

Лабораторная работа №1

Работа с git

Шияпова Дарина Илдаровна

Содержание

1 Цель работы	5
2 Теоретическое введение	6
3 Выполнение лабораторной работы	7
3.1 Подготовка	7
3.2 Создание проекта	7
3.3 Внесение изменений	8
3.4 История	9
3.5 Получение старых версий	10
3.6 Создание тегов версий	10
3.7 Отмена локальных изменений (до индексации)	11
3.8 Отмена проиндексированных изменений (перед коммитом)	12
3.9 Отмена коммитов	13
3.10 Удаление коммиттов из ветки	13
3.11 Удаление тега oops	14
3.12 Изменение предыдущего коммита	14
3.13 Перемещение файлов	15
3.14 Подробнее о структуре	15
3.15 Git внутри: Каталог .git	16
3.16 Работа непосредственно с объектами git	17
3.17 Создание ветки	18
3.18 Навигация по веткам	18
3.19 Изменения в ветке master	19
3.20 Слияние	19
3.21 Создание конфликта	20
3.22 Разрешение конфликтов	20
3.23 Сброс ветки style	21
3.24 Сброс ветки master	21
3.25 Перебазирование	21
3.26 Слияние в ветку master	22
3.27 Клонирование репозиториев	22
3.28 Что такое origin?	22
3.29 Удаленные ветки	23
3.30 Изменение оригинального репозитория	23
3.31 Слияние извлеченных изменений	24
3.32 Добавление ветки наблюдения	24

3.33 Создание чистого репозитория	25
3.34 Отправка и извлечение изменений	25
4 Выводы	27
Список литературы	28

Список иллюстраций

3.1	Настройка git	7
3.2	Создание репозитория	8
3.3	Добавление файла в репозиторий	8
3.4	Внесение изменений в содержимое репозитория	9
3.5	Просмотр истории	10
3.6	Просмотр разных версий репозитория	10
3.7	Создание тегов версий	11
3.8	Переключение по имени тега и просмотр доступных тегов	11
3.9	Отмена локальных изменений (до индексации)	12
3.10	Отмена коммитов	13
3.11	Удаление коммиттов из ветки	14
3.12	Удаление тега oops	14
3.13	Изменение предыдущего коммита	15
3.14	Результат открытия index.html	16
3.15	Каталог .git	17
3.16	Работа непосредственно с объектами git	17
3.17	Создание ветки	18
3.18	Просмотр логов новой ветки	18
3.19	Возвращение к ветке style	19
3.20	Изменения в ветке master	19
3.21	Слияние веток	20
3.22	Клонирование репозиториев	22
3.23	Просмотр имени по умолчанию удаленного репозитория	23
3.24	Изменение оригинального репозитория	23
3.25	Извлечение изменений	24
3.26	Слияние извлеченных изменений	24
3.27	Добавление ветки наблюдения	25
3.28	Создание чистого репозитория	25
3.29	Отправка изменений	26

1 Цель работы

Приобрести практические навыки работы с системой управления версиями Git.

2 Теоретическое введение

Git – распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года; координатор – Дзюн Хамано [[wiki:bash?](#)].

3 Выполнение лабораторной работы

3.1 Подготовка

Сначала настроим core.autocrlf с параметрами true и input, чтобы сделать все переводы строк текстовых файлов в главном репозитории одинаковыми, а затем настроим отображение unicode(рис. fig. 3.1).

```
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> git config --global user.name "daarinaaa"
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> git config --global user.email "darina2436@yandex.ru"
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> git config --global core.autocrlf true
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> git config --global core.safecrlf true
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> git config --global core.quotePath off
```

Рис. 3.1: Настройка git

3.2 Создание проекта

Создадим пустой каталог hello, а в нём файл с именем hello.html. Затем создадим git репозиторий из этого каталога, выполнив команду git init. Добавим файл в репозиторий и проверим статус, который сообщает, что коммитить нечего(рис. fig. 3.2-fнг. 3.3).

```
● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> mkdir hello

Каталог: C:\Users\ASUS\Documents\University\6 semestr\MatMod

Mode          LastWriteTime    Length Name
----          -----          ----- Name
d----       18.06.2025      23:29          hello

PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> cd hello
```

Рис. 3.2: Создание репозитория

```
● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> git add .
● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html
```

