CENTRO UNIVERSITÁRIO SENAC

SANTO AMARO

PROJETO INTEGRADOR III: IMPLANTAÇÃO DE BANCO DE DADOS

Bruno Costa Caiado

Felipe Augusto Santinho

Isabel Helena Hartmann

Paula Barros Ortiz

Thais Oliveira dos Santos

EAD - ENSINO À DISTÂNCIA

2024

Bruno Costa Caiado

Felipe Augusto Santinho

Isabel Helena Hartmann

Paula Barros Ortiz

Thais Oliveira dos Santos

PROJETO INTEGRADOR III: IMPLANTAÇÃO DE BANCO DE DADOS

Projeto Integrador III – Fase 1 apresentado ao Centro Universitário Senac, como exigência parcial para obtenção de aprovação na disciplina Projeto Integrador III, do curso de Banco de Dados.

Orientador (a): Prof. (ª) Alexandre Lopes Machado.

EAD - ENSINO À DISTÂNCIA

2024

RESUMO

O projeto Alimenta+ tem como objetivo melhorar a distribuição de alimentos, com foco nas regiões Norte e Nordeste do Brasil, onde há maior incidência de pobreza e insegurança alimentar. A plataforma foi desenvolvida para otimizar o gerenciamento de doações de alimentos para instituições de caridade, utilizando um banco de dados relacional que combina princípios de normalização e desnormalização para garantir tanto a integridade dos dados quanto a eficiência das consultas.

A plataforma oferece funcionalidades como cadastro de doadores, monitoramento de doações, integração com logística e transparência no processo de doação. Além de combater a fome, o Alimenta+ promove práticas sustentáveis e solidárias, incentivando a doação consciente e a construção de uma rede de apoio mais eficiente. A aplicação de estratégias de desnormalização em tabelas estratégicas permitiu melhorar o desempenho do sistema, simplificando operações e tornando a plataforma mais ágil para os usuários.

Palavras-chave: 1. Alimenta+. 2. Insegurança alimentar. 3. Plataforma digital. 4. Banco de dados relacional. 5. Monitoramento de doações. 6. Transparência.

ABSTRACT

The Alimenta+ project aims to improve food distribution, focusing on the North and Northeast regions of Brazil, where there is a higher incidence of poverty and food insecurity. The platform was developed to update the management of food drives for charities, using a related database that combines normalization and denormalization principles to ensure both data integrity and query efficiency.

The platform offers features such as donor registration, donation monitoring, integration with logistics, and transparency in the donation process. In addition to combating hunger, Alimenta+ promotes sustainable and solidarity-based practices, encouraging conscious donations and building a more efficient support network. The application of denormalization strategies in strategic tables improved system performance, simplified operations, and made the platform more agile for users.

Keywords: 1. Alimenta+. 2. Food insecurity. 3. Digital platform. 4. Relational database. 5. Stock monitoring. 6. Transparency.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

A insegurança alimentar é uma realidade que afeta bilhões de pessoas em todo o mundo. Segundo Guitarrara (2022), dados da ONU revelam que 735 milhões de pessoas sofrem de fome diariamente, enquanto 2,3 bilhões vivem em situação de insegurança alimentar. No Brasil, os números são alarmantes: mais de 21 milhões de pessoas passam fome diariamente, e cerca de 70,3 milhões enfrentam insegurança alimentar, incluindo 10 milhões de desnutridos. Embora a produção mundial de alimentos seja suficiente para atender a população global, a má distribuição e a ausência de políticas públicas eficientes aprofundam essa crise, especialmente nas regiões mais vulneráveis, como o Norte e o Nordeste do país.

Para contribuir com o combate à insegurança alimentar no Brasil, o projeto Alimenta+ foi desenvolvido com o objetivo de aprimorar a gestão e a distribuição de alimentos e outros recursos entre doadores e organizações não governamentais (ONGs). A plataforma visa proporcionar um sistema de distribuição mais eficiente e equitativo, incentivando o acesso a alimentos por meio de uma gestão integrada e sustentável. O sistema permite o cadastro de doadores, a definição de locais de coleta e o controle do ciclo de doações, desde a oferta inicial até a entrega aos beneficiários. Essa estrutura facilita uma alocação mais racional dos recursos, promovendo práticas conscientes de consumo e descarte.

A base técnica do Alimenta+ é sustentada por um banco de dados relacional desenvolvido para otimizar o gerenciamento de doações e garantir a consistência das informações. Inicialmente, o banco de dados foi projetado seguindo princípios de normalização até a Terceira Forma Normal (3FN), reduzindo redundâncias e assegurando a integridade dos dados, conforme destacado por Rozza (2023). No entanto, para melhorar o desempenho de consultas frequentes, estratégias de desnormalização foram aplicadas em tabelas-chave, como Doação, Pedido e Alimento. Por exemplo, colunas como nome_doador, nome_ong e nome_categoria foram adicionadas diretamente nessas tabelas, eliminando a necessidade de JOINs com tabelas relacionadas. Triggers foram implementados para manter a consistência

dos dados desnormalizados. Essa abordagem trouxe ganhos significativos em desempenho e simplificação de consultas, mantendo a integridade do sistema.

Além do projeto lógico, o banco de dados inclui um Diagrama Entidade-Relacionamento (DER), que ilustra os relacionamentos entre as entidades e proporciona uma visão clara do sistema. Foram desenvolvidos scripts em Data Definition Language (DDL) para implementar a estrutura em um Sistema de Gerenciamento de Banco de Dados (SGBD), tornando o sistema escalável e preparado para futuras adaptações conforme a demanda.

O Alimenta+ é uma solução tecnológica que utiliza a normalização e o uso de SQL para estruturar e apoiar o combate à fome e à desnutrição no Brasil. Com uma base de dados organizada e robusta, o sistema promove uma gestão eficaz de recursos alimentares, apoiando as ONGs e ampliando o impacto das ações contra a insegurança alimentar nas regiões mais vulneráveis. Ao aliar tecnologia e responsabilidade social, o projeto oferece um modelo de intervenção sustentável, alinhado com as necessidades de um gerenciamento eficiente e ético da distribuição de alimentos.

O objetivo é auxiliar nos esforços para a diminuição do índice de insegurança alimentar moderada e grave nas regiões mais afetadas no Brasil, como o Norte e Nordeste, por meio da criação de um sistema que facilite a cadeia de doações de alimentos para as comunidades que sofrem diretamente com os efeitos mais graves da fome.

Dessa forma, visa contribuir para a diminuição da escassez de recursos alimentares nessas regiões, proporcionando um sistema que facilite a distribuição, o gerenciamento e o acesso a alimentos de forma mais eficiente e equitativa.

Por fim, incentivar o descarte consciente e garantir a conexão ágil entre usuários, ecopontos e empreendedores, promovendo a integração de empreendimentos locais e a comunidade.

1. CONCEPÇÃO

1.1 Levantamento e análise das necessidades dos usuários

A insegurança alimentar, que afeta milhões de brasileiros, revela a necessidade crítica de uma melhor gestão e distribuição de recursos alimentares. Embora haja abundância na produção de alimentos, há má distribuição e ausência de políticas públicas eficientes mantêm o problema. ONGs e organizações comunitárias que atuam no combate à fome precisam de uma plataforma que facilite a doação e o direcionamento de alimentos para áreas vulneráveis. O projeto Alimenta+ atende essas demandas, conectando doadores e ONGs, melhorando a eficiência de distribuição e promovendo uma gestão consciente dos recursos alimentares.

1.2 Conceitos da Aplicação

O Alimenta+ é uma plataforma digital que visa centralizar e otimizar o processo de doação e distribuição de alimentos. Ele utiliza um banco de dados relacional, inicialmente projetado com técnicas de normalização (até a Terceira Forma Normal) para assegurar a consistência e integridade dos dados. Para melhorar o desempenho de consultas frequentes, estratégias de desnormalização foram aplicadas em tabelas-chave, como Doação, Pedido e Alimento, permitindo acesso direto a informações como nome do doador, nome da ONG e categoria do alimento, sem a necessidade de JOINs complexos. A plataforma permite que doadores se cadastrem, definam locais de coleta e acompanhem o ciclo das doações até a entrega final, mantendo um fluxo transparente e auditável.

1.3 Em nível de detalhe suficiente para justificar a especificação de um produto de software

O Alimenta+ oferece funcionalidades para o cadastro de doadores e entidades, definição de pontos de coleta e monitoramento de doações em tempo real. O uso de SQL e um Sistema de Gerenciamento de Banco de Dados (SGBD) escalável torna o sistema preparado para futuros aumentos na demanda, possibilitando expansões e adaptações. Os scripts em Data Definition Language (DDL) permitem fácil implementação e manutenção da estrutura no SGBD.

