**Міністерство освіти і науки України**

**Національний університет «Запорізька політехніка»**

кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи № 2

з дисципліни «Soft skills, групова динаміка та комунікації» на тему:

«ГРУПОВА РОБОТА З СИСТЕМАМИ КЕРУВАННЯ ВЕРСІЙ»

Виконала:

студентка групи КНТ-132 М. Г. Кочева

Прийняла:

асистент А. В. Бєлова

2023

**1 ГРУПОВА РОБОТА З СИСТЕМАМИ КЕРУВАННЯ ВЕРСІЙ**

* 1. **Мета роботи**

1.1.1 Вивчити основні можливості систем керування версіями на прикладі системи Subversion.

1.1.2 Вивчити основні можливості системи керування версіями Git та порівняти їх з можливостями Subversion.

1.1.3 Навчитися використовувати можливості систем керування версіями Subversion та Git для групової роботи.

* 1. **Короткі теоретичні відомості**

Система керування версіями дозволяє зберігати декілька версій одного і того ж файла, надає інструментарій переходу до попередніх версій файла, дозволяє відслідковувати зміни, які вносять різні користувачі у відповідні файли. Такі системи найбільш широко використовуються при розробленні програмного забезпечення для зберігання вихідних кодів програми, хоча можуть використовуватися для будь-яких типів файлів.

Репозиторій (сховище) – місце, де зберігаються та підтримуються будь-які дані. Найчастіше дані в репозиторії зберігаються у вигляді файлів. Репозиторії використовуються в системах управління версіями, в них зберігаються всі документи разом з історією їх зміни та іншою службовою інформацією. Репозиторій містить дані у вигляді дерева файлової системи – звичайної ієрархії файлів або тек. Клієнти підключаються до сховища, читають або змінюють ці файли.

Subversion (Apache Subversion) – вільна система управління версіями.

Subversion – централізована система. Дані зберігаються в єдиному сховищі. При збереженні нових версій використовується дельта-компресія: система знаходить відмінності нової версії від попередньої і записує тільки їх, уникаючи непотрібного дублювання даних.

Особливості Subversion:

* Subversion відстежує версії як файлів, так і каталогів;
* якщо зміни зроблені в декількох файлах і каталогах, вони публікуються як одна транзакція;
* при будь-яких оновленнях версій між клієнтом і сервером передаються тільки відмінності між файлами;
* Subversion підтримує копіювання, переміщення і перейменування файлів із збереженням історії змін;
* Subversion однаково ефективно працює як з текстовими, так і з бінарними файлами;
* для зберігання версій використовується ієрархія каталогів – для кожної гілки або мітки створюється окремий каталог.

Існує декілька клієнтів для роботи з системою контролю версій Subversion: TortoiseSVN, кросплатформені клієнти Subversion RapidSVN та SmartSVN, клієнт Subversion, реалізований у вигляді розширення для файлового менеджера в Linux RabbitVCS, розширення для Microsoft Visual Studio VisualSVN.

TortoiseSVN – клієнт для системи контролю версій Subversion, виконаний як розширення оболонки Windows. Робота з системою контролю версій виконується за допомогою контекстного меню.

Subversion не визначає обмежень на файлову структуру проекту. У найпростішому випадку у кореневій директорії сховища рекомендується створювати щонайменше три піддиректорії:

* trunk включає файлову структуру основної лінії розробки проекту;
* branches містить гілки проекту;
* tags містить мітки проекту.

Деякі команди Subversion, що можуть бути використані розробниками:

* svn checkout <URL> – створює робочу копію на основі даних зі сховища;
* svn update – оновлює зміст локальної копії до останньої версії з репозиторію;
* svn commit – відправляє всі зміни локальної копії до репозиторію;
* svn add <файл/папка> – включити файл/папку в локальну копію проекту;
* svn move <файл/папка1> <папка2> – перемістити файл/папку1 до папки2;
* svn copy <файл/папка> <папка> – скопіювати файл/папку1 до папки2;
* svn delete <файл/папка> – вилучити файл/папку з локальної копії проекту;
* svn list <URL> – перегляд каталогу репозиторію;
* svn log <файл> --verbose – історія зміни файлу за ревізіями;
* svn cat --revision <номер\_ревізії> <файл> – відбиття вмісту файлу з даної ревізії;
* svn diff --revision <номер\_ревізії1:номер\_ревізії2> <файл> — відображення змін файлу між двома ревізіями;
* svn status – відображення змін в локальній копії відносно репозиторію.

Гілка – напрямок розробки, який існує незалежно від іншого напрямку, але має з ним загальну історію. Для роботи з гілками використовуються наступні команди:

* svn copy <trunk> <нова гілка> – створює гілку зі вказаною назвою з головної лінії розробки;
* svn merge -r <номер\_ревізії1:номер\_ревізії2> <trunk> – синхронізує гілку з урахуванням ревізій з головною гілкою розробки: <номер\_ревізії1> – номер ревізії, коли гілка була «відкрита», <номер\_ревізії2> – версія головної гілки розробки, з якою виконується злиття;
* svn switch <гілка> – переключитися на зазначену гілку.

Git – розподілена система керування версіями.

Git розглядає дані, що зберігаються, у вигляді відбитків файлової системи. Якщо файл не змінювався, Git не зберігає файл знову, а створює посилання на раніше збережений файл.

У Git файли можуть зберігатися в одному з трьох станів:

* зафіксованому;
* зміненому;
* підготовленому.

Для того щоб розпочати використовувати Git для існуючого проекту, необхідно перейти в проектний каталог та в командному рядку ввести команду git init.

Основні команди роботи з Git включають наступні:

* clone – створити копію існуючого репозиторію Git;
* status – переглянути стан файлів;
* add – додати файл під версійний контроль;
* commit – зафіксувати зміни;
* rm – видалити файл з відслідковуваних;
* log – переглянути історію фіксацій;
* branch – створити (та видалити) гілку або переглянути наявні гілки;
* checkout – дозволяє створити гілку (ключ –b) та перейти на неї;
* merge – злити гілки;
* push – відправити гілку на сервер.

Для системи керування версіями Git існує популярний веб-сайт github.com, який позиціонується як веб-сервіс хостингу проектів з використанням даної системи керування версіями. Користувачі можуть створювати необмежену кількість репозиторіїв. Для open-souce проектів використання даного сервісу безкоштовне.

* 1. **Завдання роботи**

1.3.1 Ознайомитися з теоретичними відомостями, необхідними для виконання роботи.

1.3.2 Ознайомитися з основними можливостями систем керування версіями Subversion та Git.

1.3.3 Використовуючи систему керування версіями Subversion (для пунктів 1.3.3-1.3.15), створити каталог проекту та базову структуру репозиторію.

1.3.4 Створити файли проекту у каталозі. Файли проекту можуть містити код проекту або бути текстовими файлами.

1.3.5 Відредагувати файли проекту і зберегти версії.

1.3.6 Створити нову гілку та переключити робочу копію на неї.

1.3.7 Відредагувати файли проекту.

1.3.8 Переключити робочу копію на піддиректорію trunk.

1.3.9 Злити зміни між гілками.

1.3.10 Вилучити гілку.

1.3.11 Додати до репозиторію проект на основі даних одного з проектів з відкритим вихідним кодом.

1.3.12 В якості такого проекту можна використати, наприклад, Notepad++ (svn://svn.code.sf.net/p/notepad-plus/code/trunk notepad-plus-code) або будь-який власний.

1.3.13 Внести зміни в файли проекту.

1.3.14 Відправити зміни до репозиторію та пояснити отримані результати.

1.3.15 Пункти завдань 1.3.6–1.3.14 виконати за допомогою клієнту та командного рядка.

1.3.16 Використовуючи систему керування версіями Git (для пунктів 1.3.16-1.3.25), створити репозиторій на основі одного з раніше розроблених проектів.

1.3.17 Заборонити автоматичне додання в репозиторій файлів з розширенням \*.exe.

1.3.18 Внести зміни в текст проекту та зафіксувати їх, для підпису використовуючи власні дані.

1.3.19 Переглянути різницю між новою версією проекту та початковою.

1.3.20 Створити власне віддалене сховище, використовуючи сервіс GitHub.

1.3.21 Налаштувати локальне сховище для синхронізації з віддаленим та відправити локальну версію на сервер.

1.3.22 Переглянути історію проекту та сторінку проекту через web-інтерфейс.

1.3.23 Внести зміни в текст проекту та зафіксувати їх.

1.3.24 Узгодити локальну версію репозиторію з сервером.

1.3.25 Пункти завдань 1.3.17–1.3.25 виконати за допомогою графічної оболонки та командного рядка.

* + 1. Оформити звіт з роботи.
    2. Відповісти на контрольні питання.
  1. **Копії екранних форм з результатами виконання завдання та тексти файлів у декількох ревізіях (Subversion).**
     1. Cтворюємо папку, додаємо папку trunk та створюємо у ній файл проекту (рис. 1.1).

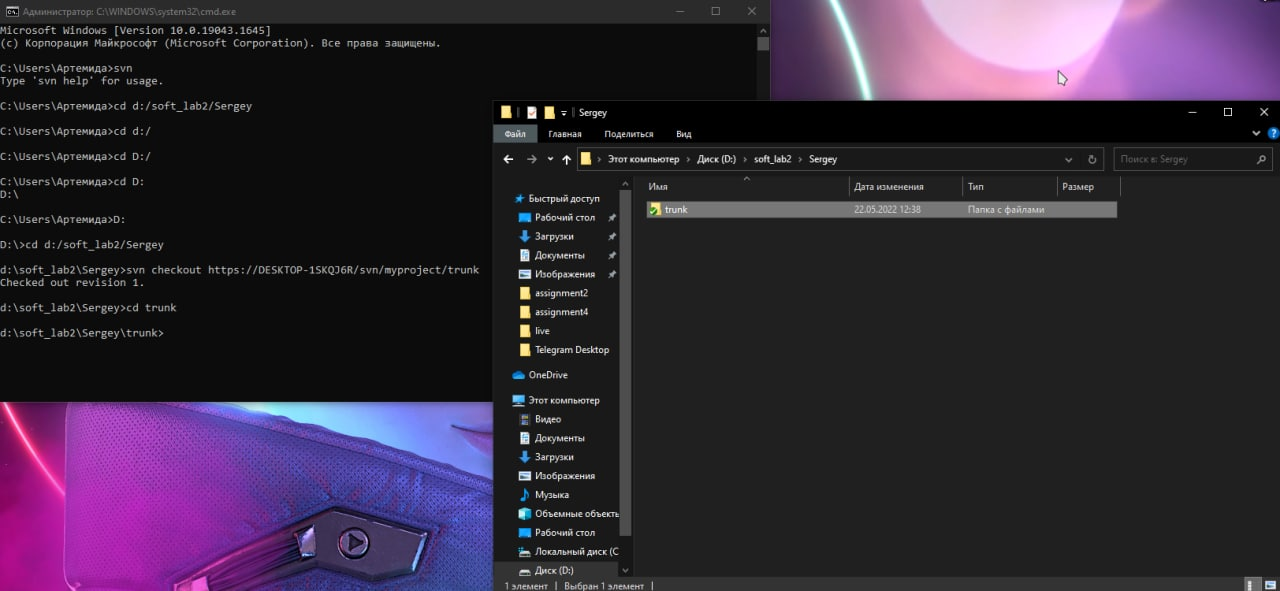


Рисунок 1.1

* + 1. Виконуємо команду додавання нового файлу (рис. 1.2).

