

Лабораторная работа №2

Тема: «Коллективная разработка приложения с использованием GitHub»

Цель работы: Получить практические навыки работы в коллективе с использованием современных систем контроля версий.

Теоретическая часть

Сайт github.com позиционируется как веб-сервис хостинга проектов с использованием системы контроля версий git, а также как социальная сеть для разработчиков. Пользователи могут создавать неограниченное число репозиторий, для каждого из которых предоставляется wiki, система issue tracking-a, есть возможность проводить code review и многое другое. GitHub на данный момент является самым популярным сервисом такого рода, обогнав Sourceforge и Google Code.

Для open-source проектов использование сайта бесплатно. При необходимости иметь приватные репозитории, есть возможность перейти на платный тарифный план (рисунок 1):

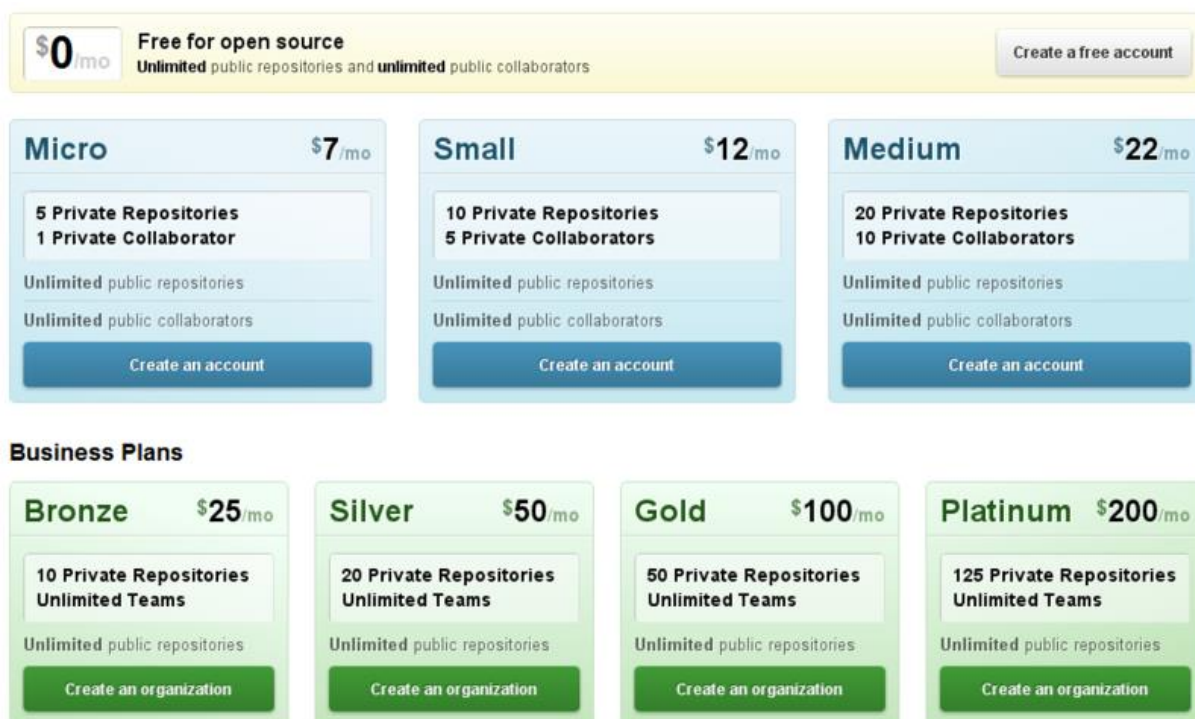


Рисунок 1 - тарифные планы

Для регистрации идем по ссылке github.com/signup/free и вводим свои данные.

После регистрации предлагается ознакомиться с инструкцией по быстрому старту либо создать новый проект (рисунок 2).

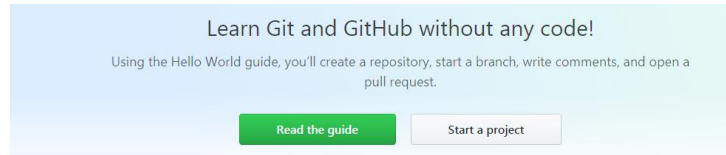


Рисунок 2 – стартовое предложение

Содержание инструкции.

