

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
SISTEMAS DE INFORMAÇÃO

Nathan Sargon Werlich, Victor Gevaerd Luiz

**DeMoDash: Um sistema web de gerenciamento do Capítulo Florianópolis nº 076 da
Ordem DeMolay**

Florianópolis
2026

Nathan Sargon Werlich, Victor Gevaerd Luiz

**DeMoDash: Um sistema web de gerenciamento do Capítulo Florianópolis
nº 076 da Ordem DeMolay**

Trabalho de Conclusão de Curso submetido ao
Curso de Graduação em Sistemas de Informa-
ção do Centro Tecnológico da Universidade Fe-
deral de Santa Catarina como requisito para ob-
tenção do título de Bacharel em Sistemas de
Informação.

Orientador: Prof. Maicon Rafael Zatelli

Florianópolis
2026

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Werlich, Nathan Sargon
DeMoDash : Um sistema web de gerenciamento do Capítulo
Florianópolis nº 076 da Ordem DeMolay / Nathan Sargon
Werlich, Victor Gevaerd Luiz ; orientador, Maicon Rafael
Zatelli, 2026.
183 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Centro Tecnológico,
Graduação em Sistemas de Informação, Florianópolis, 2026.

Inclui referências.

1. Sistemas de Informação. 2. Ordem DeMolay. 3. Capítulo
DeMolay. 4. Sistema Administrativo. 5. Otimização
Operacional. I. Luiz, Victor Gevaerd. II. Zatelli, Maicon
Rafael. III. Universidade Federal de Santa Catarina.
Graduação em Sistemas de Informação. IV. Título.

Nathan Sargon Werlich, Victor Gevaerd Luiz
**DeMoDash: Um sistema web de gerenciamento do Capítulo Florianópolis nº 076 da
Ordem DeMolay**

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Sistemas de Informação e aprovado em sua forma final pelo curso de Graduação em Sistemas de Informação.

Florianópolis, 15 de junho de 2026.

Prof. Dr. José Eduardo de Lucca
Coordenador do Curso

Banca Examinadora:

Prof. Maicon Rafael Zatelli
Orientador
Universidade Federal de Santa Catarina

Prof. José Eduardo De Lucca
Avaliador
Universidade Federal de Santa Catarina

Prof. Jean Carlo Rossa Hauck
Avaliador
Universidade Federal de Santa Catarina

Dedicamos este trabalho aos nossos pais, pelo amor, apoio incondicional e pelos valores que nos transmitiram, e aos nossos amigos, que tornaram essa jornada mais leve com seu incentivo constante.

AGRADECIMENTOS

Agradecemos às nossas famílias, que sempre estiveram ao nosso lado, sendo nosso apoio em todos os momentos desta jornada.

Agradecemos às nossas namoradas, Maria Eduarda Custódio e Ysadora Plegge, pelo apoio em diversos sentidos, pelo incentivo e pela motivação constantes ao longo deste trabalho.

Agradecemos aos nossos amigos Pedro, Manu, Luiz, Moser, Laís, Padilha, Will, André (PP) e Carlos (Carlão), que sempre foram parceiros e incentivaram este processo.

Agradecemos à Universidade Federal de Santa Catarina e ao Departamento de Informática e Estatística, pelo cuidado com a qualidade do curso e com os alunos, e ao nosso orientador, Prof. Maicon Rafael Zatelli, pela orientação dedicada a este trabalho.

Por fim, agradecemos ao Capítulo Florianópolis nº 076 da Ordem DeMolay, pelos ensinamentos e pelas oportunidades que nos proporcionou ao longo desses anos.

“Se eu vi mais longe, foi por estar de pé sobre ombros de gigantes”
(NEWTON, 1676)

RESUMO

O Capítulo Florianópolis nº 076 da Ordem DeMolay enfrenta dificuldades na gestão de suas operações internas, decorrentes da dependência de processos manuais e documentos em papel para o controle financeiro, de frequência e de eventos, o que compromete a transparência e a continuidade administrativa entre gestões. Neste trabalho foram pesquisados sistemas existentes voltados à gestão de capítulos da Ordem DeMolay e organizações similares, de modo a identificar soluções que atendessem às necessidades levantadas. Com base nessa análise, foi desenvolvido o “DeMoDash”, um sistema web no formato PWA com módulos de cadastro de membros, perfis de acesso, gestão financeira, controle de frequência, calendário de eventos e transparência.

Palavras-chave: Ordem DeMolay. Capítulo DeMolay. Sistema Administrativo. Otimização Operacional.

ABSTRACT

The Florianópolis Chapter No. 076 of the Order of DeMolay faces difficulties in managing its internal operations, due to its reliance on manual processes and paper documents for financial control, attendance, and event management, which compromises transparency and administrative continuity between administrations. This work researched existing systems aimed at managing DeMolay chapters and similar organizations, in order to identify solutions that would meet the identified needs. Based on this analysis, “DeMoDash” was developed, a web system in PWA format with modules for member registration, access profiles, financial management, attendance control, event calendar, and transparency.

Keywords: Order of DeMolay. DeMolay Chapter. Administrative System. Operational Optimization.

LISTA DE FIGURAS

Figura 1 – Tela de relatórios do sistema DeMolay Virtual.	40
Figura 2 – Tela do módulo de tesouraria do sistema SisMaçom.	41
Figura 3 – Tela inicial do aplicativo Maçonaria Virtual Mobile.	42
Figura 4 – Paleta de cores do aplicativo DeMoDash	52
Figura 5 – Logotipo do aplicativo DeMoDash	53
Figura 6 – Diagrama de Classes do Domínio do DeMoDash. Legenda: retângulos com três compartimentos representam classes (nome, atributos, métodos); retângulos com o estereótipo «enum» representam enumerados; linhas sólidas com multiplicidade indicam associação (ex.: 0..* = zero ou mais, 1 = exatamente um, 0..1 = opcional); setas com losango aberto na origem indicam composição fraca (agregação); setas tracejadas com rótulo «derives» indicam que o alvo é derivado do estado da classe de origem.	54
Figura 7 – Arquitetura hexagonal do módulo de gestão financeira. Caixas azuis representam adaptadores de entrada (HTTP); caixas verdes representam o núcleo da aplicação (regras de negócio); caixas amarelas representam o domínio puro (entidades e enums); caixas vermelhas representam adaptadores de saída (banco de dados). Linhas tracejadas com seta aberta indicam dependência/uso; linhas tracejadas com seta fechada indicam implementação de interface.	59
Figura 8 – Pesquisa - Capítulo Florianópolis nº 076	81
Figura 9 – Tela de login da versão mobile	90
Figura 10 – Tela de cadastro do usuário da versão mobile	91
Figura 11 – Tela central da versão mobile	92
Figura 12 – Tela de cadastro de membros da versão mobile	93
Figura 13 – Tela de listagem dos membros da versão mobile	94
Figura 14 – Tela de listagem dos perfis da versão mobile	95
Figura 15 – Tela de cadastro de perfis da versão mobile	96
Figura 16 – Tela de login da versão web	97
Figura 17 – Tela de cadastro do usuário da versão web	98
Figura 18 – Tela central da versão web	99
Figura 19 – Tela de cadastro de membros da versão web	100
Figura 20 – Tela de listagem dos membros da versão web	101
Figura 21 – Tela de listagem dos perfis da versão web	102
Figura 22 – Tela de cadastro de perfis da versão web	103

Figura 23 – Diagrama de Componentes do DeMoDash. Legenda: retângulos representam componentes; estereótipos entre « » indicam a natureza de cada componente (Web, WebService, Database, Mail); setas com rótulo indicam o protocolo de comunicação; a direção da seta indica a direção da dependência (o componente de origem depende do componente de destino).	104
Figura 24 – Diagrama de Casos de Uso do DeMoDash (US01–US17). Legenda: bonecos palito representam atores (primários à esquerda, secundário à direita); elipses representam casos de uso; o retângulo delimita o sistema; linhas sólidas entre ator e caso de uso representam associação (participação direta); linhas tracejadas com o rótulo «include» indicam que o caso de uso base somente se completa quando o caso incluído também se completa.	105
Figura 25 – Diagrama de Sequência — Registrar Transação Financeira. Legenda: caixas no topo são os participantes (<i>lifelines</i>); linhas verticais tracejadas são as linhas de vida de cada participante; retângulos sobre as linhas de vida indicam o período de ativação do participante; setas sólidas com ponta cheia representam mensagens síncronas (chamadas de função ou requisições HTTP); setas tracejadas representam retornos; o quadro <i>alt</i> separa o fluxo alternativo de erro do fluxo principal de sucesso.	106
Figura 26 – Tela de login.	107
Figura 27 – Tela inicial (home).	108
Figura 28 – Tela inicial (home) mobile.	109
Figura 29 – Tela de menu lateral mobile.	110
Figura 30 – Tela de lista de usuários.	111
Figura 31 – Tela de lista de usuários mobile.	112
Figura 32 – Tela de novo usuário.	113
Figura 33 – Tela de edição de usuário mobile.	114
Figura 34 – E-mail para ativar a conta	115
Figura 35 – Tela de visualização de usuário.	116
Figura 36 – Tela de visualização de usuário mobile.	117
Figura 37 – Tela de lista de perfis.	118
Figura 38 – Tela de lista de perfis mobile.	119
Figura 39 – Tela de novo perfil (parte 1).	120
Figura 40 – Tela de novo perfil (parte 2).	121
Figura 41 – Tela de edição de perfil (parte 1).	122
Figura 42 – Tela de edição de perfil (parte 2).	123
Figura 43 – Tela de edição de perfil mobile.	124
Figura 44 – Tela de exclusão de perfil.	125
Figura 45 – Tela de meu perfil.	126
Figura 46 – Tela de meu perfil mobile.	127
Figura 47 – Tela de edição do próprio perfil.	128

Figura 48 – Tela de edição do próprio perfil mobile.	129
Figura 49 – Tela de configurações.	130
Figura 50 – Tela de configurações mobile.	131
Figura 51 – Tela do <i>dashboard</i> de gestão financeira.	132
Figura 52 – Tela do <i>dashboard</i> de gestão financeira mobile.	133
Figura 53 – Tela de transações financeiras.	134
Figura 54 – Tela de nova transação.	135
Figura 55 – Tela de nova transação mobile.	136
Figura 56 – Tela de edição de transação.	137
Figura 57 – Tela de edição de transação mobile.	138
Figura 58 – Tela de exclusão de transação.	139
Figura 59 – Tela de categorias financeiras.	140
Figura 60 – Tela de categorias financeiras mobile.	141
Figura 61 – Tela de adição de categoria financeira.	142
Figura 62 – Tela de balanço financeiro mobile.	143
Figura 63 – Tela de gráfico de pizza mobile.	144
Figura 64 – Tela de gráfico de linha mobile.	145
Figura 65 – Tela de gráfico de barras mobile.	146
Figura 66 – Tela de frequência.	147
Figura 67 – Tela de registro de presença.	148
Figura 68 – Tela de registro de presença mobile.	149
Figura 69 – Tela de relatório de frequência.	150
Figura 70 – Tela de relatório de frequência mobile.	151
Figura 71 – Tela de minha frequência mobile.	152
Figura 72 – Tela de calendário (visão mensal).	153
Figura 73 – Tela de calendário (visão diária).	154
Figura 74 – Tela de calendário (visão diária) mobile.	155
Figura 75 – Tela de lista de eventos do calendário.	156
Figura 76 – Tela de lista de eventos do calendário mobile.	157
Figura 77 – Tela de novo evento.	158
Figura 78 – Tela de edição de evento.	159
Figura 79 – Tela de edição de evento mobile.	160
Figura 80 – Tela de exclusão de evento.	161
Figura 81 – Tela de exclusão de evento mobile.	162
Figura 82 – Tela de transparência — balanço mensal.	163
Figura 83 – Tela de transparência — gráfico de rosca.	164
Figura 84 – Tela de transparência — gráfico de linha.	165
Figura 85 – Tela de transparência — gráfico de colunas do balanço mensal.	166
Figura 86 – Tela de transparência — gráfico de fluxo de caixa do semestre.	167
Figura 87 – Modelo Membro - Membro.kt.	168

Figura 88 – Registro de Presença - RegistroPresenca.kt.	169
Figura 89 – Interface da porta de entrada RegistrarTransacaoUseCase - RegistrarTransacaoUseCase.kt.	170
Figura 90 – Implementação RegistrarTransacaoService - RegistrarTransacaoService.kt.	171
Figura 91 – Método registrar() do TransacaoController - TransacaoController.kt. .	172
Figura 92 – Função serverFetch<T>() em src/lib/api/server-fetch.ts: leitura do cookie de sessão e encaminhamento ao backend.	173
Figura 93 – Trecho de actions.ts: renomeação do cookie SESSION do backend para session no Next.js.	174

LISTA DE TABELAS

Tabela 1 – Comparativo de funcionalidades dos sistemas	39
Tabela 2 – Infraestrutura de implantação do DeMoDash.	61
Tabela 3 – Comparativo de funcionalidades dos sistemas, incluindo o DeMoDash. . . .	69
Tabela 4 – Distribuição dos testes automatizados do DeMoDash.	70
Tabela 5 – Casos de teste funcionais executados manualmente.	72

SUMÁRIO

1	INTRODUÇÃO	18
1.1	MOTIVAÇÃO	18
1.2	ORGANIZAÇÃO	19
1.3	METODOLOGIA	20
1.3.1	Fases e Atividades do Projeto	20
1.3.2	Aspectos Éticos e Ferramentas de Apoio	21
1.4	OBJETIVOS	21
1.5	ESCOPO DO PROJETO	23
2	FUNDAMENTAÇÃO TEÓRICA	25
2.1	MAÇONARIA	25
2.1.1	Paramaçônicas	25
2.2	ORDEM DEMOLAY	26
2.2.1	Ordem DeMolay no Brasil	27
2.2.2	Ordem DeMolay em Santa Catarina	27
2.2.3	Capítulo Florianópolis nº 076	28
2.3	SISTEMAS WEB	28
2.3.1	Como a web funciona	29
2.3.2	Aplicativos Distribuídos	29
2.3.3	Progressive Web Apps	29
2.4	LINGUAGENS DE PROGRAMAÇÃO	30
2.4.1	Frontend	30
2.4.1.1	JavaScript	31
2.4.1.2	TypeScript	31
2.4.2	Backend	31
2.4.2.1	Kotlin	32
2.5	API	32
2.5.1	Rest	32
2.6	BIBLIOTECAS E FRAMEWORKS	32
2.6.1	React	33
2.6.2	JpaRepository	34
2.6.3	SpringBoot	34
2.6.4	Liquibase	34
2.7	BANCO DE DADOS	34
2.7.1	Relacional	35
2.7.2	PostgreSQL	35
2.8	GIT (VERSIONAMENTO)	35

2.8.1	GitHub	36
2.9	PROTOTIPAÇÃO	36
3	TRABALHOS RELACIONADOS	37
3.1	PROTOCOLO DE PESQUISA	37
3.1.1	Termos de Busca Utilizados	37
3.1.2	Período de Execução	37
3.1.3	CrITÉrios de Inclusão/Exclusão	37
3.1.4	Seleção dos Resultados	38
3.2	DEMOLAY VIRTUAL - CRIE O SITE DE SEU CAPÍTULO DEMOLAY EM 3 MINUTOS!	39
3.3	SISMAÇOM - SISTEMA PARA GESTÃO DE LOJA MAÇÔNICA	40
3.4	MAÇONARIA VIRTUAL MOBILE	41
4	DESENVOLVIMENTO	43
4.1	ANÁLISE E LEVANTAMENTO DOS REQUISITOS DA APLICAÇÃO	43
4.1.1	Entrevistas	44
4.1.1.1	<i>Plano de entrevista</i>	44
4.1.1.2	<i>Objetivos da entrevista</i>	44
4.1.1.3	<i>Resultados da entrevista realizada</i>	45
4.1.2	Questionários	46
4.1.2.1	<i>Plano de questionário</i>	46
4.1.2.2	<i>Perfil dos respondentes</i>	46
4.1.2.3	<i>Objetivos do questionário</i>	46
4.1.2.4	<i>Resultados do questionário realizado</i>	47
4.1.2.5	<i>Personas</i>	47
4.1.3	Histórias de Usuário	49
4.1.4	Prototipagem	50
4.2	DESENVOLVIMENTO DA SOLUÇÃO	50
4.2.1	Identidade do Aplicativo	51
4.2.2	Modelagem do Sistema	53
4.2.2.1	<i>Diagramas Estruturais</i>	53
4.2.2.1.1	<i>Diagrama de Componentes</i>	53
4.2.2.1.2	<i>Diagrama de Classes do Domínio</i>	54
4.2.2.2	<i>Diagramas Comportamentais</i>	54
4.2.2.2.1	<i>Diagrama de Casos de Uso</i>	54
4.2.2.2.2	<i>Diagrama de Sequência</i>	55
4.2.3	Organização do Projeto	55
4.2.3.1	<i>Backend</i>	55
4.2.3.2	<i>Frontend</i>	57

4.2.3.3	<i>Banco de Dados</i>	58
4.2.4	Arquitetura do Sistema	58
4.2.4.1	<i>Arquitetura Hexagonal do Backend</i>	58
4.2.4.2	<i>Arquitetura do Frontend</i>	60
4.2.4.3	<i>Gerenciamento de Erros e Permissões</i>	61
4.2.4.4	<i>Infraestrutura de Implantação</i>	61
4.2.5	Implementação das Histórias de Usuário	62
4.2.5.1	<i>Módulo de Membros (US01–US04)</i>	62
4.2.5.2	<i>Módulo de Perfis (US05–US07)</i>	63
4.2.5.3	<i>Módulo de Gestão Financeira (US08–US09)</i>	64
4.2.5.4	<i>Módulo de Controle de Frequência (US10–US12)</i>	65
4.2.5.5	<i>Módulo de Calendário (US13–US15)</i>	66
4.2.5.6	<i>Módulo de Transparência (US16–US17)</i>	67
5	AVALIAÇÃO	69
5.1	TESTES AUTOMATIZADOS	69
5.1.1	Testes de Unidade	70
5.1.2	Testes de Integração	70
5.2	INTEGRAÇÃO CONTÍNUA E GARANTIA DE QUALIDADE	71
5.3	CASOS DE TESTE FUNCIONAIS	71
5.4	DISPONIBILIZAÇÃO	72
5.4.1	Código	73
5.4.2	Configuração de Ambiente	73
6	CONCLUSÃO	74
6.1	TRABALHOS FUTUROS	74
	REFERÊNCIAS	76
	APÊNDICE A – RESULTADOS DO QUESTIONÁRIO	81
A.1	PERGUNTAS ELABORADAS PARA O QUESTIONÁRIO	81
A.2	RESPOSTAS COLETADAS	82
A.2.1	Respondente 1	82
A.2.2	Respondente 2	83
A.2.3	Respondente 3	84
A.2.4	Respondente 4	85
A.2.5	Respondente 5	86
A.2.6	Respondente 6	87
A.2.7	Respondente 7	87
A.2.8	Respondente 8	88

APÊNDICE B – FIGURAS	90
APÊNDICE C – ARTIGO NO FORMATO SBC	175
APÊNDICE D – CÓDIGO FONTE	182

1 INTRODUÇÃO

Neste trabalho, é abordada uma iniciativa para a modernização e aprimoramento das operações internas do Capítulo Florianópolis nº 076 da Ordem DeMolay (OD). A OD é uma organização voltada para jovens de 12 a 21 anos, com uma longa história de contribuição para o desenvolvimento de cidadãos conscientes, empáticos e comprometidos com o bem-estar da sociedade (SCDB | Centro Nacional de Liderança DeMolay, 2023). No entanto, como muitas organizações tradicionais, o Capítulo Florianópolis enfrenta desafios relacionados à eficiência e à gestão de suas atividades internas.

Os Capítulos da OD, incluindo o Capítulo Florianópolis nº 076, operam com um sistema de administração independente, seguindo diretrizes estabelecidas pelo Supremo Conselho DeMolay Brasil (SCDB). Essa independência oferece flexibilidade, mas também apresenta desafios, especialmente quando se trata de realizar tarefas manuais e gerenciar documentos em papel. A necessidade de modernização e automação dos processos administrativos se tornou evidente, visando melhorar a eficiência, garantir a continuidade das operações e oferecer transparência às atividades desenvolvidas pelo Capítulo.

Assim, a solução proposta por este trabalho é implementar um sistema administrativo que possa otimizar e tornar mais eficientes os processos dentro do Capítulo Florianópolis nº 076. Este sistema consiste em diversos módulos, abrangendo desde o cadastro de membros e perfis até a gestão financeira, controle de frequência e um calendário de atividades. A implementação desses módulos visa simplificar e agilizar as tarefas cotidianas, reduzindo a dependência de processos manuais, documentos em papel e evitando a perda de informações durante as trocas de gestão, alinhando-se com os princípios da OD de desenvolvimento de jovens líderes.

1.1 MOTIVAÇÃO

A motivação para a realização deste trabalho está intrinsecamente relacionada às dificuldades operacionais enfrentadas pelo Capítulo Florianópolis nº 076 da OD no contexto de suas atividades diárias. Essas dificuldades foram inicialmente identificadas por um dos membros ativos do Capítulo, fato que desempenhou um grande papel no processo de concepção deste projeto.

A necessidade de encontrar uma solução que automatizasse e otimizasse os processos internos do Capítulo se tornou evidente após uma análise detalhada das operações em curso. Ficou claro que havia uma série de desafios que precisavam ser superados para melhorar a eficiência e a gestão das atividades do Capítulo:

- **Gestão Financeira:** Era necessário aprimorar a gestão financeira da instituição, permitindo o rastreamento de receitas e despesas de maneira mais precisa e eficaz.
- **Controle de Frequência:** O acompanhamento da presença dos membros nas reuniões e eventos era essencial para garantir o engajamento e a participação ativa.

- **Calendário de Eventos:** Um calendário unificado para criar, gerenciar e divulgar eventos se fazia necessário para facilitar a organização das atividades do Capítulo.
- **Controle de Acessos:** Era fundamental implementar perfis de usuário personalizados para garantir o acesso adequado às informações e funcionalidades do sistema.
- **Transparência:** A geração de relatórios era imprescindível para fornecer uma visão transparente das atividades e finanças do Capítulo, possibilitando a prestação de contas aos membros e patrocinadores.

A ausência de uma plataforma gratuita que englobasse todos esses recursos essenciais levou à conclusão de que era necessário criar uma solução personalizada para atender às demandas específicas da OD.

Com vistas à análise dos sistemas existentes, conduziu-se um estudo das plataformas gratuitas disponíveis e já utilizadas no Capítulo Florianópolis nº 076. As ferramentas foram instaladas e utilizadas com o objetivo de atender aos requisitos listados nas entrevistas realizadas. Com isso, identificou-se que nenhum sistema atendia completamente aos requisitos levantados.

Além disso, a motivação para a realização deste trabalho é fortalecida pelo fato de um dos autores ser membro do Capítulo Florianópolis nº 076. Essa conexão direta com o público-alvo torna a comunicação mais acessível e a compreensão dos desafios enfrentados pelo Capítulo mais aprofundada. Isso proporciona uma perspectiva única e valiosa para a concepção e implementação de um sistema administrativo que seja verdadeiramente eficaz e alinhado com as necessidades do Capítulo e da Ordem DeMolay como um todo.

1.2 ORGANIZAÇÃO

A estrutura deste trabalho segue uma organização clara, dividida em seis capítulos, cada um com seu propósito específico:

1. **Introdução:** Neste capítulo, apresenta-se uma visão geral do problema e as motivações que impulsionaram este trabalho. São ainda apresentados os objetivos do projeto, a metodologia adotada, o escopo — incluindo os requisitos funcionais e não funcionais do sistema — e a organização dos demais capítulos.
2. **Fundamentação Teórica:** O segundo capítulo apresenta as bases teóricas e tecnológicas necessárias para uma compreensão completa do trabalho.
3. **Trabalhos Relacionados:** O terceiro capítulo aborda os trabalhos relacionados encontrados, destacando seus aspectos positivos e negativos, fornecendo um contexto para o projeto.
4. **Desenvolvimento:** O quarto capítulo detalha o processo de desenvolvimento do aplicativo proposto neste trabalho, incluindo a arquitetura, tecnologias utilizadas e desafios enfrentados.

5. **Avaliação:** O quinto capítulo descreve a avaliação do sistema, abrangendo as estratégias de validação e verificação adotadas — os testes automatizados, a integração contínua, os casos de teste funcionais — e a disponibilização do sistema.
6. **Conclusão:** O sexto e último capítulo apresenta as conclusões obtidas com o projeto e sugere direções para trabalhos futuros, destacando oportunidades de aprimoramento.

Além dos capítulos principais, também há a seção de referências bibliográficas para fornecer recursos adicionais e informações complementares conforme necessário. Essa organização estruturada visa proporcionar uma compreensão clara e abrangente do projeto, desde sua concepção até suas conclusões e possibilidades de expansão.

1.3 METODOLOGIA

A seção de Metodologia deste trabalho descreve como os objetivos foram alcançados e como o projeto foi conduzido. Os métodos selecionados visaram garantir um desenvolvimento eficaz do sistema proposto. As técnicas utilizadas são contextualizadas nas fases e atividades do projeto. A seguir, são apresentadas as fases e atividades, juntamente com as técnicas correspondentes:

1.3.1 Fases e Atividades do Projeto

1. Fase 1: Levantamento de Requisitos

- **Atividade 1.1: Revisão de Documentos Institucionais**

- Técnica: Revisão detalhada dos documentos que regem o funcionamento do Capítulo Florianópolis nº 076 e da OD em nível estadual e nacional, incluindo estatutos, regimentos e regulamentos.

- **Atividade 1.2: Entrevistas**

- Técnica: Condução de entrevistas com membros do Capítulo Florianópolis nº 076 para capturar suas perspectivas e necessidades.

- **Atividade 1.3: Questionários**

- Técnica: Elaboração e aplicação de questionários para coletar informações adicionais e opiniões dos membros do Capítulo.

2. Fase 2: Análise dos Trabalhos Relacionados

- **Atividade 2.1: Revisão da Literatura e Sistemas Existentes**

- Técnica: Pesquisa extensiva em bases de dados acadêmicas e lojas de aplicativos para identificar soluções correlatas e melhores práticas.

3. Fase 3: Definição de Requisitos

- **Atividade 3.1: Documentação dos Requisitos**

- Técnica: Compilação e documentação dos requisitos funcionais e não funcionais com base nas informações coletadas nas fases anteriores.

4. Fase 4: Projeto e Desenvolvimento do Sistema

- **Atividade 4.1: Elaboração da Arquitetura do Sistema**

- Técnica: Definição da arquitetura do sistema, incluindo a especificação dos módulos e suas interações.

- **Atividade 4.2: Desenvolvimento do Sistema**

- Técnica: Implementação do sistema administrativo, abrangendo os módulos de cadastro de membros, gestão financeira, controle de frequência e calendário de atividades.

5. Fase 5: Implantação e Testes

- **Atividade 5.1: Testes Funcionais e de Usabilidade**

- Técnica: Realização de testes para assegurar que o sistema atende aos requisitos especificados.

- **Atividade 5.2: Implantação do Sistema**

- Técnica: Implantação do sistema no ambiente de homologação.

A combinação dessas fases e técnicas visou garantir que o sistema fosse projetado e desenvolvido de forma sólida, atendendo às necessidades específicas do Capítulo Florianópolis nº 076 da OD. Além disso, a documentação adequada do processo permite uma compreensão clara de como o projeto foi conduzido e quais decisões foram tomadas ao longo do caminho.

1.3.2 Aspectos Éticos e Ferramentas de Apoio

No processo de refinamento deste TCC, ferramentas de IA generativa (ChatGPT, Gemini e Claude) foram empregadas pontualmente como assistentes de suporte. O uso restringiu-se à revisão gramatical e obtenção de *feedback* sobre o texto, revisão e otimização de código, bem como auxílio na escrita de *issues*, *pull requests* e *commits*, sem interferência na criação do conteúdo científico ou na interpretação dos dados.

1.4 OBJETIVOS

O presente projeto estabelece uma série de objetivos, subdivididos em um objetivo geral e objetivos específicos, com o propósito de direcionar a implementação da aplicação destinada a auxiliar os membros do Capítulo Florianópolis nº 076 da Ordem DeMolay nas suas

atividades administrativas e na tomada de decisões. A seguir, serão apresentados esses objetivos:

Objetivo Geral

O objetivo geral deste projeto é analisar, projetar, desenvolver e testar um sistema de gerenciamento de rotinas administrativas para o Capítulo Florianópolis nº 076 da Ordem DeMolay.

Objetivos Específicos

1. Levantar e Analisar os Requisitos:

- Identificar as necessidades e expectativas dos usuários por meio de entrevistas e questionários.
- Documentar os requisitos funcionais e não funcionais do sistema.

2. Analisar os Trabalhos Relacionados:

- Realizar uma revisão da literatura e dos sistemas existentes para identificar soluções correlatas e melhores práticas.
- Comparar as funcionalidades e limitações dos sistemas analisados com os requisitos levantados.

3. Projetar e Desenvolver o Sistema:

- Elaborar a arquitetura do sistema, definindo os módulos e suas interações.
- Desenvolver o sistema administrativo para o Capítulo Florianópolis nº 076 da Ordem DeMolay, implementando os módulos de cadastro de membros, gestão financeira, controle de frequência e calendário de atividades.

4. Implantar e Testar o Sistema:

- Realizar testes funcionais e de usabilidade para assegurar que o sistema atende aos requisitos especificados.
- Implantar o sistema no ambiente de homologação.

Esses objetivos específicos são cuidadosamente delineados para assegurar que a aplicação atenda de maneira abrangente e eficaz às necessidades do Capítulo Florianópolis nº 076 da OD, contribuindo para a otimização das operações administrativas e para a promoção da transparência em todas as suas atividades.

1.5 ESCOPO DO PROJETO

O escopo deste projeto é definido para estabelecer claramente os limites e as diretrizes que guiarão o desenvolvimento da aplicação destinada a atender às necessidades do Capítulo Florianópolis nº 076 da Ordem DeMolay. Para a definição do escopo inicial do projeto, identificaram-se as principais dificuldades por meio de conversas com membros da Ordem DeMolay. Além disso, um dos autores participa ativamente da Ordem, o que permitiu, por meio de sua vivência, reconhecer diversos pontos de dificuldade enfrentados pela OD.

O escopo abrange os seguintes aspectos:

- Desenvolvimento de um software compatível com dispositivos móveis que permita aos membros do Capítulo Florianópolis administrarem suas atividades.
- O software deve incluir os seguintes módulos:
 - Módulo de membros: Capacidade de inserir, editar, visualizar e excluir membros do Capítulo.
 - Módulo de perfis: Capacidade de inserir, editar, visualizar e excluir perfis do Capítulo.
 - Módulo de gestão financeira: Cadastro de transações, categorização de transações (entrada e saída).
 - Módulo de controle de frequência: Registro de presença dos membros em eventos do Capítulo.
 - Módulo de calendário: Criação de eventos no calendário do Capítulo.
 - Módulo de transparência: Visualização de informações de gestão financeira e frequência dos membros, sendo possível filtrar por períodos.
- Os módulos mencionados anteriormente terão restrições de edição e acesso controlados pelos perfis, mas os perfis devem ser personalizáveis.
- O acesso ao sistema será exclusivo para membros do Capítulo.
- Implantação do sistema em produção.

Outros Requisitos

- O aplicativo deve ser compatível com o navegador Google Chrome e com dispositivos iOS e Android.
- O aplicativo deve ser um *Progressive Web App* (PWA).
- Segurança: O aplicativo deve permitir o acesso apenas para membros do Capítulo.
- Disponibilidade 24/7, com uma janela semanal para manutenção periódica.

Restrições

- A aplicação deve ser compatível com dispositivos móveis (Android e iOS).
- Deve ser garantida a segurança de informações sensíveis.
- Manutenção da aplicação disponível para uso na nuvem.

Limitações de Escopo

- O uso do aplicativo será exclusivo para o Capítulo Florianópolis nº 076, embora a expansão para outros Capítulos possa ser considerada no futuro.
- A filiação de membros está restrita ao Capítulo Florianópolis nº 076 e não está aberta a qualquer DeMolay.

Esta definição de escopo estabelece as bases para o projeto, delineando claramente o que será incluído e o que estará fora do âmbito do desenvolvimento do sistema.

2 FUNDAMENTAÇÃO TEÓRICA

Este trabalho está diretamente relacionado ao gerenciamento de um Capítulo da Ordem DeMolay e ao desenvolvimento de um *Progressive Web App* (PWA). Para tanto, torna-se necessário adquirir uma compreensão completa de vários termos e tecnologias aplicadas no contexto do projeto. Nesta seção, serão apresentadas explicações detalhadas de alguns conceitos fundamentais que desempenharão um papel essencial na compreensão e no desenvolvimento do sistema proposto.

2.1 MAÇONARIA

A Maçonaria é uma das fraternidades mais antigas e influentes do mundo, caracterizada por seus rituais simbólicos e forte ênfase em valores éticos e morais. Com origens que remontam às guildas de pedreiros medievais, ela evoluiu para uma organização mais espiritual e filosófica, aberta a homens de diferentes profissões e camadas sociais (AZEVEDO, 1996).

A Maçonaria moderna começou a se formar no início do século XVIII, com a fundação da Grande Loja de Londres em 1717, marcando o início da Maçonaria Especulativa. Esta fase representou uma transição dos aspectos operativos dos pedreiros medievais para um movimento filosófico e fraternal (AZEVEDO, 1996).

No Brasil, a Maçonaria teve um papel significativo no desenvolvimento histórico do país. As primeiras tentativas de estabelecimento ocorreram no final do século XVIII, com a presença de maçons avulsos em diversas regiões do país. A primeira loja maçônica propriamente dita no Brasil foi a “Reunião”, fundada em 1801 e filiada ao Grande Oriente da França, seguindo o Rito Moderno ou Francês (SILVA, 2015).

Em 1822, foi fundado o Grande Oriente do Brasil, um marco para a Maçonaria brasileira e para o movimento pela independência do país. Este evento ocorreu em uma assembleia magna na loja “Comércio e Artes”, reunindo maçons influentes, incluindo políticos e eclesiásticos, que visavam a independência do Brasil (SILVA, 2015).

Os objetivos da Maçonaria incluem o desenvolvimento moral e espiritual de seus membros, a promoção da fraternidade e da caridade, e a busca contínua pelo conhecimento e pela sabedoria. A organização enfatiza princípios como a liberdade de pensamento, a igualdade e a fraternidade universal (AZEVEDO, 1996).

A Maçonaria é uma entidade global com uma rica história e um impacto duradouro na sociedade. No Brasil, ela desempenhou um papel fundamental em momentos-chave da história do país, refletindo seus princípios de liberdade, igualdade e fraternidade.

2.1.1 Paramaçônicas

As entidades paramaçônicas representam um espectro de organizações que, embora não sejam formalmente parte da Maçonaria, estão intimamente ligadas a ela em termos de

ideais, estruturas e membros. Essas entidades incluem uma variedade de grupos que atendem a diferentes segmentos sociais, como jovens e mulheres, e possuem objetivos educacionais, filantrópicos e sociais (CASTRO et al., 2021).

A Ordem DeMolay é uma das mais significativas entidades paramaçônicas, focada no desenvolvimento de jovens do sexo masculino. Como uma organização juvenil paramaçônica, a Ordem DeMolay tem um papel significativo na sociedade, contribuindo para a formação de indivíduos comprometidos com valores éticos e morais. O engajamento social e comunitário é um aspecto central, com os membros participando de diversas atividades filantrópicas e comunitárias (CASTRO et al., 2021).

As entidades paramaçônicas, particularmente a Ordem DeMolay, desempenham um papel vital no desenvolvimento social e na perpetuação dos valores maçônicos. Elas oferecem uma plataforma para o engajamento juvenil, o desenvolvimento pessoal e a contribuição comunitária, refletindo os ideais maçônicos de fraternidade, caridade e verdade.

2.2 ORDEM DEMOLAY

A Ordem DeMolay (OD), conforme detalhado no livro “*Hi, Dad!*” pelo reverendo Herbert Ewing Duncan (1970), é uma organização fraterna juvenil. Ela foi fundada em 1919 por Frank S. Land, um homem com visão de proporcionar um ambiente de suporte e desenvolvimento para jovens do sexo masculino. A Ordem recebe o nome de Jacques DeMolay, o último Grão-Mestre dos Cavaleiros Templários, e tem como foco o desenvolvimento de valores morais e de liderança.

Na estrutura da Ordem, observa-se a ênfase em sete virtudes cardinais: amor filial, reverência pelas coisas sagradas, cortesia, companheirismo, fidelidade, pureza (de corpo e mente) e patriotismo. Estas virtudes são integradas em todas as atividades e rituais da Ordem, oferecendo um guia de comportamento para os jovens (DUNCAN, 1970).

A formação proporcionada pela Ordem DeMolay vai além do desenvolvimento moral. Há uma ênfase significativa no aprimoramento de habilidades de liderança e na promoção do serviço comunitário. Os membros são encorajados a participar de atividades que beneficiam suas comunidades, desenvolvendo um senso de responsabilidade social e cidadania ativa (DUNCAN, 1970).

O “*Hi, Dad!*” narra não apenas a história da fundação da Ordem, mas também a trajetória de vida de Frank S. Land e seu impacto na formação da Ordem DeMolay. O livro serve como uma ferramenta educacional para membros da Ordem, oferecendo *insights* sobre sua história e valores (DUNCAN, 1970).

Em resumo, a OD, como descrita no “*Hi, Dad!*”, é uma organização que desempenha um papel no desenvolvimento de jovens, enfocando tanto o crescimento pessoal como o compromisso com a comunidade.

2.2.1 Ordem DeMolay no Brasil

A Ordem DeMolay, uma organização fraterna voltada para o desenvolvimento juvenil, foi oficialmente estabelecida no Brasil em 1980, marcando o início de sua presença na América do Sul. Esta chegada foi orquestrada por Alberto Mansur, na época Soberano Grande Comendador do Supremo Conselho do Rito Escocês Antigo e Aceito da Maçonaria para a República Federativa do Brasil (SCDB | Centro Nacional de Liderança DeMolay, 2020).

Inicialmente, a Ordem DeMolay no Brasil não possuía um Supremo Conselho próprio. Sua gestão e patrocínio eram realizados pelo Supremo Conselho do Grau 33º, com Mansur atuando como Oficial Executivo para o Brasil (CASTRO et al., 2021).

A Ordem DeMolay no Brasil cresceu rapidamente, expandindo suas fronteiras para as principais cidades do país. Com mais de 100 mil membros cadastrados, entre DeMolays, Maçons, Pais, Mães e Amigos, a Ordem DeMolay brasileira ganhou reconhecimento internacional (SCDB | Centro Nacional de Liderança DeMolay, 2020).

A Ordem DeMolay no Brasil se solidificou ao longo dos anos, tornando-se uma referência para países como Estados Unidos, Alemanha, Itália e Romênia (SCDB | Centro Nacional de Liderança DeMolay, 2020).

A trajetória da Ordem DeMolay no Brasil reflete uma história de dedicação, crescimento e impacto significativo na juventude (CASTRO et al., 2021).

2.2.2 Ordem DeMolay em Santa Catarina

A Ordem DeMolay (OD) foi estabelecida em Santa Catarina em agosto de 1987, com a instalação do Capítulo Joinville nº 49, marcando o início de sua presença no estado (Grande Conselho Estadual da Ordem DeMolay para Santa Catarina, 2016).

Com a criação do Supremo Conselho da Ordem DeMolay para a República Federativa do Brasil em 2004, Santa Catarina seguiu os desenvolvimentos nacionais, fundando o Grande Conselho Estadual da Ordem DeMolay para Santa Catarina em 2005 (Grande Conselho Estadual da Ordem DeMolay para Santa Catarina, 2016).

A unificação dos poderes nacionais em 2019 e a formação do Supremo Conselho DeMolay Brasil foram cruciais para o aumento do número de Capítulos e membros em Santa Catarina (Grande Conselho Estadual da Ordem DeMolay para Santa Catarina, 2016).

Santa Catarina possui uma variedade de entidades associadas à Ordem DeMolay, incluindo Priorados, Cortes Chevalier, Castelos da Ordem dos Escudeiros, entre outros (Grande Conselho Estadual da Ordem DeMolay para Santa Catarina, 2016).

O estado de Santa Catarina comemora o Dia Estadual da OD em 18 de março (Grande Conselho Estadual da Ordem DeMolay para Santa Catarina, 2016).

A trajetória da OD em Santa Catarina reflete o compromisso com o desenvolvimento juvenil e o impacto positivo da Ordem no estado (Grande Conselho Estadual da Ordem DeMolay para Santa Catarina, 2016).

2.2.3 Capítulo Florianópolis nº 076

Fundado no dia 5 de novembro de 1987, o Capítulo Florianópolis nº 076 da Ordem DeMolay (OD) representa um marco significativo na história da Ordem em Santa Catarina. Jovens, muitos dos quais já haviam sido adotados como *lowtons*, foram convocados para se reunir no antigo templo da Loja Ordem e Trabalho, dando início à fundação do Capítulo. Este evento ocorreu sob o patrocínio do Grande Oriente de Santa Catarina (GOSC), que continua a ser o patrocinador atual do capítulo. A instalação oficial do capítulo ocorreu em 30 de abril de 1988, com a iniciação e elevação de 33 jovens nos Graus Iniciático e DeMolay (Capítulo Florianópolis nº 076, 2023d).

O Capítulo Florianópolis nº 076 da OD é ativo na organização de eventos e atividades filantrópicas. Realiza dois eventos principais durante o ano, incluindo a macarronada anual, com foco no desenvolvimento dos membros e arrecadação de fundos para ações sociais e operacionais do Capítulo. As atividades filantrópicas são uma parte essencial das ações do Capítulo, colaborando com instituições como o Asilo Irmão Joaquim, Casalar Emaús, Creche São Francisco de Assis, Lar dos Velhinhos de Zulma, Lar Recanto do Carinho e a SERTE (Capítulo Florianópolis nº 076, 2023b; Capítulo Florianópolis nº 076, 2023c).

O Capítulo possui várias comissões responsáveis por diferentes aspectos de sua gestão e atividades, como Assuntos Externos, Auditoria, Eventos, Filantropias, Publicidade e Marketing, Sindicância e Ritualística, e Tesouraria (Capítulo Florianópolis nº 076, 2023a).

O Capítulo Florianópolis nº 076 da Ordem DeMolay é um exemplo notável do espírito e dos valores da Ordem DeMolay. Por meio de seus eventos, atividades filantrópicas e da dedicação de suas comissões, o Capítulo demonstra um compromisso contínuo com o desenvolvimento de jovens líderes e a promoção do bem-estar comunitário.

2.3 SISTEMAS WEB

Sistemas web representam um conjunto diversificado de aplicações e serviços que operam na internet ou intranet, acessíveis predominantemente por meio de navegadores web. Estes sistemas são uma parte integral da arquitetura global de informação e comunicação, oferecendo uma vasta gama de funcionalidades que vão desde a simples apresentação de conteúdo estático até aplicações interativas complexas, como plataformas de *e-commerce*, redes sociais, e sistemas de gerenciamento de conteúdo (GINIGE, 2002; MOREAU et al., 2010).

Estes sistemas são construídos utilizando um conjunto de tecnologias padrão da Web, como HTML, CSS e JavaScript, e operam sobre protocolos de rede, como HTTP/HTTPS. A flexibilidade e a escalabilidade dos sistemas web os tornam adequados para uma variedade de aplicações, desde pequenos websites pessoais até grandes aplicações corporativas.

2.3.1 Como a web funciona

A funcionalidade básica da web envolve a solicitação e a entrega de recursos, como páginas da web, imagens, e vídeos, por meio do protocolo HTTP. Quando um usuário digita uma URL no navegador, este faz uma solicitação ao servidor web correspondente. O servidor, então, processa essa solicitação e envia uma resposta, que pode ser uma página web, uma imagem, ou outro tipo de conteúdo (BERNERS-LEE; FIELDING; MASINTER, 2005).

Além disso, a Web Semântica, uma extensão da web convencional, permite a interpretação e manipulação de dados de maneira mais inteligente. Utilizando tecnologias como RDF e OWL, a Web Semântica fornece uma forma de representar informações de maneira que possam ser lidas e entendidas por máquinas, permitindo novos tipos de automação e análise de dados (MATTHEWS, 2005).

As estruturas de redes de computadores são fundamentais para a infraestrutura dos sistemas web. Elas facilitam a comunicação e o compartilhamento de dados entre dispositivos distribuídos, sendo a espinha dorsal da internet. A compreensão dessas redes é vital para entender como os sistemas web operam e interagem (MACEDO, 2018).

2.3.2 Aplicativos Distribuídos

Sistemas distribuídos, como os sistemas web, consistem em componentes de software localizados em diferentes dispositivos de rede que se comunicam e coordenam suas ações ao processar requisições. Esses sistemas caracterizam-se pela separação de funcionalidades em diferentes camadas ou componentes, comumente referenciados como frontend e backend.

O frontend, geralmente, é a interface com a qual o usuário interage, enquanto o backend é a parte que processa as solicitações, gerencia dados e executa a lógica de negócios. Esta separação é um exemplo clássico de como os sistemas distribuídos operam, com diferentes componentes trabalhando em conjunto para fornecer uma experiência de usuário coesa e eficiente (GINIGE, 2002; MOREAU et al., 2010).

2.3.3 Progressive Web Apps

Progressive Web App (PWA) é uma abordagem moderna no desenvolvimento web que combina as melhores características das páginas web tradicionais com as qualidades dos aplicativos móveis. Os PWAs são aplicações web confiáveis, rápidas e envolventes, aproveitando as tecnologias web mais recentes para oferecer uma experiência de usuário semelhante à de aplicativos nativos, incluindo a capacidade de funcionar *offline*, receber notificações *push* e ser adicionado à tela inicial de dispositivos móveis (HUME, 2017).

