

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО РАБОТЕ №1
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Хачатрян Владимир Владимирович
1 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: исследование основных возможностей Git и GitHub.

Цель работы: исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

Порядок выполнения работы:

Задание 1. Создал репозиторий в GitHub.

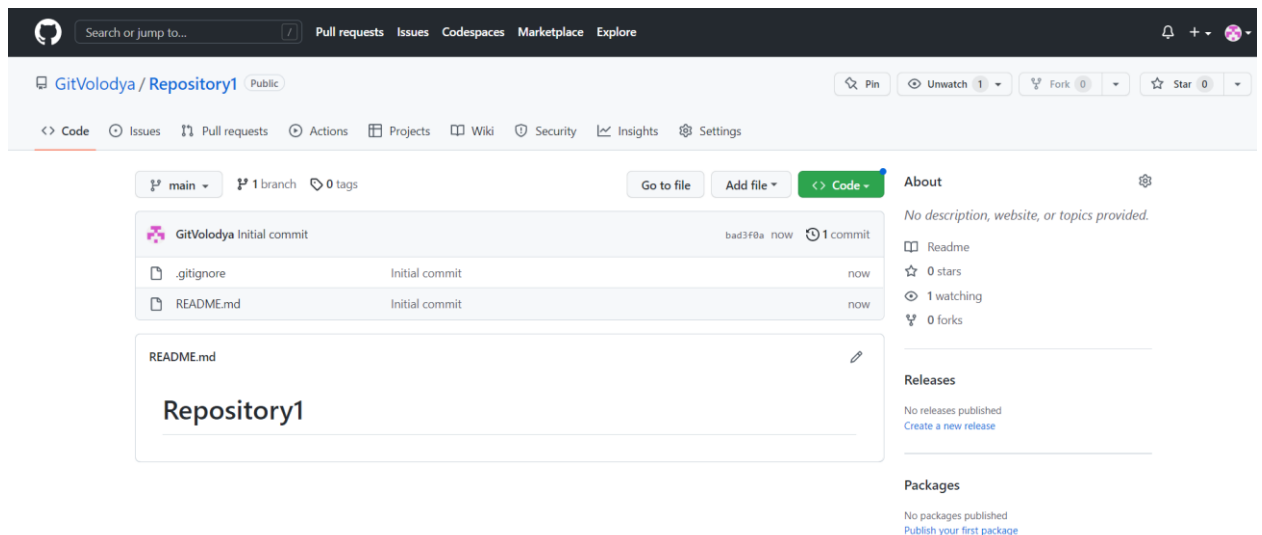


Рисунок 1. Новый репозиторий.

Задание 2. Ввел в командную строку *git version*, таким образом проверил правильность работы GIT.

