МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Донецкий национальный технический университет»

Факультет Интеллектуальных систем и программирования

Кафедра "Программная инженерия" им. Л.П. Фельдмана

Лабораторная работа № 2

на тему: «Работа с ветками»

по курсу: «Профессиональная практика программной инженерии»

Проверил:

Филипишин Д.А.

Незамова Л.В.

Выполнил:

ст. гр. ПИ-20б

Понамарёв Н.Т.

Донецк-2024

Задание:

Для каждого из описанных модулей добавить несколько файлов (по желанию), имитирующие разработку программного кода на любом выбранном языке программирования используя не менее 3х веток. Описать имеющиеся ветки,

Описать, какие файлы реализовывают какой конкретно модуль и в какой ветке находятся.

Влить все ветки в master.

Изменить данные в одной из веток так, чтобы вызвать конфликтную ситуацию (продемонстрировать ситуацию), решить конфликтную ситуацию (также продемонстрировать).

Добавить изменения в master.

Удалить все ветки кроме master.

Для работы с консолью GitBash клонируем репозиторий через Github Desktop.

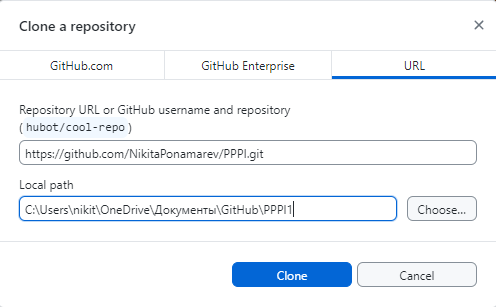


Рисунок 1 – Клонирование репозитория

1. Разделим модули по трём веткам и создадим ветки с помощью команды git branch имя\_ветки

Ветка 1: "Серверная часть" – server\_branch

1. Модуль работы с базой данных – DataBase.cpp
2. Модуль обмена данными с клиентом игры – DataExchange.cpp
3. Модуль безопасности– Security.cpp

Ветка 2: "Игровая логика" – game\_logic\_branch

1. Модуль отрисовки – render.cpp
2. Модуль поля игрока– field.cpp
3. Модуль управления – controls.cpp

Ветка 3: "Интрефейс" – interface\_branch

1. Модуль интерфейса клиента игры – interface.cpp

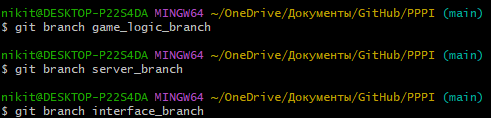
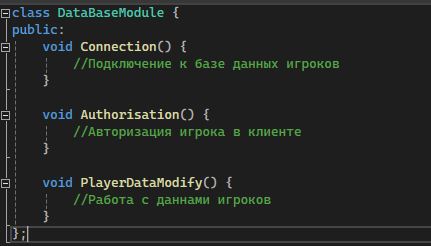


Рисунок 2 – Создание веток

2) Напишем псевдокод для каждого из модулей.

  
Рисунок 3 – Псевдокод файла DataBase.cpp

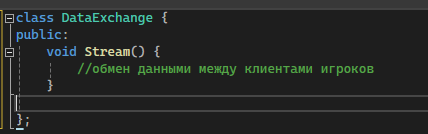


Рисунок 4 – Псевдокод файла dataexchange.cpp

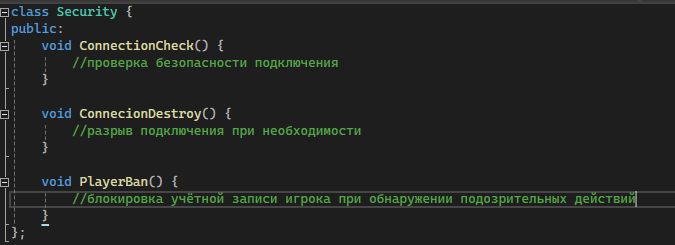
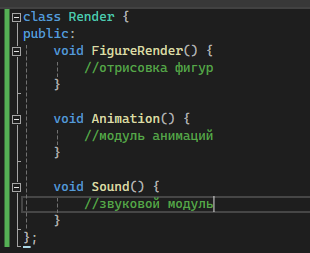


