**GitHub** - хостинг для

Хостинг GitHub удобен тем, что позволяет студентам получать от преподавателя (и не только от преподавателя) экспертизу выполненных работ. Важно, что при этом не требуется передавать преподавателю на каком либо носителе выполненную работу , достаточно разместить репозиторий с работой на GitHub и попросить преподавателя оставить отзыв.

Кроме того, можно опубликовать частично выполненную работу и попросить помощи в дальнейшей её доработке.

Рассмотрим как сдать работу, выполненную полностью или частично, на проверку преподавателю (или ментору), используя сервис GitHub.

Первым делом, следует зарегистрироваться на GitHub, инструкция с описанием этапов регистрации у вас уже должна быть, а ещё есть интернет, где не только социальные сети, ютуб, тикток и иные развлекательные ресурсы, но и полезные статьи и видео уроки и даже официальный сайт GitHub, где можно найти еще более полную информацию по работе с GitHub и не только с GitHub. А вы, вроде бы, желаете стать программистами, значит нужно учиться искать ответы на возникающие вопросы в интернете.

После того как вы создали аккаунт на GitHub, можно будет приступить к созданию репозиториев для размещения своих работ.

ВАЖНО!

Преподавателю сложно работать с таким большим числом студентов. Чтобы упростить проверку работ, преподавателю важно организовать единообразие на всех этапах выполнения, оформления и сдачи работ, для этого преподаватель выдвигает определённые требования, которым нужно в точности следовать.

**ЕСЛИ ВЫ НЕ СТАНЕТЕ СЛЕДОВАТЬ ТРЕБОВАНИЯМ, ТО РАБОТЫ НЕ БУДУТ ПРОВЕРЯТЬСЯ, ПРЕПОДАВАТЕЛЬ БУДЕТ СЧИТАТЬ, ЧТО ВЫ НЕ ВЫПОЛНИЛИ РАБОТУ!**

**Этап 1. Создание репозитория на GitHub для работ по конкретной дисциплине**

**1.1 Требования к репозиторию**

Для всех работ (и практических и самостоятельных) по одной конкретной дисциплине стоит создать один репозиторий, а не создавать под каждую отдельную работу отдельный репозиторий, чтобы вы и преподаватель не потерялись среди большого числа репозиториев.

Запомните!

Одна дисциплина или курс - один репозиторий!

Название репозитория должно соответствовать шаблону:

<шифр группы **маленькими буквами**>\_<фамилия латиницей **маленькими буквами**>\_<инициалы: имя и отчество латинскими маленькими буквами>\_<название курса латиницей **маленькими символами**>

Пример имён репозиториев:

isp02\_guslev\_mv\_mdk0101

isp01\_guslev\_mv\_mdk0102

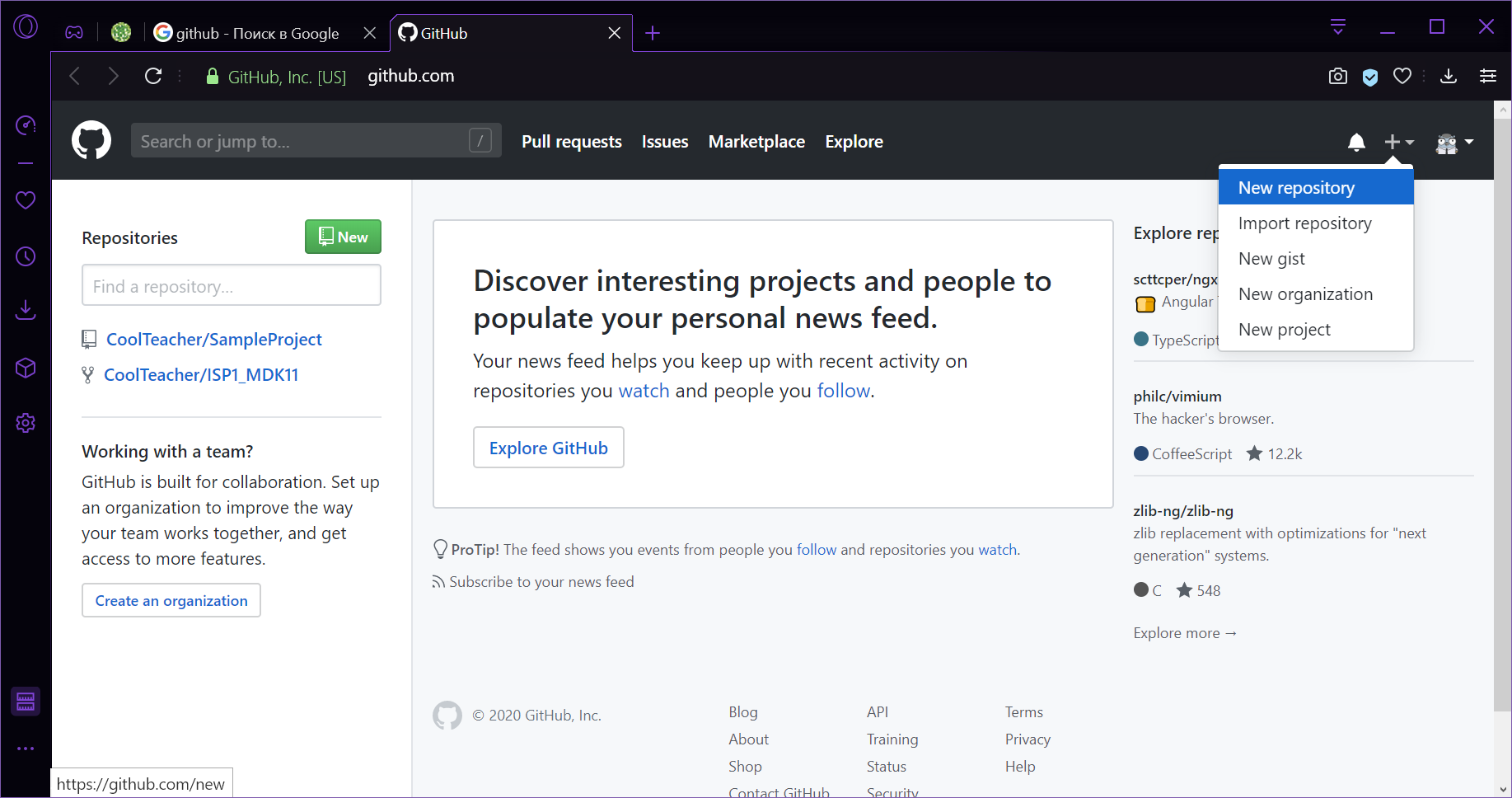
Isp02\_guslev\_mv\_discretemath

Название дисциплин СОГЛАСОВЫВАЙТЕ с преподавателем!

