



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ,

информационные технологии»

Лабораторная работа №5

«Тестирование, как способ контроля качества ПО»

ДИСЦИПЛИНА: «Методология программной инженерии»

Выполнил: студент гр. ИУК4-11М _____ (Сафронов Н.С.)
(подпись) (Ф.И.О.)

Проверил: _____ (Белов Ю.С.)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2024

Цель работы: освоение методики обеспечения качества разрабатываемого программного кода, тестирования программы в ручном режиме, анализе производительности.

Постановка задачи

Вариант 8

ИС «Торговля» (САБП, СЭДО).

1. Выполнить полное системное ручное тестирование программы, разработанной при выполнении лабораторной работы №3, методом черного ящика. При возникновении некорректных ситуаций идентифицировать логические ошибки, разработать тесты для их обнаружения, устранить ошибки.
2. Выполнить анализ производительности программы на разных вычислительных платформах (не менее трех). Оценить время выполнения пользователем определенной последовательности действий на каждой из выбранных вычислительных платформ.
3. Выполнить нагрузочное тестирование программы и оценить эффективность разработанных при выполнении лабораторной работы №3 методов. В качестве критерия эффективности использовать время выполнения метода.
4. Произвести рефакторинг методов ИС
5. Выполнить стрессовое тестирование программного продукта. В случае обнаружения некорректных ситуаций описать их, выявить причины и принять меры к их устранению.

Результат выполнения работы

Функциональные тесты

1. Авторизация пользователя

Предусловие: Пользователь зарегистрирован в системе.

Действия: Ввести корректные логин и пароль на странице авторизации, нажать "Войти".

Ожидаемый результат: Пользователь успешно входит в систему и перенаправляется на главную страницу.

Рисунок 1.1 – Ввод
данных в окно
авторизации

Рисунок 1.2 – Результат авторизации

2. Ошибка при неверной авторизации

Предусловие: Пользователь зарегистрирован в системе.

Действия: Ввести некорректные логин и/или пароль, нажать "Войти".

Ожидаемый результат: Появляется сообщение об ошибке "Неверный логин или пароль".

Рисунок 2.1 – Ввод некорректного
имени пользователя

Рисунок 2.2 – Ввод некорректного пароля

3. Просмотр списка товаров

Предусловие: Пользователь авторизован.

Действия: Перейти на страницу "Товары".

Ожидаемый результат: Отображается таблица с товарами (название, описание, цена, наличие).

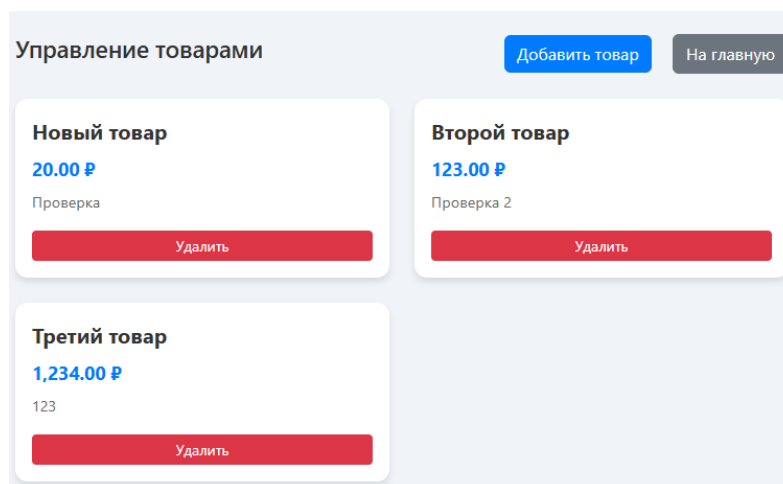


Рисунок 3 – Результат просмотра списка товаров

4. Добавление нового товара

Предусловие: Пользователь имеет роль администратора.

Действия: На странице "Товары" нажать "Добавить товар", заполнить форму (название, цена, описание, количество) и сохранить.

Ожидаемый результат: Новый товар отображается в списке.

The screenshot shows a modal form titled 'Добавить товар'. It has four input fields: 'Название' (Name) with the value 'Четвёртый товар', 'Цена' (Price) with the value '1000', 'Описание' (Description) with the value 'Проверка товара', and a fourth field labeled 'Проверка товара'. At the bottom of the form, there are two buttons: 'Отмена' (Cancel) and 'Сохранить' (Save).

