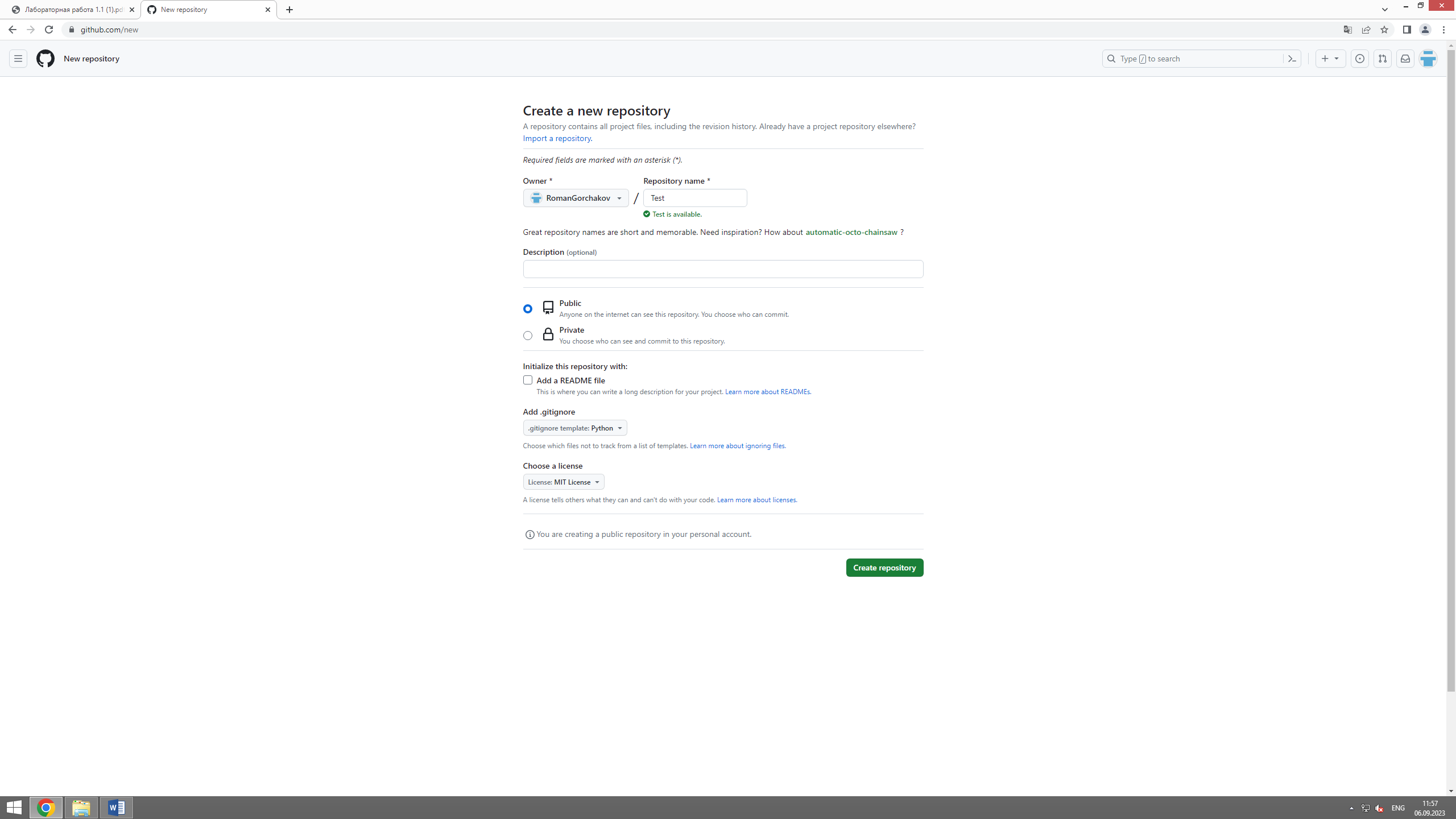
**Лабораторная работа 1.3**

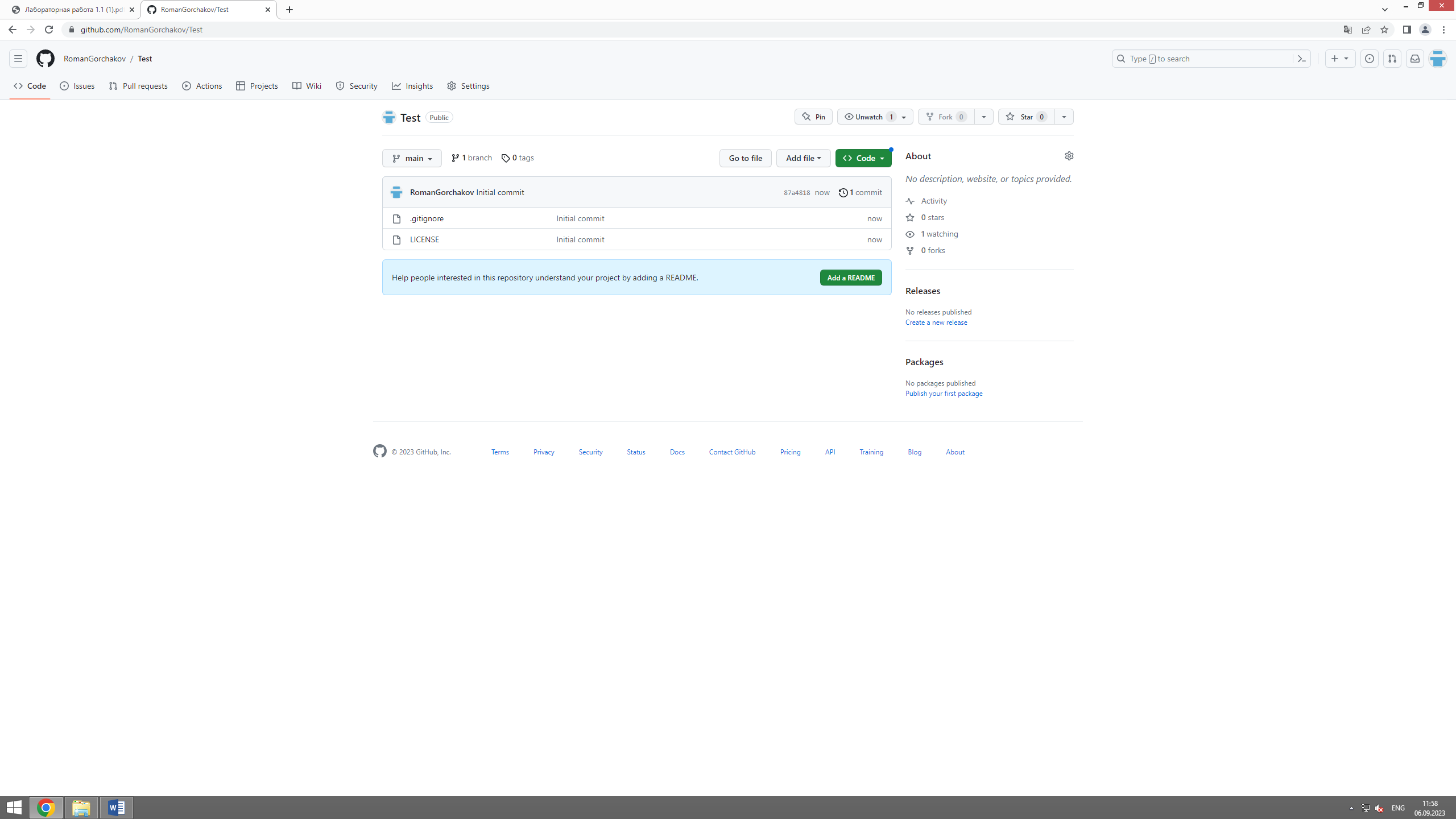
Тема: Основы вставления Git.

Цель работы: исследование базовых возможностей по работе с локальными и удаленными ветками Git.