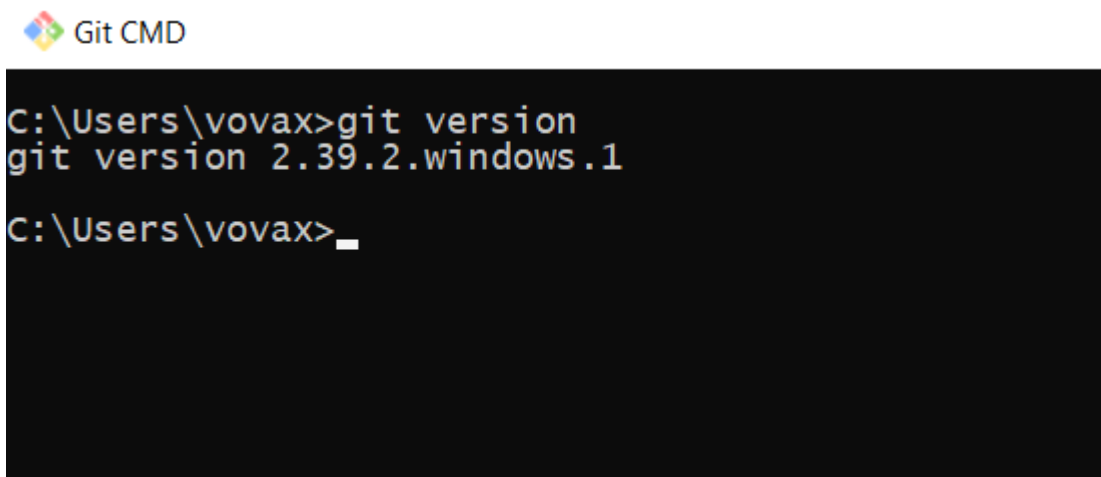
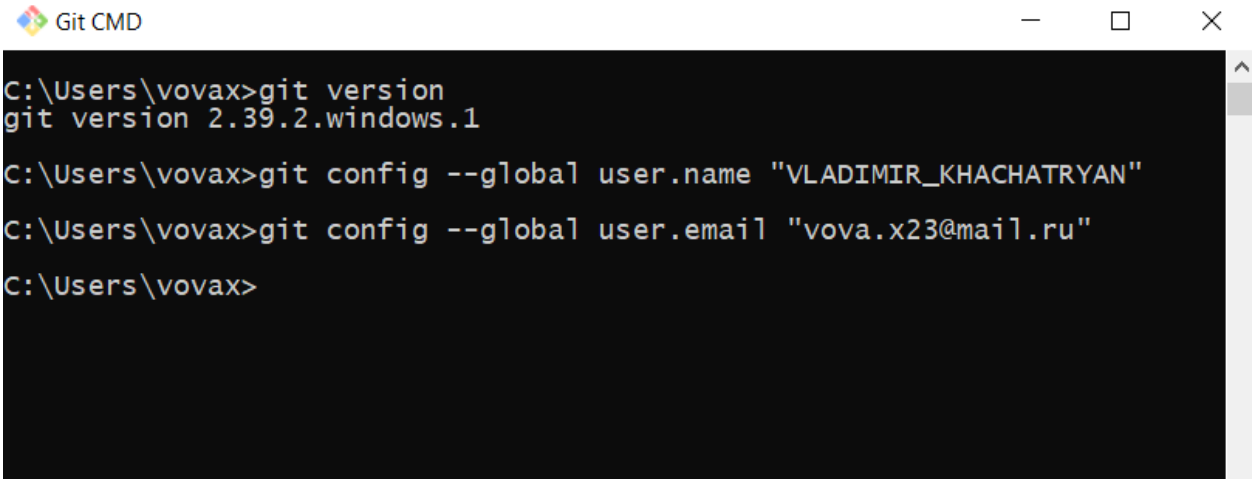


Рисунок 2. git version.

Задание 3. Ввел своё имя и свой email в командную строку.

A screenshot of a Windows command prompt window titled "Git CMD". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The terminal shows the following commands and output:

```
C:\Users\vovax>git version
git version 2.39.2.windows.1

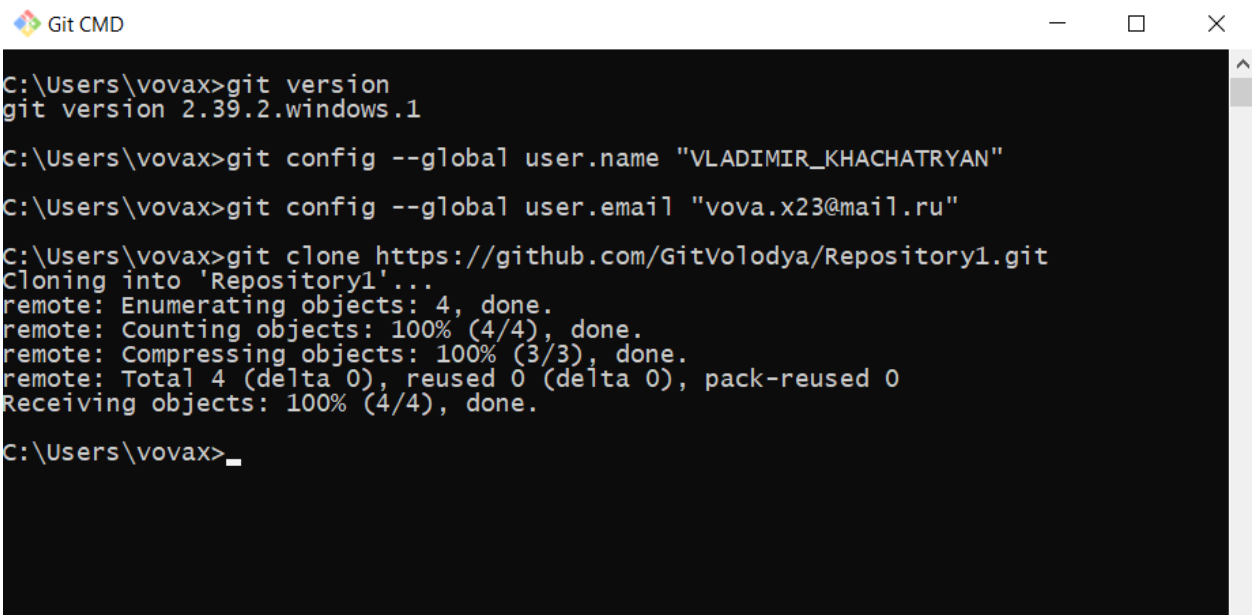
C:\Users\vovax>git config --global user.name "VLADIMIR_KHACHATRYAN"

C:\Users\vovax>git config --global user.email "vova.x23@mail.ru"

C:\Users\vovax>
```

