**Отчёт по лабораторной работе № 6**

**По дисциплине Теория разработки программного обеспечения**

**Студента группы 19-КБ-ПР2**

**Соловьева Даниила**

**Цель работы**: создать в системе контроля версий репозиторий для нового проекта и выполнить все основные действия с исходным кодом программы, связанные с контролем версий.

**Введение**: любая существующая система контроля версий направлена на то, чтобы систематизировать любые вносимые в проект изменения, сохранить целостность системы при внесении изменений, предотвратить нежелательные и непредсказуемые эффекты, а также сделать процесс внесения изменения более формальным.

К минусам таких систем относится кажущаяся сложность внесения изменений в конфигурационные файлы системы контроля версий, но данная лабораторная работа направлена на знакомство с такими системы и на обретение навыков работы с конфигурационными файлами.

**Используемое программное обеспечение**: Microsoft Word 2019, Git version 2.29.2

**Лабораторная работа выполнялась на следующем компьютере**:

* CPU - AMD Ryzen 5 2600
* GPU - Gigabyte GTX 1660 Super
* RAM - DDR4 16gb
* OC - Windows 10

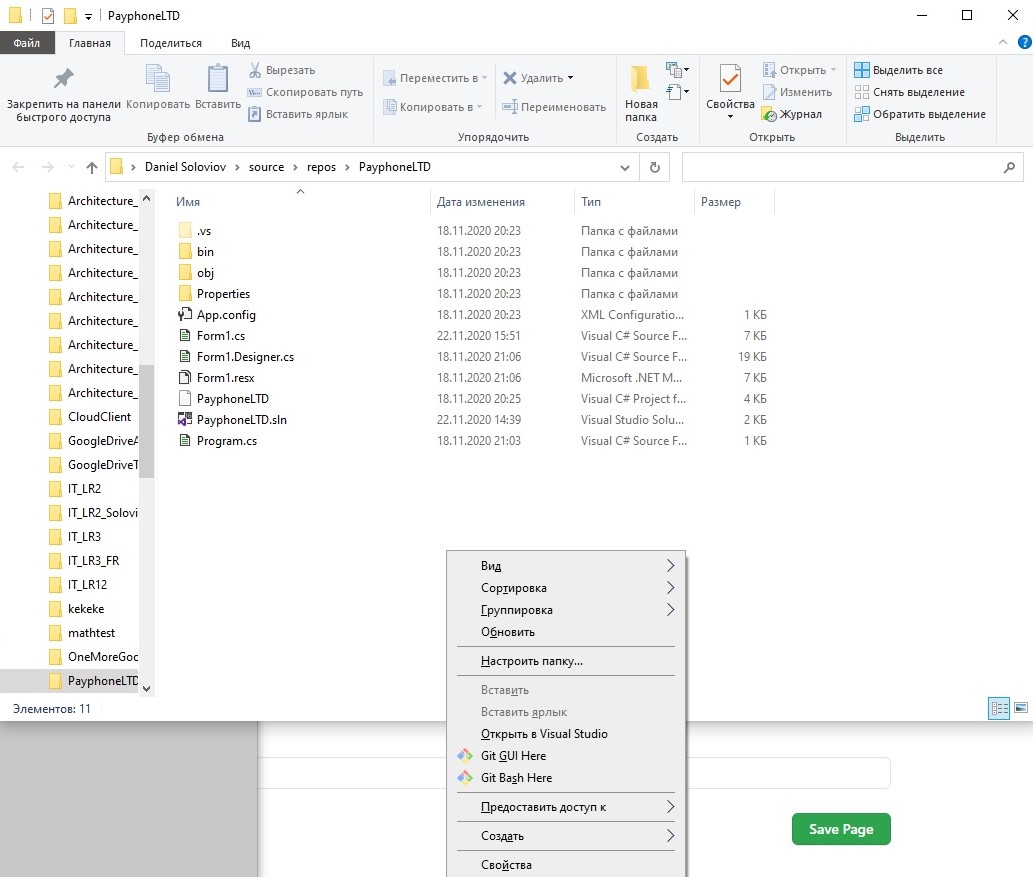
**Ход выполнения работы**:

**1. Экспортируем созданный проект в репозиторий системы контроля версий.**

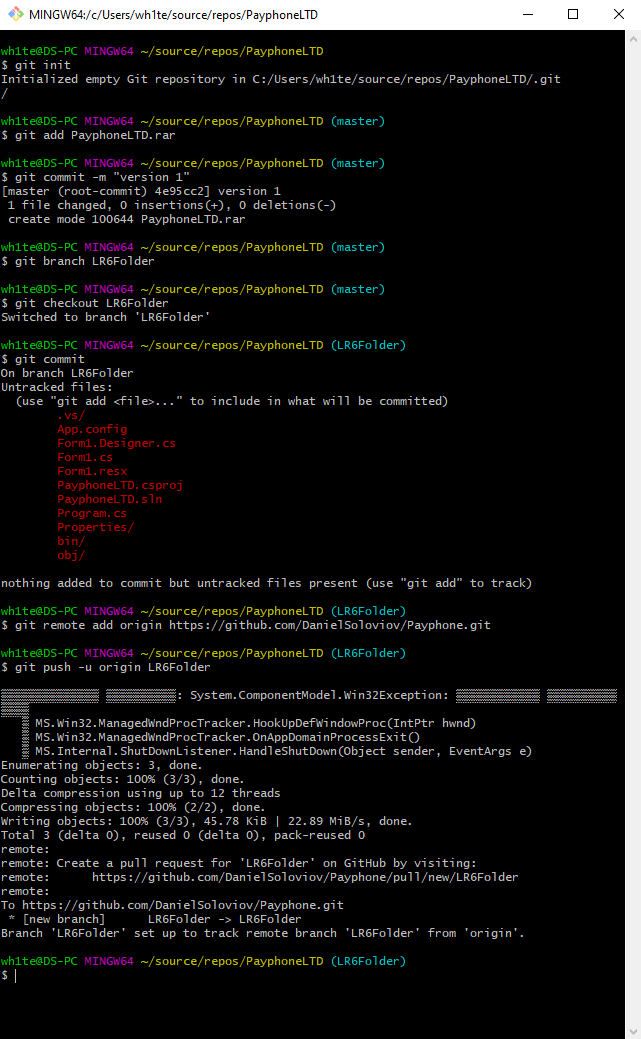
В папке с нашим проектом вызовем командную строку для настройки соединений локального проекта с онлайн-репозиторием. Этот процесс выполняется посредством нажатия правой кнопки мыши в любой части папки с проектом и нажатием кнопки Git Bash here, после чего запустится терминальная среда.

Произведём в ней следующие шаги:

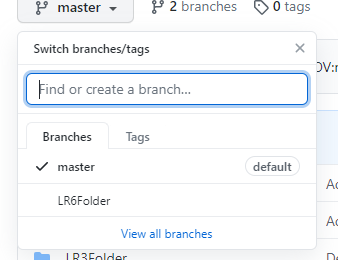
1. Инициализируем репозиторий командой $ *git init*
2. Добавим в инициализированный репозиторий наш проект командой $ *git add имяФайла.расширение*
3. Затем будет необходимо использовать команду $ *git commit -m "любой текст"*, которая берёт все данные, добавленные в индекс с помощью $ *git add*, и сохраняет их слепок во внутренней базе данных, а затем сдвигает указатель текущей ветки на этот слепок.
4. После необходимо создать интересующую нас ветку, а именно LR6Folder используя команду $ *git branch LR6Folder*, после чего совершаем переход в эту ветку командой $*git checkout LR6Folder*.
5. После всех этих произведённых операций необходимо подключить онлайн ветку к только что созданной локальной с помощью команды $ \_git remote add origin "ссылка на необходимый репозиторий", и соответственно выгрузить данные в этот репозиторий командой $ *git push -u origin "имя локального репозитория"*.
6. При выполнении всех описанных действий корректно, в онлайн репозитории будет успешно создана новая загруженная ветка

