Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Алтайский государственный технический университет

им. И.И. Ползунова»

Факультет ИТ

Кафедра ИВТ и ИБ

наименование кафедры

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_ Афонин В. С. (подпись) (и.о., фамилия)

“\_\_9\_”\_\_апреля\_\_\_\_\_\_\_2022 г.

дата

Отчет

по лабораторной работе № 1

Написание спецификации для существующего ПО

название лабораторной работы

по дисциплине Тестирование программного обеспечения автоматизированных систем

наименование дисциплины

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ЛР 090301.23.000 О\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

обозначение документа

Студент группы ИВТ-92 Д. А. Юдин

и.о., фамилия

Преподаватель Старший преподаватель Я. Б. Ерёмин

и.о., фамилия

Барнаул 2022

Порядок выполнения лабораторной работы #2  
  
1. Придумать баг, который мог бы появиться в программе, на которую вы писали спецификацию. Желательно, чтобы было несколько шагов для его воспроизведения, то есть, повторялся он не всегда, а только при определенных условиях.    
  
2. Создать issue на GitHub (в репозитории вашего потока), с заголовком, начинающимся с вашей фамилии, например "[Торвальдс] Не получается удалить сотрудника, фамилия которого содержит тире, если его запись была отредактирована"  
  
3. Назначить вашего напарника ответственным за баг. Если делаете не в паре, назначить меня.  
  
4. Заполнить поля в правой колонке и текст багрепорта в соответствии с рекомендациями: https://github.com/yakov-eremin/PI8X/issues/2  
  
5. В полноценных багтрекерах можно прикреплять медиафайлы (тут сделайте ссылками на облачные сервисы типа Яндекс.Диск), выставлять приоритеты отдельным полем и т.д. В данном случае придется обойтись текстом (и полями, которые есть в правой колонке).     
  
6. Скриншот вашего багрепорта поместить в документ с отчетом.    
  
7. Выбрать фреймворк для unit-тестов, настроить его в вашей IDE, прочитать документацию.  
  
8. Написать 10 любых unit-тестов либо для вашего собственного проекта, либо для проекта вашего напарника. Если вы писали спецификацию на сторонний OpenSource проект, можете загрузить в репозиторий какой-нибудь из своих проектов (если не загружали), и использовать его.  
  
9. Залить в репозиторий папку с тестами  
  
10. Сделать отчет о проделанной работе (с кодом тестов и тестируемых методов), залить на ИЛИАС.