2. ELABORAÇÃO

2.1 Levantamento e análise dos requisitos

A Plataforma Alimenta+ tem como objetivo conectar doadores de alimentos a organizações não governamentais (ONGs) e voluntários, com o propósito de combater a insegurança alimentar, especialmente nas regiões Norte e Nordeste do Brasil. Para atender a essas necessidades, foi realizada uma análise dos requisitos funcionais e não funcionais, com foco nas funcionalidades essenciais, nas interfaces de usuário e nas exigências técnicas.

2.2 Levantamento detalhado das funções

A plataforma deve incorporar funcionalidades fundamentais para garantir o processo eficiente e seguro de doação de alimentos:

Cadastro de Usuários: Deve permitir o cadastro de diferentes tipos de usuários (doador, ONG, voluntário, restaurante), com verificação de dados antes da ativação da conta, garantindo a confiabilidade das informações.

Cadastro de Alimentos: O sistema permitirá que os doadores registrem alimentos com detalhes como tipo, quantidade, validade e categoria. O sistema deve garantir que apenas alimentos dentro do prazo de validade sejam registrados.

Gestão de Pedidos de Doação: As ONGs poderão solicitar alimentos disponíveis, com possibilidade de aprovação ou rejeição dos pedidos por parte dos doadores. O status do pedido deve ser acompanhado em tempo real.

Gestão de Doações: O doador pode registrar as ofertas de alimentos, que serão gerenciadas até a coleta e entrega. A plataforma deve integrar mapas para facilitar a logística de coleta e entrega.

2.3 Levantamento das interfaces

A interface da plataforma precisa ser simples, intuitiva e acessível a todos os tipos de usuários. As principais interfaces incluem:

Cadastro de Usuários: Formulários de cadastro detalhados para doadores, ONGs, restaurantes e voluntários, com validação de dados antes da ativação da conta.

Cadastro de Alimentos: Interface simples para o registro de alimentos, com campos para tipo, quantidade, validade e categoria. O sistema deve alertar o usuário caso o alimento registrado seja invalido.

Gestão de Pedidos: Tela para que ONGs solicitem alimentos, visualizem a quantidade e tipos disponíveis, e acompanhem o status de seus pedidos.

2.4 Requisitos não funcionais

A plataforma deve garantir alto desempenho, com respostas rápidas para todas as operações realizadas pelos usuários. A segurança é uma prioridade, com medidas rigorosas para proteger dados pessoais e controlar o acesso a informações sensíveis. A disponibilidade do sistema deve ser de 99%, assegurando operação contínua e minimizando interrupções.

A interface deve ser intuitiva e acessível, atendendo a diferentes perfis de usuários. A plataforma deve ser escalável, com capacidade de lidar com o aumento de usuários e volume de dados à medida que cresce. Além disso, é fundamental que a plataforma esteja em conformidade com as normas legais relacionadas à doação de alimentos, incluindo alertas sobre validade dos produtos e a conformidade com as regulamentações locais.

A confiabilidade do sistema é fundamental, garantindo que os dados sejam mantidos íntegros e que o funcionamento da plataforma seja contínuo e sem falhas. O fluxo de doações deve ser claro e eficiente, com boa comunicação entre doadores, ONGs e voluntários.

3. MODELO DE DADOS

Os modelos de dados são cruciais para a organização, gestão e compreensão dos dados em sistemas de informação, desempenhando um papel fundamental em todo o ciclo de vida do desenvolvimento de software e gestão de dados (GUEDES, 2018). Com relação ao modelo de dados, foi realizado um modelo que representa a estrutura básica de um sistema de doações, cada entidade representa um conceito do mundo real e os atributos descrevem suas características. As relações mostram como as entidades se conectam.

3.1 Conceitos e Entidades

Com base nas especificações de requisitos, os conceitos do sistema foram definidos com base nas necessidades operacionais do Alimenta+. Cada conceito é representado por uma entidade, que inclui atributos que descrevem suas características, são eles:

1. Doador: ID (PK), Nome, Email, Telefone, Tipo (Pessoa Física, Restaurante, ONG, Voluntário);

2. Entidade Recebedora: ID (PK), Nome, CNPJ, Contato e Status (ativo, inativo);

3. Alimento: ID (PK), Nome, Quantidade, Data_Validade, Tipo (Frutas, Legumes, Marmita), Categoria_ID (FK) e Nome_Categoria;

4. Categoria_Alimento: ID (PK), Nome e Descrição;

5. Local_Coleta: ID (PK), Endereco_ID (FK), Horario_Funcionamento, Tipo (pedido ou doação) e Regiao_ID (FK);

6. Regiao: ID (PK), Nome e Descrição;

7. Doação: ID_Doacao (PK), Data_Doacao, Doador_ID (FK), Alimento_ID (FK), Local_Coleta_ID (FK) e Nome_Doador;

8. Pedido: ID_Pedido (PK), Data_Pedido, ONG_ID (FK) e Status_ID (FK) (Pendente, Aprovado, Rejeitado, Entregue);

9. Status_Pedido: ID (PK), Status_ID, Nome_ONG e Status_Pedido;

10. Item_Pedido: ID (PK), Pedido_ID (FK), Alimento_ID (FK) e Quantidade;

11. Entrega: ID (PK), Data_Entrega, Pedido_ID (FK);

12. Endereço: ID (PK), Logradouro, Número, Bairro, Cidade, CEP e Complemento.

3.2 Associações entre os Conceitos

As relações entre as entidades refletem os processos do sistema:

Doador - Doação: Um doador pode realizar várias doações.

Doação - Alimento: Cada doação está associada a um ou mais alimentos.

Doação - Local de Coleta: Define onde as doações são coletadas.

Pedido - Entidade Recebedora: Representa o vínculo entre pedidos e ONGs.

Pedido - Item do Pedido: Cada pedido é detalhado em itens específicos.

Entrega - Pedido: Relaciona a entrega de alimentos com o pedido correspondente.

4. REGRAS DE NEGÓCIO

4.1 Cadastro e Validação de Alimentos

Objetivo: Garantir que apenas alimentos em boas condições e dentro do prazo de validade sejam registrados no sistema de doações.

Descrição: Alimenta+ deve validar os alimentos antes de serem registrados no sistema, assegurando que as doações estejam dentro das normas de segurança alimentar e evitando a distribuição de produtos impróprios para consumo. Para isso, a regra de negócio deve verificar os seguintes critérios:

Validade do Alimento

Ao cadastrar um alimento, o sistema verifica automaticamente a data de validade, se a data de validade do alimento estiver expirada, o sistema rejeita a entrada do alimento, notificando o usuário sobre a irregularidade.

Tipo de Alimento

O sistema deve permitir o cadastro de diferentes tipos de alimentos (frutas, legumes, marmitas, etc.) e garantir que as informações sejam completas (tipo, quantidade, validade, categoria).A categoria de alimentos deve ser validada com base em um conjunto predefinido de tipos, como "frutas", "legumes", "cereais", "marmitas", etc. Cada tipo de alimento deve ser associado a uma categoria específica.

Quantidade de Alimento

O sistema deve garantir que a quantidade informada para doação seja positiva e não exceda os limites estabelecidos para o tipo de alimento.

Notificação de Irregularidades

Quando um alimento não atender aos critérios de validade ou categoria, o sistema deverá gerar uma notificação informando o erro e solicitando correção antes de permitir a submissão da doação.

Impacto

Garante que os alimentos que chegam às ONGs e beneficiários sejam seguros para consumo, evitando problemas de saúde pública, melhora a eficiência da plataforma ao assegurar que os dados de alimentos sejam consistentes e dentro das normas estabelecidas e contribui para a gestão responsável de recursos alimentares no combate à insegurança alimentar.

4.2 Conceitos do Negócio

Os conceitos de negócio do Alimenta+ estruturam suas operações para combater a insegurança alimentar de forma eficiente. O doador, representado por pessoas físicas, restaurantes, ONGs ou voluntários, é responsável pelo fornecimento de alimentos destinados às entidades recebedoras, que distribuem esses recursos aos beneficiários. Essas entidades possuem status de atividade, podendo ser ativas ou inativas.

Os alimentos, núcleo do processo, são classificados por tipo (como frutas ou marmitas) e organizados em categorias para facilitar sua gestão. A plataforma valida dados como data de validade e quantidade, assegurando a segurança alimentar. Os locais de coleta funcionam como pontos de entrega ou retirada, vinculados a regiões específicas para otimizar o transporte e armazenamento.

