMP	Data de arquivamento (exclusão lógica).

Fonte: Elaborado pelo autor (2026).

O Quadro 10 descreve a tabela `movimentacao_financeira`.

Quadro 10 – Dicionário de dados — tabela `movimentacao_financeira`.

Campo	Tipo	Descrição
`id`	UUID	Identificador único da movimentação (chave primária).
`igreja_id`	UUID	Referência à igreja proprietária (chave estrangeira).
`categoria_id`	UUID	Referência à categoria (chave estrangeira).
`criado_por_usuario_id`	UUID	Usuário que registrou a movimentação (chave estrangeira).
`atualizado_por_usuario_id`	UUID	Usuário que realizou a última alteração (chave estrangeira). Opcional.

Campo	Tipo	Descrição
`membro_id`	UUID	Membro associado à movimentação (chave estrangeira). Opcional.
`tipo`	VARCHAR(20)	Tipo da movimentação: entrada ou saída.
`valor`	NUMERIC(15,2)	Valor da movimentação. Deve ser positivo.
`data_movimentacao`	DATE	Data em que a movimentação ocorreu.
`descricao`	TEXT	Descrição da movimentação. Opcional.
`deleted_at`	TIMESTAMP	Data de arquivamento (exclusão lógica).

Fonte: Elaborado pelo autor (2026).

3.4 Ambiente de Desenvolvimento

O desenvolvimento do Domus foi conduzido em ambiente organizado em torno de ferramentas de código aberto e amplamente adotadas pela indústria de software. O Quadro 11 sintetiza as principais ferramentas empregadas e sua finalidade.

Quadro 11 – Ferramentas do ambiente de desenvolvimento.

Ferramenta	Finalidade
IntelliJ IDEA	Ambiente de desenvolvimento integrado para o backend em Java.
Visual Studio Code	Editor de código para o frontend em Next.js e TypeScript.
Git e GitHub	Versionamento de código e hospedagem dos repositórios.
Docker	Execução dos serviços de apoio (Redis e Elasticsearch) em contêineres.
PostgreSQL	Sistema gerenciador do banco de dados relacional.
Redis	Serviço de cache em memória.
Elasticsearch	Mecanismo de busca textual.
Insomnia	Cliente para teste e validação manual das requisições à API REST.

Fonte: Elaborado pelo autor (2026).

Para reproduzir com fidelidade o ambiente de execução e simplificar a configuração dos serviços de apoio, o Redis e o Elasticsearch foram executados em contêineres Docker, o que garante a consistência do ambiente entre diferentes máquinas de desenvolvimento. A organização do trabalho adotou práticas reconhecidas pela indústria, incluindo o versionamento com Git e o fluxo de ramificações (*branches*) por funcionalidade, conforme descrito no Capítulo 1.

4 RESULTADOS OBTIDOS

Este capítulo documenta o desenvolvimento do Domus. A Seção 4.1 apresenta a arquitetura geral da solução e a divisão em camadas; a Seção 4.2 relaciona e justifica as tecnologias adotadas; a Seção 4.3 discute as decisões de projeto mais relevantes tomadas durante o desenvolvimento; e a Seção 4.4 apresenta as principais telas resultantes do sistema.

4.1 Arquitetura do Sistema

O Domus adota uma arquitetura cliente-servidor com separação clara entre a aplicação de interface (*frontend*) e a interface de programação de aplicações (*backend*), comunicando-se por meio de uma API REST sobre HTTP (FIELDING, 2000). Essa separação permite que as duas camadas evoluam de forma independente e viabiliza, em evoluções futuras, a reutilização da mesma API por outros clientes, como um aplicativo móvel.

No lado do servidor, adotou-se uma organização em camadas com responsabilidades bem definidas, seguindo o princípio da separação de interesses (MARTIN, 2019). A camada de *controllers* recebe as requisições HTTP, valida os dados de entrada e delega o processamento; a camada de *services* concentra as regras de negócio e coordena as operações; e a camada de *repositories* encapsula o acesso ao banco de dados. Como política de projeto, a camada de serviço nunca expõe diretamente as entidades de persistência, retornando objetos de transferência de dados (*Data Transfer Objects* — DTOs) (FOWLER, 2002), o que desacopla o modelo interno de dados do contrato público da API.

Internamente, o backend é estruturado de forma modular, com cada módulo correspondendo a um domínio funcional do sistema — autenticação, usuários, membros, eventos, financeiro, busca e infraestrutura de sincronização. Essa modularização favorece a manutenibilidade e a coesão, mantendo próximos os artefatos que tratam de um mesmo assunto.

A persistência de dados é realizada em um banco relacional PostgreSQL, complementado por dois componentes de apoio: um servidor Redis, utilizado como cache em memória para reduzir a latência de leituras frequentes; e um mecanismo de busca Elasticsearch, responsável pela funcionalidade de busca avançada. A consistência entre o banco relacional — fonte da verdade do sistema — e o índice de busca é garantida de forma assíncrona, conforme detalhado na Seção 4.3.

4.2 Tecnologias Adotadas

A seleção das tecnologias priorizou ferramentas maduras, amplamente adotadas pela indústria e adequadas aos requisitos do projeto. O Quadro 12 sintetiza as principais tecnologias

empregadas em cada camada.

Quadro 12 – Principais tecnologias utilizadas no Domus.

