

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 №1.1
дисциплины «Основы программной инженерии»

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

(подпись)

Преподаватель:
Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Исследование основных возможностей Git и GitHub.

Цель: исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

Результаты выполнения

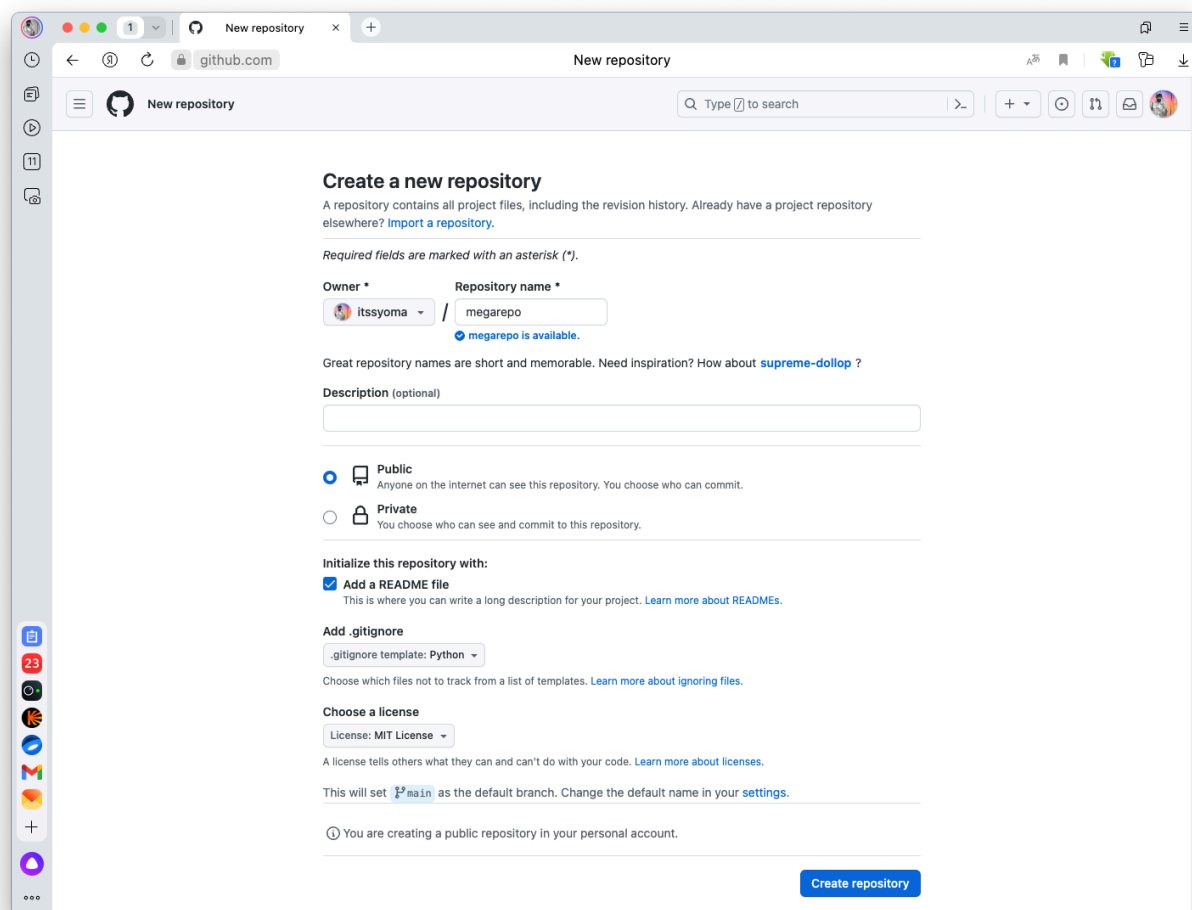


Рисунок 1 – Создание репозитория на GitHub

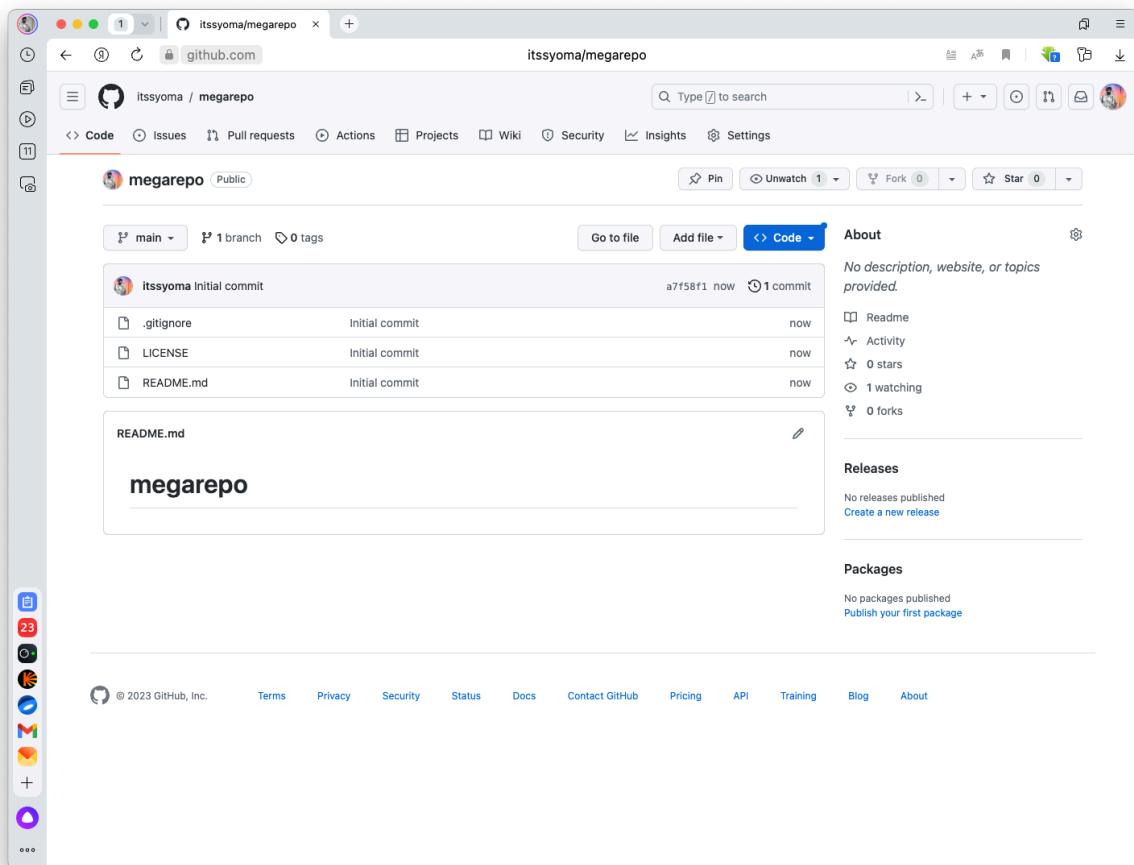


Рисунок 2 – Репозиторий после создания

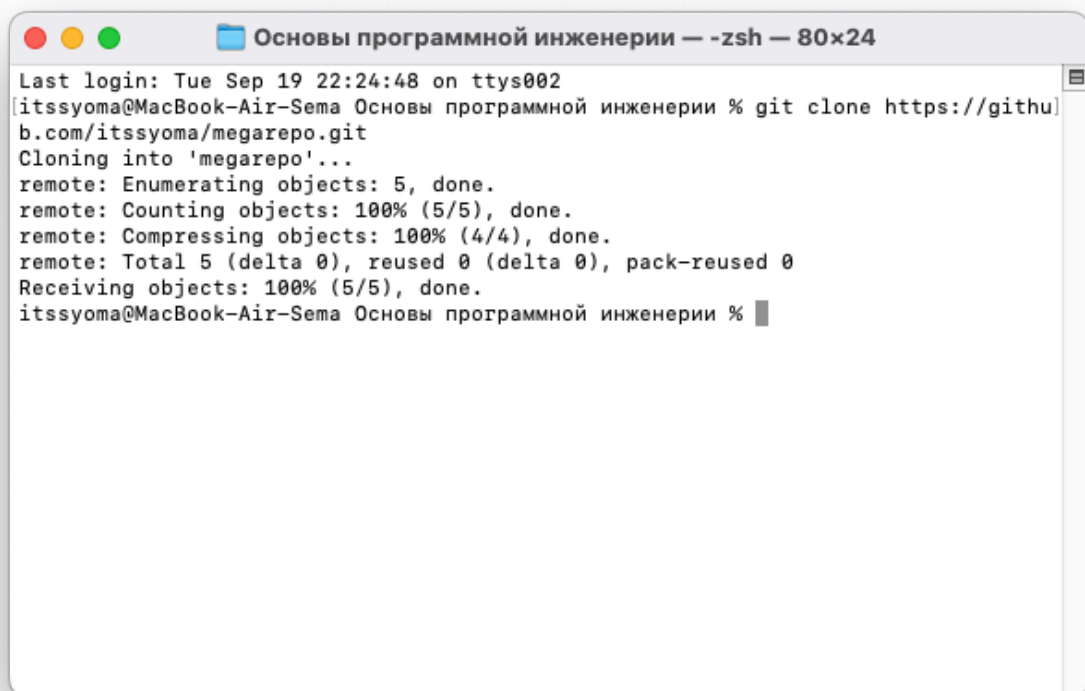
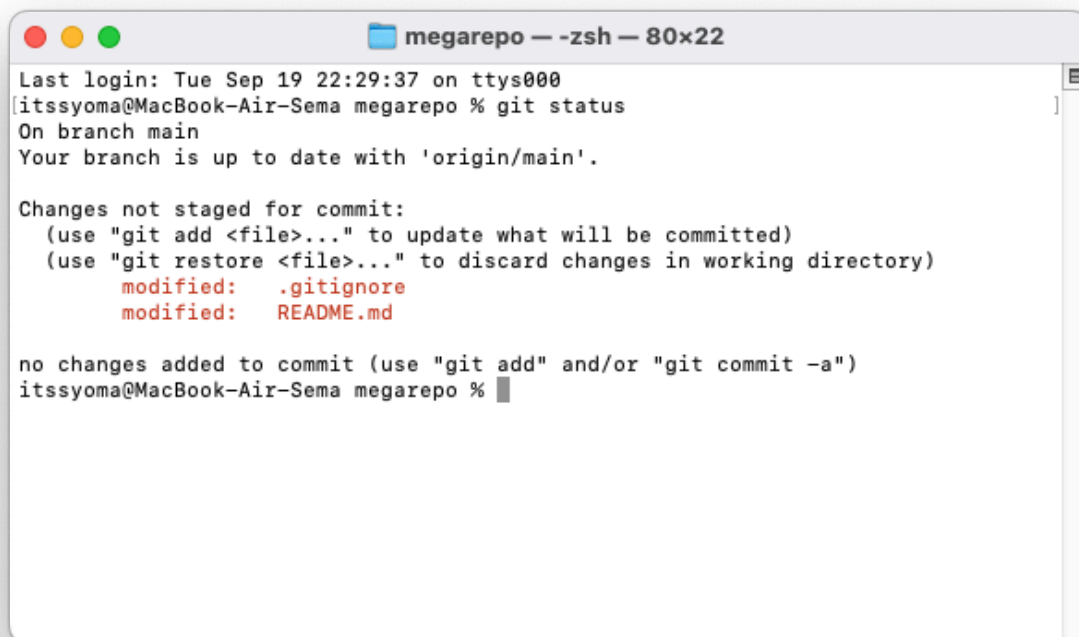


Рисунок 3 – Клонирование репозитория на рабочий компьютер

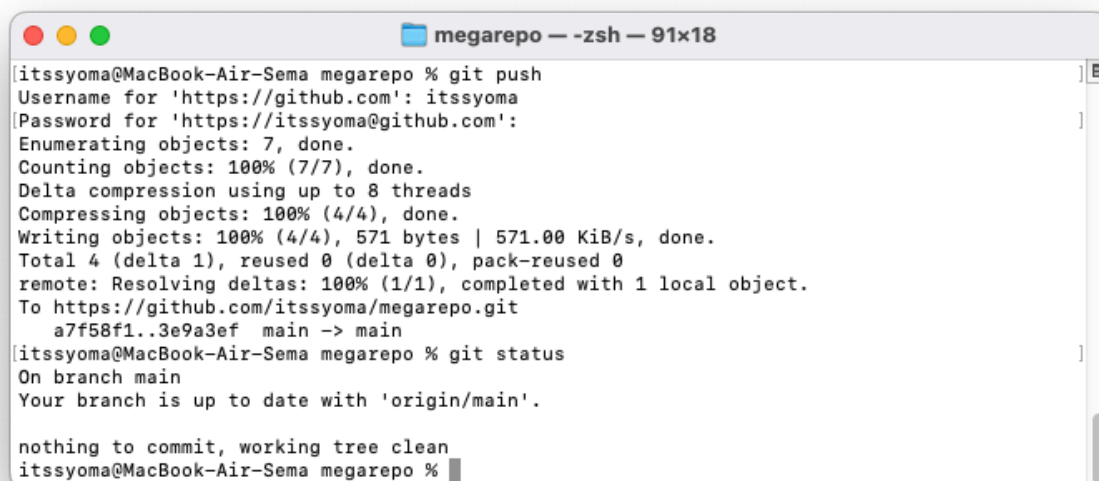
A terminal window titled 'megarepo — -zsh — 80x22'. The output shows the last login time, the user 'itssyoma' at 'MacBook-Air-Sema', and the command 'git status'. The status indicates the branch 'main' is up to date with 'origin/main'. It lists two files as modified: '.gitignore' and 'README.md'. It also provides instructions on how to stage changes for commit using 'git add' or 'git commit -a'.

```
megarepo — -zsh — 80x22
Last login: Tue Sep 19 22:29:37 on ttys000
itssyoma@MacBook-Air-Sema megarepo % git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
itssyoma@MacBook-Air-Sema megarepo %
```

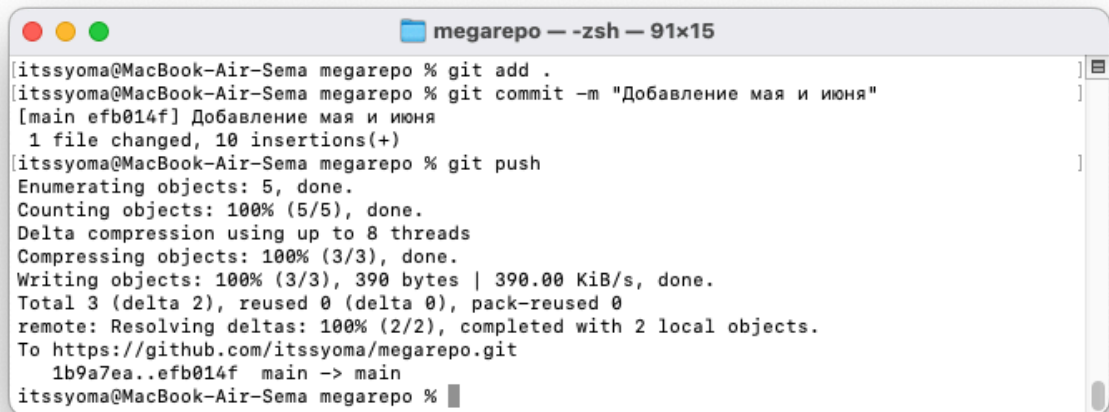
Рисунок 4 – Проверка состояния репозитория после внесения изменений в .gitignore и README

A terminal window titled 'megarepo — -zsh — 91x18'. The output shows the command 'git push' being executed. It prompts for a username and password for 'https://github.com'. The push is successful, showing progress for enumerating, counting, compressing, and writing objects. It also shows the remote repository resolving deltas and the local commit being pushed to the 'main' branch. Finally, it shows the 'git status' command output, indicating the working tree is clean.

```
megarepo — -zsh — 91x18
itssyoma@MacBook-Air-Sema megarepo % git push
Username for 'https://github.com': itssyoma
Password for 'https://itssyoma@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 571 bytes | 571.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/itssyoma/megarepo.git
   a7f58f1..3e9a3ef  main -> main
itssyoma@MacBook-Air-Sema megarepo % git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
itssyoma@MacBook-Air-Sema megarepo %
```

Рисунок 5 – Отправка измененных файлов .gitignore и README на удаленный репозиторий в GitHub



```
megarepo — zsh — 91x15
itssyoma@MacBook-Air-Sema megarepo % git add .
itssyoma@MacBook-Air-Sema megarepo % git commit -m "Добавление мая и июня"
[main efb014f] Добавление мая и июня
1 file changed, 10 insertions(+)
itssyoma@MacBook-Air-Sema megarepo % git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 390 bytes | 390.00 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/itssyoma/megarepo.git
1b9a7ea..efb014f main -> main
itssyoma@MacBook-Air-Sema megarepo %
```

Рисунок 6 – Создание коммитов и отправка файла программы на удаленный репозиторий

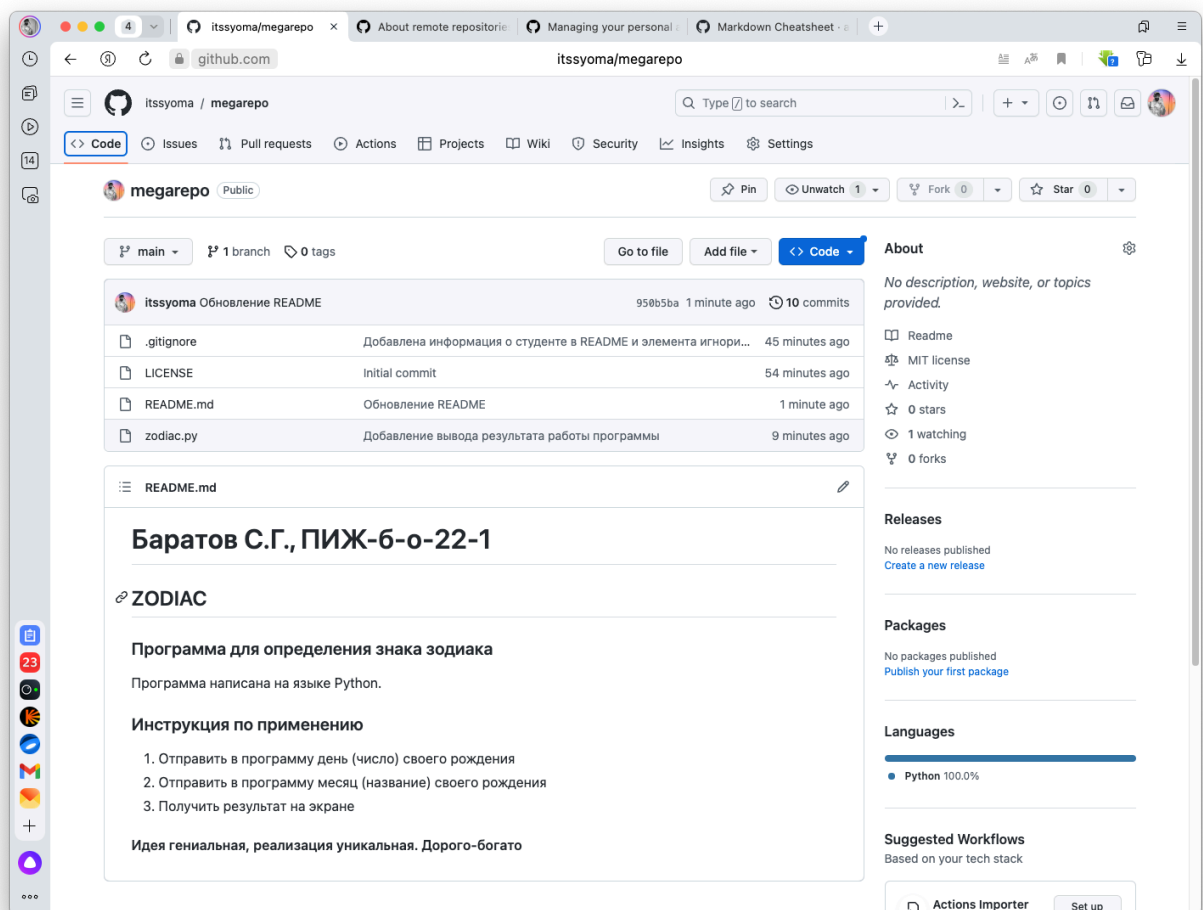


Рисунок 7 – Страница репозитория после внесения всех изменений

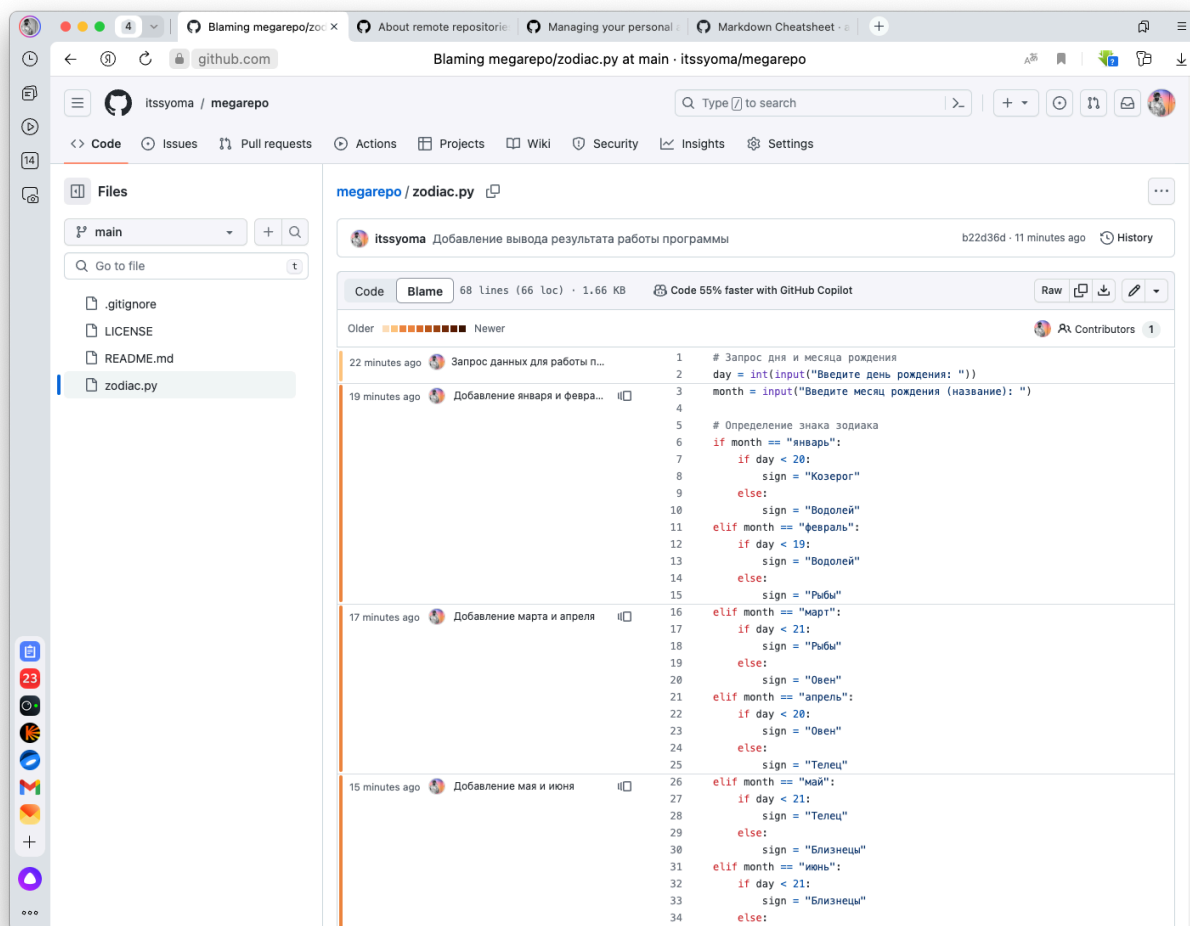


Рисунок 8 – Коммиты файла программы

Ответы на контрольные вопросы

1. Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ) – это инструмент, используемый разработчиками программного обеспечения для управления изменениями в исходном коде и других файловых ресурсах. Она позволяет отслеживать историю изменений, возвращаться к предыдущим версиям, сливать изменения из разных источников и сотрудничать с другими разработчиками.

2. В чем недостатки локальных и централизованных СКВ?

Главный недостаток как локальных, так и централизованных систем контроля версий – это их ненадежность. Если у нас что-то случится с

данными, мы потеряем данные на сервере, мы потеряем сразу все, если не позаботимся о дополнительном бэкапировании этих данных.

3. К какой СКВ относится Git?

Git относится к распределенным СКВ.

4. В чем концептуальное отличие Git от других СКВ?

От других СКВ Git отличается особым подходом к обработке информации: он не записывает отдельно внесенные правки, а делает подробный снимок проекта в момент сохранения, то есть фиксирует состояние каждого файла, и создает ссылку на эту версию.

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 философии. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git.

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). Если определённая версия файла есть в Git-директории, эта версия считается зафиксированной. Если версия файла изменена и добавлена в индекс, значит, она подготовлена. И если файл был изменён с момента последнего распаковывания из репозитория, но не был добавлен в индекс, он считается изменённым.

7. Что такое профиль пользователя в GitHub?

Профиль – это публичная страница на GitHub, как и в социальных сетях. При поиске работы в качестве программиста, работодатели могут

посмотреть профиль GitHub и принять его во внимание, когда будут решать, брать на работу или нет.

8. Какие бывают репозитории в GitHub?

Репозитории бывают публичные и приватные.

9. Укажите основные этапы модели работы с GitHub.

Создание репозитория GitHub, клонирование репозитория, локальное изменение содержимого, распространение в исходный удаленный репозиторий.

10. Как осуществляется первоначальная настройка Git после установки?

Нужно убедиться, что Git установлен, с помощью команды *git status*; добавить в настройки имя пользователя и электронную почту аккаунта GitHub.

11. Опишите этапы создания репозитория в GitHub.

Указание названия и типа репозитория, описания, выбор лицензии, создание файлов README и .gitignore, выбор языка программирования.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

GitHub поддерживает лицензии свободного и несвободного использования.

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

Клонирование репозитория осуществляется командой *git clone <url>*. Стандартный подход к работе с проектом состоит в том, чтобы иметь локальную копию репозитория и фиксировать ваши изменения в этой копии, а не в удаленном репозитории, размещенном на GitHub. Этот локальный репозиторий имеет полную историю версий проекта, которая может быть полезна при разработке без подключения к интернету.

14. Как проверить состояние локального репозитория Git?

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` ?

После добавления/изменения файла в локальный репозиторий Git, состояние локального репозитория изменится, и Git будет отслеживать эти изменения.

После добавления нового/измененного файла под версионный контроль с помощью команды `git add`, Git добавит этот файл в индекс, который содержит список файлов, которые будут включены в следующий коммит.

После фиксации (коммита) изменений с помощью команды `git commit`, Git сохранит изменения в локальном репозитории и создаст новый коммит, который будет содержать эти изменения.

После отправки изменений на сервер с помощью команды `git push`, Git отправит изменения из локального репозитория на серверный репозиторий и обновит его состояние.

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. Склонировать репозиторий с помощью команды `git clone` на оба компьютера.
2. Внести изменения в файлы проекта на одном из компьютеров.
3. Добавить изменения в индекс с помощью команды `git add` на этом же компьютере.

4. Сделать коммит изменений с помощью команды *git commit -m "описание изменений"* на этом же компьютере.

5. Отправить изменения на серверный репозиторий с помощью команды *git push* на этом же компьютере.

6. Получить изменения с серверного репозитория на другом компьютере с помощью команды *git pull*.

7. Продолжить работу над проектом на другом компьютере, внести изменения, добавить их в индекс, сделать коммит и отправить на серверный репозиторий с помощью соответствующих команд.

8. Повторить шаг 6 на первом компьютере, чтобы получить изменения, внесенные на другом компьютере.

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

Некоторые известные сервисы для работы с Git, помимо GitHub, включают в себя Bitbucket, GitLab и SourceForge. Выбор между Bitbucket и GitHub зависит от конкретных потребностей команды и проекта. Если нужны бесплатные приватные репозитории для небольших команд, то Bitbucket может быть лучшим выбором. Если же нужна большая поддержка сообщества и интеграция с другими сервисами разработки, то GitHub может быть предпочтительнее.

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.

Некоторые известные программные средства с графическим интерфейсом пользователя для работы с Git включают в себя GitKraken, Sourcetree и Git GUI.

Как операции Git могут быть выполнены с помощью GitKraken:

1. Клонирование репозитория: Чтобы клонировать репозиторий в GitKraken, нужно выбрать опцию "Clone a repo" и ввести URL репозитория. GitKraken также предоставляет возможность клонировать репозиторий из GitHub, Bitbucket и GitLab.

2. Создание ветки: Чтобы создать ветку в GitKraken, нужно перейти на вкладку "Branches" и выбрать опцию "New branch". Затем нужно ввести имя новой ветки и выбрать базовую ветку.

3. Выполнение коммита: Чтобы выполнить коммит в GitKraken, нужно выбрать файлы, которые нужно закоммитить, написать сообщение коммита и нажать кнопку "Commit".

4. Отправка изменений на сервер: Чтобы отправить изменения на сервер в GitKraken, нужно выбрать опцию "Push" и выбрать ветку, на которую нужно отправить изменения.

В целом, GitKraken обеспечивает удобный и интуитивно понятный интерфейс для работы с Git, что может быть полезно для новичков в Git или для тех, кто предпочитает работать с графическим интерфейсом пользователя.