Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

ЮНИТ-ТЕСТИРОВАНИЕ

Отчет по лабораторной работе №4 по дисциплине «Новые технологии в программировании»

Студент гр. 588-1

_____/ Колесников А.М.
(подпись)
«_19_» _апреля_ 2021г.

Руководитель старший научный сотрудник, доцент каф. КСУП

_____ / Горяинов А.Е.
(подпись)
«___» _____20__г.

Томск 2021

Оглавление

Введение 3

2 Дерево цикломатической сложности методов и классов проекта логики 4

3 Исходный код одного из юнит-тестов 6

4 История фиксаций изменений 8

5 Заключение 9

Введение

Цель работы: изучить организацию тестирования в разработке ПО и получить умения написания юнит-тестов.

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 Дерево цикломатической сложности методов и классов проекта логики

Юнит-тестирование – тестирование программы в виде отдельных, изолированных друг от друга минимальных модулей. Условие изоляции тестируемого модуля от других необходимо для того, чтобы в случае обнаружения ошибки быть уверенным, что ошибка возникла именно в тестируемом модуле. В противном случае найти место возникновении ошибки будет гораздо сложнее.

В процедурном программировании минимальными модулями для юнит-тестирования являются функции или процедуры. В ООП минимальными модулями считаются классы. Именно классы, а не методы, так как поведение методов может зависеть от текущего состояния экземпляра класса. Таким образом, юнит-тесты в ООП это тесты, выполняющие тестирование отдельных классов с максимально возможной изоляцией от поведения других классов.

Юнит-тестированию подлежит поведение класса, находящееся под модификаторами доступа public или protected. Закрытая реализация класса (private) напрямую тестированию не подвергается, только опосредованно через вызов открытых методов класса. Это обусловлено тем, что юнит-тесты фактически имитируют случаи реального использования классов логики, а, следовательно, не должны нарушать инкапсуляцию тестируемого класса.

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 NoteApp необходимо выполнить тестирование трех 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 ContactsApp (цикл. сложность = 30)

- Класс Contact(цикл. сложность = 17)
  - Свойство Surname(цикл. сложность = 3)
  - Свойство Name(цикл. сложность = 3)
  - Свойство Birthday(цикл. сложность = 3)
  - Свойство PhoneNumber(цикл. сложность = 1)
  - Свойство EMail(цикл. сложность = 2)
  - Свойство IDVKontacte(цикл. сложность = 3)
  - Метод Contact(цикл. сложность = 1)
  - Метод Clone(цикл. сложность = 1)
- Класс PhoneNumber(цикл. сложность = 3)
  - Свойство PhoneNumber(цикл. сложность = 3)
- Класс Project(цикл. сложность = 6)
  - Свойство Contacts(цикл. сложность = 1)
  - Метод SearchContactByString(цикл. сложность = 3)
  - Метод SearchContactByBirthday(цикл. сложность = 2)
- Класс ProjectManager(цикл. сложность = 4)
  - Метод SaveToFile(цикл. сложность = 1)
  - Метод LoadFromFile(цикл. сложность = 3)

Общее количество написанных юнит-тестов в проекте = 30. Покрытие кода 100%.

3 Результат выполнения юнит-тестов

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.

