Лабораторна робота №1

Тема: Знайомство з системою контролю версій Git. Поняття репозиторію.

Мета роботи: ознайомитися системами керування версіями. Дослідити та отримати практичні навики щодо створення найпростішої програми та власного репозиторію.

1. Теоретичні відомості

Система керування версіями (англ. source code management, SCM) — програмний інструмент для керування версіями одиниці інформації: вихідного коду програми, скрипту, веб-сторінки, веб-сайту, 3D моделі, текстового документу тощо.

Система керування версіями — це потужний інструмент, який дозволяє одночасно, без завад один одному, проводити роботу над груповими проектами.

Системи керування версіями зазвичай використовуються при розробці програмного забезпечення для відстеження, документування та контролю над поступовими змінами в електронних документах: у програмному коду застосунків, кресленнях, електронних моделях та інших документах, над змінами яких одночасно працюють декілька людей.

Кожна версія позначається унікальною цифрою чи літерою, зміни документу занотовуються. Зазвичай також зберігається автор зробленої зміни та її час.

Інструменти для контролю версій входять до складу багатьох інтегрованих середовищ розробки.

Система керування версіями існують двох основних типів: з централізованим сховищем та розподіленим (рис. 1).

Система збереження історії редагувань статей, що застосовується у Вікіпедії є прикладом системи керування версіями.

Система контролю дозволяє зберігати попередні версії файлів та завантажувати їх за потребою. Вона зберігає повну інформацію про версію кожного з файлів, а також повну структуру проекту на всіх стадіях розробки. Місце зберігання даних файлів називають репозиторієм. В середині кожного з репозиторіїв можуть бути створені паралельні лінії розробки — гілки.

Гілки зазвичай використовують для зберігання експериментальних, незавершених(alpha, beta) та повністю робочих версій проекту(final). Більшість систем контролю версії дозволяють кожному з об'єктів присвоювати теги, за допомогою яких можна формувати нові гілки та репозиторії.

Централізовані системи контролю версії

Централізована система контролю версії (клієнт-серверна) — система, дані в якій зберігаються в єдиному «серверному» сховищі. Весь обмін файлами відбувається з використанням центрального сервера. Є можливість створення та роботи з локальними репозиторіями (робочими копіями).

Переваги:

- Загальна нумерація версій;
- Дані знаходяться на одному сервері;
- Можлива реалізація функції блокування файлів;
- Можливість керування доступом до файлів;

Недоліки:

- Потреба в мережевому з'єднанні для оновлення робочої копії чи збереження змін;

До таких систем відносять Subversion, Team Foundation Server.

Розподілені системи контролю версії

Розподілена система контролю версії (англ. Distributed Version Control System, DVCS) — система, яка використовує замість моделі клієнт-сервер, розподілену модель зберігання файлів. Така система не потребує сервера, адже всі файли знаходяться на кожному з комп'ютерів.

Переваги:

- Кожний з розробників працює зі своїм власним репозиторієм;
- Рішення щодо злиття гілок приймається керівником проекту;
- Немає потреби в мережевому з'єднанні;

Недоліки:

- Немає можливості контролю доступу до файлів;
- Відсутня загальна нумерація версії файла;
- Значно більша кількість необхідного дискового простору;
- Немає можливості блокування файлів;

До таких систем відносять Git, Mercurial, SVK, Monotone, Codeville, BitKeeper

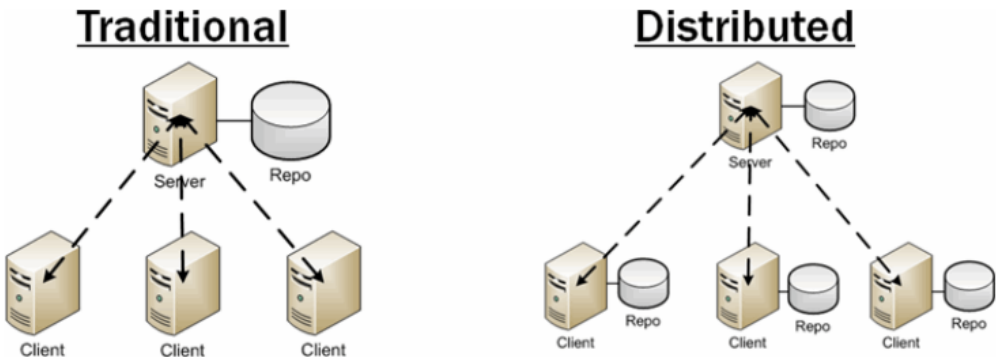


Рисунок 1 – Системи контролю версій

Використання системи контролю версії є необхідним для роботи над великими проектами, над якими одночасно працює велика кількість розробників. Системи контролю версії надають ряд додаткових можливостей:

- можливість створення різних варіантів одного документу;
- документування всіх змін (коли ким було змінено/додано, хто який рядок змінив);
- функція контролю доступу користувачів до файлів (є можливість його обмеження для різних користувачів);
- створення документації проекту з поетапним записом змін в залежності від версії;
- давання пояснення до змін та документування їх.

Найбільш відомими веб-сервісами для хостингу проектів на базі систем керування версіями є:

- GitHub (https://github.com/);
- BitBucket (https://bitbucket.org/);
- GitLab (https://gitlab.com/).

GitHub — один з найбільших веб-сервісів для спільної розробки програмного забезпечення. Існують безкоштовні та платні тарифні плани користування сайтом. Базується на системі керування версіями Git і розроблений на Ruby on Rails і Erlang компанією GitHub, Inc (раніше Logical Awesome).

Розробники сайту називають GitHub «соціальною мережею для розроб-

ників».

Окрім розміщення коду, учасники можуть спілкуватись, коментувати редагування один одного, а також слідкувати за новинами знайомих. За допомогою широких можливостей Git програмісти можуть поєднувати свої репозиторії – GitHub дає зручний інтерфейс для цього і може показувати вклад кожного учасника в вигляді дерева.

Для проектів є особисті сторінки, невеликі Вікі та система відстеження помилок. Прямо на сайті можна дивитись файли проектів з підсвічуванням синтаксису для більшості мов програмування.

Кількість приватних (закритих для перегляду користувачами Інтернету) репозиторіїв – 5. Для того, щоб мати можливість створювати більше приватних репозиторіїв потрібно переходити на платний тарифний план. Кількість відкритих репозиторіїв – необмежена.

Bitbucket — веб-сервіс для хостингу проектів на базі систем керування версіями Mercurial та Git. Bitbucket надає як безкоштовні так і платні послуги. Bitbucket є аналогом GitHub, проте на відміну від GitHub, у якого при безкоштовному профілі файли зберігаються лише у відкритому доступі, Bitbucket дозволяє безкоштовно створювати приватні репозиторії з можливостю спільної роботи з файлами до 5-ти користувачів.

