

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»**

Выполнила:  
Ламская Ксения Вячеславовна  
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:  
Воронкин Р. А., доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

## Ход работы

### 1. Изучили теоретический материал работы (рис.1.1).

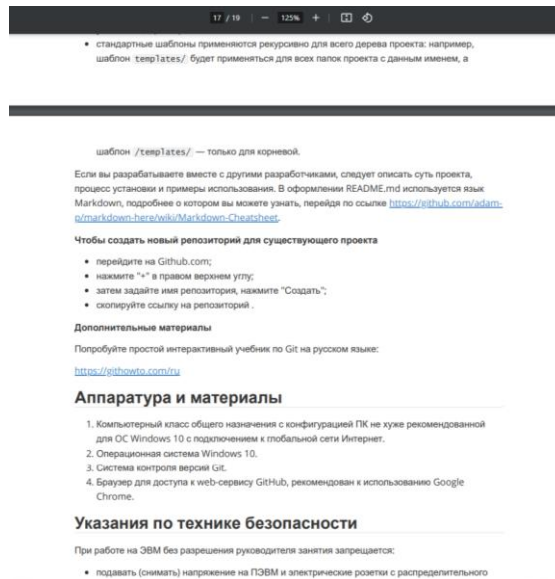


Рисунок 1.1 – Изучение материала для лабораторной работы

### 2. Создали общедоступный репозиторий на GitHub, в котором будет использован выбранный язык программирования (рис.2.1-рис.2.2).

The image shows the 'Create a new repository' form on GitHub. It includes fields for 'Owner' (ksenia-lamskaya) and 'Repository name' (chstting), with a confirmation that 'chstting is available'. There's a section for 'Description (optional)' and a choice between 'Public' (selected) and 'Private' visibility. Below that, it asks to 'Initialize this repository with:' and offers to 'Add a README file'. There's also a section for 'Add .gitignore' with a 'Python' template selected. A 'Choose a license' section has 'MIT License' selected. At the bottom, a note states 'You are creating a public repository in your personal account.' and a green 'Create repository' button is visible.

Рисунок 2.1 – Настройка репозитория

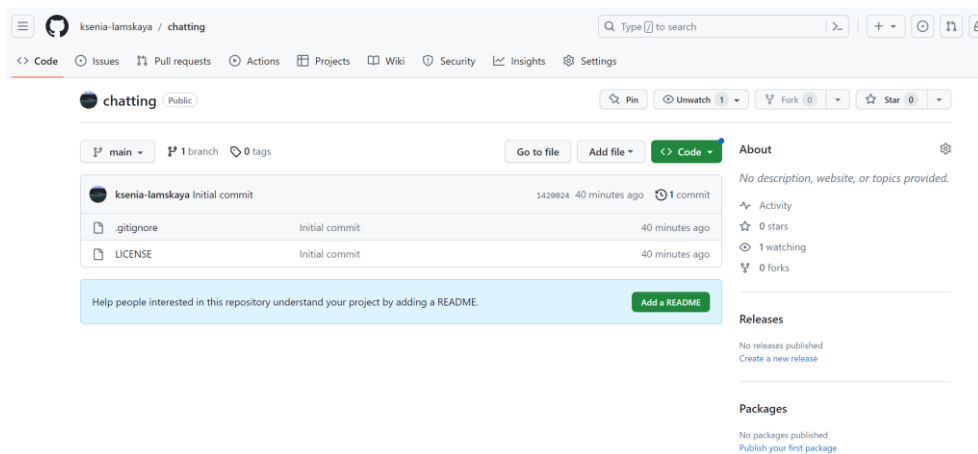


Рисунок 2.2 – Готовый репозиторий

3. Выполнили клонирование созданного репозитория на рабочий компьютер (рис.3.1-рис.3.2)

