Universidade do Estado de Santa Catarina (UDESC)
Centro de Ciências Tecnológicas (CCT)
Bacharelado em Ciência da Computação

Disciplina: Engenharia de Software
Professor: Rebeca Schroeder Freitas

Projeto de Engenharia de Software: Plataforma de Doação

Lucas Oliveira Macedo
Matheus Azevedo de Sá
Rian Carlos Valcanaia

24 de novembro de 2025

Sumário

1 Descrição do problema **2**
1.1 Introdução 2
1.2 Stakeholders 2

2 Requisitos Funcionais **2**
2.1 Requisitos do Sistema 2
2.1.1 Requisitos Funcionais 2
2.1.2 Requisitos Não Funcionais 3
2.1.3 Requisitos Suplementares 5

3 Estimativa de Tamanho, Esforço e Duração **5**
3.1 Estimativa de Tamanho (APF) 5
3.1.1 Contagem dos Elementos 5
3.1.2 Cálculo de Pontos de Função Ajustados (PF) 6
3.2 Conversão para Linhas de Código (LOC) 8
3.3 Estimativa de Esforço e Prazo (COCOMO Básico) 8
3.3.1 Cálculo do Esforço 8
3.3.2 Cálculo da Duração 8
3.4 Conclusão da Estimativa 8

4 Diagrama de classes **9**
4.1 Arquitetura do Sistema (MVC) 9
4.2 Padrões de Projeto Aplicados 10
4.2.1 Observer 10
4.2.2 Factory Method 10
4.2.3 Iterator 10
4.3 Detalhamento das Classes e Componentes 11
4.3.1 Pacote Model 11
4.3.2 Pacote Control 11
4.3.3 Pacote Security 11
4.3.4 Pacote Presentation 12

1 Descrição do problema

1.1 Introdução

A plataforma busca solucionar dois problemas sociais: o desperdício de produtos e excedentes por parte de estabelecimentos comerciais (como alimentos próximos ao vencimento, vestuário de coleções passadas ou produtos de higiene) e a carência de itens essenciais por parte de pessoas em vulnerabilidade social.

A solução proposta é uma plataforma digital, em formato de aplicativo, que atua como uma ponte entre doadores (restaurantes, supermercados, lojas de vestuário, farmácias, etc.) com excedentes de produtos consumíveis e não consumíveis, e receptores (instituições de caridade e ONGs). A plataforma visa otimizar e formalizar o processo de doação, garantindo segurança, agilidade e rastreabilidade.

1.2 Stakeholders

Os principais interessados identificados no projeto são:

- **Doadores:** Estabelecimentos comerciais (restaurantes, supermercados, farmácias) que fornecem os itens excedentes.
- **Receptores:** Instituições de caridade e ONGs que recebem as doações para distribuição.
- **Equipe de Desenvolvimento:** Lucas Oliveira Macedo, Matheus Azevedo de Sá e Rian Carlos Valcanaia, responsáveis pela implementação e manutenção do sistema.
- **Administradores da Plataforma:** Responsáveis pela validação de cadastros e auditoria de segurança (conforme requisitos de LGPD).

2 Requisitos Funcionais

2.1 Requisitos do Sistema

A seguir, são detalhados os requisitos que orientaram o projeto da plataforma e:

2.1.1 Requisitos Funcionais

- **F1. Cadastro de usuário:** O sistema deve permitir que novos usuários se cadastrem como doadores ou como receptores.
- **F2. Gerenciamento de perfil:** O sistema deve permitir que os usuários editem suas informações de perfil.
- **F3. Cadastro de doação:** O sistema deve permitir que os estabelecimentos doadores registrem os tipos e a quantidade de itens que desejam doar.
- **F4. Agendamento de doações:** O sistema deve permitir o agendamento da data e horário para a retirada dos itens doados.

- **F5. Consulta de doações:** O sistema deve permitir a visualização das doações registradas e agendadas.
- **F6. Notificação de doação:** O sistema deve enviar notificações aos receptores sobre as doações disponíveis.
- **F7. Solicitação de doação:** O sistema deve permitir que os receptores solicitem as doações disponíveis.
- **F8. Rastreamento de doações:** O sistema deve permitir o rastreamento do status da doação.
- **F9. Geração de relatório:** O sistema deve gerar relatórios sobre o volume de itens doados, a quantidade de doadores ativos e a frequência de doações.
- **F10. Avaliação e feedback:** O sistema deve permitir que doadores e receptores se avaliem mutuamente.
- **F11. Pesquisa de doadores:** O sistema deve permitir que receptores pesquisem por doadores em sua proximidade ou por tipo de item doado.

2.1.2 Requisitos Não Funcionais

- **F1. Cadastro de usuário:**
  - **RNF1.1. Usabilidade:** O processo de cadastro deve ser simples e intuitivo.
  - **RNF1.2. Segurança:** As senhas dos usuários devem ser criptografadas.
  - **RNF1.3. Desempenho:** O tempo de resposta para a conclusão do cadastro não deve exceder 2 segundos.
- **F2. Gerenciamento de Perfil:**
  - **RNF2.1. Usabilidade:** A interface para edição do perfil deve ser clara e organizada.
  - **RNF2.2. Confiabilidade:** O sistema deve garantir que as alterações no perfil sejam salvas e consistentes.
  - **RNF2.3. Segurança:** A edição de informações sensíveis deve exigir reautenticação.
- **F3. Cadastro de doação:**
  - **RNF3.1. Usabilidade:** O formulário de cadastro de doação deve ser fácil de preencher.
  - **RNF3.2. Confiabilidade:** O sistema deve garantir que o cadastro da doação seja armazenado de forma segura.
- **F4. Agendamento de doação:**
  - **RNF4.1. Usabilidade:** A interface de agendamento deve ser um calendário visual.

  - **RNF4.2. Desempenho:** A verificação de disponibilidade de horários deve ser rápida.

- **F5. Consulta de doações:**
  - **RNF5.1. Usabilidade:** A visualização das doações deve ser organizada e permitir filtros.
  - **RNF5.2. Desempenho:** O tempo de carregamento da lista de doações deve ser inferior a 3 segundos.

- **F6. Notificação de doação:**
  - **RNF6.1. Confiabilidade:** As notificações devem ser entregues de forma consistente e em tempo real.
  - **RNF6.2. Configurabilidade:** O usuário deve poder configurar o tipo de notificação que deseja receber.

- **F7. Solicitação de doação:**
  - **RNF7.1. Usabilidade:** O processo de solicitação deve ser um "fluxo de um clique".
  - **RNF7.2. Confiabilidade:** A solicitação deve ser processada e confirmada sem falhas.

- **F8. Rastreamento de Doação:**
  - **RNF8.1. Usabilidade:** A interface de rastreamento deve ser um mapa ou painel visual.
  - **RNF8.2. Confiabilidade:** O rastreamento deve ser preciso e atualizado em tempo real.

- **F9. Geração de relatórios:**
  - **RNF9.1. Usabilidade:** A interface de relatórios deve apresentar os dados de forma visual.
  - **RNF9.2. Desempenho:** A geração de um relatório complexo não deve demorar mais de 10 segundos.

- **F10. Avaliação e feedback:**
  - **RNF10.1. Usabilidade:** O formulário de avaliação deve ser fácil e rápido de preencher.
  - **RNF10.2. Confiabilidade:** As avaliações e feedbacks devem ser armazenados de forma segura.

- **F11. Pesquisa de doadores:**
  - **RNF11.1. Usabilidade:** A função de pesquisa deve ser responsiva, com filtros avançados.
  - **RNF11.2. Desempenho:** A pesquisa deve retornar resultados em no máximo 5 segundos.

2.1.3 Requisitos Suplementares

- **S1. Compatibilidade:** O sistema deve ser acessível e funcional em dispositivos móveis (iOS e Android) e navegadores web populares.
- **S2. Escalabilidade:** A arquitetura deve suportar um aumento de 200% no número de usuários e transações anuais sem degradação de desempenho.
- **S3. Segurança:** O sistema deve estar em conformidade com a LGPD e adotar medidas de segurança robustas.
- **S4. Manutenibilidade:** A base de código deve ser modular e bem documentada para facilitar futuras manutenções.
- **S5. Padrões de interface:** O design da interface do usuário deve seguir diretrizes de UX para garantir consistência.
- **S6. Tolerância a falhas:** O sistema deve ser projetado para lidar com falhas de componentes sem interromper as operações.
- **S7. Backup e recuperação:** O sistema deve implementar uma política de backup automático dos dados com capacidade de recuperação.
- **S8. Documentação:** A documentação do sistema deve incluir manuais de usuário e guias para desenvolvedores.
- **S9. Suporte a múltiplos idiomas:** O sistema deve permitir que a interface seja utilizada em diferentes idiomas.
- **S10. Disponibilidade:** O sistema deve garantir uma disponibilidade mínima de 99,9%.

3 Estimativa de Tamanho, Esforço e Duração

Para a estimativa deste projeto, utilizou-se uma abordagem híbrida baseada em métricas paramétricas: a **Análise de Pontos de Função (APF)** para determinar o tamanho funcional do software e o modelo **COCOMO (Constructive Cost Model)** para calcular o esforço e o cronograma.

3.1 Estimativa de Tamanho (APF)

A Análise de Pontos de Função mede o tamanho do software quantificando as funcionalidades fornecidas ao usuário. O cálculo baseou-se na contagem dos elementos identificados nos Requisitos Funcionais (F1 a F11) e no Modelo de Dados.

3.1.1 Contagem dos Elementos

Classificamos os elementos com complexidade "Baixa"para obter uma estimativa ponderada:

- **Arquivos Lógicos Internos (ALI):** 19, 11 Entidades (Endereço, Usuário, Avaliação, Doador, Receptor, Doação, Solicitação, Item Doado, Alimento, Vestuário e Produto de Higiêne) + 8 Relações (CadastraDoação, EndereçoUsuário, UsuárioFazAvaliacao, UsuárioRecebeAvaliação, Solicita, EndereçoColeta, ItemDoado e SolicitaDoação).
  Cálculo: 19×7 (Peso Baixo) = 133 Pontos.

- **Entradas Externas (EE):** 5 Processos de entrada (F1 - Cadastrar Usuário, F3 - Cadastrar Doação, F4 - Agendar, F7 - Solicitar, F10 - Avaliar).
  Cálculo: 5×3 (Peso Baixo) =15 Pontos.

- **Saídas Externas (SE):** 2 Saídas processadas (F9 - Relatórios, F6 - Notificações).
  Cálculo: 2×4 (Peso Baixo) = 8 Pontos.

- **Consultas Externas (CE):** 2 Consultas principais (F5 - Consulta de Doações, F11 - Pesquisa).
  Cálculo: 2×3 (Peso Baixo) = 6 Pontos.

- **Arquivos de Interface Externos (AIE):** 5 Interfaces externas:
  - API de Consulta de CEP (Google Maps).
  - API de Autenticação (OAuth Google).
  - API de Segurança anti-robô (reCAPTCHA).
  - API de Armazenamento em Nuvem (Google Cloud Storage).
  - API de E-mail Transacional (SendGrid).

  Cálculo: 5×5 (Peso Baixo) = 25 Pontos.

Total de Pontos de Função Não Ajustados: $133 + 15 + 8 + 6 + 25 = \mathbf{187}$.

3.1.2 Cálculo de Pontos de Função Ajustados (PF)

Para transformar os Pontos de Função Não Ajustados em Pontos de Função Ajustados, calculou-se o **Fator de Ajuste (FA)** com base nas 14 Características Gerais do Sistema.

A Tabela 1 detalha a pontuação atribuída (0 a 5) com base nos requisitos do projeto.

