Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №23
дисциплины «Основы программной инженерии»

Выполнил:
Плугатырев Владислав Алексеевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил Воронкин Роман Александрович
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Взаимодействие с базами данных SQLite3 с помощью языка программирования Python

Ход работы.

1. Создание репозитория.

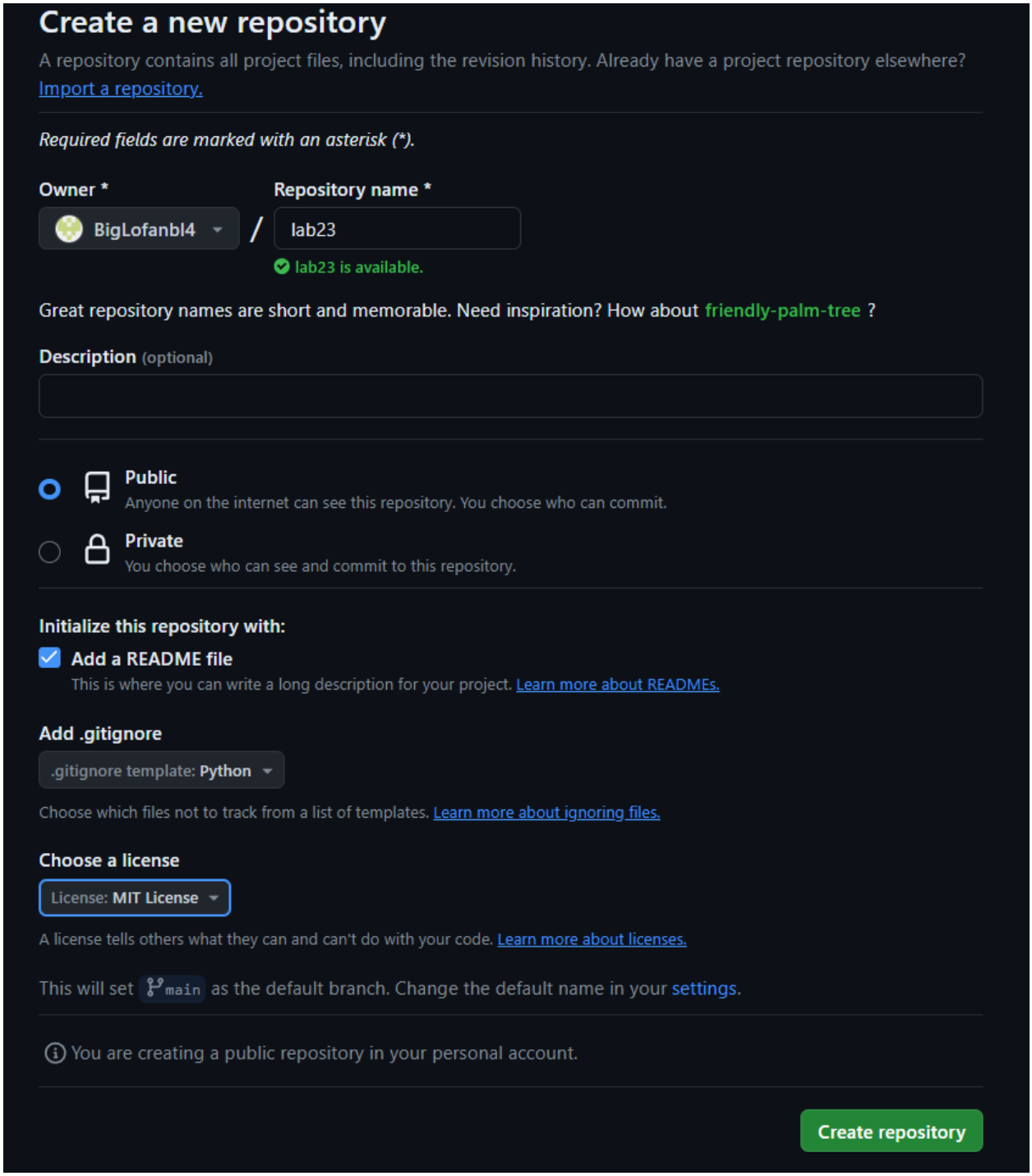

Рисунок 1.1 – Создание репозитория

2. Выполнение примера.

```
41
42
43  def create_db(database_path: Path) -> None:
44      """
45      Создать базу данных.
46      """
47      conn = sqlite3.connect(database_path)
48      cursor = conn.cursor()
49      # Создать таблицу с информацией о должностях.
50      cursor.execute(
51          """
52          CREATE TABLE IF NOT EXISTS posts (
53              post_id INTEGER PRIMARY KEY AUTOINCREMENT,
54              post_title TEXT NOT NULL
55          )
56          """
57      )
58      # Создать таблицу с информацией о работниках.
59      cursor.execute(
60          """
61          CREATE TABLE IF NOT EXISTS workers (
62              worker_id INTEGER PRIMARY KEY AUTOINCREMENT,
63              worker_name TEXT NOT NULL,
64              post_id INTEGER NOT NULL,
65               worker_year INTEGER NOT NULL,
66              FOREIGN KEY(post_id) REFERENCES posts(post_id)
67                  )
68          """
69      )
70      conn.close()
71
72
73  def add_worker(database_path: Path, name: str, post: str, year: int) -> None:
74      """
75      Добавить работника в базу данных.
76      """
77
78      conn = sqlite3.connect(database_path)
79      cursor = conn.cursor()
80      # Получить идентификатор должности в базе данных.
81      # Если такой записи нет, то добавить информацию о новой должности.
82      cursor.execute(
83          """
84          SELECT post_id FROM posts WHERE post_title = ?
85          """,
86          (post,),
```

Рисунок 2.1 – Код примера

3. Выполнение первого задания: для своего варианта лабораторной работы 2.17 необходимо реализовать хранение данных в базе данных SQLite3. Информация в базе данных должна храниться не менее чем в двух таблицах.

```python
def create_db(database_path: Path) -> None:
    """
    Создать базу данных.
    """
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()

    # Создать таблицу с информацией о зодиаках.
    cursor.execute(
        """
        CREATE TABLE IF NOT EXISTS zodiacs (
            zodiac_id INTEGER PRIMARY KEY AUTOINCREMENT,
            zodiac_title TEXT NOT NULL
        )
        """
    )

    # Создание таблицы людей
    cursor.execute(
        """
        CREATE TABLE IF NOT EXISTS people (