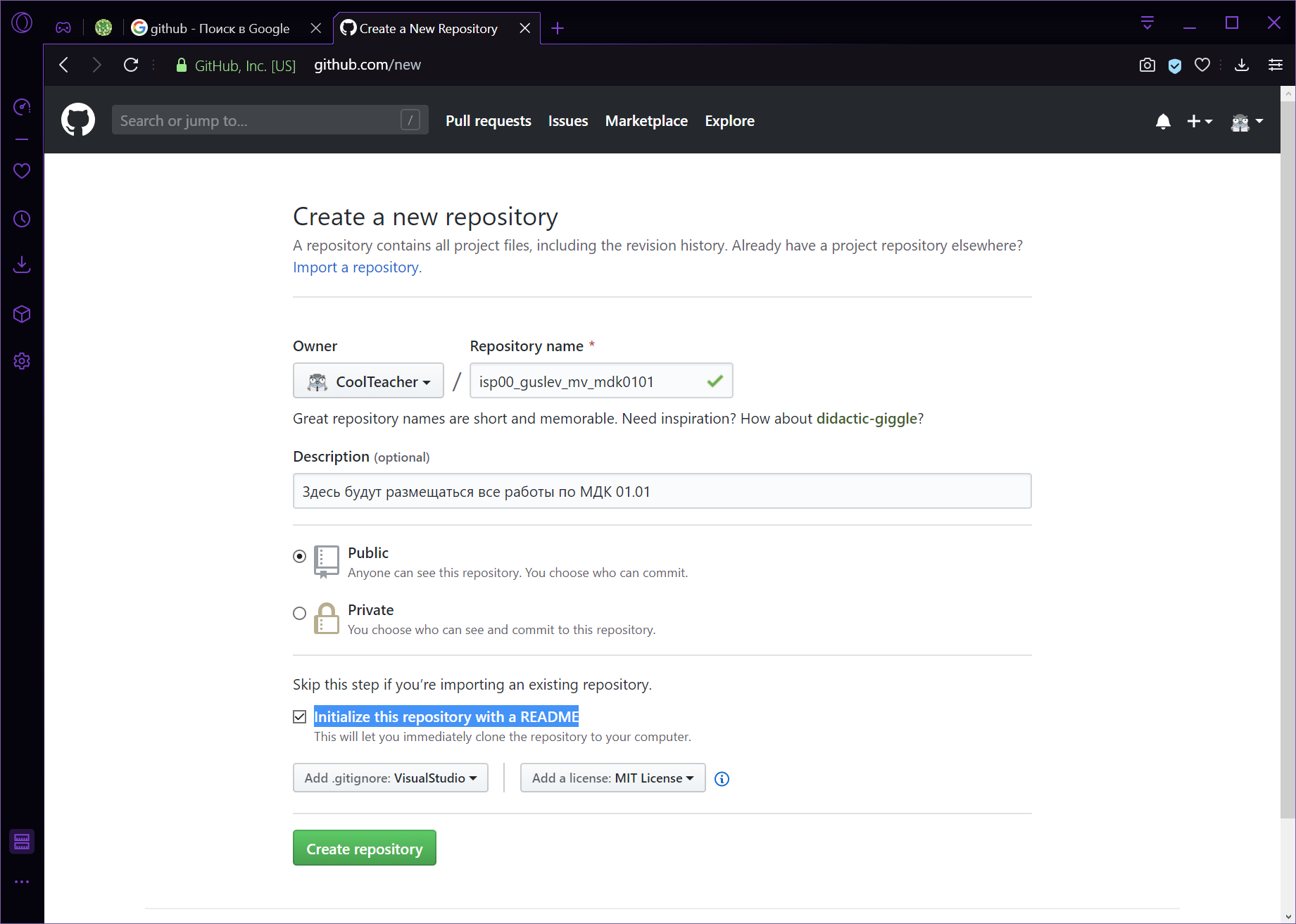
При создании репозитория, обязательно укажите создание файла **.gitignore под Visual Studi**o и обозначьте тот факт, что содержимое вашего репозитория распространяется под **MIT лицензией**.

