15.1.2025

Нездулькина Анна Владимировна

Urban university

Реализация асинхронного программирования на примере телеграмм-бота, рефакторинг кода

**Содержание:**

[**Введение** 2](#_Toc187779857)

[**Обзор проекта** 3](#_Toc187779858)

[**Структура проекта** 4](#_Toc187779859)

[***Старт*** 4](#_Toc187779860)

[***Пользовательское меню*** 4](#_Toc187779861)

[***Регистрация пользователя*** 9](#_Toc187779862)

[***Админ-панель*** 11](#_Toc187779863)

[***Сообщения по расписанию*** 11](#_Toc187779864)

[**Заключение** 13](#_Toc187779865)

[**Приложение 1. Файловая структура проекта** 14](#_Toc187779866)

[**Приложение 2. Список необходимых библиотек** 15](#_Toc187779867)

# **Введение**

Главной целью проекта является закрепление навыков асинхронного программирования на примере пользовательского телеграмм-бота и применение на практике полученных знаний об организации структуры кода. Проект предполагает создание бота, предназначенного для повседневного использования автором данного проекта в личных целях. В процессе создания потребуется обращение к уже знакомым и ранее неизведанным библиотекам и инструментам, что в целом обещает сделать созидательный процесс довольно занятным.

# **Обзор проекта**

Основной функционал бота заключается в возможности формировать список мероприятий и предоставлять к нему доступ группе пользователей.

Мероприятия подразделяются на две категории:

1. список покупок;
2. список мероприятий.

В основном для реализации проекта использовались библиотека aiogram версии 3.16.0 и встроенная библиотека sqlite3 для обращения к базам данных, полный перечень используемых библиотек приведен в Приложении № 2 настоящей дипломной работы.

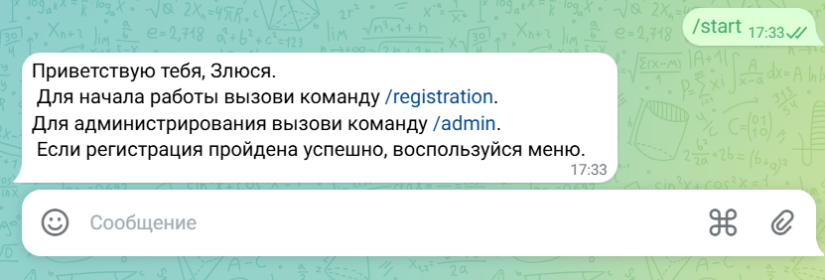
Бот по запросу представляет доступ зарегистрированным пользователям к имеющимся мероприятиям и по заданному расписанию осуществляет рассылку напоминаний, при наличии таковых в базе данных.

# **Структура проекта**

## ***Старт***

Токен бота (API) вынесен в переменную виртуального окружения и хранится в файле .env, поэтому перед запуском бота необходимо убедиться, что в корневой директории присутствует указанный файл с идентификатором, файл конфигурации config.py реализует доступ к токену.

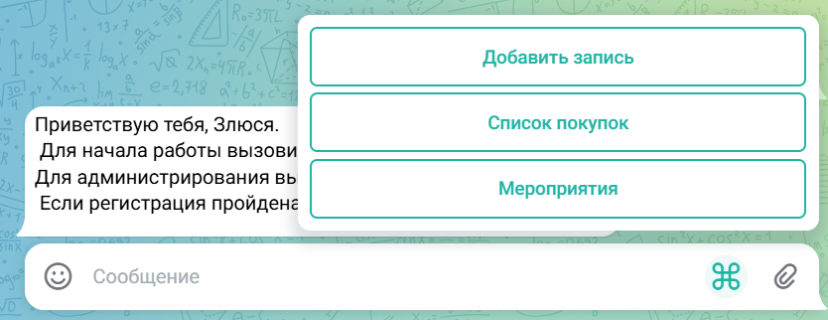
Начало работы с ботом осуществляется вызовом команды /start, после чего пользователю предлагается пройти процедуру регистрации, без которой основной функционал бота недоступен, администратору – админ-панель, к ней доступ осуществляется непосредственно по ID администратора.



*Рисунок 1 Старт работы с ботом*

## ***Пользовательское меню***

Основное меню представляет собой три кнопки, реализующие весь пользовательский потенциал бота.



