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 приложения для прослушивания подкастов с применением средств диагностики»

Выполнил студент Козека Елизавета Максимовна

(Ф.И.О.)

Руководитель работы ст. преп. Нистюк О.А.

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

И.о. зав. кафедрой доц. Блинова Е.А.

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

Курсовая работа защищена с оценкой

Минск 2024

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc90888193)

[1 Постановка задачи 4](#_Toc90888194)

[1.1 Обзор аналогичных решений 4](#_Toc90888194)

[1.2 Требования к проектируемому программному средству 8](#_Toc90888194)

[1.3 Вывод по разделу 8](#_Toc90888194)

[2 Проектирование базы данных 9](#_Toc90888195)

[3 Разработка объектов базы данных 13](#_Toc90888207)

[3.1 Таблицы 13](#_Toc90888194)

[3.2 Представления 14](#_Toc90888194)

[3.3 Пользователи 15](#_Toc90888194)

[3.4 Функции 16](#_Toc90888194)

[3.5 Хранимые процедуры 17](#_Toc90888194)

[3.6 Вывод по разделу 19](#_Toc90888194)

[4 Описание процедур импорта и экспорта 20](#_Toc90888208)

[5 Тестирование производительности 22](#_Toc90888208)

[6 Описание технологии и ее применения в базе данных 24](#_Toc90888208)

[6.1 Средства диагностики 24](#_Toc90888209)

[6.2 Мультимедийные типы данных 29](#_Toc90888210)

[6.3 Вывод по разделу 34](#_Toc90888210)

[7 Краткое описание приложения для демонстрации 35](#_Toc90888211)

[8 Руководство пользователя 36](#_Toc90888211)

[8.1 Сценарий обычного пользователя 36](#_Toc90888209)

[8.2 Сценарий администратора 43](#_Toc90888210)

[8.3 Вывод по разделу 47](#_Toc90888210)

[ЗАКЛЮЧЕНИЕ 48](#_Toc90888212)

[Список литературы 49](#_Toc90888213)

[ПРИЛОЖЕНИЕ А 50](#_Toc90888214)

[ПРИЛОЖЕНИЕ Б 52](#_Toc90888214)

[ПРИЛОЖЕНИЕ В 54](#_Toc90888214)

[ПРИЛОЖЕНИЕ Г 55](#_Toc90888214)

# ВВЕДЕНИЕ

Целью данной работы является разработка реляционной базы данных приложения для прослушивания подкастов. Эта база данных должна обеспечивать клиента доступом к аудиозаписям, загруженным в нее. Так же необходимо разработать соответствующее приложение для демонстрации её работы.

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 Oracle 21c, в связи с ее высокой производительностью и надежностью.

Так же было необходимо разработать приложение для демонстрации работы базы данных, взаимодействия с ней. Приложение было написано на языке программирования C# с графическим интерфейсом, выполненным с помощью WPF.

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 безопасности пользователей приложения в курсовом проекте используется технология шифрования паролей от аккаунта перед записью их в базу данных и средства диагностики. Также для обеспечения работы приложения использовались мультимедийные типы данных при хранении изображений и аудиофайлов.

Основные требования к приложению:

* управление информацией о подкастах;
* дополнение информации;
* реализация ролей администратора и обычного пользователя;
* поиск аудиозаписей по теме подкаста и диктору;
* загрузка аудиозаписей на платформу администратором;
* взаимодействие с базой данных при помощи хранимых процедур.

В пояснительной записке предоставлена информация о похожих продуктах, архитектуре, реализации проекта, а также руководство пользователя.

1 Постановка задачи

Подкасты стали неотъемлемой частью современного медиапространства, предоставляя пользователям доступ к разнообразному контенту на любые темы. Они играют важную роль в обучении, развлечении и обмене мнениями. С учетом растущей популярности подкастов требуется разработка мобильного приложения, которое позволит пользователям легко находить, слушать и управлять подкастами.

В рамках данного проекта необходимо спроектировать и реализовать базу данных приложения для прослушивания подкастов с использованием средств диагностики. Кроме этого, целями проекта являются обеспечение пользователям простого и интуитивно понятного интерфейса для поиска и прослушивания подкастов, реализация функций для управления подкастами, темами и дикторами, а также возможность создавать плейлисты.

Ключевые функции приложения будут включать регистрацию и авторизацию пользователей, поиск подкастов, что позволит находить подкасты по интересующей теме или диктору. Также важной частью приложения станет возможность прослушивания и управления подкастами, включая воспроизведение подкастов и создание своих плейлистов пользователями.

Весь доступный функционал будет представлен в виде UML-схемы в приложении А.

**1.1 Обзор аналогичных решений**

Для более глубокого понимания и эффективной постановки требований к проекту разработки приложения для прослушивания подкастов необходимо провести обзор аналогичных решений на рынке. Анализ существующих систем позволит выявить сильные и слабые стороны, успешные практики и возможные улучшения. При изучении аналогичных проектов следует обратить внимание на функциональность, пользовательский опыт, масштабируемость и безопасность. Такой обзор позволит извлечь ценные уроки из опыта других разработчиков и предоставит возможность определить ключевые характеристики, которые будут служить основой для успешной реализации и внедрения приложения.

Первый аналог — «Podster.FM», социальная аудио-платформа для авторов и слушателей. Она бесплатна и сосредоточена на русскоязычном сегменте. Очень важно, что на сервисе реализован встроенный плеер, который легко можно интегрировать на сайт, чтобы прослушивать эфиры не только в сторонних приложениях, но и на собственном ресурсе.

Редакция хостинга размещает тематические подборки выпусков, собирает списки популярных шоу и рекомендации для слушателей. Внутри, помимо плеера, есть подробная аналитика, где авторы следят за активностью своей аудитории. Кроме этого, вы можете создавать собственные плейлисты из разных эпизодов.

Недавно на сервисе подключили возможность сделать свои подкасты или отдельные эпизоды приватными — это полезно, если автор хочет работать на узкую аудиторию или дополнительно монетизировать контент, выдавая слушателям индивидуальный доступ.

На странице подкаста отображается обложка, заставка, социальные сети и сайт, ссылки на другие сервисы, где размещено шоу, плейлисты и, собственно, сами эпизоды. В настройках в кабинете автора можно выбрать формат записей: подкаст или сериал. В первом случае эпизоды будут располагаться от новых к более старым, во втором — в хронологическом порядке.

Преимущества:

* бесплатное размещение неограниченного количества подкастов;
* приятный и удобный интерфейс;
* большое количество вариантов представления аналитики подкаста;
* безлимитный трафик;
* встроенный плеер.

Недостатки:

* ограниченные возможности монетизации для авторов;
* зависимость от Интернет-соединения;
* отсутствие мобильного приложения.

Вывод: «Podster.FM» является отличным вариантом для начинающих авторов подкастов и слушателей, которые ищут бесплатный и доступный способ наслаждаться аудиоконтентом, но пользователям стоит учитывать некоторые ограничения платформы при использовании.

Интерфейс страницы с подкастами представлен на рисунке 1.1.

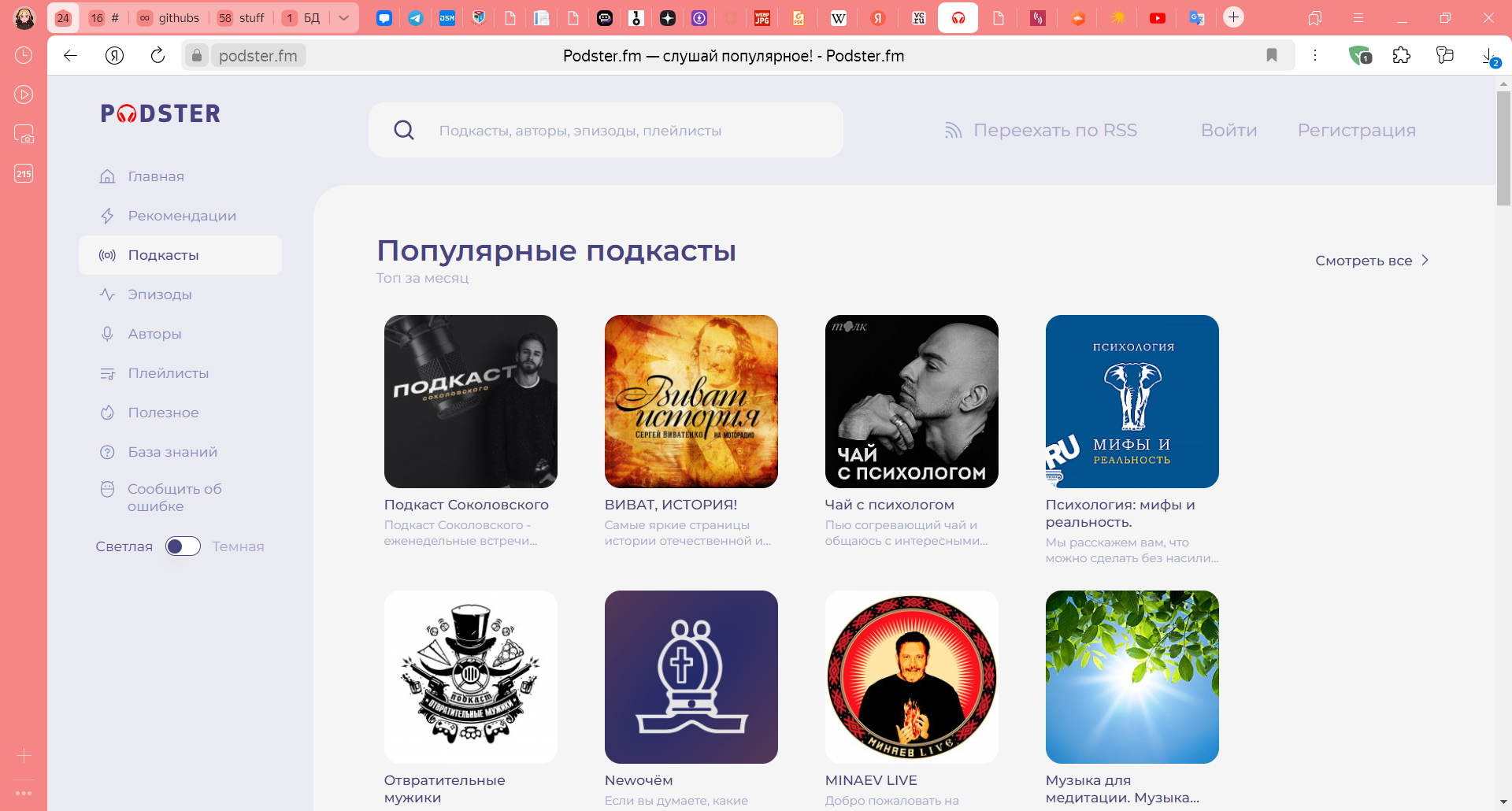


Рисунок 1.1 — Страница с подкастами на сайте «Podster.FM»

Второй аналог — «Castbox.FM», удобный мультифункциональный сервис, совмещающий в себе хостинг подкастов и платформу для их прослушивания. Для размещения аудиоблогов нужно установить программу «Castbox.FM» на компьютер или использовать ее веб-версию. После этого зайти в Creator Studio, указать свои данные и зарегистрироваться.

При правильном заполнении регистрационной заявки подкаст на Castbox появляется достаточно быстро. Ресурс предлагает для удобства поиска ссылки на размещенные подкасты. Добавлять их можно через «Настройки».

Сервис предоставляет встроенный плеер для аудиозаписей с отличной фишкой — установкой кликабельных тайм-кодов, чтобы слушатели могли перемотать запись до нужного места. На площадке также собирают рейтинги и подборки подкастов. Кроме этого, приложение дает пользователям возможность скачать подкасты для прослушивания офлайн. А для любителей слушать подкасты перед сном есть функция установки таймера сна, который автоматически останавливает воспроизведение спустя указанный промежуток времени.

Преимущества:

* бесплатное размещение подкастов;
* большое количество трафика;
* поддержка различных устройств;
* разнообразие жанров.

Недостатки:

* устаревший дизайн;
* слабая и скудная аналитика;
* неудобный редактор описания подкастов.

Вывод: «Castbox.FM» является мощной и удобной платформой для прослушивания подкастов, предлагая широкий выбор контента и доступность на различных устройствах. Несмотря на некоторые недостатки, приложение остается отличным выбором для любителей подкастов.

Интерфейс главной страницы сайта представлен на рисунке 1.2.

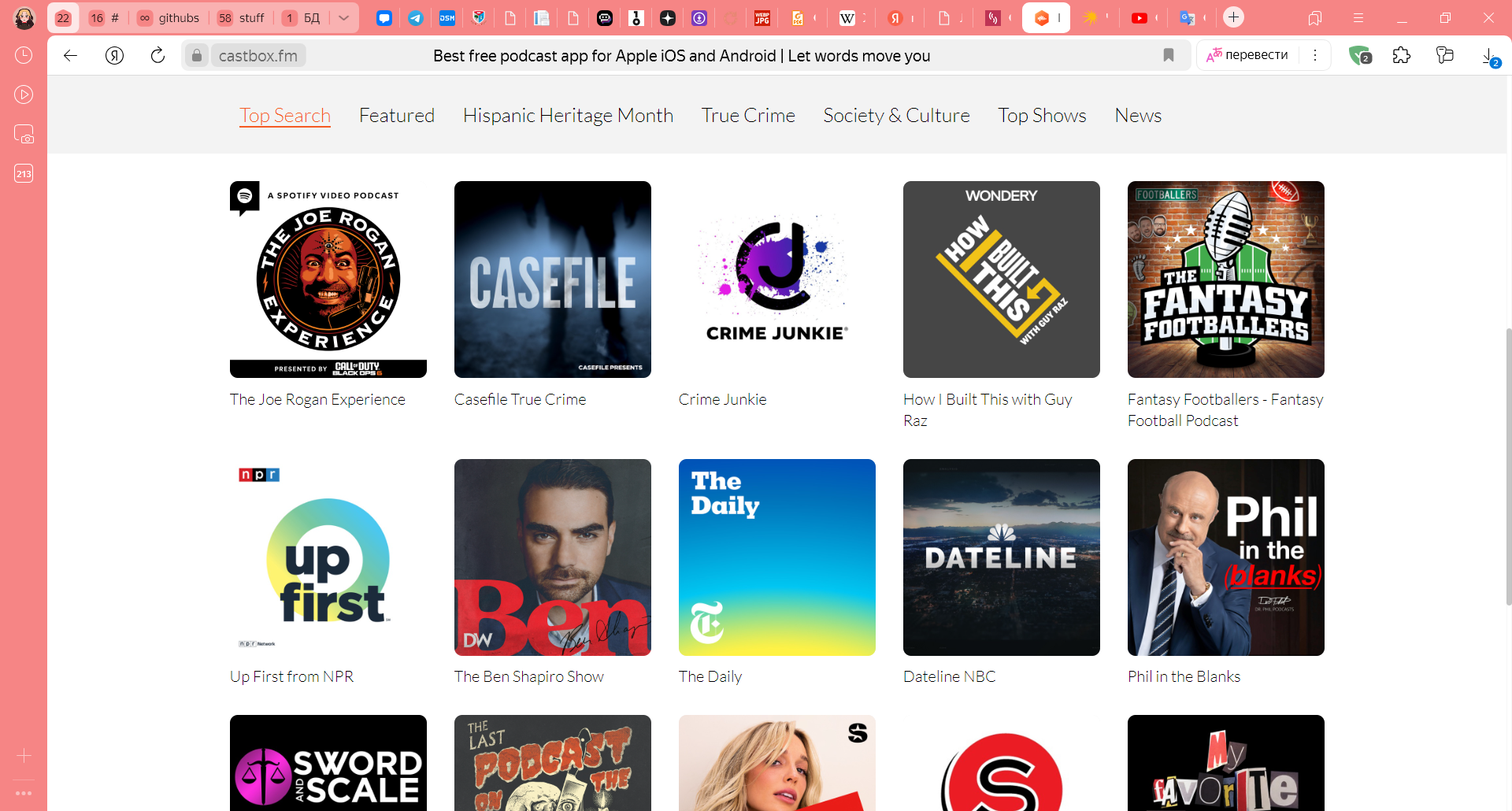


Рисунок 1.2 — Главная страница «Castbox.FM»

Третий аналог — «Mave.digital», популярный хостинг для публикации подкастов, бесплатный и без лимитов по загрузкам и прослушиваниям.

На сервисе, как и в предыдущих аналогах, создается отдельная страница для подкаста с обложкой, описанием и ссылками. Помимо этого, была анонсирована возможность подключать свой домен и настроить интеграцию с «Яндекс Метрикой». Есть отложенная публикация выпусков, а вот встроенного плеера для интеграции на сайт нет.

Здесь также есть расширенная аналитика, а еще — мониторинг чартов из Apple Podcasts, «Яндекс.Музыки» и PocketCasts. С недавнего времени в приложение была добавлена система монетизации — авторам доступны рекламные интеграции и в новых, и в уже опубликованных выпусках, а оплата производится через саму платформу.

Преимущества:

* современный и удобный интерфейс;
* нет ограничений по количеству выпусков или прослушиваний;
* подробная база знаний со всеми ресурсами;
* отображения рейтингов и количества оценок.

Недостатки:

* отсутствие встроенного плеера;
* ошибки верстки на больших экранах;
* не имеет мобильного приложения.

Вывод: «Mave.digital» является многообещающей платформой для создателей подкастов и рекламодателей, предлагая множество полезных функций и отсутствие лимитов на контент.

Интерфейс главной страницы сайта представлен на рисунке 1.3.

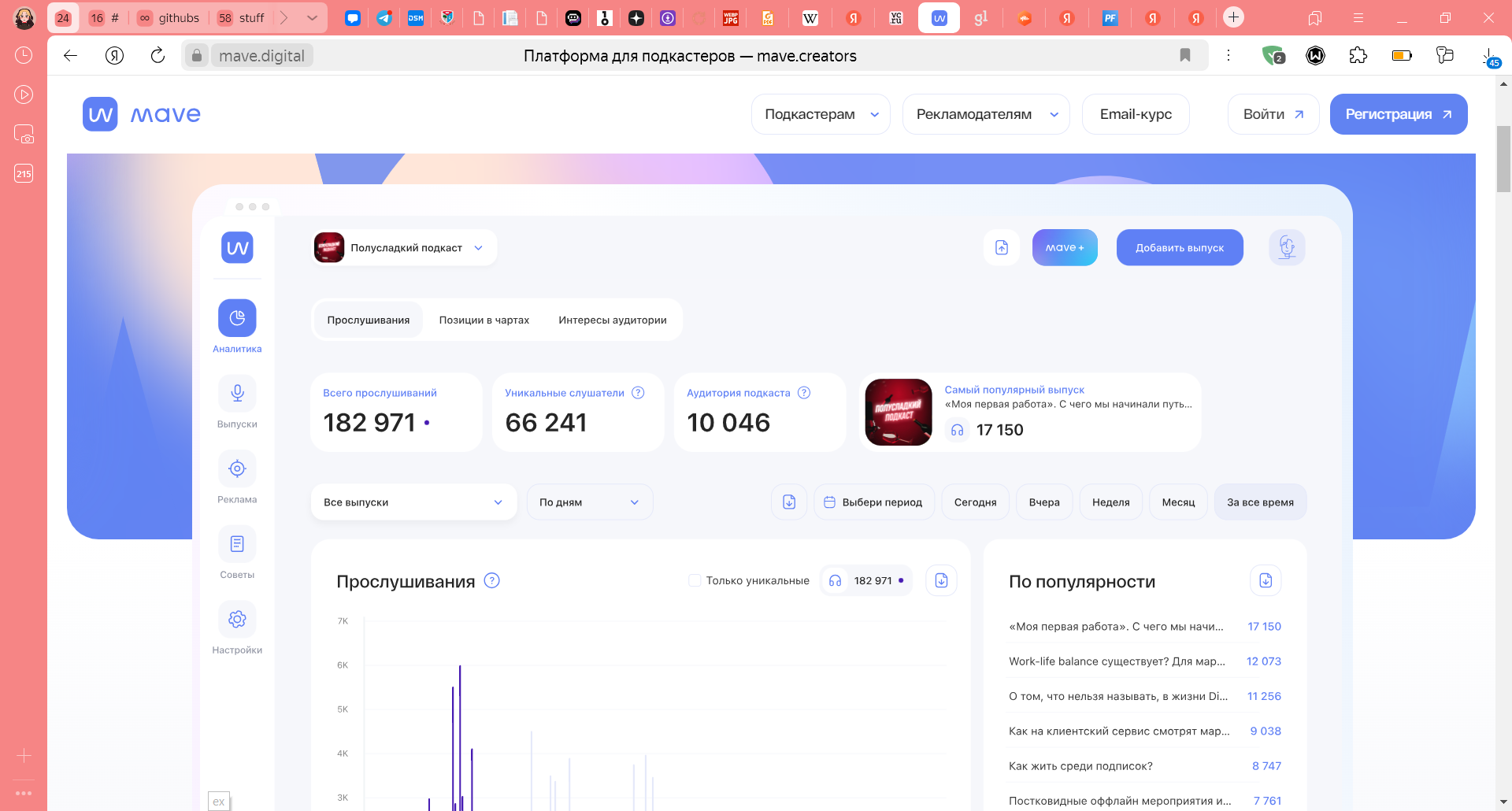


Рисунок 1.3 — Главная страница «Mave.digital»

Все рассмотренные аналоги имеют схожую тематику и функционал, но у каждого есть отличия, что и делает каждое из них уникальным

**1.2 Требования к проектируемому программному средству**

Целью данного курсового проекта является проектирование базы данных приложения для прослушивания подкастов. В качестве модели данных следует использовать реляционную модель. Проектирование необходимо произвести таким образом, чтобы конечные данные соответствовали общим требованиям к информации в базе данных.

Основываясь на анализе аналогичных решений из пункта 1.1, проектируемое программное средство должно включать следующие характеристики и функциональные возможности:

* определение ролей (администратор, пользователь);
* регистрация и авторизация пользователей;
* управление подкастами (добавление, удаление, изменение);
* управление темами (добавление, удаление, изменение);
* управление дикторами (добавление, изменение, удаление);
* просмотр информации о пользователях;
* поиск подкастов по названию темы и диктору;
* составление плейлистов.

Приложение должно предоставлять интуитивно понятный и визуально привлекательный интерфейс, который позволит пользователям искать и слушать подкасты. Графическое представление подкастов должно быть четким и удобным.

Доступ к данным должен осуществляться исключительно через соответствующие процедуры, обеспечивая тем самым контролируемый и безопасный доступ. Реализация базы данных предусматривает проведение импорта данных из XML файлов и экспорта данных в формат XML. Важным этапом является тестирование производительности базы данных на таблице, содержащей не менее 100 000 строк. Технологический компонент задачи включает в себя применение технологий Oracle, а также демонстрацию применения выбранной технологии использования средств диагностики в контексте разработанной базы данных.

**1.3 Вывод по разделу**

Таким образом, в данном разделе были изучены аналоги разрабатываемого проекта, а также описаны требования к функционалу проектируемой базы данных и прилагающегося к ней приложения.

2 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.

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.

Для реализации поставленной задачи разработана подключаемая база данных «PodSprout» в СУБД Oracle. Oracle Database — это объектно-реляционная система управления базами данных (СУБД) от компании Oracle. Она используется для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации.

