**Практическая работа №1-2**

«**Разработка структуры проекта. Разработка модульной структуры проекта**»

**Цель работы:** приобрести навыки разработки структуры проекта

**Порядок выполнения работы**

1. Ознакомиться с краткими теоретическими сведениями и заданиями к работе.
2. Выполнить задания работы.
3. Оформить отчет о выполнении.

**Краткие теоретические сведения**

Прежде всего необходимо отметить, что в Git не существует традиционной системы отмены, как в текстовых редакторах. Лучше воздержаться от сопоставления операций Git с какой бы то ни было традиционной концепцией отмены изменений. Кроме того, Git имеет собственную систему терминов для операций отмены, и в обсуждении лучше всего использовать их. В числе таких терминов — сброс (reset), возврат (revert), переключение (checkout), очистка (clean) и другие.

Так же дадим два определения ветки: на логическом и физическом уровнях.

1. Логический уровень.

С точки зрения логики, ветка – это последовательность коммитов. Чтобы проще было понять, что такое ветка, рассматривайте ее как некоторую временную шкалу. Коммиты в ней – снимки интересных моментов, идущие друг за другом в хронологической последовательности. Рисунок ниже поможет вам в интуитивном представлении.

2. Физический уровень

На физическом уровне, то есть с точки зрения внутренней реализации Git, ветка – это ссылка на последний коммит в этой ветке.

Преимущество веток в их независимости. Вы можете вносить изменения в файлы в одной ветке, например, пробовать новую функцию, и они никак не скажутся на файлах в другой ветке. Изначально в репозитории одна ветка, но позже мы рассмотрим, как создавать другие.

Зачем нужны такие сложности, ведь можно просто делать коммиты и откатывать изменения, когда нужно?

Дело в том, что Git – универсальная система контроля версий: она подходит и большим командам крупных корпораций, и индивидуальным разработчикам для их личных проектов.

Если вы работаете один – скорее всего вы будете редко использовать ветки, но если вы работаете в большой компании, без веток вам не обойтись.

**ЗАДАНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ**

Сперва давайте настроим конфигурацию для работы. Так как за вашим ПК может потом работать кто-то другой, делать глобальную конфигурацию мы не будем. Сделаем конфигурацию относительно репозитория. И так, сначала скачаем наш (ваш любой) репозиторий -

**git clone ссылка\_на\_репозиторий\_hhtps**

Дальше перейдем в ту папку (репозиторий), которую скачали -

**cd название\_вашей\_папки**

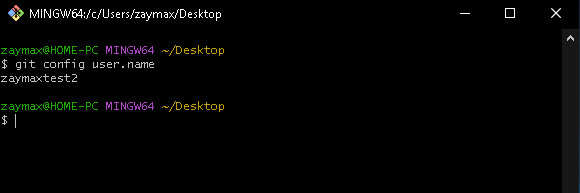
Дальше настроим конфигурацию для нее -

**git config user.name “ваш\_никнейм\_гитхаба”**

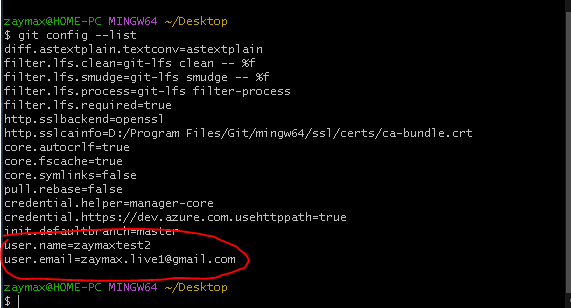
**git config user.email ваша\_почта\_гитхаба**

Так же вы можете проверить, чья сейчас конфигурация используется командой -

**git config user.name**



Или же командой **git config --list** посмотреть всю информацию о конфигурации



Далее создадим три файла в папке скачанного репозитория - index.html, style.css, text.txt. Пока что создадим пустую страницу сайта, можно в VS Code командой ! и нажать enter или же вручную вписать -

