

SOS COMIDA

Documento de Testes

1. Visão Geral

Este documento descreve o conjunto de testes realizados no Sistema SOS Comida, plataforma voltada à intermediação de doações e apoios destinados a pessoas em situação de vulnerabilidade social. São apresentados os principais tipos de testes aplicados, a estratégia adotada, os artefatos produzidos e os resultados obtidos, de modo a evidenciar o nível de qualidade alcançado pelo sistema.

Ao longo do documento, são contemplados desde testes em componentes isolados (como modelos de dados e rotas de aplicação) até avaliações de fluxos completos de uso, alinhados às histórias de usuário definidas para o projeto. Dessa forma, busca-se oferecer uma visão integrada da confiabilidade, da robustez e da aderência do sistema aos requisitos especificados.

1.1. Escopo

O escopo deste relatório abrange os *testes unitários*, *testes de integração* e *testes funcionais* realizados no Sistema SOS Comida. Tais testes contemplam, entre outros aspectos:

- Modelos de dados e seus relacionamentos;
- Rotas e endpoints da aplicação;
- Mecanismos de autenticação e autorização;
- Regras de negócio associadas a campanhas, denúncias, delegações e doações;
- Validações de campos e consistência de informações;
- Fluxos completos de uso, baseados nas histórias de usuário definidas para o sistema.

1.2. Objetivo

O objetivo deste relatório é documentar de forma sistemática o processo de teste do Sistema SOS Comida, registrando:

- A estratégia de testes adotada;

- Os tipos de testes executados
- Os principais casos de testes considerados;
- Os resultados obtidos e as não conformidades identificadas;
- Os ajustes propostos ou realizados na aplicação e na documentação.

Com isso, busca-se fornecer evidências de qualidade do software desenvolvido, bem como apoiar a manutenção e a evolução futura do sistema, servindo como referência para a equipe de desenvolvimento, para a disciplina Soluções Computacionais e para eventuais revisões técnicas do projeto.

2. Tecnologias Utilizadas

A implementação e a execução dos testes do Sistema SOS Comida foram realizadas com o apoio de um conjunto de ferramentas e bibliotecas amplamente utilizadas no ecossistema Python para garantia de qualidade de software. O uso combinado desses recursos permitiu estruturar uma suíte de testes automatizada, reproduzível e de fácil manutenção, alinhada às boas práticas de desenvolvimento.

As principais tecnologias empregadas foram:

- **pytest** – Framework principal de testes utilizado no projeto. Possibilita a escrita de casos de teste de forma simples e expressiva, com suporte a *fixtures*, organização modular e relatórios detalhados de execução.
- **pytest-cov** – Extensão do *pytest* utilizada para geração de relatórios de cobertura de código, permitindo mensurar quais partes da aplicação estão devidamente exercitadas pelos testes e identificar trechos que ainda necessitam de validação.
- **unittest.mock** – Biblioteca da biblioteca padrão do Python empregada para criação de *mocks* e *stubs*, possibilitando simular componentes externos, dependências e integrações, de modo a isolar o comportamento das unidades em teste.
- **SQLite** – Banco de dados relacional utilizado em modo em memória durante os testes, garantindo maior desempenho e permitindo que os cenários sejam executados de forma isolada, sem impacto em bancos de dados de desenvolvimento ou produção.

- **Faker** – Biblioteca destinada à geração de dados fictícios realistas (como nomes, e-mails, endereços), utilizada para compor cenários de teste mais próximos de situações reais, sem expor dados sensíveis de usuários.
- **Extensões para Flask** – Ferramentas que facilitam a criação de clientes de teste para a aplicação web, possibilitando a simulação de requisições HTTP e a verificação do comportamento de rotas, autenticação e respostas da API.

A adoção dessas tecnologias contribuiu para a construção de um ambiente de testes robusto, favorecendo a automação, a rastreabilidade dos resultados e a evolução contínua da base de testes conforme novas funcionalidades são incorporadas ao Sistema SOS Comida.

3. Estrutura de testes

A estrutura de testes do Sistema SOS Comida foi organizada de forma a refletir a própria arquitetura da aplicação, favorecendo a legibilidade, a rastreabilidade e a manutenção do código. Os testes foram separados por módulos e funcionalidades, o que permite identificar rapidamente quais partes do sistema estão sendo avaliadas e facilita a inclusão de novos cenários à medida que o projeto evolui.

3.1. Organização dos Artefatos de Teste

Os arquivos de testes seguem a mesma divisão lógica dos módulos da aplicação. Assim, para cada componente principal (por exemplo, modelos, rotas, serviços e regras de negócio) existe um arquivo de teste correspondente, localizado em uma estrutura semelhante a:

- testes/test_models.py – Testes dos modelos de dados;
- testes/test_routes.py – Testes de rotas e endpoints;
- testes/test_services.py – Testes de serviços e regras de negócio;
- testes/test_validations.py – Testes de validações e funções auxiliares.

Essa abordagem modular facilita a localização de falhas, a refatoração de trechos específicos e a leitura por novos integrantes da equipe, que conseguem associar de forma direta cada artefato de teste ao módulo funcional do sistema.

3.2. Uso de Fixtures e Dados de Teste

Para evitar duplicação de código e padronizar a preparação do ambiente de testes, foram utilizadas fixtures do pytest, centralizadas em arquivos de configuração (como `conftest.py`). Essas fixtures são responsáveis por:

- Criar e configurar o cliente de teste da aplicação;
- Inicializar um banco de dados em memória para cada suíte de teste;
- Provisionar usuários autenticados (usuário padrão, moderador, instituição parceira);
- Gerar dados fictícios por meio da biblioteca Faker, simulando cenários próximos ao uso real.

O uso de fixtures garante que cada caso de teste seja executado em um contexto controlado e previsível, reduzindo efeitos colaterais entre testes e aumentando a confiabilidade dos resultados.

3.3. Critérios de Organização e Manutenção

Além da separação por módulo, foram adotados padrões de nomeação para funções de teste (por exemplo, `test_criar_usuario_valido`, `test_login_credenciais_invalidas`), o que torna o relatório de execução mais claro e facilita a identificação do propósito de cada caso.

A estrutura proposta também favorece a manutenção contínua: novos testes podem ser adicionados nos arquivos existentes, mantendo a coesão por funcionalidade, enquanto testes obsoletos podem ser atualizados ou removidos sem comprometer o restante da suíte.

4. Testes de Modelos

Os testes de modelo têm como objetivo verificar se as entidades centrais do Sistema SOS Comida estão sendo criadas, armazenadas e manipuladas de forma correta, garantindo a integridade dos dados e o cumprimento das regras de negócio associadas.

Foram contemplados, principalmente, os modelos *Usuário*, *Campanha* e *Delegação*, por serem responsáveis pela maior parte das interações do sistema, como

cadastro de pessoas e instituições, registro de campanhas e distribuição de solicitações entre órgãos parceiros.

4.1. Modelo Usuário

Os testes do modelo Usuário verificam a criação, validação e comportamento dos diferentes perfis presentes na plataforma, incluindo usuários comuns, instituições e moderadores.

Entre os aspectos avaliados, destacam-se:

- Criação de usuários com todos os campos obrigatórios preenchidos corretamente;
- Diferenciação entre tipos de usuário (receptor, doador, moderador, órgão parceiro);
- Validação de atributos sensíveis, como CPF, e-mail e telefone;
- Tratamento de estados especiais, como revogação de conta;
- Suporte a mecanismos adicionais de segurança, como autenticação em dois fatores (2fa), quando aplicável.