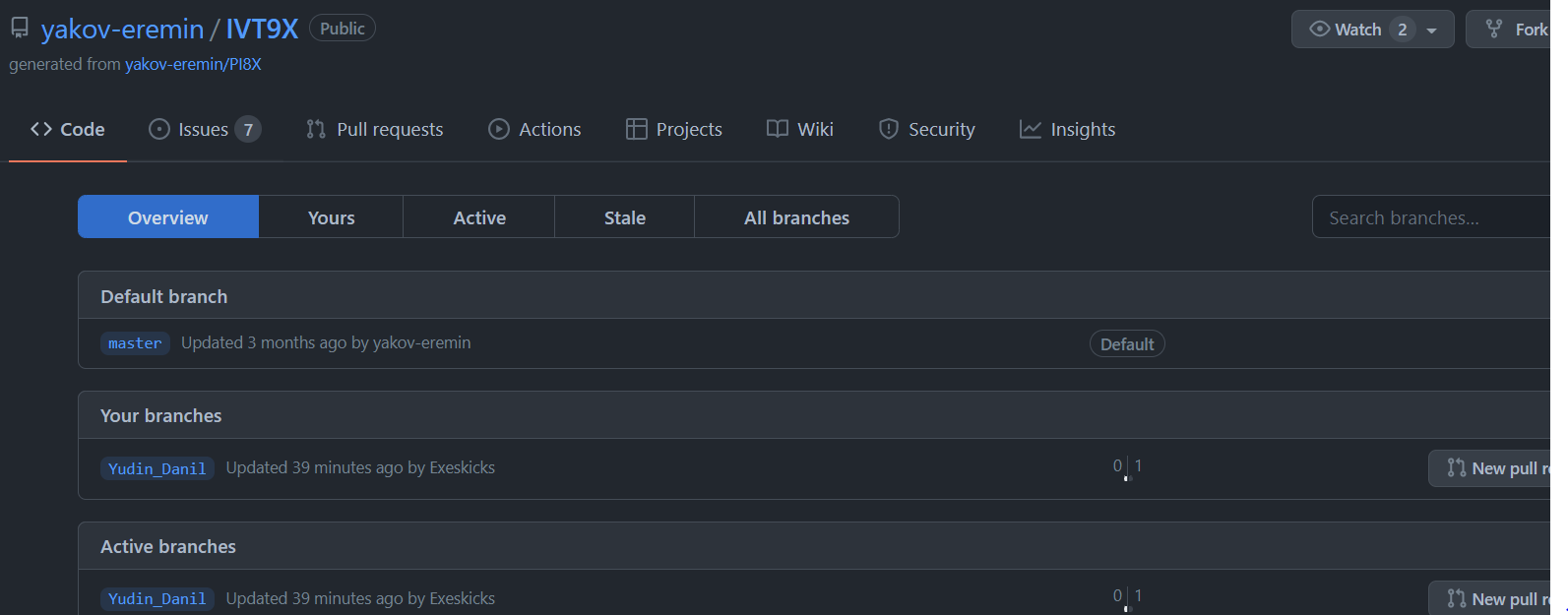


Рисунок 1 – Моя ветка

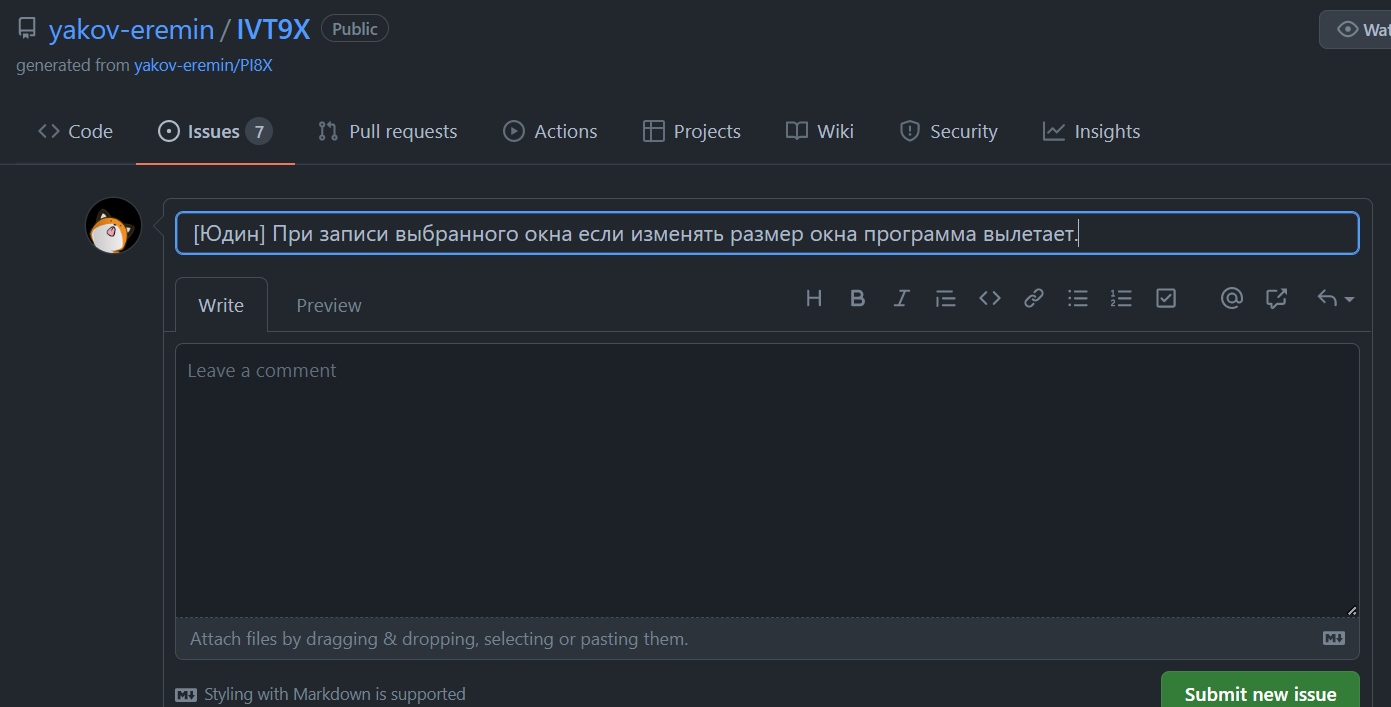


Рисунок 2 – Создание issues

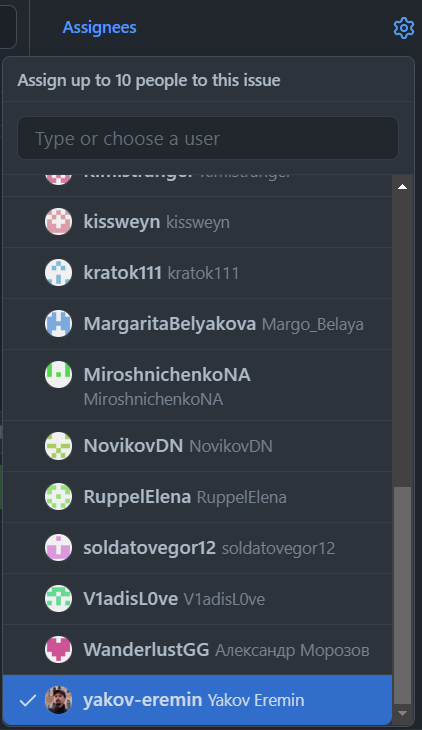


Рисунок 3 – Назначение ответственного за баг

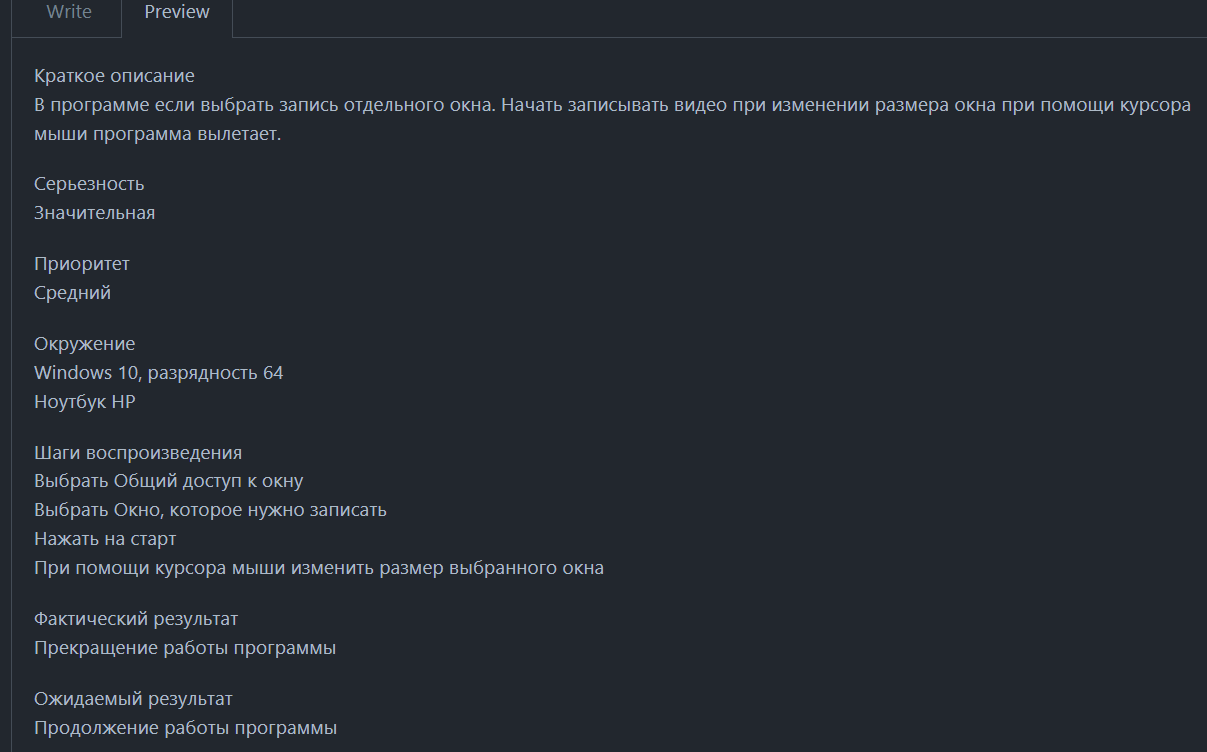


Рисунок 4 – Заполнение полей

В Python можно импортировать библиотеку для тестов

import unittest

import math

class Calculator:

#empty constructor

def \_\_init\_\_(self):

pass

def add\_x2(self, x1, x2):

return x1 + x2

def add\_x3(self, x1, x2, x3):

return x1 + x2 - x3

def multiply\_x2(self, x1, x2):

return x1 \* x2

def multiply\_x3(self, x1, x2, x3):

return x1 \* x2 + x3

def subtract\_x2(self, x1, x2):

return x1 - x2

def subtract\_x3(self, x1, x2, x3):

return x1 - x2 + x3

def divide(self, x1, x2):

if x2 != 0:

return x1/x2

def degree(self, x1, x2):

return x1\*\*x2

def area\_circle(self, r):

return (r\*\*2) \* math.pi

def area\_square(self, s):

return (s\*\*2)

def Sqrt(self, s):

math.sqrt(s)

Тесты

import unittest

import math

from Calculator import Calculator

class TestCalculator(unittest.TestCase):

def setUp(self):

self.calculator = Calculator()

def test\_add\_x2(self):

self.assertEqual(self.calculator.add\_x2(4,7), 11)

def test\_add\_x3(self):

self.assertEqual(self.calculator.add\_x3(2,2,2), 6)

def test\_multiply\_x2(self):

self.assertEqual(self.calculator.multiply\_x2(3,7), 21)

def test\_multiply\_x3(self):

self.assertEqual(self.calculator.multiply\_x3(2,2,2), 8)

def test\_subtract\_x2(self):

self.assertEqual(self.calculator.subtract\_x2(10,5), 5)

def subtract\_x3(self, x1, x2, x3):

self.assertEqual(self.calculator.subtract\_x3(10,5,5), 0)

def test\_divide(self):

self.assertEqual(self.calculator.divide(10,2), 5)

def test\_degree(self):

self.assertEqual(self.calculator.degree(3,4), 81)

def test\_area\_circle(self):

self.assertEqual(self.calculator.area\_circle(1), (1\*\*2) \* math.pi)

def test\_area\_square(self):

self.assertEqual(self.calculator.area\_circle(2), 4)

def test\_Sqrt(self):

self.assertEqual(self.calculator.area\_circle(4), 2)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()