

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**  
**„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”**  
**НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС**  
**„ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ”**

Лабораторна робота №1

з курсу «Проектування інформаційних систем» на тему:

“Системи контролю версій SVN, GIT.”

Виконав:

студент 4 курсу

групи ДА-61

Гірянський Б.П.

Київ 2019

## Зміст

1. Мета роботи.....	3
2. Завдання роботи. ....	3
3. Результати роботи.....	3
4. Опис команд, які використовувалися протягом виконання роботи з системою контролю версіями: .....	7
5. Висновки. ....	7

## 1. Мета роботи.

за допомогою системи контролю версій завантажити коди програми у репозиторій. Відтворити типовий цикл розробки програмного забезпечення з використанням системи контролю версій.

## 2. Завдання роботи.

2.1. Обрати безкоштовну систему репозиторія для системи контролю версіями,

наприклад projectlocker, або інш.

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

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

2.4. Описати цикл розробки програмного забезпечення з використанням системи контролю версій.

## 3. Результати роботи.

### 2.0. Порівняння SVN і Git.

GIT      VERSUS      SUBVERSION	
Git	Subversion
Git is a distributed version control system used for source code management.	Subversion (or SVN) is a centralized versioning and revision control system.
It creates a local repository to store everything locally instead of using a centralized server.	It uses a centralized server to store changes in source code.
Network access is not mandatory for Git operations.	Network is required for almost all of SVN operations.

A subproject is called a Git "submodule".	A subproject is called an "SVN External".
Git does not have a global revision number.	SVN does have a global revision number.
Git contents are cryptographically check-summed using the SHA-1 hash algorithm.	SVN does not have hashed contents.

2.1. - Системою контролю версій було обрано – github.

2.2. - Було встановлено програмне забезпечення git для роботи із системою контролю версій github. Посилання на дане програмне забезпечення - <https://git-scm.com/downloads>.

2.3. - Система контролю роботи в даній роботі використовується тільки для збереження протоколу лабораторної роботи №1.

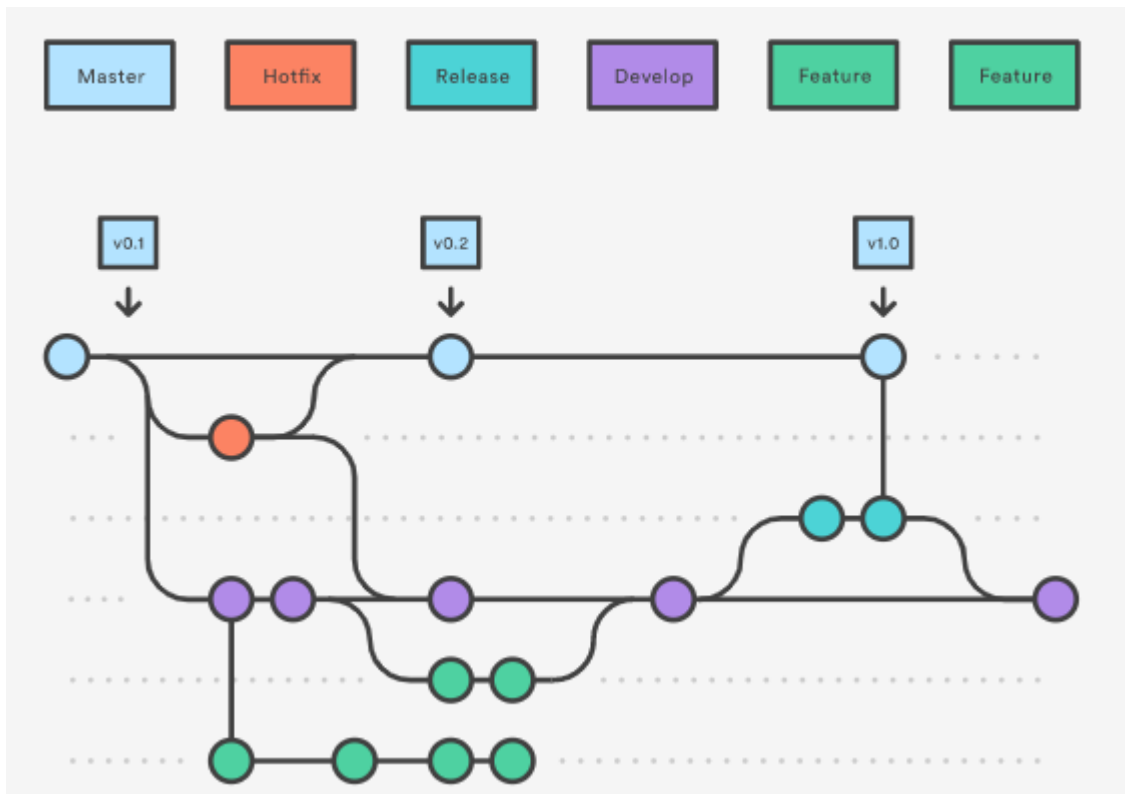
2.4. Gitflow Workflow - це дизайн робочого процесу Git, який вперше був опублікований та популярний Vincent Driessen у nvie . Gitflow Workflow визначає сувору модель розгалуження, розроблену навколо випуску проекту. Це забезпечує надійну основу для управління більшими проектами.

