Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «**Информатика и системы управления**»

КАФЕДРА «**Программное обеспечение ЭВМ и информационные технологии**»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Разработка базы данных для хранения и обработки данных цветочного магазина»

Студент ИУ7-65Б _____ **А.А. Саркисян**
(Группа) (Подпись, дата) (И.О. Фамилия)

Руководитель курсовой работы _____ **А.С. Кострицкий**
(Подпись, дата) (И.О. Фамилия)

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка 48 с., 12 рис., 4 табл., 36 источн., 1 прил.

ЦВЕТОЧНЫЙ МАГАЗИН, БАЗА ДАННЫХ, ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ, ВЕБ-САЙТ

Цель работы — разработка базы данных для хранения и обработки данных цветочного магазина.

Результатом работы являются разработанная база данных, веб-сайт для взаимодействия с базой данных. Веб-сайт предоставляет возможности просмотра, изменения, добавления и удаления данных в зависимости от роли пользователя. В качестве системы управления базами данных используется PostgreSQL.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цветочный магазин — это небольшая коммерческая организация, продающая цветы. Необходимость создания информационной системы для цветочного магазина вызвана большим количеством продукции фирмы и услугами [1].

Целью работы является разработка базы данных для хранения и обработки данных цветочного магазина.

Для достижения поставленной цели необходимо выполнить следующие задачи:

— сформулировать требования и ограничения к разрабатываемой базе данных и веб-сайту цветочного магазина;
— сформулировать описание пользователей проектируемого приложения для доступа к базе данных;
— спроектировать сущности базы данных и ограничения целостности цветочного магазина;
— спроектировать триггер обновления статуса букета при отсутствии его составных частей;
— разработать сущности базы данных цветочного магазина и реализовать ограничения целостности базы данных;
— реализовать веб-сайт, предоставляющий пользователям интерфейс доступа к базе данных;
— исследовать зависимость времени выполнения запросов от использования индексации при различных количествах записей в таблице.

1 Аналитический раздел

1.1 Сравнение существующих решений

Россия является одним из крупнейших потребителей срезанных цветов в мире, находясь на четвертом месте после США, Германии и Великобритании с затратами в 42 млрд. руб. ежегодно [2]. При этом ожидается, что во всем мире объем рынка интернет-магазинов цветов увеличится на 2,8% в 2023 году [3].

На данный момент в России существуют несколько крупных систем для заказа цветочных композиций онлайн, такие как Flor2U [4], Флорист.ру [5] и Flowwow [6].

В таблице 1 представлен сравнительный анализ известных решений.

Таблица 1 – Сравнение известных решений

Название	Возможность подписаться на рассылку цветов	Наличие истории заказов	Наличие информации об уходе за букетом
Flor2U	-	+	-
Флорист.ру	-	+	-
Flowwow	+	+	-

1.2 Формализация задачи и данных

В ходе выполнения курсовой работы необходимо спроектировать базу данных для хранения данных цветочного магазина, а также разработать веб-сайт, предоставляющий пользователям интерфейс для взаимодействия с базой

данных. Веб-сайт должен обеспечить пользователей следующими возможностями:

— просмотреть каталог товаров;
— просмотреть описание букета;
— редактировать корзину;
— оформить заказ.

Также веб-сайт должен обеспечить пользователей следующими возможностями, обусловленными их частичным отсутствием в аналогичных решениях:

— просмотреть информацию об уходе за букетами;
— подписаться на рассылку букетов;
— просмотреть историю заказов.

В соответствии со сформулированными требованиями для создания базы данных интернет-магазина цветов необходимо выделить следующие сущности:

— пользователь;
— заказ;
— цветок;
— букет;
— упаковка;
— корзина.

На рисунке 1 приведена ER-диаграмма сущностей в нотации Чена [7].

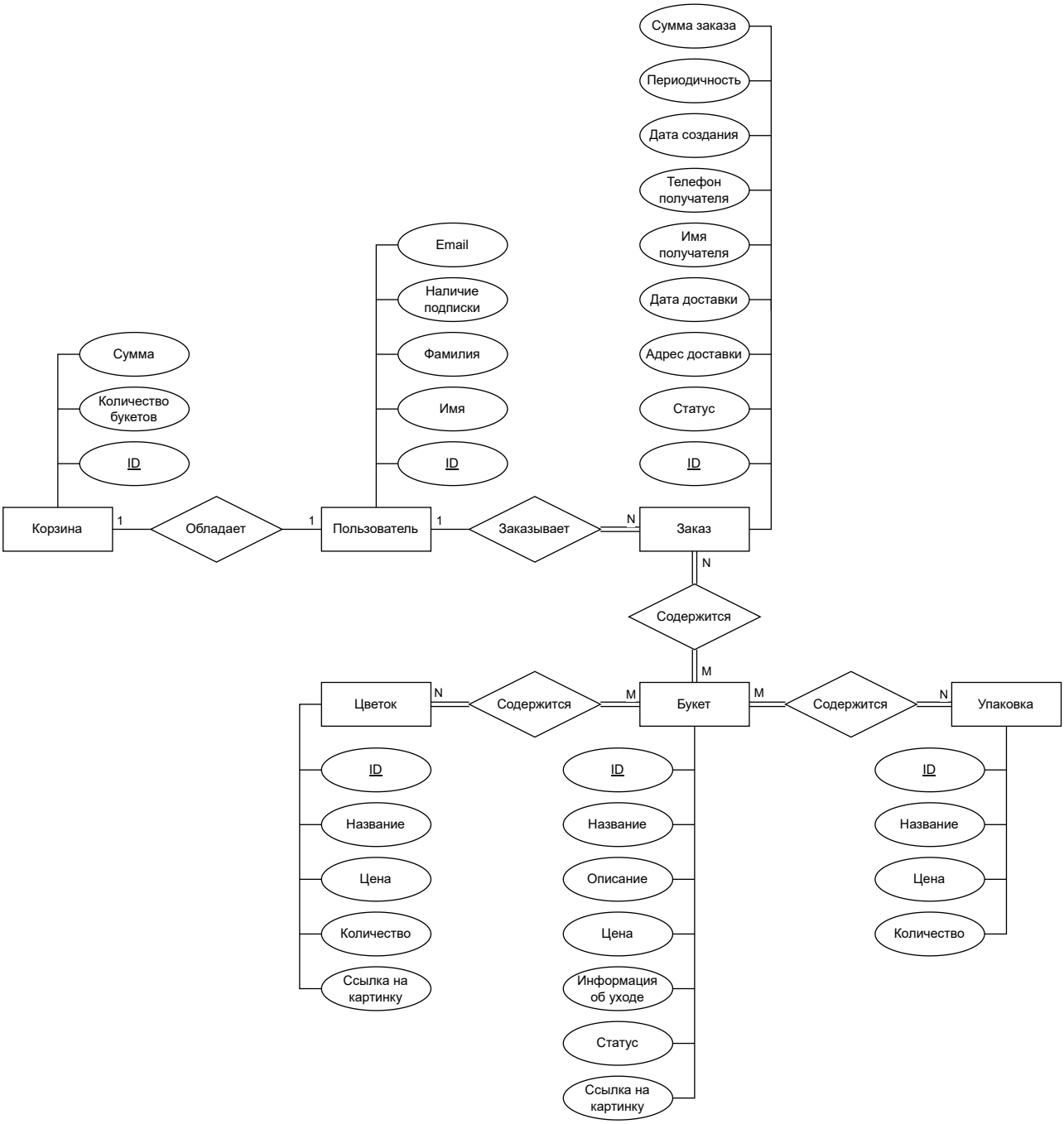

Рисунок 1 – ER-диаграмма сущностей

1.3 Ролевая модель разграничения доступа к базе данных

Для обеспечения безопасной работы с данными необходимо разработать политику управления доступом к данным в соответствии с ролевой моделью [8]. На ее основе определено, к каким объектам будет иметь доступ каждый субъект и какие действия может выполнять:

1) R — чтение;
2) W — запись/модификация;
3) C — создание;
4) D — удаление.

Пользователь интернет-магазина должен иметь возможность зарегистрироваться, а далее авторизоваться, т.е. получить группу прав на выполнение определенных действий. Соответственно, выделим следующие две роли: гость (незарегистрированный или неавторизованный пользователь) и авторизованный пользователь. Помимо этого, в системе должен присутствовать администратор, имеющий доступ ко всем данным. За учет товаров должен отвечать субъект, работающий со складом, следовательно, выделим такую роль, как продавец.

Диаграмма вариантов использования [9] представлена на рисунке 2 и политика управления доступом представлена в таблице 2.