Camada	**Tecnologia**	**Finalidade**
Backend	Java 21	Linguagem de programação principal.
Backend	Spring Boot	Framework para construção da API REST e injeção de dependências.
Backend	Spring Security	Autenticação, autorização e controle de acesso.
Persistência	PostgreSQL	Banco de dados relacional (fonte da verdade).
Persistência	Spring Data JPA	Mapeamento objeto-relacional e repositórios.
Persistência	Flyway	Versionamento e migração do esquema do banco.
Cache	Redis	Cache em memória para leituras frequentes.
Busca	Elasticsearch	Motor de busca textual e tolerante a erros.
Segurança	JSON Web Token	Tokens de autenticação sem estado.
Segurança	bcrypt	Algoritmo de hash para armazenamento de senhas.
Frontend	Next.js	Framework React para a interface web.
Frontend	TypeScript	Linguagem com tipagem estática.
Frontend	CSS Modules	Estilização com escopo por componente.
Frontend	TanStack Query	Gerenciamento de estado assíncrono e cache no cliente.
Frontend	React Hook Form + Zod	Gerenciamento e validação de formulários.

Fonte: Elaborado pelo autor (2026).

No backend, a escolha por Java 21 com Spring Boot deve-se à robustez do ecossistema, ao suporte nativo a injeção de dependênciasGAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Padrões de projeto** (GAMMA et al., 2000): soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman, 2000. e à ampla adoção do framework em ambientes corporativos. O Spring Security oferece uma base sólida para a implementação de autenticação e autorização, integrando-se ao mecanismo de tokens JWT adotado. O versionamento do esquema do banco por

meio do Flyway garante que a estrutura das tabelas evolua de forma controlada e reproduzível entre ambientes.

No frontend, o Next.js foi escolhido por organizar a aplicação em torno de um sistema de rotas baseado em arquivos e por oferecer recursos modernos de renderização. O uso de TypeScript adiciona tipagem estática, reduzindo classes de erros em tempo de desenvolvimento. A biblioteca TanStack Query gerencia o ciclo de vida das requisições ao backend, incluindo cache, revalidação e estados de carregamento, enquanto a combinação de React Hook Form com Zod garante validação declarativa e consistente dos formulários.

4.3 Decisões de Desenvolvimento

Esta seção discute as decisões de projeto de maior impacto técnico tomadas ao longo do desenvolvimento, apresentando o problema enfrentado, a alternativa adotada e sua justificativa.

4.3.1 Isolamento Lógico Multilocatário

Por tratar-se de um sistema do tipo SaaS que atende múltiplas igrejas a partir de uma mesma instância, o isolamento dos dados entre organizações é um requisito central. Adotou-se a estratégia de isolamento lógico por meio de um identificador de igreja (*tenant*) (AMAZON WEB SERVICES, 2023) presente em todas as entidades do domínio. Diferentemente de abordagens que empregam um banco de dados por cliente, o isolamento lógico mantém todos os dados em uma base compartilhada, distinguindo-os por esse identificador — estratégia adequada ao porte do projeto e economicamente mais viável para o público-alvo.

A decisão de projeto mais relevante nesse ponto refere-se à origem do identificador da igreja. Em nenhuma hipótese esse identificador é aceito a partir do corpo da requisição enviada pelo cliente; ele é sempre extraído do token de autenticação do usuário. Essa escolha é deliberada e de natureza defensiva: caso o identificador viesse do cliente, um usuário mal-intencionado poderia alterá-lo para acessar dados de outra igreja. Ao derivá-lo do token — gerado e assinado pelo servidor no momento do login —, garante-se que cada usuário opere exclusivamente sobre os dados da própria organização. Essa validação é aplicada de forma consistente na camada de serviço, em todas as operações de leitura e escrita.

4.3.2 Modelagem da Relação entre Usuário e Membro

Durante o desenvolvimento, identificou-se uma decisão de modelagem importante quanto à relação entre as entidades Usuário e Membro. Um membro representa uma pessoa vinculada à igreja, ao passo que um usuário representa uma credencial de acesso ao sistema. A questão central era determinar como esses dois conceitos deveriam se relacionar, evitando a duplicação de informações como nome e e-mail.

Optou-se por estabelecer que todo usuário do sistema é, necessariamente, um membro da igreja. A entidade Usuário passou a não armazenar dados pessoais próprios, mantendo apenas as informações de credencial e um vínculo obrigatório com um registro de Membro, de onde derivam atributos como nome e e-mail. Essa modelagem elimina a duplicação de dados pessoais, assegura uma fonte única de verdade para as informações de cada pessoa e simplifica a manutenção: a atualização do nome de um membro reflete-se automaticamente em seu usuário correspondente, sem necessidade de sincronização manual entre as duas entidades.

4.3.3 Estratégia de Cache

Determinadas operações de leitura — como a listagem de membros ou de movimentações financeiras — são executadas com frequência e tendem a retornar os mesmos resultados entre requisições sucessivas. Para reduzir a carga sobre o banco de dados e diminuir a latência dessas consultas, adotou-se um cache em memória baseado em Redis (REDIS LTD., 2024).

O principal desafio no uso de cache é a invalidação: dados em cache podem tornar-se desatualizados quando a informação de origem é alterada. Para tratar essa questão, o sistema invalida automaticamente as entradas de cache relacionadas a uma entidade sempre que ela é criada, atualizada ou removida, garantindo que leituras subsequentes reflitam o estado atual dos dados. O cache é segmentado pelo identificador da igreja, de modo que a atualização dos dados de uma organização não afeta o cache das demais.

4.3.4 Busca Avançada e Sincronização por Transactional Outbox

A funcionalidade de busca representa uma das contribuições técnicas mais significativas do projeto. Em vez de restringir a pesquisa a consultas convencionais ao banco relacional, adotou-se o motor de busca Elasticsearch, que oferece recursos avançados de busca textual (ELASTIC, 2024), incluindo tolerância a erros de digitação (busca aproximada) e correspondência por prefixo, adequados a uma experiência de busca instantânea, conforme o usuário digita.

