УДК 004.04

Н.И. Гурин, доц., А. Часнойть, магистрант
(БГТУ, г. Минск)

ГЕНЕРАЦИЯ ПРОСТЫХ TELEGRAM-БОТОВ НА ОСНОВЕ ПОЛЬЗОВАТЕЛЬСКИХ ЗАПРОСОВ

В настоящее время мессенджер Telegram широко используется не только как средство общения, но и как платформа для создания различных сервисов на основе ботов – автоматизированных программ, которые взаимодействуют с пользователями через чат и выполняют определённые действия по заранее заданной логике. Telegram-боты применяются для получения информации о погоде, напоминаний, рассылок сообщений, автоматизации рутинных задач и взаимодействия с пользователями. Однако разработка даже простого Telegram-бота требует понимания базовых принципов функционирования API Telegram и определенной настройки его окружения, что может быть затруднительно для пользователей без соответствующей технической подготовки. Используя Telegram-боты, пользователь, не обладающий навыками программирования, может получить готовый код простого Telegram-бота, который при необходимости может быть дополнительно настроен или расширен.

В связи с этим актуальной является задача автоматизации процесса создания простых Telegram-ботов. Особенно перспективным направлением является разработка систем, которые позволяют генерировать боты на основе текстовых запросов пользователя, описывающих требуемую функциональность. Такой подход снижает порог входа, ускоряет разработку и позволяет создавать прототипы сервисов без глубокого погружения в программирование.

В данной работе представлена версия программного прототипа системы, предназначенной для генерации простых Telegram-ботов на основе текстовых запросов пользователя. В рамках данной версии рассматриваются боты с базовой функциональностью, такие как бот для получения информации о погоде, бот-напоминалка и бот для рассылки сообщений.

В работе решены следующие задачи:

– разработан механизм определения типа бота на основе пользовательского текстового запроса;

– реализована система генерации Telegram-ботов на основе заранее подготовленных шаблонов;

– протестирована работа системы на примерах пользовательских запросов.

Для реализации поставленной задачи был разработан программный прототип Telegram-бота-конструктора, который принимает текстовые сообщения от пользователя и на их основе генерирует другие Telegram-боты. В качестве языка программирования был выбран Python, а для взаимодействия с Telegram API использована библиотека aiogram, позволяющая быстро создавать асинхронные боты.

Архитектура системы включает следующие основные компоненты:

– Telegram-бот-генератор, принимающий сообщения от пользователя.

– Модуль анализа запроса, определяющий тип требуемого бота.

– Модуль генерации, создающий новый бот на основе шаблона.

– Набор шаблонов простых Telegram-ботов.

Схема взаимодействия компонентов системы представлена следующей последовательностью:

1. Пользователь отправляет текстовый запрос основному генератор-боту в Telegram.

2. Генератор-бот принимает сообщение и передаёт его в модуль анализа пользовательского запроса.

3. Модуль анализа определяет тип создаваемого бота на основе ключевых слов.

4. Генератор-бот запрашивает у пользователя токен Telegram-бота.

5. Полученный токен передаётся в модуль генерации ботов.

6. Модуль генерации создаёт новый экземпляр Telegram-бота на основе выбранного шаблона.

7. Модуль управления ботами запускает созданного бота и контролирует его жизненный цикл.

8. Созданный бот начинает взаимодействовать с конечными пользователями в Telegram.

Определение типа бота реализовано с использованием rule-based подхода. Текст пользовательского запроса приводится к нижнему регистру, после чего проверяется наличие ключевых слов, характерных для определённого типа бота. Например, наличие слов «погода» или «погодный» соответствует боту погоды, слова «напоминание» или «напомни» – боту-напоминалке, а слова «рассылка» или «сообщение всем» – боту рассылки.

После определения типа бота система выполняет генерацию нового Telegram-бота. Генерация осуществляется путём копирования соответствующего шаблона из каталога шаблонов в отдельную директорию сгенерированных ботов. Каждый шаблон представляет собой

минимально работоспособный Telegram-бот с реализованной основной функциональностью. Например, бот погоды получает данные о погоде с использованием внешнего API, бот-напоминалка реализует отложенную отправку сообщений, а бот рассылки отправляет сообщения всем пользователям, взаимодействовавшим с ботом.

В ходе выполнения данной работы был разработан программный прототип системы генерации простых Telegram-ботов на основе текстовых запросов пользователя. Реализованная система позволяет автоматически создавать Telegram-боты с базовой функциональностью, используя шаблонный подход и простую классификацию пользовательских запросов.

Реализация Telegram-бот-конструктора, принимающего сообщения от пользователя, основана на работе асинхронного обработчика сообщений `handler`, зарегистрированного в диспетчере `Dispatcher` основного генератор-бота. Данный обработчик отвечает за приём пользовательского запроса, управление состоянием диалога и запуск процесса создания нового Telegram-бота.