**!** (если нет vs code можете в блокноте это сделать)

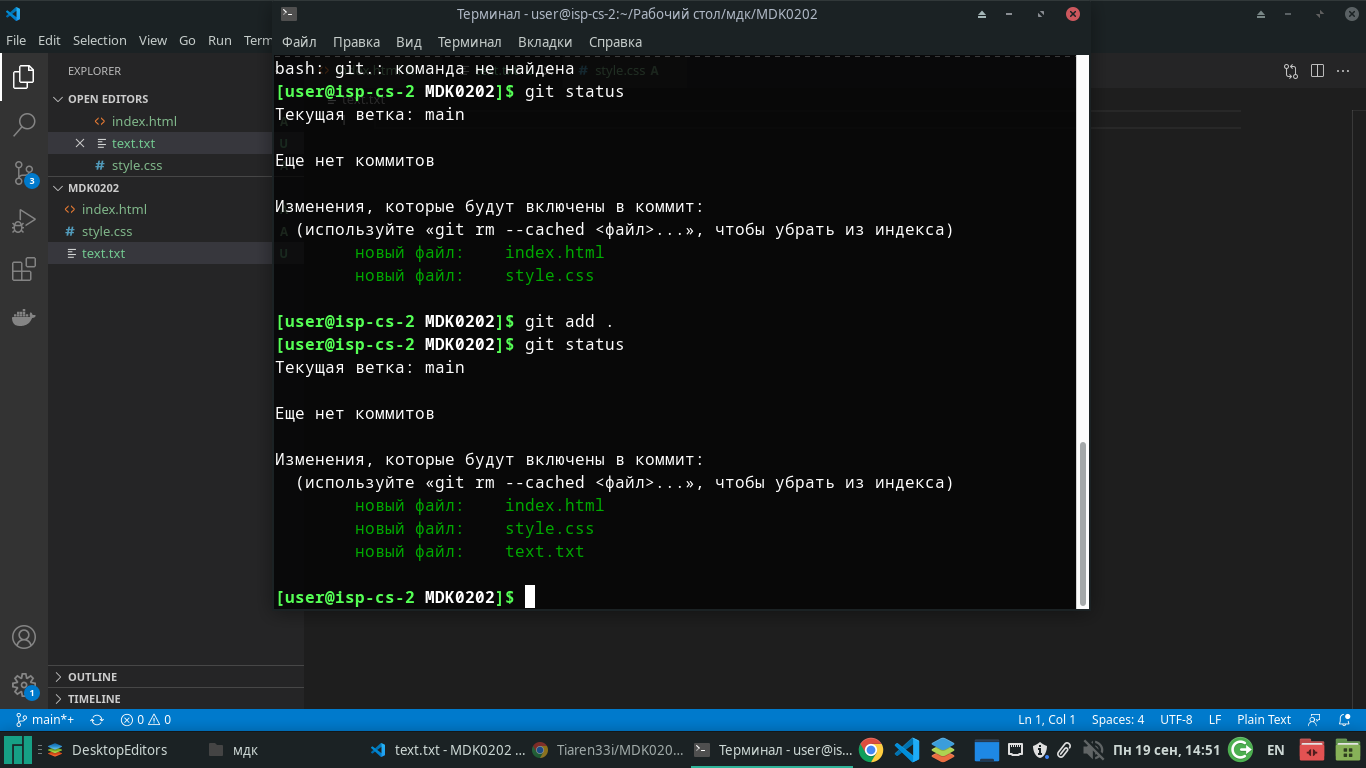


Так же не забываем, что нужно подключить наш style.css к index.html в теге <head></head> командой <link rel="stylesheet" href="style.css">



Теперь давайте проиндексируем файлы, чтобы можно было их пушить в наш репозиторий, командой -

**git add .**



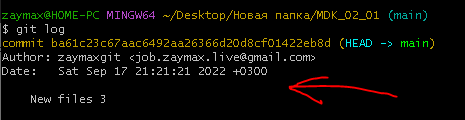
Так же сразу закоммитим изменения командой -

**git commit –m “New 3 files”**

Вдруг нам захотелось просмотреть, а какие коммиты вы писали, то можно это узнать командой -

