Petshow:

Projeto de sistema de *backoffice* para gerenciamento de uma loja de roupas do segmento *Pet*

Amanda Locatelli, Gerson da Silva Correia, Jordana Azman Buranello, Lauro Mendes do Amaral Júnior, Maíra Martins Esteves, Otoniel de Lima Filho, Thaisa Soares Brito[[1]](#footnote-1)

Orientador: Professor Victor Williams Stafusa da Silva

Faculdade Impacta de Tecnologia

São Paulo, SP, Brasil

10 de junho de 2021

**Resumo.** O nome do projeto é *DEVs in Flask* e consiste em desenvolver um sistema para facilitar e organizar as rotinas de trabalho de uma loja de vendas de roupas para Pets, chamada *PetShow*, oferecendo um sistema de *Backoffice* para controle de suas operações.

Esse sistema consiste em um controle dos pedidos recebidos, armazenamento de informações do cliente como nome, endereço, e-mail, telefone, nome do pet, espécie, raça, porte, espécie e gênero. Além da gestão dos dados de venda com extração de relatórios mensais/anuais de vendas, lucro, e relatórios de estoque.

**Palavras-chaves:** Projeto, *Petshow*, *Backoffice*, Controle de Operações, Relatórios.

# Introdução

Este trabalho visa entregar um produto técnico para um cliente de loja de roupas para Pets.

As definições do escopo do trabalho serão elencadas abaixo, assim como os requisitos de sistema, e especificações do produto final.

## Apresentação da Empresa

Para a elaboração do trabalho foi adotada a empresa fictícia *Petshow* – Seu *Pet* com Estilo, loja voltada para vendas de roupas para animais domésticos, com opções para cachorros e gatos de vários tamanhos.

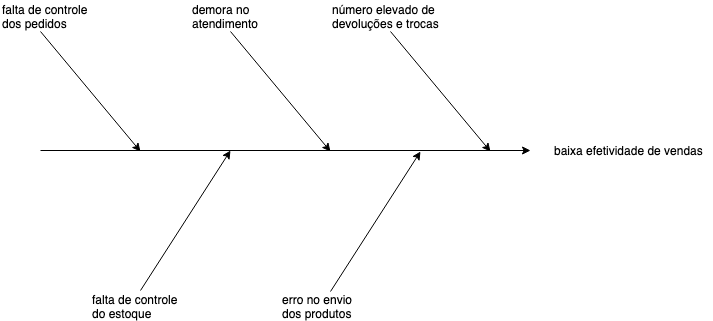
A empresa é descrita como uma loja física localizada em São Paulo - SP, fundada em agosto de 2008, a *PetShow* é uma loja voltada para a vendas de roupas para animais domésticos, com opções para cachorros e gatos de vários tamanhos.

Como contexto, foram levantados problemas comuns a pequenas lojas, tais como, falta de controle de pedidos, falta de controle de estoque, demora no atendimento, erro no envio de produtos e número elevado de devoluções e trocas.

A razão para a escolha desse nicho se dá pelo alto potencial de negócios do mercado Pet no Brasil, com diversas grandes redes de lojas atuando nas principais cidades, bem como pequenas empresas que se especializam na venda e fabricação de peças de vestuário para este segmento.

## 1.2. Declaração do Problema

O problema da baixa efetividade das vendas afeta a lucratividade da empresa devido à falta de controle de pedidos, falta de controle de estoque, demora no atendimento, erro no envio de produtos e número elevado de devoluções e trocas. Os benefícios da ferramenta de controle das vendas são: menor tempo de resposta ao cliente, organização dos registros de pedidos e estoque, registro facilitado de clientes e pets, maior controle das operações e rotinas diárias.

Figura 1 – Diagrama de Ishikawa

Fonte: Os Autores

## 1.3. *Stakeholders* e Restrições

Quadro 1 – *Stakeholders*

|  |  |
| --- | --- |
| **USUÁRIOS** | **COMENTÁRIOS** |
| Presidente | Usará o sistema para monitorar o trabalho dos funcionários e calcular o desempenho da loja. Acesso ao log do sistema, e todas as outras funcionalidades. |
| Diretor Geral | Usará o sistema para dar feedback aos funcionários, distribuir tarefas e verificar desempenho. Acesso ao *log* do sistema, e todas as outras funcionalidades. |
| Tesoureiro | Usará o sistema para gerar relatórios das vendas e impostos correspondentes para os contadores. |
| Gerente de Compras | Usará o sistema para acessar e modificar preço, estoque, imposto, código de barras, fornecedores e validade. Acesso ao relatório da evolução do preço do produto de acordo com os valores. Acesso ao relatório de venda de produto pro tempo |
| *Marketing* | Usará o sistema para acessar os totais de vendas por produto de acordo com os valores. Consulta de produto, preço, validade e estoque. |
| Gerente de RH | Usará o sistema para verificar desempenho dos funcionários de acordo com tempo de acesso e total de vendas por vendedor. |
| Auxiliar Administrativo | Usará o sistema para verificar os totais de venda, compra, estoque. Verificar imposto fiscal e detalhes das notas de compra, venda e devolução. Cadastrar produtos e fornecedores. |
| Recebedores | Usará o sistema para registrar o recebimento de mercadorias, repor e precificar as mercadorias nas prateleiras dos setores de vendas a que se destinam. Conferir se a mercadoria foi comprada, aceitar ou devolver mercadoria conforme pedido. Inserir as notas fiscais no sistema, conferir quantidades, preços, conferir e alterar códigos de barras se necessário. |
| Auxiliares de Loja | Usará o sistema para ajudar nas tarefas de conferência de código de barras dos produtos junto aos recebedores, reposição de mercadorias, precificação das prateleiras, cadastro de fornecedores e cadastro de produtos. |
| Vendedores | Usará o sistema para verificar a reposição das mercadorias do respectivo setor, atendimento de clientes online e presenciais de acordo com o setor de vendas correspondentes. Consultar o estoque, verificar disponibilidade do produto, verificar preços, visualizar perspectivas de entrega de produtos e transferir formulário de encomendas para o setor de compras. |
| Operadores de Caixa | Usará o sistema para registrar a venda de mercadorias, consultar preço, estoque. |
| Contador | Serão afetados pela implantação do novo sistema por que o cálculo do imposto fiscal da empresa é feito sobre o total das vendas do produto. Recebem os relatórios das vendas e impostos correspondentes, gerados pelo Tesoureiro. |
| Clientes | Serão afetados pela implantação do novo sistema na consulta de preços e quantidade dos produtos que ficará mais acessível; e na necessidade de preencher com mais detalhes as encomendas dos produtos. |

Fonte: Os Autores

Quadro 2 - Restrições

|  |  |
| --- | --- |
| **RESTRIÇÃO** | **RAZÃO (LÓGICA)** |
| Vendas, relatórios, dados etc, deverão ser armazenados em nuvem. | Gestor poderá consultar de qualquer local. |
| A ferramenta tem que ser acessível por outros aparelhos eletrônicos, tais como: smartphone, tablet etc. | Possibilita a inclusão de mais pessoas usando e usufruindo da ferramenta. |
| Ferramenta intuitiva. | Qualquer um pode aprender com facilidade, sem ser preciso treinamentos longos. |
| A versão 1.0 precisa ser lançada até 01/06/21. | Tempo limite para finalização do projeto. |

Fonte: Os Autores

# 2. Solução Proposta

A solução proposta é a criação de um sistema no qual consiste em um controle dos pedidos recebidos, armazenamento de informações do cliente como nome, endereço, e-mail, telefone, nome do *pet*, espécie, raça, porte, gênero. Além da gestão dos dados de venda com extração de relatórios mensais/anuais de lucro, e relatórios de estoque.

A interface gráfica deve ser em web com fácil acesso e intuitivo para que não haja uma curva de aprendizado muito grande aos usuários.

## 2.1. Lista de Necessidades

As necessidades iniciais do escopo do projeto incluem uma ferramenta de controle e execução de vendas, assim como o controle e armazenamento de informações do cliente, pedido, e status do estoque da loja.

