

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

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
дисциплины
«Основы кроссплатформенного программирования»

Выполнил:
Якушенко Антон Андреевич
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи», очная
форма обучения

(подпись)

Проверил:
Воронкин Р. А.

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

ТЕМА: ИССЛЕДОВАНИЕ ОСНОВНЫХ ВОЗМОЖНОСТЕЙ GIT И GITHUB

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

Ссылка на репозиторий: <https://github.com/Yakush766/LB2.git>

Порядок выполнения работы:

1. Использование команды git log на репозитории

```
C:\Users\anton\LB2>git log
commit 154325c1f552ad45e94274f8144e36e13d237491 (HEAD -> main, origin/main, origin/HEAD)
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 19:23:44 2024 +0300

    Внёс изменения в программе Питон

commit 9affcb329d05da8c9e60b72fb43e2a66e3c3caa0
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 19:22:08 2024 +0300

    Нашёл ошибку в программе Питон

commit a1c8c1b779142de45ecdeaccbe20bfa230428b06
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 19:15:21 2024 +0300

    Внёс изменения в файл редми

commit da2f3dc4c92cfdb6a670631b25eab1894b414105
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 19:12:24 2024 +0300

    Создал программу на Питоне

commit 9945bc4d01fde6c32b93c375bbb2ac7f98079557
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 18:38:05 2024 +0300

    Initial commit
```

Рис.1 Результат выполнения команды git log

2. Использование команды git log с аргументом -p и -2

```
C:\Users\anton\LB2>git log -p -2
commit 154325c1f552ad45e94274f8144e36e13d237491 (HEAD -> main, origin/main, origin/HEAD)
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 19:23:44 2024 +0300

    Внёс изменения в программе Питон

diff --git "a/\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py" "b/\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py"
index a136675..88def18 100644
--- "a/\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py"
+++ "b/\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py"
@@ -4,7 +4,7 @@ number = random.randint(1, 100) # Загаданное число
print("Угадайте число от 1 до 100!")
while True:
    guess = int(input("Ваш вариант: "))
-   if guess = number:
+   if guess == number:
        print("Поздравляем, вы угадали!")
        break
    print("Меньше" if guess > number else "Больше")

commit 9affcb329d05da8c9e60b72fb43e2a66e3c3caa0
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 19:22:08 2024 +0300
```

Рис. 2 Аргумент -p -2 команды git log

3. Использование опции --stat команды git log

```
C:\Users\anton\LB2>git log --stat
commit 154325c1f552ad45e94274f8144e36e13d237491 (HEAD -> main, origin/main, origin/HEAD)
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 19:23:44 2024 +0300

    Внёс изменения в программе Питон

... \321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py" | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)

commit 9affcb329d05da8c9e60b72fb43e2a66e3c3caa0
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 19:22:08 2024 +0300

    Нашёл ошибку в программе Питон

... \321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py" | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)

commit a1c8c1b779142de45ecdeaccbe20bfa230428b06
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 19:15:21 2024 +0300

    Внёс изменения в файл редми
```

Рис. 3 Опция git log --stat

4. Изменение формата вывода с помощью опции -pretty=oneline

```
C:\Users\anton\LB2>git log --pretty=oneline
154325c1f552ad45e94274f8144e36e13d237491 (HEAD -> main, origin/main, origin/HEAD) Внёс изменения в программе Питон
9affcb329d05da8c9e60b72fb43e2a66e3c3caa0 Нашёл ошибку в программе Питон
a1c8c1b779142de45ecdeaccbe20bfa230428b06 Внёс изменения в файл редми
da2f3dc4c92cfdb6a670631b25eab1894b414105 Создал программу на Питоне
9945bc4d01fde6c32b93c375bbb2ac7f98079557 Initial commit
```

Рис. 4 Вывод с опцией --pretty=oneline

5. Использование форматного вывода опции --pretty

```
C:\Users\anton\LB2>git log --pretty=format:"%h - %an, %ar : %s"
154325c - Yakush766, 29 minutes ago : Внёс изменения в программе Питон
9affcb3 - Yakush766, 30 minutes ago : Нашёл ошибку в программе Питон
a1c8c1b - Yakush766, 37 minutes ago : Внёс изменения в файл редми
da2f3dc - Yakush766, 40 minutes ago : Создал программу на Питоне
9945bc4 - Yakush766, 74 minutes ago : Initial commit
```