            person_id INTEGER PRIMARY KEY AUTOINCREMENT,
            surname TEXT NOT NULL,
            name TEXT NOT NULL,
            zodiac_id INTEGER NOT NULL,
            birthday Date NOT NULL,
            FOREIGN KEY(zodiac_id) REFERENCES zodiacs(zodiac_id)
        )
    """
    )

    conn.commit()
    conn.close()


def add_person(
    database_path: Path,
    surname: str,
    name: str,
    zodiac: str,
    birthday: datetime,
) -> None:
    """
    Добавляет работника в базу данных.
    """
    conn = sqlite3.connect(database_path)
```

Рисунок 3.1 – Код программы

```
(lab23) C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\23\lab23>python ind.py add --s="Plugatyrev" --n="Vlad" --z="Capricorn" --b="12.01.2005"

(lab23) C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\23\lab23>python ind.py display
+-----+-----------------------------+-----------------------------+----------------------+----------------------+
| №   |          Фамилия            |             Имя             |     Знак зодиака     |    Дата рождения     |
+-----+-----------------------------+-----------------------------+----------------------+----------------------+
|   1 | Plugatyre                   | Vladislav                   | Capricorn            |           12.01.2005 |
|   2 | Plugatyrev                  | Vlad                        | Capricorn            |           12.01.2005 |
+-----+-----------------------------+-----------------------------+----------------------+----------------------+
```

Рисунок 3.2 – Вывод программы

4. Выполнение задания повышенной сложности: самостоятельно изучите работу с пакетом python-psycopg2 для работы с базами данных PostgreSQL. Для своего варианта лабораторной работы 2.17 необходимо реализовать возможность хранения данных в базе данных СУБД PostgreSQL. Информация в базе данных должна храниться не менее чем в двух таблицах.

```python
def create_db() -> None:
    """
    Создать базу данных.
    """
    conn = psycopg2.connect(
        database=DATABASE, user=USER, password=PASSWORD, host=HOST, port=PORT
    )
    cursor = conn.cursor()

    # Создать таблицу с информацией о зодиаках.
    cursor.execute(
        """
        CREATE TABLE IF NOT EXISTS zodiacs (
            zodiac_id SERIAL PRIMARY KEY,
            zodiac_title TEXT NOT NULL
        )
        """
    )

    # Создание таблицы людей
    cursor.execute(
        """
        CREATE TABLE IF NOT EXISTS people (
            person_id SERIAL PRIMARY KEY,
            surname TEXT NOT NULL,
            name TEXT NOT NULL,
            zodiac_id INTEGER NOT NULL REFERENCES zodiacs(zodiac_id),
            birthday TEXT NOT NULL
        )
    """
    )

    conn.commit()
    conn.close()
```