N01 - Cadastro de produto

N02 - Cadastro de cliente

N03 - Módulo de pedidos: listagem de pedidos, acompanhamento de pedido

N04 - Módulo de finanças: relatório de vendas e lucro

N05 - Sistema de controle de estoque

N06 - Banco de dados

## 2.2. Requisitos do Sistema

Quadro 3 - Requisitos de Sistema

|  |  |  |
| --- | --- | --- |
| **#** | **CARACTERÍSTICA** | **DESCRIÇÃO** |
| R01 | Registro de dados do cliente | O usuário deverá preencher dados mínimos para serem registrados na plataforma de compra da loja, e efetuar pedidos. |
| R02 | Identificação do *pet* e do cliente | Os clientes deverão disponibilizar todas as informações de seus pets, como nome, raça, tamanho e gênero. Haverá a possibilidade de inserção de mais de um *pet* por cliente, caso seja necessário. |
| R03 | Detalhamento de produtos | Os produtos serão cadastrados, catalogados, e todas as informações de fornecedores para aquele produto, devem ser também cadastradas. |
| R04 | Controle de pedidos | Os funcionários podem inserir as informações dos pedidos em sistema, consultar os pedidos realizados. |
| R07 | Cálculo de vendas | Para toda venda realizada, será contabilizada a margem de venda por produto vendido, assim como a subtração do produto em estoque, margem bruta e margem líquida. |

Fonte: Os Autores

## 2.3. Regras de negócio

Quadro 4 – Lista de Restrição

|  |  |
| --- | --- |
| **REGRA DE NEGÓCIO** | **DESCRIÇÃO** |
| RN 01 | Somente usuários logados têm acesso ao sistema |
| RN 02 | Somente usuários do tipo ‘gerente’ podem cadastrar novos usuários ou alterar seu tipo |
| RN 03 | Somente o usuário logado pode alterar a própria senha |
| RN 04 | Tamanhos |
| RN 05 | Situação de pedidos,  Valores físicos das tabelas estáticas |
| RN 06 | Token de acesso é válido por 10 minutos |
| RN 07 | Os pedidos com situação ‘recebido’ não efetuam desconto de itens do estoque |
| RN 08 | Os pedidos com situação ‘concluído’ efetuam desconto dos itens de estoque |
| RN 09 | Somente pedidos com situação ‘cancelado’ cuja situação anterior era ‘concluído’ efetuam devolução de itens ao estoque |
| RN 10 | O sistema não efetuará remoção de registros |

Fonte: Os Autores

## 2.4. Descrição das funcionalidades

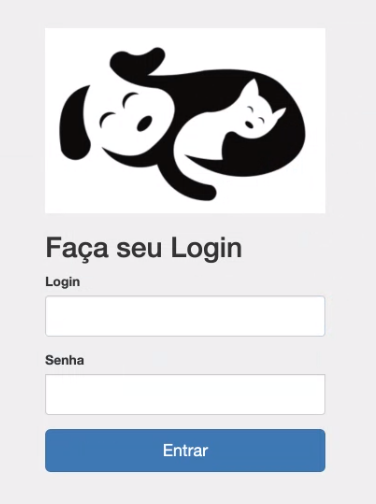
### 2.4.1. *Login*

É a primeira tela ao abrir o sistema, essa tela permite que os funcionários cadastrem suas credenciais de usuário e senha.

Após devidamente autenticado no sistema, o usuário acessa a página de Lista de Produtos, e tem acesso ao menu de navegação da interface gráfica, podendo acessar qualquer funcionalidade de sistema.

Caso o usuário cadastre uma credencial não válida, a página de *login* mostra uma mensagem de usuário ou senha inválidos, e o campo de credenciais deve ser novamente preenchido.

A tecnologia utilizada para autenticação é a de JWT (JSON Web Token), padrão da internet para criação de dados com assinatura ou criptografia, cujo *payload* contém o JSON correspondendo a um conjunto de declarações. Os tokens são assinados usando uma chave privada da aplicação. Neste caso, o usuário, ao efetuar o login, possui um token de autenticação que lhe permite fazer as operações pertinentes a seu papel na organização. Quando o token expira, o usuário é redirecionado a página de login, necessitando efetuar novo fornecimento de login e senha para voltar ao sistema.

Figura 2 – Tela de *Login*

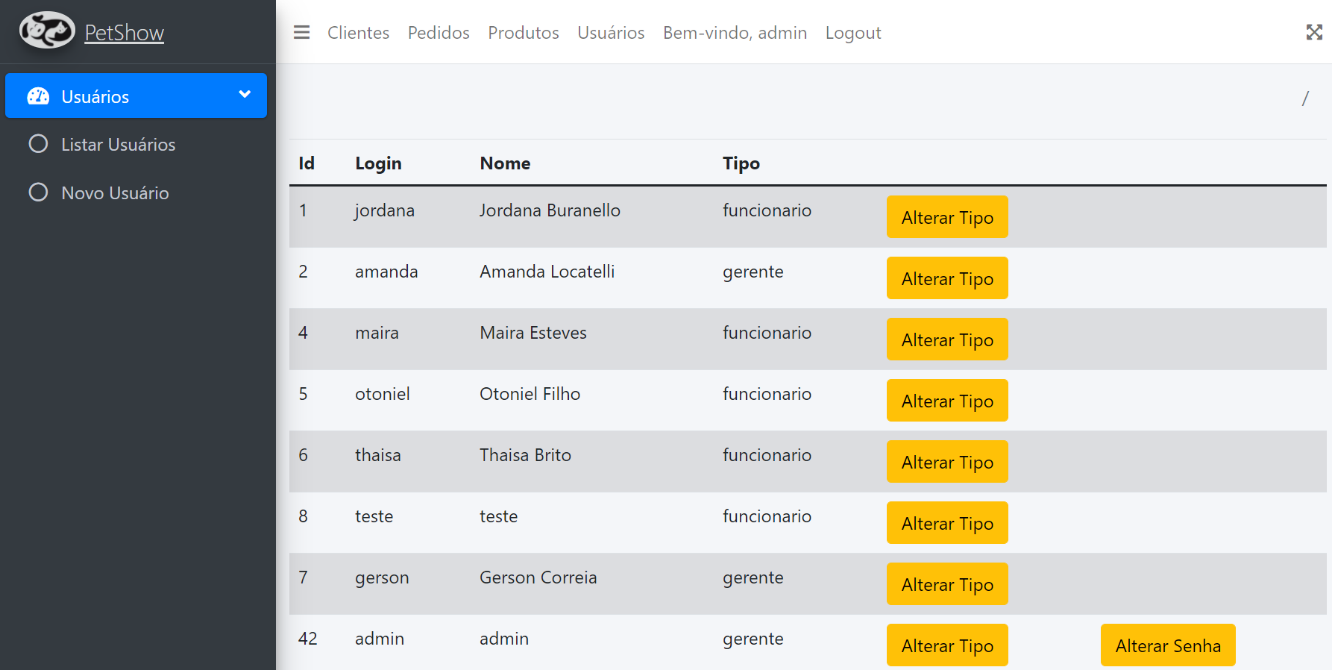
Fonte: Os Autores

### 2.4.2. Listagem de Usuários

Sua funcionalidade é exibir os dados dos usuários e prover ações de alteração em função das regras de negócio existentes.

É composto por uma tela de listagem de usuários, contendo seus dados. Ao lado de cada item listado pode ser exibido o botão de alterar tipo de usuário ou alterar senha.

Além disso, possui o painel de navegação lateral, com respeito ao escopo e no topo, para troca de escopo.

Figura 3 – Listagem de Usuários

Fonte: Os autores

### 2.4.3. Cadastro de Usuários

Sua funcionalidade é elaborar um novo cadastro de usuário.

É composto por uma tela de com campos de texto para nome, login, senha e uma caixa de opções de tipo, bem como botão para confirmar o cadastro.

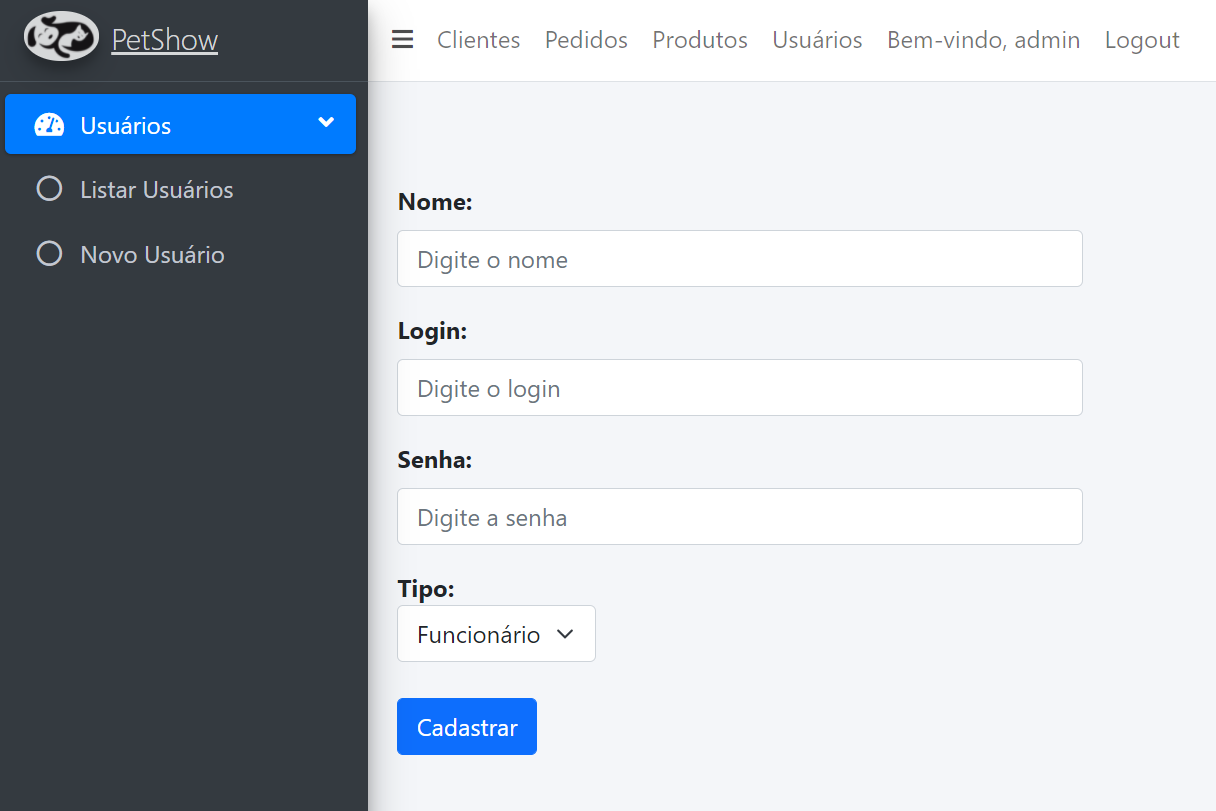
Além disso, possui o painel de navegação lateral, com respeito ao escopo e no topo, para troca de escopo.