Pedidos, feitos pelas entidades recebedoras, detalham os alimentos e suas quantidades, sendo acompanhados por um status que reflete seu progresso. Após aprovação, as entregas conectam os pedidos às entidades. Com categorias organizadas e endereços precisos, o sistema promove eficiência logística,

conectando doadores a entidades de forma responsável e garantindo alimentos seguros para quem mais precisa.

4.3 Custos estimados (Dados fictícios)

Estimativa de Custos – Projeto Alimenta+

Esta seção apresenta uma estimativa dos custos para o desenvolvimento, infraestrutura, marketing e manutenção do Projeto Alimenta+.

Desenvolvimento do Sistema

- Desenvolvedores (Full Stack, Backend, Frontend): R$ 8.000,00/mês (2 desenvolvedores por 3 meses) → R$ 48.000,00
- Designer UI/UX: R$ 5.000,00/mês (1 profissional por 2 meses) → R$ 10.000,00
- Testes e Qualidade (QA): R$ 4.000,00/mês (1 profissional por 2 meses) → R$ 8.000,00

Infraestrutura e Tecnologia

- Servidor Cloud (AWS, Azure, ou similar): R$ 1.500,00/mês (12 meses) → R$ 18.000,00
- Banco de Dados (SQL/PostgreSQL/MongoDB): R$ 800,00/mês (12 meses) → R$ 9.600,00
- Domínio e Hospedagem: R$ 500,00/ano

Marketing e Divulgação

- Criação de Identidade Visual: R$ 3.000,00
- Anúncios e Publicidade (Google Ads, Meta Ads): R$ 2.000,00/mês (6 meses) → R$ 12.000,00

- Redes Sociais e Gestão de Conteúdo: R$ 1.500,00/mês (6 meses) → R$ 9.000,00

Manutenção e Suporte

- Equipe de Suporte Técnico: R$ 5.000,00/mês (12 meses) → R$ 60.000,00
- Atualizações e Melhorias: R$ 10.000,00/ano

Custo Total Estimado

- Desenvolvimento: R$ 66.000,00
- Infraestrutura: R$ 28.100,00
- Marketing: R$ 24.000,00
- Manutenção: R$ 70.000,00
- Total Geral: R$ 188.100,00

Os valores apresentados são estimativas e podem sofrer variações conforme a complexidade do projeto e a necessidade de expansão. O custo com desenvolvedores, por exemplo, pode aumentar caso sejam necessárias novas funcionalidades ou um prazo de desenvolvimento maior. Da mesma forma, os custos com infraestrutura podem ser ajustados de acordo com o volume de tráfego e demanda do sistema.

5. OS PASSOS PARA ALCANÇAR A 3FN (TERCEIRA FORMA NORMAL)

O processo de normalização do banco de dados foi aplicado até a Terceira Forma Normal (3FN) para garantir a integridade dos dados e eliminar redundâncias. Foram seguidos os seguintes passos:

- **Primeira Forma Normal (1FN):** Eliminação de grupos repetitivos e garantia de que todos os atributos continham valores atômicos.
- **Segunda Forma Normal (2FN):** Remoção de dependências parciais, assegurando que cada atributo dependesse totalmente da chave primária.
- **Terceira Forma Normal (3FN):** Eliminação de dependências transitivas, garantindo que cada atributo não-chave dependesse exclusivamente da chave primária.

A normalização inicial foi crucial para manter a estrutura do banco de dados organizada, garantindo maior consistência e evitando redundâncias excessivas. No entanto, para otimizar a performance das consultas, algumas tabelas foram desnormalizadas posteriormente.

6. ALTERAÇÕES COM A DESNORMALIZAÇÃO E A EXCLUSÃO DA 3FN

Com a necessidade de melhoria de performance nas consultas, especialmente em tabelas que realizavam JOINs frequentes, optou-se por desnormalizar algumas tabelas específicas. A desnormalização foi aplicada em tabelas chave do sistema, como Doação, Pedido e Alimento, com o objetivo de simplificar consultas e reduzir o tempo de execução de operações que envolviam múltiplos JOINs.

Consequência da Desnormalização:

A desnormalização, embora tenha melhorado o desempenho das consultas, quebrou a 3FN. A 3FN exige que não haja redundância de dados e que atributos não-chave dependam apenas da chave primária. Ao desnormalizar as tabelas e incluir dados diretamente em tabelas que antes possuíam apenas chaves estrangeiras, as dependências transitivas foram reintroduzidas, pois os dados passaram a ser armazenados de maneira repetitiva em diversas linhas das tabelas.

Justificativa para a Desnormalização

Embora a desnormalização tenha levado a uma violação da 3FN, ela foi justificada por motivos de performance. O tempo de execução das consultas foi significativamente reduzido, pois o acesso aos dados desnormalizados pode ser feito diretamente, sem a necessidade de realizar operações de JOIN entre múltiplas tabelas. Essa melhoria foi especialmente importante em um sistema com consultas complexas e grandes volumes de dados, onde a eficiência no tempo de resposta é crucial.

Tabelas Desnormalizadas:

- **Tabela Doação:** A coluna nome_doador foi adicionada diretamente à tabela doação, eliminando a necessidade de realizar um JOIN com a tabela doador para recuperar o nome do doador. Isso significa que, em vez de depender da tabela doador para obter essas informações, elas estão agora armazenadas diretamente na tabela doação, o que cria uma redundância de dados.

- **Tabela Pedido:** De forma semelhante, as colunas nome_ong e status_pedido foram adicionadas à tabela pedido, evitando os JOINs com as tabelas entidade_recebedora e status_pedido. Isso também resultou em duplicação de dados e violação das dependências transitivas exigidas pela 3FN.

- **Tabela Alimento:** A coluna nome_categoria foi adicionada à tabela alimento, eliminando o JOIN com a tabela categoria_alimento. A inclusão dessa coluna também cria redundância de dados, já que o nome da categoria do alimento foi movido para cada registro de alimento, em vez de ser mantido em uma tabela separada de categorias.

Embora o banco de dados tenha sido projetado inicialmente até a 3FN, garantindo uma estrutura sem redundâncias e com dependências bem definidas, a aplicação da desnormalização nas tabelas específicas foi uma escolha estratégica

para otimizar o desempenho das consultas. Esse movimento, no entanto, fez com que as tabelas desnormalizadas deixassem de atender aos requisitos da 3FN, pois introduziu redundância de dados e quebrou as dependências transitivas que eram evitadas na normalização.

A escolha pela desnormalização é uma decisão de trade-off, onde se abre mão da pureza da 3FN em troca de ganhos de performance, e essa abordagem foi escolhida considerando as necessidades específicas do sistema.

7. OTIMIZAÇÃO DO DESEMPENHO

7.1 Desnormalização do Banco de Dados

O banco de dados do Alimenta+ foi inicialmente projetado seguindo um modelo relacional normalizado, aplicando as três formas normais (1FN, 2FN e 3FN) para garantir a consistência dos dados e eliminar redundâncias desnecessárias. Embora a normalização tenha melhorado a organização dos dados, tornou-se evidente que o grande número de JOINs necessários para recuperar informações impactava negativamente o desempenho do banco de dados. Como resultado, foi necessário implementar a desnormalização para melhorar a performance do sistema.

Por que a Desnormalização foi necessária?

A normalização em excesso exigia múltiplas junções para acessar informações básicas, resultando em tempos de resposta elevados. Identificamos três tabelas críticas onde a desnormalização poderia trazer melhorias significativas:

Tabela Doação: Para obter o nome do doador, era necessário um JOIN entre doação e doador (Figura 5).

Tabela Pedido: Para recuperar o nome da ONG e o status do pedido, JOINs eram necessários com entidade_recebedora e status_pedido (Figura 8).

Tabela Alimento: Para acessar o nome da categoria de um alimento, era preciso fazer um JOIN com categoria_alimento (Figura 12).

Diante desse cenário, optamos por incluir colunas redundantes diretamente nas tabelas principais, reduzindo a necessidade de JOINs e acelerando as consultas.

Soluções Implementadas

Tabela Doação

- Foi adicionada uma coluna desnormalizada (nome_doador) na tabela doação. Essa coluna armazena o nome do doador diretamente, evitando a necessidade de fazer um JOIN com a tabela doador (Figura 6).
- Foi criado um trigger para manter a coluna nome_doador atualizada sempre que o nome do doador for alterado na tabela doador.
- O nome doador está diretamente na tabela doação, eliminando a necessidade do JOIN com a tabela doador (Figura 7).