A adoção de um mecanismo de busca separado do banco relacional introduz o desafio de manter os dois repositórios sincronizados: toda alteração no banco — criação, atualização ou remoção de uma entidade — precisa refletir-se no índice de busca. Uma abordagem direta, que gravasse simultaneamente no banco e no índice dentro da mesma operação, apresentaria fragilidade: caso a gravação no índice falhasse após a confirmação no banco, os dois repositórios ficariam inconsistentes.

Para tratar esse problema de forma robusta, adotou-se o padrão *transactional outbox* (FOWLER, 2021). Nesse padrão, quando uma entidade é alterada, um registro descrevendo o evento é gravado em uma tabela auxiliar (*outbox*), dentro da mesma transação que altera a entidade. Como ambas as gravações ocorrem na mesma transação do banco relacional, garante-se atomicidade: ou as duas são confirmadas, ou nenhuma é. Um processo executado periodicamente lê os eventos pendentes na tabela *outbox* e os aplica ao índice de busca, marcando-os como

processados. Essa estratégia assegura consistência eventual entre o banco e o índice — ainda que exista um pequeno intervalo até a propagação, nenhuma alteração é perdida, mesmo diante de falhas temporárias no mecanismo de busca.

A busca incorpora, ainda, duas camadas de proteção alinhadas ao modelo de segurança do sistema. A primeira é o isolamento multilocatário: o identificador da igreja, extraído do token, é aplicado obrigatoriamente como filtro em todas as consultas ao índice, de modo que uma busca jamais retorne dados de outra organização. A segunda é o controle por perfil: o conjunto de entidades consultadas é determinado pelo papel do usuário, de forma que dados financeiros e de usuários — restritos ao Administrador da Igreja — não sejam retornados nas buscas realizadas pelos perfis Líder e Membro.

4.3.5 Exclusão Lógica de Registros

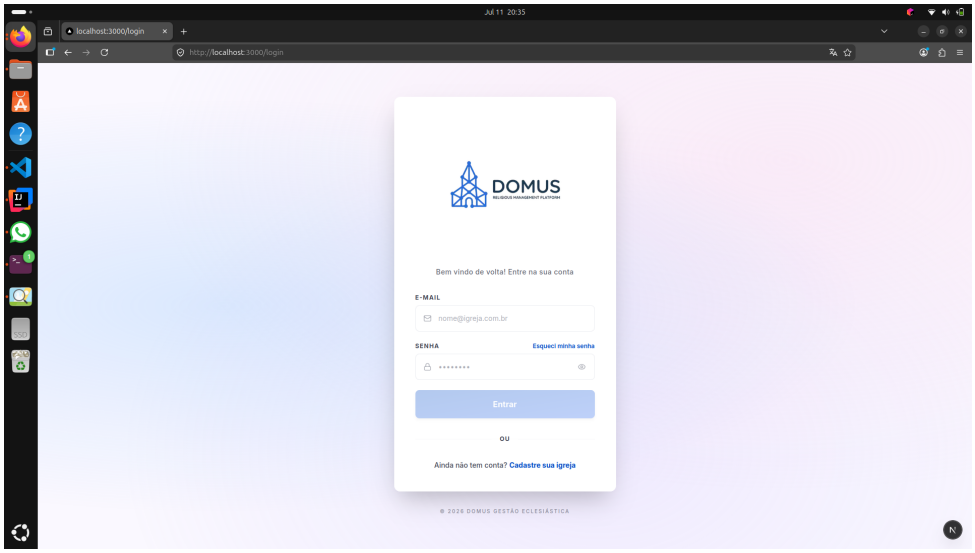
Para preservar o histórico e a integridade referencial dos dados, o sistema adota a exclusão lógica (*soft delete*) em suas entidades. Em vez de remover fisicamente um registro do banco, a operação de remoção apenas o marca como excluído por meio de um campo de data, e as consultas subsequentes passam a desconsiderá-lo automaticamente. Essa abordagem evita a perda definitiva de informações associadas a outros registros — por exemplo, o histórico de movimentações financeiras vinculadas a um membro — e possibilita, em evoluções futuras, a restauração de cadastros arquivados. Cabe registrar que, do ponto de vista da Lei Geral de Proteção de Dados, a exclusão definitiva de dados pessoais é tratada como um refinamento previsto para etapas posteriores, distinto da exclusão lógica aqui descrita.

4.4 Telas do Sistema

Esta seção apresenta as principais interfaces do Domus, ilustrando o resultado visual do desenvolvimento e evidenciando o atendimento aos requisitos funcionais. Cada figura corresponde a uma funcionalidade do sistema, permitindo relacionar as interfaces às capacidades descritas na Seção 2.3.

A Figura 4 apresenta a tela de autenticação, ponto de entrada do sistema, na qual o usuário informa e-mail e senha para acessar a plataforma. A interface disponibiliza ainda o acesso à recuperação de senha e ao cadastro de uma nova igreja, atendendo aos requisitos de autenticação e de criação de conta.

