

# 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*

7 de março de 2025

# Indice

<b>1</b>	<b>Introdução</b>	<b>4</b>
1.1	Links Úteis . . . . .	4
<b>2</b>	<b>Problema e Motivação</b>	<b>5</b>
2.1	O Problema . . . . .	5
2.2	A Solução Proposta . . . . .	5
<b>3</b>	<b>Processos de Negócio</b>	<b>6</b>
3.1	Processo de Empréstimo/Aluguer . . . . .	6
3.1.1	Etapas . . . . .	6
3.1.2	Fluxo do Processo . . . . .	6
3.1.3	Interações Essenciais . . . . .	6
3.2	Processo de Venda . . . . .	6
3.2.1	Etapas . . . . .	6
3.2.2	Fluxo do Processo . . . . .	7
3.3	Processo de Troca . . . . .	7
3.3.1	Etapas . . . . .	7
3.3.2	Fluxo do Processo . . . . .	8
3.4	Fluxo Geral . . . . .	8
<b>4</b>	<b>Requisitos Funcionais</b>	<b>9</b>
<b>5</b>	<b>Requisitos Não Funcionais</b>	<b>10</b>
<b>6</b>	<b>Arquitetura</b>	<b>11</b>
6.1	Frontend . . . . .	11
6.2	Backend . . . . .	11
6.3	Base de Dados . . . . .	12
6.4	Fluxo de Funcionamento . . . . .	12
<b>7</b>	<b>User Stories</b>	<b>13</b>
<b>8</b>	<b>Diagrama de casos de Uso</b>	<b>14</b>
8.1	Utilizador Não Autenticado . . . . .	14
8.2	Utilizador Autenticados . . . . .	15

<b>9 Diagramas BPMN</b>	<b>16</b>
9.1 Diagrama BPMN das vendas . . . . .	16
9.2 Diagrama BPMN dos empréstimos/alugueres . . . . .	17
9.3 Diagrama BPMN das trocas . . . . .	19
<b>10 Diagrama ER</b>	<b>20</b>
10.1 Utilizadores (users) . . . . .	20
10.2 Anúncios de Produtos (product_listings) . . . . .	21
10.3 Propostas (proposals) . . . . .	22
10.4 Transações (transactions) . . . . .	23
<b>11 Diagrama de Classes</b>	<b>24</b>
11.1 User Management . . . . .	24
11.2 Proposals . . . . .	25
11.3 Transactions . . . . .	26
11.4 Product Listings . . . . .	27
11.5 Controladores . . . . .	28
11.6 Processo de Venda . . . . .	29
11.7 Processo de Troca . . . . .	30
11.8 Processo de Aluguer . . . . .	31
<b>12 Diagrama de Atividades</b>	<b>32</b>
12.1 Processo de Troca . . . . .	32
12.2 Processo de Venda . . . . .	34
12.3 Processo de Aluguer . . . . .	36
<b>13 Design UI</b>	<b>38</b>
<b>14 Milestones</b>	<b>39</b>
14.1 Planeamento e Entregas . . . . .	39
<b>15 Conclusão</b>	<b>40</b>

## **Lista de Figuras**

1	Arquitetura da Plataforma . . . . .	11
2	Diagrama de casos de uso . . . . .	14
3	Diagrama BPMN das vendas . . . . .	16
4	Diagrama BPMN dos empréstimos/alugueres . . . . .	17
5	Diagrama BPMN das trocas . . . . .	19
6	Diagrama ER - Utilizadores . . . . .	20
7	Diagrama ER - Anúncios . . . . .	21
8	Diagrama ER - Propostas . . . . .	22
9	Diagrama ER - Transações . . . . .	23
10	Diagrama de Classes - Utilizadores . . . . .	24
11	Diagrama de Classes - Propostas . . . . .	25
12	Diagrama de Classes - Transações . . . . .	26
13	Diagrama de Classes - Anúncios . . . . .	27
14	Diagrama de Classes - Controladores . . . . .	28
15	Diagrama de Sequência do Processo de Venda . . . . .	29
16	Diagrama de Sequência do Processo de Troca . . . . .	30
17	Diagrama de Sequência do Processo de Aluguer . . . . .	31
18	Diagrama de atividades do Processo de Troca . . . . .	33
19	Diagrama de atividades do Processo de Vendas . . . . .	35
20	Diagrama de atividades do Processo de Aluguer . . . . .	37
21	Mockup da aplicação . . . . .	38

# 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*.

## 1.1 Links Úteis

Os seguintes *links* contêm informações relevantes sobre o projeto:

- **Product Backlog:**

*Excel*

*Jira*

- **Código-fonte:**

*Repositório no GitHub*

## 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
<b>RF 01</b>	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.
<b>RF 02</b>	O utilizador pode definir datas bloqueadas para empréstimo/aluguer do produto.
<b>RF 03</b>	O sistema deve verificar e confirmar as datas livres para empréstimo/aluguer.
<b>RF 04</b>	O utilizador pode definir um valor para aluguer ou venda.
<b>RF 05</b>	O utilizador pode permitir ou não propostas nos anúncios.
<b>RF 06</b>	O utilizador pode aceitar, rejeitar ou negociar propostas.
<b>RF 07</b>	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.
<b>RF 08</b>	O utilizador, ao criar um anúncio para troca, pode indicar qual produto deseja receber em troca.
<b>RF 09</b>	A aplicação deve permitir a pesquisa de produtos disponíveis na plataforma.
<b>RF 10</b>	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.