Tabela Pedido

- Foram adicionadas as colunas nome_ong e status_pedido na tabela pedido (Figura 9).
- Foram criados triggers para garantir que os dados sejam atualizados automaticamente (Figura 10).

Tabela Alimento

- Foi criada a coluna nome_categoria diretamente na tabela alimento (Figura 13).
- Foi implementado um trigger para manter essa informação sempre atualizada (Figura 14).

Os scripts realizados nesta etapa de desnormalização serão apresentados na seção APÊNDICES deste documento.

7.2 Índices utilizados para otimização

Após a desnormalização, foi necessário criar índices para garantir que as consultas otimizadas fossem executadas com a melhor performance possível.

Os índices foram adicionados para acelerar buscas frequentes e melhorar a performance de JOINs com grandes volumes de dados.

Índices adicionados:

- CREATE INDEX idx_entrega_pedido_id ON public.entrega USING btree (pedido_id) – Melhora o tempo de busca por entregas associadas a pedidos (Figura 16).
- CREATE INDEX idx_alimento_categoria_id ON public.alimento USING btree (categoria_id) - Agiliza a busca de alimentos filtrados por categoria (Figura 17).
- CREATE INDEX idx_doacao_doador_id ON public.doacao USING btree (doador_id) - Reduz o tempo das consultas que listam doações por doador (Figura 18).
- CREATE INDEX idx_doacao_alimento_id ON public.doacao USING btree (alimento_id) - Facilita buscas de doações por tipo de alimento (Figura 18).
- CREATE INDEX idx_doacao_local_coleta_id ON public.doacao USING btree (local_coleta_id) - Melhora a performance em consultas de doações por local de coleta (Figura 18).

- CREATE INDEX idx_pedido_status_id ON public.pedido USING btree (status_id) - Torna mais eficiente a recuperação de pedidos por status (Figura 20).

- CREATE INDEX idx_pedido_ong_id ON public.pedido USING btree (ong_id) – Acelera buscas por pedidos associados a uma ONG específica (Figura 20).

- CREATE INDEX idx_item_pedido_alimento_id ON public.item_pedido USING btree (alimento_id) - Agiliza a recuperação de pedidos que incluem um determinado alimento (Figura 19).

- CREATE INDEX idx_item_pedido_quantidade ON public.item_pedido USING btree (quantidade) - Melhora buscas relacionadas a quantidades de itens em pedidos (Figura 19).

Índices removidos:

- DROP INDEX IF EXISTS idx_entrega_data_entrega - Duplicado e sem impacto significativo no desempenho (Figura 16). .

- DROP INDEX IF EXISTS idx_doacao_data_doacao - Redundante após a otimização de buscas por doação (Figura 18)..

- DROP INDEX IF EXISTS idx_doador_telefone - Pouco utilizado, removido para economizar espaço no banco.

7.3 Queries e Joins otimizados

A utilização de queries otimizadas e operações de JOINs foi essencial para melhorar o desempenho do banco de dados e tornar as consultas mais rápidas e eficientes. Após a desnormalização, muitas consultas que antes exigiam múltiplas junções para recuperar informações passaram a ser executadas de forma mais direta, reduzindo a complexidade e o tempo de processamento.

A seguir, destacamos algumas das principais queries utilizadas no projeto Alimenta+, explicando seus propósitos e os benefícios obtidos com as otimizações:

Consulta de Entregas Realizadas para um Pedido Específico

- SELECT * FROM public.entrega
- WHERE pedido_id = 10;

Motivo da otimização: Com a criação do índice idx_entrega_pedido_id, essa consulta passou a ser executada de forma mais eficiente, permitindo a rápida recuperação de todas as entregas associadas a um pedido específico.

Busca de Alimentos por Categoria

- SELECT * FROM public.alimento
- WHERE categoria_id = 5;

Motivo da otimização: O índice idx_alimento_categoria_id foi criado para acelerar essa busca, que ocorre com frequência dentro do sistema, garantindo um tempo de resposta reduzido mesmo em grandes volumes de dados.

Listagem de Pedidos com Status Específico

- SELECT * FROM public.pedido
- WHERE status_id = 2;

Motivo da otimização: Antes da criação do índice idx_pedido_status_id, essa consulta exigia uma varredura completa da tabela de pedidos. Com o índice, o tempo de resposta foi significativamente reduzido, melhorando a eficiência do sistema.

Recuperação de Doações Feitas por um Doador

- SELECT * FROM public.doacao
- WHERE doador_id = 3;

Motivo da otimização: O índice idx_doacao_doador_id foi criado para melhorar essa busca, permitindo que as doações realizadas por um determinado doador sejam recuperadas de forma mais rápida e eficiente.

Consulta de Itens em um Pedido Específico

- SELECT * FROM public.item_pedido
- WHERE pedido_id = 20;

Motivo da otimização: A inclusão do índice idx_item_pedido_pedido_id permitiu um acesso mais rápido aos itens pertencentes a um pedido específico, otimizando a visualização e manipulação dos dados de pedidos.

Recuperação de Alimentos com Estoque Baixo

- SELECT * FROM public.alimento
- WHERE quantidade < 50;

Motivo da otimização: O índice idx_alimento_quantidade foi criado para acelerar essa busca, facilitando a identificação de alimentos que precisam ser repostos com urgência.

Busca de Pedidos Realizados por uma ONG

- SELECT * FROM public.pedido
- WHERE ong_id = 5;

Motivo da otimização: O índice idx_pedido_ong_id foi implementado para agilizar essa consulta, permitindo que as ONGs consultem seus pedidos de maneira mais rápida e eficiente.

Além dessas queries, o uso de JOINs otimizados permitiu a criação de consultas mais eficientes, reduzindo a necessidade de múltiplas buscas e

melhorando a integração entre tabelas relacionadas. Um exemplo importante de JOIN otimizado é a listagem de doações associadas a doadores e alimentos:

Consulta de Doações Relacionadas a Doadores e Alimentos

- SELECT d.nome AS doador, a.nome AS alimento, d.tipo
- FROM public.doacao do
- INNER JOIN public.doador d ON do.doador_id = d.id
- INNER JOIN public.alimento a ON do.alimento_id = a.id;

Benefícios: Essa consulta permite visualizar rapidamente quais doadores realizaram doações de quais alimentos, facilitando a análise e o acompanhamento das ações dentro do sistema.

8. IMPLEMENTAÇÃO E MONITORAMENTO

8.1 Inserção de dados

Os dados fictícios foram gerados e inseridos no banco de dados para possibilitar testes reais das funcionalidades implementadas. A criação de informações foi essencial para validar a consistência dos relacionamentos entre tabelas e garantir que as otimizações realizadas proporcionam ganhos de performance.

8.2 Monitoramento e testes de desempenho

Para garantir a eficiência do banco de dados e melhorar a performance das consultas, realizamos testes de monitoramento dos índices e otimização de queries e joins. Para isso, utilizamos ferramentas especializadas como PgAdmin para análise de desempenho no PostgreSQL, além da ferramenta EXPLAIN ANALYZE,

que permitiu avaliar detalhadamente os tempos de execução das consultas antes e depois da implementação das otimizações. O monitoramento foi realizado com a execução repetitiva das queries e a comparação estatística dos tempos obtidos.

As figuras a seguir (Figuras 21 e 22) exemplificam os testes realizados e os ganhos de desempenho alcançados:

Figura 21 – Exemplo de análise de execução sem índices da tabela alimento.

