**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Доцент департамента программной инженерии факультета компьютерных наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Д. Брейман  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |

|  |  |
| --- | --- |
| **Подп. и дата** |  |
| **Инв № дубл** |  |
| **Взам. инв. №** |  |
| **Подп. и дата** |  |
| **Инв. № подл** |  |

**Андроид приложение IndieWindy**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.01-01 ТЗ 01-1-ЛУ**

**Исполнитель**

Студент группы БПИ173

\_\_\_\_\_\_\_\_\_\_\_\_ / С. И. Ройтман /

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**УТВЕРЖДЕНО**

**RU.17701729.04.01-01 ТЗ 01-1-ЛУ**

**Андроид приложение IndieWindy**

**Текст программы**

**RU.17701729.04.01-01 ТЗ 01-1-ЛУ**

|  |  |
| --- | --- |
| **Подп. и дата** |  |
| **Инв № дубл** |  |
| **Взам. инв. №** |  |
| **Подп. и дата** |  |
| **Инв. № подл** |  |

**Листов TODO**

Оглавление

[1. ВВЕДЕНИЕ 4](#_Toc37423816)

[1.1. Наименование программы 4](#_Toc37423817)

[1.2. Краткая характеристика 4](#_Toc37423818)

[2. ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ 4](#_Toc37423819)

[3. ПРИЛОЖЕНИЕ 5](#_Toc37423820)

[3.1. ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ 5](#_Toc37423821)

[4. ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 8](#_Toc37423822)

1. ТЕКСТЫ ИСХОДНЫХ ФАЙЛОВ
   1. Сервер:
      1. DatabaseAPI
         1. IndieWindyDbContext.cs

using DatabaseAPI.Models;

using Microsoft.EntityFrameworkCore;

namespace DatabaseAPI

{

/\* Migrations commands: from DatabaseAPI folder

dotnet ef --startup-project ../WebAPI/ migrations add \*name\*

dotnet ef --startup-project ../WebAPI/ database update

\*/

public class IndieWindyDbContext:DbContext

{

public static string ConnectionString;

public IndieWindyDbContext(DbContextOptions<IndieWindyDbContext> options) : base(options)

{

ConnectionString = Database.GetDbConnection()?.ConnectionString;

}

public DbSet<AppUser> AppUser { get; set; }

public DbSet<Artist> Artist { get; set; }

public DbSet<Album> Album { get; set; }

public DbSet<Song> Song { get; set; }

public DbSet<Concert> Concert { get; set; }

public DbSet<Donation> Donation { get; set; }

public DbSet<Post> Post { get; set; }

public DbSet<LatestPost> LatestPost { get; set; }

public DbSet<ArtistConcertLink> ArtistConcertLink { get; set; }

public DbSet<ArtistPostLink> ArtistPostLink { get; set; }

public DbSet<PostSongLink> PostSongLink { get; set; }

public DbSet<UserArtistLink> UserArtistLink { get; set; }

public DbSet<UserAlbumLink> UserAlbumLink { get; set; }

public DbSet<UserSongLink> UserSongLink { get; set; }

public DbSet<UserConcertLink> UserConcertLink { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<ArtistConcertLink>()

.HasKey(i => new { i.ArtistId, i.ConcertId });

modelBuilder.Entity<ArtistPostLink>()

.HasKey(i => new { i.ArtistId, i.PostId });

modelBuilder.Entity<PostSongLink>()

.HasKey(i => new { i.PostId, i.SongId });

modelBuilder.Entity<UserAlbumLink>()

.HasKey(i => new { i.AppUserId, i.AlbumId });

modelBuilder.Entity<UserArtistLink>()

.HasKey(i => new { i.AppUserId, i.ArtistId });

modelBuilder.Entity<UserConcertLink>()

.HasKey(i => new { i.AppUserId, i.ConcertId });

modelBuilder.Entity<UserSongLink>()

.HasKey(i => new { i.AppUserId, i.SongId });

}

}

}

* + - 1. Файл models/BaseEntity.cs

namespace DatabaseAPI.Models

{

public class BaseEntity

{

}

}

* + - 1. Файл models/AppUser.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("app\_user")]

public class AppUser : BaseEntity

{

[Key]

[Column("id")]

public int Id { get; set; }

[Column("name")]

public string Name { get; set; }

[Column("password")]

public string Password { get; set; }

}

}

* + - 1. Файл models/Artist.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("artist")]

public class Artist : BaseEntity

{

[Key]

[Column("id")]

public int Id { get; set; }

[Column("name")]

public string Name { get; set; }

[Column("description")]

public string Description { get; set; }

[Column("image\_url")]

public string ImageUrl { get; set; }

}

}

* + - 1. Файл models/ArtistConcertLink.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("artist\_concert\_link")]

public class ArtistConcertLink : BaseEntity

{

[Key]

[Column("artist\_id", Order=1)]

public int ArtistId { get; set; }

[ForeignKey("ArtistId")]

public Artist Artist { get; set; }

[Key]

[Column("concert\_id", Order=2)]

public int ConcertId { get; set; }

[ForeignKey("ConcertId")]

public Concert Concert { get; set; }

}

}

* + - 1. Файл models/ArtistPostLink.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("artist\_post\_link")]

public class ArtistPostLink : BaseEntity

{

[Key]

[Column("artist\_id", Order=1)]

public int ArtistId { get; set; }

[ForeignKey("ArtistId")]

public Artist Artist { get; set; }

[Key]

[Column("post\_id", Order=2)]

public int PostId { get; set; }

[ForeignKey("PostId")]

public Post Post { get; set; }

}

}

* + - 1. Файл models/Album.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("album")]

public class Album : BaseEntity

{

[Key]

[Column("id")]

public int Id { get; set; }

[Column("name")]

public string Name { get; set; }

[Column("image\_url")]

public string ImageUrl { get; set; }

[Column("artist\_id")]

public int ArtistId { get; set; }

[ForeignKey("ArtistId")]

public Artist Artist { get; set; }

}

}

* + - 1. Файл models/Concert.cs

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("concert")]

public class Concert : BaseEntity

{

[Key]

[Column("id")]

public int Id { get; set; }

[Column("name")]

public string Name { get; set; }

[Column("image\_url")]

public string ImageUrl { get; set; }

[Column("cost")]

public int Cost { get; set; }

[Column("start\_time")]

public DateTime StartTime { get; set; }

[Column("address")]

public string Address { get; set; }

[Column("ticket\_link")]

public string TicketLink { get; set; }

[Column("description")]

public string Description { get; set; }

}

}

* + - 1. Файл models/Donation.cs

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("donation")]

public class Donation : BaseEntity

{

[Key]

[Column("id", Order=1)]

public int Id { get; set; }

[Column("app\_user\_id")]

public int AppUserId { get; set; }

[ForeignKey("AppUserId")]

public AppUser AppUser { get; set; }

[Column("artist\_id")]

public int ArtistId { get; set; }

[ForeignKey("ArtistId")]

public Artist Artist { get; set; }

[Column("amount")]

public int Amount { get; set; }

[Column("date")]

public DateTime Date { get; set; }

}

}

* + - 1. Файл models/LatestPost.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("latest\_post")]

public class LatestPost : BaseEntity

{

[Key]

[Column("id")]

public int Id { get; set; }

[Column("post\_id")]

public int PostId { get; set; }

}

}

* + - 1. Файл models/Post.cs

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("post")]

public class Post : BaseEntity

{

[Key]

[Column("id")]

public int Id { get; set; }

[Column("text")]

public string Text { get; set; }

[Column("time")]

public DateTime Time { get; set; }

[Column("artist\_id")]

public int ArtistId { get; set; }

[ForeignKey("ArtistId")]

public Artist Artist { get; set; }

}

}

* + - 1. Файл models/ PostSongLink.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("post\_song\_link")]

public class PostSongLink : BaseEntity

{

[Key]

[Column("post\_id", Order=1)]

public int PostId { get; set; }

[ForeignKey("PostId")]

public Post Post { get; set; }

[Key]

[Column("song\_id", Order=2)]

public int SongId { get; set; }

[ForeignKey("SongId")]

public Song Song { get; set; }

}

}

* + - 1. Файл models/Song.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("song")]

public class Song : BaseEntity

{

[Key]

[Column("id")]

public int Id { get; set; }

[Column("name")]

public string Name { get; set; }

[Column("song\_url")]

public string SongUrl { get; set; }

[Column("artist\_id")]

public int ArtistId { get; set; }

[ForeignKey("ArtistId")]

public Artist Artist { get; set; }

[Column("album\_id")]

public int AlbumId { get; set; }

[ForeignKey("AlbumId")]

public Album Album { get; set; }

}

}

* + - 1. Файл models/UserAlbumLink.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("user\_album\_link")]

public class UserAlbumLink : BaseEntity

{

public UserAlbumLink() {}

public UserAlbumLink(int userId, int albumId)

{

AppUserId = userId;

AlbumId = albumId;

}

[Key]

[Column("app\_user\_id", Order=1)]

public int AppUserId { get; }

[ForeignKey("AppUserId")]

public AppUser AppUser { get; set; }

[Key]

[Column("album\_id", Order=2)]

public int AlbumId { get; }

[ForeignKey("AlbumId")]

public Album Album { get; set; }

}

}

* + - 1. Файл models/ UserArtistLink.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("user\_artist\_link")]

public class UserArtistLink : BaseEntity

{

public UserArtistLink() { }

public UserArtistLink(int appUserId, int artistId)

{

AppUserId = appUserId;

ArtistId = artistId;

}

[Key]

[Column("app\_user\_id", Order=1)]

public int AppUserId { get; }

[ForeignKey("AppUserId")]

public AppUser AppUser { get; set; }

[Key]

[Column("artist\_id", Order=2)]

public int ArtistId { get; }

[ForeignKey("ArtistId")]

public Artist Artist { get; set; }

}

}

* + - 1. Файл models/ UserConcertLink.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("user\_concert\_link")]

public class UserConcertLink : BaseEntity

{

public UserConcertLink()

{

}

public UserConcertLink(int appUserId, int concertId)

{

AppUserId = appUserId;

ConcertId = concertId;

}

[Key]

[Column("app\_user\_id", Order=1)]

public int AppUserId { get; }

[ForeignKey("AppUserId")]

public AppUser AppUser { get; set; }

[Key]

[Column("concert\_id", Order=2)]

public int ConcertId { get; }

[ForeignKey("ConcertId")]

public Concert Concert { get; set; }

}

}

* + - 1. Файл models/ UserSongLink.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DatabaseAPI.Models

{

[Table("user\_song\_link")]

public class UserSongLink : BaseEntity

{

public UserSongLink() {}

public UserSongLink(int appUserId, int songId)

{

AppUserId = appUserId;

SongId = songId;

}

[Key]

[Column("app\_user\_id", Order=1)]

public int AppUserId { get; }

[ForeignKey("AppUserId")]

public AppUser AppUser { get; set; }

[Key]

[Column("song\_id", Order=2)]

public int SongId { get; }

[ForeignKey("SongId")]

public Song Song { get; set; }

}

}

* + - 1. Файл services/DatabaseBaseService.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

namespace DatabaseAPI.Services

{

public class DatabaseBaseService<T> where T : class

{

protected readonly IndieWindyDbContext \_indieWindyDb;

public DatabaseBaseService(IndieWindyDbContext indieWindyDb)

{

\_indieWindyDb = indieWindyDb;

}

/// <summary>

/// Made not implemented for catching error

/// when user forgot to override this

/// </summary>

/// <returns></returns>

public List<T> GetAll()

{

return \_indieWindyDb.Set<T>().ToList();

}

/// <summary>

/// Returns an item that was inserted in db

/// </summary>

/// <param name="item"></param>

/// <returns>null if no item was inserted</returns>

public virtual async Task<T> AddNewItem(T item)

{

var itemInDb = await \_indieWindyDb.AddAsync(item);

var success = await SaveChangesAsync();

return success ? itemInDb.Entity : null;

}

public virtual async Task<Boolean> DeleteItem<T>(T item)

{

try

{

\_indieWindyDb.Remove(item);

var success = await SaveChangesAsync();

return success;

}

catch (DbUpdateConcurrencyException )

{

// No element was removed

return false;

}

}

/// <summary>

/// Returns item found by it's identifier

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public async Task<T> FindItemByIdAsync(string id)

{

return await \_indieWindyDb.FindAsync<T>(id);

}

/// <summary>

/// Returns true if delete of item is successful. Also updates the context

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public async Task<bool> DeleteItemById(string id)

{

var item = await FindItemByIdAsync(id);

\_indieWindyDb.Remove(item);

return await SaveChangesAsync();

}

/// <summary>

/// Returns true if updating context is successful

/// </summary>

/// <returns></returns>

public async Task<bool> SaveChangesAsync()

{

try

{

await \_indieWindyDb.SaveChangesAsync();

return true;

}

catch (Exception e)

{

Console.Error.WriteLine("SaveChangesAsync: error: " + e.Message);

return false;

}

}

}

}

* + - 1. Файл services/AppUserService.cs

using System.Threading.Tasks;

using DatabaseAPI.Models;

using DatabaseAPI.Utils;

using Microsoft.EntityFrameworkCore;

namespace DatabaseAPI.Services

{

public class AppUserService : DatabaseBaseService<AppUser>

{

public AppUserService(IndieWindyDbContext indieWindyDb) : base(indieWindyDb) { }

public override async Task<AppUser> AddNewItem(AppUser user)

{

user.Password = PasswordHasherService.GetMd5Hash(user.Password);

return await base.AddNewItem(user);

}

public async Task<AppUser> FindByName(string name)

{

return await \_indieWindyDb.AppUser.

