Projeto Temático I
Carbon Tracker

Nicolas Chiarani Farina[1], **Lucas Felippi Lora**[2]

[1]Área de Conhecimento de Ciências Exatas e Engenharias

[2]Universidade de Caxias do Sul (UCS) – Caxias do Sul – RS - Brasil

Abstract. *This article discusses the development of Carbon Tracker, an application designed to monitor and reduce users' carbon footprints by tracking their daily activities. The application facilitates the calculation of carbon footprints associated with the use of transportation, household appliances and water consumption, promoting awareness and encouraging more sustainable practices. With detailed reporting features, users can track their progress and identify opportunities to minimize their environmental impact. Developed in C# using the MVP architecture, Carbon Tracker offers an intuitive and personalized experience for users, promoting the adoption of more sustainable habits.*

Resumo. *Este artigo aborda o desenvolvimento do Carbon Tracker, um aplicativo projetado para monitorar e reduzir a pegada de carbono dos usuários através do rastreamento de suas atividades cotidianas. A aplicação facilita o cálculo da pegada de carbono associada ao uso de meios de transporte, eletrodomésticos e consumo de água, promovendo a conscientização e incentivando práticas mais sustentáveis. Com funcionalidades de relatórios detalhados, os usuários podem acompanhar sua evolução e identificar oportunidades de minimizar seu impacto ambiental. Desenvolvido em C#, utilizando a arquitetura MVP, o Carbon Tracker oferece uma experiência intuitiva e personalizada para os usuários, promovendo a adoção de hábitos mais sustentáveis.*

1. Introdução

Nos dias de hoje, a sustentabilidade e a preservação ambiental tornaram-se preocupações centrais para a sociedade. A consciência crescente sobre os impactos ambientais das atividades humanas tem gerado uma demanda por soluções que permitam aos indivíduos entender e gerenciar sua pegada de carbono. Com o avanço da tecnologia, surgem oportunidades para desenvolver ferramentas inovadoras que auxiliem nessa tarefa.

Neste contexto, o Carbon Tracker se destaca como uma solução digital que oferece aos usuários a capacidade de avaliar e controlar seu impacto ambiental diário. Através de um processo simples de inserção de dados sobre transporte, consumo de energia e água, o Carbon Tracker fornece insights valiosos sobre como as escolhas diárias afetam o meio ambiente e oferece recomendações personalizadas para a redução da pegada de carbono.

Seguindo essa perspectiva, este artigo descreve a atividade realizada durante a disciplina Projeto Temático I, na qual foi desenvolvido o Carbon Tracker. A aplicação possibilita aos usuários não apenas monitorar e reduzir seu impacto ambiental, mas também adotar hábitos mais sustentáveis de forma prática e informada. Este é o cerne da aplicação desenvolvida, uma ferramenta que empodera os indivíduos a tomarem decisões conscientes e ambientalmente responsáveis.

Neste artigo, serão abordados os conceitos aplicados no desenvolvimento do Carbon Tracker, explorando como a ferramenta foi idealizada e construída. Será demonstrado como os usuários podem monitorar seu impacto ambiental de maneira detalhada, recebendo recomendações específicas para a melhoria contínua de seus hábitos diários. Além disso, serão exploradas as funcionalidades que permitem uma experiência de uso intuitiva e personalizada, incentivando a adoção de práticas sustentáveis no cotidiano.

2. Levantando os Requisitos

A primeira etapa do projeto foi compreender o problema e propor uma solução. O objetivo geral do projeto era desenvolver um aplicativo desktop seguindo as etapas clássicas do desenvolvimento de software, incluindo análise de negócio, análise de requisitos, análise de sistema, projeto de software e definição da arquitetura de software. O problema apresentado consistia em criar um software na área de mudança climática e meio ambiente, então resolvemos criar um sistema para calcular a pegada de carbono dos usuários, utilizando princípios de programação orientada a objetos, fundamentos de banco de dados (persistência) e engenharia de software.

Durante esta etapa inicial, realizamos o levantamento de requisitos do projeto, uma fase crucial onde desenvolvedores e partes interessadas colaboram para entender as necessidades e expectativas do sistema. Esse processo envolve coletar, analisar, documentar e validar as informações necessárias para o software, incluindo funcionalidades, desempenho, segurança, interfaces do usuário e outros requisitos técnicos. A agilidade na engenharia de requisitos é essencial para transferir ideias dos envolvidos para a equipe de desenvolvimento de forma harmônica e sem atrasos. É comum que novos requisitos surjam à medida que o desenvolvimento iterativo progride [Pressman and Maxim 2021].

Conforme destacado por [Pressman and Maxim 2021], a importância do levantamento de requisitos é evidenciada por aspectos como a compreensão do problema, base para o desenvolvimento, satisfação do cliente, controle de mudanças, redução de riscos e comunicação efetiva.

Os requisitos funcionais identificados para o nosso software de cálculo da pegada de carbono incluem:

RF01 Usuários: Criar, ler, atualizar e deletar informações de usuários.

RF02 Alterar Informações do Próprio Usuário: Permitir que cada usuário possa alterar suas próprias informações cadastradas.

RF03 Grupo de Usuários: Gerenciar grupos de usuários, incluindo criar, listar, atualizar e deletar grupos.

RF04 Entrar em Grupos de Usuários: Funcionalidade para usuários ingressarem em grupos existentes.