При получении сообщения система проверяет, находится ли пользователь в процессе создания бота. Если процесс ещё не начат, текст сообщения передаётся в модуль анализа запроса для определения типа требуемого бота.

```
@dp.message()
async def handler(msg: Message):
   uid = msg.from_user.id
   if uid not in user_state:
      bot_type = detect_type(msg.text)
      if not bot_type:
   await msg.answer("Не удалось определить тип бота")
return
      user_state[uid] = {"type": bot_type}
      await msg.answer("Пришлите токен от BotFather") return
```

Листинг 1 – Модуль обработки запроса

Таким образом, генератор-бот реализует интерактивный сценарий взаимодействия с пользователем и обеспечивает пошаговое формирование параметров создаваемого бота.

Модуль анализа запроса, определяющего тип требуемого Telegram-бота, реализован в виде функции `detect_type`. Данная функция выполняет анализ текстового запроса пользователя и сопоставляет его с поддерживаемыми типами ботов на основе ключевых слов.

В текущей версии системы используется правило-ориентированный подход, что позволяет обеспечить простоту и предсказуемость работы конструктора на этапе MVP.

```
def detect_type(text: str):
    text = text.lower()
    if "рассыл" in text:
        return "broadcast"
    if "напомин" in text:
        return "reminder"
    return None
```

Листинг 2 – Модуль анализа запроса

Результатом работы функции является строковый идентификатор типа бота, который далее используется модулем генерации для выбора соответствующего шаблона.

Модуль генерации, создающий новый Telegram-бот на основе выбранного шаблона, использует метод `start_bot` класса `BotEngine`. Данный метод выполняет инициализацию нового экземпляра Telegram-бота, регистрацию обработчиков сообщений и запуск процесса опроса серверов Telegram.

В зависимости от выбранного пользователем типа бота, в метод передаётся соответствующая функция настройки шаблона.

```
class BotEngine:
    async def start_bot(self, token: str, handler_factory):
        bot = Bot(token=token)
        dp = Dispatcher()
        handler_factory(dp)
        asyncio.create_task(dp.start_polling(bot))
```

Листинг 3 – Модуль генерации бота

Использование шаблонов позволяет реализовать архитектуру, при которой один программный процесс обслуживает несколько Telegram-ботов, каждый из которых обладает собственной логикой обработки сообщений. Такой подход обеспечивает масштабируемость решения и упрощает расширение функциональности конструктора за счёт добавления новых шаблонов.

В результате выполнения работы были достигнуты следующие результаты:

– реализован механизм, принимающий текстовые запросы пользователя;

– разработан механизм определения типа бота на основе ключевых слов;

– создан набор шаблонов простых Telegram-ботов;

– продемонстрирована возможность автоматизации процесса создания Telegram-ботов.

Следует отметить, что представленная версия системы является прототипом и имеет ряд ограничений. В частности, анализ пользовательских запросов реализован в упрощённом виде и не учитывает сложные формулировки. Кроме того, система не выполняет автоматическое развертывание и запуск сгенерированных ботов.

Тем не менее, полученные результаты подтверждают целесообразность выбранного подхода и демонстрируют возможность дальнейшего развития системы. В перспективе возможно использование методов машинного обучения и нейросетей для более точного анализа запросов, расширение набора поддерживаемых ботов, а также автоматизация процесса развертывания и запуска сгенерированных Telegram-ботов.

ЛИТЕРАТУРА

1 Telegram Bot API. Официальная документация. – URL: https://core.telegram.org/bots/api (дата обращения: 21.01.2026).

2 Aiogram 3.x Documentation. Асинхронный фреймворк для разработки Telegram-ботов на языке Python. – URL: https://docs.aiogram.dev/en/dev-3.x/ (дата обращения: 20.01.2026).

УДК 004.42

Я.А. Сидорик;
Н.Н. Пустовалова, доц., канд. техн. наук
(БГТУ, г. Минск)

КОНСТРУКТОР БАЗ ДАННЫХ ДЛЯ POSTGRESQL

В настоящее время часто встречающейся проблемой является необходимость организовать информацию в структурированном виде. Это может быть актуально для менеджеров проектов, аналитиков, специалистов по качеству, начинающих разработчиков. Для решения задач учета, хранения результатов или анализа информации им нужна своя небольшая база данных. Но для создания базы требуется знание языка SQL и особенностей конкретной системы управления базами данных, например, PostgreSQL [1]. Специалист, который просто хочет быстро организовать свои данные, вынужден тратить недели на изучение основ. Это неэффективно и тормозит работу.

Конструктор баз данных способен устранить этот барьер с помощью интуитивно понятного графического интерфейса. Его задача –