МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных технологий

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

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Контроль и управление задачами малых групп»

Выполнил студент Артём Елизавета Владимировна

(Ф.И.О.)

Руководитель проекта асс., Нистюк О.А.

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

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

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

Консультант:  асс., Нистюк О.А.

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

Нормоконтролер: асс., Нистюк О.А.

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

Курсовой проект защищен с оценкой

Минск 2022

**Реферат**

Пояснительная записка курсового проекта содержит 00 страниц пояснительной записки, 00 иллюстраций, 00 источников литературы, 00 приложение.

C#, .NET FRAMEWORK 4.7.2, Windows presentation foundation (WPF), ORACLE 21XE, ORACLE.MANAGEDDATAACCESS FOR C#, ORACLE Репликация данных, SOLARWINDS DPA, XML IMPORT, XML EXPORT.

Цель курсового проекта: разработка базы данных для контроля задач и управление проектами небольших групп с применением технологии с технологией репликации данных между серверами СУБД.

Первая глава описывает приложения, которыми был вдохновлен курсовой проект, и пункты, которые следует имплементировать.

Вторая глава описывает процесс создания и настройки базы данных, с которой будет происходить взаимодействие.

Третья глава описывает средства взаимодействия с базой данных, предоставленные платформой .NET и драйвером Oracle.ManagedDataAccess.

Четвёртая глава описывает изученные и использованные в процессе разработки технологии.

В пятой главе приведено руководство пользователя по взаимодействию с приложением и описаны базовые тестовые случаи.

В заключении приведены результаты проделанной работы.

**Abstract**

The course project explanatory note consists of 00 pages of explanatory note, 00 illustrations, 00 literature sources, 00 appendix.

C#, .NET FRAMEWORK 4.7.2, Windows presentation foundation (WPF), ORACLE 21XE, ORACLE.MANAGEDDATAACCESS FOR C#, ORACLE Репликация данных, SOLARWINDS DPA, XML IMPORT, XML EXPORT.

The goal of course project is to design database for task control and small group project management using technology with data replication technology between DBMS servers.

The first chapter describes the main inspirations behind the project’s theme and also some reference points.

The second chapter describes the process of building and setting up the main database.

The third chapter describes methods of interaction with database using tools provided by .NET platform and Oracle.ManagedDataAcess driver.

The fourth chapter describes the technologies used or studied during development.

The fifth chapter is a users’ guide to the app combined with test cases.

Conclusion states the results of this project.

**Содержание**

[Введение 5](#_Toc117446637)

[1 Аналитический обзор литературы 6](#_Toc117446638)

[2 Архитектура базы данных 8](#_Toc117446639)

[2.1 Таблицы базы данных 8](#_Toc117446640)

[2.2 Процедуры и функции для решения поставленных задач 9](#_Toc117446641)

[2.2.1 Идентификация пользователей 9](#_Toc117446642)

[2.2.2 Процедуры для манипуляций на странице «Профиль» 12](#_Toc117446643)

[2.2.3 Процедуры для добавления данных в таблицы 13](#_Toc117446644)

[2.2.4 Процедуры для изменения данных в таблицах 13](#_Toc117446645)

[2.2.5 Процедуры для удаления данных из таблиц 14](#_Toc117446646)

[2.2.6 Процедура для добавления 100 000 строк 15](#_Toc117446647)

[2.2.7 Процедура для экспорта и импорта таблиц в XML 16](#_Toc117446648)

[2.3 Представления базы данных 16](#_Toc117446649)

[2.4 Пользователь базы данных 17](#_Toc117446650)

[3 Обоснование технических приемов программирования 18](#_Toc117446651)

[4 Описание технологий 20](#_Toc117446652)

[4.1 Мультимедийные типы данных 20](#_Toc117446653)

[4.2 Средства диагностики 23](#_Toc117446654)

[5 Руководство пользователя, тестирование 26](#_Toc117446655)

[Заключение 37](#_Toc117446656)

[Список литературы 38](#_Toc117446657)

[Приложение А 39](#_Toc117446658)

Введение

Целью данной работы была разработка реляционной базы данных на тему «Контроль задач и управление проектами небольших групп». База данных должна быть составлена для контроля задач и управление проектами небольших групп. Так же было 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 12c, 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. Для взаимодействия с сервером базы данных Oracle использовался пакет Oracle Managed Data Access для .NET.

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 моем курсовом проекте используется технология шифрования паролей от аккаунта перед записью их в базу данных. А также использовалась технологией репликации данных между серверами СУБД для резервирования в случае выхода из строя одного из серверов.

Основные требования к приложению:

* Создание команд пользователей.
* Вход и регистрация в приложение.
* Добавление пользователей в команды.
* Реализация ролей администратора и рядового пользователя для команд.
* Добавление задач и управление ими.
* Отображение статистики по выполнению задач.
* Взаимодействие с базой данных при помощи хранимых процедур.

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

1. Аналитический обзор литературы

В современном мире большая часть работы осуществляется в команде людей. Для более продуктивной коммуникации и трудоспособности необходимо отслеживать задачи и результаты их выполнения.

Одним из примеров использования приложений для контроля и отслеживания задач является IT-сфера, где над разработкой работают люди в команде и каждый выполняет какие-то свои задачи, которые зачастую ограничиваются крайним сроком выполнения работы.

Одним из основных требований к подобным приложениям является надежность, возможность аналитики, отслеживание задач и контроль их выполнения.

При разработке приложения было обращено внимание на некоторые популярные приложения с подобной задачей. Они рассмотрены ниже.

Trello — облачная программа для управления проектами небольших групп, разработанная Fog Creek Software. Trello использует парадигму для управления проектами, известную как канбан, метод, который первоначально был популяризирован Toyota в 1980-х для управления цепочками поставок. Интерфейс этого сервиса можно увидеть на рисунке 1.1.

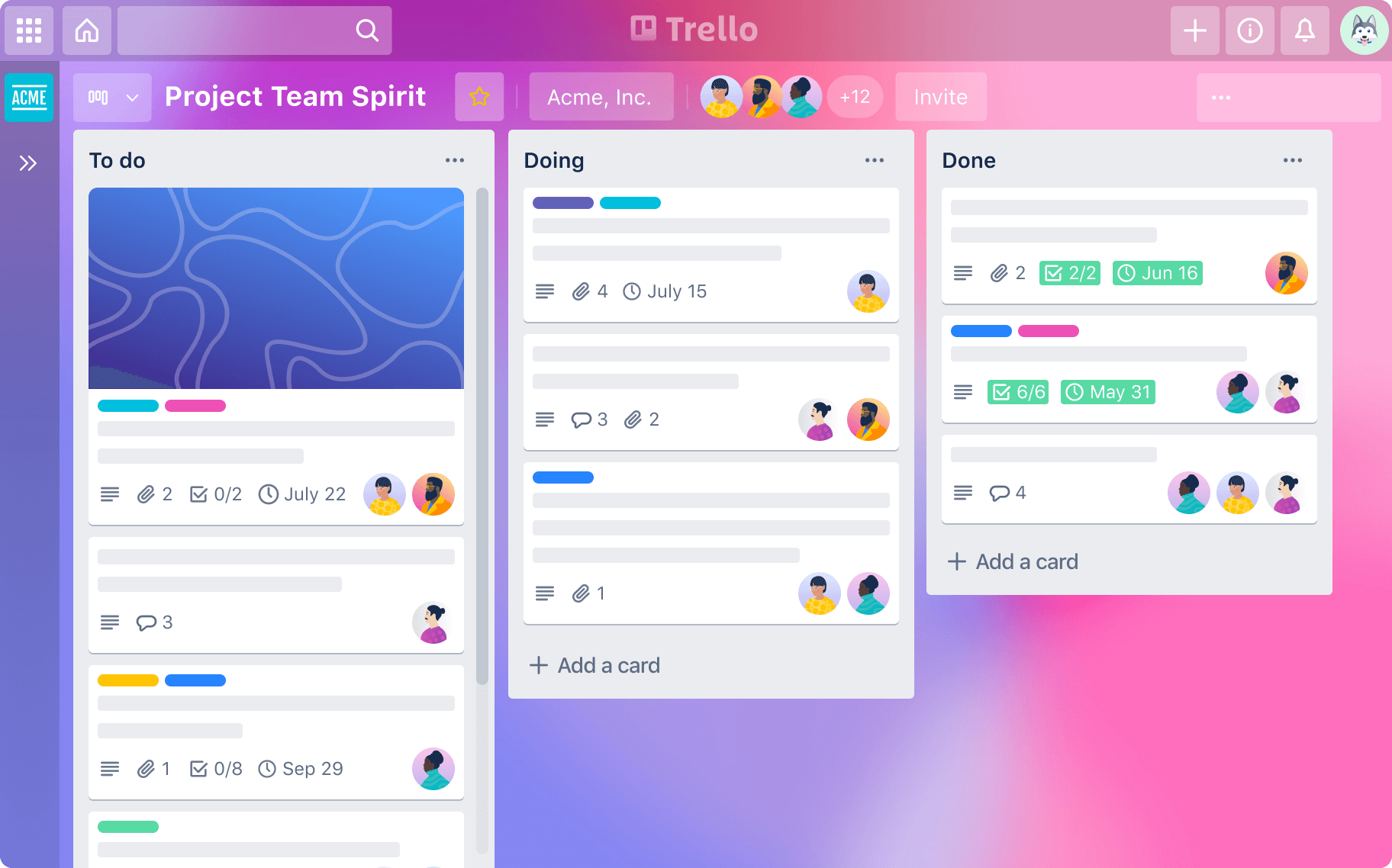


Рисунок 1.1 – Страница с задачами Trello

На странице с задачами в Trello четко видны блоки с задачами, пользователи с доступом к данным доскам. Также можно отметить, что у одного пользователя есть множество досок, как и команд, участником которых он может являться. В Trello есть возможность изменения профиля пользователя и возможность изменения дизайна для определенных досок.

Jira — коммерческая система отслеживания ошибок, предназначена для организации взаимодействия с пользователями, хотя в некоторых случаях используется и для управления проектами. Разработана компанией Atlassian, является одним из двух её основных продуктов. Имеет веб-интерфейс. Рассмотрим интерфейс на рисунке 1.2.

В Jira на странице с задачами также можно заметить список с необходимыми к выполнению задачами, их приоритет, срок исполнения, статус. В боковом меню можно перейти к самой доске с задачами, календарю, графику выполнения задач.

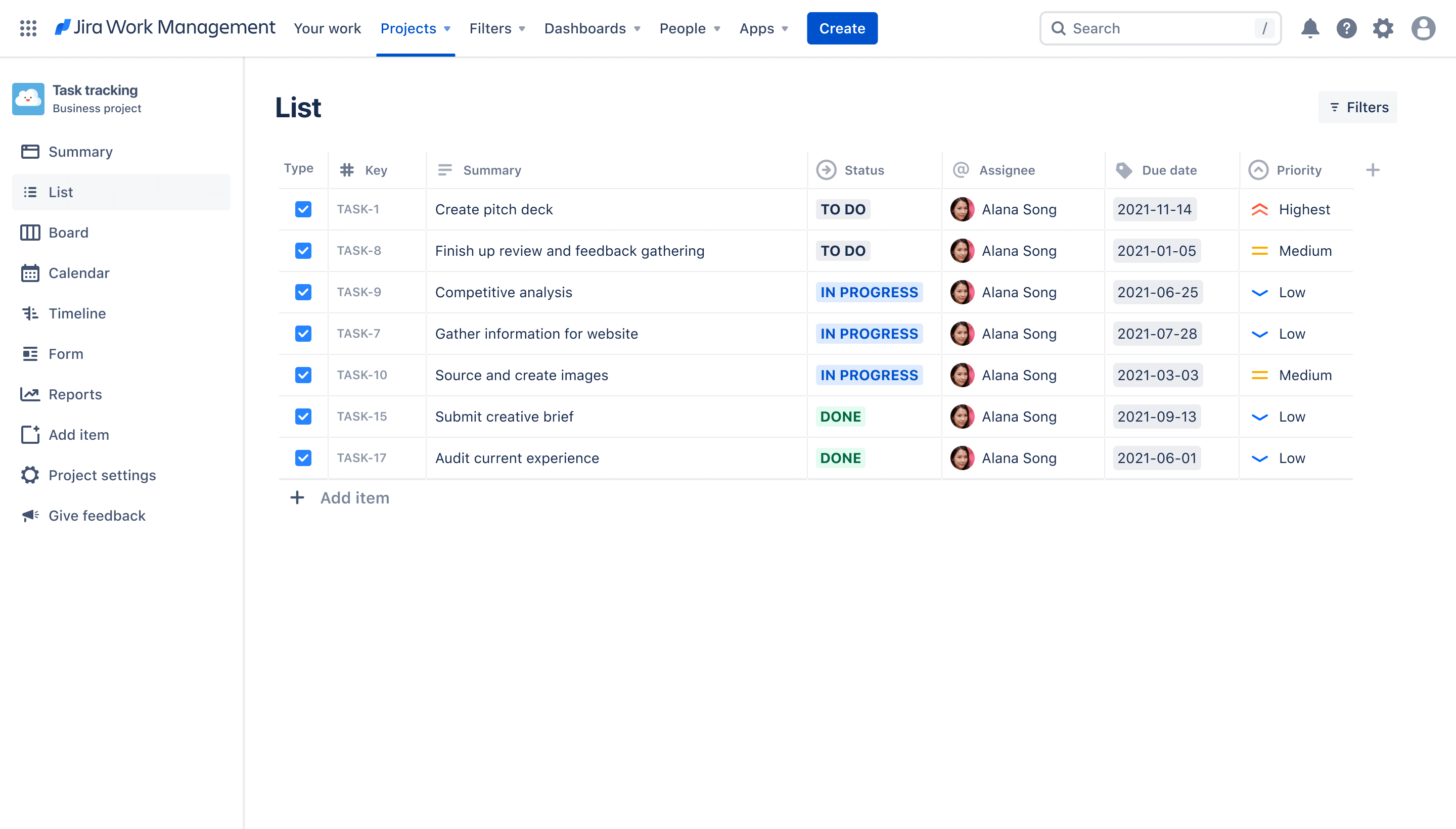


Рисунок 1.2 – Страница с задачами Jira

Разобрав два приведенных выше сервиса для контроля задач, можно сформировать общий функционал необходимый для работы приложения.

Прежде всего, естественно, база данных должна корректно работать с большим количеством задач и информации о них. Также стоит ввести меню настроек для пользователя, чтобы он мог изменить свой пароль или логин, оформление профиля по желанию.

1. Архитектура базы данных

При разработке приложения для курсового проекта была использована база данных Oracle 12c.

* 1. Таблицы базы данных

Для реализации базы данных было разработано 9 таблиц. Диаграмма базы данных со структурой связей представлена в приложении А.

При разработке приложение получило рабочее название «Noto», соответственно схема базы данных называется DBNoto. В ее структуру входят следующие таблицы: AppRoleTable, UserTable, TaskStatuses, TaskPriorities, TaskComments, TaskTable, TeamTable, UserTeamPrivs, UserTeamPrivTable. Ниже мы разберем каждую из них более подробно.

Таблица AppRoleTable представляет собой перечень ролей, которые могут иметь пользователи приложения. Имеющиеся столбцы: RoleID (идентификатор роли, типа number(10), первичный ключ), RoleName (текстовое название роли, тип varchar2(30)). Имеющиеся по умолчанию роли: USER (рядовой пользователь) и ADMIN (пользователь с правами администратора).

Таблица UserTable содержит информацию о пользователях, а именно: UserID (идентификатор пользователя, тип number(10), первичный ключ), UserLogin (логин пользователя, типа varchar2(30)), UserPassword (пароль пользователя, тип varchar2(200), шифруется в процессе регистрации и расшифровывается при проверке во время авторизации), UserName (имя пользователя, типа varchar2(30)), UserLastName (фамилия пользователя, типа varchar2(30)), UserPhoneNumber (номер телефона пользователя, типа varchar2(13)), UserEmail (электронная почта пользователя, типа varchar2(30)), UserIcon (изображение профиля пользователя), UserRole (идентификатор роли, тип number(10), внешний ключ).

Таблица TaskStatuses содержит информацию о статусе задач: TaskStatusID (идентификатор статуса задачи, типа number(10), первичный ключ), TaskStatusName (название статуса задачи, тип varchar2(30)). Имеющиеся по умолчанию статусы: NOT STARTED (выполнение задачи не начато), STARTED (выполнение задачи начато), IN PROCESS (задача выполняется) и READY (задача выполнена).

Таблица TaskPriorities содержит информацию приоритете выполнения задачи: TaskPriorityID (идентификатор команды, типа number(10), первичный ключ), TaskPriorityName (название приоритета задачи, тип varchar2(30)). Имеющиеся по умолчанию приоритеты: HIGH (высокий приоритет), MEDIUM (средний приоритет) и LOW (низкий приоритет).

Таблица TaskComments содержит информацию о комментариях к задачам: ComID (идентификатор комментария, типа number(10), первичный ключ), ComDate (дата создания комментария, типа varchar2(30)), ComText (текст комментария, тип varchar2(100)), ComUser (автор комментария, типа number(10), внешний ключ).

Таблица TaskTable содержит задачи и информацию о них: TaskID (идентификатор задачи, тип number(10), первичный ключ), TaskStatus (идентификатор состояния задачи, тип number(10), внешний ключ), TaskPriority (идентификатор приоритета задачи, тип number(10), внешний ключ), TaskTitle (название задачи, тип varchar2(30)), TaskDescription (текстовое описание задачи, тип varchar2(200)), CreationDate (дата создания задачи, типа varchar2(30)), DeadlineDate (дата завершения задачи, типа varchar2(30)), Comments (идентификатор комментария пользователя к задаче, типа varchar2(10), внешний ключ).

Таблица TeamTable содержит информацию о командах: TeamID (идентификатор команды, типа number(10), первичный ключ), TeamName (название команды, тип varchar2(10)), TeamIcon (изображение команды), TeamTasks (идентификатор задач команды, типа number(10), внешний ключ).

Таблица UserTeamPrivs отвечает за хранение информации о привилегиях пользователя в команде: UserTeamPrivID (идентификатор привилегии, тип number(10), первичный ключ), UserTeamPrivName (описание привилегии, типа varchar2(10)). Имеющиеся по умолчанию привилегии пользователя в команде: OWNER (создатель команды), STARTED (выполнение задачи начато) и TEAMMATE (участник команды).

Таблица UserTeamPrivTable отвечает за связь таблиц с пользователями, командами и привилегиями пользователей в конкретной команде: PrivUser (идентификатор пользователя, тип number(10), внешний ключ), PrivTeam (идентификатор команды, тип number(10), внешний ключ), Privelegy (идентификатор привилегии пользователя, тип number(10), внешний ключ).

* 1. Процедуры и функции для решения поставленных задач

Для управления данными через приложение администратор использует хранимые процедуры. Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

Написанные мной в ходе разработки курсового проекта процедуры и функции можно разбить на несколько категорий. Их мы рассмотрим ниже.

* + 1. Идентификация пользователей

Во-первых, был написан блок процедур и функций, отвечающих за идентификацию пользователей.

Прежде всего, рассмотрим функцию EncryptionPassword. Ее реализацию можно увидеть на рисунке 2.1. Эта функция позволяет зашифровать пароль пользователя. Она принимает один входной параметр: пароль пользователя, который следует зашифровать. В моем курсовом проекте для шифрования используется 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 «22012003147751322». 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 l\_InVal 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.

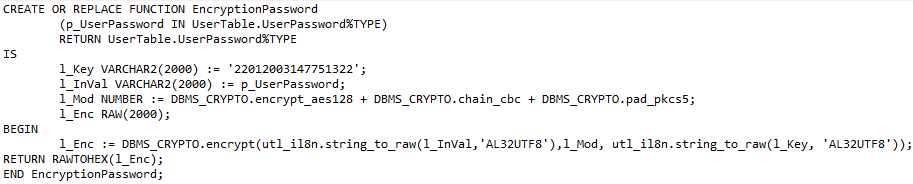


Рисунок 2.1 – Процедура EncryptionPassword

В обратных целях была разработана функция DecryptionPassword. 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. Так же не забываем 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 на рисунке 2.2.

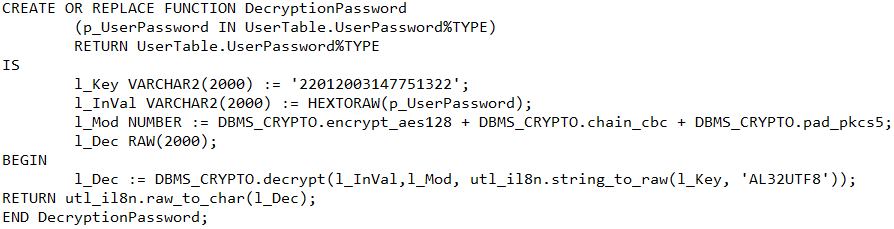


Рисунок 2.2 – Процедура DecryptionPassword

Далее была разработана хранимая процедура RegisterUser, с помощью которой происходит регистрация пользователей в приложении. Эта процедура принимает два параметра: p\_UserLogin (логин пользователя, которого нужно зарегистрировать) и p\_UserPassword (пароль от аккаунта этого пользователя). В случае, если указанный логин уже есть в базе данных, поднимается исключение. Если исключения не произошло, то строка успешно вставляется в таблицу UserTable, зашифровав пароль в процессе.

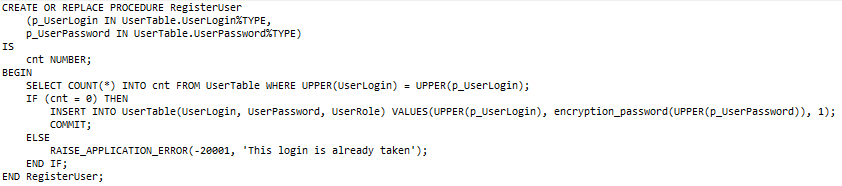


Рисунок 2.3 – Процедура RegisterUser

Для авторизации пользователя в приложении разработана процедура LoginUser. У нее есть два входных параметра: p\_UserLogin (логин пользователя) и p\_UserPassword (пароль пользователя). Если указанная комбинация логина и пароля отсутствует в таблице user\_table, 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: o\_UserID (идентификатор пользователя, прошедшего авторизацию), o\_UserLogin (логин пользователя), o\_UserRole (роль пользователя).

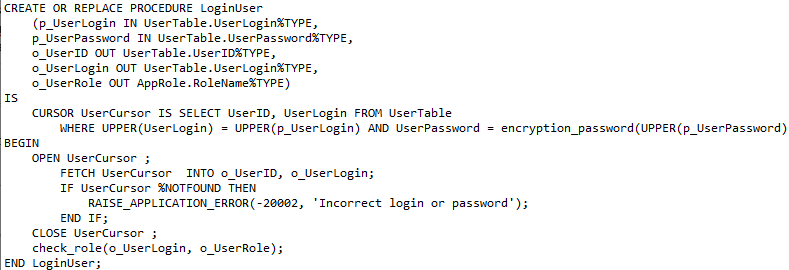


Рисунок 2.4 – Процедура LoginUser

Процедура CheckRole позволяет найти роль пользователя и выгружает её для дальнейших манипуляций. Принимает один входной параметр, а именно: логин пользователя (p\_UserLogin), а также один выходной параметр: название роли пользователя (o\_UserRole).

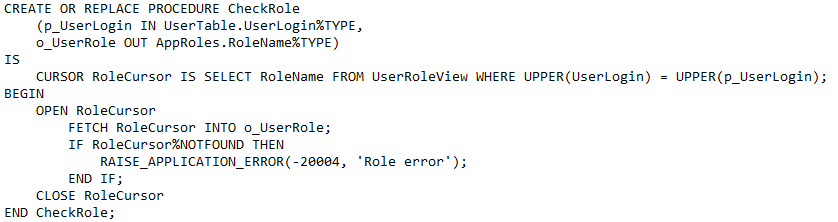


Рисунок 2.5 – Процедура CheckRole

Также были разработаны другие процедуры для взаимодействия и связи пользователя с командами и задачами.

* + 1. Процедуры для манипуляций на странице «Профиль»

Процедура SearchUser позволяет найти информацию о пользователе и выгружает данные о нем из таблицы UserTable. Она используется для отображения данных на странице «Профиль». Процедура принимает один входной параметр, а именно: логин пользователя (p\_UserLogin), а также несколько выходных параметра, а именно: логин пользователя (p\_UserLogin), имя пользователя (p\_UserName), фамилию пользователя (p\_UserLastName), номер телефона пользователя (p\_UserPhoneNumber) и электронную почту пользователя (p\_UserEmail).

Процедура UpdateUserLogin позволяет изменить логин пользователя. Данная процедура принимает два входных параметра: старый логин пользователя (p\_UserLogin) и новый логин пользователя (p\_NewUserLogin), которым следует заменить старый.

Процедура UpdateUserPassword позволяет изменить пароль пользователя. Эта принимает два входных параметра, а именно: старый пароль пользователя (p\_UserPassword) и расшифрованный новый пароль пользователя (p\_NewUserPassword), который заменит в таблице старый и будет зашифрован в процессе.

Процедура UpdateUserEmail позволяет изменить электронную почту пользователя. Эта принимает два входных параметра, а именно: старую электронную почту пользователя (p\_UserEmail) и новую электронную почту пользователя (p\_NewUserEmail), которая заменит в таблице старое значение.

Процедура UpdateUserPhoneNumber позволяет изменить номер телефона пользователя. Эта принимает два входных параметра, а именно: старый номер телефона пользователя (p\_UserPhoneNumber) и новый номер телефона пользователя (p\_NewPhoneNumber), который заменит в таблице старое значение.

Процедура UpdateUserName позволяет изменить имя пользователя. Эта принимает два входных параметра, а именно: старое имя пользователя (p\_UserName) и новое имя пользователя (p\_NewUserName), которое заменит в таблице старое значение.

Процедура UpdateUserLastName позволяет изменить фамилию пользователя. Эта принимает два входных параметра, а именно: старую фамилию пользователя (p\_UserLastName) и новую фамилию пользователя (p\_NewUserLastName), которая заменит в таблице старое значение.

Процедура UpdateUserIcon позволяет изменить фотографию профиля пользователя. Эта принимает два входных параметра, а именно: старую фотографию профиля пользователя (p\_UserIcon) и новую фотографию профиля пользователя (p\_NewUserIcon), которая заменит в таблице старое значение.

* + 1. Процедуры для добавления данных в таблицы

Процедура create\_artist служит для добавления нового артиста в базу данных. Данная процедура принимает один параметр: p\_TeamName (название команды, которое будет добавлено в базу данных). Поскольку все процедуры, описанные в данном пункте, аналогичны по своему алгоритму, имеет смысл показать в качестве примера только одну их них. Таким образом, на рисунке 2.3 можно увидеть реализацию функции CreateTeam.

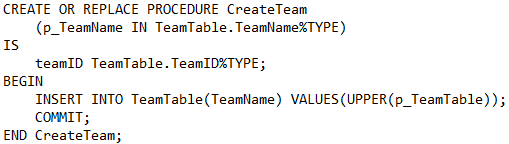


Рисунок 2.3 – Процедура CreateTeam

Далее была разработана процедура для добавления нового альбома в базу данных. Эта процедура принимает три параметра: p\_artist\_name (имя исполнителя, которому принадлежит создаваемый альбом), p\_album\_name (название создаваемого альбома) и p\_album\_released (год выпуска альбома). Так как в хранимых процедурах нельзя передавать данные объемом более 32 Кб, изображение, служащее обложкой альбома, передается отдельно через операцию INSERT.

Процедура create\_song служит для добавления новой песни в базу данных. Эта процедура принимает три параметра: p\_artist\_name (имя исполнителя, которому принадлежит создаваемая песня), p\_album\_name (название альбома, к которому будет относится песня) и p\_song\_name (название добавляемой песни). Аудиофайл песни передается отдельно через операцию INSERT.

* + 1. Процедуры для изменения данных в таблицах

В этом блоке будут описаны процедуры, с помощью которых администратором изменяются данные в таблицах.

Процедура update\_artist отвечает за замену старого имени исполнителя на новое. Она соответственно принимает эти имена как параметры: p\_old\_artist и p\_new\_artist. Реализацию этой процедуры будем использовать как пример функций изменения. Она изображена на рисунке 2.4.

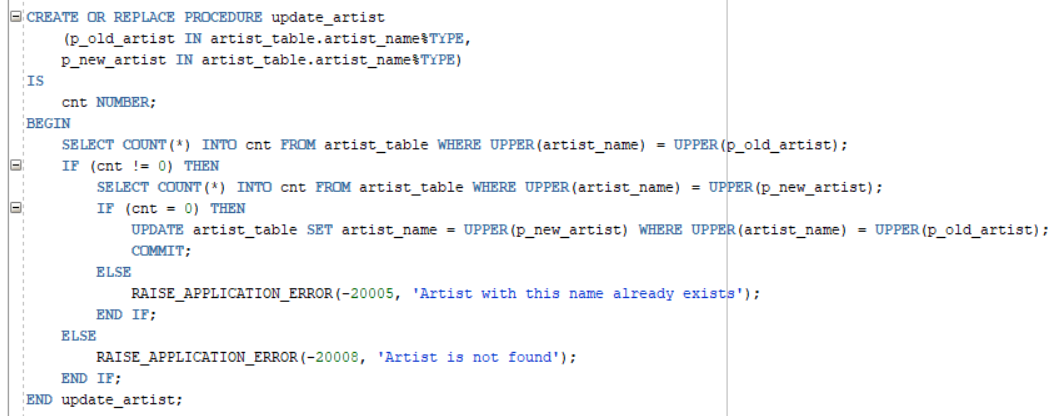


Рисунок 2.4 – Процедура update\_artist

Процедура update\_album\_name позволяет обновить название альбома. Она принимает два параметра: p\_ablum\_id (идентификатор альбома) и p\_new\_name (имя альбома, которым будет заменено старое).

Процедура update\_album\_year позволяет обновить год выпуска альбома. Она принимает два параметра: p\_ablum\_id (идентификатор альбома) и p\_new\_year (год выпуска альбома, которым будет заменен старый).

Процедура update\_song\_name позволяет обновить название уже существующей песни. Она принимает два параметра: p\_song\_id (идентификатор песни) и p\_new\_name (новое название для песни).

* + 1. Процедуры для удаления данных из таблиц

В этом блоке будут описаны процедуры, с помощью которых администратором удаляются данных из таблиц. Поскольку таблицы создавались с параметром ON DELETE CASCADE, удаление значений значительно упрощается. Когда данные в родительском ключе удаляются, все строки в дочерней таблице, которые зависят от значений удаленного родительского ключа, также удаляются.

Процедура delete\_artist отвечает за удаление исполнителя из базы данных. Она принимает один параметр: p\_id (идентификатор удаляемого исполнителя). Реализацию этой процедуры будем использовать как пример функций удаления. Она изображена на рисунке 2.5. При удалении исполнителя удалятся все связанные с ним альбомы, песни и экземпляры песен, сохраненных в плейлисты пользователей.

Процедура delete\_album отвечает за удаление альбома из базы данных. Она принимает один параметр: p\_id (идентификатор удаляемого альбома). При удалении альбома удалятся все связанные с ним песни и экземпляры песен, сохраненных в плейлисты пользователей.

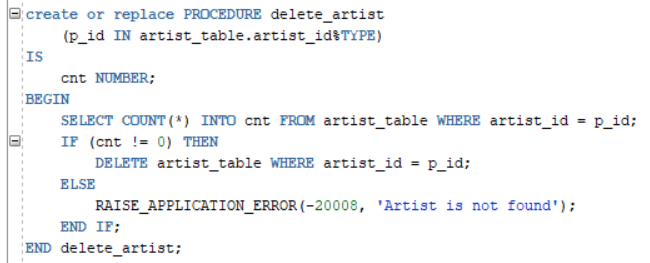


Рисунок 2.5 – Процедура delete\_artist

Процедура delete\_song отвечает за удаление песни из базы данных. Она принимает один параметр: p\_id (идентификатор удаляемой песни). При удалении песни удалятся все связанные с ней экземпляры песен, сохраненных в плейлисты пользователей.

Процедура remove\_song отвечает за удаление песни из плейлиста конкретного пользователя. Она принимает два параметра: p\_user\_id (идентификатор пользователя) и p\_song\_id (идентификатор удаляемой из плейлиста песни).

С помощью процедуры delete\_user пользователь может удалить свой аккаунт. Процедура принимает один параметр: p\_login (логин пользователя). При этом удалятся все данные о составе его плейлиста.

* + 1. Процедура для добавления 100 000 строк

С помощью процедуры InsertMoreUsers было зарегистрировано 100 000 пользователей. Она изображена на рисунке 2.6.

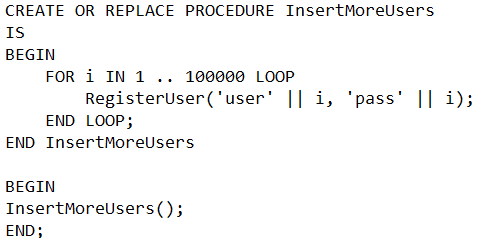


Рисунок 2.6 – Процедура InsertMoreUsers

Поскольку процесс получения администратором информации о всех пользователях занимал около двух минут, было решено отображать их в окне администратора по 20.

* + 1. Процедура для экспорта и импорта таблиц в XML

Для реализации обоих процедур потребовалось создать директорий с желаемым расположением при подключении под пользователем SYS к инстансу и выдать пользователю, владеющему базой данных, гран на чтение и запись из этого директория. Это изображено на рисунке 2.7.

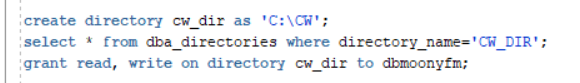


Рисунок 2.7 – Создание директория CW\_DIR

В рамках задания нужно было разработать процедуру, экспортирующую данные из таблицы, содержащей 100 000 строк, в XML документ. С этой целью разработана процедура users\_export. Она изображена на рисунке 2.8.

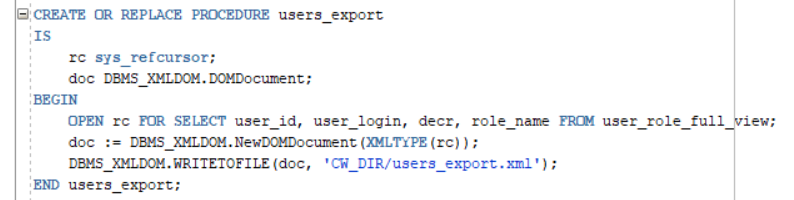


Рисунок 2.8 – Процедура users\_export

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 artist\_table. Процедура artist\_import изображена на рисунке 2.9.

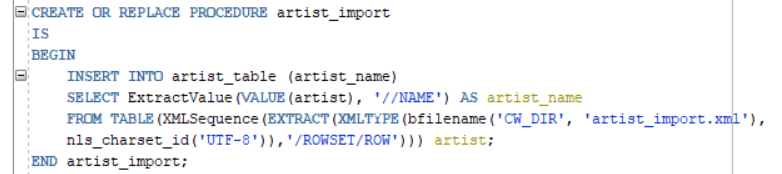


Рисунок 2.9 – Процедура artist\_import

* 1. Представления базы данных

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

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

Представление artist\_album\_view объединяет таблицы artist\_table и album\_table и возвращает следующие столбцы: album\_table.album\_id, artist\_table.artist\_name, album\_table.album\_name, album\_table.album\_released, album\_table.album\_blob. Оно используется для отображения плиток с альбомами.

Представление artist\_album\_song\_view объединяет таблицы artist\_table, album\_table и song\_table и возвращает следующие столбцы: song\_table.song\_id, artist\_table.artist\_name, album\_table.album\_name, song\_table.song\_name, album\_table.album\_released, album\_table.album\_blob, song\_table.song\_blob. Оно содержит наиболее полную информацию о каждой из композиций.

Последнее представление, artist\_album\_song\_user\_view, дополняет предыдущее представление, дополнительно объединяя его с таблицей user\_table. Это используется для корректного отображения плиток с песнями во вкладке «Плейлист».

* 1. Пользователь базы данных

В разработанной мной базе данных присутствует один пользователь, который является хозяином схемы – DBMoonyFM. Он имеет грант на подключение к базе данных, грант на выборку, добавление, изменение и удаление данных из всех таблиц, описанных в пункте 2.1, и из всех представлений, описанных в пункте 2.3. Он может создавать таблицы, процедуры, представления и сессии. Также ему выдан грант на выполнение процедур и функций стандартного пакета dbms\_crypto.

1. Обоснование технических приемов программирования

При разработке программного средстве в качестве программной платформы был выбран язык программирования C#. Проект разработан с использованием .NET Framework 4.7.2. Для взаимодействия с сервером Oracle 12c нужно было с помощью диспетчера пакетов NuGet скачать пакет Oracle.ManagedDataAccess. Интерфейс представлен с использованием Windows Presentation Foundation (WPF). Все эти средства используются для реализации поставленных задач.

Пакет Oracle.ManagedDataAccess предоставляет драйвер, позволяющий подключаться к серверу. На рисунке 3.1. изображена конфигурация подключения к серверу.

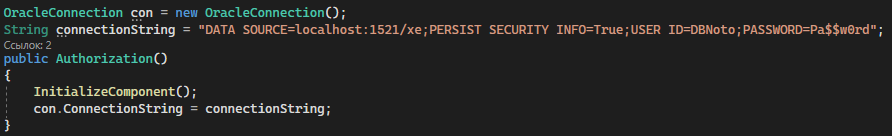


Рисунок 3.1 – Строка подключения

Большая часть взаимодействия с базой данных происходит через хранимые процедуры. На рисунке 3.2 можно увидеть пример того, как в C# вызываются хранимые на сервере Oracle процедуры. В качестве примера изображена процедура авторизации пользователя в приложении

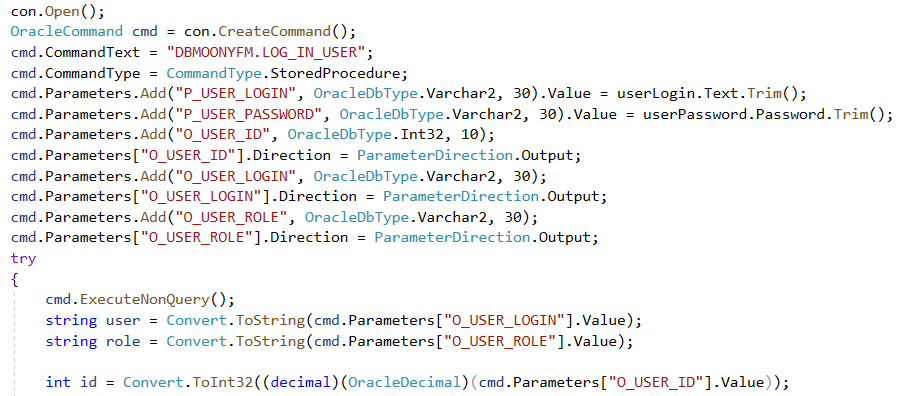


Рисунок 3.2 – Вызов процедуры авторизации

В случаях, когда используются непосредственно SELECT запросы, используются конструкции подготовленных заявлений. Они позволяют передать в запрос параметры, не допустив SQL-инъекций.

Рассмотрим этот сценарий на примере запроса, используемого для вывода плиток с песнями. В этом случае в качестве параметров :search и :search2 передаются соответственно имя исполнителя и название альбома. Данную конструкцию можно увидеть на рисунке 3.3.



Рисунок 3.3 – Защищенный от SQL-инъекций SELECT-запрос

Сценарий с командой UPDATE будет рассмотрен в разделе с технологией мультимедийных типов данных.

1. Описание технологий

В данном разделе будут описаны технологии, использованные в процессе разработки приложения.

* 1. Мультимедийные типы данных

Для работы с данными большого объема СУБД Oracle предоставляет типы данных BLOB, CLOB, NCLOB и BFILE. Здесь LOB означает Large OBject, или большой объект, и далее по тексту термины LOB и "большой объект" взаимозаменяемы. 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 музыкальных альбомов, а также аудиофайлы, без которых все приложение потеряло бы смысл. Речь идет о таблицах album\_table и song\_table, а конкретно об их столбцах album\_table. Для хранения в поле изображений и аудиофайлов в СУБД 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.

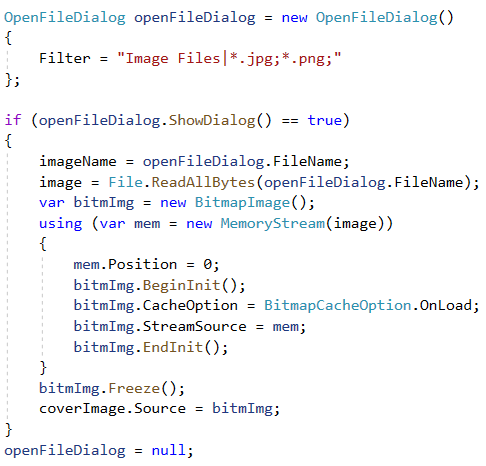


Рисунок 4.1 – Загрузка изображения и разбиение его на массив битов

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 нового альбома происходит в два этапа. Сначала вызывается хранимая процедура create\_album, 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 на рисунке 3.1. Для удобства администратора загруженное изображение дополнительно отображается в окне, чтобы можно было убедиться в правильности выбора.

После превращения изображения в массив битов дело остается за малым, а именно: 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 album\_blob. Это можно увидеть на рисунке 4.2.



Рисунок 4.2 – Загрузка изображения и разбиение его на массив битов

На этом этапе изображение успешно загружено в базу данных. Также в своем приложении я обращаюсь к этому изображению для отображения плиток альбомов и песен с соответствующей обложкой.

Процесс получения значения поля album\_blob изображен на рисунке 4.3. Он заключается в проcтейшем SELECT-запросе к таблице и чтении результата в качестве массива битов. После этого с помощь. Объекти BitmapImage() мы превращаем полученный массив битов в битовую карту. То есть, непосредственно в изображения.

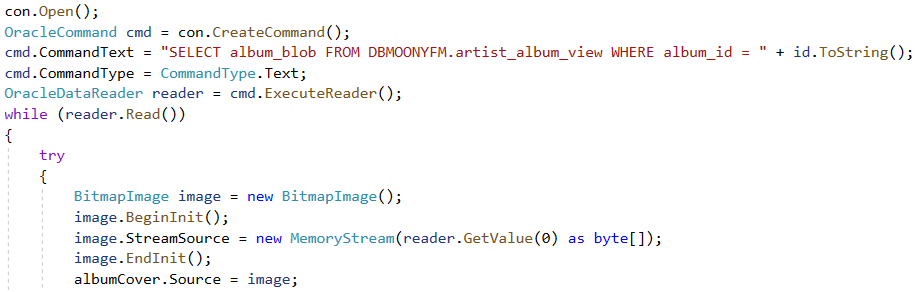


Рисунок 4.3 – Получение изображения из базы данных

Далее мы рассмотрим аналогичный процесс, но с аудио файлом. В рамках окна AddNewSong.xaml администратор заполняет поля для добавления новой песни. По причинам, описанным выше, процесс загрузки песни тоже разбит на 2 этапа. Как и в случае с изображением, аудио файл загружается с помощью диалогового окна. Процесс загрузки аудио файла в базу данных идентичен случаю с изображениями, поэтому перейдем сразу к выгрузке аудио файла из базы.

Пользователь с любой ролью может прослушивать песни в приложении. Это происходит в рамках окна Player.xaml. Логика прослушивания песни такова, при загрузке указанного окна отправляется SELECT-запрос к базе данным, в результате которого мы получаем массив битов песни с переданным идентификатором. Чтобы было возможно запустить аудио файл, 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 «CW\_DB\_MoonyFM\CW\_DB\_MoonyFM\bin\Debug».

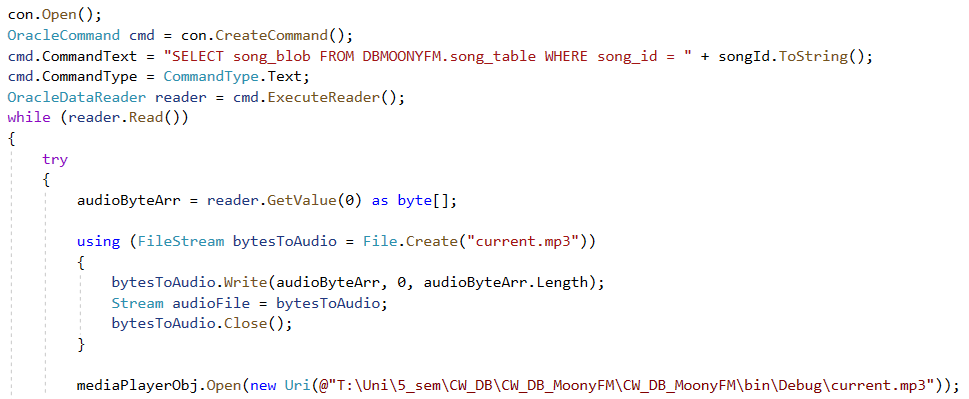


Рисунок 4.4 – Запись аудио в файл-буфер

Далее, чтобы запустить полученную из базы данных песню, мы инициализируем объект типа MediaPlayer, передав ему путь к скачанному файлу, и с этого момента можем управлять нашим аудио файлом с помощью кнопок в окне. Фрагмент кода изображен на рисунке 4.4.

* 1. Средства диагностики

Несмотря на то, что Oracle предоставляет очень богатый функционал для разработчика, его отличительной чертой является также ограниченность в бесплатном вспомогательном программном обеспечении, в отличие от MS SQL. Из очень ограниченного круга средств мой выбор упал на Database Performance Analyzer, разработанный компанией SolarWinds, поскольку с ним предоставляется 14-тидневный период оценки ПО, в рамках которого я была ознакомлена с его функционалом.

Программное обеспечение SolarWinds Database Performance Analyzer – это инструмент мониторинга производительности SQL-сервера, поддерживающий также и сервер Oracle 12c, помогающий выявлять наиболее опасные для конечных пользователей проблемы.

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. Интерфейс SolarWinds Database Performance Analyzer изображен на рисунке 4.5.

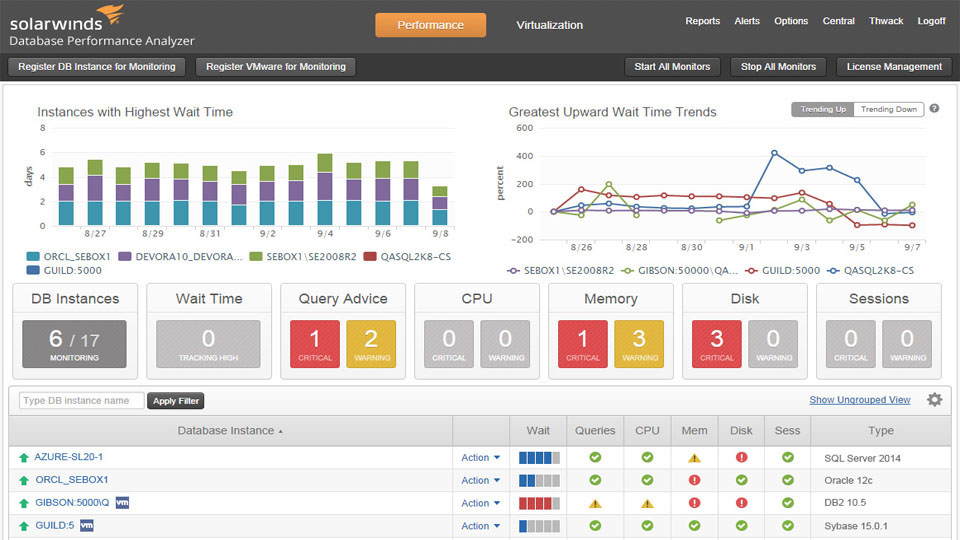


Рисунок 4.5 – Интерфейс SolarWinds DPA

В горизонтальном блоке плиток по центру рисунка 4.5 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 4.6, можно отслеживать статус ошибок и неисправностей, создавать группы оповещений, задать случаи, когда необходимо оповещать о неисправности по электронной почте.

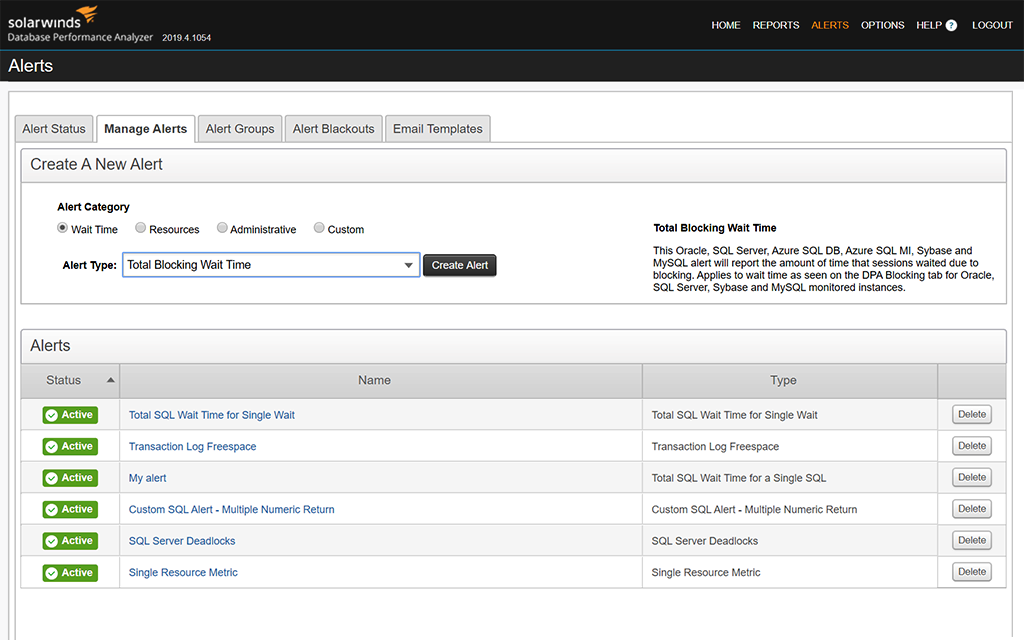


Рисунок 4.6 – Вкладка Alerts

По итогам 14-тидневного периода пользования мною были отмечены следующие преимущества данного средства диагностики. Это ПО оказалось крайне полезным, поскольку оно предоставляет полную картину. За несколько секунд программа показывает текущий статус сеанса, так что администраторы баз данных могут мгновенно реагировать на проблемы производительности.

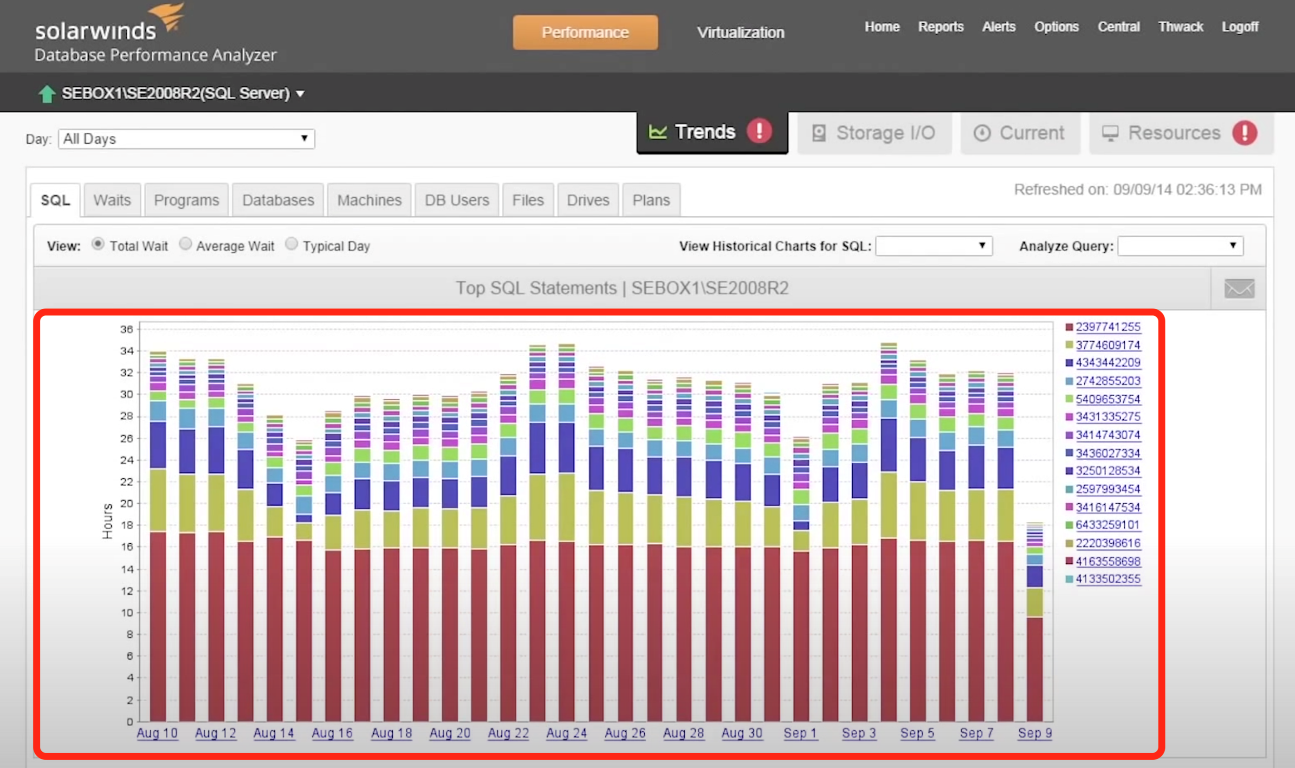


Рисунок 4.7 – Диаграмма времени ожидания запросов

На первой вкладке подробной информации будет находиться временная диаграмма, отображающая время ожидания запросов, как показано на рисунке 4.7.

Также, как оказалось, 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.

1. Руководство пользователя, тестирование

Разработанная в рамках курсового проекта музыкальная площадка поддерживает два вида ролей у пользователя: администратор и рядовой пользователь. Роль администратора дополняет функционал обычного пользователя, поэтому будет уместно в рамках руководства осветить возможности аккаунта администратора.

При запуске приложения открывается окно авторизации (рисунок 5.1), где зарегистрированному пользователю следует ввести свой логин и пароль.

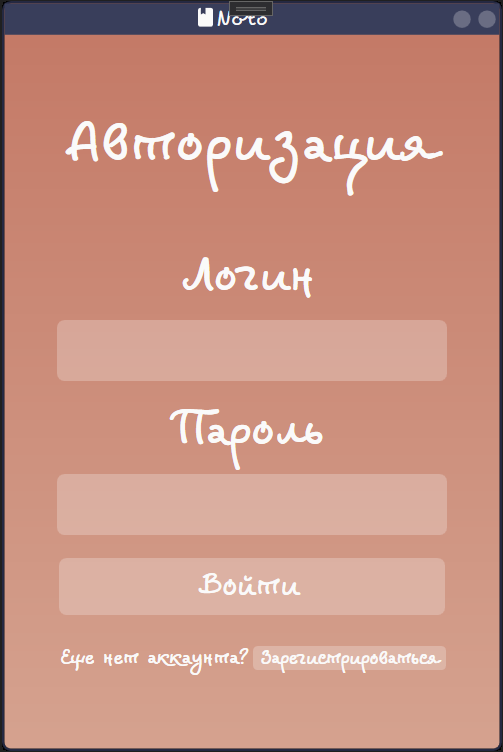


Рисунок 5.1 – Окно авторизации

При неправильном вводе пароля пользователю выводится сообщение о неправильном пароле (рисунок 5.2).

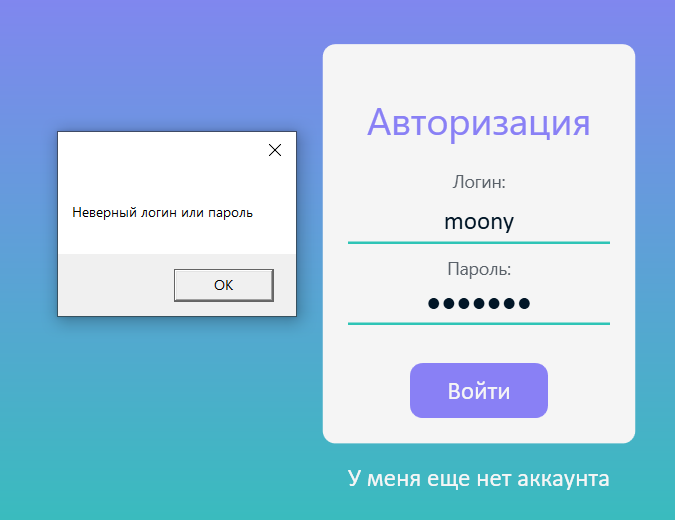


Рисунок 5.2 – Окно авторизации, неверный логин или пароль

При успешной авторизации пользователь будет перенаправлен на главную страницу приложения.

Неавторизованный пользователь не может зайти на главную страницу, пока не авторизуется. Для этого следует нажать на кнопку «У меня еще нет аккаунта» в окне авторизации. Окно авторизации изображено на рисунке 5.3.

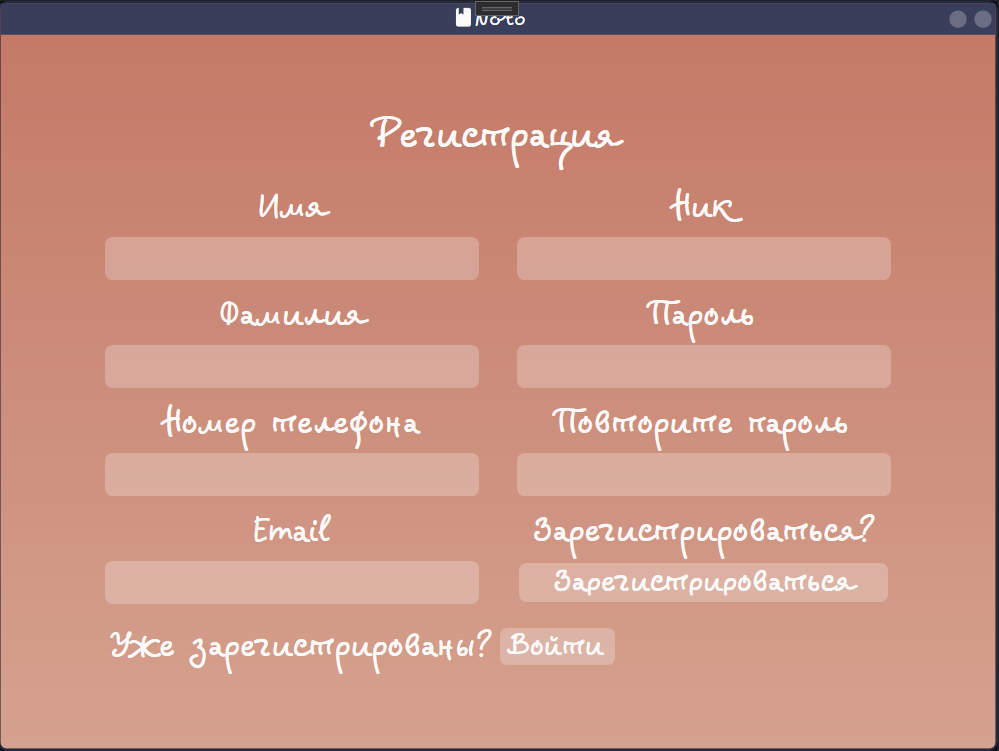


Рисунок 5.3 – Окно регистрации

Если в этом окне ввести логин пользователя, уже существующего в базе данных, пользователь увидит следующую ошибку (рисунок 5.4).

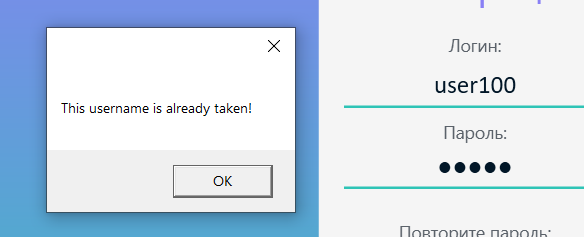


Рисунок 5.4 – Ввод занятого логина

Также в рамках данного окна обрабатывается ошибка при введении несовпадающих паролей.

Допустим, что пользователь успешно зарегистрирован. Далее, пройдя авторизацию под аккаунтом администратора, мы попадем на домашнюю страницу приложения (рисунок 5.5).

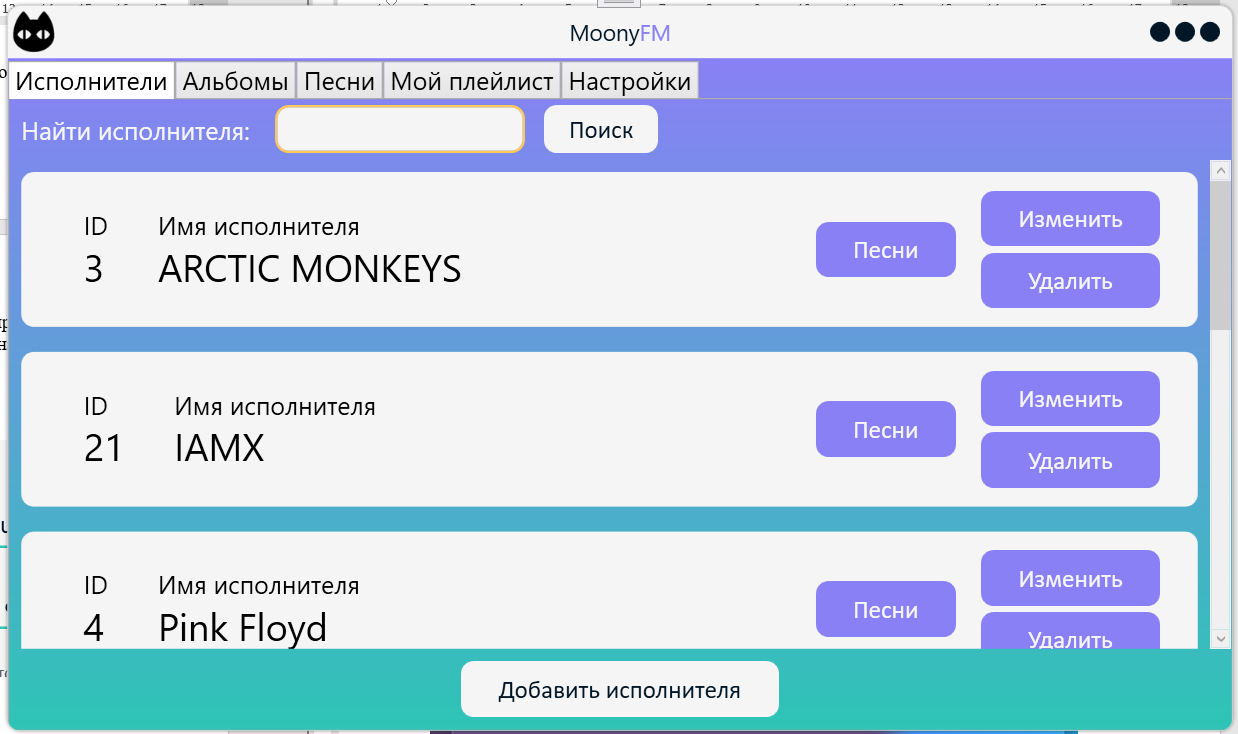


Рисунок 5.5 – Домашняя страница приложения (администратор)

Смысл аккаунта обычного пользователя сводится к поиску песен, прослушиванию их и добавлению в свой плейлист.

Администратор же волен делать то же, что и пользователь, с расширенным функционалом: администратор может добавлять, изменять и удалять исполнителей, альбомы и песни, а также имеет доступ к окну администратора (через меню «Настройки») со списком пользователей и кнопками для запуска импорта и экспорта XML.

Попав на домашнюю страницу, по умолчанию открывается вкладка «Исполнители». Если обратить внимание на меню, в нем также имеются вкладки «Альбомы», «Песни», «Мой плейлист» и «Настройки».

Вкладка «Исполнители» содержит список исполнителей, состоящий из плиток с кнопками. На этой вкладке можно осуществлять поиск по имени исполнителя, воспользовавшись текстовый полем вверху.

Любой пользователь может нажать на кнопку «Песни» напротив имени исполнителя, и тогда попадет на окно со списком всех имеющихся песен этого исполнителя. Это окно можно увидеть на рисунке 5.6.

Вкладка «Альбомы» содержит список альбомов, состоящий из плиток с кнопками. На этой вкладке можно осуществлять поиск по названию альбома, воспользовавшись текстовый полем вверху. Вкладка «Альбомы» изображена на рисунке 5.7.

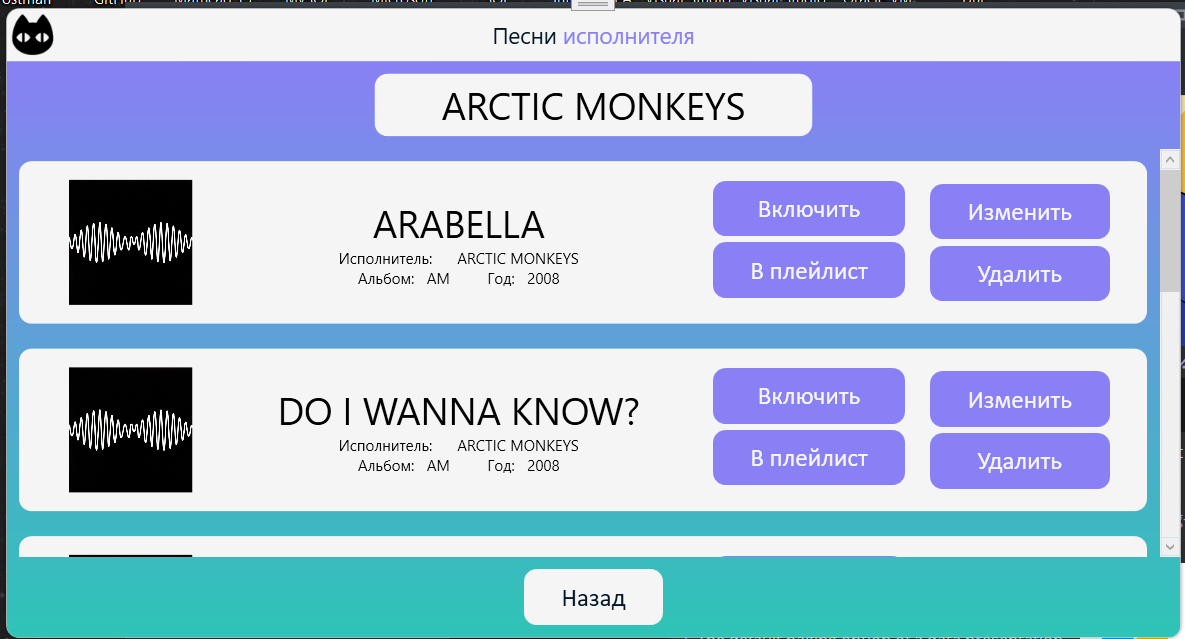


Рисунок 5.6 – Список песен исполнителя (администратор)

Плитки песен являются универсальными и имеют одинаковый функционал во всех точках приложения.

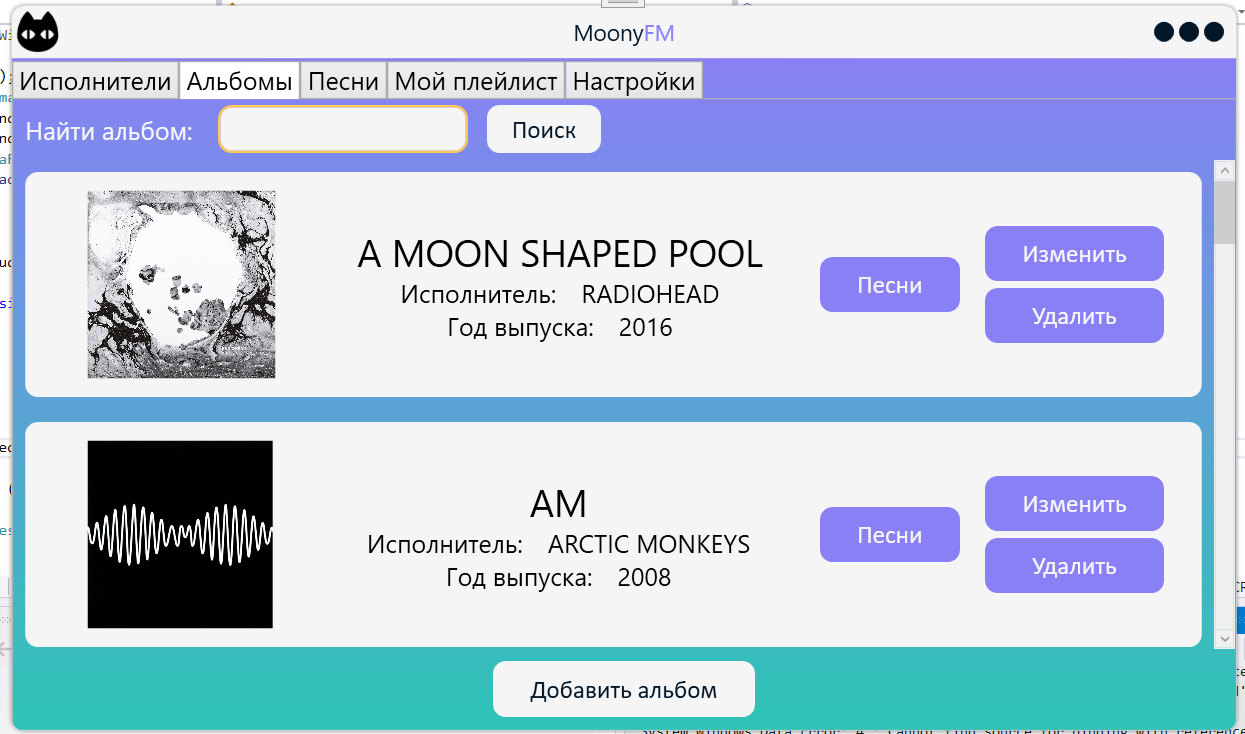


Рисунок 5.7 – Вкладка «Альбомы» (администратор)

Любой пользователь может нажать на кнопку «Песни» напротив названия альбома, и тогда попадет на окно со списком всех песен на этом альбоме. Это окно можно увидеть на рисунке 5.8.

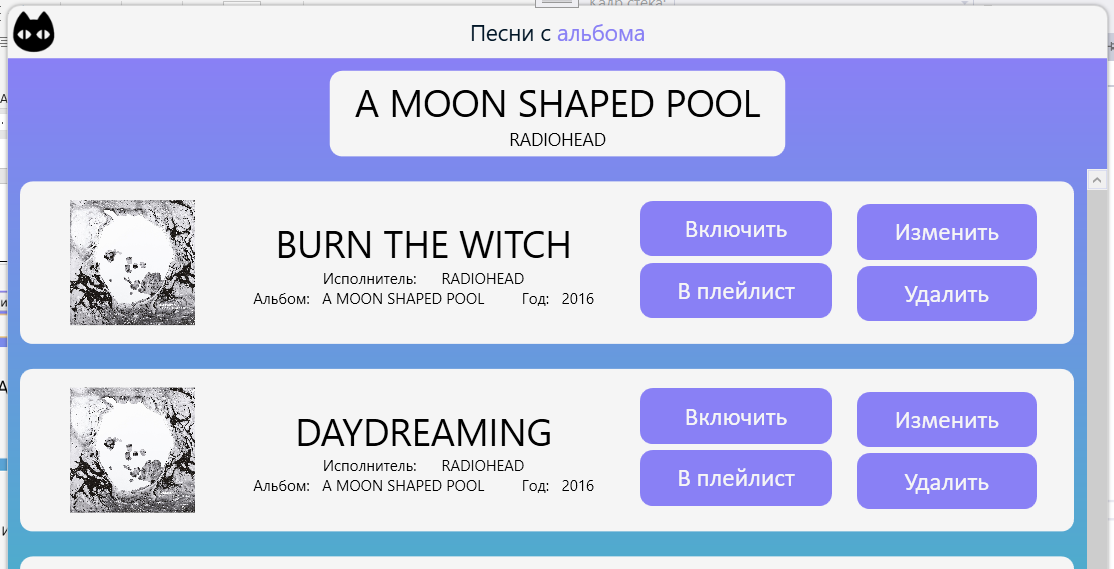


Рисунок 5.8 – Список песен с определенного альбома (администратор)

Вкладка «Песни» содержит всех песен, состоящий из плиток с кнопками. На этой вкладке можно осуществлять поиск по названию песни, воспользовавшись текстовый полем вверху.

Любой пользователь может нажать на кнопку «Включить» внутри плитки песни, и тогда попадет на окно с плеером.

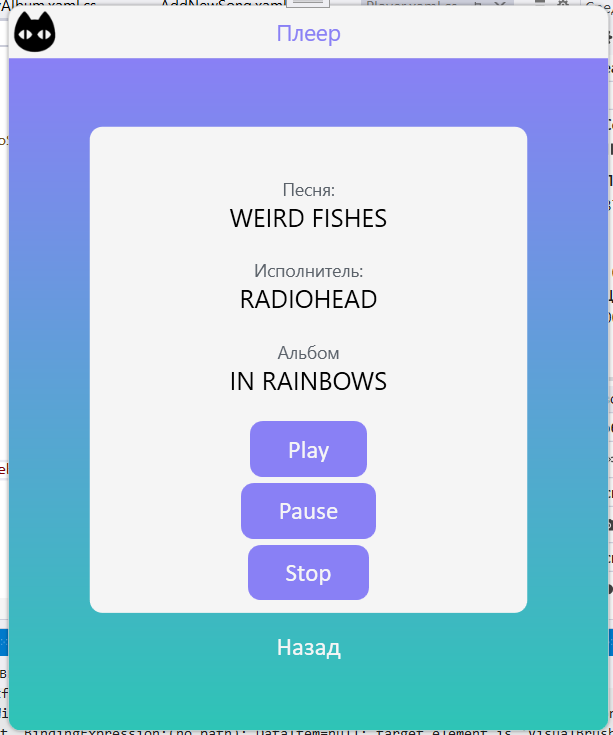


Рисунок 5.9 – Плеер

Плеер содержит информацию о текущей песне и три кнопки для управления песней. По нажатию на кнопку «Play» песня запустится. Это окно можно увидеть на рисунке 5.9.

По нажатию на кнопку «В плейлист» на плитке песни в любом месте приложения эта песня добавится в ваш плейлист (вкладка «Мой плейлист»). Если эта песня уже есть в вашем плейлисте, вы увидите ошибку с рисунка 5.10.

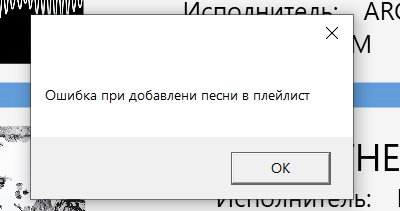


Рисунок 5.10 – Ошибка: песня уже в вашем плейлисте

На вкладке «Мой плейлист» плитки песен несколько отличаются: на них отсутствуют кнопки администратора и вместо кнопки «В плейлист» на них находится кнопка «Убрать из плейлиста». Нажав на эту кнопку, вы сможете удалить песню из вашего плейлиста.

Содержимое вкладки «Мой плейлист» можно увидеть на рисунке 5.11.

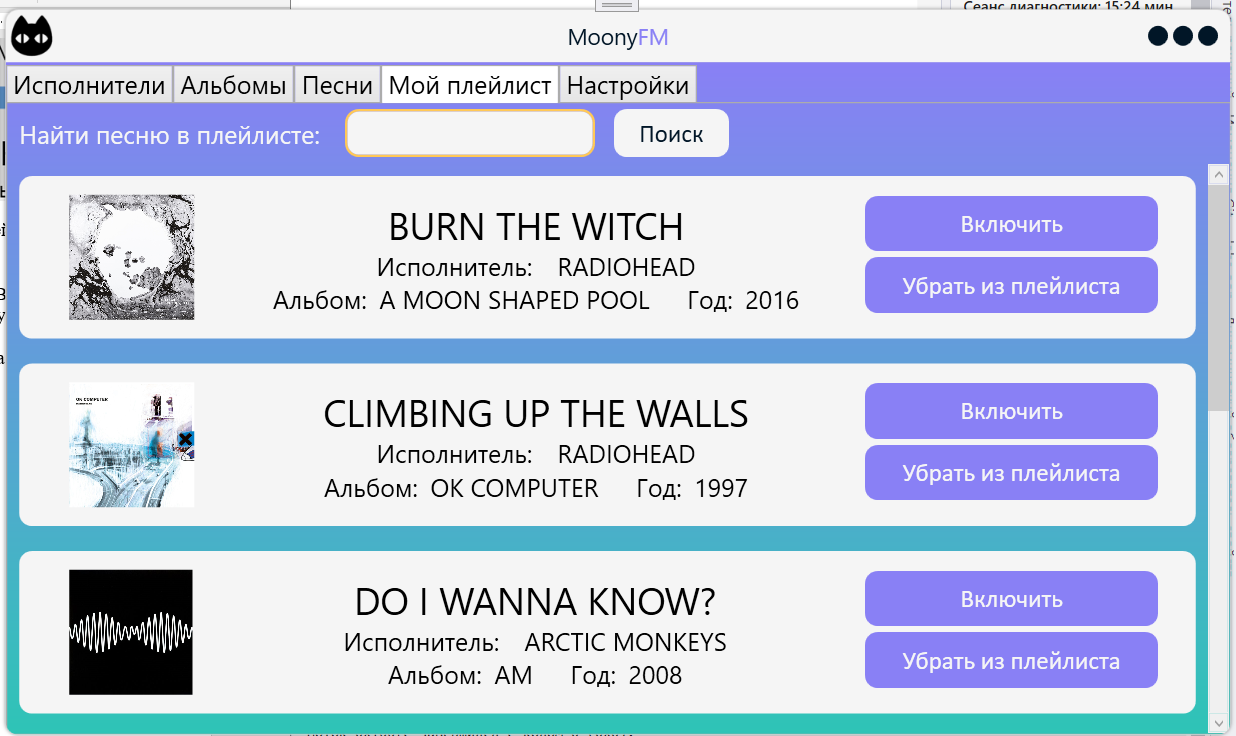


Рисунок 5.11 – Вкладка «Мой плейлист»

Перейдя на вкладку «Настройки», любой пользователь может выйти из аккаунта, изменить свой логин или свой пароль. Если введенный новый логин совпадает с уже имеющимся в базе, пользователь увидит ошибку с рисунка 5.12.

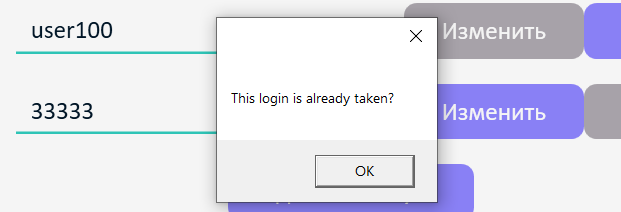


Рисунок 5.12 – Ошибка при изменении логина

Любой пользователь кроме администратора может удалить свой аккаунт по желанию. При этом надо иметь в виду, плейлист, бережно составленный этим пользователем, будет утерян.

На вкладке «Настройки» администратору также доступна кнопка «Открыть окно администратора». По нажатию на эту кнопку откроется соответствующее окно.

Это окно имеет две вкладки: «Пользователи» и «Другое». На вкладке пользователей 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, стоит нажать на кнопку «Показать следующих 20». Также на этой странице доступен поиск по логину пользователя. Эту вкладку можно увидеть на рисунке 5.13.

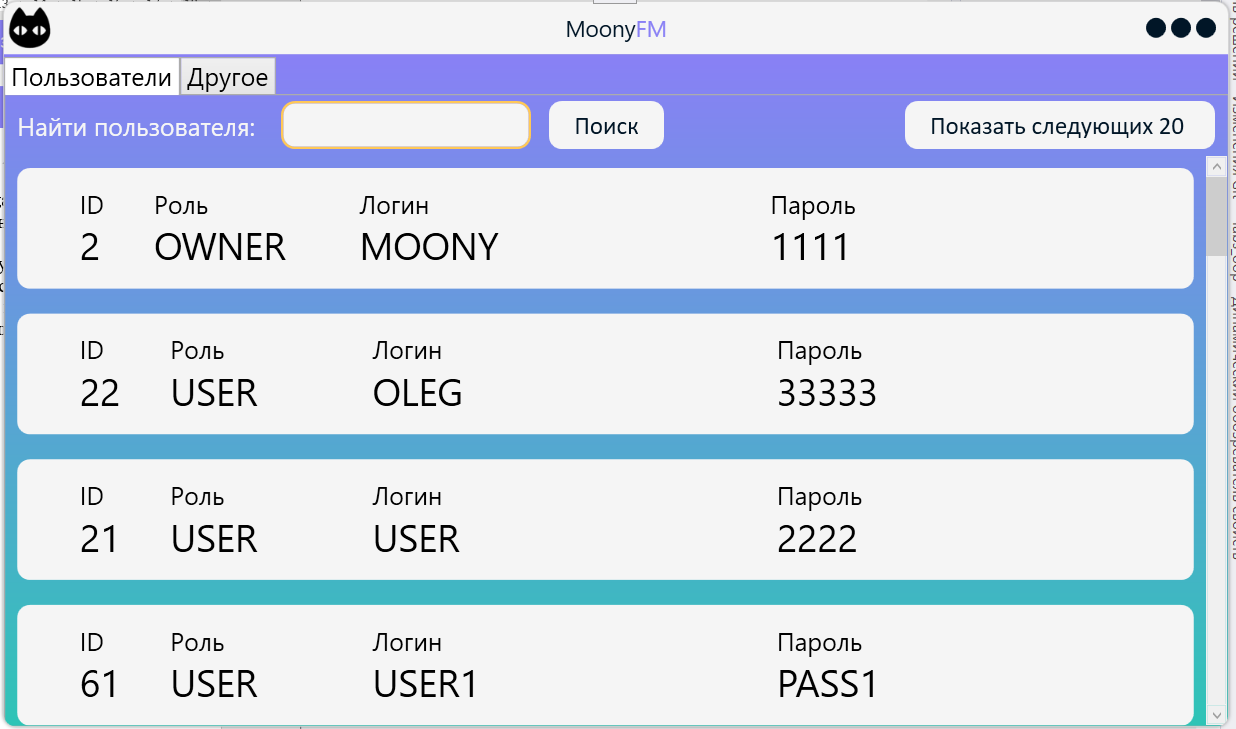


Рисунок 5.13 – Список пользователей (администратор)

Вкладка «Другое» содержит две кнопки. По нажатию на кнопку «Export to XML» запустится процедура users\_export, экспортирующая полный список пользователей в файл «C:\CW\users\_export.xml».