**git log**

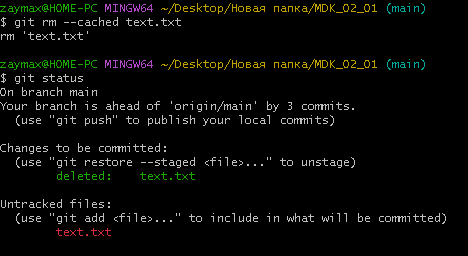
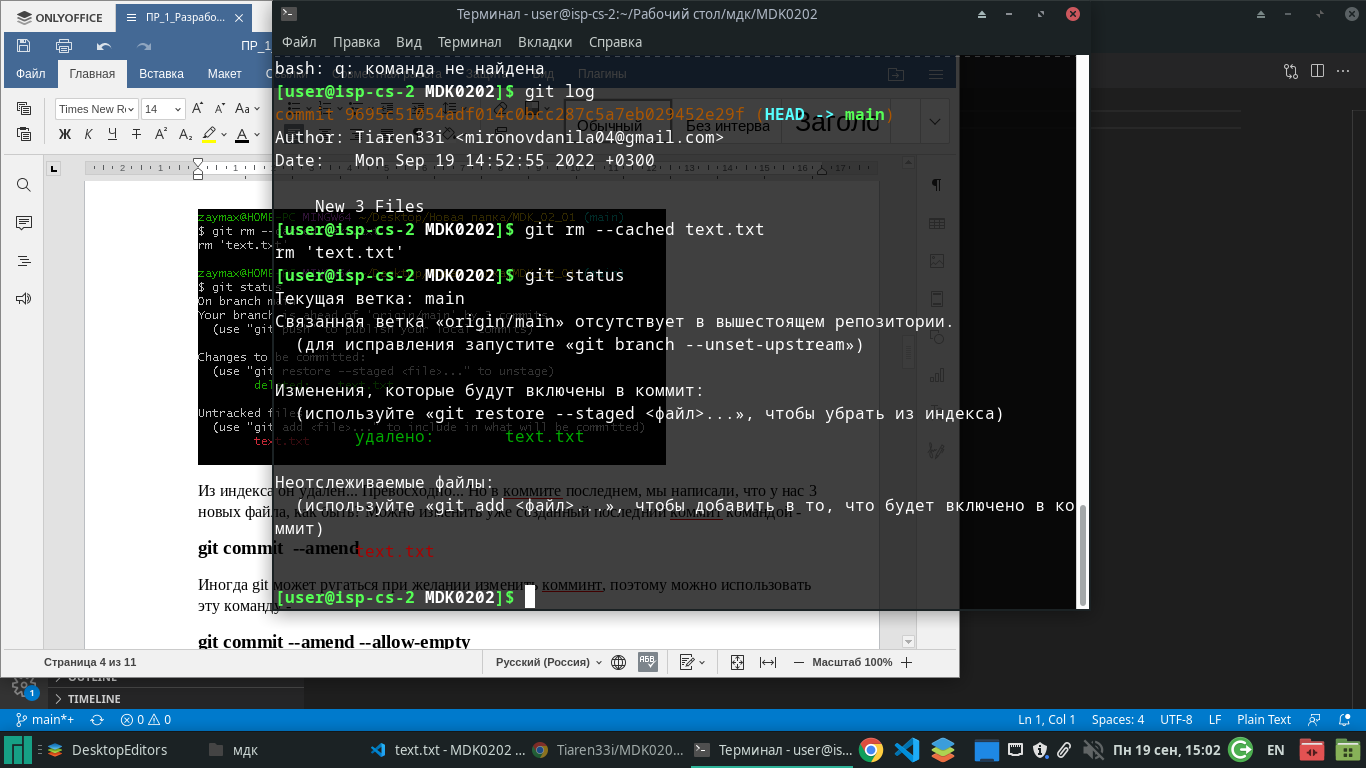
**!** (Чтобы выйти из данной команды просто нажмите q)



Но что если мы поняли, что нам не нужен был третий файл, а именно text.txt?!

