

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 ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины «Основы программной инженерии»

Выполнил:
Яблоновский Дмитрий
Николаевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Руководитель практики:
Богданов С.С., ассистент
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.

Ход выполнения работы:

1. Создать общедоступный репозиторий на GitHub с использованием лицензии MIT и выбранным языком программирования:

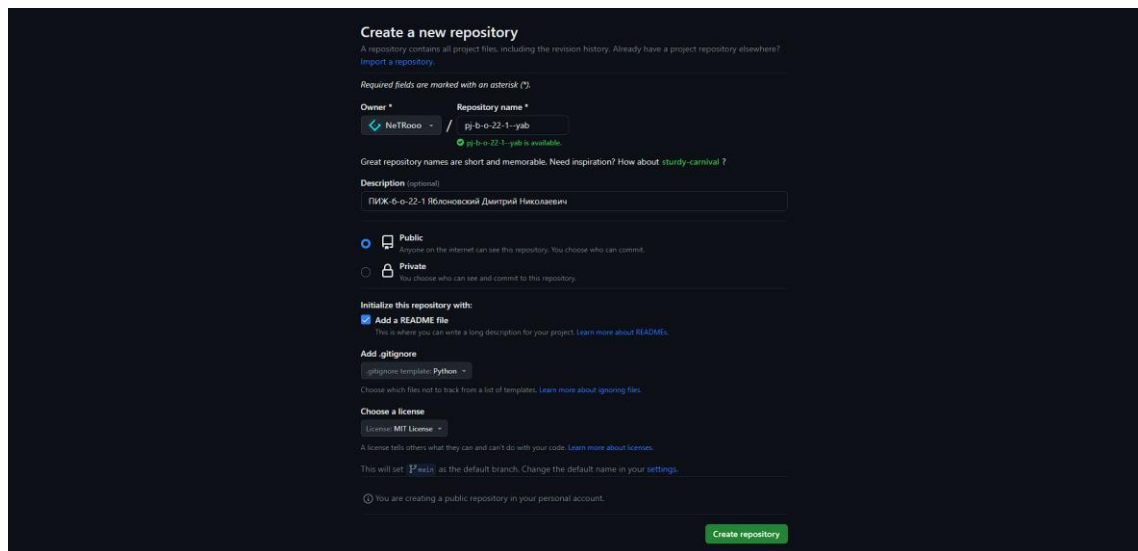


Рисунок 1 – Создание общедоступного репозитория на GitHub с заданными настройками

