

Vizzy.

Vizzy - Plataforma Comunitária

Projeto de Desenvolvimento de Software

ENRIQUE RODRIGUES Nº28602

JOSÉ ALVES Nº27967

DIOGO MACHADO Nº26042

DIOGO ABREU Nº27975

ANDRÉ SILVA Nº27965

Instituto Politécnico do Cávado e do Ave

6 de março de 2025

1 Introdução

A *Vizzy* é uma plataforma comunitária desenvolvida para facilitar a compra, venda, troca e aluguer de itens entre pessoas da mesma comunidade, promovendo uma forma prática, económica e sustentável de partilhar recursos.

Através da *Vizzy*, os utilizadores podem adquirir produtos em segunda mão, trocar itens ou até alugar ferramentas, entre outras coisas. Isto não só reduz o desperdício, como também oferece uma alternativa mais acessível, especialmente quando se trata de alugueres. Alugar uma máquina a um vizinho, por exemplo, pode ser muito mais económico do que comprá-la.

O objetivo da plataforma é criar uma rede de partilha dentro de uma comunidade, onde os membros possam beneficiar de um consumo mais consciente, poupando dinheiro e recursos, enquanto promovem a sustentabilidade e a cooperação.

Neste relatório, serão apresentados os principais detalhes da aplicação, incluindo diagramas UML e *mockups*, que ilustram as funcionalidades essenciais da *Vizzy*.

2 Problema e Motivação

2.1 O Problema

Nos dias de hoje, muitas comunidades enfrentam desafios económicos e ambientais relacionados com o consumo excessivo e a falta de acesso a certos produtos. Itens valiosos acabam por ser desperdiçados ou ficam esquecidos, enquanto outras pessoas na mesma comunidade poderiam beneficiar deles. Ao mesmo tempo, a aquisição de novos produtos nem sempre é uma opção viável devido ao custo ou à disponibilidade limitada.

Outro desafio importante é a falta de soluções práticas para facilitar a troca, venda e aluguer de itens de forma organizada, o que leva a um ciclo de consumo ineficiente e ao desperdício de recursos.

2.2 A Solução Proposta

A *Vizzy* surge como uma resposta a estes problemas, oferecendo uma plataforma onde os membros da comunidade podem comprar, vender, trocar e alugar itens de forma simples e segura. Com esta solução, é possível dar uma segunda vida a produtos que seriam descartados e permitir que as pessoas accedam a bens de que necessitam, sem precisarem de comprá-los novos.

Além disso, ao permitir o aluguer de ferramentas e outros objectos, a *Vizzy* oferece uma alternativa económica e prática, como no caso de alugar uma ferramenta a um vizinho em vez de recorrer a serviços externos ou grandes empresas. Isto não só reduz os custos para os utilizadores, mas também contribui para um modelo de consumo mais sustentável.

A plataforma visa, assim, otimizar o uso dos recursos, reduzir desperdícios e fortalecer os laços entre membros da comunidade. Com a *Vizzy*, todos podem beneficiar de uma forma mais acessível e consciente de consumir, promovendo a circulação de bens e a sustentabilidade.

3 Processos de Negócio

3.1 Processo de Empréstimo/Aluguer

3.1.1 Etapas

- **Criar um anúncio:** O utilizador cria um anúncio na plataforma, fornecendo detalhes como o produto que pretende alugar/emprestar, descrição do mesmo, estado e fotografias.
- **Definir datas bloqueadas:** O utilizador indica, caso existam, as datas em que o produto não estará disponível para empréstimo/aluguer.
- **Confirmar datas livres:** O sistema verifica e confirma as datas disponíveis para empréstimo/aluguer com base nas datas bloqueadas.
- **Definir valor associado:** O utilizador define o valor do aluguer (0€ em caso de empréstimo).
- **O anúncio não desaparece:** Durante o empréstimo/aluguer, o anúncio permanece na plataforma para futuras marcações.

3.1.2 Fluxo do Processo

Um utilizador cria o anúncio e define as datas indisponíveis (fazendo o sistema a gestão das datas disponíveis, com base nesta informação). Outros utilizadores podem solicitar o empréstimo/aluguer do produto nas datas livres. O proprietário do produto confirma ou rejeita a solicitação. O anúncio continua a estar disponível na plataforma para marcações em datas livres.

3.1.3 Interações Essenciais

- Possibilidade de enviar uma contra-proposta no anúncio. Por exemplo, em vez de alugar um corta-relva, o utilizador pode sugerir emprestar um soprador.

3.2 Processo de Venda

3.2.1 Etapas

- **Criar um anúncio:** O utilizador cria um anúncio na plataforma, fornecendo detalhes como o produto que pretende vender, descrição do

mesmo, estado e fotografias.

- **Definir um valor associado:** O utilizador define o preço de venda do produto (0€, no caso de ser um *giveaway*).
- **Permitir ou não receber contrapropostas:** O utilizador decide se aceita ou não receber contrapropostas de outros utilizadores.
- **Aceitar/Rejeitar contrapropostas:** Caso permita, o vendedor pode aceitar ou rejeitar contrapropostas recebidas.
- **Concretizada a venda, o anúncio desaparece:** Após a venda, o anúncio é marcado como concluído, deixando de aparecer nas pesquisas.

3.2.2 Fluxo do Processo

O utilizador cria o anúncio e define o preço. Outros utilizadores podem comprar diretamente ou fazer contrapropostas (se permitido pelo vendedor). O vendedor aceita, rejeita ou faz ele uma nova contraproposta. Após a venda, o anúncio é removido das pesquisas, ficando guardado para efeitos de histórico.

3.3 Processo de Troca

3.3.1 Etapas

- **Criar um anúncio:** O utilizador cria um anúncio na plataforma, fornecendo detalhes como o produto que pretende vender, descrição do mesmo, estado e fotografias.
- **Associar um produto para troca:** O utilizador indica que produto deseja receber em troca.
- **Aceitar a proposta:** Outros utilizadores podem aceitar a proposta, oferecendo o produto solicitado.
- **Fazer contrapropostas:** Outros utilizadores podem fazer contrapropostas com produtos diferentes.
- **Aceitar/Rejeitar contrapropostas:** O utilizador pode aceitar ou rejeitar as contrapropostas recebidas.

3.3.2 Fluxo do Processo

O utilizador cria o anúncio e indica o produto que deseja receber em troca. O utilizador aceita ou rejeita as contrapropostas recebidas. Após a troca, o anúncio é removido das pesquisas.

