УДК 004.512

TELEGRAM-БОТЫ В ОБРАЗОВАНИИ: ВЫБОР ОПТИМАЛЬНОЙ ТЕХНОЛОГИИ РАЗРАБОТКИ

Камболин В.А., студент гр. ИТб-212, IV курс
Вольф Е.И., студент гр. ИТб-212, IV кур
Научный руководитель: Ванеев О.Н., к.т.н., доцент
Кузбасский государственный технический университет имени Т.Ф. Горбачева
г. Кемерово

Аннотация

В статье рассматриваются возможности использования Telegram-ботов, разработанных на языке программирования Python, для автоматизации образовательных процессов. Анализируются различные технологии разработки таких ботов, включая Requests, Telebot, Aiogram, Pyrogram и Python-Telegram-Bot. В качестве объекта исследования выбраны три технологии: Requests, Aiogram и Telebot, каждая из которых имеет свои особенности, преимущества и ограничения. Представлен сравнительный анализ этих технологий, приведены примеры кода, а также рассмотрены перспективы дальнейшего развития Telegram-ботов на Python в образовательной среде.

Современные образовательные учреждения активно внедряют цифровые технологии для повышения эффективности взаимодействия между преподавателями, студентами и административным персоналом. Одним из наиболее удобных инструментов автоматизации являются Telegram-боты, которые позволяют автоматизировать рутинные задачи, такие как регистрация студентов, управление расписанием, отправка уведомлений и прием заявок. Для реализации поставленных целей при создании информационной системы для автошколы, а именно сокращения временных затрат, связанных с подачей заявления на обучение в автошколе и получением данных о расписании занятий было принято решение разработать Telegram-боты.

Рассмотрим немного подробнее автоматизируемые процессы:

1. Бот для подачи заявлений

Разработанный бот выполняет сбор данных о пользователях, проверку введенных данных (номер паспорта, СНИЛС, дату рождения), а также сохранение данных в формате Excel. Диаграмма для данного процесса представлена на рисунке 1.

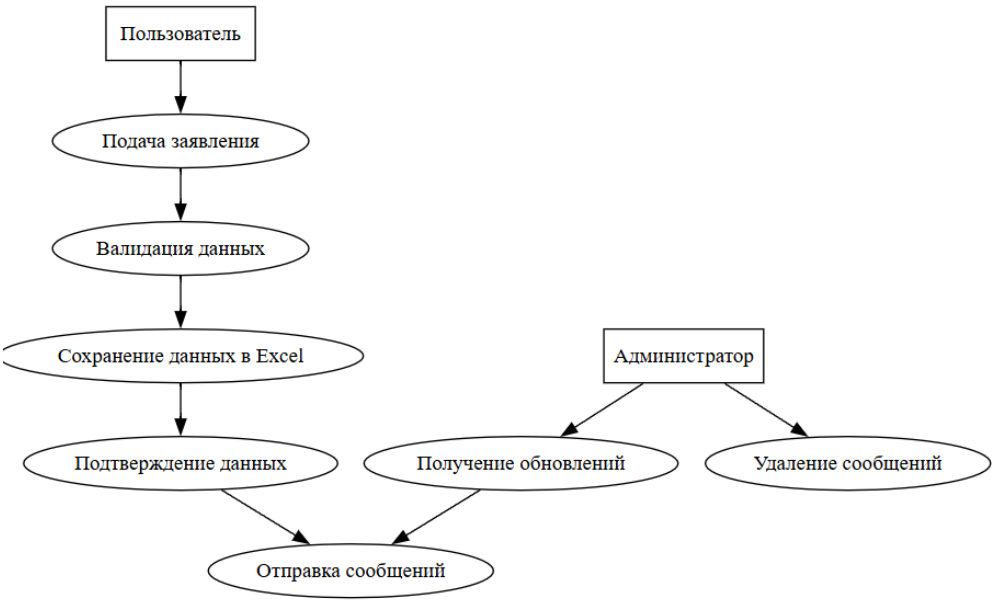

Рисунок 1

2. Бот для расписания (Aiogram)

Второй бот включает отображение расписания занятий, взаимодействие с учениками и преподавателями через клавиатуры, уведомление учеников о предстоящих занятиях и изменениях в расписании (рисунок 2).

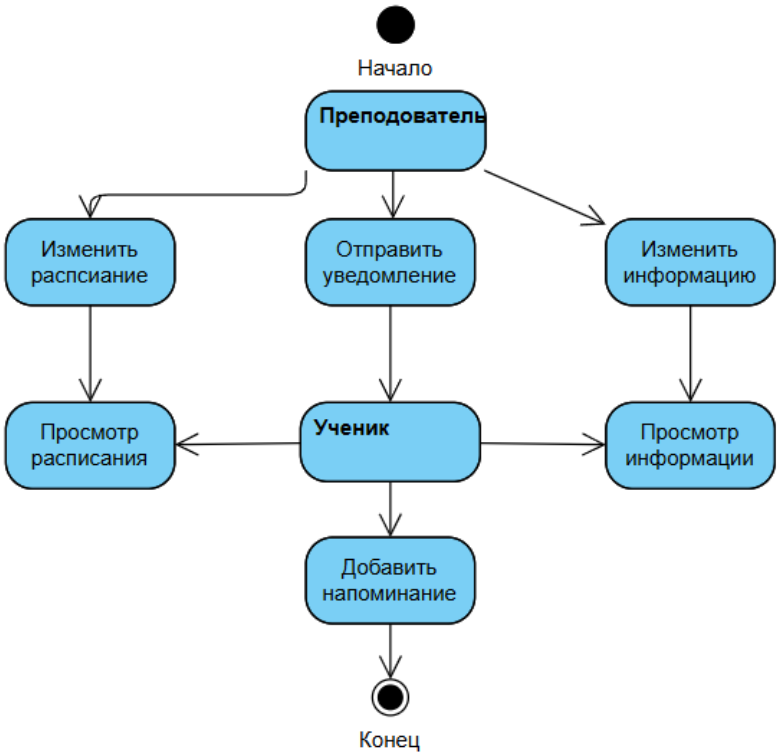

Рисунок 2

Telegram предоставляет удобный API для создания ботов, что делает возможным их реализацию на различных платформах и с использованием разных технологий. Выбор подходящей технологии разработки зависит от множества факторов, таких как производительность, удобство разработки, масштабируемость и поддержка асинхронного взаимодействия.

Рассмотрим несколько технологий разработки Telegram-ботов на Python:

1. Requests

Библиотека для выполнения HTTP-запросов. Простая, но синхронная, что ограничивает масштабируемость.

2. Telebot

Упрощает работу с Telegram API, но также в основном работает синхронно.

3. Aiogram

Асинхронный фреймворк, обеспечивающий высокую производительность.

4. Pyrogram

Позволяет работать не только с ботами, но и с обычными аккаунтами.

5. Python-telegram-bot

Мощный инструмент с поддержкой асинхронности и удобными обработчиками.

В рамках данной статьи для сравнения были выбраны три технологии: Requests, Aiogram и Telebot. Их выбор обусловлен тем, что они представляют различные подходы к разработке ботов и позволяют провести всесторонний анализ преимуществ и ограничений различных методов взаимодействия с Telegram API [1].

Рассмотренные альтернативные технологии включают Pyrogram и Python-telegram-bot, но они не были выбраны для исследования по следующим причинам:

- Pyrogram — мощная библиотека, позволяющая работать не только с ботами, но и с обычными Telegram-аккаунтами. Однако в данном исследовании акцент сделан именно на ботов, а не на взаимодействие с аккаунтами пользователей, поэтому использование Pyrogram не дает значительных преимуществ.
- Python-telegram-bot — библиотека с поддержкой асинхронности и удобными обработчиками. Хотя данная технология является перспективным инструментом, она менее популярна в русскоязычном сообществе разработчиков, а также обладает более сложным API по сравнению с Aiogram.

