**Сравнение различных подходов к реализации асинхронного программирования: asyncio, threading и multiprocessing**  
  
Содержание

1. [Введение](#_heading=h.gjdgxs) 2

[Обоснование](#_heading=h.30j0zll) выбора темы 2

[Определение цели и задач исследования 2](#_heading=h.30j0zll)

2. [Основные понятия и определения 2](#_heading=h.3znysh7)

[3 .Методы и подходы к разработке 3](#_heading=h.2et92p0)

[Архитектура телеграмм-бота](#_heading=h.3dy6vkm) 3

[Обеспечение](#_heading=h.1t3h5sf) безопасности 5

4. [Обзор](#_heading=h.4d34og8) популярных инструментов асинхронного программирования 6

[aiogram](#_heading=h.gjdgxs) 6

thread 6

multiprocessing [7](#_heading=h.30j0zll)

5. [Разработка в соответствии с созданной документацией 9](#_heading=h.1fob9te)

[Планирование разработки 9](#_heading=h.3znysh7)

[Разработка 9](#_heading=h.2et92p0)

6. [Анализ и интерпретация результатов](#_heading=h.tyjcwt) 14

7. [Заключение 15](#_heading=h.3dy6vkm)

[Приложение. Список необходимых библиотек 16](#_heading=h.4d34og8)

Введение (Обзор проекта)

**1. Обоснование выбора темы:**  
  
1. Асинхронное программирование представляет собой мощный подход к разработке программного обеспечения, позволяющий значительно повысить эффективность выполнения задач и улучшить пользовательский опыт.

2. Потребности рынка: в современном мире, где приложения становятся все более сложными и многозадачными, асинхронное программирование играет ключевую роль в обеспечении высокоскоростной обработки данных и непрерывного взаимодействия с пользователем. Этот подход позволяет программам выполнять несколько задач одновременно, не блокируя выполнение других операций, что особенно важно для приложений, требующих быстрой реакции и масштабируемости.

3. Практическая значимость: асинхронное программирование позволяет выполнять несколько задач одновременно, не блокируя выполнение других операций, что особенно важно для приложений, требующих быстрой реакции и масштабируемости.   
  
4. Личный интерес и потенциальные перспективы: Использование различных инструментов и создание продуктов, сочетающих в себе разные технологии, позволяет развить и отработать опыт создания интересных и разносторонних проектов. Кроме того, такие навыки востребованы на рынке труда, что открывает перспективы для карьерного роста и развития.  
Таким образом, выбор темы "Сравнение различных подходов к реализации асинхронного программирования: asyncio, threading и multiprocessing." обусловлен ее актуальностью, потребностями рынка, практической значимостью и личным интересом, что делает эту тему подходящей для проведения дипломной работы.

**2. Основные понятия и определения**  
  
**Обзор основных понятий в области тестирования эффективности предобученных моделей распознавания объектов:**  
  
**1. Фреймворк**: Программная платформа, которая предоставляет готовые компоненты и инструменты для разработки асинхронных программ. На практике часто используются **multiprocessing, threading, aiogram.**  
  
**2.** **Телеграмм-бот**: Программное приложение, которое работает на веб-сервере и доступно через Телеграмм. В контексте данного проекта, Телеграмм-бот будет использоваться для сравнения и демонстрации эффективности функционала бота в ответ на действия пользователя.  
  
**3. Многопоточность** – это способность программы выполнять несколько потоков одновременно. Каждый поток представляет собой отдельный поток выполнения, который может выполнять задачи независимо от других потоков. Это позволяет приложению параллельно выполнять несколько операций.

**4. Параллелизм:** многопоточность позволяет одновременно выполнять несколько потоков на многопроцессорных системах или многоядерных процессорах. Это может улучшить производительность при выполнении вычислительных задач, таких как сложные математические вычисления или обработка больших объемов данных.

**5. Изоляция потоков:** каждый поток имеет собственное пространство памяти и может выполнять задачи независимо от других потоков. Это означает, что поток может изменять данные без влияния на другие потоки.

**6. Сложность управления:**управление потоками может быть сложным, поскольку необходимо координировать доступ к общим ресурсам и предотвращать проблемы, такие как гонки данных и взаимные блокировки.

Примеры применения:

Выполнение параллельных вычислений, например, в научных вычислениях или моделировании. Обработка нескольких пользовательских запросов на сервере.

**7. Асинхронность** – это техника, при которой задачи выполняются без блокировки основного потока выполнения. Вместо того чтобы ждать завершения одной задачи перед началом другой, приложение может продолжать выполнять другие задачи, пока асинхронная операция выполняется в фоновом режиме.

**8. Однопоточность или многопоточность:** асинхронное программирование может быть реализовано в одном потоке (например, в JavaScript или Python с использованием асинхронных функций) или с использованием нескольких потоков (например, с помощью асинхронных библиотек в многопоточных языках).

**9.** **Многопроцессорность** — метод, при котором используется несколько процессов для параллельного выполнения кода. Это обеспечивает более эффективное использование многоядерных процессоров

Данные термины помогают понять ключевые аспекты проверки эффективности предобученных моделей для распознавания объектов в веб приложении.

**3. Методы и подходы к разработке**  
  
**Архитектура телеграмм-бота**   
  
Проектирование архитектуры для ботов — это ключевой этап разработки, который определяет способность бота обслуживать множество пользователей одновременно, обеспечивать высокую производительность и надежность.

Разделение на компоненты  
Одним из основных принципов при проектировании архитектуры для высоконагруженных телеграмм-ботов является разделение на компоненты. Это означает, что бот должен быть разбит на отдельные модули или сервисы, каждый из которых отвечает за определенные функции. Такой подход облегчает масштабирование, обновление и поддержку бота.

Модульная структура  
Для обеспечения гибкости и удобства разработки, высоконагруженный телеграмм-бот должен иметь модульную структуру. Модульность позволяет легко добавлять новые функции, изменять существующие и поддерживать код без необходимости переписывания всего приложения.

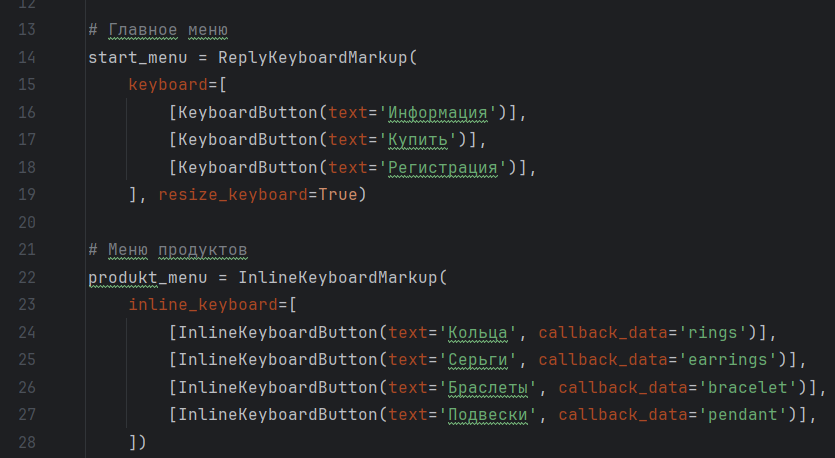


Рисунок 1 Модульная структура

Использование асинхронности для эффективной обработки запросов.  
Эффективная обработка запросов в режиме реального времени является ключевой особенностью высоконагруженных ботов. Для достижения высокой производительности и скорости ответа, широко применяется асинхронное программирование.

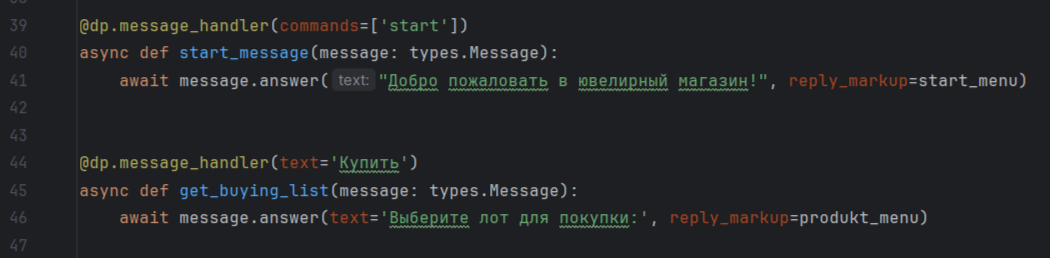


Рисунок 2 Пример асинхронности обработки запросов

Работа с базой данных

Работа с базой данных является неотъемлемой частью разработки высоконагруженных телеграмм-ботов. Эффективное хранение и управление данными позволяют обеспечить высокую производительность и надежность бота.

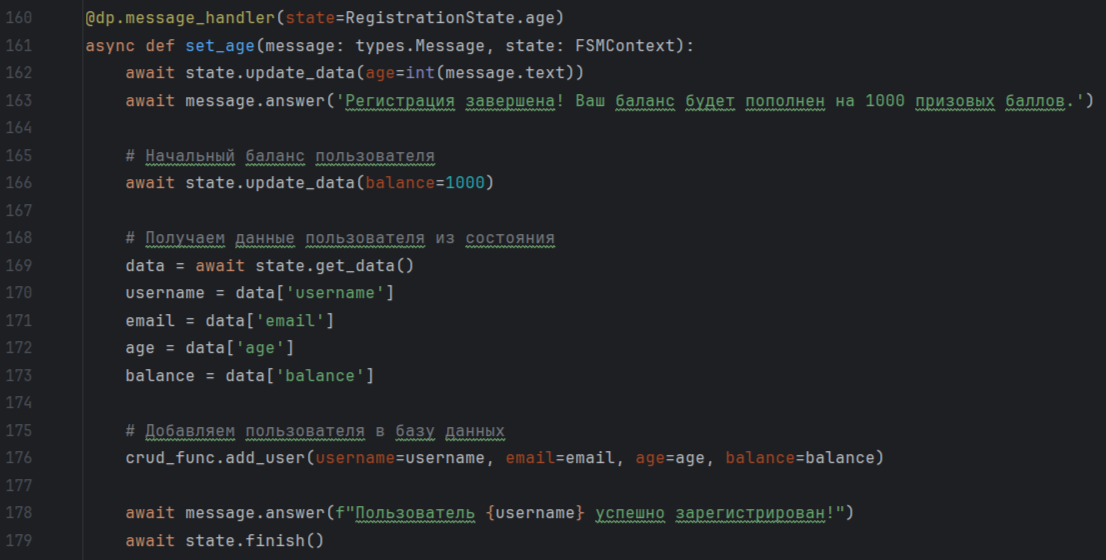


Рисунок 3 Работа с базой данных

Обеспечение безопасности  
Аутентификация и авторизация: можно настроить авторизацию пользователей, например, чтоб у каждого был доступ к балансу призовых или накопленных баллов.

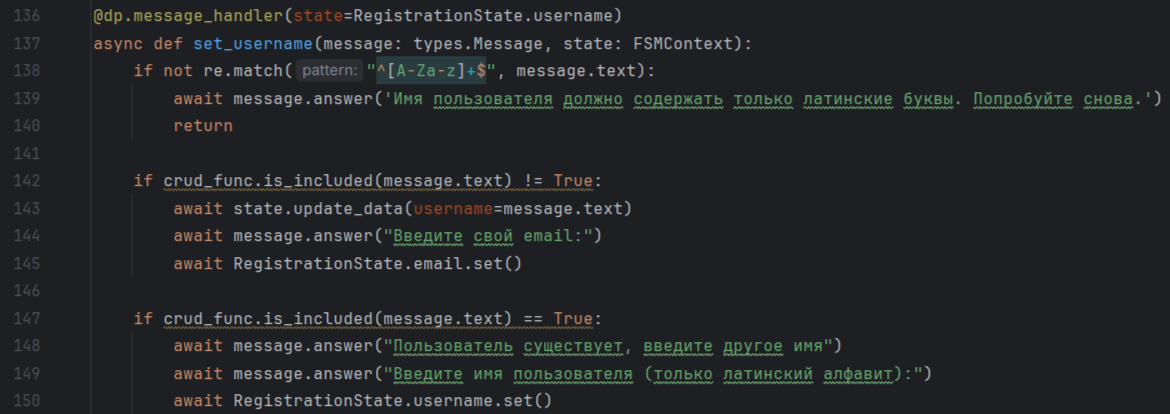


Рисунок 4 Регистрация пользователей

**4.** [**Обзор**](#_heading=h.4d34og8) **популярных инструментов асинхронного программирования**Инструменты, которые мы рассмотрим – **multiprocessing, threading, aiogram.**

Ниже более подробное описание.  
  
**aiogram**  
  
**aiogram** —фреймворк для разработки телеграмм-бота на Python.  
  
Основные возможности:  
- Асинхронный подход к выполнению кода. Это позволяет не останавливать работу бота в ожидании ответа пользователя.

- Поддержка Telegram Bot API 6.9 и быстрые обновления до последних версий Bot API.

- Автогенерация кода для интеграции с Telegram Bot API. Его можно легко перегенерировать при обновлении API.

- Использование конечного автомата.

- Мощные магические фильтры.

- Поддержка ответов в Webhook (то есть возможность делать запросы в ответ на обновления).

Особенности:  
Особенность заключается в большом наборе инструментов, которые можно использовать для добавления дополнительных функций и настроек бота.  
  
**Threading**  
  
**Threading в Python** — это **библиотека для многопоточности**.  Она позволяет выполнять несколько задач в одном процессе параллельно, что помогает улучшить отзывчивость программы и увеличить общую производительность.

Некоторые возможности threading:

* Класс Thread для создания и управления потоками выполнения. Позволяет запускать функции в новом потоке.
* Классы Lock, Event, Condition, Semaphore для синхронизации потоков и предотвращения состязаний при доступе к общим ресурсам.
* Класс Timer для выполнения функции через определённое время.
* Класс Barrier для организации точек синхронизации, где потоки могут остановиться и дождаться друг друга.
* Класс local для хранения данных в потоке-локальном хранилище, доступном только в рамках данного потока.
* Функция enumerate для получения списка всех активных потоков в программе.
* Функция current\_thread для получения объекта текущего исполняющегося потока.

Использование threading целесообразно, например, при обработке нажатия кнопки в графическом интерфейсе, когда по нажатию кнопки требуется осуществлять множество действий, или при загрузке файлов из сети и одновременной обработке уже загруженных элементов.

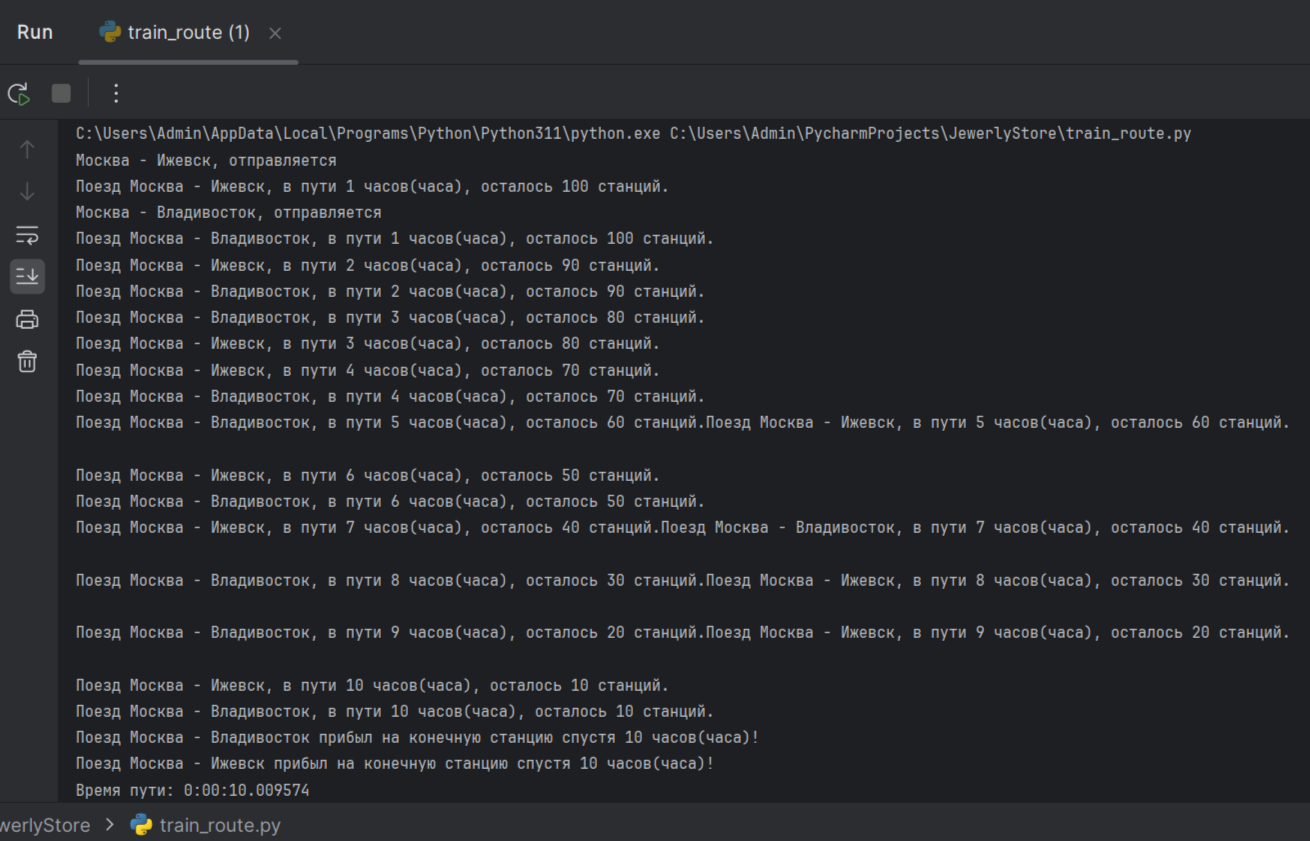


Рисунок 5 Пример использования threading

**Multiprocessing**

Возможности модуля multiprocessing в Python включают:

* Создание и управление процессами. Класс Process позволяет запускать функции в новом процессе.
* Параллельное выполнение задач. Класс Pool предоставляет пул процессов для выполнения задач параллельно, позволяя управлять пулом процессов и распределять задачи между процессами.
* Обмен данными между процессами. Класс Queue обеспечивает безопасность при работе с разделяемыми данными, позволяя передавать сообщения или результаты между экземплярами процессов.
* Управление разделяемыми объектами между процессами. Класс Manager позволяет создавать разделяемые списки, словари, очереди и другие объекты.
* Синхронизация процессов. Классы Lock, Event, Condition, Semaphore помогают синхронизировать процессы и предотвращать состязания при доступе к общим ресурсам.
* Обмен данными между двумя процессами. Механизм Pipe позволяет обмениваться данными между двумя процессами через двусторонний канал.

Использование модуля multiprocessing позволяет эффективно использовать ресурсы многопроцессорной системы, ускорить выполнение задач и реализовать параллельное выполнение вычислений.

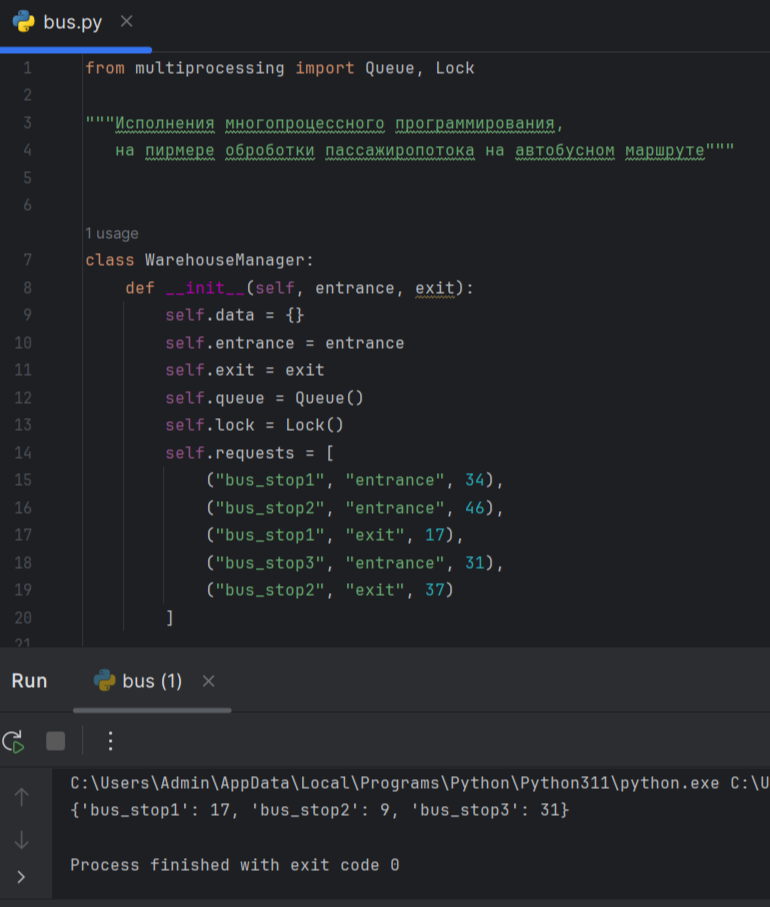


Рисунок 6 Пример использования multiprocessing

**5. Разработка в соответствии с созданной документацией**  
  
**Планирование разработки**  
  
Разработка была разделена на несколько основных этапов: формирование модулей инлайн-клавиатуры, разработка регистрации пользователей, реализация логики телеграмм-бота.  
  
**Разработка**

Разработаны стартовые команды, информация о боте.

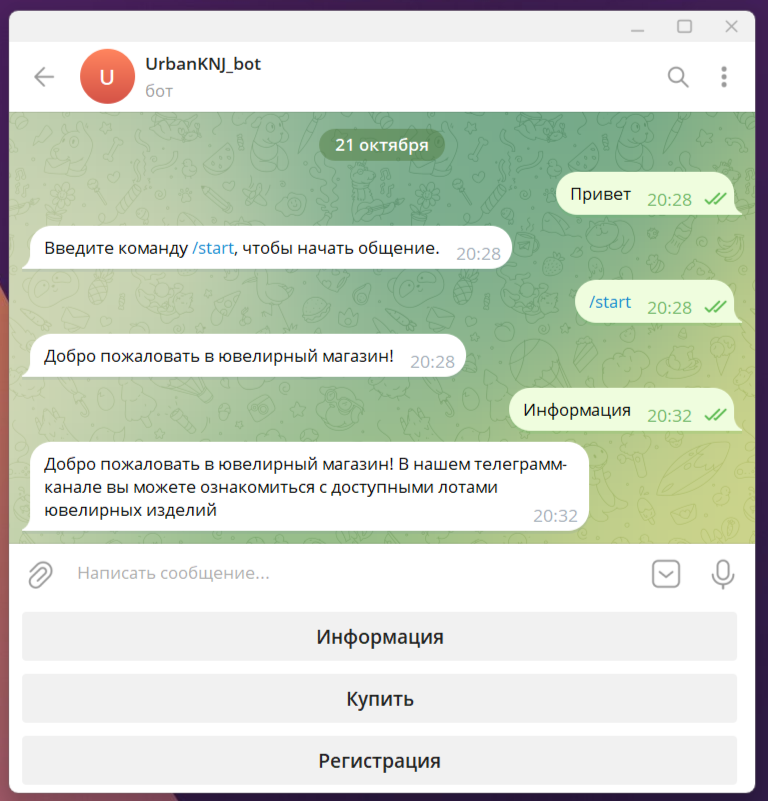


Рис. 7 Стартовые команды

Реализован модуль регистрации.



Рис. 8 Регистрация пользователя

Реализация разработки инлайн-клавиатуры.

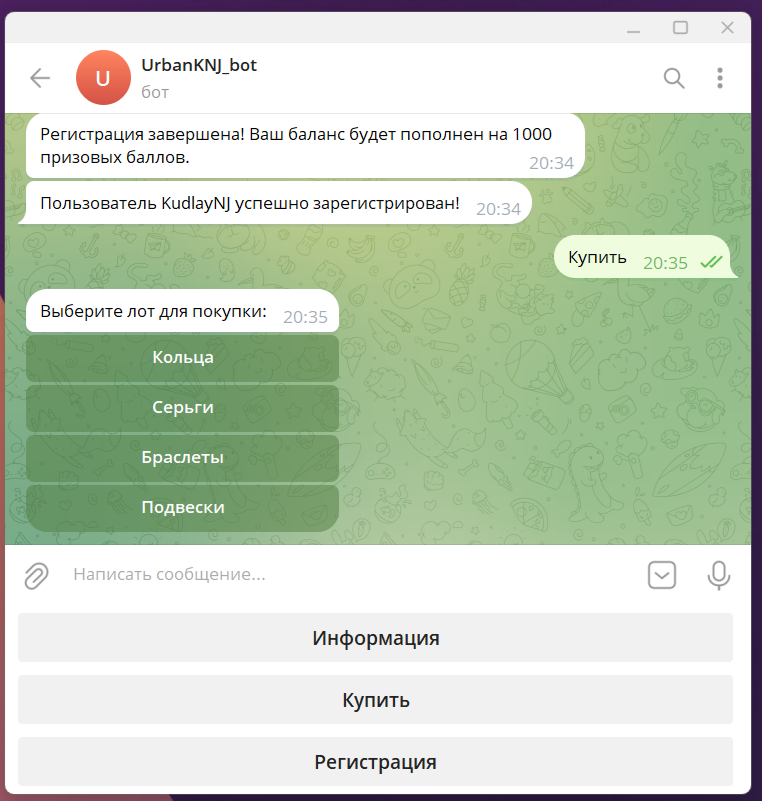


Рис. 9 Пример реализации инлайн-клавиатуры

Реализация просмотра ассортимента товаров, а также отправка заявки на обратный звонок.

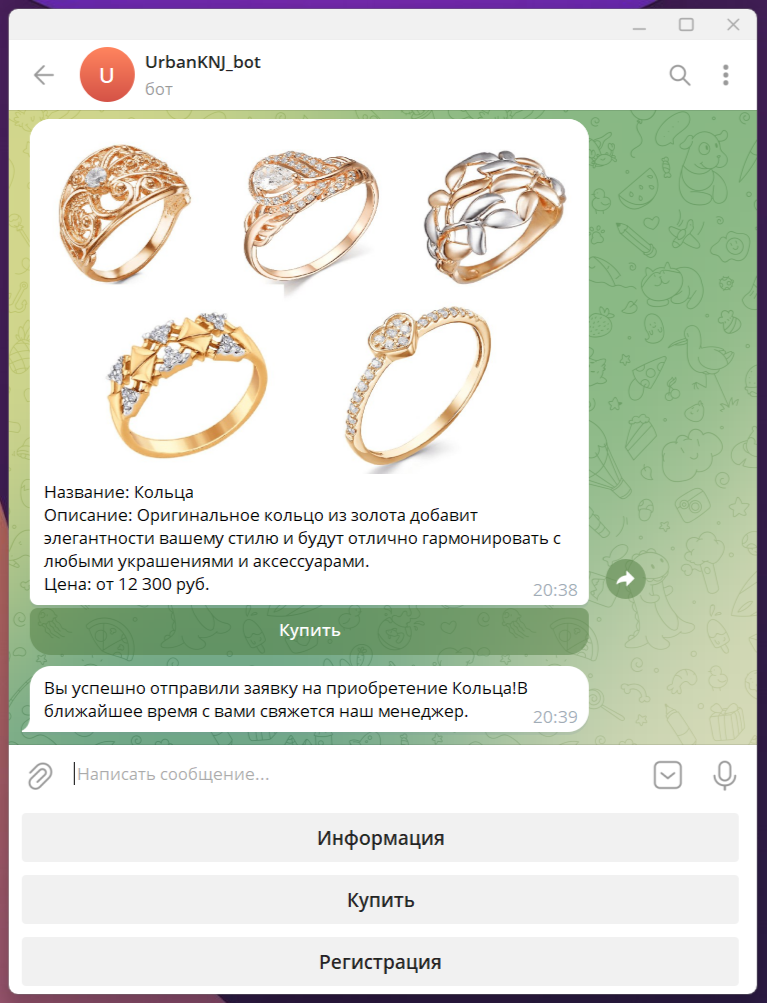


Рис. 10 Пример реализации онлайн-показа товара

C Aiogram важно понимать особенности асинхронного программирования. Эта концепция позволяет боту обрабатывать несколько задач одновременно без блокирования выполнения кода. Именно такие возможности делают его более отзывчивым и эффективным при обслуживании множества пользователей.

Для реализации таких функций Aiogram использует специальные команды:

* **async**– говорит программе, что функция может выполняться по частям, а не всё сразу;
* **await**– используется, когда программа должна подождать, пока одна часть задачи закончится, прежде чем двигаться дальше.

Это ключевые элементы асинхронного кода, которые позволяют выполнять параллельную обработку задач, не теряя время на ожидание.

Вот пример простого хендлера для команды **/start**, который отправляет пользователю приветственное сообщение:

@dp.message\_handler(commands=['start'])  
async def start\_message(message: types.Message):  
 await message.answer("Добро пожаловать в ювелирный магазин!", reply\_markup=start\_menu)

В этом примере @dp.message\_handler(commands=['start']) определяет хендлер, который реагирует на команду /start. А функция start асинхронно обрабатывает сообщения и использует await для отправки ответа, что позволяет боту продолжать обработку других событий во время ожидания отправки сообщения.

Хендлеры в Aiogram – это функции, которые определяются для обработки различных типов сообщений или команд от пользователя. В нашем примере хендлер принимает команду /start и отправляет приветственное сообщение.

Стоит отметить, что библиотека Aiogram содержит множество других полезных функций. Например, возможности для бота публиковать фото, видео, управлять клавиатурой и так далее. Все это внешние декоративные функции, но они позволяют очень разнообразить деятельность электронного ассистента.

Далее в разработке мы используем хендлер callback\_query\_handler, его функционал заключается в обработке callback-запросов (событий, возникающих при нажатии на кнопку или действии внутри сообщения бота).

Декоратор callback\_query\_handler ожидает callback и принимает лямбда-функцию для его фильтрации.  Например, если передать анонимную функцию, которая принимает объект call и всегда возвращает True, то этот обработчик будет активироваться для любых callback-запросов.

Также с помощью callback\_query\_handler можно установить обработчик для отдельных callback-запросов: будет вызван тот обработчик, для которого лямбда-функция вернёт True. Это нужно, чтобы сделать разные обработчики для разных кнопок.

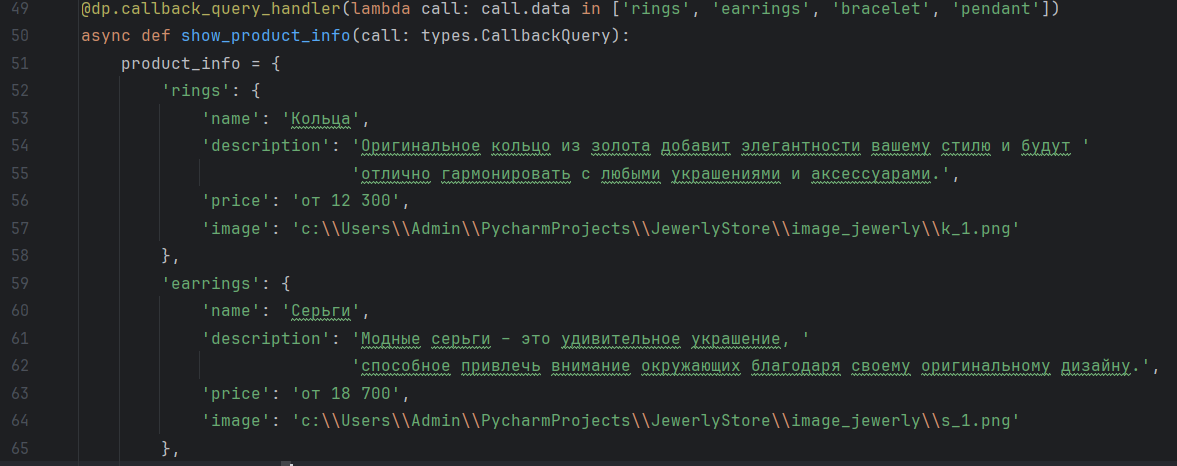


Рис. 11 Пример реализации хендлера callback\_query\_handler

В модульной структуре проекта имеется два вида клавиатуры кнопок, рассмотрим каждый отдельно.

Для реализации кнопок главного меню мы используем ReplyKeyboardMarkup.

ReplyKeyboardMarkup — это шаблоны сообщений. К примеру, ваш бот задаёт пользователю вопрос и предлагает варианты ответа. Пользователь может самостоятельно напечатать ответ, либо нажать на готовую кнопку. Такая клавиатура показывается вместо основной и не привязана ни к какому сообщению. В кнопки такой клавиатуры нельзя заложить никакой информации, отправлено будет только то, что написано на кнопке.

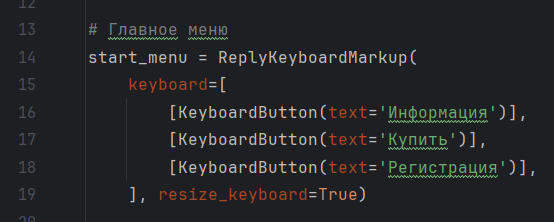


Рис. 12 Пример реализации ReplyKeyboardMarkup

Также в aiogram имеется InlineKeyboardMarkup, в Telegram используется для создания инлайн-кнопок, связанных с сообщениями в чате.

Такие кнопки позволяют скрыть в себе внутреннюю телеграм-ссылку, ссылку на внешний ресурс, а также шорткат для инлайн-запроса.

Например, использовать InlineKeyboardMarkup можно при инициализации клавиатуры, передав в качестве аргумента row\_width число, которое определяет, сколько кнопок будет находиться в одном ряду. Затем для каждой кнопки создать отдельную переменную и инициализировать класс Button. В параметрах указать текст, который будет показан на кнопке в мессенджере, и URL страницы, на которую пользователь будет переходить при нажатии на кнопку.

Также с помощью InlineKeyboardMarkup можно редактировать инлайн-кнопки вместе с сообщением, к которому они прикреплены, или только саму разметку.

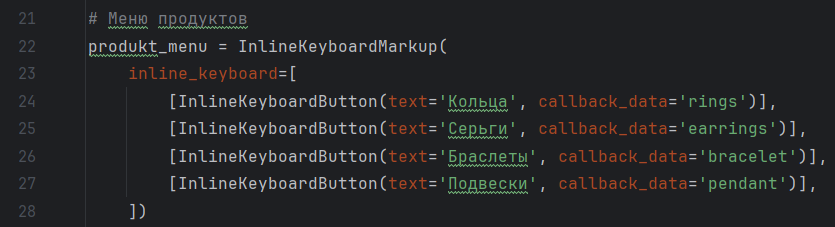


Рис. 13 Пример реализации InlineKeyboardMarkup

Для регистрации пользователя и прочих функций в разработке проекта использовался такой инструмент как Машина состояний.

State в Python-aiogram — это конкретное состояние, в котором может находиться пользователь. Оно позволяет боту запоминать предыдущие действия пользователя и сохранять информацию о его выборах. Например, состояние может содержать информацию о текущем шаге диалога, выбранной опции или настройках пользователя.

Для определения состояний в Aiogram используется класс State. Например, можно создать класс Form, который наследуется от StatesGroup, и внутри него определить состояния name и age. Это поможет структурировать и управлять последовательностью шагов анкеты.

Также в aiogram имеется FSMContext — специальный объект, который помогает управлять состояниями пользователя. Он хранит данные о текущем состоянии пользователя и позволяет изменять их, перемещая пользователя между различными состояниями.

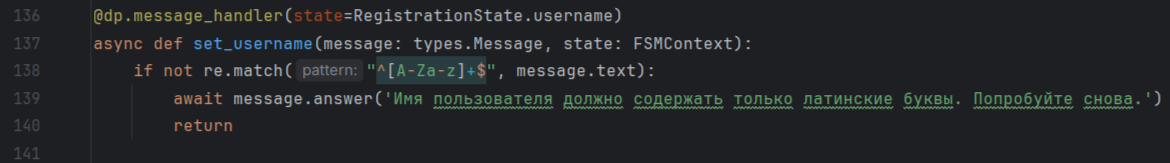


Рис. 14 Пример реализации машины состояний

**6. Анализ и интерпретация результатов**  
  
**Сравнение моделей**  
  
Были протестированы различные методы асинхронного программирования.  
Была произведена оценка различных инструментов асинхронного программирования и анализ различий функциональности методов.  
  
**Интерпретация результатов**  
**aiogram** качественно исполняет асинхронный подход к выполнению кода. Это позволяет не останавливать работу бота в ожидании ответа пользователя.

**Threading** позволяет выполнять несколько задач в одном процессе параллельно, что помогает улучшить отзывчивость программы и увеличить общую производительность.

**multiprocessing** позволяет создавать и управлять процессами, обмениваться данными между процессами, использовать пулы процессов и другие механизмы для параллельного выполнения задач.

**Рекомендации по выбору модели в зависимости от проекта**  
  
Для задач, требующих высокой точности и допускающих большее время отклика, рекомендуется использовать **threading**.

Для задач, где важна скорость обработки, можно рассмотреть использование **multiprocessing**.  
  
Для задач реализации ботов, которые могут работать параллельно с несколькими пользователями, не ожидая ответа от каждого из них рекомендуется использовать **aiogram**.

**7. Заключение**  
  
**Обзор выполненной работы**  
  
Проектирование и разработка веб-приложения для распознавания объектов были успешно завершены в соответствии с изначально созданной документацией. Приложение включает функционал загрузки изображений, обработки с использованием модели VGG и отображения результатов.  
  
Реализованное приложение соответствует требованиям и демонстрирует высокую точность распознавания объектов.  
  
Модель VGG показала хорошие результаты в тестах, однако возможно использование других моделей для улучшения времени отклика.  
  
**Дальнейшие планы**  
  
Будет полезным создание сводной статистики по всем моделям, со всеми метриками эффективности, чтобы пользователи могли самостоятельно выбирать наиболее подходящую модель на основе полных данных.  
  
Планируется добавление новых функций, таких как сохранение истории загрузок и результатов, улучшение интерфейса и расширение функциональности.

# Приложение 2. Список необходимых библиотек

aiodns==3.2.0  
aiohttp\_socks==0.9.0  
bpython==0.24  
brotli==1.1.0  
brotlicffi==1.1.0.0  
colorama==0.4.6  
ConfigParser==7.1.0  
cryptography==43.0.3  
cx\_Oracle==8.3.0  
Cython==3.0.11  
cython==3.0.11  
dl==0.1.0  
docutils==0.21.2  
email\_validator==2.2.0  
eval\_type\_backport==0.2.0  
filelock==3.16.1  
gunicorn==23.0.0  
HTMLParser==0.0.2  
hypothesis==6.115.3  
ini2toml==0.15  
ipdb==0.13.13  
ipython==8.12.3  
ipywidgets==8.1.5  
jnius==1.1.0  
keyring==25.4.1  
mod==0.3.0  
motor==3.6.0  
MySQL-python==1.2.5  
numpy==2.1.2  
oracledb==2.4.1  
path.py==12.5.0  
Pillow==11.0.0  
pkg1==0.0.3  
protobuf==5.28.2  
psycopg==3.2.3  
psycopg2==2.9.10  
psycopg\_pool==3.2.3  
pylibmc==1.6.3  
pymemcache==4.0.0  
pyOpenSSL==24.2.1  
pytest==8.3.3  
python-dotenv==1.0.1  
pywatchman==2.0.0  
PyYAML==6.0.2  
sets==0.3.2  
Sphinx==8.1.3  
tblib==3.0.0  
thread==2.0.5  
toml==0.10.2  
tomli\_w==1.1.0  
trove\_classifiers==2024.10.21.16  
urllib3\_secure\_extra==0.1.0  
uvloop==0.21.0  
xmlrpclib==1.0.1  
xx==3.3.2

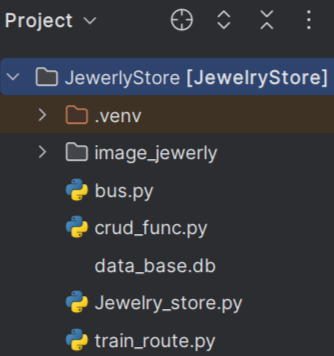


Рис. 15 Файловая структура проекта