#9 Разработка приложений UWP

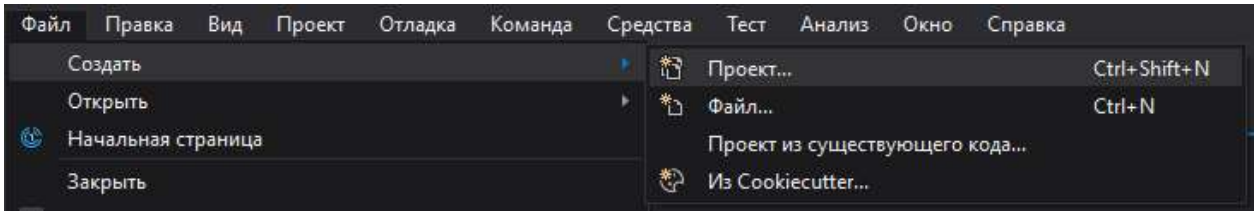
Зачем делать эту лабораторную работу?

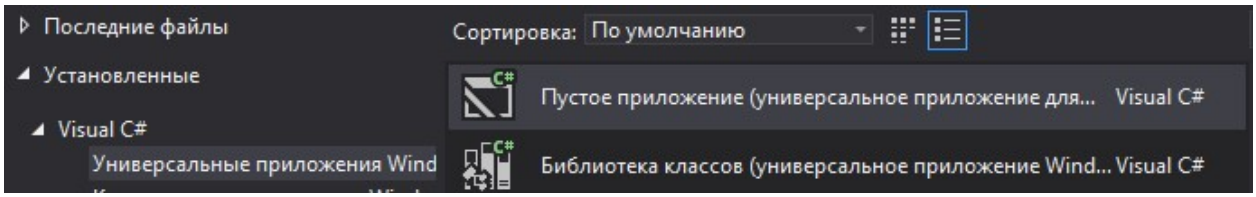
1. Чтобы научиться создавать UWP-приложения с использованием подключения к базе данных.
2. Чтобы получить представление о работе с данными в универсальных приложениях без использования серверных СУБД.
3. Чтобы освоить интерфейс Entity Framework и способы работы с данными LINQ-запросами.

Что нужно делать?

1. Создать проект приложения UWP.

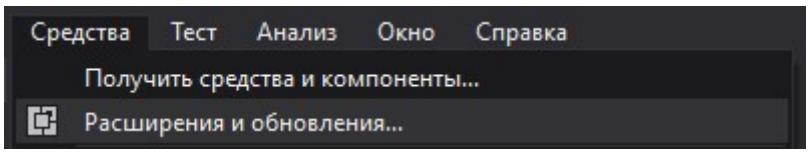
В окне Создание проекта выберите проект Пустое приложение (универсальное приложение для Windows.

