

Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого

—
Институт компьютерных наук и технологий
Кафедра «Информационная безопасность компьютерных систем»

ОТЧЕТ
ОБ УЧЕБНОЙ ПРАКТИКЕ

**«Исследование бот платформы для системы мгновенного обмена
сообщениями»**

Выполнил
студент гр. 23508/4

Е.Г. Проценко

Руководитель
от ФГАОУ ВО «СПбПУ»

М.О. Калинин

от LG Electronics Inc.
Russia R&D Lab

О.В. Яковенко

Санкт-Петербург
2016

ХОД РАБОТЫ

1. Web-сервер

Web-сервер был написан на Python (Django). Реализованы следующие возможности

- регистрация:

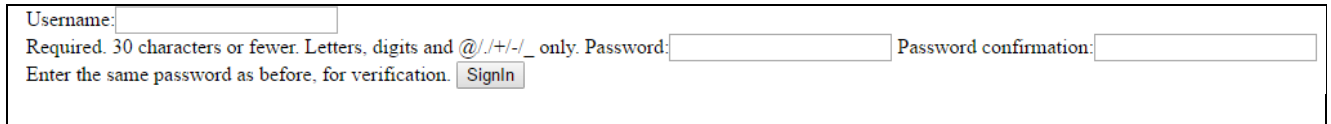
A screenshot of a web registration form. It contains three input fields: 'Username:', 'Password:', and 'Password confirmation:'. Below the 'Password:' field is a small text note: 'Required. 30 characters or fewer. Letters, digits and @/./+/-/_ only.' Below the 'Password confirmation:' field is another small text note: 'Enter the same password as before, for verification.' To the right of these fields is a 'SignIn' button.

Рис 1. Демонстрация возможности регистрации

- авторизация
- получение секретного ключа (уникального, для каждого пользователя)
- создание новой заметки
- просмотр все заметок пользователя

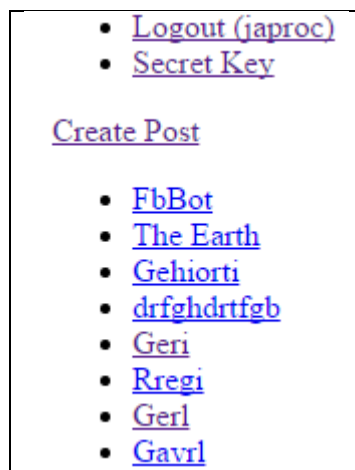


Рис 2. Функционал

- просмотр содержимого заметки
- удаление заметки



Рис 3. Функционал

ngrok

Для того, чтобы получить возможность доступа к веб-серверу с любого устройства по интернету, был использован сервис **ngrok**, который создает защищенный туннель к localhost. Софт можно скачать с официального сайта. Запуск происходит командой:

>ngrok http 8000

где http говорит на о том, что будет создан HTTP туннель, а 8000 – номер порта на локальном компьютере.

В результате исполнения команды выше получим примерно что-то такое:

```
ngrok by @inconshreveable

Tunnel Status      online
Version            2.1.3
Region             United States (us)
Web Interface      http://127.0.0.1:4040
Forwarding          http://bd8f77ad.ngrok.io -> localhost:8000
                   https://bd8f77ad.ngrok.io -> localhost:8000

Connections        ttl    opn    rt1    rt5    p50    p90
                   391    0      0.00   0.02   0.02   0.09

HTTP Requests
-----
```

Рис 4. Процесс ngrok

В нашем случае имеем: при обращении к адресу **https://bd8f77ad.ngrok.io/** весь наш трафик будет переадресовываться на localhost:8000

API

Были разработаны API, используя которые можно запрограммировать бота, например, для Telegram, Facebook, Vk, Kik, Skype и других сервисов, который предоставит возможность пользователям использовать функционал веб-сервера используя соответствующий ресурс (Telegram, Facebook Messenger, Vk Chat, Kik, Skype).

Веб-сервер поддерживает только POST запросы. Ответы с веб-сервера содержат json-объект.

