Московский авиационный институт

(Национальный исследовательский университет)

**Лабораторные работы 1 - 10**

**Архитектура программной среды**

**Лаборатория 1.** Интегрированные среды разработки и их базовые отличия.

**Лаборатория 2.** Потоковая диаграмма GitHub и Архитектура среды.

.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Алавиев О.А./

(Подпись)

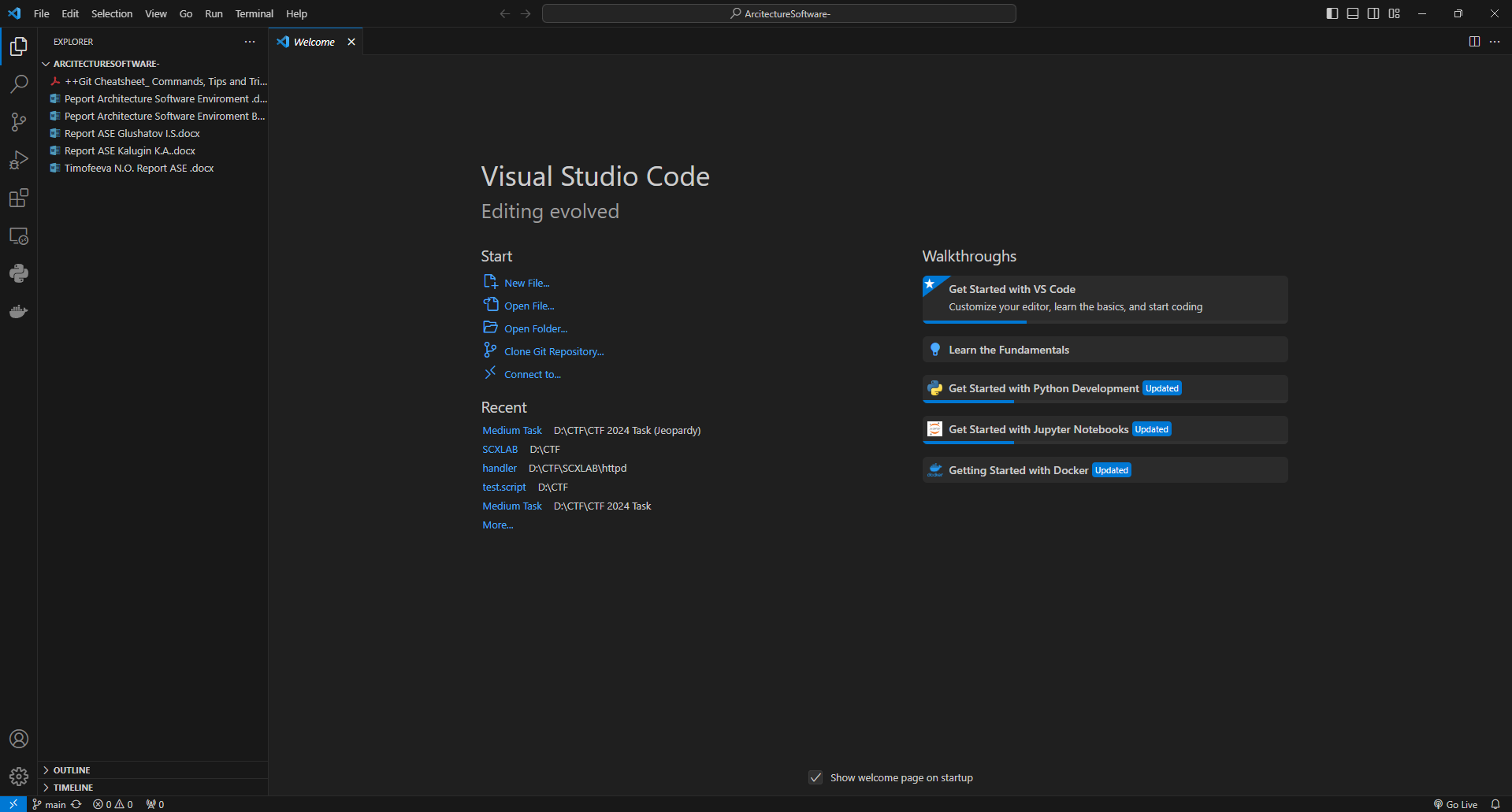
\_\_\_\_\_\_\_\_\_\_\_\_\_/Семенов АС./

(Подпись)