Проект Hello World - это давняя традиция в программировании. Это простое упражнение, с которого начинается изучение чего-то нового. Давайте начнем знакомство с GitHub!

Вы узнаете, как:

- создать и использовать репозиторий;
- создать и управлять новой веткой;
- внести изменения в файл и провести их в GitHub как коммит;
- открыть и слить запрос на извлечение.

Что такое GitHub?

GitHub - это платформа для хостинга кода для контроля версий и совместной работы. Это позволяет вам и другим совместно работать над проектами из любого места.

В этом учебном пособии представлены основные возможности GitHub, такие как репозитории, ветви, коммиты и запросы на выборки. Вы создадите свой собственный репозиторий Hello World и изучите рабочий процесс GitHub Pull Request, популярный способ создания и анализа кода.

Чтобы завершить этот урок, вам нужна учетная запись GitHub.com и доступ в Интернет. Вам не нужно знать, как программировать, использовать командную строку или устанавливать Git (на основе программного обеспечения для управления версиями GitHub).

Шаг 1. Создайте репозиторий

Обычно репозиторий используется для организации единого проекта. Хранилища могут содержать папки и файлы, изображения, видео, электронные таблицы и наборы данных - все, что требуется вашему проекту. Мы рекомендуем включать README или файл с информацией о вашем проекте. GitHub упрощает добавление в одно и то же время, когда вы создаете новый репозиторий. Он также предлагает другие общие параметры, такие как файл лицензии.

Ваш репозиторий hello-world может быть местом, где вы храните идеи, ресурсы или даже делитесь и обсуждаете вещи с другими.

Чтобы создать новый репозиторий:

1. В правом верхнем углу рядом с вашим аватаром или идентификатором нажмите и выберите «Новый репозиторий» (рисунок 3).
2. Назовите свой репозиторий “Hello world” (рисунок 4).
3. Напишите краткое описание (рисунок 4).
4. Выберите «Инициализировать» этот репозиторий с помощью README (рисунок 4).
5. Нажмите на кнопку “Create repository”.

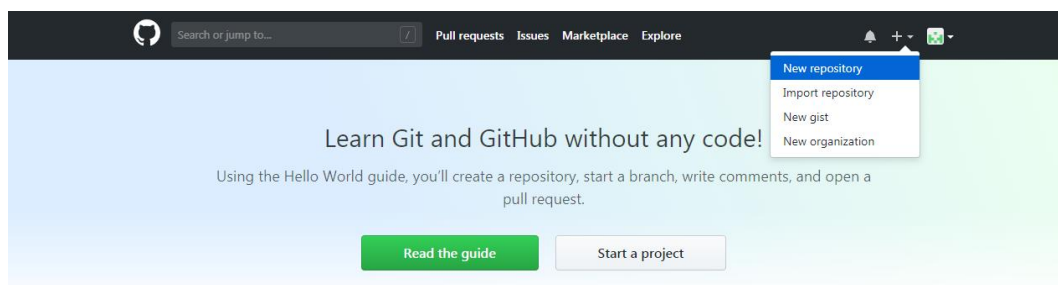


Рисунок 3 – создание нового репозитория

Search or jump to... Pull requests Issues Marketplace Explore

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: AlexandrAstafiev / Repository name: Hello world

Great repository names are short, unique, and don't contain spaces. Your new repository will be created as Hello-world

Description (optional): Мой первый репозиторий.

☒ Public
Anyone can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

☒ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None ⓘ

Create repository

Рисунок 4 – настройка нового репозитория

Шаг 2. Создание ветви

Ветвление - это способ работы с различными версиями репозитория за один раз.

По умолчанию ваш репозиторий имеет одну ветвь с именем master, которая считается окончательной ветвью. Мы используем ветви для экспериментов и редактирования, прежде чем их освоить.

Когда вы создаете ветвь от главной ветви, вы делаете копию или моментальный снимок мастера, как это было в тот момент времени. Если кто-то другой внес изменения в основную ветку, когда вы работали над своей веткой, вы можете использовать эти обновления.

Эта диаграмма (рисунок 5) показывает:

- мастер-ветвь;
- новую ветвь называется feature (потому что мы делаем «функциональные работы» на этой ветке);
- путь, который проходит ветвь feature, прежде чем будет объединена с мастер-ветвью.

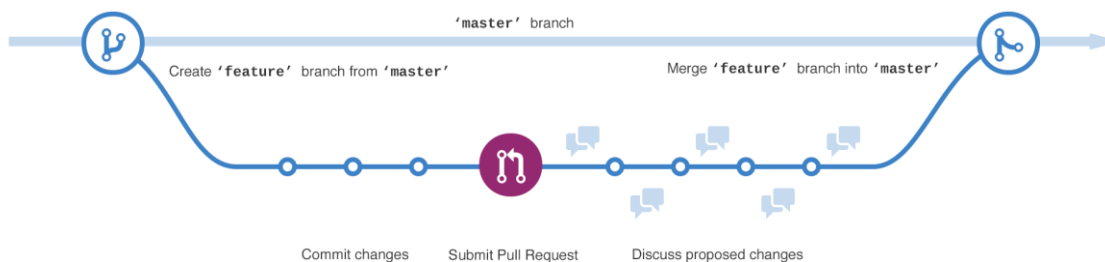


Рисунок 5 – диаграмма ветвления

Вы когда-нибудь сохраняли разные версии файла? Что-то вроде:

- story.txt
- story-joe-edit.txt
- story-joe-edit-reviewed.txt

Филиалы выполняют аналогичные цели в репозиториях GitHub.

Здесь, в GitHub, разработчики, писатели и дизайнеры используют филиалы для исправления ошибок и работы над функциями отдельно от нашей основной (производственной) ветки. Когда изменение будет готово, они объединяют свою ветвь с мастер-ветвью.

Чтобы создать новую ветку:

1. Перейдите в свой новый мир hello-world.
2. Нажмите на раскрывающийся список в верхней части списка файлов, в котором говорится: branch: master (рисунок 6).
3. Введите имя ветки, readme-edits, в новое текстовое поле филиала.
4. Выберите синюю кнопку «Создать ветку» или нажмите «Ввод» на клавиатуре.

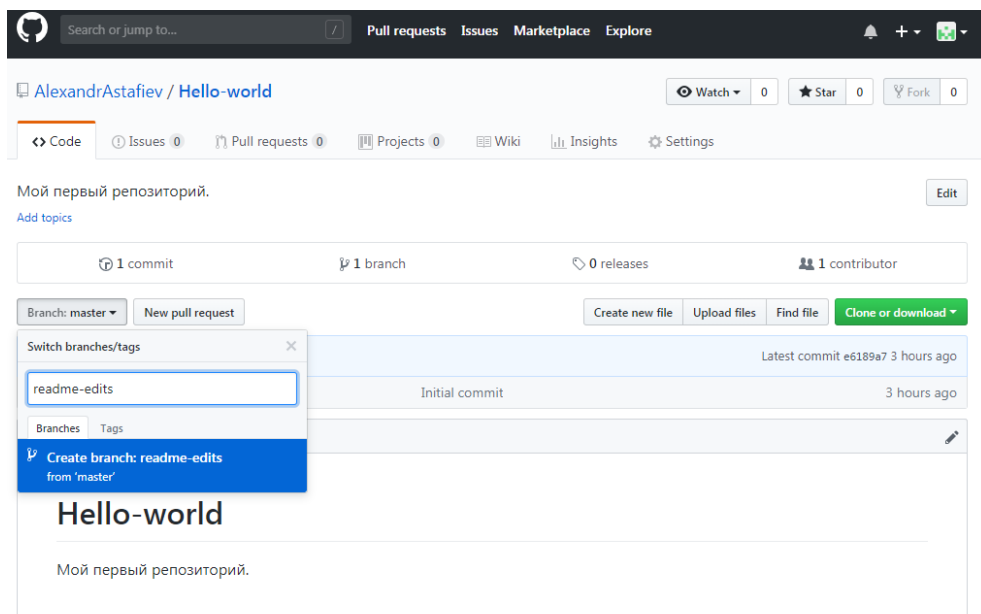


Рисунок 6 – создание новой ветви

Теперь у вас есть две ветви, мастер и readme-edits. Они выглядят точно так же, но ненадолго! Затем мы добавим наши изменения в новую ветку.