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

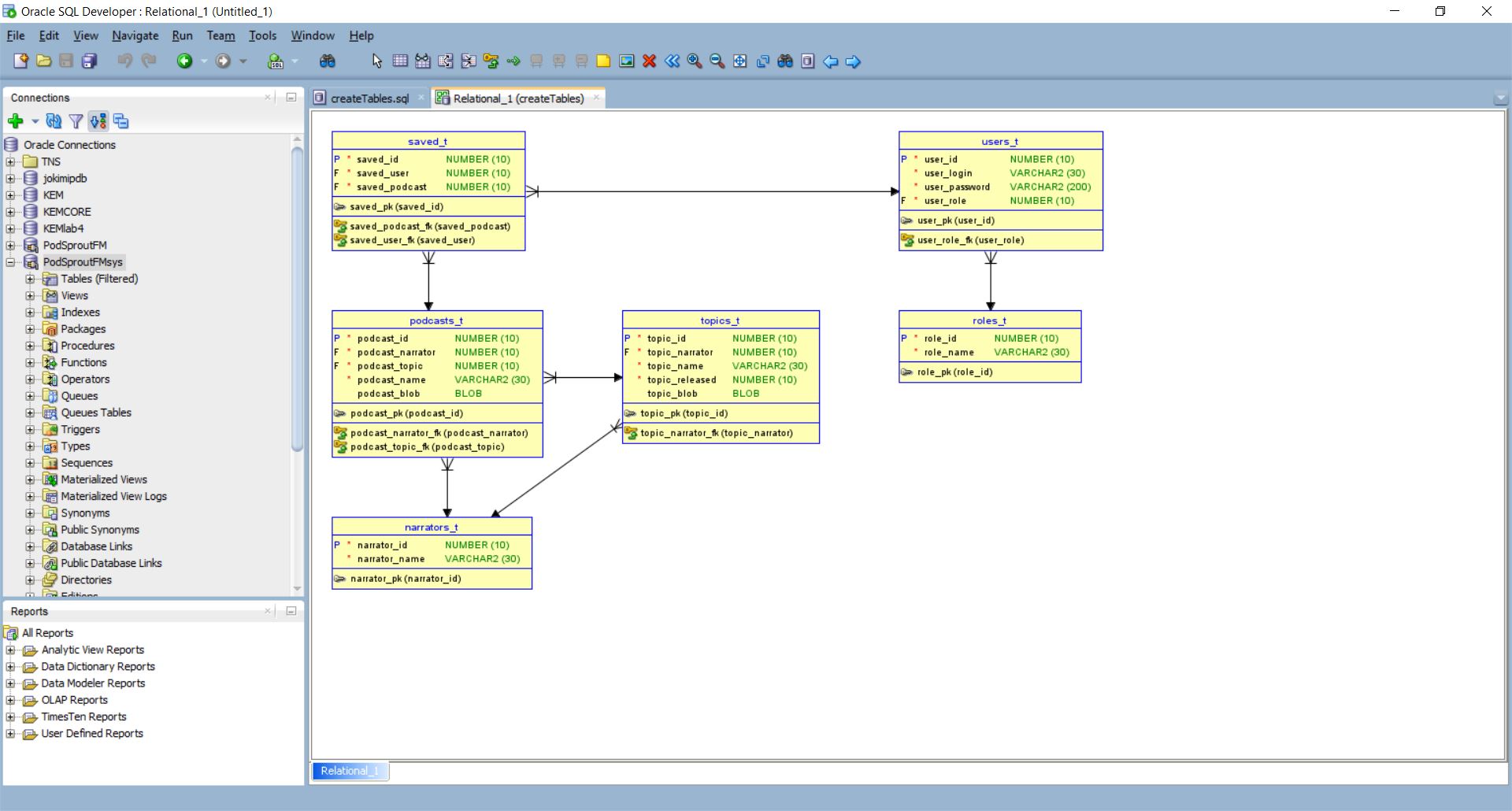


Рисунок 2.1 — Диаграмма таблиц базы данных «PodSprout»

База данных состоит из 6 таблиц, связанных между собой внешними ключами.

Таблица «ROLES\_T» хранит перечень ролей, которые могут иметь пользователи. Включает идентификатор роли и название роли.

Таблица «USERS\_T» содержит информацию о пользователях. Включает идентификатор пользователя, логин пользователя, пароль пользователя и идентификатор роли пользователя.

Таблица «NARRATORS\_T» содержит информацию о дикторах. Включает идентификатор диктора и имя диктора.

Таблица «TOPICS\_T» хранит информацию о темах, на которые дикторы выпускают подкасты. Включает идентификатор темы, идентификатор владельца темы (диктора), название темы, год выпуска темы и обложку темы.

Таблица «PODCASTS\_T» отвечает за хранение информации о подкастах, имеющихся в базе данных. Включает идентификатор подкаста, идентификатор автора подкаста (диктора), идентификатор темы, содержащего подкаст, имя подкаста и обложку подкаста.

Таблица «SAVED\_T» используется для сопоставления пользователя с понравившимся ему подкастом. Включает идентификатор экземпляра сохраненного подкаста, идентификатор пользователя, сохранившего подкаст и идентификатор сохраненного подкаста.

Скрипт создания всех таблиц представлен в Приложении Б.

Представленная структура базы данных обеспечивает эффективное и надежное хранение информации, необходимой для функционирования приложения для поиска и прослушивания подкастов. Все предусмотренные ограничения целостности и связи между таблицами гарантируют целостность данных и согласованность информации в системе.

3 Разработка объектов базы данных

**3.1 Таблицы**

Таблицы являются неотъемлемой частью любой реляционной базы данных. Краткая характеристика каждой из таблиц была предоставлена в разделе 2, а код создания можно увидеть в приложении Б.

Таблица «ROLES\_T» состоит из двух столбцов: role\_id, role\_name. Ее структура представлена в таблице 3.1.

Таблица 3.1 — Структура таблицы «ROLES\_T»

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| role\_id | number(10) | primary key |
| role\_name | varchar2(30) | not null |

Таблица «USERS\_T» состоит из четырех столбцов: user\_id, user\_login, user\_password, user\_role. Ее структура представлена в таблице 3.2.

Таблица 3.2 — Структура таблицы «USERS\_T»

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| user\_id | number(10) | primary key |
| user\_login | varchar2(30) | not null |
| user\_password | varchar2(200) | not null |
| user\_role | number(10) | not null, foreign key |

Таблица «TOPICS\_T» состоит из пяти столбцов: topic\_id, topic\_narrator, topic\_name, topic\_released, topic\_blob. Ее структура представлена в таблице 3.3.

Таблица 3.3 — Структура таблицы «TOPICS\_T»

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| topic\_id | number(10) | primary key |
| topic\_narrator | number(10) | not null, foreign key |
| topic\_name | varchar2(30) | not null |
| topic\_release | number(10) | not null |
| topic\_blob | blob | default empty\_blob() |

Таблица «PODCASTS\_T» состоит из пяти столбцов: podcast\_id, podcast\_blob, podcast\_narrator, podcast\_topic, podcast\_name. Структура представлена в таблице 3.4.

Таблица 3.4 — Структура таблицы «PODCASTS\_T»

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| podcast\_id | number(10) | primary key |
| podcast\_narrator | number(10) | not null, foreign key |

Продолжение таблицы 3.4

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| podcast\_topic | number(10) | not null, foreign key |
| podcast\_name | varchar2(30) | not null |
| podcast\_blob | blob | default empty\_blob() |

Таблица «NARRATORS\_T» состоит из двух столбцов: narrator\_id, narrator\_name. Ее структура представлена в таблице 3.5.

Таблица 3.5 — Структура таблицы «NARRATORS\_T»

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| narrator\_id | number(10) | primary key |
| narrator\_name | varchar2(30) | not null |

Таблица «SAVED\_T» состоит из трех столбцов: saved\_id, saved\_user, saved\_podcast. Ее структура представлена в таблице 3.6.

Таблица 3.6 — Структура таблицы «SAVED\_T»

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| saved\_id | number(10) | primary key |
| saved\_user | number(10) | not null, foreign key |
| saved\_podcast | number(10) | not null, foreign key |

Представленная структура базы данных обеспечивает эффективное и надежное хранение информации, необходимой для функционирования приложения для прослушивания подкастов. Предусмотренные ограничения целостности и связи между таблицами гарантируют целостность данных и согласованность информации в системе.

**3.2 Представления**

Для удобства и повышения производительности при обращении с базой данных были разработаны следующие представления.

Представление «user\_role\_full\_view» соединяет между собой таблицы «USERS\_T» и «ROLES\_T» и возвращает следующие столбцы: users\_t.user\_id, users\_t.user\_login, decr\_password(users\_t.user\_password) as decr, roles\_t.role\_name. Оно используется при выводе информации о пользователях в окне администратора, а также при авторизации пользователя для проверки роли.

Представление «narrator\_topic\_view» объединяет таблицы «TOPICS\_T» и «NARRATORS\_T» и возвращает следующие столбцы: narrators\_t.narrator\_name, topics\_t.topic\_id, topics\_t.topic\_name, topics\_t.topic\_release, topics\_t.topic\_blob. Оно используется для отображения плиток с темами.

Представление «narrator\_topic\_podcast\_view» соединяет между собой таблицы «TOPICS\_T», «NARRATORS\_T» и «PODCASTS\_T» и возвращает следующие столбцы: podcasts\_t.podcast\_id, narrators\_t.narrator\_name, podcasts\_t.podcast\_name, topics\_t.topic\_name, topics\_t.topic\_release, topics\_t.topic\_blob, podcasts\_t.podcast\_blob. Оно содержит наиболее полную информацию о каждом из подкастов.

Представление «narrator\_topic\_podcast\_user\_view», дополняет предыдущее представление, дополнительно объединяя его с таблицей «USERS\_T». Это используется для корректного отображения плиток с подкастами во вкладке «Мой плейлист».

Скрипт создания всех вышеописанных представлений можно посмотреть в приложении В.

**3.3 Пользователи**

В разработанной базе данных изначально присутствует один пользователь, который является хозяином схемы — sprout. Он имеет грант на подключение к базе данных, грант на выборку, добавление, изменение и удаление данных из всех таблиц, описанных в пункте 3.1, и из всех представлений, описанных в пункте 3.2. Он может создавать таблицы, процедуры, представления и сессии. Также ему выдан грант на выполнение процедур и функций стандартного пакета dbms\_crypto.

Код для создания пользователя-администратора представлен в листинге 3.1.

CREATE USER sprout IDENTIFIED BY sprout   
 DEFAULT TABLESPACE USERS   
 QUOTA UNLIMITED ON USERS;

Листинг 3.1 — Скрипт создания пользователя sprout

Код для выдачи грантов пользователю-администратору sprout представлен в листинге 3.2.

GRANT CONNECT TO sprout;

GRANT CREATE TABLE TO sprout;

GRANT CREATE SEQUENCE TO sprout;

GRANT CREATE VIEW TO sprout;

GRANT CREATE INDEXTYPE TO sprout;

GRANT CREATE PROCEDURE TO sprout;

GRANT CREATE TRIGGER TO sprout;

GRANT CREATE SESSION TO sprout;

GRANT CREATE JOB TO sprout;

GRANT EXECUTE ON sys.dbms\_crypto TO sprout;

Листинг 3.2 — Скрипт выдачи грантов пользователю sprout

Данные права доступа обеспечивают гибкость и возможности для выполнения администратором различных операций, необходимых для работы с базой данных.

**3.4 Функции**

Функция — объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Стоит отметить, что функции не могут содержать группы операторов DML и DDL. Данное приложении содержит две функции: EncryptionPassword и DecryptionPassword.

Прежде всего, рассмотрим функцию EncryptionPassword. Она позволяет зашифровать пароль пользователя. Она принимает один входной параметр: пароль пользователя для шифрования. Для шифрования был использован алгоритм AES с 128-разрядным ключом, метод сцепления CBC и механизм заполнения PKCS#5.

Для шифрования необходимо выбрать ключ. В приложении в качестве ключа используется строка «0710196810121972». Это значение относится к типу данных VARCHAR2. Чтобы использовать его в функции ENCRYPT, необходимо сначала преобразовать его к типу RAW. Для этого мы воспользуемся функцией STRING\_TO\_RAW встроенного пакета UTL\_I18N. Также вторым параметром указываем кодировку. Переменная lInVal типа VARCHAR2 также должна быть преобразована к типу RAW. Переменную, которая будет передаваться в качестве возвращаемого значения функции, требуется преобразовать в шестнадцатеричный формат. Для этого используем функцию RAWTOHEX.

Реализация данной функции представлена в листинге 3.3.

CREATE OR REPLACE FUNCTION EncryptionPassword

(pUserPassword IN USERS\_T.user\_password%TYPE)

RETURN USERS\_T.user\_password%TYPE

IS

lKey VARCHAR2(2000) := '0710196810121972';

lInVal VARCHAR2(2000) := pUserPassword;

lMod NUMBER := DBMS\_CRYPTO.encrypt\_aes128 + DBMS\_CRYPTO.chain\_cbc

+ DBMS\_CRYPTO.pad\_pkcs5;

lEnc RAW(2000);

BEGIN

lEnc := DBMS\_CRYPTO.encrypt(utl\_i18n.string\_to\_raw(lInVal,

'AL32UTF8'), lMod, \_i18n.string\_to\_raw(lKey, 'AL32UTF8'));

RETURN RAWTOHEX(lEnc);

EXCEPTION

WHEN VALUE\_ERROR THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Convertion error!');

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20002, 'No data for encryption!');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Error: ' || SQLERRM);

END EncryptionPassword;

Листинг 3.3 — Функция EncryptionPassword

Также была разработана функция DecryptionPassword, которая является противоположной функции EncryptionPassword. Она в качестве параметра принимает зашифрованный пароль и должна в результате своей работе вернуть его в расшифрованном виде. Используя тот же самый ключ, передаем его в качестве одного из параметров в функцию DECRYPT, перед этим преобразовав его в RAW при помощи функции STRING\_TO\_RAW. Также нужно передать в функцию параметр type и зашифрованный пароль типа RAW.

Результат функции преобразуем к строке при помощи функции RAW\_TO\_CHAR пакета UTL\_I18N. Реализацию этой функции можно увидеть в листинге 3.4.

CREATE OR REPLACE FUNCTION DecryptionPassword

(pUserPassword IN USERS\_T.user\_password%TYPE)

RETURN USERS\_T.user\_password%TYPE

IS

lKey VARCHAR2(2000) := '0710196810121972';

lInVal RAW(2000) := HEXTORAW(pUserPassword);

lMod NUMBER := DBMS\_CRYPTO.encrypt\_aes128 + DBMS\_CRYPTO.chain\_cbc

+ DBMS\_CRYPTO.pad\_pkcs5;

lDec RAW(2000);

BEGIN

lDec := DBMS\_CRYPTO.decrypt(lInVal, lMod,

utl\_i18n.string\_to\_raw(lKey, 'AL32UTF8'));

RETURN utl\_i18n.raw\_to\_char(lDec, 'AL32UTF8');

EXCEPTION

WHEN VALUE\_ERROR THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Convertion error!');

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20002, 'No data for decryption!');

WHEN INVALID\_NUMBER THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Wrong password format!');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Error: ' || SQLERRM);

END DecryptionPassword;

Листинг 3.4 — Функция DecryptionPassword

Таким образом, функции EncryptionPassword и DecryptionPassword обеспечивают безопасное управление паролями пользователей, используя надежные алгоритмы шифрования. Реализация шифрования с использованием AES и механизма CBC повышает уровень безопасности при хранении паролей в базе данных. Эти функции являются важным элементом системы безопасности приложения, позволяя защитить конфиденциальные данные пользователей от несанкционированного доступа.

**3.5 Хранимые процедуры**

Для управления данными через приложение администратор использует хранимые процедуры. Хранимая процедура — это объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Всего было разработано 25 процедур. Их назначение будет рассмотрено ниже.

Была разработана хранимая процедура RegisterUser, с помощью которой происходит регистрация пользователей в приложении. Эта процедура принимает два параметра — логин пользователя (pUserLogin), которого нужно зарегистрировать, и пароль от аккаунта этого пользователя (pUserPassword). В случае, если указанный логин уже есть в базе данных, поднимается исключение. Если исключения не произошло, то строка успешно вставляется в таблицу «USERS\_T», зашифровав пароль в процессе.

Процедура CheckRole позволяет найти роль пользователя и выгружает её для дальнейших манипуляций. Принимает один входной параметр, а именно — логин пользователя (pUserLogin), а также один выходной параметр — название роли пользователя (oUserRole).

Для авторизации пользователя в приложении разработана процедура LogInUser. У нее есть два входных параметра: логин пользователя (p\_user\_login) и пароль пользователя (pUserPassword). Если указанная комбинация логина и пароля отсутствует в таблице «USERS\_T», то поднимается исключение. Если исключения не произошло, то в приложение передается три выходных параметра, которые пригодятся далее в разработке логики приложения и будут занесены в поля статического класса DataWorker: oUserId (идентификатор пользователя, прошедшего авторизацию), oUserLogin (логин пользователя), oUserRole (роль пользователя).

Процедура SearchUser позволяет найти информацию о пользователе и выгружает данные о нем из таблицы «USERS\_T». Она используется для отображения данных в меню «Настройки». Процедура принимает один входной параметр — логин пользователя (pUserLogin), а также два выходных параметра — логин пользователя (pUserLogin) и расшифрованный пароль пользователя (pUserPassword).

Процедура UpdateUserLogin позволяет изменить логин пользователя. Данная процедура принимает два входных параметра: старый логин пользователя (pUserLogin) и новый логин пользователя (pNewUserLogin), которым следует заменить старый.

Процедура UpdateUserPassword позволяет изменить пароль пользователя. Эта принимает два входных параметра — старый пароль пользователя (pUserPassword) и расшифрованный новый пароль пользователя (pNewUserPassword), который заменит в таблице старый и будет зашифрован в процессе.

Процедура AddNarrator служит для добавления нового диктора в базу данных. Данная процедура принимает один параметр — имя диктора (pNarratorName), который будет добавлен в базу данных.

Процедура AddTopic для добавления новой темы в базу данных. Эта процедура принимает три параметра — имя диктора (pNarratorName), которому принадлежит создаваемая тема, название создаваемой темы (pTopicName) и год выпуска темы (pTopicRelease). Так как в хранимых процедурах нельзя передавать данные объемом более 32 Кб, изображение, служащее обложкой темы, передается отдельно через операцию INSERT.

Процедура AddPodcast служит для добавления нового подкаста в базу данных. Эта процедура принимает три параметра — имя диктора (pNarratorName), которому принадлежит создаваемый подкаст, название темы (pTopicName), к которой будет относится подкаст, и название добавляемого подкаста (pPodcastName). Аудиофайл подкаста передается отдельно через операцию INSERT.

Процедура SavePodcast служит для добавления пользователем подкаста в свой плейлист. Эта процедура принимает два параметра — идентификатор пользователя (pUserId) и идентификатор подкаста (pPodcastId).

Процедура UpdateNarrator отвечает за замену старого имени диктора на новое. Она принимает эти имена как параметры: pOldNarrator и pNewNarrator.

Процедура UpdateTopicName позволяет обновить название темы. Она принимает два параметра: идентификатор темы (pTopicId) и имя темы (pNewName), которым будет заменено старое.

Процедура UpdateTopicYear позволяет обновить год выпуска темы. Она принимает два параметра: идентификатор темы (pTopicId) и год выпуска темы (pNewYear), которым будет заменен старый.

Процедура UpdatePodcastName позволяет обновить название уже существующего подкаста. Она принимает два параметра: идентификатор подкаста (pPodcastId) и новое название для подкаста (pNewName).

Процедура DeleteNarrator отвечает за удаление диктора из базы данных. Она принимает один параметр: идентификатор удаляемого диктора (pId). При удалении диктора удалятся все связанные с ним темы, подкасты и экземпляры подкастов, сохраненных в плейлисты пользователей.

Процедура DeleteTopic отвечает за удаление темы из базы данных. Она принимает один параметр: идентификатор удаляемой темы (pId). При удалении темы удалятся все связанные с ней подкасты и экземпляры подкастов, сохраненных в плейлисты пользователей.

Процедура DeletePodcast отвечает за удаление подкаста из базы данных. Она принимает один параметр: идентификатор удаляемого подкаста (pId). При удалении подкаста удалятся все связанные с ним экземпляры подкастов, сохраненных в плейлисты пользователей.

Процедура RemovePodcast отвечает за удаление подкаста из плейлиста конкретного пользователя. Она принимает два параметра: идентификатор пользователя (pUserId) и идентификатор удаляемого из плейлиста подкаста (pPodcastId).

С помощью процедуры DeleteUser пользователь может удалить свой аккаунт. Процедура принимает один параметр: логин пользователя (pUserLogin). При этом удалятся все данные о составе его плейлиста.

Листинги процедур представлены в приложении Г.

**3.6 Вывод по разделу**

Таким образом, разработанные объекты базы данных обеспечивают не только структуру для хранения данных, но и функциональность, способствующую эффективному управлению приложением и взаимодействию с ним.

4 Описание процедур импорта и экспорта

Экспорт в контексте баз данных обозначает процесс сохранения данных из базы данных во внешний файл или другой хранилище данных. Это может быть использовано для резервного копирования, обмена данными между системами, создания отчетов или анализа данных в сторонних инструментах. В результате экспорта, данные из базы данных становятся доступными в структурированном формате, который можно легко передавать или анализировать вне контекста базы данных.

Для реализации процедур экспорта и импорта таблиц в XML потребовалось создать директорий с желаемым расположением при подключении под пользователем SYS к инстансу и выдать пользователю, владеющему базой данных, грант на чтение и запись из этого директория, чтобы обеспечить администратору доступ к необходимым ресурсам для выполнения операций. Это представлено в листинге 4.1.

CREATE OR REPLACE DIRECTORY cwdir as '/opt/oracle/cw';

SELECT \* FROM dba\_directories WHERE directory\_name='CWDIR';

GRANT READ, WRITE ON DIRECTORY CWDIR to sprout;

Листинг 4.1 — Создание директория CWDIR

В рамках проекта нужно было разработать процедуру, экспортирующую данные из таблицы, содержащей 100 000 строк, в XML-документ. С этой целью разработана процедура UsersExport. Данная процедура объявляет курсор, выбирающий данные из представления user\_role\_full\_view, а также переменную, хранящую XML-документ и переменную для хранения текущей строки, извлекаемой из курсора. Данные из курсора преобразуются в XML, после чего записываются в файл UserExport.xml в указанной директории CWDIR.

Вторая процедура для экспорта данных, PodcastsExport, записывает данные из таблицы «PODCASTS\_T» в XML-документ. Данная процедура объявляет переменную-курсор типа sys\_refcursor, а также переменную, хранящую XML-документ. Далее курсор выбирает данные из таблицы. Данные из курсора преобразуются в XML, после чего записываются в файл PodcastExport.xml в указанной директории CWDIR.

Экспортированные данные позже можно будет использовать для резервного копирования, обмена данными между системами, создания отчетов или анализа данных в сторонних инструментах. В результате экспорта, данные из базы данных становятся доступными в структурированном формате, который можно легко передавать или анализировать вне контекста базы данных.

Также нужно было написать процедуру, которая бы, наоборот, импортировала существующие данные из XML документа в таблицу базы данных. В этом случае данные вставляются в таблицу «NARRATORS\_T». Эта процедура извлекает данные из XML-файла NarratorsImport.xml, который находится в директории CWDIR. Сначала содержимое файла преобразуется в тип XML, далее данные извлекаются из XML, а затем результирующий набор преобразуется в таблицу. Таким образом, каждый элемент <NAME> из XML будет вставлен в таблицу как новая запись. Эти данные можно использовать для загрузки данных из файлов различных форматов или из других баз данных в существующую базу данных. Импорт позволяет обновлять, дополнять или восстанавливать данные в базе данных с учетом изменений внешних источников.

Листинги вышеописанных процедур представлены в приложении Г.

Таким образом, был рассмотрен вопрос о наполнении таблиц данными (импорте) и выгрузке данных из таблицы (экспорте), предложив решение на основе использования хранимых процедур.

5 Тестирование производительности

Тестирование производительности является важным этапом разработки, поскольку позволяет определить, насколько хорошо база данных может обрабатывать запросы и как быстро она может возвращать результаты. Оно проводилось на таблице «USERS\_T», которая заполнялась на 100 000 строк при помощи процедуры, представленной в листинге 5.1.