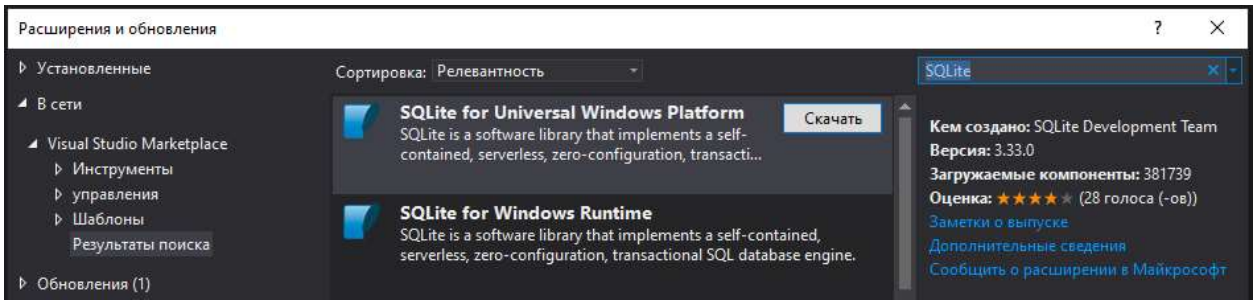




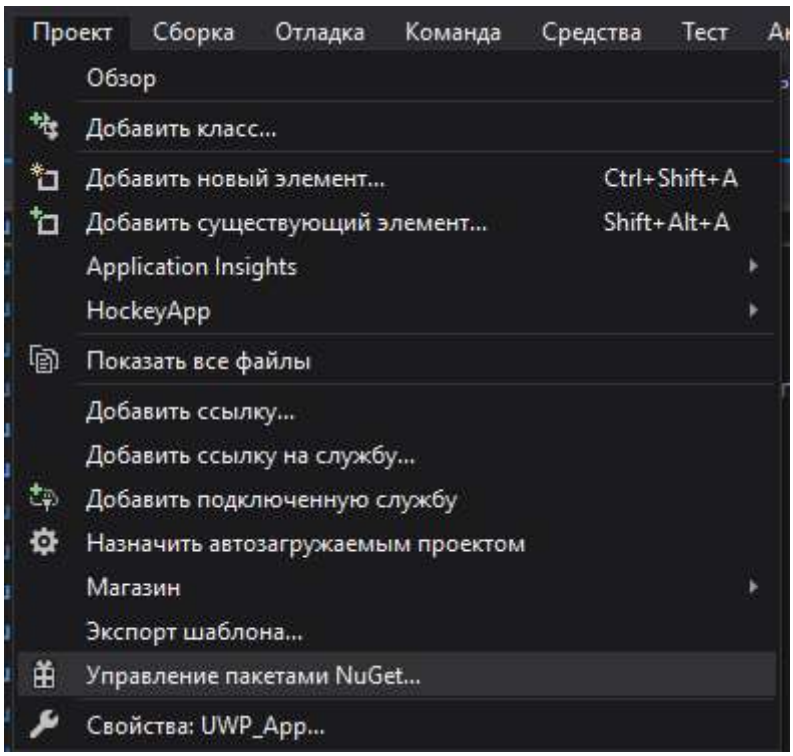
2. Установить СУБД SQLite.

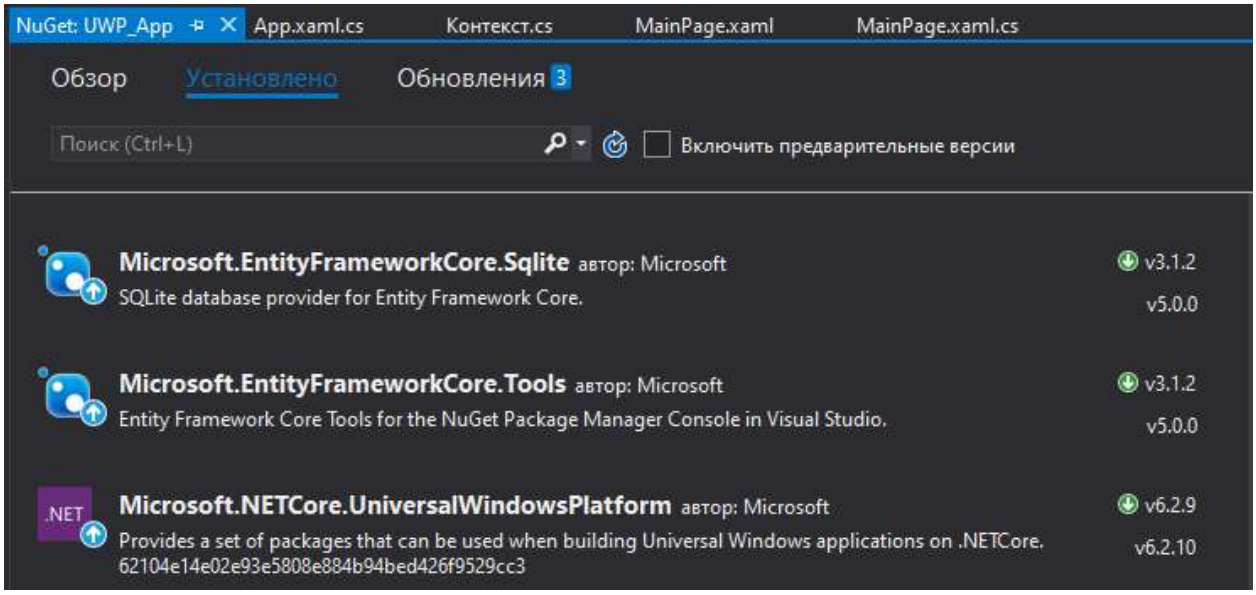
В окне Расширения и обновления найдите и скачайте SQLite for Universal Windows Platform.