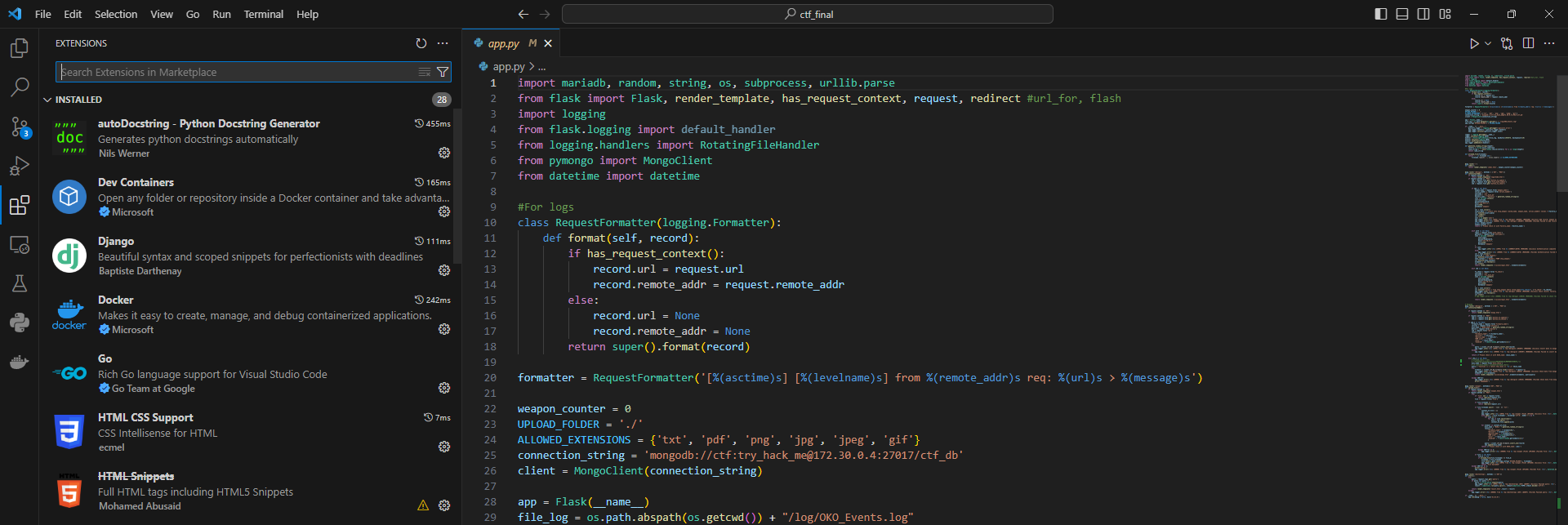
Дата : \_\_ 2024

**Лабораторная 1.**  Интегрированные среды разработки и их базовые отличия

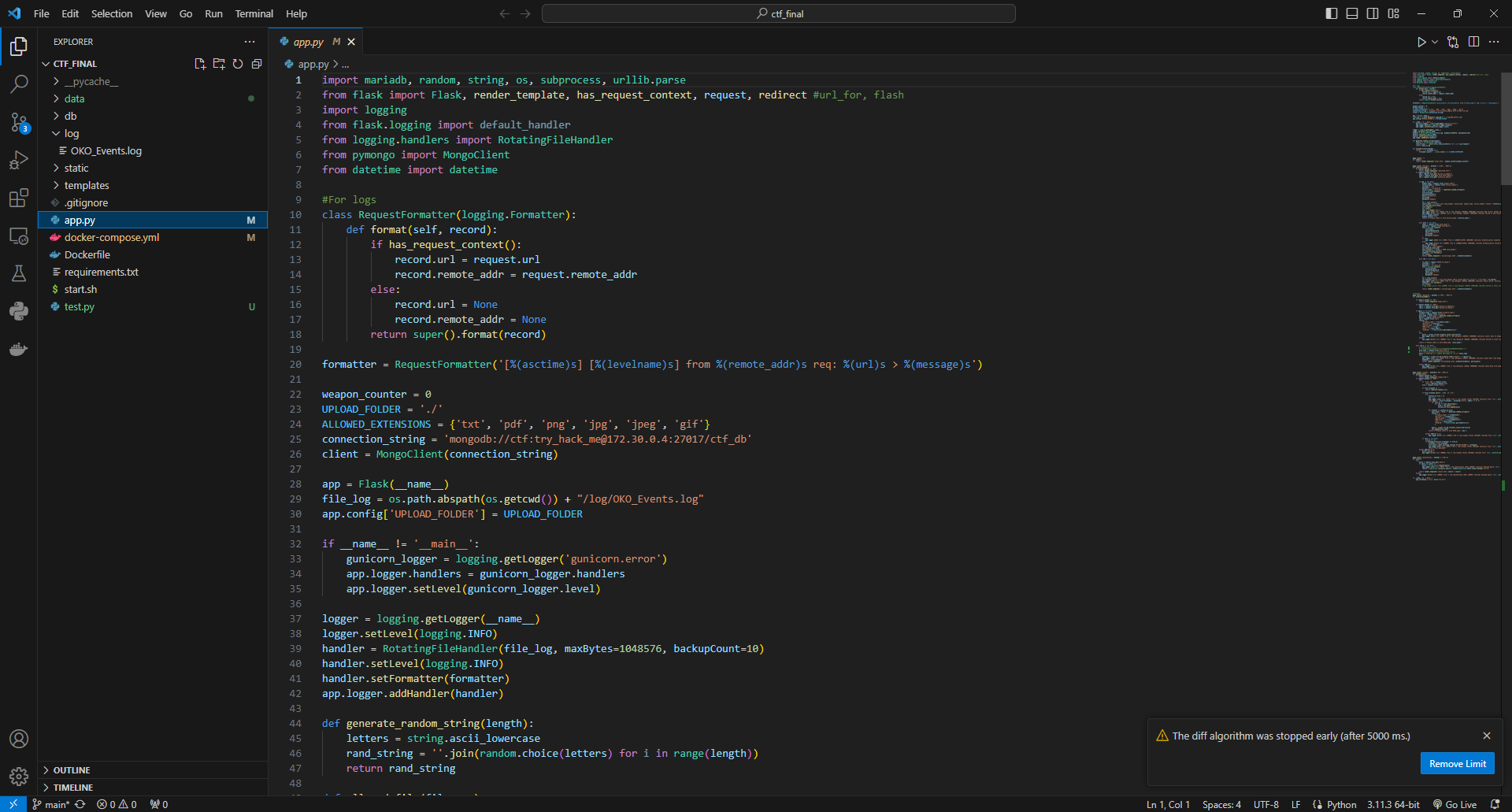
Visual Studio Code – среда разработки, которая построена на основе открытых источников, работает везде и имеет встроенные функции для запуска и отладки кода, а также расширения для установки новых языков, тем, отладчиков и подключения к дополнительным сервисам.