Выбранные технологии покрывают широкий спектр задач:

- Requests был включен в исследование, поскольку это самый базовый способ взаимодействия с Telegram API через HTTP-запросы. Его использование демонстрирует, как можно разрабатывать ботов без сторонних специализированных библиотек, но при этом страдать от недостатка гибкости и масштабируемости.
- Aiogram был выбран как асинхронная альтернатива, обеспечивающая высокую производительность и возможность обработки большого количества

запросов одновременно. Он является одним из самых популярных инструментов для создания сложных Telegram-ботов.

- Telebot был добавлен в исследование как промежуточный вариант, который предоставляет удобный интерфейс для работы с Telegram API, но остается синхронным, что может ограничивать его использование в масштабных решениях.

Таким образом, выбор Requests, Aiogram и Telebot позволяет продемонстрировать три разных подхода к разработке Telegram-ботов: от простого и наименее эффективного (Requests), через удобный, но ограниченный (Telebot), к оптимальному для высоконагруженных сервисов (Aiogram).

1. Requests является мощным инструментом для работы с HTTP-запросами, однако его синхронная природа ограничивает его возможности в высоконагруженных системах. При использовании Requests бот должен дожидаться ответа сервера перед выполнением следующего запроса, что замедляет его работу. Последовательность обработки запросы приведена на рисунке 3

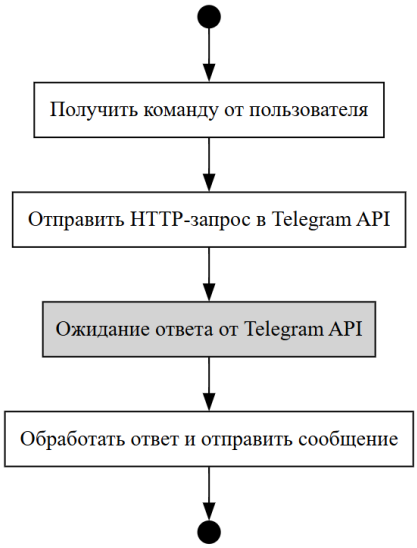

Рисунок 3

Пример программного кода для реализации:

```
import requests
TOKEN = " BOT_TOKEN"
URL = f"https://api.telegram.org/bot{TOKEN}/sendMessage"
def send_message(chat_id, text):
    data = {"chat_id": chat_id, "text": text}
    requests.post(URL, data=data)
```

2. Aiogram [2] позволяет боту обрабатывать несколько запросов одновременно, не блокируя выполнение других задач (рисунок 4). Это особенно важно

при увеличении количества пользователей, когда требуется высокая производительность.

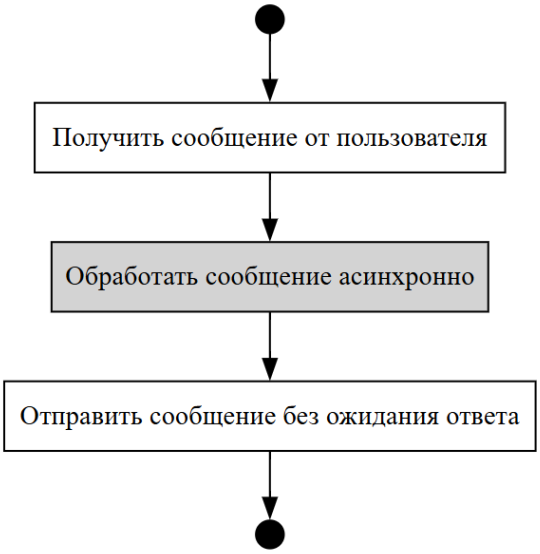

Рисунок 4

Пример программного кода для реализации:

```
from aiogram import Bot, Dispatcher, types
import asyncio
bot = Bot(token=" BOT_TOKEN")
dp = Dispatcher(bot)
@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    await message.reply("Привет! Я помогу с расписанием.")
async def main():
    await dp.start_polling()
asyncio.run(main())
```

3. Telebot — удобное решение для простых и средних по сложности ботов

Telebot предоставляет удобный интерфейс для работы с Telegram API, обеспечивая обработку команд, сообщений и различных типов контента. Хотя по умолчанию он работает синхронно, его можно использовать в асинхронном режиме при интеграции с многопоточным или асинхронным кодом (рисунок 5). Это делает его хорошим выбором для небольших и средних проектов, где высокая производительность не является критически важной.

