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 «Программирование на Python»

Выполнил:
Лейс Алексей Вячеславович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем»

_____
(подпись)

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

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2022 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:

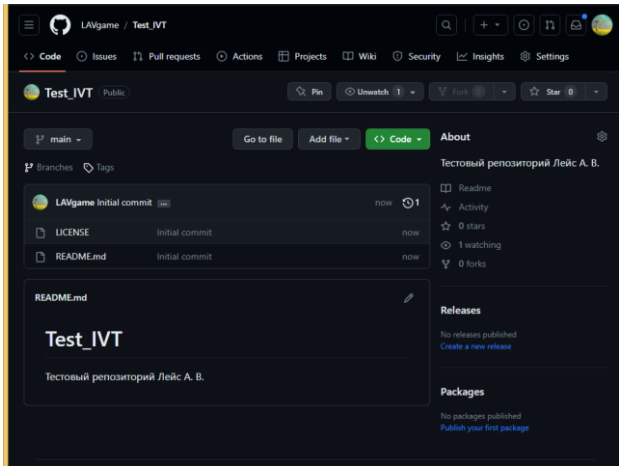

Рисунок 1 – Создание аккаунта и тестового репозитория на GitHub

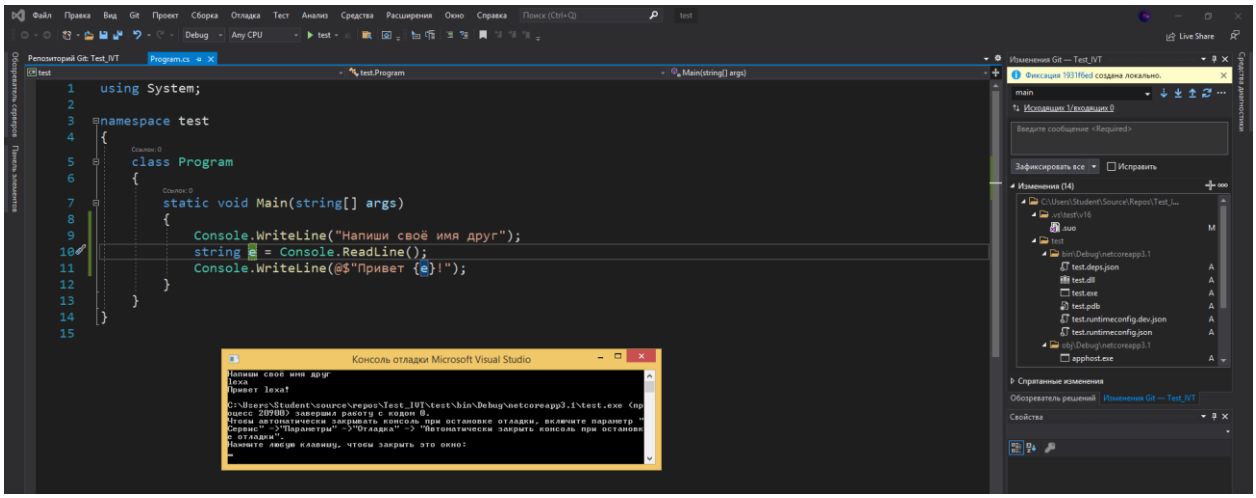

Рисунок 2 – Создание простой программы на C# и загрузка на GitHub

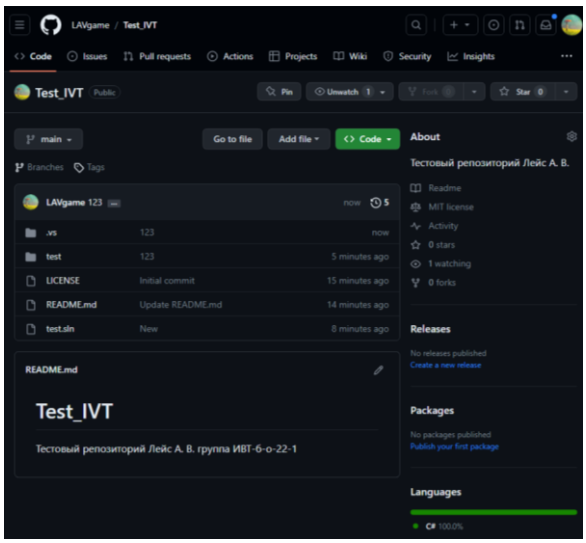

Рисунок 3 – Проект на GitHub после выполнения загрузки на него.

Те же действия с консолью:

```
Microsoft Windows [Version 6.3.9600]
(c) Корпорация Майкрософт (Microsoft Corporation), 2013. Все права защищены.

C:\Users\Student>github
"github" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.

C:\Users\Student>git coomit
git: 'coomit' is not a git command. See 'git --help'.

The most similar command is
        commit

C:\Users\Student>git clone https://github.com/LAVgame/Test_IVT
Cloning into 'Test_IVT'...
remote: Enumerating objects: 72, done.
remote: Counting objects: 100% (72/72), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 72 (delta 13), reused 60 (delta 11), pack-reused 0 eceiving object
s:  45% (33/72), 852.00 KiB | 1
Receiving objects: 100% (72/72), 871.71 KiB | 1.60 MiB/s, done.
Resolving deltas: 100% (13/13), done.

C:\Users\Student>
```

Рисунок 4 – Клонирование проекта с GitHub

```
using System;

namespace test
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Напиши своё имя прохожий");
            string e = Console.ReadLine();
            Console.WriteLine(@$"Привет {e}!");
        }
    }
}
```

Рисунок 5 – Что-то меняем в коде

```
vipal@LAV MINGW64 /i/ИВТ-б-о-22-1/Прогграммирование на Python/1/Test_IVT (main)
$ git commit -m "one commit"
[main 97b0ee7] one commit
 14 files changed, 67 insertions(+), 35 deletions(-)
 create mode 100644 .vs/ProjectEvaluation/test.metadata.v7.bin
 create mode 100644 .vs/ProjectEvaluation/test.projects.v7.bin
 create mode 100644 .vs/test/FileContentIndex/64512b9d-8926-4805-82fc-ca6d3944ac
9d.vsidx
 create mode 100644 .vs/test/v17/.futdcache.v2
 create mode 100644 .vs/test/v17/.suo
```

Рисунок 6 – Делаем коммит

```
vipal@LAV MINGW64 /i/ИВТ-б-о-22-1/Прогграммирование на Python/1/Test_IVT (main)
$ git push
Enumerating objects: 43, done.
Counting objects: 100% (43/43), done.
Delta compression using up to 12 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (26/26), 63.90 KiB | 4.56 MiB/s, done.
Total 26 (delta 9), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (9/9), completed with 8 local objects.
To https://github.com/LAVgame/Test_IVT
   3941837..97b0ee7  main -> main
```

Рисунок 7 – отправляем коммит

LAVgame one commit

Code | Blame | 14 lines (13 loc) · 304 Bytes | Code 55% faster with GitHub Copilot

```
1   using System;
2
3   namespace test
4   {
5       class Program
6       {
7           static void Main(string[] args)
8           {
9               Console.WriteLine("Напиши своё имя прохожий");
10              string e = Console.ReadLine();
11              Console.WriteLine(@$"Привет {e}!");
12          }
13      }
14  }
```

Рисунок 8 – изменения на GitHub

Ответы на 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 распределённым системам контроля версий.

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. Эти системы (CVS, Subversion, Perforce, Bazaar и т. д.) представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени (обычно это называют контролем версий, основанным на различиях). Git не хранит и не обрабатывает данные таким способом. Вместо этого, подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создаёт ссылку на предыдущую версию идентичного файла, который уже сохранён. Git представляет свои данные как, скажем, поток снимков

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 основы. В итоге информация не теряется во время её передачи и файл не повредится 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 или аккаунт конкретного пользователя на платформе где храниться проекты которыми он занимается.

8. Какие бывают репозитории в GitHub?

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 git- daemon, SSHили 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 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 сейчас не выбирать.
После заполнения этих полей нажимаем кнопку Create repository.
Отлично, ваш репозиторий готов!

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?

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.

14. Как проверить состояние локального репозитория Git?

Локальный репозиторий включает в себя все те же файлы, ветки и историю коммитов, как и удаленный репозиторий. Введите эту команду, чтобы проверить состояние вашего репозитория: 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 ?

Коммит добавляет изменения только в ваш локальный репозиторий. Если вы хотите распространить их в исходный репозиторий на GitHub, вам нужно использовать 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 .

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.

К примеру, существует GitKraken, по-моему, мнению он удобен визуализации ветвей и коммитов, в целом очень удобный интерфейс. Но я пользуюсь стандартным github потому что очень удобная встроенная система в visual studio code, тебе буквально ничего лишнего не нужно скачивать достаточно зайти в аккаунт и пользоваться.

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.

Да буквально взять то же Git Gui с уже простым интерфейсам, так же GitHub desktop тоже вполне удобный не консольный интерфейс.

Вывод: В результате выполненной работы я исследовал базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.