|  |  |  |  |
| --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  ФГБОУ ВО «Пермский государственный национальный исследовательский университет» | | | |
|  | | | |
| ОТЧЕТ  по лабораторной работе №1  «Создание клиентского приложения по выполнению CRUD-операции на популярном информационном ресурсе»  по дисциплине «Технологии разработки распределенных приложений» | | | |
| Работу выполнил студент группы ФИТ-4-НБ 4 курса:  Бродников Вадим  «\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2021 г. |  | | Проверил к.ф.-м.н., доцент кафедры ПМИ  Деменев Алексей Геннадьевич  «\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2021 г. |
|  | |  | |
| Пермь 2021 | | | |

Оглавление

[Термины и определения 3](#_Toc67076138)

[Введение 4](#_Toc67076139)

[Постановка задачи 5](#_Toc67076140)

[1 Описание приложения. Введение в предметную область 6](#_Toc67076141)

[1.1 Архитектура системы 6](#_Toc67076142)

[2 Алгоритм работы приложения 7](#_Toc67076143)

[3.1 Архитектура клиента 9](#_Toc67076144)

[3.2 Архитектура сервера 9](#_Toc67076145)

[3.3 Методы коммуникаций компонентов системы 9](#_Toc67076146)

[3.4 Структура передаваемых данных 10](#_Toc67076147)

[3.5 Язык программирования 10](#_Toc67076148)

[3.6 Исходный код 10](#_Toc67076149)

[4 Получение токена доступа для приложения 11](#_Toc67076150)

[5 Сценарии использования 14](#_Toc67076151)

[6 Тестирование 16](#_Toc67076152)

[Список файлов 23](#_Toc67076153)

[Заключение 24](#_Toc67076154)

[Перечень использованных источников 25](#_Toc67076155)

**Термины и определения**

В настоящем отчете применяют следующие термины с соответствующими определениями:

**Сниппет** (англ. snippet — отрывок, фрагмент) — фрагмент исходного текста или кода программы, применяемый в поисковых системах, текстовых редакторах и средах разработки.

**CLI (Command Line Interface)** [16] – это программа в командной строке, которая читает вводимые команды и выполняет запрошенные действия.

**Токен (access\_token)** [12] – это строка из цифр и латинских букв, которую вы передаёте на сервер вместе с запросом. По сути ключ доступа — это подпись того, от чьего лица совершается запрос.

**Ключ доступа** [12][13] — своего рода «подпись» пользователя в Вашем приложении. Он сообщает серверу, от имени какого пользователя осуществляются запросы, и какие права доступа он выдал Вашему приложению.

**JSON** [15] – простой, основанный на использовании текста, способ хранить и передавать структурированные данные.

**HTTP** [17]– широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

**OAuth 2.0** [14] — протокол авторизации, позволяющий выдать одному сервису (приложению) права на доступ к ресурсам пользователя на другом сервисе.

**Введение**

**Цель:**Изучение возможностей использования API современных популярных информационных ресурсов внешними приложениями.

**Задача:** реализовать приложение со следующими требованиями

1. Разработанное приложение должно предоставлять возможности создания, чтения, обновления и удаления некоторых объектов выбранного информационного ресурса (выполнения CRUD‑операций).
2. CRUD‑операции должны выполняться при помощи API выбранного ресурса, причём по крайней мере один тип операции должен требовать аутентификации на нём.
3. Должна быть возможность выполнения по крайней мере двух CRUD‑операций для одного и того же объекта выбранного информационного ресурса

Постановка задачи

По окончании выполнения задания должно спроектировано приложение, реализация в виде кода, тестирование и субъективное самооценивание.

Отчет по выполнению задания должен включать:

1. Общее описание приложения. Постановка задачи, введение в предметную область.
2. Архитектура системы. Обоснование выбора данного типа архитектуры распределенного приложения. Алгоритм работы приложения в целом.
3. Архитектура каждого из логических компонент системы (серверы, клиенты). Подходы к реализации. Алгоритмы работы.
4. Методы коммуникаций компонентов системы (клиент→сервер, сервер→клиент и т.д.).
5. Структура передаваемых данных.
6. Исходный код.
7. Сценария использования
8. Тестирование

Распределение баллов за выполнение работы представлено в следующей таблице:

|  |  |
| --- | --- |
| **Критерий оценивания** | **Оценка** |
| Приложение позволяет выполнять CRUD-операции над объектами выбранного ресурса | 1  (за каждый тип операции) |
| Приложение позволяет выполнять по крайней мере две CRUD-операции для одного и того же объекта выбранного ресурса | 1 |
| Приложение выполняет по крайней мере одну из CRUD-операций после успешной аутентификации | 2 |
| Приложение не требует повторной аутентификации при перезапуске программы | 1 |
| Параметры успешной аутентификации сохраняются при перезапуске программы в зашифрованном виде | 2 |

1. Описание приложения. Введение в предметную область

Предметная область – работа с текстовыми постами (записями) на стене определенного сообщества на ВК.