*Конкретные json ответы для каждого API-запроса в пределах отчета не рассматриваются.

Аутентификация

Главная задача аутентификации – дать ответ о том, существует ли пользователь с запрашиваемым секретным ключом. Каждый пользователь может получить уникальный секретный ключ на веб-сервере.

Запрос имеет следующий вид:

`https://bd8f77ad.ngrok.io/api/auth/<secret_key>`

Например, такой: `https://bd8f77ad.ngrok.io/auth/u98ghf44h1`

Запросы

Все запросы к API веб-сервера должны иметь следующий вид:

`https://bd8f77ad.ngrok.io/api/<secret_key>/METHOD_NAME`

*Все методы состоят только из символов нижнего регистра.

- **List**

Данный метод возвращает список всех заметок, принадлежащих пользователю.

`https://bd8f77ad.ngrok.io/api/<secret_key>/list`

- **Delete**

Удаляет заметку по ее уникальному номеру (<id>), если она принадлежит пользователю. Не отвечает.

`https://bd8f77ad.ngrok.io/api/<secret_key>/delete/<id>`

- **Add**

Добавляет заметку с соответствующим заголовком (<title>) и текстом (<text>) в список заметок пользователя. Не отвечает.

`https://bd8f77ad.ngrok.io/api/<secret_key>/add/<title>/<text>`

Комментарий

Ссылка на [github](#), где можно найти исходный код веб-сервера прикреплена в Приложении

2. Bot-servers

Задача Bot-сервера состоит в том, чтобы предоставить пользователю возможность просматривать, добавлять, удалять заметки, хранящиеся на web-сервере, непосредственно внутри текущего чата с ботом. Это достигается за счет использования REST-API.

Чтобы понять, какой именно пользователь веб-сервера или из какого чата пользуются бот-сервером, поддерживается система аутентификации по секретному ключу. Для этого пользователь должен ввести текст **'auth'**, а затем свой **<secret_key>**, полученный на веб-сервере.

Поскольку с ботом могут взаимодействовать множество пользователей одновременно, то со стороны бот-сервера имеется словарь, имеющий следующую структуру: **'sender_id' = 'secret_key'**, где **sender_id** – отправитель сообщения.

Бот должен обрабатывать для пользователя только те команды, которые доступны из текущего состояния. Например, в исходном, или базовом, состоянии пользователь может вызвать любую команду: **auth, delete, add, list**. А в состоянии **'auth'** бот ожидает увидеть во входящих сообщениях секретный ключ, поэтому если вы введете **delete**, вы не перейдете в состояние удаление заметки, а бот примет это за ваш секретный ключ и попробует аутентифицироваться на веб-сервере с ним. Диаграмма состояний приведена ниже:

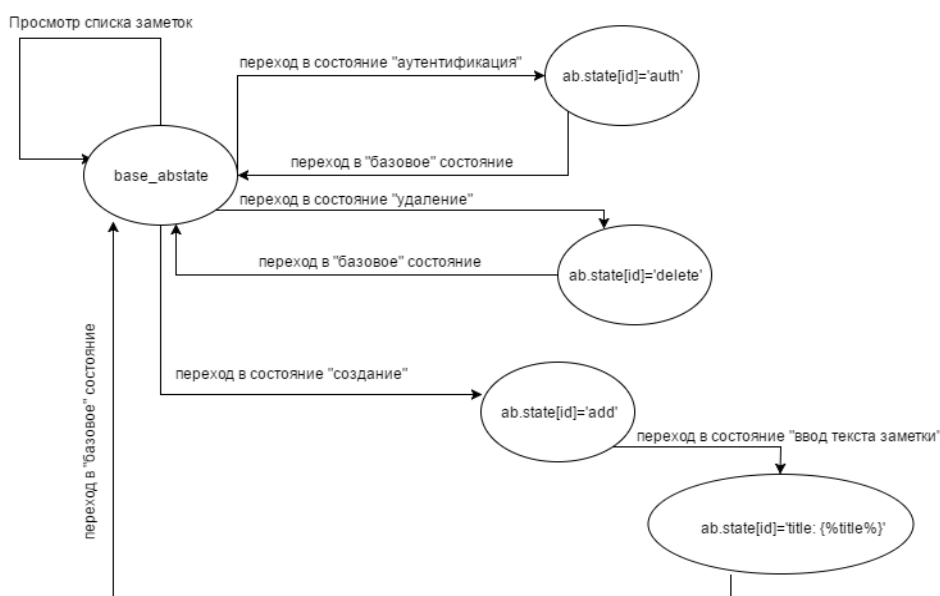


Рис 5. Диаграмма состояний

Для того, чтобы бот хранил состояния всех пользователей имеется словарь (на рисунке это **ab.state**). Он имеет следующую структуру: **'sender_id' = 'status'**, где **sender_id** – отправитель сообщения, **status** – текущее состояние. Текущее состояние определяет каким образом будут обрабатываться входящие сообщения.

Такие действия, как запрос списка всех заметок, добавление, удаление заметок, требуют, чтобы пользователь был аутентифицирован. Поэтому перед тем, как позволить пользователю использовать команды **list**, **delete**, **add**, пользователь должен аутентифицироваться командой **'auth'** и дальнейшим вводом своего секретного ключа. После того, как пользователь аутентифицировался, ему предоставляется возможность использовать все остальные команды. Аутентификация производится единожды.

Kik

Для того, чтобы получать информацию о том, что пользователь отправил боту сообщение используется метод, который называется **webhooks**. Это означает, что на наш бот-сервер будут поступать http запросы, содержащие json-объект, сразу, как только пользователь отправил сообщение.

При это данная технологий требует защищенного соединения, при помощи https-соединения. Это делается в целях защиты, например, от атак типа MITM (man in the middle). Здесь нам опять же поможет ngrok, на котором мы запустим бот-сервер, поскольку он предоставляет возможность сертифицированного, защищенного http-соединения.

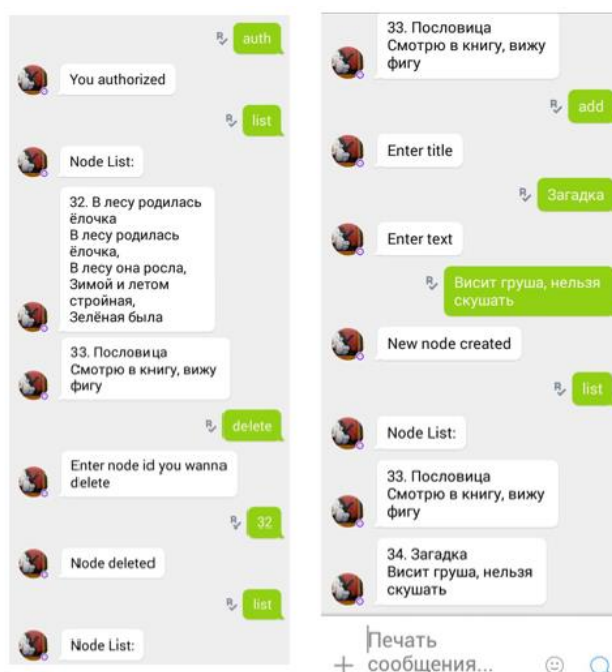


Рис 6. Работа Kik-бота

Адрес бот-сервера устанавливается при запуске самого бот-сервера, вызывая определенные функции. С ними можно будет подробнее познакомиться в исходных кодах программы.

Бот-сервер написан на Python с использованием модулей **kik**, **flask**.

Kik – специализированный модуль, который помогает настроить работу с определенным ботом по его имени и специальному ключу, настроить адрес бот-сервера, предоставляет Kik API.

Flask – предоставляет удобные настройки процесса на прослушивание определенного порта

Facebook Messenger

Здесь так же используются **webhooks**. Однако на этот раз адрес бот-сервера, на который нужно отправлять все логи задается не во время запуска процесса, а на сайте разработчиков facebook.

Сначала создадим обычную страницу (page). Можно воспользоваться ссылкой, которая находится в Приложении.