Figura - Cadastro de Usuários

Fonte: Os Autores

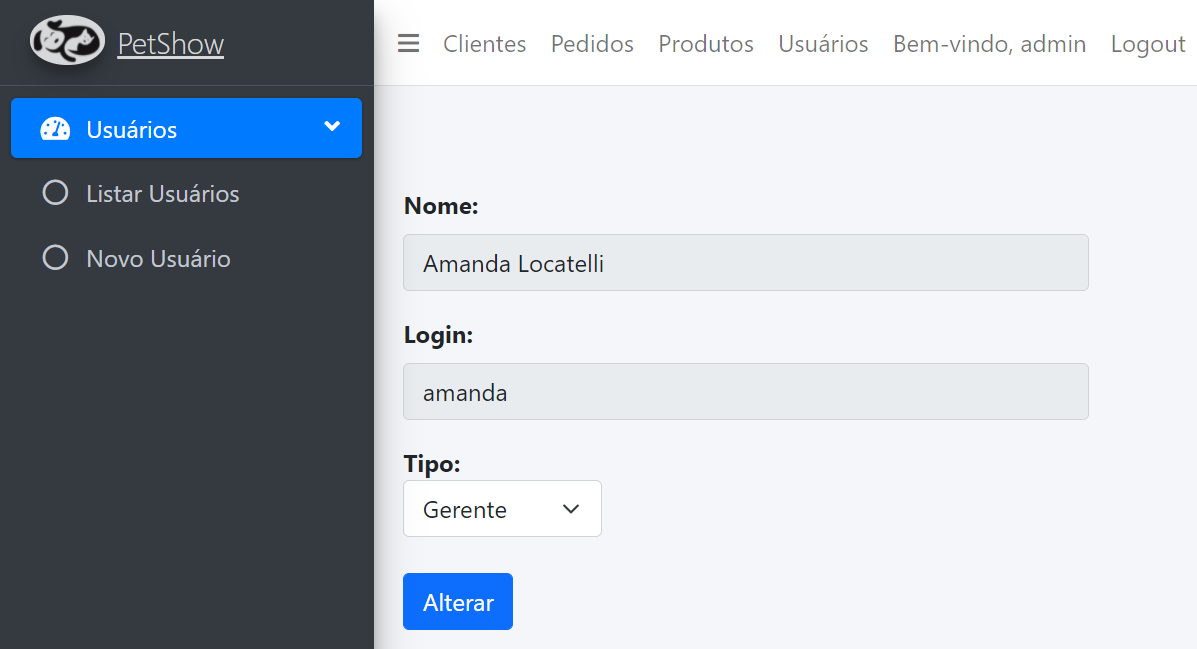
### 2.4.4. Alteração de Tipo de Usuário

Sua funcionalidade é atualizar o tipo de um usuário. Somente gerentes têm acesso a essa tela.

É composto por uma tela de com campos de texto para nome, login, senha e uma caixa de opções de tipo, bem como botão para confirmar a alteração.

Além disso, possui o painel de navegação lateral, com respeito ao escopo e no topo, para troca de escopo.

Figura 5 – Alteração Tipo de Usuário



Fonte: Os autores

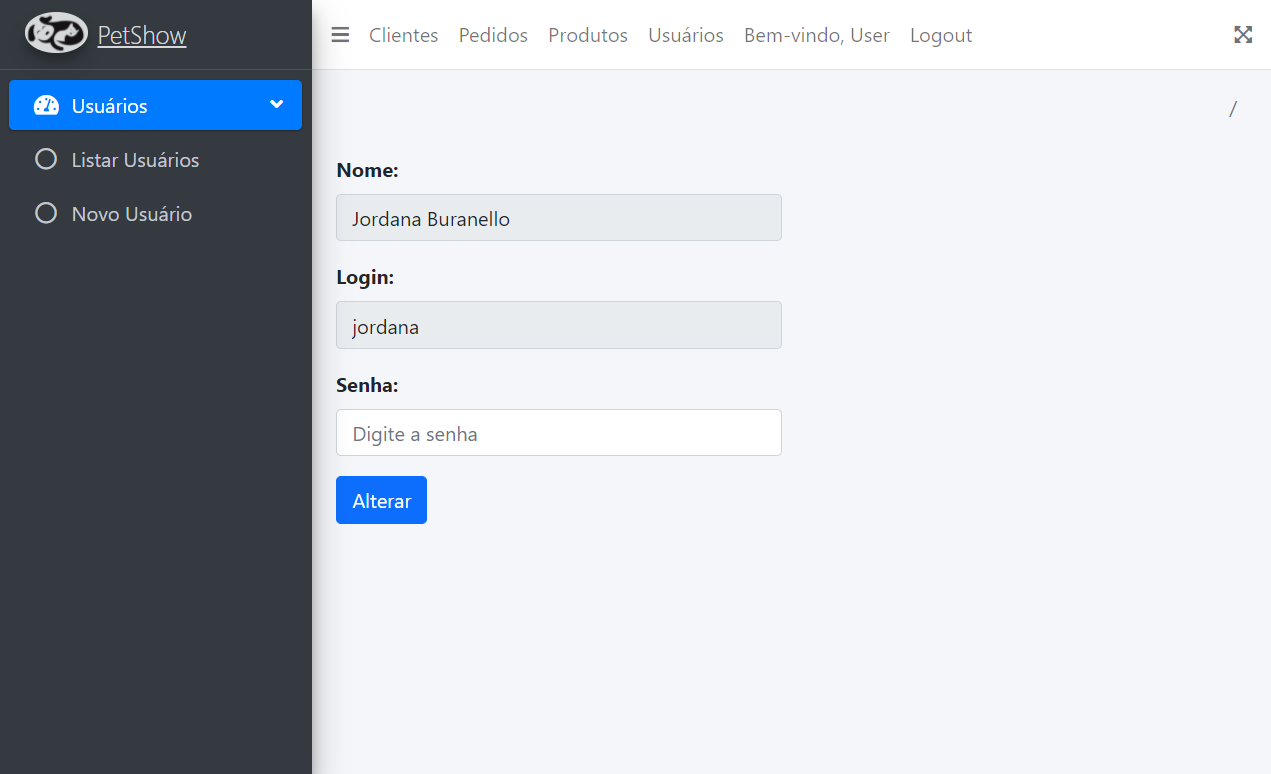
### 2.4.5. Alteração de senha de Usuário

Sua funcionalidade é atualizar a senha de um usuário. Usuário logado somente pode alterar a própria senha.

É composto por uma tela de com campos de texto para nome, login e senha, bem como botão para confirmar a alteração.

Além disso, possui o painel de navegação lateral, com respeito ao escopo e no topo, para troca de escopo.

Figura 6 – Alteração de Senha de Usuário



Fonte: Os autores

### 2.4.6. Lista de Produtos

Esta tela mostra uma lista com os dados dos produtos para consulta dos usuários bem como opção para edição de dados do produto,

Consta um menu para as telas Clientes, Pedidos, Produtos, Usuários e um botão para a tela de cadastro de produtos. Todos os produtos cadastrados sem imagem, podem ser alterados após inseridos, e enquanto não há imagem, aparece o ícone de imagem corrompida.