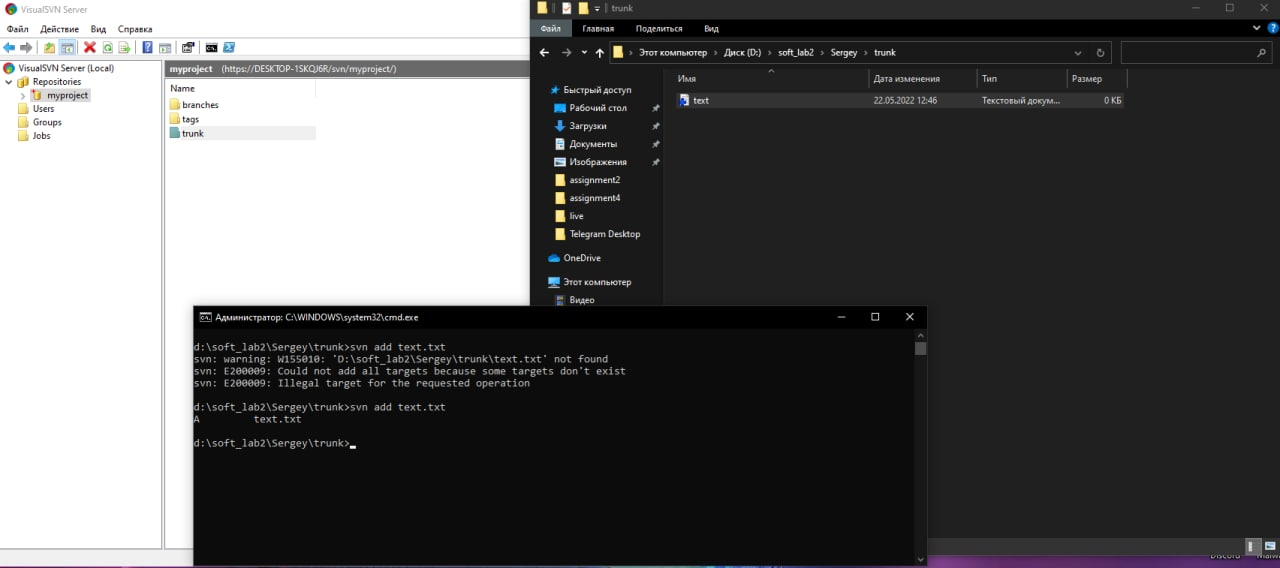


Рисунок 1.2

* + 1. Вносимо зміни до файлу і робимо commit (рис. 1.3).

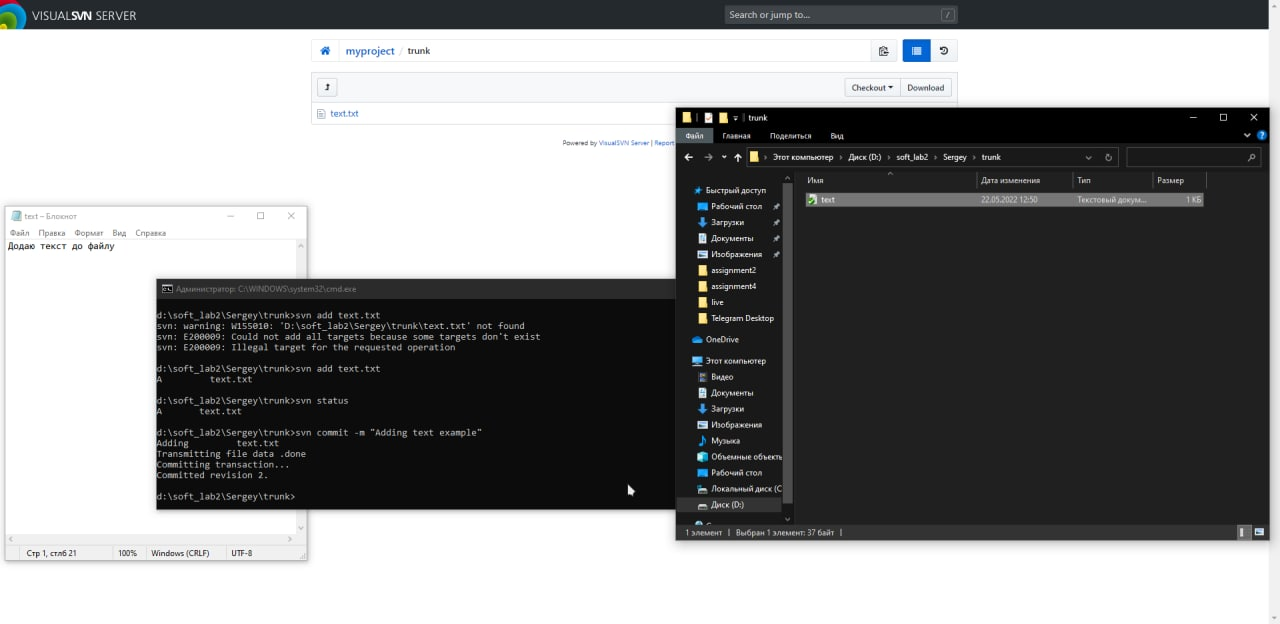


Рисунок 1.3

* + 1. Створюємо нову папку у проекті та завантажуємо ствол проекту (рис. 1.4).

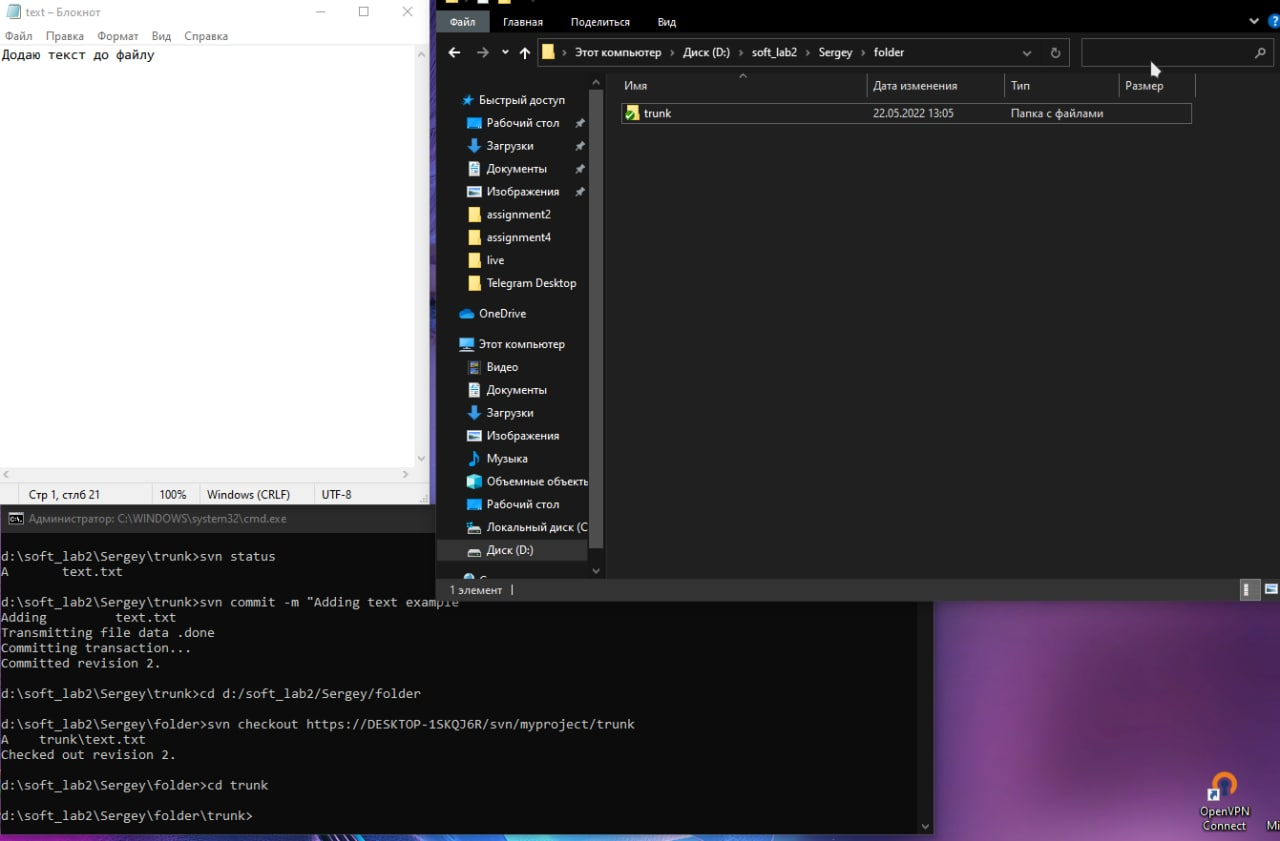


Рисунок 1.4

* + 1. Заходимо до першої папки та робимо оновлення файлу та відкриваємо його (рис. 1.5).

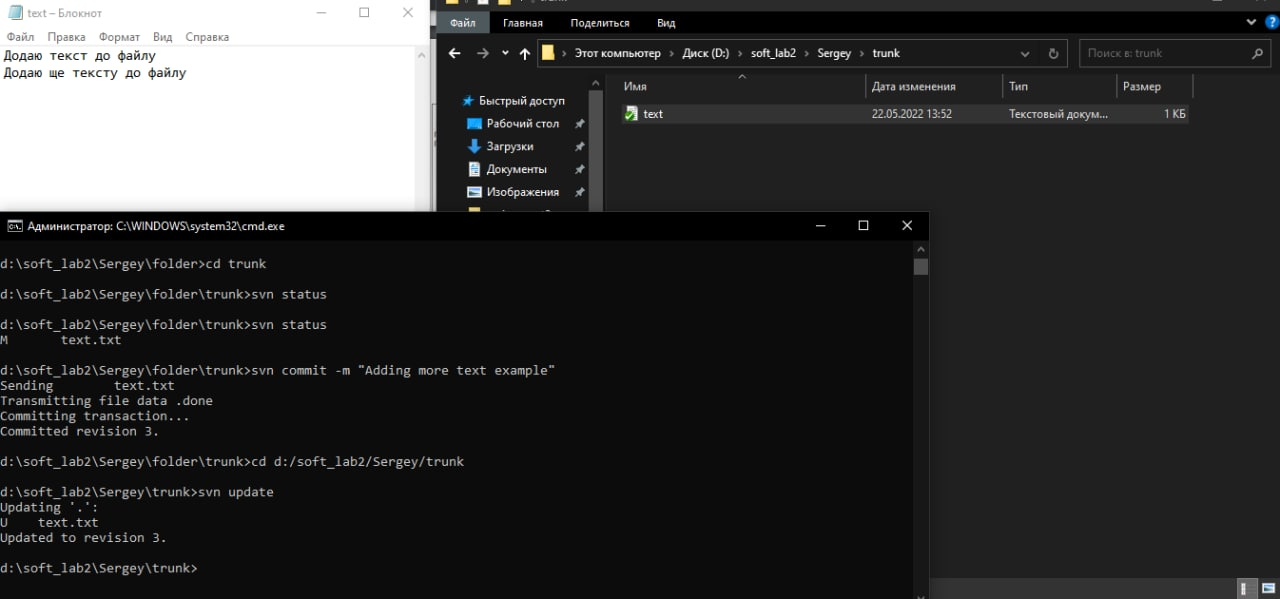


Рисунок 1.5

* + 1. Створюємо нову гілку, переміщаємо туди файл, після чого переключаємось на основну (рис. 1.6).