3.4 Fluxo Geral

- O utilizador escolhe o tipo de transação (empréstimo/aluguer, venda ou troca).
- O utilizador cria o anúncio e define os detalhes específicos para cada processo.
- Outros utilizadores interagem com o anúncio (solicitam empréstimo, compram ou propõem trocas).
- O proprietário do produto confirma ou rejeita as solicitações.
- Após a transação, o anúncio é marcado como concluído (no caso da venda).
- No caso de empréstimo/aluguer, o anúncio permanece disponível, com as datas disponíveis atualizadas.

4 Requisitos Funcionais

Os requisitos funcionais (**RFs**) são as especificações que definem o que um sistema deve fazer para atender às necessidades dos utilizadores. Eles descrevem as funcionalidades, comportamentos e operações que o sistema deve oferecer, incluindo interações entre utilizadores e o sistema, processamento de dados e regras de negócio.

Código	Requisito Funcional
RF 01	O utilizador pode criar um anúncio na aplicação, fornecendo nome, descrição, estado do produto, fotografias e definir se é para venda, troca ou empréstimo/aluguer.
RF 02	O utilizador pode definir datas bloqueadas para empréstimo/aluguer do produto.
RF 03	O sistema deve verificar e confirmar as datas livres para empréstimo/aluguer.
RF 04	O utilizador pode definir um valor para aluguer ou venda.
RF 05	O utilizador pode permitir ou não propostas nos anúncios.
RF 06	O utilizador pode aceitar, rejeitar ou negociar propostas.
RF 07	Após a transação de um produto para venda ou troca, o anúncio deve ser marcado como concluído e deixar de aparecer para os outros utilizadores.
RF 08	O utilizador, ao criar um anúncio para troca, pode indicar qual produto deseja receber em troca.
RF 09	A aplicação deve permitir a pesquisa de produtos disponíveis na plataforma.
RF 10	O utilizador pode adicionar produtos a uma lista de desejos/favoritos.

Tabela 1: Requisitos Funcionais

5 Requisitos Não Funcionais

Os requisitos não funcionais (RNFs) definem as características e restrições de um sistema que não estão diretamente relacionadas com as funcionalidades oferecidas, mas sim à qualidade, desempenho, segurança e usabilidade da aplicação. Eles garantem que o sistema seja eficiente, confiável e utilizável dentro de determinados padrões.

Código	Requisito Não Funcional
RNF 01	A interface do utilizador deve ser intuitiva e acessível para todas as faixas etárias.
RNF 02	Todos os dados dos utilizadores devem ser armazenados de forma segura.
RNF 03	A aplicação deve cumprir com o Regulamento Geral de Proteção de Dados (RGPD) da União Europeia, garantindo a privacidade e o consentimento para o uso de dados pessoais.
RNF 04	O tempo de carregamento do feed da aplicação não deve ultrapassar os 3 segundos.
RNF 05	A pesquisa de produtos deve devolver os resultados em menos de 5 segundos.
RNF 06	O sistema deve permitir a implementação de novas funcionalidades com um impacto mínimo no desempenho.
RNF 07	A aplicação deve exigir uma palavra-passe com pelo menos 8 caracteres, incluindo letras e números.
RNF 08	O sistema deve utilizar armazenamento em cache para reduzir o número de pedidos ao servidor e melhorar a eficiência.
RNF 09	O sistema deve limitar o tamanho máximo de imagens carregadas para 5MB, evitando uso excessivo de espaço.

Tabela 2: Requisitos Não Funcionais

6 Arquitetura

A arquitetura da plataforma é composta por um *frontend* desenvolvido com **React** e **Next.js**, um *backend* em **C#**, e uma base de dados gerida com **Supabase (PostgreSQL)**. Esta abordagem garante um sistema eficiente, seguro e escalável.

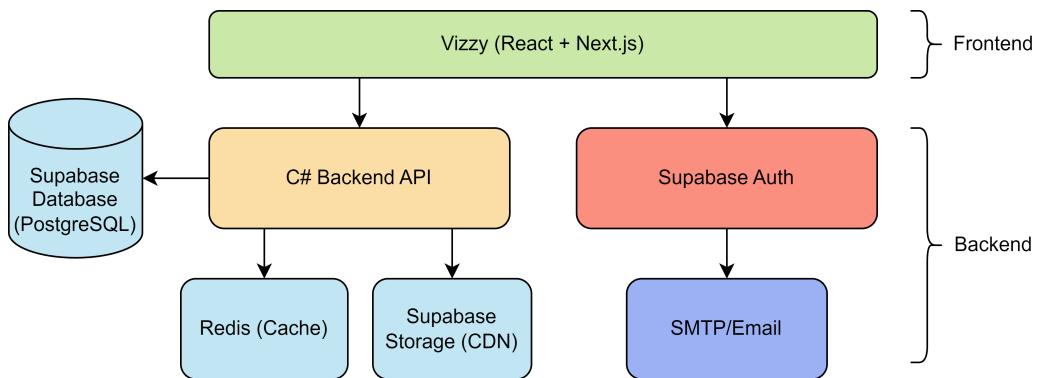


Figura 1: Arquitetura da Plataforma

6.1 Frontend

O *frontend* foi desenvolvido com **Next.js** para tirar proveito do *server-side rendering* (SSR) e *static site generation* (SSG), o que melhora o desempenho e o SEO. O frontend comunica diretamente com o **API Backend (C#)**, chamando os endpoints para obter e manipular dados.

6.2 Backend

O *backend* é composto por diversos serviços que trabalham em conjunto para garantir a funcionalidade da plataforma:

- **API Backend (C#):** Responsável pelo processamento dos pedidos HTTP do *frontend*, gestão de dados e comunicação com a base de dados. A API também implementa caching com **Redis** para melhorar o desempenho, reduzindo a carga na base de dados e melhorando o tempo de resposta.
- **Redis (Cache):** Utilizado para guardar em cache respostas frequentes da API e melhorar a eficiência do sistema.

- **Supabase Auth:** Sistema de autenticação e gestão de utilizadores, garantindo segurança no acesso à plataforma.
- **SMTP/Email:** Serviço de envio de emails para recuperação de conta.
- **Supabase Storage (CDN):** Responsável por armazenar ficheiros e gerir recursos estáticos. A API busca os URLs dos ficheiros armazenados e devolve-os ao chamador da API.

