Лабораторна робота №8
Робота з GIT та GITHUB

I. Підготовка до лабораторної роботи

У попередніх роботах ми навчилися створювати локальний репозиторій, а також віддалений безпосередньо на сервісі GitHub. Настав час навчитися синхронізувати локальний та віддалений репозиторії.

Для виконання даної роботи необхідно мати встановлений локально git і аккаунт на GitHub.

II.Теоретичні відомості

Git – це інструмент, що дозволяє реалізувати розподілену систему контролю версій.

GitHub - сервіс онлайн-хостингу репозиторіїв, що має всі функції розподіленого контролю версій і функціональність управління вихідним кодом - все, що підтримує Git і навіть більше. Також GitHub може похвалитися контролем доступу, багтрекінгом, управлінням завданнями та вікі для кожного проекту.

У попередніх роботах ми навчилися створювати локальний репозиторій, а також віддалений безпосередньо на сервісі GitHub. Настав час навчитися синхронізувати локальний та віддалений репозиторії.

III. Завдання на лабораторну роботу

1. Створення нового репозиторія на GitHub.
2. Клонування віддаленого репозиторія.
3. Створення і зміни файлу.
4. Відправлення зміни у віддалений репозиторій.
5. Зміни та відправка файлів у віддалений репозиторій.
6. Ігнорування файлів.

Створення нового репозиторія на GitHub

Для того щоб створити новий репозиторій (рис. 3.1), необхідно виконати такі дії:

- зайти в свій аккаунт на GitHub і натиснути на кнопку New repository;
- написати назву і опис робочого репозиторія;
- вибрати Initialize this repository with a README;
- створити репозиторій.

Клонування віддаленого репозиторія

Для того щоб проводити зміни локально, необхідно отримати дану версію репозиторію.

Для цього необхідно виконати такі дії:

- натиснути кнопку Clone or download в правому верхньому кутку;
- вибрати Clone with HTTPS і скопіювати посилання (рис. 3.2);

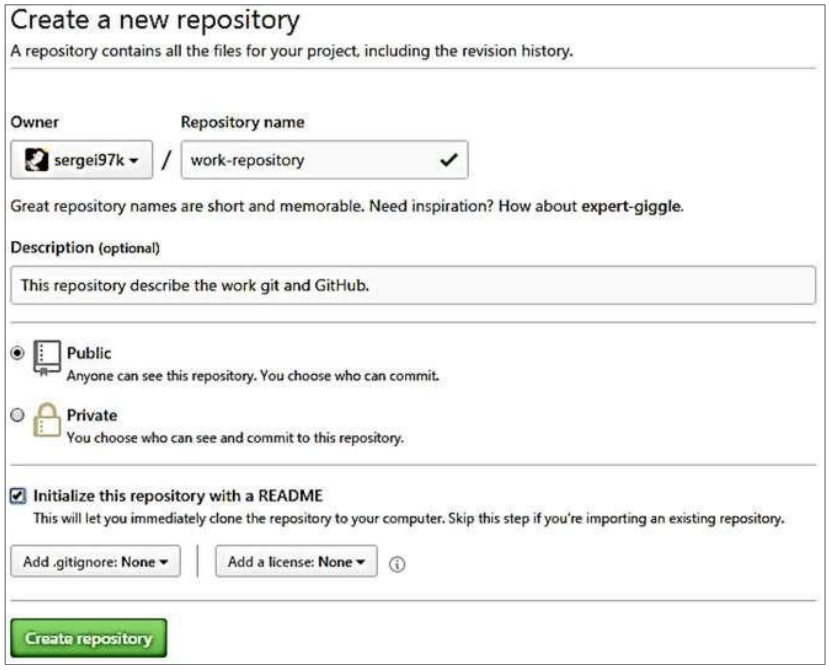

Рисунок 3.1 – Створення нового репозиторію

Рисунок 3.2 – Копіювання посилання

- запустити консольну версію git – git-bash і вибрати директорію, в яку хочемо зберегти робочий репозиторій;
- написати команду git clone і вставити збережене посилання (рис. 3.3);

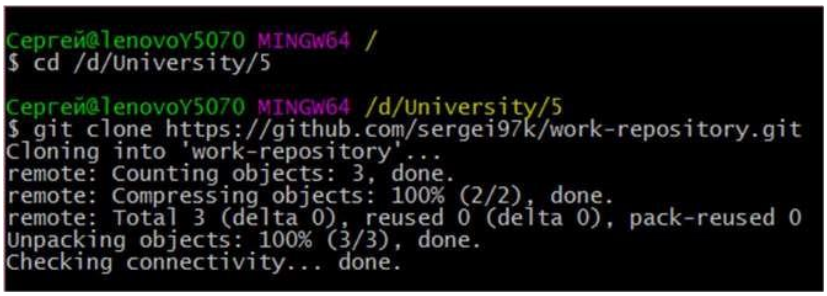

Рисунок 3.3 – Виконання команди git clone

- зайти у вибрану вище директорію і переконатися, що в ній з’явилася папка з назвою робочого сховища.