Esses testes asseguram que apenas usuários válidos sejam persistidos no banco de dados e que o modelo responda adequadamente a mudanças de estado ao longo do ciclo de vida da conta.

4.2. Modelo Campanha

Os testes do modelo Campanha validam a criação, os relacionamentos e o comportamento das campanhas cadastradas no sistema. São avaliados, principalmente:

- Criação de campanhas com dados obrigatórios (título, descrição, localização, tipo de ajuda);
- Associação correta das campanhas aos seus solicitantes (usuários ou instituições);
- Relacionamento com delegações realizadas para órgãos parceiros;
- Controle de status (pendente, ativa, concluída, removida etc.);
- Cálculo e atualização do progresso das metas de doação, quando definido.

Dessa forma, os testes asseguram que as campanhas sejam representadas de forma consistente e que as informações exibidas aos usuários reflitam o estado real de cada iniciativa.

4.3. Modelo Delegação

Os testes do modelo Delegação são fundamentais para garantir o correto funcionamento do processo de distribuição de solicitações e campanhas para órgãos parceiros.

Entre os pontos verificados, destacam-se:

- Criação de delegações vinculando moderadores, campanhas e órgãos parceiros;
- Alteração de status da delegação (enviada, aceita, recusada, concluída);
- Registro de decisões e justificativas adotadas pelos órgãos;
- Manutenção do vínculo entre delegações e as respectivas solicitações/campanhas;
- Funcionamento adequado de métodos auxiliares, como o método *to_dict*, utilizado para serialização dos dados e integração com outras camadas do sistema.

Esses testes contribuem para que o fluxo de delegação ocorra de forma previsível e rastreável, evitando inconsistências na gestão das campanhas e no acompanhamento das ações dos órgãos parceiros.

5. Testes de Rotas e Endpoints

Os testes de rotas e endpoints têm como finalidade verificar se a interface exposta pelo Sistema SOS Comida está se comportando conforme o esperado diante de diferentes tipos de requisições. Nessa etapa, são avaliados tanto o retorno das respostas (códigos de status HTTP e conteúdo) quanto o cumprimento das regras de negócio associadas a cada funcionalidade.

Os cenários contemplam operações de autenticação, registro e gerenciamento de campanhas, análise de denúncias e fluxo de delegação para órgãos parceiros, entre outros. Dessa forma, assegura-se que a aplicação web responda de forma

consistente e segura aos usuários, moderadores e instituições cadastradas na plataforma.

5.1. Autenticação e Autorização

Os testes relacionados à autenticação e autorização têm como objetivo garantir que apenas usuários devidamente cadastrados e com permissões válidas consigam acessar as funcionalidades do Sistema SOS Comida. Nessa categoria, foram definidos e executados cenários de teste que contemplam tanto o processo de login, quanto o registro de novos usuários e a manutenção de sessões ativas.

A seguir, são descritos os principais cenários de teste considerados nessa área.

5.1.1. Login com Credenciais Válidas

Este cenário verifica o comportamento do sistema quando um usuário informa credenciais corretas.

- Dado que o usuário possui cadastro ativo na plataforma,
- Quando ele informa e-mail e senha válidos na tela de login e confirma a operação,
- Então o sistema deve autenticar o usuário, criar uma sessão válida e redirecioná-lo para a área correspondente ao seu perfil (por exemplo, usuário comum, instituição, moderador ou órgão parceiro).

O teste valida o retorno de código HTTP adequado (por exemplo, 200 ou 302) e a presença de indicadores de sessão/autenticação bem-sucedida.

5.1.2. Login com Conta Revogada

Este cenário avalia a situação em que o usuário tenta acessar o sistema, mas sua conta foi revogada ou desativada.

- Dado que a conta do usuário encontra-se revogada no banco de dados,
- Quando ele tenta realizar login, ainda que com credenciais corretas,
- Então o sistema deve negar o acesso, não criar sessão válida e exibir uma mensagem informando que a conta está indisponível ou foi desativada.

O teste verifica se o sistema diferencia corretamente erros de credenciais inválidas de contas revogadas, retornando respostas apropriadas.

5.1.3. Login com Autenticação de Dois Fatores (2FA)

Quando a aplicação estiver configurada para utilizar autenticação em dois fatores, este cenário verifica o fluxo completo do processo.

- Dado que o usuário possui 2FA habilitado em sua conta,
- Quando ele informa e-mail e senha corretos,
- Então o sistema deve solicitar o código adicional de autenticação (por exemplo, código enviado por e-mail ou gerado em aplicativo).
- Dado que o usuário informa o código 2FA válido dentro do prazo,
- Quando confirma a autenticação,
- Então o sistema deve concluir o login, criar a sessão e liberar o acesso às funcionalidades.

São testados também comportamentos com código inválido ou expirado, verificando a recusa da autenticação.

5.1.4. Registro de Novo Usuário

Este cenário trata da criação de contas de usuários comuns na plataforma.

- Dado que um visitante acessa a tela de cadastro,
- Quando preenche todos os campos obrigatórios com dados válidos (como nome, e-mail, CPF, telefone e senha) e envia o formulário,
- Então o sistema deve criar um registro de usuário, armazenar os dados no banco em conformidade com as validações definidas e, opcionalmente, solicitar confirmação de e-mail, dependendo da regra de negócio.

O teste verifica se cadastros com dados incompletos, formatos inválidos ou campos duplicados (por exemplo, e-mail já utilizado) são rejeitados com mensagens de erro adequadas.

5.1.5. Registro de Nova Instituição

Este cenário verifica o fluxo de cadastro de instituições ou órgãos parceiros que desejam atuar na plataforma.

- Dado que uma instituição acessa o formulário específico de cadastro institucional,
- Quando informa os dados obrigatórios (razão social, CNPJ, endereço, contato responsável, e-mail, entre outros) e envia o pedido de cadastro,
- Então o sistema deve registrar a solicitação e criar o respectivo perfil de instituição, podendo exigir aprovação posterior por parte de um moderador antes da ativação definitiva da conta.

Os testes para esse cenário também contemplam validações de CNPJ, verificação de dados duplicados e rejeição de cadastros inconsistentes.

5.1.6. Validação de Sessão e Logout

Este cenário abrange o comportamento do sistema em relação à manutenção e encerramento de sessões de usuário.

- Validação de sessão ativa:
 - Dado que o usuário está autenticado,
 - Quando acessa rotas protegidas,
 - Então o sistema deve reconhecer a sessão ativa e permitir o acesso, respeitando o perfil de permissões do usuário.
- Logout (encerramento de sessão):
 - Dado que o usuário deseja encerrar o uso da plataforma,
 - Quando aciona a funcionalidade de logout,
 - Então o sistema deve invalidar a sessão atual, remover ou expirar os dados de autenticação e redirecionar o usuário para uma área pública (por exemplo, página inicial ou tela de login).

Os testes verificam se, após o logout, o usuário não consegue mais acessar rotas protegidas sem realizar nova autenticação.

5.2. Sistema de Campanhas

Os testes relacionados ao sistema de campanhas têm como objetivo verificar se todas as operações que envolvem a criação, publicação, consulta e acompanhamento de campanhas funcionam conforme especificado. Esses testes garantem que campanhas sejam cadastradas com dados válidos, exibidas

corretamente para os usuários e atualizadas de forma consistente ao longo do tempo, incluindo o acompanhamento do progresso das metas de doação.

