**Отчет по самостоятельной работе №3**

**по дисциплине МДК 01.02 “Инструментальные средства разработки программного обеспечения”.**

Выполнил: студент

группы 319

Меркушин Евгений Романович

Дата 17.11.2024

**Цель работы**

Изучение и применение системы контроля версий на практике, включая настройку типов импортируемых файлов, путей, фильтров и других параметров для эффективного управления репозиториями.

**Основная структура задания**

Для создания базы данных интернет-магазина можно использовать SQLite, так как это простая и удобная для работы СУБД, которая не требует установки отдельного сервера баз данных.

# Шаг 1: Создание таблицы товаров

Предположим, что у нас есть таблица products, где хранятся данные о товарах. Вот пример кода на Python с использованием библиотеки sqlite3:

import sqlite3

# Подключение к базе данных (если файл не существует, он будет создан)

conn = sqlite3.connect('internet\_shop.db')

cursor = conn.cursor()

# Создание таблицы products

cursor.execute(''' CREATE TABLE IF NOT EXISTS products ( id INTEGER PRIMARY KEY, name TEXT NOT NULL, price REAL NOT NULL, description TEXT, image BLOB ) ''')

# Сохранение изменений

conn.commit()

# Шаг 2: Определение типов файлов

Теперь определим список допустимых типов файлов для изображений товаров. Допустимые форматы могут быть, например, .jpg, .png, .gif. Мы можем хранить эти файлы непосредственно в базе данных в виде бинарных данных (BLOB), но чаще их хранят на диске, а в базе данных указывают путь к файлу.

Пример функции проверки типа файла по его расширению:

def is\_valid\_file\_type(filename):

valid\_types = ['.jpg', '.jpeg', '.png', '.gif']

return any([filename.lower().endswith(ext) for ext in valid\_types])

# Пример использования

file\_name = 'product\_image.jpg'

if is\_valid\_file\_type(file\_name):

print("Файл допустимого типа.")

else:

print("Недопустимый тип файла.")

# Полный код

Вот полный пример программы, включающий создание базы данных и проверку типов файлов:

import sqlite3

# Подключение к базе данных (если файл не существует, он будет создан)

conn = sqlite3.connect('internet\_shop.db')

cursor = conn.cursor()

# Создание таблицы products

cursor.execute(''' CREATE TABLE IF NOT EXISTS products ( id INTEGER PRIMARY KEY, name TEXT NOT NULL, price REAL NOT NULL, description TEXT, image\_path TEXT ) ''')

# Функция проверки типа файла

def is\_valid\_file\_type(filename):

valid\_types = ['.jpg', '.jpeg', '.png', '.gif']

return any([filename.lower().endswith(ext) for ext in valid\_types])

# Добавление нового продукта

def add\_product(name, price, description, image\_path):

if not is\_valid\_file\_type(image\_path):

raise ValueError(f"Недопустимый тип файла: {image\_path}")

cursor.execute(''' INSERT INTO products (name, price, description, image\_path) VALUES (?, ?, ?, ?) ''', (name, price, description, image\_path))

# Сохраняем изменения

conn.commit()

# Пример добавления продукта

add\_product(

"Товар 1",

1000.00,

"Описание товара 1",

"images/product1.jpg"

)

print("Продукт успешно добавлен!")

# Закрытие соединения с базой данных

conn.close()

Этот скрипт создает таблицу products в базе данных internet\_shop.db, добавляет новый продукт и проверяет корректность типа файла перед добавлением записи.

Дополним код, чтобы добавить несколько тестовых файлов и убедиться, что недопустимые типы файлов действительно исключаются при попытке добавить товар.

