**Лабораторная работа 11 «Настройка работы системы контроля версий (типов импортируемых файлов, путей, фильтров и др. параметров импорта в репозиторий)»**

**Цель занятия**: научиться устанавливать, настраивать и работать с репозиторием GitHub.

**Оборудование, технические и программные средства:** персональный компьютер, программа Visual Studio 2019.

**Время выполнения:** 90 мин

***КРАТКАЯ ТЕОРИЯ И МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ:***

Управление версиями — это система для управления файлами с множеством разных версий. При разработке программного обеспечения с ней обычно работает большое число разработчиков. Основная цель любой системы управления версиями (VCS) заключается в том, чтобы найти решение, позволяющее всем пользователям одновременно работать с базой кода.

Основой системы управления версиями является репозиторий, который выступает в качестве центрального хранилища данных для любых файлов, что похоже на файловый сервер. Однако в отличие от файлового сервера репозиторий хранит весь журнал проекта и все внесенные изменения.

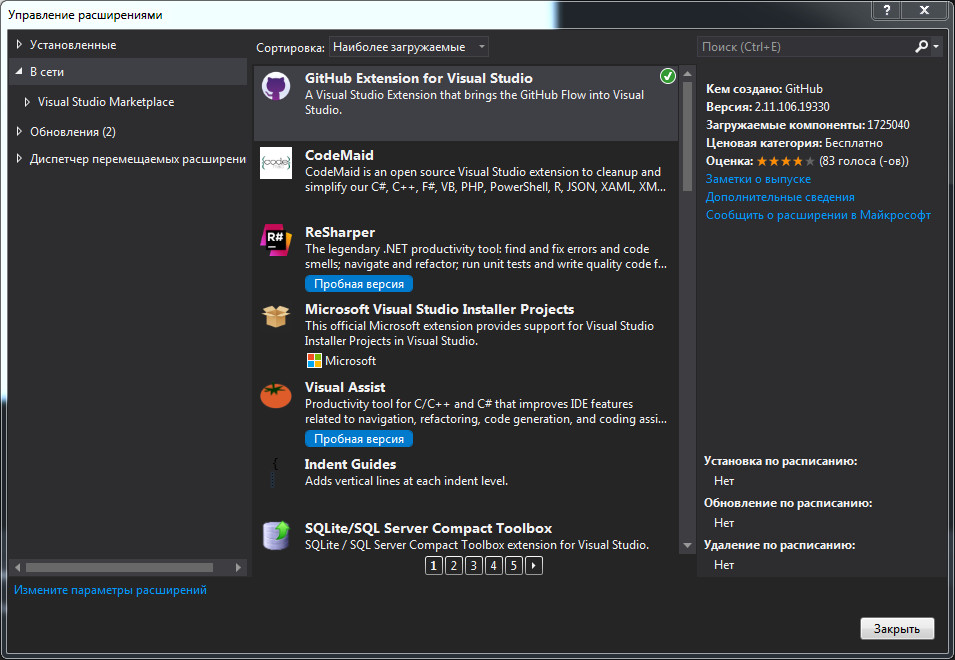
Если репозиторий является центральным хранилищем данных, логично, что каждый пользователь имеет локальное хранилище, с которым может работать. Оно называется рабочей копией. В Visual Studio для Mac рабочая копия выглядит так же, как и любой локальный каталог на компьютере, что позволяет считывать и записывать файлы в нем. Но так как Visual Studio для Mac включает возможность интеграции с системой управлениями версиями, Subversion и Git можно использовать, не выходя из интегрированной среды разработки.

Subversion — это централизованная система управления версиями, то есть все файлы и редакции находятся на одном сервере, откуда пользователи могут получить для изменения любую версию любого файла. Если файлы извлекаются из удаленного репозитория Subversion, пользователь получит моментальный снимок этого репозитория на соответствующий момент времени.