As rotas testadas abrangem: criação de campanhas, listagem geral, visualização detalhada, aplicação de filtros e ordenações, bem como o registro de participação e de doações vinculadas a cada campanha.

5.2.1. Criação de Campanhas

Este conjunto de testes valida o fluxo de criação de novas campanhas pela aplicação.

- Dado que um usuário ou instituição deseja registrar uma campanha,
- Quando envia uma requisição contendo todos os campos obrigatórios (por exemplo, título, descrição, tipo de ajuda, localização, informações de contato),
- Então o sistema deve validar os dados recebidos, criar o registro correspondente no banco de dados e retornar uma resposta indicando sucesso na operação.

Também são contemplados cenários em que campos obrigatórios estão ausentes, possuem formato inválido ou valores inconsistentes. Nesses casos, o sistema deve rejeitar a requisição, retornando mensagens de erro claras e códigos de status HTTP adequados.

5.2.2. Listagem e Visualização de Campanhas

Os testes de listagem e visualização verificam se as campanhas são exibidas corretamente para os usuários da plataforma.

- Listagem de campanhas ativas:
 - Avalia se a rota de listagem retorna apenas campanhas em estados apropriados (por exemplo, aprovadas e ativas), evitando a exibição de registros pendentes, removidos ou concluídos quando isso não é desejado.
- Visualização detalhada:
 - Verifica se, ao acessar uma campanha específica, o sistema apresenta de forma consistente as informações cadastradas, incluindo descrição, localização, tipo de ajuda, status atual,

responsáveis e, quando aplicável, dados de progresso das doações.

Esses testes asseguram que o usuário consiga compreender o contexto de cada campanha e tomar decisões informadas sobre como contribuir.

5.2.3. Filtros, Busca e Ordenação

Nesta categoria, são avaliadas as funcionalidades de filtro, busca e ordenação disponíveis nas rotas de campanhas.

Entre os cenários cobertos, incluem-se:

- aplicação de filtros por localização (por exemplo, cidade, região ou raio de distância);
- filtros por tipo de campanha ou tipo de ajuda (alimentos, itens de higiene, apoio financeiro, entre outros);
- filtros por status (pendente, ativa, concluída);
- busca por palavras-chave no título ou na descrição;
- ordenação por critérios como data de criação, prioridade ou progresso da meta.

Os testes verificam se a combinação de filtros e ordenações retorna apenas as campanhas que satisfazem os critérios informados, garantindo uma navegação eficiente e orientada às necessidades do usuário.

5.2.4. Participação e Registro de Doações

Os testes deste subgrupo analisam o relacionamento entre as campanhas e as ações dos usuários no sistema, especialmente no que diz respeito à participação e ao registro de doações.

Os principais pontos avaliados são:

- possibilidade de um usuário se engajar em uma campanha (por exemplo, manifestando interesse em ajudar ou oferecer apoio);
- registro de doações associadas à campanha, seja em forma de itens, cestas básicas ou valores monetários, garantindo que a origem e o destino das doações sejam corretamente vinculados;

- atualização do progresso da campanha com base nas doações registradas, de forma que o percentual ou a quantidade atingida refletem fielmente o estado atual da iniciativa.

Com esses testes, assegura-se que o sistema apresente uma visão transparente e confiável do impacto das campanhas, fortalecendo a confiança dos usuários na plataforma SOS Comida.

5.3. Sistema de Delegação

Os testes do sistema de delegação têm como objetivo verificar o correto funcionamento do fluxo em que campanhas ou solicitações são encaminhadas a órgãos parceiros (instituições) responsáveis por atender às demandas registradas na plataforma SOS Comida.

Esse conjunto de testes garante que a criação, o acompanhamento e a conclusão das delegações ocorram de forma consistente, rastreável e em conformidade com as regras de negócio, preservando a transparência do processo entre moderadores, instituições e beneficiários.

5.3.1. Criação de Delegações

Nesta categoria, são avaliados os cenários em que um moderador delega uma campanha ou solicitação a um órgão parceiro.

- Dado que existe uma campanha ou solicitação aprovada e elegível para delegação,
- Quando o moderador seleciona um órgão parceiro e envia a requisição de delegação,
- Então o sistema deve criar um registro de delegação, associando corretamente campanha, moderador e instituição, e definir um status inicial (por exemplo, “Pendente” ou “Aguardando resposta”).

Os testes também verificam se:

- não é possível criar delegações para campanhas inexistentes, removidas ou inválidas;
- não é permitido delegar para instituições inexistentes ou inativas;
- são retornados mensagens e códigos HTTP adequados em casos de erro.

5.3.2. Aceitação de Delegações

Os testes de aceitação asseguram que o órgão parceiro consiga assumir formalmente a responsabilidade por uma campanha ou solicitação delegada.

- Dado que uma delegação foi criada e está com status “Pendente”,
- Quando a instituição acessa a lista de delegações e opta por aceitar uma delas,
- Então o sistema deve atualizar o status da delegação para “Aceita” (ou equivalente) e registrar a decisão, vinculando a campanha à instituição responsável pela gestão.

Os testes verificam se:

- apenas a instituição destinatária consegue aceitar a delegação;
- delegações já aceitas, recusadas ou concluídas não podem ser aceitas novamente;
- a atualização de status reflete corretamente nas consultas posteriores (por exemplo, ao listar campanhas sob gestão de um órgão).

5.3.3. Recusa ou Cancelamento de Delegações

Este grupo de testes cobre os cenários em que a instituição recusa a delegação ou em que a delegação é cancelada por um moderador, quando aplicável.

- Dado que existe uma delegação pendente,
- Quando a instituição decide não assumir a responsabilidade e recusa a delegação,
- Então o sistema deve registrar o status como “Recusada” e, preferencialmente, armazenar o motivo informado.
- Dado que um moderador precisa cancelar uma delegação ainda não aceita,
- Quando ele executa a ação de cancelamento,
- Então o sistema deve atualizar o status para “Cancelada” (ou equivalente) e impedir novas interações naquela delegação.

Os testes verificam se recusas e cancelamentos:

- não alteram campanhas já concluídas;

- não permitem ações adicionais indevidas sobre delegações encerradas;
- mantêm o histórico das decisões para fins de auditoria.

5.3.4. Acompanhamento e Conclusão das Delegações

Por fim, os testes de acompanhamento e conclusão avaliam se o sistema reflete corretamente o andamento da delegação durante e após o atendimento da campanha.

- Dado que uma delegação foi aceita por um órgão parceiro,
- Quando a instituição registra o andamento das ações (como recebimento de doações, distribuição de itens ou encerramento da campanha),
- Então o sistema deve permitir a atualização do status (por exemplo, “Em andamento”, “Concluída”) e manter coerência entre o estado da delegação e o estado da campanha associada.

São verificados, entre outros pontos:

- a consistência entre o status da delegação e o status da campanha;
- a correta exibição de delegações nos relatórios ou telas de acompanhamento;
- a impossibilidade de registrar novas ações em delegações já concluídas ou canceladas.

Com esses testes, o fluxo de delegação torna-se previsível e auditável, reforçando a confiabilidade do Sistema SOS Comida na gestão de campanhas em parceria com instituições externas.

6. Testes de Segurança

Os testes de segurança do Sistema SOS Comida têm como finalidade identificar e mitigar vulnerabilidades que possam comprometer a confidencialidade, a integridade ou a disponibilidade dos dados e serviços oferecidos pela plataforma.

Esses testes são especialmente importantes por se tratar de um sistema que lida com informações pessoais de usuários, instituições e beneficiários, bem como com registros de doações e campanhas, exigindo cuidados adicionais quanto à proteção de dados e ao controle de acesso.