**1.2 Пример создания репозитория**

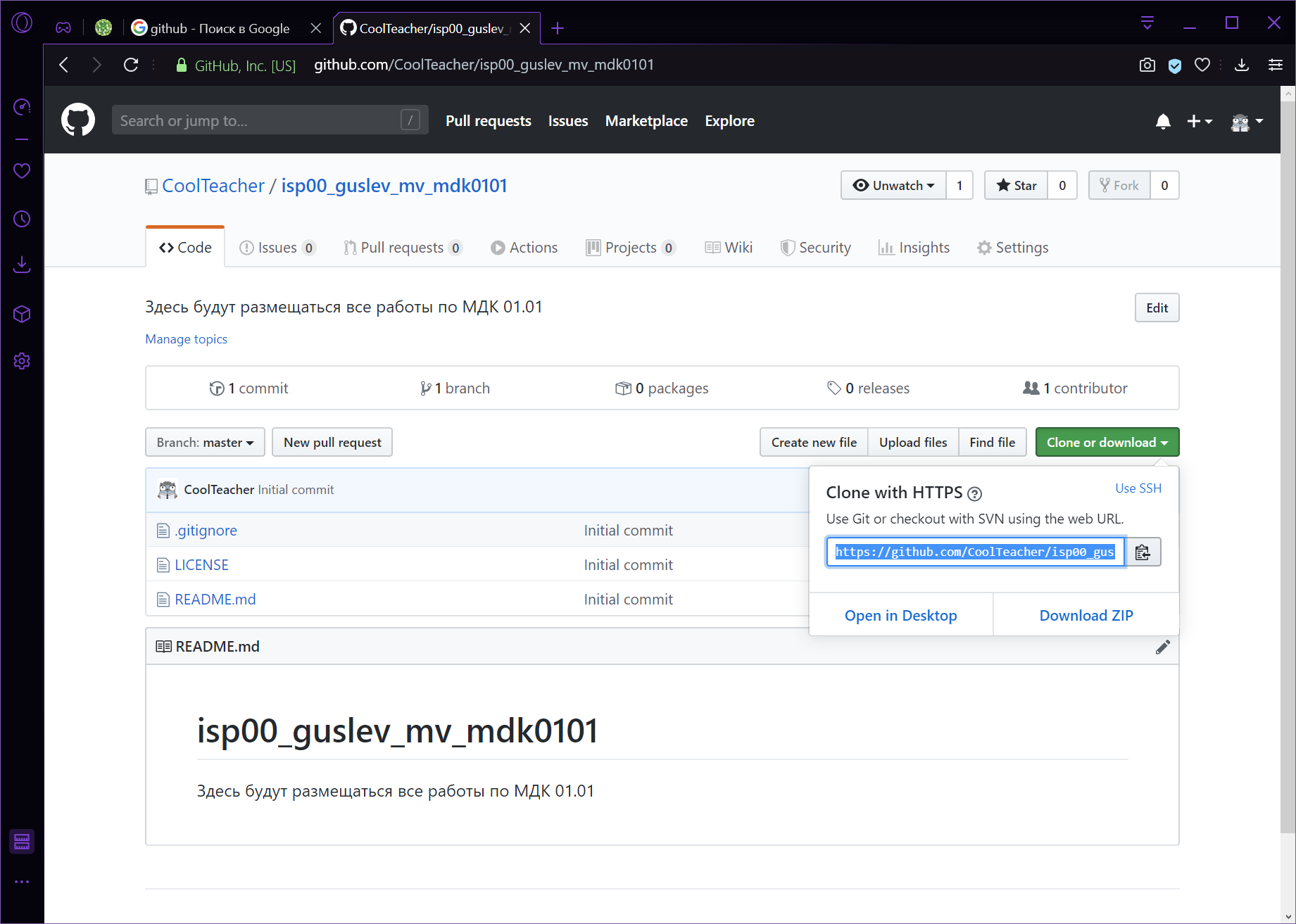
Создание:

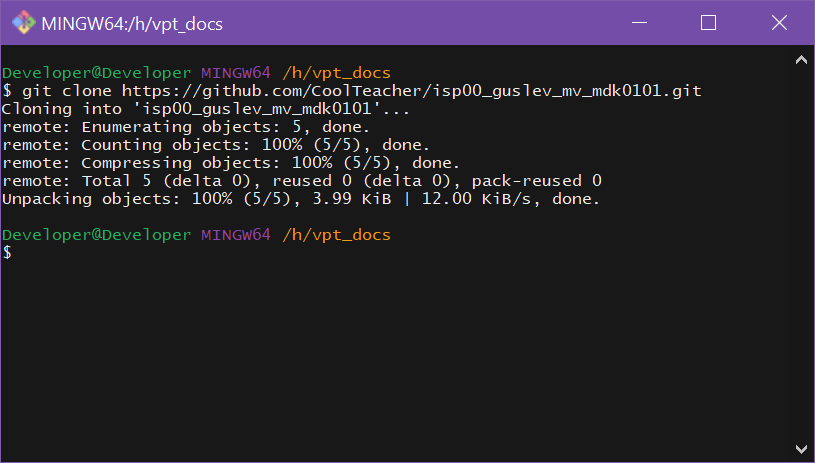


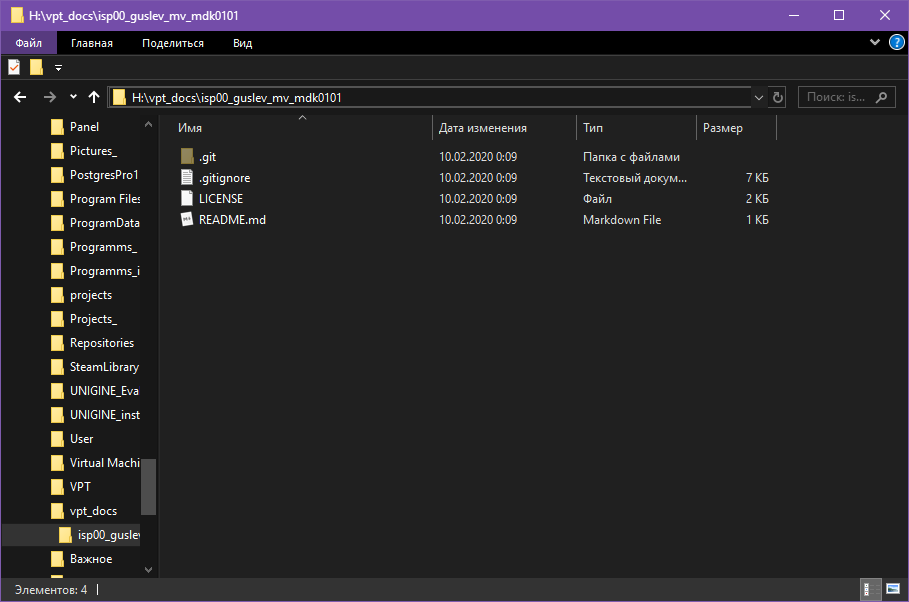
Настройка:



Теперь можно клонировать репозиторий на локальную машину в нужный каталог:







**Этап 2. Организация структуры репозитория**

**2.1 Требования к структуре репозитория**

Внутри рабочего каталога репозитория должны размещаться подкаталоги:

* self\_task - для самостоятельных работ
* practical - для лабораторных и практических работ