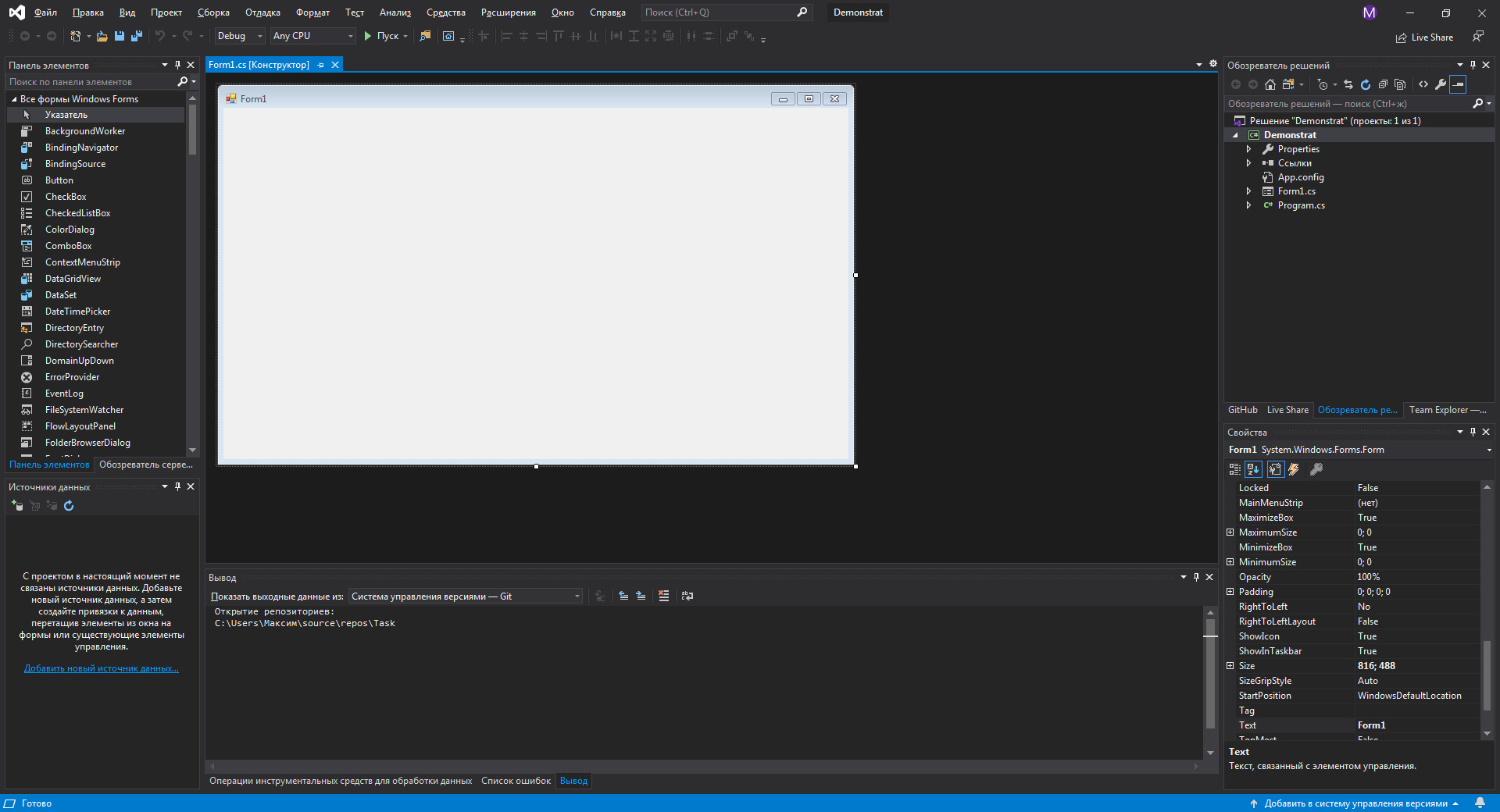
Git — это распределенная система управления версиями, которая позволяет командам одновременно работать с одним документом. В случае с Git все файлы могут храниться на одном сервере, но при каждом извлечении репозитория из этого центрального источника весь репозиторий клонируется локально на ваш компьютер.

***ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И ФОРМА ОТЧЕТНОСТИ:***

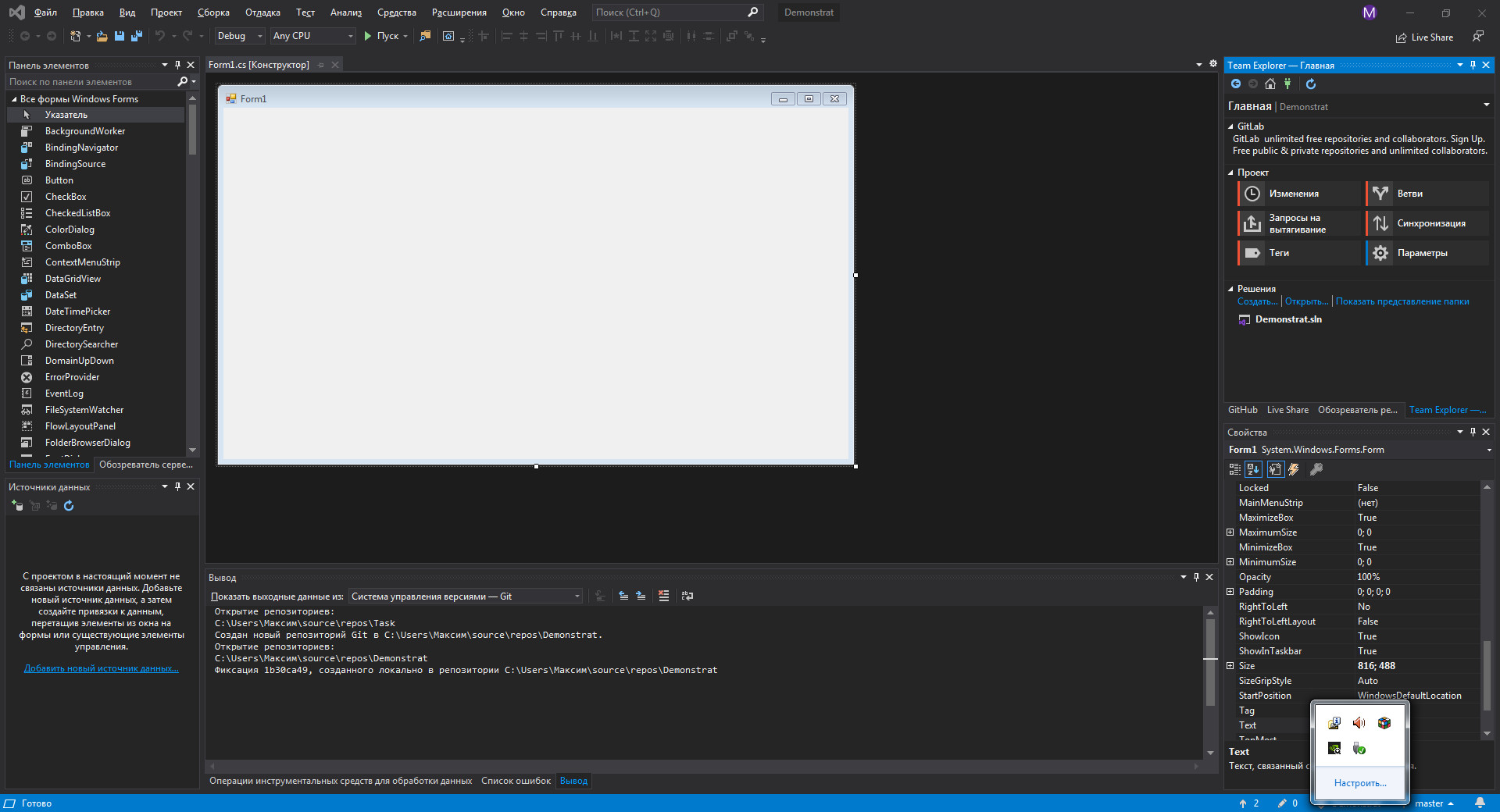
1. Первым делом необходимо установить Расширение GitHub Extension for Visual Studio.



1. Снизу должна появится строчка «Добавить в систему управления версиями».



1. Нажимаем на нее и добавляем проект в систему Git.
2. Основное окно работы (значок домика)

