*МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ*

*НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ*

*„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. СІКОРСЬКОГО”*

*НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС*

*„ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ”*

# Лабораторна робота №1

## *з курсу «ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ*»

## *Тема:* «***Системи контролю версій SVN, GIT***»

*Варіант №15*

Виконав

студент 4 курсу

групи ДА-61

Линовський Андрій

Київ – 2019

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

Для виконання лабораторних робіт було вибрано систему контролю версій Github.

**GIT**

Git считает хранимые данные набором слепков небольшой файловой системы. Каждый раз, когда пользователь фиксируете текущую версию проекта, Git, по сути, сохраняет слепок того, как выглядят все файлы проекта на текущий момент. Ради эффективности, если файл не менялся, Git не сохраняет файл снова, а делает ссылку на ранее сохранённый файл. То, как Git подходит к хранению данных, похоже на рисунок ниже



Git больше похож на небольшую файловую систему с невероятно мощными инструментами, работающими поверх неё, чем на просто СКВ.

Для совершения большинства операций в Git'е необходимы только локальные файлы и ресурсы, т.е. обычно информация с других компьютеров в сети не нужна.

К примеру, чтобы показать историю проекта, Git'у не нужно скачивать её с сервера, он просто читает её прямо из локального репозитория пользователя. Поэтому историю пользователь увидт практически мгновенно. Если нужно просмотреть изменения между текущей версией файла и версией, сделанной месяц назад, Git может взять файл месячной давности и вычислить разницу на месте, вместо того чтобы запрашивать разницу у СКВ-сервера или качать с него старую версию файла и делать локальное сравнение.

**Git следит за целостностью данных**

Перед сохранением любого файла Git вычисляет контрольную сумму, и она становится индексом этого файла. Поэтому невозможно изменить содержимое файла или каталога так, чтобы Git не узнал об этом. Эта функциональность встроена в сам фундамент Git'а и является важной составляющей его философии. Если информация потеряется при передаче или повредится на диске, Git всегда это выявит.

**Три состояния**

В Git'е файлы могут находиться в одном из трёх состояний: зафиксированном, изменённом и подготовленном. "Зафиксированный" значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

Таким образом, в проектах, использующих Git, есть три части: каталог Git'а (Git directory), рабочий каталог (working directory) и область подготовленных файлов (staging area).



Каталог Git'а — это место, где Git хранит метаданные и базу данных объектов вашего проекта. Это наиболее важная часть Git'а, и именно она копируется, когда вы клонируете репозиторий с другого компьютера.

Рабочий каталог — это извлечённая из базы копия определённой версии проекта. Эти файлы достаются из сжатой базы данных в каталоге Git'а и помещаются на диск для того, чтобы вы их просматривали и редактировали.

Область подготовленных файлов — это обычный файл, обычно хранящийся в каталоге Git'а, который содержит информацию о том, что должно войти в следующий коммит. Иногда его называют индексом (index), но в последнее время становится стандартом называть его областью подготовленных файлов (staging area).

Стандартный рабочий процесс с использованием Git'а выглядит примерно так:

1. Вы вносите изменения в файлы в своём рабочем каталоге.
2. Подготавливаете файлы, добавляя их слепки в область подготовленных файлов.
3. Делаете коммит, который берёт подготовленные файлы из индекса и помещает их в каталог Git'а на постоянное хранение.

Если рабочая версия файла совпадает с версией в каталоге Git'а, файл считается зафиксированным. Если файл изменён, но добавлен в область подготовленных данных, он подготовлен. Если же файл изменился после выгрузки из БД, но не был подготовлен, то он считается изменённым.

**Использовал следующие команды:**

**git clone** — клонирует Git-репозиторий из существующего репозитория.

**git checkout -b Andrey –** создает ветку и сразу же переходит в неё

**git commit** — записывает изменения в репозиторий

**git merge** — делает слияние для одной или нескольких веток в текущую

**git status** — показывает состояние рабочего дерева

**Висновок:** навчився, за допомогою системи контролю версій, завантажувати данні у репозиторій. Відтворювати типовий цикл розробки програмного забезпечення з використанням системи контролю версій.