RF05 Pré Cadastro de Eletrodomésticos: Realizar operações de criação, leitura, atualização e deleção de registros de eletrodomésticos.

RF06 Pré Cadastro de Transportes: Realizar operações de criação, leitura, atualização e deleção de registros de transportes.

RF07 Registro de Gastos de Eletrodomésticos: Permitir que os usuários registrem gastos relacionados a eletrodomésticos.

RF08 Registro de Gastos de Transportes: Permitir que os usuários registrem gastos relacionados a transportes.

RF09 Comparações dos Gastos por Períodos: Capacidade de comparar os gastos registrados ao longo de diferentes períodos de tempo.

RF10 Comparações dos Gastos com os Grupos de Usuários que Participam: Comparar os gastos individuais dos usuários com outros usuários dos grupos de usuários dos quais participam.

Um caso de uso descreve como um usuário (assumindo um dos vários papéis possíveis) interage com o sistema em determinadas circunstâncias [Pressman and Maxim 2021]. Neste projeto, implementamos dois perfis de usuário: cliente e administrador, cada um com permissões específicas. O administrador tem acesso a uma interface onde pode realizar operações de CRUD (Create, Read, Update, Delete) com todas as entidades do sistema, incluindo a classe User, podendo alterar o nível de permissão de outros usuários. O perfil de cliente, por outro lado, pode registrar gastos relacionados a eletrodomésticos e transportes, além de comparar seus gastos com outros membros dos grupos dos quais participa. Esses perfis de usuário e suas respectivas operações estão representados no diagrama de casos de uso da Figura 1.

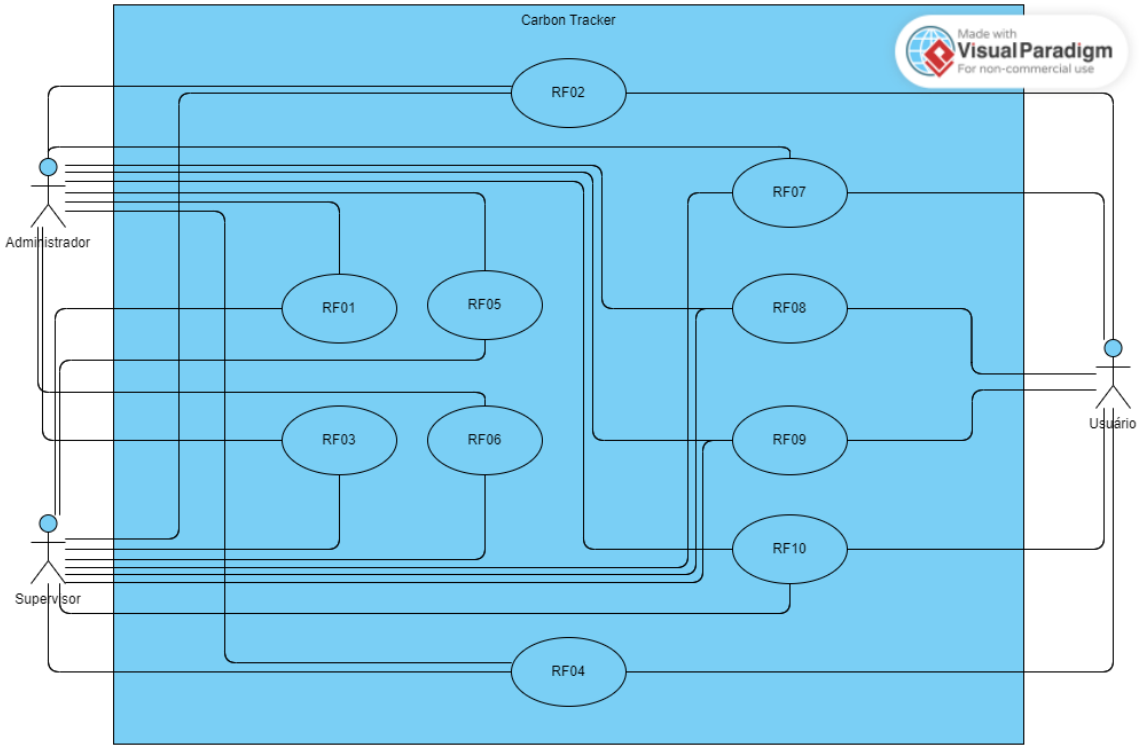

Figura 1. Diagrama de Casos de Uso

3. Arquitetura e Organização de Software

3.1. Linguagem de Programação

A escolha da linguagem de programação C# para o desenvolvimento da aplicação foi fundamentada em sua robusta capacidade de suportar a programação orientada a objetos e a programação orientada a eventos, ambas essenciais para a execução bem-sucedida deste projeto. Além disso, C# oferece uma sintaxe clara e recursos avançados que facilitam a manutenção e a escalabilidade do código.

A aplicação foi cuidadosamente projetada para operar em sistemas operacionais

Windows, garantindo compatibilidade com as versões mais recentes do .NET Framework. Esta escolha assegura um desempenho otimizado e a utilização de bibliotecas modernas e seguras fornecidas pela plataforma.

Para o desenvolvimento, foi utilizado o Visual Studio 2022 Community Edition, um ambiente de desenvolvimento integrado (IDE) que proporciona um conjunto completo de ferramentas para codificação, depuração e implementação da aplicação. A adoção do Visual Studio 2022 foi motivada por suas funcionalidades avançadas, suporte extensivo a C# e integração com o .NET Framework, o que contribui significativamente para a eficiência e a produtividade do processo de desenvolvimento.

