**Содержание**

[Введение 3](#_Toc195633611)

[1 Этапы выполнения работ 3](#_Toc195633612)

[1.1 Создание проекта на GitHub 3](#_Toc195633613)

[2. Техническое задание 4](#_Toc195633614)

[3. Разработка программного продукта 5](#_Toc195633615)

[4. Тестирование программного продукта 7](#_Toc195633616)

[5 Настройка Linter 12](#_Toc195633617)

[Заключение 13](#_Toc195633618)

[Список используемых источников 15](#_Toc195633619)

[Приложение А – Техническое задание 16](#_Toc195633620)

# Введение

Целью производственной практики являлось применение теоретических знаний на практике, а также приобретение навыков в области защиты и администрирования СУБД PostgreSQL.

Задачи практики:

– Создание проекта на GitHub;

– Составить техническое задание на разработку базы данных, выложить на GitHub;

– Написать модуль приема данных извне, проверки на корректность и последующей записи в базу данных на основе PostgreSQL;

– Выполнить отладку;

* Подготовить тестовый набор данных, для запросов GET и POST с помощью Postman;
* Указать ссылку на проект GitHub.

# Этапы выполнения работ

# Создание проекта на GitHub

В начале работы над проектом был создан репозиторий на GitHub, скриншот этого репозитория представлен на рисунке 1.

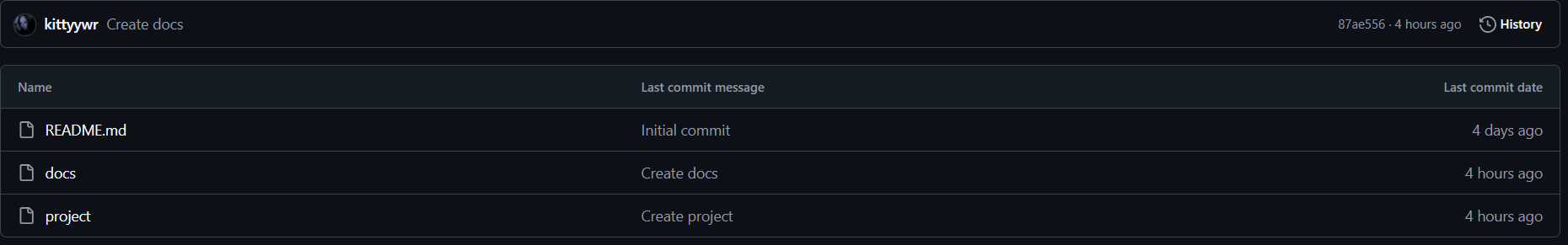


Рисунок 1 – проект на GitHub

Затем была создана база данных в PostgreSQL для выполнения следующих задач. Схема базы данных представлена на рисунке 2.

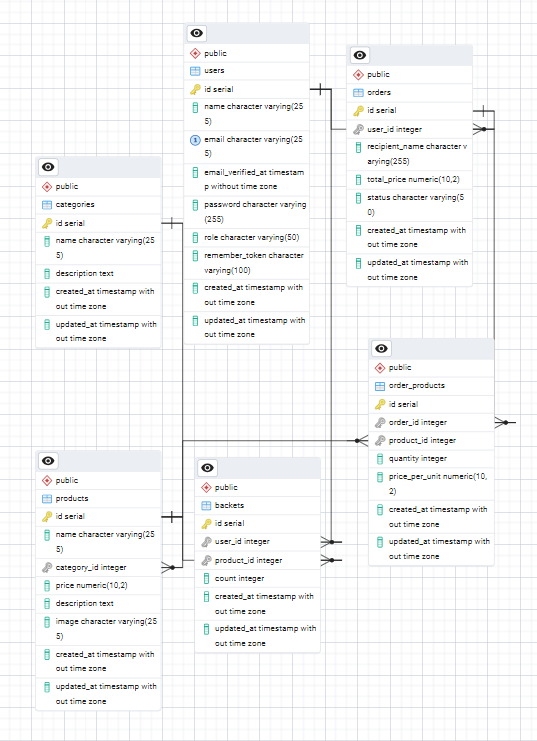


Рисунок 2 – база данных

## 2. Техническое задание

В начале разработки создавалось техническое задание, в котором указывались основные требования.