Рис. 5 Форматный вывод

6. Использование опции -pretty с опцией --graph

```
C:\Users\anton\LB2>git log --pretty=format:"%h %s" --graph
* 154325c Внёс изменения в программе Питон
* 9affcb3 Нашёл ошибку в программе Питон
* a1c8c1b Внёс изменения в файл редми
* da2f3dc Создал программу на Питоне
* 9945bc4 Initial commit
```

Рис. 6 Вывод с опцией --graph

7. Создание нового репозитория и внесение 7 коммитов

```
C:\Users\anton\LB2>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Рис. 7 Внесение коммитов

8. Создание тега

```
C:\Users\anton\LB2>git tag -a v1.0 -m "Версия 1.0"
```

Рис. 8 Создание тега

9. Просмотр журнала хранилища

```
C:\Users\anton\LB2>git log --graph --pretty=oneline --abbrev-commit
* 154325c (HEAD -> main, tag: v1.0, origin/main, origin/HEAD) Внёс изменения в программе Питон
* 9affcb3 Нашёл ошибку в программе Питон
* a1c8c1b Внёс изменения в файл редми
* da2f3dc Создал программу на Питоне
* 9945bc4 Initial commit
```

Рис. 9 Журнал хранилища

10. Просмотр содержимого последнего коммита

```
C:\Users\anton\LB2>git show HEAD
commit 154325c1f552ad45e94274f8144e36e13d237491 (HEAD -> main, tag: v1.0, origin/main, origin/HEAD)
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 19:23:44 2024 +0300

    Внёс изменения в программе Питон

diff --git "a/\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py" "b/\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py"
index a136675..88def18 100644
--- "a/\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py"
+++ "b/\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py"
@@ -4,7 +4,7 @@ number = random.randint(1, 100) # Загаданное число
print("Угадайте число от 1 до 100!")
while True:
    guess = int(input("Ваш вариант: "))
-   if guess == number:
+   if guess == number:
        print("Поздравляем, вы угадали!")
        break
    print("Меньше" if guess > number else "Больше")
```

Рис. 10 Содержимое последнего коммита

11. Просмотр содержимого предпоследнего коммита

```
C:\Users\anton\LB2>git show HEAD~1
commit 9affcb329d05da8c9e60b72fb43e2a66e3c3caa8
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 19:22:08 2024 +0300

    Нашёл ошибку в программе Питон

diff --git "a/\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py" "b/\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py"
index 88def18..a136675 100644
--- "a/\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py"
+++ "b/\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py"
@@ -4,7 +4,7 @@ number = random.randint(1, 100) # Загаданное число
print("Угадайте число от 1 до 100!")
while True:
    guess = int(input("Ваш вариант: "))
-   if guess == number:
+   if guess == number:
        print("Поздравляем, вы угадали!")
        break
    print("Меньше" if guess > number else "Больше")
```

Рисунок 11. Содержимое предпоследнего коммита

12. Просмотр содержимого коммита с указанным хэшем

```
C:\Users\anton\LB2>git show a1c8c1b
commit a1c8c1b779142de45ecdeaccbe20bfa230428b06
Author: Yakush766 <anton.yakushenko72@gmail.com>
Date: Thu Dec 26 19:15:21 2024 +0300

    Внёс изменения в файл редми

diff --git a/README.md b/README.md
index 3eeacd2..80e5107 100644
--- a/README.md
+++ b/README.md
@@ -1,3 @@
-# LB2
\ No newline at end of file
+Выполнил: Якушенко Антон
+
+Группа: ИТС-6-о-23-1
\ No newline at end of file
```

Рисунок 12. Содержимое коммита a1c8c1b

13. Удаление кода программы и возврат изменений командой git checkout

```
C:\Users\anton\LB2>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    "\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py"

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

C:\Users\anton\LB2>git checkout -- \320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260.py
fatal: Invalid path '/320': No such file or directory

C:\Users\anton\LB2>git checkout -- программа.py

C:\Users\anton\LB2>giy status
"giy" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.

C:\Users\anton\LB2>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Рисунок 13. Возврат изменений

14. Создание коммита и откат к предыдущей версии командой git reset

```
C:\Users\anton\LB2>git checkout -- list.txt

C:\Users\anton\LB2>git add list.txt

C:\Users\anton\LB2>git commit -m "Удаление файла"
[main 60a4386] Удаление файла
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 list.txt

C:\Users\anton\LB2>git reset --hard HEAD~1
HEAD is now at 154325c Внёс изменения в программе Питон
```

Рисунок 14. Откат коммита

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?

Локальные СКВ (системы контроля версий) менее удобны для совместной работы, поскольку требуют постоянной синхронизации изменений между участниками проекта. Централизованные СКВ, в свою очередь, могут быть уязвимы к отказам сервера, а также к потере данных, если главный репозиторий поврежден.