Os PWAs são construídos usando tecnologias web padrão como HTML, CSS e JavaScript, mas com a inclusão de “*Service Workers*”, “*Web App Manifests*” e “*Application Shell Architecture*”. Os *Service Workers* são *scripts* que rodam em segundo plano e permitem recur-

sos como o *caching* de conteúdo para uso *offline* e o recebimento de notificações *push* (ATER, 2017). “*Web App Manifests*” permitem que a aplicação seja “instalada” na tela inicial do dispositivo, e a “*Application Shell Architecture*” garante uma experiência de usuário rápida e responsiva, mesmo com conexões de internet limitadas.

PWAs oferecem várias vantagens em comparação com aplicativos móveis nativos. Eles são mais fáceis de desenvolver e manter, pois utilizam uma base de código comum para todas as plataformas. Além disso, PWAs têm a capacidade de atingir um público mais amplo, pois são acessíveis por qualquer usuário com um navegador web, independentemente do dispositivo ou sistema operacional (HUME, 2017).

2.4 LINGUAGENS DE PROGRAMAÇÃO

As linguagens de programação são sistemas formais compostos por instruções que são usadas para produzir diversos tipos de saída, como programas de software. As linguagens de programação possuem paradigmas, como imperativo, orientado a objetos e funcional. Cada linguagem tem sua própria sintaxe e semântica, que determinam como as instruções são escritas e interpretadas (SEBESTA, 2019).

As linguagens de programação funcionam por meio de dois processos principais: compilação e interpretação. O processo de compilação transforma o código fonte escrito em uma linguagem de programação de alto nível em código de máquina que pode ser executado por um computador (AHO et al., 2006). Por outro lado, a interpretação envolve a execução direta de instruções de código fonte sem a necessidade de compilação prévia (LOUDEN; LAMBERT, 2011).

2.4.1 Frontend

O frontend abrange a parte da aplicação web ou móvel com a qual os usuários interagem diretamente. O desenvolvimento frontend envolve a criação de interfaces de usuário intuitivas e responsivas, utilizando tecnologias como HTML, CSS e JavaScript. O HTML é a base estrutural de qualquer página web, enquanto o CSS é usado para controlar o layout e o estilo visual dessas páginas (DUCKETT, 2011). Além disso, o JavaScript é fundamental para adicionar interatividade e dinamismo às páginas web, permitindo a criação de uma experiência de usuário mais rica e envolvente (FLANAGAN, 2020).

Além das tecnologias básicas, o desenvolvimento frontend moderno também envolve o uso de frameworks e bibliotecas como React, Angular ou Vue.js, que facilitam a construção de interfaces complexas e a gestão do estado da aplicação. Assim, o desenvolvimento frontend é uma área dinâmica e em constante evolução, focada na criação de experiências digitais atraentes e acessíveis para os usuários (BANKS; PORCELLO, 2017).

2.4.1.1 JavaScript

JavaScript é uma linguagem de programação essencial no desenvolvimento web, conhecida por sua flexibilidade e abrangência. Com características robustas, possui grande potencial para criar interações ricas e dinâmicas em aplicações web (CROCKFORD, 2008).

Inicialmente projetado para execução exclusiva no lado do cliente, o JavaScript evoluiu para ser utilizado também no desenvolvimento de servidor, um avanço impulsionado pelo advento do Node.js. Esta plataforma revolucionou o uso do JavaScript ao possibilitar sua execução no servidor para criar aplicações eficientes e escaláveis, ampliando significativamente as possibilidades da linguagem (CASCIARO; MAMMINO, 2016).

2.4.1.2 TypeScript

O TypeScript, conhecido por ser uma linguagem fortemente tipada, restringe operações entre valores de tipos distintos, exigindo que sejam do mesmo tipo ou convertidos adequadamente (FAIN; MOISEEV, 2019). Como um *superset* do JavaScript, o TypeScript é *open source* e enriquece o JavaScript com recursos de programação orientada a objetos. Isso inclui a utilização de variáveis tipadas, criação de classes e aplicação de conceitos como encapsulamento, herança, abstração e polimorfismo. Essas características tornam a arquitetura do código mais robusta e alinhada com as melhores práticas de programação. Finalmente, o código TypeScript é compilado para JavaScript, garantindo compatibilidade e ampla aplicabilidade (TypeScript, 2023).

Além de fortalecer as práticas de programação orientada a objetos no JavaScript, o TypeScript oferece vantagens significativas em termos de segurança de tipo e manutenção de código. A segurança de tipo do TypeScript ajuda a prevenir muitos erros comuns em JavaScript ao realizar verificações em tempo de compilação (CHERNY, 2019). Esta característica permite aos desenvolvedores identificar e corrigir erros mais rapidamente, melhorando a qualidade geral do código. A manutenção do código também é facilitada, pois o TypeScript proporciona uma documentação mais clara e legível, o que é particularmente benéfico em projetos maiores e mais complexos.

2.4.2 Backend

O desenvolvimento backend é o componente central da programação de aplicações web e móveis, responsável pela lógica de negócios, manipulação de dados e integração com sistemas de banco de dados. O backend tem um papel fundamental no desenvolvimento, especialmente em arquiteturas de microsserviços, pois essa abordagem permite criar sistemas mais flexíveis, escaláveis e fáceis de manter (NEWMAN, 2021).

A arquitetura de microsserviços tem se tornado uma escolha popular no desenvolvimento backend, devido à sua flexibilidade e modularidade. A implementação eficaz de micros-

serviços facilita o desacoplamento de sistemas, permitindo o desenvolvimento, teste e implantação independentes de diferentes partes de um sistema (RICHARDSON, 2018).

2.4.2.1 *Kotlin*

Kotlin, uma linguagem de programação moderna desenvolvida pela JetBrains, foi oficialmente lançada em 2016 e rapidamente ganhou popularidade, especialmente no desenvolvimento de aplicativos Android. Trata-se de uma linguagem que combina orientação a objetos e recursos funcionais, oferecendo uma sintaxe concisa e segurança de tipos, além de total interoperabilidade com Java (JEMEROV; ISAKOVA, 2017).

Kotlin é aplicável no desenvolvimento backend e multiplataforma, podendo ser utilizada em diferentes ambientes, desde servidores até aplicações *desktop* (GREENHALGH; SKEEN; BAILEY, 2021).

2.5 API

Uma API (Interface de Programação de Aplicações) atua como um intermediário entre diferentes sistemas de software, permitindo a comunicação e a interação entre eles. Esta interface facilita a troca de informações e funcionalidades por meio de ações específicas, ferramentas, padrões e protocolos, possibilitando que um sistema acesse e utilize as capacidades de outro (JACOBSON; BRAIL; WOODS, 2012).

As APIs permitem que os desenvolvedores criem aplicações mais ricas e complexas com menos esforço, reutilizando funcionalidades já existentes. Isso facilita o desenvolvimento ágil de software, onde a velocidade e a eficiência são essenciais (JACOBSON; BRAIL; WOODS, 2012).

2.5.1 **Rest**

As APIs REST são um estilo arquitetural para construir serviços na internet que permitem que diferentes sistemas se comuniquem entre si. Elas funcionam como intermediários que recebem pedidos de um cliente (como abrir uma página da web ou solicitar informações) e buscam esses dados em um servidor. Essas APIs são populares por serem simples de usar e eficientes, especialmente para a transferência de dados na web. Elas permitem que os sistemas se comuniquem de maneira padronizada, o que facilita o desenvolvimento e a integração de diversas aplicações (JACOBSON; BRAIL; WOODS, 2012).

2.6 BIBLIOTECAS E *FRAMEWORKS*

Bibliotecas consistem em coleções de implementações, tais como funções, classes ou procedimentos, destinadas à execução de tarefas comuns. Elas representam um agrupamento

de recursos projetados especificamente para abordar e solucionar uma variedade de problemas específicos (KERNIGHAN; PIKE, 1999).

A função essencial das bibliotecas é simplificar o processo de programação. Quando integradas a uma linguagem de programação, elas se tornam ferramentas extremamente úteis, proporcionando maior eficiência e reduzindo a ocorrência de erros nas atividades diárias dos desenvolvedores.

Utilizar bibliotecas traz inúmeros benefícios, incluindo a disponibilidade de funções pré-definidas que promovem uma abordagem modular à programação. Isso não apenas acelera o processo, permitindo a reutilização de código em diversos programas, mas também contribui para a escrita de códigos mais concisos e estruturados. Essa organização eficiente minimiza a probabilidade de erros e a necessidade de retrabalhos, além de simplificar significativamente a atualização de vários programas (PRESSMAN, 2010).

2.6.1 React

React, uma biblioteca JavaScript de código aberto, é amplamente utilizada na construção de interfaces de usuário. Desenvolvida pelo Facebook, ela se estabeleceu como uma das bibliotecas mais populares atualmente, sendo uma ferramenta essencial no desenvolvimento frontend. Aplicações notáveis que a utilizam incluem Whatsapp, Instagram e Airbnb (BANKS; PORCELLO, 2017).

O React viabiliza a criação de interfaces abrangentes por meio da composição de componentes pequenos e independentes. Essa biblioteca é reconhecida por sua abordagem declarativa, eficiência e flexibilidade. Para implementar o React, é necessário importar duas bibliotecas essenciais: ReactDOM, que integra o React com a DOM (Document Object Model), responsável pelos elementos da página, e React, que incorpora as funcionalidades principais da biblioteca (STEFANOV, 2021).

A criação de elementos no React é realizada por meio de funções JavaScript nativas da biblioteca. Uma característica distintiva do React é o uso do Virtual DOM. O Virtual DOM é uma representação em memória das informações do DOM, permitindo que o React mantenha uma cópia virtual de todos os dados do DOM. Quando ocorre uma alteração de estado, o React não re-renderiza todo o DOM, mas sim identifica e atualiza apenas os elementos alterados na sua cópia virtual. Isso resulta em uma manipulação mais rápida do DOM e, consequentemente, em um desempenho aprimorado. Além disso, o React utiliza o JavaScript XML (JSX), que possibilita escrever elementos com uma sintaxe semelhante ao HTML, tornando o processo de desenvolvimento mais intuitivo e acessível (BANKS; PORCELLO, 2017).

O principal fundamento do React é o conceito de componentização, que visa segmentar aplicações em partes distintas, autônomas e reutilizáveis. Os componentes no React recebem entradas denominadas “props” e, a partir delas, geram novos elementos React, conhecidos como JSX. Esta abordagem modular facilita a organização e a manutenção do código, além de promover a reutilização eficiente dos componentes (BERTOLI, 2017).

2.6.2 JpaRepository

O JpaRepository, disponibilizado pelo Spring Data JPA, fornece uma abstração para a camada de acesso a dados, permitindo a realização de operações de persistência sem a necessidade de implementar manualmente consultas e operações básicas de banco de dados. Por meio dessa interface, são disponibilizados recursos para operações CRUD (Create, Read, Update e Delete), paginação e ordenação de resultados, contribuindo para a redução de código repetitivo e para o aumento da produtividade e da facilidade de manutenção das aplicações (Spring, 2025).

2.6.3 SpringBoot

Spring Boot é um projeto construído sobre o popular *framework* Spring para Java. Ele foi projetado com o intuito de simplificar o processo de configuração e implantação de aplicações baseadas em Spring. O Spring Boot oferece uma maneira de criar aplicações Spring *standalone* que podem ser executadas diretamente, focando na simplificação do desenvolvimento e na produtividade (WALLS, 2015).

O principal objetivo do Spring Boot é facilitar o desenvolvimento de aplicações, automatizando muitas das tarefas de configuração e implantação. Segundo Carnell e Sánchez (2021), o Spring Boot é eficaz na construção de microserviços, reduzindo a necessidade de configurações XML e anotações extensas que são comuns no Spring Framework.

O Spring Boot é conhecido por sua capacidade de integração e extensibilidade. O *framework* oferece um vasto conjunto de recursos, como *starters* automáticos, autoconfiguração e capacidade de criar executáveis “*fat jars*”, que são essenciais para simplificar o desenvolvimento de aplicações (GUTIERREZ, 2016).

2.6.4 Liquibase

Liquibase é uma ferramenta de código aberto para gerenciamento e rastreamento de migrações de banco de dados, conhecida por sua versatilidade e eficiência. Ela facilita o controle de versões para esquemas de banco de dados e a implementação de mudanças de forma consistente em diferentes ambientes (AMBLER; SADALAGE, 2006).

A ferramenta permite definir mudanças de banco de dados de uma maneira abstrata e independente de banco de dados. O Liquibase utiliza formatos como XML, YAML, JSON e SQL para especificar mudanças no banco de dados, o que o torna compatível com vários sistemas de banco de dados (REEVES; VOXLAND; PICKERILL, 2010).

2.7 BANCO DE DADOS

Bancos de dados são um elemento fundamental na tecnologia da informação e no desenvolvimento de software, pois eles proporcionam um método sistemático para armazenar,

gerenciar e recuperar informações. Eles não apenas armazenam dados, mas também oferecem formas e ferramentas eficientes para consultar e manipular esses dados de maneira eficaz (SILBERSCHATZ; KORTH; SUDARSHAN, 2019).

2.7.1 Relacional

Os modelos de bancos de dados são essenciais para entender como os dados são organizados e acessados. O modelo relacional, introduzido em 1970 por E.F. Codd, revolucionou a maneira como os dados são estruturados e acessados em sistemas de banco de dados. Este modelo baseia-se no uso de tabelas para representar dados e suas relações (CODD, 1970).

Bancos de dados relacionais também trazem conceitos fundamentais, incluindo esquemas, chaves primárias e estrangeiras, e normalização. Além disso, há a relevante Linguagem de Consulta Estruturada (SQL) como um padrão para a interação com bancos de dados relacionais, evidenciando sua eficácia em operações de definição, manipulação e controle de dados (SILBERSCHATZ; KORTH; SUDARSHAN, 2019).

2.7.2 PostgreSQL

O PostgreSQL, frequentemente referido apenas como Postgres, é um sistema de gerenciamento de banco de dados objeto-relacional de código aberto, conhecido por sua robustez, escalabilidade e conformidade com padrões SQL. Ele foi desenvolvido para expandir a funcionalidade do sistema de banco de dados Ingres, adicionando suporte a tipos de dados complexos e à execução de funções dentro do banco de dados (STONEBRAKER; ROWE, 1986).

O PostgreSQL é lembrado por sua arquitetura robusta e sofisticada, além de um amplo conjunto de funcionalidades, sendo projetado para lidar com uma grande variedade de cargas de trabalho, desde aplicações simples a complexas (OBE; HSU, 2017).

2.8 GIT (VERSIONAMENTO)

O Git é um sistema de código aberto e distribuído para controle de versões que representa um marco fundamental no desenvolvimento de software moderno. Ele auxilia no rastreamento de alterações em arquivos, pois mantém um registro de todas as mudanças realizadas — aspecto de suma importância para o desenvolvimento (BLISCHAK; DAVENPORT; WILSON, 2016). A estrutura distribuída do Git traz grandes vantagens sobre os antigos sistemas. Enquanto estes centralizam os arquivos em um único servidor, o Git possibilita que cada colaborador possua uma cópia completa do repositório (Git, 2023).

2.8.1 GitHub

O GitHub representa uma ferramenta valiosa na colaboração e gestão de projetos de software. Ele se destaca não apenas como uma plataforma de hospedagem de código-fonte e controle de versão usando Git, mas também como um meio eficaz de facilitar a colaboração entre desenvolvedores. O GitHub tem a capacidade de armazenar repositórios, acompanhar mudanças e permitir uma colaboração eficiente em projetos de software, destacando sua aplicabilidade tanto em projetos de código aberto quanto em ambientes corporativos. Esta abordagem destaca a multifuncionalidade do GitHub, sublinhando sua importância como uma ferramenta indispensável no cenário atual do desenvolvimento de software (PIPINELLIS, 2015).

2.9 PROTOTIPAÇÃO

A prototipação é um processo fundamental no desenvolvimento de produtos, especialmente em áreas como *design* de software e engenharia. Este processo envolve a criação de modelos iniciais ou versões experimentais de um produto, permitindo que *designers* e desenvolvedores testem ideias e conceitos antes da produção final. A prototipação pode ser usada para explorar e validar ideias rapidamente, proporcionando um *feedback* essencial para aprimorar o *design* e a funcionalidade do produto final. Os protótipos não precisam ser perfeitos ou completos; pelo contrário, eles são ferramentas iterativas para aprendizado e refinamento (WARFEL, 2009).

A prototipação oferece vários benefícios, incluindo a economia de tempo e recursos, reduzindo o risco de falhas no desenvolvimento do produto final. É um método eficaz e de baixo custo para projetar interfaces de usuário; mesmo com técnicas simples, é possível obter *insights* valiosos sobre a usabilidade e a experiência do usuário. A prototipação não é apenas para *designers*, mas uma atividade colaborativa que envolve toda a equipe do projeto, incluindo *stakeholders* e usuários finais (SNYDER, 2003).

3 TRABALHOS RELACIONADOS

Com o objetivo de identificar alternativas viáveis para o gerenciamento do Capítulo, realizou-se uma pesquisa voltada para soluções existentes em sistemas utilizados para a gestão de ordens paramaçônicas, de modo a atender às necessidades apresentadas. Esta pesquisa concentrou-se na análise de sistemas web e aplicativos disponíveis nas lojas de aplicativos, como a App Store (Apple Inc., 2023) e a Google Play (Google LLC, 2023).

Para garantir uma revisão abrangente dos sistemas correlatos e identificar aqueles mais relevantes, foi adotado um protocolo de pesquisa detalhado, conforme descrito a seguir:

3.1 PROTOCOLO DE PESQUISA

3.1.1 Termos de Busca Utilizados

Foram utilizados termos de busca específicos para garantir a cobertura abrangente do tema. Alguns dos principais termos incluem:

- “Gerenciamento de Capítulo Ordem DeMolay”
- “Automação de processos administrativos”
- “Eficiência operacional em organizações juvenis”
- “Sistemas de gestão paramaçônicos”
- “Sistemas administrativos para organizações juvenis”

3.1.2 Período de Execução

A pesquisa foi realizada durante um período de duas semanas, em agosto de 2023. Esse período foi escolhido para permitir uma revisão completa e minuciosa das opções disponíveis.

3.1.3 Critérios de Inclusão/Exclusão

Foram definidos critérios claros para inclusão e exclusão dos sistemas encontrados:

- **Critérios de Inclusão:**
 - Sistemas voltados para a gestão de capítulos da Ordem DeMolay ou organizações juvenis similares.
 - Sistemas que ofereçam funcionalidades para automação de processos administrativos.
 - Sistemas disponíveis em plataformas web ou móveis.

- Sistemas com documentação acessível que permita avaliação detalhada.

- **CrITÉrios de Exclusão:**

- Sistemas que não apresentem funcionalidades claras ou verificáveis.
- Sistemas que não atendam aos requisitos levantados nas entrevistas e questionários.
- Sistemas pagos.

3.1.4 Seleção dos Resultados

A seleção dos sistemas seguiu um processo estruturado:

- Inicialmente, uma busca foi realizada em bases de dados acadêmicas, lojas de aplicativos App Store (Apple Inc., 2023) e a Google Play (Google LLC, 2023), e sites especializados.
- Os títulos e descrições dos sistemas foram revisados para verificar a relevância.
- Os sistemas selecionados foram instalados e utilizados para avaliar se atendiam aos requisitos levantados.
- Finalmente, os sistemas mais pertinentes foram incluídos na análise, garantindo uma cobertura ampla e relevante do tema.

Este protocolo de pesquisa detalhado assegurou que a revisão dos sistemas fosse completa e que os sistemas correlatos mais relevantes fossem considerados, proporcionando uma base sólida para o desenvolvimento deste projeto.

No decorrer da pesquisa, observou-se que existiam poucas opções de software que atendiam parcialmente às demandas do Capítulo. No âmbito dos sistemas web, foram identificados o “DeMolay Virtual - Crie o site de seu Capítulo DeMolay em 3 minutos!” (Sistema DeMolay Virtual, 2023) e o “SisMaçom - Sistema para Gestão de Loja Maçônica” (SisMaçom, 2023). Para dispositivos móveis, foi analisado o aplicativo “Maçonaria Virtual Mobile” (Maçonaria Virtual Mobile, 2023).

Observou-se que a maioria das soluções apresentadas não contempla integralmente as necessidades identificadas para o Capítulo Florianópolis nº 076 da Ordem DeMolay. Além disso, muitas dessas soluções não foram desenvolvidas com foco exclusivo nas peculiaridades da OD, o que pode resultar em limitações no atendimento às necessidades específicas do Capítulo.

É relevante salientar que essas alternativas foram analisadas como parte do processo de pesquisa e coleta de informações, servindo como base de comparação para o desenvolvimento do sistema proposto neste trabalho. As limitações e deficiências identificadas nas soluções existentes motivaram a criação de um sistema personalizado que pudesse atender de maneira mais abrangente às necessidades do Capítulo Florianópolis nº 076.

Para avaliar os sistemas mencionados, cada um foi instalado e utilizado localmente, visando verificar o atendimento aos requisitos identificados. Embora alguns sistemas não sejam gratuitos, eles oferecem um período de teste gratuito ou uma conta de demonstração por alguns dias, permitindo uma análise completa de suas funcionalidades. A Tabela 1 sintetiza o atendimento de cada sistema aos requisitos levantados.

Requisitos	DeMolay Virtual	SisMaçom	MVMobile
Gestão financeira	Sim	Sim	Não
Controle de frequência	Sim	Sim	Sim
Calendário de eventos	Não	Sim	Sim
Controle de acesso	Sim	Sim	Sim
Relatório de transparência	Sim	Não	Não
Acessível web e mobile	Não	Sim	Sim
Compatibilidade com DeMolay	Sim	Não	Não

Tabela 1 – Comparativo de funcionalidades dos sistemas

Fonte: Elaborado pelos autores, 2023.

3.2 DEMOLAY VIRTUAL - CRIE O SITE DE SEU CAPÍTULO DEMOLAY EM 3 MINUTOS!

O “DeMolay Virtual” é um sistema projetado especificamente para a gestão de Capítulos DeMolay. Este programa incorpora um conjunto completo de funcionalidades para uma administração eficaz, além de oferecer uma variedade de ferramentas para melhorar a integração entre os membros do Capítulo (Sistema DeMolay Virtual, 2023). A Figura 1 apresenta a interface do sistema.

Análise dos requisitos

Os requisitos estabelecidos pelo Capítulo que podem ser atendidos pelo sistema “DeMolay Virtual” incluem:

- Realização da gestão financeira por meio de um módulo dedicado;
- Controle de frequência das sessões;
- Gerenciamento de acesso por meio de perfis;
- Disponibilidade de relatórios que contribuem para a transparência financeira do Capítulo;
- Compatibilidade com a Ordem DeMolay.

Por outro lado, existem requisitos estabelecidos pelo Capítulo que o sistema “DeMolay Virtual” não consegue atender, tais como:

- Disponibilidade de um calendário para o cadastro de eventos;
- Acessibilidade por meio de plataformas web e dispositivos móveis.

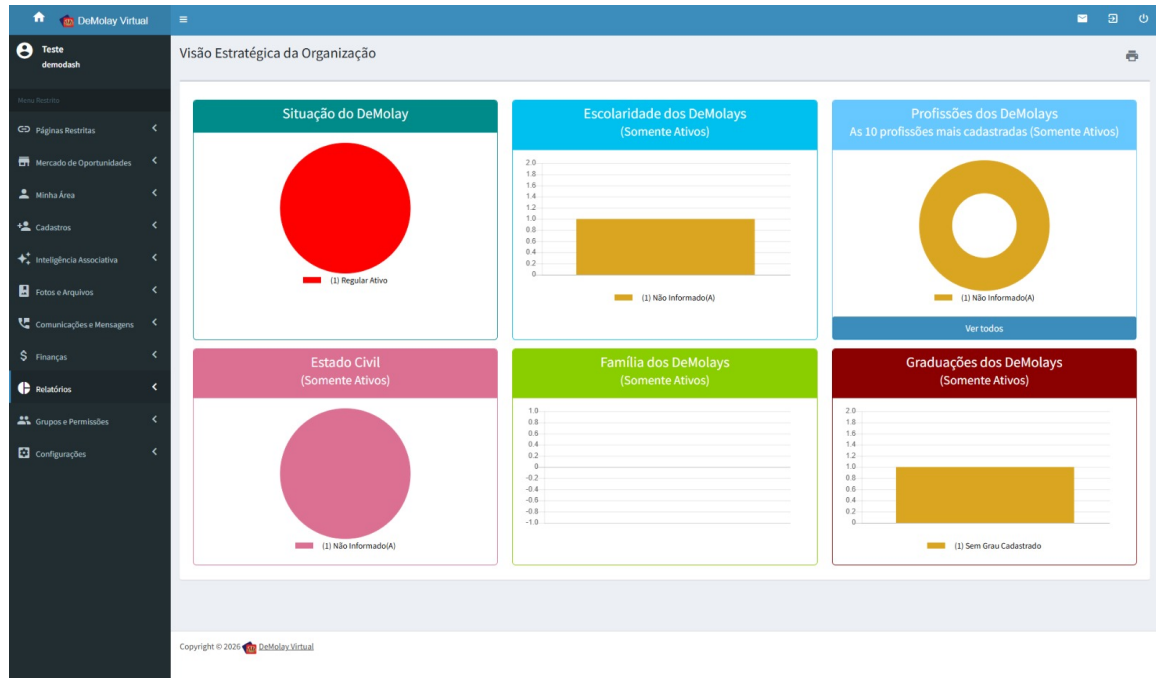


Figura 1 – Tela de relatórios do sistema DeMolay Virtual.

Fonte: (Sistema DeMolay Virtual, 2023).

3.3 SISMAÇOM - SISTEMA PARA GESTÃO DE LOJA MAÇÔNICA

“SisMaçom” é um sistema web privado desenvolvido para gestão de lojas maçônicas. Ele abrange uma gama de funcionalidades, dentre elas: controle da secretaria, chancelaria, tesouraria e presidência da loja (SisMaçom, 2023). A Figura 2 apresenta a interface do sistema.

Análise dos requisitos

Os requisitos estabelecidos pelo Capítulo que podem ser atendidos pelo sistema “Sis-Maçom” incluem:

- Realização da gestão financeira por meio de um módulo dedicado;
- Controle de frequência das sessões;
- Gerenciamento de acesso por meio de perfis;
- Disponibilidade de um calendário para o cadastro de eventos;

- Acessibilidade por meio de plataformas web e dispositivos móveis.

Por outro lado, existem requisitos estabelecidos pelo Capítulo que o sistema “SisMaçom” não consegue atender, tais como:

- Disponibilidade de relatórios que contribuem para a transparência financeira do Capítulo;
- Compatibilidade com a Ordem DeMolay.

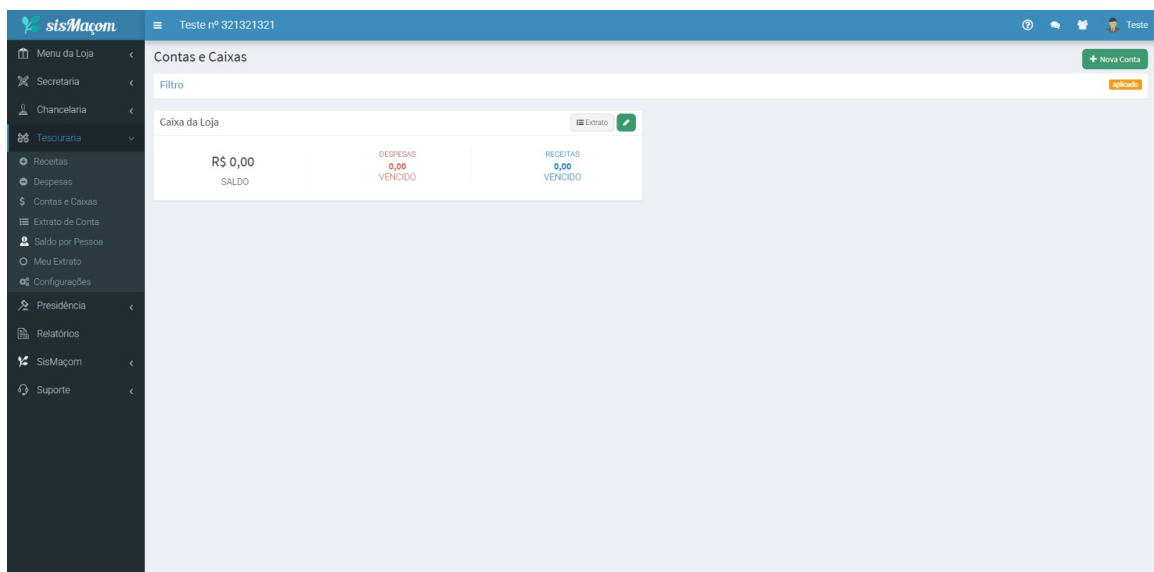


Figura 2 – Tela do módulo de tesouraria do sistema SisMaçom.

Fonte: (SisMaçom, 2023).

3.4 MAÇONARIA VIRTUAL MOBILE

“Maçonaria Virtual Mobile” é um sistema de gerenciamento desenvolvido por maçons. A solução propõe facilitar a gestão de lojas maçônicas de uma maneira simples (Maçonaria Virtual Mobile, 2023). A Figura 3 apresenta a interface do aplicativo.

Análise dos requisitos

Os requisitos estabelecidos pelo Capítulo que podem ser atendidos pelo sistema “Maçonaria Virtual Mobile” incluem:

- Controle de frequência das sessões;
- Gerenciamento de acesso por meio de perfis;
- Disponibilidade de um calendário para o cadastro de eventos;

- Acessibilidade por meio de plataformas web e dispositivos móveis.

Por outro lado, existem requisitos estabelecidos pelo Capítulo que o sistema “Maçonaria Virtual Mobile” não consegue atender, tais como:

- Realização da gestão financeira por meio de um módulo dedicado;
- Disponibilidade de relatórios que contribuem para a transparência financeira do Capítulo;
- Compatibilidade com a Ordem DeMolay.



Figura 3 – Tela inicial do aplicativo Maçonaria Virtual Mobile.

Fonte: (Maçonaria Virtual Mobile, 2023).

4 DESENVOLVIMENTO

O capítulo de desenvolvimento contempla o levantamento e a análise dos requisitos da aplicação, bem como a descrição do processo de implementação do sistema web desenvolvido.

4.1 ANÁLISE E LEVANTAMENTO DOS REQUISITOS DA APLICAÇÃO

A análise de requisitos em projetos de software é uma etapa crítica no desenvolvimento de sistemas de informação. O levantamento de requisitos é essencialmente o ponto de partida, em que as necessidades e expectativas dos clientes são identificadas e compreendidas (WIEGERS; BEATTY, 2013). Esta fase inicial visa garantir que a solução desenvolvida esteja alinhada ao que os usuários finais esperam do sistema.

Utilizar métodos e técnicas apropriados no levantamento de requisitos é essencial para garantir essa precisão, evitando assim as dificuldades comuns que surgem de uma definição imprecisa dos requisitos do aplicativo (SOMMERVILLE, 2011). Uma especificação clara e completa dos requisitos é vital para o sucesso do projeto, pois falhas nesta etapa podem levar a problemas significativos durante o desenvolvimento e após a implantação do sistema.

Neste projeto, adotou-se uma abordagem multifacetada para o levantamento de requisitos, combinando técnicas diversas para obter uma compreensão abrangente das necessidades dos usuários. A estratégia inclui entrevistas, questionários e prototipagem, cada uma oferecendo vantagens únicas na captura de requisitos.

- **Entrevistas** são valiosas para obter *insights* profundos e compreender o contexto de uso do sistema. Elas permitem um diálogo direto com os usuários, facilitando a identificação de suas necessidades e expectativas específicas (KRUG, 2014).
- **Questionários** são eficazes para coletar informações de um grande número de usuários de forma estruturada, permitindo análises quantitativas que podem validar ou refutar suposições sobre os requisitos do usuário (SAURO; LEWIS, 2009).
- **Prototipagem** é uma técnica usada para criar modelos preliminares do sistema, permitindo que os usuários interajam com representações concretas das funcionalidades propostas, fornecendo *feedback* essencial para o refinamento dos requisitos (BUXTON, 2010).

A combinação dessas técnicas visa garantir uma cobertura ampla e diversificada dos requisitos, aumentando a probabilidade de que o sistema final atenda efetivamente às necessidades e expectativas dos usuários.

4.1.1 Entrevistas

A técnica da entrevista, frequentemente utilizada no levantamento de requisitos, é uma abordagem tradicional e eficaz na fase inicial de coleta de dados. O entrevistador deve permitir que o entrevistado compartilhe suas ideias livremente, o que pode revelar *insights* valiosos sobre as necessidades do sistema. Distinguem-se dois tipos de entrevistas: a entrevista fechada, em que o analista segue um conjunto pré-definido de perguntas, e a entrevista aberta, que permite uma discussão mais livre sobre as expectativas dos interessados em relação ao sistema (SOMMERVILLE, 2011).

Embora esses dois tipos de entrevista sejam distintos, na prática, eles muitas vezes se sobrepõem. Uma entrevista pode começar com perguntas definidas e evoluir para um diálogo mais aberto à medida que surgem novos tópicos. A principal vantagem das entrevistas é a capacidade de coletar um conjunto rico de informações, enquanto a desvantagem reside na complexidade de analisar a grande quantidade de dados qualitativos e o potencial de informações conflitantes vindas de diferentes partes interessadas (WIEGERS; BEATTY, 2013).

É de suma importância o registro do conteúdo das entrevistas, seja por meio de anotações detalhadas ou utilizando dispositivos de gravação de áudio ou vídeo. Esta prática assegura que as informações valiosas coletadas durante as entrevistas sejam preservadas e possam ser referenciadas posteriormente no processo de desenvolvimento (SOMMERVILLE, 2011).

4.1.1.1 Plano de entrevista

A entrevista para o desenvolvimento deste trabalho ocorreu com o autor Nathan Werlich entrevistando o autor Victor Luiz, membro do Capítulo alvo. O principal objetivo da entrevista foi entender as dores de um DeMolay perante as atividades desempenhadas dentro do Capítulo e obter o conhecimento dos processos da OD, considerando que o primeiro autor não possui conhecimento suficiente do tema. O entrevistado é formando em Sistemas de Informação pela Universidade Federal de Santa Catarina e é membro da OD desde 2019, tendo passado pelos cargos de secretário, tesoureiro e presidente da comissão de tecnologia. Este trabalho adota a abordagem de entrevista aberta, buscando que o entrevistado fique livre para expor suas ideias. Para o registro da entrevista foram realizadas anotações manuais.

4.1.1.2 Objetivos da entrevista

1. Buscar informações sobre os processos da OD;
2. Identificar as principais dores nos processos diários do Capítulo;
3. Levantar as funcionalidades que são consideradas fundamentais para a gestão do Capítulo;
4. Descobrir sistemas que são utilizados hoje pelo Capítulo.

4.1.1.3 Resultados da entrevista realizada

A entrevista realizada com o Ir. (Irmão) Victor Luiz, membro do Capítulo alvo, proporcionou *insights* valiosos em relação aos processos e à gestão do Capítulo da OD. As respostas obtidas estão alinhadas aos objetivos propostos e são apresentadas a seguir:

1. Identificou-se que o Capítulo gera uma série de processos, tanto internos quanto externos. Os processos externos incluem atividades filantrópicas, participação e organização de eventos, emissão de ofícios e o levantamento de candidatos potenciais para iniciação na OD. Os processos internos abrangem o controle financeiro da instituição, o monitoramento da frequência dos membros, a transição de gestão, a logística de filantropias e eventos, a organização de calendários para sessões ritualísticas e administrativas, e o suporte a novos membros;
2. A falta de padronização e a ausência de manuais de procedimento são as principais dificuldades enfrentadas pelo Capítulo. Com cada transição de gestão, novos membros levam tempo para se adaptar ou acabam por reinventar processos, resultando em uma curva de aprendizagem prolongada e potencial perda de informações valiosas. Além disso, a constante mudança dos processos compromete a transparência e dificulta a consciência dos membros sobre a existência e a funcionalidade de certos procedimentos;
3. Fundamental para a gestão efetiva da OD é a manutenção do controle sobre os membros regulares, que estão em dia com suas obrigações financeiras e mantêm uma presença assídua nas sessões. Portanto, o controle financeiro e de frequência são essenciais. Uma padronização nos processos e maior transparência são necessárias para gerenciar de forma mais eficiente as operações da Ordem;
4. O Capítulo frequentemente reinicia o controle financeiro com novas planilhas do Google Sheets a cada mudança de gestão, com algumas gestões optando por verificar as transações diretamente por meio de aplicativos bancários. O registro de frequência é feito manualmente durante as sessões e, ocasionalmente, esses dados são transferidos para uma planilha do Google Sheets. No entanto, há sessões onde esse registro não é realizado. Para a indicação de novos membros, o processo é feito em papel e mantido nos arquivos físicos da instituição. Enquanto alguns processos ainda se baseiam no uso de papel, outros utilizam ferramentas digitais como Google Drive, Docs e Sheets, mas a falta de continuidade e monitoramento histórico é um desafio constante.

A partir dessas informações, pode-se concluir que o Capítulo enfrenta desafios significativos na gestão de seus processos, com uma necessidade urgente de digitalização e padronização para garantir a eficiência e a sustentabilidade a longo prazo de suas operações.

4.1.2 Questionários

Questionários são uma ferramenta eficaz para coletar informações de um amplo grupo de usuários, especialmente quando estão dispersos em diferentes locais. Um questionário bem projetado permite alcançar um número maior de pessoas de forma prática e eficiente, comparado às entrevistas que seriam impraticáveis em grande escala. Existem diversos formatos de questionários, incluindo opções de múltipla escolha, listas de verificação e questões abertas. Entretanto, é de extrema importância elaborar questionários que sejam fáceis e rápidos de responder (KRUG, 2014).

Durante a preparação do questionário, define-se claramente o tipo de informação desejada (SAURO; LEWIS, 2009). As perguntas devem ser simples, claras e concisas, com espaço suficiente para respostas descritivas. Agrupar questões relacionadas sob um título específico ajuda na organização e na análise dos dados coletados. Uma carta explicativa pode ser incluída para destacar a importância da pesquisa para o projeto.

Identificar quem receberá os questionários e fornecer instruções detalhadas sobre como preenchê-los, juntamente com um prazo para devolução, são práticas recomendadas. Após a coleta das respostas, é feita a consolidação e documentação das informações, permitindo a extração de *insights* valiosos para o projeto (SAURO; LEWIS, 2009).

4.1.2.1 Plano de questionário

O questionário neste trabalho foi aplicado no mês de outubro de 2023, com a participação de oito pessoas que fazem parte do Capítulo. O objetivo do questionário foi validar os pontos levantados na entrevista realizada anteriormente com o autor Victor Luiz. Os respondentes foram escolhidos levando em conta tempo como membro do Capítulo e cargos ocupados, buscando trazer uma visão mais ampla sobre o problema.

4.1.2.2 Perfil dos respondentes

O questionário foi direcionado especificamente aos integrantes do Capítulo Florianópolis nº 076 da Ordem DeMolay, abrangendo tanto aqueles que exerceram ou exercem cargos administrativos quanto os membros que possuem familiaridade, em qualquer grau de profundidade, com os procedimentos administrativos da organização. Este grupo inclui indivíduos de variados níveis de experiência e senioridade, proporcionando uma visão abrangente e detalhada sobre as operações internas do Capítulo.

4.1.2.3 Objetivos do questionário

O questionário tem como objetivo:

- Mapear e compreender em profundidade as dificuldades e desafios enfrentados pelos membros nos processos administrativos e operacionais do Capítulo Florianópolis nº 076.
- Avaliar a receptividade e o interesse dos membros na adoção de um sistema integrado para aprimorar o gerenciamento das atividades do Capítulo.

4.1.2.4 Resultados do questionário realizado

Após a análise das respostas ao questionário realizado com os membros do Capítulo Florianópolis nº 076, algumas observações e conclusões podem ser destacadas. Primeiramente, em relação ao controle financeiro, a grande maioria dos membros demonstrou insatisfação com os métodos atuais e indicou a necessidade de melhorias. Isso reforça a necessidade de uma solução estruturada que centralize e padronize as informações financeiras do Capítulo. Além disso, sugestões como a criação de um portal de transparência e a implementação de sistemas mais eficientes para controle de frequência e finanças refletem uma procura por maior clareza e eficiência nos processos administrativos.

Por outro lado, quanto à organização de eventos e filantropias, as respostas revelaram uma necessidade de melhorias na gestão e comunicação. A sugestão de desenvolver um calendário interativo, por exemplo, demonstra o desejo dos membros por ferramentas mais dinâmicas e interativas que facilitem o engajamento e a participação. Esse *feedback* sugere que, embora exista um bom nível de satisfação com os processos atuais, há um claro espaço para aprimoramento, especialmente em termos de tecnologia e comunicação digital.

Os detalhes das respostas fornecidas às perguntas do questionário estão devidamente elencados no apêndice “Resultados do Questionário” deste documento.

4.1.2.5 Personas

Este trabalho utiliza o conceito de personas para representar o perfil dos usuários que responderam o questionário. Personas são representações semi-fictícias de usuários típicos, criadas com base em dados reais, e são utilizadas extensivamente no desenvolvimento de produtos e no design de experiência do usuário (COOPER, 2004). A utilização de personas oferece uma série de vantagens no processo de *design* e desenvolvimento, possibilitando que as equipes mantenham o foco nas necessidades e experiências dos usuários finais.

Seguindo esse conceito, foram desenvolvidas as seguintes personas:

1. **Persona:** O Novato Engajado

Nome: Gabriel

Idade: 17 anos

Escolaridade: Ensino Médio Completo

Ocupação: Estudante

Tempo na Ordem DeMolay: 1 ano

Cargo Atual/Passado na Ordem: Membro sem cargo específico

Conhecimento dos Processos: Básico

Necessidades: Gabriel precisa de um sistema intuitivo que possa ajudá-lo a se familiarizar com os processos internos do Capítulo. Ele deseja um guia passo a passo que o auxilie a entender suas responsabilidades e a se preparar para assumir cargos futuros.

2. **Persona:** O Líder Experiente

Nome: Lucas

Idade: 21 anos

Escolaridade: Ensino Superior Incompleto — Administração

Ocupação: Estudante Universitário e estagiário

Tempo na Ordem DeMolay: 5 anos

Cargo Atual/Passado na Ordem: Ex-Mestre Conselheiro e Presidente da Comissão de Tecnologia

Conhecimento dos Processos: Avançado

Necessidades: Lucas procura eficiência e transparência. Ele valoriza um sistema que possa oferecer relatórios detalhados sobre o controle financeiro e de frequência dos membros. Como alguém que já liderou, ele quer garantir que as informações sejam facilmente transferíveis e compreensíveis para as futuras gestões.

3. **Persona:** O Membro Ativo

Nome: Rafael

Idade: 19 anos

Escolaridade: Técnico em Informática

Ocupação: Técnico em uma empresa de TI

Tempo na Ordem DeMolay: 3 anos

Cargo Atual/Passado na Ordem: Tesoureiro e Secretário

Conhecimento dos Processos: Moderado

Necessidades: Rafael precisa de um sistema que facilite o acompanhamento da frequência e das contribuições financeiras dos membros. Ele deseja um aplicativo que o ajude a manter o controle sem a necessidade de manusear papéis ou múltiplas planilhas, e que simplifique a transição entre os cargos do capítulo.

4.1.3 Histórias de Usuário

As histórias de usuário foram derivadas a partir de entrevistas e questionários aplicados aos membros do Capítulo Florianópolis nº 076. Esses instrumentos de coleta de dados foram cuidadosamente elaborados para captar as necessidades e expectativas dos usuários em relação ao novo sistema administrativo proposto. Durante as entrevistas, discutiram-se os desafios enfrentados e as melhorias desejadas. Os questionários, por sua vez, permitiram obter uma visão quantitativa e qualitativa das funcionalidades esperadas. Com base nas respostas obtidas, definiu-se um conjunto de histórias de usuário que representam as principais demandas e requisitos para o sistema. Essas histórias servem como base para o desenvolvimento, garantindo que o sistema atenda de forma eficaz às necessidades do Capítulo.

Seguem as histórias de usuário identificadas:

- Módulo de Membros
 1. US01 - Como administrador do Capítulo, quero inserir novos membros, para manter nosso registro de membros atualizado.
 2. US02 - Como administrador do Capítulo, quero editar informações dos membros, para manter os dados precisos e atualizados.
 3. US03 - Como membro do Capítulo, quero visualizar os perfis dos membros, para conhecer melhor os outros membros.
 4. US04 - Como administrador do Capítulo, quero poder inativar membros, para manter a base de dados limpa e relevante.
- Módulo de Perfis
 1. US05 - Como administrador do Capítulo, quero definir e gerenciar perfis de acesso, para controlar as permissões de cada membro dentro do sistema.
 2. US06 - Como administrador do Capítulo, quero atribuir perfis específicos aos membros, para assegurar que cada um tenha acesso aos recursos adequados de acordo com seu papel.
 3. US07 - Como membro do Capítulo, quero acessar apenas as funções permitidas pelo meu perfil, para garantir a segurança e a organização dos dados.
- Módulo de Gestão Financeira
 1. US08 - Como tesoureiro do Capítulo, quero registrar transações financeiras, para manter um registro preciso das finanças.
 2. US09 - Como tesoureiro, quero categorizar transações em entradas e saídas, para entender melhor a saúde financeira do Capítulo.
- Módulo de Controle de Frequência

1. US10 - Como secretário do Capítulo, quero registrar a presença dos membros em reuniões e eventos, para manter um registro atualizado da frequência.
2. US11 - Como administrador do Capítulo, quero visualizar relatórios de frequência, para avaliar o engajamento dos membros e planejar ações de incentivo.
3. US12 - Como membro do Capítulo, quero consultar meu histórico de frequência, para estar ciente da minha participação nas atividades.

