

Лабораторная работа № 3

Работа в системе управления версиями Git

Цель работы: познакомиться с распределенной системой управления версиями Git и сервисом GitHub.

Методические указания.

Git – распределенной системой управления версиями, программные средства которой 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 файлом. Каждое внесенное изменение фиксируется, 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 средствами GitHub.

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 (<https://help.github.com/en>) или ресурсе HTMLAcademy (<https://htmlacademy.ru/blog/useful/git/register-on-github-work-with-github-desktop>).

Порядок работы с GitHub.

1. Создание аккаунта на сервисе GitHub.

1.1. Зарегистрируйтесь на сервисе GitHub (<https://github.com/>).

Для этого перейдите на сайт и нажмите кнопку Sign up (рис.1). В открывшемся меню введите параметры учетной записи: имя пользователя; адрес электронной почты; пароль.

Рис. 1. Регистрация на сервисе GitHub

1.2. Полное имя аккаунта посмотрите на вкладке Профиля, например kafedra41.github.io

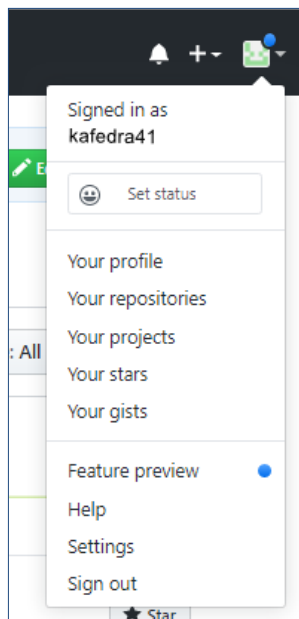


Рис.2. Проверка имени аккаунта

1.3. После регистрации создайте свой первый репозиторий. Задайте имя и сделайте публичным (рис.3). При платном доступе к сервису репозиторий можно сделать приватным.

Рис. 3. Создание репозитория на сервисе GitHub

1.4. После завершения регистрации откроется страница профиля и на электронную почту, указанную при регистрации, придет письмо с просьбой подтвердить электронный адрес. Для завершения регистрации пройдите по полученной ссылке.

1.5. Посмотрите полное название созданного репозитория

2. Скачивание и установка GitHub Desktop.

2.1. Перейдите на сайт <https://desktop.github.com/>. На главной странице сайта скачайте последнюю актуальную версию для своей операционной системы (рис. 4).

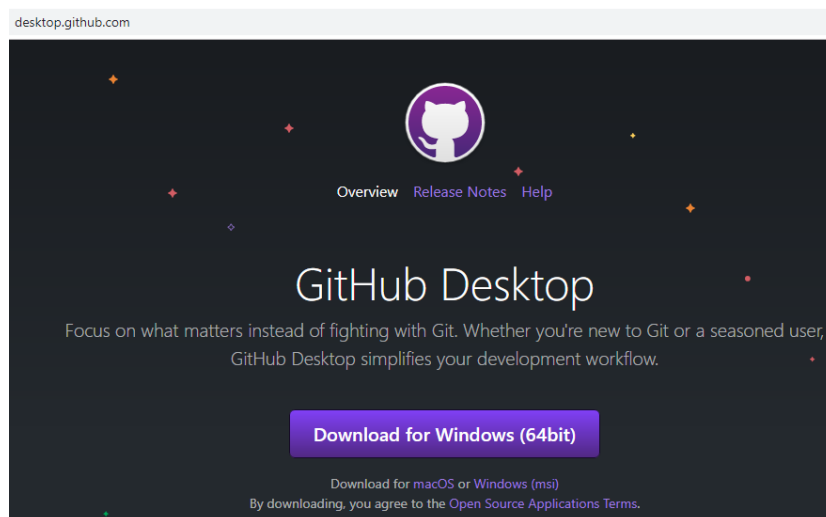


Рис. 4. Выбор версии программного средства GitHub Desktop

2. После активации приложения пройдите процедуру авторизации в GitHub Desktop (рис.5). Для этого введите пароль и логин от аккаунта на сервисе GitHub. Обратите внимание, что имя и почта пользователя должна совпадать с аналогичными данными на сервисе.

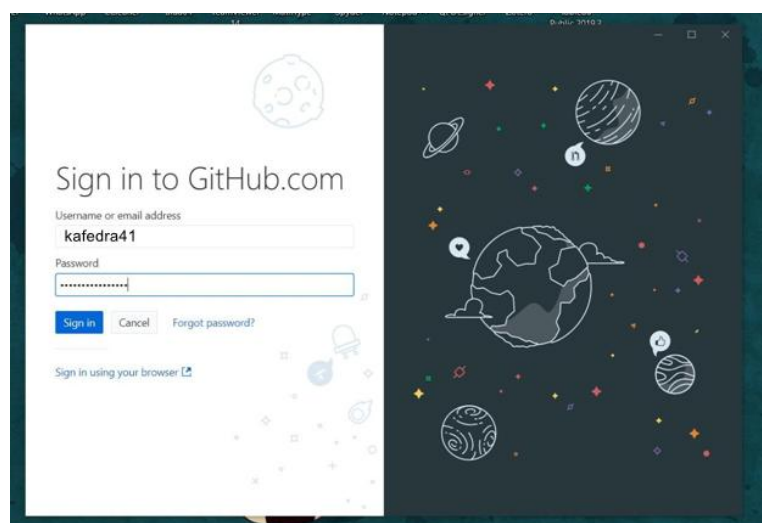


Рис. 5. Авторизация в GitHub Desktop

3. Создание первого файла проекта на сервисе GitHub.

3.1. На сервисе GitHub войдите в свой репозиторий, например FirstRepository, и задайте название главной ветки проекта, например, master (<https://guides.github.com/activities/hello-world/>).

3.2. Наберите или скопируйте ранее созданный программный код в редактируемый файл master.

3.3. Просмотрите содержимое на вкладке Preview и сохраните файл, нажав на кнопку Commit new file (рис.6).

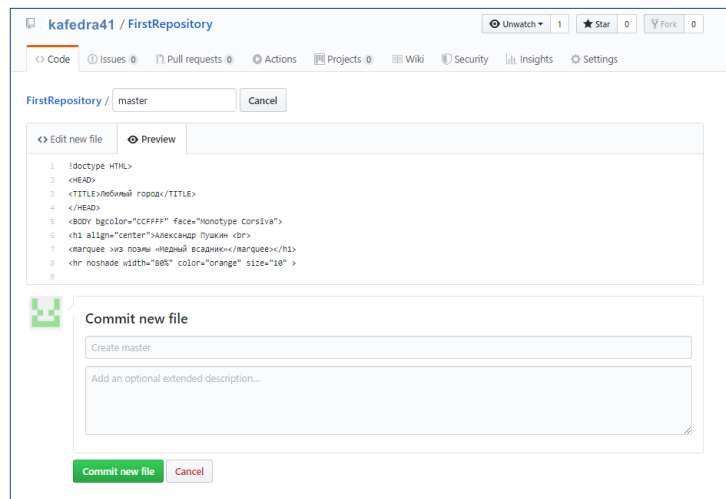


Рис. 6. Создание файла master проекта FirstRepository

3.4. В том же репозитории создайте файл README.md с описанием своего проекта (рис.7).

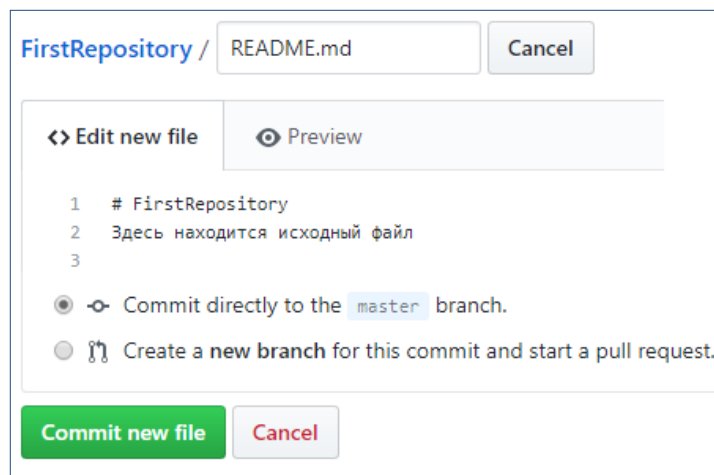


Рис. 7. Создание файла README.md проекта FirstRepository

Таким образом, в исходной ветке проекта master находится два файла: README.md и master (рис. 8) и master.

