Лабораторная Работа №3

# Цель работы: Изучение системы контроля версий Git.

В качестве системы контроля версий для данной лабораторной работы будем использовать git. Git – распределённая система управления версиями файлов.

Как и в случае с Mercurial, по сути дела, git-репозиторий отличается от любого другого только папкой, в данном случае – .git . Все остальные файлы и папки – это просто файлы и папки, такие же, как и все остальные на рабочей машине.

Для того, чтобы получить git-репозиторий (под которым подразумеваются все файлы и папки + папка .git), необходимо выполнить git clone

После этого, в рамках данной лабораторной работы, вам необходимо будет сделать свою ветку. Это делается при помощи команды git checkout -b style

ПРИМЕЧАНИЕ**:** git checkout -b <имяветки> является шорткатом для git branch <имяветки> за которым идет git checkout <имяветки>.

Далее, вам необходимо будет сделать изменения в файлах, которые после нужно будет добавить себе в ветку. Команда, которая делает это – git commit –m "Здесь сообщение".

Теперь, когда у вас есть весь необходимый набор файлов, их можно запустить в общий репозиторий. Это делается командой git push origin <имя ветки>.

Комбинация git commit – git push продолжается до тех пор, пока в вашей ветке не закончена работа. Если она закончена, то теперь нужно как-то синхронизировать вашу ветку с общей (традиционное название – master).

Для этого вам нужно:

1. Перейти в ветку master
2. Забрать последние изменения ( git pull ( git pull = git fetch + git merge ))
3. Соеденить свою ветку с мастер-веткой: git merge <имя ветки>, например, git merge task1
4. Отослать изменения на сервер: git push origin master

Также следует понимать, что только команда git push меняет состояние другого репозитория; все остальные команды меняют только ваш репозиторий.

Итого, цикл работы с git выглядит следующим образом:

1. git clone – создаёт копию проекта на локальной машине разработчика;
2. работа непосредственно с файлами (в default или, в другой ветке – git checkout branch\_name)
3. git pull – забрать изменения, которые были сделаны разработчиками во время, прошедшее с момента git clone
4. git add – добавляем файлы в тот планируемый commit
5. git status – просмотр всех изменённых и добавленных файлов
6. git commit – создание пакета со всеми staged файлами
7. git push – отправка пакета на сервер

Если у вас возникают вопросы в процессе работы с git, то вы можете воспользоваться командой git --help, которая позволит вам узнать наиболее часто встречающиеся функции; также вы можете воспользоваться справочником по отдельным командам. Так, в частности, команда git checkout --help откроет браузер с информацией об использовании checkout, вместе с параметрами и примерами.

## Задание:

Необходимо исправить и дополнить html-сайт. Для начала работы склонируйте репозиторий по ссылке, предоставленной преподавателем.

Требования:

1. Перед началом работы создать свою ветку; всю работу вести в ней.
2. Когда всё готово, выполняется мерж с мастер-веткой.
3. Каждую строчку изменений в файле, если правки отличаются по смыслу, разбейте на разные коммиты. Пример – правка ссылки на логотип и правка названия – в разные коммиты; исправление трёх одинаковых ссылок на три разные – в один.
4. Каждый коммит необходимо сопроводить качественным сообщением.

Варианты:

1. index.html – проверить логотип, название, а также все ссылки.

Во всех остальных случаях требуется добавить информацию о кофе (два разных параграфа) и логотип, а также поменять название у странички. Пример – 1.html – Jacobs.

1. 2.html – Nescafe
2. 3.html – Tchibo
3. 4.html – Чёрная карта
4. 5.html – Жокей
5. 6.html – Ambassador
6. 7.html – MacCoffee
7. 8.html – Fort
8. 9.html – Carte Noire
9. 10.html – Jardin
10. 11.html – Lacomba
11. 12.html – Café Pele