Теперь создаем новое приложение (ссылка в Приложении), используя ручную настройку (basic setup):

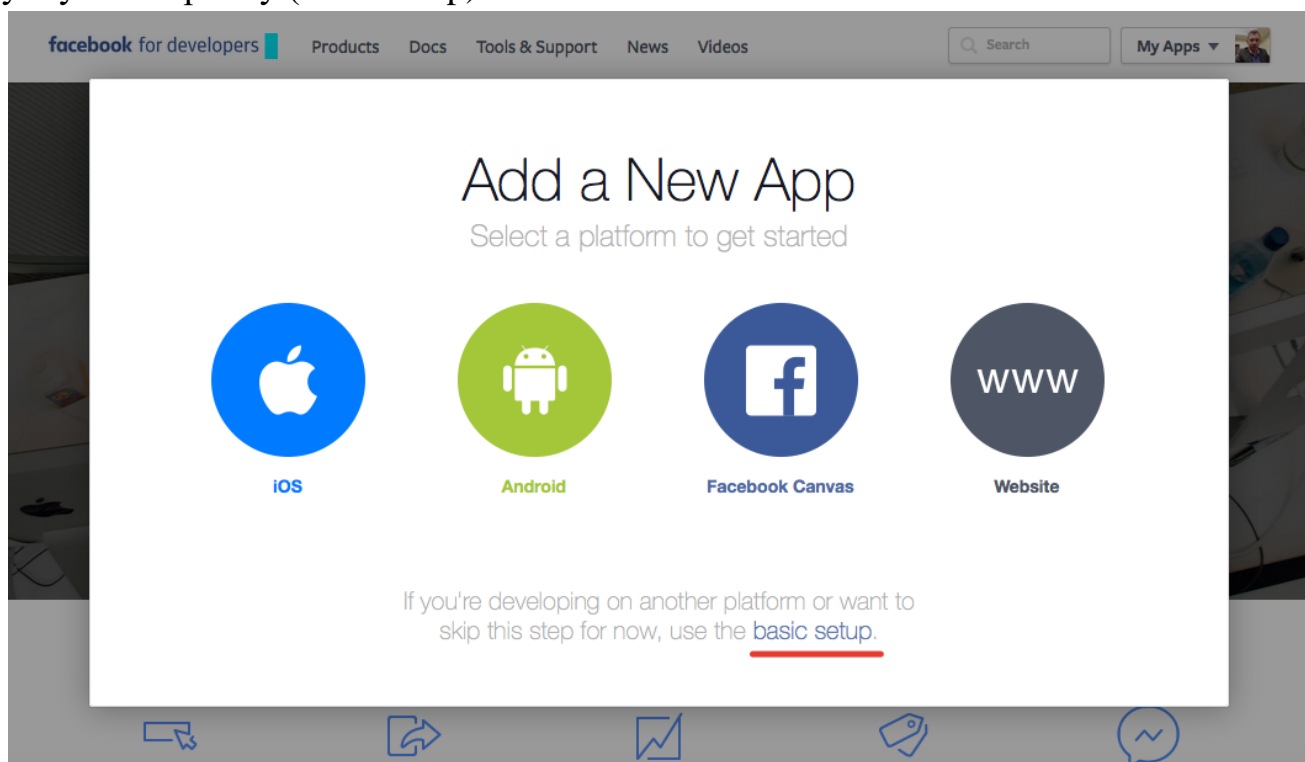


Рис 7. Создание приложения Facebook

Заполняем форму, выбираем категорию ‘App for Pages’