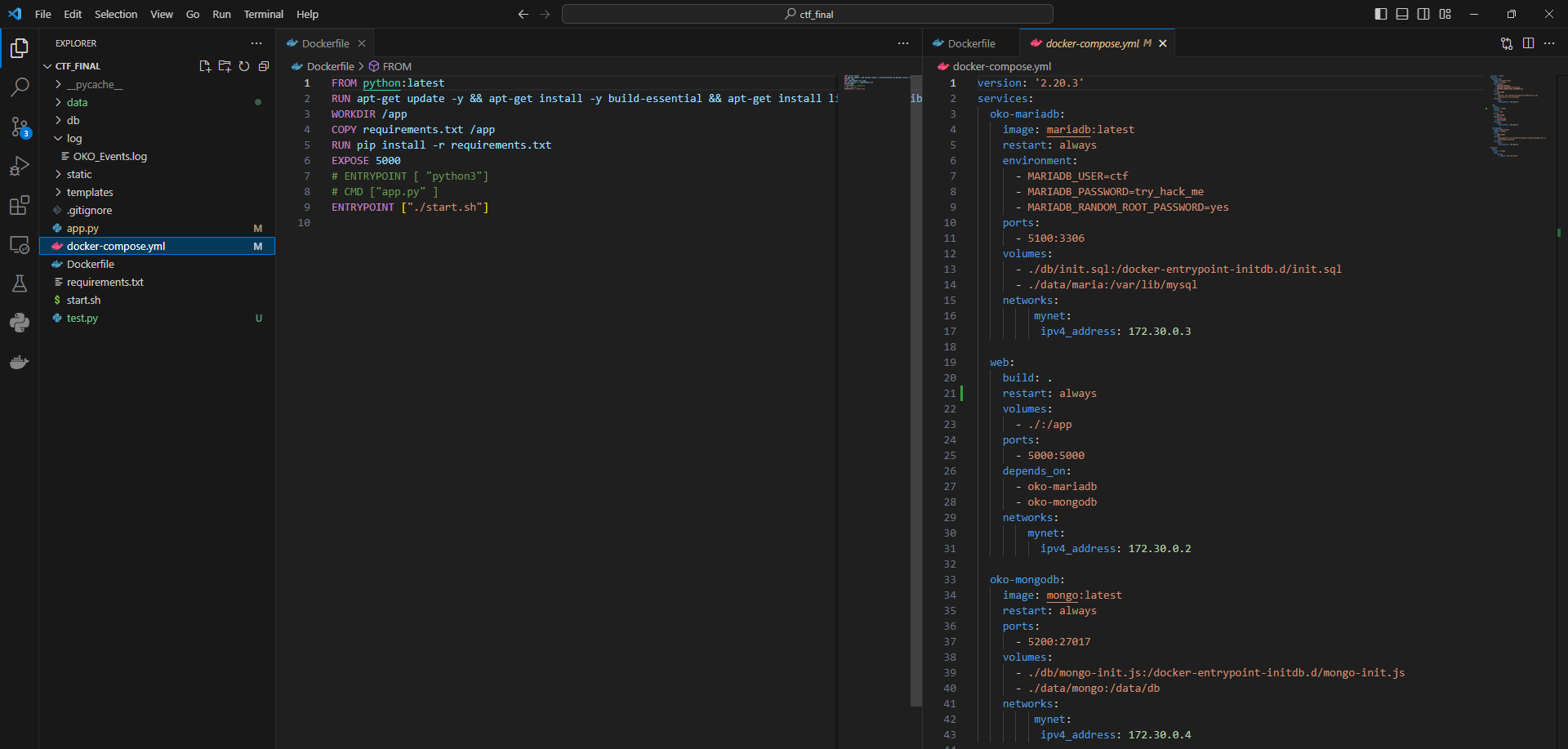


Основная особенность данной IDE заключается в том, что она полностью гибкая и позволяет её полностью настроить под себя и свои задачи. Это можно делать путём установки расширений.



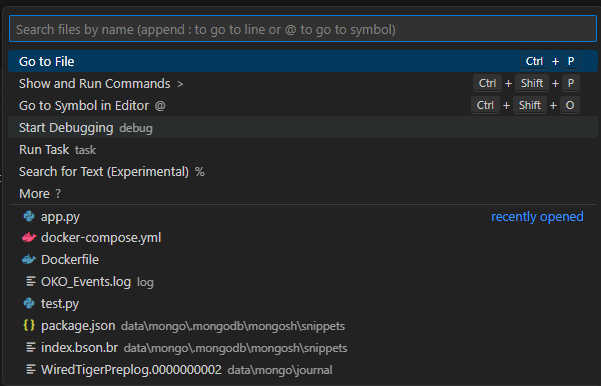
