

StoreMusicApi front-end application:

Used: **C#, WPF, .NET 6.0, Entity Framework, MS_SQL, Swagger**(main) documentation. Page), **Microsoft Azure cloud, NLogger, JWT Token, Postman, CORS policy, Middleware.**

The desktop application allows you to browse (without logging in) as a guest, then you can only Browse all artists, albums and songs. You can log in as **USER** -> who can only browse everything, without access to the **ContactNumber** and **ContactEmail** fields of a given Artist. Then you can log in as **PREMIUM_USER** -> the ability to create your Artists with unique names so that they don't duplicate in the list that already exists for that user.

Create albums only for your artist and songs for your album. Artist, Album, Song names can duplicate between users, that is, the first user will create an artist named **Iron Maiden 1** with id -> **1**, and the second user will create an artist named **Iron Maiden 1** with id -> **2**, this situation is allowed. It is possible to log in as **ADMIN** -> create, delete, add, view, only if it creates the name of the artist, Album, Song then it must be a unique name. In the login window, you can register and after logging in, each user has a generated **JWT Token** through which authentication is checked to see if they have access to the activity. For example, a user who has the role **USER, PREMIUM_USER** when he wants to update an artist, album, song not created by him, create a new song, album in an artist not created by him, an error message -> Status code: 403 -> Forbidden will occur. If logged in as a guest wants to access an option forbidden for him then he gets a message that he is not logged in and has no access because he does not have the given JWT Token.

In the **User Details** -> there are details of a given user and the artists he created, plus additional data **ContactEmail** and **ContactNumber**.

In the Show Artists window, you can add, update, delete, see details of a particular artist. You can use pagination: **Record per page** -> **5, 10, 15** number of records per page. **1 of 30** -> 1-> current page, **30**-> number of pages of all, current page changes relative to the selection of a page with additional buttons '>' -> **nextPage**, '<' -> **PreviousPage**, '>>' -> **LastPage**, '<<' -> **FirstPage**. Ability to sort using **SortBy**-> against 3 columns: **Name, Description, KindOfMusic, SortDirection** -> **ASC**: ascending, **DESC**: descending sorts by the given column selected. **Search** button -> searches for a given word by **Name** column. Everything works when the **Search** button is pressed.

In the **Show Albums and Songs** window it works similarly, in addition you can use the **Delete All** -> button.

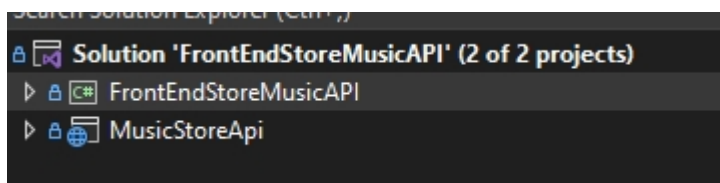
deleting all records. The **Show All Albums** -> window automatically displays all albums from the database. The **Show All Songs** -> window automatically displays all songs from the database. The **Show All Artists** -> window automatically displays all the artists in the database. When you return from the **ShowAllSongsWindow** to the **ShowAlbumsWindow**, it automatically display all the albums of a given artist. Using **UpdateCreateArtist, UpdateCreateAlbum, UpdateCreateSong** window -> for two actions, to create and update. Using **ShowAllAlbumsWindow** to display all albums from the database or to

Displaying all the albums of a given artist. Using the **ShowAllSongsWindow** to display all songs in the database or to display all songs of a given album.

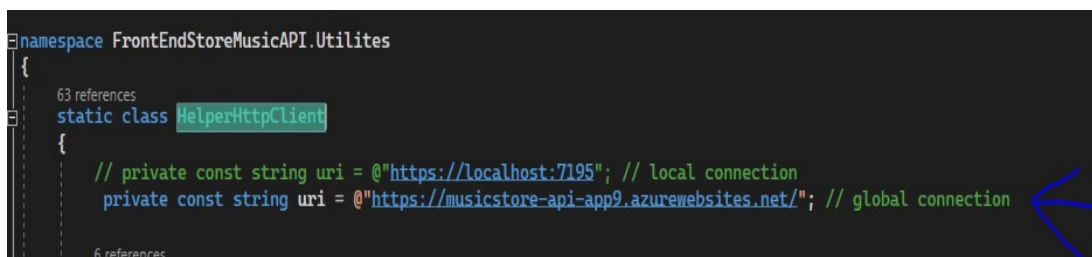
When hovering the mouse cursor over any button, text box, additional information appears what the button is responsible for. The ability to connect the Frontend application directly to the web application, which is on the **Azure cloud**, at the link : <https://musicstore->

api-app9.azurewebsites.net , you just need to change the **uri** field in the **Utilites/HelperHttpClient.cs** directory. During the first launch locally will create records for testing and 3 users also for testing.

The desktop application was created to connect to the **MusicStoreApi** backend application in 3 ways:



1. It connects directly to the **web API** , which is together with the Frontend application in the project directory, only you need to set the value of the **uri** variable to the correct local connection: <https://localhost:7195> -> port number on which the web API application is running, this can be made in the directory **MusicStoreApi/Properties/launchSettings.json**, it is best to run via **HTTPS**, then the application works without problems.
2. It connects to the same **web API** , which is running as another project on the local computer and changing the uri variable as described above and runs over **HTTPS**.

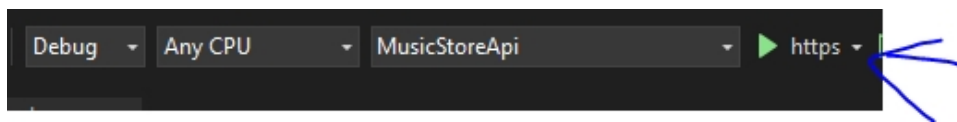


3. It connects to the same **web API** that is running in the cloud: only you need to set the correct value of the uri variable to <https://musicstore-api-app9.azurewebsites.net/swagger/index.html> , which is located in the **Utilites/HelperHttpClient.cs** directory .

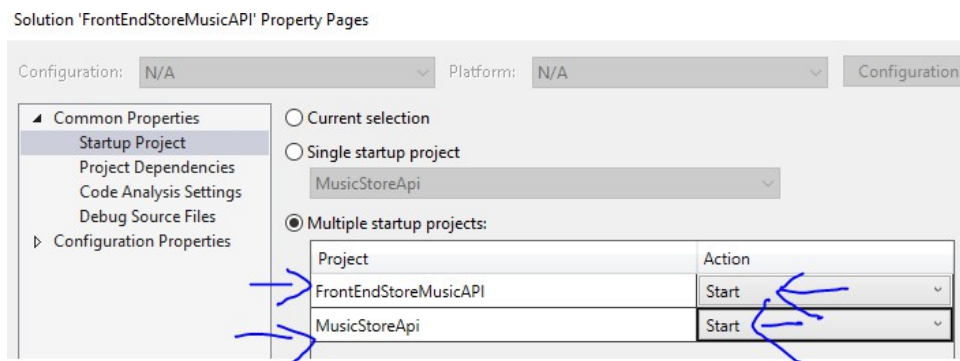
Description of how the application works:

