FACULDADE DE TECNOLOGIA ARARAS – ANTONIO BRAMBILLA

DESENVOLVIMENTO DE SOFTWARE MULTIPLATAFORMA

Bruno Henrique Guinerio

Gabriel Cardoso Schranck

Kalliel Marcos Pinheiro

Pedro Rufino da Mata Neto

Stephan Mendes de Oliveira

DOSSIÊ – CESTAS DO AMOR

DOCUMENTAÇÃO DA IMPLEMENTAÇÃO DA APLICAÇÃO

ARARAS

2025

SUMÁRIO

1 TERMO DE ABERTURA DO PROJETO (TAP)

1.1 Cestas do Amor

Sistema para gerenciamento de doações de cestas básicas a famílias em situação de vulnerabilidade social, distribuídas pela prefeitura e instituições religiosas de Mogi Guaçu. O gerenciamento de doações atualmente é feito de forma manual, gerando muitos erros. Por isso este projeto visa digitalizar este processo, diminuindo o número de erros, além de facilitar o acompanhamento de métricas importantes.

1.2 Instituição

O projeto Cestas do Amor foi desenvolvido por uma equipe de alunos do quinto semestre do curso de Desenvolvimento de Softwares Multiplataformas da FATEC Araras Antônio Brambilla.

1.3 Responsável técnico

O projeto foi desenvolvido sob a supervisão de Gabriel Cardoso Schranck, membro do time responsável pelo gerenciamento do projeto e contato direto com o cliente. As suas principais responsabilidades foram:

- Supervisão da metodologia e cronograma, garantindo o cumprimento adequado de todas as etapas;
- Delegar tarefas aos desenvolvedores, garantindo que todas as funcionalidades fossem implementadas;

- Analisar as entregas incrementais, decidir pela aprovação ou solicitar ajustes, além de fornecer feedback importantes aos desenvolvedores;
- Realizar o contato direto com o cliente para esclarecimento de dúvidas sobre o processo.

1.4 Equipe de desenvolvimento

A equipe de desenvolvimento foi composta por quatro alunos, sob a supervisão do responsável técnico Gabriel. Os integrantes foram:

1.4.1 Bruno Henrique Guinerio

Membro da equipe responsável pela documentação do projeto. Suas principais responsabilidades foram:

- Trabalhar em conjunto com os desenvolvedores da aplicação para extrair informações sobre o funcionamento do software e transformá-las em documentação;
- Produzir diversos tipos de documentos, como manuais, guias, tutorias;
- Elaborar diagramas que descrevessem aspectos do sistema, incluindo comportamento e arquitetura. Os principais diagramas produzidos foram: diagrama de caso de uso, classe, sequência e atividade.

1.4.2 Kalliel Marcos Pinheiro

Membro da equipe responsável pelo desenvolvimento da lógica de negócio do projeto (backend). Suas principais responsabilidades foram:

- Construir fluxos de trabalhos alinhados às necessidades do cliente, seguindo boas práticas para facilitar o entendimento e manutenção do código.
- Garantir a segurança, integridade, manipulação e persistência dos dados;
- Aplicar boas práticas de desenvolvimento para otimizar o desempenho, resultando em uma experiência fluída ao usuário.

1.4.3 Pedro Rufino da Mata Neto

Membro da equipe responsável pela integração entre a interface do usuário e a lógica de negócio. Suas principais responsabilidades foram:

- Agrupar as informações de entrada do usuário e realizar requisições para o servidor;
- Receber as respostas do servidor e comunicá-las ao usuário de forma clara, fornecendo feedback sobre a próxima ação a ser tomada.

1.4.4 Stephan Mendes de Oliveira

Membro da equipe responsável pela criação da interface do usuário. Suas principais responsabilidades foram:

- Desenvolver as interfaces garantindo design responsivo e acessível em diferentes dispositivos;
- Criar telas seguindo boas práticas de design, assegurando navegação intuitiva e uma experiência positiva para o usuário.

1.5 Data de início e conclusão

O tempo de desenvolvimento do projeto é de quatro meses, a partir do início do segundo semestre (05 de agosto de 2025) até a data de entrega do projeto ao cliente, dia 05 de dezembro de 2025. Este tempo foi distribuído em instruções para o desenvolvimento do projeto, planejamento e codificação, ficando da seguinte maneira:

1. Durante o primeiro mês, foram fornecidas instruções sobre a utilização das tecnologias necessárias para o desenvolvimento da aplicação.
2. Durante o segundo mês foi definido os requisitos do projeto com o cliente e detalhes técnicos da aplicação
3. Durante os últimos dois meses foi realizado o desenvolvimento da aplicação, utilizando todo o conhecimento adquirido durante o primeiro mês e criando as funcionalidades que foram definidas durante o segundo mês.

1.6 Local de Execução

As atividades de desenvolvimento do projeto ocorreram na FATEC Araras Antônio Brambila e nas residências dos integrantes da equipe.

1.7 Contato

1.7.1 Bruno Henrique Guinerio

- Email: bruno.guinerio@fatec.sp.gov.br;

- WhatsApp: (19) 98286 – 9853.

1.7.2 Gabriel Cardoso Schranck

- Email: gabriel.schranck@fatec.sp.gov.br;
- WhatsApp: (19) 98349 – 2015.

1.7.3 Kalliel Marcos Pinheiro

- Email: kalliel.pinheiro@fatec.sp.gov.br;
- WhatsApp: (19) 99197 – 1960.

1.7.4 Pedro Rufino da Mata Neto

- Email: pedro.neto@fatec.sp.gov.br;
- WhatsApp: (19) 99867 – 2207.

1.7.5 Stephan Mendes de Oliveira

- Email: stephan.oliveira@fatec.sp.gov.br;
- WhatsApp: (19) 99894 – 3699.

2 DOCUMENTO DE VISÃO

2.1 Contextualização

O projeto foi desenvolvido com o objetivo de solucionar o problema de má gestão de doações de cestas básicas destinadas a famílias em situação de vulnerabilidade social, distribuídas por igrejas locais e pela Prefeitura Municipal de Mogi Guaçu. Atualmente, o controle de recebimento e distribuição dessas cestas básicas ocorre de maneira manual, o que resulta em diversas dificuldades de gestão, como falta de transparência, duplicidade de atendimentos e dificuldade de identificar famílias realmente necessitadas.

Essa ausência de sistematização prejudica tanto os beneficiários, que podem deixar de receber o auxílio, quanto os doares, que não conseguem acompanhar com clareza os impactos das ações realizadas.

Por isso, o projeto foi desenvolvido com o propósito de modernizar o processo de gerenciamento de doações por meio de um sistema que permite acompanhar métricas para obter insights sobre as doações, bem como o cadastro e a gestão de produtos, famílias, visitas, usuários e entregas.

2.2 Justificativa

O desenvolvimento deste sistema justificou-se pela necessidade de modernizar o processo de gestão das doações de cestas básicas destinadas a famílias em situação de vulnerabilidade social. Atualmente, a ausência de um sistema dificulta a organização das informações, gerando diversas falhas durante todo o processo.

Com a implementação de um sistema digital, será possível centralizar todos os dados, compartilhando os mesmos com as instituições religiosas e a prefeitura da cidade de maneira eficiente.