Рисунок 3. Имя и почта.

Задание 4. Клонировал репозиторий на свой компьютер.

A screenshot of a Windows command prompt window titled "Git CMD". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The terminal shows the following commands and output:

```
C:\Users\vovax>git version
git version 2.39.2.windows.1

C:\Users\vovax>git config --global user.name "VLADIMIR_KHACHATRYAN"

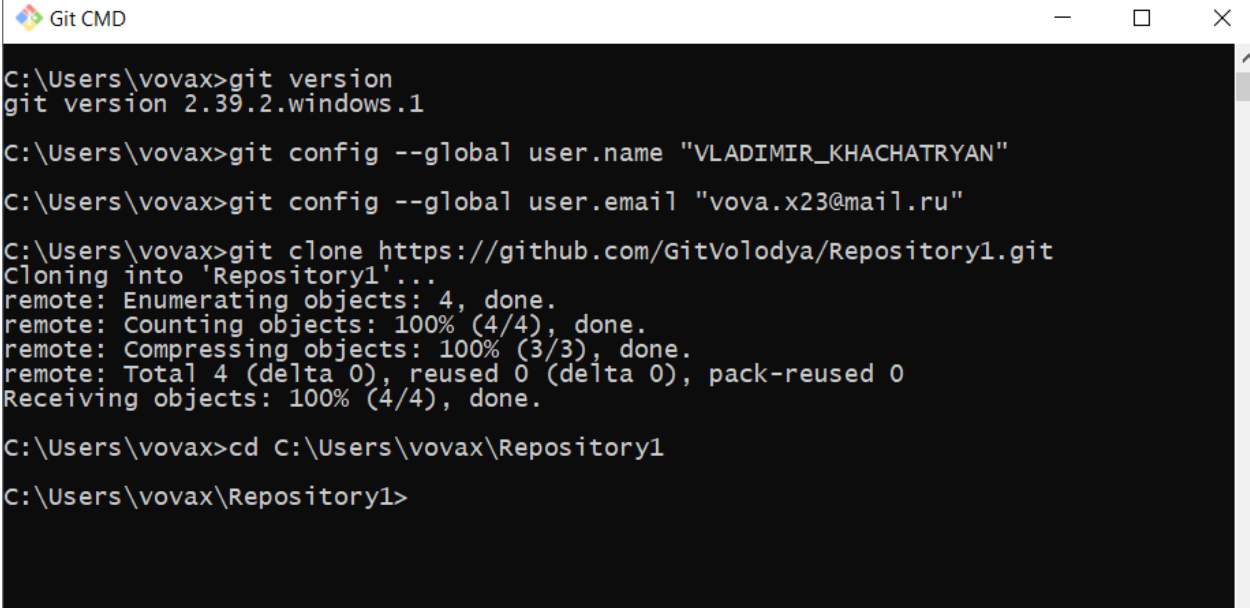
C:\Users\vovax>git config --global user.email "vova.x23@mail.ru"

C:\Users\vovax>git clone https://github.com/GitVolodya/Repository1.git
Cloning into 'Repository1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

C:\Users\vovax>_
```

Рисунок 4. Клонирование репозитория.