De forma geral, os cenários avaliados contemplam:

- validação de entradas de usuário para reduzir o risco de injeção de código;
- restrição de acesso a recursos sensíveis;
- tratamento adequado de credenciais e senhas;
- exposição controlada de informações em respostas da aplicação.

6.1. Proteção Contra-ataques de Injeção e XSS

Nesta categoria, são realizados testes voltados à prevenção de ataques como SQL Injection e Cross-Site Scripting (XSS), que exploram entradas de usuário maliciosas.

Entre os principais cenários avaliados, destacam-se:

- Injeção de SQL em campos de formulário ou parâmetros de URL:
 - Verifica se entradas contendo comandos SQL ou trechos suspeitos (por exemplo, '; DROP TABLE, OR 1=1) são corretamente tratadas pela aplicação, sem alterar o comportamento esperado das consultas ao banco de dados.
- Inserção de scripts em campos de texto (XSS):
 - Testa o envio de conteúdos que simulam scripts, tags HTML ou código JavaScript em campos como descrição de campanhas, mensagens ou nomes de usuário, avaliando se a aplicação faz a sanitização adequada antes de exibir essas informações em páginas ou retornos JSON.
- Mensagens de erro controladas:
 - Verifica se, em situações de falha, o sistema não expõe detalhes internos de consultas, stack traces ou estruturas do banco de dados, retornando mensagens genéricas o suficiente para não auxiliar um potencial atacante.

Esses testes contribuem para reduzir a superfície de ataque da aplicação e aumentar a resiliência contra entradas maliciosas.

6.2. Controle de Acesso e Isolamento de Perfis

Os testes de controle de acesso avaliam se cada tipo de usuário (doador/receptor, instituição, moderador, órgão parceiro) possui acesso apenas às funcionalidades e informações compatíveis com seu perfil.

Entre os cenários previstos, incluem-se:

- Acesso não autenticado a rotas protegidas:
 - Garante que usuários não autenticados não consigam acessar endpoints que exigem login, obtendo como resposta códigos de status apropriados (por exemplo, 401 ou 403) e mensagens de acesso negado.
- Restrição de ações por perfil:
 - Verifica se operações como aprovação de campanhas, análise de denúncias, delegação para órgãos parceiros e remoção de usuários só podem ser realizadas por perfis autorizados (principalmente moderadores e instituições), impedindo que usuários comuns executem essas ações.
- Isolamento de dados entre usuários:
 - Testa se um usuário não consegue visualizar, editar ou remover recursos que não lhe pertencem (por exemplo, alterar campanhas de outro solicitante ou acessar dados sensíveis de terceiros) apenas alterando identificadores em URLs ou requisições.

Esses testes garantem que a lógica de autorização esteja alinhada às regras de negócio, prevenindo o acesso indevido a dados e funcionalidades sensíveis.

6.3. Tratamento de Credenciais e Dados Sensíveis

Esta categoria concentra-se no tratamento adequado de credenciais e de informações pessoais, considerando tanto boas práticas de segurança quanto diretrizes de proteção de dados.

Os principais pontos verificados são:

- Armazenamento de senhas:
 - Confirma se as senhas de usuários não são armazenadas em texto puro, mas sim utilizando funções de hash adequadas (com

sal e algoritmos modernos), inviabilizando a recuperação direta da senha a partir do banco de dados.

- Transmissão de credenciais:
 - Avalia se as credenciais são enviadas apenas pelos canais previstos (por exemplo, formulários de login) e se a aplicação desestimula o envio de senha em rotas não específicas para autenticação.
- Exposição de dados sensíveis em respostas:
 - Testa se respostas da API ou páginas não retornam informações desnecessárias, como senhas, hashes, tokens internos ou dados pessoais além do estritamente necessário para o contexto de uso.
- Logs e auditoria:
 - Verifica se eventuais registros de log não armazenam credenciais em texto puro e se ações críticas (como revogação de conta, remoção de campanhas ou análise de denúncias) podem ser rastreadas por meio de registros de auditoria.

Com esses testes, busca-se reforçar a conformidade do sistema com boas práticas de segurança e com princípios de proteção de dados, reduzindo riscos para os usuários e para a própria plataforma.

7. Testes de Validação

Os testes de validação têm como objetivo garantir que apenas dados consistentes, completos e em formato adequado sejam aceitos e armazenados pelo Sistema SOS Comida.

Esse tipo de teste é fundamental para manter a integridade e a confiabilidade das informações, reduzindo a ocorrência de erros lógicos, inconsistências cadastrais e problemas de segurança decorrentes de entradas malformadas ou maliciosas.

As validações contemplam especialmente campos considerados críticos, como CPF, telefone, e-mail, senhas e dados de identificação de usuários e instituições.

7.1. Validação de Campos

Nesta categoria, os testes verificam se as regras de validação definidas para cada campo de entrada estão sendo corretamente aplicadas pela aplicação. Entre os principais pontos avaliados, destacam-se:

- Campos obrigatórios
 - Verifica se campos marcados como obrigatórios (por exemplo, nome, e-mail, senha, tipo de usuário, dados básicos da campanha) não podem ser deixados em branco.
 - Resultado esperado: requisições com campos obrigatórios ausentes devem ser rejeitadas, com mensagens de erro claras indicando quais informações estão faltando.
- Validação de CPF
 - Testa o algoritmo de validação de CPF, cobrindo casos com:
 - CPF válido;
 - CPF com formato incorreto (quantidade de dígitos diferente de 11);
 - CPF com dígitos inválidos segundo o cálculo dos verificadores;
 - CPF duplicado já cadastrado no sistema.
 - Resultado esperado: apenas CPFs válidos e ainda não cadastrados são aceitos.
- Validação de telefone
 - Verifica se o telefone informado segue o padrão adotado pelo sistema (por exemplo, DDD + número, com quantidade adequada de dígitos).
 - Resultado esperado: telefones em formato inválido ou com quantidade incorreta de dígitos devem ser rejeitados, evitando cadastros inconsistentes.
- Validação de e-mail
 - Avalia se o e-mail informado atende aos requisitos básicos de formato (presença de “@”, domínio, ausência de caracteres proibidos) e se não está sendo utilizado por outro usuário já cadastrado.

- Resultado esperado: e-mails com formato inválido ou duplicados devem gerar erro de validação.
- Validação de senha
 - Testa regras mínimas para criação de senha, tais como:
 - comprimento mínimo (por exemplo, 6 caracteres ou mais);
 - correspondência entre senha e confirmação de senha;
 - Resultado esperado: senhas muito curtas, vazias ou que não coincidam com o campo de confirmação são rejeitadas, prevenindo cadastros frágeis ou incorretos.
- Validações específicas de cadastro de instituições e campanhas
 - No caso de instituições, podem ser testados campos como CNPJ, razão social e dados de contato, garantindo que estejam em formato adequado e completos.
 - Para campanhas, são validados campos como título, descrição, tipo de ajuda, localização e demais atributos obrigatórios para que a campanha seja comprehensível e possa ser analisada/avaliada pela moderação.

Esses testes de validação, em conjunto, contribuem para que o Sistema SOS Comida trabalhe sempre com dados coerentes e bem estruturados, reduzindo falhas em etapas posteriores (como delegação, denúncias, doações e relatórios) e facilitando a manutenção futura da aplicação.