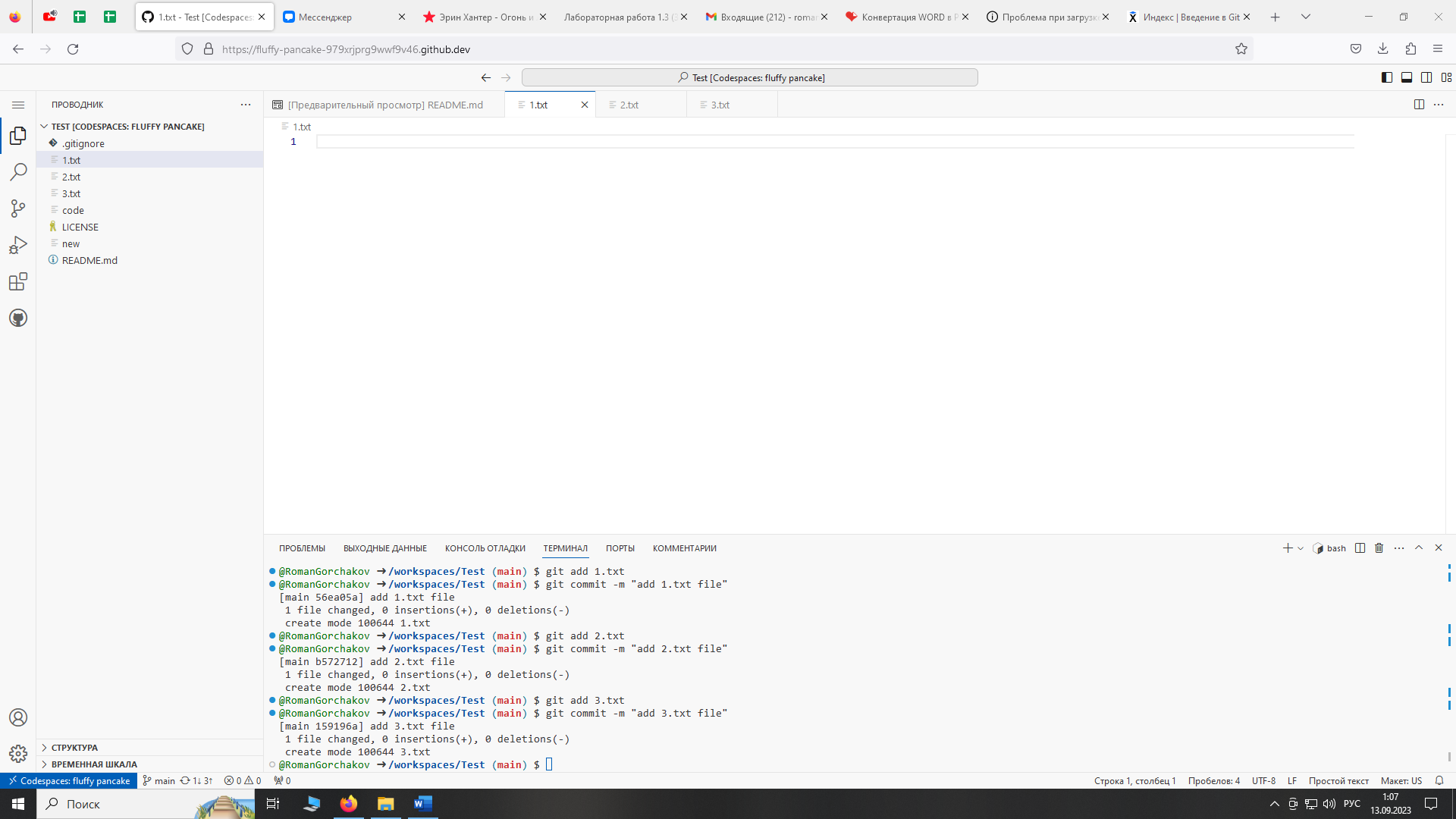
Порядок выполнения работы

1. Создаём аккаунт в GitHub. Затем создаём новый общедоступный репозиторий, в котором будет использована лицензия MIT и язык программирования Python.