Рис. 3.3: Добавление файла в репозиторий

3.3 Внесение изменений

Изменим содержимое файла hello.html на:

```
<h1>Hello, World!</h1>
```

Проверив состояние рабочего каталога увидим, что git знает, что файл hello.html был изменен, но при этом эти изменения еще не зафиксированы в репозитории. Теперь проиндексируем изменения и снова посмотрим статус, в нём указано, что изменения пока не записаны в резервный репозиторий. И наконец закоммитим изменения, внеся их в репозиторий и снова посмотрим статус, который теперь показывает, что все изменения внесены в репозиторий (рис. fig. 3.4).

```
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git add hello.html
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html

PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git commit
hint: Waiting for your editor to close the file...
Aborting commit due to empty commit message.
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git
[master 895073e] Added h1 tag
  1 file changed, 0 insertions(+), 0 deletions(-)
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git
On branch master
nothing to commit, working tree clean
```

Рис. 3.4: Внесение изменений в содержимое репозитория

Изменим страницу «Hello, World», чтобы она содержала стандартные теги и

```
.
```

```
<html>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Теперь добавим это изменение в индекс git и добавим заголовки HTML (секцию) к странице «Hello, World». Проверив текущий статус увидим, что hello.html указан дважды в состоянии. Первое изменение (добавление стандартных тегов) проиндексировано и готово к коммиту. Второе изменение (добавление заголовков HTML) является непроиндексированным. Произведем коммит проиндексированного изменения, затем проиндексируем оставшееся изменение, посмотрим статус и прокоммитим его.

3.4 История

Получим список произведённых изменений в стандартном виде, затем в односрочном, а также с указанием времени и количества(рис. fig. 3.5).

```

commit d40d0c8ee1511a16708b85cf95f6142d46c851fd (HEAD -> master)
Author: daarinaaa <darina2436@yandex.ru>
Date: Thu Jun 19 00:01:14 2025 +0300

    Added standard HTML page tags

commit 895073e3ed6f23b2112c0a57c1bc23ca3a29593d
Author: daarinaaa <darina2436@yandex.ru>
Date: Wed Jun 18 23:51:30 2025 +0300

    Added h1 tag

commit 7fdd1dc7a79e7df7391e59a72fae546972a4199d
Author: daarinaaa <darina2436@yandex.ru>
Date: Wed Jun 18 23:41:53 2025 +0300

    Initial Commit

```

Рис. 3.5: Просмотр истории

3.5 Получение старых версий

Изучим данные лога и найдем там хэш первого коммита, используя его вернемся к первой версии и просмотрим файл hello.html, действительно, увидим первую версию. Затем вернемся к последней версии в ветке master и вновь просмотрим на файл(рис. fig. 3.6).

```

PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git checkout 7fdd1dc7a79e7df7391e59a72fae546972a4199d
Note: switching to '7fdd1dc7a79e7df7391e59a72fae546972a4199d'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

git switch -c <new-branch-name>

```

Рис. 3.6: Просмотр разных версий репозитория

3.6 Создание тегов версий

Назовем текущую версию страницы hello первой (v1). Создадим тег первой версии и используем его для того чтобы венуться к предыдущей, которой также присвоим тег(рис. fig. 3.7).

```

● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git tag v1
● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git checkout v1^
Note: switching to 'v1^'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHe

```

Рис. 3.7: Создание тегов версий

Переключимся по тегам между двумя отмеченными версиями. Просмотрим все доступные теги(их два) и посмотрим теги в логе(рис. fig. 3.8).

```

commit 895073e3ed6f23b2112c0a57c1bc23ca3a29593d (HEAD, tag: v1-beta)
Author: daarinaaa <darina2436@yandex.ru>
Date:   Wed Jun 18 23:51:30 2025 +0300

    Added h1 tag

commit 7fdd1dc7a79e7df7391e59a72fae546972a4199d
Author: daarinaaa <darina2436@yandex.ru>
Date:   Wed Jun 18 23:41:53 2025 +0300

    Initial Commit

```

Рис. 3.8: Переключение по имени тега и просмотр доступных тегов

3.7 Отмена локальных изменений (до индексации)

Убедимся, что мы находимся на последнем коммите ветки master и внесем изменение в файл hello.html в виде нежелательного комментария. Затем проверим статус, увидим, что изменения ещё не проиндексированы. Используем команду git checkout для переключения версии файла hello.html в репозитории(рис. fig. 3.9).

```
● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   hello.html

no changes added to commit (use "git add" and/or "git commit -a")
● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git checkout hello.html
Updated 1 path from the index
● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git status
On branch master
nothing to commit, working tree clean
```

Рис. 3.9: Отмена локальных изменений (до индексации)

3.8 Отмена проиндексированных изменений (перед коммитом)

Внесем изменение в файл hello.html в виде нежелательного комментария

```
<html>
  <head>
    <!-- This is an unwanted but staged comment -->
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Проиндексируем это изменение и проверим состояние. Состояние показывает, что изменение было проиндексировано и готово к коммиту. Используем команду `git reset`, чтобы сбросить буферную зону к HEAD. Это очищает буферную зону от изменений, которые мы только что проиндексировали. И переключимся на последнюю версию коммита, посмотрев статус увидим, что наш каталог опять чист.