8. Testes de Casos Extremos

Os testes de casos extremos (ou testes de borda) têm como objetivo avaliar o comportamento do Sistema SOS Comida em situações-limite, pouco prováveis no uso cotidiano, mas que podem revelar falhas importantes de validação, tratamento de erros ou robustez da aplicação.

Esses testes complementam os testes de validação e de regras de negócio, verificando como o sistema reage diante de entradas muito grandes, muito pequenas, ausentes ou fora do intervalo esperado, além de operações repetidas ou inconsistentes.

8.1. Cenários de Casos Extremos Avaliados

Entre os principais cenários de casos extremos considerados na suíte de testes, destacam-se:

- Campos obrigatórios ausentes em combinações não triviais
 - Verifica situações em que múltiplos campos obrigatórios são omitidos ao mesmo tempo (por exemplo, título e descrição da campanha, ou e-mail e senha em um cadastro).
 - Resultado esperado: o sistema deve identificar todos os campos ausentes relevantes, rejeitar a operação e retornar mensagens de erro claras, sem falhas inesperadas ou comportamentos inconsistentes.
- Valores negativos ou fora de faixa em campos numéricos
 - Testa o envio de valores negativos, zero ou muito acima do esperado em campos como quantidade de itens, metas de doação ou outros atributos numéricos.
 - Resultado esperado: o sistema deve rejeitar valores que não façam sentido para o contexto (por exemplo, quantidade negativa de cestas básicas) e, quando houver faixas definidas, impedir a gravação de valores fora dos limites estabelecidos.
- Uso de caracteres especiais, Unicode e textos muito longos
 - Avalia o comportamento da aplicação quando campos de texto (como nome, título, descrição ou mensagem) recebem caracteres especiais, emojis, acentos, símbolos ou textos com tamanho muito superior ao usual.
 - Resultado esperado: a aplicação deve suportar corretamente caracteres Unicode válidos (como acentos e emojis, quando desejado) e, ao mesmo tempo, limitar ou truncar textos excessivamente longos, prevenindo erros de banco de dados ou falhas na renderização das páginas.
- Tentativas de overflow ou tamanhos máximos
 - Testa o envio de valores e textos próximos ou além dos limites definidos em banco ou na aplicação (por exemplo, campos VARCHAR ou inteiros com valores muito grandes).

- Resultado esperado: o sistema deve tratar essas situações de forma controlada, evitando estouros de limite (overflow), erros de conversão ou interrupções inesperadas no processamento da requisição.
- Operações duplicadas ou repetidas rapidamente
 - Analisa o comportamento quando um mesmo usuário tenta, por exemplo, registrar a mesma doação diversas vezes em sequência, criar campanhas idênticas ou reenviar formulários de forma repetida.
 - Resultado esperado: a aplicação deve lidar com essas situações sem gerar inconsistências (como registros duplicados indevidos) e, quando possível, sinalizar ao usuário que a ação já foi processada.
- Operações com divisão por zero ou cálculos inválidos
 - Em cenários que envolvem cálculos de progresso, percentuais ou métricas derivadas, são testadas situações em que o denominador é zero ou inexistente (por exemplo, meta igual a zero em uma campanha).
 - Resultado esperado: o sistema deve evitar explicitamente divisões por zero, tratando esses casos com lógica alternativa (por exemplo, exibindo progresso “não definido” ou “0%”) sem lançar exceções em tempo de execução.

A execução desses testes de casos extremos contribui para tornar o Sistema SOS Comida mais robusto e resiliente, reduzindo a probabilidade de falhas em situações atípicas e melhorando a experiência do usuário mesmo diante de entradas inesperadas ou incorretas.

9. Testes de Performance

Os testes de performance têm como finalidade avaliar se o Sistema SOS Comida responde de maneira eficiente às requisições dos usuários, mesmo em cenários com maior volume de dados e acessos.

Esse tipo de teste é fundamental para garantir uma boa experiência de uso, evitando tempos de resposta excessivos em operações críticas, como listagem de

campanhas, buscas e registro de doações, que são ações frequentes dentro da plataforma.

9.1. Objetivos dos Testes de Performance

Os principais objetivos definidos para os testes de performance são:

- verificar se o sistema mantém tempos de resposta aceitáveis em operações essenciais;
- avaliar o comportamento da aplicação em cenários com carga de dados elevada;
- identificar possíveis gargalos em consultas ao banco de dados, processamento de regras de negócio ou renderização de páginas;
- subsidiar melhorias futuras na arquitetura, nas consultas e na implementação das funcionalidades mais acessadas.

9.2. Métricas e Cenários Avaliados

Para a avaliação da performance, foram estabelecidas algumas métricas e cenários de referência, entre os quais se destacam:

- Listagem de campanhas com grande volume de dados
 - Cenário: consulta à lista de campanhas ativas em um contexto com mais de 100 registros cadastrados.
 - Métrica observada: tempo total de resposta da requisição.
 - Referência adotada: o tempo de resposta deve ser de, no máximo, 2 segundos para a maior parte das requisições (por exemplo, percentil 90 – P90).
- Busca de campanhas com filtros e palavras-chave
 - Cenário: uso simultâneo de filtros (por localização, tipo de ajuda, status) e, quando disponível, busca por palavras-chave em título ou descrição.
 - Métrica observada: tempo de resposta para retorno dos resultados filtrados.
 - Referência adotada: o tempo de resposta deve se manter em até 1 segundo para consultas típicas em ambiente de teste, considerando volume de dados previamente definido.

- Operações de banco de dados em rotas críticas
 - Cenário: análise de operações de criação, atualização e leitura de dados em rotas mais acessadas (como registro de doações, criação de campanhas e autenticação).
 - Métrica observada: tempo médio de execução das principais consultas e transações.
 - Referência adotada: idealmente, cada operação de banco de dados deve ser concluída em até 500 ms, em ambiente de teste controlado.
- Renderização de páginas complexas
 - Cenário: visualização de telas que agregam múltiplas informações, como detalhamento de campanhas com histórico de doações e delegações associadas.
 - Métrica observada: tempo percebido para carregamento completo da página no cliente (quando avaliado em interface web).
 - Referência adotada: o tempo total de carregamento deve ficar em torno de 3 segundos ou menos, de forma a manter uma navegação fluida.

9.3. Resultados Esperados e Limites de Aceitação

Com base nas métricas definidas, são considerados aceitáveis os seguintes limites de desempenho em ambiente de testes:

- listagem de campanhas: resposta em até 2 segundos na maioria das requisições;
- buscas com filtros: resposta em até 1 segundo em cenários típicos;
- operações de banco de dados em rotas críticas: conclusão em até 500 ms por operação, em média;
- renderização de páginas mais complexas: carregamento em até 3 segundos.

Caso os testes indiquem tempos significativamente superior a esses valores, recomenda-se:

- revisar consultas e índices de banco de dados;

- otimizar trechos de código responsáveis por loops ou processamento intensivo;
- reduzir a quantidade de dados retornados em uma única requisição, utilizando paginação ou filtros mais específicos;
- reavaliar o formato das respostas (por exemplo, tamanho de payloads JSON).

9.4. Considerações Sobre Evolução da Performance

Os valores de referência adotados neste relatório devem ser encarados como parâmetros iniciais, passíveis de revisão conforme o sistema evolua ou passe a operar com volumes maiores de dados e usuários.

A manutenção de uma boa performance depende da execução periódica de novos testes, especialmente após alterações significativas na arquitetura, na modelagem do banco de dados ou em funcionalidades críticas, garantindo que o Sistema SOS Comida continue oferecendo respostas rápidas e estáveis ao longo de seu ciclo de vida.