6.3 Base de Dados

A plataforma utiliza o **PostgreSQL** como base de dados relacional. Esta base de dados oferece alta escalabilidade, confiabilidade e desempenho, permitindo operações eficientes de leitura e escrita.

6.4 Fluxo de Funcionamento

1. O utilizador interage com o **frontend (Vizzy)**.
2. Os pedidos são enviados para o **API Backend (C#)**, que os processa e consulta a base de dados. Para melhorar o desempenho, o **Redis** atua como cache, guardando dados frequentes. Isto reduz consultas repetitivas à base de dados.
3. A autenticação é gerida pelo **Supabase Auth**, garantindo a segurança no acesso.
4. Os ficheiros carregados são armazenados no **Supabase Storage (CDN)**, e a API backend obtém os URLs dos ficheiros para passá-los ao chamador da API.
5. O serviço de email (SMTP) é utilizado principalmente para a recuperação de senhas e outras funcionalidades de gestão de contas.

Esta arquitetura assegura um sistema robusto, seguro e otimizado, permitindo uma gestão eficiente dos recursos e uma experiência fluida para os utilizadores.

7 User Stories

O *Product Backlog* do projeto está definido e vai sendo atualizado através dos seguintes *links*:

- *Excel*
- *Jira*

ID	Como Utilizador...	Pretendo...
VIZZY-13	Utilizador Não Registado	Criar conta
VIZZY-14	Utilizador Não Registado	Fazer login
VIZZY-15	Utilizador Não Registado	Reset Password
VIZZY-17	Utilizador Registado	Alterar Password
VIZZY-18	Utilizador Registado	Apagar conta
VIZZY-41	Utilizador Registado	Criar uma publicação (venda/empréstimo/troca/aluguer)
VIZZY-45	Utilizador Registado	Aceitar/Rejeitar Proposta
VIZZY-46	Utilizador Registado	Comprar/Alugar um produto
VIZZY-50	Utilizador Registado	Ver o perfil de outros utilizadores
VIZZY-51	Utilizador Registado	Editar o próprio perfil
VIZZY-52	Utilizador Registado	Adicionar foto de perfil
VIZZY-57	Utilizador Registado	Lista de desejos/favoritos
VIZZY-58	Todos Os Utilizadores	Pesquisa avançada
VIZZY-60	Utilizador Registado	Anexar fotografias a uma proposta
VIZZY-61	Utilizador Registado	Enviar uma proposta/contraproposta
VIZZY-111	Todos Os Utilizadores	Ver Anúncios

Tabela 3: Tabela de Funcionalidades com Links para Jira

8 Diagrama de casos de Uso

Um diagrama de casos de uso é uma representação visual que descreve as interações entre os utilizadores (atores) e um sistema. Este mostra as funcionalidades que o sistema oferece e quem pode utilizá-las, ajudando a compreender os requisitos funcionais de uma aplicação.

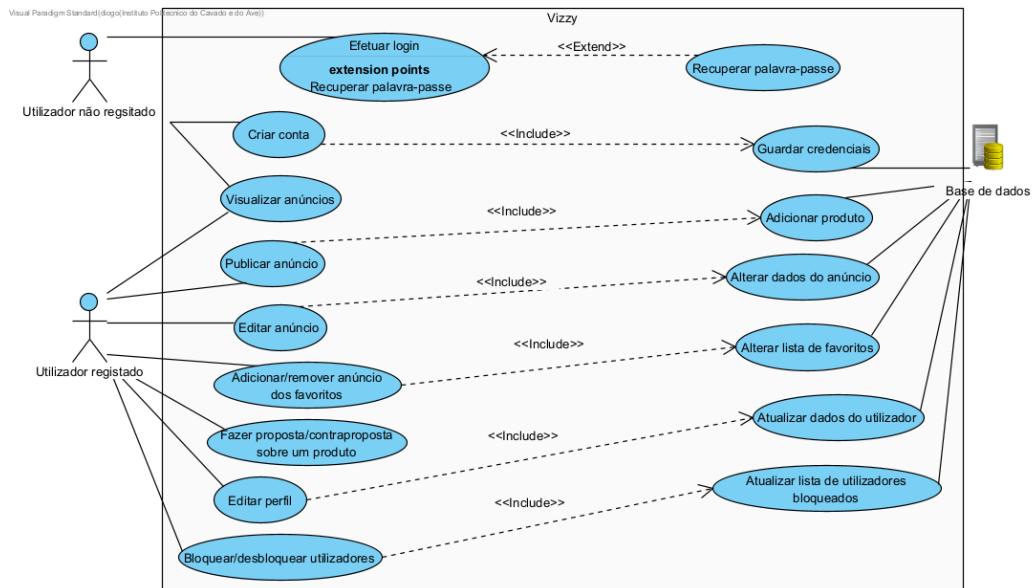


Figura 2: Diagrama de casos de uso

O diagrama de casos de uso apresentado ilustra as principais interações dos utilizadores com o sistema *Vizzy*. Existem dois tipos de utilizadores: **Utilizador não registado** e **Utilizador registrado**, cada um com diferentes permissões e funcionalidades disponíveis.

8.1 Utilizador Não Registado

Os utilizadores que ainda não possuem conta podem realizar as seguintes ações:

- **Efetuar login** – Possibilidade de iniciar sessão no sistema. Este caso de uso pode ser estendido pela funcionalidade de **Recuperar palavra-passe**, caso o utilizador tenha perdido as credenciais de acesso.

- **Criar conta** – Processo de registo de novos utilizadores, que inclui a ação de **Guardar credenciais** na base de dados.
- **Visualizar anúncios** – Permite que os utilizadores não registados consultem os anúncios disponíveis.

8.2 Utilizador Registado

Após o registo e autenticação, os utilizadores passam a ter acesso a funcionalidades adicionais, tais como:

- **Publicar anúncio** – Possibilidade de adicionar um novo produto ao sistema. Este caso de uso inclui a funcionalidade de **Adicionar produto** à base de dados.
- **Editar anúncio** – Permite modificar um anúncio já publicado, incluindo a funcionalidade de **Alterar dados do anúncio**.
- **Adicionar/remover anúncio dos favoritos** – Possibilidade de gerir uma lista de favoritos, incluindo a ação de **Alterar lista de favoritos**.
- **Fazer proposta/contraproposta sobre um produto** – Funcionalidade que permite negociar produtos entre utilizadores registados.
- **Editar perfil** – Opção para alterar as informações da conta, incluindo a funcionalidade de **Atualizar dados do utilizador**.
- **Bloquear/desbloquear utilizadores** – Permite que um utilizador bloquee ou desbloqueie outros, com a ação correspondente de **Atualizar lista de utilizadores bloqueados** na base de dados.

