Министерство образования и науки России

ФГБОУ ВО "Севастопольский государственный университет"

Кафедра ИС

Отчет

по лабораторной работе № 2

**ПРИМЕНЕНИЕ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ GIT В ПРОЦЕССЕ РАЗРАБОТКИ ИС**

Выполнил:

ст. гр. ИC-42

Лисянский А. И.

Проверил:

Севастополь

2015

**Цель работы:**

изучить основы локальной работы с системой контроля версий Git.

**Ход работы**

Запустите Git.

Установите имя и электронную почту

git config --global user.name "Your Name"

git config --global user.email "your\_email@whatever.com"

Начните работу в пустом рабочем каталоге

mkdir <имя каталога>

Выберите каталог

cd <имя каталога>

Создайте там текстовый файл

touch <имя файла. >

Создайте репозиторий

git init

Добавьте страницу в репозиторий

git add hello.html

Создайте коммит

git commit -m "First Ccmmit"

Проверьте состояние репозитория

git status

Команда проверки состояния сообщит, что коммитить нечего. Это означает, что в репозитории хранится текущее состояние рабочего каталога, и нет никаких изменений, ожидающих записи.

Внесите изменение в исходный файл

Проверьте состояние

git status

Первое, что нужно заметить, это то, что git знает, что созданный ранее файл  был изменен, но при этом эти изменения еще не зафиксированы в репозитории.

Также обратите внимание на то, что сообщение о состоянии дает вам подсказку о том, что нужно делать дальше. Если вы хотите добавить эти изменения в репозиторий, используйте команду git add. В противном случае используйте команду git сheckout для отмены изменений.

Добавьте изменения

git add <имя файла. >

Проверьте состояние

git status

Изменения файла  были проиндексированы. Это означает, что git теперь знает об изменении, но изменение пока не *перманентно* (*навсегда*) записано в репозиторий. Следующий коммит будет включать в себя проиндексированные изменения.

Если вы решили, что *не* хотите коммитить изменения, команда состояния напомнит вам о том, что с помощью команды git reset можно снять индексацию этих изменений.

Закоммитьте изменения

git commit

Если вы опустите метку -m из командной строки, git перенесет вас в редактор по вашему выбору. Редактор выбирается из следующего списка (в порядке приоритета):

переменная среды GIT\_EDITOR

параметр конфигурации core.editor

переменная среды VISUAL

переменная среды EDITOR

В первой строке редактора введите комментарий: «Added h1 tag». Сохраните файл и выйдите из редактора (для этого в редакторе по-умолчанию (Vim) вам нужно нажать клавишу ESC, ввести :wq и нажать Enter).

Проверьте состояние

git status

Рабочий каталог чистый, можете продолжить работу.

Измените созданный ранее файл

Добавьте это изменение в индекс git.

git add <имя файла>

Добавьте ещё одно изменение в исходный файл

Проверьте текущий статус

git status

Обратите внимание на то, что файл указан дважды в состоянии.

Первое изменение проиндексировано (git add) и готово к коммиту.

Второе изменение является непроиндексированным. Если бы вы делали коммит сейчас, заголовки не были бы сохранены в репозиторий.

Произведите коммит проиндексированного изменения (значение по умолчанию)

git commit -m "Added …"

Еще раз проверьте состояние

git status

Состояние команды говорит о том, что файл имеет незафиксированные изменения, но уже не в буферной зоне.

Добавьте второе изменение в индекс

git add .

В качестве файла для добавления, использовался текущий каталог («.»). Это самый краткий и удобный путь для добавления всех изменений в файлы текущего каталога и его подкаталоги. Но поскольку он добавляет все, *не лишним* будет проверить состояние перед запуском add, просто чтобы убедиться, что вы не добавили какой-то файл, который добавлять было не нужно.

Проверьте состояние с помощью команды git status. Второе изменение было проиндексировано и готово к коммиту.

Сделайте коммит второго изменения

git commit -m. "Added …"

Просмотрите историю проекта

git log

вы увидите список всех коммитов в репозиторий, которые были совершены.