1. Application Launch:
 - Download from github -> **Code** -> **Download Zip**.
 - Open applications in **Visual Studio Community 2022**

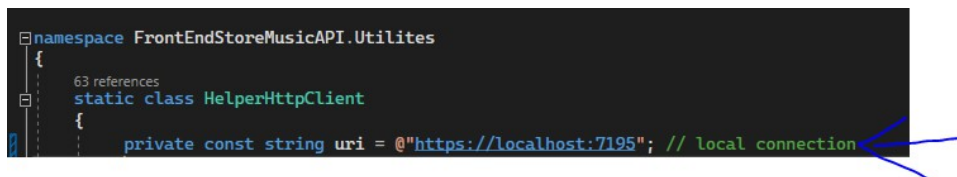
Running from the **Web Api** , which is included in the project, is:



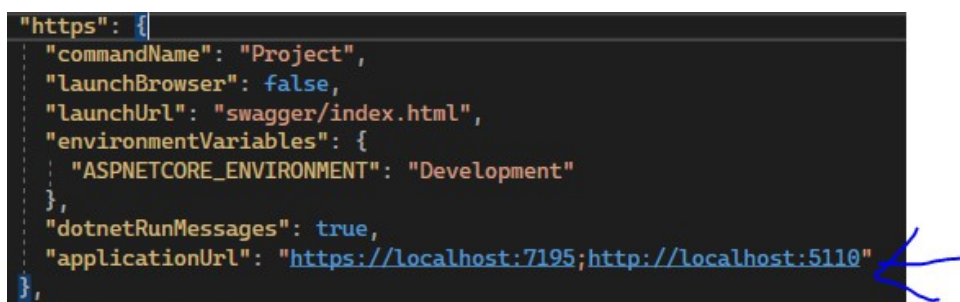
- Make sure you have the **Web Api** running on **HTTPS**.
- Then go into **Https-> Configure Startup Projects**



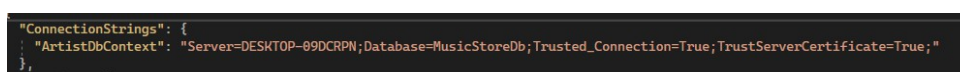
- Set both projects to **Start** .



- Go to the **FrontEndStoreMusicAPI/Utilites/HelperHttpClient.cs** directory .



- Make sure you have the correct address of the variable **uri** , which you can check in the directory **MusicStoreApi/Properties/LaunchSettings.json** in **HTTPS**.



- Enter a valid database connection in the directory **MusicStoreApi/appsettings.Development.json**

- Then enter the **Package Manager Console** and enter the **update-database** command, If no migration, then **add-migration Init**, then **update-database**.
- If everything works correctly, then launch the applications. At the first launch, sample data in the database will be created , so that you can test and use the application.

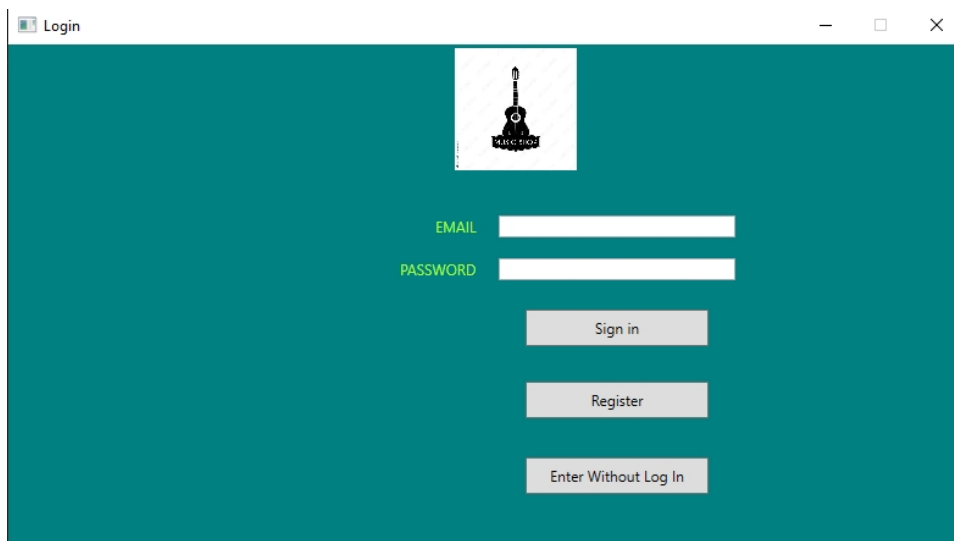
Running from the **Web Api** , which resides in the **Azure cloud**, is:

```
private const string uri = @"https://musicstore-api-app9.azurewebsites.net/"; // global connection
```

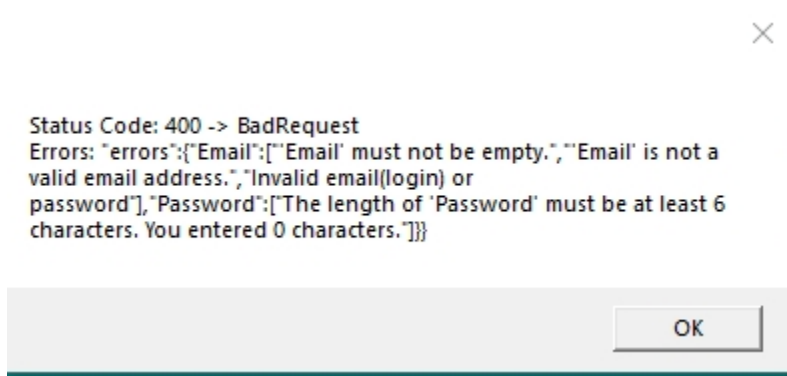
6 references

- Make sure you have the correct uri variable address at <https://musicstore-api-app9.azurewebsites.net>
- Run only the Frontend application and test the applications.
- If you get the error "**Microsoft.Data.SqlClient.SqlException: 'There is already an object named "Addresses" in the database.'**" -> then in **Package Manager Console** enter the commands: **drop-database**, then **update-database**, if everything passed ok, then run the frontend-only application.

Running application:



1. **Login window:**
 - To log in you need to enter your **email** and **password**




- If nothing is specified then an information window about the error data will pop up:

Email -> cannot be empty, it is checked if there is a valid address, Invalid email or password: reports an invalid email.

Password -> must have at least 6 characters, reports if password is invalid.