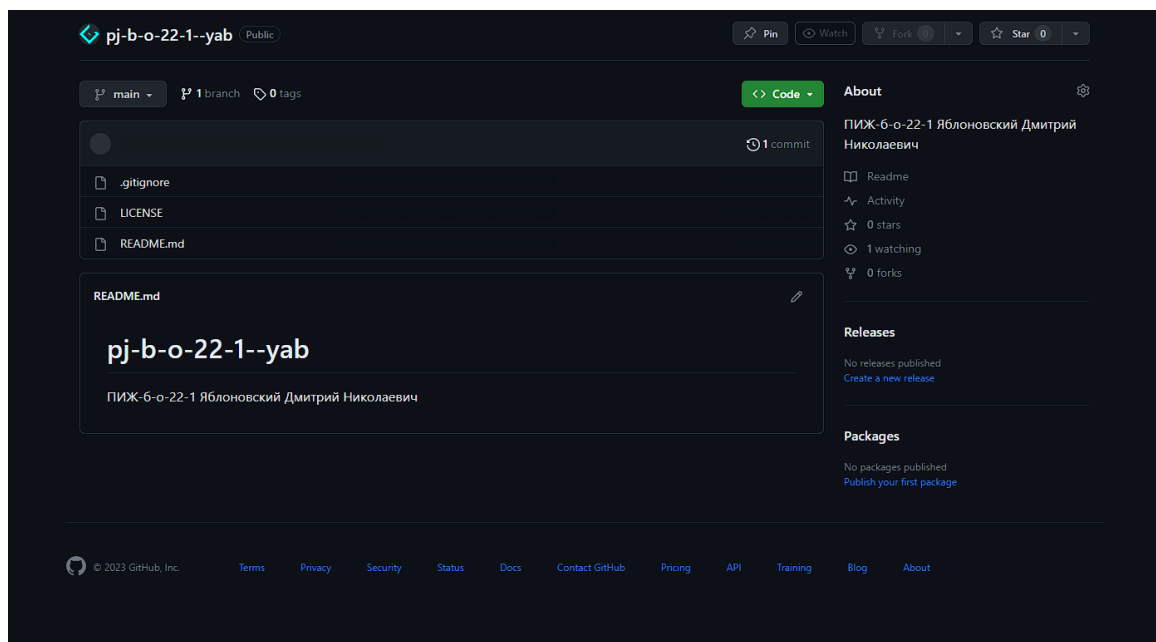


Рисунок 2 – Результат создания репозитория

2. Клонировать репозиторий на рабочий компьютер:

```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Установите последнюю версию PowerShell для новых функций и улучшения! https://aka.ms/PSWindows

PS D:\utorrent\vh> git clone https://github.com/NeTROoo/pj-b-o-22-1-yab.git
Cloning into 'pj-b-o-22-1-yab'...
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 26 (delta 11), reused 17 (delta 6), pack-reused 0
Receiving objects: 100% (26/26), 5.25 KiB | 2.62 MiB/s, done.
Resolving deltas: 100% (11/11), done.
PS D:\utorrent\vh> |
```

Рисунок 3 – Клонирование созданного репозитория на локальный компьютер

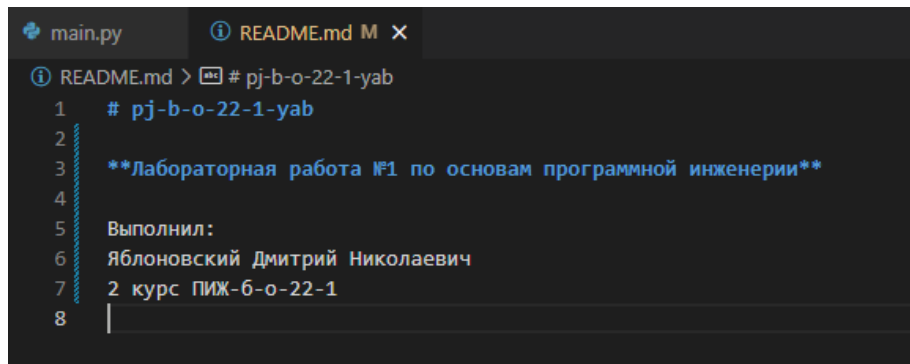
3. Дополнить файл «.gitignore» необходимыми правилами для выбранного языка:

При создании репозитория и выборе языка Python, GitHub автоматический создал «.gitignore» с нужными правилами игнорирования файлов Python и IDE.

```
main.py  .gitignore x
.gitignore
99 # This is especially recommended for binary packages to ensure reproducibility, and is more
100 # commonly ignored for libraries.
101 # https://python-poetry.org/docs/basic-usage/#commit-your-poetrylock-file-to-version-control
102 #poetry.lock
103
104 # pdm
105 # Similar to Pipfile.lock, it is generally recommended to include pdm.lock in version control.
106 #pdm.lock
107 # pdm stores project-wide configurations in .pdm.toml, but it is recommended to not include it
108 # in version control.
109 # https://pdm.fming.dev/#use-with-ide
110 .pdm.toml
111
112 # PEP 582; used by e.g. github.com/David-OConnor/pyflow and github.com/pdm-project/pdm
113 __pypackages__
114
115 # Celery stuff
116 celerybeat-schedule
117 celerybeat.pid
118
119 # SageMath parsed files
120 *.sage.py
121
122 # Environments
123 .env
124 .venv
125 env/
126 venv/
127 ENV/
128 env.bak/
129 venv.bak/
130
131 # Spyder project settings
132 .spyderproject
```

Рисунок 4 – Часть созданных, игнорируемых файлов и расширений для Python и IDE

4. Добавить в файл README.md информацию о группе и ФИО автора лабораторной работы:



```
main.py  README.md M x
README.md > # pj-b-o-22-1-yab
1 # pj-b-o-22-1-yab
2
3 **Лабораторная работа №1 по основам программной инженерии**
4
5 Выполнил:
6 Яблоновский Дмитрий Николаевич
7 2 курс ПИЖ-6-о-22-1
8 |
```

Рисунок 5 – Добавление соответствующей информации в README.md

5. Написать небольшую программу на Python. Фиксировать изменения при написании программы в локальном репозитории:

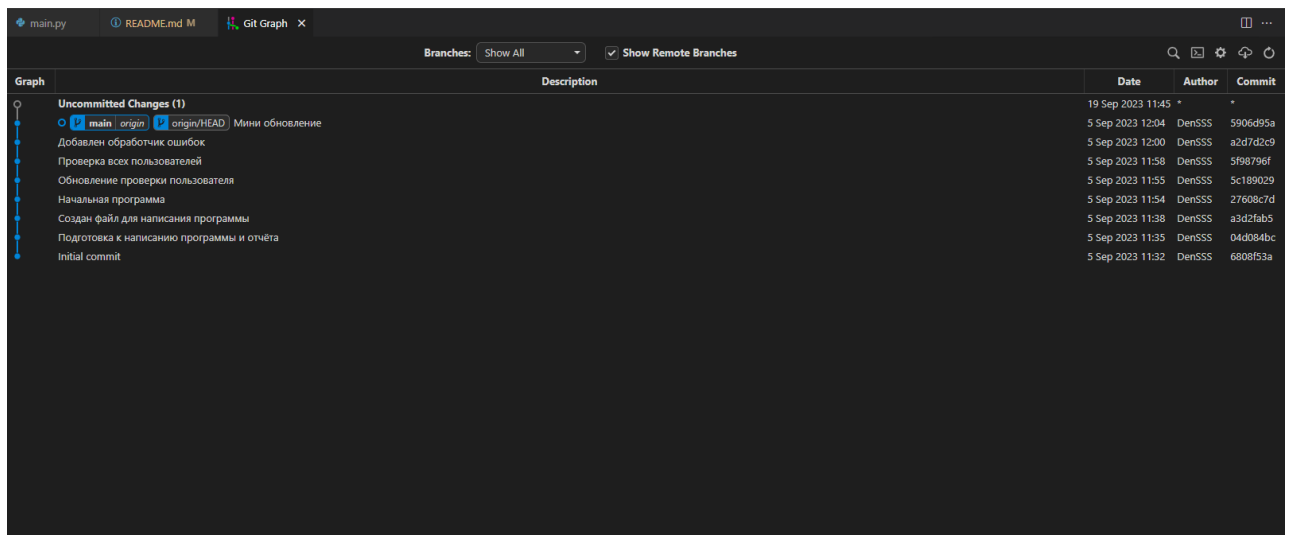


Рисунок 6 – История коммитов (1)

Листинг полученной программы:

```
import requests
import time
import json

def check_users(url):
    while True:
        response = requests.get(url)
        data = response.json()
        for i in range(0, 10):
            try:
                if data[i] is not None:
                    print(f"Пользователь {data[i]['nick']} - существует")
                else:
                    print("Пользователь не существует")
            except Exception as e:
                print("Пользователь не существует")
        time.sleep(300)

if __name__ == '__main__':
    check_users(url='https://api.heyron.ru/api/players')
```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ
PS D:\github\pj-b-o-22-1-yab> & C:/Users/maxik/AppData/Local/Programs/Python/Python39/python.exe d:/github/pj-b-o-22-1-yab/main.py
Пользователь alternativa - существует
Пользователь AVHELLHEART - существует
Пользователь kutj2122 - существует
Пользователь __Varti336__ - существует
Пользователь neon00745 - существует
Пользователь yaand - существует
Пользователь kOk0s - существует
Пользователь solfed - существует
Пользователь GermanSk - существует
Пользователь FollJar - существует
█
```

Рисунок 8 – Пример работы программы

6. Добавим описание в файл README.md:

```
README.md

# pj-b-o-22-1-yab

Лабораторная работа №1 по основам программной инженерии

Выполнил: Яблоновский Дмитрий Николаевич 2 курс ПИЖ-б-о-22-1

Программа выводит в консоль информацию о том существует ли игрок в базе данных или нет

Пример вывода:



- Пользователь alternativa - существует
- Пользователь AVHELLHEART - существует
- Пользователь kutj2122 - существует
- Пользователь Varti336 - существует
- Пользователь neon00745 - существует
- Пользователь yaand - существует
- Пользователь kOk0s - существует
- Пользователь solfed - существует
- Пользователь GermanSk - существует
- Пользователь FollJar - существует

```

Рисунок 9 – Итоговый README.md файл

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?

У локальных самый большой недостаток является отсутствие

возможности удаленной работы над проектом другими людьми. У ЦСКВ единая точка отказа, представленная центральным сервером. Если этот сервер выйдет из строя на неопределенное время, 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 является распределенной СКВ (РСКВ).

4. В чем концептуальное отличие Git от других СКВ?

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.

5. Как обеспечивается целостность хранимых данных в Git?

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. Механизм, которым пользуется Git при вычислении хеш-сумм, называется SHA-1 хеш.

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 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. GitHub – это платформа для размещения кода. Иными словами, это место, где разработчики могут хранить свои проекты и работать вместе.

8. Какие бывают репозитории в GitHub?

Репозитории могут быть открытыми (public) или закрытыми (private).

9. Укажите основные этапы модели работы с GitHub.

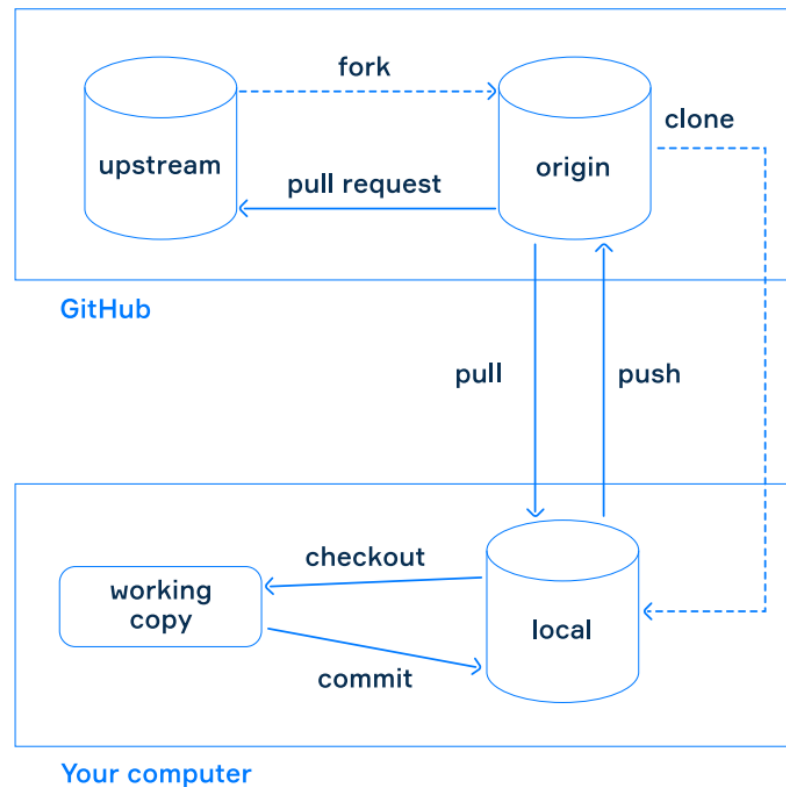


Рисунок 10 – Схема модели работы с GitHub

10. Как осуществляется первоначальная настройка Git после установки?

Чтобы убедиться, что git установлен используется команда «git version». После используются команды для привязки git к аккаунту GitHub: «git config --global user.name <YOUR_NAME>» и «git config --global user.email <EMAIL>».

11. Опишите этапы создания репозитория в GitHub.

Сначала обязательно вводится имя репозитория. Добавляется описания, но его можно оставить пустым. Выбирается вид доступа к репозиторию (public/private), а также файлы .gitignore и LICENSE.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Поддерживаемые лицензии: Apache License 2.0, GNU General Public License v3.0, MIT License, BSD 2-Clause “Simplified” License, BSD 3-Clause “New” or “Revised” License, Boost Software License 1.0, Creative Commons Zero v1.0 Universal, Eclipse Public License 2.0, GNU Affero General Public License v3.0, GNU General Public License v2.0, GNU Lesser General Public License v2.1, Mozilla Public License 2.0, The Unlicense.

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

Клонирование репозитория на локальный компьютер осуществляется командой: «git clone <repository url>». Этот процесс передает данные с удаленного репозитория GitHub на локальный компьютер.

14. Как проверить состояние локального репозитория Git?

Можно воспользоваться командой: «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?

Если использовать проверку статуса репозитория после добавления/изменения файлов в локальном репозитории, то файлы помечаются статусом modified. При использовании команды «git add» файлы, которые находились под статусом modified, получают статус staged и при дальнейшем использовании команды «git commit» фиксируются, создавая новый снимок. Команда «git push» передает изменения ветки на удаленный репозиторий GitHub.

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`.

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

Наиболее популярные сервисы, работающие с Git: BitBucket и GitLab.

Небольшое сравнение GitHub и Bitbucket:

GitHub:

- Владелец является компания Microsoft.
- Поддерживает только РСКВ Git.
- Огромное число сторонников и проектов.
- Обширная система интеграций со сторонними продуктами.

Bitbucket:

- Владелец является компания Atlassian.
- Поддерживает РСКВ Git и Mercurial.
- Интеграция только с другими продуктами Atlassian.

Решение выбора конкретной платформы зависит от проекта и предпочтений команды или автора.

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.

Примером программ с графическим интерфейсом для работы с Git могут являться: GitHub Desktop, Source Tree, GitKraken и т.д.