O diagrama demonstra as relações entre os casos de uso através de conexões do tipo **include** («include») e **extend** («extend»), assegurando que determinadas ações dependem de outras funcionalidades do sistema. Além disso, todas as operações que envolvem a manipulação de dados refletem diretamente na **Base de Dados**, que armazena e gere as informações essenciais para o funcionamento da plataforma.

9 Diagramas BPMN

Os **diagramas BPMN** (*Business Process Model and Notation*) são uma forma padronizada de representar visualmente processos de negócio. Eles utilizam símbolos gráficos para descrever o fluxo de atividades, decisões e interações dentro de uma aplicação, facilitando a compreensão e a análise dos processos de negócio.

9.1 Diagrama BPMN das vendas

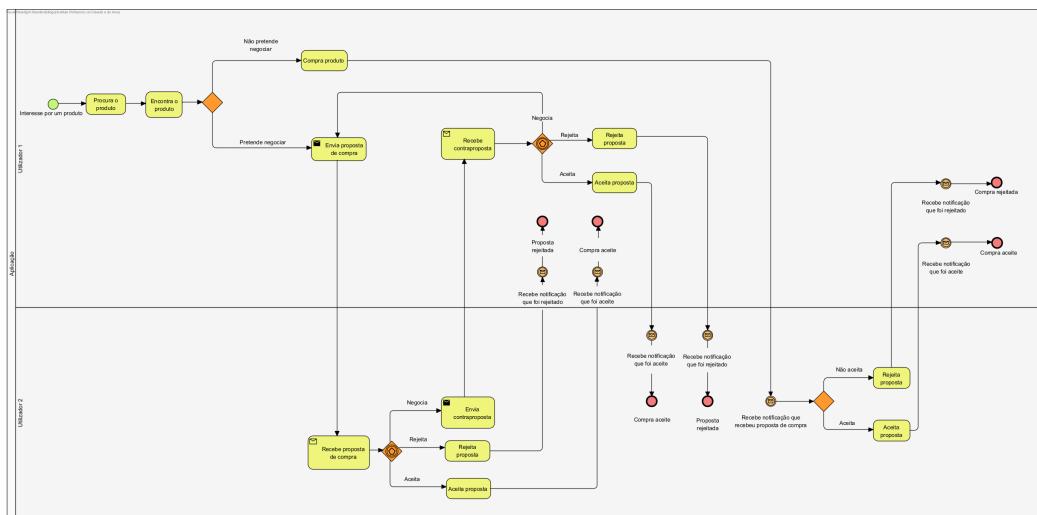


Figura 3: Diagrama BPMN das vendas

O diagrama BPMN apresentado representa o processo de negociação e compra de um produto entre dois utilizadores. O fluxo de atividades está dividido em duas *pools*, correspondendo ao **Utilizador 1** e ao **Utilizador 2**, cada um desempenhando diferentes papéis dentro do processo.

O processo inicia-se quando o **Utilizador 1** demonstra **interesse por um produto**. A partir deste ponto, segue-se o seguinte fluxo:

- O **Utilizador 1** realiza a **procura pelo produto** e, ao encontrá-lo, decide se pretende negociar ou comprá-lo diretamente.
- Se optar por **não negociar**, procede à **compra do produto**.
- Caso opte por negociar, envia uma **proposta de compra** ao **Utilizador 2**.

- O Utilizador 2 pode:
 - Aceitar a proposta, finalizando a compra.
 - Rejeitar a proposta, encerrando a negociação.
 - Enviar uma contraproposta, iniciando um ciclo de negociação.
- Se o Utilizador 1 receber uma contraproposta, pode novamente decidir entre aceitá-la, rejeitá-la ou enviar uma nova proposta.

O processo pode resultar em dois desfechos:

- **Compra aceite:** Quando uma das partes aceita a proposta ou contraproposta.
- **Compra rejeitada:** Se uma das partes rejeitar a proposta.

9.2 Diagrama BPMN dos empréstimos/alugueres

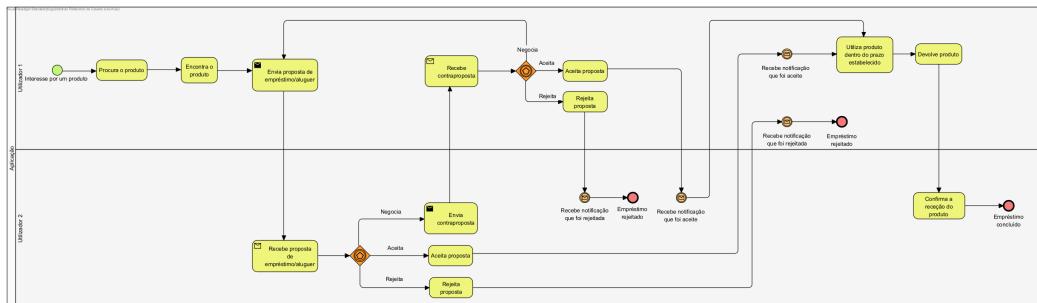


Figura 4: Diagrama BPMN dos empréstimos/alugueres

O diagrama BPMN apresentado representa o processo de negociação de empréstimo/aluguer de um produto entre dois utilizadores. O fluxo de atividades está dividido em duas *pools*, correspondendo ao **Utilizador 1** e ao **Utilizador 2**, cada um desempenhando diferentes papéis dentro do processo. O processo inicia-se quando o **Utilizador 1** demonstra **interesse por um produto**. A partir deste ponto, segue-se o seguinte fluxo:

- O Utilizador 1 realiza a **procura pelo produto** na aplicação.
- Após encontrar o produto, envia uma **proposta de empréstimo/aluguer** ao Utilizador 2.

- O Utilizador 2 pode:
 - Aceitar a proposta, iniciando o processo de empréstimo.
 - Rejeitar a proposta, encerrando a negociação.
 - Enviar uma contraproposta, iniciando um ciclo de negociação.
- Se o Utilizador 1 receber uma contraproposta, pode novamente decidir entre aceitá-la, rejeitá-la ou enviar uma nova proposta.