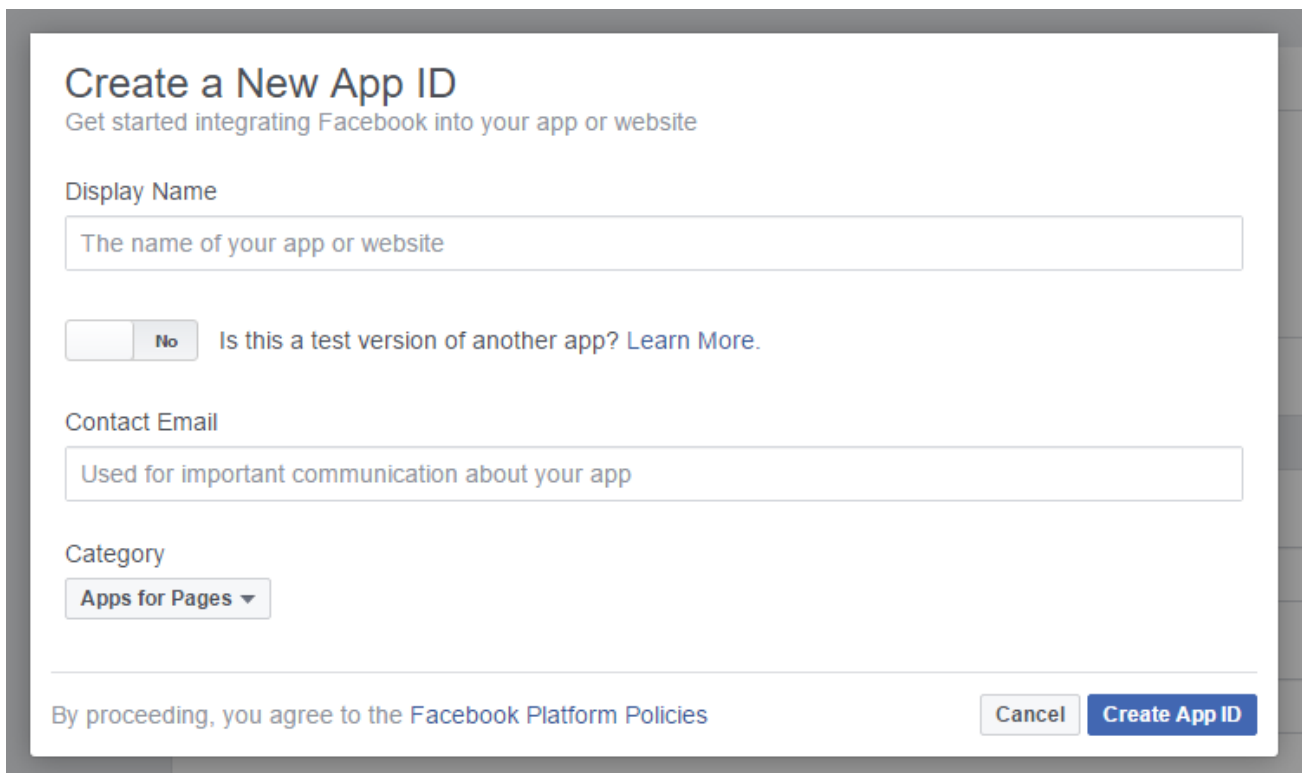
The image shows a web form titled "Create a New App ID" with the subtitle "Get started integrating Facebook into your app or website". The form contains several input fields: "Display Name" with a placeholder "The name of your app or website", a checkbox labeled "No" followed by the text "Is this a test version of another app? Learn More.", "Contact Email" with a placeholder "Used for important communication about your app", and a "Category" dropdown menu currently set to "Apps for Pages". At the bottom, there is a line of text: "By proceeding, you agree to the Facebook Platform Policies". To the right of this text are two buttons: "Cancel" and "Create App ID".

Рис 8. Форма при создании приложения Facebook

Теперь в настройках самого приложения нужно добавить 2 компонента: **Messenger** и **Webhooks**, – используя **+Add Product**