SingleOrDefaultAsync(u => u.Name.Equals(name));

}

}

}

* + - 1. Файл services/ArtistService.cs

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Dapper;

using DatabaseAPI.Models;

using DatabaseAPI.Utils;

using Microsoft.EntityFrameworkCore;

using Npgsql;

namespace DatabaseAPI.Services

{

public class ArtistService : DatabaseBaseService<Artist>

{

public ArtistService(IndieWindyDbContext indieWindyDb) : base(indieWindyDb) { }

public async Task<List<UserArtistLink>> FindByName(string query, int userId)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserArtistLink, Artist, UserArtistLink>(

@"select link.\*, a.\*

from user\_artist\_link as link

right join artist a on link.artist\_id = a.id and link.app\_user\_id = @user

where lower(a.name) like @name",

(link, artist) =>

{

link.Artist = artist;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

public async Task<List<UserArtistLink>> FindByNameLinked(string query, int userId)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

// TODO sort

var res = await con.QueryAsync<UserArtistLink, Artist, UserArtistLink>(

@"select link.\*, a.\*

from user\_artist\_link as link

right join artist a on link.artist\_id = a.id and link.app\_user\_id = @user

where lower(a.name) like @name and link.app\_user\_id is not null",

(link, artist) =>

{

link.Artist = artist;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

public async Task<List<UserAlbumLink>> GetAlbums(int artistId, int userId)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserAlbumLink, Album, UserAlbumLink>(

@"select \* from user\_album\_link as link

right join album a on link.album\_id = a.id and link.app\_user\_id = @userId

where a.artist\_id = @artistId",

(link, album) =>

{

link.Album = album;

return link;

},

param: new {artistId, userId});

return res.ToList();

}

}

}

* + - 1. Файл services/ConcertService.cs

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Dapper;

using DatabaseAPI.Models;

using Npgsql;

namespace DatabaseAPI.Services

{

public class ConcertService : DatabaseBaseService<Concert>

{

public ConcertService(IndieWindyDbContext indieWindyDb) : base(indieWindyDb) { }

public async Task<List<UserConcertLink>> FindByName(string query, int userId)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserConcertLink, Concert, UserConcertLink>(

@"select link.\*, c.\*

from user\_concert\_link as link

right join concert c on link.concert\_id = c.id and link.app\_user\_id = @user

where lower(c.name) like @name",

(link, concert) =>

{

link.Concert = concert;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

public async Task<List<UserConcertLink>> GetNearest(int userId, string query)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserConcertLink, Concert, UserConcertLink>(

@"select link.\*, c.\*

from user\_concert\_link as link

right join concert c on link.concert\_id = c.id and link.app\_user\_id = @user

where start\_time > now() and lower(c.name) like @name

order by c.start\_time",

(link, concert) =>

{

link.Concert = concert;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

public async Task<List<UserConcertLink>> GetNearestByArtist(int userId, string query)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserConcertLink, Concert, UserConcertLink>(

@"select link.\*, c.\*

from user\_concert\_link as link

right join concert c on link.concert\_id = c.id and link.app\_user\_id = @user

where c.start\_time > now() and c.id in

(select distinct acl.concert\_id from artist\_concert\_link acl

join artist a on acl.artist\_id = a.id

where lower(a.name) like @name)

order by c.start\_time;",

(link, concert) =>

{

link.Concert = concert;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

public async Task<List<UserConcertLink>> GetBySubscription(int userId, string query)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserConcertLink, Concert, UserConcertLink>(

@"select link.\*, c.\*

from user\_concert\_link link

right join concert c on link.concert\_id = c.id and app\_user\_id = @user

where lower(c.name) like @name and c.id in

(select distinct concert\_id from artist\_concert\_link

where artist\_id in

(select artist\_id from user\_artist\_link

where app\_user\_id = @user))

order by c.start\_time;",

(link, concert) =>

{

link.Concert = concert;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

public async Task<List<UserConcertLink>> GetBySubscriptionByArtist(int userId, string query)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserConcertLink, Concert, UserConcertLink>(

@"select link.\*, c.\*

from user\_concert\_link link

right join concert c on link.concert\_id = c.id and app\_user\_id = @user

where c.id in

(select distinct concert\_id from artist\_concert\_link

where artist\_id in

(select artist\_id from user\_artist\_link

join artist a on user\_artist\_link.artist\_id = a.id

where app\_user\_id = @user and lower(a.name) like @name))

order by c.start\_time;",

(link, concert) =>

{

link.Concert = concert;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

public async Task<List<UserConcertLink>> GetSaved(int userId, string query)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserConcertLink, Concert, UserConcertLink>(

@"select link.\*, c.\*

from user\_concert\_link link

right join concert c on link.concert\_id = c.id and app\_user\_id = @user

where link.app\_user\_id is not null and lower(c.name) like @name

order by c.start\_time;",

(link, concert) =>

{

link.Concert = concert;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

public async Task<List<UserConcertLink>> GetSavedByArtist(int userId, string query)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserConcertLink, Concert, UserConcertLink>(

@"select link.\*, c.\*

from user\_concert\_link link

right join concert c on link.concert\_id = c.id and app\_user\_id = @user

where link.app\_user\_id is not null and c.id in

(select distinct acl.concert\_id from artist\_concert\_link acl

join artist a on acl.artist\_id = a.id

where lower(a.name) like @name)

order by c.start\_time;",

(link, concert) =>

{

link.Concert = concert;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

public async Task<List<UserArtistLink>> GetArtists(int userId, int concertId)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserArtistLink, Artist, UserArtistLink>(

@"select link.\*, a.\*

from user\_artist\_link link

right join artist\_concert\_link acl

on link.artist\_id = acl.artist\_id and link.app\_user\_id = @user

join artist a on acl.artist\_id = a.id

where acl.concert\_id = @concert;",

(link, artist) =>

{

link.Artist = artist;

return link;

},

param: new {@user = userId, @concert = concertId});

return res.ToList();

}

}

}

* + - 1. Файл services/AlbumService.cs

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Dapper;

using DatabaseAPI.Models;

using DatabaseAPI.Utils;

using Microsoft.EntityFrameworkCore;

using Npgsql;

namespace DatabaseAPI.Services

{

public class AlbumService : DatabaseBaseService<Album>

{

public AlbumService(IndieWindyDbContext indieWindyDb) : base(indieWindyDb)

{ }

public async Task<List<UserAlbumLink>> FindByName(string query, int userId)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserAlbumLink, Album, Artist, UserAlbumLink>(

@"select link.\*, al.\*, a.\*

from user\_album\_link as link

right join album al on link.album\_id = al.id and link.app\_user\_id = @user

join artist a on al.artist\_id = a.id

where lower(al.name) like @name",

(link, album, artist) =>

{

album.Artist = artist;

link.Album = album;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

public async Task<List<UserAlbumLink>> FindByNameLinked(string query, int userId)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

// TODO sort

var res = await con.QueryAsync<UserAlbumLink, Album, Artist, UserAlbumLink>(

@"select link.\*, al.\*, a.\*

from user\_album\_link as link

right join album al on link.album\_id = al.id and link.app\_user\_id = @user

join artist a on al.artist\_id = a.id

where lower(al.name) like @name and link.app\_user\_id is not null",

(link, album, artist) =>

{

album.Artist = artist;

link.Album = album;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

public async Task<List<UserSongLink>> GetSongs(int albumId, int userId)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserSongLink, Song, Artist, Album, UserSongLink>(

@"select \* from user\_song\_link as link

right join song s on link.song\_id = s.id and link.app\_user\_id = @userId

join artist a on s.artist\_id = a.id

join album al on s.album\_id = al.id

where s.album\_id = @albumId",

(link, song, artist, album) =>

{

album.Artist = artist;

song.Album = album;

song.Artist = artist;

link.Song = song;

return link;

},

param: new {albumId, userId});

return res.ToList();

}

}

}

* + - 1. Файл services/DonationService.cs

using System;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using Microsoft.EntityFrameworkCore;

namespace DatabaseAPI.Services

{

public class DonationService : DatabaseBaseService<Donation>

{

public DonationService(IndieWindyDbContext indieWindyDb) : base(indieWindyDb) { }

}

}

* + - 1. Файл services/PostService.cs

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Dapper;

using DatabaseAPI.Models;

using Npgsql;

namespace DatabaseAPI.Services

{

public class PostService : DatabaseBaseService<Concert>

{

public PostService(IndieWindyDbContext indieWindyDb) : base(indieWindyDb) { }

public async Task<List<ArtistPostLink>> GetBySubscription(int userId)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<ArtistPostLink, Post, Artist, ArtistPostLink>(

@"select apl.\*, p.\*, a.\* from artist\_post\_link apl

join post p on apl.post\_id = p.id

join artist a on apl.artist\_id = a.id

where apl.artist\_id in

(select user\_artist\_link.artist\_id from user\_artist\_link

where app\_user\_id = @user)

order by p.time desc;",

(link, post, artist) =>

{

link.PostId = post.Id;

link.Post = post;

link.ArtistId = artist.Id;

link.Artist = artist;

return link;

},

param: new {@user = userId});

return res.ToList();

}

public async Task<List<ArtistPostLink>> GetLatest()

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<ArtistPostLink, Post, Artist, ArtistPostLink>(

@"select apl.\*, p.\*, a.\* from latest\_post lp

join post p on lp.post\_id = p.id

join artist a on p.artist\_id = a.id

join artist\_post\_link apl on apl.artist\_id = a.id and apl.post\_id = p.id

order by lp.id desc;",

(link, post, artist) =>

{

link.PostId = post.Id;

link.Post = post;

link.ArtistId = artist.Id;

link.Artist = artist;

return link;

});

return res.ToList();

}

public async Task<List<UserSongLink>> GetSongs(int userId, int postId)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserSongLink, Song, Artist, Album, UserSongLink>(

@"select usl.\*, s.\*, a.\*, al.\*

from post\_song\_link psl

join song s on psl.song\_id = s.id

join artist a on s.artist\_id = a.id

join album al on s.artist\_id = al.id

left join user\_song\_link usl on s.id = usl.song\_id and usl.app\_user\_id = @userId

where psl.post\_id = @postId;",

(link, song, artist, album) =>

{

song.Album = album;

song.Artist = artist;

link.Song = song;

return link;

