Дипломная работа

по теме

«Сравнение различных подходов к реализации асинхронного программирования: asyncio, threading и multiprocessing – на примере сайта вуза»

Выполнил:

Феофанов В. А.

[1. Введение 3](#_Toc182842519)

1.1 [Общие понятия. 3](#_Toc182842520)

1.2 [Асинхронность в Python. 3](#_Toc182842521)

[2. Обзор проекта 4](#_Toc182842522)

[3. Структура проекта 7](#_Toc182842523)

[4. Использование asyncio, threading и multiprocessing в проекте 11](#_Toc182842524)

4.1[asyncio 11](#_Toc182842525)

4.2 [threading 12](#_Toc182842527)

4.3 [multiprocessing 12](#_Toc182842528)

[5. Заключение 13](#_Toc182842529)

[Приложение 1. Файловая структура проекта 14](#_Toc182842530)

[Приложение 2. Список необходимых библиотек 15](#_Toc182842531)

## Введение

### 1.1 Общие понятия.

Асинхронное программирование – это подход к написанию кода, при котором несколько задач могут выполняться параллельно и независимо друг от друга. Это отличается от синхронного программирования, где операции выполняются последовательно, и каждая следующая начинается только после завершения предыдущей. Асинхронный подход позволяет грамотнее распределить ресурсы компьютера и при правильной реализации ускорить выполнение программы.

Однако у рассматриваемого метода есть и недостатки. Одним из них является состояние гонки (race condition). Это ситуация, возникающая в многопоточных системах, когда результаты работы программы зависят от порядка выполнения потоков (процессов) и могут меняться от запуска к запуску. Как правило, такая проблема возникает, если потоки обращаются к общим ресурсам и изменяют их. Это необходимо учитывать при проектировании программы и настраивать синхронизацию между процессами.

### 1.2 Асинхронность в Python.

В языке программирования Python есть три основных подхода к асинхронному программированию: с помощью библиотеки asyncio, threading или multiprocessing.

В первом случае на уровне операционной системы выполнение программы не изменяется. Асинхронные функции, объявленные с помощью ключевого слова async (корутины), выполняются последовательно, однако, если корутина “задерживается” (например ожидает данные на вход), она приостанавливается и выполняется следующая команда. Когда “задержавшаяся” функция будет готова продолжить, она запустится с момента остановки.

Библиотека threading позволяет запустить несколько потоков внутри одного процесса. Потоки относительно нетребовательны к ресурсам компьютера и имеют общее пространство памяти, что упрощает обмен данными. Однако они тоже не выполняются одновременно из-за глобальной блокировки интерпретатора (GIL – Global Interpreter Lock), которая позволяет работать только одному потоку в данный момент времени.

Наконец, multiprocessing создает несколько процессов, которые распределяются по ядрам компьютера и, как следствие, способны обрабатываться одновременно. Это выгодно отличает их от асинхронных функций и потоков. Однако процессы не имеют общей памяти, их создание, обработка и взаимодействие значительно тяжеловеснее. Кроме того, возрастает количество возможных ошибок и багов, даже не уровне операционной системы.

Вследствие особенностей каждого подхода на практике они используются для разных задач. asyncio хороша для отправки и получения запросов в веб-сервисе из-за возможной долгой передачи данных по сети. threading подходит для ввода и вывода данных, которые не требуют большой вычислительной мощности. multiprocessing можно использовать для оптимизации обработки данных, которую можно разбить на независимые части и доверить разным процессам. Именно для таких задач будут использоваться рассматриваемые библиотеки в данной работе. Однако необходимо помнить о недостатках и особенностях каждого метода и аккуратно подходить к написанию кода.

1. Обзор проекта

В данном проекте рассматривается сайт, который представляет сильно упрощённую модель вуза. В институте есть несколько направлений, у каждого из которых задано название, максимальное количество будущих студентов, id и короткое название, использующееся в запросе к серверу для обращения к базе данных. На главной странице (рисунок 1) предоставлен список направлений со ссылкой на страницу каждого, где представлена таблица выбравших его (рисунок 2).