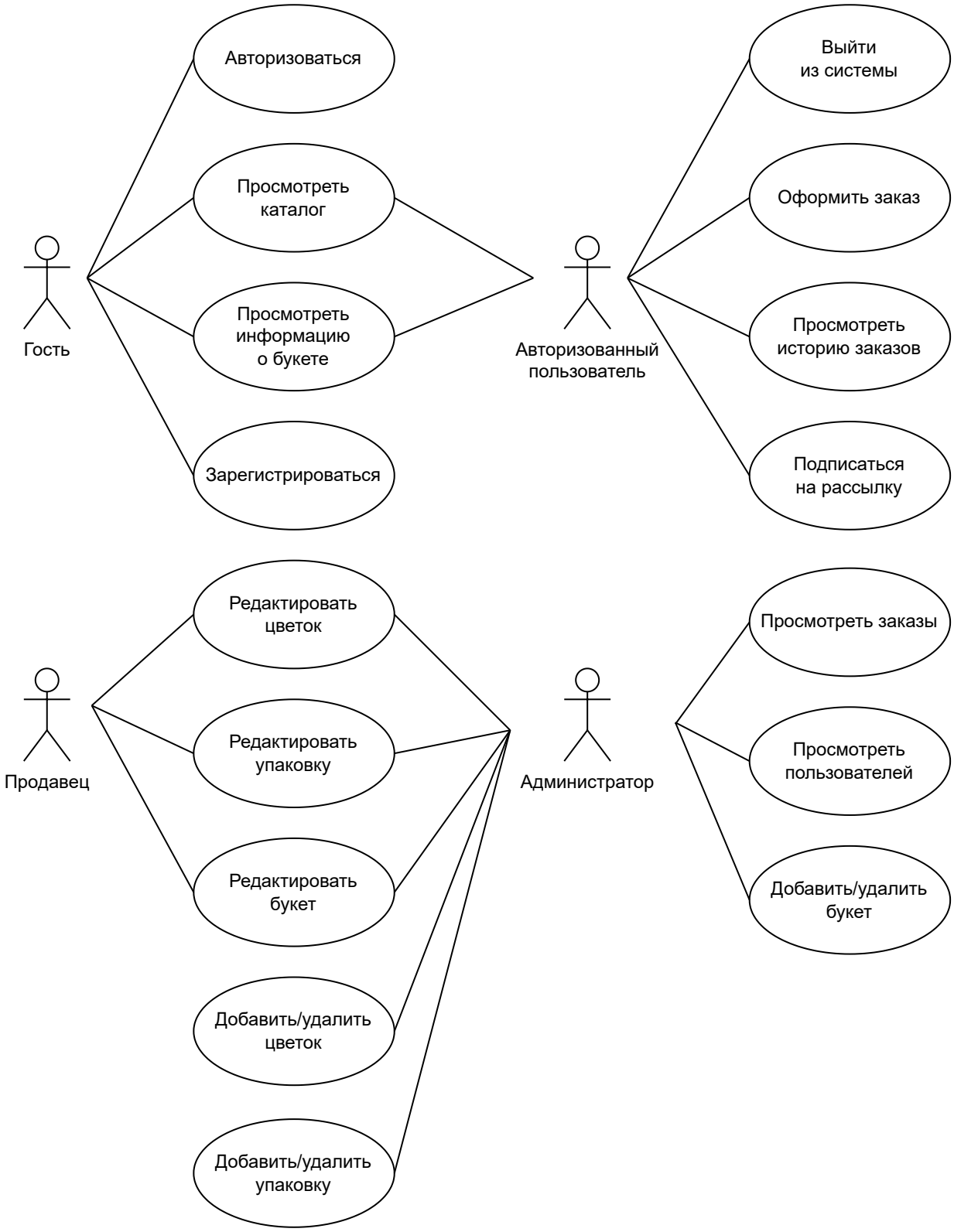

Рисунок 2 – Диаграмма вариантов использования

Таблица 2 – Политика управления доступом

Роль	Системные данные	Данные о заказах	Данные о товарах	Данные о пользователях
Гость			R	C
Зарегистрированный пользователь			R	
Продавец			RW	
Администратор	R	R	RWCD	R

1.4 Выбор модели данных

База данных (БД) — совокупность структурированных взаимосвязанных данных, относящихся к определённой предметной области и организованных таким образом, что эти данные могут быть использованы для решения многих задач многими пользователями [10].

Основой базы данных является модель данных — средство абстракции, позволяющее видеть обобщенную структуру данных, хранимых в базе данных, а не их конкретные значения [11].

Модели данных можно разделить на следующие категории [12]:

1) дореляционные;
2) реляционные;
3) постреляционные.

Дореляционные модели данных

В иерархической модели данные можно описать с помощью упорядочен-

ного графа (или дерева). К достоинствам иерархической модели данных относятся эффективное использование памяти и неплохие показатели времени выполнения основных операций над данными. Иерархическая модель данных удобна для работы с иерархически упорядоченной информацией. Недостатком иерархической модели является ее громоздкость для обработки информации с достаточно сложными логическими связями [11].

Сетевая модель данных позволяет отображать разнообразные взаимосвязи элементов данных в виде произвольного графа, обобщая тем самым иерархическую модель данных. В сравнении с иерархической моделью сетевая модель предоставляет большие возможности в смысле допустимости образования произвольных связей. Недостатком сетевой модели данных является высокая сложность и жесткость схемы базы данных, построенной на ее основе, а также в сетевой модели данных ослаблен контроль целостности связей [11].

Модель на основе инвертированных списков представляет собой совокупность файлов, содержащих записи. В такой модели данных отсутствуют ограничения целостности как таковые. Модель данных на основе инвертированных списков не подходит при постоянных изменениях, дополнениях и модификациях базы данных, так как наличие большого количества файлов может сильно замедлить процесс обработки информации [11].

Реляционная модель данных

Реляционная модель данных включает в себя следующие компоненты: структурный (данные в базе данных представляют собой набор отношений), целостностный (отношения отвечают определенным условиям целостности), манипуляционный (манипулирования отношениями осуществляется средствами реляционной алгебры и/или реляционного исчисления) [13].

Преимущества реляционной модели данных заключаются в простоте, понятности и удобстве физической реализации на ЭВМ, поддержке стандартизи-

рованного языка запросов SQL [14]. К недостаткам реляционной модели данных относятся медленный доступ к данным, трудоемкость разработки.

Постреляционная модель данных

Постреляционная модель является расширенной реляционной моделью, которая снимает ограничение неделимости хранящихся в записях таблиц данных [15]. Преимуществом постреляционной модели является возможность представления совокупности связанных реляционных таблиц одной постреляционной таблицей. Недостатками постреляционной модели являются сложность решения проблемы обеспечения целостности и непротиворечивости хранимых данных и отсутствие поддержки SQL [11].

Вывод

В разрабатываемой базе данных требуется поддержка ограничений целостности данных, отношений между сущностями, а также необходимо выполнение сложных запросов. Ввиду наличия опыта работы с SQL и в соответствии с перечисленными требованиями к базе данных необходимо выбрать реляционную модель.

1.5 Выводы к аналитическому разделу

В данном разделе проведен анализ предметной области цветочных магазинов при помощи сравнения известных решений. На его основе формализована задача и описаны требования к веб-сайту, предоставляющему пользователям интерфейс для взаимодействия с базой данных. Для обеспечения безопасной работы с данными разработана политика управления доступом к данным.

Проведена формализация данных и классификация моделей данных, по результатам их сравнения выбрана реляционная модель данных.

2 Конструкторский раздел

2.1 Проектирование базы данных

В соответствии с ER-диаграммой сущностей, изображенной на рисунке 1, база данных должна содержать следующие таблицы (отношения):

— actor: таблица пользователей;

— cart: таблица корзин;

— order: таблица заказов;

— bouquet: таблица букетов;

— flower: таблица цветов;

— packaging: таблица упаковок;

— cart_bouquet: таблица букетов в корзине;

— order_bouquet: таблица букетов в заказе;

— bouquet_flower: таблица цветов в букете;

— bouquet_packaging: таблица упаковок в букете;

На рисунке 3 представлена диаграмма базы данных в нотации Мартина [16].

