# Содержание

[Введение](#_Toc196220410)

1. [Теоретическая часть 4](#_Toc196220410)

1.1. [Описание программируемой системы 4](#_Toc196220410)

1.1.1 [Назначение и роль баз данных в информационных системах 4](#_Toc196220410)

1.1.2 [Основные сущности предметной области 4](#_Toc196220410)

1.1.3 [Связи между сущностями 5](#_Toc196220410)

1.1.4 [Реляционная модель и организация хранения данных 5](#_Toc196220410)

1.1.5 [Хранимые процедуры, функции и триггеры 6](#_Toc196220410)

1.1.6 [Веб-приложение как средство взаимодействия с базой данных 6](#_Toc196220410)

1.1.7 [Преимущества выбранной архитектуры 7](#_Toc196220410)

1.2. Обзор существующих решений 8

1.3. [Требования к программируемой системме 11](#_Toc196220410)

2. [Технологическая часть 13](#_Toc196220410)

**ВВЕДЕНИЕ**

В условиях цифровизации и стремительного роста объёмов данных информационные системы играют решающую роль в деятельности организаций. Основой таких систем являются базы данных, обеспечивающие хранение, обработку и структурированный доступ к информации. Особенно важно это в сфере торговли, где требуется оперировать различными данными: информации о товарах, поставщиках, истории продаж, взаимодействия с клиентами и прочим.

Музыкальные магазины (онлайн) - это специализированные торговые структуры, работа которых связана с большими объёмами информации. Такие магазины хранят сведения об исполнителях, ансамблях, музыкальных произведениях, записи на различных носителях и компаниях-издателях. Для эффективного учёта и анализа необходима современная база данных, позволяющая автоматизировать процессы и снизить количество ошибок при ручной обработке данных.

**Цель проекта:**

Разработка реляционной базы данных для музыкального магазина и создание веб-приложения, обеспечивающего эффективное взаимодействие с этой базой.

**Задачи:**

В соответствии с поставленной целью, ставятся следующие задачи:

* **Анализ предметной области** — важно детально изучить музыкальную сферу, чтобы точно определить сущности, которые будут включены в базу данных. Это поможет выявить связи между ними и сформировать модель данных.
* **Разработка логической и физической модели базы данных** — создание модели, которая будет чётко отображать взаимосвязи между сущностями. Логическая модель определяет связи между сущностями, а физическая модель уточняет, как эти данные будут храниться в базе (например, выбор типа данных для столбцов, создание индексов для оптимизации запросов).
* **Реализация структуры базы данных** — на основе модели создаются таблицы и связи между ними. Важно правильно определить типы данных и связи (внешние ключи), чтобы база данных была целостной и нормализованной.
* **Создание хранимых процедур и функций** — это помогает автоматизировать многие операции, например, добавление новых данных, обновление информации или вычисление статистики.
* **Реализация триггеров** — триггеры позволяют автоматизировать реакции на изменения в базе данных, например, при удалении записи или при изменении количества товара.
* **Разработка веб-приложения** — создание интерфейса для пользователей с использованием React на фронтенде и Go на бэкенде. Приложение должно позволять взаимодействовать с базой данных через API.
* **Обеспечение взаимодействия frontend и backend через REST API** — это основной способ связи между клиентской и серверной частью приложения. REST API предоставляет доступ к данным и операциям базы данных.
* **Организация подключения к СУБД** — важно правильно настроить соединение с СУБД, выбрать подходящий драйвер и обеспечить безопасность при доступе к базе.
* **Продумывание аспектов безопасности, удобства и масштабируемости** — обеспечение безопасной работы приложения (например, с использованием JWT-аутентификации, защита от SQL-инъекций), создание интерфейса с удобной навигацией и возможностью масштабирования системы на случай роста данных и пользователей.

1. **Теоретическая часть**

**1.1 Описание программируемой системы**

**1.1.1 Назначение и роль баз данных в информационных системах**

Базы данных (БД) являются ядром любой информационной системы. Они обеспечивают:

* централизованное хранение информации
* быстрый и безопасный доступ к данным
* возможность выполнять сложные аналитические запросы
* поддержку целостности, непротиворечивости и надёжности информации

В контексте музыкального магазина БД позволяет автоматизировать следующие процессы:

* учёт и обновление ассортимента
* фиксация и анализ продаж
* формирование отчётности
* отслеживание информации об авторах и исполнителях
* хранение истории записей и выпусков носителей

Использование базы данных снижает зависимость от ручного труда, минимизирует ошибки, ускоряет обслуживание клиентов и обеспечивает возможность масштабирования.

**1.1.2 Основные сущности предметной области**

Анализ предметной области музыкального магазина позволил выделить **ключевые сущности**, которые будут представлены в базе данных:

* Музыкант — физическое лицо, занимающееся музыкальной деятельностью: исполнитель, композитор, дирижёр
* Ансамбль — группа музыкантов, совместно исполняющая музыкальные произведения
* Произведение — музыкальная композиция, созданная одним или несколькими композиторами
* Исполнение — запись произведения в конкретном исполнении, чаще всего ансамблем
* Пластинка — носитель, содержащий одно или несколько музыкальных исполнений.
* Компания — юридическое лицо, выпускающее и распространяющее музыкальные пластинки.