3.2. Interface Gráfica

A interface gráfica do usuário (GUI, do inglês Graphical User Interface) revolucionou a interação com computadores ao torná-la mais intuitiva através de elementos visuais como ícones e botões. O Windows Forms, parte do .NET Framework da Microsoft, foi a tecnologia escolhida para implementar nosso software, oferecendo facilidade na criação de interfaces gráficas no ambiente Windows.

Com o Windows Forms, podemos desenvolver aplicativos que utilizam uma ampla variedade de controles, como caixas de texto e botões, usando o Visual Studio. Este ambiente de desenvolvimento fornece ferramentas de design intuitivas baseadas em arrastar e soltar, o que simplifica bastante o processo de criação. Dada a escolha da linguagem C# e o uso do IDE Visual Studio, que integra nativamente o Windows Forms, essa biblioteca se mostra ideal para nosso projeto.

3.3. Diagrama de Classes

A Unified Modeling Language (UML) é uma ferramenta essencial no processo de desenvolvimento de software, oferecendo uma abordagem padronizada para a visualização, especificação, construção e documentação dos artefatos de um sistema. A UML desempenha um papel crucial na fase inicial do desenvolvimento, pois permite a organização clara e detalhada das principais entidades de negócio envolvidas. Ao utilizar diagramas de classes, de casos de uso, de sequência e outros, a equipe de desenvolvimento pode alinhar as expectativas com os requisitos dos stakeholders, garantindo que todos compreendam a estrutura e as funcionalidades do sistema desde o início [Booch et al. 2005].

Antes do desenvolvimento do nosso software, fizemos um diagrama de classes para orientar na construção do aplicativo, verificado na figura 2. A aplicação da UML resulta em uma abordagem mais prática e estruturada para o desenvolvimento de software. Os diagramas permitem que os desenvolvedores identifiquem e resolvam problemas de design antes mesmo de começar a codificação, economizando tempo e recursos a longo prazo [Fowler 2004].

O diagrama de classes apresentado ilustra a estrutura fundamental do sistema de software, destacando as principais classes e suas interações. Na parte superior, temos as classes de repositório, como EletrodomesticoRepository, GastosEletrodomesticosRepository, TransportesRepository, GastosTransportesRepository, UsuariosRepository, GrupoUsuariosRepository e UsuariosGrupoUsuariosRepository. Essas classes são responsáveis por gerenciar entidades específicas do sistema, implementando métodos básicos para adicionar, alterar, excluir e retornar dados, seguindo o padrão de repositório. O uso do Data Access Object (DAO) nestas classes é evidente, encapsulando a lógica de

acesso a dados e proporcionando uma interface consistente para operações de CRUD.

Na parte inferior do diagrama, estão as classes de modelo, como EletrodomesticoModel, GastosEletrodomesticoModel, TransportesModel, GastosTransportesModel, UsuariosModel, GrupoUsuariosModel e UsuariosGrupoUsuariosModel. Essas classes representam as entidades de negócio do sistema com seus atributos e relacionamentos específicos. Por exemplo, a EletrodomesticoModel possui atributos como Id, Nome, TipoEletrodomestico, e KWPorHoraEletricidade, caracterizando um eletrodoméstico no contexto do sistema.

Além disso, o diagrama inclui classes enumeradas como TipoEletrodomestico e TipoCombustivel, que definem conjuntos de valores possíveis para certos atributos, garantindo a consistência dos dados. Este diagrama de classes fornece uma visão clara da arquitetura do sistema, facilitando a compreensão do design e assegurando que a implementação seja alinhada com os requisitos de negócios.