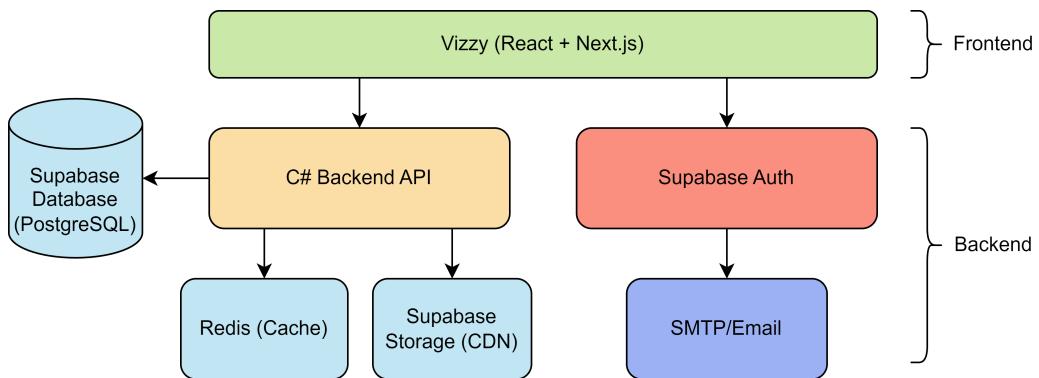


Figura 1: Arquitetura da Plataforma

### 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*

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

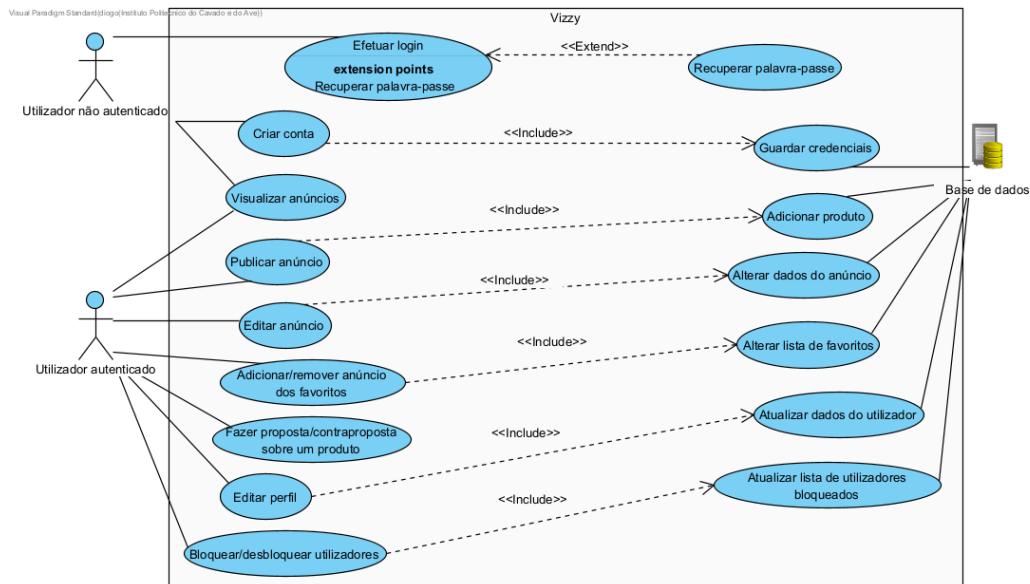


Figura 2: Diagrama de casos de uso

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

### 8.1 Utilizador Não Autenticado

Os utilizadores que ainda não estão autenticados 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 autenticados consultem os anúncios disponíveis.

## 8.2 Utilizador Autenticados

Após a 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 autenticados.
- **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

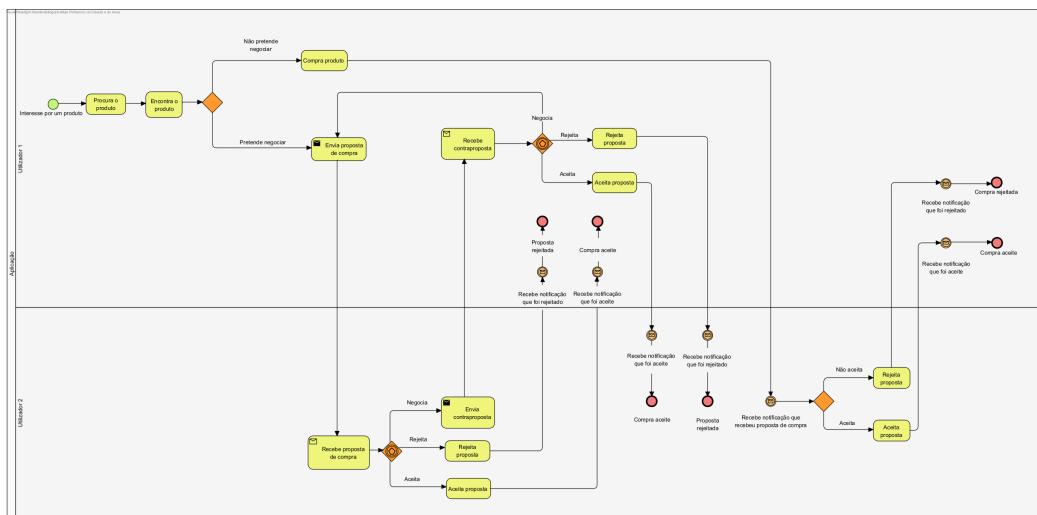


Figura 3: Diagrama BPMN das vendas

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

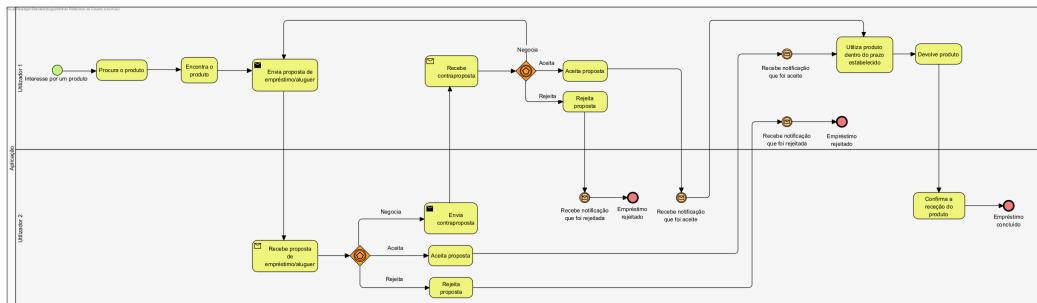


Figura 4: Diagrama BPMN dos empréstimos/alugueres

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

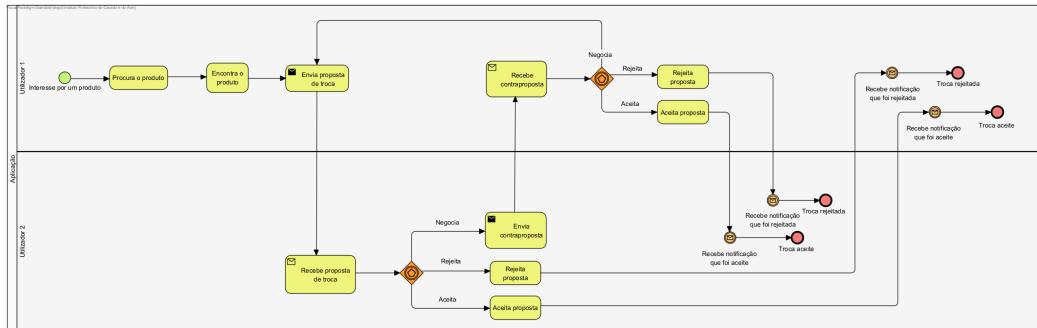


Figura 5: Diagrama BPMN das trocas

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 Utilizadores (users)

Os utilizadores terão como dados um *username*, um email e o seu nome. Para além disso terão números de contacto e uma localização associados e também uma lista de utilizadores bloqueados e de anúncios favoritos.

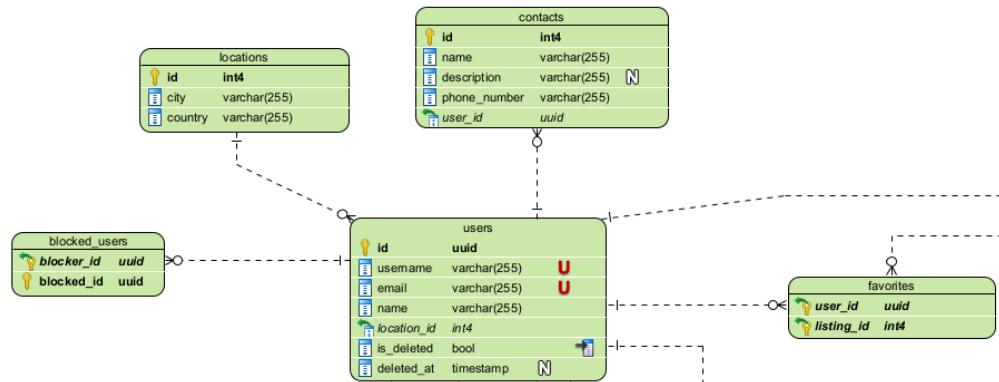


Figura 6: Diagrama ER - Utilizadores

## 10.2 Anúncios de Produtos (product\_listings)

A tabela de *product\_listing* terá os atributos que serão comuns a todos os tipos de anúncios como o título, a descrição e a data que foi criado, mas depois irão existir três tipo de produtos:

- **sale\_listing** - Anúncios para venda, com preço e condição do produto.
- **swap\_listing** - Anúncios de troca.
- **giveaway\_listing** - Anúncios de oferta gratuita, com requisitos opcionais.
- **rental\_listing** - Anúncios de aluguer, incluindo caução e preços.