Для людей VK отличная русскоязычная (и не только) социальная сеть, которую можно использовать для общения с другими пользователями, а для разработчиков - сервис, позволяющий внедрять на свои сайты код программы. Предположим, вам дали задание прочесть все записи определенной группы без картинок, без комментариев, а только текстовую часть.

Само приложение имеет простой интерфейс и удобство использования, без лишних деталей.

* 1. Архитектура системы

На рисунке 1 изображена архитектура системы.

User

Client

Resource server

Authorization server

*Рис. 1 архитектура системы*

**2 Алгоритм работы приложения**

1. Пользователь использует адрес http-запрос вида –https://api.vk.com/method/METHOD\_NAME?PARAMETERS&access\_token=ACCESS\_TOKEN&v=V для использования технологии API Вконтакте. И с помощью этих запросов клиент реализует создание, удаление, обновление или получение ресурса при помощи CRUD-операций.
2. Все операции доступны, при использовании уникального токена, который был заранее получен. Тем самым и осуществляется авторизация по протоколу OAuth 2.0.
3. В приложении используется интерфейс командной строки(CLI). Чтобы создать, удалить, обновить, получить ресурс нужно соответственно в программе прописать create, delete, edit, read.
4. Операция «create». Программа запрашивает сообщение, который пользователь хочет создать: «Пожалуйста, введите новое сообщение». Подтверждением операции «create» будет служить сообщение «Запись на стене успешно была добавлена».
5. Операция «delete». Программа запрашивает id сообщения, который пользователь хочет удалить: «Пожалуйста, введите существующий id сообщения, который хотите удалить». Если id существует, операция осуществится. Подтверждением операции «delete» будет служить сообщение: «Запись на стене успешно была удалена, если вы ввели правильный id».
6. Операция «edit». Программа запрашивает id сообщения, который пользователь хочет изменить: «Пожалуйста, введите существующий id сообщения, который собирайтесь изменить». Затем программа запрашивает, какое новое сообщение хотите записать: «Пожалуйста, введите новое измененное сообщение». Если id существует, операция осуществится. Подтверждением операции «edit» будет служить сообщение: «Запись на стене успешно была изменена, если вы ввели правильный id».
7. Операция «read». Программа показывает все записи стены, а именно их текстовое содержание и id сообщения в таком формате: «id поста номер id,Текст сообщения = содержание сообщения»
8. Для выхода из приложения, необходимо написать программе exit.
9. Приложение позволяет выполнять 4 CRUD-операции для одного и того же объекта, а именно записи на стене конкретной группы в ВК.

3.1 Архитектура клиента

Клиент представляет собой CLI-приложение. Архитектура клиента изображена на рисунке 2.

Задачи клиента – организовать правильную работу CRUD-операций.

Client

Authorization server

Resource server

Коммуникационный модуль

*Рис. 2 архитектура клиента*

**3.2 Архитектура сервера**

Сервер представляет собой консольное приложение, архитектура сервера изображена на рисунке 3.

Задача сервера: обеспечивать корректность выполнения OAuth 2.0 авторизации.

Client

Server

Resource server (VK API)

Рис. 3 архитектура сервера

3.3 Методы коммуникаций компонентов системы

Для всех коммуникации в системе используется синхронное взаимодействие по протоколу HTTP. Передаваемый формат данных - JSON

3.4 Структура передаваемых данных

Формат данных описан в документации VK API 5.130 (https://github.com/VKCOM/vk-api-schema)