Рисунок 4.1 – Добавление
товара

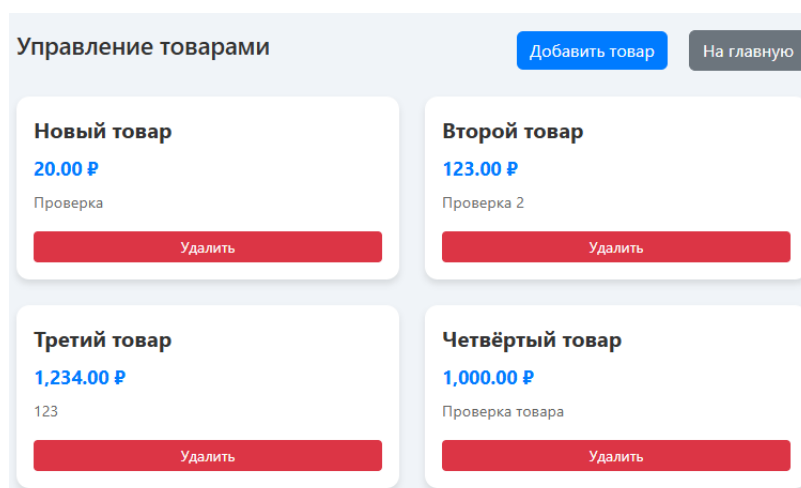


Рисунок 4.2 – Добавленный товар

5. Удаление товара

Предусловие: Товар существует в базе данных.

Действия: На странице "Товары" выбрать товар, нажать "Удалить", подтвердить действие.

Ожидаемый результат: Товар удален из системы и больше не отображается в списке.

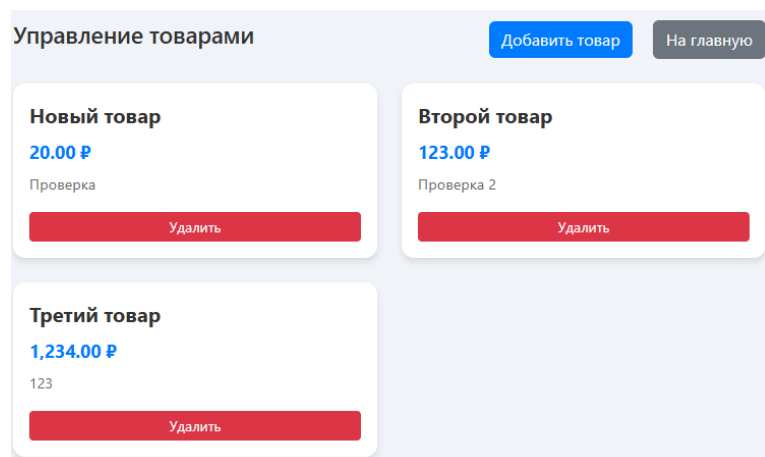
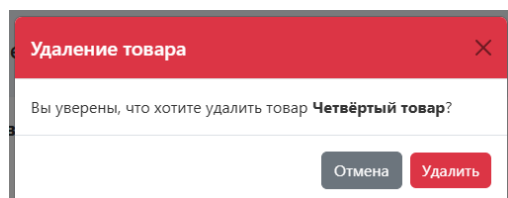


Рисунок 5.1 – Удаление товара

Рисунок 5.2 – Результат удаления товара

6. Просмотр списка клиентов

Предусловие: Пользователь авторизован.

Действия: Перейти на страницу "Клиенты".

Ожидаемый результат: Отображается таблица с данными клиентов (имя, контактные данные, история заказов).

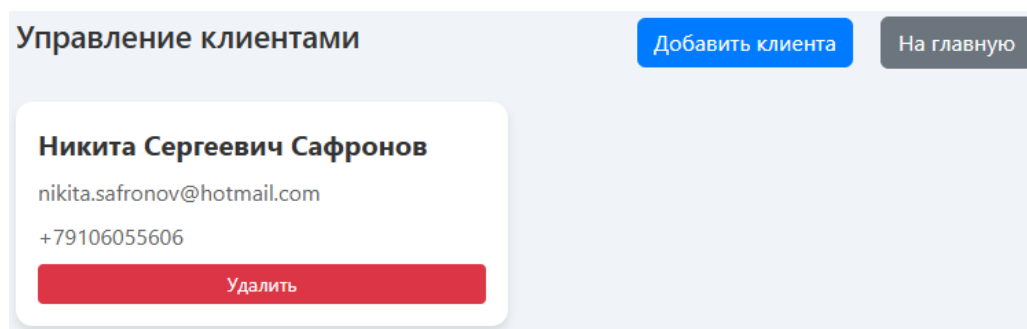


Рисунок 6 – Список клиентов

7. Добавление клиента

Предусловие: Пользователь имеет доступ к управлению клиентами.

Действия: На странице "Клиенты" нажать "Добавить клиента", заполнить форму (имя, контакты, примечания) и сохранить.

Ожидаемый результат: Новый клиент отображается в списке.

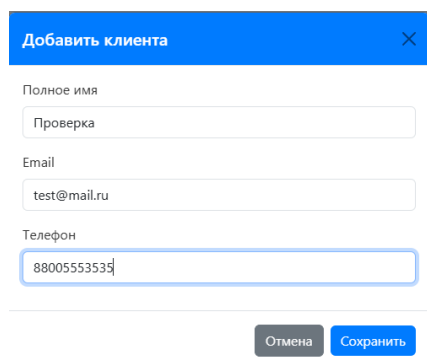


Рисунок 7.1 – Добавление клиента

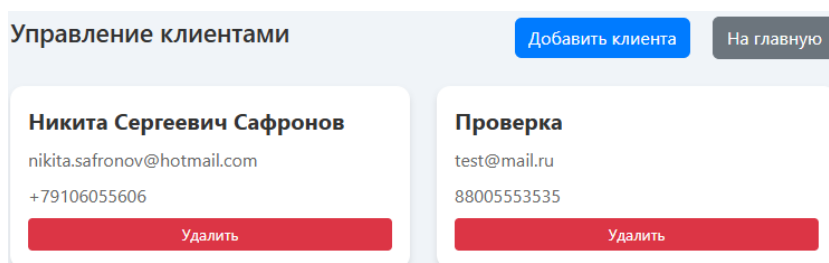


Рисунок 7.2 – Результат добавления клиента

8. Просмотр списка заказов

Предусловие: Пользователь авторизован.

Действия: Перейти на страницу "Заказы".

Ожидаемый результат: Отображается таблица с заказами (стоимость, клиент, сотрудник и товары).

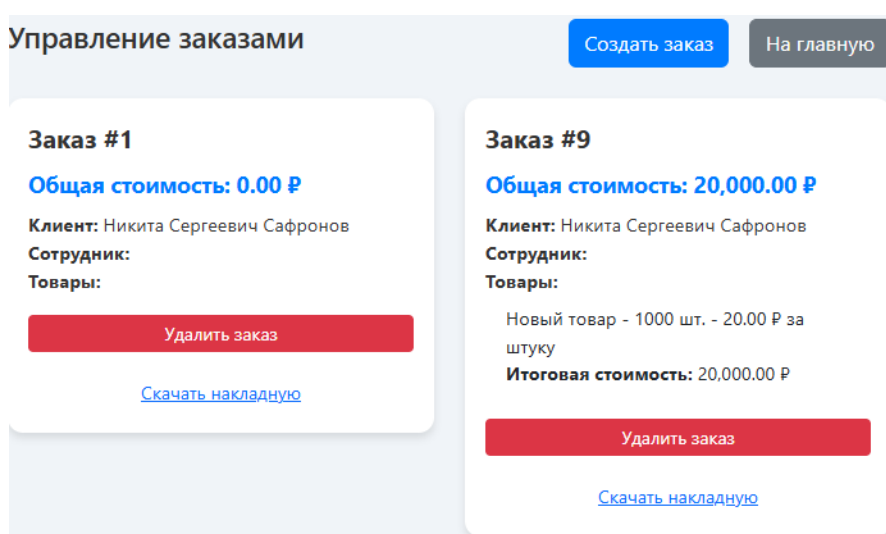


Рисунок 8 – Список заказов

8. Создание нового заказа

Предусловие: Пользователь имеет доступ к управлению клиентами.

Действия: Перейти на страницу "Заказы", нажать "Создать заказ", выбрать клиента и товары, указать количество, сохранить заказ.

Ожидаемый результат: Новый заказ отображается в списке.

Скриншот формы "Создать новый заказ". Вверху есть заголовок "Создать новый заказ" и уведомление "Товар добавлен в заказ!". Форма содержит поля для ввода ID клиента (значение 1), ID товара (значение 4) и количества товара (значение 100). Ниже этих полей отображается информация о выбранном товаре: "Третий товар (1234 Р)". Внизу формы есть кнопка "Добавить товар" и таблица "Товары в заказе:" с двумя строками: "Второй товар - 100 шт. - 123 Р за штуку" и "Третий товар - 100 шт. - 1234 Р за штуку". В каждой строке есть кнопка "Удалить". В самом низу формы находятся кнопки "Отмена" и "Создать заказ".

Рисунок 8.1 – Добавление заказа

Скриншот карточки заказа #12. Вверху указано "Заказ #12" и "Общая стоимость: 135,700.00 Р". Далее следуют поля "Клиент: Никита Сергеевич Сафронов" и "Сотрудник:". Раздел "Товары:" содержит две строки: "Второй товар - 100 шт. - 123.00 Р за штуку" и "Третий товар - 100 шт. - 1,234.00 Р за штуку". В конце раздела "Товары:" указаны "Итоговая стоимость: 12,300.00 Р" и "Итоговая стоимость: 123,400.00 Р". Внизу карточки есть красная кнопка "Удалить заказ" и ссылка "Скачать накладную".

Рисунок 8.2 – Результат добавления заказа

9. Удаление заказа

Предусловие: Пользователь авторизован.

Действия: Перейти на страницу "Заказы".

Ожидаемый результат: Отображается таблица с заказами (стоимость, клиент, сотрудник и товары).

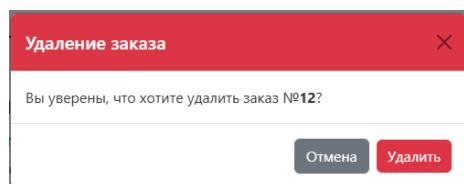


Рисунок 9.1 – Удаление заказа

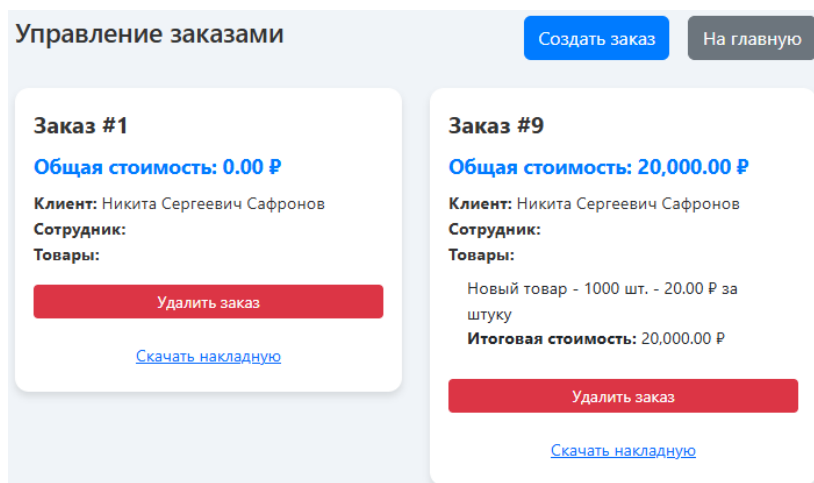


Рисунок 9.2 – Результат удаления заказа

Анализ производительности программы на разных вычислительных платформах

Для анализа выбран один из ключевых функциональных запросов системы — операция добавления нового клиента. Этот запрос отражает характерные действия, которые регулярно выполняются в системе, и включает взаимодействие с базой данных, обработку пользовательских данных и выполнение бизнес-логики. Тестирование будет проводиться на трех вычислительных платформах, различающихся по техническим характеристикам, что позволит определить влияние аппаратных и программных факторов на время выполнения операции.

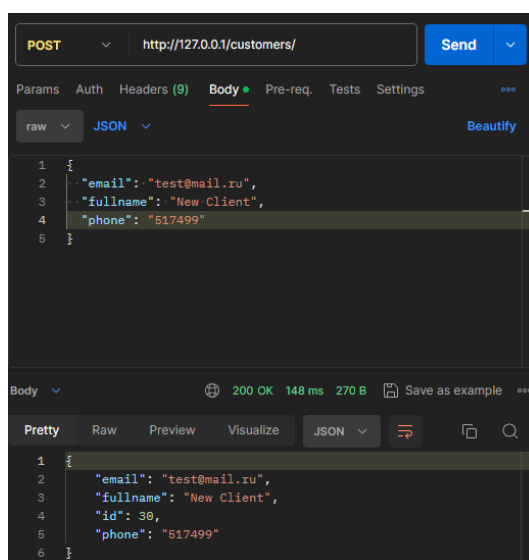


Рисунок 10 – Запрос на добавление клиента

Запустим серверное приложение на трёх различных платформах и получим следующие результаты:

Таблица 1

Выполнение запроса на добавление клиента

Платформа	Конфигурация	Среднее время выполнения запроса, мс
ПК	Intel Core I5-10400F 16 GB RAM	35
Ноутбук	Intel Core I5-1235 24 GB RAM	23
Платформа	Intel Core N300 8 GB RAM	148

Нагрузочное тестирование программы

Для оценки производительности системы было проведено нагрузочное тестирование, целью которого являлось определение времени выполнения ключевых запросов при различных сценариях работы. Основное внимание уделено операциям, связанным с обработкой большого объема данных и интенсивными вычислениями.

Таблица 2

Время выполнения различных запросов

Запрос	Среднее время выполнения запроса, мс
Получение страницы с 1000 товаров	91
Получение страницы с 1000 клиентов	93
Получение страницы с 1000 заказов	1355
Добавление заказа с 1000 позиций товаров	278

Операции, связанные с выборкой данных (например, получение страницы с заказами), требуют значительно большего времени, что связано с более сложной структурой данных заказов.

Добавление заказа демонстрирует приемлемое время выполнения, но при увеличении числа позиций в заказе может потребоваться дополнительная оптимизация.

Стрессовое тестирование программного продукта

Для выполнения стрессового тестирования в рамках данной работы будет использован инструмент Locust — популярный фреймворк для проведения нагрузочного и стрессового тестирования, который позволяет эмулировать действия пользователей и измерять производительность системы при увеличении количества запросов. Locust предоставляет гибкий подход к моделированию нагрузки с помощью Python-скриптов, что позволяет легко настраивать сценарии тестирования и получать точные данные о производительности приложения.

Листинг сценария тестирования:

```
import time
from locust import HttpUser, task, between

class QuickstartUser(HttpUser):
    wait_time = between(1, 5)

    @task(3)
    def view_orders(self):
        self.client.get("/orders/")

    @task(3)
    def view_products(self):
        self.client.get("/products/")

    @task(3)
    def view_customers(self):
        self.client.get(f"/customers/")

    @task(4)
    def create_customer(self):
        self.client.post(
            "/customers/",
            json=dict(
                fullname='test',
                email='test email',
                phone='88005553535',
            )
        )
```

```

    )

    @task(3)
    def create_product(self):
        self.client.post(
            "/products/",
            json=dict(
                name='test',
                price=10.0,
                description='stress testing',
            )
        )

    @task(3)
    def create_order(self):
        self.client.post(
            "/orders/",
            json=dict(
                items=[
                    dict(
                        product_id=2,
                        quantity=100,
                    ),
                ],
                customer_id=1,
            )
        )

    def on_start(self):
        self.client.post("/users/", json={"username": "admin",
        "password": "121212aa"})

```

Запустим стресс-тест для 100 пользователей по этому сценарию:

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/customers/	26	0	30	67	67	36.78	26	67	622984.96	0.1	0
POST	/customers/	31	0	6	8	8	6.56	6	8	73	0.1	0
GET	/orders/	34	0	1300	1600	1700	1351.66	1195	1660	1998494.18	0.4	0
POST	/orders/	22	0	12	16	16	12.32	10	16	269	0	0
GET	/products/	16	0	31	78	78	37.51	28	78	627259.13	0.2	0
POST	/products/	18	0	7	9	9	7.07	6	9	72	0.2	0
POST	/users/	38	0	45000	106000	107000	50372.76	95	107407	4387	0.3	0
Aggregated		185	0	31	88000	106000	10606.92	6	107407	510047.24	1.3	0

Рисунок 11 – Результаты стресс-теста

Таким образом, получаем следующие результаты тестирования:

- Все запросы, как GET, так и POST, завершились без ошибок, что подтверждается нулевым количеством сбоев в каждом из случаев.
- Время отклика для большинства запросов оказалось в пределах допустимых норм. Запросы типа GET к страницам /customers/, /orders/ и /products/ показали медианное время отклика от 30 до 78 мс, что является хорошим результатом для API.
- Запросы POST, такие как /customers/, /orders/ и /products/, также показали низкое время отклика с медианой от 6 до 9 мс, что указывает на оптимальную работу этих операций.
- Однако запросы, связанные с заказами (GET /orders/), показали значительно более высокое время отклика, с медианой в 1300 мс и 99-м перцентилем на уровне 1600 мс. Это обусловлено алгоритмами, лежащими в основе подсчёта стоимости заказов, а также получением и созданием айтемов заказов.
- Особое внимание стоит уделить POST-запросу для /users/, который показал крайне высокие значения времени отклика: медиана составила 45,000 мс, а 99-й перцентиль — 107,000 мс. Приложение использует FlaskLogin в качестве системы аутентификации – возможно, придётся искать более приемлемое решение.
- В плане пропускной способности, текущий RPS (Requests Per Second) для большинства запросов оставался низким (около 0.1), что указывает на возможное увеличение нагрузки на систему, однако все запросы обрабатывались без ошибок, и система справлялась с тестовой нагрузкой.

Вывод: в ходе выполнения лабораторной работы были сформированы навыки разработки информационной системы в соответствии с предъявляемыми требованиями.