- Módulo de Calendário

1. US13 - Como secretário do Capítulo, quero gerenciar o calendário de eventos, para organizar e comunicar as atividades programadas aos membros.
2. US14 - Como membro, quero visualizar eventos no calendário do Capítulo, para planejar minha participação e estar informado sobre as atividades futuras.
3. US15 - Como administrador do Capítulo, quero adicionar e editar eventos no calendário, para assegurar que todas as informações estejam atualizadas e acessíveis.

- Módulo de Transparência

1. US16 - Como membro, quero visualizar relatórios de gestão financeira e frequência, para entender a situação atual e o histórico do Capítulo.
2. US17 - Como tesoureiro, quero gerar relatórios de fechamento de gestão, para prover transparência e responsabilidade nas operações do Capítulo.

4.1.4 Prototipagem

O uso de protótipos de baixa fidelidade é fundamental em projetos de *design* e desenvolvimento por várias razões. Eles fornecem uma maneira rápida e econômica de materializar ideias, permitindo avaliações iniciais sem grandes investimentos (BUXTON, 2010). Esses protótipos facilitam a colaboração, tornando conceitos complexos mais acessíveis a todos os envolvidos, independentemente de seu conhecimento técnico (BUXTON, 2010).

Os protótipos elaborados podem ser encontrados no apêndice “Figuras” deste documento.

4.2 DESENVOLVIMENTO DA SOLUÇÃO

A solução proposta para o Capítulo Florianópolis nº 076 da OD integra tanto aspectos visuais quanto técnicos, visando aprimorar a gestão e a experiência do usuário. A identidade visual do aplicativo é cuidadosamente projetada para refletir os valores e a cultura do Capítulo, enquanto a arquitetura do software é construída para garantir eficiência, segurança e escalabilidade, abordando de forma efetiva as necessidades identificadas. Esta seção apresenta o

desenvolvimento do sistema web proposto, considerando a coerência com os conceitos estudados e com os requisitos definidos pelas histórias de usuário levantadas. A exposição parte da modelagem do sistema e avança para a sua implementação: primeiro a identidade visual e a modelagem (diagramas UML), que registram o projeto da solução, e em seguida a organização do projeto, a arquitetura e a implementação de cada história de usuário. O desenvolvimento ocorreu em um repositório hospedado no GitHub, organizado em dois diretórios principais: `backend/`, contendo o servidor de aplicação, e `frontend/`, contendo a interface do usuário.

O ambiente de desenvolvimento utilizado foi o IntelliJ IDEA para o módulo de backend e o Visual Studio Code para o módulo de frontend. O controle de versão foi realizado com *Git* seguindo o padrão *Conventional Commits* (Conventional Commits, 2024), cujo cumprimento é verificado automaticamente pelas ferramentas Husky e CommitLint a cada *commit*. O fluxo de trabalho adota *branches* de funcionalidade (com prefixo `feat/issue-N`) e *branches* de integração intermediárias para funcionalidades compostas por múltiplas etapas, integradas em *staging* ao final de cada módulo e em *main* a cada versão em produção.

A camada de servidor foi desenvolvida com a linguagem Kotlin e o *framework* Spring Boot 3.5, utilizando o banco de dados relacional PostgreSQL 17 como camada de persistência. O frontend foi desenvolvido com TypeScript e o *framework* Next.js 16, baseado na biblioteca React 19. Em produção, o frontend é implantado na plataforma Vercel e o backend na plataforma Render, enquanto o banco de dados é gerenciado pelo serviço Supabase.

4.2.1 Identidade do Aplicativo

O *design* e a estética de um aplicativo são essenciais para criar uma conexão imediata com seus usuários. Para o “DeMoDash”, a identidade visual foi meticulosamente planejada para refletir as tradições da OD, enquanto infunde modernidade e funcionalidade, indispensáveis em um software de gerenciamento.

Nome

O nome “**DeMoDash**” foi concebido com múltiplas camadas de significado. “DeMo”, uma abreviação direta de DeMolay, homenageia a fraternidade que une os membros do Capítulo Florianópolis. “Dash”, uma referência a “*dashboard*”, não só implica um centro de comando e controle para gestão eficiente, mas também remete a um antigo projeto vivido pelos membros no Capítulo, evocando o sentimento de nostalgia. A sonoridade de “ash” ecoa o sotaque manezinho da região, criando um vínculo cultural com o local de origem do aplicativo.

Paleta de Cores

A paleta de cores do “DeMoDash” foi escolhida para transmitir uma mistura de tradição e inovação (ver imagem a seguir). As cores incluem:

- **Rosa Antigo (#A63F65):** Um toque de calor e humanidade, quebrando a formalidade e adicionando uma dimensão amigável ao *design*.
- **Azul Cerúleo (#6872A6):** Representando confiança e calma, facilita a concentração e é agradável aos olhos durante o uso prolongado.
- **Azul-Escuro (#261340):** Confere profundidade e seriedade, servindo de base sólida para os outros elementos visuais.
- **Vermelho-Coral (#BF3939):** Acentua elementos que requerem atenção, sem ser agressivo.
- **Branco (#F2F2F2):** Oferece um contraste limpo e claro, essencial para a legibilidade e uma experiência de usuário refinada.

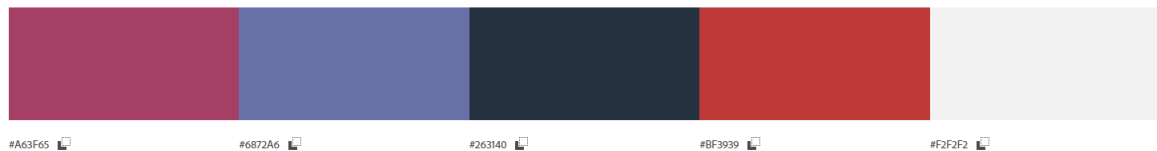


Figura 4 – Paleta de cores do aplicativo DeMoDash

Fonte: Elaborado pelos autores, 2023.

Logotipo

O logotipo incorpora, de forma inspirada, diversos elementos fundamentais da iconografia da Ordem DeMolay, cada um com um significado profundo:

- A Cruz Central Branca representa a retidão moral, a honra e a firmeza de caráter que devem orientar a vida de um DeMolay. Posicionada no centro da composição, simboliza os princípios que servem como base para as ações, decisões e relações dentro da irmandade.
- A Estrela Superior simboliza a luz, a sabedoria e a busca constante pelo aperfeiçoamento pessoal. Sua posição acima da cruz transmite a ideia de inspiração e propósito elevado, lembrando que os valores da Ordem devem sempre conduzir o caminho dos irmãos.
- O Círculo Colorido Fragmentado representa a união entre diferentes personalidades, talentos e histórias. Mesmo distintos, todos os elementos convergem para um mesmo centro, simbolizando fraternidade, cooperação e a força construída por meio da diversidade e da união do Capítulo.



Figura 5 – Logotipo do aplicativo DeMoDash

Fonte: Elaborado pelos autores, 2023

4.2.2 Modelagem do Sistema

A UML (*Unified Modeling Language*) é uma linguagem de modelagem padronizada usada para especificar, visualizar e documentar sistemas de software (BOOCH; RUMBAUGH; JACOBSON, 2005). Os diagramas a seguir documentam o modelo do sistema — suas entidades, componentes e comportamentos —, servindo de referência para a implementação descrita nas seções seguintes. Foram produzidos com a ferramenta PlantUML, e cada figura é acompanhada de legenda explicativa dos símbolos utilizados.

4.2.2.1 Diagramas Estruturais

Os diagramas estruturais descrevem os aspectos estáticos do sistema, representando componentes, classes e relacionamentos independentemente do tempo.

4.2.2.1.1 Diagrama de Componentes

O Diagrama de Componentes (Figura 23) apresenta os principais componentes do DeMoDash e as dependências entre eles. O sistema é composto por quatro componentes principais: o frontend Next.js, hospedado na Vercel; o backend Spring Boot, hospedado no Render; o banco de dados PostgreSQL, gerenciado pelo Supabase; e o serviço externo de e-mail Resend.

A comunicação entre o navegador e o frontend ocorre via HTTPS. O frontend, por sua vez, realiza chamadas REST ao backend via HTTPS, encaminhando o *cookie* de sessão em cada requisição. O backend acessa o banco de dados via JDBC (Spring Data JPA) e envia e-mails transacionais ao serviço externo Resend via SMTP; em caso de falha, recorre ao HTTPS como *fallback*.

4.2.2.1.2 Diagrama de Classes do Domínio

O Diagrama de Classes do Domínio (Figura 6) representa as entidades centrais do sistema e seus relacionamentos. As classes apresentadas correspondem às *data classes* Kotlin do pacote `domain/model/` de cada módulo, sem nenhuma dependência de *framework*. Os enumerados (*enums*) `TipoTransacao`, `CategoriaTransacao`, `StatusMembro` e `TipoEvento` capturam as restrições de domínio de forma tipada.

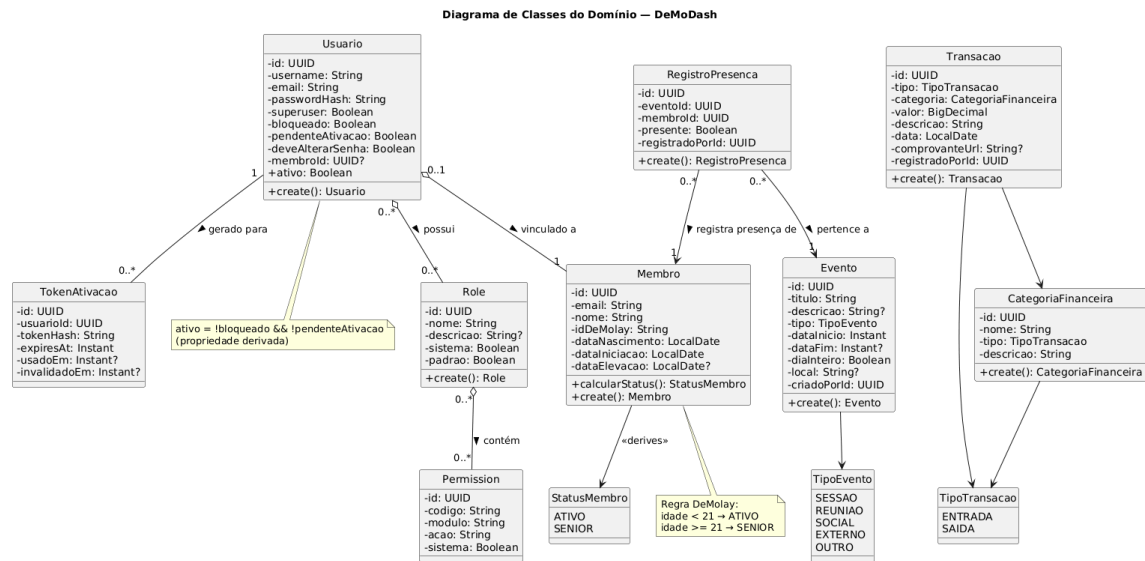


Figura 6 – Diagrama de Classes do Domínio do DeMoDash. **Legenda:** retângulos com três compartimentos representam classes (nome, atributos, métodos); retângulos com o estereótipo «enum» representam enumerados; linhas sólidas com multiplicidade indicam associação (ex.: 0..* = zero ou mais, 1 = exatamente um, 0..1 = opcional); setas com losango aberto na origem indicam composição fraca (agregação); setas tracejadas com rótulo «derives» indicam que o alvo é derivado do estado da classe de origem.

Fonte: Elaborado pelos autores, 2026.

Os relacionamentos centrais são: um `Usuario` pode estar vinculado a zero ou um `Membro` (o vínculo é obrigatório para membros e ausente para o superusuário administrador); um `Usuario` possui zero ou mais `Roles`; cada `Role` contém zero ou mais `Permissions`. A entidade `RegistroPresenca` vincula um `Membro` a um `Evento`, registrando sua presença (ou ausência) em uma chamada.

4.2.2.2 Diagramas Comportamentais

Os diagramas comportamentais descrevem como o sistema se comporta ao longo do tempo, capturando interações entre atores e o sistema e entre componentes internos.

4.2.2.2.1 Diagrama de Casos de Uso

O Diagrama de Casos de Uso (Figura 24) identifica os atores do sistema e as funcionalidades que cada um pode executar, mapeadas diretamente sobre as histórias de usuário US01 a

US17.

Os atores primários (à esquerda do sistema) são aqueles que iniciam as interações: Administrador (gestão de membros e perfis), Tesoureiro (registro de transações financeiras e relatórios), Secretário (registro de frequência e gestão do calendário) e Membro (consultas pessoais e visualizações). O ator secundário Serviço de E-mail (à direita) reage ao sistema, sendo acionado quando um novo usuário é criado para receber o *token* de ativação de conta.

4.2.2.2.2 Diagrama de Sequência

O Diagrama de Sequência (Figura 25) ilustra o fluxo de mensagens para o cenário *Registrar Transação Financeira*, descrevendo a interação entre todos os componentes envolvidos desde a ação do tesoureiro no navegador até a persistência no banco de dados.

O fluxo inicia com o tesoureiro preenchendo o formulário na página Next.js. A submissão invoca uma *Server Action*, que realiza a chamada POST `/v1/financeiro/transacoes` ao backend com o *cookie* de sessão. O `TransacaoController` verifica que o usuário possui `FINANCEIRO:REGISTRAR_ENTRADA` ou `FINANCEIRO:REGISTRAR_SAIDA`, delega ao `RegistrarTransacaoService`, que verifica a permissão granular por tipo de transação, cria o objeto de domínio via `Transacao.create()` e persiste via `TransacaoRepository`. Em caso de permissão insuficiente, retorna 403; em caso de sucesso, retorna 201 com o DTO da transação criada e o frontend redireciona para a listagem.

4.2.3 Organização do Projeto

O projeto é organizado como um *monorepo* com dois módulos principais: o backend, responsável pela lógica de negócio e persistência, e o frontend, responsável pela interface do usuário. Cada módulo possui estrutura, convenções e ferramentas próprios, descritos a seguir.

4.2.3.1 Backend

O backend segue a Arquitetura Hexagonal (também chamada de *Ports and Adapters*), detalhada na Seção 4.2.4.1, combinada com organização vertical por módulo de negócio. Cada módulo de negócio é um *slice* vertical independente, com a seguinte estrutura interna de pacotes:

1. `application/port/` — interfaces de casos de uso (*inbound ports*), que representam o contrato do que a aplicação sabe fazer. São implementadas pelas classes em `application/impl/`.
2. `application/impl/` — implementações dos casos de uso, anotadas com `@Service`. Dependem apenas de interfaces de domínio, nunca de JPA diretamente.

3. `domain/model/` — entidades e objetos de valor do domínio puro: Kotlin *data classes* sem dependência de nenhum *framework*. Validações de negócio são executadas no método de fábrica `create()` de cada entidade.
4. `domain/port/` — interfaces de repositório e serviços externos (*outbound ports*), que representam o que a aplicação precisa do mundo externo. São implementadas pelos adaptadores em `infrastructure/`.
5. `infrastructure/persistence/` — adaptadores de persistência: entidades JPA (`@Entity`), repositórios Spring Data e implementações das portas de saída (`@Repository`), além de mapeadores MapStruct para conversão entre modelo de domínio e entidade JPA.
6. `infrastructure/web/` — controladores REST (`@RestController`), DTOs de requisição/resposta e mapeadores MapStruct para conversão entre modelo de domínio e DTO.
7. `infrastructure/security/` ou `infrastructure/mail/` — adaptadores específicos do módulo quando necessário (por exemplo, `UserDetailsService` no módulo `usuario` e adaptadores de e-mail no módulo `email`).

Além dos módulos de negócio, o projeto possui dois pacotes transversais: `common/`, que concentra exceções de domínio, o `GlobalExceptionHandler`, o catálogo de permissões (`Permissao`) e utilitários de segurança; e `config/`, que contém a configuração do Spring Security, propriedades externas e os *runners* de inicialização (*bootstrap*) que criam o administrador padrão, as permissões e os perfis padrão na primeira execução da aplicação.

Os módulos de negócio implementados são:

- **auth** — autenticação (login, logout, alteração de senha);
- **usuario** — contas de usuário do sistema (criação, edição, ativação, inativação, atribuição de perfis);
- **role e permission** (dentro do módulo `role`) — perfis de acesso e permissões (RBAC);
- **email** — serviço assíncrono de envio de e-mail, com adaptadores SMTP, HTTP (*Resend*) e de *log* (modo de desenvolvimento);
- **status** — verificação de saúde da aplicação (*health check*);
- **ativacao** — fluxo de ativação de conta via *token* enviado por e-mail;
- **reset_senha** — fluxo de redefinição de senha iniciado pelo administrador;
- **membro** — registro de membros do Capítulo (entidade de domínio, regras de status ATIVO/SÊNIOR);
- **gestaofinanceira** — transações financeiras, categorização e relatórios de agregação;

- **calendario** — eventos do Capítulo (criação, edição, exclusão, listagem);
- **frequencia** — controle de presença dos membros em eventos.

4.2.3.2 Frontend

O frontend utiliza o App Router do Next.js, onde cada pasta dentro de `src/app/` corresponde a um segmento de rota. A documentação do React incentiva a co-localização de código relacionado (Meta Platforms, Inc., 2025), e esse princípio é adotado aqui por meio da convenção de pastas privadas com prefixo `_`: o Next.js exclui do roteamento qualquer pasta cujo nome comece com `_`, permitindo que componentes, hooks, ações e testes convivam com a página sem serem interpretados como rotas.

A estrutura principal de `src/` é organizada da seguinte forma:

1. `app/(app)/` — grupo de rotas protegidas (autenticadas), envolvidas por um *layout* comum que renderiza o *shell* da aplicação (cabeçalho, navegação lateral e navegação inferior). O grupo `(app)` não adiciona nenhum segmento à URL.
2. `app/login/`, `app/ativar/`, `app/resetar-senha/` — rotas públicas de autenticação, fora do grupo protegido.
3. `app/status/` — painel público de saúde do sistema.
4. Dentro de cada rota, pastas privadas com prefixo `_` concentram o código co-localizado:
 - `_components/` — componentes específicos da rota;
 - `_hooks/` — *hooks* SWR para re-busca de dados no cliente;
 - `_actions.ts` — *Server Actions* (`'use server'`);
 - `_utils.ts` — funções auxiliares locais;
 - `__tests__` — testes Jest co-localizados.
5. `components/ui/` — primitivos de interface reutilizáveis em toda a aplicação (botões, *inputs*, modais, *accordions*, *pagination*), construídos sobre Radix UI para acessibilidade e estilizados com Tailwind CSS.
6. `components/app/` — componentes de nível de aplicação (*logo*, cabeçalho de página, *breadcrumbs*).
7. `lib/api/` — camada de cliente HTTP tipada: `server-fetch.ts` (controlador central de requisições), `errors.ts` (hierarquia de erros tipados), `types.ts` (tipos compartilhados como `ServerActionResponse<T>`), `error-mapper.ts` e `form-result.ts`.
8. `utils/` — utilitários transversais: autenticação (`auth.ts`, `auth-server.ts`), formata-dores (`formatters.ts`), verificação de redirecionamento seguro (`redirect.ts`).

Diferentemente do modelo de organização por tipo de arquivo comum em projetos React mais simples, o DeMoDash agrupa código por funcionalidade e rota, o que facilita a manutenção e a localização de código relacionado em um projeto de múltiplos módulos.

4.2.3.3 Banco de Dados

O banco de dados é o PostgreSQL 17, gerenciado localmente via Docker Compose durante o desenvolvimento e pelo serviço Supabase em produção e homologação. As migrações de esquema são controladas pelo Liquibase, cujos *changelogs* estão organizados por módulo em `backend/src/main/resources/db/changelog/v0.1.0/<módulo>/`, facilitando rastrear quais alterações pertencem a cada funcionalidade.

Todas as tabelas seguem a convenção de nomenclatura `TB_*` (ex.: `TB_USUARIOS`, `TB_MEMBROS`, `TB_TRANSACOES_FINANCEIRAS`), e as tabelas de junção muitos-para-muitos utilizam o prefixo `RL_*` (ex.: `RL_USUARIO_ROLES`, `RL_ROLE_PERMISSIONS`).

Um ponto de segurança importante é o isolamento por schema: todas as tabelas da aplicação residem no schema `demodash`, e não no schema padrão `public`. O Supabase expõe o schema `public` automaticamente via sua API REST interna (*PostgREST*), portanto manter as tabelas em `demodash` as torna inacessíveis por essa API mesmo que a chave anônima do projeto seja exposta, eliminando essa superfície de ataque.

O Spring Session JDBC armazena os dados de sessão dos usuários autenticados nas tabelas `TB_SESSIONS` e `TB_SESSIONS_ATTRIBUTES`, também criadas via Liquibase dentro do schema `demodash`.

4.2.4 Arquitetura do Sistema

Esta subseção descreve as decisões arquiteturais que nortearam a construção do DeMoDash. O backend adota a Arquitetura Hexagonal para separar as regras de negócio da infraestrutura, enquanto o frontend segue o modelo de roteamento por sistema de arquivos do Next.js. As duas camadas se comunicam exclusivamente por meio de uma API REST, e a estratégia de gerenciamento de erros e permissões é compartilhada entre elas.

4.2.4.1 Arquitetura Hexagonal do Backend

A Arquitetura Hexagonal, proposta por Alistair Cockburn (COCKBURN, 2005), organiza o sistema em torno de um núcleo de domínio isolado, que se comunica com o mundo externo exclusivamente por meio de portas (*ports*) e adaptadores (*adapters*). O objetivo é garantir que as regras de negócio não dependam de nenhum detalhe de infraestrutura, como banco de dados, *framework* HTTP ou serviço externo.

No DeMoDash, cada módulo respeita a seguinte cadeia de dependências:

1. O controlador REST (adaptador de entrada) recebe a requisição HTTP, constrói um *command* e delega ao caso de uso pela sua interface (*inbound port*).
2. O caso de uso (implementação em *application/impl/*) aplica as regras de negócio, criando entidades de domínio e interagindo com o repositório apenas pela sua interface (*outbound port*).
3. O repositório (adaptador de saída, em *infrastructure/persistence/*) é o único ponto do sistema que conhece JPA. Ele implementa a interface de outbound port e mapeia o modelo de domínio para entidades JPA e vice-versa.

A Figura 7 ilustra essa estrutura para o módulo gestaofinanceira.

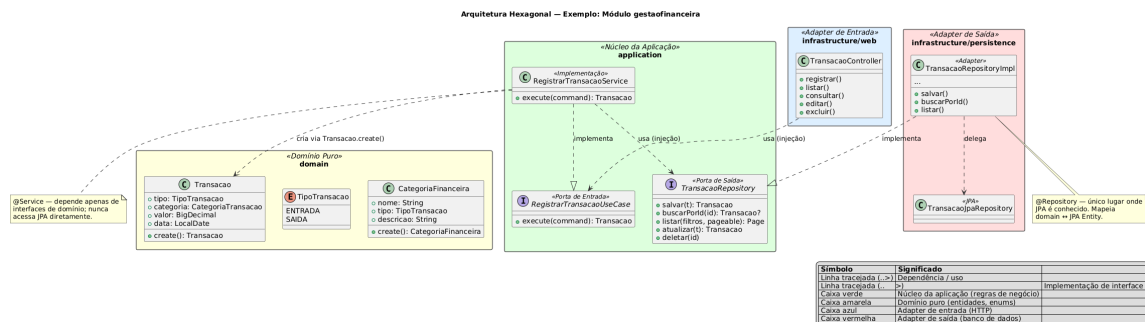


Figura 7 – Arquitetura hexagonal do módulo de gestão financeira. Caixas azuis representam adaptadores de entrada (HTTP); caixas verdes representam o núcleo da aplicação (regras de negócio); caixas amarelas representam o domínio puro (entidades e enums); caixas vermelhas representam adaptadores de saída (banco de dados). Linhas tracejadas com seta aberta indicam dependência/uso; linhas tracejadas com seta fechada indicam implementação de interface.

Fonte: Elaborado pelos autores, 2026

Para exemplificar, a Figura 89 apresenta a interface da porta de entrada para o caso de uso *Registrar Transação*. Trata-se de uma fun interface Kotlin (interface funcional de método único), que declara o contrato do caso de uso e o objeto de comando (*command*) que encapsula todos os seus parâmetros.

A Figura 90 apresenta a implementação desse caso de uso. Nela é possível observar a verificação de permissão por *tipo* de transação: registrar uma entrada exige a permissão `FINANCEIRO:REGISTRAR_ENTRADA`, enquanto registrar uma saída exige `FINANCEIRO:REGISTRAR_SAIDA`. Essa granularidade permite que um perfil do tipo *Tesoureiro* possa ter ambas as permissões, enquanto um perfil auxiliar tenha apenas uma delas.

A Figura 91 mostra o método `registrar()` do controlador REST correspondente. O controlador é deliberadamente enxuto: não contém lógica de negócio, apenas converte a requisição HTTP em um *command* e repassa ao caso de uso, recebendo de volta a entidade de domínio e convertendo-a para a resposta DTO.

4.2.4.2 Arquitetura do Frontend

O frontend combina três estratégias complementares de busca de dados, cada uma adequada a um cenário distinto:

- **React Server Components (RSC)** — componentes de servidor assíncronos que buscam dados diretamente no servidor Next.js antes de renderizar a página. São usados para carregamentos iniciais de página (ex.: o painel financeiro busca o resumo do mês antes de enviar o HTML ao navegador).
- **Server Actions** (`'use server'`) — funções executadas exclusivamente no servidor, invocadas tanto por RSC quanto por componentes de cliente. Constituem a camada de acesso à API do backend; cada arquivo `_actions.ts` agrupa as *actions* da rota correspondente.
- **SWR** — biblioteca de *stale-while-revalidate* usada para re-busca de dados no cliente em resposta a mudanças de estado da UI (por exemplo, os gráficos financeiros refazem a requisição ao alterar o período selecionado via parâmetro de URL).

O elemento central da camada de API é a função `serverFetch<T>()` (`src/lib/api/server-fetch.ts`), apresentada na Figura 92. Ela lê o *cookie session* armazenado pelo Next.js, encaminha-o ao backend como `Cookie: SESSION=<token>` e retorna sempre um `ServerActionResponse<T>` — um tipo discriminado `{ success: true; data: T }` ou `{ success: false; error: ErrorResponse }` — nunca lançando exceções, o que simplifica o tratamento de erros em toda a aplicação.

A gestão do *cookie* de sessão exige uma etapa de renomeação, detalhada na Figura 93. O backend Spring emite o *cookie* com o nome `SESSION` (maiúsculo, convenção do Spring Session). Em produção, os domínios do frontend (`demodash.com.br`) e do backend (`api.demodash.com.br`) são distintos, portanto o navegador não enviaria automaticamente o *cookie* do backend nas requisições ao frontend. A solução é o Next.js copiar o valor do *cookie* sob o nome `session` (minúsculo) — gerenciado pelo próprio frontend — e encaminhá-lo manualmente em cada chamada ao backend via `serverFetch`.

A proteção das rotas autenticadas é realizada pelo *middleware* `src/proxy.ts`, que é executado pelo servidor Next.js antes de cada requisição. Ele redireciona usuários não autenticados para `/login` e usuários que precisam alterar a senha para `/alterar-senha`, além de impedir que usuários já autenticados acessem `/login`. Como segunda linha de defesa, o *layout* do grupo (`app`) invoca a função `requireMe()` ao renderizar, que busca `GET /v1/me` no backend; se a sessão estiver inválida (resposta 401), o usuário é redirecionado para a limpeza de *cookie* e em seguida para `/login`.

4.2.4.3 Gerenciamento de Erros e Permissões

Todos os erros do backend seguem um formato JSON padronizado, produzido pelo `GlobalExceptionHandler` (`common/exception/GlobalExceptionHandler.kt`), conforme exemplificado na Listagem 4.1:

```

1 {
2   "status_code": 403,
3   "error_code": "TIPO_TRANSACAO_NAO_AUTORIZADO",
4   "message": "Usuario nao possui permissao para registrar transacoes do tipo 'SAIDA' .",
5   "user_action": "Voce nao tem permissao para registrar esse tipo de transacao.",
6   "details": []
7 }
```

Listing 4.1 – Formato padrão de resposta de erro da API.

O catálogo de permissões do sistema é definido no *enum* `Permissao` (`common/security/permissoao/Permissao.kt`), que funciona como **fonte única de verdade** para permissões. O código de cada permissão é derivado automaticamente da combinação de módulo e ação ("`${modulo.name}`:\$acao"), eliminando o risco de inconsistência por digitação. Um *runner* Maven (`exec:java@gen-permissions-ts`) gera automaticamente o arquivo TypeScript `frontend/src/lib/auth/permissions.generated.ts` a partir desse *enum*, mantendo frontend e backend sincronizados. A tarefa Maven `verify` verifica que os arquivos não divergiram, falhando o *build* em caso de inconsistência.

4.2.4.4 Infraestrutura de Implantação

A Tabela 2 resume a infraestrutura de implantação do sistema em dois ambientes: produção (*branch* `main`) e homologação (*branch* `staging`).

Tabela 2 – Infraestrutura de implantação do DeMoDash.

Camada	Plataforma	Produção	Homologação
Frontend	Vercel	demodash.com.br	staging.demodash.com.br
Backend	Render	api.demodash.com.br	staging.api.demodash.com.br
Banco de dados	Supabase	projeto demodash	projeto demodash-staging
DNS	Vercel	Nameservers	CNAME staging.api

Fonte: Elaborado pelos autores, 2026.

Ambos os ambientes de backend são implantados a partir de um único backend/Dockerfile e utilizam variáveis de ambiente próprias apontando para os respectivos projetos Supabase. O frontend no Vercel é implantado automaticamente a cada *push* nas *branches* `main` (produção) e `staging` (homologação).

4.2.5 Implementação das Histórias de Usuário

Esta seção descreve como cada história de usuário levantada foi implementada no sistema, relacionando as funcionalidades aos componentes de código correspondentes no backend e no frontend.

4.2.5.1 Módulo de Membros (US01–US04)

- **US01** — *Como administrador do Capítulo, quero inserir novos membros, para manter nosso registro de membros atualizado.*
- **US02** — *Como administrador do Capítulo, quero editar informações dos membros, para manter os dados precisos e atualizados.*
- **US03** — *Como membro do Capítulo, quero visualizar os perfis dos membros, para conhecer melhor os outros membros.*
- **US04** — *Como administrador do Capítulo, quero poder inativar membros, para manter a base de dados limpa e relevante.*

A entidade de domínio central deste módulo é a classe `Membro` (`modules/membro/domain/model/Membro.kt`), apresentada na Figura 87. Ela encapsula os dados cadastrais de um membro do Capítulo, incluindo datas de nascimento, iniciação e elevação (data em que o membro torna-se Sênior). A regra oficial DeMolay de classificação por idade é implementada no método `calcularStatus()`: membros com menos de 21 anos são classificados como **ATIVO**; com 21 anos ou mais, como **SÊNIOR**. O método de fábrica `create()` valida as consistências de data antes de instanciar o objeto, lançando `ValidacaoException` com os detalhes dos campos inválidos.

A criação de um membro é realizada pelo administrador por meio do endpoint `POST /v1/usuarios` (permissão `USUARIO:CRIAR`), que cria simultaneamente a entidade `Usuario` (conta de acesso ao sistema) e a entidade `Membro` (dados cadastrais), enviando um e-mail de ativação para que o novo membro defina sua senha inicial. As telas de listagem, criação de usuário e e-mail de ativação estão disponíveis nas Figuras 30 a 34. A edição dos dados é realizada via `PATCH /v1/usuarios/{username}` (permissão `USUARIO:EDITAR`), com telas ilustradas nas Figuras 33 a 48. A visualização de perfis (US03) é atendida pelos endpoints `GET /v1/me` (perfil próprio) e `GET /v1/usuarios/{username}` (qualquer membro com permissão `USUARIO:LISTAR`), com as respectivas páginas no frontend em `(app)/perfil` e `(app)/usuarios/[username]` (Figuras 45 e 46). A tela de login pode ser vista na Figura 26.

A inativação (US04) é realizada via `PATCH /v1/usuarios/{username}/desativar` (permissão `USUARIO:DESATIVAR`), com tela ilustrada na Figura 35. O sistema marca o campo `bloqueado = true` na entidade `Usuario`, o que faz com que `ativo = false` (propriedade

derivada: `!bloqueado && !pendenteAtivacao`). Além disso, todas as sessões ativas do usuário são invalidadas imediatamente, impedindo que o membro inativado continue navegando com uma sessão existente.

4.2.5.2 Módulo de Perfis (US05–US07)

- **US05** — *Como administrador do Capítulo, quero definir e gerenciar perfis de acesso, para controlar as permissões de cada membro dentro do sistema.*
- **US06** — *Como administrador do Capítulo, quero atribuir perfis específicos aos membros, para assegurar que cada um tenha acesso aos recursos adequados de acordo com seu papel.*
- **US07** — *Como membro do Capítulo, quero acessar apenas as funções permitidas pelo meu perfil, para garantir a segurança e a organização dos dados.*

O controle de acesso é implementado por meio de **RBAC** (*Role-Based Access Control*). O modelo de domínio é composto por duas entidades: **Role** (perfil de acesso, módulo `role`) e **Permission** (permissão individual). Um **Role** possui um conjunto de **Permissions** e um **Usuario** pode ter múltiplos **Roles**.

O CRUD de perfis (US05) é exposto pelo `RoleController` no caminho base `/v1/roles`:

- `POST /v1/roles` — cria um novo perfil (`ROLE:CRIAR`);
- `GET /v1/roles` — lista todos os perfis (`ROLE:LISTAR`);
- `PUT /v1/roles/{id}` — edita um perfil (`ROLE:EDITAR`);
- `DELETE /v1/roles/{id}` — exclui um perfil (`ROLE:DELETAR`).

A atribuição de perfis a usuários (US06) é realizada via `PUT /v1/usuarios/{username}/roles` (permissão `USUARIO:ATRIBUIR_ROLE`). O serviço `ValidarAtribuicaoRoleService` aplica um conjunto de invariantes de segurança na operação: (i) nenhum usuário pode modificar seus próprios perfis (guarda de auto-elevação); (ii) usuários sem a flag `superuser` não podem adicionar ou remover perfis que contenham permissões marcadas como `sistema = true`.

O controle de acesso efetivo (US07) é aplicado em dois níveis. No backend, os controladores e serviços utilizam a anotação `@PreAuthorize` do Spring Security com expressões do tipo `hasAuthority('MODULO:ACAO')`, habilitadas pela configuração `@EnableMethodSecurity(prePostEnabled = true)`. O superusuário possui um `MethodSecurityExpressionHandler` customizado que contorna essas verificações. No frontend, as funções utilitárias `hasAuthority(me, permission)` e `isSuperuser(me)` são usadas

em componentes de servidor para condicionalmente renderizar botões de ação e links de navegação — por exemplo, o botão “Nova transação” só é exibido a usuários com `FINANCEIRO:REGISTRAR_ENTRADA` ou `FINANCEIRO:REGISTRAR_SAIDA`. As telas correspondentes a essa funcionalidade estão disponíveis nas Figuras 37 a 44.

4.2.5.3 Módulo de Gestão Financeira (US08–US09)

- **US08** — *Como tesoureiro do Capítulo, quero registrar transações financeiras, para manter um registro preciso das finanças.*
- **US09** — *Como tesoureiro, quero categorizar transações em entradas e saídas, para entender melhor a saúde financeira do Capítulo.*

O módulo de Gestão Financeira é composto pelos seguintes elementos:

Entidade de domínio: a classe `Transacao` (`modules/gestaofinanceira/domain/model/Transacao.kt`) possui os campos `tipo` (`TipoTransacao`: `ENTRADA` ou `SAIDA`), `categoria` (`CategoriaTransacao`: `MENSALIDADE`, `DOACAO`, `EVENTO`, `MATERIAL`, `TAXA` ou `OUTROS`), `valor` (`BigDecimal`, com validação mínima de R\$ 0,01), `descricao`, `data` (data de competência, distinta da data de registro), `comprovanteUrl` (opcional, URL de comprovante) e `registradoPorId` (ID do usuário que registrou).

API de transações: o `TransacaoController` em `/v1/financeiro/transacoes` expõe operações CRUD completas:

- `POST /` — registra nova transação (permissão por tipo, conforme demonstrado na Seção 4.2.4.1);
- `GET /` — lista transações com filtros opcionais de tipo, categorias (múltiplas), data de início e data de fim, paginado (20 por página, ordenado por data descendente);
- `GET /{id}` — consulta uma transação pelo ID;
- `PATCH /{id}` — edita parcialmente uma transação (semântica `PATCH`: campos nulos mantêm o valor existente);
- `DELETE /{id}` — exclui uma transação (permissão `FINANCEIRO:DELETAR`).

API de agregação financeira: o `RelatorioFinanceiroController` em `/v1/financeiro` expõe quatro *endpoints* de leitura agregada, todos protegidos pela permissão `FINANCEIRO:LER_TUDO`:

- `GET /resumo?mes=yyyy-MM` — retorna saldo acumulado, total de receitas e total de despesas do mês;
- `GET /por-categoria?tipo={ENTRADA|SAIDA}&mes=yyyy-MM` — retorna a soma por categoria para o tipo e mês informados;

- GET /serie-temporal?tipo&granularidade&inicio&fim — retorna uma série temporal diária ou mensal (validação: máximo de 366 dias ou 120 meses de intervalo), com zeros preenchidos para períodos sem transações;
- GET /serie-mensal?inicio=yyyy-MM&fim=yyyy-MM — retorna receitas e despesas mensais lado a lado para construção de gráficos de balanço.

Interface do usuário: o frontend expõe o módulo financeiro em quatro páginas principais:

1. /financeiro — painel financeiro com cards de resumo mensal, despesas por categoria (gráfico de rosca), balanço mensal (gráfico de barras agrupadas) e últimas transações. O mês exibido é controlado pelo parâmetro de URL ?mes=yyyy-MM.
2. /financeiro/transacoes — listagem completa com filtros avançados (tipo, múltiplas categorias via *multi-combobox*, intervalo de datas), paginação e ações de edição e exclusão.
3. /financeiro/balanco — página de balanço mensal por categoria. Exibe o *card* de resumo e dois *cards* laterais com o detalhamento de receitas e despesas separados por categoria para o mês selecionado.
4. /financeiro/graficos — página de gráficos interativos com três abas: **Rosca** (distribuição por categoria via *PieChart* do MUI X Charts), **Linha** (evolução temporal via *LineChart*) e **Coluna** (balanço mensal e fluxo de caixa via *BarChart*). Todo o estado dos gráficos (aba ativa, tipo de dataset, período selecionado) é persistido na URL como *query params*, tornando cada visualização compartilhável por link e imune a perdas de estado ao recarregar a página.

As telas do módulo financeiro estão disponíveis nas Figuras 53 a 61.

4.2.5.4 Módulo de Controle de Frequência (US10–US12)

- **US10** — *Como secretário do Capítulo, quero registrar a presença dos membros em reuniões e eventos, para manter um registro atualizado da frequência.*
- **US11** — *Como administrador do Capítulo, quero visualizar relatórios de frequência, para avaliar o engajamento dos membros e planejar ações de incentivo.*
- **US12** — *Como membro do Capítulo, quero consultar meu histórico de frequência, para estar ciente da minha participação nas atividades.*

A entidade central do módulo é RegistroPresenca (modules/frequencia/domain/model/RegistroPresenca.kt), que vincula um membro a um evento com um indicador booleano de presença, conforme a Figura 88.

O registro de frequência (US10) é realizado pelo secretário por meio do endpoint `POST /v1/calendario/eventos/{id}/frequencia` (permissão `FREQUENCIA:REGISTRAR`), que recebe em uma única requisição a lista de todos os membros do evento com seus indicadores de presença. Esse *endpoint* é tratado pelo `FrequenciaEventoController`, que invoca o caso de uso `RegistrarPresencasUseCase` para persistir os registros em lote. As telas correspondentes estão disponíveis nas Figuras 66 a 68.

O relatório de frequência que abrange todos os membros (US11) é exposto pelo endpoint `GET /v1/frequencia/membros` (permissão `FREQUENCIA:LER_TODOS`), paginado e ordenado por total de presenças decrescente, atendido pelo `FrequenciaController` via o caso de uso `ListarFrequenciaMembrosUseCase`. Sua implementação está disponível nas Figuras 69 e 70.

A consulta do histórico pessoal (US12) é atendida pelo endpoint `GET/v1/frequencia/me`, disponível a qualquer membro autenticado sem exigir permissão adicional. O caso de uso `ConsultarMinhaFrequenciaUseCase` extrai o `membroId` do principal autenticado (`UsuarioPrincipal`) e retorna o histórico de eventos e presenças do membro. Sua tela está disponível na Figura 71.

4.2.5.5 Módulo de Calendário (US13–US15)

- **US13** — *Como secretário do Capítulo, quero gerenciar o calendário de eventos, para organizar e comunicar as atividades programadas aos membros.*
- **US14** — *Como membro, quero visualizar eventos no calendário do Capítulo, para planejar minha participação e estar informado sobre as atividades futuras.*
- **US15** — *Como administrador do Capítulo, quero adicionar e editar eventos no calendário, para assegurar que todas as informações estejam atualizadas e acessíveis.*

A entidade de domínio `Evento` (`modules/calendario/domain/model/Evento.kt`) representa uma atividade do Capítulo. Seus campos principais são: `titulo`, `descricao` (opcional), `tipo` (enumerado `TipoEvento`: `SESSAO`, `REUNIAO`, `SOCIAL`, `EXTERNNO` ou `OUTRO`), `dataInicio` e `dataFim` (ambos `Instant`), `diaInteiro` (booleano para eventos de dia inteiro) e `local` (opcional). O método de fábrica `create()` valida que `dataFim` não é anterior a `dataInicio`, lançando `ValidacaoException` em caso de violação.

O CRUD de eventos é exposto pelo `EventoController` no caminho base `/v1/calendario/eventos`:

- `POST /` — cria evento (`CALENDARIO:CRIAR`);
- `GET /` — lista eventos com filtros de tipo e intervalo de datas (`CALENDARIO:LER`);
- `GET /{id}` — consulta evento (`CALENDARIO:LER`);

- `PUT /{id}` — edita evento (`CALENDARIO:EDITAR`);
- `DELETE /{id}` — exclui evento (`CALENDARIO:DELETAR`).

A visualização dos eventos pelos membros (US14) é atendida pela página `(app)/calendario` no frontend, que exibe um calendário interativo com visualizações por mês e por dia (Figuras 72 a 76). A gestão e criação de eventos (US13 e US15) é realizada pelas páginas `(app)/calendario/novo` e `(app)/calendario/[id]/editar`, restritas a usuários com as permissões correspondentes (Figuras 77 a 81). O cadastro de participantes do evento (membros esperados para a chamada de frequência) é suportado pela funcionalidade de seleção de membros integrada ao formulário de criação e edição de eventos.

4.2.5.6 Módulo de Transparência (US16–US17)

- **US16** — *Como membro, quero visualizar relatórios de gestão financeira e frequência, para entender a situação atual e o histórico do Capítulo.*
- **US17** — *Como tesoureiro, quero gerar relatórios de fechamento de gestão, para prover transparência e responsabilidade nas operações do Capítulo.*

As histórias US16 e US17 foram implementadas de forma distribuída ao longo do módulo de Gestão Financeira, sem a criação de um módulo separado de transparência. Essa decisão de design surgiu durante o desenvolvimento: ao construir a API de agregação financeira e o painel principal, verificou-se que as mesmas telas já atendiam plenamente às necessidades de visualização (US16) e de geração de relatório de fechamento (US17), tornando desnecessária uma camada adicional.

US16 — Visualização de relatórios (como membro): a permissão `FINANCEIRO:LER_TUDO` possui `sistema = false`, o que permite ao administrador atribuí-la ao perfil padrão *Membro*. Com essa permissão, qualquer membro tem acesso ao conjunto completo de relatórios financeiros do Capítulo, servidos pelas três páginas do módulo:

- `/financeiro` — painel com KPIs do mês (saldo acumulado, total de receitas, total de despesas), gráfico de rosca de despesas por categoria e balanço mensal.
- `/financeiro/graficos` — visualizações interativas em três abas (Rosca, Linha e Coluna), com filtros de período e *dataset* persistidos na URL.
- `/financeiro/balanco` — detalhamento do balanço mensal por categoria financeira, separado em receitas e despesas.

Os *dashboards* desenvolvidos estão disponíveis nas Figuras 51 a 86.

US17 — Relatório de fechamento de gestão (como tesoureiro): o tesoureiro acessa as páginas do módulo de transparência e aplica o filtro de período correspondente à gestão em

encerramento. O sistema exibe o resumo completo — KPIs de saldo, receitas e despesas, além dos gráficos de distribuição e evolução — consolidado exclusivamente para o intervalo selecionado, servindo como relatório de fechamento sem necessidade de geração de um documento separado.

5 AVALIAÇÃO

Este capítulo apresenta a avaliação do sistema desenvolvido, abrangendo tanto a **validação** — certificar que o sistema atende às necessidades e expectativas reais de seus usuários — quanto a **verificação** — assegurar que o sistema foi construído corretamente. Segundo Pressman (PRESSMAN; MAXIM, 2020), a validação responde à pergunta “o produto certo está sendo construído?”. No DeMoDash, a validação ocorreu de forma incremental ao longo de todo o desenvolvimento: à medida que cada história de usuário era implementada, a funcionalidade correspondente era exercitada e apresentada aos membros do Capítulo DeMolay — os usuários finais do sistema —, que atuaram como fonte de requisitos e de *feedback*. Esse contato contínuo permitiu esclarecer dúvidas de domínio (por exemplo, a distinção entre membros *Ativos* e *Seniores*, as regras de uma *gestão* e o fluxo de registro de frequência por sessão) e ajustar o sistema de modo a aproximá-lo da rotina administrativa do Capítulo. Diversos refinamentos de fluxo e de interface foram solicitados e incorporados durante esse processo.