Register

Return



FIRST NAME

LAST NAME

EMAIL

PASSWORD

CONFIRM PASSWORD

NATIONALITY

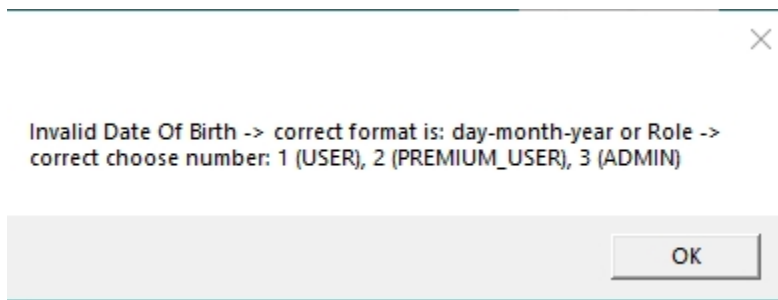
DATE OF BIRTH

ROLE

Save

Clear

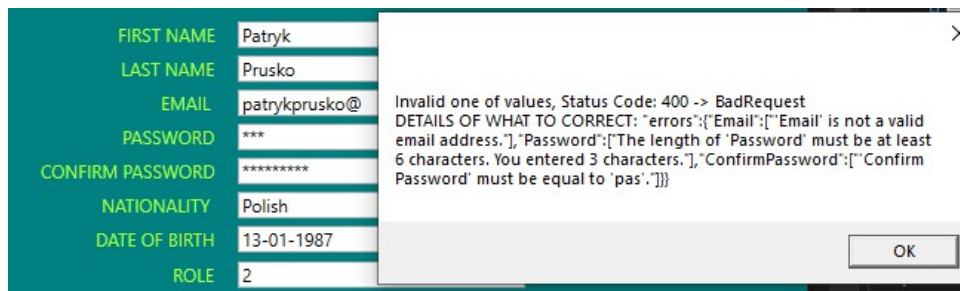
- The **Register** -> button opens a new registration window where you can create your new account and be added as a new user to the database.



- Error information window:

DateOfBirth -> must have **day-month-year** format e.g. 13-01-1987.

Roles -> 1, 2, 3 (USER, PREMIUM_USER, ADMIN).

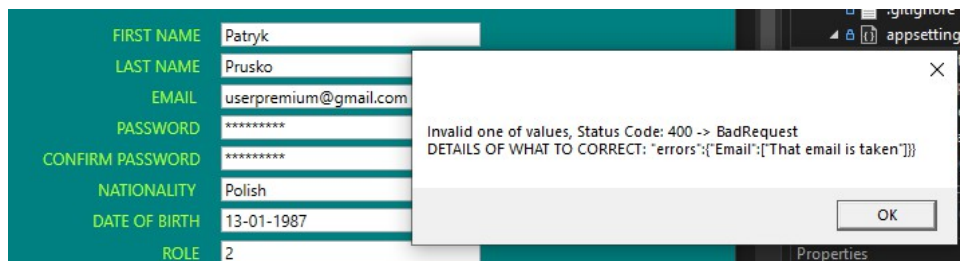


The screenshot shows a registration form with the following fields: FIRST NAME (Patryk), LAST NAME (Prusko), EMAIL (patrykprusko@), PASSWORD (***), CONFIRM PASSWORD (*****), NATIONALITY (Polish), DATE OF BIRTH (13-01-1987), and ROLE (2). An error message window is displayed over the form, stating: "Invalid one of values, Status Code: 400 -> BadRequest. DETAILS OF WHAT TO CORRECT: 'errors':{'Email':{'Email' is not a valid email address.'}, 'Password':{'The length of 'Password' must be at least 6 characters. You entered 3 characters.'}, 'ConfirmPassword':{'Confirm Password' must be equal to 'pas'.'}}". An "OK" button is visible in the bottom right of the error window.

The following error information window:

Email -> validates,

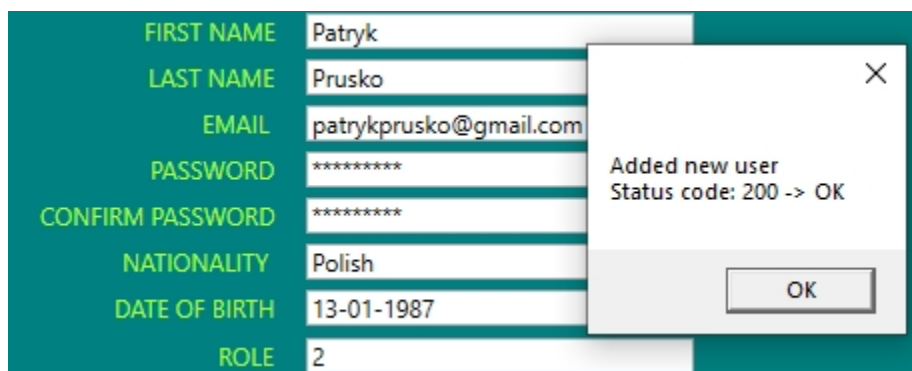
Password -> must have at least 6 characters, must be the same as **ConfirmPassword**.



The screenshot shows the same registration form as before, but with the EMAIL field filled with "userpremium@gmail.com". The error message window now states: "Invalid one of values, Status Code: 400 -> BadRequest. DETAILS OF WHAT TO CORRECT: 'errors':{'Email':{'That email is taken'}}". An "OK" button is visible in the bottom right of the error window.

The following error information window:

If you enter an email that already exists in the database it displays a message that it is already taken.



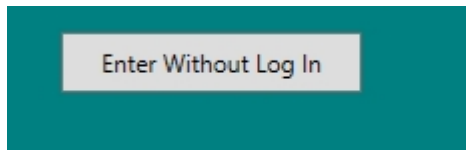
The screenshot shows the registration form with the EMAIL field filled with "patrykprusko@gmail.com". A success message window is displayed over the form, stating: "Added new user. Status code: 200 -> OK". An "OK" button is visible in the bottom right of the success window.

Adding a new user:

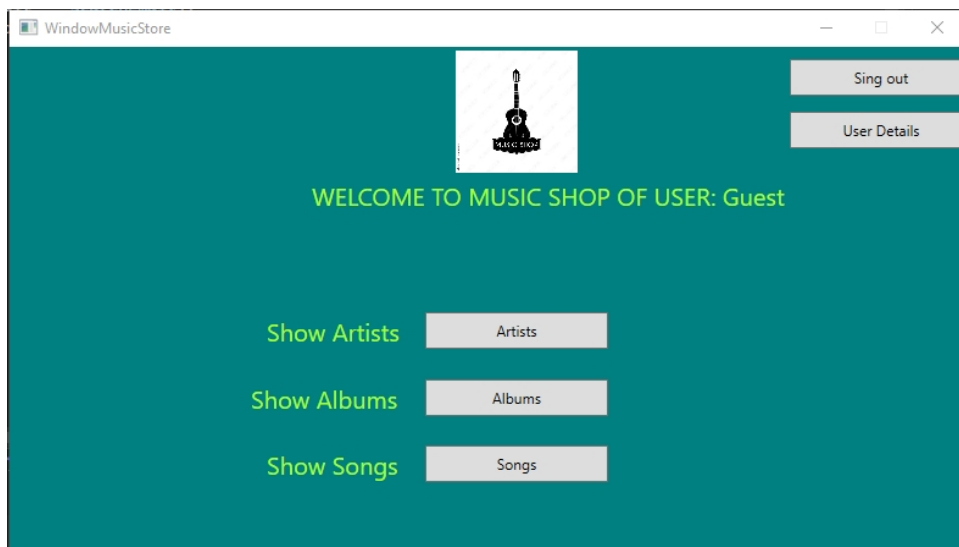
If everything is correct then it appears that a new user has been added and **status code 200**.

Then it goes to the login window. **Return** button -> return to login window, **Clear** button -> clears all fields, **Save** -> button creates and saves the new user.

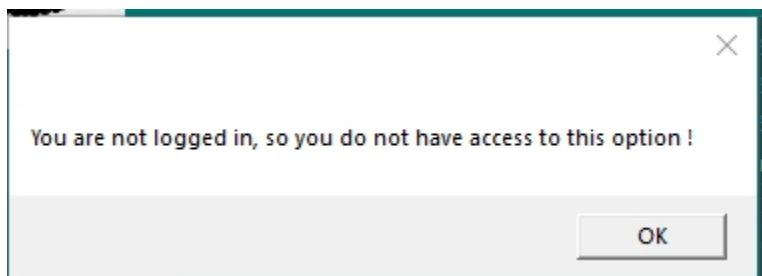
Operation of logging in as a **GUEST**:



- **Enter Without Log In** -> button logs in as a guest, a user who is not logged in.



The next main window of the Music Store opens. Username As: **Guest**.



If you try to use the **User Details** button then you will get a message -> **You are not logged in, so you do not have access to this option**. Only able to view **Artists, Albums, Songs**.

Acting as a logged-in user:

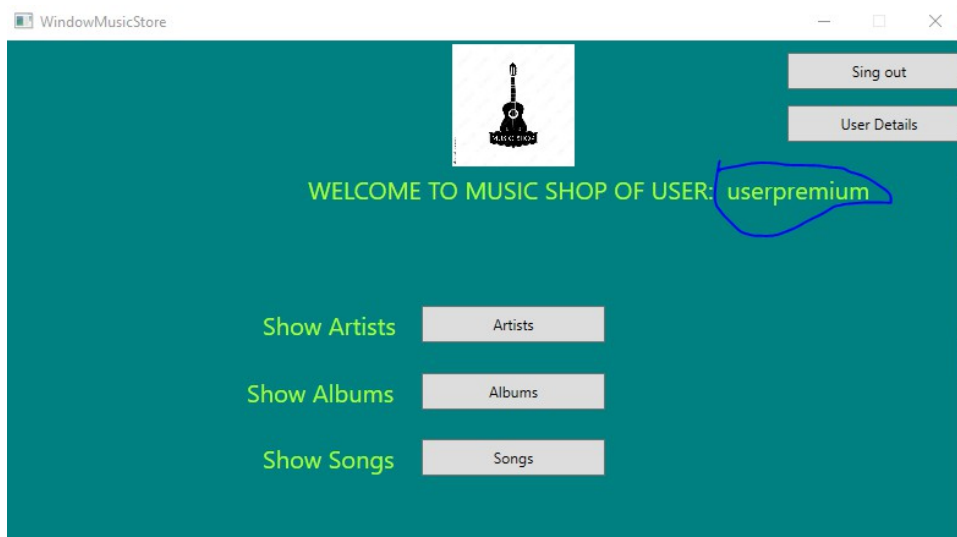
There are 3 options as:

- **USER** -> ability to only browse general and specific artists, albums, songs.
- **USER_PREMIUM** -> Ability to browse, create new artists, but with a unique name, so that it does not repeat in the list already created by a given user. Ability to create albums, only to the artists that a given user has created. The ability to create

Songs to albums that a given user has created. Delete and update artists, albums and songs only created by a given user. Only the given user_premium has access to the **ContactEmail** and **ContactNumber** data in the **Details of Artist** window or the **User Details** window to the data of the Artist he created.

- **ADMIN** -> ability to view, no access ContactEmail and ContactNumber of a given Artist if he did not create it. When it comes to and creating, updating then the names are to be unique.

Operation of the main window of Music Shop of User:



If you log in as, for example, **User_premium** then your **FirstName** will always appear at the top.

Operation of the **Show Artists** window:



A list of all artists in the database is immediately generated. Using **pagination** :

TotalPages	ItemFrom	ItemTo	TotalItemsCount
30	1	5	150

Records per page: 5 | 1 of 30 | < > << >> | Sort Direction: ASC | Sort By:

Records per page -> selection of 5, 10, 15 records per page. **NextPage**-> next page, **PreviousPage**-> earlier page, **FirstPage**-> first page, **LastPage**-> last page. **Sort Direction**-> **ASC**(ascending), **DESC**(descending) sorting, **Sort By** -> **Name**, **Description**, **Kind Of Music** must be selected for this to work. **TotalPages** -> number of pages to display, **totalItemsCount**-> result of all records. **1 of 30** -> 1-> current page, **30** -> number of pages to view, when changing the current page it displays which page you are currently on.

Search -> search button looks for the word in the Name value. To set all the changes you need to use this button.

Return




You are in the Details of Artist section

Id	Name	Description	KindOfMusic	Country	City	ContactEmail	ContactNumber	AlbumsSongs
95	Sepultura 95	information about ... 95	rock, metal 95	USA 95	New York 95	contact@email95.com	9874238595	Album: 1. Id: 189, Title: Armagedon 95, Length: 3.5, Nun Songs: 1. Id: 565, Name: Planet is good 1 95, AlbumId: 189 2. Id: 566, Name: This is a cat 2 95, AlbumId: 189 3. Id: 567, Name: So tired 3 95, AlbumId: 189 Album: 2. Id: 190, Title: Dance to death 95, Length: 3.5, 1 Songs: 1. Id: 568, Name: Where are you baby 1 95, Albu 2. Id: 569, Name: Alicia is nice 2 95, AlbumId: 19 3. Id: 570, Name: I'm crying 3 95, AlbumId: 190

Details Artist button -> you have to select some record and it shows the details of that artist in a new window.

UpdateCreateArtist

Return



You are in the Create Artist section

NAME

DESCRIPTION

KIND OF MUSIC

CONTACT EMAIL

CONTACT PHONE

COUNTRY

CITY

Save

Clear

Add -> button adds a new artist.

✕

Status Code: 400 -> BadRequest
Errors: "errors":{"City":["The City field is required."],"Name":["The Name field is required."],"Country":["The Country field is required."],"Description":["The Description field is required."],"ContactEmail":["The ContactEmail field is not a valid e-mail address."],"ContactNumber":["The ContactNumber field is not a valid phone number."]}}

OK

Error information:

Name -> cannot be empty. **City** -

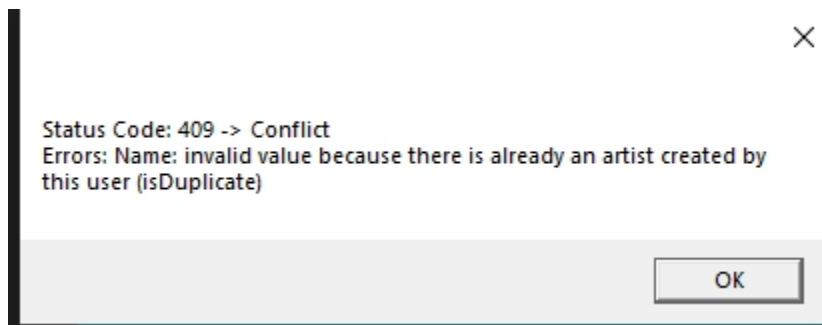
> cannot be empty. **Country** ->

cannot be empty.