3. К какой СКВ относится Git?

Git относится к децентрализованным системам контроля версий.

4. В чем концептуальное отличие Git от других СКВ?

Git отличается от других СКВ тем, что он децентрализован: каждый разработчик имеет полную копию репозитория, а не только доступ к нему. Это позволяет работать автономно, делать коммиты локально и синхронизировать изменения с другими разработчиками.

5. Как обеспечивается целостность хранимых данных в Git?

Git использует криптографическую хеш-функцию SHA-1 для вычисления уникального хэша для каждого файла и коммита. Изменения в файле изменяют его хэш, что позволяет Git отслеживать изменения и обеспечивать целостность данных.

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

Файлы в Git могут находиться в 3 состояниях:

- Untracked: файл не отслеживается Git, т.е. не включен в репозиторий.
- Staged: файл был изменен и подготовлен к коммиту, т.е. включен в следующий коммит.

- Committed: файл был зафиксирован в истории репозитория.
- Связь между состояниями:
 1. Untracked -> Staged: файл переходит в состояние Staged после команды `git add`.
 2. Staged -> Committed: файл переходит в состояние Committed после команды `git commit`.

7. Что такое профиль пользователя в GitHub?

Профиль пользователя на GitHub — это публичная страница, которая представляет вас как разработчика. На ней отображаются ваши репозитории, активности, подписки, а также информация о вас: имя, местоположение, сайт, био, аватар.

8. Какие бывают репозитории в GitHub?

- Публичные: доступны всем пользователям GitHub. Их код можно просматривать, скачивать и использовать бесплатно.
- Приватные: доступны только владельцу репозитория и приглашенным сотрудникам. Для приватных репозиторий нужна платная подписка GitHub.
- Также существуют fork-репозитории: это копии публичных репозиторий, которые можно модифицировать и использовать для собственных проектов. Fork-репозитории могут быть как публичными, так и приватными.

9. Укажите основные этапы модели работы с GitHub.

Основные этапы работы с GitHub:

1. Создание репозитория: Создается новый репозиторий на GitHub, где будет храниться код проекта.
2. Клонирование репозитория: Локальная копия репозитория скачивается на компьютер с помощью команды `git clone`.
3. Работа с кодом: В локальной копии вносятся изменения в код проекта.

4. Добавление изменений в индекс: Измененные файлы добавляются в индекс с помощью команды `git add`.

5. Коммит: Изменения в индексе сохраняются в локальной истории с помощью команды `git commit`.

6. Пуш: Изменения из локального репозитория отправляются на сервер GitHub с помощью команды `git push`.

7. Pull: Изменения из репозитория GitHub скачиваются в локальную копию с помощью команды `git pull`.

10. Как осуществляется первоначальная настройка Git после установки?

`git config` - это команда, которая позволяет вам настроить Git под свой стиль работы и предпочтения. Она используется для установки глобальных, локальных и системных параметров Git.

Глобальные параметры: применяются ко всем репозиториям на вашем компьютере. Их установка осуществляется с флагом `--global`.

Локальные параметры: применяются только к текущему репозиторию. Их установка осуществляется без флага `--global`.

Системные параметры: применяются ко всем пользователям системы. Их установка осуществляется с флагом `--system`.

11. Опишите этапы создания репозитория в GitHub.

1. Авторизоваться на сайте.

2. Нажать кнопку "New" и ввести название, описание и тип доступа для нового репозитория.

3. Нажать кнопку "Create repository".

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

— MIT: Позволяет свободное использование, модификацию и распространение, включая коммерческие цели.

- Apache 2.0: Позволяет свободное использование, модификацию и распространение, включая коммерческие цели, с условием сохранения исходной лицензии.
- GPL 3.0: Позволяет свободное использование, модификацию и распространение, включая коммерческие цели, но с обязательством публикации исходного кода под той же лицензией.
- BSD 3-Clause: Позволяет свободное использование, модификацию и распространение, включая коммерческие цели, с условием сохранения имени автора и отказа от ответственности за использование программного обеспечения.
- Public Domain: Отказывается от всех прав на программное обеспечение, позволяя свободное использование, модификацию и распространение, без каких-либо ограничений.
- Также GitHub предоставляет возможность выбрать "None" (без лицензии), что означает, что вы не предоставляете никаких прав на использование вашего кода.

13. Как осуществляется клонирование репозитория GitHub?

Зачем нужно клонировать репозиторий?