Рисунок 4.1 – Код программы

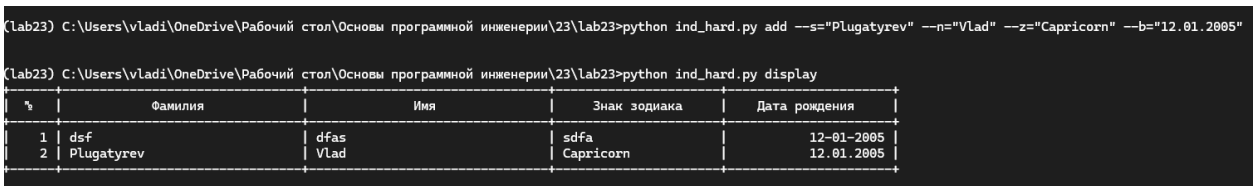

Рисунок 4.2 – Вывод программы

Ответы на контрольные вопросы

1. Назначение модуля sqlite3:

Модуль «sqlite3» в Python предоставляет возможности для работы с базой данных SQLite. Он позволяет выполнять SQL-запросы, управлять транзакциями, взаимодействовать с базой данных SQLite без необходимости установки отдельного сервера БД. SQLite - это СУБД, которая хранит всю базу данных в одном файле, что делает её идеальной для небольших проектов, тестирования и обучения.

2. Соединение с базой данных SQLite3 и курсор базы данных:

Соединение с базой данных SQLite3 устанавливается с помощью функции «sqlite3.connect(path_to_db)», где «path_to_db» - путь к файлу базы данных. Если файл по указанному пути не существует, SQLite создаст новую базу данных.

Курсор базы данных используется для выполнения SQL-запросов и получения результатов. Он создается методом «cursor()» объекта соединения.

3. Подключение к базе данных в оперативной памяти:

Чтобы подключиться к базе данных SQLite3, хранящейся в оперативной памяти компьютера, используйте строку «":memory:"« в качестве аргумента функции «sqlite3.connect()»: «conn = sqlite3.connect(":memory:")».

4. Корректное завершение работы с базой данных SQLite3:

Для корректного завершения работы с SQLite необходимо закрыть соединение с базой данных, используя метод «close()» объекта соединения. Если были сделаны изменения в базе данных, которые требуется сохранить, перед закрытием соединения проведите фиксацию транзакции с помощью метода «commit()».

5. Вставка данных в таблицу:

Вставка данных осуществляется с помощью SQL-запроса «INSERT INTO имя_таблицы (поле1, поле2, ...) VALUES (значение1, значение2, ...)». Для выполнения используется метод «execute()» курсора, в который передается строка запроса и кортеж значений.

6. Обновление данных таблицы:

Для обновления данных используется SQL-запрос «UPDATE имя_таблицы SET поле1 = значение1, поле2 = значение2 WHERE условие». Выполнение запроса аналогично вставке.

7. Выборка данных из базы данных:

Выборка данных осуществляется с помощью SQL-запроса «SELECT * FROM имя_таблицы WHERE условие». Результаты выборки можно получить, используя методы «fetchone()», «fetchall()» или итерируясь по курсору после выполнения запроса.

8. Назначение метода rowcount:

Метод «rowcount» возвращает количество строк, затронутых последним выполненным SQL-запросом (INSERT, UPDATE, DELETE). Для SELECT-запросов поведение «rowcount» не определено и может варьироваться в зависимости от используемой библиотеки и её версии.

9. Получение списка всех таблиц:

Список всех таблиц можно получить, выполнив SQL-запрос в системную таблицу «sqlite_master»: «SELECT name FROM sqlite_master WHERE type ='table'«.

10. Проверка существования таблицы:

Проверить существование таблицы можно, выполнив запрос на выборку из таблицы «sqlite_master» с условием на имя таблицы: «SELECT name FROM sqlite_master WHERE type='table' AND name='имя_таблицы'«. Если запрос возвращает результат, таблица существует.

11. Массовая вставка данных:

Для массовой вставки данных используется метод «executemany()», который принимает SQL-запрос и последовательность кортежей с данными для вставки.

12. Работа с датой и временем:

SQLite поддерживает работу с датой и временем через строковый формат (например, "YYYY-MM-DD HH:MM:SS"). Функции «date()», «time()», «datetime()», «julianday()» и «strftime()» можно использовать для манипуляций с датой и временем при выполнении SQL-запросов.