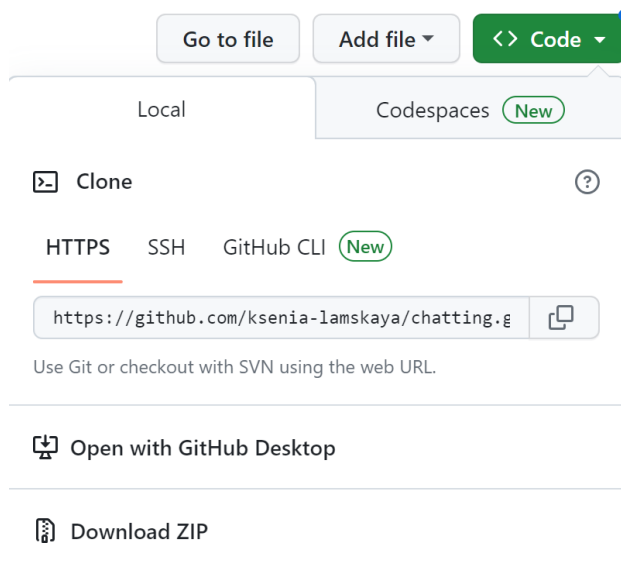


Рисунок 3.1 – Копирование ссылки репозитория

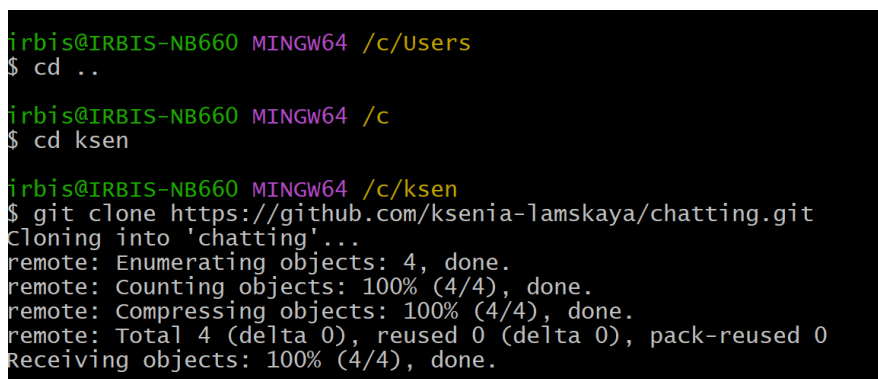



Рисунок 3.2 – Копирование репозитория на рабочий компьютер

4. Дополнили файл .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки (рис.4.1).

```
1  # Byte-compiled / optimized / DLL files
2  __pycache__
3  *.py[co]
4  *.pyc
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28
29 # PyInstaller
30 # Usually these files are written by a python script from a template
31 # before PyInstaller builds the exe, so as to inject date/other infos into it.
32 *.manifest
33 *.spec
```

Рисунок 4.1 – Файл с необходимыми правилами для языка Python

5. Добавили в файл README.md информацию о группе и моём ФИО (рис.5.1-рис.5.3).

 README – Блокнот

Файл Правка Формат Вид Справка

# chatting

Ламская Ксения Вячеславовна ПИЖ-6-о-22-1

Рисунок 5.1 – Добавление информации

```
irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$ git add .

irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   README.md

irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$ git commit -m "change"
[main 6030841] change
1 file changed, 1 insertion(+), 2 deletions(-)

irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$ git push
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 324 bytes | 64.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ksenia-lamskaya/chatting.git
   4e8fe19..6030841  main -> main

irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$
```

Рисунок 5.2 – Добавление в файл READ.md информации о группе и моём ФИО на GitHub через Git Bush