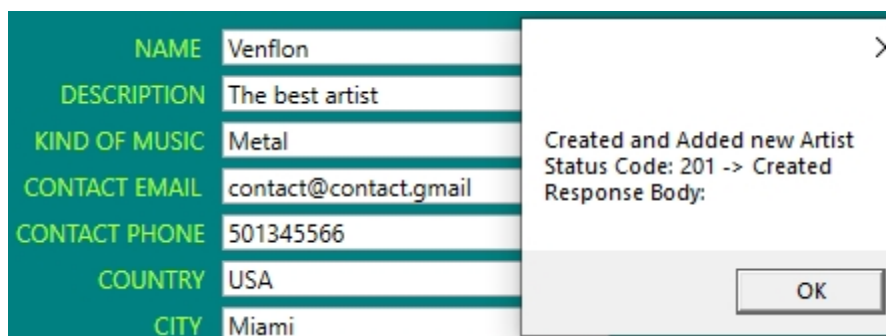
Description -> cannot be empty.

ContactNumber -> is incorrect, there should be numbers.

ContactEmail -> there is an invalid address.

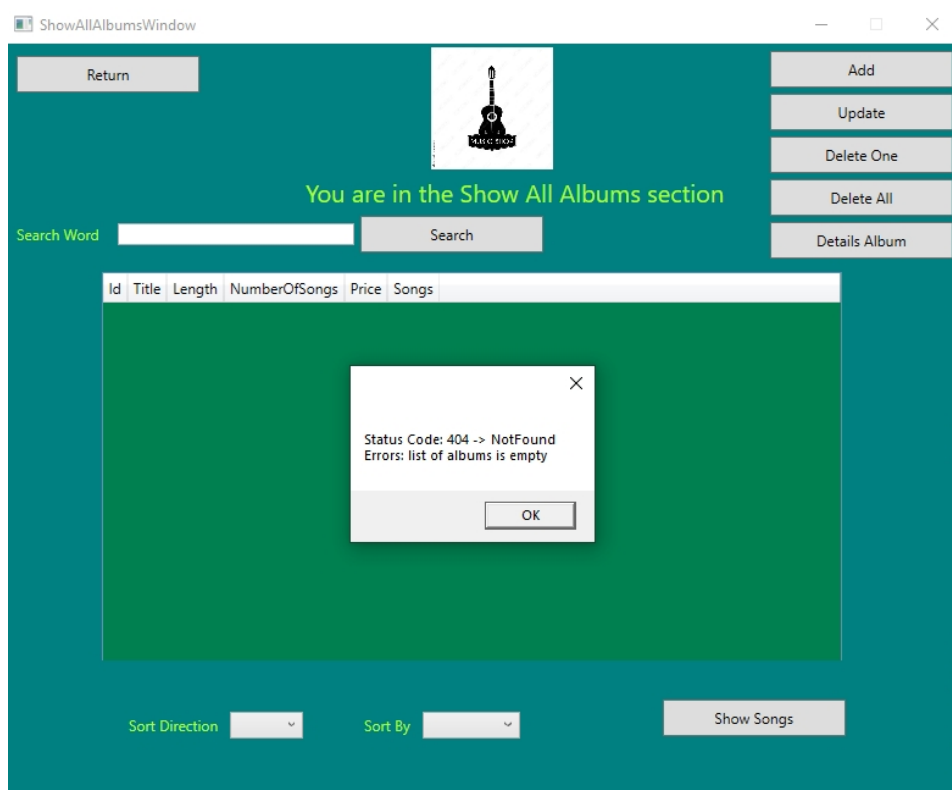


When creating an album name that is already in the list of a given user, it creates a duplicate name, that is, an error is displayed.

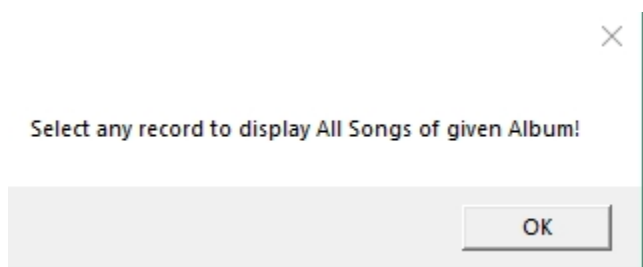


All Artists						
Id	Name	Description	KindOfMusic	Country	City	AlbumsAndSongs
151	Venflon	The best artist	Metal	USA	Miami	

Created correctly, then goes to the **All Artists** window



We enter a new window thanks to the **Show Songs** button and we get the information that the list is missing Albums.



An error message if you enter a song list that is not yet in a non-existent album.

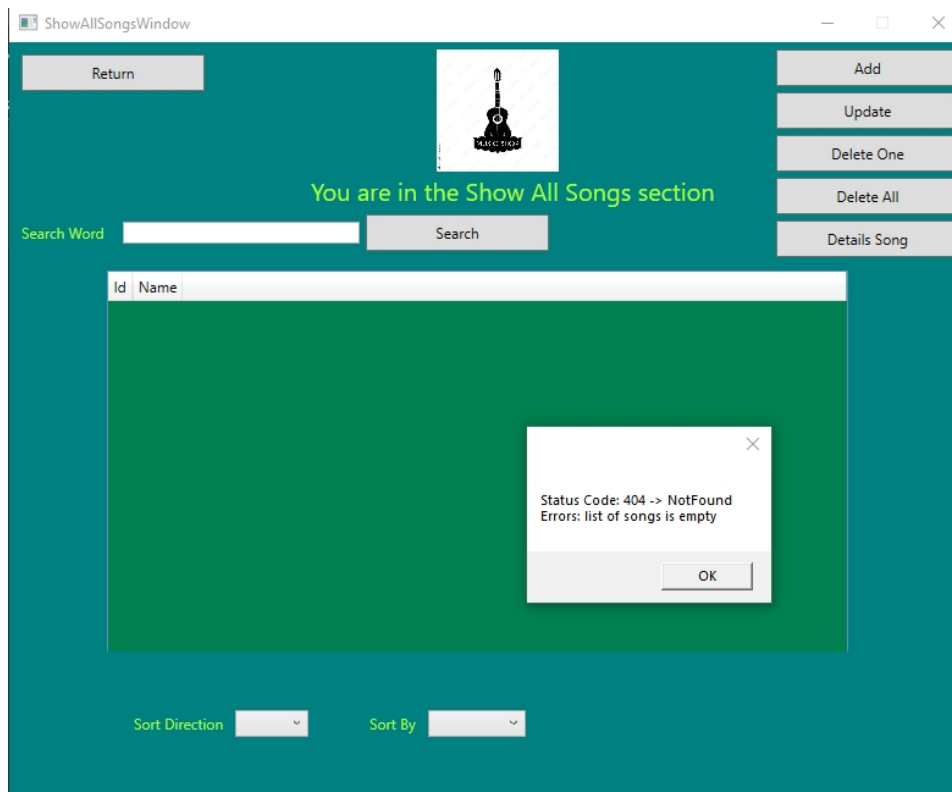
I create a new album with the **Add** button.

Error list -> **Price** : must be a digit: 33 or 15.5, **Length** -> also must be a digit.

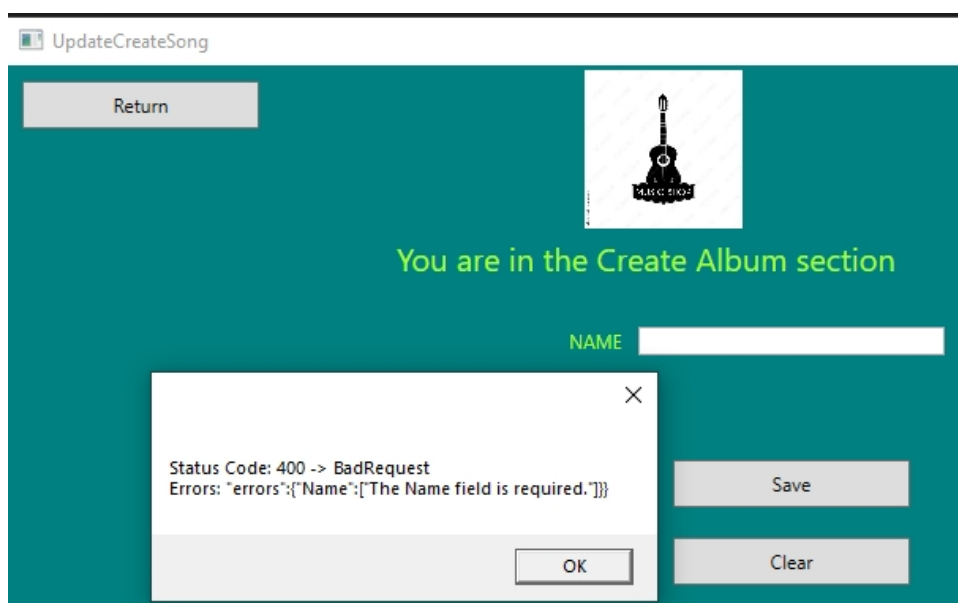
Creating a new song. Then it automatically returns to the **All Albums** window.

Id	Title	Length	NumberOfSongs	Price	Songs
301	Armagedon	3.5	0	15.8	
302	Second album	22	0	44	
303	second	213	0	22	

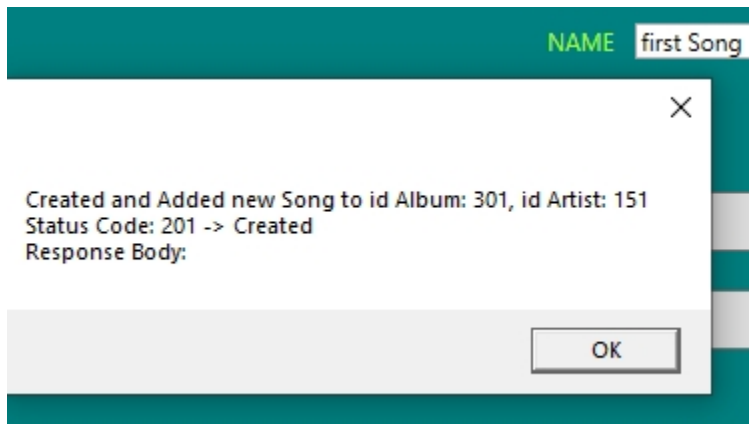
In the **All Albums** window, you can sort **ASC** ub **DESC** using **Sort By** -> **Title** values. Button **Search** -> searches by **Title** name.



Show Songs -> button goes to the song window of the album and information that there is no list songs.

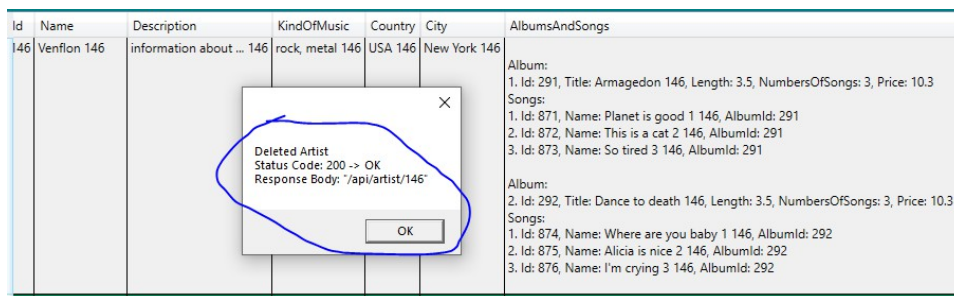


When creating a new song, the error message that **Name** -> cannot be empty.

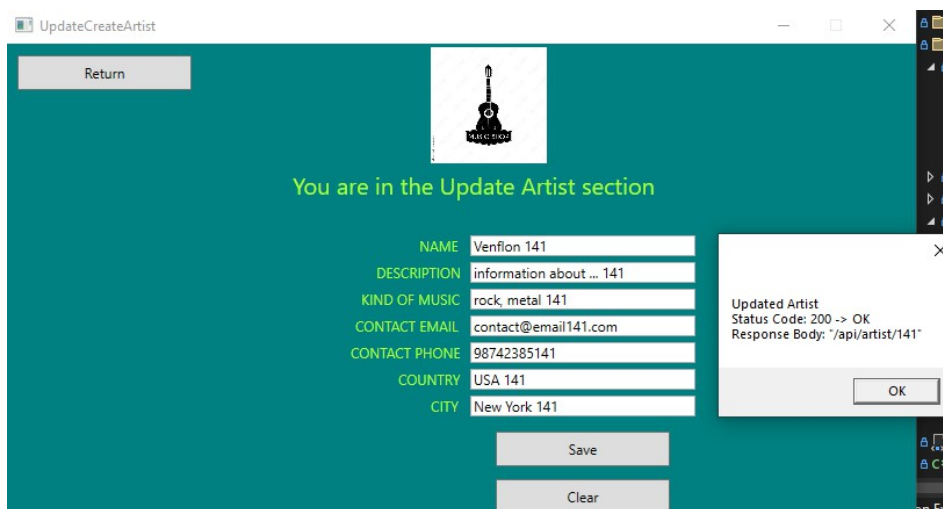


Id	Name	
901	first Song	
902	second song	
903	third song	

Creating a new song and returns to the **All Songs** window. Sorting works the same as in the **All Albums** window.



Action of the **Delete** button in the **Show All Artists** window, information about the deletion and the given path where the artist was previously located.