27. Используйте однострочный формат

git log --precty=oneline

Выведите несколько вариантов истории

git log —pretty=oneline --max-count=2

git log —pretty=oneline --since='5 minutes ago'

git log —pretty=oneline —-until='5 minutes ago'

git log —pretty=oneline --author=<your name>

git log —pretty=oneline –-all

git log --pretty=format: "%h %ad | %s%d [%an]" --graph --date=short

--pretty="..." – определяет формат вывода.

%h – укороченный хэш коммита

%d – дополнения коммита («головы» веток или теги)

%ad – дата коммита

%s – комментарий

%an – имя автора

--graph – отображает дерево коммитов в виде ASCII-графика

--date=short – сохраняет формат даты коротким и симпатичным

Просмотрите историю проекта

gitk

Настройте алиасы и шоркаты для команд Git

git config --global alias.со checkout

git config --global alias.ci commit

git config --global alias.st status

git config --global alias.br branch

git config --global alias.hist 'log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short'

Теперь вы можете вводить git st там, где раньше приходилось использовать git status. Аналогичным образом, пишем git co вместо git checkout и git ci вместо git commit.

Используйте новую команду для просмотра состояния

Git st

Создайте алиасы для следующих команд

alias gs='git status '

alias ga='git add '

alias gb='git branch'

alias gc='git commit'

alias gd='git diff'

alias go='git checkout '

Сокращение go для команды git checkout особенно полезно. Оно позволяет вводить: go <branch>

Просмотрите историю проекта

git hist

Изучите данные лога и найдите хэш для первого коммита. Он должен быть в последней строке данных git hist. Используйте этот хэш-код (достаточно первых 7 знаков) в команде ниже.

git checkout <hash>

cat <имя файла. >

<hash> – это первые семь знаков хэш-кода, например $ git checkout 911e8c9

Проверьте содержимое созданного файла.

Вернитесь к последней версии в ветке master

git checkout master

cat <имя файла. >

«master» – имя ветки по умолчанию. Переключая имена веток, вы попадаете на последнюю версию выбранной ветки.

Внесите изменение в файл.

Иногда случается, что вы изменили файл в рабочем каталоге, и хотите отменить последние коммиты. Для этого используется команда checkout.

Проверьте состояние

git status

Видим, что файл  был изменен, но еще не проиндексирован.

Отмените изменения в рабочем каталоге.

git checkout <имя файла. >

Проверьте состояние

git status

Команда status показывает, что не было произведено никаких изменений, не зафиксированных в рабочем каталоге. И нежелательное изменение больше не является частью содержимого файла.

cat <имя файла. >

В очередной раз измените файл.

Проиндексируйте изменения

git add <имя файла. >

Проверьте состояние нежелательного изменения.

git status

Состояния показывает, что изменение было проиндексировано и готово к коммиту.

Выполните сброс буферной зоны.

git reset HEAD <имя файла. >

Команда reset сбрасывает буферную зону к HEAD. Это очищает буферную зону от изменений, которые мы только что проиндексировали.

Команда reset (по умолчанию) не изменяет рабочий каталог. Поэтому рабочий каталог все еще содержит нежелательный комментарий.

Удалите нежелательные изменения в рабочем каталоге, используя команду checkout из задания.

Просмотрите изменения

git status

Иногда вы понимаете, что новые коммиты являются неверными, и хотите их отменить. Есть несколько способов решения этого вопроса, далее мы будем использовать самый безопасный.

Отмените коммит путем создания нового коммита, отменяющего нежелательные изменения.

Измените файл.

Проиндексируйте его и создайте коммит.

git add <имя файла. >

git commit -m "Oops, we didn't want this commit"

Чтобы отменить коммит, необходимо сделать коммит, который удаляет изменения, сохраненные нежелательным коммитом.

git revert HEAD -–no-edit

Проверьте историю

git hist

Внесите изменение в файл (добавьте автора проекта).

Выполните

git add <имя файла. >

git commit -m "Add an author comment"

После совершения коммита вы понимаете, что любой хороший комментарий должен включать электронную почту автора. Обновите файл, включив в него email автора.