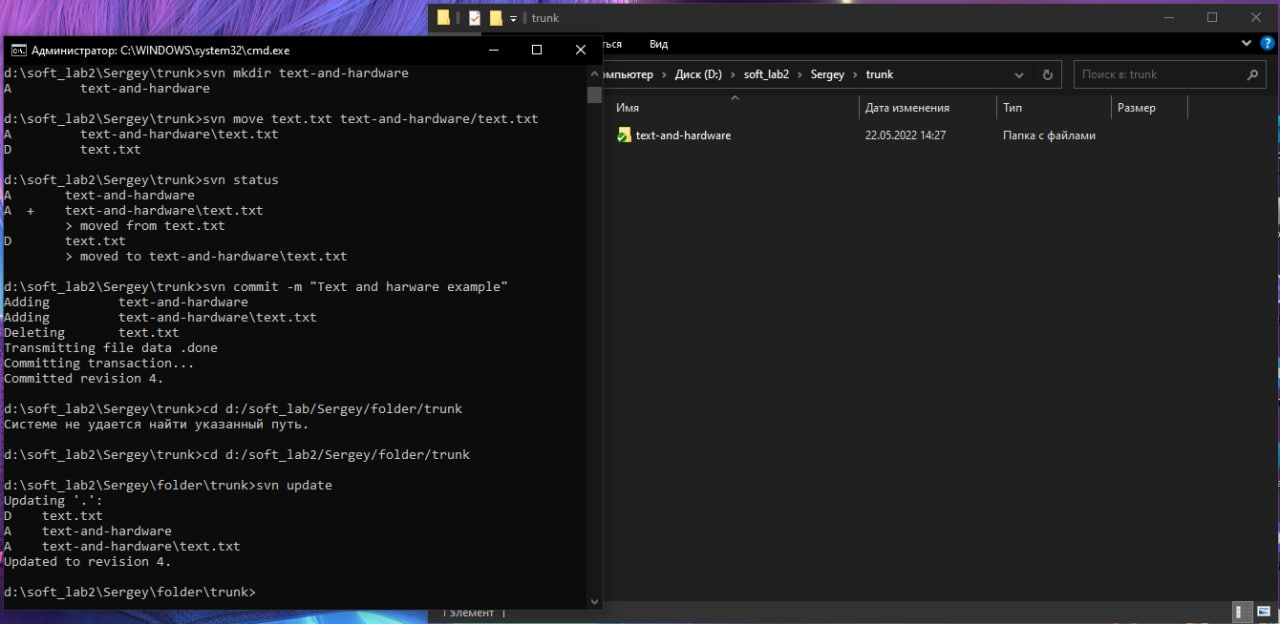


Рисунок 1.6

* + 1. Створюємо нову директорію (рис. 1.7).

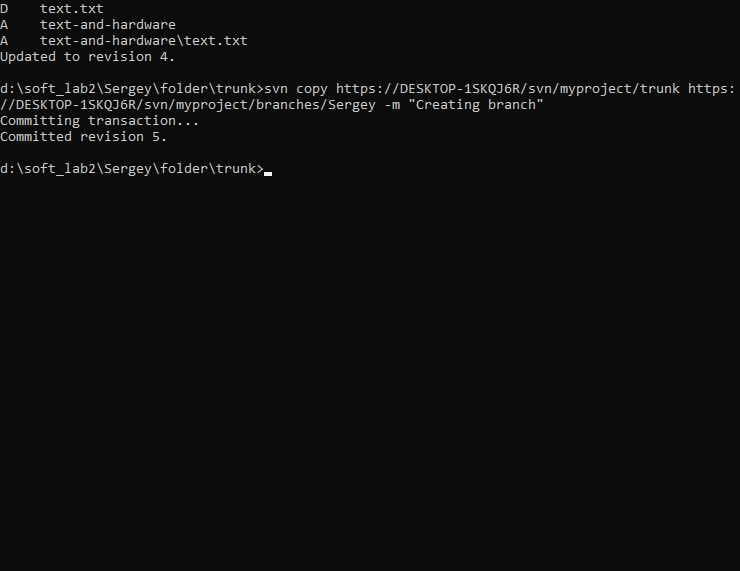


Рисунок 1.7

* + 1. Переключаємось зі ствола на гілку (рис. 1.8).

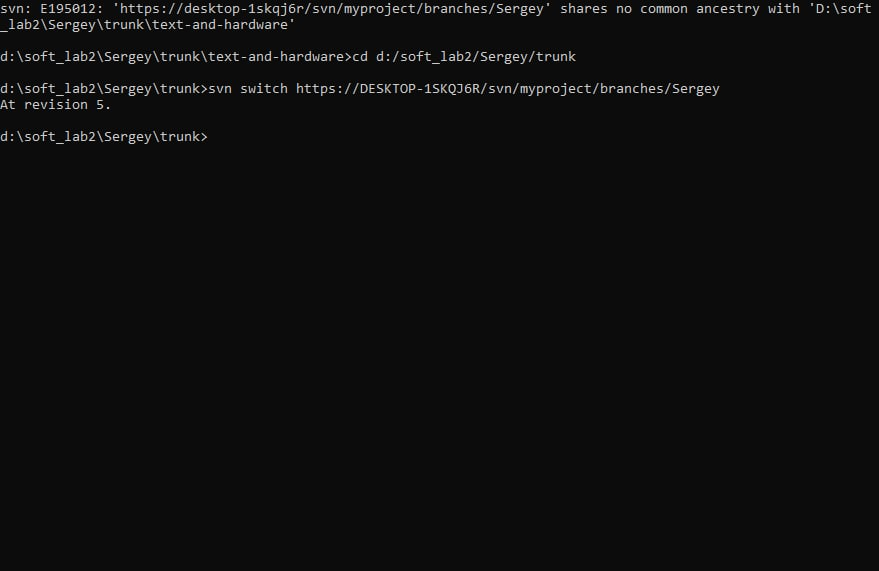


Рисунок 1.8

* + 1. Вносимо зміни та фіксуємо їх на сервері (рис. 1.9).

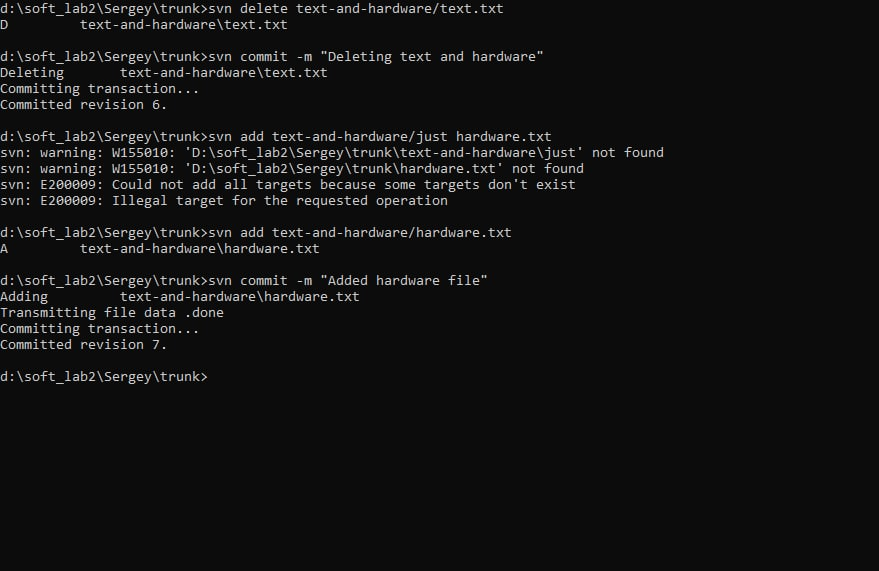


Рисунок 1.9

* + 1. Зливаємо зміни між гілками (рис. 1.10).

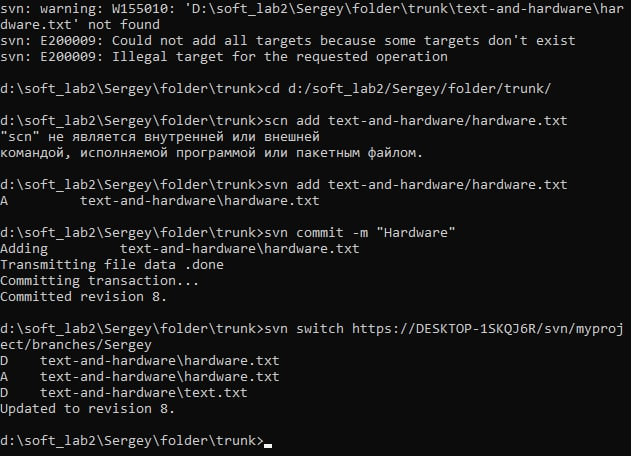


Рисунок 1.10

* + 1. Вилучення гілки (рис. 1.11).

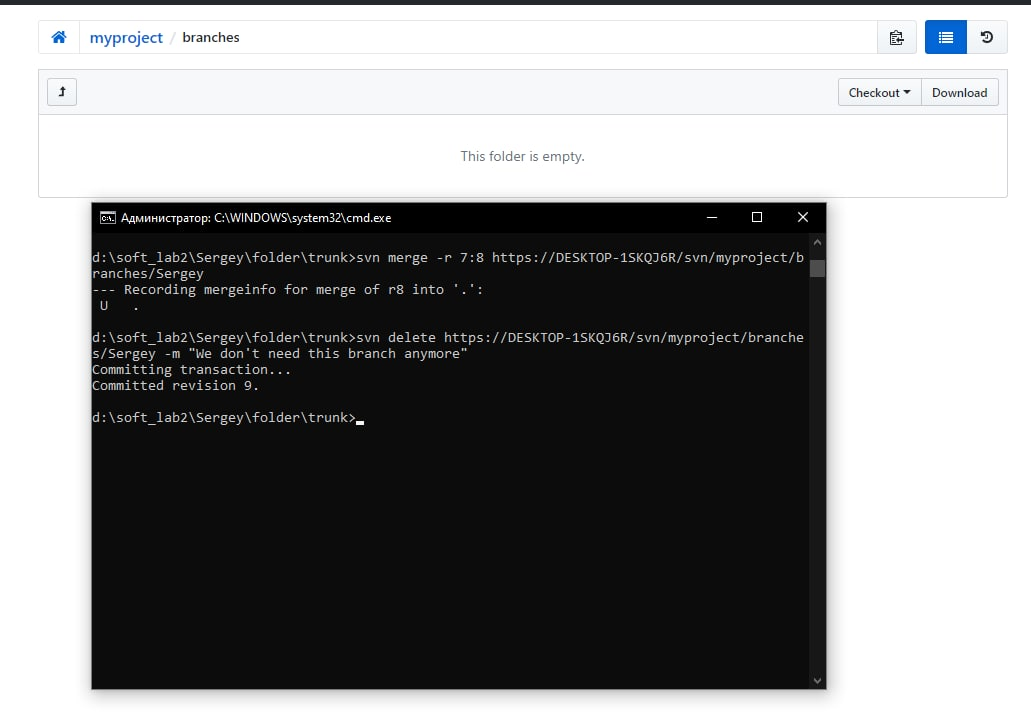


Рисунок 1.11

* + 1. Створення проекту з кодом (рис. 1.12).

