Министерство образования и науки РФ

Институт информационных технологий и управления в технических системах

Севастопольский государственный университет

Лабораторная работа №7

по дисциплине: «ТСПП»

«**Исследование распределенных системы контроля версий Git при**

**коллективной разработке программных продуктов**»

Выполнил:

ст.гр. ИСб/22о

Воронин И.Ю.

Проверил:

Дрозин А.Ю.

г. Севастополь

1. Цель работы

Исследовать основные подходы к организации взаимодействия команды разработчиков с использованием распределенной системы контроля версий (DVCS). Приобрести практические навыки установки и настойки DVCS Git, организации ветвей разработки и осуществление слияния.

1. Вариант задания

2.1. Разработать модель командной работы согласно варианту, полученному у преподавателя.

2.2. Создать необходимое количество репозиториев, разработать соглашение по предназначению репозиториев.

2.3. Создать изменения в одном локальном репозитории, сохранить их в удаленном.

2.4. Получить набор изменений из удаленного репозитория в репозиторий отличный от описанного в п.2.3, внести дополнительные изменения и сохранить их в удаленном репозитории.

2.5. Внести одновременно разные изменения в локальные репозитории сохранить их все в удаленном, продемонстрировать процесс слияния.

2.6. Продемонстрировать создание именованных веток в локальном репозитории.

2.7. Проанализировать результаты работы, сделать выводы.

3. Ход работы

Создаём в аккаунте на GitHub новый репозиторий, для хранения данных о проектах. После этого открываем SmartGit, создаём новый локальный репозиторий и привязываем его по адресу центрального репозитория GitHub: <https://github.com/Angainor17/VoroninIgor>.