O processo pode resultar em dois desfechos:

- **Empréstimo aceite:** Quando uma das partes aceita a proposta ou contraproposta. O produto é utilizado dentro do prazo estabelecido e depois devolvido.
- **Empréstimo rejeitado:** Se uma das partes rejeitar a proposta, o processo é encerrado.

Após a devolução do produto, o Utilizador 2 confirma a sua receção, concluindo o processo de empréstimo.

9.3 Diagrama BPMN das trocas

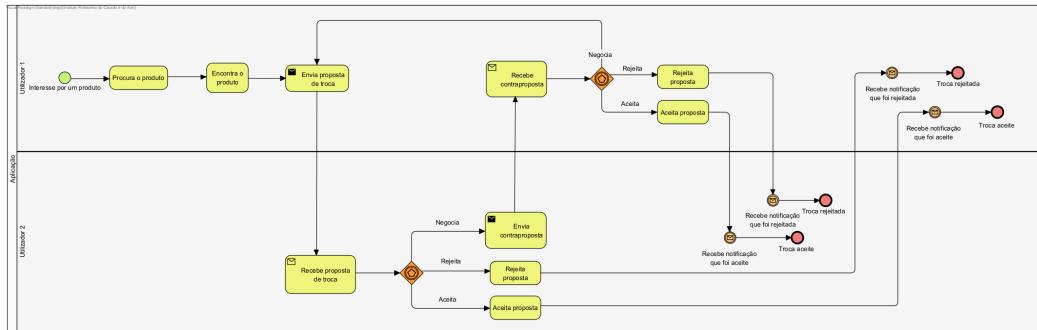


Figura 5: Diagrama BPMN das trocas

O diagrama BPMN apresentado descreve o processo de negociação e troca de produtos entre dois utilizadores. O fluxo está dividido em duas *pools*, representando o **Utilizador 1** e o **Utilizador 2**, cada um com diferentes papéis no processo. O processo inicia-se quando o **Utilizador 1** demonstra **interesse por um produto**. O fluxo de atividades segue a seguinte sequência:

- O **Utilizador 1** procura um produto e envia uma **proposta de troca** ao **Utilizador 2**.
- O **Utilizador 2** pode:
 - **Aceitar a proposta**, finalizando a troca.
 - **Rejeitar a proposta**, encerrando a negociação.
 - **Enviar uma contraproposta**, iniciando um ciclo de negociação.
- Se o **Utilizador 1** receber uma contraproposta, pode decidir entre aceitá-la, rejeitá-la ou enviar uma nova proposta.

O processo pode resultar em dois desfechos:

- **Troca aceite**: Quando uma das partes aceita a proposta ou contraproposta.
- **Troca rejeitada**: Se uma das partes rejeitar a proposta.

10 Diagrama ER

O diagrama representa um sistema onde os utilizadores podem comprar, vender, alugar ou trocar produtos. A estrutura do modelo de dados é composta por várias tabelas interligadas, refletindo diferentes tipos de anúncio, propostas e transações.

10.1 Principais Tabelas e Relações

10.1.1 Utilizadores (users)

- Contém informações dos utilizadores, como nome, email e localização.
- Campos importantes:
 - `id` (UUID): Identificador único do utilizador.
 - `username`, `email`, `name`: Dados pessoais.
 - `location_id`: Referência à localização do utilizador.
 - `is_deleted`, `deleted_at`: Indicam se a conta foi apagada.
- Ligações:
 - `locations` (cidade e país do utilizador).
 - `blocked_users` (gestão de bloqueios entre utilizadores).
 - `favorites` (anúncios favoritos do utilizador).
 - `contacts` (dados de contacto do utilizador).
 - `user_transactions` (associação com transações).

10.1.2 Listagens de Produtos (product_listings)

- Representa os produtos anunciados pelos utilizadores.
- Campos importantes:
 - `id`: Identificador único do anúncio.
 - `title`, `description`: Informações do item.
 - `date_created`: Data de criação do anúncio.
 - `owner_id`: Utilizador que criou o anúncio.

- `category_id`: Categoria do produto.
- `listing_status`: Estado do anúncio.

- Ligações:

- `listing_statuses` (estado do anúncio: ativo, expirado, etc.).
- `product_categories` (categoria do produto).
- `product_images` (imagens associadas ao produto).
- Diferentes tipos de anúncios: `sale_listings`, `rental_listings`, `swap_listings` e `giveaway_listings`.

10.1.3 Tipos de Anúncios

- `sale_listings`: Inclui preço, condição do produto e se o preço é negociável.
- `rental_listings`: Contém dados como custo diário, depósito e limite de duração do aluguer.
- `swap_listings`: Define o item desejado para troca.
- `giveaway_listings`: Especifica requisitos para doação do item.

10.1.4 Propostas (proposals)

- Regista propostas de transação entre utilizadores.
- Campos principais:
 - `id`: Identificador único.
 - `sender_id`, `receiver_id`: Utilizadores envolvidos.
 - `listing_id`: Produto associado à proposta.
 - `proposal_type_id`: Tipo de proposta (troca, venda, aluguer, doação).
 - `proposal_status_id`: Estado da proposta.
- Ligações:
 - `proposal_types` (tipo de proposta).
 - `proposal_statuses` (estado da proposta).
 - `proposal_images` (imagens associadas às propostas).

- Tipos específicos de propostas:
 - * **sale_proposals** (inclui preço oferecido).
 - * **rental_proposals** (informações de aluguer).
 - * **swap_proposals** (itens de troca).
 - * **giveaway_proposals** (mensagem para doação).

10.1.5 Transações (transactions)

- Armazena transações concluídas na plataforma.
- Campos principais:
 - **id**: Identificador único.
 - **date_created**: Data da transação.
 - **transaction_status_id**: Estado da transação.
 - **listing_id**: Produto transacionado.
 - **proposal_id**: Proposta que originou a transação.
 - **user_id**: Utilizador envolvido.
- Ligações:
 - **transaction_statuses** (estado da transação).
 - **user_transactions** (relação entre comprador e vendedor).

10.1.6 Outras tabelas de suporte

- **locations**: Armazena informações sobre a localização do utilizador.
- **contacts**: Regista detalhes de contacto.
- **favorites**: Guarda anúncios marcados como favoritos pelos utilizadores.
- **blocked_users**: Relação entre utilizadores bloqueados.
- **product_categories**: Define as categorias dos produtos.
- **product_images**: Associa imagens aos anúncios.
- **rental_availabilities**: Especifica a disponibilidade de aluguer.