},

param: new {userId, postId});

return res.ToList();

}

}

}

* + - 1. Файл services/SongService.cs

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Dapper;

using DatabaseAPI.Models;

using DatabaseAPI.Utils;

using Microsoft.EntityFrameworkCore;

using Npgsql;

namespace DatabaseAPI.Services

{

public class SongService : DatabaseBaseService<Song>

{

public SongService(IndieWindyDbContext indieWindyDb) : base(indieWindyDb)

{ }

public async Task<List<Song>> FindByName(string query)

{

var songs = await \_indieWindyDb.Song

.Include(s => s.Artist)

.Include(s => s.Album)

.ToListAsync();

return songs.Where(s => SearchUtil.StartsWith(s.Name, query)).ToList();

}

public async Task<List<UserSongLink>> FindByName(string query, int userId)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

var res = await con.QueryAsync<UserSongLink, Song, Artist, Album, UserSongLink>(

@"select link.\*, s.\*, a.\*, al.\*

from user\_song\_link as link

right join song s on link.song\_id = s.id and link.app\_user\_id = @user

join artist a on s.artist\_id = a.id

join album al on s.album\_id = al.id

where lower(s.name) like @name",

(link, song, artist, album) =>

{

song.Artist = artist;

song.Album = album;

album.Artist = artist;

link.Song = song;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

public async Task<List<UserSongLink>> FindByNameLinked(string query, int userId)

{

await using var con = new NpgsqlConnection(IndieWindyDbContext.ConnectionString);

// TODO sort by position

var res = await con.QueryAsync<UserSongLink, Song, Artist, Album, UserSongLink>(

@"select link.\*, s.\*, a.\*, al.\*

from user\_song\_link as link

right join song s on link.song\_id = s.id and link.app\_user\_id = @user

join artist a on s.artist\_id = a.id

join album al on s.album\_id = al.id

where lower(s.name) like @name and link.app\_user\_id is not null",

(link, song, artist, album) =>

{

song.Artist = artist;

song.Album = album;

album.Artist = artist;

link.Song = song;

return link;

},

param: new {@name = $"{query.ToLower()}%", @user = userId});

return res.ToList();

}

}

}

* + - 1. Файл services/UserAlbumLinkService.cs

using System;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using Microsoft.EntityFrameworkCore;

namespace DatabaseAPI.Services

{

public class UserAlbumLinkService: DatabaseBaseService<UserAlbumLink>

{

public UserAlbumLinkService(IndieWindyDbContext indieWindyDb) : base(indieWindyDb) { }

public async Task<Boolean> LinkExist(int userId, int albumId)

{

return await \_indieWindyDb.UserAlbumLink

.AnyAsync(l => l.AppUserId == userId && l.AlbumId == albumId);

}

}

}

* + - 1. Файл services/UserArtistLinkService.cs

using System;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using Microsoft.EntityFrameworkCore;

namespace DatabaseAPI.Services

{

public class UserArtistLinkService : DatabaseBaseService<UserArtistLink>

{

public UserArtistLinkService(IndieWindyDbContext indieWindyDb) : base(indieWindyDb) { }

public async Task<Boolean> LinkExist(int userId, int artistId)

{

return await \_indieWindyDb.UserArtistLink

.AnyAsync(l => l.AppUserId == userId && l.ArtistId == artistId);

}

}

}

* + - 1. Файл services/UserConcertLinkService.cs

using DatabaseAPI.Models;

namespace DatabaseAPI.Services

{

public class UserConcertLinkService: DatabaseBaseService<UserConcertLink>

{

public UserConcertLinkService(IndieWindyDbContext indieWindyDb) : base(indieWindyDb) {}

}

}

* + - 1. Файл services/UserSongLinkService.cs

using System;

using System.Linq;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using Microsoft.EntityFrameworkCore;

namespace DatabaseAPI.Services

{

public class UserSongLinkService: DatabaseBaseService<UserSongLink>

{

public UserSongLinkService(IndieWindyDbContext indieWindyDb) : base(indieWindyDb) { }

}

}

* + - 1. Файл utils/DapperMappingConfiguration.cs

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using Dapper;

using DatabaseAPI.Models;

namespace DatabaseAPI.Utils

{

public static class DapperMappingConfiguration

{

public static void Configure()

{

ConfigureType<Album>();

ConfigureType<AppUser>();

ConfigureType<Artist>();

ConfigureType<ArtistConcertLink>();

ConfigureType<ArtistPostLink>();

ConfigureType<Donation>();

ConfigureType<Concert>();

ConfigureType<Song>();

ConfigureType<Post>();

ConfigureType<PostSongLink>();

ConfigureType<UserAlbumLink>();

ConfigureType<UserArtistLink>();

ConfigureType<UserConcertLink>();

ConfigureType<UserSongLink>();

}

private static void ConfigureType<T>()

{

SqlMapper.SetTypeMap(typeof(T),

new CustomPropertyTypeMap(typeof(T),

(type, columnName) => type.GetProperties()

.FirstOrDefault(prop => prop.GetCustomAttributes(false)

.OfType<ColumnAttribute>()

.Any(attr => attr.Name == columnName))));

}

}

}

* + - 1. Файл utils/ PasswordHasherUtil.cs

using System;

using System.Security.Cryptography;

using System.Text;

namespace DatabaseAPI.Utils

{

public static class PasswordHasherService

{

public static MD5 \_md5Hasher = MD5.Create();

public static string GetMd5Hash(string input)

{

// Convert the input string to a byte array and compute the hash.

var dataBytes = \_md5Hasher.ComputeHash(Encoding.UTF8.GetBytes(input));

// Create a new Stringbuilder to collect the bytes

// and create a string.

var sBuilder = new StringBuilder();

// Loop through each byte of the hashed data

// and format each one as a hexadecimal string.

for (int i = 0; i < dataBytes.Length; i++)

{

sBuilder.Append(dataBytes[i].ToString("x2"));

}

// Return the hexadecimal string.

return sBuilder.ToString();

}

// Verify a hash against a string.

public static bool VerifyMd5Hash(string input, string hash, bool inputIsHashed = false)

{

// Hash the input.

var hashOfInput = inputIsHashed ? input : GetMd5Hash(input);

// Create a StringComparer an compare the hashes.

StringComparer comparer = StringComparer.OrdinalIgnoreCase;

if (0 == comparer.Compare(hashOfInput, hash))

{

return true;

}

else

{

return false;

}

}

}

}

* + - 1. Файл utils/SearchUtil.cs

using System;

namespace DatabaseAPI.Utils

{

public static class SearchUtil

{

public static bool StartsWith(string name, string query)

{

return name.IndexOf(query, StringComparison.OrdinalIgnoreCase) == 0;

}

}

}

* + 1. WebAPI
       1. Файл Program.cs

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

namespace WebAPI

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder => { webBuilder.UseStartup<Startup>(); });

}

}

* + - 1. Файл Startup.cs

using System;

using DatabaseAPI;

using DatabaseAPI.Services;

using DatabaseAPI.Utils;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

namespace WebAPI

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

// Database services

services.AddScoped<AppUserService>();

services.AddScoped<ArtistService>();

services.AddScoped<AlbumService>();

services.AddScoped<SongService>();

services.AddScoped<ConcertService>();

services.AddScoped<UserArtistLinkService>();

services.AddScoped<UserAlbumLinkService>();

services.AddScoped<UserSongLinkService>();

services.AddScoped<UserConcertLinkService>();

services.AddScoped<DonationService>();

services.AddScoped<PostService>();

services.AddControllers();

// Azure db in production, local db otherwise

String connectionString = Configuration.GetConnectionString(

Environment.GetEnvironmentVariable("ASPNETCORE\_ENVIRONMENT").Equals("Production") ?

"AzureConnection" : "LocalConnection");

services.AddEntityFrameworkNpgsql()

.AddDbContext<IndieWindyDbContext>(opt =>

opt.UseNpgsql(connectionString));

// Automatically perform database migration

services.BuildServiceProvider().GetService<IndieWindyDbContext>().Database.Migrate();

// Dapper mapping configuration

DapperMappingConfiguration.Configure();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints => { endpoints.MapControllers(); });

}

}

}

* + - 1. Файл appsettings.json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"ConnectionStrings": {

"LocalConnection": "User ID=postgres;Password=1234;Server=localhost;Port=5430;Database=indieWindyDB;Integrated Security=true;Pooling=true;",

"AzureConnection": "Server=indiewindydb.postgres.database.azure.com;Database=postgres;Port=5432;User Id=indie\_windy\_admin@indiewindydb;Password=enu872Tt;Ssl Mode=Require;"

}

}

* + - 1. Файл appsettings.Development.json

{

"Logging": {

"LogLevel": {

"Default": "Debug",

"System": "Information",

"Microsoft": "Information"

}

},

"DB" : {

"ConnectionStrings": "User ID=postgres;Password=1234;Server=localhost;Port=5430;Database=indieWindyDB;Integrated Security=true;Pooling=true;"

}

}

* + - 1. Файл properties/launchSettings.json

{

"$schema": "http://json.schemastore.org/launchsettings.json",

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://localhost:16366",

"sslPort": 44335

}

},

"profiles": {

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"launchUrl": "appuser",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"IndieWindyServer": {

"commandName": "Project",

"launchUrl": "appuser",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

},

"applicationUrl": "https://localhost:5001;http://localhost:5000"

}

}

}

* + - 1. Файл controllers/AlbumController.cs

using System.Collections.Generic;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using DatabaseAPI.Services;

using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class AlbumController : ControllerBase

{

private readonly AlbumService \_albumService;

public AlbumController(AlbumService albumService)

{

\_albumService = albumService;

}

[Route("find/{query}/{userId}")]

public async Task<List<UserAlbumLink>> FindByName(string query, int userId)

{

return await \_albumService.FindByName(query, userId);

}

[Route("findLinked/{query}/{userId}")]

public async Task<List<UserAlbumLink>> FindByNameLinked(string query, int userId)

{

query = query.Equals("null") ? "" : query;

return await \_albumService.FindByNameLinked(query, userId);

}

[HttpGet]

[Route("{userId}/{albumId}/songs")]

public async Task<List<UserSongLink>> GetSongs(int userId, int albumId)

{

return await \_albumService.GetSongs(albumId, userId);

}

}

}

* + - 1. Файл controllers/AppUserController.cs

using System.Collections.Generic;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using DatabaseAPI.Services;

using DatabaseAPI.Utils;

using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class AppUserController : ControllerBase

{

private readonly AppUserService \_appUserService;

public AppUserController(AppUserService appUserService)

{

\_appUserService = appUserService;

}

[HttpGet]

[Route("all")]

public IEnumerable<AppUser> GetAll()

{

return \_appUserService.GetAll();

}

[HttpPost]

[Route("register")]

public async Task<AppUser> CreateAppUser([FromBody] AppUser user)

{

var duplicateCheck = await \_appUserService.FindByName(user.Name);

return duplicateCheck == null ? await \_appUserService.AddNewItem(user) : null;

}

[HttpPost]

[Route("login/{passwordIsHashed}")]

public async Task<AppUser> Login([FromBody] AppUser loginUser, bool passwordIsHashed)

{

var user = await \_appUserService.FindByName(loginUser.Name);

if (user == null)

return null;

var res = PasswordHasherService.VerifyMd5Hash(loginUser.Password, user.Password, passwordIsHashed);

return res ? user : null;

}

}

}

* + - 1. Файл controllers/ArtistController.cs

using System.Collections.Generic;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using DatabaseAPI.Services;

using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class ArtistController : ControllerBase

{

private readonly ArtistService \_artistService;

public ArtistController(ArtistService artistService)

{

\_artistService = artistService;

}

[HttpGet]

public string Index()

{

return "This is ArtistController";

}

[HttpGet]

[Route("all")]

public IEnumerable<Artist> GetAll()

{

return \_artistService.GetAll();

