Documentação do Projeto de CollectionStep

Visão Geral

Este é o registro e documentação de um projeto web de um projeto desenvolvido com Flask e SQLAlchemy. Este projeto contém algumas funcionalidades, como login, registro, cadastro de produtos, edição de produtos, pesquisa e exclusão de produtos. Abaixo estão as ferramentas e linguagens utilizadas, interface do usuário, as principais partes do registro do projeto, os arquivos HTML, configuração do banco de dados, as principais rotas e os modelos de dados.

Ferramentas Utilizadas

- VSCode
- Figma
- Mysql Worckbanch
- Trello

Linguagens Utilizadas

- Python
- SQL

Interface do Usuário

- HTML
- CSS
- Bootstrap

Estrutura do Projeto

Arquivos HTML

`cadastrar_produto.html`

Este arquivo contém formulários para registro de novos produtos. Inclua código de rastreamento, tipo de produto, data estimada de recebimento e campos de descrição.

edit.html

Este arquivo é usado para editar produtos existentes. Ele preenche previamente os campos com os dados do produto selecionados para edição.

index.html

Esta é a tela inicial do projeto, onde você pode visualizar os produtos cadastrados, pesquisar produtos pelo código de rastreamento e acessar as páginas de cadastro e edição.

Login.html

A tela de login permite que os usuários se autentiquem no sistema.

reg.html

Tela de cadastro para um novo usuário criar uma conta.

Arquivo de Configuração

config.py

Configurações essenciais para o projeto, incluindo a chave secreta para a sessão e a configuração do banco de dados usando SQLAlchemy.

```
SECRET_KEY = 'quatro&20'

SQLALCHEMY_DATABASE_URI =
'mysql+mysqlconnector://{usuario}:{senha}@{host}/{database}'.format(
    usuario='root',
    senha='cmb110205',
    host='localhost',
    database='collectionstep'
)

SQLALCHEMY_TRACK_MODIFICATIONS = False
```

Banco de Dados

Neste arquivo abaixo encontra-se a estrutura do banco de dados, desde a sua criação à inserção de dados na tabela

```
create database CollectionStep;

use CollectionStep;
```

```
create table usuarios(
    id_user int primary key auto_increment not null,
    name_user varchar(50) not null,
    mail varchar(50) not null,
    passW  varchar(20) not null
);

CREATE TABLE produtos (
    rastreio_pedido varchar(50),
    id_produto int PRIMARY KEY auto_increment not null,
    datareceb_prod varchar(20),
    tipo_prod varchar(50),
    desc_prod varchar(255),
    tamanho_prod boolean,
    status_prod boolean
);

INSERT INTO usuarios (name_user, mail, passW) VALUES
('Caio', 'caiomirandab@gmail.com', 'cmb110205');
```

Arquivo Principal

main.py

Este arquivo inicializa o aplicativo Flask, configura o banco de dados e define a rota principal.

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config.from_pyfile('config.py')

db = SQLAlchemy(app)

if __name__ == '__main__':
    app.run(debug=True)
```

Modelos de Dados

models.py

Define os modelos de dados para os produtos e usuários usando SQLAlchemy.

```
from main import db

class Produtos(db.Model):
    __tablename__ = 'produtos'
```

```
    id_produto = db.Column(db.Integer, primary_key=True,
autoincrement=True)
    rastreio_pedido = db.Column(db.String(50), nullable=False)
    datareceb_prod = db.Column(db.String(20), nullable=False)
    tipo_prod = db.Column(db.String(50), nullable=False)
    desc_prod = db.Column(db.String(255), nullable=False)

    def __repr__(self):
        return '<Produto %r>' % self.rastreio_pedido

class Usuarios(db.Model):
    __tablename__ = 'usuarios'
    id_user = db.Column(db.Integer, primary_key=True,
autoincrement=True)
    name_user = db.Column(db.String(50), nullable=False)
    mail = db.Column(db.String(50), nullable=False)
    passW = db.Column(db.String(20), nullable=False)

