Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

**дисциплины**

**«Информационные технологии и программирование»**

**Вариант**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Маликов Максим Егорович  1 курс, группа ИНС-б-о-25-1  09.03.02 «Информационные технологии и сети», направленность (профиль) «Прикладное программирование в интеллектуальных информационных системах», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Проверила:  Ассистент департамента цифровых, робототехнических систем и электроники Хацукова А.И.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2025 г.

**Тема:** Основы Git: От Репозитория до первого коммита

**Цель: Научится работать с Git: Разобрать понятия репозитория. Первый коммит.**

**Порядок выполнения работы:**

**Теория**

Представьте, что вы работаете над кодом, рефератом или дизайном игры (Game Design Document). Вы сохраняете первый файл GDD.docx. Затем вносите изменения и сохраняете его как GDD\_v2.docx. Потом появляется новая идея и новый файл — GDD\_FINAL.docx, а затем и GDD\_FINAL2.docx. Вскоре папка превращается в хаос, и невозможно понять, какая версия является последней или в какой из них было внесено ключевое изменение.

**Система контроля версий (Version Control System — VCS)** — это инструмент, который решает эту проблему . Он записывает изменения в файлах, позволяя в любой момент вернуться к любой предыдущей версии проекта. Git — самая популярная распределенная VCS в мире.

**GitHub** — это веб-платформа для хостинга проектов с использованием Git. Это своего рода «социальная сеть для разработчиков», где можно хранить код, совместно работать над ним и делиться им с миром.

**Репозиторий (Repository)** — это хранилище вашего проекта. Он содержит все файлы проекта и всю историю их изменений. Репозиторий может быть локальным (на вашем компьютере) или удаленным (на сервере, например, на GitHub). В нашем случае мы работаем с удаленным репозиторием на GitHub.

**Коммит (Commit)** — это основная единица истории проекта. Коммит представляет собой снимок (snapshot) состояния всех файлов в репозитории на определенный момент времени. Каждый *коммит* имеет:

* **Уникальный идентификатор (хеш)**
* **Автора**
* **Дату**
* **Сообщение (Commit message)** — краткое и понятное описание того, что было изменено и почему. Например: "Добавлен класс главного героя", "Исправлена ошибка с столкновениями", "Обновлен дизайн меню".

**Ветка (Branch)** — это "параллельная вселенная" вашего проекта. По умолчанию всегда существует главная ветка, которая обычно называется main. Она содержит стабильную, рабочую версию проекта.

*Зачем нужны ветки?* Представьте, что вам нужно добавить в игру нового босса. Если вы будете работать прямо в main, то можете случайно сломать рабочую версию. Вместо этого вы создаете новую ветку, например, feature/new-boss. В этой ветке вы можете спокойно экспериментировать, не затрагивая main. Когда работа над боссом будет завершена и протестирована, вы сможете "влить" изменения обратно в главную ветку.

**Слияние (Merge)** — это процесс объединения изменений из одной ветки в другую. Например, когда вы завершили работу над новой функцией в ветке feature/new-weapon, вы сливаете ее с веткой main, чтобы эта функция стала частью основной игры.

**Практическая часть работы:**

**Задание №1. Подготовка**

Зарегистрируйтесь на [**github.com**](https://github.com/) и создайте **репозиторий**, в котором будет ваша биография (Рисунок 1.).