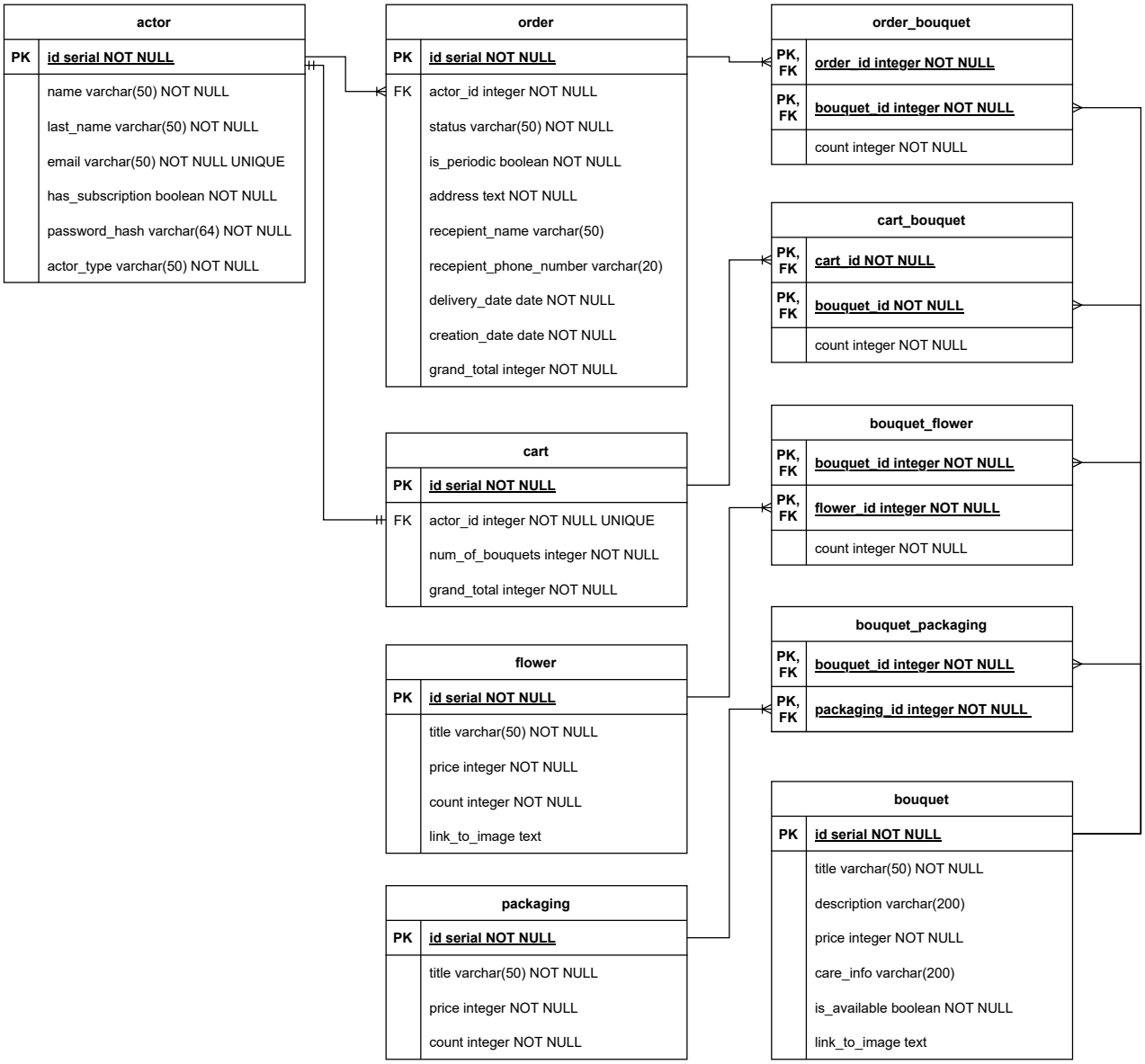

Рисунок 3 – Диаграмма базы данных

Таблица **actor** содержит информацию о пользователях и имеет следующий заголовок:

— id — уникальный идентификатор пользователя, не может повторяться в пределах таблицы;

— name — имя пользователя, задается при регистрации;

— last_name — фамилия пользователя, задается при регистрации;

— email — электронная почта пользователя, используется для авторизации и регистрации и является уникальным;

— password_hash — хэш пароля пользователя;

— has_subscription — наличие у пользователя подписки на рассылку букетов, значение по умолчанию — false;
— actor_type — тип пользователя, может принимать следующие значения: regular (значение по умолчанию), seller, admin.

Таблица **cart** содержит информацию о корзинах и имеет следующий заголовок:

— id — уникальный идентификатор корзины, не может повторяться в пределах таблицы;
— actor_id — идентификатор владельца корзины, является уникальным;
— num_of_bouquets — количество букетов в корзине;
— grand_total — сумма стоимости букетов, находящихся в корзине.

Таблица **order** содержит информацию о заказах и имеет следующий заголовок:

— id — уникальный идентификатор заказа, не может повторяться в пределах таблицы;
— actor_id — идентификатор пользователя, сделавшего заказ;
— status — статус заказа, может принимать следующие значения: accepted, delivered;
— is_periodic — флаг, показывающий, является ли заказ заказом по рассылке;
— address — адрес доставки;
— recepient_name — имя получателя;
— recepient_phone_number — номер телефона получателя;
— delivery_date — дата доставки;
— creation_date — дата создания;
— grand_total — сумма заказа.

Таблица **bouquet** содержит информацию о букетах и имеет следующий заголовок:

— id — уникальный идентификатор букета, не может повторяться в преде-

лах таблицы;

— title — название букета;

— description — описание;

— price — цена букета;

— care_info — информация об уходе;

— is_available — флаг, показывающий, доступен ли букет для заказа;

— link_to_image — ссылка на картинку.

Таблица **flower** содержит информацию о цветах и имеет следующий заголовок:

— id — уникальный идентификатор цветка, не может повторяться в пределах таблицы;

— title — название цветка;

— price — цена цветка;

— count — количество цветов на складе;

— link_to_image — ссылка на картинку.

Таблица **packaging** содержит информацию об упаковках и имеет следующий заголовок:

— id — уникальный идентификатор упаковки, не может повторяться в пределах таблицы;

— title — название упаковки;

— price — цена упаковки;

— count — количество упаковки на складе.

Таблица **cart_bouquet** содержит информацию о букетах в корзине и имеет следующий заголовок:

— cart_id — идентификатор корзины, внешний ключ на поле id таблицы cart;

— bouquet_id — идентификатор букета, внешний ключ на поле id таблицы bouquet;

— count — количество букетов с идентификатором bouquet_id в корзине с идентификатором cart_id.

Таблица **order_bouquet** содержит информацию о букетах в заказе и имеет следующий заголовок:

— order_id — идентификатор заказа, внешний ключ на поле id таблицы order;
— bouquet_id — идентификатор букета, внешний ключ на поле id таблицы bouquet;
— count — количество букетов с идентификатором bouquet_id в заказе с идентификатором order_id.

Таблица **bouquet_flower** содержит информацию о цветах в букете и имеет следующий заголовок:

— bouquet_id — идентификатор букета, внешний ключ на поле id таблицы bouquet;
— flower_id — идентификатор цветка, внешний ключ на поле id таблицы flower;
— count — количество цветов с идентификатором flower_id в букете с идентификатором bouquet_id.

Таблица **bouquet_packaging** содержит информацию об упаковках в букете и имеет следующий заголовок:

— bouquet_id — идентификатор букета, внешний ключ на поле id таблицы bouquet;
— packaging_id — идентификатор упаковки, внешний ключ на поле id таблицы packaging;

2.2 Проектирование триггеров базы данных

При проектировании базы данных разработана таблица bouquet, содержащая информацию о букетах. Букеты могут быть недоступны для заказа, если для их создания отсутствуют необходимые цветы или упаковка. Для обновления статуса букета при отсутствии его составных частей принято решение

реализовать эту функциональность в форме триггеров.

Триггер является именованным модулем PL/SQL, который хранится в базе данных и может быть вызван повторно. Триггер можно включать и отключать, но его нельзя явно вызывать. Когда триггер включен, база данных автоматически вызывает его — триггер срабатывает всякий раз, когда происходит событие, вызывающее триггер [17].

В разрабатываемой базе данных определены следующие триггеры обновления статуса букета:

— trig_flower_delete;
— trig_packaging_delete;
— trig_flower_update;
— trig_packaging_update.

Триггеры trig_flower_delete и trig_packaging_delete должны срабатывать до удаления записи из соответствующих таблиц; триггеры trig_flower_update и trig_packaging_update должны срабатывать после обнуления поля count в записях в соответствующих таблицах. Если букет содержит цветы или упаковку, которых нет на складе, поле is_available таблицы букетов принимает значение false.

На рисунках 4 и 5 представлены схемы алгоритмов обновления статуса букета.

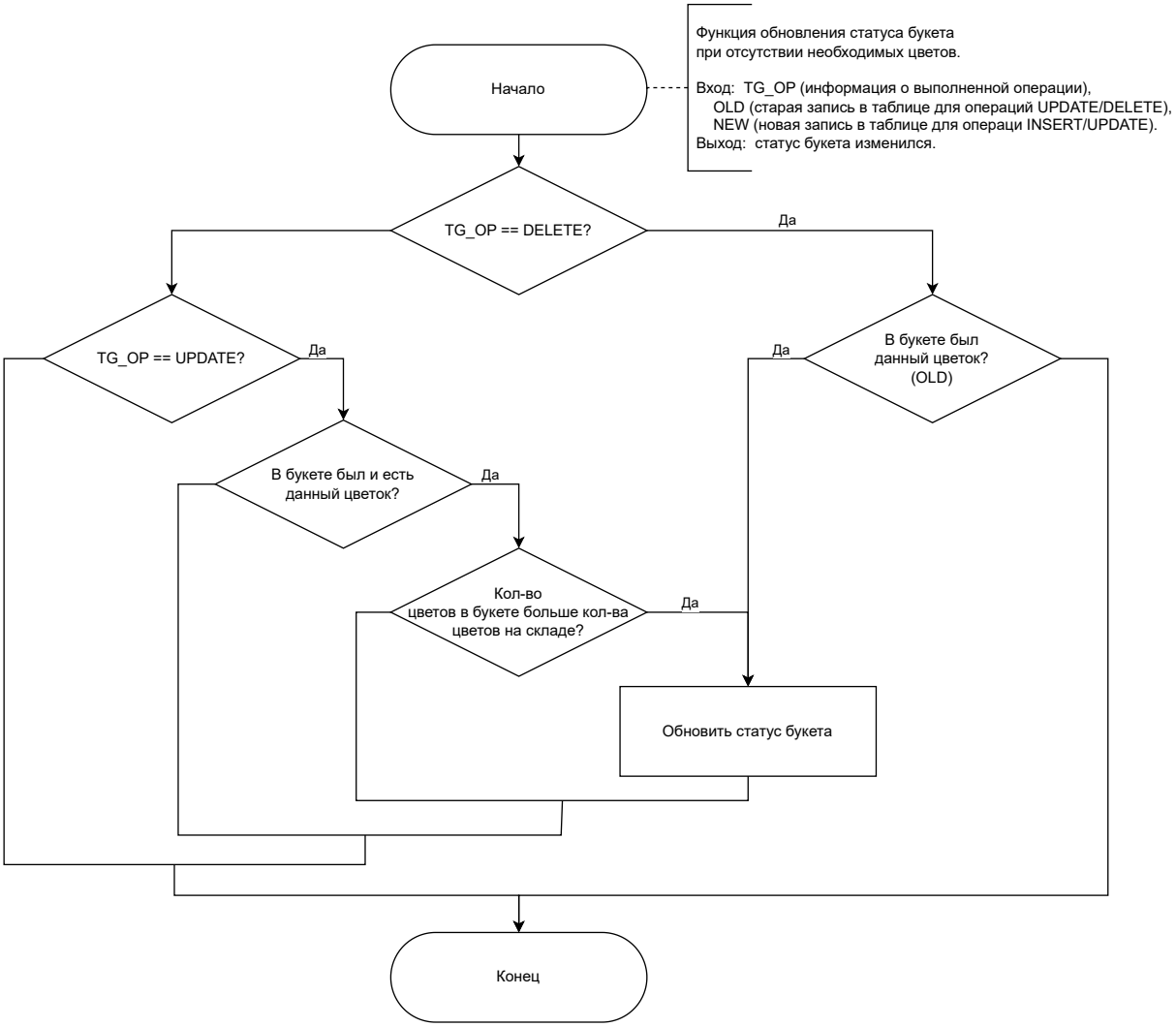

Рисунок 4 – Схема алгоритма обновления статуса букета при изменении таблицы цветов