3.5 Язык программирования

Приложение разработано на языке программирования Python3 с версией 3.9.1

3.6 Исходный код

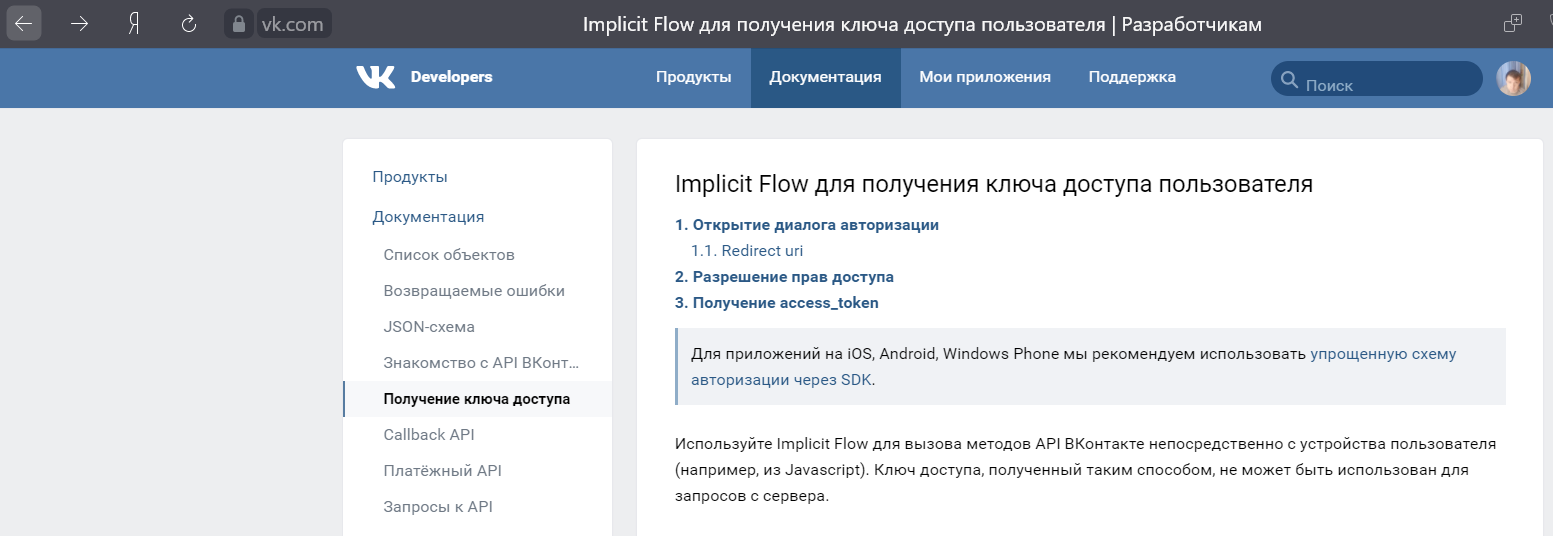
Реализация приложения находится по адресу:

Проект - <https://github.com/vadikTheBest/vki_api_python>

**4 Получение токена доступа для приложения**

Воспользуемся способом Implicit Flow [4]

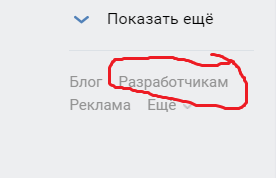
Implicit flow [14] — самый короткий и простой вариант. Ключ возвращается на устройство пользователя, где был открыт диалог авторизации (в виде дополнительного параметра URL). Такой ключ может быть использован только для запросов непосредственно с устройства пользователя (например, для выполнения вызовов из Javascript на веб-сайте или из мобильного приложения).



*Рис. 4 Implicit Flow для получения ключа доступа пользователя*

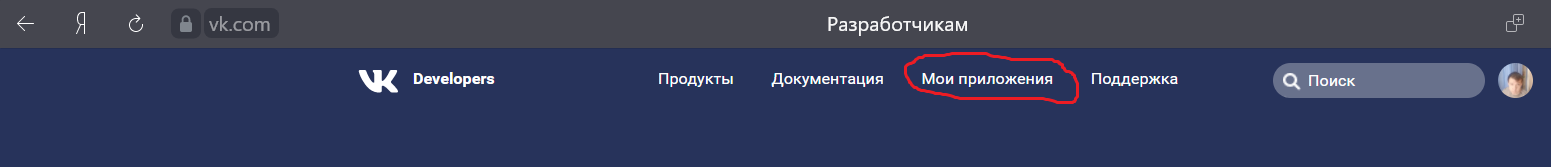
Надо создать приложение в Вконтакте для получение ключа доступа.

Заходим Моя страница – > Разработчикам (нижний колонтитул)

