Изображение выглядит как панорама, черно-белый

Автоматически созданное описание

Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем і технологій

Лабораторна робота №1  
**Технології розроблення програмного забезпечення**

«Системи контролю версій. Git.»

Виконав:

Перевірив:

Мягкий М. Ю.

Студент групи ІА-23

Ширяєв Д. Ю.

Київ 2024

**Зміст**

[**Теоретичні відомості** 3](#_Toc177734450)

[**Хід роботи** 4](#_Toc177734451)

[**Висновок** 6](#_Toc177734452)

# **Теоретичні відомості**

git init – ініціалізує локальний гіт-репозиторій у поточній директорії.

git branch <branch\_name> - створює нову гілку, залишаючись у поточній. Якщо не писати назву гілки, виведе у консолі список існуючих гілок з вказанням поточної.

git checkout <branch\_name> - переключає на гілку чи комміт, при використанні «-b» створюється нова гілка с вказаною назвою та одразу робить її поточною.

git switch <branch\_name> - схожий функціонал з checkout, для створення і переключення на нову гілку використовується «-c»

git add <file\_name> - додає вказаний файл до стейджу, для подальшого комміту змін. Якщо замість назви файлу написати «.», будуть застосовані зміни в усіх файлах.

git commit -m ‘<message>’ – додає до історії змін новий комміт з вказаним повідомленням.

git merge <branch\_name> - об'єднує зміни з однієї гілки в іншу, створюючи новий комміт злиття, який містить всі зміни з обох гілок.

git rebase <branch\_name> - переносить комміти з однієї гілки на вершину іншої, переписуючи історію комітів.

git cherry-pick <commit\_code> - переносить конкретні комміти з однієї гілки на іншу.

git status – виводить перелік загальної інформації про статус репозиторію, вказуючи поточну гілку, незакоммічені зміни та файли, в яких виник конфлікт злиття.

git log – виводить перелік коммітів поточної гілки. Якщо дописати «--all», виведе повний перелік з усіх гілок.

# **Хід роботи**

git init #ініціалізація гіту

git commit --allow-empty -m 'init'

git branch branch1 #створення 3 гілок різними способами

git checkout -b branch2

git checkout master

git switch -c branch3

git branch

echo 1 > 1.txt #заповнення гілок коммітами

git add 1.txt

git commit -m"create 1.txt"

git echo 2 > 2.txt

echo 2 > 2.txt

git add 2.txt

git commit -m"create 2.txt"

git log

git switch cf588f

git switch cf588f --detach

git switch branch2

echo 1 > 2.txt

git add 2.txt

git commit -m"add 2.txt"

git merge branch3 #демонстрація злиття з конфліктами

git status

git add 2.txt

git status

git merge --continue

git switch branch3

git switch branch1

echo 3 > 1.txt

git add 1.txt

git commit -m"add 1.txt"

git rebase branch3

git status

git rebase --continue

git add 1.txt

git status

git rebase --continue

git log

git log --all

git switch branch1

git cherry-pick 1f5c4

git status

echo 1 > 2.txt

git commit -am"edit 2.txt"

git switch branch2

git log --all

git cherry-pick f53b500

git status

git add 2.txt

git cherry-pick --continue

# **Висновок**

В ході виконання даної лабораторної роботи я поглибив свої знання та навички у роботі с Git: робота з гілками, створення коммітів, злиття змін з різних гілок та вирішення виявлених при цьому конфліктів.