Além de todos os benefícios operacionais, o projeto também apresenta uma justificativa social. A implementação deste sistema contribuirá para o fortalecimento das ações solidárias para o combate à desigualdade, garantindo que as doações cheguem aos mais necessitados.

2.3 Objetivos gerais e específicos

Desenvolveu-se um sistema informatizado para o controle de doações de cestas básicas, com o intuito de tornar o processo de cadastro, controle e distribuição mais eficiente. Os principais objetivos alcançados com o sistema foram:

2.3.1 Objetivos organizacionais:

- Padronizar o processo de distribuições das doações entre as instituições religiosas participantes do projeto e a prefeitura;
- Centralizar as informações e reduzir o trabalho administrativo.

2.3.2 Objetivos sociais

- Garantir que as famílias em maior situação de vulnerabilidade social sejam priorizadas no recebimento das doações;
- Fortalecer a relação entre as instituições religiosas e o poder público.

2.3.3 Objetivos técnicos:

- Criar um módulo para cadastro e acompanhamento de famílias beneficiadas;
- Desenvolver um módulo para cadastro de produtos e criação de cestas básicas;
- Realizar a criação de uma tela para o registro e leitura de visitas destinadas às famílias;
- Integrar o sistema com diferentes usuários e níveis de permissões.

2.4 Escopo e delimitação

2.4.1 Entregas do projeto

- Levantamento de requisitos: Reunião realizada com o responsável pela intermediação entre a prefeitura e as instituições religiosas, Nilton Sacco, juntamente com a equipe de desenvolvimento do projeto;
- Desenvolvimento do esquema do banco de dados: Reuniões com a equipe de desenvolvimento para definir o esquema necessário para o armazenamento das entidades de dado do projeto;
- Interface Web, lógica de negócio e integração: Desenvolvimento da interface web, implementação da lógica de negócio e realização da integração entre os módulos do sistema. Ao final de cada etapa, a equipe validou as funcionalidades entregues para garantir a conformidade com os requisitos levantados.

2.4.2 Fora do escopo do projeto

- Integração com outros sistemas: O sistema não será integrado a plataformas municipais externas, restringindo-se ao ecossistema próprio de gerenciamento das doações de cestas básicas;
- Manutenção após o desenvolvimento: A equipe de desenvolvimento não será responsável pela manutenção após a entrega final do projeto. Essa atividade ficará a cargo de outra equipe.

2.5 Público alvo

O sistema se destinará a diferentes perfis de usuários, cada um com níveis específicos de acessos. O público alvo inclui agentes operacionais, responsável por lidar com os processos das doações e representantes de instituições religiosas.

2.5.1 Grupos de usuários

- Membros das instituições religiosas: Responsáveis pelo registro de famílias e de cestas;
- Voluntários: Usuários que realizarão as entregas das cestas, com níveis de acesso limitado;
- Famílias beneficiadas: Embora não possuam acesso direto ao sistema, constituem o público final a qual se beneficiará com o projeto.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Multimodalidade

Em um sistema digital há diversos meios de comunicações, como voz, vídeo, imagens, textos, entre outros, com o objetivo de transmitir uma ideia ou informação. Na nossa solução de um sistema de gerenciamento de doações de cestas básicas, a multimodalidade se manifesta em diferentes meios de entrada e saída de dados.

As principais maneiras pelas quais a multimodalidade se apresenta em nosso sistema são:

- Textos: Utilizado para transmitir informações instrucionais sobre cadastros e telas da aplicação;
- Emojis: Recursos visuais utilizados para construir significado e comunicar uma ideia;
- Elementos gráficos: Ícones, cores e símbolos utilizados para guiar o usuário, indicar ações e transmitir informações de forma intuitiva.
- Tecnologias envolvidas

Na solução proposta, as tecnologias foram selecionadas visando proporcional uma experiência multimodal moderna e acessível.

- Textos e Emojis: As informações instrucionais são exibidas por meio da interface desenvolvida em Flutter, utilizando widgets textuais e de formulários integrados à API;
- Elementos gráficos: Implementados a partir dos componentes visuais do Flutter baseado no Material Design, contribuindo para uma aplicação mais acessível, consistente e intuitiva.

3.2 Trabalhos Correlatos e Estado da Arte

A interface da aplicação foi inspirada no Material Design, sistema de design criado pela Google em 2014, com o objetivo de estabelecer um padrão de design unificado. Este sistema propõe o uso e posicionamento de elementos visuais coerentes, que simulam camadas físicas e promovem uma experiência mais intuitiva, fluida e acessível ao usuário.

Este padrão de design é amplamente adotado por grandes aplicações, como Gmail, YouTube, Spotify e WhatsApp, que utilizam os princípios do Material Design para construir interfaces consistentes e com alta fluidez. Devido à sua popularidade, o Material Design foi classificado como um padrão de referência e usabilidade em experiência do usuário.

No contexto deste projeto, o Material Design possibilitou uma interface moderna, responsiva e intuitiva, alinhada às melhores práticas de design. Deste modo, o nosso projeto integrará interfaces multimodais a partir de boas práticas do design, desempenho e foco na experiência do usuário.

4 PLANEJAMENTO DE PROJETO

4.1 Metodologia de Desenvolvimento

A metodologia adotada neste projeto foi baseada no modelo ágil Scrum, uma metodologia voltada para a gestão e o desenvolvimento iterativo de projetos de software. O Scrum organiza o trabalho em ciclos curtos e contínuos, denominados sprints, permitindo entregas incrementais do produto.

O Scrum opera sob os princípios da filosofia ágil, priorizando a entrega de valor, colaboração entre os membros da equipe e a adaptação às mudanças nos requisitos durante o desenvolvimento. Ao contrário de seguir um fluxo linear, como o desenvolvimento em cascata, o Scrum promove a evolução progressiva do sistema, permitindo revisões e melhorias a cada ciclo.

Todas as sprints foram planejadas com metas pré-definidas, tarefas específicas e prazos curtos, garantindo um progresso mensurável e que as funcionalidades sejam desenvolvidas de forma gradual. Ao final de cada Sprint foram realizadas reuniões entre os membros da equipe para monitorar o andamento das atividades, identificar empecilhos e redefinir as prioridades se necessário.

A adoção desta metodologia proporcionou maior flexibilidade e eficiência ao projeto, permitindo mudanças de forma rápida, conforme os requisitos. Além disso, favoreceiu a comunicação contínua entre os integrantes da equipe, promovendo um ambiente colaborativo.

4.2 Etapas do projeto

O projeto foi dividido em diversas etapas, permitindo o desenvolvimento das partes do sistema de maneira individual, separando as responsabilidades e facilitando a implementação de novas funcionalidades. Estas etapas foram:

4.2.1 Planejamento e prototipagem

Inicialmente foi realizado o planejamento do projeto, com a definição do escopo, requisitos e funcionalidades. Nesta fase foi desenvolvido o protótipo das principais telas da aplicação, o fluxo de navegação no Figma e o esquema do banco de dados. Além disso, foi realizada uma reunião com o cliente para alinhar e validar os requisitos do sistema. A partir disso, foram organizadas as tarefas no backlog e definido o cronograma das sprints.

