UNIVERSIDADE PAULISTA – UNIP
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS (ICET)

Kauã Henrique Cogo Pedro

Gabriel Zacarias Andrade

Ithalo Pulcini Santos

Marcos Eduardo Gonçalves

Felipe de Araújo Ferreira

TECNOLOGIA A SERVIÇO DA SOLIDARIEDADE: UM ESTUDO DE CASO SOBRE O DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO DE DOAÇÕES E VOLUNTÁRIOS PARA ENTIDADES DE AUXÍLIO SOCIAL

Ribeirão Preto
2025

Kauã Henrique Cogo Pedro
Gabriel Zacarias Andrade
Ithalo Pulcini Santos
Marcos Eduardo Gonçalves
Felipe de Araújo Ferreira

TECNOLOGIA A SERVIÇO DA SOLIDARIEDADE: UM ESTUDO DE CASO SOBRE O DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO DE DOAÇÕES E VOLUNTÁRIOS PARA ENTIDADES DE AUXÍLIO SOCIAL

Monografia submetida ao Curso de Ciência da Computação da Universidade Paulista, para a obtenção do Título de Bacharel em Ciência da Computação.

Orientadores:
Prof. Dr. Avelino Palma Pimenta Júnior
Prof.ª Dr.ª Eliana Leão do Prado Battaglion

Ribeirão Preto
2025

Kauã Henrique Cogo Pedro

Gabriel Zacarias Andrade

Ithalo Pulcini Santos

Marcos Eduardo Gonçalves

Felipe de Araújo Ferreira

TECNOLOGIA A SERVIÇO DA SOLIDARIEDADE: UM ESTUDO DE CASO SOBRE O DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO DE DOAÇÕES E VOLUNTÁRIOS PARA ENTIDADES DE AUXÍLIO SOCIAL

Monografia submetida ao Curso de Ciência da Computação da Universidade Paulista, para a obtenção do Título de Bacharel em Ciência da Computação.

Aprovado em: 04/12/2025

Média Final - MF: 10,0 (dez)

BANCA EXAMINADORA

gov.br Documento assinado digitalmente
ELIANA LEAO DO PRADO BATTAGLION
Data: 15/12/2025 18:13:06-0300
Verifique em https://validar.iti.gov.br

Prof.ª Dr.ª Eliana Leão do Prado Battaglion
Universidade Paulista – UNIP

Dedicamos este trabalho à Pastoral São Francisco de Assis, aos voluntários e às famílias atendidas, que inspiraram a criação deste projeto.
Também dedicamos este trabalho às nossas famílias, que nos ofereceram incentivo, paciência e motivação para seguir adiante.

AGRADECIMENTOS

Agradecemos à Universidade Paulista e todos os professores que fizeram parte da nossa trajetória acadêmica, e à Pastoral São Francisco de Assis pelo apoio e colaboração durante o desenvolvimento deste projeto.

“Comece fazendo o que é necessário; depois o que é possível; e, de repente, você estará fazendo o impossível.”

(São Francisco de Assis)

RESUMO

COGO PEDRO, Kauã Henrique; ANDRADE, Gabriel Zacarias; SANTOS, Ithalo Pulcini; GONÇALVES, Marcos Eduardo; FERREIRA, Felipe de Araújo. **Tecnologia a Serviço da Solidariedade: Um Estudo de Caso Sobre o Desenvolvimento de um Sistema de Gerenciamento de Doações e Voluntários para Entidades de Auxílio Social.** 2025. 60 f. Trabalho de Conclusão de Curso (Curso de Ciência da Computação) – Instituto de Ciências Exatas e Tecnologia, Universidade de Paulista, Campus Vargas, Ribeirão Preto, 2025.

Este trabalho apresenta o desenvolvimento de um sistema web feito para a Pastoral São Francisco de Assis, da Paróquia Santa Luzia de Cravinhos, criado com o intuito de auxiliar no gerenciamento de famílias atendidas, voluntários, estoque e cestas básicas, procurando também modernizar processos e aumentar a eficiência operacional. Para o início do projeto, adotou-se uma pesquisa aplicada, com levantamento de requisitos por meio de reuniões e entendimento do fluxo de trabalho da pastoral; em seguida, foi feita uma modelagem dos dados e por fim a criação da aplicação. A implementação prática utilizou Laravel (backend, padrão MVC e Eloquent ORM), React (frontend) e PostgreSQL (banco de dados), com controle de versão por meio do Git e gestão ágil baseada em Scrum. O sistema oferece módulos de cadastro e gestão de famílias, agentes e cestas; distribuição de entregas entre agentes; controle de estoque; painel administrativo com indicadores; e integração com o aplicativo de mensagens Telegram para notificação dos agentes voluntários. Foram implementados mecanismos de autenticação e perfis de acesso (administrador e agente), com base em boas práticas de segurança e atenção à LGPD. Os resultados demonstram que a solução substitui controles manuais por um ambiente centralizado e confiável, reduzindo inconsistências, melhorando a rastreabilidade das doações e apoiando decisões com informações atualizadas. Conclui-se que os objetivos propostos foram alcançados e que a plataforma é escalável para evoluções futuras, como novos relatórios, automações e captação online de doações.

Palavras-chave: Sistemas web. Laravel. Gestão de doações. React. Assistência social.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A insegurança alimentar é um grave problema social que atinge milhões de pessoas ao redor de todo mundo, e no Brasil esse cenário infelizmente não é diferente.

Ainda que a alimentação seja um direito humano essencial, como previsto no artigo 6º da Constituição Federal de 1988 (BRASIL, 1988), boa parte da população brasileira enfrenta esse desafio diariamente. Segundo o último relatório “O Estado da Segurança Alimentar e da Nutrição no Mundo” (SOFI), que foi divulgado pela ONU em 2024, 8,4 milhões de brasileiros enfrentaram fome nos últimos dois anos (FAO et al., 2024).

Nesse contexto, entende-se que é papel do Estado garantir apoio aos que sofrem desse mal, porém a sociedade também assume responsabilidades e sempre procura formas de ajudar quem mais necessita. Destaca-se que no Brasil, a concessão de alimentos é um dos grandes legados históricos da assistência social, vista desde as formas mais primitivas da prestação de auxílios e caridade (BOVOLENTA, 2017).

Em Cravinhos, na Paróquia Santa Luzia de religião católica, existe uma pastoral que tem o nome “São Francisco de Assis para Promoção Humana”, pastoral essa que é o braço de assistência social da igreja, atuando majoritariamente com a entrega de cestas básicas, e que também é o foco total deste projeto. Temos como objetivo, por meio da tecnologia e do desenvolvimento de software, modernizar e inovar como esse trabalho é feito, para que cada vez mais pessoas possam ser beneficiadas.

O objetivo final é que todas as atividades feitas na pastoral estejam concentradas em um sistema Web, que organiza tudo de forma mais prática, ágil e confiável. Também tivemos como objetivo a disponibilização painéis com os dados armazenados, que tornam as tomadas de decisão dentro da pastoral mais lógicas e coerentes.

A principal motivação para a escolha desse tema foi que hoje a adoção de tecnologia nos processos não é mais um diferencial, e sim uma necessidade. Muitas das atividades desenvolvidas na Paróquia ainda são feitas de forma arcaica, com controles e anotações manuais, e apesar da relevância de todas as atividades feitas, muitas pastorais enfrentam desafios significativos na sua organização, especialmente

quando se fala em controle de estoque, registro de famílias atendidas e organização de voluntários.

Dessa forma, a falta de um planejamento bem definido e das ferramentas adequadas nos leva ao dado de que apenas 30% das instituições dessa vertente possuem um plano de gestão. Quando não há esse cuidado, a pastoral corre o risco de fazer muitas ações, porém ter poucos resultados (SILVA; COSTA, 2007).

Em suma, além de contribuir diretamente para a melhoria do serviço prestado pela Pastoral São Francisco de Assis, este projeto demonstra como a tecnologia e o desenvolvimento de software podem ser utilizados de forma acessível para gerar impacto positivo em comunidades carentes, valorizando o papel da informática como uma ferramenta capaz de complementar perfeitamente projetos de assistência social, e torná-los ainda melhores.

2 OBJETIVOS

2.1 Geral

O principal objetivo do projeto consistiu na criação de um sistema Web que ajude no gerenciamento e controle das famílias assistidas pela Pastoral São Francisco de Assis de Cravinhos, bem como no controle do estoque e das cestas básicas em si, tornando todo processo mais moderno e eficiente.

2.2 Específicos

Desenvolver um sistema web para cadastro e gerenciamento de famílias, agentes de entrega e cestas básicas, de acordo com o solicitado pelos membros da Pastoral.

Implementar um sistema que permita a distribuição de entregas entre os agentes disponíveis.

Elaborar uma integração com serviços de comunicação (Telegram) para que os detalhes das entregas sejam enviados aos responsáveis

Criar um painel administrativo para que o responsável pela Pastoral possa acompanhar e organizar as entregas.

Disponibilizar um controle de estoque para acompanhar a quantidade de produtos disponíveis, bem como a quantidade de cestas básicas montadas.

Garantir que o sistema seja seguro, com autenticação para administradores e agentes, restringindo acessos segundo as regras estabelecidas na Pastoral.

3 METODOLOGIA

Nessa seção, apresentamos e descrevemos a metodologia utilizada no projeto, com ênfase nas técnicas e ferramentas utilizadas, desde a concepção do projeto, passando pelo controle durante a fase de desenvolvimento, até a finalização. A metodologia consiste em práticas sistemáticas e racionais que auferem segurança ao nosso projeto, garantindo que tudo que for obtido como resultado no projeto é um conhecimento válido (LAKATOS; MARCONI, 2010), sendo assim um tópico fundamental.

3.1 Tipo de pesquisa

Dada a natureza do projeto, que é envolvido diretamente com as causas sociais no dia a dia, utilizou-se uma linha de pesquisa aplicada, visando gerar conhecimento que pudesse ser aplicado diretamente, na prática, e com uma maneira fácil de mensurar os resultados, uma vez que no momento que o projeto entrasse em produção, já poderíamos observar o ganho de tempo e produtividade dentro da pastoral.

Diferente da pesquisa puramente teórica, a pesquisa aplicada foca na resolução de problemas práticos. Ela se orienta pela aplicação dos resultados, visando trazer benefícios tangíveis e efetivos para a comunidade ou para o próprio contexto estudado (GIL, 2007). É justamente essa característica prática e, por vezes interativa, que a diferencia da pesquisa básica, pois tem como seu objetivo transformar uma realidade identificada (MARCONI; LAKATOS, 2004).

Seguindo essa lógica, a pesquisa aplicada está diretamente conectada com o desenvolvimento de soluções funcionais, destinadas a aprimorar processos ou contextos já existentes (PRODANOV; FREITAS, 2013).

Dessa forma, ao nos propormos para desenvolver um sistema informatizado de gerenciamento de atividades dessas entidades sociais, nossa investigação se alinha aos regulamentos da pesquisa aplicada. O objetivo não é apenas teórico, mas sim, a criação de uma ferramenta que promova uma mudança positiva e uma melhoria real no dia a dia dessas organizações.

3.2 Procedimentos metodológicos

Após a definição do tipo de pesquisa que desenvolvemos, focando em uma abordagem muito mais prática e aplicada, as próximas etapas foram necessárias para estruturar o projeto e garantir um bom gerenciamento durante cada etapa a seguir.

3.2.1 Levantamento de requisitos

Juntamente a Pastoral São Francisco de Assis, foram realizadas reuniões e diversas entrevistas informais, tanto com o coordenador quanto com a equipe em si.

Por meio desses encontros, pudemos observar diretamente como é o cotidiano da pastoral, englobando o contato com as famílias carentes, como o grupo organiza-

se para gerenciar o estoque e como é feito hoje todo lançamento dos controles manuais. Também foi feita uma análise de como esses registros são armazenados hoje, e qual seria a melhor maneira de converter o que é feito no papel para software.

Com o decorrer das reuniões e entrevistas, foi possível identificar as principais necessidades, tais como o controle das famílias assistidas, o gerenciamento do estoque, o registro de entregas e a emissão de relatórios gerenciais. Também identificou-se quais eram as maiores “dores” da Pastoral, como as dificuldades para gerenciar e vincular os voluntários às entregas de sua responsabilidade.

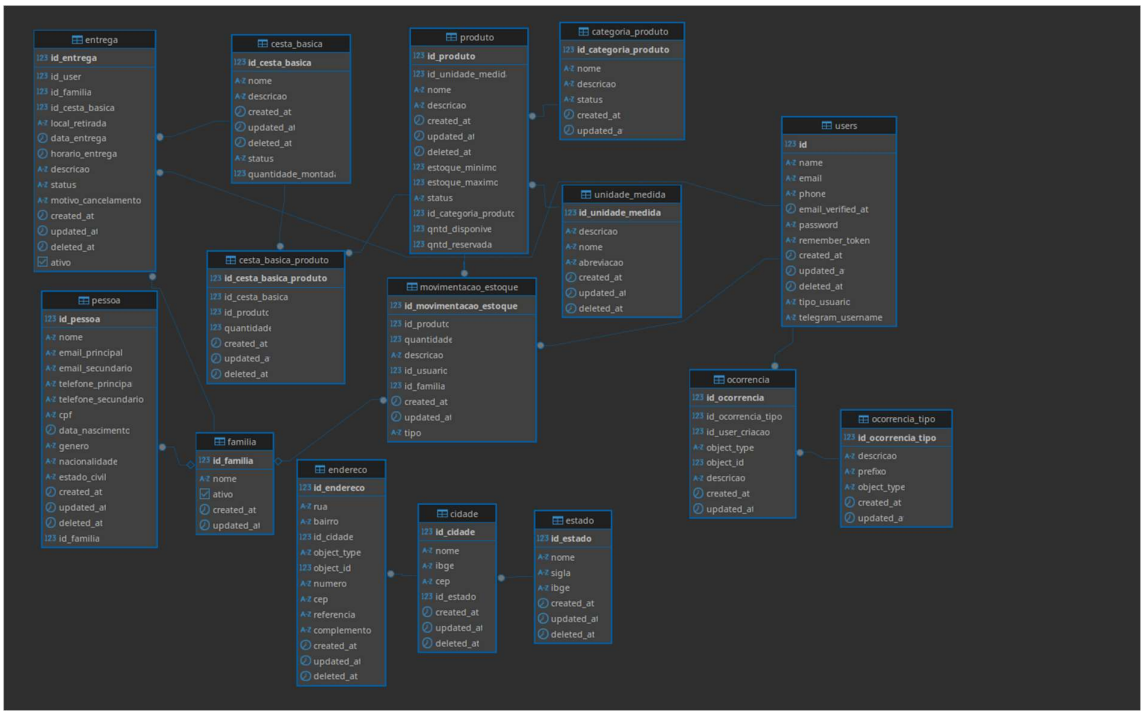
Por fim, apresentamos estes requisitos (funcionais e não funcionais) para a coordenação, e seguimos com o desenvolvimento a partir do momento que se obteve a aprovação.

3.2.2 Modelagem do sistema

Continuando os procedimentos metódicos, seguimos para a etapa de modelagem dos dados. A princípio, fizemos a modelagem do banco de dados para apresentar como as informações seriam armazenadas e distribuídas dentro do sistema.

Utilizamos o modelo lógico, o qual é o modelo de dados que representa a estrutura do banco no nível visto pelo usuário do SGBD (HEUSER, 1998). Como o modelo lógico é diretamente dependente do SGBD, já o fizemos da maneira aceita pelo PostgreSQL, o sistema usado no projeto. O diagrama gerado pode ser visto abaixo, na Figura 1, tendo sido feito com auxílio do software *DBeaver*:

Figura 1 - Diagrama do banco de dados



Fonte: Elaborado pelos autores (2025).

Além do diagrama, demonstrando a estrutura física das tabelas e campos, dedicamos o Quadro 1 para descrever o papel das principais tabelas dentro da aplicação:

Quadro 1 - Principais tabelas e suas funcionalidades

TABELA	FUNCIONALIDADE
users	responsável por armazenar informações de autenticação e perfil dos usuários
familia	representa o núcleo familiar atendido pela pastoral
pessoa	armazena os dados pessoais dos membros das famílias
cesta_basica e cesta_basica_produto	gerenciam os tipos de cestas e os produtos que as compõem
produto, categoria_produto e unidade_medida	formam o módulo de controle de estoque e inventário
entrega	registra as entregas realizadas, vinculando famílias, voluntários e cestas básicas
ocorrencia e ocorrencia_tipo	registram alterações e eventos importantes no sistema
movimentacao_estoque	controla as entradas e saídas de produtos, garantindo rastreabilidade das doações e saídas por entrega
telegram_data	organiza as informações para a integração da aplicação com o Telegram

Fonte: Elaborado pelos autores (2025).

Todas as tabelas possuem chaves primárias do tipo bigseria e restrições de integridade como foreign keys e check constraints, o que usamos para garantir a consistência dos dados e o relacionamento entre as diferentes tabelas.