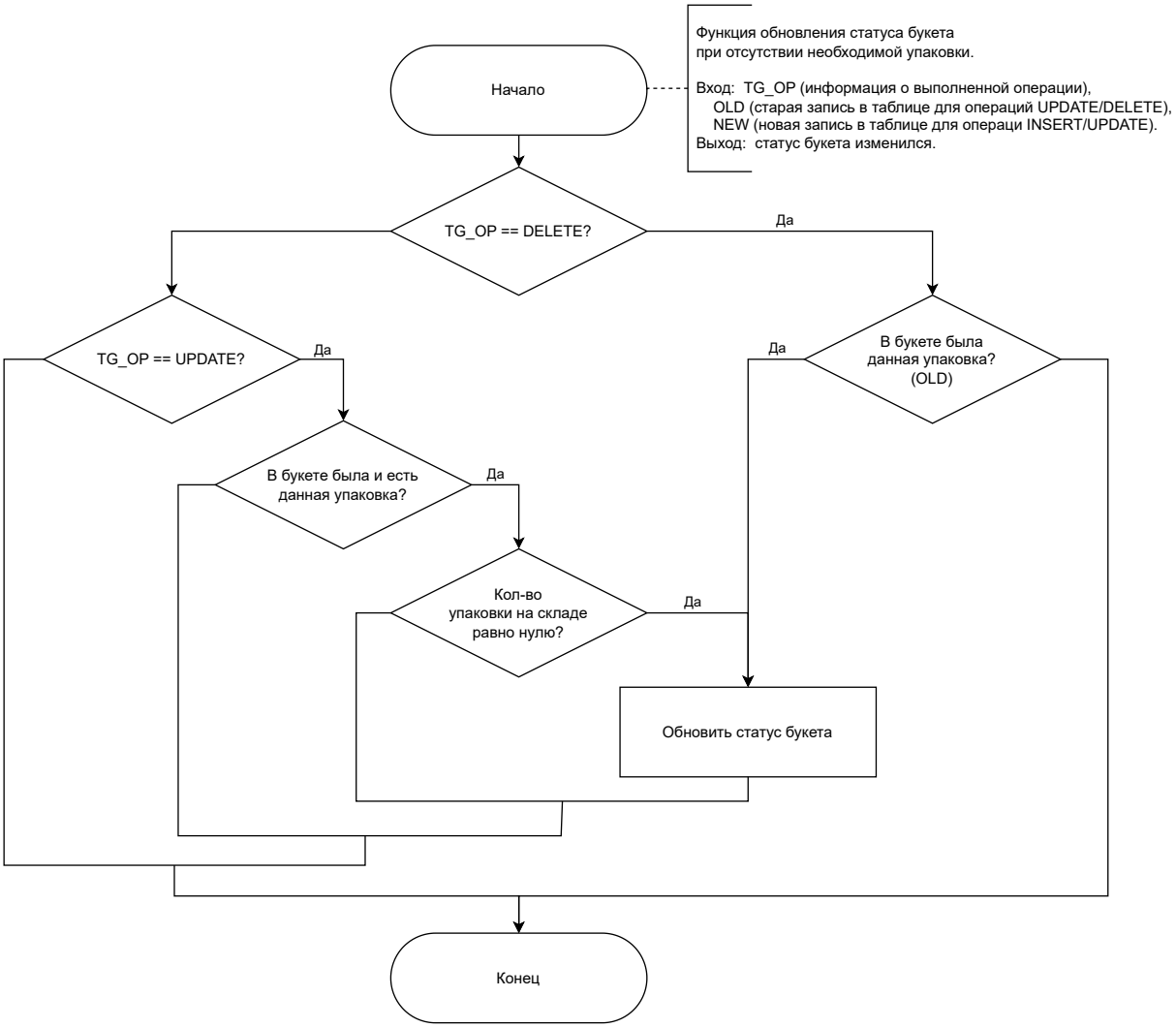

Рисунок 5 – Схема алгоритма обновления статуса букета при изменении таблицы упаковок

2.3 Роли базы данных

В аналитической части выделено четыре категории пользователей: гость, авторизованный пользователь, продавец и администратор. Для каждой категории требуется создать роль в базе данных.

1. Гость (незарегистрированный или неавторизованный пользователь) может просмотреть каталог букетов, цветы и упаковки в данном букете, следовательно, он должен иметь возможность просматривать таблицы bouquet, flower, packaging, bouquet_flower и bouquet_packaging. Также гость может зарегистрироваться или авторизоваться, для этого ему требуются права на добавление записей в таблицу actor.
2. Авторизованный пользователь, помимо прав гостя, может оформлять заказы и просматривать историю своих заказов, а также он может добавлять и удалять букеты из корзины, следовательно он должен иметь права на просмотр, изменение таблиц cart, order, cart_bouquet, order_bouquet и добавление в них записей.
3. Продавец может редактировать цветы, упаковку и букеты, следовательно он должен иметь права на изменение таблиц flower, packaging, bouquet, bouquet_flower, bouquet_packaging.
4. Администратор может просматривать все таблицы, редактировать цветы, упаковку и букеты и добавлять новые или удалять старые, следовательно, он должен иметь права на изменение таблиц flower, packaging, bouquet, bouquet_flower, bouquet_packaging, добавление новых записей в них и удаление старых.

2.4 Выводы к конструкторскому разделу

В данном разделе спроектирована база данных: на основе ER-диаграммы сущностей описаны таблицы и отношения между этими таблицами в форме диаграммы базы данных в нотации Мартина. Обоснована необходимость триггеров для обновления статуса букета при отсутствии его составных частей: цветов или упаковки. Представлены схемы функций триггеров при изменении таблицы цветов и при изменении таблицы упаковки.

3 Технологический раздел

3.1 Выбор системы управления базами данных

Существует большое количество реляционных СУБД, и самые популярные из них это [18] — Oracle [19], MySQL [20], Microsoft SQL Server [21], PostgreSQL [22].

Для сравнения выбранных СУБД выделены следующие критерии:

1) бесплатное распространение СУБД;
2) производительность (на основе источника [23]).
3) наличие опыта работы с СУБД.

Результаты сравнения выбранных СУБД по заданным критериям представлены в таблице 3.

Таблица 3 – Сравнение выбранных СУБД

Критерий	Oracle	MySQL	Microsoft SQL Server	PostgreSQL
1	-	+	-	+
2	2	3	4	1
3	-	-	-	+

По результатам сравнения в качестве СУБД для реляционной модели баз данных выбран PostgreSQL, удовлетворяющий всем критериям и являющийся наиболее производительным.

Для реализации работы с сессиями в веб-сайте необходима NoSQL система управления базами данных [24]. В этой СУБД записи хранятся в парах «ключ — значение», где ключ выступает уникальным идентификатором. Ключи и значения фиксируются в виде простой или составной информации [25]. В качестве NoSQL СУБД выбрана Redis [26] — in-memory система управления

базами данных, используемая как для баз данных, так и для реализации кэшей. Ориентирована на достижение максимальной производительности за счет работы в оперативной памяти [27].

3.2 Средства реализации программного обеспечения

Для реализации веб-сайта, предоставляющего пользователям интерфейс доступа к базе данных, выбрана клиент-серверная архитектура [28].

Для разработки серверной части веб-сайта выбран язык программирования Go [29]. Выбор обусловлен наличием опыта разработки на этом языке. В качестве фреймворка для реализации API выбран Fiber [30]. Fiber предоставляет наибольшую производительность среди существующих Go фреймворков для реализации REST API [30]. Для разработки пользовательского интерфейса выбран язык программирования Typescript [31] и библиотека ReactJS [32]. ReactJS позволяет создавать SPA приложения [33], основанные на компонентах, которые можно использовать заново без дублирования кода.

В качестве среды разработки выбран Visual Studio Code [34], что обусловлено тем, что Visual Studio Code является бесплатным программным обеспечением.

3.3 Создание таблиц

Создание таблиц базы данных и ограничений целостности полей представлено в листингах 1–10.

Листинг 1 – Создание таблицы actor

```
CREATE TABLE IF NOT EXISTS actor (
    id serial NOT NULL PRIMARY KEY,
    "name" varchar(50) NOT NULL,
    last_name varchar(50) NOT NULL,
    email varchar(50) NOT NULL UNIQUE,
    has_subscription boolean NOT NULL DEFAULT false,
    password_hash varchar(64) NOT NULL,
    actor_type varchar(50) NOT NULL
);
```

Листинг 2 – Создание таблицы cart

```
CREATE TABLE IF NOT EXISTS cart (
    id serial NOT NULL PRIMARY KEY,
    actor_id integer NOT NULL UNIQUE,
    num_of_bouquets integer NOT NULL DEFAULT 0,
    grand_total integer NOT NULL DEFAULT 0
);
```

Листинг 3 – Создание таблицы flower

```
CREATE TABLE IF NOT EXISTS flower (
    id serial NOT NULL PRIMARY KEY,
    title varchar(50) NOT NULL,
    price integer NOT NULL,
    count integer NOT NULL,
    link_to_image text
);
```

Листинг 4 – Создание таблицы packaging

```
CREATE TABLE IF NOT EXISTS packaging (
    id serial NOT NULL PRIMARY KEY,
    title varchar(50) NOT NULL,
    price integer NOT NULL,
    count integer NOT NULL
);
```

Листинг 5 – Создание таблицы bouquet

```
CREATE TABLE IF NOT EXISTS bouquet (
    id serial NOT NULL PRIMARY KEY,
    title varchar(50) NOT NULL,
    "description" varchar(200),
    price integer NOT NULL,
    care_info varchar(200),
    is_available boolean NOT NULL,
    link_to_image text
);
```

Листинг 6 – Создание таблицы order

```
CREATE TABLE IF NOT EXISTS "order" (
    id serial NOT NULL PRIMARY KEY,
    actor_id integer NOT NULL,
    "status" varchar(50) NOT NULL,
    is_periodic boolean NOT NULL DEFAULT false,
    "address" text NOT NULL,
    recipient_name varchar(50),
    recipient_phone_number varchar(20),
    delivery_date timestamp NOT NULL,
    creation_date timestamp NOT NULL DEFAULT now(),
    grand_total integer NOT NULL
);
```

Листинг 7 – Создание таблицы bouquet_flower

```
CREATE TABLE IF NOT EXISTS bouquet_flower (
    bouquet_id integer NOT NULL,
    flower_id integer NOT NULL,
    count integer NOT NULL DEFAULT 0
);

ALTER TABLE bouquet_flower
    ADD CONSTRAINT bouquet_flower_fk1 FOREIGN KEY (bouquet_id) REFERENCES
       bouquet (id) ON DELETE CASCADE,
    ADD CONSTRAINT bouquet_flower_fk2 FOREIGN KEY (flower_id) REFERENCES
       flower (id) ON DELETE CASCADE,
    ADD CONSTRAINT bouquet_flower_pk PRIMARY KEY (bouquet_id, flower_id);
```

Листинг 8 – Создание таблицы bouquet_packaging

```
CREATE TABLE IF NOT EXISTS bouquet_packaging (
    bouquet_id integer NOT NULL,
    packaging_id integer NOT NULL
);

ALTER TABLE bouquet_packaging
    ADD CONSTRAINT bouquet_packaging_fk1 FOREIGN KEY (bouquet_id) REFERENCES
       bouquet (id) ON DELETE CASCADE,
    ADD CONSTRAINT bouquet_packaging_fk2 FOREIGN KEY (packaging_id) REFERENCES
        packaging (id) ON DELETE CASCADE,
    ADD CONSTRAINT bouquet_packaging_pk PRIMARY KEY (bouquet_id, packaging_id)
       ;
```

Листинг 9 – Создание таблицы cart_bouquet

```
CREATE TABLE IF NOT EXISTS cart_bouquet (
    cart_id integer NOT NULL,
    bouquet_id integer NOT NULL,
    count integer NOT NULL DEFAULT 0
);

ALTER TABLE cart_bouquet
    ADD CONSTRAINT cart_bouquet_fk1 FOREIGN KEY (cart_id) REFERENCES cart (id)
        ON DELETE CASCADE,
    ADD CONSTRAINT cart_bouquet_fk2 FOREIGN KEY (bouquet_id) REFERENCES
       bouquet (id) ON DELETE CASCADE,
    ADD CONSTRAINT cart_bouquet_pk PRIMARY KEY (cart_id, bouquet_id);
```

Листинг 10 – Создание таблицы order_bouquet

```
CREATE TABLE IF NOT EXISTS order_bouquet (
    order_id integer NOT NULL,
    bouquet_id integer NOT NULL,
    count integer NOT NULL DEFAULT 0
);

ALTER TABLE order_bouquet
    ADD CONSTRAINT order_bouquet_fk1 FOREIGN KEY (order_id) REFERENCES "order"
        (id) ON DELETE CASCADE,
    ADD CONSTRAINT order_bouquet_fk2 FOREIGN KEY (bouquet_id) REFERENCES
       bouquet (id) ON DELETE CASCADE,
    ADD CONSTRAINT order_bouquet_pk PRIMARY KEY (order_id, bouquet_id);
```

3.4 Реализация ролей базы данных

Реализация ролей базы данных представлена в листингах 11–14.

Листинг 11 – Создание роли неавторизованного пользователя

```
CREATE ROLE not_auth_actor WITH
    LOGIN
    NOSUPERUSER
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    PASSWORD 'not_auth_actor'
    CONNECTION LIMIT -1;

GRANT SELECT ON bouquet,
                bouquet_flower,
                bouquet_packaging,
                flower,
                packaging
             TO not_auth_actor;

GRANT INSERT ON actor TO not_auth_actor;
```

Листинг 12 – Создание роли авторизованного пользователя

```
CREATE ROLE auth_actor WITH
    LOGIN
    NOSUPERUSER
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    PASSWORD 'auth_actor'
    CONNECTION LIMIT -1;

GRANT SELECT ON bouquet,
                bouquet_flower,
                bouquet_packaging,
                flower,
                packaging,
                cart,
                cart_bouquet,
                "order",
                order_bouquet
             TO auth_actor;

GRANT INSERT ON "order",
                cart_bouquet,
                order_bouquet
             TO auth_actor;

GRANT UPDATE ON cart,
                cart_bouquet
             TO auth_actor;

GRANT DELETE ON cart_bouquet TO auth_actor;
```

Листинг 13 – Создание роли продавца

```
CREATE ROLE seller WITH
    LOGIN
    NOSUPERUSER
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    PASSWORD 'seller'
    CONNECTION LIMIT -1;

GRANT SELECT, UPDATE ON bouquet,
                        bouquet_flower,
                        bouquet_packaging,
                        flower,
                        packaging
                        TO seller;
```

Листинг 14 – Создание роли администратора

```
CREATE ROLE administrator WITH
    LOGIN
    NOSUPERUSER
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    PASSWORD 'administrator'
    CONNECTION LIMIT -1;

GRANT SELECT ON bouquet,
                bouquet_flower,
                bouquet_packaging,
                flower,
                packaging,
                cart,
                cart_bouquet,
                "order",
                order_bouquet,
                actor
             TO administrator;

GRANT UPDATE, INSERT, DELETE ON bouquet,
                                bouquet_flower,
                                bouquet_packaging,
                                flower,
                                packaging
                                TO seller;
```

3.5 Реализация триггеров базы данных

Триггеры обновления статуса букета при отсутствии его составных частей (нужного количества цветов или упаковки) представлены в листингах 15 и 16.

Листинг 15 – Триггеры и функция обновления статуса букета при изменениях в таблице цветов

```
CREATE OR REPLACE FUNCTION bouquet_available_by_flower() RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        UPDATE bouquet
        SET is_available = FALSE
        WHERE id IN (SELECT bouquet_id FROM bouquet_flower WHERE flower_id =
            OLD.id);
        RETURN OLD;

    ELSIF (TG_OP = 'UPDATE') THEN
        UPDATE bouquet
        SET is_available = FALSE
        WHERE id IN (SELECT bouquet_id FROM bouquet_flower
                     WHERE flower_id = OLD.id AND flower_id = NEW.id AND
                         count > NEW.count);
        RETURN NEW;

    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trig_flower_delete BEFORE
DELETE ON flower
FOR EACH ROW EXECUTE PROCEDURE bouquet_available_by_flower();

CREATE TRIGGER trig_flower_update AFTER
UPDATE ON flower
FOR EACH ROW EXECUTE PROCEDURE bouquet_available_by_flower();
```

Листинг 16 – Триггеры и функция обновления статуса букета при изменениях в таблице упаковки

```
CREATE OR REPLACE FUNCTION bouquet_available_by_packaging() RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        UPDATE bouquet
        SET is_available = FALSE
        WHERE id IN (SELECT bouquet_id FROM bouquet_packaging WHERE packaging_id = OLD.id);
        RETURN OLD;

    ELSIF (TG_OP = 'UPDATE') THEN
        UPDATE bouquet
        SET is_available = FALSE
        WHERE id IN (SELECT bouquet_id FROM bouquet_packaging
                     WHERE packaging_id = OLD.id AND packaging_id = NEW.id
                        AND NEW.count = 0);
        RETURN NEW;

    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trig_packaging_delete BEFORE
DELETE ON packaging
FOR EACH ROW EXECUTE PROCEDURE bouquet_available_by_packaging();

CREATE TRIGGER trig_packaging_update AFTER
UPDATE ON packaging
FOR EACH ROW EXECUTE PROCEDURE bouquet_available_by_packaging();
```

3.6 Примеры работы веб-сайта

Пример главной страницы сайта представлен на рисунке 6. Примеры формы для авторизации и формы для регистрации представлены на рисунках 7 и 8 соответственно. Примеры корзины и оформления заказа представлены на рисунках 9 и 10. Пример страницы пользователя с историей заказов представлен на рисунке 11.

