

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 №1.1
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Медяник Даниил 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 новый репозиторий в GitHub.

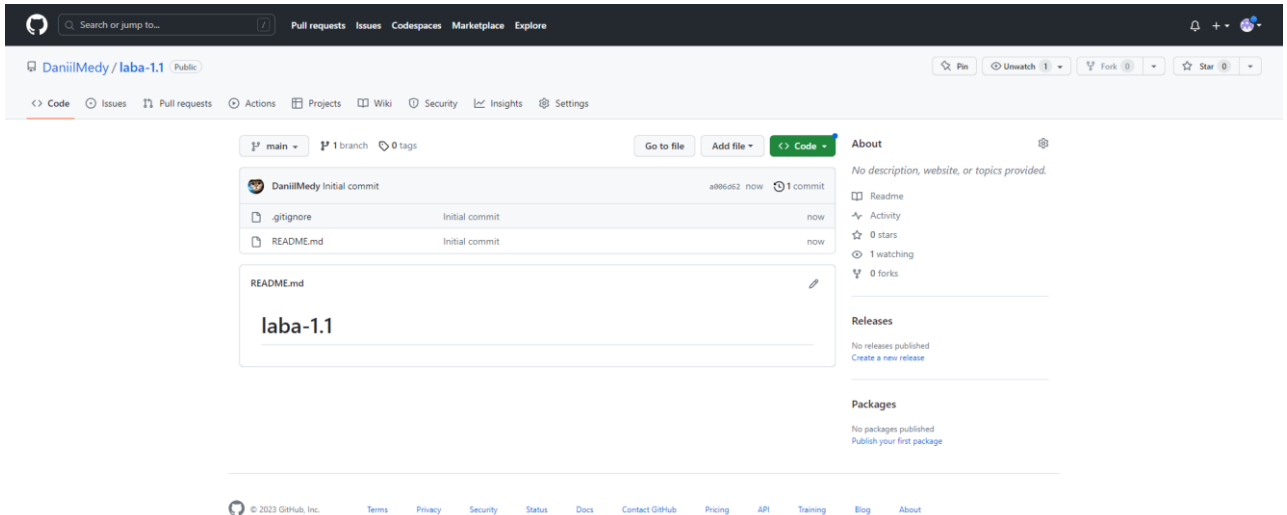


Рисунок 1. Новый репозиторий

Задание 2. Ввел в командную строку `git version`, таким образом проверил правильность работы Git'a.

```
C:\Users\Gaming-PC>git version
git version 2.39.2.windows.1
```

Рисунок 2. `git version`

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

```
C:\Users\Gaming-PC>git config --global user.name "Daniil"
C:\Users\Gaming-PC>git config --global user.email "daniilmedanik7@gmail.com"
```

Рисунок 3. Имя и почта

Задание 4. Клонировал репозиторий на свой компьютер.

```
C:\Users\Gaming-PC>git clone https://github.com/DaniilMedy/laba-1.1.git
Cloning into 'laba-1.1'...
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.
```