Para a criação de todo o banco de dados, utilizamos a estrutura das *Migrations*, fornecida pelo Laravel por meio do seu sistema ORM. Na seção 4.6.4 essa tecnologia foi devidamente detalhada.

3.3 Escolhas das tecnologias

No processo do desenvolvimento precisou-se tomar algumas decisões sobre o ambiente de desenvolvimento e a arquitetura da aplicação, desde a escolha do sistema operacional até as tecnologias envolvidas no projeto. Além da parte do desenvolvimento em si, também foram usadas ferramentas de controle para que pudéssemos colaborar e manter um desenvolvimento coerente mesmo com várias ações de várias pessoas.

Além da parte técnica, no processo de escolha também foi levado em conta a confiabilidade e a aderência ao projeto que cada tecnologia possuía. Também levamos em conta o conhecimento da equipe e a disponibilidade de materiais e documentação, se necessário.

Dessa forma, apresentamos abaixo o que foi utilizado:

3.3.1 Debian

O Debian é um sistema operacional baseado em Linux de livre distribuição, criado por Ian Murdock. Ele é de simples utilização visando atender desde usuários iniciantes até administradores de servidores, um sistema seguro e estável (HERTZOG, 2013).

Além de ser um sistema robusto, o Debian é compatível com os principais servidores Web e bancos de dados relacionais. Outro ponto forte que justificou sua escolha é a flexibilidade. Uma vez que se trata de um S.O *open source*, seu código-fonte é disponibilizado publicamente para que qualquer pessoa possa usá-lo, modificá-lo e distribuí-lo.

Dadas essas características, o Debian foi o sistema operacional escolhido para servir como base da aplicação.

3.3.2 Backend

No *Backend*, onde reside toda a lógica e regras de negócio da aplicação, optamos por trabalhar com PHP juntamente ao *framework* Laravel, uma combinação

consolidada e reconhecida devido sua alta produtividade para ambientes web, bem como ampla documentação.

Conforme a documentação oficial descreve, o Laravel também segue a arquitetura MVC (*Model–View–Controller*), que facilita a organização do projeto e o reaproveitamento do código. Dentre outras funcionalidades que esse *framework* inclue nativamente, podemos destacar o *middleware* de segurança, sistema de autenticação e o ORM Eloquent, que foram importantes na etapa de desenvolvimento (LARAVEL, 2025).

Dessa forma, tanto a linguagem PHP quanto o Laravel contribuem diretamente para a agilidade, segurança e padronização do projeto, o que é indispensável para um projeto desse porte, e que precisa principalmente ser confiável aos usuários.

Essas tecnologias foram melhor detalhadas conceitualmente na seção de revisão de literatura.

3.3.3 Frontend

Para o desenvolvimento do *frontend*, que consiste em toda parte visual que o usuário interage, optamos pela utilização do React, uma biblioteca JavaScript de código aberto mantida pela Meta. Sua capacidade de reaproveitamento de componentes foi um diferencial na escolha dessa tecnologia.

O React também utiliza o conceito de *Single Page Applications* (SPA), que nada mais é do que executar toda a aplicação em uma única página web, atualizando dinamicamente apenas os elementos necessários (SCOTT, 2015). Essa tecnologia torna a experiência do usuário mais fluida, com menor tempo de resposta entre as ações.

Além disso, como a documentação oficial descreve (REACT, 2025), o React possui integração nativa com APIs REST, o que facilita a comunicação entre o frontend e o backend construído em Laravel.

3.3.4 Integração com Telegram

A integração com o Telegram foi implementada utilizando a API oficial de bots disponibilizada para desenvolvedores.

Para isso, o backend em Laravel realiza requisições aos endpoints do Telegram a partir do token do bot configurado no ambiente da aplicação. Os voluntários da Pastoral devem vincular suas contas ao sistema enviando uma mensagem ao bot

(*"/register")*, e essas mensagens são obtidas pelo método *getUpdates*, que recupera os *chat_id* e nomes de usuário registrados no Telegram. Posteriormente, esses dados são então associados aos usuários do sistema e armazenados na tabela telegram_data, o que nos permitirá identificar qual voluntário deve receber as notificações durante a execução das rotinas do projeto.

Quando uma entrega é agendada no sistema, o controller responsável monta a mensagem com o endereço, data e observações da entrega e a envia diretamente ao *chat_id* correspondente.

3.3.5 Git

Desde o início do desenvolvimento do sistema, optou-se pela ferramenta Git como sistema de controle de versão, devido à sua fama e conhecimento geral, e o uso frequente no mercado de trabalho. De acordo com Chacon e Straub (2014), o Git armazena todo o histórico de alterações do código em registros chamados *commits*, os quais funcionam como "pontos de restauração" que podem ser revertidos ou reaplicados conforme a necessidade. Além disso, o uso de branches — "ramificações" isoladas para desenvolvimento paralelo de funcionalidades — permite que novas features ou correções sejam implementadas sem impactar diretamente a linha principal de produção, podendo claro ser feito um trabalho em conjunto sem perder alterações e montando tanto um controle individual de cada usuário como um controle na branch principal (LOELIGER; McDONALD, 2012). Portanto, uma vez que o trabalho foi desenvolvido em grupo, o Git foi indispensável.

Essa abordagem usada para o versionamento também contribui para a segurança e rastreabilidade do projeto, pois cada *commit* pode ser descrito por meio de mensagens padronizadas, facilitando auditorias e revisões de código (SILVERMAN, 2013).

3.3.6 Scrum

Scrum é um framework ágil de gestão de projetos que estrutura o desenvolvimento em ciclos curtos, conhecidos como sprints, cujo prazo varia entre duas e quatro semanas. Cada sprint inicia a partir de um Product Backlog, ou seja, uma lista priorizada de requisitos e funcionalidades, onde a equipe pretende concluir até o final do ciclo.

No Scrum, tudo o que se faz desde o planejamento até o resultado precisa estar sempre claro para toda a equipe e para quem recebe o produto. Quando informações como o que está pendente, o que foi concluído e em que nível de qualidade o trabalho se encontra ficam escondidas ou mal representadas, decisões erradas viram peça-chave de atrasos, desperdícios e retrabalhos.

Por isso, é preciso checar com frequência tanto o andamento das tarefas quanto os artefatos gerados. O Scrum oferece momentos definidos para planejamento, reuniões diárias, revisão e retrospectiva em que a equipe avalia o que foi feito, identifica problemas e decide como seguir em frente. Mas inspecionar sem agir não adianta: sempre que algo sair do esperado, o time deve ajustar rapidamente seu jeito de trabalhar ou o próprio produto, mantendo o ritmo e evitando que pequenos desvios cresçam e comprometam o valor entregue.

> Scrum é baseado no empirismo e lean thinking. O empirismo afirma que o conhecimento vem da experiência e da tomada de decisões com base no que é observado. O lean thinking reduz o desperdício e se concentra no essencial. Scrum emprega uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar o risco. Scrum envolve grupos de pessoas que, coletivamente, possuem todas as habilidades e conhecimentos necessários para fazer o trabalho e compartilhar ou adquirir essas habilidades conforme necessário (Schwaber; Sutherland, 2020, p. 4).

4 REVISÃO DE LITERATURA

Para nos aprofundarmos no entendimento sobre o sistema proposto, alguns tópicos importantes precisam ser explicados, como o surgimento da ideia do projeto, o modelo do sistema e como ele foi executado na aplicação específica da nossa solução.

4.1 Ação sócio pastoral da igreja católica

A influência da Igreja Católica foi marcante para o desenvolvimento de ações direcionadas à assistência social no Brasil que remontam ao período colonial e teve seu apogeu com a ação dos jesuítas. Este espírito esteve presente até cerca da década de 1930, quando foi fundada a Escola de Serviço Social em São Paulo, em 1936 (Silva, 2010). A Igreja teve, assim, uma atuação fundamental na formulação de uma proposta profissional de ação social do país.

Essas práticas deram luz a centenas de programas e projetos voltados para o atendimento social, a recuperação da autoestima e da cidadania, inspirando diferentes

sujeitos coletivos que se se comprometiam com a defesa da vida e do direito humano vinculado ao bem viver na cidade.

4.2 AÇÕES SOCIAIS

As ações sociais promovidas por organizações sem fins lucrativos desempenham um papel crucial na transformação das comunidades atendidas, impactando positivamente a qualidade de vida dos beneficiários e promovendo o desenvolvimento social.

Um estudo publicado na Revista Pensamento Contemporâneo em Administração avaliou a percepção de 322 indivíduos de diferentes estados brasileiros sobre as iniciativas de uma organização não governamental, onde os resultados indicaram que as atividades desenvolvidas estavam alinhadas às necessidades locais, trazendo benefícios significativos para as comunidades (Anese, Costa, Coelho, 2018).

No entanto, o estudo também apontou a necessidade de melhorar a divulgação das ações e estabelecer mais parcerias público-privadas para ampliar o alcance e a eficácia dos projetos sociais.

4.3 Transformação digital no terceiro setor

Como é possível verificar em diversos setores, a tecnologia é uma tendência de crescimento constante, sendo um elemento essencial para o funcionamento de empresas e organizações. Todavia, no Terceiro Setor, que como descreve Mendes (1999), consiste em todas as organizações sem fins lucrativos que realizam ações de interesse público, essa tendência enfrenta maiores dificuldades para crescer em comparação com outras áreas.

Mesmo reconhecendo o grande potencial de transformação que os avanços digitais podem oferecer, o Terceiro Setor é um dos que mais apresentam barreiras para sua adoção. Conforme apontado no artigo *Digital Transformation in Non-Profit Organizations: Strategies, Challenges, and Successes* (Ahmed et al., 2024), as principais dificuldades envolvem limitações financeiras, que dificultam a aquisição de tecnologias e a capacitação de equipes, além de uma estrutura organizacional muitas vezes carente de recursos técnicos e com resistência cultural a recursos digitais.

Em contraste a esses obstáculos, assim como também é apontado no artigo citado anteriormente, quando as ferramentas de tecnologia da informação são

implementadas de maneira alinhada à missão da organização, os resultados de produtividade e eficácia são notáveis.

A escolha do tema deste projeto nasce justamente dessa necessidade de inserir tecnologia nas práticas sociais, tornando-as mais eficientes e sustentáveis. A Pastoral São Francisco de Assis, foco deste estudo, ainda realizava suas atividades de controle e registro de doações de forma manual, com fichas e anotações físicas. Assim, a proposta de um sistema web para gerenciamento de doações e voluntários vai ao encontro da modernização, agilidade e confiabilidade necessárias para a execução das atividades pastorais.

Dessa forma, o trabalho propõe não apenas o desenvolvimento de um sistema, mas também a aplicação prática da tecnologia como agente de inclusão e impacto social.

4.4 CRM (*Customer Relationship Management*)

O conceito de um sistema CRM (*Customer Relationship Management*) refere-se a um conjunto de práticas e ferramentas que visam centralizar informações e fortalecer o relacionamento entre uma organização e o seu público-alvo. É amplamente utilizado para compras, propostas, solicitações de serviço, relatórios e muitas mais funcionalidades que fazem parte da gestão do relacionamento da organização com os *stakeholders*. Embora esta ideia de sistema tenha surgido no contexto corporativo, estendemos sua aplicação para o contexto social, de forma que se encaixe no projeto, e possamos usufruir de seus benefícios.

A utilização de sistemas CRM pode gerar uma série de benefícios, como redução de trabalhos manuais com automatização de processos e principalmente a centralização das informações, eliminando a necessidade de se utilizar planilhas ou quaisquer outros tipos de anotações que ficam dispersas do sistema. Aplicando este conceito ao gerenciamento de produtos, voluntários, famílias e cestas básicas, o sistema pode oferecer uma visão mais detalhada de todos esses pontos de interesse, facilitando o contato com os usufruidores, otimizando a distribuição dos recursos e tornando o processo mais eficiente.

> Inserido nesse cenário, surge como diferencial competitivo um novo conceito representado pela sigla CRM - Customer Relationship Management. A princípio, fazendo-se a tradução para a língua portuguesa, CRM significa a

> Gestão do Relacionamento com o Cliente. Resumidamente, implicaria o processo organizacional de administrar o relacionamento com os clientes (Barretto, 2004, p.15).

4.5 SISTEMAS WEB

Um sistema web é uma aplicação ou software acessado por meio de navegadores de internet. Essas aplicações ou softwares operam a partir de um servidor remoto, que é disponível através da internet. Logo, a arquitetura utilizada é a de cliente-servidor, onde o cliente envia requisições para o servidor, que as processa e retorna as respostas.

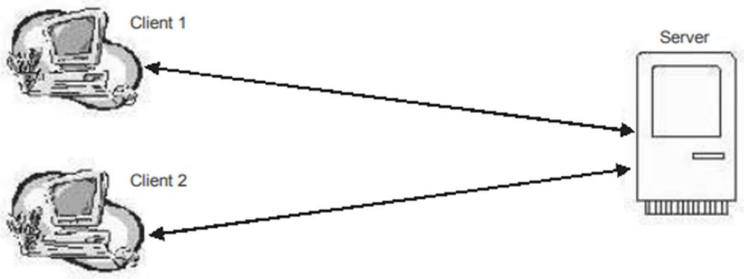
Yadav e Singh (2009), definem o conceito de cliente como qualquer processo que solicite algum serviço específico do servidor. O servidor, por sua vez, é o que provê todos os serviços requisitados pelo cliente. Eles podem coexistir na mesma máquina, ou então, o que é mais comum, existirem em diferentes computadores ligados via rede.

É possível existirem diferentes cenários de comunicação entre dispositivos em uma rede, como a interação entre um único cliente e um único servidor, a conexão de múltiplos clientes para um único servidor, ou até mesmo a comunicação simultânea de múltiplos clientes com múltiplos servidores. As Figuras 2, 3 e 4 exemplificam visualmente essa ideia:

Figura 2 - Um cliente, um servidor

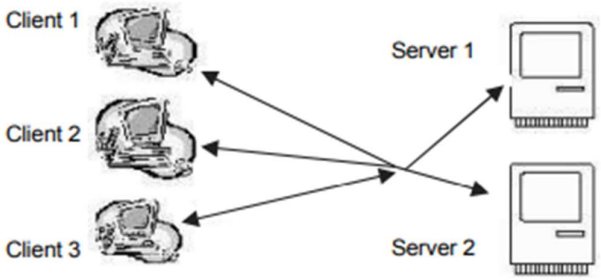
Fonte: YADAV; SINGH (2009, p. 8).

Figura 3 - Múltiplos clientes, um servidor



Fonte: YADAV; SINGH (2009, p. 8).

Figura 4 - Múltiplos clientes, múltiplos servidores



Fonte: YADAV; SINGH (2009, p. 8).

Sistemas distribuídos nessa estrutura têm grande destaque na acessibilidade e na independência de plataformas, já que podem ser utilizados a partir de qualquer dispositivo conectado à internet (Sommerville, 2019).

Portanto, devido a essa flexibilidade, esse modelo permite distribuição de responsabilidades e uma boa escalabilidade, o que torna esse tipo de sistema predominante em aplicações modernas. Outra vantagem dentro do contexto da nossa aplicação é a redução drástica de possíveis custos de implementação, já que tudo fica centralizado no servidor. Dessa forma, eliminamos a instalação e configuração individual do projeto para cada voluntário da pastoral, e também a necessidade de prestar suporte técnico presencialmente.

Em resumo, a escolha por desenvolver a aplicação como um sistema web vem principalmente da praticidade de configuração e utilização, alinhado aos padrões modernos de mercado. O acesso multiplataforma possibilita que os voluntários e gestores da Pastoral São Francisco de Assis utilizem a aplicação a partir de qualquer dispositivo conectado, aumentando diretamente a eficiência da operação.

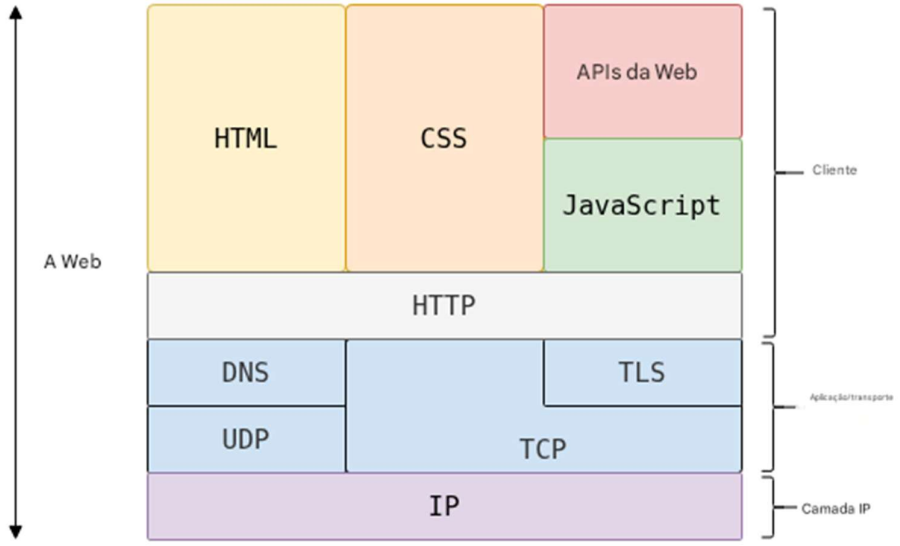
4.5.1 Requisições HTTP e comunicação cliente–servidor

Segundo a Mozilla Foundation (2025), a comunicação entre cliente e servidor em aplicações web ocorre em sua grande maioria por meio do protocolo HTTP (*Hypertext Transfer Protocol*): o cliente envia uma requisição contendo método, alvo (URI), cabeçalhos e corpo; o servidor, por sua vez, processa a solicitação e retorna uma resposta com código de status, cabeçalhos e, quando necessário, corpo.

