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

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

«Реализация базы данных лейбла метал – музыки «Dalvhikki Records» применением технологии полнотекстового поиска в БД»

Выполнил студент Горощеня Владислав Сергеевич

(Ф.И.О.)

Руководитель работы асс. Нистюк О.А.

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

И.о. зав. кафедрой ст. преп. Блинова Е.А.

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

Курсовая работа защищена с оценкой

Минск 2023

# Содержание

[Введение 3](#_Toc153925076)

[1 Постановка задачи 4](#_Toc153925077)

[1.1 Обзор аналогичных решений 4](#_Toc153925078)

[1.2 Спецификация требований 4](#_Toc153925079)

[2 Разработка объектов базы данных 6](#_Toc153925080)

[2.1 Проектирование модели базы данных 6](#_Toc153925081)

[3 Разработка объектов базы данных 9](#_Toc153925082)

[3.1 Разработка таблиц 9](#_Toc153925083)

[3.2 Разработка хранимых процедур 9](#_Toc153925084)

[3.3 Разработка функций 10](#_Toc153925085)

[3.4 Разработка индексов 11](#_Toc153925086)

[3.5 Разработка триггеров 11](#_Toc153925087)

[4 Описание процедур импорта и экспорта 12](#_Toc153925088)

[5 Тестирование производительности 14](#_Toc153925089)

[6 Описание технологии и её применения в базе данных 15](#_Toc153925090)

[7 Краткое описание приложения для демонстрации 17](#_Toc153925091)

[8 Руководство пользователя 18](#_Toc153925092)

[Заключение 25](#_Toc153925093)

[Список используемых источников 26](#_Toc153925094)

[Приложение А 27](#_Toc153925095)

[Приложение Б 29](#_Toc153925096)

[Приложение В 38](#_Toc153925097)

[Приложение Г 39](#_Toc153925098)

# Введение

Метал-музыка – это одно из самых известных направлений экстремальной му­зыки, которое было популярно как раньше, так и сейчас. Появляется всё большее количество коллективов и исполнителей, которые желают внести свой вклад в её развитие. Но для этого им необходимы ресурсы для того, чтобы заявить о себе в индустрии. И в большинстве своём, эти ресурсы предоставляют лейблы – бренды, производящие и распространяющие аудиозаписи, а также помогающие музыкантам, заключившим с ними контракт, в продвижении их продукта.

Целью данного курсового проекта – разработка реляционной базы данных для лейбла метал – музыки «Dalvhikki Records». База данных представляет собой организационную структуру, предназначенную для хранения информации, которая должна быть полезной, полной, точной, непротиворечивой и актуальной. Самой широко распространённой моделью базы данных является реляционная, основанная на теории множеств. В качестве СУБД для её разработки была выбрана Oracle 23c ввиду своей высокой производительности и надёжности.

Для демонстрации работы базы данных необходимо разработать приложение, взаимодействующее с ней. Приложение реализовано с помощью языка программирования Java с использованием технологии Java Servlet Pages.

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.

Основные требования к приложению:

* реализация ролей администратор, пользователь, гость;
* поиск по группам, альбомам, песням;
* добавление, удаление изменение групп, альбомов, песен администратором;
* предоставление популярных песен, новинок;
* предоставление прослушивания музыки.

# Постановка задачи

## 1.1 Обзор аналогичных решений

К первому аналогичному решению можно отнести приложение известного американского лейбла Metal Blade Records. Оно хранит информацию о группах, которые подписаны на лейбл, список песен и альбомов группы, а также расписание их туров. Также приложение предоставляет новости из мира метал – музыки и событиях внутри групп. К достоинствам данного аналога можно отнести подробное описание информации о группах, таких как их биография и список ближайших туров. Имеется свой мини – магазин с винилами и мерчем музыкальных групп. Помимо этого, просматривать не только те группы, что на данный момент выступают на лейбле, но и те, что были подписаны на него раньше с помощью механизма фильтрации. Механизм фильтрации также поддерживается и для релизов, что значительно упрощает поиск нужного пользователю релиза. Также к достоинствам можно отнести наличие возможности прослушивания музыки, которое сделано через доступ к музыкальным платформам iTunes и SoundCloud. К недостаткам можно отнести ухудшение производительности при фильтрации релизов по годам, заметно тормозящее работу приложения, и невозможность перейти к магазину для европейских стран.

В качестве второго аналога рассматривалось приложение немецкого лейбла Massacre Records. В данном приложении предоставляются списки вышедших альбомов и альбомов, которые будут выпущены в ближайшее время, а также список групп вместе со списком выпущенных ими альбомов. К достоинствам данного приложения можно отнести возможность прослушивания музыку без доступа к различным музыкальным платформам, наличие чарта, позволяющим анализировать популярность альбомов и групп в целом в различных странах мира и возможность фильтрации альбомов по годам и месяцам их релиза. В качестве недостатков следует выделить ухудшение производительности приложения при просмотре списка групп, не позволяющее в некоторых случаях просмотреть о них информацию, а также невозможность просмотреть информацию обо всех альбомах группы, а только лишь последнего релиза.

## 1.2 Спецификация требований

База данных должна быть реализована в СУБД Oracle, подключение к которой следует осуществлять при помощи одного из перечисленных специальных пользователей: администратор, авторизованный пользователь, гость. Доступ к данным осуществляется с помощью процедур, права на выполнение которых должны быть выданы пользователям.

Необходимо реализовать импорт и экспорт данных из XML протестировать производительность базы данных при вставке в неё 100000 строк и проанализировать планы запросов, а также применить технологию полнотекстового поиска в базах данных.

Помимо этого, на основании рассмотренных аналогичных решений, можно сформулировать следующие требования к разрабатываемому программному продукту:

* Возможность добавления песни в избранное для анализа популярности исполнителей;
* Возможность прослушивания музыки;
* Возможность фильтрации песен, альбомов, группы по их названию.

Диаграмма вариантов использования приведена на рисунке 1.1.

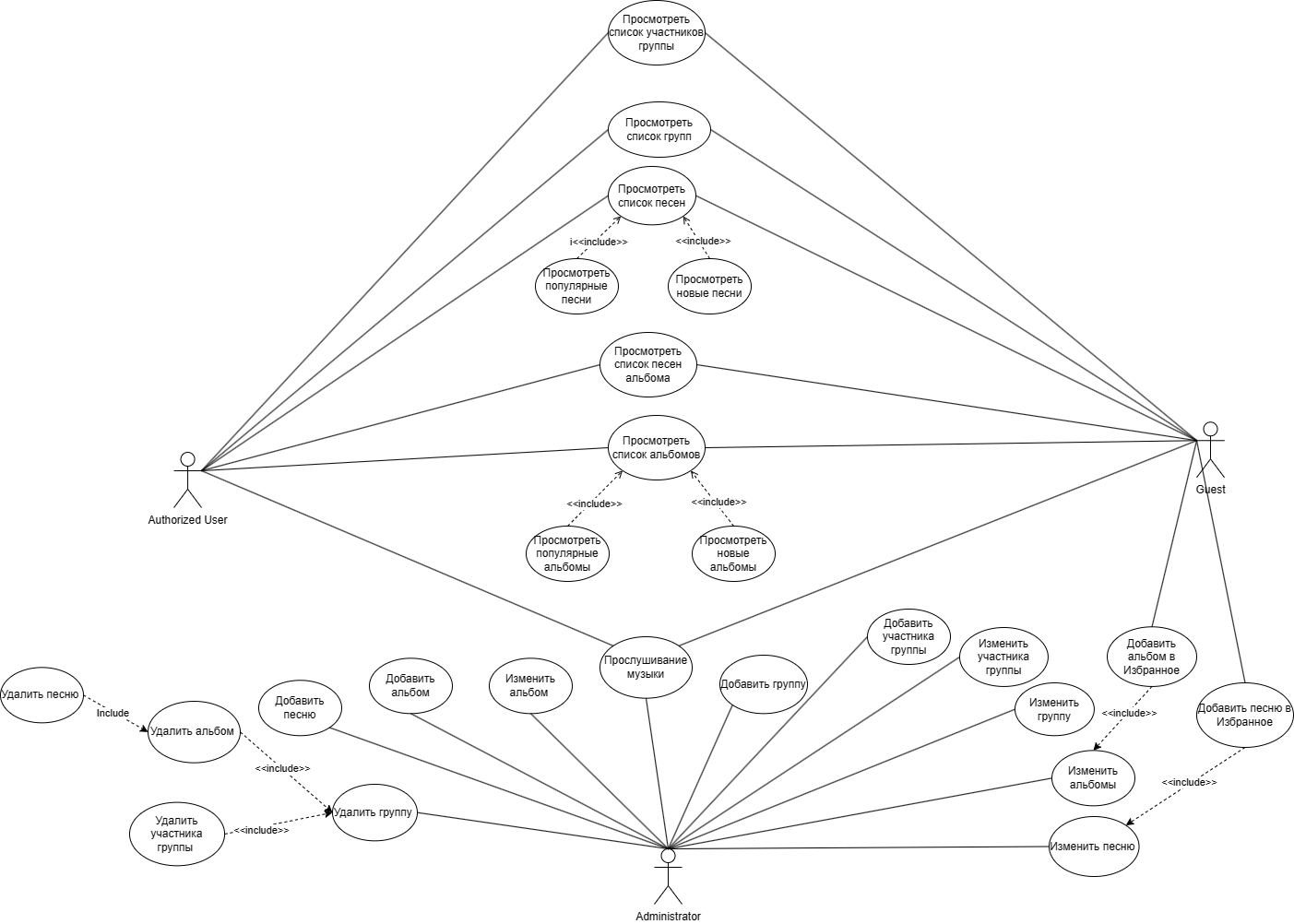


Рисунок 1.1 – диаграмма вариантов использования базы данных

Все пользователи, включая гостей, имеют право просматривать списки групп, песен, альбомов и прослушивать музыку.