После выполнения процедуры users\_export вторая кнопка («Import from XML») станет доступной для нажатия. По нажатию на эту кнопку запустится процедура artist\_import, импортирующая в базу данных исполнителей, описанных в ней.

Вкладка «Другое» изображена на рисунке 5.14.

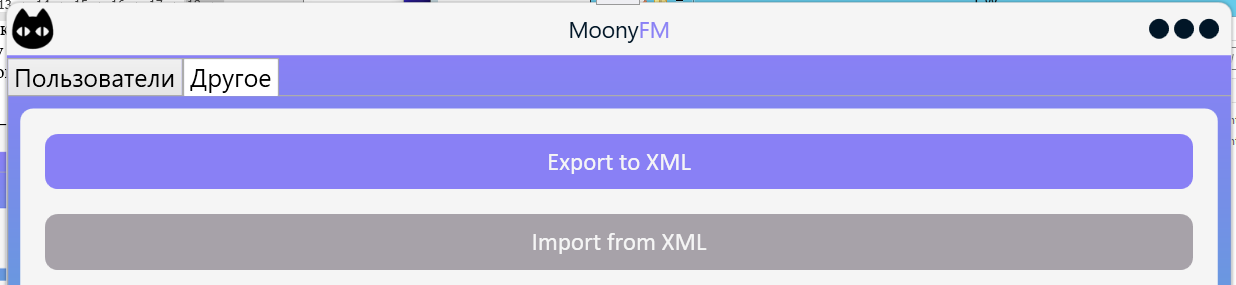


Рисунок 5.14 – Вкладка «Другое»

Будучи администратором, можно создавать исполнителя по нажатию на кнопку «Добавить исполнителя» на вкладке «Исполнители». Если попытаться добавить исполнителя с именем, которое уже есть в базе, получим ошибку, которую видно на рисунке 5.15.

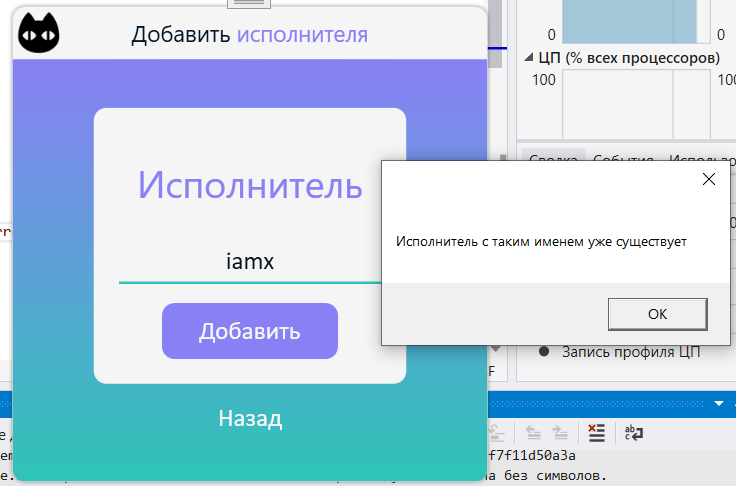


Рисунок 5.15 – Вкладка «Другое»

Также по нажатию на кнопку «Изменить» на плитке исполнителя можно изменить имя данного исполнителя. Если попытаться изменить имя на уже существующее, то получим ошибку, аналогичную изображенной на рисунке 5.15.

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

Нажав на кнопку «Добавить альбом» на вкладке «Альбомы», можно добавить новый альбом для существующего исполнителя. Если нажать «Добавить» в появившемся окне, не заполнив все поля, получим ошибку с рисунка 5.16. Аналогичная ошибка обрабатывается во всех остальных окнах. Также, если попробовать создать уже существующий альбом, мы также получим ошибку.

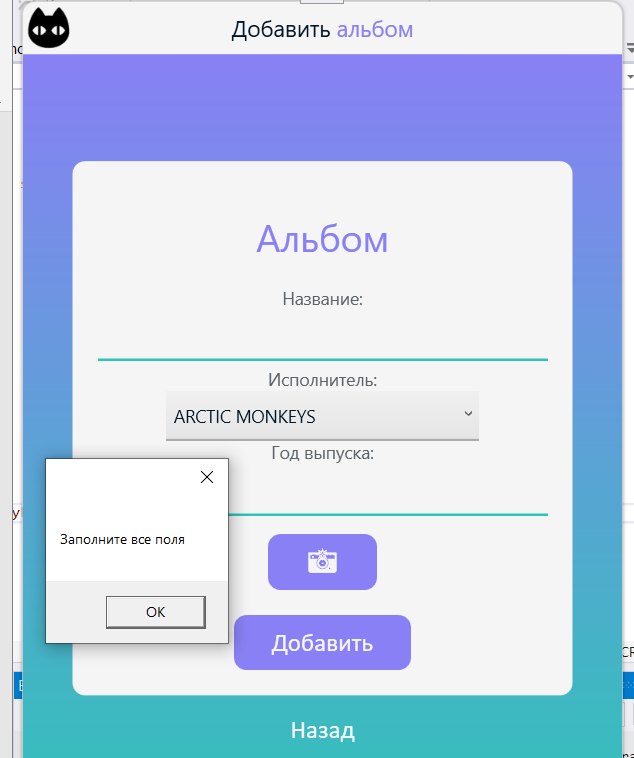


Рисунок 5.16 – Ошибка при добавлении альбома

Также по нажатию на кнопку «Изменить» на плитке альбома можно изменить название данного альбома и год его выпуска. Если попытаться изменить имя на уже существующее, то получим ошибку, аналогичную изображенной на рисунке 5.15.

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

Нажав на кнопку «Добавить песню» на вкладке «Песни», можно добавить новую песню для существующего альбома. Если попробовать прикрепить в качестве аудиофайла файл с расширением, отличным от \*.MP3, появится окно с ошибкой, изображенной на рисунке 5.17

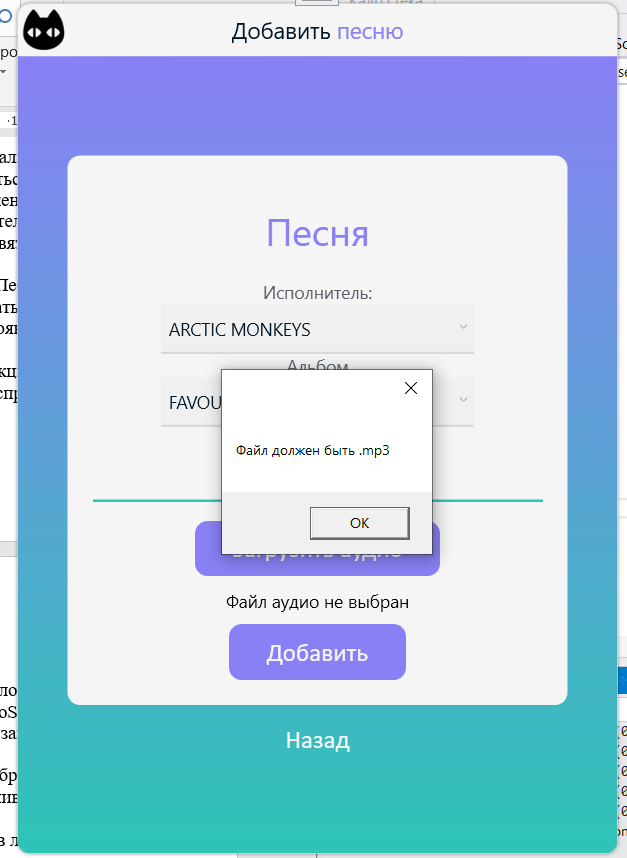


Рисунок 5.17 – Ошибка при добавлении альбома

В проекте был протестирован весь базовый функционал пользователя и администратора приложения. Все компоненты работают исправно.

Заключение

Результатом выполнения курсового проекта стало .NET приложение с использованием технологии 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.

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

* регистрация и авторизация пользователей музыкальной площадки с учётом выданных им ролей;
* пароли при регистрации шифруются;
* мультимедийные типы данных, корректно записывающиеся в базу данных и затем воспроизводящиеся;
* в процессе разработки производительность была проверена средством диагностики SolarWinds DPA;
* поиск песен по исполнителю, альбому и названию самой песни;
* группировка песен по исполнителям, альбомам;
* возможность каждому пользователю сохранять понравившиеся песни в свой плейлист;
* добавление исполнителей администратором;
* добавление альбомов администратором;
* добавление песен 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 на 100 000 записей (таблица пользователей).

Проект не может окончательно претендовать на полноценный коммерческий продукт, поскольку для демонстрации технологии и работы с базой данных были опущены некоторые атрибуты современных стриминговых площадок, 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 литературы

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

2. Oracle-dba.ru [Электронный ресурс] / Режим доступа: https://oracle-dba.ru – Дата доступа: 27.10.2021.

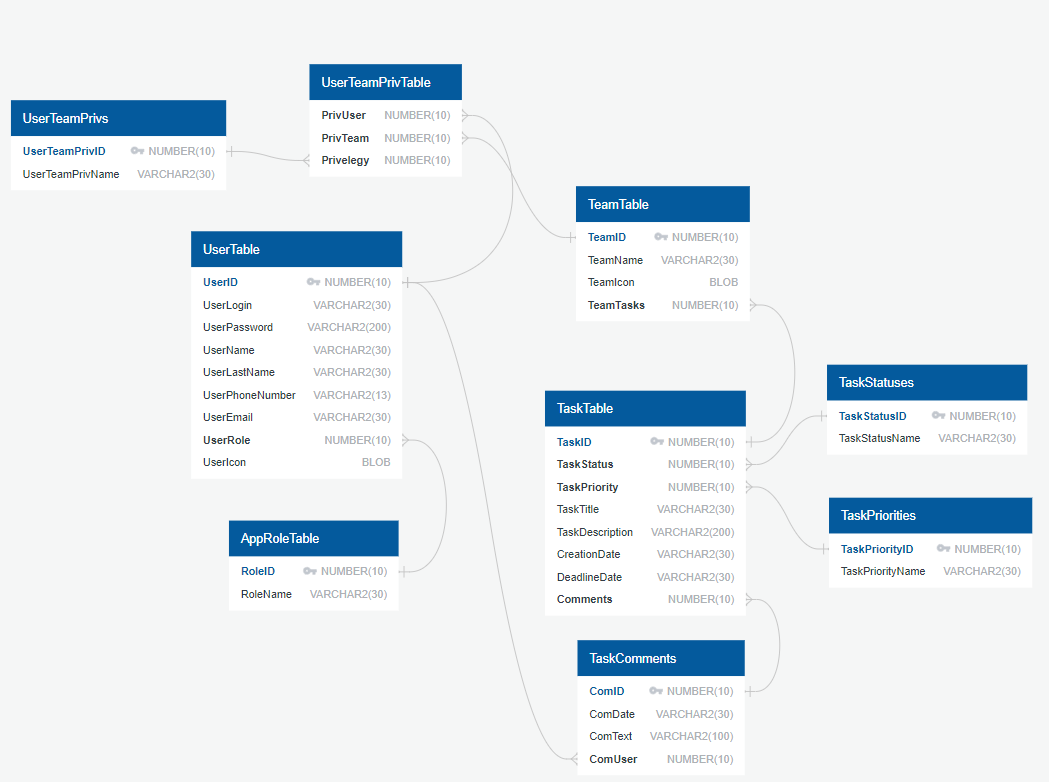
3. Шифрование в PL/SQL: уроки программирования [Электронный ресурс] / Режим доступа: <https://oracle-patches.com/db/sql/3939-shifrovanie-v-pl-sql-uroki-programmirovaniya> – Дата доступа: 16.10.2021.

4. DBMS\_XMLDOM [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/cd/B1930601/appdev.102/b14258/dxmldom.htm#i1076719> – Дата доступа: 20.10.2021.

5. Developing and Using Stored Procedures [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/cd/B2835901/appdev.111/b28843/tdddgprocedures.htm> – Дата доступа: 26.10.2021.

6. Stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com – Дата доступа: 08.12.2021

Приложение А



Приложение Б

