# Общие рекомендации

Онлайн-сервис GitHub позволяет создавать репозитории проектов без глу- боких знаний программирования. Все требуемые команды git будут сгенериро- ваны автоматически на основе действий пользователя. В репозитории проекта можно хранить любые файлы. Текстовые файлы можно создавать непосредст- венно в репозитории с помощью языка Markdown. Там же можно их изменять. Если возможностей тектсового 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 простран- ство полностью синхронизировано с онлайн-пространством GitHub. При внесе- нии изменений в файлы они будут автоматически переданы в облачную версию.

В практической работе предусмотрен вариант использования только облач- ной 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 не- сколько 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 соз- дали репозиторий, заполнили его и научились создавать ветви. Также можете форкнуть любой выбранный репозиторий. Выбор репозитория, который плани- руете перенести осуществите самостоятельно, например используя список са- мых интересных репозиториев [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 дается, 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 репо- зитория, который выложен в курсе.

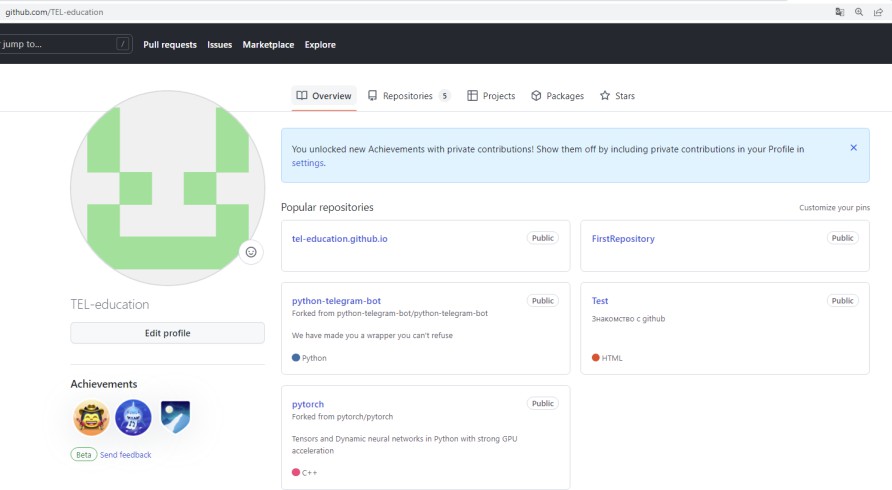
# Последовательность выполнения задания

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 язы- ке программирования, или текстовые файлы, например, отчеты по ла- бораторным работам, графические материалы и т.д.
3. Каждое изменение (новую ветвь, дополнение репозитория) фиксиро- вать в системе контроле версий.
4. Скопируйте любой программный проект, который хранится на GitHub, например, библиотеку pytorch, в свой удаленный репозиторий.
5. Выполните отчет в соответствие с требованиями к структуре и напол- нению.

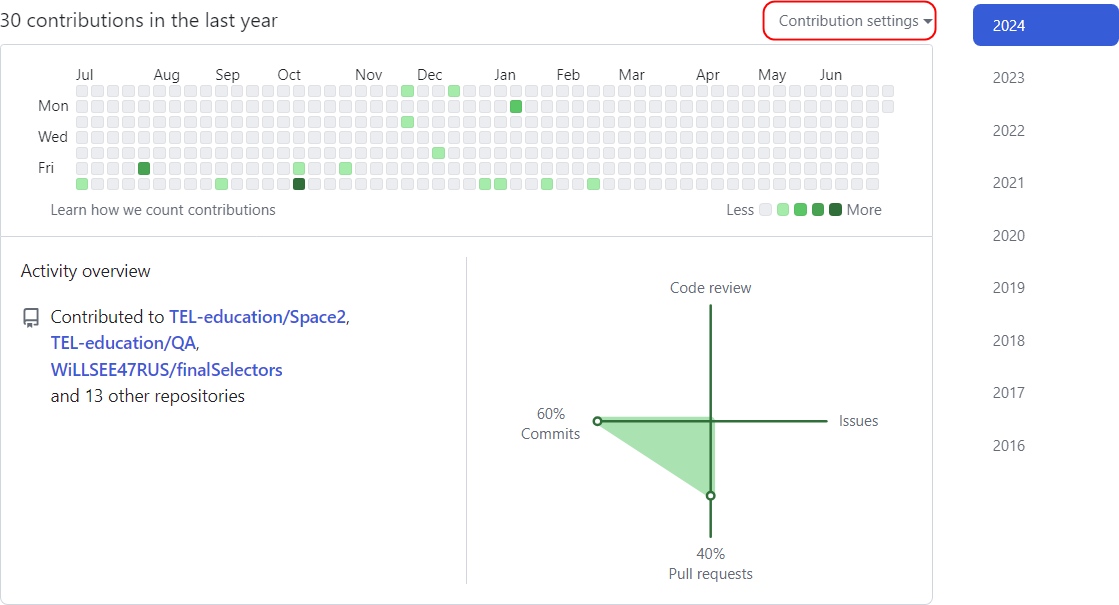
Примеры выполнения пунктов задания приведены в [2], а также в много- численных интернет-источниках.

# 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).

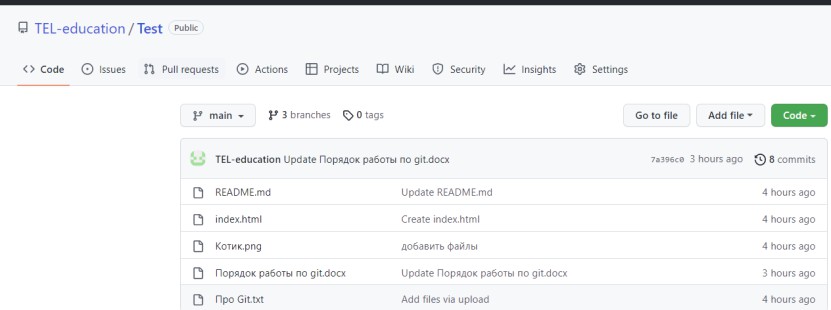


*Рисунок 1.1 - Пример аккаунта*

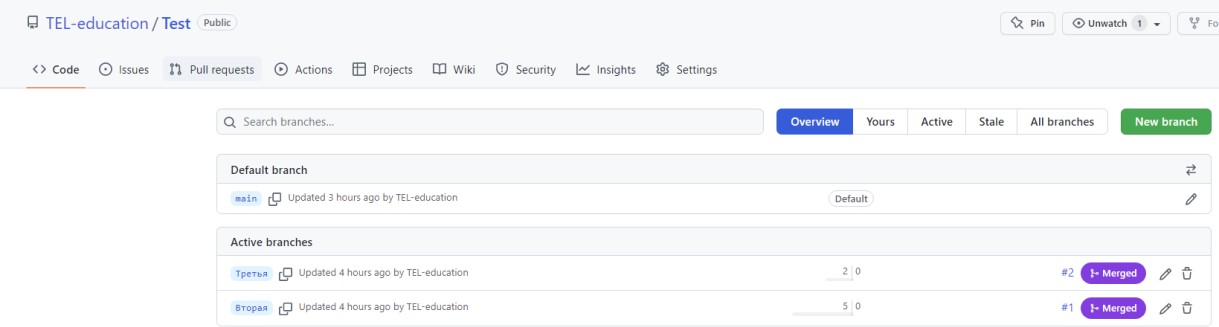
**

*Рисунок 1.2 - Активность аккаунта*

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

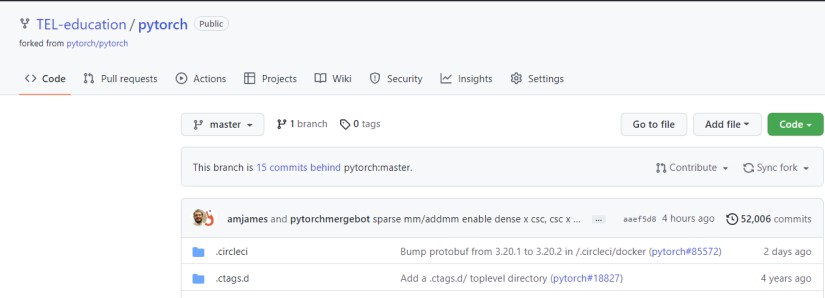


*Рисунок 1.3 - Ветвь main*



*Рисунок 1.4 - Ветви проекта*

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



*Рисунок 1.5 - Пример скриншота копии библиотеки pytorch*

1. Ссылка на удаленный репозиторий на GitHub, реализованный в ходе выполнения практической работы.
2. 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.
3. Список использованных источников.
4. **НАЗНАЧЕНИЕ СИСТЕМ УПРАВЛЕНИЯ ВЕРСИЯМИ**

При работе над программным проектом каждый из команды разработчи- ков должен иметь доступ ко всем файлам. Поэтому возникает необходимость организации виртуального хранилища программных компонент с системой кон- троля 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: бло- кирующие, использующие слияние до фиксации и выполняющие фиксацию до слияния;
  + системы управления версиями могут выполнять небольшие опера- ции или операции с файлами.

Работу с системами контроля версий рассмотрим на примере распределен- ной системы управления версиями Git1.

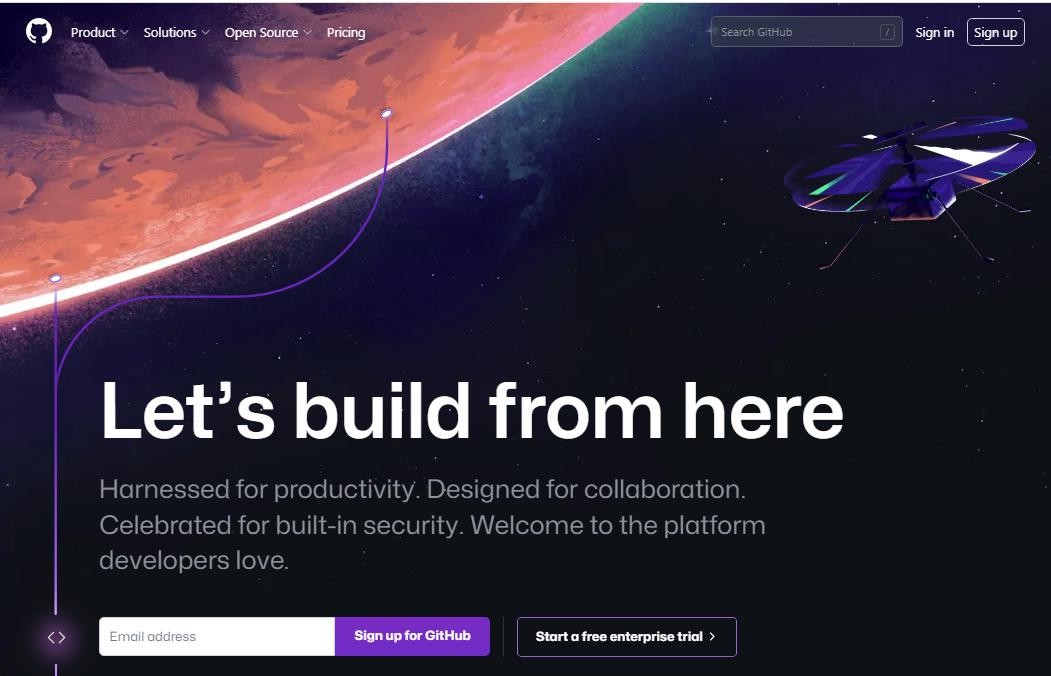
Программные средства данной системы позволяют сохранить изменения в файл или набор файлов в процессе их модификации и при необходимости вер- нуться к конкретной версии файла. Если над проектом работают несколько че- ловек, то каждому из них обеспечивают доступ для совместной работы над фай-

1 https://git-scm.com/

лом. Каждое внесенное изменение фиксируют, поэтому возникает многоверси- онность разрабатываемого программного продукта. Недостатком Git можно счи- тать то, что он имеет интерфейс командной строки. Чтобы упростить взаимо- действие разработчиков с этой системой, был создан графический интерфейс GitHub2.

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) .

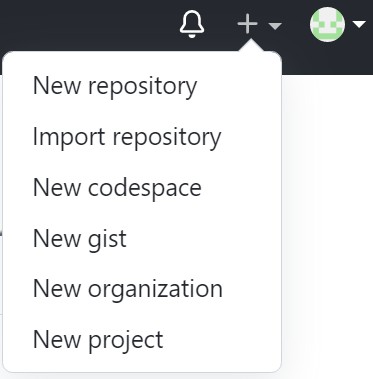
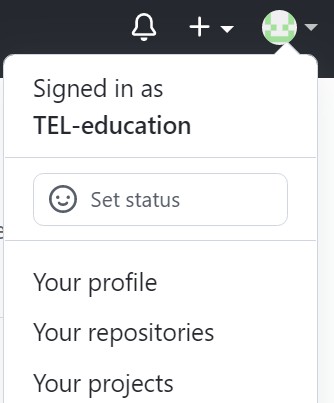
# Упражнение 1. Создание аккаунта GitHub.

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*

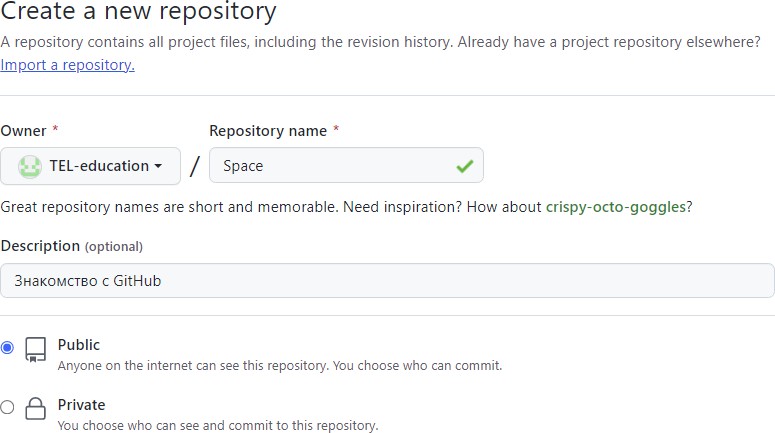
1. Полное имя аккаунта посмотрите на вкладке Профиль, например TEL-education.github.io (рис. 3.2).

2 https://github.com



|  |  |
| --- | --- |
| *Рисунок 3.2 - Проверка имени аккаунта* | *Рисунок 3.3 - Создание нового репозитория*  *New repository* |

1. 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 рас- смотренных ниже примерах приводятся на русском языке, чтобы упростить по- нимание процесса взаимодействия с удаленным репозиторием. В описание репо- зитория внесите: «Знакомство с GitHub».

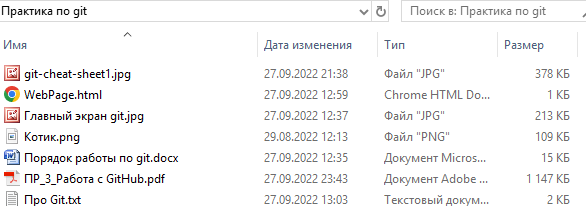


*Рисунок 3.4 - Создание репозитория на сервисе GitHub*

# Упражнение 2. Копирование файлов проекта в репозиторий на GitHub

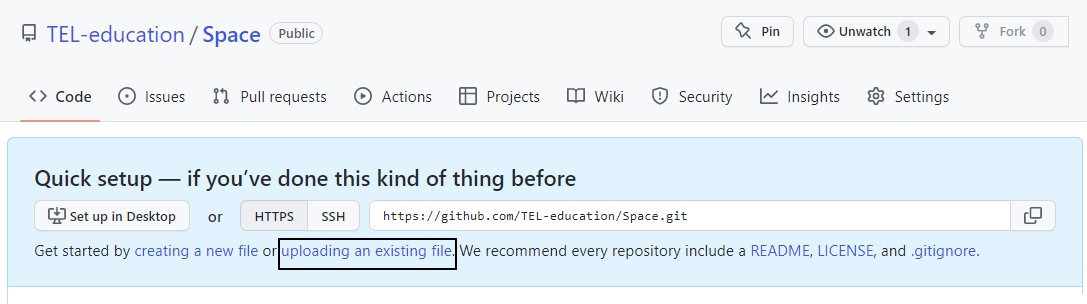
* 1. Подготовьте на персональном компьютере не менее пяти файлов для

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



*Рисунок 3.5 - Подготовленные для копирования файлы*

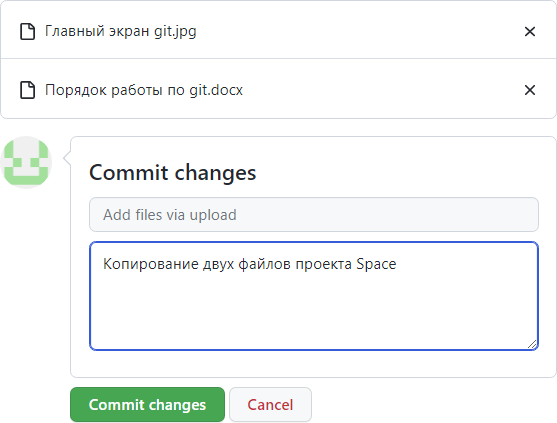
* 1. После создания репозитория будет доступно окно для выбора дальне- ших действий пользователя (рис.3.6). Выберите uploading an existing file (загрузку внешних файлов).



*Рисунок 3.6 - Выбор опции uploading an existing*

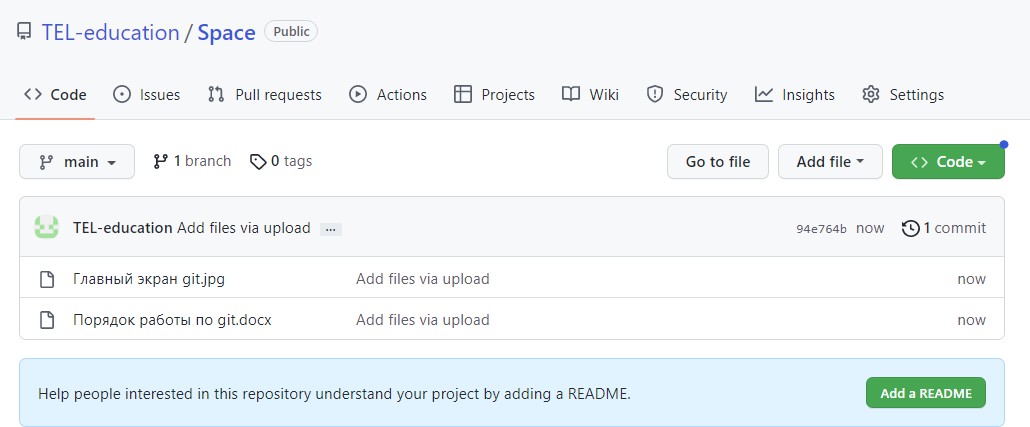
* 1. Затем перетяните в открывшееся окно два любых файла для копирова- ния в основную ветвь проекта. Основная ветвь проекта по умолчанию названа main. Обратите внимание, на автоматически сгенерированную команду Add files via upload, которая появилась в разделе *Commit changes* (рис.3.7).

Добавьте описание, объясняющее, какое именно изменение было сделано. В рассматриваемом примере: «Копирование двух файлов проекта Space». Опи- сания 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*

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



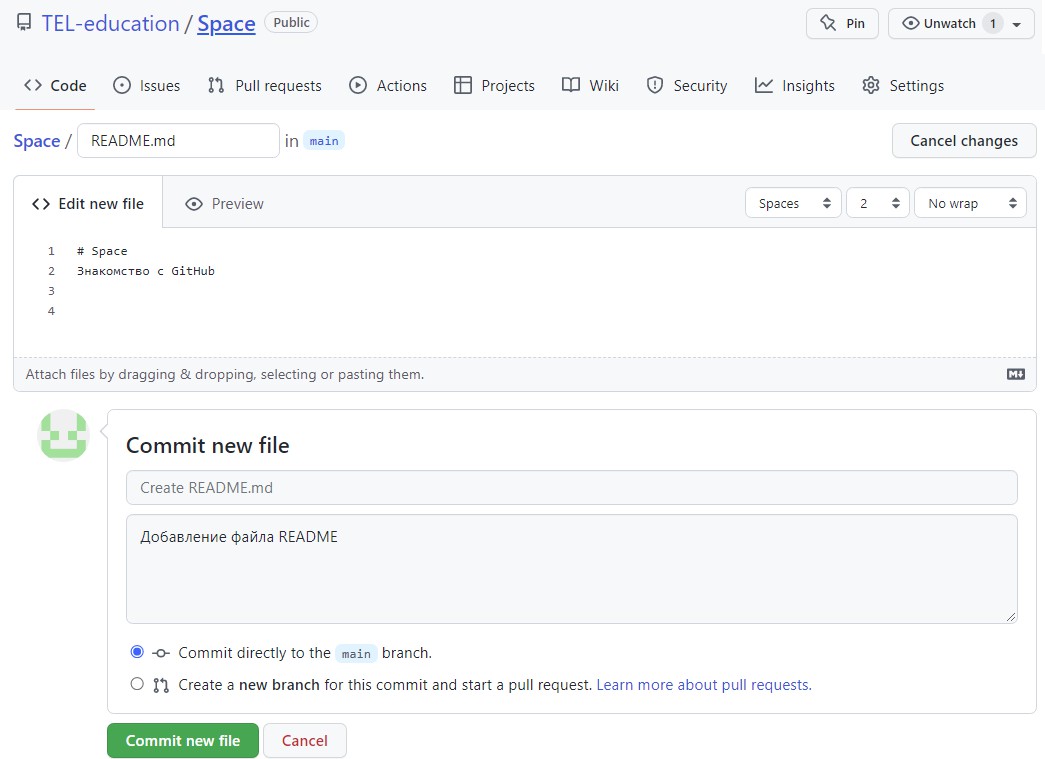
*Рисунок 3.8 - Ветви проекта Обучение в репозитории TEL-education*

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

1. В каждом проекте предусмотрено его описание в файле README.md. Нажмите на кнопку **Add a README** для перехода в область для создания фай- ла.
2. В текстовую область вкладки меню *Edit new file* (рис.3.9) уже внесено описание репозитория: «Знакомство с GitHub». В области Commit new file обра- тите внимание на команду Create README.md. Внесите описание коммита:

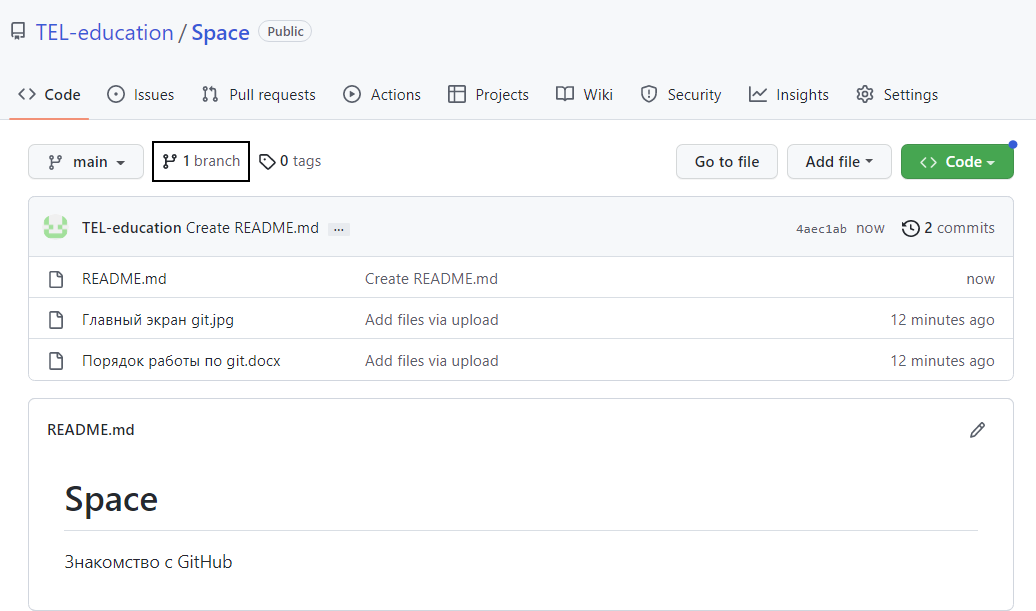
«Добавление файла README».

1. 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.*

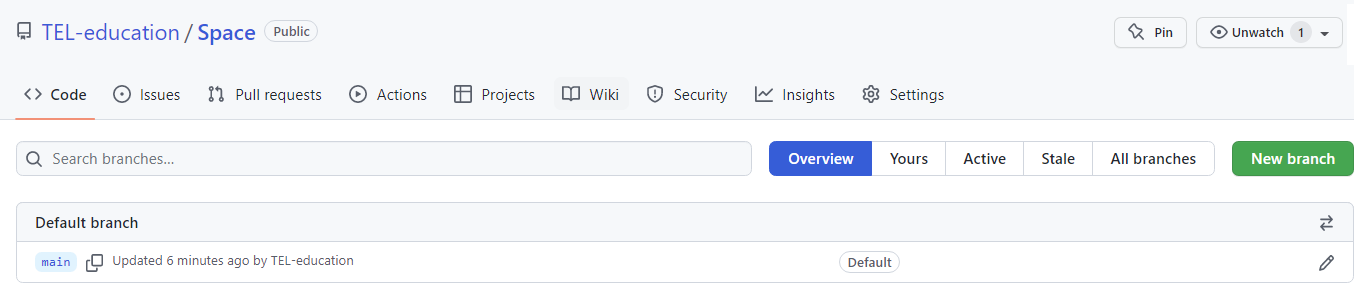
После этого шага в репозитории Space сохранены три файла (рис.3.10).



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

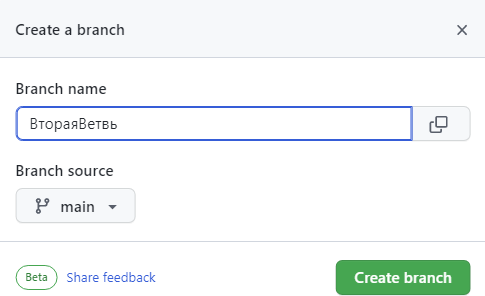
# Упражнение 3. Добавление второй ветви проекта Space

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

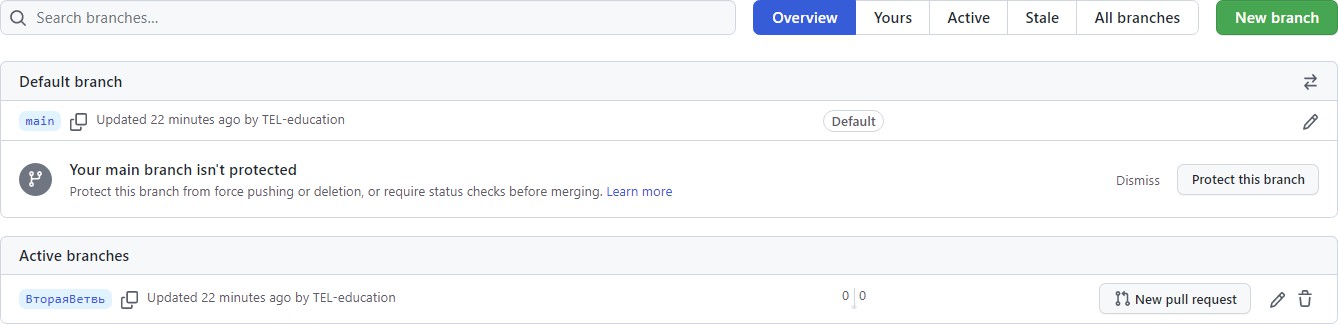


*Рисунок 3.11 - Начало создания новой ветви в репозитории Space*

1. Дайте новой ветви понятное и лаконичное название, например, Вто- раяВетвь. 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.



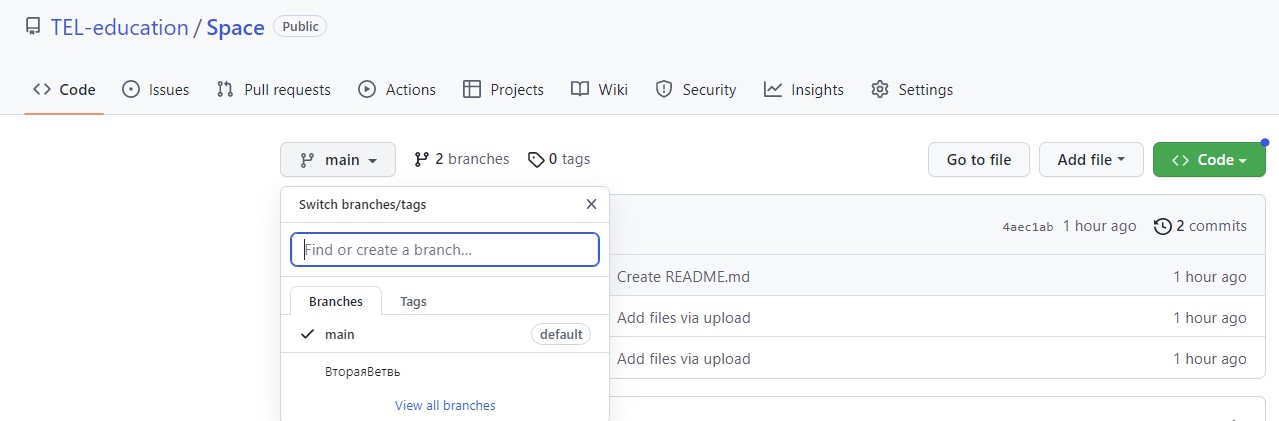
*Рисунок 3.12 - Создание второй ветви в репозитории Space*



*Рисунок 3.13 - Две ветви репозитория Space*

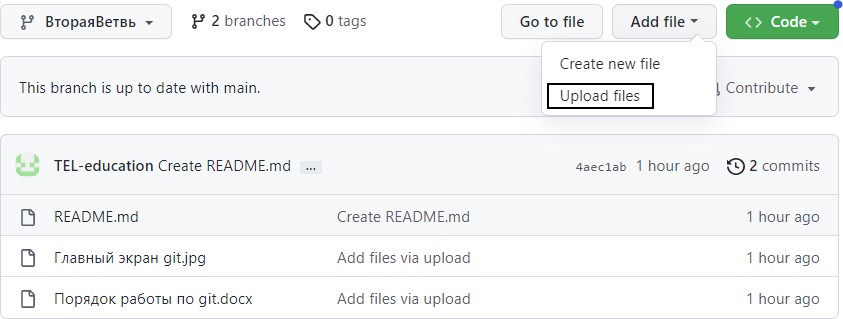
При создании второй ветви GitHub потребует уточнений в политике безо- пасного слияния ветвей проекта. Нажмите **Protect this branche**. Будет произве- ден переход на новую страницу с политиками GitHub. В учебном проекте не ак- тивуйте чек-боксы, оставьте все настройки без изменений и примите соглашение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

1. Нажмите на название репозитория Space. Затем выберите из раскры- вающегося меню название ВтораяВетвь (рис.3.14).



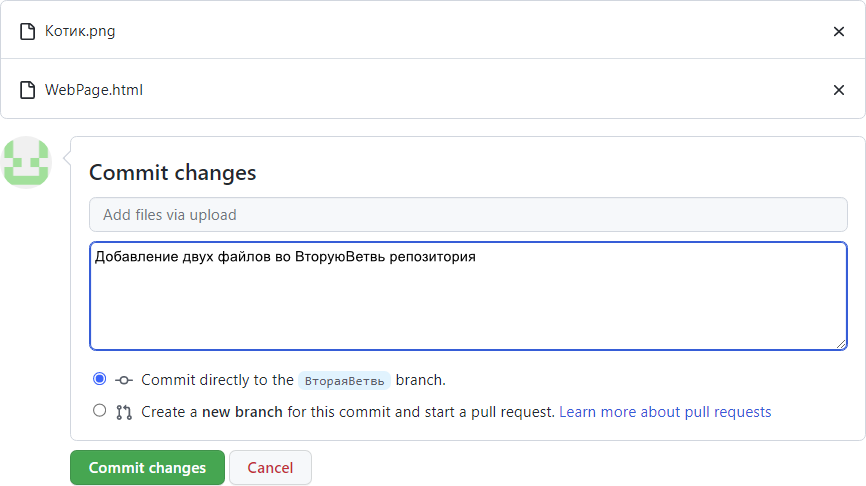
*Рисунок 3.14 - Переход к данным, расположенным во второй ветви*

1. Добавьте еще два файла во вторую ветвь репозитория. Дл яэтого нажми- те на **Add files** и выберите из раскрывающегося списка **Upload files** (рис.3.15).



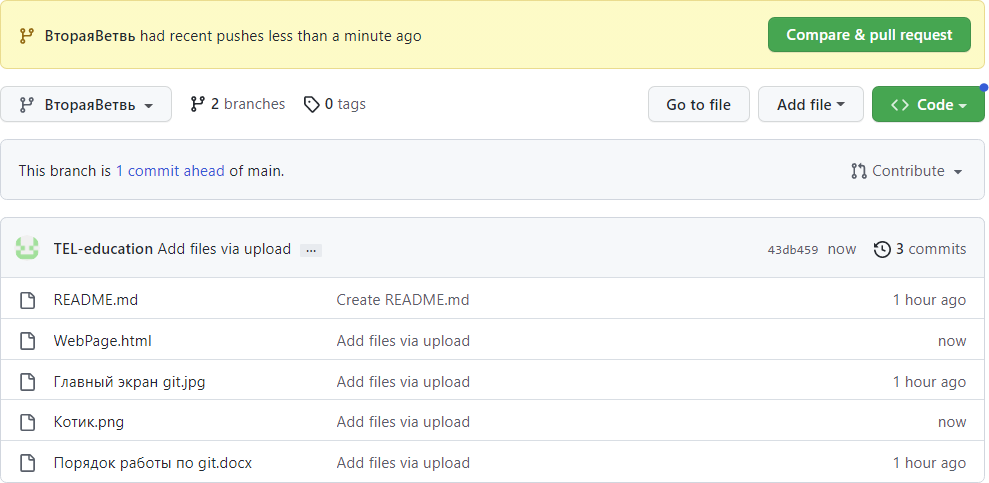
*Рисунок 3.15 - Выбор варианта загрузки файлов*

1. Скопируйте два файла из папки, подготовленной на персональном ком- пьютере. Добавьте описание коммита и нажмите **Commit change** (рис. 3.16).



*Рисунок 3.16- Добавление файлов во вторую ветвь*

1. Нажмите на название репозитория Space и посмотрите на результат из- менения количества файлов (рис.3.17).

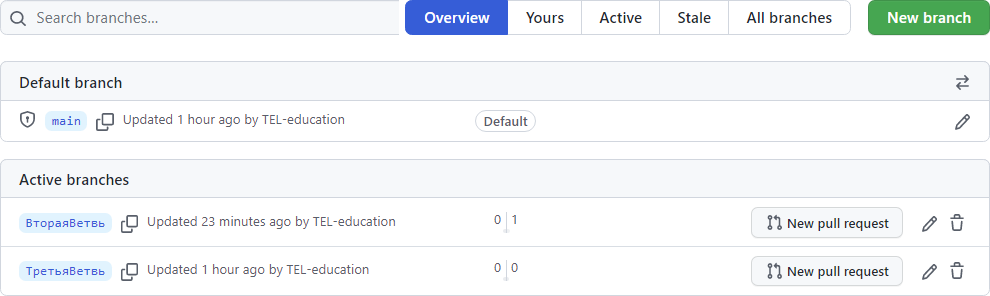


*Рисунок 3.17 - Просмотр новых данных во второй ветви*

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

# Упражнение 4. Добавление третьей ветви проекта Space

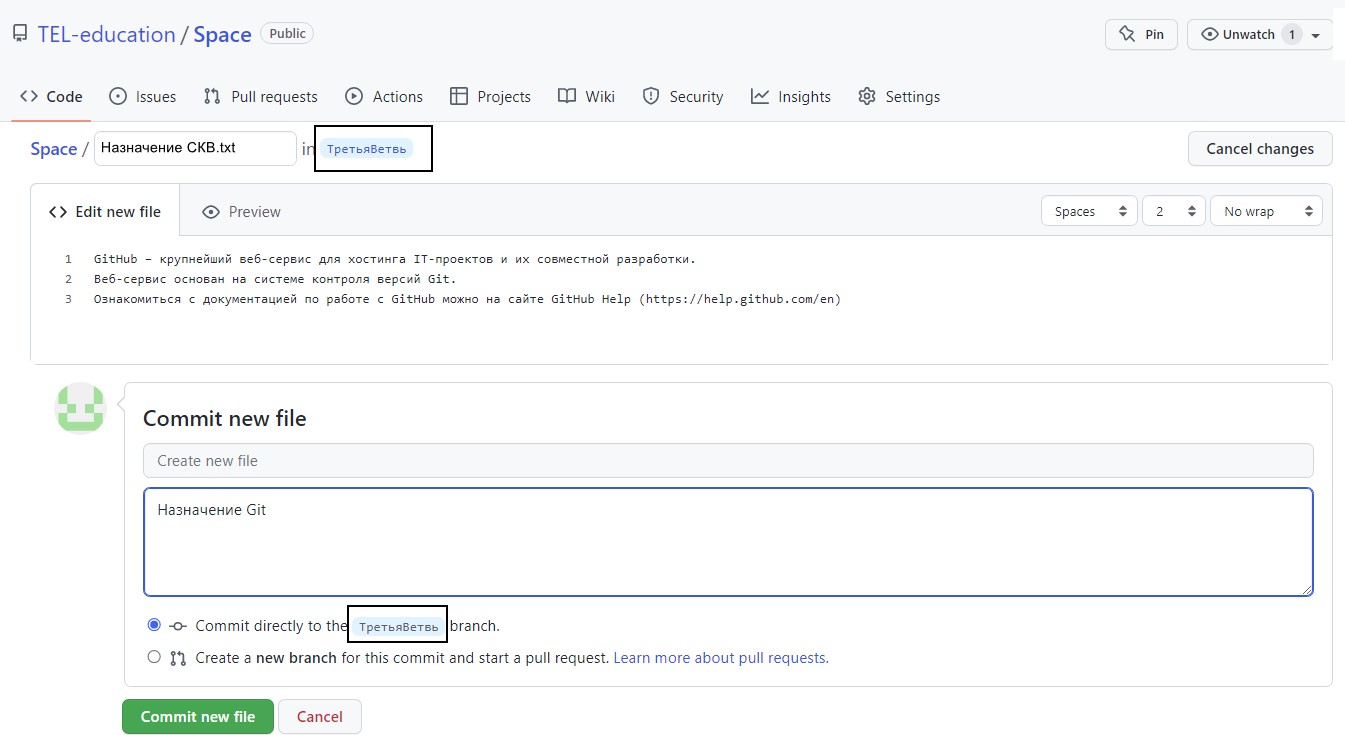
1. Нажмите на пиктограмму 2 branches, перейдите в область создания новой ветви (повторите шаги 2 и 3 предыдщего упражения). Назовите третью ветвь репозитория ТретьяВетвь (рис.3.18). Укажите, что источником данных для нее является также ветвь main.



*Рисунок 3.18 - Три ветви проекта*

1. Перейдите в третью ветвь и создайте в ней текстовый файл. Для этого нажмите на **Add files** и в ракрывшемся 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 сис- теме контроля версий Git. Ознакомиться с документацией по работе с GitHub можно на сайте GitHub Help».

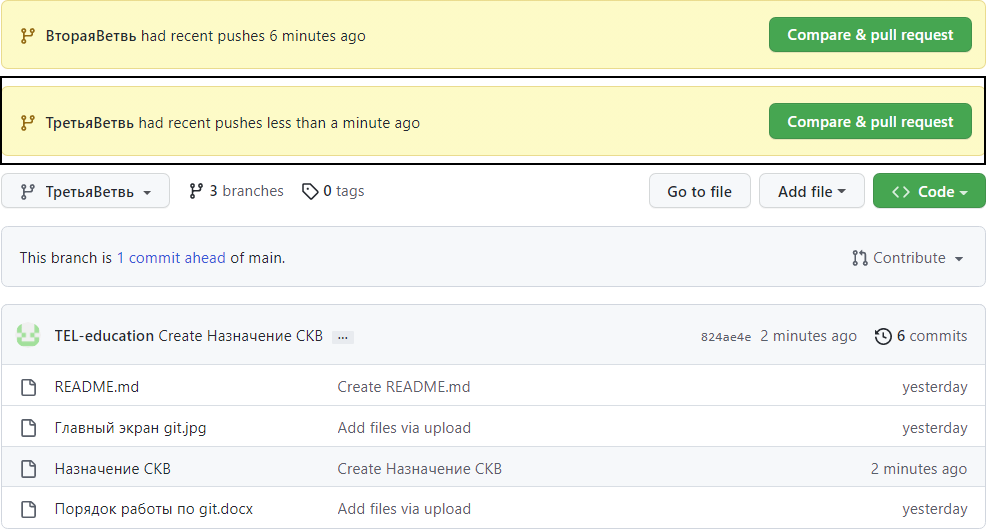
Назовите файл, например, Назначение СКВ.txt. Заполните коммента- рий и нажмите на кнопку **Commit new files** (рис.3.19).



*Рисунок 3.19 - Добавление текстового файла в третью ветвь репозитория*

Обратите внимание, что добавление файла происходит в ветвь под назва- нием ТретьяВетвь.

1. В репозитории созданы три ветви: main, ВтораяВетвь, Третья- Ветвь (рис.3.20).



*Рисунок 3.20 - Три ветви проекта*

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

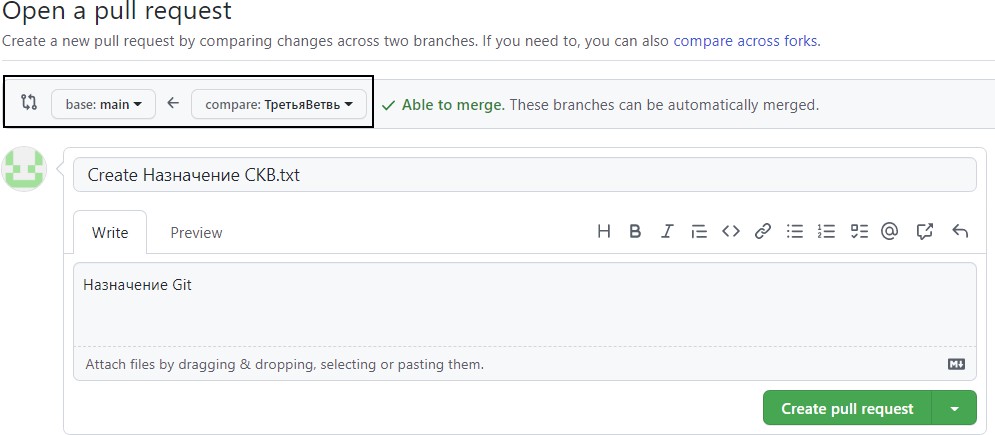
тернативные варианты развития проекта, в которых к исходным файлам добав- лены дополнительные. На следующем этапе разработчики могут выбрать, какой из двух вариантов выбрать и, выполнив запрос на слияние, сделать его основ- ным.

# Упражнение 5. Объединение ветвей проекта Space

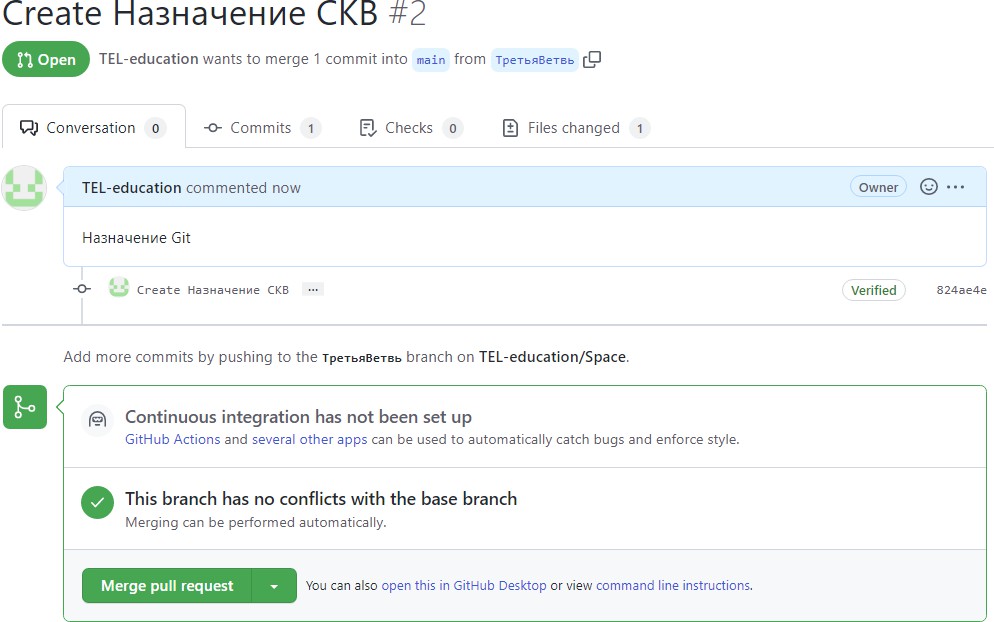
1. Проведите слияние третьей ветви проекта с основной ветвью. Нажмите на кнопку **Compare & pull request** (рис. 3.20). Pull request (пул-реквест) – это за- прос на интеграцию изменений в выбранную ветвь проекта.
2. Проверьте правильность интеграционного потока: из ТретьейВетви в основную main (рис.3.20). 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 re- quest** (рис.3.22).
5. Кнопка **Merge Pull request** изменила название на **Confirm merge**

(рис.3.23).

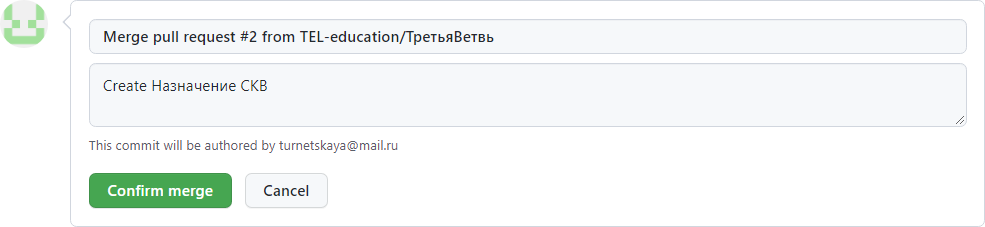
1. Результат успешного завершения интеграции показан на рис.3.24.



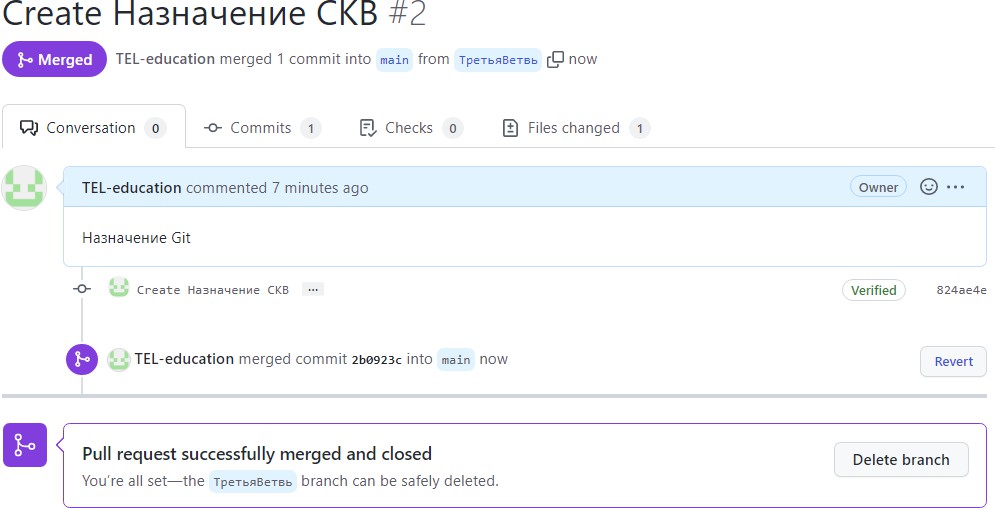
*Рисунок 3.21 - Интеграция данных третьей ветви и основной*



*Рис. 3.22. Подтверждение запроса на интеграцию*

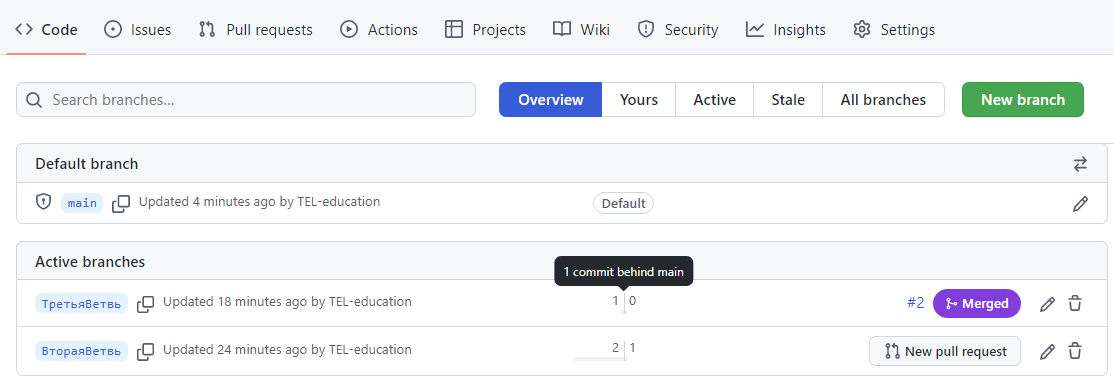


*Рисунок 3.23 - Принятие подтверждения нажатием на кнопку Confirm merge*



*Рисунок 3.24 - Сообщение об успешной интеграции данных в основную ветвь*

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

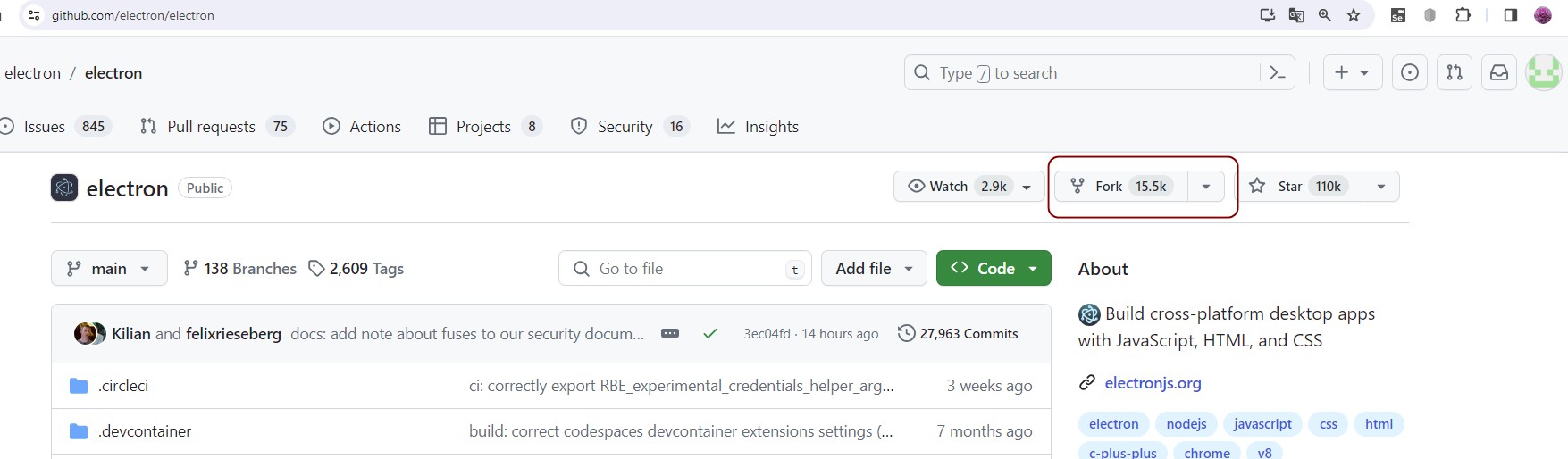


*Рисунок 3.25 - Информация о действиях по ветвям репозитория*

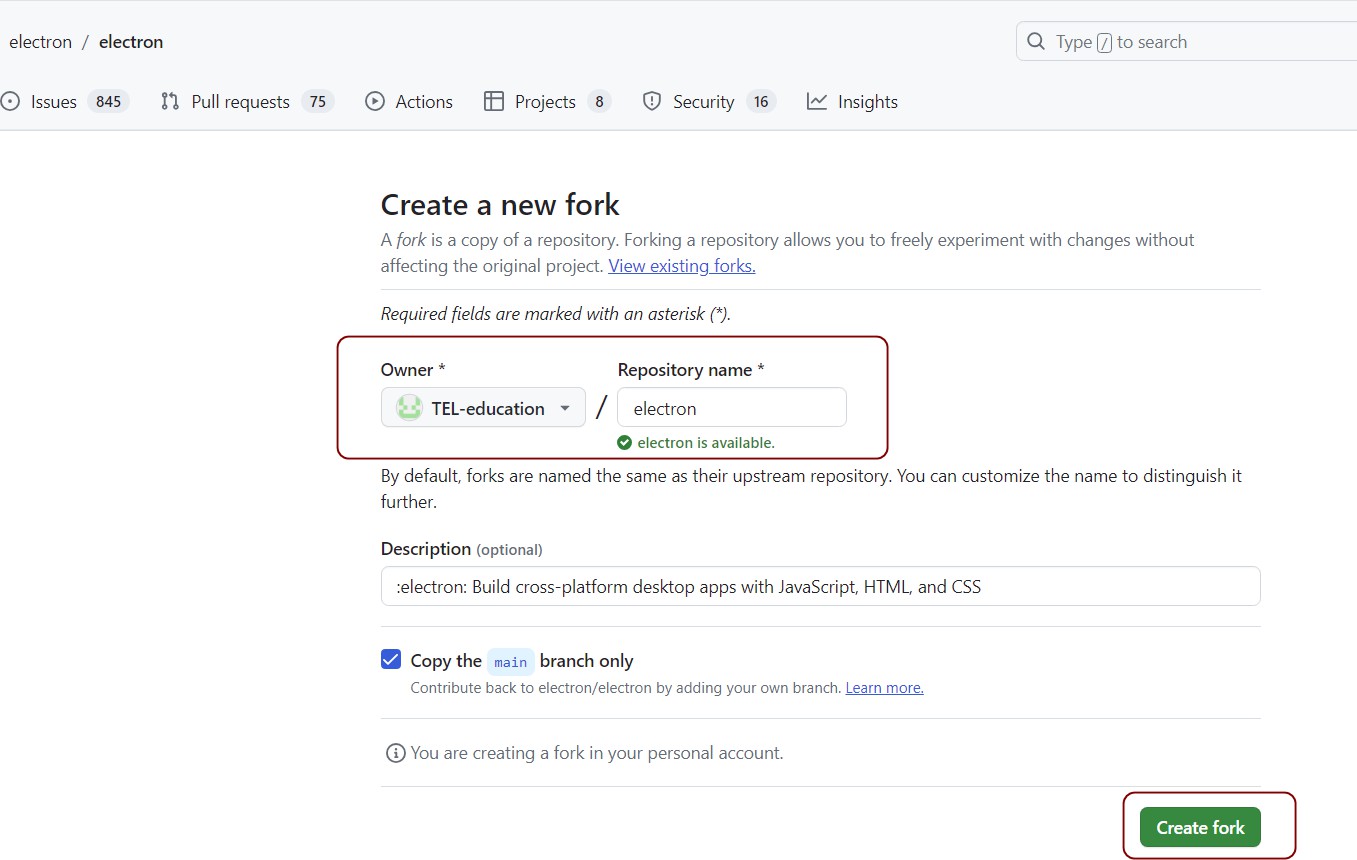
# Упражнение 6. Форк репозитория

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

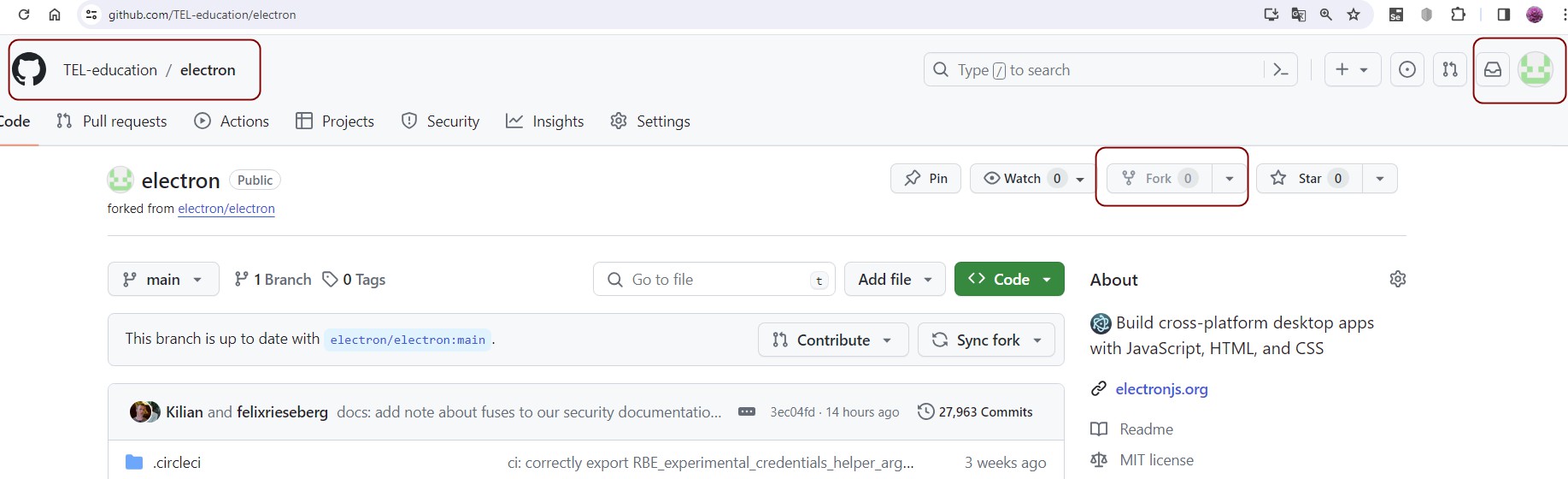
Для сохранения таких репозиториев в своем рабочем пространстве необ- ходимо сделать их клонирование или форк. Для выполнения этой задачи.

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*



*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 но- вый репозиторий в вашей рабочей области.

*Рисунок 3.28 - Форк репозитория в рабочем пространстве пользователя*

**7. СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. 100 cамых ценных репозиториев GitHub URL: <https://habr.com/ru/articles/453444/>(дата посещения 03.02.2024)
2. Программная инженерия. Интеграционный подход к разработке / Е. Л. Тур- нецкая, 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-](https://github.blog/2023-11-08-the-state-of-open-source-and-ai/) [source-and-ai/](https://github.blog/2023-11-08-the-state-of-open-source-and-ai/) (дата посещения 03.02.2024)