То в таком случае первое, что нужно сделать нам это удалить его из индекса командой -

**git rm --cached название\_файла**



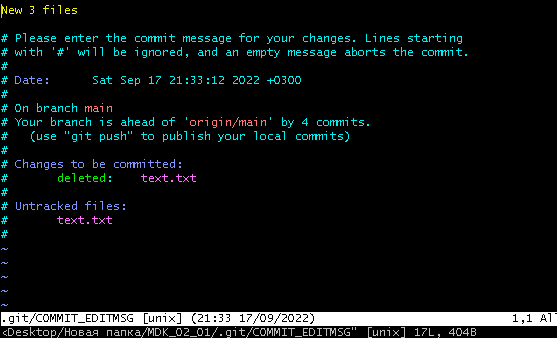
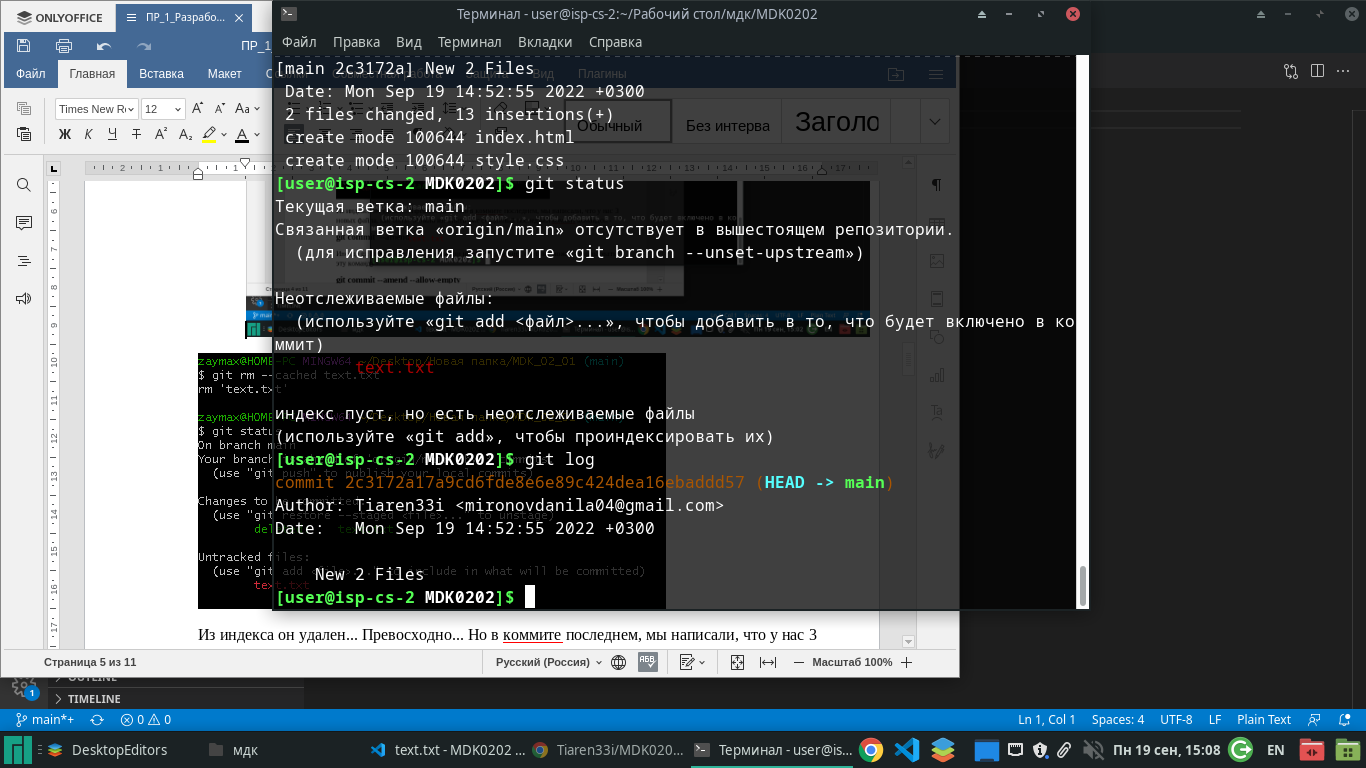
Из индекса он удален... Превосходно... Но в коммите последнем, мы написали, что у нас 3 новых файла, как быть? Можно изменить уже созданный последний коммит командой -

**git commit --amend**

Иногда git может ругаться при желании изменить комминт, поэтому можно использовать эту команду -

**git commit --amend --allow-empty**

Откроется редактор VIM (простите, но вам придется его познать).



Как им пользоваться? Нам сначала надо включить режим редактирования нажав кнопку i, дальше удалить \ изменить комментарий, после чего изменения нужно сохранить, чтобы сохранить изменения в нем нужно прописать **:w**, а после всего этого нам необходимо выйти из редактора VIM командой **:q**

И так вы сохранили наш сайт, но давайте его дополним и запушим на другую ветку как вторую версию проекта.

И так, сначала создадим ветку и перейдем на нее. Создать ветку и перейти на нее можно так -

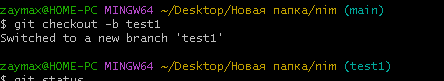
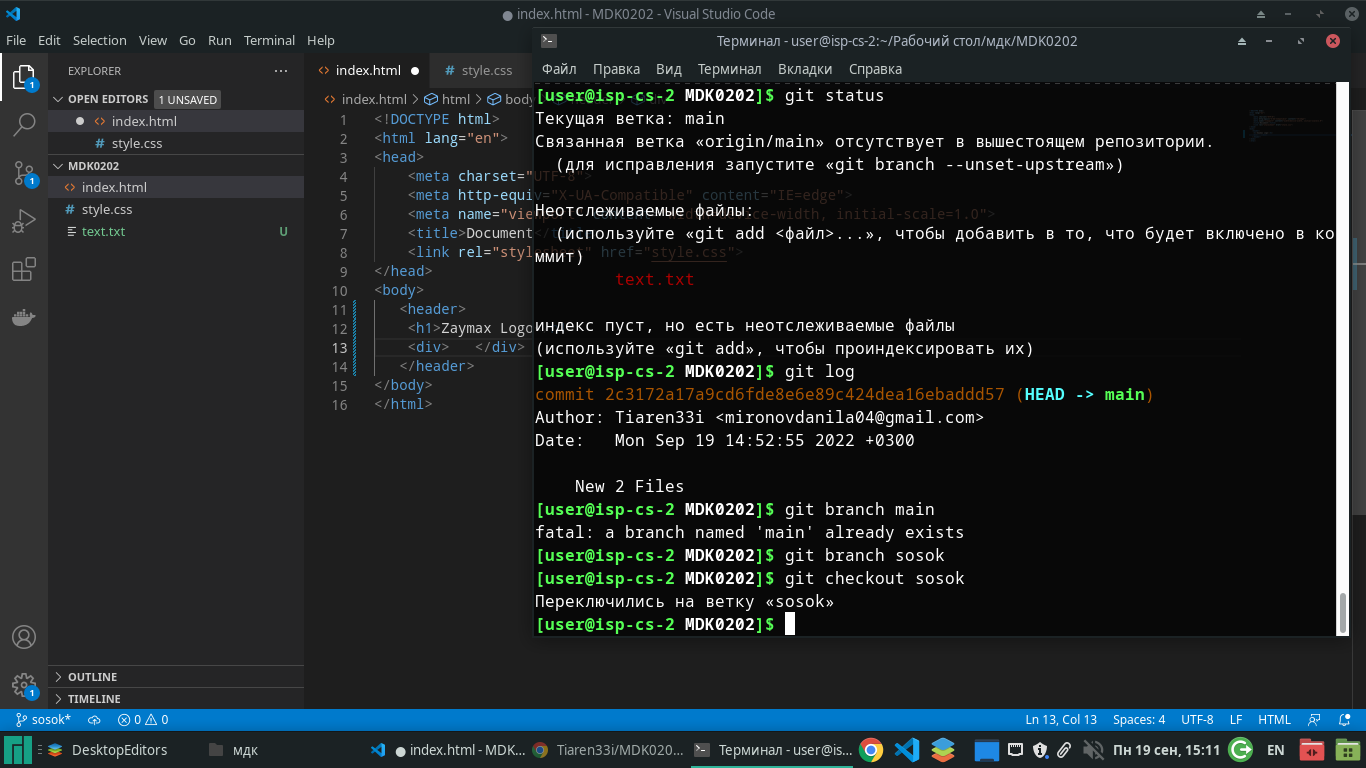
**git branch название\_ветки**

Чтобы перейти команда -

**git checkout название\_ветки**

Если хотите создать ветку и сразу на нее перейти, то команда -

**git checkout –b название\_веки**



Теперь давайте добавим, что-нибудь нового в сайт. Создадим шапку и добавим приветствие.

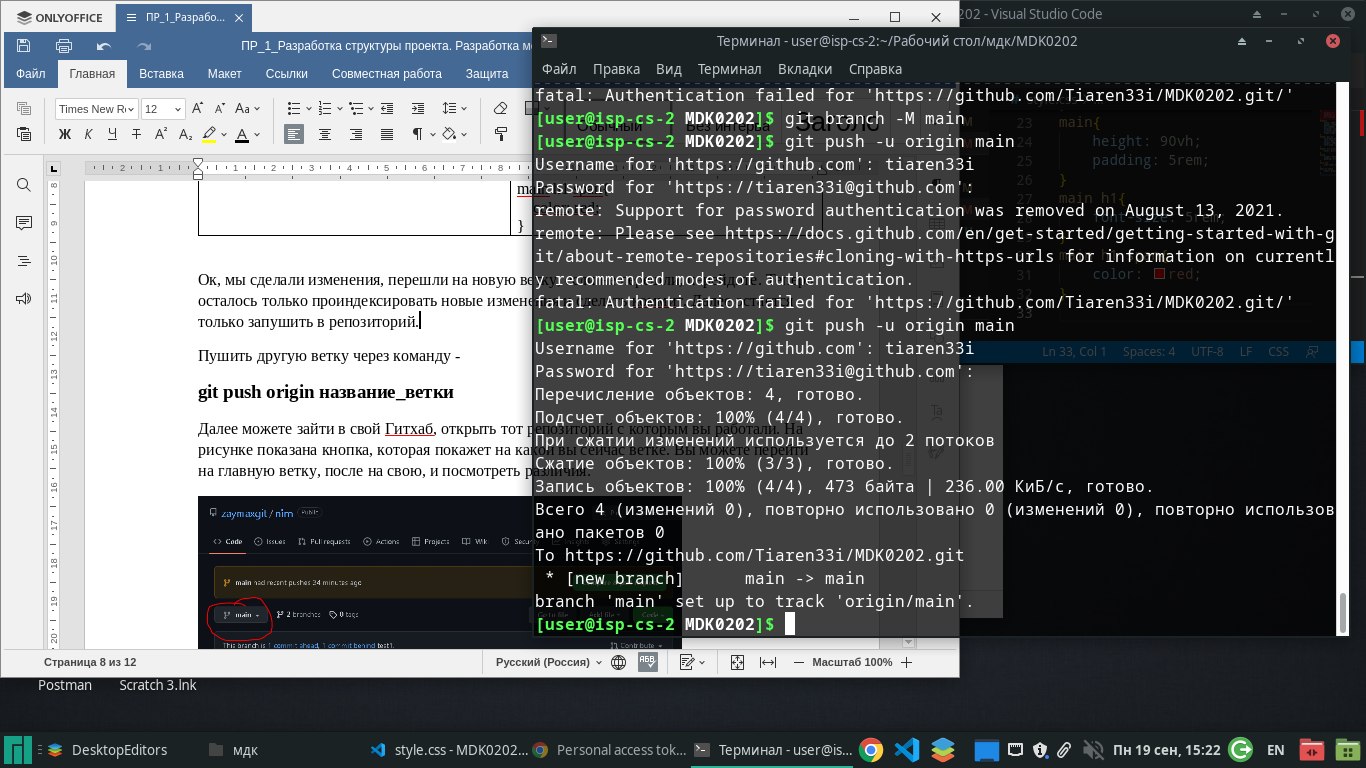
Если что-то в коде не понятно, спросите, объясню)

|  |  |
| --- | --- |
| **Index.html** | **Style.css** |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <link rel="stylesheet" href="style.css">  </head>  <body>  <header>  <h1>Zaymax Logo</h1>  <div>  <a href="#">Home</a>  <a href="#">About</a>  <a href="#">Contact</a>  <a href="#">Blog</a>  </div>  </header>  <main>  <h1>Hi my <span>Lord!</span></h1>  </main>  </body>  </html> | \*{  margin: 0;  padding: 0;  box-sizing: border-box;  }  header{  width: 100%;  height: 10vh;  box-shadow: 0px 1px 10px 1px #d3d3d3;  display: flex;  justify-content: space-around;  align-items: center;  }  header div a{  text-decoration: none;  color: #111;  margin: 10px;  font-size: 1.1rem;  }  header div a:hover{  color: red;  }  main{  height: 90vh;  padding: 5rem;  }  main h1{  font-size: 5rem;  }  main h1 span{  color: red;  } |