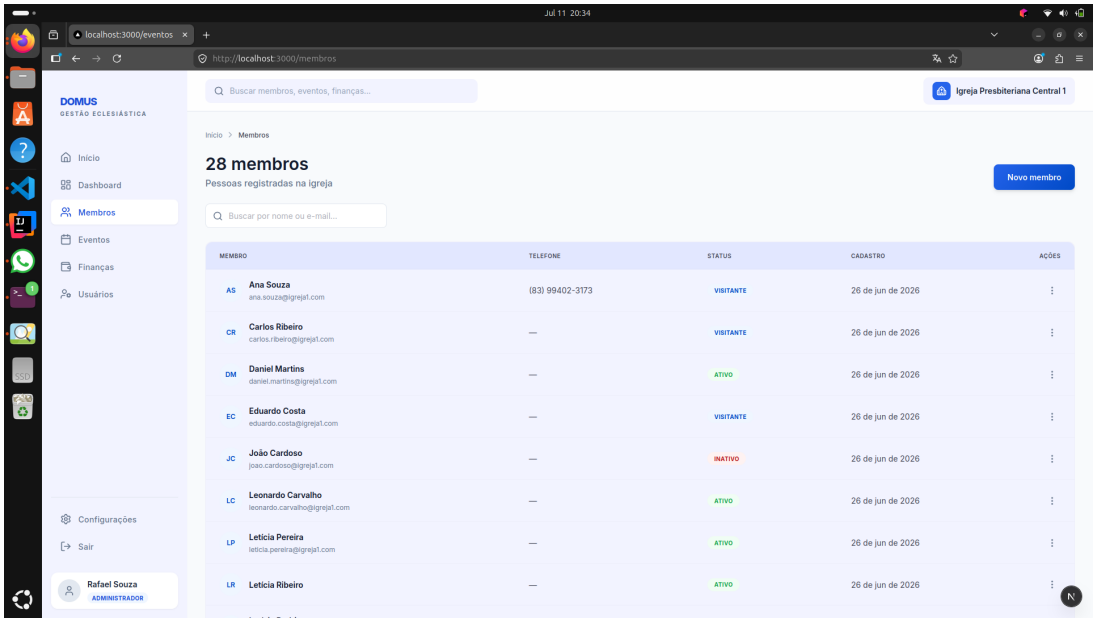
Figura 4 – Tela de autenticação do Domus.



Fonte: Elaborado pelo autor (2026).

A Figura 5 apresenta a listagem de membros, organizada em formato tabular com nome, telefone, situação e data de cadastro de cada pessoa. No topo da aplicação encontra-se a barra de busca global, disponível de forma transversal a partir de qualquer módulo, e à esquerda o menu de navegação lateral, que identifica o usuário autenticado e seu perfil de acesso.

Figura 5 – Listagem de membros com a barra de busca global.

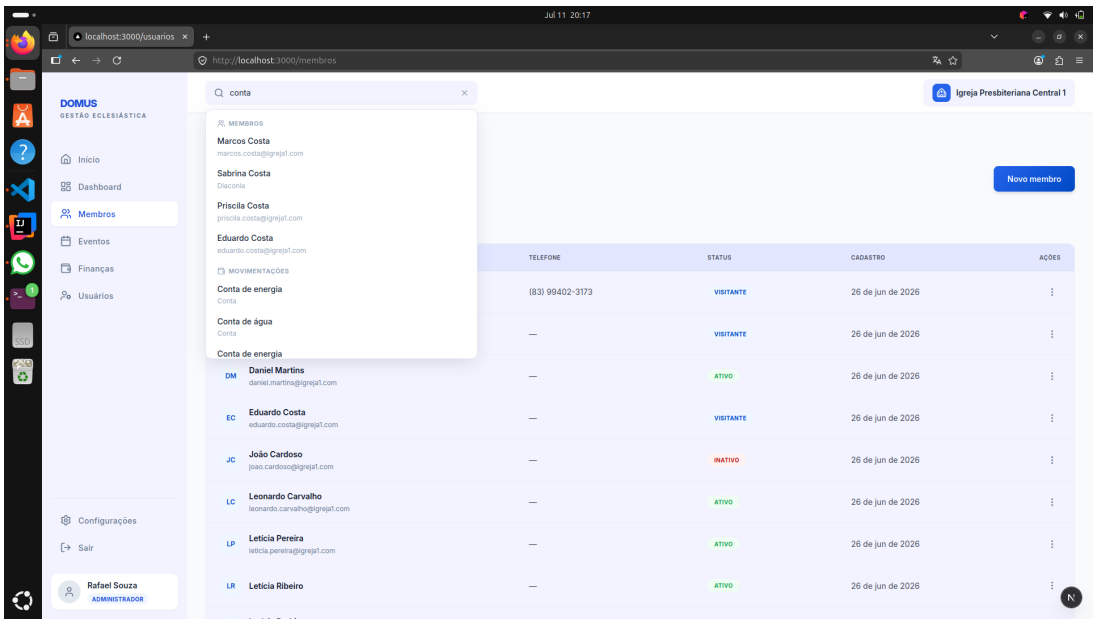


Fonte: Elaborado pelo autor (2026).

A Figura 6 apresenta o comportamento da busca global em funcionamento. À medida que o usuário digita, os resultados são exibidos em um painel suspenso e organizados por tipo de entidade — no exemplo, membros e movimentações financeiras. A busca é tolerante a erros de

digitação e retorna apenas resultados das entidades permitidas ao perfil do usuário e pertencentes à sua igreja.

