**Лабораторна робота №1**

з курсу «**Проектування інформаційних систем**»

на тему: «**Системи контролю версій SVN,GIT**»

Виконав:

студент групи ДА-61

Попов Григорій

**Мета роботи:** за допомогою системи контролю версій завантажити коди програми у репозіторій. Відтворити типовий цикл розробки програмного забезпечення з використанням системи контролю версій.

**Завдання:**

1. Обрати безкоштовну систему репозиторія для системи контролю версіями, наприклад projectlocker, або інш.

2. Встановити клієнтське безкоштовне програмне забезпечення для роботи с системою контролю версій (GIT, SVN clients).

3. Протягом роботи над лабораторними роботами 2-6 використовувати систему контролю версіями.

4. Описати цикл розробки програмного забезпечення з використанням системи контрою версій.

**Короткі теоретичні відомості:**

**Система контролю версій** - це система, що полегшує роботу із мінливими даними шляхом зберігання різних версій документів, що відстежуються з можливістю швидко переходити від однієї версії до іншої. В більш глобальному розумінні СКВ відстежує певний набір файлів та версіонує їх в цілому, відстежуючи зміни стану директорії.

У далекому минулому усі СКВ відстежували зміни даних на локальному компьютері користувача безпосередньо. Вони створювали окрему копію файлу аби зберігти його версію. Однак, із розвитком інформаційних технологій в цілому та розподілених обчислень в особливості, з’явилась потреба в єдиному централізованому сховищі, до якого могли б мати доступ декілька людей одночасно. Таким чином з’явився новий тип СКВ, який для кожної нової гілки створював окремий зліпок файлової системи. Зрозуміло, що такий підхід витрачав чимало пам’яті, тому повний перелік версій зберігався лише в централізованому сховищі, а користувачі мали змогу зберігати на локальному сховищі лише певну одну версію, з якою працюють на даний момент. Представниками цього типу є такі СКВ як: Subversion, CVS та Perforce. В такому принципі роботи виявився суттєвий недолік: у випадку збоїв на централізованому сховищі виникали проблеми з синхронізацією локальних (користувачі не мали можливості переходити на іншу версію/гілку) та навіть дані могли бути втрачені (у випадку їхньої цілковитої втрати з серверу-ЦС).

Основною відмінністю від інших систем (таких як Subversion та подібних їй) є те, як Git сприймає дані. Концептуально, більшість СКВ зберігають кожний наступний стан файлової системи як різницю між поточним та попереднім станами. Ці інші системи (CVS, Subversion, Perforce, Bazaar тощо) розглядають інформацію як список файлів та змін кожного з них протягом деякого часу (це зазвичай називають **оснований на дельтах** контроль версій).

Git не оброблює та не зберігає свої дані таким чином. Замість цього, Git сприймає свої дані радше як низку знімків мініатюрної файлової системи. Щоразу, як ви створюєте коміт, тобто зберігаєте стан вашого проекту, Git запам’ятовує як виглядають всі ваші файли в той момент і зберігає посилання на цей знімок. Для ефективності, якщо файли не змінилися, Git не зберігає файли знову, просто робить посилання на попередній ідентичний файл, котрий вже зберігається. Git вважає свої дані більш як **потік знімків**. Однак, такий підхід *змусив* Git повністю оновити принцип своєї роботи, тоді як інші дельта-орієнтовані СКВ багато чого взяли від своїх попередників.

Більшість операцій у Git потребують лише локальних файлів та ресурсів для здійснення операцій — немає необхідності в інформації з інших комп’ютерів вашої мережі. Якщо ви звикли до ЦСКВ, де більшість операцій обтяжені такими мережевими запитами, то цей аспект може привести вас до думки, що боги швидкості наділили Git неземною силою. Через те, що повна історія проекту знаходиться на вашому локальному диску, більшість операцій здійснюються майже миттєво. Також, цим шляхом Git вирішує проблему втрати даних: якщо з якихось причин дані будуть втрачені з цетрального сховища, їх завжди можна буде відновити з будь-якого з локальних сховищ.

Підсумуємо відмінності Git та Subversion (svn):

Git: повністю розподілена СКВ, кожне локальне сховище є цілковитою копією центрального.

Subversion: відноситься до СКВ більш старого неповністю розподіленого типу, усі версії зберігаються лише на центральному сховищі.

Git: Кожний коміт – це «знімок» усіх даних, що може бути стиснутий. Сукупність комітів утворюють орієнтований граф.

Svn: Кожний коміт зберігає дані лише про відмінність його від попереднього, таким чином розмір коміту менший.

Git: Має вбудовану можливість оперувати бінарними даними (хоча робить це менш ефективно, ніж у випадку з текстовими даними).

Svn: Не призначений для обробки бінарних даних.

Git: Швидко працює для будь-якої кількості гілок.

Svn: Операції з гілками виконуються повільніше за аналогічні у Git (окрім, власне, створення гілки).

Підсумок: Git являє собою більш універсальну СКМ, його ефективніше використовувати у більшості випадків. Svn може бути кращим рішенням, якщо не планується велика кількість гілок/їхнє часте поєднання та версіонуванню підлягятимуть лише текстові дані.

**Стратегія розгалуження для даного проекту(Git flow):**

Оскільки в даному циклу лабораторних робіт процес розробки буде виконуватись однією людиною, стратегія розаглуження буде спрощенною у порівнянні з класичними Git flow/Github flow.

Головна гілка проекту: Popov-dev – слугуватиме як дещо модифікований аналог master у Git flow.

Гілка розробки: Popov-app-dev – аналог develop у моделі Git Flow. Звідси відгалужуватимуться гілки розробки особливостей (feature-branches). У випадку невеликих змін, наприклад, завантаження протоколу, гілка особливості може не створюватись.

Гілки розробки особливостей: відгалужуватимуться від гілки розробки та будуть зливатися з нею після завершення особливості (без видалення гілки). Шаблон назви: Popov-app-\*featurename\*-dev.

Кожна стабільна версія гілки розробки, що матиме риси завершеності (завершена лабораторна робота, написані тести, тощо) буде тагуватися як певна остаточна версія додатку (release). На цьому етапі також можливе злиття гілки розрообки зі спільною гілкою репозиторію – dev.

При наявності виправлень(hotfix), кожне виправлення буде виконано як коміт у гілці розробки, головна гілка матиме відповідну мітку (tag).

**Висновок:**

В даній лабораторній роботі було досліджено можливості системи контролю версій Git, порівняно її із СКВ Subversion та викладено стратегію розгалуження, яка буде застосована до даного проекту.