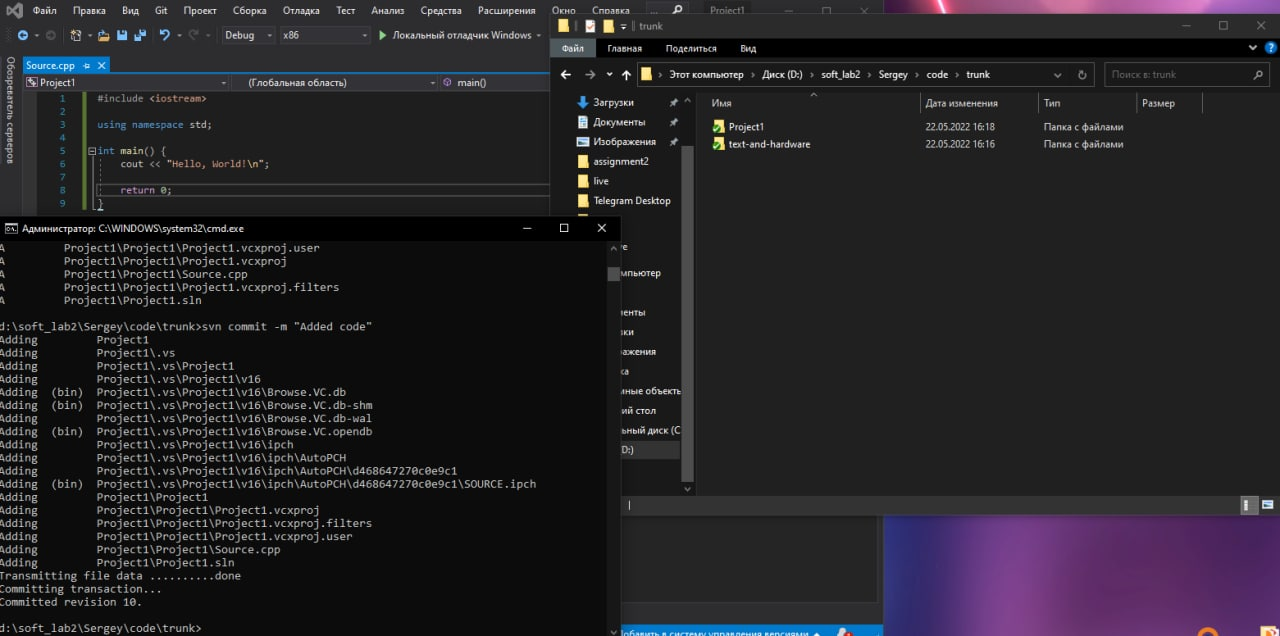


Рисунок 1.12

* + 1. Додаємо зміни та відправляємо до репозиторію (рис. 1.13).

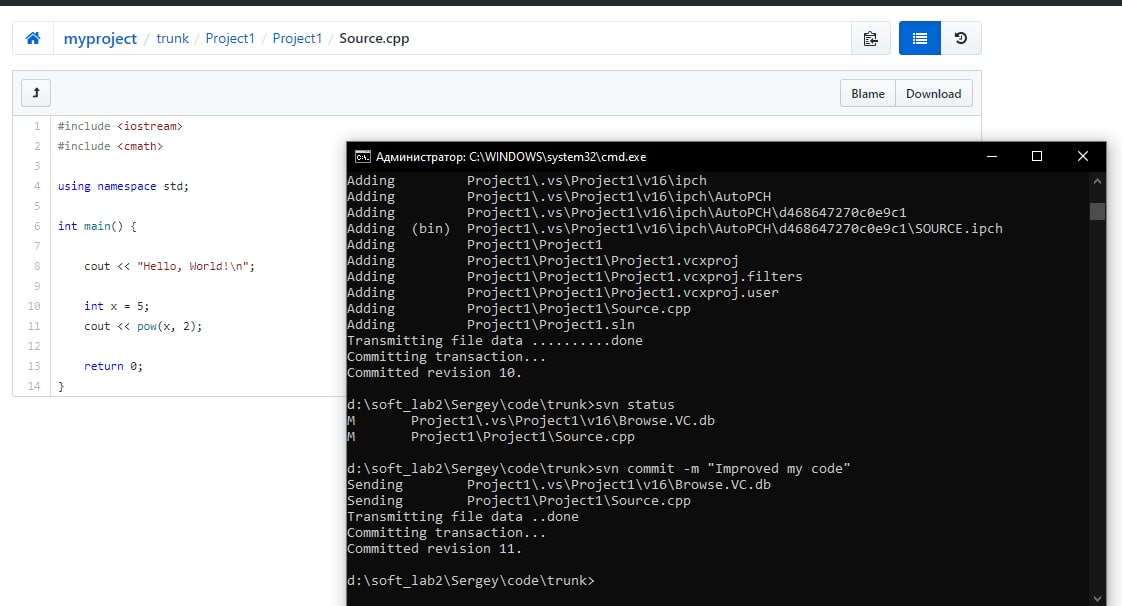


Рисунок 1.13

* 1. **Копії екранних форм з результатами виконання завдання та тексти файлів у декількох ревізіях (Git).**
     1. Створюємо репозиторій на основі одного з раніше розроблених проектів (рис. 1.14 – 1.15; рис 1.16 – 1.17 за допомогою GitKraken).

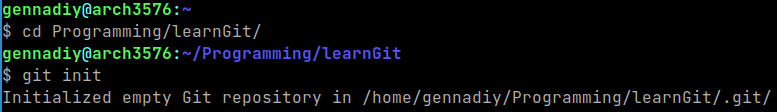


Рисунок 1.14

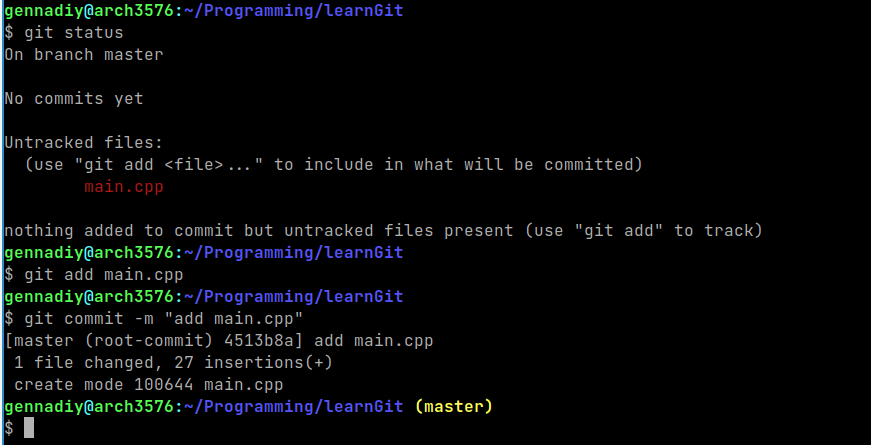


Рисунок 1.15

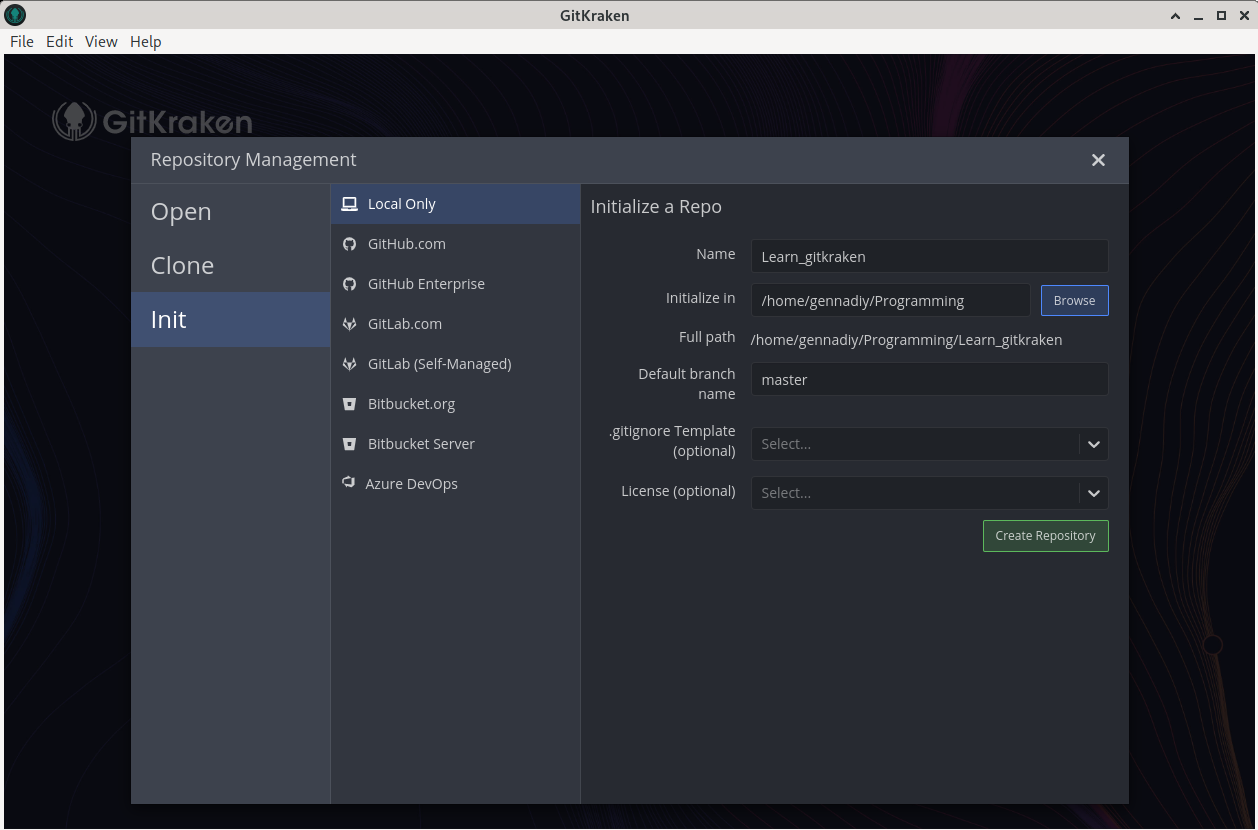


Рисунок 1.16

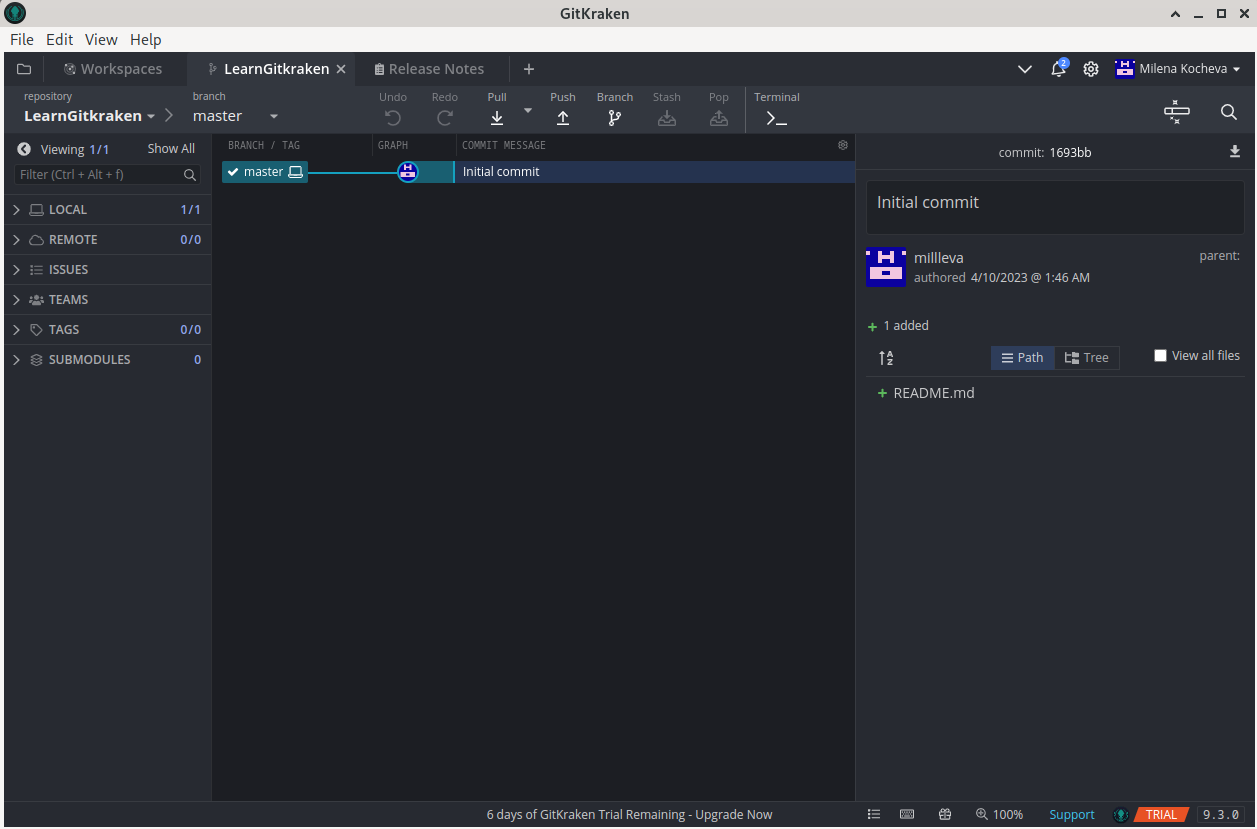


Рисунок 1.17

* + 1. Забороняємо автоматичне додання в репозиторій файлів з розширенням \*.exe (рис. 1.18 – 1.19; рис 1.20 – 1.22 за допомогою GitKraken).

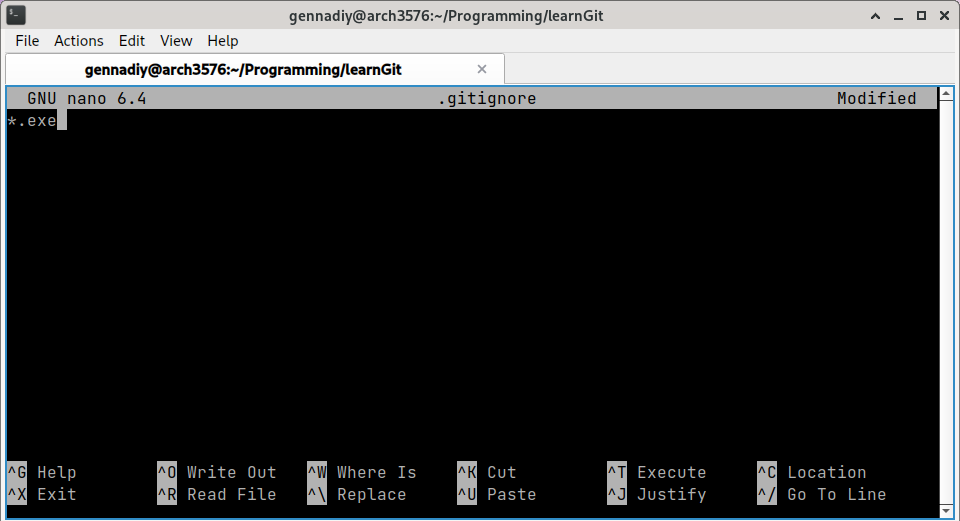


Рисунок 1.18

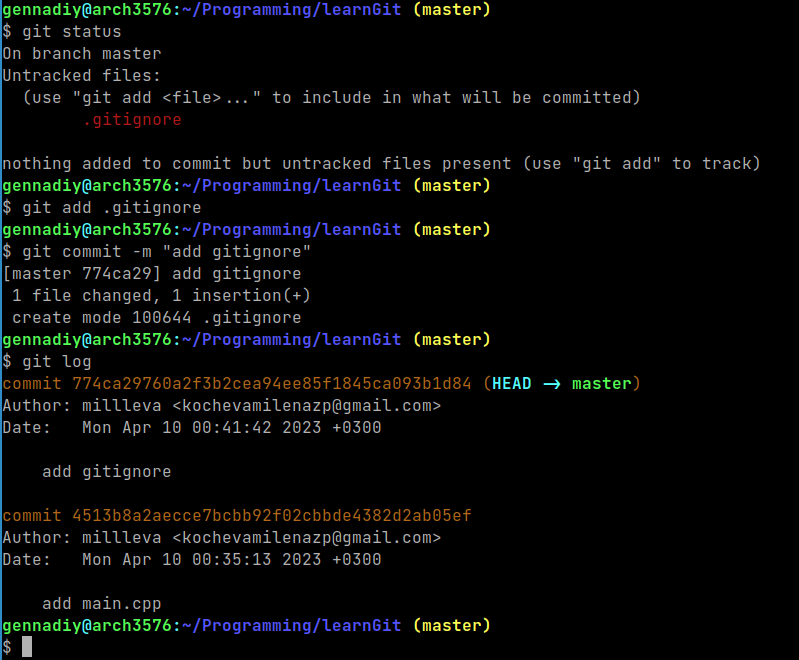


Рисунок 1.19

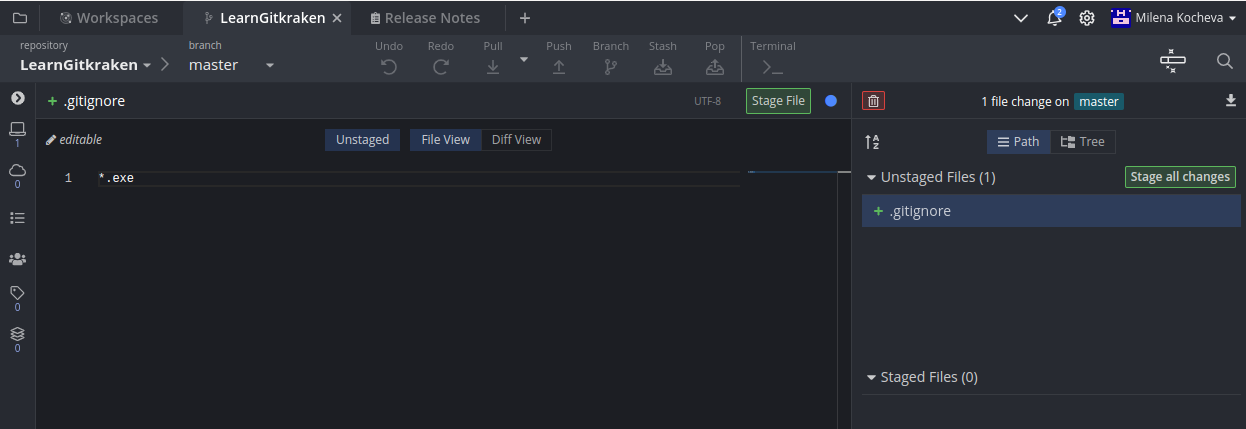


Рисунок 1.20

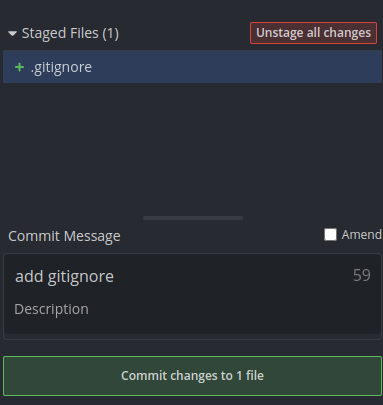


Рисунок 1.21

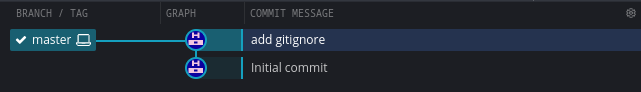


Рисунок 1.22

* + 1. Вносимо зміни в текст проекту та фіксуємо їх, для підпису використовуючи власні дані (рис. 1.23; рис 1.24 – 1.26 за допомогою GitKraken).

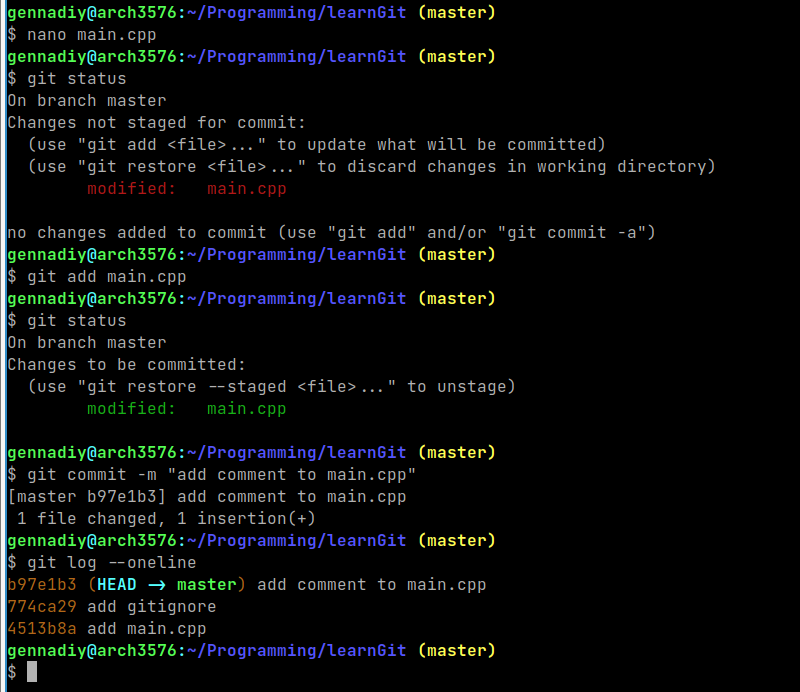


Рисунок 1.23

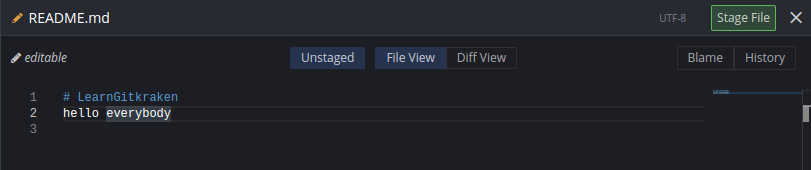


Рисунок 1.24



Рисунок 1.25

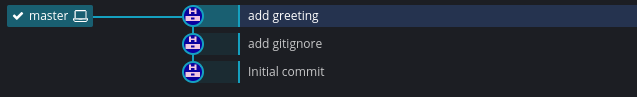


Рисунок 1.26

* + 1. Переглядаємо різницю між новою версією проекту та початковою (рис. 1.27; рис 1.28 за допомогою GitKraken).

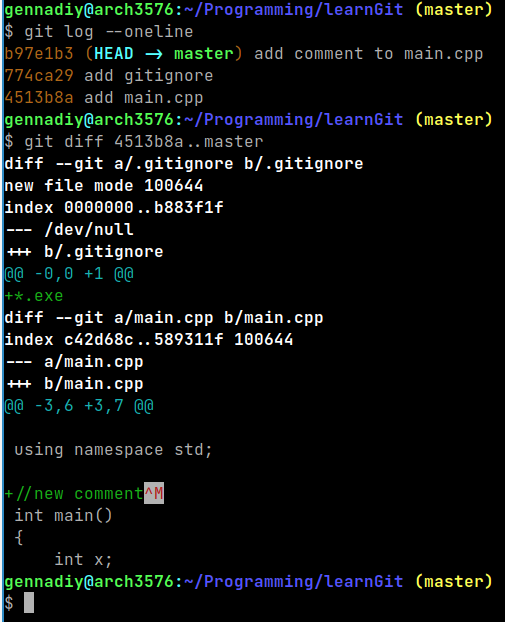


Рисунок 1.27

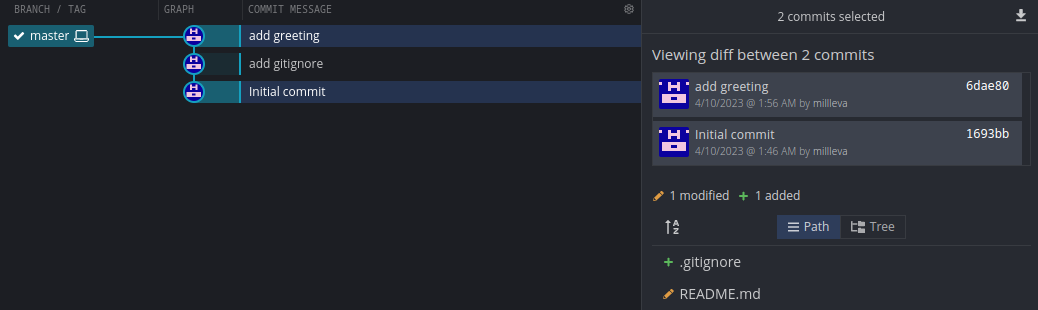


Рисунок 1.28

* + 1. Створюємо власне віддалене сховище, використовуючи сервіс GitHub (рис. 1.29; рис 1.30 за допомогою GitKraken).

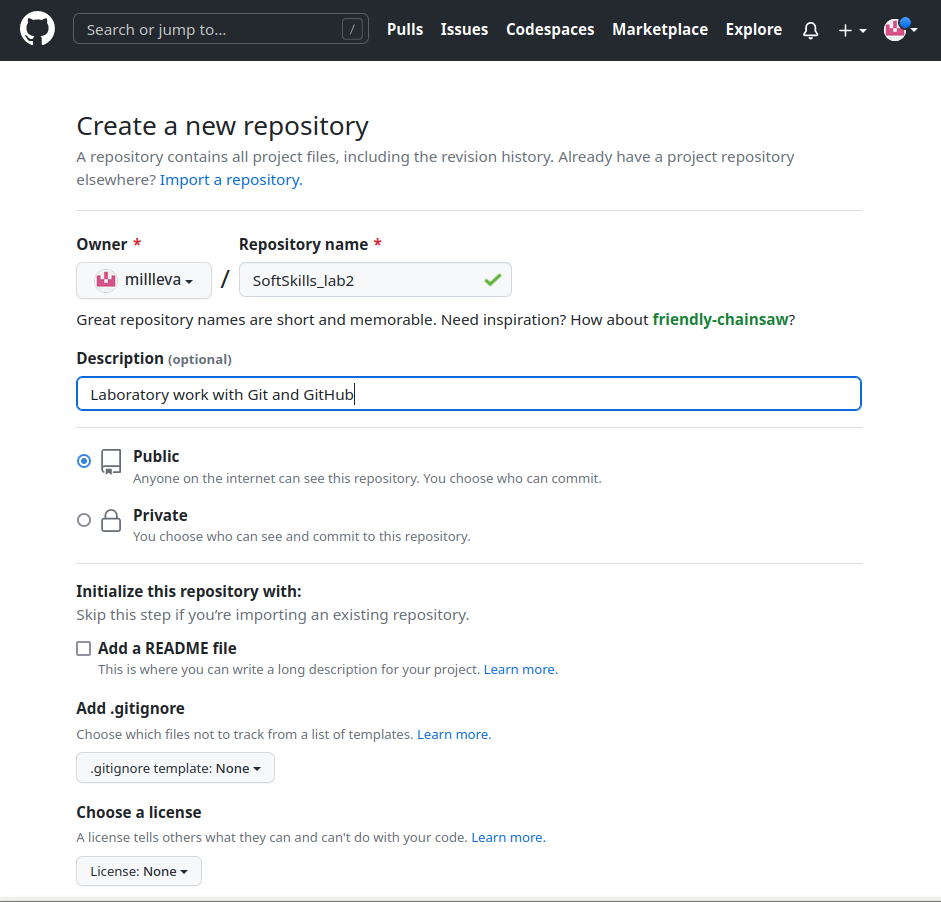


Рисунок 1.29

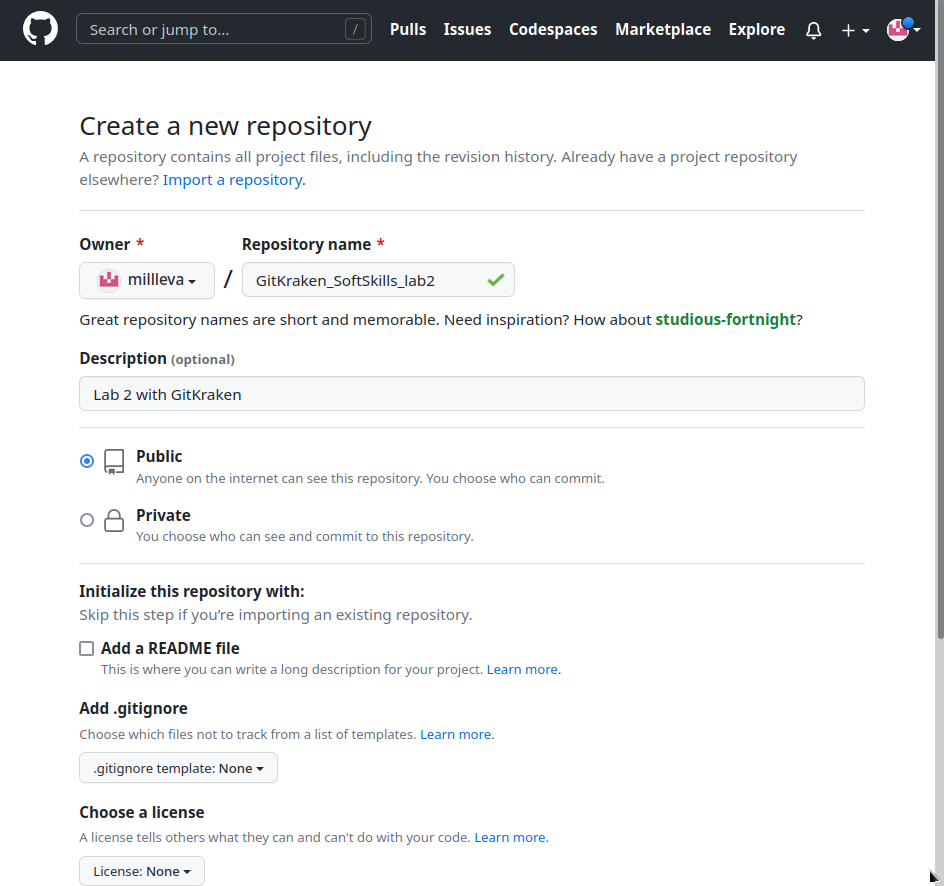


Рисунок 1.30

* + 1. Налаштовуємо локальне сховище для синхронізації з віддаленим та відправляємо локальну версію на сервер (рис. 1.31 – 1.32; рис 1.33 – 1.36 за допомогою GitKraken).