4.2.2 Desenvolvimento das interfaces

Nesta etapa foram desenvolvidas as interfaces da aplicação, utilizando o framework Flutter e seguindo os princípios do Material Design. Durante esta etapa, foram implementados os componentes visuais e textuais, como formulários, ícones, botões, elementos de navegação, de acordo com o que foi proposto durante a etapa de prototipagem.

4.2.3 Desenvolvimento do Backend e banco de dados

Nesta fase foi desenvolvido o backend, responsável por lidar com a lógica de negócio, conforme os requisitos funcionais e não funcionais definidos na prototipagem. O backend foi integrado ao banco de dados PostgreSQL para armazenar os dados da aplicação. Como resultado, foram definidas as rotas da API.

4.2.4 Integração e comunicação entre os servidores

Após o desenvolvimento das camadas principais da aplicação, o backend e o frontend, foi realizada a integração entre elas, testando as rotas da API e a consistência entre ambos os servidores. Esta etapa garantiu que as interfaces da aplicação estejam em conformidade com o esperado no backend.

4.2.5 Documentação e Entrega

Ao final do desenvolvimento, foi elaborada a documentação técnica do sistema, incluindo os guias necessários para os usuários. Por conseguinte, o sistema será implementado em ambiente de produção, tornando-se disponível para utilização.

4.3 Cronograma de execução

Durante o desenvolvimento foram realizados diversos ciclos de trabalho, com o objetivo de promover entregas incrementais. Assim, a cada semana ocorreram avanços no desenvolvimento ou validações do sistema, acompanhados de eventuais reajustes conforme o feedback da equipe.

4.3.1 Primeira a sexta semana

Durante este período foi estudado as tecnologias que seriam utilizadas no projeto, a fim de adquirir os conhecimentos necessários para desenvolver a aplicação de maneira eficiente escalável.

4.3.2 Sétima semana

Na sétima semana foi realizada uma reunião com o cliente a fim de definir os objetivos da aplicação. Com base nas informações obtidas, foram discutidos as telas necessárias e o esquema do banco de dados.

4.3.3 Oitava semana

Para a oitava semana iniciou-se a construção de protótipos das primeiras telas que foram definidas na primeira semana de desenvolvimento, tanto para a versão web quanto para a versão mobile.

4.3.4 Nona semana

Na nona semana foram realizadas reuniões com a equipe para tratar dos protótipos de telas desenvolvidos, de modo a definir o padrão de interface do usuário que foi mantido nas versões finais. Além disso, foi iniciado a implementação da lógica de negócio, com a definição dos principais fluxos do sistema.

4.3.5 Décima semana

Na décima semana ocorreu a finalização das versões iniciais das telas em desenvolvimento, seguida do feedback da equipe. Foram realizados reajustes nas telas de acordo com as avaliações obtidas, com foco em promover melhorias na interface e na experiência do usuário.

4.3.6 Décima primeira semana

Durante a décima primeira semana foram conduzidas reuniões diárias com a equipe, com o objetivo de planejar as próximas etapas do projeto e definir prioridades.

4.3.7 Décima segunda semana

Para esta semana ocorreu a continuidade da implementação da lógica da aplicação e a realização da integração com o banco de dados, consolidando as funcionalidades desenvolvidas até o momento.

4.3.8 Décima terceira semana

Para a décima terceira semana de desenvolvimento ocorreu a construção do esquema de rotas da aplicação, permitindo a navegação entre as telas de forma fluida e organizada.

4.3.9 Décima quarta semana

Nesta semana continuou o desenvolvimento do esquema de rotas da aplicação, validando a fluidez da navegação da aplicação.

4.3.10 Décima quinta semana

Durante essa semana de desenvolvimento foi realizado a integração entre ambos os servidores da aplicação, permitindo a comunicação do cliente com o servidor.

4.3.11 Décima sexta semana

Para essa semana foi feito a revisão da aplicação, buscando possíveis pontos de falhas e aplicando as correções necessárias.

4.3.12 Décima sétima semana

Nesta semana foi desenvolvido os materiais necessários para a apresentação do projeto e além disso, foi realizado a prévia da apresentação do projeto.

4.3.13 Décima oitava semana

Na última semana, foi realizado a apresentação final do projeto, mostrando tudo o que foi criado durante esse período, as funcionalidades e os benefícios da aplicação.

4.4 Riscos e Estratégias de Mitigação

Durante o processo de desenvolvimento de um sistema é fundamental considerar a possibilidade de surgirem imprevistos que, se não previstos e tratados corretamente, podem resultar em atrasos na entrega do produto. Alguns riscos foram identificados no projeto e para evitar contratempos foram definidas estratégias de mitigação.

4.4.1 Alteração nos requisitos da aplicação

A alteração de requisitos durante o desenvolvimento de um produto é algo comum. Para contornar este risco, foram realizadas reuniões semanais com o cliente, a fim de coletar feedbacks e garantir que a aplicação estava de acordo com os requisitos estabelecidos. Além disso, a metodologia adotada permitiu adaptar o sistema de maneira simples, quando ocorreram alterações nos requisitos.

4.4.2 Atraso na entrega do sistema

Outro problema comum é o atraso na entrega do projeto, com a data prevista sendo alcançada antes da conclusão de todas as funcionalidades. Para mitigar este risco, foram realizadas reuniões semanais para ajustar o cronograma sempre que necessário.

4.4.3 Falta de comunicação da equipe

Durante o desenvolvimento de um projeto, é comum ocorrer falta de comunicação, o que pode levar ao desalinhamento de objetivos e à obtenção de resultados inesperados. A medida mais eficaz para mitigar esse risco foi promover reuniões semanais para discutir o progresso, as dificuldades e eventuais dúvidas da equipe.

4.4.4 Falta de comunicação com o cliente

A ausência de comunicação com o cliente pode causar diversos problemas, sendo o principal a entrega de uma aplicação diferente da esperada. Para mitigar este risco, a estratégia mais eficiente foi manter contato frequente com o cliente, validando os requisitos à medida que a aplicação era desenvolvida.

5 ESPECIFICAÇÕES TÉCNICAS

A definição das especificações técnicas é uma fase crucial para o desenvolvimento de um sistema. A partir desta definição é possível começar a desenvolver o projeto e suas funcionalidades. Ao início do projeto houve uma reunião com o cliente, a partir disso foi definido os requisitos.

5.1 Requisitos funcionais

5.1.1 Módulo de autenticação

1. O sistema permite que os usuários cadastrados realizem login informando o usuário e a senha.
2. O sistema valida as credencias fornecidas e permitir o acesso apenas a usuários autenticados.
3. O sistema permite que o usuário encerre a sessão (logout).
4. O sistema restringe o acesso às funcionalidades somente a usuários autenticados.

5.1.2 Módulo de Famílias

5. O sistema permite o cadastro de famílias pedindo os seguintes dados: nome, CPF e telefone do responsável, além de informações de endereço (CEP, rua, número, bairro, estado e comprovante de endereço).
6. O sistema permite o upload de comprovante de endereço (imagem).
7. O sistema permite a edição dos dados das famílias cadastradas.

8. O sistema permite a visualização das informações das famílias cadastradas.
9. O sistema exibe as famílias nas seguintes subdivisões: famílias ativas e aguardando visitas.
10. O sistema permite a pesquisa e filtragem de famílias por nome, endereço e CPF.

