МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Нереляционные базы данных»

Тема: Онлайн-магазин цветов и комнатных растений

Студентка гр. 0382 _____ Охотникова Г.С.

Студентка гр. 0382 _____ Михайлова О.Д.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2024

ЗАДАНИЕ

Студентка Охотникова Г.С.

Студентка Михайлова О.Д.

Группа 0382

Тема работы: Онлайн-магазин цветов и комнатных растений

Исходные данные:

Разработать Онлайн-магазин цветов и комнатных растений: продавцы выставляют товар, покупатели заказывают.

Содержание пояснительной записки:

1. «Содержание»
2. «Введение»
3. «Сценарий использования»
4. «Модель данных»
5. «Разработка приложения»
6. «Вывод»
7. «Будущее развитие решения»
8. «Приложения»

Предполагаемый объем пояснительной записки:

Не менее 30 страниц.

Дата выдачи задания: 26.09.2023

Дата сдачи реферата: 28.01.2024

Дата защиты реферата: 28.02.2024

Студентка гр. 0382 Охотникова Г.С.

Студент гр. 0382 _____ Михайлова О.Д.

Преподаватель _____ Заславский М.М.

АННОТАЦИЯ

Разработано приложение, являющееся онлайн-магазинов для продажи/покупки цветов и комнатных растений. Реализован просмотр каталога и отдельно карточки товара, добавления товаров в корзину, осуществление заказа. Есть склад и возможность добавлять в него товары. Для хранения данных используется neo4j.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Постановка задачи

Реализовать веб-приложение, представляющее собой интернет-магазин цветов с возможностями интерфейса как для покупателей, так и продавцов.

1.2. Предлагаемое решение

Разработка разделена на клиентскую и серверную части: клиент на Vue.js, сервер на Flask. Будет использована СУБД neo4j.

1.3. Качественные требования к решению

Разработать web-приложение, предоставляющее пользователю практичный интерфейс для покупки и продажи цветов.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2. 1. Макет UI

Рисунок 1 — Макет панели администрирования

2. 2. Сценарии использования для задачи

Сценарий использования - "Вход в аккаунт покупателя"

Действующее лицо: Пользователь

Основной сценарий:

1) Пользователь заходит на главную страницу.

2) Пользователь кликает на кнопку "Войти" и переходит на страницу входа в личный кабинет.

3) Пользователь вводит свою почту и пароль, кликает на кнопку "Войти" и переходит на страницу профиля покупателя.

Результат: Выполнен вход в аккаунт покупателя.

Альтернативный сценарий:

1) У пользователя нет аккаунта.

Сценарий использования - "Регистрация на сайте"

Действующее лицо: Пользователь

Основной сценарий:

1) Пользователь заходит на главную страницу.

2) Пользователь кликает на кнопку "Войти" и переходит на страницу входа в личный кабинет.

3) Пользователь кликает на ссылку "Зарегистрироваться" и переходит на страницу регистрации.

4) Пользователь вводит свою почту и пароль (2 раза), кликает на кнопку "Зарегистрироваться" и переходит на страницу профиля покупателя.

Результат: Создан аккаунт пользователя.

Альтернативный сценарий:

1) У пользователя уже есть аккаунта.

2) Пользователь ввел почту неверного формата.

3) Пользователь неверно ввел повторный пароль при регистрации.

Сценарий использования - "Заказ цветов"

Действующее лицо: Пользователь

Основной сценарий:

1) Пользователь выполняет сценарий "Вход в аккаунт покупателя".

2) Пользователь кликает на вкладку "Каталог" и переходит на страницу каталога.

3) Пользователь выбирает понравившийся букет, кликает на него и переходит на страницу с информацией о букете.

4) Пользователь кликает на кнопку "Добавить в корзину".

5) Пользователь кликает на значок корзины и переходит на страницу корзины.

6) Пользователь вводит данные о доставке и кликает на кнопку "Оплатить".

Результат: Оформлен заказ пользователя.

Альтернативный сценарий:

1) Пользователь добавляет букет в "Избранное".

2) Выбранный товар отсутствует на складе.

Сценарий использования - "Вход в роли продавца/администратора"

Действующее лицо: Пользователь-продавец/пользователь-администратор

Основной сценарий:

1) Пользователь заходит на главную страницу.

2) Пользователь кликает на значок профиля и переходит на страницу профиля или выполняет сценарий "Вход в аккаунт покупателя".

3) Пользователь кликает на ссылку "продавец"/"админ" и переходит на страницу профиля продавца/администратора.

Результат: Выполнен вход пользователя в роли продавца/администратора.

Альтернативный сценарий:

1) Пользователь не является администратором.

Сценарий использования - "Добавление товаров на склад"

Действующее лицо: Пользователь-продавец

Основной сценарий:

1) Пользователь выполняет сценарий "Вход в роли продавца".

2) Пользователь кликает на ссылку "Склад" и переходит на страницу склада.

3) Пользователь кликает на кнопку "Добавить товар" и переходит на страницу добавления товара.

4) Пользователь вводит данные о товаре, кликает на кнопку "Добавить товар" и переходит на страницу склада.

Результат: Добавлен новый товар на склад.

Сценарий использования - "Просмотр статистики"

Действующее лицо: Пользователь-продавец/пользователь-администратор

Основной сценарий:

1) Пользователь выполняет сценарий "Вход в роли продавца/администратора".

