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.

**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.

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

**Лабораторная 6.**

**Лабораторная 7.**

**Лабораторная 8.**

**Лабораторная 9.**

**Лабораторная 10.**

**\*Laboratory 11.**

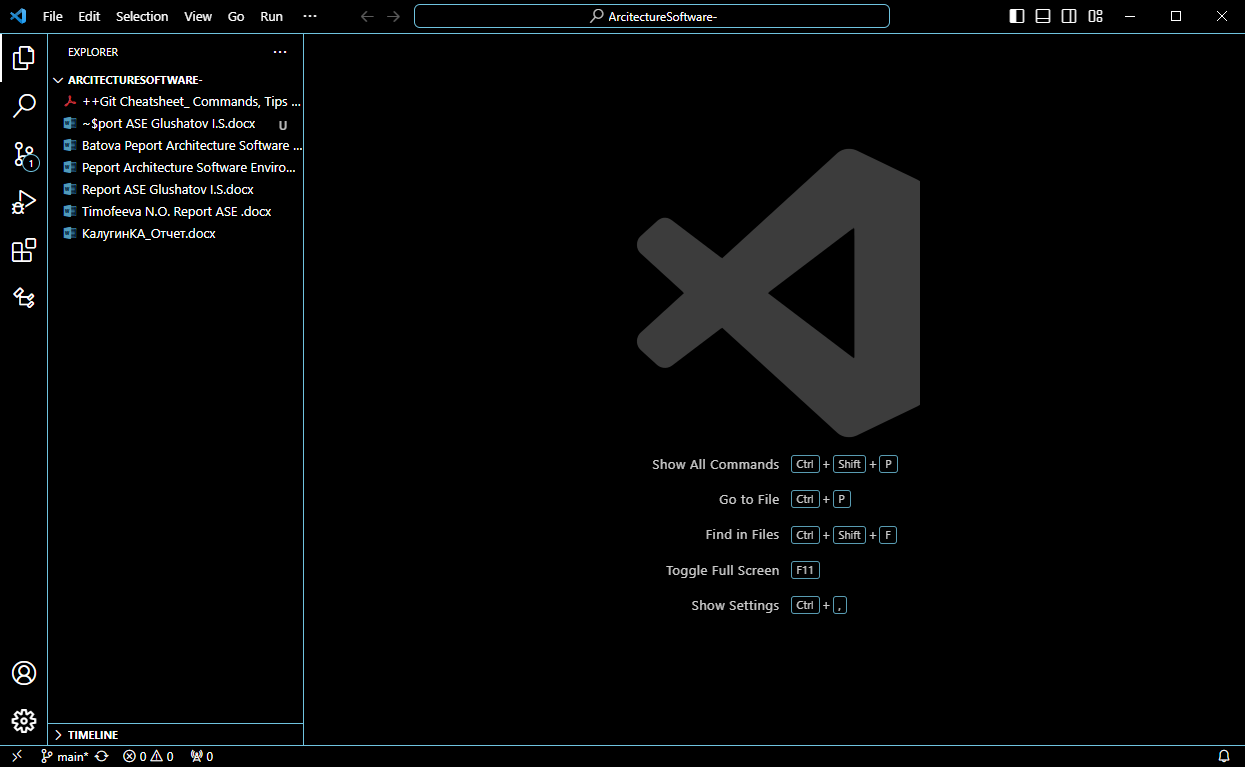
**\*Laboratory 12.**

|  |  |  |
| --- | --- | --- |
| Глушатов И. С. | подпись: | \_\_\_\_\_\_ |
| Семенов А. С. | подпись: | \_\_\_\_\_\_ |
| Дата: | \_\_\_\_\_\_\_ | 2024 |

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

IDE – интегрированная среда разработки, которая используется для редактирования кода, дебага, структурирования проектов, использования дополнительных плагинов для ускорения и удобства разработки.

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.