5.1.3 Módulo de Estoque

11. O sistema permite o cadastro de produtos pedindo as seguintes informações: nome do produto, estoque mínimo recomendado e quantidade do produto.
12. O sistema permite a edição dos dados dos produtos cadastrados.
13. O sistema permite a visualização das informações dos produtos cadastrados nas seguintes subdivisões: produtos ativos, estoque baixo, estoque alto, sem estoque e todos.
14. O sistema sinaliza quando o estoque de um produto estiver abaixo de um limite mínimo.
15. O sistema permite a pesquisa e filtragem de produtos por nome.

5.1.4 Módulo de Cestas Básicas

16. O sistema permite o cadastro de cestas básicas, utilizando os produtos cadastrados.
17. O sistema permite a edição dos dados das cestas básicas cadastradas.
18. O sistema permite a visualização das informações das cestas básicas cadastradas.

5.1.5 Módulo de Visitas:

19. O sistema permite o cadastro das visitas pedindo os seguintes dados: família, responsável pela visita, data, status e observações, se necessário.
20. O sistema permite a edição dos dados da visita.
21. O sistema permite a visualização das informações das visitas nas seguintes subdivisões: realizadas, pendentes, agendadas, canceladas e total de visitas.
22. O sistema permite a pesquisa e filtragem das visitas por nome, celular e CPF.

5.1.6 Módulo de Entregas

23. O sistema permite o cadastro das entregas de uma cesta a uma família pedindo os seguintes dados: a família que receberá, a data, o entregador e observações, se necessário.
24. O sistema permite a edição dos dados de uma entrega.
25. O sistema permite a visualização das informações das entregas nas seguintes subdivisões: pendentes, entregues, não entregues e total.
26. O sistema possui um botão que leva para a localização da casa da família no Google Maps.
27. O sistema permite a pesquisa e filtragem das entregas por família.

5.1.7 Módulo de Administração

28. O sistema restringe funcionalidades de acordo com o nível de permissão ao sistema do usuário.

29. O sistema permite o cadastro de funcionários pedindo os seguintes dados: e-mail, senha, confirmação da senha, nome, CPF, telefone e tipo de atividade.
30. O sistema permite a edição dos dados de um funcionário.
31. O sistema permite a visualização das informações de um funcionário nas seguintes subdivisões: coordenadores, entregadores, assistentes sociais e voluntários.
32. O sistema permite a pesquisa e filtragem das entregas por família.

5.2 Requisitos não funcionais

5.2.1 Usabilidade

1. O sistema apresenta uma interface intuitiva, responsiva e consistente, desenvolvida em Flutter utilizando princípios do Material Design.
2. O sistema exige o menor número possíveis de passos para a execução das tarefas, reduzindo a carga cognitiva do usuário.

5.2.2 Acessibilidade

3. O sistema apresenta elementos visuais (ícones, cores, botões), seguindo boa hierarquia visual e contraste adequado, facilitando a navegação por todos os usuários.
4. O sistema apresenta mensagens de feedback visual por descrições textuais que possam ser interpretadas por tecnologias assistivas.

5.2.3 Desempenho

5. O sistema responde operações simples em até 200ms e operações complexas em até 500ms, conforme os testes exibidos.
6. O sistema, durante o carregamento inicial não excede 3 segundos em condições normais de rede.

5.2.4 Escalabilidade

7. O sistema possui uma arquitetura modular do backend (FastAPI + SQLAlchemy) e frontend (Flutter), para permitir fácil expansão sem comprometer funcionalidades existentes.
8. O sistema suporta aumento de usuários simultâneos sem perda de performance.

5.2.5 Segurança da Informação

9. O sistema utiliza comunicação HTTPS com TLS, autenticação via JWT e controle de permissões por nível de acesso.
10. O sistema garante que cada usuário acesse apenas funcionalidades correspondente ao seu nível de permissão.

5.2.6 Conformidade com a LGPD

11. O sistema coleta apenas os dados mínimos necessários para o funcionamento da aplicação.
12. O sistema não compartilha dados pessoas com terceiros.

5.2.7 Manutenibilidade

13. O sistema segue arquitetura em camadas e versionamento semântico (GitHub), permitindo evolução e correções sem impacto em módulos independentes.
14. O sistema foi desenvolvido com baixo acoplamento e alta coesão, facilitando substituição e evolução de componentes isolados.

5.2.8 Confiabilidade

15. O sistema mantém funcionamento estável, garantindo consistência nos dados e taxas de sucesso alta nas requisições.
16. O sistema apresenta taxa de erro inferior a 1%.

5.2.9 Portabilidade

17. O sistema roda em ambiente Web e Mobile, suportada pelo Flutter, permitindo acesso multiplataformas.
18. O sistema garante consistência visual independentemente da resolução ou tamanho da tela do dispositivo.

5.2.10 Comunicação Cliente-Servidor

19. O sistema possui comunicação cliente-servidor por meio de API RESTful utilizando JSON e métodos HTTP adequados.
20. O sistema apresenta as respostas da API de maneira uniforme, facilitando a integração entre o cliente-servidor.

5.3 Interfaces Multimodais

O sistema conta com interfaces multimodais que possibilitam a interação entre o usuário e a aplicação por meio de diferentes canais de comunicação, combinando elementos visuais e textuais.

As interações entre o usuário e o sistema ocorre através das telas desenvolvidas em Flutter, com base no Material Design, garantindo ao usuário uma experiência acessível, consistente e responsiva.

Das diversas modalidades de comunicação presentes, as principais são:

5.3.1 Modalidade Visual

A interface gráfica é composta por diversos elementos gráficos (botões, menus, formulários, emojis, entre outros) que orientam os usuários em suas ações. O uso de cores segue o princípio de contraste e hierarquias visual, assim auxiliando na distinção entre as funções (exemplo: cadastrar família) e os estados do sistema (exemplo: sucesso, erro).

5.3.2 Modalidade Textual

As instruções ao usuário do sistema são exibidas em formato de texto, orientando o usuário em todas as funcionalidades da aplicação. As mensagens de feedbacks são apresentadas de forma clara e objetiva, indicando o resultado de cada operação e como prosseguir.

5.3.3 Arquitetura do Sistema

O projeto foi desenvolvido seguindo a arquitetura cliente-servidor, sendo composta pelo cliente, desenvolvido em Dart, utilizando o framework Flutter e o backend, desenvolvido em Python, utilizando FastAPI. A partir dessa divisão permite a independência entre a interface do usuário e a lógica de negócio, favorecendo manutenção e reuso de código.

A arquitetura é baseada em camadas lógicas, separando as responsabilidades em níveis distintos:

- Camada de Apresentação (Frontend): Responsável pela interface gráfica e interação com o usuário;
- Camada de Lógica de Negócio (Backend): Responsável pelo processamento das regras de negócios e comunicação com o banco de dados;
- Camada de Dados: Responsável pelo armazenamento e persistências das informações.

A comunicação entre o frontend e o backend da aplicação é feito via requisições HTTP, enviando e recebendo informações no formato JSON seguindo os padrões de uma RESTful api.

5.3.4 Arquitetura do Backend

O backend foi desenvolvido seguinte uma arquitetura modular em camadas, abordagem que promove a separação de responsabilidades, a facilidade de manutenção e a evolução incremental do sistema.