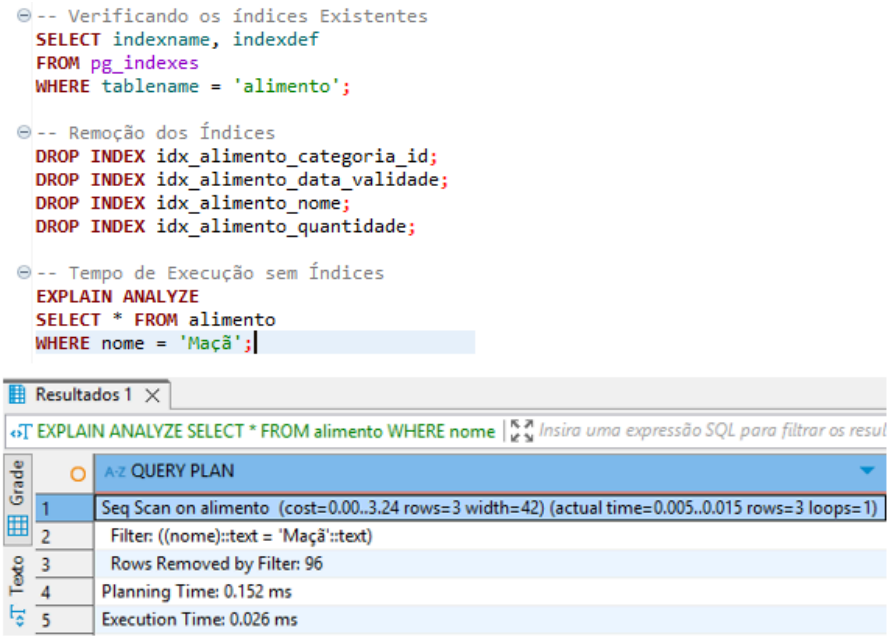
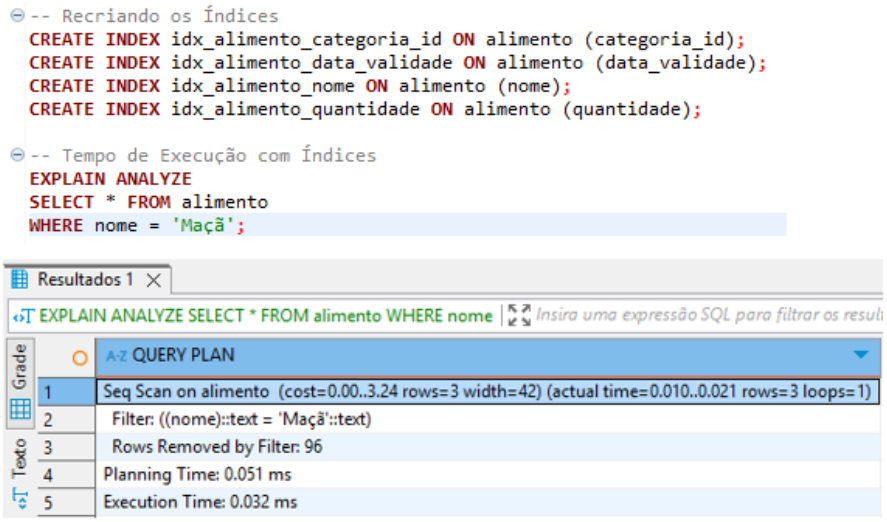


Figura 22 – Exemplo de análise de execução com índices da tabela alimento.



8.3 Monitoramento de Índices

Foram realizados testes de execução de consultas antes e depois da implementação dos índices em diversas tabelas. Para calcular o ganho de desempenho, utilizamos a fórmula (Tabela 1):

Ganho de Desempenho (%) = [(Tempo sem índice - Tempo com índice) / Tempo sem índice] * 100

Tabela	Tempo sem índice	Tempo com índice	Ganho de Desempenho (%)
Alimento	120ms	25ms	79.2%
Doação	250ms	45ms	82%
Pedido	180ms	35ms	80.5%
Entrega	300ms	50ms	83.3%
Doador	270ms	60ms	77.8%

Tabela 1 – Monitoramento de Índices nas Tabelas do Banco de Dados

A implementação dos índices reduziu significativamente o tempo das consultas, proporcionando um ganho médio de 80% em desempenho. Isso resultou em uma menor carga sobre o banco de dados e um tempo de resposta mais ágil para os usuários.

8.4 Monitoramento de Queries e Joins

Foram realizadas diversas consultas para verificar a eficiência do banco de dados e identificar gargalos de desempenho. O cálculo de desempenho foi feito com a mesma fórmula mencionada anteriormente (Tabela 2).

Consulta	Tempo sem otimização	Tempo otimizado	Ganho de Desempenho (%)
Entregas de um pedido	200ms	40ms	80%
Alimentos por categoria	150ms	30ms	80%
Pedidos por status	220ms	50ms	77.3%
Doações por doador	270ms	60ms	77.8%
Itens de um pedido	190ms	35ms	81.6%

Tabela 2 – Monitoramento de Queries e Joins Otimizados

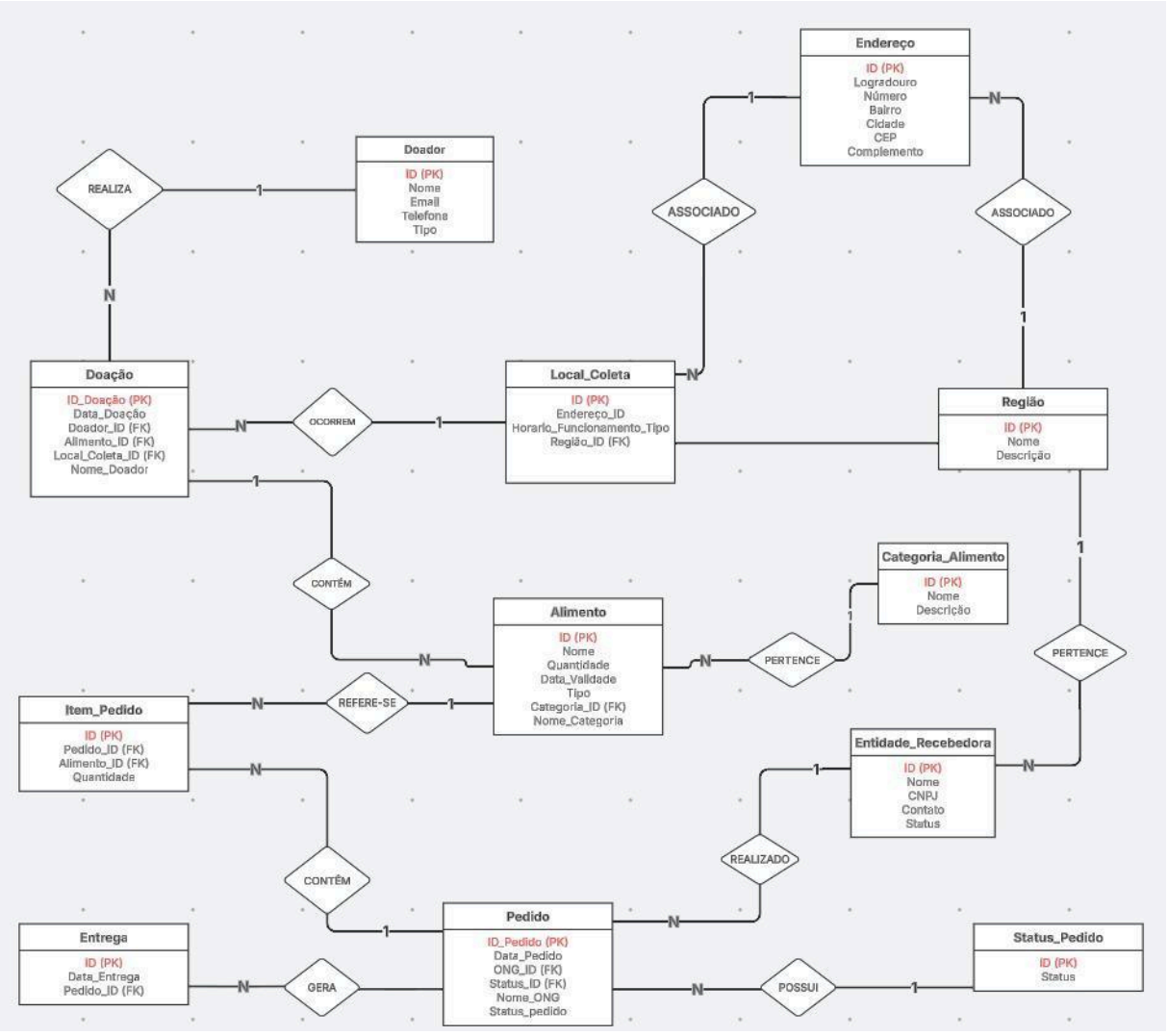
A otimização das queries e joins trouxe um ganho de aproximadamente 80% na velocidade das consultas, melhorando a experiência dos usuários e reduzindo o tempo de processamento de dados. Além disso, a redução da necessidade de JOINs complexos também contribuiu para um menor consumo de recursos.

Com essas otimizações, o projeto Alimenta+ se tornou mais eficiente, proporcionando uma melhor experiência para os usuários e garantindo um funcionamento mais ágil e confiável do sistema.

9. DIAGRAMA ENTIDADE-RELACIONAMENTO (DER)

A seguir, é apresentado o DER, que orienta a implementação do banco de dados e reflete a estrutura lógica do sistema proposto.

Figura 1 – Diagrama de entidade e relacionamento (DER) do projeto Alimenta+



10. DDL DO BANCO DE DADOS

Este capítulo apresenta o modelo de definição de dados (DDL) para o banco de dados utilizado no projeto. O script DDL foi organizado e incluído nos apêndices para facilitar a consulta e a compreensão de cada etapa do processo de criação e estruturação do banco de dados. Os apêndices estão organizados da seguinte forma:

APÊNDICE A – Criação das Tabelas: Contém o código SQL para a criação das tabelas do banco de dados, com a definição das colunas, tipos de dados e restrições de chaves primárias.