10. Testes de Integração

Os testes de integração têm como objetivo verificar o funcionamento conjunto dos diferentes módulos do Sistema SOS Comida, avaliando se a interação entre camadas (modelos, rotas, serviços, regras de negócio e banco de dados) ocorre de forma correta ao longo dos principais fluxos de uso da aplicação.

Enquanto os testes unitários se concentram em componentes isolados, os testes de integração validam fluxos completos, desde a entrada de dados pelo usuário até a persistência no banco e o retorno das informações, simulando situações mais próximas do uso real do sistema.

10.1. Objetivos dos Testes de Integração

Os principais objetivos definidos para os testes de integração são:

- garantir que módulos que funcionam corretamente de forma isolada continuem se comportando conforme o esperado quando combinados;
- validar fluxos de negócio completos, envolvendo autenticação, cadastro, campanhas, denúncias, delegações e doações;

- identificar inconsistências em regras de negócio distribuídas entre camadas diferentes;
- verificar se o sistema mantém a integridade e coerência dos dados ao longo de operações encadeadas.

10.2. Fluxo Integrado de Campanha

Um dos fluxos centrais avaliados nos testes de integração é o ciclo de vida de uma campanha, envolvendo múltiplos perfis de usuário.

Entre os cenários contemplados, destacam-se:

- Criação de campanha por usuário ou instituição
 - Autenticação do solicitante;
 - Envio dos dados da campanha por rota apropriada;
 - Validação de campos obrigatórios e regras de negócio;
 - Registro da campanha no banco de dados com status inicial adequado (por exemplo, “Pendente de aprovação”).
- Revisão e aprovação por moderador
 - Autenticação como moderador;
 - Listagem de campanhas pendentes;
 - Aprovação ou rejeição da campanha segundo os critérios da plataforma;
 - Atualização do status para “Ativa” em caso de aprovação.
- Exibição para usuários e doadores
 - Listagem de campanhas ativas para usuários comuns;
 - Visualização detalhada, incluindo informações de contato, localização e tipo de ajuda;
 - Verificação de filtros e busca integrados ao fluxo.

Esses cenários garantem que o fluxo desde a criação até a publicação das campanhas ocorra de forma encadeada e sem inconsistências entre módulos.

10.3. Fluxo Integrado de Delegação para Órgãos Parceiros

Outro fluxo relevante testado é o processo de delegação de campanhas ou solicitações para órgãos parceiros.

Os principais passos integrados avaliados são:

- aprovação prévia da campanha ou solicitação por um moderador;
- seleção de um órgão parceiro apto a receber a delegação;
- criação do registro de delegação, vinculando campanha, moderador e instituição;
- autenticação da instituição e acesso à lista de delegações pendentes;
- aceitação ou recusa da delegação, com atualização do status e registro da decisão;
- acompanhamento do atendimento pela instituição e, quando apropriado, conclusão da delegação e da campanha associada.

Os testes verificam se todas essas etapas funcionam de maneira contínua, com dados coerentes em cada ponto do fluxo e sem perda de informações entre as transições de estado.

10.4. Fluxo Integrado de Denúncias

Os testes de integração de denúncias avaliam o caminho completo desde o registro da denúncia por parte de um usuário até a análise e decisão de um moderador.

Entre os cenários avaliados, incluem-se:

- envio de denúncia por usuário autenticado, com registro de dados mínimos necessários (motivo, descrição, campanha ou usuário relacionado);
- persistência da denúncia com status inicial (por exemplo, “Pendente”);
- listagem das denúncias para o perfil de moderador;
- análise da denúncia, com possibilidade de classificação, arquivamento ou aplicação de medidas (como remoção de campanha ou revogação de conta, conforme regras de negócio);
- atualização do status da denúncia e registro de decisões para fins de auditoria.

Esses testes asseguram que o fluxo de denúncias integra corretamente o cadastro, a análise e as ações aplicadas no sistema.

10.5. Fluxo Integrado de Doações e Acompanhamento

Os testes de integração também contemplam o fluxo de registro de doações e atualização do progresso das campanhas, uma parte crítica para a transparência da plataforma.

Os passos principais avaliados são:

- autenticação do doador na plataforma;
- seleção de uma campanha ativa para apoio;
- envio dos dados da doação (itens, quantidade, valor ou tipo de contribuição);
- registro da doação no banco de dados, vinculando doador, campanha e, quando pertinente, instituição responsável;
- atualização dos indicadores de progresso da campanha, considerando as doações registradas;
- visualização, pelos usuários, da campanha com o progresso atualizado.

Os testes verificam se os dados de doação fluem corretamente entre rotas, modelos e camadas de negócio, e se os valores apresentados para os usuários refletem com precisão o estado atual da campanha.

10.6. Resultados e Importância dos Testes de Integração

A execução dos testes de integração permite identificar problemas que não aparecem em testes unitários isolados, como:

- falhas de comunicação entre camadas;
- inconsistências nos dados trocados entre módulos;
- regras de negócio parcialmente aplicadas em apenas uma parte do fluxo;
- estados incorretos de campanhas, delegações ou denúncias após operações encadeadas.

Dessa forma, os testes de integração contribuem para assegurar que o Sistema SOS Comida funcione como um todo coeso, oferecendo uma experiência estável e confiável para todos os perfis de usuário da plataforma.

11. Cobertura de Código

A cobertura de código é uma métrica utilizada para avaliar em que medida o código-fonte da aplicação é exercitado pelos testes automatizados. No contexto do Sistema SOS Comida, essa métrica é empregada como um indicador da abrangência da suíte de testes, auxiliando na identificação de trechos de código que ainda não foram validados e que podem representar potenciais pontos de falha.

A análise de cobertura foi realizada utilizando ferramentas integradas ao framework de testes (como o `pytest-cov`), permitindo a geração de relatórios detalhados por módulo, arquivo e função.

11.1. Metas de Cobertura Estabelecidas

Com o objetivo de garantir um nível mínimo de qualidade e confiança no sistema, foram definidas metas de cobertura de código para os principais componentes da aplicação:

- Cobertura global da aplicação:
 - Meta mínima: 80% do código-fonte exercitado por testes automatizados.
- Modelos de dados (models):
 - Meta mínima: 90%, considerando a importância desses componentes na integridade das informações e na aplicação das regras de negócio.
- Rotas e endpoints (controllers):
 - Meta mínima: 85%, devido ao seu papel na exposição de funcionalidades ao usuário e na coordenação de fluxos entre camadas.
- Módulos de validação e regras críticas:
 - Meta esperada: próxima de 100%, dado que esses trechos concentram validações de entrada, cálculos e decisões lógicas que impactam diretamente a confiabilidade do sistema.

Esses valores servem como referência para a equipe de desenvolvimento, orientando esforços de escrita e manutenção de testes.

11.2. Ferramentas e Forma de Medição

A cobertura foi medida por meio da execução dos testes com o uso do plugin `pytest-cov`, utilizando comandos semelhantes a:

```
pytest --cov=. --cov-report=html
```

Esse comando gera tanto um índice numérico de cobertura quanto um relatório em formato HTML, permitindo:

- visualizar a cobertura por arquivo e por linha de código;
- identificar trechos não cobertos (linhas nunca executadas durante os testes);
- apoiar decisões sobre onde concentrar esforços na criação de novos casos de teste.

A inspeção desses relatórios possibilita um acompanhamento contínuo da evolução da cobertura, especialmente à medida que novas funcionalidades são adicionadas ao sistema.