Основні можливості:

- безкоштовний дисковий простір до 2 Гб;
- необмежена кількість відкритих репозиторіїв;
- необмежена кількість приватних репозиторіїв (до 5-ти користувачів);

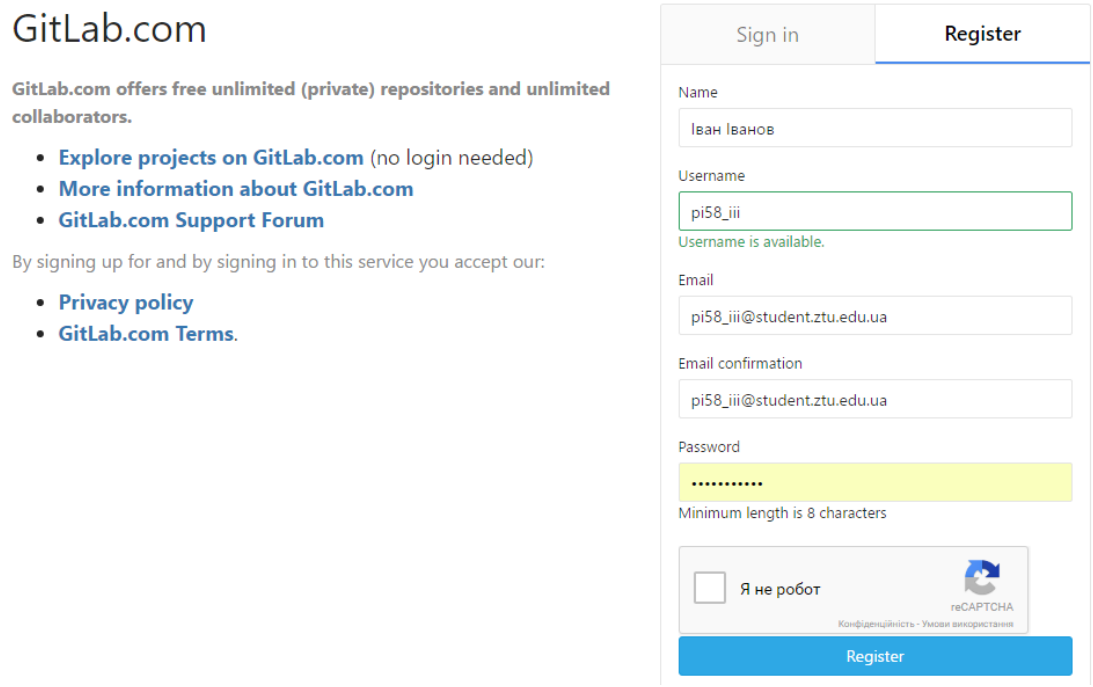
GitLab — сайт та система керування репозиторіями програмного коду для Git, з додаткових можливостей: власна вікі та система відстеження помилок. GitLab — компанія, що пропонує схожі з GitHub користувацькі послуги із додатковими перевагами, як то приватні репозиторії для безкоштовних підписників. Також суттєвою перевагою є можливість розгорнути систему на сторонніх серверах. Програмне забезпечення для GitLab була написана Дмитром Запорожцем з України.

Завдання на лабораторну роботу:

1. Ознайомитись з теоретичними відомостями, ретельно опрацювати матеріал. Вміти давати пояснення термінам та поняттям: система керування версіями; централізовані та розподілені системи контролю версіями; репозиторій; приватні та відкриті репозиторії; GitHub; GitLab; BitBucket.

2. Зареєструватися на сайті GitLabта створити репозиторій:

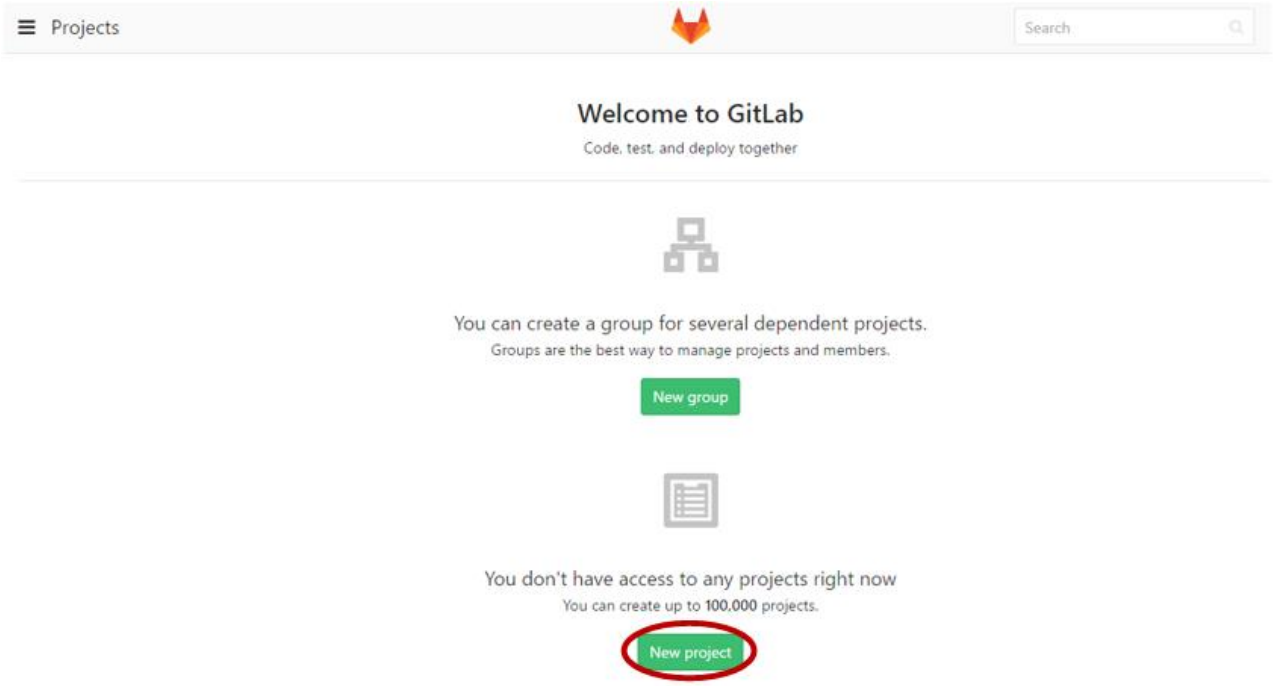
2.1. Зареєструватися на сайті https://gitlab.com/