Для создания технического задания использовался стандарт ГОСТ 34.602-2020.

Согласно ГОСТ 34.602-2020 техническое задание должно включать следующие разделы:

Введение.

1. Общие сведения.

1.2 Цели и назначение создания автоматизированной системы

2. Основания для разработки.

2.1. Нормативные документы.

3. Назначение системы.

3.1. Общее описание.

3.2. Преимущества и новизна.

4. Требование к системе.

4.1 Функциональные требования.

4.2 Технические требования.

4.3. Эксплуатационные требования.

5. Требования к техническому обеспечению.

5.1 Оборудование.

5.2 Сетевые требования.

6. Требования к программному обеспечению.

6.1. Программные компоненты.

6.2. Интерфейсы.

# 3. Разработка программного продукта

Для разработки программного продукта использовался фреймворк Laravel.

После того, как был создан интерфейс программного продукта и реализована база данных, необходимо создать модели, контроллеры и маршруты.

Модели необходимы для взаимодействия с базой данных. На рисунке 3 представлен код модели категорий.

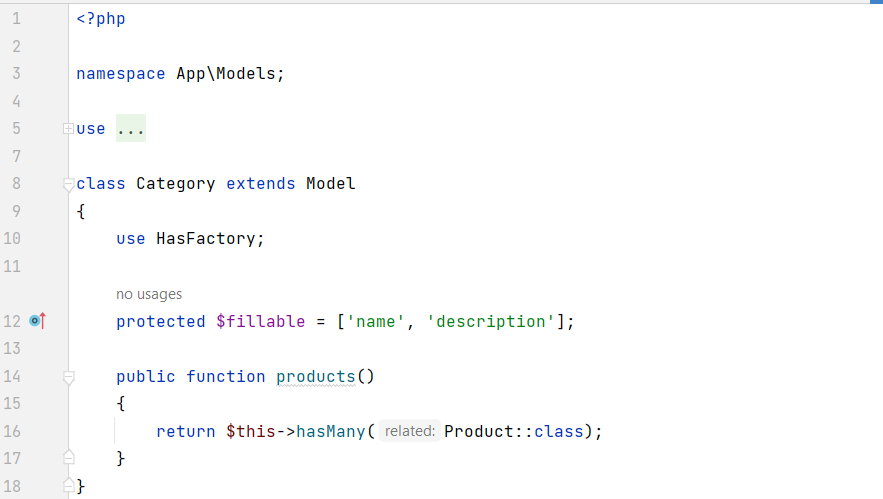


Рисунок 3 – Код модели категорий

На данном рисунке представлена модель категорий, в которой описаны атрибуты и связи между моделями.

Далее создаются контроллеры для обработки запросов и данных. На рисунке 4 представлен контроллер администратора, в котором созданы все методы для управления данными в веб-приложении.

