Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

**Лабораторна робота № 1**

з дисципліни «Основи клієнтської розробки»

Тема: «Системи контролю версій. Git. Проєкт. Структура проєкту»

|  |  |
| --- | --- |
| Виконав:  студент групи -  Коломойченко Катерина  Дата здачі \_\_\_\_\_\_\_\_\_\_\_\_  Захищено з балом \_\_\_\_\_\_\_\_\_\_\_ | Перевірила:  ст. вик. кафедри ІСТ  Хмелюк Марина Сергіївна |

**Обрана тематика:** 22)Фармацевтика

**Завдання:**

1. Встановити Git на комп’ютер. Створити каталог для проєкту. Ініціалізувати Git директорію у своєму проєкті (git init).
2. Створити 4 сторінки відповідно до обраної тематики з розширенням .html. Перша сторінка має назву - index.html. В вказати автора документа. Кожна сторінка повинна мати назву . Кожна сторінка повинна мати заголовок <head></head>.
3. Додати створені сторінки під контроль Git на комп’ютері.
4. Зареєструватись на GitHub. Створити репозиторій. Завантажити проєкт на віддалений репозиторій (на GitHub).

**Теоретичні відомості:**

Системи контролю версіями (Version Control Systems, VCS) важливий аспект розробки сучасного ПЗ.

VCS надає такі можливості:

- підтримку зберігання файлів у репозиторії;

- підтримку історії версій файлів у репозиторії;

- знаходження конфліктів при зміні вихідного коду та забезпечення

синхронізації при роботі в багатокористувацькому середовищі розробки;

- відстеження змін авторів.

**Git** – це розподілена система контролю версій, яка виникла внаслідок розробки ядра Linux.

Git моделює свої дані як набір знімків, зберігаючи посилання на них.

Git має три основні стани, в яких можуть знаходитися ваші файли: зафіксовано, змінено і підготовлено.

- **Зафіксовано (Підтверджено):** дані безпечно зберігаються локально.

- **Змінено:** ще не підтверджено.

- **Підготовлено:** позначений для переходу до наступного коміту.

Ці стани приводять до трьох розділів:

**Робочий каталог:** це копія версії проекту.

**Staging Area:** простий файл, в якому зберігається інформація про те, що буде у наступному коміті.

**Каталог Git:** зберігає метадані та об'єкти бази даних для вашого проекту.

**Структура директорії .git/**

робота з Git буде починається з того, що потрібно проініціалізувати Git директорію у своєму проєкті. Це робиться за допомогою команди:

**git init**

Її необхідно ввести в корені вашого проєкту. Це створить у поточному каталозі новий підкаталог .git.

У цій директорії буде вся конфігурація Git та історія проєкту. За бажанням можна редагувати ці файли вручну, вносячи необхідні зміни в історію проекту.

1. **config**

Цей файл містить налаштування Git репозиторію. тут можна зберігати email та ім'я користувача.

2. **description**

Цей файл призначений для GitWeb - це веб-інтерфейс, написаний для перегляду Git-репозиторія використовуючи веб-браузер і містить інформацію про проект (назва проекту та його опис).

3. **hooks**

У цьому каталозі Git надає набір скриптів, які можуть автоматично запускатись під час виконання git команд. Наприклад, можна написати скрипт, який редагуватиме повідомлення комміту відповідно до ваших вимог.

4. **info**

Каталог info містить файл exclude, в якому можна вказувати будь-які файли, і Git не додаватиме їх у свою історію.

5. **refs**

Каталог refs зберігає копію посилань на об'єкти коммітів у локальних і віддалених гілках.

6. **logs**

Каталог logs зберігає історію проекту для всіх гілок у вашому проекті.

7. **objects**

Каталог objects зберігає у собі BLOB об'єкти, кожен із яких проіндексований унікальним SHA (Secure Hash Algorithm).

8. **index**

Проміжна область з метаданими, такими як тимчасові мітки, імена файлів, а також файли SHA, які вже упаковані Git. У цю область потрапляють файли, над якими ви працювали, під час виконання команди git add (додавання файлів).

9. **head**

Файл містить посилання на поточну гілку, в якій ви працюєте.

10. **orig\_head**

Щоразу під час злиття до цього файлу потрапляє SHA гілка, з якою проводилося злиття.