Внутри каждого из подкаталогов self\_task иpractical должны размещаться подкаталоги под каждую отдельную работу. Название каталога для работы должен составляться по шаблону:

work\_<номер дня.номер месяца.год **выдачи задания. *Именно выдачи!***>

Внутри каталога для работы должны быть подкаталоги:

* projects - для размещения каталогов и файлов проектов с выполненной работой
* reports - для отчётов по выполненной работе

Пример дерева подкаталогов (и файлов) репозитория:

self\_task

work\_25.01.2020

projects

SampleConsoleApp\_01

SampleConsoleApp\_02

SampleConsoleApp\_03

reports

report.docx

work\_27.01.2020

practical

work\_15.02.2020

work\_17.02.2020

projects

SampleConsoleApp\_01

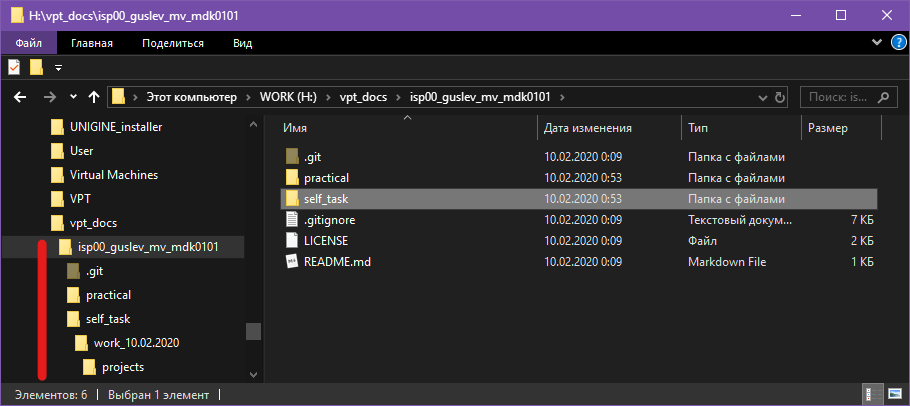
SampleConsoleApp\_02

SampleConsoleApp\_03

reports

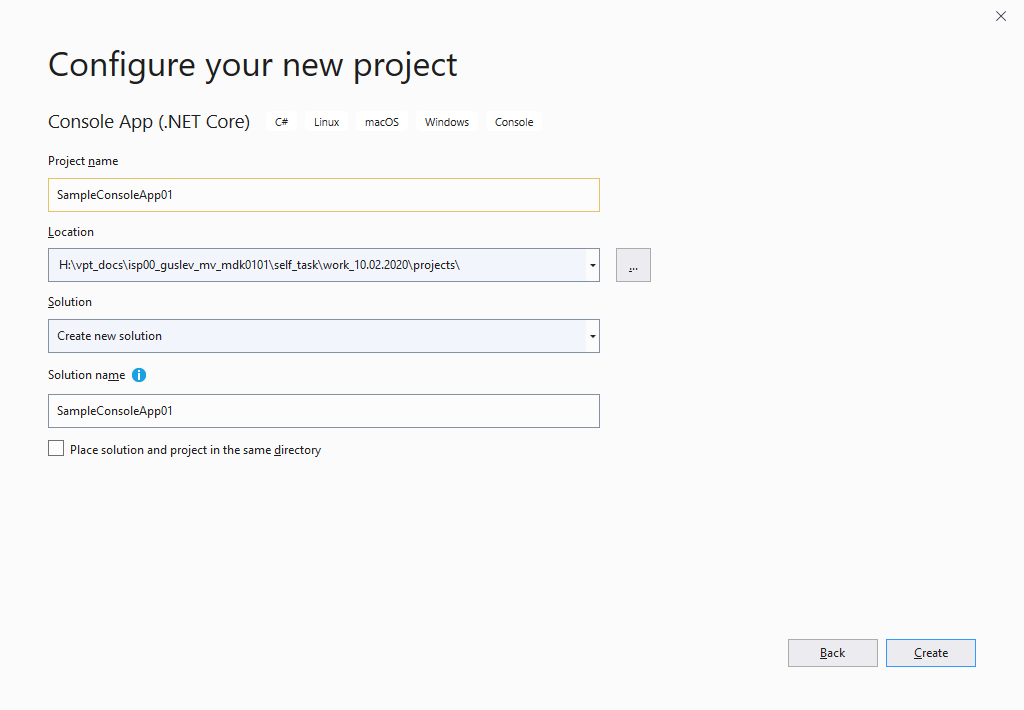
report.docx

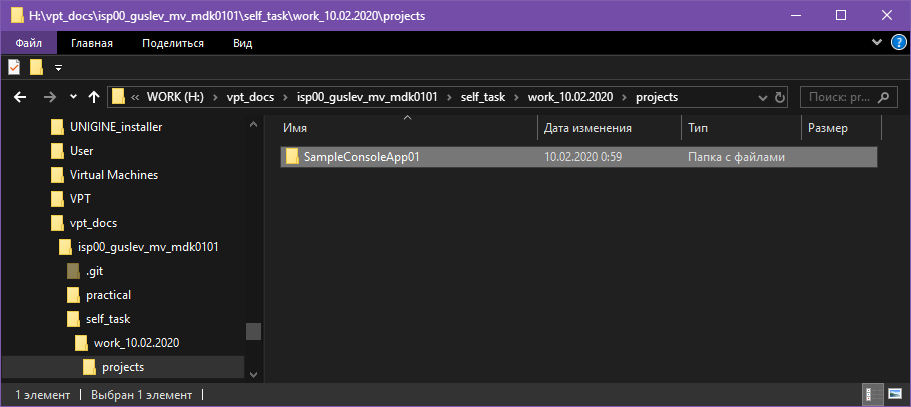
**2.2 Пример организации структуры репозитория**



Создадим для примера проект консольного типа и разместим его в каталоге:

H:\vpt\_docs\**isp00\_guslev\_mv\_mdk0101\self\_task\work\_10.02.2020\projects**



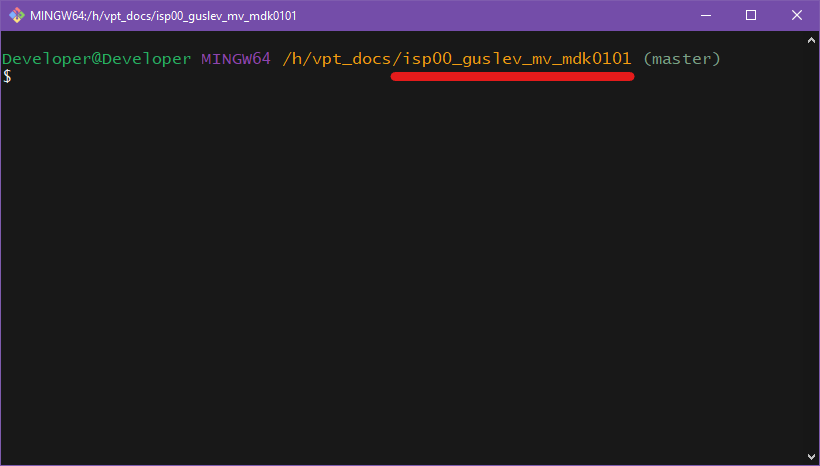


Оставим автоматически сгенерированный код без изменений, сохраним и проект и закроем VS.

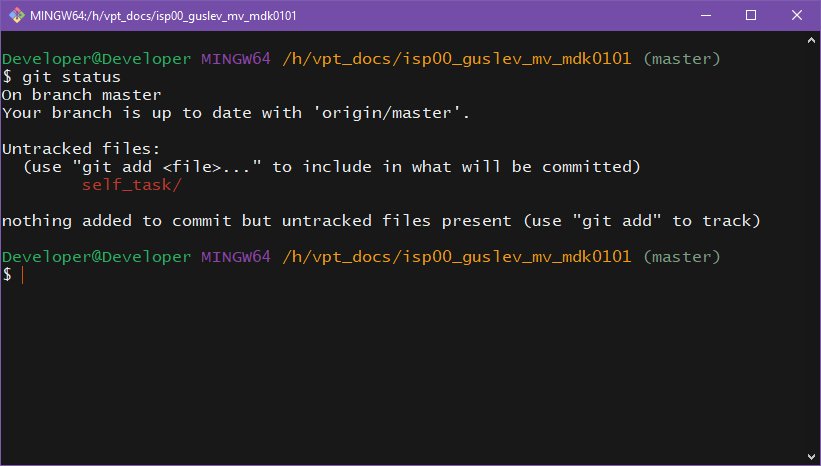
**Этап 3. Отправка изменений репозитория на GitHub (на удалённый репозиторий)**

**3.1 Подготовительный этап**

В GitBush переходим к каталогу репозитория:



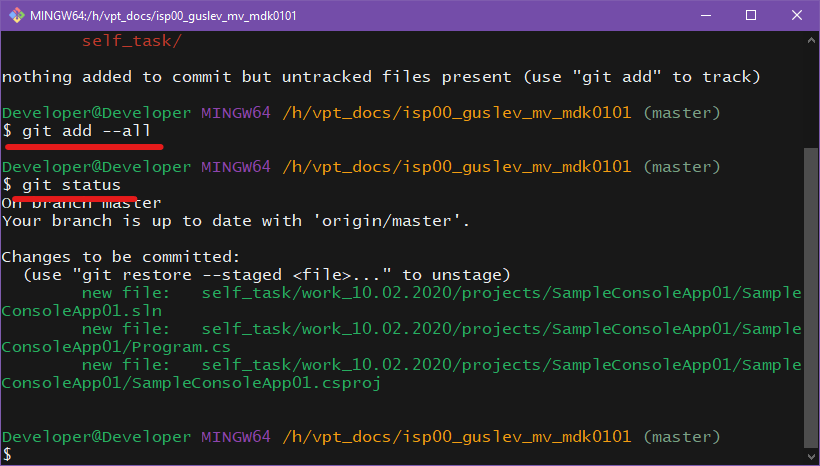
Выполним команду git status для проверки состояния репозитория:



Как показано в примере, появились новые файлы, который не отслеживаются git’отм. Это созданный каталоги и файлы нашего проекта, нужно их добавить под отслеживание git’у.

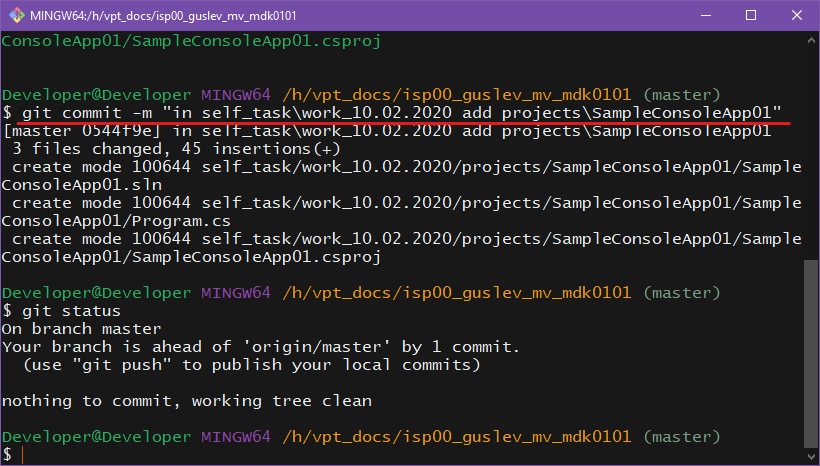
Выполняем команду git add --all для добавления под контроль git’у всех новых файлов ***и индексации как новых, удалённых и модифицированных файлов.***

После выполнения git add --all, выполним ещё раз git status, чтобы проверить результат:



Выполняем git commit -m "комментарий латиницей" для фиксации изменений:

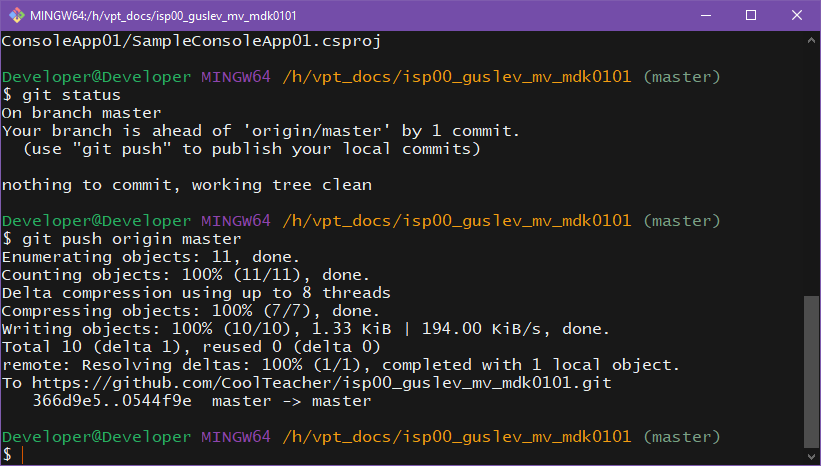
Заметка: комментарий важно задавать осмысленно и латинскими буквами



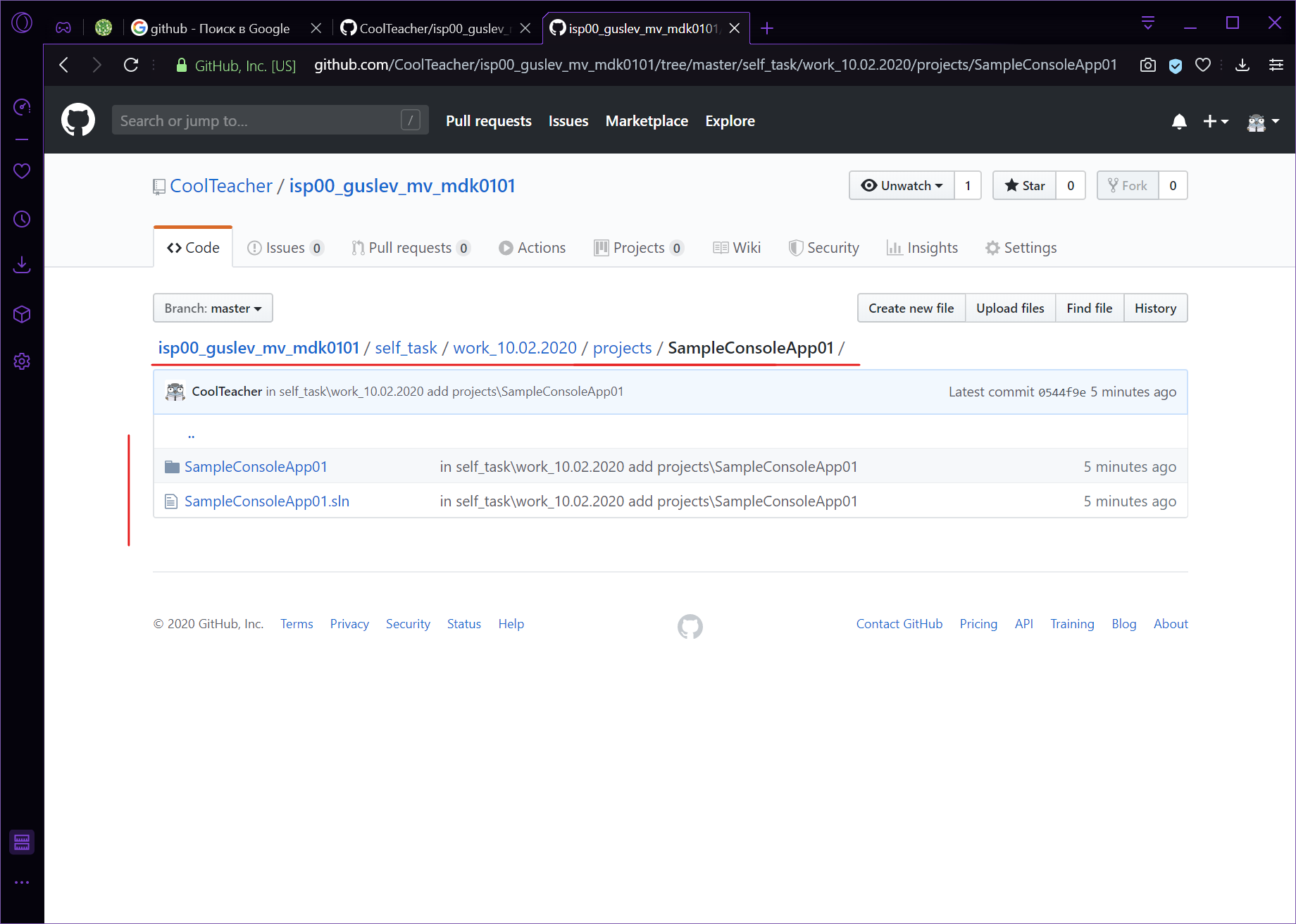
**3.2 Отправка изменений в удалённый репозиторий**

Выполним команду git push origin master для сохранения истории изменений в удалённом репозитории:

Заметка: git потребует ввести имя учётной записи github и пароль



Результат:

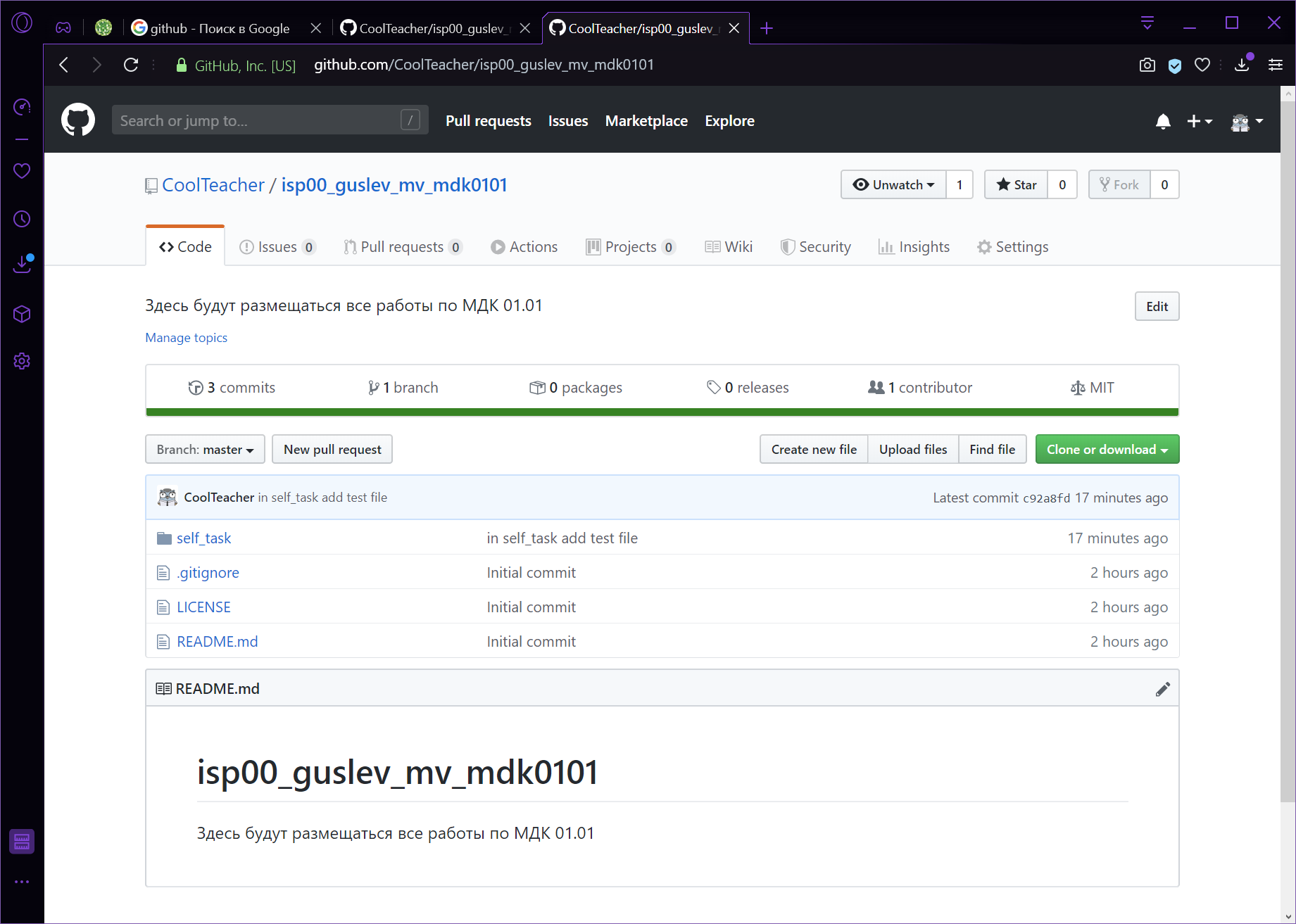


**Этап 4. Запрос ревизии выполненной работы**

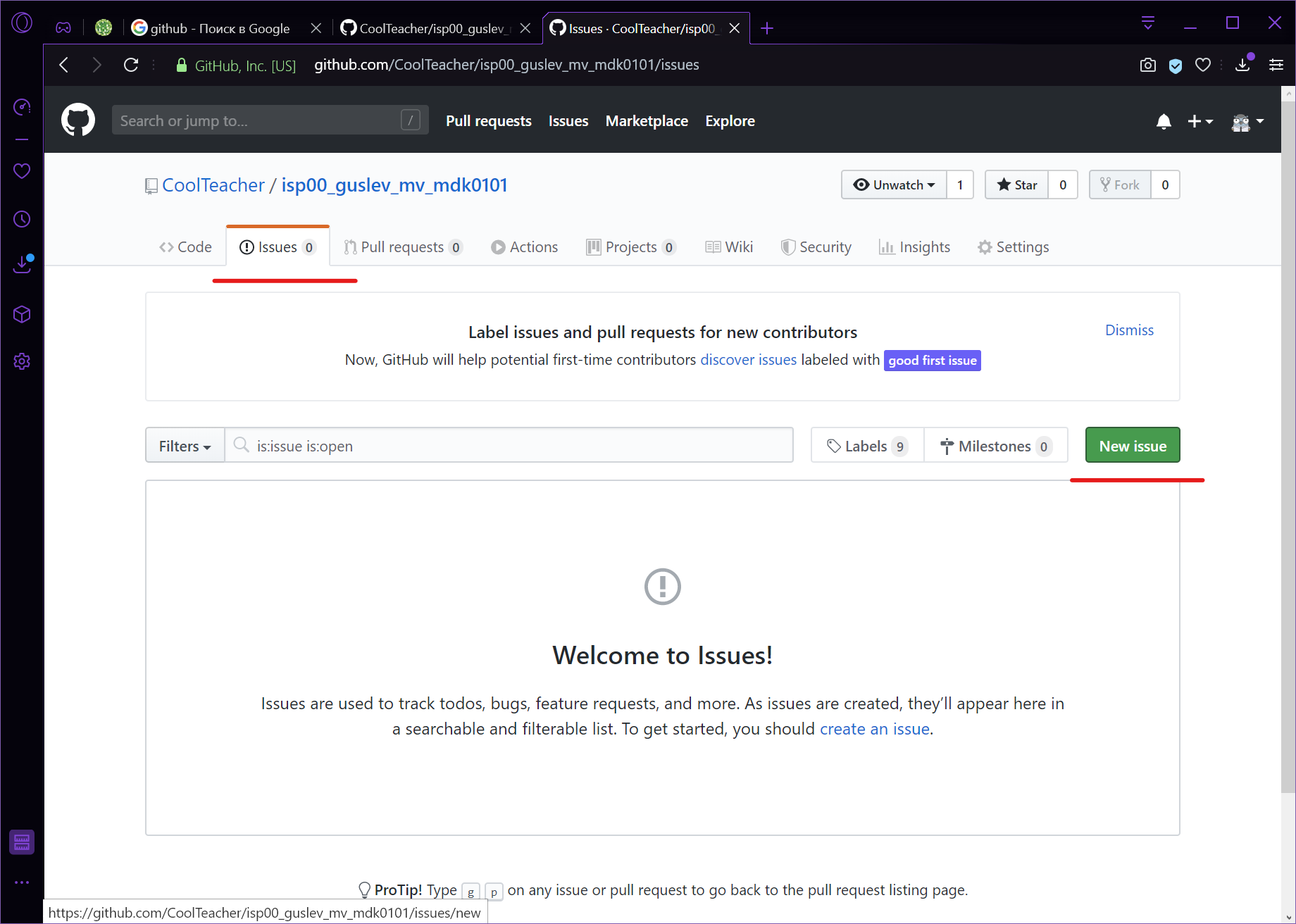
**4.1 Создание issue сообщения (сообщения об ошибках)**