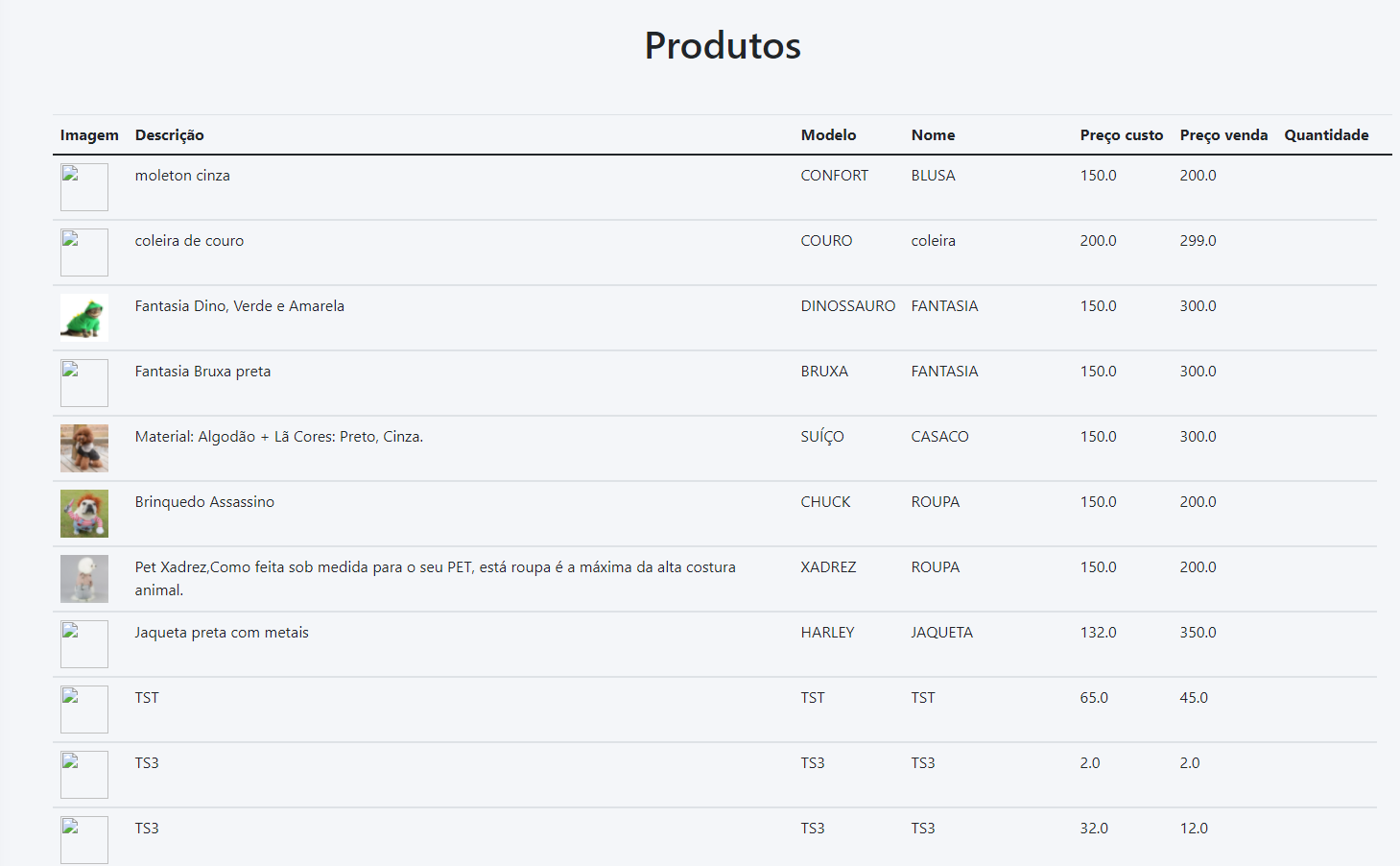


Figura Listagem de Produtos

Fonte: Os Autores

### 2.4.7. Cadastro de Clientes e Pets

### 2.4.8. Cadastro de Produtos

Esta é onde será realizado o cadastro dos produtos, constam os campos: Nome, Descrição, Modelo, Cód. de barras, Porcentagem, Preço custo, Preço venda, Quantidade, Foto, Marca, Tamanho e para qual animal e um botão para registro do cadastro.  
Há também o menu para as demais telas e um botão que leva para a tela de “Lista Produto”.

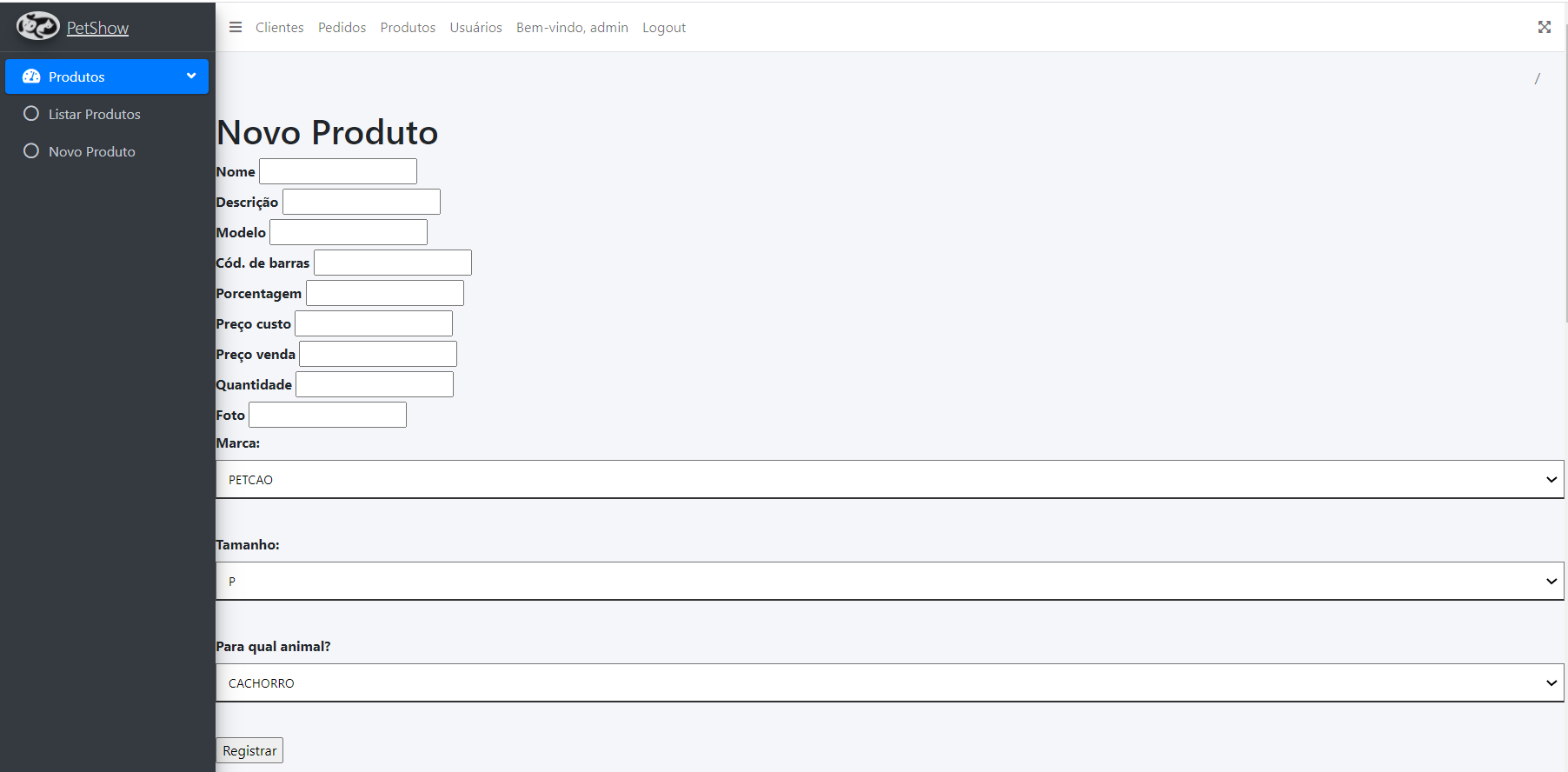


Figura – Cadastro de Produtos

Fonte: Os Autores

### 2.4.9. Cadastro de Pedidos

Lista de cadastro dos pedidos dos clientes. É possível cadastrar o pedido, e a página já calcula o valor contabilizando as quantidades e valores de cada item adicionado ao carrinho.

# Projeto, análise e implementação

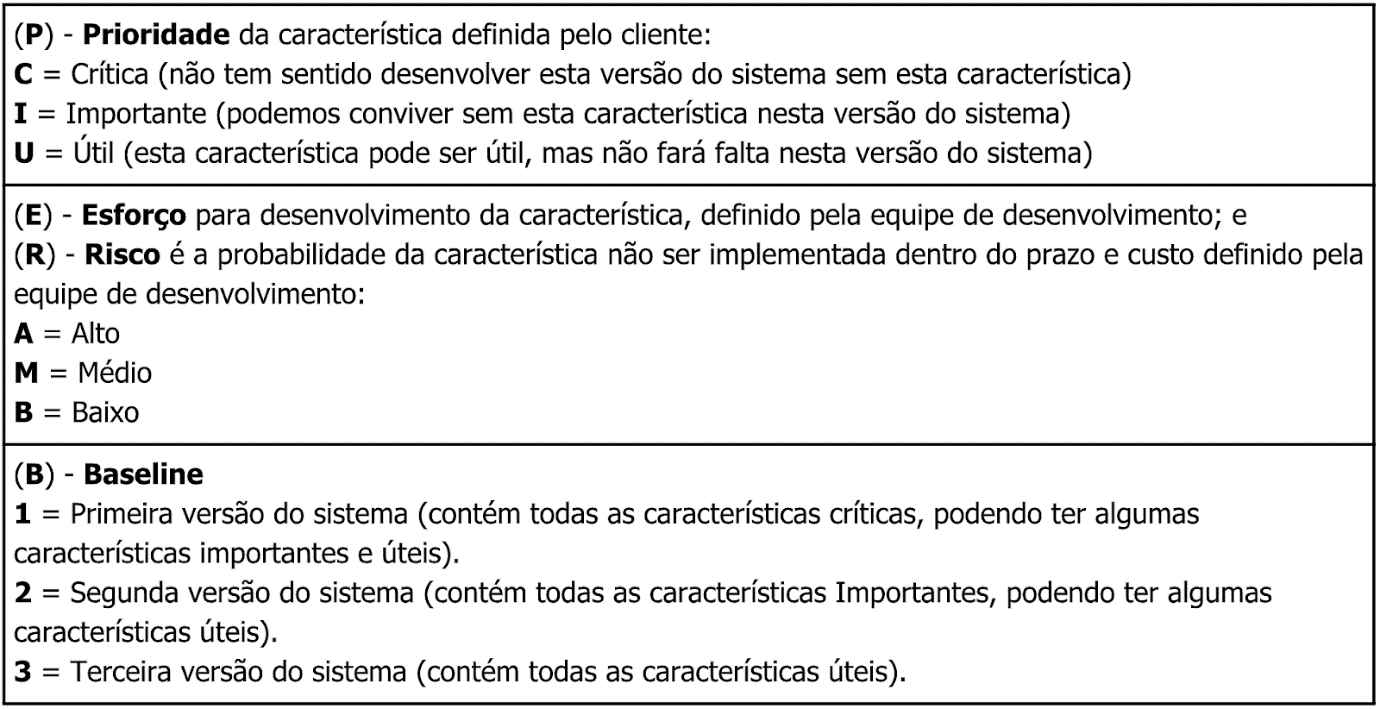
## Lista de Características - Classificações

