

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 РАБОТЕ №2.21
дисциплины «Основы кроссплатформенного программирования»

Выполнил:

Баканов Артем Вадимович

2 курс, группа ИТС-б-о-22-1,

11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:

Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Цель работы: исследовать базовые возможности системы управления базами данных SQLite3

Порядок выполнения работы:

1. Создала новый репозиторий и клонировала его на свой компьютер.

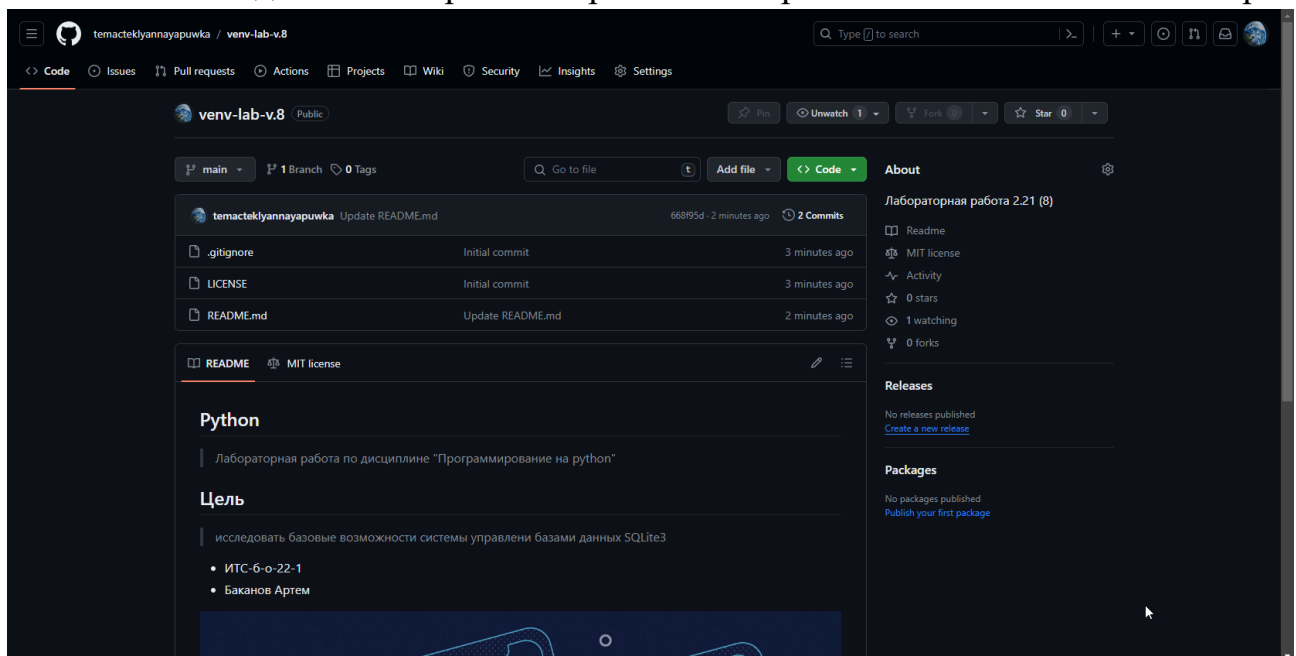


Рисунок 1 – Создан новый репозиторий

2. Клонировал репозиторий на свой компьютер. В ходе данной лабораторной работы работала с моделью ветвления git-flow.

```
PS D:\study\2course\pythonUnic\venv-lab> git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/study/2course/pythonUnic/venv-lab/.git/hooks]
PS D:\study\2course\pythonUnic\venv-lab>
```

