**Частное учреждение образования «Минский колледж предпринимательства»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №23**

**по учебному предмету**

*technicaldocs.ru*

*Подпись и дата*

*Взам. инв. №*

*Инв. № дубл.*

*Инв. № подл.*

*Подпись и дата*

**«ТЕХНОЛОГИЯ РАБЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | Выполнили учащиеся 3 курса группы П-2007 |
|  |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д.В. Рудаковский  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_И.А. Скриндевский  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.В. Кисель  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М.Ю. Рубченя  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М.А. Войтеховский |
|  |  |  |
|  |  |  |
|  |  | Преподаватель |
|  |  |  |
|  |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.П.Кислюк |

2022

**Цель работы:**

Изучить базовые возможности систем управления версиями с применением CASE средств.

В материалах рассмотрены принципы управления версиями. Приведены инструкции по работе с версиями программ в среде управления версиями Git. Рассмотрены примеры использования указанных средств. В заключительной части методических указаний приведены контрольные вопросы, список рекомендуемой литературы и пример задания. Ознакомившись с методическими указаниями и разобрав приведенные в нем примеры, студент может получить у преподавателя свой вариант задания и приступить к его выполнению.

1. Если система контоля версий Git не установлена, то установить ее (параметры оставить по умолчанию).
2. Запустить Git GUI или TortoiseGit (или консоль). Создать новый репозиторий (в папке по фамилии студента).



Рис.1

1. Добавить в папку репозитория файлы. Зафиксировать состояние репозитория (выполнить commit).

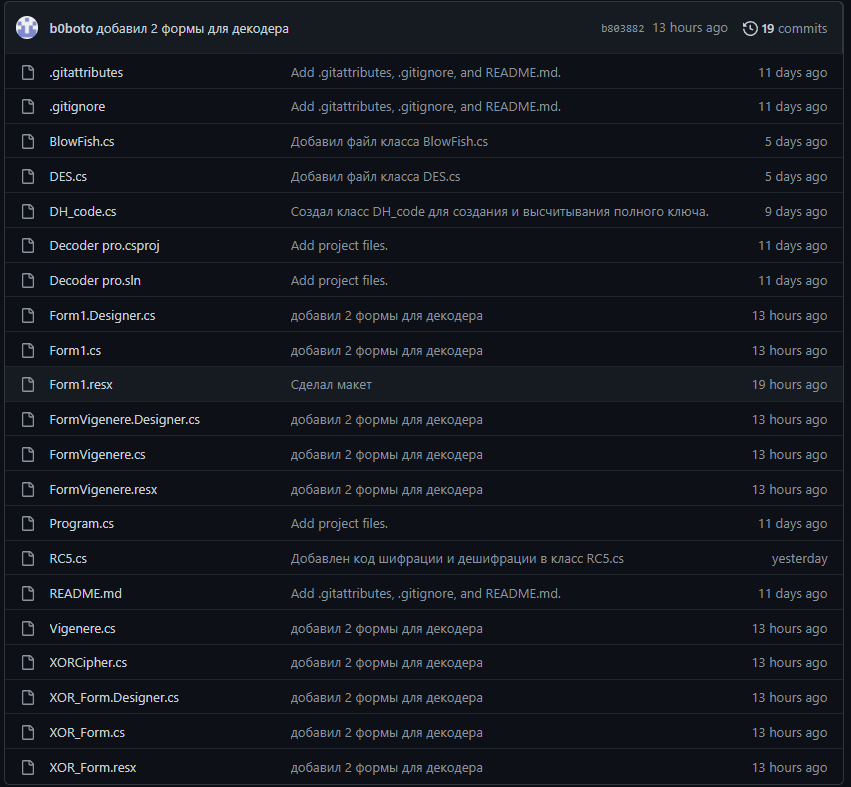


Рис.2

1. Внести изменения в файлы. Зафиксировать новое состояние репозитория.