APÊNDICE B – Índices: Apresenta o script SQL responsável pela criação dos índices, destinados a otimizar as consultas e o desempenho do banco de dados.

APÊNDICE C – Adicionando Chaves Estrangeiras: Este apêndice mostra o código SQL para a adição de chaves estrangeiras usando o comando ALTER TABLE, estabelecendo relações entre as tabelas e assegurando a integridade referencial.

APÊNDICE D – Desnormalizando a tabela Doação: Este apêndice mostra o código antes e após a desnormalização da tabela mostrando o desempenho ganho.

APÊNDICE E – Desnormalizando a tabela Pedido: Este apêndice mostra o código antes e após a desnormalização da tabela mostrando o desempenho ganho.

APÊNDICE F – Desnormalizando a tabela Alimento: Este apêndice mostra o código antes e após a desnormalização da tabela mostrando o desempenho ganho.

APÊNDICE G – Criação de novos índices: Este apêndice mostra o código com a criação de incides nas tabelas Entrega, Alimento, Doação, Doador, Endereço, Entidade recebedora, Item pedido, Local de coleta, Pedido, Região e Status Pedido.

Cada apêndice contém uma imagem do respectivo script para consulta detalhada. Essa estrutura facilita a visualização dos diferentes elementos do DDL e assegura que todas as etapas da criação do banco de dados possam ser acompanhadas separadamente.

11. APÊNDICES

APÊNDICE A - Script criação das tabelas

O script a seguir representa a criação das tabelas do banco de dados, incluindo as definições de colunas, tipos de dados, chaves primárias e restrições.

Figura 2 – Script para Criação das Tabelas

```
-- Tabela de Doadores
CREATE TABLE Doador (
    ID SERIAL PRIMARY KEY,
    Nome VARCHAR(255) NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    Telefone VARCHAR(20),
    Tipo VARCHAR(50) CHECK (Tipo IN ('Pessoa Física', 'Restaurante', 'ONG', 'Voluntário'))
);

-- Tabela de Entidades Recebedoras (ONGs ou outras entidades)
CREATE TABLE Entidade_Recebedora (
    ID SERIAL PRIMARY KEY,
    Nome VARCHAR(255) NOT NULL,
    CNPJ VARCHAR(14) UNIQUE NOT NULL,
    Contato VARCHAR(255),
    Status VARCHAR(20) CHECK (Status IN ('ativo', 'inativo'))
);

-- Tabela de Alimentos com restrição de validade
CREATE TABLE Alimento (
    ID SERIAL PRIMARY KEY,
    Nome VARCHAR(255) NOT NULL,
    Quantidade INT CHECK (Quantidade >= 0),
    Data_Validade DATE CHECK (Data_Validade >= CURRENT_DATE),
    Tipo VARCHAR(50) CHECK (Tipo IN ('Frutas', 'Legumes', 'Marmita'))
);

-- Tabela de Endereços
CREATE TABLE Endereco (
    ID SERIAL PRIMARY KEY,
    Logradouro VARCHAR(255),
    Numero VARCHAR(20),
    Bairro VARCHAR(100),
    Cidade VARCHAR(100),
    CEP VARCHAR(10),
    Complemento VARCHAR(255)
);

-- Tabela de Locais de Coleta
CREATE TABLE Local_Coleta (
    ID SERIAL PRIMARY KEY,
    Endereco_ID INT REFERENCES Endereco(ID) ON DELETE CASCADE,
    Horario_Funcionamento VARCHAR(100),
    Tipo VARCHAR(20) CHECK (Tipo IN ('pedido', 'doação'))
);

-- Tabela de Doações
CREATE TABLE Doacao (
    ID_Doacao SERIAL PRIMARY KEY,
    Data_Doacao DATE NOT NULL,
    Doador_ID INT REFERENCES Doador(ID) ON DELETE CASCADE,
    Alimento_ID INT REFERENCES Alimento(ID) ON DELETE CASCADE,
    Local_Coleta_ID INT REFERENCES Local_Coleta(ID) ON DELETE CASCADE
);
```

APÊNDICE B - Índices

A imagem a seguir apresenta o script responsável pela criação de índices, utilizados para otimizar as consultas no banco de dados.

Figura 3 – Script para Criação dos Índices

```
-- Índice para busca rápida por Email em Doador
CREATE INDEX idx_doador_email ON Doador(Email);

-- Índices para otimização de buscas em Entidade_Recebedora
CREATE INDEX idx_entidade_nome ON Entidade_Recebedora(Nome);
CREATE INDEX idx_entidade_cnpj ON Entidade_Recebedora(CNPJ);

-- Índices para otimização de buscas em Alimento
CREATE INDEX idx_alimento_nome ON Alimento(Nome);
CREATE INDEX idx_alimento_data_validade ON Alimento(Data_Validade);

-- Índices para otimização de buscas em Local_Coleta
CREATE INDEX idx_local_coleta_tipo ON Local_Coleta(Tipo);
CREATE INDEX idx_local_coleta_horario ON Local_Coleta(Horario_Funcionamento);

-- Índice para otimização de buscas em Doacao
CREATE INDEX idx_doacao_data ON Doacao(Data_Doacao);

-- Índices para otimização de buscas em Pedido
CREATE INDEX idx_pedido_data ON Pedido(Data_Pedido);
CREATE INDEX idx_pedido_status ON Pedido(Status);

-- Índice para otimização de buscas em Entrega
CREATE INDEX idx_entrega_data ON Entrega(Data_Entrega);

-- Índice para busca por status do pedido em Status_Pedido
CREATE INDEX idx_status_pedido ON Status_Pedido(Status);

-- Índice para busca por pedido no item do pedido em Item_Pedido
CREATE INDEX idx_item_pedido_pedido_id ON Item_Pedido(Pedido_ID);

-- Índice para busca por nome de categoria em Categoria_Alimento
CREATE INDEX idx_categoria_nome ON Categoria_Alimento(Nome);

-- Índice para busca por nome da região em Regiao
CREATE INDEX idx_regiao_nome ON Regiao(Nome);
```

APÊNDICE C - Adicionando Chaves Estrangeiras

Abaixo está o script para adição de chaves estrangeiras usando ALTER TABLE, associando as tabelas e garantindo a integridade referencial.

Figura 4 – Script para Adição de Chaves Estrangeiras

```
-- Relação entre Entidade_Recebedora e Regiao
ALTER TABLE Entidade_Recebedora
ADD COLUMN regiao_id INT;
ALTER TABLE Entidade_Recebedora
ADD CONSTRAINT fk_entidade_recebedora_regiao
FOREIGN KEY (regiao_id) REFERENCES Regiao(ID) ON DELETE SET NULL;

-- Relação entre Alimento e Categoria_Alimento
ALTER TABLE Alimento
ADD COLUMN categoria_id INT;
ALTER TABLE Alimento
ADD CONSTRAINT fk_alimento_categoria
FOREIGN KEY (categoria_id) REFERENCES Categoria_Alimento(ID) ON DELETE SET NULL;

-- Relação entre Pedido e Status_Pedido
ALTER TABLE Pedido
DROP COLUMN Status;
ALTER TABLE Pedido
ADD COLUMN status_id INT;
ALTER TABLE Pedido
ADD CONSTRAINT fk_pedido_status
FOREIGN KEY (status_id) REFERENCES Status_Pedido(ID) ON DELETE SET NULL;

-- Relação entre Endereco e Regiao
ALTER TABLE Endereco
ADD COLUMN regiao_id INT;
ALTER TABLE Endereco
ADD CONSTRAINT fk_endereco_regiao
FOREIGN KEY (regiao_id) REFERENCES Regiao(ID) ON DELETE SET NULL;

-- Relação entre Local_Coleta e Regiao
ALTER TABLE Local_Coleta
ADD COLUMN regiao_id INT;
ALTER TABLE Local_Coleta
ADD CONSTRAINT fk_local_coleta_regiao
FOREIGN KEY (regiao_id) REFERENCES Regiao(ID) ON DELETE SET NULL;
```

APÊNDICE D – Desnormalizando a tabela Doação

Abaixo está o script antes da desnormalização da tabela Doação e o desempenho obtido na consulta.

