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 ПРАКТИЧЕСКОЙ РАБОТЕ №1
дисциплины
«Основы кроссплатформенного программирования»
Исследование
основных возможностей Git и GitHu

Выполнила студентка группы
ИВТ-б-о-21-1
Яковлева Е.А. « »_____20__г.
Подпись студента_____
Работа защищена
« »_____20__г.

Проверил доцент
Кафедры инфокоммуникаций,
старший преподаватель
Воронкин Р.А.

_____
(подпись)

Ставрополь, 2022 г.

Тема: исследование основных возможностей GIT и GITHub.

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

Порядок выполнения работы:

№1.Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add .gitignore).

1) Регистрируемся на GitHub.

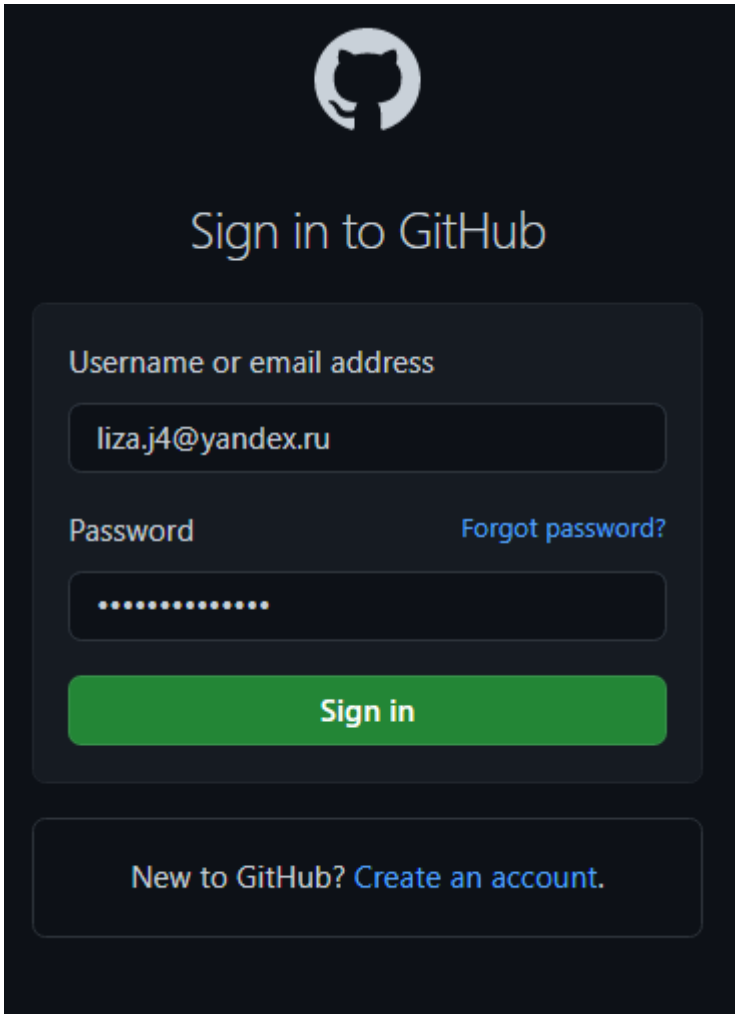


Рисунок 1 – Вход на GitHub

2) Создадим новый репозиторий, нажимая New repository, назовём и проведём настройки.

