МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Специализация 1–40 01 01 Программное обеспечение информационных технологий (программирование интернет – изданий)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

База данных «Площадка для музыкальных исполнителей»

Выполнил студент Валдайцев Александр Денисович

(Ф.И.О.)

Руководитель проекта асс., Копыток Д. В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: асс., Копыток Д. В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: асс., Копыток Д. В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2022

**Содержание**

[**ВВЕДЕНИЕ** 3](#_Toc121404689)

[**1. Аналитический обзор литературы и формирование требований** 4](#_Toc121404690)

[**1.1. Анализ прототипов** 4](#_Toc121404691)

[**1.1.1. BeatStars** 4](#_Toc121404692)

[**1.1.2. Maestric Music** 5](#_Toc121404693)

[**1.1.3. BeastBeat.ru** 5](#_Toc121404694)

[**1.1.4. Kwork.ru** 6](#_Toc121404695)

[**1.2. Требования к проекту** 7](#_Toc121404696)

[**2. Разработка программной системы** 8](#_Toc121404697)

[**2.1. Проектирование базы данных** 8](#_Toc121404698)

[**2.2. Хранимые процедуры** 12](#_Toc121404699)

[**2.3. Функции** 12](#_Toc121404700)

[**2.4. Представления** 13](#_Toc121404701)

[**2.5. Технологии базы данных** 13](#_Toc121404702)

[**2.6. Спецификация функциональных требований** 14](#_Toc121404703)

[**3. Обоснование технических приемов программирования** 15](#_Toc121404704)

[**4. Тестирование, экспериментальные исследования и анализ полученных результатов** 16](#_Toc121404705)

[**4.1. Ошибка при регистрации существующего логина** 16](#_Toc121404706)

[**4.2. Ошибка при некорректных данных для входа** 16](#_Toc121404707)

[**4.3. Ошибка при поиске несуществующей услуги** 17](#_Toc121404708)

[**Заключение** 18](#_Toc121404709)

[**Список литературы** 19](#_Toc121404710)

[**ПРИЛОЖЕНИЕ А** 20](#_Toc121404711)

## **ВВЕДЕНИЕ**

В современном мире многие люди постоянно прослушивают музыкальные композиции. И если с точки зрения слушателей существует множество площадок, сервисов и программных средств для прослушивания, то с точки зрения исполнителей выбор программных средств для централизованной последовательной записи и выпуска композиции сравнительно невелик.

Целью данной работы являлась разработка реляционной базы данных для площадки для музыкальных исполнителей. Эта база данных должна составлялась для обеспечения артистом доступа к функциям заказа услуг от поставщиков, а для самих поставщиков услуг — связи с артистами.

База данных — это организованная структура, предназначенная для хранения информации, систематизированная таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. Реляционная база данных — база данных, основанная на реляционной модели данных. В качестве СУБД для базы данных была выбрана Microsoft SQL Server, в связи с ее высокой производительностью и надежностью.

В основной части будут затронуты все аспекты разработки проекта и обоснованы некоторые технические приёмы, к которым приходилось прибегнуть, с целью реализации работы веб-сервера с базой данных.

Для обеспечения работы приложения использовались мультимедийные типы данных при хранении аудио файлов.

Основные требования к приложению:

* Возможность аутентификации артистов и поставщиков услуг.
* Составление расписания работы поставщика.
* Заказ услуги.
* Поиск по услугам.
* Просмотр корзины и статуса заказа.
* Просмотр выпущенных альбомов и композиций.

В пояснительной записке вы сможете найти краткую информацию о похожих продуктах, архитектуре, реализации проекта, руководстве пользователя.

## **Аналитический обзор литературы и формирование требований**

## **Анализ прототипов**

Были проанализированы цели и задачи, поставленные в данном курсовом проекте, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

# **BeatStars**

Самое популярное альтернативное решение — интернет-ресурс «BeatStars», являющийся самым известным в мире пространством, собравшим полноценную информацию о музыке. Главными преимуществами данного ресурса является его наполненность информацией и широкая известность. Практически все музыканты и поставщики услуг по написанию и сведению музыки размещают свои заказы и услуги именно на этом сервисе. Пользователь данного ресурса имеет возможность детально и подробно ознакомиться с каждым поставщиком услуг. Ещё одним немаловажным достоинством данного программного средства является возможность пользователей выпустить свою музыку на стриминговые площадки, а также сортировка по жанрам, плейлистам трекам, исполнителям, ключевым словам. Данное программное средство предоставляет пользователю возможность по записи, сведению и публикации своих музыкальных композиций.

Из выявленных недостатков данного ресурса необходимо выделить отсутствие возможности предоставления услуги сведения композиции.

Интерфейс интернет-ресурса представлен на рисунке 1.1.

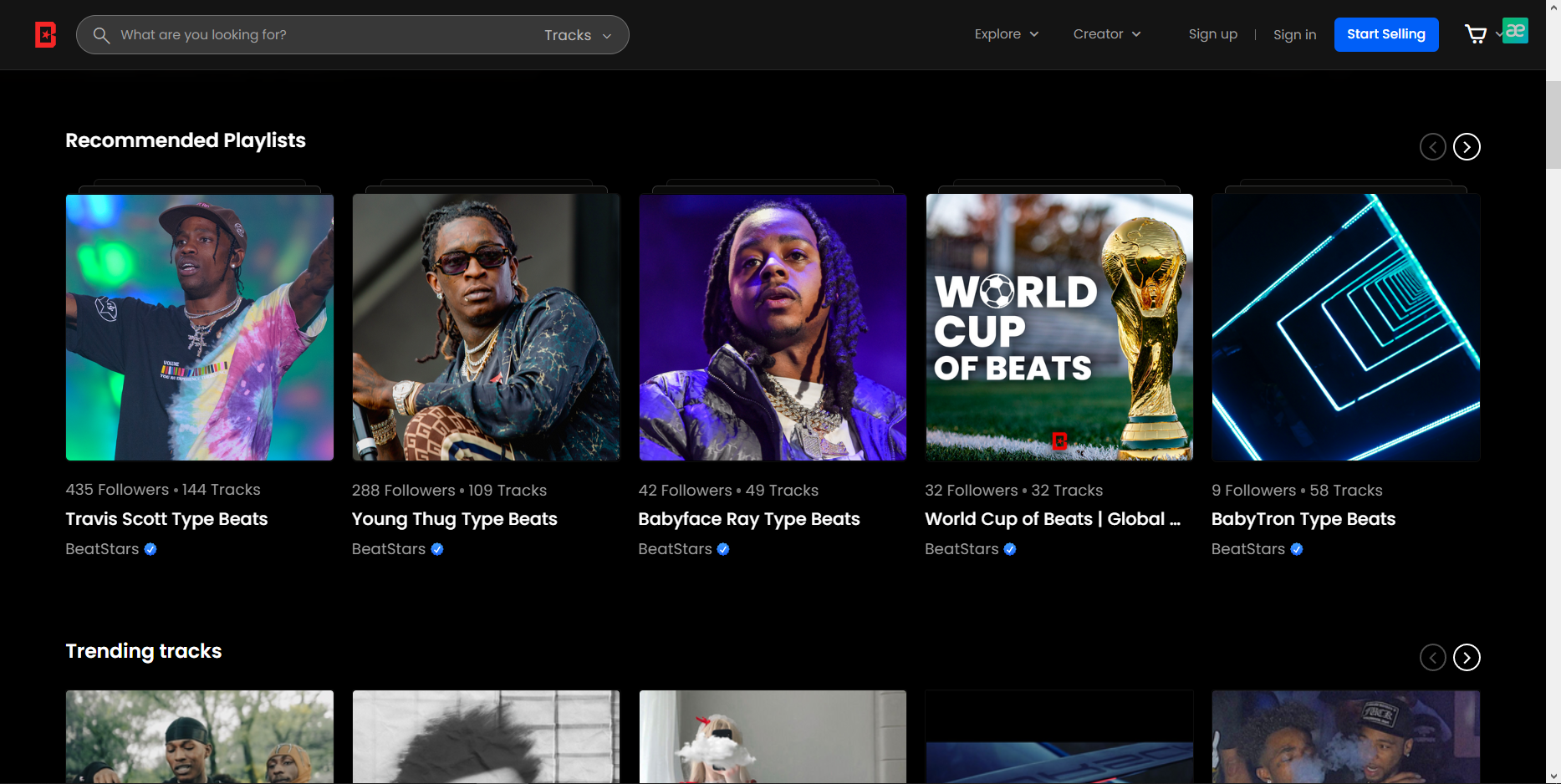


Рисунок 1.1 – Интернет-ресурс «BeatStars»

# **Maestric Music**

Еще одним альтернативным решением задач, поставленных в данном курсовом проекте, является интернет-ресурс «Maestric Music». С помощью данного программного средства пользователь может загрузить музыкальную композицию или альбом на стриминговые площадки, подписаться на лейбл, централизованно получать выплаты за стриминг своих композиций, получить консультацию по публикации своих композиций.

Одним из главных недостатков выбора такой альтернативы является отсутствие возможности заказа услуги по самому созданию композиции, а также невозможность для поставщика услуг разместить свои услуги. Данное альтернативное решение является узконаправленным и неявно предоставляет услуги по публикации композиций и альбомов.

Интерфейс интернет-ресурса представлен на рисунке 1.2.



Рисунок 1.2 – Интернет-ресурс «Maestric Music»

# **BeastBeat.ru**

Ещё одной альтернативой решения поставленных задач является интернет-ресурс «BeatBeat.ru». С помощью данного программного средства пользователь может ознакомиться со списком доступных для заказа битов, выбрать бит для заказа, просмотреть плейлисты и популярных авторов битов. Немаловажным преимуществом является то, что сервис предоставляет возможность как покупать, так и продавать биты.

Однако, недостатками данной альтернативы являются узкая направленность только на продажу битов и отсутствие другого функционала.

Интерфейс интернет-ресурса представлен на рисунке 1.3.

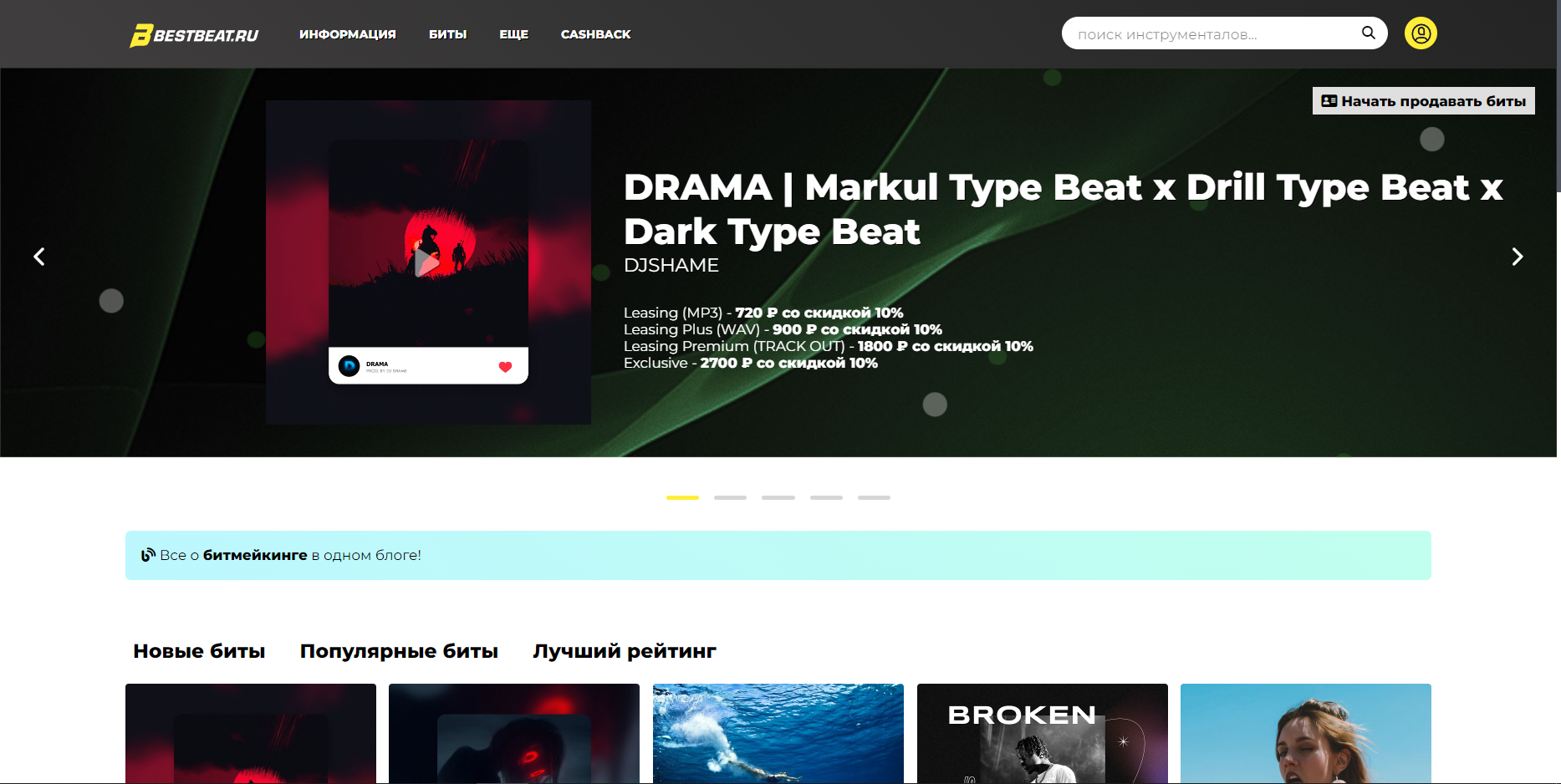


Рисунок 1.3 – Интернет-ресурс «BeatBeat.ru»

# **Kwork.ru**

В список рассматриваемых альтернатив решения поставленных задач также можно включить интернет-ресурс «Kwork.ru». Данное программное средство предоставляет возможность найти практически любые услуги, в том числе и любые услуги по созданию музыки. Это и является важнейшим преимуществом, так как можно найти услуги по сведению, мастерингу, записи, созданию треков, а также по покупке и продаже битов.

Недостатками данного программного средства является отсутствие возможности публикации альбома и отсутствие связи с лейблами, так как услуги предоставляются поставщиками услуг напрямую.

Интерфейс интернет-ресурса представлен на рисунке 1.4.



Рисунок 1.4 – Интернет-ресурс «Kwork.ru»

## **Требования к проекту**

Обзор вышеперечисленных известных аналогов позволяет проанализировать все преимущества и недостатки альтернативных возможностей и позволяет сформулировать список требований, предъявляемых к программному средству, разрабатываемому в данном курсовом проекте. Программное средство должно обеспечивать возможность выполнения перечисленных ниже функций:

* Регистрация и авторизация пользователя;
* Поиск услуги;
* Оформление заказа;
* Уведомление пользователя о заказе;
* Просмотр корзины;
* Просмотр статуса заказа;

## **Разработка программной системы**

# **Проектирование базы данных**

Процесс проектирования базы данных состоит из следующих этапов:

* сбор информации;
* определение сущностей;
* определение атрибутов для каждой сущности;
* определение связей между сущностями;
* нормализация.

На этапе сбора информации вам необходимо точно определить, как будет использоваться база данных, и какая информация будет в ней храниться.

Далее следует этап определения сущностей и на нем определяются сущности, из которых будет состоять база данных.

Сущность — это объект в базе данных, в котором хранятся данные. Сущность может представлять собой нечто вещественное (дом, человек, предмет, место) или абстрактное (банковская операция, отдел компании, маршрут автобуса). В физической модели сущность называется таблицей. Сущности состоят из атрибутов (столбцов таблицы) и записей (строк в таблице).

Обычно базы данных состоят из нескольких основных сущностей, связанных с большим количеством подчиненных сущностей. Основные сущности называются независимыми: они не зависят ни от какой-либо другой сущности. Подчиненные сущности называются зависимыми: для того, чтобы существовала одна из них, должна существовать связанная с ней основная таблица.

Следующий этап — это определение атрибутов созданных сущностей. Атрибут представляет свойство, описывающее сущность. Атрибуты часто бывают числом, датой или текстом. Все данные, хранящиеся в атрибуте, должны иметь одинаковый тип и обладать одинаковыми свойствами.

В физической модели атрибуты называют колонками. После определения сущностей необходимо определить все атрибуты этих сущностей. На диаграммах атрибуты обычно перечисляются внутри прямоугольника сущности.

На этапе определения связей работа происходит с ключами сущностей. Ключом называется набор атрибутов, однозначно определяющий запись. Ключи делятся на два класса: простые и составные.

Простой ключ состоит только из одного атрибута. Например, в базе «Паспорта граждан страны» номер паспорта будет простым ключом: ведь не бывает двух паспортов с одинаковым номером.

Составной ключ состоит из нескольких атрибутов. В той же базе «Паспорта граждан страны» может быть составной ключ со следующими атрибутами:  
фамилия, имя, отчество, дата рождения. Это — как пример, т. к. этот составной ключ, теоретически, не обеспечивает гарантированной уникальности записи.

Первичным ключом называется совокупность атрибутов, однозначно идентифицирующих запись в таблице (сущности). Один из возможных ключей становится первичным ключом. На диаграммах первичные ключи часто изображаются выше основного списка атрибутов или выделяются специальными символами.

Любой возможный ключ, не являющийся первичным, называется альтернативным ключом. Сущность может иметь несколько альтернативных ключей. Внешним ключом называется совокупность атрибутов, ссылающихся на первичный или альтернативный ключ другой сущности. Если внешний ключ не связан с первичной сущностью, то он может содержать только неопределенные значения. Если при этом ключ является составным, то все атрибуты внешнего ключа должны быть неопределенными. На диаграммах атрибуты, объединяемые во внешние ключи, обозначаются специальными символами.

Реляционные базы данных позволяют объединять информацию, принадлежащую разным сущностям. Отношение — это ситуация, при которой одна сущность ссылается на первичный ключ второй сущности. Они определяются в процессе проектирования базы. Для этого следует проанализировать сущности и выявить логические связи, существующие между ними.

Тип отношения определяет количество записей сущности, связанных с записью другой сущности. Отношения делятся на три основных типа:

* Один-к-одному. Каждой записи первой сущности соответствует только одна запись из второй сущности. А каждой записи второй сущности соответствует только одна запись из первой сущности. Например, есть две сущности: Люди и Свидетельства о рождении. И у одного человека может быть только одно свидетельство о рождении.
* Один-ко-многим. Каждой записи первой сущности могут соответствовать несколько записей из второй сущности. Однако каждой записи второй сущности соответствует только одна запись из первой сущности. Например, есть две сущности: Заказ и Позиция заказа. И в одном заказе может быть много товаров.
* Многие-ко-многим. Каждой записи первой сущности могут соответствовать несколько записей из второй сущности. Однако и каждой записи второй сущности может соответствовать несколько записей из первой сущности. Например, есть две сущности: Автор и Книга. Один автор может написать много книг. Но у книги может быть несколько авторов.

По критерию обязательности отношения делятся на обязательные и необязательные.

* Обязательное отношение означает, что для каждой записи из первой сущности непременно должны присутствовать связанные записи во второй сущности.
* Необязательное отношение означает, что для записи из первой сущности может и не существовать записи во второй сущности.

Далее следует этап нормализации. Нормализацией называется процесс удаления избыточных данных из базы данных. Каждый элемент данных должен храниться в базе в одном и только в одном экземпляре. Существует пять распространенных форм нормализации. Как правило, база данных приводится к третьей нормальной форме.

В процессе нормализации выполняются определенные действия по удалению избыточных данных. Нормализация повышает быстродействие, ускоряет сортировку и построение индекса, уменьшает количество индексов на сущность, ускоряет операции вставки и обновления.

Нормализованная база данных обычно отличается большей гибкостью. При модификации запросов или сохраняемых данных в нормализованную базу обычно приходится вносить меньше изменений, а внесение изменений имеет меньше последствий.

Чтобы преобразовать сущность в первую нормальную форму, следует исключить повторяющиеся группы значений и добиться того, чтобы каждый атрибут содержал только одно значение, списки значений не допускаются.  
Другими словами, каждый атрибут, в сущности, должен храниться только в одном экземпляре.

Для соответствия второй нормальной форме сущности должны быть в первой нормальной форме. Таблица во второй нормальной форме содержит только те данные, которые к ней относятся. Значения не ключевых атрибутов сущности зависят от первичного ключа. Если более точно, то атрибуты зависят от первичного ключа, от всего первичного ключа и только от первичного ключа.

В третьей нормальной форме исключаются атрибуты, не зависящие от всего ключа. Любая сущность, находящаяся в третьей нормальной форме, находится также и во второй. Это самая распространенная форма базы данных.  
В третьей нормальной форме каждый атрибут зависит от ключа, от всего ключа и ни от чего, кроме ключа.

Ограничения — это правила, за соблюдением которых следит система управления базы данных. Ограничения определяют множество значений, которые можно вводить в столбец или столбцы.

Организовав данные в таблицы и определив связи между ними, можно считать, что была создана модель, правильным образом отражающая бизнес-среду. Теперь нужно обеспечить, чтобы данные, вводимые в базу, давали правильное представление о состоянии дела. Иными словами, нужно обеспечить выполнение деловых правил и поддержку целостности базы данных.

Например, ваша компания занимается доставкой книг. Вы вряд ли примете заказ от неизвестного клиента, ведь тогда вы даже не сможете доставить заказ. Отсюда бизнес-правило: заказы принимаются только от клиентов, информация о которых есть в базе данных.

Корректность данных в реляционных базах обеспечивается набором правил. Правила целостности данных делятся на четыре категории.

Для реализации необходимого функционала была создана база данных, таблицы которой можно разделить на 3 логические группы: таблицы для хранения информации о продуктах, таблицы для хранения информации о заказах пользователей и таблицы для хранения пользовательских данных.

Рассмотрим таблицы базы данных на примере диаграммы, представленной на рисунке 2.1.

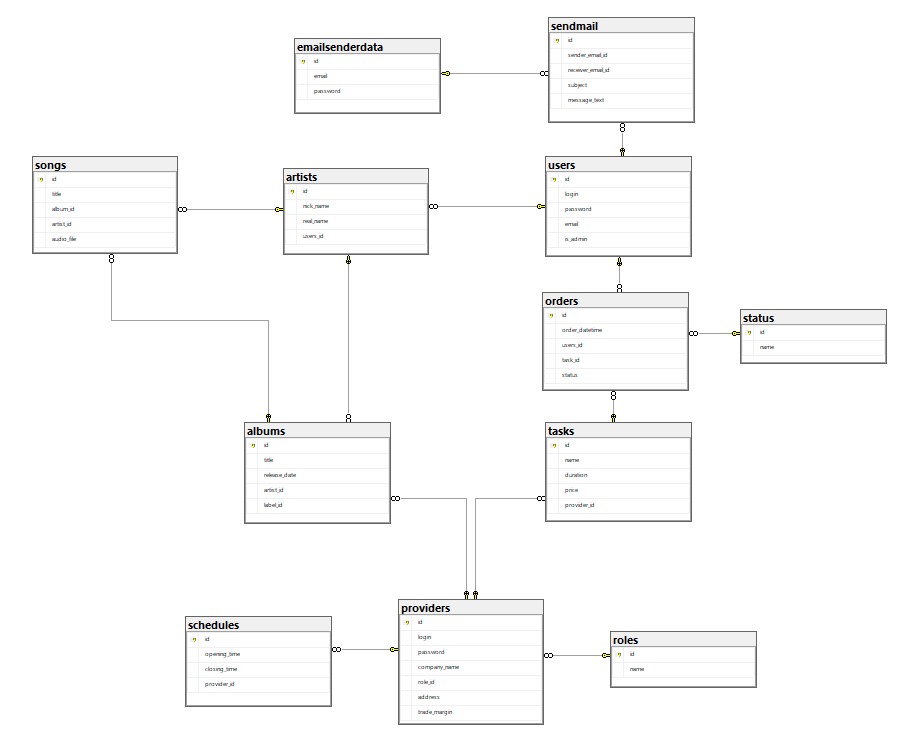


Рисунок 2.1 – Диаграмма таблиц базы данных

База данных состоит из 12 таблиц, связанных между собой внешними ключами.

Таблица «*STATUS*» хранит информацию о возможных статусах заказа.

Таблица «*ROLES*» хранит информацию о ролях, которые могут назначаться поставщиках услуг: битмейкер, мастеринг, лейбл, музыкант и так далее.

Таблица «*USERS*» хранит в себе всю информацию о пользователях, то есть музыкантах, включая пароль, логин, электронную почту и т.д.

Таблица «*PROVIDERS*» хранит информацию о поставщиках услуг аналогично таблице «*USERS*».

Таблица «*ARTISTS*» хранит информацию о артистах, которые могут выпускать треки и альбомы, и может быть связана с таблицей «*USERS*».

Таблица «*ALBUMS*» хранит информацию о выпущенных лейблами альбомах и может быть связана внешним ключом с таблицей «*PROVIDERS*».

Таблица «*SONGS*» содержит информацию о выпущенных артистами песнях, которые могут располагаться в определенном альбоме.

Таблица «*TASKS*» совмещает в себе информацию о вариантах услуг, которые предоставляют поставщики. Содержит информацию о длительностях и стоимости услуг определенного провайдера.

Таблица «*SCHEDULES*» хранит информацию о расписаниях поставщиков, их времени открытия и закрытия, в которые можно разместить заказы.

Таблица «*ORDERS*» хранит информацию о заказах, размещенными пользователями и обработанными провайдерами.

Таблица «*EMAILSENDERDATA*» хранит информацию о данных для входа в аккаунт электронной почты для отправки сообщений, уведомляющих пользователя о заказе.

Таблица «*SENDMAIL*» хранит информацию о самом сообщении, которое будет отправляться пользователю, его электронной почте, тексте и теме сообщения.

# **Хранимые процедуры**

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

Наиболее используемые процедуры представлены в приложении Б.

При разработке курсового проекта было создано большое количество процедур для следующих целей:

Для пользователя:

* регистрация;
* авторизация;
* поиск услуг;
* записи песни;
* размещения заказа;
* оплаты заказа;
* получения списка заказов в корзине;
* получения статуса заказа;

Для поставщика услуг:

* авторизация;
* просмотр фильмов;
* получения статуса заказа;
* создания альбома;
* получения списка задач;
* подтверждения заказа пользователя.

# **Функции**

Функция SQL — это набор операторов SQL, которые принимают входные данные и выполняют с ними действия SQL, а затем возвращают результаты в виде выходных данных.

Функции выполняют утилитарные дополнительные функции и служат для упрощенной выборки некоторых часто получаемых данных, так как они позволяют возможность удобно получить некоторые данные:

* Заказы поставщика на определенный день;
* Время открытия провайдера;
* Заказы на этот день;
* Длительность задачи по ее названию;
* Идентификатор задачи по ее названию;
* Цена заказа по его идентификатору;
* Цена заказа с учетом надбавки провайдера.

# **Представления**

Представление - это виртуальная таблица, основанная на результирующем наборе инструкции SQL.

Они содержит строки и столбцы, как и настоящая таблица. Поля в представлении - это поля из одной или нескольких реальных таблиц в базе данных.

Представления используются для абстракции от соединения таблиц и сокрытия их реализации внутри доступных пользователям процедур.

* Поставщики услуг и их задачи;
* Пользователи и их заказы.

# **Технологии базы данных**

В данном подразделе будут описаны технологии, использованные в процессе разработки приложения. В программном средстве была реализована технология мультимедийного хранения данных. Далее будет описана схема работы технологии.

Для работы с данными большого объема СУБД Microsoft SQL Server предоставляет тип данных VARBINARY. В данном курсовом проекте технология мультимедийных типов данных отражена в необходимости хранить аудио композиции, выпущенные с помощью исполнителя и поставщика услуг. Речь идет о таблице «*SONGS*», а конкретно об ее столбце audio\_file. Тип VARBINARY представляет из себя массив двоичных данных, поэтому процесс сводился к разбиению мультимедийных файлов на массив битов при загрузке их в базу данных, и к сборке аудиофайла из массива битов на обратном пути.

Для реализации технологии необходимо использовать OPENROWSET, что представлено на рисунке 2.2:

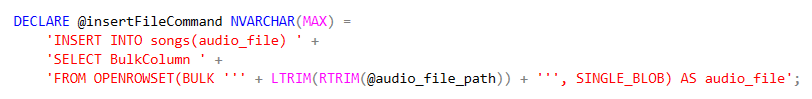


Рисунок 2.2 – Использование OPENROWSET

Далее, в процедуре recordSong, если пользователь передал в качестве параметра путь к файлу аудио композиции, то данный скрипт выполняется и данные в двоичном виде записываются в базу данных. Выполнение скрипта отображено на рисунке 2.3.

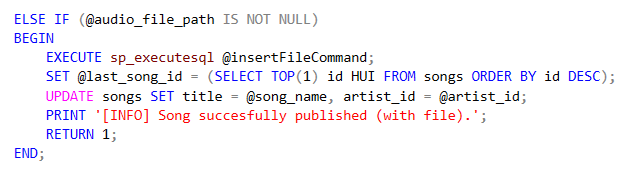


Рисунок 2.3 – Выполнение скрипта

# **Спецификация функциональных требований**

В программном средстве необходимо реализовать регистрацию и авторизацию пользователей для доступа ко всем возможностям приложения. Для авторизации входными параметрами являются логин или почта, и пароль пользователя. Для регистрации входными данными являются имя, фамилия, логин, пароль, электронная почта. Для поставщика также при регистрации вводятся название компании адрес и торговая надбавка.

Пользователь имеет возможность регистрироваться и входить в аккаунт, искать услуги, просматривать свою корзину и статусы заказов, размещать и оплачивать заказы, выпускать песни и альбомы, получать уведомление о заказе.

Поставщик услуг имеет возможность регистрироваться и аутентифицироваться, подтверждать заказ пользователя, составлять свое расписание, просматривать и редактировать свои задачи, выпускать альбом и песни пользователя.

## **Обоснование технических приемов программирования**

При разработке программного средства в качестве сервера и системы управления базы данных были использованы сервер Microsoft SQL Server Express и СУБД Microsoft SQL Server Management Studio 18.

Данный выбор СУБД был обоснован требованиями к программному средству и возможностями сервера базы данных. Microsoft SQL Server редакции Express предоставляет встроенную возможность отправки сообщений по электронной почте по протоколу SMTP. Таким образом, есть возможность абстрагироваться и инкапсулировать отправку электронных сообщений в базе данных, предоставляя соответствующей процедуре только входные данные с темой и текстом сообщения.

Для отправки электронных сообщений в СУБД Microsoft SQL Server Management Studio 18 необходимо предварительно настроить почтового агента с помощью T-SQL команд или явно через конфигурацию в SSMS.

Технология отправки сообщений в Microsoft SQL Server отображена на рисунке 3.1.

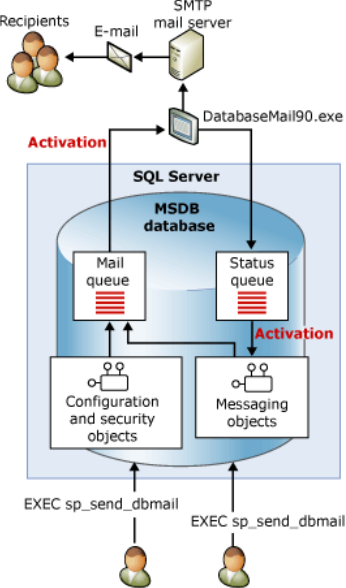


Рисунок 3.1 – Технология отправки сообщений

## **Тестирование, экспериментальные исследования и анализ полученных результатов**

Тестирование отыгрывает важную роль при разработке любого программного продукта. Чем качественнее тестирование, тем лучше в итоге должен выйти конечный продукт. Часто можно столкнуться с ситуацией, когда тестирование программного кода проходит очень кропотливо, а на тестирование базы данных времени уже не остается либо оно делается по остаточному принципу.

Соответственно, для исключения такой ситуации в данном разделе будет описано тестирование базы данных.

# **Ошибка при регистрации существующего логина**

При регистрации пользователя может возникнуть ситуация, при которой данный логин или же электронная почта уже существует в базе данных. Ошибка отображена на рисунке 4.1.

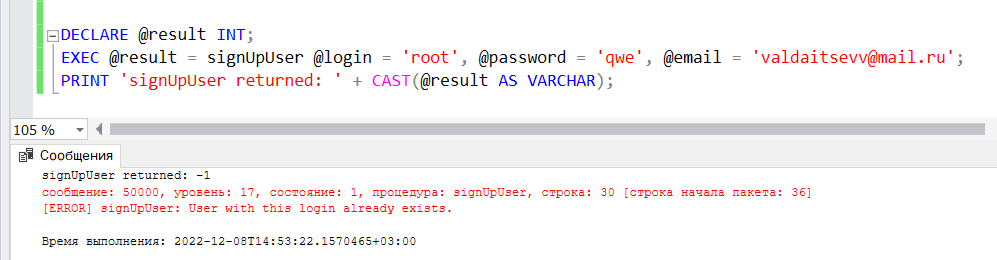


Рисунок 4.1 – Ошибка при регистрации существующего логина

# **Ошибка** **при некорректных данных для входа**

При предоставлении некорректных данных для входа (несуществующий логин или почта, неверный пароль) также должна возвращаться ошибка и сообщение. Ошибка отображена на рисунке 4.2.

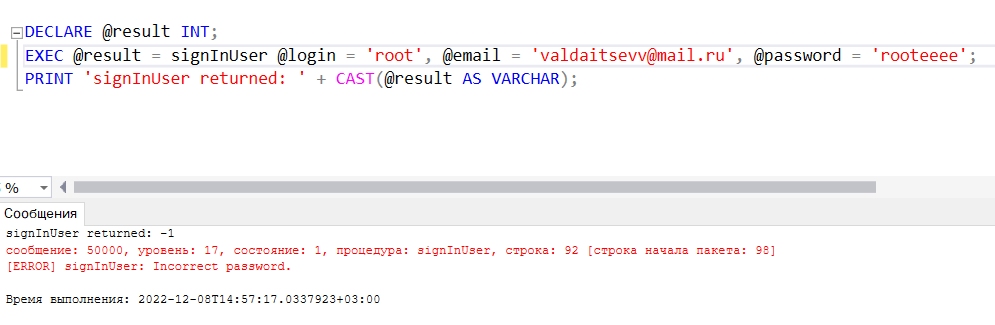


Рисунок 4.2 – Ошибка при некорректных данных для входа

# **Ошибка при поиске несуществующей услуги**

При поиске услуги, которой не существует ни у одного провайдера, должна возвращаться ошибка и выводиться сообщение. Ошибка отображена на рисунке 4.3.