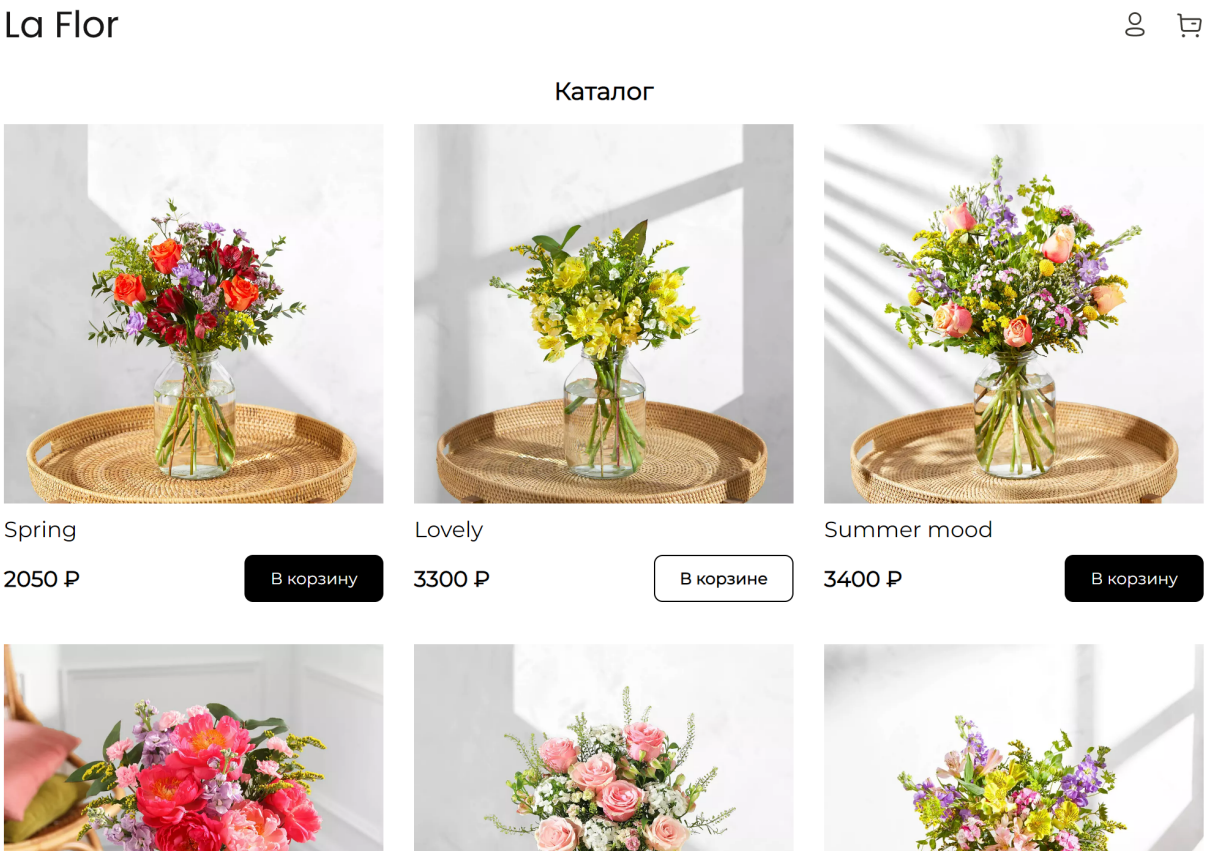

Рисунок 6 – Пример главной страницы

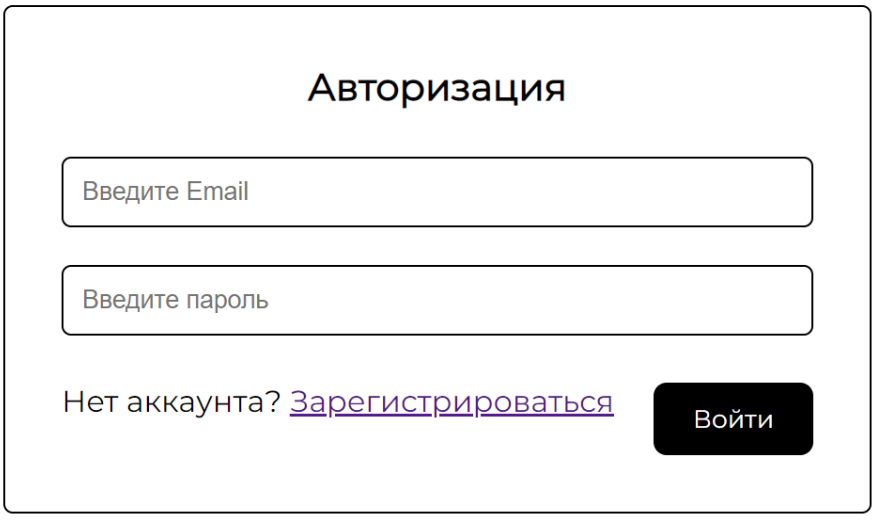

Рисунок 7 – Пример формы для авторизации

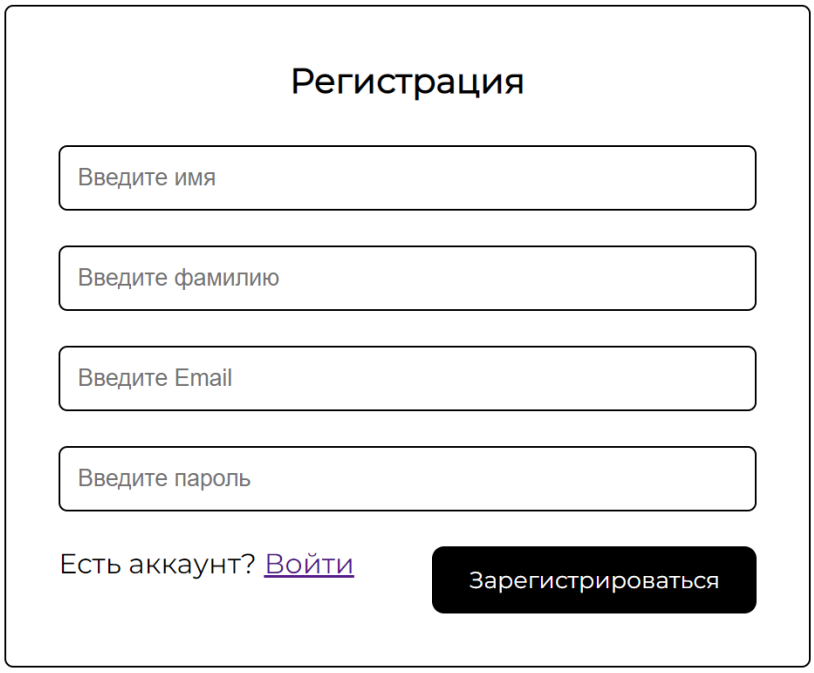

Рисунок 8 – Пример формы для регистрации

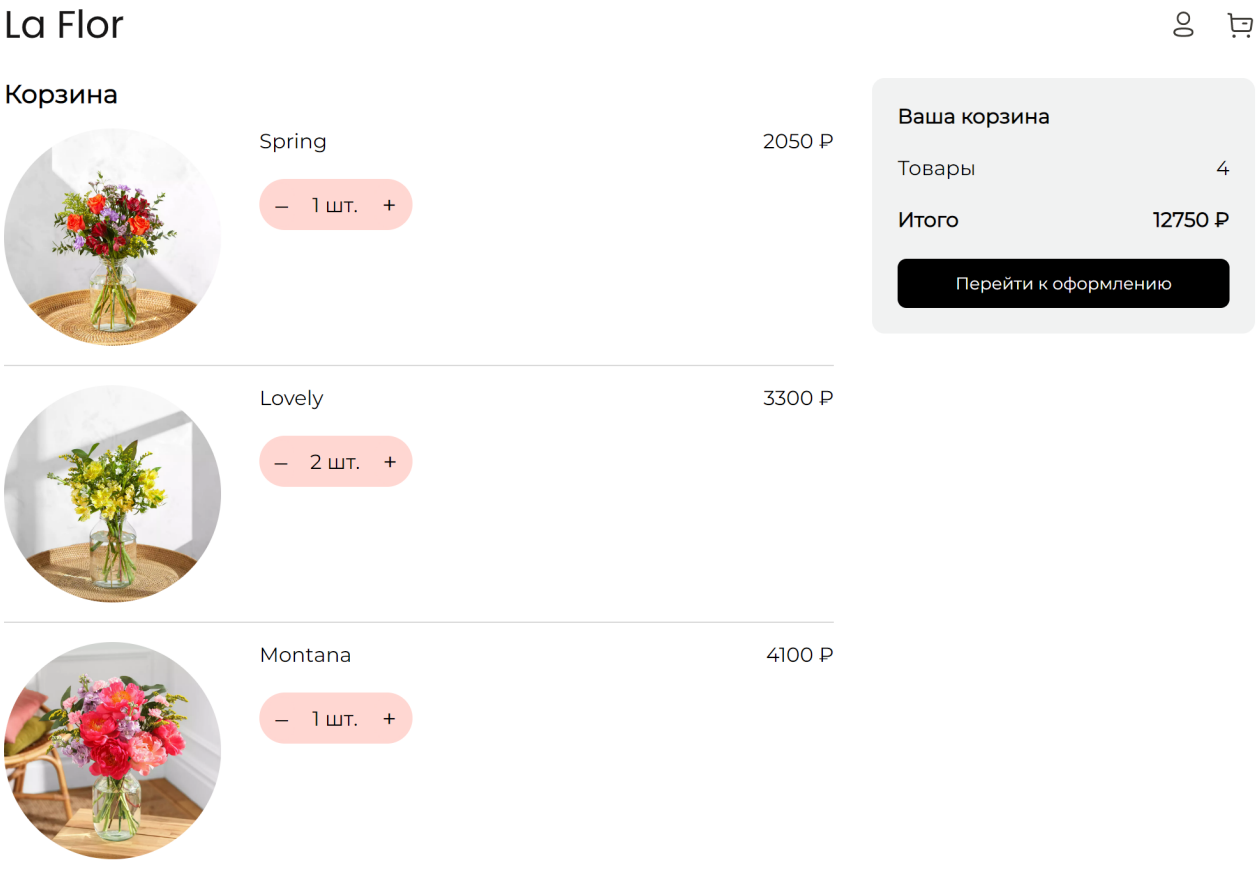

Рисунок 9 – Пример корзины пользователя

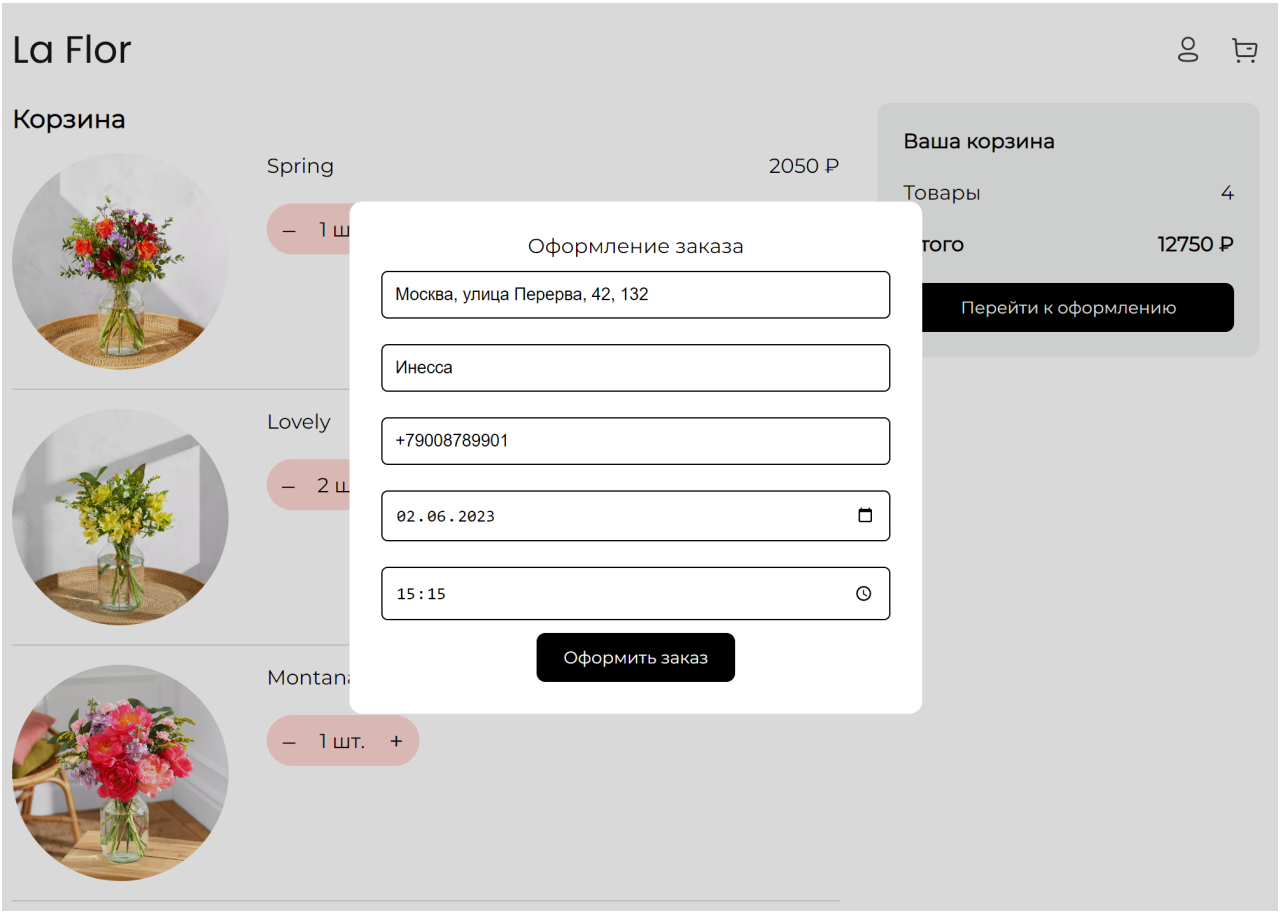

Рисунок 10 – Пример оформления заказа

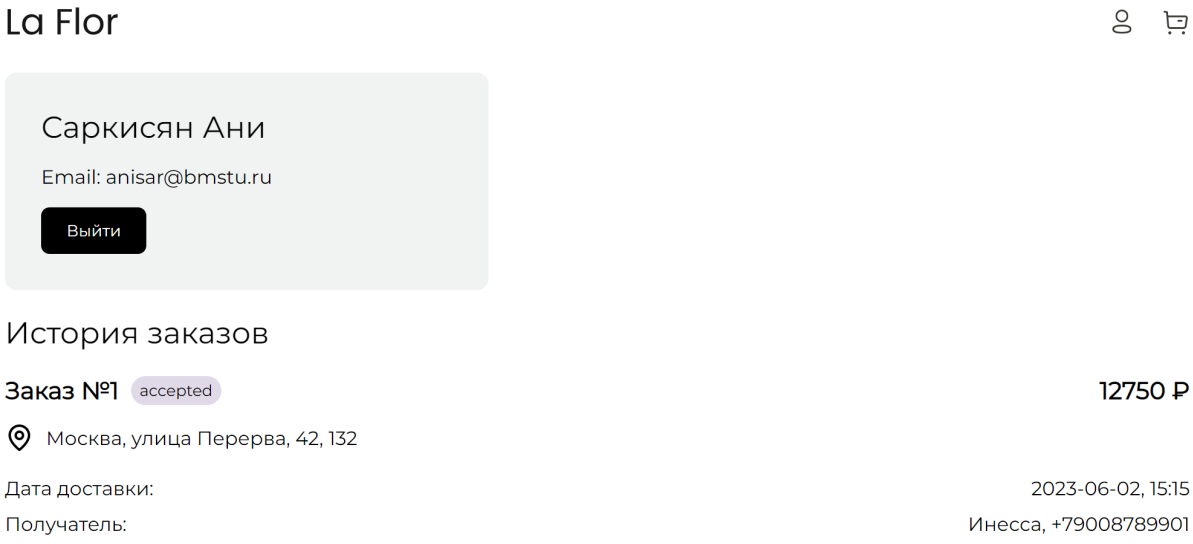

Рисунок 11 – Пример страницы пользователя

3.7 Выводы к технологическому разделу

В данном разделе выбраны и описаны средства реализации: в качестве системы управления базами данных выбрана PostgreSQL; для реализации интерфейса к базе данных выбрана клиент-серверная архитектура и такие языки программирования, как Go (серверная часть) и Typescript (клиентская часть). Приведены детали реализации ролей, триггеров и создания таблиц базы данных и представлены примеры работы программы.

4 Исследовательский раздел

4.1 Описание исследования

Целью исследования является определение зависимости времени выполнения запроса от наличия индексации. Для проведения исследования выбрана таблица bouquet, для индексации выбрано поле price.

Создание индекса приведено в листинге 18.

Листинг 17 – Создание индекса по полю price таблицы packaging

```
CREATE INDEX bouquet_price_index
          ON bouquet USING btree(price);
```

Время выполнения запроса анализировалось с помощью встроенной в PostgreSQL команды EXPLAIN [35].

Листинг 18 – Запрос для анализа времени выполнения запроса к БД

```
EXPLAIN ANALYSE
SELECT * FROM bouquet
ORDER BY price;
```

4.2 Результаты исследования

Технические характеристики устройства, на котором выполнялись замеры времени, представлены далее:

— операционная система Windows 11, версия 22H2;

— память 16 Гб;

— 12th Gen Intel(R) Core(TM) i7-12700H 2.30 ГГц [36].

Результаты замеров времени (в миллисекундах) выполнения запроса в зависимости от количества записей и наличия индекса приведены в таблице 4.

При этом было проведено усреднение для получения наиболее точных результатов.

Таблица 4 – Результаты замеров времени

Количество записей	Время без индекса, мс	Время с индексом, мс
10	0.047	0.043
50	0.086	0.074
100	0.112	0.145
500	0.182	0.206
1000	0.317	0.347
2000	0.556	0.402
5000	1.262	0.824
10000	2.323	1.949
15000	4.969	2.793
20000	5.671	3.704

График, построенный по результатам замеров, представлен на рисунке 12.

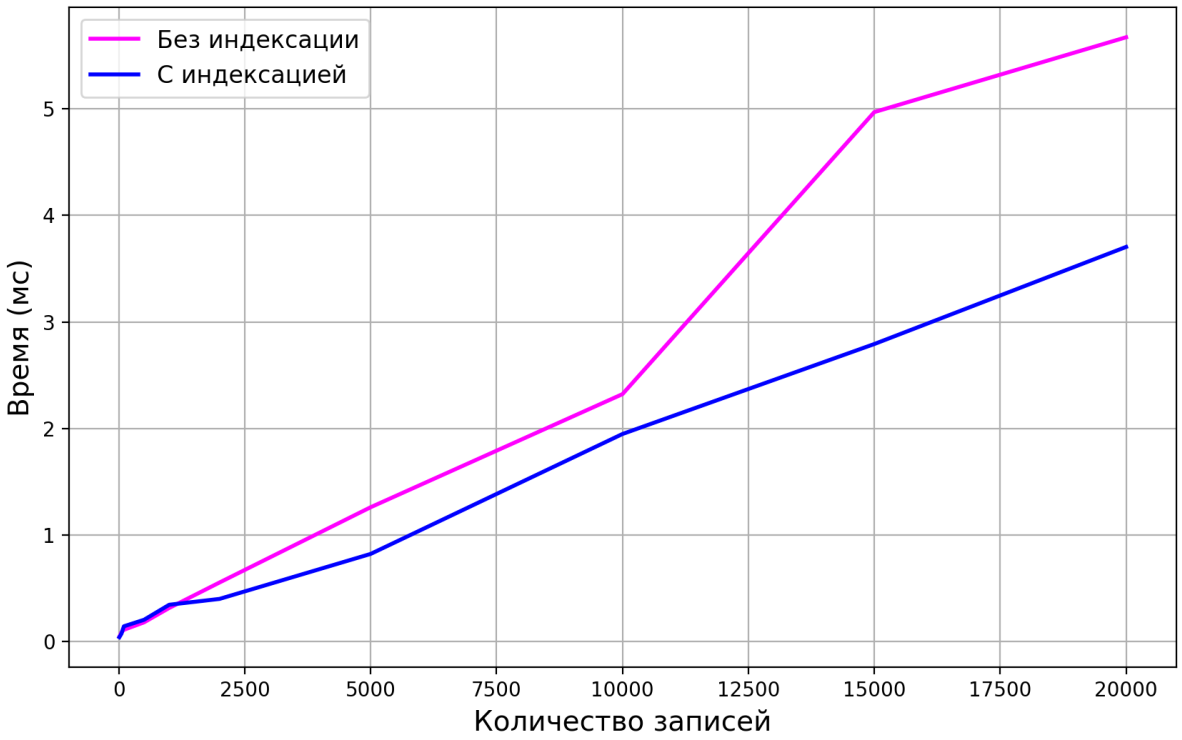

Рисунок 12 – Зависимость времени выполнения запроса от количества записей и наличия индексации

4.3 Выводы к исследовательскому разделу

В данном разделе описано исследование зависимости времени выполнения запросов от использования индексации при различных количествах записей в таблице bouquet; приведены технические характеристики устройства, на котором выполнялось исследование и представлены его результаты в виде таблицы и графика. Наличие индекса ускоряет время запроса приблизительно в 1,5 раза.

ЗАКЛЮЧЕНИЕ

Цель, поставленная в начале курсовой работы, была достигнута: была разработана база данных для хранения и обработки данных цветочного магазина.

Все задачи решены:

— были сформулированы требования и ограничения к разрабатываемой базе данных и веб-сайту цветочного магазина;
— были сформулированы описание пользователей проектируемого приложения для доступа к базе данных;
— были спроектированы сущности базы данных и ограничения целостности цветочного магазина;
— был спроектирован триггер обновления статуса букета при отсутствии его составных частей;
— были разработаны сущности базы данных цветочного магазина и реализовать ограничения целостности базы данных;
— было реализовано приложение, предоставляющее интерфейс доступа к базе данных;
— была исследована зависимость времени выполнения запросов от использования индексации при различных количествах записей в таблице.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Семенова Ю.А., Мохнаткина К.В. Методы разработки информационной системы цветочного магазина — Тенденции развития науки и образования, 2022 — С. 123-124.

2 Floristry and Floriculture Industry Statistics & Trends (2023) [Электронный ресурс]. URL: https://www.petalrepublic.com/floristry-and-floriculture-statistics/#28-sources (дата обращения: 11.03.2023).

3 Florist Statistics 2023 [Электронный ресурс]. URL: https://webinarcare.com/best-florist-software/florist-statistics/ (дата обращения: 11.03.2023).

4 Flor2U [Электронный ресурс]. URL: https://flor2u.ru/ (дата обращения: 11.03.2023).

5 Флорист.ру [Электронный ресурс]. URL: https://www.florist.ru/ (дата обращения: 11.03.2023).

6 Flowwow [Электронный ресурс]. URL: https://flowwow.com/ (дата обращения: 11.03.2023).

7 Ксенз А.С. Яхонтова И.М. Модель «Сущность-связь» в нотациях Чена и Баркера (модель Чена) // Информационное общество: современное состояние и перспективы развития — Краснодар: Кубанский государственный аграрный университет имени И.Т. Трубилина, 2017 — С. 247–249.

8 Швейкин В.В. Ролевая модель разграничения доступа в СУБД // Научные исследования и разработки студентов. Сборник материалов IV студенческой научно–практической конф. / г. Чебоксары (июнь 2017), — Самара, 2017. — С. 213–215.

9 Gemino A., Parker D. Use Case Diagrams in Support of Use Case Modeling: Deriving Understanding from the Picture // Journal of Database Management — Simon Fraser University, Canada, 2009 — Pp. 1–24.

10 Карпова И.П. Базы данных. Курс лекций и материалы для практических заданий — М: Питер, 2013 — 7 с.

11 Сергеева Т.И. Базы данных: модели данных, проектирование, язык SQL: учеб. пособие / Воронеж: ФГБОУ ВПО «Воронежский государственный технический университет», 2012. — 233 с.

12 Росланов И.Ю., Кудряшова Е.С. Классификация СУБД // Наука, инновации и технологии: от идей к внедрению. Материалы II Международной научно-практической конф. молодых ученых / Комсомольск-на-Амуре, 2022. — С. 256–259.

13 What is a Relational Database (RDBMS)? [Электронный ресурс]. URL: https://www.oracle.com/database/what-is-a-relational-database/ (дата обращения: 11.03.2023).

14 SQL [Электронный ресурс]. URL: https://docs.oracle.com/en/database/oracle/oracle-database/18/cncpt/sql.html (дата обращения: 12.03.2023).

15 Парфенов Ю.П. Постреляционные хранилища данных — Екатеринбург: Издательство Уральского университета, 2016. — 120 с.

16 Нотация Мартина (Crow's Foot) [Электронный ресурс]. URL: https://studme.org/77223/informatika/notatsiya_martina_crows_foot#999/ (дата обращения: 13.04.2023).

17 PL/SQL Triggers [Электронный ресурс]. URL: https://docs.oracle.com/en/database/oracle/oracle-database/21/lnpls/plsql-triggers.

html#GUID-217E8B13-29EF-45F3-8D0F-2384F9F1D231 (дата обращения: 14.04.2023).

18 Ranking of the most popular database management systems worldwide, as of February 2023 [Электронный ресурс]. URL: https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/ (дата обращения: 14.04.2023).

19 Oracle database [Электронный ресурс]. URL: https://www.oracle.com/database/ (дата обращения: 14.04.2023).

20 MySQL [Электронный ресурс]. URL: https://www.mysql.com/ (дата обращения: 14.04.2023).

21 SQL Server 2022 [Электронный ресурс]. URL: https://www.microsoft.com/ru-ru/sql-server/sql-server-2022 (дата обращения: 14.04.2023).

22 PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс]. URL: https://www.postgresql.org/ (дата обращения: 14.04.2023).

23 Oracle, MySQL, PostgreSQL, SQLite, SQL Server: Performance based competitive analysis [Электронный ресурс]. URL: http://dspace.daffodilvarsity.edu.bd:8080/handle/123456789/4437 (дата обращения: 20.05.2022).

24 Нереляционные данные и базы данных NoSQL [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/azure/architecture/data-guide/big-data/non-relational-data (дата обращения: 10.05.2023).

25 NoSQL: виды, особенности и применение [Электронный ресурс]. URL:

https://cloud.yandex.ru/blog/posts/2022/10/nosql (дата обращения: 10.05.2023).

26 Redis [Электронный ресурс]. URL: https://redis.io/ (дата обращения: 20.04.2023).

27 Redis CPU Profiling [Электронный ресурс]. URL: https://redis.io/docs/management/optimization/cpu-profiling/ (дата обращения: 20.04.2023).

28 Client Server Architecture [Электронный ресурс]. URL: https://www.enjoyalgorithms.com/blog/client-server-architecture (дата обращения: 20.05.2022).

29 The Go Programming Language [Электронный ресурс]. URL: https://go.dev/ (дата обращения: 14.04.2023).

30 An Express-inspired web framework written in Go [Электронный ресурс]. URL: https://gofiber.io/ (дата обращения: 14.04.2023).

31 TypeScript is JavaScript with syntax for types [Электронный ресурс]. URL: https://www.typescriptlang.org/ (дата обращения: 18.04.2023).

32 React [Электронный ресурс]. URL: https://react.dev/ (дата обращения: 18.04.2023).

33 Single Page Application: как работает сайт-приложение [Электронный ресурс]. URL: https://thecode.media/spa/ (дата обращения: 19.04.2023).

34 Code editing. Redefined. [Электронный ресурс]. URL: https://code.visualstudio.com/ (дата обращения: 18.04.2023).

35 EXPLAIN [Электронный ресурс]. URL: https://www.postgresql.org/docs/current/sql-explain.html (дата обращения: 18.04.2023).

36 Процессор Intel® Core™ i7-12700H [Электронный ресурс]. URL: https://ark.intel.com/content/www/ru/ru/ark/products/132228/intel-core-i712700h-processor-24m-cache-up-to-4-70-ghz.html (дата обращения: 10.05.2022).