**1.1.3 Связи между сущностями:**

* Один ансамбль может исполнять множество произведений
* Одно произведение может быть исполнено в разное время разными ансамблями
* Одна пластинка может содержать исполнения нескольких произведений
* Одна компания может выпускать множество пластинок

Эти взаимосвязи реализуются через внешние ключи и вспомогательные таблицы

**1.1.4 Реляционная модель и организация хранения данных**

Проектируемая база данных строится по реляционной модели, которая основывается на следующем принципе: каждая сущность представлена отдельной таблицей, а связи между сущностями — через внешние ключи. Это обеспечивает:

* структурированность и логичность хранения
* нормализацию данных (устранение дублирования)
* согласованность и целостность информации
* удобство выполнения запросов

Применяются основные нормальные формы до третьей включительно, что исключает избыточность и логические противоречия в базе.

**1.1.5 Хранимые процедуры, функции и триггеры**

Для автоматизации работы с базой данных используются следующие инструменты:

* Хранимые процедуры — блоки SQL-кода, выполняющие однотипные задачи. Примеры:
  + добавление нового исполнителя;
  + массовое обновление записей при изменении информации о компании.
* Функции — SQL-команды, возвращающие результат. Примеры:
  + подсчёт общего количества произведений, исполненных конкретным ансамблем;
  + вычисление среднего времени звучания произведений.
* Триггеры — автоматические реакции на изменение данных. Примеры:
  + каскадное удаление исполнений при удалении ансамбля;
  + обновление поля "в наличии" при продаже пластинки.

Применение этих средств обеспечивает автоматическое выполнение бизнес-логики, консистентность данных и минимизацию человеческого вмешательства.

**1.1.6 Веб-приложение как средство взаимодействия с базой данных**

Для обеспечения удобного доступа к базе данных разрабатывается веб-приложение с архитектурой «клиент–сервер».

**Клиентская часть (Frontend)**

* Реализуется с использованием библиотеки React.
* Предоставляет формы, таблицы и фильтры для отображения и ввода данных.
* Поддерживает взаимодействие с сервером через HTTP-запросы (Axios, fetch).
* Обеспечивает интуитивно понятный интерфейс, адаптивный дизайн и высокую отзывчивость.

**Серверная часть (Backend)**

* Разрабатывается на языке Go.
* Использует стандартные библиотеки (например, net/http) или фреймворки (например, Gin).
* Реализует REST API для работы с базой данных.
* Обеспечивает безопасность, проверку данных и обработку ошибок.
* Подключается к СУБД (например, PostgreSQL) через драйверы database/sql или pgx.

**База данных**

* Служит надёжным хранилищем всех сущностей предметной области.
* Обеспечивает быстрый доступ, структурность и защиту информации.
* Интегрируется с backend-сервером через драйверы Go.

**1.1.7 Преимущества выбранной архитектуры**

* **Надёжность:** язык Go отличается стабильностью и высокой производительностью при работе с большим количеством запросов.
* **Масштабируемость:** веб-приложение легко дополняется новыми функциями и интерфейсами.
* **Удобство:** React позволяет создавать современный интерфейс с высоким уровнем интерактивности.
* **Безопасность:** возможно реализовать JWT-аутентификацию, разграничение прав пользователей, защиту от SQL-инъекций и XSS-атак.
* **Универсальность:** система может быть адаптирована под другие магазины (например, книжные, кинопродукции) с минимальными изменениями.

**1.2 Обзор существующих решений**

В последние десятилетия развитие технологий и автоматизация процессов значительно улучшили многие аспекты торговли и управления в музыкальных магазинах, включая учет товаров, управление продажами и анализ данных. Существуют несколько программных решений, которые могут быть адаптированы для решения задач в музыкальном магазине, включая управление каталогом музыки, продажами винилов, компакт-дисков и других товаров. Рассмотрим несколько таких решений:

1. **Discogs**

[**Discogs**](https://www.discogs.com/) - это крупнейшая онлайн-платформа для коллекционеров виниловых пластинок и музыкальных произведений. Она предлагает подробную информацию о миллионах записей, альбомах и исполнителях, а также предоставляет возможности для отслеживания продаж и состояния товаров.

* Плюсы:
  + **Обширная база данных:** Discogs содержит огромный каталог музыкальных произведений, что делает его ценным ресурсом для музыкальных магазинов, нуждающихся в точных данных о пластинках, альбомах и исполнителях.
  + **Поддержка коллекционеров:** Платформа позволяет отслеживать состояние и редкость винилов, что важно для магазинов, ориентированных на коллекционеров.
  + **API:** Discogs предоставляет API для интеграции с внешними системами, что может быть полезно для автоматизации учета товаров и продаж.
* Минусы:
  + **Ориентированность на коллекционеров:** Платформа в основном используется коллекционерами винилов и может быть не так удобна для традиционных магазинов, продающих более широкий ассортимент музыкальных товаров.
  + **Ограниченная функциональность для бизнеса:** Несмотря на наличие API, интеграция с коммерческими системами может требовать дополнительных усилий и настроек.