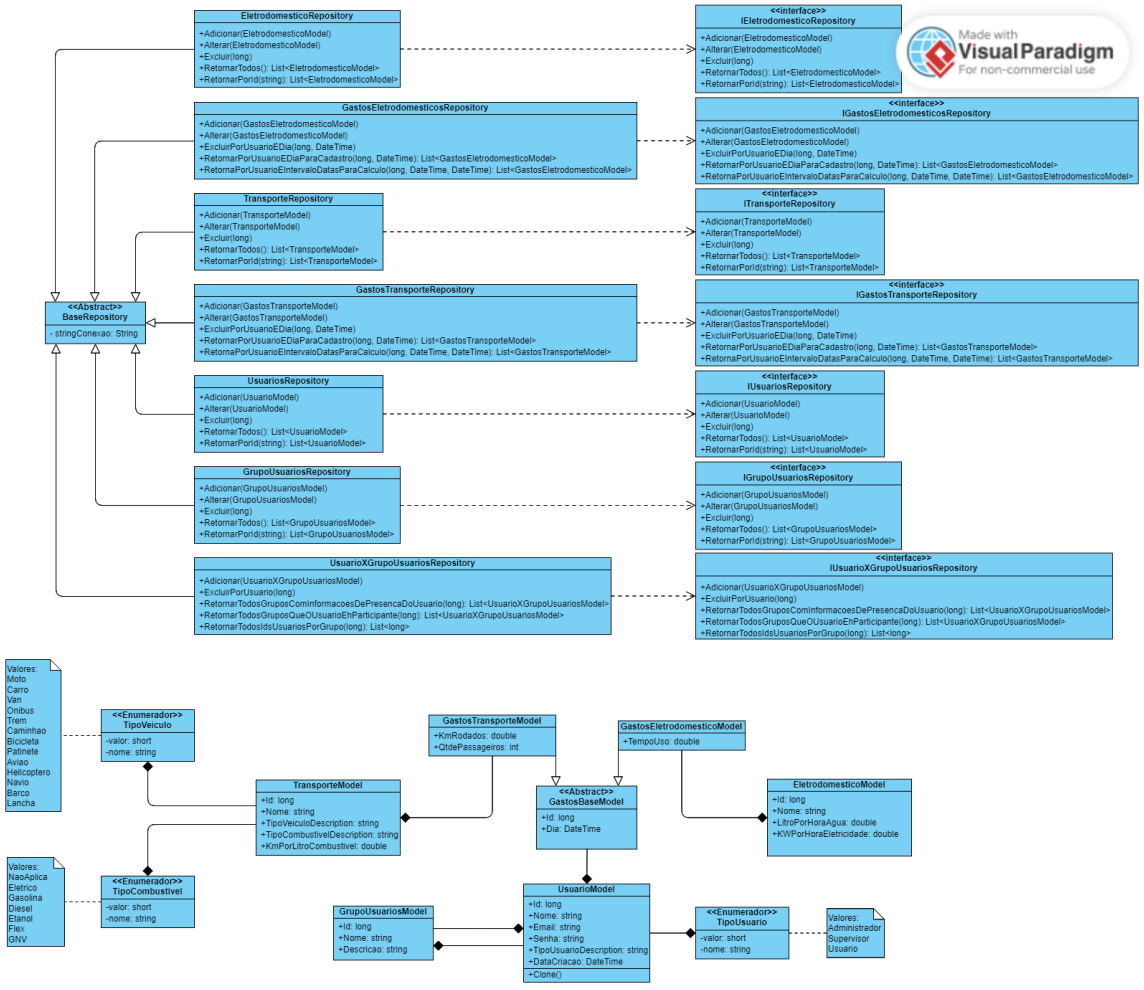

Figura 2. Diagrama de Classes

3.4. Banco de Dados

Os bancos de dados desempenham um papel fundamental no armazenamento, organização e recuperação de dados em sistemas de informação. Entre os sistemas de gerenciamento de banco de dados (SGBDs) mais populares, o PostgreSQL se destaca

como uma solução robusta e de código aberto. Ele oferece suporte a uma ampla gama de recursos avançados, como suporte transacional ACID, índices avançados, replicação, e extensibilidade através de procedimentos armazenados e funções definidas pelo usuário [Group 2023].

A Linguagem de Consulta Estruturada (SQL) é essencial para interagir com bancos de dados relacionais como o PostgreSQL. Ela fornece uma maneira padronizada de realizar consultas, inserções, atualizações e exclusões de dados. SQL permite a definição de esquemas de banco de dados, a manipulação de dados com comandos como SELECT, INSERT, UPDATE e DELETE, e a criação de restrições de integridade para garantir a consistência dos dados [Date 2004].

O Diagrama de Entidade-Relacionamento (DER ou ERD, do inglês Entity-Relationship Diagram) é uma ferramenta visual usada para modelar e descrever o esquema de um banco de dados. Ele representa entidades como tabelas, atributos como colunas e os relacionamentos entre entidades através de chaves estrangeiras. O DER facilita a compreensão da estrutura do banco de dados, ajudando na identificação de requisitos de dados e na definição de relações entre diferentes entidades [Chen 1976].