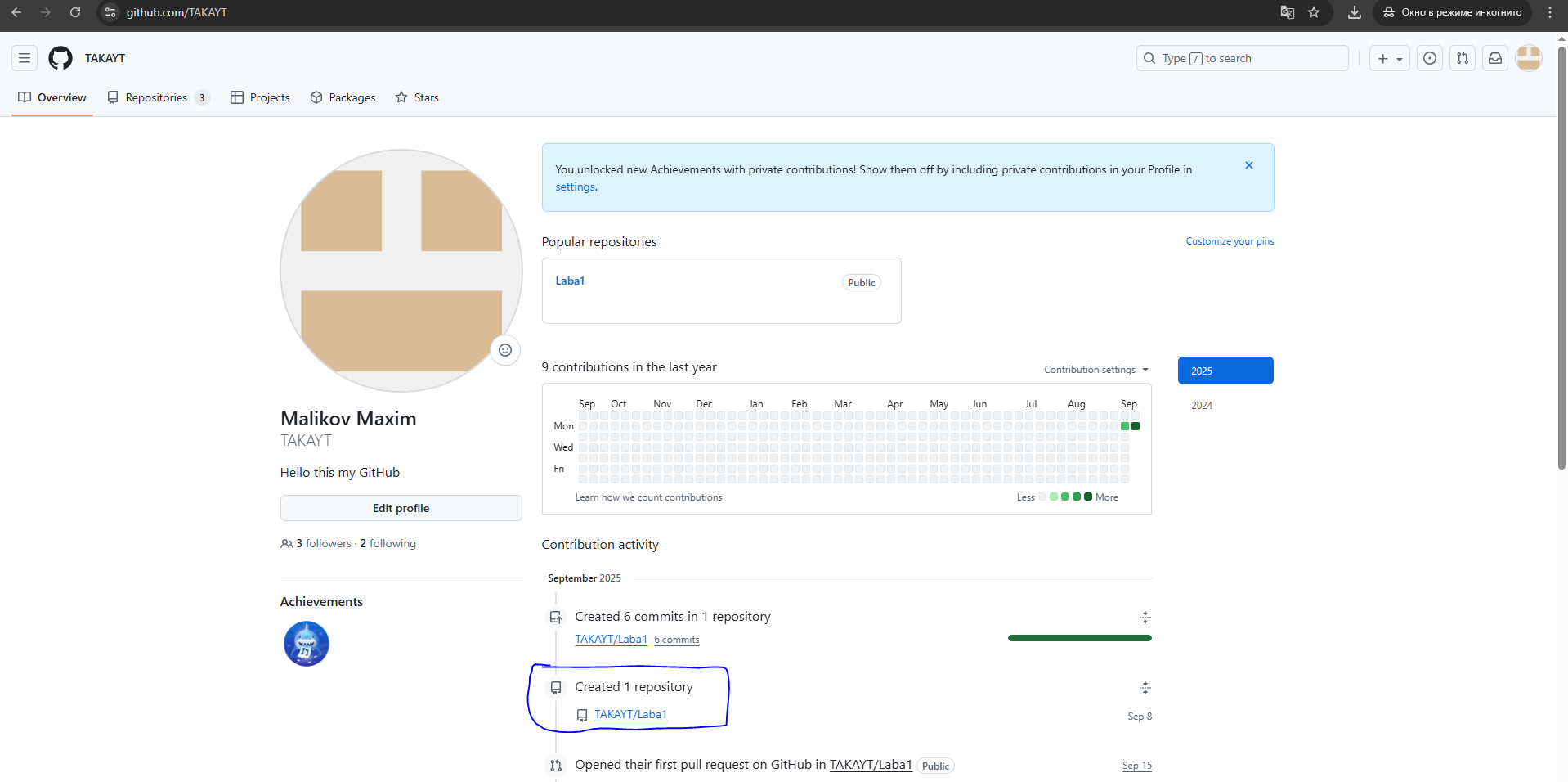


Рисунок 1. Изображение успешного создания профиля, и создания репозитория.

**Задание № 2. Наполнение репозитория.**

В ветке **main** добавьте свои *ФИО* и *группу*.

Создайте две ветки: одну для **образования** (например, *education*) и вторую для **навыков** (например, *skills*). В каждую из созданных веток добавьте соответствующую информацию.

Заполнение репозитория приведено ниже на рисунках 2-4.

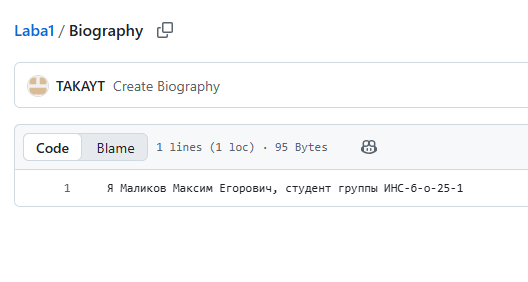


Рисунок 2. Создание файла «[Biography](https://github.com/TAKAYT/Laba1/commit/7f8f2cdb2d0ba62f735f72cc78d921c6812d794a)» main в репозитория и его содержимое.

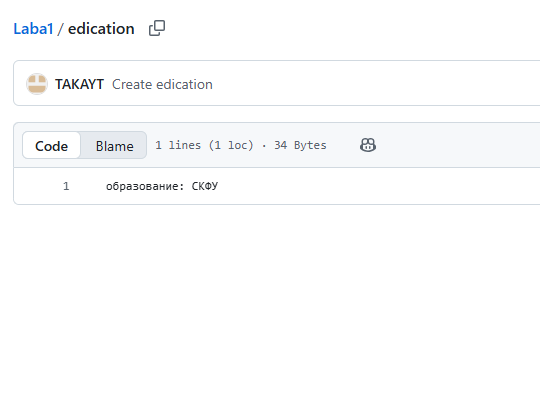


Рисунок № 3 Создание файла «[edication](https://github.com/TAKAYT/Laba1/commit/c1607691e48003aebfa341d23a1f3f0edd9cbac1)» в ветке education и его содержимое файла.

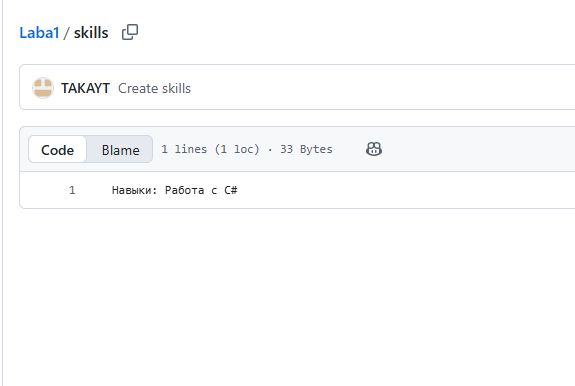


Рисунок № 4 Создание файла «Skills» в ветке skills и его содержимое файла.