3.9 Отмена коммитов

Изменим файл hello.html на следующий.

```
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <!-- This is an unwanted but committed change -->
  </body>
</html>
```

Проиндексируем изменения файла и прокоммитим их. Чтобы отменить коммит, нам необходимо сделать коммит, который удаляет изменения, сохраненные нежелательным коммитом. Перейдем в редактор, где изменим нежелательный коммит. Проверим лог. Проверка лога показывает нежелательные и отмененные коммиты в наш репозиторий(рис. fig. 3.10).

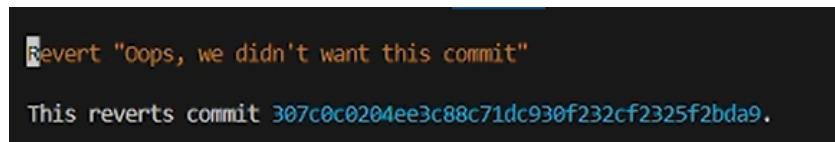


Рис. 3.10: Отмена коммитов

3.10 Удаление коммитов из ветки

Удалим последние два коммита с помощью сброса, сначала отметим последний коммит тегом, чтобы его можно было потом найти. Используем команду git reset, чтобы вернуться к версии до этих коммитов. Теперь в логе их нет, но если посмотреть логи с опцией –all можно всё ещё их увидеть, но метка HEAD находится на нужной нам версии(рис. fig. 3.11).

```
commit 307c0c0204ee3c88c71dc930f232cf2325f2bda9
Author: daarinaaa <darina2436@yandex.ru>
Date: Thu Jun 19 00:11:41 2025 +0300

    Oops, we didn't want this commit

commit d40d0c8ee1511a16708b85cf95fe142d46c851fd (tag: v1)
Author: daarinaaa <darina2436@yandex.ru>
Date: Thu Jun 19 00:01:14 2025 +0300

    Added standard HTML page tags

commit 895073e3ed6f23b2112c0a57c1bc23ca3a29593d (tag: v1-beta)
```

Рис. 3.11: Удаление коммиттов из ветки

3.11 Удаление тега oops

Удалим тег oops и коммиты, на которые он ссылался, сборщиком мусора. Теперь этот тег не отображается в репозитории(рис. fig. 3.12).

```
PS C:\Users\ASUS\Documents\University\6 semestr\МатМод\hello> git log --all
commit d40d0c8ee1511a16708b85cf95fe142d46c851fd (HEAD -> master, tag: v1)
Author: daarinaaa <darina2436@yandex.ru>
Date: Thu Jun 19 00:01:14 2025 +0300

    Added standard HTML page tags

commit 895073e3ed6f23b2112c0a57c1bc23ca3a29593d (tag: v1-beta)
Author: daarinaaa <darina2436@yandex.ru>
Date: Wed Jun 18 23:51:30 2025 +0300

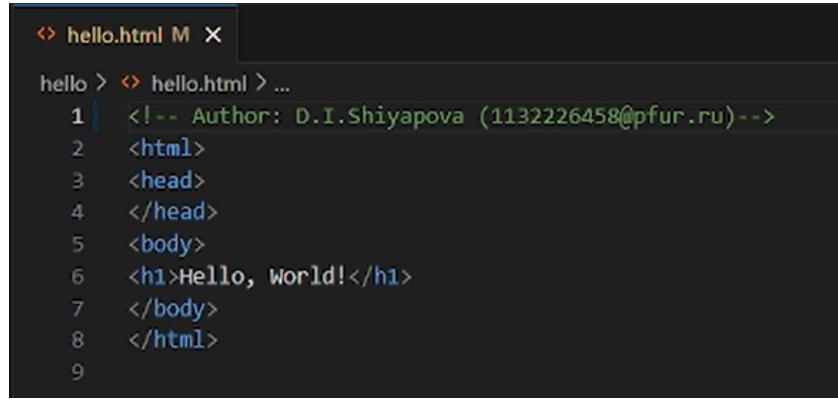
    Added h1 tag

commit 7fdd1dc7a79e7df7391e59a72fae546972a4199d
Author: daarinaaa <darina2436@yandex.ru>
Date: Wed Jun 18 23:41:53 2025 +0300
```

Рис. 3.12: Удаление тега oops

3.12 Изменение предыдущего коммита

Добавим в страницу комментарий автора (рис. fig. 3.13).



```
↳ hello.html M X
hello > < hello.html > ...
1  <!-- Author: D.I.Shiyapova (1132226458@pfur.ru) -->
2  <html>
3  <head>
4  </head>
5  <body>
6  <h1>Hello, world!</h1>
7  </body>
8  </html>
9
```

Рис. 3.13: Изменение предыдущего коммита

Затем добавим их в репозиторий. Теперь мы хотим добавить в комментарий автора почту, обновим страницу `hello`, включив в неё почту. Чтобы у нас остался один коммит, а не два, изменим последний с помощью опции `-amend`, теперь в логах отображается последняя версия коммита.

3.13 Перемещение файлов

Переместим наш файл в каталог `lib`. Для этого создадим его и используем команду `git mv`, сделаем коммит этого перемещения.

3.14 Подробнее о структуре

Добавим файл `index.html` в наш репозиторий

```
<html>
  <body>
    <iframe src="lib/hello.html" width="200" height="200" />
  </body>
</html>
```

Добавим файл и сделаем коммит.

Теперь при открытии index.html, увидим кусок страницы hello в маленьком окошке(рис.@fig:019).

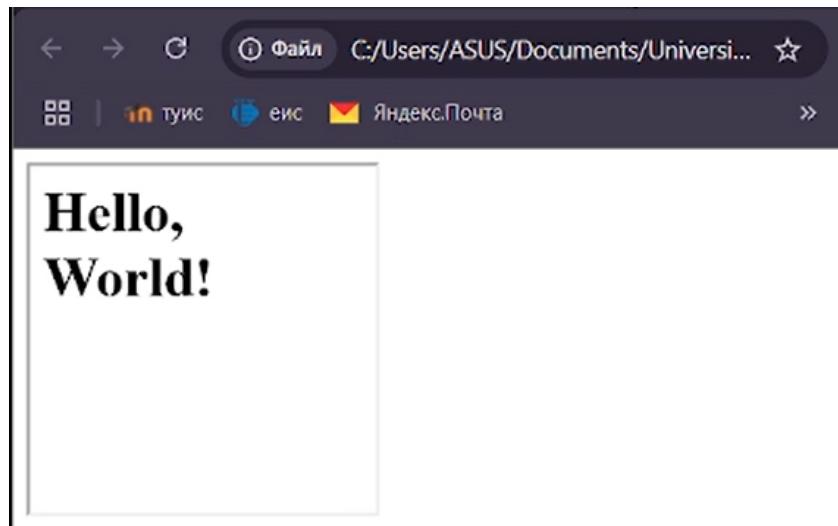


Рис. 3.14: Результат открытия index.html

3.15 Git внутри: Каталог .git

Просмотрим каталог, в котором хранится вся информация git. Затем посмотрим набор каталогов, имена которых состоят из 2 символов. Имена каталогов являются первыми двумя буквами хэша sha1 объекта, хранящегося в git. Посмотрим в один из каталогов с именем из 2 букв. Увидим файлы с именами из 38 символов. Это файлы, содержащие объекты, хранящиеся в git. Посмотрим файл конфигурации, создающийся для каждого конкретного проекта. Затем посмотрим подкаталоги .git/refs/heads и .git/refs/tags, а также содержимое файла v1, в нём хранится хэш коммита, привязанный к тегу. Также посмотрим содержимое файла HEAD, который содержит ссылку на текущую ветку, в данный момент это ветка master(рис. fig. 3.15).

```

● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> cat .git/config
[core]
    repositoryformatversion = 0
    filemode = false
    bare = false
    logallrefupdates = true
    symlinks = false
    ignorecase = true
● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> ls .git\refs

    Каталог: C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello\.git\refs

      Mode           LastWriteTime     Length Name
      ----          -----          ----- 
d----       19.06.2025        0:20      heads
d----       19.06.2025        0:14      tags

● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello>

```

Рис. 3.15: Каталог .git

3.16 Работа непосредственно с объектами git

Найдем последний коммит и выедем его с помощью SHA1 хэша. Затем посмотрим дерево каталогов, ссылка на который идёт в последнем коммите, выведем каталог lib и файл hello.html(рис. fig. 3.16).

```

● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git cat-file -t c02311905ae16ef3f902638311dfe599
c184f29
commit
● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git cat-file -p c02311905ae16ef3f902638311dfe599
c184f29
tree a5008632083521bf72a587e/b59e24db0fd77297
parent cd419ef025c38cbf7d2ae0621f3a76778a55472b
author daarinaaa <darina2436@yandex.ru> 1750281620 +0300
committer daarinaaa <darina2436@yandex.ru> 1750281620 +0300

Added index.html.
● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git cat-file -p a5008632083521bf72a587e/b59e24db0fd77297

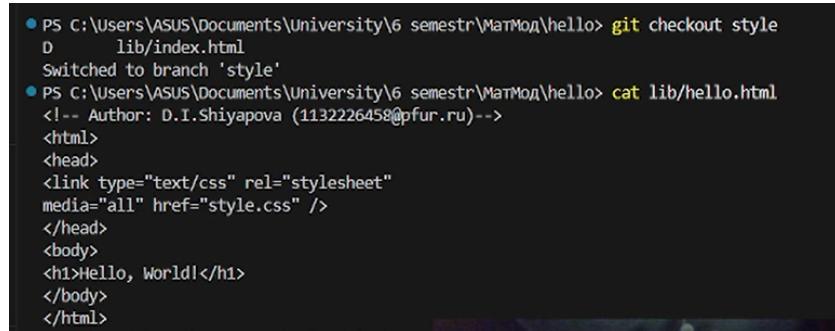
```