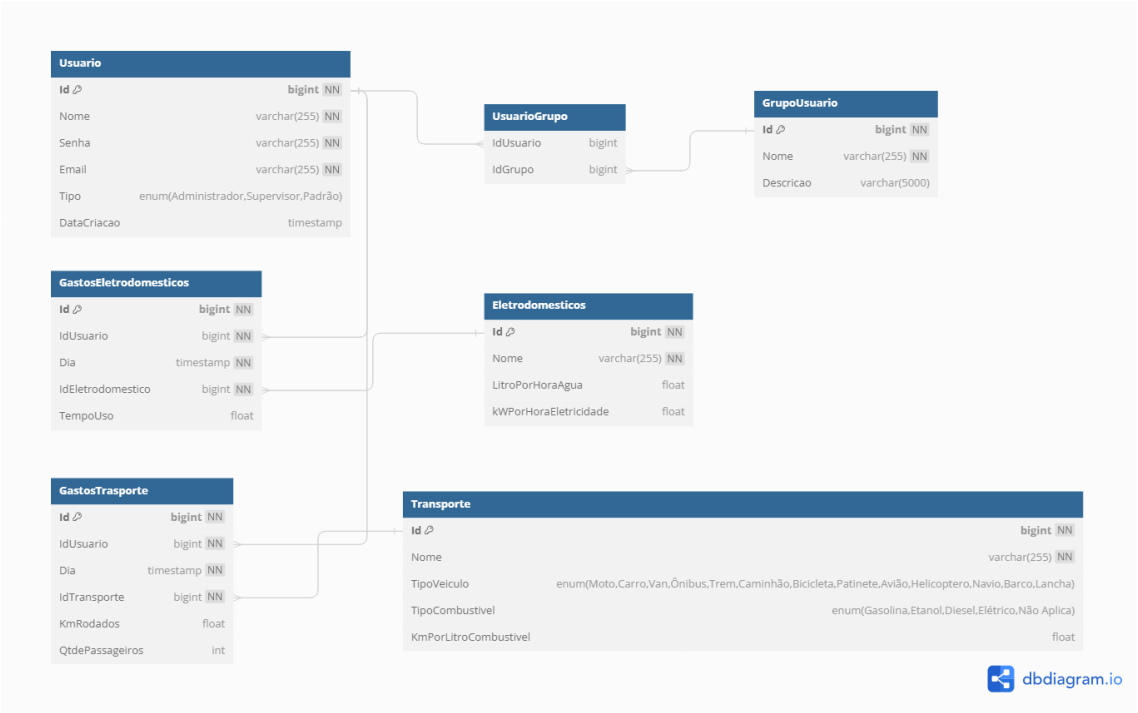

Figura 3. Diagrama de Entidade-Relacionamento

O diagrama de entidade-relacionamento (DER) apresentado ilustra a estrutura do banco de dados do sistema, destacando as principais tabelas e seus relacionamentos. A tabela Usuario armazena informações sobre os usuários do sistema, incluindo campos como Id, Nome, Senha, Email, Tipo (que pode ser Administrador, Supervisor ou Padrão) e DataCriacao. A tabela GrupoUsuario armazena informações sobre os grupos de usuários, com campos como Id, Nome e Descricao. A tabela UsuarioGrupo estabelece o relacionamento entre usuários e grupos, contendo os campos IdUsuario e IdGrupo, representando a associação de um usuário a um grupo específico.

A tabela Eletrodomesticos armazena informações sobre eletrodomésticos, incluindo

Id, Nome, LitroPorHoraAgua, e kWPorHoraEletricidade. A tabela GastosEletrodomesticos registra os gastos de eletrodomésticos, com campos como Id, IdUsuario, Dia, IdEletrodomestico e TempoUso, representando o tempo de uso de um eletrodoméstico específico por um usuário em um determinado dia. A tabela Transporte armazena informações sobre os meios de transporte, incluindo Id, Nome, TipoVeiculo, TipoCombustivel e KmPorLitroCombustivel. Os tipos de veículos e combustíveis são definidos através de enums, garantindo que os valores sejam consistentes.

A tabela GastosTransporte registra os gastos com transporte, contendo os campos Id, IdUsuario, Dia, IdTransporte, KmRodados e QtdePassageiros, representando os quilômetros rodados e o número de passageiros transportados por um usuário em um determinado dia. Este DER proporciona uma visão clara das entidades do banco de dados e de como elas se inter-relacionam. Ele ajuda a garantir a integridade referencial e a eficiência na recuperação de dados, facilitando o desenvolvimento e a manutenção do sistema.

3.5. Arquitetura utilizada

A arquitetura utilizada foi a Model-View-Presenter (MVP) que é uma derivação da arquitetura Model-View-Controller (MVC), projetada para separar mais claramente a lógica de apresentação da lógica de negócio, aumentando a testabilidade e a manutenção do código. No MVP, o Presenter atua como um intermediário entre a View e o Model. A View exibe dados e delega a lógica de interação ao Presenter, enquanto o Model é responsável pela lógica de negócios e pela manipulação de dados. Este padrão é frequentemente usado em aplicações desktop e mobile, onde a separação clara das responsabilidades é crucial para a criação de interfaces de usuário interativas e responsivas. A introdução de um Repository, que em nosso software utiliza o padrão Data Access Object (DAO) para gerenciar a lógica de acesso aos dados, e de interfaces para View e Model são variações que aumentam ainda mais a modularidade e a flexibilidade da arquitetura [Fowler 2002].

Na utilização do MVP no desenvolvimento de nosso software percebemos várias vantagens. O Presenter coordena as interações entre a View e o Model, recebendo eventos da View e solicitando atualizações do Model, além de manipular os dados recebidos e atualizar a View conforme necessário. O Repository atua como uma camada adicional de abstração entre o Model e as fontes de dados, facilitando o acesso e a manipulação dos dados, e permitindo uma maior flexibilidade na troca ou atualização das fontes de dados sem impactar outras partes do sistema. O uso do padrão Data Access Object (DAO) dentro do Repository ajuda a gerenciar a lógica de acesso aos dados. As interfaces para View e Model promovem um acoplamento fraco e permitem a substituição fácil das implementações concretas, melhorando a testabilidade e a manutenção do código. Cada componente tem uma responsabilidade bem definida, promovendo uma arquitetura mais limpa e sustentável a longo prazo.

4. Funcionalidade do Software

Ao inicializar a aplicação, o usuário é direcionado para a tela de login, conforme mostrado na Figura 4. Para acessar o sistema, o usuário deve entrar em contato com o administrador para ser pré-cadastrado.

Após o login, a tela inicial do sistema é exibida, com o menu lateral e a logo do Carbon Tracker. No menu, temos as seguintes opções: Informações de Usuários (contém

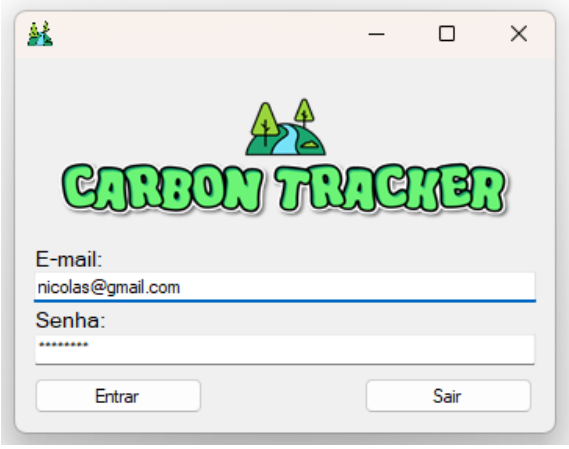

Figura 4. Tela de Login

informações do usuário, opção para alterar senha, cadastro de grupos de usuários e entrar em um grupo), Cadastro de Gastos (inclui cadastro de eletrodomésticos e de transportes), Registro de Gastos (permite registrar os gastos do dia, sejam de eletrodomésticos ou de transportes) e Comparações (permite comparar os gastos entre grupos de usuários e visualizar o resultado do cálculo da pegada de carbono).

