Wprowadzenie i opis:

System zarządzania magazynem. System umożliwia dodawanie produktów do magazynu, ich wyszukiwanie, usuwanie, aktualizację stanu magazynowego oraz generowanie raportów. System posiada proste GUI.

Autorzy:

Marcin Milewski s25278

Użytkowanie:

Dokumentacja Techniczna:

Klasa GUI:

Atrybuty:

main\_window (tkinter.Tk): Główne okno interfejsu graficznego.

top\_frame (tkinter.Frame): Górna ramka zawierająca przyciski i funkcjonalność wyszukiwania.

middle\_frame (tkinter.Frame): Środkowa ramka zawierająca wyniki wyszukiwania.

bottom\_frame (tkinter.Frame): Dolna ramka zawierająca pola wprowadzania do dodawania/aktualizowania rekordów.

entry\_fields (list): Lista pól wprowadzania danych dla rekordów.

attribute\_checkbuttons (list): Lista checkbuttonów do wyboru atrybutów.

attribute\_var (tkinter.StringVar): Zmienna przechowująca wybrany atrybut.

magazine\_var (tkinter.BooleanVar): Zmienna przechowująca wybór filtrowania według dostępności w magazynie.

Metody:

clearAllValues: Czyści wszystkie pola wprowadzania danych w interfejsie graficznym.

add: Dodaje nowy rekord do bazy danych na podstawie pól wprowadzania danych.

throwError: Sprawdza poprawność wprowadzanych danych i wyświetla ewentualne komunikaty o błędach.

delete: Usuwa rekord z bazy danych na podstawie pól wprowadzania danych.

update: Aktualizuje rekord w bazie danych na podstawie pól wprowadzania danych.

search: Wykonuje wyszukiwanie w bazie danych na podstawie wprowadzonego tekstu i wybranych atrybutów.

on\_tree\_double\_click: Obsługuje zdarzenie podwójnego kliknięcia na element wyniku wyszukiwania.

createRaport: Tworzy plik raportu z aktualnymi rekordami bazy danych.

def \_\_init\_\_(self):

Inicjalizuje interfejs graficzny, tworząc główne okno i konfigurując ramki oraz widżety.

def clearAllValues(self):

Czyści wszystkie pola wprowadzania danych w interfejsie graficznym.

def add(self):

Dodaje nowy rekord do bazy danych na podstawie pól wprowadzania danych.

def throwError(self, data):

Sprawdza poprawność wprowadzanych danych i wyświetla ewentualne komunikaty o błędach.

Args:

data (list): Dane do sprawdzenia.

Returns:

bool: True, jeśli wystąpiły błędy walidacji, False w przeciwnym razie.

def delete(self):

Usuwa rekord z bazy danych na podstawie podanego identyfikatora (ID).

def update(self):

Aktualizuje rekord w bazie danych na podstawie podanego identyfikatora (ID).

def search(self):

Wyszukuje rekordy w bazie danych na podstawie wprowadzonego słowa kluczowego i opcji sortowania.

Wyświetla wyniki wyszukiwania w postaci tabeli.

def on\_tree\_double\_click(self, event):

Wywoływana po podwójnym kliknięciu na rekord w tabeli.

Pobiera informacje o wybranym rekordzie i wyświetla je w polach tekstowych.

def createRaport(self):

Tworzy raport na podstawie wybranych opcji.

Pobiera wybrane opcje od użytkownika i generuje raport w wybranym formacie.

Klasa database:

Klasa `database` odpowiada za zarządzanie bazą danych produktów.

Metody:

- \_\_init\_\_(): Konstruktor klasy, tworzy bazę danych, jeśli nie istnieje.

- createDatabase(): Tworzy tabelę Produkt i wstawia przykładowe dane.

- add(): Dodaje nowy produkt do bazy danych.

- update(): Aktualizuje informacje o produkcie w bazie danych.

- delete(): Usuwa produkt z bazy danych.

- search(): Wyszukuje produkty w bazie danych.

Atrybuty:

Brak.

def \_\_init\_\_(self):

Konstruktor klasy, tworzy bazę danych, jeśli nie istnieje.

Argumenty:

Brak.

Zwraca:

None.

def createDatabase(self):

Tworzy tabelę Produkt w bazie danych i wstawia przykładowe dane.

Argumenty:

Brak.

Zwraca:

None.

@staticmethod

def add(Id=None, Nazwa=None, Opis=None, Cena=None, Miara\_Ilosci=None, Ilosc=None):

Dodaje nowy produkt do bazy danych.

Argumenty:

- Id (int): Identyfikator produktu. Jeśli None, zostanie automatycznie wygenerowany.

- Nazwa (str): Nazwa produktu.

- Opis (str): Opis produktu.

- Cena (float): Cena produktu.

- Miara\_Ilosci (str): Jednostka miary ilości produktu.

- Ilosc (float): Dostępna ilość produktu.

Zwraca:

- 'Ok' (str): Dodanie produktu powiodło się.

- 'Not' (str): Nieprawidłowa wartość ilości produktu.

@staticmethod

def update(Id, Nazwa=None, Opis=None, Cena=None, Miara\_Ilosci=None, Ilosc=None):

Aktualizuje informacje o produkcie w bazie danych.

Argumenty:

- Id (int): Identyfikator produktu.

- Nazwa (str): Nowa nazwa produktu. Jeśli None, nazwa nie zostanie zmieniona.

- Opis (str): Nowy opis produktu. Jeśli None, opis nie zostanie zmieniony.

- Cena (float): Nowa cena produktu. Jeśli None, cena nie zostanie zmieniona.

- Miara\_Ilosci (str): Nowa jednostka miary ilości produktu. Jeśli None, jednostka miary ilości nie zostanie zmieniona.

- Ilosc (float): Nowa dostępna ilość produktu. Jeśli None, ilość nie zostanie zmieniona.

Zwraca:

- 'Ok' (str): Aktualizacja produktu powiodła się.

- 'Not1' (str): Nieprawidłowa wartość ilości produktu.

- 'Not2' (str): Brak identyfikatora produktu.

@staticmethod

def delete(Id):

Usuwa produkt z bazy danych.

Argumenty:

- Id (int): Identyfikator produktu.

Zwraca:

- 'Ok' (str): Usunięcie produktu powiodło się.

- 'Not' (str): Brak identyfikatora produktu.

@staticmethod

def search(keyword=None, sort\_by=None, include\_zero=False):

Wyszukuje produkty w bazie danych.

Argumenty:

- keyword (str): Słowo kluczowe do wyszukania. Jeśli None, zostaną zwrócone wszystkie produkty.

- sort\_by (str): Pole, według którego mają zostać posortowane wyniki. Jeśli None, wyniki nie będą sortowane.

- include\_zero (bool): Określa, czy wyniki powinny zawierać produkty o ilości równej zero.

Zwraca:

- results (list): Lista wyników zawierających krotki (Id, Nazwa, Opis, Cena, Ilosc, Miara\_Ilosci).

Klasa Main:

Klasa Main jest odpowiedzialna za inicjalizację obiektów database i GUI oraz uruchomienie głównego programu.