Nessa arquitetura, o código é organizado em módulos independentes, onde cada camada possui uma função específica e se comunica com as demais. Essa abordagem de arquitetura contribui com a extensão do sistema de maneira simples, sem afetar os demais módulos, garantindo um baixo acoplamento e alta coesão entre os componentes.

5.3.5 Arquitetura do Frontend

O frontend foi desenvolvido seguindo uma arquitetura modular de apresentação, que organiza a aplicação em camadas responsáveis pela exibição, navegação e controle da interface. Essa arquitetura que cada parte da aplicação mantenha uma função bem definida, promovendo a separação de responsabilidades, componentes reutilizáveis e alta manutenibilidade.

Essa arquitetura adota o princípio da composição de interfaces, em que as telas são constituídas de diversos componentes independentes integrados. Essa abordagem permite coerência visual, reduz redundância e facilita a extensão das interfaces sem afetar as demais partes do sistema.

5.4 Modelagem de Dados

A modelagem de dados do sistema foi elaborada com base no paradigma entidade-relacionamento, representando de forma estruturada as informações

manipuladas pela aplicação, como famílias, produtos, cestas, visitas, instituições e cestas básicas.

O modelo adota o padrão relacional, assegurando integridade referencial e consistências dos dados, sendo implementando por meio do ORM SQLAlchemy, que realiza o mapeamento das classes no Python para as tabelas no banco de dados PostgreSQL. Para o sistema há os seguintes modelos implementados:

5.4.1 Usuário

A entidade Usuário representa os agentes que interagem com o sistema, exercendo diferentes papéis, como coordenador, assistentes social, entregador e voluntário.

Seus campos são: id, login, email, password, account_type, family_id, institution_id, created, modified e active.

Além disso o usuário possui as seguintes relações:

- 1:N com Instituição: Todos os usuários se associam a uma instituição, assim como todas as instituições se associam a um ou mais usuários;
- N:N com Família: Todos os usuários se associam a uma ou mais famílias, assim como todas as famílias se associam a um ou mais usuários;
- 1:N com Visita e Entrega: Todos os usuários se associam a um ou várias visitas e entregas, conforme seu papel no sistema, assim como todas as visitas e entregas se associam a um usuário.

5.4.2 Família

A entidade Família representa os beneficiados pelo programa social.

Seus campos são: id, owner_id, family_size, addres, number, complement, zipcode, district, city, state, monthly_income, description, situation_type, membership_id, benefit_expiration_date, created, modified, active.

Além disso, a família possui as seguintes relações:

- N:1 com Instituição: Todas as famílias se associam a uma instituição, assim como todas as instituições se associam com uma ou mais famílias;
- N:N com Usuário: Todas as famílias se associam a um ou mais usuários, assim como todos os usuários se associam a uma ou mais famílias.

5.4.3 Instituição

A entidade Instituição representa a organização responsável por gerir as famílias, usuários e recursos do estoque.

Seus campos são: id, name, institutions_type, owner_id, created, modified e active.

Além disso, a instituição possui as seguintes relações:

- 1:N com Usuário, Família, Produto e Cesta Básica: Todas as instituições podem se associar a um ou mais registros dessas entidades, assim como esses registros se associam com uma instituição.

5.4.4 Produto

A entidade Produto representa os itens cadastrados no estoque de uma instituição, utilizado nas composições das cestas básicas.

Seus campos são: id, institution_id, name, sku, quantity, type_stock, created, modified e active.

Além disso, o produto possui as seguintes relações:

- N:1 com Instituição: Todos os produtos se associam a uma instituição, assim como todas as instituições se associam a um ou mais produtos;
- N:N com Cesta Básica: Todos os produtos se associam a uma ou mais cestas básicas, assim como todas as cestas básicas se associam a um ou mais produtos.

5.4.5 Cesta básica

A entidade Cesta Básica é a composição de produtos destinado às famílias beneficiadas pelo programa social.

Seus campos são: institution_id, family_id e products.

Além disso, a cesta básica possui as seguintes relações:

- N:1 com Instituição e Família: Todas as cestas básicas e famílias se associam a uma instituição, assim como todas as instituições se associam a um ou mais instituições e famílias;
- N:N com Produto: Todas as cestas básicas se associam com diversos produtos, assim como todos os produtos se associam a uma ou mais cestas básicas.

5.4.6 Visita

A entidade Visita representa o acompanhamento social realizado às famílias beneficiadas.

Seus principais campos são: id_family, day_visit, attempt_count e vist_date.

Além disso, a visita possui as seguintes relações:

- N:1 com Família e Usuário: Toda as visitas se associam a uma família e a um usuário, assim como todos os usuários se associam com uma ou mais famílias.

5.5 Padrões e Protocolos de Comunicação

Para a comunicação entre o cliente e servidor da aplicação é utilizado o protocolo HTTPS (Hypertext Transfer Protocol Secure), versão segura do HTTP, utilizando TLS (Transport Layer Security) para criptografam os dados trafegados.

5.5.1 Protocolo HTTPS

O HTTPS assegura três propriedades fundamentais para uma conexão segura:

- Integridade: Garante que a mensagem não seja alterada durante o transporte;
- Confidencialidade: Impede a leitura dos dados por terceiros;
- Autenticação: Valida a identidade do servidor e do cliente.

5.5.2 Métodos de requisição HTTP

As interações entre o cliente e servidor seguem os protocolos HTTP, que definem a ação a ser executada sobre um recurso. Os protocolos utilizados na aplicação são:

- GET: Utilizado para recuperar dados de um recurso;

- POST: Utilizado para enviar dados ao servidor, comumente utilizado para a criação de um novo recurso;
- PUT: Utilizado para atualizar completamente um recurso;
- PATCH: Utilizado para atualizar parcialmente um recurso;
- DELETE: Utilizado para remover um recurso.

5.5.3 Padrões RESTful

A aplicação adota o padrão RESTful, em que cada recurso da aplicação é identificado por uma URL única e as requisições são feitas utilizando os verbos HTTP e a URL única. Neste formato de comunicação, é enviado e recebido dados no formato JSON.

5.6 Segurança da Informação e LGPD

A aplicação foi desenvolvida em conformidade com a Lei Geral De Proteção de Dados (Lei nº 13.709/2018), garantindo a proteção, privacidade e integridade dos dados tratados pelo sistema.

5.6.1 Práticas Adotadas

O sistema adota práticas que garantem a segurança das informações tratadas. As práticas adotadas são:

- Confidencialidade: A aplicação restringe o acesso apenas a usuários autenticados, com diferentes níveis de permissões, restringindo a visualização aos dados sensíveis;

- Integridade: Os dados são trafegados utilizando protocolos de comunicações seguros, assegurando a integridade dos dados;
- Autenticação: O sistema utiliza validação de autenticidade utilizando token JWT, assegurando que apenas usuários legítimos autenticados tenham acessos aos dados.

5.6.2 Proteção de Dados Pessoais

O sistema coleta apenas os dados necessários para executar as suas funcionalidades, de acordo com os princípios da finalidade específica e minimização de dados previstos na LGPD.

Os dados pessoas armazenados são utilizados exclusivamente para identificação dos beneficiários e registros administrativos. Nenhuma informação é compartilhada com terceiros.