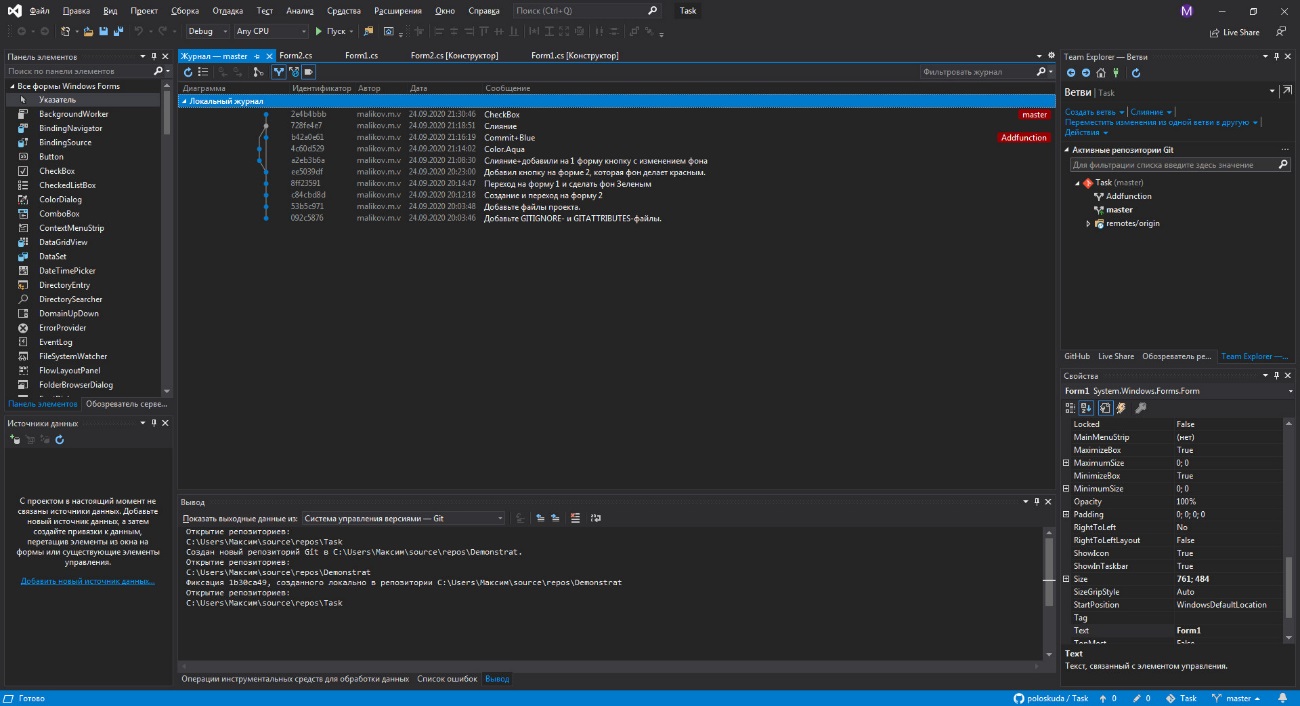


Изменения – сделать записать (commit)

Ветви – можно создать ветвь либо объединить.

Синхронизация – синхронизировать данные локального репозитория и Интернет репозитория.

Так же эти действия дублируются в самом низу панели, просмотрите их. Так же там находится одна из важнейших функций View History. Пример истории рис. Ниже.

