**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**

**Отчет о выполнении лабораторной работы №8**

**по дисциплине**

**«Технологии проектирования программного обеспечения»**

Направление подготовки 01.03.02 Прикладная математика и информатика

Работу выполнил студент группы 4ИТ/2 Д.А. Качура

(подпись)

Работу проверил доц. каф. ИТ, к.т.н., доц. А.Н. Полетайкин

(подпись)

Краснодар

2023

**ВВЕДЕНИЕ**

**Тема**: Тестирование программной системы

**Цель**: Освоение методики тестирования разработанной программной системы в ручном режиме.

**Задание:**

**1.** Выполнить системное пользовательское тестирование работоспособности ПС посредством воздействия на её интерфейсную часть, разработанную при выполнении лабораторной работы №6. При возникновении несоответствий задокументировать их, произвести поиск логических ошибок, разработать тесты для их обнаружения, устранить ошибки.

**2.** Выполнить нагрузочное тестирование программы и оценить эффективность разработанных при выполнении лабораторной работы №6 автоматизированных функций ПС, запросов к БД. В качестве критерия эффективности использовать время выполнения функции. Предварительно подготовить 5 массивов исходных данных на 10, 50, 100, 500 и 1000 записей. Построить графики зависимости времени вычислений от объема исходных данных.

**3.** Выполнить стрессовое тестирование ПС. В случае обнаружения некорректных ситуаций описать их, выявить причины и принять меры к их устранению.

**4.** Разработать модульный тест для проверки работоспособности кода, реализующего автоматизированные функции ПС.

**Индивидуальная тема:** программное средство для моделирования вольтамперных характеристик

**1. Системное (ручное) пользовательское тестирование ПС.**

Системное тестирование – это тестирование всей системы в целом, как правило, через ее пользовательский интерфейс. Следует проверить соответствует ли разработанная ПС заявленным требованиям.

Система адаптивна под различные разрешения экрана, интерфейс простой и незапутанный.

Для корректного тестирования необходимо провести 2 теста: со стороны пользователя и со стороны администратора.

**1.1 Тест №1 (Инженер)**

Состоит из проверки функционала пользователя состоит из следующих действий:

1. Зайти на страницу авторизации профиля
2. Авторизоваться под неправильными данными пользователя
3. Проверить валидацию данных
4. Авторизоваться с помощью корректных данных пользователя
5. Заполнить поля параметров некорректными данными и проверить валидацию
6. Проверить, что программа корректно отображает введенные параметры и вычисляет соответствующие вольтамперных характеристики
7. Проверить, что графики ВАХ отображаются правильно и соответствуют ожидаемым значениям
8. Загрузить файл с экспериментальной ВАХ
9. Проверить, что программа корректно загружает файл и отображает экспериментальные данные на графике
10. Загрузить файл с неверно введенной экспериментальной вольтамперной характеристикой и проверить, что программа корректно обработает ошибку
11. Нажать на кнопку сохранить отчет и проверить корректность работы

Ожидается, что все элементы интерфейса корректно работают. Этот тест успешно пройден.

Скриншоты срабатывания некоторых пунктов проверки:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1 – Интерфейс начальной страницы приложения при неправильно введенных данных

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2 – Интерфейс начальной страницы приложения при правильно введенных данных

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание

Рисунок 3 – Интерфейс экрана моделирования

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание

Рисунок 4 – Обработка ошибки файла с неправильными характеристиками

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание

Рисунок 5 – Загрузка корректного файла с характеристиками

**2. Нагрузочное тестирование**

Выполним нагрузочное тестирование программы и оценим эффективность автоматизированных функций ПС, запросов к БД. В качестве критерия эффективности используем время выполнения функции. Для тестирования подготовим 5 массивов исходных данных на 10, 50, 100, 500 и 1000. Тестирование проводилось на таблице клиентов. Результаты тестирования представлены в таблице 1.

Технические характеристики используемой машины:

* Оперативная память: 16 Гб
* Процессор: 4 ядра с тактовой частотой 2.333 ГГц, 8 потоков
* Разрядность Windows: 64-разрядная операционная система
* Размер жёсткого диска: 512 Гб

|  |  |  |  |
| --- | --- | --- | --- |
| № | Кол-во записей | Операция | Результат (сек.) |
| 1 | 10 | Вставка | 0,2445 |
| 2 | 50 | Вставка | 0,2636 |
| 3 | 100 | Вставка | 0,3134 |
| 4 | 500 | Вставка | 0,4321 |
| 5 | 1000 | Вставка | 0,7611 |

Таблица 1 – Сравнительная таблица нагрузочного тестирования

При данном тестировании в базу данных загружается сначала 10 записей об объектах, далее 50, 100, 500 и 1000. Далее строится проект, тем самым проверяется скорость работы ПС с различными объёмами данных

После проведения теста мы получили следующие данные:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Количество исходных данных | 10 | 50 | 100 | 500 | 1000 |
| Время вычислений | 1870 мс | 1980 мс | 5093 мс | 5516 мс | 9452 мс |

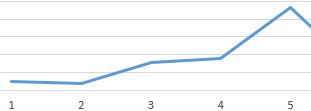


Рисунок 6 – График работы программы

**3. Стрессовое тестирование**

В этом виде тестирования необходимо проверить работу системы при чрезмерной нагрузке на неё. Стрессовое тестирование ПС будет проведено посредством отправки многократных запросов к БД PostgreSQL. Во время тестирование было обнаружено, что даже в стрессовых ситуациях приложение продолжает функционировать, БД быстро обрабатывает запросы. Могут возникать небольшие задержки в работе интерфейса, но после выполнения череды запросов задержки перестают себя проявлять.

**4. Модульное тестирование**

Для тестирования работы вставки был написан следующий тест:

import unittest  
from login import check\_credentials  
  
class TestAuthorization(unittest.TestCase):  
 def test\_login\_valid\_credentials(self):  
 # Проверка авторизации с правильными учетными данными  
 email = "user@example.com"  
 password = "password123"  
 expected\_result = True  
  
 result = check\_credentials(email, password)  
 self.assertEqual(result, expected\_result)  
  
 def test\_login\_invalid\_credentials(self):  
 # Проверка авторизации с неправильными учетными данными  
 email = "user@example.com"  
 password = "wrongpassword"  
 expected\_result = False  
  
 result = check\_credentials(email, password)  
 self.assertEqual(result, expected\_result)  
  
 def test\_login\_empty\_credentials(self):  
 # Проверка авторизации с пустыми учетными данными  
 email = ""  
 password = ""  
 expected\_result = False  
  
 result = check\_credentials(email, password)  
 self.assertEqual(result, expected\_result)  
  
 def test\_login\_case\_sensitive(self):  
 # Проверка авторизации с учетом регистра символов  
 email = "USER@example.com"  
 password = "Password123"  
 expected\_result = True  
  
 result = check\_credentials(email, password)  
 self.assertEqual(result, expected\_result)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 7 – результат модульного теста

**Вывод:** в рамках данной работы была протестирована программная система различными способами, освоены методики тестирования разработанной программной системы в ручном режиме, проведено успешное системное, нагрузочное, стрессовое и модульное тестирования, также в работе приведён код модульных тестов.