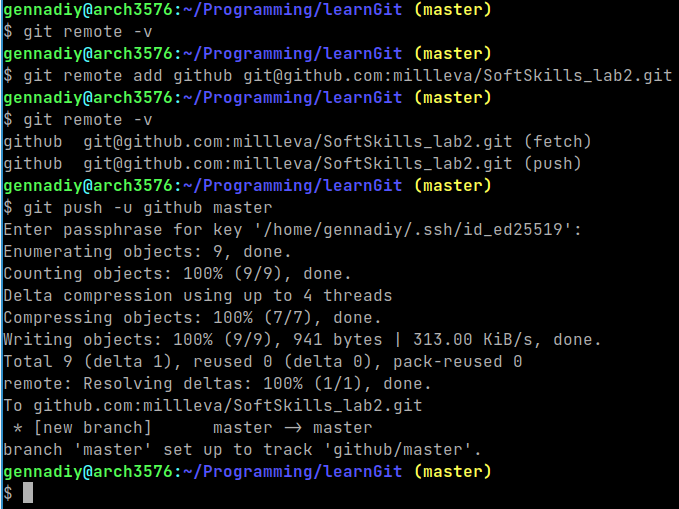


Рисунок 1.31

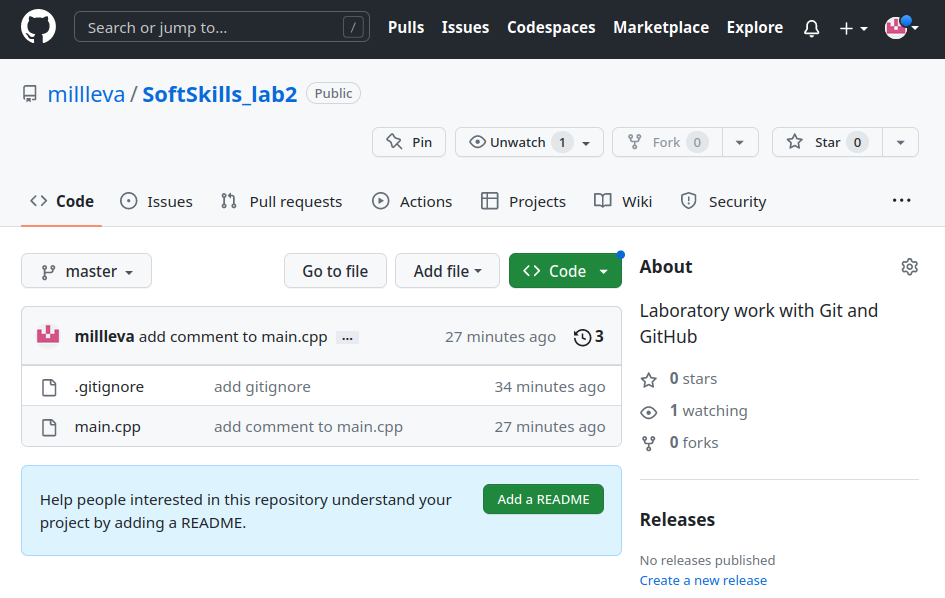


Рисунок 1.32

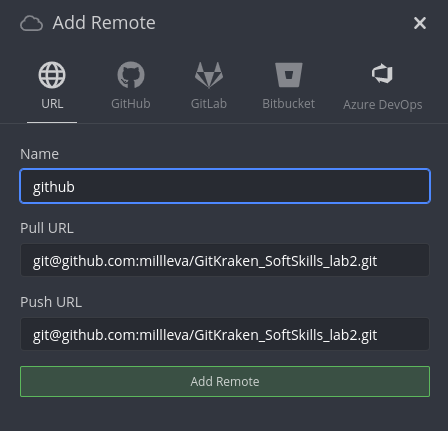


Рисунок 1.33

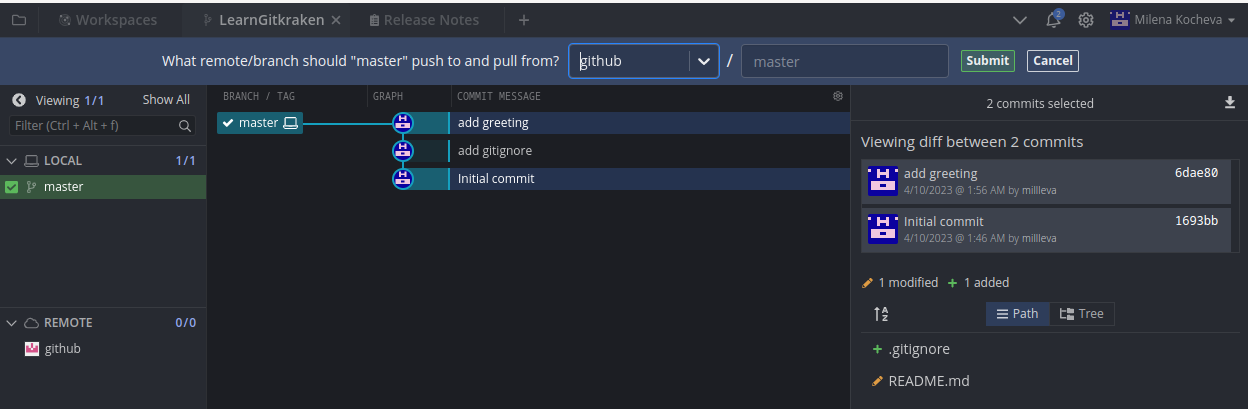


Рисунок 1.34

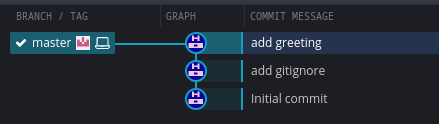


Рисунок 1.35

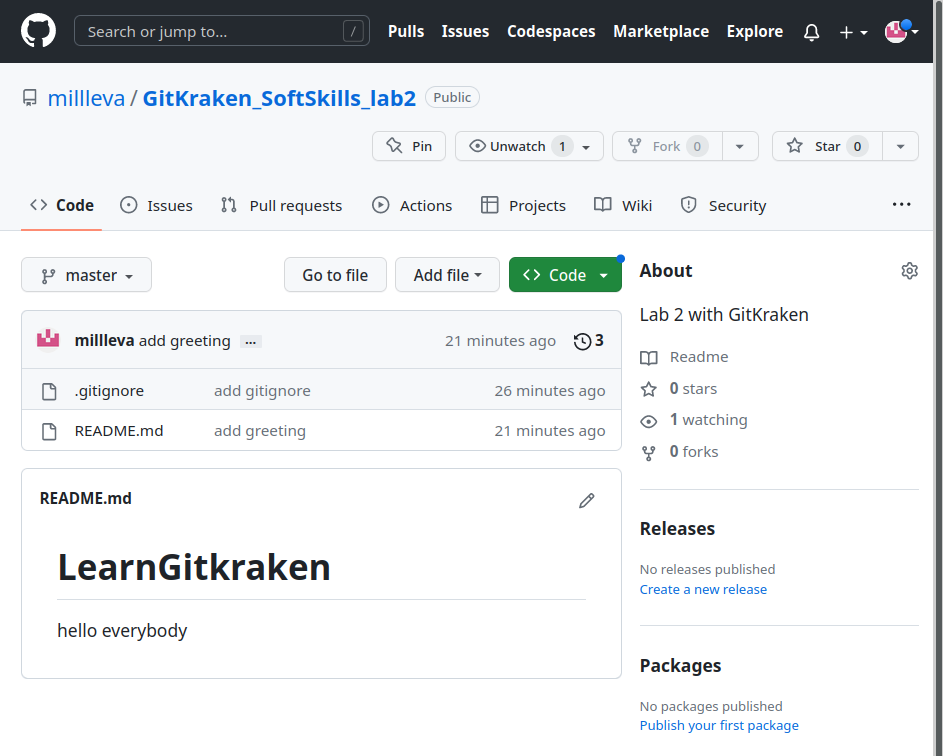


Рисунок 1.36

* + 1. Переглядаємо історію проекту та сторінку проекту через web-інтерфейс (рис. 1.37; рис 1.38 за допомогою GitKraken).

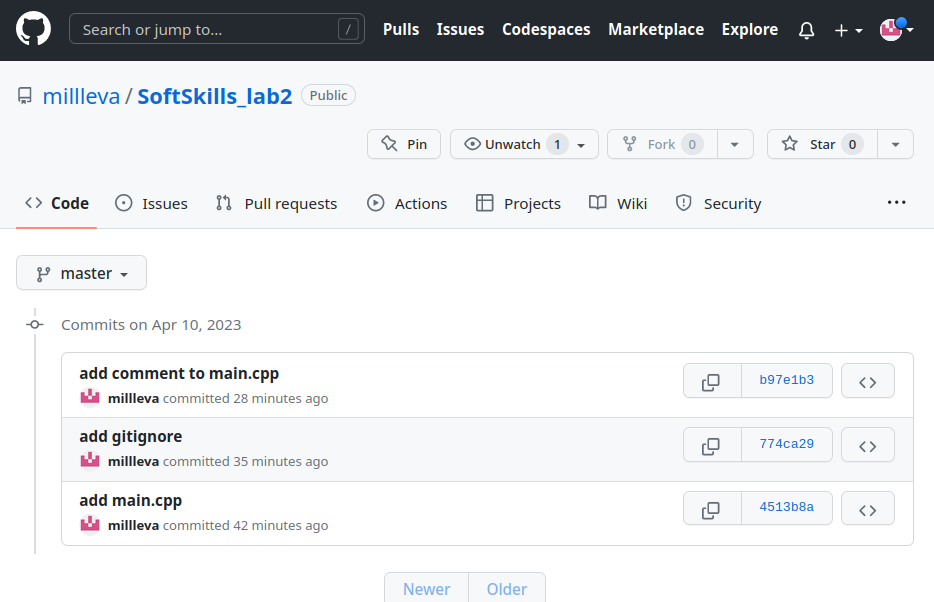


Рисунок 1.37

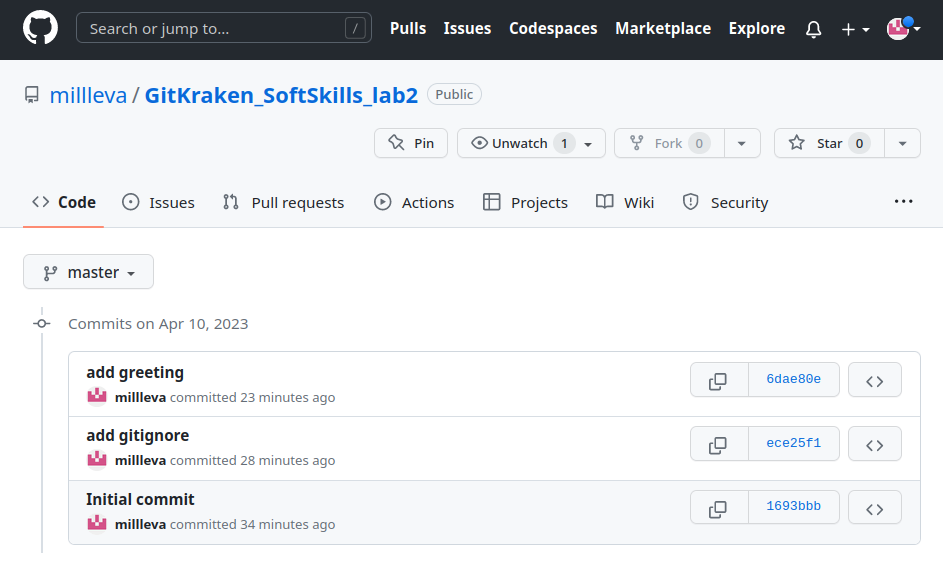


Рисунок 1.38

* + 1. Вносимо зміни в текст проекту та фіксуємо їх (рис. 1.39;

рис 1.40 – 1.41 за допомогою GitKraken).

