

# **ПРАКТИЧЕСКАЯ РАБОТА**

## **ОРГАНИЗАЦИЯ РЕПОЗИТОРИЯ В СИСТЕМЕ УПРАВЛЕНИЯ ВЕРСИЯМИ GIT**

### **1. ЦЕЛЬ И ЗАДАЧИ ПРАКТИЧЕСКОЙ РАБОТЫ**

**Цель:** получение практических навыков работы с распределенной системой управления версиями Git и сервисом GitHub.

Для достижения поставленной цели требуется решить следующие **задачи**:

1. Создать пространство проекта на сервисе GitHub.
2. Разработать три ветви проекта.
3. Выполнить форк внешнего репозитория.

#### **Структура методических рекомендаций**

В разделе 1 представлены цели и задачи практической работы. Также дается пояснение выбора проекта разработки.

В разделе 2 показаны критерии оценивания. Внимательно изучите этот раздел.

В разделе 3 представлены назначение систем управления версиями и рассмотрены возможности системы под управлением Git. В этом же разделе показаны возможности онлайн-сервиса GitHub.

В конце методических указаний представлен список литературы.

#### **Общие рекомендации**

Онлайн-сервис GitHub позволяет создавать репозитории проектов без глубоких знаний программирования. Все требуемые команды git будут сгенерированы автоматически на основе действий пользователя. В репозитории проекта можно хранить любые файлы. Текстовые файлы можно создавать непосредственно в репозитории с помощью языка Markdown. Там же можно их изменять. Если возможностей текстового редактора не хватает для внесения изменений, то файлы скачивают на персональный компьютер. В этом случае на компьютере у пользователя должна быть установлена программа GitHub Desktop. Её пространство полностью синхронизировано с онлайн-пространством GitHub. При внесении изменений в файлы они будут автоматически переданы в облачную версию.

В практической работе предусмотрен вариант использования только облачной версии GitHub. На рабочий компьютер ничего дополнительно устанавливать

не требуется. Перед выполнением задания подготовьте 5 файлов разного типа, например с расширениями .txt, .jpg, .docx, .py, .pdf.

Внимательно прочитайте методические указания, посмотрите мастер-класс и выполните упражнения.

Если вы знакомы и работаете с GitHub (GitLab и др.), то можно сделать несколько скриншотов из своего репозитория. Достаточно продемонстрировать факт того, что вы умеете работать с такими сервисами.

Если вы мало знакомы или не имеете своего репозитория, то выполните все упражнения. Скриншоты выполнения каждого упражнения в отчете показывать не требуется. Достаточно 5-6 скриншотов, которые подтверждают, что вы создали репозиторий, заполнили его и научились создавать ветви. Также можете форкнуть любой выбранный репозиторий. Выбор репозитория, который планируете перенести осуществите самостоятельно, например используя список самых интересных репозиториях [1].

Обратите внимание, что платформа GitHub может обновить интерфейс и расположение функциональных элементов может измениться. В методических указаниях показаны элементы взаимодействия по состоянию на февраль 2024 года.

Если справиться с заданием не удастся, то в отчете покажите на скриншотах, на каком этапе произошли критические проблемы, в выводе попробуйте дать им объяснения. Также рекомендуем посмотреть мастер-класс по организации репозитория, который выложен в курсе.

### **Последовательность выполнения задания**

1. Зарегистрироваться на сервере GitHub.
2. Создать не менее трех ветвей проекта, в каждой из которых сохранить файлы разных форматов: код программы, выполненный на любом языке программирования, или текстовые файлы, например, отчеты по лабораторным работам, графические материалы и т.д.
3. Каждое изменение (новую ветвь, дополнение репозитория) фиксировать в системе контроле версий.
4. Скопируйте любой программный проект, который хранится на GitHub, например, библиотеку pytorch, в свой удаленный репозиторий.
5. Выполните отчет в соответствии с требованиями к структуре и наполнению.

Примеры выполнения пунктов задания приведены в [2], а также в многочисленных интернет-источниках.

## Структура отчет по практической работе

- 1) Цель работы.
- 2) Прогресс по реализации проекта в Kaiten.
- 3) Созданный аккаунт на GitHub с календарем активностей (рис.1.1- 1.2).

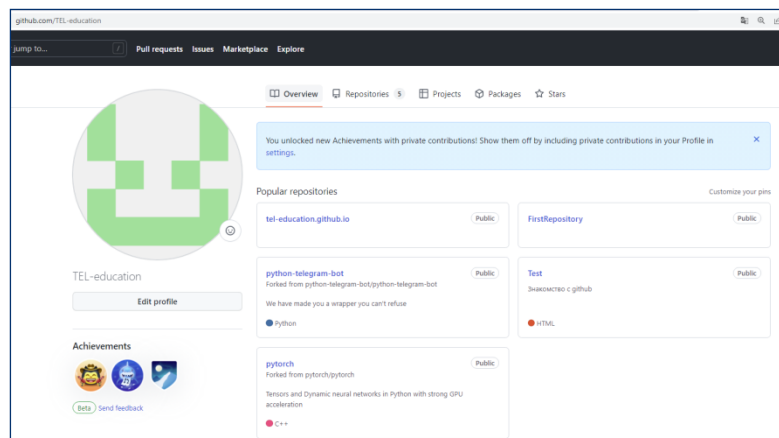


Рисунок 1.1 - Пример аккаунта

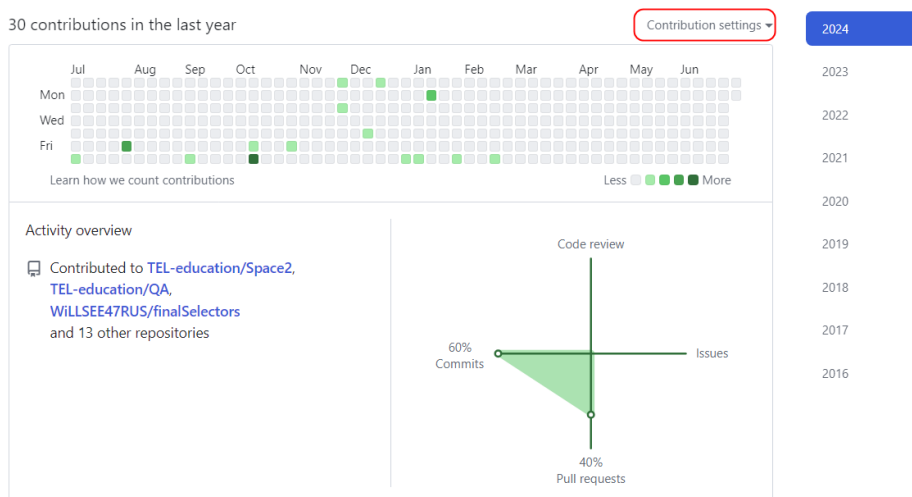


Рисунок 1.2 - Активность аккаунта

- 4) Скриншоты двух ветвей проекта на удаленном сервере GitHub (рис.1.3- 1.4).