11. **fetch\_head**

Файл зберігає посилання у вигляді SHA на гілки, які брали участь у git fetch (завантаження комітів, файлів, посилань з віддаленого репозиторія в локальний)

12. **merge\_head**

Файл зберігає посилання у вигляді SHA на гілки, які брали участь у git merge (злиття)

13. **commit\_editmsg**

Файл містить останнє введене вами повідомлення комміту.

Робота над проектом з використанням СКВ обмежена певними діями:

1. Внести зміни до проекту;

2. Додати зміни до індексу (staging area) - git add (ви повідомляєте Git, які саме зміни повинні бути занесені в історію);

3. Закомітити зміни - git commit (зберегти зміни до історії проекту);

4. Запушити - git push (надіслати результати роботи на віддалений сервер, щоб інші розробники теж мали до них доступ).

Команда **git status** відображає стан директорії та індексу (staging area). Це дозволяє визначити, які файли в проекті відстежуються Git, а також які зміни будуть включені до наступного коміту.

**git remote add <remote\_name> <remote\_repo\_url>** - команда створює віддалене сховище з посиланням на локальний репозиторій. З цього моменту можна звернутися до видаленого репозиторію через це посилання.

**Гілка в Git** - це простий переміщуваний покажчик на один з коммітів. За замовчуванням, ім'я основної гілки в Git — master. Як тільки ви почнете створювати комміти, гілка master завжди вказуватиме на останній комміт.

Для перегляду історії в Git є ряд команд:

**git log** - призначена для відображення всієї історії вашого проєкту, Дозволяє дізнатися, які зміни ви внесли раніше.

**git show** - використовується для відображення повної інформації про будьякий об’єкт у Git, комміт або гілку. За замовчуванням git show відображає інформацію коміту, на який у даний момент часу вказує HEAD.

**git reflog** - виводить упорядкований список комітів, на який вказує HEAD (відображає історію всіх ваших переміщень по проєкту). Основна перевага цієї команди полягає в тому, що якщо ви випадково видалили частину історії або відкотилися назад, ви зможете переглянути момент втрати потрібної вам інформації та відкотитись назад. git reflog зберігає свою інформацію на вашій машині окремо від комітів, тому при видаленні будь-чого в історії можна знайти її в git reflog.

**git reset** - дозволяє відкотити проєкт до визначеної точки. Цю команду можна використовувати з трьома параметрами.

**git reset --soft** - вміст вашого індексу, а також робочої директорії, залишаються незмінними. якщо ми відкотимося назад на пару комітів, ми змінимо посилання вказівника HEAD на вказаний комміт і всі зміни, які були до цього внесені, покажуться в індексі.

**git reset --mixed** - змінимо посилання вказівника HEAD, але всі попередні зміни в індекс не попадуть, а будуть відображатися як не занесені в індекс. Це дає можливість внести в індекс тільки ті зміни, які нам необхідні.

**git reset --hard** - знову змінимо посилання вказівника HEAD, але всі попередні зміни не попадуть ні в індекс, ні в зону шуканих файлів, ми повністю зітремо всі зміни, які внесли раніше.

**Гілкування в Git**

Гілкування означає, що у вас є можливість працювати над різними версіями проєкту.

Команда **git pull** використовується для вилучення та завантаження вмісту з віддаленого репозиторію та негайного оновлення локального репозиторію цим вмістом.

Git за замовчуванням під час ініціалізації створює гілку master і вже веде свою роботу в ній. Перевірити можна **git branch**.

Для того, щоб перейти на іншу гілку, є команда

**git checkout <branch\_name>**

Окрім розділеної історії в Git (гілкування), ми також можемо об’єднати одночасно два потоки розробки. Це означає, що нашу роботу в новій гілці ми можемо злити в master. Такий процес злиття можна виконати за допомогою команди:

**git merge <branch\_name>**

**HTML** - це мова розмітки, який представляє прості правила оформлення і компактний набір структурних і семантичних елементів розмітки (тегів).

HTML-документ складається з тексту, який являє собою інформаційний вміст і спеціальних засобів мови HTML - тегів розмітки, які визначають структуру і зовнішній вигляд документа при його відображенні браузером. Структура HTML документа досить проста:

1. Опис документа починається з вказівки його типу (секція DOCTYPE).

2. Текст документа полягає в тег <html>. Текст документа складається з заголовка і тіла, які виділяються відповідно тегами <head> і <body>.

- У заголовку (<head>) вказують назву HTML-документа і інші параметри, які браузер буде використовувати при відображенні документа.

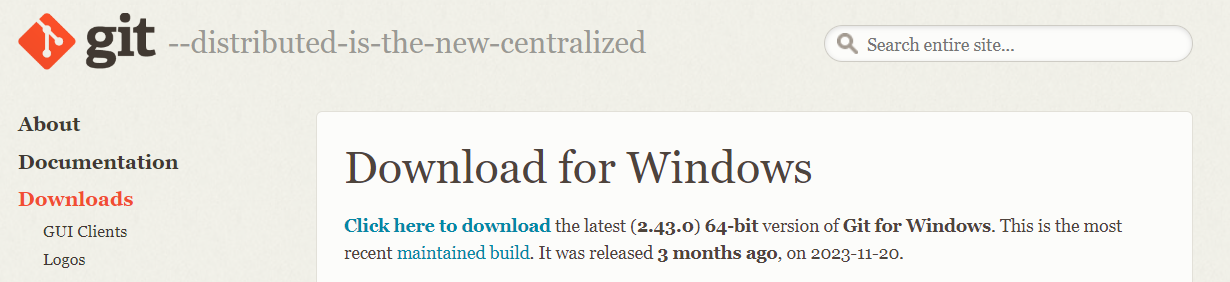
- Тіло документа (<body>) - це та частина, в яку поміщається власне вміст HTML-документа. Тіло включає призначений для відображення текст і керуючу розмітку документа (теги), які використовуються браузером.

Наявність секції DOCTYPE дозволяє вказати браузеру, який тип документа йому належить розбирати, тобто, які вимоги потрібно виконувати при обробці гіпертексту.

Створити html-документ можна в будь-якому текстовому редакторі, зберегти файл з розширенням .html.

**Виконання лабораторної роботи:**

1. Установка Git.



Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

2. Виведемо список усіх команд за допомогою команди git help.

Obraz zawierający tekst, zrzut ekranu, Czcionka, menu

Opis wygenerowany automatycznie

3. Переходимо в директорію проєкту і ініціалізуємо Git директорію за допомогою команди git init.

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Вміст підкаталогу .git:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

4. Створимо 4 HTML-сторінки за допомогою текстового редактора Notepad++.

Налаштування:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Opis wygenerowany automatycznie

Структура створеного HTML-документа:

Obraz zawierający tekst, oprogramowanie, Ikona komputerowa, Strona internetowa

Opis wygenerowany automatycznie

Збереження документа index.html:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

5. Аналогічно, створюємо ще 3 сторінки. У кожній присутній блок <head></head>, в якому розміщуються метатеги (включно з метаданними стосовно автора документа) і назву <title></title>.

Obraz zawierający tekst, oprogramowanie, zrzut ekranu, linia

Opis wygenerowany automatycznie

Obraz zawierający tekst, oprogramowanie, zrzut ekranu, Ikona komputerowa

Opis wygenerowany automatycznie

Obraz zawierający tekst, oprogramowanie, zrzut ekranu, Ikona komputerowa

Opis wygenerowany automatycznie

Obraz zawierający tekst, oprogramowanie, Czcionka, Ikona komputerowa

Opis wygenerowany automatycznie

6. Додамо файли до індексу, закомітимо внесені зміни, додамо коментар. Також переглянемо стан директорії та індексу.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

7. Створення репозиторію.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

8. Додаємо віддалений репозиторій у Git.



9. Відправимо результат роботи в репозиторій.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

10. Переглянемо зміни.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

11. Переглянемо інформацію коміту, на який у даний час вказує HEAD.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

12. Результат виконання всіх кроків.

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

**Висновки:**

У ході виконання лабораторної роботи №1 з теми «Системи контролю версій. Git. Проєкт. Структура проєкту» було обрано тематику «Фармацевтика» для цієї та подальших лабораторних робіт. Відповідно до завдань лабораторної роботи, було виконано наступні кроки:

1. Було встановлено Git та ініціалізовано директорію у проєкті за допомогою команди git init.
2. Було створено 4 HTML-сторінки за допомогою текстового редактора Notepad++.
3. Додано створені сторінки під контроль Git на комп’ютері за допомогою команд git add (додати зміни до індексу) та git commit (зберегти зміни до історії проєкту).
4. Після реєстрації і створення репозиторію, результати роботи було надіслано на віддалений сервер за допомогою команди git push.

Результати роботи відображено на скріншоті, що відповідає кроку №12 при виконанні лабораторної роботи.