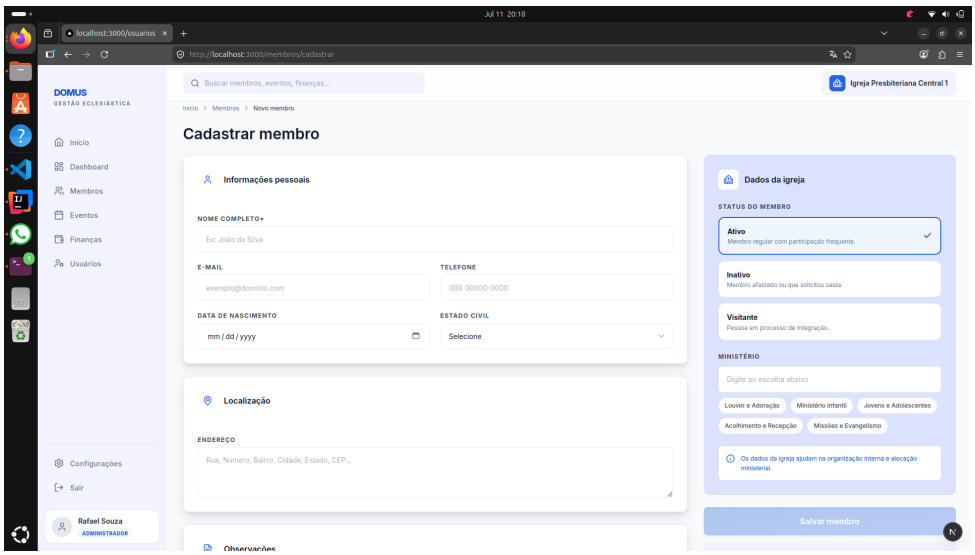
Figura 6 – Resultados da busca global agrupados por tipo de entidade.



Fonte: Elaborado pelo autor (2026).

A Figura 7 apresenta o formulário de cadastro e edição de membro, organizado em seções que agrupam as informações pessoais, a localização e os dados relativos à igreja, como situação do membro e ministério. Os campos obrigatórios são sinalizados e submetidos à validação antes do envio.

Figura 7 – Formulário de cadastro e edição de membro.

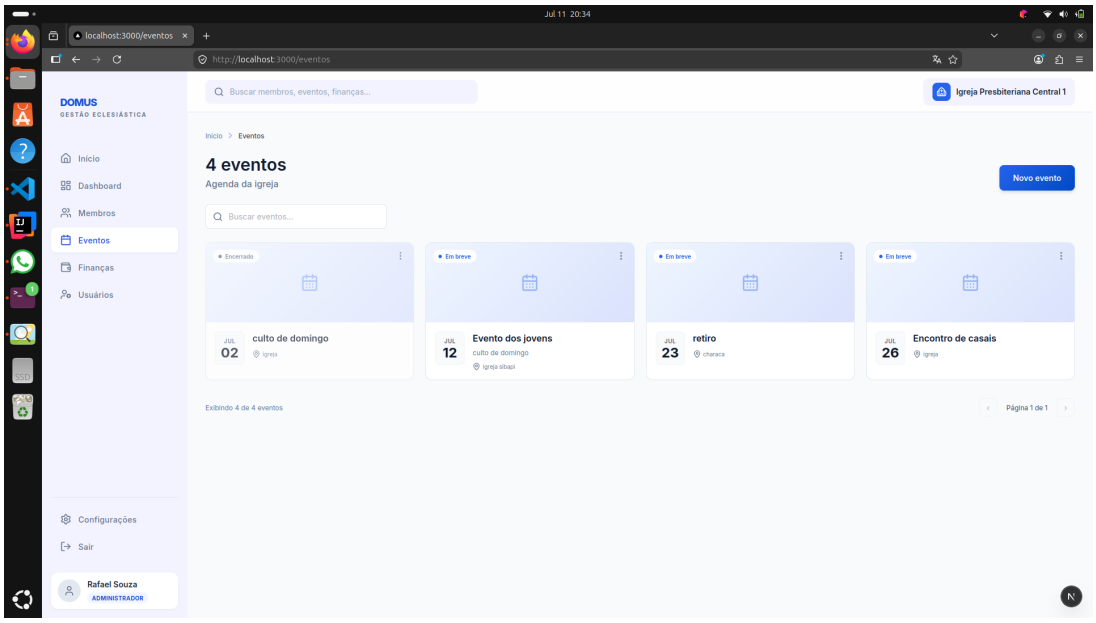


Fonte: Elaborado pelo autor (2026).

A Figura 8 apresenta a listagem de eventos em formato de *cards*, exibindo a agenda da

igreja. Cada *card* destaca a data, o título, o local e a situação do evento, sinalizando visualmente se ele já foi encerrado ou está por ocorrer.

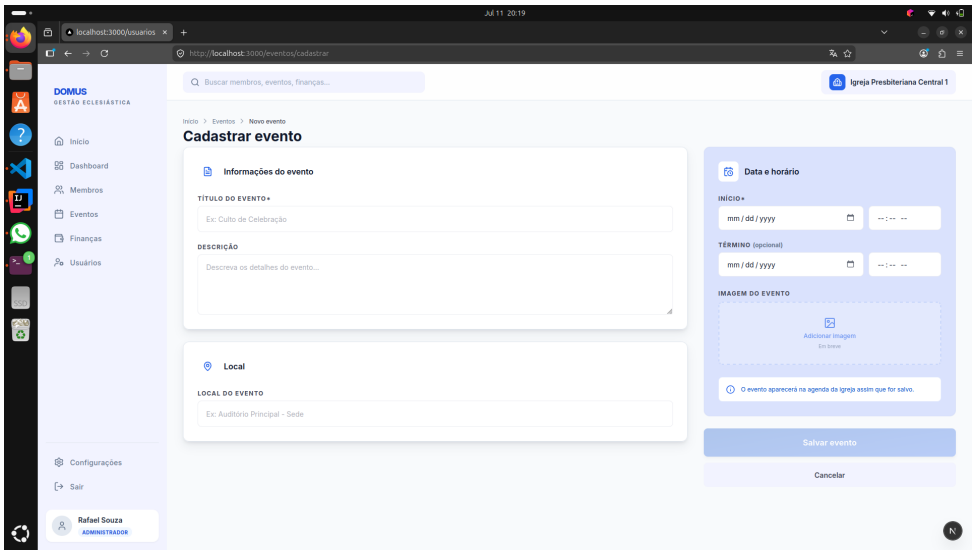
Figura 8 – Listagem de eventos em formato de cards.