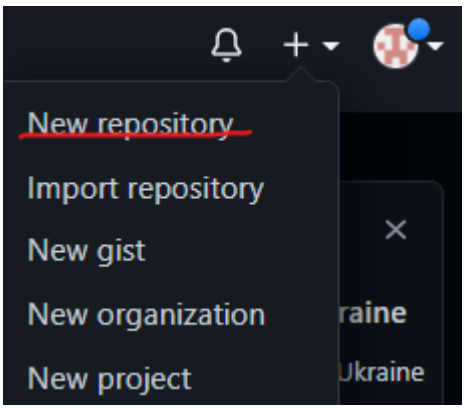


Рисунок 2 – Выбор нового репозитория

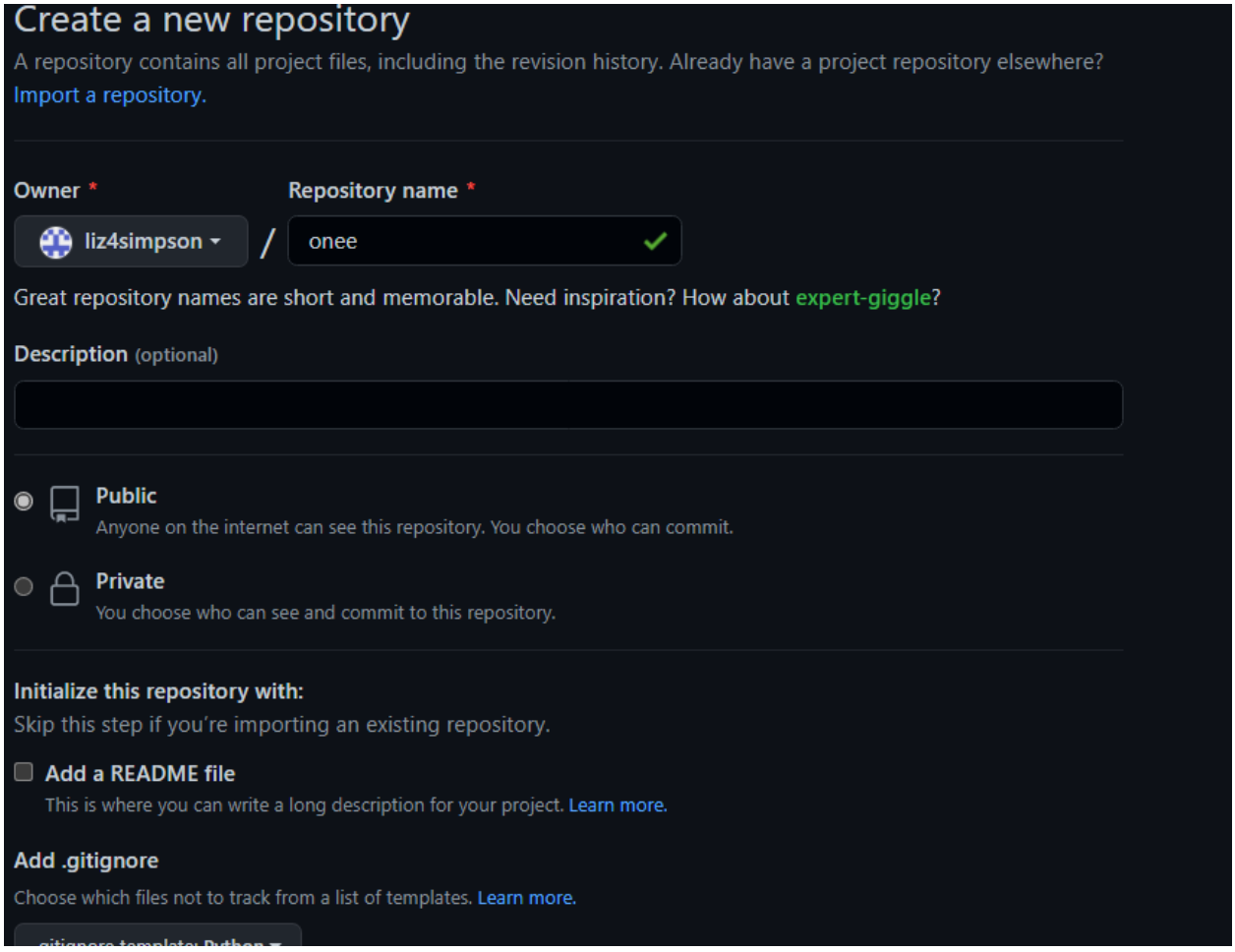
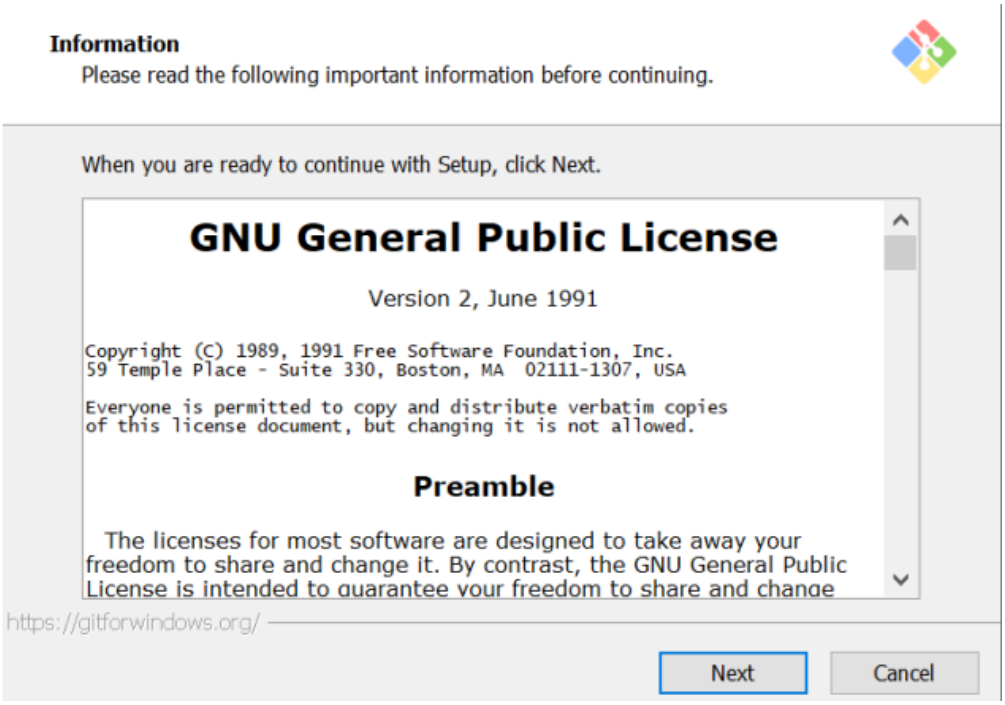