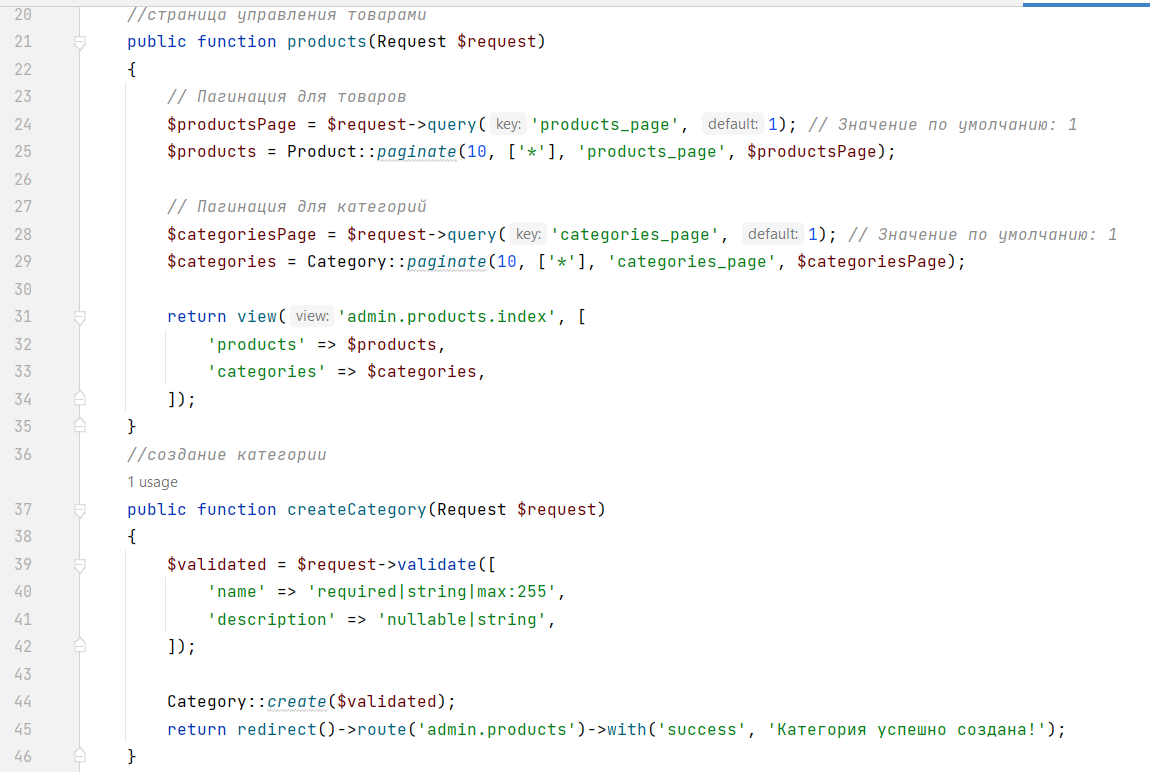


Рисунок 4 – Контроллер администратора

На рисунке представлена часть кода, а именно обработка и открытие страницы управления товарами, а также метод создания новой категории.

Далее были созданы маршруты. Маршрутизация в Laravel – это процесс определения, какие действия должны выполняться при обращении к определённым URL-адресам веб-приложения. На рисунке 5 представлен файл маршрутов.



Рисунок 5 – Код файла маршрутов

На данном рисунке отображены маршруты доступные только администратору.

## 4. Тестирование программного продукта

Тестирование веб-приложения – это процесс проверки и оценки качества веб-приложений или сайтов. Он помогает обеспечить их надёжность, производительность, безопасность и совместимость с различными браузерами, устройствами и операционными системами.

Модульное тестирование (Unit Testing)– это тип тестирования программного обеспечения, при котором тестируются отдельные модули или компоненты программного обеспечения. Его цель заключается в том, чтобы проверить, что каждая единица программного кода работает должным образом. Данный вид тестирование выполняется разработчиками на этапе кодирования приложения. Модульные тесты изолируют часть кода и проверяют его работоспособность.

Было проведено функциональное тестирование с использованием PHPUnit.

Для проведения тестирования был создан тест-план, который отражает какие ресурсы были использованы и какие этапы пройдены. Тест-план представлен в таблице 10.

Таблица 10 – Тест-план

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название теста | Описание | Ожидаемый результат |
| 1 | Создание товара | Проверка создания нового товара с валидными данными. | Товар успешно создан, данные сохранены в базе данных. |
| 2 | Обновление товара | Проверка обновления существующего товара с валидными данными. | Товар успешно обновлен, данные обновлены в базе данных. |
| 3 | Удаление товара | Проверка удаления существующего товара. | Товар успешно удален, запись удалена из базы данных. |

Далее были разработаны чек листы для тестов.

