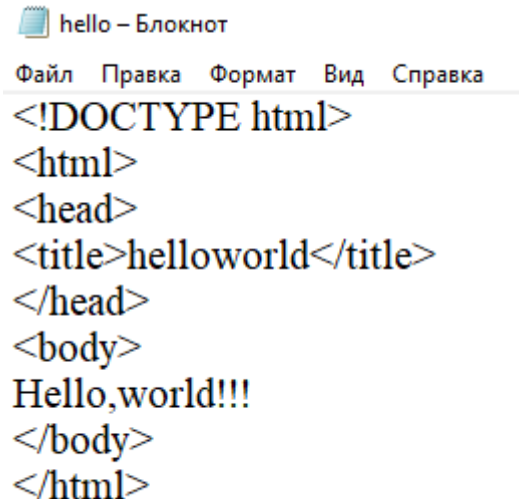


Лабораторная работа №2.

Создание репозитория

Цель: научиться создавать репозитории и работать с ними.

1. Создайте в блокноте файл со следующим содержимым:



```
hello - Блокнот
Файл  Правка  Формат  Вид  Справка
<!DOCTYPE html>
<html>
<head>
<title>helloworld</title>
</head>
<body>
Hello,world!!!
</body>
</html>
```

Сохраните его в папке C:\LabGit\hello под именем hello.html. Этот примерный простой файл нужен для того, чтобы научиться создавать репозитории. В последствии Вы так же сможете работать со своими файлами.

Если Вы уже умеете работать с командной строкой, то можно в GitBash написать следующие команды:

```
mkdir hello
```

```
cd hello
```


```
touch hello.html
```

Данные команды создают папку (каталог) hello, переходят в него и создают там html-документ с именем hello.

2. Научимся создавать репозитории¹. Открываем Git Bash в папке hello. Чтобы создать git репозиторий из этого каталога, выполните команду, “запускающую” Git для данного репозитория:

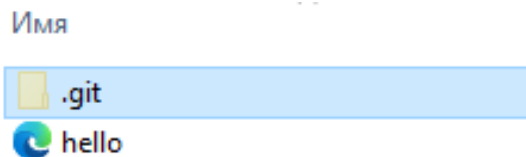
```
git init
```

¹ **Репозиторий** - специальное хранилище файлов и папок проекта, изменения в которых отслеживаются.

 MINGW64:/c/LabGit/hello

```
пользователь@LAPTOP-27F0P5IS MINGW64 /c/LabGit/hello
$ git init
Initialized empty Git repository in C:/LabGit/hello/.git/
```

Чтобы инициализировать репозиторий, Git создает скрытый каталог с именем .git. В этом каталоге хранятся все объекты и ссылки, которые Git использует и создает как часть истории вашего проекта.



3. Теперь добавим в репозиторий созданную страницу hello.html.

```
git add hello.html
```

```
git commit -m "First Commit"
```

```
пользователь@LAPTOP-27F0P5IS MINGW64 /c/LabGit/hello (master)
$ git add hello.html
```

```
пользователь@LAPTOP-27F0P5IS MINGW64 /c/LabGit/hello (master)
$ git commit -m "First Commit"
[master (root-commit) 69664c1] First Commit
1 file changed, 9 insertions(+)
create mode 100644 hello.html
```

Первая команда добавляет файлы в индекс², то есть сделает указанный файл готовым для коммита³. Вторая команда создает новый коммит с файлами из индекса. Коммит хранит не только снимок (все индексированные файлы) репозитория, но и имя автора со временем, что бывает полезно.

Если у вас несколько файлов, то просто перечисляем, например, `git add file1 file2`.

`git add .` — сделать все измененные файлы готовыми для коммита

² **Индекс в Git** — это специальная промежуточная область, в которой хранятся изменения файлов на пути от рабочей директории до репозитория. При выполнении коммита в него попадают только те изменения, которые были добавлены в индекс.

³ **Один коммит** — это пакет изменений, хранящий информацию с добавленными, отредактированными или удалёнными файлами кода.

`git add '*.txt'` — добавить только файлы, соответствующие указанному выражению

`git add --patch filename` — позволяет выбрать какие изменения из файла добавятся в коммит

`git reset [file]` — убрать файлы из индекса коммита (изменения не теряются)

`git diff --staged` — показать, что было добавлено в индекс с помощью `git add`, но еще не было закоммичено

4. Чтобы удалить файл из репозитория используют команду `git rm`.

Эта команда физически удалит файл с диска. Поэтому чтобы Git перестал отслеживать файл, но он остался на диске, используют команду:

`git rm --cached имя_файла`

`git rm [file]` — удалить файл из рабочей директории и добавить в индекс информацию об удалении

`git mv [file-original] [file-renamed]` — изменить имя файла и добавить в индекс коммита

5. После создания репозитория и коммита следует проверить текущее состояние репозитория, используя команду:

`git status`

```
пользователь@LAPTOP-27F0P5IS MINGW64 /c/LabGit/hello (master)
$ git status
On branch master
nothing to commit, working tree clean
```

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

Добавив ключ `-s` можно сделать вид изменений более кратким:

`git status -s`

Примечание: Чтобы удобнее было писать команды, используют горячие клавиши. Их можно изучить, перейдя по ссылке <https://gist.github.com/dev->

<pwa/2936e3b9c9afa669be0ab812ad148f85>. Например, стрелка вверх – предыдущая команда.