Створення і зміни файлу

Робоче сховище розміщено локально і можна робити перші зміни.

Для цього необхідно створити файл index.html і виконати наступні дії:

– зайти в робочу папку і створити файл з таким змістом, як показано на рис. 3.4;

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>work-repository</title>
</head>
<body>
  <div>
    Department «Information and intellectual property»
    faculty «Computer and information technology» National technical university «Kharkov polytechnical institute»
  </div>
</body>
</html>
```

Рисунок 3.4 – Зміст файла index.html

– зберегти зміни (рис. 3.5).

```
Сергей@lenovoY5070 MINGW64 /d/University/5
$ cd /d/University/5/work-repository

Сергей@lenovoY5070 MINGW64 /d/University/5/work-repository (master)
$ git add .

Сергей@lenovoY5070 MINGW64 /d/University/5/work-repository (master)
$ git commit -m "added index.html"
[master ab0f239] added index.html
 1 file changed, 13 insertions(+)
 create mode 100644 index.html
```

Рисунок 3.5 – Збереження змін

Відправлення змін у віддалений репозиторій

Файл index.html знаходиться локально, але для того, щоб інші користувачі змогли побачити актуальні зміни, необхідно відправити їх в віддалений репозиторій на GitHub.

Для цього необхідно виконати такі дії:

– переконатися, що були зроблені всі необхідні зміни – команда git log (рис. 3.6);

```
$ git log
commit ab0f2392c3e5ce4e5539106cdb7f00ef6bb8f664
Author: Sergei Kononov <sergei97k@gmail.com>
Date:   Sun Oct 2 18:04:36 2016 +0300

    added index.html

commit 6041aa77d04d6ff2842d8caae82cf14ea93c6db8
Author: Sergei Kononov <sergei97k@gmail.com>
Date:   Sun Oct 2 17:32:47 2016 +0300

    Initial commit
```

Рисунок 3.6 – Виконання команди git log

– для відправки змін у віддалений репозиторій необхідно написати команду git push (рис. 3.7).

Примітка: якщо потрібно відправити зміни на конкретну гілку, то не обхідно використовувати команду git push з прапором -u і назвою гілки (наприклад: git push -u origin master);

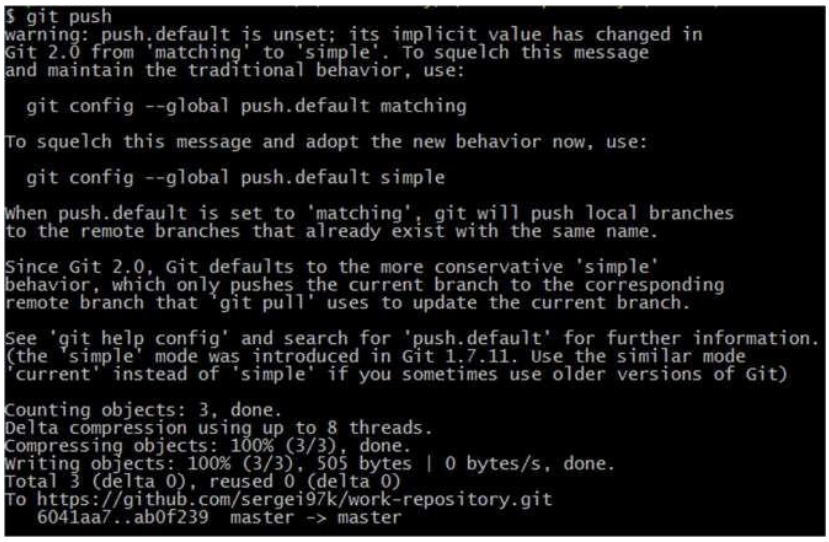

Рисунок 3.7 – Виконання команди git push

– ввести власний логін і пароль на GitHub, після чого віддалений репозиторій буде змінений;
– зайти до головного репозиторію на GitHub, щоб побачити чи з'явилися зміни в віддаленому репозиторії (рис. 3.8).

Рисунок 3.8 – Перегляд змін в репозиторії

Зміни та відправка файлів у віддалений репозиторій

Потрібно додати зміни в файл index.html на сторінці сховища. При натисканні на файл, можна побачити кнопку history. Тут зберігаються зміни, що стосуються конкретного файлу, а не всього сховища. Адже не в кожному commit'і можуть змінюватися усі файли сховища.

Для додавання змін до файлу необхідно зробити такі дії:

– натиснути на іконку олівця в правому верхньому куті, для того щоб зробити зміни у файлі;
– додати новий рядок в файл (рис. 3.9);

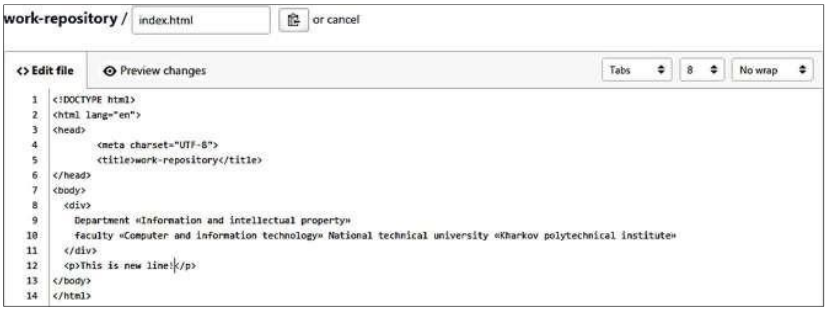

Рисунок 3.9 – Зміни до файлу

– ввести назву commit'у і його опис;

– натиснути кнопку Commit changes (рис. 3.10);

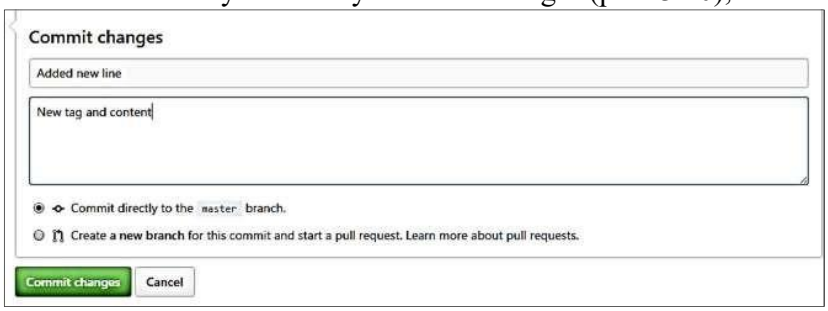

Рисунок 3.10 – Введення назви commit'а і його опису

– переконатися, що файл у віддаленому репозиторії змінився;
– перейти на файл і натиснути кнопку blame. Тут можна побачити хто і які зміни здійснював в даному файлі (рис. 3.11);

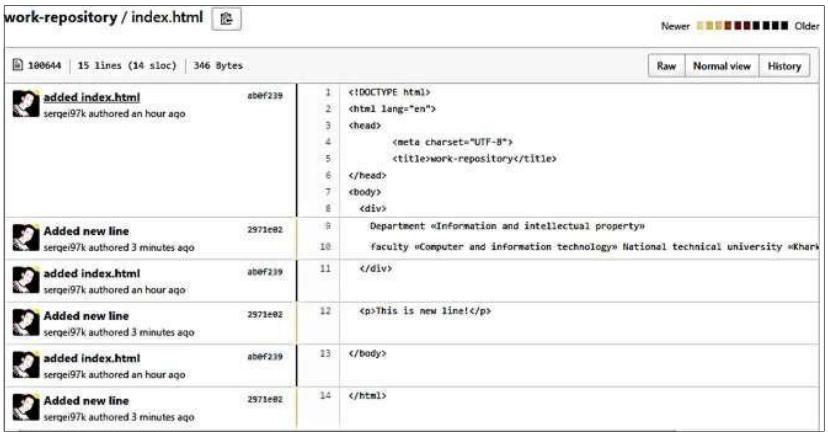

Рисунок 3.11 – Візуалізація змін в файлі

– написати в командному рядку git pull, для того щоб злити дані зміни в свій локальний репозиторій (рис. 3.12);

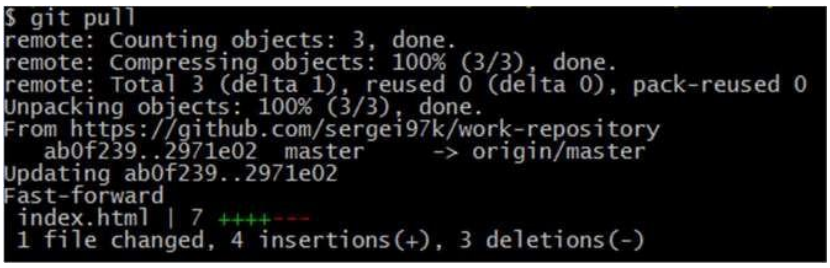

Рисунок 3.12 – Виконання команди git pull

– зайти в текстовий редактор і переконатися, що додано новий рядок (рис. 3.13). Таким чином відбувається синхронізація вашого локального сховища із розміщеним на GitHub.

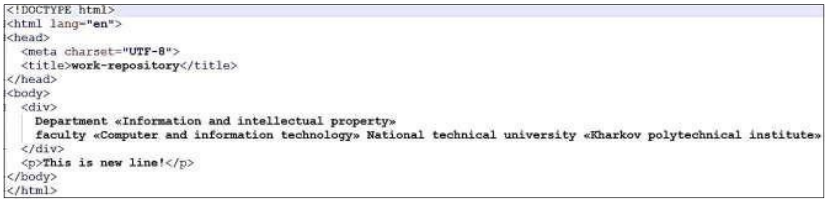

Рисунок 3.13 – Візуалізація змін в файлі

Ігнорування файлів

Ігноровані файли не обов’язково розміщувати у віддалений репозиторій. По суті ці файли необхідні в процесі розробки, але можуть не знадобитися в основний гілці.

Для ігнорування файлів необхідно виконати такі дії:

– створити файл з розширенням .gitignore. Це можна зробити при створенні сховища або локально (рис. 3.14);

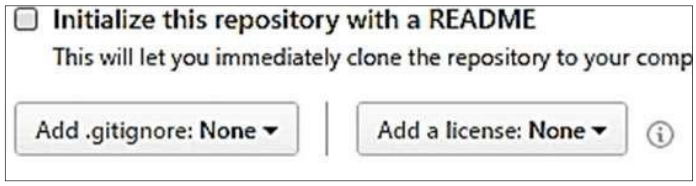

Рисунок 3.14 – Створення файлу

– створити файл .gitignore локально і текстовий файл, який треба ігнорувати (рис. 3.15);

Рисунок 3.15 – Створення файла .gitignore

– відкрити файл .gitignore і записати назву файлу, який необхідно ігнорувати (рис. 3.16);

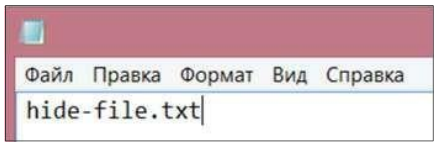

Рисунок 3.16 – Запис назви файлу для ігнорування

– зайти в командний рядок і переконатися, що файли змінилися (рис. 3.17), використовуючи команду git status;

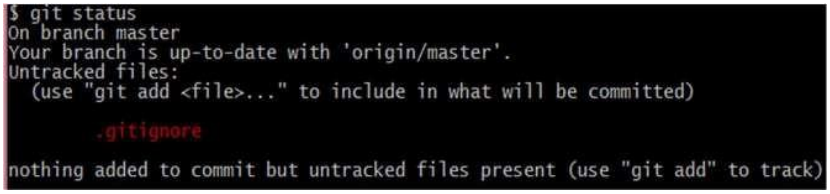

Рисунок 3.17 – Використання команди git status

– зберегти всі зміни командою git add (рис. 3.18).

Рисунок 3.18 – Використання команди git add

– відправити зміни у віддалений репозиторій командою git push (рис. 3.19);

```
$ git push
warning: push.default is unset; its implicit value has changed in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the traditional behavior, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

When push.default is set to 'matching', git will push local branches
to the remote branches that already exist with the same name.

Since Git 2.0, Git defaults to the more conservative 'simple'
behavior, which only pushes the current branch to the corresponding
remote branch that 'git pull' uses to update the current branch.

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 331 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/sergei97k/work-repository.git
   2971e02..0f07db0  master -> master
```

Рисунок 3.19 – Використання команди git push

– переконатися, що у віддаленому репозиторії немає файлу hide-file.txt (рис. 3.20), хоча він є локально.

Це зменшує розмір вашого сховища і звільняє його від непотрібних файлів.

Рисунок 3.20 – Результат роботи по ігнорування файлу

IV. Контрольні питання

1. Як створити копію віддаленого репозиторія у локальній директорії?
2. Як перевірити наявність змін у файлі?
3. Яким чином відбувається відправлення змін у віддалений репозиторій?
4. Як відправити зміни на конкретну гілку у віддаленому репозиторії?
5. Яким чином можна відокремити файли у локальному репозиторії, які не підлягають синхронізації із віддаленим репозиторієм?
6. Яку функцію у репозиторії відіграє файл .gitignore?