CREATE OR REPLACE PROCEDURE Insert100KUsers IS

usernames SYS.ODCIVARCHAR2LIST := SYS.ODCIVARCHAR2LIST();

passwords SYS.ODCIVARCHAR2LIST := SYS.ODCIVARCHAR2LIST();

BEGIN

FOR i IN 1 .. 100000 LOOP

usernames.EXTEND;

passwords.EXTEND;

usernames(usernames.COUNT) := 'user' || i;

passwords(passwords.COUNT) := 'pass' || i;

IF MOD(i, 1000) = 0 THEN

BEGIN

RegisterUsersBatch(usernames, passwords);

COMMIT;

usernames.DELETE;

passwords.DELETE;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

RAISE\_APPLICATION\_ERROR(-20030, 'Error at

iteration ' || i || ': ' || SQLERRM);

END;

END IF;

END LOOP;

IF usernames.COUNT > 0 THEN

BEGIN

RegisterUsersBatch(usernames, passwords);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

RAISE\_APPLICATION\_ERROR(-20031, 'Err: ' || SQLERRM);

END;

END IF;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

RAISE\_APPLICATION\_ERROR(-20032, 'Err: ' || SQLERRM);

END Insert100KUsers;

Листинг 5.1 — Процедура Insert100KUsers

Эта процедура заполняет таблицу «USERS\_T» логинами и соответствующими им паролями пользователей, формируемыми по шаблону. Для создания списком используются переменные типа SYS.ODCIVARCHAR2LIST, который позволяет хранить список строк. Далее в цикле от 1 до 100 000 происходит добавление новых логинов и паролей в соответствующие списки. Каждые 1000 итераций вызывается вспомогательная процедура RegisterUsersBatch для вставки текущей партии пользователей, после успешного выполнения списки очищаются. После завершения цикла проверяется, остались ли незарегистрированные пользователи, далее при необходимости выполняется заключительная вставка.

Как было указано выше, процедура Insert100KUsers для регистрации новых пользователей использует дополнительную процедуру RegisterUsersBatch, которая осуществляет вставку строк в таблицу «USERS\_T» партиями. Она представлена в листинге 5.2.

CREATE OR REPLACE PROCEDURE RegisterUsersBatch (

pUsernames IN SYS.ODCIVARCHAR2LIST, pPasswords IN

SYS.ODCIVARCHAR2LIST) IS

BEGIN

IF pUsernames.COUNT != pPasswords.COUNT THEN

RAISE\_APPLICATION\_ERROR(-20020, ‘logins != passwords.');

END IF;

FOR i IN 1 .. pUsernames.COUNT LOOP

BEGIN

INSERT INTO users\_t (user\_login, user\_password,

user\_role) (UPPER(pUsernames(i)),

EncryptionPassword(UPPER(pPasswords(i))), 1);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

RAISE\_APPLICATION\_ERROR(-20021, 'Username "' ||

pUsernames(i) || '" is already taken.');

WHEN VALUE\_ERROR THEN

RAISE\_APPLICATION\_ERROR(-20022, 'Invalid data format

for "' || pUsernames(i) || '".');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20023, 'Error for "' ||

pUsernames(i) || '": ' || SQLERRM);

END;

END LOOP;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

RAISE\_APPLICATION\_ERROR(-20024, 'Error: ' || SQLERRM);

END RegisterUsersBatch;

Листинг 5.2 — Процедура RegisterUsersBatch

Данная процедура принимает два параметра: pUsernames (список имен пользователей) и pPasswords (список паролей). Сначала проверяется, что количество логинов и паролей совпадает, после чего происходит вставка данных в таблицу, при этом пароли шифруются с помощью функции EncryptionPassword. В конце процедуры происходит коммит, чтобы сохранить изменения в базе данных.

6 Описание технологии и ее применения в базе данных

**6.1 Средства диагностики**

Несмотря на то, что Oracle предоставляет достаточно богатый функционал для разработчика, его отличительной чертой является также ограниченность в бесплатном вспомогательном программном обеспечении, в отличие от MS SQL. Из ограниченного круга средств было выбрано ПО Database Performance Analyzer, разработанное компанией SolarWinds, поскольку с ним предоставляется 14-тидневный период для оценки ПО, в рамках которого можно ознакомиться с его функционалом.

Программное обеспечение SolarWinds Database Performance Analyzer for Oracle — это инструмент мониторинга производительности сервера Oracle, помогающий выявлять наиболее опасные для конечных пользователей проблемы. Благодаря многомерному анализу SolarWinds Database Performance Analyzer for Oracle помогает выявлять корневые причины неполадок. Продукт превосходит обычные решения для наблюдения за состоянием сервера; это единый инструмент администраторов, разработчиков и менеджеров баз данных, которым необходимо экономично и эффективно совершенствовать работу серверов SQL. Интерфейс SolarWinds Database Performance Analyzer изображен на рисунке 6.1.

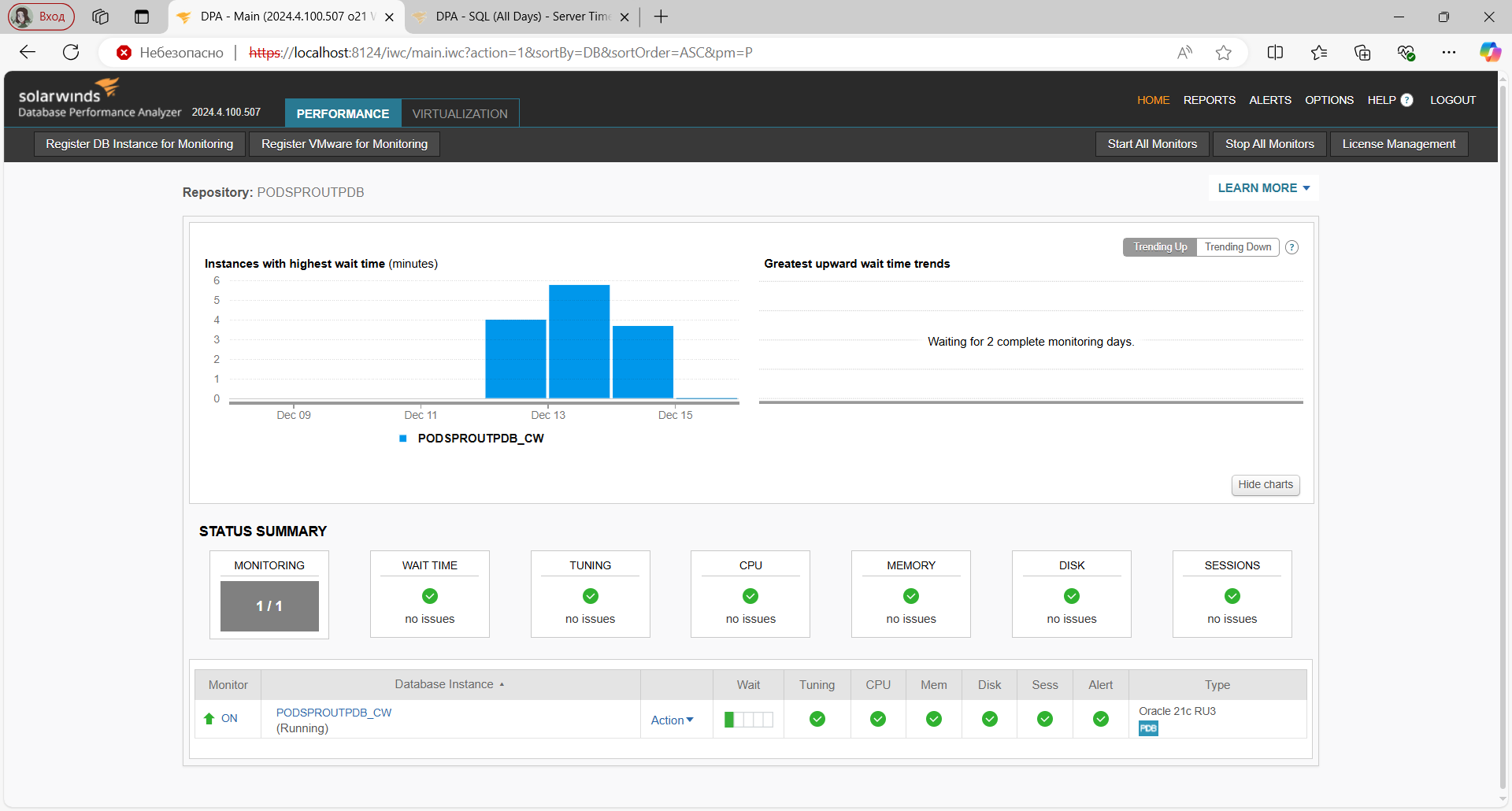


Рисунок 6.1 — Интерфейс SolarWinds DPA

Анализ и оптимизация производительности сервера и приложений выполняются на базе систем запросов, сеансов и хранилищ; проверяются базовые показатели, оповещения и данные ввода/вывода в целях сопоставления ресурсов со временем ответа SQL-сервера. Благодаря безагентской архитектуре и независимости от пакетов Oracle Packs продукт SolarWinds Database Performance Analyzer устанавливается и развертывается за несколько минут на любом сервере Windows, Linux или UNIX.

Основные преимущества SolarWinds DPA:

* анализ истории времени отклика и статистики сервера;
* отображение задержек во времени ответа на текущий период, а также за недели и месяцы;
* проактивное уведомление администраторов о проблемах по электронной почте или через SNMP-ловушку;
* автоматическая подготовка графических отчетов по расписанию и рассылка на электронную почту;
* поддержка браузеров Internet Explore, Mozilla Firefox и Google Chrome;
* менее 1% нагрузки на отслеживаемые серверы.

Благодаря многомерному анализу SolarWinds Database Performance Analyzer помогает выявлять корневые причины неполадок. Продукт превосходит обычные решения для наблюдения за состоянием сервера; это единый инструмент администраторов, разработчиков и менеджеров баз данных, который позволяет экономично и эффективно совершенствовать работу серверов SQL.

В горизонтальном блоке плиток по центру рисунка 6.1 можно увидеть, что DPA позволяет отслеживать следующие показатели: количество отслеживаемых баз данных, количество БД с критическим временем задержки, очереди запросов, состояния процессоров, состояние памяти, состояние дисков, сессии.

Во вкладке Alerts (Оповещения), изображенной на рисунке 6.2, можно отслеживать статус ошибок и неисправностей, создавать группы оповещений, задать случаи, когда необходимо оповещать о неисправности по электронной почте.

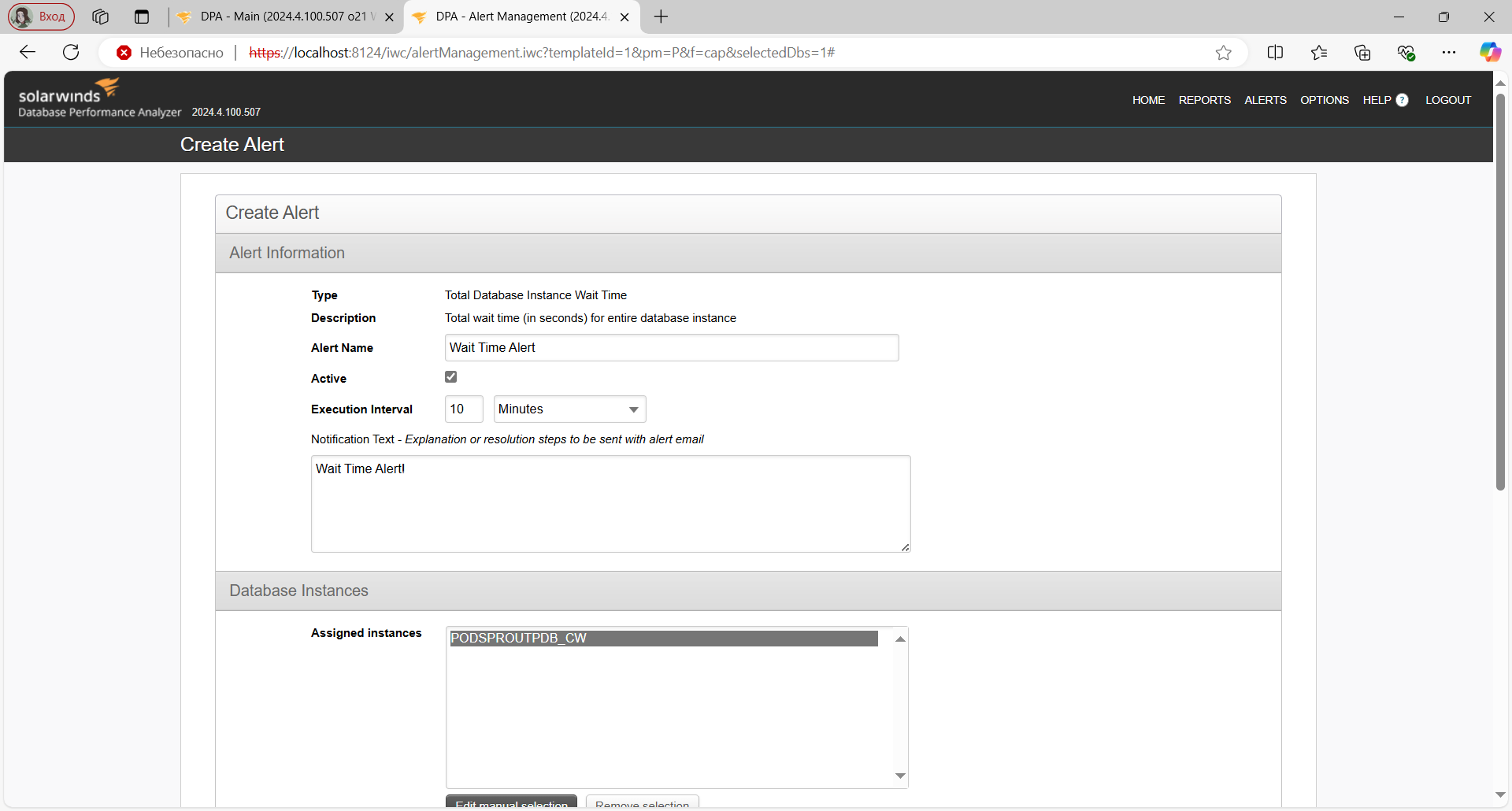


Рисунок 6.2 — Вкладка Alerts

На этой же странице можно протестировать создаваемое оповещение перед его сохранением. При этом появится окно, в котором будет указано состояние теста, название используемого инстанса, время выполнения теста и описание оповещения, если оно было задано. Также в тестовом режиме сообщение на электронную почту не отправляется. Результат теста оповещения представлен на рисунке 6.3.

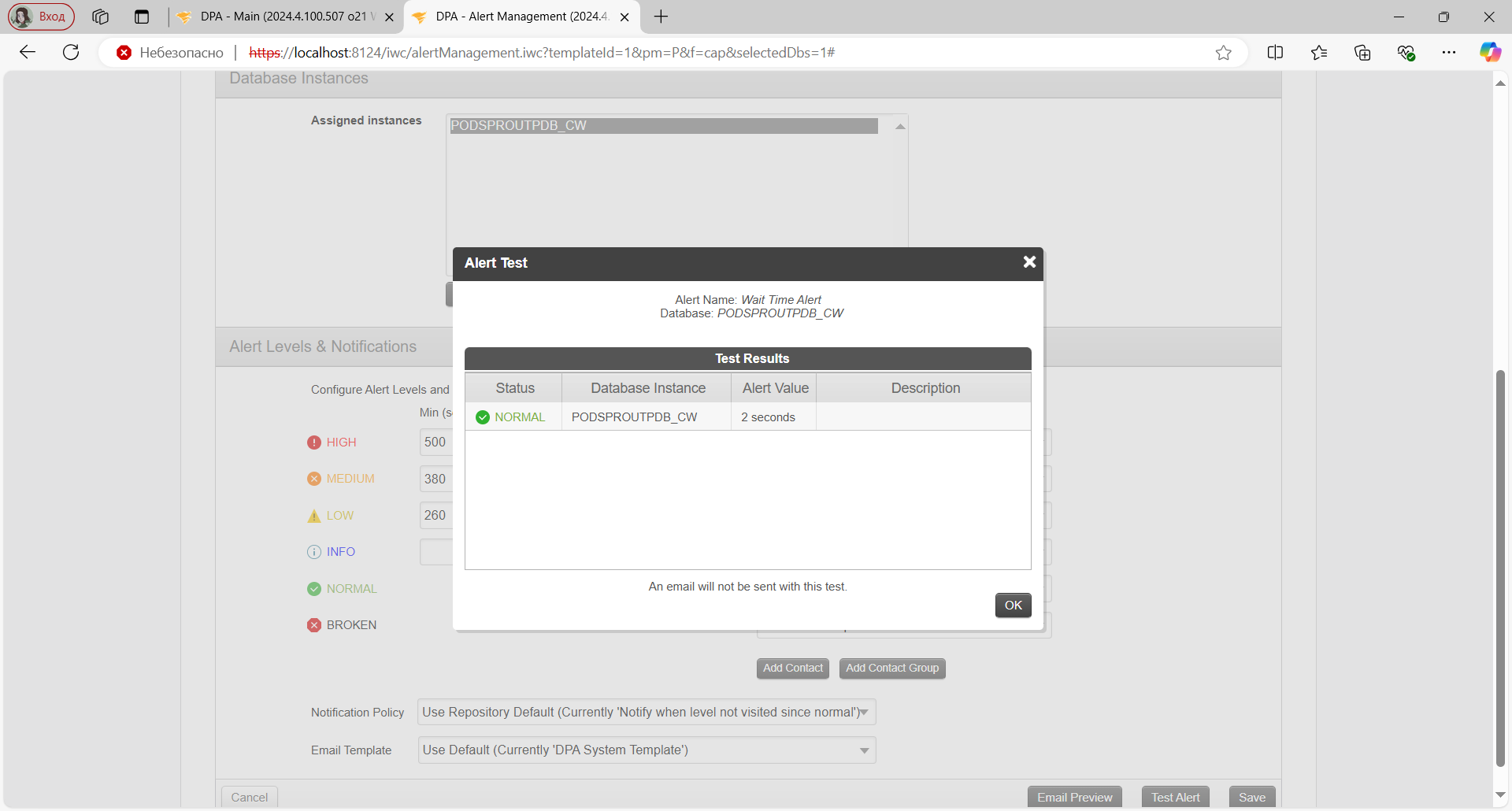


Рисунок 6.3 — Тестирование оповещения

Программа за несколько секунд показывает текущий статус сеанса, чтобы администраторы могли мгновенно реагировать на проблемы производительности. На первой вкладке подробной информации (Trends) находится временная диаграмма, отображающая время ожидания запросов, как показано на рисунке 6.4.

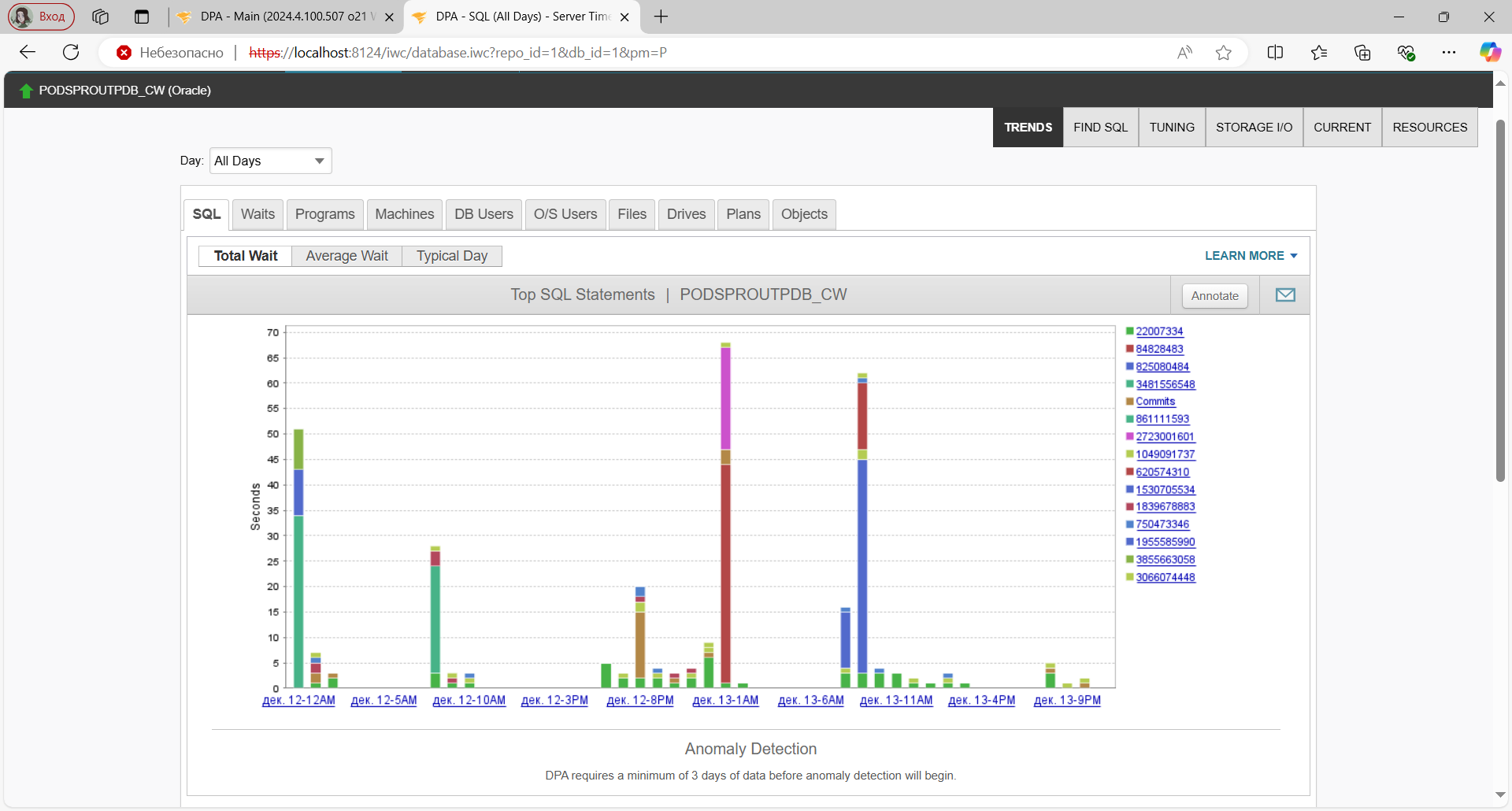


Рисунок 6.4 — Диаграмма времени ожидания запросов

На второй вкладке (Find SQL) можно просматривать список запросов к базе данных. При этом можно посмотреть все запросы за определенный промежуток времени или использовать фильтры (можно выбрать пользователя базы данных, который выполнял запрос, программу или машину). Интерфейс этой вкладки продемонстрирован на рисунке 6.5.