3. Установить соответствующие версии пакетов Entity Framework Core





4. Описать классы модели базы данных, которая будет использоваться в проекте.

Для приложения, позволяющего пользователю выбрать исполнителя, альбом и трек последовательно, пользуясь раскрывающимися списками, и его последующей загрузки в плеер, необходимо описать структуру двух таблиц:

Альбомы (Код_альбома, Название, Исполнитель, Обложка)

Треки (Код_трека, Название_трека, Адрес, Код_альбома)

Класс, описывающий таблицу Альбомы выглядит так:

```
namespace UWP_App
{
    public class Альбом
```

```
    {
        public int Id { get; set; }
        public string Название { get; set; }
        public string Исполнитель { get; set; }
        public byte[] Обложка { get; set; }

        public List<Трек> Треки { get; set; }

        public override string ToString()
        {
            return Название;
        }
    }
}
```

Класс, описывающий таблицу Треки выглядит так:

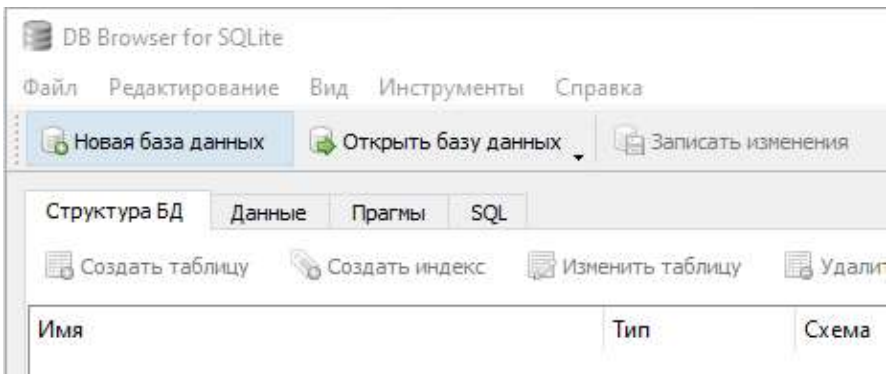
```
namespace UWP_App
{
    public class Трек
    {
        public int Id { get; set; }
        public string Название_трека { get; set; }
        public string Адрес { get; set; }
        public int Код_альбома { get; set; }
        public Альбом Альбом { get; set; }
    }
}
```

5. Описать класс контекста данных для взаимодействия приложения с базой данных.