Рисунок 4. Клонирование репозитория

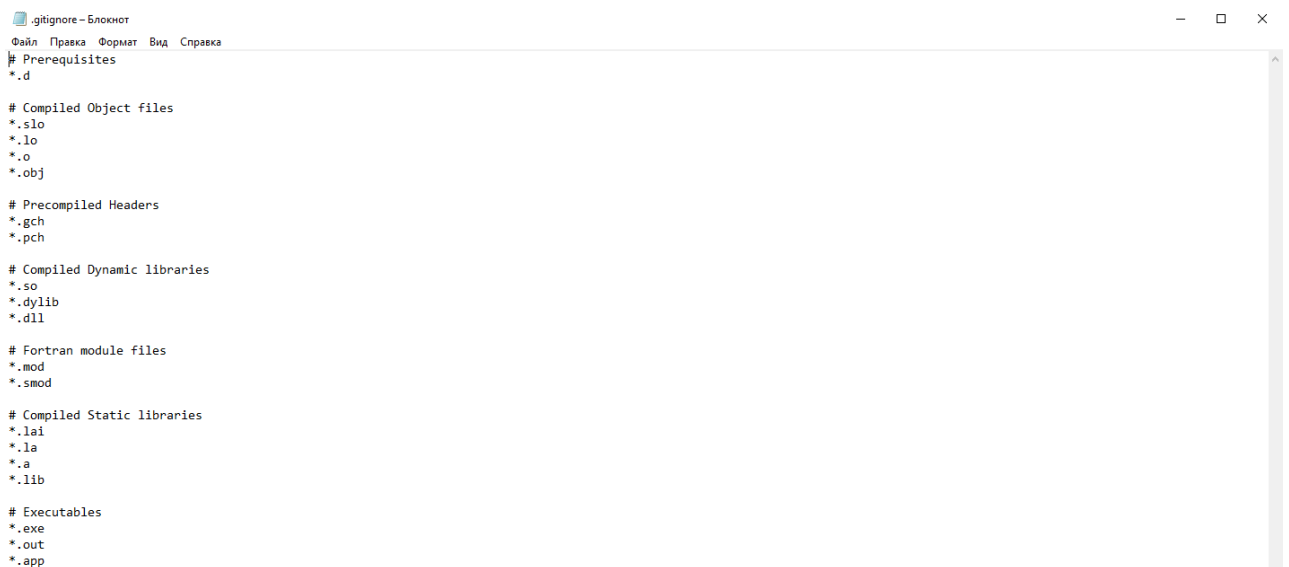
Задание 5. Проверил состояние репозитория с помощью команды git status.

```
C:\Users\Gaming-PC\laba-1.1>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Рисунок 6. Состояние репозитория

Задание 6. Дополнил файл .gitignore необходимым правилом. Сохранил, закомитил и отправил на удаленный репозиторий, с помощью команды git push.



The screenshot shows a Notepad window titled ".gitignore - Блокнот". The menu bar includes "Файл", "Правка", "Формат", "Вид", and "Справка". The content of the .gitignore file is as follows:

```
# Prerequisites
*.d

# Compiled Object files
*.slo
*.lo
*.o
*.obj

# Precompiled Headers
*.gch
*.pch

# Compiled Dynamic libraries
*.so
*.dylib
*.dll

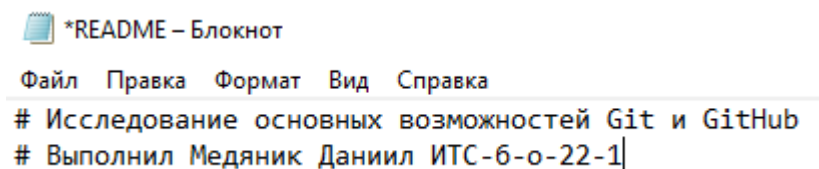
# Fortran module files
*.mod
*.smod

# Compiled Static libraries
*.lai
*.la
*.a
*.lib

# Executables
*.exe
*.out
*.app
```

Рисунок 6. Дополнение файла .gitignore

Задание 7. Внес изменения в файл README (имя и фамилию, группу) и проделал те же действия как и с .gitignore.



The screenshot shows a Notepad window titled "*README - Блокнот". The menu bar includes "Файл", "Правка", "Формат", "Вид", and "Справка". The content of the README file is as follows:

```
# Исследование основных возможностей Git и GitHub
# Выполнил Медяник Даниил ИТС-6-о-22-1|
```

Рисунок 7. Изменение файла README

Задание 8. Написал небольшую программу на языке C++, фиксировал изменения при написании в локальном репозитории, сделал не менее 7 КОММИТОВ.

```

C:\Users\Gaming-PC\laba-1.1>git add prog.cpp

C:\Users\Gaming-PC\laba-1.1>git commit -m "main"
[main c1eda49] main
1 file changed, 4 insertions(+), 3 deletions(-)

C:\Users\Gaming-PC\laba-1.1>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 286 bytes | 286.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/DaniilMedy/laba-1.1.git
6180de7..c1eda49 main -> main

C:\Users\Gaming-PC\laba-1.1>git add prog.cpp

C:\Users\Gaming-PC\laba-1.1>git commit -m "peremennie"
[main 5879656] peremennie
1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\Gaming-PC\laba-1.1>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 341 bytes | 341.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/DaniilMedy/laba-1.1.git
c1eda49..5879656 main -> main