**Производим инициализацию терминальной среды** 

**Команды для ввода в терминал**



**Созданная ветка**

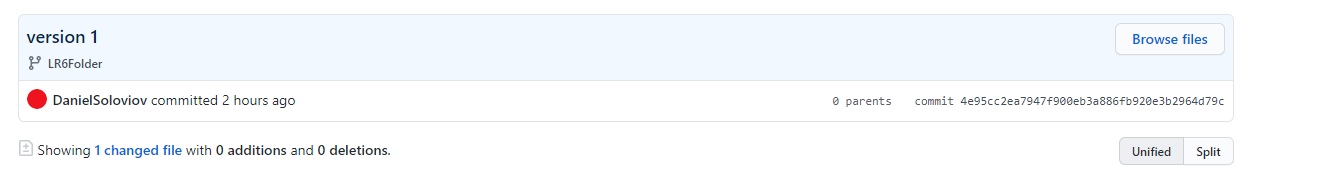


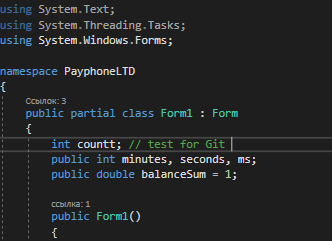
**2. Теперь в загруженный проект необходимо внести изменения.**

Для этого произведём следующие шаги:

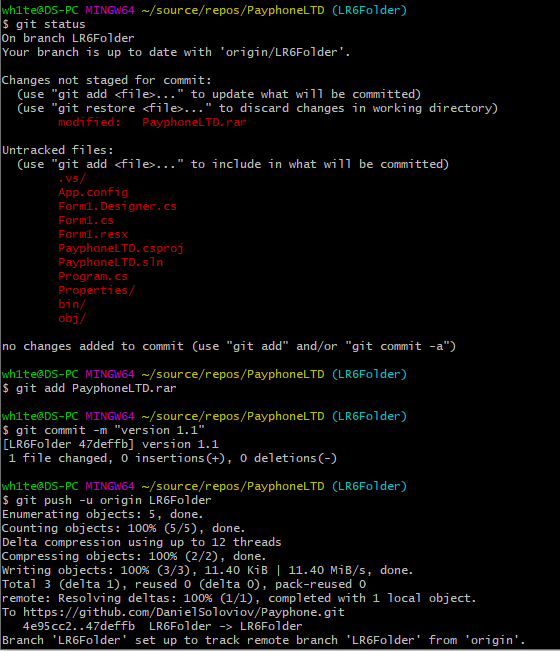
1. Скачаем загруженный в онлайн репозиторий архив с проектом.
2. Внесём любые изменения в проект. (это может быть внесение комментария, изменение имени переменной, добавление unit test в проект и т.п.)
3. После этого, в терминальной среде Git при вводе команды $ *git status* мы увидим, что система заметила изменения, и нам будет предложено вернуться на старую версию либо обновить проект во **всех** репозиториях.
4. Обновим версию проекта. Для этого необходимо ввести команду $ *git add имяФайла.расширение*, а после $ *git commit -m "любойТекст"*
5. Заметим, что файл был изменён.

**Скачиваем исходно загруженную версию из созданной ветки и вносим изменения в код программы**



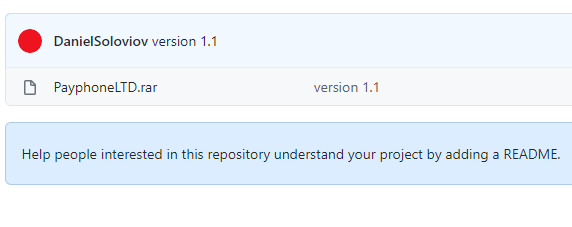


**Архивируем измененный проект и используем терминальную среду для обновления версии проекта**



Как можно заметить, система контроля версий распознала изменение в проекте и пометила его красным цветом