    def __repr__(self):
        return '<name %r>' % self.name_user
```

Rotas e Lógica do Aplicativo

views.py

Define as rotas e a lógica para login, registro, cadastro, edição, pesquisa e exclusão de produtos.

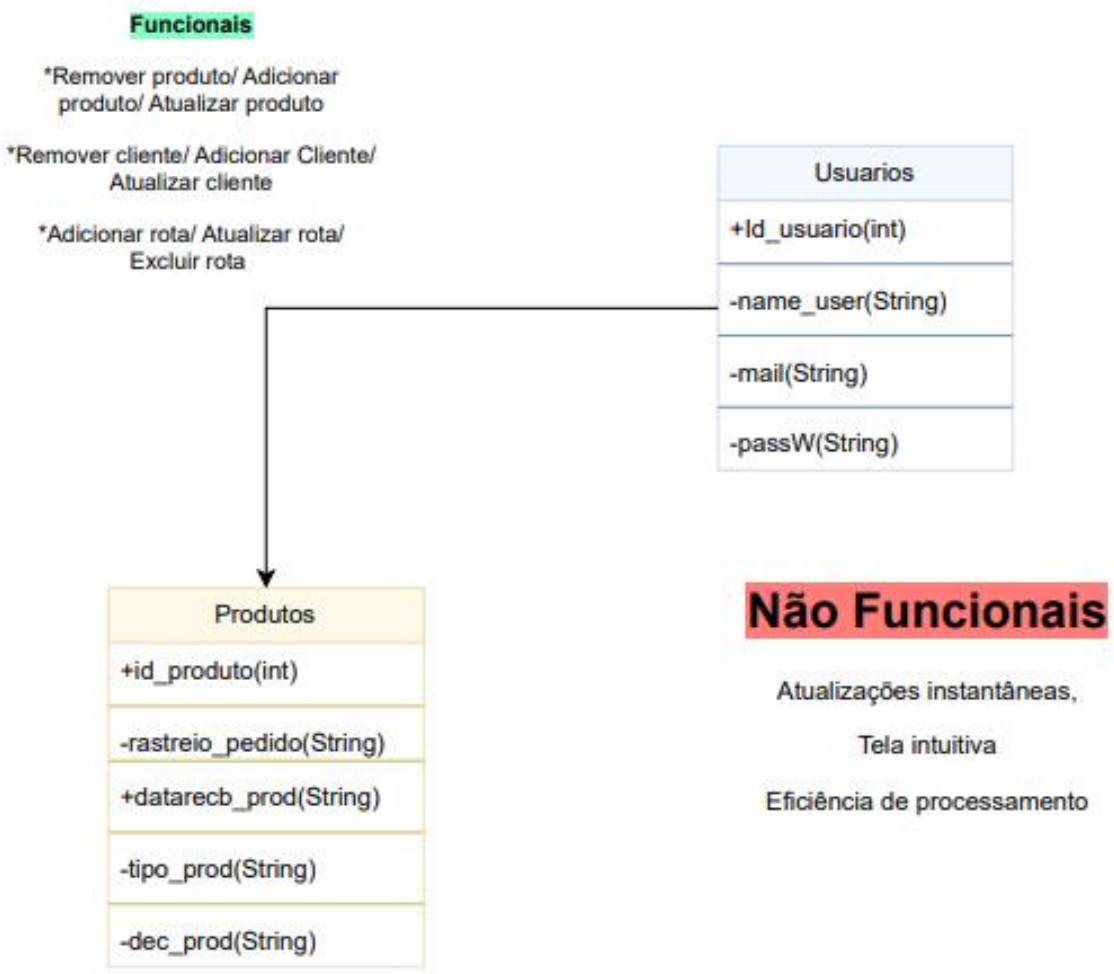
```
from flask import render_template, redirect, request, session, flash,
url_for
from models import *
from main import db, app

# Rota inicial
@app.route("/")
def paginaInicial():
    if 'User_Only' not in session or session['User_Only'] == None:
        return redirect('login')
    produtos_cadastrados =
Produtos.query.order_by(Produtos.id_produto).all()
    return render_template("index.html", Produtos =
produtos_cadastrados)

# Rota para cadastrar produto
@app.route('/cadastrar')
def cadastrar_produto():
    return render_template('cadastrar_produto.html')