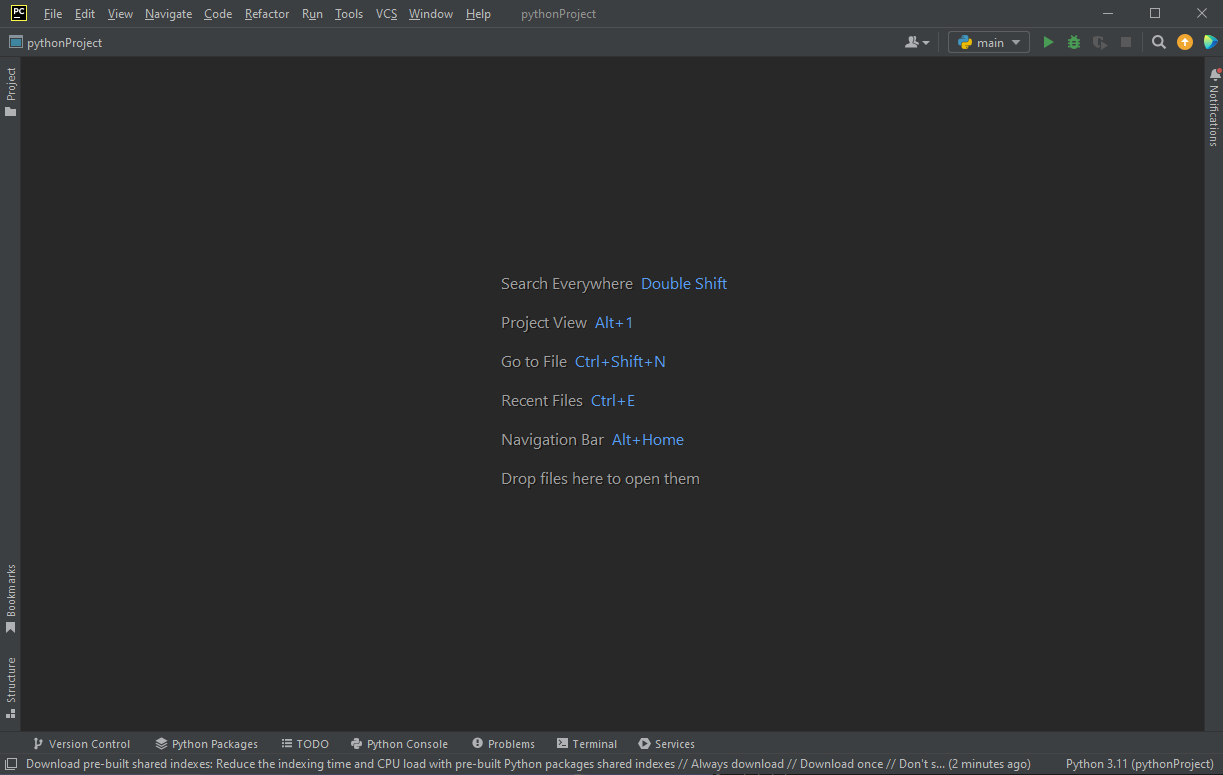
Данная IDE очень гибкая и позволяет полностью настроить под себя и свои задачи. Это можно делать путём установки расширений.

Расширения осуществляют подсветку кода, инструментами для взаимодействия с другими программами и сервисами через интернет.

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

PyCharm – это инструмент разработки для программистов Python. Он позволяет разработчикам писать, тестировать, отлаживать и развертывать код быстрее. Такие функции, как завершение кода, поддержка рефакторинга, проверка кода, контроль версий, управление пакетами и многое другое, позволяют программистам лучше писать код Python за меньшее время.

Его главное меню выглядит так:



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

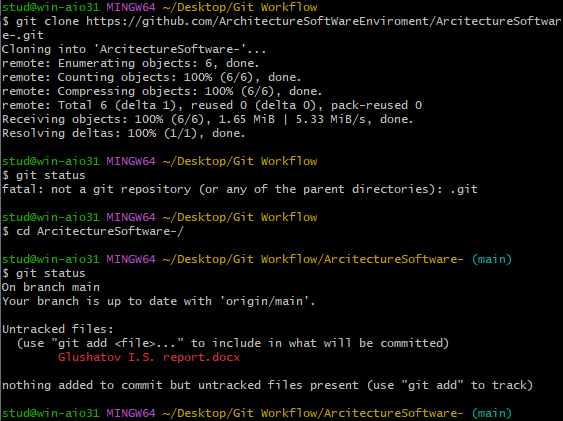
Ниже представлена общая потоковая диаграмма при работе над проектом как в локальном, так и в удаленном репозитории.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