11.3. Interpretação dos Resultados

Embora um alto percentual de cobertura não garanta, por si só, a ausência de defeitos, valores baixos de cobertura indicam claramente a existência de funcionalidades não testadas. Assim:

- trechos com cobertura elevada tendem a ter seu comportamento mais bem conhecido e validado;
- trechos com cobertura baixa ou nula representam risco maior, pois podem conter erros que não são evidenciados pela suíte de testes automatizados.

No caso do Sistema SOS Comida, a meta é manter a cobertura global acima de 80%, buscando, sempre que possível, aproximar-se ou superar a marca de 90% em módulos centrais, como modelos, regras de validação e rotas mais utilizadas. A cada incremento ou refatoração do código, recomenda-se reexecutar os testes com relatório de cobertura para garantir que a evolução da aplicação não reduza o nível de confiabilidade obtido.

11.4. Uso da Cobertura Como Ferramenta de Melhoria Contínua

Mais do que um número estático, a cobertura de código é tratada como uma ferramenta de melhoria contínua. A partir dos relatórios gerados, a equipe pode:

- priorizar a criação de novos testes para módulos com baixa cobertura;
- revisar trechos de código muito extensos ou complexos que estejam pouco testados;
- identificar códigos mortos ou obsoletos (nunca executados), que podem ser simplificados ou removidos;
- acompanhar o impacto de novas funcionalidades na qualidade geral dos testes.

Dessa forma, o monitoramento sistemático da cobertura de código contribui para um processo de desenvolvimento mais seguro, estruturado e alinhado às boas práticas de engenharia de software.

12. Comandos de Execução

Esta seção apresenta os principais comandos utilizados para execução da suíte de testes do Sistema SOS Comida, bem como orientações gerais sobre quando utilizar cada um deles. Todos os exemplos consideram o uso do framework *pytest* como ferramenta padrão para testes automatizados em Python.

12.1. Instalação de Dependências de Teste

Antes da execução dos testes, é necessário garantir que as bibliotecas de apoio estejam instaladas no ambiente virtual do projeto. Um conjunto típico de dependências pode ser instalado com o seguinte comando:

```
pip install pytest pytest-cov pytest-flask pytest-mock faker
```

Esse comando instala:

- *pytest* – framework principal de testes;
- *pytest-cov* – medição de cobertura de código;
- *pytest-flask* e *pytest-mock* – suporte a aplicações Flask e mocks;
- *faker* – geração de dados fictícios para cenários de teste.

12.2. Execução Geral da Suíte de Testes

Para executar todos os testes automatizados definidos no projeto, utiliza-se o comando:

```
pytest
```

Esse comando procura automaticamente por arquivos do padrão `test_*.py` ou `*_test.py` no diretório configurado e exibe um resumo com a quantidade de testes executados, aprovados e falhos.

12.3. Execução com Relatório de Cobertura de Código

Quando se deseja avaliar a cobertura de código atingida pelos testes, pode-se utilizar:

```
pytest --cov=. --cov-report=html
```

Esse comando:

- executa a suíte de testes;
- calcula a cobertura de código do diretório atual (.);
- gera um relatório em formato HTML (normalmente na pasta `htmlcov/`), permitindo análise detalhada por arquivo e por linha de código.

12.4. Execução de Conjuntos Específicos de Testes

Em alguns momentos, pode ser útil executar apenas um subconjunto de testes, seja para focar em um módulo em particular, seja para acelerar o ciclo de desenvolvimento.

Exemplos:

- Executar apenas os testes de modelos:

```
pytest tests/test_models.py
```

- Executar apenas os testes de rotas/endereços da API:

```
pytest tests/test_routes.py
```

Também é possível restringir a execução a uma função de teste específica dentro de um arquivo:

```
pytest tests/test_models.py::test_criar_usuario_valido
```

12.5. Execução com Maior Nível de Detalhamento

Para obter mais detalhes sobre quais testes estão sendo executados (por exemplo, o nome de cada caso de teste), pode-se utilizar a opção verbose:

```
pytest -v
```

Essa opção pode ser combinada com as demais, por exemplo:

```
pytest -v --cov=. --cov-report=html
```

Dessa forma, além do relatório de cobertura, o desenvolvedor visualiza, de forma mais explícita, os testes executados e eventuais falhas, facilitando o diagnóstico de problemas.

13. Testes Funcionais da Aplicação SOS Comida

Os testes funcionais do Sistema SOS Comida foram estruturados a partir das 19 Histórias de Usuário (HU01–HU19) definidas na etapa de especificação de requisitos.

Cada história de usuário foi decomposta em critérios de aceitação, descritos em formato comportamental (Dado / Quando / Então), e avaliada diretamente na aplicação em funcionamento. Ao final, cada critério foi classificado como ACEITO, NÃO ACEITO ou PARCIALMENTE ACEITO, de acordo com o grau de aderência entre o comportamento esperado e o comportamento observado no sistema.

No total, foram avaliados 75 critérios de aceitação, distribuídos entre as 19 histórias de usuário, com o seguinte resultado geral:

- 57 critérios ACEITO – comportamento da aplicação plenamente aderente ao especificado;
- 16 critérios NÃO ACEITO – funcionalidade não implementada ou implementada de forma divergente;
- 2 critérios PARCIALMENTE ACEITO – comportamento parcialmente atendido, com pequenas diferenças em relação ao especificado.

Essa análise permitiu tanto comprovar a cobertura funcional do sistema quanto identificar pontos de ajuste na documentação e oportunidades de melhoria futura na aplicação.

13.1. Objetivo e Metodologia dos Testes Funcionais

O objetivo dos testes funcionais foi verificar se o Sistema SOS Comida:

- implementa corretamente os requisitos descritos nas histórias de usuário;
- mantém a coerência entre fluxos de uso documentados e funcionalidades disponibilizadas na interface;
- oferece uma experiência alinhada ao propósito de apoiar pessoas em situação de vulnerabilidade por meio de campanhas, denúncias, delegações e doações.

A metodologia adotada foi a seguinte:

- Para cada HU (HU01 a HU19), foram identificados os critérios de aceitação documentados;
- Cada critério foi convertido em um cenário de teste funcional, executado diretamente na aplicação;
- O resultado observado foi comparado com o comportamento esperado;
- Quando houve divergência, foram registrados Relatório de Erros e Correção, indicando se a ação foi:
 - ajuste na documentação (para refletir o comportamento real do sistema); ou
 - sugestão de melhoria futura na aplicação (para atender ao requisito originalmente descrito).

13.2. Cobertura dos Testes Funcionais por Grupo De Histórias

Para facilitar a análise, as histórias de usuário foram organizadas em grupos temáticos.

a) Cadastro e Autenticação de Usuários (HU01-HU03)

- HU01 – Criar cadastro e HU02 – Gerenciar cadastro: todos os critérios de aceitação foram ACEITOS, confirmando que o sistema permite criar cadastros válidos, manter dados atualizados e aplicar validações básicas (campos obrigatórios, formato de e-mail etc.).

