Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №1

Технології розроблення програмного забезпечення

«Системи контроля версій. Розподілена система контролю версій «Git».»

Виконала:
студентка групи ІА-32
Красоха В.О.

Перевірив:
Мягкий М.Ю.

Київ 2025

Тема: Системи контроля версій. Розподілена система контролю версій «Git».

Мета: Навчитися виконувати основні операції в роботі з децентралізованими системами контролю версій на прикладі роботи з сучасною системою Git.

Теоретичні відомості:

Система управління версіями (від англ. Version Control System або Source Control System) – програмне забезпечення яке призначено допомогти команді розробників керувати змінами в вихідному коді під час роботи. Система керування версіями дозволяє додавати зміни в файлах в репозиторій і таким чином після кожної фіксації змін мани нову ревізію файлів. Це дозволяє повертатися до попередніх версій коду для аналізу внесених змін або пошуку, які зміни привели до появи помилки. Таким чином можна знайти хто, коли і які зміни зробив в коді, а також чому ці зміни були зроблені.

Такі системи найбільш широко використовуються при розробці програмного забезпечення для зберігання вихідних кодів програми, що розробляється. Однак вони можуть з успіхом застосовуватися і в інших областях, в яких ведеться робота з великою кількістю електронних документів, що безперервно змінюються. Зокрема, системи керування версіями застосовуються у САПР, зазвичай у складі систем керування даними про виріб (PDM). Керування версіями використовується у інструментах конфігураційного керування (Software Configuration Management Tools).

Робота з Git може виконуватися з командного рядка, а також за допомогою візуальних оболонок. Командний рядок використовується програмістами, як можливість виконання всіх доступних команд, а також можливості складання складних макросів. Візуальні оболонки як правило дають більш наглядне представлення репозиторію у вигляді дерева, та більш зручний спосіб роботи з репозиторієм, але, дуже часто доступний не весь набір команд Git, а лише саме ті, що найчастіше використовуються.

Прикладами візуальних оболонок для роботи з Git є Git Extension, SourceTree, GitKraken, GitHub Desktop та інші.

Основна ідея Git, як і будь-якої іншої розподіленої системи контроля версій – кожен розробник має власний репозиторій, куди складаються зміни (версії) файлів, та синхронізація між розробниками виконується за допомогою синхронізації репозиторіїв.

Відповідно, є ряд основних команд для роботи:

1. Клонувати репозиторій (git clone) – отримати копію репозиторію на локальну машину для подальшої роботи з ним;
2. Синхронізація репозиторіїв (git fetch або git pull) – отримання змін із віддаленого (вихідного, центрального, або будь-якого іншого такого ж) репозиторію;

3. Фіксація змін в репозиторій (git commit) – фіксація виконаних змін в програмному коді в локальний репозиторій розробника;

4. Синхронізація репозиторіїв (git push) – переслати зміни – push – передача власних змін до віддаленого репозиторію – Записати зміни – commit – створення нової версії;

5. Оновитись до версії – update – оновитись до певної версії, що є у репозиторії.

6. Об’єднання гілок (git merge) – об’єднання вказаною гілки в поточну (часто ще називається «злиттям»).

Таким чином, якщо розглядати основний робочий процес програміста в команді, то він виглядає наступним чином: На початку роботи з проєктом виконується клонування, після цього, в рамках виконання поставленої задачі, створюється бранч і всі зміни в коді, зроблені в рамках цієї задачі фіксуються в репозиторії (періодично виконується синхронізація з основним репозиторіем). Далі, коли задача виконана, то виконується об’єднання гілки з основною гілкою і фінальна синхронізація з центральним репозиторієм.

Хід роботи

1. Створити репозиторій

```
C:\Users\Valeria>git init tl1
Initialized empty Git repository in C:/Users/Valeria/tl1/.git/

C:\Users\Valeria>cd tl1
```

2. Створити дві додаткові гілки.

```
C:\Users\Valeria\tl1>git branch dev
```

```
C:\Users\Valeria\tl1>git switch -c nbn
Switched to a new branch 'nbn'
```

3. Створити конфлікт.

```
C:\Users\Valeria\tl1>git switch dev
Switched to branch 'dev'

C:\Users\Valeria\tl1>echo "зміна з dev">file.txt

C:\Users\Valeria\tl1>git add file.txt

C:\Users\Valeria\tl1>get commit -m "зміна з dev"
'get' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Valeria\tl1>git commit -m "зміна в dev"
[dev 13c6a9d] зміна в dev
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
C:\Users\Valeria\tl1>git switch nbn
Switched to branch 'nbn'

C:\Users\Valeria\tl1>echo "зміна з nbn">file.txt

C:\Users\Valeria\tl1>git add file.txt

C:\Users\Valeria\tl1>git commit -m "зміна в nbn"
[nbn e9581dd] зміна в nbn
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\Valeria\tl1>git merge dev
Auto-merging file.txt
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.
```

4. Вирішити конфлікт.

```
C:\Users\Valeria\tl1>git branch
  dev
  lkl
  master
* nbn

C:\Users\Valeria\tl1>git status
On branch nbn
Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
        both modified:   file.txt

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\Valeria\tl1>type file.txt
<<<<<<< HEAD
"зм?на з nbn"
=======
"зм?на з dev"
>>>>>>> dev

C:\Users\Valeria\tl1>notepad file.txt
```

```
"§¬?  § dev"
"§¬?  § dev"
```

```
C:\Users\Valeria\tl1>git add file.txt

C:\Users\Valeria\tl1>git commit -m "вирішено конфлікт"
[nbn 9b36760] вирішено конфлікт
 1 file changed, 2 insertions(+)

C:\Users\Valeria\tl1>git status
On branch nbn
nothing to commit, working tree clean
```

Висновок: Під час виконання лабораторної робити я навчилася виконувати основні операції в роботі з децентралізованими системами контролю версій на прикладі роботи з сучасною системою Git. А саме створення репозиторію, гілок, створення конфлікту та його вирішення.