Рисунок 2 – Клонирование и модель ветвления git-flow

3. Создал виртуальное окружение Anaconda с именем репозитория.

```
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate venv
#
# To deactivate an active environment, use
#
#     $ conda deactivate
#
(base) PS D:\study\2course\pythonUnic\venv-lab\venv> conda activate venv
(venv) PS D:\study\2course\pythonUnic\venv-lab\venv> conda install django, pandas
Collecting package metadata (current_repodata.json): \ DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): r
repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
/ DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-64/current_repodata.json HTTP/1.1" 304
0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/noarch/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/noarch/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/noarch/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/win-64/current_repodata.json HTTP/1.1" 304 0
- DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/win-64/current_repodata.json HTTP/1.1" 304 0
done
Solving environment: \ _
```

Рисунок 3 – Создание виртуального окружения

Выполнение индивидуального задания

С помощью команд sqlite3 создадим базу данных, а затем в ней создадим таблицы

Добавим магазины и товары в них с помощью команд консоли, а затем отобразим содержимое таблицы

```
+-----+-----+-----+-----+
| 1 | 234 | maslo | magnit |
+-----+-----+-----+-----+
| 2 | 2334 | kolbasa | pyterka |
+-----+-----+-----+-----+
| 3 | 233 | kolbasa | lenta |
+-----+-----+-----+-----+
| 4 | 453 | nokia | Okey |
+-----+-----+-----+-----+
PS C:\Users\maxim\YandexDisk\Лабы 4 семестр\Технологии программирования\2.21>
```

Рисунок 4 – Содержимое таблицы с магазинами

При помощи команды выбора магазина найдем нужный магазин

```
+-----+-----+-----+-----+
| No | Название | Товар | Цена |
+-----+-----+-----+-----+
| 1 | lenta | kolbasa | 233 |
+-----+-----+-----+-----+
PS C:\Users\maxim\YandexDisk\Лабы 4 семестр\Технологии программирования\2.21>
```

Рисунок 5 – Нужный магазин найден

Ответы на контрольные вопросы

1. Непосредственно модуль `sqlite3` – это API к СУБД SQLite. Своего рода адаптер, который переводит команды, написанные на Питоне, в команды, которые понимает SQLite. Как и наоборот, доставляет ответы от SQLite в python-программу.

2. Для взаимодействия с базой данных SQLite3 в Python необходимо создать объект `cursor`. Вы можете создать его с помощью метода `cursor()`. Курсор SQLite3 – это метод объекта соединения. Для выполнения инструкций SQLite3 сначала устанавливается соединение, а затем создается объект курсора с использованием объекта соединения

3. При создании соединения с SQLite3 автоматически создается файл базы данных, если он еще не существует. Этот файл базы данных создается на диске, мы также можем создать базу данных в оперативной памяти с помощью функции `:memory:` with the connect. Такая база данных называется базой данных в памяти.

4. С помощью команды закрытия `close()`.

5. Чтобы вставить данные в таблицу, используется оператор `INSERT INTO`.

6. Чтобы обновить данные в таблице, просто создайте соединение, затем создайте объект курсора с помощью соединения и, наконец, используйте оператор `UPDATE`.

7. Оператор `SELECT` используется для выбора данных из определенной таблицы. Если вы хотите выбрать все столбцы данных из таблицы, вы можете использовать звездочку (*).

8. SQLite3 `rowcount` используется для возврата количества строк, которые были затронуты или выбраны последним выполненным SQL-запросом.

9. Чтобы перечислить все таблицы в базе данных SQLite3, вы должны запросить данные из таблицы `sqlite_master`, а затем использовать `fetchall()` для получения результатов из инструкции `SELECT`

10. При создании таблицы мы должны убедиться, что она еще не существует. Аналогично, при удалении/удалении таблицы она должна существовать. Чтобы проверить, не существует ли таблица уже, мы используем `IF NOT EXISTS` с оператором `CREATE TABLE` следующим образом.

11. Метод `executemany` можно использовать для вставки нескольких строк одновременно.

12. В базе данных Python SQLite3 мы можем легко хранить дату или время, импортируя модуль `datetime`. Следующие форматы являются наиболее часто используемыми форматами для `datetime`:

Вывод: в результате выполнения работы были приобретены навыки по работе с базовыми возможностями системы управления базами данных SQLite3.