Ainda podemos complementar este conceito, caracterizando o protocolo HTTP como sem estado (*stateless*). Ou seja, como o servidor não mantém estado nenhum entre duas requisições, cada requisição deve conter informações suficientes para ser compreendida de maneira individual.

A Figura 5 apresenta uma visão geral das camadas que compõem a arquitetura da Web, destacando o papel do protocolo HTTP na comunicação entre cliente e servidor, bem como sua interação com outros protocolos de rede e de transporte:

Figura 5 - Visão geral do protocolo HTTP



Fonte: Adaptado de Mozilla Foundation (2025).

Dentro do protocolo HTTP, ainda temos o que chamamos de métodos (ou verbos), que expressam a intenção da operação. Entre os mais utilizados, GET recupera representações de recursos; POST submete dados para processamento; PUT substitui a representação de um recurso; PATCH aplica modificações parciais; DELETE remove o recurso (Gourley; Totty, 2002).

Esses métodos são essenciais para a arquitetura que utilizamos na construção da aplicação, servindo como base para a comunicação entre as camadas e para a implementação da arquitetura RESTful, abordada na subseção seguinte.

4.5.2 APIs e arquitetura restful

Antes de nos aprofundarmos dentro da arquitetura RESTful, é preciso primeiro contextualizar o que é uma API:

> API é a sigla da expressão em inglês *Application Programming Interface*, que se refere a um conjunto de definições, regras e protocolos para construir e integrar aplicações e sistemas de software, permitindo que um programa se comunique com outro. Por exemplo, o Instagram utiliza a API do Facebook para publicar uma foto simultaneamente nas duas redes sociais. Basicamente, uma API é a interface entre clientes, que são aplicações que requisitam serviços, e o servidor, que provê os serviços requisitados (Torres, 2021, p. 18).

Conhecendo esse conceito, podemos partir para a arquitetura RESTful.

Para aplicações web, o padrão mais usado para a criação de APIs é o REST (*Representational State Transfer*), que foi proposto por Roy Fielding em 2000. O REST se baseia em conceitos que utilizam o protocolo HTTP como meio de comunicação, explorando seus métodos/verbos para representar as operações realizadas.

Segundo a Red Hat (2025), o REST consiste em um conjunto de restrições de arquiteturas aplicadas ao desenvolvimento de APIs, e não em um protocolo ou padrão específico. Ou seja, o REST pode ser implementado de diversas maneiras, ficando a cargo do desenvolvedor definir como será a melhor forma de fazê-lo.

Sua principal função é permitir a comunicação entre cliente e servidor por meio do protocolo HTTP, transferindo representações de recursos em formatos como JSON, HTML ou texto simples.

Uma API é considerada RESTful quando segue determinados princípios, entre os quais podemos citar: a arquitetura cliente-servidor, que separa responsabilidades entre interface e processamento; a comunicação sem estado (stateless), em que cada requisição é independente; a armazenabilidade em cache, que reduz a necessidade de solicitações repetidas; e uma interface uniforme, em que os recursos são identificados por URIs e descritos por representações autoexplicativas. (RED HAT, 2025).

Durante o desenvolvimento da aplicação, um dos principais focos foi manter toda a arquitetura de acordo com esse padrão, tornando o projeto mais limpo, entendível, e de fácil aplicação.

4.6 TECNOLOGIAS UTILIZADAS

Para o desenvolvimento do sistema web proposto, foi desenvolvido um software que utiliza diferentes tecnologias, cada uma com seu propósito. Temos a camada de *backend*, responsável pela operação e controle interno do sistema, gerenciando todas as chamadas de outra camada, o *frontend,* que é onde temos a parte de interação entre o usuário e o software, com as telas e visuais. Para armazenar todas as informações, utilizaremos conceitos de banco de dados relacionais.

Dedicamos os próximos tópicos a apresentar de maneira mais profunda as tecnologias que fizeram parte da elaboração do sistema.

4.6.1 Php

A linguagem PHP (Hypertext Preprocessor) foi criada em 1994 por Rasmus Lerdorf. Sua concepção inical era de um conjunto de scripts para automatizar páginas HTML dinâmicas. No entanto, ao longo dos anos, o PHP evoluiu para uma linguagem completa, com suporte a programação orientada a objetos e integração nativa com bancos de dados e SGBDs diversos, como PostgreSQL e MySQL (Lerdorf; Tatlock, 2013).

Nesse processo de amadurecimento da linguagem, o PHP consolidou-se como referência no desenvolvimento de aplicações web. Tatlock e Lerdorf (2013) também citam que em maio de 2012, os dados apontavam que 77.8% de dos websites cuja linguagem de desenvolvimento era conhecida, utilizavam PHP.

Converse e Park (2003) ainda destacam que o PHP é uma linguagem de programação voltada para a criação de scripts para a web do lado do servidor, que são interpretados à medida que as páginas são disponibilizadas para os usuários, o que chamamos de linguagens *server side*.

Destaca-se também o ecossistema que a linguagem construiu durante sua trajetória. Bibliotecas, frameworks e ferramentas complementam as funcionalidades

do PHP e consequentemente oferecem mais opções para o desenvolvedor utilizar durante a criação das aplicações.

Devido a esses trunfos, esta foi a linguagem escolhida para ser o alicerce de nosso software, pois os seus destaques complementam perfeitamente o que foi proposto, além de fornecer uma boa integração com as outras tecnologias utilizadas.

4.6.2 Laravel

Como optamos por utilizar a linguagem PHP para o *backend*, o Laravel será utilizado na aplicação como *framework* da linguagem, uma vez que ele fornece um ecossistema completo para desenvolvimento web, oferecendo todos os pacotes necessários para a construção e monitoração do software (Laravel, 2025).

O Laravel segue o padrão MVC (*model*, *view*, *controller*) em sua arquitetura, sendo este padrão um dos *Design Patterns* mais conhecidos da área de tecnologia, com conceitos que referenciam uma plataforma da década de 1970, a Smaltalk (Dall'Oglio, 2015). Nesse padrão, a aplicação é separada em três componentes, e dentro do ambiente Laravel, cada um desses componentes seguem uma estrutura de pastas padronizada.

Segundo Nogueira (2024) estes componentes são caracterizados da seguinte maneira:

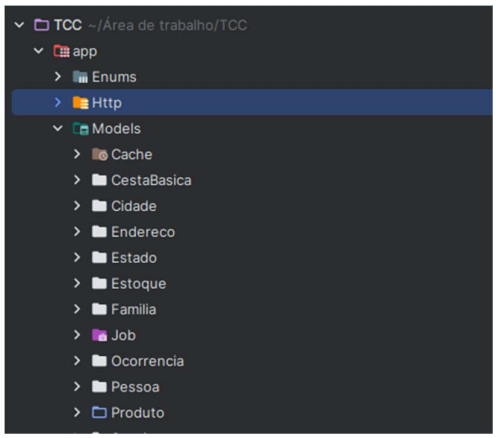
4.6.2.1 Model (Modelo)

O *model* é a camada responsável por representar as entidades e regras de negócio da aplicação. Ela também é responsável por interagir com o banco de dados para recuperar, armazenar e processar informações, garantindo a integridade e consistência dos dados.

Por norma, a maioria dos projetos utiliza um ORM (*Object-Relational Mapping*) para simplificar o mapeamento e a comunicação com o banco, podendo utilizar uma sintaxe mais próxima da orientação a objetos ao invés de comandos SQL diretamente.

Dentro do ambiente Laravel, todos os *models* ficam no caminho app/Models, conforme mostra a Figura 6 a seguir.

Figura 6 - Models dentro de uma aplicação Laravel



Fonte: Elaborado pelos autores (2025).

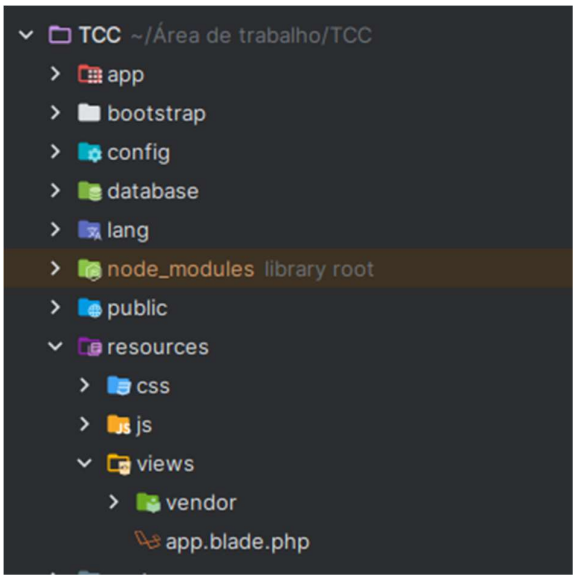
4.5.2.2 View (Visão)

A *view* representa a camada de apresentação do sistema, tudo que o usuário vê e interage. Sua principal responsabilidade é exibir os dados processados pelo *model*, de maneira legível ao usuário. É importante que essa camada seja bem trabalhada para ter um design amigável e com uma boa usabilidade, uma vez que é a porta de entrada da aplicação.

A qualidade das interfaces desenvolvidas aqui está ligada diretamente a satisfação final do usuário.

No ambiente Laravel, as *views* ficam no caminho resources/views, conforme a Figura 7 demonstra:

Figura 7 - Views dentro de uma aplicação Laravel



Fonte: Elaborada pelos autores (2025).

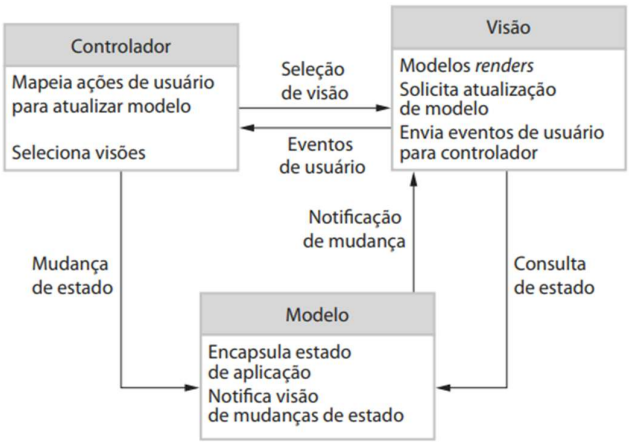
4.6.2.3 Controller (Controlador)

O *controller* atua como intermediário entre o *model* e a *view*, recebendo as requisições do usuário, processando os dados e responsável por definir qual resposta ou visualização será retornada. Ele também obtém os resultados do *model* e os encaminha para a View, garantindo que a interface apresente os dados corretamente.

No Laravel, os controladores também são estruturados para gerenciar rotas e aplicar regras de negócio, garantindo que cada parte da aplicação mantenha sua função específica.

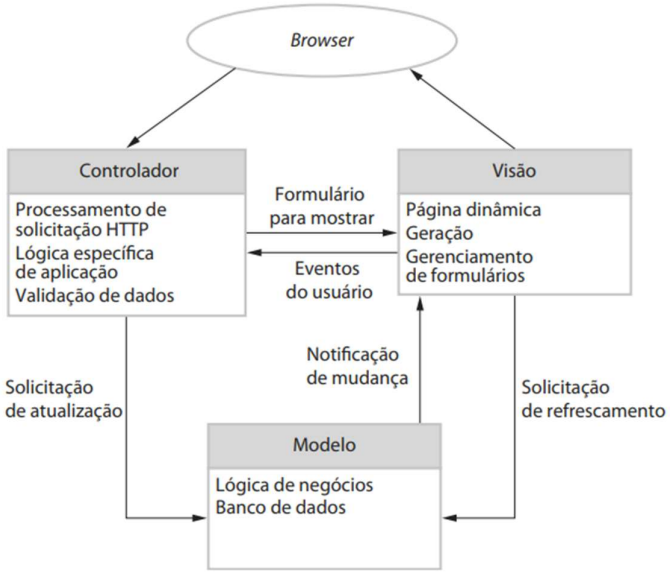
Na Figura 8 temos uma representação visual resumida do padrão arquitetural MVC, enquanto na Figura 9 temos um diagrama que representa esse padrão especificamente no contexto de aplicações web, conforme definição de Sommerville (2019), apresentadas a seguir.

Figura 8 - A organização do MVC



Fonte: Sommerville (2019, p.109).

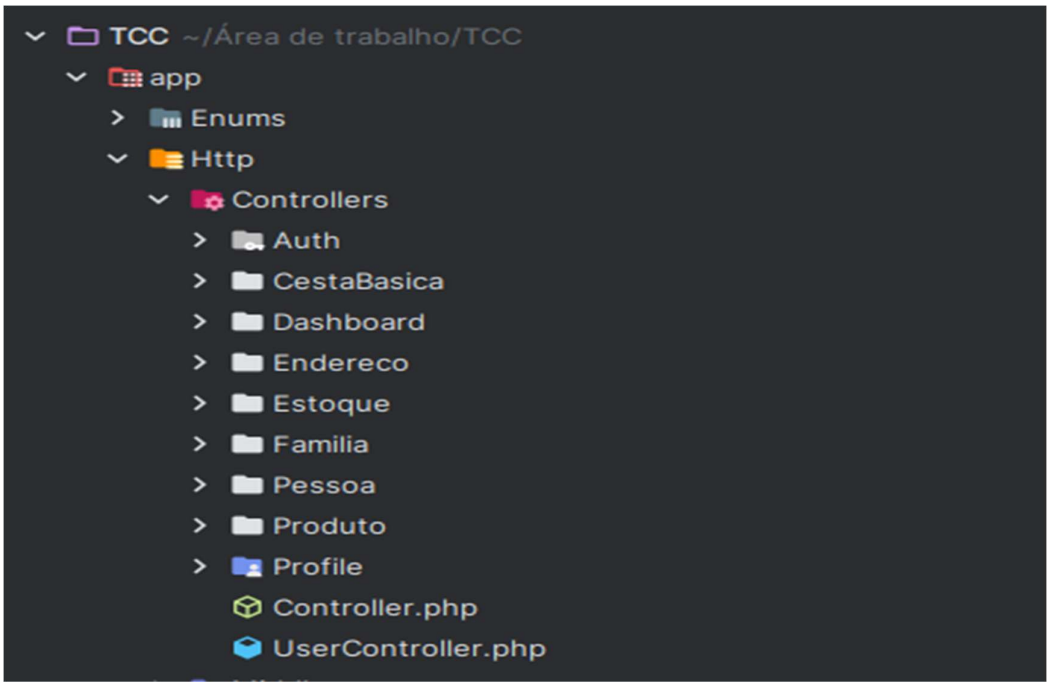
Figura 9 - Arquitetura de aplicações web usando o padrão MVC



Fonte: Sommerville (2019, p.110).

Assim como as camadas de *model* e *view*, os *controllers* também possuem um diretório predeterminado para uma aplicação Laravel, que fica em app/Http/Controllers, apresentada na Figura 10 a seguir.

Figura 10 - Controllers dentro de uma aplicação Laravel



Fonte: Elaborada pelos autores (2025).

4.6.3 PostgreSQL

Referente ao serviço de banco de dados, optamos pela utilização do PostgreSQL, que podemos definir da seguinte maneira:

> O PostgreSQL é um Sistema Gerenciador de Banco de Dados (SGBD Relacional), utilizado para armazenar informações de soluções de informática em todas as áreas de negócios existentes, bem como administrar o acesso a estas informações (Milani, 2008, p. 25).

Para qualquer software moderno, é indispensável haver um lugar confiável para armazenar as informações. Uma vez que nossa proposta foi realizar a migração de um controle de processos que eram feitos manualmente para algo digital, precisamos que haja a confiança do usuário em depositar estas informações no banco de dados.

De acordo com a documentação oficial (POSTGRESQL, 2025), esse SGBD segue estritamente às propriedades ACID (atomicidade, consistência, isolamento e durabilidade), um conceito que garante a total integridade das informações, ainda que ocorram situações de falha. Por meio do banco, também conseguimos usufruir de recursos de controle de concorrência multiversão (MVCC), que garante que vários usuários podem acessar e modificar os dados simultaneamente, sem conflitos.