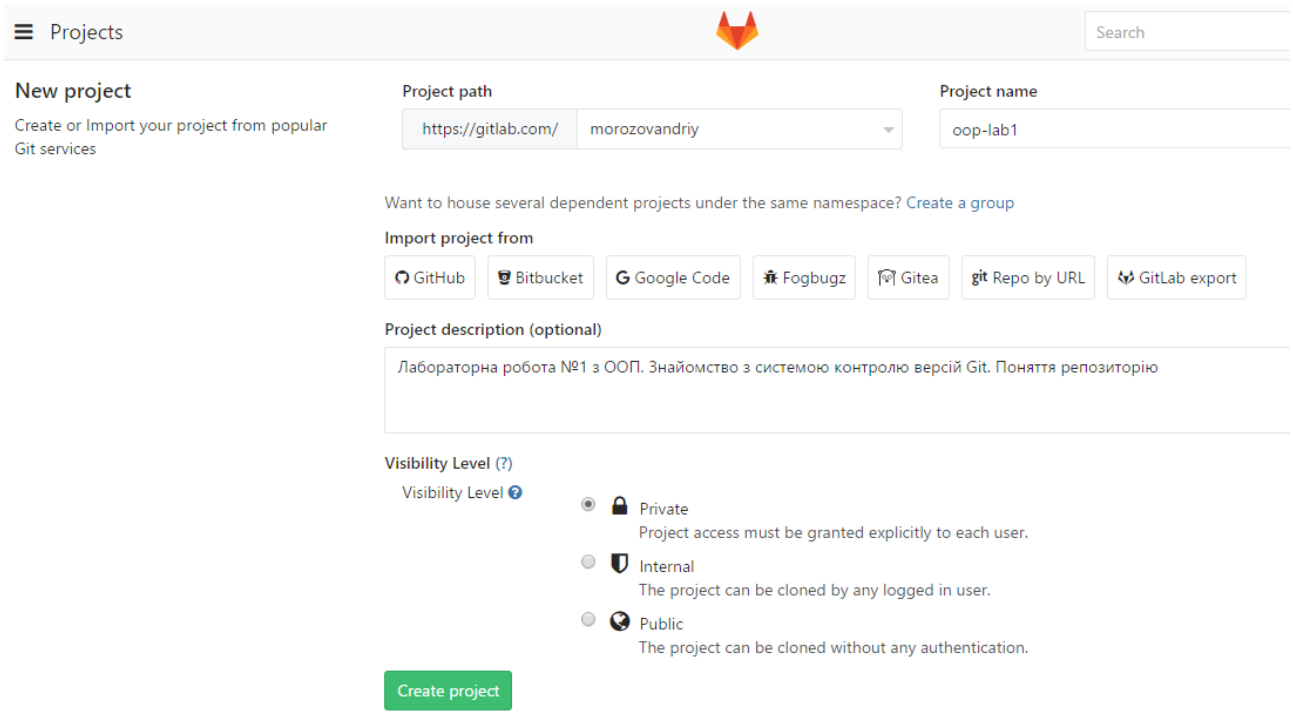
2.2. Увійти на власну пошту та підтвердити реєстрацію у листі, який надійшов з сервера GitLab.

2.3. Повернутися на сайт GitLabі увійти під власним логіном та паролем.

2.4. Створити перший репозиторій з назвою «ООР-Lab1»:



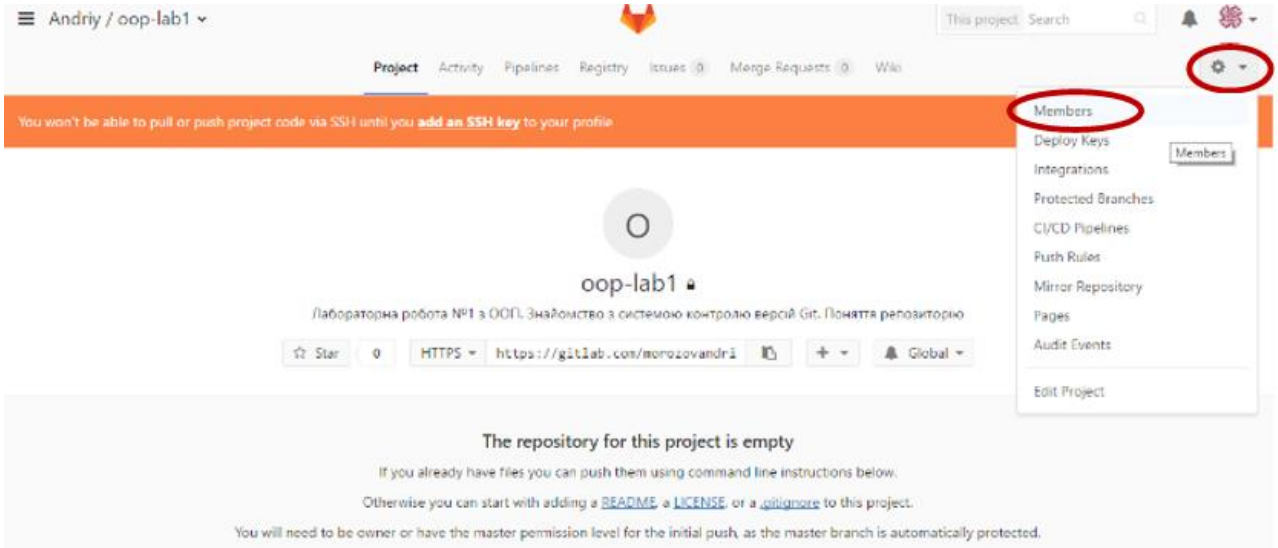
Project name: oop-lab1
Project description: Лабораторна робота №1 з ООП. Знайомство з системою контролю версій Git. Поняття репозиторію
Visibility Level:Private