В таблице 11 представлен чек-лист для теста "Создание товара".

Таблица 11 – Чек-лист для теста "Создание товара"

|  |  |  |
| --- | --- | --- |
| № | Действие | Ожидаемый результат |
| 1 | Открыть форму создания товара. | Форма создания товара отображается корректно. |
| 2 | Ввести валидные данные (название, цена, описание, категория). | Поля формы заполняются без ошибок. |
| 3 | Отправить форму. | Товар успешно создан, появляется сообщение об успешном создании. |
| 4 | Проверить, что товар появился в базе данных. | Товар сохранен в базе данных с введенными данными. |
| 5 | Проверить, что товар отображается в списке товаров. | Товар отображается в списке товаров с корректными данными. |

На рисунке 6 представлен код данного теста.

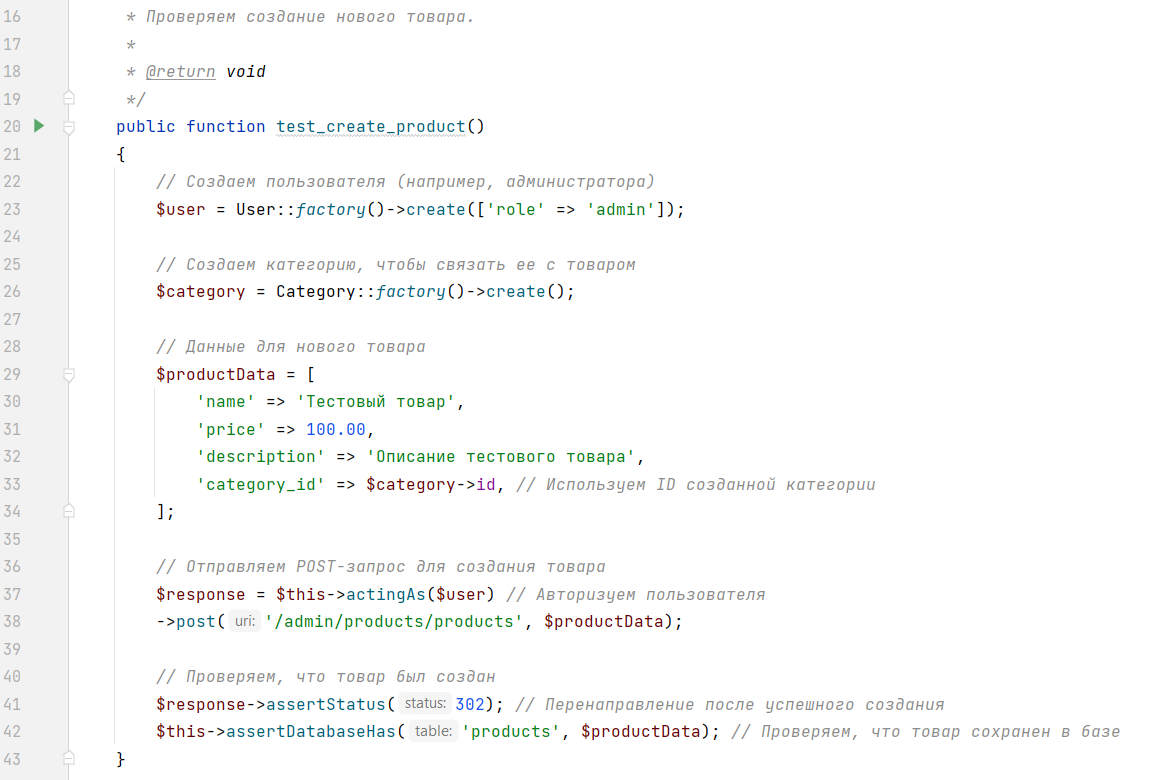


Рисунок 6 – Код теста на создание товара

В таблице 12 представлен чек-лист для теста "Обновление товара".