3) Пользователь кликает на ссылку "Статистика" и переходит на страницу со статистикой прибыли.

4) При желании пользователь может кликнуть на ссылку "Эффективность работы склада"/"Эффективность работы сервиса" и перейти на страницу со статистикой эффективности.

Результат: Пользователь получает информацию о статистике.

Сценарий использования - "Просмотр каталога/контактов/информации о компании"

Действующее лицо: Пользователь

Основной сценарий:

1) Пользователь заходит на главную страницу.

2) Пользователь кликает на вкладку "Каталог"/"Контакты"/"О компании" и переходит на интересующую страницу.

Результат: Пользователь получает интересующую информацию.

Сценарий использования - "Просмотр истории заказов или избранных товаров"

Действующее лицо: Пользователь

Основной сценарий:

1) Пользователь заходит на главную страницу.

2) Пользователь кликает на значок профиля и переходит на страницу профиля или выполняет сценарий "Вход в аккаунт покупателя".

3) Пользователь кликает на ссылку "Заказы"/"Избранные" и переходит на интересующую страницу.

Результат: Пользователь получает интересующую информацию.

Сценарий использования - "Выход из аккаунта / удаление аккаунта"

Действующее лицо: Пользователь

Основной сценарий:

1) Пользователь заходит на главную страницу.

2) Пользователь кликает на значок профиля и переходит на страницу профиля или выполняет сценарий "Вход в аккаунт покупателя".

3) Пользователь кликает на ссылку "Выйти" / "Удалить аккаунт".

Результат: Пользователь вышел из аккаунта или удалил аккаунт.

Сценарий использования - "Обращение в поддержку"

1) Пользователь заходит на главную страницу.

2) Пользователь кликает на значок профиля и переходит на страницу профиля или выполняет сценарий "Вход в аккаунт покупателя".

3) Пользователь кликает на ссылку "Поддержка" и переходит на страницу поддержки.

4) Пользователь описывает проблему в текстовом поле, кликает на кнопку "Отправить", и видит на экране сообщение о подтверждении отправки сообщения.

Результат: Пользователь сообщил о проблеме в поддержку.

Альтернативный сценарий:

1) Возникла ошибка при отправке сообщения.

3. МОДЕЛЬ ДАННЫХ

3. 1. Нереляционная модель данных

1) Графическое представление

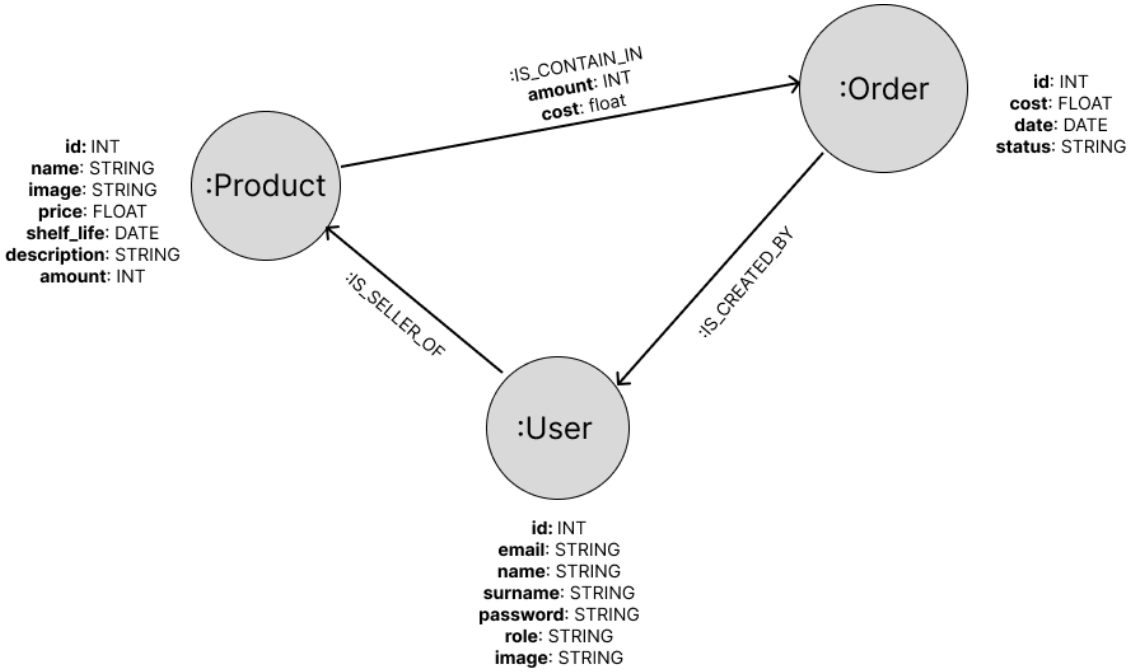

Рисунок 2 — Схема для нереляционной модели БД

2) Описание назначений коллекций, типов данных и сущностей

Все сущности и отношения имеют одну из меток:

:User - пользователь, зарегистрированный на сайте магазина. Содержит следующие атрибуты:

- **id** - уникальный идентификатор. Тип INT, размер 4b.
- **email** - email пользователя. Тип STRING, размер 255b.
- **name** - имя пользователя. Тип STRING, размер 255b.
- **surname** - фамилия пользователя. Тип STRING, размер 255b.
- **password** - пароль пользователя. Тип STRING, размер 255b.
- **role** - максимальная роль, которая есть у пользователя (покупатель, продавец или администратор). При регистрации для каждого пользователя устанавливается роль покупателя, далее он может ее изменить на покупателя. Покупателю доступны только функции покупателя, продавцу - функции

покупателя и продавца, администратору - функции покупателя, продавца и администратора. Тип STRING, размер 26b.