Задание 5. Перешел в консоли в папку, где находится сам файл README.



```
Git CMD
C:\Users\vovax>git version
git version 2.39.2.windows.1

C:\Users\vovax>git config --global user.name "VLADIMIR_KHACHATRYAN"

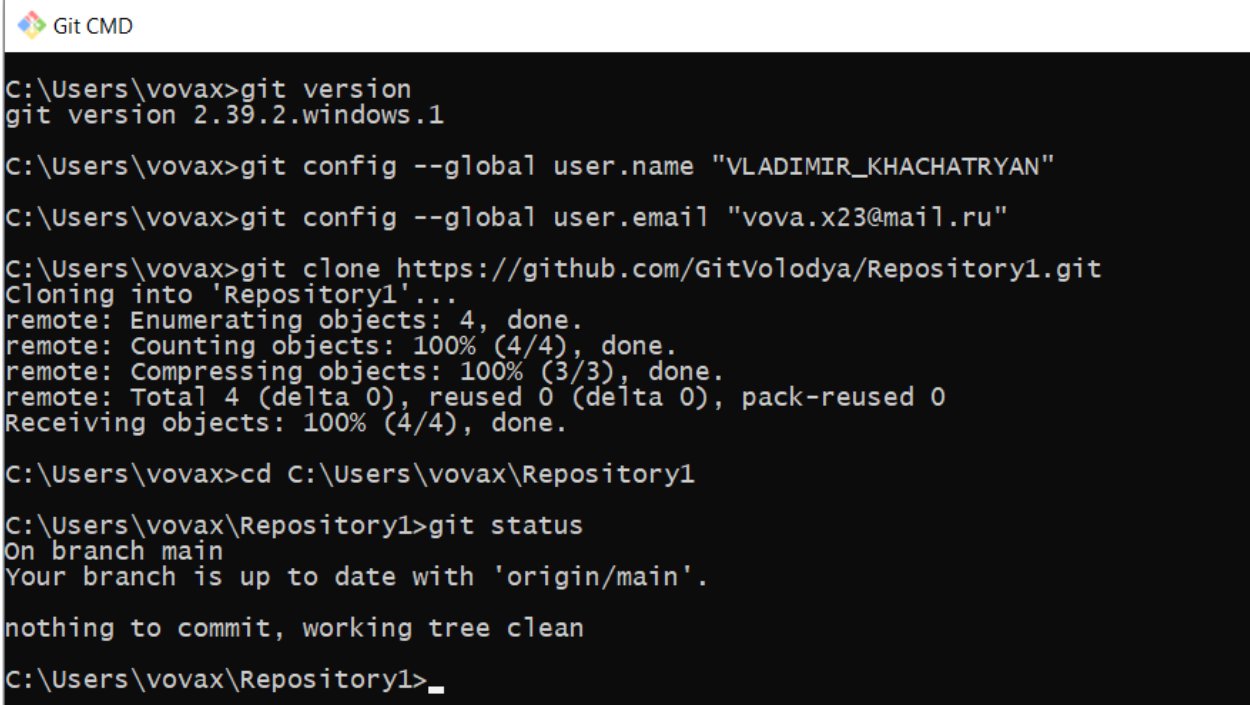
C:\Users\vovax>git config --global user.email "vova.x23@mail.ru"

C:\Users\vovax>git clone https://github.com/GitVolodya/Repository1.git
Cloning into 'Repository1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

C:\Users\vovax>cd C:\Users\vovax\Repository1
C:\Users\vovax\Repository1>
```

Рисунок 5. Переход на папку выше.