Пользователь, помимо этого, может добавлять понравившиеся ему песни и альбомы в Избранное.

Администратору присвоена привилегия добавлять, изменять, удалять группы и их участников, песни и альбомы.

2 Разработка объектов базы данных

## 2.1 Проектирование модели базы данных

Логическая схема БД приведена на рисунке 2.1.

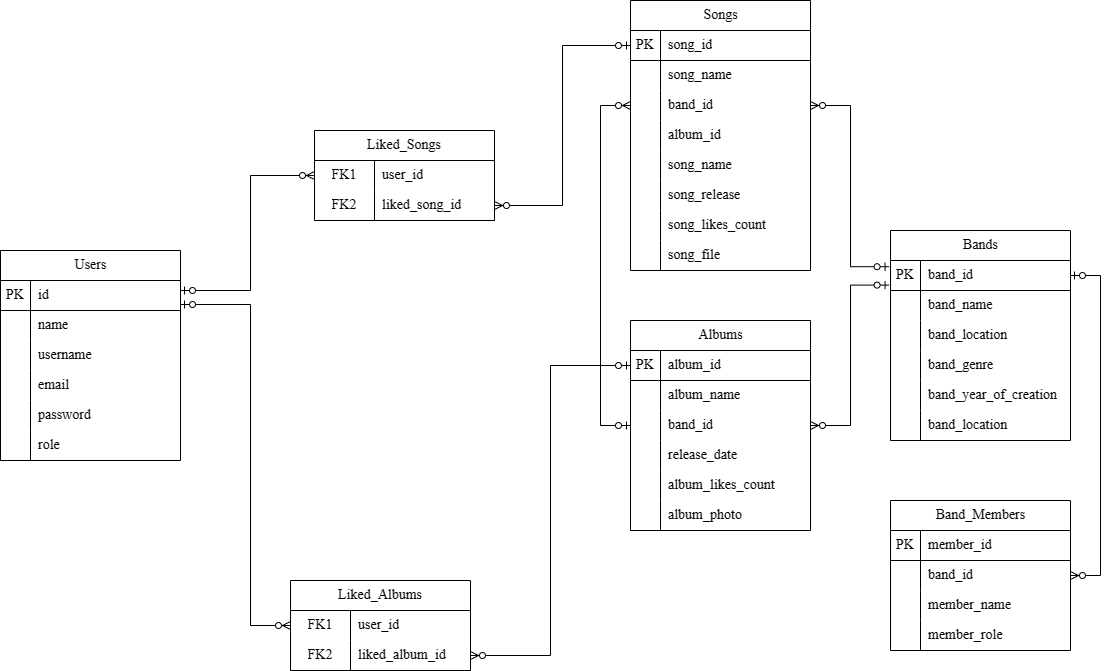


Рисунок 2.1 – логическая схема базы данных

Она включает в себя 7 таблиц: USERS, BANDS, ALBUMS, BAND\_MEMBERS, SONGS, LIKED\_SONGS, LIKED\_ALBUMS.

Таблица USERS хранит информацию о пользователях базы данных. Описание её столбцов представлено в таблице 2.1.

Таблица 2.1 – Описание столбцов таблицы USERS

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | number | Идентификатор пользователя, первичный ключ |
| name | varchar2(30) | Имя пользователя |
| email | varchar2(30) | Электронная почта пользователя |
| password | varchar2(32) | Пароль пользователя |
| role | varchar2(20) | Роль пользователя в БД, принимает значения «user» или «admin» |

Таблица BANDS хранит информацию о музыкальных группах лейбла. Её описание приведено в таблице 2.2

Таблица 2.2 – Описание столбцов таблицы BANDS

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| band\_id | number | Идентификатор группы, первичный ключ |
| band\_name | varchar2(50) | Имя группы |
| band\_location | varchar2(50) | Место дислокации группы |
| band\_genre | varchar2(50) | Направление, в котором играет группа |
| band\_year\_of\_creation | number | Год основания группы |
| band\_logo | varchar2(1000) | Логотип группы |

Таблица ALBUMS содержит информацию об альбомах, выпущенных группами на лейбле. Описание её столбцов представлено в таблице 2.3.

Таблица 2.3 – Описание столбцов таблицы ALBUMS

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| album\_id | number | Идентификатор альбома, первичный ключ |
| album\_name | varchar2(50) | Имя альбома |
| band\_id | number | Идентификатор группы, ассоциированного с этим альбомом, вторичный ключ |
| release\_date | date | Дата выхода альбома |
| album\_likes\_count | number | Количество добавлений в Избранное альбома пользователями |
| album\_photo | varchar2(1000) | Логотип альбома |

Таблица SONGS содержит информацию о песнях, выпущенных на альбомах группы лейбла. Описание её столбцов приведено в таблице 2.4.

Таблица 2.4 – Описание столбцов таблицы SONGS

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| song\_id | number | Идентификатор песни, первичный ключ |
| band\_id | number | Идентификатор группы, ассоциированный с этой песней, вторичный ключ |
| album\_id | number | Идентификатор альбома, ассоциированный с этой песней, вторичный ключ |
| song\_name | varchar2(50) | Название песни |
| song\_release | date | Дата выхода песни |
| song\_likes\_count | number | Количество добавлений в Избранное альбома пользователями |
| song\_file | varchar2(1000) | Файл песни |

Таблица LIKED\_SONGS содержит информацию о добавленных в Избранное песнях. Описание её столбцов приведено в таблице 2.5.

Таблица 2.5 – Описание столбцов таблицы LIKED\_SONGS.

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| user\_id | number | Идентификатор пользователя, добавляющего песню в Избранное, вторичный ключ |
| liked\_song\_id | number | Идентификатор песни, добавленной в Избранное, вторичный ключ |

Таблица LIKED\_ALBUMS содержит информацию о добавленных в Избранное альбомах. Описание её столбцов приведено в таблице 2.6.

Таблица 2.6 – Описание столбцов таблицы LIKED\_ALBUMS

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| user\_id | number | Идентификатор пользователя, добавляющего песню в Избранное, вторичный ключ |
| liked\_albums\_id | number | Идентификатор альбома, добавленного в Избранное, вторичный ключ |

Таблица BAND\_MEMBERS содержит информацию об участниках группы. Описание её столбцов приведено в таблице 2.7.

Таблица 2.7 – Описание столбцов таблицы BAND\_MEMBERS

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| member\_id | number | Идентификатор участника группы, первичный ключ |
| band\_id | number | Идентификатор группы, ассоциированной с участником группы, вторичный ключ |
| member\_name | varchar2(50) | Имя участника группы |
| member\_instrument | varchar2(50) | Роль участника в группе |

Скрипт создания всех таблиц базы данных приведён в приложении А.

# 3 Разработка объектов базы данных

## 3.1 Разработка таблиц

Все таблицы создаются в табличном пространстве специального пользователя developer подключаемой базы данных FREEPDB1.

Каждая таблица, кроме LIKED\_SONGS и LIKED\_ALBUMS имеет столбец идентификатора строки типа number, который генерируется при каждом добавлении строки в таблицу как идентичность. Для хранения строковых значений и медиафайлов применяется тип данных varchar2, хранения даты – date. Описание столбцов представлено в разделе 2.1.

## 3.2 Разработка хранимых процедур

Для доступа к данным были реализованы хранимые процедуры, которые можно разделить на 3 категории в зависимости от вариантов использования: общие для всех, только для пользователя и только для администратора.

Список хранимых процедур, являющихся общими для всех, приведён в таблице 3.1.

Таблица 3.1 – Список общих процедур для всех пользователей

|  |  |
| --- | --- |
| Название процедуры | Описание |
| GetAllSongs | Получить все песни |
| GetAllBands | Получить все группы |
| GetAllAlbums | Получить все альбомы |
| GetMembersOfBand | Получить всех участников группы |
| GetBandById | Поучить группу по её идентификатору |
| GetAlbumById | Поучить альбом по его идентификатору |
| GetBestSongs | Поучить 6 самых популярных песен |
| GetNewSongs | Получить 6 новых песен |
| GetNewAlbums | Получить 6 новых альбомов |
| GetBestAlbums | Получить 6 самых популярных альбомов |
| GetAlbumByBandId | Получить альбом по идентификатору выпустившего его группы |
| GetSongsByAlbumId | Получить песни по идентификатору их альбома |
| FullTextSearchBands | Полнотекстовый поиск по группам |
| FullTextSearchSongs | Полнотекстовый поиск по песням |
| FullTextSearchAlbums | Полнотекстовый поиск по альбомам |
| FullTextSearchBandsByFuzzy | Полнотекстовый поиск по группам с неточным значением |
| FullTextSearchSongsByFuzzy | Полнотекстовый поиск по песням с неточным значением |

Продолжение таблицы 3.1

|  |  |
| --- | --- |
| FullTextSearchAlbumsByFuzzy | Полнотекстовый поиск по альбомам с неточным значением |

Гость, помимо общих для всех пользователей процедур, может также вызывать процедуру USER\_REGISTRATION для записи в таблицу USERS.

Список процедур только для пользователя приведён в таблице 3.2.

Таблица 3.2 – Список процедур только для пользователя

|  |  |
| --- | --- |
| Название процедуры | Описание |
| LikeSong | Добавить песню в Избранное |
| LikeAlbum | Добавить альбом в Избранное |
| GetUserByNameAndPassword | Получить пользователя по имени и паролю |
| GetCurrentUser | Получить текущего пользователя |
| GetLikedSongsByUserId | Получить все добавленные в Избранное пользователем песни |
| GetLikedAlbumsByUserId | Получить все добавленные в Избранное пользователем альбомы |

Список процедур только для администратора приведён в таблице 3.3.

Таблица 3.3 – Список процедур только для администратора

|  |  |
| --- | --- |
| Название процедуры | Описание |
| AddBand | Добавить группу |
| AddAlbum | Добавить альбом |
| AddSong | Добавить песню |
| AddBandMember | Добавить участника группы |
| DeleteSong | Удалить песню |
| DeleteBand | Удалить группу |
| DeleteAlbum | Удалить альбом |
| DeleteMemberOfBand | Удалить участника из группы |
| UpdateSong | Обновить песню |
| UpdateAlbum | Обновить альбом |
| UpdateBand | Обновить группу |
| UpdateMemberInBand | Обновить участника группы |

Реализация всех приведённых выше процедур и присвоение привилегий на их вызов представлено в приложении Б.

## 3.3 Разработка функций

В базе данных была создана единственная функция, которая возвращает общее количество песен. Её реализация представлена в приложении В.

## 3.4 Разработка индексов

Индексы нужны для ускоренного доступа к данным таблиц Users по столбцу name, содержащему имя пользователя. Также были разработаны специальные индексы song\_name\_idx по столбцу song\_name, band\_name\_idx по столбцу band\_name и album\_name\_idx по столбцу album\_name для реализации технологии полнотекстового поиска в БД.

## 3.5 Разработка триггеров

Триггеры позволяют автоматически оповещать о произошедших событиях в программе и применяются для реализации сложных ограничений целостности базы данных. В рамках базы данных были разработаны 4 триггера. Триггеры like\_the\_song и like\_the\_album срабатывают после вставки в таблицу LIKED\_SONGS и LIKED\_ALBUMS соответственно на уровне строки, обновляя значение SONG\_LIKES\_COUNT и ALBUM\_LIKES\_COUNT в таблицах SONGS и ALBUMS соответственно. Триггер DeleteBandsTrigger срабатывает при удалении строки из группы BANDS, также удаляя о ней всю информацию, во избежание нарушений целостности таблиц базы данных. Триггер DeleteSongsTriggers срабатывает при удалении альбома, удаляя все песни с него с той же целью, что и предыдущий описанный триггер. Реализация данных триггеров представлена в приложении Г.

# 4 Описание процедур импорта и экспорта

Одним из требований к курсовому проекту является импорт данных из XML – файлов и их экспорт из данного формата. XML представляет собой расширяемый язык разметки для передачи и хранения данных. Данный формат рекомендован Консорциумом Всемирной паутины и довольно лёгок в использовании и распространении, поэтому он так часто используется при разработке API.

Для реализации импорта и экспорта данных из XML необходимо создать директорий, где создаём файлы формата XML для импорта и экспорта данных таблицы BANDS. Процедура XmlImport позволяет импортировать данные из XML – файла в таблицу BANDS. Её реализация представлена на рисунке 4.1.

|  |
| --- |
| CREATE OR REPLACE DIRECTORY DIR AS '/opt/oracle/product/23c/dbhomeFree/xml\_oracle';  create or replace procedure XmlImport  is  begin  insert into BANDS (band\_name, band\_location, band\_genre, band\_year\_of\_creation, band\_logo)  SELECT ExtractValue(VALUE(BANDS), '//BAND\_NAME') AS band\_name ,  ExtractValue(VALUE(BANDS), '//BAND\_LOCATION') AS band\_location,  ExtractValue(VALUE(BANDS), '//BAND\_GENRE') as band\_genre,  ExtractValue(VALUE(BANDS), '//BAND\_YEAR\_OF\_CREATION') as band\_year\_of\_creation,  ExtractValue(VALUE(BANDS), '//BAND\_LOGO') as band\_logo  FROM TABLE(XMLSequence(EXTRACT(XMLTYPE(bfilename('DIR', 'BANDS\_import.xml'),  nls\_charset\_id('UTF-8')),'/ROWSET/ROW'))) BANDS;  end XmlImport; |

Листинг 4.1 – процедура импорта из XML – файла

В данной процедуре данные вставляются в таблицу BANDS, которые извлекаются из XML – файла, находящегося в папке DIR, через функцию ExtractValue. Содержимое данного файла представлено на рисунке 4.1.

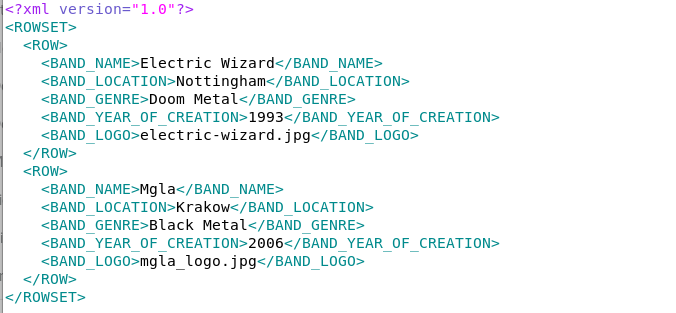


Рисунок 4.1 – содержимое XML – файла для импорта

Процедура XmlExport позволяет экспортировать содержимое таблицы BANDS в XML – файл. Её реализация представлена в листинге 4.2.

|  |
| --- |
| create or replace procedure XmlExport  is  rc sys\_refcursor;  doc DBMS\_XMLDOM.DOMDocument;  begin  OPEN rc FOR  SELECT band\_name, band\_location, band\_genre, band\_year\_of\_creation, band\_logo FROM BANDS;  doc := DBMS\_XMLDOM.NewDOMDocument(XMLTYPE(rc));  DBMS\_XMLDOM.WRITETOFILE(doc, 'DIR/BANDS\_export.xml');  commit;  exception  when others then  raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);  end XmlExport; |

Листинг 4.2 – процедура экспорта в XML – файл

В данной процедуре происходит извлечение данных из таблицы BANDS с помощью курсора и сохранение их в XML – файле. Затем эти данные преобразуются в формат DOMDocument и с помощью функции WRITETOFILE записываются в BANDS\_export.xml.

# 5 Тестирование производительности

Тестирование производительности заключается в проверке времени выполнения запроса к таблице, содержащей очень большое количество строк. В листинге 5.1 приведён анонимный блок, вставляющий 100000 строк в таблицу USERS.

|  |
| --- |
| BEGIN  FOR i IN 1 .. 100000  LOOP  insert into users(name, email, password, role) VALUES ('User' || i,'user'||i||'@gmail.com','qwe'||i, 'user');  END LOOP;  END; |

Листинг 5.1 – анонимный блок для вставки 100000 строк

До перестройки индекса время вызова составило 564 миллисекунды, а после перестройки заняло 52 миллисекунды.

# 6 Описание технологии и её применения в базе данных

Полнотекстовый поиск – это поиск по всему содержимому документа или очень большого текста в базе данных, а не по определённым атрибутам, что позволяет заметно ускорять запросы к подобным данным и экономить вычислительные мощности.

В основе полнотекстовых поисков лежит индекс, куда загружаются все слова или словосочетания из текстовых документов или очень длинных строк. И при загрузке пользователем своего поискового запроса, поиск осуществляется по этому индексу.

В Oracle данный механизм реализован с помощью технологии Oracle Text, которая позволяет работать с текстовыми данными и документами. Она содержит администратора CTXSYS, который позволяет создавать индексы полнотекстового поиска. Для доступа к администратору Oracle Text необходимо, подключившись в качестве администратора БД, выполнить скрипт catctx.sql через PL/SQL. Вызов данного скрипта представлен на рисунке 6.1



Рисунок 6.1 – вызов установщика пользователя CTXSYS

Помимо пути к SQL – файлу также указываются пароль для ctxsys, табличное пространство для него по умолчанию и временное, а также незаблокированный тип его учётной записи. После выполнения данного скрипта необходимо присвоить роль CTXAPP пользователю, которая позволяет ему выполнять все задачи CTXSYS. Для создания индексов также надо выдать пользователю привилегии на использование пакетов, которые их содержат. Список пакетов CTXSYS приведён в таблице 6.1.

Таблица 6.1 – Список пакетов CTXSYS

|  |  |
| --- | --- |
| Название пакета | Описание |
| CTX\_DDL | Содержит ряд хранимых процедур и функций, которые могут использоваться для создания и управления настройками индексов текста, таких как предпочтения, группы разделов и списки стоп-слов |
| CTX\_DOC | Содержит ряд процедур и функций, которые могут использоваться для запроса документ-сервисов, таких как генерация обычного текста или HTML-версии документа, выделение текста, генерация списка тем и т.д. |

Продолжение таблицы 6.1

|  |  |
| --- | --- |
| CTX\_OUTPUT | Содержит хранимые процедуры и функции, которые могут использоваться для управления логированием запросов, таких как создание отчетов о запросах, а также для управления настройками индексации текста, таких как включение и отключение сбора статистики запросов |
| CTX\_QUERY | Содержит ряд процедур и функций, которые могут использоваться для генерации отчетов о запросах, подсчета количества совпадений и создания хранимых выражений запросов |
| CTX\_REPORT | Cодержит несколько процедур, которые могут использоваться для создания отчетов, помогающих настраивать приложения и устранять проблемы |
| CTX\_THES | Содержит ряд хранимых процедур и функций, которые могут использоваться для создания, изменения и удаления тезаурусов (множество смысловыражающих единиц некоторого языка с заданной на нём системой семантических отношений), а также для поиска и замены терминов в тексте с использованием тезауруса |
| CTX\_ULEXER | Cодержит ряд хранимых процедур и функций, которые могут использоваться для создания пользовательского лексера, а также для управления списками символов-разделителей, символов-терминаторов и т.д. |

Для проверки работы индеска полнотекстового поиска было разработано несколько процедур для работы с таблицами BANDS, SONGS, ALBUMS. Коды данных процедур содержатся в приложении Б.

# 7 Краткое описание приложения для демонстрации

Для демонстрации работоспособности было разработано приложение на языке Java с использованием технологии Java Servlet Pages и веб – сервера Glassfish. Сценарий взаимодействия пользователя с приложением можно построить следующим образом: пользователь переходит по адресу http://localhost:8080/dalvhikkiRecordsApp, где на главной странице ему предоставляется список лучших альбомов и песен, а также недавних релизов. В системе он определён как гость, поэтому он не может добавлять песни и альбомы в Избранное. Он нажимает на кнопку «Registration» для регистрации в системе или «Authorization» для входа в аккаунт, где вводит учётные данные. Далее пользователя переносит на главную страницу, однако сейчас он имеет возможность добавлять песни и альбомы в Избранное и для просмотра этого списка он может воспользоваться пунктами верхнего меню «Liked Songs» и «Liked Albums». При нажатии в верхем меню на своё имя, он выходит из системы.

# 8 Руководство пользователя

При первом открытии сайта пользователю представляются списки самых популярных альбомов и песен, а также новых релизов, представленной на рисунках 8.1. При нажатии на кнопку добавления в Избранное, находящуюся в правой части блока с песней или альбомом, система никак не отреагирует, ибо пользователь не вошёл в учётную запись.

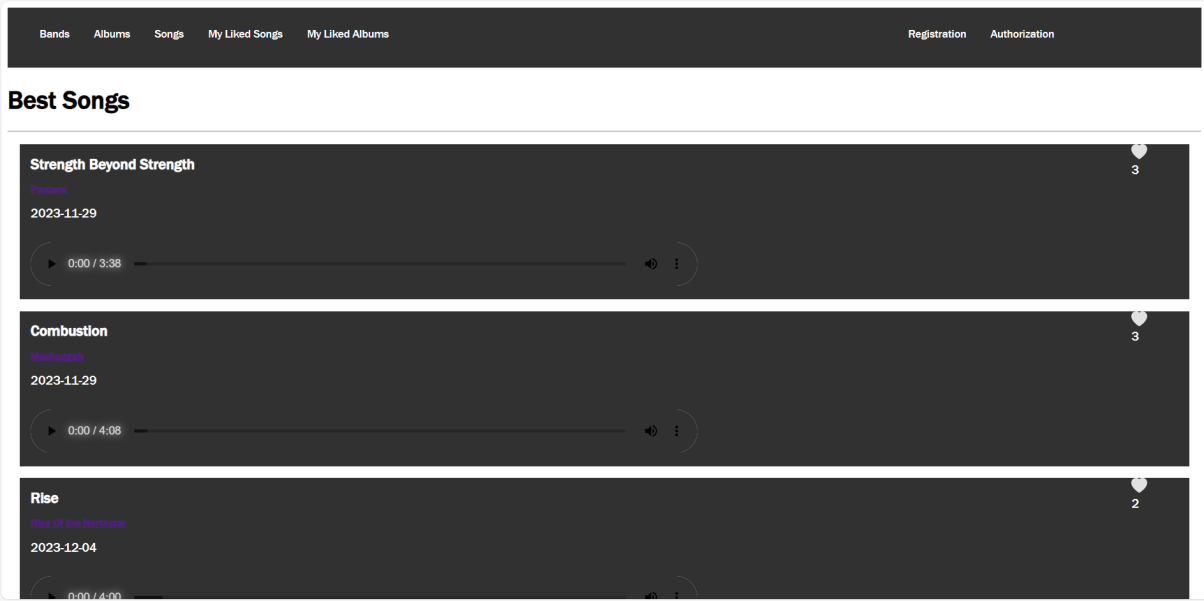


Рисунок 8.1 – главная страница

Самые популярные песни и альбомы отсотрированы по количеству добавлений в Избранное пользователями, а новые песни и альбомы – по дате добавления в таблицу.

Гостю также доступны просмотр всех альбомов, песен и альбомов. Он может перейти к ним по пунктам верхнего меню. Страница со списком всех групп представлена на рисунке 8.2.

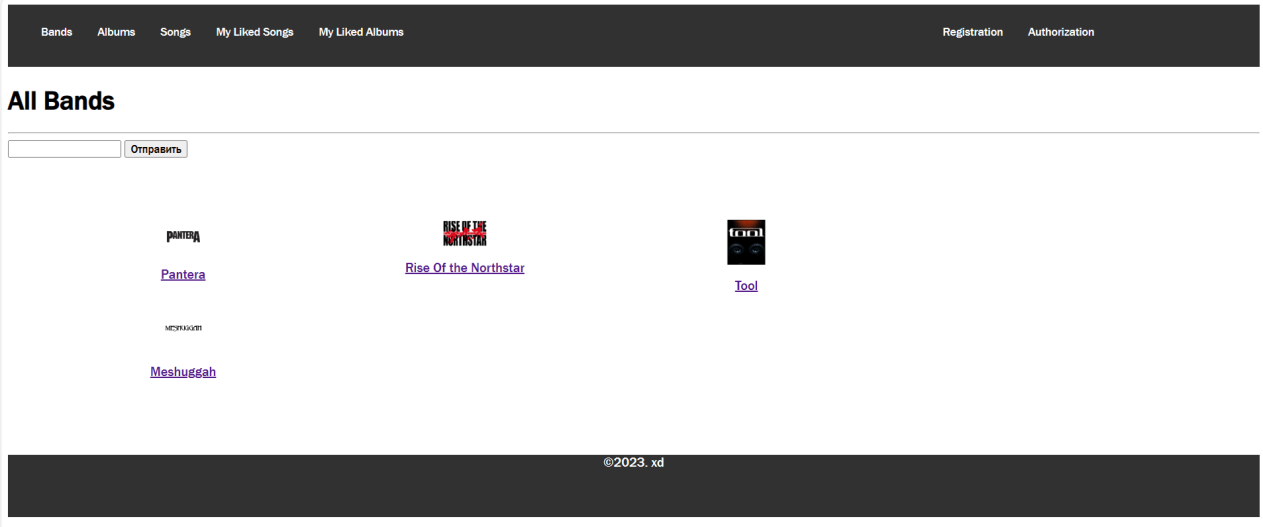


Рисунок 8.2 – Страница со списком всех групп

На данной странице также имеется поисковая строка, позволяющая выбрать группу по её названию. На страницах со всеми песнями и альбомами также имеется данный механизм.

Дальше он может перейти на страницу интересующую его группу, где ему предоставляется информация о группе такая как место и год основания, жанр музыки, список её участников, а также список её альбомов. Это представлено на рисунке 8.3.

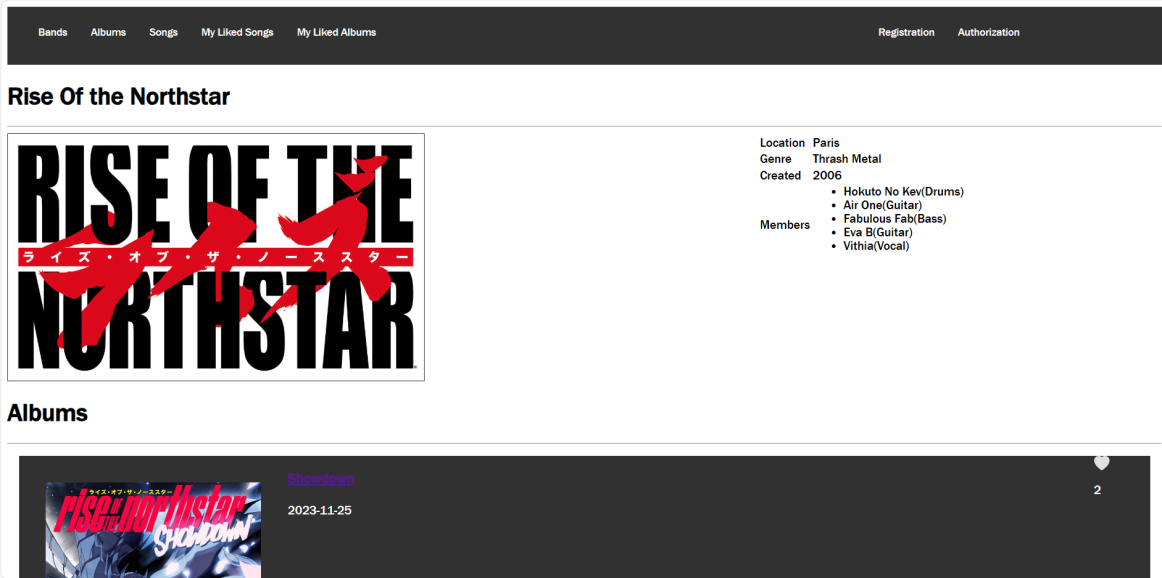


Рисунок 8.3 – страница с информацией о группе

Страница с информацией об альбоме представлена на рисунке 8.4. Здесь пользователь может просмотреть информации о дате релиза и группе, выпустившей его, а также список песен, включённых в него.

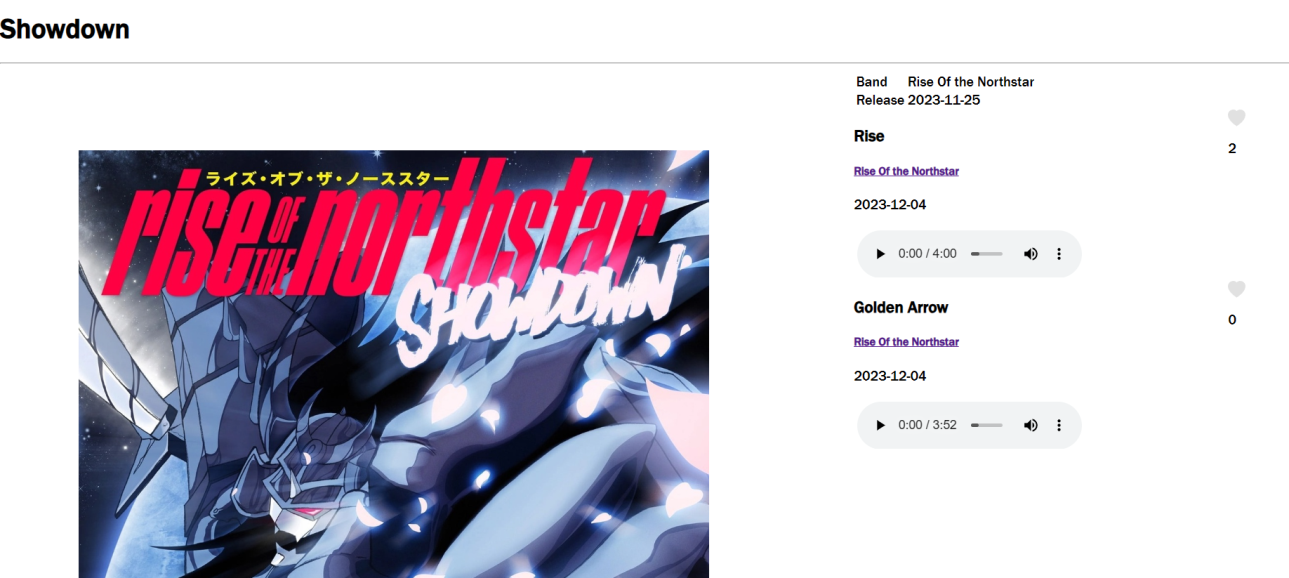


Рисунок 8.4 – страница с альбомом

При переходе на пункты меню «My Liked Songs» и «My Liked Albums» гостя перенесёт на страничку с просьбой войти или зарегистрироваться в системе, что продемонстрировано на рисунке 8.5.

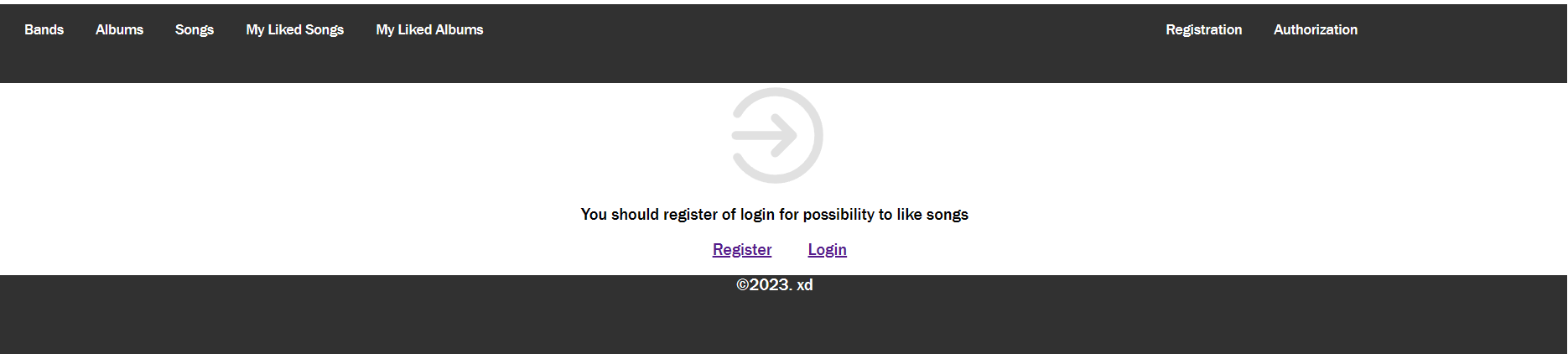


Рисунок 8.5 – страница с просьбой зарегистрироваться в системе

При переходе на страницу регистрации или входа пользователь входит в учётную запись. Регистрация приведена на рисунке 8.6.

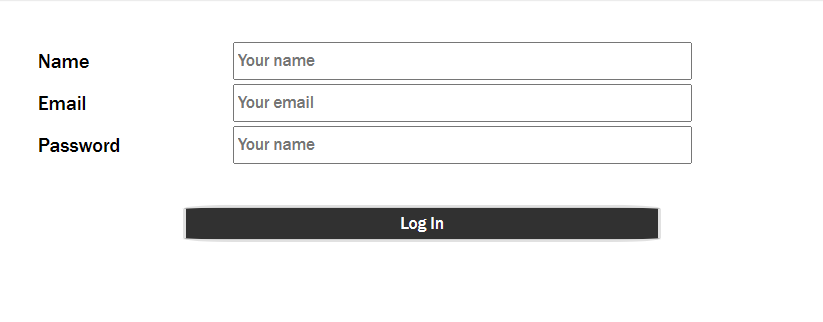


Рисунок 8.6 – страница регистрации

При успешной регистрации в системе пользователь входит в приложение как «авторизованный пользователь».

Страница входа представлена на рисунке 8.7.

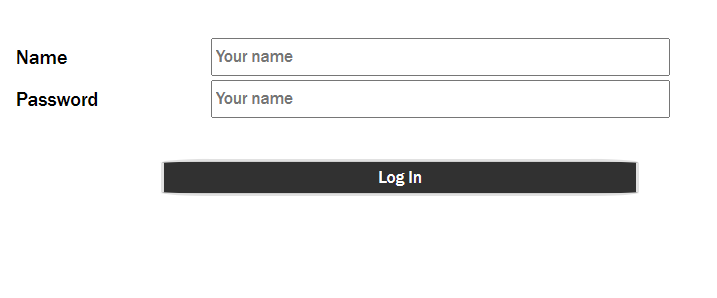


Рисунок 8.7 – страница входа в учётную запись

При неудачной регистрации, например, ввода уже существующего имени или почты, пользователя переносит на страницу с ошибкой и предоставлением возможности повторения попытки. Страница ошибки представлена на рисунке 8.8.

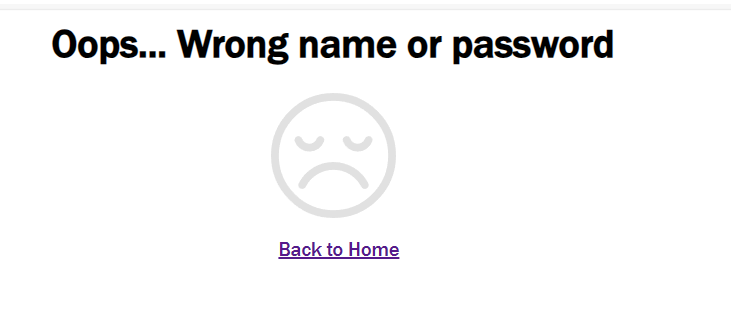


Рисунок 8.8 – страница ошибки

При успешном же входе в учётную запись пользователя переносит на главную страницу. Здесь он уже может добавлять песни и альбомы в Избранное, причём они помечаются красным цветом. При переходе на пункты меню «Liked Songs» и «My Liked Albums» он может просмотреть понравившиеся ему песни или же альбомы. Это представлено на рисунке 8.9.

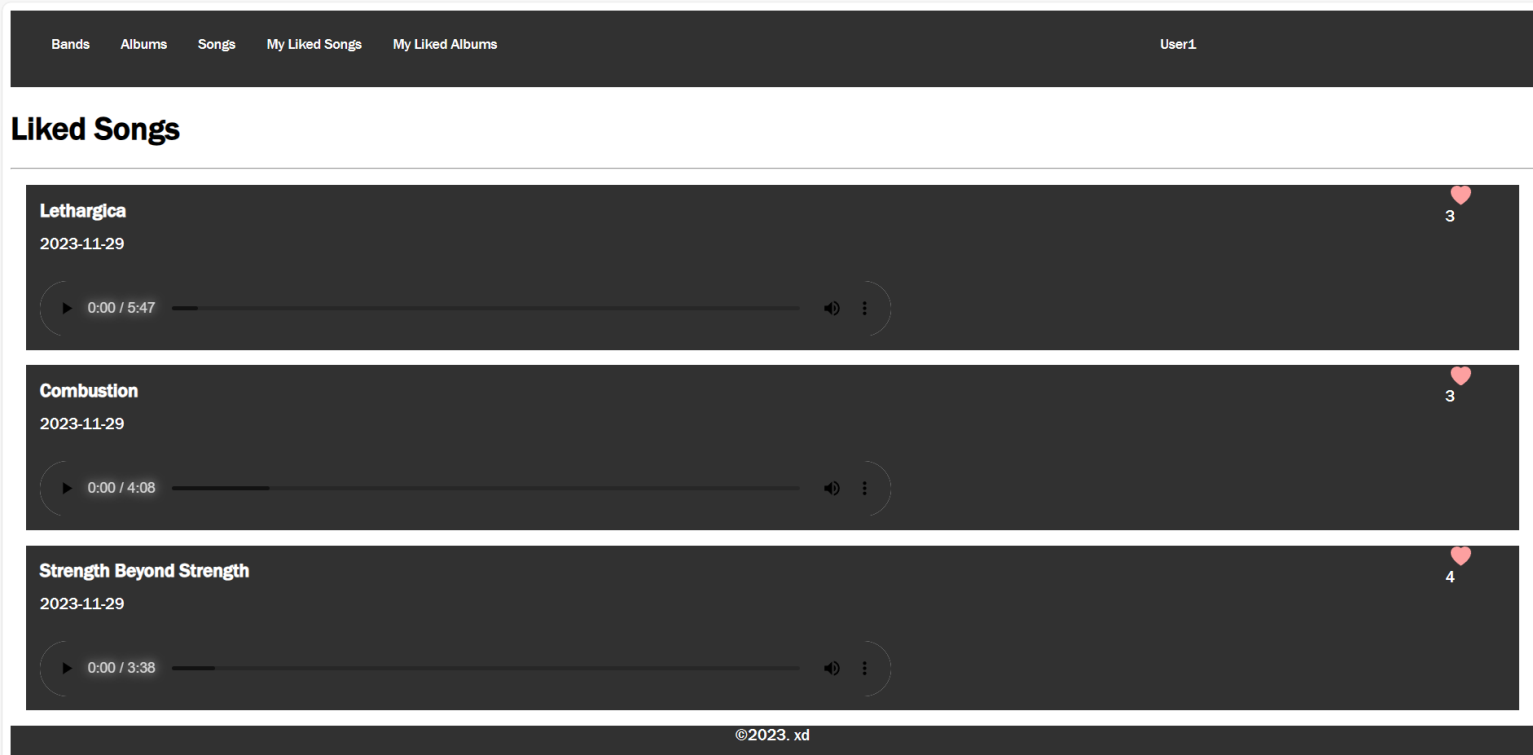


Рисунок 8.9 – список песен, добавленные в Избранное пользователя

При входе в учётную запись в качестве администратора, он переходит на главную страницу, но уже с расширенным функционалом. Теперь он может изменять, добавлять и удалять песни, альбомы, группы и их участников. Важно отметить, что файл с логотипом группы записывается на веб – сервер, а в базу данных – его название.

Страница для добавления группы представлена на рисунке 8.10.

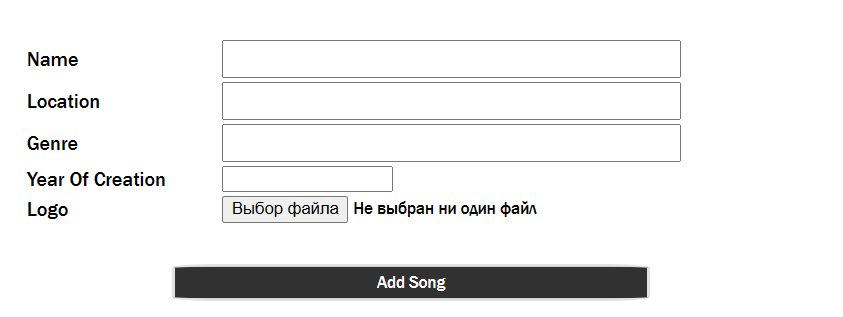


Рисунок 8.10 – страница для добавления группы

Страница для обновления группы представлена на рисунке 8.11. Здесь администратор получает данные через идентификатор группы в таблице, и вставляет их в форму для их изменения.



Рисунок 8.11 – страница для обновления группы

При переходе на страницу с информацией о группе администратор может менять, добавлять и удалять участников группы, а также добавлять и изменять альбомы. При добавлении или обновлении участника группы он указывает его имя и роль в группе.

Форма для добавления участника группы представлена на рисунке 8.12.

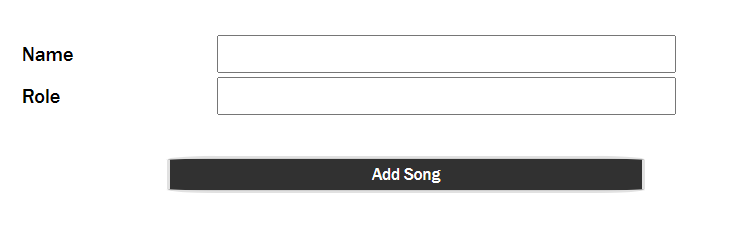


Рисунок 8.12 – форма для добавления участника в группу

Форма для обновления участника группы представлена на рисунке 8.13.

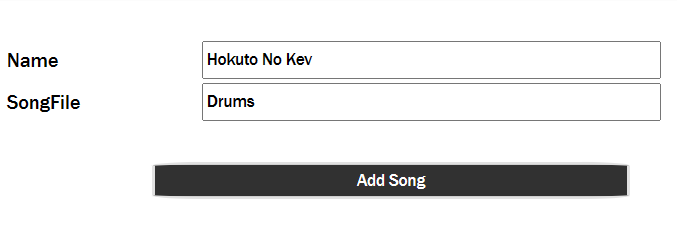


Рисунок 8.13 – форма для обновления участника группы

Cтраница добавления альбома группы представлена на рисунке 8.14. Здесь он указывает имя альбома и файл с логотипом альбома, причём в базу данных так же, как и с добавлением групп, записывается название добавленного файла, а сам файл записывается на веб – сервер.

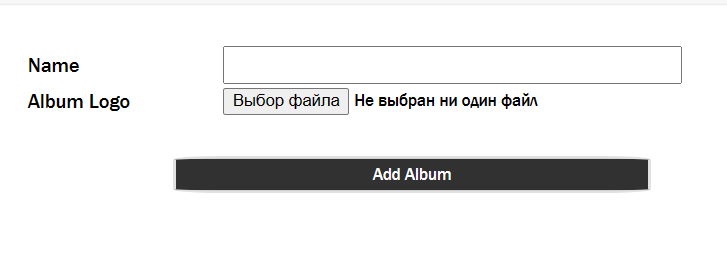


Рисунок 8.14 – форма для добавления альбома

При переходе на страницу с альбомом администратор может добавлять песню в альбом, а также изменять или удалять её.

Страница для добавления песни представлена на рисунке 8.15. При добавлении он указывает имя песни и её файл.

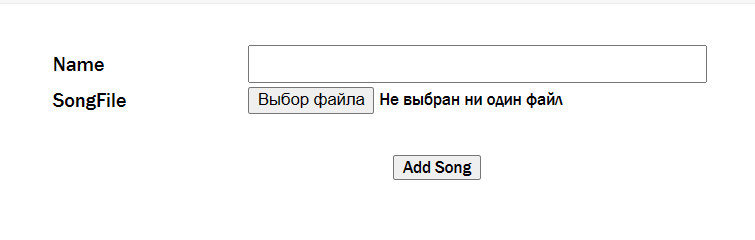


Рисунок 8.15 – форма добавления песни в альбом

Страница для обновления песни представлена на рисунке 8.16.

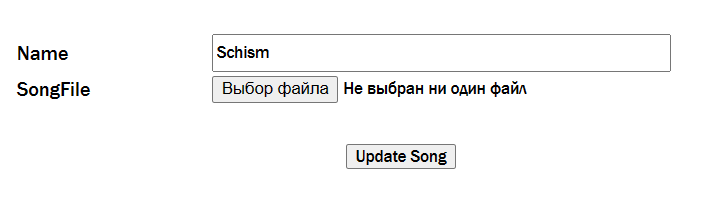


Рисунок 8.16 – форма обновления песни

Таким образом, был разработан пользовательский интерфейс для взаимодействия пользователя с базой данных с помощью 3 ролей: гостя, пользователя и администратора с возможностью работы с файлами мультимедиа.

# Заключение

При выполнении курсового проекта была разработана база данных на основе СУБД Oracle для лейбла метал – музыки, а также разработано приложение для демонстрации взаимодействия с ней. Были выполнены следующие функциональные требования:

* реализация ролей администратор, пользователь, гость;
* поиск по группам, альбомам, песням;
* добавление, удаление изменение групп, альбомов, песен администратором;
* предоставление популярных песен, новинок;
* предоставление прослушивания музыки.

Также был реализован импорт и экспорт данных БД в формат XML, протестирована производительность работы базы данных с большим числом данных и добавлены индексы для улучшения её производительности. Была рассмотрена технология полнотекстового поиска в БД и реализованы процедуры для этого. Исключения, возникшие внутри процедур, обрабатываются программно или самими процедурами.

# Список используемых источников

1. Oracle Database Documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/index.html. Дата доступа: 26.09.2023.
2. Oracle Text Reference 21c [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/21/ccref/text-reference.pdf. Дата доступа: 09.12.2023.
3. Java Servlet Specification [Электронный ресурс]. – Электронные данные. – Режим доступа: https://javaee.github.io/servlet-spec/downloads/servlet-4.0/servlet-4\_0\_FINAL.pdf . Дата доступа: 21.10.2023.
4. Java EE – Java Servlets and JSPs [Электронный ресурс]. – Электронные данные. – Режим доступа: https://medium.com/javarevisited/java-ee-java-servlets-and-jsps-6ed1d2fca237. Дата доступа: 07.11.2023.

# Приложение А

Листинг создания таблиц

|  |
| --- |
| create table USERS(  id number generated always as identity primary key,  name varchar2(30) not null,  email varchar2(30) not null,  password varchar2(32) not null,  role nvarchar2(20) not null check(role in ('user','admin'))  );  alter table USERS add constraint unique\_user unique(id,name,email,password);  create table BANDS(  band\_id number generated always as identity primary key,  band\_name varchar2(50) not null,  band\_location varchar2(50) not null,  band\_genre varchar2(50) not null,  band\_year\_of\_creation number not null check (band\_year\_of\_creation>=1900 and band\_year\_of\_creation<=2023),  band\_logo varchar2(1000) not null  );  alter table BANDS add constraint unique\_band unique(band\_id, band\_name, band\_logo);  create table SONGS(  song\_id number generated always as identity primary key,  band\_id number not null,  album\_id number not null,  song\_name varchar2(50) not null,  song\_release date not null,  song\_likes\_count number,  song\_file varchar2(1000) not null,  constraint fk\_band\_id foreign key(band\_id) references BANDS(band\_id),  constraint fk\_album\_id foreign key(album\_id) references ALBUMS(album\_id)  );  alter table SONGS add constraint unique\_song unique(song\_id, song\_file);  create table ALBUMS(  album\_id number generated always as identity primary key,  album\_name varchar2(50) not null,  band\_id number not null,  release\_date date,  album\_likes\_count number,  album\_photo varchar2(1000) not null,  constraint fk\_album\_band\_id foreign key(band\_id) references BANDS(band\_id)  );  alter table ALBUMS add constraint unique\_album unique(album\_id, album\_photo);  create table BAND\_MEMBERS(  member\_id number generated always as identity,  band\_id number not null,  member\_name varchar2(50) not null,  member\_instrument varchar2(20) not null,  constraint fk\_member\_band\_id foreign key(band\_id) references BANDS(band\_id)  );  create table LIKED\_SONGS(  user\_id number not null,  liked\_song\_id number not null,  constraint fk\_user\_id foreign key(user\_id) references USERS(id),  constraint fk\_song\_id foreign key(liked\_song\_id) references SONGS(song\_id)  );  create table LIKED\_ALBUMS(  user\_id number not null,  liked\_album\_id number not null,  constraint fk\_user\_liked\_album\_id foreign key(user\_id) references USERS(id),  constraint fk\_album\_user\_id foreign key(liked\_album\_id) references ALBUMS(album\_id)  ); |

# Приложение Б

Листинг реализации пользователей базы данных

|  |
| --- |
| create tablespace dalvhikkiRecords  datafile 'dalvhikkiRecords.dbf'  size 100m  autoextend on next 100m  maxsize unlimited  extent management local  segment space management auto;  create user developer  identified by qwert  default tablespace dalvhikkiRecords  temporary tablespace temp  quota unlimited on dalvhikkiRecords;  create user guest identified by qwerty  default tablespace dalvhikkiRecords  temporary tablespace temp  quota unlimited on dalvhikkiRecords;  create user admin identified by asdfg  default tablespace dalvhikkiRecords  temporary tablespace temp  quota unlimited on dalvhikkiRecords;  create user authorized\_user identified by zxcvb  default tablespace dalvhikkiRecords  temporary tablespace temp  quota unlimited on dalvhikkiRecords; |

Листинг реализации процедур базы данных

|  |
| --- |
| create or replace PROCEDURE USER\_REGISTRATION  (user\_name varchar2,user\_email varchar2,user\_password varchar2,user\_role nvarchar2,cur out sys\_refcursor) as  begin  insert into USERS(name,email,password,role) values (user\_name,user\_email,user\_password,user\_role);  open cur for select \* from USERS where name=user\_name;  end USER\_REGISTRATION;    drop procedure USER\_REGISTRATION;  --Get procedures  create or replace procedure GetAllSongs(cur out sys\_refcursor) as  begin  open cur for select \* from SONGS;  end GetAllSongs;  create or replace procedure GetAllAlbums(cur out sys\_refcursor) as  begin  open cur for select \* from ALBUMS;  end GetAllAlbums;  create or replace procedure GetUserByNameAndPassword(nameOfUser varchar2, passwordOfUser varchar2, cur out sys\_refcursor) as  begin  open cur for select \* from USERS where name=nameOfUser and password=passwordOfUser;  end GetUserByNameAndPassword;  create or replace procedure GetAllBands(cur out sys\_refcursor)  as  begin  open cur for select \* from BANDS;  end GetAllBands;  select \* from Bands;  create or replace procedure GetMembersOfBand (id number, cur out sys\_refcursor) as  begin  open cur for select member\_id, band\_id, member\_name, member\_instrument from BAND\_MEMBERS where band\_id=id;  end GetMembersOfBand;  create or replace procedure GetMemberOfBand (id number, memberid number, cur out sys\_refcursor) as  begin  open cur for select member\_id, band\_id, member\_name, member\_instrument from BAND\_MEMBERS where band\_id=id and member\_id=memberid;  end GetMemberOfBand;  create or replace procedure GetSongsOfAlbum (id number, cur out sys\_refcursor) as  begin  open cur for select \* from Songs where album\_id=id;  end GetSongsOfAlbum;  create or replace procedure GetCurrentUser(login varchar2, pass varchar2, cur out sys\_refcursor) as  begin  open cur for select \* from USERS where name=login and password=pass;  end GetCurrentUser;  commit;  create or replace procedure GetNewSongs(cur out sys\_refcursor) as  begin  open cur for select \* from SONGS where rownum<=6 order by song\_release desc;  end GetNewSongs;  create or replace procedure GetNewAlbums(cur out sys\_refcursor) as  begin  open cur for select \* from ALBUMS where rownum<=6 order by release\_date desc;  end GetNewAlbums;  create or replace procedure GetBestSongs(cur out sys\_refcursor) as  begin  open cur for select \* from SONGS where rownum<=6 order by song\_likes\_count desc;  end GetBestSongs;  create or replace procedure GetBestAlbums(cur out sys\_refcursor) as  begin  open cur for select \* from ALBUMS where rownum<=6 order by album\_likes\_count desc;  end GetBestAlbums;  create or replace procedure GetBandNameByBandId(bandid number,cur out sys\_refcursor) as  begin  open cur for select \* from Bands where band\_id=bandid;  end GetBandNameByBandId;  create or replace PROCEDURE GetAlbumById(id number, cur out sys\_refcursor) as  begin  open cur for select \* from Albums where album\_id=id;  end GetAlbumById;  create or replace PROCEDURE GetAlbumByBandId(id number, cur out sys\_refcursor) as  begin  open cur for select \* from Albums where band\_id=id;  end GetAlbumByBandId;  create or replace PROCEDURE GetSongsByAlbumId(id number, cur out sys\_refcursor) as  begin  open cur for select \* from SONGS where album\_id=id;  end GetSongsByAlbumId;  create or replace PROCEDURE GetAllLikedSongsById(id number, cur out sys\_refcursor) as  begin  open cur for select \* from LIKED\_SONGS where user\_id=id;  end GetAllLikedSongsById;  create or replace PROCEDURE GetLikedSongBySongIdAndUserId(userId number, songId number, cur out sys\_refcursor) as  begin  open cur for select \* from LIKED\_SONGS where user\_id=userId and liked\_song\_id=songId;  end GetLikedSongBySongIdAndUserId;  create or replace PROCEDURE GetAllLikedAlbumsById(id number, cur out sys\_refcursor) as  begin  open cur for select \* from LIKED\_ALBUMS where user\_id=id;  end GetAllLikedAlbumsById;  create or replace PROCEDURE GetBandByBandId(id number, cur out sys\_refcursor) as  begin  open cur for select \* from BANDS where band\_id=id;  end GetBandByBandId;  create or replace PROCEDURE GetSongBySongId(id number, cur out sys\_refcursor) as  begin  open cur for select \* from SONGS where song\_id=id;  end GetSongBySongId;  --Add - Procedures  create or replace procedure AddSong(albumid number, songname varchar2, bandid number, songrelease date, songfile varchar2) as  begin  insert into SONGS(band\_id,album\_id,song\_name,song\_release,song\_likes\_count,song\_file) values (bandid,albumid, songname, songrelease, 0, songfile);  end AddSong;  create or replace procedure AddAlbum(albumname varchar2, bandid number, albrelease date, albphoto varchar2) as  begin  insert into ALBUMS(album\_name,band\_id,release\_date,album\_likes\_count, album\_photo) values (albumname, bandid, albrelease,0,albphoto);  end AddAlbum;  create or replace procedure AddBand(bandname varchar2, bandlocation varchar2, bandgenre varchar2, yearofcreation number,bandlogo varchar2) as  begin  insert into BANDS(band\_name, band\_location, band\_genre, band\_year\_of\_creation, band\_logo) values(bandname, bandlocation, bandgenre, yearofcreation, bandlogo);  exception  when others  then dbms\_output.put\_line(sqlcode || ': error = ' || sqlerrm);  end AddBand;  create or replace procedure AddMemberInBand(bandid number, membername varchar2, memberrole varchar2) as  begin  insert into BAND\_MEMBERS(band\_id, member\_name, member\_instrument) values(bandid, membername, memberrole);  end AddMemberInBand;  --Update procedures  create or replace procedure UpdateSong(id number, songname varchar2, songfile varchar2) as  begin  update SONGS set song\_name=songname, song\_file=songfile where song\_id=id;  end UpdateSong;  create or replace procedure UpdateAlbum(albumname varchar2, bandid number, albumphoto varchar2) as  begin  update ALBUMS set album\_name=albumname, album\_photo=albumphoto where band\_id=bandid;  end UpdateAlbum;  create or replace procedure UpdateBand(bandname varchar2, bandid number, bandlocation varchar2, bandgenre varchar2, yearofcreation number, bandlogo varchar2) as  begin  update BANDS set band\_name=bandname, band\_location=bandlocation, band\_genre=bandgenre, band\_year\_of\_creation=yearofcreation, band\_logo=bandlogo where band\_id=bandid;  end UpdateBand;  create or replace procedure UpdateMemberInBand(memberid number,membername varchar2, memberrole varchar2) as  begin  update BAND\_MEMBERS set member\_name=membername, member\_instrument=memberrole where member\_id=memberid;  end UpdateMemberInBand;  --Delete procedures  create or replace procedure DeleteSong (id number) as  begin  delete from Songs where song\_id=id;  end DeleteSong;  create or replace procedure DeleteBand (id number) as  begin  delete from BANDS where band\_id=id;  end DeleteBand;  create or replace procedure DeleteMemberOfBand (id number) as  begin  delete from BAND\_MEMBERS where member\_id=id;  end DeleteMemberOfBand;  create or replace procedure DeleteAlbum (id number) as  begin  delete from ALBUMS where album\_id=id;  end DeleteAlbum;  --like/unlike songs/albums  create or replace procedure LikeSong(user\_id\_ number, song\_id\_ number) as  begin  insert into LIKED\_SONGS values(user\_id\_,song\_id\_);  end LikeSong;  create or replace procedure LikeAlbum(user\_id\_ number, album\_id\_ number) as  begin  insert into LIKED\_ALBUMS values(user\_id\_,album\_id\_);  end LikeAlbum; |

Листинг реализации выдачи привилегий пользователям

|  |
| --- |
| grant connect,create session to guest;  grant create session to guest;  grant execute on GetAllSongs to guest;  grant execute on GetAllBands to guest;  grant execute on GetAllAlbums to guest;  grant execute on GetNewSongs to guest;  grant execute on GetBandNameByBandId to guest;  grant execute on GetBandByBandId to guest;  grant execute on GetNewAlbums to guest;  grant execute on GetMembersOfBand to guest;  grant execute on USER\_REGISTRATION to guest;  grant execute on GetBestSongs to guest;  grant execute on GetBestAlbums to guest;  grant execute on GetAlbumById to guest;  grant execute on GetAlbumByBandId to guest;  grant execute on GetSongsByAlbumId to guest;  grant execute on GetUserByNameAndPassword to guest;  grant connect to admin;  grant create session to admin;  grant execute on GetAllSongs to admin;  grant execute on GetAllBands to admin;  grant execute on GetAllAlbums to admin;  grant execute on GetMembersOfBand to admin;  grant execute on GetMemberOfBand to admin;  grant execute on GetNewSongs to admin;  grant execute on GetNewAlbums to admin;  grant execute on GetBestSongs to admin;  grant execute on GetBestAlbums to admin;  grant execute on GetBandNameByBandId to admin;  grant execute on GetSongBySongId to admin;  grant execute on GetBandByBandId to admin;  grant execute on UpdateSong to admin;  grant execute on UpdateAlbum to admin;  grant execute on UpdateBand to admin;  grant execute on UpdateMemberInBand to admin;  grant execute on AddSong to admin;  grant execute on AddAlbum to admin;  grant execute on AddBand to admin;  grant execute on AddMemberInBand to admin;  grant execute on DeleteSong to admin;  grant execute on DeleteAlbum to admin;  grant execute on DeleteBand to admin;  grant execute on DeleteMemberOfBand to admin;  grant execute on GetAlbumById to admin;  grant execute on GetAlbumByBandId to admin;  grant execute on GetSongsByAlbumId to admin;  grant execute on GetCurrentUser to admin;  grant execute on GetUserByNameAndPassword to admin;  grant connect to authorized\_user;  grant create session to authorized\_user;  grant execute on GetLikedSongBySongIdAndUserId to authorized\_user;  grant execute on GetAllSongs to authorized\_user;  grant execute on GetAllBands to authorized\_user;  grant execute on GetAllAlbums to authorized\_user;  grant execute on GetMembersOfBand to authorized\_user;  grant execute on GetNewSongs to authorized\_user;  grant execute on GetNewAlbums to authorized\_user;  grant execute on GetBestSongs to authorized\_user;  grant execute on GetBestAlbums to authorized\_user;  grant execute on GetBandNameByBandId to authorized\_user;  grant execute on GetBandByBandId to authorized\_user;  grant execute on LikeSong to authorized\_user;  grant execute on LikeAlbum to authorized\_user;  grant execute on GetAlbumById to authorized\_user;  grant execute on GetAlbumByBandId to authorized\_user;  grant execute on GetSongsByAlbumId to authorized\_user;  grant execute on GetCurrentUser to authorized\_user;  grant execute on GetUserByNameAndPassword to authorized\_user; |

Листинг реализации процедур полнотекстового поиска

|  |
| --- |
| grant CTXAPP to developer;  GRANT EXECUTE ON CTXSYS.CTX\_DDL TO developer;  GRANT EXECUTE ON CTXSYS.CTX\_DOC TO developer;  GRANT EXECUTE ON CTXSYS.CTX\_OUTPUT TO developer;  GRANT EXECUTE ON CTXSYS.CTX\_QUERY TO developer;  GRANT EXECUTE ON CTXSYS.CTX\_REPORT TO developer;  GRANT EXECUTE ON CTXSYS.CTX\_THES TO developer;  GRANT EXECUTE ON CTXSYS.CTX\_ULEXER TO developer;  begin  ctx\_ddl.create\_preference('my\_wordlist','BASIC\_WORDLIST');  CTX\_DDL.CREATE\_stoplist('default\_stoplist');  ctx\_ddl.create\_preference('my\_lexer', 'BASIC\_LEXER');  ctx\_ddl.set\_attribute('my\_lexer', 'INDEX\_STEMS', 'ENGLISH');  end;  begin  ctx\_ddl.drop\_preference('my\_wordlist');  ctx\_ddl.drop\_preference('my\_lexer');  ctx\_ddl.drop\_stoplist('default\_stoplist');  end;  create index song\_name\_idx on SONGS (SONG\_NAME) indextype is CTXSYS.CONTEXT parameters('LEXER my\_lexer WORDLIST my\_wordlist STOPLIST default\_stoplist SYNC(on commit)');  create index index\_for\_bands on BANDS(band\_name) indextype is ctxsys.context;  create index index\_for\_albums on ALBUMS(album\_name) indextype is ctxsys.context;  create or replace procedure FullTextSearchSongs(search\_string varchar2, cur out sys\_refcursor)  as  begin  open cur for select \* from SONGS where contains(song\_name, search\_string)>0;  end FullTextSearchSongs;  create or replace procedure FullTextSearchBands(search\_string varchar2, cur out sys\_refcursor)  as  begin  open cur for select \* from BANDS where contains(band\_name, search\_string)>0;  end FullTextSearchBands;  create or replace procedure FullTextSearchAlbums(search\_string varchar2, cur out sys\_refcursor)  as  begin  open cur for select \* from ALBUMS where contains(album\_name, search\_string)>0;  end FullTextSearchAlbums;  create or replace procedure FullTextSearchSongsByFuzzy(search\_string varchar2, cur out sys\_refcursor)  as  begin  open cur for select \* from SONGS where contains(song\_name, 'fuzzy('||search\_string||')')>0;  end FullTextSearchSongsByFuzzy;  create or replace procedure FullTextSearchBandsByFuzzy(search\_string varchar2, cur out sys\_refcursor)  as  begin  open cur for select \* from BANDS where contains(band\_name, 'fuzzy('||search\_string||')')>0;  end FullTextSearchBandsByFuzzy;  create or replace procedure FullTextSearchAlbumsByFuzzy(search\_string varchar2, cur out sys\_refcursor)  as  begin  open cur for select \* from ALBUMS where contains(album\_name, 'fuzzy('||search\_string||')')>0;  end FullTextSearchAlbumsByFuzzy; |

# Приложение В

Листинг реализации функции подсчёта общего количества песен

|  |
| --- |
| create or replace function GETCOUNTOFSONGSONLABEL return number is  rc number(6);  begin  select count(\*) into rc from SONGS;  return rc;  end GETCOUNTOFSONGSONLABEL; |

# Приложение Г

Листинг реализации триггеров

|  |
| --- |
| create or replace trigger like\_the\_song  after insert on LIKED\_SONGS  for each row  begin  update songs set song\_likes\_count = song\_likes\_count + 1 where song\_id = :new.liked\_song\_id;  end;  create or replace trigger like\_the\_album  after insert on LIKED\_ALBUMS  for each row  begin  update albums set album\_likes\_count = album\_likes\_count + 1 where album\_id = :new.liked\_album\_id;  end;  CREATE OR REPLACE TRIGGER DeleteBandsTrigger  AFTER DELETE ON Bands  FOR EACH ROW  BEGIN  DELETE FROM Albums WHERE band\_id = :old.band\_id;  DELETE FROM Songs WHERE band\_id = :old.band\_id;  DELETE FROM BAND\_MEMBERS where band\_id=:old.band\_id;  END;  CREATE OR REPLACE TRIGGER DeleteUsersTrigger  AFTER DELETE ON USERS  FOR EACH ROW  BEGIN  DELETE FROM Liked\_Songs WHERE user\_id = :old.id;  Delete from Liked\_albums where user\_id = :old.id;  END;  CREATE OR REPLACE TRIGGER DeleteSongsTrigger  AFTER DELETE ON Albums  FOR EACH ROW  BEGIN  DELETE FROM Songs WHERE album\_id = :old.album\_id;  END; |