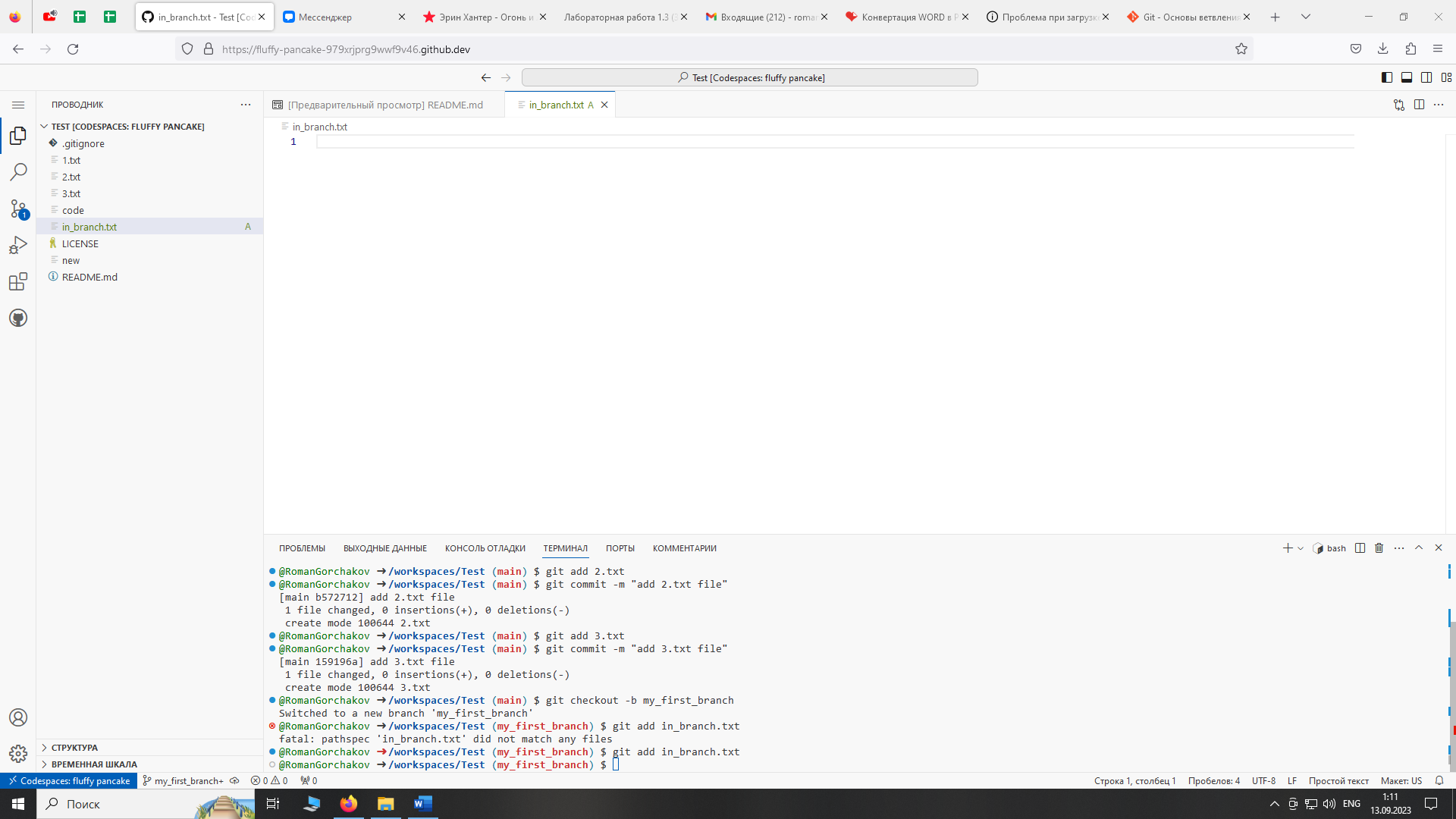




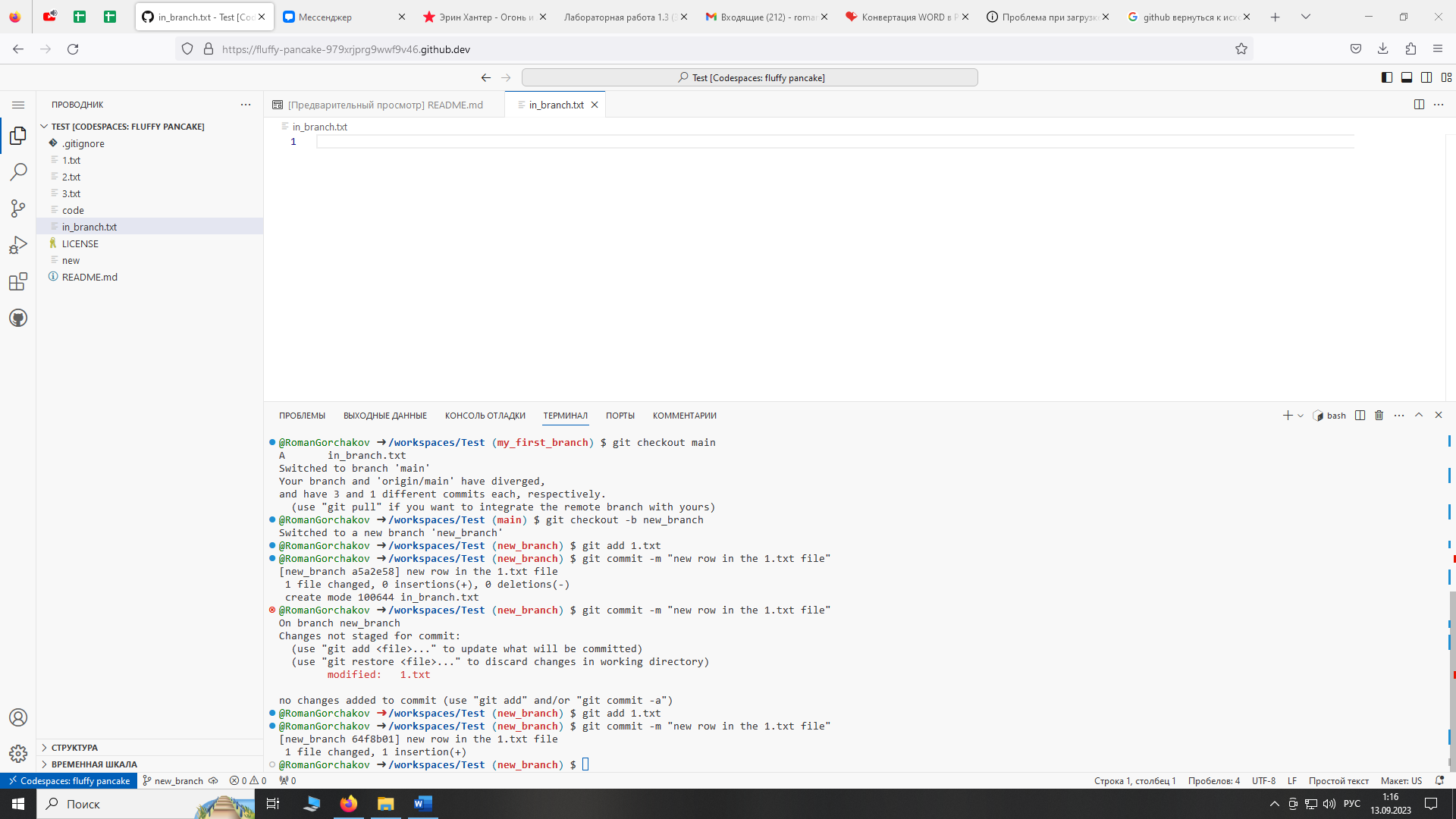
1. Создаём три файла: «1.txt», «2.txt» и «3.txt». Проиндексируем каждый файл через команды «git add» и делаем коммиты с комментариями с помощью команд «git commit -m».



