МИНИСТЕРСТВО НАУКИ ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций.

ОТЧЁТ

по лабораторной работе №1

Дисциплина: «Программирование на Python»

Тема: «Исследование основных возможностей Git и GitHub»

Выполнил:
студент 2 курса группы ИВТ-б-о-22-1

Бабенко Артём Тимофеевич

Проверил:

Доцент кафедры инфокоммуникаций

Воронкин Р.А

Работа защищена с оценкой: _____

Ставрополь, 2023

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

Теория:

Системы контроля версий

Что такое контроль версий, и зачем он вам нужен? 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. Программисты обычно помещают в систему контроля версий исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа. Если вы графический или веб дизайнер и хотели бы хранить каждую версию изображения или макета, то пользоваться системой контроля версий будет очень мудрым решением. СКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь СКВ, вы всё испортите или потеряете файлы, всё можно будет легко восстановить. Вдобавок, накладные расходы будут очень маленькими. Локальные системы контроля версий

Многие люди в качестве метода контроля версий применяют копирование файлов в отдельную директорию (возможно даже, директорию с отметкой по времени, если они достаточно сообразительны). Данный подход очень распространён из-за его простоты, однако он невероятно сильно подвержен появлению ошибок. Можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели.

Для того, чтобы решить эту проблему, программисты давным-давно разработали локальные СКВ с простой базой данных, которая хранит записи о всех изменениях в файлах, осуществляя тем самым контроль ревизий

Ход работы:

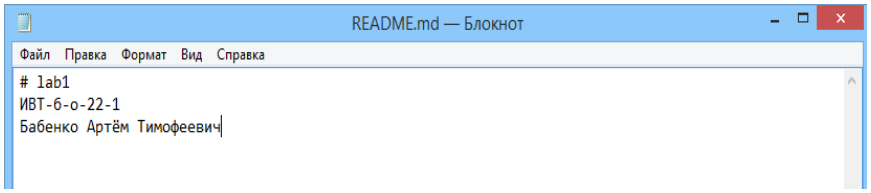

Рисунок 1. Добавление имени и группы

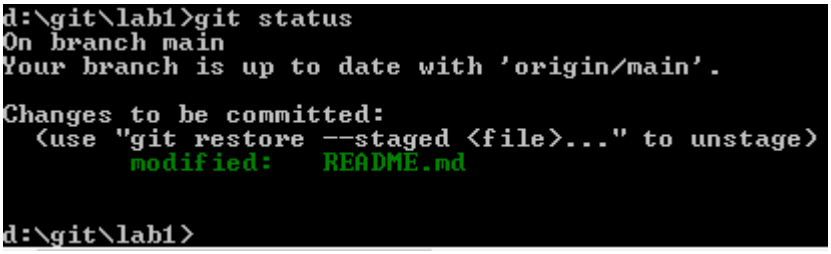

Рисунок 2. Изменения в Git CMD

```
d:\git\lab1>git commit -m "Add grop info"
[main 66e9772] Add grop info
 1 file changed, 2 insertions(+), 1 deletion(-)

d:\git\lab1>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

d:\git\lab1>
```

Рисунок 3. Первый коммит

```
d:\git\lab1>git add .

d:\git\lab1>git commit -m " Add main code"
[main 9e90dcf]  Add main code
 1 file changed, 11 insertions(+), 1 deletion(-)

d:\git\lab1>git status
On branch main
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

d:\git\lab1>git add .

d:\git\lab1>git status
On branch main
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   code/code.txt


d:\git\lab1>git commit -m "Add Name in code"
[main 1b57cc7] Add Name in code
 1 file changed, 3 insertions(+), 1 deletion(-)

d:\git\lab1>git status
On branch main
Your branch is ahead of 'origin/main' by 6 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

d:\git\lab1>_
```

