

**Лабораторна робота №1.3**

з дисципліни "Супроводження програмного забезпечення"

тема: «Дослідження ефективності процесів супроводження програмного забезпечення»

Виконав: студент ПІ-414

Воловін Є. О.

Прийняла:

Корнієнко С. П.

**Завдання**

1. Вивчити теоретичні відомості про системи контролю версій.
2. Обрати відкрите програмне забезпечення для контролю версій і використати його для програмного продукту з попередніх лабораторних робіт.
3. Зробити скріншоти проведених дій, описати кроки виконання.
4. Обчислити відповідні оцінки для програмного продукту використовуючи теоретичну інформацію про вартісну оцінку.

# 

# Хід виконання роботи

**Теоретична частина**

Система контролю версій - це система, що записує зміни у файл або набір файлів протягом деякого часу, так що ви зможете повернутися до певної версії пізніше. Як приклад, в цій книзі, для файлів, що знаходяться під контролем версій, буде використовуватися код програмного забезпечення, хоча насправді ви можете використовувати контроль версій практично для будь-яких типів файлів.

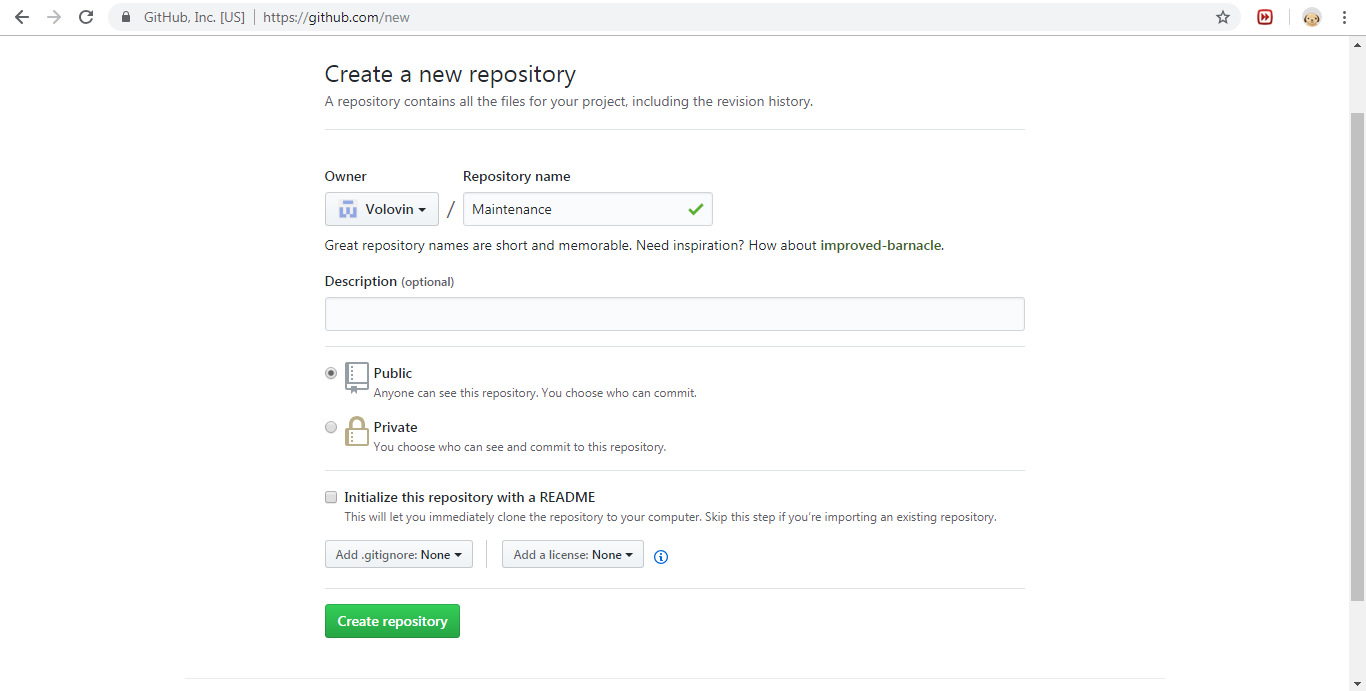
Git — розподілена система керування версіями файлів та спільної роботи. Проект створив Лінус Торвальдс для управління розробкою ядра Linux, а сьогодні підтримується Джуніо Хамано (англ. Junio C. Hamano). Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів.