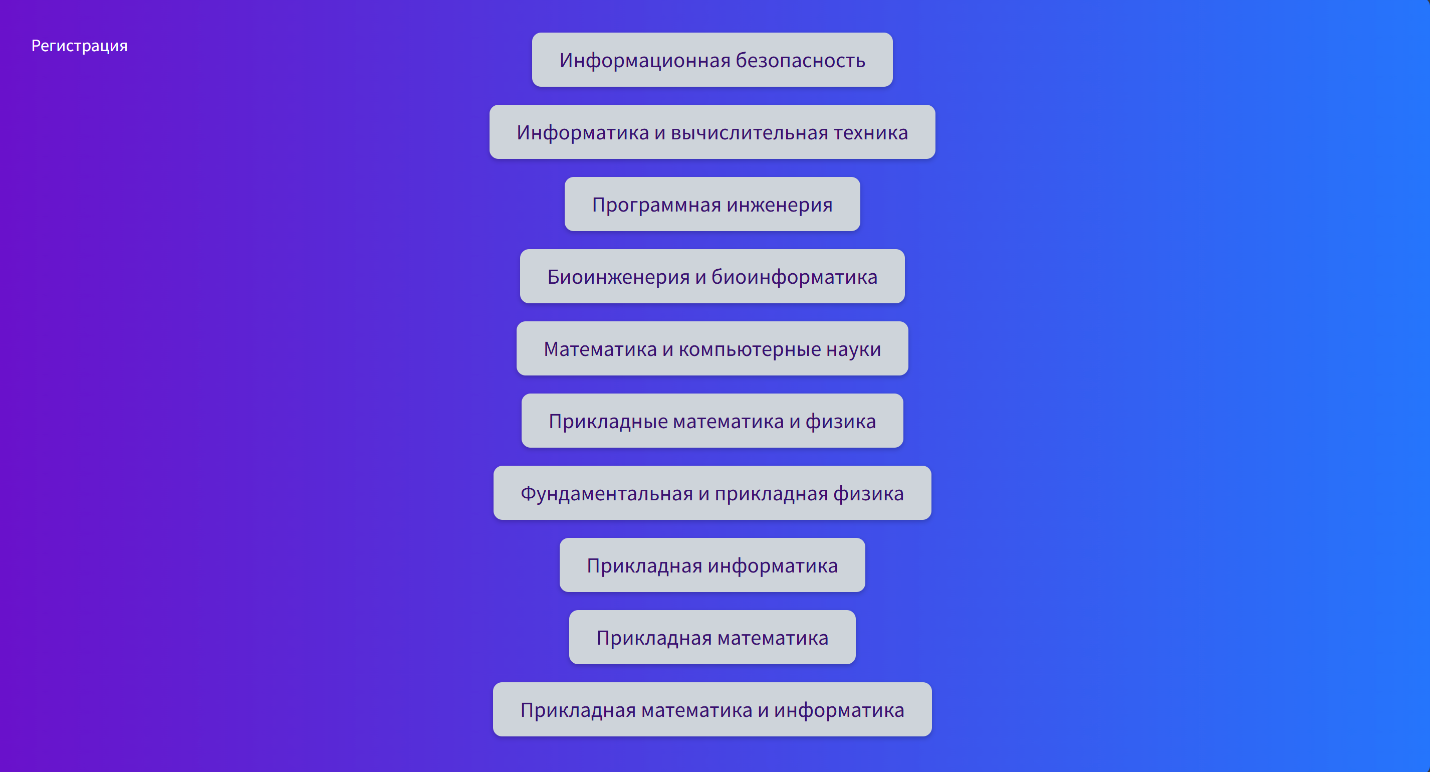


Рисунок 1 – Главная страница

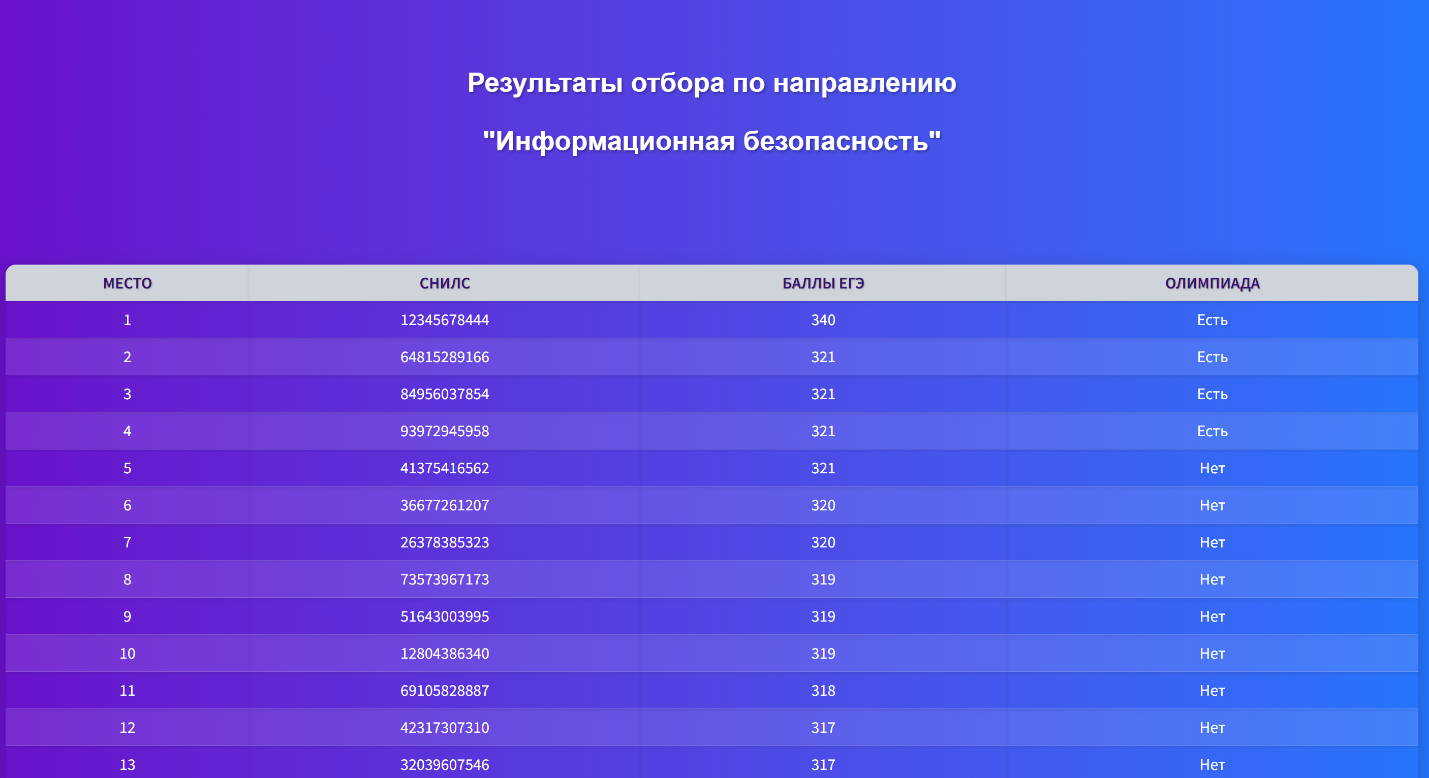


Рисунок 2 – Таблица абитуриентов для направления

“Информационная безопасность”