Ainda temos diversos recursos disponíveis para transações e otimizações de consulta SQL, garantindo ganho de performance, além da segurança.

Dessa forma, o PostgreSQL foi utilizado para nos auxiliar nesta parte da aplicação, sendo um dos SGBD mais bem reconhecidos do mercado, e além de ser extremamente completo é gratuito e de código aberto.

4.6.4 Eloquent ORM

Com a definição que a aplicação usaria o ecossistema Laravel com o banco de dados PostgreSQL, o uso do Eloquent também foi de suma importância para o projeto.

Como define Fowler (2002), o mapeamento objeto-relacional (ORM) é uma técnica usada para mapear objetos da memória para tabelas de bancos de dados relacionais, estabelecendo uma ponte com o paradigma de orientação a objetos.

Nesse contexto, o Eloquent nada mais é do que a ferramenta ORM (*Object-Relational Mapping*) disponibilizada nativamente dentro do ambiente Laravel, permitindo a simplificação da interação entre nossa aplicação e o banco de dados.

Além do conceito de ORM, Fowler (2002) também define um padrão de implementação, chamado de *Active Record*. Nessa abordagem, cada objeto de domínio mapeia diretamente uma linha de uma tabela no banco de dados, encapsulando tanto os dados quanto a lógica de persistência, permitindo que os objetos interajam com o banco de dados sem a necessidade de SQL explícito.

Em outras palavras, conforme cita a documentação oficial do Eloquent (Laravel, 2025), cada *model* da nossa aplicação representa uma tabela do banco de dados, e cada instância dessa classe corresponde a um registro dessa tabela. Essa estrutura nos permite realizar qualquer operação de CRUD (criar, ler, atualizar e deletar registros) de maneira muito mais limpa e intuitiva.

Além desse nível de abstração, o Eloquent ainda fornece diversos outros recursos no mapeamento com o banco de dados, entre os quais podemos destacar:

Relacionamentos entre entidades (*hasOne*, *hasMany*, *belongsTo*, *belongsToMany*), que permitem a criação de vínculos entre tabelas de forma simples e sem necessidade de *joins* explícitos;

Migrations e *Seeders*, que facilitam o versionamento, a reprodução do esquema de banco de dados em diferentes ambientes e a população de registros nas tabelas;

Soft Deletes, funcionalidade que permite excluir registros logicamente, mantendo o histórico no banco de dados.

Em suma, a utilização desse ORM nos proporcionou uma forma consistente e organizada de realizar todo o tratamento de dados.

4.6.5 React

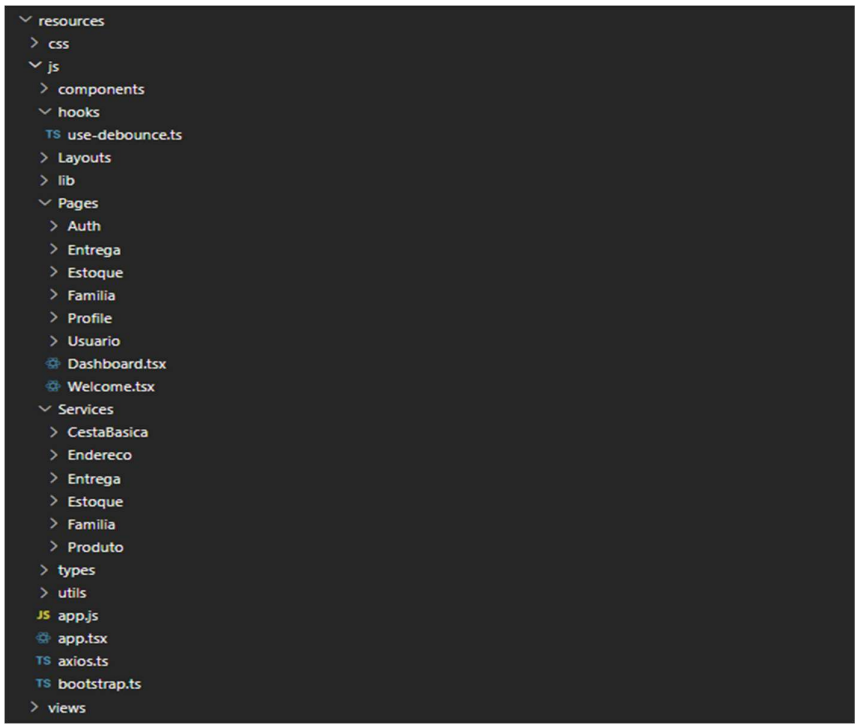
Todas as telas e visuais que o usuário interage na aplicação foram construídos com a biblioteca React. Portanto, é esta tecnologia que tem a responsabilidade de funcionar como a porta de entrada para as chamadas que serão feitas ao backend.

Segundo Gackenheimer (2015), o React é um *framework* da linguagem de programação JavaScript, e sua proposta é possibilitar o desenvolvimento de interfaces escaláveis, modulares e de fácil manutenção. O autor ainda destaca que o React introduz um modelo baseado na reutilização de componentes, sendo uma abordagem especialmente útil no desenvolvimento de aplicações do tipo *single page applications*, onde temos uma atualização dinâmica da interface, sem recarregamento completo da página, trazendo uma fluidez maior para nossa aplicação enquanto fornece um grande ganho de produtividade no processo de desenvolvimento.

Dentro do contexto de uma aplicação Laravel, os arquivos que compõem a parte visual do projeto são organizados dentro do diretório *resources*. Nessa pasta principal, existem outros diretórios, cada um com sua responsabilidade específica.

A pasta *components* reúne os elementos reutilizáveis da interface, enquanto as telas completas são estruturadas em *pages*. Funções responsáveis pela comunicação com o backend ficam em *services*, e lógicas reutilizáveis baseadas em Hooks são armazenadas em *hooks*. A Figura 11 demonstra como o React é comumente organizado em um projeto a seguir.

Figura 11 - Estrutura de arquivos do React



Fonte: Elaborada pelos autores (2025).

4.7 Lei Geral de Proteção de Dados (LGPD) e segurança de dados

A LGPD, instituída pela lei n.º 13.709/2018, estabelece diretrizes para o tratamento de dados pessoais e sensíveis no Brasil, assegurando maior segurança, transparência e fiscalização sobre o uso dessas informações.

Garantindo que os dados sensíveis dos cidadãos brasileiros sejam tratados de maneira responsável e ética. A legislação confere aos titulares diversos direitos, como o direito de acesso, retificação, portabilidade e exclusão dos seus dados pessoais.

Além disso, impõe exigências e obrigações às empresas e organizações que realizam o tratamento dessas informações, incluindo a implementação de medidas de segurança e privacidade para garantir a proteção dos dados armazenados.

> A implementação da LGPD é um processo complexo e desafiador, que envolve a conscientização de toda a equipe de uma organização, incluindo a designação de um Encarregado de Proteção de Dados (DPO). Este papel é responsável por supervisionar o tratamento de dados pessoais e a adoção de novos processos e tecnologias para garantir o comprimento da lei (Rodrigues, 2023, p. 10).

Dentro do projeto apresentado, o tratamento de dados segue os princípios da LGPD, utilizando os mecanismos de segurança já fornecidos pelo Laravel, como

autenticação, criptografia, permissões e proteção de acesso. As informações são armazenadas e utilizadas apenas para as finalidades previstas, garantindo sigilo e conformidade com a legislação.

5 RESULTADOS E DISCUSSÃO

Dedicamos essa seção à apresentação dos resultados obtidos, e como os objetivos descritos na Seção 2 foram atingidos.

A seguir, apresentamos uma visão geral dos resultados e subseções para cada objetivo específico, acompanhados de capturas de tela para demonstrarmos as funcionalidades do sistema.

Todas as informações apresentadas são fictícias, respeitando a proteção de dados.

5.1 Visão geral dos resultados

O sistema foi desenvolvido utilizando as tecnologias Laravel no backend, React no frontend e PostgreSQL como banco de dados, conforme definido na metodologia. A aplicação foi construída seguindo o padrão arquitetural MVC (Model–View–Controller), mantendo uma estrutura de diretórios organizada e seguindo padrões de desenvolvimento modernos.

Além disso, todos os módulos foram devidamente testados pela equipe e validados em reuniões com os membros da Pastoral, comprovando que o sistema atendeu aos requisitos levantados e proporcionou melhorias significativas em relação ao controle manual que era utilizado.

5.2 Atendimento aos objetivos específicos

Nessa seção, temos como objetivo elaborar e detalhar como respondemos a cada um dos objetivos específicos inicialmente propostos.

5.2.1 Cadastro e gerenciamento de famílias, agentes e cestas básicas

O sistema permite o cadastro, edição e exclusão de famílias, agentes e cestas básicas, por meio de uma interface intuitiva, onde fica claro para o usuário como

consultar as informações, incluir um novo registro, editar ou excluir um registro já existente.

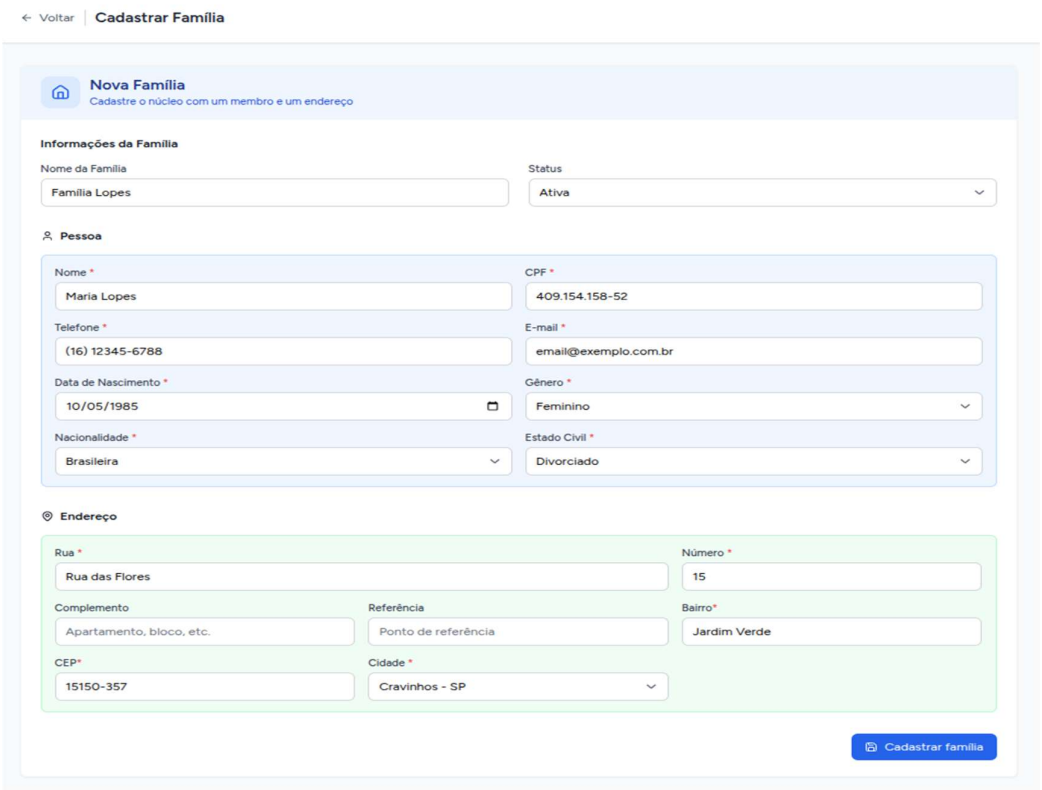
Todas as telas de cadastro foram desenvolvidas com validações de campos e integração direta ao banco de dados via Eloquent ORM, garantindo a consistência das informações.

Incluímos filtros nas telas de consulta, de acordo com o que foi solicitado e julgado como interessante para filtragem dos dados.

Por fim, o sistema apresenta uma auditoria demonstrando todas as ações feitas em um determinado recurso.

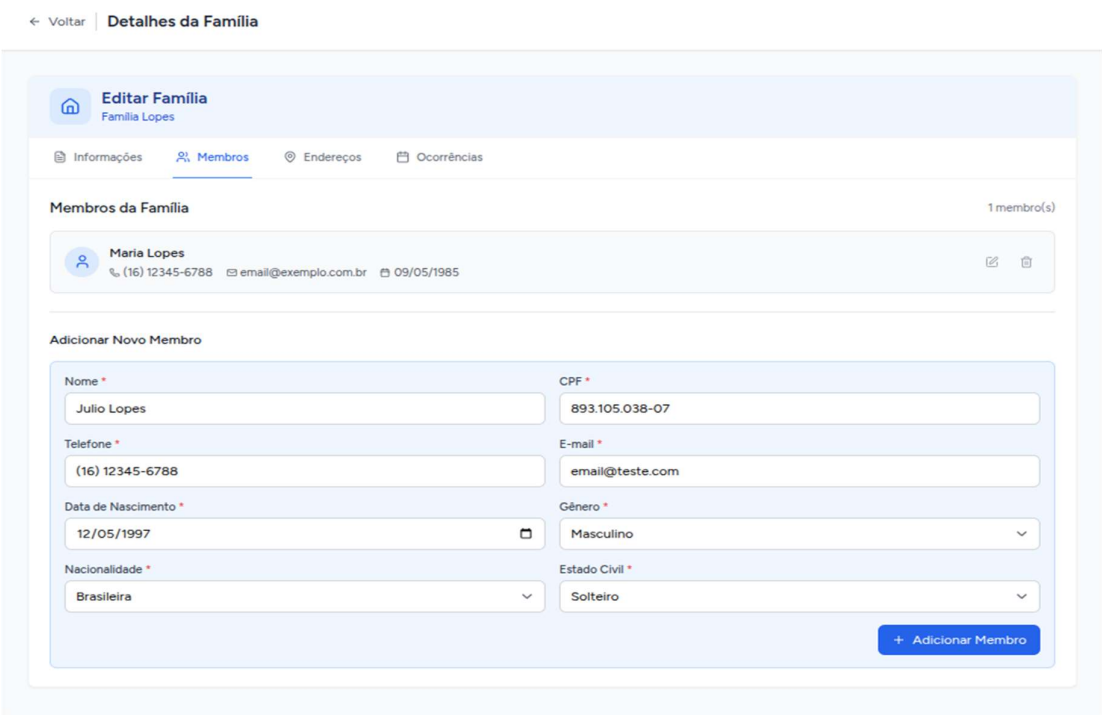
As figuras 12, 13, 14 e 15 demonstram como é feito o gerenciamento das famílias, que são apresentadas a seguir.

Figura 12 - Tela de cadastro de famílias



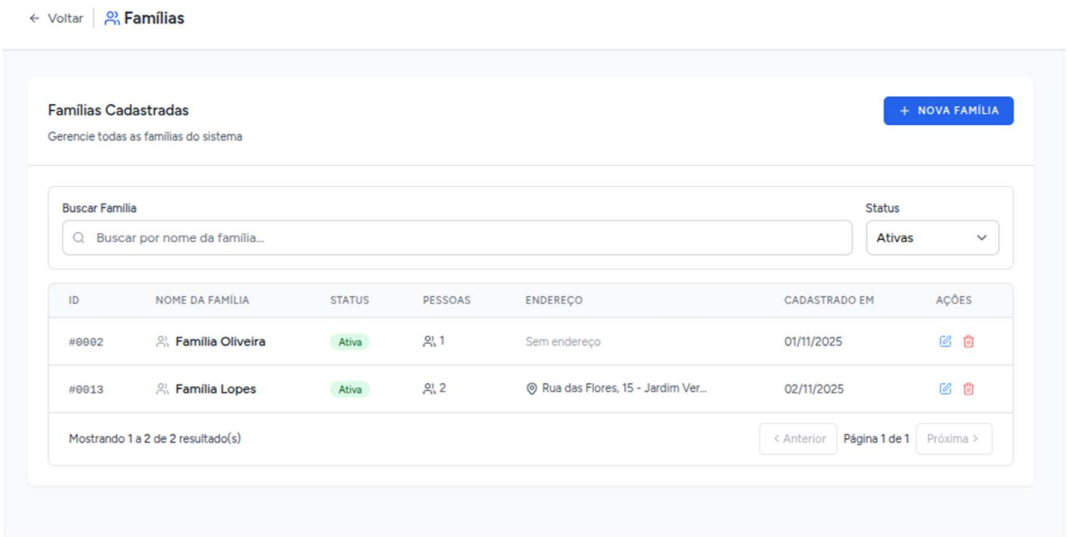
Fonte: Elaborada pelos autores (2025).