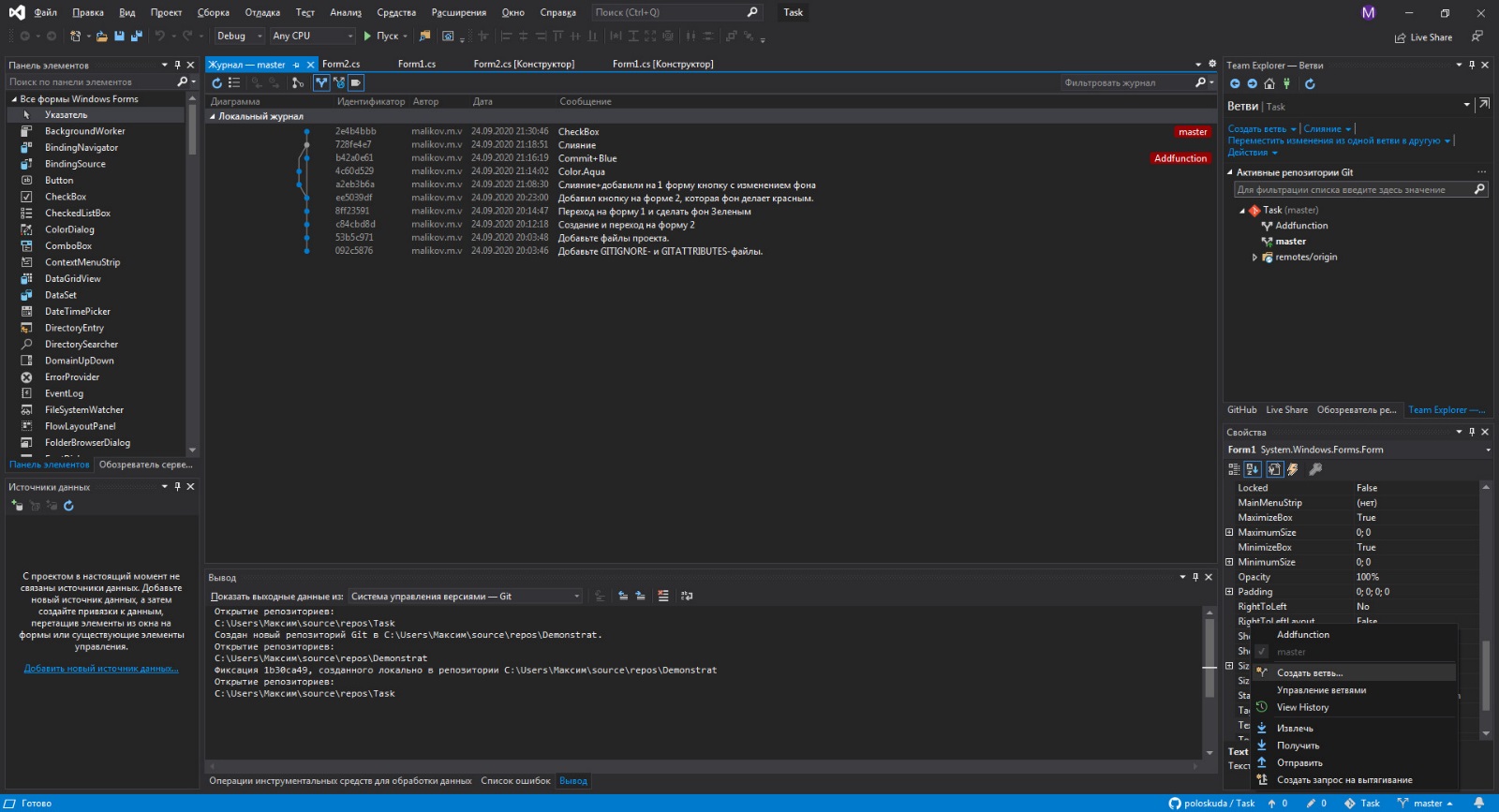


При написании какой-либо части когда, можно делать commit (и не забываем делать комментарии, короткие но понятные). Каждый commit будет отображаться в Истории, так же можно будет посмотреть на каком этапе что было изменено и вернуться обратно, либо начать ветвление с конкретного этапа, хоть вы и ушли далеко вперед.

1. Давайте напишем небольшую программу для наглядного примера. Допустим переход с одной формы на другу, добавим пару кнопок, и текста. Не забываем каждую часть когда commit.
2. Теперь можно попробовать добавить ветвь. Например решили попробовать другой способ решения, а тот что сейчас чтобы не комментировать и не удалять. Либо когда работает команда разработчиков, можно было над 1 проектом работать нескольким людям. Разделяются на задачи, и решает каждый свою, когда они все сделают, можно будет собрать решение целиком.

Снизу на панели нажимаем на master и выбираем Создать ветвь…вводим имя и жмем создать.