Ниже представлен пример кода на языке программирования Python с учётом предустановленных расширений (Python, Docker).



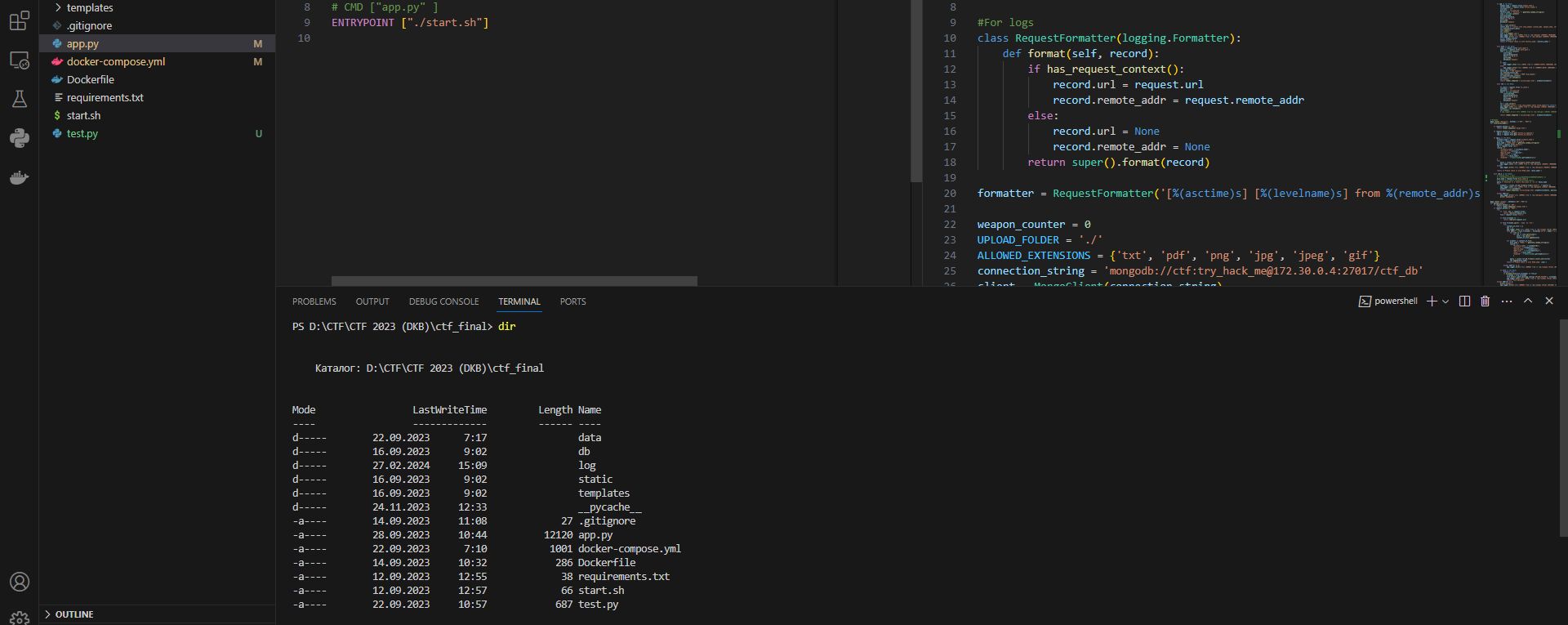
****

Расширения осуществляют подсветку кода, большинство из них поставляются с функционалом помощи разработчику в виде подсказок.