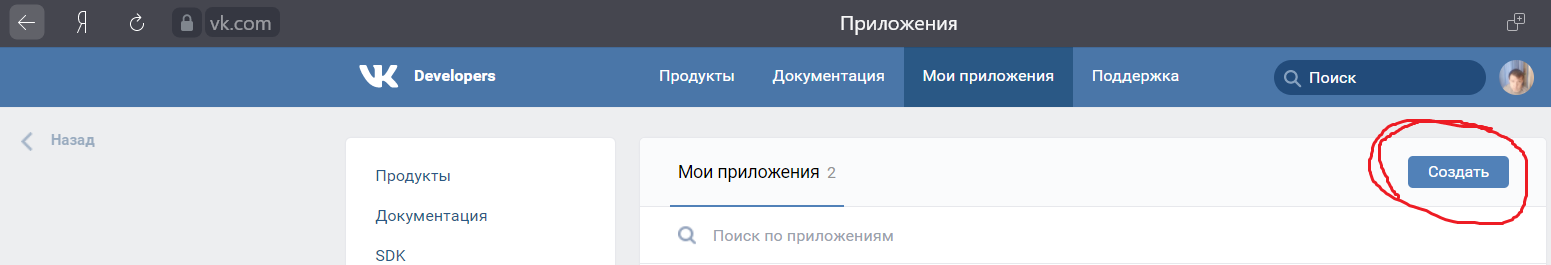


*Рис. 5 Нижний колонтитул Вконтакте*

Теперь надо зайти в – > Мои приложения

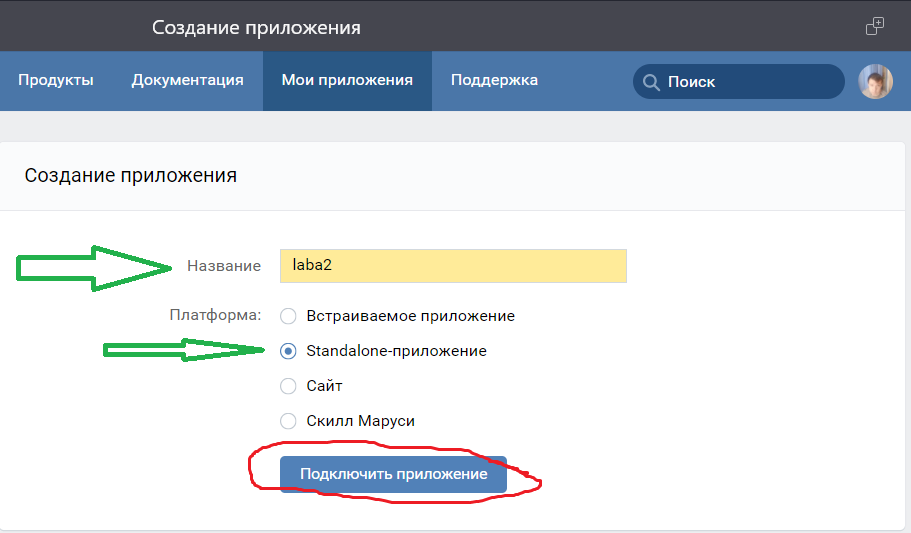


Выбрать кнопку «Создать»



*Рис. 6 страница разработчикам в вк*

Написать название будущего приложения(в моем случае laba2). Также отметить платформу как Standalone-приложение. Затем перейти – >Подключить приложение.



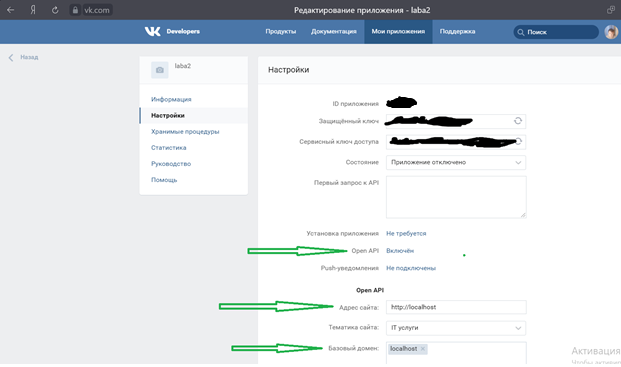
*Рис. 7 ВКонтакте. Создание приложения*

Перейти в настройки

Поставить Open API – включен

Адрес сайта написать <http://localhost>

Базовый домен написать localhost.Так как мы пользуемся программой на локальной машине. Далее – Сохранить изменения.



*Рис. 8 ВКонтакте. Мои приложения*

Далее в адресной строке ввести запрос на получение нужного токена.

Для запроса нужно будет

- ID нашего приложения;

далее надо указать какие будут права доступа для токена;

-wall(для доступа к обычным методам для работы со стеной);

-offline(для доступа к API в любое время);

- groups(для доступа к группам пользователя);

https://oauth.vk.com/authorize?client\_id=id\_приложения&display=page&redirect\_uri=https://oauth.vk.com/blank.html&scope=wall,offline,groups&response\_type=token&v=5.130

Далее наше приложение запросит права доступа для токена. Соглашаемся. К нам в адресной строке возвращается токен. Копируем его для использования в виде ключа

**5 Сценарии использования**

Сценарий №1 – CRUD-операция “Read”

Предусловия: нет

1. Пользователь запускает client.py
2. Вводит через клавиатуру слово «read»
3. Ожидание завершения программы

Ожидаемый результат: показаны записи стены, а именно их ID и их текст.