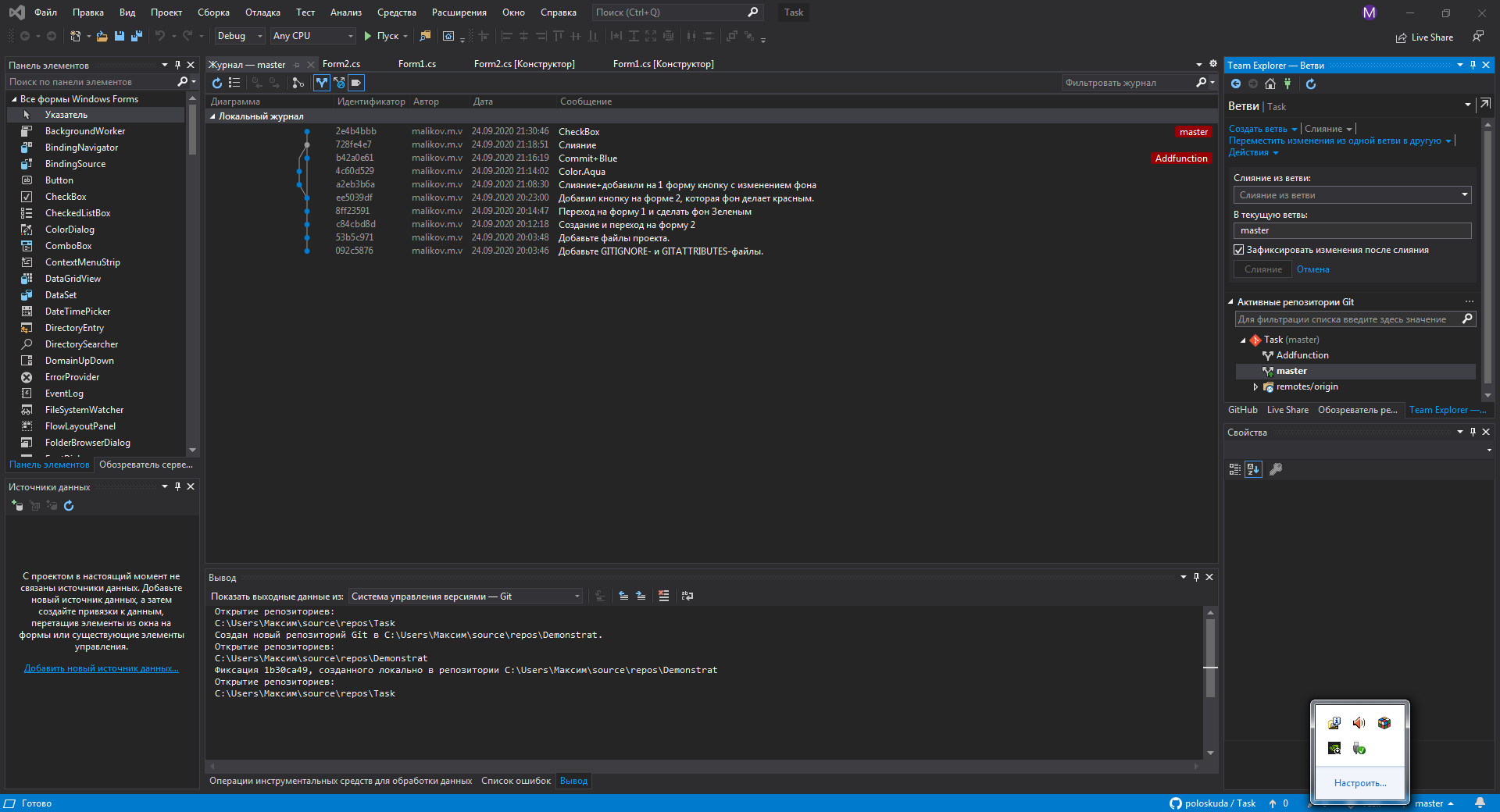
В дальнейшем можно будем менять ветвь и работать сразу в 2 направлениях, заметьте проект 1, но изменения в 1 ветви (master) ни как, не затрагивают проект во 2 ветви (Addfunction).



1. Попробуйте добавить какой-либо функционал во 2 ветвь. (Не забываем постоянно делать commit) Смотрим результат в View History и наблюдаем различия в проектах.

Между ветвями можно быстро перемещаться, просто выбрав нужную ветвь на панели снизу.