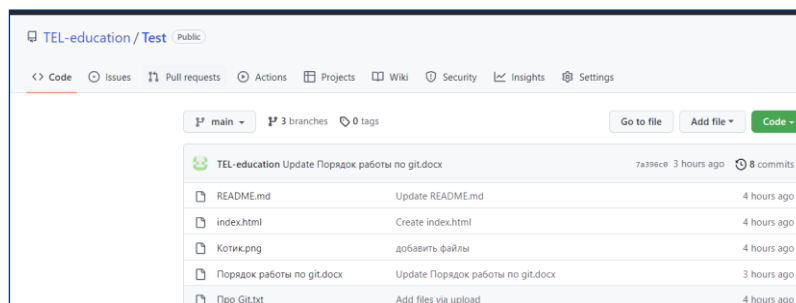


Рисунок 1.3 - Ветвь main

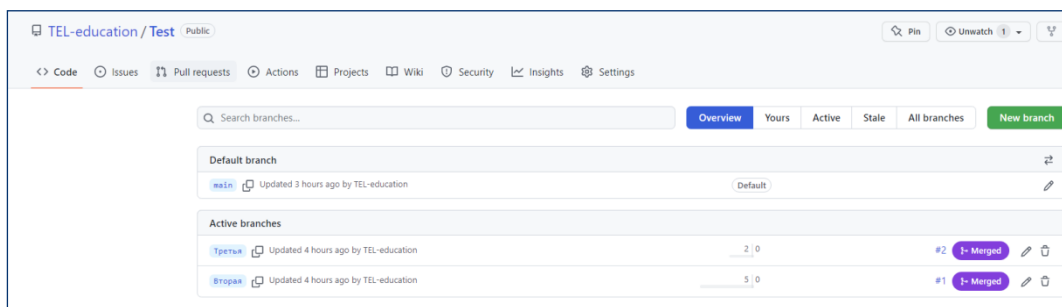


Рисунок 1.4 - Ветви проекта

- 5) Скриншот копии выбранного проекта с полезными библиотеками (рис.1.5).

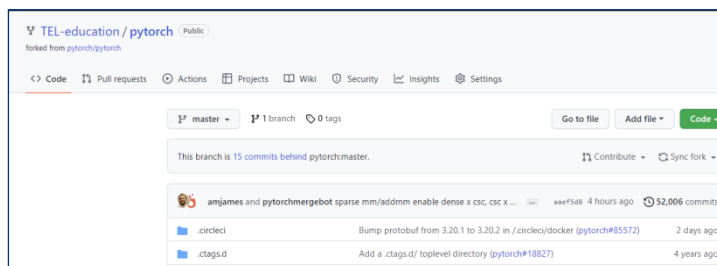


Рисунок 1.5 - Пример скриншота копии библиотеки pytorch

- 6) Ссылка на удаленный репозиторий на GitHub, реализованный в ходе выполнения практической работы.
- 7) Вывод, в котором в формате эссе раскройте цель выполненной работы, описываются знания и навыки, полученные в процессе ее выполнения, а также возникшие проблемы и пути их решения.
- 8) Список использованных источников.

## 2. КРИТЕРИИ ОЦЕНИВАНИЯ ПРАКТИЧЕСКОЙ РАБОТЫ

Оценивание практической работы выполняется по следующим критериям. За выполнение работы выставляются баллы, которые потом переводятся в оценку в случае дифференцированного зачета или зачет/не зачет в случае зачета без оценки.

Таблица 2.1 – Критерии оценивания практической работы

| № | Критерий                                  | Балл                           |
|---|---|--------------------------------|
| 1 | Сформулирована цель работы                | да – 1 балл<br>нет – 0 баллов  |
| 2 | Дана ссылка на репозиторий проекта        | да – 1 балл<br>нет – 0 баллов  |
| 3 | Показано не менее 3 ветвей проекта        | да – 1 балла<br>нет – 0 баллов |
| 4 | Создана копия (fork) внешнего репозитория | да – 1 балл<br>нет – 0 баллов  |

|                            |  |   |
|----------------------------|--|---|
| 5                          | В отчете представлены все предусмотренные заданием скриншоты | да – 3 баллов<br>частично – 2 балла<br>нет – 0 баллов |
| 6                          | Наличие комментариев к скриншотам                            | да – 1 балла<br>нет – 0 баллов                        |
| 7                          | Вывод, соответствующий требованиям                           | да – 1 балла<br>нет – 0 баллов                        |
| 8                          | Наличие списка использованных источников                     | да – 1 балл<br>нет – 0 баллов                         |
| Максимально возможный балл |  | 10  |

Таблица 2.2 – Перевод баллов в оценку

| Баллы          | Оценка              | Зачет/не зачет |
|----------------|---------------------|----------------|
| Менее 4 баллов | неудовлетворительно | Не зачет       |
| 4-6 баллов     | удовлетворительно   | Зачет          |
| 6-8 баллов     | хорошо              |                |
| 8-10 баллов    | отлично             |                |

### 3. НАЗНАЧЕНИЕ СИСТЕМ УПРАВЛЕНИЯ ВЕРСИЯМИ

При работе над программным проектом каждый из команды разработчиков должен иметь доступ ко всем файлам. Поэтому возникает необходимость организации виртуального хранилища программных компонент с системой контроля версий.

Существуют множество систем управления версиями. Условно их можно разделить три их главные группы:

- в соответствии с расположением репозитория: централизованные и распределенные;
- в соответствии с методами проверки слияния и передачи кода: блокирующие, использующие слияние до фиксации и выполняющие фиксацию до слияния;
- системы управления версиями могут выполнять небольшие операции или операции с файлами.

Работу с системами контроля версий рассмотрим на примере распределенной системы управления версиями Git<sup>1</sup>.

Программные средства данной системы позволяют сохранить изменения в файл или набор файлов в процессе их модификации и при необходимости вернуться к конкретной версии файла. Если над проектом работают несколько человек, то каждому из них обеспечивают доступ для совместной работы над фай-

<sup>1</sup> <https://git-scm.com/>

лом. Каждое внесенное изменение фиксируют, поэтому возникает многоверсионность разрабатываемого программного продукта. Недостатком Git можно считать то, что он имеет интерфейс командной строки. Чтобы упростить взаимодействие разработчиков с этой системой, был создан графический интерфейс GitHub<sup>2</sup>.

GitHub (<https://github.com/>) – крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Веб-сервис основан на системе контроля версий Git. Ознакомиться с документацией по работе с GitHub можно на сайте GitHub Help (<https://help.github.com/en>) .

### Упражнение 1. Создание аккаунта GitHub.

1.Зарегистрируйтесь на сервисе GitHub (<https://github.com/>). Для этого перейдите на сайт и нажмите кнопку Sign up (рис.3.1). В открывшемся меню введите параметры учетной записи: имя пользователя; адрес электронной почты; пароль.

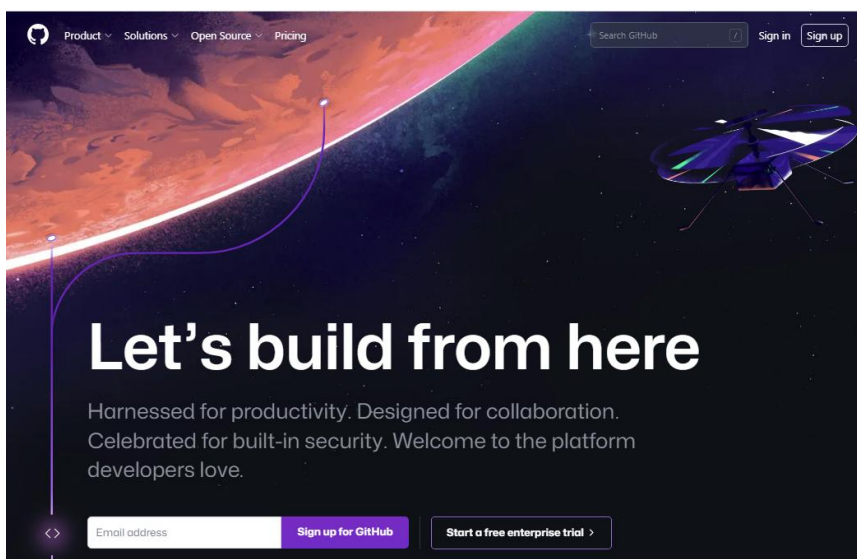


Рисунок 3.1 - Регистрация на сервисе GitHub

2. Полное имя аккаунта посмотрите на вкладке Профиль, например TEL-education.github.io (рис.3.2).

---

<sup>2</sup> <https://github.com>

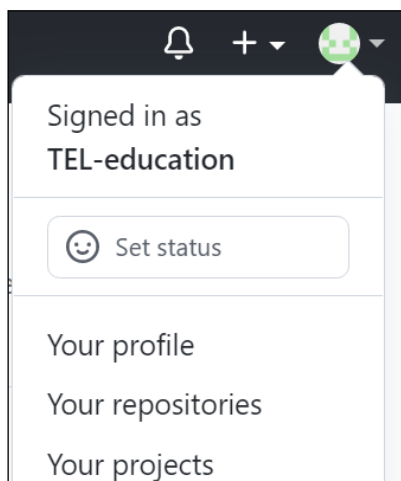


Рисунок 3.2 - Проверка имени аккаунта

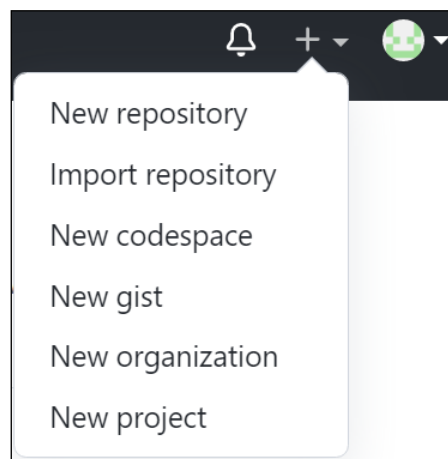


Рисунок 3.3 - Создание нового репозитория  
New repository

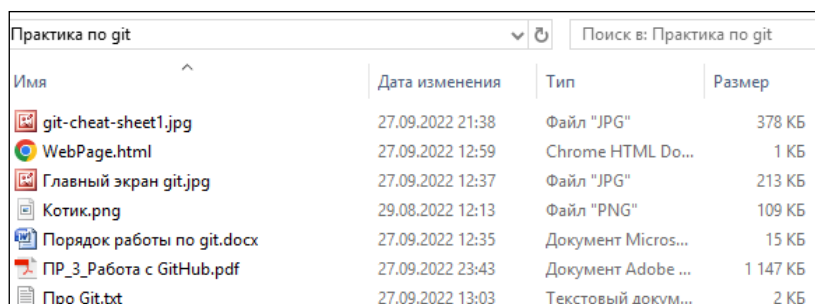
3. После регистрации создайте свой первый репозиторий. Для этого нажмите на знак Плюс в правом верхнем углу окна GitHub. Назовите репозиторий, например *Space* и нажмите кнопку **Create Repository** (рис. 3.3-3.4). Все названия проектов на GitHub необходимо писать латинским алфавитом. Названия файлов в рассмотренных ниже примерах приводятся на русском языке, чтобы упростить понимание процесса взаимодействия с удаленным репозиторием. В описание репозитория внесите: «Знакомство с GitHub».

Рисунок 3.4 - Создание репозитория на сервисе GitHub

## Упражнение 2. Копирование файлов проекта в репозиторий на GitHub

1. Подготовьте на персональном компьютере не менее пяти файлов для

копирования их в репозиторий (рис.3.5). Файлы могут иметь любой формат и размер и быть созданы в любых программных средах.



| Имя                        | Дата изменения   | Тип                | Размер   |
|----------------------------|------------------|--------------------|----------|
| git-cheat-sheet1.jpg       | 27.09.2022 21:38 | Файл "JPG"         | 378 КБ   |
| WebPage.html               | 27.09.2022 12:59 | Chrome HTML Do...  | 1 КБ     |
| Главный экран git.jpg      | 27.09.2022 12:37 | Файл "JPG"         | 213 КБ   |
| Котик.png                  | 29.08.2022 12:13 | Файл "PNG"         | 109 КБ   |
| Порядок работы по git.docx | 27.09.2022 12:35 | Документ Micros... | 15 КБ    |
| ПР_3_Работа с GitHub.pdf   | 27.09.2022 23:43 | Документ Adobe ... | 1 147 КБ |
| Про Git.txt                | 27.09.2022 13:03 | Текстовый докум... | 2 КБ     |

Рисунок 3.5 - Подготовленные для копирования файлы

2. После создания репозитория будет доступно окно для выбора дальнейших действий пользователя (рис.3.6). Выберите `uploading an existing file` (загрузку внешних файлов).

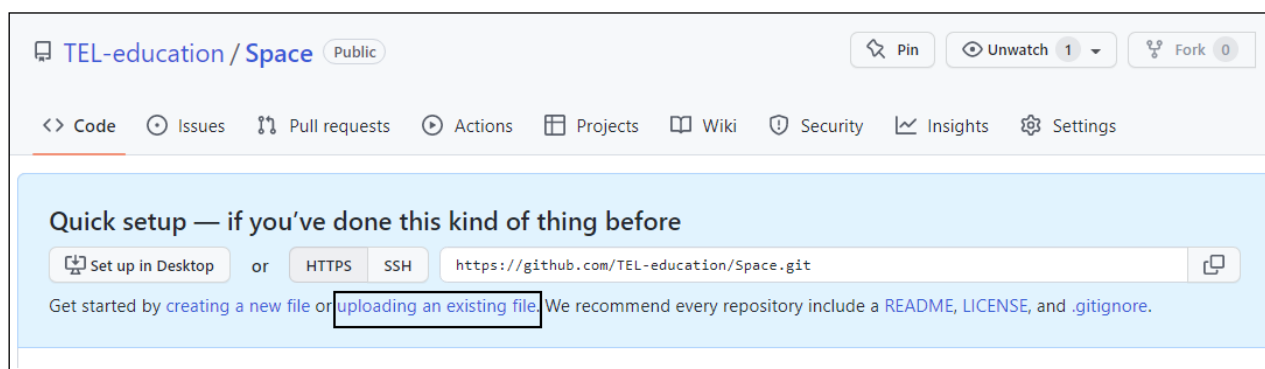


Рисунок 3.6 - Выбор опции `uploading an existing`

3. Затем перетяните в открывшееся окно два любых файла для копирования в основную ветвь проекта. Основная ветвь проекта по умолчанию названа `main`. Обратите внимание, на автоматически сгенерированную команду `Add files via upload`, которая появилась в разделе *Commit changes* (рис.3.7).

Добавьте описание, объясняющее, какое именно изменение было сделано. В рассматриваемом примере: «Копирование двух файлов проекта Space». Описания фиксируют историю изменений в коде проекта, чтобы другие участники проекта могли увидеть изменения и понять их назначение. Нажмите на кнопку **Commit changes** и завершите первый коммит.



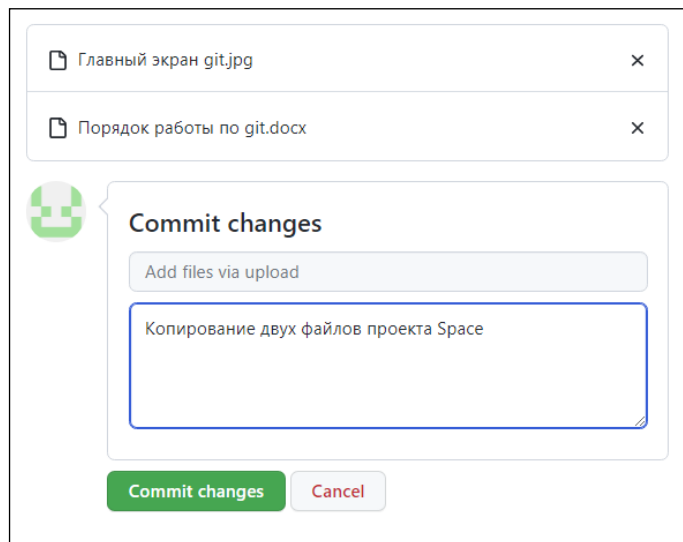


Рисунок 3.7 - Добавление файлов проекта в репозиторий

2. Результат копирования файлов с персонального компьютера в облачное хранилище показан на рис.3.8.

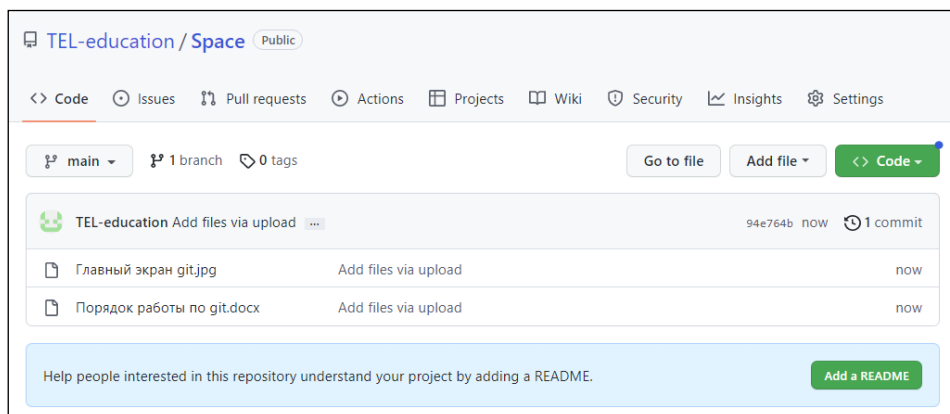


Рисунок 3.8 - Ветви проекта Обучение в репозитории TEL-education

На этом этапе в репозитории **TEL-education** создан проект **Space**, в котором присутствует только одна ветвь проекта `1branch` (первая ветвь) под названием `main`, в которой сохранены два файла.

3. В каждом проекте предусмотрено его описание в файле `README.md`. Нажмите на кнопку **Add a README** для перехода в область для создания файла.

4. В текстовую область вкладки меню *Edit new file* (рис.3.9) уже внесено описание репозитория: «Знакомство с GitHub». В области *Commit new file* обратите внимание на команду `Create README.md`. Внесите описание коммита: «Добавление файла README».

5. Выполните слияние с основной ветвью проекта `main branch`, нажав на кнопку **Commit new file**.

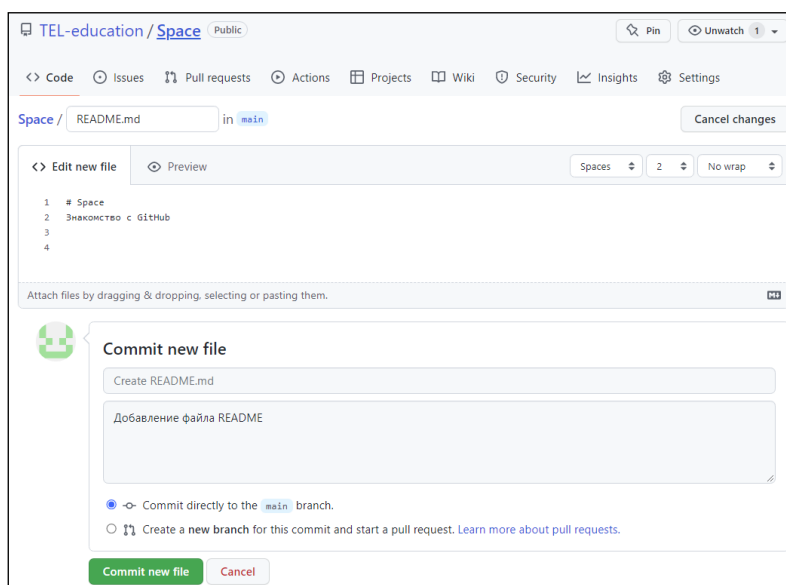


Рисунок 3.9 - Подготовка нового коммита по добавлению файла `README.md`.

После этого шага в репозитории `Space` сохранены три файла (рис.3.10).

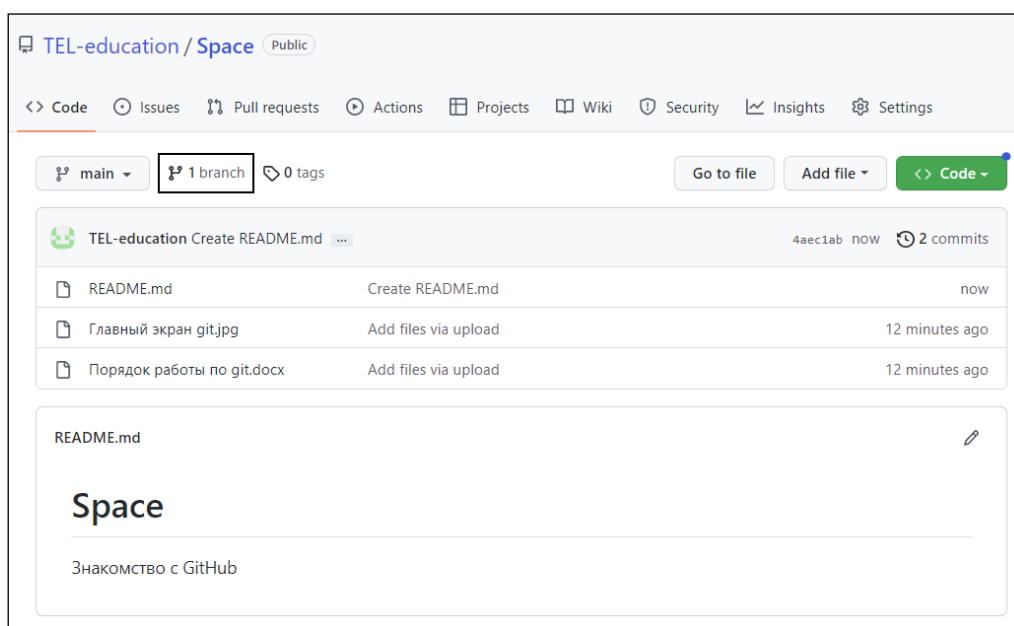


Рисунок 3.10 - Количество файлов в ветви `main` после первого коммита

### Упражнение 3. Добавление второй ветви проекта `Space`

1. Кликните по пиктограмме `1 branch` (рис. 3.10).
2. Нажмите на кнопку **New branch** для добавления новой ветви проекта (3.11).

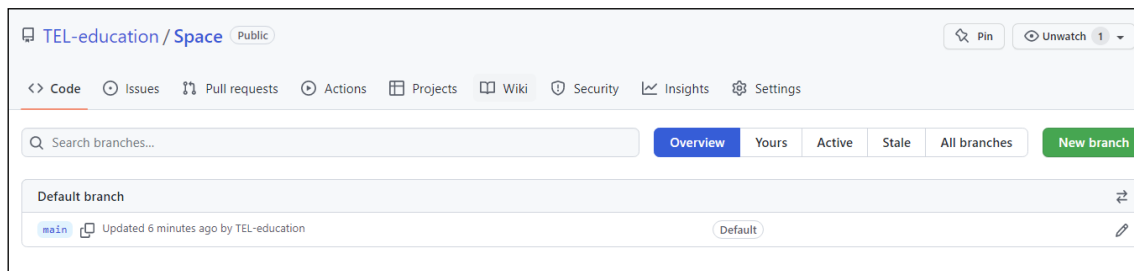


Рисунок 3.11 - Начало создания новой ветви в репозитории Space

3. Дайте новой ветви понятное и лаконичное название, например, ВтораяВетвь . Источником данных этой ветви будут данные ветви main – три файла проекта Space. Нажмите **Create branch** (рис.3.12). После создания новой ветви репозиторий включает две ветви (рис. 3.13). На данном шаге в каждой ветви сохранено по три одинаковых файла.

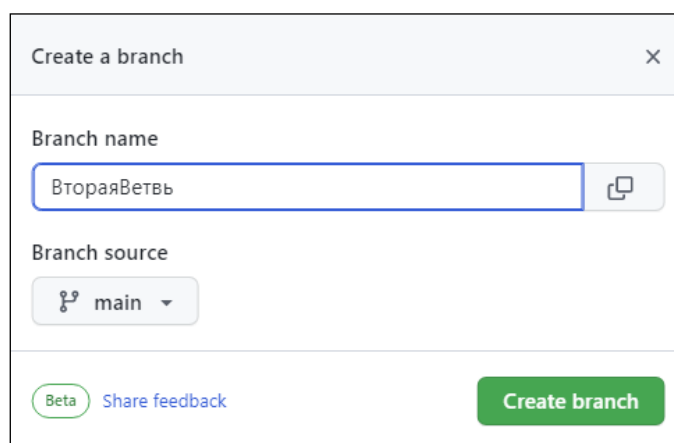


Рисунок 3.12 - Создание второй ветви в репозитории Space

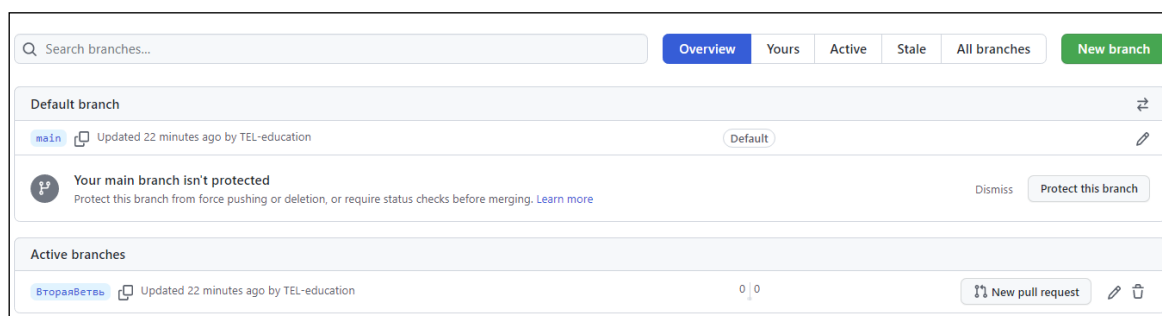


Рисунок 3.13 - Две ветви репозитория Space

При создании второй ветви GitHub потребует уточнений в политике безопасного слияния ветвей проекта. Нажмите **Protect this branche**. Будет произведен переход на новую страницу с политиками GitHub. В учебном проекте не активуйте чек-боксы, оставьте все настройки без изменений и примите соглашение о политике управления интеграцией.

4. Нажмите на название репозитория Space. Затем выберите из раскрывающегося меню название ВтораяВетвь (рис.3.14).

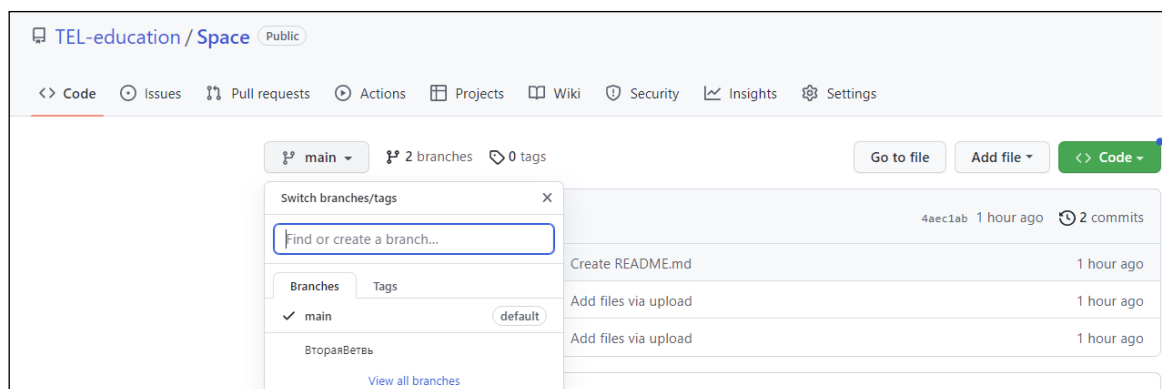


Рисунок 3.14 - Переход к данным, расположенным во второй ветви

5. Добавьте еще два файла во вторую ветвь репозитория. Для этого нажмите на **Add files** и выберите из раскрывающегося списка **Upload files** (рис.3.15).

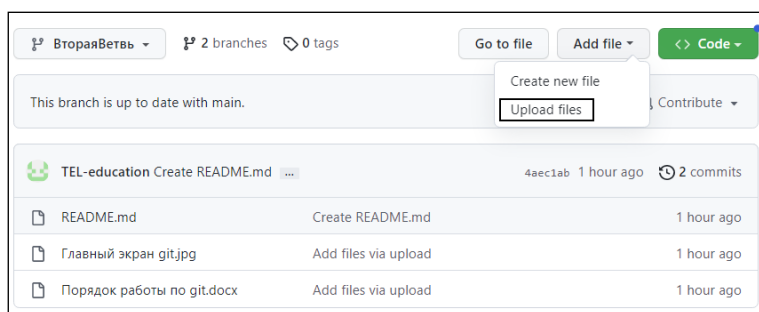


Рисунок 3.15 - Выбор варианта загрузки файлов

6. Скопируйте два файла из папки, подготовленной на персональном компьютере. Добавьте описание коммита и нажмите **Commit change** (рис. 3.16).

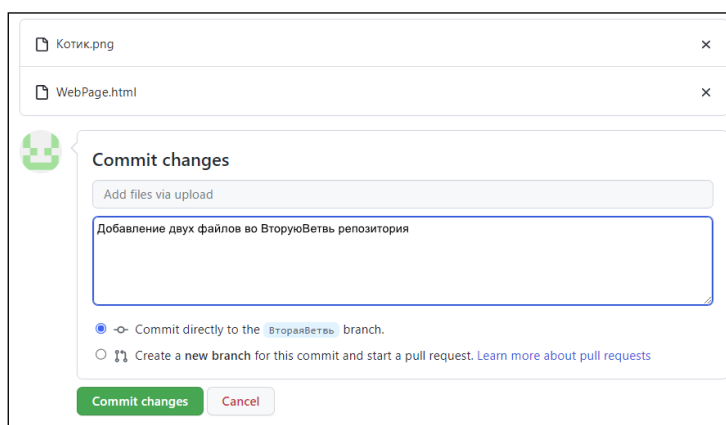


Рисунок 3.16- Добавление файлов во вторую ветвь

7. Нажмите на название репозитория Space и посмотрите на результат изменения количества файлов (рис.3.17).

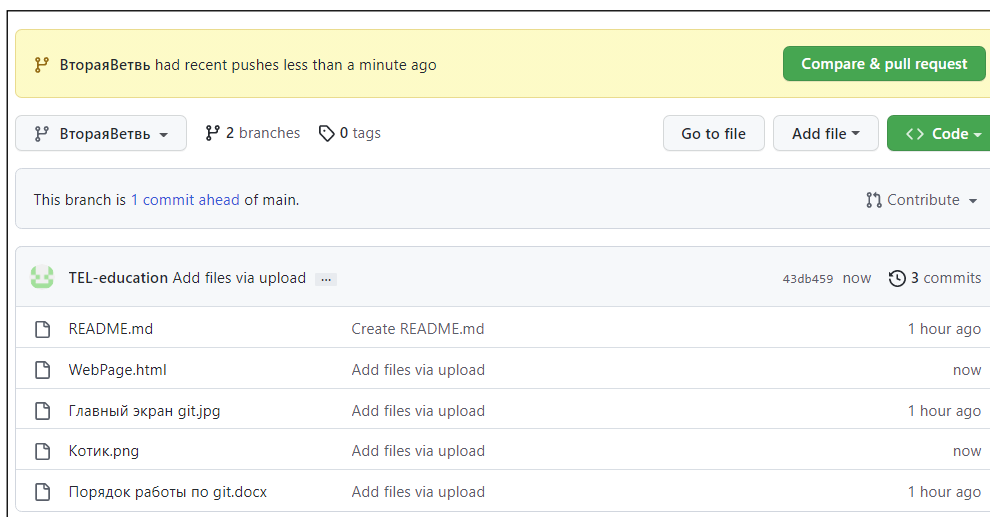


Рисунок 3.17 - Просмотр новых данных во второй ветви

Обратите внимание, что количество ветвей проекта увеличилось на единицу и вместо 1 branches записано 2 branches.

#### Упражнение 4. Добавление третьей ветви проекта Spase

1. Нажмите на пиктограмму 2 branches, перейдите в область создания новой ветви (повторите шаги 2 и 3 предыдущего упражнения). Назовите третью ветвь репозитория ТретьяВетвь (рис.3.18). Укажите, что источником данных для нее является также ветвь main.

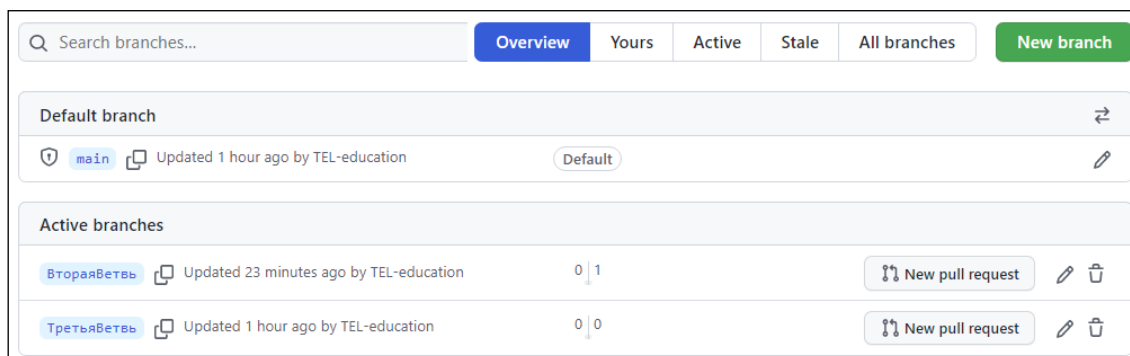


Рисунок 3.18 - Три ветви проекта

2. Перейдите в третью ветвь и создайте в ней текстовый файл. Для этого нажмите на **Add files** и в раскрывшемся списке выберите **Create New file** (рис. 3.17). Добавьте следующий текст: «GitHub – крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Веб-сервис основан на системе контроля версий Git. Ознакомиться с документацией по работе с GitHub можно на сайте GitHub Help».

Назовите файл, например, Назначение СКВ.txt. Заполните комментарий и нажмите на кнопку **Commit new files** (рис.3.19).

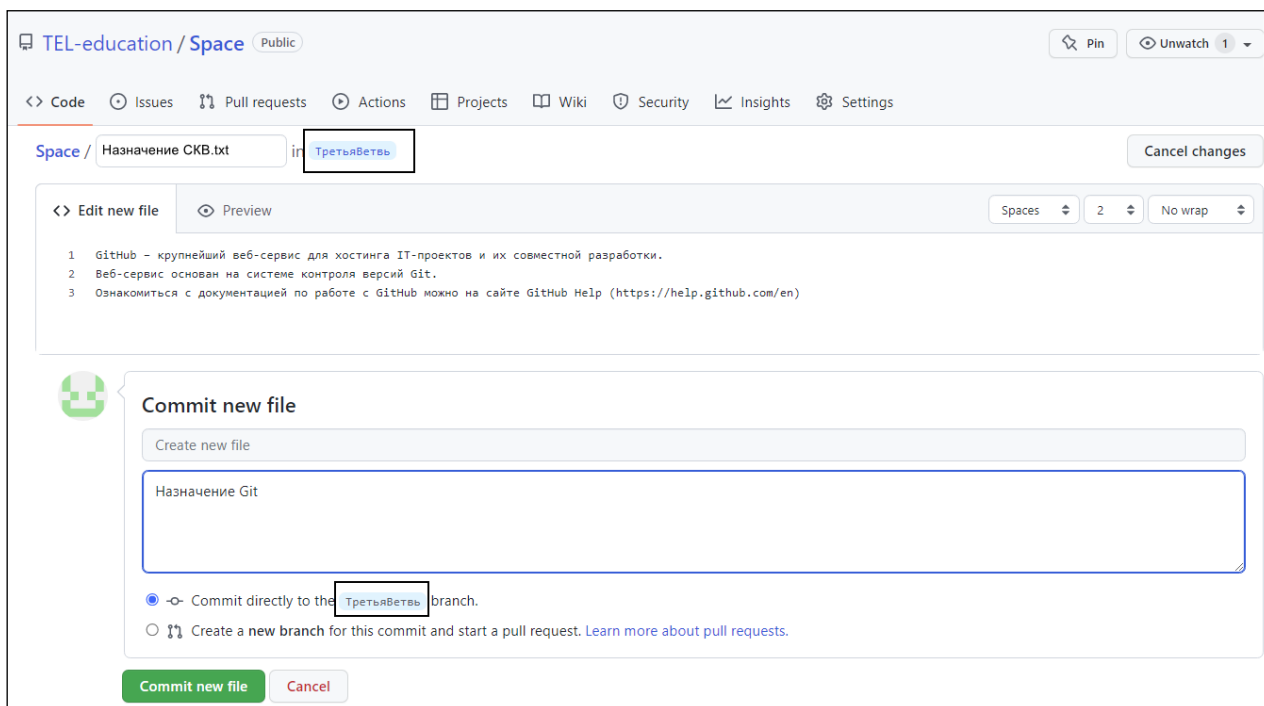


Рисунок 3.19 - Добавление текстового файла в третью ветвь репозитория

Обратите внимание, что добавление файла происходит в ветвь под названием ТретьяВетвь.

3. В репозитории созданы три ветви: main, ВтораяВетвь, ТретьяВетвь (рис.3.20).

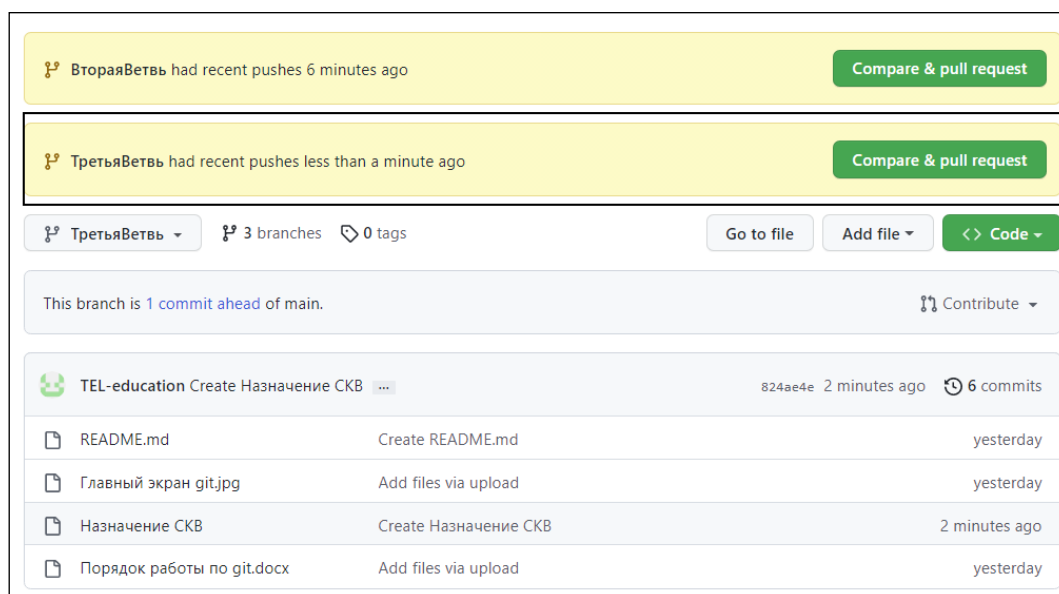


Рисунок 3.20 - Три ветви проекта

В ветви main сохранены исходные файлы проекта, в двух других – аль-

тернативные варианты развития проекта, в которых к исходным файлам добавлены дополнительные. На следующем этапе разработчики могут выбрать, какой из двух вариантов выбрать и, выполнив запрос на слияние, сделать его основным.

## Упражнение 5. Объединение ветвей проекта Space

1. Проведите слияние третьей ветви проекта с основной ветвью. Нажмите на кнопку **Compare & pull request** (рис. 3.20). Pull request (пул-реквест) – это запрос на интеграцию изменений в выбранную ветвь проекта.

2. Проверьте правильность интеграционного потока: из ТретьейВетви в основную main (рис.3.20). Будет проведено автоматическое сравнение данных в двух ветвях и добавление в основную ветвь файлов, которые присутствуют только в ТретьейВетви.

3. Нажмите **Create Pull request** (рис.3.21).

4. Подтвердите запрос на интеграцию нажатием на кнопку **Merge Pull request** (рис.3.22).

5. Кнопка **Merge Pull request** изменила название на **Confirm merge** (рис.3.23).

6. Результат успешного завершения интеграции показан на рис.3.24.

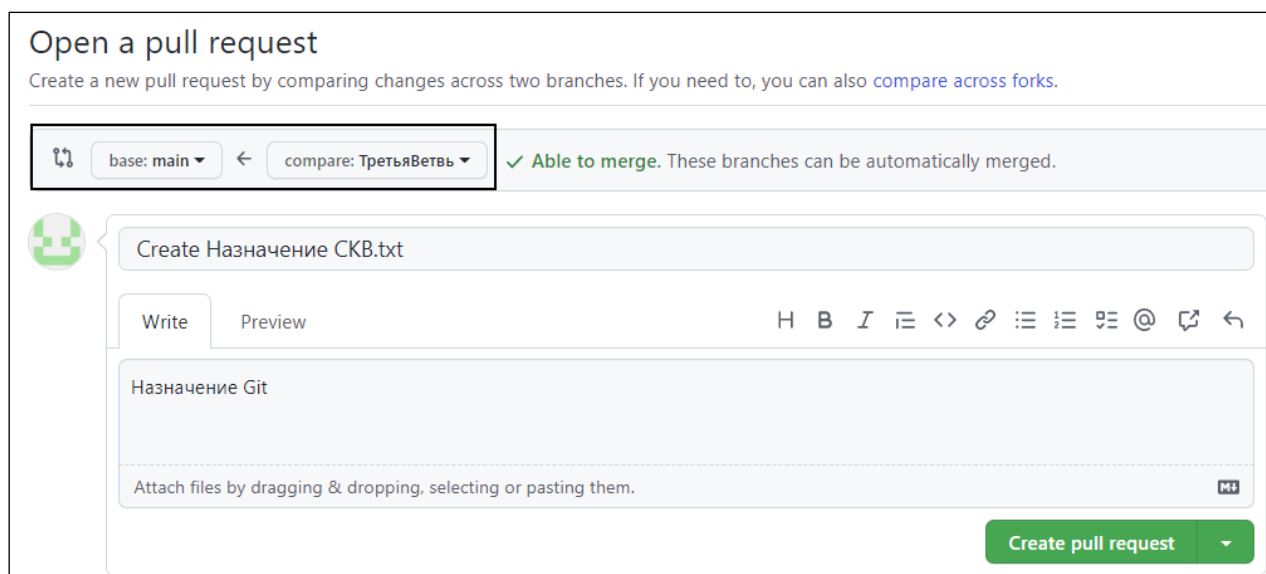


Рисунок 3.21 - Интеграция данных третьей ветви и основной

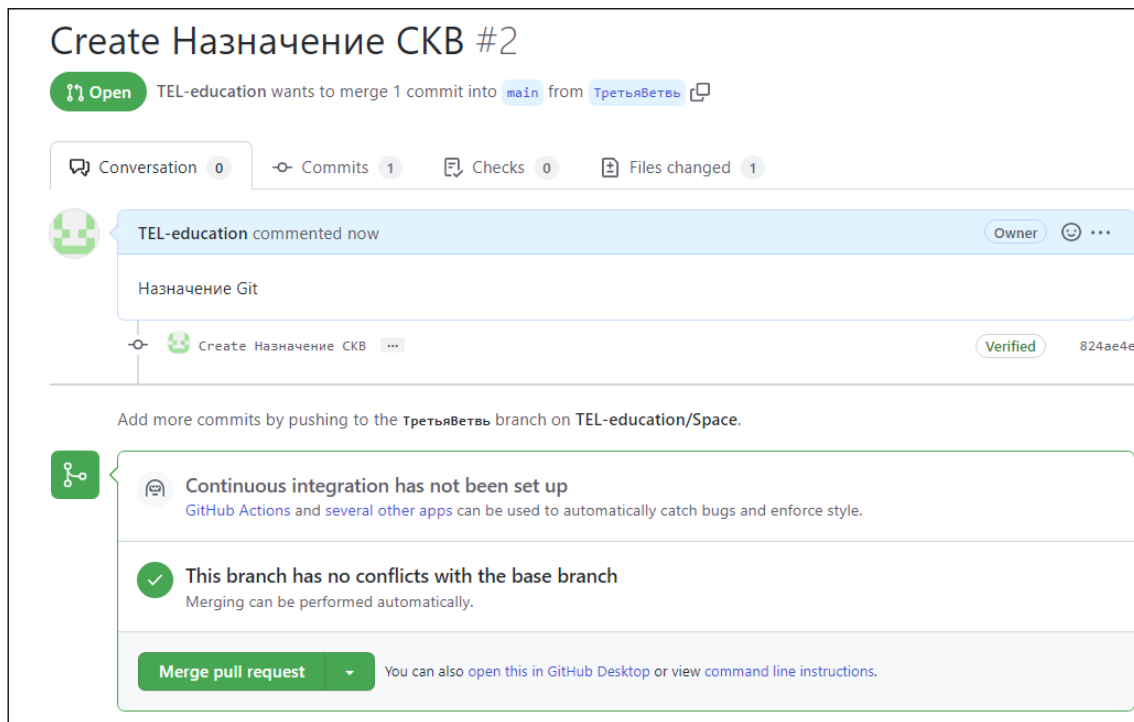


Рис. 3.22. Подтверждение запроса на интеграцию

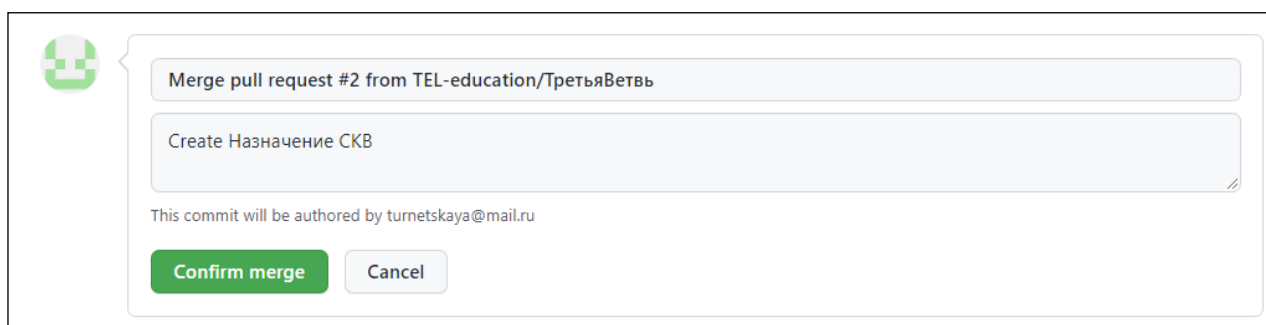


Рисунок 3.23 - Принятие подтверждения нажатием на кнопку Confirm merge

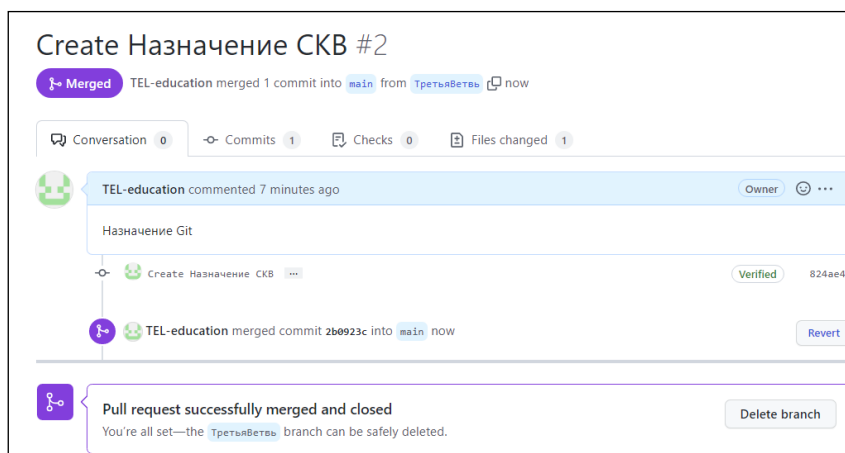


Рисунок 3.24 - Сообщение об успешной интеграции данных в основную ветвь

7. Перейдите в режим просмотра ветвей проекта: нажмите на **Space>3 branches**. В комментариях ТретьейВетви появилась информация о выполненной интеграции (рис. 3.25).



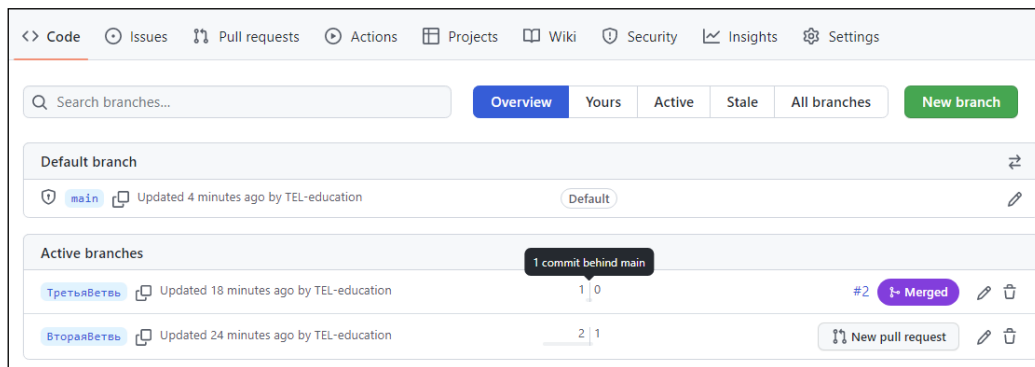


Рисунок 3.25 - Информация о действиях по ветвям репозитория

## Упражнение 6. Форк репозитория

При работе над проектом используют множество библиотек, компонентов программного обеспечения или документацию, которая находится в открытом доступе и хранится на GitHub.

Для сохранения таких репозиториях в своем рабочем пространстве необходимо сделать их клонирование или форк. Для выполнения этой задачи.

1. Находят интересующий пользователя репозиторий, например <https://github.com/electron/electron>.

2. Находясь внутри выбранного для форка репозитория, нажимают Fork. Затем пошагово выполняют действия по клонированию (рис.3.26). На первом шаге предложено создать репозиторий с таким же именем, как оригинальный репозиторий (тот, который требуется форкнуть) (рис.3.27)

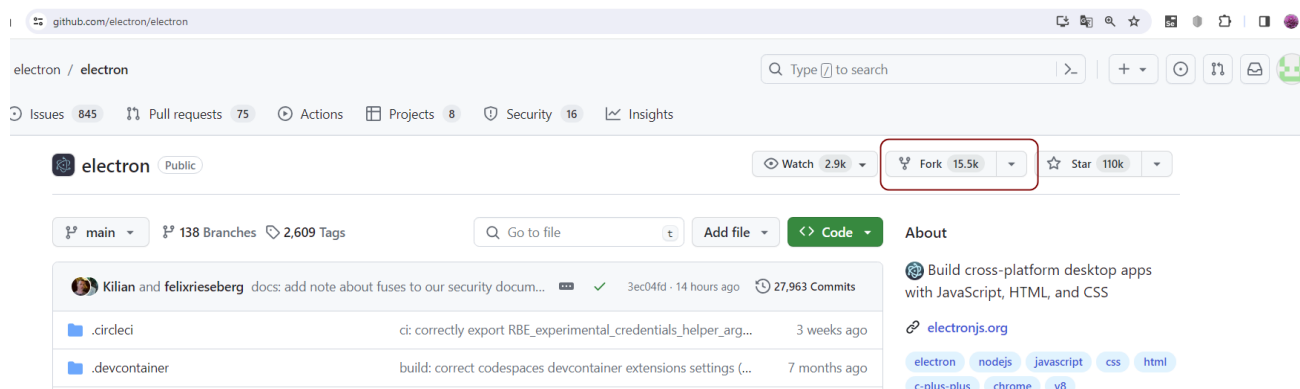


Рисунок 3.26 - Нажатие на Fork

electron / electron

Q Type [ ] to search

Issues 845 Pull requests 75 Actions Projects 8 Security 16 Insights

### Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk (\*).

Owner \*  
TEL-education

Repository name \*  
electron  
electron is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)  
:electron: Build cross-platform desktop apps with JavaScript, HTML, and CSS

☒ Copy the `main` branch only  
Contribute back to electron/electron by adding your own branch. [Learn more.](#)

*i* You are creating a fork in your personal account.

Create fork

Рисунок 3.27 - Создание нового репозитория в рабочем пространстве

Автоматически будет создан репозиторий и осуществлен перенос (рис. 3.28). Обратите внимание, что в результате выполненных шагов, появляется новый репозиторий в вашей рабочей области.

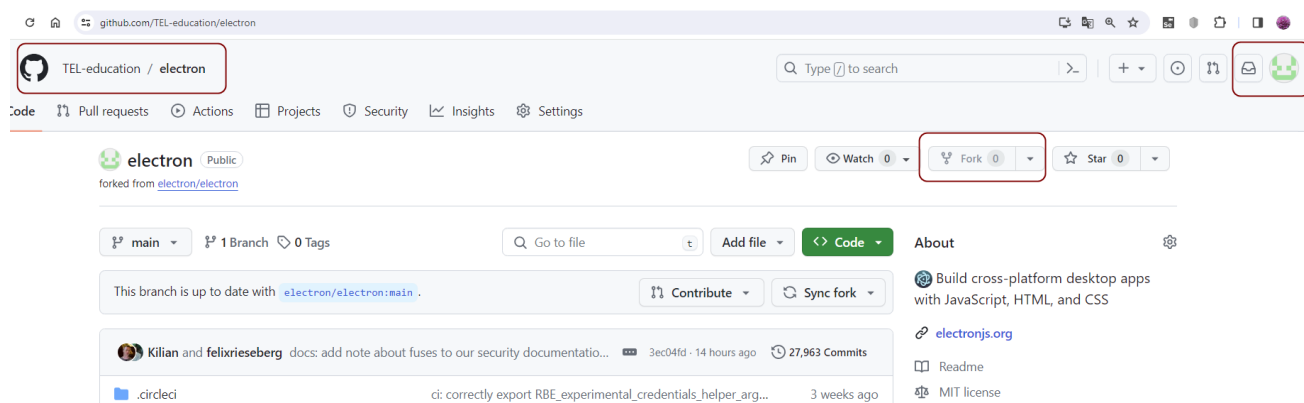


Рисунок 3.28 - Форк репозитория в рабочем пространстве пользователя

## 7. СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. 100 самых ценных репозиториях GitHub URL: <https://habr.com/ru/articles/453444/> (дата посещения 03.02.2024)
2. Программная инженерия. Интеграционный подход к разработке / Е. Л. Турнецкая, А. В. Аграновский. — Санкт-Петербург: Лань, 2023. — 216 с. — ISBN 978-5-507-46898-0. — Текст : электронный // Лань: электронно-библиотечная система. — (access : <https://e.lanbook.com/book/352307>).
3. Отчет GitHub за 2023 г. URL: <https://github.blog/2023-11-08-the-state-of-open-source-and-ai/> (дата посещения 03.02.2024)