

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

# (ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт цифровых Кафедра

интеллектуальных систем компьютерных систем управления

Дисциплина «Основы системного программного обеспечения»

**Отчет по лабораторной работе №\_1\_**

# Работа Git Hub

**Выполнил**

**студент гр. АДБ-21-12** \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  **Cаидов Р.М.**

*(дата) (подпись)*

**Проверил**

**к.т.н., доцент** \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ **Ковалев И.А.**

*(дата) (подпись)*

**Москва 2023 г.**

**Содержание**

1. Введение
2. Создать учетную запись на Git Hub.
3. Создать репозиторий и научиться доgбавлять туда файлы.
4. Научиться отправлять изменения на сервер.
5. Описание и выполнение тестовых и индивидуальных заданий
6. Цель: научиться работать с системой контроля версий Git Hub.
7. Задачи работы:

**Введение**

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов. Для примеров будем использовать исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа.

Если вы графический или веб-дизайнер и хотели бы хранить каждую версию изображения или макета — а этого вам наверняка хочется — то пользоваться системой контроля версий будет очень мудрым решением. СКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь СКВ, вы всё испортите или потеряете файлы, всё можно будет легко восстановить.

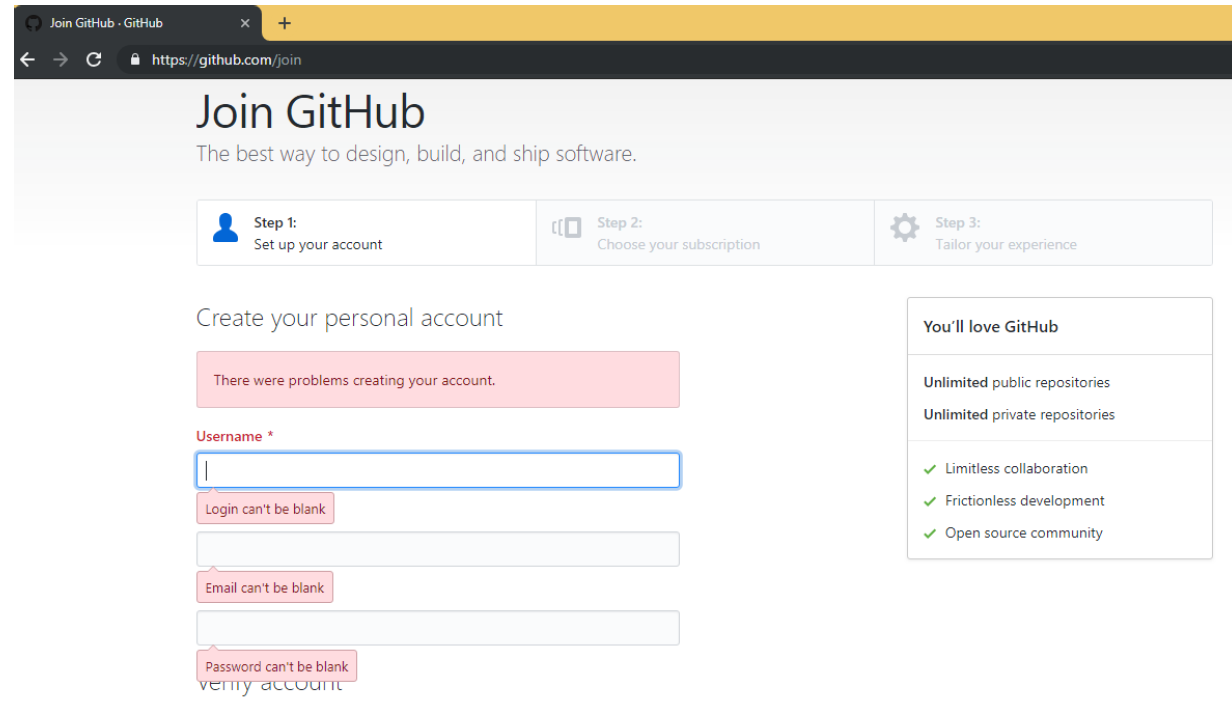
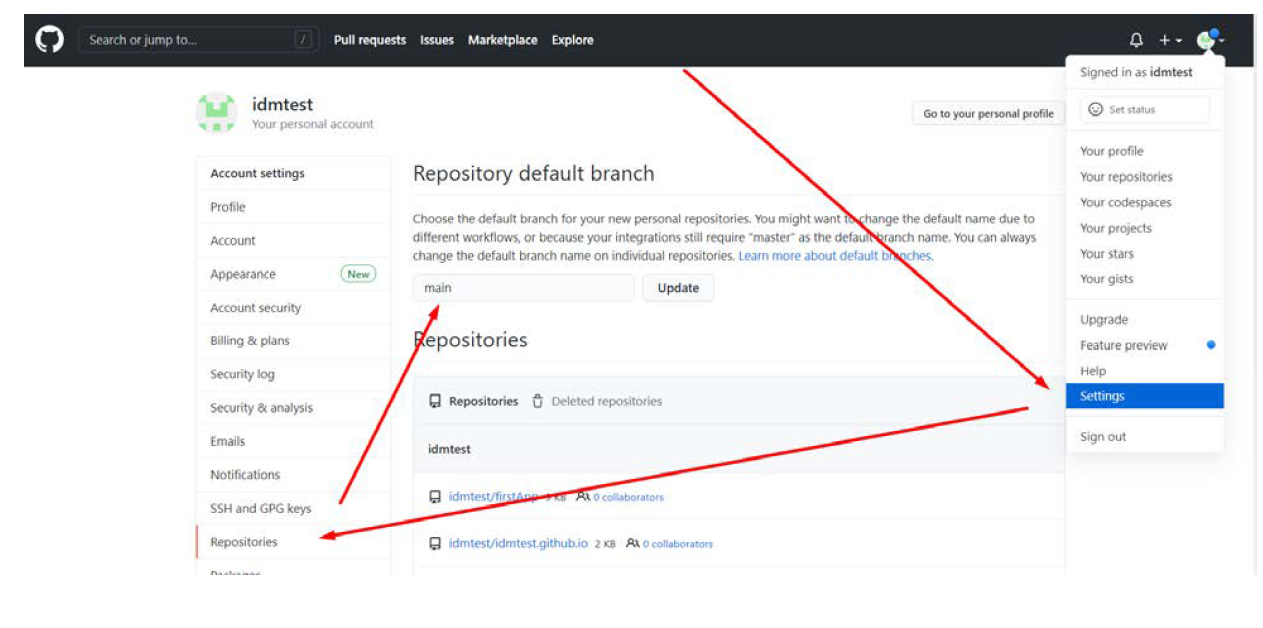
Индивидуальные задания

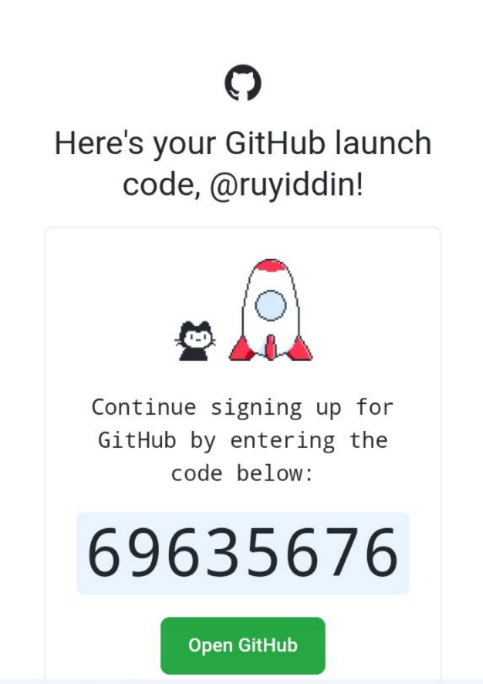
Создание учетной записи на github.com

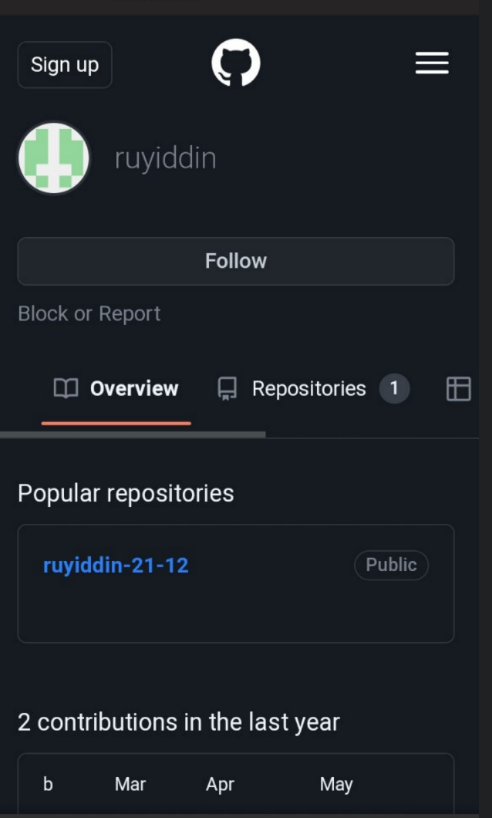
Зайдите на github.com

Зарегистрируйтесь на нем и подтвердите регистрацию на своей почте.

В процессе регистрации могут появиться вопросы: чем занимаетесь, какие ЯП знаете, какой аккаунт хотите. Все шаги можно пропустить, выбираете бесплатный аккаунт. Перед тем, как создавать репозиторий, перейдите в Settings и изменить название главной ветки с «main» на «master» и нажмите Update.





Дале создайте новый репозиторий

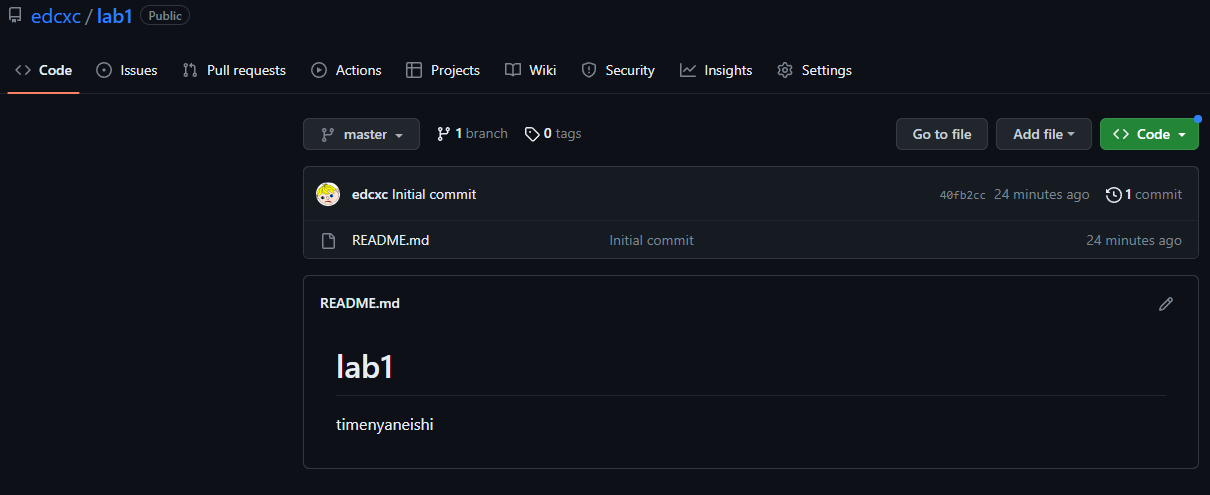
Перед вами появится оно создания нового репозитория.

Придумайте имя репозитория, можете вставить описание и выбрать тип. С недавнего времени github в бесплатной версии стал поддерживать скрытые репозитории, но на лабораторных работах мы будем использовать публичные.

Также можете установить галочку напротив предложения создать файл README **–** считается хорошим тоном, если кто-то заходит в ваш репозиторий и первым делом ему будет открываться описание вашего проекта.

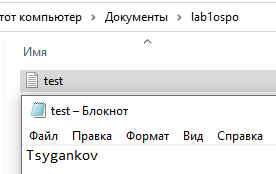
Итак, мы создали собственные репозиторий, в примере это https://github.

**3. Описание и выполнение тестовых и индивидуальных заданий** Создадим новый репозиторий LR1.

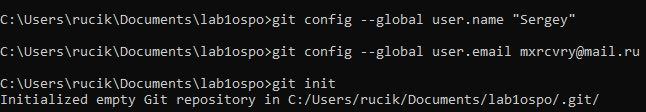


Создадим теперь локальную папку, которая будет связана с нашим облачным git хранилищем.

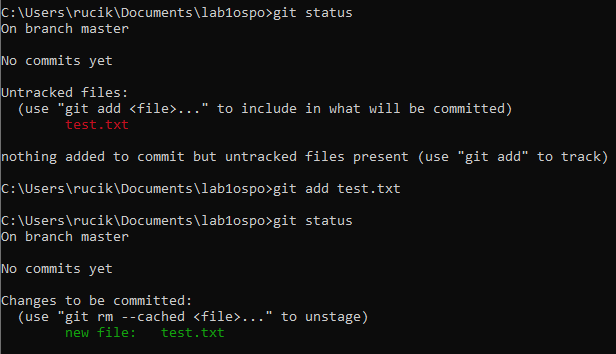
В ней создадим файлик test.txt с фамилией.



Настроим наше имя и адрес электронной почты. А также проинициализируем эту папку как git репозиторий.



Командная строка вернула нам сообщение, что проинициализирована пустая git директория. Теперь используя проводник windows создадим в своей папке любой текстовый файл и пропишем туда наше имя. Введем команду:

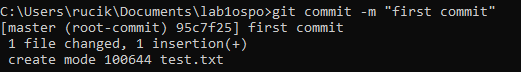


Нам отобразилось сообщение, что есть новый файл, но он не отслеживается. Это значит, что система еще не знает, нужно следить за изменениями в файле или его можно игнорировать. Для этого нужно добавить файлы командой add.

Выполнив эту команду посмотрим на статус репозитория и видим, что файл готов к коммиту.

Теперь закомитим нащ файл.

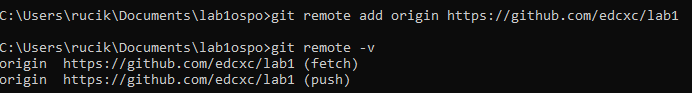
Чтобы зафиксировать изменения, нам нужно хотя бы одно изменение в области подготовки.



Коммит создался. И после вызова git status видно, что файлов для коммитов больше нет.



Теперь свяжем наш локальный репозиторий с репозиторием на GitHub, для этого выполним следующее.



Мы указываем, что подключаемся к репозиторию по URI, так как проект может иметь несколько репозиториев одновременно нужно давать им разные имена, а также указать origin у главного репозитория.

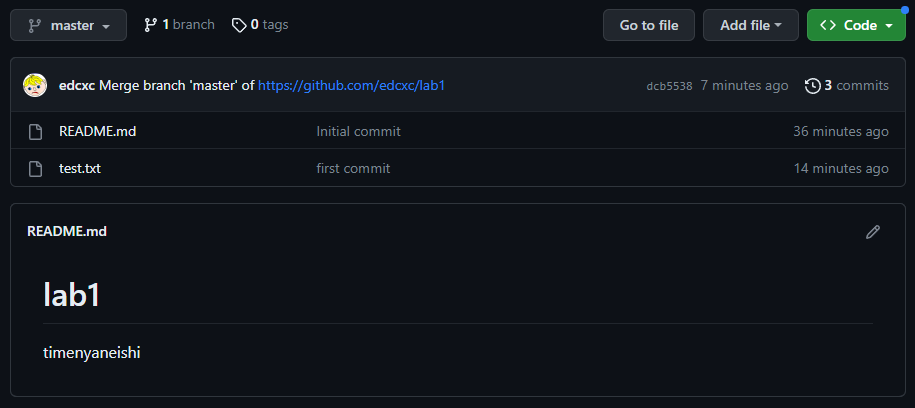
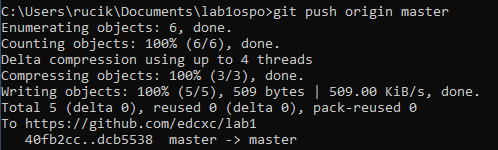
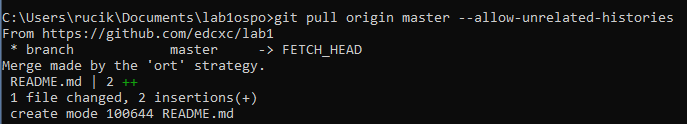
Затем просмотрим с помощью команды git remote –v к какому проекту мы подключены.

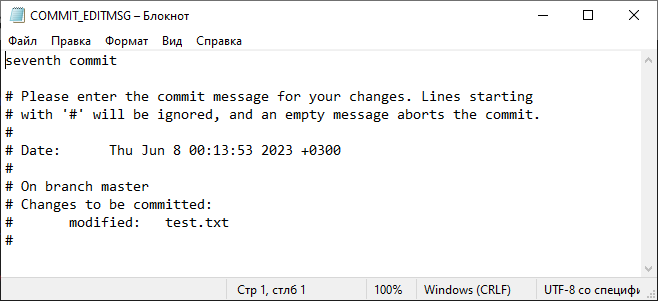
git pull origin master --allow-unrelated-histories – команда нужная, если у нас указано несколько репозиториев и у нас образовалось запутывание веток. В нашем случаем таких проблем не было. Так же можно удалить не правильный репозиторий командой git remote rm origin.

В нашем случае у нас есть файл и на сервере и локальном репозитории, поэтому нам вначале надо получить все данные с нашего репозитория, а потом уже их заливать на сервер нашими локальными данными.

Мы сделали изменения в нашем репозитории, теперь другие пользователи могут скачать изменения при помощи команды pull. git pull origin master

Теперь перешлем локальный коммит на сервер. Для этого существует команда push. Она принимает два параметра: имя удаленного репозитория (мы назвали нашу ветку origin) и ветку, в которую необходимо внести изменения (master – это ветка по умолчанию для всех репозиториев).





В большинстве проектов есть файлы или целые директории, в которые мы не хотим (и, скорее всего, не захотим) коммитить. Создадим файл .gitignore и сохраним его в директорию проекта.

И перечислим файлы которые хотим игнорировать.