Figura 13 - Tela de cadastro de membros adicionais na família



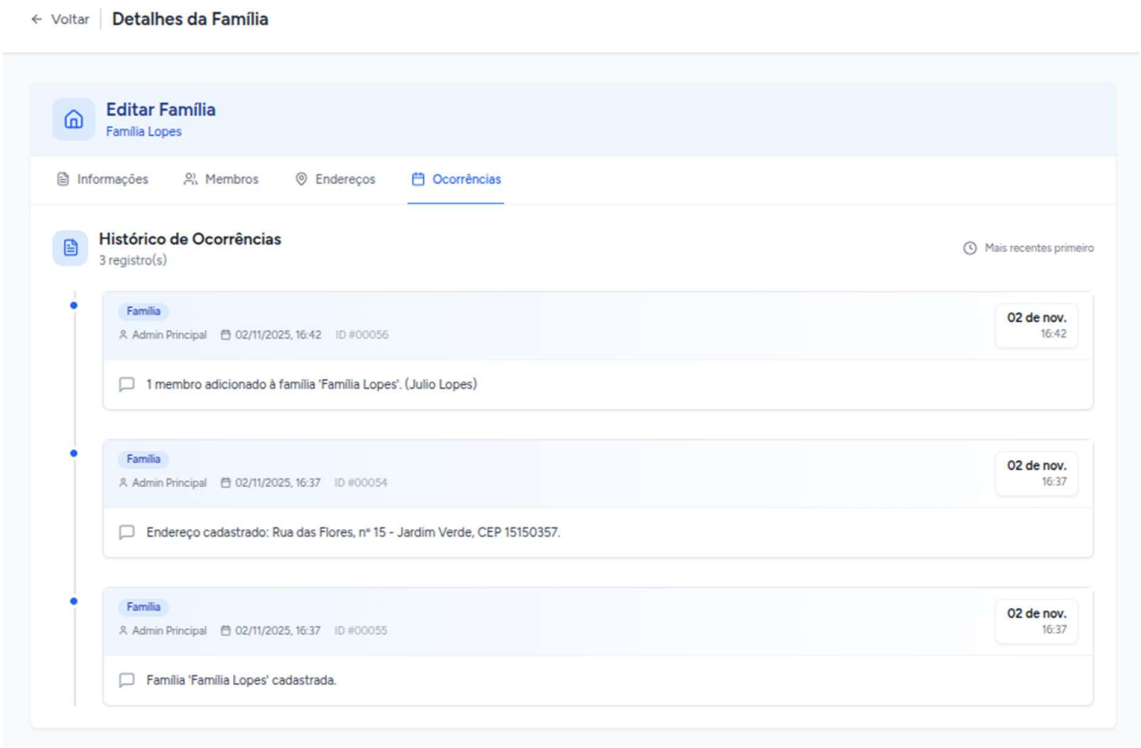
Fonte: Elaborado pelos autores (2025).

Figura 14 - Tela de consulta de famílias:



Fonte: Elaborado pelos autores (2025).

Figura 15 - Ocorrência de famílias

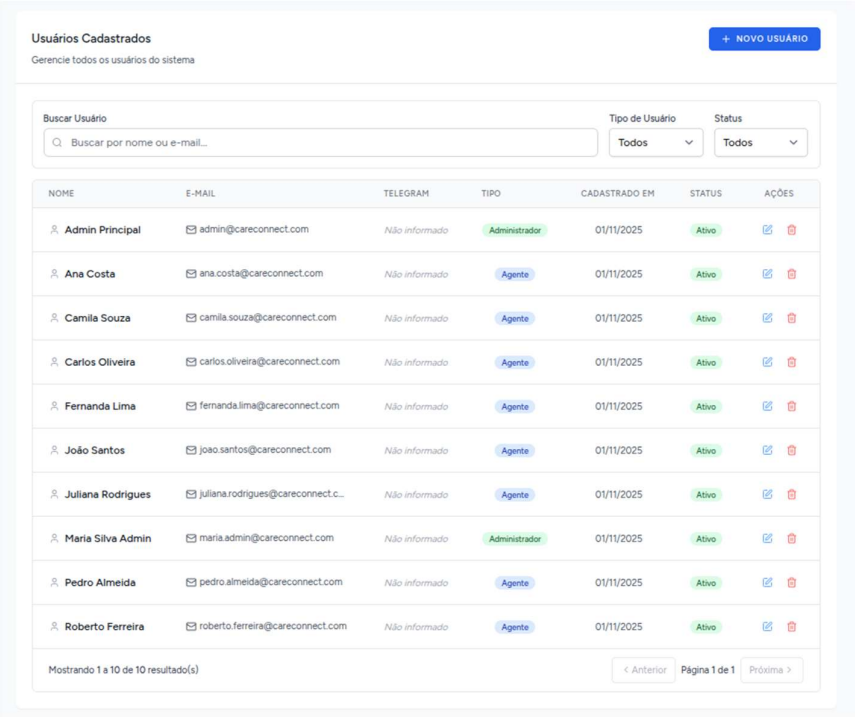


Fonte: Elaborado pelos autores (2025).

As figuras 16 e 17 demonstram como é feito o controle de todos os usuários do sistema, bem como o cadastro de novos usuário, se preciso. Separamos estes usuários entre agentes voluntários e administradores, para controle de permissões dentro da aplicação.

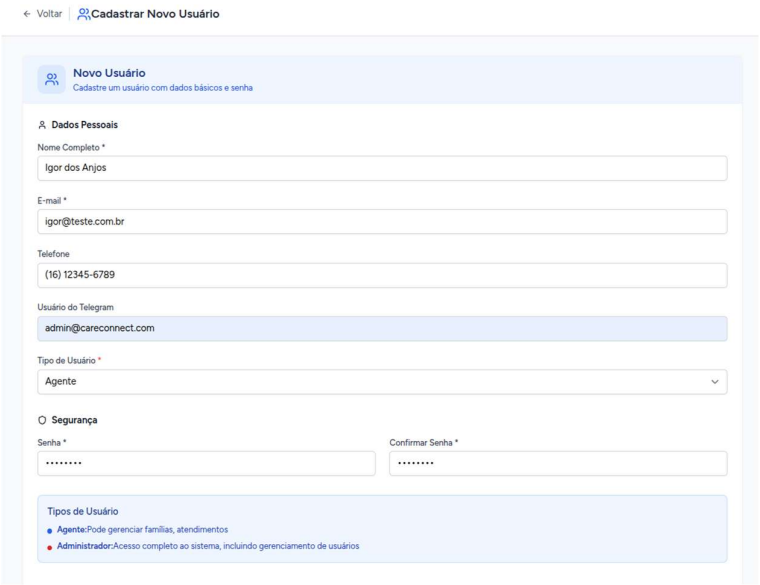
Essa diferença será melhor abordada nas seções seguintes.

Figura 16 - Tela de consulta de usuários



Fonte: Elaborada pelos autores (2025).

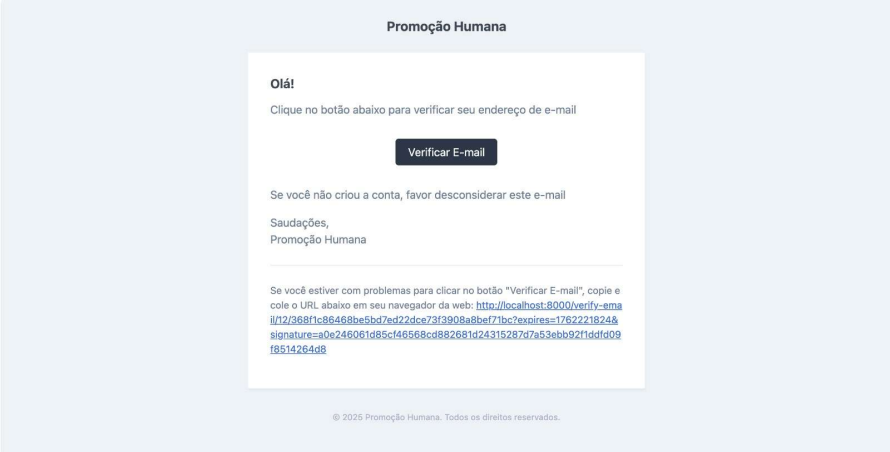
Figura 17 - Tela de cadastro de usuários



Fonte: Elaborada pelos autores (2025).

Para termos uma camada ainda maior de segurança dentro do cadastro de usuários, trabalhamos com a verificação de cada conta criada. Ao incluir um novo usuário no sistema, um e-mail será enviado para o endereço utilizado no cadastro, e a ativação deve ser feita por meio desse link, como demonstra a Figura 18 a seguir.

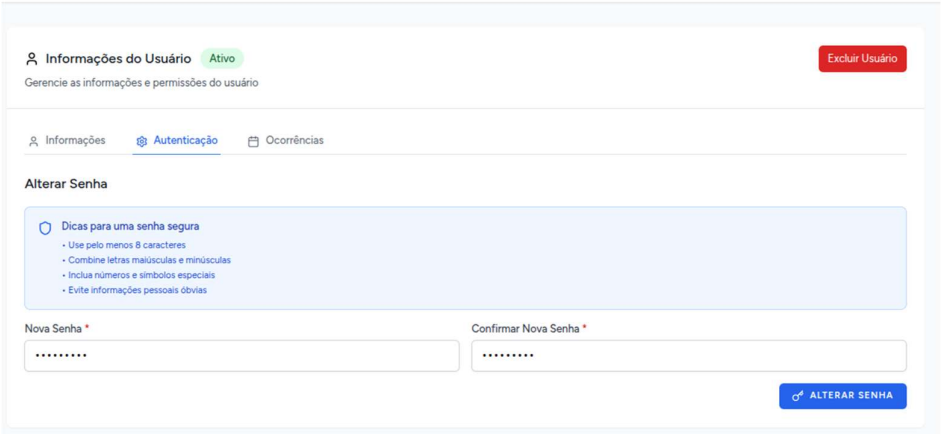
Figura 18 - Ativação de usuários via e-mail



Fonte: Elaborada pelos autores (2025).

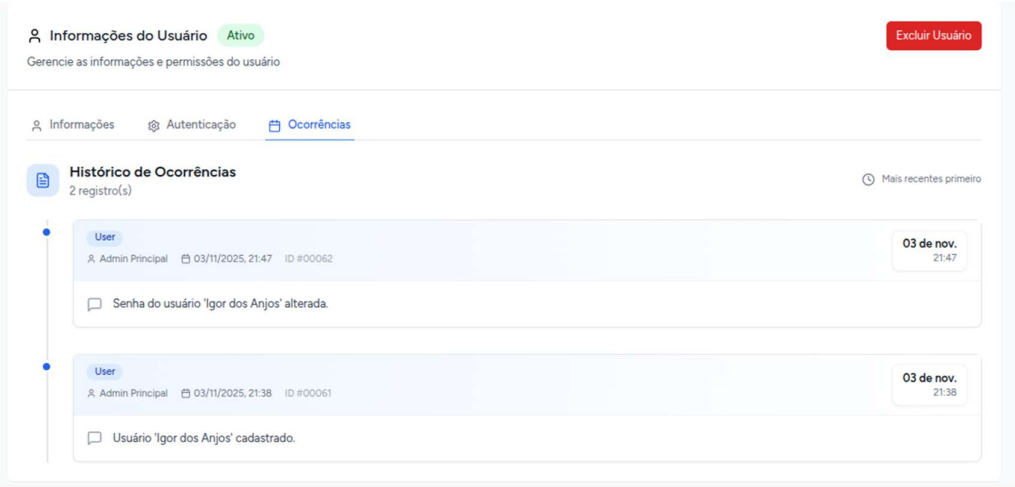
Por fim, disponibilizamos as opções para que o administrador possa alterar a senha dos demais usuários se necessário, e as informações de auditoria para conferência de tudo que tenha sido alterado, como é possível ver nas Figuras 19 e 20:

Figura 19 - Tela de troca de senha dos usuários



Fonte: Elaborada pelos autores (2025).

Figura 20 - Tela de ocorrências do usuário



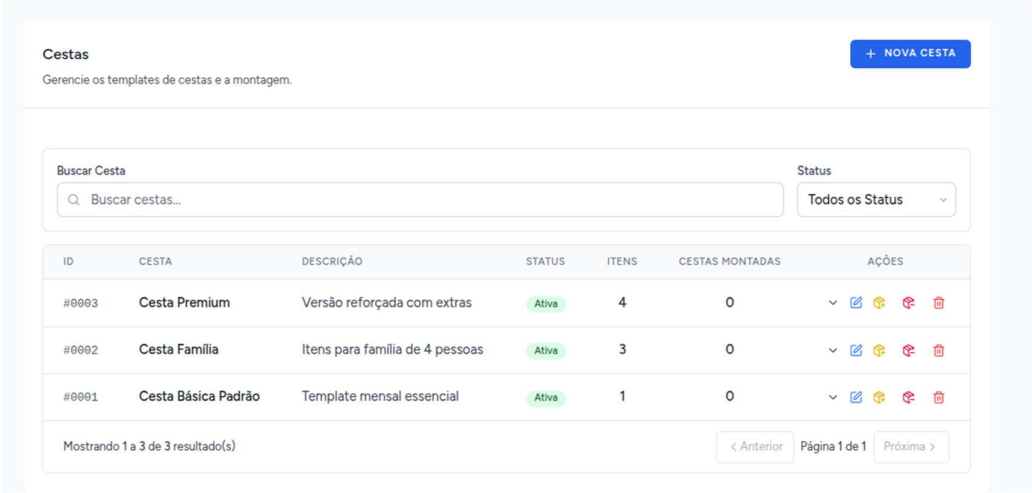
Fonte: Elaborada pelos autores (2025).

As cestas básicas também possuem seu próprio módulo dedicado dentro do sistema. De acordo com o levantamento de requisitos e as reuniões realizadas junto à Pastoral São Francisco de Assis, optamos por trabalhar com um formato onde os administradores podem cadastrar e controlar *templates* de cestas básicas.

Nesse formato, podemos cadastrar N tipos de cestas diferentes, cada uma com sua determinada composição de produtos, detalhando quantos produtos fazem parte da cesta e a quantidade utilizada de cada um. Quando ocorrem as entregas, conseguimos facilmente mapear qual tipo de cesta foi entregue, e com isso temos também a informação de quais produtos fazem parte da cesta que foi entregue.

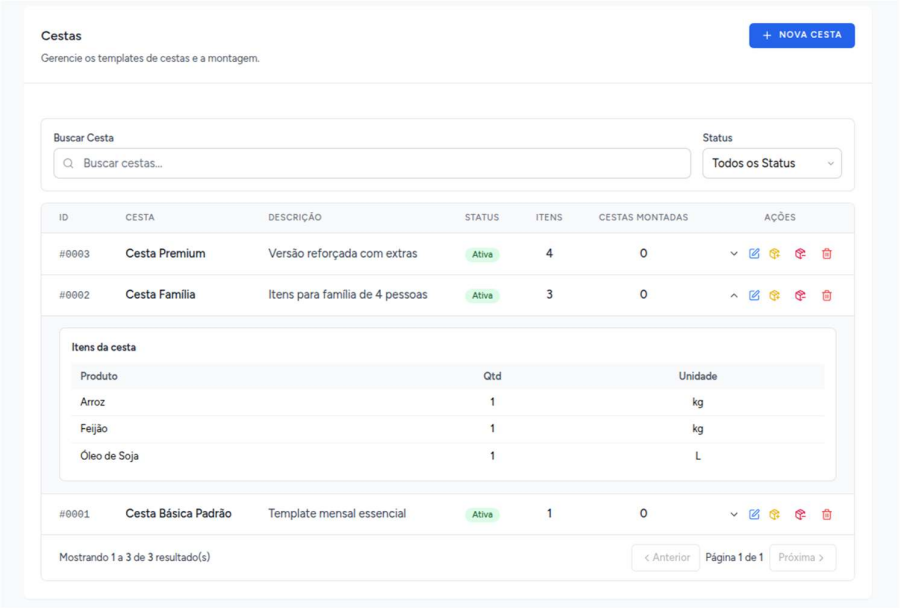
As figuras 21, 22 e 23 demonstram como é feito o cadastro, edição e gerenciamento das cestas, conforme apresentado a seguir.