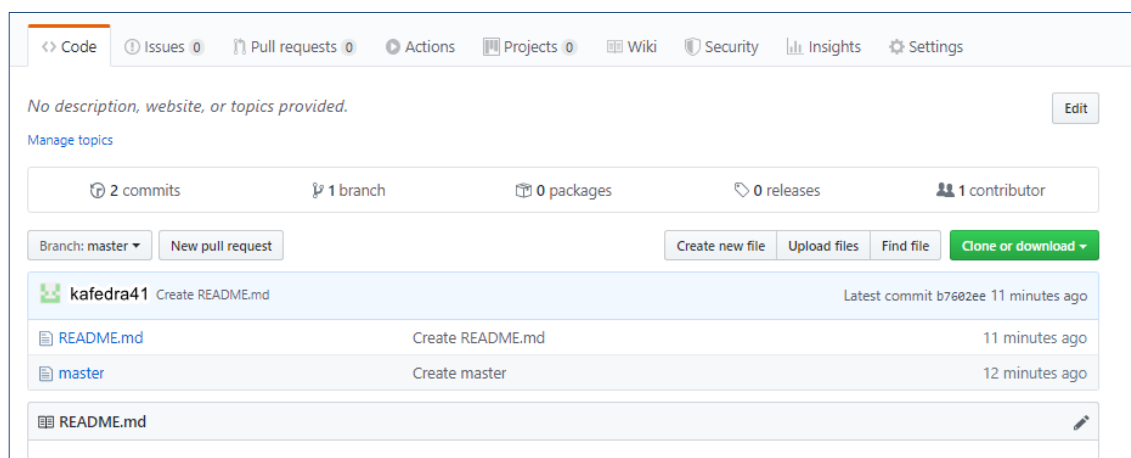


Рис. 8. Файлы проекта FirstRepository

3.4 Создайте вторую ветку проекта (рис. 9). Для этого откройте выпадающее меню Branch: и введите название, например, second_edit и активируйте Create branch: second_edit.

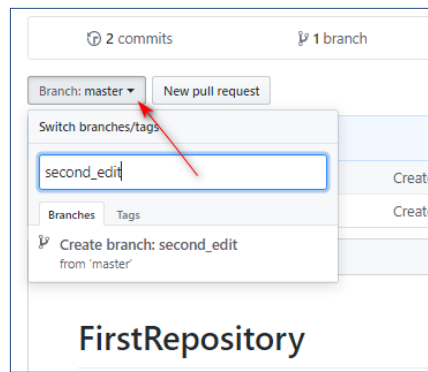


Рис. 9. Создание второй ветки `second_edit` проекта `FirstRepository`

3.5. Посмотрите, нажав на 1branch (рис.9), что существуют две ветви проекта: `master` и `second-edit` (рис.10).

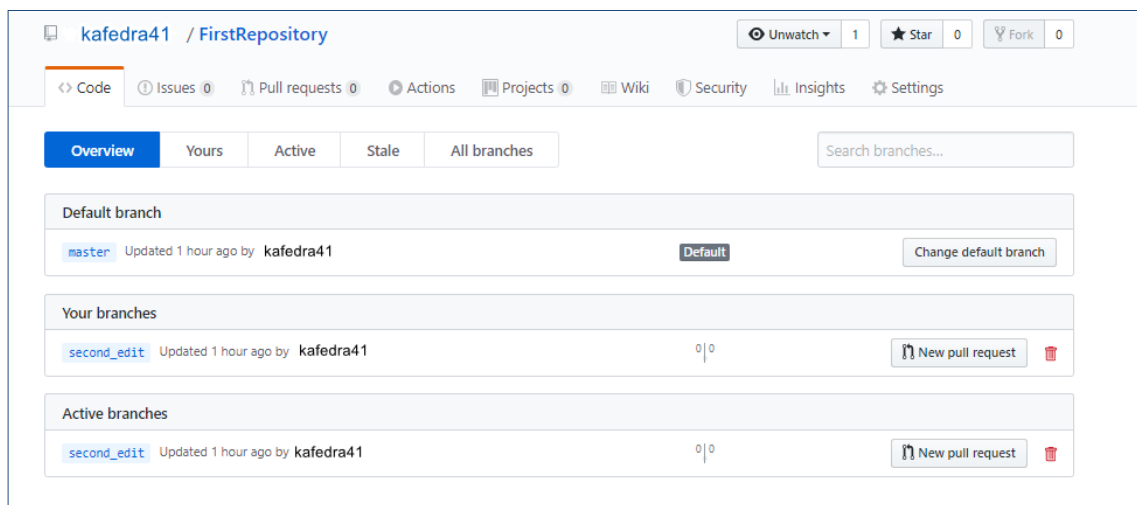


Рис. 10. Просмотр существующих веток проекта

4. Внесение изменений в файл проекта.

4.1. После создания ветки `second_edit` именно она стала активной веткой проекта, в которую будем вносить изменения.

На GitHub сохраненные изменения называются *коммитами*. Каждый коммит имеет соответствующее *сообщение коммита*, которое представляет собой описание, объясняющее, почему было сделано конкретное изменение. Сообщения фиксации фиксируют историю ваших изменений, чтобы другие участники могли понять, что вы сделали и почему.

Все изменения требуется описывать в файле `README.mb`. Для редактирования этого файла нажмите на значок карандаша в правом углу файла, внесите изменения и сохраните их, нажав на кнопку `Commit Changes` (рис.11). Обратите внимание, что изменения коснутся только файла `README.mb`, который хранится в ветке `second_edit`.

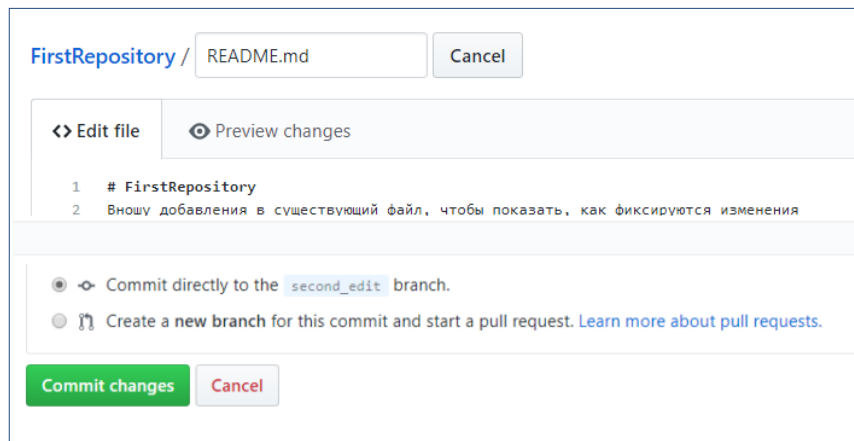


Рис. 11. Внесение записей об изменениях программного кода в файл README.mb.

4.2. Внесите изменения в файл master, который находится в ветке second_edit и сохраните их нажатием на кнопку Commit Changes (рис. 12).

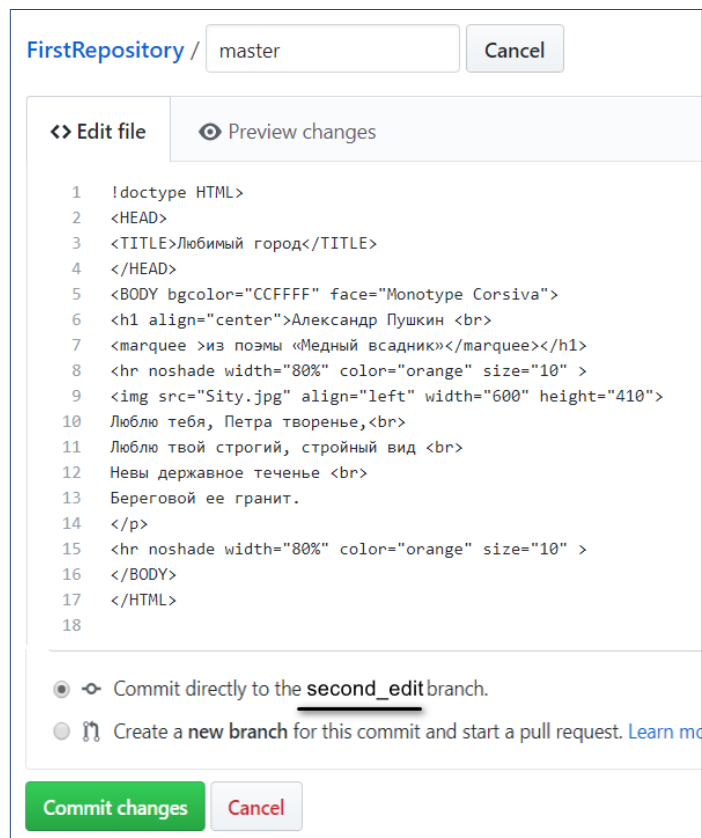


Рис. 12. Внесение изменений в файл master ветки second_edit

5. Извлечение изменений с помощью пул-запроса (Pull Requests).

Запрос на добавление измененных или добавленных строк кода или созданных в другой ветви проекта файлов показывает эти различия этих ветвей. В GitHub предусмотрена возможность работы одновременно нескольких человек над одним проектом. Эта система контроля версий была создана Линусом Торвальдсом именно для реализации возможности редактирования ядра операционной системы с открытым кодом Linux одновременно множеством разработчиков программного обеспечения. В GitHub предусмотрена

подсистема @mention для сбора данных об изменениях от конкретных участников проекта вне зависимости от их географического положения.

Порядок выполнения запроса.

5.1. Нажмите на кнопку Pull Requests и выполните New Pull Request (рис.13).

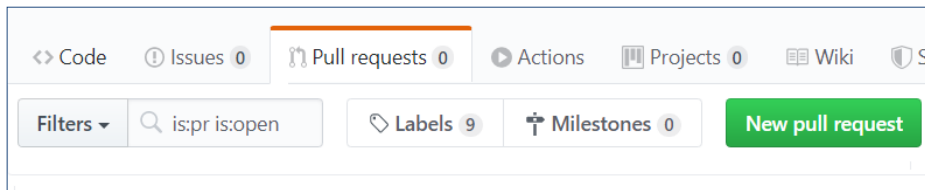


Рис. 13. Создание запроса на внесение изменений New Pull Request

5.2. В поле Слияние выберите источник слияния ветки second_edit для объединения изменений с исходными файлами из ветки master (рис.14) .

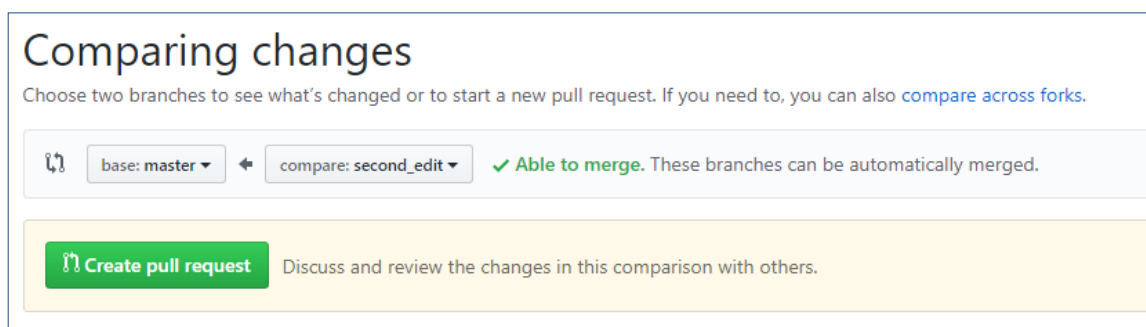


Рис. 14. Выбор источника слияния

5.3. Посмотрите на различия в файлах на странице сравнения, убедитесь, что это модификации, которые требуется сохранить в исходных файлах. Обратите внимание на условное форматирование цветом в программных кодах файлов областей изменений, дополнений и удалений соответственно зеленым, синим или красным цветами (рис.15).

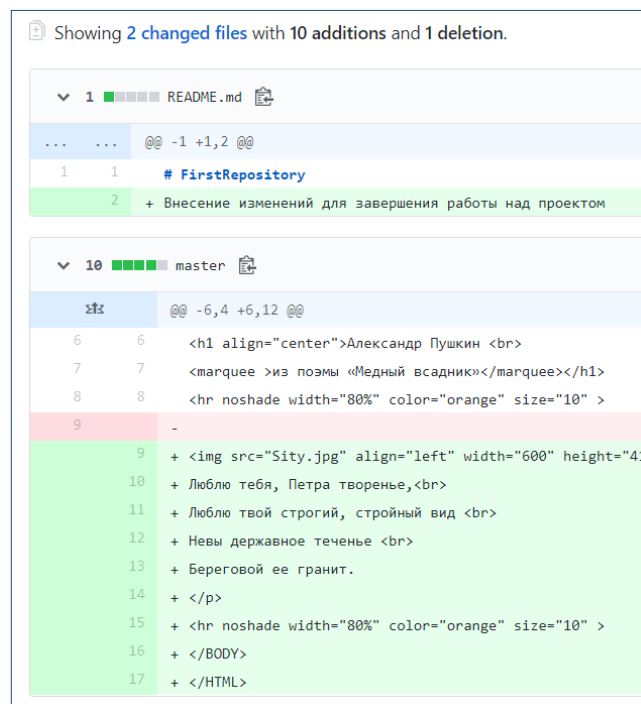


Рис. 15. Цветовое выделение изменений программного кода в новой редакции

5.4. После проверки вносимых изменений нажмите на кнопку Create pull request (рис. 14).

5.5. Дайте запросу на получение названия заголовков и напишите краткое описание ваших изменений (рис. 16) и выполните запрос.

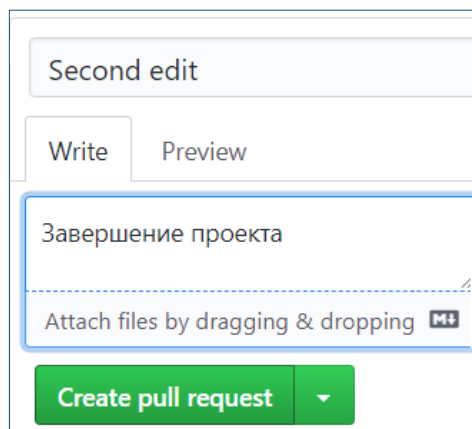


Рис. 16. Название запроса на изменения

5.5 Подтвердите выполнение слияния нажатием на Merge pull request (рис. 17) и затем Confirm merge.

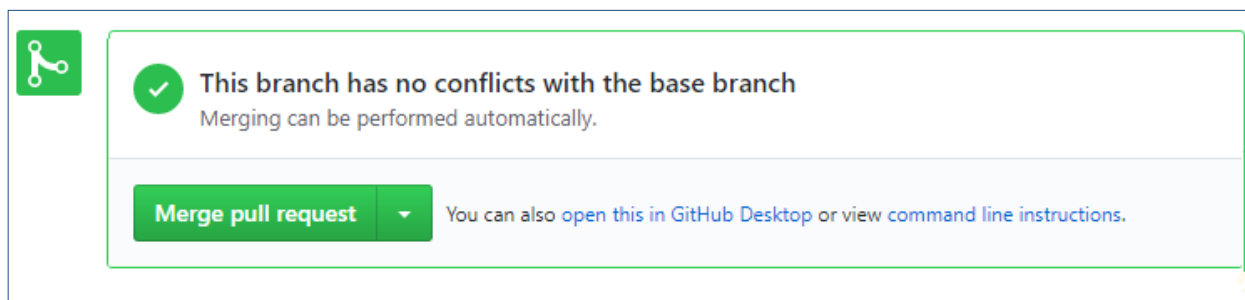


Рис. 17. Подтверждения слияния запроса на изменения с исходными файлами

5.6. После успешного слияния удалите ветку `second_edit` (рис. 18) на фиолетовом фоне.

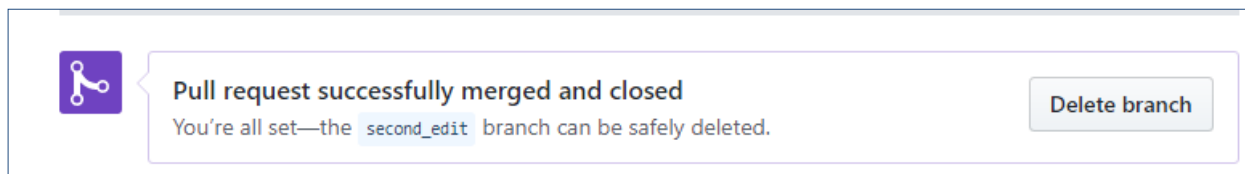


Рис. 18. Удаление ветки `second_edit` на фиолетовом фоне

6. Работа в приложении GitHub Desktop.

Действия по изменению исходного кода проводят не только на сервере, но и на десктопном приложении, которое было установлено на персональный компьютер после выполнения в п.2 методических указаний. Как правило, удобнее работать на персональном компьютере, на котором предустановленно множество полезных программ необходимых для написания эффективного кода. Затем эти изменения вносят в файлы в приложение GitHub Desktop и передают в удаленный репозиторий на GitHub.

6.1. Для клонирования репозитория нажмите на кнопку и выберите из выпадающего меню Clone Desktop. Автоматически создастся связь с приложением на компьютере пользователя и на персональном компьютере разработчика будет создана папка для репозитория, например E:\GIT\FirstRepository (рис.19). При необходимости можно увидеть полное имя папки при наведении курсора на название текущего репозитория (рис. 20).