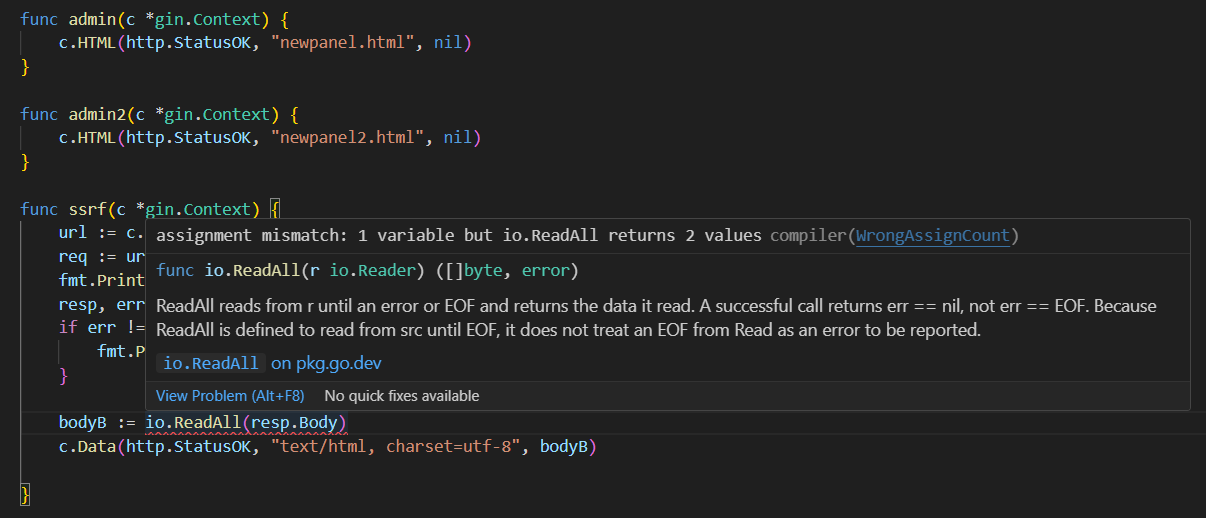
Также в данной среде предусмотрен функционал быстрого поиска и настройки проекта с помощью строки, показанной ниже.



В данной IDE существует встроенный терминал для пользователей, привыкших использовать данный инструмент в Linux или имеют хорошие навыки в использовании Power Shell. На рисунке ниже представлен результат работы команды **dir**, которая выводит список файлов в рабочей директории.

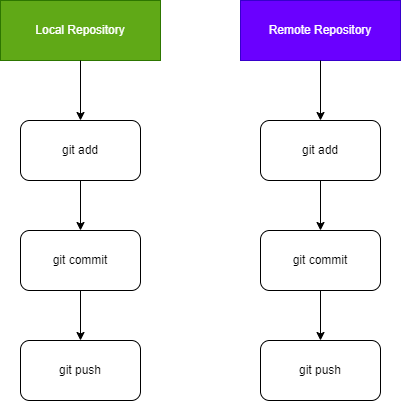


Далее было установлено расширение для языка Go, и будет показан пример работа режима отладки в режиме реального времени.

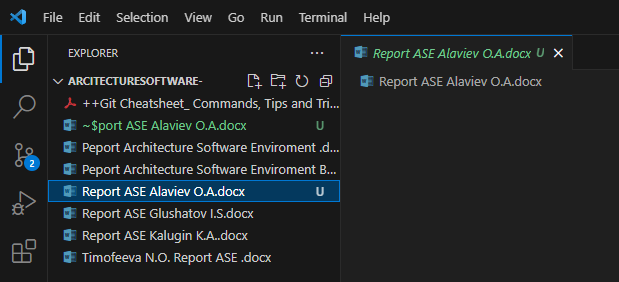


**Лабораторная 2.** Потоковая диаграмма GitHub и Архитектура среды.

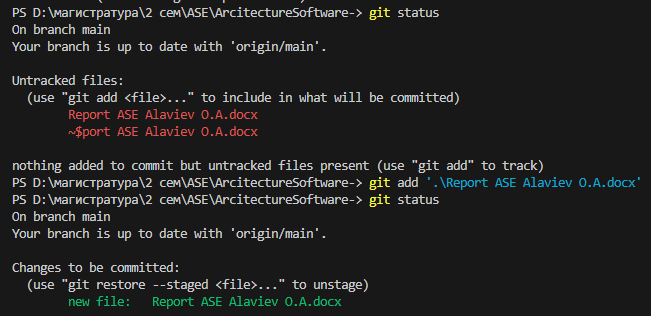
Ниже представлена общая потоковая диаграмма при работе над проектом как в локальном, так и в удаленном репозитории.