Клонирование репозитория GitHub осуществляется с помощью команды git clone: `git clone <URL_репозитория>`

Клонирование репозитория создает локальную копию репозитория на вашем компьютере, что позволяет вам:

Работать с кодом локально: Вносить изменения, создавать ветки, использовать Git без подключения к Интернету.

Вносить вклад в проект: Вносить изменения в код проекта и отправлять их на GitHub.

Сохранять историю изменений: Локальная копия сохраняет полную историю всех изменений в репозитории.

Совместная работа: Клонирование позволяет нескольким разработчикам работать над проектом одновременно и синхронизировать свои изменения.

14. Как проверить состояние локального репозитория Git?

Чтобы проверить состояние локального репозитория Git, используется команда `git status`.

Она покажет:

- Измененные файлы: Файлы, которые были изменены, но еще не добавлены в индекс.
- Файлы, добавленные в индекс: Файлы, которые были изменены и добавлены в индекс для следующего коммита.
- Неотслеживаемые файлы: Файлы, которые не отслеживаются Git (не входят в репозиторий).
- Текущая ветка: Ветка, в которой вы находитесь.
- Состояние ветки: Информировать о том, соответствует ли локальная копия ветки удаленной копии на GitHub.

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` ?

После добавления или изменения файла в локальном репозитории Git он переходит в состояние "Untracked" (не отслеживается). Git еще не знает об этом файле и не включает его в историю изменений.

Команда `git add <имя_файла>` добавляет файл в индекс Git. Файл переходит в состояние "Staged" (подготовлен к коммиту). Теперь Git знает об этом файле и будет включать его в следующий коммит.

Команда `git commit -m "Сообщение о коммите"` создает новый коммит в локальной истории Git. Файл переходит в состояние "Committed"

(зафиксирован). Изменения, которые были в индексе, теперь стали частью истории репозитория.

Команда `git push origin main` отправляет изменения из локальной истории на сервер GitHub. Локальные коммиты теперь доступны в удаленном репозитории.

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`.

1. Клонировать репозиторий на оба компьютера: `git clone <URL_репозитория>`.
2. Работать над проектом на первом компьютере, фиксировать изменения (`git add`, `git commit`).
3. Отправить изменения на GitHub: `git push origin main`.
4. Обновить локальный репозиторий на втором компьютере: `git pull origin main`.
5. Повторить шаги 2-4 для второго компьютера.

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

Bitbucket: Разработан Atlassian, известен своей интеграцией с другими продуктами Atlassian, такими как Jira и Confluence. Предлагает как бесплатные, так и платные планы с поддержкой приватных репозиторий.

- GitLab: Открытый сервис с возможностью развертывания на собственных серверах. Предлагает множество функций для управления кодом, CI/CD, и безопасности.

- Azure DevOps: Облачный сервис от Microsoft, включающий в себя систему контроля версий Git, CI/CD, управление проектами и другие инструменты.

GitHub и Bitbucket - популярные платформы для хостинга Git-репозиториях, но у них есть некоторые различия. GitHub предлагает бесплатные публичные репозитории и платные приватные, в то время как Bitbucket предоставляет бесплатные приватные репозитории для небольших команд и платные планы с расширенными функциями.

GitHub интегрируется с различными сервисами, такими как Travis CI и CircleCI, в то время как Bitbucket отлично интегрируется с продуктами Atlassian, такими как Jira и Confluence. GitHub предлагает более широкий набор функций для управления кодом, сотрудничества и развертывания, в то время как Bitbucket предлагает набор функций, достаточный для большинства проектов. Сообщество разработчиков на GitHub значительно больше, чем на Bitbucket, но Bitbucket обеспечивает хорошую поддержку Atlassian.

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.

GitHub Desktop: Разработан самой компанией GitHub, интегрирован с веб-платформой GitHub и предлагает простой и интуитивно понятный интерфейс.

- GitKraken: Известен своим стильным дизайном, большим количеством функций и поддержкой многих систем контроля версий, включая Git, Mercurial и SVN.

- Sourcetree: Разработан Atlassian, производителем Jira и Bitbucket, имеет хорошую интеграцию с этими сервисами и предлагает широкий набор функций для управления репозиториями.

- TortoiseGit: Бесплатная программа с открытым исходным кодом для Windows, известна своей интеграцией в контекстное меню Проводника Windows и простотой использованием.

Как реализуются операции Git в GitHub Desktop

В GitHub Desktop необходимо нажать кнопку "Clone a repository" и ввести URL репозитория. После чего выбрать папку для клонирования.

Вывод: в ходе лабораторной работы были исследованы базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.