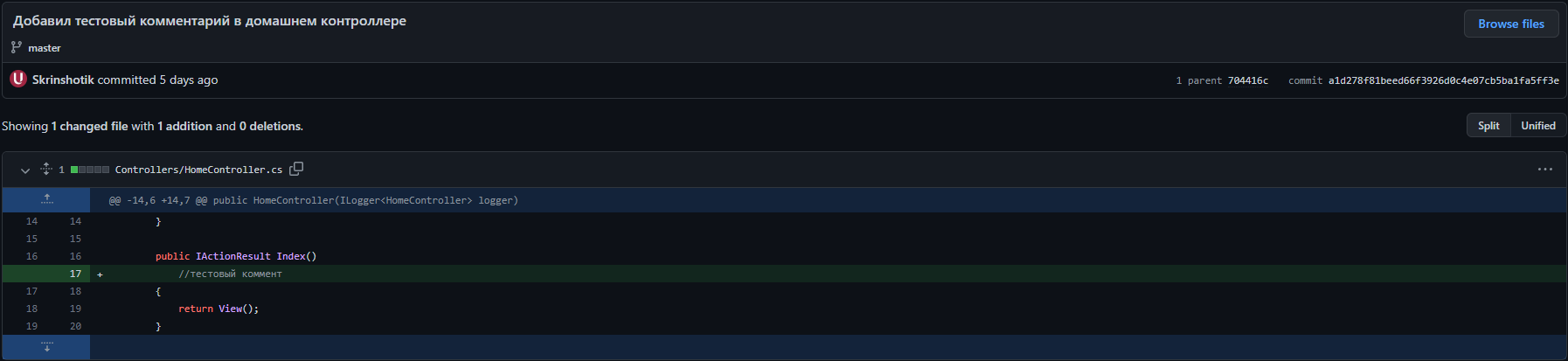


Рис.3

1. Создать новую ветку 1. Внести в нее изменения (добавить новый файл и изменить существующий файл: добавить, удалить и изменить строки) и зафиксировать их.

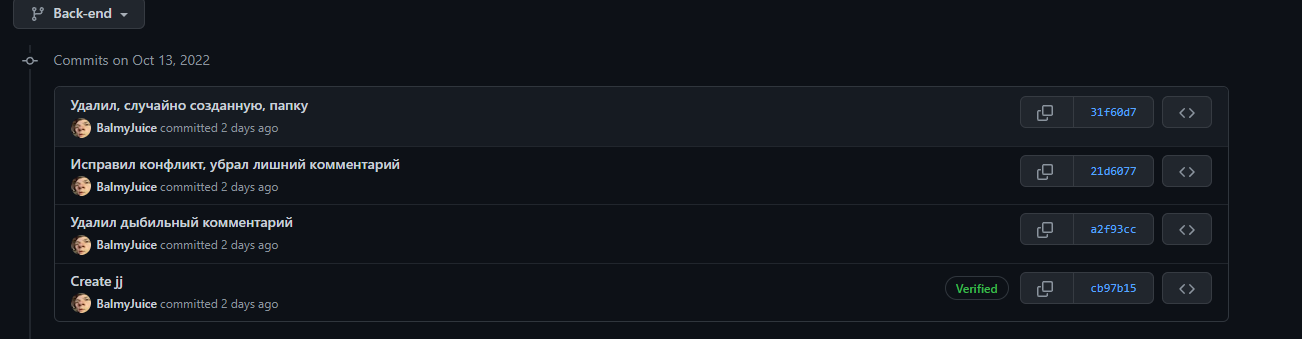


Рис.4

1. Переключиться на ветку мастера. Внести в нее изменения (добавить новый файл; изменить существующие файлы: добавить, удалить и изменить строки первоначального файла) и зафиксировать их.

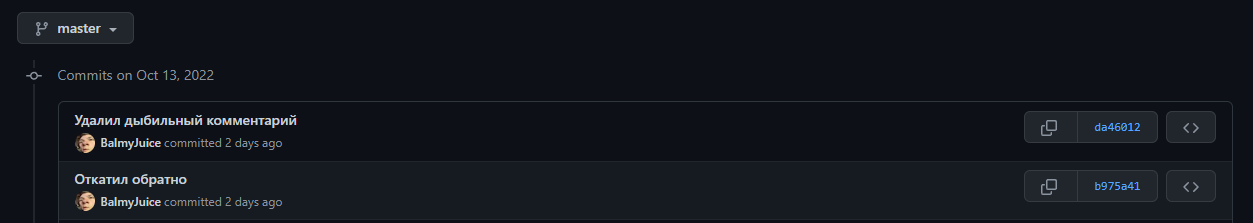


Рис.5

1. Продемонстрировать слияние веток. Разрешить возникший конфликт.



Рис.6

1. Просмотреть дерево изменений веток (историю).