6 DESENVOLVIMENTO

6.1 Etapas de Implementação

A implementação do sistema foi conduzida de forma incremental, seguindo os princípios da metodologia ágil Scrum. O desenvolvido ocorreu em ciclos semanais, permitindo ajustes e validações constantes conforme o feedback da equipe e do cliente.

As etapas a seguir descrevem o processo técnico adotado desde a configuração do projeto até a conclusão da aplicação.

6.1.1 Definição de Requisitos e Configuração do Ambiente

Nesta fase foram levantados os requisitos funcionais e não funcionais do sistema em conjunto com o cliente, definindo o escopo e as principais funcionalidades.

Com base nos requisitos foram selecionadas as tecnologias que melhor atenderiam às necessidades do projeto, sendo elas: Flutter, FastAPI, SQLAlchemy e PostgreSQL.

Além disso foi configurado o ambiente de desenvolvimento:

- Instalação do Flutter e de suas dependências;
- Criação de um ambiente virtual em Python para a utilização com a FastAPI;
- Criação do banco de dados e a definição dos modelos iniciais da aplicação;
- Criação do repositório no GitHub para o versionamento do código.

6.1.2 Prototipagem

Posteriormente após a definição das tecnologias e dos modelos de dados, foram elaborados os protótipos iniciais de interface no Figma, alinhado à estrutura do banco de dados. Essas interfaces representam as versões iniciais do sistema, passando por reajustes conforme o refinamento dos requisitos durante o desenvolvimento.

6.1.3 Criação da estrutura do projeto

Com os protótipos de interfaces definidos foi conduzido o desenvolvimento da estrutura de pasta do projeto, organizada de forma modular e em camadas, facilitando a manutenção e expansão futura do sistema. Além disso, também foi definido os padrões e convenções que seriam seguidos durante o desenvolvimento.

6.1.4 Implementação das interfaces

As implementações das interfaces foram conduzidas com base nos protótipos desenvolvidos no Figma. Essas telas foram implementadas utilizando o Flutter, seguindo os princípios do Material Design, garantindo a consistência visual, responsividade e acessibilidade.

As telas implementadas nesta etapa foram: tela de autenticação, dashboard, famílias, estoque, cestas, visitas, entregas e membros da equipe.

6.1.5 Implementação da API e Banco de dados

O backend foi implementado utilizando o framework FastAPI, escolhido por sua facilidade na criação de APIs RESTful. As atividades realizadas nessa etapa incluíram:

- Criação das rotas para os diversos recursos da aplicação;
- Implementação de autenticação utilizando JWT;
- Integração com o banco de dados PostgreSQL por meio da ORM SQLAlchemy, permitindo o mapeamento de objetos Python para as tabelas no banco de dados;
- Tratamento de exceções e padronização das respostas da API.
- Integração entre o Frontend e o Backend

Com o backend e o frontend desenvolvido, foi realizado a integração entre as camadas da aplicação. O frontend passou a consumir os serviços da API através de requisições HTTP, enviando e recebendo mensagens no formato JSON.

6.2 Ferramentas e Linguagens utilizadas

Com base nos requisitos da aplicação, foram escolhidas as ferramentas e linguagens que melhor solucionariam o problema. Essa escolha foi essencial para garantir a escalabilidade no desenvolvimento. As tecnologias adotadas foram:

- Flutter: Framework baseado em Dart para o desenvolvimento de interfaces para aplicações mobile e web, escolhido por sua alta eficiência;
- FastAPI: Framework baseado em Python moderno e de alta performance, selecionado pela facilidade de implementação e escalabilidade na criação de APIs RESTFul;

- SQLAlchemy: Ferramenta de mapeamento objeto-relacional (ORM) utilizada pela facilidade de integração com o FastAPI e pela eficiência no gerenciamento entre o Python e o banco de dados;
- PostgreSQL: Sistema gerenciador de banco de dados (SGBD), escolhido por sua robustez, segurança e flexibilidade, garantindo confiabilidade na persistência dos dados.

6.3 Integração de Módulos Multimodais

A integração dos módulos multimodais teve como objetivo garantir que os diferentes modos de comunicação funcionassem de maneira coesa e consistente da aplicação.

Durante o desenvolvimento a multimodalidade foi tratada como parte essencial da experiência do usuário, influenciando no design das interfaces, fluxos e as formas de retorno do sistema. Cada modalidade tem um objetivo dentro da aplicação:

- Integração dos elementos textuais e emojis: Os componentes textuais foram integrados na aplicação para fornecer instruções clara dos processos e mensagens de feedack a respeito das ações executadas;
- Integração dos elementos gráficos e símbolos: Os elementos gráficos foram implementados a partir do Material Design, com o objetivo de padronização e acessibilidade ao usuário. Além disso, os símbolos foram integrados com o objetivo de representar uma ação ou estado.

Para promover coerência entre os meios multimodais, foram seguidos os princípios do Material Design, garantindo uma unidade comunicativa em todas as telas da aplicação. Dessa maneira, os meios multimodais trabalham de forma sincronizada, resultando em uma interface mais acessível e compreensível, influenciando positivamente na experiência do usuário.

6.4 Versionamento e Controle de Mudanças

O controle de versões foi essencial durante o projeto do sistema, garantindo organização, rastreabilidade e colaboração entre os membros da equipe. Essa prática possibilitou que o projeto fosse realizado de maneira incremental, evitando conflitos entre versão e perdas de código.

O versionamento foi realizado por meio do Git, hospedando o repositório no GitHub sob o controle do líder técnico da equipe, em que todas as alterações foram devidamente registradas de maneira semântica, promovendo a rastreabilidade das versões do código.

Essa abordagem garantiu um histórico detalhado com todas as versões da aplicação, promovendo um ambiente de trabalho colaborativo, seguro e transparente.

7 TESTES E VALIDAÇÃO

Os testes automatizados são fundamentais para garantir a segurança e a confiabilidade da aplicação, assegurando seu funcionamento a cada nova funcionalidade implementada.

No projeto, foram realizados testes automatizados nas rotas da API, fornecendo valores de entrada e comparando as saídas com os resultados esperados, a fim de validar o comportamento do sistema conforme os requisitos definidos.

7.1 Plano de Testes

O plano de testes tem como objetivo assegurar o funcionamento do sistema e a conformidade com os requisitos estabelecidos. Para isso, foram realizados testes no backend, desenvolvido em FastAPI, validando as rotas responsáveis pelas operações da aplicação.

Os testes foram planejados para ocorrer de forma incremental, acompanhando a evolução do sistema. Para todas as rotas que foram implementados, eram elaborados casos de testes correspondentes, assegurando que o comportamento do código permanecesse estável e que regressões fossem evitadas ao longo do desenvolvimento.

7.1.1 Ferramentas utilizadas

Os testes foram executados em um ambiente local, configurado com a API e o banco de dados PostgreSQL. Para a automação e a execução dos testes, foi utilizada a ferramenta Pytest e Unitest, que permitiu comparar as respostas

retornada pelas rotas com os resultados esperados, garantindo exatidão durante o processo de validação.

7.1.2 Conclusões dos testes

Os planejamentos de testes contribuíram para a detecção de falhas precocemente e assegurar a estabilidade da API, garantindo maior confiabilidade no funcionamento do backend.