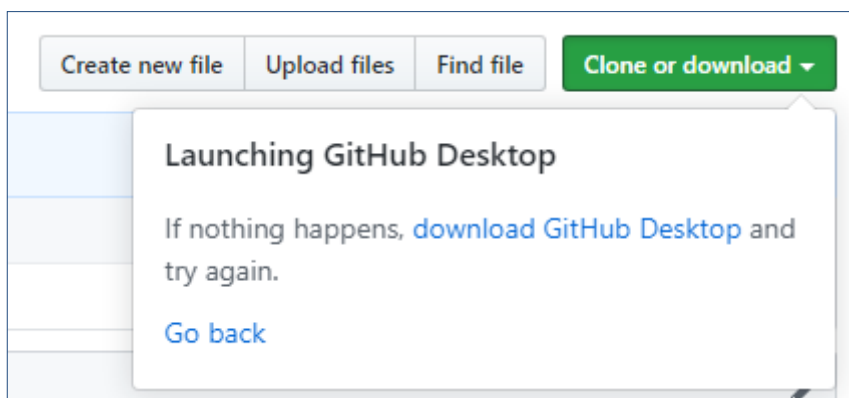


Рис. 19. Создание связи с приложением GitHub Desktop



Рис. 20. Полное имя папки репозитория на компьютере пользователя

В рассматриваемом случае в папке репозитория FirstRepository будут сохранены два файла: master и README.mb.

6.2. В приложении GitHub Desktop создайте новую ветвь third_edit (рис.21).

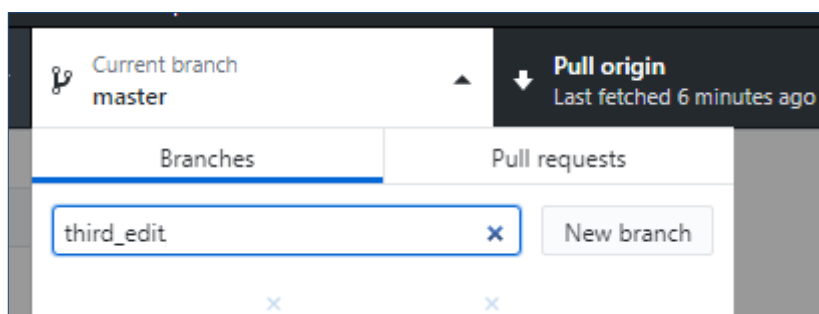


Рис. 21. Создание новой ветви проекта third_edit

6.3. Перенесите файлы проекта, которые хранились ранее только на персональном компьютере в папку E:\GIT\FirstRepository. Они сразу будут отображены в окне проекта на GitHub Desktop (рис.22).

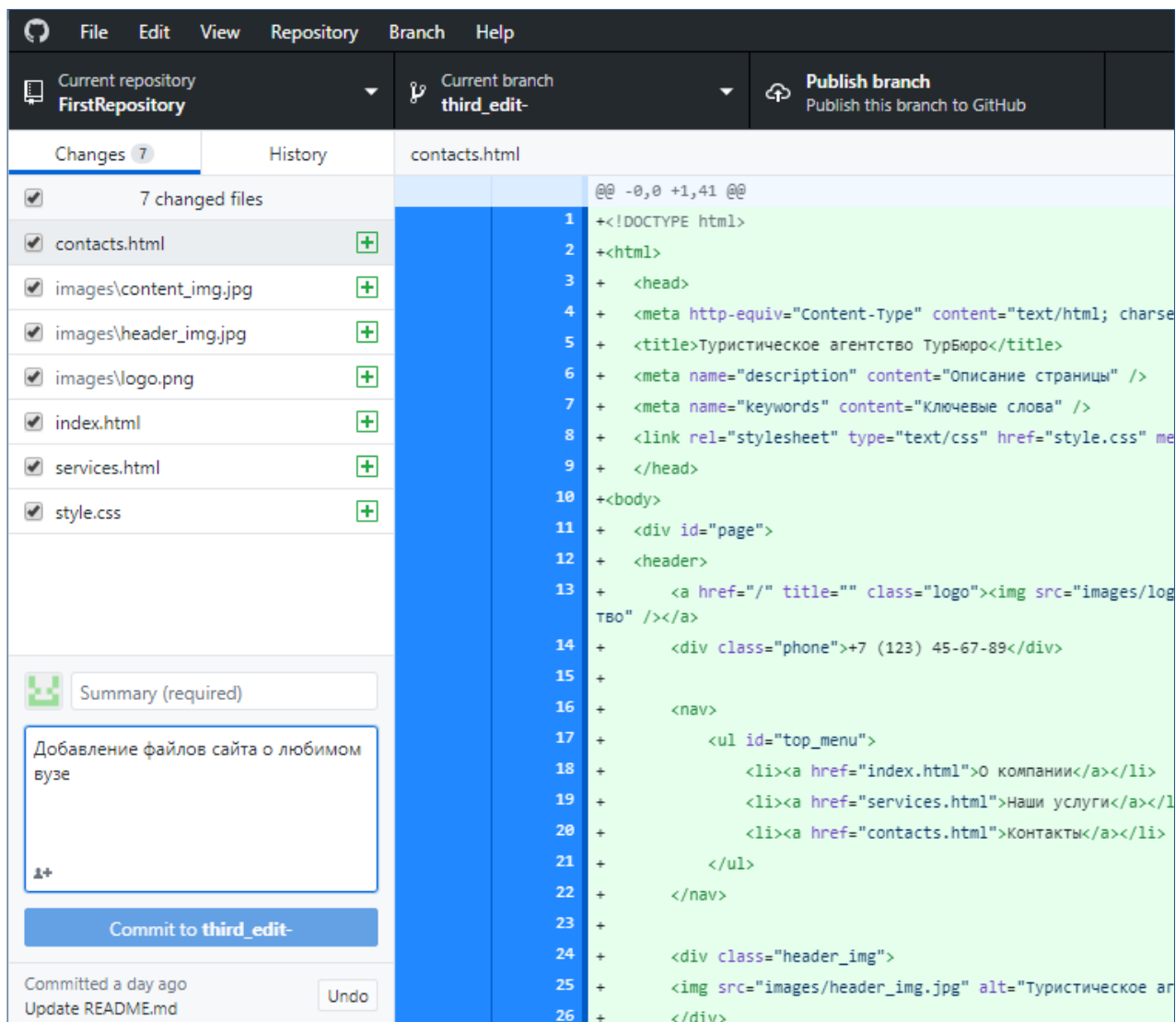


Рис. 22. Отображение новых файлов проекта в ветви `third_edit`

6.4. Добавьте описание изменений, в окне `Summary (required)` напишите название изменений, нажмите `Commit to third_edit`. Добавленные в проект файлы сохранены на локальном репозитории.

6.5. Для переноса изменений в ветви `third_edit` в удаленный репозиторий нажмите `Publish branch`.

6.6. Откройте Github и посмотрите на изменения в репозитории `FirstRepository`: появилась ветвь `third_edit` с новыми файлами проекта (рис. 23).

При необходимости редактирования файлов на локальном хранилище данных их следует открыть в соответствующей программе обработки.

6.7. Для демонстрации такой возможности создайте новую ветвь `fourth_edit` локального репозитория. Затем откройте файл `README.md` в текстовом редакторе и внесите в него новую строку (рис.25). Сохраните изменения в файле.

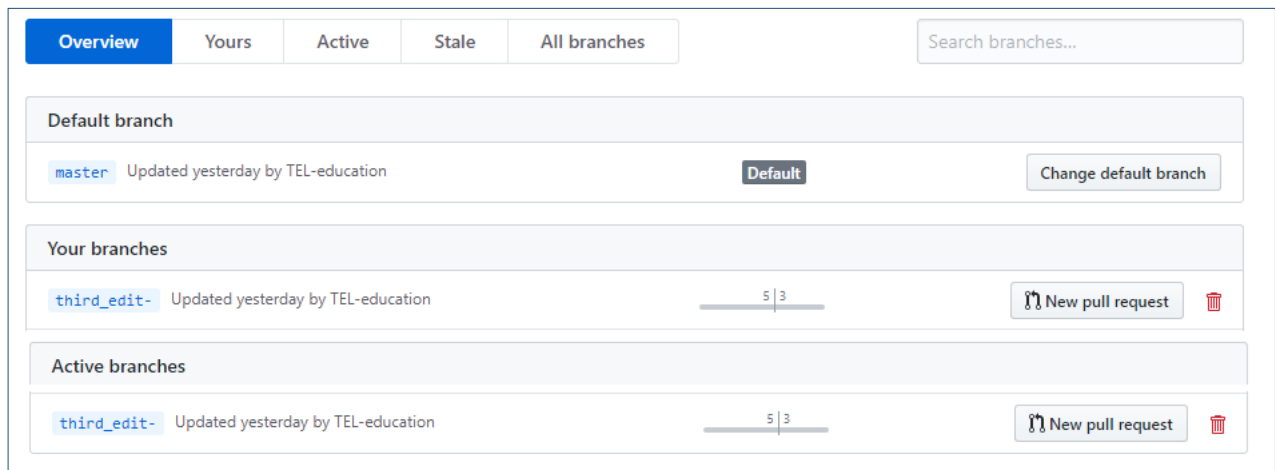


Рис. 23. Отображение изменений на GitHub

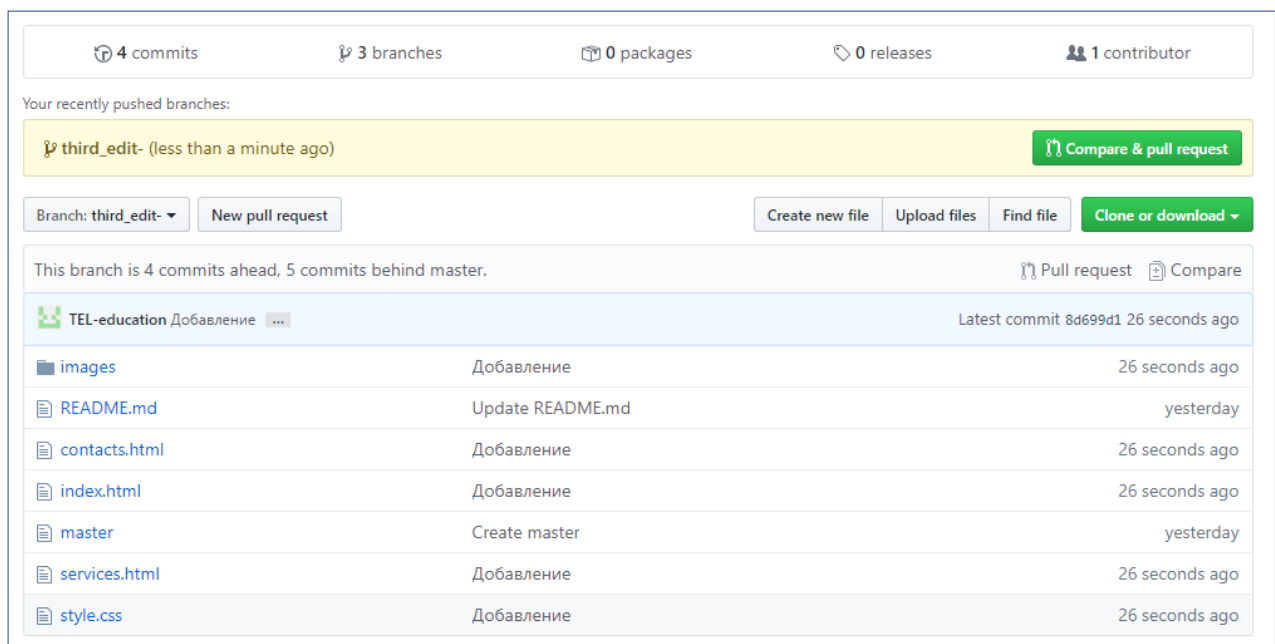


Рис. 24. Новые файлы проекта на удаленном репозитории

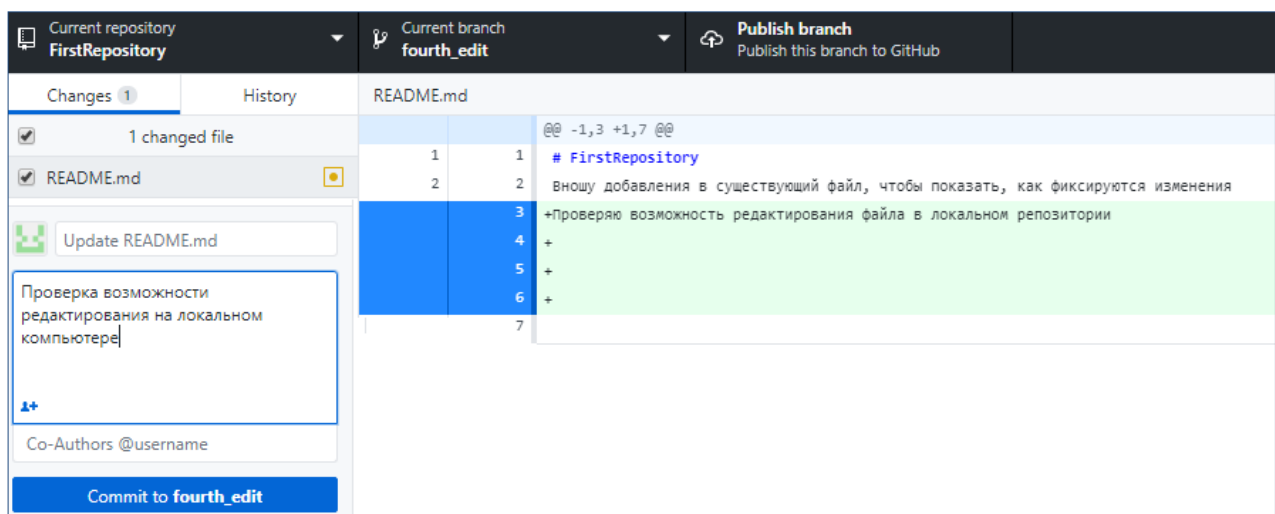


Рис. 25. Редактирование файлов в локальном репозитории

6.7. Нажмите последовательно Commit to third_edit и Publish branch для внесения изменений в удаленной хранилище файлов.

Таким образом, в репозитории FirstRepository хранятся 4 ветви проекта, каждая из которых обладает характерными чертами и позволяет развивать проект в конкретном направлении. Пользователь имеет право выбрать и развивать любую из этих ветвей.

7. Для добавления новых участников проекта на вкладке Setting активируйте в меню Collaborators и введите имя участника (рис.26).

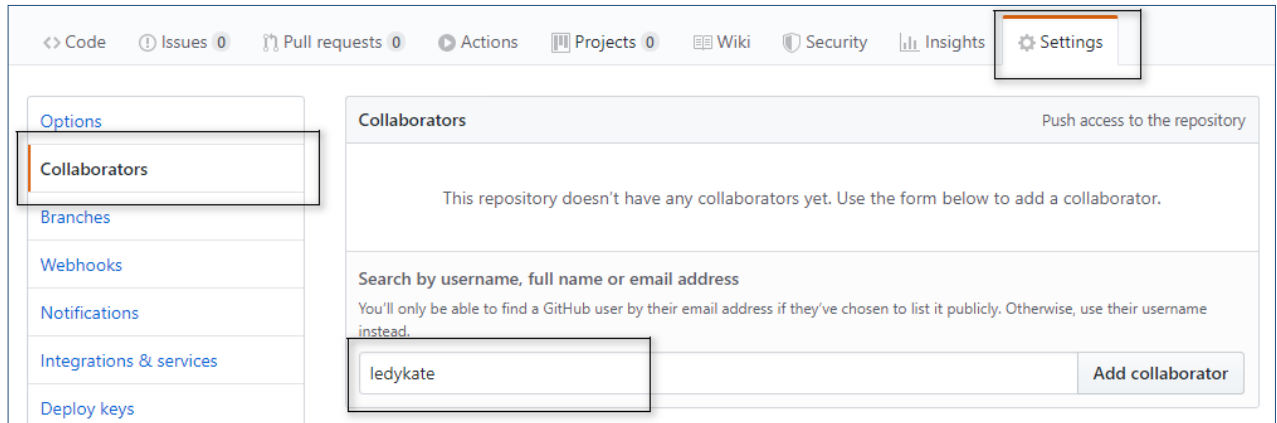


Рис. 26. Добавление нового участника проекта

Порядок выполнения работы.

1. Зарегистрироваться на сервере GitHub.
2. Установить приложение GitHub Desktop.
3. Процессы подготовки программной среды показать на скриншотах.
4. При написании кода для выполнения 4 лабораторной работы по разработке архиватора рабочие файлы программы сохранять на GitHub.
5. Каждое изменение фиксировать в системе контроле версий.
6. На скриншотах показать процесс работы с GitHub при внесении изменений в программный код архиватора.

При выполнении работы рекомендуется использовать приложение на GitHub Desktop для внесения изменений в исходные файлы, хранящиеся на сервере распределенной системы управления версиями Git.

Содержание отчета.

1. Цель работы.
2. Краткое описание назначения GitHub.
3. Скриншоты процессов подготовки программной среды для работы с системой контроля версий с пояснениями.
4. Скриншоты процесс работы с GitHub при внесении изменений в программный код архиватора с пояснениями.

Пример выполнения работы

Цель: познакомиться с технологией Git и сайтом GitHub.

Порядок выполнения работы:

Git — это распределенная система контроля версий с открытым исходным кодом.

GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Веб-сервис основан на системе контроля версий Git.

Зарегистрируемся на сайте GitHub, что показано на рисунке 1.

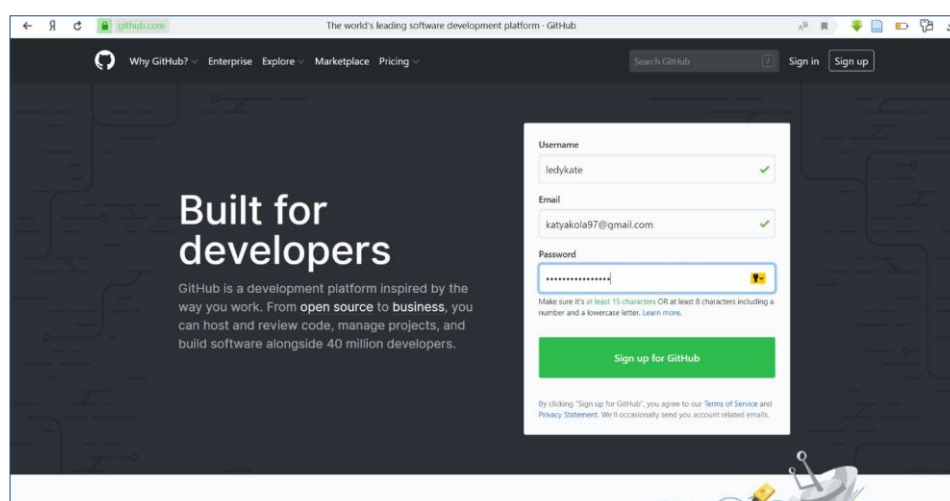


Рисунок 1 – Регистрация на сайте GitHub

После регистрации нам предложат создать свой первый репозиторий. Дадим ему имя и сделаем публичным, что показано на рисунке 2, а на рисунке 3 созданный удалённый репозиторий

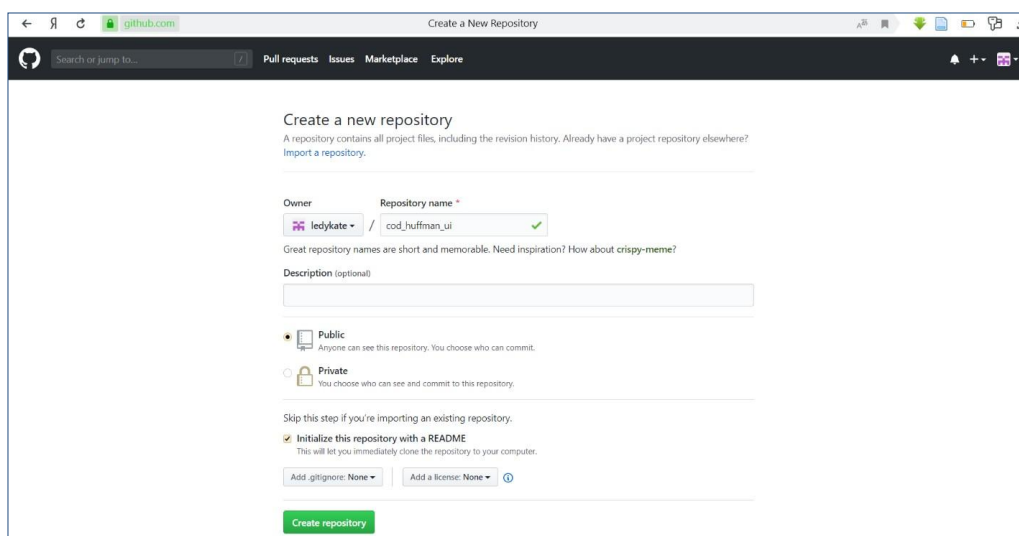


Рисунок 2 – Создание первого репозитория

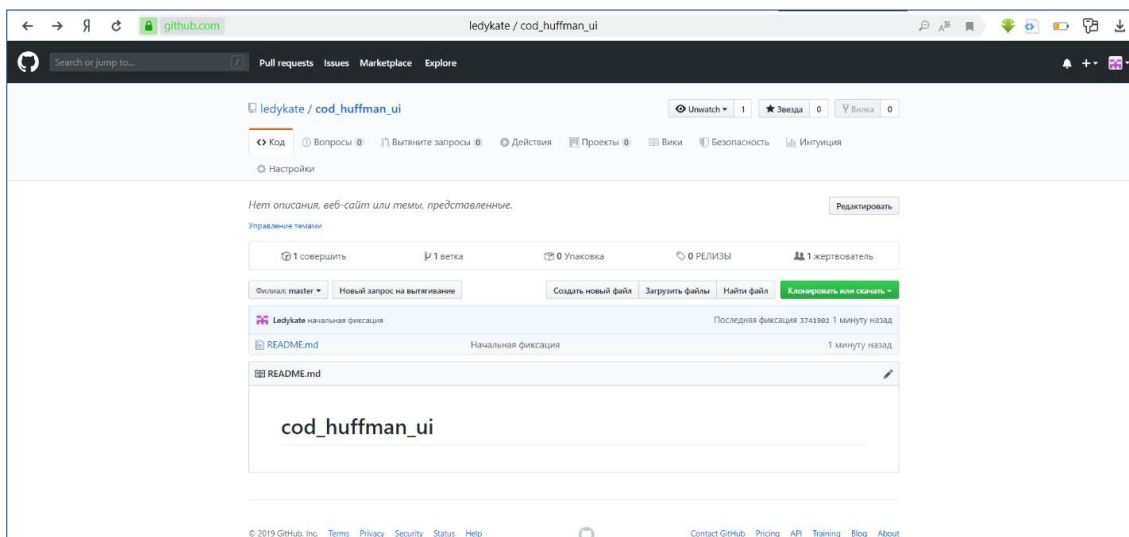


Рисунок 3 – Удалённый вид репозитория

Для удобной работы с Git скачаем приложения для работы в локальном репозитории на своём компьютере. На рисунке 4 показан сайт приложения GitHub Desktop для работы с системой Git на своём локальном компьютере.

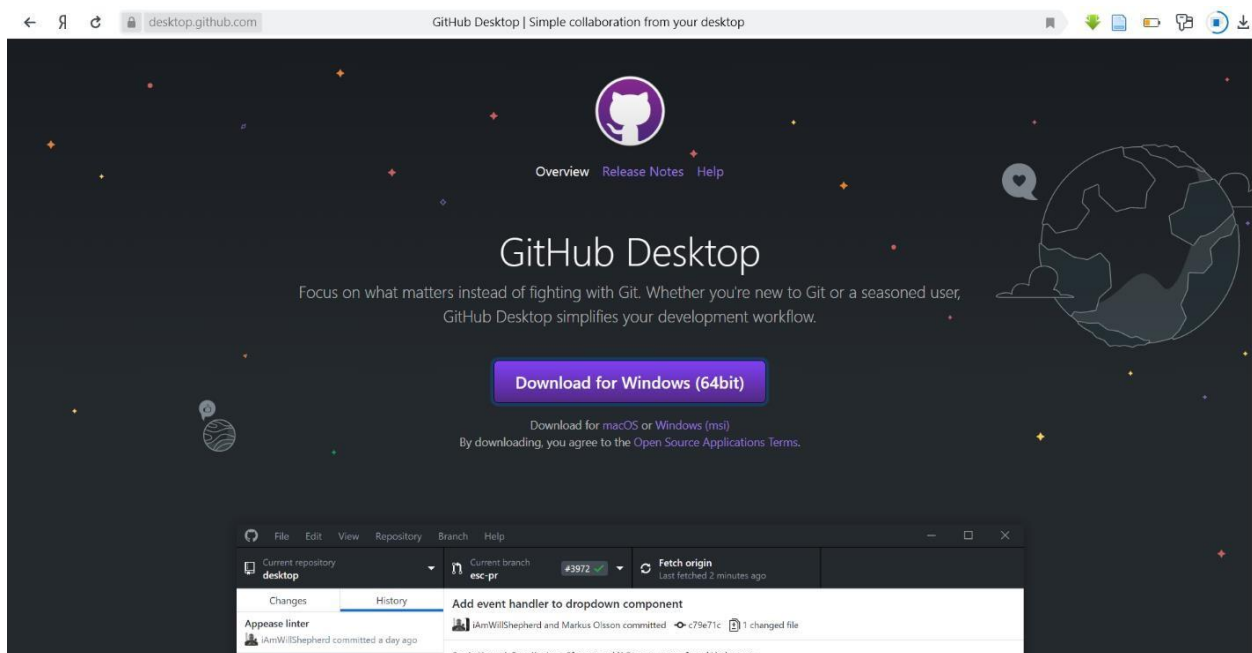


Рисунок 4 – Приложение GitHub Desktop

Подгружаем приложение и, если есть зарегистрированный аккаунт, проходим процедуру авторизации. На рисунке 5 показан вход в приложении GitHub Desktop.

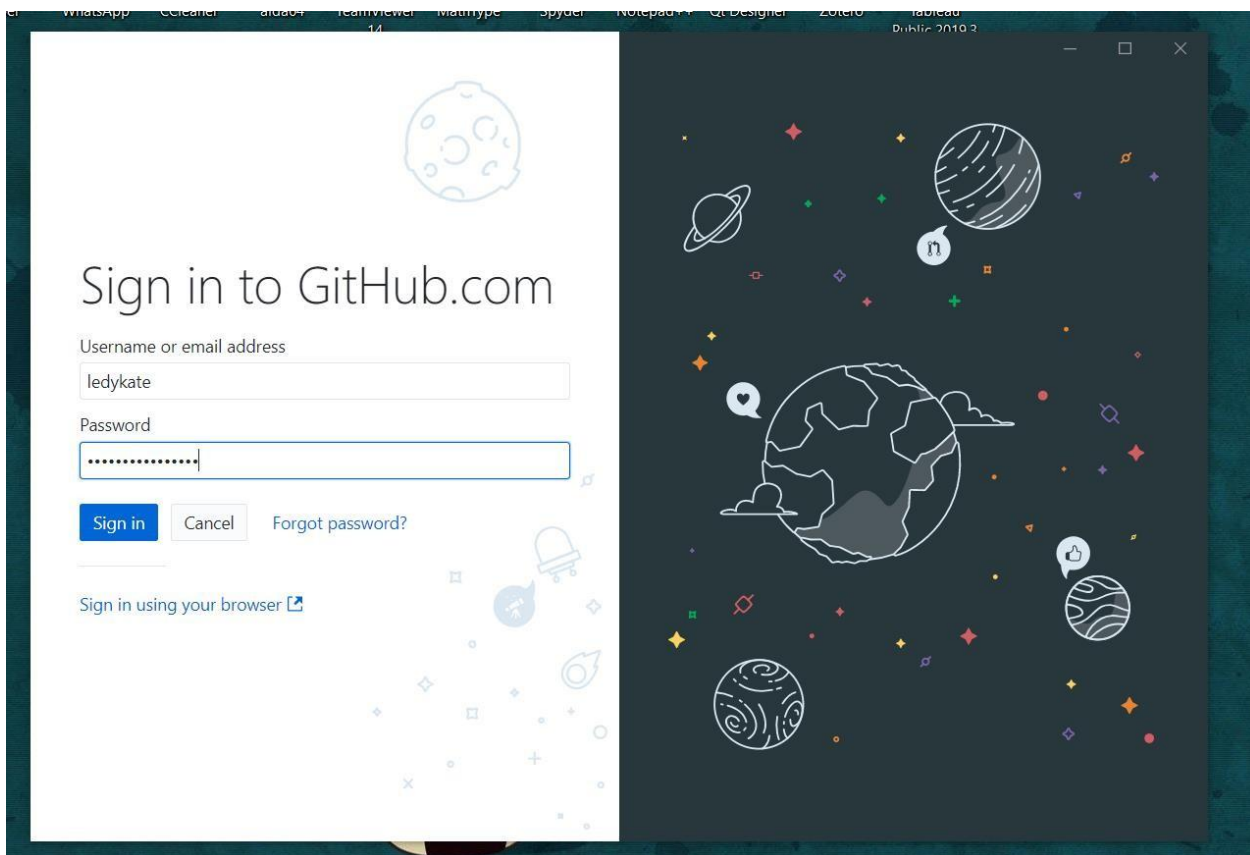


Рисунок 5 – Ввод логина и пароля в приложении GitHub Desktop

Затем можно клонировать удалённый репозиторий, созданный на сайте, в приложение GitHub Desktop. На рисунке 6 показано клонирование репозитория.

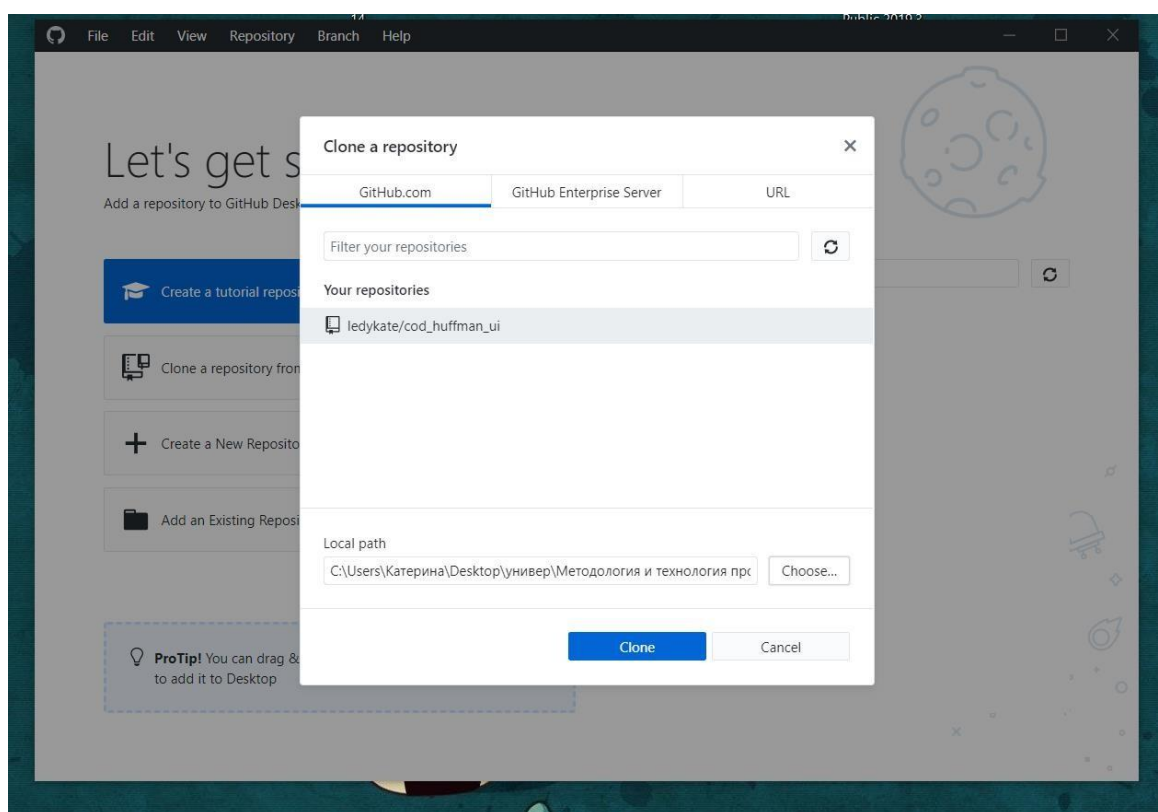


Рисунок 6 – Клонирование репозитория в GitHub Desktop

В выбранную папку для локального репозитория переместим файл с кодом на языке python. Можно добавить также название изменение и его описания, затем нажимаем Commit to master. Данные действия показана на рисунке 7.

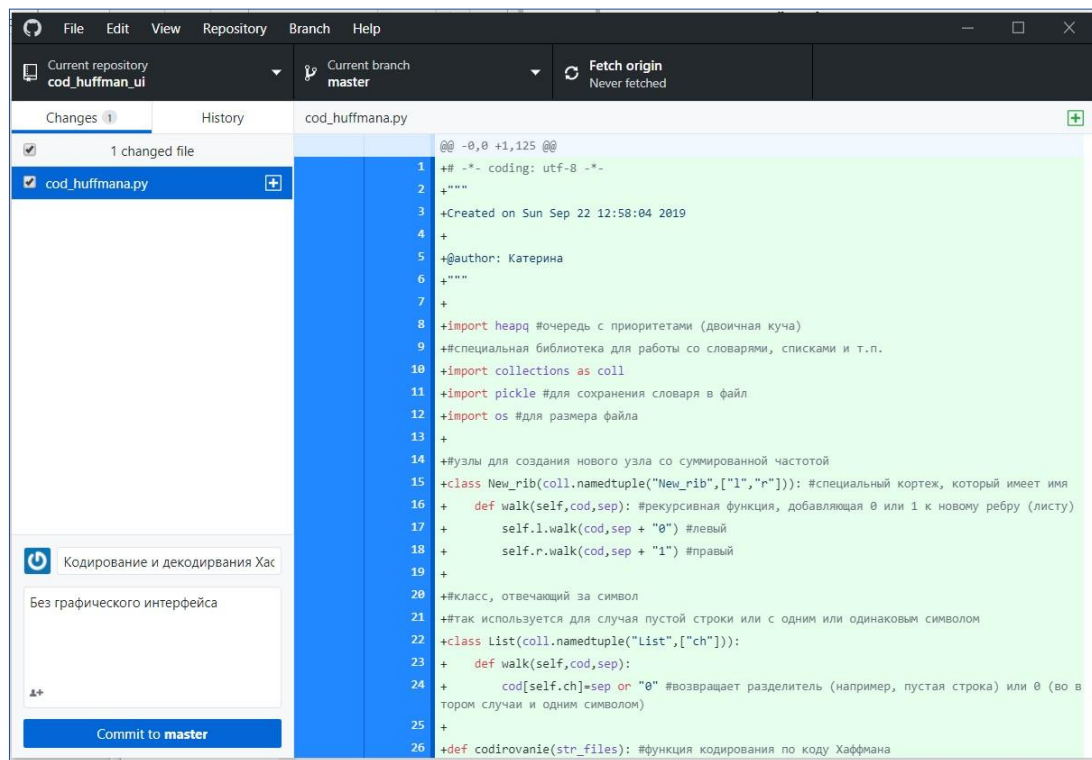


Рисунок 7 – Добавление файла в репозиторий в GitHub Desktop

На рисунке 8 показан процесс обновления удалённого репозитория на сайте с помощью кнопки Pushing to origin.

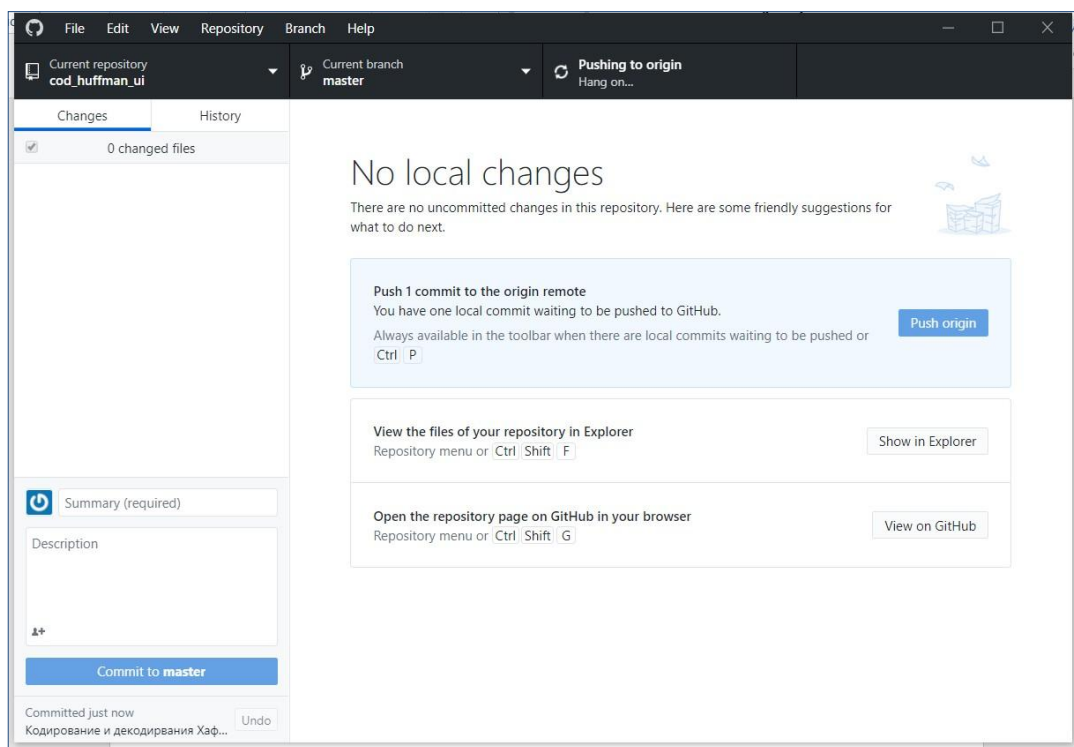


Рисунок 8 – Обновление удалённого репозитория в GitHub Desktop

Теперь на сайте GitHub отображается добавленный файл.

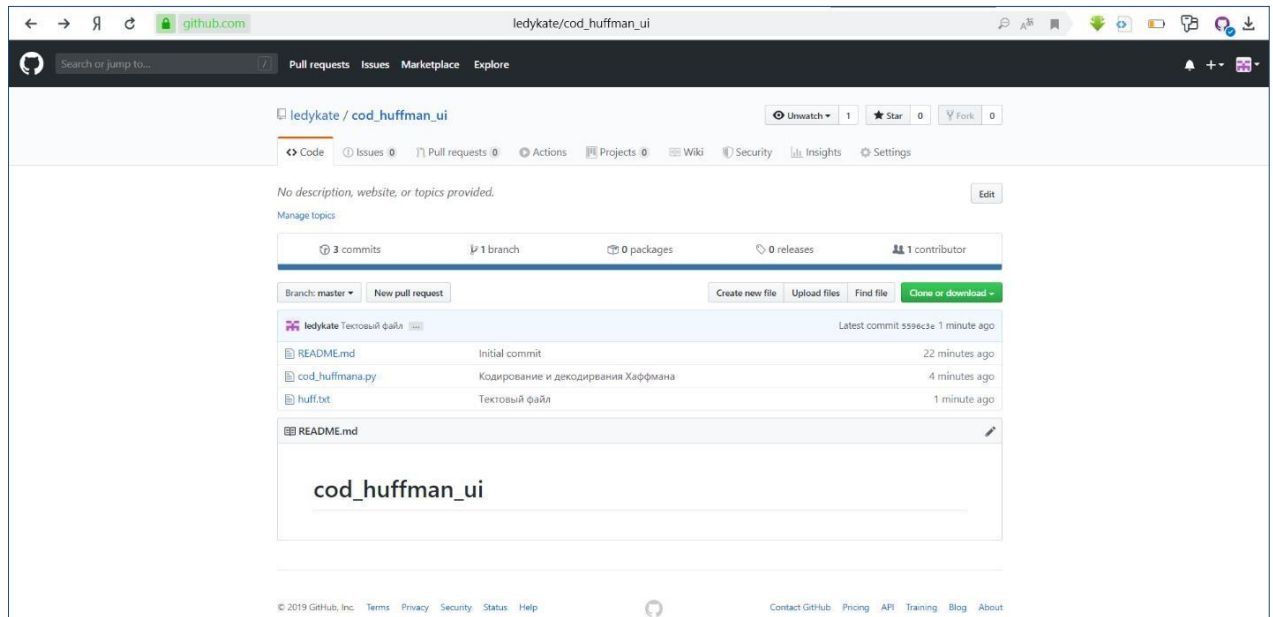


Рисунок 9 – Вид удалённого репозитория на сайте

На рисунке 10 представлен история изменения репозитория в приложении GitHub Desktop.

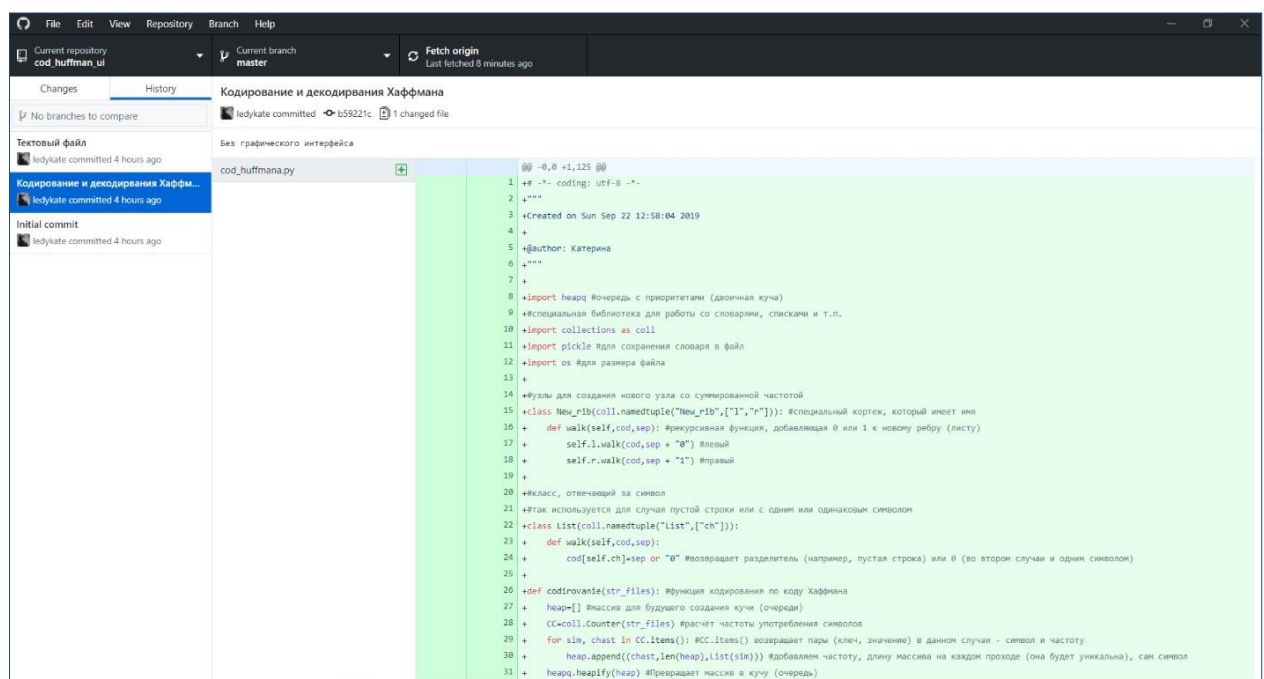


Рисунок 10 – История в приложении GitHub Desktop

Вывод: в ходе выполнения работы познакомились с системой Git. Создали свой первый репозиторий в GitHub. Установили приложение GitHub Desktop для удобной работы с репозиторием.