♥ MUSICO ♥

Online Music Catalog

Курсов проект – Национална програма „Обучение за ИТ умения и кариера“

Уеб приложение за стрийминг и управление на музикално съдържание

Математическа гимназия „Академик Кирил Попов“
4001 Пловдив, ул. „Чемшир“ 11
тел.: + 359 32 643 157,
E – mail: https://omg-bg.com/

Ръководител: Росен Вълчев

Изготвили проекта:
Анастасия Хронева
Деница Тренчева
Васил Пищялов

Съдържание

1. Цели на проекта

Целта на проекта е разработването на уеб приложение за стрийминг и управление на музикално съдържание, вдъхновено от платформата Spotify. Приложението предоставя възможност на потребителите да откриват музика, да създават плейлисти и да управляват своето съдържание, като същевременно осигурява различни нива на достъп според ролята на потребителя.

Основните цели на проекта включват:

- управление на музикално съдържание (песни и плейлисти);
- поддръжка на потребителски профили и социални взаимодействия;
- разделение на потребителите по роли;
- интуитивен и модерен уеб интерфейс;
- сигурност и контрол на достъпа.

2. Разпределение на ролите

Разработката на проекта е реализирана в екип, като задачите са разпределени между участниците – работа по backend логиката, базата данни, потребителския интерфейс, документацията и тестването.

3. Ниво на сложност на проекта

Проектът има средно към високо ниво на сложност, тъй като комбинира уеб разработка, ORM framework,

архитектура.релационна база данни, авторизация с роли и MVC

3.1. Многослойна архитектура

Приложението е реализирано чрез следните слоеве:

- **Models (Data Layer)** – описват ентитетите и връзките между тях
- **Controllers (Service Layer)** – съдържат бизнес логиката и обработката на заявки
- **Views (Presentation Layer)** – уеб интерфейсът на приложението

Това разделение улеснява поддръжката и разширяемостта на системата.

3.2. Работа с база данни

За работа с базата данни се използва **Entity Framework Core**.
Контекстът **`Music Context`** управлява връзката между приложението и SQL Server базата данни.

Реализирани са:

- миграции;
- навигационни свойства;
- много към много връзки;
- CRUD операции

3.3. Уеб интерфейс и потребителско изживяване

Интерфейсът е изграден с Razor Views, HTML, CSS и Bootstrap. Навигацията е интуитивна, като различните роли имат достъп до различни функционалности.

3.4. Работа с Git и GitHub

Проектът е разработен с използване на Git и GitHub, което позволява:

- контрол на версиите;
- екипна работа;
- проследяване на промените.

3.5. Разширяемост на проекта

Архитектурата позволява лесно добавяне на:

- препоръчваща система;
- харесвания и коментари;
- мобилна версия;
- външни API интеграции.

4. Инсталация и деинсталация

Изисквания:

- Visual Studio 2022

- .NET 8.0
- SQL Server

Инсталация:

1. Клониране на проекта от GitHub
2. Отваряне на **`Spotify2.sln`**
3. Настройване на connection string в **`appsettings.json`**
4. Стартиране на миграциите
5. Стартиране на приложението

Деинсталация:
Изтриване на проекта и базата данни.

5. Основни етапи при създаване на проекта

- Проектиране на базата данни
- Създаване на модели и контекст
- Имплементация на контролери
- Реализация на уеб интерфейс
- Добавяне на авторизация и роли
- Тестване и документация

6. Реализация

6.1. Архитектура

Проектът е ASP.NET Core MVC уеб приложение с ясно разделение на отговорностите между моделите, контролерите и изгледите.