Num anúncio também poderão estar associados imagens do produto e a sua categoria. Para além disso também teremos o estado do anúncio.

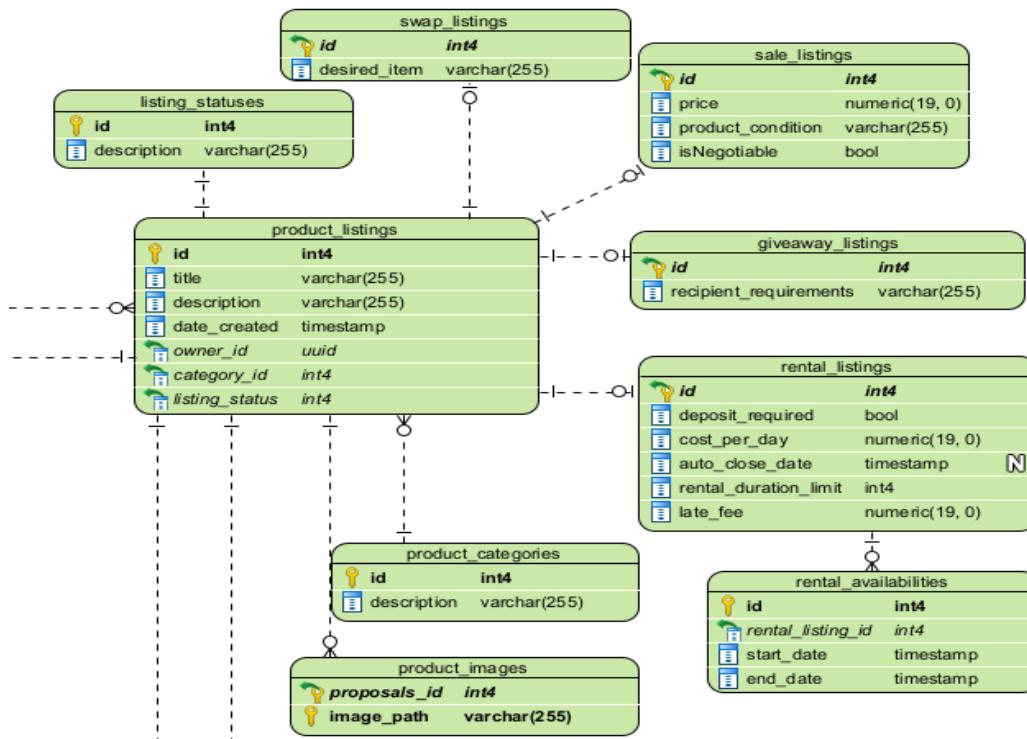


Figura 7: Diagrama ER - Anúncios

## 10.3 Propostas (proposals)

A tabela de *proposals* contém mais uma vez os atributos que são comuns a todo o tipo de propostas que são o título e a descrição. Depois teremos tabelas para cada tipo de proposta que são:

- **swap\_proposal** - Propostas de troca de produtos.
- **sale\_proposal** - Propostas de venda, com um preço proposto.
- **giveaway\_proposal** - Propostas de oferta sem custo.
- **rental\_proposal** - Propostas de aluguer, incluindo datas de início e fim.

Para além disso também teremos uma tabela com o estado da proposta

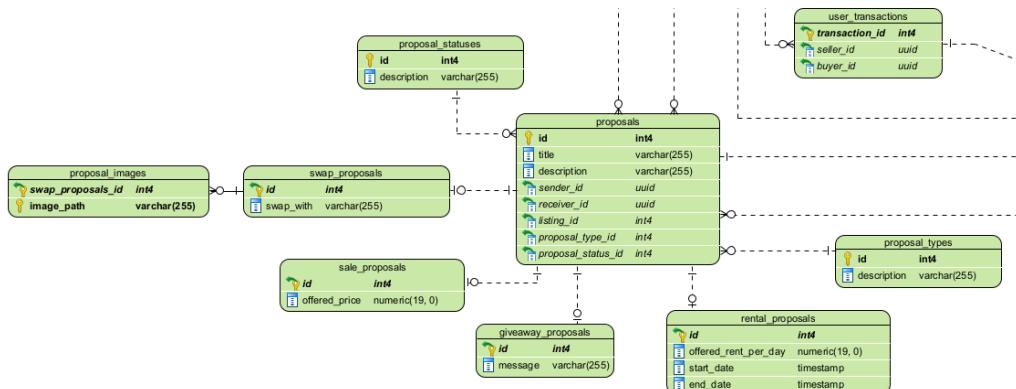


Figura 8: Diagrama ER - Propostas

## 10.4 Transações (transactions)

Na tabela *transactions* temos a data que essa transação foi criada e uma associação a uma tabela com o estado dessa mesma transação. Para além disso também temos uma tabela *user\_transactions* que terá como função servir de tabela intermediária que terá o **id** do comprador e do vendedor.

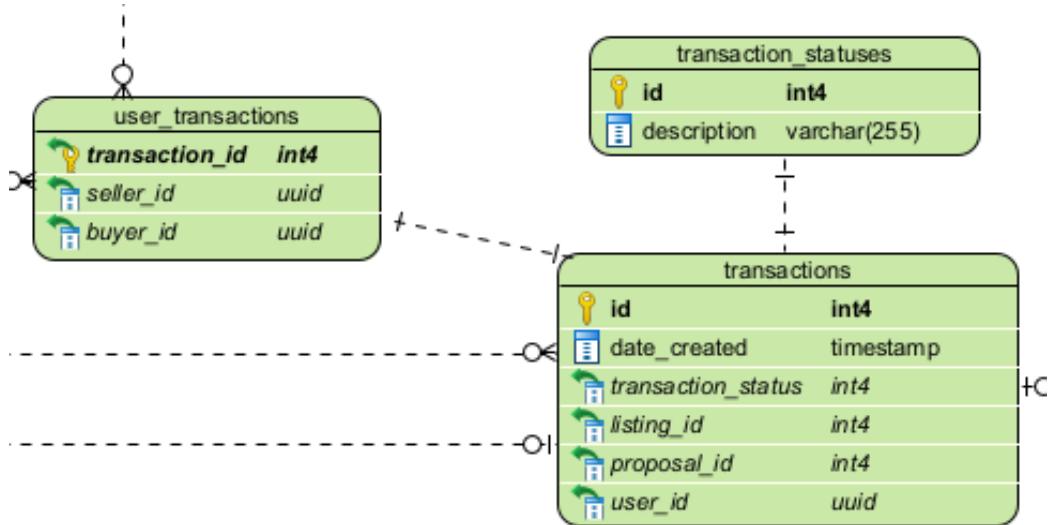


Figura 9: Diagrama ER - Transações

# 11 Diagrama de Classes

A API backend foi estruturada com modelos e controladores, organizados em pacotes como *User Management*, *Proposals*, *Transactions* e *Product Listings*. Cada pacote define classes com atributos e relações, que garante modularidade e escalabilidade.