Quadro 5 – Lista de Características - Classificações

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **LISTA DE CARACTERÍSTICAS** | **(P)rioridade (E)sforço (R)isco (B)*aseline*** | **(P)** | **(E)** | **(R)** | **(B)** |
| C01 | Registro de dados do cliente | C | A | B | 1 |  |
| C02 | Identificação do *pet* do cliente | C | A | B | 1 |  |
| C03 | Detalhamento de produtos | C | A | B | 1 |  |
| C04 | Controle de pedidos | I | B | B | 2 |  |
| C05 | Cálculo de vendas | C | M | M | 1 |  |
| C06 | Relatório de Vendas | C | A | M | 1 |  |

Fonte: Os autores

Figura 2 - Legenda

****

Fonte: Os autores

## 3.2. Matriz de Rastreabilidade – Necessidade x Características

Quadro 6 - Necessidades

|  |  |
| --- | --- |
| **#** | **NECESSIDADES** |
| N01 | Detalhamento de Produtos |
| N02 | Registro de Dados do Cliente |
| N03 | Módulo de pedidos: listagem de pedidos, acompanhamento de pedido, cancelamento, alterações em geral |
| N04 | Módulo de finanças: relatórios de vendas e lucro |
| N06 | Sistema de controle de estoque |
| N07 | Banco de dados |

Fonte: Os autores

Quadro 7 – Matriz de Rastreabilidade

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MATRIZ DE RASTREABILIDADE** | **NECESSIDADES** |  |  |  |  |  |  |  |  |  |
| **#** | **CARACTERÍSTICA** | **N01** | **N02** | **N03** | **N04** | **N05** | **N06** | **N07** | **N08** | **N09** |
| C01 | Registro de dados do cliente |  | x |  |  |  |  | x |  |  |
| C02 | Identificação do *pet* do cliente |  | x |  |  |  |  | x |  |  |
| C03 | Detalhamento de produtos | x |  |  |  |  | x | x |  |  |
| C04 | Controle de pedidos |  |  | x |  |  |  |  |  |  |
| C06 | Cálculo de vendas |  |  |  | x |  |  |  |  |  |

Fonte: Os autores

## Arquitetura, módulos e subsistemas

A solução proposta consiste de um sistema formado pelos seguintes componentes:

### Subsistema *Back-End* (API)

Responsável pela comunicação com o banco de dados. Possui *controllers* e modelos específicos para *login*, cadastro de usuários, cadastro de clientes, cadastro de produtos e cadastro de pedidos. Possui *endpoints* especializados que recebem ou servem dados no formato JSON. Está hospedada em container próprio no serviço na nuvem Heroku sob a URL <http://petshow-api.herokuapp.com/>.

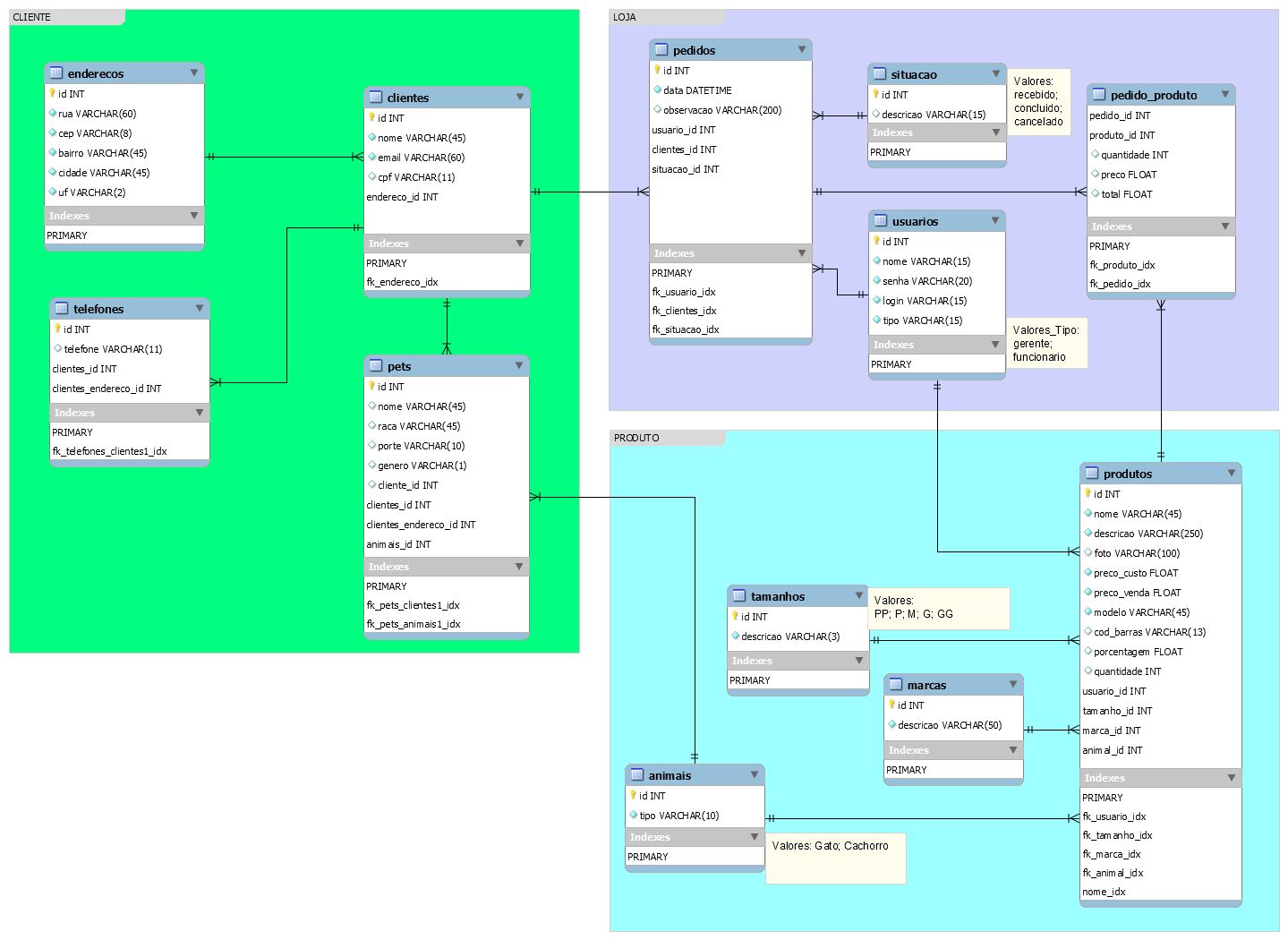
### Subsistema *Front-End*

Responsável pela comunicação com a API e servir o cliente com páginas HTML baseadas nos templates existentes no sistema. Possui controllers específicos para login, listagem e cadastro de usuários, clientes, produtos e pedidos. Comunica-se com a API por meio de requisições HTTP servindo ou recebendo dados no formato JSON. Está hospedada em container próprio no serviço na nuvem Heroku sob a URL <http://petshow-app.herokuapp.com/>.

### Projeto do Banco de Dados

Responsável por armazenar os dados da aplicação. Utiliza-se uma instância do PostgreSQL, hospedada no serviço na nuvem Heroku.

Figura 9 – Modelo Físico



Fonte: Os Autores

Conforme ilustrado no modelo (figura 9), o banco de dados se divide em três conjuntos de tabelas, relacionadas a produto, cliente e pedido.

Na seção relacionada ao produto temos as tabelas:

**Produtos:** para o cadastro de cada produto da loja, com os campos:

ID (chave primária): identificador individual de cada produto;

Nome: registro do nome do produto;

Descrição: contém as informações mais relevantes do produto como material, cor e etc.;

Foto: este campo armazena o caminho para a imagem do produto;

Preco\_custo: valor de custo do produto;

Preco\_venda: valor de venda do produto;

Modelo: descrição do modelo informado pelo fabricante;

Cod\_barras: número do código de barras;

Porcentagem: valor do lucro em porcentagem;

Quantidade: a quantidade do produto no estoque;

Tabelas relacionadas:

**Tamanhos** (chave estrangeira): contém os tamanhos disponíveis, com os valores: PP, P, M, G, GG;

**Marcas** (chave estrangeira): tabela com as marcas de produtos;

**Animais** (chave estrangeira): tipo de animal a qual o produto se destina, valores: Cachorro, Gato;

**Usuários** (chave estrangeira): registro do usuário que realizou o cadastro do produto

Na seção relacionada ao cliente temos:

Clientes: para registro dos clientes da loja proprietários ou não dos pets. Campos:

ID (chave primária): identificador individual de cada cliente;

Nome: recebe o nome do cliente;

Email: recebe o email de contato;

Cpf: documento CPF do cliente;

Endereço\_id (chave estrangeira):

Tabelas Relacionadas:

**Endereço**: tabela com o endereço do cliente nos campos:

ID (chave primária): identificador individual;

Rua: nome da rua do cliente;

Cep: número do CEP da rua;

Bairro: nome do bairro;

Cidade: nome da cidade;

Uf: sigla do estado com dois dígitos;

**Telefones:** números de telefones de contato do cliente, podendo ser mais de um;

ID (chave primária): identificador único;

Telefone: registro do número de telefone do cliente;

### Estrutura do sistema

O sistema utiliza a arquitetura de microsserviços, isto é, construído desmembrando-se em serviços independentes.