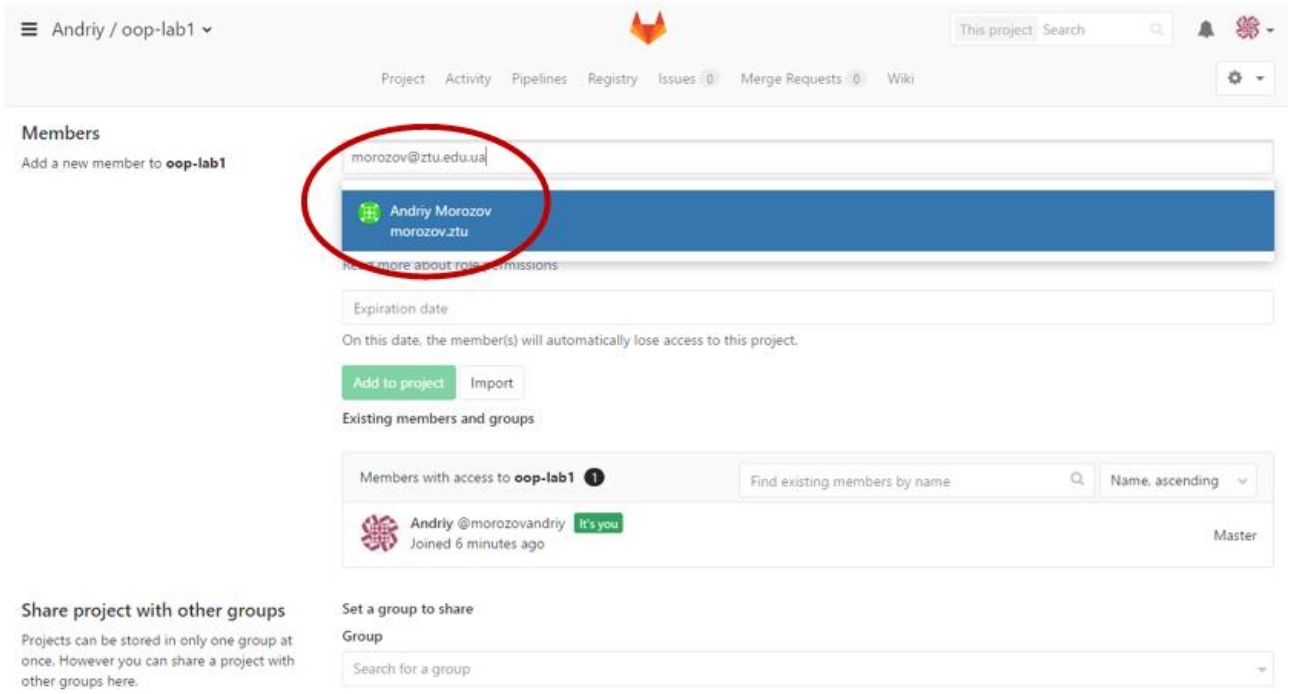
2.5. Надати доступ до репозиторію викладачам (обрати тих хто працює з вашою підгрупою):

- Левківський В.Л. (levkivskyy@ztu.edu.ua)
- Терещук С.О. (kkn_tso@ztu.edu.ua)
- Оринчак А.І. (kkn_oai@ztu.edu.ua)
- Праздніков О.В. (kkn_pov@ztu.edu.ua)

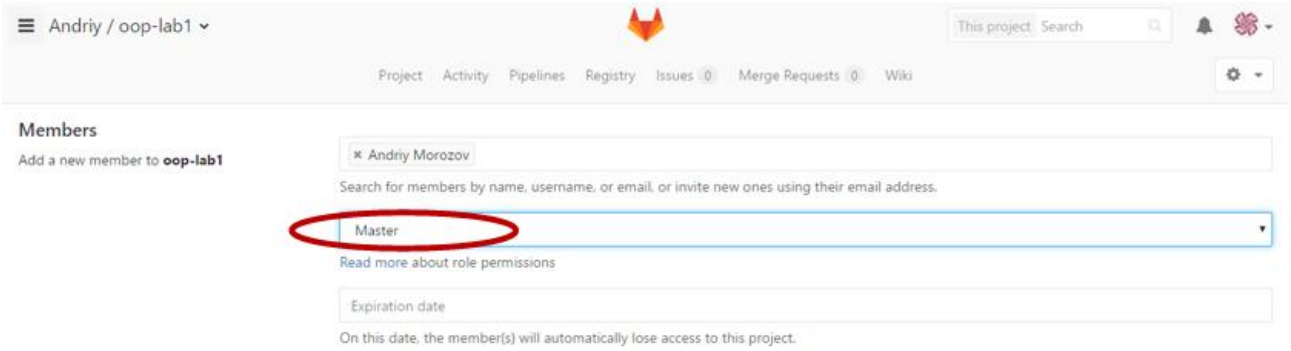
Для цього перейти до розділу «Members»:



Ввести електронну пошту викладача, вибрати зі списку здайненого користувача:



Поставити рівень доступу: «Master»



Натиснути «Add to project».

3.Створення проекту yVisual Studio.

3.1. Створити консольний проект Visual C# з такими параметрами:

- Назва рішення: OOP1Lab1;
- Назва проекту: ConsoleApp.

3.2. У програму додати два рядки:

```
Console.WriteLine("Hello world!");
Console.ReadKey();
```

4. Встановити програмне забезпечення для системи контролю версій

4.1. Зайти на сайт (https://git-scm.com/), скачати установочний файл.

4.2. Запустити установочний файл і у діалогових вікнах вибрати відповідні налаштування.

5. Виконати початкове налаштування локального репозиторію.

5.1. Відкрити командний рядок (комбінація клавіш Win-R, команда

“cmd”).

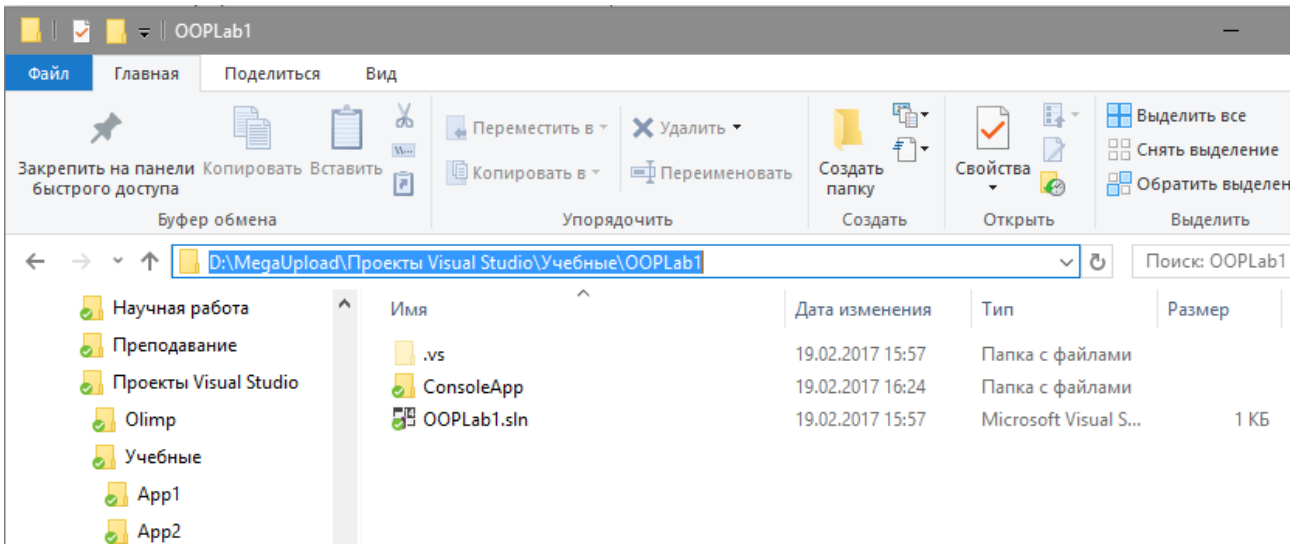
5.2. Виконуємо початкові налаштування:

```
git config --global user.name "IvanIvanov"
git config --global user.email "pi58_iii@student.ztu.edu.ua"
```

Ці дії потрібно виконати лише один раз, після встановлення Githa комп'ютер.