Рисунок 4. Коммит с добавлением кода

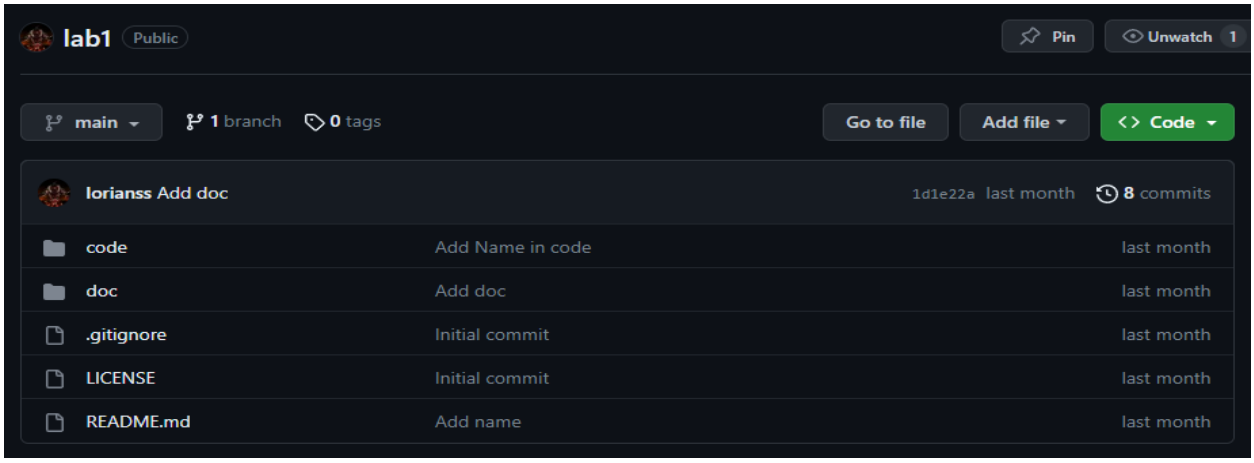

Рисунок 5. Окно браузера после внесений изменений

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

Контрольные вопросы:

1. Что такое СКВ и каково ее назначение?

Системы контроля версий — это программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени. В свете усложнения сред разработки они помогают командам разработчиков работать быстрее и эффективнее.

2 В чем недостатки локальных и централизованных СКВ?

Централизованными легко управлять из-за наличия единственного сервера. Но при этом наличие централизованного сервера приводит к возникновению единой точки отказа в виде этого самого сервера. В случае отключения этого сервера разработчики не смогут выкачивать файлы. Самым худшим сценарием является физическое уничтожение сервера (или вылет жесткого диска), он приводит к потерю кодовой базы.

Для устранения единой точки отказа используются распределенные системы контроля версий. Они подразумевают, что клиент выкачает себе весь репозиторий целиком заместо выкачки конкретных интересующих клиента файлов. Если умрет любая копия репозитория, то это не приведет к потере кодовой базы, поскольку она может быть восстановлена с компьютера любого разработчика. Каждая копия является полным бэкапом данных.

3 К какой СКВ относится Git?

Git — система управления версиями с распределенной архитектурой.

4 В чем концептуальное отличие Git от других СКВ?

В отличие от некогда популярных систем вроде CVS и Subversion (SVN), где полная история версий проекта доступна лишь в одном месте, в Git каждая рабочая копия кода сама по себе является репозиторием.

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. Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его философии. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git.

6. В каких состояниях могут находится 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 ваша публичная страница на GitHub, как и в социальных сетях. Когда вы ищете работу в качестве программиста, работодатели могут посмотреть ваш профиль GitHub и принять его во внимание, когда будут решать, брать вас на работу или нет.

8. Какие бывают репозитории в GitHub?
Репозиторий Git бывает локальный и удалённый. Локальный репозиторий — это подкаталог . git, создаётся (в пустом виде) командой git init и (в непустом виде с немедленным копированием содержимого родительского удалённого репозитория и простановкой ссылки на родителя) командой git clone.

9. Укажите основные этапы модели работы с GitHub.
Стандартный подход к работе с проектом состоит в том, чтобы иметь локальную копию репозитория и фиксировать ваши изменения в этой копии, а не в удаленном репозитории, размещенном на GitHub. Этот локальный репозиторий имеет полную историю версий проекта, которая может быть полезна при разработке без подключения к интернету. После того, как вы что-то изменили в локальном, вы можете отправить свои изменения в удаленный репозиторий, чтобы сделать их видимыми для других разработчиков

10. Как осуществляется первоначальная настройка Git после установки?
Чтобы убедиться, что Git был успешно установлен, введите команду ниже в терминале, чтобы отобразить текущую версию вашего Git:
git version
Если она сработала, добавим в настройки Git ваше имя, фамилию и адрес электронной почты, связанный с вашей учетной записью GitHub:
git config --global user.name
git config --global user.email

11. Опишите этапы создания репозитория в GitHub.
В правом верхнем углу, рядом с аватаром есть кнопка с плюсиком, нажимая которую мы переходим к созданию нового репозитория
В результате будет выполнен переход на страницу создания репозитория. Наиболее важными на ней являются следующие поля:

- Имя репозитория. Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиториев, которые вы создавали.
- Описание (Description). Можно оставить пустым.
- 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).
- gitignore и LICENSE можно не выбирать.

После заполнения этих полей нажимаем кнопку Create repository.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?
Apache License 2.0, GNU General Public License v3.0 MIT License и т.д.

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 адрес:
git clone “адрес”
Это нужно, что бы получить локальное хранилище проекта.

14. Как проверить состояние локального репозитория Git?
Ввести команду 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 ?

Файл добавленный/изменённый в локальном репозетории будет сохранён, а позже отправлен в исходный репозеторий, после команды push.

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 .
git clone
git status
git pull

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

GitFlic. Российский GitHub.
Интерфейсные решения соответствуют очень давнему 2008-2010 годам. Версионность инструментов оставляет желать лучшего. Проект очень сырой, они не предоставляют уникальных “фич” по сравнению с конкурентами.

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.

Visual studio Code

Удобный интерфейс, позволяющий через интуитивно понятные кнопки, выполнять гит команды внутри программы. Так же есть возможность самому вписывать команды, с подсказками от программы.