Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

РАЗРАБОТКА БОТА ДЛЯ ПОИСКА БЛИЖАЙШИХ МЕСТ НА КАРТЕ

Руководитель:
Козуб Виктор Николаевич

Выполнила: студентка гр. 653501
Говзич Анастасия Викторовна

Минск 2017

Содержание

ВВЕДЕНИЕ

Причины разработки приложения

В течение последних лет боты для социальных сетей и мессенджеров набирают все большую популярность. При помощи специального API и знания любого языка программирования пользователь может создать бота – аккаунт, управляемый программой. Боты могут отвечать на сообщения пользователей в персональных или групповых чатах, а также выполнять задачи в развлекательных или бизнес целях. Помимо этого, многие банки, информационные ресурсы заводят себе ботов для расширения аудитории пользователей и поддержания с ними постоянной связи.

Боты в Telegram – это специально обученные программные роботы, которые могут выполнять за своего создателя множество задач:

– рутинный поиск и обработка информации;

– организация массовой рассылки определенной информации определенной группе абонентов или всем абонентам;

– перевод текстов / сайтов / публикаций в социальных сетях с одного языка на другой;

– извлечение с различных новостных лент самых свежих новостей;

– игра с пользователем и т.д.

Список возможностей ботов не ограничивается нашим перечислением, его можно продолжить и пополнить, однако мы выделили самые основные и часто используемые их функции.

Одними из популярных ботов для мессенджера Telegram являются YouTube Search, Giphy GIF Search, Foursquare. Это так называемые инлайновые (встраиваемые) боты, созданные для того, чтобы делиться информацией с другими пользователями.

YouTube Search – бот для быстрого поиска видеофайлов и отправления их определенным образом выбранной аудитории.

Giphy GIF Search – бот для быстрого поиска анимации, используемый с той же целью, как и YouTube Search.

Одним из самых популярных является Foursquare – бот, который предлагает места рядом с пользователем. Однако из информации о месте Foursquare выдает подсказку только о его данных на карте и адрес.

Поэтому нами было решено сделать бота, который не только определяет места поблизости, но и дает пользователю более значимую и полезную информацию о заведении, такую как время его работы и рейтинг.

Одной из главных причин разработки являлось недостаточное количество известных и многофункциональных ботов, используемых в этом направлении.

Помимо этого, функционал библиотек для работы с социальными сетями и мессенджерами находится на начальной стадии разработки, что обеспечивает

быстрое изучение API и последующее расширение функционала бота в зависимости от направления развития инструментов разработки.

Важно отметить то, что социальные сети и мессенджеры предполагают наличие своей аудитории, возможность привлечь которую проще, чем собирать аудиторию для отдельного приложения. При разработке отдельного приложения также надо учитывать наличие разнообразных устройств, под которые его надо оптимизировать.

Кроме того, создание подобных ботов является основой для развития искусственного интеллекта, не требующего излишних затрат, которые предполагались при программировании роботов.

Постановка целей

Целью данного курсового проекта является создание Telegram-бота (SnugPointsBot), который будет использоваться не только для определения ближайшего общественного места к пользователю, но и способного, основываясь на его запросе, представить интересующую информацию о работе, рейтинге заведения.

Для реализации цели требуется выполнить следующие задачи:

1. Выбрать язык программирования и платформу, учитывая готовые библиотеки для взаимодействия с серверами социальных сетей и мессенджеров.
2. Изучить определение и получение геолокации с помощью платформы для бота.
3. Найти инструмент для обработки полученных геоданных.
4. Развернуть приложение на сервере.

Предмет исследования – программное конструирование и создание нового Telegram-бот (SnugPointsBot).

Объект исследования – бот (аккаунт, который отвечает на запросы пользователя с помощью разработанного скрипта).

Структура курсовой работы: курсовая работа состоит из «Введения», 3 глав «Сравнение с аналогичными продуктами», «Интерфейс Бота», «Технические моменты разработки Бота» с подразделами, «Заключения», «Списка использованной литературы» и 5 приложений.

Глава 1. Сравнение с аналогичными продуктами

Как уже упоминалось ранее, основной причиной разработки данного приложения является желание добавить к существующему встраиваемому боту Foursquare дополнительный функционал и исправление некоторых недочетов, имеющихся в нем.

К сожалению, изначально Foursquare является социальной сетью с функцией геопозиционирования, поэтому данное приложение имеет свою базу данных позволяющую выдавать ответы в одинаковом формате.

SnugPointsBot, который работает с помощью API Google Places, не имеет возможности привести названия мест и их адреса к одному формату. Можно использовать Яндекс API для более подробного и форматированного вывода, так у Яндекс Карта ориентирована на белорусских потребителей, но получить APIkey не удалось из-за проблем с личным кабинетом разработчика.