Figura 5. Menu

O sistema possui quatro telas dedicadas a cadastros: cadastro de transportes, eletrodomésticos, grupos de usuários e usuários. Todas as telas seguem as funcionalidades de um CRUD (Create, Read, Update, Delete), conforme exemplificado na Figura 6. No cadastro de transportes e eletrodomésticos, é registrado o impacto ambiental de cada item, para que o usuário possa posteriormente registrar seus gastos diários. No cadastro de usuários, é possível registrar um usuário como supervisor ou padrão, sendo que haverá apenas um administrador. O administrador define a senha do usuário, que poderá alterá-la

posteriormente. No cadastro de grupo de usuários, são registrados o nome e a descrição do grupo, para que os usuários possam se juntar a ele posteriormente.

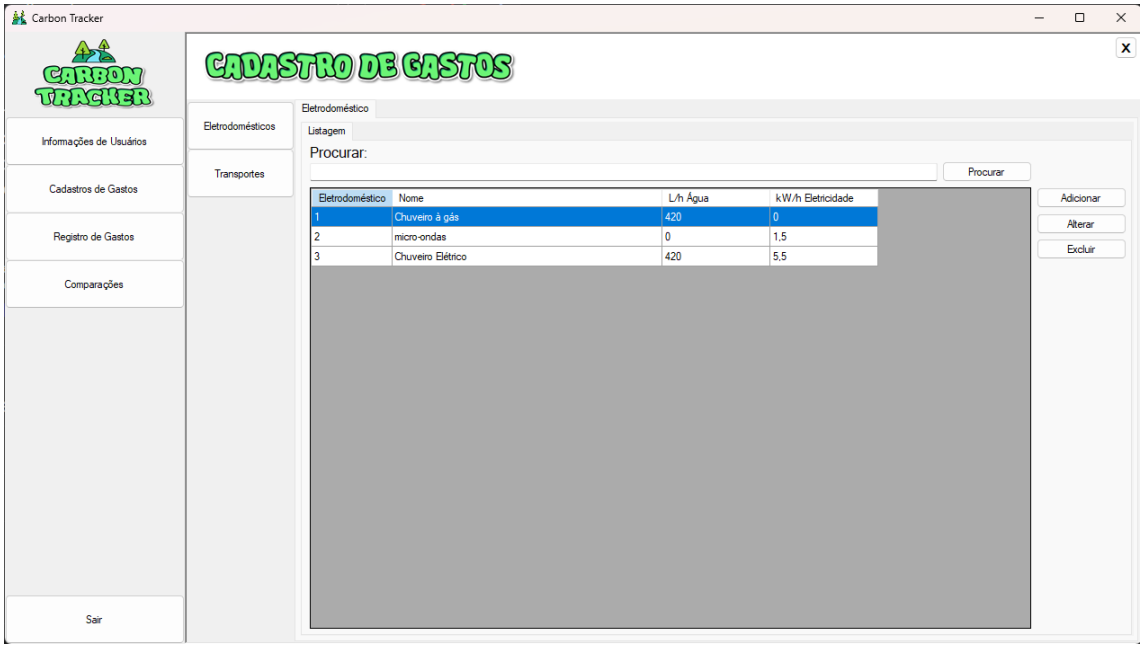

Figura 6. Cadastro de Eletrodoméstico

Nas Informações de Usuários, há alguns submenus. O submenu ”Entrar no Grupo”lista os grupos de usuários disponíveis, permitindo que o usuário marque e salve os grupos desejados, conforme a Figura 7. Esses grupos são utilizados para comparações entre os usuários.

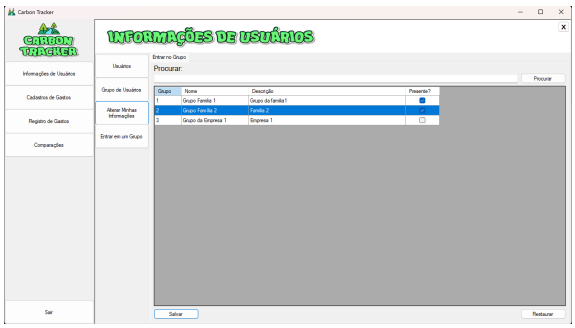

Figura 7. Entrar no Grupo

Outro submenu é ”Alterar Minhas Informações”, que permite ao usuário logado redefinir sua senha. Após redefinir a senha, o usuário precisará fazer login novamente com a nova senha.

O sistema possui permissões específicas para os usuários. Quando um usuário não tem permissão para acessar uma tela, uma mensagem de restrição é exibida, conforme mostrado na Figura 9.