Figura 21 - Tela de consulta de cestas básicas



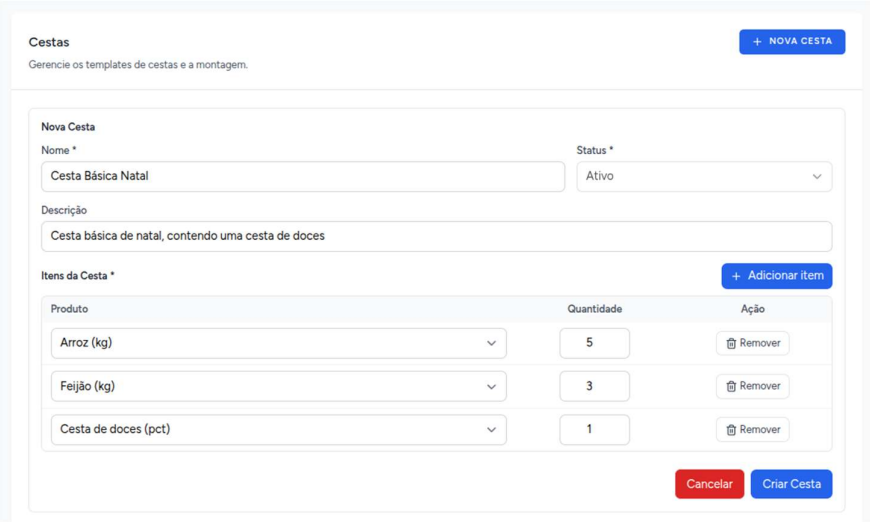
Fonte: Elaborada pelos autores (2025).

Figura 22 - Consulta de composição de cada cesta básica



Fonte: Elaborada pelos autores (2025).

Figura 23 - Tela de cadastro de nova cesta básica



Fonte: Elaborada pelos autores (2025).

Na seção 5.2.5 detalharemos como o módulo de cestas básicas integra-se ao módulo de produtos para que juntos formem um controle de estoque efetivo.

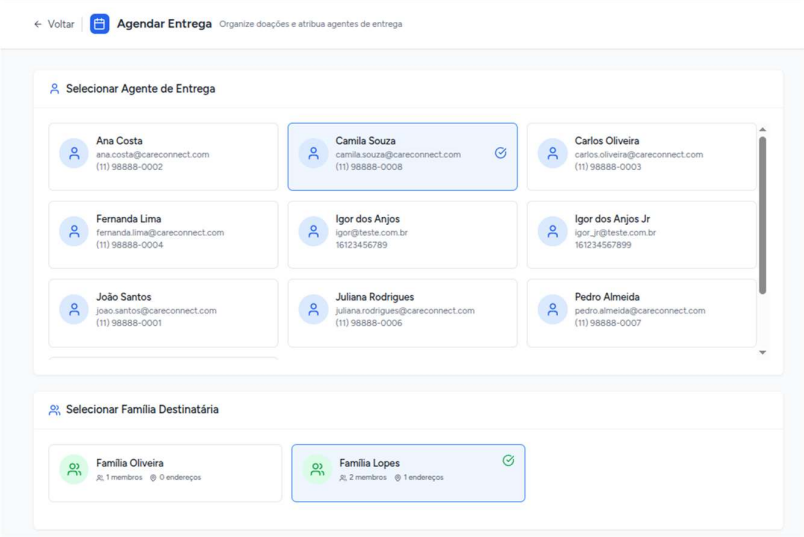
5.2.2 Distribuição de entregas entre agentes

Para o atendimento desse requisito, foi implementado um módulo que permite que os administradores façam a atribuição das entregas aos agentes disponíveis, centralizando a informação e evitando sobrecarga individual.

Esse ponto foi um dos que mais se notou melhora com o sistema, pois antes da aplicação eram escritos diversos papéis manualmente e entregues para cada agente individualmente. Ou seja, uma tarefa de horas agora é realizada dentro de minutos.

A Figura 24 ilustra como essas entregas são atribuídas, bastando selecionar qual o agente voluntário e qual a família que irá receber a cesta básica:

Figura 24 - Tela de atribuição de entregas

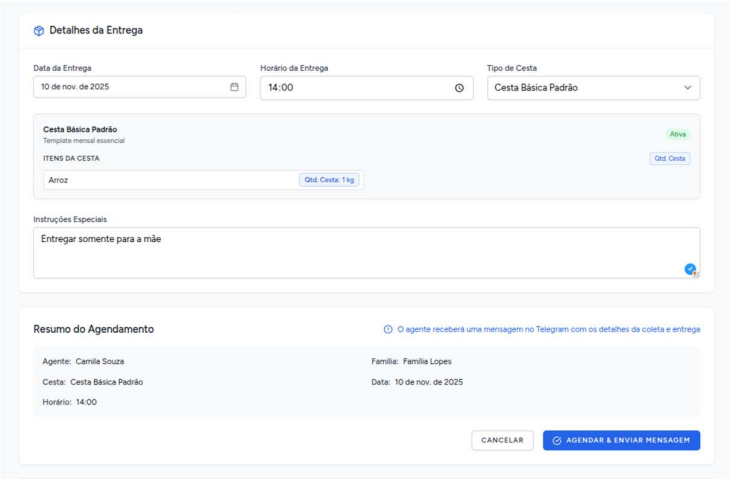


Fonte: Elaborada pelos autores (2025).

Antes da conclusão do agendamento, apresentamos ao usuário a opção de preencher os detalhes dessa entrega, como a data, hora, e alguma observação especial, se necessário.

Também criamos um resumo do agendamento, onde o usuário consegue visualizar a composição da cesta e as demais informações cadastradas, sem precisar sair da tela que se encontra, como demonstra a Figura 25:

Figura 25 - Detalhes e resumo da entrega

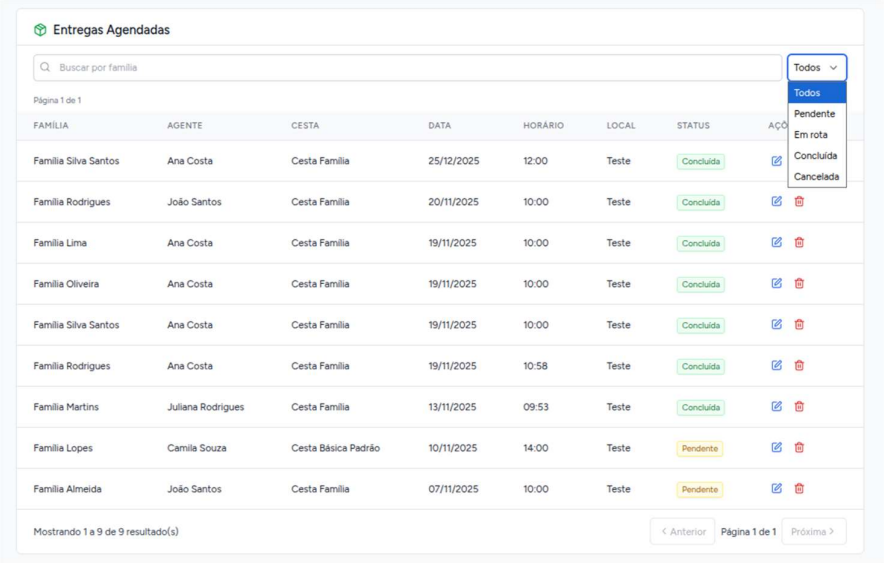


Fonte: Elaborada pelos autores (2025).

Ainda nesse mesmo módulo, foi criado um painel para consulta de todas as entregas cadastradas dentro do sistema. Essas entregas podem ser filtradas de acordo com seu status (cadastrada, em rota, concluída ou cancelada), família beneficiada ou agente responsável pela entrega.

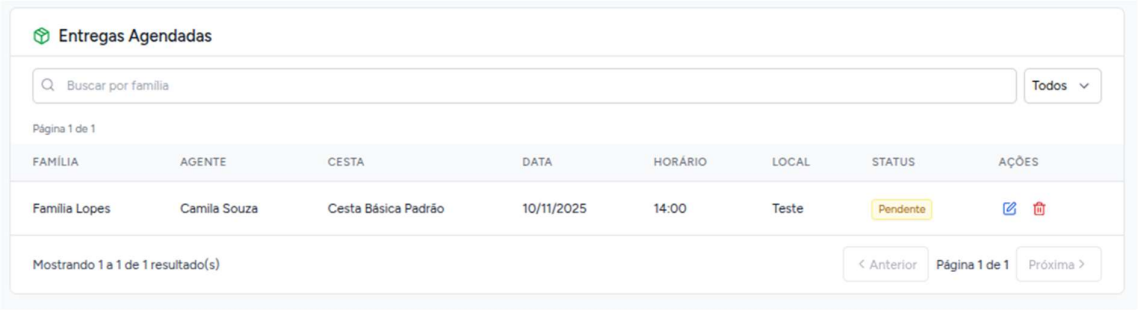
Esse painel foi elaborado de forma que os usuários administradores consigam visualizar todas as entregas existentes, enquanto os usuários agentes consigam visualizar apenas as suas próprias entregas, conforme as Figuras 26 e 27 apresentadas a seguir.

Figura 26 - Tela de consulta de entregas: visão do administrador



Fonte: Elaborada pelos autores (2025).

Figura 27 - Tela de consulta de entregas: visão do agente



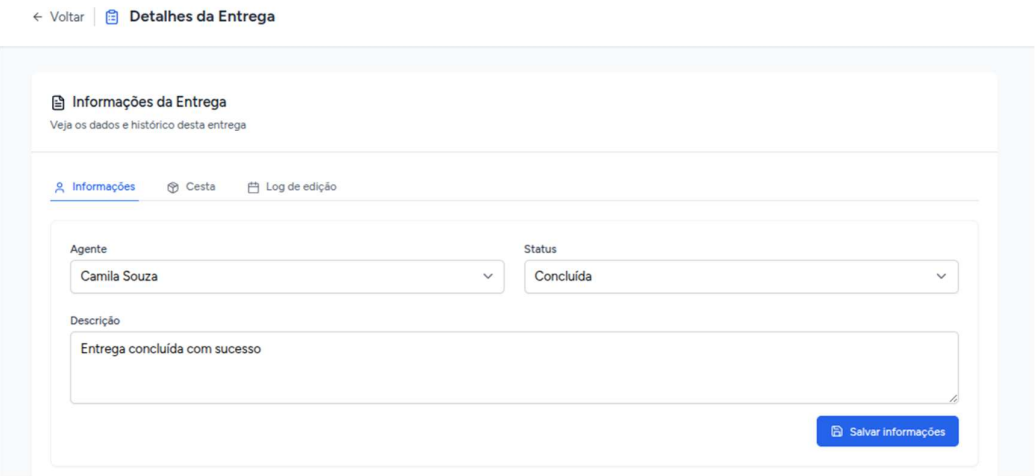
Fonte: Elaborada pelos autores (2025).

Para conclusão deste módulo, criamos uma tela de detalhamento dentro de cada entrega. O propósito desse detalhamento é que o próprio agente vá atualizando

os status de suas entregas, seja quando ele estiver em deslocamento ou quando ele concluir sua entrega.

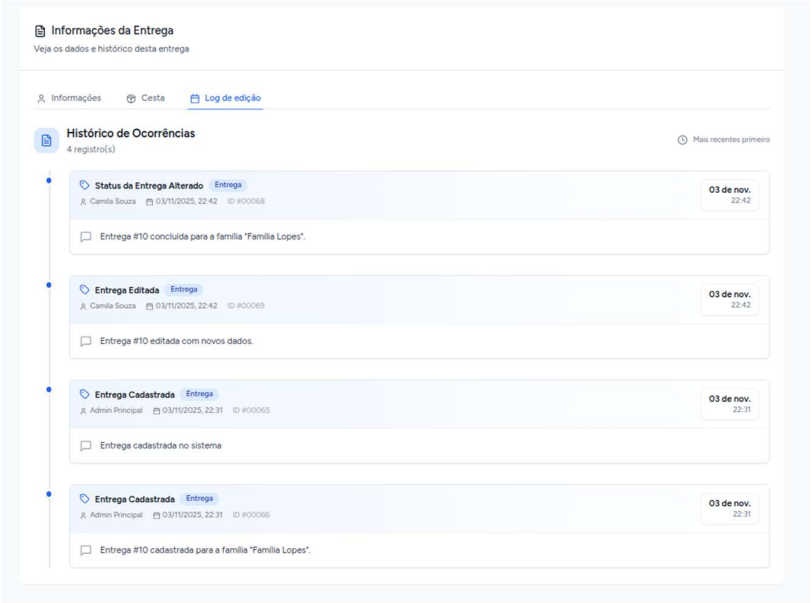
Assim como nas outras rotinas apresentadas, também temos uma seção de ocorrências, que armazena toda a auditoria de quando cada etapa foi realizada, ilustradas nas Figuras 28 e 29 a seguir.

Figura 28 - Detalhamento da entrega



Fonte: Elaborada pelos autores (2025).

Figura 29 - Ocorrências da entrega



Fonte: Elaborada pelos autores (2025).

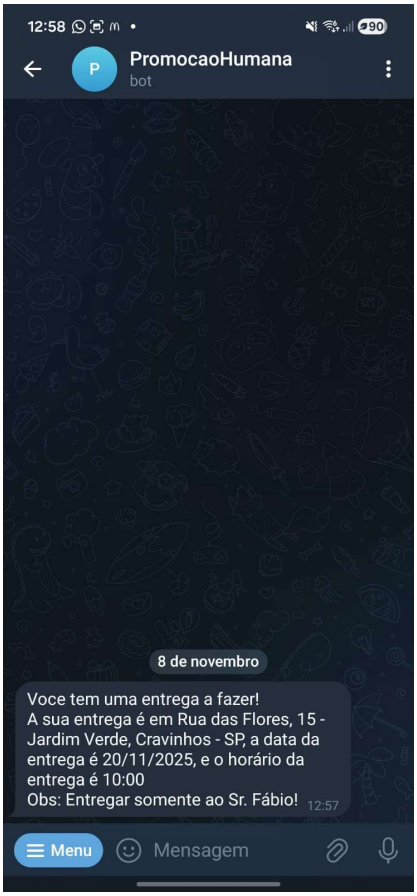
5.2.3 Integração com o Telegram

Visando aprimorar e agilizar a comunicação entre coordenadores e os agentes voluntários, nossa aplicação foi integrada ao serviço de mensagens Telegram. A ideia é que ao invés de passar manualmente para cada agente onde ele deve realizar suas entregas, com alguns cliques todos já estejam cientes do que deve ser feito, com o responsável pela entrega recebendo uma notificação no aplicativo.

No momento que uma entrega é criada, a aplicação realiza uma consulta dentro do banco de dados, conseguindo através da família selecionada encontrar seu endereço, que posteriormente é descrito na mensagem enviada.

A Figura 30 exemplifica como o agente recebe suas entregas:

Figura 30 - Exemplo de mensagem recebida pelo agente de entrega



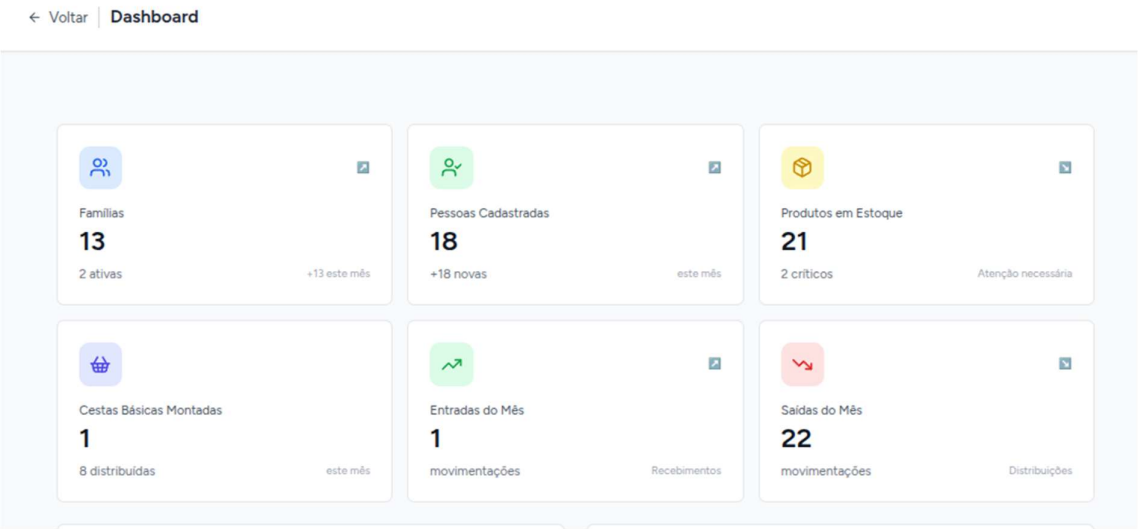
Fonte: Elaborada pelos autores (2025).

5.2.4 Painel administrativo

Para o atendimento desse requisito, optamos pela criação de um *dashboard*, que contém as principais informações do sistema em uma única tela centralizada. Dados que antes eram controlados manualmente e com baixa confiabilidade agora são apresentados digitalmente, por meio de gráficos, que auxiliam também na tomada de decisão dos coordenadores.