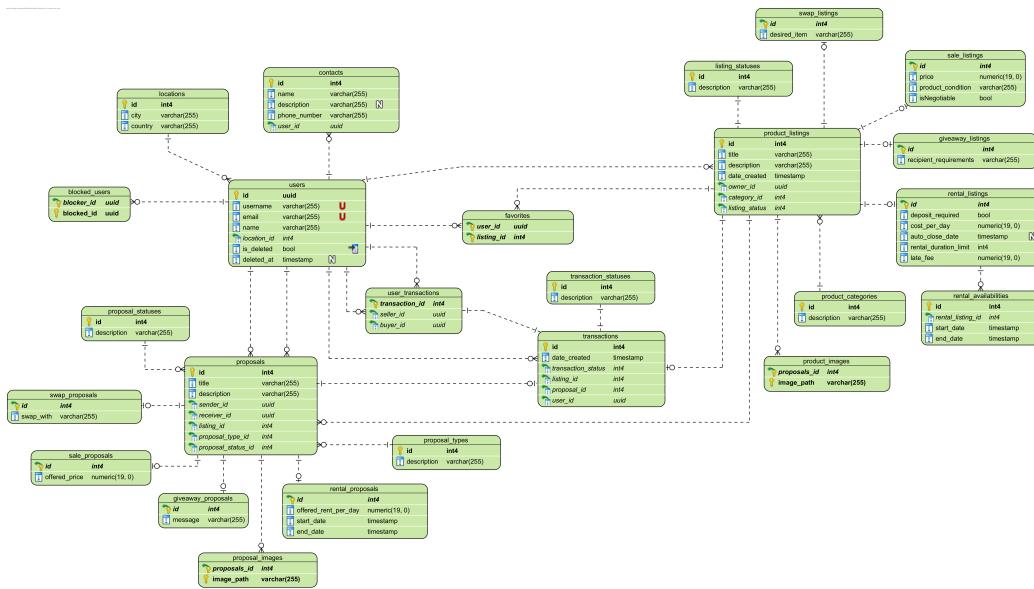


Figura 6: Diagrama ER

11 Diagrama de Classes

O diagrama de classes define a estrutura do sistema, organizando as suas entidades principais em pacotes: *User Management*, *Proposals*, *Transactions* e *Product Listings*. Cada pacote contém classes com atributos e relações bem definidas, garantindo uma arquitectura modular e escalável.

11.1 User Management

O pacote *User Management* gere as entidades relacionadas com utilizadores e as suas interacções. A classe **User** contém informações essenciais, como ID, nome, e localização. Os utilizadores podem bloquear outros utilizadores (**BlockedUser**) e adicionar anúncios aos favoritos (**Favorite**). Também existem classes auxiliares como **Location** e **Contact**, que armazenam detalhes adicionais.

11.2 Proposals

O pacote *Proposals* define os diferentes tipos de propostas que os utilizadores podem criar. A classe base **BaseProposal** contém atributos comuns a todas as propostas, como descrição, estado e utilizador associado. Tipos específicos incluem:

- **SwapProposal** - Propostas de troca de produtos.
- **SaleProposal** - Propostas de venda, com um preço proposto.
- **GiveawayProposal** - Propostas de oferta sem custo.
- **RentalProposal** - Propostas de aluguer, incluindo datas de início e fim.

A classe **ProposalStatus** define os estados possíveis de uma proposta.

11.3 Transactions

O pacote *Transactions* gere as transacções entre utilizadores. A classe **Transaction** regista as transacções com data, estado (definido pela classe **TransactionStatus**) e utilizadores envolvidos. A classe **UserTransaction** associa um utilizador a uma transacção específica.

11.4 Product Listings

O pacote *Product Listings* define os produtos disponíveis na plataforma. A classe **BaseProductListing** contém atributos comuns a todos os anúncios, como descrição, localização e categoria. Subtipos específicos incluem:

- **SaleListing** - Anúncios para venda, com preço e condição do produto.
- **SwapListing** - Anúncios de troca.
- **GiveawayListing** - Anúncios de oferta gratuita, com requisitos opcionais.
- **RentalListing** - Anúncios de aluguer, incluindo caução e preços.

As categorias de produtos são geridas pela classe **ProductCategory** e as imagens pela classe **ProductImage**. A disponibilidade para aluguer é gerida pela classe **RentalAvailability**.

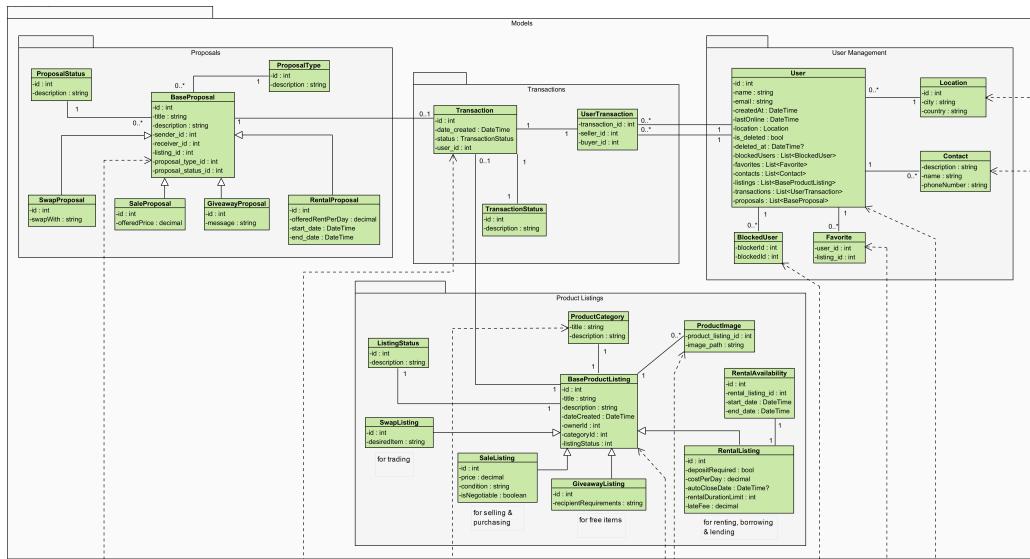


Figura 7: Diagrama de Classes - Modelos

11.5 Controladores

O sistema implementa vários controladores para gerir operações sobre as entidades definidas:

- **ProposalsController** - Gestão de propostas (criação, actualização, aceitação, rejeição).
- **TransactionsController** - Gestão de transacções (criação, actualização, cancelamento).
- **ProductListingsController** - Gestão de anúncios de produtos (criação, actualização, remoção).
- **CategoriesController** - Gestão de categorias de produtos.
- **ImagesController** - Gestão de imagens de produtos.
- **UserController** - Gestão de utilizadores.
- **BlockedUserController** - Gestão de utilizadores bloqueados.
- **FavoritesRepository** - Gestão de favoritos.
- **ContactRepository** - Gestão de contactos.
- **LocationRepository** - Gestão de localizações.