}

[Route("find/{query}/{userId}")]

public async Task<List<UserArtistLink>> FindByName(string query, int userId)

{

return await \_artistService.FindByName(query, userId);

}

[Route("findLinked/{query}/{userId}")]

public async Task<List<UserArtistLink>> FindByNameLinked(string query, int userId)

{

query = query.Equals("null") ? "" : query;

return await \_artistService.FindByNameLinked(query, userId);

}

[HttpGet]

[Route("{userId}/{artistId}/albums")]

public async Task<List<UserAlbumLink>> GetAlbums(int userId, int artistId)

{

return await \_artistService.GetAlbums(artistId, userId);

}

}

}

* + - 1. Файл controllers/ConcertController.cs

using System.Collections.Generic;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using DatabaseAPI.Services;

using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class ConcertController : ControllerBase

{

private readonly ConcertService \_concertService;

public ConcertController(ConcertService concertService)

{

\_concertService = concertService;

}

[HttpGet]

[Route("find/{query}/{userId}")]

public async Task<List<UserConcertLink>> FindByName(string query, int userId)

{

return await \_concertService.FindByName(query, userId);

}

[HttpGet]

[Route("getNearest/{userId}/{query}")]

public async Task<List<UserConcertLink>> GetNearest(int userId, string query)

{

query = query.Equals("null") ? "" : query;

return await \_concertService.GetNearest(userId, query);

}

[HttpGet]

[Route("getNearestByArtist/{userId}/{query}")]

public async Task<List<UserConcertLink>> GetNearestByArtist(int userId, string query)

{

query = query.Equals("null") ? "" : query;

return await \_concertService.GetNearestByArtist(userId, query);

}

// Returns UserConcertLinks for concerts where subscribed artists participate

[HttpGet]

[Route("getBySubscription/{userId}/{query}")]

public async Task<List<UserConcertLink>> GetBySubscription(int userId, string query)

{

query = query.Equals("null") ? "" : query;

return await \_concertService.GetBySubscription(userId, query);

}

[HttpGet]

[Route("getBySubscriptionByArtist/{userId}/{query}")]

public async Task<List<UserConcertLink>> GetBySubscriptionByArtist(int userId, string query)

{

query = query.Equals("null") ? "" : query;

return await \_concertService.GetBySubscriptionByArtist(userId, query);

}

[HttpGet]

[Route("getSaved/{userId}/{query}")]

public async Task<List<UserConcertLink>> GetSaved(int userId, string query)

{

query = query.Equals("null") ? "" : query;

return await \_concertService.GetSaved(userId, query);

}

[HttpGet]

[Route("getSavedByArtist/{userId}/{query}")]

public async Task<List<UserConcertLink>> GetSavedByArtist(int userId, string query)

{

query = query.Equals("null") ? "" : query;

return await \_concertService.GetSavedByArtist(userId, query);

}

[HttpGet]

[Route("{userId}/{concertId}/artists")]

public async Task<List<UserArtistLink>> GetArtists(int userId, int concertId)

{

return await \_concertService.GetArtists(userId, concertId);

}

}

}

* + - 1. Файл controllers/DonationController.cs

using System;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using DatabaseAPI.Services;

using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class DonationController: ControllerBase

{

private readonly DonationService \_donationService;

public DonationController(DonationService donationService)

{

\_donationService = donationService;

}

[HttpPost]

[Route("add")]

public async Task<Donation> AddNewItem([FromBody] Donation item)

{

item.Date = DateTime.Now;

return await \_donationService.AddNewItem(item);

}

}

}

* + - 1. Файл controllers/PostController.cs

using System.Collections.Generic;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using DatabaseAPI.Services;

using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class PostController : ControllerBase

{

private readonly PostService \_postService;

public PostController(PostService postService)

{

\_postService = postService;

}

[HttpGet]

[Route("getBySubscription/{userId}")]

public async Task<List<ArtistPostLink>> GetBySubscription(int userId)

{

return await \_postService.GetBySubscription(userId);

}

[HttpGet]

[Route("getLatest")]

public async Task<List<ArtistPostLink>> GetLatest()

{

return await \_postService.GetLatest();

}

[HttpGet]

[Route("getSongs/{userId}/{postId}")]

public async Task<List<UserSongLink>> GetSongs(int userId, int postId)

{

return await \_postService.GetSongs(userId, postId);

}

}

}

* + - 1. Файл controllers/SongController.cs

using System.Collections.Generic;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using DatabaseAPI.Services;

using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class SongController : ControllerBase

{

private readonly SongService \_songService;

public SongController(SongService songService)

{

\_songService = songService;

}

[Route("find/{query}/{userId}")]

public async Task<List<UserSongLink>> FindByName(string query, int userId)

{

return await \_songService.FindByName(query, userId);

}

[Route("findLinked/{query}/{userId}")]

public async Task<List<UserSongLink>> FindByNameLinked(string query, int userId)

{

query = query.Equals("null") ? "" : query;

return await \_songService.FindByNameLinked(query, userId);

}

}

}

* + - 1. Файл controllers/UserAlbumLinkController.cs

using System;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using DatabaseAPI.Services;

using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class UserAlbumLinkController : ControllerBase

{

private readonly UserAlbumLinkService \_userAlbumLinkService;

public UserAlbumLinkController(UserAlbumLinkService userAlbumLinkService)

{

\_userAlbumLinkService = userAlbumLinkService;

}

[HttpPost]

[Route("add")]

public async Task<UserAlbumLink> AddNewItem([FromBody] UserAlbumLink item)

{

return await \_userAlbumLinkService.AddNewItem(item);

}

[HttpDelete]

[Route("delete/{userId}/{albumId}")]

public async Task<Boolean> Delete(int userId, int albumId)

{

UserAlbumLink link = new UserAlbumLink(userId, albumId);

return await \_userAlbumLinkService.DeleteItem(link);

}

[HttpGet]

[Route("linkExist/{userId}/{albumId}")]

public async Task<Boolean> linkExist(int userId, int albumId)

{

return await \_userAlbumLinkService.LinkExist(userId, albumId);

}

}

}

* + - 1. Файл controllers/UserArtistLinkController.cs

using System;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using DatabaseAPI.Services;

using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class UserArtistLinkController : ControllerBase

{

private readonly UserArtistLinkService \_userArtistLinkService;

public UserArtistLinkController(UserArtistLinkService userArtistLinkService)

{

\_userArtistLinkService = userArtistLinkService;

}

[HttpPost]

[Route("add")]

public async Task<UserArtistLink> AddNewItem([FromBody] UserArtistLink item)

{

return await \_userArtistLinkService.AddNewItem(item);

}

[HttpDelete]

[Route("delete/{userId}/{artistId}")]

public async Task<Boolean> Delete(int userId, int artistId)

{

UserArtistLink link = new UserArtistLink(userId, artistId);

return await \_userArtistLinkService.DeleteItem(link);

}

[HttpGet]

[Route("linkExist/{userId}/{artistId}")]

public async Task<Boolean> linkExist(int userId, int artistId)

{

return await \_userArtistLinkService.LinkExist(userId, artistId);

}

}

}

* + - 1. Файл controllers/UserConcertLinkController.cs

using System;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using DatabaseAPI.Services;

using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class UserConcertLinkController : ControllerBase

{

private readonly UserConcertLinkService \_userConcertLinkService;

public UserConcertLinkController(UserConcertLinkService userConcertLinkService)

{

\_userConcertLinkService = userConcertLinkService;

}

[HttpPost]

[Route("add")]

public async Task<UserConcertLink> AddNewItem([FromBody] UserConcertLink item)

{

return await \_userConcertLinkService.AddNewItem(item);

}

[HttpDelete]

[Route("delete/{userId}/{concertId}")]

public async Task<Boolean> Delete(int userId, int concertId)

{

UserConcertLink link = new UserConcertLink(userId, concertId);

return await \_userConcertLinkService.DeleteItem(link);

}

}

}

* + - 1. Файл controllers/UserSongLinkController.cs

using System;

using System.Threading.Tasks;

using DatabaseAPI.Models;

using DatabaseAPI.Services;

using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class UserSongLinkController : ControllerBase

{

private readonly UserSongLinkService \_userSongLinkService;

public UserSongLinkController(UserSongLinkService userSongLinkService)

{

\_userSongLinkService = userSongLinkService;

}

[HttpPost]

[Route("add")]

public async Task<UserSongLink> AddNewItem([FromBody] UserSongLink item)

{

return await \_userSongLinkService.AddNewItem(item);

}

[HttpDelete]

[Route("delete/{userId}/{songId}")]

public async Task<Boolean> Delete(int userId, int songId)

{

UserSongLink link = new UserSongLink(userId, songId);

return await \_userSongLinkService.DeleteItem(link);

}

}

}

* 1. Android приложение
     1. Java
        1. Файл api/ApiController.java

package com.siroytman.indiewindymobile.api;

import android.util.Log;

import com.android.volley.Response;

import com.android.volley.RetryPolicy;

import com.android.volley.VolleyError;

import com.android.volley.toolbox.JsonArrayRequest;

import com.android.volley.toolbox.JsonObjectRequest;

import com.android.volley.toolbox.StringRequest;

import com.siroytman.indiewindymobile.BuildConfig;

import com.siroytman.indiewindymobile.R;

import com.siroytman.indiewindymobile.controller.AppController;

import org.json.JSONArray;

import org.json.JSONObject;

public class ApiController {

private static final String serverUrl;

static {

// Choosing server url

int serverUrlId = BuildConfig.DEBUG ? R.string.server\_url : R.string.azure\_server\_url;

serverUrl = AppController.getContext().getString(serverUrlId);

}

/\*\*

\* Volley queue for executing requests to server

\*/

private VolleyQueue volleyQueue;

/\*\*

\* A singleton instance of the application class for easy access in other places

\*/

private static ApiController instance;

private ApiController() {

// initialize the singleton

instance = this;

// initialize volley queue

volleyQueue = new VolleyQueue(AppController.getContext());

}

/\*\*

\* @return ApplicationController singleton instance

\*/

public static synchronized ApiController getInstance() {

if (instance == null) {

instance = new ApiController();

}

return instance;

}

/\*\*

\* Returns string response from server

\* @param method restMethod: GET, POST, ... {@link com.android.volley.Request.Method}

\* @param apiUrl api address

\* @param callback function to call when got response

\*/

public void getStringResponse(int method, String apiUrl, final VolleyCallbackString callback) {

StringRequest request = new StringRequest(method,

serverUrl + "/" + apiUrl,

new Response.Listener <String> () {

@Override

public void onResponse(String response) {

callback.onSuccessResponse(response);

}

}, new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError e) {

Log.d("ApiController", "Error: " + e.toString());

}

});

request.setRetryPolicy(new RetryPolicy() {

@Override

public int getCurrentTimeout() {

return 50000;

}

@Override

public int getCurrentRetryCount() {

return 50000;

}

@Override

public void retry(VolleyError error) throws VolleyError {

}

});

volleyQueue.addToRequestQueue(request);

}

public void getJSONObjectResponse(String apiUrl, JSONObject json, final VolleyCallbackJSONObject callback) {

JsonObjectRequest request = new JsonObjectRequest(serverUrl + "/" + apiUrl,

json,

new Response.Listener<JSONObject> () {

@Override

public void onResponse(JSONObject response) {

callback.onSuccessResponse(response);

}

}, new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError e) {

callback.onErrorResponse(e);

Log.d("ApiController", "Error: " + e.toString());

}

});

request.setRetryPolicy(new RetryPolicy() {

@Override

public int getCurrentTimeout() {

return 50000;

}

@Override

public int getCurrentRetryCount() {

return 50000;

}

@Override

public void retry(VolleyError error) throws VolleyError {

}

});

volleyQueue.addToRequestQueue(request);

}

public void getJSONArrayResponse(int method, String apiUrl, JSONArray json, final VolleyCallbackJSONArray callback) {

