Министерство науки и высшего образования Российской̆ Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

ОТЧЕТ

По лабораторной работе №4

«Юнит-тестирование»

по дисциплине «Новые технологии в программировании»

Выполнил:

студент гр. 587-2

\_\_\_\_\_\_\_\_ Баринов Д.А.

"\_\_" \_\_\_\_\_\_\_\_\_ 2020 г.

Проверил:

Доцент кафедры КСУП:

\_\_\_\_\_\_\_\_ Горяинов А.Е.

"\_\_" \_\_\_\_\_\_\_\_\_ 2020 г.

# 1 Введение

## 1.1 Цель работы

В данной работе необходимо изучить организацию тестирования в разработке ПО, и получить умения написания юнит-тестов.

## 1.2 Задачи

1. Изучить организацию процесса тестирования в разработке ПО, виды тестирования, сроки их проведения и ответственных исполнителей.

2. Изучить основные атрибуты и классы библиотеки NUnit для написания юнит-тестов.

3. Научиться рассчитывать цикломатическую сложность методов, классов и проектов, оценивать степень покрытия кода тестами.

4. Написать юнит-тесты для классов логики приложения с использованием библиотеки NUnit.

# 2 Цикломатическая сложность проекта

Цикломатической сложностью называют количество линейно независимых алгоритмических маршрутов через программный код. Расчет цикломатической сложности позволяет определить все требуемые тесты (тестовые случаи) для того или иного класса. Она зависит от ветвлений алгоритма, содержащихся в нем циклов и операторов перехода. Также на цикломатическую сложность влияют вызовы других методов.

Для приложения NotesApp необходимо выполнить тестирование двух классов логики. Их цикломатическая сложность определяется как сумма сложностей каждого метода соответствующего класса. Далее по сложностям классов рассчитывается цикломатическая сложность всего проекта, которая может быть представлена в виде дерева следующим образом:

Проект NoteApp (цикл. сложность проекта = 28)

1. Класс Note (цикл. сложность класса = 22)

Конструктор Note() (цикл. сложность = 4)

Метод Clone () (цикл. сложность = 6)

Свойства Title (цикл. сложность = 4)

Свойства Category (цикл. сложность = 2)

Свойства NoteText (цикл. сложность = 2)

Свойство Created (цикл. сложность = 2)

Свойства Modified (цикл. сложность = 2)

2. Класс ProjectManager (цикл. сложность класса = 6)

Метод SaveToFile(Project) (цикл. сложность = 2)

Метод LoadFromFile() (цикл. сложность = 4)

Общее количество написанных юнит-тестов в проекте = 12. Покрытие кода 100%.

# 3 Выполнение юнит-тестов и история коммитов

В процессе проверки и отладки программы бывает необходимо знать время выполнения автоматических проверок. Пакеты библиотеки NUnit позволяют наблюдать не только статус тестов, но и время, затраченное на их выполнение. На рисунке 3.1 представлена информация о проведении тестов.

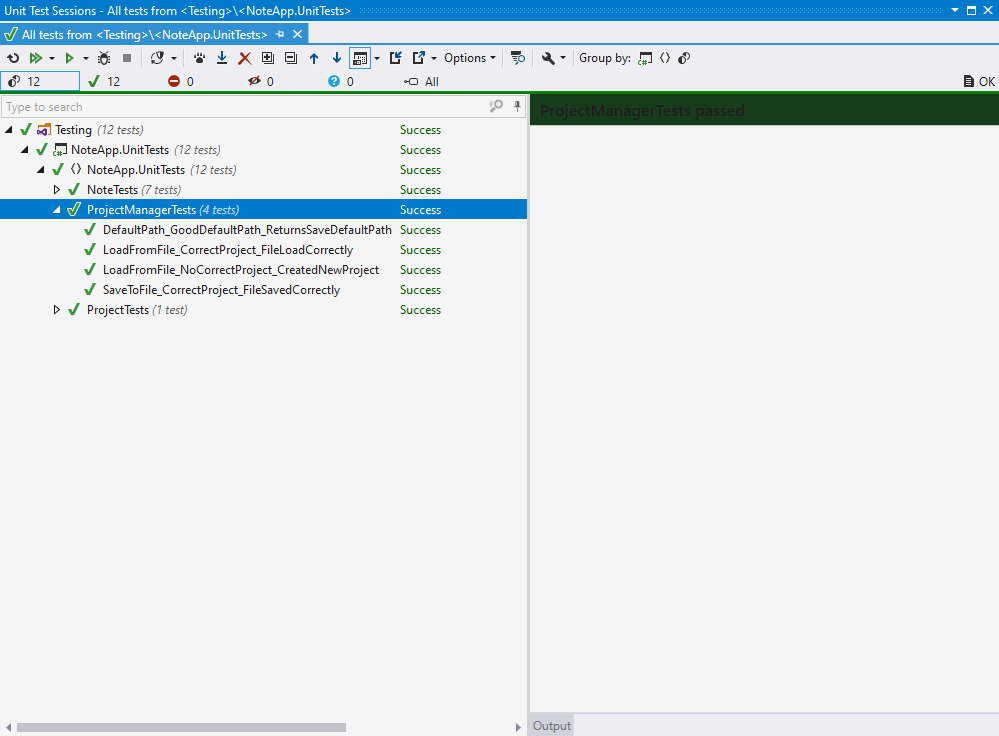


Рисунок 3.1 – Результат выполнения юнит-тестов.

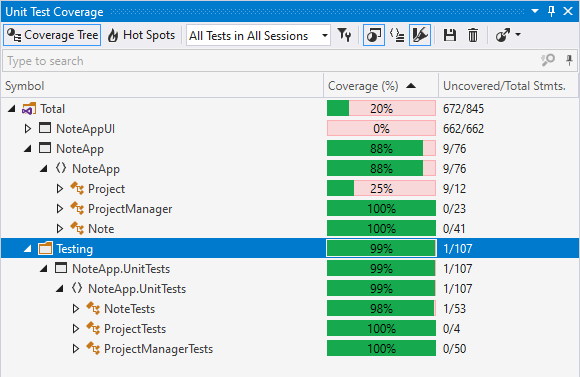


Рисунок 3.2 – Покрытие кода юнит-тестами.

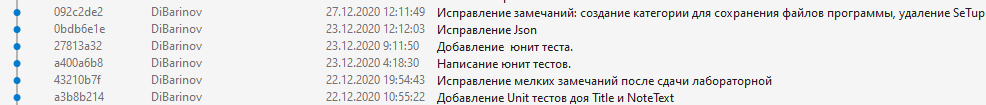


Рисунок 3.3 – История коммитов разработки проекта тестов

# Заключение

Проведенное юнит-тестирование показало корректность работы логики проекта разрабатываемого приложения. Необходимые тесты были организованы с помощью библиотеки тестирования NUnit с использованием инструментов, соответствующих логике работы тестируемых модулей. Результаты обработаны и представлены в требуемой форме.