Além da validação conduzida junto aos usuários, o projeto adotou uma estratégia sistemática de **verificação** por meio de testes automatizados, integração contínua e barreiras de qualidade de código, detalhadas nas seções seguintes.

A Tabela 3 retoma o comparativo dos sistemas analisados no capítulo de Trabalhos Relacionados, acrescentando o DeMoDash e evidenciando que a solução desenvolvida atende a todos os requisitos levantados.

Tabela 3 – Comparativo de funcionalidades dos sistemas, incluindo o DeMoDash.

Requisitos	DeMolay Virtual	SisMaçom	MVMobile	DeMoDash
Gestão financeira	Sim	Sim	Não	Sim
Controle de frequência	Sim	Sim	Sim	Sim
Calendário de eventos	Não	Sim	Sim	Sim
Controle de acesso	Sim	Sim	Sim	Sim
Relatório de transparência	Sim	Não	Não	Sim
Acessível web e mobile	Não	Sim	Sim	Sim
Compatibilidade com DeMolay	Sim	Não	Não	Sim

Fonte: Elaborado pelos autores, 2026.

5.1 TESTES AUTOMATIZADOS

Os testes automatizados constituem a análise dinâmica do sistema, executando o código sob diferentes cenários para identificar e eliminar divergências em relação ao comportamento esperado antes da entrada em produção. O DeMoDash adota dois níveis de testes automatizados no backend — unitários e de integração — e testes unitários no frontend. A Tabela 4 resume a distribuição.

Tabela 4 – Distribuição dos testes automatizados do DeMoDash.

Camada	Ferramentas	Qtd. de arquivos de teste
<i>Backend</i> (unitários)	JUnit 5, Mockito, AssertJ	59
<i>Backend</i> (integração)	JUnit 5, Testcontainers, Mailpit	37
<i>Frontend</i> (unitários)	Jest, ts-jest	34
Total		130

Fonte: Elaborado pelos autores, 2026.

5.1.1 Testes de Unidade

Os testes de unidade verificam componentes isoladamente, com suas dependências substituídas por dublês (*mocks*). No backend, são escritos com JUnit 5 (JUnit Team, 2026), *mockito-kotlin* e AssertJ, e priorizam as regras de negócio das classes de aplicação. Um exemplo é `ValidarAtribuicaoRoleServiceTest`, que garante a governança de atribuição de perfis: um usuário não superusuário não pode **adicionar** nem **remover** um perfil que contenha permissões de sistema, enquanto o superusuário pode. Outro exemplo é `LoginRateLimitFilterTest`, que valida o filtro de limitação de taxa, inclusive a defesa contra contorno por parâmetros de caminho (`/v1/auth/login;jsessionid=...`).

No frontend, os testes são escritos com Jest (OpenJS Foundation, 2026) e executados em ambiente Node, exercitando a lógica das *server actions*, dos *schemas* de validação Zod e das funções utilitárias por meio da substituição das APIs do Next.js (`next/headers`, `next/navigation`) e de `global.fetch`. O arquivo de teste das ações de autenticação, por exemplo, verifica que um login bem-sucedido armazena o *cookie* de sessão como `HttpOnly`, que redirecionamentos abertos são bloqueados (`//evil.com` recai para `/`) e que os erros do backend (`INVALID_CREDENTIALS`, `VALIDATION_ERROR`) são mapeados corretamente.

5.1.2 Testes de Integração

Os testes de integração exercitam o sistema de ponta a ponta no backend, incluindo a camada web, a segurança, a persistência e o banco de dados real. Eles utilizam **Testcontainers** (AtomicJar / Docker, Inc., 2026), que provisiona automaticamente um contêiner PostgreSQL17-alpine para cada execução, sobre o qual as migrações reais do Liquibase são aplicadas no esquema *demodash* — ou seja, os testes validam contra o mesmo *layout* de esquema usado em produção. Um contêiner **Mailpit** captura os e-mails enviados, permitindo asserções sobre o conteúdo das mensagens de ativação de conta.

Todas as 37 classes de integração estendem `BaseIntegrationTest`, que provê a infraestrutura compartilhada: limpeza do banco entre testes, reinício dos *buckets* de limitação de taxa para isolamento e um auxiliar `loginAs` que retorna cabeçalhos autenticados. A título de exemplo, `AuthControllerIT` realiza chamadas HTTP completas e verifica: o retorno de um

cookie SESSION em um login válido; o erro 401 INVALID_CREDENTIALS para senha incorreta ou e-mail inexistente; a proteção contra fixação de sessão (um *cookie* forjado é substituído por um novo identificador); e o retorno 429 com cabeçalho Retry-After após exceder o limite de tentativas de login.

5.2 INTEGRAÇÃO CONTÍNUA E GARANTIA DE QUALIDADE

Para assegurar que cada alteração preserve a qualidade do código e o funcionamento dos testes, o projeto emprega **integração contínua** por meio do GitHub Actions, com cinco fluxos de trabalho (*workflows*) acionados em cada *pull request*:

- **Backend CI** — executa `./mvnw -B verify`, que dispara, em uma única etapa, a verificação de estilo com *ktlint*, os testes unitários (*surefire*), os testes de integração com Testcontainers (*failsafe*) e a checagem de sincronização do arquivo de permissões gerado para o frontend.
- **Frontend CI** — executa, em ordem, a checagem de formatação (Prettier), a análise estática com ESLint sob política de **zero avisos** (`--max-warnings=0`), os testes Jest e a *build* de produção.
- **Commit Lint** — valida que todas as mensagens de *commit* sigam o padrão *Conventional Commits*.
- **Docker and Config Validation** — analisa os Dockerfile com *Hadolint* e valida o `compose.yaml`.
- **Lint GitHub Workflows** — valida a sintaxe dos próprios *workflows* com *actionlint*.

Localmente, ganchos *Husky* reforçam a validação das mensagens de *commit* antes que cheguem ao repositório remoto. Esse conjunto de barreiras automatizadas garante que código fora do padrão ou que quebre algum teste não seja incorporado às *branches* staging e main.

5.3 CASOS DE TESTE FUNCIONAIS

Complementando os testes automatizados, foram conduzidos testes funcionais **manuais**, nos quais o sistema foi acessado sob a ótica do usuário final, exercitando os principais fluxos de cada módulo diretamente pela interface.

Os testes foram executados pelos dois autores do trabalho, no ambiente de homologação (`staging.demodash.com.br`), acessando a aplicação pelo navegador como um usuário administrador. Cada caso de teste foi elaborado a partir das histórias de usuário e percorre o fluxo principal do módulo correspondente: parte-se de um estado inicial conhecido, executa-se a ação descrita pela interface e observa-se o resultado apresentado pelo sistema (mensagem,

redirecionamento ou dado persistido). O critério de aprovação de cada caso foi a correspondência entre o resultado esperado e o resultado obtido; divergências identificadas ao longo do desenvolvimento foram corrigidas e o caso reexecutado até a aprovação. Nesta etapa não houve um painel externo de avaliadores: a participação dos membros do Capítulo ocorreu de forma incremental ao longo do desenvolvimento, como descrito na abertura deste capítulo. A Tabela 5 apresenta os casos de teste executados e seus resultados.

Tabela 5 – Casos de teste funcionais executados manualmente.

ID	Descrição	Resultado esperado	Resultado obtido
CT01	Autenticar usuário com credenciais válidas	Realizar login	Login realizado
CT02	Bloquear login após 5 tentativas inválidas	Retornar bloqueio (429)	Bloqueio retornado
CT03	Exigir troca de senha no primeiro acesso	Forçar alteração	Alteração exigida
CT04	Ativar conta via <i>token</i> de e-mail	Ativar a conta	Conta ativada
CT05	Cadastrar um novo membro	Cadastrar com sucesso	Membro cadastrado
CT06	Editar os dados de um membro	Editar com sucesso	Membro editado
CT07	Desativar um membro	Desativar com sucesso	Membro desativado
CT08	Registrar transação financeira (entrada/saída)	Registrar com sucesso	Transação registrada
CT09	Registrar frequência em uma sessão	Registrar presenças	Frequência registrada
CT10	Criar um evento no calendário	Criar com sucesso	Evento criado
CT11	Visualizar relatórios de transparência	Exibir os relatórios	Relatórios exibidos
CT12	Impedir atribuição de perfil de sistema	Negar a operação (403)	Operação negada

Fonte: Elaborado pelos autores, 2026.

5.4 DISPONIBILIZAÇÃO

A disponibilização compreende o acesso público ao código-fonte e a execução do sistema em ambiente de produção. O DeMoDash está hospedado na infraestrutura descrita na Tabela 2 e pode ser acessado diretamente pelos membros do Capítulo por meio de um navegador ou instalado como PWA em dispositivos móveis. O sistema em produção está disponível em <https://demodash.com.br>.

5.4.1 Código

O código-fonte do DeMoDash é mantido em um repositório público hospedado na plataforma GitHub, organizado como um *monorepo* que reúne o backend (Spring Boot/Kotlin) e o frontend (Next.js/TypeScript). O repositório pode ser acessado em <https://github.com/victorgevaerd/demodash>. Uma cópia também está disponível em <https://codigos.ufsc.br/nathan/demodash>.

5.4.2 Configuração de Ambiente

Por questões de segurança, credenciais e segredos não são versionados no repositório. O backend é configurado por variáveis de ambiente, cujos nomes e formato são documentados no arquivo `.env.example`; para executar o projeto localmente, cria-se um arquivo `.env` a partir desse modelo, preenchendo valores como as credenciais do banco PostgreSQL e os parâmetros do administrador inicial (`BOOTSTRAP_ADMIN_*`). A infraestrutura de implantação em produção e homologação — Vercel, Render e Supabase — já foi detalhada na Seção 4.2, na Tabela 2.

6 CONCLUSÃO

O desenvolvimento do DeMoDash compreendeu o estudo, o levantamento de requisitos, a implementação e a disponibilização de um sistema web para o Capítulo Florianópolis nº 076 da OD. Primeiramente, foi realizada uma pesquisa de fundamentação teórica abrangendo conceitos de sistemas web, PWA, as tecnologias adotadas e a própria OD. Em seguida, foram pesquisados sistemas existentes voltados à gestão de capítulos da OD e organizações paramaçônicas similares, encontrando-se poucas opções viáveis e nenhuma que atendesse integralmente aos requisitos levantados.

A partir de entrevistas e questionários aplicados aos membros do Capítulo, foram definidos os requisitos e as histórias de usuário que guiaram o desenvolvimento. Com base nisso, foram projetadas a identidade visual, a arquitetura do sistema e os diagramas UML, e então implementados os seis módulos de cadastro de membros, perfis de acesso, gestão financeira, controle de frequência, calendário de eventos e transparência.

Por fim, o sistema foi validado por meio de 130 testes automatizados — unitários e de integração — em backend e frontend, 12 casos de teste funcionais com todos os resultados obtidos iguais aos esperados, e integração contínua com cinco *workflows* automatizados no GitHub Actions. O sistema foi disponibilizado em ambiente de produção e homologação utilizando a infraestrutura Vercel, Render e Supabase.

Dois pontos importantes merecem destaque. Primeiro, a infraestrutura apoia-se em planos gratuitos dos serviços de hospedagem: o backend hospedado no Render está sujeito a *cold starts* por hibernação após períodos de inatividade, e o banco de dados no Supabase possui limites de armazenamento e de conexões simultâneas que podem se tornar insuficientes com o crescimento do uso, exigindo a migração para planos pagos. Segundo, embora o sistema tenha sido validado em ambiente de homologação com membros do Capítulo durante o desenvolvimento, sua utilização contínua em produção ainda não foi consolidada, o que representa uma etapa importante para confirmar a aderência do sistema à rotina administrativa real do Capítulo.

Conforme as informações apresentadas, conclui-se que o DeMoDash cumpriu todos os objetivos definidos e atendeu a todos os requisitos do escopo, entregando os seis módulos planejados e validando suas funcionalidades por meio de testes sistemáticos.

6.1 TRABALHOS FUTUROS

Como continuidade natural do projeto, quatro frentes de desenvolvimento se destacam.

A **expansão para outros Capítulos** da OD é a evolução mais significativa: o sistema foi concebido com escopo exclusivo para o Capítulo Florianópolis nº 076, mas a arquitetura modular e o controle de acesso por perfis oferecem uma base adequada para uma eventual adaptação multi-capítulo, permitindo que outras unidades da Ordem se beneficiem da solução.

O suporte a **notificações push** é outro aprimoramento relevante, aproveitando a infraestrutura PWA já existente para informar os membros sobre novos eventos, sessões ou alterações

de frequência diretamente em seus dispositivos.

O módulo de transparência pode ser enriquecido com **relatórios avançados**, como exportação de dados em PDF ou planilha e visualizações mais detalhadas do histórico financeiro e de frequência por período.

Por fim, o fluxo atual de registro de presença exige que um responsável opere o sistema durante a sessão para marcar manualmente a presença de cada membro, substituindo o livro de assinaturas físico por um processo igualmente manual no DeMoDash. A implementação de **registro de presença por QR Code** — ou por outro mecanismo de autoatendimento — permitiria que cada membro confirmasse sua própria presença ao chegar à sessão, eliminando a dependência de um operador dedicado, reduzindo o tempo de chamada e garantindo que o registro seja feito de forma imediata e precisa no sistema.

REFERÊNCIAS

AHO, A. V. et al. **Compilers: Principles, Techniques, and Tools**. [S.l.]: Pearson, 2006.

AMBLER, S. W.; SADALAGE, P. J. **Refactoring Databases: Evolutionary Database Design**. [S.l.]: Addison-Wesley Professional, 2006. ISBN 0321293533.

Apple Inc. **App Store**. 2023. Plataforma digital para distribuição de aplicativos. Acesso em: 14 nov. 2023. Disponível em: <https://www.apple.com/br/app-store/>.

ATER, T. **Building progressive web apps: bringing the power of native to the browser**. [S.l.]: O'Reilly Media, Inc., 2017.

AtomicJar / Docker, Inc. **Testcontainers for Java – Documentation**. 2026. Disponível em: <https://java.testcontainers.org/>.

AZEVEDO, C. M. M. de. Maçonaria: história e historiografia. **Revista USP**, n. 32, p. 178–189, 1996.

BANKS, A.; PORCELLO, E. **Learning React: functional web development with React and Redux**. [S.l.]: O'Reilly Media, Inc., 2017.

BERNERS-LEE, T.; FIELDING, R.; MASINTER, L. **Uniform Resource Identifier (URI): Generic Syntax**. 2005. RFC 3986, Internet Engineering Task Force. Available: <https://www.rfc-editor.org/rfc/rfc3986.html>.

BERTOLI, M. **React Design Patterns and Best Practices**. [S.l.]: Packt Publishing Ltd, 2017.

BLISCHAK, J. D.; DAVENPORT, E. R.; WILSON, G. A quick introduction to version control with git and github. **PLoS computational biology**, Public Library of Science, v. 12, n. 1, 2016.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The Unified Modeling Language User Guide**. 2. ed. [S.l.]: Addison-Wesley Professional, 2005. ISBN 9780321267979.

BUXTON, B. **Sketching user experiences: getting the design right and the right design**. [S.l.]: Morgan kaufmann, 2010.

Capítulo Florianópolis nº 076. **Comissões do Capítulo Florianópolis nº 076**. 2023. Portal 076. Acesso em: 16 abr. 2023. Disponível em: <https://portal.capflorianopolis.org.br/o-cap%C3%ADtulo/comiss%C3%B5es>.

Capítulo Florianópolis nº 076. **Eventos do Capítulo Florianópolis nº 076**. 2023. Portal 076. Acesso em: 16 abr. 2023. Disponível em: <https://portal.capflorianopolis.org.br/o-cap%C3%ADtulo/eventos>.

Capítulo Florianópolis nº 076. **Filantropias do Capítulo Florianópolis nº 076**. 2023. Portal 076. Acesso em: 16 abr. 2023. Disponível em: <https://portal.capflorianopolis.org.br/o-cap%C3%ADtulo/filantropias>.

Capítulo Florianópolis nº 076. **Nós - Capítulo Florianópolis nº 076**. 2023. Portal 076. Acesso em: 13 abr. 2023. Disponível em: <https://portal.capflorianopolis.org.br/o-cap%C3%ADtulo/n%C3%B3s>.

- CARNELL, J.; SÁNCHEZ, I. H. **Spring microservices in action**. [S.l.]: Simon and Schuster, 2021.
- CASCIARO, M.; MAMMINO, L. **Node.js Design Patterns**. [S.l.]: Packt Publishing Ltd, 2016.
- CASTRO, I. K. de M. et al. A maçonaria regular e as entidades paramaçônicas: Conceitos básicos e breve histórico. **Revista Científica Maçônica Ad Lucem**, Ad Lucem | São Luís, v. 1, n. 1, p. 4–9, 2021.
- CHERNY, B. **Programming TypeScript: making your JavaScript applications scale**. [S.l.]: O'Reilly Media, 2019.
- COCKBURN, A. **Hexagonal Architecture**. 2005. <https://alistair.cockburn.us/hexagonal-architecture/>. Acesso em: 14 jun. 2026.
- CODD, E. F. A relational model of data for large shared data banks. **Communications of the ACM**, ACM, v. 13, n. 6, p. 377–387, 1970.
- Conventional Commits. **Conventional Commits 1.0.0**. 2024. <https://www.conventionalcommits.org/en/v1.0.0/>. Acesso em: 08 jun. 2025.
- COOPER, A. **The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity**. [S.l.]: Sams Publishing, 2004.
- CROCKFORD, D. **JavaScript: The Good Parts: The Good Parts**. [S.l.]: O'Reilly Media, Inc., 2008.
- DUCKETT, J. **HTML & CSS: design and build websites**. [S.l.]: Wiley Indianapolis, IN, USA., 2011. v. 15.
- DUNCAN, H. E. **Hi, Dad!** Kansas City, Missouri: The International Supreme Council Order of DeMolay, 1970. ISBN 0-913504-69-6.
- FAIN, Y.; MOISEEV, A. **TypeScript Quickly**. [S.l.]: Manning Publications, 2019.
- FLANAGAN, D. **JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language**. [S.l.]: O'Reilly Media, 2020.
- GINIGE, A. Web engineering: managing the complexity of web systems development. In: **Proceedings of the 14th international conference on Software engineering and knowledge engineering**. [S.l.: s.n.], 2002. p. 721–729.
- Git. **Começando - Sobre Controle de Versão**. 2023. Git. Acesso em: 15 nov. 2023. Disponível em: <https://git-scm.com/book/pt-br/v2/Começando-Sobre-Controle-de-Versão>.
- Google LLC. **Google Play**. 2023. Plataforma digital para distribuição de aplicativos. Acesso em: 14 nov. 2023. Disponível em: <https://play.google.com>.
- Grande Conselho Estadual da Ordem DeMolay para Santa Catarina. **Ordem DeMolay em Santa Catarina**. 2016. DeMolay SC. Acesso em: 16 nov. 2023. Disponível em: <https://www.demolaysc.org.br/ordem-demolay-em-sc/>.
- GREENHALGH, D.; SKEEN, J.; BAILEY, A. **Kotlin Programming: The Big Nerd Ranch Guide**. [S.l.]: Pearson Technology Group, 2021.

- GUTIERREZ, F. **Pro Spring Boot**. [S.l.]: Apress, 2016. ISBN 9781484214329.
- HUME, D. A. **Progressive Web Apps**. [S.l.]: Manning Publications, 2017. ISBN 9781617294587.
- JACOBSON, D.; BRAIL, G.; WOODS, D. **APIs: A strategy guide**. [S.l.]: O'Reilly Media, Inc., 2012.
- JEMEROV, D.; ISAKOVA, S. **Kotlin in action**. [S.l.]: Simon and Schuster, 2017.
- JUnit Team. **JUnit 5 User Guide**. 2026. Disponível em: <https://junit.org/junit5/docs/current/user-guide/>.
- KERNIGHAN, B. W.; PIKE, R. **The practice of programming**. [S.l.]: Addison-Wesley Professional, 1999.
- KRUG, S. **Don't Make Me Think, Revisited: A Common Sense Approach to Web and Mobile Usability**. 3. ed. [S.l.]: New Riders, 2014.
- LOUDEN, K. C.; LAMBERT, K. A. **Programming languages: principles and practices**. [S.l.]: Cengage Learning, 2011.
- MACEDO, R. T. **Redes de Computadores**. [s.n.], 2018. Universidade Federal de Santa Maria. Acesso em: 15 nov. 2023. Disponível em: https://repositorio.ufsm.br/bitstream/handle/1/18351/Curso_Lic-Comp_Red-Computadores.pdf.
- MATTHEWS, B. **Semantic Web Technologies**. 2005. JISC Technology and Standards Watch.
- Maçonaria Virtual Mobile. **Maçonaria Virtual**. 2023. Maçonaria Virtual. Acesso em: 14 nov. 2023. Disponível em: <https://maconariavirtual.com.br/>.
- Meta Platforms, Inc. **Thinking in React**. 2025. Disponível em: <https://react.dev/learn/thinking-in-react>.
- MOREAU, L. et al. The foundations for provenance on the web. **Foundations and Trends® in Web Science**, Now Publishers, Inc., v. 2, n. 2–3, p. 99–241, 2010.
- NEWMAN, S. **Building microservices**. [S.l.]: O'Reilly Media, Inc., 2021.
- NEWTON, I. [Carta]. 1676. Cambridge, Inglaterra. Destinatário: HOOKE, R. Se eu vi mais longe, foi por estar de pé sobre ombros de gigantes.
- OBE, R. O.; HSU, L. S. **PostgreSQL: up and running: a practical guide to the advanced open source database**. [S.l.]: O'Reilly Media, Inc., 2017.
- OpenJS Foundation. **Jest Documentation**. 2026. Disponível em: <https://jestjs.io/docs/getting-started>.
- PIPINELLIS, A. **GitHub Essentials: Unleash the Power of Collaborative Development Workflow Using GitHub**. [S.l.]: Packt Publishing Ltd, 2015. ISBN 9781783553716.
- PRESSMAN, R. S. **Software Engineering: A Practitioner's Approach**. 7. ed. [S.l.]: McGraw-Hill International, 2010. ISBN 9780073375977.

PRESSMAN, R. S.; MAXIM, B. R. **Software Engineering: A Practitioner's Approach**. 9. ed. [S.l.]: McGraw-Hill Education, 2020.

REEVES, R.; VOXLAND, N.; PICKERILL, P. **Database Refactoring: Making Database Changes Safe and Simple**. [S.l.]: O'Reilly Media, 2010.

RICHARDSON, C. **Microservices Patterns: With Examples in Java**. [S.l.]: Manning Publications, 2018. ISBN 9781617294549.

SAURO, J.; LEWIS, J. R. **Quantifying the User Experience: Practical Statistics for User Research**. 1. ed. [S.l.]: Morgan Kaufmann, 2009. ISBN 978-0-12-384968-7.

SCDB | Centro Nacional de Liderança DeMolay. **Ordem DeMolay completa 40 anos de história no Brasil**. 2020. DeMolay Brasil. Acesso em: 16 nov. 2023. Disponível em: <https://www.demolaybrasil.org.br/noticias/ordem-demolay-completa-40-anos-de-historia-no-brasil>.

SCDB | Centro Nacional de Liderança DeMolay. **O que é a Ordem DeMolay?** 2023. DeMolay Brasil. Acesso em: 13 abr. 2023. Disponível em: <https://www.demolaybrasil.org.br/institucional/O-que-e-a-Ordem-DeMolay>.

SEBESTA, R. W. **Concepts of Programming Languages**. 12. ed. [S.l.]: Pearson, 2019.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Database System Concepts**. 7. ed. [S.l.]: McGraw-Hill Education, 2019.

SILVA, M. G. D. **Maçonaria no Brasil**. [S.l.]: Paco Editorial, 2015.

SisMaçom. **SisMaçom - Sistema para Gestão de Loja Maçônica**. 2023. Sismaçom. Acesso em: 14 nov. 2023. Disponível em: <https://sismacom.com.br/>.

Sistema DeMolay Virtual. **DeMolay Virtual — Crie o site de seu Capítulo DeMolay em 3 minutos**. 2023. DeMolay Virtual. Acesso em: 14 nov. 2023. Disponível em: <https://www.demolayvirtual.com.br/Portal/DemolayVirtual/Sistema.aspx>.

SNYDER, C. **Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces**. [S.l.]: Morgan Kaufmann, 2003. ISBN 9781558604111.

SOMMERVILLE, I. **Software Engineering**. 9. ed. Boston, MA, USA: Pearson Education, 2011. ISBN 9780137035151.

Spring. **JpaRepository API Documentation**. 2025. <https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html>. Acesso em: 20 jun. 2025.

STEFANOV, S. **React: Up & Running: Building Web Applications**. [S.l.]: O'Reilly Media, Inc., 2021. ISBN 9781492051466.

STONEBRAKER, M.; ROWE, L. A. The design of postgres. **ACM Sigmod Record**, ACM New York, NY, USA, v. 15, n. 2, p. 340–355, 1986.

TypeScript. **TypeScript Official Documentation**. 2023. Acesso em: 14 nov. 2023. Disponível em: <https://www.typescriptlang.org/docs/handbook/intro.html>.

WALLS, C. **Spring Boot in Action**. [S.l.]: Manning Publications, 2015. ISBN 9781617292545.

WARFEL, T. Z. **Prototyping: A Practitioner's Guide**. [S.l.]: Rosenfeld Media, 2009. ISBN 9781933820224.

WIEGERS, K. E.; BEATTY, J. **Software Requirements**. 3. ed. [S.l.]: Pearson Education, 2013. ISBN 9780735679665.

APÊNDICE A – RESULTADOS DO QUESTIONÁRIO

As respostas não podem ser editadas

Pesquisa - Capítulo Florianópolis nº 076

Olá! Somos Nathan Sargon Werlich e Victor Gevaerd Luiz, graduandos de **Sistemas de Informação na Universidade Federal de Santa Catarina** e esta pesquisa faz parte do nosso **Trabalho de Conclusão de Curso (TCC)**.

O objetivo desta pesquisa é coletar informações sobre as **dificuldades encontradas pelos membros dentro dos processos do Capítulo Florianópolis nº 076**.

- O público-alvo dessa pesquisa são membros do Capítulo Florianópolis nº 076 da Ordem DeMolay que já ocuparam algum cargo administrativo ou que conhecem os processos administrativos da Instituição (com qualquer grau/nível de senioridade).

Este formulário coletará dados que só serão utilizados para fins da pesquisa e desenvolvimento do projeto, bem como seus possíveis desdobramentos, seguindo a [Lei Geral de Proteção de Dados Pessoais \(LGPD\), Lei nº 13.709/2018](#). Ao prosseguir, você estará concordando com esses termos de utilização.

- O formulário leva em média de **8 a 10 minutos** para ser respondido.

Se tiver alguma dúvida ou interesse em colaborar mais profundamente, fique à vontade para entrar em contato conosco pelos e-mails: **victorgevaerdluiz@gmail.com** ou **nathanwerlich@gmail.com**

Agradecemos muito pela sua participação!

*** Indica uma pergunta obrigatória**

Figura 8 – Pesquisa - Capítulo Florianópolis nº 076

Fonte: Elaborado pelos autores, 2023.

A.1 PERGUNTAS ELABORADAS PARA O QUESTIONÁRIO

1. Estou ciente
2. Você é membro DeMolay do Capítulo Florianópolis nº 076 da Ordem DeMolay?
3. Em qual faixa etária você melhor se enquadra?
4. Qual é o seu nível de escolaridade?
5. Qual sua profissão/formação?
6. Há quanto tempo você iniciou na Ordem DeMolay?
7. Quais cargos administrativos dentro de um Capítulo DeMolay você já assumiu?
8. Você está familiarizado com os processos do Capítulo Florianópolis nº 076?

9. Você sabe encontrar o calendário com facilidade?
10. Você consegue compreender com facilidade as informações do calendário?
11. Você está satisfeito com o calendário atual ou acredita que poderia ser melhorado?
12. Você sabe encontrar o controle de frequência com facilidade?
13. Você consegue compreender com facilidade as informações do controle de frequência?
14. Você está satisfeito com o controle de frequência atual ou acredita que poderia ser melhorado?
15. Você sabe encontrar o controle financeiro com facilidade?
16. Você consegue compreender com facilidade as informações do controle financeiro?
17. Você está satisfeito com o controle financeiro atual ou acredita que poderia ser melhorado?
18. Você tem alguma sugestão de melhoria para os processos* citados anteriormente?

A.2 RESPOSTAS COLETADAS

A.2.1 Respondente 1

1. Sim
2. Sim
3. 22 a 25 anos (Sênior)
4. Ensino superior incompleto
5. Medicina Veterinária
6. 5 a 9 anos
7. Presidente (Mestre Conselheiro), Vice-presidente (1º Conselheiro), Vice do vice-presidente (2º Conselheiro), Tesoureiro, Auditor, Presidente da comissão de Eventos, Presidente da comissão de Filantropias, Presidente da comissão de Sindicância e Ritualística, Membro de comissão
8. Sim, conheço bem e consigo encontrar as informações com facilidade
9. Não
10. Não

11. Não, estou insatisfeito e acho que poderia ser melhorado
12. Não
13. Não
14. Não, estou insatisfeito e acho que poderia ser melhorado
15. Não
16. Não
17. Não, estou insatisfeito e acho que poderia ser melhorado
18. Poderia-se criar um local em que as informações pudessem ficar centralizadas e de fácil acesso (assim como a tentativa do Ir. Gabriel Pescador tentou com o Dashboard, mas com uma maneira mais fácil e fluida de inserir informações pela administração capitular, considerando que nem sempre terão conhecimento sobre o desenvolvimento da planilha/aplicativo); antigamente era feito em planilhas independentes dentro da Nuvem, mas que acabavam não se conversando e gerando necessidade de mão de obra pra inserir e interpretar posteriormente; existe uma tentativa do SisDM de integrar a frequência e a tesouraria no sistema, mas nunca foi tentado utilizar pelo capítulo, ao meu conhecimento. Acredito que exista a necessidade da transparência desses dados pelos DeMolays ativos, porém até o momento não foi encontrada maneira que pudesse satisfazer os ativos e permanecer dentro do Capítulo de uma gestão à outra. Entretanto acredito ainda que essas informações sejam muito importantes por decorrerem em função de atividades importantes ao Capítulo como em eleições e iniciação ou elevação. Apesar do apresentado, acredito que os ativos não deem a devida importância para esses processos por acabarem não entendendo a finalidade de observá-las para seu processo dentro da sua vida DeMolay por existir uma cultura de “quebra de pré-requisito” considerando a constante falta de Graus DeMolay ativos.

A.2.2 Respondente 2

1. Sim
2. Sim
3. 16 a 18 anos (Ativo)
4. Ensino médio completo
5. Estudante
6. 2 a 4 anos

7. Vice do vice-presidente (2º Conselheiro), Membro de comissão
8. Um pouco, conheço por cima mas não sei encontrar com facilidade
9. Não
10. Não
11. Não, estou insatisfeito e acho que poderia ser melhorado
12. Não
13. Não
14. Não, estou insatisfeito e acho que poderia ser melhorado
15. Não
16. Não
17. Não, estou insatisfeito e acho que poderia ser melhorado
18. Acho importante incluir também os “guias” de cada comissão.

A.2.3 Respondente 3

1. Sim
2. Sim
3. 19 a 21 anos (Ativo)
4. Ensino superior incompleto
5. Analista de sistemas
6. 5 a 9 anos
7. Presidente (Mestre Conselheiro), Vice-presidente (1º Conselheiro), Vice do vice-presidente (2º Conselheiro), Presidente da comissão de Eventos, Presidente da comissão de Filantropias, Presidente da comissão de Publicidade e Marketing, Presidente da comissão de Sindicância e Ritualística
8. Sim, conheço bem e consigo encontrar as informações com facilidade
9. Não
10. Não
11. Não, estou insatisfeito e acho que poderia ser melhorado

12. Não
13. Não
14. Não, estou insatisfeito e acho que poderia ser melhorado
15. Sim
16. Não
17. Não, estou insatisfeito e acho que poderia ser melhorado
18. Implementar um sistema de controle de frequência com capacidade de visualização e geração de relatórios. Criar um portal de transparência para consultar despesas e receitas, com funcionalidade de geração de relatórios para auditorias. Desenvolver um calendário interativo onde os membros possam adicionar eventos, com notificações automáticas e opção de confirmação de presença.

A.2.4 Respondente 4

1. Sim
2. Sim
3. 22 a 25 anos (Sênior)
4. Ensino superior incompleto
5. Desenvolvedor de Software
6. 2 a 4 anos
7. Secretário, Tesoureiro, Presidente da comissão de Tecnologia, Membro de comissão
8. Um pouco, conheço por cima mas não sei encontrar com facilidade
9. Não
10. Sim
11. Não, estou insatisfeito e acho que poderia ser melhorado
12. Não
13. Sim
14. Não, estou insatisfeito e acho que poderia ser melhorado
15. Não

- 16. Não
- 17. Não, estou insatisfeito e acho que poderia ser melhorado
- 18. n/a

A.2.5 Respondente 5

- 1. Sim
- 2. Sim
- 3. 19 a 21 anos (Ativo)
- 4. Ensino superior incompleto
- 5. Sistemas de informação
- 6. 2 a 4 anos
- 7. Vice-presidente (1º Conselheiro), Vice do vice-presidente (2º Conselheiro), Secretário, Tesoureiro, Presidente da comissão de Eventos, Presidente da comissão de Tecnologia, Presidente da comissão de Publicidade e Marketing, Presidente da comissão de Sindicância e Ritualística, Membro de comissão
- 8. Sim, conheço bem e consigo encontrar as informações com facilidade
- 9. Sim
- 10. Sim
- 11. Não, estou insatisfeito e acho que poderia ser melhorado
- 12. Não
- 13. Sim
- 14. Não, estou insatisfeito e acho que poderia ser melhorado
- 15. Não
- 16. Sim
- 17. Não, estou insatisfeito e acho que poderia ser melhorado
- 18. Acredito que o problema é a forma como se disseminam as informações, então algo integrado seria muito mais fácil, rápido e eficaz

A.2.6 Respondente 6

1. Sim
2. Sim
3. 16 a 18 anos (Ativo)
4. Ensino médio incompleto
5. Estudante
6. 2 a 4 anos
7. Hospitaleiro, Membro de comissão
8. Sim, conheço bem e consigo encontrar as informações com facilidade
9. Não
10. Não encontrei
11. Não, estou insatisfeito e acho que poderia ser melhorado
12. Não
13. Não encontrei
14. Não, estou insatisfeito e acho que poderia ser melhorado
15. Sim
16. Sim
17. Sim, estou satisfeito e não acho que possa ser feito algo para melhorá-lo
18. Deveria haver mais transparência. A comunicação devia ser mais fluída e bem pensada. Tudo é feito meio “nas coxas”. É necessário mais planejamento e levantar o “espírito DeMolay”, que pode contribuir para maior engajamento de todos. Abs

A.2.7 Respondente 7

1. Sim
2. Sim
3. 26 anos ou mais (Sênior)
4. Mestrado

5. Empresário
6. 10 anos ou mais
7. Presidente (Mestre Conselheiro), Vice-presidente (1º Conselheiro), Tesoureiro, Auditor, Hospitaleiro, Presidente da comissão de Eventos, Presidente da comissão de Filantropias, Presidente da comissão de Assuntos Externos, Presidente da comissão de Tecnologia, Membro de comissão
8. Sim, conheço bem e consigo encontrar as informações com facilidade
9. Não
10. Não
11. Não, estou insatisfeito e acho que poderia ser melhorado
12. Não
13. Não
14. Não, estou insatisfeito e acho que poderia ser melhorado
15. Não
16. Não
17. Não, estou insatisfeito e acho que poderia ser melhorado
18. Criação de um Hub que centralizasse e padronizasse os processos. Para evitar que a cada gestão sejam criadas novas ferramentas e “regras de negócio”.

A.2.8 Respondente 8

1. Sim
2. Sim
3. 22 a 25 anos (Sênior)
4. Ensino superior incompleto
5. Analista de Inovação
6. 5 a 9 anos
7. Presidente (Mestre Conselheiro), Vice-presidente (1º Conselheiro), Vice do vice-presidente (2º Conselheiro), Auditor, Presidente da comissão de Eventos, Presidente da comissão de Sindicância e Ritualística, Membro de comissão

8. Sim, conheço bem e consigo encontrar as informações com facilidade
9. Não
10. Não
11. No caso, hoje inexistente calendário das atividades do Capítulo
12. Não
13. Não
14. Não, estou insatisfeito e acho que poderia ser melhorado
15. Não
16. Não
17. Não, estou insatisfeito e acho que poderia ser melhorado
18. n/a

APÊNDICE B – FIGURAS

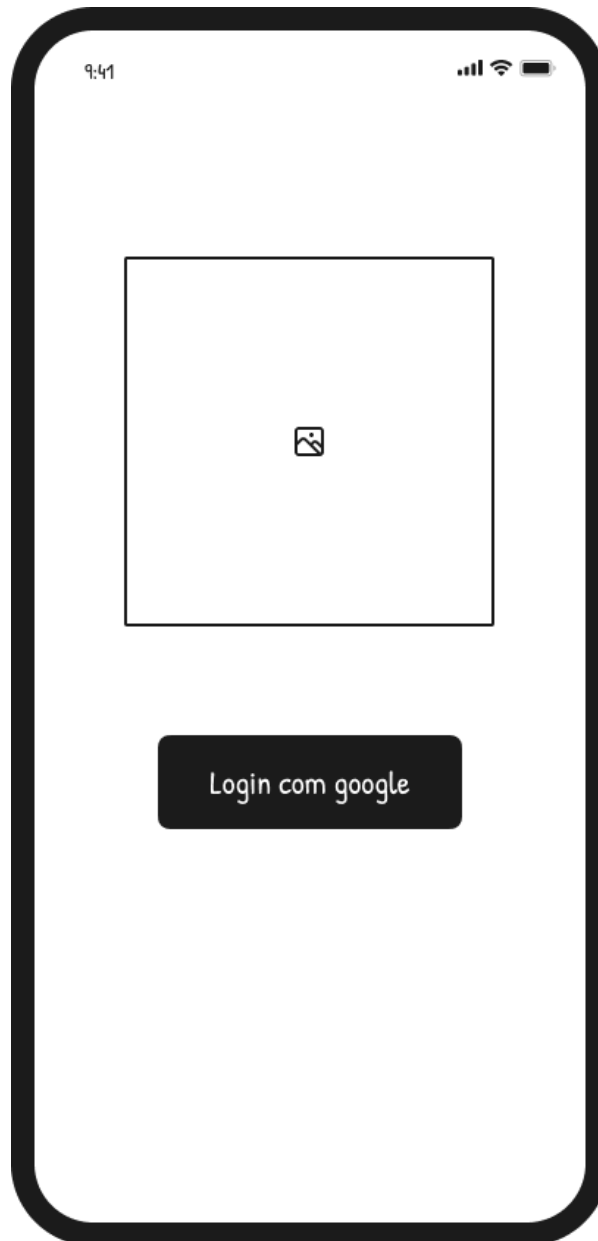


Figura 9 – Tela de login da versão mobile

Fonte: Elaborado pelos autores, 2023.

9:41

Cadastro Do Usuário



ID DeMolay *

Digite seu ID DeMolay

e-Mail alternativo

Digite seu e-Mail alternativo

CPF *

Digite seu CPF

Telefone *

(xx) x xxxx-xxxx

Data de iniciação *

dd/mm/aaaa

Data de elevação

dd/mm/aaaa

Avançar →

Figura 10 – Tela de cadastro do usuário da versão mobile

Fonte: Elaborado pelos autores, 2023.

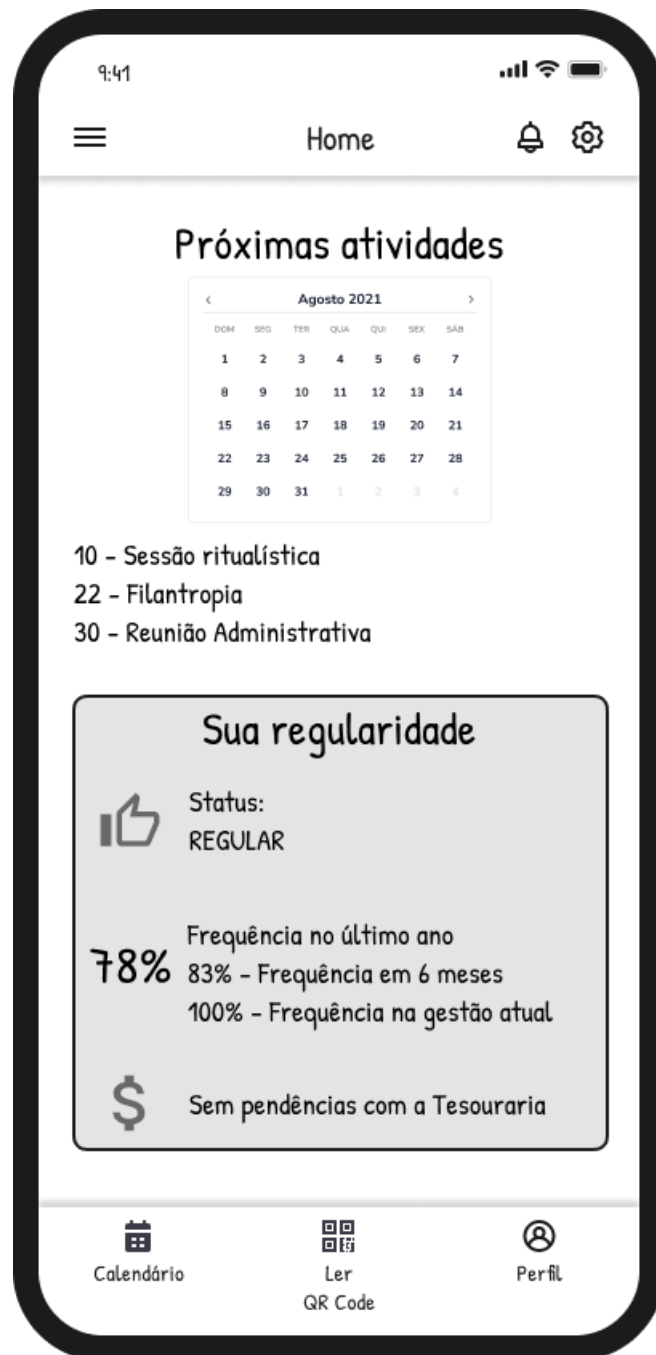


Figura 11 – Tela central da versão mobile

Fonte: Elaborado pelos autores, 2023.

9:41

📶

☰ Cadastro de Membros 🔔 ⚙️

Nome *

⬅ Digite seu nome completo ➡

Data de nascimento *

⬅ Digite sua data de nascimento ➡

ID DeMolay *

⬅ Digite seu ID DeMolay ➡

e-mail do capítulo *

⬅ Digite seu e-mail do capítulo ➡

e-Mail alternativo

⬅ Digite seu e-Mail alternativo ➡

CPF *

⬅ Digite seu CPF ➡

Telefone *

⬅ (xx) x xxxx-xxxx ➡

Data de iniciação *

⬅ dd/mm/aaaa ➡

Data de elevação

⬅ dd/mm/aaaa ➡

Selecione os perfis de acesso

⬅ Select ▼ ➡

Salvar

📅 Calendário

📄 Ler QR Code

👤 Perfil

Figura 12 – Tela de cadastro de membros da versão mobile



Figura 13 – Tela de listagem dos membros da versão mobile

Fonte: Elaborado pelos autores, 2023.

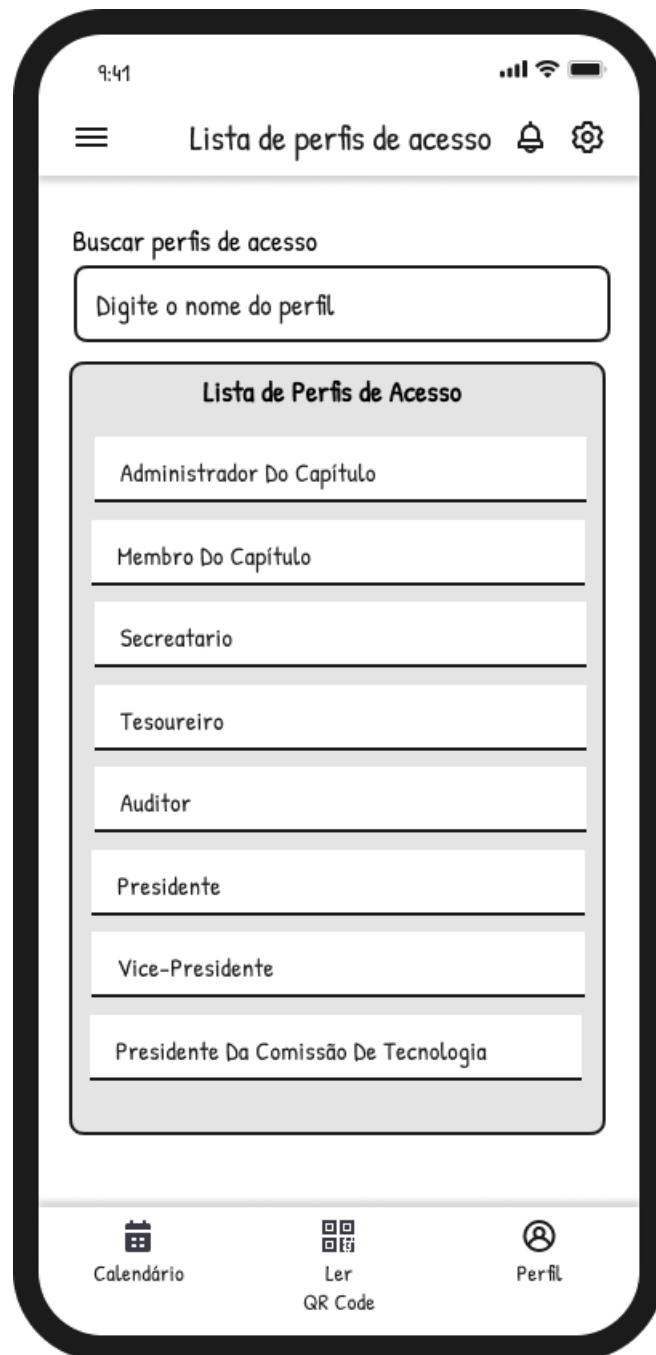


Figura 14 – Tela de listagem dos perfis da versão mobile

Fonte: Elaborado pelos autores, 2023.

9:41

≡ Criar Perfil 🔔 ⚙️

Nome do perfil

Árvore de recursos

- ☐ Calendário de Eventos
 - ☐ Cadastrar
 - ☐ Editar
 - ☐ Excluir
- ☐ Controle de Frequência
 - ☐ Cadastrar
 - ☐ Editar
 - ☐ Excluir
- ☐ Controle Financeiro
 - ☐ Cadastrar
 - ☐ Editar
 - ☐ Excluir
- ☐ Portal de Transparência
- ☐ Membros
 - ☐ Cadastrar
 - ☐ Editar
 - ☐ Excluir
- ☐ Perfis
 - ☐ Cadastrar
 - ☐ Editar
 - ☐ Excluir

Salvar

Calendário Ler QR Code Perfil

Figura 15 – Tela de cadastro de perfis da versão mobile

Fonte: Elaborado pelos autores, 2023.

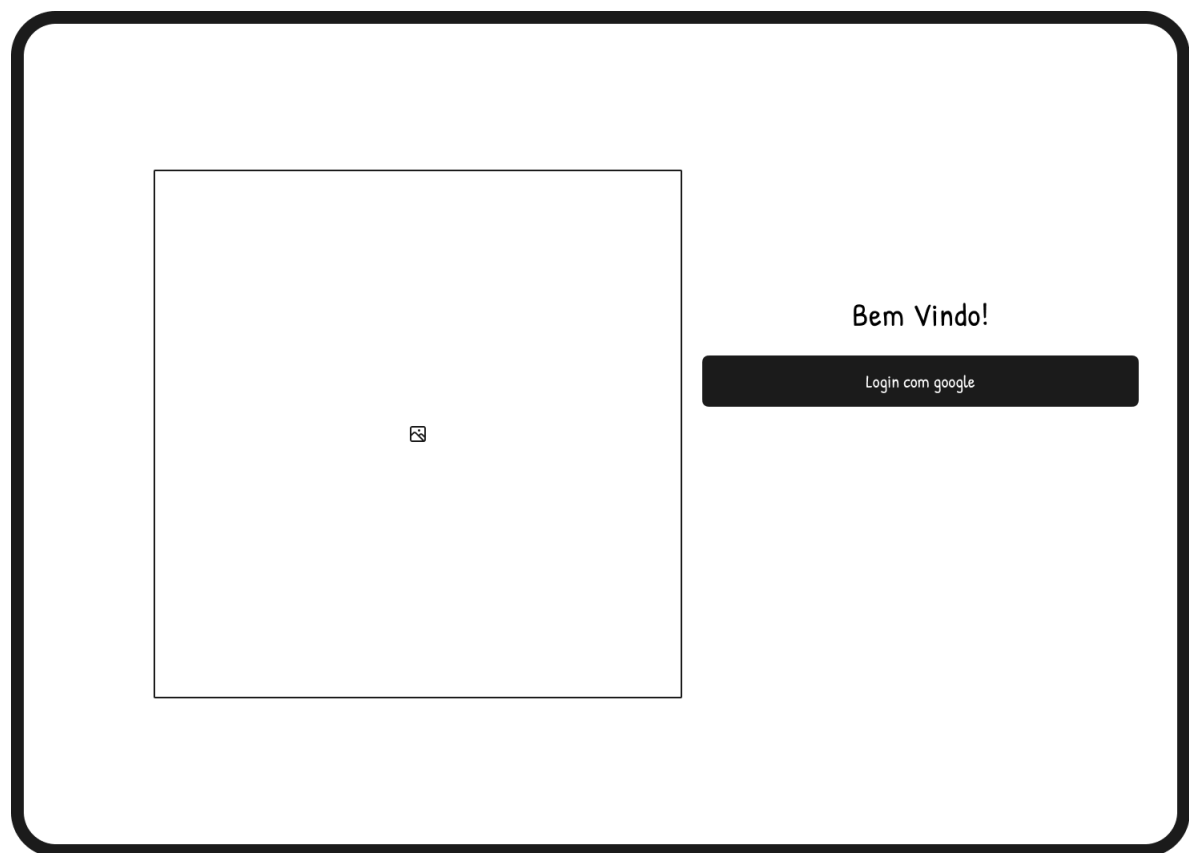


Figura 16 – Tela de login da versão web

Fonte: Elaborado pelos autores, 2023.

Cadastro Do Usuário



ID DeMolay *	CPF *
Digite seu ID DeMolay	Digite seu CPF
e-Mail alternativo	Telefone *
Digite seu e-Mail alternativo	(xx) x xxxx-xxxx
Data de Iniciação *	Data de elevação
dd/mm/aaaa	dd/mm/aaaa

Salvar

Figura 17 – Tela de cadastro do usuário da versão web

Fonte: Elaborado pelos autores, 2023.

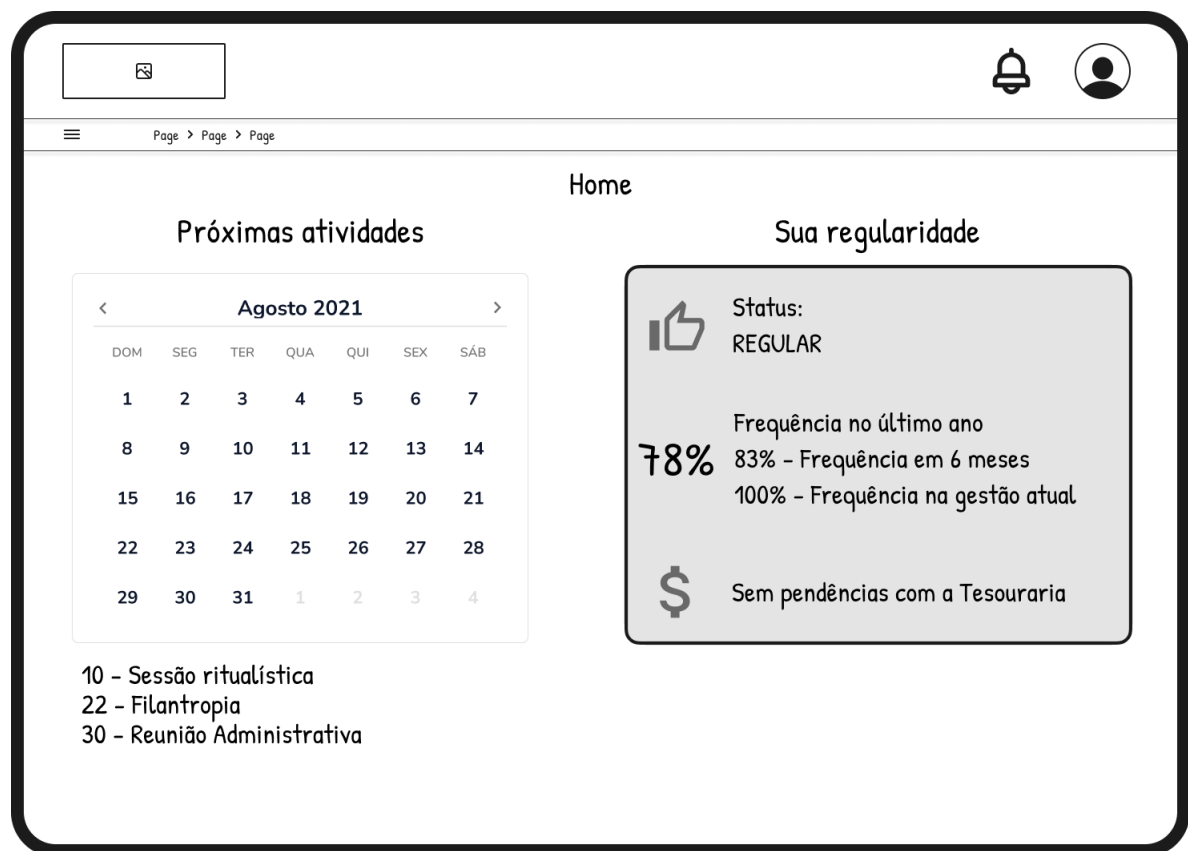


Figura 18 – Tela central da versão web

Fonte: Elaborado pelos autores, 2023.

The image shows a web browser window with a registration form titled "Cadastro de membros". The browser's address bar shows "Page > Page > Page". The form is organized into two columns. The left column contains fields for "Nome *", "Data de nascimento *", "Telefone *", "Data de iniciação *", and "Data de elevação". The right column contains fields for "ID DeMolay *", "e-mail do capítulo *", "e-Mail alternativo", "CPF *", and "Selecione os perfis de acesso". Each field has a placeholder text indicating what to enter. At the bottom center of the form is a "Salvar" button.

Cadastro de membros	
Nome *	ID DeMolay *
Digite seu nome completo	Digite seu ID DeMolay
Data de nascimento *	e-mail do capítulo *
Digite sua data de nascimento	Digite seu e-mail do capítulo
Telefone *	e-Mail alternativo
(xx) x xxxx-xxxx	Digite seu e-Mail alternativo
Data de iniciação *	CPF *
dd/mm/aaaa	Digite seu CPF
Data de elevação	Selecione os perfis de acesso
dd/mm/aaaa	Select
Salvar	

Figura 19 – Tela de cadastro de membros da versão web

Fonte: Elaborado pelos autores, 2023.

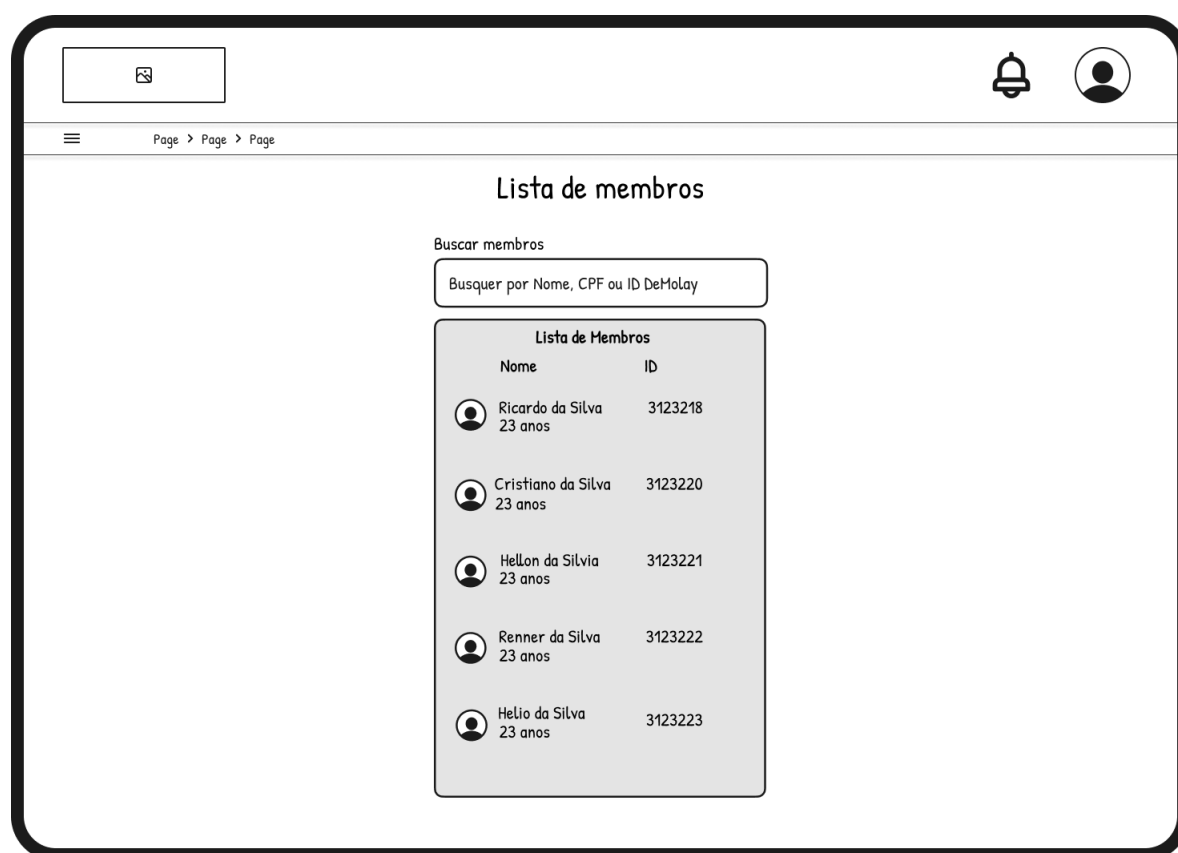


Figura 20 – Tela de listagem dos membros da versão web

Fonte: Elaborado pelos autores, 2023.



Figura 21 – Tela de listagem dos perfis da versão web

Fonte: Elaborado pelos autores, 2023.

Nome do perfil

Digite o nome do perfil

Árvore de recursos

- ☐ Calendário de Eventos
 - ☐ Cadastrar
 - ☐ Editar
 - ☐ Excluir
- ☐ Controle de Frequência
 - ☐ Cadastrar
 - ☐ Editar
 - ☐ Excluir
- ☐ Controle Financeiro
 - ☐ Cadastrar
 - ☐ Editar
 - ☐ Excluir
- ☐ Portal de Transparência
- ☐ Membros
 - ☐ Cadastrar

Figura 22 – Tela de cadastro de perfis da versão web

Fonte: Elaborado pelos autores, 2023.

Diagrama de Componentes — DeMoDash

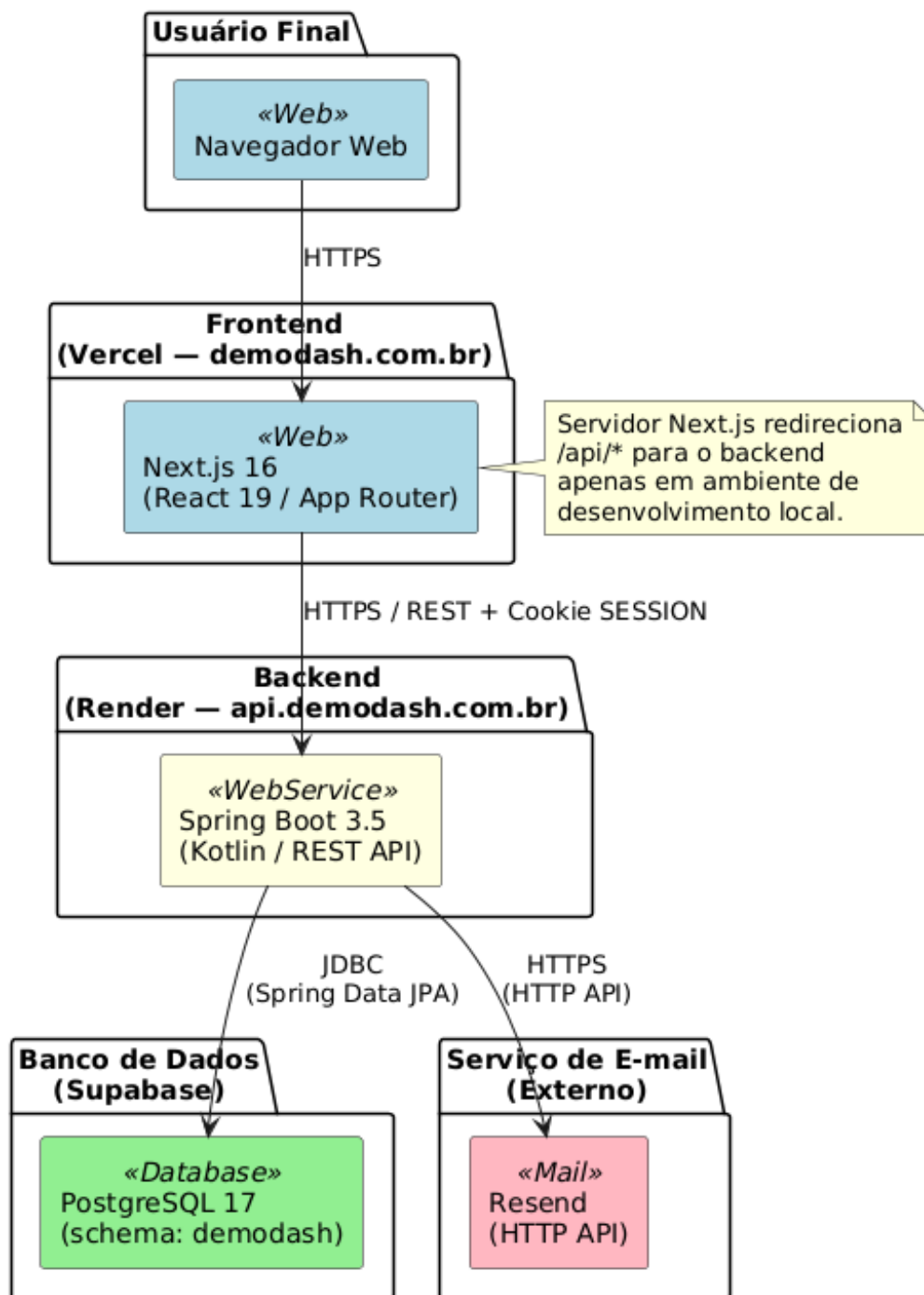


Figura 23 – Diagrama de Componentes do DeMoDash. **Legenda:** retângulos representam componentes; estereótipos entre « » indicam a natureza de cada componente (Web, WebService, Database, Mail); setas com rótulo indicam o protocolo de comunicação; a direção da seta indica a direção da dependência (o componente de origem depende do componente de destino).

Fonte: Elaborado pelos autores, 2026.

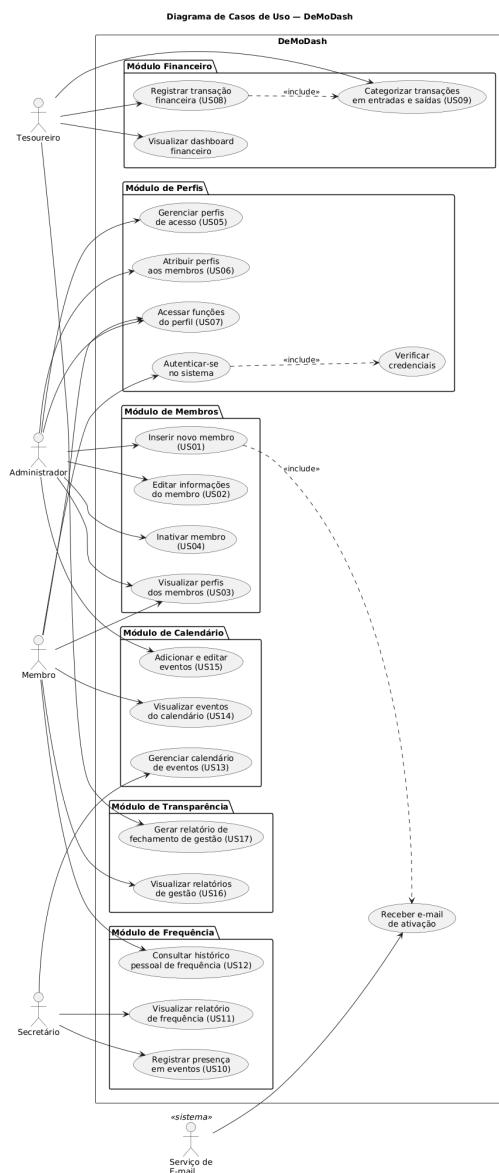


Figura 24 – Diagrama de Casos de Uso do DeMoDash (US01–US17). **Legenda:** bonecos palito representam atores (primários à esquerda, secundário à direita); elipses representam casos de uso; o retângulo delimita o sistema; linhas sólidas entre ator e caso de uso representam associação (participação direta); linhas tracejadas com o rótulo «include» indicam que o caso de uso base somente se completa quando o caso incluído também se completa.

Fonte: Elaborado pelos autores, 2026.

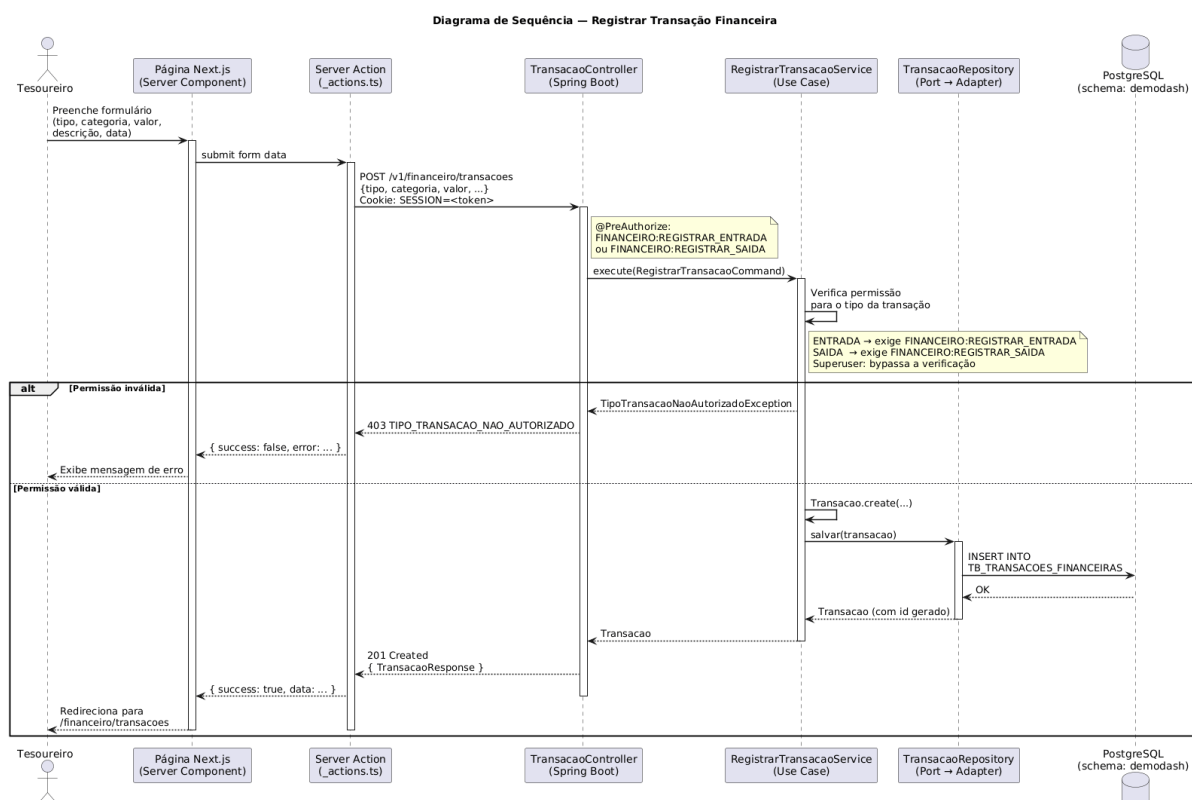


Figura 25 – Diagrama de Sequência — Registrar Transação Financeira. **Legenda:** caixas no topo são os participantes (*lifelines*); linhas verticais tracejadas são as linhas de vida de cada participante; retângulos sobre as linhas de vida indicam o período de ativação do participante; setas sólidas com ponta cheia representam mensagens síncronas (chamadas de função ou requisições HTTP); setas tracejadas representam retornos; o quadro *alt* separa o fluxo alternativo de erro do fluxo principal de sucesso.

Fonte: Elaborado pelos autores, 2026.

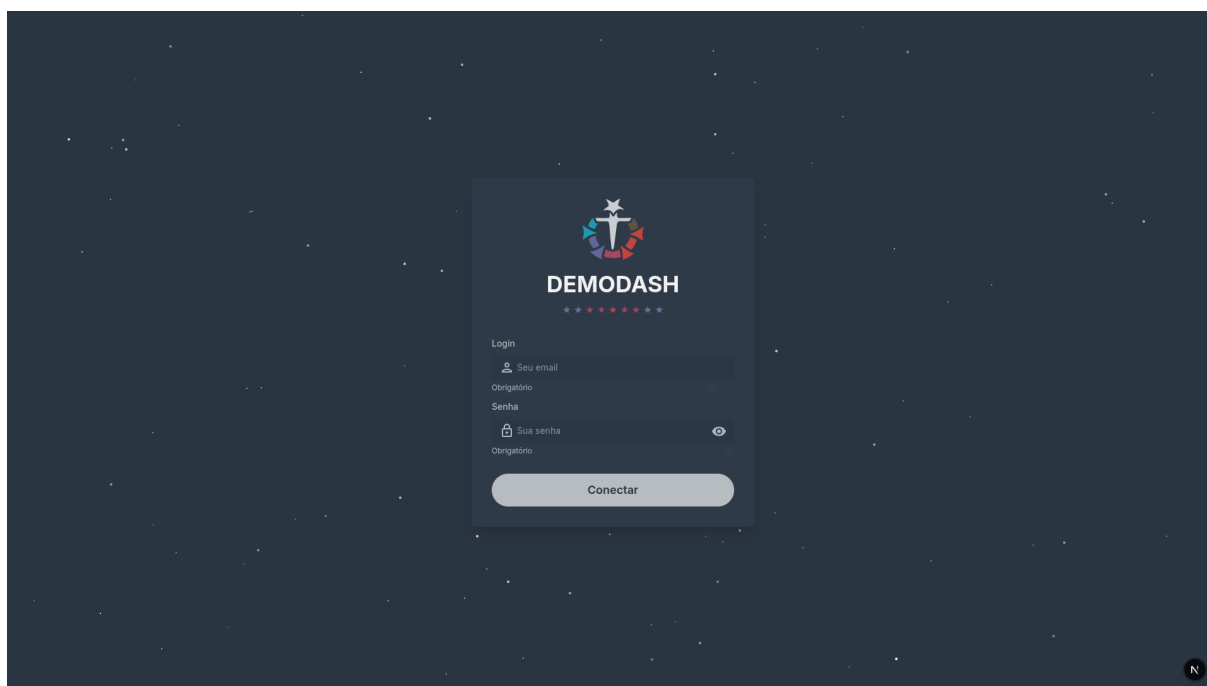


Figura 26 – Tela de login.

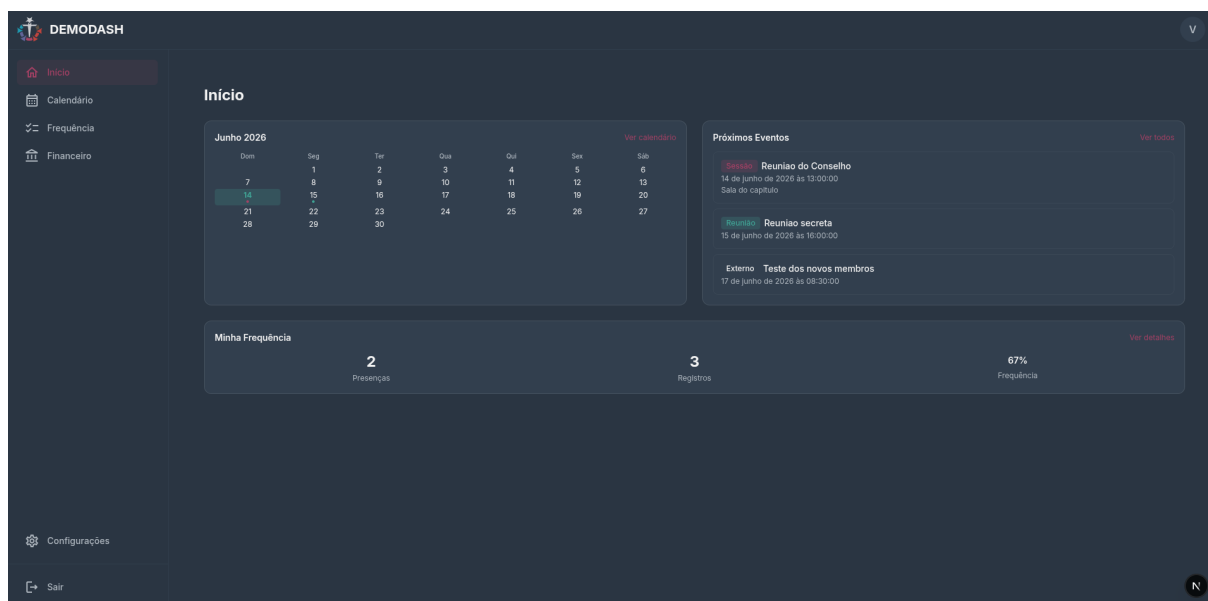


Figura 27 – Tela inicial (home).



Figura 28 – Tela inicial (home) mobile.

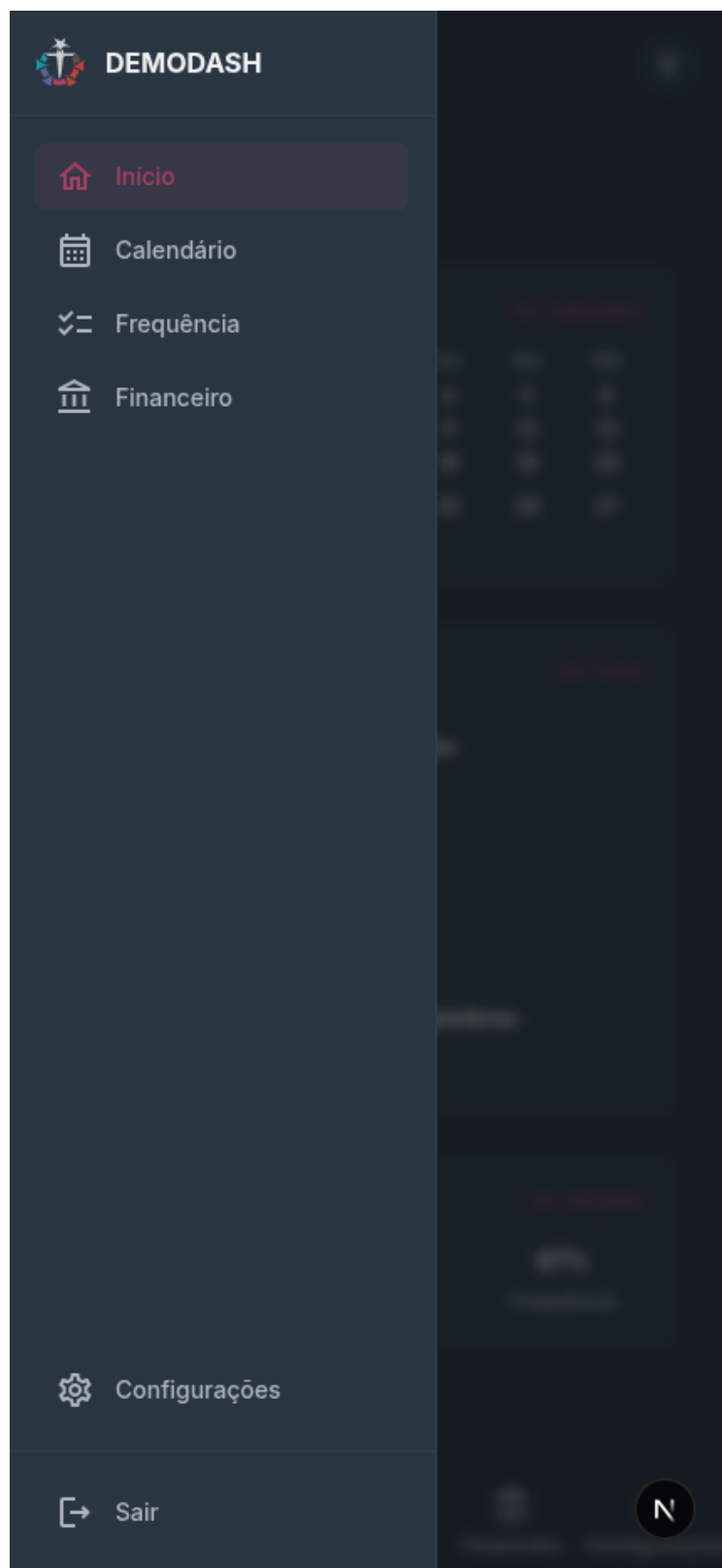


Figura 29 – Tela de menu lateral mobile.

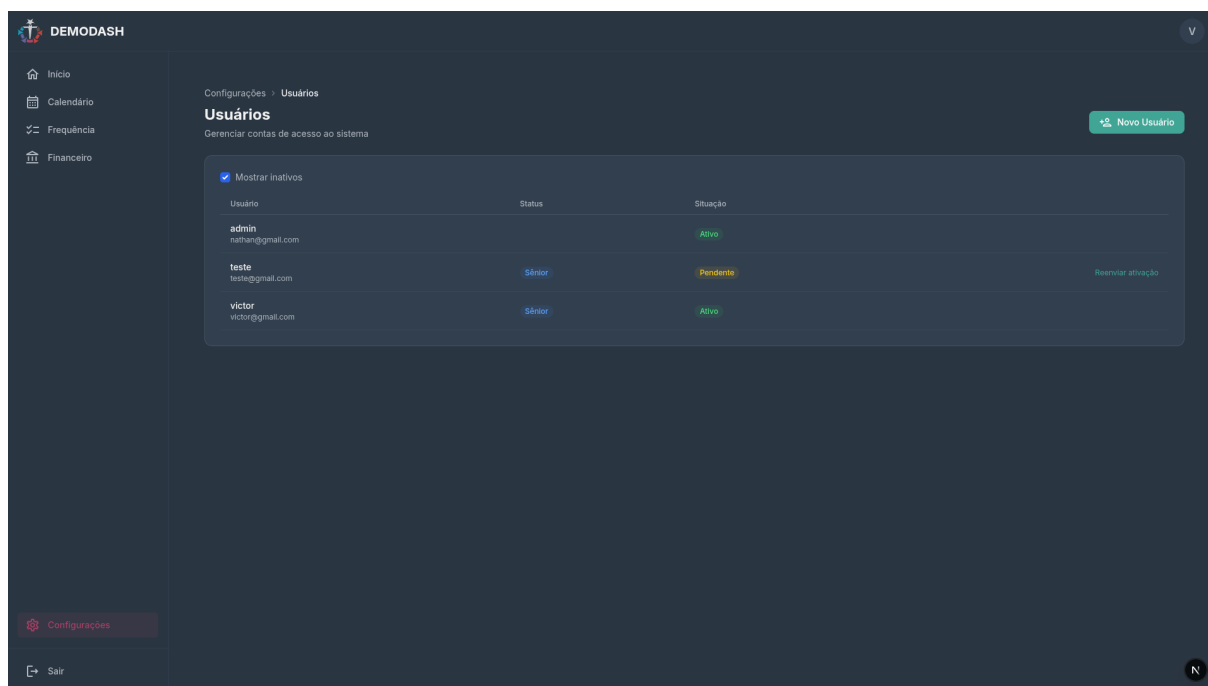


Figura 30 – Tela de lista de usuários.

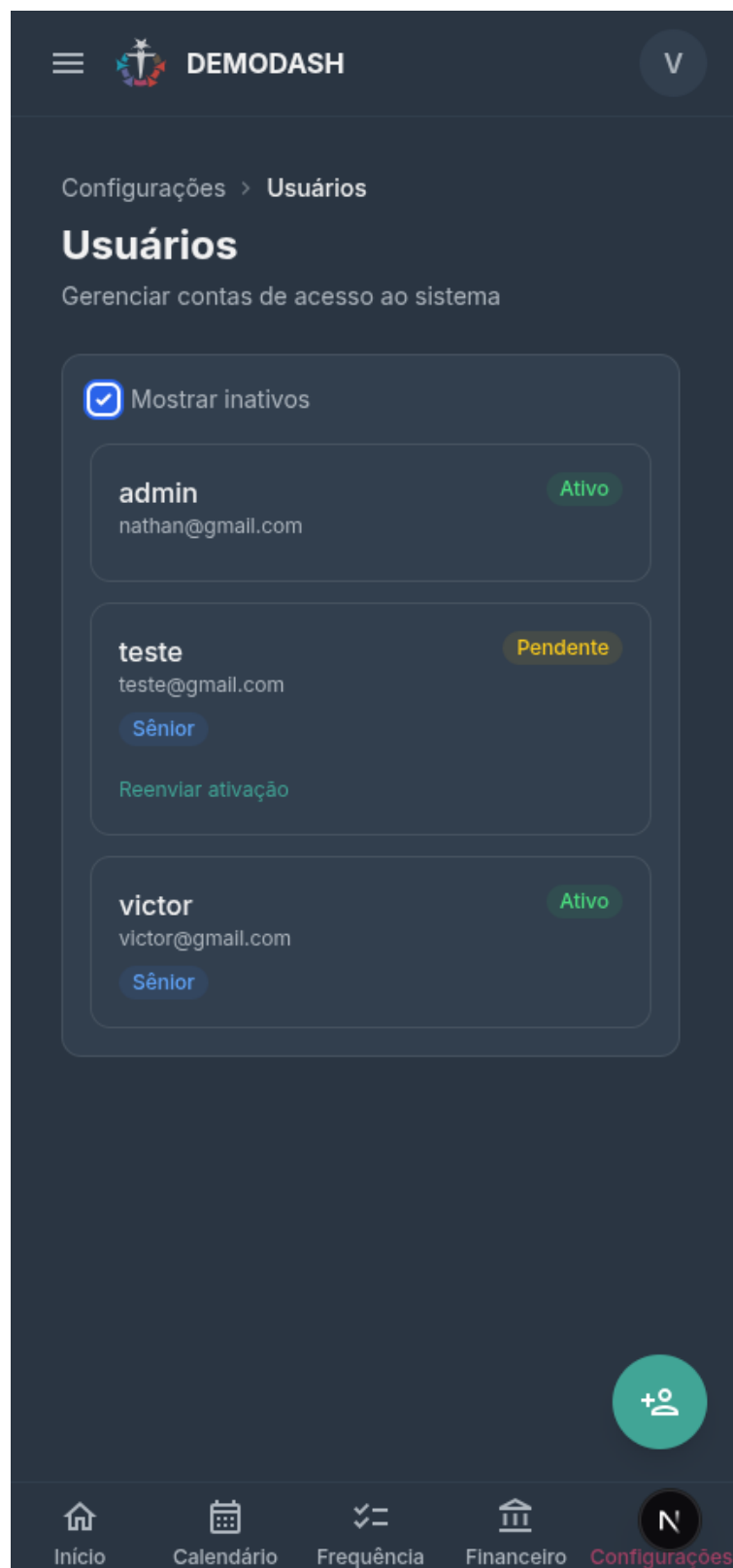


Figura 31 – Tela de lista de usuários mobile.

DEMODASH

Configurações > Usuários > Novo usuário

[Voltar](#)

Novo Usuário

O usuário receberá um email para definir sua senha.

Username

ex.: joao.silva

Obrigatório

Email

email@exemplo.com

Obrigatório

Nome completo

Nome como aparece nos registros

Obrigatório

ID DeMolay

Ex.: DM0001

Obrigatório

Data de nascimento *

dd/mm/aaaa

Obrigatório

Data de iniciação *

dd/mm/aaaa

Obrigatório

Data de elevação (opcional)

dd/mm/aaaa

Perfil *

☒ Membro ☐ Teste

[Cancelar](#) [Criar Usuário](#)

Configurações

Sair

Figura 32 – Tela de novo usuário.

The image shows a mobile application interface for editing a user profile. At the top, there is a dark blue header with a hamburger menu icon, a logo with a star and anchor, the text "DEMODASH", and a circular profile icon with the letter "V". Below the header, a breadcrumb trail reads "Configurações > Usuários > Editar teste". A back arrow icon is followed by the text "Voltar". The main title "Editar teste" is displayed in a large, bold font. The form contains several input fields: "Username" with the value "teste", "Email" with "teste@gmail.com", "Nome completo" with "Teste da Silva", "ID DeMolay" with "123", "Data de nascimento" with "01/01/2000", and "Data de iniciação" with "10/01/2014". Each field has a label above it and a "Obrigatório" (Required) note below it. The date fields include a calendar icon. At the bottom, there is a navigation bar with five icons and labels: "Início", "Calendário", "Frequência", "Financeiro", and "Configurações" (which is highlighted in red). A circular profile icon with the letter "N" is also present in the bottom right corner of the navigation bar.

Configurações > Usuários > Editar teste

← Voltar

Editar teste

Username

teste

Email

teste@gmail.com

Nome completo

Teste da Silva

Obrigatório

ID DeMolay

123

Obrigatório

Data de nascimento *

01/01/2000

Obrigatório

Data de iniciação *

10/01/2014

Início Calendário Frequência Financeiro Configurações

Figura 33 – Tela de edição de usuário mobile.

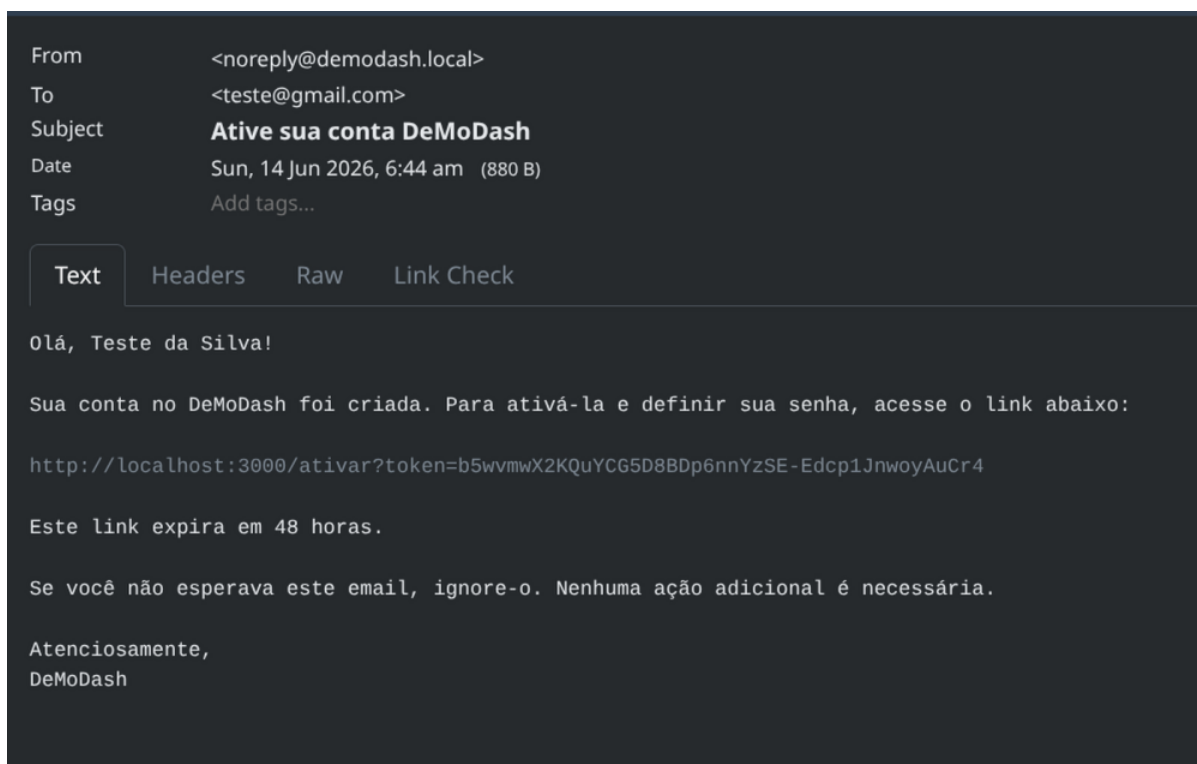


Figura 34 – E-mail para ativar a conta

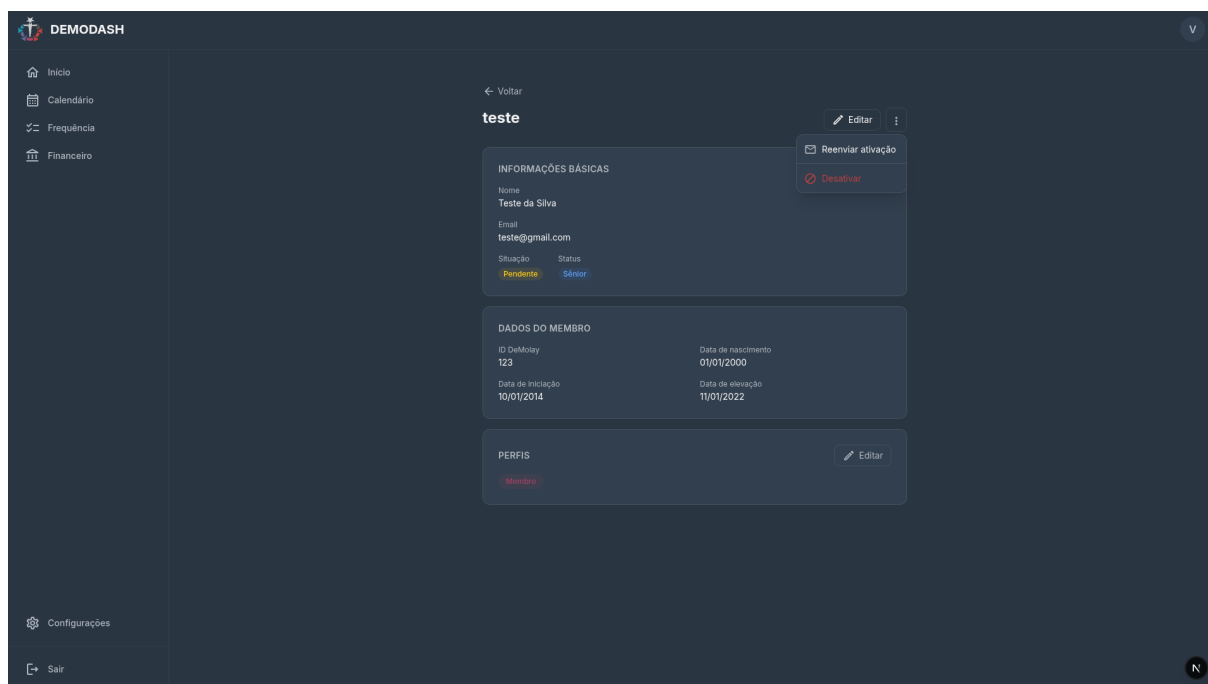


Figura 35 – Tela de visualização de usuário.

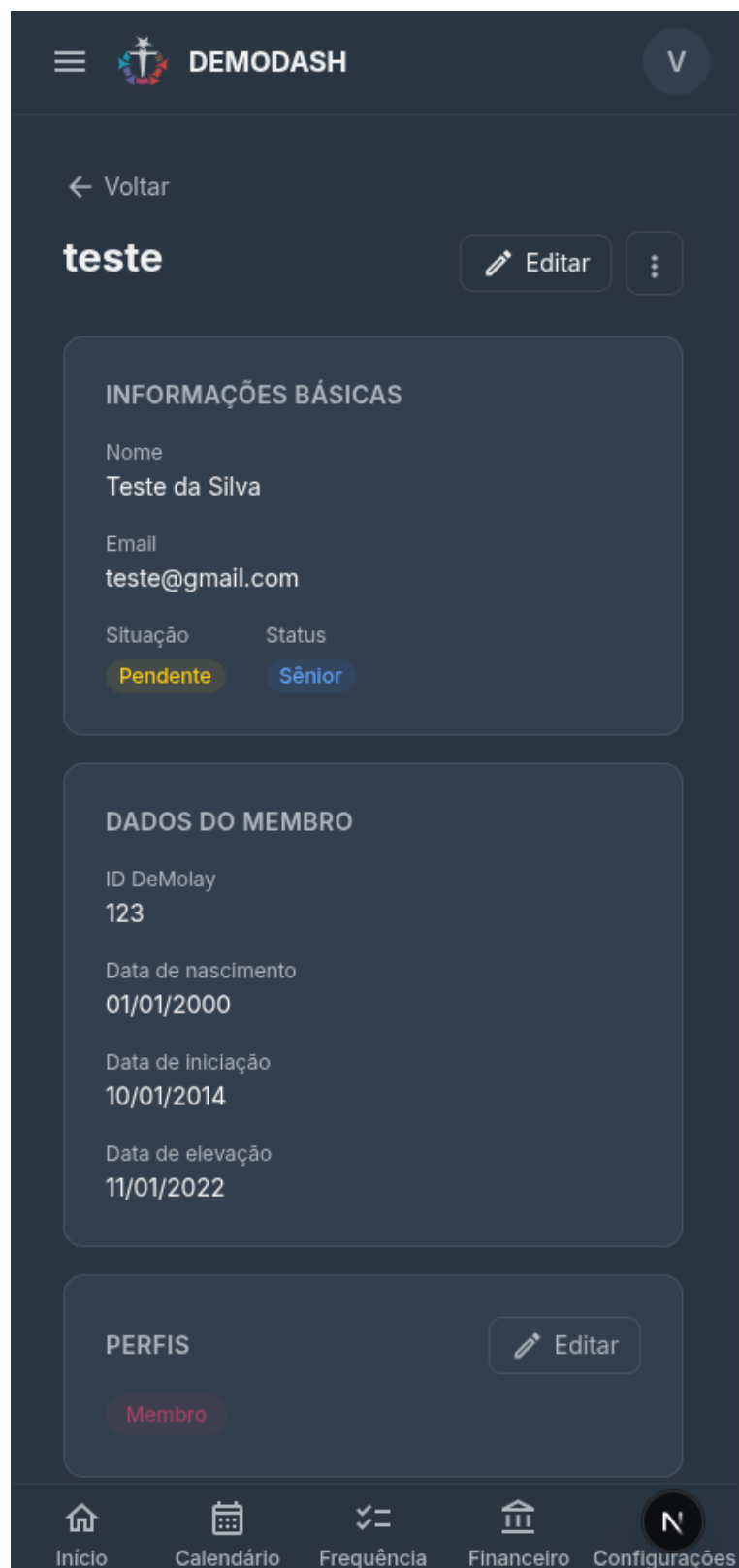


Figura 36 – Tela de visualização de usuário mobile.

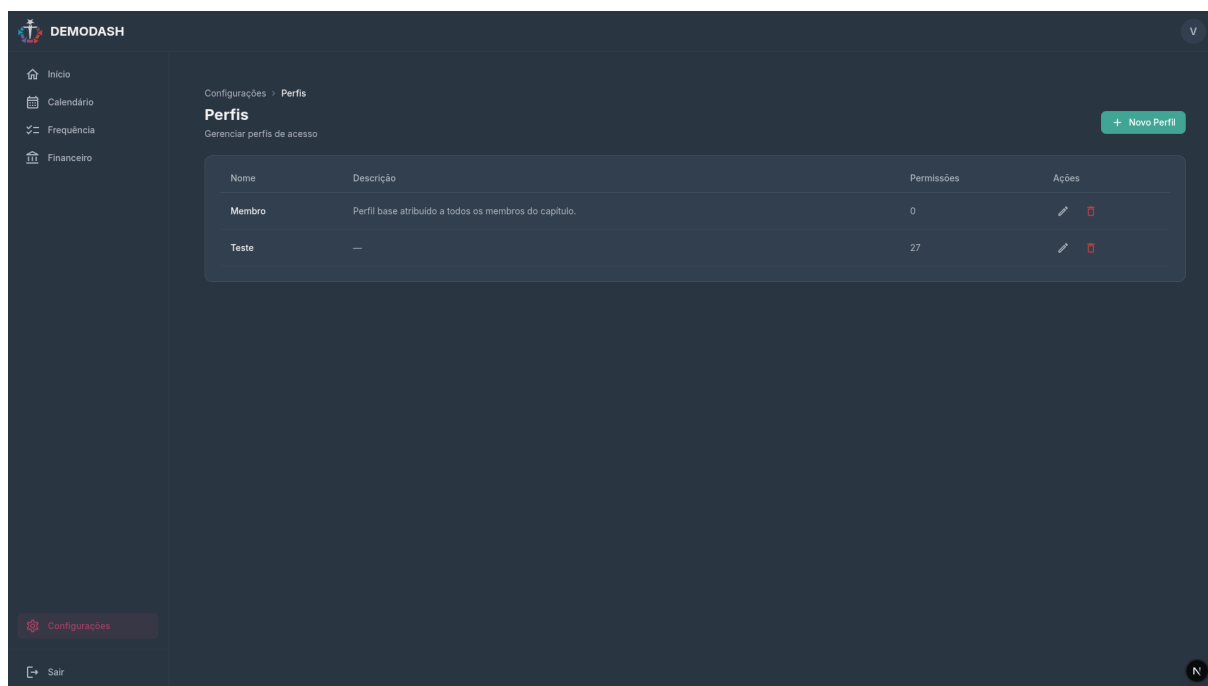


Figura 37 – Tela de lista de perfis.

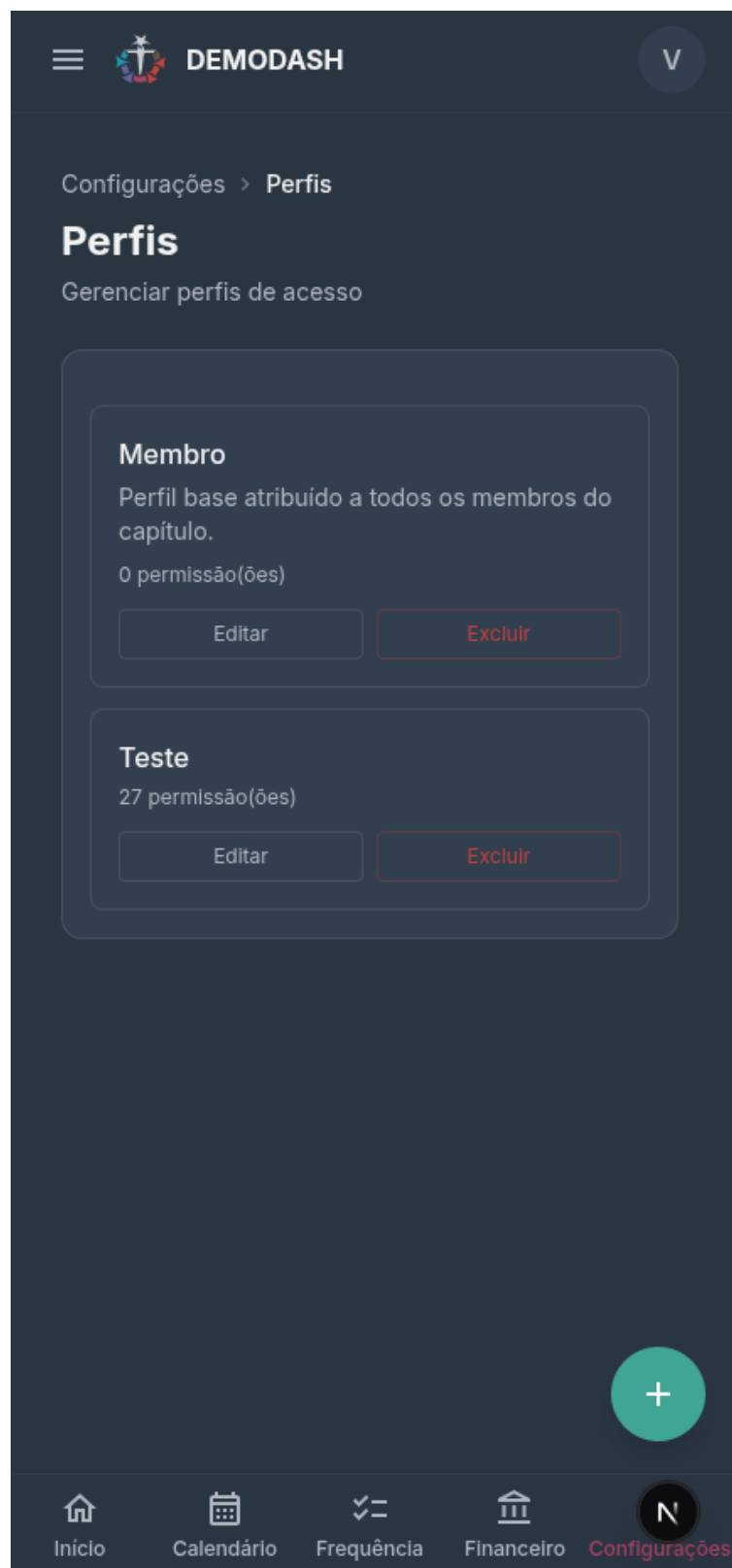


Figura 38 – Tela de lista de perfis mobile.

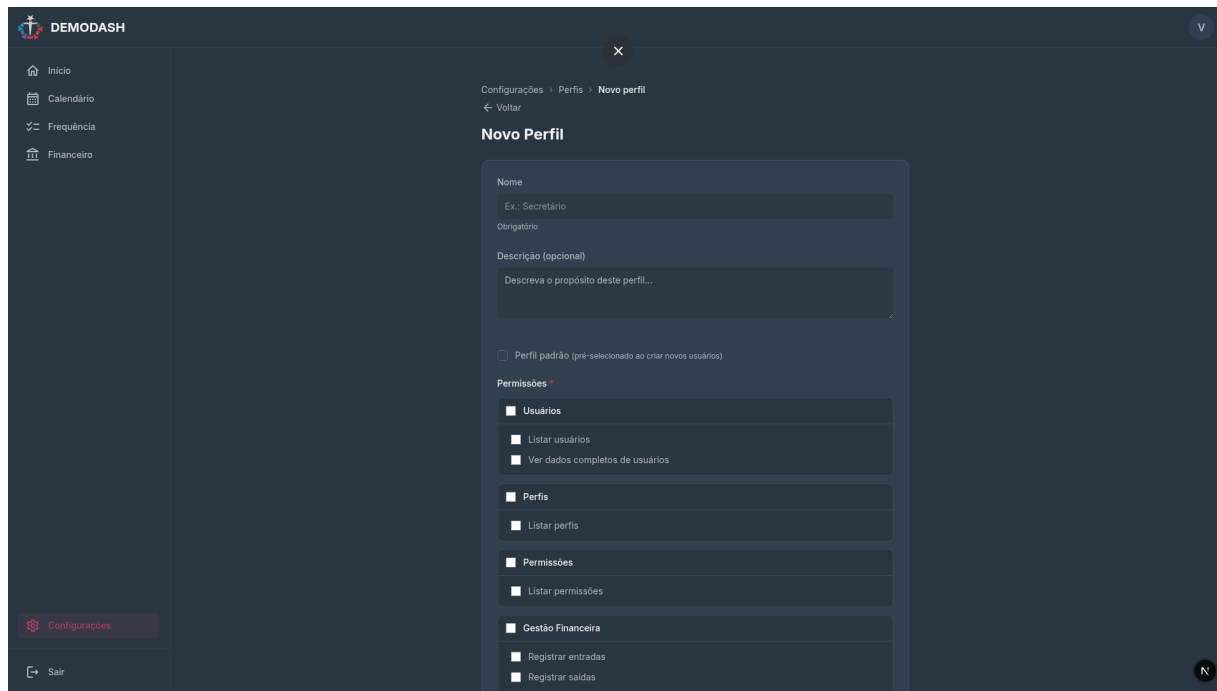


Figura 39 – Tela de novo perfil (parte 1).

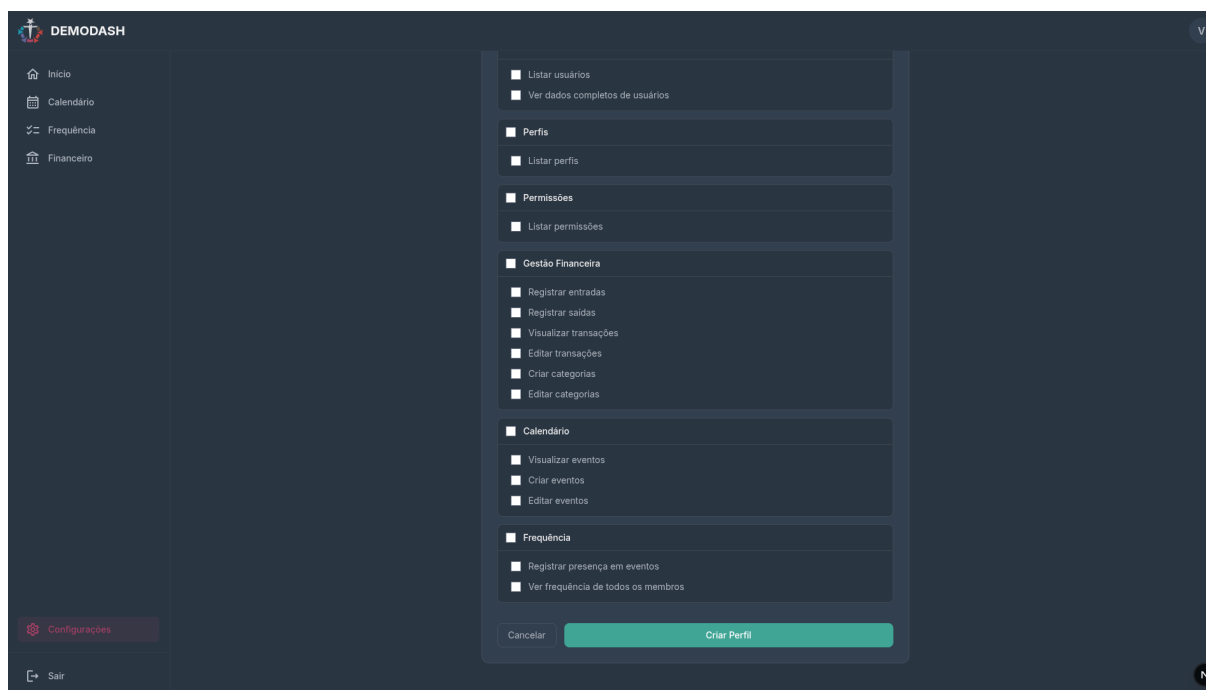


Figura 40 – Tela de novo perfil (parte 2).

DEMODASH

Início

Calendário

Frequência

Financeiro

Configurações

Sair

Editar Perfil

Nome

Teste

Obrigatório

Descrição (opcional)

Descreva o propósito deste perfil...

☐ Perfil padrão (pré-selecionado ao criar novos usuários)

Permissões *

☒ Usuários

☒ Listar usuários

☒ Ver dados completos de usuários

☒ Perfis

☒ Listar perfis

☒ Permissões

☒ Listar permissões

☒ Gestão Financeira

☒ Registrar entradas

☒ Registrar saídas

☒ Visualizar transações

☒ Editar transações

☒ Criar categorias

☒ Editar categorias

Figura 41 – Tela de edição de perfil (parte 1).

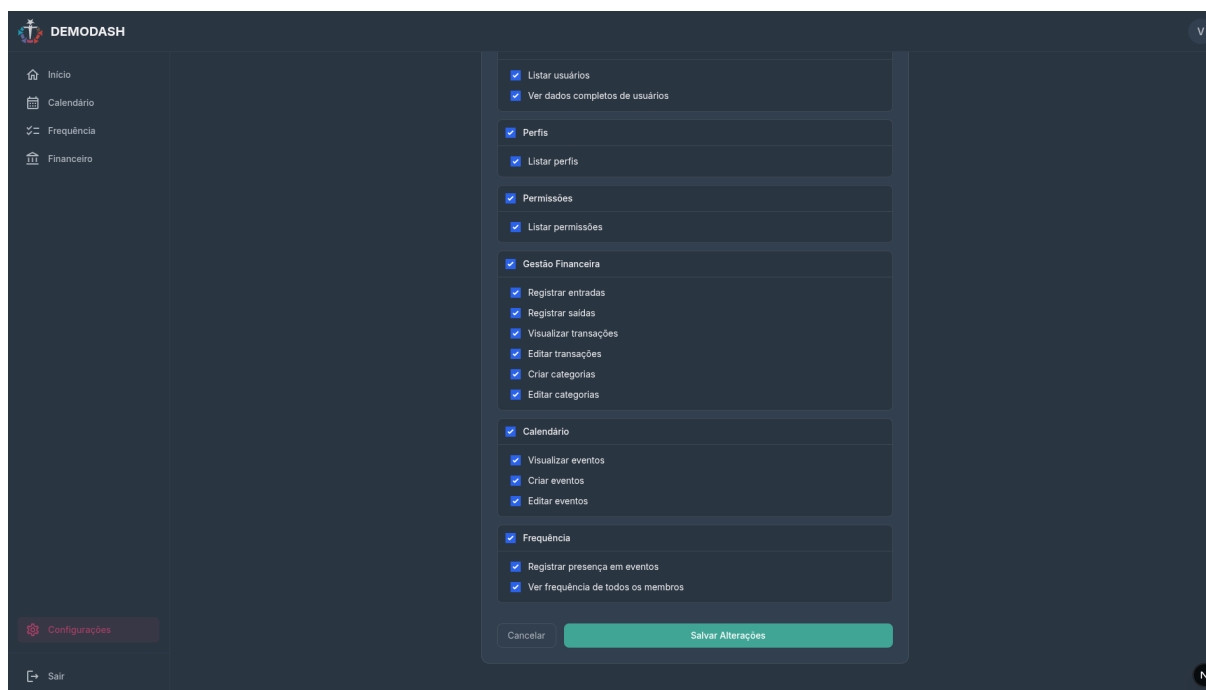


Figura 42 – Tela de edição de perfil (parte 2).



Figura 43 – Tela de edição de perfil mobile.

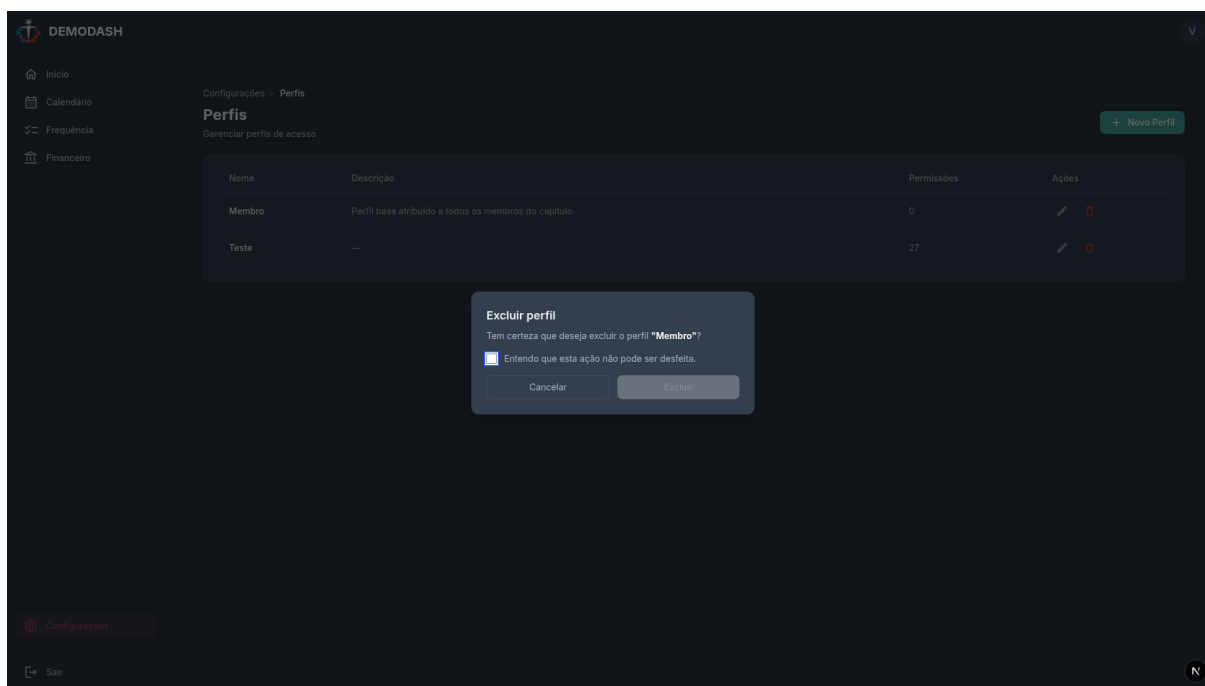


Figura 44 – Tela de exclusão de perfil.

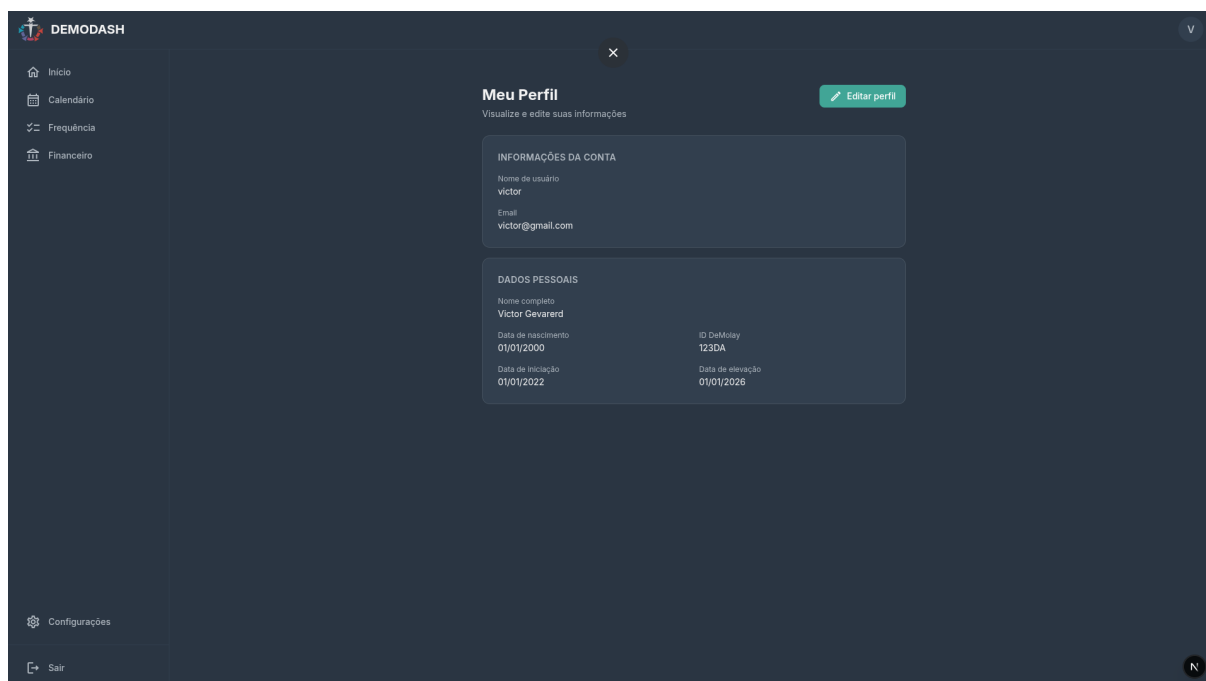


Figura 45 – Tela de meu perfil.

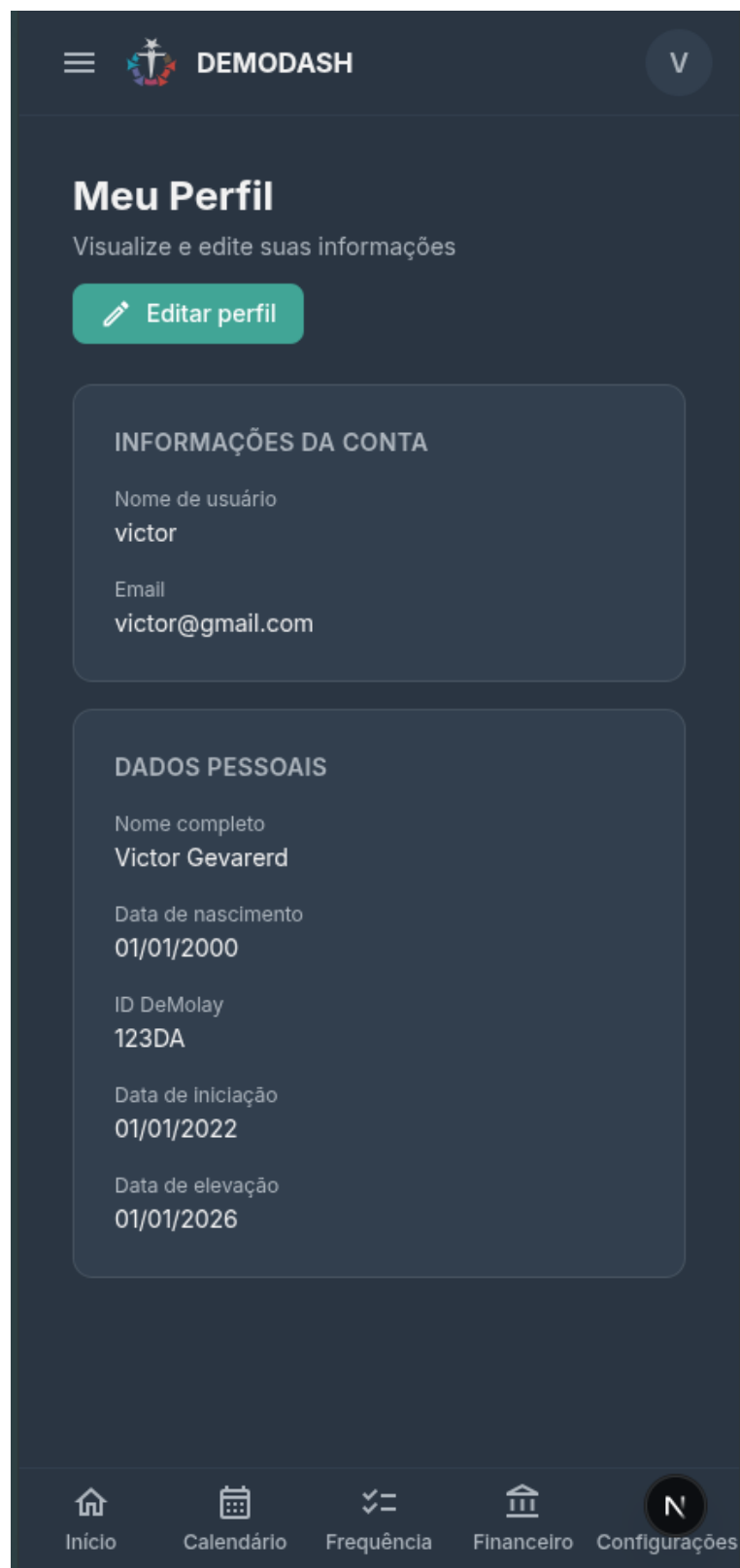


Figura 46 – Tela de meu perfil mobile.

The image shows a web application interface for editing a user profile. The application is titled 'DEMODASH' in the top left corner. A sidebar on the left contains navigation links: 'Início' (Home), 'Calendário' (Calendar), 'Frequência' (Frequency), 'Financeiro' (Financial), 'Configurações' (Settings), and 'Sair' (Logout). The main content area is titled 'Editar Perfil' and includes a 'Voltar' (Back) link. The form contains the following fields:

- Nome de usuário**: A text input field containing the value 'victor'.
- Email**: A text input field containing the value 'victor@gmail.com'.
- Nome completo**: A text input field containing the value 'Victor Gevarerd', with a red asterisk indicating it is required.
- Data de nascimento**: A date picker field showing '01/01/2000', with a red asterisk indicating it is required.

At the bottom of the form are two buttons: 'Cancelar' (Cancel) and 'Salvar alterações' (Save changes).

Figura 47 – Tela de edição do próprio perfil.

DEMODASH

V

← Voltar

Editar Perfil

Nome de usuário

victor

Email

victor@gmail.com

Nome completo

Victor Gevarerd

Obrigatório

Data de nascimento *

01/01/2000

Obrigatório

Cancelar Salvar alterações

Início Calendário Frequência Financeiro Configurações

Figura 48 – Tela de edição do próprio perfil mobile.

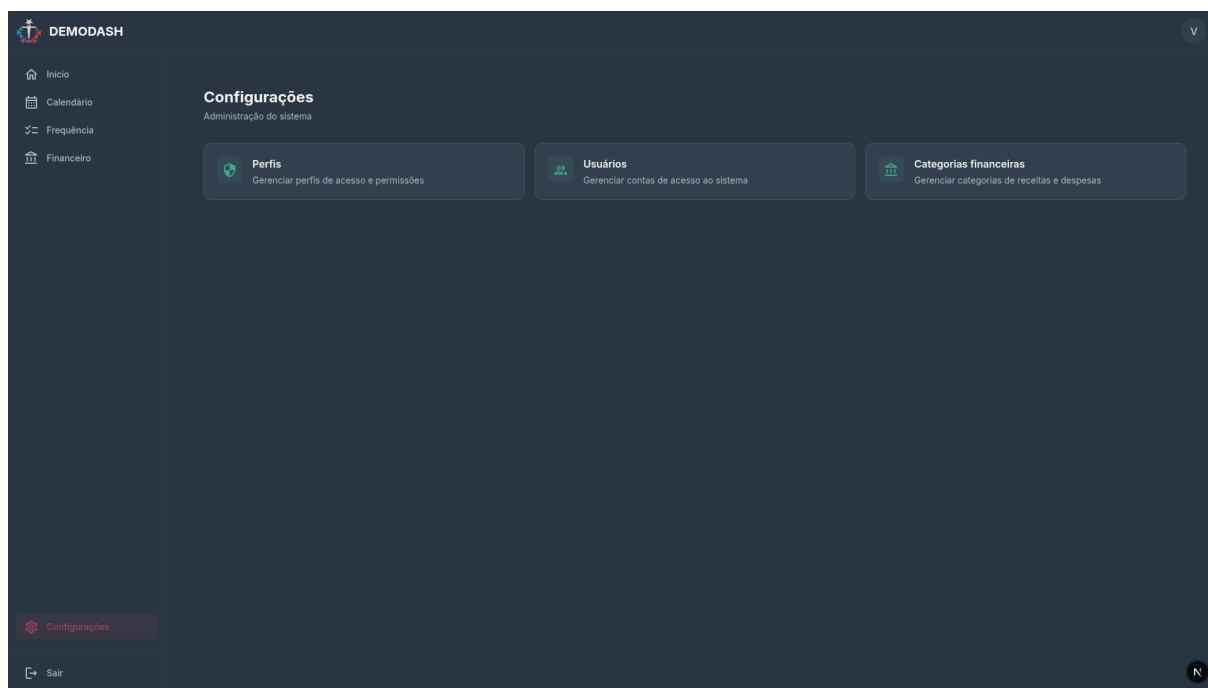


Figura 49 – Tela de configurações.

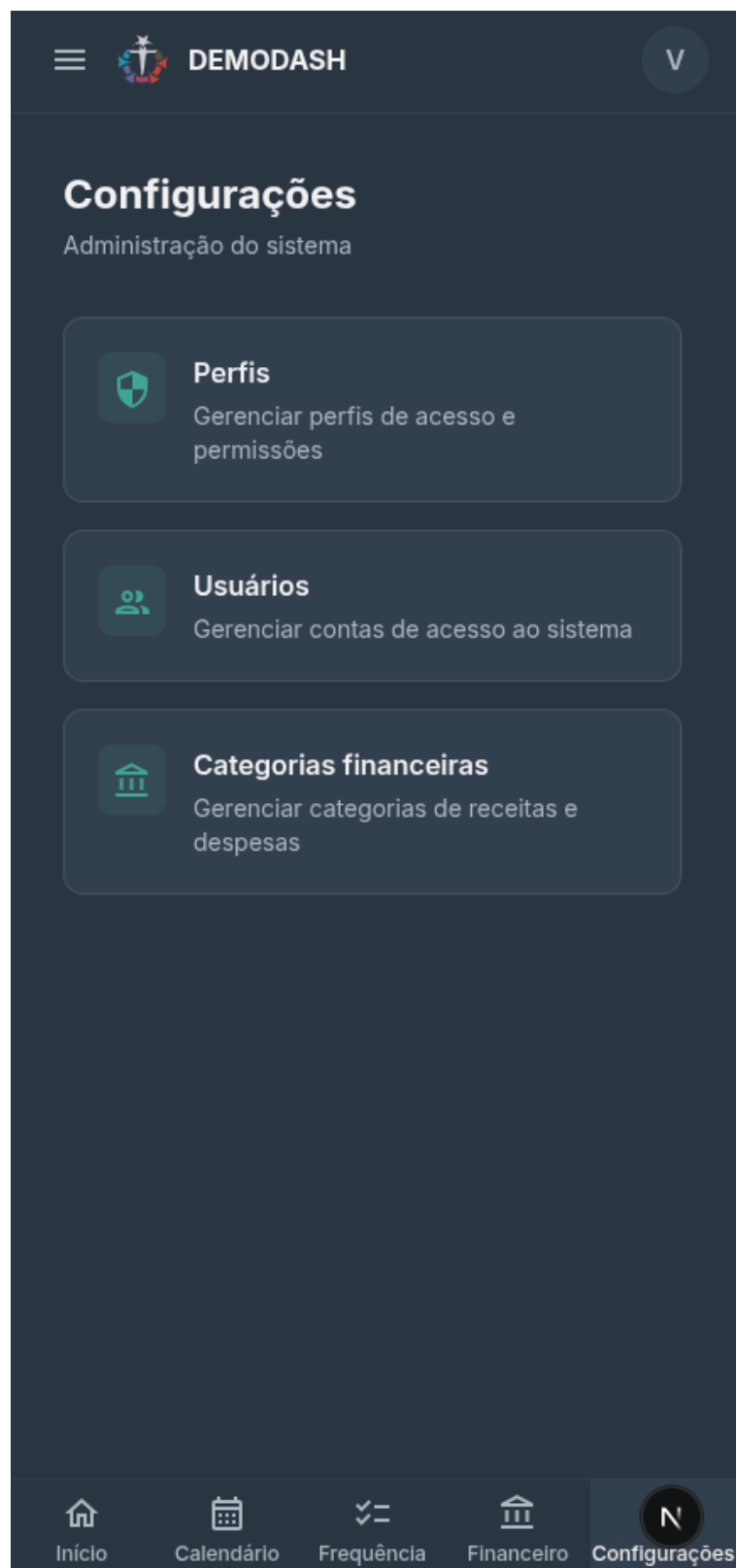


Figura 50 – Tela de configurações mobile.

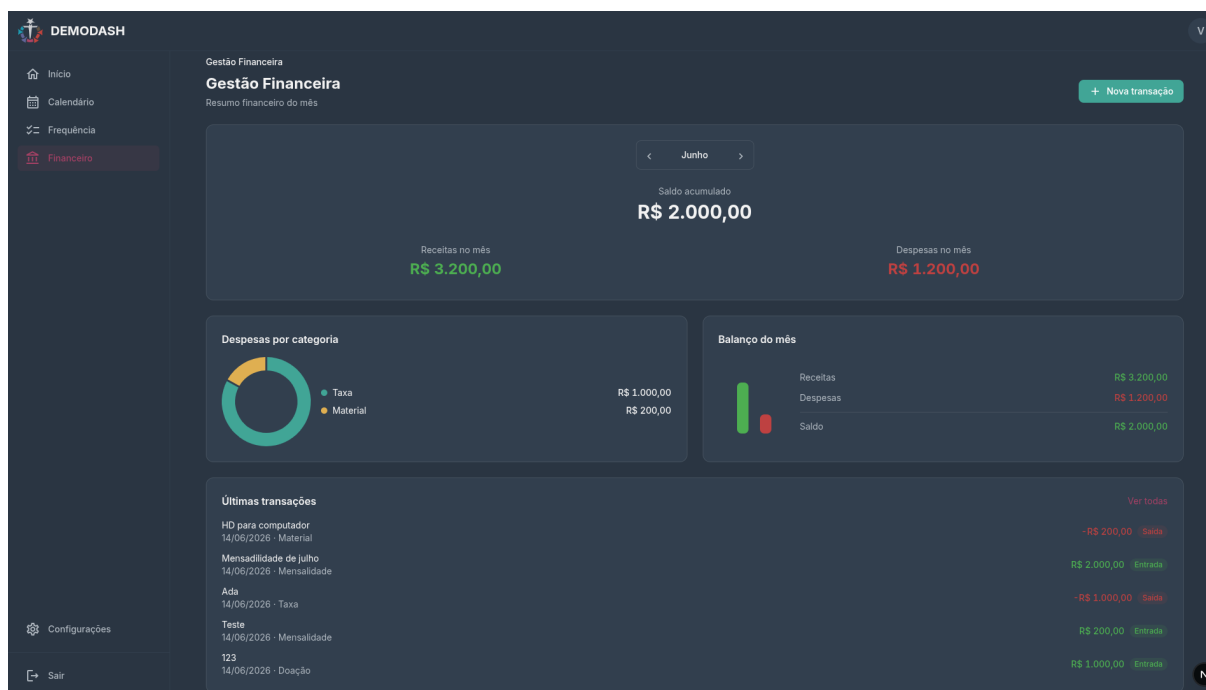
Figura 51 – Tela do *dashboard* de gestão financeira.



Figura 52 – Tela do *dashboard* de gestão financeira mobile.

DEMODASH

Gestão Financeira > Transações

Transações
Histórico de entradas e saídas

Filtros + Nova transação

Data	Tipo	Categoria	Valor	Descrição
14/06/2026	Saída	Material	-R\$ 200,00	HD para computador
HD para computador Comprovante Ver comprovante Criado em 14/06/2026, 05:57 Última edição em 14/06/2026, 05:57				
14/06/2026	Entrada	Mensalidade	R\$ 2.000,00	Mensalidade de julho
14/06/2026	Saída	Taxa	-R\$ 1.000,00	Ada
14/06/2026	Entrada	Mensalidade	R\$ 200,00	Teste
Teste Comprovante Criado em 14/06/2026, 04:07 Última edição em 14/06/2026, 04:07				
14/06/2026	Entrada	Doação	R\$ 1.000,00	123

Configurações

Sair

Figura 53 – Tela de transações financeiras.

The screenshot shows the 'Nova transação' (New Transaction) form in the DEMODASH application. The form is located in the main content area, with a sidebar on the left containing navigation links: Início, Calendário, Frequência, and Financeiro (highlighted). The top navigation bar shows the breadcrumb path: Gestão Financeira > Transações > Nova transação. The form itself is titled 'Nova transação' and includes the subtitle 'Registrar uma nova entrada ou saída'. The form fields are as follows:

- Tipo ***: A dropdown menu with 'Entrada' selected.
- Categoria ***: A dropdown menu with 'Doação' selected.
- Valor**: A text input field containing 'R\$ 0,00'.
- Descrição**: A text input field containing 'Ex.: Mensalidade de junho'.
- Data de competência ***: A date picker field showing '14/06/2026'.
- URL do comprovante**: A text input field containing 'https://...'.

At the bottom of the form, there are two buttons: 'Cancelar' and 'Registrar transação'.

Figura 54 – Tela de nova transação.

DEMODASH

Gestão Financeira > Transações > Nova transação

Nova transação

Registrar uma nova entrada ou saída

Tipo *

Entrada

Categoria *

Doação

Valor

R\$ 0,00

Obrigatório

Descrição

Ex.: Mensalidade de junho

Obrigatório

Data de competência *

14/06/2026

Obrigatório

URL do comprovante

https://...

Cancelar Registrar transação

Início Calendário Frequência Financeiro Configurações

Figura 55 – Tela de nova transação mobile.

DEMODASH

Gestão Financeira > Transações > Editar transação

Editar transação

Teste

Tipo *

Entrada

Categoria *

Mensalidade

Valor

R\$ 200,00

Obrigatório

Descrição

Teste

Obrigatório

Data de competência *

14/06/2026

Obrigatório

URL do comprovante

https://...

Cancelar Salvar alterações

Configurações

Figura 56 – Tela de edição de transação.

 DEMODASH

Editar transação

Mensadilidade de julho

Tipo *

Entrada

Categoria *

Mensalidade

Valor

R\$ 2.000,00

Obrigatório

Descrição

Mensadilidade de julho

Obrigatório

Data de competência *

14/06/2026

Obrigatório

URL do comprovante

https://...

Cancelar

Salvar alterações

 Início

 Calendário

 Frequência

 Financeiro

 Configurações

Figura 57 – Tela de edição de transação mobile.

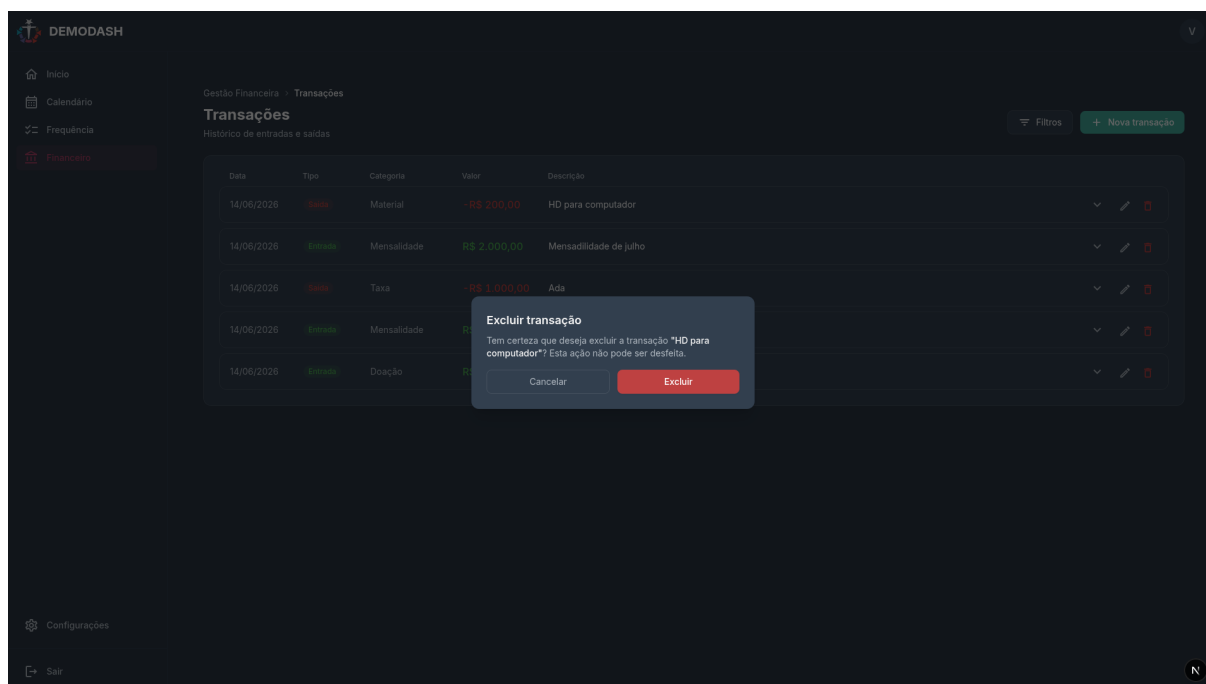


Figura 58 – Tela de exclusão de transação.

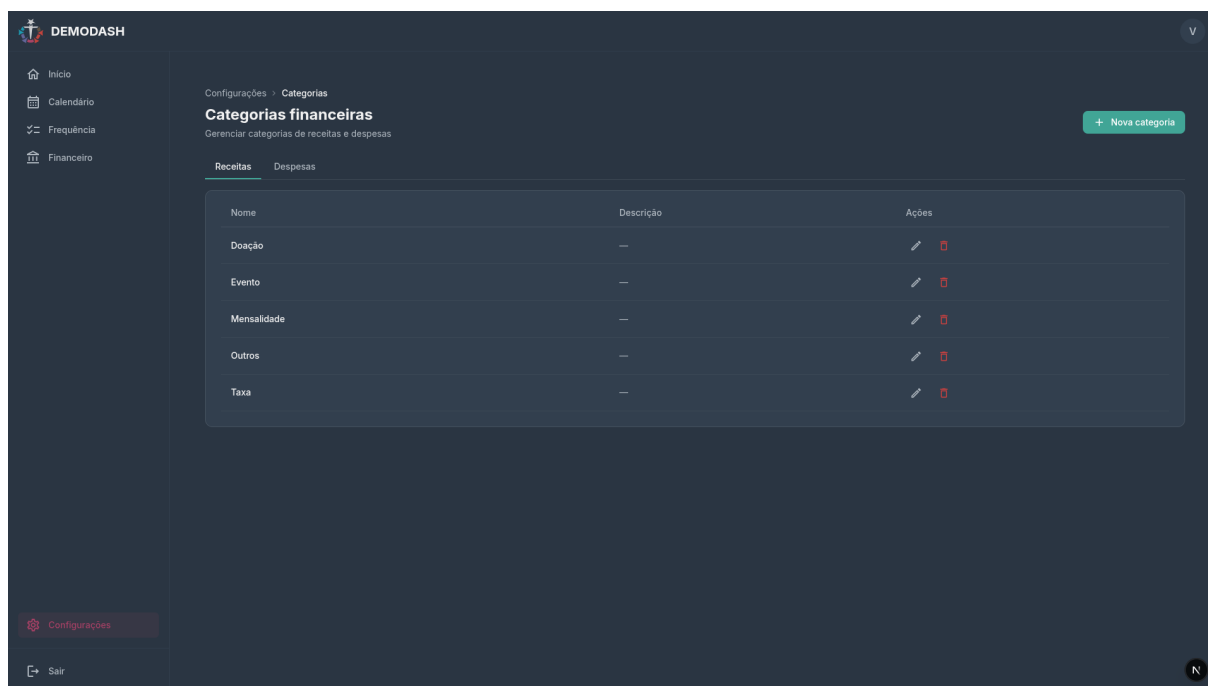


Figura 59 – Tela de categorias financeiras.

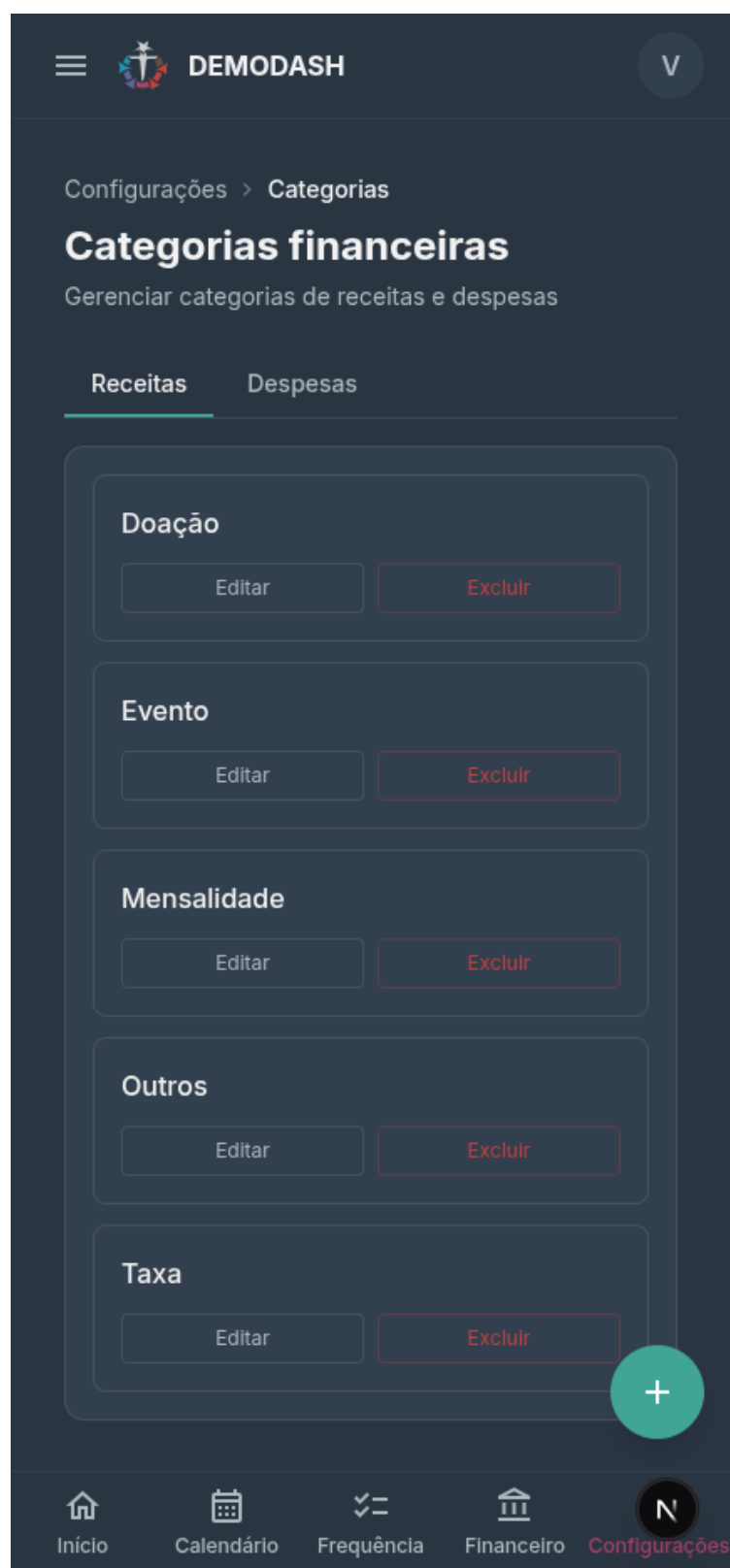


Figura 60 – Tela de categorias financeiras mobile.

The image shows a web application interface for 'DEMODASH'. On the left is a dark sidebar with navigation links: 'Início', 'Calendário', 'Frequência', 'Financeiro', 'Configurações' (highlighted with a gear icon), and 'Sair'. The main content area has a breadcrumb trail 'Configurações > Categorias > Nova categoria' and a 'Voltar' link. The title 'Nova Categoria' is centered above a form. The form contains the following fields: 'Tipo' (with a dropdown menu showing 'Receita'), 'Nome' (with a text input field containing 'Ex.: Patrocínio'), 'Obrigatório' (with a checkbox), and 'Descrição (opcional)' (with a text area containing 'Descreva o uso desta categoria...'). At the bottom of the form are two buttons: 'Cancelar' and 'Criar Categoria' (in green). A user profile icon with the letter 'N' is in the bottom right corner.

Figura 61 – Tela de adição de categoria financeira.



Figura 62 – Tela de balanço financeiro mobile.

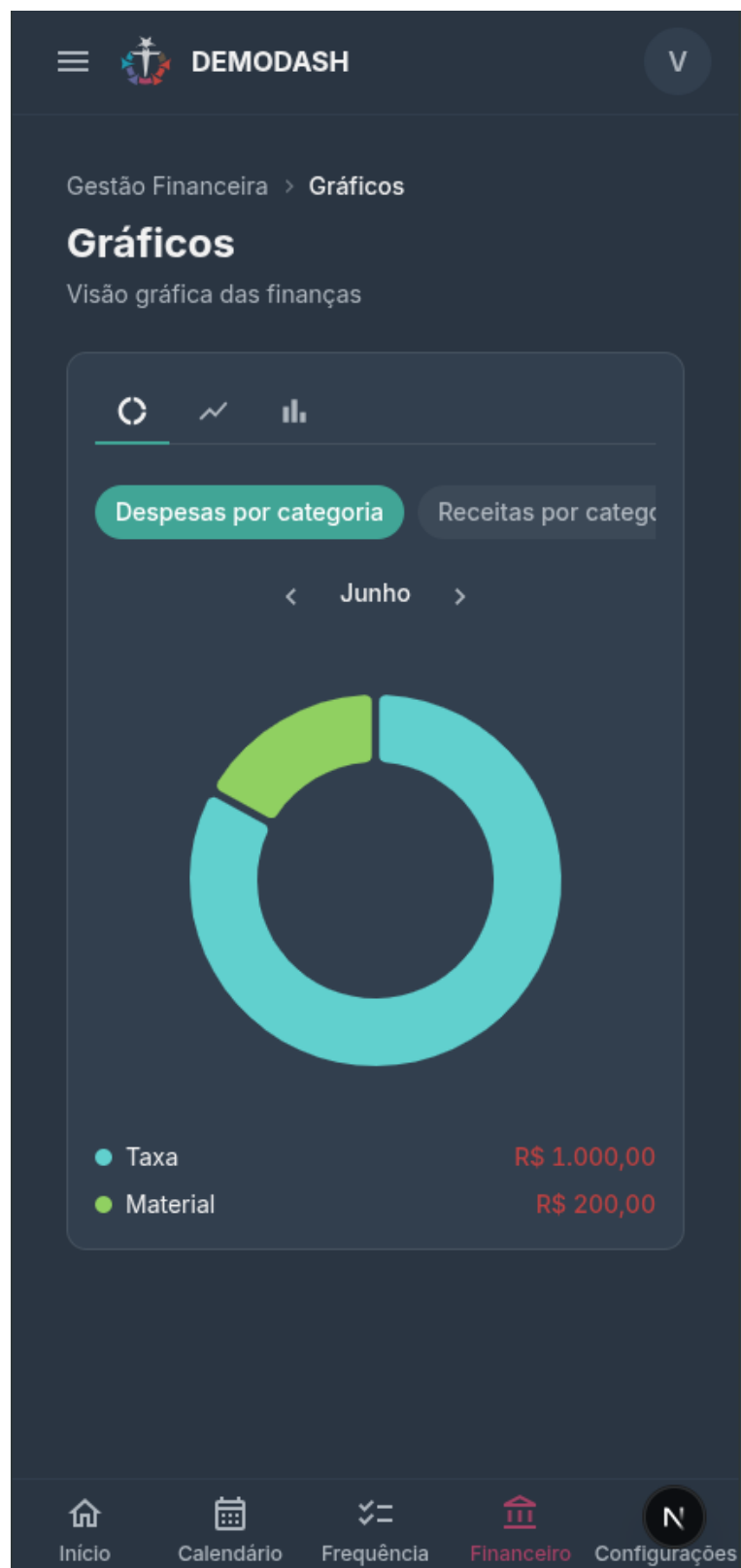


Figura 63 – Tela de gráfico de pizza mobile.



Figura 64 – Tela de gráfico de linha mobile.



Figura 65 – Tela de gráfico de barras mobile.

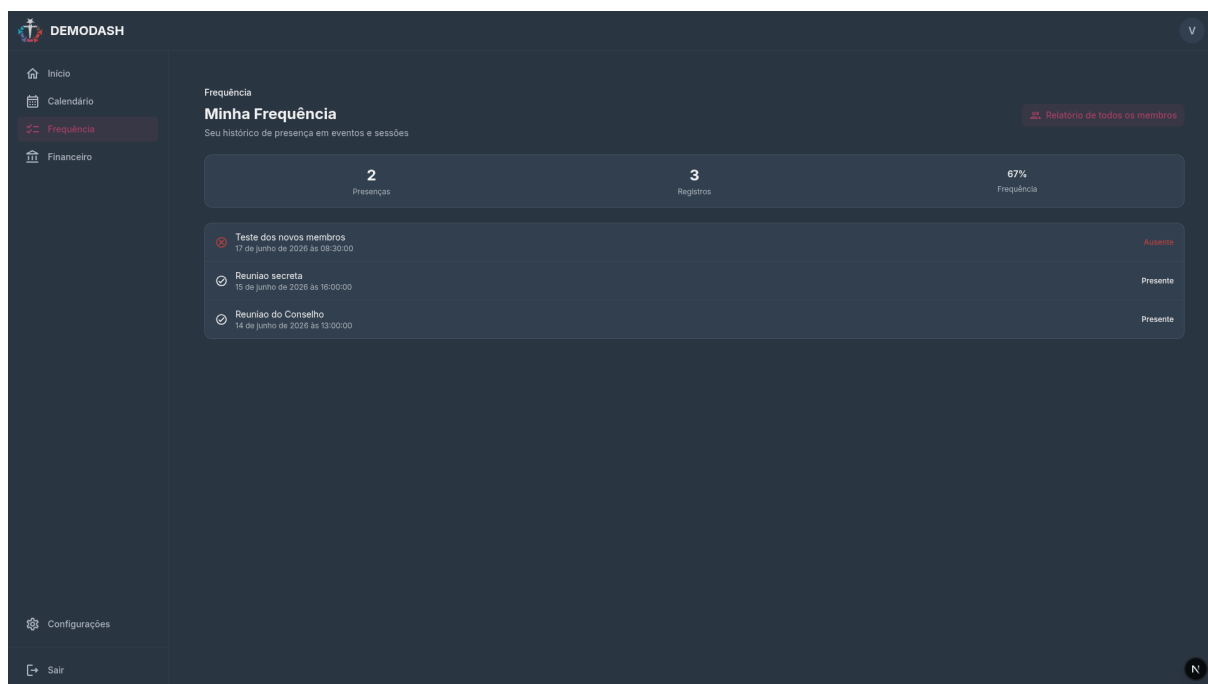


Figura 66 – Tela de frequência.

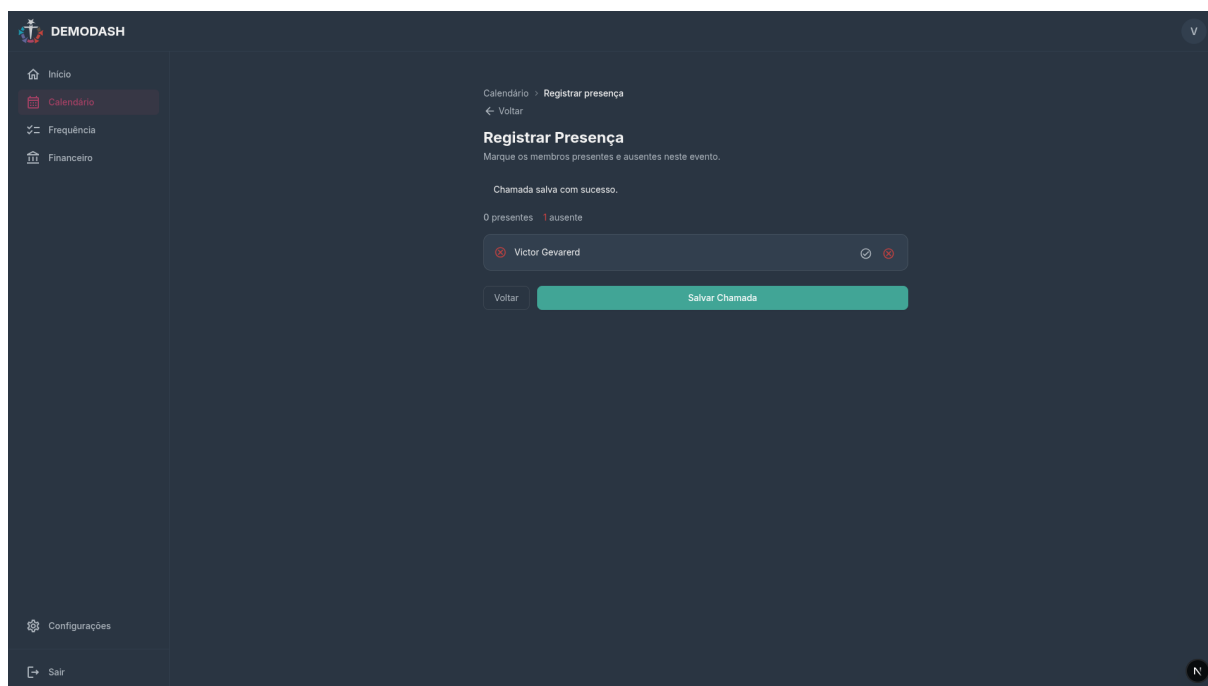


Figura 67 – Tela de registro de presença.

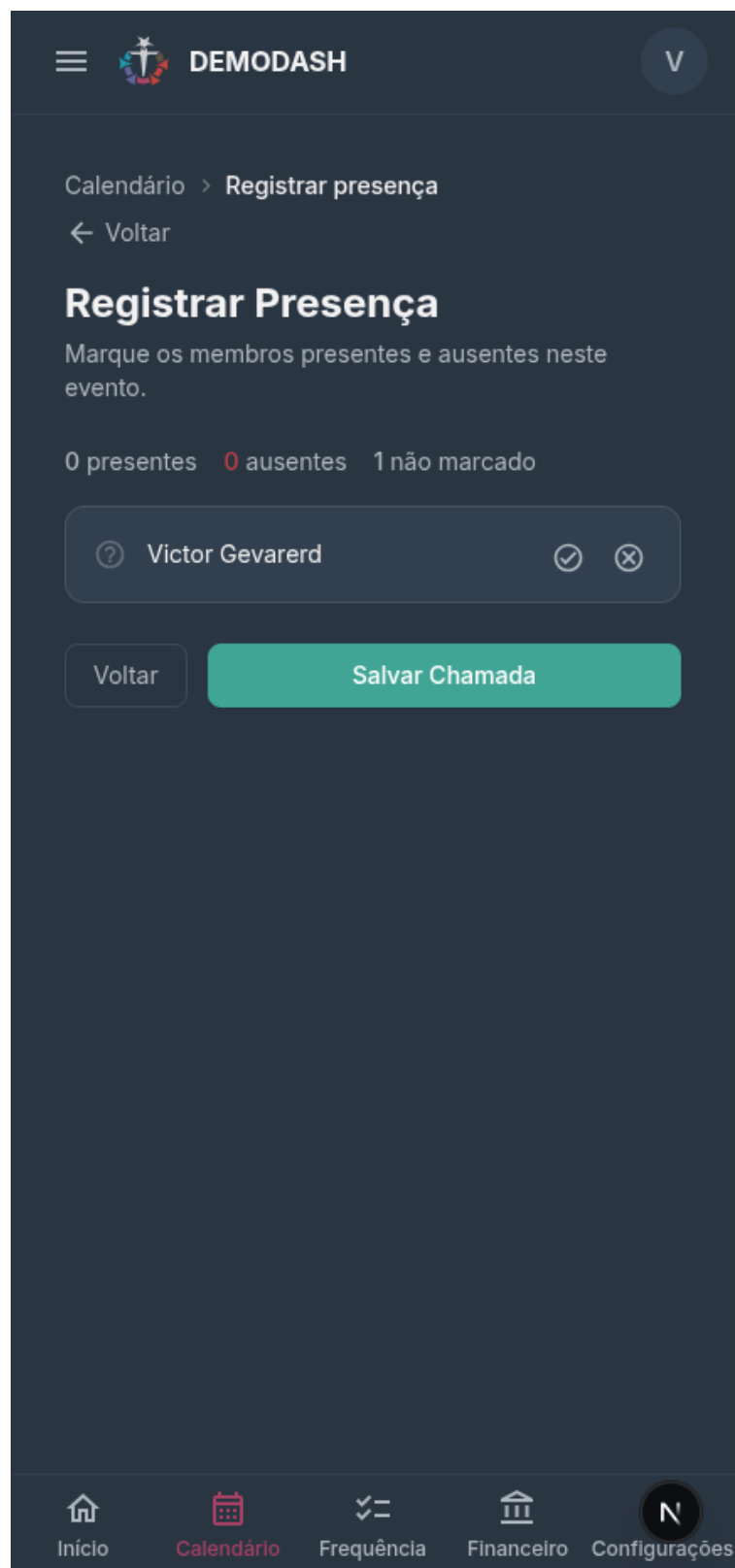


Figura 68 – Tela de registro de presença mobile.

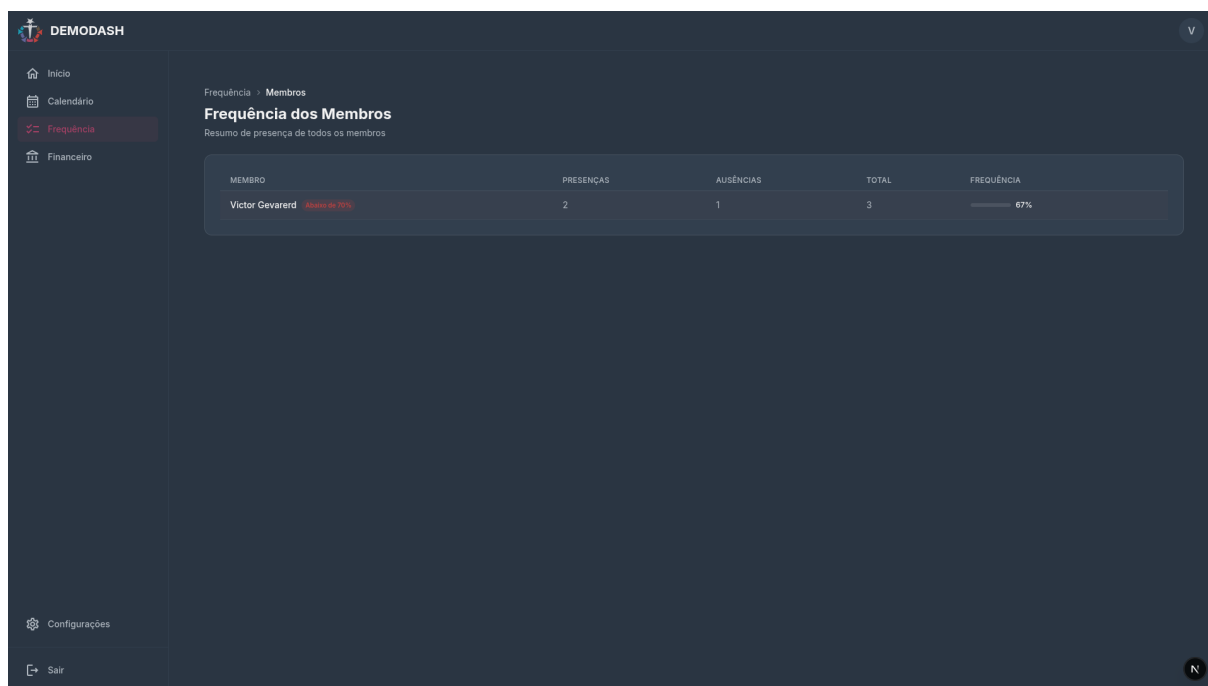


Figura 69 – Tela de relatório de frequência.



Figura 70 – Tela de relatório de frequência mobile.



Figura 71 – Tela de minha frequência mobile.

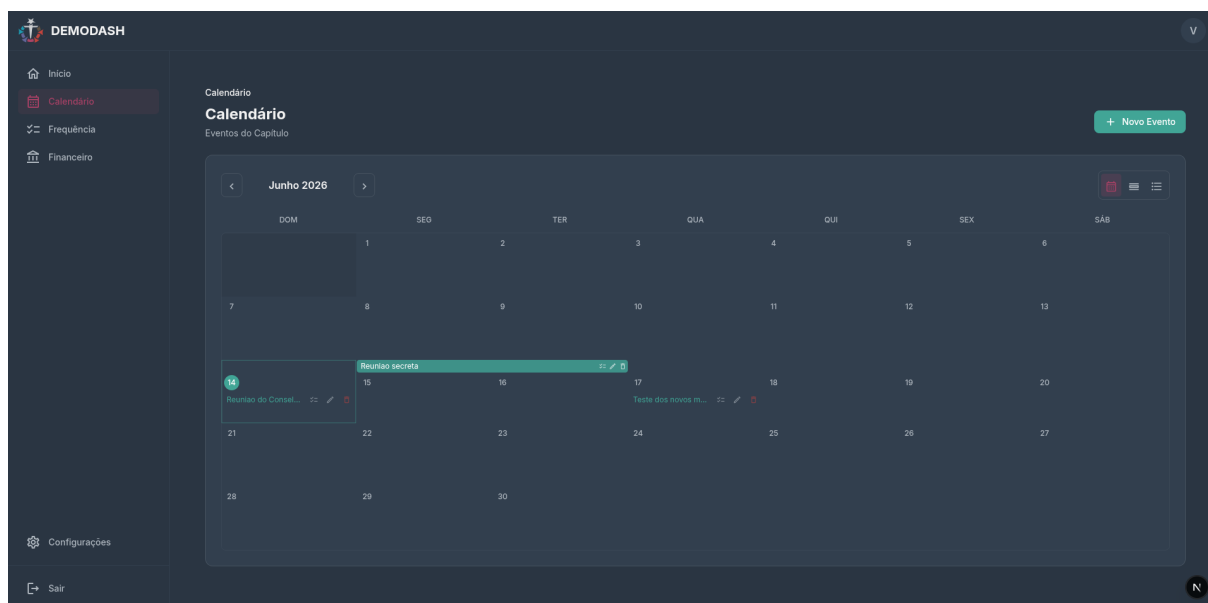


Figura 72 – Tela de calendário (visão mensal).

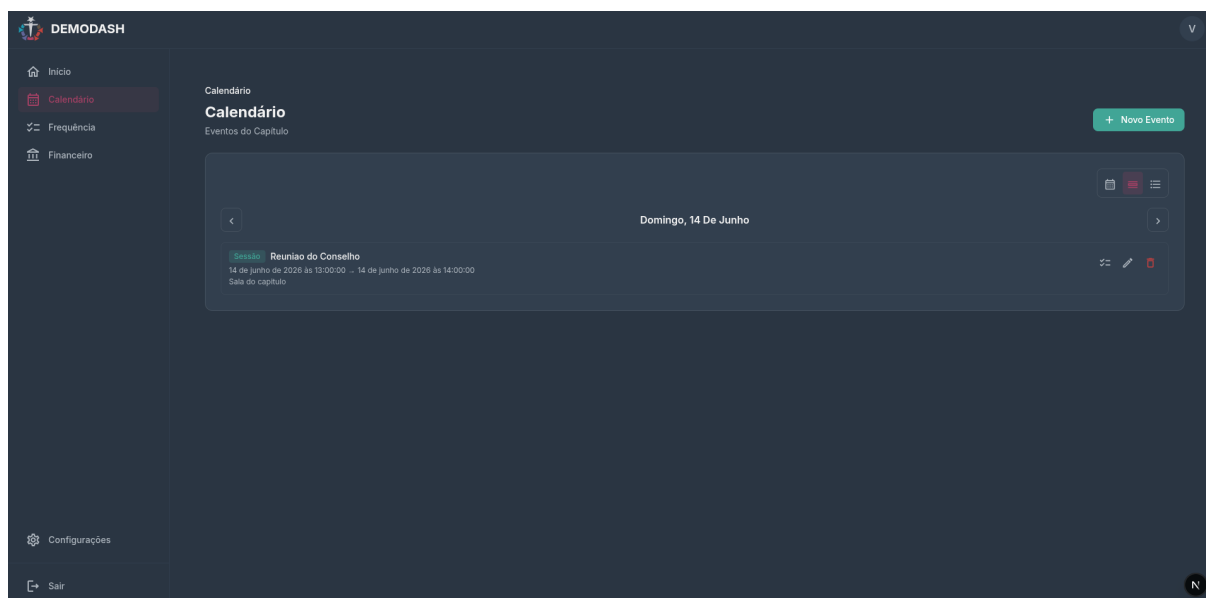


Figura 73 – Tela de calendário (visão diária).

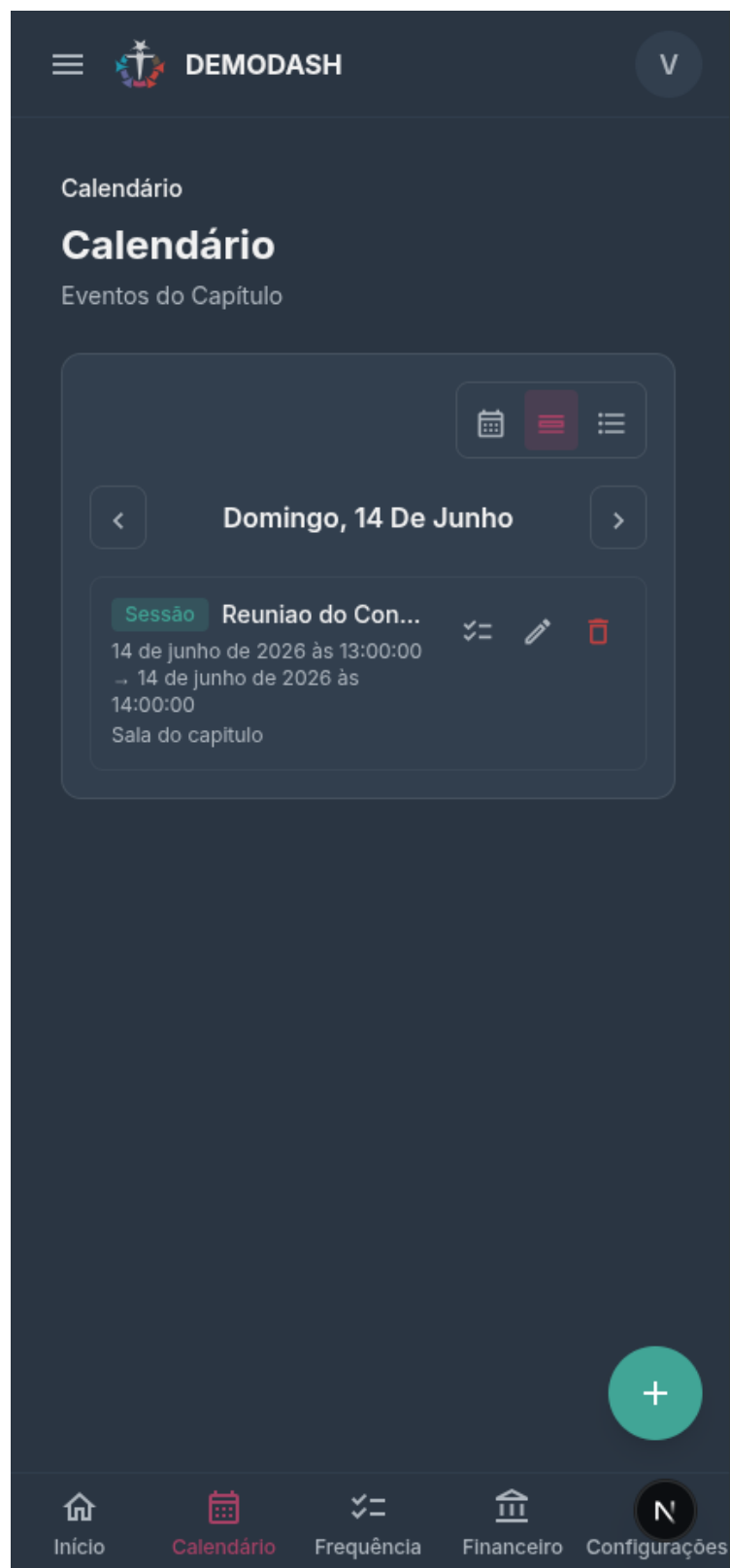


Figura 74 – Tela de calendário (visão diária) mobile.

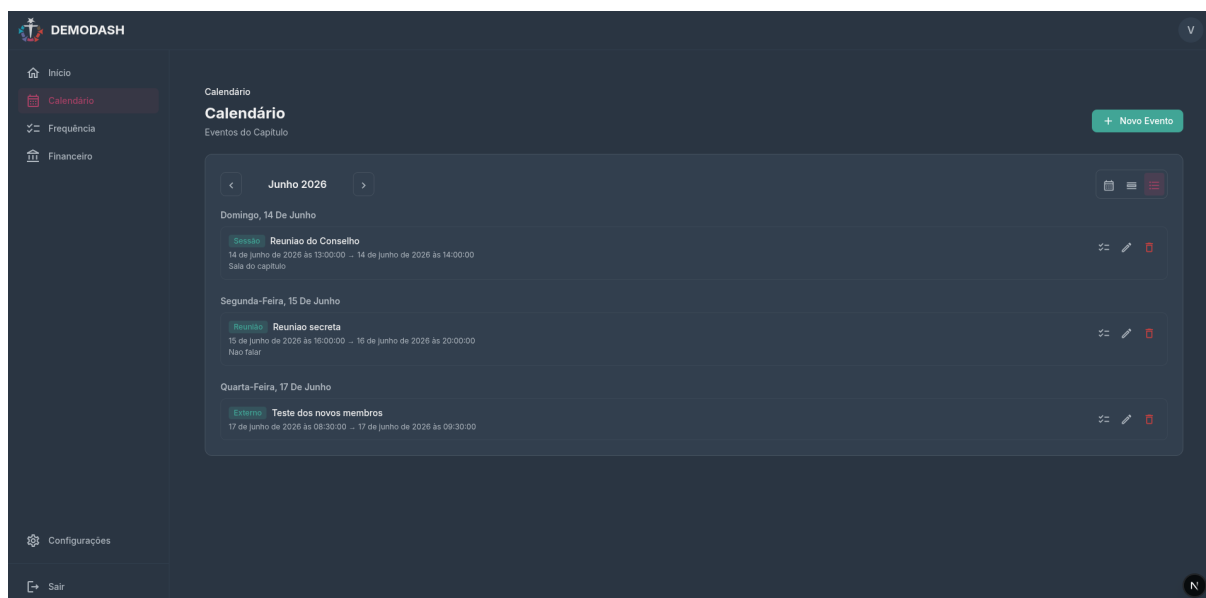


Figura 75 – Tela de lista de eventos do calendário.



Figura 76 – Tela de lista de eventos do calendário mobile.

The image shows a web application interface for creating a new event. The header bar is dark blue with the 'DEMODASH' logo on the left and a user profile icon 'V' on the right. A sidebar on the left contains navigation links: 'Início', 'Calendário' (highlighted in red), 'Frequência', 'Financeiro', 'Configurações', and 'Sair'. The main content area is titled 'Novo Evento' and contains a form with the following fields: 'Título' (with an example 'Ex.: Sessão Ordinária de Junho'), 'Obrigatório' (checkbox), 'Descrição (opcional)' (text area with placeholder 'Descreva o evento...'), 'Tipo' (dropdown menu with 'Sessão' selected and an 'Evento de dia inteiro' checkbox), 'Data e hora de início' (calendar icon and placeholder 'dd/mm/aaaa hh:mm'), 'Data e hora de término (opcional)' (calendar icon and placeholder 'dd/mm/aaaa hh:mm'), 'Participantes' (dropdown menu with placeholder 'Selecione os participantes...'), and 'Local (opcional)' (text field with placeholder 'Ex.: Salão do Capítulo'). At the bottom of the form are two buttons: 'Cancelar' and 'Criar Evento'.

DEMODASH

V

Início

Calendário

Frequência

Financeiro

Configurações

Sair

Novo Evento

Título

Ex.: Sessão Ordinária de Junho

Obrigatório

Descrição (opcional)

Descreva o evento...

Tipo *

Sessão

☐ Evento de dia inteiro

Data e hora de início *

dd/mm/aaaa hh:mm

Data e hora de término (opcional)

dd/mm/aaaa hh:mm

Participantes

Selecione os participantes...

Local (opcional)

Ex.: Salão do Capítulo

Cancelar Criar Evento

N

Figura 77 – Tela de novo evento.

DEMODASH

Calendário > Editar evento

← Voltar

Editar Evento

Título

Reuniao secreta

Obrigatório

Descrição (opcional)

Nao falar

Tipo *

Reunião

☐ Evento de dia inteiro

Data e hora de início *

15/06/2026 16:00

Data e hora de término (opcional)

16/06/2026 20:00

Participantes

Victor Gevared

Local (opcional)

Ex.: Salão do Capítulo

← Cancelar Salvar Alterações

Figura 78 – Tela de edição de evento.

The image shows a mobile application interface for editing an event. The app is titled "DEMODASH" and has a user profile icon with the letter "V" in the top right corner. The main form is titled "Evento" and contains the following fields:

- Titulo:** A text input field containing "Reuniao do Conselho".
- Obrigatório:** A checkbox that is currently checked.
- Descrição (opcional):** A text input field containing "Descreva o evento...".
- Tipo *:** A dropdown menu with "Sessão" selected.
- Evento de dia inteiro:** An unchecked checkbox.
- Data e hora de início *:** A date and time picker showing "14/06/2026 13:00".
- Data e hora de término (opcional):** A date and time picker showing "14/06/2026 14:00".
- Participantes:** A dropdown menu with "Victor Gevarerd" selected.
- Local (opcional):** A text input field containing "Sala do capitulo".

At the bottom of the form are two buttons: "Cancelar" (with a back arrow) and "Salvar Alterações" (in green). The bottom navigation bar has five icons: a house for "Início", a calendar for "Calendário" (highlighted in red), a list for "Frequência", a building for "Financeiro", and a person for "Configurações".

Figura 79 – Tela de edição de evento mobile.

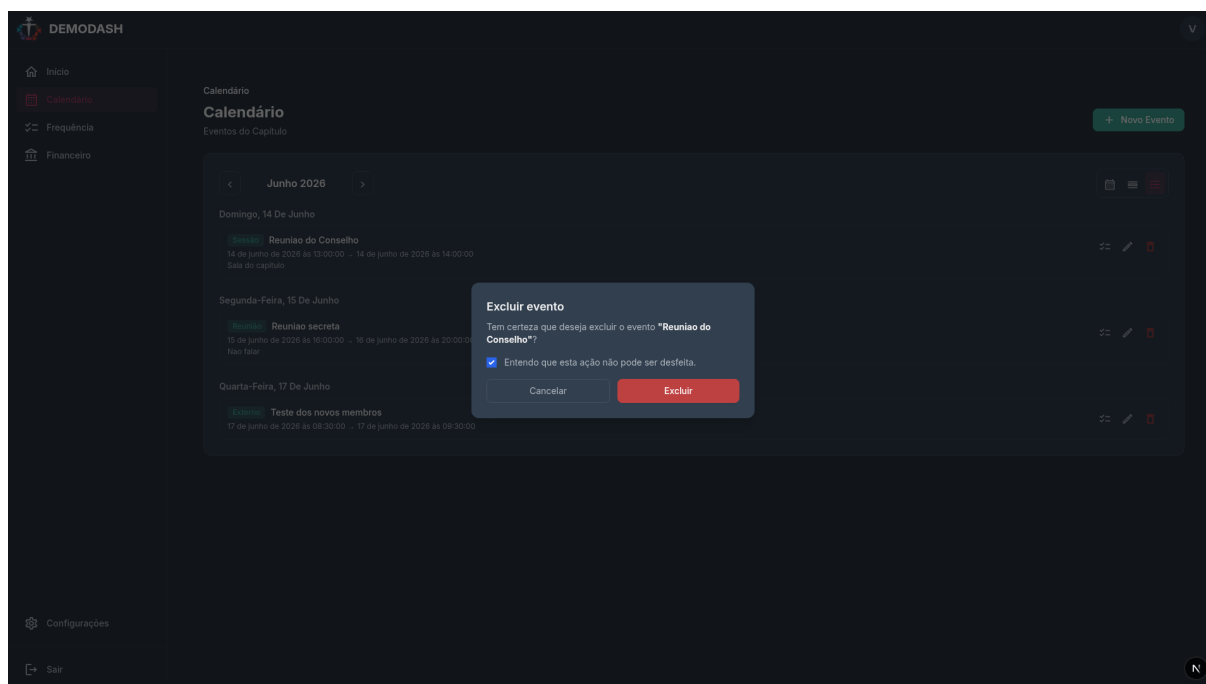


Figura 80 – Tela de exclusão de evento.

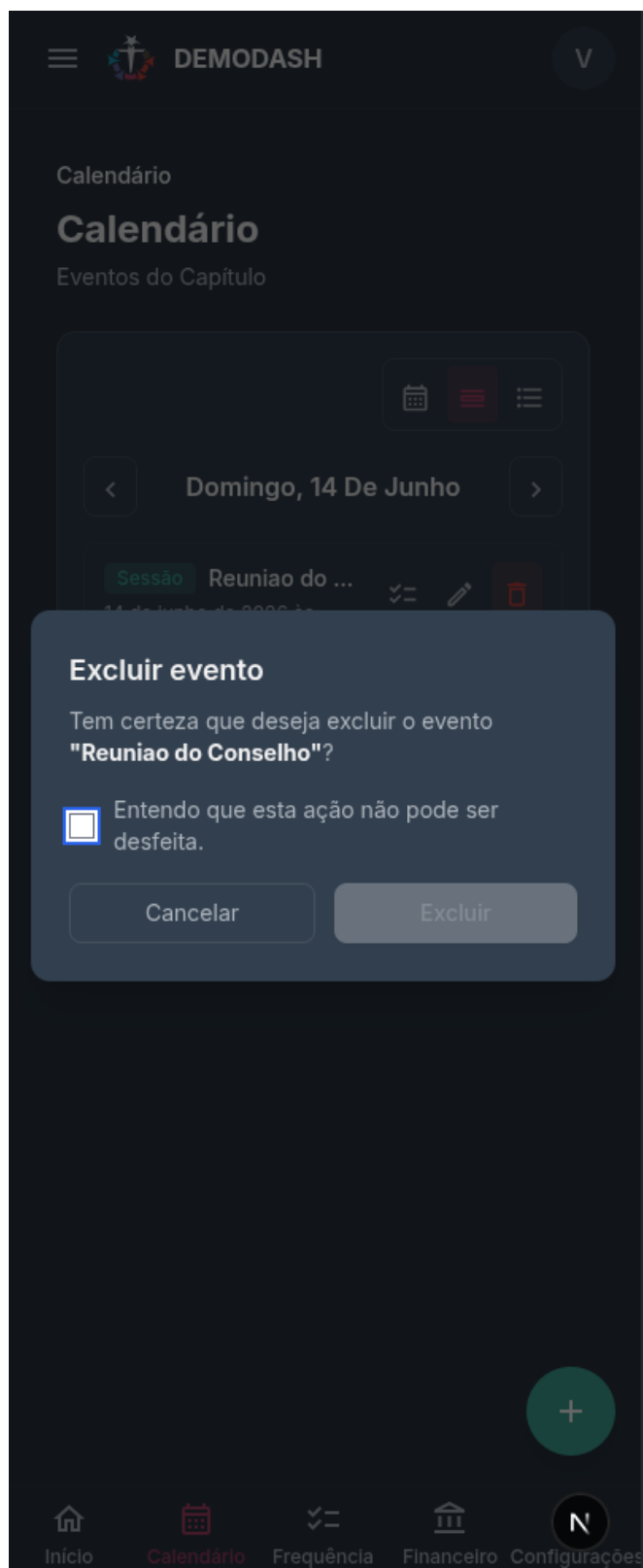


Figura 81 – Tela de exclusão de evento mobile.

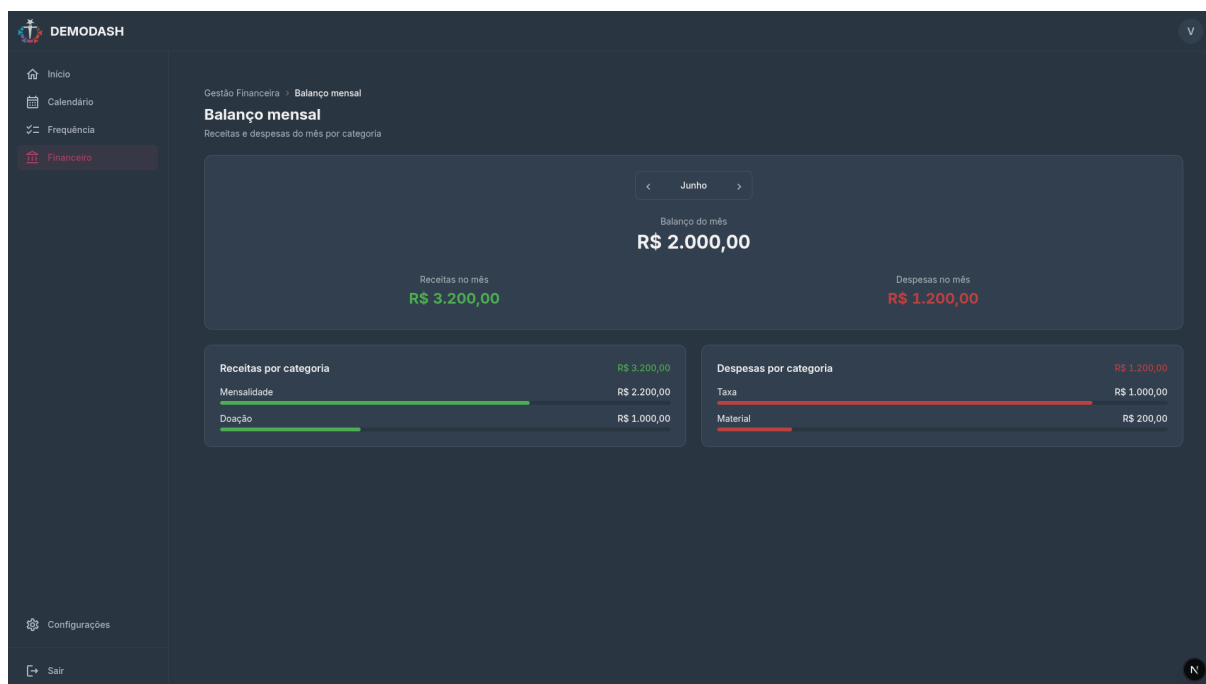


Figura 82 – Tela de transparência — balanço mensal.