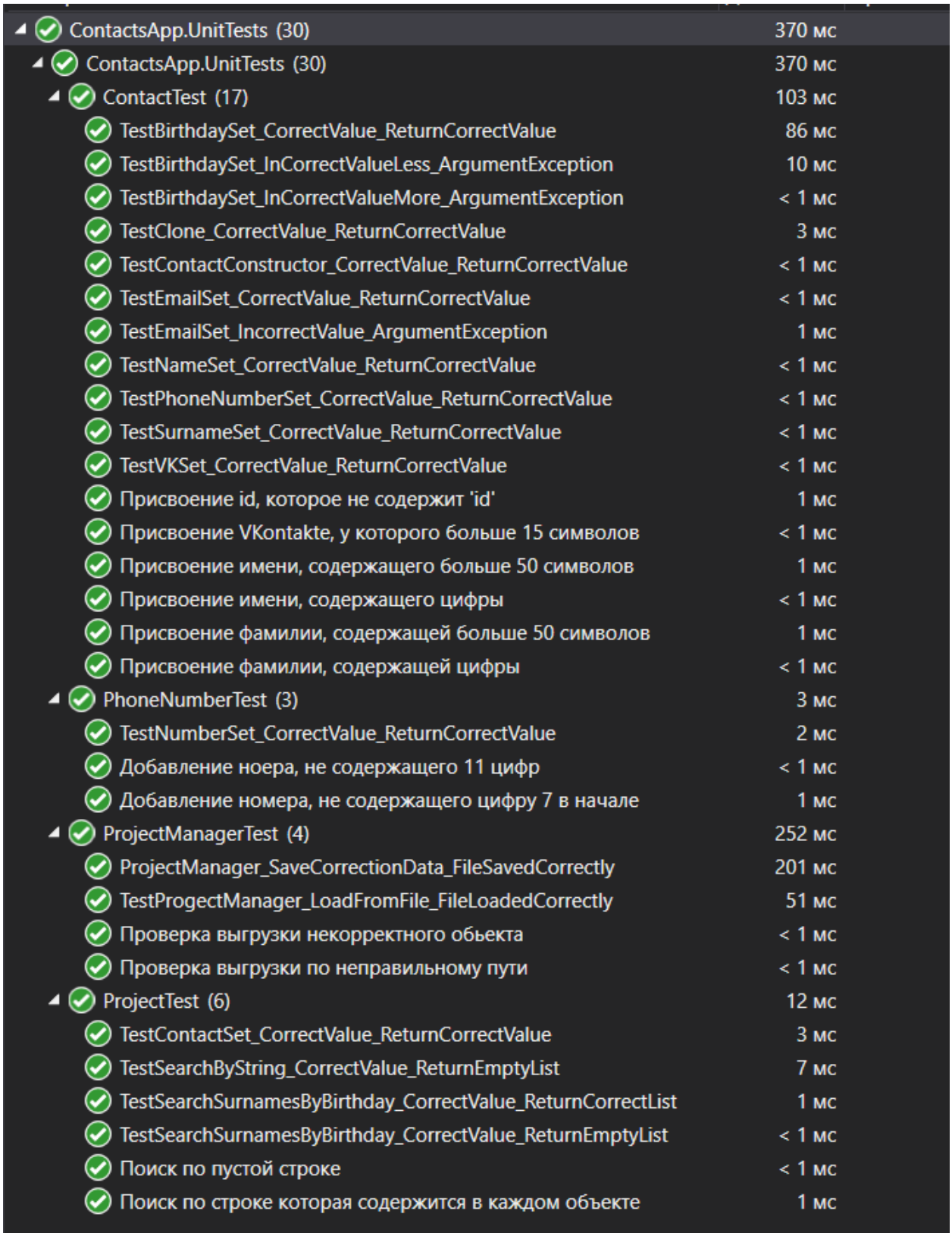

Рисунок 3.1 – Результат выполнения юнит-тестов

4 История фиксаций изменений

По итогу выполнения лабораторной работы история изменений в репозитории на сервисе GitHub стала выглядеть следующим образом (рисунок 4.1).

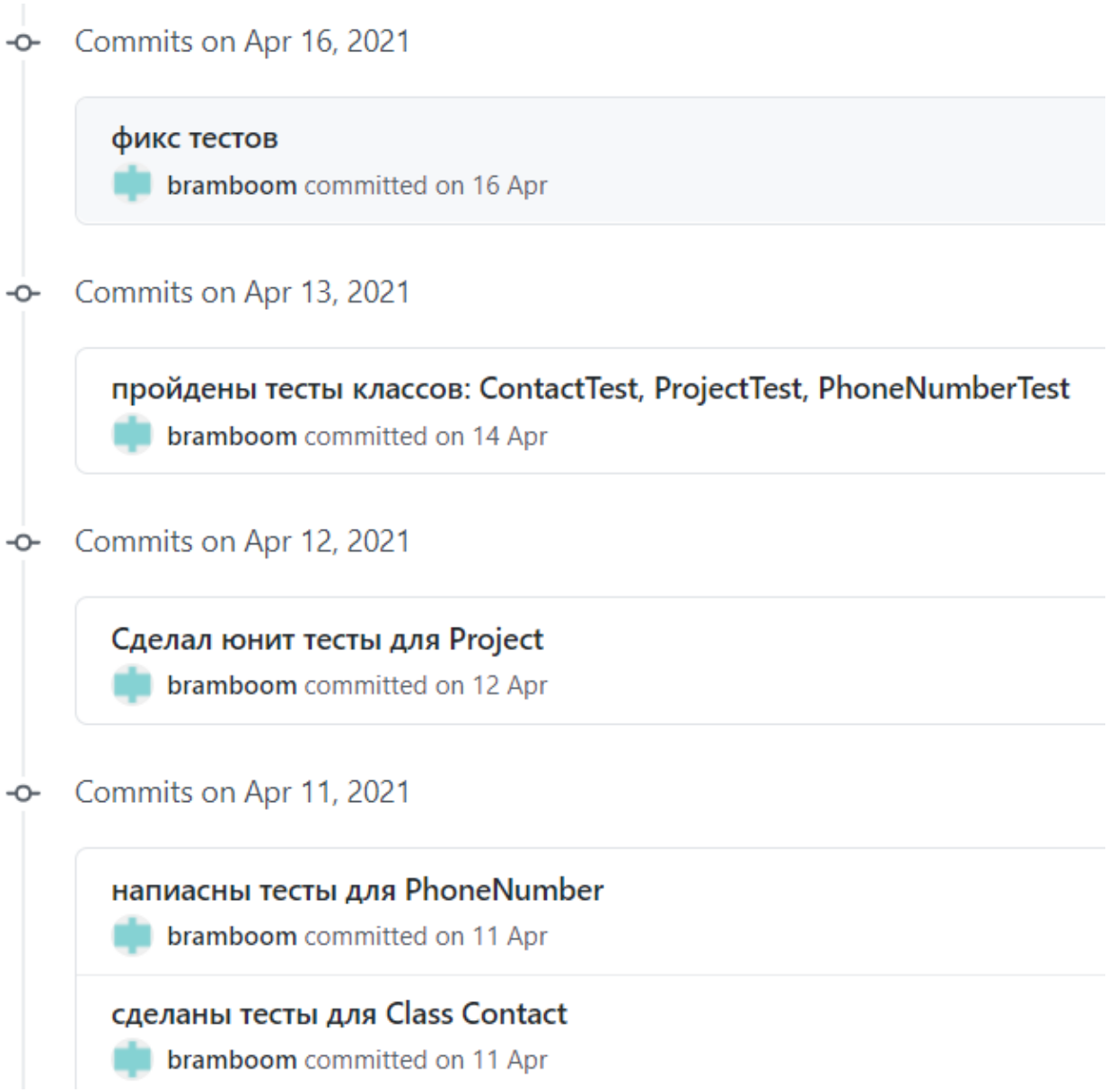

Рисунок 4.1 – История изменений в репозитории

5 Заключение

В ходе лабораторной работы изучена организация тестирования в разработке ПО и получены умения написания юнит-тестов.