Рисунок 3 – Создание нового репозитория.

№2.Выполнить клонирование созданного репозитория на рабочий стол.

1) Для этого скачаем и установим Git на компьютер.



Проверим установлен ли Git.

Рисунок 6 – Git установлен успешно

2) Произведём клонирования репозитория на компьютер.

Рисунок 7 – Ссылка для клонирования репозитория.

3) Откроем командную строку Windows и при помощи неё скопируем необходимые данные.

Рисунок 8 – При помощи команды cd /d переходим в необходимый каталог.

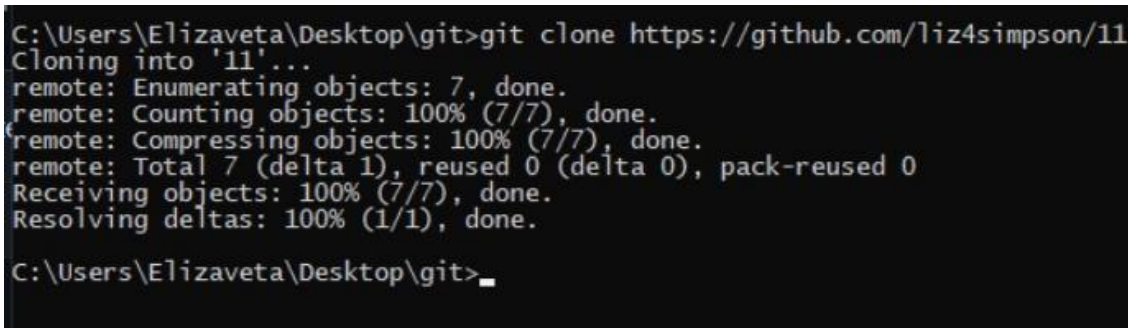


Рисунок 9 – С помощью команды git clone копируем необходимые данные

Добавим в файл README.md информацию о группе и ФИО студента, выполняющего лабораторную работу