Na parte de Registro de Gastos, existem dois submenus: Eletrodomésticos e Transportes. Ambas as telas listam os itens cadastrados e solicitam que o usuário informe

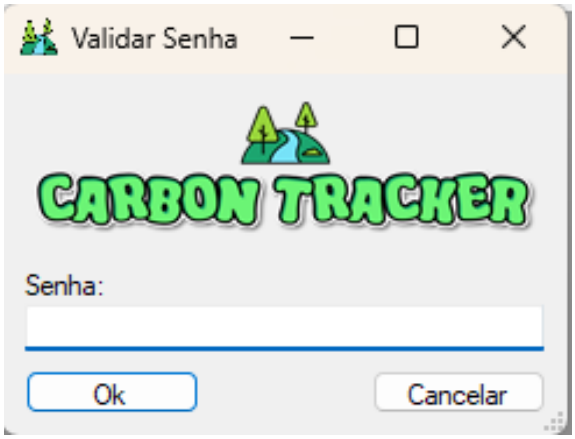

Figura 8. Alterar Minhas Informações

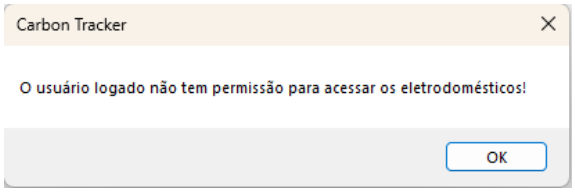

Figura 9. Permissão negada

os minutos de uso, salvando os dados em seguida. O usuário pode escolher a data para o registro, permitindo registrar gastos de dias anteriores caso tenha esquecido, conforme a Figura 10. Esses registros são essenciais para o objetivo principal do nosso software: calcular a pegada de carbono e mostrar ao usuário o gasto excessivo de alguns recursos.

