

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины
«Основы кроссплатформенного программирования»

Выполнил:
Выполнил:
Дедио Всеволод Юрьевич
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)

(подпись)

Проверил:
Доцент департамента цифровых,
робототехнических систем и
электроники Воронкин Р.А.

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: исследование основных возможностей Git и GitHub

Цель: исследовать базовые возможности системы контроля версий Git или версий для хостинга IT-проекта GitHub

Порядок выполнения работы:

1. Ознакомился с теоретическим материалом
2. Создал аккаунт в GitHub
3. Установил Git
4. Создал репозиторий

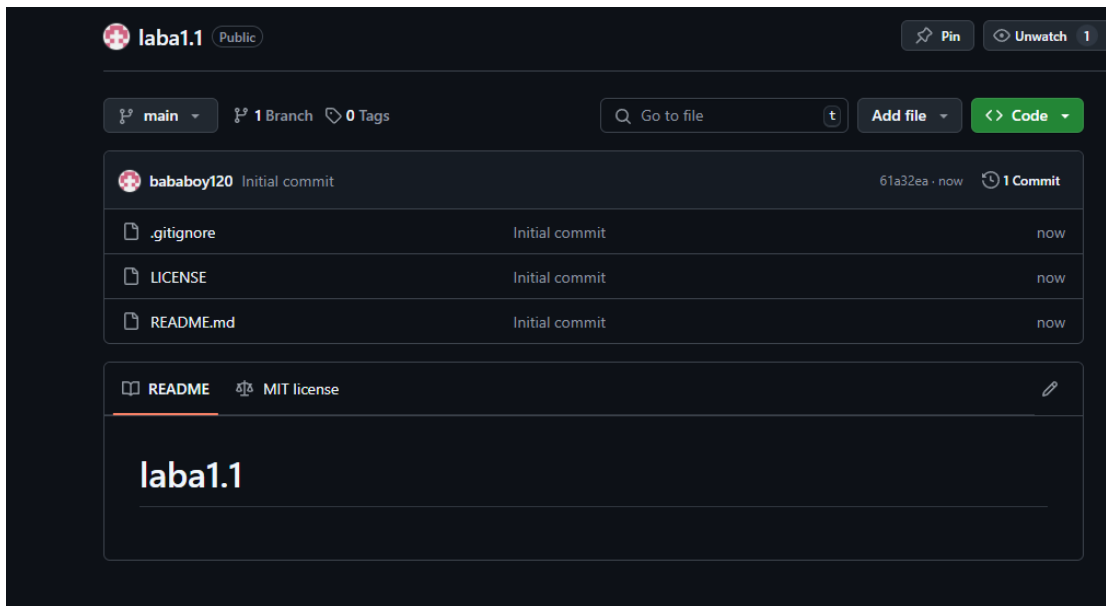


Рисунок 1. Репозиторий

5. Настройка конфига

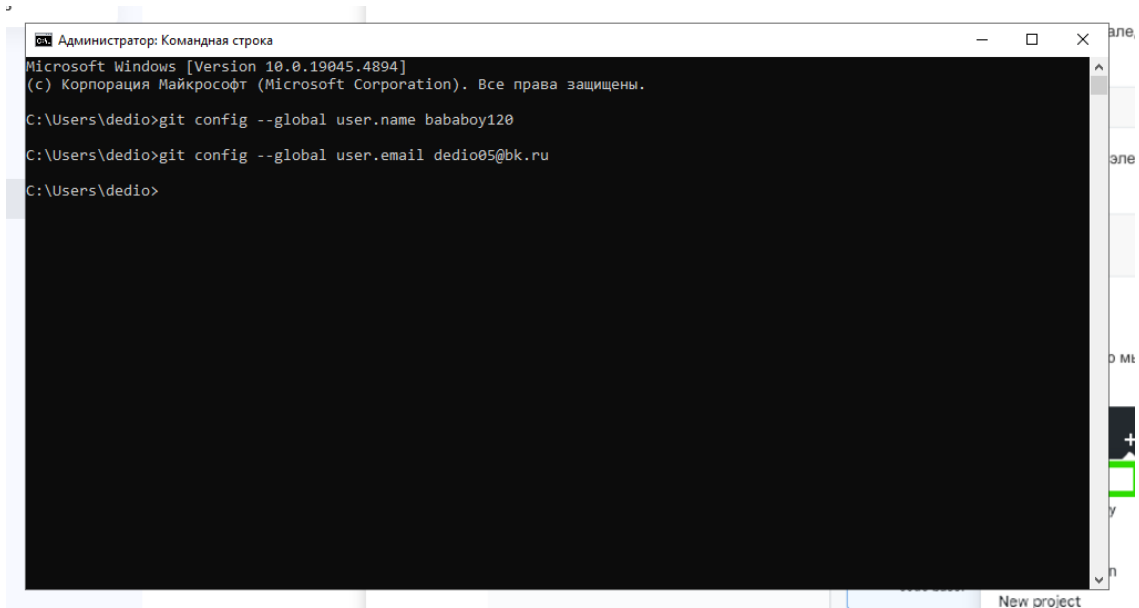
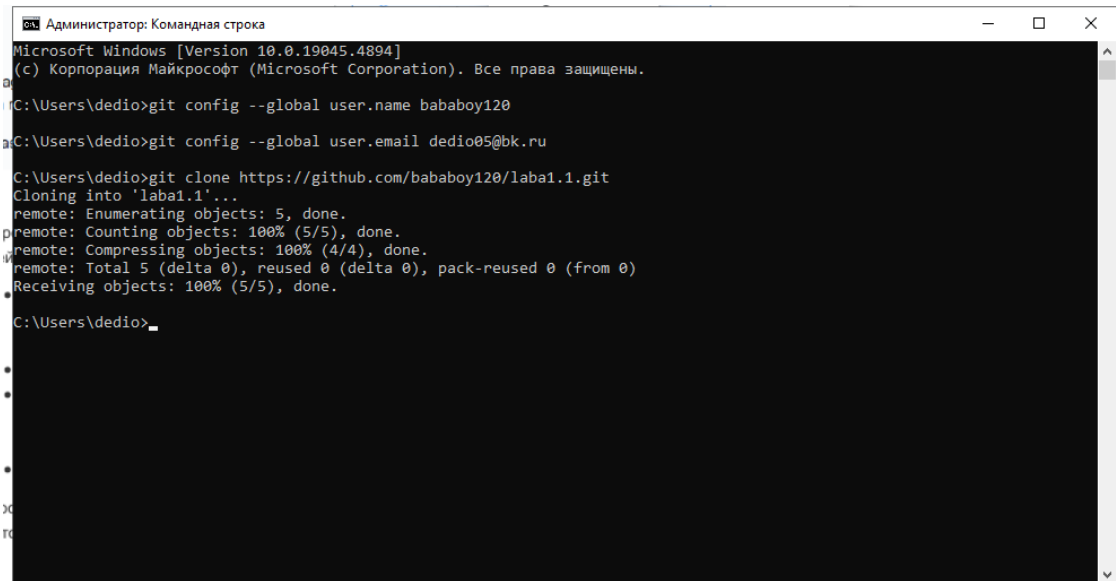