Este diagrama e organização asseguram um sistema modular, escalável e bem estruturado para gerir os diferentes aspectos da plataforma.

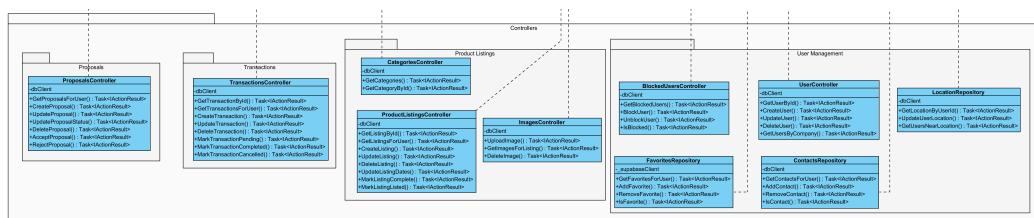


Figura 8: Diagrama de Classes - Controladores

Diagramas de Sequência

11.6 Processo de Venda

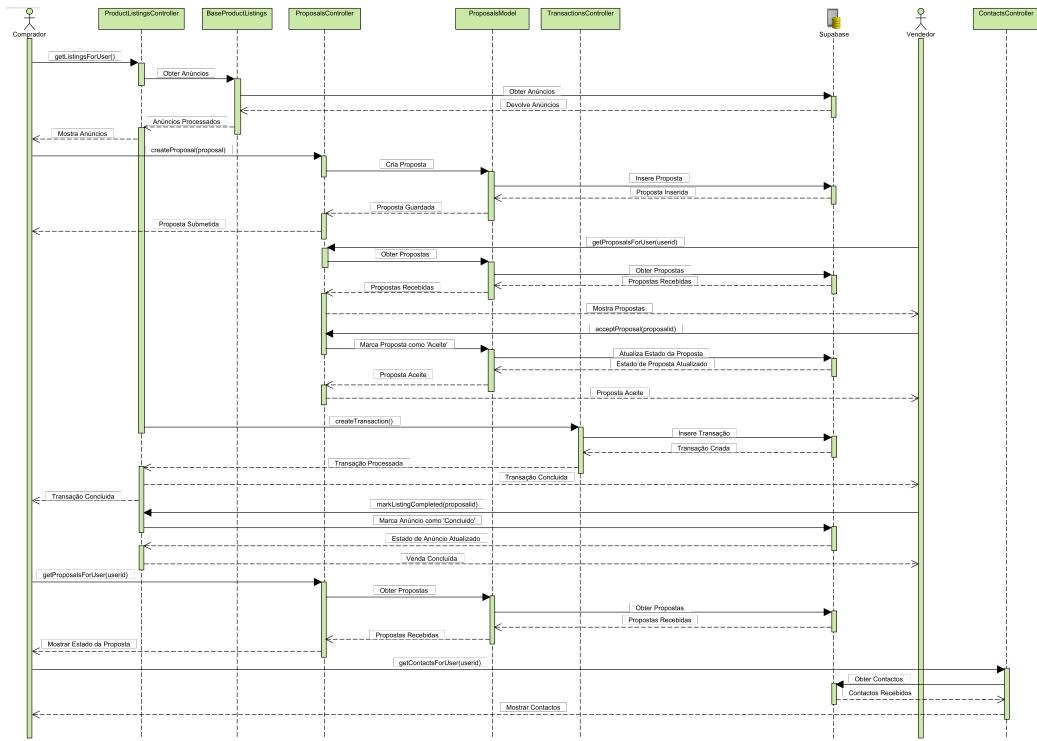


Figura 9: Diagrama de Sequência do Processo de Venda

O diagrama de sequência apresentado ilustra o fluxo de interações entre os diferentes componentes do sistema no contexto do processo de venda, descrevendo a troca de mensagens e a ordem dos eventos.

Neste cenário, o processo inicia-se quando o utilizador (potencial comprador) acede à aplicação e pesquisa os anúncios nos quais está interessado, desencadeando uma série de chamadas entre os controladores e modelos responsáveis pelo tratamento da informação. A proposta é registada e armazenada na base de dados (PostgreSQL, através do Supabase), podendo posteriormente ser consultada e processada pelo utilizador. Caso a proposta seja aceite, o sistema procede à criação da transação e à atualização do estado do anúncio, concluindo a venda.

11.7 Processo de Troca

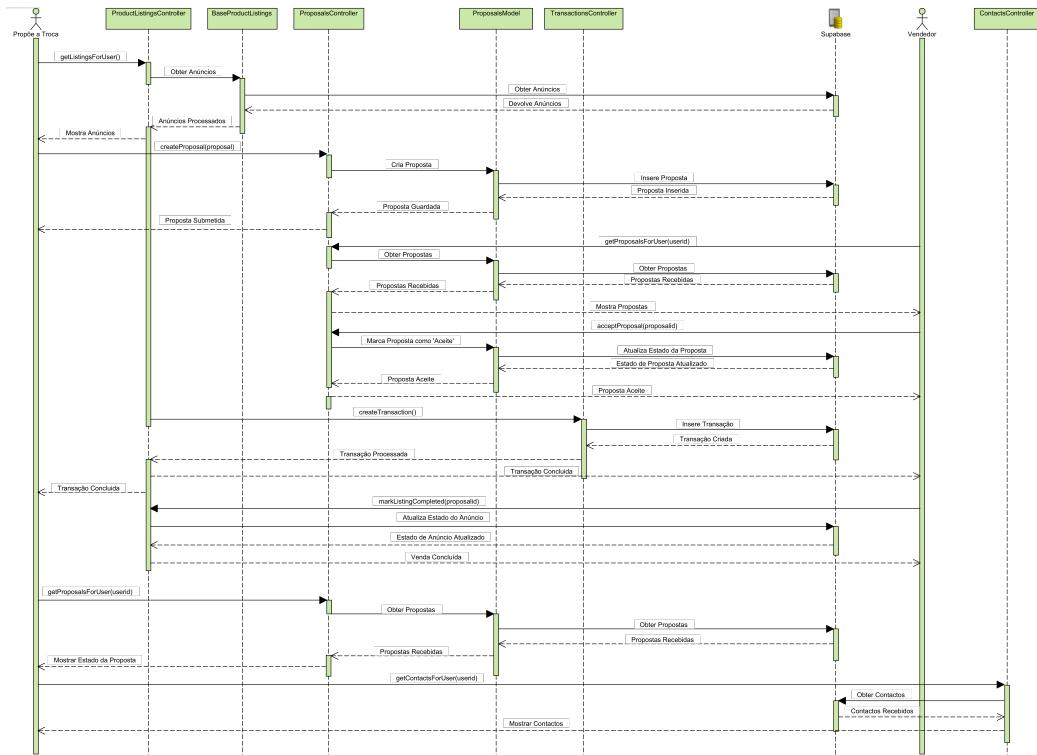


Figura 10: Diagrama de Sequência do Processo de Troca

O diagrama de sequência apresentado descreve o fluxo de interações entre os diferentes controladores e modelos envolvidos no processo de troca de produtos. Inicialmente, o utilizador visualiza os anúncios disponíveis, mostrados pelo *Product Listings Controller*. Posteriormente, o utilizador pode criar uma proposta de troca, que é submetida e guardada no sistema pelo *Proposals Controller*, que por sua vez comunica com o *Proposals Model*. As propostas recebidas podem ser visualizadas e, caso sejam aceites, o estado da proposta é atualizado. Uma vez aceite a proposta, é iniciada uma transação através do *Transactions Controller*, sendo registada e concluída a troca entre os utilizadores. Por fim, o estado do anúncio e da proposta são atualizados para refletir a conclusão da transação, e os contactos dos utilizadores envolvidos são disponibilizados para facilitar a comunicação entre ambos.

11.8 Processo de Aluguer

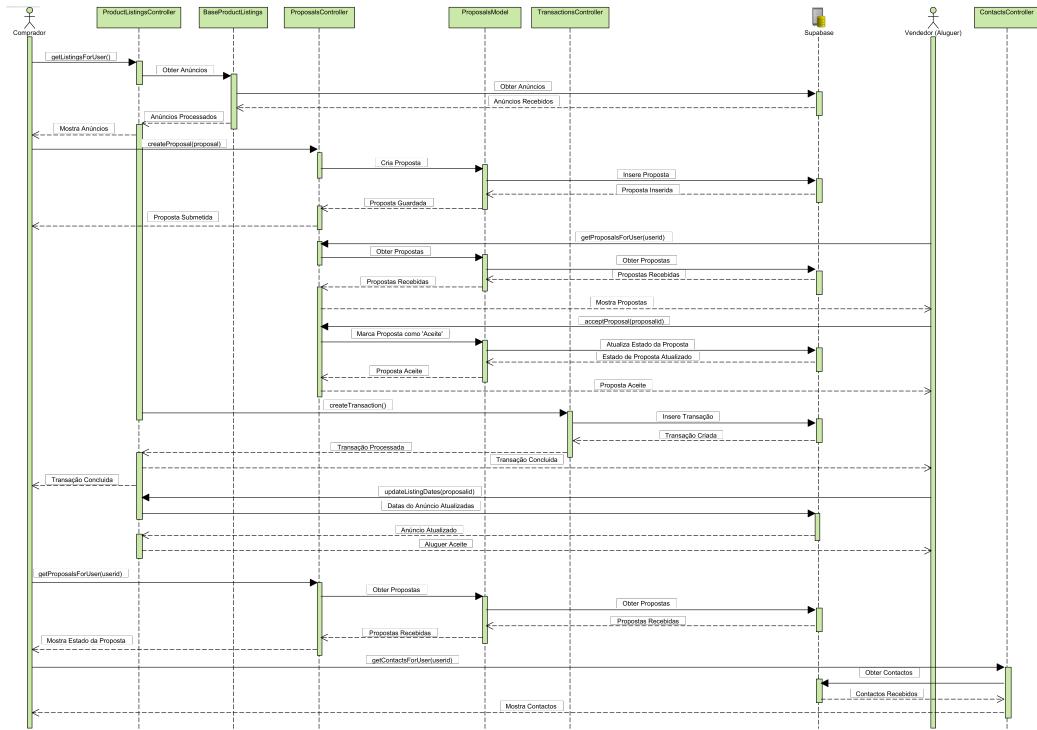


Figura 11: Diagrama de Sequência do Processo de Aluguer

O processo de aluguer inicia-se quando o utilizador vai aos anúncios disponíveis e cria uma proposta, que é submetida e armazenada na base de dados.

Após o envio da proposta, o sistema recupera as propostas recebidas e apresenta-as ao proprietário do anúncio. Caso o proprietário aceite a proposta, o estado da mesma é atualizado e uma transação é criada através do `TransactionsController`. Esta é então processada e dada como concluída.

Além disso, os dados do anúncio são atualizados para refletir a alteração no estado de disponibilidade do produto. Os utilizadores podem posteriormente visualizar o estado das suas propostas e, caso necessário, aceder aos contactos da outra parte envolvida no aluguer, disponibilizados pelo `ContactsController`.

Este processo assegura que o aluguer ocorre de forma controlada e registada, garantindo que tanto o proprietário quanto o cliente tenham acesso a informações relevantes durante toda a transação.

Diagrama de Atividades

A plataforma pr

Design UI

A plataforma pr

Plano das Sprints para a Versão Alpha

A plataforma pr

Data da sprint Beta

A plataforma pr

Conclusion

A plataforma proposta tem como objetivo criar um ambiente seguro e eficiente para a troca, venda e empréstimo de produtos dentro de uma comunidade. Ao permitir a marcação de encontros para a realização das transações, o projeto fomenta a interação entre os membros e incentiva práticas de consumo consciente. Com a definição clara dos requisitos e o suporte de diagramas UML e mockups, este relatório serve como um guia para o desenvolvimento da solução, garantindo que todos os aspectos essenciais sejam considerados.

Referências