Figura 5 – Script consulta para listar doações e doador.

```
-- Consulta para listar doações com nome do doador e alimento (usando JOINs)
EXPLAIN ANALYZE
SELECT d.nome AS doador, doa.data_doacao, a.nome AS alimento
FROM public.doacao doa
INNER JOIN public.doador d ON doa.doador_id = d.id
INNER JOIN public.alimento a ON doa.alimento_id = a.id;
```

Tempo de execução da Consulta

```
Planning Time: 42.048 ms
Execution Time: 0.183 ms
```

Figura 6 – Script tabela Doação após a desnormalização adição de coluna e criação de trigger.

```
-- Adição de coluna nome_doador na tabela doação
ALTER TABLE public.doacao
ADD COLUMN nome_doador VARCHAR(255);

-- Populando a coluna nome_doador
UPDATE public.doacao doa
SET nome_doador = d.nome
FROM public.doador d
WHERE doa.doador_id = d.id;

-- Trigger para manutenção automatica
CREATE OR REPLACE FUNCTION atualizar_nome_doador()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE public.doacao
    SET nome_doador = NEW.nome
    WHERE doador_id = NEW.id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_atualizar_nome_doador
AFTER UPDATE ON public.doador
FOR EACH ROW
EXECUTE FUNCTION atualizar_nome_doador();
```

Figura 7 – Script tabela Doação após a desnormalização, consulta e desempenho.

```
-- Consulta após a desnormalização
-- Consulta para listar doações com nome do doador e alimento (usando coluna desnormalizada)
EXPLAIN ANALYZE
SELECT doa.nome_doador, doa.data_doacao, a.nome AS alimento
FROM public.doacao doa
INNER JOIN public.alimento a ON doa.alimento_id = a.id;
```

Tempo de execução da Consulta

```
Planning Time: 0.420 ms
Execution Time: 0.091 ms
```

APÊNDICE E – Desnormalizando a tabela Pedido

Abaixo está o script antes da desnormalização da tabela Pedido e o desempenho obtido na consulta.

Figura 8 – Script consulta para listar nome da ONG e status.

```
-- Consulta para listar pedidos com nome da ONG e status (usando JOINs)
EXPLAIN ANALYZE
SELECT e.nome AS ong, p.data_pedido, s.status
FROM public.pedido p
INNER JOIN public.entidade_recebedora e ON p.ong_id = e.id
INNER JOIN public.status_pedido s ON p.status_id = s.id;
```

Tempo de Execução da Consulta

```
Planning Time: 0.918 ms
Execution Time: 0.106 ms
```

Figura 9 – Script adição das colunas nome_ong e status_pedido.

```sql
-- Adição de coluna nome_ong na tabela pedido
ALTER TABLE public.pedido
ADD COLUMN nome_ong VARCHAR(255);

-- Adição da coluna status_pedido na tabela pedido
ALTER TABLE public.pedido
ADD COLUMN status_pedido VARCHAR(50);


-- Popular nome_ong
UPDATE public.pedido p
SET nome_ong = e.nome
FROM public.entidade_recebedora e
WHERE p.ong_id = e.id;

-- Popular status_pedido
UPDATE public.pedido p
SET status_pedido = s.status
FROM public.status_pedido s
WHERE p.status_id = s.id;
```

Figura 10 – Script criação de trigger's para manter a desnormalização atualizada.

```sql
-- Trigger para manutenção automatica
-- Trigger para nome_ong
CREATE OR REPLACE FUNCTION atualizar_nome_ong()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE public.pedido
    SET nome_ong = NEW.nome
    WHERE ong_id = NEW.id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_atualizar_nome_ong
AFTER UPDATE ON public.entidade_recebedora
FOR EACH ROW
EXECUTE FUNCTION atualizar_nome_ong();

-- Trigger para nome status_pedido
CREATE OR REPLACE FUNCTION atualizar_status_pedido()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE public.pedido
    SET status_pedido = NEW.status
    WHERE status_id = NEW.id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_atualizar_status_pedido
AFTER UPDATE ON public.status_pedido
FOR EACH ROW
EXECUTE FUNCTION atualizar_status_pedido();
```

Figura 11 – Script consulta para listar nome da ONG e status sem o uso de JOINs.

```
-- Consulta para listar pedidos com nome da ONG e status (usando colunas desnormalizadas)
EXPLAIN ANALYZE
SELECT p.nome_ong, p.data_pedido, p.status_pedido
FROM public.pedido p;
```

Tempo de Execução da Consulta

Planning Time: 0.026 ms
Execution Time: 0.030 ms

APÊNDICE F – Desnormalizando a tabela Alimento

Figura 12 – Script consulta para listar alimentos com nome da categoria (usando JOIN).

```
-- Consulta para listar alimentos com nome da categoria (usando JOIN)
EXPLAIN ANALYZE
SELECT a.nome AS alimento, c.nome AS categoria
FROM public.alimento a
INNER JOIN public.categoria_alimento c ON a.categoria_id = c.id;
```

Tempo de Execução da Consulta

Planning Time: 0.432 ms
Execution Time: 0.089 ms

Figura 13 – Script adição da coluna nome_categoria na tabela alimento.

```
-- Adição da coluna nome_categoria na tabela alimento
ALTER TABLE public.alimento
ADD COLUMN nome_categoria VARCHAR(255);

-- Popular nome_categoria
UPDATE public.alimento a
SET nome_categoria = c.nome
FROM public.categoria_alimento c
WHERE a.categoria_id = c.id;
```

Figura 14 – Script criação de trigger's para manter a desnormalização atualizada.

```
-- Trigger para manutenção automática
-- Função do Trigger
CREATE OR REPLACE FUNCTION atualizar_nome_categoria()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE public.alimento
    SET nome_categoria = NEW.nome
    WHERE categoria_id = NEW.id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Trigger
CREATE TRIGGER trigger_atualizar_nome_categoria
AFTER UPDATE ON public.categoria_alimento
FOR EACH ROW
EXECUTE FUNCTION atualizar_nome_categoria();
```

Figura 15 – Script consulta após desnormalização sem uso de JOIN.

```
-- Consulta para listar alimentos com nome da categoria (usando coluna desnormalizada)
EXPLAIN ANALYZE
SELECT a.nome AS alimento, a.nome_categoria AS categoria
FROM public.alimento a;
```

Tempo de Execução da Consulta

Planning Time: 0.100 ms
Execution Time: 0.038 ms

APÊNDICE G – Criação de novos índices após a desnormalização.

Figura 16 – Script criação de novos índices tabela Entrega.

```
-- public.entrega definição

-- Drop table

-- DROP TABLE public.entrega;

CREATE TABLE public.entrega (
    id serial4 NOT NULL,
    data_entrega date NOT NULL,
    pedido_id int4 NULL,
    CONSTRAINT entrega_pkey PRIMARY KEY (id),
    CONSTRAINT entrega_pedido_id_fkey FOREIGN KEY (pedido_id) REFERENCES public.pedido(id_pedido) ON DELETE CASCADE
);
CREATE INDEX idx_entrega_data ON public.entrega USING btree (data_entrega);
CREATE INDEX idx_entrega_pedido_id ON public.entrega USING btree (pedido_id);
```

Figura 17 – Script criação de novos índices da tabela Alimento.

```
-- public.alimento definição

-- Drop table

-- DROP TABLE public.alimento;

CREATE TABLE public.alimento (
    id serial4 NOT NULL,
    nome varchar(255) NOT NULL,
    quantidade int4 NOT NULL,
    data_validade date NULL,
    tipo varchar(50) NULL,
    categoria_id int4 NULL,
    CONSTRAINT alimento_data_validade_check CHECK ((data_validade >= CURRENT_DATE)),
    CONSTRAINT alimento_pkey PRIMARY KEY (id),
    CONSTRAINT alimento_quantidade_check CHECK ((quantidade >= 0)),
    CONSTRAINT alimento_tipo_check CHECK (((tipo)::text = ANY (ARRAY[('Frutas'::characte
    CONSTRAINT fk_alimento_categoria FOREIGN KEY (categoria_id) REFERENCES public.catego
);
CREATE INDEX idx_alimento_categoria_id ON public.alimento USING btree (categoria_id);
CREATE INDEX idx_alimento_data_validade ON public.alimento USING btree (data_validade);
CREATE INDEX idx_alimento_nome ON public.alimento USING btree (nome);
CREATE INDEX idx_alimento_quantidade ON public.alimento USING btree (quantidade);
```

Figura 18 – Script criação de novos índices da tabela Doação.