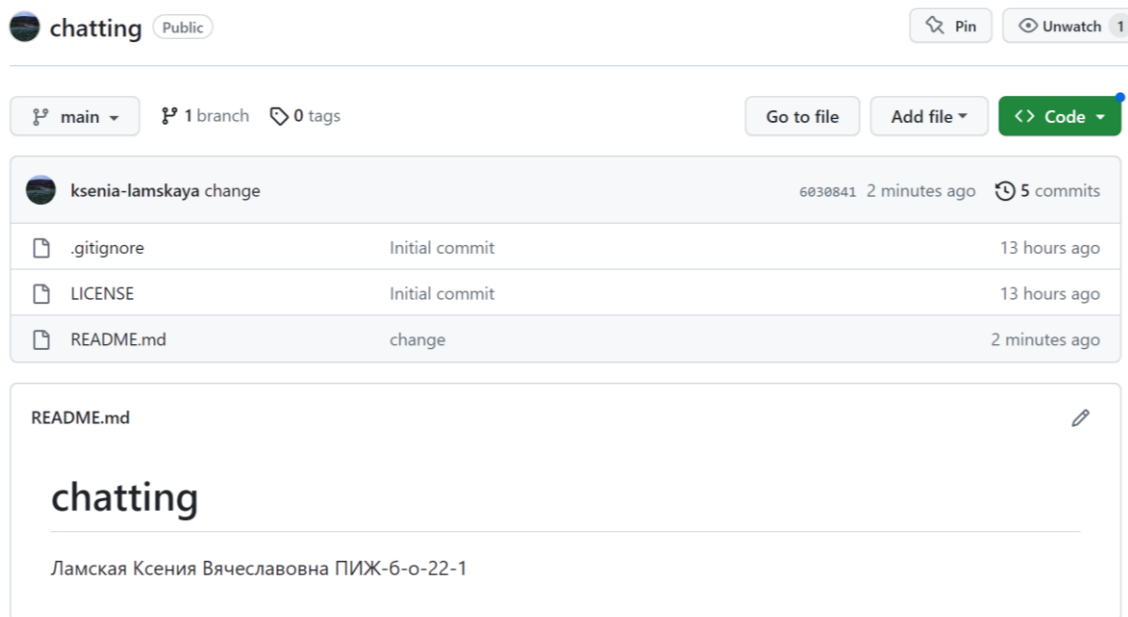


Рисунок 5.2 – Мой файл README.md на GitHub

6. Написали небольшую программу на выбранном языке программирования. Зафиксировали изменения при написании программы в локальном репозитории. Сделали не менее 7 коммитов (рис.6.1-рис.6.5).

```

chat.py 1 • Extension: Python
C: > Ksen > chatting > chat.py > divide_fractions
1  from fractions import Fraction
2
3  def add_fractions(num1, den1, num2, den2):
4      result = Fraction(num1, den1) + Fraction(num2, den2)
5      return result.numerator, result.denominator
6
7  def subtract_fractions(num1, den1, num2, den2):
8      result = Fraction(num1, den1) - Fraction(num2, den2)
9      return result.numerator, result.denominator
10
11 def multiply_fractions(num1, den1, num2, den2):
12     result = Fraction(num1, den1) * Fraction(num2, den2)
13     return result.numerator, result.denominator
14
15 def divide_fractions(num1, den1, num2, den2):
16     result = Fraction(num1, den1) / Fraction(num2, den2)
17     return result.numerator, result.denominator
18
19 def main():
20     num1 = int(input("Введите числитель первой дроби: "))
21     den1 = int(input("Введите знаменатель первой дроби: "))
22     num2 = int(input("Введите числитель второй дроби: "))
23     den2 = int(input("Введите знаменатель второй дроби: "))
24
25     print("Сложение дробей: ", add_fractions(num1, den1, num2, den2))
26     print("Вычитание дробей: ", subtract_fractions(num1, den1, num2, den2))
27     print("Умножение дробей: ", multiply_fractions(num1, den1, num2, den2))
28     print("Деление дробей: ", divide_fractions(num1, den1, num2, den2))
29
30 if name == "main":
31     main()
32

```

Рисунок 6.1 – Моя программа на Python

```
chat.py 9+ X Extension: Python
C: > Ksen > chatting > chat.py > ...
1  from fractions import Fraction
2
3  def add_fractions(num1, den1, num2, den2):
4      result = Fraction(num1, den1) + Fraction(num2, den2)
5      return result.numerator, result.denominator
6
7  def subtract_fractions(num1, den1, num2, den2):
8      result = Fraction(num1, den1) - Fraction(num2, den2)
9      return result.numerator, result.denominator
10
11 def multiply_fractions(num1, den1, num2, den2):
12     result = Fraction(num1, den1) * Fraction(num2, den2)
13     return result.numerator, result.denominator
14
15 def divide_fractions(num1, den1, num2, den2):
16     result = Fraction(num1, den1) / Fraction(num2, den2)
17     return result.numerator, result.denominator
18
```

Рисунок 6.2 – Изменение программы

```
MINGW64:/c/ksen/chatting
To https://github.com/ksenia-lamskaya/chatting.git
385545b..6624a82  main -> main

irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$ git add .

irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   chat.py

irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$ git commit -m "six"
[main 6dbbef8] six
1 file changed, 5 insertions(+)

irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 421 bytes | 210.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/ksenia-lamskaya/chatting.git
6624a82..6dbbef8  main -> main

irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$
```

Рисунок 6.3 – Создание 7 коммитов

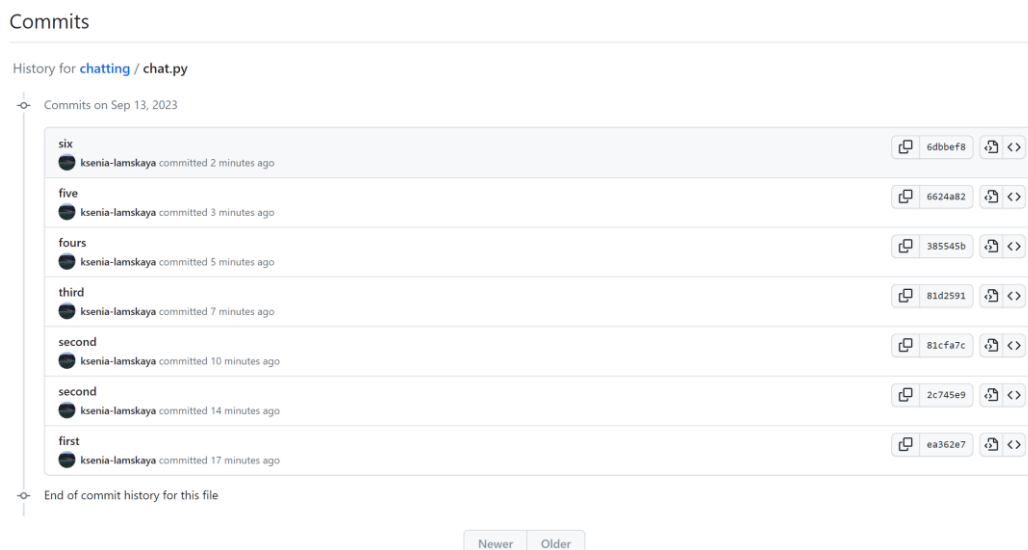


Рисунок 6.4 – Изменения, отображённые на GitHub

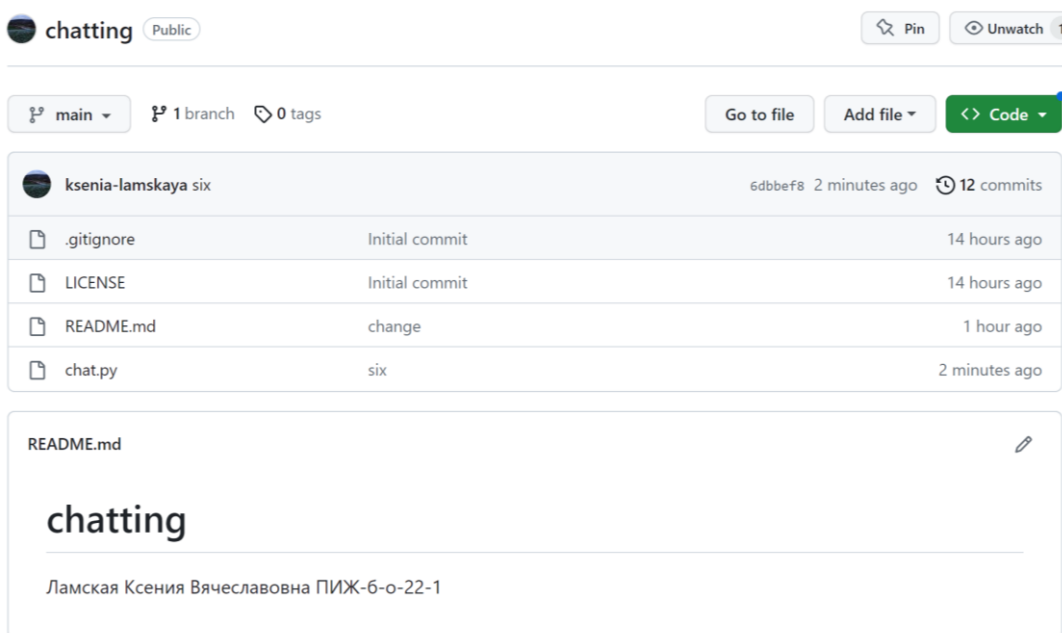


Рисунок 6.5 – Мой репозиторий

7. Добавили файл README и зафиксировали сделанные изменения (рис.7.1- рис.7.3).

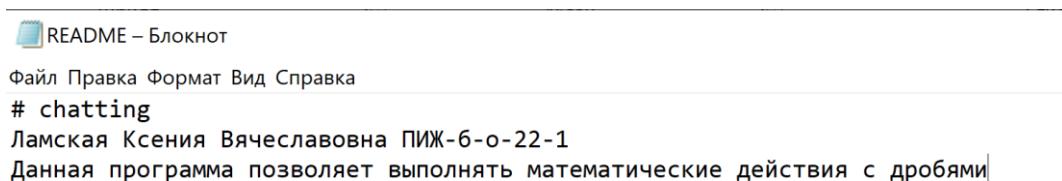


Рисунок 7.1 – Добавление файла README с описанием программы

```

irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$ git add .

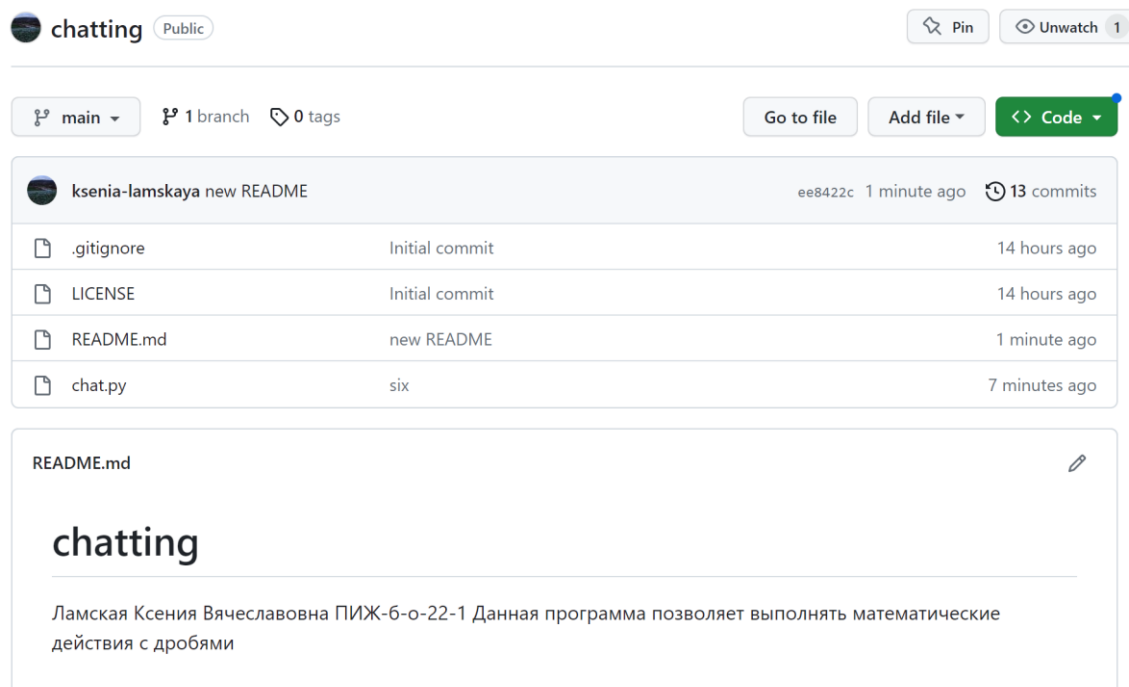
irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

git
irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$ git commit -m "new README"
[main ee8422c] new README
1 file changed, 1 insertion(+)
git
irbis@IRBIS-NB660 MINGW64 /c/ksen/chatting (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 413 bytes | 21.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ksenia-lamskaya/chatting.git
6dbbef8..ee8422c  main -> main

```

Рисунок 7.2 – Фиксация сделанных изменений



The screenshot shows a GitHub repository named 'chatting' by user 'ksenia-lamskaya'. The repository is public and has 1 branch (main) and 0 tags. The commit history shows a recent commit 'ee8422c' made 1 minute ago, which includes a new README file. The README content is as follows:

File	Commit	Time
.gitignore	Initial commit	14 hours ago
LICENSE	Initial commit	14 hours ago
README.md	new README	1 minute ago
chat.py	six	7 minutes ago

The README file content is:

```

chatting

Ламская Ксения Вячеславовна ПИЖ-6-о-22-1 Данная программа позволяет выполнять математические действия с дробями

```

Рисунок 7.3 – Мой репозиторий с измененным файлом README



## 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?*

В локальных СКВ можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели. В централизованных СКВ самый очевидный минус — это 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. Если жёсткий диск, на котором хранится центральная БД, повреждён, а своевременные бэкапы отсутствуют, вы потеряете всё.

### 3. *К какой СКВ относится Git?*

Распределённая система контроля версий

### 4. *В чем концептуальное отличие Git от других СКВ?*

Основное отличие Git от любой другой СКВ — это подход к работе со своими данными.

### 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.

### 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?***

На странице профиля отображаются сведения о вашей работе через репозитории, которые вас интересуют, вклад, который вы сделали, и беседы, в которых вы участвовали.

#### **8. *Какие бывают репозитории в GitHub?***

Публичные и приватные

#### **9. *Укажите основные этапы модели работы с GitHub.***

Создание аккаунта, создание репозитория, клонирование репозитория на локальный диск, отправка изменений на GitHub с помощью git push.

#### **10. *Как осуществляется первоначальная настройка Git после установки?***

```
git config --global <user.name>
```

```
git config --global <user.email>
```

#### **11. *Опишите этапы создания репозитория в GitHub.***

На 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, описание Public/private, gitignore и LICENSE. После заполнения этих полей нажимаем кнопку Create repository.

#### **12. *Какие типы лицензий поддерживаются GitHub при создании репозитория?***

Список лицензий можно просмотреть при создании нового репозитория

### ***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. Затем открыть командную строку или терминал и перейти в каталог, куда вы хотите скопировать хранилище. Затем написать `git clone` и ввести адрес. Это нужно для того, чтобы сохранить наши данные на локальный диск.

### ***14. Как проверить состояние локального репозитория Git?***

В командной строке с помощью команды `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`?***

1. В репозитории появляется изменённый, но не подготовленный файл.
2. Появляются изменённые, но подготовленные файлы.
3. После того как выполнен коммит происходит фиксация изменений всех подготовленных файлов, в результате репозиторий не содержит подготовленных файлов. После этого с помощью команды `git push` фиксированные файлы отправляются на 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`.***

На обоих компьютерах выполняется команда `git clone`. Оба пользователя вносят в программу необходимые изменения, фиксируют их с помощью команды коммит. Перед командой `push` для синхронизации обоих компьютеров с GitHub выполняют команду `git pull` и только потом `git push`.

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

Преимущества и недостатки GitLab

Преимущества:

- Бесплатный план без ограничений, но есть планы оплаты.
- Это лицензия с открытым исходным кодом.
- Разрешает самостоятельный хостинг на любом плане.
- Он очень хорошо интегрирован с Git.

Недостатки:

- Его интерфейс может быть несколько медленнее по сравнению с конкурентами.

Преимущества и недостатки GitHub

Преимущества:

- Бесплатное обслуживание, хотя есть и платные.
- Очень быстрый поиск в структуре репозитория.
- Большое сообщество и легко найти помощь.
- Он предлагает практические инструменты для сотрудничества и

хорошую интеграцию с Git.

- Легко интегрируется с другими сторонними сервисами.
- Он также работает с TFS, HG и SVN.

Недостатки:

- У него есть ограничения по пространству, так как вы не можете превышать 100 МБ в одном файле, в то время как репозитории ограничены 1 ГБ в бесплатной версии.

*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.*

На графическом интерфейсе программы GitHub Desktop выбираем необходимый репозиторий. Указываем название коммита и его описание, затем нажимаем кнопку коммит. Происходит фиксация изменений на локальной машине. После этого нажимаем кнопку push и происходит передача данных на GitHub.