Шаг 3. Внести изменения и внести изменения

Теперь вы находитесь в представлении кода для своей ветви readme-edits, которая является копией мастера. Сделаем некоторые изменения.

На GitHub сохраненные изменения называются коммитами. Каждая фиксация имеет связанное сообщение фиксации, которое является описанием, объясняющим, почему было сделано определенное изменение. Фиксированные сообщения фиксируют историю ваших изменений, поэтому другие участники могут понять, что вы сделали и почему.

Создание и фиксация изменений (рисунок 7):

- Нажмите на файл README.md.
- нажмите на значок карандаша в правом верхнем углу файла, чтобы изменить его.
- в редакторе немного напишите о себе.
- напишите сообщение фиксации, которое описывает ваши изменения.
- нажмите кнопку «Commit changes».

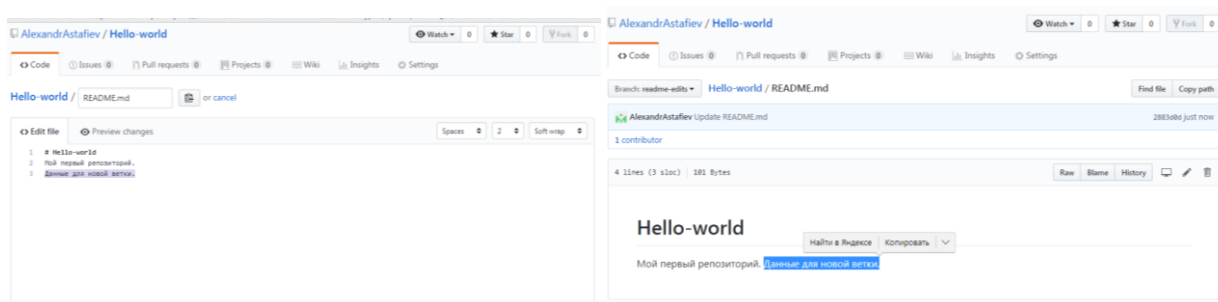


Рисунок 7 – внесение изменений

Эти изменения будут внесены только в файл README в вашей ветке редактирования readme-edits, поэтому теперь эта ветка содержит контент, отличный от master.

Шаг 4.Создание Pull Request

Хорошие изменения! Теперь, когда у вас есть изменения в ветке мастера, вы можете открыть Pull Request (запрос на вытягивание).

Pull Requests - это сердце сотрудничества GitHub. Когда вы открываете запрос на вытягивание, вы предлагаете свои изменения и запрашиваете, чтобы кто-то просмотрел и вложил ваш вклад и объединил их в свою ветку. Pull Request показывают различия или различия между содержимым обеих ветвей. Изменения, добавления и вычитания показаны зеленым и красным.

Как только вы сделаете фиксацию, вы можете открыть запрос на перенос и начать обсуждение, даже до того, как код будет завершен.

Используя систему @mention от GitHub в сообщении к Pull Request, вы можете запросить отзывы от конкретных людей или команд, независимо от того, находятся они в зале или в 10 часовых поясах.

Вы даже можете открыть Pull Request в своем собственном репозитории и объединить их самостоятельно. Это отличный способ изучить GitHub Flow, прежде чем работать над более крупными проектами.

Откройте Pull Request для изменений в README:

1. Перейдите на вкладку «Pull Requests», затем на странице «Pull Requests» нажмите зеленую кнопку «New Pull Requests» (рисунок 8).