Рисунок 2. Настройки

6. Клонирование репозитория



```
Администратор: Командная строка
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\dedio>git config --global user.name bababoy120

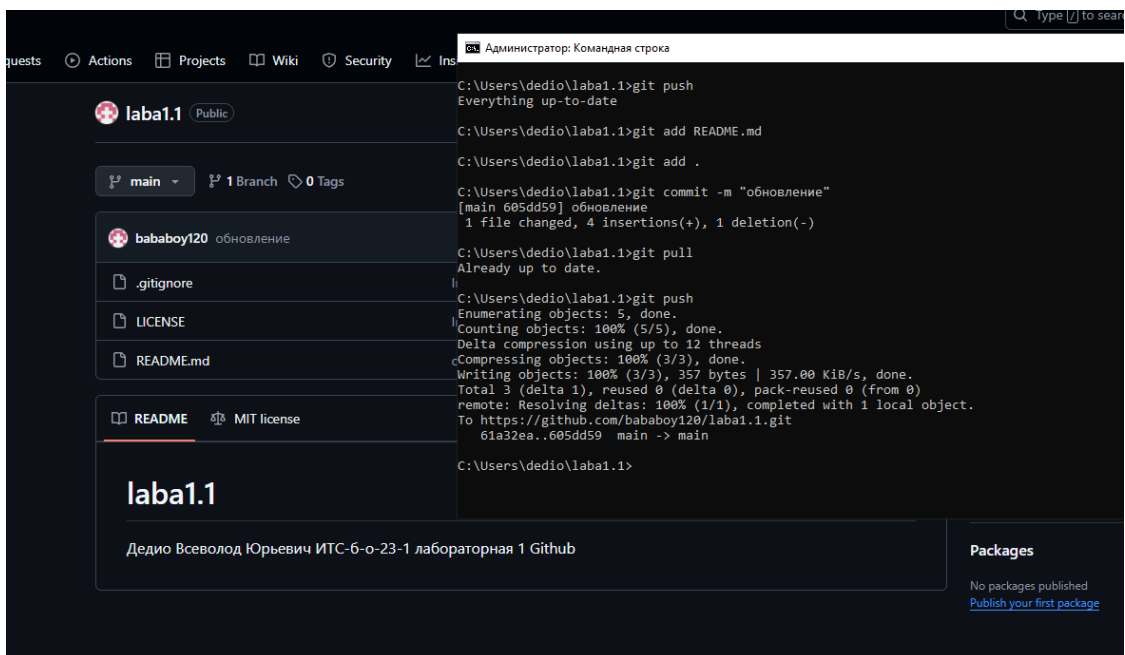
C:\Users\dedio>git config --global user.email dedio05@bk.ru

C:\Users\dedio>git clone https://github.com/bababoy120/laba1.1.git
Cloning into 'laba1.1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), done.

C:\Users\dedio>
```

Рисунок 3. Клонирование

7. Добавил файл readme и нужные данные в него



laba1.1 Public

main 1 Branch 0 Tags

bababoy120 обновление

.gitignore

LICENSE

README.md

README MIT license

laba1.1

Дедю Всеволод Юрьевич ИТС-6-о-23-1 лабораторная 1 Github

```
Администратор: Командная строка
C:\Users\dedio\laba1.1>git push
Everything up-to-date

C:\Users\dedio\laba1.1>git add README.md

C:\Users\dedio\laba1.1>git add .

C:\Users\dedio\laba1.1>git commit -m "обновление"
[main 605dd59] обновление
1 file changed, 4 insertions(+), 1 deletion(-)

C:\Users\dedio\laba1.1>git pull
Already up to date.

C:\Users\dedio\laba1.1>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 357 bytes | 357.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/bababoy120/laba1.1.git
61a32ea..605dd59 main -> main

C:\Users\dedio\laba1.1>
```

Packages

No packages published

[Publish your first package](#)

Рисунок 4. README

8. Написал не большую программу

```

C: > Users > dedio > Desktop > питон.ру > ...
1  print('0 в качестве знака операции'
2  '\nзавершит работу программы\n')
3
4  while True:
5      s = input('Знак (+, -, *, /): ')
6      if s == '0':
7          break
8      if s in ('+', '-', '*', '/'):
9          a = float(input('a = '))
10         b = float(input('b = '))
11         if s == '+':
12             print('%.2f' % (a + b))
13         elif s == '-':
14             print('%.2f' % (a - b))
15         elif s == '*':
16             print('%.2f' % (a * b))
17         elif s == '/':
18             if b != 0:
19                 print('%.2f' % (a / b))
20             else:
21                 print('Деление на ноль!')
22     else:
23         print('Неверный знак операции!')

```

Рисунок 5. Программа

9. Добавил в репозиторий

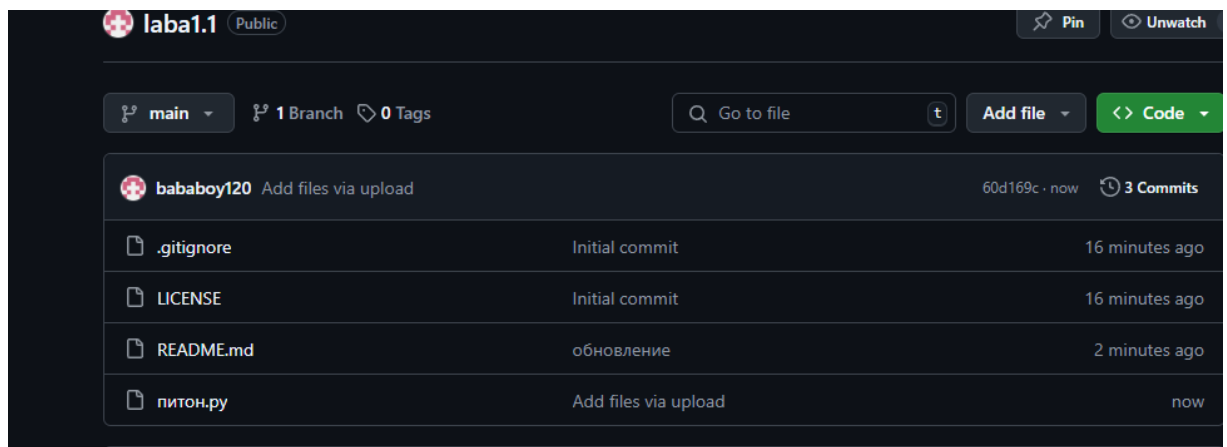


Рисунок 6. Добавление программы

10. Сделал не менее 7 коммитов

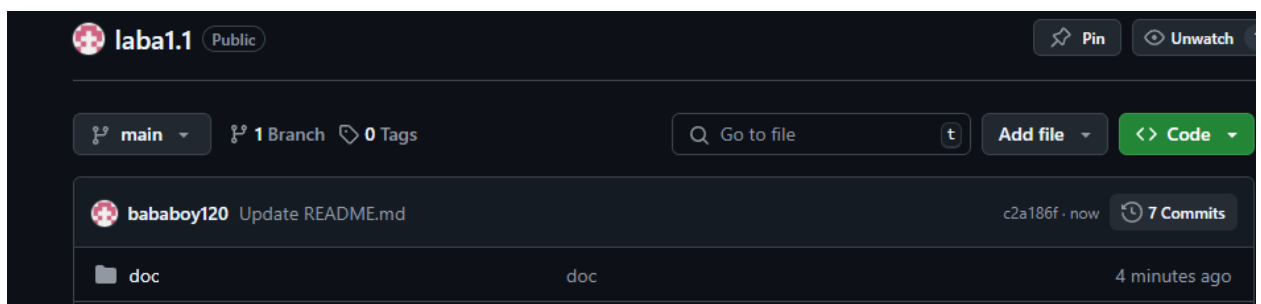


Рисунок 7. коммиты

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?

– Локальных:

Можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели.

когда вся история проекта хранится в одном месте, вы рискуете потерять всё.

– Централизованных:

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 — всю историю проекта, не считая единичных снимков репозитория, которые сохранились на локальных машинах разработчиков.

3. К какой СКВ относится Git?

– Распределенные СКВ

4. В чем концептуальное отличие Git от других СКВ?

– Git — это децентрализованная система контроля версий, которая хранит историю проекта как моментальные снимки, а не просто различия между файлами. Это делает его быстрым, гибким и надежным. В Git легко ветвиться, сливать изменения и работать автономно. В отличие от централизованных систем (SVN, CVS), в Git каждый разработчик имеет

локальную копию всей истории проекта, что обеспечивает большую независимость и свободу действий.

5. Как обеспечивается целостность хранимых данных в Git?

– Git использует SHA-1 хеширование для создания уникальных идентификаторов для каждого объекта в репозитории. Эти объекты неизменяемы, поэтому Git может гарантировать, что прошлые версии файлов не будут изменены. При чтении файлов, Git проверяет их хеши, чтобы убедиться, что данные не были повреждены. Кроме того, Git создает специальный файл. `git/index`, который отслеживает все объекты в репозитории, и также проверяет его целостность.

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

- 1) Untracked: Git не знает о файле.
- 2) Tracked (Unmodified): Файл отслеживается, но не изменён.
- 3) Tracked (Modified): Файл изменён, но изменения ещё не добавлены в staging area.
- 4) Staged: Файл подготовлен к коммиту.
- 5) Committed: Изменения в файле зафиксированы в коммите.

Файлы перемещаются между этими состояниями с помощью команд `git add` и `git commit`.

– Состояния файлов в Git связаны как этапы "путешествия" файла от неизвестного Git до зафиксированного в истории

7. Что такое профиль пользователя в GitHub?

– В целом, профиль GitHub — это ваш цифровой профиль как разработчика, который помогает вам взаимодействовать с сообществом, демонстрировать свои навыки, искать работу и повышать свой авторитет в области разработки программного обеспечения.

8. Какие бывают репозитории в GitHub?

– Приватные и публичные

9. Укажите основные этапы модели работы с GitHub.

- 1) Создание аккаунта;
- 2) Создание репозитория;
- 3) Клонирование репозитория;
- 4) Внесение изменений;
- 5) Добавление изменений в индекс
- 6) Создание коммита;
- 7) Работа с Pull Requests;
- 8) Обновление локальной копии;
- 9) Взаимодействие с сообществом.

10. Как осуществляется первоначальная настройка Git после установки?

- Добавляем свои данные и почту связанную с GitHub.

11. Опишите этапы создания репозитория в GitHub.

- 1) Авторизация в GitHub;
- 2) Переход на страницу создания репозитория;
- 3) Заполнение формы создания репозитория;
- 4) Создание репозитория;
- 5) Настройка репозитория;

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

- Open Source Licenses (Лицензии с открытым исходным кодом);
- Proprietary Licenses (Проприетарные лицензии)
- Other Licenses (Другие лицензии)

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

- Найти репозиторий, найти кнопку «Code», затем скопировать ссылку на репозиторий, открыть Git CMD и использовать команду `git clone «ссылка»`
- Клонировать репозиторий для того, чтобы работать с кодом локально, вносить изменения. Так же для совместной работы, клонирование позволяет

получить доступ к коду, внести свои изменения и отправить их на сервер GitHub для дальнейшей совместной работы.

14. Как проверить состояние локального репозитория Git?

– Проверить состояние можно 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` ?

– `git add`: подготавливает изменения к коммиту.

– `git commit`: фиксирует изменения в локальной истории.

– `git push`: отправляет изменения на сервер GitHub.

– После `git push` состояние файла в локальном репозитории не меняется, но он синхронизируется с сервером.

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` .

– Клонирование репозитория. На обоих компьютерах выполнить команду `git clone [ссылка на репозитори]`. Это создаст локальные копии репозитория.

– Работа над проектом. На каждом компьютере нужно внести изменения в код и файлы проекта.

– Добавление изменений в индекс. На обоих компьютерах нужно использовать `git add` . для добавления всех измененных файлов в индекс.

– Фиксация изменений. На обоих компьютерах выполнить `git commit -m "Описание изменений"` для фиксации изменений.

– Отправка изменений на GitHub. На первом компьютере выполните `git push` (название ветки, над которой работаете) для отправки изменений на GitHub. На втором компьютере нужно сделать точно так же.

– Проверка синхронизации. Нужно убедиться, что `git status` на обоих компьютерах не показывает никаких изменений или конфликтов.

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

– GitLab и Bitbucket

– GitHub имеет ряд преимуществ, таких как: самый популярный сервис, большое сообщество разработчиков, простая в использовании платформа, хорошо развитые функции для совместной работы, богатая экосистема интеграций.

Так же имеются недостатки: бесплатная версия ограничена в функциях, некоторые функции не так развиты, как в GitLab

– GitLab имеет такие преимущества как: децентрализованная платформа, большой набор функций, бесплатный план предлагает неограниченное кол-во приватных репозиториев.

Недостатки: интерфейс менее интуитивный, чем у GitHub, сообщество меньше, чем у GitHub

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.

– Программные средства с графическим интерфейсом пользователя для работы с Git: GitHub Desktop, GitKraken, Sourcetree, TortoiseGit, Fork

Вывод: в ходе работы исследовал базовые возможности системы контроля версий Git или версий для хостинга IT-проекта GitHub.