Fonte: Elaborado pelo autor (2026).

A Figura 9 apresenta o formulário de cadastro de evento, no qual são informados o título, a descrição, o local e o período de realização, com data e horário de início e de término. Um aviso indica ao usuário que o evento passa a integrar a agenda da igreja assim que é salvo.

Figura 9 – Formulário de cadastro de evento.

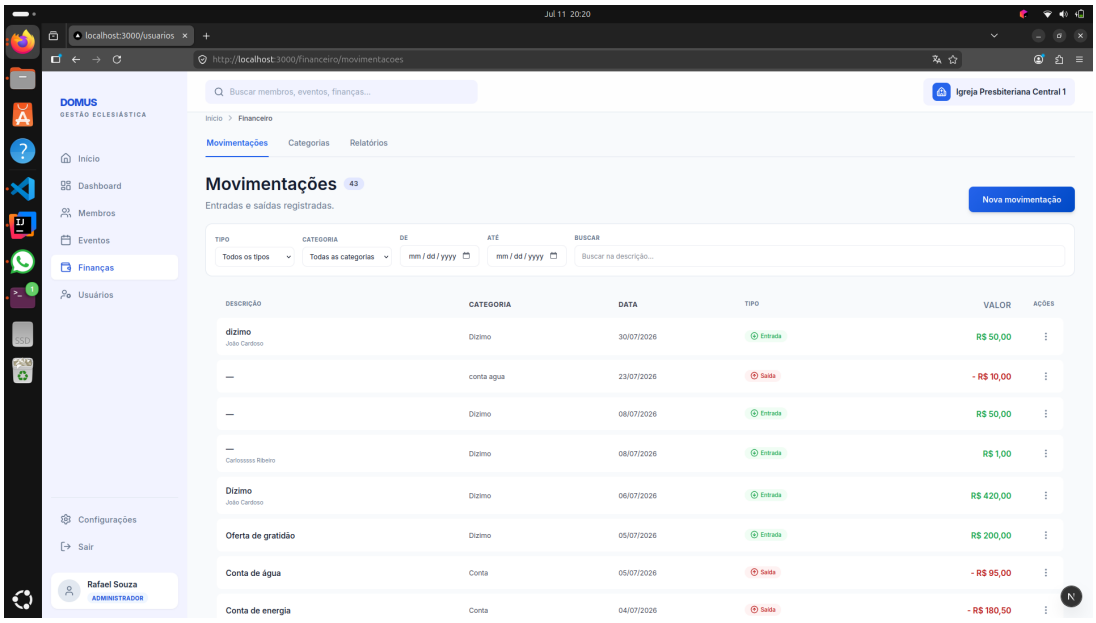


Fonte: Elaborado pelo autor (2026).

A Figura 10 apresenta a listagem de movimentações financeiras, com os filtros por tipo,

categoria e período dispostos no topo. Cada lançamento exibe descrição, categoria, data, tipo e valor, com destaque de cor distinto para entradas e saídas, facilitando a leitura do fluxo de caixa.

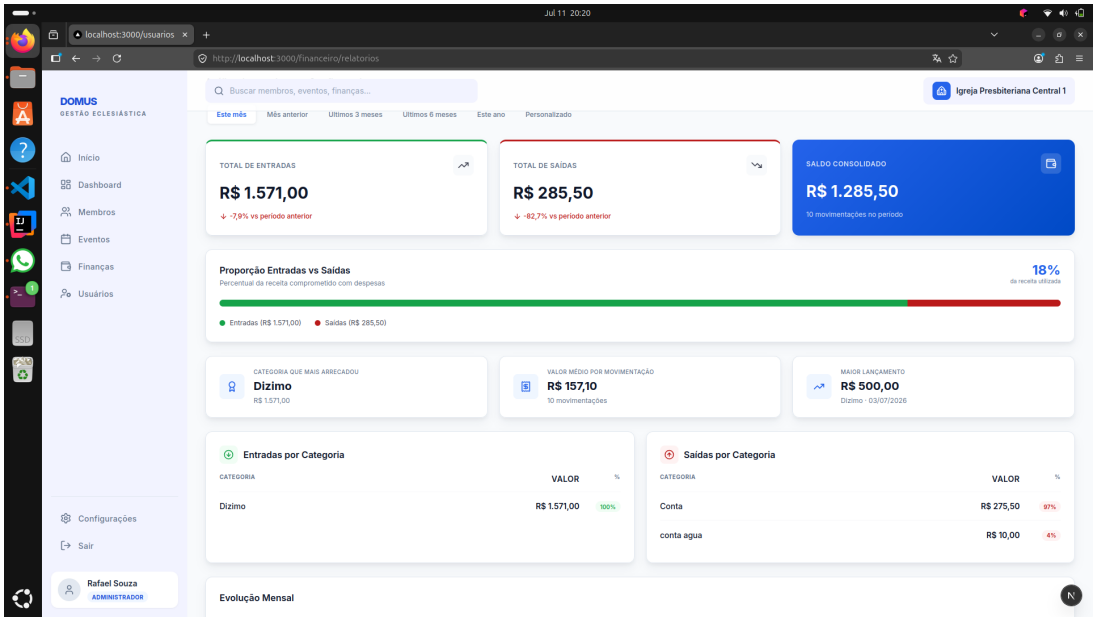
Figura 10 – Listagem de movimentações financeiras com filtros.