Таблица 12 – Чек-лист для теста "Обновление товара"

|  |  |  |
| --- | --- | --- |
| № | Действие | Ожидаемый результат |
| 1 | Открыть форму редактирования существующего товара. | Форма редактирования товара отображается корректно. |
| 2 | Ввести новые валидные данные (название, цена, описание, категория). | Поля формы заполняются без ошибок. |
| 3 | Отправить форму. | Товар успешно обновлен, появляется сообщение об успешном обновлении. |
| 4 | Проверить, что данные товара обновлены в базе данных. | Данные товара в базе данных соответствуют введенным значениям. |
| 5 | Проверить, что обновленный товар отображается в списке товаров. | Обновленный товар отображается в списке товаров с новыми данными. |

На рисунке 7 представлен код данного теста.

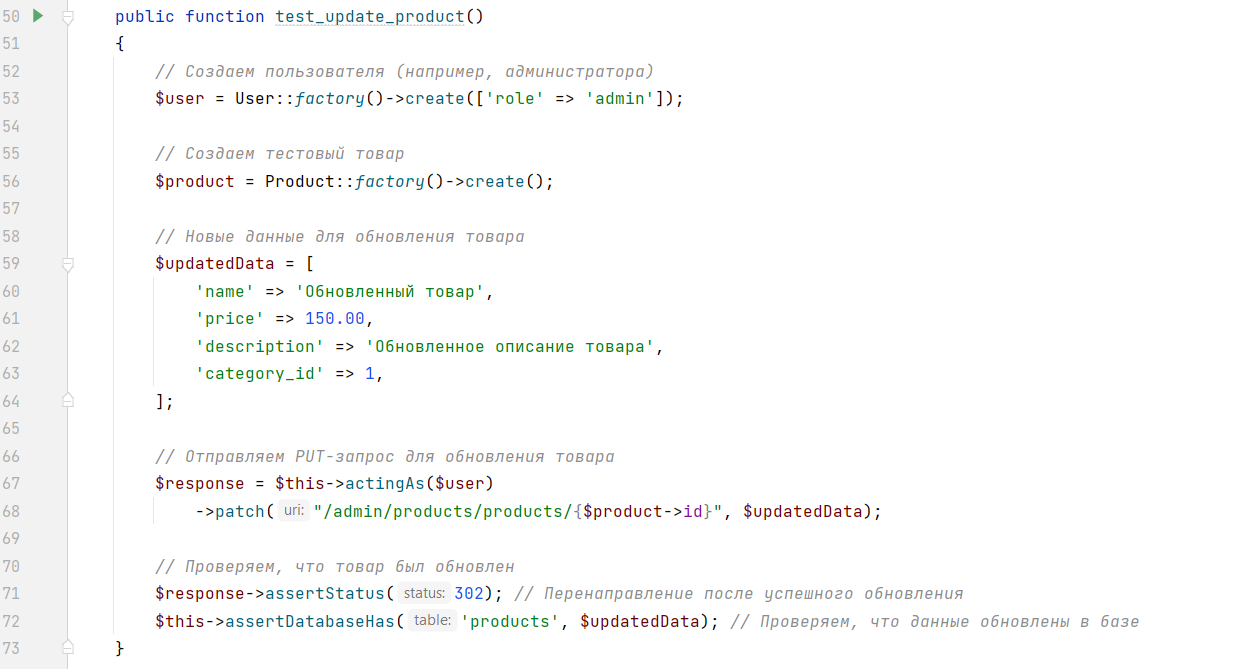


Рисунок 7 – Код теста на обновление товара

В таблице 13 представлен чек-лист для теста "Удаление товара".

Таблица 13 – Чек-лист для теста "Удаление товара"

|  |  |  |
| --- | --- | --- |
| № | Действие | Ожидаемый результат |
| 1 | Открыть список товаров. | Список товаров отображается корректно. |
| 2 | Найти товар, который нужно удалить. | Товар найден в списке. |
| 3 | Нажать кнопку "Удалить" для выбранного товара. | Появляется подтверждение удаления (если требуется). |
| 4 | Подтвердить удаление. | Товар успешно удален, появляется сообщение об успешном удалении. |
| 5 | Проверить, что товар удален из базы данных. | Товар больше не существует в базе данных. |
| 6 | Проверить, что товар больше не отображается в списке товаров. | Товар больше не отображается в списке товаров. |

На рисунке 8 представлен код данного теста.