## 11.1 User Management

O pacote *User Management* gera as entidades relacionadas com utilizadores e as suas interações. A classe **User** armazena diversas informações, como ID, nome, localização, contactos e outros dados relevantes. Os utilizadores podem bloquear outros utilizadores (**BlockedUser**) e adicionar anúncios aos favoritos (**Favorite**).

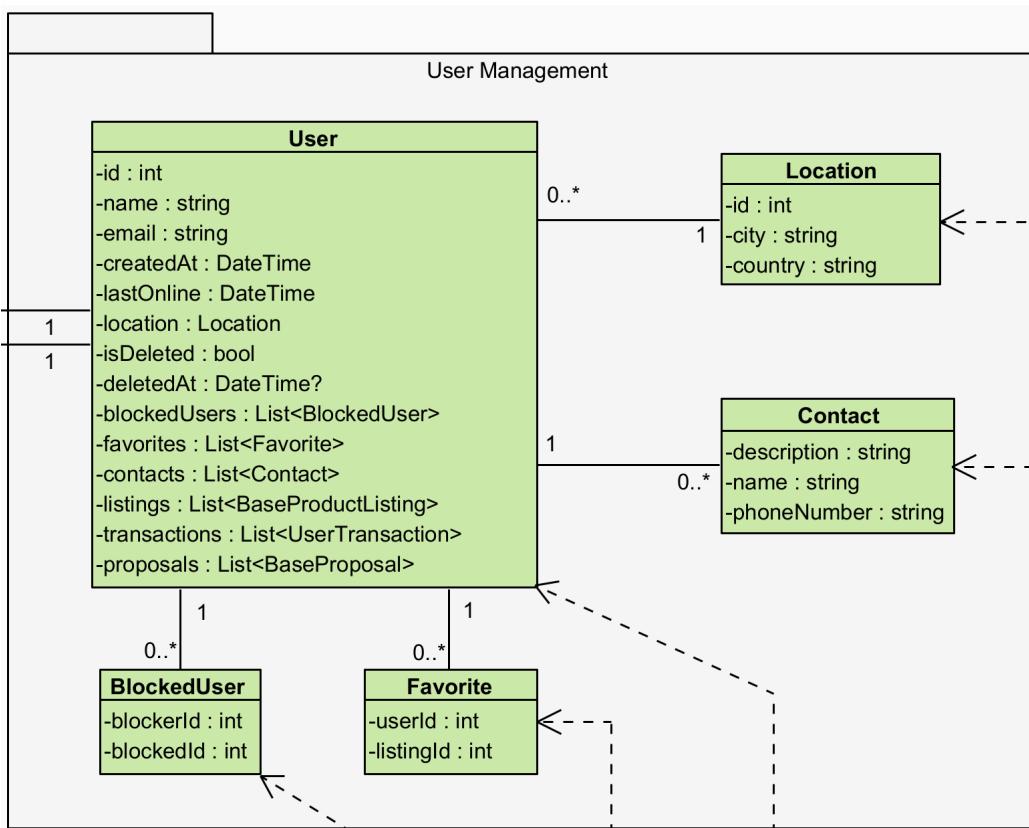


Figura 10: Diagrama de Classes - Utilizadores

## 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** guarda o estado de uma proposta.

Propostas do tipo **SwapProposal** (troca) têm imagens associadas.

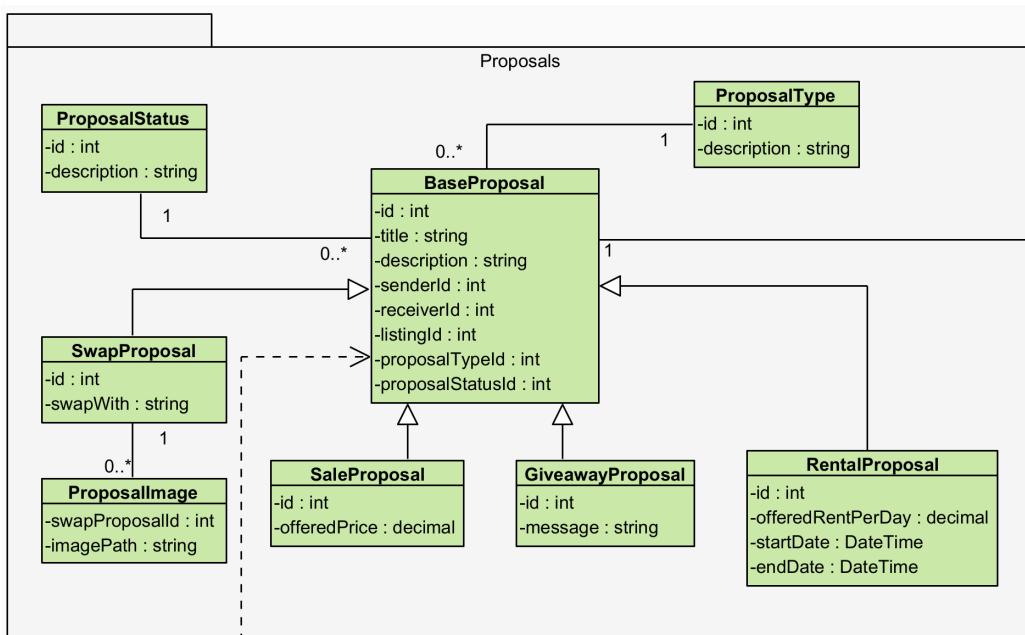


Figura 11: Diagrama de Classes - Propostas

## 11.3 Transactions

O pacote *Transactions* é responsável pela gestão das transações entre utilizadores. A classe *Transaction* representa uma transação e guarda a sua data de criação, estado (referenciado pela classe *TransactionStatus*) e o utilizador associado.

A classe **TransactionStatus** define os estados possíveis de uma transação, com um identificador único e uma descrição textual do estado.

A classe **UserTransaction** estabelece a relação entre utilizadores e transações, identificando explicitamente o vendedor e o comprador envolvidos numa transação específica. Cada transação pode estar associada a vários registo de *UserTransaction*.

