Лабораторная работа №7: Создание telegram-бота с расписанием

Техническое задание:

Создать телеграм-бота с расписанием для Вашей группы.

Минимальные требования к разрабатываемой системе:

1. Бот должен иметь ник-нейм формата <номер группы>_<фамилия разработчика>_bot.

2. Бот должен иметь имя формата <номер группы>_<фамилия разработчика>.

3. Манера общения бота – "на Вы".

4. Использование базы данных PostgreSQL.

5. Использование pyTelegramBotAPI.

6. Использование адаптера psycopg2.

7. Формат вывода на каждый день недели:

   <День недели>
   _____
   <Предмет> <Кабинет> <Время> <Преподаватель>
   …
   <Предмет> <Кабинет> <Время> <Преподаватель>
   _____

8. Во время работы бота должны быть отображены следующие графические кнопки:

   a. Понедельник
   b. Вторник
   c. Среда
   d. Четверг
   e. Пятница
   f. Расписание на текущую неделю
   g. Расписание на следующую неделю

9. При нажатии на кнопку с днем недели бот должен выводить информацию из базы данных с расписанием на выбранный день текущей недели.

10. При нажатии на кнопку "Расписание на текущую неделю" бот должен выводить информацию из базы данных с расписанием на всю текущую неделю.

11. При нажатии на кнопку "Расписание на следующую неделю" бот должен выводить информацию из базы данных с расписанием на всю следующую неделю неделю.

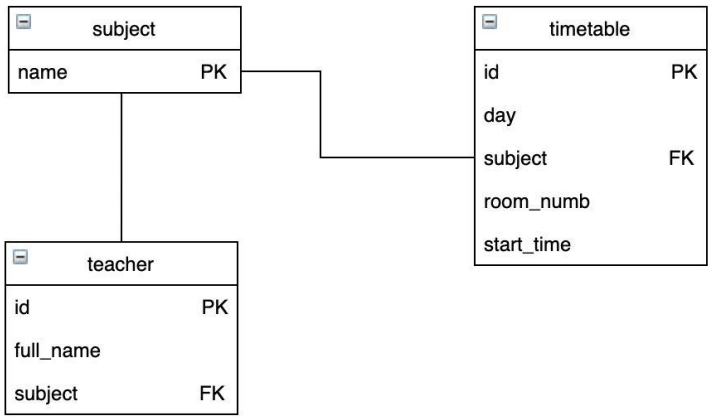
12. При использовании команды /week бот должен выводить какая на данный момент неделя – верхняя/нижняя.

13. При использовании команды /mtuci бот должен выводить ссылку на официальный сайт МТУСИ – https://mtuci.ru/

14. При использовании команды /help бот должен выводить краткую информацию о себе, краткую документацию и список команд с их пояснениями.

15. При вводе неизвестной команды или неизвестного боту сообщения, бот должен отправлять пользователю сообщение – "Извините, я Вас не понял".

16. Структура базы данных может дополняться полями в таблицах, но при этом должна иметь следующую структуру:



Система может быть дополнена Вашим функционалом, но должна соответствовать минимальным требованиям.

Создание простого Telegram-бота

1. **Создаем новый проект с именем Simple-bot**
2. **Загружаем библиотеку для создания Telegram-ботов**

   pip3 install pyTelegramBotAPI

3. **Регистрируем бота в Telegram**

   - В поисковой строке ищем бота с именем @BotFather

   - Отправляем боту команду/сообщение – /start
   - Отправляем боту команду/сообщение с именем бота, которого хотим зарегистрировать, в нашем случае <Ваша фамилия>_<Ваше имя>_bot
   - Для ознакомления с перечнем команд @BotFather отправляем боту команду/сообщение – /help

4. **В файл main.py импортируем библиотеки для создания back-end части бота**

```
import telebot
from telebot import types
```

5. **Из сообщения об успешной регистрации бота копируем токен для управления нашим ботом. В файле main.py оздаем переменную, хранящую в себе токен**

```
token = "Ваш токен"
```

6. **Создаем объект бота, к которому мы будем в дальнейшем обращаться**

```
bot = telebot.TeleBot(token)
```

   В класс TeleBot передаем переменную token, для того, чтобы обращаться именно к вашему боту

7. **Создаем декоратор, отвечающий за команду /start**

```
@bot.message_handler(commands=['start'])
def start(message):
  keyboard = types.ReplyKeyboardMarkup()
```

```
  keyboard.row("Хочу", "/help")
  bot.send_message(message.chat.id, 'Привет! Хочешь узнать свежую информацию
о МТУСИ?', reply_markup=keyboard)
```

Класс ReplyKeyboardMarkup создает пользовательскую клавиатуру с текстовыми кнопками на месте стандартной клавиатуры.
Метод row() заполняет клавиатуру кнопками.
Метод send_message отправляет пользователю сообщение.

Аргумент message.chat.id используется для того, чтобы бот отправил сообщение тому пользователю, который отправил сообщение, на которое бот в данный момент времени отвечает.
Аргумент reply_markup=keyboard используется для отправки пользовательской клавиатуры, для ее дальнейшего отображения.

8. **Создаем декоратор отвечающий за команду /help**

```
@bot.message_handler(commands=['help'])
def start_message(message):
  bot.send_message(message.chat.id, 'Я умею...')
```

В сообщении вы можете указать что умеет бот, включая команды, на которые он умеет реагировать.

9. **Создаем декоратор отвечающий за ответ на сообщение "Хочу"**

```
@bot.message_handler(content_types=['text'])
def answer(message):
  if message.text.lower() == "хочу":
    bot.send_message(message.chat.id, 'Тогда тебе сюда - https://mtuci.ru/')
```

Данный декоратор должен стоять ниже, чем декораторы команд, так как в противном случае декораторы команд обрабатываться не будут, потому что команды в своем роде тоже текстовые сообщения.
В этом декораторе аргумент content_types=['text'] отвечает за реакцию на текстовый тип контента сообщения.
Для проверки конкретного текста используется условная конструкция с условием message.text.lower() == "<текст>". Причем функция lower() отвечает за перевод текста в нижний регистр для удобства использования, и может применяться не только для библиотеки telebot, но и для любых строковых операций и переменных.

Домашнее задание:

- Создать обработку трех любых сообщений.
- Создать обработку трех любых команд.
- Обработать команду /help