

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:  
Мелтонян Одиссей Гарикович  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
очная форма обучения

---

(подпись)

Проверил: Воронкин Р.А.

---

(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 работы:

1. Изучил теоретический материал.
2. Создал общедоступный репозиторий на GitHub, в котором была использована лицензия MIT и выбранный Вами язык программирования.

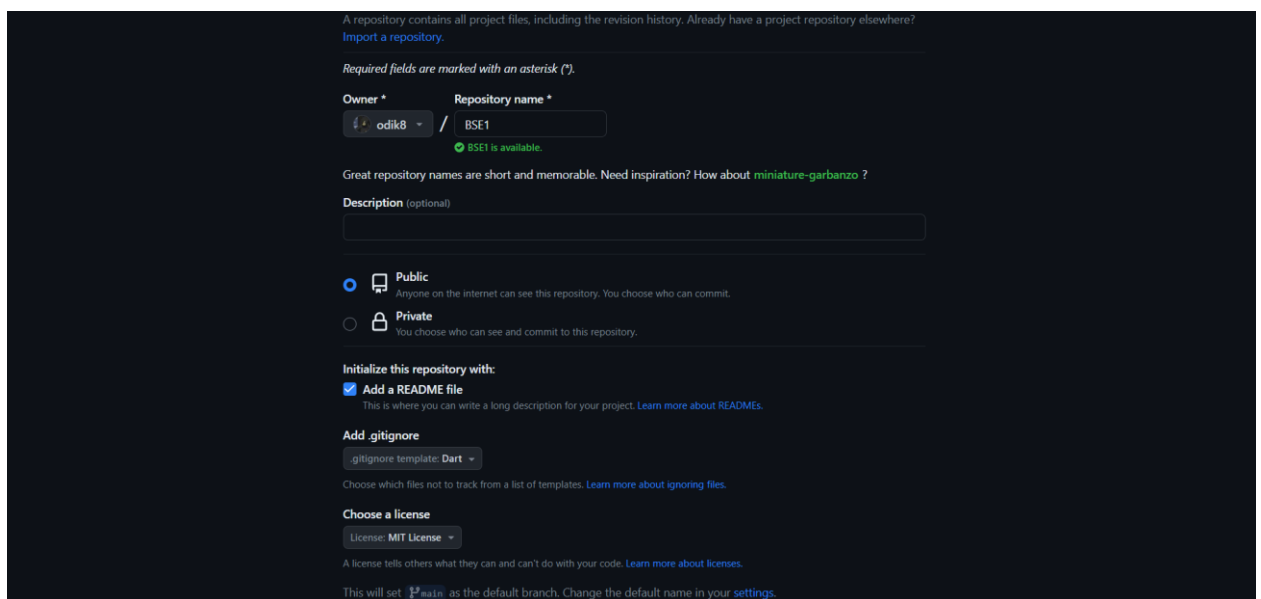


Рисунок 1. – Страница создания репозитория

3. Выполнил клонирование созданного репозитория на рабочий компьютер

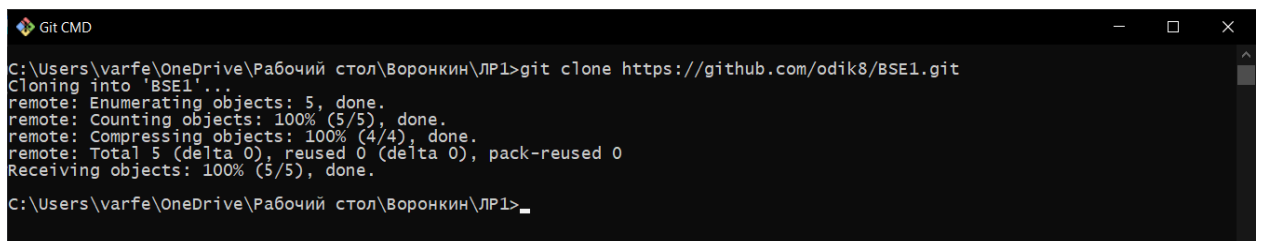
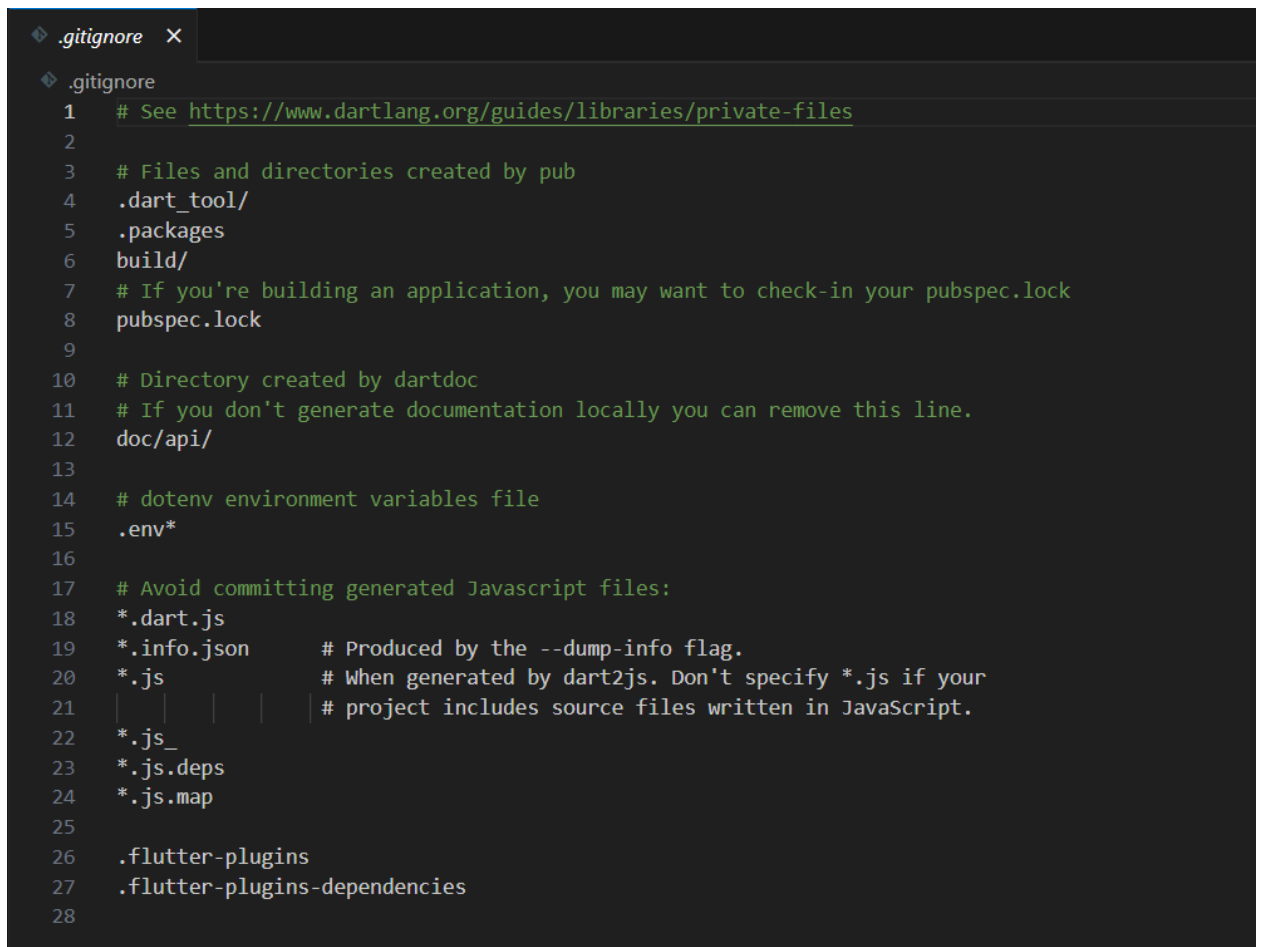


Рисунок 2. – Окно клонирования репозитория

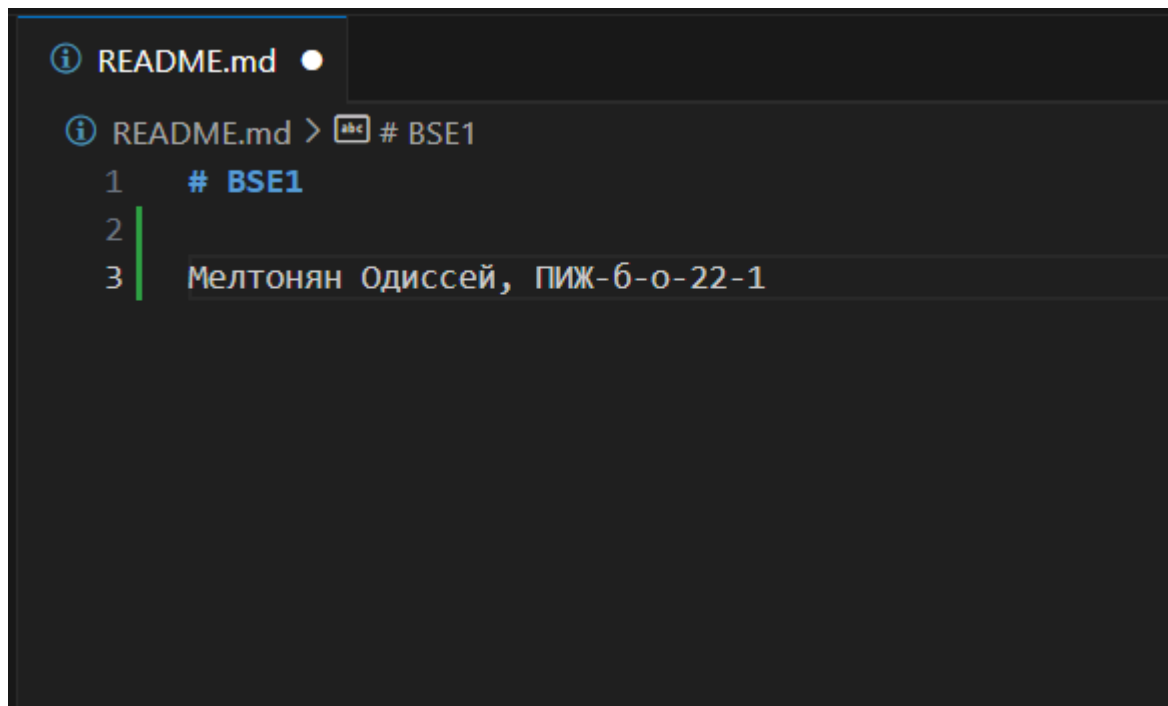
4. Дополнил файл .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.



```
.gitignore
1 # See https://www.dartlang.org/guides/libraries/private-files
2
3 # Files and directories created by pub
4 .dart_tool/
5 .packages
6 build/
7 # If you're building an application, you may want to check-in your pubspec.lock
8 pubspec.lock
9
10 # Directory created by dartdoc
11 # If you don't generate documentation locally you can remove this line.
12 doc/api/
13
14 # dotenv environment variables file
15 .env*
16
17 # Avoid committing generated Javascript files:
18 *.dart.js
19 *.info.json      # Produced by the --dump-info flag.
20 *.js             # When generated by dart2js. Don't specify *.js if your
21 |               | # project includes source files written in JavaScript.
22 *.js_
23 *.js.deps
24 *.js.map
25
26 .flutter-plugins
27 .flutter-plugins-dependencies
28
```

Рисунок 3. – Файл .gitignore

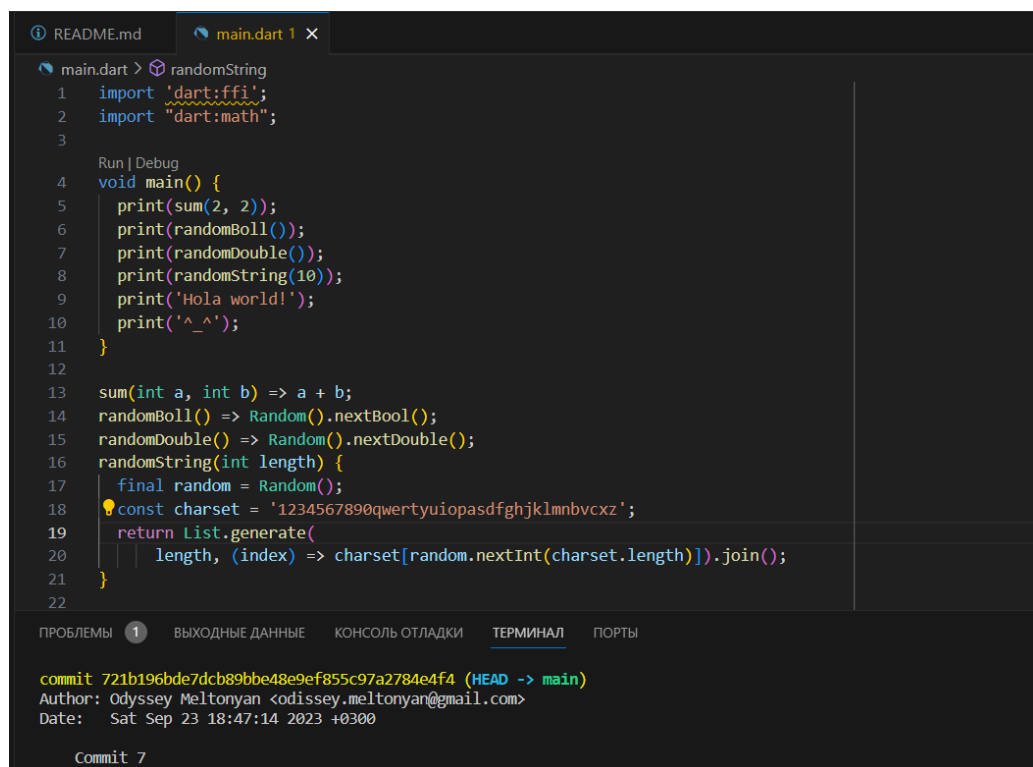
5. Добавил в файл README.md информацию о группе и ФИО студента, выполняющего лабораторную работу.



```
1 # BSE1
2
3 Мелтонян Одиссей, ПИЖ-б-о-22-1
```

Рисунок 4. – Файл README.md

6. Написал небольшую программу на выбранном языке программирования. Зафиксировал изменения при написании программы в локальном репозитории. Было сделано 7 коммитов.



```
1 import 'dart:ffi';
2 import "dart:math";
3
4 void main() {
5   print(sum(2, 2));
6   print(randomBoll());
7   print(randomDouble());
8   print(randomString(10));
9   print('Hola world!');
10  print('^_^');
11 }
12
13 sum(int a, int b) => a + b;
14 randomBoll() => Random().nextBool();
15 randomDouble() => Random().nextDouble();
16 randomString(int length) {
17   final random = Random();
18   const charset = '1234567890qwertyuiopasdfghjklmbvcxz';
19   return List.generate(
20     length, (index) => charset[random.nextInt(charset.length)]).join();
21 }
22
```

commit 721b196bde7dcb89bbe48e9ef855c97a2784e4f4 (HEAD -> main)  
Author: Odyssey Meltonyan <odissey.meltonyan@gmail.com>  
Date: Sat Sep 23 18:47:14 2023 +0300

Commit 7

Рисунок 5. Файл main.dart

7. Зафиксировал сделанные изменения в файле README.
8. Добавил отчет по лабораторной работе в формате PDF в папку doc репозитория.
9. Отправил изменения в локальном репозитории в удаленный репозиторий GitHub

**Вывод:** были исследованы базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

### **Вопросы для защиты работы:**

1. Что такое СКВ и какого его назначение? – СКВ – система контроля версий. Позволяет отслеживать все вносимые изменения в проекте при совместной работе
2. В чем недостатки локальных и централизованных СКВ? – Локальные СКВ ограничены работой на одном компьютере и не поддерживают совместную работу. Централизованные СКВ могут создавать единую точку отказа и зависеть от доступности центрального сервера.
3. К какой СКВ относится Git? – Git относится к распределенным системам контроля версий.
4. В чем концептуальное отличие Git от других СКВ? – Концептуальное отличие Git заключается в распределенности, где каждый участник имеет полную копию репозитория, и целостности данных, обеспечиваемой хэшем.
5. Как обеспечивается целостность хранимых данных в Git? – Целостность данных в Git обеспечивается с помощью хэш-сумм (SHA-1), которые проверяются для каждого коммита и файла.
6. В каких состояниях могут находиться файлы в Git? Как связаны эти состояния? – Файлы в Git могут находиться в трех состояниях: рабочий каталог, индекс и репозиторий. Файлы сначала изменяются

в рабочем каталоге, затем добавляются в индекс и затем фиксируются в репозитории.

7. Что такое профиль пользователя в GitHub? – Профиль пользователя в GitHub — это учетная запись пользователя, которая содержит информацию о нем, его репозиториях, активности и другие данные.
8. Какие бывают репозитории в GitHub? – В GitHub существуют публичные репозитории, к которым все могут получить доступ, и приватные репозитории, доступные только выбранным пользователям.
9. Укажите основные этапы модели работы с GitHub. – Создание аккаунта, создание/клонирование репозитория, добавление/изменение файлов, фиксация изменений (коммит), отправка изменений (push), работа с ветками, запросы на слияние (pull requests), совместная работа.
10. Как осуществляется первоначальная настройка Git после установки? – После установки Git необходимо настроить имя пользователя и email с помощью команд `git config --global user.name "Your Name"` и `git config --global user.email "youremail@example.com"`.
11. Опишите этапы создания репозитория в GitHub. – Зайти в аккаунт на GitHub, нажать на "+" в верхнем правом углу и выбрать "New Repository", заполнить информацию о репозитории, создать.
12. Какие типы лицензий поддерживаются GitHub при создании репозитория? – GitHub предоставляет выбор из различных лицензий, включая MIT License, GNU General Public License и другие.
13. Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий? – Клонирование репозитория GitHub выполняется командой `git clone URL`, это создает локальную копию удаленного репозитория. Клонирование необходимо для работы с проектом на локальном компьютере и совместной разработки.

14. Как проверить состояние локального репозитория Git? – Состояние локального репозитория Git можно проверить с помощью команды `git status`, которая покажет измененные и неотслеженные файлы.
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` ? – После добавления/изменения файла с помощью `git add`, файлы попадают в индекс. После коммита с помощью `git commit`, изменения фиксируются в локальном репозитории. Отправка изменений на сервер с помощью `git push` обновляет удаленный репозиторий.
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`. – Для того чтобы оба локальных репозитория, связанных с репозиторием GitHub, оставались в синхронизированном состоянии, вам нужно выполнить следующую последовательность команд. В данном примере предполагается, что у вас уже есть клон репозитория на одном из компьютеров:
- 1) Сначала склонируйте репозиторий с GitHub на первом компьютере, если вы этого еще не сделали: `git clone <URL репозитория GitHub>`
  - 2) Теперь у вас есть локальная копия репозитория на первом компьютере.

- 3) На втором компьютере также клонируйте репозиторий с GitHub: `git clone <URL репозитория GitHub>`
- 4) Теперь у вас есть две локальные копии репозитория - одна на первом компьютере и другая на втором.
- 5) Перед началом работы на любом из компьютеров, убедитесь, что вы находитесь в рабочей директории вашего локального репозитория: `cd <директория репозитория>`
- 6) Прежде чем начать работу, убедитесь, что вы находитесь на актуальной ветке и ваш локальный репозиторий обновлен: `git checkout <название ветки> git pull origin <название ветки>`
- 7) Теперь вы можете вносить изменения и работать над проектом.
- 8) Когда вы завершите работу и хотите сохранить изменения в удаленном репозитории на GitHub, выполните следующие команды: `git add .`

`git commit -m "Ваш комментарий к коммиту" git push origin <название ветки>` Это отправит ваши локальные изменения на сервер GitHub и обновит удаленный репозиторий.

Теперь ваши два локальных репозитория и репозиторий на GitHub будут находиться в синхронизированном состоянии, и изменения, внесенные с одного компьютера, будут доступны на другом.

17. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub. – GitHub и GitLab - два популярных сервиса для работы с Git. Оба предоставляют облачные и локальные варианты, инструменты для управления кодом и CI/CD, а также интеграции с другими приложениями. GitHub более популярен для открытых проектов и имеет большее сообщество, в то время как GitLab предоставляет бесплатный вариант с открытым



исходным кодом и больше гибкости в управлении сервером. Выбор зависит от потребностей вашего проекта.

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. – Существует множество программных средств с графическим интерфейсом пользователя (GUI) для работы с Git. Некоторые из наиболее известных и популярных инструментов в этой категории включают: GitHub Desktop: GitHub Desktop — это официальный клиент GitHub с графическим интерфейсом. Он предоставляет интуитивный способ клонирования репозитория, управления ветками, внесения изменений и отправки коммитов на GitHub. Вам даже не нужно использовать командную строку для выполнения основных операций. GitKraken: GitKraken — это популярный графический Git-клиент с поддержкой Windows, macOS и Linux. Он предоставляет интуитивный пользовательский интерфейс для клонирования репозитория, внесения изменений, управления ветками и выполнения других операций Git. SourceTree: SourceTree — это бесплатный Git-клиент от Atlassian. Он предоставляет графический интерфейс для работы с Git и Mercurial, включая возможности управления репозиториями, просмотра истории коммитов и разрешения конфликтов.

Давайте рассмотрим, как выполняются некоторые операции Git с использованием GitHub Desktop в качестве примера:

1. Клонирование репозитория:

- Запустите GitHub Desktop.

- Выберите "File" (Файл) в верхнем левом углу и выберите "Clone Repository" (Клонировать репозиторий).
- Выберите репозиторий из списка или введите URL репозитория.
- Укажите путь для локального клонирования.
- Нажмите "Clone" (Клонировать).

## 2. Внесение изменений и коммит:

- В разделе "Changes" (Изменения) вы увидите все неотправленные изменения.
- Выберите файлы, которые вы хотите включить в коммит.
- Введите комментарий к коммиту.