УДК: 004.43
ББК: 32.973
ГРНТИ: 50.05.09

Виолина Андреевна Деменкова, *студент ФГБОУ ВО «Московский педагогический государственный университет», Москва, Россия, e-mail: violina.demenkova@mail.ru*
Violina Andreevna Demenkova, *student, Moscow Pedagogical State University, Moscow, Russia, e-mail: violina.demenkova@mail.ru*

Виктория Сергеевна Курбатова, *студент, ФГБОУ ВО «Московский педагогический государственный университет», Москва, Россия, e-mail: vskurbatova.2000@gmail.com*
Viktoriya Sergeevna Kurbatova, *student, Moscow Pedagogical State University, Moscow, Russia, e-mail: vskurbatova.2000@gmail.com*

Ольга Игоревна Панкратова, *магистрант ФГБОУ ВО «Московский педагогический государственный университет», Москва, Россия, e-mail: helga.bren@yandex.ru*
Olga Igorevna Pankratova, *master's student, Moscow Pedagogical State University, Moscow, Russia, e-mail: helga.bren@yandex.ru*

Алина Олеговна Самкова, *магистрант ФГБОУ ВО «Московский педагогический государственный университет», Москва, Россия, e-mail: alina.samkova1998@yandex.ru*
Alina Olegovna Samkova, *master's student, Moscow Pedagogical State University, Moscow, Russia, e-mail: alina.samkova1998@yandex.ru*

Любовь Рушановна Сираждинова, *специалист по учебно-методической работе Центра довузовской подготовки иностранных граждан по русскому языку Института филологии ФГБОУ ВО «Московский педагогический государственный университет», Москва, Россия, e-mail: slr150900@mail.ru*
Lubov Rushanovna Sirazhdinova, *specialist, Center for Pre-University Training of Foreign Citizens in the Russian language, Institute of Philology, Moscow Pedagogical State University, Moscow, Russia, e-mail: slr150900@mail.ru*

Мелисса Алексеевна Склифус, *студент ФГБОУ ВО «Московский педагогический государственный университет», Москва, Россия, e-mail: melissa.sklifus@mail.ru*
Melissa Alekseevna Sklifus, *student, Moscow Pedagogical State University, Moscow, Russia, e-mail: melissa.sklifus@mail.ru*

Разработка чат-бота MEMORIZER на площадке Telegram для запоминания иностранных слов

Аннотация. *В статье рассматривается процесс разработки чат-бота для запоминания иностранных слов на площадке Telegram с использованием языка программирования Python, а также обосновывается актуальность использования ботов в образовательных целях. В работе используется анализ существующих образовательных ресурсов, сравнение и моделирование собственного образовательного продукта.*

Ключевые слова: *мессенджер; Telegram; Python; чат-бот; разработка бота; переводчик.*

Development of a chatbot MEMORIZER on the Telegram platform for memorizing foreign words

Abstract. *The article discusses the process of developing a chatbot for memorizing foreign words on the Telegram platform using the Python programming language and substantiates the relevance of using bots for educational purposes. The work uses analysis of existing educational resources, comparison and modeling of own educational product.*

Keywords: *messenger; Telegram; Python; chatbot; bot development; translator.*

Введение

В настоящее время глобализация и развитие Интернета открывают для людей всего мира огромное количество различной информации. Большое число современных тенденций в науке, технике, информационных и производственных технологий подвергается стандартизации. Иностранные языки сейчас перестают быть объектом внимания и изучения учёных и становятся доступны для большего числа людей, в том числе не связанных с наукой, вместе с этим растёт и роль, которую иностранные языки оказывают на нашу жизнь. В современном мире знание иностранных языков становится необходимостью, связанной не только с организацией грамотной работы и пониманием информационных технологий, но и с повседневной жизнью. В XXI веке существенно вырос запрос на быстрое освоение и получение информации, поэтому широкое распространение получили приложения и программы для изучения иностранных языков и запоминания иностранных слов. Например, функционал чат-ботов позволяет пользователю работать с программой в формате диалога. Примером может быть чат-бот на базе мессенджера Telegram для изучения иностранных языков, в частности для запоминания слов с элементами игры в процессе обучения.

Анализ платформы для создания чат-бота («ВКонтакте» и Telegram)

В мире быстро развивающихся технологий и взаимосвязанного цифрового общества чат-боты становятся всё более популярным инструментом для автоматизации и улучшения взаимодействия с пользователями. Однако перед разработкой чат-бота возникает вопрос: где лучше создавать их – в популярной социальной сети «ВКонтакте» или в популярном мессенджере Telegram? Обе платформы предоставляют возможности для создания и развёртывания чат-ботов, но каждая имеет свои особенности и преимущества. Мы провели сравнительный анализ функционала и работы платформ «Вконтакте» и Telegram.

Согласно данным исследования, опубликованного в марте 2023 года, Telegram обошёл «ВКонтакте» по числу пользователей: у мессенджера Telegram 100 млн пользователей, у «ВКонтакте» – 81 млн. Суммарное количество пользователей соцсетей по миру составляет 3 млрд человек. По количеству аудитории Telegram входит в пятёрку самых популярных мессенджеров в мире. По состоянию на 2023 год Telegram популярнее «ВКонтакте» и часто используется как платформа для разработки и использования чат-ботов. Это связано с несколькими факторами:

1. *Фокус на мессенджере*: Telegram изначально был разработан как мессенджер, он активно продвигает функциональность чат-ботов и предоставляет разработчикам широкие возможности для создания ботов, включая API, документацию и инструменты для автоматизации интеракций.

2. *Распространённость и популярность*: Telegram является одним из самых популярных мессенджеров во многих странах, у него большая и активная пользовательская база. Это создаёт благоприятную среду для развития и использования чат-ботов.

3. *Конфиденциальность и безопасность*: Telegram активно продвигает конфиденциальность и безопасность обмена сообщениями. Это делает его привлекательным для пользователей, которые предпочитают взаимодействовать с чат-ботами в защищённой среде.

Несмотря на то, что «ВКонтакте» также предоставляет возможность создания и использования чат-ботов, сравнительно большая активность чат-ботов наблюдается в Telegram. Это может быть связано с фокусом Telegram на мессенджер и его распространённостью в качестве платформы для коммуникации и обмена сообщениями.

После тщательного изучения функциональности и возможностей обеих платформ нами было принято решение выбрать Telegram в качестве платформы для разработки чат-бота. Telegram предлагает широкий набор инструментов и API, позволяющих создавать высокофункциональные и интерактивные чат-боты. Нельзя не отметить и активно растущую аудиторию Telegram, которая обеспечивает потенциально большую базу пользователей для нашего чат-бота.

Анализ существующих чат-ботов в Telegram

В ходе анализа были рассмотрены аналогичные чат-боты на платформе Telegram: *@english4you_bot, @Eng4meBot, @eddy_en_bot, @LingvoBot, @slovarikbot*.

Выявлены недостатки уже существующих чат-ботов, которые были учтены при разработке нашего образовательного продукта. К основным недостаткам большинства чат-ботов, направленных на изучение иностранных языков, можно отнести:

- отсутствие возможности выбора тематики, т.е. слова изучаются рандомно;
- ограниченность бесплатного периода использования;
- в боте не предусмотрена функция принятия ответов, которые напечатаны пользователем, т.е. ручным вводом, вместо этого предлагаются варианты ответов (формат теста).

С учётом выявленных недостатков мы стремились создать чат-бот, который предлагает выбор тематики, обеспечивает длительный бесплатный период использования и активно вовлекает пользователя, позволяя ему самостоятельно печатать ответы. Мировым научным сообществом доказано, что в процессе письма, в том числе и набора текста, активность мозга увеличивается, за счёт чего улучшается способность к запоминанию.

Таким образом, цель нашего проекта – создать доступный, удобный и интерактивный чат-бот, который поможет пользователям улучшить свои языковые и орфографические навыки английского языка и запомнить новые лексические единицы более эффективным и увлекательным способом.

Архитектура

Чат-бот MEMORIZER создан на языке Python и основан на использовании шаблонов. Программа рассчитана на запоминание слов английского языка. Для того, чтобы чат-бот имел возможность распознавать сообщение пользователя, каждая кнопка inline-клавиатуры имеет так называемую обратную информацию (англ. *callback data*). После нажатия кнопки пользователем серверная часть сравнивает полученный ответ со значением или массивом значений.

Для создания чат-бота на платформе Telegram нами были выполнены следующие действия:

1. Обращение к специальному боту @BotFather в Telegram.
2. При помощи команды «/start» запуск @BotFather.
3. При помощи команды «/newbot» происходит регистрация нового чат-бота.
4. Вводится название бота для генерации @BotFather уникального токена авторизации.
5. @BotFather выдаёт токен, с которым будет осуществляться дальнейшая работа. Токен имеет вид: 6045322541:AAFOHW_L8uX4BcoWh1Ww47-4e7Xf47NWXHQ.

При помощи команды «/mybots» редактируется имя бота, добавляется информация о боте, прописываются подсказки команд, меняется аватарка бота и добавляется описание бота.

Информация о боте – это текст, который будет виден в профиле бота (рис. 1), а описание бота – это текст, который пользователи будут видеть в начале диалога с ботом.

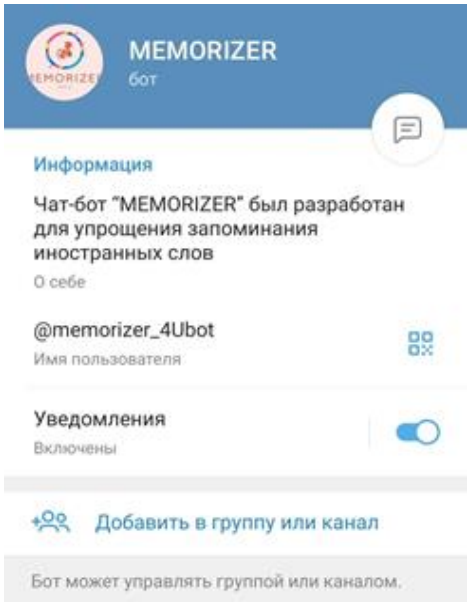

Рис. 1. Информация о боте MEMORIZER

Отметим, что так же у чат-бота присутствуют пострегистрационные настройки, описанные в таблице 1.

Таблица 1

Команда	**Описание**
/setname	Позволяет сменить имя чат-бота
/setdescription	Устанавливает описание бота, которое будет показано, при первом открытии чат бота
/setabouttext	Присваивает текст в поле «О чат-боте»
/setuserpic	Присваивает выбранную картинку
/setcommands	Позволяет создать список доступных команд
/deletbot	Удаляет указанный чат-бот

6. Подключается сервер, и осуществляется его настройка.
7. Скачивание Python и установка необходимых библиотек.
8. Написание кода, импорт библиотеки и подключение нашего бота.
9. При помощи языка программирования Python прописывается работа команд и кнопок.
10. Проверка работы нашего бота, исправление ошибок.

Ниже представлен код с документацией, отмеченной значком «#»:

```
# Импортируем необходимые библиотеки
import telebot
from telebot import types
import time

# Создаём экземпляр бота с помощью токена
bot = telebot.TeleBot('6045322541:AAFoHW_L8uX4BcoWh1Ww47-
4e7Xf47NWXHQ')

# Словарь с тематиками и словами
  words = {
    'literature': ['книга', 'роман', 'стих'],
    'technic': ['двигатель', 'машина', 'механизм'],
    'culture': ['искусство', 'музыка', 'театр'],
    'digital technologies': ['компьютер', 'интернет', 'смартфон']
}

# Словарь с переводами слов
translations = {
```

```
    'книга': 'book',
    'роман': 'novel',
    'стих': 'poem',
    'двигатель': 'engine',
    'машина': 'machine',
    'механизм': 'mechanism',
    'искусство': 'art',
    'музыка': 'music',
    'театр': 'theatre',
    'компьютер': 'computer',
    'интернет': 'internet',
    'смартфон': 'smartphone'
}

# Переменные для хранения текущей тематики и слова
current_topic = ''
current_word = ''


# Функция для обработки команды /start
@bot.message_handler(commands=['start'])
def start(message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for key in words.keys():
        markup.add(key)
    send_mess = f"<b>Привет, {message.from_user.first_name}
{message.from_user.last_name}</b>! \n Я бот для запоминания слов! " \
                f"Выберите тематику"
    bot.send_message(message.chat.id, send_mess, parse_mode='html',
reply_markup=markup)

# Функция для обработки выбора тематики и слова
@bot.message_handler(content_types=['text'])
def choose_topic(message):
    global current_topic
    global current_word
    if message.text in words.keys():
        current_topic = message.text
        current_word = words[current_topic][0]
        bot.send_message(message.chat.id, current_word.capitalize())
        time.sleep(2)
```

```
        bot.delete_message(message.chat.id, message.message_id + 1)
        bot.send_message(message.chat.id, "Let’s write")

    elif message.text.lower() == translations[current_word].lower():
        bot.send_message(message.chat.id, "Всё верно! Молодец!")
        bot.send_message(message.chat.id, "Продолжим?")
        bot.send_message(message.chat.id, "Напишите: Да или Нет")
        bot.register_next_step_handler(message, continue_learning)

    elif message.text.lower() == 'Да':
        current_word = words[current_topic][words[current_topic].index
(current_word) + 1]
        bot.send_message(message.chat.id, current_word.capitalize())
        time.sleep(2)
        bot.delete_message(message.chat.id, message.message_id + 1)
        bot.send_message(message.chat.id, "Let’s write")

    elif message.text.lower() == 'Нет':
        bot.send_message(message.chat.id, "Буду ждать тебя снова!")

    else:
        bot.send_message(message.chat.id, "Попробуй еще раз!")
        translation = translations[current_word]
        bot.send_message(message.chat.id,f"{current_word.capitalize()} - {translation}")
        time.sleep(2)
        bot.delete_message(message.chat.id, message.message_id + 2)
        bot.send_message(message.chat.id, "Let’s write")

# Функция для продолжения обучения
def continue_learning(message):
    if message.text.lower() in ['да', 'lf', 'Да', 'LF', 'ДА']:
        global current_word
        if words[current_topic].index(current_word) + 1 < len(words[current_topic]):
            current_word = words[current_topic][words[current_topic].index
(current_word) + 1]
            bot.send_message(message.chat.id, current_word.capitalize())
            time.sleep(2)
            bot.delete_message(message.chat.id, message.message_id + 1)
            bot.send_message(message.chat.id, "Let’s write")
        else:
            markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
```

```
            for key in words.keys():
                markup.add(key)
            bot.send_message(message.chat.id, "Вы прошли все слова по этой тематике. Выберите другую тематику:", reply_markup=markup)
        elif message.text.lower() == 'нет' or "ytn" or 'YTN' or "Нет" or "НЕТ":
            bot.send_message(message.chat.id, "Буду ждать тебя снова!")

bot.polling()
```

Основная логика программы:

- При запуске бота пользователю отправляется сообщение с кнопками выбора тематики.
- Пользователь выбирает тематику, и бот случайным образом выбирает слово из выбранной тематики и отправляет пользователю.
- Пользователь пишет перевод слова.
- Если перевод правильный, бот отправляет сообщение с поздравлением и предлагает продолжить либо завершить обучение.
- Если перевод неправильный, бот отправляет сообщение с правильным переводом и предлагает пользователю повторить попытку либо завершить обучение.
- В случае продолжения обучения бот выбирает следующее слово из той же тематики и повторяет процесс.
- Если слова в тематике закончились, бот предлагает пользователю выбрать другую тематику и повторяет процесс.

Тестирование

Для тестирования чат-бота было использовано как модульное, так и функциональное тестирование. Для проведения модульного тестирования использовался фреймворк unittest, встроенный в Python. Он предоставляет набор инструментов для создания и запуска тестовых сценариев и проверки правильности работы отдельных модулей, функций или классов программы. Данные тесты используют мок-объекты (mock objects) из модуля unittest.mock, чтобы имитировать некоторые компоненты и взаимодействие с внешними зависимостями, такими как Telegram API. Функциональное тестирование направлено на проверку выполнения функциональных требований программного обеспечения (табл. 2).

Таблица 2

Функциональное тестирование чат-бота

№	Элемент тестирования	Действие	Ожидаемый результат	Полученный результат
1	Запуск бота	Открыть бот и нажать кнопку «Запустить»	Отображение сообщения с приветствием	Отображение сообщения с приветствием

Окончание таблицы 2

№	Элемент тестирования	Действие	Ожидаемый результат	Полученный результат
2	Выбор тематики	Нажать на одну из предложенных тематик	Отображение слова из выбранной тематики	Отображение слова из выбранной тематики
3	Ввод правильного перевода	Ввести правильный перевод слова	Отображение сообщений «Всё верно! Молодец! Продолжим? Напишите: Да или Нет»	Отображение сообщений «Всё верно! Молодец! Продолжим? Напишите: Да или Нет»
4	Продолжение обучения	Написать «Да»	Отображение следующего слова для изучения	Отображение следующего слова для изучения
5	Прекращение обучения	Написать «Нет»	Отображение сообщения «Буду ждать тебя снова!»	Отображение сообщения «Буду ждать тебя снова!»
6	Ввод неправильного перевода	Ввести неправильный перевод слова	Отображение сообщения «Попробуй ещё раз!»	Отображение сообщения «Попробуй ещё раз!»
7	Показ правильного перевода	Ввести любой ответ, отличающийся от правильного	Отображение правильного перевода слова	Отображение правильного перевода слова
8	Продолжение обучения	Написать «Да»	Отображение следующего слова для изучения	Отображение следующего слова для изучения
9	Полное прохождение слов	Продолжить обучение до последнего слова в тематике	Отображение сообщения о завершении тематики	Отображение сообщения о завершении тематики
10	Выбор новой тематики	Выбрать другую тематику из предложенных	Отображение новой тематики и первого слова	Отображение новой тематики и первого слова

Функциональное и модульное тестирование чат-бота было пройдено успешно. В ходе тестирования не выявлено критических ошибок или неполадок в функциональности чат-бота.

Заключение

По результатам тестирования программы было сделано заключение, что образовательный продукт работает устойчиво и демонстрирует высокие показатели стабильности функционирования. Взаимодействие программы с пользователем происходит без сбоев, интерфейс визуально понятен и удобен в использовании. Чат бот MEMORIZER отвечает требованиям и запросам рынка образовательных услуг.

Таким образом, приложение способствует эффективному запоминанию иностранной лексики в интерактивном формате в условиях диджитал-среды. Тестирование выявило высокий уровень вовлечённости пользователей и удобство коммуникации в формате диалога. Данный проект имеет перспективы развития и масштабирования на рынке образовательных услуг.

Список литературы

1. Библиотека pyTelegramBotAPI. URL: https://github.com/eternnoir/pyTelegramBotAPI (дата обращения: 16.05.2023).
2. Библиотека AIOGram. URL: https://github.com/aiogram/aiogram (дата обращения: 01.05.2023).
3. Воройский Ф. С. Информатика. Новый систематизированный толковый словарь-справочник (Введение в современные информационные и телекоммуникационные технологии в терминах и фактах). М.: ФИЗМАТЛИТ, 2003. 760 с.
4. Зубкова А. Задачи, которые решают чат-боты. URL: https://www.cossa.ru/trends/190984/ (дата обращения: 02.06.2023).
5. Как создать своего бота в BotFather? URL: https://botcreators.ru/blog/kak-sozdat-svoego-bota-v-botfather/ (дата обращения: 14.05.2023).
6. Федоров Д. Ю. Программирование на языке высокого уровня Python: учебное пособие для вузов. М.: Юрайт, 2021. 210 с.

References

1. Biblioteka ruTelegramBotAPI. Available at: https://github.com/eternnoir/pyTelegramBotAPI (accessed: 16.05.2023).
2. Biblioteka AIOGram. Available at: https://github.com/aiogram/aiogram (accessed: 01.05.2023).
3. Voroiskiy F. S. *Informatika. Novyj sistematizirovannyj tolkovyj slovar'-spravochnik (Vvedenie v sovremennye informacionnye i telekommunikacionnye tehnologii v terminah i faktah)*. Moscow: FIZMATLIT, 2003. 760 p.
4. Zubkova A. Zadachi, kotorye reshajut chat-boty. Available at: https://www.cossa.ru/trends/190984/ (accessed: 02.06.2023).
5. Kak sozdat' svoego bota v BotFather? Available at: https://botcreators.ru/blog/kak-sozdat-svoego-bota-v-botfather/ (accessed: 14.05.2023).
6. Fedorov D. Yu. *Programmirovanie na jazyke vysokogo urovnja Python: uchebnoe posobie dlja vuzov*. Moscow: Jurajt, 2021. 210 p.