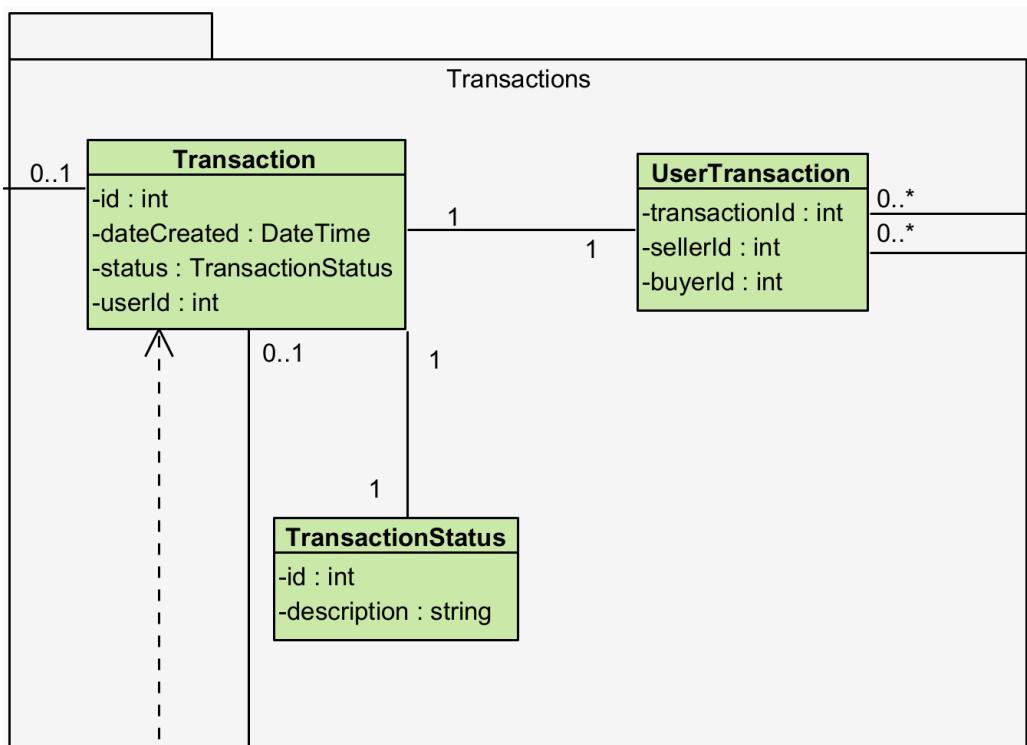


Figura 12: Diagrama de Classes - Transações

## 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**.

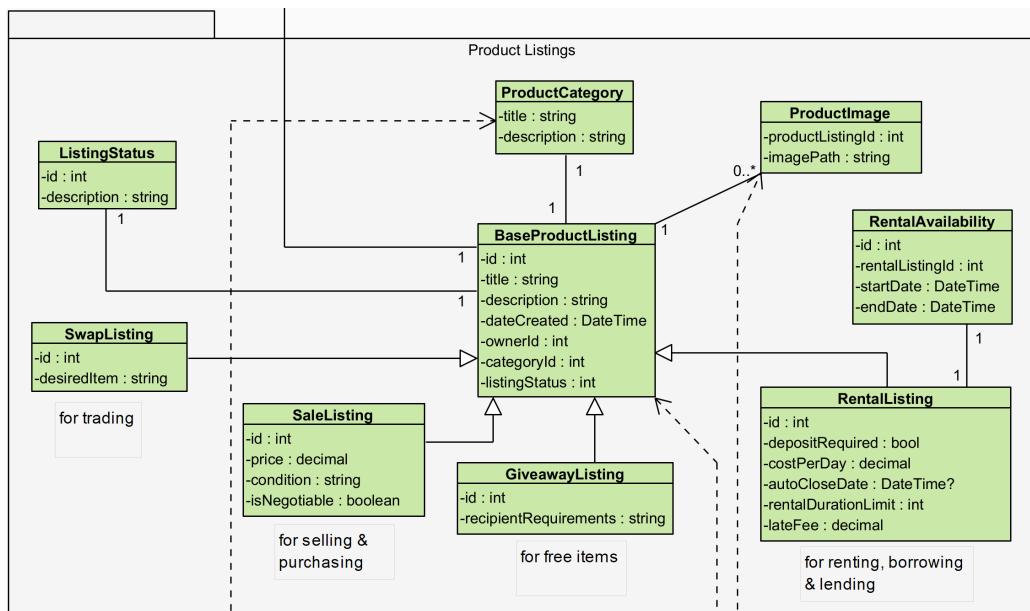


Figura 13: Diagrama de Classes - Anúncios

## 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.

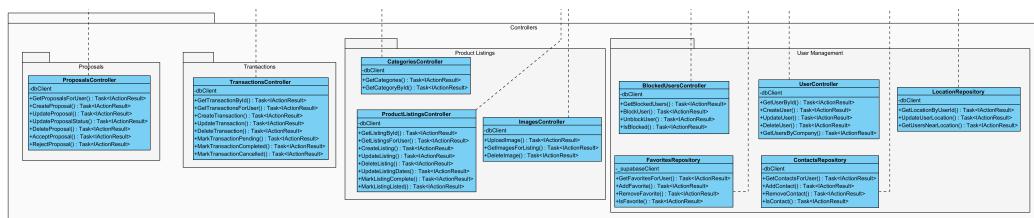


Figura 14: Diagrama de Classes - Controladores

# Diagramas de Sequência

## 11.6 Processo de Venda

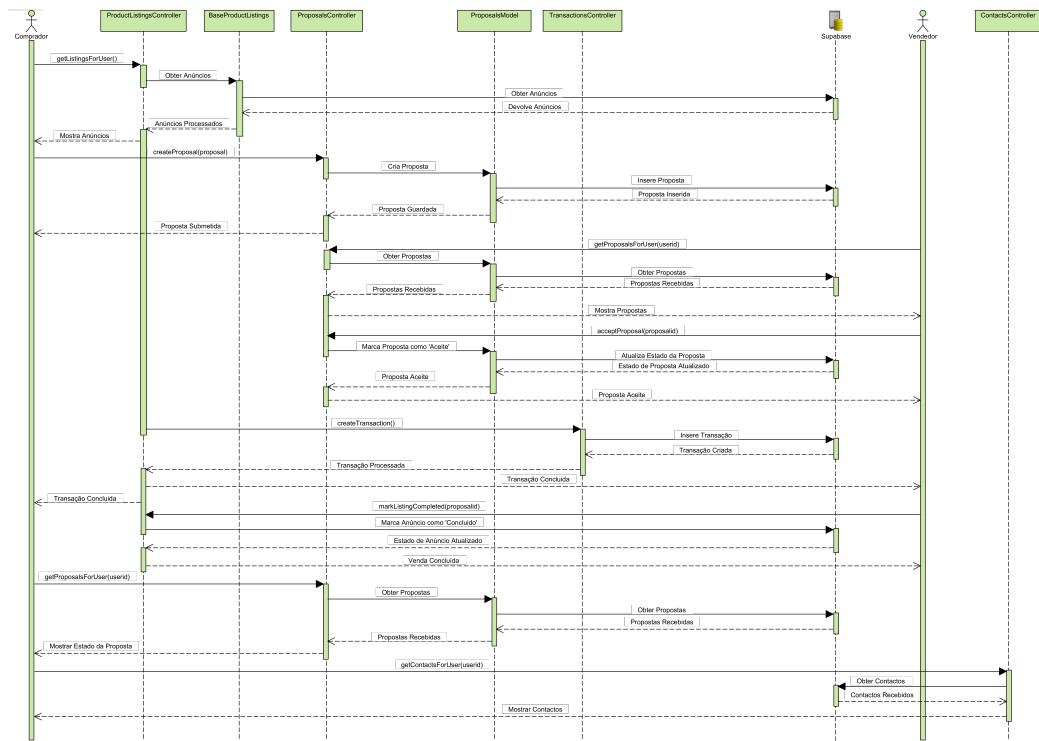


Figura 15: Diagrama de Sequência do Processo de Venda

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

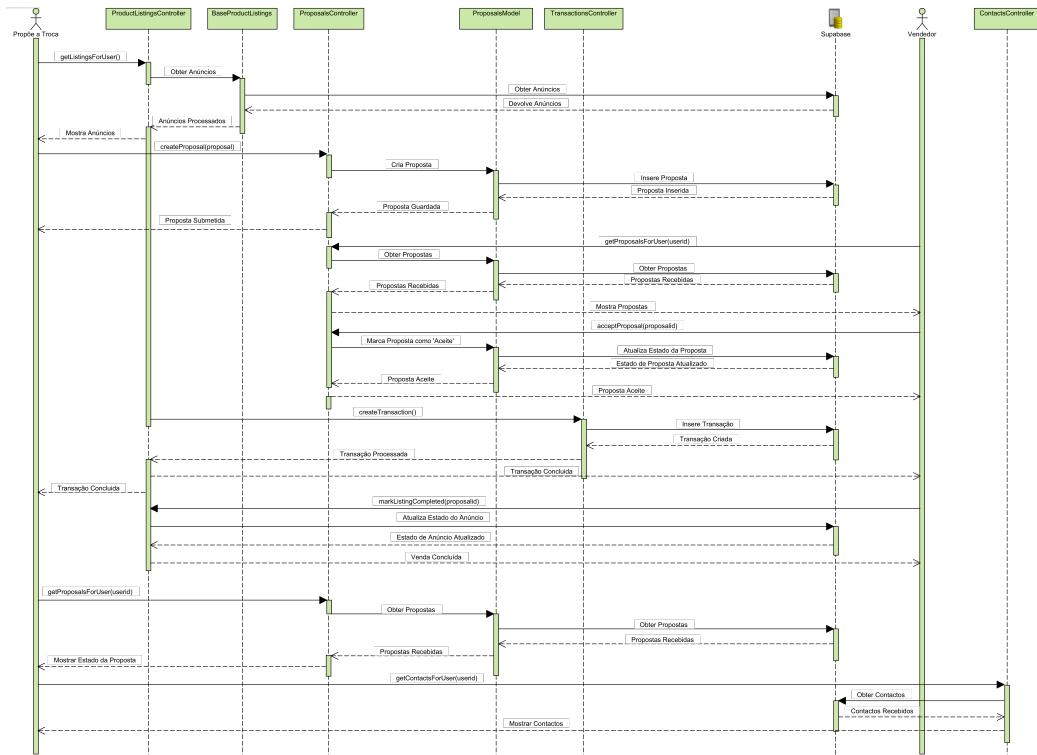


Figura 16: Diagrama de Sequência do Processo de Troca

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 *ProductListingsController*. Posteriormente, o utilizador pode criar uma proposta de troca, que é submetida e guardada no sistema pelo *ProposalsController*, que por sua vez comunica com o *ProposalsModel*. 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 *TransactionsController*, 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

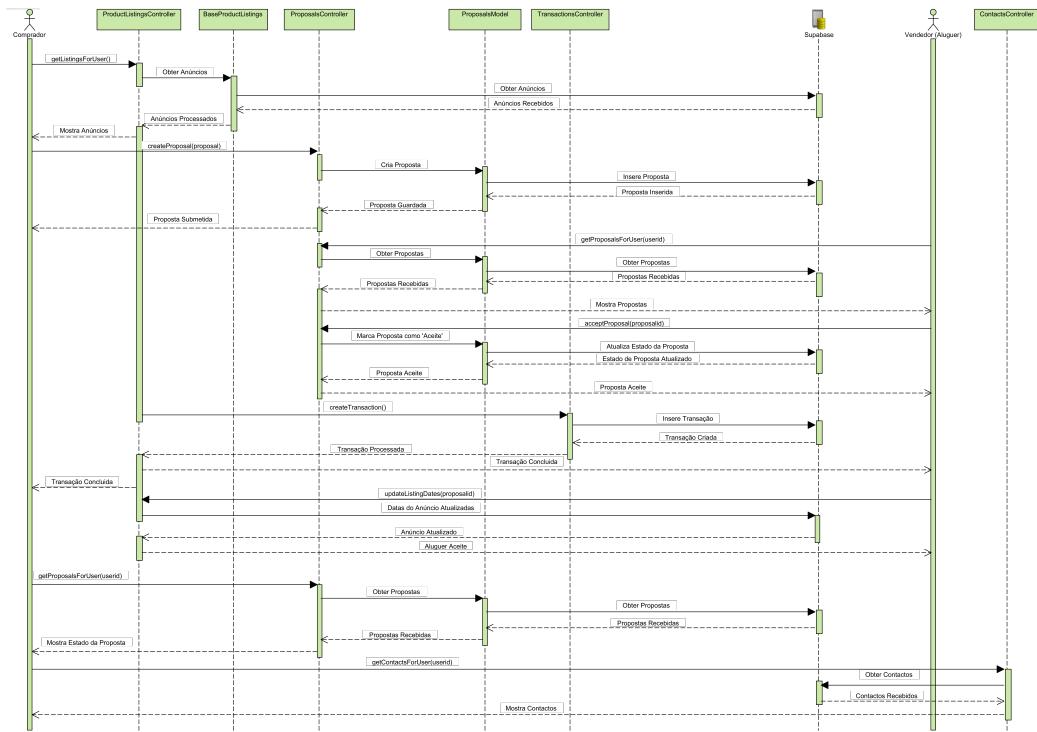


Figura 17: Diagrama de Sequência do Processo de Aluguer

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.

## 12 Diagrama de Atividades

Os diagramas de atividade representam o fluxo de operações dentro de um sistema, destacando as ações realizadas e a interação entre os participantes. Eles são utilizados para modelar processos, ilustrando a sequência de atividades, decisões e possíveis caminhos alternativos. No contexto deste sistema, os diagramas de atividade descrevem os processos de vendas, aluguer e trocas, facilitando a compreensão do funcionamento e das etapas envolvidas.

### 12.1 Processo de Troca

O processo de troca inicia-se quando um cliente está interessado num item e propõe uma troca ao seu dono. O fluxo segue da seguinte forma:

- O cliente, interessado num item, envia uma proposta de troca ao dono do item.
- O dono recebe a proposta e pode:
  - Aceitar a proposta, concluindo a troca.
  - Rejeitar a proposta, encerrando o processo.
  - Criar uma contraproposta se tomou a decisão de negociar.
- Se o dono criar uma contraproposta:
  - O cliente pode aceitar a contraproposta, realizando a troca.
  - O cliente pode rejeitar a contraproposta, encerrando o processo.
  - O cliente pode optar por continuar a negociar, enviando uma nova proposta.
- O processo continua até que uma proposta seja aceite por ambas as partes ou até que uma das partes decida encerrar a negociação.