Рис. 3.16: Работа непосредственно с объектами git

Исследуем git репозиторий вручную самостоятельно. Используя хэш родительского коммита последовательно дойдем до первой версии файла hello.html и посмотрим его.

3.17 Создание ветки

Создадим новую ветку «style» и перейдем в неё. Добавим туда файл стилей style.css и добавим его в репозиторий. Обновим файл hello.html, чтобы использовать стили style.css и index.html, также обновим их в репозиторий(рис. fig. 3.17).

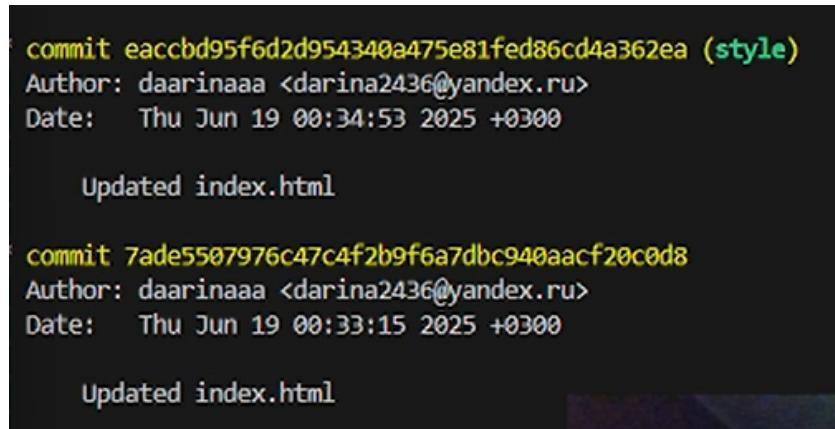


```
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git checkout style
D lib/index.html
Switched to branch 'style'
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> cat lib/hello.html
<!-- Author: D.I.Shiyapova (113222645@pfur.ru)-->
<html>
<head>
<link type="text/css" rel="stylesheet"
media="all" href="style.css" />
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Рис. 3.17: Создание ветки

3.18 Навигация по веткам

Посмотрим все логи(рис. fig. 3.18).



```
commit eaccbd95f6d2d954340a475e81fed86cd4a362ea (style)
Author: daarinaaa <darina2436@yandex.ru>
Date: Thu Jun 19 00:34:53 2025 +0300

    Updated index.html

commit 7ade5507976c47c4f2b9f6a7dbc940aacf20c0d8
Author: daarinaaa <darina2436@yandex.ru>
Date: Thu Jun 19 00:33:15 2025 +0300

    Updated index.html
```

Рис. 3.18: Просмотр логов новой ветки

Переключимся обратно на основную ветку и просмотрим содержимое файла lib/hello.html, заметим, что он не использует стили, также просмотрим содержимое этого файла в новой ветке(рис. fig. 3.19).

```
hello > lib > hello.html > ...
1   <!-- Author: D.I.Shiyapova (1132226458@pfur.ru)-->
2   <html>
3   <head>
4   <link type="text/css" rel="stylesheet"
5   media="all" href="style.css" />
6   </head>
7   <body>
8   <h1>Hello, World!</h1>
9   </body>
10  </html>
11
```

Рис. 3.19: Возвращение к ветке style

3.19 Изменения в ветке master

Вернемся в основную ветку и добавим файл README.md. Просмотрим ветки и их различия(рис. fig. 3.20).

```
* commit 9637e1b49f3c7792429a0fcfa0de51336f4bc1c45 (master)
  Author: daarinaaa <darina2436@yandex.ru>
  Date:   Thu Jun 19 00:36:29 2025 +0300

    Added README

* commit eaccbd95f6d2d954340a475e81fed86cd4a362ea
  Author: daarinaaa <darina2436@yandex.ru>
  Date:   Thu Jun 19 00:34:53 2025 +0300

    Updated index.html
```

Рис. 3.20: Изменения в ветке master

3.20 Слияние

Слияние переносит изменения из двух веток в одну. Вернемся к ветке style и сольем master с style(рис. fig. 3.21).

```
* commit 88dfbf059544247fef7102c7125e4cc0f22b3460
| Merge: eac cbd9 9637e1b
| Author: daarinaaa <darina2436@yandex.ru>
| Date: Thu Jun 19 00:37:29 2025 +0300
|
|       Merge branch 'master' into style
|
* commit 9637e1b49f3c7792429a0fc a0de51336f4bc1c45
| Author: daarinaaa <darina2436@yandex.ru>
| Date: Thu Jun 19 00:36:29 2025 +0300
|
|       Added README
```

Рис. 3.21: Слияние веток

3.21 Создание конфликта

Вернемся в ветку master и создадим конфликт, внеся изменения в файл hello.html. Просмотрим ветки. После коммита «Added README» ветка master была объединена с веткой style, но в настоящее время в master есть дополнительный коммит, который не был слит с style. Последнее изменение в master конфликтует с некоторыми изменениями в style.

3.22 Разрешение конфликтов

Вернемся к ветке style и попытаемся объединить ее с новой веткой master. В файле lib/hello.html можно увидеть записи с обеих версий этого файла. Первый раздел — версия текущей ветки (style). Второй раздел — версия ветки master. Внесем изменения в lib/hello.html, оставив только необходимую нам запись и добавим этот файл в репозиторий, чтобы вручную разрешить конфликт.

3.23 Сброс ветки style

Вернемся на ветку style к точке перед тем, как мы слили ее с веткой master. Мы хотим вернуться в ветку style в точку перед слиянием с master. Нам необходимо найти последний коммит перед слиянием.

Мы видим, что коммит «Updated index.html» был последним на ветке style перед слиянием. Сбросим ветку style к этому коммиту.

Поищим лог ветки style. Увидим, что у нас в истории больше нет коммитов слияний.

3.24 Сброс ветки master

Добавив интерактивный режим в ветку master, мы внесли изменения, конфликтующие с изменениями в ветке style. Давайте вернемся в ветку master в точку перед внесением конфликтующих изменений. Это позволяет нам продемонстрировать работу команды git rebase, не беспокоясь о конфликтах. Просмотрим коммиты ветки master.

Коммит «Added README» идет непосредственно перед коммитом конфликтующего интерактивного режима. Мы сбросим ветку master к коммиту «Added README».

3.25 Перебазирование

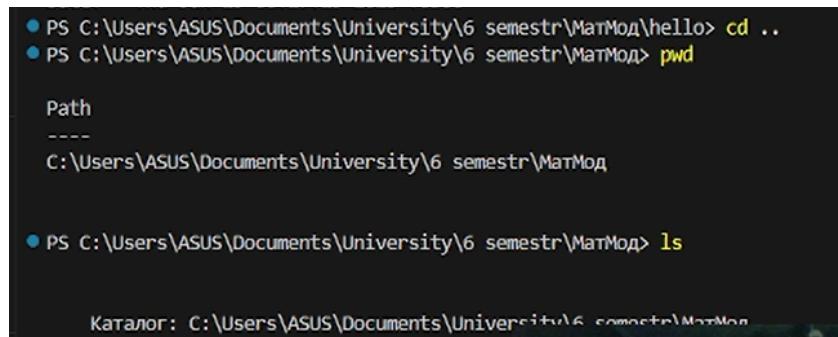
Используем команду rebase вместо команды merge. Мы вернулись в точку до первого слияния и хотим перенести изменения из ветки master в нашу ветку style. На этот раз для переноса изменений из ветки master мы будем использовать команду git rebase вместо слияния.

3.26 Слияние в ветку master

Вернемся в ветку master и сольем ветку style в неё с помощью команды git merge.

3.27 Клонирование репозиториев

Перейдем в наш рабочий каталог и сделаем клон репозитория hello, затем создадим клон репозитория. Просмотрев его увидим список всех файлов на верхнем уровне оригинального репозитория README.md, index.html и lib. Затем просмотрим историю репозитория и увидим список всех коммитов в новый репозиторий, и он совпадает с историей коммитов в оригинальном репозитории. Единствен в названиях веток(рис. fig. 3.22).



```
● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> cd ..
● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> pwd
Path
-----
C:\Users\ASUS\Documents\University\6 semestr\MatMod

● PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> ls
Каталог: C:\Users\ASUS\Documents\University\6 semestr\MatMod
```

Рис. 3.22: Клонирование репозиториев

3.28 Что такое origin?

Клонированный репозиторий знает об имени по умолчанию удаленного репозитория. Посмотрим, подробную информацию об имени по умолчанию. Для того, чтобы увидеть все ветки используем опцию -a(рис. fig. 3.23).

```
Author: daarinaaa <darina2436@yandex.ru>
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\cloned_hello> git remote
origin
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\cloned_hello> git remote show origin
* remote origin
  Fetch URL: C:/Users/ASUS/Documents/University/6 semestr/МатМод/hello
  Push URL: C:/Users/ASUS/Documents/University/6 semestr/МатМод/hello
  HEAD branch: master
  Remote branches:
```

Рис. 3.23: Просмотр имени по умолчанию удаленного репозитория

3.29 Удаленные ветки

Посмотрим на ветки, доступные в нашем клонированном репозитории. Можно увидеть, что в списке только ветка master.

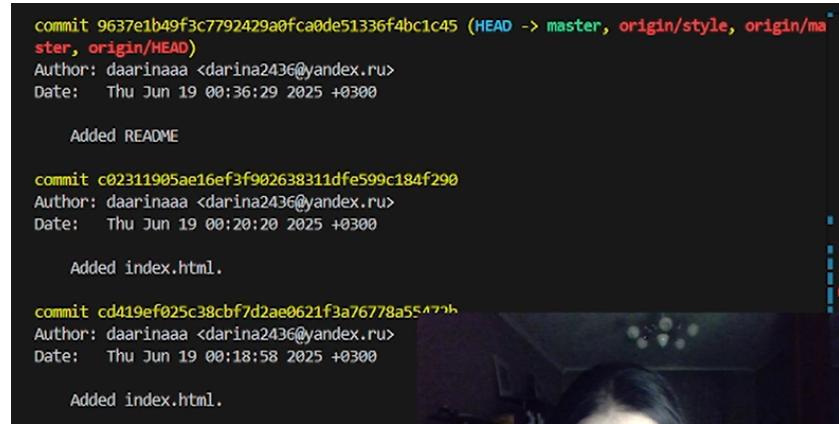
3.30 Изменение оригинального репозитория

Перейдем в репозиторий hello. Внесем изменения в файл README.md. Затем добавим их в репозиторий(рис. fig. 3.24).

```
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git add README
fatal: pathspec 'README' did not match any files
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git add README.md
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git commit -m "Changed README in original repo"
```

Рис. 3.24: Изменение оригинального репозитория

Перейдём в клон репозитория и используем команду git fetch, которая будет извлекать новые коммиты из удаленного репозитория, но не будет сливать их с наработками в локальных ветках(рис. fig. 3.25).



```
commit 9637e1b49f3c7792429a0fc0de51336f4bc1c45 (HEAD -> master, origin/style, origin/master, origin/HEAD)
Author: daarinaaa <darina2436@yandex.ru>
Date: Thu Jun 19 00:36:29 2025 +0300

    Added README

commit c02311905ae16ef3f902638311dfe599c184f290
Author: daarinaaa <darina2436@yandex.ru>
Date: Thu Jun 19 00:20:20 2025 +0300

    Added index.html.

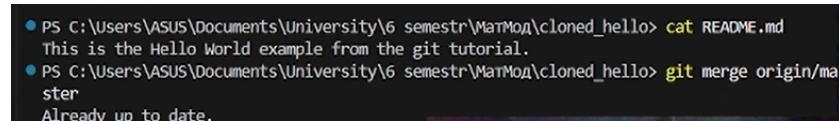
commit cd419ef025c38cbf7d2ae0621f3a76778a554724
Author: daarinaaa <darina2436@yandex.ru>
Date: Thu Jun 19 00:18:58 2025 +0300

    Added index.html.
```

Рис. 3.25: Извлечение изменений

3.31 Слияние извлечённых изменений

Сольем внесённые изменения в главную ветку. Также можно было бы использовать команду git pull, которая является объединением fetch и merge в одну команду (рис. fig. 3.26).

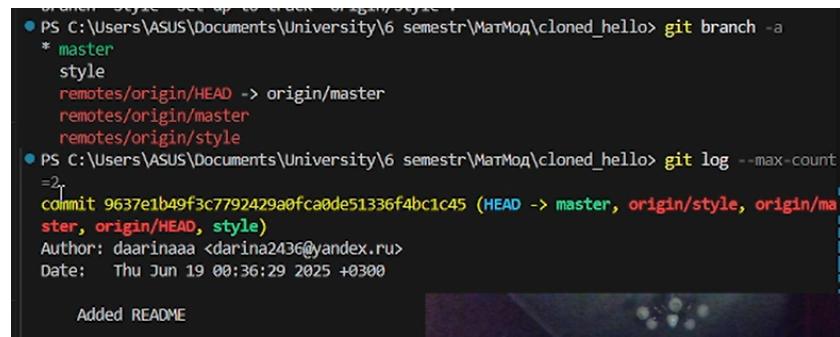


```
PS C:\Users\ASUS\Documents\University\6 semestr\MarMod\cloned_hello> cat README.md
This is the Hello World example from the git tutorial.
PS C:\Users\ASUS\Documents\University\6 semestr\MarMod\cloned_hello> git merge origin/master
Already up to date.
```

Рис. 3.26: Слияние извлечённых изменений

3.32 Добавление ветки наблюдения

Добавим локальную ветку, которая отслеживает удаленную ветку, теперь мы можем видеть ветку style в списке веток и логе(рис. fig. 3.27).



```
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\cloned_hello> git branch -a
* master
  style
    remotes/origin/HEAD -> origin/master
    remotes/origin/master
    remotes/origin/style

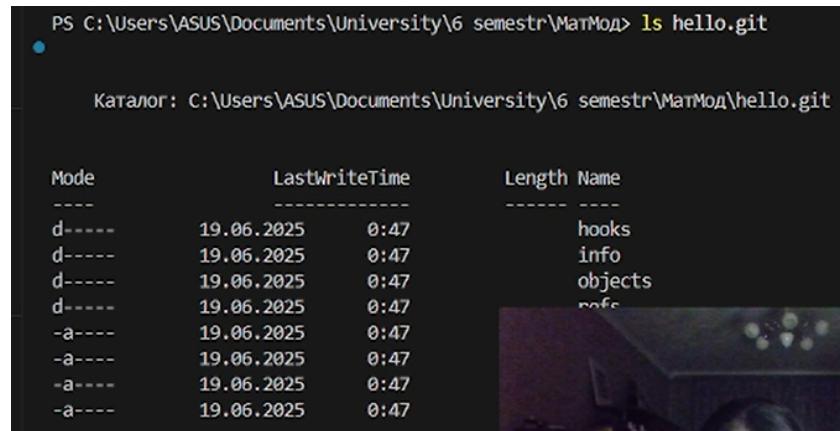
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\cloned_hello> git log --max-count=2
commit 9637e1b49f3c7792429a0fc0de51336f4bc1c45 (HEAD -> master, origin/style, origin/master, origin/HEAD, style)
Author: daarinaaa <darina2436@yandex.ru>
Date:   Thu Jun 19 00:36:29 2025 +0300

    Added README
```

Рис. 3.27: Добавление ветки наблюдения

3.33 Создание чистого репозитория

Как правило, репозитории, оканчивающиеся на .git являются чистыми репозиториями. Создадим такой в рабочем каталоге. Затем добавим репозиторий hello.git к нашему оригинальному репозиторию(рис. fig. 3.28).



```
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod> ls hello.git
.
+
Каталог: C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello.git

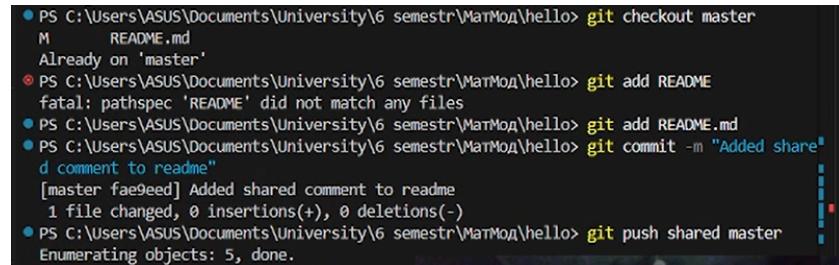
Mode          LastWriteTime      Length Name
----          -----          ----
d----
```

Рис. 3.28: Создание чистого репозитория

3.34 Отправка и извлечение изменений

Так как чистые репозитории, как правило, расшариваются на каком-нибудь сетевом сервере, нам необходимо отправить наши изменения в другие репозитории. Начнем с создания изменения для отправки. Отредактируем файл

README.md и сделаем коммит, затем отправим изменения в общий репозиторий. Затем извлечем изменения из общего репозитория(рис. fig. 3.29).



```
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git checkout master
M      README.md
Already on 'master'
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git add README
fatal: pathspec 'README' did not match any files
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git add README.md
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git commit -m "Added shared comment to readme"
[master fae9eed] Added shared comment to readme
 1 file changed, 0 insertions(+), 0 deletions(-)
PS C:\Users\ASUS\Documents\University\6 semestr\MatMod\hello> git push shared master
Enumerating objects: 5, done.
```

Рис. 3.29: Отправка изменений

4 Выводы

В процессе выполнения данной лабораторной работы я приобрела практические навыки работы с Git.

Список литературы