Название проекта – Plagiat\_Notion\_2.0

Автор проекта: Парфёнова Лидия

Описание идеи: Telegram bot, написанный на библиотеке aiogram главная суть, чтобы пользователь мог записать важные дела и придать им описание и другие параметры, не скачивая никакое приложение, а пользуясь только telegram. Также есть функции: создание изображение, присылаемое ботом, на основе название профессии, отметка метки на карте, получение информации о погоде, информация о репозиториях

Описание реализации:

Бот написан не на классах, так как ботостроение не предлагает использование классов из-за асинхронности и особенности библиотек по ботам. Классов в проекте нет за исключение классов для ORM-моделей.

**Как запустить**

***Запустить app.py***

***Перейти в telegram***

***Вставить ссылку на бота https://t.me/plagiat\_notion\_2\_bot***

Проект разделен по папкам.

--API

----Github

----hh

----Weather

----Maps

--Bot\_logging

--common

--database

--handlers

Основной файл app.py

В основном файле app.py, через который происходит запуск бота роутеры, отвечающие за хэндлеры каждый импортирует в главный файл. Всеми роутерами руководит диспечер. У бота есть определенный набор команд, с которыми он умеет взаимодействовать с пользователем, эти команды хранятся в папке common файл bot\_cmd\_list.py. При запуске асинхронной функции main() в файле сразу создаются все таблицы базы данной, удаляются все попытки взаимодействия с ботом, когда бот был недоступен (не включен)

Database. Особенности реализации

База данных для хранения информация о задачах пользователя, написанный на orm-моделях, используя самый популярный для этого язык sqlaclhemy

В папке database есть файл models.py. В котором создаётся движок и «делатель асинхронный сессий (доступ к базе данных)»

Стоит отметить, что sqlachemy посторена на декларативном подходе. Удобство в **декларативном** **подходе** в том, что вся метаинформация и дополнительные атрибуты, такие как связи с другими классами, также объявляются встроенными в определение класса

Есть абстрактный класс SqlaclhemyBase, который наследуется от AsyncAttrs и DeclarativeBase

От абстрактного классе наследуется класс User (таблица users), который имеет такие поля как id (номер строчки в таблице) тип int и tg\_id (телеграмм id пользователя) тип BigInteger (т.к, число аккаунтов в телеграмме перевалила за 8 миллиардов)

От абстрактного класса наследуется класс Task (таблица tasks), который имеет такие поля как id (номер строчки в таблице) тип int, и name (название задачи) тип str, owner (владелец задачи, берётся из таблицы users, то есть внешним ключом является users.id)

В этой же директории есть асинхронная функция create\_tables, которая создаёт все таблицы, если они ещё не созданы

В файле requests есть асинхронные функции подключения к базе данных, асинхронность получается из-за того, что из файла models импортируется async\_session.

Асинхронные функции:

Set\_user - проверяет заходил ли пользователь в бота или нет, проверка происходит по telegram id, так как он имеет уникальность

Add\_task\_db – добавляет задачу, если такой же задачи (с таким же текстом и автором) нету в базе данных

Show\_task\_db – выводит все задачи пользователя, функция удобно, тем, что к результату функции можно обращаться и получить необходимую информацию о каждой задаче

Get\_info\_task – получить информацию о конкретной задаче

В папке handlers лежат файлы, отвечающие за хэндлеры, благодаря которым и происходит работа бота

Github\_info.py:

Хэндлер (/get\_github\_info), работает, благодаря асинхронной функции, которая отвечает за команду вывода названий репозиториев пользователя

Хэндлер (/create\_repo), работает, благодаря асинхронной функции, отвечающей за создание репозитория

Greeting.py:

Хэндлер (/старт), работает, благодаря асинхронной функции, отвечающая за команду приветствия, если пользователь уже заходил в бота, то будет сообщение с минимальным описание команд в боте, если не заходил, то будет представлено подробное описание возможностей бота

Help.py

Хэндлер (/help), работает, благодаря асинхронной функции, отвечающий за команду помощи, высылающая подсказку

Hh.py

Хэндлер (/get\_diagram\_about\_job), работает, благодаря асинхронной функции, отвечающая и умеющая отправлять диаграмму пользователю картинкой местоположение определяется по ip компьютера или по заданному городу

Manage\_tasks.py

Хэндлер (/add\_task), работает, благодаря асинхронной функции, add\_task, которая создает задачу и добавляют её в таблицу Task базы данных

Хэндлер (/show\_my\_tasks), работает, благодаря асинхронной функции, show\_my\_tasks, показ задач из бд осуществляется, с помощью встроенный клавиатуры telegram InlineKeyboardBuilder. При клике на кнопку в клавиатуре (кнопка отвечает за конкретную задачу) происходит переход в информацию этой задаче. Она показывается текстом в чёрном блоке

Хэндлер (/get\_image\_place), работает, благодаря асинхронной функции, get\_image\_place, которая создает и отображает фотографию с меткой на карте по заданным координатам места

Хэндлер (/get\_weather\_5\_days), работает, благодаря асинхронной функции, get\_weather\_5\_days, которая выдает погоду (минимальная температура и максимальная, краткое описание погоды) на 5 дней по текущему местоположение или по заданному месту