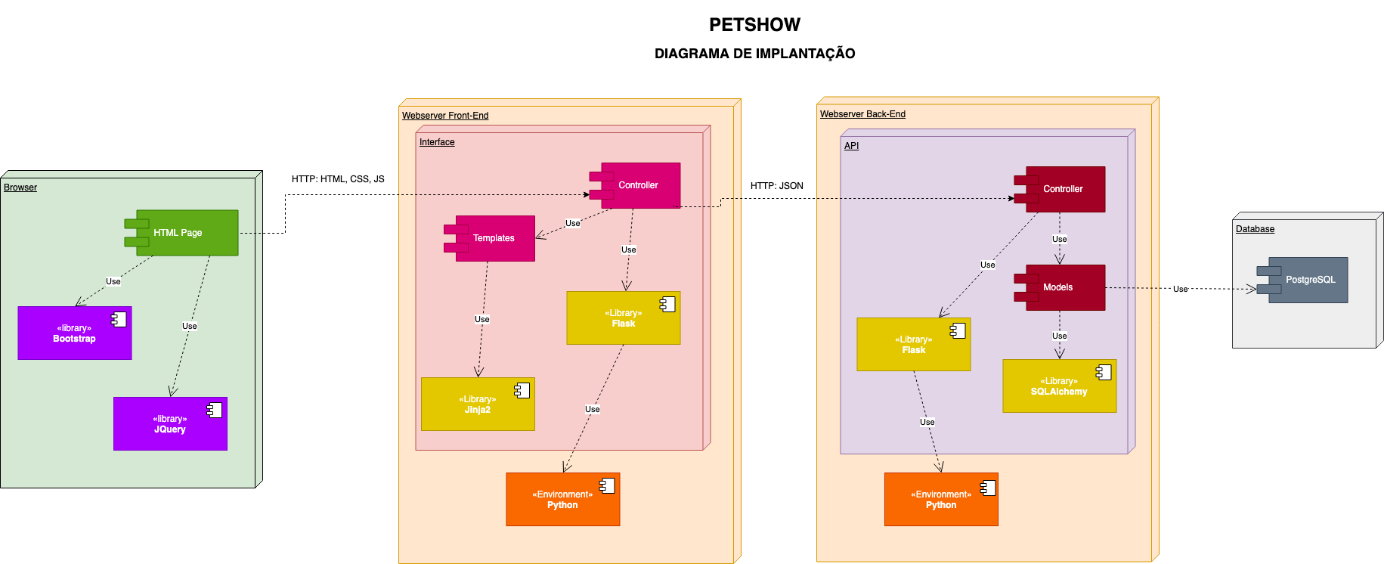
Neste caso específico, o sistema foi dividido em duas grandes aplicações: a aplicação principal *Front-End*, responsável por prover visões para interação de usuário, tais como telas de cadastro ou listagem de dados; assim como uma aplicação *Back-End*, a nossa API (*Application Programming Interface*), esta responsável por receber e enviar dados à aplicação principal, bem como, fornecer tokens de acesso ao sistema.

Ambas as aplicações têm subdivisões em microsserviços. O sistema faz uso do conceito de *blueprint*, fornecido pelo framework *Flask*, recurso que permite e gerencia a divisão de cada microsserviço presente em cada uma das duas aplicações. Conforme o modelo de dados descrito no item 3.2.3, teremos microsserviços específicos para cada conjunto de tabelas, sendo Clientes, Loja e Produto. Cada microsserviço tem um controller que gerencia rotas pertinentes a seu grupo, bem como seus respectivos templates.

O sistema segue, em suma, o princípio da responsabilidade única, o que também facilita sua manutenibilidade, e no caso do processo de desenvolvimento, uma divisão de tarefas no grupo de trabalho.

A figura 10 contém o diagrama de implantação, descrevendo os componentes do sistema e como se comunicam.

Figura 10 – Arquitetura



Fonte: Os Autores

## .

## 3.4. Tecnologias utilizadas

Quadro 8 – Tecnologias utilizadas

|  |  |
| --- | --- |
| **Tecnologia** | **Justificativa** |
| Git | Sistema de versionamento de código distribuído entre todos os desenvolvedores.  Necessário também devido as ferramentas de hospedagem escolhidas. |
| Python | Linguagem de programação utilizada para desenvolver o *backend* do projeto. |
| PostgresSQL | Sistema gerenciador de banco de dados relacional utilizado para persistirem-se os dados cadastrais do projeto. |
| JSON | Notação de objetos em *Javascript*, utilizado para formatar os dados de entrada e saída da API |
| Flask | Biblioteca em Python utilizada para servir-se as funcionalidades do *backend* por meio do protocolo HTTP. |
| JWT | Tecnologia de autenticação via token *JSON*, utilizada para proteger as rotas da API e da aplicação principal. |
| Jinja 2 | Biblioteca do Python utilizada para, em conjunto com o *Flask*, montar-se no *backend* as páginas HTML que serão servidas ao navegador cliente. |
| HTML | Padrão no qual as páginas servidas ao navegador cliente estão codificadas. |
| CSS | Padrão utilizado pelo navegador para estilizar e formatar as páginas clientes adequadamente. |
| Javascript | Linguagem de programação utilizada para desenvolver o *frontend* do projeto. |
| Bootstrap | Biblioteca utilizada para simplificar e padronizar a estilização do *frontend*. |
| jQuery | Biblioteca em *Javascript* utilizada para simplificar e agilizar o desenvolvimento de diversas funcionalidades do *frontend*. |
| Heroku | Utilizado para hospedar o *backend* e o banco de dados. |
| AdminLTE | Biblioteca contendo componentes HTML, folhas de estilo CSS e scripts *Javascript*, utilizada para facilitar o desenvolvimento dos *templates* do sistema |

**Fonte: Os autores.**

# 4. Considerações Finais

Esse projeto conseguiu trazer para o cliente a resolução do problema de administração e gestão das vendas, assim como a gestão do seu estoque de forma automatizada.

Com a interface gráfica intuitiva e acesso restrito às funções específicas a gestão de estoque e de vendas agora é realizada de forma automatizada.

O cliente também se beneficiou de um local para a gestão de suas vendas e o acompanhamento diário, assim como a gestão por funcionário de cada operação realizada dentro da loja.

O cliente recebeu o acesso e treinamento à plataforma, e terá direito à *debug* do sistema por 3 meses após o treinamento.

Citação direta do cliente:

*O Sistema salvou nossas vidas. Agora consigo cadastrar rapidamente pedidos, identificar os vendedores com melhor performance, e consigo centralizar funções específicas para cada senioridade no meu time de funcionários (Maria das Dores, 2021)*

O projeto desenvolvido pode ser acessado por meio da URL abaixo:

<http://petshow-app.herokuapp.com/>

# Referências bibliográficas

CLÁUDIO DIAS NETO, Arilo. **Modelagem de Dados Tutorial**. 2011. Disponível em: < <https://www.devmedia.com.br/modelagem-de-dados-tutorial/20398>>. Acesso em: 01 mai 2021.

DUCKETT, Jon. HTML & CSS – Design and Build Websites. John Wiley & Sons. 2011.

DUCKETT, Jon. Javascript & JQuery – Interactive Front-End Development. John Wiley & Sons. 2014

GIRIDHAR, Chetan. **Aprendendo Padrões de Projeto em Python**. NOVATEC. 2016.

GRINBERG Miguel. **Flask Web Development.** O’Reilly Media, Inc. May 2014.

M. F. Lungu. **Bootstrapping an ubiquitous monitoring ecosystem for accelerating vocabulary acquisition, Proccedings of the 10th European Conference on Software Architecture Workshops ser**. April 2016.

NUNES, Rodrigo. CAVALLIERI, Beatriz. BERNARDO, Fernanda. **Introdução às Web APIs** **– WEBDOCS.** 2021. Disponível em: < <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Client-side_web_APIs/Introduction>>. Acesso em: 15 mar 2021.

NUNES, Rodrigo. CAVALLIERI, Beatriz. BERNARDO, Fernanda. **FormData.** 2021. Disponível em: < <https://developer.mozilla.org/pt-BR/docs/Web/API/FormData>>. Acesso em: 25 maio 2021.

# Glossário

1. Aplicação ou App– Software que faz uso de serviços de rede tais como transferência de arquivos, login remoto e correio eletrônico.
2. Catalogar – Termo usado para gerar um código único de identificação do pedido, ou cliente, ou produto, e armazenamento da informação em banco de dados local ou em nuvem.
3. Carrinho de Compras – Programa que permite gerenciar e visualizar compras realizadas em um site.
4. *Pet* – Nomenclatura em inglês para animal doméstico, e no contexto da *Petshow* nomenclatura para cachorros e gatos.

# Apêndice A

**PETSHOW API**

API para uso na aplicação da loja *PETSHOW*. Utiliza *framework* *Flask* (linguagem *Python*) e banco de dados *PostgreSQL*. Também implementa classes utilizando ORM *Flask*-*SQLAlchemy*.