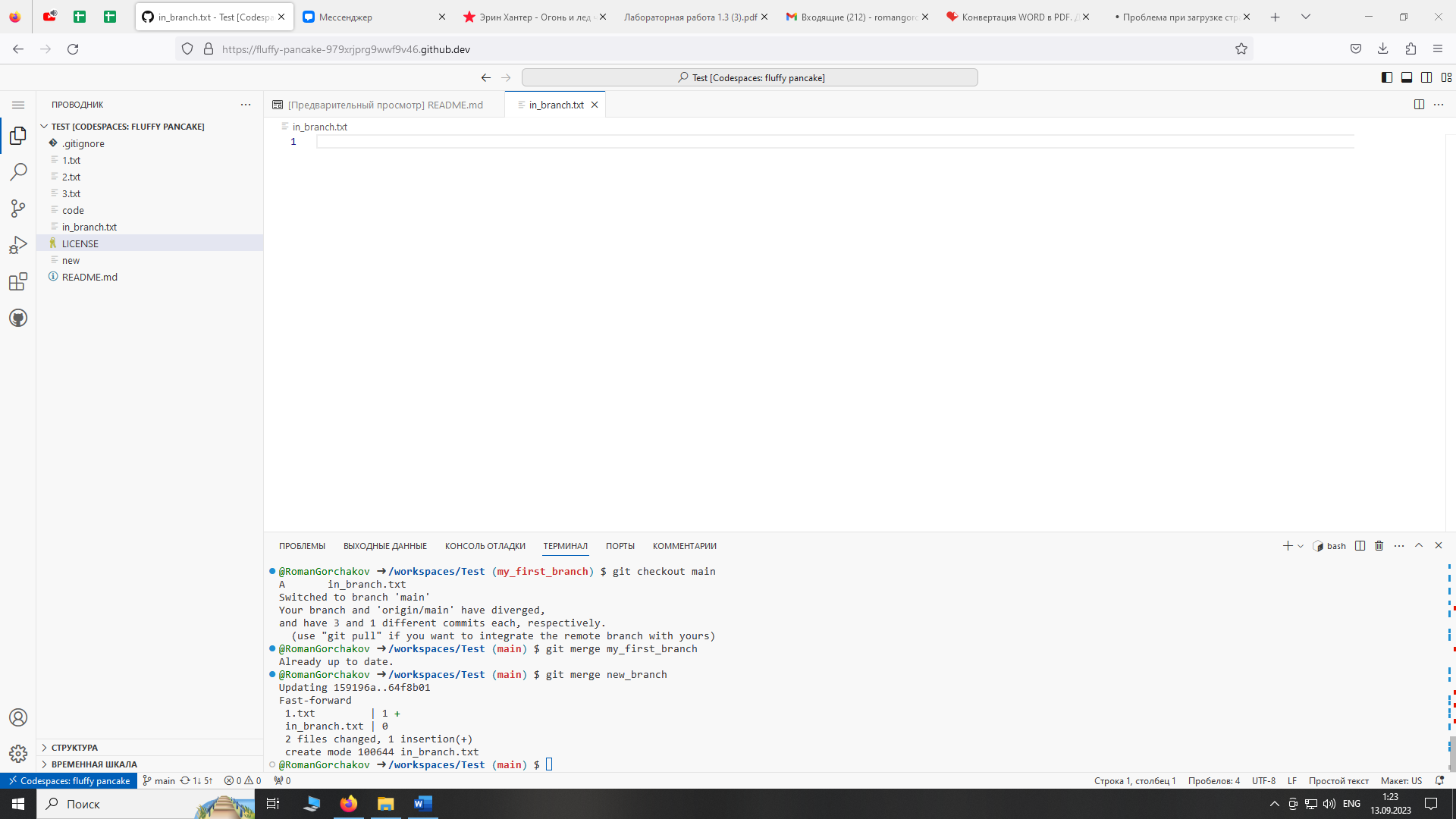
1. Создаём новую ветку «my\_first\_branch» с помощью команды «git checkout -b». Переходим на неё и создаём на ней новый файл «in\_branch.txt».