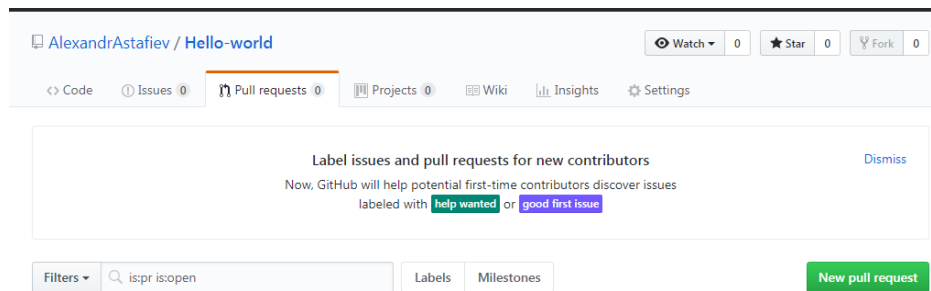


Рисунок 8 – закладка Pull Requests

2. В поле «Example comparisons» выберите ветку, которую вы сделали, readme-edits, чтобы сравнить с мастером (оригиналом) (рисунок 9).

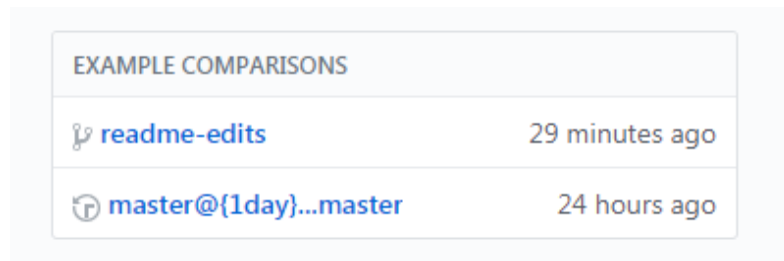


Рисунок 9 - поле «Example comparisons»

3. Ознакомьтесь с изменениями на странице сравнения на странице сравнения, убедитесь, что они являются тем, что вы хотите отправить (рисунок 10) .