Рисунок 8 – Код теста на удаление товара

На рисунке 9 и 10 представлены результаты выполнения тестов.



Рисунок 9 – Результат теста GET-запрос

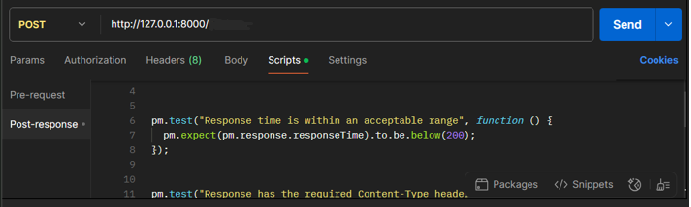


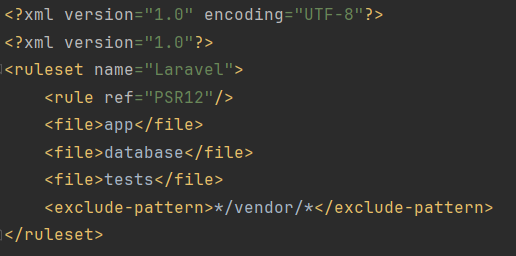
Рисунок 10 – Результат теста POST-запрос

## 5 Настройка Linter

**Linter** – это инструмент для статического анализа кода, который проверяет его на соответствие стандартам оформления, потенциальные ошибки и уязвимости. В PHP-проектах он помогает:

1. Соблюдать **PSR-стандарты** (PSR-1, PSR-2, PSR-12)
2. Находить **синтаксические ошибки** до запуска кода
3. Улучшать **читаемость** и поддерживаемость кода
4. Автоматически **форматировать** код

На картинках 10 и 11 представлена проверка стиля кода для стандарта PSR-12.



Картинка 10 – Проверка стиля кода



Картинка 11 – Проверка стиля кода

# Заключение

В ходе производственной практики были успешно выполнены все поставленные задачи, направленные на **применение теоретических знаний** в области администрирования и защиты СУБД **PostgreSQL**, а также разработки веб-приложений на **Laravel**.

**Основные достижения:**

1. **Создан и размещен проект на GitHub**

Репозиторий включает полную структуру Laravel-приложения для управления библиотекой.

Ссылка на проект: https://github.com/kittyywr/pp2/tree/main.

1. **Разработано техническое задание**

Составлено в соответствии с **ГОСТ 34.602-2020**, включая требования к функционалу, безопасности и архитектуре БД.

1. **Реализован модуль приема данных**

Настроена валидация и запись данных в PostgreSQL.

Использованы миграции, модели Eloquent и RESTful API.

1. **Проведена отладка и тестирование**

Настроены **PHP\_CodeSniffer** и **Laravel Pint** для контроля качества кода.

Созданы тестовые запросы (**GET/POST**) в **Postman** и автоматизированные тесты.

1. **Настроена СУБД PostgreSQL**

Оптимизированы запросы, добавлены индексы.

Реализовано резервное копирование.

**Приобретенные навыки:**

1. Работа с **PostgreSQL**: проектирование БД, репликация, мониторинг.
2. Разработка API на **Laravel** (Eloquent, миграции, REST).
3. Инструменты контроля версий (**Git/GitHub**) и CI/CD.
4. Тестирование (**Postman**, PHPUnit).