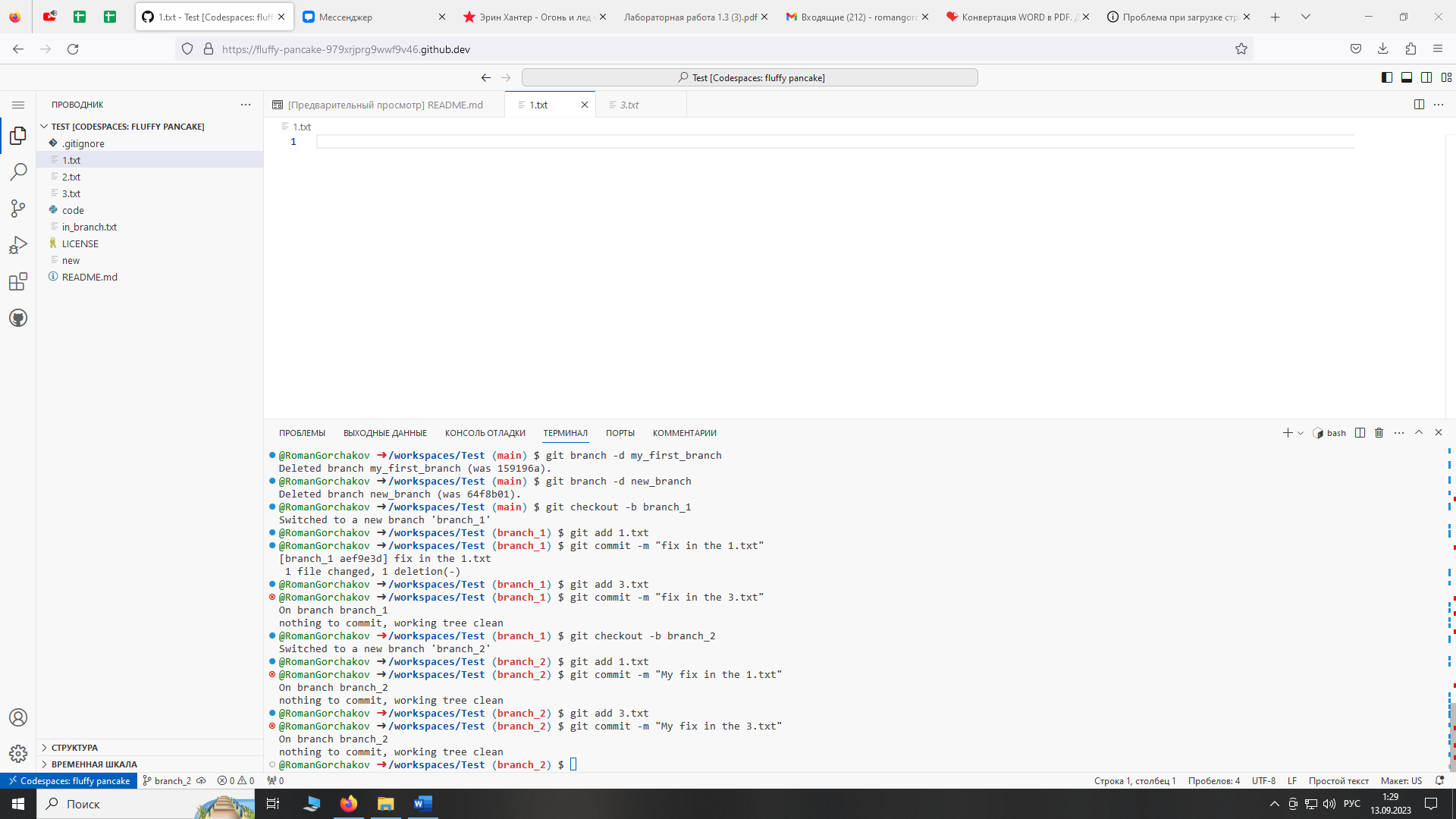
1. Возвращаемся на основную ветку «main» с помощью команды «git checkout main». Создаём новую ветку «new\_branch» с помощью команды «git checkout -b». Сохраняем изменения с помощью команды «git commit -m».



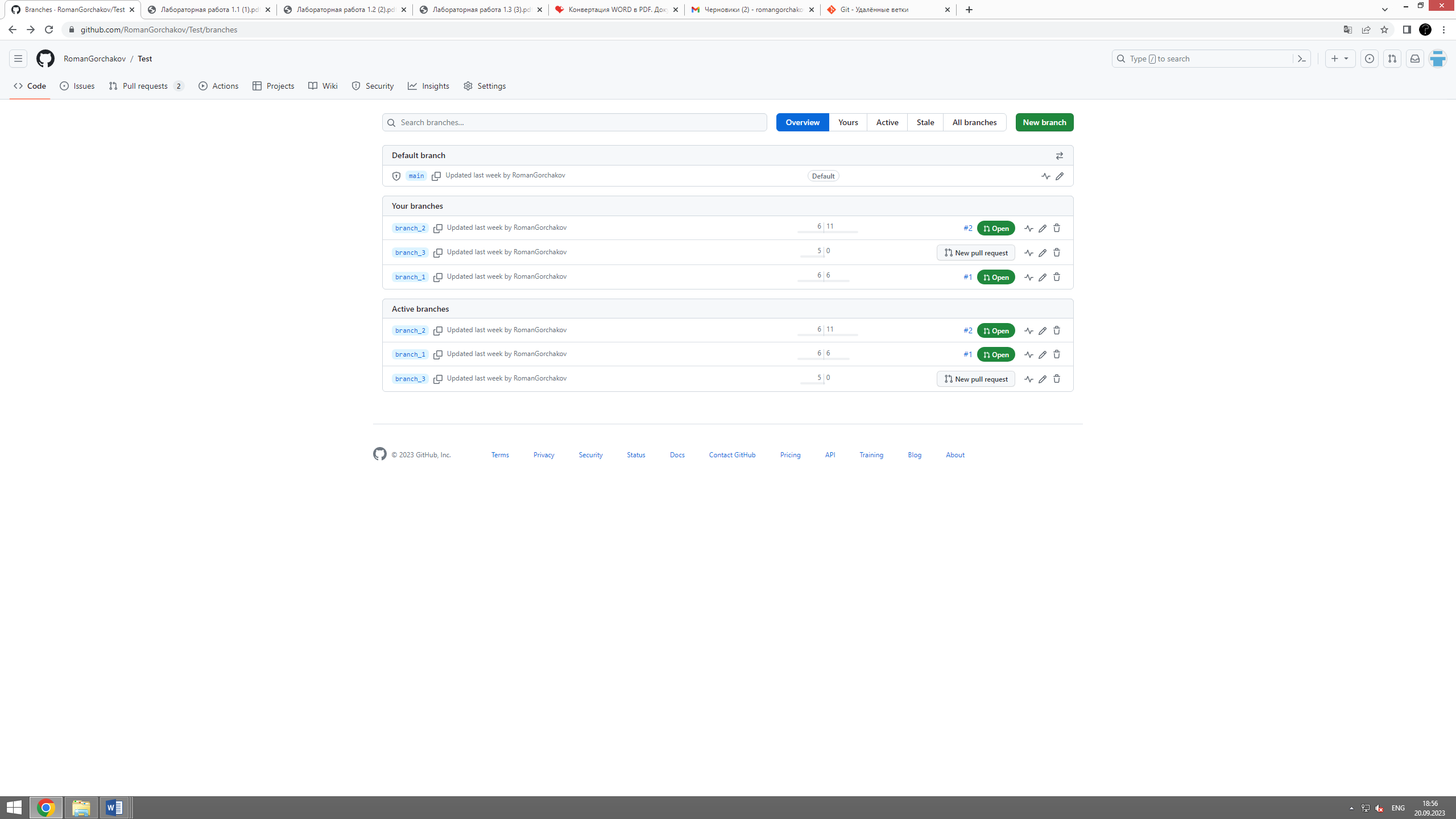
1. Возвращаемся на основную ветку «main» с помощью команды «git checkout main». После этого сливаем ветки «main» и «my\_first\_branch», а затем «main» и «new\_branch» с помощью команды «git merge».

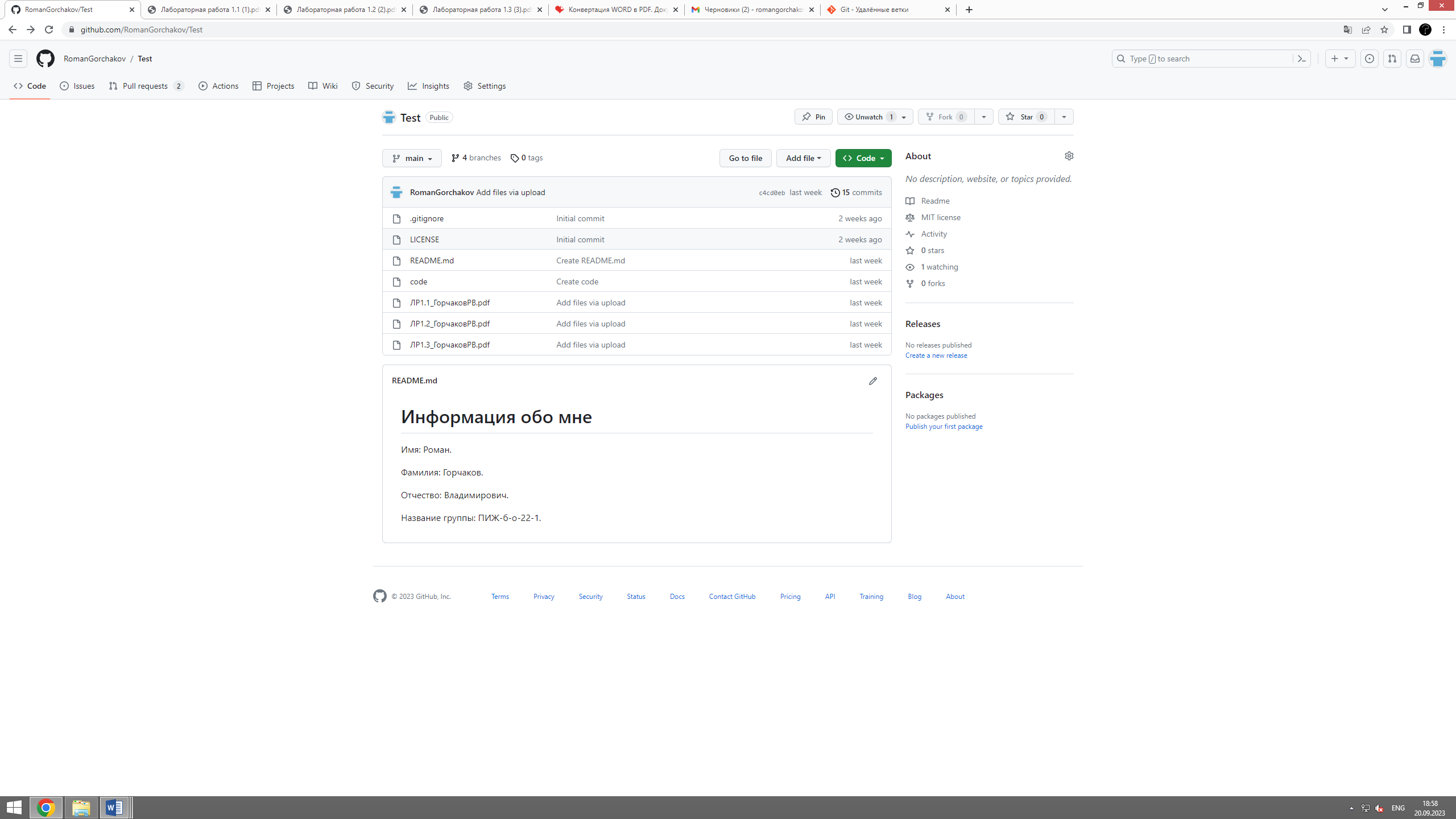


1. Удаляем ветки «my\_first\_branch» и «new\_branch» через команду «git branch -d». Создаём ветки «branch\_1» и «branch\_2». Переходим на ветку «branch\_1», удалим содержимое файлов «1.txt» и «3.txt» и сохраняем изменения с помощью команды «git commit -m». Делаем то же самое в ветке «branch\_2».



1. Сливаем изменения ветки «branch\_2» в «branch\_1» с помощью команды «git merge». Отправляем ветку «branch\_1» на GitHub с помощью команды «git push». Создаём удалённую ветку «branch\_3» с помощью команды «git push origin». Создаём в локальном репозитории ветку отслеживания «searcher» удалённой ветки «branch\_3» с помощью команды «git remote add». Переходим на ветку «branch\_3» через «git checkout» и добавляем в файл «2.txt» строку «the final fantasy in the 4.txt file». Сохраняем изменения через «git commit -m». Переносим ветку «branch\_2» на ветку «main» через «git rebase» и обе ветки – на удалённый репозиторий в GitHub с помощью команды «git push».





Контрольные вопросы

1. Что такое ветка?

Ветка в Git – простой перемещаемый указатель на один из таких коммитов.

1. Что такое HEAD?

HEAD – указатель, задача которого ссылаться на определенный коммит в репозитории.

1. Способы создания веток.

Создание веток происходит с помощью команды «git branch».

1. Как узнать текущую ветку?

Текущую ветку можно узнать с помощью команды «git log –oneline –decorate».

1. Как переключаться между ветками?

Между ветками можно переключаться с помощью команды «git checkout».

1. Что такое удалённая ветка?

Удалённые ссылки – ссылки (указатели) в ваших удалённых репозиториях, включая ветки, теги и так далее.

1. Что такое ветка отслеживания?

Ветки слежения – ссылки на определённое состояние удалённых веток. Это локальные ветки, которые нельзя перемещать; Git перемещает их автоматически при любой коммуникации с удаленным репозиторием, чтобы гарантировать точное соответствие с ним.

1. Как создать ветку отслеживания?

Ветку слежения можно создать с помощью команды «git remote add».

1. Как отправить изменения из локальной ветки в удалённую ветку?

Отправить изменения из локальной ветки в удалённую можно с помощью команды «git push».

1. В чем отличие команд git fetch и git pull?

Команда «git pull» автоматически сливает коммиты, не давая пользователю просмотреть их сразу. При использовании «git fetch» Git собирает все коммиты из целевой ветки, которых нет в текущей ветке, и сохраняет их в локальном репозитории.

1. Как удалить локальную и удалённую ветки?

Удалить локальную и удалённую ветки можно с помощью команды «git push --delete».

1. Изучить модель ветвления git-flow (использовать материалы статей https://www.atlassian.com/ru/git/tutorials/comparing-workflows/gitflow-workflow, https://habr.com/ru/post/106912/). Какие основные типы веток присутствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?

Git-flow – устаревшая версия рабочего процесса Git, в свое время ставшая принципиально новой стратегией управления ветками в Git.

Основные типы веток: главные, вспомогательные, функциональные, ветви релизов, ветви исправлений

Первый шаг рабочего процесса заключается в создании ветки develop от стандартной ветки main. Под каждую новую функцию нужно выделить собственную ветку, которую можно отправить в центральный репозиторий для создания резервной копии или совместной работы команды. Когда в ветке develop оказывается достаточно функций для выпуска (или приближается назначенная дата релиза), от ветки develop создается ветка release. Создание этой ветки запускает следующий цикл релиза, и с этого момента новые функции добавить больше нельзя – допускается лишь исправление багов, создание документации и решение других задач, связанных с релизом.

Git Flow может замедлять работу, когда приходится ревьювить большие пулл реквесты, когда вы пытаетесь выполнить итерацию быстро. Релизы сложно делать чаще, чем раз в неделю. Большие функции могут потратить дни на мерж и резолв конфликтов и форсировать несколько циклов тестирования.

1. На прошлой лабораторной работе было задание выбрать одно из программных средств с GUI для работы с Git. Необходимо в рамках этого вопроса привести описание инструментов для работы с ветками Git, предоставляемых этим средством.