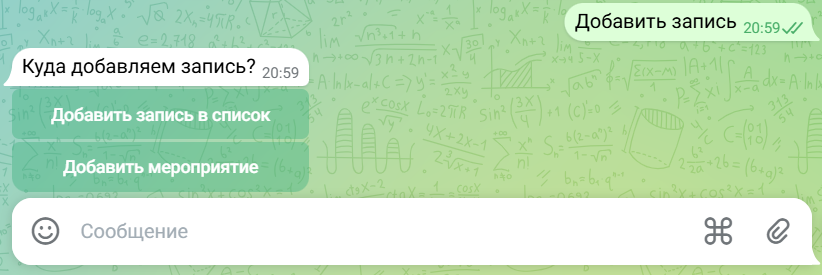
*Рисунок 2 Основное меню*

Основной функционал заключается в возможности отправлять боту информацию, которая хранится в базах данных и затем предоставляется пользователю по запросу или посредством отправки напоминаний по расписанию, реализованных в функциях отправки сообщений в файле time\_messages.py на базе библиотек apscheduler и datetime, старт которых выполняется в функции запуска бота main в исполняемом файле bot.py.

Визуально основная иерархия проекта представлена на схеме ниже:

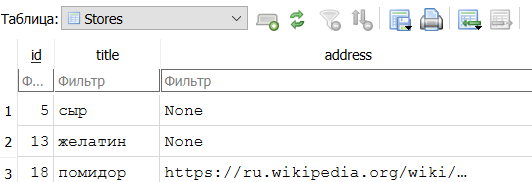
*Рисунок 3 Структура бота*

Взаимодействие с кнопкой главного меню «Добавить запись» вызывает инлайн-клавиатуру, предлагающую выбрать тип вносимой записи.



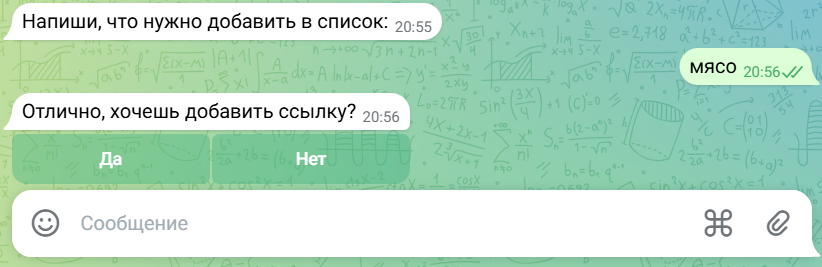
*Рисунок 4 Добавить запись*

Нажатие на кнопку инлайн-клавиатуры «Добавить запись в список» вызывает цепочку хендлеров, посредством FSM осуществляющих сохранение сообщений пользователя в базу данных Stores.



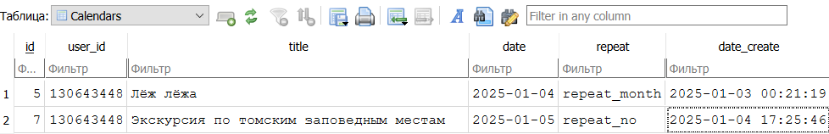
*Рисунок 5 База данных Stores*

На этапе внесения данных пользователю предлагается явный выбор, если есть необходимость, к записи можно добавить ссылку на конкретный продукт.



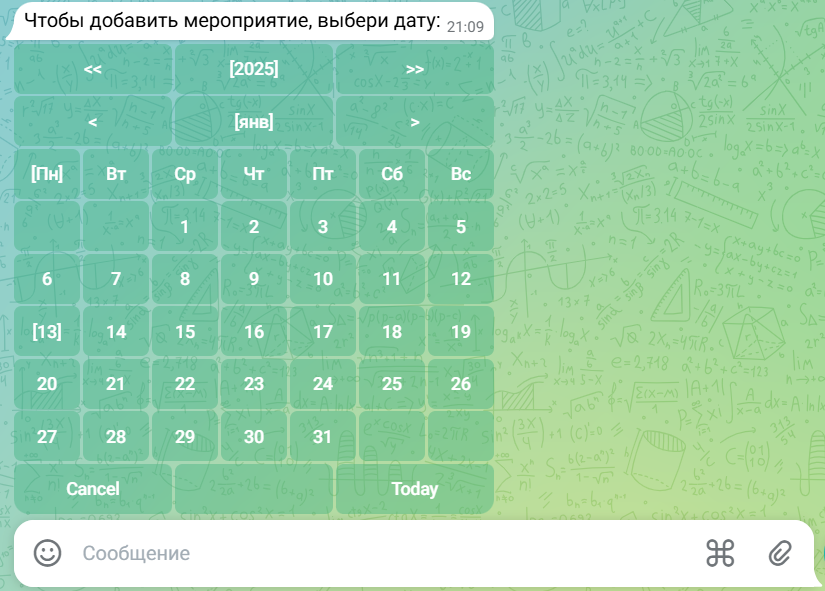
*Рисунок 6 Добавление ссылки в Stores*

Аналогично нажатие на кнопку инлайн-клавиатуры «Добавить мероприятие» вызывает цепочку хендлеров, посредством FSM осуществляющих сохранение сообщений пользователя в базу данных Calendars. Здесь user\_id необходим для осуществления рассылки напоминаний по расписанию по ID пользователя в телеграмм, дата создания не нужна, автор диплома была сражена её красотой в моменте инициализации базы данных.



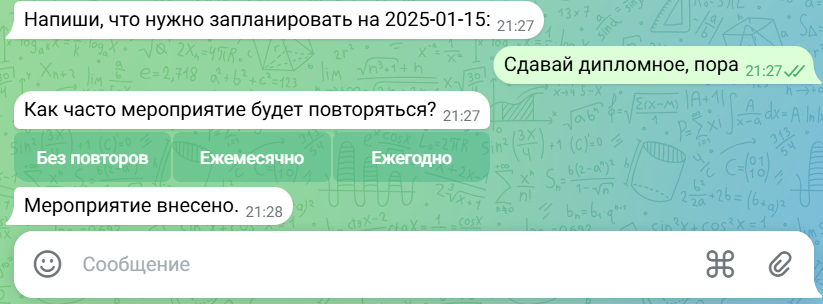
*Рисунок 7 База данных Calendars*

Для исключения ошибок при вводе дату пользователю предлагается выбирать в календаре, реализованном на базе инлайн-клавиатуры.



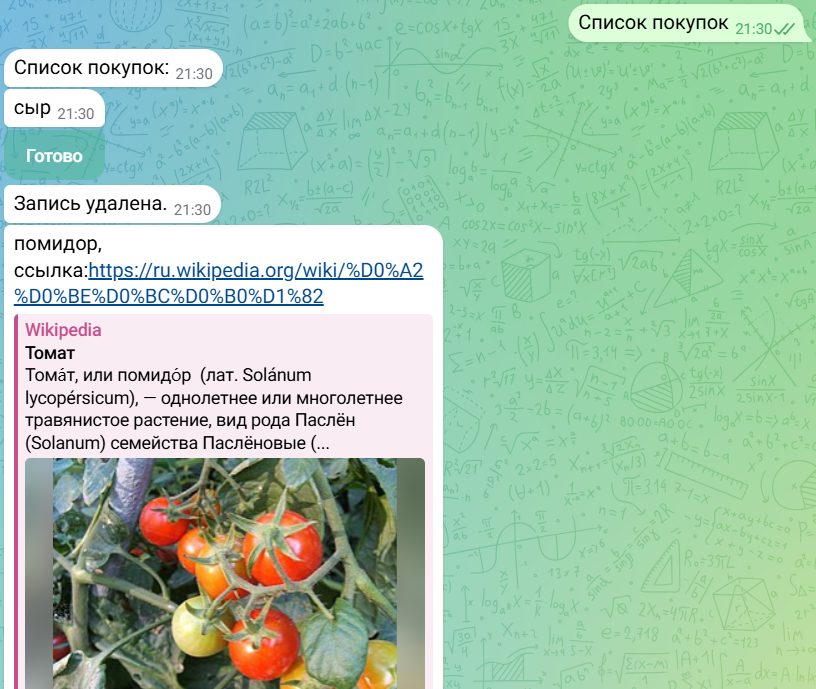
*Рисунок 8 Выбор даты*

После выбора даты и сообщения о содержании мероприятия пользователю предлагается выбрать его повторяемость (repeat в базе данных), чтобы в дальнейшем реализовать возможность не вносить повторяющиеся мероприятия с нуля, а переносить имеющееся на следующий плановый период.



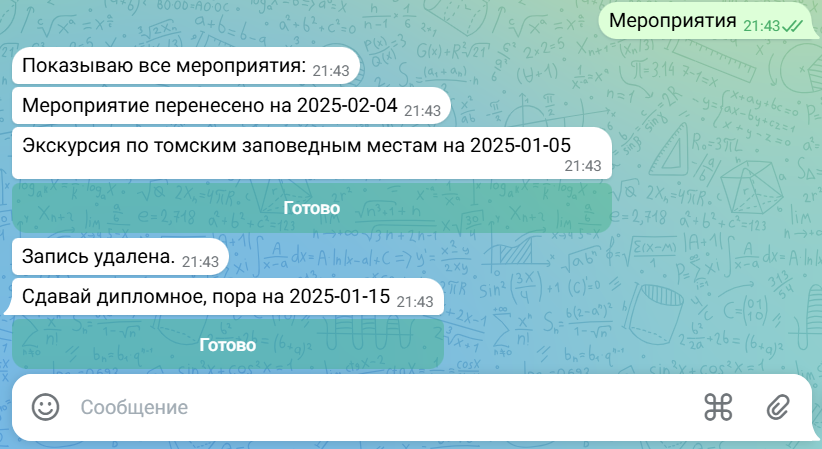
*Рисунок 9 Выбор периодичности мероприятия*

Взаимодействие с кнопкой главного меню «Список покупок» выводит в диалоговое окно бота содержимое базы данных Stores, здесь кнопка «Готово» удаляет запись из таблицы БД, пользователю даётся обратная связь о том, что запись удалена.



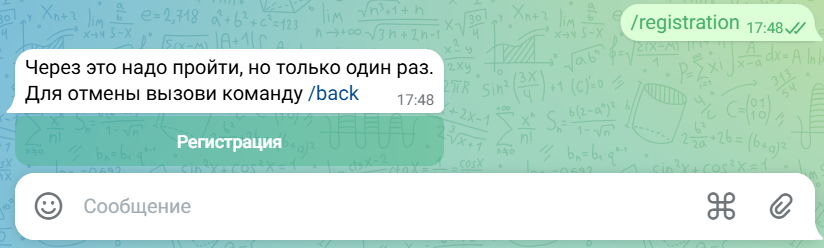
*Рисунок 10 Список покупок*

Взаимодействие с кнопкой главного меню «Мероприятия» выводит в диалоговое окно бота содержимое базы данных Calendars, здесь эффект от нажатия кнопки «Готово» зависит от повторяемости мероприятия, после нажатия пользователю так же даётся обратная связь о произошедших изменениях в базе данных.



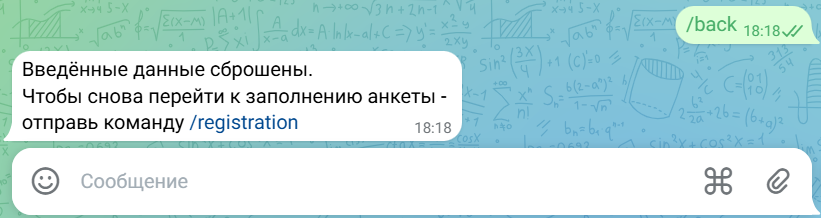
*Рисунок 11 Список мероприятий*

## ***Регистрация пользователя***



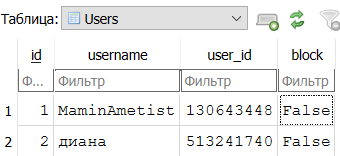
*Рисунок 12 Регистрация пользователя*

Регистрация пользователя осуществляется посредством вызова команды /registration. Непосредственно сама регистрация реализована на основе инлайн-клавиатуры и сохранении данных о пользователе в базу данных Users посредством применения функционала машины состояний библиотеки aiogram. Для корректного выхода из состояний предусмотрена команда /back, вызов которой на любом этапе прерывает процесс регистрации, выполняет сброс всех сохранённых данных и предлагает пользователю возобновить процесс регистрации.



*Рисунок 13 Сброс состояний*

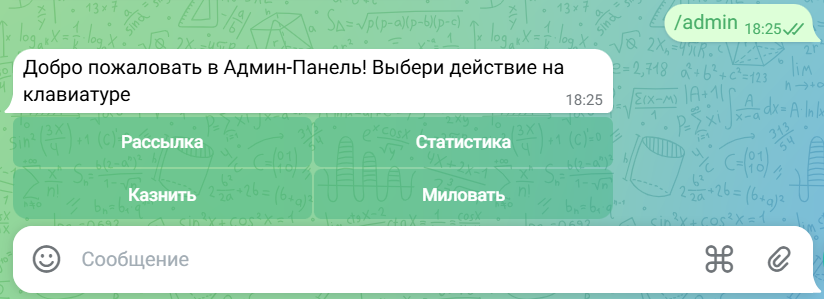
Авторизация участников не требуется, доступ к функционалу бота осуществляется по IP личного аккаунта телеграмм и регулируется администратором. При этом в базе данных участников группы хранятся следующие сведения: имя пользователя, ИД пользователя и его статус (boolean).



*Рисунок 14 База данных Users*

## ***Админ-панель***

Админ-панель также реализована на базе инлайн-клавиатуры и машины состояний, доступ к клавиатуре органичен фильтром, реагирующим на ID администратора.

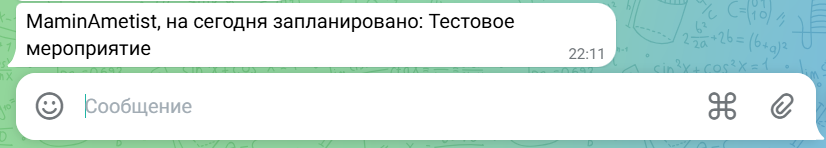


*Рисунок 15 Админ-панель*

Функционал администратора представляет собой возможность осуществлять рассылку сообщений пользователям телеграмм-бота, выводить в чат бота сведения о количестве пользователей, изменять статус пользователя по его ID в телеграмм.

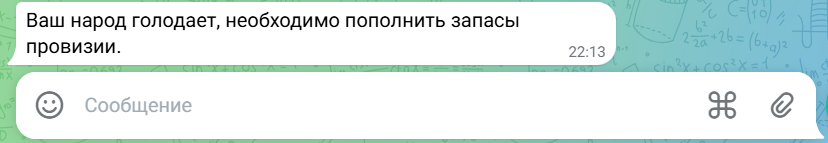
## ***Сообщения по расписанию***

В боте реализованы сообщения по расписанию для напоминания о мероприятиях, при их наличии рассылка осуществляется один раз в день по ID пользователя в телеграмм.



*Рисунок 16 Сообщение о наличии мероприятия*

Так же реализована отправка администратору напоминания о непустом списке покупок.



*Рисунок 17 Сообщение о наличии записей в списке покупок*

# **Заключение**

В процессе выполнения дипломного проекта была достигнута основная цель – создание телеграмм-бота с функционалом календаря и списков напоминаний, это позволило закрепить навыки асинхронного программирования и углубить понимание структурирования кода.

Бот разработан с учетом повседневных задач автора и позволяет создавать списки покупок и мероприятий, управлять напоминаниями с использованием библиотеки apscheduler и обеспечивать доступ к данным через удобное меню и интуитивно понятный интерфейс.

В рамках проекта применены современные инструменты Python, включая библиотеки aiogram и sqlite3, что позволило создать масштабируемое и стабильное приложение. Тестирование подтвердило работоспособность всех заявленных функций.

Разработанный бот демонстрирует успешное сочетание теоретических знаний и практических навыков, полученных в процессе обучения. Он готов к использованию и обладает потенциалом для дальнейшего развития и интеграции нового функционала.

# **Приложение 1. Файловая структура проекта**

.env

calendars.db

stores.db

users.db

Diploma

config\_data

config.py

handlers

\_\_init\_\_.py

admin.py

other\_handlers.py

registration\_user.py

keyboards

keyboards.py

.py

lexicon

lexicon.py

.py

\_\_init\_\_.py

bot.py

time\_messages.py

crud\_funcions\_user.py

crud\_funcions\_store.py

crud\_funcions\_calendar.py

requirements.txt

runner\_tests.log

*конфигурация доступа к токену бота*

*хендлеры админ-панели*

*хендлеры основного меню*

*хендлеры регистрации пользователя*

*клавиатуры бота*

*словари с именами кнопок*

*исполняемый файл*

*функции отправки сообщений по расписанию*

*функции обращения к БД Users (пользователей)*

*функции обращения к БД Stores (покупок)*

*функции обращения к БД Calendars (мероприятий)*

*файл зависимостей*

*файл логгирования*

*БД пользователей*

*БД покупок*

*БД мероприятий*

*токен бота*

# **Приложение 2. Список необходимых библиотек**

aiofiles==24.1.0

aiogram==3.16.0

aiogram\_calendar==0.6.0

aiohappyeyeballs==2.4.4

aiohttp==3.11.11

aioschedule==0.5.2

aiosignal==1.3.2

annotated-types==0.7.0

anyio==4.7.0

APScheduler==3.11.0

async-timeout==4.0.3

attrs==24.3.0

Babel==2.9.1

cachetools==5.5.0

certifi==2024.12.14

charset-normalizer==3.4.0

environs==11.2.1

exceptiongroup==1.2.2

frozenlist==1.5.0

h11==0.14.0

httpcore==1.0.7

httpx==0.28.1

idna==3.10

Jinja2==3.1.5

magic-filter==1.0.12

MarkupSafe==3.0.2

marshmallow==3.23.2

multidict==6.1.0

packaging==24.2

propcache==0.2.1

pydantic==2.10.4

pydantic\_core==2.27.2

python-dotenv==1.0.1

pytz==2024.2

sniffio==1.3.1

typing\_extensions==4.12.2

tzdata==2024.2

tzlocal==5.2

yarl==1.18.3