Выполните

git add <имя файла. >

git commit --amend -m "Add an author/email comment"

Просмотрите историю

git hist

Видно, что оригинальный коммит «автор» заменен коммитом «автор/email»

Переместите созданный ранее файл в другой каталог

mkdir lib

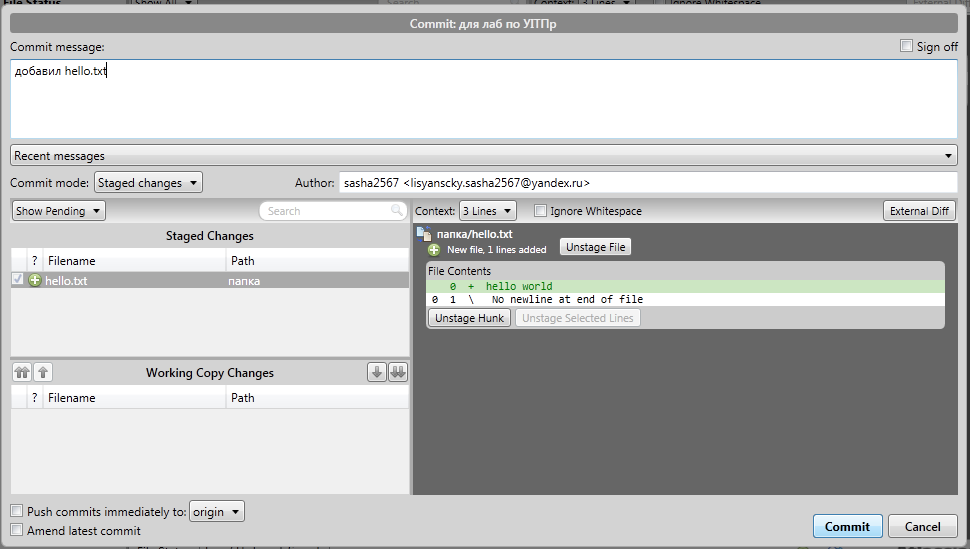
git mv <имя файла. > lib

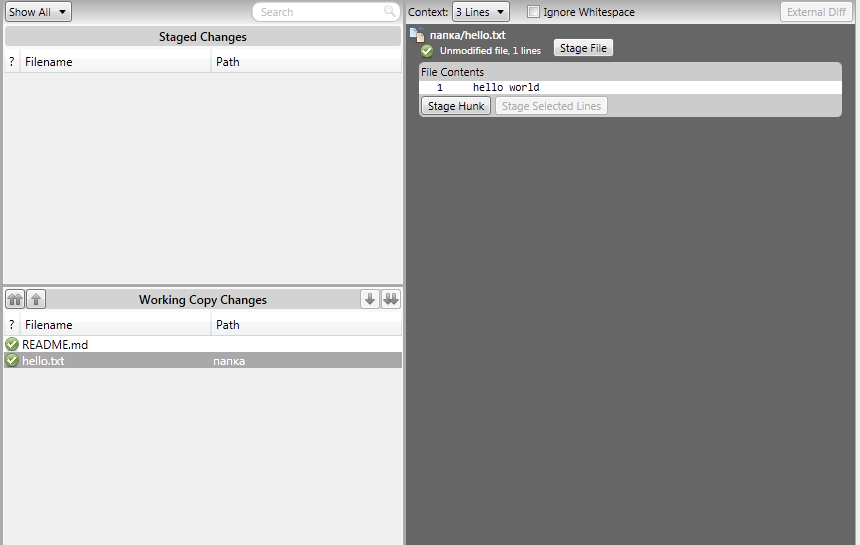
Просмотрите состояние