Fonte: Elaborado pelo autor (2026).

A Figura 11 apresenta o relatório financeiro consolidado por período. A parte superior reúne os totais de entradas, saídas e saldo, acompanhados da variação em relação ao período anterior; em seguida, são exibidos indicadores de destaque e a distribuição das movimentações por categoria, separando entradas e saídas. O período de análise pode ser selecionado entre intervalos predefinidos ou personalizado pelo usuário.

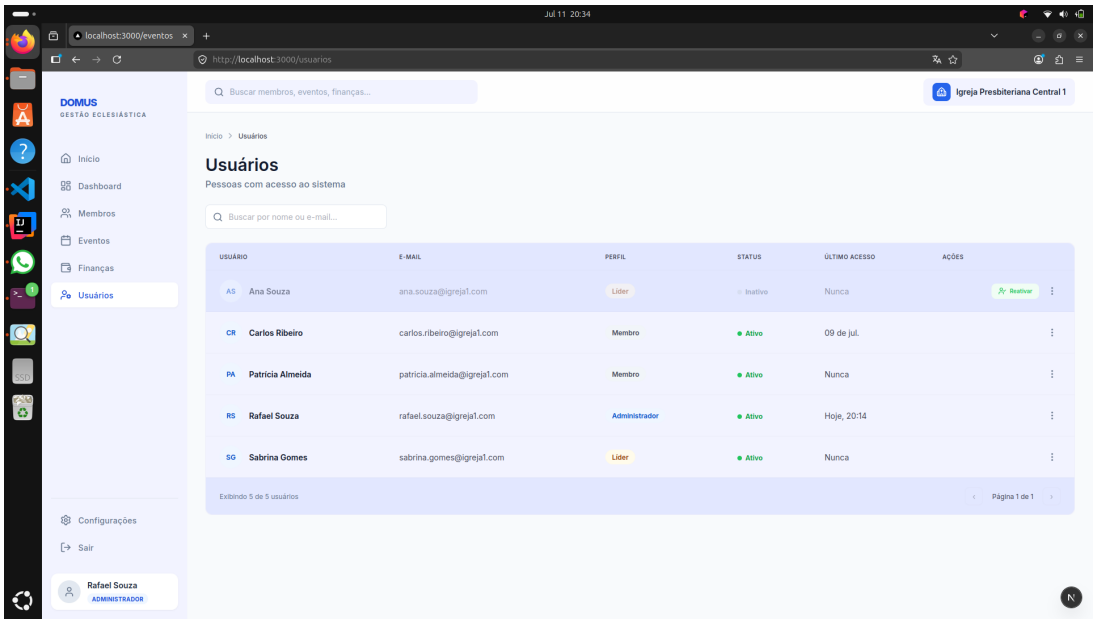
Figura 11 – Relatório financeiro consolidado por período.



Fonte: Elaborado pelo autor (2026).

A Figura 12 apresenta a tela de gestão de usuários, restrita ao Administrador da Igreja, na qual são listados os usuários com acesso ao sistema, seus e-mails, perfis, situação e data do último acesso. A partir dessa interface, o administrador concede, reativa e gerencia os acessos e os respectivos perfis de permissão.

Figura 12 – Gestão de usuários e perfis de acesso.



Fonte: Elaborado pelo autor (2026).

5 CONCLUSÃO

Este trabalho apresentou o desenvolvimento do Domus, um sistema web do tipo SaaS voltado à gestão administrativa de múltiplas igrejas. O projeto originou-se da constatação de que igrejas de pequeno e médio porte, diante da ausência de ferramentas especializadas e acessíveis, recorrem a processos manuais e dispersos para conduzir atividades essenciais como o cadastro de membros, a organização de eventos e o controle financeiro — prática que compromete a consistência dos dados, a rastreabilidade das operações e a conformidade com a legislação de proteção de dados.

O objetivo geral consistiu em desenvolver uma plataforma centralizada que cobrisse os módulos de membros, eventos e financeiro, com isolamento lógico de dados por igreja e controle de acesso baseado em perfis. Conforme detalhado no Capítulo 4, verifica-se que foi desenvolvida uma plataforma centralizada como disposto no objetivo geral. Avaliando o cumprimento dos objetivos específicos estabelecidos, verifica-se que todos foram alcançados: o banco de dados relacional foi modelado com isolamento lógico por igreja; a API REST foi desenvolvida em Java com Spring Boot, cobrindo os módulos previstos; a autenticação foi construída com tokens JWT e proteção contra força bruta; o controle de acesso por papéis foi aplicado segundo uma matriz de permissões; o frontend foi desenvolvido em Next.js com TypeScript e interface responsiva; o isolamento lógico foi assegurado em todas as operações a partir do identificador extraído do token; e a documentação técnica foi formalizada neste relatório.

Além do escopo inicialmente previsto, o desenvolvimento incorporou recursos que elevaram a maturidade técnica da solução. A adoção de cache em memória com Redis teve por objetivo reduzir a latência de leituras frequentes — cuja comprovação empírica depende de testes de carga previstos como trabalho futuro —, e o desenvolvimento de uma funcionalidade de busca avançada com Elasticsearch — sincronizada ao banco relacional por meio do padrão *transactional outbox* — ofereceu uma experiência de pesquisa unificada, tolerante a erros de digitação e alinhada ao modelo de segurança multilocatário do sistema.

Do ponto de vista acadêmico, para o autor deste relatório, o projeto permitiu consolidar, de forma integrada, competências desenvolvidas ao longo do curso, abrangendo modelagem de dados, programação orientada a objetos, segurança em aplicações web, arquitetura de software e desenvolvimento de interfaces. A necessidade de tratar desafios reais — como o isolamento entre organizações, a consistência entre repositórios distintos e a invalidação de cache — aproximou a experiência acadêmica das práticas adotadas na indústria de software.

Como toda solução em estágio inicial, o Domus apresenta limitações que delimitam o alcance dos resultados obtidos. A validação foi conduzida por meio de testes manuais, sem a incorporação de uma suíte de testes automatizados; metas de qualidade que dependem de

operação contínua em ambiente de produção — como disponibilidade monitorada, rotina de *backup* e validação de escalabilidade sob carga — não integraram o escopo verificável deste trabalho; e determinadas funcionalidades foram deliberadamente postergadas para ciclos futuros.

Ficou claro, com este relatório, que ainda há trabalhos futuros a serem realizados. Nesse sentido, vislumbram-se diversas possibilidades de evolução do Domus. Entre os trabalhos futuros, destacam-se: o desenvolvimento da visualização de eventos em formato de calendário; a autenticação por meio de provedores externos; a inscrição de membros em eventos pela própria plataforma; a exportação de relatórios em formatos como PDF e planilhas eletrônicas; o processamento de pagamentos online; o desenvolvimento de um aplicativo móvel nativo; a incorporação de testes automatizados; e o desenvolvimento de mecanismos de exclusão definitiva de dados pessoais, em atendimento pleno aos direitos previstos na Lei Geral de Proteção de Dados. Tais desdobramentos reforçam o potencial do Domus como produto e confirmam a solidez da base arquitetural construída, projetada desde sua concepção para acomodar essas evoluções.

Conclui-se que o trabalho atingiu os objetivos propostos, resultando em uma solução funcional, tecnicamente consistente e socialmente relevante. Ao oferecer uma alternativa acessível e especializada a organizações com baixa adoção de software de gestão especializado, o Domus contribui para a democratização do acesso a ferramentas profissionais de gestão e para a digitalização de um segmento com baixa adoção de software.

REFERÊNCIAS

AMAZON WEB SERVICES. *SaaS Tenant Isolation Strategies*: isolating resources in a multi-tenant environment. 2023. Disponível em: <https://docs.aws.amazon.com/whitepapers/latest/saas-tenant-isolation-strategies/saas-tenant-isolation-strategies.html>. Acesso em: 12 jul. 2026. Citado na página 37.

BRASIL. *Lei nº 13.709, de 14 de agosto de 2018*: Lei Geral de Proteção de Dados Pessoais (LGPD). Brasília, DF: [s.n.], 2018. Citado na página 18.

COHN, M. *User Stories Applied*: for agile software development. Boston: Addison-Wesley, 2004. Citado na página 14.

DATE, C. J. *Introdução a sistemas de bancos de dados*. 8. ed. Rio de Janeiro: Elsevier, 2004. Citado na página 28.

ELASTIC. *Elasticsearch Guide*. 2024. Disponível em: <https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>. Acesso em: 12 jul. 2026. Citado na página 38.

ELMASRI, R.; NAVATHE, S. B. *Sistemas de banco de dados*. 7. ed. São Paulo: Pearson, 2018. Citado na página 28.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado em Ciência da Computação) — University of California, Irvine, Irvine, 2000. Citado na página 35.

FOWLER, M. *Patterns of Enterprise Application Architecture*. Boston: Addison-Wesley, 2002. Citado na página 35.

FOWLER, M. *Transactional Outbox*. 2021. Disponível em: <https://microservices.io/patterns/data/transactional-outbox.html>. Acesso em: 12 jul. 2026. Citado na página 38.

GAMMA, E. et al. *Padrões de projeto*: soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman, 2000. Citado na página 36.

MARTIN, R. C. *Arquitetura limpa*: o guia do artesão para estrutura e design de software. Rio de Janeiro: Alta Books, 2019. Citado na página 35.

ORGANIZAÇÃO DAS NAÇÕES UNIDAS. *Transformando Nosso Mundo*: a Agenda 2030 para o Desenvolvimento Sustentável. 2015. Disponível em: <https://brasil.un.org/pt-br/sdgs>. Acesso em: 18 jul. 2026. Citado na página 17.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL Documentation*. 2024. Disponível em: <https://www.postgresql.org/docs/>. Acesso em: 12 jul. 2026. Citado na página 16.

PROVOS, N.; MAZIÈRES, D. A future-adaptable password scheme. In: USENIX ANNUAL TECHNICAL CONFERENCE. *Proceedings [...]*. Berkeley: USENIX, 1999. Citado na página 18.

REDIS LTD. *Redis Documentation*. 2024. Disponível em: <https://redis.io/docs/latest/>. Acesso em: 12 jul. 2026. Citado na página 38.

SOMMERVILLE, I. *Engenharia de software*. 10. ed. São Paulo: Pearson, 2019. Citado na página 17.

VMWARE. *Spring Boot Reference Documentation*. 2024. Disponível em: <https://docs.spring.io/spring-boot/docs/current/reference/html/>. Acesso em: 12 jul. 2026. Citado na página 16.