# Обновленный код

В этом примере мы создадим функцию для генерации случайного набора продуктов с разными типами файлов, включая допустимые и недопустимые. Затем попробуем добавить каждый из этих продуктов в базу данных и убедимся, что те продукты, у которых тип файла недопустим, вызывают исключение.

import sqlite3

import random

from string import ascii\_letters

# Подключение к базе данных (если файл не существует, он будет создан)

conn = sqlite3.connect('internet\_shop.db')

cursor = conn.cursor()

# Создание таблицы products

cursor.execute(''' CREATE TABLE IF NOT EXISTS products ( id INTEGER PRIMARY KEY, name TEXT NOT NULL, price REAL NOT NULL, description TEXT, image\_path TEXT ) ''')

# Функция проверки типа файла

def is\_valid\_file\_type(filename):

valid\_types = ['.jpg', '.jpeg', '.png', '.gif']

return any([filename.lower().endswith(ext) for ext in valid\_types])

# Генерация случайной строки

def generate\_random\_string(length=10):

return ''.join(random.choice(ascii\_letters) for \_ in range(length))

# Генерация случайного числа

def generate\_random\_price():

return round(random.uniform(50, 500), 2)

# Генерация случайного описания

def generate\_random\_description():

return f'Описание товара: {generate\_random\_string()}'

# Генерация случайного имени файла

def generate\_random\_filename(valid=True):

extensions = ['.jpg', '.jpeg', '.png', '.gif'] if valid else ['.txt', '.pdf', '.exe']

return f'{generate\_random\_string()}{random.choice(extensions)}'

# Добавление нового продукта

def add\_product(name, price, description, image\_path):

if not is\_valid\_file\_type(image\_path):

raise ValueError(f"Недопустимый тип файла: {image\_path}")

cursor.execute(''' INSERT INTO products (name, price, description, image\_path) VALUES (?, ?, ?, ?) ''', (name, price, description, image\_path))

# Сохраняем изменения

conn.commit()

# Генерация тестовых данных

for i in range(5):

try:

product\_name = f"Тестовый товар {i+1}"

product\_price = generate\_random\_price()

product\_desc = generate\_random\_description()

# Генерируем случайное имя файла (с вероятностью 50% допустимое/недопустимое расширение)

valid\_file = bool(random.getrandbits(1)) # True or False

file\_name = generate\_random\_filename(valid\_file)

print(f"Попытка добавить товар '{product\_name}' с файлом '{file\_name}':")

add\_product(product\_name, product\_price, product\_desc, file\_name)

print("Успешно добавлено!\n")

except ValueError as e:

print(f"Ошибка: {e}\n")

# Закрытие соединения с базой данных

conn.close()

# Объяснение обновленного кода

1. **Генерация случайных данных**:
   * Используются функции для генерации случайных строк, чисел и описаний.
   * Функция generate\_random\_filename() генерирует случайные имена файлов с допустимыми и недопустимыми расширениями.
2. **Цикл для тестирования**:
   * В цикле создается пять тестовых продуктов, для каждого из которых случайно выбирается допустимость расширения файла.
   * Если файл имеет недопустимое расширение, то вызывается исключение ValueError.

# Результат выполнения

При запуске этого скрипта вы увидите примерно такой вывод:

Попытка добавить товар 'Тестовый товар 1' с файлом 'qUodjWmQ.pdf':

Ошибка: Недопустимый тип файла: qUodjWmQ.pdf

Попытка добавить товар 'Тестовый товар 2' с файлом 'pHxLmOwD.png':

Успешно добавлено!

Попытка добавить товар 'Тестовый товар 3' с файлом 'kZKmuoSw.exe':

Ошибка: Недопустимый тип файла: kZKmuoSw.exe

Попытка добавить товар 'Тестовый товар 4' с файлом 'FvczUpVb.gif':

Успешно добавлено!

Попытка добавить товар 'Тестовый товар 5' с файлом 'AaxgXhWh.txt':

Ошибка: Недопустимый тип файла: AaxgXhWh.txt

Как видно из вывода, попытки добавить товары с недопустимыми типами файлов завершились ошибкой, тогда как допустимые файлы были успешно добавлены в базу данных.

Начало формы