Система має ряд користувацьких інтерфейсів: наприклад, gitk та git-gui розповсюджуються з самим Git.

Віддалений доступ до репозиторіїв Git забезпечується git-демоном, SSH або HTTP сервером. TCP-сервіс git-daemon входить у дистрибутив Git і є разом з SSH найпоширенішим і надійним методом доступу. Метод доступу HTTP, незважаючи на ряд обмежень, дуже популярний в контрольованих мережах, тому що дозволяє використання існуючих конфігурацій мережевих фільтрів.

**Практична частина**

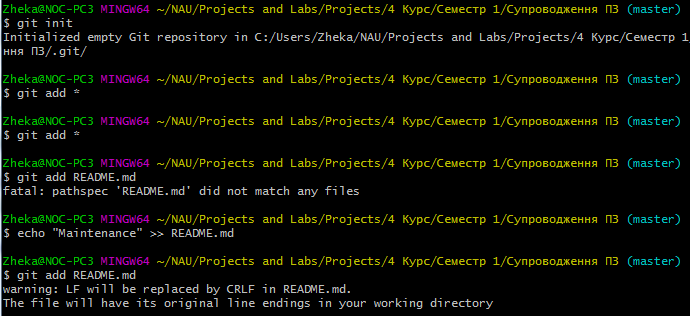
Створення віддаленого репозиторію на сайті системи контролю версій Github.



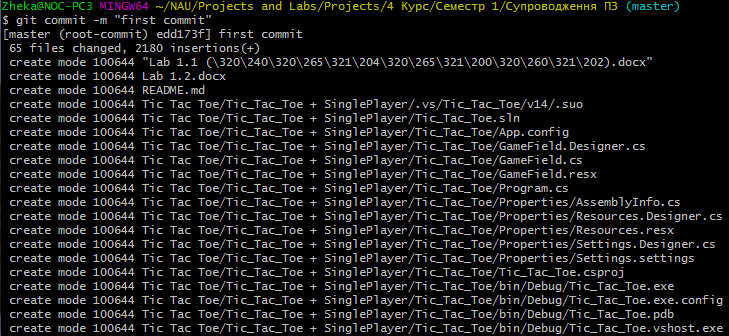
Для того, щоб розпочати роботу з системою контролю, потрібно створити локальний репозиторій для потрібного проекту. Для цього потрібно відкрити GitBash (чи інший надаваний інтерфейс взаємодії) та перейти до папки проекту.



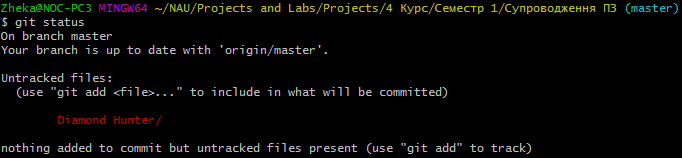
Перейшовши до потрібної папки пишемо команду “git init” вона створює по заданому шляху власну папку до якої буде додаватись потрібна інформація. Командою “git add \* “ додаємо всі файли, що є в папці до локального репозиторію.



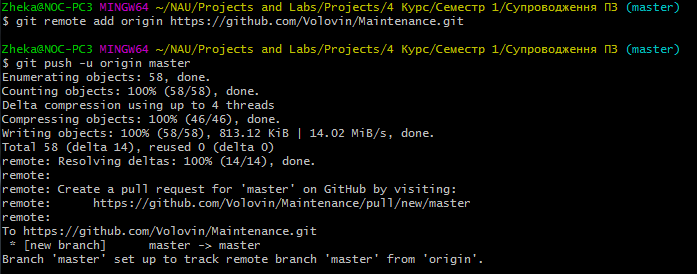
Наступним кроком є реєстрація змін проведених при роботі з репозиторієм, оскільки проект тільки-тільки був доданий до репозиторію – за допомогою команди “git commit -m ” вносимо до репозиторію інформацію про проведені дії.

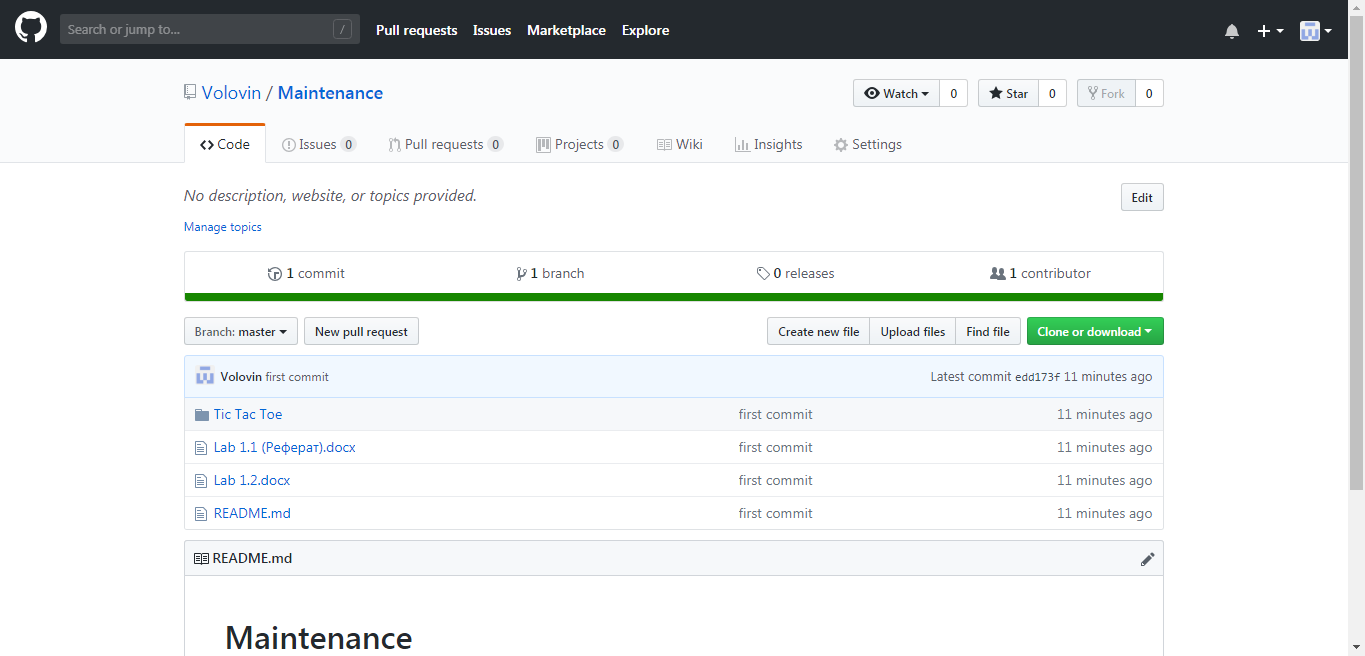


Для перевірки статусу репозиторію виконуємо команду “git status” є дана команда порівнює стан локального репозиторію зі станом проекту відстежуваного системою контролю – в разі наявності недоданих до репозиторію файлів вивід буде помічати їх червоним кольором, при потребі зареєструвати оновлення та зміни у репозиторії інтерфейс сповіщує про це та зеленим кольором помічає інформацію про зміни.



Останнім кроком є підключення до створеного на початку віддаленого репозиторію (команда “git remote add origin ”) та синхронізація станів репозиторіїв (команда “git push -u remote origin”).





**Розрахунки**

Значення констант для оцінок проекту були використані з огляду на органічний тип процесу розробки ПЗ. KDSI – кількість рядків коду в тисячах.



Для базової моделі були пораховані показники людиночасів, затрачених на роботу з розробки програмного продукту - , орієнтований показник тривалості -  , та потрібної кількості розробників .





Для середньої моделі були обчислені показники людиночасів - , середнє значення трудоємкості розробки -  , та орієнтовний показник тривалості з огляду на трудоємкість - .

Останніми були вирахувані оцінки вартості супроводження для даного програмного забезпечення

1 рівень моделі



2 рівень моделі



**Висновок**

В ході виконання третьої лабораторної роботи я дослідив використання систем контролю версій для відслідковування змін при розробці програмного забезпечення, отримав практичні навички з обчислення вартісних оцінок з супроводження програмного забезпечення. Освоїв теоретичні відомості, потрібні для проведення вартісного аналізу програмного забезпечення.