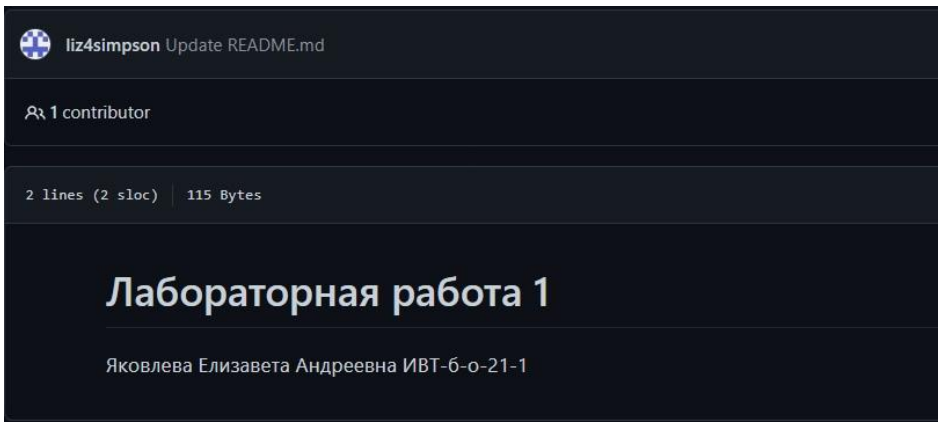


Рисунок 10 – Добавление необходимой информации в файл README.md

Дополним файл .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.

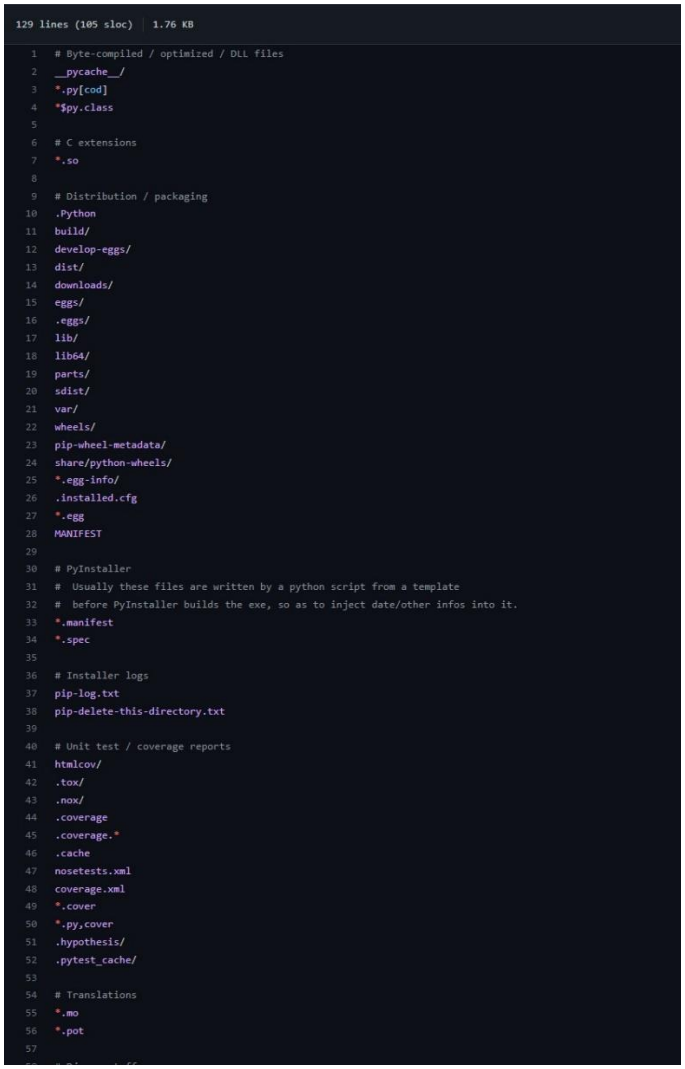


Рисунок 11 – Изменения необходимой информации в файле .gitignore

3. Написать небольшую программу на выбранном языке программирования.