Задание 6. Проверил состояние репозитория с помощью команды *git status*.



```
Git CMD
C:\Users\vovax>git version
git version 2.39.2.windows.1

C:\Users\vovax>git config --global user.name "VLADIMIR_KHACHATRYAN"

C:\Users\vovax>git config --global user.email "vova.x23@mail.ru"

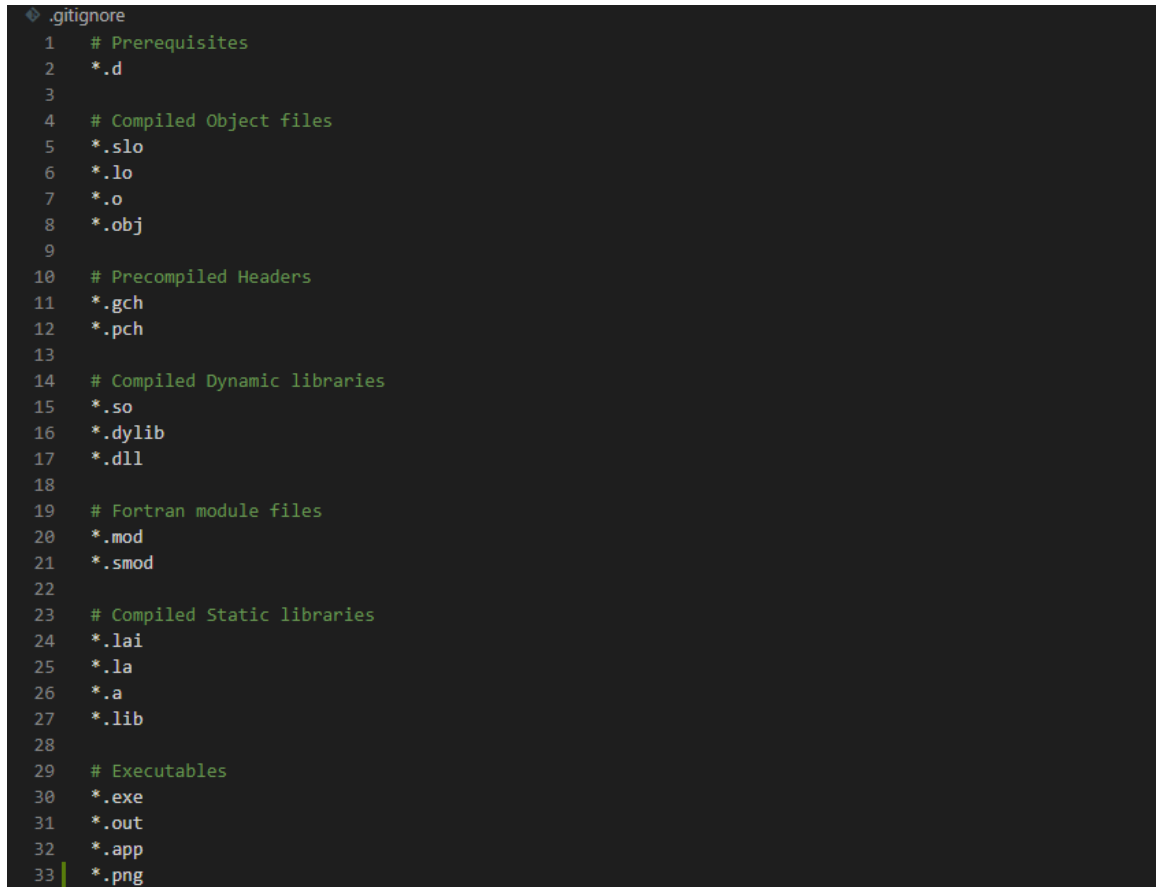
C:\Users\vovax>git clone https://github.com/GitVolodya/Repository1.git
Cloning into 'Repository1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

C:\Users\vovax>cd C:\Users\vovax\Repository1
C:\Users\vovax\Repository1>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
C:\Users\vovax\Repository1>
```