Na primeira metade do *dashboard*, temos indicadores que apresentam dados como quantidade de famílias cadastradas e ativas, quantos produtos existem no estoque, quantas entradas e saídas houveram no mês corrente, dentre outros, como é possível ver na Figura 31:

Figura 31 - Dashboard do sistema (indicadores)



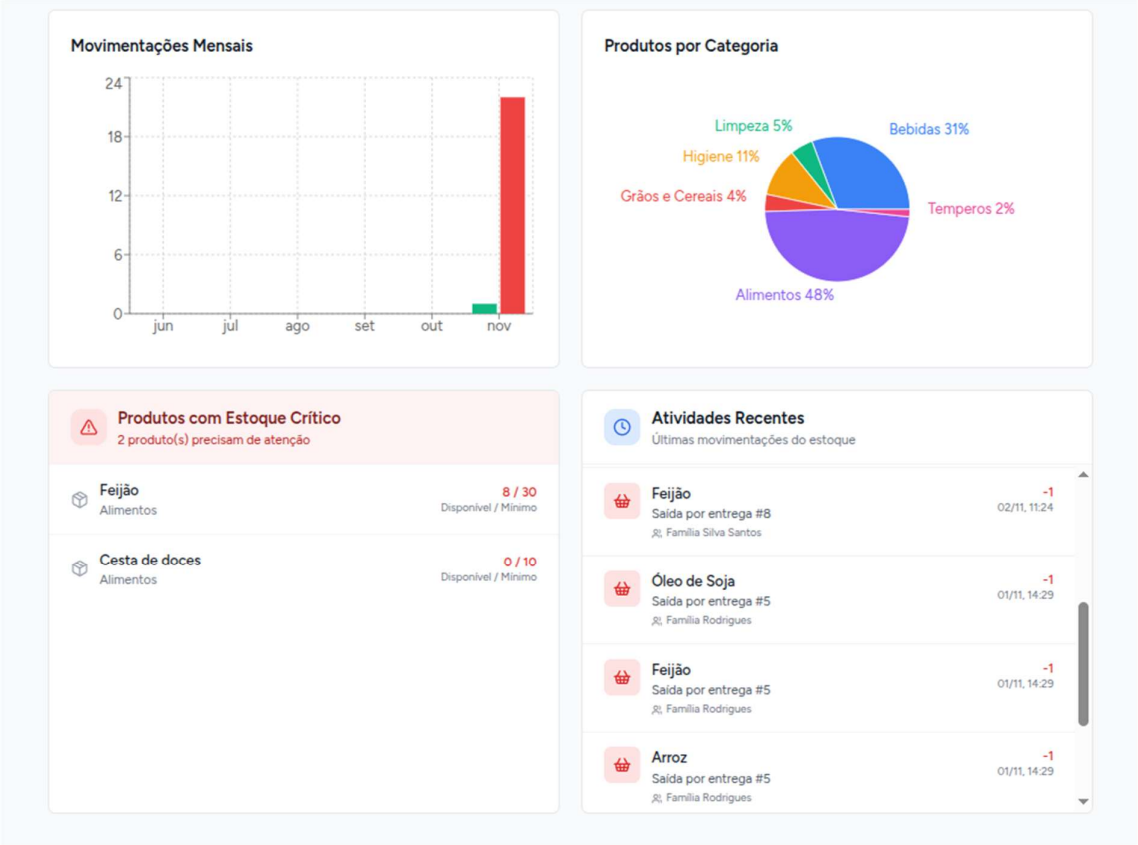
Fonte: Elaborada pelos autores (2025).

Na segunda metade do *dashboard* temos dois gráficos e duas seções de avisos. Os gráficos demonstram a porcentagem de produtos cadastrados por categoria e um acompanhamento mensal de quantas entradas e saídas foram realizadas historicamente no sistema.

Também apresentamos nessa tela centralizada um alerta de produtos que estão com níveis de estoque abaixo do mínimo, de acordo com o que é definido no momento do cadastro do produto.

Por fim, a última seção do *dashboard* apresenta um breve resumo sobre as últimas movimentações de estoque que tenham sido feitas. A Figura 32 demonstra esses dados:

Figura 32 - Dashboard do sistema (gráficos e alertas)



Fonte: Elaborada pelos autores (2025).

Todas as informações selecionadas para compor esse painel administrativo foram levantadas em conjunto com a Pastoral, de maneira que pudemos entregar um *dashboard* elegante e satisfatório do ponto de vista funcional.

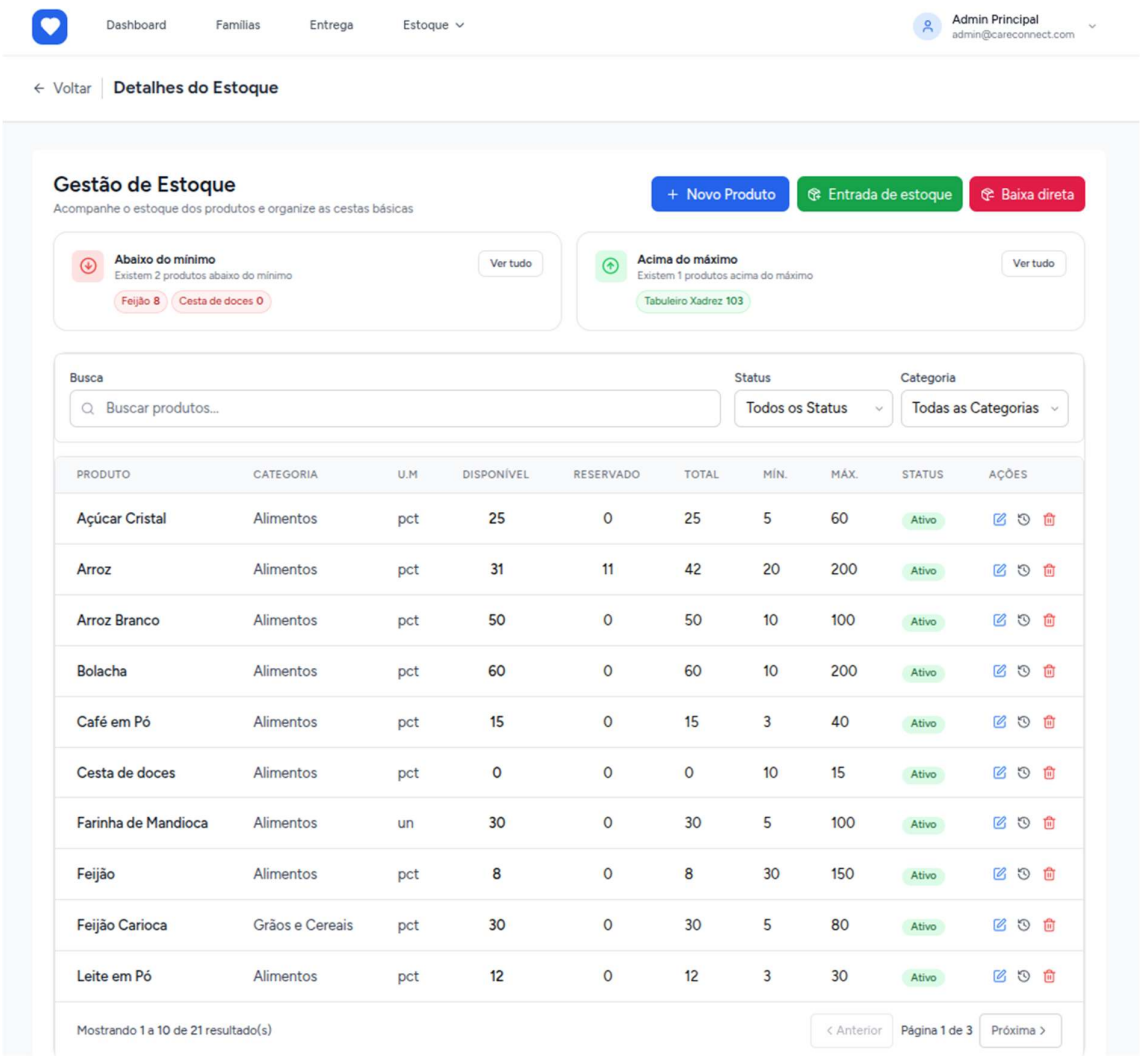
5.2.5 Controle de estoque

Juntamente a seção de entregas, este módulo do sistema era considerado primordial, pois o controle de estoque era considerado outra grande dor da Pastoral quando feito de maneira manual.

Dessa forma, o objetivo aqui foi criar uma maneira fácil e confiável para que o estoque dos produtos seja devidamente controlado, sem margens para quaisquer inconsistências.

A Figura 33 demonstra a tela inicial da gestão do estoque a seguir.

Figura 33 - Tela inicial de gestão do estoque



Fonte: Elaborada pelos autores (2025).

Mantendo o padrão de outras áreas do sistema, temos um grid que apresenta um apanhado das informações, mostrando todos os produtos cadastrados e com possibilidade de filtragem.

Nesse grid, temos colunas voltadas a informação do produto em si, como seu nome, sua categoria, sua unidade de medida e seu status (ativo ou inativo). Durante

o cadastro do produto, também é possível selecionar qual é o estoque mínimo e o estoque máximo que esse produto deve ter, e essa informação é usada para apresentar alertas se há produtos faltando ou sobrando.

No entanto, também temos colunas que são variáveis, e são nessas colunas que o acompanhamento efetivo do estoque é feito.

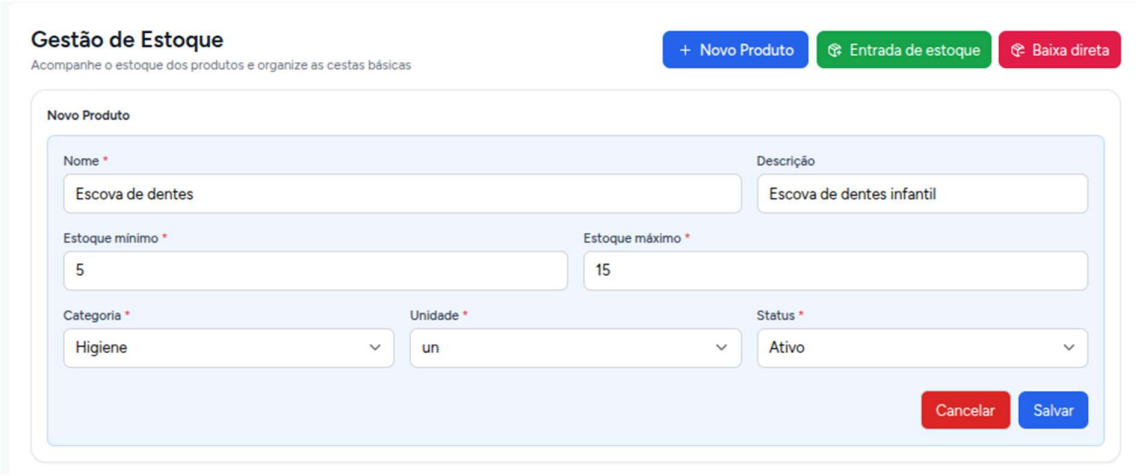
Apresentamos ao usuário qual o saldo disponível, que são as unidades do produto que não estão alocadas em nenhuma rotina, ou seja, o saldo que está disponível para uso. Também utilizamos um conceito de estoque reservado, que se trata de quando determinada quantidade do produto é alocada em alguma cesta básica.

Por exemplo, se temos 10 unidades disponíveis, e na sequência são utilizadas 5 unidades nas montagens de cestas, o resultado será um produto com 5 unidades disponíveis e 5 unidades reservadas.

O último campo variável dentro do grid é o que demonstra o estoque total, que nada mais é que a soma da quantidade disponível e reservada.

Como a Figura 34 demonstra, tudo que é considerado informativo é possível de ser digitado pelo usuário no formulário de cadastro de novos produtos a seguir.

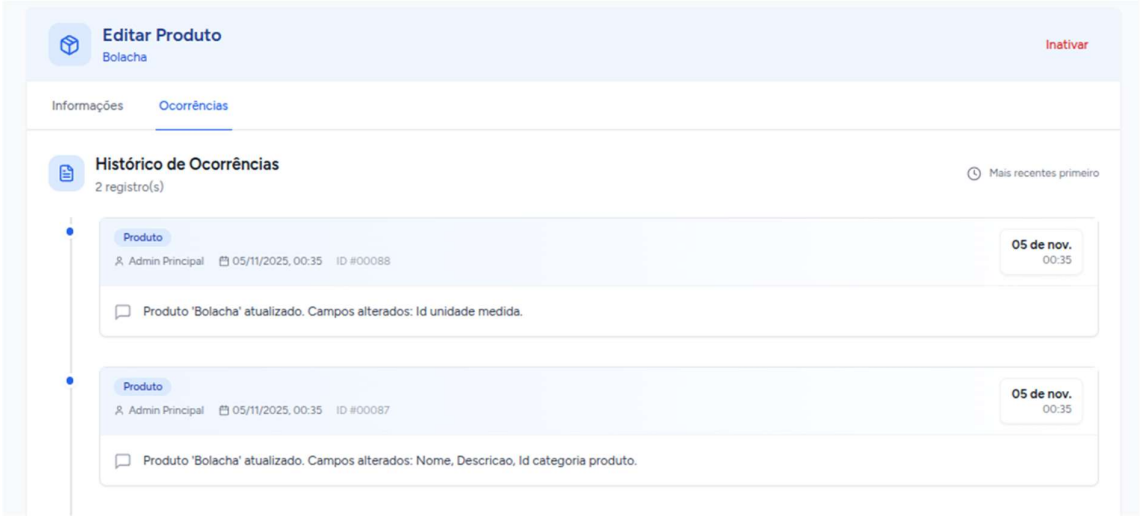
Figura 34 - Tela de cadastro de produtos



Fonte: Elaborada pelos autores (2025).

Na Figura 35, demonstramos que conforme o padrão da aplicação, qualquer edição feita no produto é registrada como uma ocorrência. É importante ressaltar que de maneira nenhuma o usuário pode editar quantidades em estoque, apenas dados cadastrais.

Figura 35 - Tela de ocorrência de produtos

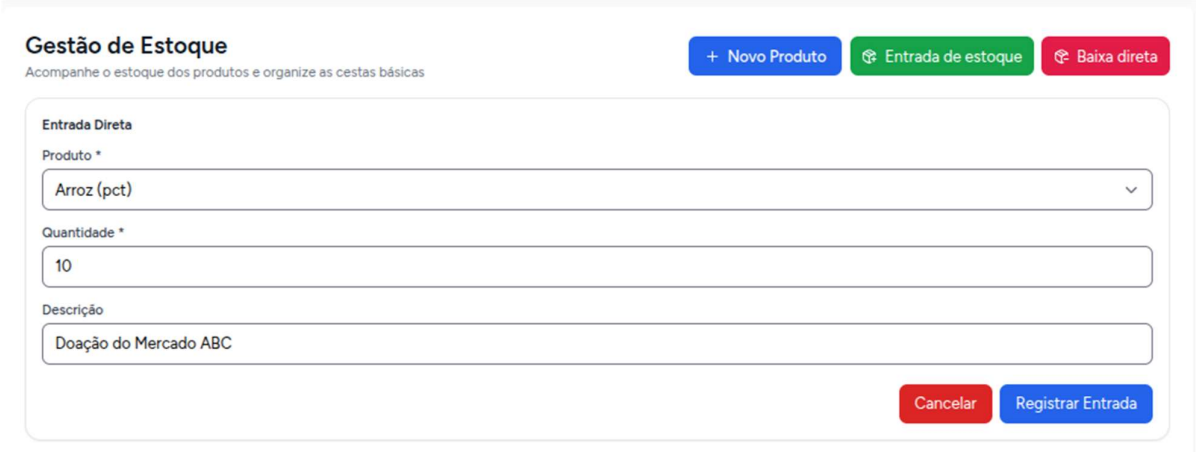


Fonte: Elaborada pelos autores (2025).

Foi levantado junto à Pastoral que também seria necessário que houvesse alguma forma de realizar a entrada dos produtos no saldo, bem como uma forma de realizar saídas que não sejam por cesta básica, pois eventualmente há perdas, avarias, etc.

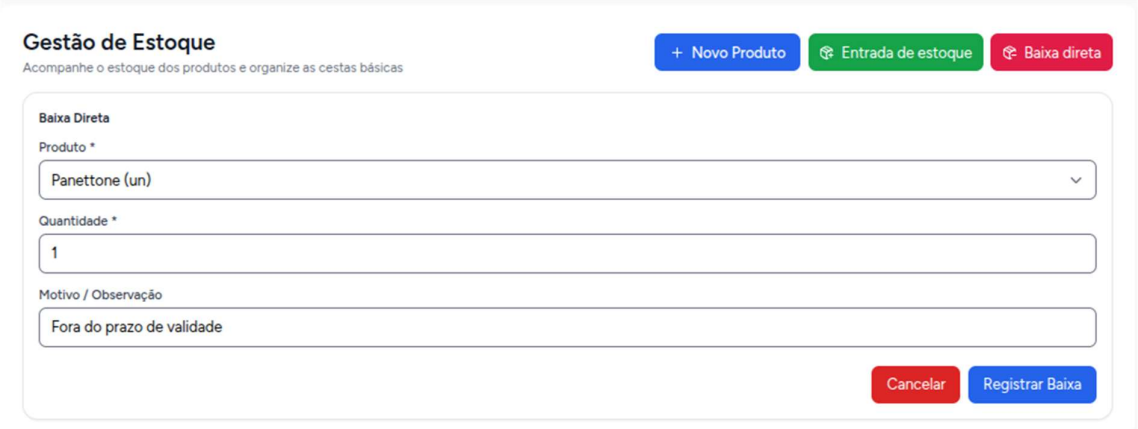
Para isso, criamos as opções de “Entrada de Estoque” e “Baixa Direta”, onde é possível selecionar um determinado produto e registrar suas entradas e saídas, conforme demonstrado nas Figuras 36 e 37:

Figura 36 - Entrada de estoque



Fonte: Elaborada pelos autores (2025).