# Rota para login
@app.route('/login')
def login():
   return render_template('Login.html')

# Rota para logar usuário
@app.route('/logar', methods=['POST'])
def logar():
    usuario =
Usuarios.query.filter_by(mail=request.form['txtEmail']).first()
    if usuario and request.form['txtSenha'] == usuario.passW:
```

```
            session['User_Only'] = usuario.name_user
            flash("Usuario Logado")
            return redirect(url_for('paginaInicial'))
        flash('Usuario/Senha Incorreta!')
        return redirect(url_for('login'))

# Rota para registro
@app.route('/registro')
def registro():
    return render_template('reg.html')

# Rota para adicionar produto
@app.route('/adicionar_produto', methods=['POST'])
def adicionar_produto():
    if 'User_Only' not in session or session['User_Only'] == None:
        return redirect('login')
    rastreio = request.form['txtRastreio']
    tipo = request.form['txtTipo']
    data = request.form['txtData']
    descricao = request.form['txtDescricao']

    produto_adicionado = Produtos(
        rastreio_pedido=rastreio,
        datareceb_prod=data,
        tipo_prod=tipo,
        desc_prod=descricao
    )

    db.session.add(produto_adicionado)
    db.session.commit()

    return redirect('/')

# Rota para editar produto
@app.route('/editar_produto/<int:produto_id>', methods=['GET'])
def editar_produto(produto_id):
    if 'User_Only' not in session or session['User_Only'] == None:
        return redirect('login')
    produto = Produtos.query.get(produto_id)
    if produto:
        return render_template('edit.html', produto=produto)
    flash("Produto não encontrado.")
    return redirect('/')

# Rota para salvar edição de produto
@app.route('/salvar_edicao/<int:produto_id>', methods=['POST'])
def salvar_edicao(produto_id):
    produto = Produtos.query.get(produto_id)
    if produto:
        produto.rastreio_pedido = request.form['txtRastreio']
        produto.tipo_prod = request.form['txtTipo']
        produto.datareceb_prod = request.form['txtData']
        produto.desc_prod = request.form['txtDescricao']
        db.session.commit()
        flash("Produto editado com sucesso.")
    else:
        flash("Produto não encontrado.")
    return redirect('/')

# Rota para pesquisar produto por rastreio
@app.route('/pesquisar_rastreio', methods=['POST'])
```

```
def pesquisar_rastreio():
    rastreio = request.form['rastreio']
    if rastreio:
        produto =
Produtos.query.filter(Produtos.rastreio_pedido.ilike(f'%{rastreio}%'))
.first()
        if produto:
            return redirect(url_for('editar_produto',
produto_id=produto.id_produto))
        flash('Produto não encontrado com o número de rastreamento
informado.')
    else:
        flash('Informe o número de rastreamento para realizar a
pesquisa.')
    return redirect(url_for('paginaInicial'))

# Rota para apagar produto
@app.route('/apagar_produto/<int:produto_id>', methods=['GET'])
def apagar_produto(produto_id):
    produto = Produtos.query.get(produto_id)
    if produto:
        db.session.delete(produto)
        db.session.commit()
        flash("Produto apagado com sucesso.")
    else:
        flash("Produto não encontrado.")
    return redirect('/')

# Rota para logout
@app.route('/logout', methods=['POST'])
def logout():
    session.pop('User_Only', None)
    return redirect(url_for('login'))

app.run(debug=True)
```

Diagrama UML

Abaixo encontra-se o diagrama UML (Unified Modeling Language) que é uma representação visual utilizada para especificar, visualizar, construir e documentar os artefatos de um sistema.



Considerações Finais

Este projeto demonstra um aplicativo básico de gerenciamento de produtos com funcionalidade de autenticação e operações CRUD (Criar, Ler, Atualizar, Excluir) usando Flask e SQLAlchemy. A estrutura do projeto é modular com separação clara entre configuração, modelo de dados, roteamento e templates. A documentação aqui fornecida deve ajudar na compreensão e manutenção do projeto, bem como adicionar novos recursos conforme necessário.

Link do projeto: https://github.com/CollectionStep/CollectionStep.git