5.3.Перейти у каталог, де зберігається папка рішення, використовуючи команди командного рядка Windows.

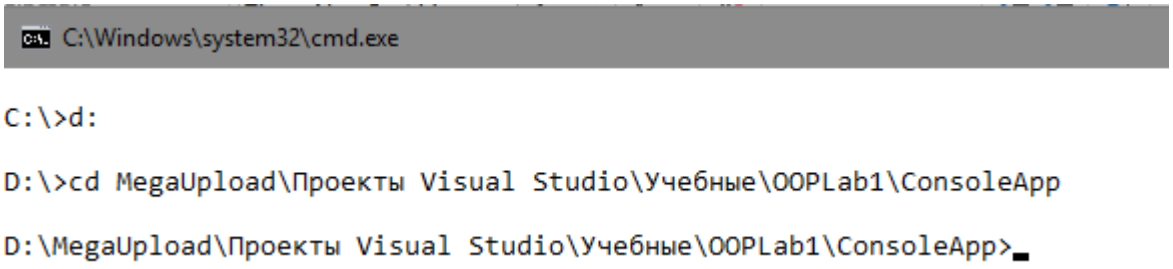
Наприклад, якщо рішення зберігається на диску D у папці MegaUpload\Проекты VisualStudio\Учебные\OOPLab1\:



Тоді потрібно виконати дві команди:

```
d:
cd MegaUpload\Проекты Visual Studio\Учебные\OOPLab1\
```

Результат виконання:



Звертаємо увагу, що потрібно перейти саме до папки з рішенням, а не окремим проектом.

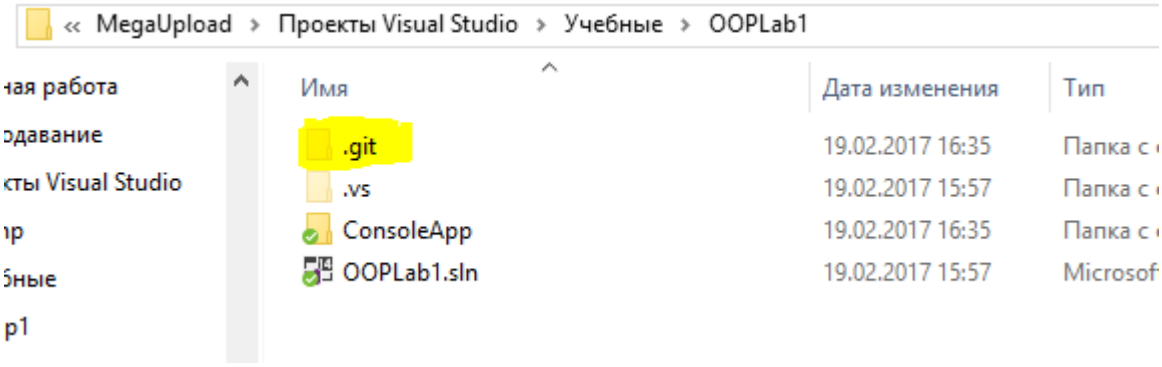
5.4.Виконуємо ініціалізацію локального репозиторію для поточного рішення. Цю дію потрібно виконати один раз для нового рішення:

```
git init
```

Результат роботи команди:

```
C:\Windows\system32\cmd.exe

D:\MegaUpload\Проекты Visual Studio\Учебные\OOPLab1>git init
Initialized empty Git repository in D:/MegaUpload/Проекты Visual Studio/Учебные/OOPLab1/.git/

D:\MegaUpload\Проекты Visual Studio\Учебные\OOPLab1>
```

Повинен з’явитися службовий каталог .git, в якому будуть зберігатися службові файли репозиторію. Перевіримо, чи створився він:



5.5. Переглянемо статус локального репозиторію:

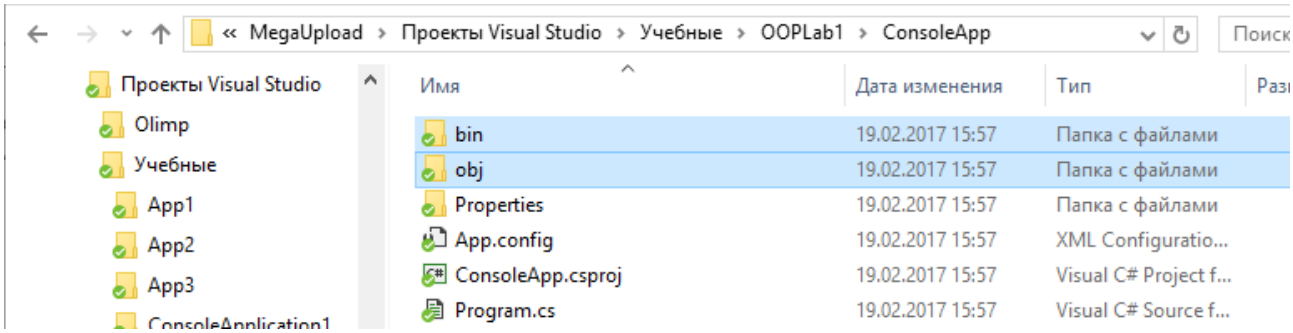
```
git status
```

Результат роботи команди:

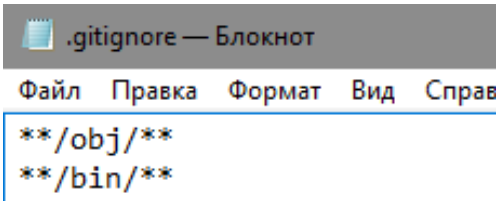
```
C:\Windows\system32\cmd.exe

D:\MegaUpload\Проекты Visual Studio\Учебные\OOPLab1>git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        ConsoleApp/
        OOPLab1.sln

nothing added to commit but untracked files present (use "git add" to track)

D:\MegaUpload\Проекты Visual Studio\Учебные\OOPLab1>
```

5.6. Тепер потрібно додати файли, які будуть індексуватися у git-репозиторії.