Figura 37 - Baixa direta

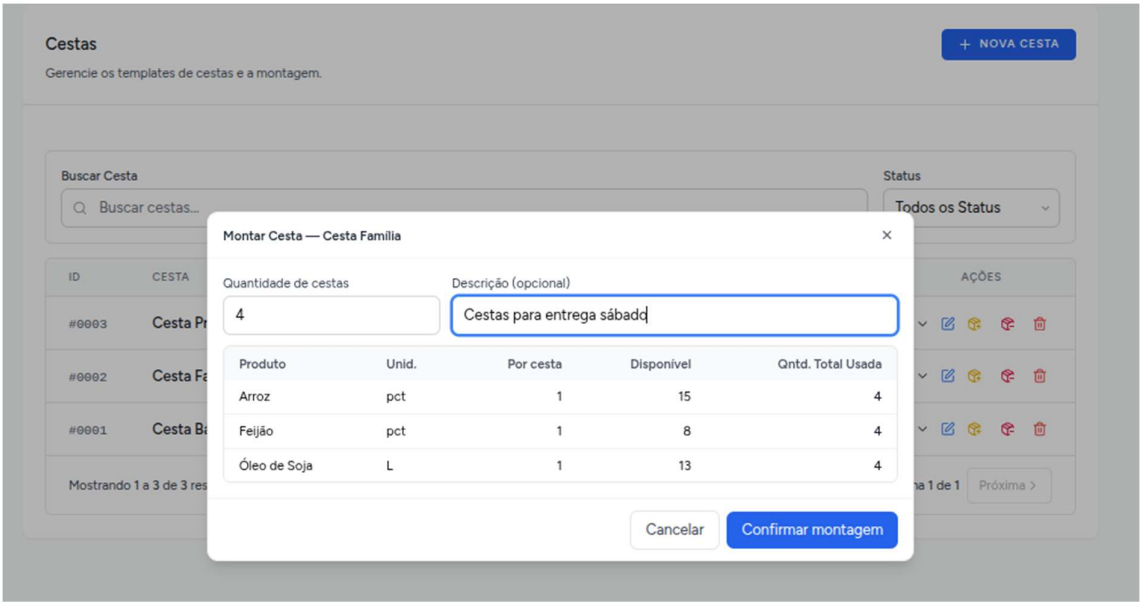


Fonte: Elaborada pelos autores (2025).

Para a integração com o Módulo de Cestas Básicas, foi criado uma ação dentro do grid das cestas.

Essa ação abre um modal, onde o usuário visualiza quais produtos compõe a cesta, quantas cestas ele está montando e qual o total que será utilizado. A reserva do estoque é realizada no momento que o usuário clica no botão “Confirmar montagem”. A Figura 38 apresenta esse modal, logo a seguir.

Figura 38 - Modal de montagem de cesta



Fonte: Elaborada pelos autores (2025).

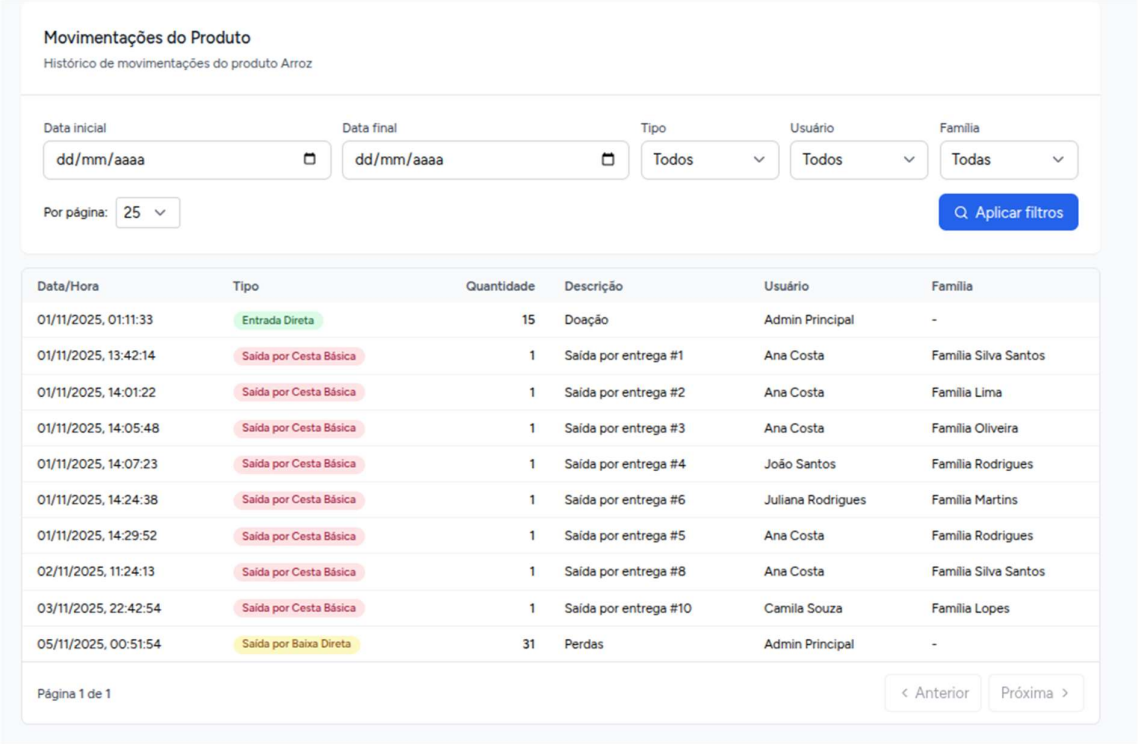
O usuário também tem a opção de desmontar cestas já montadas, se necessário. Nesse caso, o movimento inverso é realizado, a quantidade que estava reservada retorna ao disponível.

Finalizando essa integração entre os módulos, quando uma cesta é montada, temos um campo utilizado para contar quantas cestas daquele *template* estão montadas. No momento que sua entrega for concluída, esse campo será decrementado e a quantidade reservada do produto será efetivamente baixada e sairá do estoque.

Por fim, para manter a rastreabilidade e acompanhar tudo isso, criamos uma tela para visualização das movimentações de estoque realizadas por produto.

Nessa tela, é possível visualizar e filtrar quando ocorreu a movimentação, qual foi essa movimentação (entrada, saída direta ou saída por cesta), quem recebeu esse produto, entre outros. A demonstração está na Figura 39 a seguir.

Figura 39 - Tela de movimentação de estoque



Fonte: Elaborada pelos autores (2025).

Com esse módulo, a Pastoral passou a ter total controle sobre entradas, saídas e reservas de produtos, eliminando inconsistências e reduzindo erros de contagem.

6 CONSIDERAÇÕES FINAIS

O desenvolvimento do sistema web proposto foi muito importante, tanto para nossa trajetória acadêmica quanto para o contexto de atividades sociais da Pastoral São Francisco de Assis. Por meio da aplicação dos conhecimentos adquiridos ao longo do curso nas áreas de programação, banco de dados e desenvolvimento web, foi possível atender a uma demanda real da Pastoral, modernizando processos que antes eram realizados de maneira manual e os tornando muito mais ágeis e eficientes.

Todos os objetivos definidos na Seção 2 foram integralmente alcançados: o sistema permite o cadastro e gerenciamento de famílias, agentes e produtos; permite a distribuição de entregas entre os voluntários; oferece integração com o Telegram para envio de notificações; disponibiliza um painel administrativo com informações em tempo real; implementa controle de estoque confiável e seguro; e gerencia a autenticação de usuários.

A utilização de tecnologias modernas, como o Laravel no backend e o React no frontend proporcionou uma estrutura robusta e moderna, semelhante ao que é encontrando em aplicações no mercado. O banco de dados PostgreSQL, aliado ao ORM Eloquent, garantiu a consistência das informações e a rastreabilidade das operações. Todo o desenvolvimento seguiu práticas ágeis baseadas no framework Scrum, o que possibilitou melhor organização e divisão de tarefas entre os membros da equipe.

Os resultados alcançados demonstraram que a digitalização dos processos da Pastoral trouxe benefícios imediatos, como a redução do tempo gasto em cadastros, maior controle de estoque e maior transparência na distribuição das doações.

Do ponto de vista acadêmico, o trabalho permitiu a aplicação prática de diversos conceitos abordados ao longo do curso de Ciência da Computação, desde a modelagem de sistemas e banco de dados até o desenvolvimento orientado a objetos e a segurança da informação.

Como melhorias futuras, temos a implementação de novos módulos, como um painel de relatórios avançados com estatísticas e indicadores de desempenho, um sistema de doações online integrado a plataformas de pagamento, além de um

aplicativo próprio da Pastoral para agentes e coordenadores. Essas evoluções poderiam ampliar ainda mais o alcance e a eficiência do sistema.

Em suma, conclui-se que o projeto atingiu plenamente seus objetivos, demonstrando que a união entre conhecimento técnico e assistência social é capaz de gerar boas soluções tecnológicas, capaz de resolver problemas reais.

REFERÊNCIAS

AHMED, Nabila; AZIM, Kazi Sanwarul; JAFOR, A. H. M.; SHAYED, Azher Uddin; HOSSAIN, Mir Abrar; KHAN, Obyed Ullah. Digital Transformation in Non-Profit Organizations: Strategies, Challenges, and Successes. **Advanced International Journal of Multidisciplinary Research**, v. 2, n. 5, p. 8-9, set./out. 2024. DOI: 10.62127/aijmr.2024.v02i05.1097. Disponível em: https://www.researchgate.net/publication/384667845_Digital_Transformation_in_Non-Profit_Organizations_Strategies_Challenges_and_Successes. Acesso em: 16 out. 2025.

ANESE, Vivian; COSTA, Carlos; COELHO, Elenise. Impacto social das ações de uma organização sem fins lucrativos. **Revista Pensamento Contemporâneo em Administração**, v. 12, n. 1, p. 61-75. DOI: https://doi.org/10.12712/rpca.v12i1.1193. Disponível em: https://periodicos.uff.br/pca/article/view/11345. Acesso em: 28 mar. 2018.

BARRETTO, Maria Isabel Franco. **Estudo da gestão do relacionamento do cliente - CRM (Customer Relationship Management) e proposta de soluções para uma empresa do setor sucroalcooleiro.** 2004. Dissertação (Mestrado em Engenharia de Produção) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2004. DOI: https://doi.org/10.11606/D.18.2017.tde-31072017-114817. Disponível em: https://teses.usp.br/teses/disponiveis/18/18140/tde-31072017-114817/publico/Dissert_Barretto_MariaIF.pdf. Acesso em: 27 mar. 2025.

BRASIL. [Constituição (1988)]. **Constituição da República Federativa do Brasil de 1988**. Brasília, DF: Presidente da República, [2024]. Disponível em: http://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm. Acesso em 18 maio. 2025.

BOVOLENTA, G. A. **Cesta básica e assistência social:** notas de uma antiga relação. Serviço Social & Sociedade, SciELO Brasil. p. 507–525, 2017.

CHACON, Scott; STRAUB, Ben. **Pro Git**. 2. ed. São Francisco: Apress, 2014.

CONVERSE, Tim; PARK, Joyce. **PHP**: a Bíblia**.** 2. ed. São Paulo: Elsevier, 2003.

DALL'OGLIO, Pablo. **PHP - Programando com Orientação a Objetos**. 3. ed. São Paulo: Novatec, 2015.

FAO; IFAD; UNICEF; WFP; WHO. **The State of Food Security and Nutrition in the World 2024:** Financing to end hunger, food insecurity and malnutrition in all its forms. Roma: FAO, 2024. Disponível em: https://www.fao.org/publications/fao-flagship-publications/the-state-of-food-security-and-nutrition-in-the-world/en. Acesso em: 18 maio. 2025.

FOWLER, Martin. **Patterns of Enterprise Application Architecture**. Boston: Addison-Wesley, 2002.

GACKENHEIMER, Cory. **Introduction to React**. New York: Apress, 2015.

GIL, Antonio Carlos. **Métodos e técnicas de pesquisa social.** 6. ed. São Paulo: Atlas, 2007.

GOURLEY, David; TOTTY, Brian. **HTTP**: The Definitive Guide. Sebastopol: O'Reilly Media, Inc., 2002.

HERTZOG, Raphaël; MASSON, Roland. **Manual do Administrador Debian**: Da instalação à administração de servidores Debian GNU/Linux. 2. ed. São Paulo: Livro Livre, 2013.

LAKATOS, Eva; MARCONI, Marina. **Fundamentos de metodologia científica.** 5. ed. São Paulo: Atlas, 2003.

LARAVEL. **Laravel Documentation.** 2025. Disponível em: https://laravel.com/docs/12. Acesso em: 28 mar. 2025.

LOELIGER, Jon; McDONALD, Matthew. **Version Control with Git**: Powerful tools and techniques for collaborative software development. 2. ed. Sebastopol: O'Reilly Media, 2012.

MENDES, Luiz Carlos Abreu. **Visitando o "terceiro setor" (ou parte dele).** Texto para Discussão, n. 647. Brasília: Instituto de Pesquisa Econômica Aplicada (Ipea), 1999. Disponível em: http://repositorio.ipea.gov.br/handle/11058/2618. Acesso em: 17 out. 2025.

MILANI, André. **PostgreSQL - Guia do Programador.** São Paulo: Novatec, 2008.

MOZILLA FOUNDATION. **HTTP – Visão geral.** Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP. Acesso em: 23 out. 2025.

NOGUEIRA, Alexandre. **Laravel**: o poder do framework PHP mais popular da atualidade. Disponível em: https://www.hostgator.com.br/blog/laravel-framework-php/. Acesso em: 28 mar. 2025.

POSTGRESQL. **PostgreSQL Documentation – Version 18**. 2025. Disponível em: https://www.postgresql.org/docs/. Acesso em: 07 abr. 2025.

REACT. **React Documentation.** 2025. Disponível em: https://react.dev/reference/react. Acesso em: 21 ago. 2025

RED HAT. **O que é uma API Rest?.** 2023. Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. Acesso em 23 out. 2025.

RODRIGUES, Lucas. **Implantação do Privacy By Design no Sistema de Recrutamento de Pessoas**: Um Estudo Sobre a Adequação à LGPD. TCC (Graduação em Ciência da Computação) - PUC Goiás. Goiânia, p. 37. 2023. Disponível em: https://repositorio.pucgoias.edu.br/jspui/bitstream/123456789/6277/1/TCC%20Lucas%20Melo%20Rodrigues%5b1%5d.pdf. Acesso em 28 mar. 2025.

SCHWABER, Ken; SUTHERLAND, Jeff. **O guia do Scrum.** 3. ed. ScrumGuides, 2020.

SCOTT, Emmit. **SPA Design and Architecture:** Understanding Single Page Web Applications. Shelter Island: Manning Publications, 2015.

SILVA, Edson. **Serviço social e a ação sócio-pastoral da Igreja Católica:** assistência, promoção humana e emancipação social. Pontifícia Universidade Católica de São Paulo, 2025. Disponível em: https://repositorio.pucsp.br/jspui/handle/handle/17471. Acesso em: 25 mar. 2025.

SILVA, Claudia; COSTA, Selma. As ações assistenciais promovidas pelas igrejas pentecostais e suas expressões na política de assistência social do município de Londrina. **Semina**: Ciências Sociais e Humanas, [S. l.], v. 28, n. 1, p. 45-58, 2007. DOI: 10.5433/1679-0383.2007v28n1p45. Disponível em: https://ojs.uel.br/revistas/uel/index.php/seminasoc/article/view/3773. Acesso em: 18 maio. 2025.

SILVERMAN, Richard. **Git Pocket Guide:** A working introduction. Sebastopol: O'Reilly Media, 2013.

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

TATROE, Kevin; MACINTYRE, Peter. **Programming PHP:** Creating Dynamic Web Pages. 4. ed. Sebastopol: O'Reilly Media, 2013.

TORRES, Fernando Esquírio. **Desenvolvimento de API REST**. São Paulo: Editora Senac São Paulo, 2021.

YADAV, S. Chandra; SINGH, Sanjay Kumar. **An introduction to client/server computing.** New Delhi: New Age International Publishers, 2009.