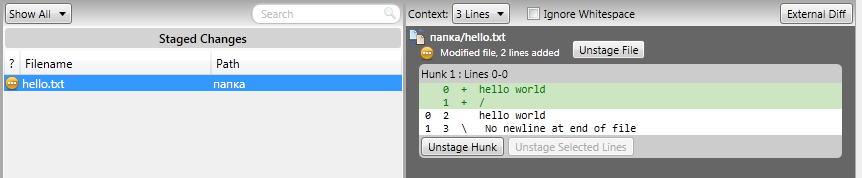
git status

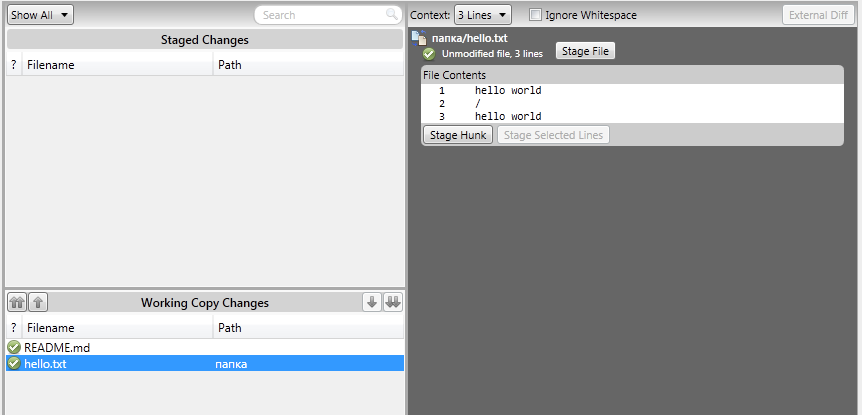
Сделайте коммит этого перемещения

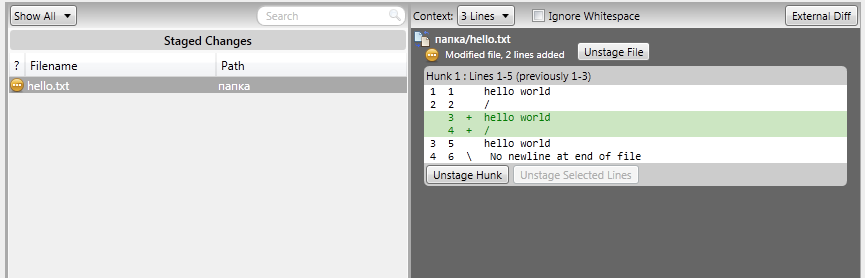
git commit -m "Moved <имя файла. > to lib"

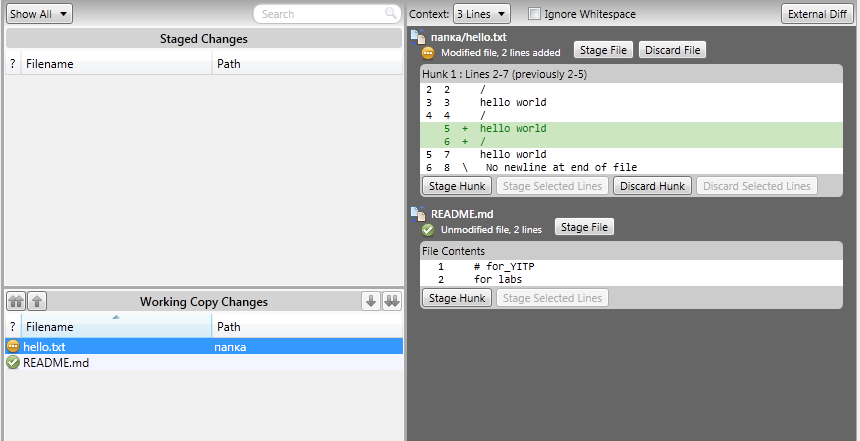
Рисунок 1 – Commit первого файла после его индексации

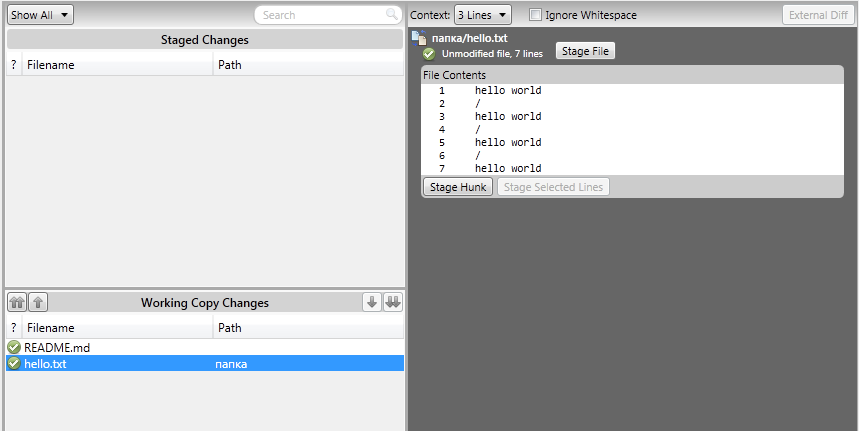
Рисунок 2 – Проверка состояния репозитория