Рисунок 5 – Псевдокод файла Security.cpp

  
Рисунок 6 – Псевдокод файла render.cpp

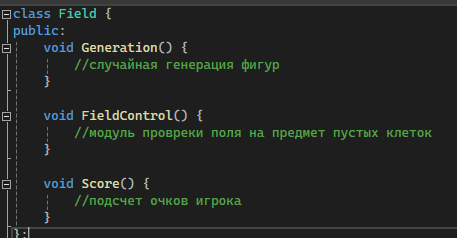


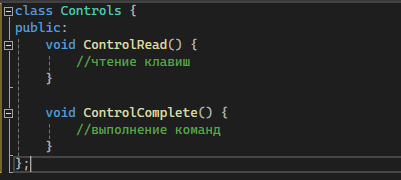
Рисунок 7 – Псевдокод файла field.cpp 

Рисунок 8 – Псевдокод файла controls.cpp

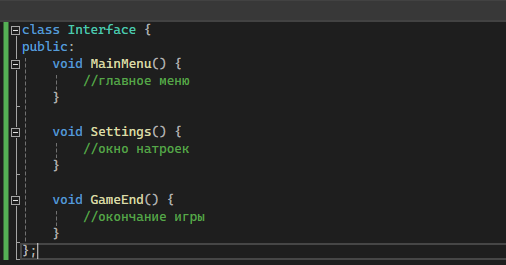


Рисунок 9 – Псевдокод файла interface.cpp

3) Загружаем модули по разным веткам

Переключаемся на ветку server\_branch и добавляем туда файлы

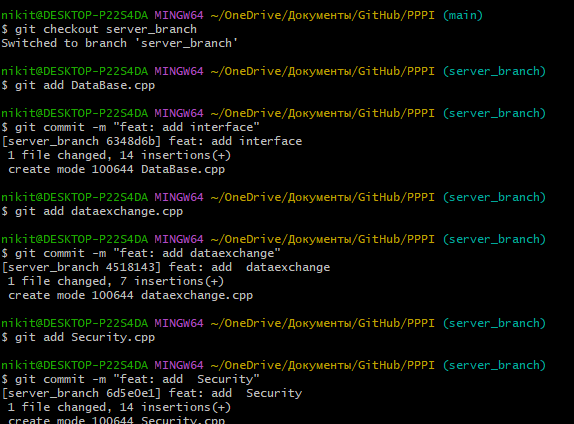


Рисунок 11 – Загрузка модулей в server\_branch

Переключаемся на ветку game\_logic\_branch и добавляем туда файлы

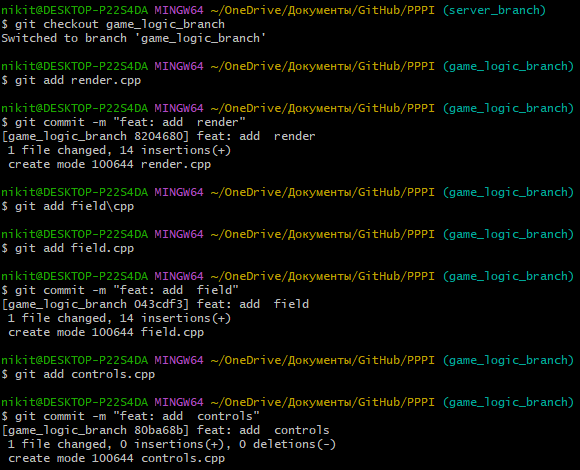


Рисунок 12 – Загрузка модулей в game\_logic\_branch

Переключаемся на ветку interface\_branch и добавляем туда файлы

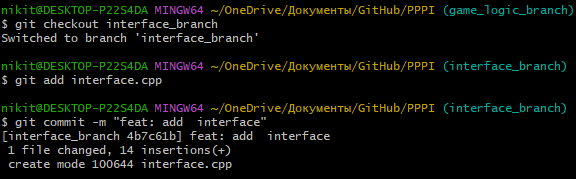


Рисунок 13 – Загрузка модулей в interface\_branch

Проверяем лог ревизий для каждой ветки