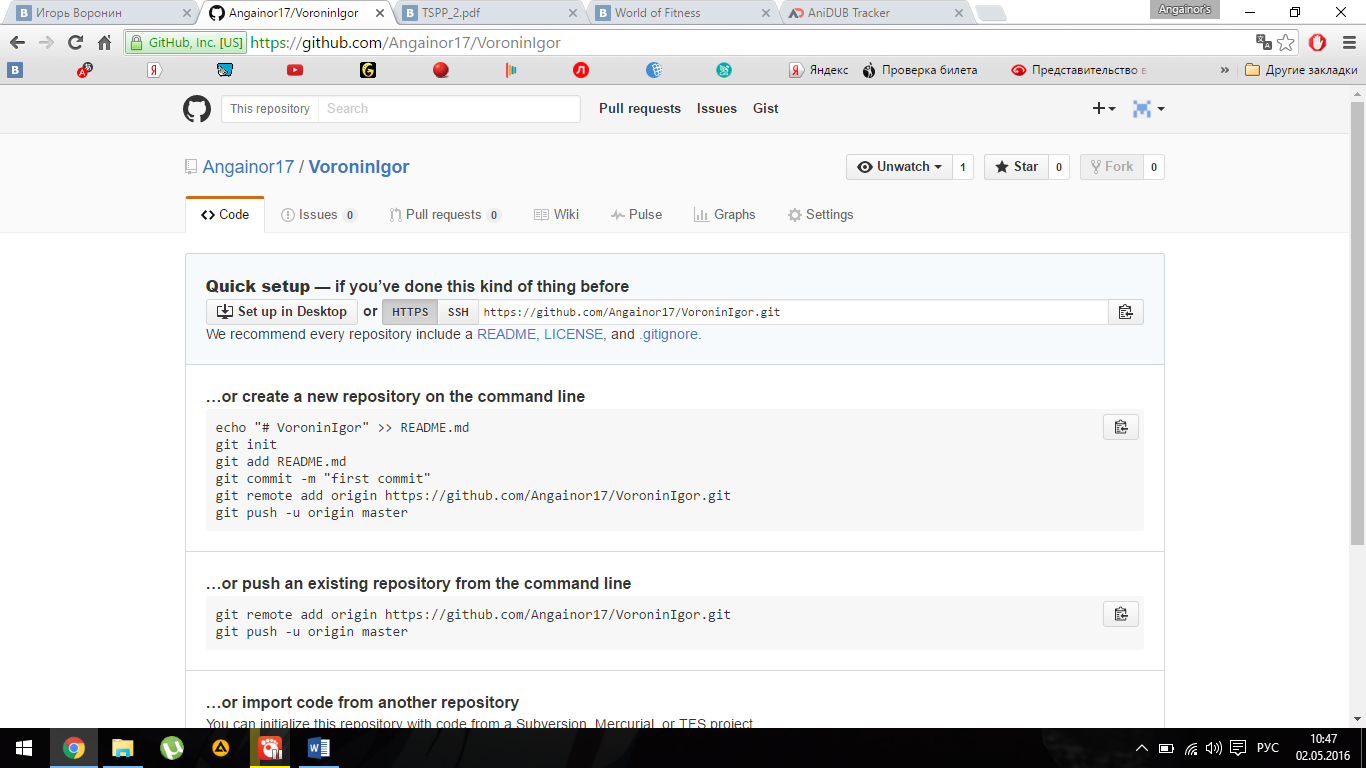


Рисунок 3.1 – Репозиторий GitHub.

Далее, при помощи команды Commit, мы добавляем в локальный репозиторий все данные, которые находятся в обозначенной папке, но ещё не были добавлены. При этом всегда необходимо указывать Commit Message, в котором» должна быть аннотация к данному действию. После этого, при любой их корректировке среда будет отображать изменённые файлы в окне Files, а в окнах Changes будут подсвечены изменения внутри самого файла.

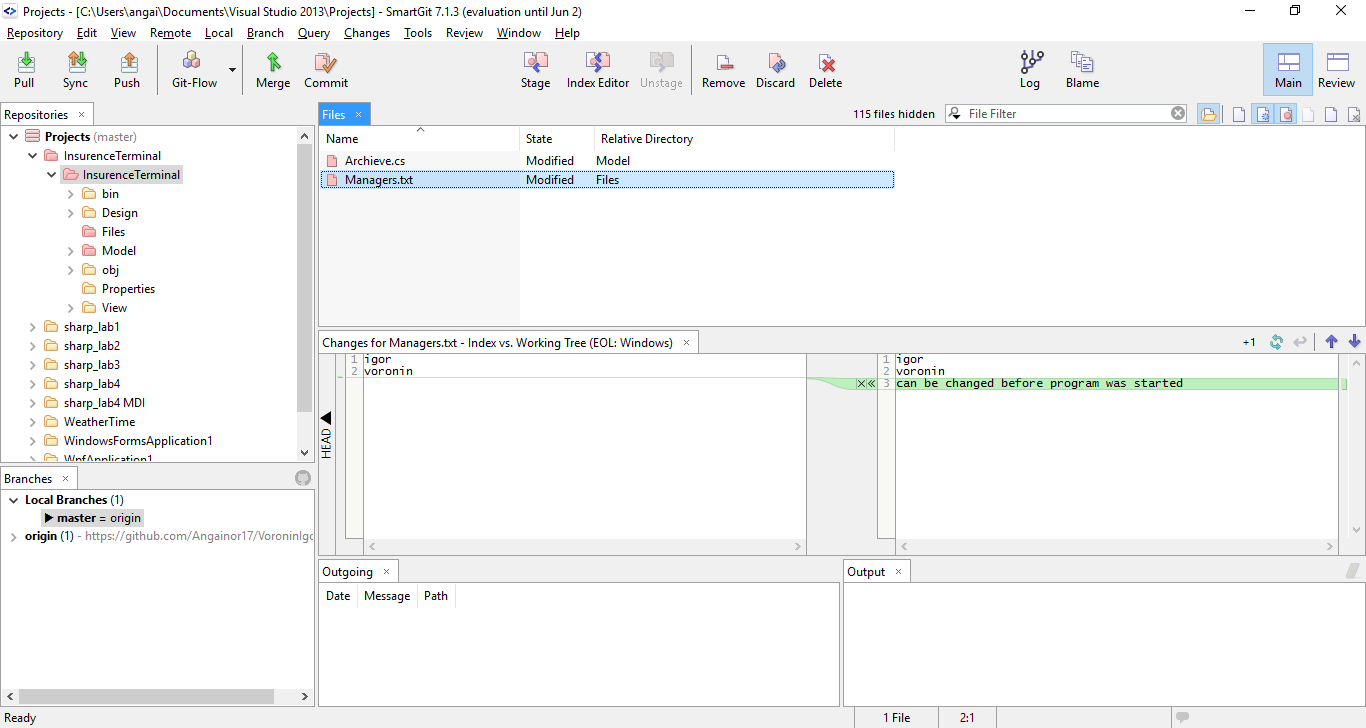


Рисунок 3.2 – Модифицированные файлы.

В случае, если необходимо сохранить файлы с локального репозитория на центральный, то необходимо использовать функцию Push, которая при указании точного адреса репозитория, логина и пароля автоматически поместить данные в хранилище, которое можно просмотреть на сайте GitHub в личном кабинете.

При помощи функции Pull можно получить данные из удалённого репозитория.

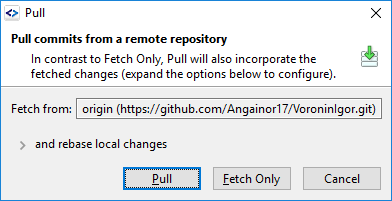


Рисунок 3.3 – Операция Pull.

SmartGit также обеспечивает удобное манипулирование Ветвями разработки.

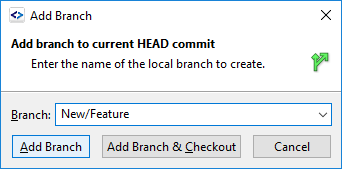


Рисунок 3.4 – Добавление Ветвления.

При наличии более одного ветвления можно переключаться между ними двоёным кликом мыши. Тем самым локальный репозиторий будет принимать различные вариации. Изначально, при создании нового ветвления, оно будет идентично предыдущему, из которого вышло.

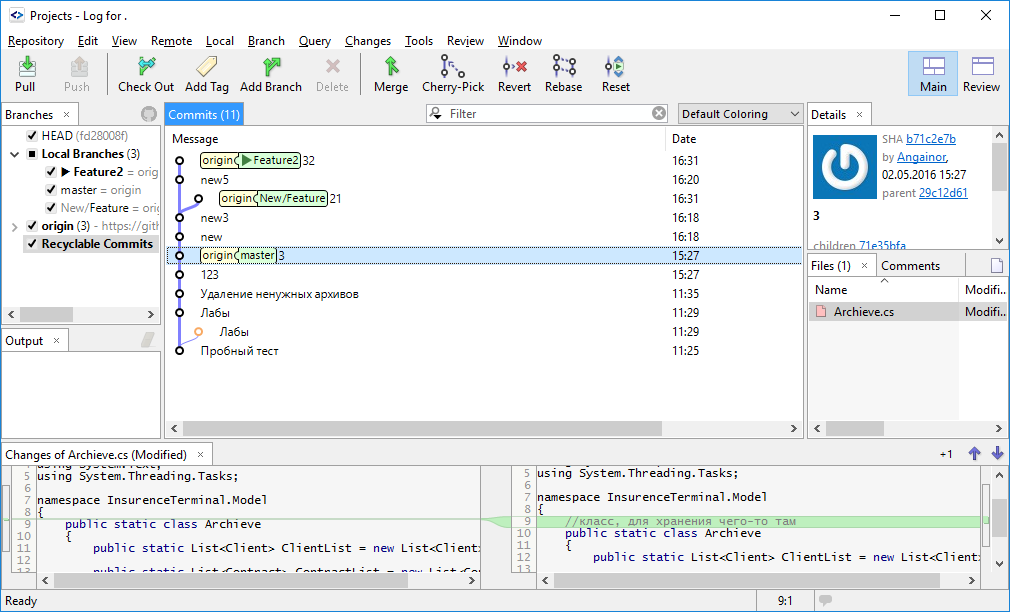


Рисунок 3.5 – Ветвления.

ВЫВОДЫ

В данной лабораторной работе была изучена распределённая система контроля версий Git на примере GitHub и SmartGit. Были использованы ветвления, локальный и центральный репозиторий. На GitHub были помещены все данные о программе, заданной по варианту.