JsonArrayRequest request = new JsonArrayRequest(method, serverUrl + "/" + apiUrl,

json,

new Response.Listener<JSONArray> () {

@Override

public void onResponse(JSONArray response) {

callback.onSuccessResponse(response);

}

}, new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError e) {

callback.onErrorResponse(e);

Log.d("ApiController", "Error: " + e.toString());

}

});

request.setRetryPolicy(new RetryPolicy() {

@Override

public int getCurrentTimeout() {

return 50000;

}

@Override

public int getCurrentRetryCount() {

return 50000;

}

@Override

public void retry(VolleyError error) throws VolleyError {

}

});

volleyQueue.addToRequestQueue(request);

}

}

* + - 1. Файл api/ErrorHandler.java

package com.siroytman.indiewindymobile.api;

import android.content.Context;

import android.util.Log;

import android.widget.Toast;

import com.android.volley.NoConnectionError;

import com.android.volley.TimeoutError;

import com.android.volley.VolleyError;

import com.siroytman.indiewindymobile.R;

public class ErrorHandler {

public static final String TAG = "ErrorHandler";

// Return true if handled, false otherwise

public static Boolean HandleError(Context context, VolleyError error)

{

if (error.getClass() == NoConnectionError.class)

{

Toast.makeText(context, context.getString(R.string.error\_handler\_\_no\_connection), Toast.LENGTH\_LONG)

.show();

Log.d(TAG, context.getString(R.string.error\_handler\_\_no\_connection));

return true;

}

if (error.getClass() == TimeoutError.class)

{

Toast.makeText(context, context.getString(R.string.error\_handler\_\_timeout), Toast.LENGTH\_LONG)

.show();

Log.d(TAG, context.getString(R.string.error\_handler\_\_timeout));

return true;

}

Log.d(TAG, error.getMessage());

return false;

}

}

* + - 1. Файл api/SSLSocketFactoryProvider.java

package com.siroytman.indiewindymobile.api;

import android.content.Context;

import android.util.Log;

import com.siroytman.indiewindymobile.R;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStream;

import java.security.KeyManagementException;

import java.security.KeyStore;

import java.security.KeyStoreException;

import java.security.NoSuchAlgorithmException;

import java.security.cert.Certificate;

import java.security.cert.CertificateException;

import java.security.cert.CertificateFactory;

import java.security.cert.X509Certificate;

import javax.net.ssl.HostnameVerifier;

import javax.net.ssl.HttpsURLConnection;

import javax.net.ssl.SSLContext;

import javax.net.ssl.SSLSession;

import javax.net.ssl.SSLSocketFactory;

import javax.net.ssl.TrustManagerFactory;

/\*\*

\* Class that provides SSLSockets for connection to server

\*/

final class SSLSocketFactoryProvider {

// Returns SSLSocketFactory

static SSLSocketFactory getSocketFactory(final Context appContext) {

CertificateFactory cf;

try {

cf = CertificateFactory.getInstance("X.509");

InputStream caInput = appContext.getResources().openRawResource(R.raw.ssl\_certificate);

Certificate ca;

try {

ca = cf.generateCertificate(caInput);

Log.e("CERT", "ca=" + ((X509Certificate) ca).getSubjectDN());

} finally {

caInput.close();

}

String keyStoreType = KeyStore.getDefaultType();

KeyStore keyStore = KeyStore.getInstance(keyStoreType);

keyStore.load(null, null);

keyStore.setCertificateEntry("ca", ca);

String tmfAlgorithm = TrustManagerFactory.getDefaultAlgorithm();

TrustManagerFactory tmf = TrustManagerFactory.getInstance(tmfAlgorithm);

tmf.init(keyStore);

HostnameVerifier hostnameVerifier = new HostnameVerifier() {

@Override

public boolean verify(String hostname, SSLSession session) {

return hostname.compareTo(appContext.getString(R.string.server\_host))==0;

}

};

HttpsURLConnection.setDefaultHostnameVerifier(hostnameVerifier);

SSLContext context = SSLContext.getInstance("TLS");

context.init(null, tmf.getTrustManagers(), null);

HttpsURLConnection.setDefaultSSLSocketFactory(context.getSocketFactory());

return context.getSocketFactory();

} catch (CertificateException e) {

e.printStackTrace();

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

} catch (KeyStoreException e) {

e.printStackTrace();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (KeyManagementException e) {

e.printStackTrace();

}

return null;

}

}

* + - 1. Файл api/ VolleyCallbackJSONArray.java

package com.siroytman.indiewindymobile.api;

import com.android.volley.VolleyError;

import org.json.JSONArray;

public interface VolleyCallbackJSONArray {

void onSuccessResponse(JSONArray result);

void onErrorResponse(VolleyError error);

}

* + - 1. Файл api/ VolleyCallbackJSONObject.java

package com.siroytman.indiewindymobile.api;

import com.android.volley.VolleyError;

import org.json.JSONObject;

public interface VolleyCallbackJSONObject {

void onSuccessResponse(JSONObject result);

void onErrorResponse(VolleyError error);

}

* + - 1. Файл api/ VolleyCallbackString.java

package com.siroytman.indiewindymobile.api;

import com.android.volley.VolleyError;

public interface VolleyCallbackString

{

void onSuccessResponse(String result);

void onErrorResponse(VolleyError error);

}

* + - 1. Файл api/VolleyQueue.java

package com.siroytman.indiewindymobile.api;

import android.content.Context;

import android.text.TextUtils;

import com.android.volley.Network;

import com.android.volley.Request;

import com.android.volley.RequestQueue;

import com.android.volley.VolleyLog;

import com.android.volley.toolbox.BasicNetwork;

import com.android.volley.toolbox.HurlStack;

import com.android.volley.toolbox.Volley;

import com.siroytman.indiewindymobile.BuildConfig;

/\*\*

\* Volley queue for executing requests to server

\*/

class VolleyQueue {

/\*\*

\* Log or request TAG

\*/

static final String TAG = "VolleyPatterns";

/\*\*

\* Global request queue for Volley

\*/

private RequestQueue mRequestQueue;

Context context;

VolleyQueue(Context context){

this.context = context;

}

/\*\*

\* @return The Volley Request queue, the queue will be created if it is null

\*/

RequestQueue getRequestQueue() {

// lazy initialize the request queue, the queue instance will be

// created when it is accessed for the first time

if (mRequestQueue == null) {

if (BuildConfig.DEBUG) {

// Https connection with localhost

mRequestQueue = Volley.newRequestQueue(context,

new HurlStack(null, SSLSocketFactoryProvider.getSocketFactory(context)));

} else {

// Http connection with Azure

mRequestQueue = Volley.newRequestQueue(context);

}

}

return mRequestQueue;

}

/\*\*

\* Adds the specified request to the global queue, if tag is specified

\* then it is used else Default TAG is used.

\*

\* @param req

\* @param tag

\*/

<T> void addToRequestQueue(Request<T> req, String tag) {

// set the default tag if tag is empty

req.setTag(TextUtils.isEmpty(tag) ? TAG : tag);

VolleyLog.d("Adding request to queue: %s", req.getUrl());

getRequestQueue().add(req);

}

/\*\*

\* Adds the specified request to the global queue using the Default TAG.

\*

\* @param req

\*/

<T> void addToRequestQueue(Request<T> req) {

// set the default tag if tag is empty

req.setTag(TAG);

getRequestQueue().add(req);

}

/\*\*

\* Cancels all pending requests by the specified TAG, it is important

\* to specify a TAG so that the pending/ongoing requests can be cancelled.

\*

\* @param tag

\*/

void cancelPendingRequests(Object tag) {

if (mRequestQueue != null) {

mRequestQueue.cancelAll(tag);

}

}

}

* + - 1. Файл controller/AlbumController.java

package com.siroytman.indiewindymobile.controller;

import android.util.Log;

import com.android.volley.Request;

import com.android.volley.VolleyError;

import com.siroytman.indiewindymobile.api.ApiController;

import com.siroytman.indiewindymobile.api.ErrorHandler;

import com.siroytman.indiewindymobile.api.VolleyCallbackJSONArray;

import com.siroytman.indiewindymobile.api.VolleyCallbackJSONObject;

import com.siroytman.indiewindymobile.api.VolleyCallbackString;

import com.siroytman.indiewindymobile.interfaces.ILinkAdd;

import com.siroytman.indiewindymobile.interfaces.ISearchableAlbum;

import com.siroytman.indiewindymobile.model.Album;

import com.siroytman.indiewindymobile.model.UserAlbumLink;

import com.siroytman.indiewindymobile.model.UserSongLink;

import com.siroytman.indiewindymobile.ui.activity.AlbumActivity;

import org.json.JSONArray;

import org.json.JSONObject;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class AlbumController {

private static final String TAG = "AlbumController";

private ApiController apiController;

private static AlbumController instance;

private AlbumController() {

apiController = ApiController.getInstance();

}

public static synchronized AlbumController getInstance() {

if (instance == null) {

instance = new AlbumController();

}

return instance;

}

public void searchAlbums(final ISearchableAlbum view, String query){

String url = "album/find/" + query + "/" + AppController.user.getId();

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "Albums search: request completed");

ArrayList<UserAlbumLink> links = UserAlbumLink.parseLinks(result);

view.albumsFoundViewUpdate(links);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "Albums search: request not completed!");

}

});

}

public void searchAlbumsLinked(final ISearchableAlbum view) {

searchAlbumsLinked(view, "null");

}

public void searchAlbumsLinked(final ISearchableAlbum view, String query){

String url = "album/findLinked/" + query + "/" + AppController.user.getId();

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "Albums search: request completed");

ArrayList<UserAlbumLink> links = UserAlbumLink.parseLinks(result);

view.albumsFoundViewUpdate(links);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "Albums search: request not completed!");

}

});

}

public void getAlbumSongs(final AlbumActivity albumActivity) {

final Album album = albumActivity.getItem();

String url = "album/" + AppController.user.getId() + "/" + album.getId() + "/songs";

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "Songs found");

ArrayList<UserSongLink> links = UserSongLink.parseLinks(result);

albumActivity.songsFoundViewUpdate(links);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

if (!ErrorHandler.HandleError(albumActivity, error)) {

Log.d(TAG, "Songs not found!");

}

}

});

}

public void linkExist(final ILinkAdd<Album> view){

final Album album = view.getItem();

String url = "userAlbumLink/linkExist/" + AppController.user.getId() + "/" + album.getId();

apiController.getStringResponse(Request.Method.GET, url, new VolleyCallbackString() {

@Override

public void onSuccessResponse(String result) {

if (result.equals("true")) {

view.added();

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "linkExist error");

}

});

}

public void addUserAlbumLink(final ILinkAdd<Album> view) {

String url = "userAlbumLink/add";

final Album album = view.getItem();

Map<String, Integer> postParam = new HashMap<>();

postParam.put("AppUserId", AppController.user.getId());

postParam.put("AlbumId", album.getId());

apiController.getJSONObjectResponse(url, new JSONObject(postParam), new VolleyCallbackJSONObject() {

@Override

public void onSuccessResponse(JSONObject result) {

Log.d(TAG, "UserAlbumLink added: " + album.getName());

view.added();

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "UserAlbumLink not added: " + album.getName());

}

});

}

public void removeUserAlbumLink(final ILinkAdd<Album> view) {

final Album album = view.getItem();

String url = "userAlbumLink/delete/" + AppController.user.getId() + "/" + album.getId();

apiController.getStringResponse(Request.Method.DELETE, url, new VolleyCallbackString() {

@Override

public void onSuccessResponse(String result) {

if(result.equals("true")) {

view.removed();

Log.d(TAG, "UserAlbumLink removed: " + album.getName());

} else {

Log.d(TAG, "UserAlbumLink not removed: " + album.getName());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "UserAlbumLink not removed: " + error.getMessage());

}

});

