Университет Правительства Москвы

VII Конкурс исследовательских и проектных работ обучающихся образовательных организаций города Москвы и Московской области
«Мегаполис XXI века – город для жизни» в 2022/2023 учебном году

Конкурсная работа
На тему: «Создание чат-бота на языке Python»

Выполнена: учащимися 7 класса
ГБОУ города Москвы «Школа 1363»
Егоровой Екатериной Сергеевной,
Стаурской Валерией Алексеевной

Научный руководитель работы:
Анохина Дарья Сергеевна

Директор ГБОУ Школы №1363
Лавриненко Елена Валерьевна
Подпись

г. Москва

2022

Паспорт проекта

Название проекта: “Создание чат-бота на языке python”

Руководитель проекта: Анохина Дарья Сергеевна

Авторы проекта: Егорова Екатерина Сергеевна и

Стаурская Валерия Алексеевна

Учебная дисциплина: Информатика

Тип проекта: Материальный

Цель работы: разработать бота, с помощью которого можно просматривать, добавлять, удалять оценки и домашнее задание, а также просматривать расписание звонков

Задачи работы: определить материалы для реализации проекта, зарегистрировать чат-бота в Телеграм, написать программный код для бота, запустить бота с помощью облачного сервера

Краткое содержание проекта: В связи с возросшей популярностью и удобством использования чат-ботов, мы решили создать бот в Telegram на языке python. Были определены материалы для создания проекта и написан программный код. В итоге получился чат-бот, с помощью которого можно просматривать, добавлять, удалять оценки и домашнее задание, а также просматривать расписание звонков

Результат проекта (продукт): чат-бот в Telegram на языке python для образовательных целей «@Analogoviybot»

Реализация: регистрация чат-бот в Телеграм с помощью BotFather, подготовка к написанию программного кода, написание программного кода на языке python, подключение бота к облачному серверу

Оглавление

Введение

Чат-бот — это программа, которая имитирует реальный разговор с пользователем. Они позволяют общаться с помощью текстовых или аудио сообщений на сайтах, в мессенджерах, мобильных приложениях или по телефону. Боты могут быть очень простыми и обрабатывать всего несколько команд или сложными, например цифровые помощники и интерактивные агенты. Чат-боты могут быть частью приложений или полностью автономными. Их можно использовать не только для огромных предприятий, а также для решения проблем, с которыми мы сталкиваемся каждый день, например отсутствие возможности у преподавателей установления быстрого контакта с учениками и передачи информации большому количеству лиц на значительное расстояние. С этой трудностью может с лёгкостью справиться чат-бот, способный обращаться к базе данных.

Актуальность темы

На данный момент чат-боты имеют огромную популярность и продолжают её набирать. По данным «**Consumer News and Business Channel**» (CNBC) в 2022 году чат-боты сократили расходы бизнеса на 8 миллиардов долларов. По данным **Sales Force** 69% пользователей предпочитают общение с ботами, потому что они могут получить ответы от бренда с удобной для себя скоростью. Роботу не сложно ответить 100 пользователям одновременно, а менеджера придётся ждать. Из этого можно сделать вывод, что чат-бот – актуальное решение проблем многих бизнес-компаний, связанных с общением с клиентом напрямую.

Цель проекта

Мы решили создать бота, с помощью которого можно просматривать, добавлять, удалять оценки и домашнее задание, а также просматривать расписание звонков.