Однак, не всі файли потрібно буде завантажувати у віддалений репозиторій, який зберігатиметься на сайті. Зокрема, не потрібно завантажувати файли, отримані в результаті компіляції проекту. Це папки «obj», «bin», які розташовуються у папці проекту:



Створимо файл .gitignore, який буде розміщуватись у папці рішення і міститиме два рядки з назвами каталогів, які не потрібно буде індексувати.



Для того, щоб створити цей файл виконаємо дві команди:

```
echo **/obj/**>> .git ignore
echo **/bin/**>> .git ignore
```

«**» Означає будь-які файли. Тобто з індексування виключаються папки «obj»та «bin», які розміщуються у будь-якому підкаталозі та які містять будь-які файли.

5.7.Тепер можна виконати додавання файлів до індексування. Для цього виконуємо команду:

```
git add *.*
```

5.8. Переглядаємо файли, які було додано до індексування:

```
git status
```

Результат виконання команди:

```
D:\MegaUpload\Проекты Visual Studio\Учебные\OOPLab1>git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitignore
        new file:   ConsoleApp/App.config
        new file:   ConsoleApp/ConsoleApp.csproj
        new file:   ConsoleApp/Program.cs
        new file:   ConsoleApp/Properties/AssemblyInfo.cs
        new file:   OOPLab1.sln
```

5.9. Тепер зафіксуємо поточний стан цих файлів. Операції фіксації змін називається комітом. Для того, щоб зробити коміт потрібно виконати команду:

```
git commit -m "Створено найпростіший консольний додаток"
```

Обов’язково додавайте зрозумілий і розширений опис для комітів.

5.10. Подивимось список комітів з поясненнями виконавши команду:

```
git log
```

Результат виконання команди:

```
D:\MegaUpload\Проекты Visual Studio\Учебные\OOPLab1>git log
commit c21bb7cfcdf9b9d0d841070eeb04ff7953da151b
Author: Andriy Morozov <morozov.andriy@gmail.com>
Date:   Sun Feb 19 17:34:16 2017 +0200

    Створено найпростіший консольний додаток
```

6. Підключення віддаленого репозиторіюта відправка комітів

6.1. Переглядаємо список віддалених репозиторіїв, які вже підключені у вашій системі:

```
git remote
```

Якщо команда нічого не вивела, то це означає, що віддалені репозиторії у вашій системі ще не налаштовувались.

Але якщо команда вивела список репозиторіїв, то потрібно від’єднатися від них.

Наприклад, якщо результат роботи команди такий:

```
D:\MegaUpload\Проекты Visual Studio\Учебные\OOPLab1>git remote
origin
```

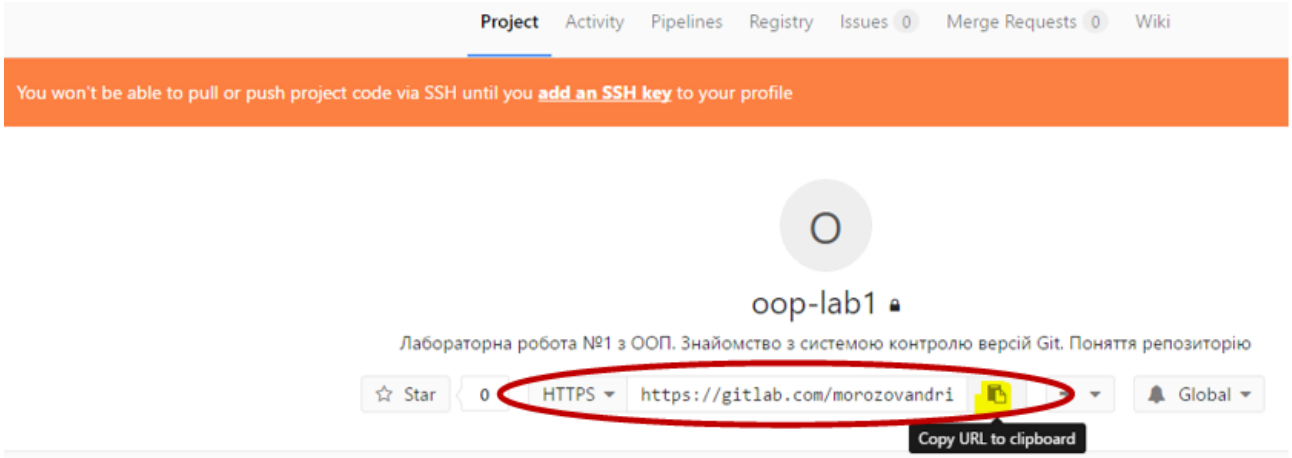
то потрібно виконати команду для кожного віддаленого репозиторію зі списку:

```
git remote remove origin
```

6.2. Додаємо зв’язок з віддаленим репозиторієм на GitLab. Для цього виконуємо команду:

```
git remote add origin https://gitlab.com/pi58_iii/oop-lab1.git
```

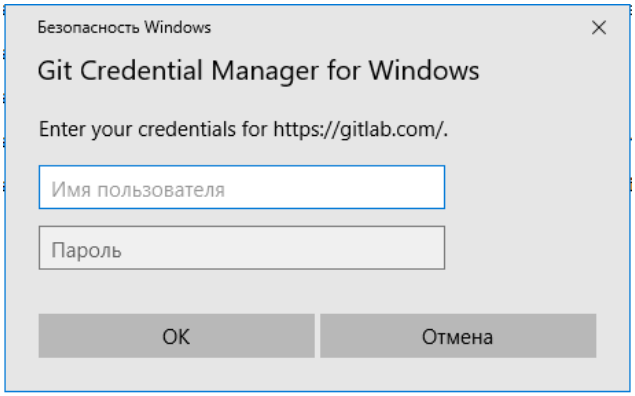
Адресу репозиторію можна подивитися на сайті GitLab:



6.3. Завантажити підготовлені локальні коміти на сервер.

```
git push -u origin master
```

6.4. З’явиться вікно, у якому потрібно буде ввести логін та пароль від сайту GitLab:



6.5. Якщо всі дії виконано правильно, то з’явиться повідомлення про успішне завантаження файлів:

```
D:\MegaUpload\Проекты Visual Studio\Учебные\OOPLab1>git push -u origin master
Counting objects: 10, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (10/10), 3.05 KiB | 0 bytes/s, done.
Total 10 (delta 0), reused 0 (delta 0)
Branch master set up to track remote branch master from origin.
To https://gitlab.com/morozovandriy/oop-lab1.git
 * [new branch]      master -> master
```