}

}

* + - 1. Файл controller/AppController.java

package com.siroytman.indiewindymobile.controller;

import android.app.Application;

import android.content.Context;

import com.google.android.exoplayer2.DefaultRenderersFactory;

import com.google.android.exoplayer2.RenderersFactory;

import com.google.android.exoplayer2.database.DatabaseProvider;

import com.google.android.exoplayer2.database.ExoDatabaseProvider;

import com.google.android.exoplayer2.offline.ActionFileUpgradeUtil;

import com.google.android.exoplayer2.offline.DefaultDownloadIndex;

import com.google.android.exoplayer2.offline.DefaultDownloaderFactory;

import com.google.android.exoplayer2.offline.DownloadManager;

import com.google.android.exoplayer2.offline.DownloaderConstructorHelper;

import com.google.android.exoplayer2.ui.DownloadNotificationHelper;

import com.google.android.exoplayer2.upstream.DataSource;

import com.google.android.exoplayer2.upstream.DefaultDataSourceFactory;

import com.google.android.exoplayer2.upstream.DefaultHttpDataSourceFactory;

import com.google.android.exoplayer2.upstream.FileDataSource;

import com.google.android.exoplayer2.upstream.HttpDataSource;

import com.google.android.exoplayer2.upstream.cache.Cache;

import com.google.android.exoplayer2.upstream.cache.CacheDataSource;

import com.google.android.exoplayer2.upstream.cache.CacheDataSourceFactory;

import com.google.android.exoplayer2.upstream.cache.NoOpCacheEvictor;

import com.google.android.exoplayer2.upstream.cache.SimpleCache;

import com.google.android.exoplayer2.util.Log;

import com.google.android.exoplayer2.util.Util;

import com.siroytman.indiewindymobile.BuildConfig;

import com.siroytman.indiewindymobile.model.AppUser;

import com.siroytman.indiewindymobile.services.DownloadTracker;

import java.io.File;

import java.io.IOException;

public class AppController extends Application {

public static final String DOWNLOAD\_NOTIFICATION\_CHANNEL\_ID = "download\_channel";

private static final String TAG = "AppController";

private static final String DOWNLOAD\_ACTION\_FILE = "actions";

private static final String DOWNLOAD\_TRACKER\_ACTION\_FILE = "tracked\_actions";

private static final String DOWNLOAD\_CONTENT\_DIRECTORY = "downloads";

protected String userAgent;

private DatabaseProvider databaseProvider;

private File downloadDirectory;

private Cache downloadCache;

private DownloadManager downloadManager;

private DownloadTracker downloadTracker;

private DownloadNotificationHelper downloadNotificationHelper;

public static AppUser user;

public static Context getContext() {

return mInstance.getApplicationContext();

}

private static AppController mInstance;

public static synchronized AppController getInstance() {

return mInstance;

}

@Override

public void onCreate() {

super.onCreate();

userAgent = Util.getUserAgent(this, "ExoPlayer");

mInstance = this;

}

/\*\* Returns a {@link DataSource.Factory}. \*/

public DataSource.Factory buildDataSourceFactory() {

DefaultDataSourceFactory upstreamFactory =

new DefaultDataSourceFactory(this, buildHttpDataSourceFactory());

return buildReadOnlyCacheDataSource(upstreamFactory, getDownloadCache());

}

/\*\* Returns a {@link HttpDataSource.Factory}. \*/

public HttpDataSource.Factory buildHttpDataSourceFactory() {

return new DefaultHttpDataSourceFactory(userAgent);

}

/\*\* Returns whether extension renderers should be used. \*/

public boolean useExtensionRenderers() {

return "withExtensions".equals(BuildConfig.FLAVOR);

}

public RenderersFactory buildRenderersFactory(boolean preferExtensionRenderer) {

@DefaultRenderersFactory.ExtensionRendererMode

int extensionRendererMode =

useExtensionRenderers()

? (preferExtensionRenderer

? DefaultRenderersFactory.EXTENSION\_RENDERER\_MODE\_PREFER

: DefaultRenderersFactory.EXTENSION\_RENDERER\_MODE\_ON)

: DefaultRenderersFactory.EXTENSION\_RENDERER\_MODE\_OFF;

return new DefaultRenderersFactory(/\* context= \*/ this)

.setExtensionRendererMode(extensionRendererMode);

}

public DownloadNotificationHelper getDownloadNotificationHelper() {

if (downloadNotificationHelper == null) {

downloadNotificationHelper =

new DownloadNotificationHelper(this, DOWNLOAD\_NOTIFICATION\_CHANNEL\_ID);

}

return downloadNotificationHelper;

}

public DownloadManager getDownloadManager() {

initDownloadManager();

return downloadManager;

}

public DownloadTracker getDownloadTracker() {

initDownloadManager();

return downloadTracker;

}

protected synchronized Cache getDownloadCache() {

if (downloadCache == null) {

File downloadContentDirectory = new File(getDownloadDirectory(), DOWNLOAD\_CONTENT\_DIRECTORY);

downloadCache =

new SimpleCache(downloadContentDirectory, new NoOpCacheEvictor(), getDatabaseProvider());

}

return downloadCache;

}

private synchronized void initDownloadManager() {

if (downloadManager == null) {

DefaultDownloadIndex downloadIndex = new DefaultDownloadIndex(getDatabaseProvider());

upgradeActionFile(

DOWNLOAD\_ACTION\_FILE, downloadIndex, /\* addNewDownloadsAsCompleted= \*/ false);

upgradeActionFile(

DOWNLOAD\_TRACKER\_ACTION\_FILE, downloadIndex, /\* addNewDownloadsAsCompleted= \*/ true);

DownloaderConstructorHelper downloaderConstructorHelper =

new DownloaderConstructorHelper(getDownloadCache(), buildHttpDataSourceFactory());

downloadManager =

new DownloadManager(

this, downloadIndex, new DefaultDownloaderFactory(downloaderConstructorHelper));

downloadTracker =

new DownloadTracker(/\* context= \*/ this, buildDataSourceFactory(), downloadManager);

}

}

private void upgradeActionFile(

String fileName, DefaultDownloadIndex downloadIndex, boolean addNewDownloadsAsCompleted) {

try {

ActionFileUpgradeUtil.upgradeAndDelete(

new File(getDownloadDirectory(), fileName),

/\* downloadIdProvider= \*/ null,

downloadIndex,

/\* deleteOnFailure= \*/ true,

addNewDownloadsAsCompleted);

} catch (IOException e) {

Log.e(TAG, "Failed to upgrade action file: " + fileName, e);

}

}

private DatabaseProvider getDatabaseProvider() {

if (databaseProvider == null) {

databaseProvider = new ExoDatabaseProvider(this);

}

return databaseProvider;

}

private File getDownloadDirectory() {

if (downloadDirectory == null) {

downloadDirectory = getExternalFilesDir(null);

if (downloadDirectory == null) {

downloadDirectory = getFilesDir();

}

}

return downloadDirectory;

}

protected static CacheDataSourceFactory buildReadOnlyCacheDataSource(

DataSource.Factory upstreamFactory, Cache cache) {

return new CacheDataSourceFactory(

cache,

upstreamFactory,

new FileDataSource.Factory(),

/\* cacheWriteDataSinkFactory= \*/ null,

CacheDataSource.FLAG\_IGNORE\_CACHE\_ON\_ERROR,

/\* eventListener= \*/ null);

}

}

* + - 1. Файл controller/ArtistController.java

package com.siroytman.indiewindymobile.controller;

import android.util.Log;

import com.android.volley.Request;

import com.android.volley.VolleyError;

import com.siroytman.indiewindymobile.api.ApiController;

import com.siroytman.indiewindymobile.api.ErrorHandler;

import com.siroytman.indiewindymobile.api.VolleyCallbackJSONArray;

import com.siroytman.indiewindymobile.api.VolleyCallbackJSONObject;

import com.siroytman.indiewindymobile.api.VolleyCallbackString;

import com.siroytman.indiewindymobile.interfaces.ILinkAdd;

import com.siroytman.indiewindymobile.interfaces.ISearchableArtist;

import com.siroytman.indiewindymobile.model.Artist;

import com.siroytman.indiewindymobile.model.UserAlbumLink;

import com.siroytman.indiewindymobile.model.UserArtistLink;

import com.siroytman.indiewindymobile.ui.activity.ArtistActivity;

import org.json.JSONArray;

import org.json.JSONObject;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class ArtistController {

private static final String TAG = "ArtistController";

private ApiController apiController;

private static ArtistController instance;

private ArtistController() {

apiController = ApiController.getInstance();

}

public static synchronized ArtistController getInstance() {

if (instance == null) {

instance = new ArtistController();

}

return instance;

}

public void searchArtists(final ISearchableArtist view, String query){

String url = "artist/find/" + query + "/" + AppController.user.getId();

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "Artists search: request completed");

ArrayList<UserArtistLink> links = UserArtistLink.parseLinks(result);

view.artistsFoundViewUpdate(links);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "Artists search: request not completed!");

}

});

}

public void searchArtistsLinked(final ISearchableArtist view) {

searchArtistsLinked(view, "null");

}

public void searchArtistsLinked(final ISearchableArtist view, String query){

String url = "artist/findLinked/" + query + "/" + AppController.user.getId();

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "Artists search: request completed");

ArrayList<UserArtistLink> links = UserArtistLink.parseLinks(result);

view.artistsFoundViewUpdate(links);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "Artists search: request not completed!");

}

});

}

public void getArtistAlbums(final ArtistActivity artistActivity, Artist artist){

String url = "artist/" + AppController.user.getId() + "/" + artist.getId() + "/albums";

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "Albums found");

ArrayList<UserAlbumLink> links = UserAlbumLink.parseLinks(result);

artistActivity.albumsFoundViewUpdate(links);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

if (!ErrorHandler.HandleError(artistActivity, error)) {

Log.d(TAG, "Albums not found!");

}

}

});

}

public void linkExist(final ILinkAdd<Artist> view){

final Artist artist = view.getItem();

String url = "userArtistLink/linkExist/" + AppController.user.getId() + "/" + artist.getId();

apiController.getStringResponse(Request.Method.GET, url, new VolleyCallbackString() {

@Override

public void onSuccessResponse(String result) {

if (result.equals("true")) {

view.added();

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "linkExist error");

}

});

}

public void addUserArtistLink(final ILinkAdd<Artist> view) {

String url = "userArtistLink/add";

final Artist artist = view.getItem();

Map<String, Integer> postParam = new HashMap<>();

postParam.put("AppUserId", AppController.user.getId());

postParam.put("ArtistId", artist.getId());

apiController.getJSONObjectResponse(url, new JSONObject(postParam), new VolleyCallbackJSONObject() {

@Override

public void onSuccessResponse(JSONObject result) {

Log.d(TAG, "UserArtistLink added: " + artist.getName());

view.added();

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "UserArtistLink not added: " + artist.getName());

}

});

}

public void removeUserArtistLink(final ILinkAdd<Artist> view) {

final Artist artist = view.getItem();

String url = "userArtistLink/delete/" + AppController.user.getId() + "/" + artist.getId();

apiController.getStringResponse(Request.Method.DELETE, url, new VolleyCallbackString() {

@Override

public void onSuccessResponse(String result) {

if(result.equals("true")) {

view.removed();

Log.d(TAG, "UserArtistLink removed: " + artist.getName());

} else {

Log.d(TAG, "UserArtistLink not removed: " + artist.getName());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "UserArtistLink not removed: " + error.getMessage());

}

});