• **image** - ссылка на изображение, если пользователь добавил фото. Тип STRING, размер до 2000b.

:Product - товар. Содержит следующие атрибуты:

• **id** - уникальный идентификатор. Тип INT, размер 4b.

• **name** - наименование товара. Тип STRING, размер 255b.

• **image** - ссылка на изображение товара. Тип STRING, размер до 2000b.

• **price** - цена товара. Тип FLOAT, размер 4b.

• **shelf_life** - дата, когда заканчивается срок годности товара. Тип DATE, размер 3b.

• **description** - описание товара. Тип STRING, размер 255b.

• **amount** - количество данного товара. Тип INT, размер 4b.

:Order - заказ. Содержит следующие атрибуты:

• **id** - уникальный идентификатор. Тип INT, размер 4b.

• **cost** - общая стоимость товара. Тип FLOAT, размер 4b.

• **date** - дата оформления заказа (дата его последнего изменения). Тип DATE, размер 3b.

• **status** - статус заказа (оформляется или оплачен). Тип STRING, размер 22b.

:IS_CONTAIN_IN - товар **содержится** в заказе. Атрибуты:

• **amount** - количество данного товара в данном заказе. Тип INT, размер 4b.

• **cost** - стоимость данного товара в данном заказе с учетом его количества. Тип FLOAT, размер 4b.

:IS_CREATED_BY - заказ **создан** пользователем.

:IS_SELLER_OF - пользователь **является продавцом** товара.

3) Оценка удельного объема информации

Объем памяти для хранения информации об одном пользователе: 3050b.

Объем памяти для хранения информации одного товара: 2525b.

Объем памяти для хранения информации одного заказа: 33b.

Объем памяти для хранения одной связи IS_CONTAIN_IN: 8b.

Пусть в моделе хранится информация о N пользователях. Каждый пользователь является покупателем и имеет в среднем 4 заказа с 2 товарами каждый. Треть пользователей является продавцами и в среднем продают 3 разных товара. Тогда получим общий общий объем:

$V = 3050 * N + (33 + 8 * 2) * 4 * N + 2525 * 3 * (N/3) = 5771*N$ b.

Избыточность модели:

Для вычисления чистого объема данных необходимо убрать атрибут id из сущностей пользователя и товара:

Vч = (3050 - 4) * N + (33 + 8 * 2) * 4 * N + (2525 - 4) * 3 * (N/3) = 5763*N b.

Отношение между фактическим объемом модели и “чистым” объемом данных:

V\Vч = 5771N\5763N = 1,001

Направление роста модели при увеличении количества объектов каждой сущности**:**

При увеличении количества объектов каждой сущности модель будет расти линейно.

4) Запросы к модели, с помощью которых реализуются сценарии использования

Вход в аккаунт

MATCH (user:User {email: 'введенный_email', password: 'введенный_пароль'})

RETURN user;

Кол-во запросов: 1

Регистрация на сайте

CREATE (user:User {id: 'введенный_id', email: 'введенный_email', name: 'введенное_имя', surname: 'введенная_фамилия', password: 'введенный_пароль', role: 'Customer', image: 'ссылка_на_изображение'});

Кол-во запросов: 1

Заказ цветов (создание нового заказа)

MATCH (m:User {id: 'введенный_id'}), (p:Product {id: 'введенный_id'})

//создаем новый заказ или обновляем, если заказ уже создан

MERGE (o:Order {status: 'Оформляется'})-[:IS_CREATED_BY]->(m)

ON CREATE SET o.id = 'введенный_id', o.cost = 0, o.date = 'текущая_дата'

ON MATCH SET o.date = 'текущая_дата'

//добавляем новые продукты, или увеличиваем количество существующего

MERGE (p)-[r:IS_CONTAINED_IN]->(o)

ON CREATE SET r.amount = 1, r.cost = p.price

*ON MATCH SET r.amount = r.amount + 1, r.cost = r.amount * p.price;*

//обновляем стоимость всего заказа

MATCH (o:Order {id: 'введенный_id'})-[:IS_CREATED_BY]->(m), (n)-[r:IS_CONTAINED_IN]->(o)

WITH o, sum(r.cost) AS sum_cost

SET o.cost = sum_cost;

//после оплаты обновляем статус заказа

MATCH (o:Order {id: 'введенный_id'}) SET o.status = 'Оплачен';

//уменьшаем количество товара на складе

MATCH (p:Product {id: 'введенный_id'})-[r:IS_CONTAINED_IN]->(o:Order {id: 'введенный_id'})

SET p.amount = p.amount - r.amount;

Кол-во запросов: 2N + 1 + 1 = 2N+2, где N - количество товаров в заказе

Вход в роли продавца/администратора

//обновляем максимальную роль пользователя

MATCH (n:User {id: 'введенный_id'})

SET n.role = 'продавец'/'администратор';

Кол-во запросов: 1

Добавление товара на склад

MATCH (n:User {id: 'id_текущего_пользователя'})

