**ROTEIRO DO PROJETO – SOS COMIDA**

1. **Requisito**
   1. **Visão e escopo**

* **Declaração do problema:** O problema central é a falha na comunicação e logística entre projetos de doação, ONGs e pessoas em situação de vulnerabilidade. Isso afeta a capacidade das organizações de alcançar quem precisa e, consequentemente, o impacto é que muitas pessoas necessitadas não conseguem receber doações de forma eficiente.
* **Visão:** Desenvolver uma plataforma de software inovadora, chamada "SOS Comida", que sirva como um canal de comunicação eficiente, seguro e acessível. O objetivo é criar uma ponte direta entre quem precisa de ajuda (alimentos, roupas etc.) e quem deseja contribuir com recursos, tempo (voluntariado) ou apoio logístico, transformando a solidariedade em ação concreta e organizada.
* **Escopo:** O escopo do projeto está focado em criaras funcionalidades essenciais para conectar os diferentes perfis de usuários. O quadro abaixo detalha o que está incluído no projeto:

|  |  |  |
| --- | --- | --- |
| **Necessidade** | **Funcionalidade** | **Prioridade** |
| Pessoas em vulnerabilidade precisam receber ajuda | Permitir que beneficiários submetam solicitações de ajuda, detalhando suas necessidades. | Alta |
| Doadores precisam encontrar formas confiáveis de ajudar. | Disponibilizar um portal para doadores consultarem e filtrarem as solicitações de ajuda verificadas. | Alta |
| Garantir a transparência sobre o impacto das contribuições. | Gerar e enviar relatórios periódicos aos doadores sobre o status das solicitações que apoiaram. | Média |
| Assegurar a segurança e a integridade da plataforma. | Implementar um painel administrativo para moderadores gerenciarem usuários, solicitações e denúncias. | Alta |
| Usuários precisam acompanhar o andamento de seus pedidos/doações. | Implementar um sistema de notificações para informar os usuários sobre atualizações de status. | Média |
| Incentivar a continuidade das doações e do engajamento. | Desenvolver um sistema de reconhecimento (selos/gamificação) para doadores e voluntários. | Alta |