Сценарий №2 – CRUD-операция “Create”

Предусловия: нет

1. Пользователь запускает client.py
2. Вводит через клавиатуру слово «read»
3. Печатает текст, который хочет создать
4. Нажимает Enter
5. Ожидание завершения программы

Ожидаемый результат: создана запись на стене сообществе

Сценарий №3 – CRUD-операция “Delete”

Предусловия: нет

1. Пользователь запускает client.py
2. Вводит через клавиатуру слово «delete»
3. Пользователь пишет ID существующей записи, который хочет удалить
4. Ожидание завершения программы

Ожидаемый результат: создана запись в стене сообщества

Сценарий №4 – CRUD-операция “Update”

Предусловия: нет

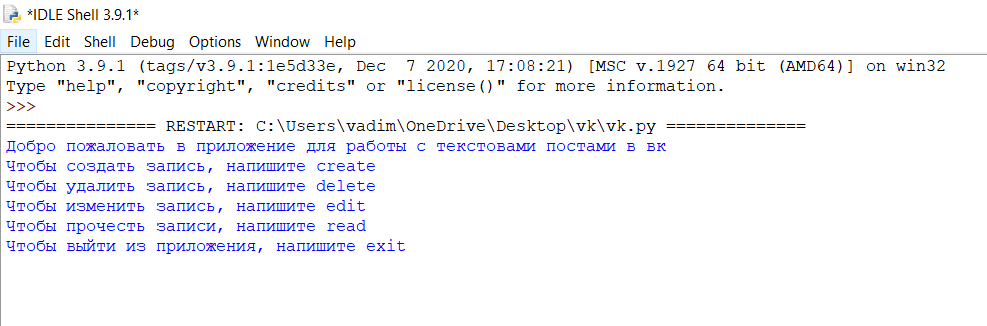
1. Пользователь запускает client.py
2. Вводит через клавиатуру слово «edit»
3. Пользователь пишет ID существующей записи, который хочет редактировать
4. Печатает текст на который хочет заменить
5. Нажимает Enter
6. Ожидание завершения программы

Ожидаемый результат: запись на стене сообщества была изменена

**6 Тестирование**

Запустим код vk.py. Для этого воспользуемся стандартным редактором Python(Python IDLE v.3.9.1.). Run -> Run Module (или горячей клавишей F5).

Приложение распечатывает приветственный текст с инструкцией по использованию программы



*Рис. 9 – Демонстрация приложения в IDLE Shell 3.9.1 ч.1*

За это отвечает следующий сниппет:

|  |
| --- |
| print("Добро пожаловать в приложение для работы с текстовами постами в вк\nЧтобы создать запись, напишите create\nЧтобы удалить запись, напишите delete\nЧтобы изменить запись, напишите edit\nЧтобы прочесть записи, напишите read\nЧтобы выйти из приложения, напишите exit\n") |

Для того, чтобы приложение работало бесконечно, используется цикл True. CRUD-операции реализовано через отдельные функции. Чтобы их вызвать нужно вводить специальные слова: create для CRUD-операции Create, delete для CRUD-операции Delete, edit для CRUD-операции Update, read для CRUD-операции Read, exit для прерывания цикла, то есть выхода из программы. Реализация приведена ниже:

|  |
| --- |
| while True:  keyboard = input()  if keyboard == 'create':  new\_message = input("Пожалуйста, введите новое сообщение\n")  crud\_create(new\_message)  elif keyboard == 'delete':  id\_delete\_post = input("Пожалуйста, введите существующий id сообщения, который хотите удалить\n")  crud\_delete(id\_delete\_post)  elif keyboard == 'edit':  id\_edit\_post = input("Пожалуйста, введите существующий id сообщения, который собирайтесь изменить\n")  edit\_message = input("Пожалуйста, введите новое измененное сообщение\n")  crud\_update(id\_edit\_post, edit\_message)  elif keyboard == 'read':  crud\_read()  elif keyboard == 'exit':  break |

**Сценарий №1.**

Для этого в приложении наберем – read и нажмем Enter.

В результате получаем



*Рис. 10 демонстрация приложения в IDLE Shell 3.9.1 ч.2*

То есть нам показывается список постов, а именно id поста и какой текст у этого поста.

За исполнение этой части программы отвечает следующий сниппет:

|  |
| --- |
| def getjson(api\_method):  response = requests.get(f"{api\_url}{api\_method}", params={"owner\_id": id\_group, "count": 10, "offset": 0, "access\_token": my\_token, "v": "5.52"})  response = response.json()  print(type(response))  return response  def crud\_read():  wall = getjson("wall.get")  counter = wall['response']['count']  while counter != 0:  print("id поста =", wall['response']['items'][counter-1]['id'], " Текст сообщения = ", wall['response']['items'][counter-1]['text'], '\n')  counter -= 1 |

Первая функция отвечает за получение данных

|  |
| --- |
| def getjson(api\_method):  response = requests.get(f"{api\_url}{api\_method}", params={"owner\_id": id\_group, "count": 10, "offset": 0, "access\_token": my\_token, "v": "5.52"})  response = response.json()  print(type(response))  return response |

То есть мы составляем сначала запрос, в котором указываем куда мы обращаемся(VK API Methods). Указываем нужные нам параметры:

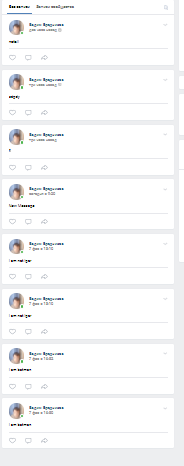
* адрес, куда мы обращаемся(https://api.vk.com/method/);
* владельца группы(Указываю свою созданную группу);
* сколько постов мы хотим отобразить (К примеру 10);
* токен доступа;
* и версию вк API(в нашем случае v.5.52).