1) Создадим в репозитории файл.

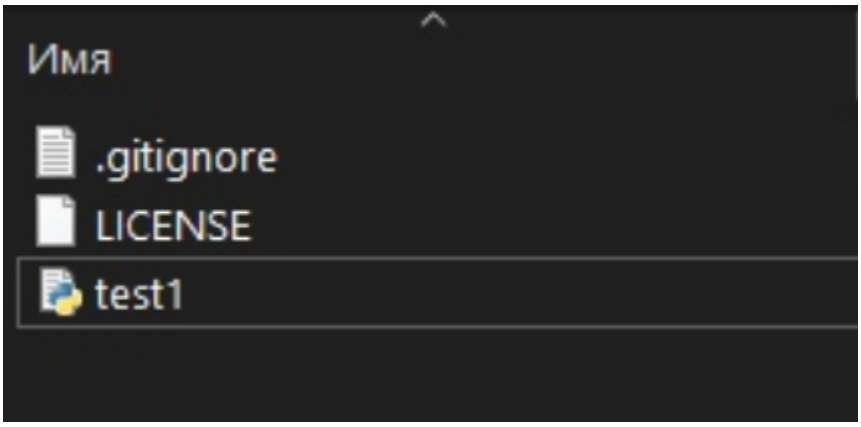


Рисунок 13 – Созданный файл Python

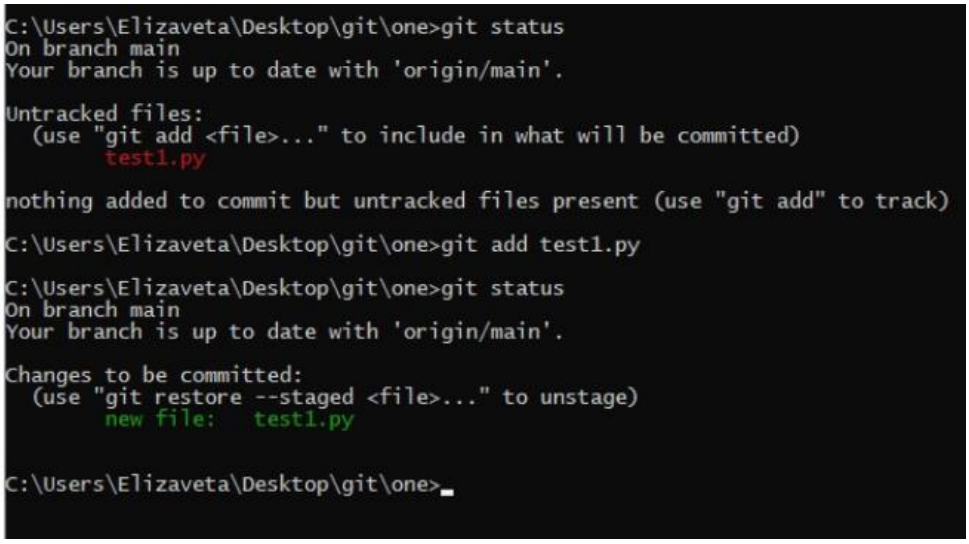


Рисунок 14 – Добавление нового файла

4) Добавим комментарий к изменению используя команду: git commit -m "Update file"

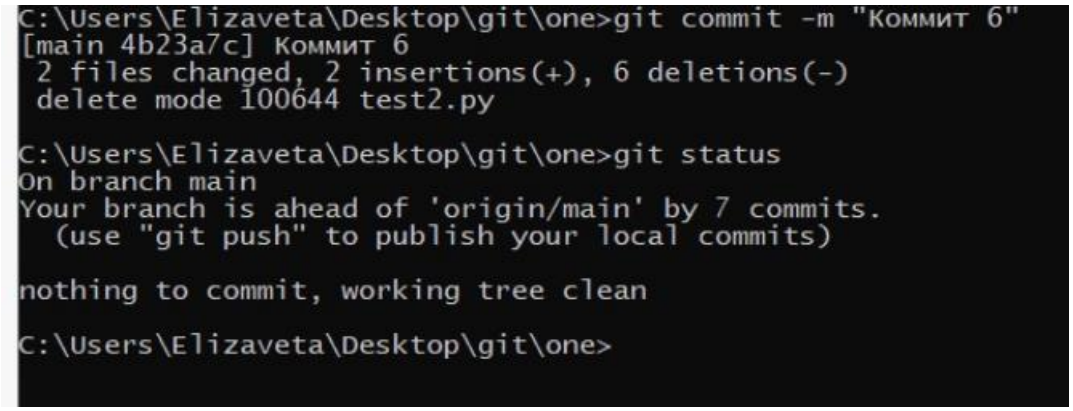


Рисунок 15 – Сделанные 7 коммитов репозитория

5) Отправим изменения в локальном репозитории в удалённый репозиторий GitHub используя команду: git push.

```
C:\Users\Elizaveta\Desktop\git\one>git push --force
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 383 bytes | 191.00 KiB/s, do
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/liz4simpson/one.git
   4b23a7c..f64e301  main -> main
```

Рисунок 16 – Отправка изменения на удалённый репозиторий

6) Проконтролируем изменения, произошедшие в репозитории GitHub

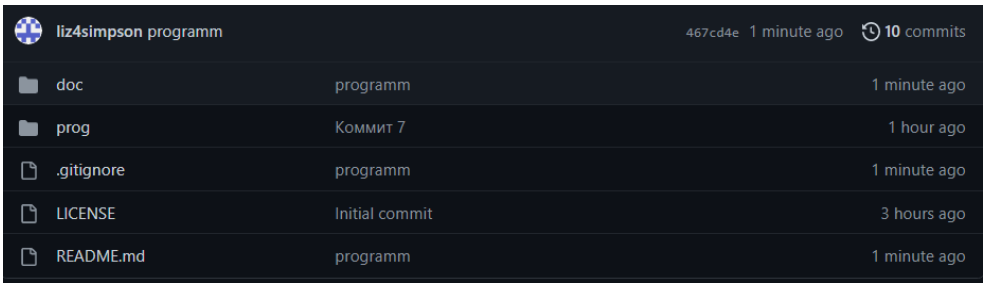


Рисунок 170– Проверка изменений

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: многие в качестве метода контроля версий применяют копирование файлов в отдельную директорию. Такой подход очень распространён из-за его простоты, однако он невероятно сильно подвержен появлению ошибок. Можно легко забыть, в какой директории находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы

хотели.

Централизованные СКВ: 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 системам, поэтому не зависит от

центрального сервера, где хранятся файлы.

4. В чем концептуальное отличие Git от других СКВ?

Git не хранит и не обрабатывает данные таким же способом как другие

СКВ. Каждый раз, когда вы делаете коммит, т. е. сохраняете состояние своего

проекта в Git, система запоминает, как выглядит каждый файл в этот момент,

и сохраняет ссылку на этот снимок. Следует, что Git эффективен в хранении

бэкапов, поэтому известно мало случаев, когда кто-то терял данные при его

использовании.

5. Как обеспечивается целостность хранимых данных в Git?

В Git для всего вычисляется хеш-сумма, и только потом происходит

сохранение. В дальнейшем обращение к сохраненным 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 основы.

В итоге информация не теряется во время её передачи и файл не повредится

без ведома Git.

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

Зафиксированный файл – файл уже сохранён в вашей локальной базе.

Измененный файл – файл, который поменялся, но ещё не был зафиксирован.

Подготовленный файл — это изменённый файл, отмеченный для включения в следующий коммит.

7. Что такое профиль пользователя в GitHub?

Профиль – ваша публичная страница на GitHub, как и в социальных

сетях. Когда мы ищем работу в качестве программиста, работодатели могут

посмотреть наш профиль GitHub и принять его во внимание, когда будут

решать, брать нас 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. Практически все обычные операции с

системой контроля версий, такие, как коммит и слияние, производятся только

с локальным репозиторием.

Удалённый доступ к репозиториям Git обеспечивается gitdaemon, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в

дистрибутив Git и является наряду с SSH наиболее распространённым и

надёжным методом доступа. Удалённый репозиторий можно только синхронизировать с локальным как «вверх» (push), так и «вниз» (pull).

9. Укажите основные этапы модели работы с GitHub.

1) Регистрация.

2) Создание репозитория.

3) Клонирование репозитория.

10. Как осуществляется первоначальная настройка Git после установки?

1) Убедимся, что Git установлен используя команду: git version;

2) Перейдём в папку с локальным репозиторием, используя команду: cd /d;

3) Свяжем локальный репозиторий и удалённый командами:

git config --global user.name <YOUR_NAME>

git config --global user.email <EMAIL>

11. Опишите этапы создания репозитория в GitHub.

1) В правом верхнем углу, рядом с аватаром есть кнопка с плюсиком, нажимая которую переходим к созданию нового репозитория.

2) 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 сейчас не выбирать. После

заполнения этих полей нажимаем кнопку Create repository.

12. Какие типы лицензий поддерживаются GitHub при создании

репозитория?

Microsoft Reciprocal License, The Code Project Open License (CPOL), The

Common Development and Distribution License (CDDL), The Microsoft Public

License (Ms-PL), The Mozilla Public License 1.1 (MPL 1.1), The Common Public

License Version 1.0 (CPL), The Eclipse Public License 1.0, The MIT License, The

BSD License, The Apache License, Version 2.0, The Creative Commons

Attribution-ShareAlike 2.5 License, The zlib/libpng License, A Public Domain

dedication, The Creative Commons Attribution 3.0 Unported License, The

Creative Commons Attribution-Share Alike 3.0 Unported License, The Creative

Commons Attribution-NoDerivatives 3.0 Unported, The GNU Lesser General

Public License (LGPLv3), The GNU General Public License (GPLv3).

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

1) После создания репозитория его необходимо клонировать на ваш компьютер. Для этого на странице репозитория необходимо найти кнопку

Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для

клонирования.

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

адрес.

14. Как проверить состояние локального репозитория Git?

Используя 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 ?

Файлы обновятся на удвлённом 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) Клонируем репозиторий на каждый из компьютеров, используя команду git clone и ссылку.

2) Для синхронизации изменений используем команду 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.

GitLab — альтернатива GitHub номер один. GitLab предоставляет не только веб-сервис для совместной работы, но и программное обеспечение с открытым исходным кодом.

SourceForge — ещё одна крупная альтернатива GitHub, сконцентрировавшаяся на Open Source. Многие дистрибутивы и приложения Linux обитают на SourceForge

Launchpad — платформа для совместной работы над программным обеспечением от Canonical, компании-разработчика Ubuntu. На ней размещены PPA-репозитории Ubuntu, откуда пользователи загружают приложения и обновления.

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.

1) GitHub Desktop это бесплатное приложение с открытым исходным кодом, разработанное GitHub. С его помощью можно взаимодействовать с GitHub, а также с другими платформами (включая GitLab).

2) Fork это весьма продвинутый GUI-клиент для macOS и Windows (с бесплатным пробным периодом). В фокусе этого инструмента скорость, дружественность к пользователю и эффективность. К особенностям Fork можно отнести красивый вид, кнопки быстрого доступа, встроенную систему разрешения конфликтов слияния, менеджер репозитория, уведомления GitHub.

3) Sourcetree это бесплатный GUI Git для macOS и Windows. Его применение упрощает работу с контролем версий и позволяет сфокусироваться на действительно важных задачах.

4) martGit это Git-клиент для Mac, Linux и Windows. Имеет богатый функционал. В арсенале SmartGit вы найдете CLI для Git, графическое отображение слияний и истории коммитов, SSH-клиент, Git-Flow, программу для разрешения конфликтов слияния.

Вывод: в ходе лабораторной работы исследованы базовые возможности системы контроля версий Git и веб-сервиса для хостинга ITпроектов GitHub. Создан GitHub репозиторий, клонирован репозиторий на компьютер, написана небольшая программа, изменения отправлены на удалённый репозиторий