* 1. **Especificação de requisitos**
* **História de usuário:** 
  + **HU01:** Como um novo usuário, eu quero criar e gerenciar meu cadastro, para eu possa acessar a plataforma de forma segura e personalizada.
  + **HU02:** Como um usuário cadastrado, eu quero fazer login com minhas credenciais, para acessar as funcionalidades correspondentes ao meu perfil.
  + **HU03:** Como um doador, eu quero denunciar irregularidades em uma campanha ou perfil, para ajudar a manter a plataforma segura e confiável.
  + **HU04:** Como um receptor, eu quero solicitar a criação de uma campanha de arrecadação, para divulgar minha necessidade e obter ajuda da comunidade.
  + **HU05:** Como um moderador, eu quero visualizar e analisar as denúncias feitas pelos usuários, para tomar as ações necessárias e garantir a integridade da plataforma.
  + **HU06:** Como um moderador, eu quero revisar e aprovar as campanhas solicitadas, para que elas se tornem públicas e possam receber doações.
  + **HU07:** Como um moderado, eu quero editar as informações de uma campanha ativa, para corrigir dados ou atualizar o status conforme necessário.
  + **HU08:** Como um moderador, eu quero remover campanhas inadequadas ou finalizadas, para manter a lista de campanhas relevante e atualizada.
  + **HU09:** Como um moderador, eu quero remover usuários (colaboradores) da plataforma, para banir contas que violem os termos de uso.
  + **HU10:** Como um moderador, eu quero delegar a responsabilidade de uma campanha a um órgão parceiro, para envolver instituições na gestão e distribuição das doações.
  + **HU11:** Como um órgão, eu quero aceitar a responsabilidade por uma campanha delegada, para assumir oficialmente a gestão da arrecadação e distribuição.
  + **HU12:** Como um órgão, eu quero recusar a responsabilidade por uma campanha delegada, para informar que não tenho capacidade de gerenciá-la no momento.
  + **HU13:** Como um órgão, eu quero revogar o cadastro de usuários sob minha responsabilidade, para gerenciar os membros associados à minha instituição.
  + **HU14:** Como um usuário (doador ou voluntário), eu quero visualizar a lista de campanhas ativas, para encontrar oportunidades de ajuda.
  + **HU15:** Como um usuário, eu quero buscar por campanhas ou solicitações específicas usando filtros, para encontrar rapidamente o que procuro.
  + **HU16:** Como um receptor, eu quero fazer uma solicitação específica de doação dentro da minha campanha, para detalhar os itens que mais preciso.
  + **HU17:** Como um doador, eu quero registrar a doação de um item ou serviço para uma campanha, para que o receptor e os moderadores saibam da minha contribuição.
* **Regras de negócio:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Título** | **Descrição** | **Tipo** |
| RN001 | Localização obrigatória para pedidos de ajuda | Todo pedido de ajuda só será aceito se contiver localização válida, obtida por GPS, e que seja validada e aprovada por um moderador. | Validação |
| RN002 | Pedido único por beneficiário | Cada beneficiário só poderá ter um pedido de ajuda ativo por vez, sendo identificado pelo CPF (quando fornecido) para evitar duplicidade. | Restrição de processos |
| RN003 | Dados de contato verificáveis | O sistema deve garantir que o telefone ou e-mail informados no cadastro possam ser utilizados para retorno de contato (formato válido e não vazio). | Validação |
| RN004 | Classificação de tipo de ajuda | Os pedidos devem indicar claramente o tipo de ajuda solicitada para facilitar a filtragem e a doação (ex: alimentos, roupas, higiene, entre outros). | Classificação |
| RN005 | Cadastro com papel definido | Cada usuário deve se registrar com um papel específico na plataforma: Beneficiário, Doador ou Voluntário. O sistema adaptará os recursos e permissões conforme o papel. | Regra de acesso |
| RN006 | Registro de doações com origem e destino | Toda doação registrada no sistema deve conter informações essenciais: quem doou, o que foi doado, quando foi feito, e para qual pedido ou local foi direcionada. | Regra de rastreabilidade |
| RN007 | Voluntários para entregas | Voluntários que se oferecem para realizar entregas são direcionados para os pedidos de ajuda de acordo com a proximidade de sua localização. | Regras de distribuição |
| RN008 | Geração de relatórios periódicos | Relatórios com estatísticas de atendimentos, doações realizadas e regiões com maior demanda são disponibilizados regularmente para monitoramento e transparência. | Regras de monitoramento |
| RN009 | Notificações automáticas | Mensagens são enviadas aos usuários de forma automática sempre que houver atualizações importantes em seus pedidos, doações ou campanhas. | Regra de comunicação |
| RN010 | Privacidade dos dados | Os dados sensíveis dos beneficiários (como endereço exato) só podem ser acessados por usuários autorizados (voluntários e administradores) no contexto de uma ajuda ou entrega confirmada. | Segurança |
| RN011 | Ação Requerida para Denúncias | Toda denúncia registrada deve ser analisada por um moderador em até 48 horas. A análise deve resultar em uma ação (ex: advertência, remoção de conteúdo, suspensão da conta) ou no arquivamento justificado da denúncia. | Processo de Segurança |
| RN012 | Validade e Encerramento de Campanhas | Uma campanha terá uma duração máxima de 90 dias, podendo ser encerrada antes pelo criador ou por um moderador. Após o período, será automaticamente marcada como "Finalizada". | Regra de processo |
| RN013 | Verificação de Identidade para Receptores | Para aumentar a confiança, receptores (indivíduos ou ONGs) podem passar por um processo opcional de verificação de identidade. Perfis verificados receberão um selo de destaque na plataforma. | Segurança / Confiança |
| RN014 | Padrão de Conteúdo | As descrições de campanhas e pedidos não podem conter linguagem ofensiva, discurso de ódio, informações falsas ou qualquer tipo de conteúdo ilegal. Conteúdos que violem esta regra serão removidos. | Validação de Conteúdo |

* **Requisitos não funcionais (RNF):**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Título** | **Descrição** | **Classificação** |
| RNF001 | Tempo de resposta | O sistema deve responder às requisições de busca em até 2 segundos para 90% das requisições, sob uma carga de até 500 usuários simultâneos. | Desempenho |
| RNF002 | Segurança de Dados em Trânsito | Toda a comunicação entre o cliente (navegador) e o servidor deve ser protegida com certificados de segurança válidos (HTTPS). | Segurança |
| RNF003 | Usabilidade | Usuários iniciantes devem ser capazes de completar o processo de cadastro e realizar a sua primeira ação principal (doação ou solicitação) em até 7 dias, com uma taxa de sucesso de, no mínimo, 85%. | Usabilidade |
| RNF004 | Disponibilidade | O sistema deve manter 99,9% de disponibilidade mensal, excluindo as janelas de manutenção previamente agendadas. | Confiabilidade |
| RNF005 | Compatibilidade com Navegadores | A aplicação web deve ser totalmente compatível com as versões mais recentes dos navegadores Google Chrome, Mozilla Firefox e Apple Safari, incluindo suporte total ao design responsivo. | Compatibilidade |
| RNF006 | Armazenamento Seguro de Senhas | As senhas dos usuários devem ser armazenadas no banco de dados utilizando um algoritmo de hash forte, como o bcrypt, com um custo computacional (cost factor) de, no mínimo, 10. | Segurança |
| RFN007 | Facilidade de Manutenção | A arquitetura do sistema deve permitir que correções de falhas críticas sejam implementadas em menos de 5 horas. Os testes automatizados devem cobrir, no mínimo, 80% da base de código do backend. | Manutenibilidade |
| RNF008 | Controle de Acesso (RBAC) | O sistema deve implementar um mecanismo de controle de acesso baseado em papéis (Beneficiário, Doador, Moderador etc.) para restringir o acesso às funcionalidades. | Segurança |
| RNF009 | Escalabilidade | A arquitetura do sistema deve suportar um crescimento de 20% no número de usuários e volume de dados por mês durante o primeiro ano sem degradação do desempenho. | Escalabilidade |
| RNF010 | Recuperabilidade de Dados | O sistema deve realizar backups automáticos diários do banco de dados. Em caso de falha, o tempo de recuperação (RTO) deve ser de no máximo 4 horas, com uma perda de dados máxima (RPO) de 24 horas. | Confiabilidade |
| RNF011 | Conformidade Legal (LGPD) | O sistema deve estar em conformidade com a Lei Geral de Proteção de Dados (LGPD) do Brasil, garantindo a gestão de consentimento e a segurança dos dados pessoais dos usuários. | Legal / Segurança |
| RNF012 | Acessibilidade | A aplicação web deve atender às diretrizes de acessibilidade WCAG 2.1 nível AA, garantindo que pessoas com deficiência possam utilizar a plataforma. | Acessibilidade |
| RNF013 | Auditabilidade | Ações críticas realizadas por moderadores e órgãos (ex: aprovação de campanhas, banimento de usuários) devem ser registradas em logs de auditoria para fins de rastreabilidade. | Segurança |

1. **Projeto de software**
   1. **Projeto de interface com usuário**

* FALTA
  1. **Arquitetura lógica**

**FALTA DIAGRAMA DE PACOTES LÓGICOS – DISGRAMAS DE CLASSES – ESPECIFICAÇÃO DAS TECNOLOGIAS DEFINIDAS**

A Arquitetura escolhida para o SOS Comida é uma *Arquitetura em Camadas (Layered Architecture)* com uma abordagem orientada a serviços no back-end. Este modelo foi selecionado por sua clareza na separação de responsabilidades, facilitando a manutenção, testabilidade e a escalabilidade do sistema.

A estrutura é dividida em duas grandes camadas principais:

* + Front-end: Camada de Apresentação - Uma Single Page Application (SPA) responsável por toda a interação com o usuário. Ela consome os dados fornecidos pelo backend através de uma API e renderiza as interfaces. Sua única responsabilidade é a experiência do usuário.
  + Back-end: Camada de Negócio e Dados - Responsável por todas as regras de negócio, processamento de dados e comunicação com o banco de dados. Ele expõe uma API RESTful segura para o front-end.
* Especificação das Tecnologias Definidas

|  |  |  |
| --- | --- | --- |
| **Componente** | **Tecnologia** | **Justificativa** |
| Linguagem: Back-end | Pyhton | Python fornece uma ampla variedade de utilidades no desenvolvimento web, diversas bibliotecas e frameworks robustos, além de sua facilidade de aprendizado e manutenção. |
| Framework: Back-end | Flask | O Flask foi escolhido por ser um framework leve e flexível, permitindo construir aplicações do zero de forma rápida, com boa escalabilidade e simples de ultilizar. |
| Linguagem: Front-end | Html e CSS, integrados ao Flask. | O Html e o CSS foram escolhidos por garantirem uma boa estruturação, de maneira intuitiva. Além de serem integrados ao framework, Flask, facilmente. |
| Persistência a Banco de Dados | Flask\_sqlalchemy | O sqlalchemy foi escolhido por facilitar a interação com o banco de dados, tornando o código mais legível, seguro e de fácil manutenção. Também possui integração com o framework escolhido. |
| Autenticação | Flask-login | O Flask-login foi escolhido por simplificar a implementação de autenticação e gerenciamento de sessões de usuários, garantindo segurança e integração direta com o Flask. |
| Integração de Serviços |  |  |
| Componentes (UI) |  |  |

* 1. **Arquitetura física**

Esta seção descreve a infraestrutura física e de nuvem onde o sistema SOS Comida será implantado, focando em atender aos requisitos não funcionais de disponibilidade, escalabilidade e segurança. A abordagem será totalmente baseada em um provedor de nuvem (Cloud-based), abstraindo a necessidade de gerenciamento de hardware físico.

* Descrição dos Componentes da Infraestrutura
  + *Provedor de Nuvem*: Será utilizado o Amazon Web Services (AWS) devido à sua maturidade, vasta gama de serviços gerenciados e documentação robusta.
  1. **Projeto de banco de dados**

**Falta**

1. **Construção**
   1. **Código fonte**

Esta seção define as diretrizes e ferramentas para o desenvolvimento, versionamento e garantia da qualidade do código-fonte do projeto SOS Comida.

* Repositório do Código-Fonte:
  + *Ferramenta*: Git, o sistema de controle de versão distribuído padrão da indústria.
  + *Hospedagem*: O código-fonte do frontend e do backend será hospedado em repositórios privados na plataforma GitHub. O GitHub foi escolhido por suas robustas ferramentas de colaboração, como Pull Requests, Issues e Actions para automação de CI/CD.
* Padrões de Qualidade de Código:
  1. **Teste de unidade**

Esta seção descreve a abordagem para os testes de unidade, que são cruciais para garantir a qualidade e a corretude do código em seu menor nível. Os testes serão realizados pelos próprios desenvolvedores durante os ciclos de desenvolvimento.

* Ferramenta de Teste:
* Escopo dos Teste:

Os testes de unidade focarão em validar as **regras de negócio** e a lógica contida nos serviços do backend. O objetivo é testar cada função ou método de forma isolada, garantindo que ele produza o resultado esperado para um conjunto de entradas conhecidas.

Exemplos de unidades a serem testadas:

* + Validação de Usuário: Uma função que verifica se um e-mail é válido ou se uma senha atende aos critérios de complexidade.
  + Criação de Campanha: Um teste que simula a criação de uma campanha e verifica se o status inicial é definido como "Pendente de Aprovação".
  + Regra de Negócio (RN002): Um teste para a função que verifica se um beneficiário já possui uma campanha ativa, garantindo que ele não possa criar uma nova.
  + Cálculos e Lógica: Funções que calculam o progresso de uma campanha ou que processam o status de uma doação.
* Indicadores de Qualidade e Resultado:

A qualidade do código será medida continuamente através de dois indicadores principais, que serão gerados automaticamente a cada execução da suíte de testes no ambiente de Integração Contínua (CI).

|  |  |  |
| --- | --- | --- |
| **Indicador** | **Meta (Critério de Aceitação)** | **Descrição** |
| Taxa de Sucesso | 100% | Todos os testes de unidade escritos devem passar com sucesso. Uma falha em qualquer teste impedirá a integração do novo código à branch principal (main). |
| Cobertura de Código | ≥ 80% (Backend) | Conforme o requisito **RNF007**, no mínimo 80% das linhas de código da camada de negócio e serviços do backend devem ser cobertas por testes automatizados. Isso garante que a maior parte da lógica crítica do sistema foi validada. |

* Apresentação dos Resultados:

Os resultados serão exibidos diretamente na plataforma GitHub, dentro de cada Pull Request, e em dashboards de ferramentas de Integração Contínua (como GitHub Actions), garantindo total visibilidade para a equipe de desenvolvimento.

* 1. **Interfaces com usuário**
* **Teste**
  1. **Banco de dados**
* **Teste**

1. **Teste**
   1. **Teste funcional**

* **Teste**
  1. **Teste de integração**
* **Teste**
* **Teste**
  1. **Teste de usabilidade (heurísticas de Nielsen)**
* **teste**