Ниже будет показан пример прохождения вышеуказанных этапов. Сначала был склонирован удаленный репозиторий. Создали текущий документ и внесли в него изменения.



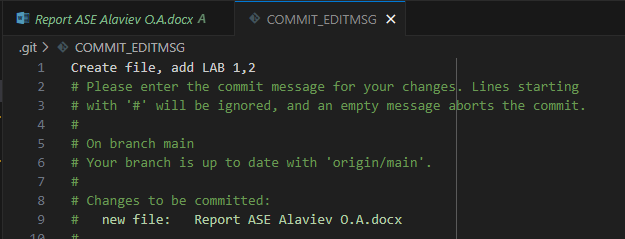
Мы видим, что наш файл находится в состоянии untracked. Сначала необходимо зафиксировать локальное состояние (commit)



Далее вводим команду git commit

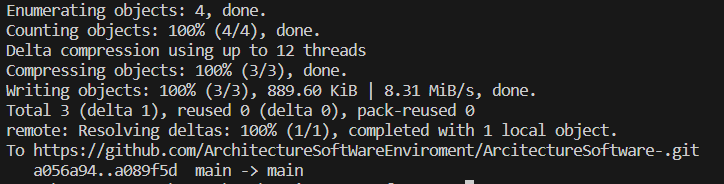


На каждый коммит лучше всего оставлять комментарии, чтобы в дальнейшем было понятно на каком этапе разработки были сделаны изменения.

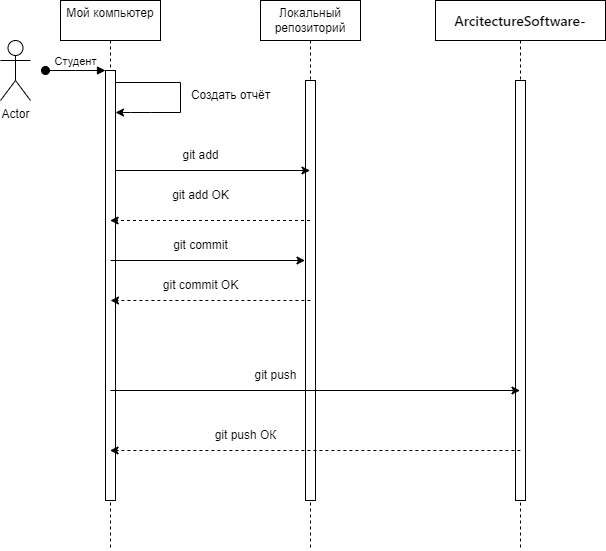


После этого необходимо загрузить данное состояние в удаленный репозиторий с помощью команды git push.

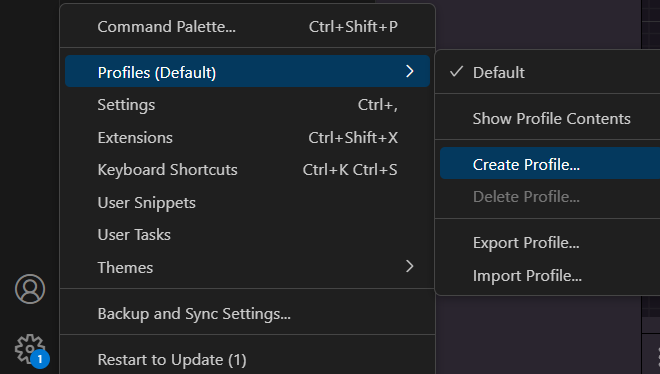




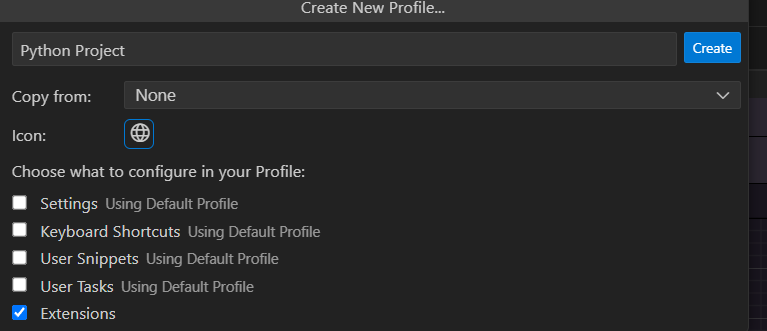
**Лабораторная 3.** Разработка потоковых диаграмм их реализация в среде VS Code



**Лабораторная 4.** Создание профайла VS code для работы в среде Python. Виды профайлов. Их использование при распределенной разработки. Структура профайла. GitHub copilot. UML (Unified Model Language) - диаграмма взаимодействий.



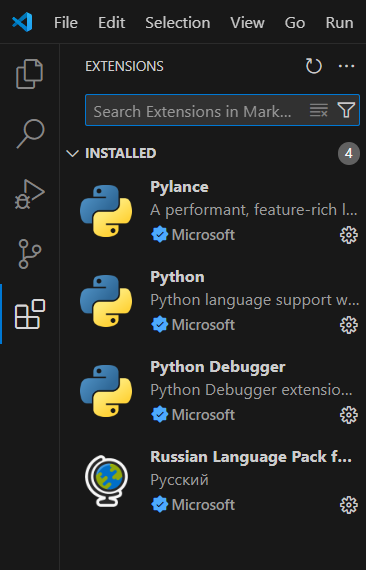
Задаем название для создаваемого профилю и выбираем корректируемые параметры.



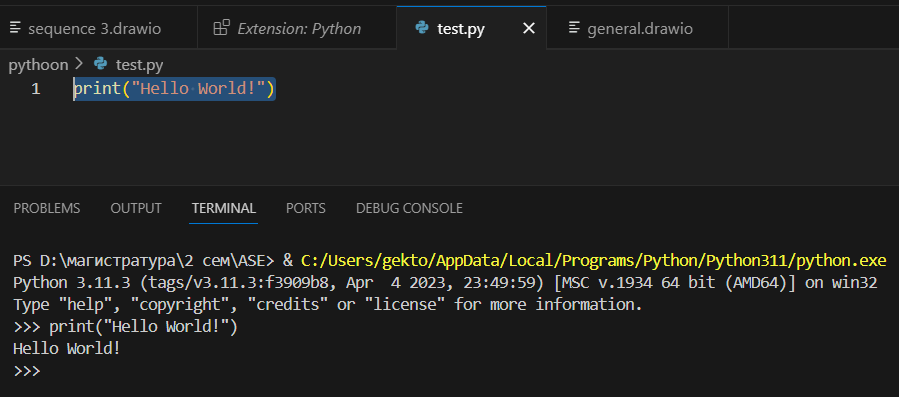
После этого видим, что в созданном профиле есть базовая часть и настройка расширений.



Устанавливаем расширения для работы с Python проектом

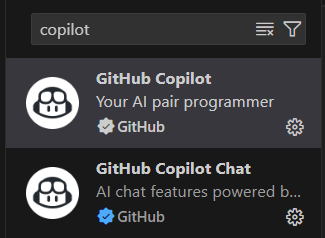


Напишем простую программу

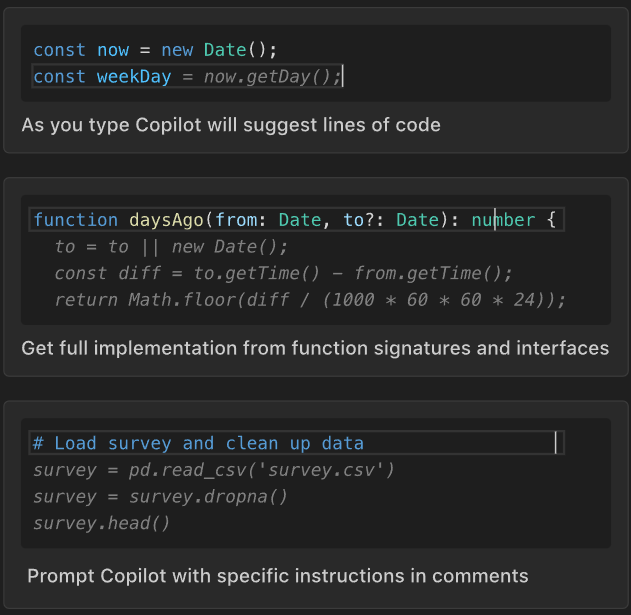


Как видим расширение работает, программа запустилась.Таким образом, можно сделать вывод, что создание профайлов это по сути дела гибкая настройка IDE для конкретного проекта или направления. Т.е можно создать проект для веб разработки на Go, либо профиль для разработки нативных приложений. При переходе на свой профиль разработчик будет чётко понимать, что у него есть всё необходимое для выполнения задачи.

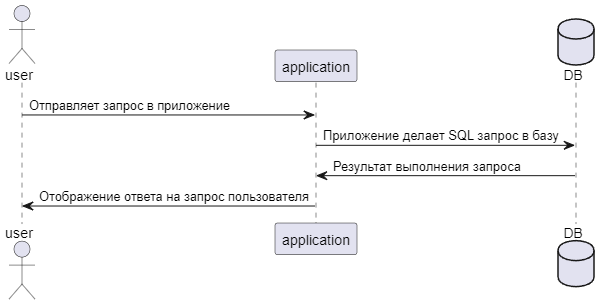
Теперь установим Github Copilot.