Os testes promoveram a evolução contínua do sistema com segurança, facilitando a adição de novas funcionalidades e integrações com o frontend.

7.2 Casos de Teste e Critérios de Aceitação

Os casos de testes foram desenvolvidos com o propósito de validar os comportamentos da API, garantindo que cada funcionalidade retornasse os valores adequados conforme os requisitos estabelecidos.

As execuções dos testes basearam-se na comparação entre as saídas obtidas e os valores esperados, permitindo identificar falhas e corrigi-las.

7.2.1 Casos de testes

Todos os casos de testes foram definidos com base nos módulos da aplicação. Para cada rota do sistema, foram simuladas diferentes situações de uso, contemplando diversos cenários e garantindo o seu sucesso.

Os casos de testes foram:

- Verificação de login, gerando um JWT para credencias corretas e rejeitando para credenciais incorretas;
- Verificação de logout, removendo o JWT salvo, garantindo o encerramento seguro da aplicação;
- Criação, edição e leitura para os recursos de família, estoque, cesta, visita, entrega e equipe.

7.2.2 Critérios de aceitação

Os critérios de aceitação foram definidos para determinar o sucesso ou a falha de cada teste. A falha ou sucesso da aplicação são definidas com os seguintes códigos de status:

- A aplicação deve retornar o código 200 para leitura e atualizações bem-sucedidas;
- O retorno deve ser 201 quando um novo recurso for criado;
- Em casos de falha, o sistema deve retornar 204 ou um erro acima de 400, acompanhado de uma mensagem descritiva.

Além disso, nenhum dado inválido deve ser persistido no banco de dados, ele deve ser rejeitado pela aplicação antes de começar o seu tratamento. Para todas as respostas, de sucesso ou falha, deve vir o status e uma mensagem, estruturados em formato JSON.

7.3 Resultado dos Testes Unitários e Integrados

Os testes foram executados com o objetivo de validar o funcionamento dos módulos da aplicação e a integração entre eles. Durante o processo, buscou-se assegurar que cada rota da API comportasse conforme o esperado diante de diferentes cenários.

Os testes unitários tiverem focos de funções que tratam das regras de negócio da aplicação, garantindo que cada componente operasse de forma previsível. Por outro lado, os testes integrados verificaram a comunicação entre os módulos, testando fluxos completos, como operações no banco de dados.

7.3.1 Execução dos Testes

Todos os testes foram executados automaticamente através do Pytest e Unitest, em um ambiente controlado e isolado, com um banco de dados em modo de teste. Além disso, a cada nova funcionalidade criada, era reexecutado os tentes, garantindo que não houve regressão.

7.3.2 Resultados Obtidos

Todas as rotas da API foram testadas, obtendo uma taxa de cobertura de teste altas. As margens de sucesso para os testes foram de 100%, dessa forma, todas as rotas se comportando como esperado.

Durante o desenvolvimento dos testes e evolução do sistema foram identificadas diversas falhas nos testes, deste modo, corrigindo a falha e reexecutando os testes, obtendo sucesso.

7.4 Testes de Desempenho e Usabilidade

Os testes de desempenho e usabilidade foram realizados com o objetivo de avaliar a estabilidade da aplicação com o sistema sob demanda, recebendo muitas requisições simultaneamente. Esses testes buscaram identificar possíveis gargalos

de performance, lentidão e aspectos que poderiam impactar negativamente a experiência do usuário.

7.4.1 Testes de Desempenho

Os testes de desempenho tiveram como foco o tempo de resposta da API, capacidade de processamento sob demanda e a eficiência das consultas no banco de dados.

Para esses testes foram utilizados scripts no Postman, simulando múltiplas requisições simultaneamente. Além disso, foi avaliado o tempo médio de respostas para diferentes métodos de requisição, obtendo resultados satisfatórios, abaixo de 200ms para requisições simples e abaixo de 500ms para operações que envolviam consultas ao banco de dados.

7.5 Laudos Técnicos e Ensaios de Validação

O seguinte laudo técnico tem como finalidade apresentar a análise sobre os testes realizados no sistema, abrangendo os aspectos funcionais, desempenho e usabilidade, com o objetivo de comprovar sua conformidade com os requisitos estabelecidos.

7.5.1 Metodologia

Todos os testes foram conduzidos em um ambiente local, simulando condições reais de operação. Os testes foram executados conforme descrito abaixo:

- Testes unitários com Pytest e Unitest: Validação das rotas da API, testando a integrações com diferentes camadas do sistema e banco de dados;
- Múltiplas requisições simultâneas: Scripts em Postman para simular requisições paralelas, avaliando o comportamento da aplicação sob demanda.

7.5.2 Resultados Obtidos

Situação de teste	Tempo médio (ms)	Limite aceitável (ms)	Status
Consultas simples (GET)	170	≤ 200	Sucesso
Consulta com dados (POST, PUT e PATCH)	390	≤ 500	Sucesso
100 requisições simultâneas	520	≤ 1000	Sucesso

7.5.3 Consumo de Recursos

Recurso	Uso médio	Limite aceitável	Status
CPU	32%	≤ 60%	Sucesso

Memória RAM	2.8	≤ 8 GB	Sucesso

7.5.4 Análise de Resultados

Os resultados dos testes demonstram que a API apresenta desempenho estável em confiável em diferentes condições de uso. Inclusive, o desempenho do sistema sob de manda se mostrou satisfatório, simulando condições reais de uso.

Todas as respostas mantiveram-se abaixo do limite de latência propostas e com baixo consumo dos recursos de hardware.

Com base nos resultados, conclui-se que o sistema se encontra validado, atendendo os critérios de desempenho, estabilidade propostos. Com isso, conclui-se que o sistema está apto para um ambiente de produção, garantindo confiabilidade e eficiência operacional.

8 RESULTADOS E DISCUSSÕES

Ao fim do processo de desenvolvimento do sistema, foram implementadas todas as funcionalidades propostas, garantindo o cumprimento dos requisitos funcionais e não funcionais estabelecidos. Nesta seção serão apresentados as funcionalidades implementadas, impactos, limitações e melhorias futuras.

8.1 Funcionalidades Implementadas

Durante o desenvolvimento, foram implementadas todas as funcionalidades previstas no escopo do projeto, garantindo o funcionamento completo do sistema. As funcionalidades desenvolvidas foram:

- Autenticação de usuários com token JWT, garantindo acesso a apenas usuários autorizados na aplicação, promovendo segurança;
- Cadastro edição e listagem para os módulos de família, estoque, cesta básica, visita, entrega;
- Cadastro de funcionário, edição e listagem, com diferentes níveis de permissões diferentes;
- Restrição de acesso às funcionalidades de acordo com o nível de permissão do usuário.

Para todas as funcionalidades desenvolvidas, foi desenvolvida sua interface utilizando o Flutter, a sua lógica utilizando FastAPI e a integração de ambas, garantindo a comunicação entre as partes e sua funcionalidade.

8.2 Indicadores de Desempenho

Os indicadores de desempenho foram avaliados durante os testes da aplicação, medindo o tempo de resposta, o consumo de recursos e a conformidade das respostas com o esperado. Para todos esses casos, foram obtidos resultados satisfatórios, conforme apresentado na seção de testes e validação do dossiê.