In the **All Artist** window, the Update-> button updates and downloads the current one of the artist in question and returns to the **All Albums** window when everything has gone through without problems.

You are in the Update Artist section

NAME	Iron Maiden 2
DESCRIPTION	information about ... 1
KIND OF MUSIC	rock, metal 1
CONTACT EMAIL	contact@email1.com
CONTACT PHONE	987423851
COUNTRY	USA 1
CITY	New York 1

Status Code: 409 -> Conflict
 Errors: Name: invalid value because there is already an artist created by this user (isDuplicate)

OK

If there is an attempt to duplicate an album name that already exists in the list of a particular user, it will appear notification.

DetailsArtist

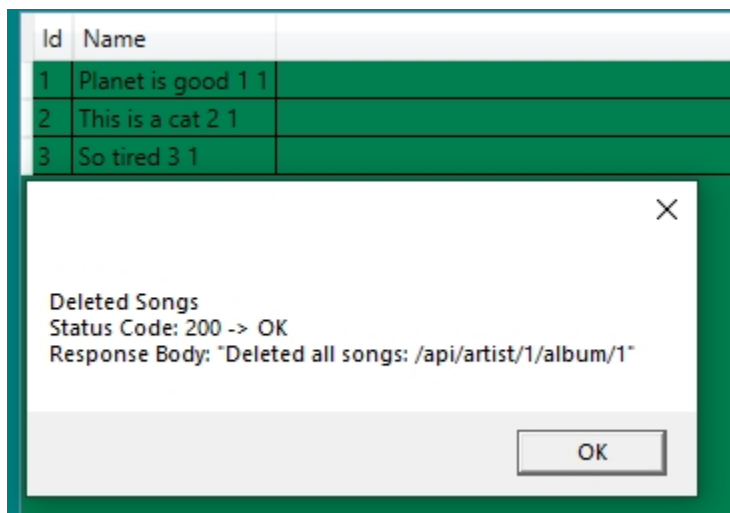
Return

You are in the Details of Artist section

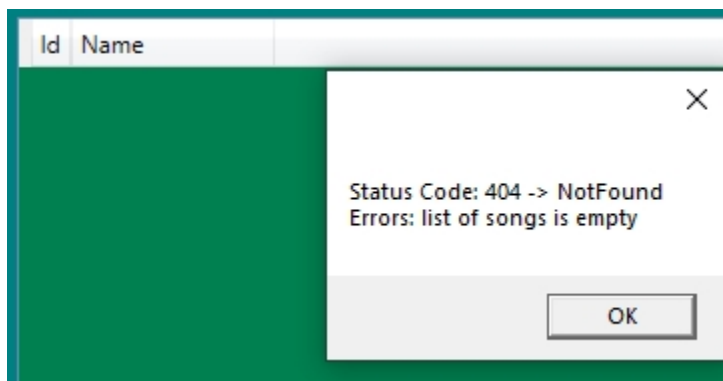
Id	Name	Description	KindOfMusic	Country	City	ContactEmail	ContactNumber	AlbumsSongs
1	Iron Maiden 1	information about ... 1	rock, metal 1	USA 1	New York 1	contact@email1.com	987423851	<p>Album:</p> <p>1. Id: 1, Title: Armagedon 1, Length: 3.5, NumbersO</p> <p>Songs:</p> <p>1. Id: 1, Name: Planet is good 1 1, AlbumId: 1</p> <p>2. Id: 2, Name: This is a cat 2 1, AlbumId: 1</p> <p>3. Id: 3, Name: So tired 3 1, AlbumId: 1</p> <p>Album:</p> <p>2. Id: 2, Title: Dance to death 1, Length: 3.5, Numb</p> <p>Songs:</p> <p>1. Id: 4, Name: Where are you baby 1 1, AlbumId: 2</p> <p>2. Id: 5, Name: Alicia is nice 2 1, AlbumId: 2</p> <p>3. Id: 6, Name: I'm crying 3 1, AlbumId: 2</p>

In the All Artist window, the **Details Artist** -> button opens a new **Details of Artist** window. **ContactEmail** and **ContactNumber** -> columns are displayed only for the user who created the album, for others they are restricted.

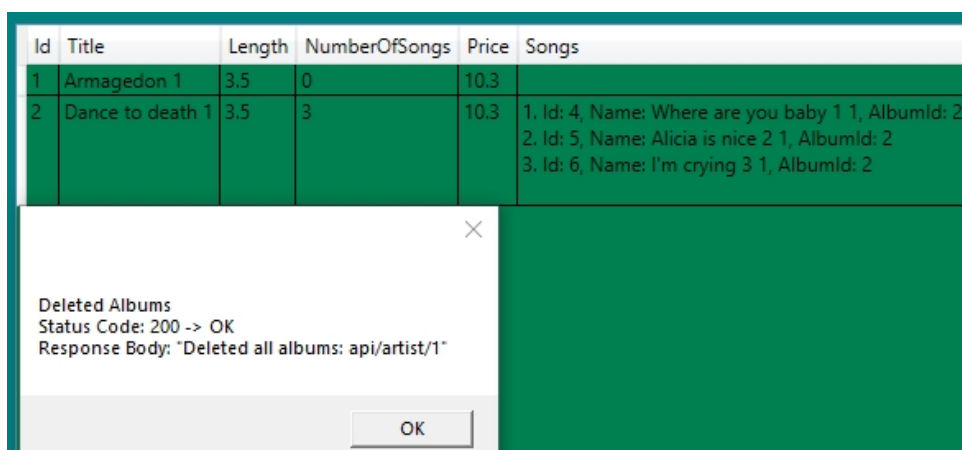
In the **All Albums** and **All Songs** windows, everything works similarly, only additionally there is a **Delete All** -> button, which deletes all given albums or songs, then it appears that the list of albums, songs is empty.



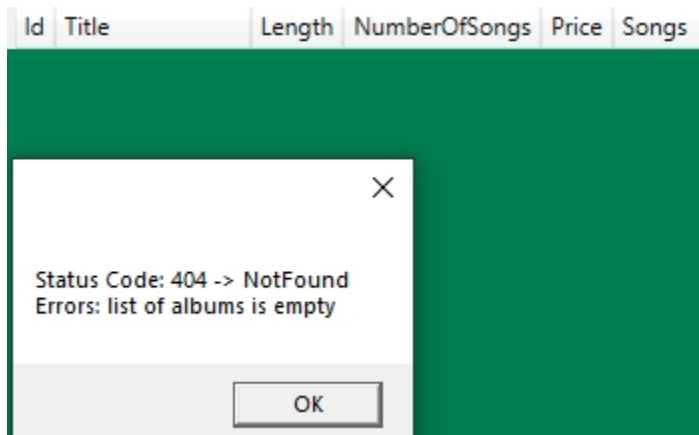
Before using the **Delete All** button in the **All Songs** window.



After use.