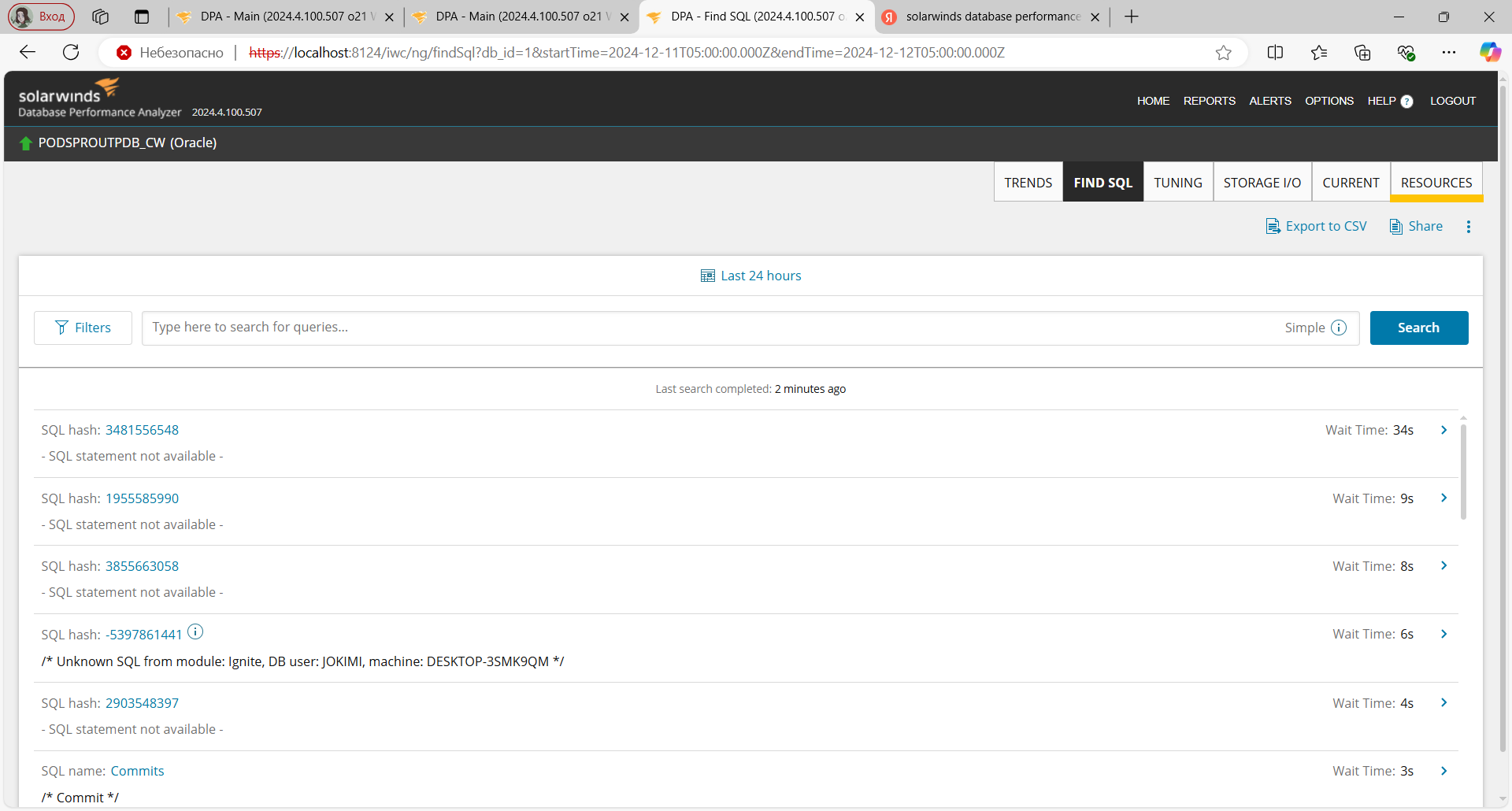


Рисунок 6.5 — Список запросов за последние 24 часа

На третьей вкладке (Tuning) можно увидеть более подробную информацию о запросе, в том числе время ожидания выполнения запроса, текст запроса и программа, через которую он был выполнен. Внешний вид вкладки можно увидеть на рисунке 6.6.

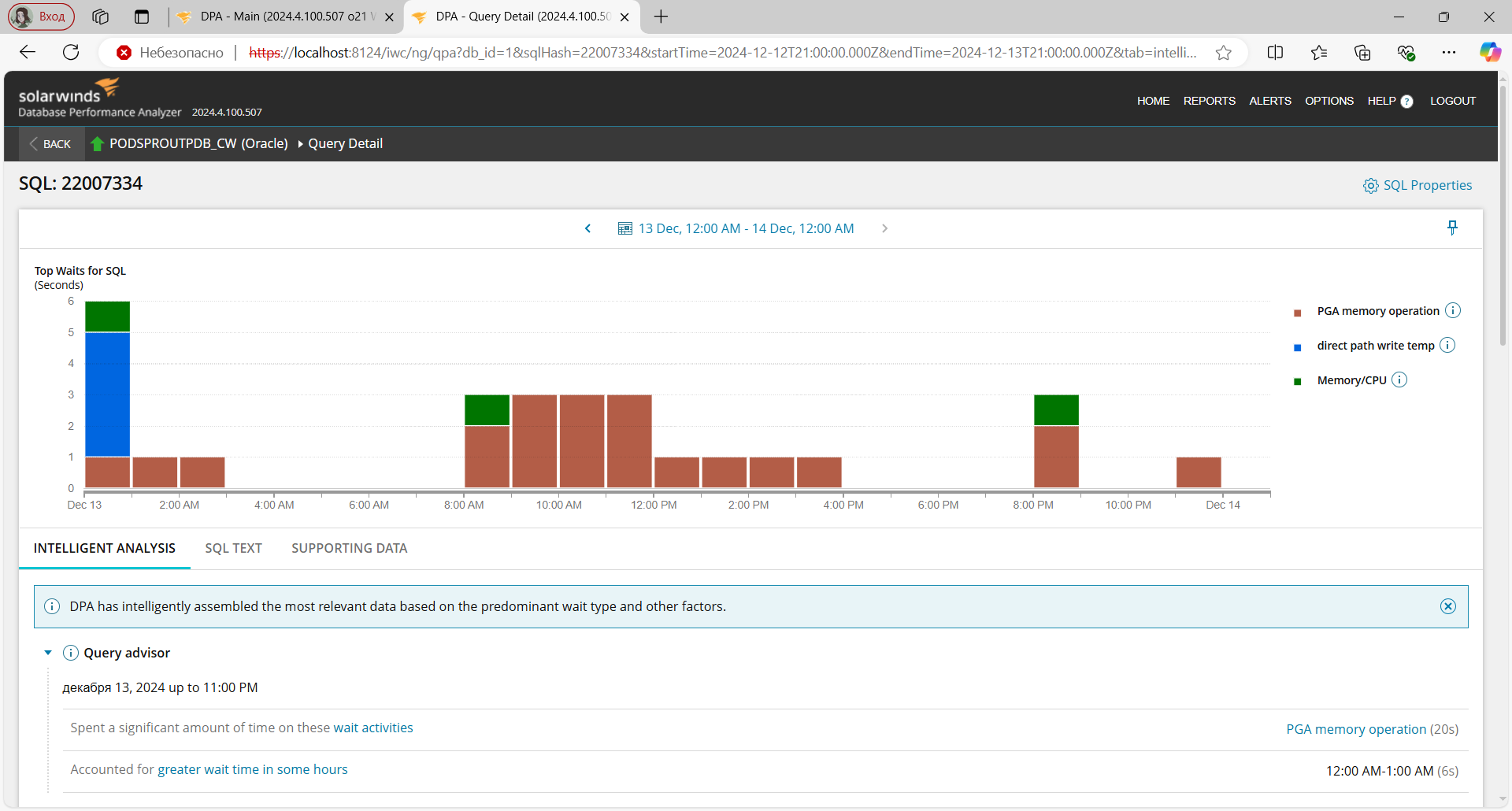


Рисунок 6.6 — Подробная информация о запросе

На четвертой вкладке (Storage I/O) можно находится анализ ввода-вывода хранилища. Здесь администраторы могут увидеть влияние хранилища на рабочую нагрузку базы данных, а также получить представление о том, как проблемы хранилища, такие как задержка и производительность диска, влияют на время отклика базы данных и влияют на конечного пользователя приложения. Информация о чтении/записи файлов базы данных представлено на рисунке 6.7.

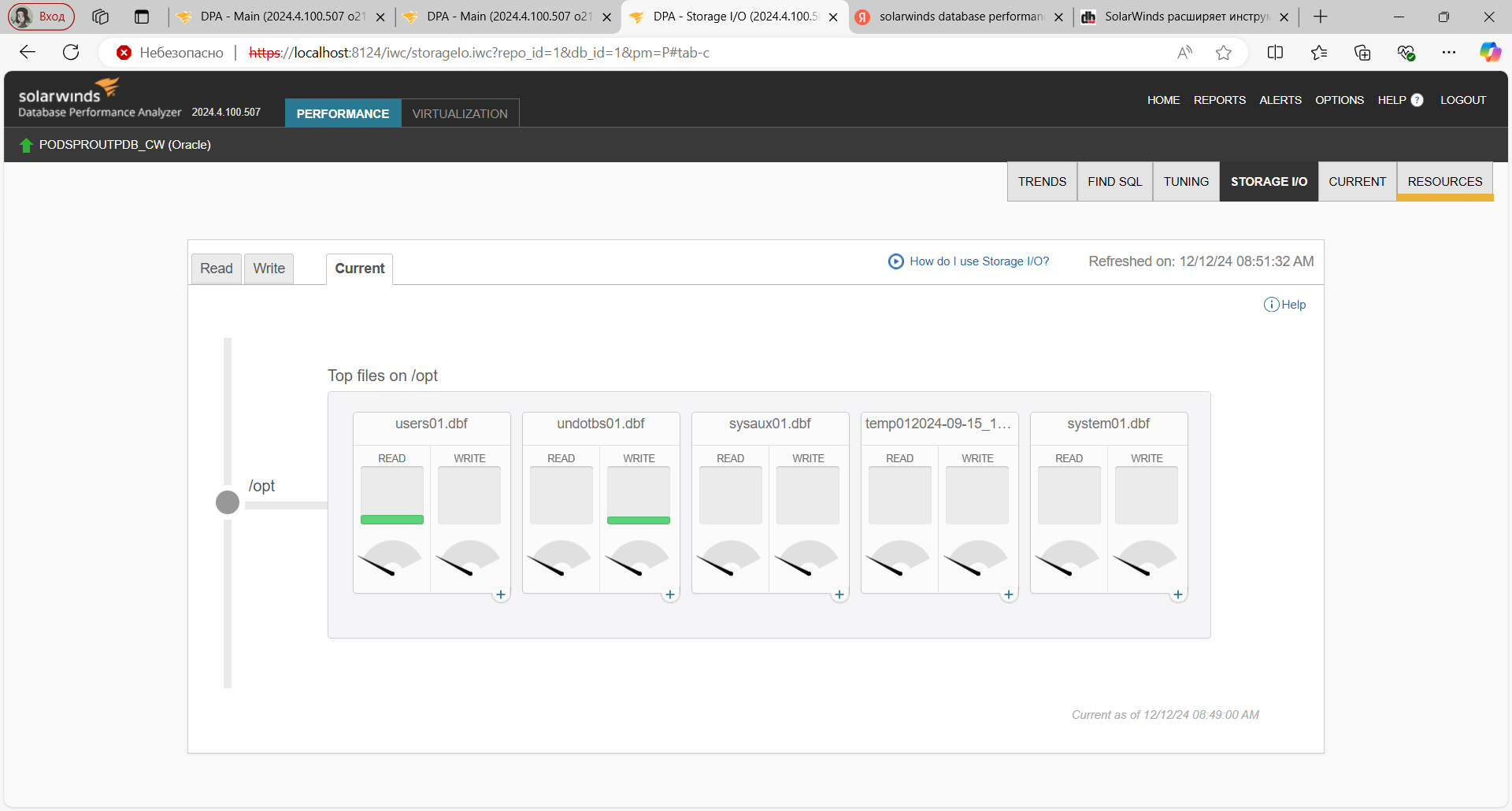


Рисунок 6.7 — Чтение/запись файлов БД

На пятой вкладке (Current) можно содержится информация о тех запросах, выполнение которых занимает наибольшее количество времени, о текущих активных и заблокированных сессиях и об использовании ресурсов. Интерфейс вкладки можно увидеть на рисунке 6.8.

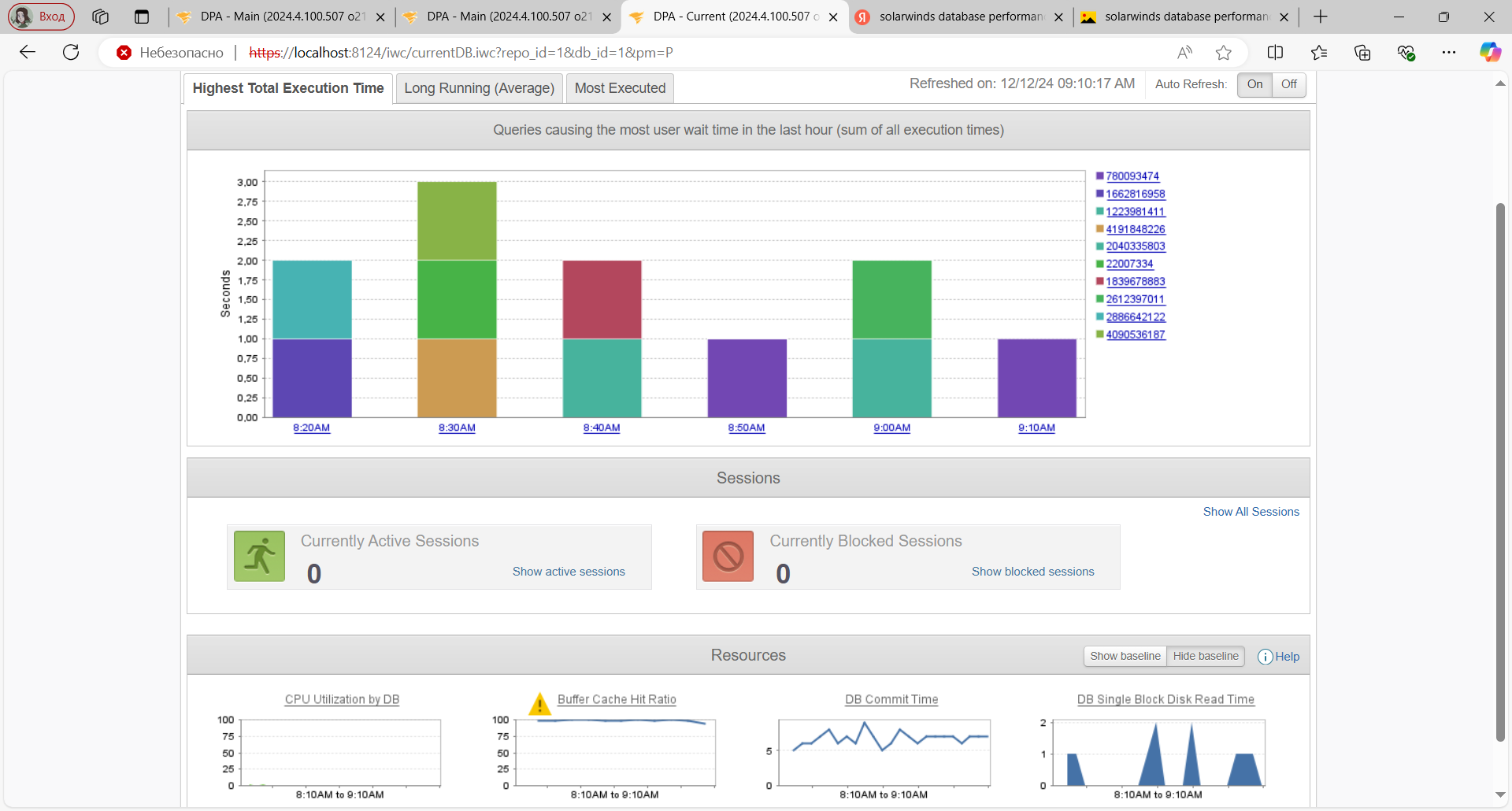


Рисунок 6.8 — Вкладка «Current»

На шестой вкладке (Resources) можно увидеть более подробную информацию об использовании ресурсов и места на диске, соединениях, подключении к сети, сессиях, времени ожидания за определенный промежуток времени. Внешний вид данной вкладки представлен на рисунке 6.9.

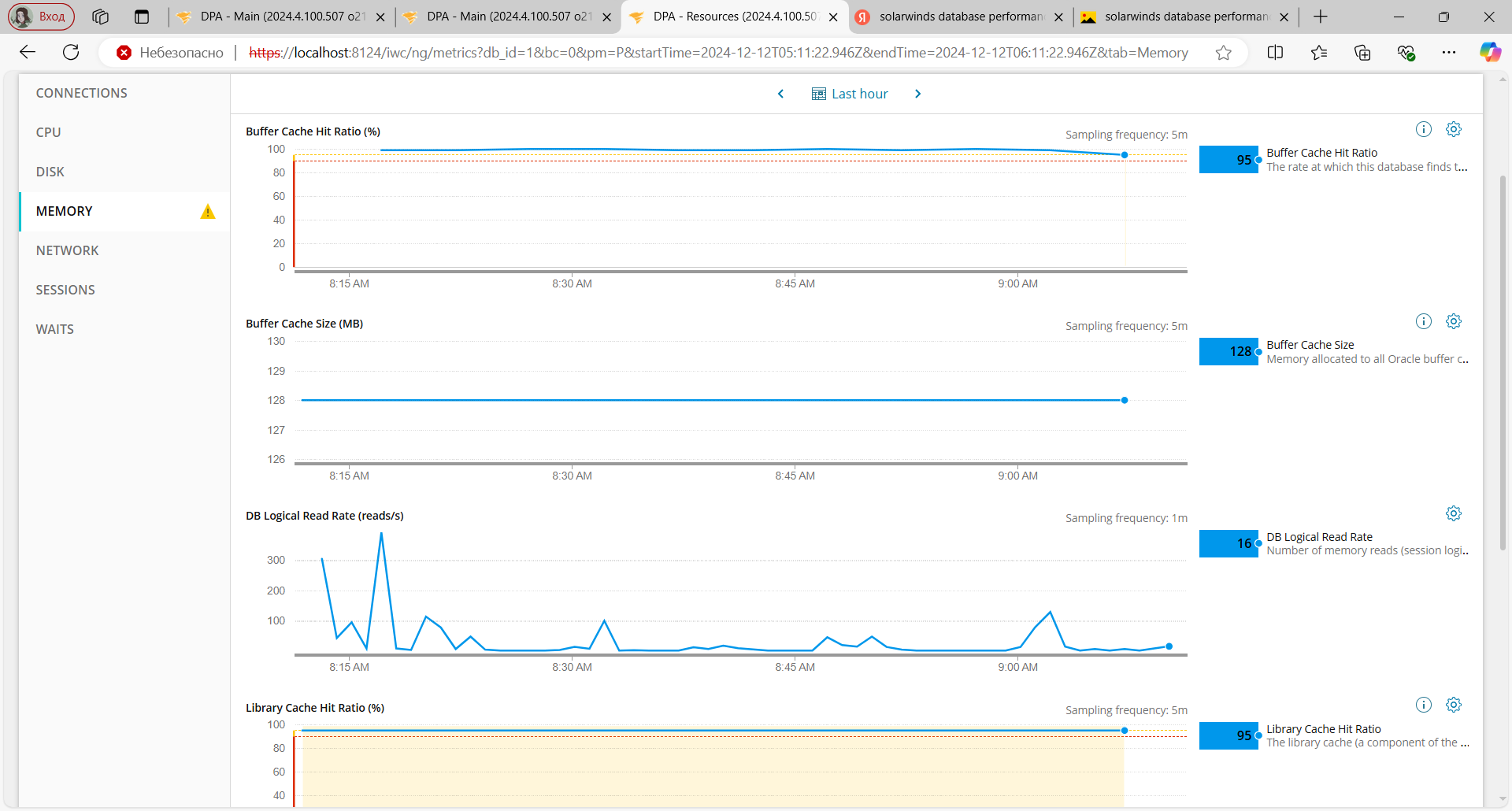


Рисунок 6.9 — Использование ресурсов за последний час

К тому же, SolarWinds DPA поддерживает все возможные серверы баз данных, что сводит на «нет» нужды в нескольких средствах для мониторинга базы данных. Эта система «все в одном» справится с любым типом сред: физической, виртуальной, облачной. Еще одной сильной стороной SolarWinds DPA оказалась графическая составляющая: имеющиеся графики очень информативны и интуитивно понятны.

**6.2 Мультимедийные типы данных**

Мультимедийные типы данных – это типы данных позволяющие хранить данные формата мультимедиа. В Oracle данная возможность реализована через Oracle Multimedia.

Oracle Multimedia (в прошлом Oracle interMedia) — это возможность Oracle Database 21c для управления мультимедийным контентом (изображения, звук, видео). Использование этой функции позволяет управлять мультимедийными данными интегрированно с реляционной информацией. Oracle Multimedia умеет работать с большинством популярных мультимедийных форматов. Хранить такие мультимедийные данные можно разными способами (внутри и вовне базы). Также предоставляется целый спектр интерфейсов для использования этих сервисов (объектный, реляционный). Для разработки приложений Oracle поставляет набор Java интерфейсов и компонентов.

Oracle Multimedia расширяет возможности Oracle Database в области надёжности, доступности и управления данными для мультимедийного контента в традиционных приложениях, медицинских приложениях, интернет-приложениях, приложениях для электронной коммерции и мультимедийных приложениях. Oracle Multimedia не управляет устройствами захвата или вывода мультимедиа; эта функция остаётся за прикладным программным обеспечением. На рисунке 6.10 представлена архитектура Oracle Multimedia.

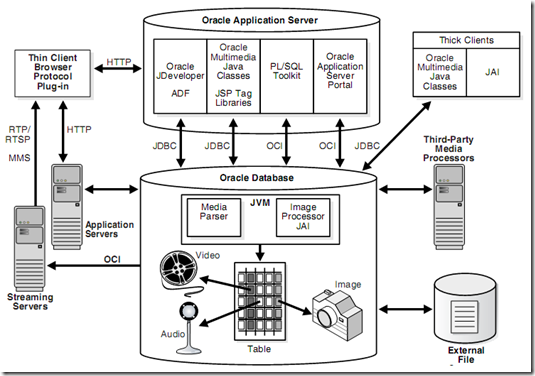


Рисунок 6.10 — Архитектура Oracle Multimedia

Использование процедур или методов Oracle Multimedia позволяет выполнять операции импорта и экспорта между БД и внешними системами хранения файлов.

Основные компоненты включают ORDImage для работы с изображениями, ORDAudio для аудиофайлов, а также поддержку DICOM для обработки медицинских изображений. Он также предоставляет функциональность анализа изображений и видео напрямую в базе данных.

Как и любые объектные, типы экземпляры ORDAudio и ORDImage состоят из атрибутов (включают метаданные и медиаданные) и методов. Медиаданные — это реальное видео, аудио или графическое изображение. Метаданные — это информация о данных: длина объекта, метод компрессии, формат. Методы — это процедуры, которые могут быть выполнены для объекта.

Oracle Multimedia интегрирует возможности пространственных данных с мультимедийными, что позволяет эффективно управлять пространственной и мультимедийной информацией в рамках единой базы данных. С его помощью можно выполнять SQL-операции с мультимедийными данными, что упрощает их интеграцию в приложения, использующие Oracle Database.

Для работы с данными большого объема СУБД Oracle предоставляет типы данных BLOB, CLOB, NCLOB и BFILE. Здесь LOB означает Large OBject, или большой объект. По сути, большой объект — это абстрактный тип для манипуляции данными большого объема внутри БД, а типы BLOB, CLOB, NCLOB и BFILE — его конкретные реализации.

В данном курсовом проекте технология мультимедийных типов данных отражена в необходимости хранить изображения, являющиеся обложками для тем подкастов, а также аудиофайлы подкастов, без которых приложение потеряло бы смысл. Речь идет о таблицах «TOPICS\_T» и «PODCASTS\_T». Для хранения в поле изображений и аудиофайлов в базе данных применяется тип BLOB. Тип BLOB 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 изображения или аудиофайла из массива битов на обратном пути.

Рассмотрим сначала процесс преобразования изображения в тип данных, понятный Oracle. Так как в хранимых процедурах нельзя передавать данные объемом более 32 Кб, процесс создания администратором новой темы происходит в два этапа. Сначала вызывается хранимая процедура AddTopic, после чего происходит добавление обложки к появившейся в базе данных строке.

Процесс выглядит следующим образом: по нажатию на кнопку для добавления обложки открывается стандартное диалоговое окно Windows, с помощью которого пользователь волен выбрать нужное изображение с расширением .jpg или .png. Последующие действия можно увидеть в листинге 6.1.

OpenFileDialog openFileDialog = new OpenFileDialog()

{

Filter = "Image Files|\*.jpg;\*.png;"

};

if (openFileDialog.ShowDialog() == true)

{

imageName = openFileDialog.FileName;

image = File.ReadAllBytes(openFileDialog.FileName);

var bitmImg = new BitmapImage();

using (var mem = new MemoryStream(image))

{

mem.Position = 0;

bitmImg.BeginInit();

bitmImg.CacheOption = BitmapCacheOption.OnLoad;

bitmImg.StreamSource = mem;

bitmImg.EndInit();

}

bitmImg.Freeze();

coverImage.Source = bitmImg;

}

openFileDialog = null;

Листинг 6.1 — Загрузка изображения и разбиение его на массив битов

Для удобства администратора загруженное изображение отображается в окне приложения, чтобы можно было убедиться в правильности выбора.