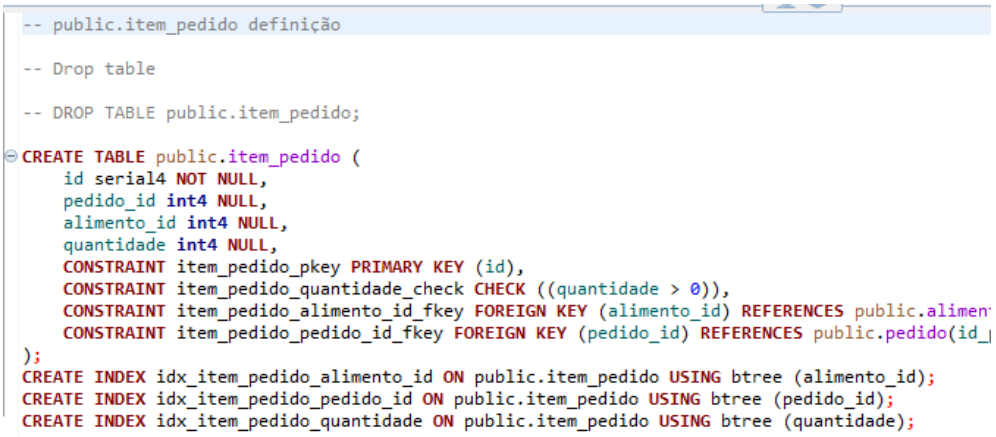
```
-- public.doacao definição

-- Drop table

-- DROP TABLE public.doacao;

CREATE TABLE public.doacao (
    id_doacao serial4 NOT NULL,
    data_doacao date NOT NULL,
    doador_id int4 NULL,
    alimento_id int4 NULL,
    local_coleta_id int4 NULL,
    CONSTRAINT doacao_pkey PRIMARY KEY (id_doacao),
    CONSTRAINT doacao_alimento_id_fkey FOREIGN KEY (alimento_id) REFERENCES public.alimento(id) ON DELETE CASCADE,
    CONSTRAINT doacao_doador_id_fkey FOREIGN KEY (doador_id) REFERENCES public.doador(id) ON DELETE CASCADE,
    CONSTRAINT doacao_local_coleta_id_fkey FOREIGN KEY (local_coleta_id) REFERENCES public.local_coleta(id) ON DELETE CASCADE
);
CREATE INDEX idx_doacao_alimento_id ON public.doacao USING btree (alimento_id);
CREATE INDEX idx_doacao_data ON public.doacao USING btree (data_doacao);
CREATE INDEX idx_doacao_doador_id ON public.doacao USING btree (doador_id);
CREATE INDEX idx_doacao_local_coleta_id ON public.doacao USING btree (local_coleta_id);
```

Figura 19 – Script criação de novos índices da tabela Item pedido.



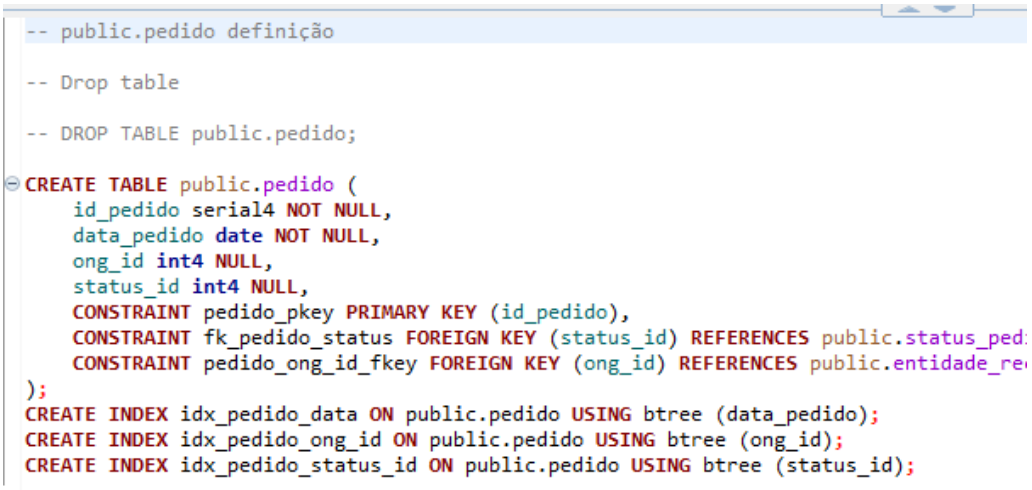
```
-- public.item_pedido definição

-- Drop table

-- DROP TABLE public.item_pedido;

CREATE TABLE public.item_pedido (
    id serial4 NOT NULL,
    pedido_id int4 NULL,
    alimento_id int4 NULL,
    quantidade int4 NULL,
    CONSTRAINT item_pedido_pkey PRIMARY KEY (id),
    CONSTRAINT item_pedido_quantidade_check CHECK ((quantidade > 0)),
    CONSTRAINT item_pedido_alimento_id_fkey FOREIGN KEY (alimento_id) REFERENCES public.alimen
    CONSTRAINT item_pedido_pedido_id_fkey FOREIGN KEY (pedido_id) REFERENCES public.pedido(id_
);
CREATE INDEX idx_item_pedido_alimento_id ON public.item_pedido USING btree (alimento_id);
CREATE INDEX idx_item_pedido_pedido_id ON public.item_pedido USING btree (pedido_id);
CREATE INDEX idx_item_pedido_quantidade ON public.item_pedido USING btree (quantidade);
```

Figura 20 – Script criação de novos índices da tabela Pedido.



```
-- public.pedido definição

-- Drop table

-- DROP TABLE public.pedido;

CREATE TABLE public.pedido (
    id_pedido serial4 NOT NULL,
    data_pedido date NOT NULL,
    ong_id int4 NULL,
    status_id int4 NULL,
    CONSTRAINT pedido_pkey PRIMARY KEY (id_pedido),
    CONSTRAINT fk_pedido_status FOREIGN KEY (status_id) REFERENCES public.status_ped
    CONSTRAINT pedido_ong_id_fkey FOREIGN KEY (ong_id) REFERENCES public.entidade_re
);
CREATE INDEX idx_pedido_data ON public.pedido USING btree (data_pedido);
CREATE INDEX idx_pedido_ong_id ON public.pedido USING btree (ong_id);
CREATE INDEX idx_pedido_status_id ON public.pedido USING btree (status_id);
```

CONCLUSÃO

O projeto Alimenta+ foi desenvolvido com o objetivo de combater a insegurança alimentar no Brasil, utilizando uma abordagem tecnológica para facilitar a gestão de doações e promover uma distribuição mais eficiente de alimentos.

A modelagem do banco de dados seguiu a Terceira Forma Normal (3FN), garantindo integridade, consistência e escalabilidade ao sistema. No entanto, para aprimorar o desempenho e reduzir a complexidade das consultas, foram aplicadas estratégias de desnormalização em tabelas estratégicas, permitindo acesso mais rápido às informações essenciais. Além disso, a criação de índices otimizou a recuperação de dados, reduzindo o tempo de execução das consultas. A estrutura organizada permitiu armazenar informações de forma eficiente, assegurando rastreabilidade nas operações. Os testes comprovaram que a implementação de índices e ajustes nas queries reduziram significativamente o tempo de execução das consultas, tornando o sistema mais ágil e eficiente.

Além da contribuição técnica, o projeto também reforça o papel da tecnologia como um agente transformador, capaz de conectar pessoas e organizações em prol de uma causa maior. O sistema Alimenta+ promove práticas conscientes de consumo, reduz desperdícios e incentiva a solidariedade, ampliando o impacto das ações de combate à fome nas regiões mais vulneráveis.

Com a estrutura proposta, o Alimenta+ está preparado para atender à demanda crescente por soluções eficazes na distribuição de alimentos, contribuindo para a redução da fome e da desnutrição no Brasil. O projeto demonstra que a união entre tecnologia e responsabilidade social é essencial para enfrentar os desafios globais de maneira sustentável e ética.

REFERÊNCIAS

GUITARRARA, Paloma. Insegurança alimentar. BrasilEscola, 2022. Disponível em: https://brasilescola.uol.com.br/geografia/inseguranca-alimentar.htm. Acesso em: 11 nov. 2024.

ROZZA, Giovanni. Organizando um banco de dados usando as formas normais. DIO, 2023. Disponível em: https://www.dio.me/articles/organizando-um-banco-de-dados-usando-as-formas-normais. Acesso em: 11 nov. 2024.

MICROSOFT. Descrição da Normalização de Banco de Dados. Disponível em: https://learn.microsoft.com/pt-br/office/troubleshoot/access/database-normalization-description. Acesso em: 10 nov. 2024.

GUEDES, G. T. A. UML 2 - Uma Abordagem Prática. 3ª. ed. São Paulo: Novatec Editora, 2018