}

}

* + - 1. Файл controller/ConcertController.java

package com.siroytman.indiewindymobile.controller;

import android.util.Log;

import com.android.volley.Request;

import com.android.volley.VolleyError;

import com.siroytman.indiewindymobile.api.ApiController;

import com.siroytman.indiewindymobile.api.VolleyCallbackJSONArray;

import com.siroytman.indiewindymobile.api.VolleyCallbackJSONObject;

import com.siroytman.indiewindymobile.api.VolleyCallbackString;

import com.siroytman.indiewindymobile.interfaces.ILinkAdd;

import com.siroytman.indiewindymobile.model.Concert;

import com.siroytman.indiewindymobile.model.UserArtistLink;

import com.siroytman.indiewindymobile.model.UserConcertLink;

import com.siroytman.indiewindymobile.ui.activity.ConcertActivity;

import com.siroytman.indiewindymobile.ui.fragments.concert.NearestConcertFragment;

import com.siroytman.indiewindymobile.ui.fragments.concert.SavedConcertFragment;

import com.siroytman.indiewindymobile.ui.fragments.concert.SubscriptionConcertFragment;

import org.json.JSONArray;

import org.json.JSONObject;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class ConcertController {

private static final String TAG = "ConcertController";

private ApiController apiController;

private static ConcertController instance;

private ConcertController() {

apiController = ApiController.getInstance();

}

public static synchronized ConcertController getInstance() {

if (instance == null) {

instance = new ConcertController();

}

return instance;

}

private String createGetConcertsUrl(String urlStart, String query, boolean byArtist) {

String urlEnd = "/" + AppController.user.getId() + "/" + query;

return byArtist ? urlStart + "ByArtist" + urlEnd : urlStart + urlEnd;

}

public void getNearestConcerts() {

getNearestConcerts("null", false);

}

public void getNearestConcerts(String query, boolean byArtist){

String url = createGetConcertsUrl("concert/getNearest", query, byArtist);

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "getNearestConcerts: search request completed");

ArrayList<UserConcertLink> links = UserConcertLink.parseLinks(result);

NearestConcertFragment.getInstance().concertsFoundViewUpdate(links);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "getNearestConcerts: search request not completed!" + error.getMessage());

}

});

}

public void getSubscriptionConcerts() {

getSubscriptionConcerts("null", false);

}

public void getSubscriptionConcerts(String query, boolean byArtist){

String url = createGetConcertsUrl("concert/getBySubscription", query, byArtist);

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "getSubscriptionConcerts: search request completed");

ArrayList<UserConcertLink> links = UserConcertLink.parseLinks(result);

SubscriptionConcertFragment.getInstance().concertsFoundViewUpdate(links);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "getSubscriptionConcerts: search request not completed!" + error.getMessage());

}

});

}

public void getSavedConcerts() {

getSavedConcerts("null", false);

}

public void getSavedConcerts(String query, boolean byArtist){

String url = createGetConcertsUrl("concert/getSaved", query, byArtist);

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "getSavedConcerts: search request completed");

ArrayList<UserConcertLink> links = UserConcertLink.parseLinks(result);

SavedConcertFragment.getInstance().concertsFoundViewUpdate(links);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "getSavedConcerts: search request not completed!" + error.getMessage());

}

});

}

public void getArtists(final ConcertActivity concertActivity, final int concertId){

String url = "concert/" + AppController.user.getId() + "/" + concertId + "/artists";

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "Artists search: request completed");

ArrayList<UserArtistLink> links = UserArtistLink.parseLinks(result);

concertActivity.artistsFoundViewUpdate(links);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "Artists search: request not completed!");

}

});

}

public void addUserConcertLink(final ILinkAdd<Concert> view) {

String url = "userConcertLink/add";

final Concert concert = view.getItem();

Map<String, Integer> postParam = new HashMap<>();

postParam.put("AppUserId", AppController.user.getId());

postParam.put("ConcertId", concert.getId());

apiController.getJSONObjectResponse(url, new JSONObject(postParam), new VolleyCallbackJSONObject() {

@Override

public void onSuccessResponse(JSONObject result) {

Log.d(TAG, "userConcertLink added: " + concert.getName());

view.added();

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "userConcertLink not added: " + concert.getName());

}

});

}

public void removeUserConcertLink(final ILinkAdd<Concert> view) {

final Concert concert = view.getItem();

String url = "userConcertLink/delete/" + AppController.user.getId() + "/" + concert.getId();

apiController.getStringResponse(Request.Method.DELETE, url, new VolleyCallbackString() {

@Override

public void onSuccessResponse(String result) {

if(result.equals("true")) {

view.removed();

Log.d(TAG, "userConcertLink removed: " + concert.getName());

} else {

Log.d(TAG, "userConcertLink not removed: " + concert.getName());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "userConcertLink not removed: " + error.getMessage());

}

});

}

}

* + - 1. Файл controller/DonationController.java

package com.siroytman.indiewindymobile.controller;

import android.content.Context;

import android.util.Log;

import android.widget.Toast;

import com.android.volley.VolleyError;

import com.siroytman.indiewindymobile.R;

import com.siroytman.indiewindymobile.api.ApiController;

import com.siroytman.indiewindymobile.api.VolleyCallbackJSONObject;

import com.siroytman.indiewindymobile.model.Artist;

import org.json.JSONObject;

import java.util.HashMap;

import java.util.Map;

public class DonationController {

private static final String TAG = "DonationController";

private ApiController apiController;

private static DonationController instance;

private DonationController() {

apiController = ApiController.getInstance();

}

public static synchronized DonationController getInstance() {

if (instance == null) {

instance = new DonationController();

}

return instance;

}

public void addDonation(final Context context, final Artist artist, final int donationAmount) {

String url = "donation/add";

Map<String, Integer> postParam = new HashMap<>();

postParam.put("AppUserId", AppController.user.getId());

postParam.put("ArtistId", artist.getId());

postParam.put("Amount", donationAmount);

apiController.getJSONObjectResponse(url, new JSONObject(postParam), new VolleyCallbackJSONObject() {

@Override

public void onSuccessResponse(JSONObject result) {

Log.d(TAG, "donation added:"

+ "user=" + AppController.user.getName()

+ "; artist=" + artist.getName()

+ "; amount=" + donationAmount);

Toast.makeText(context, R.string.donation\_activity\_\_donation\_message, Toast.LENGTH\_LONG)

.show();

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "donation not added:"

+ "user=" + AppController.user.getName()

+ "; artist=" + artist.getName()

+ "; amount=" + donationAmount);

Toast.makeText(context, R.string.donation\_activity\_\_donation\_message, Toast.LENGTH\_LONG)

.show();

}

});

}

}

* + - 1. Файл controller/LoginController.java

package com.siroytman.indiewindymobile.controller;

import android.content.Context;

import android.content.Intent;

import android.util.Log;

import android.widget.Toast;

import com.android.volley.VolleyError;

import com.siroytman.indiewindymobile.R;

import com.siroytman.indiewindymobile.api.ApiController;

import com.siroytman.indiewindymobile.api.ErrorHandler;

import com.siroytman.indiewindymobile.api.VolleyCallbackJSONObject;

import com.siroytman.indiewindymobile.model.AppUser;

import com.siroytman.indiewindymobile.services.SharedPrefsService;

import com.siroytman.indiewindymobile.ui.activity.LoginActivity;

import com.siroytman.indiewindymobile.ui.activity.NavigationActivity;

import com.siroytman.indiewindymobile.ui.activity.PlayerActivity;

import org.json.JSONObject;

import java.util.HashMap;

import java.util.Map;

public class LoginController {

public static final String TAG = "LoginController";

private ApiController apiController;

private SharedPrefsService sharedPrefsService;

private static LoginController instance;

private LoginController() {

apiController = ApiController.getInstance();

sharedPrefsService = new SharedPrefsService();

}

public static synchronized LoginController getInstance() {

if (instance == null) {

instance = new LoginController();

}

return instance;

}

public void login(LoginActivity loginActivity, String name, String password){

login(loginActivity, name, password, false);

}

private void login(final LoginActivity loginActivity, String name, String password, Boolean passwordIsHashed){

String url = "appuser/login/" + passwordIsHashed;

Map<String, String> postParam = new HashMap<>();

postParam.put("Name", name);

postParam.put("Password", password);

apiController.getJSONObjectResponse(url, new JSONObject(postParam), new VolleyCallbackJSONObject() {

@Override

public void onSuccessResponse(JSONObject result) {

try {

AppController.user = AppUser.ParseAppUser(result);

// Save to shared prefs

sharedPrefsService.saveAppUser(AppController.user);

String msg = String.format(loginActivity.getString(R.string.login\_\_hello), AppController.user.getName());

Toast.makeText(loginActivity, msg, Toast.LENGTH\_LONG)

.show();

// Swap to next activity

loginActivity.startActivity(new Intent(loginActivity, NavigationActivity.class));

loginActivity.finish();

}

catch (Exception e)

{

loginActivity.stopLoadingProgressBar();

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

loginActivity.stopLoadingProgressBar();

if (!ErrorHandler.HandleError(loginActivity, error)) {

Toast.makeText(loginActivity, loginActivity.getString(R.string.login\_\_incorrect\_login),

Toast.LENGTH\_LONG)

.show();

}

}

});

}

public void loginFromSharedPrefs(LoginActivity loginActivity){

AppUser user = sharedPrefsService.getAppUser();

if (user != null){

loginActivity.setLoginAndPasswordText(user.getName(), user.getPassword());

loginActivity.startLoadingProgressBar();

login(loginActivity, user.getName(), user.getPassword(), true);

Log.d(TAG, "Login by shared prefs");

}

else{

Log.d(TAG, "No shared prefs exists for login");

loginActivity.stopLoadingProgressBar();

}

}

public void logout(Context context) {

removeSharedPrefs();

AppController.user = null;

context.startActivity(new Intent(context, LoginActivity.class));

}

public void removeSharedPrefs(){

sharedPrefsService.remove();

}

public void register(final LoginActivity loginActivity, String name, String password){

String url = "appuser/register";

Map<String, String> postParam = new HashMap<>();

postParam.put("Name", name);

postParam.put("Password", password);

apiController.getJSONObjectResponse(url, new JSONObject(postParam), new VolleyCallbackJSONObject() {

@Override

public void onSuccessResponse(JSONObject result) {

AppController.user = AppUser.ParseAppUser(result);

// Save to shared prefs

sharedPrefsService.saveAppUser(AppController.user);

String msg = String.format(loginActivity.getString(R.string.login\_\_hello), AppController.user.getName());

Toast.makeText(loginActivity, msg, Toast.LENGTH\_LONG)

.show();

// Swap to next activity

loginActivity.startActivity(new Intent(loginActivity, NavigationActivity.class));

loginActivity.finish();

}

@Override

public void onErrorResponse(VolleyError error) {

if (!ErrorHandler.HandleError(loginActivity, error)) {

Toast.makeText(loginActivity, loginActivity.getString(R.string.login\_\_user\_exist),

Toast.LENGTH\_LONG)

.show();

}

}

});

}

}

* + - 1. Файл controller/PostController.java

package com.siroytman.indiewindymobile.controller;

import android.util.Log;

import com.android.volley.Request;

import com.android.volley.VolleyError;

import com.siroytman.indiewindymobile.api.ApiController;

import com.siroytman.indiewindymobile.api.VolleyCallbackJSONArray;

import com.siroytman.indiewindymobile.model.ArtistPostLink;

import com.siroytman.indiewindymobile.model.Post;

import com.siroytman.indiewindymobile.model.UserSongLink;

import com.siroytman.indiewindymobile.ui.adapter.ArtistPostLinkListAdapter;

import com.siroytman.indiewindymobile.ui.fragments.home.LatestHomeFragment;

import com.siroytman.indiewindymobile.ui.fragments.home.SubscriptionHomeFragment;

import org.json.JSONArray;

import java.util.ArrayList;

public class PostController {

private static final String TAG = "PostController";

private ApiController apiController;

private static PostController instance;

private PostController() {

apiController = ApiController.getInstance();

}

public static synchronized PostController getInstance() {

if (instance == null) {

instance = new PostController();

}

return instance;

}

public void getSubscriptionPosts(){

String url = "post/getBySubscription/" + AppController.user.getId();

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "getSubscriptionPosts: search request completed");

ArrayList<ArtistPostLink> posts = ArtistPostLink.parseArray(result);

SubscriptionHomeFragment.getInstance().postsFoundViewUpdate(posts);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "getSubscriptionPosts: search request not completed!" + error.getMessage());

}

});