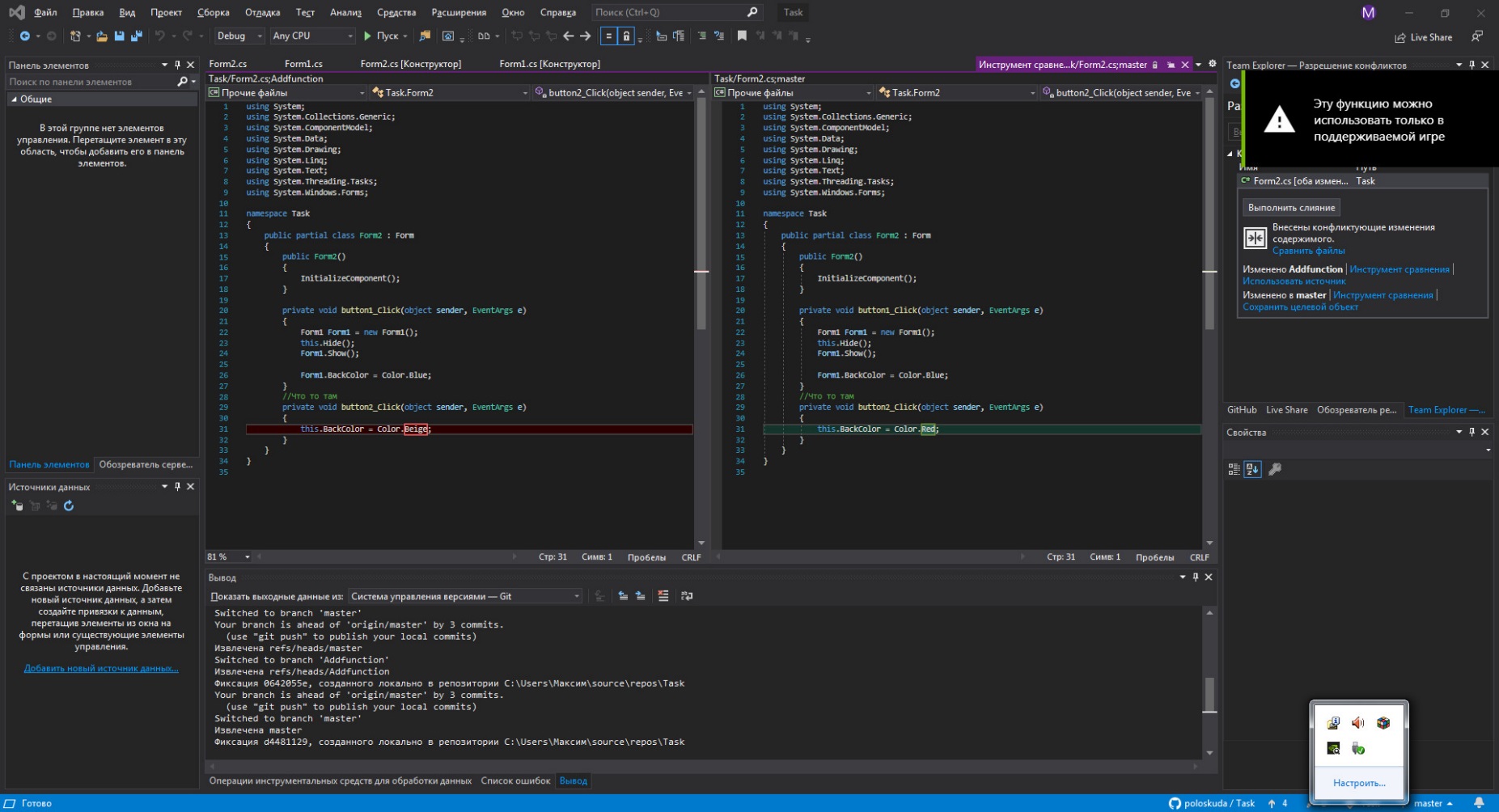
1. Так же ветви можно объединять. Допустим в 1 ветви сделана авторизация, во 2 ветви сделан тест.



На примере: если выбрать слияние из ветви (Addfunction) в ветвь (master), то получится следующее: в ветви master будет находится проект master+addfunction,а в addfunction только он сам.

**Проверьте функцию слияния.**

1. При слиянии может произойти конфликт, такое происходит в тех случаях, когда в 1 и 2 ветви в одной и тоже строке присутствуют различия, например:

****

Нам будет предложено какой из вариантов ветвей оставить, либо оставить и тот и другой вариант.

**Использование Интернет репозитория GitHub.**

1. Необходимо зарегестрировать на сайте <https://github.com/>
2. В Visual Studio зайти под своим Login/Password от GitHub.
3. Выбираем пункт Синхронизировать, и тут 2 варианта:
4. Прикрепить к уже созданному репозиторию вставит URL адрес.
5. Опубликовать в GitHub.

Мы выберем 2. Нажимаем Опубликовать в GitHub. Можем добавить описание по желанию и жмем опубликовать. Проверить можем зайти на сайт GitHub, там должен появится наш репозиторий со всей историей и комментариями.

**Задание:**

1. Научиться удалять репозитории.
2. Скачивать чужие репозитории.
3. Разделиться по 2 человека, создать 1 репозиторий. Разработать программу таким образом, чтобы проверить возможности GitHub на слияние ветвей при работе в команде.