- HU03 – Realizar login: os critérios relacionados ao login básico foram aceitos; porém, critérios adicionais como limite de tentativas, tempo de expiração de sessão e recuperação de senha por e-mail foram classificados como NÃO ACEITO, por não estarem implementados na versão atual. Esses itens foram ajustados na documentação ou apontados como melhorias futuras.

b) Registro de Denúncias e Moderação (HU04–HU07)

- HU04 – Registrar denúncia: o fluxo principal de registro de denúncias e definição de status inicial foi ACEITO. Um critério ligado à rejeição automática de conteúdo proibido foi classificado como NÃO ACEITO, dado que a filtragem é feita manualmente pelo moderador.
- HU05 – Solicitar campanha e HU06 – Visualizar campanhas: a criação e visualização de campanhas foram validadas, com critérios essenciais aceitos. Em HU05, um critério sobre limitação rígida de campanhas por usuário foi ajustado (um dos critérios ficou NÃO ACEITO ou PARCIALMENTE ACEITO, conforme o comportamento real da aplicação).
- HU07 – Analisar denúncias: o moderador consegue listar e analisar denúncias, mas a notificação automática ao usuário denunciado não está presente, resultando em um critério NÃO ACEITO.

c) Gestão de Campanhas pelos Moderadores (HU08–HU11)

- HU08 – Aprovar campanhas, HU09 – Editar campanhas e HU10 – Remover campanhas: os fluxos principais (aprovar, editar e remover) foram confirmados como funcionais.
 - Critérios ligados a prazos automáticos de campanha e manutenção de histórico detalhado após remoção foram classificados como NÃO ACEITO, por não corresponderem exatamente ao funcionamento atual do sistema.

- HU11 – Remover usuários: a remoção de usuários está implementada, mas a preservação de histórico em determinados cenários diverge da especificação, resultando em pelo menos um critério NÃO ACEITO e consequente ajuste na documentação.

d) Delegação para Órgãos Parceiros (HU12–HU15)

- HU12 – Delegar campanhas a órgãos e HU13 – Aceitar campanhas: a criação da delegação e a aceitação pela instituição parceira foram confirmadas (critérios principais ACEITOS). Critérios relacionados a prazo máximo para resposta e notificações automáticas foram classificados como NÃO ACEITO.
- HU14 – Recusar campanhas: os critérios de recusa e registro de motivo foram ACEITOS, sem não conformidades relevantes.
- HU15 – Revogar cadastro de usuários (pelos órgãos): alguns critérios foram marcados como NÃO ACEITO, pois a aplicação não permite, na versão atual, que o órgão remova diretamente usuários em todos os cenários imaginados na documentação. Essa diferença motivou ajustes e sugestão de melhoria na aplicação.

e) Visualização, Busca e Doações em Campanhas (HU16–HU19)

- HU16 – Visualizar campanhas ativas e HU17 – Buscar campanhas: os critérios referentes à listagem de campanhas ativas, aplicação de filtros e tempos de resposta foram ACEITOS, comprovando a aderência da interface às necessidades de busca e exploração de campanhas.
- HU18 – Solicitar doações específicas: o cadastro de itens específicos em uma campanha possui critérios aceitos, mas um critério relacionado à aprovação especial de itens sensíveis foi marcado como NÃO ACEITO, por não estar implementado como descrito originalmente.

- HU19 – Registrar doações: o fluxo de registro de doações e vinculação à campanha foi validado. Um critério sobre notificação automática a todos os envolvidos foi classificado como PARCIALMENTE ACEITO, já que as doações ficam visíveis na campanha, mas não há uma notificação ativa específica para cada doação; esse ponto foi ajustado na documentação.

13.3. Principais Não Conformidades e Ajustes Realizados

A análise dos testes funcionais evidenciou um padrão importante:

- Em vários casos, a aplicação funciona corretamente, mas alguns critérios de aceitação estavam mais ambiciosos do que a implementação real, prevendo:
 - limites rígidos de tempo (SLA, expiração de sessão, prazos automáticos de campanha ou delegação);
 - notificações automáticas em situações em que, atualmente, a plataforma utiliza apenas visualização em tela;
 - preservação de histórico detalhado em cenários em que os dados são simplificados ou removidos.

Diante disso, foram tomadas duas ações principais:

- Ajuste da documentação
 - Em critérios marcados como NÃO ACEITO ou PARCIALMENTE ACEITO devido à diferença entre o que estava escrito e o que o sistema realmente faz, optou-se, em vários casos, por atualizar a documentação para refletir com fidelidade o comportamento atual da aplicação.
- Registro de melhorias futuras
 - Em outros casos, especialmente quando o critério representava uma boa prática ou uma funcionalidade desejável (por exemplo, notificações automáticas, históricos mais completos ou controles adicionais de segurança), o item foi mantido como objetivo futuro, servindo de base para um backlog de melhorias do Sistema SOS Comida.

13.4. Conclusão dos Testes Funcionais

Os testes funcionais baseados nas Histórias de Usuário permitem concluir que:

- a maior parte dos fluxos centrais do Sistema SOS Comida (cadastro, login, campanhas, denúncias, delegações e doações) está implementada e aderente ao que foi proposto;
- as divergências identificadas concentram-se, principalmente, em critérios adicionais de prazo, notificação automática e histórico detalhado, que podem ser tratados como melhorias incrementais;
- a revisão conjunta de aplicação em funcionamento + documentação de requisitos resultou em um alinhamento mais realista entre o que o sistema faz hoje e o que está descrito nos artefatos do projeto.

Dessa forma, os testes funcionais contribuem não apenas para validar a solução implementada, mas também para orientar a evolução futura do Sistema SOS Comida, com base em evidências observadas diretamente no uso da aplicação.

14. Conclusão

O presente relatório consolidou a estratégia, a execução e os resultados dos testes realizados no Sistema SOS Comida, contemplando testes unitários, de integração, de segurança, de validação, de casos extremos, de performance e testes funcionais baseados em histórias de usuário.

De forma geral, os resultados indicam que o sistema apresenta um nível consistente de qualidade, especialmente nos fluxos centrais de uso, tais como:

- cadastro e autenticação de usuários;
- criação, aprovação, edição, visualização e remoção de campanhas;
- registro e análise de denúncias;
- delegação de campanhas para órgãos parceiros;
- registro de doações e acompanhamento do progresso das campanhas.

Os testes unitários e de integração demonstraram que os principais módulos da aplicação (modelos, rotas e regras de negócio) funcionam de maneira coesa quando combinados, preservando a integridade dos dados e o encadeamento correto dos fluxos. A análise de cobertura de código reforçou essa percepção, ao indicar níveis

satisfatórios de exercício do código pelos testes automatizados, sobretudo em componentes considerados críticos.

Os testes de segurança, validação e casos extremos contribuíram para evidenciar cuidados com entrada de dados, controle de acesso e robustez do sistema frente a situações atípicas, reduzindo a probabilidade de falhas em cenários reais de uso. Já os testes de performance mostraram que, dentro dos parâmetros definidos em ambiente de teste, o sistema tende a oferecer tempos de resposta adequados nas operações mais frequentes, como listagem e busca de campanhas.

No âmbito dos testes funcionais, a verificação sistemática dos critérios de aceitação das Histórias de Usuário (HU01–HU19) permitiu alinhar a documentação de requisitos ao comportamento efetivamente implementado. Em diversos casos, foram identificadas divergências pontuais — principalmente relacionadas a prazos automáticos, notificações e preservação de histórico detalhado — que foram tratadas por meio de ajustes na documentação ou registradas como oportunidades de melhoria futura para o sistema.

Em síntese, o conjunto de testes descrito neste relatório cumpre o papel de evidenciar a qualidade atual do Sistema SOS Comida e de servir como base para sua manutenção e evolução, contribuindo para que a aplicação continue apoiando, de maneira confiável e transparente, ações de solidariedade voltadas a pessoas em situação de vulnerabilidade.