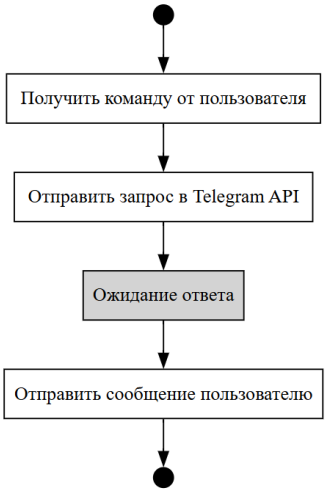

Рисунок 5

Пример программного кода для реализации:

```
import telebot
bot = telebot.TeleBot("BOT_TOKEN"
@bot.message_handler(commands=['start'])
def start(message):
    bot.send_message(message.chat.id, "Привет! Я бот на Telebot.")
bot.polling()
```

Сравнительный анализ рассмотренных технологий приведён в таблице 1.

Таблица 1

Критерии для сравнения	Requests	Aiogram	Telebot
Тип обработки данных	Синхронная	Асинхронная	Синхронная
Хранение информации	Excel	В памяти (runtime)	В памяти (runtime)
Основные функции	Валидация данных, сохранение	Отправка уведомлений, управление расписанием	Обработка команд, взаимодействие с пользователями
Производительность	Средняя (может замедляться при высокой нагрузке)	Высокая (оптимизирована для многозадачности)	Средняя (может замедляться при высокой нагрузке)
Гибкость и масштабируемость	Ограниченная	Высокая	Средняя
Поддержка состояний пользователей	Нет	Да	Частично (реализуется вручную)
Возможность работы с callback-запросами	Ограниченная	Полная	Средняя (поддерживает, но не так гибко, как Aiogram)

Перспективы дальнейшего развития Telegram-ботов для автоматизируемых процессов:

- Интеграция базы данных для хранения информации о пользователях и их запросах.
- Улучшение безопасности хранения персональных данных.
- Расширение функционала, например, добавление голосовых команд для взаимодействия с ботом.
- Использование вебхуков вместо polling для повышения эффективности бота.
- Интеграция с другими сервисами, например, Google Calendar для автоматического планирования занятий.

В заключение следует отметить, что Telegram-боты являются мощным инструментом для автоматизации образовательных процессов, позволяя упрощать взаимодействие между студентами, преподавателями и администрацией. В ходе исследования были рассмотрены три технологии разработки ботов: Requests, Aiogram и Telebot.

Выбор технологии зависит от требований к производительности, масштабируемости и удобства разработки. Для бота подачи заявлений была выбрана Requests, так как этот бот выполняет простые синхронные запросы и не требует обработки большого количества одновременных запросов. Для бота расписания был использован Aiogram, поскольку его асинхронный подход обеспечивает высокую производительность и позволяет эффективно работать с большим количеством пользователей.

Технология Telebot не была выбрана в качестве основной, поскольку она, хотя и удобна в использовании, не обладает преимуществами асинхронной обработки, необходимыми для масштабируемых решений. В случае увеличения нагрузки Telebot может столкнуться с проблемами производительности, что делает его менее предпочтительным для сложных сервисов, требующих быстрой и эффективной обработки запросов. Однако Telebot остается хорошим вариантом для небольших ботов с ограниченным числом пользователей.

Список литературы:

1. Telegram API. [Электронный ресурс] – Режим доступа: https://core.telegram.org/bots/api, свободный (дата обращения: 20.03.2025).

2. Aiogram Documentation. [Электронный ресурс] – Режим доступа: https://docs.aiogram.dev, свободный (дата обращения: 20.03.2025).

3. PyTelegramBotAPI. [Электронный ресурс] – Режим доступа: https://pypi.org/project/pyTelegramBotAPI, свободный (дата обращения: 20.03.2025).