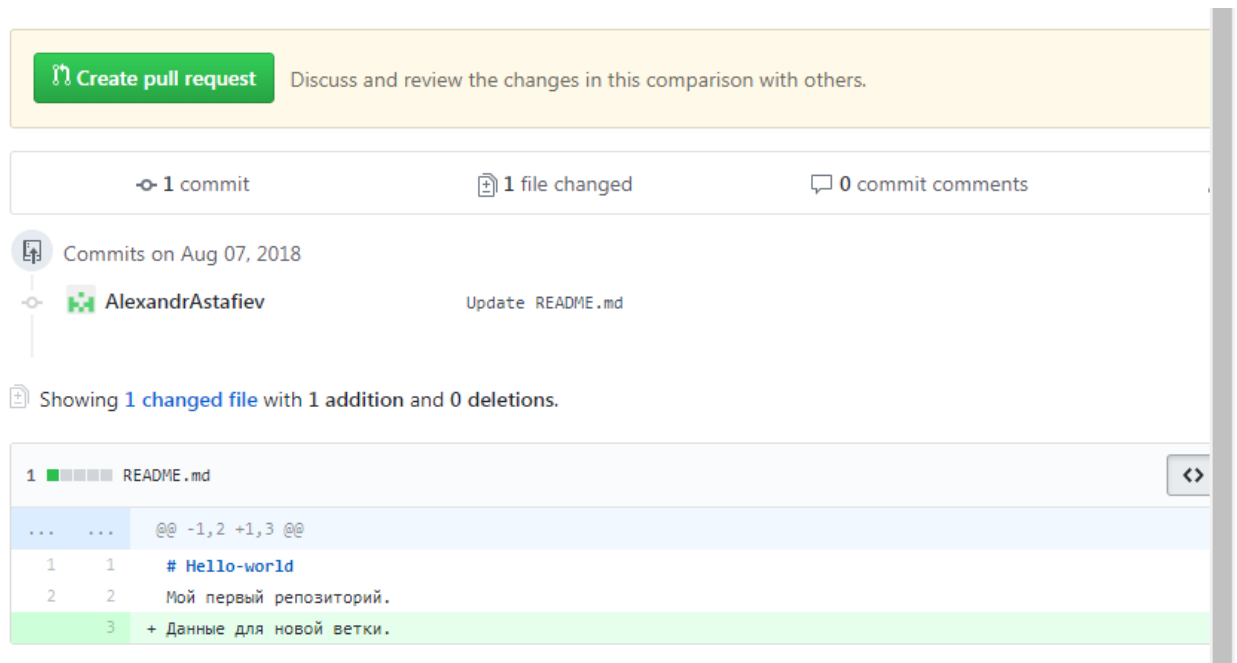
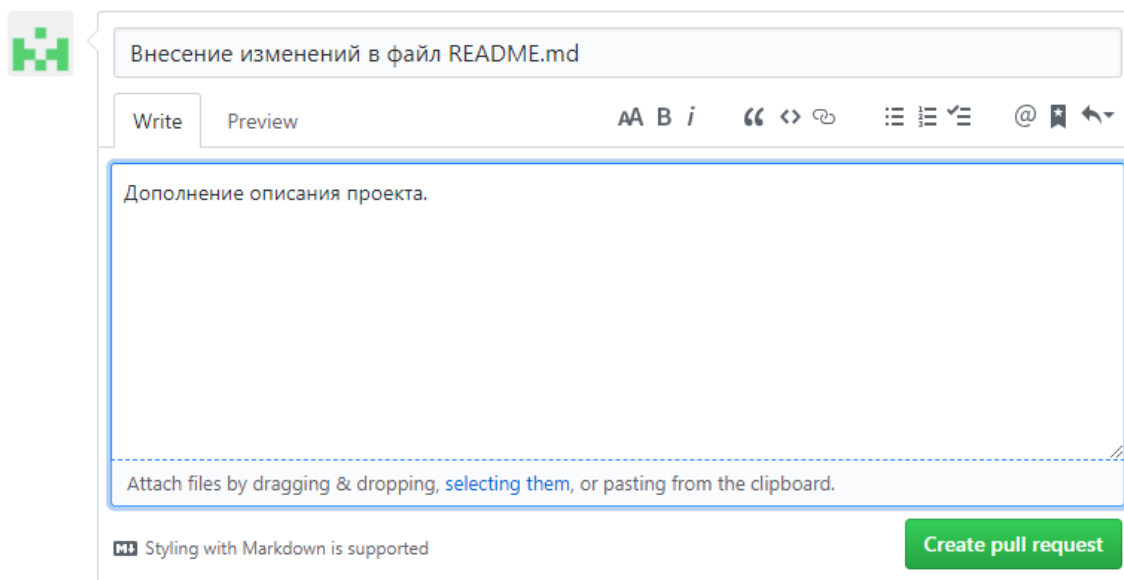


Рисунок 10 – сравнение изменений

4. Когда вы будете удовлетворены тем, что это те изменения, которые вы хотите отправить, нажмите большую зеленую кнопку Create Pull Request (рисунок 10).

5. Дайте вашему запросу заголовок и напишите краткое описание ваших изменений (рисунок 11).



Внесение изменений в файл README.md

Write Preview

Дополнение описания проекта.

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Create pull request

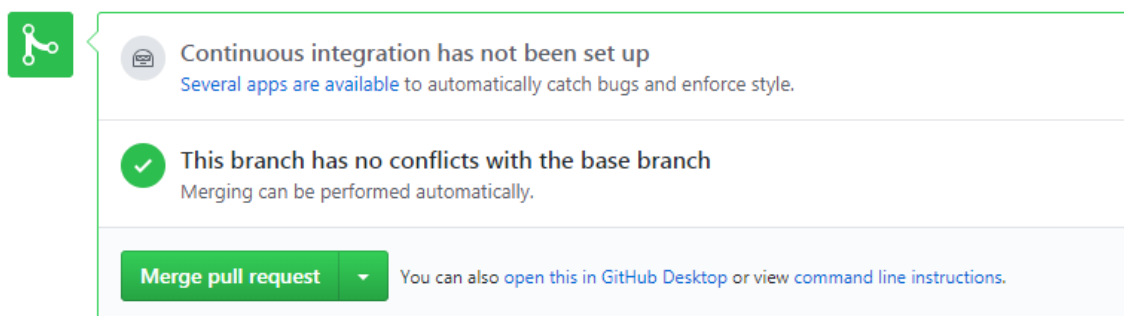
Рисунок 11 – описание Pull Request

6. Когда вы закончите свое сообщение, нажмите «Create Pull Request».

Шаг 5. Объедините свой запрос на вытягивание

На этом последнем этапе пришло время объединить ваши изменения - объединение ветки readme-edits в главную ветку.

Нажмите зеленую кнопку запроса на слияние, чтобы объединить изменения в мастер. Нажмите «Merge Pull Request» (рисунок 12).



Continuous integration has not been set up
Several apps are available to automatically catch bugs and enforce style.

✓ This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Рисунок 12 – объединение ветвей

Идем дальше и удаляем ветку, так как ее изменения были включены, с кнопкой «Удалить ветвь» в фиолетовом поле (рисунок 13).

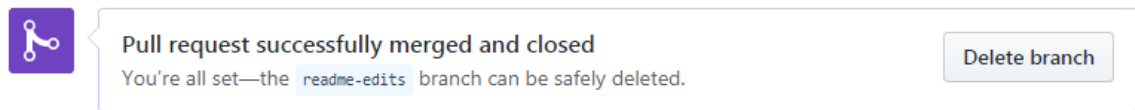


Рисунок 13 – удаление ветви

Пример коллективной разработки калькулятора

1. Необходимо разработать план работ по проекту с распределением обязанностей по членам коллектива.

Проект: Калькулятор		
Этап №	Наименование	Ответственное лицо
1	Разработка внешнего интерфейса и заглушек функций	Астафьев А.В.
2	Разработка функций сложения и вычитания	Астафьев А.В.
3	Разработка функций умножения и деления	Астафьев А.В.
4	Разработка функции обработки результата	Астафьев А.В.

2. На первом этапе обговорите заглушки функций, реализация которых перекладывается на исполнителей. Формат: имя функции, входные и выходные данные.

Для реализации калькулятора понадобится переменная для хранения типа операции и данных. Пусть это будет:

```
public int oper;  
public string value;
```

Определение функций заглушек:

```
public void sum(string str)  
{  
    return;  
}  
public void subtraction(string str)  
{  
    return;  
}  
public void multiplication(string str)  
{  
    return;  
}  
public void division(string str)  
{  
    return;  
}  
  
public string equal(string str)  
{  
    return "";
```

}

3. Исполнитель, ответственный за разработку интерфейса, разрабатывает каркас приложения и загружает его в репозиторий.

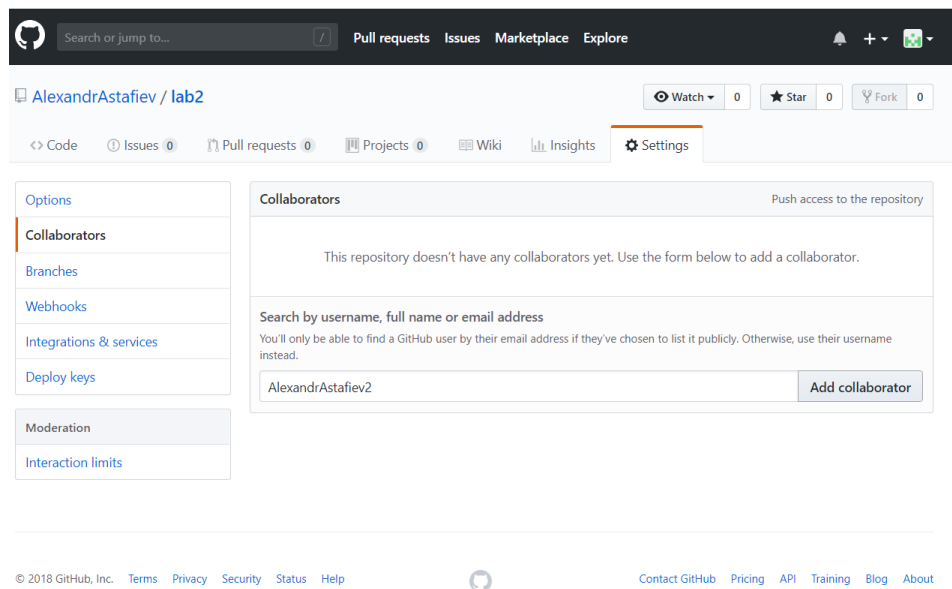
The image shows the GitHub 'Create a new repository' form on the left and a preview of the repository on the right. The form includes fields for 'Owner' (AlexandraAstafiev) and 'Repository name' (lab2), a 'Description' field, and options for 'Public' or 'Private' visibility. It also has checkboxes for 'Initialize this repository with a README' and 'Add .gitignore' and 'Add a license'. The preview on the right shows a list of files being uploaded, including 'TestCalc' and 'README.md', and a 'Commit changes' section.

4. Исполнители дополнительных функций разрабатывают их отдельно. Исполнители, при необходимости могут скопировать текущую версию проекта из репозитория. Также можно скопировать проект в собственный репозиторий с использованием функции Fork.

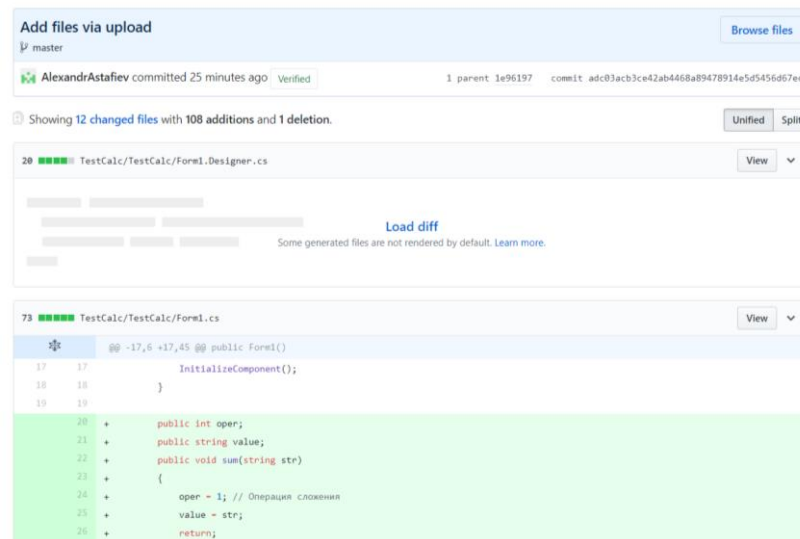
The image shows the GitHub repository page for 'AlexandraAstafiev / lab2'. It displays the repository's statistics: 4 commits, 1 branch, 0 releases, and 1 contributor. The 'Code' tab is selected, showing the repository's files: 'TestCalc', 'README.md', and another 'README.md'. The 'TestCalc' file is highlighted, showing its commit history and the option to 'Add files via upload'.

В результате выполнения должны получиться примерно такие функции:

5. Для получения возможности внесения изменений в проект руководитель проекта должен внести пользователей в соответствующий список. Для это нужно указать их в разделе «Settings - Collaborators». После чего вы получите ссылку для приглашения пользователя в проект.



6. После того, как пользователь получил доступ к работе с репозиторием можно переходить к слиянию готовых функций.



Практическая часть

1. Сформировать группу из 3-4 человек.
2. Разработать план работ по проекту с распределением обязанностей по членам коллектива.
3. Разработать проект системы в соответствии с заданием.
4. Создать собственный репозиторий на GitHub.
5. Пригласить членов коллектива.
6. Произвести инициализацию репозитория проектом из пункта 3.
7. Разработать части системы в соответствии с планом работ из пункта 2.

8. Произвести объединение результатов и тестирование полученного программного продукта.
9. Результаты работы представить в виде отчета.

Варианты заданий:

Команда 1: Автоматизированная информационная система библиотеки.

Команда 2: Автоматизированная информационная система автовокзала.

Команда 3: Автоматизированная информационная система отдела кадров.

Команда 4: Автоматизированная информационная система склада.

Команда 5: Автоматизированная информационная система больницы.