Стоит отметить, что Foursquare выдает ответы, основываясь на рейтинге заведений, а не на близости к пользователю, также не выводит некоторые места, которые есть на картах. Например, указывая @foursquare coffee near, первым результатом может оказаться локация в 5 километрах от пользователя, а дальше могут быть более близкие результаты, но даже 2 километра достаточно большая дистанция для тех пользователей, которые, например, после длительной прогулки ищут кафе (см. приложение 1). SnugPointsBot выдает результат поиска в радиусе не более 1 километра от текущего местоположения рользователя, что является предпочтительным.

Помимо этого, не понятна основная задача Foursquare, так как кроме карты и адреса рекомендованного места, как правило, удаленного от пользователя, нельзя получить дополнительную информацию об интересующих достопримечательностях либо гастрономических заведений. А у него, например, может быть выходной, особенно актуально с музеями. Понятно, что Foursquare был создан для того, чтобы делиться любимыми местами с друзьями, но это не одна из самых полезных и часто используемых функций. Куда удобнее получать другую нужную информацию из бота, чем и отличается SnugPointsBot.

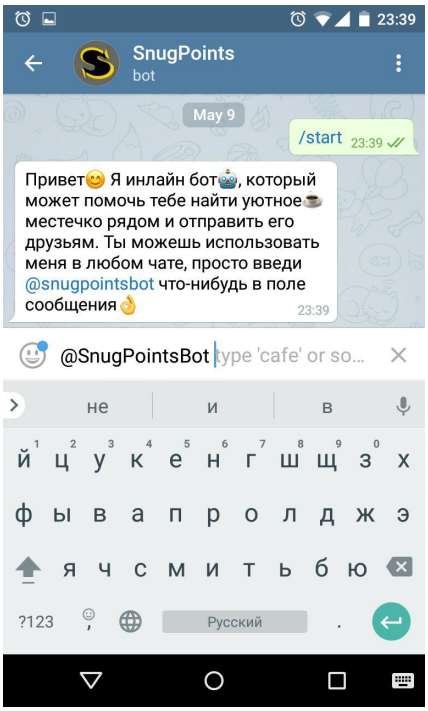
С другой стороны, если говорить об интерфейсе и процессе взаимодействия с пользователем, то Foursquare и SnugPointsBot являются схожими. Особенностью второго продукта является его ориентированность на русскоязычного пользователя в связи с созданием наиболее форматированного вывода ответов.

Таким образом, оба приложения имеют некоторые недостатки. SnugPointsBot выдает более правильные ответы на запросы, имеет больший функционал, а Foursquare охватывает более широкую аудиторию.

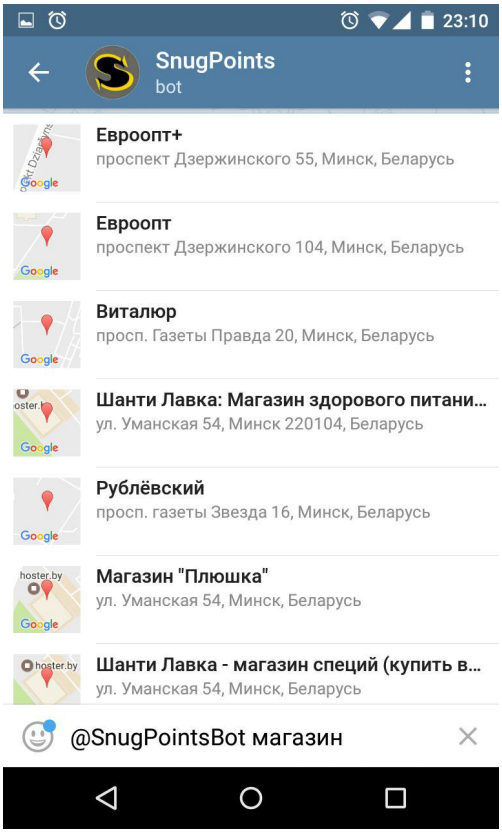
Глава 2. Интерфейс бота

Telegram уже имеет свой интерфейс диалогов, который нельзя изменить. Однако Telegram Bot API предоставляет дополнительные возможности для разработки интерфейса бота как встраиваемая клавиатура, пользовательская клавиатура, редактирование сообщений и т.д.