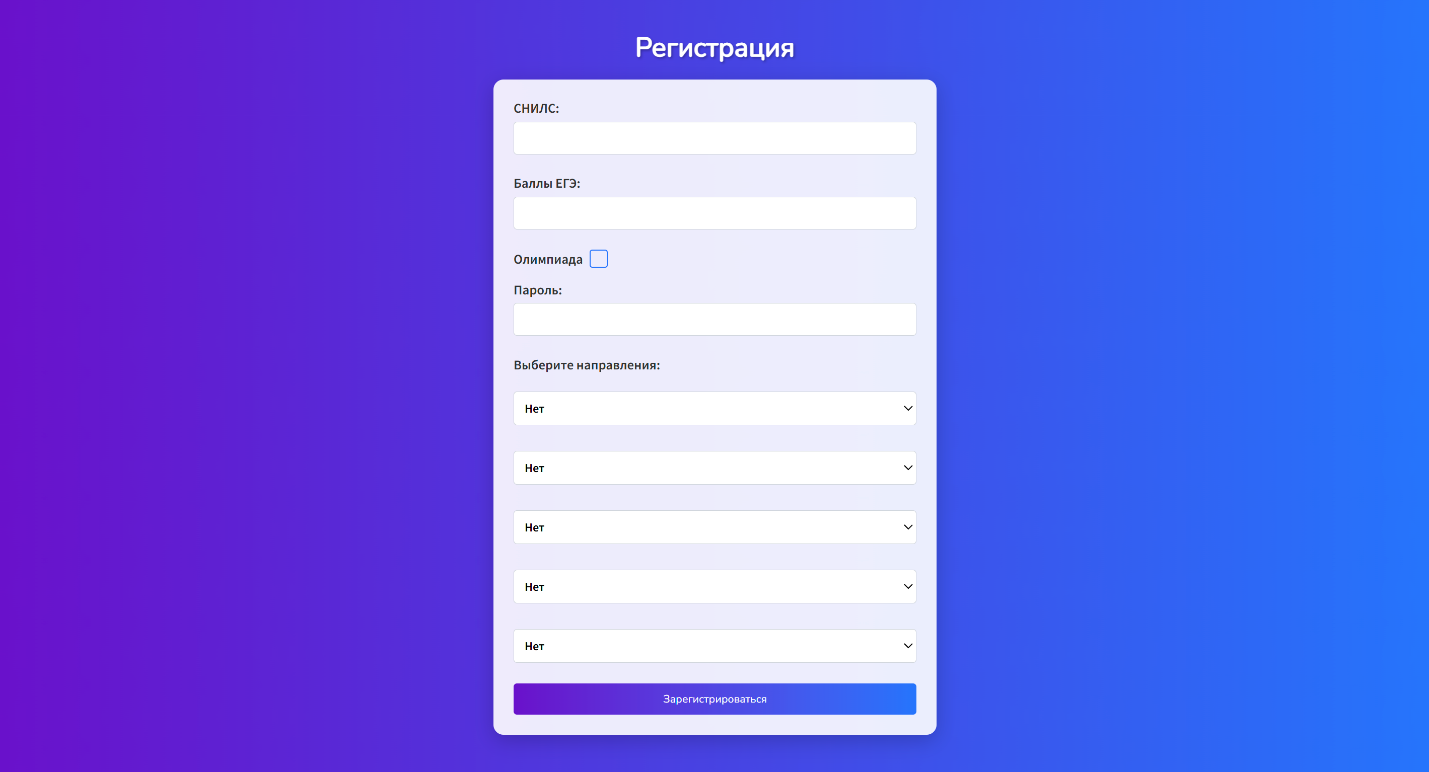
Абитуриент может зарегистрировать себя, пройдя по ссылке “Регистрация” и введя свои данные в специально отведенные поля (рисунок 4). 

Рисунок 4 – Форма регистрации

После регистрации появится страница с предложением перейти на главную (рисунок 5).

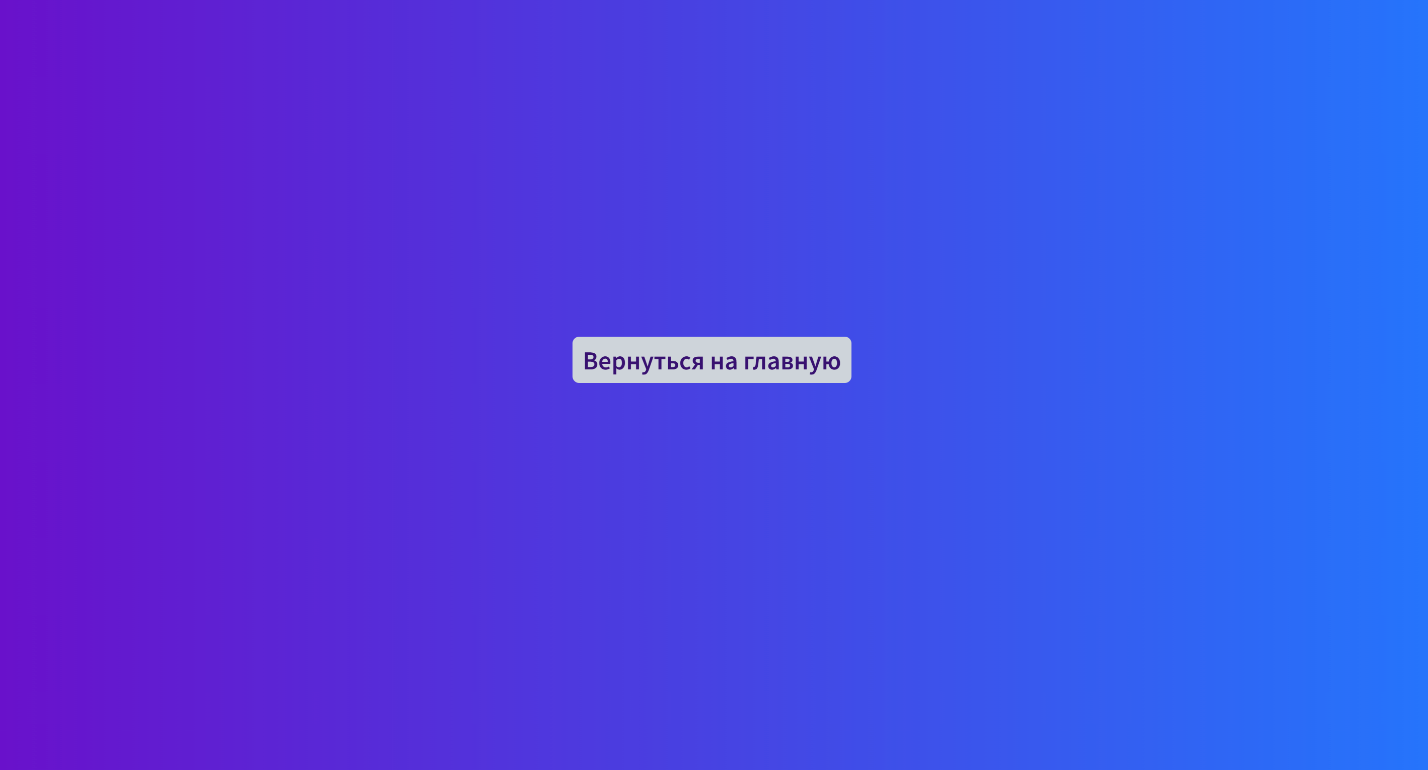


Рисунок 5 – Страница после успешной регистрации

1. Структура проекта

В приложении 1 приведена файловая структура проекта. Рассмотрим ее подробнее и объясним серверную логику сайта.

В папке backend находятся файлы для подключения к базе данных:

* В async\_database.py создается движок и фабрика, создающую сессии, для асинхронной работы с БД
* В database.py создается движок и фабрика, создающую сессии, для синхронной работы с БД
* В async\_database\_depends.py создается генератор сессий для асинхронной работы с БД
* В database\_depends.py создается генератор сессий для синхронной работы с БД
* В Base.py создается класс, базовый для наследующих его моделей, определяющих таблицы в БД
* В course\_sort.py определяются функции для сортировки данных абитуриентов, осуществляется такая сортировка и запись полученных значений в БД с помощью функций из fill\_db.py
* В fill\_db.py определяются функции для заполнения БД: есть методы, заполняющие базу случайными значениями и отдельно метод для записи отсортированных

Разбиение на синхронные и асинхронные сессии необходимо, так как асинхронные чтение и запись эффективнее синхронных, однако могут вызывать конфликты с потоками и процессами. Поэтому для взаимодействия с клиентом используются асинхронные сессии, в то время как для внутренней обработки данных на сервере – синхронные.

В папке migrations находятся данные о миграциях для обновления БД. Она создается автоматически с помощью модуля alembic.

В папке models объявляются модели таблиц БД:

* Applicant – таблица с зарегистрировавшимися абитуриентами (пример на рисунке 6)
* Course – таблица с доступными курсами (пример на рисунке 7)
* ApplicantCourse – вспомогательная таблица, реализующая отношение many-to-many между первыми двумя таблицами (пример на рисунке 8)
* Sorted – таблица с отсортированными абитуриентами (пример на рисунке 9)

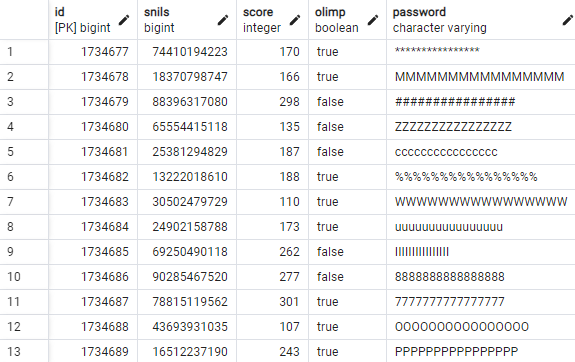


Рисунок 6 – Пример таблицы applicants

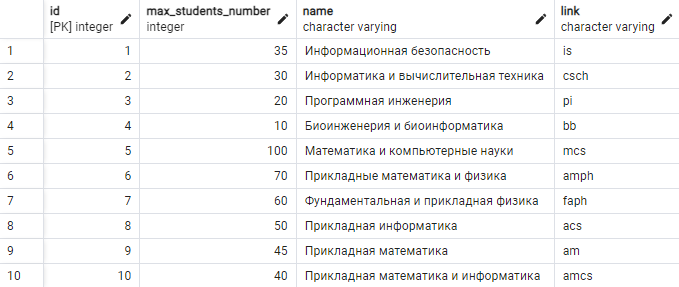


Рисунок 7 – Пример таблицы courses

Applicant и Course связаны отношением много-ко-многим (many-to-many), так как абитуриенты могут выбрать несколько направлений, а одно направление может принять несколько абитуриентов.

Sorted связана с Course отношением один-ко-многим (one-to-many), а с Applicant – один-к-одному (one-to-one).

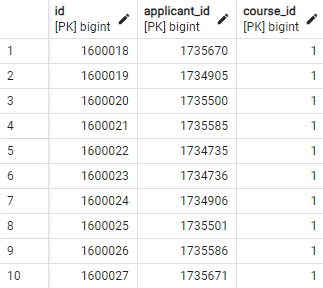


Рисунок 8 – Пример таблицы applicant\_courses

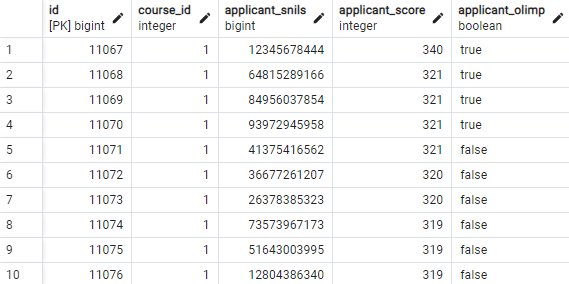


Рисунок 9 – Пример таблицы sorted

В папке routers объявляются роутеры:

* ApplicantRouter для регистрации
* ResultRouter для выдачи отсортированных списков поступающих

В папке schemas представлен файл CreateApplicant, в котором определена pydantic-модель для получения данных абитуриента из HTML-формы регистрации.

В папке static находятся CSS-файлы, определяющие стили страниц сайта.

В папке templates – HTML-шаблоны страниц сайта.

alembic.ini – файл, создающийся с помощью модуля alembic и необходимый для создания миграций.

В файле main.py создается само FastAPI-приложение, функция обработки запросов главной страницы и подключаются роутеры.

В файле README.md представлена инструкция по установке и эксплуатации приложения.

Файл requirements.txt содержит все необходимые библиотеки. Также они есть в приложении 2.

## 4. Использование asyncio, threading и multiprocessing в проекте

### 4.1 asyncio

В данном приложении asyncio не используется напрямую. Однако присутствуют асинхронные функции и асинхронные модули, в том или ином виде требующие asyncio. Этот подход в проекте используется для получения запросов от клиента и отправки ответа от сервера. Также с помощью расширения sqlalchemy.ext.asyncio осуществляется асинхронное взаимодействие с базой данных.

Асинхронные функции встречаются в файлах [routers\ResultRouter.py](routers/ResultRouter.py), [routers\ApplicantRouter.py](routers/ApplicantRouter.py), <main.py>, [backend\async\_database.py](backend/async_database.py), [backend\async\_database\_depends.py](backend/async_database_depends.py).

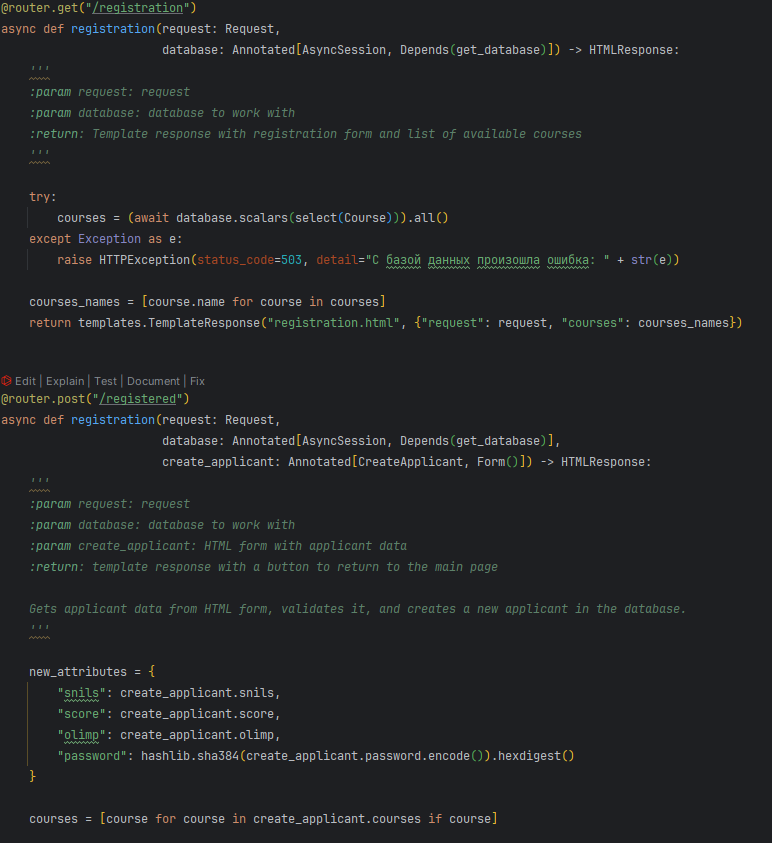


Рисунок 10 – Использование асинхронных функций

На рисунке 10 приведен пример использования асинхронных функций в проекте.

4.2 threading

threading используется в файле [backend\fill\_db.py](backend/fill_db.py) для оптимизации записи большого количества абитуриентов в таблицы. Отрывок из кода с применением threading приведен на рисунке 11.

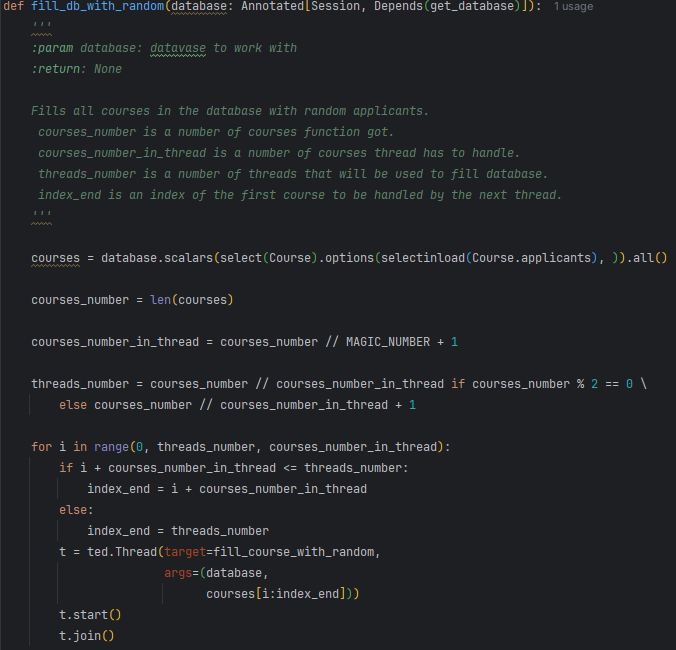


Рисунок 11 – Использование threading

### 4.3 multiprocessing

multiprocessing используется в файле [backend\course\_sort.py](backend/course_sort.py) для оптимизации сортировки данных. Отрывок из кода с применением multiprocessing приведен на рисунке 12.

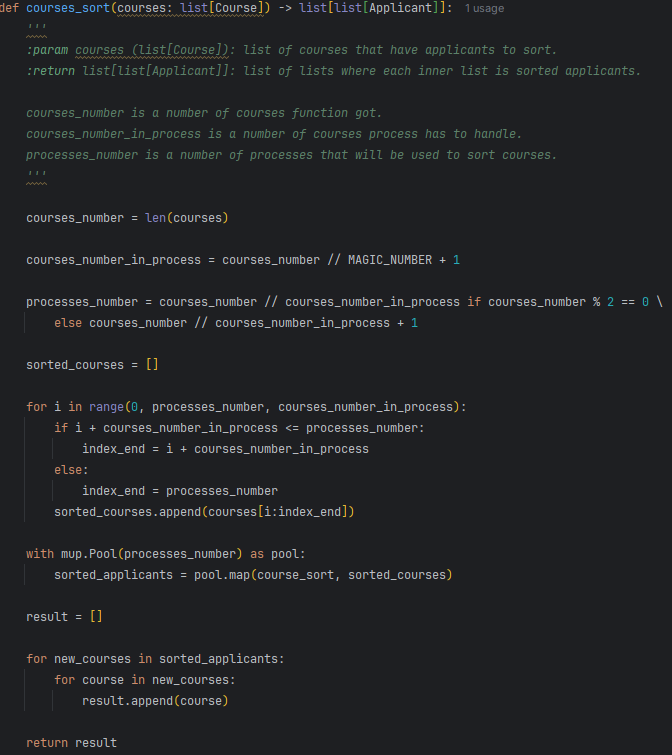


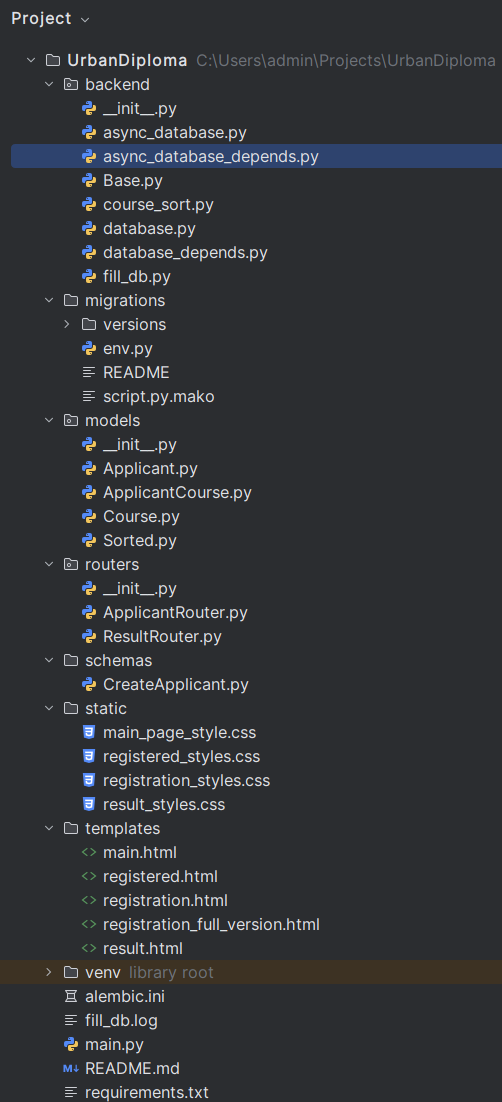
Рисунок 12 – Использование multiprocessing

1. Заключение

Это приложение представляет собой пример использования трех основных подходов к асинхронному программированию на языке Python: asyncio, threading, multiprocessing – в одном проекте, но для разных задач. Важно понимать, что асинхронное программирование, несмотря на свои очевидные преимущества, имеет значительные недостатки и является опасным при неосторожном применении. Поэтому необходимо аккуратно подходить к написанию асинхронного кода и обдумывать потребность в нем в рамках конкретной задачи.

К сожалению, функционал сайта чрезмерно простой и не подходит для использования в рамках реального вуза из-за множества сложностей в современной системе поступления, однако может использоваться как практика навыков и основа для более сложного проекта.

Приложение 1. Файловая структура проекта



Приложение 2. Список необходимых библиотек

alembic==1.14.0  
annotated-types==0.7.0  
anyio==4.6.2.post1  
asyncpg==0.30.0  
click==8.1.7  
colorama==0.4.6  
fastapi==0.115.4  
greenlet==3.1.1  
h11==0.14.0  
idna==3.10  
Jinja2==3.1.4  
Mako==1.3.6  
MarkupSafe==3.0.2  
psycopg2==2.9.10  
pydantic==2.9.2  
pydantic\_core==2.23.4  
python-multipart==0.0.17  
setuptools==75.4.0  
sniffio==1.3.1  
SQLAlchemy==2.0.36  
starlette==0.41.2  
typing\_extensions==4.12.2  
uvicorn==0.32.0  
wheel==0.45.0