**Задание № 3. Update & Merge.**

В каждую из веток добавьте еще немного информации о себе и **смержите** все ветки с **main**. (Рисунок № 5 Commit’s)

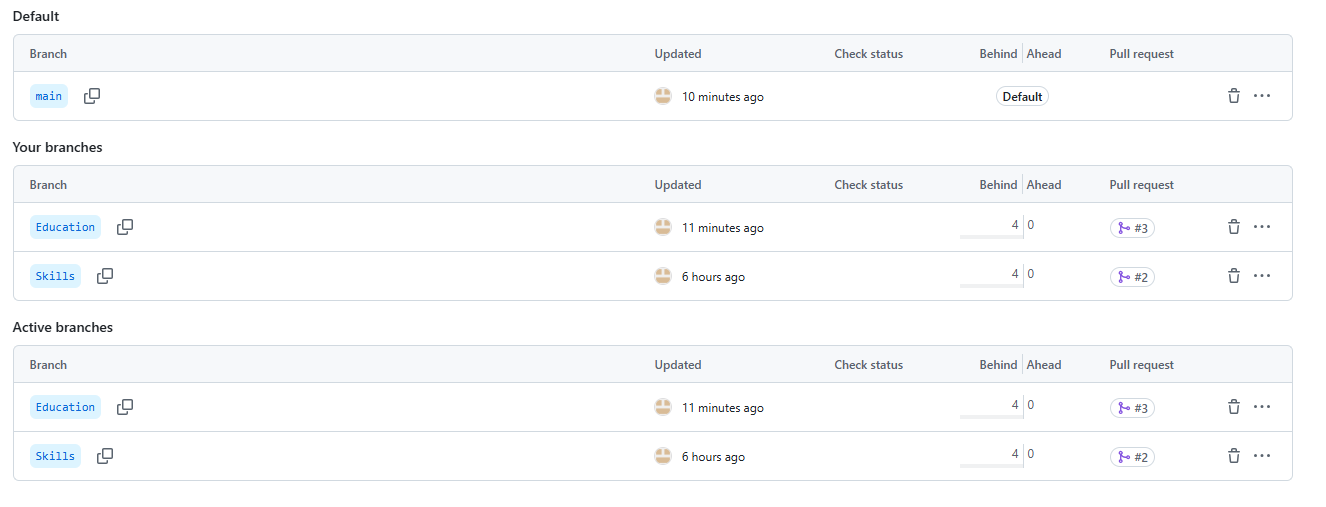


Рисунок № 5 Объединение файлов education & skills в ветку main.

**Ответы на контрольные вопросы:**

**1.**  **Что такое система контроля версий? Для чего она нужна и где может быть использована?**

Ответ: Система контроля версий – это система, позволяющая следить за всеми изменениями проекта, которая позволяет возвращаться к прошлым версиям к примеру, в случае обнаружения бага.

**2.**  **Что такое репозиторий?**

Ответ: Репозиторий – это хранилище данных, которое позволяет распространять файлы по сети.

**3.**  **Что такое коммит (commit)? Почему каждое изменение должно сопровождаться осмысленным сообщением коммита?**

Ответ: Коммит – это система, фиксирующая различные изменения в репозитории, и позволяющая делать откаты к прошлым версиям. Каждое изменение должно сопровождаться осмысленным сообщением коммита чтобы четко понимать различные изменения в прошлых версиях, или, например, при работе в команде чтобы каждый ее член мог посмотреть прошлые изменения файлов.

**4.**  **Объясните разницу между публичным (Public) и приватным (Private) репозиторием. В каком случае какой лучше использовать?**

Ответ: Публичный репозиторий используется для того чтобы распространять проект по сети, любой пользователь мог получить к нему доступ и посмотреть проект. Приватный же используется только для определенного круга лиц. Или например - по ссылке. Вопрос ‘Какой лучше использовать? будет неуместным, так, как и тот и другой служат для разных целей, если имеется цель свободно распространить проект по сети уместным будет использовать Public, если же вы создаете проект для ограниченного круга лиц, нужно использовать Private.

**5.**  **Что такое ветка (branch) в Git? Как вы можете определить, в какой ветке вы сейчас находитесь, глядя на интерфейс GitHub?**

Ответ: Ветка в гит – это система рабочего пространства, с помощью которой можно удобно разрабатывать новый функционал или исправлять ошибки старого, ветки позволяют просматривать изменения и параллельно вести работу над разными задачами.

**Вывод:** Я научился создавать репозитории в GitHub, проводить коммиты и создавать файлы в ветках, а также создавать и объединять ветки.