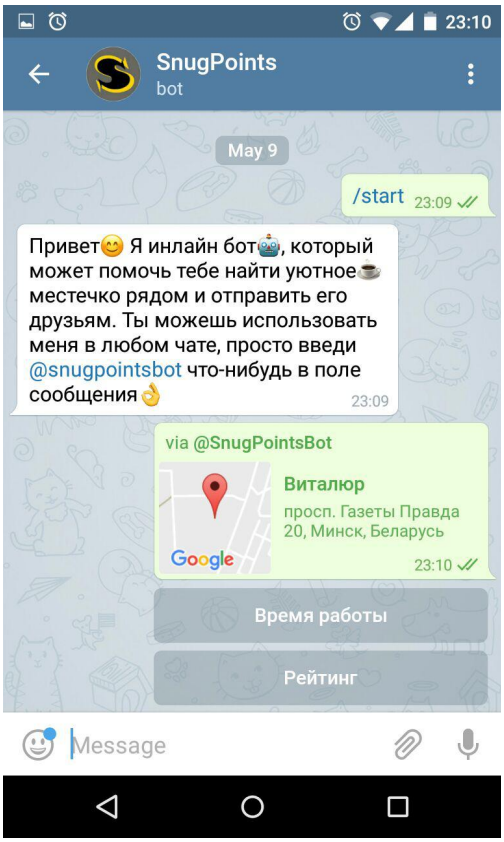
В Telegram бот не может первым начать взаимодействовать с пользователем, поэтому сначала бота надо найти по названию, а затем общение с любым ботом стандартно начинается с команды /start или клавиши, которую можно увидеть вместо поля для ввода текста



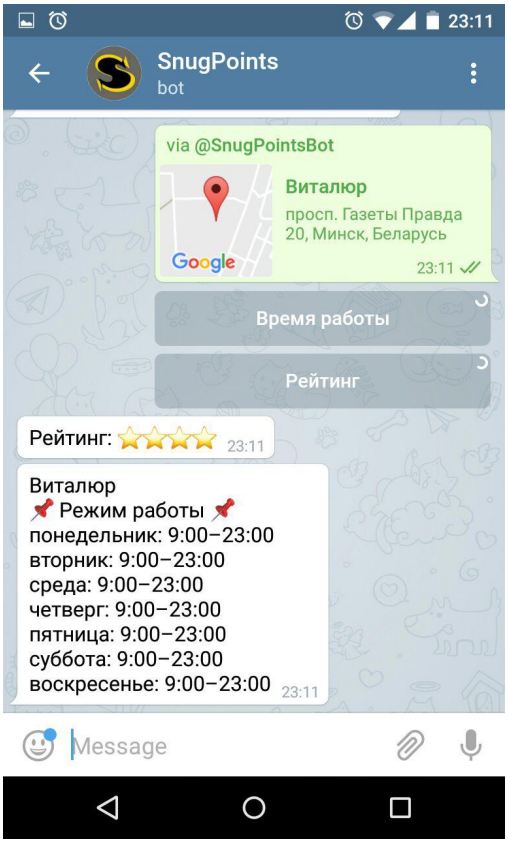
После отправки команды /start или нажатия на кнопку, пользователь получит приветствие от бота и инструкцию к нему. Для того чтобы использовать бота в дальнейшем, надо будет вводить @snugpointsbot в любом из чатов. Так же при вводе @snugpointsbot появляется подсказка, что делать дальше



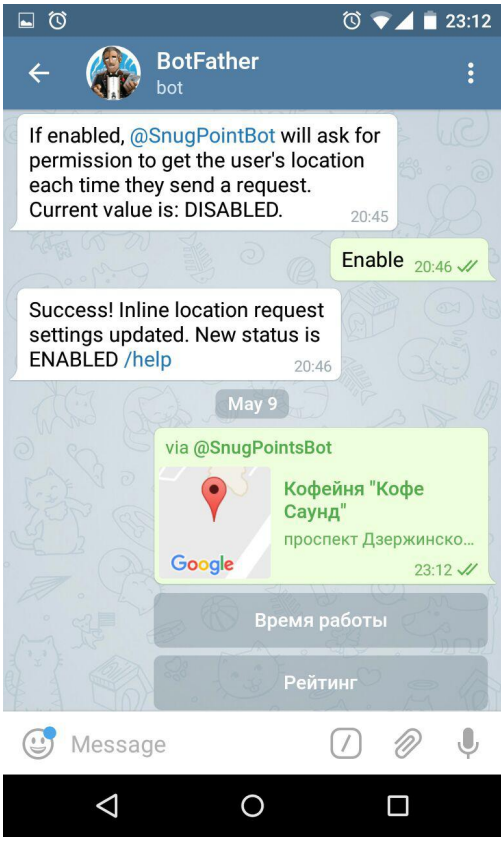
После ввода места или типа места, можно увидеть, подсказки мест, которые находятся в радиусе 1 километра от вас.



Когда пользователь нажмет на одну из подсказок, то бот пришлет в ответ ссылку на карту, адрес места и кнопки для получения дополнительной информации о заведении.



При нажатии на кнопку Время работы, пользователь получит расписание для данного места. Кнопка Рейтинг, пришлет рейтинг в звездочках. Если место ранее не оценивалось, то пользователь будет уведомлен об этом.



Как уже говорилось ранее, данного бота можно использовать и в других диалогах.

Глава 3. Технические моменты разработки Бота

3.1. Создание Бота в Telegram

Процесс создания ботов значительно отличается в зависимости от социальных сетей и мессенджеров. У Facebook есть несколько видов ботов, различающихся по уровню доступа. Для создания любого из них надо заполнить форму. Конкретно для бота первого уровня надо указать, на какую первую команду будет реагировать бот и как. Затем придет пару тестовых сообщений и бота подтвердят. Однако такой тип бота не подойдет для создания планируемого бота, как собственно и бот второго уровня для Facebook. Бот второго уровня используется для рассылок и оповещений подписанных пользователей. Боты третьего уровня доступны лишь для граждан Америки.

В Viber для того, чтобы создавать бота необходимо подать заявку на публичный аккаунт. Для этого надо заполнить формы: одну со своими данными, другую информацию о боте. После этого на электронную почту приходит уведомление, чаще всего с отказом. Если вашу заявку подтвердили, то письмо будет содержать ключ доступа к созданному аккаунту. Дополнительную информацию для разработчиков получить не удастся.

Самый простой процесс получения токена в Telegram. Для этого в мессенджере существует специальный бот – @BotFather (см. приложение 2). Надо написать ему /start и получить список всех его команд. Помимо этого /start является первой командой для начала взаимодействия с любым ботом. Далее надо использовать команду /newbot (см. приложение 3). BotFather предложит выбрать, как будут звать вашего бота, его можно назвать абсолютно по любому. Затем надо придумать имя пользователя, оно должно быть уникально. Когда вы выполните все действия, то бот будет создан, только он ничего не может делать. Чтобы «обучить» его чему-нибудь, можно использовать языки программирования. А можно и Paquebot – сервис для создания коммуникативных роботов.

Кроме этого в командах для BotFather можно найти настройки к ботам (Bot Settings). С помощью команды /setinline (см. приложение 4) можно включить режим встраивания, ваш бот будет работать как YouTube Search, Giphy GIF Search, а также можно написать подсказку, что надо вводить пользователю. Команда /setinlinegeo (см. приложение 5) позволит запрашивать у пользователей местоположение, чем и будем пользоваться для разработки SnugPointsBot.

Работа с приложением Telegram закончена, теперь переходим к программированию.

3.2. Принцип работы Telegram Bot API

Bot API это основанный на HTTP интерфейс, созданный для разработчиков с целью построения Telegram-ботов.

По сути, Telegram Боты специальные счета, которые не требуют дополнительного номера телефона, чтобы настроить. Пользователи могут взаимодействовать с ботами двумя способами:

- Отправка сообщений и команд для ботов, открыв чат с ними или путем добавления их в группы. Это полезно для чат-ботов или новостных ботов , как официальный TechCrunch и Forbes ботами.
- Отправлять запросы непосредственно из поля ввода, набрав @Username бота и запрос. Это позволяет отправлять контент от встроенных ботов непосредственно в любой чат, группы или канала.

Сообщения, команды и запросы, отправленные пользователями, передаются на сервера Telegram. Каждый бот получает уникальный ключ аутентификации при его создании, и выглядит примерно так “123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11”, но мы будем использовать просто <token>. Telegram сервер посредник, он обрабатывает все шифрование и связь с API Telegram. Пользователь общается с этим сервером через простой HTTPS-интерфейс, который предлагает упрощенную версию API телеграмм. Мы называем это интерфейс нашего Bot API.

Создание запросов

Все запросы к API Телеграммы Bot должны быть поданы через HTTPS и должны быть представлены в следующем виде: https://api.telegram.org/bot<token>/METHOD_NAME. Как этот, например:

https://api.telegram.org/bot123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11/getMe

Telegram поддерживает четыре способа передачи параметров в запросах Bot API:

- URL query string;
- application/x-www-form-urlencoded;
- application/json (кроме отправки файлов);
- multipart/form-data (используется для отправки файлов);

Ответ содержит объект JSON, который всегда имеет Boolean поле «ok» и может иметь дополнительное поле string «description» с удобочитаемым описанием результата. Если «ok» правда, запрос был успешным, и результат запроса можно найти в поле «result». В случае неудачного запроса «ok» ложь и ошибка объясняется в «description». Поле «error_code» Integer также возвращается, но его содержание может быть изменено в будущем. Некоторые ошибки могут также иметь необязательное поле «parameters» типа

ResponseParameters, которые могут помочь для автоматической обработки ошибок.

- Все методы в Bot API регистронезависимы.
- Все запросы должны быть сделаны с использованием UTF-8.

Получение обновлений

Есть два взаимоисключающих способа получения обновлений для бота getUpdates метод, с одной стороны, и Webhooks с другой. Входящие обновления сохраняются на сервере Telegram, пока боты не получают их так или иначе, но они не будут храниться дольше, чем 24 часа.

Независимо от того, какой вариант вы выбрали, вы получите JSON-serialize обновления объектов в качестве результата.

GetUpdates используется для получения входящих обновлений с помощью long polling. При long polling клиент запрашивает информацию с сервера с ожиданием, что сервер может не ответить сразу. Если у сервера нет новой информации для клиента при получении опроса, вместо отправки пустого ответа сервер держит запрос открытым и ждет, когда информация о реакции станет доступной. Как только у нее появляется новая информация, сервер немедленно отправляет клиенту ответ HTTP/S, завершая открытый HTTP/S-запрос. После получения ответа сервера клиент часто сразу же выдает другой запрос на сервер.

SetWebhook используется, чтобы, указав URL, получать входящие обновления с помощью исходящего webhook. Всякий раз, когда есть обновления для бота, Telegram вышлет запрос POST HTTPS к указанному URL, содержащему JSON-serialize Update.

Оба метода возвращают массив объектов Updates, который выглядит следующим образом:


```
{"ok":true,"result":[{"update_id":505245009,
"message":{"message_id":109,"from":
{"id":338846382,"first_name":"Nastik","last_name":"Kitsan"},"chat":
{"id":338846382,"first_name":"Nastik","last_name":"Kitsan","type":"private"},
"date":1494188558,"location":
{"latitude":53.858537,"longitude":27.462028},"venue":{"location":
{"latitude":53.858537,"longitude":27.462028},"title":"\u0410\u0442\u043b\u043
5\u0442\u0438\u0437\u043c \u0438
\u0433\u0440\u0430\u0446\u0438\u044f","address":"\u043f\u0440\u043e\u0441\u04
3f. \u0433\u0430\u0437\u0435\u0442\u044b
\u00ab\u0417\u0432\u044f\u0437\u0434\u0430\u00bb, 61"}}},
{"update_id":505245010,
"message":{"message_id":110,"from":
{"id":338846382,"first_name":"Nastik","last_name":"Kitsan"},"chat":
{"id":338846382,"first_name":"Nastik","last_name":"Kitsan","type":"private"},
"date":1494240238,"text":"/start","entities":
[{"type":"bot_command","offset":0,"length":6}]}},{"update_id":505245011,
"message":{"message_id":111,"from":
{"id":338846382,"first_name":"Nastik","last_name":"Kitsan"},"chat":
{"id":338846382,"first_name":"Nastik","last_name":"Kitsan","type":"private"},
"date":1494240254,"text":"/start","entities":
[{"type":"bot_command","offset":0,"length":6}]}}]}
```

Поле	**Тип**	**Описание**
update_id	Integer	Уникальный идентификатор объекта.
message	Message	*Необязательное*. Новое входящее сообщение любого типа – текст, стикер, фото и так далее
edited_message	Message	*Необязательное*. Новая версия сообщения, которое было отредактировано
inline_query	InlineQuery	*Необязательное*. Новый входящий встроенный запрос
chosen_inline_result	ChosenInlineResult	*Необязательное*. Результат строкового запроса, который был выбран пользователем и отослан чат-партнеру.
callback_query	CallbackQuery	*Необязательное*. Новый запрос входящего обратного вызова

Объект Updates представляет собой входящее обновление. В лучшем случае один из необязательных параметров может присутствовать в любой момент обновления.

3.3. Использование библиотеки pyTelegramBotAPI

В курсовом проекте будем писать бота с помощью Python, так как это один из языков, для которого создано множество библиотек. Для работы с Telegram API надо установить библиотеку pyTelegramBotAPI, это можно сделать с помощью pip:
pip install pyTelegramBotAPI

Класс TeleBot (определяется в _init_.py) инкапсулирует все вызовы API в одном классе. Он обеспечивает такие функции, как send_x (send_message, send_document и т.д.) и несколько способов для получения входящих сообщений.

Для создания бота, надо создать экземпляр класса TeleBot, инициализируем его:

```
bot = telebot.TeleBot(
"390635267:AAFITwOnH_oMHM9auAz-bSTSpAMRbwI5Goc ")
```

После этого заявления, мы должны зарегистрировать некоторые так называемые хэндлеры (обработчики сообщений). Обработчики сообщений определяют фильтры для сообщений, которые должны пройти. Если сообщение проходит фильтр, декорированная функция вызывается, и входящее сообщение передается в качестве аргумента:

```
@bot.message_handler(commands=['start'])
def instruction_to_bot(message):
    bot.send_message(message.chat.id, text=intro)
```

Данный пример из SnugPointsBot. Функция реагирует только на сообщения содержащие команду /start.

Также можно создать функцию, которая реагирует на вызовы из встраиваемой клавиатуры:

```
@bot.callback_query_handler(func=lambda call: True)
def callback_inline(call):
    bot.send_message(chat_id=call.from_user.id, text=open_hours(call.data))
```

Данная функция после Данная функция после нажатия клавиши на встраиваемой клавиатуре отправит пользователю сообщение с режимом работы заведения на текущий день.

Так как SnugPointsBot – встраиваемый бот, то надо разобраться с тем, как работает inline режим.

Следующие методы и объекты позволяют ботам работать в инлайн режиме. Чтобы включить данный режим, надо отправить /setinline команду @BotFather и предоставить шаблонный текст, который пользователь увидит в поле ввода после ввода имени бота.

InlineQueryResult объект представляет собой один результат строкового запроса. Клиенты Telegram в настоящее время поддерживают результаты следующих 20 видов:

- InlineQueryResultCachedAudio
- InlineQueryResultCachedDocument
- InlineQueryResultCachedGif
- InlineQueryResultCachedMpeg4Gif
- InlineQueryResultCachedPhoto
- InlineQueryResultCachedSticker
- InlineQueryResultCachedVideo
- InlineQueryResultCachedVoice
- InlineQueryResultArticle
- InlineQueryResultAudio
- InlineQueryResultContact
- InlineQueryResultGame
- InlineQueryResultDocument
- InlineQueryResultGif
- InlineQueryResultLocation
- InlineQueryResultMpeg4Gif
- InlineQueryResultPhoto
- InlineQueryResultVenue
- InlineQueryResultVideo
- InlineQueryResultVoice

Чтобы обрабатывать результаты запросов, поставим инлайн декоратор. Затем для использования метода answerInlineQuery, который будет отправлять результаты запросов, нам надо сформировать массив данных, который будет отправляться пользователю. В нашем случае это будет массив мест, а места в свою очередь будут содержать название, координаты, адрес и клавиатуру для получения дополнительной информации о месте. Нам для этого понадобится InlineQueryResultVenue:

```
@bot.inline_handler(func=lambda query: len(query.query) > 0)
    places = info_about_place(query.query, query.location.latitude,
    query.location.longitude)
    i_place = []
```

```
    for place in places['results']:
        n_id = str(i)
        keyboard = types.InlineKeyboardMarkup()
        keyboard.add(types.InlineKeyboardButton('Время работы',
        callback_data=place['place_id']))
        i_place.append(types.InlineQueryResultVenue(
        id=n_id, title=place['name'], latitude=place['geometry']['location']['lat'],
        longitude=place['geometry']['location']['lng'],
        address=address_info(place['place_id']),
        reply_markup=keyboard
        ))
    bot.answer_inline_query(query.id, i_place)
```

В данной части кода использовались такие функции, как address_info(), info_about_place(), о которых можно прочитать в 3.4.Использование Places API Web Service. Также здесь используется метод InlineKeyboardMarkup(). InlineKeyboardMarkup – представляет собой встроенную клавиатуру, которая появляется рядом с сообщением. Она содержит массив кнопок-строк, каждая из которых представлена массивом InlineKeyboardButton объектов. В свою очередь объект InlineKeyboardButton представляет собой одну кнопку инлайн клавиатуры. Первый параметр метода кнопки является надписью на ней, параметр callback_data – данные, которые будут отправлены после нажатия кнопки.

3.4.Использование Places API Web Service

Получать информацию для формирования InlineQueryResultVenue будем с помощью Places API Web Service.

Подобные запросы позволяют найти все места в выбранной области. Чтобы уточнить запрос, можно указать ключевые слова или тип мест.

Запрос поиска мест поблизости – это URL-адрес в формате HTTP, который выглядит следующим образом:

https://maps.googleapis.com/maps/api/place/nearbysearch/output?parameters

Output может принимать одно из следующих значений:

- Json – задает вывод в формате JavaScript Object Notation (JSON)
- Xml – задает вывод в формате XML

Чтобы инициировать запрос поиска мест поблизости, необходимо указать определенные параметры. Обязательными параметрами являются:

- Key – ключ API приложения.
- Location – широта и долгота точки, вокруг которой выполняется поиск мест.
- Radius – расстояние в метрах, в пределах которого должны находиться найденные результаты.

Также в запросе в курсовом проекте используются такие дополнительные параметры:

- Name – термин, который следует сопоставить со всем контентом, индексированным Google для этого местоположения.
- Language – код языка, на котором следует по возможности возвращать результаты.

Ответы на поисковые запросы возвращаются в формате, указанном с помощью флага output в URL-адресе запроса и выглядят таким образом:


```
{
   html_attributions" : [],
   results" : [
   {
      "geometry" : {
         "location" : {
            "lat" : -33.870775,
            "lng" : 151.199025
         }
      },
      "icon" : "http://maps.gstatic.com/mapfiles/place_api/icons/travel_agent-71.png",
      "id" : "21a0b251c9b8392186142c798263e289fe45b4aa",
      "name" : "Rhythmboat Cruises",
```

```
            "opening_hours" : {
                  "open_now" : true
            },
            "photos" : [
            {
                  "height" : 270,
                  "html_attributions" : [],
                  "photo_reference"                  :                  "CnRnAAAAF-
            LjFR1ZV93eawe1cU_3QNMCNmaGkowY7CnOf-
            kcNmPhNnPEG9W979jOuJJ1sGr75rhD5hqKzjD8vbMbSsRnq_Ni3ZIGfY6hKWmsOf3qH
            KJInkm4h55lzvLAXJVc-
            Rr4kI9O1tmIblblUpg2oqoq8RIQRMQJhFsTr5s9haxQ07EQHxoUO0ICubVFGYfJiMUPor1
            GnIWb5i8",
                  "width" : 519
            }
            ],
            "place_id" : "ChIJyWEHuEmuEmsRm9hTkapTCrk",
            "scope" : "GOOGLE",
            "alt_ids" : [
            {
                  "place_id" : "D9iJyWEHuEmuEmsRm9hTkapTCrk",
                   "scope" : "APP"
            }
            ],
            "reference"      :      "CoQBdQAAAFSiijw5-cAV68xdf2O18pKIZ0seJh03u9h9wk_lEdG-
            cP1dWvp_QGS4SNCBMk_fB06YRsfMrNkINtPez22p5lRIlj5ty_HmcNwcl6GZXbD2RdXs
            VfLYlQwnZQcnu7ihkjZp_2gk1-fWXql3GQ8-1BEGwgCxG-
            eaSnIJIBPuIpihEhAY1WYdxPvOWsPnb2-
            nGb6QGhTipN0lgaLpQTnkcMeAIEvCsSa0Ww",
            "types" : [ "travel_agency", "restaurant", "food", "establishment" ],
            "vicinity" : "Pyrmont Bay Wharf Darling Dr, Sydney"
      }
      ],
      "status" : "OK"
}
```


Из данного запроса мы возьмем адрес, координаты местоположения, название и идентификатор места для получения дополнительной информации об этом месте.

Запрос данных о месте – это URL-адрес в формате HTTP, который выглядит следующим образом:

https://maps.googleapis.com/maps/api/place/details/output?parameters

Формат вывода данных и параметры аналогичны как при запросе поиска мест.

Данные запросы выполняются в таких функциях, как info_about_place, open_hours, address_info.

ЗАКЛЮЧЕНИЕ

Конечной целью курсовой работы являлось создание Telegram-бота (SnugPointsBot), который будет использоваться не только для определения ближайшего общественного места к пользователю, но и способного, основываясь на его запросе, представить интересующую информацию о работе, рейтинге заведения.

Первоначальной задачей было найти наиболее удобную и функциональную платформу для размещения бота. По итогу изучения и сравнения таких платформ как Facebook, Viber и Telegram предпочтение было отдано последнему в связи с самым удобным способом регистрации аккаунта для бота и широким набором инструментов для разработчиков.

Далее были изучены основные методы взаимодействия человека с ботом и схема работы Telegram API. Также протестированы различные методы класса TeleBot. Учитывая большое количество инструкций по созданию ботов в интернете довольно легко получить начальные навыки. Но чем сложнее бот, тем сложнее и взаимодействие с ним, поэтому знаний одного языка программирования может и не хватить, так как может возникнуть вероятность организации схемы диалога.

Places API Web Service также имеет хорошо организованную инструкцию для разработчиков, однако могут возникнуть проблемы с количеством запросов без дополнительного вложения средств.

Интерфейс бота был разработан быстро благодаря разработчикам Telegram, которые продумали основные элементы интерфейса как встраиваемые клавиатуры, боты, использование основных типов данных (фото, текст, координаты и т.д.)

Конечным этапом разработки бота было развертывание его на сервере. Наиболее удачный выбор для этого – DigitalOcean. Для его использования есть много инструкций, а оплата производится не за место, а за вычислительную мощность.

Все поставленные задачи были выполнены, бот получился более функциональным, чем его конкурент Foursquare, как и планировалось. Однако при создании ботов, выдающих информацию основываясь на геолокации, важную роль играют источники данных. Ни у SnugPointsBot, ни у Foursquare нет хороших баз данных, чтобы выдавать наиболее информационно-полные результаты запросов.

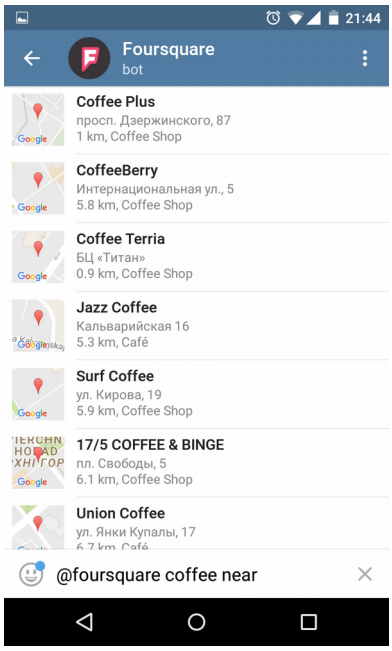
Также стоит отметить, что создание разработчиками специальных инструментов для работы с социальными сетями и мессенджерами является отличным решением. Во-первых, разработчику-любителю не стоит много времени уделять интерфейсу и взаимодействиям частей приложения. Во-вторых,

идет привлечение и наполнение платформы без дополнительных усилий от ее создателей.

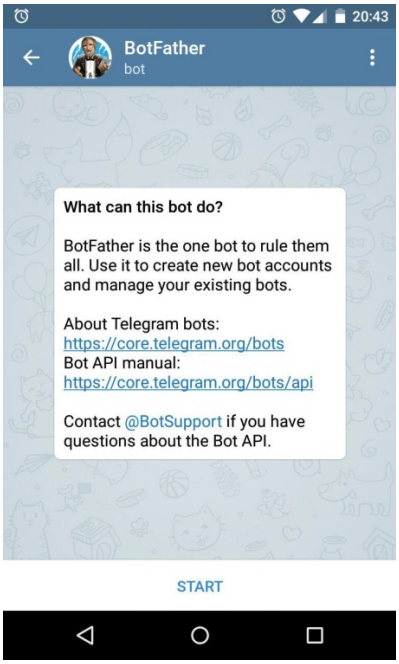
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Places API Web Service [Ссылка на ресурс] https://developers.google.com/places/web-service/search?hl=ru
2. Bots: An introduction for developers [Ссылка на ресурс] https://core.telegram.org/bots
3. pyTelegramBotAPI [Ссылка на ресурс] https://github.com/eternnoir/pyTelegramBotAPI
4. Урок 7. Встраиваемые боты (Inline) [Ссылка на ресурс] https://groosha.gitbooks.io/telegram-bot-lessons/content/chapter7.html
5. API Карт/Документация [Ссылка на ресурс] https://tech.yandex.ru/maps/doc/geosearch/concepts/request-docpage/
6. Как создать своего бота для Telegram за 10 минут [Ссылка на ресурс] https://www.iphones.ru/iNotes/560747
7. Инструкция: Как создавать ботов в Telegram [Ссылка на ресурс] https://habrahabr.ru/post/262247/

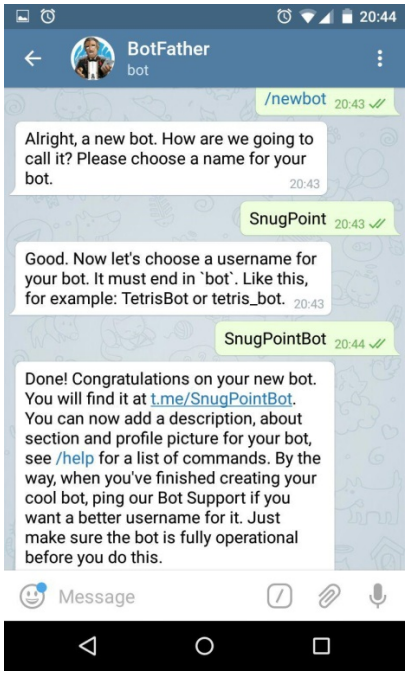
Приложения



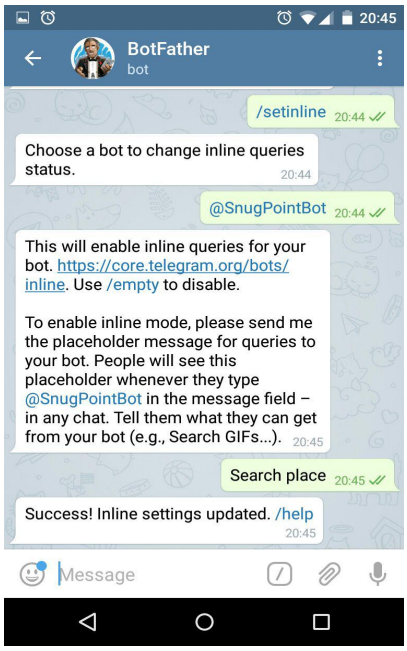
приложение 1



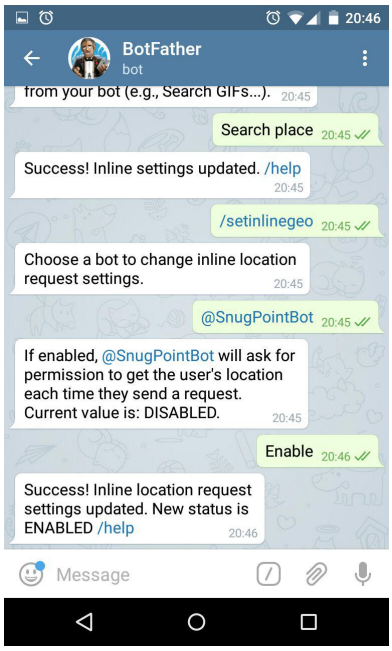
приложение 2



приложение 3



приложение 4



приложение 5