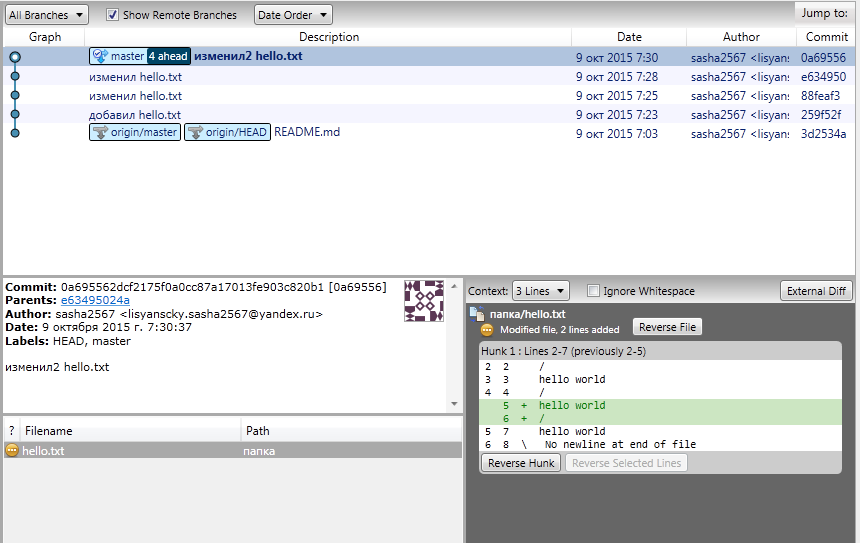
Рисунок 3 – Добавление изменений в файл и их индексация

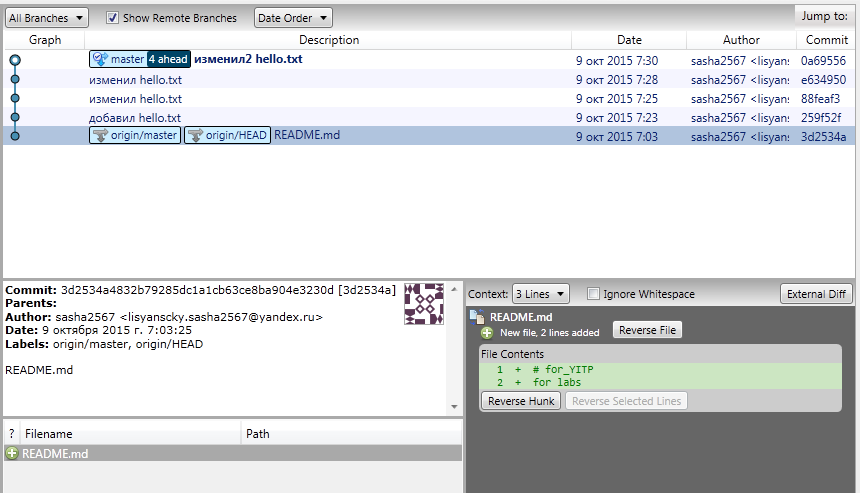
Рисунок 4 – Проверка состояния репозитория