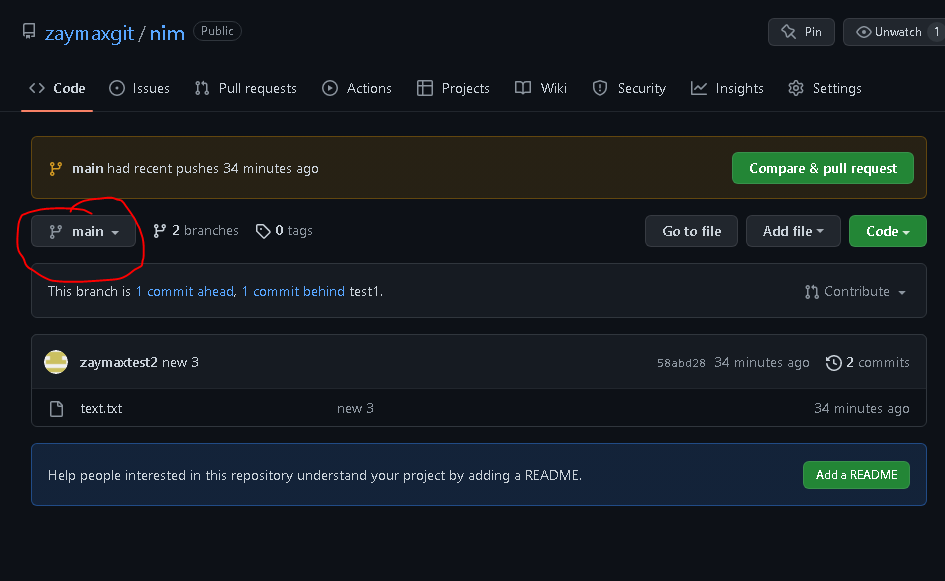
Ок, мы сделали изменения, перешли на новую ветку, если не перешли, перейдите. Теперь осталось только проиндексировать новые изменения и сделать коммит. Далее остается только запушить в репозиторий.

Пушить другую ветку через команду -

**git push origin название\_ветки**



Далее можете зайти в свой Гитхаб, открыть тот репозиторий с которым вы работали. На рисунке показана кнопка, которая покажет на какой вы сейчас ветке. Вы можете перейти на главную ветку, после на свою, и посмотреть различия.



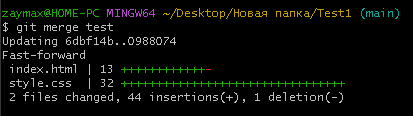
На данном моменте мы грубо говоря добавили новый функционал к нашему проекту, но в другой ветке.

Когда наши наставники на работе Senior разработчики или Тим-лид посмотрели, ваш код прошел проверку, можно его добавлять в основной проект, то есть добавлять в главную ветку, как это называют «Мердж» – слияние ветки, это перенос кода из одной ветки в другую.

Как слить ветки воедино? Нужно сначала перейти на главную ветку, в нашем случае это main.

А дальше использовать команду - ghp\_bDdosT4IxjKvKyUgiCsWP3p9g3ttSe1Dmr96

**git merge название\_ветки\_которую\_хотите\_слить\_с\_основной**

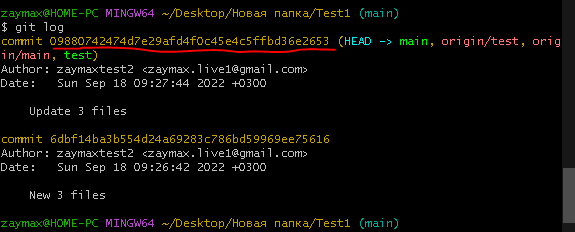


После чего ваша главная ветка (main) будет слита с другой вашей веткой. После этого, нужно опять запушить изменения **(git push)**

Объединение может привести к нежелаемым результатам, и что если, после слияния веток, мы обнаружили ошибку... Нам необходимо откатить слияние, как это сделать?