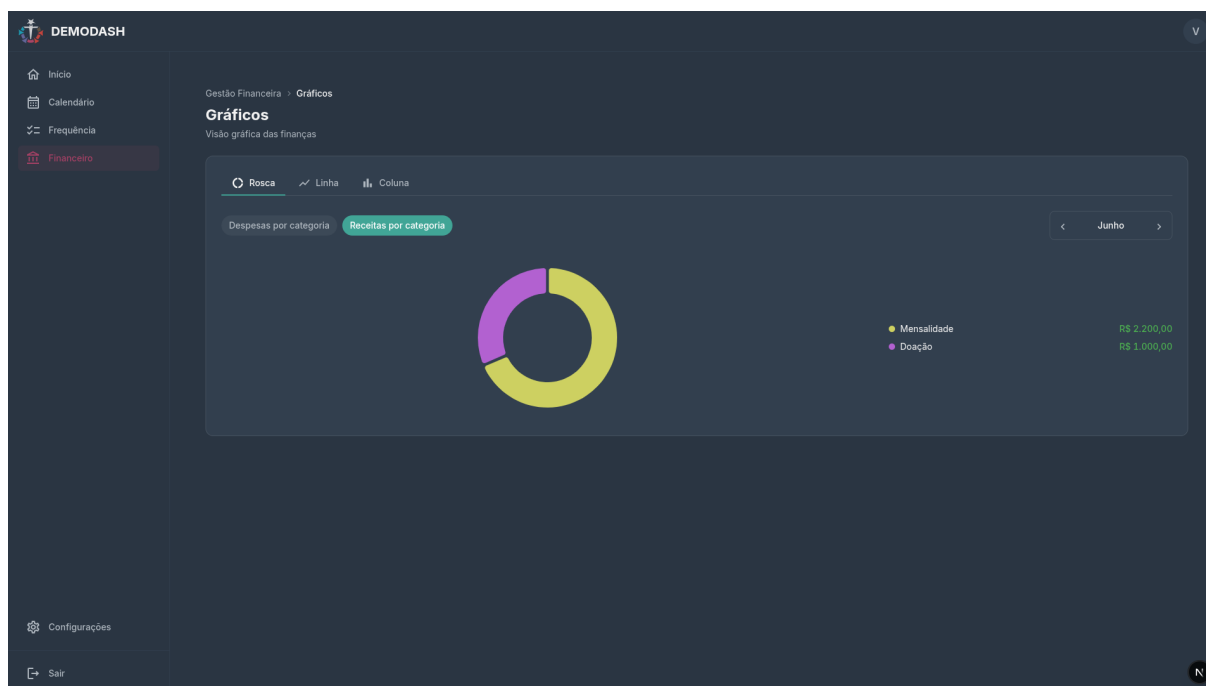


Figura 83 – Tela de transparência — gráfico de rosca.

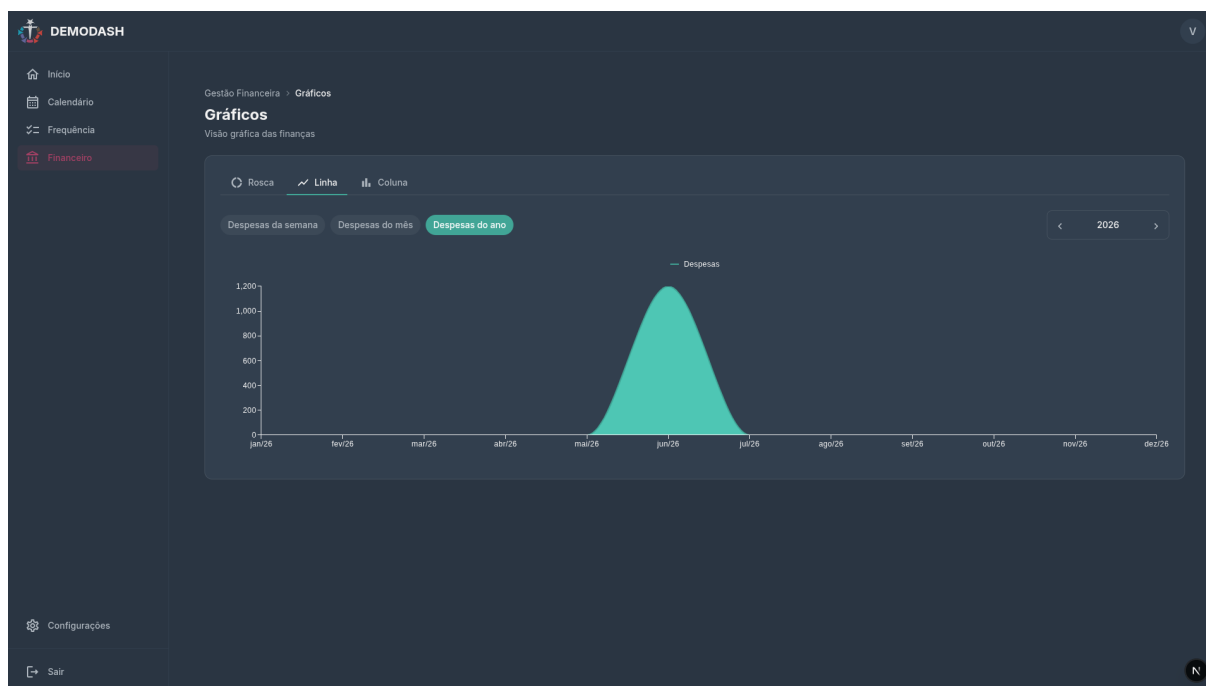


Figura 84 – Tela de transparência — gráfico de linha.

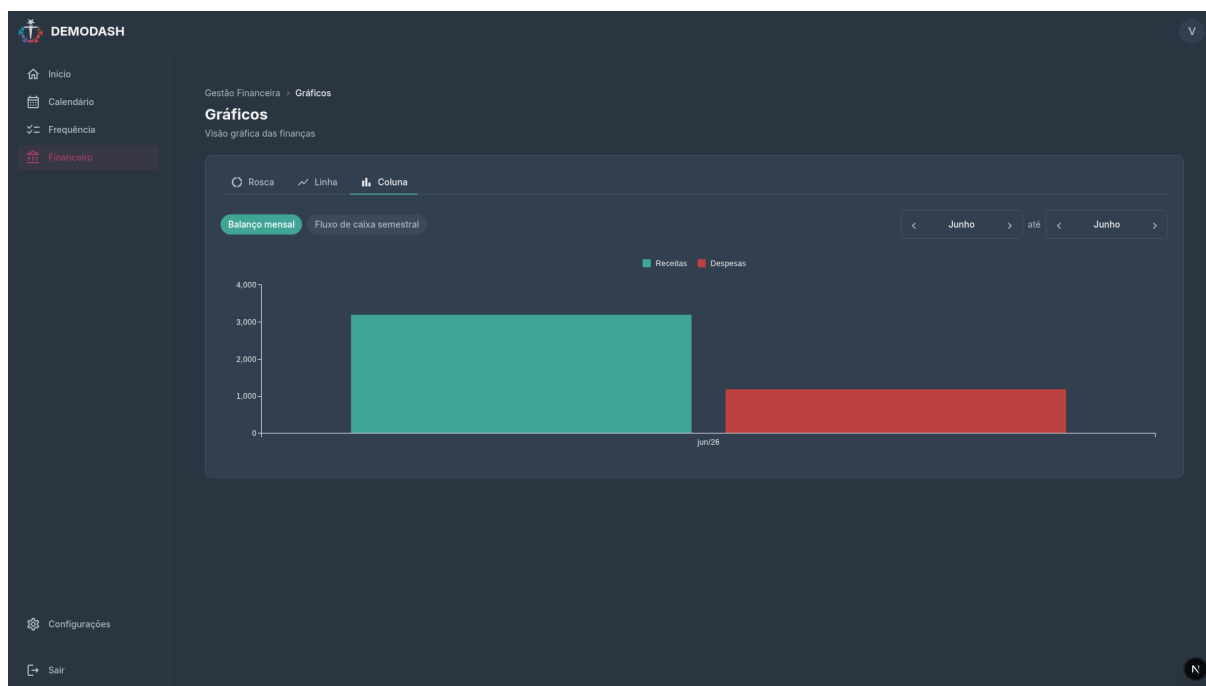


Figura 85 – Tela de transparência — gráfico de colunas do balanço mensal.

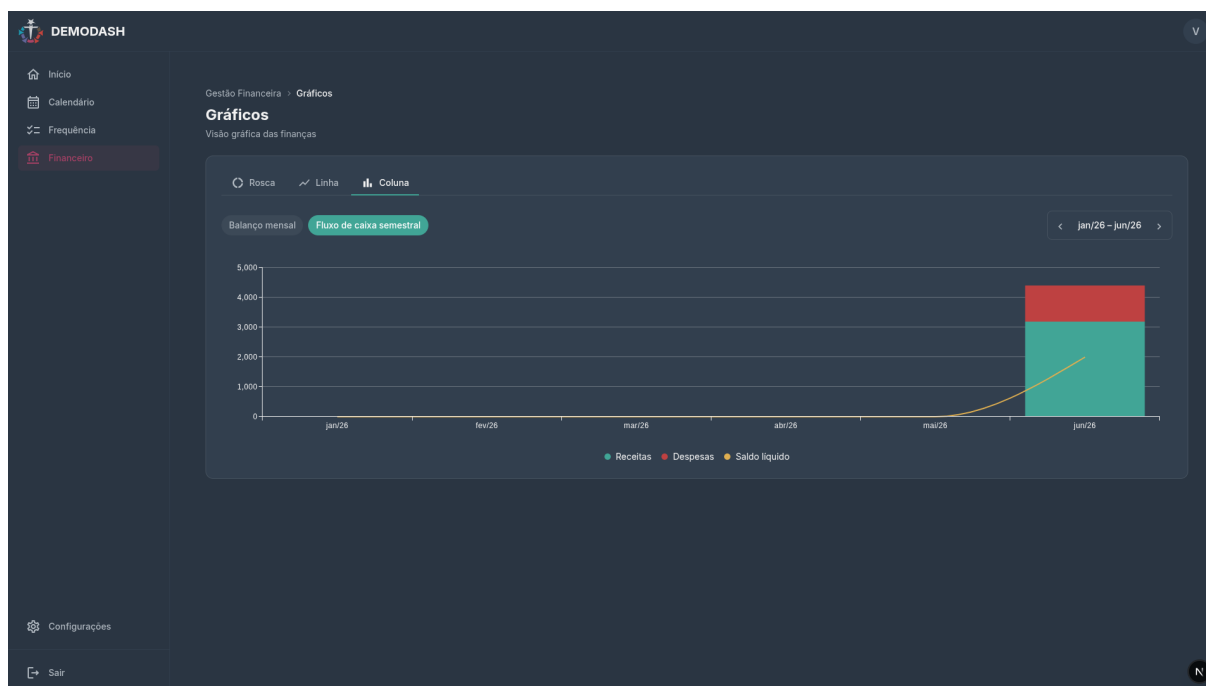


Figura 86 – Tela de transparência — gráfico de fluxo de caixa do semestre.

```

data class Membro(
    val id: UUID,
    val email: String,
    val nome: String,
    val idDeMolay: String,
    val dataNascimento: LocalDate,
    val dataIniciacao: LocalDate,
    val dataElevacao: LocalDate?,
    val createdAt: Instant,
    val updatedAt: Instant,
) {
    // Regra oficial DeMolay: idade < 21 → ATIVO; idade ≥ 21 → SENIOR.
    fun calcularStatus(dataInteresse: LocalDate = LocalDate.now()): StatusMembro {
        val idade = Period.between(dataNascimento, dataInteresse).years
        return if (idade < 21) StatusMembro.ATIVO else StatusMembro.SENIOR
    }

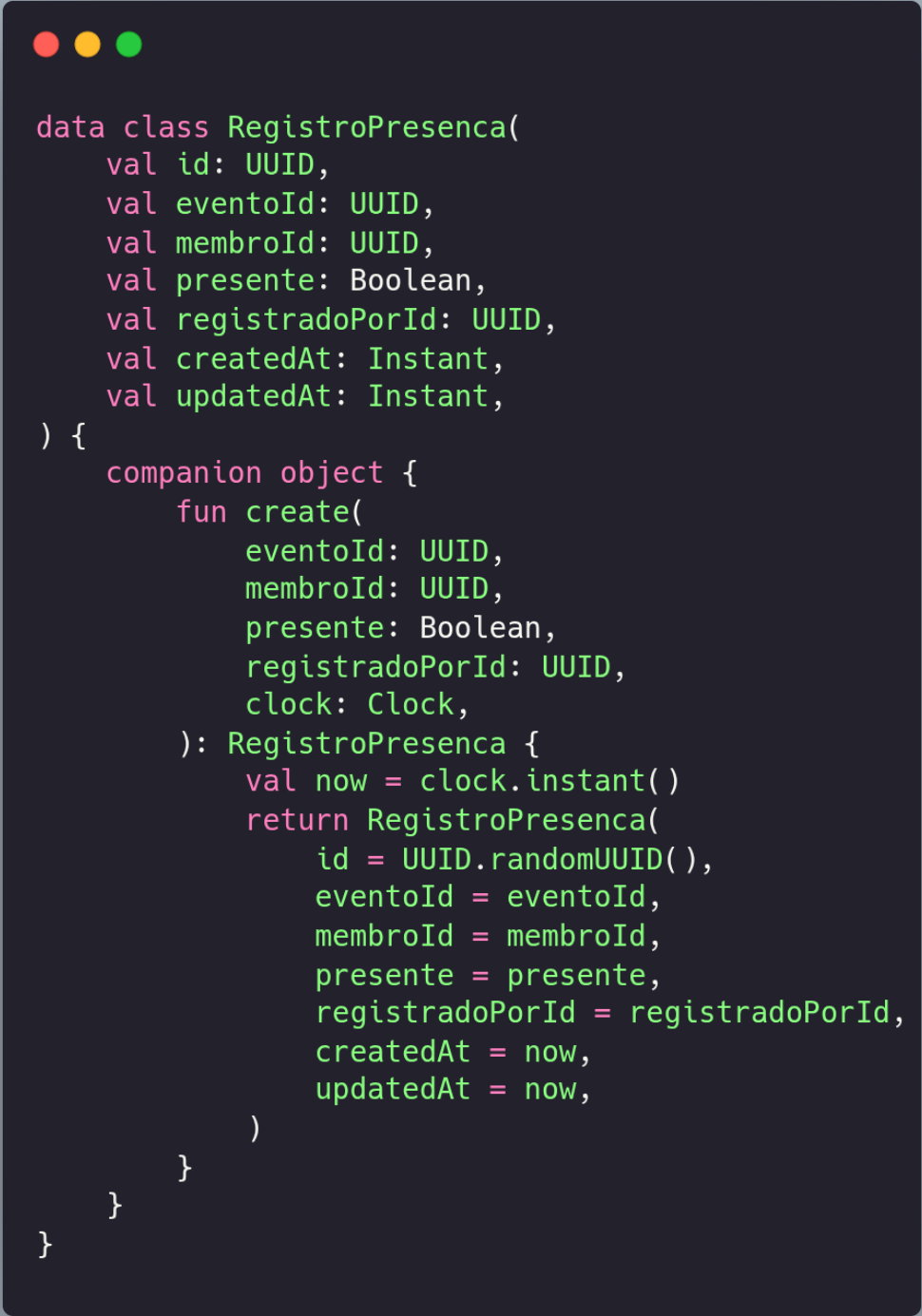
    companion object {
        fun create(
            email: String,
            nome: String,
            idDeMolay: String,
            dataNascimento: LocalDate,
            dataIniciacao: LocalDate,
            dataElevacao: LocalDate?,
            clock: Clock,
        ): Membro {
            val hoje = LocalDate.now(clock)
            val erros = mutableMapOf<String, String>()

            if (!dataNascimento.isBefore(hoje)) {
                erros["dataNascimento"] = "A data de nascimento deve ser anterior à data atual."
            }
            if (!dataIniciacao.isAfter(dataNascimento)) {
                erros["dataIniciacao"] = "A data de iniciação deve ser posterior à data de nascimento."
            }
            if (dataElevacao != null && !dataElevacao.isAfter(dataIniciacao)) {
                erros["dataElevacao"] = "A data de elevação deve ser posterior à data de iniciação."
            }
            if (erros.isNotEmpty()) {
                throw ValidacaoException(
                    message = "Datas do membro inválidas.",
                    details = erros,
                )
            }

            val now = clock.instant()
            return Membro(
                id = UUID.randomUUID(),
                email = email,
                nome = nome,
                idDeMolay = idDeMolay,
                dataNascimento = dataNascimento,
                dataIniciacao = dataIniciacao,
                dataElevacao = dataElevacao,
                createdAt = now,
                updatedAt = now,
            )
        }
    }
}

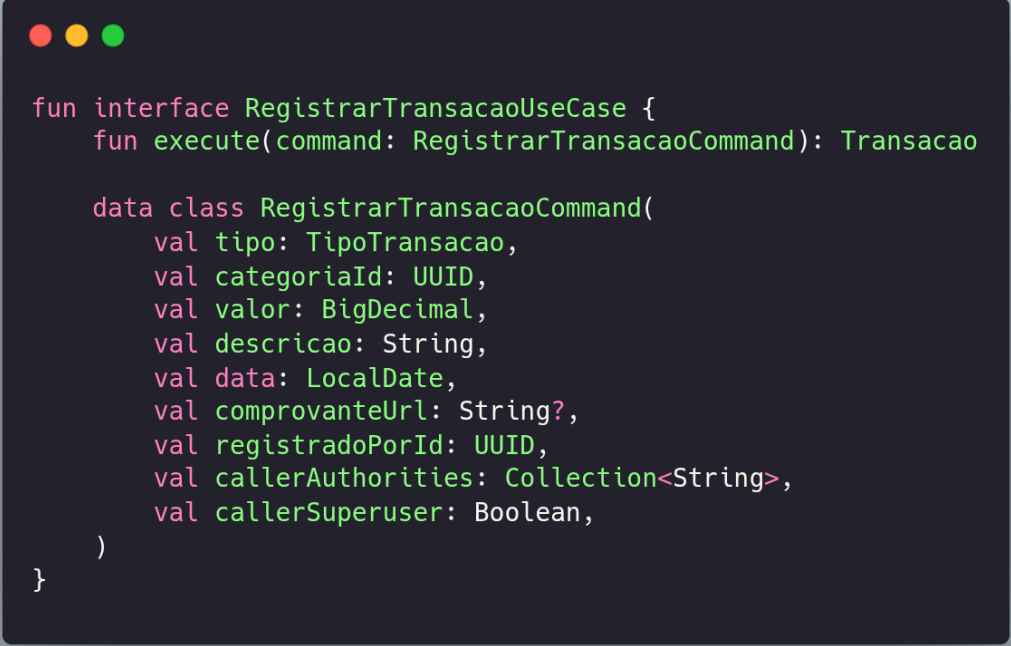
```

Figura 87 – Modelo Membro - Membro.kt.



```
data class RegistroPresenca(  
    val id: UUID,  
    val eventoId: UUID,  
    val membroId: UUID,  
    val presente: Boolean,  
    val registradoPorId: UUID,  
    val createdAt: Instant,  
    val updatedAt: Instant,  
) {  
    companion object {  
        fun create(  
            eventoId: UUID,  
            membroId: UUID,  
            presente: Boolean,  
            registradoPorId: UUID,  
            clock: Clock,  
        ): RegistroPresenca {  
            val now = clock.instant()  
            return RegistroPresenca(  
                id = UUID.randomUUID(),  
                eventoId = eventoId,  
                membroId = membroId,  
                presente = presente,  
                registradoPorId = registradoPorId,  
                createdAt = now,  
                updatedAt = now,  
            )  
        }  
    }  
}
```

Figura 88 – Registro de Presença - RegistroPresenca.kt.



```
fun interface RegistrarTransacaoUseCase {  
    fun execute(command: RegistrarTransacaoCommand): Transacao  
  
    data class RegistrarTransacaoCommand(  
        val tipo: TipoTransacao,  
        val categoriaId: UUID,  
        val valor: BigDecimal,  
        val descricao: String,  
        val data: LocalDate,  
        val comprovanteUrl: String?,  
        val registradoPorId: UUID,  
        val callerAuthorities: Collection<String>,  
        val callerSuperuser: Boolean,  
    )  
}
```

Figura 89 – Interface da porta de entrada RegistrarTransacaoUseCase - RegistrarTransacaoUseCase.kt.

```

@Service
@Transactional
class RegistrarTransacaoService(
    private val transacaoRepository: TransacaoRepository,
    private val categoriaFinanceiraRepository:
        CategoriaFinanceiraRepository,
    private val clock: Clock,
) : RegistrarTransacaoUseCase {
    override fun execute(command:
        RegistrarTransacaoUseCase.RegistrarTransacaoCommand): Transacao
    {
        val permissaoNecessaria =
            when (command.tipo) {
                TipoTransacao.ENTRADA ->
                    Permissao.FINANCEIRO_REGISTRAR_ENTRADA.codigo
                TipoTransacao.SAIDA ->
                    Permissao.FINANCEIRO_REGISTRAR_SAIDA.codigo
            }

        if (!command.callerSuperuser && permissaoNecessaria !in
            command.callerAuthorities) {
            throw
                TipoTransacaoNaoAutorizadoException(command.tipo)
        }

        val categoria =
            categoriaFinanceiraRepository.findById(command.categoriaId)
                ?: throw ResourceNotFoundException("Categoria",
                    command.categoriaId)

        if (categoria.tipo != command.tipo) {
            throw ValidacaoException(
                message = "A categoria '${categoria.nome}' não
                    pertence ao tipo '${command.tipo.name}'.",
                details = mapOf("categoriaId" to "Categoria
                    incompatível com o tipo da transação."),
            )
        }

        val transacao =
            Transacao.create(
                tipo = command.tipo,
                categoria = categoria,
                valor = command.valor,
                descricao = command.descricao,
                data = command.data,
                comprovanteUrl = command.comprovanteUrl,
                registradoPorId = command.registradoPorId,
                clock = clock,
            )

        return transacaoRepository.salvar(transacao)
    }
}

class TipoTransacaoNaoAutorizadoException(
    tipo: TipoTransacao,
) : ApiException(
    message = "Usuário não possui permissão para registrar
        transações do tipo '${tipo.name}'.",
    httpStatus = HttpStatus.FORBIDDEN,
    errorCode = "TIPO_TRANSACAO_NAO_AUTORIZADO",
    userAction = "Você não tem permissão para registrar
        esse tipo de transação.",
)

```

Figura 90 – Implementação RegistrarTransacaoService - RegistrarTransacaoService.kt.

```

@RestController
@RequestMapping("/v1/financeiro/transacoes")
class TransacaoController(
    private val registrarTransacaoUseCase:
    RegistrarTransacaoUseCase,
    // demais casos de uso omitidos para brevidade
    private val webMapper: TransacaoWebMapper,
) {
    @PostMapping
    @PreAuthorize(
        "hasAnyAuthority('FINANCEIRO:REGISTRAR_ENTRADA','FINANCEIRO:REGISTRAR_SAIDA')"
    )
    fun registrar(
        @Valid @RequestBody request: RegistrarTransacaoRequest,
        @AuthenticationPrincipal principal: UsuarioPrincipal,
    ): ResponseEntity<TransacaoResponse> {
        val command =
        RegistrarTransacaoUseCase.RegistrarTransacaoCommand(
            tipo                = request.tipo,
            categoriaId         = request.categoriaId,
            valor                = request.valor,
            descricao            = request.descricao,
            data                 = request.data,
            comprovanteUrl       = request.comprovanteUrl,
            registradoPorId      = principal.usuarioId,
            callerAuthorities    = principal.authorityNames,
            callerSuperuser      = principal.superuser,
        )
        val transacao =
        registrarTransacaoUseCase.execute(command)
        return ResponseEntity
            .status(HttpStatus.CREATED)
            .body(webMapper.toResponse(transacao))
    }
}

```

Figura 91 – Método registrar() do TransacaoController - TransacaoController.kt.

```

export async function serverFetch<T>(opts: FetchOptions):
Promise<ServerActionResponse<T>> {
  const cookieStore = await cookies()
  const token = cookieStore.get('session')?.value

  let response: Response
  try {
    response = await fetch(`${env.SERVER_API_URL}${opts.path}`,
  {
    method: opts.method,
    headers: {
      ...(opts.body !== undefined ? { 'Content-Type':
'application/json' } : {}),
      ...(token ? { Cookie: `SESSION=${token}` } : {}),
    },
    body: opts.body !== undefined ? JSON.stringify(opts.body)
: undefined,
    cache: 'no-store',
  })
  } catch (err) {
    return {
      success: false,
      error: new ClientError(
        'NETWORK_ERROR',
        err instanceof Error ? err.message : 'Erro de rede',
        'Não foi possível conectar ao servidor.'
      ).toJSON(),
    }
  }

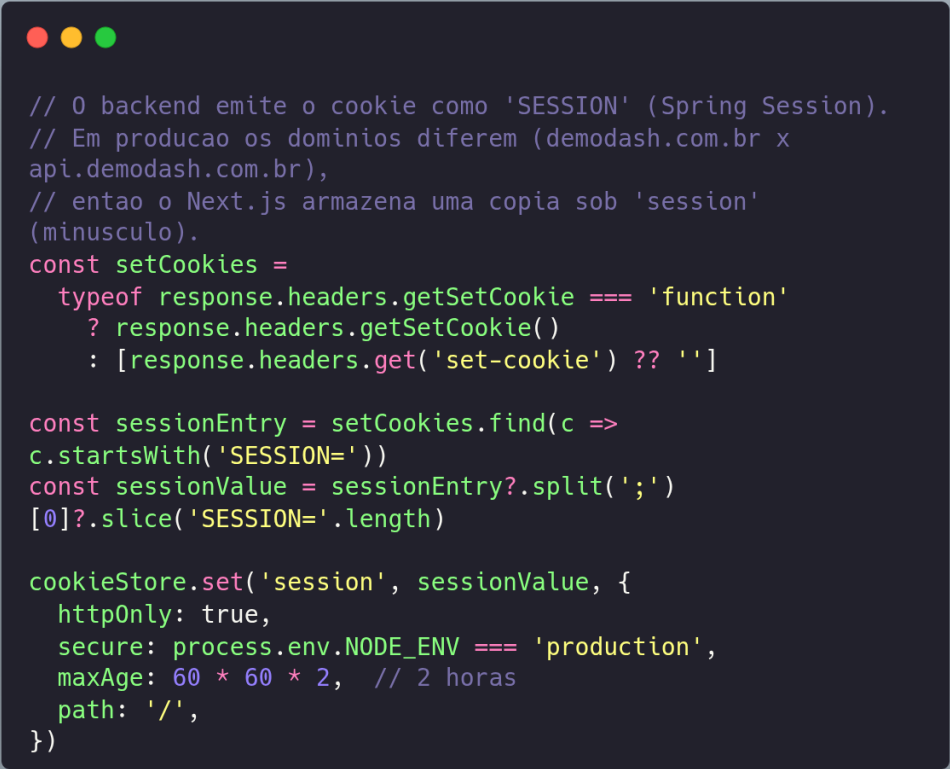
  if (response.status === 204) {
    return { success: true, data: undefined as T }
  }

  let body: unknown
  try {
    body = await response.json()
  } catch {
    return {
      success: false,
      error: new ApiError(
        'UNKNOWN_API_ERROR',
        `API retornou ${String(response.status)} sem corpo
JSON.`,
        response.status
      ).toJSON(),
    }
  }

  if (!response.ok) {
    if (isApiErrorResponse(body)) {
      const fieldErrors = body.details?.reduce<Record<string,
string>>((acc, { field, message }) => {
        acc[field] = message
        return acc
      },
      {})
    }
    return {
      success: false,
      error: new ApiError(
        body.error_code,
        body.message,
        body.status_code,
        body.user_action,
        fieldErrors
      ).toJSON(),
    }
  }
  return {
    success: false,
    error: new ApiError(
      'UNKNOWN_API_ERROR',
      `Erro ${String(response.status)}`,
      response.status
    ).toJSON(),
  }
}
return { success: true, data: body as T }
}

```

Figura 92 – Função serverFetch<T>() em src/lib/api/server-fetch.ts: leitura do cookie de sessão e encaminhamento ao backend.



```
// O backend emite o cookie como 'SESSION' (Spring Session).  
// Em producao os dominios diferem (demodash.com.br x  
api.demodash.com.br),  
// entao o Next.js armazena uma copia sob 'session'  
(minusculo).  
const setCookies =  
  typeof response.headers.getSetCookie === 'function'  
    ? response.headers.getSetCookie()  
    : [response.headers.get('set-cookie') ?? '']  
  
const sessionEntry = setCookies.find(c =>  
  c.startsWith('SESSION='))  
const sessionValue = sessionEntry?.split(';')  
[0]?.slice('SESSION='.length)  
  
cookieStore.set('session', sessionValue, {  
  httpOnly: true,  
  secure: process.env.NODE_ENV === 'production',  
  maxAge: 60 * 60 * 2, // 2 horas  
  path: '/',  
})
```

Figura 93 – Trecho de actions.ts: renomeação do cookie SESSION do backend para session no Next.js.

APÊNDICE C – ARTIGO NO FORMATO SBC

Este apêndice apresenta, no formato de artigo científico da Sociedade Brasileira de Computação (SBC), uma síntese deste Trabalho de Conclusão de Curso.

DeMoDash: Um sistema web de gerenciamento do Capítulo Florianópolis nº 076 da Ordem DeMolay

Nathan Sargon Werlich¹, Victor Gevaerd Luiz¹

¹Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

nathan.werlich@grad.ufsc.br, victor.gevaerd@grad.ufsc.br

Resumo. O Capítulo Florianópolis nº 076 da Ordem DeMolay enfrenta dificuldades na gestão de suas operações internas, decorrentes da dependência de processos manuais e documentos em papel para o controle financeiro, de frequência e de eventos, o que compromete a transparência e a continuidade administrativa entre gestões. Neste trabalho foram pesquisados sistemas existentes voltados à gestão de capítulos da Ordem DeMolay e de organizações paramaçônicas similares. Com base nessa análise, foi desenvolvido o “DeMoDash”, um sistema web no formato PWA com módulos de cadastro de membros, perfis de acesso, gestão financeira, controle de frequência, calendário de eventos e transparência. O sistema foi validado por meio de 130 testes automatizados, 12 casos de teste funcionais e integração contínua com cinco fluxos automatizados no GitHub Actions.

Abstract. The Florianópolis Chapter No. 076 of the Order of DeMolay faces difficulties in managing its internal operations due to its reliance on manual processes and paper documents for financial control, attendance, and event management, which compromises transparency and administrative continuity between administrations. This work researched existing systems aimed at managing DeMolay chapters and similar para-Masonic organizations. Based on this analysis, “DeMoDash” was developed, a Progressive Web App with modules for member registration, access profiles, financial management, attendance control, event calendar, and transparency. The system was validated through 130 automated tests, 12 functional test cases, and continuous integration with five automated workflows on GitHub Actions.

1. Introdução

A Ordem DeMolay é uma organização voltada para jovens de 12 a 21 anos, fundada em 1919 nos Estados Unidos, com uma longa história de contribuição ao desenvolvimento de cidadãos conscientes, empáticos e comprometidos com o bem-estar da sociedade [SCDB — Centro Nacional de Liderança DeMolay 2023]. Os Capítulos da Ordem DeMolay operam de forma independente, seguindo diretrizes estabelecidas pelo Supremo Conselho DeMolay Brasil (SCDB), o que oferece flexibilidade mas também apresenta desafios de gestão local.

O Capítulo Florianópolis nº 076 conduz suas operações administrativas — controle financeiro, registro de frequência em sessões e calendário de atividades — por meio

de processos manuais e documentos em papel. Essa dependência gera dificuldades de continuidade entre gestões, risco de perda de registros históricos e ausência de um canal centralizado de transparência para os membros.

Diante desse contexto, este trabalho propõe o **DeMoDash**, um sistema web no formato Progressive Web App (PWA) [Hume 2017] desenvolvido especificamente para o Capítulo Florianópolis nº 076. O sistema reúne seis módulos funcionais — membros, perfis de acesso, gestão financeira, controle de frequência, calendário de eventos e transparência — com o objetivo de digitalizar e centralizar os processos administrativos, reduzir a dependência de papel e garantir a continuidade das informações entre gestões.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 descreve a solução proposta, suas tecnologias e arquitetura; a Seção 4 apresenta a avaliação do sistema; e a Seção 5 conclui o trabalho.

2. Trabalhos Relacionados

Com o objetivo de identificar alternativas existentes para a gestão do Capítulo, realizou-se uma pesquisa em lojas de aplicativos (App Store e Google Play) e em sistemas web, adotando termos como “gerenciamento de Capítulo Ordem DeMolay” e “sistemas de gestão paramaçônicos”. Três soluções foram identificadas e avaliadas.

O **DeMolay Virtual** [Sistema DeMolay Virtual 2023] é um sistema web desenvolvido exclusivamente para Capítulos DeMolay, oferecendo módulos de gestão financeira, controle de frequência e relatório de transparência. No entanto, não suporta calendário de eventos e não disponibiliza interface mobile responsiva, limitando o acesso em dispositivos portáteis.

O **SisMaçom** [SisMaçom 2023] é um sistema para gestão de Lojas Maçônicas que inclui módulos financeiros, controle de frequência e calendário. Embora seja acessível em dispositivos móveis, não foi projetado para as especificidades da Ordem DeMolay, carecendo de compatibilidade com seus processos administrativos e relatório de transparência.

O **Maçonaria Virtual Mobile** [Maçonaria Virtual Mobile 2023] é um aplicativo móvel voltado a membros de lojas maçônicas, com suporte a controle de frequência e calendário de eventos. Não possui, contudo, módulo financeiro nem relatório de transparência.

A Tabela 1 sintetiza a comparação entre as soluções analisadas, demonstrando que nenhuma delas atende integralmente ao conjunto de requisitos levantados para o Capítulo Florianópolis nº 076, o que motivou o desenvolvimento do DeMoDash.

3. Solução Proposta

3.1. Tecnologias Utilizadas

O DeMoDash é composto por um frontend e um backend desacoplados, comunicando-se por uma API REST.

O **frontend** foi desenvolvido com React [Stefanov 2021] e TypeScript [Fain and Moiseev 2019], utilizando o framework *Next.js* para roteamento baseado em sistema de arquivos e renderização híbrida (React Server Components e ações de servidor). A aplicação é distribuída como PWA [Hume 2017], permitindo instalação em

Tabela 1. Comparativo de funcionalidades dos sistemas analisados.

Requisito	DeMolay Virtual	SisMaçom	MV Mobile	DeMoDash
Gestão financeira	Sim	Sim	Não	Sim
Controle de frequência	Sim	Sim	Sim	Sim
Calendário de eventos	Não	Sim	Sim	Sim
Controle de acesso	Sim	Sim	Sim	Sim
Rel. de transparência	Sim	Não	Não	Sim
Acessível web e mobile	Não	Sim	Sim	Sim
Compat. com DeMolay	Sim	Não	Não	Sim

dispositivos móveis e acesso offline parcial sem necessidade de publicação em lojas de aplicativos. O deploy do frontend é realizado na plataforma Vercel.

O **backend** foi implementado em Kotlin [Jemerov and Isakova 2017] com Spring Boot [Walls 2015], seguindo a Arquitetura Hexagonal [Cockburn 2005]. A persistência é gerenciada por JPA com PostgreSQL [Obe and Hsu 2017] e as migrações de esquema são versionadas com Liquibase. O backend é hospedado no Render e o banco de dados no Supabase. O repositório público do sistema está disponível no GitHub, e o sistema em produção pode ser acessado em <https://demodash.com.br>.

3.2. Arquitetura do Sistema

O backend adota a **Arquitetura Hexagonal** [Cockburn 2005], que organiza o sistema em torno de um núcleo de domínio isolado, comunicando-se com o mundo externo por meio de portas (*ports*) e adaptadores (*adapters*). Essa abordagem garante que as regras de negócio não dependam de detalhes de infraestrutura como banco de dados ou framework HTTP.

Em cada módulo do DeMoDash, a cadeia de dependências segue a seguinte direção: o controlador REST (adaptador de entrada) recebe a requisição, constrói um comando e delega ao caso de uso pela sua interface de porta de entrada; o caso de uso aplica as regras de negócio e interage com o repositório apenas pela porta de saída; o repositório (adaptador de saída) é o único componente que conhece JPA. A Figura 1 ilustra essa estrutura para o módulo de gestão financeira.

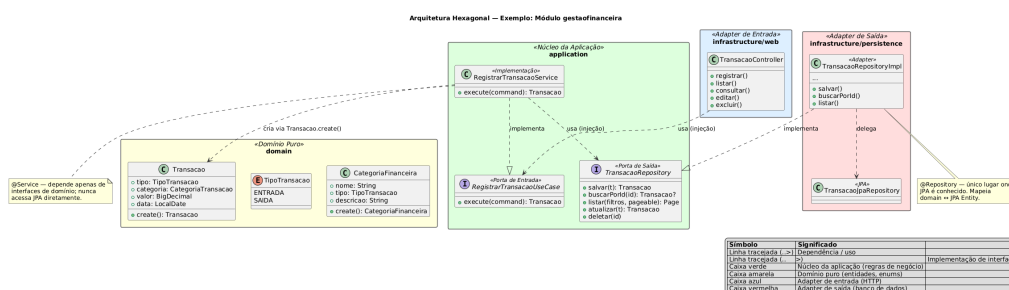


Figura 1. Arquitetura hexagonal do módulo de gestão financeira. Caixas azuis: adaptadores de entrada (HTTP); verdes: núcleo de aplicação; amarelas: domínio puro; vermelhas: adaptadores de saída (BD).

O **frontend** utiliza o Next.js com três estratégias complementares de busca de

dados: React Server Components para carregamentos iniciais de página; Server Actions para operações de escrita disparadas pelo usuário; e SWR para consultas reativas com revalidação automática. As permissões exibidas na interface são sincronizadas automaticamente com as do backend por meio de um arquivo gerado no processo de build.

3.3. Módulos e Funcionalidades

O DeMoDash é estruturado em seis módulos, cada um correspondendo a um conjunto de histórias de usuário definidas durante o levantamento de requisitos:

- **Membros:** cadastro, edição, visualização de perfil e inativação de membros. Suporta dois status oficiais DeMolay: *Ativo* (até 21 anos) e *Sênior* (21 anos ou mais), calculados automaticamente a partir da data de nascimento. O cadastro envia e-mail de ativação para que o novo membro defina sua senha.
- **Perfis de Acesso:** gerenciamento de perfis (conjuntos de permissões), com granularidade por operação (ex.: `FINANCEIRO:REGISTRAR_ENTRADA` e `FINANCEIRO:REGISTRAR_SAIDA` como permissões distintas). Perfis de sistema protegem as permissões administrativas críticas contra atribuição indevida.
- **Gestão Financeira:** registro de transações (entradas e saídas), categorização e geração de balanços periódicos. O *dashboard* financeiro exhibe resumo mensal e fluxo de caixa.
- **Controle de Frequência:** registro de presenças em sessões, com relatório individual por membro e visão consolidada por período.
- **Calendário de Eventos:** criação, edição e exclusão de eventos com visualizações em formato mensal, semanal e lista.
- **Transparência:** relatórios financeiros e de frequência de acesso público, com gráficos de balanço mensal, fluxo de caixa semestral e rosca por categoria.

A Figura 2 ilustra o dashboard de gestão financeira, exibindo o resumo mensal e os controles de registro de transações.

4. Avaliação

A avaliação do DeMoDash compreendeu validação junto aos usuários finais, testes automatizados, integração contínua e testes funcionais manuais.

A **validação** ocorreu de forma incremental ao longo do desenvolvimento: à medida que cada módulo era implementado, as funcionalidades eram apresentadas aos membros do Capítulo DeMolay, que atuaram como fonte de requisitos e de *feedback*. Esse contato contínuo permitiu ajustar fluxos e interfaces à rotina administrativa real do Capítulo [Pressman and Maxim 2020].

Os **testes automatizados** cobrem backend e frontend em dois níveis. No backend, 59 arquivos de testes de unidade verificam as regras de negócio com JUnit 5 e Mockito, e 37 arquivos de testes de integração exercitam o sistema de ponta a ponta com Testcontainers, provisionando um contêiner PostgreSQL real e aplicando as migrações reais do Liquibase. No frontend, 34 arquivos de testes de unidade com Jest verificam *server actions*, *schemas* Zod e funções utilitárias. O total é de **130 arquivos de teste**.

A **integração contínua** é realizada por cinco fluxos automatizados no GitHub Actions, executados a cada *pull request*: Backend CI (build, ktlint, testes unitários e de

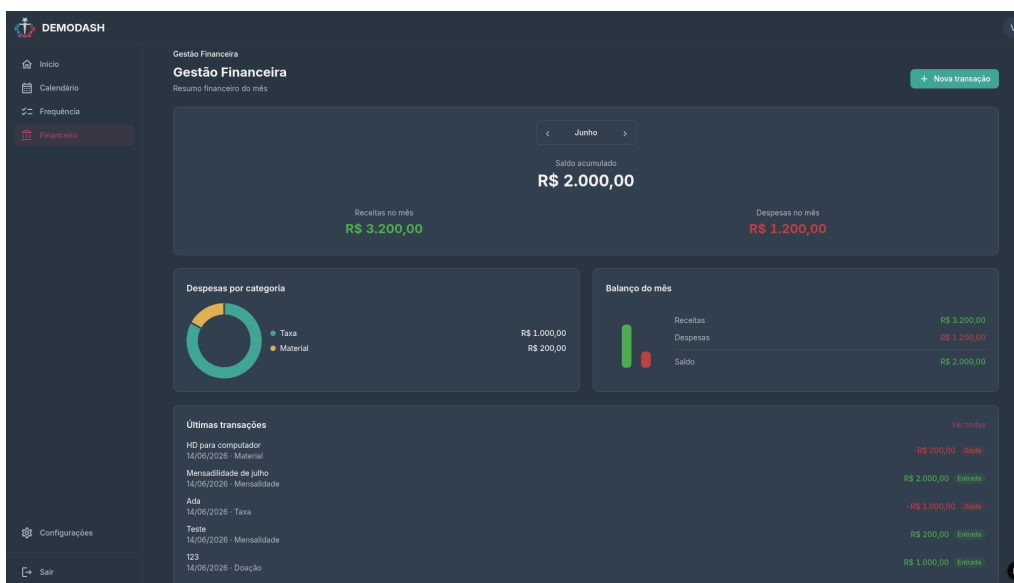


Figura 2. Dashboard do módulo de gestão financeira do DeMoDash, com resumo mensal de entradas, saídas e saldo.

integração); Frontend CI (Prettier, ESLint zero-aviso, Jest e build de produção); Commit Lint (Conventional Commits); Docker and Config Validation (Hadolint e compose.yaml); e Lint GitHub Workflows (actionlint).

Complementando, foram conduzidos **12 casos de teste funcionais** manuais, cobrindo os fluxos principais de autenticação, membros, financeiro, frequência, calendário e transparência. Todos os 12 casos obtiveram resultado igual ao esperado.

O sistema encontra-se disponibilizado em ambiente de produção (<https://demodash.com.br>) e homologação (<https://staging.demodash.com.br>), com o código-fonte publicado no GitHub em (<https://github.com/victorgevaerd/demodash>) e no CodigosUFSC em (<https://codigos.ufsc.br/nathan/demodash>).

5. Conclusão

O DeMoDash foi desenvolvido para modernizar a gestão administrativa do Capítulo Florianópolis nº 076 da Ordem DeMolay, substituindo processos manuais e documentos em papel por um sistema web centralizado, acessível em navegadores e instalável como PWA em dispositivos móveis. O sistema cumpriu todos os objetivos definidos, entregando os seis módulos planejados e validando suas funcionalidades por meio de 130 testes automatizados, 12 casos de teste funcionais e integração contínua.

Como trabalhos futuros, destacam-se: a **expansão para outros Capítulos** da Ordem DeMolay, aproveitando a arquitetura modular existente; o suporte a **notificações push** via infraestrutura PWA; a geração de **relatórios avançados** com exportação para PDF e planilha; e o **registro de presença por QR Code**, eliminando a necessidade de um operador dedicado durante as sessões.

Referências

- Cockburn, A. (2005). Hexagonal architecture. <https://alistair.cockburn.us/hexagonal-architecture/>. Acesso em: 14 jun. 2026.
- Fain, Y. and Moiseev, A. (2019). *TypeScript Quickly*. Manning Publications.
- Hume, D. A. (2017). *Progressive Web Apps*. Manning Publications.
- Jemerov, D. and Isakova, S. (2017). *Kotlin in Action*. Simon and Schuster.
- Maçonaria Virtual Mobile (2023). Maçonaria virtual. Maçonaria Virtual. Acesso em: 14 nov. 2023.
- Obe, R. O. and Hsu, L. S. (2017). *PostgreSQL: up and running: a practical guide to the advanced open source database*. O'Reilly Media, Inc.
- Pressman, R. S. and Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 9 edition.
- SCDB — Centro Nacional de Liderança DeMolay (2023). O que é a ordem demolay? DeMolay Brasil. Acesso em: 13 abr. 2023.
- SisMaçom (2023). Sismaçom - sistema para gestão de loja maçônica. Sismaçom. Acesso em: 14 nov. 2023.
- Sistema DeMolay Virtual (2023). Demolay virtual — crie o site de seu capítulo demolay em 3 minutos. DeMolay Virtual. Acesso em: 14 nov. 2023.
- Stefanov, S. (2021). *React: Up & Running: Building Web Applications*. O'Reilly Media, Inc.
- Walls, C. (2015). *Spring Boot in Action*. Manning Publications.

APÊNDICE D – CÓDIGO FONTE

<https://codigos.ufsc.br/nathan/demodash>