**Вариант 1. Когда нужна проверка участка кода или файла с кодом**

В репозитории на GitHub переходим к репозиторию:



Открываем вкладку «issues» и жмём кнопку «new issue»:

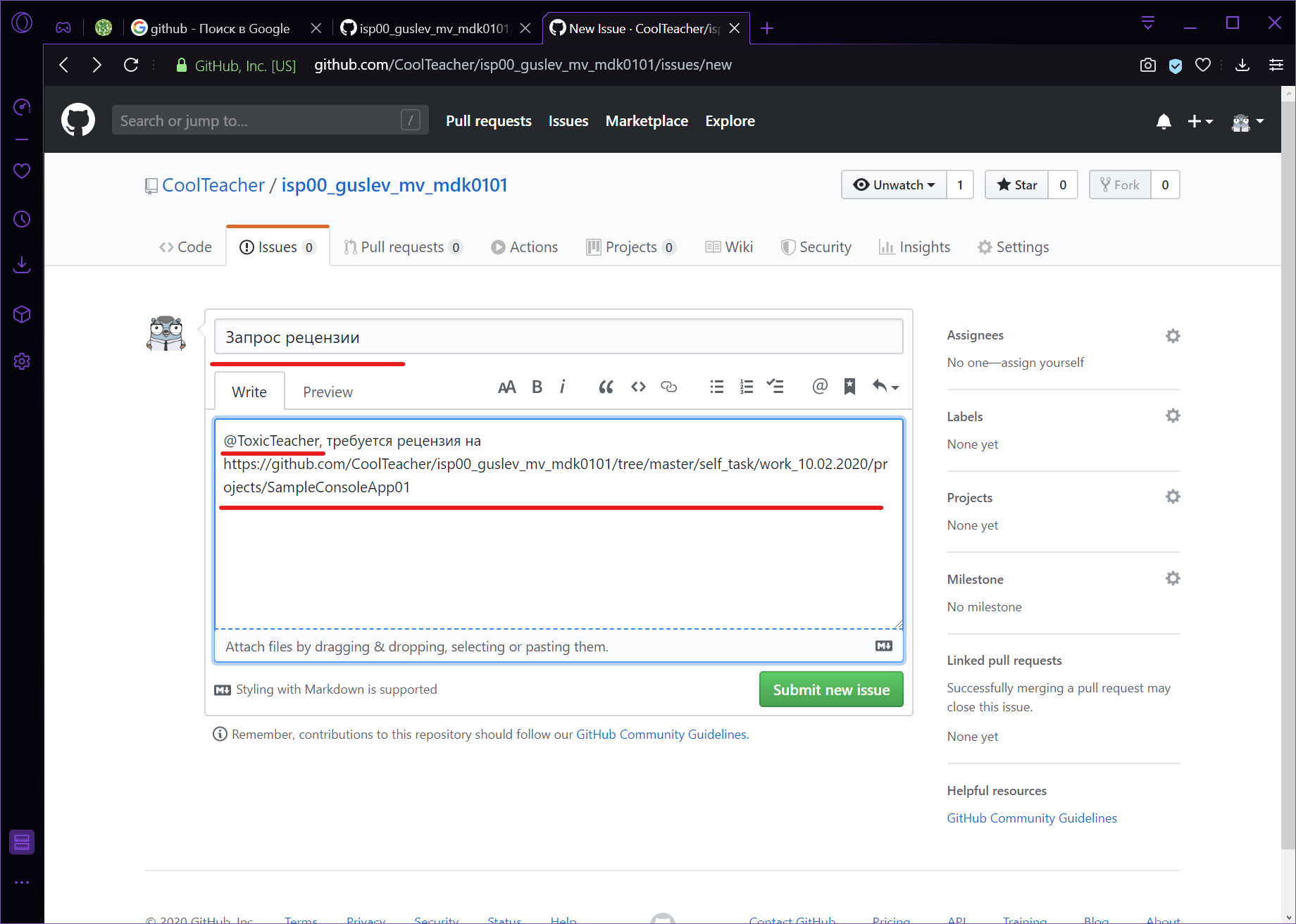


Указываем в заголовке «Запрос рецензии».

В описании указываем:

* **@ToxicTeacher** - адресат запроса. Где @ - начало запроса, ToxicTeacher - GitHub аккаунт преподавателя (да! Именно такой у меня аккаунт!)
* **ссылку** к файлу или каталогу проекта который нужно проверить

Пример:



Нажимаем кнопку «Submit new issue» для подтверждения создания issue сообщения

Ждём ответа от преподавателя.

**ВАЖНО!**

Как получить из удалённого репозитория в ранее клонированный локальный все изменения:

В GitBush перейти в каталог локального репозитория и выполнить:

git pull origin master

Это важно когда вы сделали выполнили работу на рабочем ПК в техникуме и залили на удалённый репозиторий и прейдя домой желаете получить все измнения в локальный репозиторий на домашнем ПК (чтобы всё с нуля не клонировать).

И наоборот, дома сделали работу, залили на удалённый репозиторий, пришли в техникум и выполнили эту команду.

Другой подход. Можно ***(только на первых этапах работы с git, пока не освоитесь)*** после заливки изменений на удалённый репозиторий выполнять удаление локального, а после клонировать его снова, когда он потребуется.

Но такое делать хорошему спецу нельзя! Лучше гуглите, если возникают проблемы.