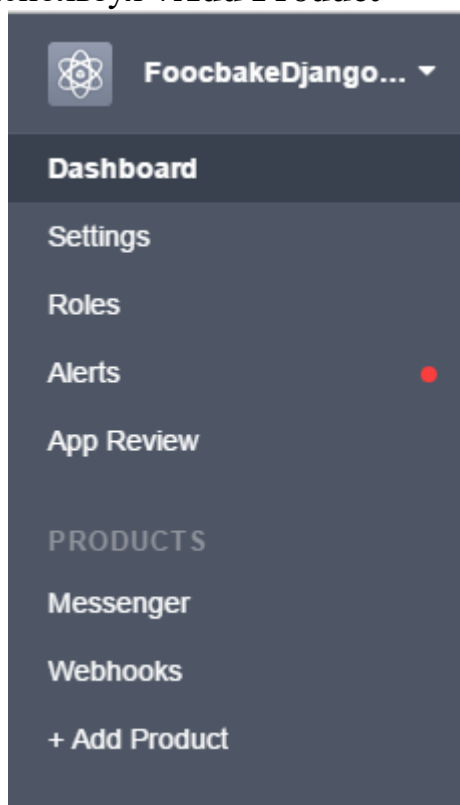


Рис 9. Настройки приложения

Подписываемся на webhooks в одноименной вкладке. Для этого вам нужно сначала запустить следующий скрипт. VERIFY_TOKEN – придумайте любой.

```
from flask import Flask, request
import requests

app = Flask(__name__)

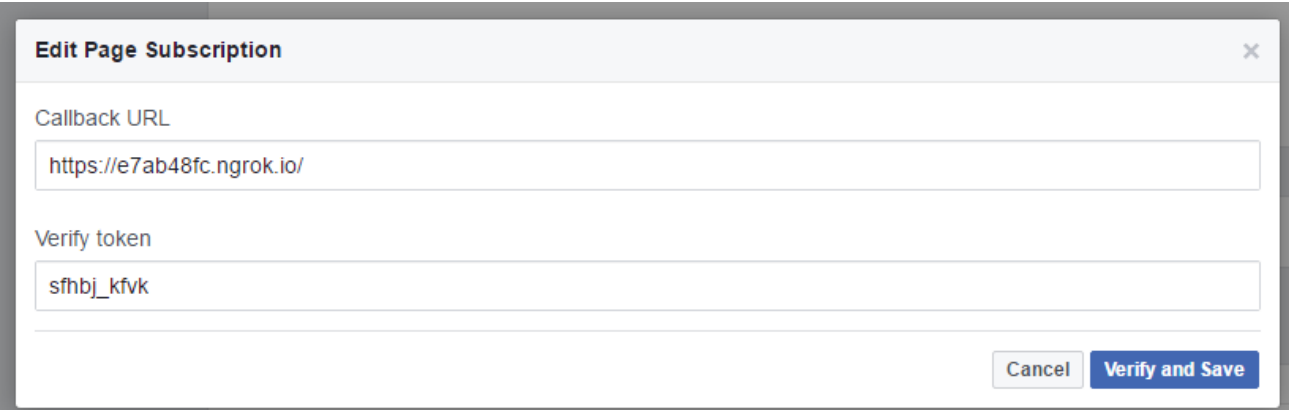
VERIFY_TOKEN="sfhbj_kfvk"

@app.route('/', methods=['GET'])
def handle_verification():
    if request.args['hub.verify_token'] == VERIFY_TOKEN:
        return request.args['hub.challenge']
    else:
        return "Invalid verification token"

if __name__ == '__main__':
    app.run(debug=True)
```

Рис 10. Скрипт для подписки на Webhooks

Теперь сама подписка. Создаем новый туннель через ngrok. И в поле Verify token вводит ранее придуманный VERIFY_TOKEN



The image shows a web interface for editing a page subscription. The title bar says "Edit Page Subscription". There are two input fields: "Callback URL" containing "https://e7ab48fc.ngrok.io/" and "Verify token" containing "sfhbj_kfvk". At the bottom right, there are two buttons: "Cancel" and "Verify and Save".

Рис 11. Подписка на Webhooks

Если у вас не будет запущен скрипт, то при попытке подписаться при GET запросе вернется ошибка 404 и вам будет отказано в подписке.

После того, как мы успешно подписались на webhooks, прикрепляем их к определенной странице и получаем ее секретный ключ для работы с API

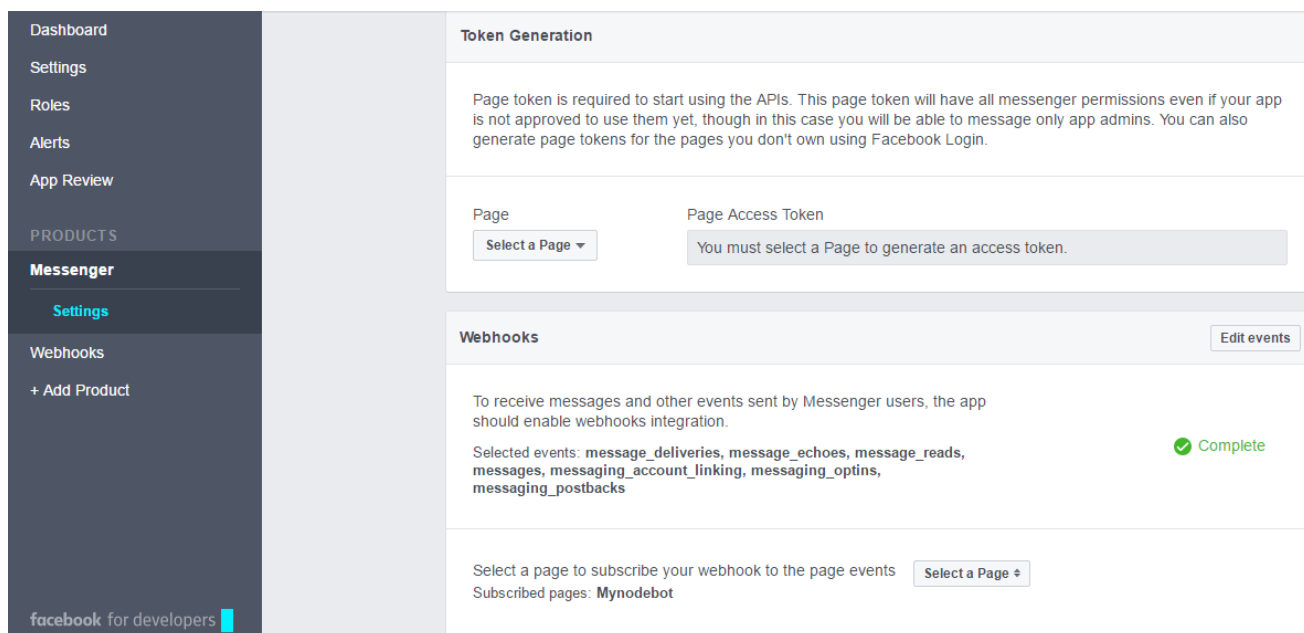


Рис 12. Привязка Webhooks к странице и получения секретного ключа страницы

Теперь можно запускать серверный скрипт с обработкой POST запросов, которыми являются webhooks.

Telegram

Telegram предоставляет два метода получения логов. Первый, все те же webhooks, а второй так называемый GetUpdate (так называемые long polling). GetUpdate представляет из себя GET запрос, в ответ на который приходит json-объект, хранящий логи за последний период. Этот метод можно использовать, чтобы не пробрасывать туннель через ngrok. Так и было сделано.

Бот-сервер в бесконечном цикле обращается за новыми логами с определенным периодом или без него.

Бот-сервер написан на Python с использованием модуля **telebot**. Telebot – интерфейс, упрощающий работу с Telegram Bot API.

Комментарий

Все боты написано на Python и поддерживают одинаковый функционал.
Ссылка на github, где можно найти исходные код всех бот-серверов
прикреплена в Приложении

ВЫВОД

Получены знания о создании web-сервера посредством Python(Django).
Написал API для работы с данными, хранящимися на веб-сервере. Придуман и
реализован механизм взаимодействия между бот-сервером и веб-сервером.

ПРИЛОЖЕНИЕ

1. <https://github.com/Yookkee/Django/> - исходный код веб-сервера
2. <https://github.com/Yookkee/BotServers/> - исходный код бот-серверов
3. https://www.facebook.com/pages/create/?ref_type=bookmark – создание страницы
4. <https://developers.facebook.com/apps/> - создание нового приложения Facebook