Первое, что нужно, это перейти на главную ветку с которой мы сливали другую ветку.

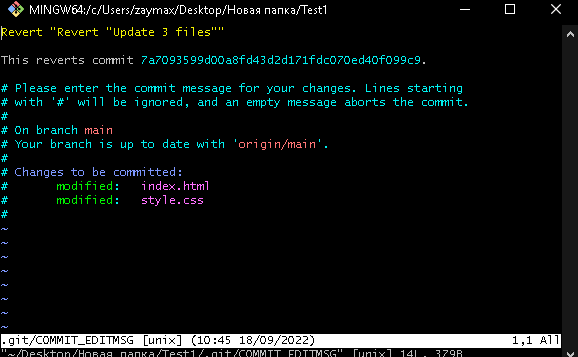
После чего запустить команду **git log**, чтобы получить ID сделанного объединения.



Далее стоит выполнить данную команду -

**git revert -m 1 id\_объединения**

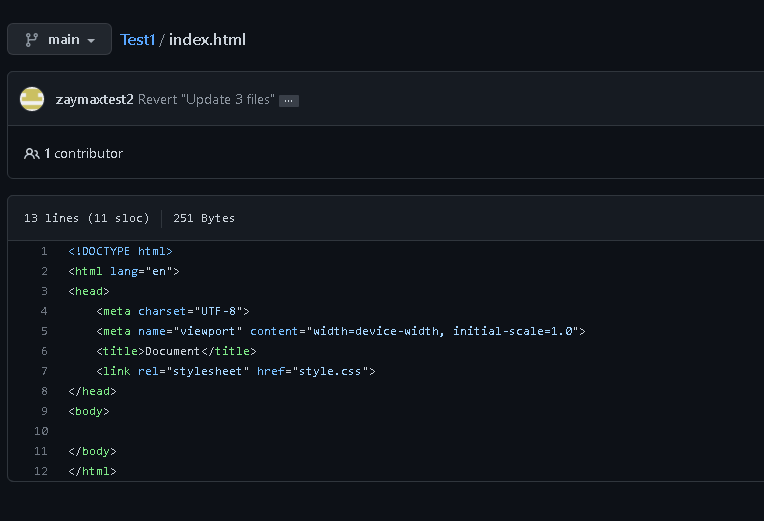
Далее опять откроется редактор VIM, в нем мы можете изменить коммит, но по вашему усмотрению.



Сохраняем изменения и выходим из редактора (напоминаю команды - **:w :q**)

Далее не забываем запушить изменение **(git push)**

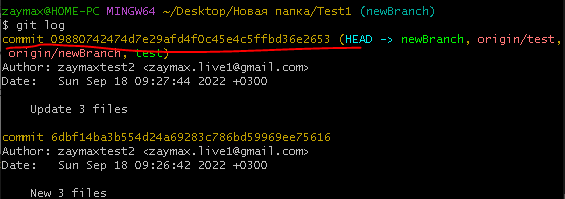
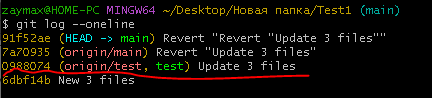
Теперь если вы посмотрите на Гитхабе вашу главную ветку, то увидите, что она вернулась к прежнему состоянию.



Но что если вы уже сделали какое-то изменение в проекте, но хотите это изменение перенести на другую ветку? Тогда можно создать новую ветку на основе коммита.

Найдите коммит, который вы указывали для изменения файлов (то есть, когда вы их изменили, вы создали коммит, он нам и нужен). Именно нам понадобиться ID коммита. Это можно посмотреть с помощью -

**git log**  или  **git log --oneline** (выбирать вам)



После чего создаем ветку на основе коммита -

**git branch название\_ветки id\_коммита**



Ура, ветка создана, дальше можно в нее перейти и запушить в репозиторий. Теперь ваше новое изменение будет храниться и в другой ветке.

**Конец практической.**



Отчеты