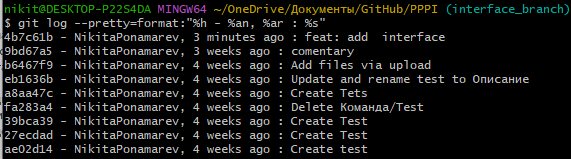


Рисунок 14 – Лог ревизий для interface\_branch

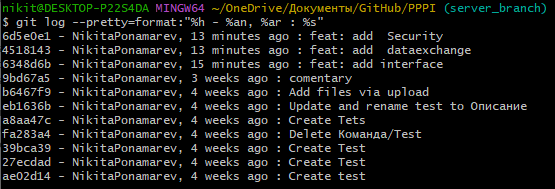


Рисунок 15 – Лог ревизий для server\_branch

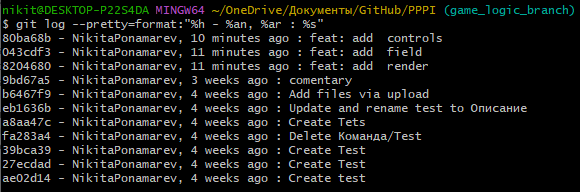


Рисунок 16 – Лог ревизий для game\_logic\_branch

4) Чтобы получить конфликт слияния, добавим в ветке game\_logic\_branch файл с таким же названием как в ветке server\_branch, но другим содержанием.

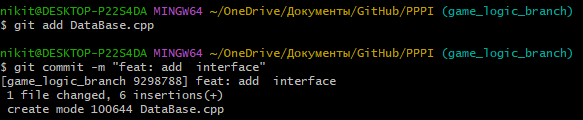


Рисунок 17 – Добавление файла для создания конфликта слияний

5) Переключаемся на ветку мастера и выполняем слияние веток.

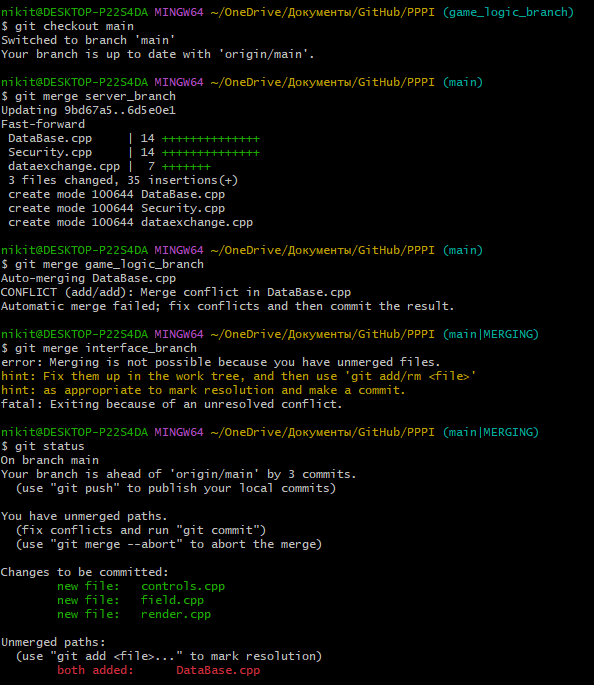


Рисунок 18 – Попытка слияния веток

Получаем конфликт слияния



Рисунок 19 – Конфликт слияния

Чтобы разрешить конфликт достаточно отредактировать проблемный файл и закоммитить его. После чего снова попробовать слить ветки.

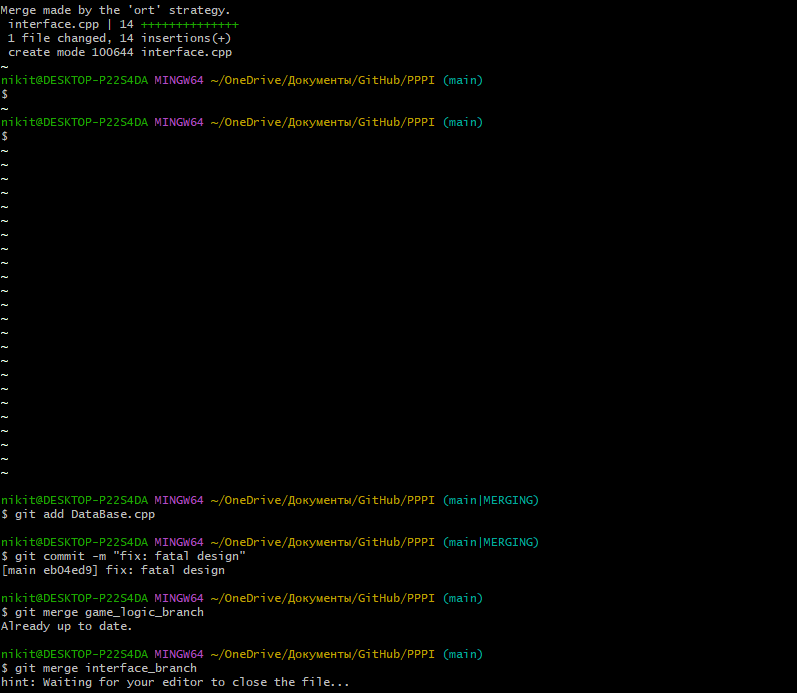


Рисунок 20 – Решение конфликта слияния

Лог ревизии

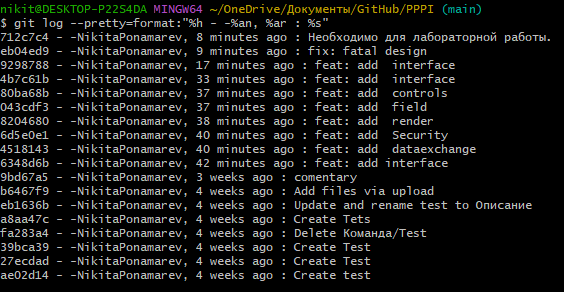


Рисунок 21 – Лог ревизии

6) Удаляем все ветки

Удаление веток после проверки ЛР.

В результате работы добавляются следующие файлы в репозиторий

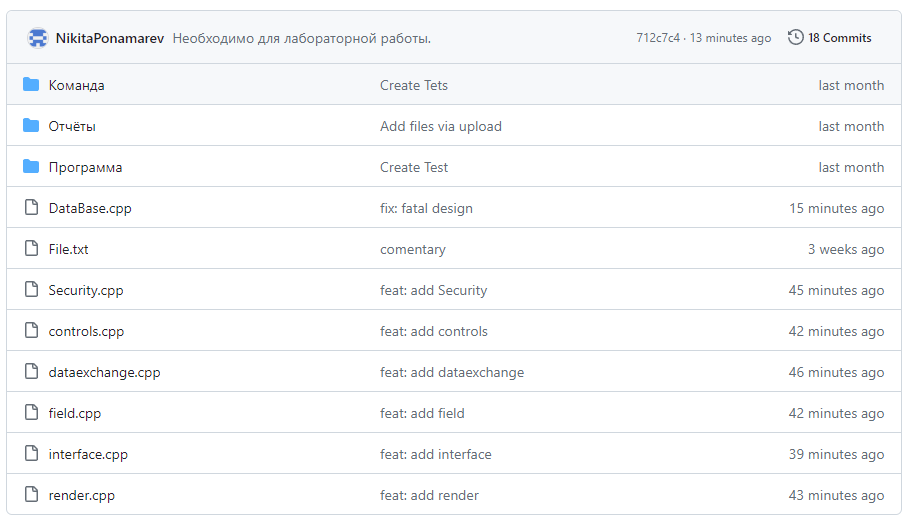


Рисунок 23 – Список добавленных файлов в репозитории

7) Загружаем отчёты в репозиторий

Рисунок 24 – Загрузка отчётов в репозиторий