}

public void getLatestPosts(){

String url = "post/getLatest";

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "getLatestPosts: completed");

ArrayList<ArtistPostLink> posts = ArtistPostLink.parseArray(result);

LatestHomeFragment.getInstance().postsFoundViewUpdate(posts);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "getLatestPosts: not completed!" + error.getMessage());

}

});

}

public void getPostSongs(final ArtistPostLinkListAdapter.ViewHolder viewHolder, int postId){

String url = "post/getSongs/" + AppController.user.getId() + "/" + postId;

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "getPostSongs: completed");

if(result.length() > 0) {

ArrayList<UserSongLink> songLinks = UserSongLink.parseLinks(result);

viewHolder.updatePostSongsView(songLinks);

}

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "getPostSongs: not completed!" + error.getMessage());

}

});

}

}

* + - 1. Файл controller/SearchController.java

package com.siroytman.indiewindymobile.controller;

import com.siroytman.indiewindymobile.api.ApiController;

import com.siroytman.indiewindymobile.interfaces.ISearchableAlbum;

import com.siroytman.indiewindymobile.interfaces.ISearchableArtist;

import com.siroytman.indiewindymobile.interfaces.ISearchableSong;

public class SearchController {

private static final String TAG = "SearchController";

private ApiController apiController;

private static SearchController instance;

private SearchController() {

apiController = ApiController.getInstance();

}

public static synchronized SearchController getInstance() {

if (instance == null) {

instance = new SearchController();

}

return instance;

}

public void searchSongs(final ISearchableSong view, final String query){

SongController.getInstance().searchSongs(view, query);

}

public void searchArtists(final ISearchableArtist view, String query){

ArtistController.getInstance().searchArtists(view, query);

}

public void searchAlbums(final ISearchableAlbum view, String query){

AlbumController.getInstance().searchAlbums(view, query);

}

}

* + - 1. Файл controller/SongController.java

package com.siroytman.indiewindymobile.controller;

import android.util.Log;

import com.android.volley.Request;

import com.android.volley.VolleyError;

import com.siroytman.indiewindymobile.api.VolleyCallbackJSONArray;

import com.siroytman.indiewindymobile.interfaces.ILinkAdd;

import com.siroytman.indiewindymobile.api.ApiController;

import com.siroytman.indiewindymobile.api.VolleyCallbackJSONObject;

import com.siroytman.indiewindymobile.api.VolleyCallbackString;

import com.siroytman.indiewindymobile.interfaces.ISearchableSong;

import com.siroytman.indiewindymobile.model.Song;

import com.siroytman.indiewindymobile.model.UserSongLink;

import org.json.JSONArray;

import org.json.JSONObject;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class SongController {

private static final String TAG = "SongController";

private ApiController apiController;

private static SongController instance;

private SongController() {

apiController = ApiController.getInstance();

}

public static synchronized SongController getInstance() {

if (instance == null) {

instance = new SongController();

}

return instance;

}

public void searchSongs(final ISearchableSong view, final String query){

String url = "song/find/" + query + "/" + AppController.user.getId();

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "Songs search: request completed");

ArrayList<UserSongLink> links = UserSongLink.parseLinks(result);

view.songsFoundViewUpdate(links);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "Songs search: request not completed!");

}

});

}

public void searchSongsLinked(final ISearchableSong view) {

searchSongsLinked(view, "null");

}

public void searchSongsLinked(final ISearchableSong view, final String query){

String url = "song/findLinked/" + query + "/" + AppController.user.getId();

apiController.getJSONArrayResponse(Request.Method.GET, url, null, new VolleyCallbackJSONArray() {

@Override

public void onSuccessResponse(JSONArray result) {

try {

Log.d(TAG, "Songs search: request completed");

ArrayList<UserSongLink> links = UserSongLink.parseLinks(result);

view.songsFoundViewUpdate(links);

}

catch (Exception e) {

Log.d(TAG, "Unable to parse response: " + e.getMessage());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "Songs search: request not completed!");

}

});

}

public void addUserSongLink(final ILinkAdd<Song> view) {

String url = "userSongLink/add";

final Song song = view.getItem();

Map<String, Integer> postParam = new HashMap<>();

postParam.put("AppUserId", AppController.user.getId());

postParam.put("SongId", song.getId());

apiController.getJSONObjectResponse(url, new JSONObject(postParam), new VolleyCallbackJSONObject() {

@Override

public void onSuccessResponse(JSONObject result) {

Log.d(TAG, "UserSongLink added: " + song.getName());

view.added();

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "Error: Song not added: " + song.getName());

}

});

}

public void removeUserSongLink(final ILinkAdd<Song> view) {

final Song song = view.getItem();

String url = "userSongLink/delete/" + AppController.user.getId() + "/" + song.getId();

apiController.getStringResponse(Request.Method.DELETE, url, new VolleyCallbackString() {

@Override

public void onSuccessResponse(String result) {

if(result.equals("true")) {

view.removed();

Log.d(TAG, "UserSongLink removed: " + song.getName());

}

}

@Override

public void onErrorResponse(VolleyError error) {

Log.d(TAG, "UserSongLink not removed: " + error.getMessage());

}

});

}

}

* + - 1. Файл interfaces/ILinkAdd.java

package com.siroytman.indiewindymobile.interfaces;

public interface ILinkAdd<T> {

T getItem();

void removed();

void added();

}

* + - 1. Файл interfaces/ ILinkEmpty.java

package com.siroytman.indiewindymobile.interfaces;

public interface ILinkEmpty {

boolean isEmpty();

void makeEmpty();

}

* + - 1. Файл interfaces/ISearchableAlbum.java

package com.siroytman.indiewindymobile.interfaces;

import com.siroytman.indiewindymobile.model.UserAlbumLink;

import com.siroytman.indiewindymobile.model.UserArtistLink;

import java.util.ArrayList;

public interface ISearchableAlbum {

void albumsFoundViewUpdate(ArrayList<UserAlbumLink> links);

}

* + - 1. Файл interfaces/ ISearchableArtist.java

package com.siroytman.indiewindymobile.interfaces;

import com.siroytman.indiewindymobile.model.UserArtistLink;

import com.siroytman.indiewindymobile.model.UserSongLink;

import java.util.ArrayList;

public interface ISearchableArtist {

void artistsFoundViewUpdate(ArrayList<UserArtistLink> links);

}

* + - 1. Файл interfaces/ ISearchableSong.java

package com.siroytman.indiewindymobile.interfaces;

import com.siroytman.indiewindymobile.model.UserSongLink;

import java.util.ArrayList;

public interface ISearchableSong {

void songsFoundViewUpdate(ArrayList<UserSongLink> links);

}

* + - 1. Файл model/Album.java

package com.siroytman.indiewindymobile.model;

import android.os.Parcel;

import android.os.Parcelable;

import android.util.Log;

import org.json.JSONException;

import org.json.JSONObject;

public class Album implements Parcelable {

public static final String TAG = "Album";

// Required

private int id;

private String name;

private String imageUrl;

// Possible

private Artist artist;

public Album() { }

protected Album(Parcel in) {

id = in.readInt();

name = in.readString();

imageUrl = in.readString();

artist = in.readParcelable(Artist.class.getClassLoader());

}

public static final Creator<Album> CREATOR = new Creator<Album>() {

@Override

public Album createFromParcel(Parcel in) {

return new Album(in);

}

@Override

public Album[] newArray(int size) {

return new Album[size];

}

};

@Override

public int describeContents() {

return 0;

}

@Override

public void writeToParcel(Parcel dest, int flags) {

dest.writeInt(id);

dest.writeString(name);

dest.writeString(imageUrl);

dest.writeParcelable(artist, flags);

}

public static Album Parse(JSONObject jsonObject) {

Album album = new Album();

try {

album.setId(jsonObject.getInt("id"));

album.setName(jsonObject.getString("name"));

album.setImageUrl(jsonObject.getString("imageUrl"));

} catch (JSONException e)

{

Log.d(TAG, "Error: " + e.getMessage());

return null;

}

try {

album.artist = Artist.parse(jsonObject.getJSONObject("artist"));

} catch (JSONException e) {

Log.d(TAG, "No artist");

}

return album;

}

public static Album Parse(JSONObject jsonObject, Artist artist) {

try {

Album album = new Album();

album.setId(jsonObject.getInt("id"));

album.setName(jsonObject.getString("name"));

album.setImageUrl(jsonObject.getString("imageUrl"));

album.setArtist(artist);

return album;

} catch (JSONException e)

{

Log.d(TAG, "Error: " + e.getMessage());

return null;

}

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getImageUrl() {

return imageUrl;

}

public void setImageUrl(String imageUrl) {

this.imageUrl = imageUrl;

}

public Artist getArtist() {

return artist;

}

public void setArtist(Artist artist) {

this.artist = artist;

}

}

* + - 1. Файл model/AppUser.java
      2. Файл model/Artist.java
      3. Файл model/ArtistPostLink.java
      4. Файл model/Concert.java
      5. Файл model/Post.java
      6. Файл model/Song.java
      7. Файл model/UserAlbumLink.java
      8. Файл model/UserArtistLink.java
      9. Файл model/UserConcertLink.java
      10. Файл model/ UserSongLink.java
      11. Файл services/DownloadService.java
      12. Файл services/DownloadTracker.java
      13. Файл services/FragmentService.java
      14. Файл services/IconChangerService.java
      15. Файл services/KeyboardService.java
      16. Файл services/PlayerForegroundService.java
      17. Файл services/PlayerNotificationIntentService.java
      18. Файл services/ PlayerServiceConnection.java
      19. Файл services/ SharedPrefsService.java
      20. Файл ui/activity/AlbumActivity.java
      21. Файл ui/activity/ArtistActivity.java
      22. Файл ui/activity/ConcertActivity.java
      23. Файл ui/activity/DonationActivity.java
      24. Файл ui/activity/LoginActivity.java
      25. Файл ui/activity/MoreActivity.java
      26. Файл ui/activity/NavigationActivity.java
      27. Файл ui/activity/PlayerActivity.java
      28. Файл ui/activity/SettingsActivity.java
      29. Файл ui/adapter/ ArtistPostLinkListAdapter.java
      30. Файл ui/adapter/ ConcertPagerAdapter.java
      31. Файл ui/adapter/ HomePagerAdapter.java
      32. Файл ui/adapter/ PersonalPagerAdapter.java
      33. Файл ui/adapter/ UserAlbumLinkListAdapter.java
      34. Файл ui/adapter/ UserArtistLinkListAdapter.java
      35. Файл ui/adapter/ UserConcertLinkListAdapter.java
      36. Файл ui/adapter/ UserSongLinkListAdapter.java
      37. Файл ui/custom\_views/ MyListView.java
      38. Файл ui/fragments /concert/ ConcertFragment.java
      39. Файл ui/fragments /concert/ NearestConcertFragment.java
      40. Файл ui/fragments /concert/ SavedConcertFragment.java
      41. Файл ui/fragments /concert/ SubscriptionConcertFragment.java
      42. Файл ui/fragments /home/ HomeFragment.java
      43. Файл ui/fragments /home/ LatestHomeFragment.java
      44. Файл ui/fragments /home/ SubscriptionHomeFragment.java
      45. Файл ui/fragments /links/ ArtistPostLinkListFragment.java
      46. Файл ui/fragments /links/ UserAlbumLinkListFragment.java
      47. Файл ui/fragments /links/ UserArtistLinkListFragment.java
      48. Файл ui/fragments /links/ UserConcertLinkListFragment.java
      49. Файл ui/fragments /links/ UserSongLinkListFragment.java
      50. Файл ui/fragments /personal/ PersonalFragment.java
      51. Файл ui/fragments /personal/ AlbumsPersonalFragment.java
      52. Файл ui/fragments /personal/ ArtistsPersonalFragment.java
      53. Файл ui/fragments /personal/ SongsPersonalFragment.java
      54. Файл ui/fragments /search SearchFragment.java/

1. ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |