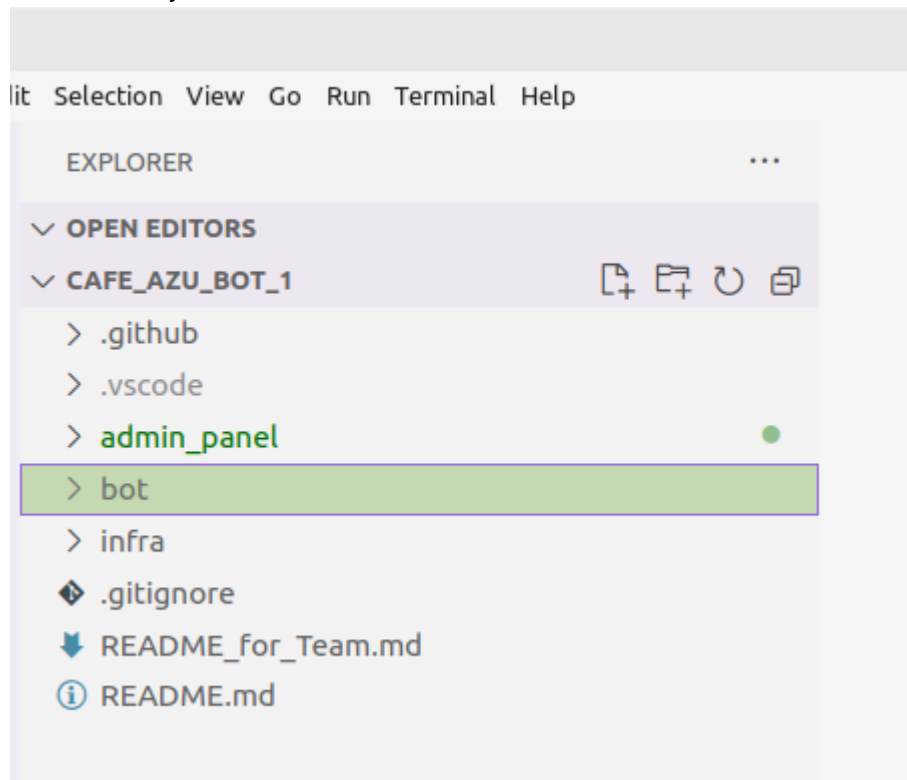


Запуск бота и администраторской панели по отдельности (для отладки)

ВАЖНО: Все дальнейшие действия описываются для VSCode

Открываем <папка, куда клонировали репозиторий>/cafe_azu_bot_1/ как папку в VSCode.
Должно получиться что-то типа такого:



1. Подготовка виртуального окружения

ПРИМЕЧАНИЕ: все дальнейшие пути указываются относительно корневой папки проекта. Т.е. путь вида `/infra/config` означает <папка, куда клонировали репозиторий>/cafe_azu_bot_1/infra/config

- в терминале переходим в папку `/bot`
- если в системе установлено несколько версий python, то выполняем команду (если одна, то проверяем, что бы это была версия 3.10 и до устанавливаем при необходимости)

```
poetry env use <полный путь к исполняемому файлу python версии 3.10>
```

- устанавливаем виртуальное окружение

```
poetry install --no-root
```

- в папке `/bot` создается папка `.venv`

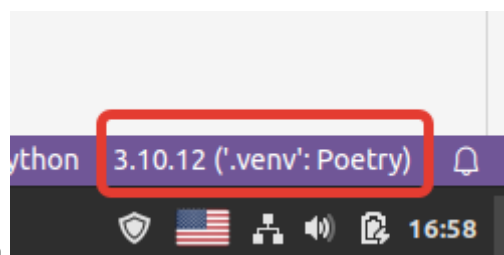
- переходим в папку `/admin_panel` и повторяем те же действия по установке виртуального окружения

2. Настройка виртуального окружения для VSCode

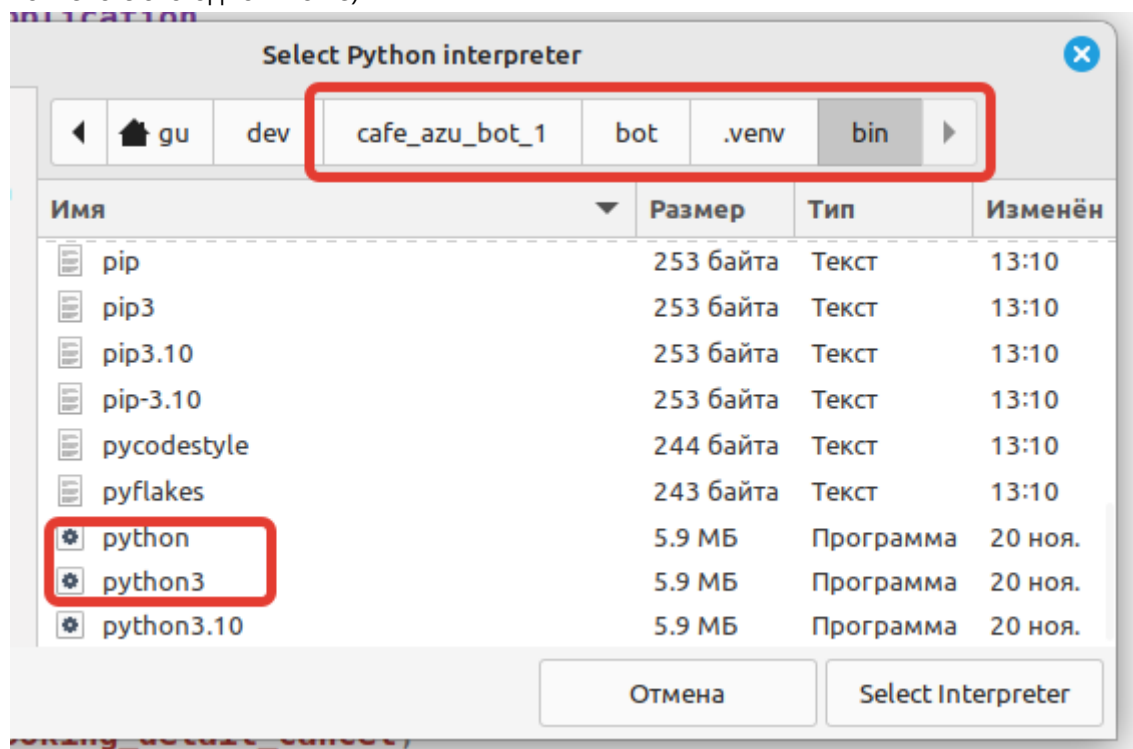
Возможно при первой установке ВО для бота или администраторской панели VSCode это обнаружит и предложит использовать только что созданное окружение

2.1 Настройка ВО для работы с кодом бота

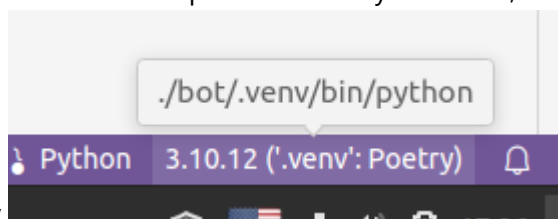
- открываем файл `/bot/main.py`



- в правом нижнем углу щелкаем по версии питона
- в открывшемся окне выбираем `+ Enter interpreter path...`, затем `Find...`
- идем по пути `/bot/.venv/bin` и выбираем файл `python` или `python3` (без разницы, в данном контексте это одно и то же)



- в правом нижнем углу должна появиться версия питона с указанием, что она из виртуального



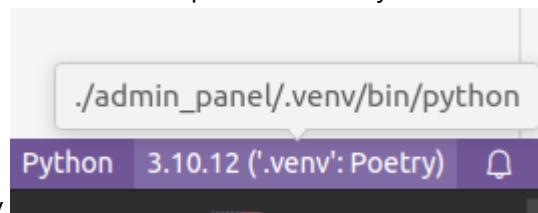
окружения созданного poetry

2.2 Настройка ВО для работы с кодом администраторской панели

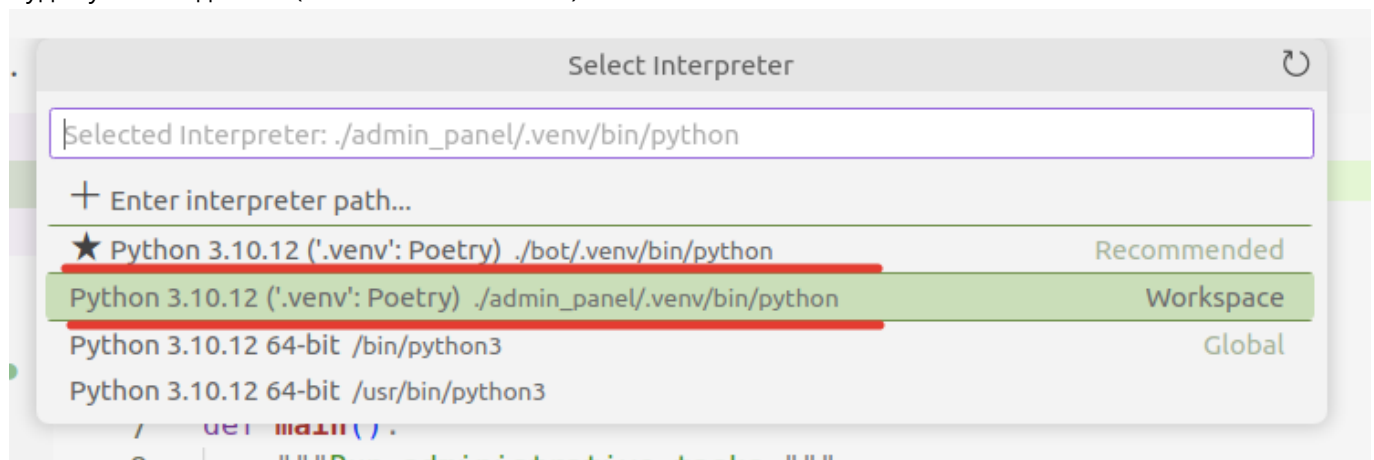
- открываем файл `/admin_panel/manage.py`

- в правом нижнем углу щелкаем на версию питона
- в открывшемся окне выбираем **+ Enter interpreter path...**, затем **Find...**
- идем по пути `/admin_panel/.venv/bin` и выбираем файл `python` или `python3` (без разницы, в данном контексте это одно и то же)
- в правом нижнем углу должна появиться версия питона с указанием, что она из виртуального

окружения созданного poetry



Теперь, если еще раз щелкнуть на версию питона (в правом нижнем углу), то в раскрывающемся списке будет указано два ВО (по мимо глобальных)



- одно для бота (путь начинается с `./bot/...`)
- второе для администраторской панели (путь начинается с `./admin_panel/...`)

3. Подготовка файла для запуска локального кода

- закрываем все открытые файлы
- щелкаем на иконку VSCode **Run and Debug** и в открывшемся окне под кнопкой **Run and Debug** выбираем **create a launch.json file**.
- в открывшемся окне выбираем **Python > Python file** (может появиться сразу **Python file**)
- в открывшееся окно вставляем следующий текст (удалив предварительно все, что там было)

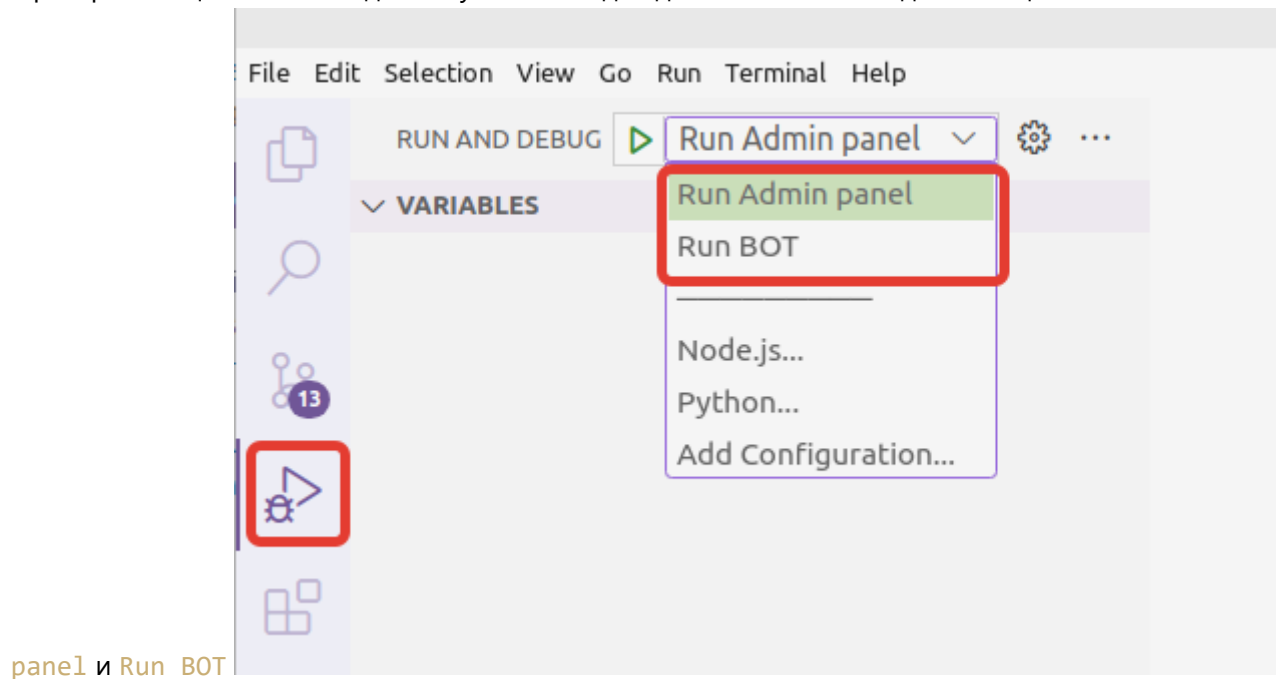
```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Run Admin panel",
      "type": "python",
      "request": "launch",
      "program": "${workspaceFolder}/admin_panel/manage.py",
      "args": [
        "runserver"
      ],
      "django": true,
    }
  ]
}
```

```

    "justMyCode": true,
    "cwd": "${workspaceFolder}/admin_panel"
  },
  {
    "name": "Run BOT",
    "type": "python",
    "request": "launch",
    "program": "${workspaceRoot}/bot/main.py",
    "console": "integratedTerminal",
    "justMyCode": true,
    "cwd": "${workspaceFolder}/bot"
  }
]
}

```

- сохраняем его
- в проекте должна появиться папка `/.vscode` с файлом `launch.json` внутри
- в раскрывающемся списке для запуска и отладки должно появиться две позиции `Run Admin`



ВАЖНО: Перед запуском не забываем проверять, что бы было активно соответствующее ВО (правый нижний угол где версия питона)

4. Запуск локального кода

- запускаем группу контейнеров из папки `/infra`

```
docker compose --file docker-compose_local_dev.yml up --build
```

- убеждаемся, что поднялись все контейнеры (для работы с контейнерами можно установить расширение `Docker` от Microsoft для VSCode)

ВАЖНО: Запуск локального кода будет работать с локальными версиями медиафайлов (папка `media`). Поэтому ее надо создать в (или скопировать и в `/bot` и в `/admin_panel`).

Если какой-то компонент работает в контейнере, то что бы локальные медиафайлы туда попали надо их туда скопировать (в контейнера путь `/app/media`) (ну или остановить группу контейнеров, пересобрать их, запустить и остановить нужный 😊).

4.1 Запуск локального кода для бота

- останавливаем контейнер `azubot-infra-telegram_bot-1` (**только его**, а не всю группу)
- открываем файл `/bot/main.py`
- переключаемся на ВО для бота
 - щелкаем на версию питона в правом нижнем углу
 - в появившемся окне выбираем соответствующее ВО
- открываем вкладку VSCode `Run and Debug`, в раскрывающемся списке для запуска выбираем `Run BOT`
- жмем зеленую стрелу перед раскрывающимся списком
- должно появиться окно терминала с примерно следующим содержанием

```

(gu@guvb) - [~/dev/cafe_azubot_1/bot]
$ source /home/gu/dev/cafe_azubot_1/bot/.venv/bin/activate

(.venv) (gu@guvb) - [~/dev/cafe_azubot_1/bot]
$ /usr/bin/env /home/gu/dev/cafe_azubot_1/bot/.venv/bin/python /home/gu/.vscode/extensions/ms-python.python-2023.22.0/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 37515 -- /home/gu/dev/cafe_azubot_1/bot/main.py
/home/gu/dev/cafe_azubot_1/bot/main.py:95: PTBUserWarning: If 'per_message=False', 'CallbackQueryHandler' will not be tracked for every message. Read this FAQ entry to learn more about the per_* settings: https://github.com/python-telegram-bot/python-telegram-bot/wiki/Frequently-Asked-Questions#what-do-the-per_*-settings-in-conversationhandler-do.
  conv_handler = ConversationHandler(
2023-12-10 17:37:12,715 - apscheduler.scheduler - INFO - Scheduler started
2023-12-10 17:37:12,716 - telegram.ext.Application - INFO - Application started

```

- теперь можно устанавливать точки останова и отлаживать программу

4.1 Запуск локального кода для администраторской панели

- останавливаем контейнер `azubot-infra-django-1` (**только его**, а не всю группу)
- открываем файл `/admin_panel/manage.py`
- переключаемся на ВО для администраторской панели
 - щелкаем на версию питона в правом нижнем углу
 - в появившемся окне выбираем соответствующее ВО
- открываем вкладку VSCode `Run and Debug`, в раскрывающемся списке для запуска выбираем `Run Admin panel`
- жмем зеленую стрелу перед раскрывающимся списком
- должно появиться окно терминала с примерно следующим содержанием

```

(.venv) (gu@guvb) - [~/dev/cafe_azubot_1/admin_panel]
$ /usr/bin/env /home/gu/dev/cafe_azubot_1/admin_panel/.venv/bin/python /home/gu/.vscode/extensions/ms-python.python-2023.22.0/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 37945 -- /home/gu/dev/cafe_azubot_1/admin_panel/manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
December 10, 2023 - 14:44:31
Django version 4.2.7, using settings 'admin_panel.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

```

- теперь можно устанавливать точки останова и отлаживать программу

Выбрать интерпритатор питона можно побыстрее 😊:

- вызываем командную панель (Ctrl + Shift + P)
- в поле ввода пишем `Select interpreter` и выбираем `Python: Select interpreter`
- далее выбираем нужный интерпритатор