**Лабораторна робота**

*Тема роботи:* GitHub

*Мета роботи:* ознайомитись з загальними підходами і принципами роботи СКВ Git, отримати

практичні навики застосування основних команд Git.

*При підготовці до лабораторної роботи:*

Відпрацювати матеріал лекції, вказану літературу та продумати тему своєї розробки.

*При виконанні лабораторної роботи:*

1. Зареєструйтеся на сервісі спільної розробки GitHub
2. Завантажте клієнт зайдіть до нього та створить там проект, та синхронізуй його з сайтом
3. Внесіть зміни до проекту. Відобразіть зміни у проекті на сайті
4. Синхронізуйтеся між системою та локальним комп’ютером
5. Створіть порожній репозиторій на сервісі github.com.
6. Виконайте інструкцію по створенню першого коммітов на сторінці сховища, створеного на предидещем кроці.
7. Створіть файл hello\_world.html в локальній копії сховища (який повинен був з'явитися на кроці 2).
8. Реалізуйте програму Hello world мовою html використовуючи поганий стиль коду. Наприклад з помилками в тегах .Додайте цей файл в локальну копію сховища.
9. Закоммітьте зміни з осмисленим повідомленням.
10. Змініть вихідний код так, щоб текст був з практичної роботи 3.
11. Закоммітьте нову версію програми. Чому не треба додавати файл повторно git add?
12. Запуште зміни у віддаленому репозиторій.
13. Перевірте, що історія коммітов доступна у віддаленому репозиторії.
14. У локальної копії сховища створіть локальну гілку patch1. Внесіть зміни в гілці patch1 щодо виправлення коду і позбавлення від using namespace std
15. commit, push локальну гілку в віддалений репозиторій. Перевірте, що гілка patch1 доступна у віддаленому репозиторії.
16. Створіть pull-request patch1 -> master.
17. У локальної копії в гілці patch1 додайте в вихідний код коментарі. commit, push.
18. У віддалений репозиторії виконайте злиття PR patch1 -> master і видаліть гілку patch1 в віддаленому репозиторії.
19. Локально виконайте pull.
20. За допомогою команди git log перегляньте історію в локальній версії гілки master.
21. Видаліть локальну гілку patch1. Part III Note: Працювати продовжуйте з тими ж репоззіторіямі, що і в першій частині завдання. 1. Створіть нову локальну гілку patch2.
22. Змініть code style за допомогою утиліти clang-format. Наприклад, використовуючи опцію -style = Mozilla. commit, push, створіть pull-request patch2 -> master.
23. У гілці master в віддаленому репозиторії змініть коментарі, наприклад, розставте розділові знаки, переведіть коментарі на іншу мову.
24. Переконайтеся, що в pull-request з'явилися конфліктні. (Для цього локально виконайте pull + rebase (точну послідовність команд, слід дізнатися самостійно)) Виправте конфлікти.
25. Зробіть force push в гілку patch2 Переконайтесь, що в pull-request пропали конфліктні.
26. Вмержіте pull-request patch2 -> master.

*Контрольні питання*

*1. Що таке Git?*

Git — розподілена система керування версіями файлів та спільної роботи.

*2. Які види систем керування версіями Ви знаєте?*

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). Приклад: Subversion, Concurrent Versions System, Team Foundation Server.

Розподілена система контролю версії (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. Приклад: Git, Mercurial, Bazaar, Monotone, Codeville, BitKeeper.

*3. Які можливості надають системи контролю версій?*

* Можливість створення різних варіантів одного документу;
* Документування всіх змін (коли ким було змінено/додано, хто який рядок змінив);
* Реалізує функцію контролю доступу користувачів до файлів. Є можливість його обмеження;
* Дозволяє створювати документацію проєкту з поетапним записом змін в залежності від версії;
* Дозволяє давати пояснення до змін та документувати їх;

*4. Які системи контролю версій називають централізованими?*

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).

*5. Які системи контролю версій називають локальними?*

Використання лише в локальному середовищі, всі розробники повинні використовувати той же самий комп'ютер.

*6. Які системи контролю версій називають розподіленими?*

В розподілених системах управління версіями кожен розробник працює безпосередньо зі своїми локальними копіями кодів. Зміни розподіляються між сховищами окремим кроком.

*7. Які переваги централізованих систем контролю версій?*

* загальна нумерація версій;
* дані знаходяться на одному сервері;
* можлива реалізація функції блокування файлів;
* можливість керування доступом до файлів;

*8. Які недоліки централізованих систем контролю версій?*

* оновлення робочої копії та збереження змін можливе лише за наявності мережевого з'єднання з основним сервером.

*9. Які переваги локальних систем контролю версій?*

* проста у використанні і добре підходить для ознайомлення з принципами роботи систем контролю версій.
* Добре підходить для резервного копіювання окремих файлів, які не потребують частої зміни групою користувачів.
* Широко поширена і предустановленна в більшості вільно розповсюджуваних операційних системах.

*10. Які недоліки локальних систем контролю версій?*

* Чи відстежує зміни лише окремих файлів, що не дозволяє використовувати її для управління версіями великих проектів.
* Не дозволяє одночасно вносити зміни в один і той же файл кількома користувачами.
* Низька функціональність, в порівнянні з сучасними системами контролю версій.

*11. Які переваги розподілених систем контролю версій?*

* Кожний з розробників працює зі своїм власним репозитарієм;
* Рішення щодо злиття гілок приймається керівником проєкту;
* Немає потреби в мережевому з'єднанні;

*12. Які недоліки розподілених систем контролю версій?*

* не можливо контролювати доступ до файлів;
* відсутня загальна нумерація версій файлів;
* потрібно значно більше дискового простору на сервері для зберігання всіх гілок як глобальних, так й локальних;
* відсутня можливість блокування файлів.

*13. В якому статусі можуть перебувати файли при роботі з Git?*

Git має три основних стани, в яких можуть перебувати ваші файли: збережений у коміті (commited), змінений (modified) та індексований (staged): Збережений у коміті означає, що дані безпечно збережено в локальній базі даних.

*14. Для чого використовується команда add?*

Додає змінені та нові файли

*15. Для чого використовується команда status?*

Файл є новим і git ще не знає, потрібно слідкувати за його змінами, чи краще просто ігнорувати його

*16. Для чого використовується команда commit?*

Створення нової версії, фіксація змін. У деяких СУВ (Subversion) - нова версія автоматично переноситься до сховища документів.

*17. Для чого використовується команда init?*

Ця команда створює в поточному каталозі новий підкаталог, що містить всі необхідні файли репозиторію - структуру Git репозиторію.

*18. Що необхідно писати в коментарях комітів?*

Описувати, що було змінено, де змінено, чому змінено.

*19. Як переглянути список комітів?*

Скористуватися командою git log.

*20. Для чого використовується команда checkout?*

Вилучення документа зі сховища та створення робочої копії.