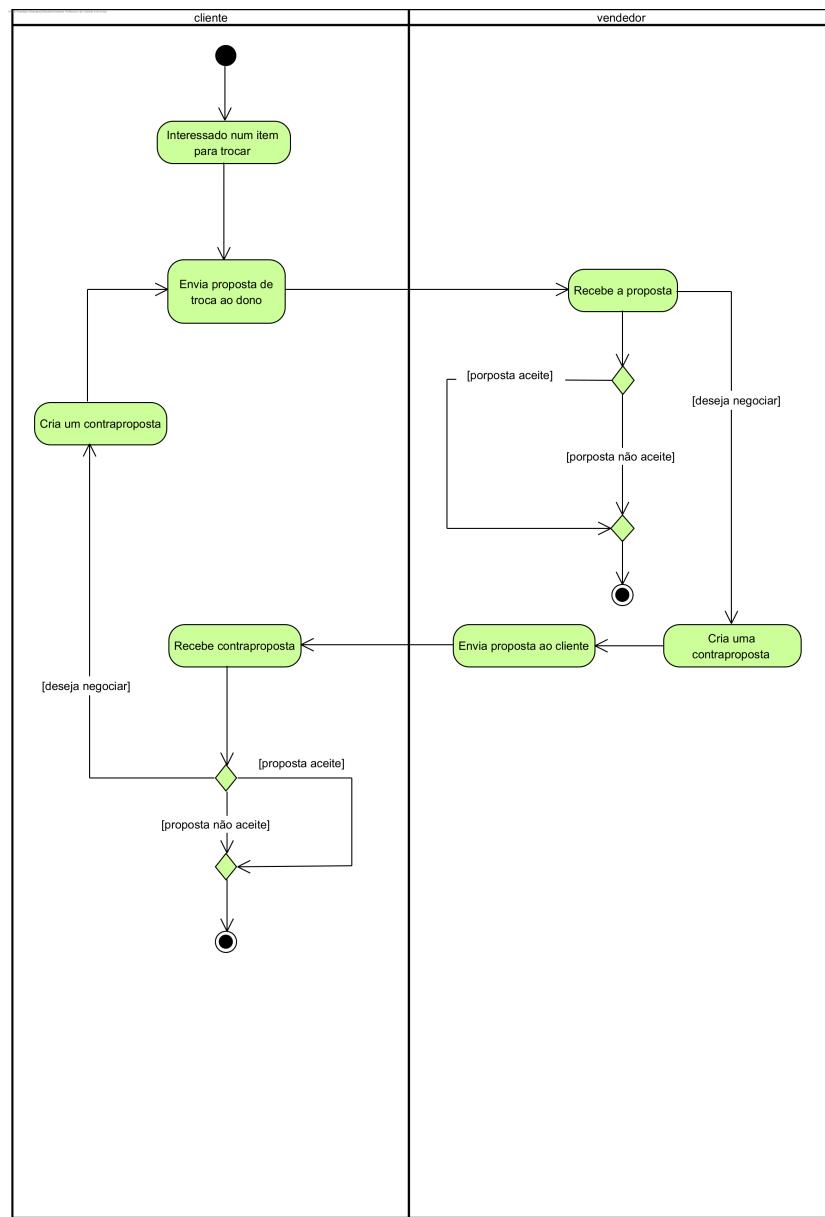


Figura 18: Diagrama de atividades do Processo de Troca

## 12.2 Processo de Venda

O processo de venda inicia-se quando um cliente está interessado em comprar um item. O fluxo segue da seguinte forma:

- O cliente, interessado em comprar um item, verifica se o item é negociável.
  - Se o item não for negociável, o cliente pode aceitar as condições de venda ou desistir da compra.
  - Se o item for negociável, o cliente pode enviar uma proposta de compra ao vendedor.
- O vendedor recebe a proposta e pode:
  - Aceitar a proposta, concluindo a venda.
  - Rejeitar a proposta, encerrando o processo.
  - Criar uma contraproposta para continuar a negociação.
- Se o vendedor criar uma contraproposta:
  - O cliente pode aceitar a contraproposta, concluindo a venda.
  - O cliente pode rejeitar a contraproposta, encerrando o processo.
  - O cliente pode optar por continuar a negociar, enviando uma nova proposta.
- O processo de negociação continua até que:
  - Ambas as partes cheguem a um acordo, concluindo a venda.
  - Uma das partes decida encerrar a negociação.
- Caso um acordo seja alcançado, a venda é concluída e o item é entregue ao cliente.

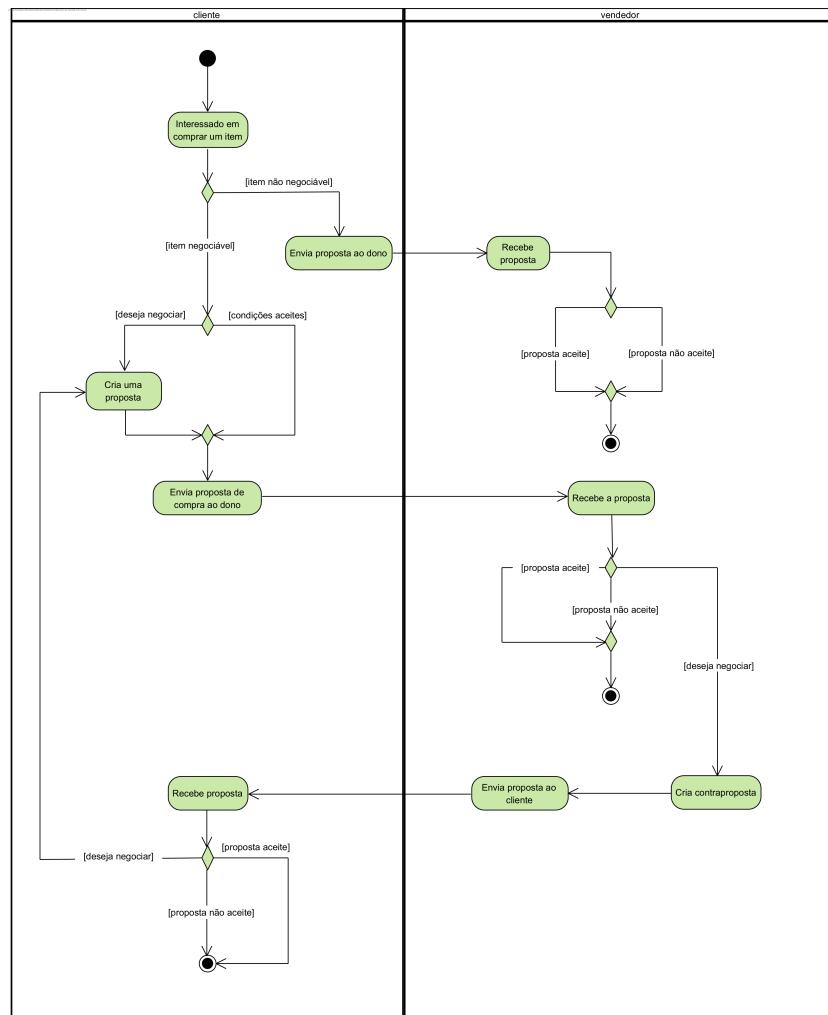


Figura 19: Diagrama de atividades do Processo de Vendas

## 12.3 Processo de Aluguer

O processo de aluguer inicia-se quando um cliente está interessado em alugar um item. O fluxo segue da seguinte forma:

- O cliente, interessado em alugar um item, verifica as condições de aluguer.
  - Se as condições forem aceites, o cliente pode prosseguir com o aluguer.
  - Se o cliente desejar negociar, pode criar uma contraproposta e enviá-la ao dono do item.
- O dono do item recebe a proposta e pode:
  - Aceitar a proposta, concluindo o acordo de aluguer.
  - Rejeitar a proposta, encerrando o processo.
  - Criar uma contraproposta para continuar a negociação.
- Se o dono criar uma contraproposta:
  - O cliente pode aceitar a contraproposta, concluindo o aluguer.
  - O cliente pode rejeitar a contraproposta, encerrando o processo.
  - O cliente pode optar por continuar a negociar, enviando uma nova proposta.
- O processo de negociação continua até que:
  - Ambas as partes cheguem a um acordo, concluindo o aluguer.
  - Uma das partes decida encerrar a negociação.
- Caso um acordo seja alcançado, o aluguer é registado no sistema.