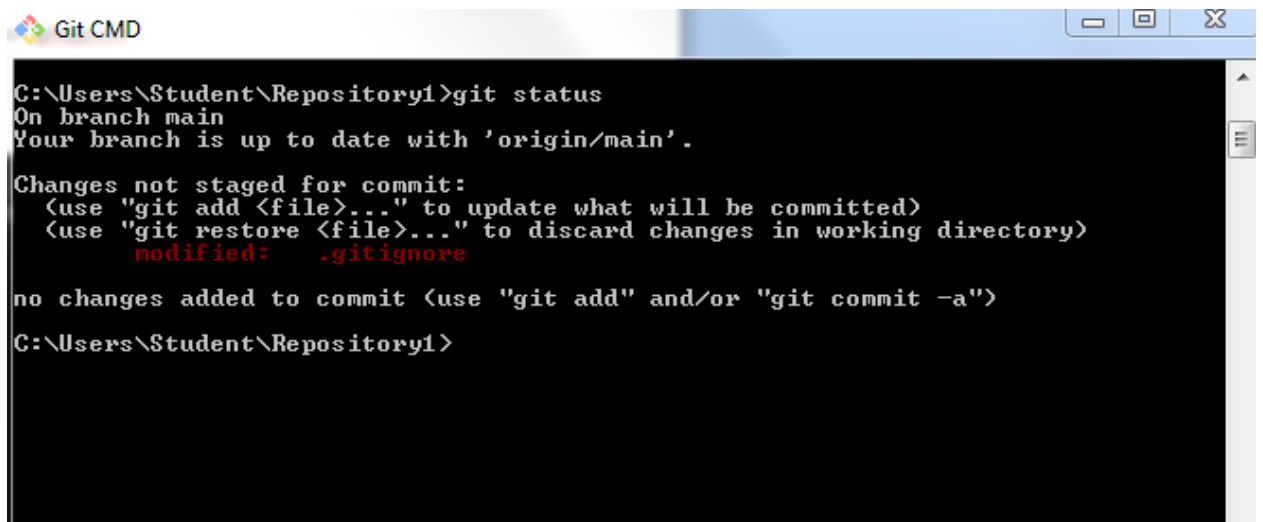
Рисунок 6. Состояние репозитория.

Задание 7. Дополнил файл .gitignore необходимым правилом игнорировать файлы .png



```
.gitignore
1  # Prerequisites
2  *.d
3
4  # Compiled Object files
5  *.slo
6  *.lo
7  *.o
8  *.obj
9
10 # Precompiled Headers
11 *.gch
12 *.pch
13
14 # Compiled Dynamic libraries
15 *.so
16 *.dylib
17 *.dll
18
19 # Fortran module files
20 *.mod
21 *.smod
22
23 # Compiled Static libraries
24 *.lai
25 *.la
26 *.a
27 *.lib
28
29 # Executables
30 *.exe
31 *.out
32 *.app
33 *.png
```

Рисунок 7. Дополнение файла gitignore.



```
Git CMD
C:\Users\Student\Repository1>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore

no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\Student\Repository1>
```

Рисунок 8. Изменение файла gitignore.

Задание 8. Внёс изменения в файл README.

```
# Repository1
ITS-b-o-22-1, 11.03.02, Khachatryan Vladimir Vladimirovich|
```

Рисунок 9. Изменение файла README.

```
C:\Users\Student\Repository1>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore
        modified:   README.md

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

C:\Users\Student\Repository1>git add .

C:\Users\Student\Repository1>
```

Рисунок 10. Внёс изменения.

Задание 9. Написал программу на языке C++, фиксировал изменения при написании в локальном репозитории, сделал 7 коммитов.

```
: \Users\Student\Repository>git add .  
:  
:\Users\Student\Repository>git commit -m "FirstPart"  
main adc749e1 FirstPart  
2 files changed, 9 insertions(+), 1 deletion(-)  
  
:\Users\Student\Repository>
```

Рисунок 11. Сделал 1 коммит.

```
C:\Users\Student\Repository1>git add .  
C:\Users\Student\Repository1>git commit -m "Part2"  
[main bea01aa] Part2  
1 file changed, 6 insertions(+)  
C:\Users\Student\Repository1>
```

Рисунок 12. Сделал 2 коммит.

```
C:\Users\Student\Repository1>git add .  
C:\Users\Student\Repository1>git commit -m "Part3"  
[main b7dc088] Part3  
1 file changed, 10 insertions(+)  
C:\Users\Student\Repository1>
```

Рисунок 13. Сделал 3 коммит.

```
C:\Users\Student\Repository1>git add .  
C:\Users\Student\Repository1>git commit -m "Part4"  
[main 6e009c7] Part4  
1 file changed, 6 insertions(+)  
C:\Users\Student\Repository1>
```