После превращения изображения в массив битов используя драйвер Managed Data Access мы отправляем UPDATE-запрос для вставки изображения в существующую строку, где с помощью подготовленного заявления передаем этот массив для вставки в поле topic\_blob. Это можно увидеть в листинге 6.2.

try

{

cmd.ExecuteNonQuery();

con.Close();

con.Open();

FileStream fls;

fls = new FileStream(imageName, FileMode.Open, FileAccess.Read);

byte[] blob = new byte[fls.Length];

fls.Read(blob, 0, System.Convert.ToInt32(fls.Length));

fls.Close();

if (imageName != "")

{

OracleCommand cmd2 = con.CreateCommand();

OracleTransaction txn;

txn = con.BeginTransaction(IsolationLevel.ReadCommitted);

cmd2.Transaction = txn;

cmd2.CommandText = "UPDATE SYS.topics\_t " +

"SET " +

"topic\_blob = :ImageFront " +

"WHERE UPPER(topic\_name) = UPPER('" +

topicName.Text.Trim() + "')";

cmd2.Parameters.Add(":ImageFront", OracleDbType.Blob);

cmd2.Parameters[":ImageFront"].Value = blob;

cmd2.ExecuteNonQuery();

txn.Commit();

con.Close();

con.Dispose();

TopicAdded?.Invoke();

MessageBox.Show("Тема успешно добавлена!");

this.Close();

Application.Current.Windows[0].Show();

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при добавлении темы: {ex.Message}");

}

con.Close();

Листинг 6.2 — Добавление изображения в поле topic\_blob

На этом этапе изображение загружается в базу данных. Приложение обращается к этому изображению для отображения плиток тем и подкастов с соответствующей обложкой.

Процесс получения значения поля topic\_blob изображен в листинге 6.3. Он заключается в простейшем SELECT-запросе к таблице и чтении результата в качестве массива битов. После этого с помощью объекта BitmapImage() мы превращаем полученный массив битов в битовую карту. То есть, непосредственно в изображение.

con.ConnectionString = connectionString;

InitializeComponent();

CheckRole();

this.topicId = id;

this.topic = topicName;

this.narrator = narratorName;

this.year = yearReleased;

blockTopicName.Text = topicName;

blockNarratorName.Text = narratorName;

blockYear.Text = yearReleased.ToString();

con.Open();

OracleCommand cmd = con.CreateCommand();

cmd.CommandText = "SELECT topic\_blob FROM SYS.narrator\_topic\_view WHERE topic\_id = " + id.ToString();

cmd.CommandType = CommandType.Text;

OracleDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

try

{

BitmapImage image = new BitmapImage();

image.BeginInit();

image.StreamSource = new MemoryStream(reader.GetValue(0) as

byte[]);

image.EndInit();

topicCover.Source = image;

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

}

Листинг 6.3 — Получение изображения из базы данных

Далее мы рассмотрим аналогичный процесс, но с аудио файлом. В рамках окна добавления нового подкаста администратор все необходимые заполняет поля для добавления. По причинам, описанным выше, процесс загрузки подкаста тоже разбит на два этапа. Как и в случае с изображением, аудиофайл загружается с помощью диалогового окна. Процесс загрузки аудиофайла в базу данных идентичен случаю с изображениями, поэтому перейдем сразу к выгрузке аудио файла из базы.

Пользователь с любой ролью может прослушивать подкасты в приложении. Это происходит в рамках окна плеера. Логика прослушивания подкаста такова: при загрузке указанного окна отправляется SELECT-запрос к базе данным, в результате которого мы получаем массив битов аудиофайла с переданным идентификатором. Чтобы было возможно запустить аудио, с помощью объекта FileStream открывается поток записи на диск, в процессе которого полученный массив битов записывается в файл «current.mp3». Относительно корневого директория с решением, этот файл находится в папке «PodSproutFM\PodSproutFM\bin\Debug». Далее, чтобы запустить полученный из базы данных подкаст, мы инициализируем объект типа MediaPlayer, передав ему путь к скачанному файлу, и с этого момента можем управлять нашим аудиофайлом с помощью кнопок в окне плеера. Фрагмент кода представлен в листинге 6.4.

while (reader.Read())

{

try

{

audioByteArr = reader.GetValue(0) as byte[];

using (FileStream bytesToAudio = File.Create("current.mp3"))

{

bytesToAudio.Write(audioByteArr, 0, audioByteArr.Length);

Stream audioFile = bytesToAudio;

bytesToAudio.Close();

}

mediaPlayerObj.Open(new Uri(

@"D:\BSTU\Projects\БД\PodSproutFM\PodSproutFM\bin\Debug\current.mp3"

));

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

}

Листинг 6.4 — Запись аудио в файл-буфер

Реализованный функционал предоставляет администратору возможность сохранения и извлечения мультимедийного контента, обеспечивая полноценное взаимодействие с базой данных в контексте хранения и представления разнообразной медийной информации.

**6.3 Вывод по разделу**

Таким образом, в рамках курсового проекта был проведен мониторинг базы данных при помощи средства диагностики SolarWinds DPA, а также была успешно реализована функциональность хранения мультимедийных типов данных, таких как изображения и аудио.

7 Краткое описание приложения для демонстрации

Приложение «PodSproutFM» — это десктопное приложение, которое позволяет загружать подкасты, управлять ими и прослушивать их, а также дает всем пользователям возможность добавлять в избранное понравившиеся подкасты. Приложение поддерживает две роли — обычный пользователь и администратор.

В этом разделе мы рассмотрим цели и задачи, для которых разрабатывалось данное приложение. Подробное руководство по использованию, включая наглядные демонстрации, будет представлено в следующем разделе.

«PodSproutFM» — это инновационное приложение, созданное для удобной загрузки и прослушивания подкастов. Оно предлагает пользователям легкий доступ к подкастам и помогает организовать прослушивание в удобном формате. «PodSproutFM» — это незаменимый помощник в открытии новых идей и тем.

Особенности приложения:

1. Удобный интерфейс для загрузки подкастов:

«PodSproutFM» предоставляет интуитивно понятный интерфейс для поиска и загрузки подкастов. Вы можете легко находить новые выпуски, добавлять их в избранное и загружать для офлайн-прослушивания.

2. Сохранение подкастов в избранное:

Поддержка добавления понравившихся подкастов в избранное позволяет вам легко найти подкаст, который вы бы хотели прослушать снова. Легко добавляйте или удаляйте выпуски, чтобы всегда иметь под рукой любимые эпизоды.

3. Офлайн-прослушивание:

Прослушивание подкастов в приложении «PodSproutFM» не требует доступа к Интернету. Это удобно и экономит трафик.

4. Безопасность и надежность:

«PodSproutFM» обеспечивает высокий уровень безопасности данных. Такие личные данные, как пароль, шифруются и в дальнейшем работа происходит с зашифрованным паролем.

5. Простота использования:

Приложение «PodSproutFM» было разработано с акцентом на удобство. Интуитивный интерфейс позволяет быстро освоиться даже новым пользователям. Благодаря понятной структуре приложения, вы сможете начать слушать подкасты сразу же без необходимости в обучении.

«PodSproutFM» — это идеальное решение для любителей подкастов. Оно позволяет удобно организовывать прослушивание и открывать для себя новые интересные темы. Забудьте о сложностях в поиске интересующего контента — «PodSproutFM» поможет вам открыть для себя мир подкастов.

8 Руководство пользователя

Так как база данных, поддерживает 2 типа пользователей, то есть и 2 сценария использования. Первый — при входе в качестве обычного пользователя, второй — при входе в качестве администратора.

**8.1 Сценарий обычного пользователя**

При запуске приложения открывается окно авторизации (рисунок 8.1), где зарегистрированному пользователю следует ввести свой логин и пароль.

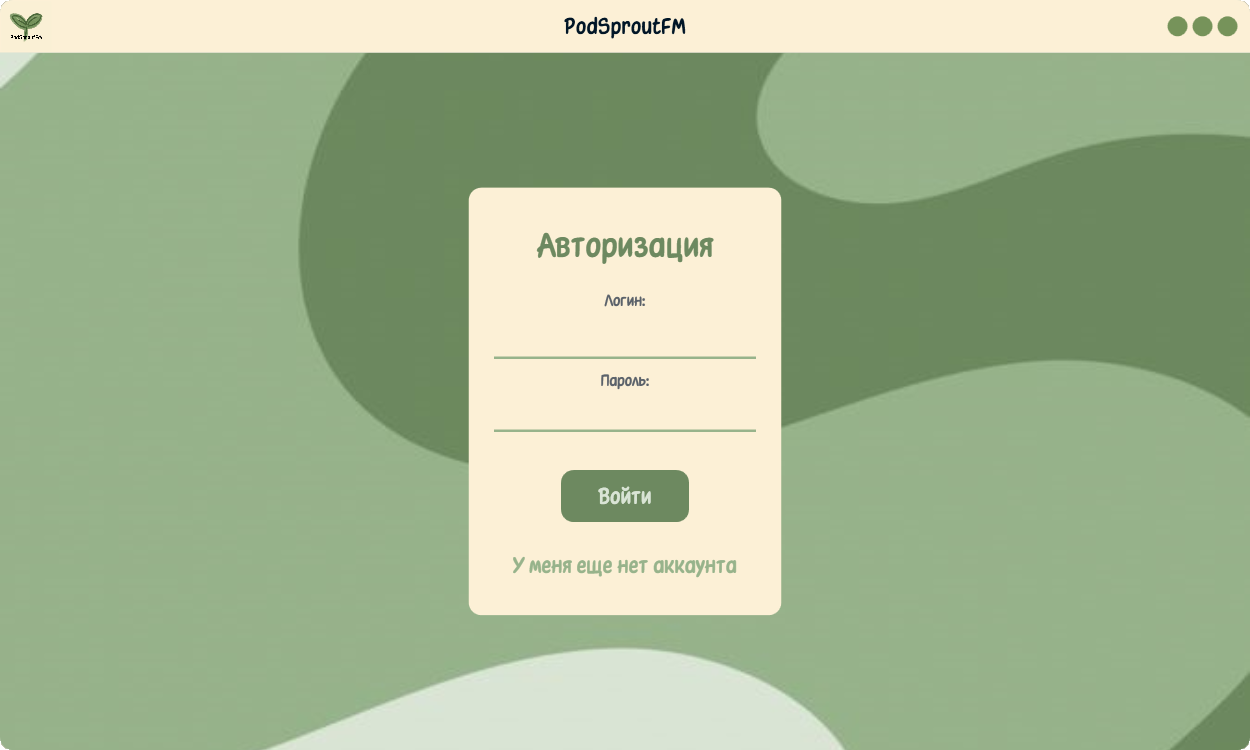


Рисунок 8.1 — Окно авторизации

При вводе неправильного логина или пароля пользователь увидит сообщение об ошибке авторизации (рисунок 8.2).

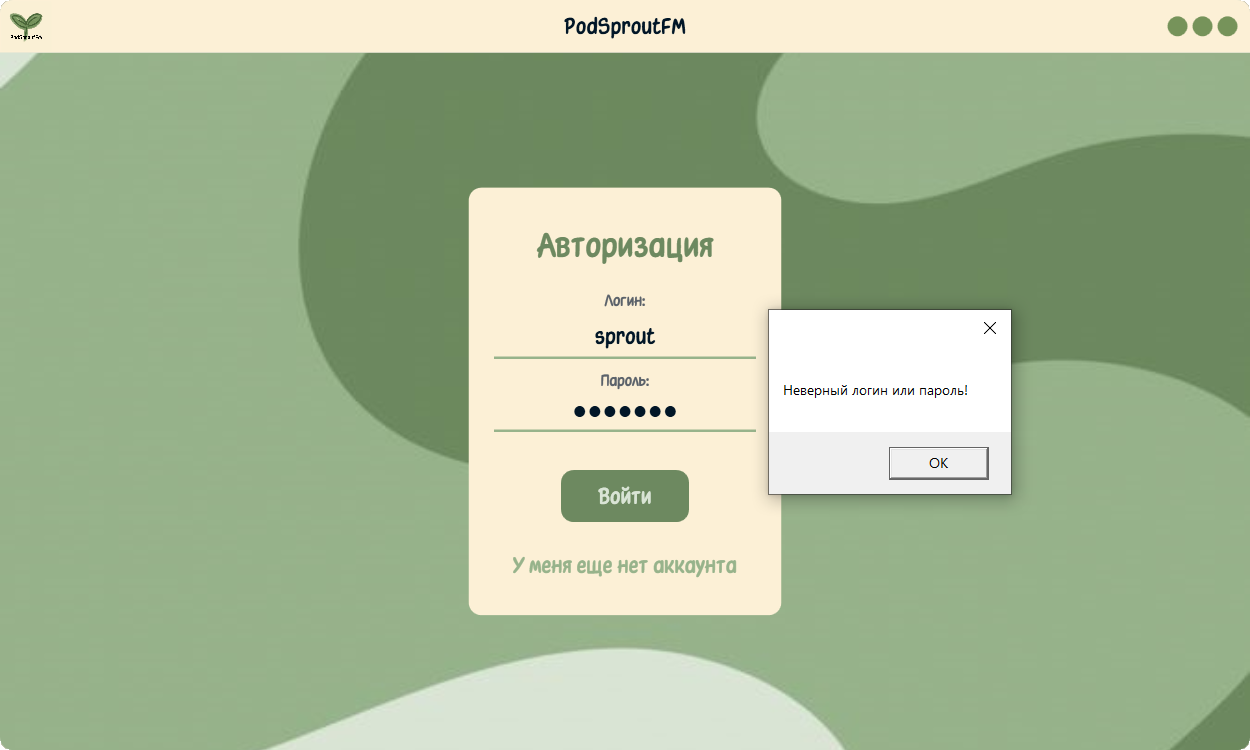


Рисунок 8.2 — Ошибка авторизации (неверный логин или пароль)

При успешной авторизации пользователь будет перенаправлен на главную страницу приложения.

Пользователь не получить доступ к приложению, пока не зарегистрируется. Для этого следует нажать на кнопку «У меня еще нет аккаунта» в окне авторизации, после чего произойдет перенаправление на окно регистрации. Окно регистрации изображено на рисунке 8.3.

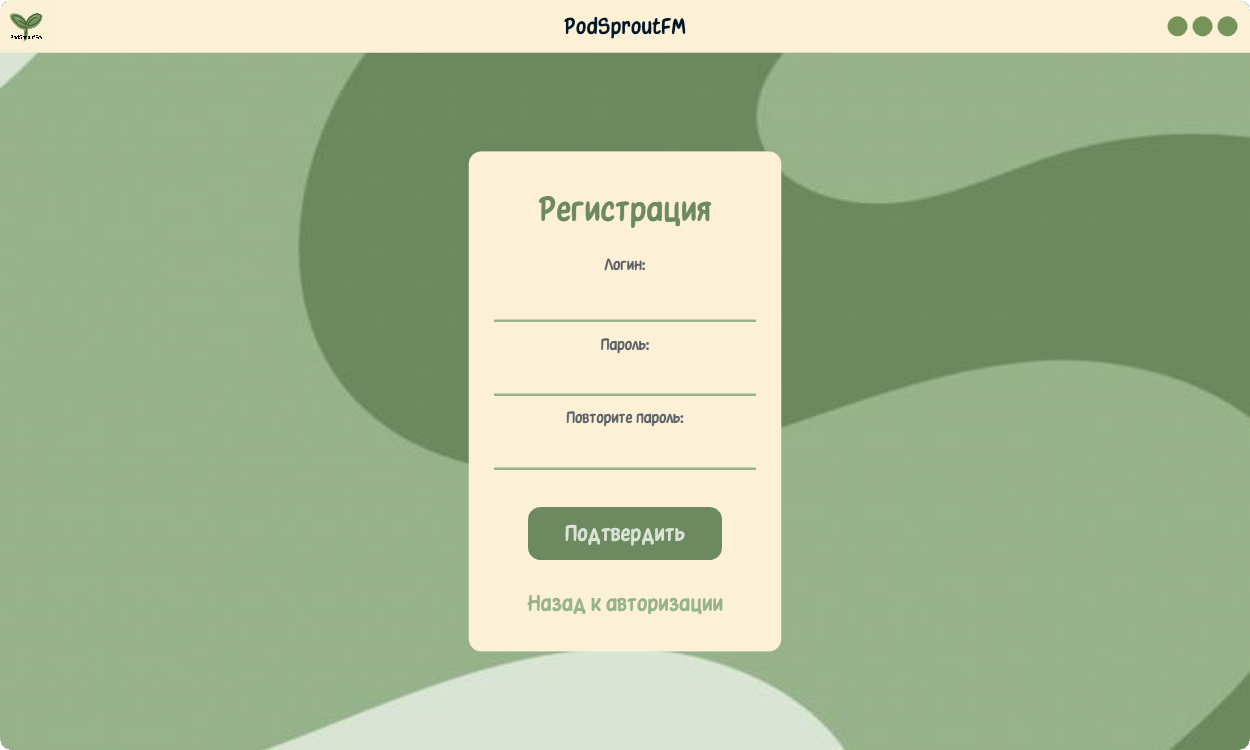


Рисунок 8.3 — Окно регистрации

Если в этом окне ввести логин пользователя, уже существующего в базе данных, пользователь увидит следующую ошибку (рисунок 8.4).

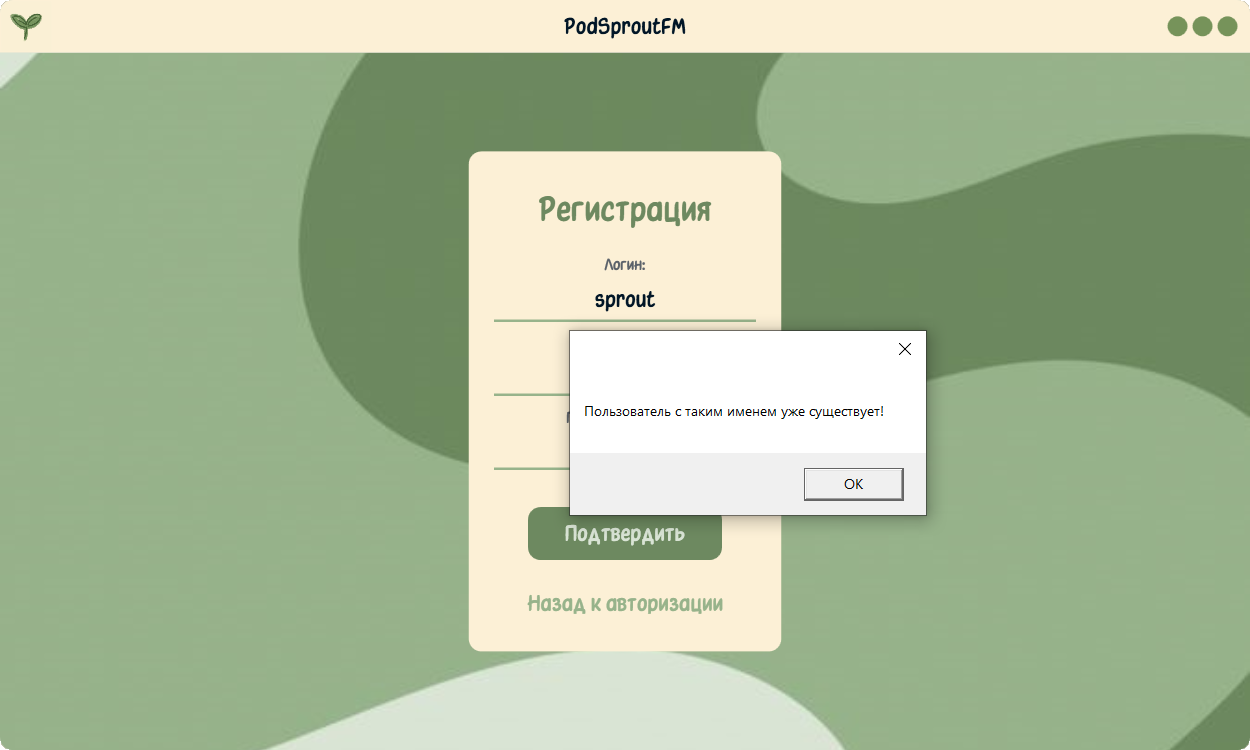


Рисунок 8.4 — Ошибка регистрации (пользователь уже существует)

Если ввести несовпадающие пароли в поля «Пароль» и «Повторите пароль», появится соответствующее сообщение об ошибке, показанное на рисунке 8.5. Пользователю будет предложено попробовать ввести пароли снова перед повторной попыткой завершения регистрации.

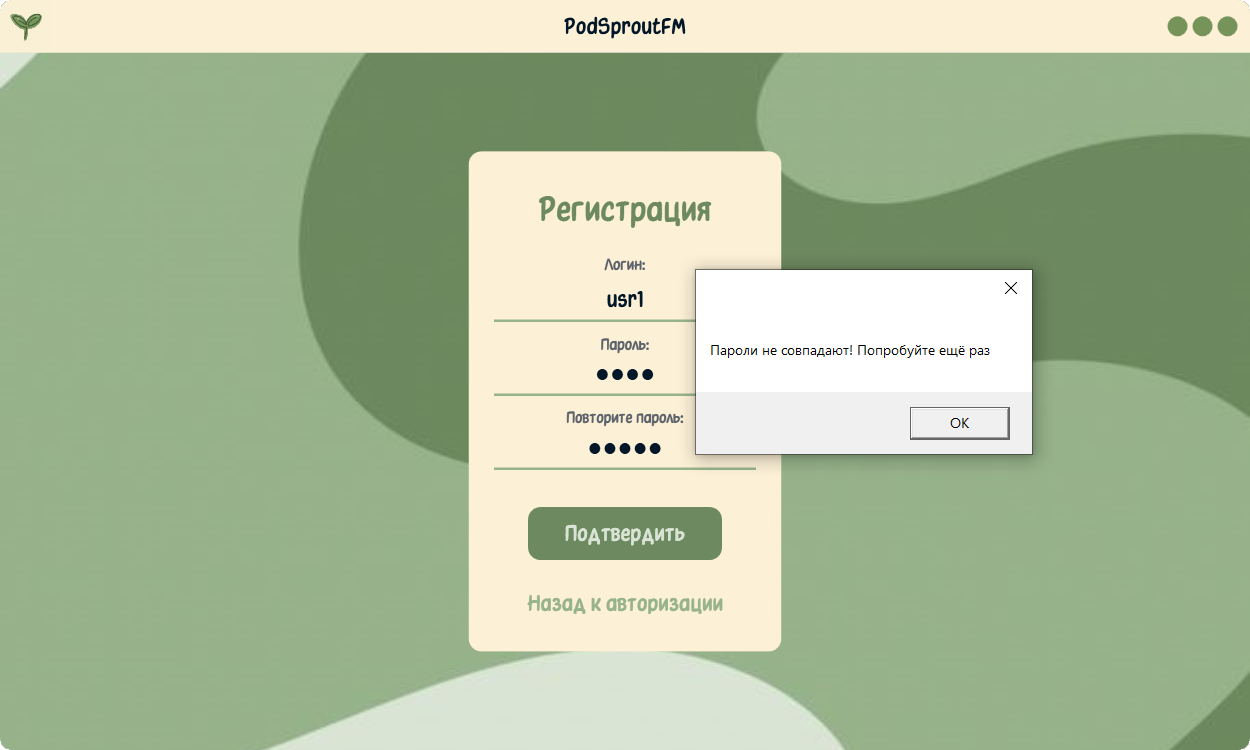


Рисунок 8.5 — Ошибки регистрации (несовпадение паролей)

Если заполнить не все поля формы регистрации, появится соответствующее сообщение об ошибке, изображенное на рисунке 8.6.