CREATE (n)-[:IS_SELLER_OF]->(product:Product {id: 'введенный_id', name: 'введенное_название', image: 'ссылка_на_изображение', price: 'введенная_цена', shelf_life: 'введенный_срок_годности', description: 'введенное_описание', amount: 'введенное_количество'});

Кол-во запросов: N, где N - количество различных добавляемых товаров

Просмотр статистики (например, суммарная прибыль)

MATCH (n:User {id: 'id_текущего_пользователя'})-[:IS_SELLER_OF]->(p:Product), (p)-[r:IS_CONTAIN_IN]->(m)

RETURN SUM(r.cost);

Кол-во запросов: 1

Просмотр каталога

MATCH (p:Product) WHERE p.amount > 0 RETURN p;

Кол-во запросов: 1

Просмотр истории заказов

MATCH (o:Order)-[:IS_CREATED_BY]->(n:User {id: 'id_текущего_пользователя'}) RETURN o;

Кол-во запросов: 1

Удаление пользователя

MATCH (m)-[r]->(n:User {id: 'id_текущего_пользователя'}), (n)-[p]->(k) DELETE n, r, p

Кол-во запросов: 1

3.2. Реляционная модель данных

1) Графическое представление

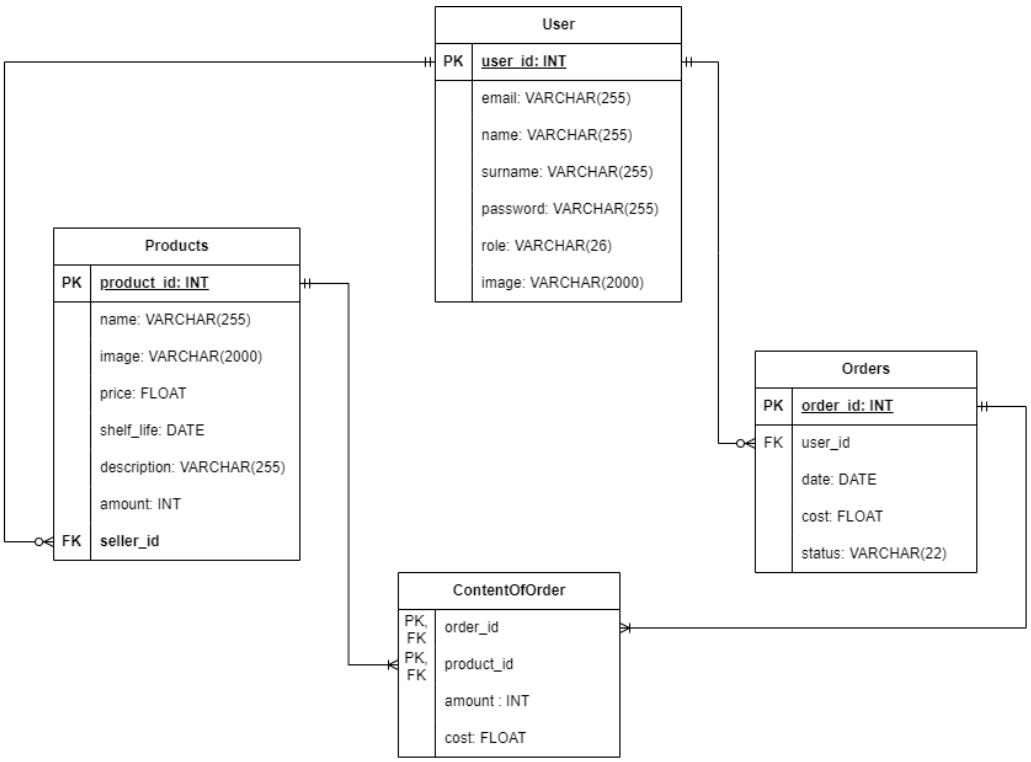

Рисунок 3 — Схема для реляционной модели БД

2) Описание назначений коллекций, типов данных и сущностей

Таблица **User** содержит данные о пользователях:

- **user_id** - уникальный идентификатор. Тип INT, размер 4b.
- **email** - email пользователя. Тип VARCHAR, размер 255b.
- **name** - имя пользователя. Тип VARCHAR, размер 255b.
- **surname** - фамилия пользователя. Тип VARCHAR, размер 255b.
- **password** - пароль пользователя. Тип VARCHAR, размер 255b.
- **role** - максимальная роль, которая есть у пользователя (покупатель, продавец или администратор). При регистрации для каждого пользователя устанавливается роль покупателя, далее он может ее изменить на покупателя. Покупателю доступны только функции покупателя, продавцу - функции покупателя и продавца, администратору - функции покупателя, продавца и администратора. Тип VARCHAR, размер 26b.
- **image** - ссылка на изображение, если пользователь добавил фото. Тип VARCHAR, размер до 2000b.

Таблица **Products** содержит данные о товарах:

- **product_id** - уникальный идентификатор. Тип INT, размер 4b.
- **name** - наименование товара. Тип VARCHAR, размер 255b.
- **image** - ссылка на изображение товара. Тип VARCHAR, размер до 2000b.
- **price** - цена товара. Тип FLOAT, размер 4b.
- **shelf_life** - дата, когда заканчивается срок годности товара. Тип DATE, размер 3b.
- **description** - описание товара. Тип VARCHAR, размер 255b.
- **amount** - количество данного товара. Тип INT, размер 4b.
- **seller_id** - id пользователя, который является продавцом данного товара. Тип INT, размер 4b.

Таблица **Orders** - содержит информацию о заказах:

- **order_id** - уникальный идентификатор. Тип INT, размер 4b.
- **user_id** - id пользователя, который оформляет заказ. Тип INT, размер 4b.
- **cost** - общая стоимость товара. Тип FLOAT, размер 4b.
- **date** - дата оформления заказа (дата его последнего изменения). Тип DATE, размер 3b.
- **status** - статус заказа (оформляется или оплачен). Тип VARCHAR, размер 22b.

Таблица **ContentOfOrder** - хранит информацию о содержании заказов:

- **order_id** - id заказа. Тип INT, размер 4b.
- **product_id** - id товара. Тип INT, размер 4b.
- **amount** - количество данного товара в данном заказе. Тип INT, размер 4b.

• **cost** - стоимость данного товара в данном заказе с учетом количества. Тип FLOAT, размер 4b.

3) Оценка удельного объема информации

Объем памяти для хранения информации об одном пользователе: 3050b.

Объем памяти для хранения информации одного товара: 2529b.

Объем памяти для хранения информации одного заказа: 37b.

Объем памяти для хранения содержания одного заказа с одним товаром: 16b.

Пусть в моделе хранится информация о N пользователях. Каждый пользователь является покупателем и имеет в среднем 4 заказа с 2 товарами каждый. Треть пользователей является продавцами и в среднем продают 3 разных товара. Тогда получим общий общий объем:

V = 3050 * N + (37 + 16 * 2) * 4 * N + 2529 * 3 * (N/3) = 5855*N b.

Избыточность модели:

Для вычисления чистого объема данных необходимо убрать поле id из таблиц User и Products, а также все внешние ключи:

Vч = (3050 - 4) * N + (37 - 4 + (16 - 8) * 2) * 4 * N + (2529 - 8) * 3 * (N/3) = 5763*N b.

Отношение между фактическим объемом модели и “чистым” объемом данных:

V\Vч = 5855N\5763N = 1,016

Направление роста модели при увеличении количества объектов каждой сущности:

При увеличении количества объектов каждой сущности модель будет расти линейно.

4) Запросы для выполнения сценариев

Вход в аккаунт

SELECT name, surname, image, role FROM User WHERE email = 'введенный_email' AND password = 'введенный_пароль';

Кол-во запросов: 1

Регистрация на сайте

INSERT INTO User(email, name, surname, password, image)

VALUES ('введенный_email', 'введенное_имя', 'введенная_фамилия', 'введенный_пароль', 'ссылка_на_изображение');

Кол-во запросов: 1

Заказ цветов (создание нового заказа)

//добавляем в таблицу заказов новый заказ

INSERT INTO Orders(user_id, date)

VALUES ('id_текущего_пользователя', 'текущая_дата')

//добавляем в таблицу содержания заказа выбранный продукт, если он до этого не был добавлен, или обновляем значение в таблице, если этот продукт уже добавлен

INSERT INTO ContentOfOrder(order_id, product_id, amount)

VALUES ('id_текущего_заказа', 'id_заказываемого_продукта', 1)

ON DUPLICATE KEY UPDATE amount = amount + 1;

//обновляем стоимость для добавленного продукта

UPDATE ContentOfOrder

SET cost = (

//высчитываем стоимость для добавленного в заказ продукта с учетом его количества

*SELECT price * ContentOfOrder.amount*

```
    FROM ContentOfOrder INNER JOIN Products
    ON ContentOfOrder.product_id = Products.product_id
    WHERE Products.product_id = 'id_заказываемого_продукта' AND ContentOfOrder.order_id = 'id_текущего_заказа'
  )

  //обновляем стоимость всего заказа
  UPDATE Orders
  SET cost = (
    //высчитываем суммарную стоимость всех добавленных в заказ продуктов
    SELECT SUM(cost)
    FROM ContentOfOrder
    WHERE ContentOfOrder.order_id = 'id_текущего_заказа'
  )
  WHERE order_id = 'id_текущего_заказа';

  //после оплаты обновляем статус заказа
  UPDATE
  SET status = 'оплачен'
  WHERE order_id = 'id_текущего_заказа';

  //уменьшаем количество товара на складе
  UPDATE Products
  SET amount = amount - (
    SELECT amount FROM ContentOfOrder WHERE product_id = 'id_заказываемого_продукта' AND order_id = 'id_текущего_заказа';
  )
  WHERE product_id = 'id_заказываемого_продукта';
```

Кол-во запросов: 1 + 3*N + 1 + 1 = 3N + 3, где N - количество добавленных товаров

Вход в роли продавца/администратора

//обновляем максимальную роль пользователя

UPDATE User

SET role = 'продавец'/'администратор'

WHERE user_id = 'id_текущего_пользователя'

Кол-во запросов: 1

Добавление товара на склад

INSERT INTO Products(name, image, price, shelf_life, decription, amount, seller_id)

VALUES ('введенное_название', 'ссылка_на_изображение', 'введенная_цена', 'введенный_срок_годности', 'введенное_описание', 'введенное_количество', 'id_текущего_пользователя');

Кол-во запросов: 1

Просмотр статистики (например, суммарная прибыль)

SELECT SUM(ContentOfOrder.cost)

FROM ContentOfOrder INNER JOIN Products

ON ContentOfOrder.product_id = Products.product_id

WHERE Products.seller_id = 'id_текущего_продавца'

Кол-во запросов: 1

Просмотр каталога

*SELECT * FROM Products*

WHERE amount > 0;

Кол-во запросов: 1

Просмотр истории заказов

SELECT order_id, date, cost, status FROM Orders WHERE user_id = 'id_текущего_пользователя';

Кол-во запросов: 1

Удаление пользователя

DELETE FROM User

WHERE user_id = 'id_текущего_пользователя';

Кол-во запросов: 1

3.3. Сравнение моделей

При одинаковой хранимой информации у нереляционной БД данные занимают меньший объем, чем у реляционной.

Кол-во запросов у реляционной и нереляционной БД отличаются только для сценария 3. Заказ цветов (создание нового заказа). При добавлении в заказ N товаров:

- для нереляционной БД потребуется выполнить 2N+2 запросов
- для реляционной БД потребуется вполнить 3N+3 запросов

Для остальных сценариев количество запросов одинаковое для обеих БД. Следовательно, при одинаковых данных для нереляционной БД потребуется выполнить меньше запросов.

3.4. Вывод

Лучше использовать нереляционную БД.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

В ходе выполнения работы было реализована база данных, клиентская и серверная части веб-приложения:

- Серверная часть реализована с помощью фреймоворка Flask: получает запросы от клиента, делает запрос к базе данных и возвращает требуемую информацию.
- Клиентская часть реализована на Vue.js: осуществляет запросы к серверу и отображает информацию.
- База данных — синтетический набор данных, а также команды для взаимодействия.

4.2. Использованные технологии

- Сервер: Fkask
- Клиент: Vue, Pinia, Vite, Vue-router
- База данных: Neo4j
- Развёртывание приложения: Docker Compose

4.3. Снимки экрана приложения

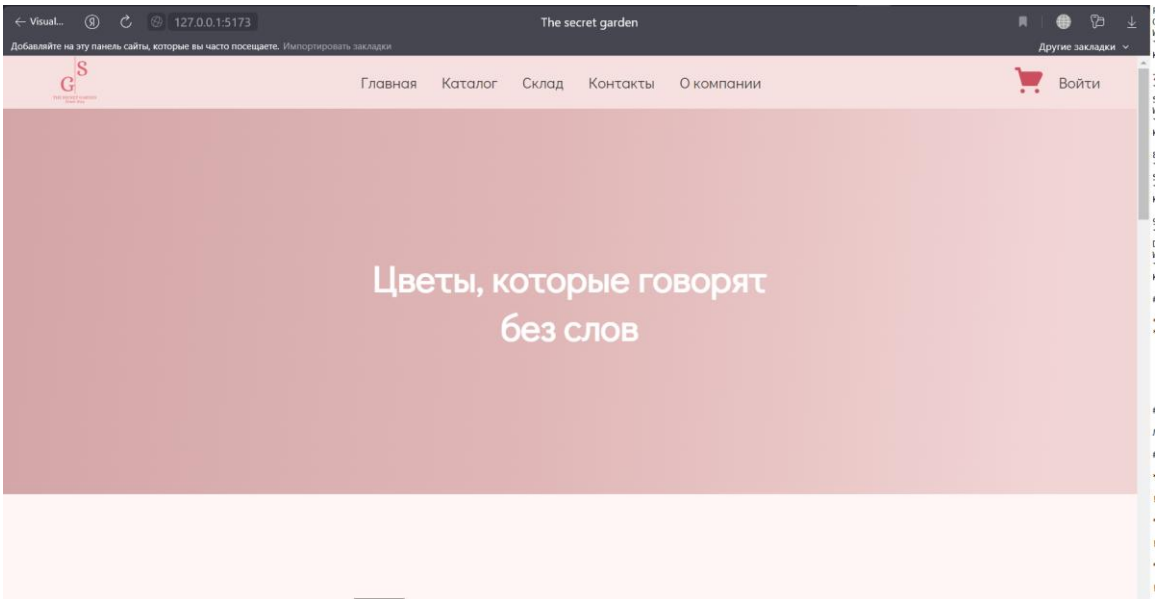

Рисунок 1 - Главная страница

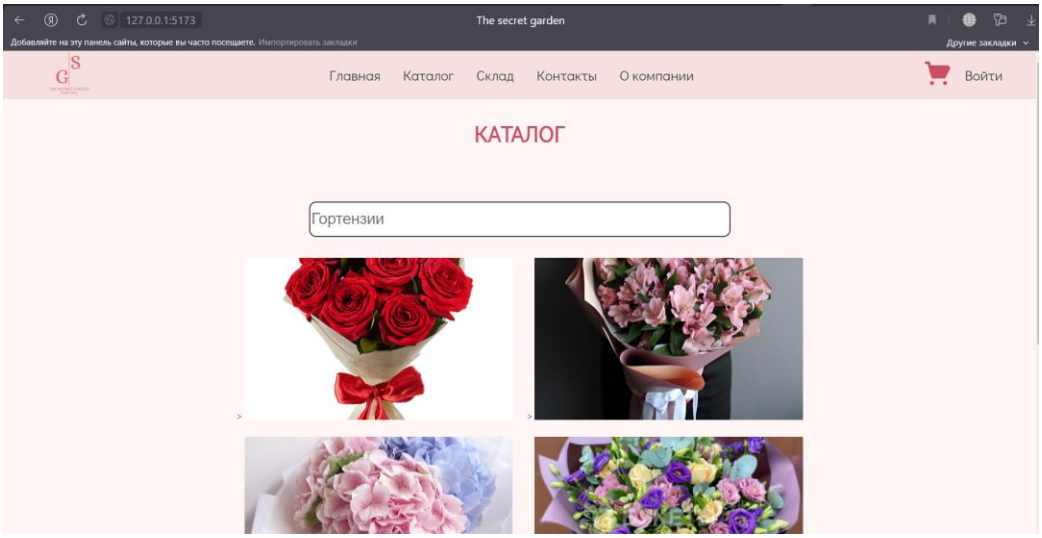

Рисунок 2 - Страница с каталогом

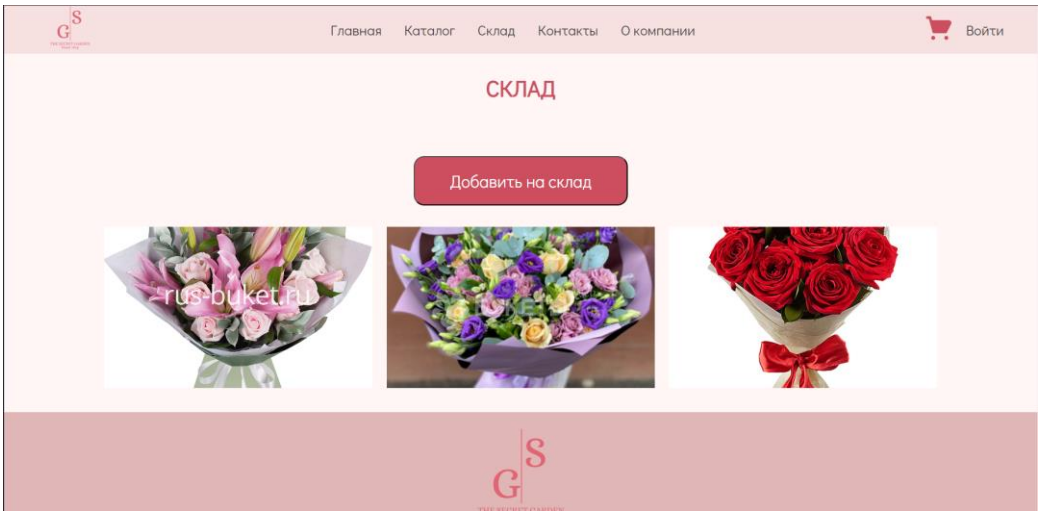

Рисунок 3 - Страница со складом

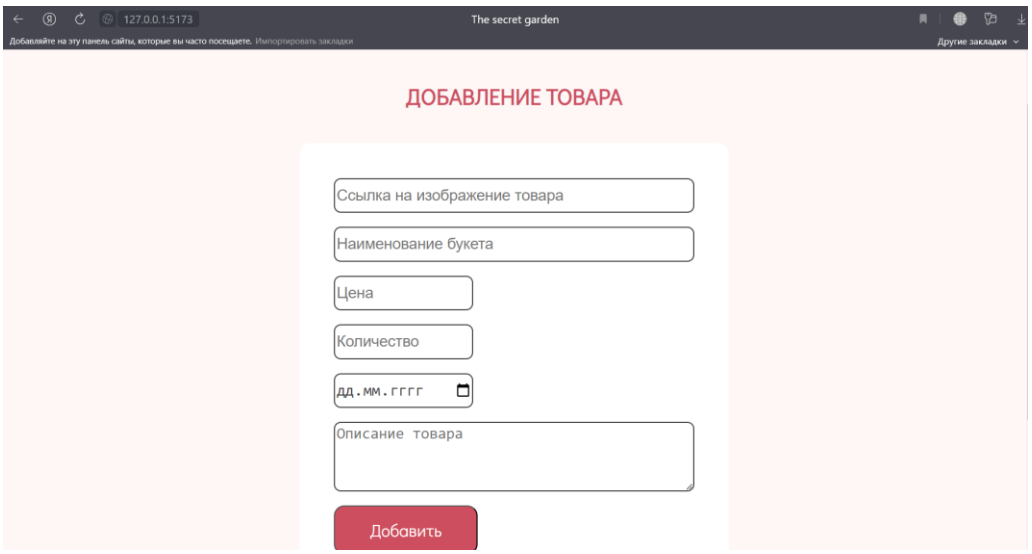

Рисунок 4 - Страница добавления товара на склад

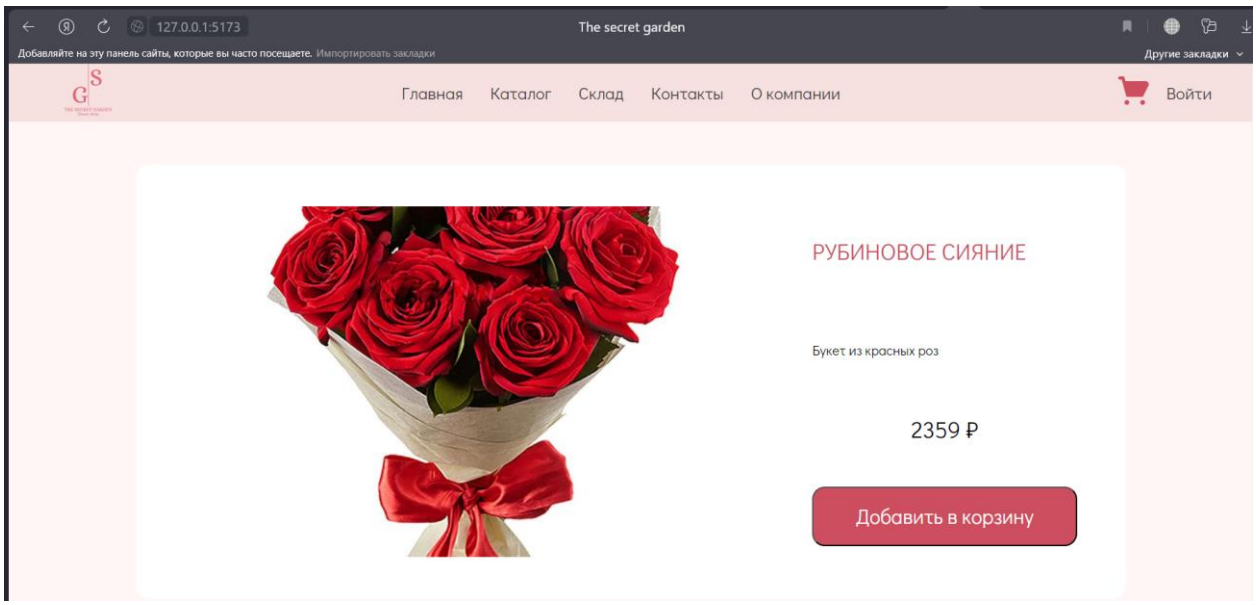

Рисунок 5 - Страница одного товара

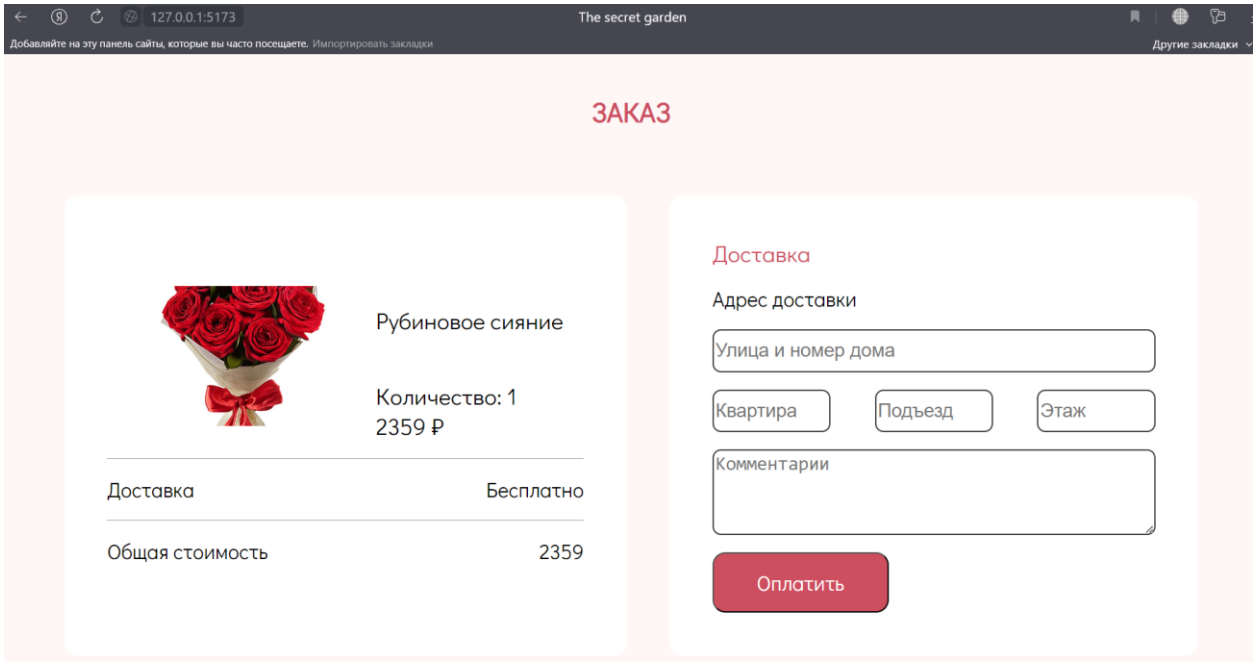

Рисунок 6 - Страница корзины

5. ВЫВОДЫ

5.1. Достигнутые результаты

В ходе работы было разработано веб-приложение Онлайн-магазин цветов и комнатных растений, реализован просмотр каталога, карточки товара, добавление в корзину, склад и добавление товара на склад.

5.2. Недостатки и пути для улучшения полученного решения

У приложения ограниченный функционал: нет авторизации, нет профиля пользователя и ролей. Так же нет статистики и возможности загружать изображение не ссылкой.

6. БУДУЩЕЕ РАЗВИТИЕ РЕШЕНИЯ

Реализовать авторизацию, сделать раздел статистики, добавить профиль пользователя и смену ролей, добавить возможность общаться с поддержкой.

7. ЛИТЕРАТУРА

1. GitHub-репозиторий: https://github.com/moevm/nosql2h23-flowers-shop
2. Документация Flask: https://flask.palletsprojects.com/en/3.0.x/
3. Документация Vue: https://vuejs.org/
5. Документация Docker: https://docs.docker.com/
6. Документация Neo4j: https://neo4j.com/