8.3 Benefícios e Impactos esperados

A implementação do sistema trouxe diversos benefícios, sendo os principais e mais relevantes:

- Automatização de processos, reduzindo tempo de execução manual e minimizando erros;
- Padronização do acesso à informação, garantindo que as diversas instituições religiosas e a prefeitura de Mogi Guaçu tenham acessos aos mesmos dados de maneira uniforme;
- Diminuição de fraude do benefício, garantindo que somente as famílias em situação de vulnerabilidade obtenham o benefício de maneira controlada;
- Escalabilidade, permitindo a expansão do sistema conforme o benefício for se expandindo.

8.4 Limitações Identificadas

Apesar dos diversos resultados positivos, algumas limitações foram identificadas durante o desenvolvimento e os testes:

- Não há, até o momento, mecanismo de cache para otimizar requisições repetitivas;
- Ausência de uma camada de monitoramento de logs automatizada.

Essas limitações não comprometem o funcionamento atual do sistema, mas representam pontos de atenção para futuras versões.

8.5 Sugestões de Melhorias Futuras

Com base na análise de desempenho, nos resultados dos testes e limitações identificadas, são sugeridas as seguintes melhorias futuras:

- Implementação de cache em requisições para reduzir o tempo de respostas em consultas repetitivas;
- Integração de logs em tempo real, permitindo alertas para casos de falha;
- Ampliação dos testes automatizadas, incluindo para casos de falhas de serviços externos.

A adoção dessas melhorias poderá melhorar o nível do sistema, tornando-o mais robusto, escalável e preparado para ambientes de alta disponibilidade.

9 CONSIDERAÇÕES FINAIS

9.1 Conclusões Técnicas

O desenvolvimento do sistema demonstrou ser uma solução eficiente e moderna para o objetivo proposto.

A arquitetura adotada permitiu uma integração fluida entre as camadas da aplicação, garantindo bom desempenho e segurança nas execuções das operações.

As funcionalidades implementadas, conforme definidas nos requisitos estabelecidos, mostraram-se adequadas ao domínio da aplicação.

De maneira geral, os resultados obtidos validaram a viabilidade técnica do sistema, que pode ser facilmente expandido ou adaptado conforme a ampliação do programa social ou a sua aplicação em outras regiões.

9.2 Lições Aprendidas

Durante o processo de desenvolvimento do sistema, foram adquiridos diversos conhecimentos técnicos e práticos. O uso integrado do Flutter e do FastAPI proporcionou um aprofundamento no domínio de ambas as tecnologias, além de reforçar as boas práticas na construção de APIs REST, nos aspectos de segurança e na modelagem de banco de dados relacional eficiente, com consultas eficientes.

Além disso, evidenciou-se a importância do planejamento colaborativo entre as entidades envolvidas no projeto, as instituições religiosas e prefeitura de Mogi Guaçu, ressaltando a necessidade de uma comunicação clara e objetiva entre os participantes.

Outro aprendizado importante foi a compreensão de que sistemas voltado ao serviço público devem priorizar a simplicidade na usabilidade e transparência nos registros, fatores essenciais para promover a confiança e a segurança dos usuários.

9.3 Recomendações

Considerando a finalização do projeto e os resultados obtidos, as recomendações relacionam-se em aspectos relacionados à gestão, continuidade operacional e uso responsável do sistema:

- Garantir que o sistema seja mantido em produção, sendo geridos por pessoas de confianças, assegurando a atualização dos dados;
- Realizar avaliações periódicas de impacto social, para mensurar a efetividade do uso do sistema na gestão de doações;
- Promover a colaboração contínua entre as instituições religiosas e a prefeitura, assegurando alinhamento nas ações de gestão e distribuições de doações.

REFERÊNCIAS

2 SOMMERVILLE, Ian. **Engenharia de Software.** 9ª ed. Louveira: Pearson Education, 2011.

3 PRESSMAN, Roger. **Engenharia de Software - Uma Abordagem Profissional.** 8ª ed. Porto Aleagre: AMGH, 2016.

3 **LGPD na Prática: Princípios, Aplicações e Conformidade.** Clavis – Segurança da Informação, 2024. Disponível em: https://clavis.com.br/blog/lgpd-lei-geral-de-protecao-de-dados/. Acesso em: 01 nov. 2025.

4 **FASTAPI framework**. FastAPI, s.d. Disponível em: https://fastapi.tiangolo.com/. Acesso em: 01 nov. 2025

5 **FLUTTER Documentation.** Flutter Documentation, s.d. Disponível em: https://flutter-website-staging.firebaseapp.com/docs/. Acesso em: 03 nov. 2025

6 **POSTGRESQL Documentation.** PostgreSQL Documentation, s.d. Disponível em: https://www.postgresql.org/docs/. Acesso em: 07 nov. 2025.

7 **SQLALCHEMY 2.0 Documentation.** SQLAlchemy, 2025. Disponível em: https://docs.sqlalchemy.org/. Acesso em: 11 nov. 2025

8 **MATERIAL DESIGN.** Material Design – Google’s latest open source design system, s.d. Disponível em: https://m3.material.io/. Acesso em: 16 nov. 2025.

9 **MOGI Guaçu Prefeitura Governo de Trabalho**. Prefeitura de Mogi Guaçu, 2025. Disponível em: https://mogiguacu.sp.gov.br/. Acesso em: 21 nov. 2025.

GLOSSÁRIO

API - Conjunto de regras e definições que permitem a comunicação entre diferentes sistemas ou componentes, possibilitando a interação entre o frontend e o backend.

Backend - Camada responsável pela lógica de negócio da aplicação, incluindo processamento de dados, regras internas e comunicação com o banco de dados.

Endpoint - Endereço específico dentro de uma API que representa uma funcionalidade ou recurso acessível pelo cliente.

FastAPI - Framework moderno e de alta performance escrito em Python, utilizado para desenvolvimento de APIs rápidas, seguras e de fácil manutenção.

Flutter - Framework desenvolvido pelo Google para criação de interfaces modernas e responsivas para aplicações web**,** mobile e desktop.

Frontend - Camada visual da aplicação, responsável pela interface e pela interação direta com o usuário.

JWT - Token de segurança utilizado para autenticação e autorização, permitindo validar identidades de forma segura em sistemas distribuídos.

LGPD - Legislação brasileira que regula o tratamento de dados pessoais, garantindo direitos aos usuários e impondo obrigações a quem coleta ou processa essas informações.

Material Design - Sistema de design criado pelo Google que define padrões visuais e comportamentais para interfaces, promovendo consistência, acessibilidade e boa experiência do usuário.

ORM - Técnica que converte dados de tabelas relacionais em objetos de programação, facilitando o acesso, manipulação e persistência de dados sem escrever SQL manualmente.

PostgreSQL - Sistema de gerenciamento de banco de dados relacional conhecido pela robustez, segurança e conformidade com padrões SQL.

RESTful - Estilo de arquitetura para comunicação entre cliente e servidor, baseado em recursos, métodos HTTP e respostas padronizadas.

SQLAlchemy - Ferramenta ORM usada em aplicações Python para modelagem, consulta e gerenciamento de bancos de dados relacionais de forma eficiente e estruturada.