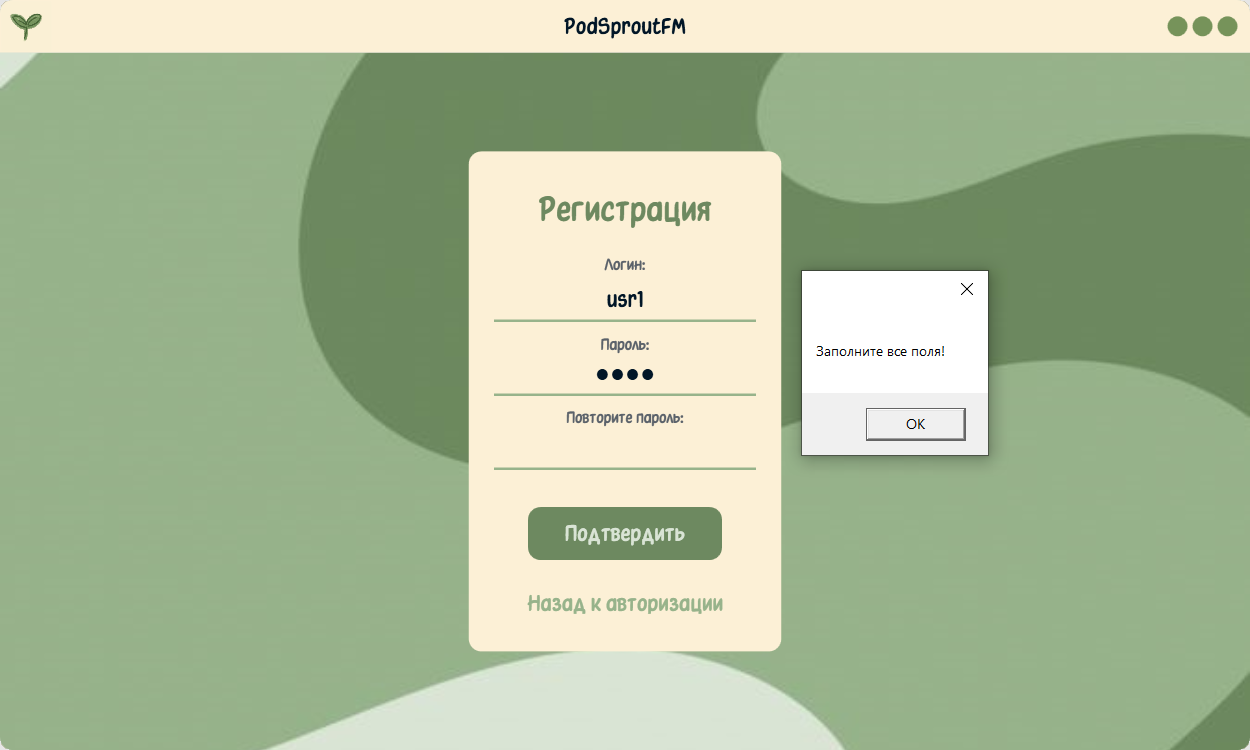


Рисунок 8.6 — Ошибка регистрации (не все поля заполнены)

После успешной регистрации пользователь будет перенаправлен обратно на страницу авторизации, после чего ему нужно ввести свои учетные данные, чтобы обеспечить доступ к контенту. Допустим, пользователь успешно зарегистрировался и авторизовался в приложении «PodSproutFM». После входа в систему пользователь будет перенаправлен на домашнее окно, интерфейс которого представлен на рисунке 8.7.



Рисунок 8.7 — Домашнее окно (вкладка «Дикторы»)

Смысл аккаунта рядового пользователя сводится к поиску подкастов, их прослушиванию и добавлению в свой плейлист. По умолчанию в окне открывается вкладка «Дикторы». Если обратить внимание на меню, в нем также имеются вкладки «Темы», «Подкасты», «Избранное» и «Настройки».

Вкладка «Дикторы» содержит список рассказчиков, состоящий из плиток с кнопками. На этой вкладке можно осуществлять поиск по имени диктора, воспользовавшись текстовый полем внизу. Любой пользователь может нажать на кнопку «Подкасты» напротив имени диктора, после чего откроется окно со списком всех имеющихся подкастов этого диктора. Это окно изображено на рисунке 8.8.



Рисунок 8.8 — Список подкастов диктора

Плитки подкастов универсальны и имеют одинаковый функционал во всех точках приложения.

Вкладка «Темы» содержит список тем подкастов, так же состоящий из плиток с кнопками. На этой вкладке можно осуществлять поиск подкастов по названию темы, воспользовавшись текстовый полем внизу. Интерфейс вкладки «Темы» изображен на рисунке 8.9.

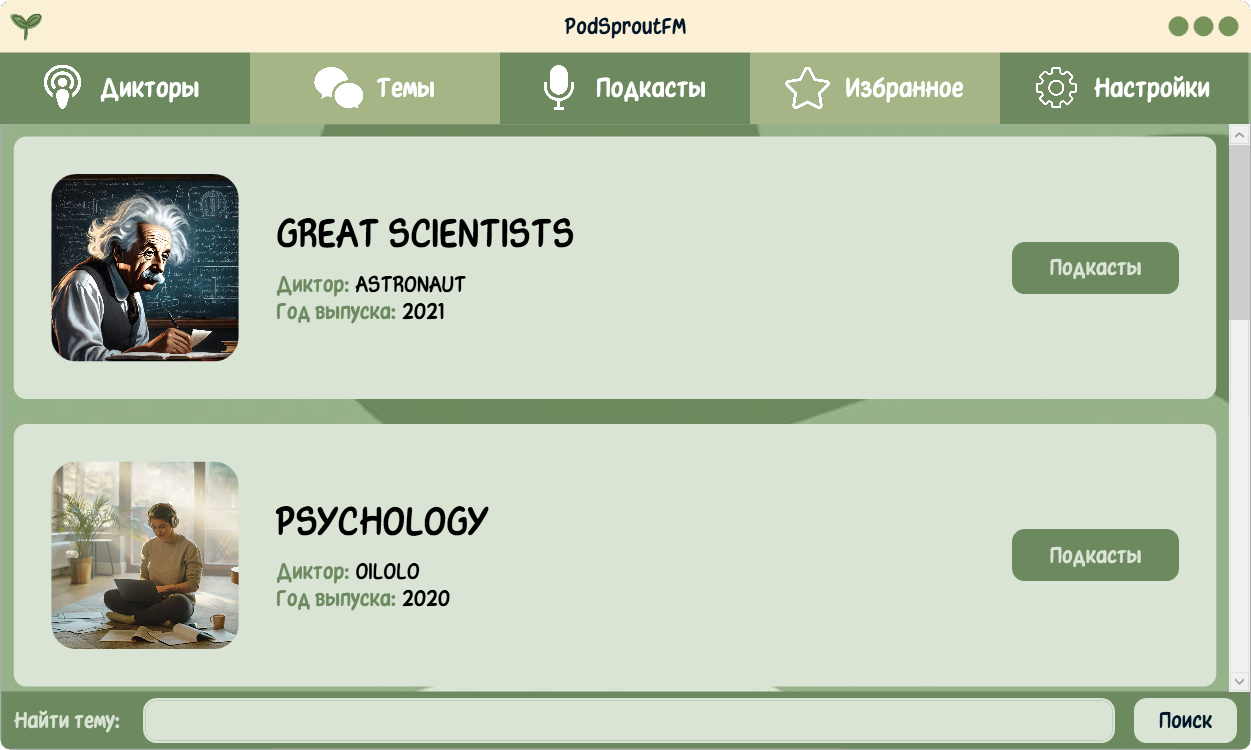


Рисунок 8.9 — Вкладка «Темы»

Любой пользователь может нажать на кнопку «Подкасты» напротив названия темы, и тогда попадет на окно со списком всех подкастов темы. Это окно можно увидеть на рисунке 8.10.

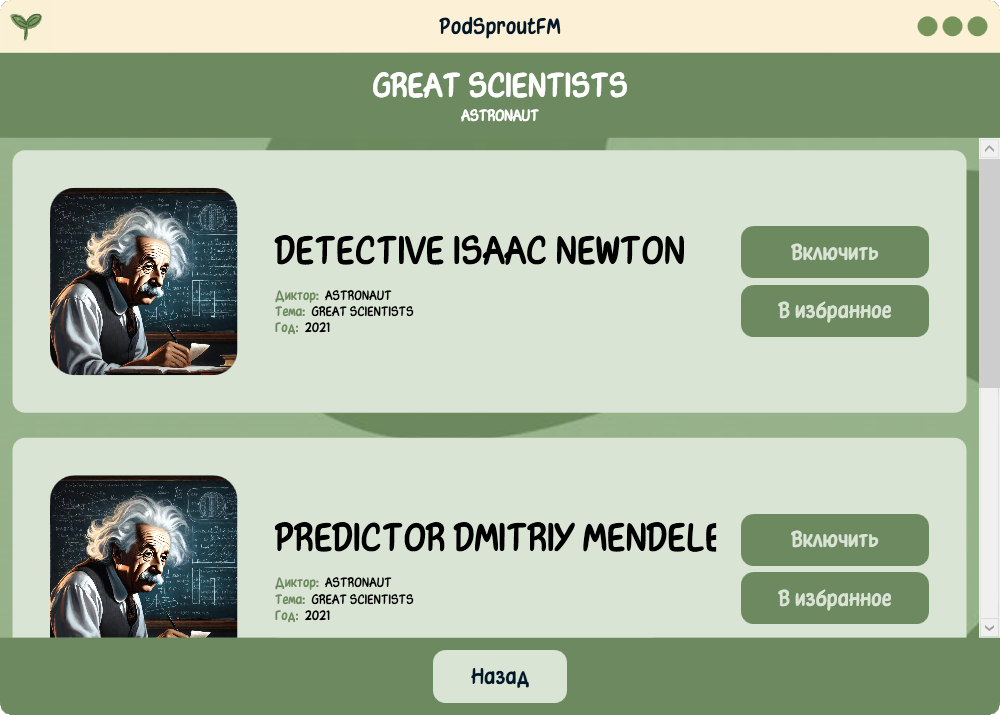


Рисунок 8.10 — Список подкастов темы

Вкладка «Подкасты» содержит всех подкастов из базы данных, состоящий из плиток с кнопками. На этой вкладке можно осуществлять поиск по названию подкаста, воспользовавшись текстовый полем внизу. Данная вкладка представлена на рисунке 8.11.



Рисунок 8.11 — Вкладка «Подкасты»

Любой пользователь может нажать на кнопку «Включить» внутри плитки подкаста, после чего откроется окно с плеером. Плеер содержит информацию о текущем подкасте и три кнопки для управления: воспроизведение, пауза и сброс. Это окно можно увидеть на рисунке 8.12.



Рисунок 8.12 — Плеер

По нажатию на кнопку «В избранное» на плитке подкаста в любом месте приложения этот подкаст добавится в плейлист данного пользователя (вкладка «Избранное»). Если данный подкаст уже есть в плейлисте пользователя, он увидит ошибку, показанную на рисунке 8.13.



Рисунок 8.13 — Ошибка добавления подкаста в избранное (подкаст уже сохранен)

На вкладке «Избранное» плитки подкастов несколько отличаются: на них вместо кнопки «В избранное» на них находится кнопка «Убрать из избранного». Нажав на эту кнопку, пользователь может удалить подкаст из своего плейлиста. Содержимое вкладки «Избранное» можно увидеть на рисунке 8.14.



Рисунок 8.14 — Вкладка «Избранное»

Перейдя на вкладку «Настройки», пользователь может выйти из аккаунта или удалить его, изменить свой логин или свой пароль. Если введенный новый логин совпадает с уже имеющимся в базе, пользователь увидит ошибку, изображенную на рисунке 8.15.



Рисунок 8.15 — Ошибка изменения логина (логин уже занят)

Любой рядовой пользователь может удалить свой аккаунт при желании. При этом надо иметь в виду, что плейлист, составленный пользователем, будет утерян.

**8.2 Сценарий администратора**

Роль администратора дополняет функционал обычного пользователя, поэтому будет уместно в рамках данного руководства осветить возможности аккаунта администратора.

Аккаунт администратора находится в базе данных приложения изначально, и зарегистрировать его нельзя. Администратор входит в свой аккаунт, после чего перенаправляется на главное окно, на вкладку «Дикторы».

Администратор может делать то же, что и обычный пользователь (кроме удаления своего аккаунта), но с расширенным функционалом: администратор может добавлять, изменять и удалять дикторов, темы и подкасты, а также имеет доступ к специальному окну администратора (через вкладку «Настройки») со списком пользователей и кнопками для запуска импорта и экспорта XML, а также для добавления 100 000 пользователей.

Плитки дикторов, тем и подкастов здесь имеют две дополнительные кнопки: «Изменить» и «Удалить». Внешний вид плитки подкаста при входе под аккаунтом администратора изображен на рисунке 8.16.



Рисунок 8.16 — Плитка подкаста (администратор)

На вкладке «Настройки» администратору также доступна кнопка «Открыть окно администратора». По нажатию на эту кнопку откроется соответствующее окно. Оно имеет две вкладки: «Пользователи» и «Другое». На вкладке с пользователями находятся плитки со списком пользователей. Поскольку в базе данных зарегистрировано 100 000 пользователей, на этой вкладке пользователи отображаются группами по 20. Чтобы увидеть следующих 20 пользователей, нужно нажать на кнопку «Показать следующих 20». Также на этой странице доступен поиск по логину пользователя. Интерфейс этой вкладки можно рассмотреть на рисунке 8.17.



Рисунок 8.17 — Вкладка «Пользователи»

Вкладка «Другое» содержит 5 кнопок. При нажатии на кнопку «Экспорт списка пользователей в XML» запустится процедура UsersExport, экспортирующая полный список пользователей в файл «/opt/oracle/cw/UserExport.xml». При нажатии на кнопку «Экспорт списка подкастов в XML» запустится процедура PodcastsExport, экспортирующая список подкастов в файл «/opt/oracle/cw/PodcastExport.xml».

После выполнения одной из процедур экспорта третья кнопка «Импорт списка дикторов из XML» станет доступной для нажатия. По нажатию на эту кнопку запустится процедура NarratorsImport, импортирующая в базу данных дикторов из файла «/opt/oracle/cw/NarratorImport.xml».

Четвертая кнопка «Добавить 100 000 пользователей» запускает процедуру регистрации 100 000 пользователей (Insert100KUsers). После этого можно перейти обратно на вкладку «Пользователи» и увидеть полный список добавленных пользователей.

Последняя кнопка «Удалить 100 000 пользователей» позволяет удалить из базы данных 100 000 пользователей, добавленных после нажатия на предыдущую кнопку. После нажатия на эту кнопку можно вернуться на вкладку «Пользователи» и убедиться, что все эти пользователи действительно были удалены из базы приложения.

Внешний вид вкладки «Другое» изображен на рисунке 8.18.

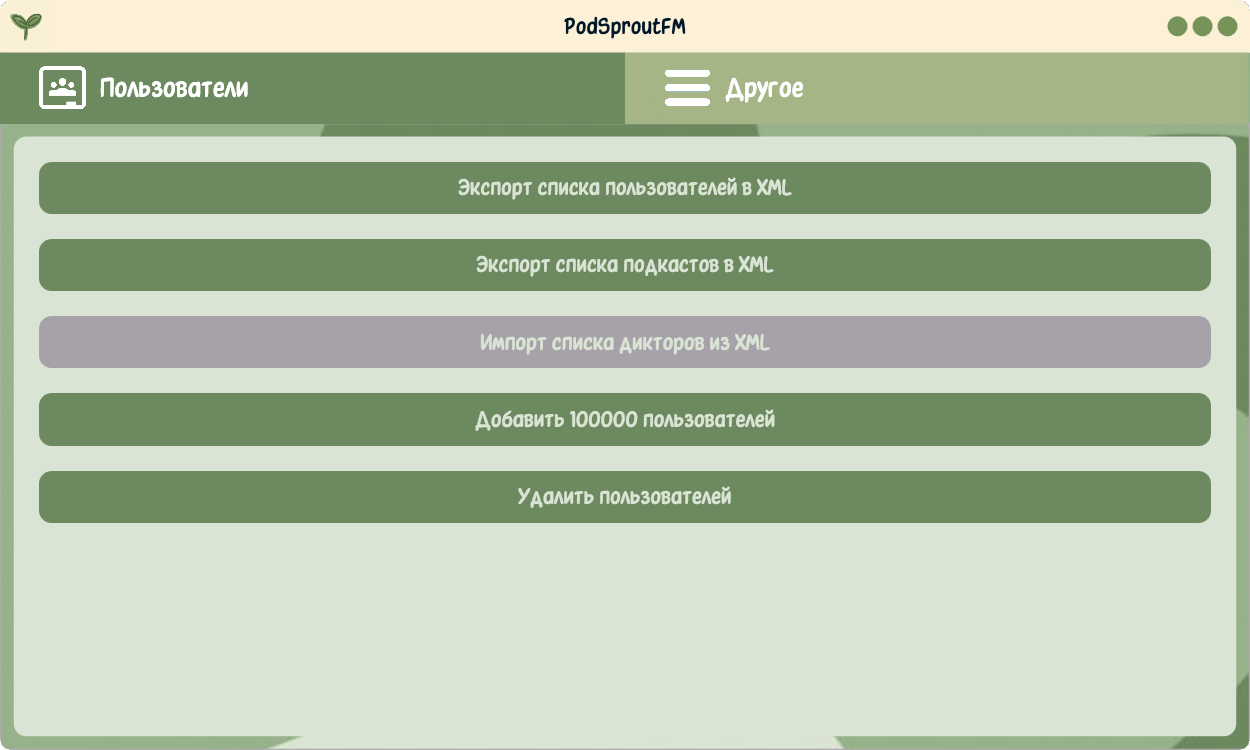


Рисунок 8.18 — Вкладка «Другое»

Будучи администратором, можно добавлять дикторов по нажатию на кнопку «Добавить диктора» на вкладке «Дикторы». Если попытаться добавить диктора с именем, которое уже есть в базе, получим ошибку, которую видно на рисунке 8.19.

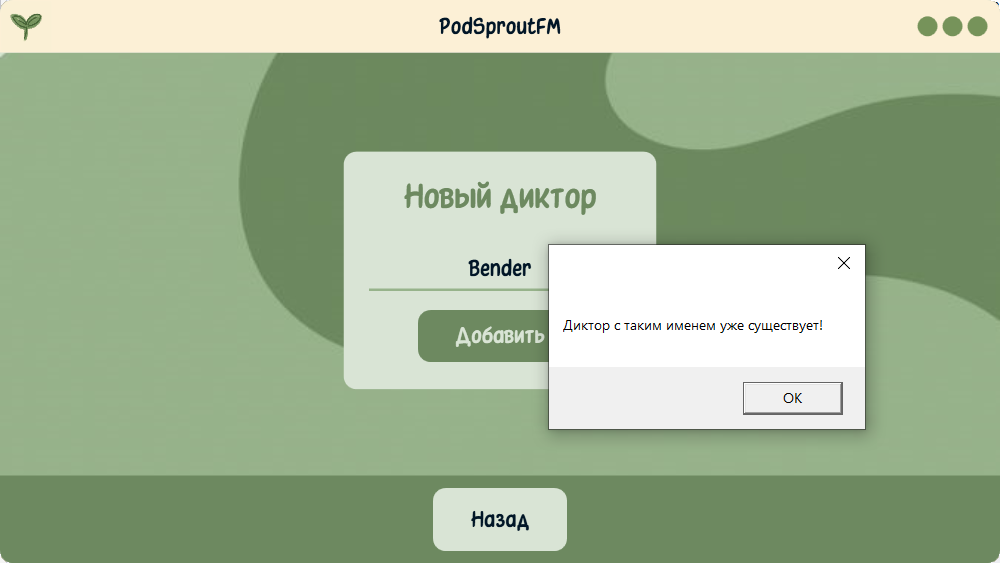


Рисунок 8.19 — Ошибка добавления диктора (диктор уже существует)

По нажатию на кнопку «Изменить» на плитке диктора можно изменить имя данного диктора. Если попытаться изменить имя на уже существующее, то получим ошибку, аналогичную изображенной на рисунке 8.19.

По нажатию на кнопку «Удалить» на плитке диктора можно удалить этого диктора. Стоит учитывать, что это также удалит все связанные с ним темы, подкасты и копии подкастов в плейлистах пользователей.

Нажав на кнопку «Добавить тему» на вкладке «Темы», можно добавить новую тему для существующего диктора. Если нажать «Добавить» в появившемся окне, не заполнив все поля, получим ошибку с рисунка 8.20. Аналогичная ошибка обрабатывается во всех остальных окнах. Также, если попробовать добавить уже существующую тему, мы также получим ошибку.

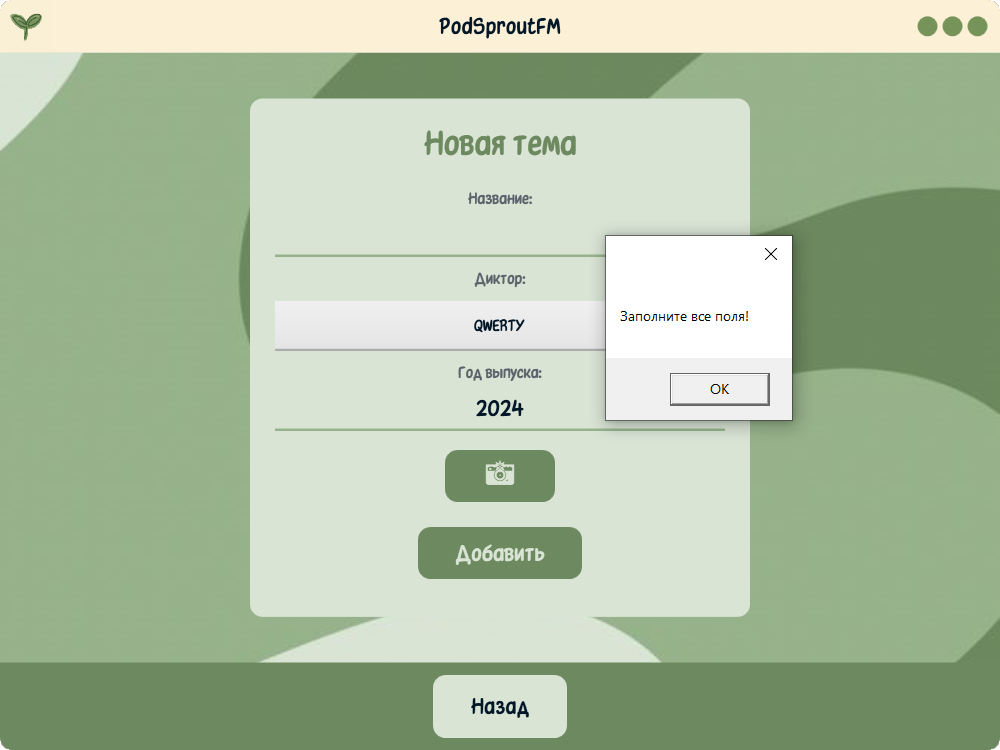


Рисунок 8.20 — Ошибка добавления темы (не все поля заполнены)

По нажатию на кнопку «Изменить» на плитке темы можно изменить название данной темы и год ее выпуска. Если попытаться изменить имя темы на уже существующее, то получим ошибку, аналогичную изображенной на рисунке 8.19.

По нажатию на кнопку «Удалить» на плитке темы можно удалить данную тему. Стоит учитывать, что это также удалит все связанные с ней подкасты и копии подкастов в плейлистах пользователей.

Нажав на кнопку «Добавить подкаст» на вкладке «Подкасты», можно добавить новый подкаст для существующей темы. Если попробовать прикрепить в качестве аудиофайла файл с расширением, отличным от \*.mp3, появится окно с ошибкой, изображенной на рисунке 8.21.

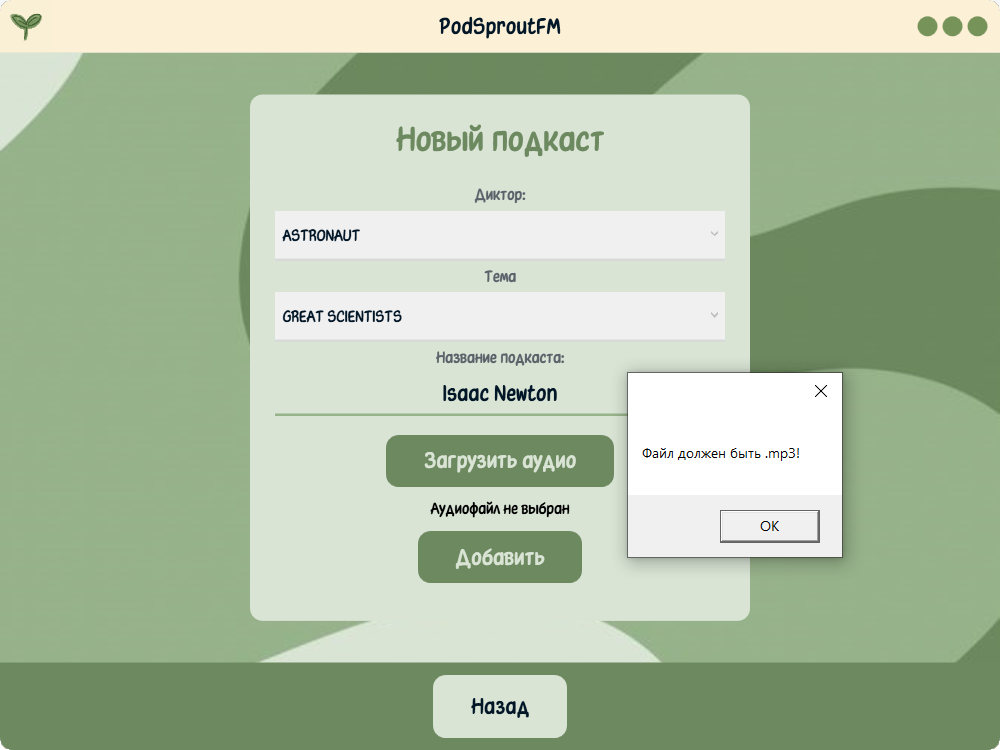


Рисунок 8.21 — Ошибка добавления подкаста (неверный формат аудиофайла)

По нажатию на кнопку «Изменить» на плитке подкаста можно изменить название данного подкаста. Если попытаться изменить имя подкаста на уже существующее, то получим ошибку, аналогичную изображенной на рисунке 8.19.

По нажатию на кнопку «Удалить» на плитке подкаста можно удалить данный подкаст. Стоит учитывать, что это также удалит все его копии в плейлистах пользователей.

**8.3 Вывод по разделу**

Таким образом, в проекте был протестирован весь базовый функционал рядового пользователя и администратора приложения. Все компоненты работают исправно. Приложение получилось простым и понятным в использовании.

# ЗАКЛЮЧЕНИЕ

Результатом выполнения курсового проекта стала база данных для работы связанного с ней приложения для прослушивания подкастов. База данных была спроектирована на базе СУБД Oracle 21c. Приложение было разработано с использованием технологии С# WPF для операционной системы Windows. В ходе курсовой работы была разработана база данных в Oracle 12c на тему «Музыкальная площадка». Приложение является средством взаимодействия клиента с базой данных.

Во время выполнения было изучено немало теоретического материала, а также просмотрено и разобрано большое количество уже готовых решений тех или иных задач. Были разработаны, описаны и применены на практике процедуры экспорта и импорта данных; подробно изучено и рассмотрено средство диагностики SolarWinds Database Performance Analyzer; дополнительно изучена технология хранения мультимедийных данных.

Функционально были выполнены следующие задачи:

* регистрация и авторизация пользователей приложения с учётом выданных им ролей;
* пароли при регистрации шифруются;
* мультимедийные типы данных, корректно записывающиеся в базу данных и затем воспроизводящиеся в приложении;
* в процессе разработки производительность была проверена средством диагностики SolarWinds DPA;
* поиск подкастов по диктору, теме и названию самого подкаста;
* возможность каждому пользователю сохранять понравившиеся подкасты в избранное;
* добавление дикторов, тем и подкастов администратором;
* администратор имеет доступ к информации о пользователях (включая их пароли);
* все пользователи могут изменять свой логин и пароль;
* все пользователи кроме администратора могут удалить свой аккаунт по желанию;
* импорт и экспорт данных таблиц в XML формат;
  + - заполнение таблицы с пользователями на 100 000 записей;
    - реализация технологии шифрования на базе DBMS\_CRYPTO, а также технологии хранения мультимедийных данных на базе DBMS\_LOB.

Проект не может окончательно претендовать на полноценный коммерческий продукт, поскольку для демонстрации технологии и работы с базой данных были опущены некоторые атрибуты современных площадок для подкастинга, реализация которых не относится к основной теме курсового проекта (например, программирование искусственного интеллекта, собирающего информацию о предпочтениях пользователя и формирующее на их основе подборки).

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

Список используемых источников

1. Официальный сайт Oracle [Электронный ресурс] / Режим доступа – URL:

https://www.oracle.com/ — Дата доступа: 04.10.2024.

1. Официальная документация Oracle [Электронный ресурс] / Режим доступа – URL: https://docs.oracle.com/en/ — Дата доступа: 05.10.2024.
2. Сайт о программировании metanit.com [Электронный ресурс] / Режим доступа: https://metanit.com/ — Дата доступа: 10.10.2024
3. Система вопросов и ответов stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com/ — Дата доступа: 15.10.2024
4. oracle-dba.ru [Электронный ресурс] / Режим доступа: https://oracle-dba.ru/ — Дата доступа: 26.10.2024
5. DBMS\_XMLDOM Package [Электронный ресурс] / Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/21/arpls/ — Дата доступа: 11.11.2024
6. Ручное шифрование данных [Электронный ресурс] / Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/21/dbseg/ — Дата доступа: 16.11.2024

# ПРИЛОЖЕНИЕ A

Диаграмма вариантов использования (обычный пользователь)

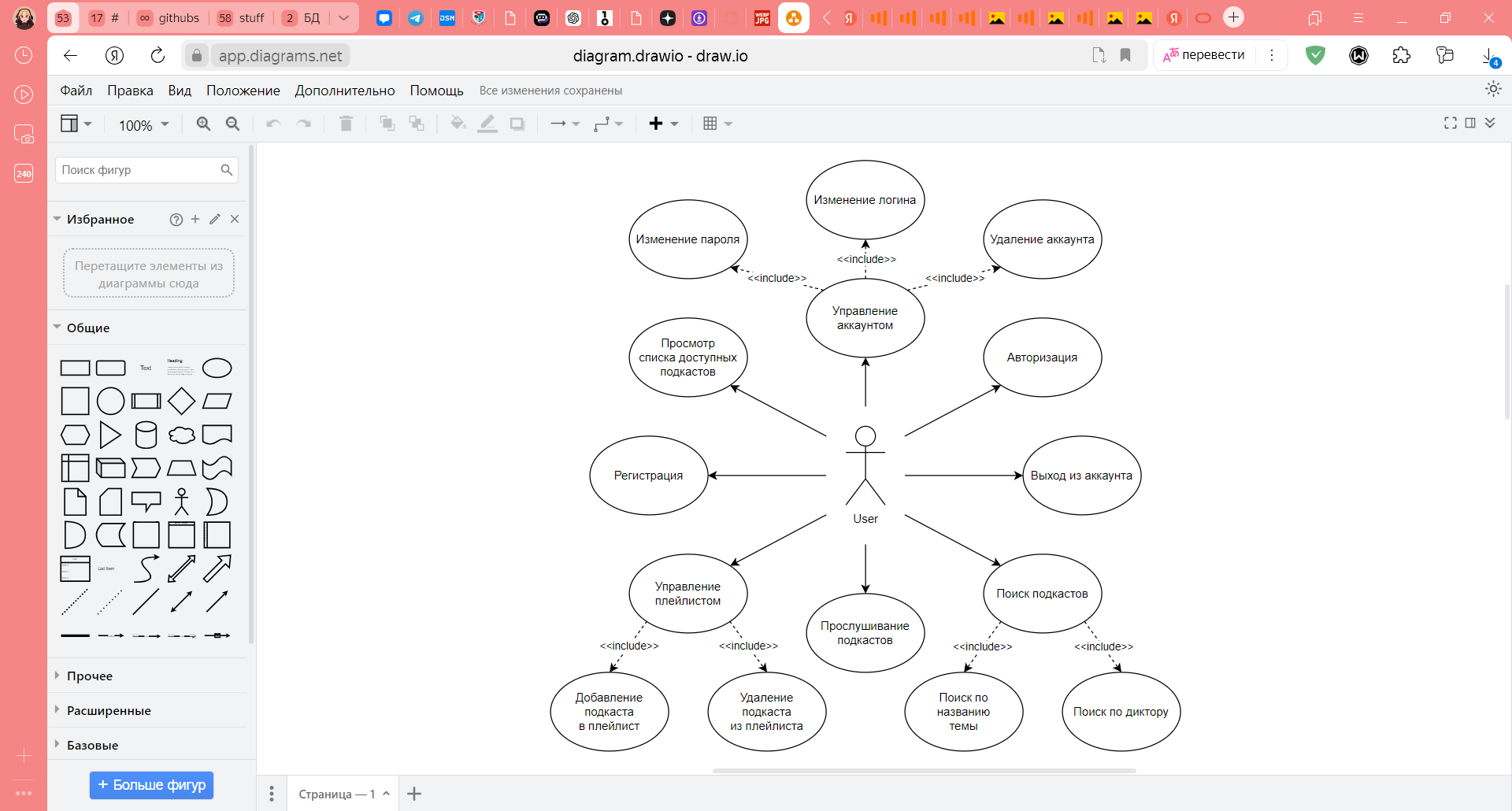
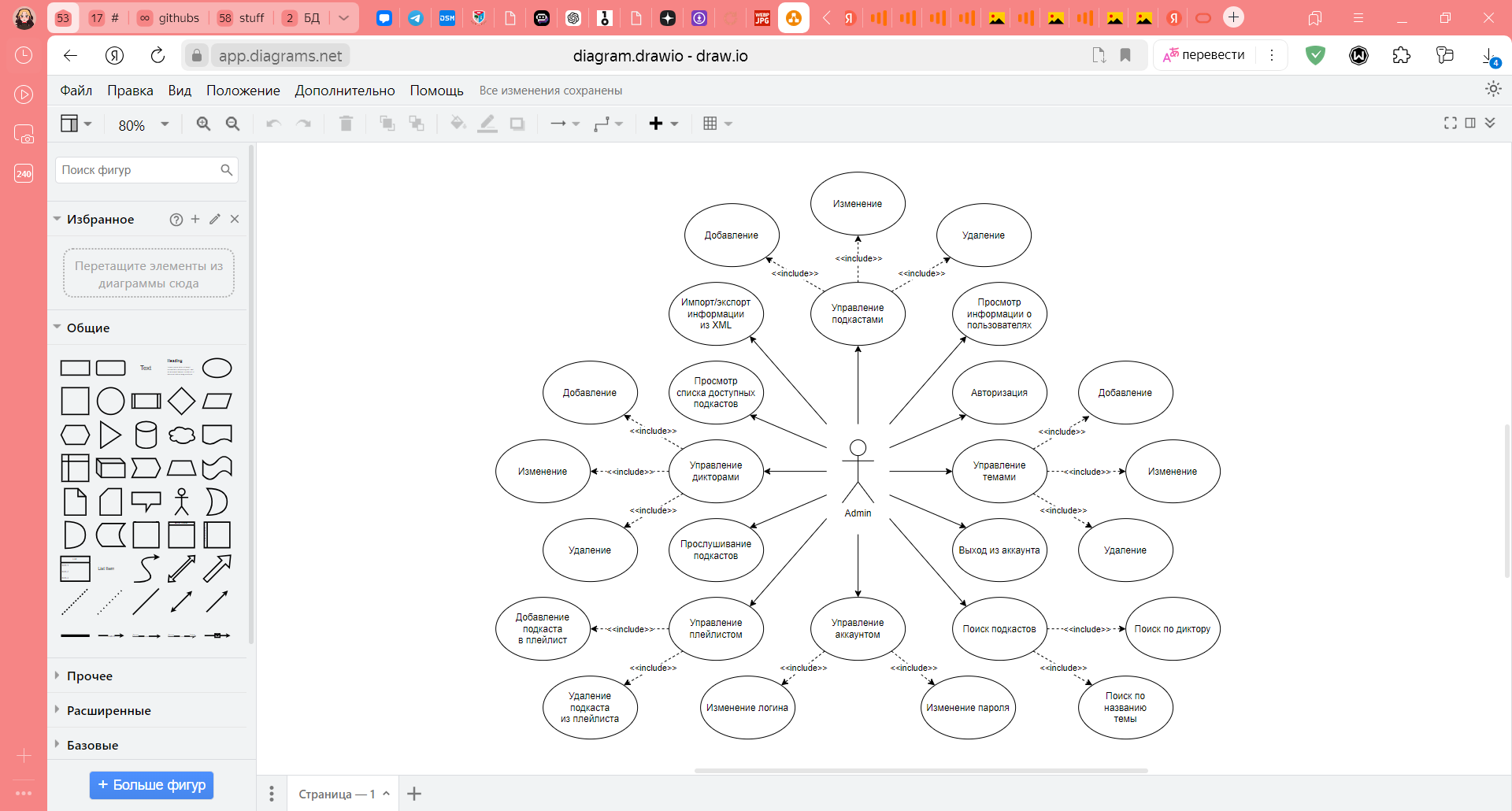


Диаграмма вариантов использования (администратор)



# ПРИЛОЖЕНИЕ Б

Скрипт создания таблиц базы данных

|  |
| --- |
| ------------------------------ ROLES ------------------------------  CREATE TABLE roles\_t (  role\_id NUMBER(10) GENERATED AS IDENTITY  (START WITH 1 INCREMENT BY 1),  role\_name VARCHAR2(30) NOT NULL,  CONSTRAINT role\_pk PRIMARY KEY (role\_id)  );  INSERT INTO roles\_t(role\_name) VALUES('USER');  INSERT INTO roles\_t(role\_name) VALUES('ADMIN');  ------------------------------ USERS ------------------------------  CREATE TABLE users\_t (  user\_id NUMBER(10) GENERATED AS IDENTITY  (START WITH 1 INCREMENT BY 1),  user\_login VARCHAR2(30) NOT NULL,  user\_password VARCHAR2(200) NOT NULL,  user\_role NUMBER(10) NOT NULL,  CONSTRAINT user\_pk PRIMARY KEY (user\_id),  CONSTRAINT user\_role\_fk FOREIGN KEY (user\_role) REFERENCES roles\_t(role\_id)  );  ---------------------------- NARRATORS ----------------------------  CREATE TABLE narrators\_t (  narrator\_id NUMBER(10) GENERATED AS IDENTITY  (START WITH 1 INCREMENT BY 1),  narrator\_name VARCHAR2(30) NOT NULL,  CONSTRAINT narrator\_pk PRIMARY KEY (narrator\_id)  );  ----------------------------- TOPICS ------------------------------  CREATE TABLE topics\_t (  topic\_id NUMBER(10) GENERATED AS IDENTITY  (START WITH 1 INCREMENT BY 1),  topic\_narrator NUMBER(10) NOT NULL,  topic\_name VARCHAR2(30) NOT NULL,  topic\_released NUMBER(10) NOT NULL,  topic\_blob BLOB DEFAULT EMPTY\_BLOB(),  CONSTRAINT topic\_pk PRIMARY KEY (topic\_id),  CONSTRAINT topic\_narrator\_fk FOREIGN KEY (topic\_narrator)  REFERENCES narrators\_t(narrator\_id)  );  ---------------------------- PODCASTS -----------------------------  CREATE TABLE podcasts\_t (  podcast\_id NUMBER(10) GENERATED AS IDENTITY  (START WITH 1 INCREMENT BY 1),  podcast\_narrator NUMBER(10) NOT NULL,  podcast\_topic NUMBER(10) NOT NULL,  podcast\_name VARCHAR2(30) NOT NULL,  podcast\_blob BLOB DEFAULT EMPTY\_BLOB(),  CONSTRAINT podcast\_pk PRIMARY KEY (podcast\_id),  CONSTRAINT podcast\_narrator\_fk FOREIGN KEY (podcast\_narrator)  REFERENCES narrators\_t(narrator\_id),  CONSTRAINT podcast\_topic\_fk FOREIGN KEY (podcast\_topic)  REFERENCES topics\_t(topic\_id)  );  ------------------------------ SAVED ------------------------------  CREATE TABLE saved\_t (  saved\_id NUMBER(10) GENERATED AS IDENTITY  (START WITH 1 INCREMENT BY 1),  saved\_user NUMBER(10) NOT NULL,  saved\_podcast NUMBER(10) NOT NULL,  CONSTRAINT saved\_pk PRIMARY KEY (saved\_id),  CONSTRAINT saved\_user\_fk FOREIGN KEY (saved\_user)  REFERENCES users\_t(user\_id),  CONSTRAINT saved\_podcast\_fk FOREIGN KEY (saved\_podcast)  REFERENCES podcasts\_t(podcast\_id)  ); |

# ПРИЛОЖЕНИЕ В

Скрипт создания представлений

|  |
| --- |
| CREATE VIEW user\_role\_view AS SELECT  USERS\_T.user\_login, USERS\_T.user\_password, ROLES\_T.role\_name  FROM USERS\_T  LEFT JOIN ROLES\_T  ON USERS\_T.user\_role = ROLES\_T.role\_id;  CREATE VIEW user\_role\_full\_view AS SELECT  USERS\_T.user\_id, USERS\_T.user\_login,  DecryptionPassword(USERS\_T.user\_password) as decr,  ROLES\_T.role\_name  FROM users\_t LEFT JOIN roles\_t  ON USERS\_T.user\_role = ROLES\_T.role\_id;  CREATE VIEW narrator\_topic\_view AS SELECT  TOPICS\_T.topic\_id, NARRATORS\_T.narrator\_name,  TOPICS\_T.topic\_name, TOPICS\_T.topic\_released,  TOPICS\_T.topic\_blob  FROM narrators\_t JOIN topics\_t  ON NARRATORS\_T.narrator\_id = TOPICS\_T.topic\_narrator;  CREATE VIEW narrator\_topic\_podcast\_view AS SELECT  PODCASTS\_T.podcast\_id, NARRATORS\_T.narrator\_name,  TOPICS\_T.topic\_name, PODCASTS\_T.podcast\_name,  TOPICS\_T.topic\_released, TOPICS\_T.topic\_blob,  PODCASTS\_T.podcast\_blob  FROM narrators\_t JOIN topics\_t  ON NARRATORS\_T.narrator\_id = TOPICS\_T.topic\_narrator  JOIN podcasts\_t  ON PODCASTS\_T.podcast\_topic = TOPICS\_T.topic\_id;  CREATE VIEW narrator\_topic\_podcast\_user\_view AS SELECT  USERS\_T.user\_id, PODCASTS\_T.podcast\_id,  NARRATORS\_T.narrator\_name, TOPICS\_T.topic\_name,  PODCASTS\_T.podcast\_name, TOPICS\_T.topic\_released,  TOPICS\_T.topic\_blob, PODCASTS\_T.podcast\_blob  FROM narrators\_t JOIN topics\_t  ON NARRATORS\_T.narrator\_id = TOPICS\_T.topic\_narrator  JOIN podcasts\_t  ON PODCASTS\_T.podcast\_topic = TOPICS\_T.topic\_id  JOIN saved\_t  ON PODCASTS\_T.podcast\_id = SAVED\_T.saved\_podcast  JOIN users\_t  ON SAVED\_T.saved\_user = USERS\_T.user\_id; |