Tabela 1: Pontuação das Características Gerais do Sistema

Questão	**Nota**	**Justificativa (Baseada nos Requisitos)**
1. Backup e recuperação	5	O requisito S7 exige explicitamente política de backup automático e recuperação de dados.
2. Comunicação de dados	4	Integração com múltiplas APIs externas (Google, SendGrid) e notificações em tempo real (RNF6.1).
3. Processamento distribuído	3	Arquitetura em nuvem com serviços externos.
4. Performance	4	Requisitos estritos ($< 2s$ cadastro, $< 3s$ carregamento).
5. Muitos usuários em paralelo	3	Requisito S2 prevê escalabilidade de 200%.
6. Entrada de dados on-line	5	Funções principais são transacionais via interface.
7. Multiplas telas online	5	O sistema possui fluxos distintos para Cadastro, Doação, Agendamento e Avaliação.
8. Atualização de entidades on-line	5	Atualizações de perfil e status em tempo real.
9. E/S complexas	3	Geração de relatóios visuais e filtros avançados de pesquisa.
10. Processamento complexo	4	Geolocalização e filtros avançados.
11. Reusabilidade de código	3	Código modular (Requisito S4).
12. Migração e instalação	1	Implementação centralizada (Saas) não necessitando migração.
13. Múltiplos locais	1	Implantação centralizada (SaaS).
14. Facilidade de mudança	4	Padrões de interface e manutenibilidade.
Total ($\sum Ci$)	**50**	

O Fator de Ajuste é calculado pela fórmula:

$$FA = 0,65 + (0,01 \times \sum Ci)$$

$$FA = 0,65 + (0,01 \times 50) = \mathbf{1,15}$$

Aplicando o ajuste ao tamanho funcional:

$$PF = PFNA \times FA$$

$$PF = 187 \times 1,15 \approx \mathbf{215} \text{ Pontos de Função}$$

3.2 Conversão para Linhas de Código (LOC)

Para aplicar o modelo COCOMO, converteu-se a métrica funcional em Linhas de Código. Utilizou-se a contagem ajustada (PF) e o fator de conversão para a linguagem **Java** (53 LOC/PF).

$$Tamanho(KLOC) = \frac{215 \times 53}{1000} \approx \mathbf{11,4} \text{ KLOC}$$

3.3 Estimativa de Esforço e Prazo (COCOMO Básico)

Aplicou-se o modelo COCOMO no modo simples (*Organic Mode*), utilizando os coeficientes padrão ($a = 2,4$, $b = 1,05$, $c = 2,5$, $d = 0,38$).

3.3.1 Cálculo do Esforço

$$Esforço = 2,4 \times (KLOC)^{1,05}$$

$$Esforço = 2,4 \times (11,4)^{1,05} \approx \mathbf{30,9} \text{ pessoas-mês}$$

3.3.2 Cálculo da Duração

$$Duração = 2,5 \times (Esforço)^{0,38}$$

$$Duração = 2,5 \times (30,9)^{0,38} \approx \mathbf{9,21} \text{ meses}$$

3.4 Conclusão da Estimativa

O projeto da Plataforma de Doação possui um tamanho funcional ajustado de **215 PF**. As estimativas indicam um prazo de desenvolvimento de aproximadamente **9 meses**, exigindo um esforço total de cerca de **31 pessoas-mês**.

