МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

Факультет компьютерных технологий и прикладной математики

**Кафедра информационных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №7**

**ТЕСТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ**

Работу выполнил А.А.Козин

(подпись)

Направление подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем

Направленность Программирование и информационные технологии

Руководитель

канд. пед. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Н.Ю. Добровольская

(подпись)

Краснодар

2023

Оглавление

[РУЧНОЕ ТЕСТИРОВАНИЕ 3](#_Toc134786145)

[НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ 5](#_Toc134786146)

[СТРЕССОВОЕ ТЕСТИРОВАНИЕ 7](#_Toc134786147)

[UNIT-ТЕСТ 8](#_Toc134786148)

# РУЧНОЕ ТЕСТИРОВАНИЕ

Были протестированы следующие функции системы:

* Добавление и удаление преподавателей/курсов/кассиров/клиентов
* Добавление и удаление курсов
* Добавление и удаление кассиров
* Добавление и удаление клиентов
* Изменение списка преподавателей/курсов/кассиров/клиентов
* Экспорт базы данных в таблицу Microsoft Excel

В результате ручного тестирования была найдена следующая ошибка.

*Последовательность для обнаружения*: при удалении преподавателя/курса/кассира/клиента под номером N, следующая добавленная сущность будет под номером N+1, а не заменять удаленную сущность.

*Внешнее проявление ошибки* представлено на рисунке 1.

*Способ решения*: в классе преподавателя/курса/кассира/клиента переработать логику добавления и удаления. Также можно добавить кнопку “Изменить” для удобства пользователя.

Изображение выглядит как текст, монитор, снимок экрана, черный

Автоматически созданное описание

Рис. 1 – внешнее проявление ошибки.

# НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ

Технические характеристики машины:

* Процессор – Apple M1 Pro
* Оперативная память – 16ГБ
* Жесткий диск – Macintosh HD 512 ГБ

Для проведения нагрузочного тестирования используется библиотека “Faker”. Она позволяет генерировать большое количество данных, которые будут использованы для заполнения таблиц.

Было проведено нагрузочное тестирование на следующих объемах случайных данных:  
 N преподавателей, N курсов, N кассиров, N клиентов

Для значений N = 100, 500, 1000, 2500, 5000. Были произведены замеры времени экспорта данных в Excel и загрузки главного окна, в него входят:

* Список всех преподавателей
* Список всех курсов
* Список всех кассиров
* Список всех клиентов

По результатам нагрузочного тестирования, заметно меняется лишь время загрузки самого окна.

На таблице 1 и рисунке 2 приведены результаты по замерам.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Замеры | N = 100 | N = 500 | N = 1000 | N = 2500 | N = 5000 |
| Главное окно | 0.14 | 0.19 | 0.24 | 0.3 | 0.35 |
| Экспорт в Excel | 0.15 | 0.18 | 0.25 | 0.32 | 0.36 |

Таб. 1 – результаты нагрузочного тестирования (в секундах).

Рис. 2 – зависимость времени выполнения от N

# СТРЕССОВОЕ ТЕСТИРОВАНИЕ

Стрессовое тестирование проводилось на случайных данных, сгенерированных с N = 100000, после чего было замерено время ответа каждого из окон. Также был найден верхний порог генерации данных, они не генерируются начиная с N = 500000.

Результаты измерений приведены в таблице 2

|  |  |
| --- | --- |
| Измерения | N = 100000 |
| Главное окно | 2.5 |
| Экспорт в Excel | 15 |

Таб. 2 – результаты стрессового тестирования (в секундах).

Рис. 3 – график зависимости затраченного времени от количества записей при наличии большой сторонней нагрузки на оборудование

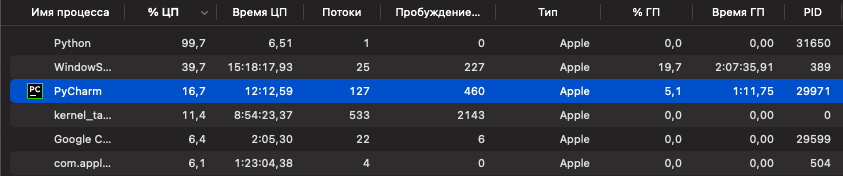


Рис. 4 – нагрузка ноутбука

# UNIT-ТЕСТ

Пустой проект Unit-тестов:

**import** unittest  
**import** sqlite3  
**from** main **import** Application  
**from** tkinter **import** Tk  
**from** database **import** create\_tables, insert\_sample\_data  
  
  
**class** TestApp(unittest.TestCase):  
  
 **def** setUp(self):  
 # Создание базы данных для тестирования  
 self.conn = sqlite3.connect(":memory:")  
 create\_tables(self.conn)  
 insert\_sample\_data(self.conn)  
  
 self.root = Tk()  
 self.app = Application(self.root, self.conn)  
  
 **def** tearDown(self):  
 self.conn.close()  
 self.root.destroy()  
  
 **def** test\_instructors\_tab(self):  
 # Добавление нового преподавателя  
 self.app.instructors\_frame.instructor\_first\_name\_entry.insert(0, "Иван")  
 self.app.instructors\_frame.instructor\_last\_name\_entry.insert(0, "Иванов")  
 self.app.instructors\_frame.instructor\_middle\_name\_entry.insert(0, "Иванович")  
 self.app.instructors\_frame.add\_instructor\_button.invoke()  
  
 # Проверка, что новый преподаватель добавлен в список  
 last\_instructor = self.app.instructors\_frame.instructors\_listbox.get(  
 self.app.instructors\_frame.instructors\_listbox.size() - 1)  
 self.assertEqual(last\_instructor, "Иванов Иван Иванович")  
  
 # Удаление преподавателя  
 self.app.instructors\_frame.instructors\_listbox.selection\_set(0)  
 self.app.instructors\_frame.delete\_instructor\_button.invoke()  
  
 # Проверка, что преподаватель был удален  
 deleted\_instructor = self.app.instructors\_frame.instructors\_listbox.get(0)  
 self.assertNotEqual(deleted\_instructor, "Иванов Иван Иванович")  
  
  
**if** \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()