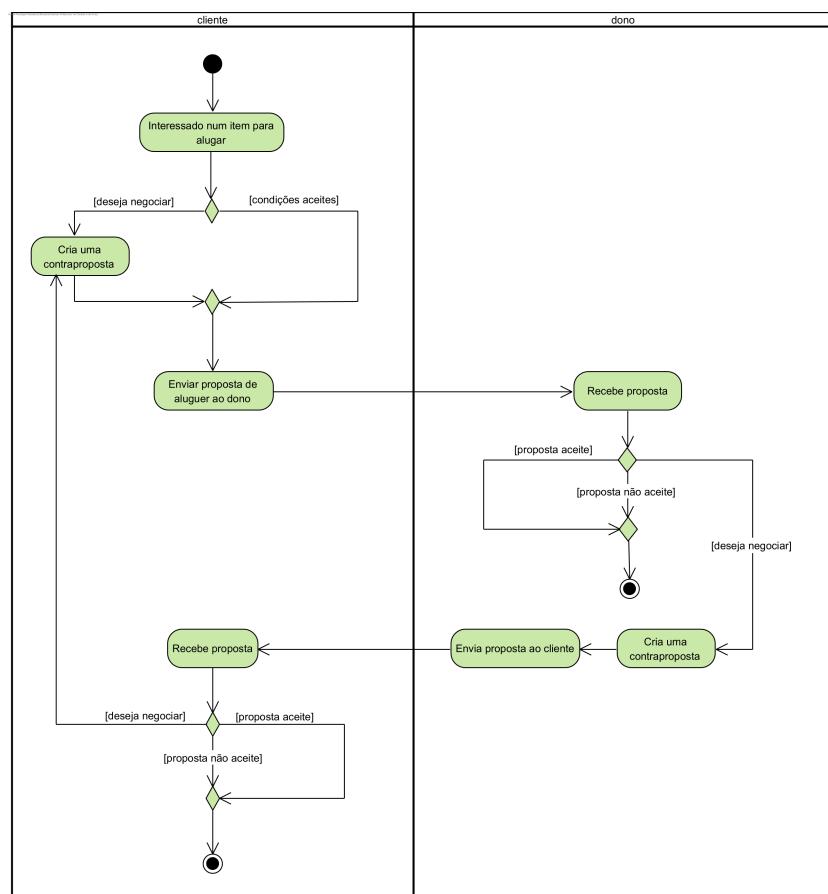


Figura 20: Diagrama de atividades do Processo de Aluguer

## 13 Design UI

A interface do utilizador (UI) da plataforma foi concebida para proporcionar uma experiência visual agradável e coerente, garantindo uma navegação intuitiva e acessível. Os mockups desenvolvidos ilustram o design das páginas, incluindo a organização dos elementos, a paleta de cores, a tipografia e os componentes interativos.

Para consulta detalhada dos mockups e da estrutura visual da plataforma, estes estão disponíveis numa pasta anexada a este relatório.

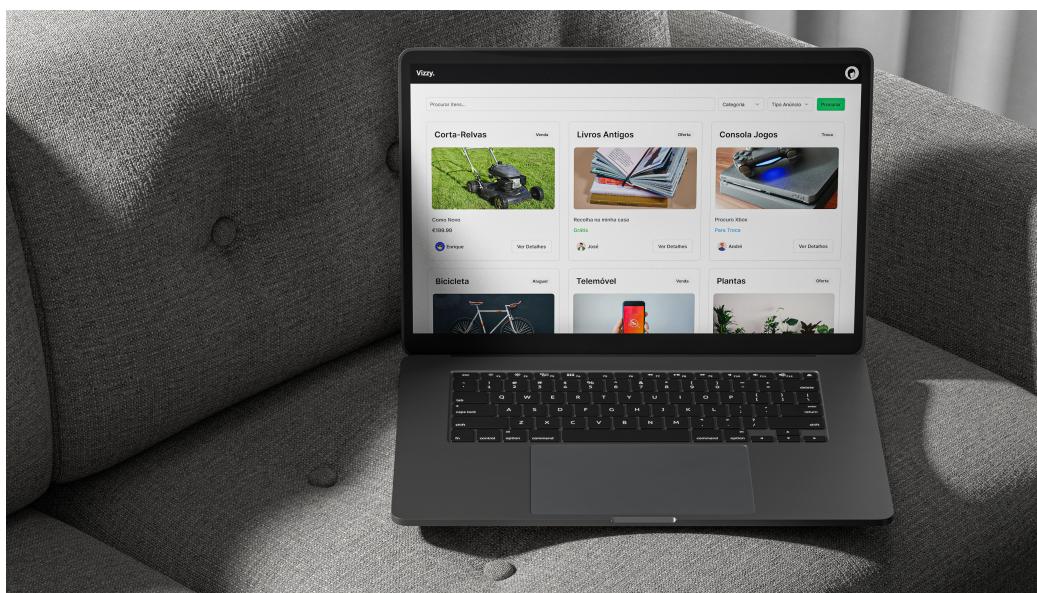


Figura 21: Mockup da aplicação

## 14 *Milestones*

O desenvolvimento da plataforma seguirá uma abordagem faseada, garantindo um planeamento estruturado e entregas progressivas ao longo do processo. Para assegurar um acompanhamento claro do progresso, foram definidos marcos essenciais que representam as principais fases do projeto.

### 14.1 Planeamento e Entregas

As etapas-chave do desenvolvimento e as respetivas datas de entrega são as seguintes:

- **Entrega da Especificação** (*9 de março de 2025*) – Nesta fase, será elaborado e entregue o documento de especificação do projeto, contendo a definição dos requisitos funcionais e não funcionais, bem como a arquitetura do sistema e os principais diagramas UML.
- **Fase Alpha** (*6 de abril de 2025*) – Esta versão irá corresponder ao primeiro protótipo funcional da aplicação. O foco está em seguir a arquitetura e integrar os principais componentes.
- **Versão Beta** (*20 de abril de 2025*) – Com base no feedback da fase Alpha, esta versão tem como objetivo incorporar melhorias gerais da aplicação. Além disso, serão realizados testes mais aprofundados, abrangendo diferentes cenários de utilização para garantir estabilidade e usabilidade.
- **Versão RTW (Ready to Web)** (*11 de maio de 2025*) – Esta versão representará a aplicação finalizada, pronta para disponibilização ao público. Nesta fase, serão concluídas as últimas correções, ajustes de funcionalidades e serão realizadas as validações finais para assegurar a conformidade com os objetivos do projeto.

A definição destes marcos permitirá uma gestão eficiente do desenvolvimento, garantindo que cada etapa seja devidamente planeada, implementada e validada antes do avanço para a fase seguinte.

## **15 Conclusão**

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

- [1] Geeks for Geeks. *Redis Cache*. Acessado em 07 Mar. 2025. 2025. URL: <https://www.geeksforgeeks.org/redis-cache/>.
- [2] Next.js Documentation. *NextJS Caching*. Acessado em 07 Mar. 2025. 2025. URL: <https://nextjs.org/docs/app/building-your-application/caching>.
- [3] Next.js Documentation. *Server-Side Rendering (SSR)*. Acessado em 07 Mar. 2025. 2025. URL: <https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering>.
- [4] Next.js Documentation. *Static Site Generation (SSG)*. Acessado em 07 Mar. 2025. 2025. URL: <https://nextjs.org/docs/pages/building-your-application/rendering/static-site-generation>.
- [5] Redis Documentation. *Redis Caching for Microservices*. Acessado em 07 Mar. 2025. 2025. URL: <https://redis.io/learn/howtos/solutions/microservices/caching>.
- [6] Shiiyan. *Active Record Pattern vs Repository Pattern: Making the Right Choice*. Acessado em 07 Mar. 2025. 2025. URL: <https://medium.com/@shiiyan/active-record-pattern-vs-repository-pattern-making-the-right-choice-f36d8deece94>.
- [7] Supabase Documentation. *JWT Authentication in Supabase*. Acessado em 07 Mar. 2025. 2025. URL: <https://supabase.com/docs/guides/auth/jwts>.