4 Diagrama de classes

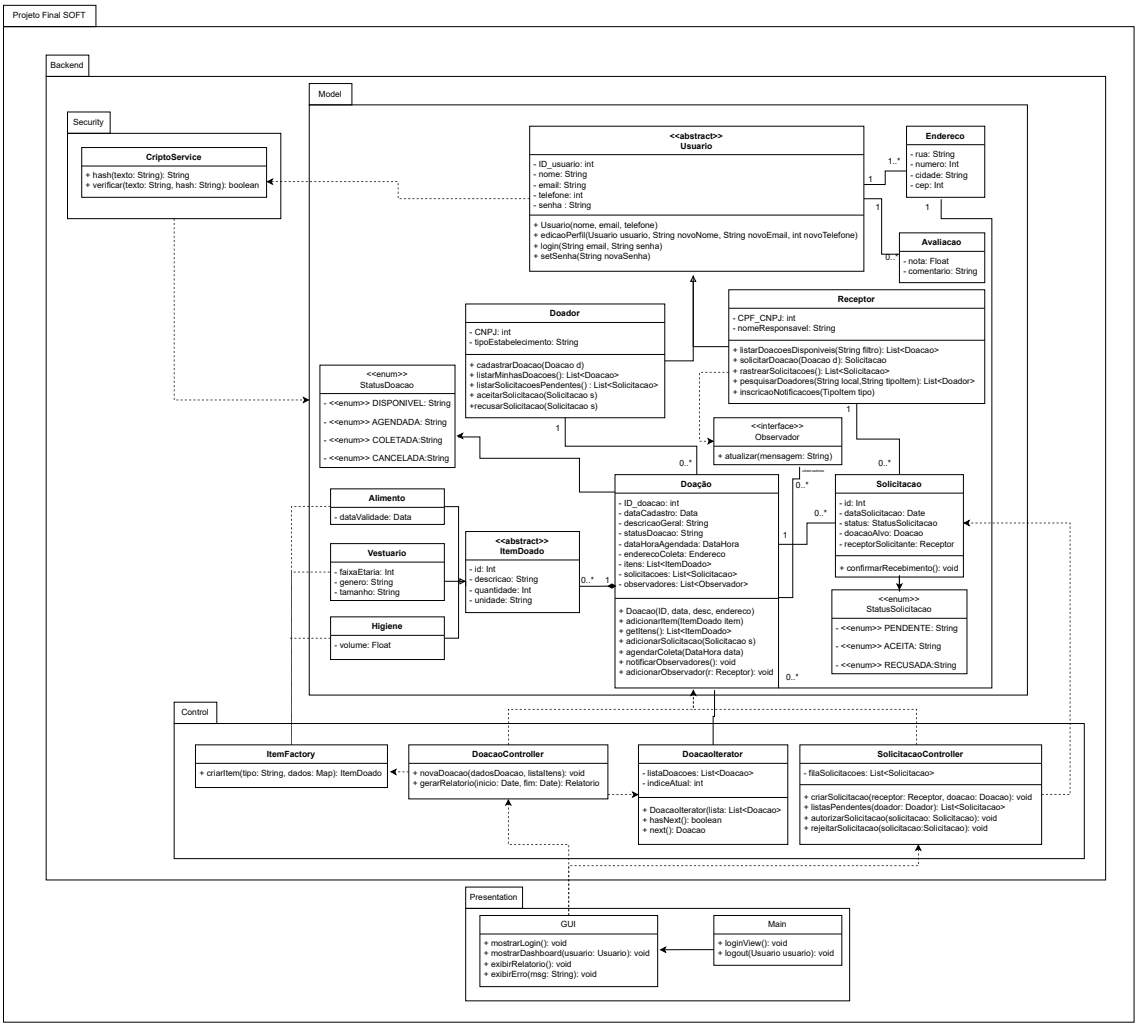

Figura 1: Diagrama de Classes do Projeto

4.1 Arquitetura do Sistema (MVC)

O sistema segue o padrão arquitetural **MVC (Model-View-Controller)**, o que garante a separação de responsabilidades entre a interface do usuário, a lógica de negócios e os dados. No diagrama, essa separação é evidenciada pelos pacotes:

- **Model (Modelo):** Contém as classes que representam as entidades do domínio (ex: `Usuario`, `Doacao`, `ItemDoado`) e as regras de negócio fundamentais. É responsável por gerenciar os dados e notificar a camada de visão sobre alterações de estado (via padrão Observer).
- **Control (Controlador):** Atua como intermediário entre a View e o Model. As classes `DoacaoController` e `SolicitacaoController` processam as entradas do usuário, invocam a lógica de negócios no modelo e decidem qual resposta apresentar.

- **Presentation (Visão/View):** Representada pelo pacote `GUI` e a classe `Main`, é responsável pela interação com o usuário final, exibindo dados e capturando comandos que são repassados aos controladores.

Além do MVC, temos o pacote **Security**, que fornece serviços transversais de criptografia e validação, garantindo a segurança dos dados sensíveis do usuário.

4.2 Padrões de Projeto Aplicados

A análise do diagrama revela a aplicação de três padrões de projeto *GoF (Gang of Four)*:

4.2.1 Observer

O padrão Observer foi implementado para gerenciar as notificações de novas doações.

- **Participantes:** A classe `Doacao` atua como o *Subject* (sujeito), mantendo uma lista de observadores. A interface `Observador` é implementada pela classe `Receptor`.
- **Funcionamento:** Quando uma doação sofre alterações ou precisa ser divulgada, o método `notificarObservadores()` é chamado, disparando o método `atualizar()` em todos os receptores inscritos. Isso permite um acoplamento fraco entre o objeto de doação e os interessados nela.

4.2.2 Factory Method

Utilizado para encapsular a lógica de criação de itens de doação.

- **Participantes:** A classe `ItemFactory` possui o método `criarItem()`.
- **Funcionamento:** Baseado em um parâmetro de tipo (String), a fábrica decide qual subclasse concreta de `ItemDoado` instanciar (`Alimento`, `Vestuario` ou `Higiene`), abstraindo a complexidade de criação das classes clientes.

4.2.3 Iterator

Empregado para percorrer coleções de doações sem expor a estrutura interna da lista.

- **Participantes:** A classe `DoacaoIterator`.
- **Funcionamento:** Fornece métodos padronizados como `hasNext()` e `next()` para iterar sequencialmente sobre a lista de doações, facilitando a navegação pelos dados na camada de controle.

4.3 Detalhamento das Classes e Componentes

4.3.1 Pacote Model

Este pacote contém o núcleo da lógica de negócios.

- **Usuario (Abstrata):** Classe base para os atores do sistema. Contém atributos comuns como nome, email, telefone e senha. Define métodos para login e edição de perfil.
- **Doador:** Especialização de usuário. Possui CNPJ/CPF e métodos para cadastrar doações (`cadastrarDoacao`), listar suas doações e gerenciar solicitações recebidas (`aceitarSolicitacao`, `recusarSolicitacao`).
- **Receptor:** Especialização de usuário que implementa a interface `Observador`. Pode solicitar doações, rastreá-las e receber notificações de disponibilidade através do método `atualizar()`.
- **Doacao:** Entidade central que agrega itens e gerencia o ciclo de vida da oferta. Possui atributos de data, descrição, status e endereço de coleta. Relaciona-se com a classe `Solicitacao` e gerencia a lista de observadores.
- **ItemDoado (Abstrata):** Generalização dos itens. Subclasses concretas:
  - **Alimento:** Possui data de validade.
  - **Vestuario:** Possui faixa etária, gênero e tamanho.
  - **Higiene:** Possui volume.
- **Solicitacao:** Representa o pedido de um Receptor para uma Doação. Controla o status do pedido (Pendente, Aceita, Recusada) através do Enum `StatusSolicitacao`.
- **Endereco e Avaliacao:** Classes auxiliares que compõem os dados de usuários e transações.

4.3.2 Pacote Control

Responsável pela orquestração das operações.

- **DoacaoController:** Gerencia o fluxo de criação de doações. Utiliza o método `novaDoacao` para instanciar dados e `gerarRelatorio` para extração de informações gerenciais. Interage com o `DoacaoIterator`.
- **SolicitacaoController:** Gerencia a fila de solicitações. Possui métodos para `criarSolicitacao`, listar pendências e arbitrar (autorizar ou rejeitar) os pedidos.
- **ItemFactory:** Classe auxiliar de controle para a fabricação de objetos do tipo item.

4.3.3 Pacote Security

- **CriptoService:** Fornece métodos estáticos para segurança da informação, como `hash(texto)` para criptografar senhas antes da persistência e `verificar()` para autenticação segura.

4.3.4 Pacote Presentation

- **GUI / Main:** Pontos de entrada da aplicação. Métodos como `mostrarLogin`, `mostrarDashboard` e `exibirRelatorio` indicam o fluxo de navegação do usuário.