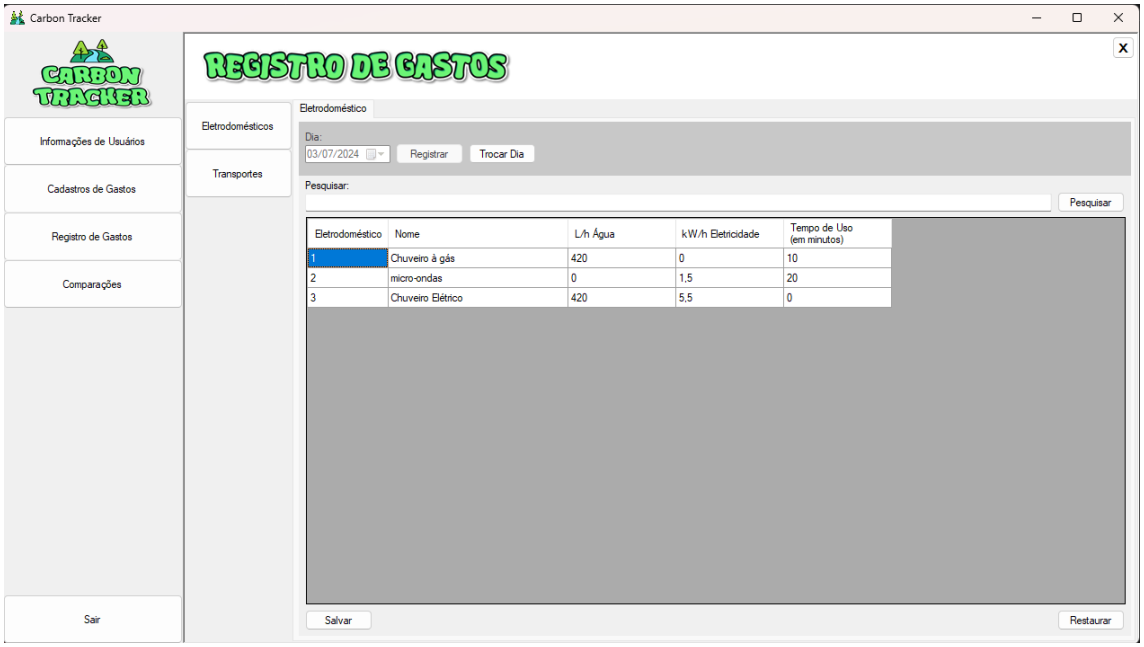

Figura 10. Registro de Gastos

Por último, temos as comparações que colocam os usuários frente aos seus grupos,

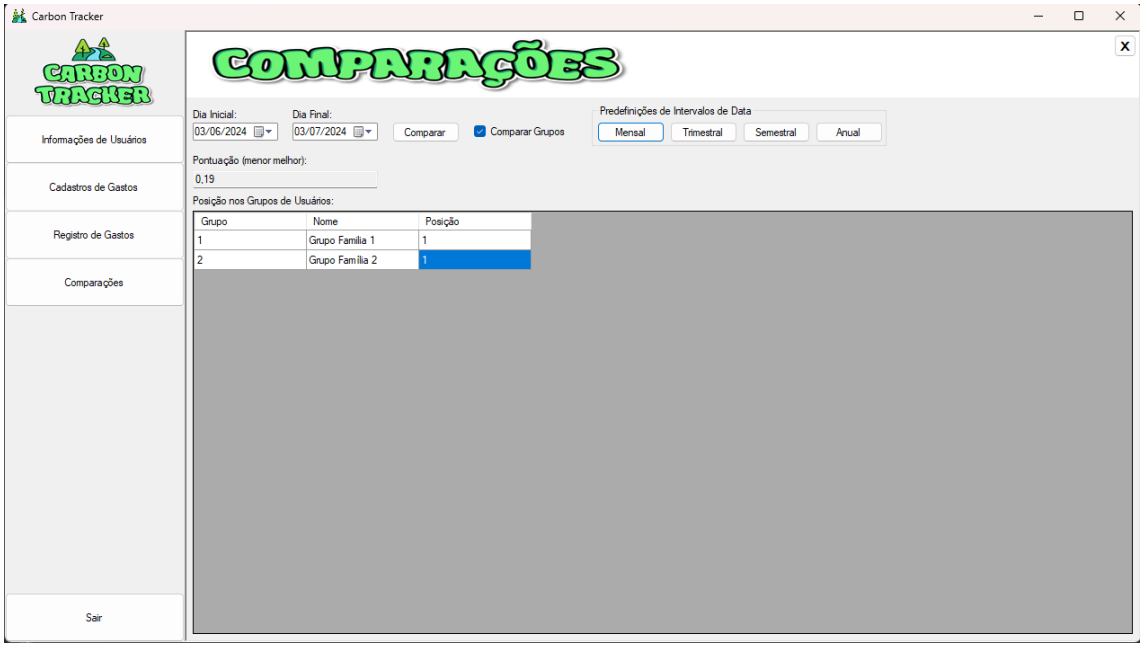

Figura 11. Comparações

exibindo a posição do usuário logado entre seus grupos. Essa funcionalidade opera como uma forma de incentivar nossos clientes através de uma ”competição”saudável, visando favorecer cada vez mais a utilização dos recursos naturais e a redução de gases poluentes ao meio-ambiente. A tela funciona da seguinte maneira: o usuário pode escolher uma data inicial e uma final, um período específico (mensal, trimestral, semestral ou anual). Ele pode optar por comparar sua pegada de carbono com a média do grupo ao qual pertence. Em seguida, são listados todos os grupos dos quais faz parte, mostrando a posição relativa em cada um deles.

Por fim, a pontuação exibida é o resultado do cálculo da pegada de carbono, que combina as emissões provenientes do transporte e dos eletrodomésticos do usuário durante o período selecionado. Para o transporte, o sistema considera quilômetros rodados, tipo de combustível utilizado (gasolina, diesel, etanol, GNV, elétrico), eficiência do veículo em km/l e número de passageiros. Já para os eletrodomésticos, são levados em conta o consumo de energia elétrica em kWh e o consumo de água em litros. Esses dados são processados para determinar a quantidade de dióxido de carbono (CO2) emitida, contribuindo para a pontuação total da pegada de carbono do usuário.

5. Considerações Finais

O desenvolvimento do Carbon Tracker foi uma experiência valiosa que destacou a importância das soluções tecnológicas na luta contra as mudanças climáticas. Durante o processo, aprendemos muito sobre as etapas de criação de software, desde a ideia inicial até a implementação e testes.

5.1. Aprendizados Técnicos

No projeto, utilizamos C# para garantir um aplicativo robusto e PostgreSQL para gerenciar dados de forma eficaz. Aprendemos a importância de boas práticas de programação, como a separação clara entre a lógica de apresentação e a de negócio, o que facilita a

manutenção do código. Também percebemos o valor dos diagramas para planejar e comunicar as estruturas do software.

5.2. Futuras Melhorias

Apesar de útil, o Carbon Tracker ainda precisa de muitas melhorias para atingir todo o seu potencial. Algumas ideias para o futuro incluem:

- **Versão Mobile**: Facilitaria o uso diário e aumentaria o alcance do aplicativo.
- **Inteligência Artificial**: Poderia fornecer recomendações mais personalizadas para os usuários.
- **Novas Métricas**: Incluir dados sobre consumo de alimentos, uso de vestimentas e outras práticas para uma visão mais completa do impacto ambiental.

5.3. Conclusão

Embora o Carbon Tracker ofereça uma ferramenta útil para o monitoramento da pegada de carbono, ele ainda não cumpre totalmente o seu propósito de ajudar os usuários a reduzirem significativamente seu impacto ambiental. Isso exige um maior desenvolvimento e a adição de novas funcionalidades. No entanto, ele representa um passo importante para conscientizar e incentivar escolhas mais sustentáveis, mostrando como pequenas ações podem contribuir para um futuro mais verde.

6. Referências Bibliográficas

As referências bibliográficas estão apresentadas a seguir. Cada obra citada contribui significativamente para a compreensão e análise dos aspectos discutidos. A inclusão dessas fontes tem por objetivo contextualizar informações e demonstrar a profundidade da pesquisa realizada.

Referências

[Booch et al. 2005] Booch, G., Rumbaugh, J., and Jacobson, I. (2005). *The Unified Modeling Language User Guide*. Addison-Wesley, Boston, MA, 2 edition.

[Chen 1976] Chen, P. P. (1976). The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36.

[Date 2004] Date, C. J. (2004). *An Introduction to Database Systems*. Addison-Wesley.

[Fowler 2002] Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley, Boston, MA.

[Fowler 2004] Fowler, M. (2004). *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, Boston, MA, 3 edition.

[Group 2023] Group, P. G. D. (2023). *PostgreSQL Documentation*. PostgreSQL Global Development Group.

[Pressman and Maxim 2021] Pressman, R. S. and Maxim, B. R. (2021). *Engenharia de Software*. Grupo A. E-book.