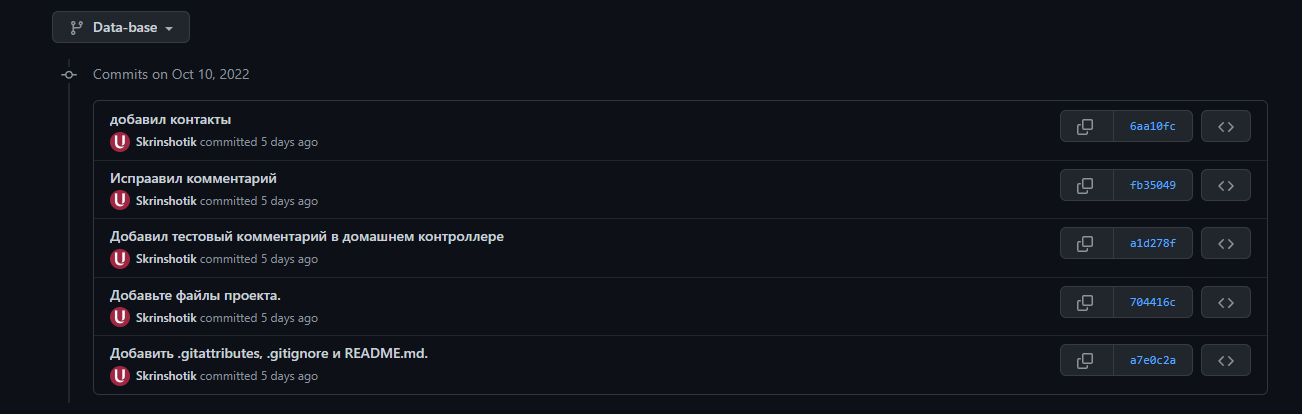


Рис.7

1. Продемонстрировать откат изменений в ветке 1.
2. Создать удаленный репозиторий (на github.com или в сети).



Рис.8

1. Отправить данные на удаленный репозиторий (выполняется одним из студентов подгруппы).Добавить к удаленному репозиторию участников проекта.



Рис.9

1. Получить данные из удаленного репозитория (выполняется прочими студентами).

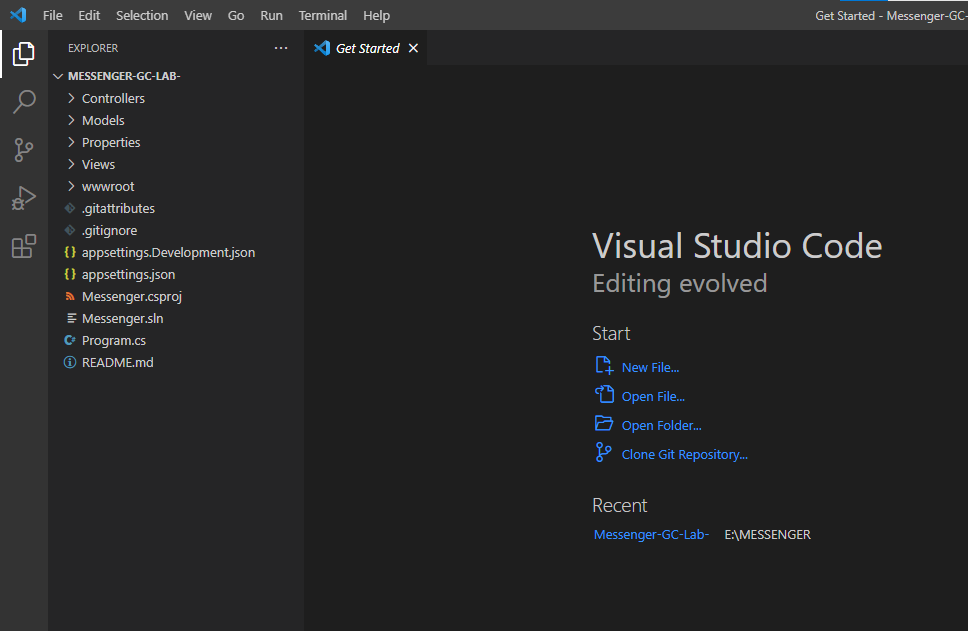


Рис. 10

1. Изменить полученные данные.

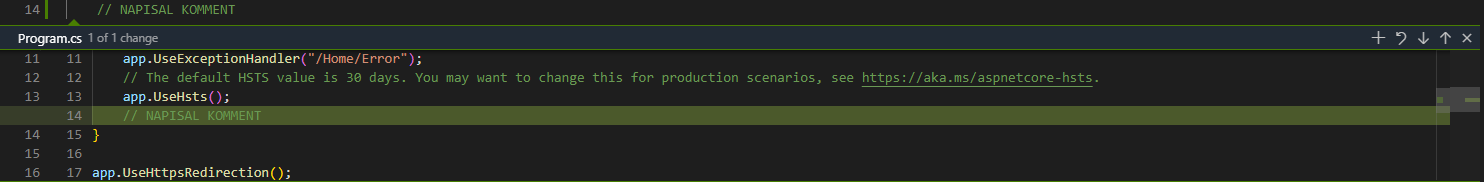


Рис.11

1. Зафиксировать изменения и отправить их на удаленный репозиторий(выполняется всеми студентами подгруппы).

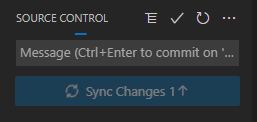


Рис.12



Рис.13

1. Получить данные из удаленного репозитория.



Рис.14

1. Просмотреть историю изменений.



Рис.15

**Контрольные вопросы:**

1. Что такое система управления версиями?

Система управления версиями — программное обеспечение для облегчения работы с изменяющейся информацией.

1. Как создать репозиторий?

### Создание репозитория в существующем каталоге

Если у вас уже есть проект в каталоге, который не находится под версионным контролем Git, то для начала нужно перейти в него. Это выглядит по-разному:

для Windows:



а затем выполнить команду:



Эта команда создаёт в выбранном каталоге новый подкаталог с именем .git, необходимые все необходимые файлы репозитория — репозитория Git

1. Как создать ветку?

**Создать** новую **ветку** можно с помощью команды **git** branch <название **ветки**>. Важно, чтобы название было понятным для вас и других участников проекта

1. Как провести слияние? Как разрешить конфликт и что это такое?

Выполнение слияния:

Для этого выполните команду **git merge <название ветки> , где <название ветки> — название ветки, которая будет объединена с принимающей**.

Самый простой способ разрешить конфликт — отредактировать конфликтующий файл. Для завершения слияния создайте новый коммит. Git обнаружит, что конфликт разрешен, и создаст новый коммит слияния для завершения процедуры слияния.

1. Как зафиксировать изменения?

Для того, чтобы **зафиксировать изменения** - необходимо выбрать то, что нужно сохранить и выполнить коммит. Для выбора нужных **изменений** из совершенных используется команда **git** add.

1. Как провести откат? Различия в reset и revert, мягкий и жесткий reset.

Отмена последних коммитов в **Git**  
Команду **git** reset можно использовать для отмены внесенных изменений: **git** reset — soft HEAD~x (or **git** reset — soft commit hash). Вместо ~x введите число. Например, если вы укажете ~4 , то команда повлияет на четвертый снизу коммит.

git **revert** действует только на уровне коммита и не работает на уровне файлов. Операция сброса (команда **reset**) принимает в качестве аргумента коммит и сбрасывает «три дерева» до состояния репозитория при указанном коммите. Ее можно выполнить в трех разных режимах, соответствующих трем деревьям.

1. Какова последовательность действий при работе с локальным репозиторием?

**Руководство по системе управления версиями - Git**

Для начала, скачайте msysgit (инсталляционный пакет <http://git> scm.com/download/win) и запустить его. Все настройки в инсталляторе оставляем по умолчанию.

1. Какова последовательность действий при работе с удаленным репозиторием?

Чтобы загрузить что-нибудь в удаленный репозиторий, сначала нужно к нему подключиться. Регистрация и установка может занять время, но все подобные сервисы предоставляют хорошую документацию.  
Чтобы связать наш локальный репозиторий с репозиторием на GitHub, выполним следующую команду в терминале. Обратите внимание, что нужно обязательно изменить URI репозитория на свой.

Проект может иметь несколько удаленных репозиториев одновременно. Чтобы их различать, мы дадим им разные имена. Обычно главный репозиторий называется origin.

Сейчас самое время переслать наш локальный коммит на сервер. Этот процесс происходит каждый раз, когда мы хотим обновить данные в удаленном репозитории.

Если вы сделали изменения в вашем удаленном репозитории, другие пользователи могут скачать изменения при помощи команды pull.

9. Каковы возможности при работе с удаленным репозиторием? Как его клонировать,

получать и отправлять данные?

 Удалённые репозитории — это модификации проекта, которые хранятся в интернете или ещё где-то в сети. Их может быть несколько, каждый из которых, как правило, доступен для вас либо только на чтение, либо на чтение и запись. Совместная работа включает в себя управление удалёнными репозиториями и помещение (push) и получение (pull) данных в и из них тогда, когда нужно обменяться результатами работы. Управление удалёнными репозиториями включает умение добавлять удалённые репозитории, удалять те из них, которые больше не действуют, умение управлять различными удалёнными ветками и определять их как отслеживаемые (tracked) или нет и прочее.

**git clone**

На самом деле git clone работает как обёртка над некоторыми другими командами. Она создаёт новый каталог, переходит внутрь и выполняет git init для создания пустого репозитория, затем она добавляет новый удалённый репозиторий (git remote add) для указанного URL (по умолчанию он получит имя origin), выполняет git fetch для этого репозитория и, наконец, извлекает последний коммит в ваш рабочий каталог, используя git checkout.

Вывод: На сегодняшней лабораторной работе мы с командой ознакомились со всеми инструкциями и литературой, которой преподаватель нам дал сделали данную лабораторную работу.