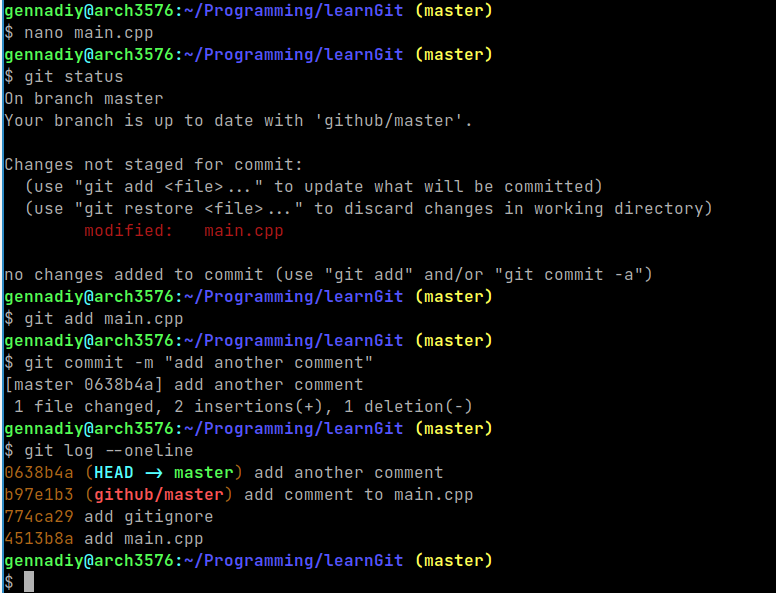


Рисунок 1.39

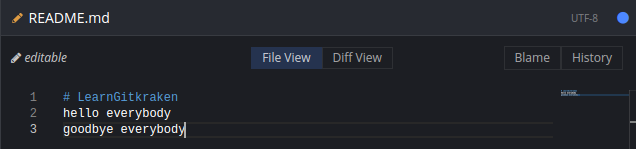


Рисунок 1.40

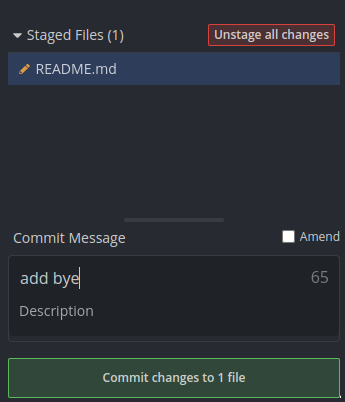
****

Рисунок 1.41

* + 1. Узгоджуємо локальну версію репозиторію з сервером

(рис. 1.42 – 1.43; рис 1.44 – 1.45 за допомогою GitKraken).

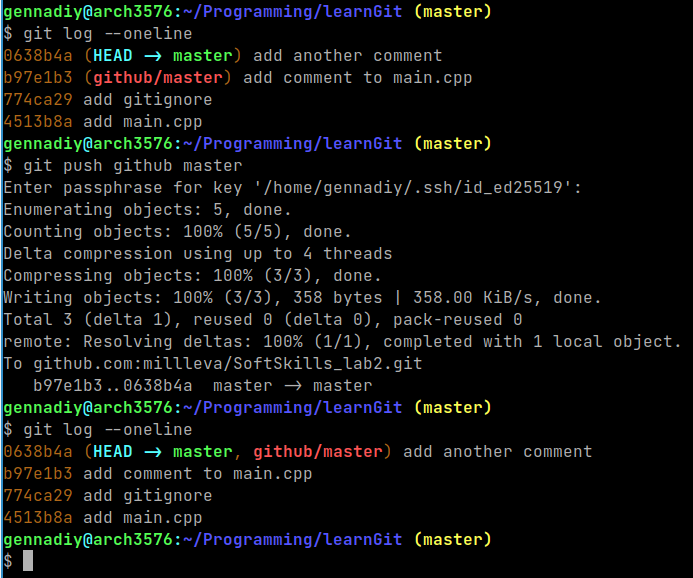


Рисунок 1.42

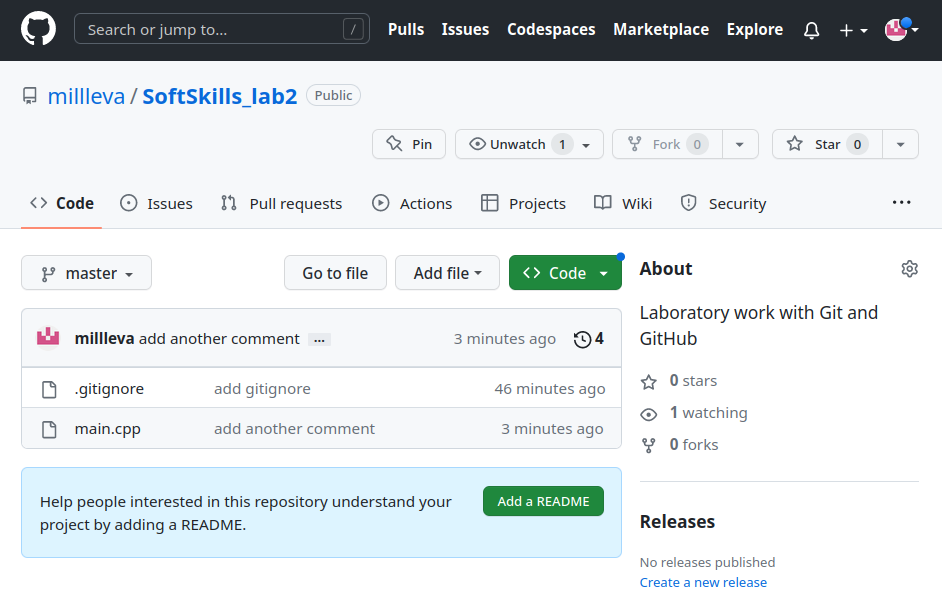


Рисунок 1.43

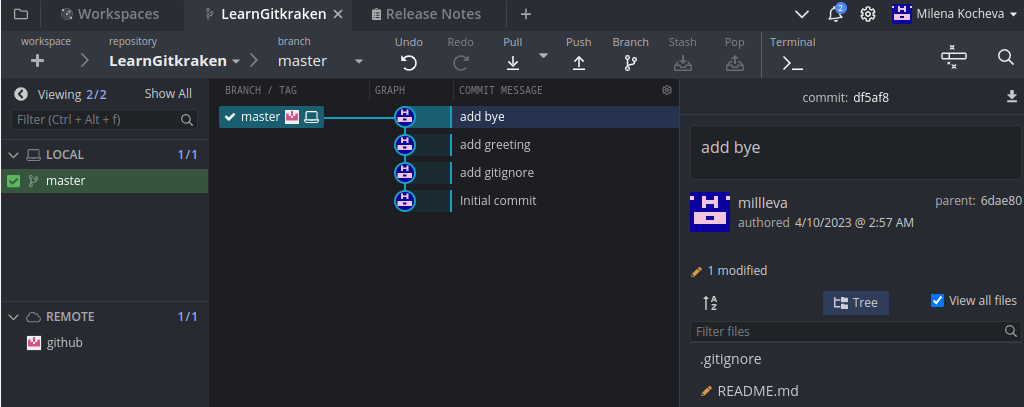
****

Рисунок 1.44

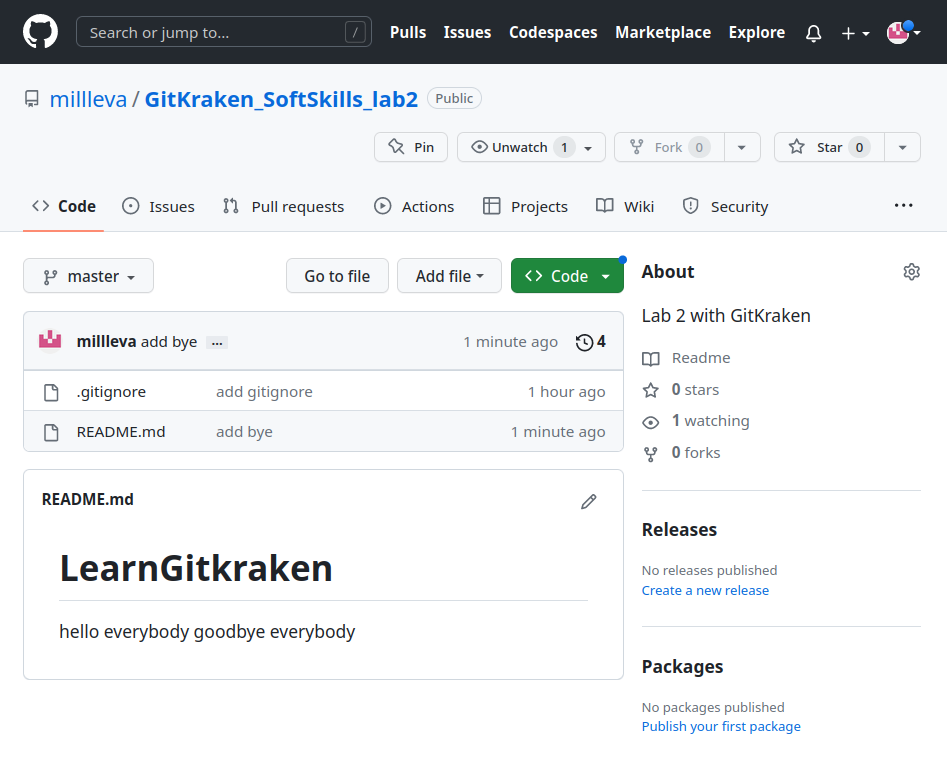
****

Рисунок 1.45

**Висновки**

Я вивчила основні можливості систем керування версіями на прикладі двох популярних систем - Subversion та Git. Порівняла їх можливості і навчилася використовувати їх для групової роботи, що дозволить мені ефективно керувати версіями і змінами в проектах з багатьма користувачами, забезпечувати безпеку коду та легко відстежувати зміни.

* + 1. Що таке система керування версіями?

Система, що дозволяє зберігати декілька версій одного і того ж файла, надає інструментарій переходу до попередніх версій файла, дозволяє відслідковувати зміни, які вносять різні користувачі у відповідні файли

* + 1. Чим відрізняється система керування версіями Git від Subversion?

Git є розподіленою системою керування версіями з можливістю локального комітування та створення гілок, що дозволяє більш ефективно працювати з кодом та вести розробку відокремлено одне від одного. Subversion же є централізованою системою керування версіями, де зміни зберігаються на сервері та відображаються у всіх користувачів, що може викликати конфлікти та проблеми при одночасній роботі над одними та тими ж документами.

* + 1. Які можна виділити особливості Subversion?

Особливості Subversion:

* Subversion відстежує версії як файлів, так і каталогів;
* якщо зміни зроблені в декількох файлах і каталогах, вони публікуються як одна транзакція;
* при будь-яких оновленнях версій між клієнтом і сервером передаються тільки відмінності між файлами;
* Subversion підтримує копіювання, переміщення і перейменування файлів із збереженням історії змін;
* Subversion однаково ефективно працює як з текстовими, так і з бінарними файлами;
* для зберігання версій використовується ієрархія каталогів – для кожної гілки або мітки створюється окремий каталог.
  + 1. Які існують конфлікти під час оновлення робочої копії?

Існує два типи конфліктів:

* конфлікти файлів виникають, коли два (або більше) розробників змінили одні й ті ж рядки файла;
* конфлікти дерев виникають, коли розробник перемістив, перейменував або вилучив файл або теку, які інший розробник також перемістив, перейменував або вилучив або тільки змінив.
  + 1. Для чого виконується злиття гілки та яка команда використовується для цього?

Злиття гілок використовується для об'єднання двох або більше гілок в одну, щоб скомбінувати зміни, які були внесені в кожну з гілок. Це дозволяє розробникам працювати над різними функціями чи різними аспектами проекту одночасно та без конфліктів. Команда merge.