# Лабораторна робота №2. Система контролю версій Git

Тема: Система контролю версій Git

Мета: набути навичок використання системи контролю версій Git і репозиторію у процесі створення програмного забезпечення.

Час: 2 год.

**Короткі теоретичні відомості**

Робота з Git через термінал – обов’язкова частина практики кожного сучасного фронтенд-фахівця. Але для початківця робота з Git може здаватися складною.

Робота з системою контролю версій Git починається з її налаштування. Git можна налаштувати лише один раз, а зміни вносити лише за необхідності.

Налаштування імені користувача, від якого будуть йти комміти:

git config --global user.name "User Name"

Встановлення адреси електронної пошти здійснюється такою командою (але ця електронна пошта повинна співпадати з тією, за якою зареєстрований акаунт на гітхабі):

git config --global user.email mail@gmail.com

Встановлення текстового редактора, в якому будуть відкриватися файли для вирішення конфліктів:

git config --global core.editor editor

Список всіх налаштувань можна переглянути за допомогою команди

git config --list

Клонування репозиторію

Для клонування репозиторію необхідно виконати команду

git clone address

Замість address вводиться адреса. Репозиторій клонується у поточну вибрану папку в консолі.

Робота зі змінами починається з отримання останньої версії віддаленого репозиторію. Отримати останню версію можна за допомогою команди git pull. Виклик цієї команди зітре всі незафіксовані зміни.

Після внесення будь-яких змін у проект можна переглянути статус файлів за допомогою команди git status. Будуть показані файли, у яких були зміни, видалені і нові, які потребують додавання.

Для того, щоб додати відслідковування нових файлів, необхідно використати команду

git add filename filename

для додавання кількох файлів за іменем.

У випадку, коли файлів для додавання багато, можна використати команду

git add .

Ця команда додає відслідковування файлів з поточного каталогу.

Команда

git add -A додає ще й видалені файли, не лише з поточного каталогу, а й з всього репозиторію.

Для видалення файлів існує команда

git rm filename

Ця команда видаляє файли за їх іменами.

Після того, як додані всі нові і видалені старі файли, можна зафіксувати зміни. Фіксація змін (комміт) дуже важлива, оскільки до виконання цієї команди жодні локальні зміни записані не будуть.

Для того, щоб додати коміт, необхідно записати команду:

git commit -m “commentar”

Необхідно правильно розбивати зміни на коміти і давати повні коментарі до змін. Якщо потрібно швидко відмінити зміни, необхідно виконати команду

git reset

Ця команда відміняє всі незафіксовані зміни.

За замовчуванням, ця команда видаляє лише з індекса.

Команда

git reset --hard

видаляє безповоротньо всі незафіксовані зміни з локального репозиторію і з індексу.

Всі вищезазначені дії відбуваються в локальній копії репозиторію, тому цю копію слід відправити на сервер. Для цього використовується команда

git push

Ця команда відправляє всі зафіксовані зміни на віддалений репозиторій.

Робота з гілками

Працюючи з Git, доводиться постійно створювати гілки і переміщатися по ним. Команда

git checkout -b branch-name

створює гілку з вказаним іменем і автоматично перемкнеться на неї.

Після створення гілку можна відправляти на сервер за допомогою команди

git push origin branch-name

Аналогічно можна забрати собі на комп’ютер гілку з віддаленого репозиторію за допомогою команди

git checkout origin/branch-name -b branch-name

Для того, щоб не зберігати назви гілок у пам’яті і не шукати їх, існують спеціальні команди: для перегляду гілок локального репозиторію git branch і всіх гілок віддаленого репозиторію git branch -r.

Переключитися на будь-яку локальну гілку можна за допомогою команди git checkout branch-name.

Після роботи в репозиторії можуть залишитися різні непотрібні, невідслідковані файли і інше сміття. Для видалення всього зайвого слід виконати команду

git clean -f -d

**Хід роботи**

1. Отримати інструкції від викладача.
2. Вибрати та погодити з викладачем тему індивідуального робочого проекту.
3. Ознайомитися з основними утилітами командного рядка для навігації.
4. На одному з дисків вашого комп´ютера створити папку, названу ICS-1234567, де 1234567 – номер перепустки.
5. Перевірити наявність на комп´ютері системи контролю версій git:

* запустити режим командного рядка: cmd;
* у режимі командного рядка набрати команду git - - version. Якщо цієї програми немає, то встановити git з веб-сторінки [5] і встановити параметри за замовчуванням.

1. Створити та персоніфікувати локальний git-репозиторій (інструкція [9]):

* ініціалізувати репозиторій: git init;
* задати ім´я власника та його електронну адресу: git config.

1. Скопіювати файл з індивідуальним завданням у папку проекта:
2. Створити у папці проекту файл з назвою README.txt і за допомогою мови розмітки markdown записати в нього такі дані:

* відомості про студента (прізвище та ініціали, група, факультет);
* назву і короткі відомості про проект.

1. Зареєструватися на GitHub (інструкція [6]).
2. Синхронізувати github-репозиторій у відповідності з локальним (інструкція [7]).

**Домашнє завдання**

1. Вибрати тему проекту і записати свої дані поряд з темою.
2. Завантажити та встановити на свій комп´ютер систему контролю версій git зі сторінки [5].
3. Завантажити та встановити навігатор Anaconda.
4. Налаштувати плагіни VSCode для розробки програм на Python.
5. Заповнити повністю свій профіль на GitHub.
6. Зробити копію свого github-репозиторію на локальний комп´ютер [9].

**Контрольні питання**

1. Що таке репозиторій?
2. Як використовується репозиторій у роботі програміста?
3. Що таке система контролю версій?
4. Для чого використовується система контролю версій?
5. Як створити репозиторій Git?
6. Як синхронізувати локальний і он-лайн репозиторії Git?
7. Як створити відгалуження версій?
8. Як створити коміт?

**Список рекомендованих джерел**

1. Вступ - Про систему контролю версій. URL: <https://git-scm.com/book/uk/v2/%D0%92%D1%81%D1%82%D1%83%D0%BF-%D0%9F%D1%80%D0%BE-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%83-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8E-%D0%B2%D0%B5%D1%80%D1%81%D1%96%D0%B9>
2. Пугачов Р.В. Системи контролю версіями: навч.-метод. посібник / Пугачов Р.В., Любченко Н.Ю. Соболь М.О.– Харків : НТУ "ХПІ", 2019. – 130 с.. URL: <http://repository.kpi.kharkov.ua/bitstream/KhPI-Press/54660/1/Book_2019_Puhachov_Systemy.pdf>
3. Система контролю версій (Git). URL: <https://www.youtube.com/playlist?list=PL9mn2EBC_SSyu6I4DQ9-r1vm_CX4jaPWf>
4. Системи контролю версій на прикладі Git. URL: <https://dut.edu.ua/ua/news-1-626-9170-sistemi-kontrolyu-versiy-na-prikladi-git_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy>
5. Що таке GitHub і навіщо він потрібен розробнику. URL: <https://foxminded.ua/shho-take-github-i-navishho-vin-potriben-rozrobniku/>