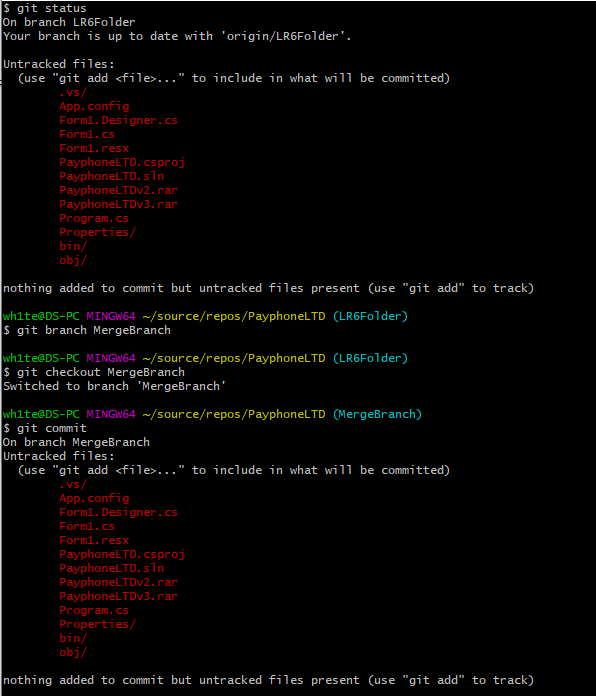
**После загрузки обновленной версии проекта проверяем его наличие в созданной ветке**

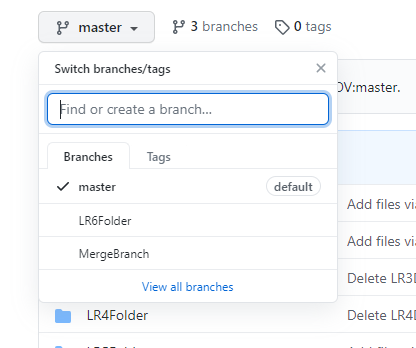


**3. Создадим вторичную ветку для слияния версий внутри одной ветки.**

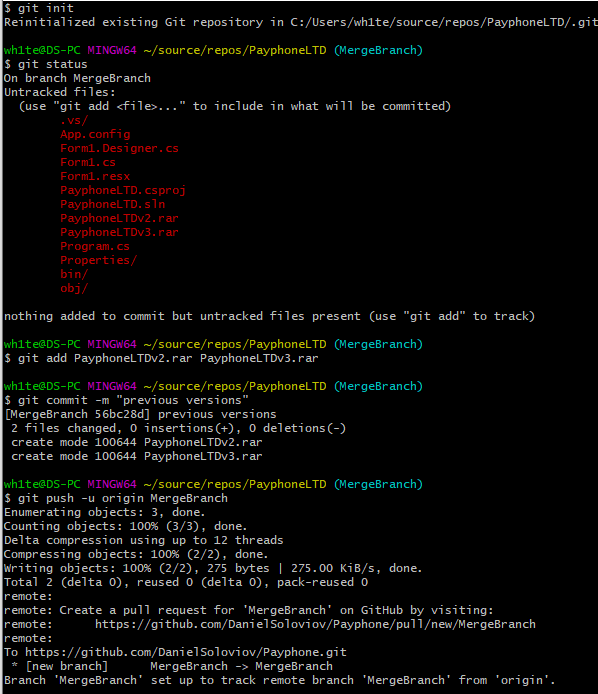
1. Необходимо создать новую ветку (пусть будет MergeBranch), и добавить в нее изменённые проекты командой $ *git add имяФайла.расширение*.
2. Проверяем факт появления новой ветки в репозитории.
3. После успешного создания новой ветки будет необходимо перейти в прошлую ветку и начать процесс слияния с новообразовавшейся командой $ \_git merge "имяВеткиДляСлияния"
4. Оцениваем успешность произведённой операции.

**Создаём новую ветку для последующего слияния и проверяем ее наличие в репозитории**

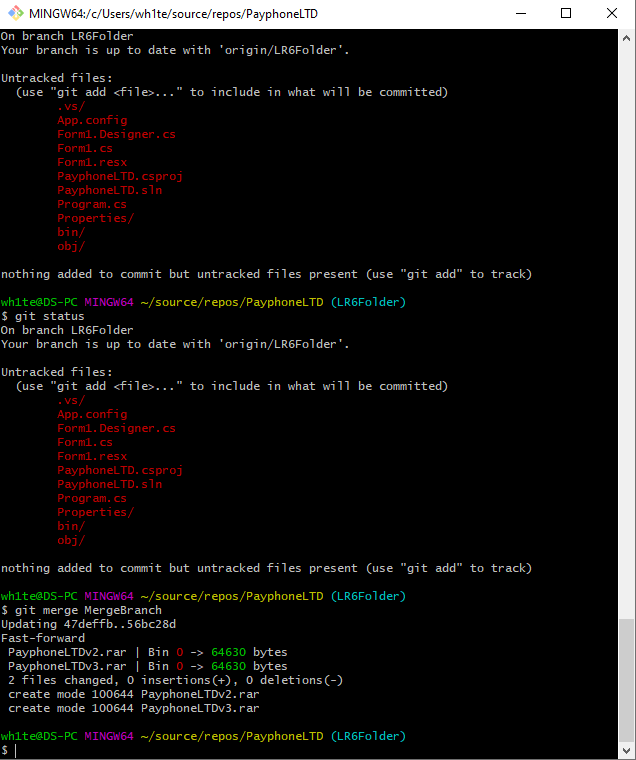




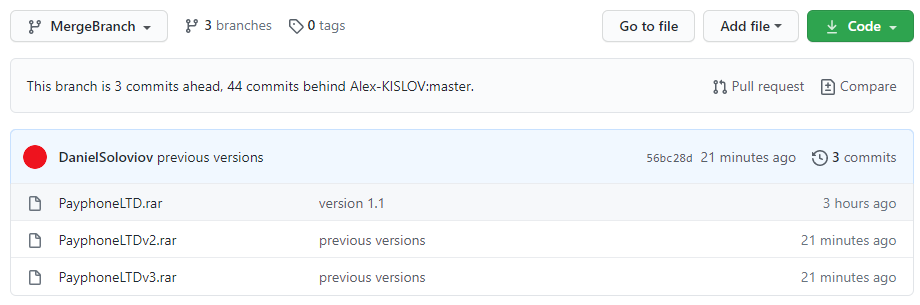
**Добавляем в созданную ветку изменённые проекты**

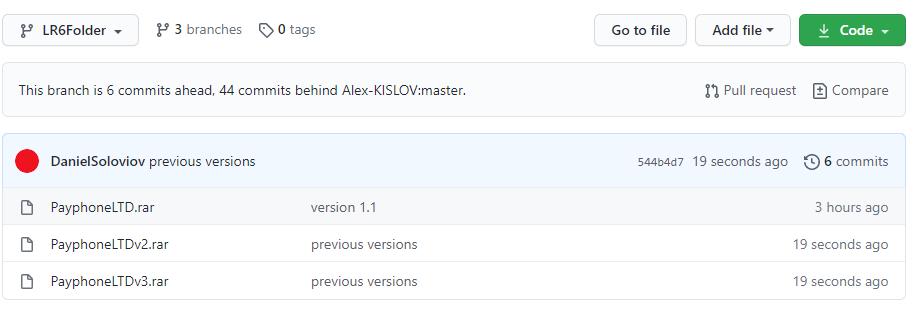


**Переходим в предыдущую ветку и начинаем процедуру слияния веток**



**Проверяем успешность процедуры слияния веток**





Как можно заметить, ветка под названием LR6Folder была успешно объединена с веткой MergeBranch

**Заключение**: система контроля версий программного обеспечения - это незаменимая часть работы любого программного инженера и навыки работы с ней (в данном случае с системой контроля версий Git version 2.29.2), приобретённые в процессе выполнения данной лабораторной работы, окажут незаменимую помощь в дальнейшей профессиональной деятельности.

**Список используемой литературы**:

1. Попова О.Б. Теория разработки программного обеспечения. Методические указания к выполнению Л/Р
2. Попова О.Б. Теория разработки программного обеспечения. Конспекты лекций.
3. Зараменских, Е. П. Управление жизненным циклом информационных систем : учебник и практикум для академического бакалавриата / Е. П. Зараменских. — Москва : Издательство Юрайт, 2019. — 431 с. — (Бакалавр. Академический курс). — ISBN 978-5-9916-9200-7.
4. Интернет-ресурс: [https://git-scm.com/book/ru/v2/Appendix-C%3A-Команды-Git-Основные-команды](https://git-scm.com/book/ru/v2/Appendix-C%3A-%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B-Git-%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5-%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B)