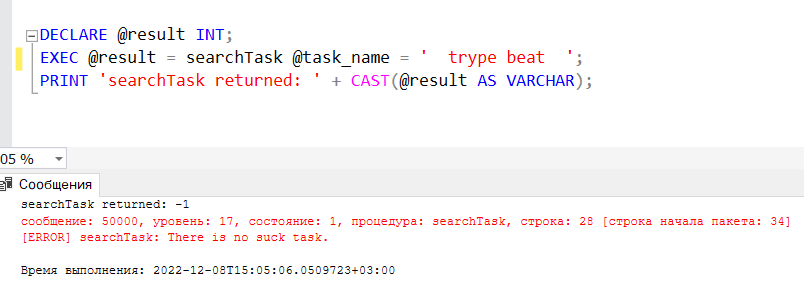


Рисунок 4.3 – Ошибка при поиске несуществующей услуги

# **Заключение**

Результатом выполнения курсового проекта стала база данных в базе данные SQL Server в СУБД SQL Server Management Studio 18.

Были реализованы основные требования, указанные в листе задания вместе со следующими пунктами:

* Регистрация и авторизация пользователя;
* Поиск услуги;
* Оформление заказа;
* Уведомление пользователя о заказе;
* Просмотр корзины;
* Просмотр статуса заказа;
* Импорт и экспорт данных таблицы в XML формат;
* Заполнение таблицы на 100 000 записей;
* Технология мультимедийного хранения данных.

Проект не может окончательно претендовать на полноценный коммерческий продукт, поскольку для демонстрации технологии и работы с базой данных были опущены некоторые атрибуты современных площадок, реализация которых не относится к основной теме курсового проекта (например, платежные системы).

Протестировав все компоненты приложения, можно прийти к заключению, что все основные требования выполнены и приложение работает исправно.

# **Список литературы**

1. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com – Дата доступа: 15.10.2021.

2. DBMS\_XMLDOM [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/cd/B1930601/appdev.102/b14258/dxmldom.htm#i1076719> – Дата доступа: 20.10.2022.

3. Developing and Using Stored Procedures [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/cd/B2835901/appdev.111/b28843/tdddgprocedures.htm> – Дата доступа: 26.10.2022.

4. Stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com – Дата доступа: 08.12.2022

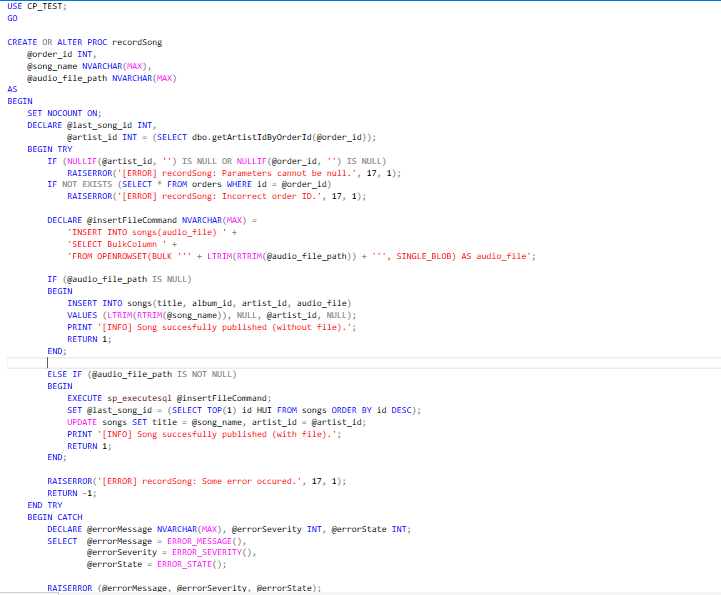
5. Блинова, Е.А. Курс лекций по Базам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.

# **ПРИЛОЖЕНИЕ А**

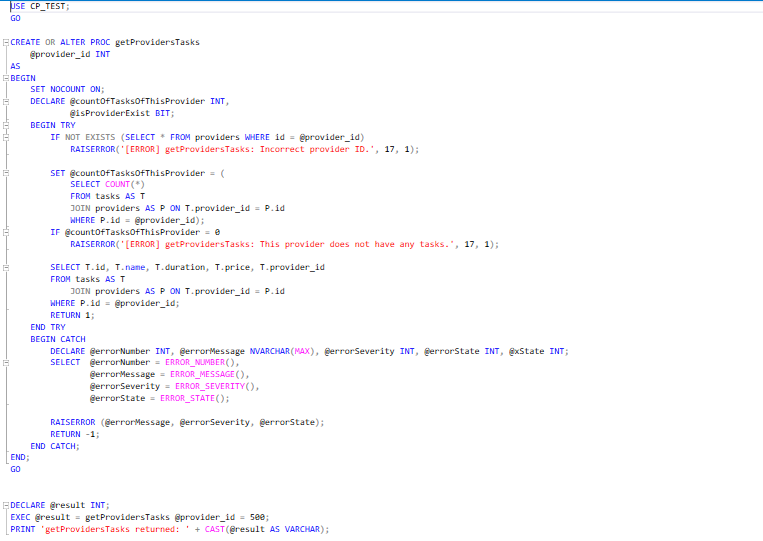
Процедура searchTask



Процедура recordSong



Процедура getProvidersTasks



Процедура payForAnorder