Gitflow ідеально підходить для проектів, які мають запланований цикл випуску. Цей робочий процес не додає нових понять або команд, що перевищує необхідні для робочого процесу Feature Branch . Натомість він присвоює дуже специфічні ролі різним галузям і визначає, як і коли вони мають взаємодіяти. Окрім featureфілій, він використовує окремі гілки для підготовки, ведення та запису випусків. Звичайно, ви також можете скористатися всіма

перевагами робочого процесу Feature Branch: витягніть запити, поодинокі експерименти та ефективніше співробітництво.

Gitflow насправді просто абстрактна ідея робочого процесу Git. Це означає, що він диктує, які гілки створити та як їх об'єднати разом. Ми торкнемось цілей галузей нижче. Набір інструментів git-flow - це фактичний інструмент командного рядка, який має процес встановлення. Процес установки git-flow є простим. Пакети для git-flow доступні у багатьох операційних системах. У системах OSX можна виконати `brew install git-flow`. У Windows вам потрібно буде завантажити та встановити git-flow . Після встановлення git-flow ви можете використовувати його у своєму проєкті, виконавши `git flow init`. Git-flow - це обгортка навколо Git. `git flow init` Команда є розширенням за замовчуванням `git init` команди і нічого в вашому сховищі, крім створення філій для вас не зміниться.

Вигляд структури гілок при роботі над проєктом при використанні системи контролю версій.



Дана структура не є обов'язковою, але приймається багатьма командами, бо добра підходить для більшості проєктів.

Master – вітка для збереження версій продукту.

Hotfix – вітка для проведення швидких вирішень помилок у версії програмного продукту після збереження в Master.

Release – вітка в якій зберігається продукт у передостанньому етапі виходу нової версії. Проводиться тестування і якщо воно є успішним, то між вітками проводять merge commit.

Develop – вітка у якій проводять розробку програмного продукту, безпосереднє написання функціоналу між версіями продукту.

Feature – вітка для розробки особливого функціоналу продукту, яким від буде особливим по відношенню із іншими продуктами.

## 2.5. Структура репозиторію

This PC > New Volume (D:) > repos > PAIS > peaceLab					Search peaceLab
Name	Date modified	Type	Size		
lab1	9/17/2019 8:45 PM	File folder			
lab2	9/17/2019 7:38 PM	File folder			
lab3	9/17/2019 7:38 PM	File folder			
lab4	9/17/2019 7:38 PM	File folder			
lab5	9/17/2019 7:38 PM	File folder			
lab6	9/17/2019 7:38 PM	File folder			
Practice	9/17/2019 7:38 PM	File folder			
projects	9/17/2019 7:38 PM	File folder			
README.md	9/17/2019 7:38 PM	MD File	1 KB		

This PC > New Volume (D:) > repos > PAIS > peaceLab > lab1					Search lab1
Name	Date modified	Type	Size		
KhaletaMariia1Lab	9/17/2019 7:38 PM	Документ Micros...	6,030 KB		
empty	9/17/2019 7:38 PM	File	0 KB		
LW1Boiko	9/17/2019 7:38 PM	Документ Micros...	147 KB		
PAIS_LAB_1_Giriansk_B_P	9/17/2019 8:40 PM	Документ Micros...	50 KB		

4. *Опис команд, які використовувалися протягом виконання роботи з системою контролю версіями:*

1. Внесення коміту – git commit
2. Завантаження на сервер – git push
3. Клонування репозиторія – git clone
4. Створення нової гілки – git branch
5. Перехід між гілками – git checkout
6. З'єднання гілок – git merge
7. Збереження змін - git add

5. *Висновки.*

Системи контролю версій є незамінним програмним забезпеченням при роботі на великому проекті із великою кількістю розробників, або команд

які відповідають за різні етапи розробки: розробка, тестування, вихід кінцевої версії .