Мы получили json-данные. Далее вторая функция отображает данные, которые мы ходим отобразить. То есть нам нужно показывать комментарии, сколько лайков, сколько просмотров, кто запостил и т.д. Слишком много данных, нам надо отобразить только ID поста и текст этой самой записи.

|  |
| --- |
| def crud\_read():  wall = getjson("wall.get")  counter = wall['response']['count']  while counter != 0:  print("id поста =", wall['response']['items'][counter-1]['id'], " Текст сообщения = ", wall['response']['items'][counter-1]['text'], '\n')  counter -= 1 |

То есть мы записываем в переменную wall полученные json-данные. Затем настраиваем счетчик. Получаем через переменную wall. Потом создаем цикл, который выводит нужные нам данные: ID и текст.

Проверяем через браузер



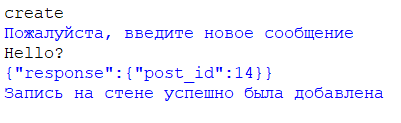
*Рис. 11 демонстрация реализации кода в браузере ч.1*

Данные совпадают

**Сценарий №2**

Для этого в приложении наберем – read и нажмем Enter.

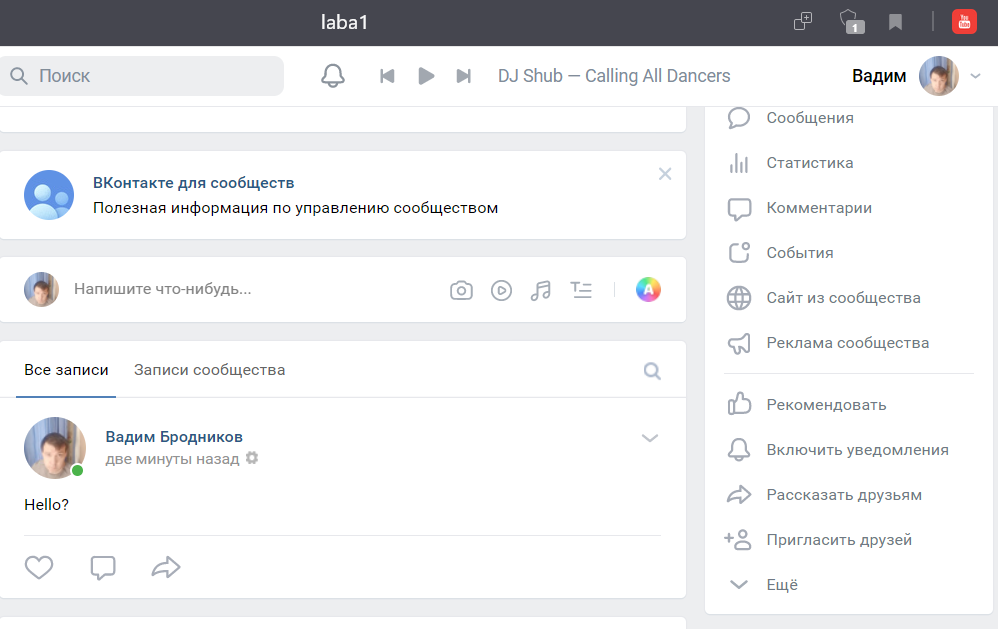
В результате получаем



*Рис. 12 демонстрация приложения в IDLE Shell 3.9.1 ч.3*

То есть программа запрашивает, какое сообщение опубликовать. Пользователь набирает сообщение. Нажимает Enter. Приложение уведомляет о том, что сообщение было добавлено.

Проверяем в браузере



*Рис. 13 демонстрация реализации кода в браузере ч.2*

Работает

За исполнение этой части программы отвечает следующий сниппет:

|  |
| --- |
| def crud\_create(text\_message):  api\_method = 'wall.post'  response = requests.get(f"{api\_url}{api\_method}", params={"owner\_id": id\_group, "message": text\_message, "offset": 0, "access\_token": my\_token, "v": "5.52"})  print(response.text)  print("Запись на стене успешно была добавлена\n") |

То есть мы составляем GET-запрос, в котором указываем куда мы обращаемся(VK API Methods). Указываем нужные нам параметры:

* адрес, куда мы обращаемся(<https://api.vk.com/method/>)
* метод, который мы используем(wall.post).Создание поста
* владельца группы(Указываю свою созданную группу),
* сообщение, которое хотим опубликовать,
* токен доступа
* версию вк API(в нашем случае v.5.52).

**Сценарий №3**

Для этого в приложении наберем – delete и нажмем Enter. Программа запрашивает ID поста, который пользователь собирается удалить. Если пользователь введет ID, который существует в БД, то приложение удалит его.

В результате получаем:

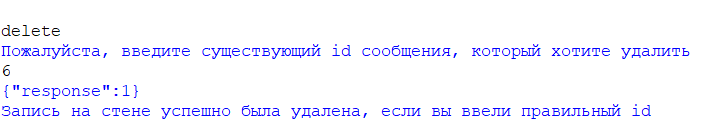


Рис. 14 демонстрация приложения в IDLE Shell 3.9.1 ч.4

Проверяем в браузере

Для этого вводим в адресной строке

<https://vk.com/club202262558?w=wall-202262558_6>

То есть адрес записи. Браузер уведомляет, что записи не существует. К сожалению, вк эту уведомление показывает не надолго(2-3 секунды) и переходит на страницу группы, поэтому не получилось сделать скриншот.

**Сценарий №4**

Для этого в приложении наберем – edit и нажмем Enter. Программа запрашивает ID поста, который пользователь собирается изменить. Если пользователь введет ID, который существует в БД, то приложение удалит его.

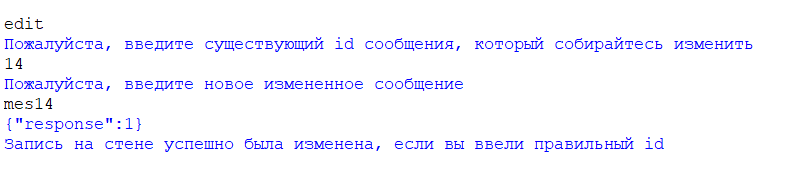


Рис. 15 демонстрация приложения в IDLE Shell 3.9.1 ч.5

Проверяем в браузере

Для этого вводим в адресной строке

<https://vk.com/club202262558?w=wall-202262558_14>

То есть адрес записи

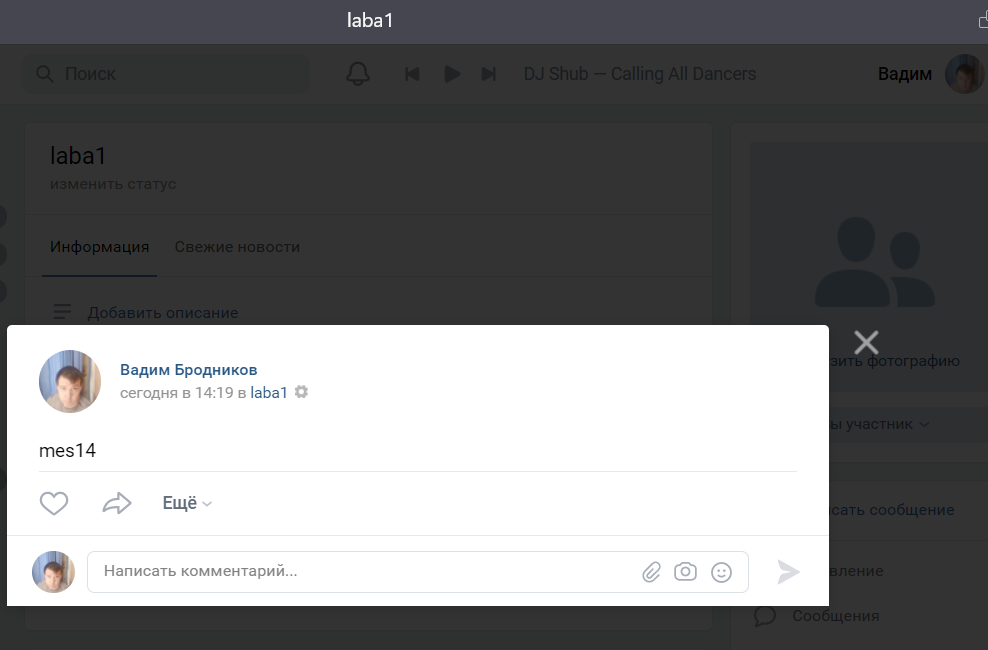


Рис. 16 демонстрация реализации кода в браузере ч.3

Работает

За исполнение этой части программы отвечает следующий сниппет:

|  |
| --- |
| def crud\_update(id\_post, new\_message):  api\_method = 'wall.edit'  response = requests.get(f"{api\_url}{api\_method}", params={"owner\_id": id\_group, "post\_id": id\_post, "message": new\_message, "offset": 0, "access\_token": my\_token, "v": "5.52"})  print(response.text)  print("Запись на стене успешно была изменена, если вы ввели правильный id\n") |

То есть мы составляем GET-запрос, в котором указываем куда мы обращаемся(VK API Methods). Указываем нужные нам параметры:

* адрес, куда мы обращаемся(https://api.vk.com/method/)
* метод, который мы используем(wall.edit). Изменение поста
* владельца группы (Указываю свою созданную группу),
* ID поста, которое хотим изменить
* сообщение, которое хотим изменить,
* токен доступа
* версию вк API(в нашем случае v.5.52).

**Список файлов**

|  |  |  |
| --- | --- | --- |
| Название | Размер | Тип файла |
| vk.py | 3282 байт | Исполняемый |
| api\_key.py | 100 байт | Исполняемый |

В vk.py реализовано приложение, в целях безопасности токен для приложения хранится в отдельном файле api\_key.py.

**Заключение**

В результате работы была достигнута цель и выполнены следующие требования:

1. Требование «Приложение позволяет выполнять CRUD-операции над объектами выбранного ресурса» выполнено полностью. Ожидаемое количество баллов: 4.
2. Требование «Приложение позволяет выполнять по крайней мере две CRUD-операции для одного и того же объекта выбранного ресурса» выполнено полностью. Ожидаемое количество баллов: 1.
3. Требование «Приложение выполняет по крайней мере одну из CRUD-операций после успешной аутентификации» выполнено полностью. Ожидаемое количество баллов: 2.
4. Требование «Приложение не требует повторной аутентификации при перезапуске программы» выполнено полностью. Ожидаемое количество баллов: 1.
5. Требование «Параметры успешной аутентификации сохраняются при перезапуске программы в зашифрованном виде» выполнено некорректно. Ожидаемое количество баллов: 2.

**Перечень использованных источников**

1. ВКонтакте. Документация. Разработчикам [Электронный ресурс]. URL: <https://vk.com/dev/manuals> (Дата обращения: 06.03.2021)
2. ВКонтакте. Возвращаемые ошибки [Электронный ресурс]. URL: [https://vk.com/dev/errors /](https://vk.com/dev/errors%20/) (Дата обращения: 06.03.2021)
3. ВКонтакте. JSON-схема. Разработчикам [Электронный ресурс]. URL: <https://vk.com/dev/json_schema> (Дата обращения: 06.03.2021)
4. ВКонтакте. Implicit Flow для получения ключа доступа пользователя [Электронный ресурс]. URL: <https://vk.com/dev/implicit_flow_user> (Дата обращения: 06.03.2021)
5. Requests в Python – Примеры выполнения HTTP запросов [Электронный ресурс]. URL: <https://python-scripts.com/requests> (Дата обращения: 06.03.2021)
6. JSON в Python [Электронный ресурс]. URL: <https://python-scripts.com/json> (Дата обращения: 06.03.2021)
7. Лицензия Python 3.9.1 [Электронный ресурс]. URL: <https://docs.python.org/3/license.html> (Дата обращения: 06.03.2021)
8. Download Python [Электронный ресурс]. URL: <https://www.python.org/downloads/> (Дата обращения: 06.03.2021)
9. Спецификация Python IDLE 3.9.1 [Электронный ресурс]. URL: <https://docs.python.org/3.9/library/idle.html> (Дата обращения: 06.03.2021)
10. Лицензия языка PHP [Электронный ресурс]. URL: <https://www.php.net/license/> (Дата обращения: 06.03.21)
11. Лицензия KittenPHP [Электронный ресурс]. URL: <https://github.com/VKCOM/kphp/blob/master/LICENSE> (Дата обращения: 14.02.21)
12. ВКонтакте. Что такое токен? [Электронный ресурс]. URL: <https://vk.com/faq11764> (Дата обращения: 19.03.21)
13. ВКонтакте. Получение ключа доступа [Электронный ресурс]. URL: https://vk.com/dev/access\_token?f=2.%20%CA%EB%FE%F7%20%E4%EE%F1%F2%F3%EF%E0%20%F1%EE%EE%E1%F9%E5%F1%F2%E2%E0 (Дата обращения: 19.03.21)
14. OAuth 2.0 простым и понятным языком [Электронный ресурс]. URL: <https://habr.com/ru/company/mailru/blog/115163/> (Дата обращения: 19.03.21)
15. JSON: основы использования [Электронный ресурс]. URL: <https://ruseller.com/lessons.php?id=1212> (Дата обращения: 19.03.21)
16. ЧТО ТАКОЕ GUI И CLI [Электронный ресурс]. URL: <https://losst.ru/chto-takoe-gui-i-cli> (Дата обращения: 19.03.21)
17. Простым языком об HTTP [Электронный ресурс]. URL: <https://habr.com/ru/post/215117/> (Дата обращения: 19.03.21)