GitHub Copilot — ваш искусственный напарник-программист. Система в реальном времени анализирует код, который пишет пользователь, а затем предлагает варианты его продолжения в виде отдельных фрагментов или целых функций. Пример работы GitHub Copilot представлен ниже.



Теперь попробуем нарисовать UML диаграмму простой программы на python.



Исходный код приложения

import sqlite3

# Устанавливаем соединение с базой данных

connection = sqlite3.connect('my\_database.db')

cursor = connection.cursor()

# Создаем таблицу Users

cursor.execute('''

CREATE TABLE IF NOT EXISTS users (

id INTEGER PRIMARY KEY,

username TEXT NOT NULL)

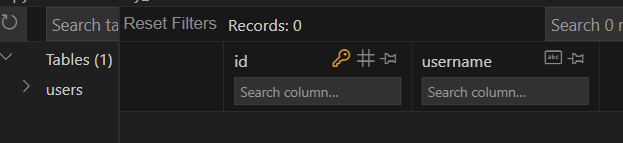
''')

# Сохраняем изменения

connection.commit()

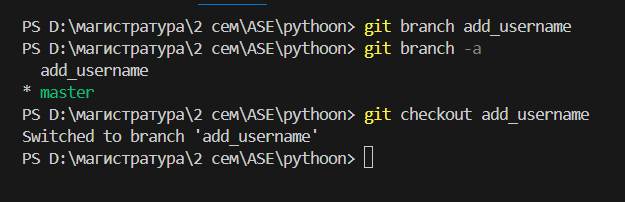
connection.close()

Результат работы



**Лабораторная 5**. Создание ветвей в GitHub для распределенной разработки и их слияние из среды VS Code.

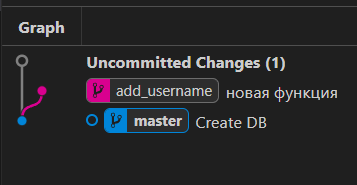
В прошлой лабораторной работе мы написали программу, которая создаем базу данный. Теперь допишем код так, чтобы была возможность добавлять данные в таблицу. Для этого создаем новую ветку и переходим в неё.



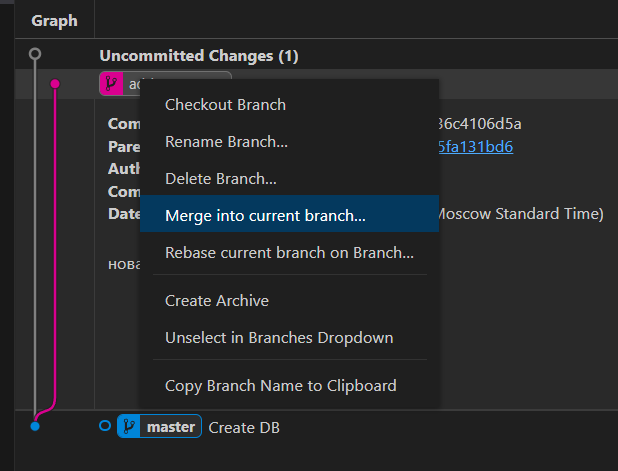
Исходный код в ветке **add\_username**



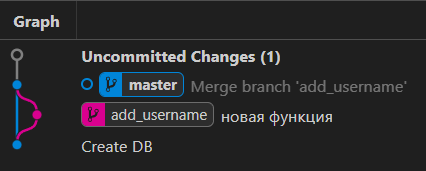
В VS Code увидим изменения после перемещения в мастер ветку



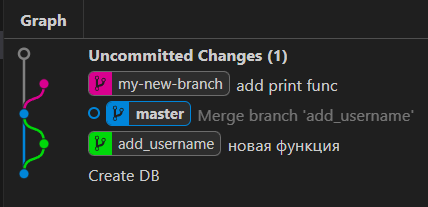
Далее можем выполнить слияние созданной ветки с мастер.



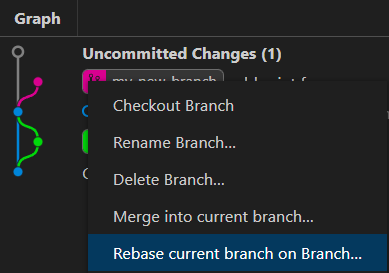
Результат слияния веток



Теперь выполним смещения статуса мастер ветки на новую ветку. Для этого создаем новую ветку и изменяем немного исходный код.



И выполняем функцию rebase branch



Результат выполнения операции rebase