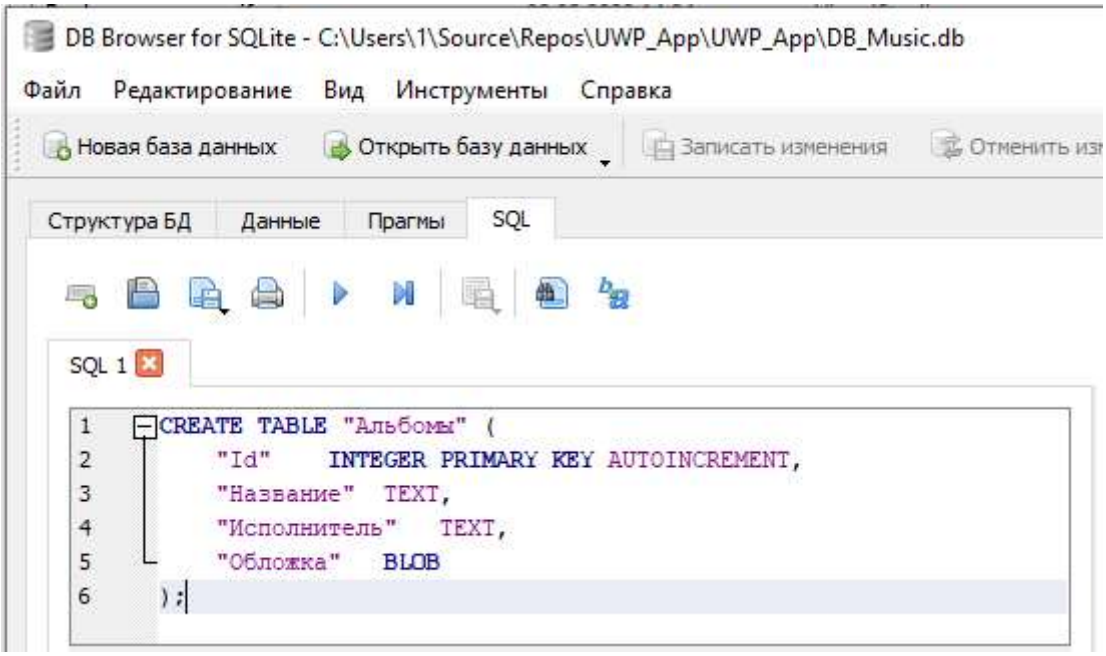
```
namespace UWP_App
{
    public class Контекст: DbContext
    {
        public DbSet<Альбом> Альбомы { get; set; }
        public DbSet<Трек> Треки { get; set; }

        public Контекст()
        {
            Database.EnsureCreated();
        }
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlite("Filename=DB_Music.db");
        }
    }
}
```

6. Создать базу данных, содержащую необходимые таблицы.

Для создания базы данных можно использовать SQLite Browser:



Создать базу данных можно в режиме конструктора или выполнив SQL инструкции.

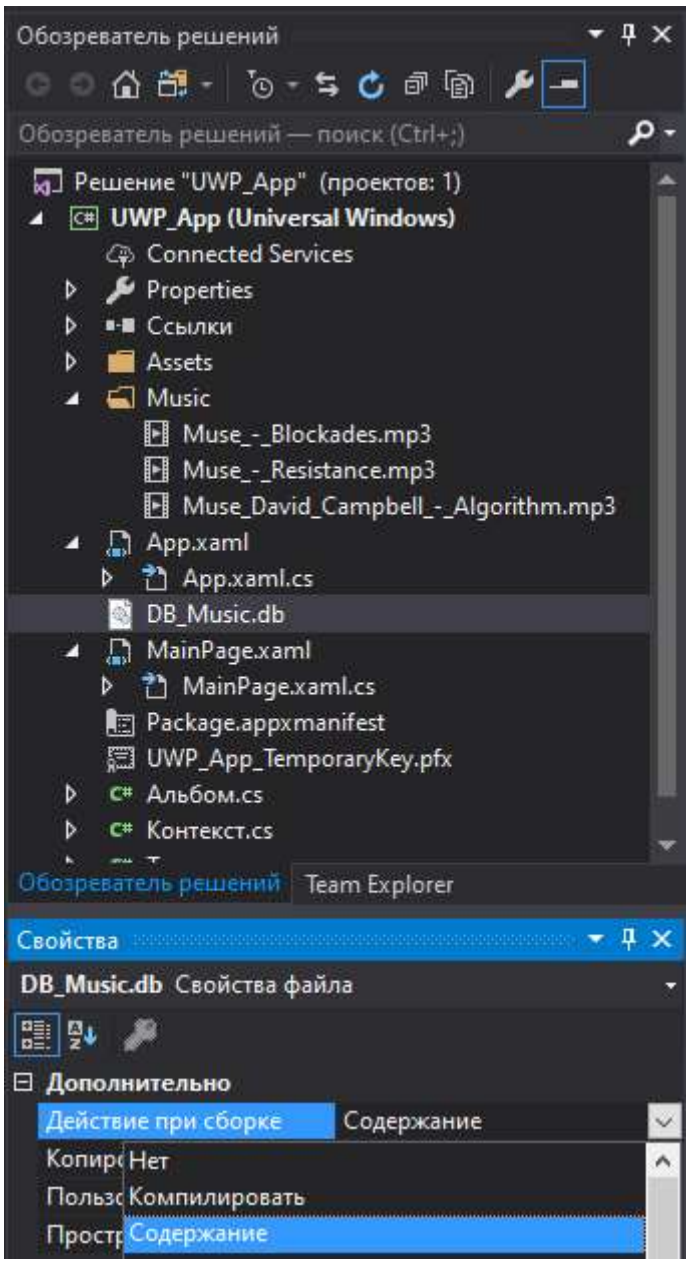


Код SQL-инструкций для создания таблиц Альбомы и Треки будет выглядеть так:

```
CREATE TABLE "Треки" (
	"Id"	INTEGER PRIMARY KEY AUTOINCREMENT,
	"Название_трека"	TEXT,
	"Адрес"	TEXT,
	"Код_альбома"	INTEGER,
	FOREIGN KEY("Код_альбома") REFERENCES "Альбомы"("Id") ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE "Альбомы" (
	"Id"	INTEGER PRIMARY KEY AUTOINCREMENT,
	"Название"	TEXT,
	"Исполнитель"	TEXT,
	"Обложка"	BLOB
);
```

7. Добавить базу данных в созданный проект.

Добавьте файл базы данных с расширением .db в папку проекта и установите для свойства Build Action (Действия при сборке) значение Content (Содержание).



8. Обеспечить копирование базы данных при первом запуске приложения.

Копирование выполняется в папку проекта ApplicationData.Current.LocalFolder:

```
namespace UWP_App
{
    sealed partial class App : Application
    {
        public App()
        {
            this.InitializeComponent();
            this.Suspending += OnSuspending;
        }

        protected override async void OnLaunched(LaunchActivatedEventArgs e)
        {
            if (await ApplicationData.Current.LocalFolder.TryGetItemAsync("DB_Music.db")
== null)
            {
                StorageFile databaseFile = await
Package.Current.InstalledLocation.GetFileAsync("DB_Music.db");
                await databaseFile.CopyAsync(ApplicationData.Current.LocalFolder);
```

```
            }
        //.... остальное содержимое метода OnLaunched
        }
    //.... остальное содержимое класса App
    }
}
```

9. Реализовать интерфейс приложения.

Для приведённого выше задания можно разместить элементы управления, описанные в коде XAML. Также в коде выполняются привязки данных к элементам управления, с которыми взаимодействует пользователь:

```
<Page
    x:Class="UWP_App.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:UWP_App"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>
        <StackPanel>
            <ComboBox x:Name="artistsList"
SelectionChanged="ArtistsList_SelectionChanged">
                <ComboBox.ItemTemplate>
                    <DataTemplate>
                        <StackPanel>
                            <TextBlock Text="{Binding}" FontSize="26" />
                        </StackPanel>
                    </DataTemplate>
                </ComboBox.ItemTemplate>
            </ComboBox>
            <ComboBox x:Name="albumsList" SelectionChanged="AlbumsList_SelectionChanged">
                <ComboBox.ItemTemplate>
                    <DataTemplate x:DataType="local:Альбом">
                        <StackPanel>
                            <TextBlock Text="{Binding Название}" FontSize="26" />
                        </StackPanel>
                    </DataTemplate>
                </ComboBox.ItemTemplate>
            </ComboBox>
            <ComboBox x:Name="trackList" SelectionChanged="TrackList_SelectionChanged">
                <ComboBox.ItemTemplate>
                    <DataTemplate x:DataType="local:Трек">
                        <StackPanel>
                            <TextBlock Text="{Binding}" FontSize="26" />
                        </StackPanel>
                    </DataTemplate>
                </ComboBox.ItemTemplate>
            </ComboBox>
            <MediaElement x:Name="media" HorizontalAlignment="Stretch" Height="100"
Margin="0,0,0,0" VerticalAlignment="Stretch"
                          AreTransportControlsEnabled="True"/>
        </StackPanel>

    </Grid>
</Page>
```

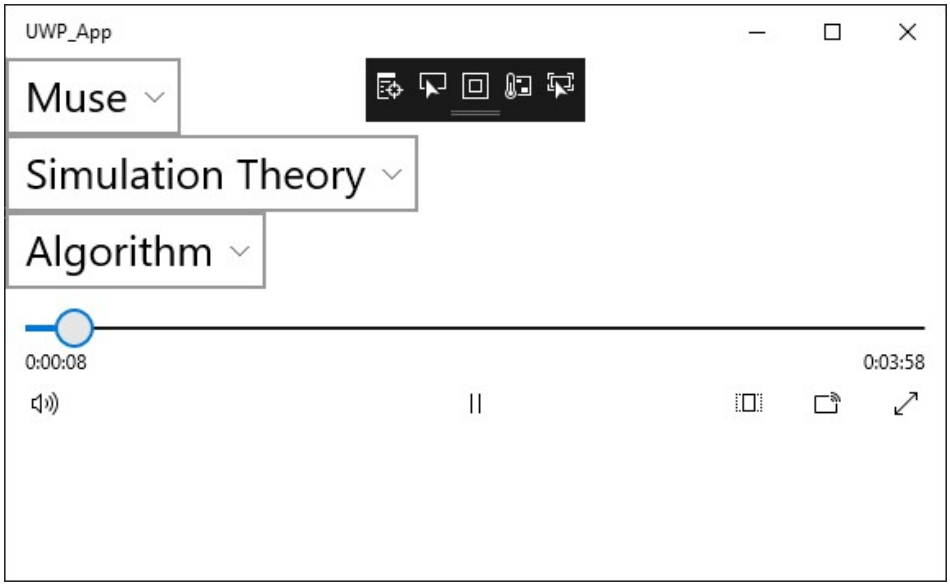
10. Реализовать функциональную часть приложения.

Поскольку в проекте используется Entity Framework, получение данных удобно реализовывать LINQ-запросами.

```
namespace UWP_App
{
    /// <summary>
    /// Пустая страница, которую можно использовать саму по себе или для перехода внутри
фрейма.
    /// </summary>
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
            this.Loaded += MainPage_Loaded;
        }

        private void MainPage_Loaded(object sender, RoutedEventArgs e)
        {
            using (Контекст db = new Контекст())
            {
                List<String> albums = db.Альбомы.Select(p =>
p.Исполнитель).Distinct().ToList();

                artistsList.ItemsSource = albums;

            }
        }

        private void ArtistsList_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            using (Контекст db = new Контекст())
            {

                albumsList.ItemsSource = db.Альбомы.Where(p => p.Исполнитель==
e.AddedItems[0].ToString()).ToList();

            }

        }

        private void AlbumsList_SelectionChanged(object sender, SelectionChangedEventArgs
e)
        {
            using (Контекст db = new Контекст())
            {
                int n = db.Альбомы.Where(c => c.Исполнитель ==
artistsList.SelectedItem.ToString()
                                        && c.Название ==
albumsList.SelectedItem.ToString()).Select(p => p.Id).FirstOrDefault();

                trackList.ItemsSource = db.Треки.Where(p => p.Код_альбома == n).Select(c
=> c.Название_трека).ToList();
            }
        }

        private async void TrackList_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            using (Контекст db = new Контекст())
            {
                int n = db.Треки.Where(c => c.Название_трека ==
e.AddedItems[0].ToString()).Select(p => p.Id).FirstOrDefault();
```

```
                string path = db.Треки.Where(c => c.Id == n && c.Адрес!=null).Select(p =>
p.Адрес).FirstOrDefault().ToString();
                ContentDialog noWifiDialog1 = new ContentDialog()
                {
                    Content = path,
                    CloseButtonText = "Ok"
                };

                await noWifiDialog1.ShowAsync();

                var file = await
Package.Current.InstalledLocation.GetFileAsync("music\\"+path);
                var stream = await file.OpenReadAsync();
                media.SetSource(stream, "");
                media.Play();


            }
        }
    }
}
```

Результат работы приложения:



Как узнать, что все выполнено?

Проверьте пункты в этом чек-листе:

☐ Создана база данных с необходимыми таблицами

☐ Таблицы заполнены тестовыми данными

☐ В приложении описана структура классов для таблиц базы данных

☐ Для управления данными в проекте используются классы Entity Framework

☐ Работа с данными осуществляется с помощью LINQ-запросов

☐ Приложение работает без ошибок

☐ Работа приложения протестирована на различных наборах данных

☐ В приложении предусмотрена обработка исключений

☐ Интерфейс приложения выполнен эргономично

Варианты заданий:

1 вариант:
В базе данных создать следующие таблицы:
Треки (Код трека, Название трека, Адрес, Код альбома);
Альбомы (Код альбома, Название, Исполнитель, Обложка).
Разработать приложение, позволяющие пользователю загружать обложки для альбомов. При выборе альбома осуществляется загрузка изображения обложки, если оно существует.

2 вариант:
В базе данных создать следующие таблицы:
Треки (Код трека, Название трека, Адрес, Код альбома);
Альбомы (Код альбома, Название, Исполнитель, Обложка).
Разработать приложение, позволяющие пользователю создавать собственные плейлисты. Выбор треков осуществляется путем добавления в список, а затем переноса его элементов в плейлист.

3 вариант:
В базе данных создать следующие таблицы:
Треки (Код трека, Название трека, Адрес, Код альбома);
Альбомы (Код альбома, Название, Исполнитель, Обложка).
Разработать приложение, позволяющие в произвольном порядке выбирать музыку из базы данных и проигрывать в плеере. Пользователь может ограничивать набор треков путем выбора альбомов или исполнителей.

4 вариант:
В базе данных создать следующие таблицы:
Треки (Код трека, Название трека, Адрес, Код альбома);
Альбомы (Код альбома, Название, Исполнитель, Обложка).
Разработать приложение, позволяющие пользователю редактировать базу данных с музыкальными записями. При добавлении должен быть обеспечен просмотр хранящихся в таблицах данных.

5 вариант:
В базе данных создать следующие таблицы:
Фотографии (Код фотографии, Код пользователя, Адрес, Комментарий автора)
Комментарии (Код фотографии, Код пользователя, Комментарий)
Разработать приложение, позволяющие пользователю просматривать фотографии в виде слайд-шоу, увеличивать и уменьшать фотографии.

6 вариант:
В базе данных создать следующие таблицы:
Фотографии (Код фотографии, Код пользователя, Адрес, Комментарий автора)
Комментарии (Код фотографии, Код пользователя, Комментарий)
Пользователи (Код пользователя, Никнейм, Аватар)
Разработать приложение, позволяющие пользователям оставлять комментарии к фотографиям. При этом необходимо предоставить пользователю возможность скрыть свое имя.

7 вариант:
В базе данных создать следующие таблицы:
Фотографии (Код фотографии, Код пользователя, Адрес, Комментарий автора)
Комментарии (Код фотографии, Код пользователя, Комментарий)
Разработать приложение, позволяющие пользователю загружать в базу данных фотографии и делать подписи. Все фотографии должны просматриваться перед добавлением.

8 вариант:
В базе данных создать следующие таблицы:
Фотографии (Код фотографии, Код пользователя, Адрес, Комментарий автора)
Комментарии (Код фотографии, Код пользователя, Комментарий)
Пользователи (Код пользователя, Никнейм, Аватар)
Разработать приложение, позволяющие пользователям просматривать фотографии и сохранять понравившиеся в качестве аватара. Для этого необходимо уменьшить размер картинки перед загрузкой в базу данных.

9 вариант:
Фотографии (Код фотографии, Код пользователя, Адрес, Комментарий автора)
Пользователи (Код пользователя, Никнейм, Аватар)
Отметки на фото (Код фотографии, Код пользователя, Область на фото)
Разработать приложение, позволяющие пользователям отмечать на фотографиях других пользователей. Координаты отмеченной области должны сохраняться в базе данных.

10 вариант:
Фотографии (Код фотографии, Код пользователя, Адрес, Комментарий автора)
Пользователи (Код пользователя, Никнейм, Аватар)
Отметки на фото (Код фотографии, Код пользователя, Область на фото)
Разработать приложение, позволяющие пользователям просматривать все фото, на которых они отмечены. Предусмотреть возможность удаления отметки.

11 вариант:
Видеозаписи (Код видеозаписи, Код пользователя, Адрес, Комментарий автора)
Пользователи (Код пользователя, Никнейм, Аватар)
Разработать приложение, позволяющие пользователю добавлять и просматривать видеофайлы. Вся информация должна сохраняться в базе данных.

12 вариант:
Видеозаписи (Код видеозаписи, Код пользователя, Адрес, Комментарий автора)
Пользователи (Код пользователя, Никнейм, Аватар)

Разработать приложение, позволяющие пользователям просматривать видеозаписи других пользователей. Все видеозаписи выбранного пользователя должны загружаться в плейлист.

13 вариант:
Кадры (Код видеозаписи, Код изображения)
Изображения (Код изображения, Адрес, Комментарий)
Разработать приложение, позволяющие пользователям добавлять и просматривать кадры к фильмам. Просмотр осуществляется в плеере.

14 вариант:
Фильмы (Код видеозаписи, Название, Адрес, Комментарий)
Кадры (Код видеозаписи, Код изображения)
Изображения (Код изображения, Адрес, Комментарий)
Разработать приложение, позволяющие пользователям выбирать фильмы. При выборе названия должны просматриваться кадры из фильма, а затем, по желанию пользователя, сам фильм.

15 вариант:
Изображения (Код изображения, Адрес, Комментарий)
Комментарии (Код фотографии, Код пользователя, Комментарий)
Пользователи (Код пользователя, Никнейм, Аватар)
Разработать приложение, позволяющие пользователям оставлять комментарии к кадрам фильма. Каждый пользователь может оставить только один комментарий к изображению.

16 вариант:
В базе данных создать следующие таблицы:
Треки (Код трека, Название трека, Адрес, Код альбома)
Альбомы (Код альбома, Название, Исполнитель, Обложка)
Разработать приложение, позволяющие пользователю миксовать треки из базы данных. Полученный список необходимо хранить в виде текстового файла и загружать в плеер при необходимости.

17 вариант:
Фильмы (Код видеозаписи, Название, Адрес, Комментарий)
Рейтинги (Код фильма, Рейтинг)
Разработать приложение, позволяющие пользователю просматривать фильм и влиять на рейтинг. Шкала рейтинга должна отображаться визуально, и составлять не менее 5 пунктов.

18 вариант:
Фильмы (Код видеозаписи, Название, Адрес, Комментарий)
Рейтинги (Код фильма, Рейтинг)
Разработать приложение, позволяющие пользователю просматривать фильм и влиять на рейтинг. При этом каждый пользователь может сделать оценку только один раз.

19 вариант:
Фильмы (Код видеозаписи, Название, Жанр, Адрес, Комментарий)
Рейтинги (Код фильма, Рейтинг)

Разработать приложение, позволяющие пользователю осуществлять поиск фильмов путем варьирования значения рейтинга и выбора жанра.

20 вариант:
Фильмы (Код видеозаписи, Название, Жанр, Адрес, Комментарий)
Рейтинги (Код фильма, Рейтинг)
Разработать приложение, позволяющие пользователю получать информацию о фильмах с самыми высокими рейтингами. При выводе фильмы необходимо упорядочить по жанрам.