Рисунок 14. Сделал 4 коммит.

```
C:\Users\Student\Repository1>git add .  
C:\Users\Student\Repository1>git commit -m "Part5"  
[main 8e0112a] Part5  
1 file changed, 7 insertions(+)  
C:\Users\Student\Repository1>
```

Рисунок 15. Сделал 5 коммит.

```
C:\Users\Student\Repository1>git add .  
C:\Users\Student\Repository1>git commit -m "Part6"  
[main c4a826d] Part6  
1 file changed, 6 insertions(+)  
C:\Users\Student\Repository1>
```

Рисунок 16. Сделал 6 коммит.

```
C:\Users\Student\Repository1>git add .  
C:\Users\Student\Repository1>git commit -m "Part7"  
[main 0abc712] Part7  
1 file changed, 14 insertions(+)  
C:\Users\Student\Repository1>
```

Рисунок 17. Сделал 7 коммит.

Задание 10. Отправил в репозиторий GitHub.

Part7

main

GitVolodya committed 33 minutes ago

Showing 1 changed file with 14 additions and 0 deletions.

14 README.md

```
...  ...  @@ -1,43 +1,57 @@
1 1  # Repository1
2 2  ITS-b-o-22-1, 11.03.02, Khachatryan Vladimir Vladimirovich
3 3
4 4  #include <conio.h>
5 5  #include <stdio.h>
6 6  #include <stdlib.h>
7 7  #include <locale.h>
8 8  int** input(int n, int m)
9 9  {
10 10      setlocale(LC_ALL, "Rus");
11 11      int i, j;
12 12      int** a;
13 13      a = (int**)malloc(m * sizeof(int*));
14 14      for (i = 0; i < m; i++)
15 15          {
16 16              a[i] = (int*)malloc(n * sizeof(int*));
17 17              for (j = 0; j < n; j++)
18 18                  {
19 19                      a[i][j] = 0;
20 20                  }
21 21          }
22 22
23 23
24 24      for (i = 0; i < m; i++)
25 25          for (j = 0; j < n; j++)
26 26              {
27 27                  printf("\n ÃÄÅæÈëÄ åäåîäîð ìàððèòÛ A(%d,%d)", i + 1, j + 1);
28 28                  scanf_s("%d", &a[i][j]);
29 29              }
30 30      return a;
31 31  }
32 32  void output(int** z, int m, int n)
33 33  {
34 34      int i, j;
```

Рисунок 19. Проверка изменений в GitHub.

Ссылка на репозиторий: <https://github.com/GitVolodya/Repository1.git>

Ответы на контрольные вопросы:

1) Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

2) В чем недостатки локальных и централизованных СКВ?

Основной недостаток локальных СКВ – можно легко забыть, в какой директории мы находимся, и случайно изменить не тот файл или скопировать не те файлы, которые мы хотели.

Основной недостаток централизованных СКВ заключается в том, что это единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками.

3) К какой СКВ относится Git?

Git относится к распределенным СКВ (РСКВ)

4) В чем концептуальное отличие Git от других СКВ?

Основное отличие Git от любой другой СКВ (включая Subversion и её собратьев) — это подход к работе со своими данными. Концептуально, большинство других систем хранят информацию в виде списка изменений в файлах.

Git не хранит и не обрабатывает данные таким способом. Вместо этого, подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы.

5) Как обеспечивается целостность хранимых данных в Git?

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом.

6) В каких состояниях могут находиться файлы в Git? Как связаны эти состояния?

У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged).

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.

7) *Что такое профиль пользователя в GitHub?*

Профиль - это наша публичная страница на GitHub, как и в социальных сетях.

8) *Какие бывают репозитории в GitHub?*

Репозиторий бывает трех видов: локальных, централизованный, распределенный.

9) *Укажите основные этапы модели работы с GitHub.*

GitHub содержит в себе два хранилища:

А) *upstream* - это оригинальный репозиторий проекта, который мы скопировали.

Б) *origin* - ваш fork (копия) на GitHub, к которому у вас есть полный доступ.

Чтобы перенести изменения с вашей копии в исходному репозиторий проекта, нам нужно сделать запрос на извлечение.