C:\Users\Gaming-PC\laba-1.1>git add prog.cpp

C:\Users\Gaming-PC\laba-1.1>git commit -m "KLAVA"
[main ff30ca8] KLAVA
1 file changed, 4 insertions(+), 1 deletion(-)

C:\Users\Gaming-PC\laba-1.1>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 386 bytes | 386.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/DaniilMedy/laba-1.1.git
5879656..ff30ca8 main -> main

C:\Users\Gaming-PC\laba-1.1>git add prog.cpp

C:\Users\Gaming-PC\laba-1.1>git commit -m "Pi"
[main 0d389a7] Pi
1 file changed, 1 insertion(+)

C:\Users\Gaming-PC\laba-1.1>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 298 bytes | 298.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/DaniilMedy/laba-1.1.git
ff30ca8..0d389a7 main -> main

C:\Users\Gaming-PC\laba-1.1>git add prog.cpp

C:\Users\Gaming-PC\laba-1.1>git commit -m "for"
[main 8b1c45d] for
1 file changed, 8 insertions(+), 1 deletion(-)

```

Рисунок 8. Написание программы

Задание 9. Отправил изменения в удаленный репозиторий GitHub.

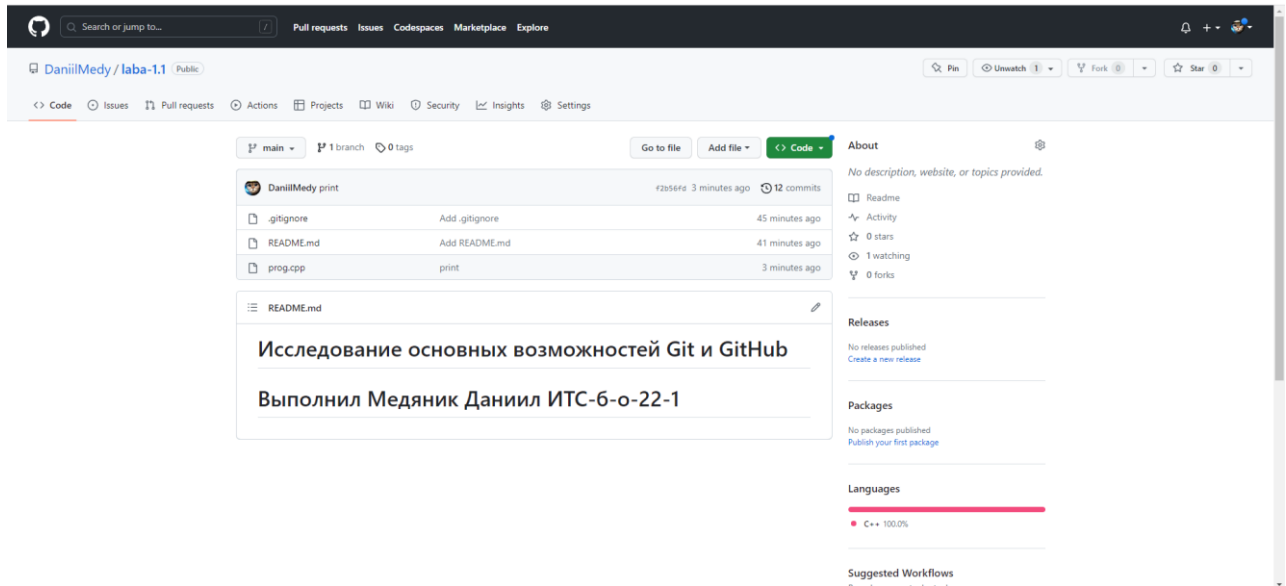


Рисунок 9. Проверка изменений в GitHub.

Ссылка на репозиторий: <https://github.com/DaniilMedy/laba-1.1>

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 распределённым 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.

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

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

в) *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).

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

12) Какие типы лицензий поддерживаются GitHub при создании репозитория?

а) Лицензия Apache 2.0;

б) MIT License;

в) Публичная лицензия Eclipse 2.0;

г) 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 измениться на *modified* – измененное.

При добавлении нового/изменного файла под версионный контроль состояние локального репозитория измениться 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 спомощью 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.

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