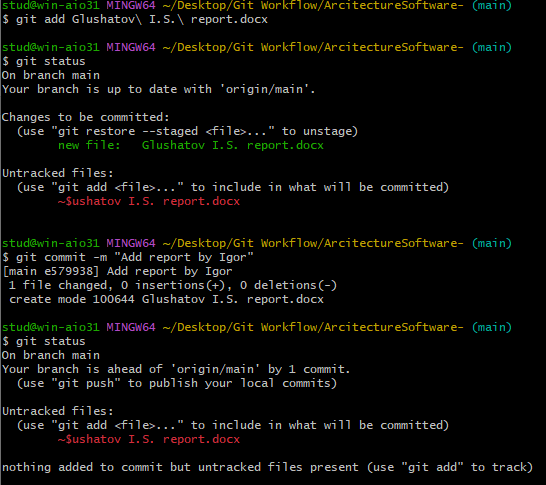
Автоматически созданное описание

Ниже будет показан пример прохождения вышеуказанных этапов.

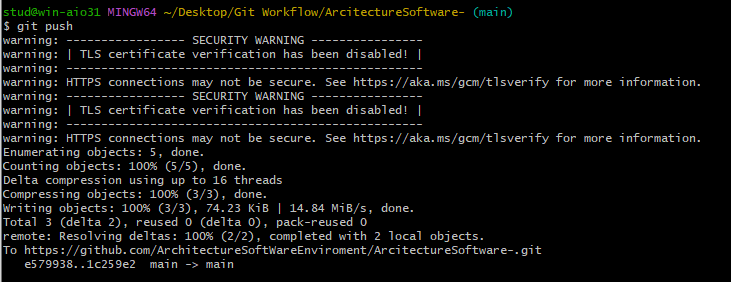
После клонирования репозитория и добавления нового файла:



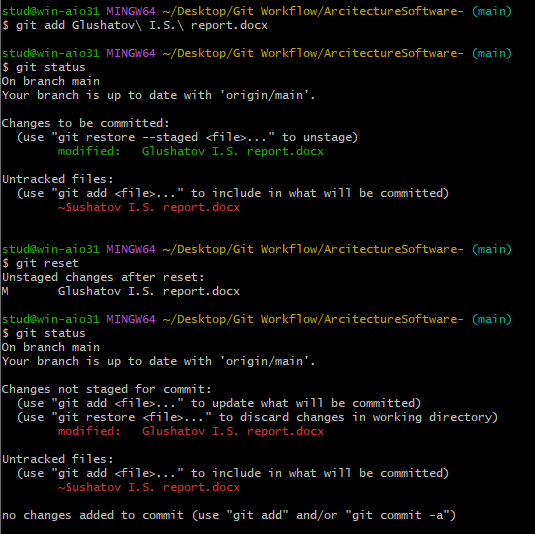
Состояние после команды add и commit:



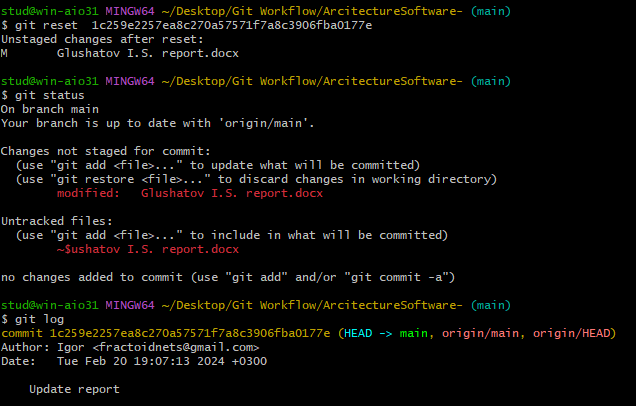
После команды push и отправления изменений на удаленный репозиторий:



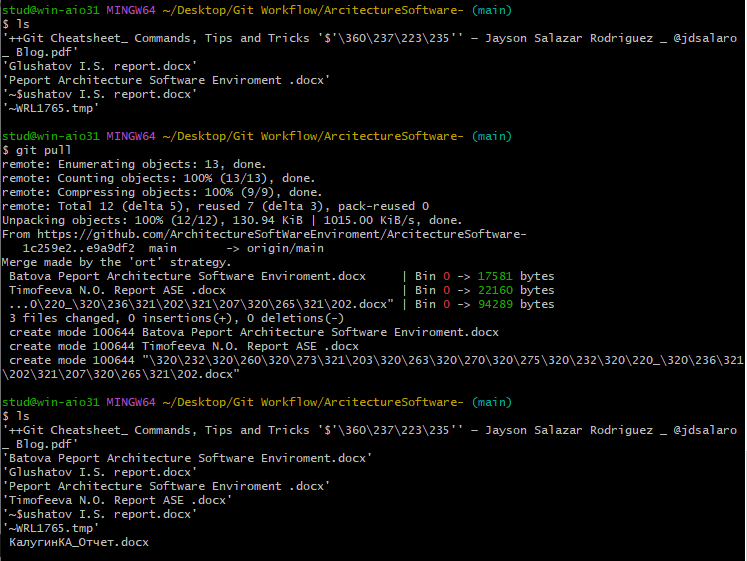
После команды add выполняем команду reset и отменяем добавление в индекс:



После команды git reset [коммит] состояние восстанавливается до указанного коммита:



После изменений на удаленном репозитории делаем команду pull:

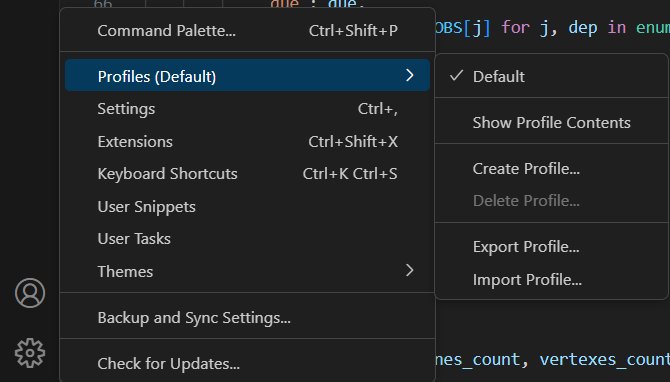


**Лабораторная 3.** Разработка потоковых диаграмм их реализация в среде VS Code

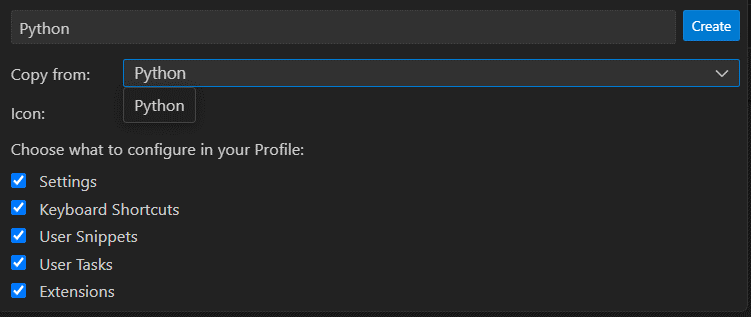
Изображение выглядит как снимок экрана, текст, Параллельный, линия

Автоматически созданное описание

**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.

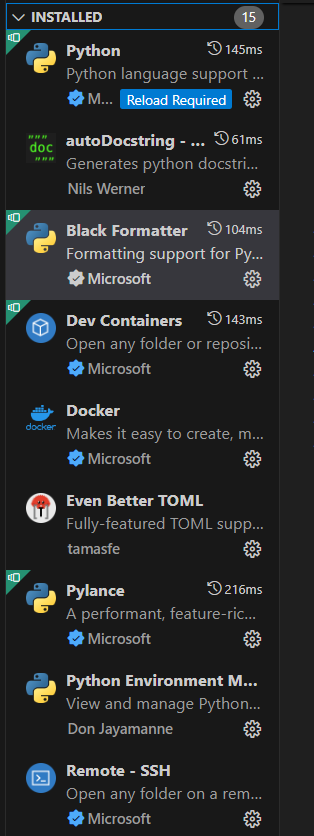


Задаем название для создаваемого профиля и выбираем корректируемые параметры. Надо ввести имя для профайла, всё остальное можно оставить по-умолчанию.



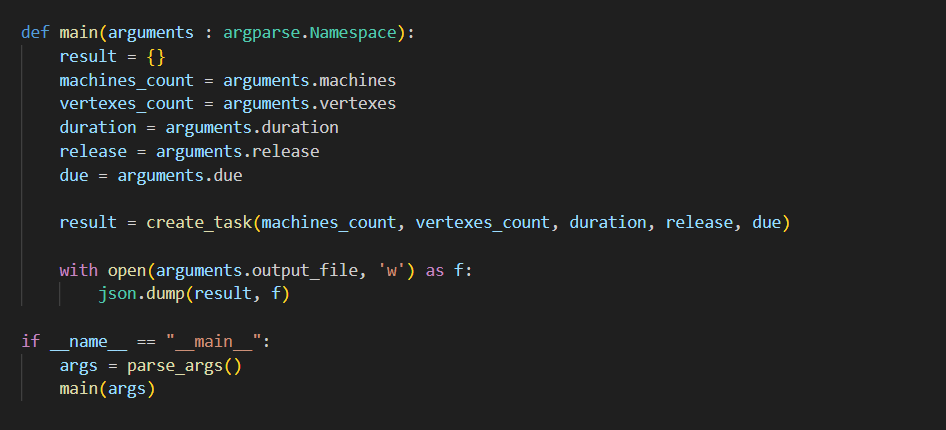
После этого видим, что в созданном профиле есть базовая часть и настройка расширений.

Устанавливаем расширения для работы с Python проектом

 Изображение выглядит как текст, снимок экрана, программное обеспечение

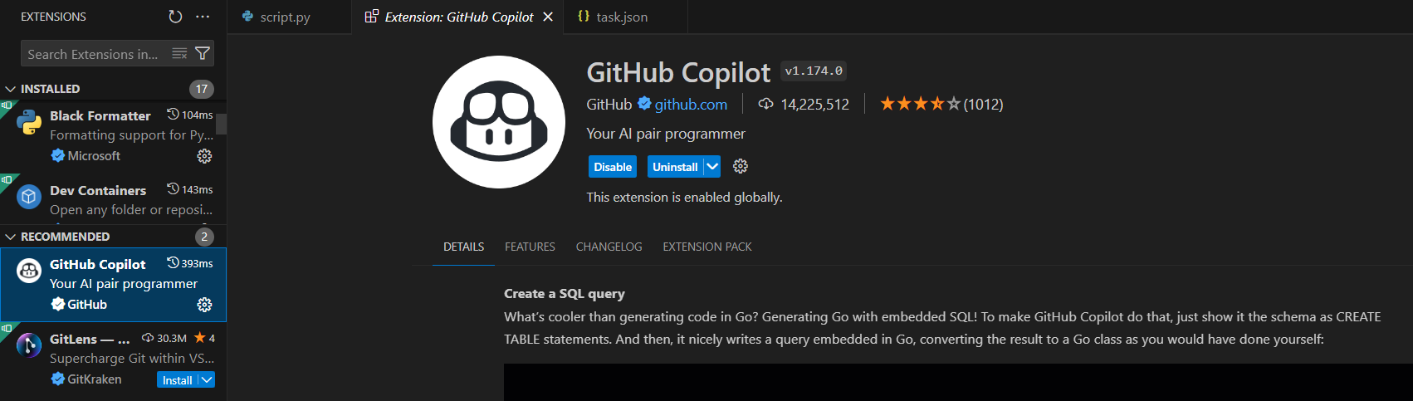
Автоматически созданное описание

На написанной программе можно проверить, что всё работает.



Как видим расширение работает, подсветка синтаксиса есть. 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 C++, либо профиль для фронтенд разработки. При переходе на свой профиль разработчик будет чётко понимать, что у него есть всё необходимое для выполнения задачи.

Теперь установим Github Copilot.

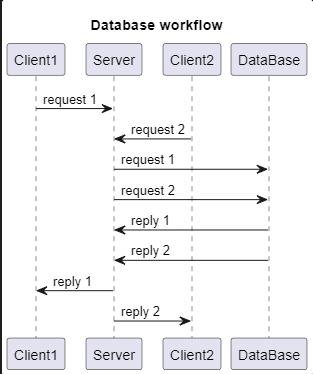


GitHub Copilot — ваш искусственный напарник-программист. Система в реальном времени анализирует код, который пишет пользователь, а затем предлагает варианты его продолжения в виде отдельных фрагментов или целых функций. Пример работы GitHub Copilot представлен ниже.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Теперь попробуем нарисовать UML диаграмму клиент-серверной архитектуры программы на python с базой данных с помощью плагина PlantUML.



Код диаграммы:

@startuml

title Database workflow

Client1 -> Server : request 1

Client2 -> Server : request 2

Server -> DataBase : request 1

Server -> DataBase : request 2

DataBase -> Server : reply 1

DataBase -> Server : reply 2

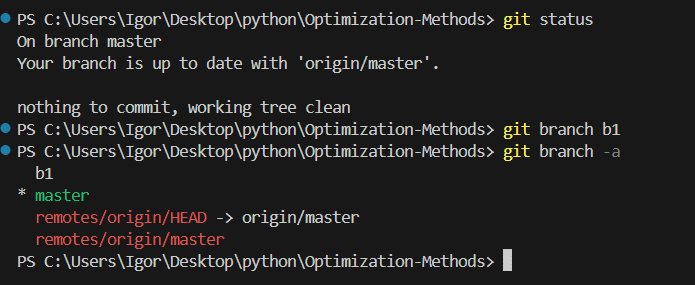
Server -> Client1 : reply 1

Server -> Client2 : reply 2

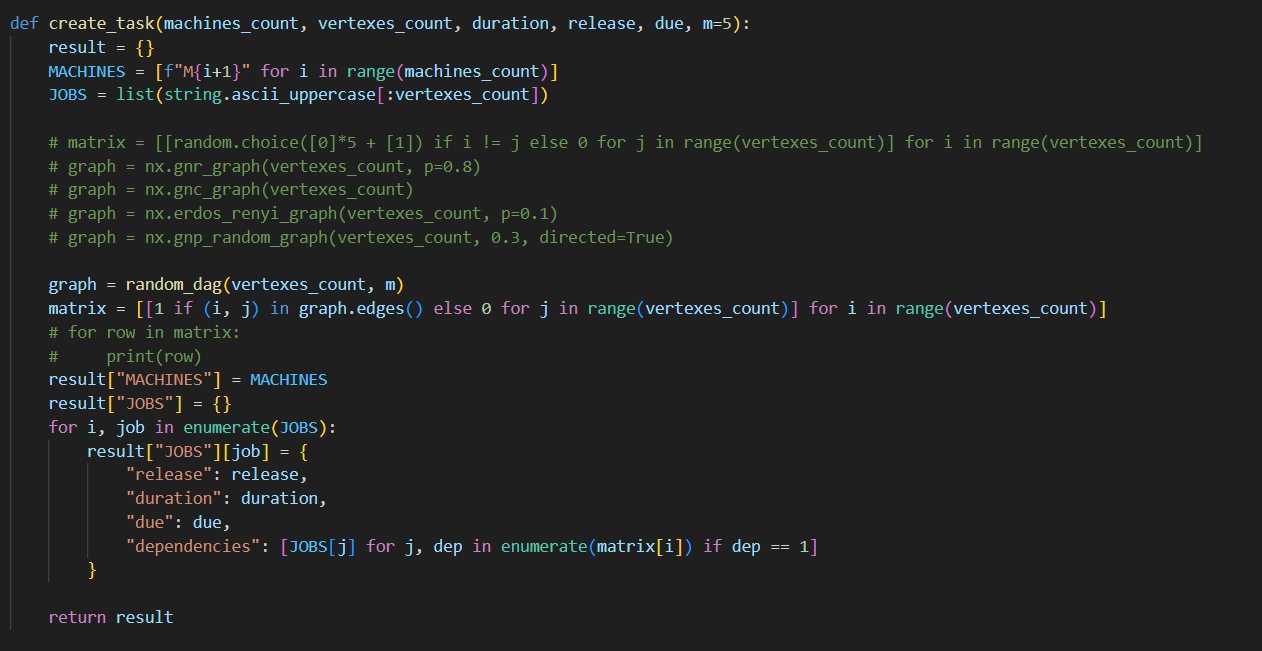
@enduml

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

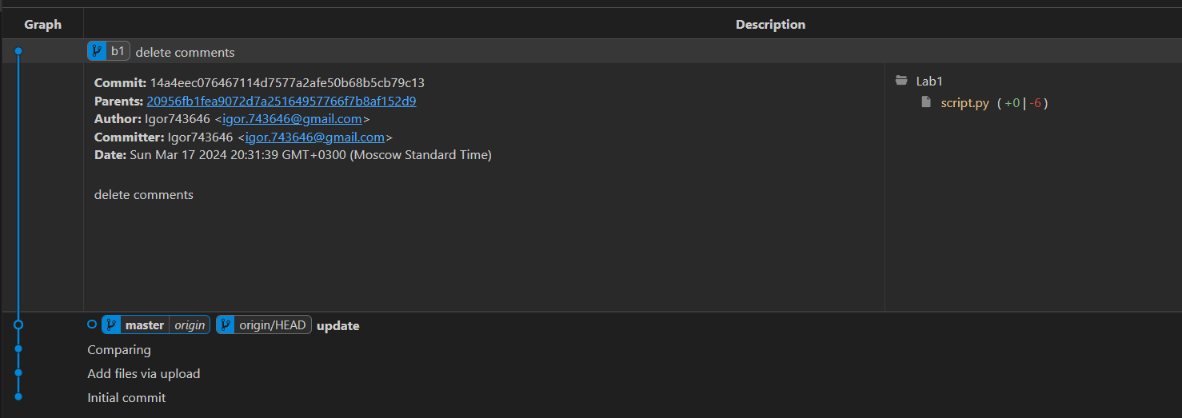
В прошлой лабораторной работе мы написали программу, которая создаем базу данный. Теперь допишем код так, чтобы была возможность добавлять данные в таблицу. Для этого создаем новую ветку и переходим в неё.



Исходный код в ветке **b1** функции **create\_task**:

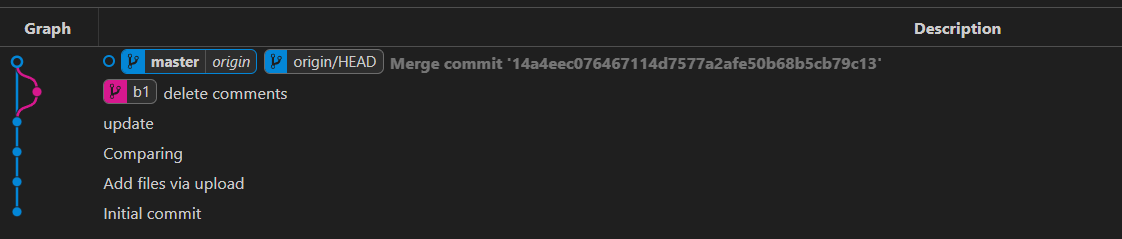


В VS Code увидим изменения после перемещения в мастер ветку

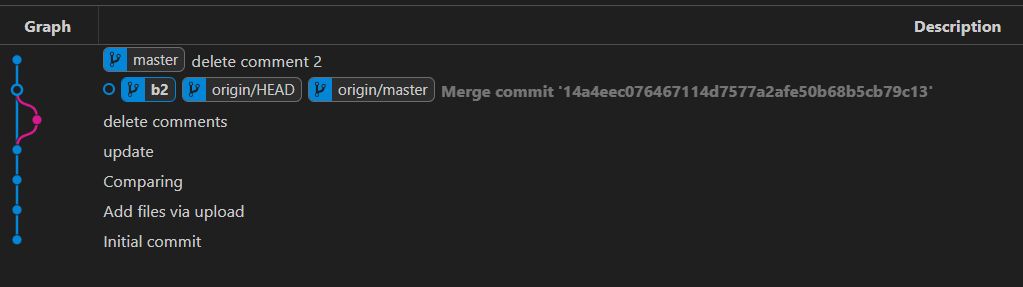


Далее можем выполнить слияние созданной ветки с мастер.

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:

