Урок 1-2. Общее знакомство с GIT.

**Git** необходим для разделения проекта на ветки (ветвления). Он отслеживает каждый этап разработки проекта с помощью коммитов.

Деятельность при разработке проекта принято реализовывать на нескольких ветках - **branch** (*параллельных линий разработки*), чтобы при допущении ошибок в одной версии проекта, не пострадала его другая версия.

**Gitflow - стандартная модель ветвления**, включает в себя 5 веток:

1. **Master** – главная ветка, в которую сливаются все остальные. Должна содержать исключительно рабочую версию проекта;
2. **Develop** – ветка для разработки, служащая основой для функциональных веток. Должная содержать последние разработки.
3. **Feature** - ветка для разработки новых функций. Создается от ветки develop и сливается обратно в нее, когда функция завершена.
4. **Hotfix** - ветка для быстрого исправления ошибок в рабочей версии проекта. Создается от ветки master и сливается обратно в нее и develop, когда ошибки исправлены.
5. **Release** - ветка для подготовки нового продуктового релиза. Создается от ветки develop и сливается обратно в нее и master, когда релиз готов.

Урок 3. Коммиты и репозиторий.

**Сommit** (коммит) – точка сохранения проекта.

Работа с commit имеет 3 этапа:

1. Изменение файлов проекта
2. Добавление нужных файлов проекта в область caching files (область подготовки перед отправкой файлов на ветку)
3. Сам коммит. Помещается в репозиторий

**Репозиторий** – хранилище версий (коммитов) проекта.

Урок 4. Практика с Git в терминале

cd «название\_папки» - переход к указанной папке. Без указания папки перейдет в корневой каталог.

ls – отображение файлов текущей папки.

ls -a - отображение в т.ч. скрытых файлов текущей папки.

git init – создание репозитория из текущей папки.

git status – отображение файлов возможных к коммиту (записи в репозиторий) и их статусов.

git add . – добавление всех файлов в очередной коммит. При указании вместо точки названия файлов, добавлен в очередной коммит будет только он.

git commit -m «сообщение» - создание коммита в текущей ветке

git log – отображает все коммиты текущей ветки.

clear - очистка экрана терминала

Урок 5. Ветки

В файл .gitignore помещаются расширения файлов, которые не будут учтены при подготовке коммита.

git branch – отображение всех веток и пометка текущей ветки

git branch «название\_ветки» - создание и добавление ветки

git checkout «название\_ветки» - переход на указанную ветку. Ветка, созданная удаленно, начнет отображаться только после перехода.

git checkout -b «название\_ветки» - создание и переход на новую ветку

git branch -d «название\_ветки» - удаление указанной ветки

git merge «название\_ветки» - слияние указанной ветки с текущей

Урок 6-7. Удаленный репозиторий

Кнопка fork в GitHub позволяет добавить указанный репозиторий в свой профиль GitHub.

git remote add origin URL – cвязь текущего локального репозитория с удаленным по указанному URL.

git push «название\_репозитория» «название\_ветки» - отправка данных локального репозитория в удаленный. git знает название своего текущего репозитория по ключ.слову origin

git pull – получение коммитов созданных удаленно

Урок 8. Конфликт слияния

git mergetool -t opendiff – показывает все файлы, которые участвуют в конфликте при попытке их слияния