Задачи

1. Определить материалы для создания
2. Зарегистрировать чат-бота в Телеграм
3. Написать программный код для бота
4. Запустить бота с помощью облачного сервера

История развития

Прообразом чат-ботов стал известный тест Тьюринга, появившийся в 1950 году. Он заключался в том, что человек должен был определить в диалоге между машиной и другим человеком, где компьютер, а где живой собеседник. В 1966 году Джозеф Вейценбаум создал программу-психиатра Элизу, которая строила диалог по ключевым фразам и могла каждый раз создавать новую ветку беседы по слову из реплики пользователя. Продолжил тему психиатрических экспериментов робот Парри, который имитировал поведение больного шизофренией и служил для обучения студентов медицинских вузов. С 1988 года благодаря Ролло Карпентеру и его **JabberWacky** диалоговые программы стали частью развлечений. В 1991 году в Сингапуре создали компьютерного психолога **Dr.Sbaitso**, а в 1995 году появился первый проект с открытым исходным кодом — алгоритм с женским образом **A.L.I.C.E**. 2010 год стал стартом для развития виртуальных голосовых помощников от ведущих мировых IT-брендов. Один за другим в сети появлялись **Siri**, **Cortana**, **Alexa**, Алиса, ассистенты Салют. А с 2016 года простые диалоговые программы стали набирать популярность в бизнесе.

Зачем использовать чат ботов?

Чат-боты помогают пользователям взаимодействовать с технологиями и автоматизировать задачи. Достижения в области ИИ, машинного обучения, обработки естественного языка и анализа данных способствовали их стремительному распространению. Чат-ботов начали активно использовать в совершенно разных целях благодаря простоте создания и преимуществам для бизнеса, клиентов и сотрудников.

Чат-боты предоставляют компаниям множество преимуществ. Их можно использовать в качестве виртуальных агентов для обслуживания клиентов и обработки обращений в службу поддержки. Улучшение качества обслуживания и сокращения расходов повышает рентабельность инвестиций.

Чат-боты на базе ИИ ускоряют цикл продаж, а также помогают привлекать больше потенциальных клиентов и повышать лояльность к бренду. С их помощью компании могут персонализировать обслуживание, чтобы повысить вовлеченность клиентов, их удовлетворенность и коэффициент конверсии.

Высокая рентабельность инвестиций — не единственная причина, по которой компании начали активно внедрять чат-ботов. Клиенты и сотрудники любят их за простоту и удобство. Организации начали внедрять все более сложные технологии и использовать различные каналы взаимодействия. Ввиду этого чат-боты быстро стали необходимым связующим звеном между людьми и большими объемами информации, системами и приложениями.

Чат-боты также предоставляют ощутимые преимущества клиентам. Они выводят обслуживание на новый уровень без каких-либо ограничений. Клиенты могут круглосуточно получать ответы на интересующие их вопросы. Кроме того, чат-боты упрощают покупки, а также персонализируют взаимодействие с брендами.

Как работают чат-боты?

Чат-бот — это приложение, с которыми пользователи могут взаимодействовать в режиме диалога с помощью текста, графических объектов или речи. Существуют разные типы чат-ботов, но принцип их работы примерно одинаков.

Шаг 1

Пользователь отправляет чат-боту текстовое или голосовое сообщение через любой канал, например приложение или веб-сайт. Сообщение может быть в формате команды или вопроса.

Шаг 2

Чат-бот получает сообщение и записывает связанную информацию, например канал взаимодействия, а затем с помощью алгоритмов NLP определяет цель сообщения и намерения.

Шаг 3

Чат-бот определяет соответствующий ответ и возвращает его пользователю через тот же канал. Остальной диалог с чат-ботом происходит по тому же принципу. Беседа продолжается до тех пор, пока пользователь не получит нужный ответ или пока обращение не переадресовывается агенту.

Подготовка к созданию

Сначала нам стоит определить тип нашего чат-бота, чтобы в дальнейшем подобрать нужные материалы и ресурсы для создания. Алгоритмы работы программ бывают двух типов:

- **основанные на правилах;**
- **созданные с помощью искусственного интеллекта.**

Первая группа для общения с пользователями задействует ключевые слова и фразы, прописанные в разработанном сценарии. На экране могут присутствовать кнопки-подсказки с такими словами. В ответ на запрос робот даёт запрограммированное сообщение. Если решение не нашлось, алгоритм переключает беседу на реального оператора. Это быстрый и простой способ

автоматизации рутинных процессов в бизнесе. Именно такую форму можно встретить на большинстве площадок, где различные компании коммуницируют с аудиторией. **Вторая группа** связана со сложными технологиями машинного обучения и искусственного интеллекта. При разработке используются техники NLP, синтеза и распознавания речи. За работу такого виртуального помощника отвечает нейронная сеть, которая способна обучаться. Для создания голосового бота нужен большой объём данных. Также можно выделить еще одну классификацию чат-ботов: **текстовые, голосовые** и **универсальные**. Эта классификация основана на формате работы чат-бота – каким именно способом система обрабатывает запрос и выдает результат.

Текстовый бот может анализировать текстовые запросы пользователя и выдавать результат в соответствии с командой (например, “@PreAmbulaBot” от ПреАмбула).

Голосовой бот умеет распознавать речь человека и действует в соответствии с репликами собеседника (например, “Афина” от СберБанк).

Универсальный бот способен анализировать текстовые и голосовые запросы и отвечать как письменно, так и устно (например, “Алиса” от Яндекс).

Мы будем использовать текстового бота, основанного на правилах, для экономии времени и ресурсов. Теперь разберёмся со способами создания. Основных способов всего 2: **с кодом** и **без кода**.

Первый способ – наиболее сложный в плане реализации. Он требует определенных навыков и знаний в области программирования, однако открывает широкие возможности по кастомизации бота.

Второй способ – более простой. Он заключается в создании бота с помощью специальных платформ (конструкторов). В этом случае вся работа будет сводиться к выбору подходящих блоков, настройке связи между ними, прописыванию ключевых запросов и так далее. После этого останется лишь подключить бота к выбранной площадке. К сожалению, создание ботов с помощью конструктора ограничивает нас в функциях, поэтому мы создадим бота с помощью кода. Итак, мы разобрались с основными характеристиками нашего чат-бота, теперь можно перейти к созданию

Начало создания. Регистрация бота

Сначала нужно зарегистрировать нашего чат-бота в Телеграм и получить токен. Для этого нужно зайти в данное приложение и ввести в строку поиска BotFather, после чего выбрать бота и запустить его (У официального бота Telegram будет стоять синий подтверждающий знак возле имени в виде галочки). После запуска мы получаем список команд для управления ботом. Выбираем «**/newbot**», вводим имя и никнейм нашего чат-бота (Analogoviybot). После ввода

данных нам приходит сообщение со ссылкой на бота (t.me/<никнейм_бота>) и токен, который мы будем использовать при написании кода.

Теперь мы можем добавить нашему чат-боту описание (выбрав команду «**/setabouttext**» или «**/setdescription**», после чего отправив текст описания) и аватар (выбрав команду «**/setuserpic**», после чего отправив картинку для аватара), при этом указав никнейм нашего бота. Теперь, когда у нас есть токен, мы можем начать подготовку к написанию кода.

Подготовка к написанию кода

Нам необходимо подготовить файлы для нашего проекта. Создаём папку, назовём её **project**, открываем её. Установим нужные нам библиотеки, для этого в ней зажимаем Shift, кликаем правой кнопкой мыши и открываем командную строку или PowerShell, после чего вводим в командную строку «**python -m venv**», даём имя нашему виртуальному окружению – «**venv**», запускаем. В **project** появилась папка venv. В ней нам нужно активировать скрипты, для этого вводим в командную строку «**venv\Scripts\activate**», запускаем. Далее вызываем пакетный менеджер pip и загружаем aiogram: «**pip install aiogram»**. После установки загружаем ещё одну библиотеку: «**pip install sqlalchemy**».

Теперь в папке **project** создаём файл с расширением **py**, в котором будет храниться основной код. Назовём его **bot_telegram.py**.

Для того, чтобы чат-бот функционировал, нам нужен сервер, но перезапускать сервер после каждого изменения кода для проверки довольно трудно, поэтому, как временную альтернативу, мы будем использовать файл с расширением **bat**. Назовём его **bot_run.bat**. Заходим в Visual Studio Code или в любой другой текстовый редактор, открываем в нём **bot_run.bat**. В первую строку вводим «**@echo off**», чтобы служебная информация самого **bat** файла не спамила нам в консоль. Теперь активируем виртуальное окружение: «**call %~dp0project\venv\Scripts\activate**». Командной строкой переходим в файл с проектом: «**cd %~dp0project**». Немного ранее мы получили токен для управления нашим чат-ботом. Теперь нам нужно ввести: «**set TOKEN=<значение_токена>**» Запускаем скрипт: «**python bot_telegram.py**». Затем пишем **pause**, чтобы в случае ошибки окно не закрылось автоматически, и мы могли бы её исправить. Код для **bot_run.bat** написан. Теперь мы можем запускать нашего бота, просто кликнув на него. На данный момент подготовка к написанию кода завершена, можно приступать к непосредственному написанию кода.

Написание кода

Заходим в Visual Studio Code и открываем папку **project**. Слева выбираем **bot_telegram.py**. Начнём писать код.

Сначала импортируем нужные нам команды из библиотеки aiogram. Вводим: «**from aiogram import Bot, Dispatcher, executor, types**», где **types** – специальные типы данных для написания аннотации типов в наших функциях, а также: «**import markup as nav**», «**import db**» (это потребуется нам позже). Далее мы пишем: «**import os**», чтобы импортировать токен нашего бота следующим образом: «**TOKEN = os.environ['TELEGRAM_TOKEN']**». Токен обозначен. Пишем: «**bot = Bot(token=TOKEN)**», и на следующей строке вводим: «**dp = Dispatcher(bot)**». Теперь наш файл подключён к чат-боту. На следующей мы зададим переменную **WRITE_USERS**, которая нам пригодится немного позже.

Теперь мы определим действия нашего бота при его запуске. Вводим: «**@dp.message_handler(commands=['start'])**» и «**async def command_start (message: types.Message):**». Сделаем так, чтобы бот здоровался с пользователем, используя при этом имя, который пользователь указал как свой никнейм. Для этого используем **0.first_name**:

«**await bot.send_message(message.from_user.id, 'Привет, {0.first_name}!'.format(message.from_user), reply_markup = nav.main)**», где **reply_markup** – запуск меню, которое мы добавим позже. Далее, для возможности изменения домашнего задания и оценок, мы напишем ID пользователя: «**await bot.send_message(message.from_user.id, 'Ваш ID {0.id}.\nЧтобы вы хотели знать?'.format(message.from_user))**». Это сделано для того, чтобы ученик не смог их изменить, и возможность модификации была только у аккаунта учителя. Таким образом, когда мы узнаём ID учителя, мы вводим его в переменную **WRITE_USERS** в следующем формате: «**WRITE_USERS = [id_учителя1, id_учителя2]**». Минус данного способа в том, что возможности добавить учителя в базу данных как модератора без вмешательства в код нет, но в отличии от простейшей аутентификации учителю не придётся каждый раз подтверждать свою личность перед добавлением домашнего задания или оценки.

Теперь добавим возможность просматривать расписание звонков. Так как расписание у нас будет в виде картинке, мы создаём папку **img** в папке **project** и загружаем нужную картинку с названием **schedule.png**. Завершив цикл, пишем на следующей строчке в нашем коде: «**@dp.message_handler()**», «**async def bot_message(message: types.Message):**», тем самым обозначив наличие новых команд. Вводим: «**if 'асписани' in message.text:**». Мы ввели лишь часть слова «Расписание», чтобы пользователь мог ввести это слово как с большой, так и с маленькой буквы. Далее задаём переменную **photo**: «**photo = open('img/schedule.png', 'rb')**», где **rb** - дата. Теперь вводим: «**await bot.send_photo(message.from_user.id, photo**)», где **photo** – переменная, созданная нами ранее. Теперь наш бот будет отправлять картинку при запросе, который включает в себя «асписани». Пока что мы остановимся с написанием кода в нашем основном файле.

Добавление базы данных

В папке **project** создаём файл с названием **models.py**. В нём мы создадим классы для базы данных. Импортируем нужные нам модули: «**from sqlalchemy.ext.declarative import declarative_base**», «**from sqlalchemy import Column, Integer, String, Date**». Создаём переменную **Base** для удобства: «**Base = declarative_base()**». Начинаем создавать класс оценок: «**class Mark(Base):**», «**__tablename__ = 'marks'**». Теперь создадим переменную **id**, так как она потребуется нам при удалении оценки: «**id = Column(Integer, primary_key=True)**», переменную **date**: «**date = Column(String(100))**», переменную **pupil_name**: «**pupil_name = Column(String(100), nullable=False)**», переменную **subject**: «**subject = Column(String(255), nullable=False)**», переменную **mark**: «**mark = Column(Integer, nullable=False)**». Далее создадим класс домашних заданий. В нём будет всё практически идентично классу оценок, но **__tablename__** будет равен **homeworks**, переменной **pupil_name** не будет.

В папке **project** создаём файл с названием **db.py**. Импортируем классы и **Base**: «**from model.models import Base, Mark, Homework**». Импортируем нужные нам команды: «**from sqlalchemy import create_engine**» и «**from sqlalchemy.orm import sessionmaker**». Теперь нам нужно создать саму базу данных, в которой будет храниться информация о сохранённых оценках и заданиях. Для этого вводим: « **engine = create_engine('sqlite:///db.sqlite')**», «**Base.metadata.create_all(engine**)», « **Session = sessionmaker(bind=engine)**», «**session = Session()**». После запуска у нас появится файл с названием **db.sqlite**, который и является базой данных.

Добавим функцию добавления оценок в базу данных: «**def add_mark(date, pupil_name, subject, mark):**», где слова в скобках – список необходимых данных для ввода. Введём переменную **mark**, чтобы в дальнейшем не переписывать одну и ту же длинную команду каждый раз: «**mark = Mark(date=date, pupil_name=pupil_name, subject=subject, mark=mark)**». Далее вводим: «**session.add(mark)**», «**session.commit()**».

Теперь добавим функцию просмотра оценок. После завершения цикла пишем: «**def get_marks(pupil_name):**». Теперь выведем список оценок, добавленных ранее: «**query = session.query(Mark.id, Mark.pupil_name, Mark.subject, Mark.date, Mark.mark).filter_by(pupil_name=pupil_name)**», «**return query**», где **Mark.id** – ID оценки, знание которого потребуется нам при удалении оценки.

И напоследок добавим саму функцию удаления оценок. После завершения цикла вводим: «**def delete_mark(id):**». Сделаем проверку на наличие оценки с таким ID с помощью цикла **try**: «**try:**», «**mark = session.query(Mark).filter(Mark.id==id).first()**», проводим процедуру удаления оценки: «**session.delete(mark)**», «**session.commit()**», «**return True**». Введём действие при несоблюдении условия: «**except:**», «**return False**»

Теперь мы добавим возможность добавления, просмотра и удаления домашнего задания. Код будет практически идентичный, кроме некоторых деталей – для добавления домашнего задания не нужно будет вводить **pupil_name**, для просмотра домашнего задания нужно будет ввести название предмета вместо имени ученика соответственно.

Создание меню

Для удобства использования нашего чат-бота мы создадим меню. Для начала в папке **project** создаём файл с названием **markup.py**. В нём через текстовый редактор импортируем нужные нам команды: «**from aiogram.types import ReplyKeyboardMarkup, KeyboardButton**». Всего у нас будет 2 меню.

Для первого добавим две кнопки в качестве переменных – «**расписание**» и «**другое**»: «**btnMe = KeyboardButton('Расписание')**», «**btnMe1 = KeyboardButton('Другое')**». Теперь добавим эти кнопки вместо клавиатуры: «**main = ReplyKeyboardMarkup(resize_keyboard = True).add(btnMe, btnMe1)**», где **main** – название первого меню

Во втором меню добавим кнопки «**Оценки**», «**Дом. задание**» и «**Обратно**»: «**btn1 = KeyboardButton('Оценки')**», «**btn2 = KeyboardButton('Дом. задание')**», «**btnMain = KeyboardButton('Обратно')**». Добавим их на клавиатуру: «**other = ReplyKeyboardMarkup(resize_keyboard = True).add(btn1, btn2, btnMain)**», где **other** – название второго меню. Кнопки «**Другое**» и «**Обратно**» будут использоваться для перехода между меню.

Теперь при вводе команды **/start** вместо клавиатуры будет появляться меню с кодовыми словами. Это сделает общение с нашим ботом гораздо удобнее.

Последние детали в коде

Возвращаемся к редактированию файлу **bot_telegram.py**. Оформим переход из меню **main** в меню **other**. Завершая предыдущий цикл, пишем: «**elif message.text == 'Другое':**», «**await bot.send_message(message.from_user.id, 'Пожалуйста!', reply_markup = nav.other**)». Затем, завершая предыдущий цикл, оформим переход из меню **other** в меню **main**: «**elif message.text == 'Обратно':**», «**await bot.send_message(message.from_user.id, 'Пожалуйста!', reply_markup = nav.main)**». Теперь мы можем переходить в меню **other** с помощью команды «**Другое**» и в меню **other** с помощью команды «**Обратно**».

Теперь добавим возможность просматривать оценки при вводе имени ученика. Завершая предыдущий цикл, пишем: «**elif 'Оценки:' in message.text:**», «**pupil_name = message.text.replace(": ", ":").split(":")[1]**». Вводим переменную **marks**, взяв ранее написанные нами данные из файла **db.py**: «**marks =**

db.get_marks(pupil_name)». Теперь проверим, есть ли у такого ученика оценки: «**if marks.count() > 0:**». Далее выведем все оценки, которые на данный момент находятся в базе данных: «**for row in marks:**», «**await bot.send_message(message.from_user.id, list(row._asdict().values()))**». Завершая предыдущий цикл, введём ответ при несоблюдения условия: «**else:**», «**await bot.send_message(message.from_user.id, 'Ученик с таким именем не найден')**»

Добавим возможность добавлять оценки в базу данных. Завершая предыдущий цикл, пишем: «**elif 'Добавить оценку:' in message.text:**». Чтобы ученик не имел возможность добавлять оценки, введём проверку на присутствие ID пользователя внутри переменной **WRITE_USERS**: «**if message.from_user.id in WRITE_USERS:**». Далее вводим: «**mark_record = message.text.replace(": ", ":").split(":")[1].split(", ")**», и саму функцию добавления оценки: «**db.add_mark(mark_record[0], mark_record[1], mark_record[2], mark_record[3])**». Затем вводим действия бота при соблюдении всех условий и успешной обработке данных: «**await bot.send_message(message.from_user.id, 'Оценка добавлена')**». Теперь введём ответ бота при отсутствии ID пользователя внутри переменной **WRITE_USERS**: «**else:**», «**await bot.send_message(message.from_user.id, 'Нет прав на добавление оценок')**».

Также стоит добавить функцию удаления оценок, чтобы не засорять базу данных устаревшими данными. Завершаем два предыдущих цикла и пишем: «**elif 'Удалить оценку:' in message.text:**». Вводим проверку на присутствие ID пользователя внутри переменной **WRITE_USERS**: «**if message.from_user.id in WRITE_USERS:**». Далее вводим: «**mark_id = message.text.replace(": ", ":").split(":")[1]**» Теперь проверим, есть ли оценка с таким ID в базе данных: «**if db.delete_mark(int(mark_id)):**». Вводим действие при соблюдении всех условий: «**await bot.send_message(message.from_user.id, 'Оценка удалена')**». Введём действие при отсутствии ID оценки в базе данных: «**else:**», «**await bot.send_message(message.from_user.id, 'Ошибка при удалении оценки.**

Проверьте ID оценки.')». Завершаем цикл, вводим ответ при отсутствии ID пользователя внутри переменной **WRITE_USERS**: «**else:**», «**await bot.send_message(message.from_user.id, 'Нет прав на удаление оценок')**»

Скорее всего пользователю потребуется подсказка в использовании бота, а именно ключевые слова, формат записи команд. Для этого мы добавим вывод инструкции при запросах, связанных с оценками: «**elif 'ценк' in message.text:**», и содержание самой инструкции: «**await bot.send_message(message.from_user.id, 'Для просмотра оценок напишите:\nОценки: Имя Ученика\n \nДля добавления оценки напишите:\nДобавить оценку: Дата, Имя ученика, Предмет, Оценка\n \nДля удаления оценки напишите:\nУдалить оценку: ID оценки')**»

Теперь нам нужно добавить возможность добавления, просмотра, удаления домашнего задания, а также инструкцию при запросах, связанных с домашним заданием. Код будет практически идентичен вышеперечисленному, но с небольшими изменениями - для добавления домашнего задания не нужно будет вводить имя ученика, для просмотра домашнего задания нужно будет ввести название предмета вместо имени ученика соответственно, содержание инструкции изменится.

Завершая все текущие циклы, пишем команду, запускающую программный код: «**if __name__ == '__main__':**», «**executor.start_polling(dp, skip_updates = True)**». Когда наш бот офлайн, ему могут приходить команды от пользователей. Если после этого бот выйдет в онлайн, он ответит на все введённые во время его отключения запросы. Мы ввели **skip_updates = True**, чтобы не допустить это. Написание программного кода завершено, теперь нужно подключить нашего чат-бота к облачному серверу.

Подключение чат-бота к облачному серверу

Открываем папку **project**, в ней зажимаем Shift, кликаем правой кнопкой мыши и открываем командную строку или PowerShell. Для активации ключа от облачного сервера вводим: «**ssh -i pocketsiem-key.pem ubuntu@<ip_сервера>»** (важно, чтобы файл **pocketsiem-key.pem** находился в папке **project**). Далее копируем файл **bot_telegram.py** на сервер в папку **project**: «**scp -i pocketsiem-key.pem .\bot_telegram.py ubuntu@<ip_сервера>:project**». Затем вводим «**tmux a**» для активации сервера. Теперь можно закрыть консоль, и сервер продолжит функционировать. В случае необходимости для остановки сервера нужно нажать сочетание клавиш Ctrl+C.

Заключение

В современном мире виртуальные помощники взаимодействуют с пользователями в различных сферах. Их формат идеально подходит для поддержки юзеров. Общение осуществляется в рамках мессенджеров, что позволяет людям максимально быстро решать повседневные вопросы.

Результатом нашего проекта стал функциональный, работающий на облачном сервере чат-бот - помощник в сфере образования. Используя наш чат-бот, можно просматривать, удалять, добавлять оценки и домашнее задание. Это помогло нам решить проблему, связанную с отсутствием возможности у преподавателей установления быстрого контакта с учениками и передачи информации большому количеству лиц на значительное расстояние.

В процессе работы мы повысили свои навыки в программировании на языке Python, изучили и прошли все этапы создания чат-бота в мессенджере.

Нашего бота можно найти в Телеграм под названием **Analogoviybot**.

Библиографический список

- Сайт, с которого мы брали данные про актуальность чат-ботов - **https://www.unisender.com/ru/blog/kuhnya/chat-boty-vnedrenie/**

- Сайт, из которого мы брали данные про причину использования чат-ботов и алгоритм их работы – **https://powervirtualagents.microsoft.com/ru-ru/what-is-a-chatbot/**

- Сайт, с которого мы брали данные про историю развития и первую классификацию чат-ботов – **https://developers.sber.ru/help/salutebot/about-chatbots**

- Сайт, с которого мы брали данные про вторую классификацию чат-ботов - **https://www.mtt.ru/support/blog/why-business-needs-chatbots-how-to/**

- Видео, с помощью которого мы подготовились к написанию кода - **https://www.youtube.com/watch?v=TYs3-uyjC30&t=1099s**

- Видео, с помощью которого мы создали меню - **https://www.youtube.com/watch?v=yetfif4j_go&t=546s**

Приложение

Содержание всех используемых при создании чат-бот файлов вы можете найти здесь: **https://github.com/AllsOkay/Analogoviybot.git**

РЕЦЕНЗИЯ

на проектную (исследовательскую) работу обучающихся 7Б класса
ГБОУ Школы № 1363
Егоровой Екатерины, Стаурской Валерии

по теме: «Создание чат-ботов на языке Python»

Работа Егоровой Екатерины и Стаурской Валерии «Создание чат-ботов на языке Python» посвящена популяризации чат-ботов на различных сервисах в сети интернет. В частности, ученицы исследуют данное явление как неотъемлемую часть продуктивного взаимодействия пользователей интернета. Авторы проекта отмечает, что изучение данной темы приобретает возрастающую актуальность в условиях развития современного мира и образования, и настоящим исследованием подчеркивает важность эффективного взаимодействия учащихся и педагогов в цифровой образовательной среде.

Работа состоит из четырех частей.

Первая часть посвящена обоснованию актуальности данного исследования, поставлена цель и задачи. Кроме того, ученицы приводят в пример статистические данные использования чат-ботов и особенность их применения в сфере бизнеса.

Вторая часть посвящена подготовке к созданию бота и его регистрации. Выделены основные классификации и способы разработки.

Третья часть является ключевой в работе. Проведена подготовка к написанию кода, написан код на языке программирования Python, добавлена база данных с функциями бота (добавление, удаление оценок и домашнего задания). Создано меню для удобного взаимодействия человека и бота. Добавлены подсказки.

Четвертая часть - подключение бота к облачному серверу и апробация его работы.

Данный проект имеет практическую значимость, так как он может быть использован в повседневной жизни учащихся и учителей. Ученицами был изучен опыт использования современных информационно-коммуникационных технологий в образовательном процессе, а также был проведен анализ учебно-методической литературы по проблеме исследования.

Данная проектная работа имеет и личную значимость для ее авторов, так как она свидетельствует о способности самостоятельно ставить проблему и находить пути ее решения.

Образовательный потенциал работы заключается в том, что в процессе исследования ученицы продемонстрировали повышенную способность приобретать новые знания, умение работать с информацией (осуществлять сбор и анализировать), комбинировать ее, выдавая готовый IT-продукт.

Заявленная тема проекта соответствует его содержанию. Егорова Е. и Стаурская В. смогли убедительно обосновать актуальность выбранной темы, правильно поставить цели и задачи, которым соответствуют полученные выводы.

Тема проекта интересна и актуальна, вопрос для изучения выбран перспективный. Работа тщательно спланирована и последовательно реализована, своевременно пройдены все необходимые этапы обсуждения и представления. Тема работы раскрыта. Все мысли выражены ясно, логично, последовательно.

Работа выполнена на персональном компьютере с использованием современных средств разработки. Качество оформления соответствует действующим правилам и стандартам. Текст работы выполнен аккуратно и грамотно. Ошибки отсутствуют.

Вывод: в работе обоснована актуальность проблемы, четко обозначена цель и пути ее достижения. Работа выполнена на актуальную тему, в достаточном объеме и удовлетворяет требованиям, предъявляемым к проектным работам и рекомендуется к защите.

РЕЦЕНЗЕНТ Анохина Дарья Сергеевна *(подпись)* Анохина Д. С.
(ФИО)

«8» февраля 2023 г