# ПРИЛОЖЕНИЕ Г

Скрипт создания хранимых процедур

|  |
| --- |
| -- ПРОЦЕДУРА 1. Регистрация пользователя - RegisterUser  CREATE OR REPLACE PROCEDURE RegisterUser  (pUserLogin IN USERS\_T.user\_login%TYPE,  pUserPassword IN USERS\_T.user\_password%TYPE)  IS  cnt NUMBER;  BEGIN  SELECT COUNT(\*) INTO cnt FROM users\_t  WHERE UPPER(user\_login) = UPPER(pUserLogin);  IF (cnt = 0) THEN  INSERT INTO users\_t (user\_login, user\_password, user\_role)  VALUES (UPPER(pUserLogin),  EncryptionPassword(UPPER(pUserPassword)), 1);  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20001,  'This login is already taken!');  END IF;  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20002,  'An error occurred during registration: ' || SQLERRM);  END RegisterUser;  -- ПРОЦЕДУРА 2. Нахождение роли пользователя - CheckRole  CREATE OR REPLACE PROCEDURE CheckRole  (pUserLogin IN USERS\_T.user\_login%TYPE,  oUserRole OUT ROLES\_T.role\_name%TYPE)  IS  CURSOR role\_cursor IS  SELECT role\_name FROM user\_role\_view  WHERE UPPER(user\_login) = UPPER(pUserLogin);  BEGIN  OPEN role\_cursor;  FETCH role\_cursor INTO oUserRole;  CLOSE role\_cursor;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(-20004, 'Role not found!');  WHEN OTHERS THEN  IF role\_cursor%ISOPEN THEN role\_cursor;  END IF;  RAISE\_APPLICATION\_ERROR(-20001,  'An unexpected error occurred: ' || SQLERRM);  END CheckRole;  -- ПРОЦЕДУРА 3. Авторизация пользователя - LogInUser  CREATE OR REPLACE PROCEDURE LogInUser  (pUserLogin IN USERS\_T.user\_login%TYPE,  pUserPassword IN USERS\_T.user\_password%TYPE,  oUserId OUT USERS\_T.user\_id%TYPE,  oUserLogin OUT USERS\_T.user\_login%TYPE,  oUserRole OUT ROLES\_T.role\_name%TYPE)  IS  CURSOR user\_cursor IS  SELECT user\_id, user\_login FROM users\_t  WHERE UPPER(user\_login) = UPPER(pUserLogin) AND  user\_password=EncryptionPassword(UPPER(pUserPassword));  BEGIN  DBMS\_OUTPUT.PUT\_LINE('LogInUser: Checking user ' || pUserLogin);  OPEN user\_cursor;  FETCH user\_cursor INTO oUserId, oUserLogin;  IF user\_cursor%NOTFOUND THEN  RAISE\_APPLICATION\_ERROR(-20002,  'Incorrect login or password!');  ELSE  DBMS\_OUTPUT.PUT\_LINE('User found: ' || oUserLogin);  END IF;  CLOSE user\_cursor;  DBMS\_OUTPUT.PUT\_LINE('Checking user ' || oUserLogin ||  '`s role');  CheckRole(oUserLogin, oUserRole);  DBMS\_OUTPUT.PUT\_LINE('User role: ' || oUserRole);  EXCEPTION  WHEN OTHERS THEN  RAISE\_APPLICATION\_ERROR(-20003,  'Ошибка в процедуре LogInUser: ' || SQLERRM);  END LogInUser;  -- ПРОЦЕДУРА 4. Поиск информации о пользователе - SearchUser  CREATE OR REPLACE PROCEDURE SearchUser  (pUserLogin IN USERS\_T.user\_login%TYPE,  oUserLogin OUT USERS\_T.user\_login%TYPE,  oUserPassword OUT USERS\_T.user\_password%TYPE)  IS  CURSOR user\_cursor IS  SELECT user\_login, DecryptionPassword(user\_password)  FROM users\_t  WHERE UPPER(user\_login) = UPPER(pUserLogin);  BEGIN  OPEN user\_cursor;  FETCH user\_cursor INTO oUserLogin, oUserPassword;  IF user\_cursor%NOTFOUND THEN  RAISE\_APPLICATION\_ERROR(-20003, 'User is not found!');  END IF;  CLOSE user\_cursor;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(-20003, 'User is not found!');  WHEN OTHERS THEN  RAISE\_APPLICATION\_ERROR(-20001,  'An unexpected error occurred: ' || SQLERRM);  END SearchUser;  -- ПРОЦЕДУРА 5. Изменение логина пользователя - UpdateUserLogin  CREATE OR REPLACE PROCEDURE UpdateUserLogin  (pUserLogin IN USERS\_T.user\_login%TYPE,  pNewUserLogin IN USERS\_T.user\_login%TYPE)  IS  cnt NUMBER;  BEGIN  BEGIN  SELECT COUNT(\*) INTO cnt FROM users\_t  WHERE UPPER(user\_login) = UPPER(pUserLogin);  IF cnt = 0 THEN  RAISE\_APPLICATION\_ERROR(-20003, 'User is not found!');  END IF;  SELECT COUNT(\*) INTO cnt FROM users\_t  WHERE UPPER(user\_login) = UPPER(pNewUserLogin);  IF cnt > 0 THEN  RAISE\_APPLICATION\_ERROR(-20001,  'This login is already taken!');  END IF;  UPDATE users\_t SET user\_login = UPPER(pNewUserLogin)  WHERE UPPER(user\_login) = UPPER(pUserLogin);  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20004,  'User is not found (NO\_DATA\_FOUND)');  WHEN DUP\_VAL\_ON\_INDEX THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20001,  'This login is already taken (DUP\_VAL\_ON\_INDEX)');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20002,  'An error occurred during login update: ' || SQLERRM);  END;  END UpdateUserLogin;  -- ПРОЦЕДУРА 6. Изменение пароля пользователя - UpdateUserPassword  CREATE OR REPLACE PROCEDURE UpdateUserPassword  (pUserLogin IN USERS\_T.user\_login%TYPE,  pNewUserPassword IN USERS\_T.user\_password%TYPE)  IS  cnt NUMBER;  BEGIN  BEGIN  SELECT COUNT(\*) INTO cnt FROM users\_t  WHERE UPPER(user\_login) = UPPER(pUserLogin);  IF cnt = 0 THEN  RAISE\_APPLICATION\_ERROR(-20003, 'User is not found!');  END IF;  UPDATE users\_t SET user\_password =  EncryptionPassword(UPPER(pNewUserPassword))  WHERE UPPER(user\_login) = UPPER(pUserLogin);  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20004,  'User is not found (NO\_DATA\_FOUND)');  WHEN DUP\_VAL\_ON\_INDEX THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20001,  'Duplicate value error (DUP\_VAL\_ON\_INDEX)');  WHEN VALUE\_ERROR THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20005,  'Value error: Check input data!');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20002,  'An unexpected error occurred: ' || SQLERRM);  END;  END UpdateUserPassword;  -- ПРОЦЕДУРА 7. Удаление аккаунта пользователя - DeleteUser  CREATE OR REPLACE PROCEDURE DeleteUser  (pUserLogin IN USERS\_T.user\_login%TYPE)  IS  cnt NUMBER;  userId USERS\_T.user\_id%TYPE;  BEGIN  SELECT COUNT(\*) INTO cnt FROM users\_t  WHERE UPPER(user\_login) = UPPER(pUserLogin);  IF (cnt != 0) THEN  SELECT user\_id INTO userId FROM users\_t  WHERE UPPER(user\_login) = UPPER(pUserLogin);  DELETE FROM users\_t  WHERE UPPER(user\_login) = UPPER(pUserLogin);  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20003, 'User is not found!');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20004,  'User data could not be found during deletion!');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20005,  'An unexpected error occurred: ' || SQLERRM);  END DeleteUser;  -- ПРОЦЕДУРА 8. Добавление нового диктора - AddNarrator  CREATE OR REPLACE PROCEDURE AddNarrator  (pNarratorName IN NARRATORS\_T.narrator\_name%TYPE)  IS  cnt NUMBER;  BEGIN  SELECT COUNT(\*) INTO cnt FROM narrators\_t  WHERE UPPER(narrator\_name) = UPPER(pNarratorName);  IF (cnt = 0) THEN  INSERT INTO narrators\_t (narrator\_name)  VALUES (UPPER(pNarratorName));  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20005,  'This name is already taken by another narrator!');  END IF;  EXCEPTION  WHEN DUP\_VAL\_ON\_INDEX THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20006,  'Duplicate entry: This narrator name already exists.');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20007,  'An unexpected error occurred: ' || SQLERRM);  END AddNarrator;  -- ПРОЦЕДУРА 9. Добавление новой темы - AddTopic  CREATE OR REPLACE PROCEDURE AddTopic  (pNarratorName IN NARRATORS\_T.narrator\_name%TYPE,  pTopicName IN TOPICS\_T.topic\_name%TYPE,  pTopicReleased IN TOPICS\_T.topic\_released%TYPE)  IS  narratorId NARRATORS\_T.narrator\_id%TYPE;  cnt NUMBER;  BEGIN  SELECT COUNT(\*) INTO cnt FROM narrator\_topic\_view  WHERE UPPER(topic\_name) = UPPER(pTopicName) AND  UPPER(narrator\_name) = UPPER(pNarratorName);  IF (cnt = 0) THEN  SELECT narrator\_id INTO narratorId FROM narrators\_t  WHERE UPPER(narrator\_name) = UPPER(pNarratorName);  INSERT INTO topics\_t  (topic\_narrator, topic\_name, topic\_released)  VALUES (narratorId, UPPER(pTopicName), pTopicReleased);  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20006,  'This narrator already has this topic!');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20007, 'Narrator not found!');  WHEN DUP\_VAL\_ON\_INDEX THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20008,  'Duplicate entry: The topic already exists.');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20009,  'An unexpected error occurred: ' || SQLERRM);  END AddTopic;  -- ПРОЦЕДУРА 10. Добавление нового подкаста - AddPodcast  CREATE OR REPLACE PROCEDURE AddPodcast  (pNarratorName IN NARRATORS\_T.narrator\_name%TYPE,  pTopicName IN TOPICS\_T.topic\_name%TYPE,  pPodcastName IN PODCASTS\_T.podcast\_name%TYPE)  IS  narratorId NARRATORS\_T.narrator\_id%TYPE;  topicId TOPICS\_T.topic\_id%TYPE;  cnt NUMBER;  BEGIN  SELECT COUNT(\*) INTO cnt FROM narrator\_topic\_podcast\_view  WHERE UPPER(topic\_name) = UPPER(pTopicName)  AND UPPER(narrator\_name) = UPPER(pNarratorName)  AND UPPER(podcast\_name) = UPPER(pPodcastName);  IF (cnt = 0) THEN  SELECT narrator\_id INTO narratorId FROM narrators\_t  WHERE UPPER(narrator\_name) = UPPER(pNarratorName);  SELECT topic\_id INTO topicId FROM topics\_t  WHERE UPPER(topic\_name) = UPPER(pTopicName);  INSERT INTO podcasts\_t  (podcast\_narrator, podcast\_topic, podcast\_name)  VALUES (narratorId, topicId, UPPER(pPodcastName));  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20007,  'This podcast + topic + narrator combo already exists!');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20008,  'Narrator or topic not found!');  WHEN DUP\_VAL\_ON\_INDEX THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20009,  'Duplicate entry: The podcast already exists.');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20010,  'An unexpected error occurred: ' || SQLERRM);  END AddPodcast;  -- ПРОЦЕДУРА 11. Изменение имени диктора - UpdateNarrator  CREATE OR REPLACE PROCEDURE UpdateNarrator  (pOldNarrator IN NARRATORS\_T.narrator\_name%TYPE,  pNewNarrator IN NARRATORS\_T.narrator\_name%TYPE)  IS  cnt NUMBER;  BEGIN  SELECT COUNT(\*) INTO cnt FROM narrators\_t  WHERE UPPER(narrator\_name) = UPPER(pOldNarrator);  IF (cnt != 0) THEN  SELECT COUNT(\*) INTO cnt FROM narrators\_t  WHERE UPPER(narrator\_name) = UPPER(pNewNarrator);  IF (cnt = 0) THEN  UPDATE narrators\_t  SET narrator\_name = UPPER(pNewNarrator)  WHERE UPPER(narrator\_name) = UPPER(pOldNarrator);  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20005,  'This narrator already exists!');  END IF;  ELSE  RAISE\_APPLICATION\_ERROR(-20008, 'Narrator is not found!');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20009,  'Narrator not found during update process!');  WHEN DUP\_VAL\_ON\_INDEX THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20010,  'Duplicate narrator name error!');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20011,  'An unexpected error occurred during narrator update: '  || SQLERRM);  END UpdateNarrator;  -- ПРОЦЕДУРА 12. Изменение названия темы - UpdateTopicName  CREATE OR REPLACE PROCEDURE UpdateTopicName  (pTopicId IN TOPICS\_T.topic\_id%TYPE,  pNewName IN TOPICS\_T.topic\_name%TYPE)  IS  cnt NUMBER;  BEGIN  SELECT COUNT(\*) INTO cnt FROM topics\_t WHERE topic\_id = pTopicId;  IF (cnt != 0) THEN  UPDATE topics\_t SET topic\_name = UPPER(pNewName)  WHERE topic\_id = pTopicId;  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20009, 'Topic is not found!');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20010,  'No topic found with the given ID.');  WHEN DUP\_VAL\_ON\_INDEX THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20011,  'A topic with this name already exists.');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20012,  'An unexpected error occurred: ' || SQLERRM);  END UpdateTopicName;  -- ПРОЦЕДУРА 13. Изменение года выпуска темы - UpdateTopicYear  CREATE OR REPLACE PROCEDURE UpdateTopicYear  (pTopicId IN TOPICS\_T.topic\_id%TYPE,  pNewYear IN TOPICS\_T.topic\_released%TYPE)  IS  cnt NUMBER;  BEGIN  SELECT COUNT(\*) INTO cnt FROM topics\_t WHERE topic\_id = pTopicId;  IF (cnt != 0) THEN  UPDATE topics\_t SET topic\_released = pNewYear  WHERE topic\_id = pTopicId;  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20009, 'Topic is not found!');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20010,  'No topic found with the given ID during year update.');  WHEN VALUE\_ERROR THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20011,  'Invalid year format. Please enter a valid year.');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20012,  'An unexpected error occurred: ' || SQLERRM);  END UpdateTopicYear;  -- ПРОЦЕДУРА 14. Изменение названия подкаста - UpdatePodcastName  CREATE OR REPLACE PROCEDURE UpdatePodcastName  (pPodcastId IN PODCASTS\_T.podcast\_id%TYPE,  pNewName IN PODCASTS\_T.podcast\_name%TYPE)  IS  cnt NUMBER;  BEGIN  SELECT COUNT(\*) INTO cnt FROM podcasts\_t  WHERE podcast\_id = pPodcastId;  IF (cnt != 0) THEN  UPDATE podcasts\_t SET podcast\_name = UPPER(pNewName)  WHERE podcast\_id = pPodcastId;  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20010, 'Podcast is not found!');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20011,  'No podcast found with the given ID.');  WHEN DUP\_VAL\_ON\_INDEX THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20012,  'A podcast with this name already exists.');  WHEN VALUE\_ERROR THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20013,  'Invalid input value. Please check the data format.');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20014,  'An unexpected error occurred: ' || SQLERRM);  END UpdatePodcastName;  -- ПРОЦЕДУРА 15. Пакетная вставка - RegisterUsersBatch  CREATE OR REPLACE PROCEDURE RegisterUsersBatch  (pUsernames IN SYS.ODCIVARCHAR2LIST,  pPasswords IN SYS.ODCIVARCHAR2LIST)  IS  BEGIN  IF pUsernames.COUNT != pPasswords.COUNT THEN  RAISE\_APPLICATION\_ERROR(-20020,  'The number of usernames and passwords must be equal.');  END IF;  FOR i IN 1 .. pUsernames.COUNT LOOP  BEGIN  INSERT INTO users\_t  (user\_login, user\_password, user\_role)  VALUES (UPPER(pUsernames(i)),  EncryptionPassword(UPPER(pPasswords(i))), 1);  EXCEPTION  WHEN DUP\_VAL\_ON\_INDEX THEN  RAISE\_APPLICATION\_ERROR(-20021,  'Username "' || pUsernames(i) || '" is already  taken.');  WHEN VALUE\_ERROR THEN  RAISE\_APPLICATION\_ERROR(-20022,  'Invalid data format for user "' || pUsernames(i)  || '".');  WHEN OTHERS THEN  RAISE\_APPLICATION\_ERROR(-20023,  'An error occurred during user registration for  "' || pUsernames(i) || '": ' || SQLERRM);  END;  END LOOP;  COMMIT;  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20024,  'An error occurred: ' || SQLERRM);  END RegisterUsersBatch;  -- ПРОЦЕДУРА 16. Добавление 100000 пользователей - Insert100KUsers  CREATE OR REPLACE PROCEDURE Insert100KUsers IS  usernames SYS.ODCIVARCHAR2LIST := SYS.ODCIVARCHAR2LIST();  passwords SYS.ODCIVARCHAR2LIST := SYS.ODCIVARCHAR2LIST();  BEGIN  FOR i IN 1 .. 100000 LOOP  usernames.EXTEND;  passwords.EXTEND;  usernames(usernames.COUNT) := 'user' || i;  passwords(passwords.COUNT) := 'pass' || i;  IF MOD(i, 1000) = 0 THEN  BEGIN  RegisterUsersBatch(usernames, passwords);  COMMIT;  usernames.DELETE;  passwords.DELETE;  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20030,  'An error occurred during batch insertion at  iteration ' || i || ': ' || SQLERRM);  END;  END IF;  END LOOP;  IF usernames.COUNT > 0 THEN  BEGIN  RegisterUsersBatch(usernames, passwords);  COMMIT;  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20031,  'An error occurred: ' || SQLERRM);  END;  END IF;  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20032,  'An unexpected error occurred: ' || SQLERRM);  END Insert100KUsers;  -- ПРОЦЕДУРА 17. Экспорт подкастов в XML-файл - PodcastExport  CREATE OR REPLACE PROCEDURE PodcastExport IS  rc sys\_refcursor;  doc DBMS\_XMLDOM.DOMDocument;  BEGIN  OPEN rc FOR SELECT podcast\_id, podcast\_name FROM podcasts\_t;  doc := DBMS\_XMLDOM.NewDOMDocument(XMLTYPE(rc));  DBMS\_XMLDOM.WRITETOFILE(doc, 'CWDIR/PodcastExport.xml');  DBMS\_XMLDOM.FREEDOCUMENT(doc);  CLOSE rc;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(-20050, 'No podcasts found.');  WHEN INVALID\_CURSOR THEN  RAISE\_APPLICATION\_ERROR(-20051, 'Invalid cursor detected.');  WHEN OTHERS THEN  IF DBMS\_XMLDOM.ISNULL(doc) = FALSE THEN  DBMS\_XMLDOM.FREEDOCUMENT(doc);  END IF;  IF rc%ISOPEN THEN  CLOSE rc;  END IF;  RAISE\_APPLICATION\_ERROR(-20052,  'An unexpected error: ' || SQLERRM);  END PodcastExport;  -- ПРОЦЕДУРА 18. Экспорт пользователей в XML-файл - UserExport  CREATE OR REPLACE PROCEDURE UserExport IS  v\_file\_handle UTL\_FILE.FILE\_TYPE;  CURSOR c\_user IS  SELECT user\_id, user\_login, decr, role\_name  FROM user\_role\_full\_view;  v\_user\_row c\_user%ROWTYPE;  BEGIN  v\_file\_handle := UTL\_FILE.FOPEN('CWDIR', 'UserExport.xml', 'W');  UTL\_FILE.PUT\_LINE(v\_file\_handle,  '<?xml version="1.0" encoding="UTF-8"?>');  UTL\_FILE.PUT\_LINE(v\_file\_handle, '<users>');  OPEN c\_user;  LOOP  FETCH c\_user INTO v\_user\_row;  EXIT WHEN c\_user%NOTFOUND;  UTL\_FILE.PUT\_LINE(v\_file\_handle, ' <user>');  UTL\_FILE.PUT\_LINE(v\_file\_handle, ' <user\_id>' ||  v\_user\_row.user\_id || '</user\_id>');  UTL\_FILE.PUT\_LINE(v\_file\_handle, ' <user\_login>' ||  v\_user\_row.user\_login || '</user\_login>');  UTL\_FILE.PUT\_LINE(v\_file\_handle, ' <decr>' ||  v\_user\_row.decr || '</decr>');  UTL\_FILE.PUT\_LINE(v\_file\_handle, ' <role\_name>' ||  v\_user\_row.role\_name || '</role\_name>');  UTL\_FILE.PUT\_LINE(v\_file\_handle, ' </user>');  END LOOP;  CLOSE c\_user;  UTL\_FILE.PUT\_LINE(v\_file\_handle, '</users>');  UTL\_FILE.FCLOSE(v\_file\_handle);  EXCEPTION  WHEN UTL\_FILE.INVALID\_PATH THEN  IF UTL\_FILE.IS\_OPEN(v\_file\_handle) THEN  UTL\_FILE.FCLOSE(v\_file\_handle);  END IF;  RAISE\_APPLICATION\_ERROR(-20053, 'Invalid file path.');  WHEN UTL\_FILE.WRITE\_ERROR THEN  IF UTL\_FILE.IS\_OPEN(v\_file\_handle) THEN  UTL\_FILE.FCLOSE(v\_file\_handle);  END IF;  RAISE\_APPLICATION\_ERROR(-20054, 'Write error occurred.');  WHEN UTL\_FILE.INVALID\_OPERATION THEN  IF UTL\_FILE.IS\_OPEN(v\_file\_handle) THEN  UTL\_FILE.FCLOSE(v\_file\_handle);  END IF;  RAISE\_APPLICATION\_ERROR(-20055, 'Invalid file operation.');  WHEN OTHERS THEN  IF UTL\_FILE.IS\_OPEN(v\_file\_handle) THEN  UTL\_FILE.FCLOSE(v\_file\_handle);  END IF;  RAISE\_APPLICATION\_ERROR(-20057,  'An unexpected error occurred: ' || SQLERRM);  END UserExport;  -- ПРОЦЕДУРА 19. Импорт дикторов из XML-файла - NarratorImport  CREATE OR REPLACE PROCEDURE NarratorImport IS  BEGIN  INSERT INTO narrators\_t (narrator\_name)  SELECT ExtractValue(VALUE(narrator), '//NAME') AS narrator\_name  FROM TABLE(XMLSequence(EXTRACT(  XMLTYPE(bfilename('CWDIR', 'NarratorImport.xml'),  nls\_charset\_id('UTF-8')), '/ROWSET/ROW'))  ) narrator;  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(-20060, 'No narrators found.');  WHEN UTL\_FILE.INVALID\_PATH THEN  RAISE\_APPLICATION\_ERROR(-20061, 'Invalid file path.');  WHEN UTL\_FILE.READ\_ERROR THEN  RAISE\_APPLICATION\_ERROR(-20062, 'Error reading from file.');  WHEN VALUE\_ERROR THEN  RAISE\_APPLICATION\_ERROR(-20063, 'Invalid data format.');  WHEN OTHERS THEN  RAISE\_APPLICATION\_ERROR(-20064,  'An unexpected error occurred: ' || SQLERRM);  END NarratorImport;  -- ПРОЦЕДУРА 20. Удаление диктора - DeleteNarrator  CREATE OR REPLACE PROCEDURE DeleteNarrator  (pId IN NARRATORS\_T.narrator\_id%TYPE)  IS  cnt NUMBER;  BEGIN  DELETE FROM SAVED\_T WHERE saved\_podcast IN (  SELECT podcast\_id FROM PODCASTS\_T  WHERE podcast\_narrator = pId  );  DELETE FROM PODCASTS\_T WHERE podcast\_narrator = pId;  DELETE FROM TOPICS\_T WHERE topic\_narrator = pId;  SELECT COUNT(\*) INTO cnt FROM narrators\_t  WHERE narrator\_id = pId;  IF (cnt != 0) THEN  DELETE FROM narrators\_t WHERE narrator\_id = pId;  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20008, 'Narrator is not found!');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  NULL;  WHEN OTHERS THEN  RAISE\_APPLICATION\_ERROR(-20001,  'An unexpected error occurred: ' || SQLERRM);  END DeleteNarrator;  -- ПРОЦЕДУРА 21. Удаление темы - DeleteTopic  CREATE OR REPLACE PROCEDURE DeleteTopic  (pId IN TOPICS\_T.topic\_id%TYPE)  IS  cnt NUMBER;  BEGIN  DELETE FROM SAVED\_T WHERE saved\_podcast IN (  SELECT podcast\_id FROM PODCASTS\_T WHERE podcast\_topic = pId);  DELETE FROM PODCASTS\_T WHERE podcast\_topic = pId;  SELECT COUNT(\*) INTO cnt FROM topics\_t WHERE topic\_id = pId;  IF (cnt != 0) THEN  DELETE FROM topics\_t WHERE topic\_id = pId;  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20009, 'Topic is not found!');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(-20010, 'No data found!');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20001,  'An unexpected error occurred: ' || SQLERRM);  END DeleteTopic;  -- ПРОЦЕДУРА 22. Удаление подкаста - DeletePodcast  CREATE OR REPLACE PROCEDURE DeletePodcast  (pId IN PODCASTS\_T.podcast\_id%TYPE)  IS  cnt NUMBER;  BEGIN  DELETE FROM SAVED\_T WHERE saved\_podcast = pId;  SELECT COUNT(\*) INTO cnt FROM podcasts\_t WHERE podcast\_id = pId;  IF (cnt != 0) THEN  DELETE FROM podcasts\_t WHERE podcast\_id = pId;  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20008, 'Podcast is not found!');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(-20009, 'No data found!');  WHEN DUP\_VAL\_ON\_INDEX THEN  RAISE\_APPLICATION\_ERROR(-20010,  'Duplicate value error occurred!');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20011,  'An unexpected error occurred: ' || SQLERRM);  END DeletePodcast;  -- ПРОЦЕДУРА 23. Удаление пользователя администратором - DeleteUserAdmin  CREATE OR REPLACE PROCEDURE DeleteUserAdmin  (pLogin IN USERS\_T.user\_login%TYPE)  IS  cnt NUMBER;  BEGIN  SELECT COUNT(\*) INTO cnt FROM users\_t  WHERE UPPER(user\_login) = UPPER(pLogin);  IF (cnt != 0) THEN  DELETE FROM users\_t WHERE UPPER(user\_login) = UPPER(pLogin);  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20003, 'User is not found!');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(-20004, 'No data found!');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20001,  'An unexpected error occurred: ' || SQLERRM);  END DeleteUserAdmin;  -- ПРОЦЕДУРА 24. Сохранение подкаста в плейлист - SavePodcast  CREATE OR REPLACE PROCEDURE SavePodcast  (pUserId IN SAVED\_T.saved\_user%TYPE,  pPodcastId IN SAVED\_T.saved\_podcast%TYPE)  IS  cnt NUMBER;  BEGIN  SELECT COUNT(\*) INTO cnt FROM saved\_t  WHERE saved\_user = pUserId AND saved\_podcast = pPodcastId;  IF (cnt = 0) THEN  INSERT INTO saved\_t (saved\_user, saved\_podcast)  VALUES (pUserId, pPodcastId);  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20011,  'This podcast is already in the playlist!');  END IF;  EXCEPTION  WHEN DUP\_VAL\_ON\_INDEX THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20012,  'Duplicate entry error: Podcast is already saved!');  WHEN NO\_DATA\_FOUND THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20013, 'No data found.');  WHEN VALUE\_ERROR THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20014, 'Invalid input value.');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20015,  'An unexpected error occurred: ' || SQLERRM);  END SavePodcast;  -- ПРОЦЕДУРА 25. Удаление подкаста из плейлиста - RemovePodcast  CREATE OR REPLACE PROCEDURE RemovePodcast  (pUserId IN SAVED\_T.saved\_user%TYPE,  pPodcastId IN SAVED\_T.saved\_podcast%TYPE)  IS  cnt NUMBER;  BEGIN  SELECT COUNT(\*) INTO cnt FROM saved\_t  WHERE saved\_user = pUserId AND saved\_podcast = pPodcastId;  IF (cnt != 0) THEN  DELETE FROM saved\_t WHERE saved\_user = pUserId AND  saved\_podcast = pPodcastId;  COMMIT;  ELSE  RAISE\_APPLICATION\_ERROR(-20012,  'This podcast is not on the playlist!');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20013, 'No matching data found.');  WHEN VALUE\_ERROR THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20014, 'Invalid input value.');  WHEN INVALID\_NUMBER THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20015, 'Invalid number format.');  WHEN OTHERS THEN  ROLLBACK;  RAISE\_APPLICATION\_ERROR(-20016,  'An unexpected error occurred: ' || SQLERRM);  END RemovePodcast; |