10) *Как осуществляется первоначальная настройка Git после установки?*

Чтобы убедиться в том, что мы установили Git правильно необходимо вписать команду *git version*, если она сработала необходимо написать свое имя и почту с помощью следующих команд:

git config --global user.name "Name"

git config --global user.email "Email"

11) *Опишите этапы создания репозитория в GitHub.*

a) *Имя репозитория*. Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиториях, которые вы создавали.

b) *Описание (Description)*. Можно оставить пустым.

c) *Public/private*. Выбираем открытый (Public), НЕ ставим галочку “Initialize this repository with a README” (В README потом будет лежать какая-то основная информация, что же такое ваш проект и как с ним работать).

d) *.gitignore и LICENSE* можно сейчас не выбирать.

12) Какие типы лицензий поддерживаются GitHub при создании репозитория?

a) Лицензия Apache 2.0;

b) MIT License;

c) Публичная лицензия Eclipse 2.0;

d) GNU Affero General Public License 2.0;

И многие другие.

13) Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий?

Для этого на странице репозитория необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования.

Откройте командную строку или терминал и перейдите в каталог, куда вы хотите скопировать хранилище. Затем напишите *git clone* и введите скопированный адрес.

14) Как проверить состояние локального репозитория Git?

Проверить состояние локального репозитория можно с помощью команды *git status*.

*15) Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/ измененного файла под версионный контроль с помощью команды *git add* ; фиксации (коммита) изменений с*

помощью команды `git commit` и отправки изменений на сервер с помощью команды `git push` ?

При добавлении/изменении файла в локальных репозиторий Git состояние локального репозитория изменится на `modified` – измененное.

При добавлении нового/изменного файла под версионный контроль состояние локального репозитория изменится на `staged` – подготовленное.

При фиксации и отправки изменений на сервер состояние перейдет в `committed` – зафиксированное.

16) У Вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы можете осуществлять работу над некоторым проектом с использованием этого репозитория. Опишите последовательность команд, с помощью которых оба локальных репозитория, связанных с репозиторием GitHub будут находиться в синхронизированном состоянии.

Примечание: описание необходимо начать с команды `git clone` .

Для получения обновлений с удаленного репозитория можно воспользоваться командой: `git pull`.

Если вы изменили ваши локальные файлы, то команда `git pull` выдаст ошибку. Если вы уверены, что хотите перезаписать локальные файлы, файлами из удаленного репозитория то выполните команды:

`git fetch --all`

`git reset --hard github/master`

17) GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.

Сервисы работающие с Git:

- a) Fork;
- b) Tower;
- c) Sourcetree;

- d) SmartGit;
- e) GitKraken.

Сравню сервис Fork с GitHub. В фокусе этого инструмента скорость, дружелюбность к пользователю и эффективность. К особенностям Fork можно отнести красивый вид, кнопки быстрого доступа, встроенную систему разрешения конфликтов слияния, менеджер репозитория. Основная его черта – скорость и простота для пользователя.

18) Интерфейс командной строки является не единственным и далеко не самым удобным способом работы с Git. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git? Приведите как реализуются описанные в лабораторной работе операции Git с помощью одного из таких программных средств.

Существует и другое программное средство с графическим интерфейсом, например, Git GUI – предназначен для тех, кто не любит командную строку.

Для создания локального репозитория: в нашем графическом интерфейсе Git нажмите “Создать новый репозиторий”.

Выбрать местоположение, в котором вы хотите сохранить свой репозиторий.

Чтобы клонировать репозиторий, нажмите на ссылку “Клонировать существующий репозиторий” в окне Git GUI.

Существующий репозиторий - это тот, который уже инициализирован и / или имеет отправленные в него коммиты.

Когда мы перемещаем файлы в каталог Git, вы увидите все файлы в окне “Неустановленные изменения”. Это в основном означает, что новые файлы были добавлены, удалены, обновлены и т.д.

Когда мы нажимаем кнопку “Этап изменен”, он попытается добавить все новые файлы в индекс Git.

Так осуществляются похожие действия в Git GUI, которые были описаны в лабораторной работе.

Выводы: исследовал базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.