1. **MusicBrainz**

[**MusicBrainz**](https://musicbrainz.org/) - это проект с открытым исходным кодом, который создает базу данных для организации и хранения информации о музыке, включая данные о исполнителях, альбомах, песнях и записях. Платформа предоставляет API для интеграции с другими приложениями и системами.

* Плюсы:
  + **Открытая база данных:** MusicBrainz предоставляет бесплатный доступ к метаданным о музыке, что может быть полезно для музыкальных магазинов, которые хотят наполнить свою базу данных информацией о произведениях и исполнителях.
  + **Интеграция с другими сервисами:** API позволяет интегрировать MusicBrainz с различными системами учета и продаж, что упрощает автоматизацию процессов.
  + **Открытый исходный код:** Возможность модификации и адаптации платформы под конкретные нужды бизнеса.
* Минусы:
  + **Потребность в модификациях:** Для успешной интеграции с магазинами и системой учета потребуется дополнительная настройка и развитие.
  + **Ограниченная поддержка продаж:** Платформа ориентирована на метаданные и не включает специализированных функций для управления продажами музыкальных товаров.

3. **Vinyl Me, Please**

[**Vinyl Me, Please**](https://www.vinylmeplease.com/) - это подписной сервис для коллекционеров виниловых пластинок, который продает эксклюзивные и редкие винилы. Платформа фокусируется на продажах ограниченных изданий, что делает её интересной для магазинов, ориентированных на коллекционные товары.

* Плюсы:
  + **Эксклюзивные издания:** Платформа работает с уникальными музыкальными записями, что может быть полезно для магазинов, работающих с редкими винилами.
  + **Система подписки:** Пользователи могут подписаться на ежемесячные поставки пластинок, что дает стабильный поток продаж.
  + **Фокус на коллекционеров:** Отличный ресурс для магазинов, ориентированных на коллекционеров и поклонников винтажной музыки.
* Минусы:
  + **Ограниченный ассортимент:** Основной акцент сделан на винилы, что ограничивает возможности для магазинов, продающих широкий ассортимент музыкальных товаров.
  + **Не подходит для обычных покупок:** Система подписки может не быть полезной для магазинов, работающих с покупками по мере необходимости.

**4. Tidal и Spotify Marketplace**

[**Tidal**](https://www.tidal.com/) и [**Spotify**](https://www.spotify.com/) - популярные стриминговые сервисы, которые также предоставляют возможности для продажи физических товаров через их маркетплейсы. Эти платформы ориентированы на цифровую музыку, но интегрируют возможности для продажи физических носителей, таких как винилы и мерч.

* Плюсы:
  + **Большая аудитория:** благодаря большой пользовательской базе, эти сервисы предоставляют доступ к огромному числу потенциальных покупателей.
  + **Интеграция с музыкальными платформами:** Возможность интеграции с цифровыми сервисами для привлечения покупателей к физическим носителям.
  + **Персонализированные рекомендации:** ромогают продвигать товары через систему рекомендаций, ориентированную на музыкальные предпочтения пользователей.
* Минусы:
  + **Ограниченная поддержка физических товаров:** Эти сервисы ориентированы в первую очередь на цифровую музыку, а возможности для работы с физическими товарами ограничены.
  + **Конкуренция с крупными игроками:** Сервисы как Tidal и Spotify имеют сильную конкуренцию, и интеграция с ними может быть сложной для небольших музыкальных магазинов.

**1.3 Требования к программируемой системе**

|  |  |  |
| --- | --- | --- |
| № | Требование | Значение |
| 1 | Управление данными о музыкантах, произведениях, ансамблях, исполнениях, записях (CD/LP) | Система должна обеспечивать возможность создания, ввода, просмотра, изменения и удаления основных данных, связанных с сущностями предметной области. |
| 2 | Управление связями между сущностями | Система должна позволять устанавливать и изменять связи (например, кто входит в ансамбль, кто композитор произведения, какие исполнения на диске).инструментами, на которых они играют (добавление/удаление связи). |
| 3 | Обеспечение каскадных изменений при удалении данных | Система (через триггеры и/или внешние ключи ON DELETE CASCADE) должна автоматически удалять связанные данные в дочерних таблицах при удалении родительских записей (например, удалить треки записи при удалении самой записи). |
| 4 | Авторизация пользователей | Система должна требовать аутентификации пользователей для доступа к функциям управления данными. |
| 5 | Защищенное взаимодействие через API (включая JWT) | Бэкенд должен предоставлять защищенный REST API для взаимодействия с фронтендом, используя, например, JWT для аутентификации запросов. |
| 6 | Взаимодействие Frontend и Backend | Фронтенд на React должен взаимодействовать с бэкендом на Go через разработанный REST API для получения и отправки данных. |
| 7 | Реализация специализированной логики базы данных через пакет PL/SQL | Система должна выполнять специфические операции над данными (подсчет произведений ансамбля, поиск записей ансамбля, определение лидеров продаж, операции с данными записей и ансамблей) посредством вызова процедур и функций, инкапсулированных в пакете PL/SQL базы данных. |

1. **ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ**