МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 Программное обеспечение информационных технологий (программирование интернет приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство для предоставления доступа к файлам

Исполнитель

студент 2 курса группы 5 Ворович Кирилл Сергеевич

(Ф.И.О.)

Руководитель работы ассистент Панченко О.Л.

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

Курсовой проект защищен с оценкой

Председатель Смелов В.В.

(подпись)

Минск 2024

# Содержание

[Содержание 2](#_Toc166688301)

[Введение 4](#_Toc166688302)

[1. Постановка задачи и обзор литературы 5](#_Toc166688303)

[1.1 Анализ аналогов и прототипов 5](#_Toc166688304)

[1.2 Постановка задачи 7](#_Toc166688305)

[1.3 Вывод по разделу 8](#_Toc166688306)

[2. Анализ требований к программному средству и разработке функциональных требований 9](#_Toc166688307)

[2.1 Технические средства разработки 9](#_Toc166688308)

[2.2 Описание разрабатываемой функциональности приложения 9](#_Toc166688309)

[2.3 Спецификация функциональных требований 10](#_Toc166688310)

[3 Проектирование программного средства 11](#_Toc166688311)

[3.1 Проектирование логической структуры базы данных 11](#_Toc166688312)

[3.2 Обобщенная структура проекта и детализация функций проектируемого программного средства 14](#_Toc166688313)

[3.3 Проектирование архитектуры приложения 15](#_Toc166688314)

[3.4 Схема работы приложения 16](#_Toc166688315)

[3.5 Проектирование логики сценариев использования 16](#_Toc166688316)

[4. Реализация программного средства 18](#_Toc166688317)

[4.1 Реализация Model 18](#_Toc166688318)

[4.2 Реализация ViewModel 22](#_Toc166688319)

[4.3 Реализация сервиса локализации 23](#_Toc166688320)

[4.4 Реализация функции регистрации 25](#_Toc166688321)

[4.5 Выводы по разделу 25](#_Toc166688322)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 26](#_Toc166688323)

[5.1 Тестирование возможности добавления разбора в избранные 26](#_Toc166688324)

[5.2 Тестирование возможности регистрации 27](#_Toc166688325)

[5.3 Тестирование возможности добавления разбора 28](#_Toc166688326)

[5.4 Тестирование возможности смены тональности 29](#_Toc166688327)

[5.5 Тестирование возможности добавления разбора 30](#_Toc166688328)

[5.6 Тестирование возможности входа в аккаунт 31](#_Toc166688329)

[5.7 Выводы по разделу 31](#_Toc166688330)

[6. Методика использования программного средства 32](#_Toc166688331)

[6.1 Руководство по использованию гостем 32](#_Toc166688332)

[6.2 Руководство по использованию авторизованном пользователем 34](#_Toc166688333)

[6.3 Руководство по использованию администратором 37](#_Toc166688334)

[6.4 Выводы по разделу 38](#_Toc166688335)

[ЗАКЛЮЧЕНИЕ 39](#_Toc166688336)

[Список использованных источников 40](#_Toc166688337)

[ПРИЛОЖЕНИЕ А 41](#_Toc166688338)

# Введение

В эпоху цифровизации и беспрерывного роста объемов информации, возникает необходимость в разработке инновационных инструментов для эффективного доступа и управления данными. Особенно это касается организаций и индивидуальных пользователей, для которых важен быстрый и безопасный доступ к файлам и контенту. В этом контексте, разработка программного средства, позволяющего предоставлять доступ к файлам, становится актуальной и востребованной задачей. Такой инструмент не только обеспечивает пользователей необходимым контентом, но и вносит значительный вклад в оптимизацию рабочих процессов, повышая общую производительность и удобство в использовании.

Целью данного курсового проекта является создание программного средства, которое будет предоставлять пользователям доступ к файлам и контенту, удовлетворяя при этом современные требования безопасности, масштабируемости и удобства использования. Разрабатываемое программное средство должно стать надежным инструментом в руках пользователей, позволяя им эффективно управлять доступом к нужным файлам и контенту, а также способствовать оптимизации рабочих и творческих процессов.

Для достижения поставленной цели, проект предусматривает решение следующих задач:

* Разработка удобного и интуитивно понятного интерфейса для пользователей, обеспечивающего легкий доступ к файлам и контенту.
* Обеспечение высокого уровня безопасности доступа к файлам, включая аутентификацию пользователей и шифрование данных.
* Реализация функционала для управления доступом к файлам, позволяя пользователю настраивать права доступа для различных категорий контента.
* Интеграция с современными облачными хранилищами данных для обеспечения масштабируемости и надежности хранения информации.

Основной целью курсового проекта является создание программного средства, которое удовлетворит все поставленные требования и задачи. Для этого будут использованы принципы объектно-ориентированного программирования, база данных MS SQL Server для хранения данных и технология Windows Presentation Foundation (WPF) для создания графического интерфейса.

# 1. Постановка задачи и обзор литературы

## 1.1 Анализ аналогов и прототипов

При разработке приложения для музыкантов, важно рассмотреть существующие аналоги и альтернативные решения, которые также предназначены для решения подобных задач. Этот обзор позволяет выявить преимущества и недостатки конкурирующих продуктов и определить, каким образом программное средство может выделиться на фоне аналогов.

patreon.com: Одна из крупнейших зарубежных веб-сайтов, на котором авторы творческих произведений могут распространять свои работы по платной подписке или предоставлять дополнительный контент для своих подписчиков. Они конкурируют напрямую с сервисом подписки на контент, предлагая аналогичные услуги. Рисунок 1.1.

|  |
| --- |
|  |

Рисунок 1.1 – Страница автора на сайте “patreon.com”

К недостаткам сайта можно отнести сильную перегруженность кнопками.

У сайта есть большое количество преимуществ: возможность установить картинки и название для подписок, минималистичный дизайн.

boosty.to: Еще один сервис подписки на контент, распространен среди российских авторов. Они также предстовляют аналогичные услуги. Рисунок 1.2.

|  |
| --- |
|  |

Рисунок 1.2 – Страница автора на сайте “boosty.to”

Вид страницы добавления постов представлен на рисунке 1.3.

На странице сразу видно описание автора, шапка страницы, иконка страницы, возможные подписки. На сайте отсутствует поиск авторов, поэтому автора можно найти только через ссылки со сторонних сайтов.

|  |
| --- |
|  |

Рисунок 1.3 – Страница добавления постов на сайте “boosty.to”

sponsr.ru: Менее известный сервис подписки на контент.  Платформа для размещения всех видов медиа-контента и заработка на подписках. Они также предлагают аналогичные услуги. Рисунок 1.4.

|  |
| --- |
|  |

Рисунок 1.4 – Страница поиска на сайте “ sponsr.ru ”

Стоит отметить, что данный сайт имеет возможность разделения авторов по категориям.

## 1.2 Постановка задачи

На основе анализа аналогичных решений и с учетом темы проекта был составлен список ролей пользователей и список функций, которые должно выполнять разрабатываемое приложение:

Основные задачи:

* создать каталог страниц авторов с возможностью перехода на страницу;
* разработать функционал для поиска страниц автора;
* разработать профиль пользователя, где будет располагаться созданных пользователем страниц, имя, почта, иконка пользователя;
* разработать добавления постов на страницу;
* разработать управления постами странице;
* провести тестирование программного средства;
* составить руководство по установке и использованию приложения.

## 1.3 Вывод по разделу

После изучения аналогичных продуктов были выявлены их основные преимущества и недостатки, что послужило основой для составления списка функций. Поскольку большинство аналогов обладают схожим функционалом, основное внимание было уделено красивому дизайну и удобству использования. Цель заключалась в том, чтобы обеспечить такой интерфейс, который был бы интуитивно понятен для любого пользователя, даже если он имел опыт работы лишь с одним из подобных приложений.

# Анализ требований к программному средству и разработке функциональных требований

Анализ требований в проекте программного обеспечения включает сбор, систематизацию и документирование требований от заказчика. Цель этого процесса - получить полное понимание задачи заказчика, определить границы проекта, выявить возможные проблемы и риски.

На этом этапе определяются основные методологические и технические требования, формулируются цели и задачи проекта, а также выявляются ключевые факторы успеха. Определение и описание требований являются критическими шагами, которые влияют на успешное завершение всего проекта, поскольку они определяют основу для всех последующих этапов. Этот процесс начинается с описания используемых технических средств, затем переходит к архитектуре системы и созданию диаграмм использования.

2.1 Спецификация функциональных требований к программному средству

Для выявления функциональных требований были определены следующие роли и группы пользователей:

Функции гостя:

* + Выполнение регистрации, авторизации;
  + Просмотр списка авторов;
  + Возможность поиска автора по имени;
  + Просмотр постов, не требующих подписки.

Функции Пользователь:

* + Возможность отправить запрос на подписку на автора;
  + Возможность оставить комментарий;
  + Возможность настроить профиль;
  + Возможность создать свои страници автора.

Функции автора:

* + Публикация постов;
  + Настройка страницы автора;
  + Удаление постов;
  + Удаление подписчиков;
  + Настройка уровней подписки;
  + Возможность настроить профиль;
  + Возможность принять запрос на подписку.

Функции администратора:

* + Добавление, удаление, редактирование постов пользователей;
  + Управления подписками пользователей;
  + Удаление страниц автора;
  + Удаление пользователей.

Причём, функции гостя наследует авторизованный пользователь, а функции авторизованного пользователя наследует администратор. Автор наследует функции пользователя.

В приложении А представлена диаграмма использования программного средства разными группами пользователей.

В данной главе были определены роли и группы пользователей, а также представлена диаграмма использования программного средства разными группами пользователей. В целом, спецификация функциональных требований к программному средству позволяет определить функциональность и ожидания пользователей, а также задает рамки для разработки и тестирования программного продукта.

2.2 Описание средств разработки

Для разработки программного средства будут использоваться следующие инструменты:

* Среда разработки Microsoft Visual Studio 2022;
* Программная платформа .NET;
* Язык программирования C#;
* Технология WPF;
* Microsoft SQL Server MS 19;
* Entity Framework (для создания и взаимодействия с базой данных).

2.3 Вывод по разделу

В данном разделе были определены требования к проекту, составлена User Case диаграмма, а также средства, которые будут использоваться в процессе разработки и функционирования программного средства.

3. Проектирование программного средства

3.1 Архитектура системы

В данном курсовом проекте используется архитектурный паттерн MVVM (Model-View-ViewModel). Он позволяет отделить логику приложения от визуальной части. MVVM состоит из трех частей:

* модель описывает используемые в приложении данные;
* представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением;
* модель представления связывает модель и представление через механизм привязки данных. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление.

Программное средство для предоставления доступа к файлам имеет следующую структуру, представленную на рисунке 3.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | |  |  |
|  |  |  |

Рисунок 3.1 – Структура проекта

Описание структуры основных папок проекта представлено в таблице 3.1.

Таблица 3.1 – Папки верхнего уровня

|  |  |
| --- | --- |
| Название папки/файла | Содержание |
| Resources | Папка, содержащая ресурсы для проекта |
| Utilities | Содержит разные утилиты, которые могут использовать как в View, так и в ViewModel |
| MVVM | Папка, содержащая компоненты, составляющие архитектуру приложения |

Более подробно можно рассмотреть содержимое папок “MVVM” и “Utilities”. Описание этих папок представлены в таблицах 3.2 и 3.3.

Таблица 3.2 – Папки в директории “MVVM”

|  |  |
| --- | --- |
| Название | Описание |
| Model | Хранит модели данных |
| View | Хранит представления окон |
| ViewModel | Хранит реализацию ViewModel |

Таблица 3.3 – Папки в директории “Utilities”

|  |  |
| --- | --- |
| Название | Описание |
| BaseModel | Класс для манипуляции базой данных |
| NavigationService | Содержит классы для реализации навигации между окнами |
| Store | Содержит классы, реализующие обмен данных между ViewModel |
| Service | Содержит сервисы, которые используются в классах ViewModel |

Таким образом, сформированная таблица помогает понять общую структуру проектируемого программного средства.

3.2 Взаимоотношения между классами

Для визуализации взаимосвязей между классами используется диаграмма UML – графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Для представления внутренней структуры программы в виде классов и связей между ними используется диаграмма классов. Приложение спроектировано таким образом, что каждый класс выполняет свои функции. Диаграмма классов представлена в приложении Б.

3.3 Модель базы данных

Для реализации поставленной задачи была создана база данных SUBSCRIPTIONBD. Для ее создания использовалась система управления реляционными базами данных MS SQL Server. В дальнейшем база данных будет создаваться с помощью Entity Framework и подхода Database-First. Database-First это один из подходов к разработке программного обеспечения, особенно к разработке приложений, которые используют базы данных. В этом подходе сначала проектируется и создается база данных, а затем на основе её структуры автоматически генерируется модель приложения (например, объектно-реляционная модель данных). Таким образом, база данных служит источником исходных данных для создания приложения. База данных состоит из 9 таблиц: Users, Logins, Files, Posts, AuthorPages, Comments, Requests, Subscriptions, Subscription\_types. Диаграмма базы данных представлена на рисунке 3.2.

|  |
| --- |
|  |

Рисунок 3.2 – Диаграмма базы данных

На рисунке 3.3 изображена структура таблицы Users, которая содержит информацию о пользователях.



Рисунок 3.3 – Структура таблицы Users

На рисунке 3.4 приведена структура таблицы AuthorPages, содержащая информацию о страницах автора.

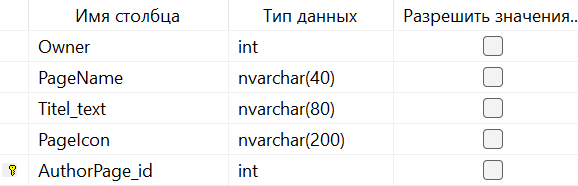


Рисунок 3.4 – Структура таблицы AuthorPages

На рисунке 3.5 изображена структура таблицы Logins, которая хранит информацию о логине и пароле пользователя.

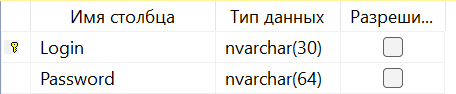


Рисунок 3.5 – Структура таблицы Logins

На рисунке 3.3 изображена структура таблицы Files, которая содержит информацию о файлах для постов.

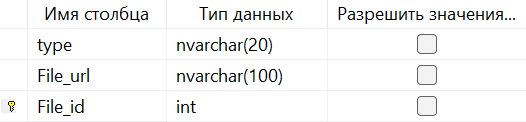


Рисунок 3.3 – Структура таблицы Files

На рисунке 3.3 изображена структура таблицы Posts, которая содержит информацию о постах.

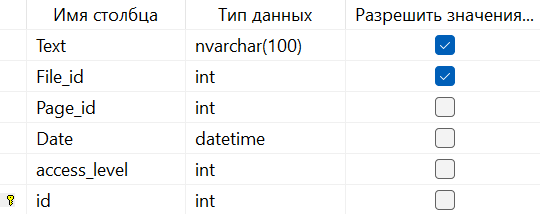


Рисунок 3.3 – Структура таблицы Posts

На рисунке 3.3 изображена структура таблицы Comments, которая содержит информацию о комментариях пользователей.

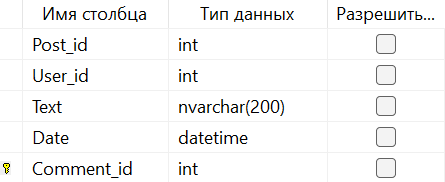


Рисунок 3.3 – Структура таблицы Сomments

На рисунке 3.3 изображена структура таблицы Requsts, которая содержит запросы на подписку.



Рисунок 3.3 – Структура таблицы Requsts

На рисунке 3.3 изображена структура таблицы Subscriptions, которая содержит информацию о подписках.

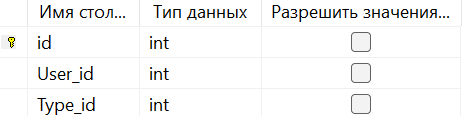


Рисунок 3.3 – Структура таблицы Subscriptions

На рисунке 3.3 изображена структура таблицы Subscription\_types, которая содержит информацию о типах подписок.

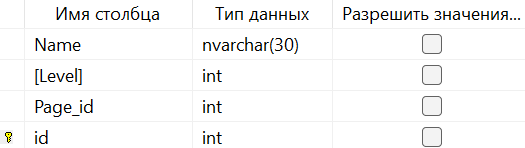


Рисунок 3.3 – Структура таблицы Subscription\_types

3.4 Проектирование последовательности взаимодействия

При проектировании последовательности взаимодействия объектов приложения была использована UML-диаграмма последовательности.

Диаграмма последовательности (Sequence Diagram) – это один из видов UML-диаграмм, который используется для описания последовательности взаимодействия объектов в системе. Диаграмма последовательности часто используется для описания процессов бизнес-логики, взаимодействия между компонентами системы, процессов ввода-вывода, взаимодействия между пользователем и системой и т.д.

Диаграмма последовательности взаимодействия соискателя и администратора с приложением представлена в Приложении В.

3.5 Вывод по разделу

В данном разделе представлена структура программного средства, включающая основные элементы с подробными пояснениями к каждому из них. Описана диаграмма классов, которая иллюстрирует взаимосвязи между различными сущностями в программном средстве, а также модификаторы доступа, необходимые для их функционирования. Также разработана диаграмма последовательности, которая используется для описания процессов бизнес-логики, взаимодействия между компонентами системы, а также процессов ввода-вывода и взаимодействия между пользователем и системой. Эти диаграммы помогают в понимании архитектуры и функционирования программного средства, а также облегчают коммуникацию между разработчиками и другими участниками проекта.

# 4. Реализация программного средства

## 4.1 Реализация Model

В рамках разрабатываемого проекта можно выделить 4 модели данных, которые будут представлены классами: User, Song, SubmittedSong, AppDbContext.

Класс User представляет пользователя. Код этого класса представлен в листинге 4.1. Как можно видеть на листинге, внутри каждого объекта данного класса хранится коллекция объектов класса Song, которые в данном контексте являются список избранных разборов. Также был реализован метод Equals, который будет автоматически вызывается при работе метода Contains пространства имён LINQ.

public class User

{

public int Id { get; set; }

public string? Login { get; set; }

public string? Password { get; set; }

public List<Song> Favorites { get; set; }

public User()

{

Login = "";

Password = "";

Favorites = new List<Song>();

}

public User(int id, string? login, string? password, List<Song> favorites)

{

Id = id;

Login = login;

Password = password;

Favorites = favorites;

}

public User(string? login, string? password)

{

Login = login;

Password = password;

Favorites = new List<Song>();

}

public User(string? login, string? password, List<Song> favorites)

{

Login = login;

Password = password;

Favorites = favorites;

}

public override bool Equals(object? obj)

{

return obj is User user &&

Login == user.Login &&

Password == user.Password;

}

}

Листинг 4.1 – Класс User

Класс Song является представлением разбора песни. Его код можно увидеть в листинге 4.2. Класс реализует интерфейс ICloneable для возможности клонирования объекта. Так же как и класс User, класс Song переопределяет метод Equals. Также в классе присутствует поле Chords, которое возвращает строку с найденными в тексте аккордами.

public class Song : ICloneable

{

public int Id { get; set; }

public string? Title { get; set; } = string.Empty;

public string? Artist { get; set; } = string.Empty;

public string? Text { get; set; } = string.Empty;

public int ViewsCount { get; set; } = 0;

public DateTime PublishedDate { get; set; }

public Song() { }

public Song(int id, string? title, string? artist, string? text, int viewsCount, DateTime publishedDate)

{

Id = id;

Title = title;

Artist = artist;

Text = text;

ViewsCount = viewsCount;

PublishedDate = publishedDate;

}

public string Chords

{

get

{

string pattern = @"\b[A-G][#|b]?(maj|m|min|dim|aug|sus)?[0-9]?\b";

MatchCollection matches = Regex.Matches(Text!, pattern);

List<string> uniqueChords = new List<string>();

foreach (Match match in matches)

{

string chord = match.Value;

if (!uniqueChords.Contains(chord))

{

uniqueChords.Add(chord);

}

}

return string.Join(' ', uniqueChords);

}

}

public object Clone()

{

return new Song(Id, Title, Artist, Text, ViewsCount, PublishedDate);

}

public override bool Equals(object? obj)

{

return obj is Song song &&

Id == song.Id &&

Title == song.Title &&

Artist == song.Artist &&

Text == song.Text;

}

public override int GetHashCode()

{

return HashCode.Combine(Id, Title, Artist, Text);

}

}

Листинг 4.2 – Класс Song

Класс SubmittedSong нужен для представления заявки пользователя на добавление разбора песни, код класса можно увидеть в листинге 4.3. Как можно видеть в листинге, кроме свойства самого разбора и идентификатора пользователя, который создал эту заявку, в классе есть свойство SubmittedStatus, которое показывает текущий статус заявки. Может быть три статуса: на рассмотрении, одобрено, отклонено. Также стоит обратить внимание на инкапсуляцию класса. Все свойства кроме Id и SubmittedStatus менять нельзя.

public class SubmittedSong

{

public int Id { get; set; }

public Song Song { get; }

public SubmittedStatus SubmittedStatus { get; set; }

public int UserId { get; }

public SubmittedSong(int id, Song song, SubmittedStatus submittedStatus, int userId)

{

Id = id;

Song = song;

SubmittedStatus = submittedStatus;

UserId = userId;

}

public SubmittedSong(Song song, SubmittedStatus submittedStatus, int userId)

{

Song = song;

SubmittedStatus = submittedStatus;

UserId = userId;

}

}

Листинг 4.3 – Класс SubmittedSong

Следующим рассмотренным классом будет AppDbContext, который необходим для взаимодействия с базой данных. Код этого класса представлен в листинге 4.4. Этот класс наследуется от DbContext и хранит в себе три коллекция для трёх таблиц. Также стоит уточнить, что в качестве типа этих коллекций являются классы с постфиксом “DTO”.

public class AppDbContext : DbContext

{

public DbSet<SongDTO> Songs { get; set; } = null!;

public DbSet<UserDTO> Users { get; set; } = null!;

public DbSet<SubmittedSongDTO> SubmittedSongs { get; set; } = null!;

public AppDbContext()

{

Database.EnsureCreated();

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer(@"Server=localhost;Database=CourseProjectDb;Trusted\_Connection=True;TrustServerCertificate=True;");

}

}

Листинг 4.4 – Класс AppDbContext

DTO, или объект передачи данных – это специальный объект, который используется при передаче данных между системами. В данном случае, происходит передачи данных между приложением и базой данных.

В качестве параметра для DbSet можно было бы использовать уже существующие классы без постфикса “DTO”, но тогда в базе данных могли бы хранится ненужные данные, например, вычисляемое свойство Chords в классе Song. Также, использование DTO позволяет отделить бизнес-логику от данных. Код класса SongDTO можно увидеть в листинге 4.5. В данном классе есть исключительно публичные свойства, необходимые для работы с базой данных. По аналогии с эти классом реализованы классы: SubmittedSongDTO и UserDTO.

public class SongDTO

{

public int Id { get; set; }

public string? Title { get; set; }

public string? Artist { get; set; }

public string? Text { get; set; }

public int ViewsCount { get; set; }

public DateTime PublishedDate { get; set; }

public List<UserDTO> Users { get; set; } = new();

}

Листинг 4.5 – Класс SongDTO

## 4.2 Реализация ViewModel

Как было сказано в подразделе 3.3, ViewModel связывает данные и представление. Для реализации функции связывания должен быть реализован интерфейс INotifyPropertyChanged. Его реализует класс ObservableObject, код которого представлен в листинге 4.6.

public class ObservableObject : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string propName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propName));

}

}

Листинг 4.6 – Класс ObservableObject

Базовым для всех ViewModel является абстрактный класс ViewModelBase. Код класса можно увидеть в листинге 4.7.

public abstract class ViewModelBase : ObservableObject

{ }

Листинг 4.7 – Класс ViewModelBase

Использование связывания можно рассмотреть на примере класса MainViewModel. В данном классе есть свойство UserLogin, код которого представлен в листинге 4.8. При присваивании значения данному свойству будет вызывается метод OnPropertyChanged(), сигнализирующий об изменении значения этого свойства.

public string UserLogin

{

get => \_userLogin;

set { \_userLogin = value; OnPropertyChanged(); }

}

Листинг 4.8 – Свойство UserLogin

Кроме реализации интерфейса INotifyPropertyChanged необходимо указать свойство DataContext для представления. Значением для этого свойства будет объект класса MainViewModel. И затем, с помощью команды Binding в представлении происходит привязка данных. Фрагмент привязки свойства UserLogin представлен в листинге 4.9.

<TextBlock

IsEnabled="{Binding IsAccountHighlight}"

Margin="0 5 0 0"

Style="{StaticResource HeaderTextBlock}"

Text="{Binding UserLogin}"/>

Листинг 4.9 – Фрагмент представления

## 4.3 Реализация сервиса локализации

В приложении есть возможность выбора одно из двух языков: русского и английского. Чтобы реализовать данный функционал были созданы два словаря ресурсов для каждого из языков. Фрагмент кода словаря для русского языка находится в листинге 4.10. Каждая строчка, которую может увидеть пользователь, находится в данном словаре.

<!-- MainView -->

<system:String x:Key="MainSongsString">Разборы</system:String>

<system:String x:Key="MainFavoritesString">Избранные</system:String>

<system:String x:Key="MainSubmitSongString">Добавить</system:String>

<system:String x:Key="MainMySongsString">Мои разборы</system:String>

<system:String x:Key="MainSignInString">Войти</system:String>

Листинг 4.10 – Фрагмент словаря для русской локализации

Чтобы подключить динамическую смену локализации к представлению достаточно использовать DynamicResource при подключении необходимого ресурса. Однако приложение имеет механизм оповещения, который должен получать строку из словаря текущей локализации. Для того, чтобы способ получения необходимой строки был наиболее надёжным и простым, был реализован класс LocalizationService. Код реализации данного класса представлен в листинге 4.11.

public class LocalizationService : ILocalizationService

{

private ResourceDictionary \_currentResourceDictionary;

private Language \_language;

public event PropertyChangedEventHandler PropertyChanged;

public string this[StringResourceKey key]

=> \_currentResourceDictionary[key.ToString()] as string ?? string.Empty;

public Language Language

{

get => \_language;

set

{

if (value == \_language)

return;

\_language = value;

ResourceDictionary newLocalization = new ResourceDictionary();

ResourceDictionary? currentLocalization;

if (Language == Language.English)

{

newLocalization.Source = new Uri("Styles/Localization/Language.en.xaml", UriKind.Relative);

currentLocalization = Application.Current.Resources.MergedDictionaries.FirstOrDefault(

m => m.Source.OriginalString.EndsWith($"Language.ru.xaml"));

}

else

{

newLocalization.Source = new Uri("Styles/Localization/Language.ru.xaml", UriKind.Relative);

currentLocalization = Application.Current.Resources.MergedDictionaries.FirstOrDefault(

m => m.Source.OriginalString.EndsWith($"Language.en.xaml"));

}

if (currentLocalization != null)

Application.Current.Resources.MergedDictionaries.Remove(currentLocalization);

Application.Current.Resources.MergedDictionaries.Add(newLocalization);

\_currentResourceDictionary = newLocalization;

OnPropertyChanged();

}

}

public void OnPropertyChanged([CallerMemberName] string propName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propName));

}

}

Листинг 4.11 – Класс LocalizationService

Таким образом, чтобы сменить языка всего приложения достаточно присвоить свойству Language значение языка. Индексатор служит для того, чтобы из объектов хранилищ получать строку оповещения с текущей локализацией. В перечислении StringResourceKey хранятся все имена ресурсов, используемые в словарях. Также в данном классе реализован интерфейс INotifyPropertyChanged, благодаря которому все ViewModel, которые используют видимые пользователю строки, будут оповещены об изменении локализации.

## 4.4 Реализация функции регистрации

Диаграмма последовательности функции регистрации представлена в Приложении А. Как можно видеть на диаграмме, после ввода всех данных производится запрос в объект класса UserStore для регистрации нового пользователя. Затем, из UserStore происходит обращение к репозиторию, где происходит добавление нового пользователя и сохранение изменения базы данных. После выполнения всех действия с базой данных происходит возвращение объекта уведомления. Этот объекта передаётся в объект класса AlertStore, которое выводит уведомление пользователю.

## 4.5 Выводы по разделу

Были реализованы модели данных, который используются приложением, а также были реализованы специальные модели, которые используются в базе данных. Такое разделение позволило отделить бизнес-логику от данных.

Далее были реализованы ViewModel, которые связывают представления и данные.

Важной функцией является смена языка, которая также была реализована с помощью словарей ресурсов, динамической привязки и сервиса LocalizationService. Именно с помощью LocalizationService была получена возможность удобного и безопасного использования ресурсов в коде C#, а не только в файлах представления.

Была разработана и описана функция регистрация нового пользователя.

5. Тестирование, проверка работоспособности и анализ полученных результатов

## 5.1 Тестирование возможности добавления разбора в избранные

В приложении используется три роли пользователя: гость, авторизированный пользователь, администратор. У гостя не должно быть возможности добавления разбора в избранные. Как видно на рисунке 5.1, у гостя данная кнопка заблокирована. Тогда как у авторизированного пользователя и администратора соответствующая кнопка доступна.

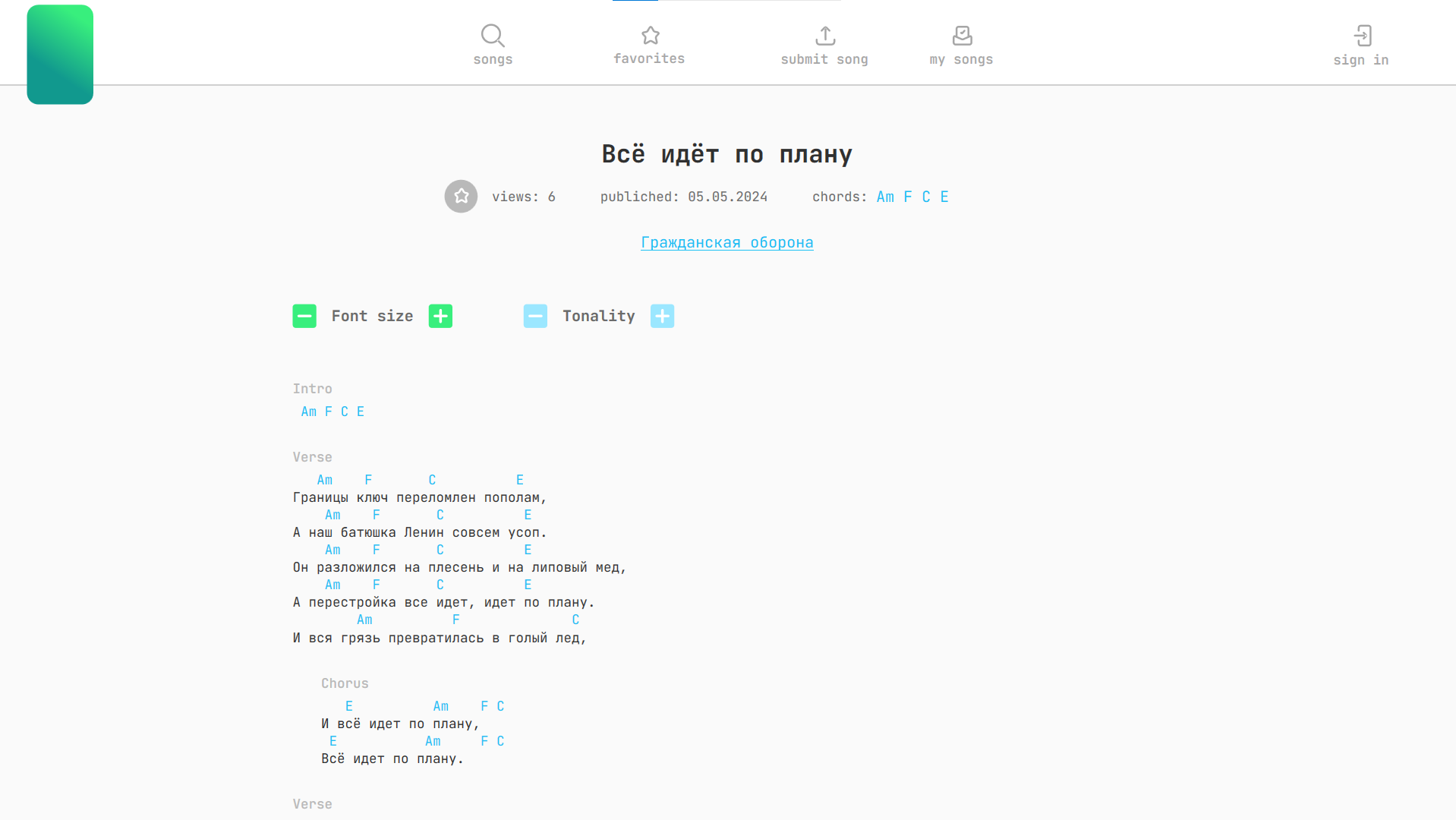


Рисунок 5.1 – Окно выбранного разбора

При нажатии этой кнопки авторизованный пользователем или администратором появляется соответствующее уведомление, как это показано на рисунке 5.2.

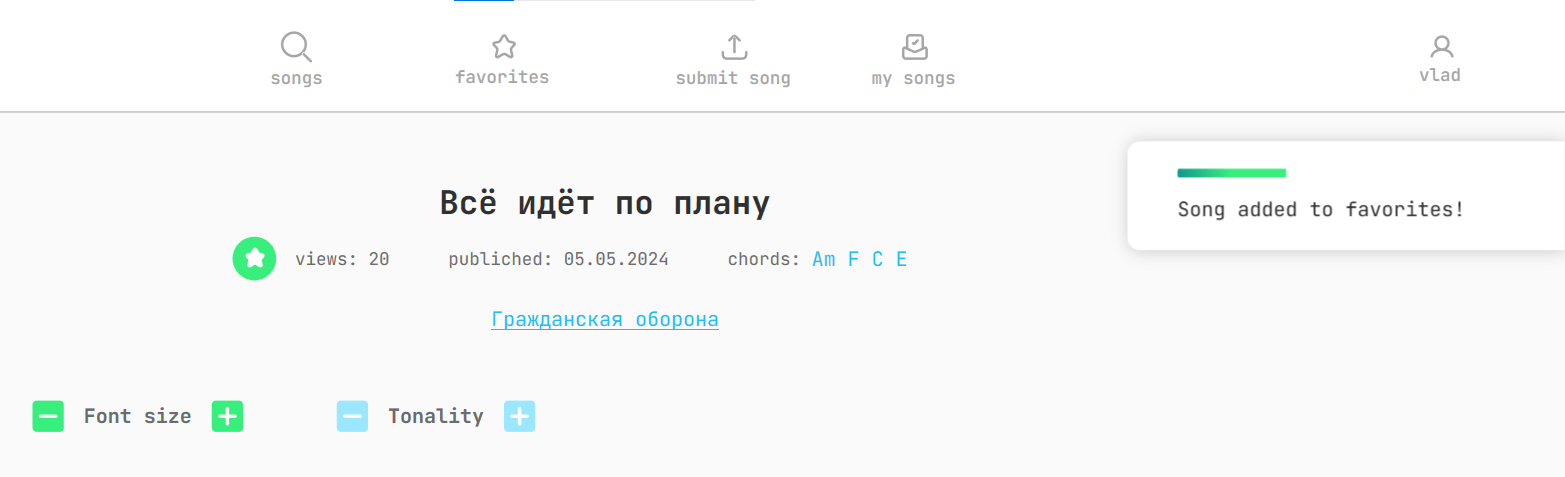


Рисунок 5.2 – Окно выбранного разбора

Также, при попытке зайти в окно с избранными разборами гость должен автоматически переходить в окно со входом в аккаунт, а в правом углу должно появляется соответствующее уведомление, что и продемонстрировано на рисунке 5.3.

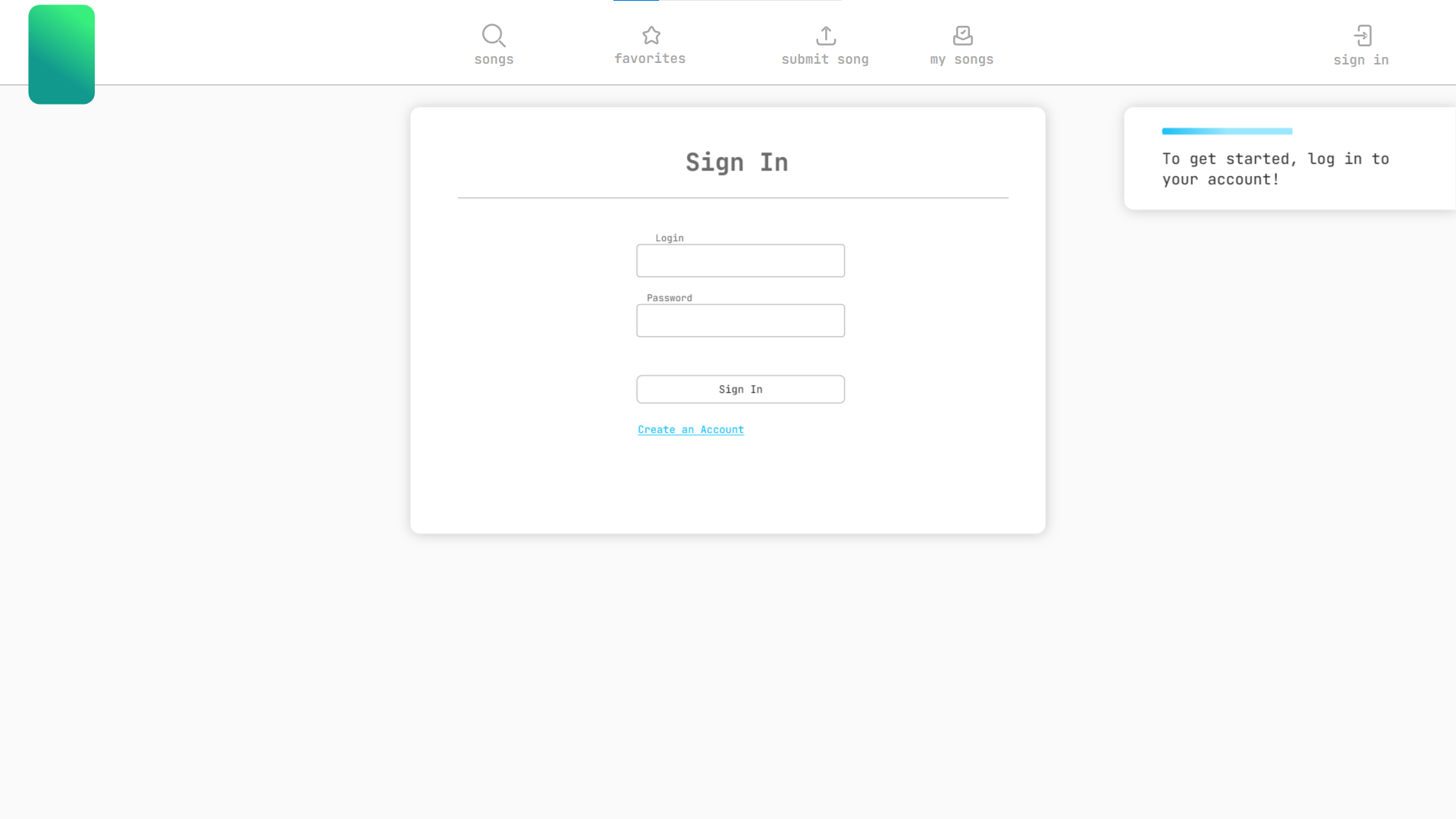


Рисунок 5.3 – Окно входа в аккаунт

## 5.2 Тестирование возможности регистрации

Если поля регистрации являются не валидными, то нажатие на кнопку регистрации не доступно, как показано на рисунке 5.4.