6.2. База данни

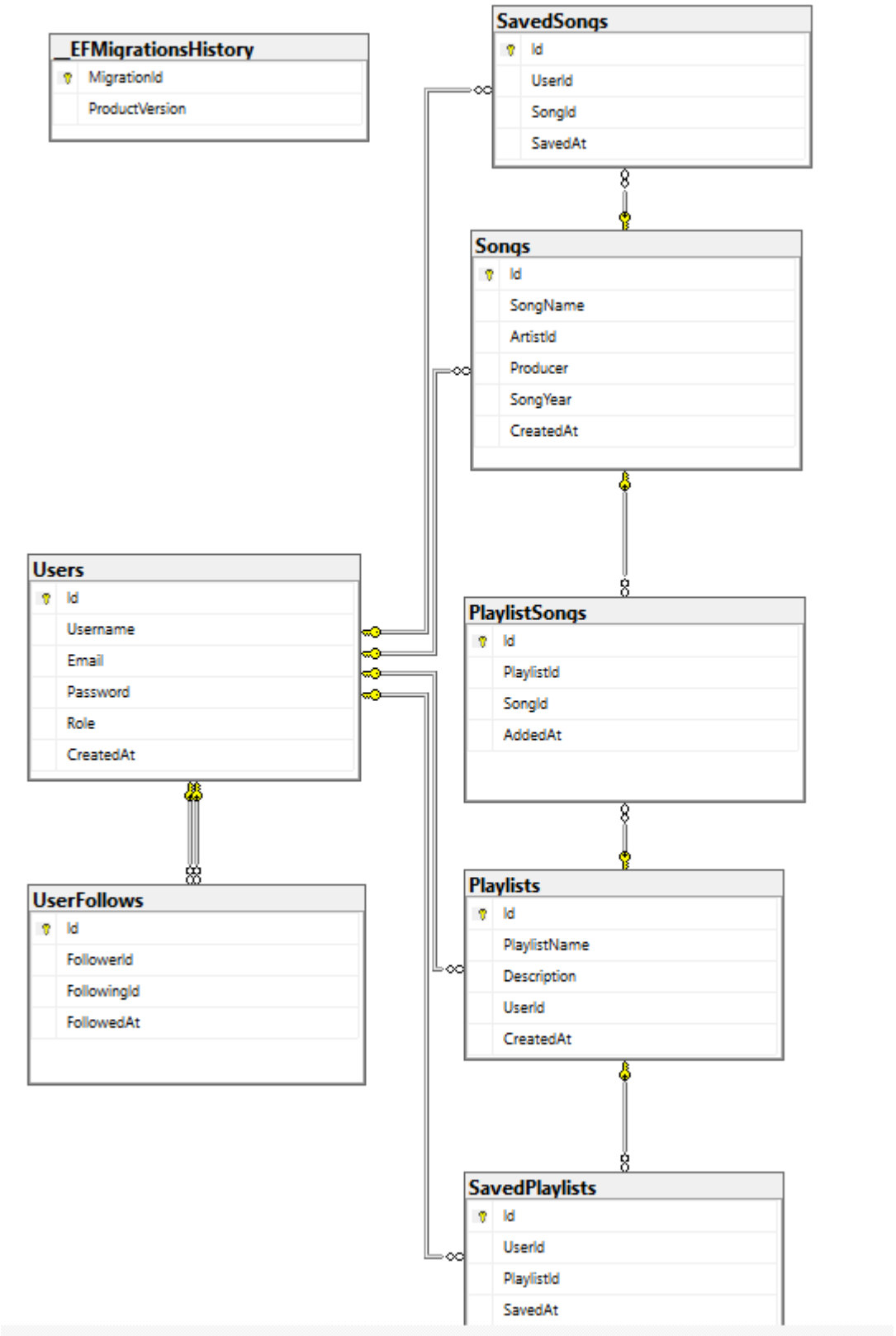
Основни ентитети:

- **User**
- **Song**
- **Playlist**
- **PlaylistSong** (many-to-many)
- **SavedSong**
- **SavedPlaylist**
- **UserFollow**

Налице е връзка **много към много** между:

- Songs ↔ Playlists

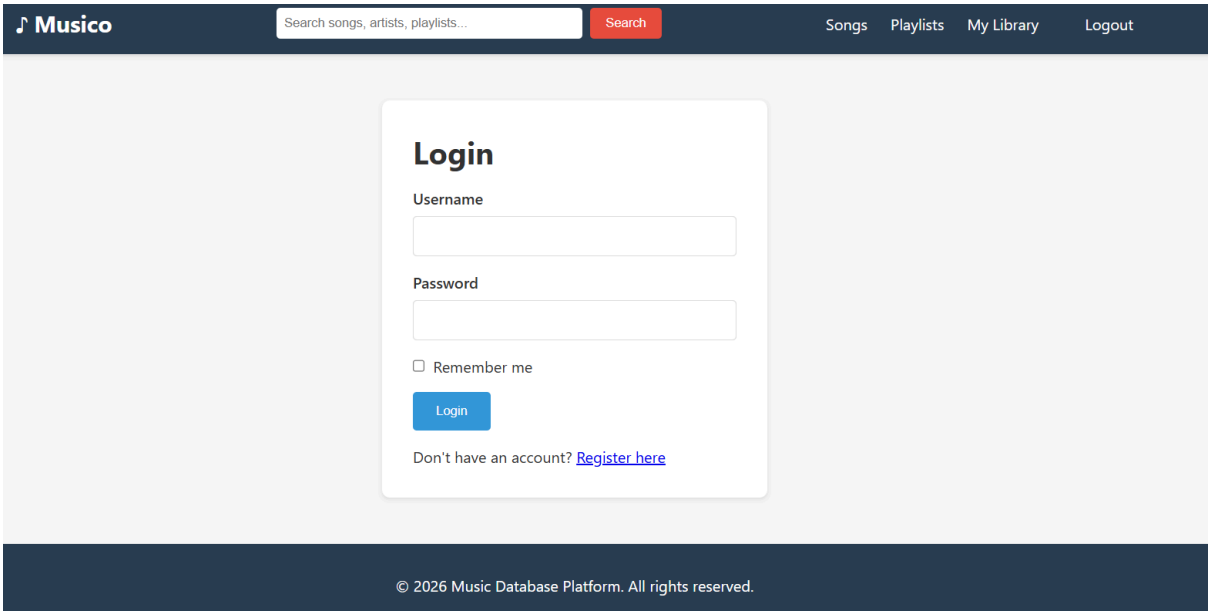
ER диаграма



6.3. Описание на приложението

Вход в системата: Потребителите влизат чрез уникално потребителско име и парола.

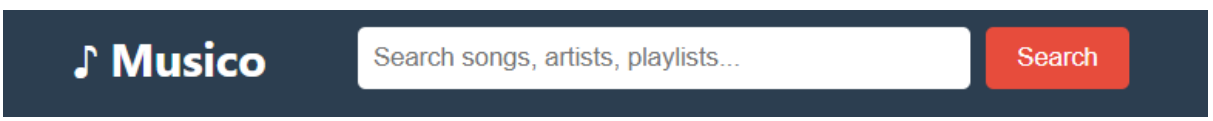
Регистрация: Системата предлага бърза връзка към форма за регистрация за нови потребители.



Класът User е основният модел в приложението, който дефинира структурата на потребителските данни и техните връзки в базата данни.

Контролерът AccountController реализира пълната логика за идентификация и оторизация на потребителите. Той осигурява сигурен преход между публичната част на сайта и личните функции на платформата.
След успешна идентификация, контролерът съхранява Username, Role и UserId в HttpContext.Session.

- **Потребителите могат да разглеждат и търсят песни**



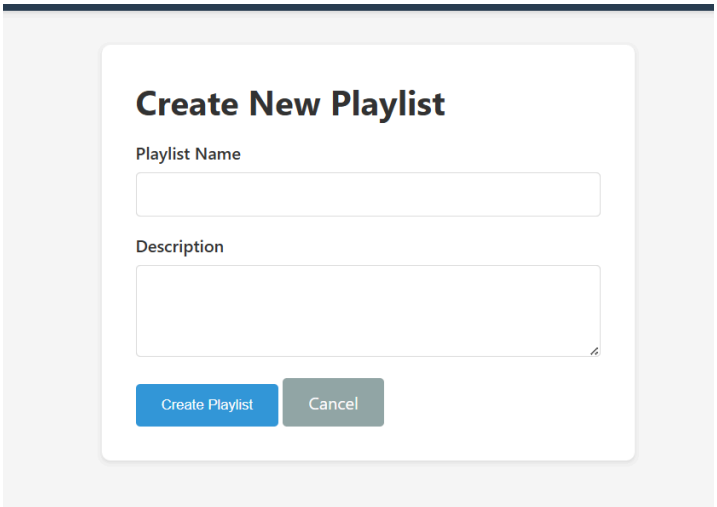
Контролерът SearchController реализира функционалността за глобално търсене в платформата.

Търсенето се извършва чрез динамично филтриране на няколко потока от данни:

Songs: Търси съвпадения в заглавията на песните SongName, като автоматично включва информация за техните изпълнители чрез .Include(s => s.Artist).

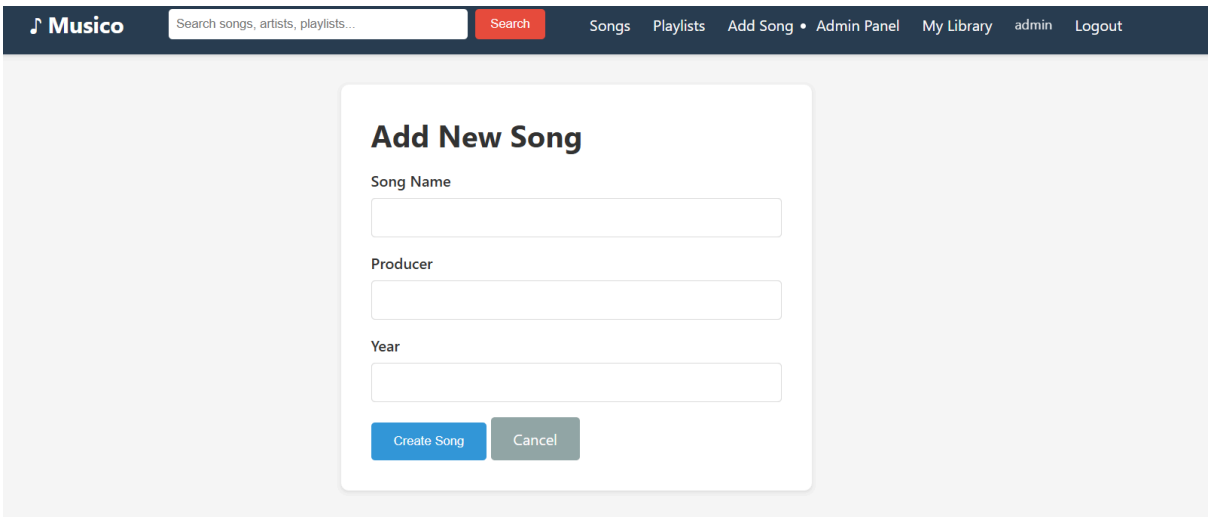
Playlists: Търси съвпадения в имената на плейлистите, като едновременно изчислява и броя на песните в тях чрез .Include(p => p.PlaylistSongs).

- **Създаване и управление на плейлисти**



Контролерът PlaylistsController отговаря за създаването, редактирането и управлението на потребителски колекции от песни. Той реализира връзката „Много към Много“ между таблиците Songs и Playlists чрез междинната таблица PlaylistSongs.

- **Добавяне на песни**



Контролерът SongsController управлява основния жизнен цикъл на музикалното съдържание в приложението – от визуализацията на наличните песни до тяхното създаване, редактиране и изтриване (CRUD операции). Системата стриктно следи правата на достъп. Само потребители с роля "Artist" или "Admin" имат право да използват методите Create, Edit и Delete.

- **Различни изгледи според ролята**

6.4. Използвани технологии

- ASP.NET Core MVC

  ASP.NET Core MVC е богата рамка за изграждане на уеб приложения и API, използвайки шаблона за проектиране Model-View-Controller.

  Архитектурният модел Model-View-Controller (MVC) разделя приложението на три основни групи компоненти: Модели, Изгледи и Контролери. Този модел помага за постигане на разделяне на задачите . Използвайки този модел, потребителските заявки се насочват към Контролер, който е отговорен за

работата с Модела за извършване на потребителски действия и/или извличане на резултати от заявки. Контролерът избира Изгледа, който да покаже на потребителя, и му предоставя всички данни за Модела, които са му необходими.

- Entity Framework Core

  Позволява на .NET разработчиците да работят с база данни, използвайки .NET обекти.Елиминира необходимостта от по-голямата част от кода за достъп до данни, който обикновено трябва да бъде написан.

  С EF Core достъпът до данни се осъществява с помощта на модел. Моделът е съставен от класове обекти и контекстен обект, който представлява сесия с базата данни. Контекстният обект позволява заявки и запазване на данни.

  EF поддържа следните подходи за разработване на модели:

  - ❖ Генериране на модел от съществуваща база данни.
  - ❖ Напишете ръчен код на модел, който да съответства на базата данни.
  - ❖ След като моделът е създаден, използвайте EF Migrations, за да създадете база данни от модела. Миграциите позволяват развитието на базата данни с промяната на модела.

- Microsoft SQL Server

  Microsoft SQL Server е собствена система за управление на релационни бази данни, разработена от Microsoft, използваща Structured Query Language

(SQL). Като сървър за бази данни , той е софтуерен продукт с основната функция да съхранява и извлича данни, изисквани от други софтуерни приложения , които могат да работят или на същия компютър, или на друг компютър в мрежа.

Представлява колекция от таблици с типизирани колони. Основният начин за извличане на данни от база данни на SQL Server е заявката за тях. Заявката декларативно указва какво трябва да се извлече. Тя се обработва от процесора за заявки, който определя последователността от стъпки, необходими за извличане на заявените данни.

- Bootstrap

  Библиотека за дизайн (CSS framework), която прави интерфейса на приложението модерен и отзивчив. Основната цел на добавянето ѝ към уеб проект е да се приложат изборите на Bootstrap за цвят, размер, шрифт и оформление към този проект.Благодарение на нея проектът изглежда добре както на компютър, така и на мобилни устройства.

- GitHub

  GitHub е собствена платформа за разработчици, която позволява на разработчиците да създават, съхраняват, управляват и споделят своя код. Тя използва Git, за да осигури разпределен контрол на версиите, а самият GitHub предоставя контрол на достъпа,проследяване на грешки,заявка за софтуерни функции управление на задачи,непрекъсната интеграция и уикита за всеки проект.

6.5. Използвани езици за програмиране

- **C#** – backend логика и модели

  Езикът C# е най-популярният език за платформата .NET , безплатна, междуплатформена среда за разработка с отворен код.C# е от семейството езици C. Синтаксисът на C# е познат, ако сте използвали C, C++, JavaScript, TypeScript или Java. Подобно на C и C++, точката и запетаята (;) определят края на операторите. Идентификаторите на C# са чувствителни към главни и малки букви. C# използва по същия начин скоби, {както и }, управляващи оператори като if, else и switch, и циклични конструкции като for и while. C# също има foreach оператор за всеки тип колекция.

- **SQL** – чрез Entity Framework

- **HTML / CSS** – уеб интерфейс
  - ❖ HTML е стандартният език за маркиране на уеб страници. Уеб браузърите получават HTML документи от уеб сървър или от локално хранилище и ги превръщат в мултимедийни уеб страници. HTML описва семантично структурата на уеб страницата и първоначално включва сигнали за нейния външен вид.
  - ❖ CSS е основна технология на World Wide Web , наред с HTML и JavaScript. Проектиран е да позволи разделянето на съдържанието и представянето , включително оформление , цветове и шрифтове. CSS също така има правила за алтернативно форматиране, ако достъпът до съдържанието се осъществява на мобилно устройство .

- **JavaScript** – базова клиентска логика

  JavaScript е използван за поведение на уеб страниците от страна на клиента. Уеб браузърите имат специален JavaScript енджин , който изпълнява клиентския код. Тези енджин - и се използват и в някои сървъри и различни приложения . Най-популярната система за изпълнение за употреба извън браузър е Node.js. Той има динамично типизиране, обектно-базирана ориентация и първокласни функции. Той е мулти парадигмален, поддържайки стилове на програмиране, управлявани от събития, функционални и императивни. Разполага с интерфейси за приложно програмиране (API) за работа с текст, дати, регулярни изрази, стандартни структури от данни и модела на обектите на документи (DOM).

7. Система за удостоверяване и роли

Реализирана е система за:

- регистрация и вход;
- различни нива на достъп;
- потребителски профили;
- администраторски функционалности.

8. Тестване

Проектът може да бъде разширен с unit тестове за валидиране на CRUD операциите и бизнес логиката.

9. Развитие и бъдещи подобрения

- персонализирани препоръки;
- социални функции;
- мобилно приложение;
- подобрена сигурност.