Está dividida em 4 componentes denominados ‘Usuários’, ‘Clientes’, ‘Produtos’ e ‘Pedidos’, definidos nas rotas a seguir:

**Usuários:**

**GET ‘/usuarios/’**

Função: buscar a listagem de usuários.

Exemplo de entrada:

{

"nome":"usuario",

"senha":"senha",

"login":"usuario",

"tipo":"funcionario"

}

Resposta esperada: Lista de usuários.

Exemplo de resposta:

[

{

"id": 1,

"login": "jose",

"nome": "Jose",

"tipo": "funcionario"

},

{

"id": 2,

"login": "maria",

"nome": "Maria",

"tipo": "gerente"

}

]

Mensagens de erro:

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

**GET ‘/usuarios/<login>’**

Função: buscar os dados de usuário específico em função do seu login

Resposta esperada: detalhes do usuário

Exemplo de resposta:

{

"id": 2,

"login": "jose",

"nome": "Jose",

"tipo": "funcionario"

}

Mensagens de erro:

Usuário inexistente: ‘Usuário não encontrado’

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

**POST ‘/usuarios/autenticar’**

Função: autenticar usuário comparando login e senha recebidos no *body* da requisição.

Exemplo de entrada:

{

"login":"usuario",

"senha":"senha"

}

Resposta esperada: ‘Usuário autenticado’

Mensagens de erro:

Usuário ou senha incorreta: ‘Usuário ou senha incorretos’

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

**POST ‘/usuarios/novo’**

Função: cadastrar novo usuário passando os dados no *body* da requisição

Exemplo de entrada:

{

"nome":"usuario",

"senha":"senha",

"login":"usuario",

"tipo":"funcionario"

}

Resposta esperada: ‘Usuário cadastrado’

Mensagens de erro:

Falta de dados: ‘Os dados do usuário não foram inseridos’

Usuário existente: ‘Usuário já cadastrado’

**PATCH ‘/usuarios/alterar\_senha**

Função: alterar a senha de usuário passando *login* e nova senha no *body* da requisição

Exemplo de entrada:

{

"login":"usuario",

"senha":"senha"

}

Resposta esperada: ‘Senha alterada’

Mensagens de erro:

Erro de integridade: ‘Não foi possível fazer a alteração’

Falta de dados: ‘Os dados do usuário não foram inseridos’

**PATCH ‘/usuarios/alterartipo’**

Função: alterar tipo de usuário (Gerente ou Funcionário) passando login e tipo no *body* da requisição

Exemplo de entrada:

{

"login":"usuario",

"tipo":"gerente"

}

Resposta esperada: ‘Tipo alterado’

Mensagens de erro:

Falta de dados: ‘Os dados do usuário não foram inseridos’

Erro de integridade: ‘Não foi possível fazer a alteração’

**Clientes:**

**GET ‘/clientes/pets/’**

Função: lista os pets cadastrados no sistema

Resposta esperada: lista de *pets*

Exemplo de resposta:

[

{

"animal\_id": 1,

"genero": "m",

"id": 1,

"nome": "Floquinho",

"porte": "pequeno",

"raca": "Vira-Lata"

},

{

"animal\_id": 2,

"genero": "m",

"id": 2,

"nome": "Soft Kitty",

"porte": "medio",

"raca": "Siamês"

}

]

Mensagens de erro:

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

**GET ‘/clientes/’**

Função: listar todos os clientes com seus detalhes

Resposta esperada: lista dos clientes

Exemplo de resposta:

{

"clientes": [

{

"cliente": {

"cpf": "10000000001",

"email": "[beakiddo@kiddo.com](mailto:beakiddo@kiddo.com)",

"endereco": {

"bairro": "Vila Frente",

"cep": "060000-160",

"cidade": "Osasco",

"id": 1,

"numero": "23",

"rua": "Rua 01",

"uf": "SP"

},

"id": 1,

"nome": "Beatrix Kiddo",

"pets": [

{

"animal\_id": 1,

"genero": "m",

"id": 1,

"nome": "Floquinho",

"porte": "pequeno",

"raca": "Vira-Lata"

},

{

"animal\_id": 2,

"genero": "m",

"id": 2,

"nome": "Soft Kitty",

"porte": "medio",

"raca": "Siamês"

}

],

"telefones": [

{

"cliente\_id": 1,

"id": 1,

"telefone": "01199938884"

},

{

"cliente\_id": 1,

"id": 2,

"telefone": "01199882244"

}

]

}

},

{

"cliente": {

"cpf": "10000000009",

"email": "[bill@kill.com](mailto:bill@kill.com)",

"endereco": {

"bairro": "Vila Ré",

"cep": "060020-161",

"cidade": "Barueri",

"id": 2,

"numero": "43",

"rua": "Rua 02",

"uf": "SP"

},

"id": 2,

"nome": "Bill",

"pets": [

{

"animal\_id": 1,

"genero": "m",

"id": 1,

"nome": "Floquinho",

"porte": "pequeno",

"raca": "Vira-Lata"

}

],

"telefones": [

{

"cliente\_id": 2,

"id": 3,

"telefone": "01199238884"

},

{

"cliente\_id": 2,

"id": 4,

"telefone": "01199882244"

}

]

}

},

{

"cliente": {

"cpf": "10000000011",

"email": "[lorelai@kill.com](mailto:lorelai@kill.com)",

"endereco": {

"bairro": "Vila Vazia",

"cep": "060001-162",

"cidade": "Mauá",

"id": 3,

"numero": "63",

"rua": "Rua 03",

"uf": "SP"

},

"id": 3,

"nome": "Rory",

"pets": [

{

"animal\_id": 2,

"genero": "f",

"id": 4,

"nome": "Garfilda",

"porte": "grande",

"raca": "Zebrado"

}

],

"telefones": [

{

"cliente\_id": 3,

"id": 5,

"telefone": "01199966884"

},

{

"cliente\_id": 3,

"id": 6,

"telefone": "01199886644"

}

]

}

}

]

}

Mensagens de erro:

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

**POST ‘/clientes/’**

Função: cadastrar novo cliente passando os dados, seus pets e telefones no *body* da requisição

Exemplo de entrada:

{

"nome":"Fulano",

"[email":"fulano@det.al](mailto:email%22:%22fulano@det.al)",

"cpf":"2",

"endereco":{

"rua":"rua quinze",

"numero":"25 fundos",

"cep":"06000-000",

"bairro":"Jardim Europa",

"cidade":"Osasco",

"uf":"AC"

},

"telefones": [{

"telefone":"4545-4665"

},

{

"telefone":"4545-4699"

}],

"pets": [{

"nome":"Spike",

"raca":"Pequines",

"porte":"pequeno",

"genero":"m",

"animal\_id":1

},

{

"nome":"Garfield",

"raca":"vira lata",

"porte":"medio",

"genero":"m",

"animal\_id":2

}]

}

Resposta esperada: ‘Cliente cadastrado’

Mensagens de erro:

Falta de dados: ‘Os dados do cliente não foram inseridos’

Problemas com dados de telefone: ‘Nao foi possivel cadastrar o telefone’

Problemas com dados de pet: ‘Nao foi possivel cadastrar o pet’

Problemas com dados de cliente: ‘Não foi possível cadastrar o cliente’

Problemas com dados de endereço: ‘Nao foi possivel cadastrar o endereço’

**PUT ‘/clientes/<id>/alterar/’**

Função: alterar dados cadastrais de cliente, passando os dados no *body* da requisição

Observação: Para o atributo ‘*pets’*, o *body* deve receber seus respectivos ids. Caso seja cadastrado um novo pet na requisição, o id deverá receber o valor 0 (‘zero’). No caso de remoção de todos os pets ou telefones do cadastro, basta passar uma lista vazia para o atributo.

Exemplo de entrada:

{

"nome":"Fulano",

"[email":"fulano@det.al](mailto:email%22:%22fulano@det.al)",

"cpf":"3",

"endereco":{

"rua":"rua quinze",

"numero":"25 fundos",

"cep":"06000-000",

"bairro":"Jardim Europa",

"cidade":"Osasco",

"uf":"AC"

},

"telefones": [{

"telefone":"9999-4665"

},

{

"telefone":"4545-4699"

},

{

"telefone":"2335-4699"

}],

"pets": [{

"id": 5,

"nome":"Spike",

"raca":"Pequines",

"porte":"pequeno",

"genero":"m",

"animal\_id":1

},

{

"id": 6,

"nome":"Garfield",

"raca":"vira lata",

"porte":"medio",

"genero":"m",

"animal\_id":2

}]

}

Resposta esperada: ‘Cliente alterado’

Mensagens de erro:

Falta de dados: ‘Os dados do cliente não foram inseridos’

Cliente inexistente: ‘Cliente não encontrado’

**Produtos:**

**GET ‘/produtos/’**

Função: listar todos os produtos

Resposta esperada: lista dos produtos

Exemplo de resposta:

[

{

"animal\_id": 1,

"cod\_barras": 39232839,

"descricao": "Azul",

"foto": "",

"id": 1,

"marca\_id": 1,

"modelo": "Pullover",

"nome": "Blusa Cão",

"porcentagem": 10.0,

"preco\_custo": 100.0,

"preco\_venda": 120.0,

"quantidade": 10,

"tamanho\_id": 1,

"usuario\_id": 1

},

{

"animal\_id": 1,

"cod\_barras": 39232831,

"descricao": "Azul",

"foto": "",

"id": 2,

"marca\_id": 1,

"modelo": "Pullover",

"nome": "Blusa Cão",

"porcentagem": 10.0,

"preco\_custo": 100.0,

"preco\_venda": 120.0,

"quantidade": 10,

"tamanho\_id": 2,

"usuario\_id": 1

},

{

"animal\_id": 1,

"cod\_barras": 39232834,

"descricao": "Azul",

"foto": "",

"id": 3,

"marca\_id": 1,

"modelo": "Pullover",

"nome": "Blusa Cão",

"porcentagem": 10.0,

"preco\_custo": 100.0,

"preco\_venda": 120.0,

"quantidade": 10,

"tamanho\_id": 3,

"usuario\_id": 1

},

{

"animal\_id": 2,

"cod\_barras": 39232849,

"descricao": "Vermelha",

"foto": "",

"id": 4,

"marca\_id": 2,

"modelo": "Pullover",

"nome": "Blusa Gato",

"porcentagem": 10.0,

"preco\_custo": 100.0,

"preco\_venda": 120.0,

"quantidade": 30,

"tamanho\_id": 2,

"usuario\_id": 2

}

]

Mensagens de erro:

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

**POST ‘/produtos/’**

Função: cadastrar novo produto passando seus dados no *body* da requisição

Exemplo de entrada:

{

"nome":"blusinha de ossos",

"descricao":"blusinha de ossos",

"modelo":"0205050",

"cod\_barras": 101012,

"porcentagem": 20,

"preco\_custo":15,

"preco\_venda":20,

"quantidade":10,

"foto":"foto.jpg",

"marca\_id":1,

"animal\_id":1,

"tamanho\_id":2,

"usuario\_id": 1

}

Resposta esperada: ‘Produto cadastrado’

Mensagens de erro:

Erro de integridade: ‘Produto já cadastrado’

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

Falta de dados: ‘Os dados do produto não foram inseridos’

**PUT ‘/produtos/<id>/alterar/’**

Função: alterar dados de produto

Exemplo de entrada:

{

"nome":"blusinha de ossos",

"descricao":"blusinha de ossos",

"modelo":"0205050",

"cod\_barras": 101012,

"porcentagem": 20,

"preco\_custo":15,

"preco\_venda":20,

"quantidade":10,

"foto":"foto.jpg",

"marca\_id":1,

"animal\_id":1,

"tamanho\_id":2,

"usuario\_id": 1

}

Resposta esperada: ‘Produto alterado’

Mensagens de erro:

Erro de integridade: ‘Não foi possível fazer a alteração’

Falta de dados: ‘Os dados do produto não foram inseridos’

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

**GET ‘/produtos/marcas/’**

Função: listar marcas

Resposta esperada: lista de marcas

Exemplo de resposta:

[

{

"id": 1,

"marca": "PetDog"

},

{

"id": 2,

"marca": "PetCat"

}

]

Mensagens de erro:

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

**POST ‘/produtos/marcas/’**

Funçao: cadastrar nova marca

Exemplo de entrada:

{

"marca":"Pet Fashions"

}

Resposta esperada: ‘Marca cadastrada’

Mensagens de erro:

Erro de integridade: ‘Marca já cadastrada’

Falta de dados: ‘Os dados da marca não foram inseridos’

**GET ‘/produtos/tamanhos/’**

Função: listar tamanhos

Resposta esperada: lista de tamanhos

Exemplo de resposta:

[

{

"id": 1,

"tamanho": "P"

},

{

"id": 2,

"tamanho": "M"

},

{

"id": 3,

"tamanho": "G"

}

]

Mensagens de erro:

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

**POST ‘/produtos/animais/’**

Funçao: cadastrar novo tamanho

Exemplo de entrada:

{

"tamanho":"G"

}

Resposta esperada: Animal cadastrado’

Mensagens de erro:

Erro de integridade: ‘Animal já cadastrado’

Falta de dados: ‘Os dados do animal não foram inseridos’

**GET ‘/produtos/animais/’**

Função: listar animais

Resposta esperada: lista de animais

Exemplo de resposta:

[

{

"animal": "cachorro",

"id": 1

},

{

"animal": "gato",

"id": 2

}

]

Mensagens de erro:

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

**POST ‘/produtos/animais/’**

Funçao: cadastrar novo animal

Exemplo de entrada:

{

"animal":"Porquinho"

}

Resposta esperada: ‘Animal cadastrado’

Mensagens de erro:

Erro de integridade: 'Animal já cadastrado’

Falta de dados: ‘Os dados de animal não foram inseridos’

**Pedidos:**

**GET ‘/pedidos/’**

Função: listar pedidos

Exemplo de resposta:

{

"pedidos": [

{

"cliente": {

"cpf": "3",

"email": "[fulano@det.al](mailto:fulano@det.al)",

"endereco": {

"bairro": "Jardim Europas",

"cep": "06000-000",

"cidade": "Osasco",

"id": 1,

"numero": "25 fundos",

"rua": "rua quinze",

"uf": "AC"

},

"id": 1,

"nome": "Fulanou"

},

"itens": [

{

"id": 1,

"pedido\_id": 1,

"preco": 120.0,

"produto\_id": 1,

"quantidade": 1,

"total": 120.0

},

{

"id": 2,

"pedido\_id": 1,

"preco": 120.0,

"produto\_id": 2,

"quantidade": 1,

"total": 120.0

}

],

"pedido": {

"cliente\_id": 1,

"data": "Tue, 04 May 2021 13:45:47 GMT",

"id": 1,

"observacao": "separado",

"situacao\_id": 1,

"usuario\_id": 2

},

"situacao": "recebido"

},

{

"cliente": {

"cpf": "3",

"email": "[fulano@det.al](mailto:fulano@det.al)",

"endereco": {

"bairro": "Jardim Europas",

"cep": "06000-000",

"cidade": "Osasco",

"id": 1,

"numero": "25 fundos",

"rua": "rua quinze",

"uf": "AC"

},

"id": 1,

"nome": "Fulanou"

},

"itens": [

{

"id": 3,

"pedido\_id": 2,

"preco": 120.0,

"produto\_id": 1,

"quantidade": 2,

"total": 240.0

},

{

"id": 4,

"pedido\_id": 2,

"preco": 120.0,

"produto\_id": 2,

"quantidade": 1,

"total": 120.0

}

],

"pedido": {

"cliente\_id": 1,

"data": "Tue, 04 May 2021 13:45:47 GMT",

"id": 2,

"observacao": "",

"situacao\_id": 2,

"usuario\_id": 2

},

"situacao": "concluido"

},

{

"cliente": {

"cpf": "10000000009",

"email": "[bill@kill.com](mailto:bill@kill.com)",

"endereco": {

"bairro": "Vila Ré",

"cep": "060020-161",

"cidade": "Barueri",

"id": 2,

"numero": "43",

"rua": "Rua 02",

"uf": "SP"

},

"id": 2,

"nome": "Bill"

},

"itens": [

{

"id": 5,

"pedido\_id": 3,

"preco": 120.0,

"produto\_id": 4,

"quantidade": 1,

"total": 120.0

}

],

"pedido": {

"cliente\_id": 2,

"data": "Tue, 04 May 2021 13:45:47 GMT",

"id": 3,

"observacao": "cheque sem fundos",

"situacao\_id": 3,

"usuario\_id": 3

},

"situacao": "cancelado"

}

]

}

Mensagens de erro:

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

**POST ‘/pedidos/’**

Função: cadastrar novo pedido

Exemplo de entrada:

{

"cliente\_id": 1,

"usuario\_id": 1,

"observacao": "mostruarios",

"itens": [

{

"produto\_id": 1,

"quantidade": 1

},

{

"produto\_id": 4,

"quantidade": 1

}

]

}

Mensagens de erro:

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

**PUT ‘/pedidos/<id>/situacao/’**

Função: atualizar a situação e observações de um pedido

Exemplo de entrada:

{

"situacao\_id": 1,

"observacao": "itens de mostruario",

}

Resposta esperada: ‘Pedido alterado’, ‘Pedido concluído’ ou ‘Pedido cancelado’

Mensagens de erro:

Falta de dados: ‘Os dados do pedido não foram inseridos’

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

Quantidade: ‘Quantidade insuficiente de produto em estoque’

**PUT ‘/pedidos/<id>/itens/’**

Função: atualiza a lista de produtos de um pedido

Exemplo de entrada:

{

"itens": [

{

"produto\_id": 1,

"quantidade": 1

},

{

"produto\_id": 4,

"quantidade": 2

}

]

}

Resposta esperada: ‘Pedido alterado’

Mensagens de erro:

Pedido não habilitado para alteração: ‘Pedido não pode ser alterado’

Falta de dados: ‘Os dados do pedido não foram inseridos’

Problemas de comunicação com o banco: ‘Não foi possível acessar os dados’

1. Os autores podem ser contatados respectivamente pelos seus correios eletrônicos:

   amanda.locatelli@aluno.faculdadeimpacta.com.br, gerson.correia@aluno.faculdadeimpacta.com.br, jordana.buranello@aluno.faculdadeimpacta.com.br, lauro.junior@aluno.faculdadeimpacta.com.br, maira.esteves@aluno.faculdadeimpacta.com.br, otoniel.filho@aluno.faculdadeimpacta.com.br, thaisa.brito@aluno.faculdadeimpacta.com.br. [↑](#footnote-ref-1)