Рисунок 5.4 – Окно регистрации

Однако после заполнения полей правильными данными кнопка становится доступной, что видно на рисунке 5.5.

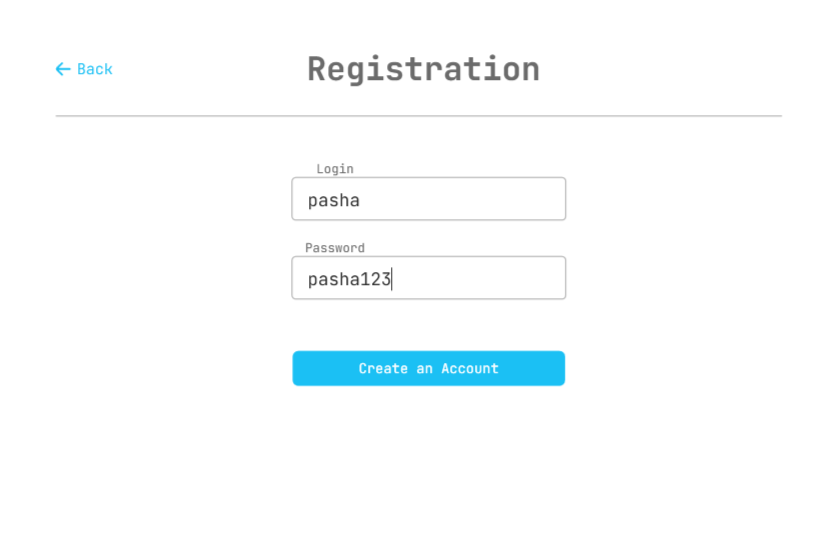


Рисунок 5.5 – Окно регистрации

После нажатия на кнопку будет произведена проверка на наличие пользователя с таким логином в базе данных, и если такой пользователь существует, то будет выведено уведомление, иначе пользователя перенаправит на окно входа и новый пользователь будет добавлен в базу данных. На рисунке 5.6 показан случай с повторяющимся логином.

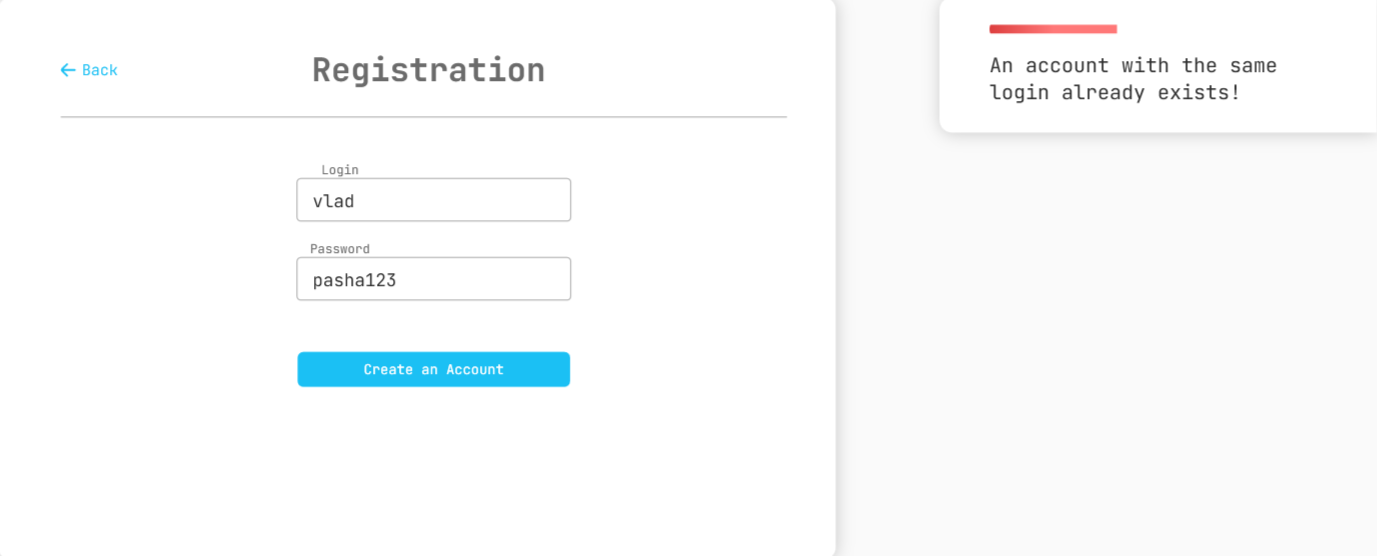


Рисунок 5.6 – Окно регистрации

## 5.3 Тестирование возможности добавления разбора

Если авторизованный пользователь хочет добавить свой разбор, то он переходит в окна для добавления разбора и заполняет поля. Все поля должны быть заполнены. В ином случае будет выведено оповещение, как это показано на рисунке 5.7.

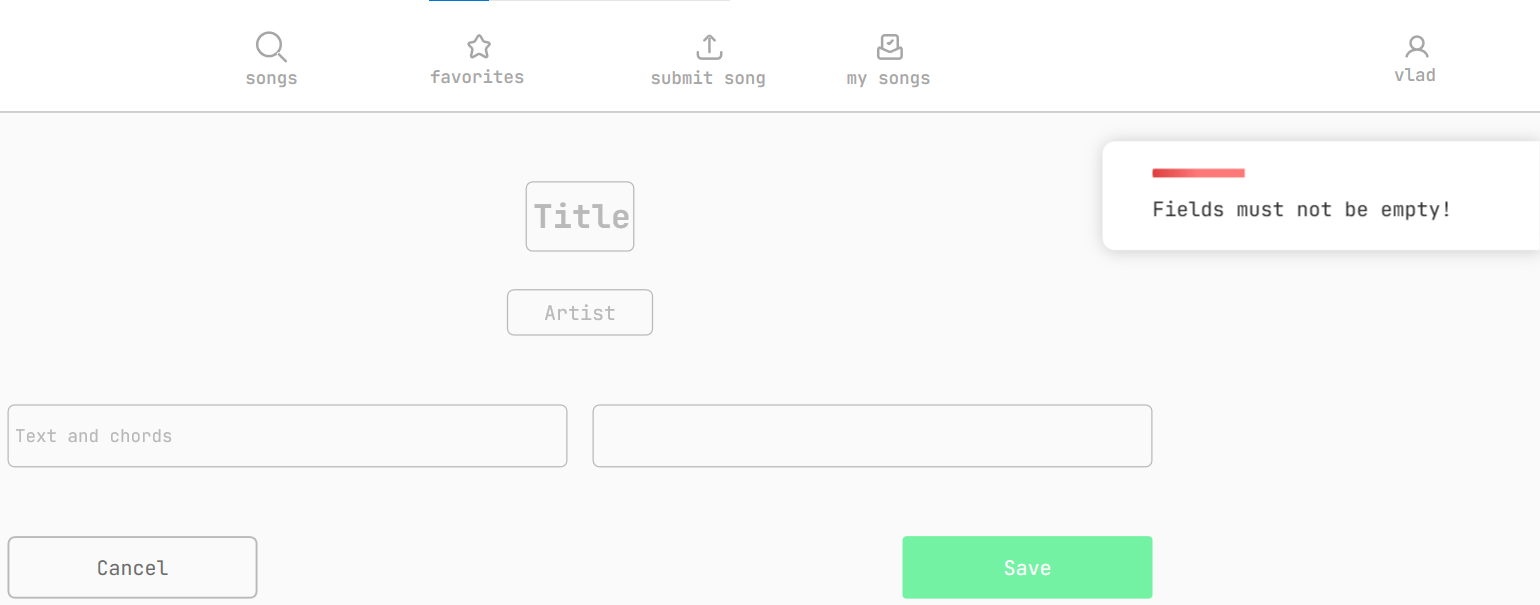


Рисунок 5.7 – Окно добавления разбора

Если не была нажата кнопка “*Save*” или “*Cancel*”, а пользователь вышел из этого окна, то в следующий раз при нажатии на кнопку для перехода в окно добавления песни пользователь перейдёт в окно добавления прошлого, не сохранённого разбора. Это продемонстрировано на рисунке 5.8.

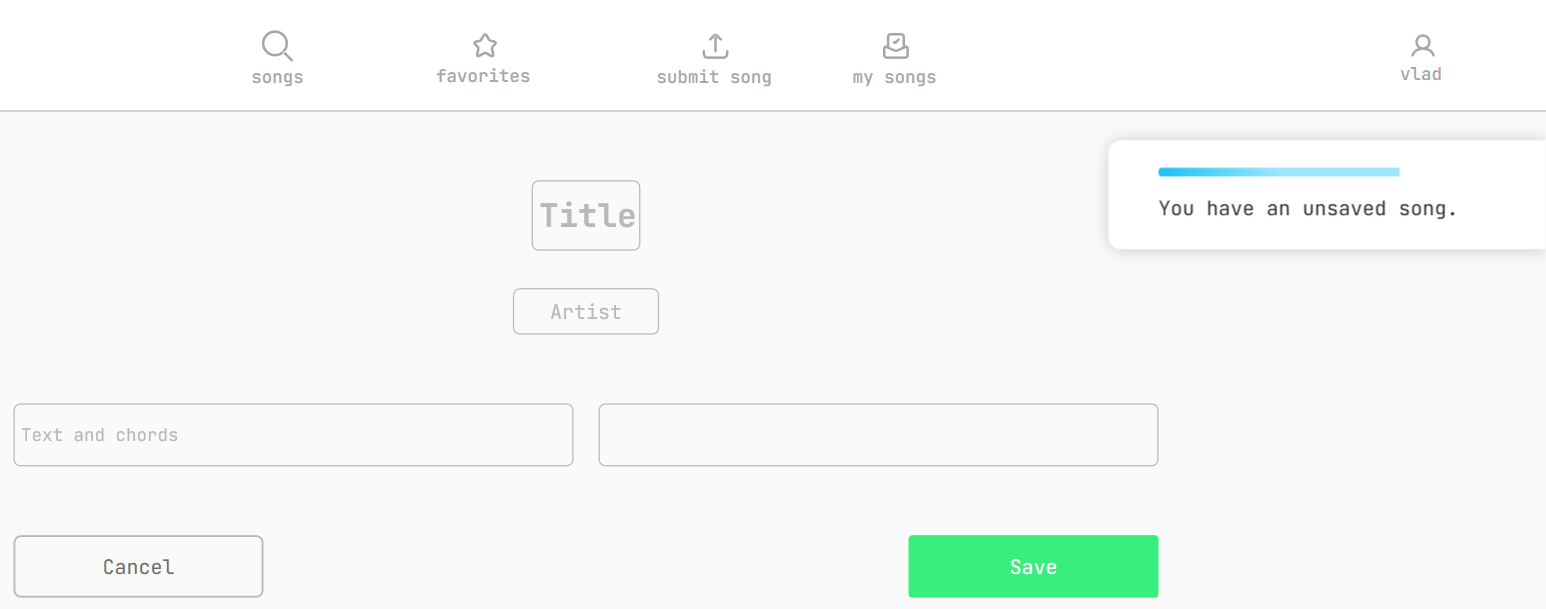


Рисунок 5.8 – Окно добавления разбора

## 5.4 Тестирование возможности смены тональности

Изначально разбора песни “Всё идёт по плану” имеет аккорды: Am, F, C, E. Однако после поднятия тональности разбора будет иметь аккорды, показанные на рисунке 5.9.

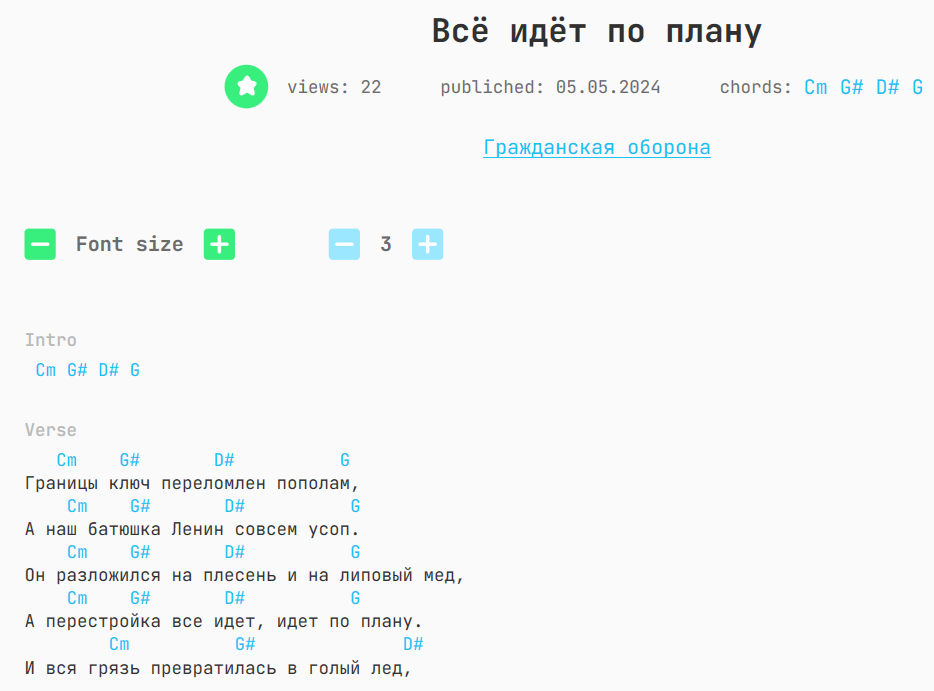


Рисунок 5.9 – Окно выбранного разбора

## 5.5 Тестирование возможности добавления разбора

Авторизованным пользователь был добавлен разбор с названием “Новый”. Данную заявку может видеть администратор в окне “My Songs”, что видно на рисунке 5.10.

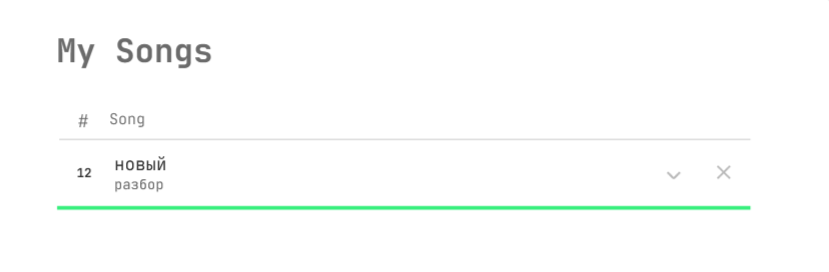


Рисунок 5.10 – Окно заявок

После нажатия на кнопку одобрения разбор будет добавлен в база данных, и заявка будет иметь статус одобренной. Также администратору высветиться уведомление, как на рисунке 5.11.



Рисунок 5.11 – Окно заявок

## 5.6 Тестирование возможности входа в аккаунт

У гостя есть возможность входа в свой аккаунт. Для это необходимо ввести в соответствующие поля логин и пароль, а затем нажать на кнопку входа.

Если логин и пароль совпадают с ведёнными, то производится вход в аккаунт, пользователя перенаправляет в окно аккаунта и выводится уведомление. Это можно увидеть на рисунке 5.12.

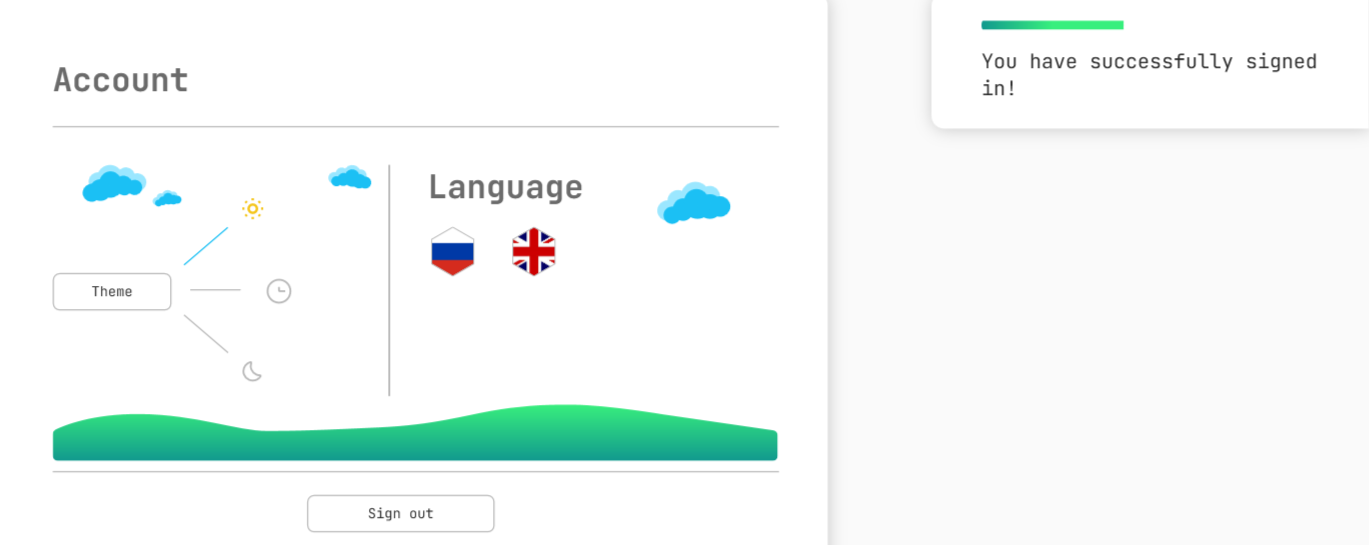


Рисунок 5.12 – Окно аккаунта

В ином случае, пользователь остаётся на странице входа, выводится уведомление и на время блокируется кнопка входа, что и показано на рисунке 5.13.

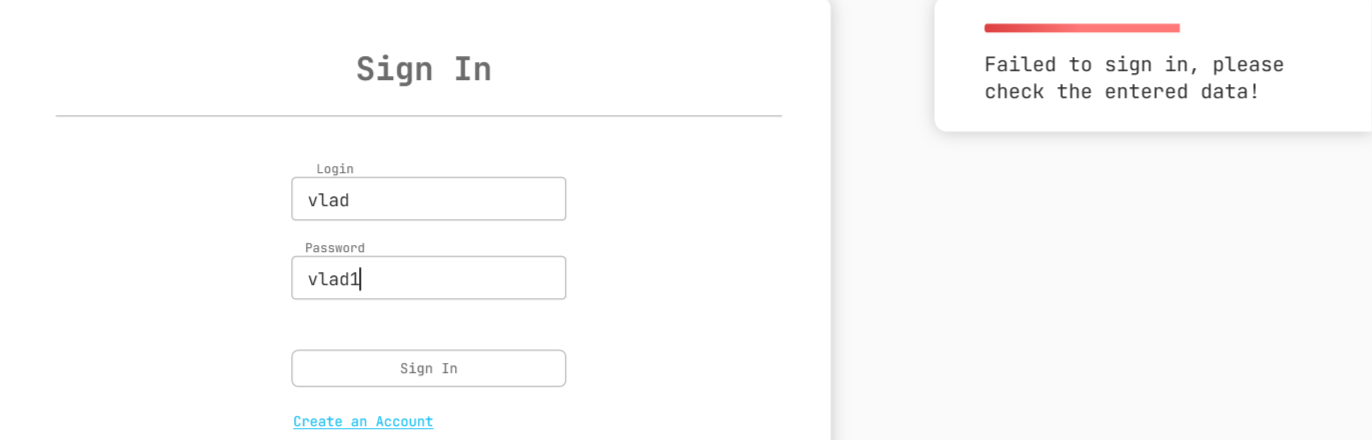


Рисунок 5.13 – Окно входа в аккаунт

## 5.7 Выводы по разделу

Разными ролями были протестированы основные функции приложения, а именно возможности: добавления разбора в избранные, регистрации и входа в аккаунт, добавления разбора песни, смены тональности.

По результатам тестирования можно сказать, что приложение работает правильно.

# 6. Методика использования программного средства

## 6.1 Руководство по использованию гостем

При запуске приложения начальная роль пользователя будет гостем. При данной роли ему доступно:

* поиск разбора песни;
* сортировка разборов песен;
* просмотр разбора песни;
* смена размера шрифта текста разбора;
* смена тональности разбора песни;
* вход в аккаунт;
* регистрация.

Для поиска разбора песни и сортировки всех разборов можно воспользоваться соответствующими кнопками, показанными на рисунке 6.1.

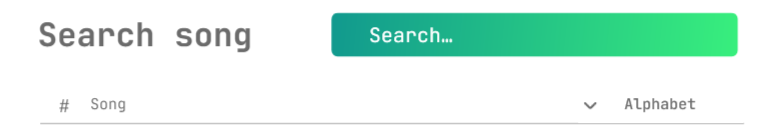


Рисунок 6.1 – Поиск и сортировка

Чтобы просмотреть разбора, надо дважды нажать на выбранный разбор. При наведении на разбор он будет подчёркиваться, как на рисунке 6.2.

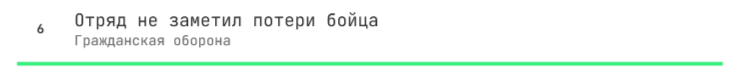


Рисунок 6.2 – Выбранный разбор

Для смены размера текста разбора есть две кнопки над текстом разбора, вид которого продемонстрирован на рисунке 6.3. При нажатии на кнопку со знаком “-” размер будет уменьшаться, при нажатии на “+” увеличиваться. Также кнопки можно зажимать.



Рисунок 6.3 – Кнопки изменения размера текста разбора

Для транспонирования можно воспользоваться двумя кнопками, который показаны на рисунке 6.4. Они также расположены над текстом разбора. При изменении смещения тональности между кнопок будет указано смещение. Значение “0” означает, что смещения нет.



Рисунок 6.4 – Кнопки изменения тональности

Для входа в аккаунт необходимо ввести логин и пароль и нажать на кнопку для входа. Вид окна входа показан на рисунке 6.5. В том случае, если введённый данные верны, то производится вход в аккаунт. Если данные введены неправильно, но приложение уведомляет пользователя у неверно ведённый данных.

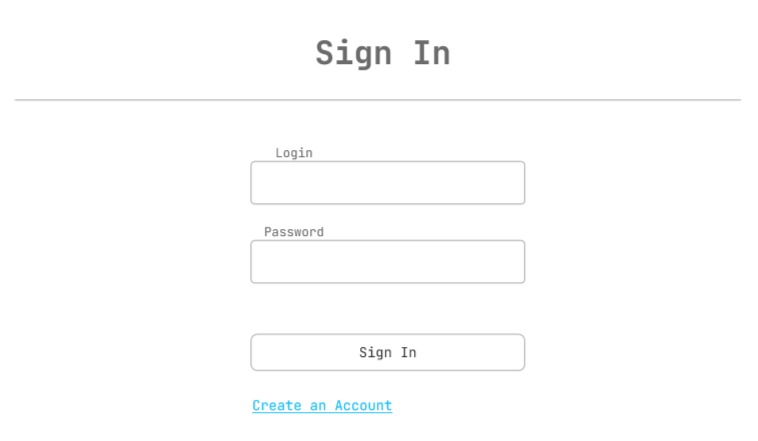


Рисунок 6.5 – Окно входа в аккаунт

Чтобы попасть в окно регистрации, необходимо нажать на кнопку “*Create an Account*”. Окно регистрации продемонстрировано на рисунке 6.6.

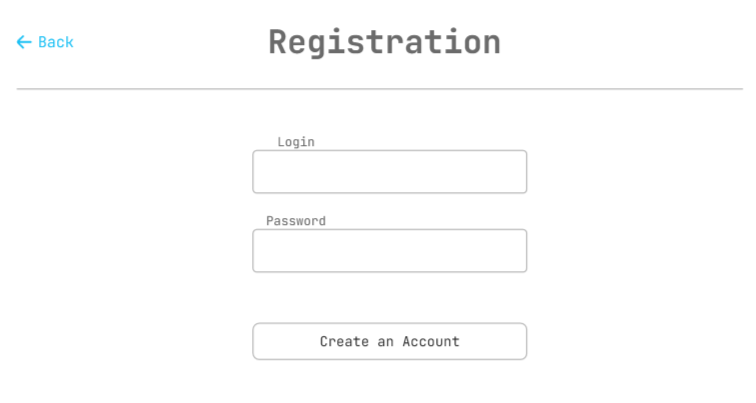


Рисунок 6.6 – Окно регистрации

У пользователя есть возможность перейти обратно в окно для входа, нажав на кнопку в левом верхнем углу. Кнопка для регистрации заблокирована до тех пор, пока все данные не введены корректно.

При наличии ошибок в полях сразу же будет показываться красная надпись с подсказкой. Пример таких подсказок показан на рисунке 6.7.

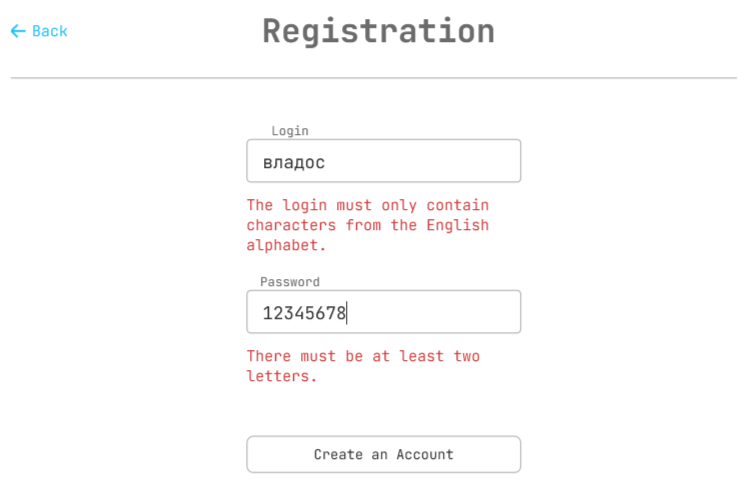


Рисунок 6.7 – Окно регистрации

При правильно вводе всех полей и нажатии кнопку для регистрации будет создан аккаунт и произведён переход в окно входа.

## 6.2 Руководство по использованию авторизованном пользователем

Пользователь получает роль авторизованного пользователя после входа в аккаунт. Данный пользователь может всё то, что может гость, а также имеет следующие возможности:

* добавление разбора в избранные;
* удаление разбора из избранных;
* работа с заявками на добавление собственного разбора песни;
* смена языка;
* смена темы;
* выход из аккаунта.

Чтобы добавить разбор в избранные, необходимо перейти в окно конкретного разбора и нажать на кнопку в форме звезды, вид которого продемонстрирован на рисунке 6.8. Если звезда закрашена, то это означает, что разбор уже добавлен в избранные. При нажатии на закрашенную кнопку разбора будет удалён из избранных.

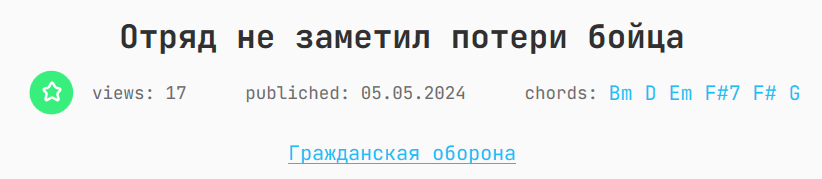


Рисунок 6.8 – Окно выбранной песни

У пользователя есть возможность посмотреть список всех своих избранных разборов. Вид окна показан на рисунке 6.9. При наведении на разбор в правой части появляется кнопка для удаления разбора из избранных.

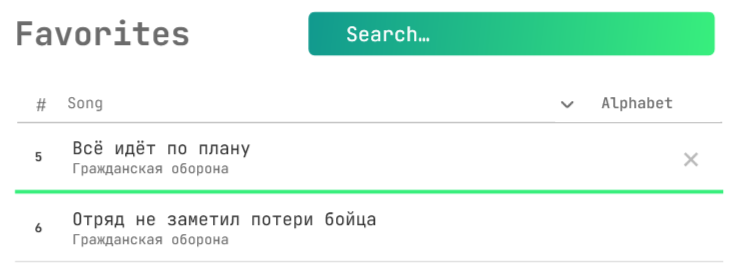


Рисунок 6.9 – Окно избранных разборов

Собственный разбор песни можно добавить в окне “submit song”, вид которого можно посмотреть на рисунке 6.10. Здесь предлагается ввести название разбора и автора песни, но данные поля вводить необязательно.

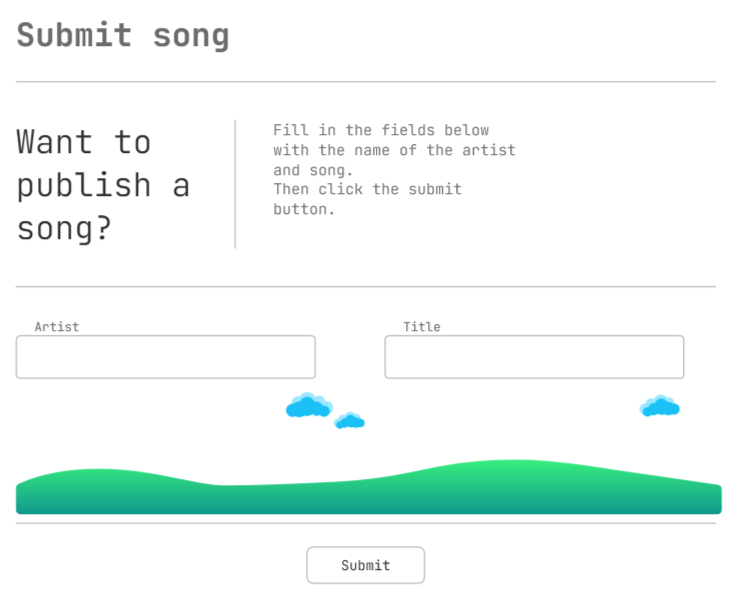


Рисунок 6.10 – Окно добавления разбора

После нажатия на кнопку “*Submit*” пользователь перейдёт в окно редактирования разбора. Как видно на рисунке 6.11, имеется три поля для названия разбора, исполнителя, текста и аккордов. Если хотя бы одно из полей не заполнено, то пользователь будет уведомлён о том, что необходимо заполнить все поля, чтобы добавить разбор.

Нажатие на кнопку “Save” сохраняет разбор, а кнопка “Cancel” удаляет текущий разбор. Если пользователь выйдет из этого окна, не сохранив и не отменив добавление разбора, то при возврате на окно “Submit song” он попадёт в свой не сохранённый разбор.

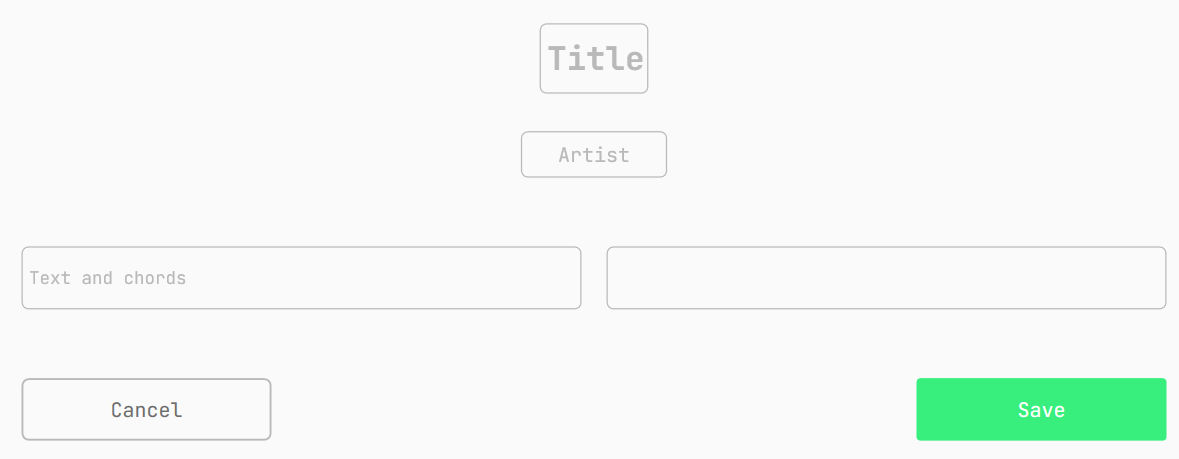


Рисунок 6.11 – Окно редактирования разбора

В окне “My Songs” можно просмотреть все свои заявки на добавления разборов. Заявки разбиты на три группы. Отклонённые разборы можно удалять из списка, как показано на рисунке 6.12.



Рисунок 6.12 – Окно редактирования разбора

Смена темы и языка осуществляется в окне аккаунта, что показано на рисунке 6.13. Там же есть кнопка для выхода из аккаунта.

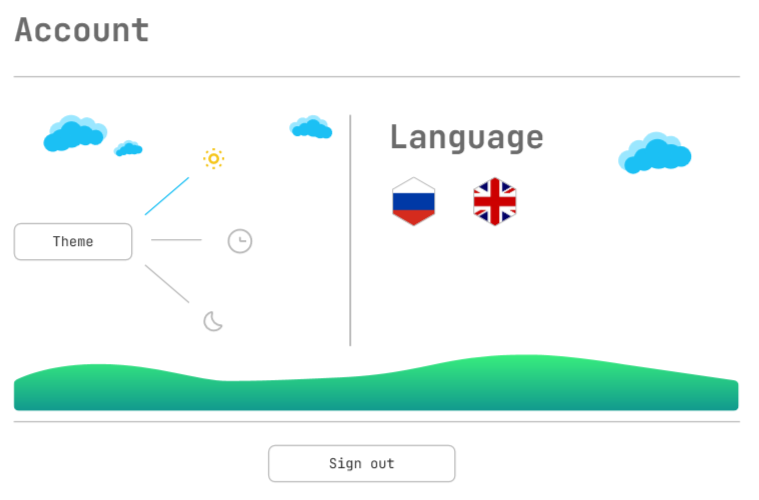


Рисунок 6.13 – Окно аккаунта

## 6.3 Руководство по использованию администратором

У администратора есть все возможности, которые имеет авторизированный пользователь, а также следующие:

* редактирование разбора;
* добавление разбора;
* удаление разбора;
* одобрение заявки на добавление разбора;
* отклонение заявки на добавление разбора.

Панель с кнопками редактирования и добавления разбора появляется при наведении курсора на разбор песни в окне со всеми разборами. Это показано на рисунке 6.14.



Рисунок 6.14 – Панель инструментов администратора

При нажатии на иконку крестика разбор будет удалён. Об этом также будет сигнализировать уведомление.

При нажатии на иконку карандаша будет произведён переход в окна редактирование, которое имеет такой же функционал и дизайн, что и окно на рисунке 6.11.

Администратор может добавлять разборы песен без какого-либо одобрения со стороны, как это было у авторизованного пользователя.

Окно “My Songs” для администратора имеет вид, изображённый на рисунке 6.15. При наведении на заявку из данного списка появляется две кнопки: одобрения и отклонения заявки на добавление разбора песни.

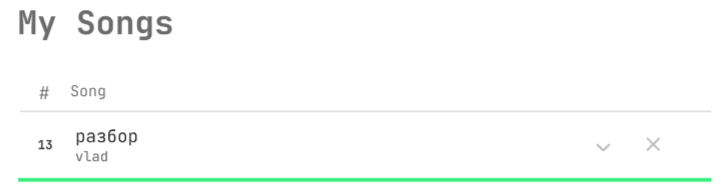


Рисунок 6.15 – Окно заявок

## 6.4 Выводы по разделу

В данном разделе представлено руководство по использованию приложения в различных ролях: гость, авторизованный пользователь и администратор. Самое большое количество возможностей имеет администратор, а самое маленькое – гость.

Данное руководство в полной мере показывает и описывает весь функционал приложения по ролям пользователя.

# ЗАКЛЮЧЕНИЕ

В итоге выполнения данного курсового проекта было разработано приложение для музыкантов. При разработке были выполнены все пункты из указанного списка предполагаемого основного функционала приложения. Также были реализованы дополнительные функции, которые лаконично вписываются и дополняют приложение.

Спроектирована и реализована эффективная архитектура проекта и её модулей, налажена не только логика взаимодействия с базой данных, а ещё и логики взаимодействия между разными компонентами приложения, создан алгоритм смены тональности песни, все основные функции приложения протестированы, а также, написано подробное руководство использования приложения согласно роли пользователя.

# Список использованных источников

1. AmDm [Электронный ресурс]. – Режим доступа: https://amdm.ru/.
2. MyChords [Электронный ресурс]. – Режим доступа: https://mychords.net/ru/.
3. Songsterr [Электронный ресурс]. – https://www.songsterr.com.
4. Работа с Entity Framework Core [Электронный ресурс] – https://professorweb.ru/my/entity-framework/6/level1/.
5. Полное руководство по языку программирования C# 8.0 и платформе .NET 4.7. Режим доступа: https://metanit.com/sharp/tutorial/.
6. Руководство по WPF [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/.

# ПРИЛОЖЕНИЕ А

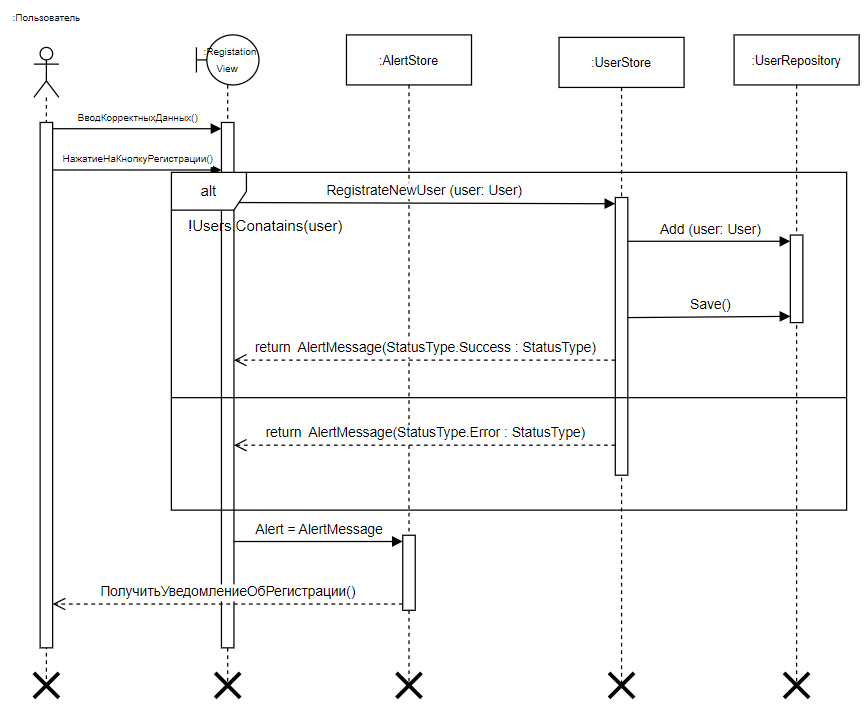


Рисунок A.1 – Диаграмма последовательности для регистрации