Якщо ж логін та пароль введено не правильно, то отримаємо повідомлення «remote: HTTPBasic: Accessdenied»:

```
D:\MegaUpload\Проекты Visual Studio\Учебные\OOPLab1>git push -u origin master
remote: HTTP Basic: Access denied
fatal: Authentication failed for 'https://gitlab.com/morozovandriy/oop-lab1.git/'
```

Якщо виникає помилка виду:

```
C:\Users\ki2_kvv1\Documents\Visual Studio 2013\Projects\OOP1Lab1>git push -u ori
gin master
fatal: unable to access 'https://gitlab.com/ki2_kvv1/oop-lab1.git/': error setti
ng certificate verify locations:
  CAfile: C:/Users/pi54_vpp/AppData/Local/Programs/Git/mingw64/ssl/certs/ca-bund
le.crt
  CApath: none
```

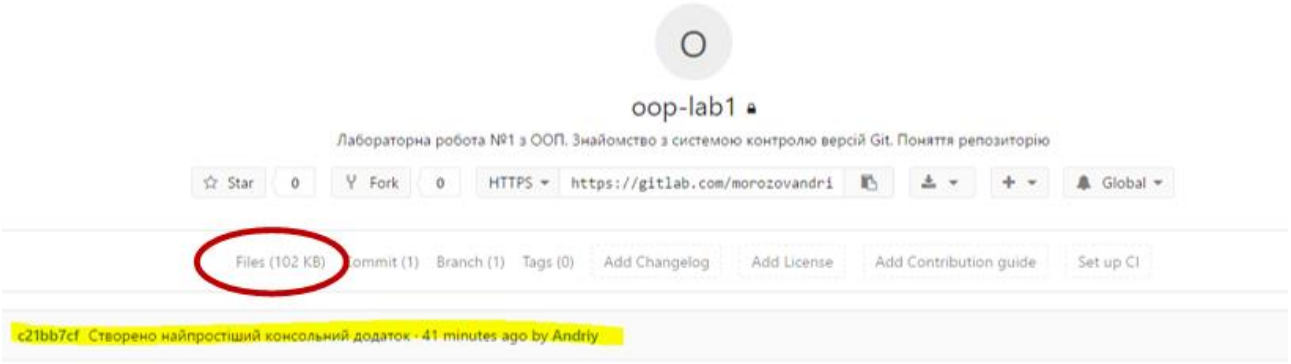
то виконайте команду:

```
git config --system http.sslverify false
```

А потім:

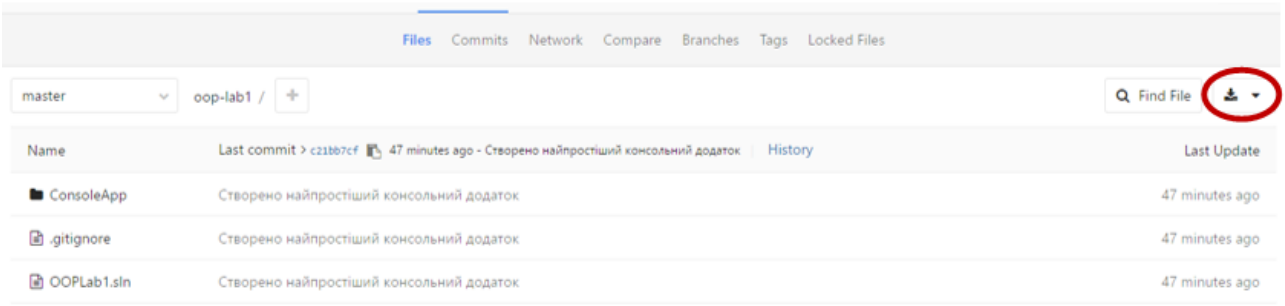
```
git push -u origin master
```

6.6.Переконаємось, що всі файли завантажились на сервер. Для цього зайдемо на сторінку проекту на сайті GitLab:



Натиснувши на посилання «Files» побачимо усі файли, які було завантажено на сервер.

Для того, щоб скачати у вигляді архіву файли репозиторію потрібно натиснути на кнопку «Download»:



7. Додавання змін до рішення та завантаження їх на сервер

7.1. Внесемо зміни у програмний код консольного додатку:

```
namespace ConsoleApp
{
    ссылок: 0
    class Program
    {
        ссылок: 0
        static void Main(string[] args)
        {
            Console.WriteLine("Hello world!");
            Console.ReadKey();
        }
    }
}
```

```
namespace ConsoleApp
{
    ссылок: 0
    class Program
    {
        ссылок: 0
        static void Main(string[] args)
        {
            int x = 5;
            int y = 10;
            Console.WriteLine("5 + 10 = {0}", x + y);
            Console.ReadKey();
        }
    }
}
```

7.2.Для того, щоб їх завантажити на сервер GitLab виконаємо команди:

```
git add *.*
git add-u
git commit -m "Додано підрахунок суми чисел і виведення результату"
```

Перша команда визначає, які з файлів рішення були змінені. Друга команда фіксує ці зміни і готує коміт.

Результат:

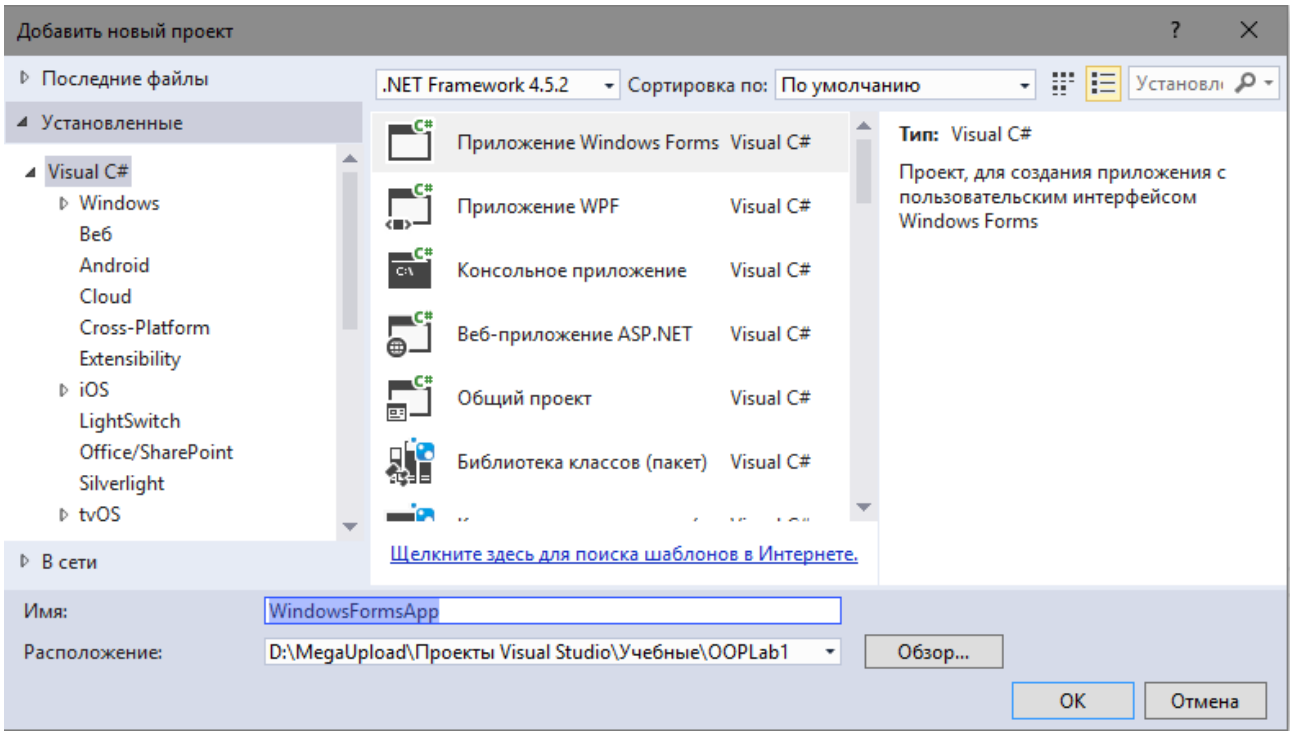
```
D:\MegaUpload\Проекты Visual Studio\Учебные\OOPLab1>git commit -m "Додано підрахунок суми чисел і виведення результату"
[master 4044a56] Додано підрахунок суми чисел і виведення результату
 2 files changed, 5 insertions(+), 1 deletion(-)
```

7.3. Тепер відправимо зміни на сервер командою:

```
git push -u origin master
```

7.4. Перегляньте, чи з’явилися файли на сайті GitLab.

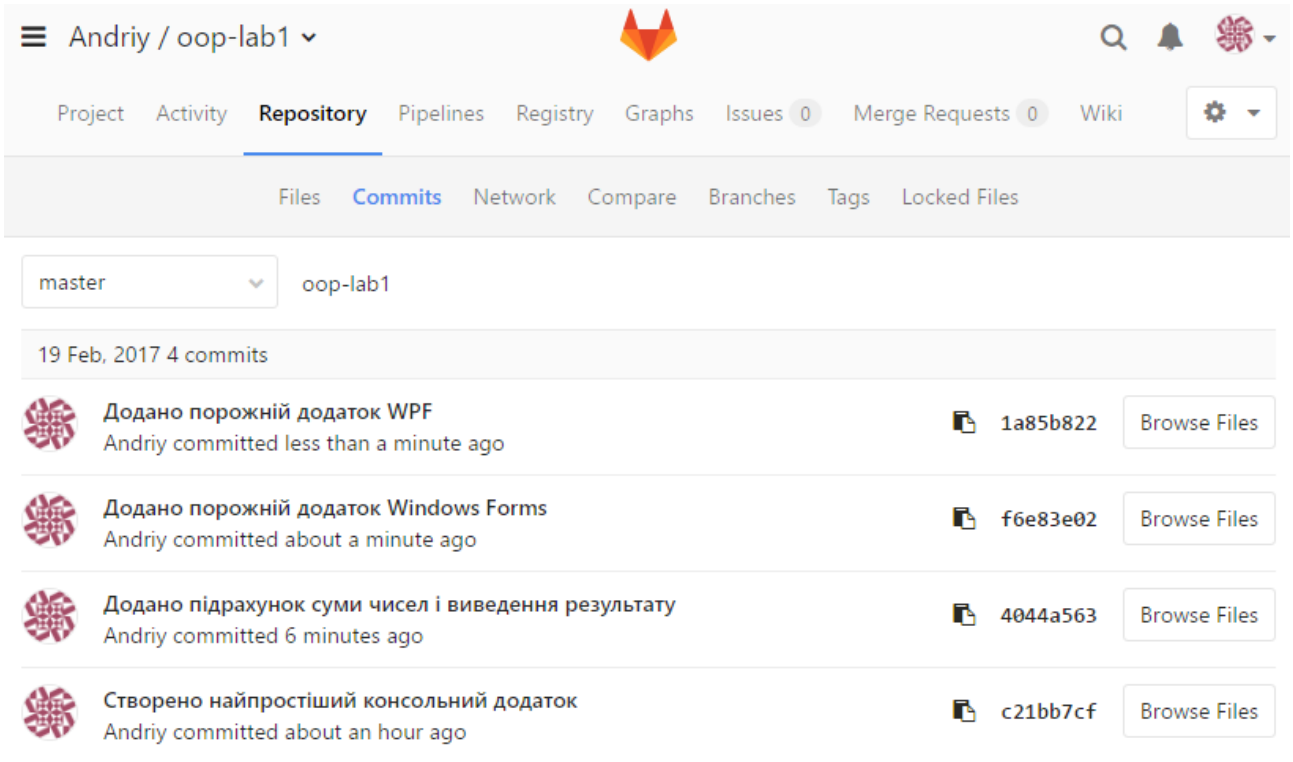
7.5. Створіть ще один проект у рішенні з назвою «WindowsFormsApp»:



7.6. Завантажте зміни на сервер.

7.7. Створіть додаток з назвою «WPFApp»типу «Приложение WPF» і завантажте зміни на сервер.

Тепер в списку комітів на сайті GitLab має бути чотири коміти:



8. Клонування репозиторію

8.1. Перейдемо у кореневий каталог диску C (або іншого):

```
c:
cd \
```

8.2. Виконаємо клонування репозиторію з сервера GitLab:

```
git clone https://gitlab.com/morozovandriy/oop-lab1.git
```

8.3. Перевіряємо, чи з’явилася папка з рішенням на диску C (або іншому).

9. Вивчити усі виконані команди напам’ять (на наступному занятті буде письмова контрольна робота на знання команд та розуміння усіх виконаних кроків).

10. Вивчити матеріал щодо графічного оформлення алгоритмів у вигляді блок-схем:
https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA-%D1%81%D1%85%D0%B5%D0%BC%D0%B0

Намалювати блок-схему виконаної роботи.

11. Оформити звіт з лабораторної роботи