Список используемых источников

1. **Документация PostgreSQL** – [Электронный ресурс]. – URL:  https://www.postgresql.org/docs/ (дата обращения: 24.03.2025).
2. **Официальный сайт PgBouncer** – [Электронный ресурс]. – URL: https://pgbouncer.github.io/  (дата обращения: 31.03.2025).
3. **Руководство по NMAP** – [Электронный ресурс]. – URL: https://nmap.org/book/man.html (дата обращения: 03.04.2025).
4. ГОСТ 34.602-2020 – [Электронный ресурс]. – URL:  https://docs.cntd.ru/document/1200176640  (дата обращения: 05.04.2025).

Приложение А – Техническое задание

**1. Введение**

**1.1.** **Общие сведения**

Документ представляет собой техническое задание на создание веб-приложения «Цветы», которое позволит эффективно управлять каталогом цветов, оформлять заказы и обеспечит удобный интерфейс для покупателей.

**1.2. Цели и задачи**

Целью создания веб-приложения «Цветы» является автоматизация процессов управления каталогом и заказами, а также упрощение покупки цветов для клиентов. Задачи системы включают:

* авторизация и регистрация пользователей;
* размещение, редактирование и удаление товаров (цветов, букетов);
* поиск по названию, категории;
* просмотр детальной информации о товарах, включая фотографии и описание;
* оформление заказов;
* управление корзиной покупок.
* модерация товаров и пользователей администраторами.

**2. Основания для разработки**

**2.1.** **Нормативные документы**

Документ основывается на следующих нормативных документах:

* ГОСТ 34.602-2020 "Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы".
* ГОСТ Р 56477-2015 "Проектирование и внедрение информационных систем. Общие требования".

**3. Назначение системы**

**3.1. Общее описание**

Веб-приложение «Цветы» предназначено для использования цветочными магазинами и их клиентами. Веб-приложение будет поддерживать процессы управления каталогом, оформления заказов и взаимодействия с клиентами.

**3.2.** **Преимущества и новизна**

Веб-приложение «Цветы» будет предоставлять:

* Интуитивно понятный интерфейс.
* Механизмы для автоматической генерации отчетов.

**4. Требования к системе**

**4.1.** **Функциональные требования**

1. Авторизация и регистрация пользователей:
   * Разделение ролей: администратор, покупатель.
2. Управление каталогом:
   * Размещение, редактирование и удаление товаров (цветов, букетов);
   * Загрузка фотографий и описания товаров;
   * Указание цены, характеристик.
3. Поиск и фильтрация товаров:
   * Поиск по названию, категории.
4. Просмотр товаров:
   * Детальная информация о каждом товаре, включая фотографии, описание, цену.
5. Оформление заказов:
   * Добавление товаров в корзину;
   * Указание адреса доставки и контактной информации.
6. Управление корзиной:
   * Добавление, удаление и изменение количества товаров в корзине.
   * Просмотр итоговой стоимости заказа.

**4.2.** **Технические требования**

1. Производительность:
   * Время отклика системы не более 2 секунд при загрузке данных.
2. Надежность:
   * Доступность системы не менее 99,5% в год.
3. Безопасность:
   * Аутентификация пользователей.

**4.3. Эксплуатационные требования**

1. Удобство использования:
   * Дружественный пользовательский интерфейс.

**5. Требования к техническому обеспечению**

**5.1. Оборудование**

* Сервер: Серверная платформа с процессором не менее 4 ядер, 8 ГБ ОЗУ, SSD объемом 100 МБ.
* Клиентские рабочие станции: ПК с ОС Windows 7, 4 ГБ ОЗУ, 100 МБ свободного места на диске.

**5.2. Сетевые требования**

* Сеть: Доступ в Интернет со скоростью не менее 10 Мбит/с.
* Сетевые протоколы: Поддержка TCP/IP, HTTPS.

**6. Требования к программному обеспечению**

**6.1. Программные компоненты**

* Операционная система: Серверная версия Windows Server.
* Базы данных: PostgreSQL
* Программное обеспечение: Веб-сервер Nginx, интерпретатор PHP.

**6.2. Интерфейсы**

Интерфейс пользователя: Веб-интерфейс с поддержкой браузеров Chrome 117.x, Яндекс 23.9.x, Edge 117.x