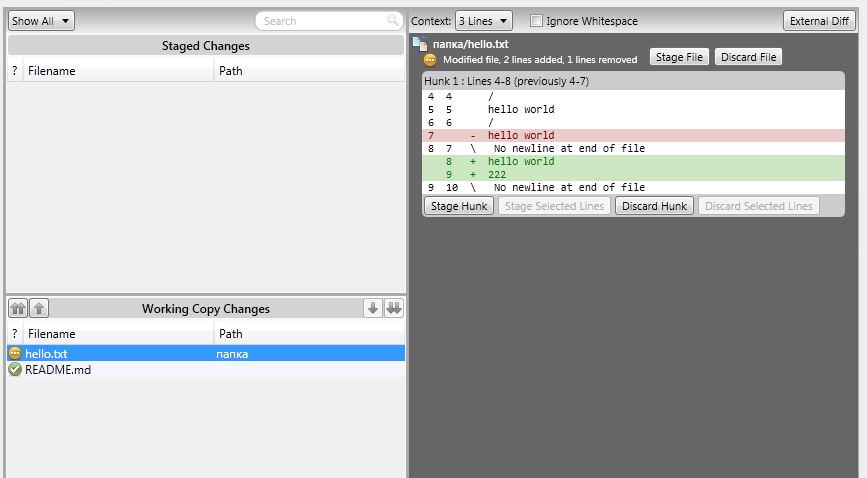
Рисунок 5 - Индексация нового изменения

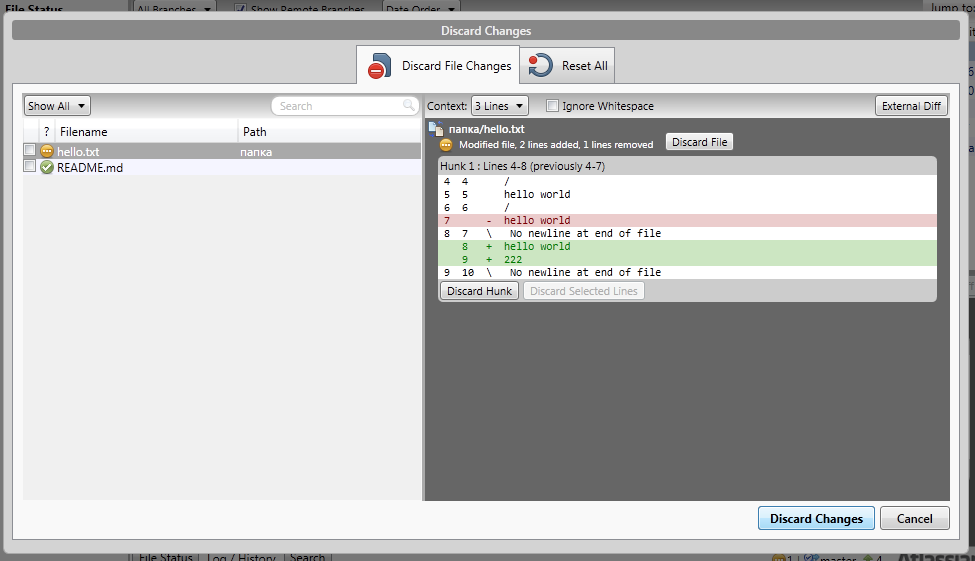
Рисунок 6 - Индексация нового изменения

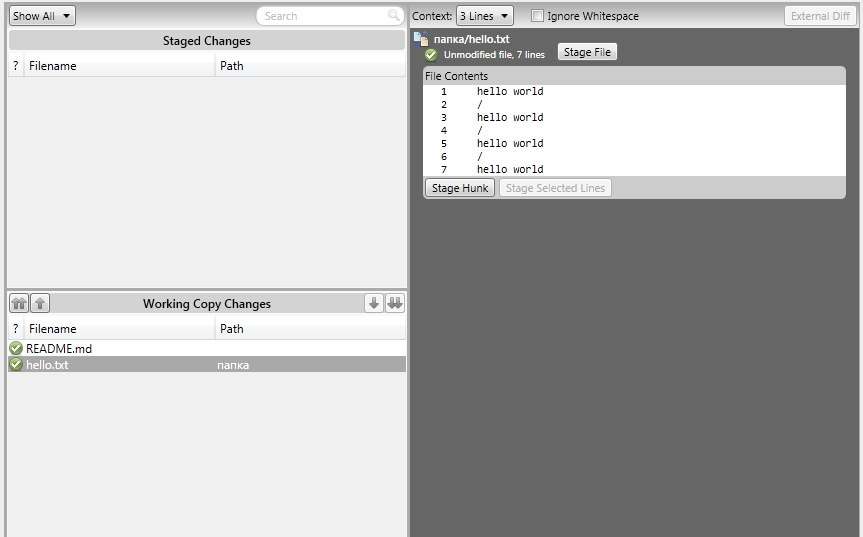
Рисунок 7 - Проверка состояния репозитория

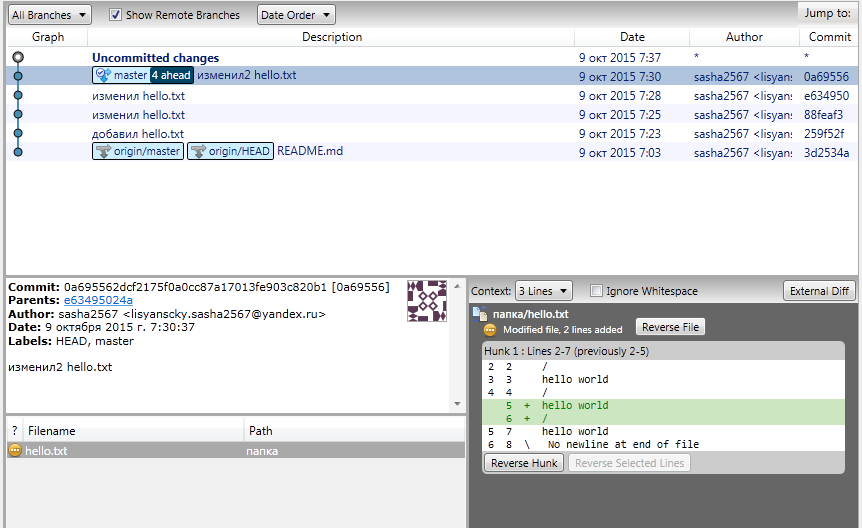
Рисунок 8 – История проекта

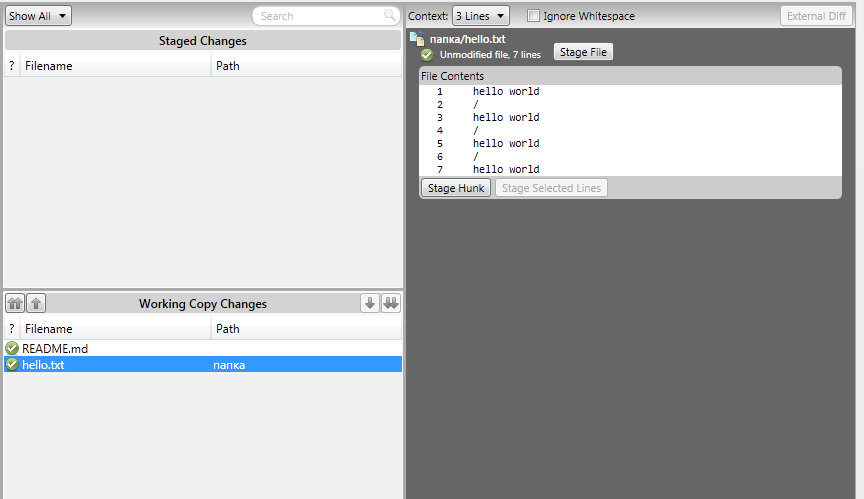
Рисунок 9 – Просмотр первого commitа

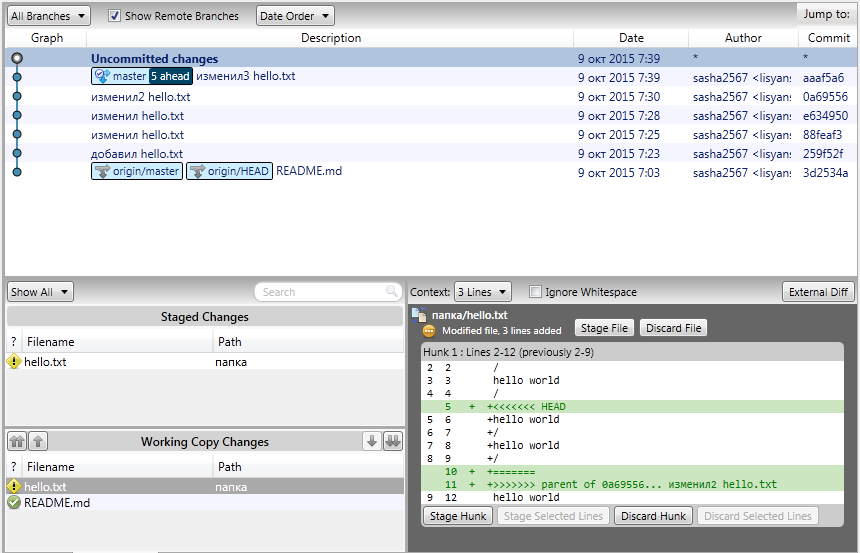
Рисунок 10 - Проверка состояния репозитория

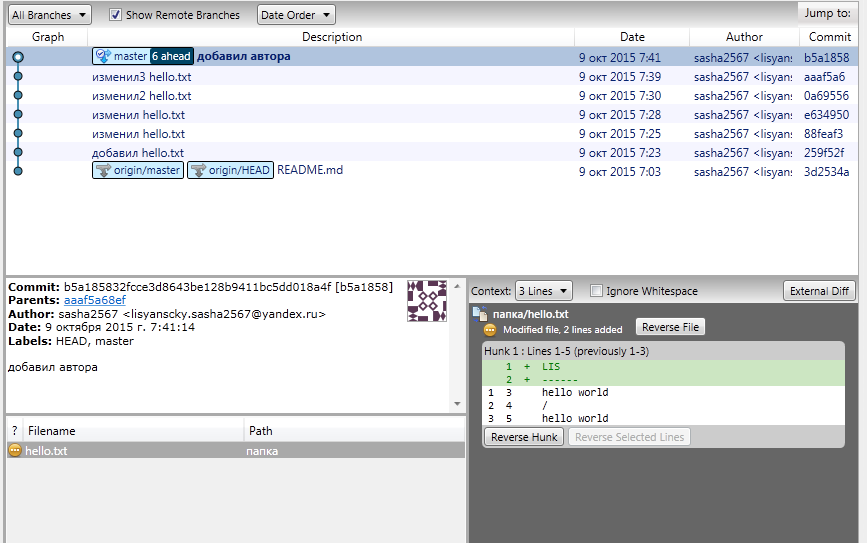
Рисунок 11 – Возврат случайных изменений в файле

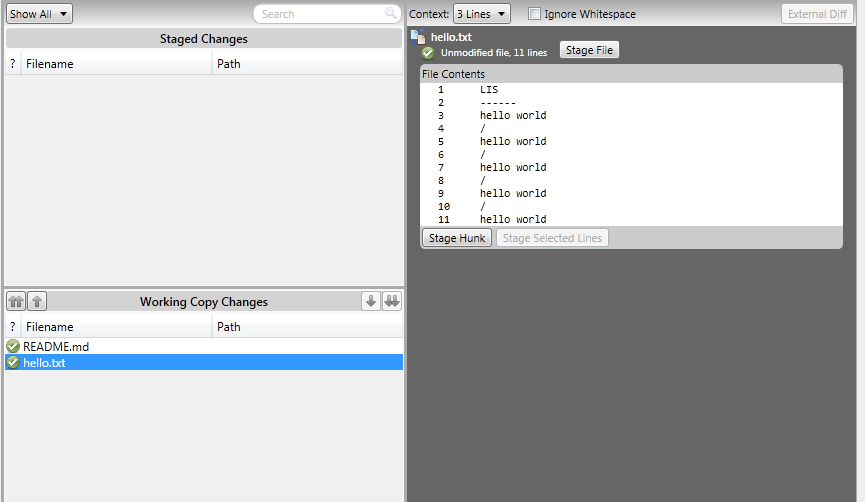
Рисунок 12 - Проверка состояния репозитория

Рисунок 13 – Проверка состояния нежелательного изменения

Рисунок 14 - Проверка состояния репозитория

Рисунок 15 – Просмотр истории после отмены commitа

Рисунок 16 – Просмотр истории

Рисунок 17 - Проверка состояния репозитория

**Вывод**

В ходе выполнения лабораторной работы были проделаны рядовые действия с локальным хранилищем, исследованы графические интерпритации команд гита.