

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Івано-Франківський національний технічний університет нафти і газу

Кафедра ІТТС

ЛАБОРАТОРНА РОБОТА №1.1

Керування версіями за допомогою GIT

Виконав:

студент групи ПЗС-24-1

Срібний Р. Ю.

Перевірила:

доцент Штаєр Л. О.

Івано-Франківськ

2025

Мета роботи: вивчити та закріпити на практиці можливості системи керування версіями, одержати практичний досвід у використанні Git та GitHub.

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 перешкод один одному, проводити роботу над груповими проєктами.

Системи керування версіями зазвичай використовуються при розробці програмного забезпечення для відстеження, документування та контролю над поступовими змінами в електронних документах: у коді додатків, 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.

Системи керування версіями існують двох основних типів: з централізованим сховищем та розподіленим.

Централізовані системи контролю версій. Централізована система контролю версій (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 містяться на одному сервері;
- можлива реалізація функції блокування файлів;
- можливість керування доступом до файлів.

Недоліки:

- необхідність мережевого з'єднання для оновлення робочої копії чи збереження змін.

До таких систем відносять Subversion, Team Foundation Server.

Розподілені системи контролю версій. Розподілена система контролю версій (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:

- кожен з розробників працює зі своїм власним репозиторієм;
- рішення щодо злиття віток приймається керівником проєкту;
- немає потреби в мережевому з'єднанні.

Недоліки:

- немає можливості контролю доступу до файлів;
- відсутня загальна нумерація версії файлу;
- значно більша кількість необхідного дискового простору;
- немає можливості блокування файлів.

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

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 проєкту 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 проєкту (final). Більшість

систем контролю версії дозволяють кожному з об'єктів присвоювати теги, за допомогою яких можна формувати нові гілки та репозиторії.

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

- можливість створення різних варіантів одного документу;
- документування всіх змін (коли і 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.

Словник основних термінів-сленгів:

- транк (trunk) – основна гілка коду;
- бранч (branch) – відгалуження;
- чекін (Check in (submit, commit)) – відправлення коду в репозиторій;
- чекаут (Check out) – одержання зміни з репозиторію;
- конфлікти – виникають, коли кілька людей правлять один і той же код, конфлікти можна вирішувати;
- патч – шматок з записаними змінами, які можна застосувати до сховища з кодом.

Керування версіями розробки програмного забезпечення продемонстровано на прикладі сервісу GitHub.

Для керування версіями розробки програмного забезпечення обрано сервіс GitHub.

Переваги:

- простий спосіб реєстрації;
- адаптовані під розповсюдженні версії операційних систем клієнти;
- простий спосіб синхронізації між сервісом та локальною версією;
- підтримка багатьох мов програмування;

– можливість відновлення репозиторію після видалення.

Недоліки:

– для Windows необхідний NET Framework 4.0;

– ручна синхронізація;

– для проєкту що складається з декількох директорій, потрібно створити відповідно декілька репозиторіїв у системі та відслідковувати кожен з них;

– невірне відображення кирилиці після відновлення з віддаленого репозиторію.

Таблиця 1.1 – Основні команди git

Команда	Призначення
1	2
git add	додає вміст робочої директорії в індекс (staging) area) для подальшого commit
git status	показує стану файлів в робочій директорії і індексі: які файли змінені, але не додані в індекс; які очікують комітів в індексі
git commit	бере всі дані, додані в індекс за допомогою git add, і зберігає їх зліпок у внутрішній базі даних, а потім переміщує вказівник поточної гілки на цей зліпок. Ключі: -a для додавання всіх змін в індекс без використання git add, -m для передачі повідомлення комітів без запуску редактора
git fetch	зв'язується з віддаленим репозиторієм і забирає з нього всі зміни, яких поки немає і зберігає їх локально
git pull	працює як комбінація команд git fetch і git merge, тобто Git спочатку забирає зміни із зазначеного віддаленого сховища, а потім намагається злити їх з поточної гілкою
git push	використовується для встановлення зв'язку з віддаленим репозиторієм, обчислення локальних змін відсутніх в ньому, і їх передачі в вищезгаданий репозиторій