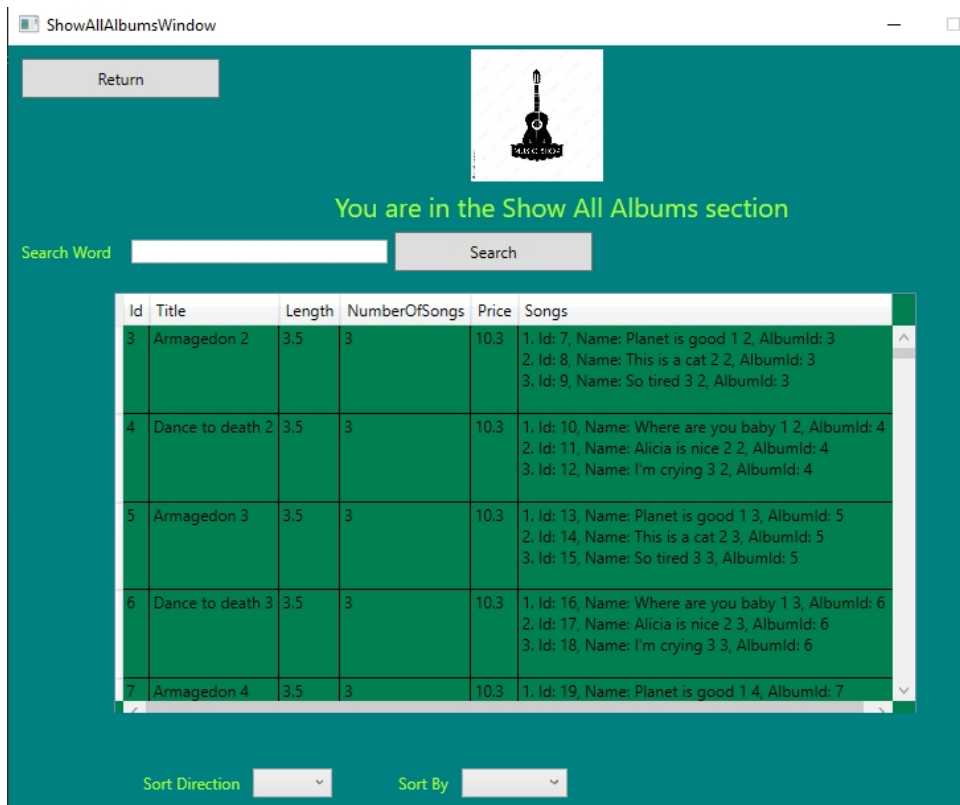
Before using the **Delete All** button in the **All Albums** window.



After use.

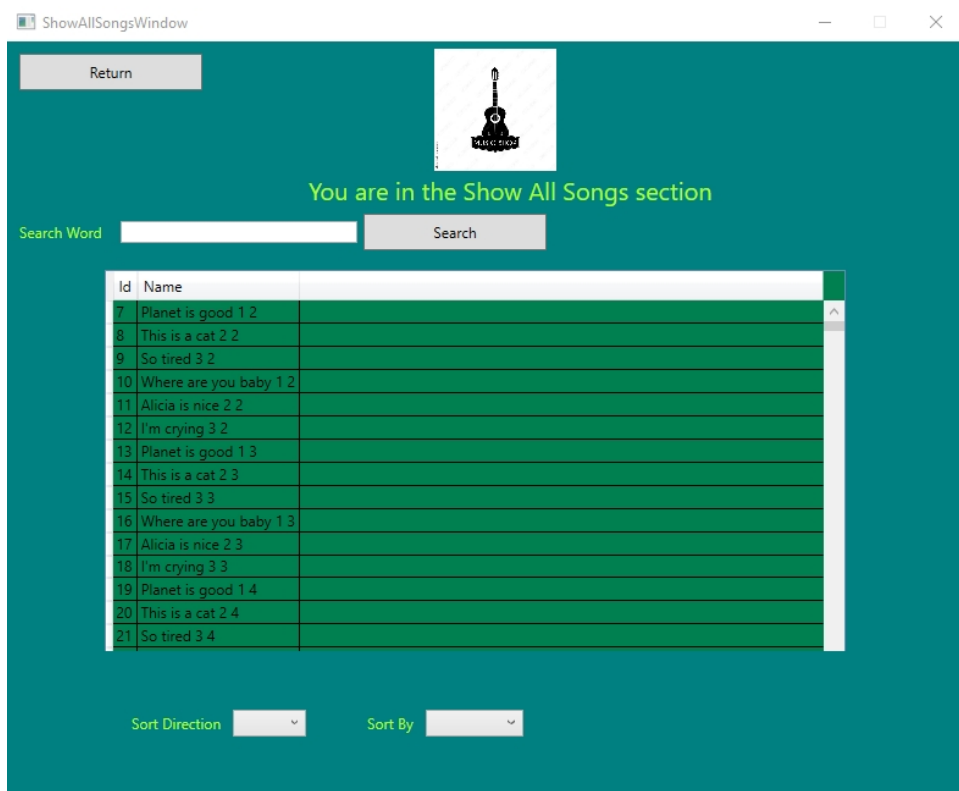


There are two buttons in the **Music Shop** window:

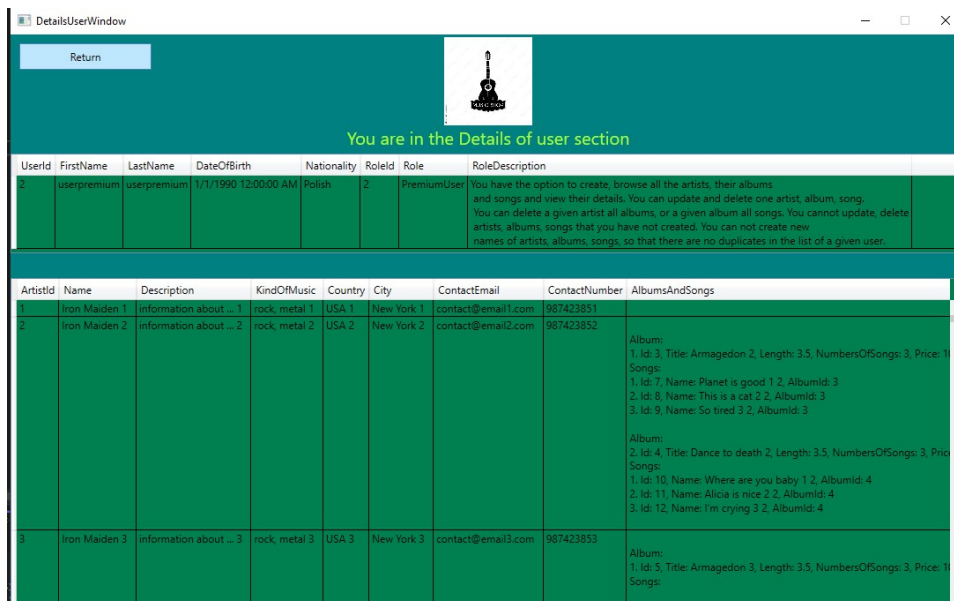


The **Albums** -> button enters a new window that shows all the albums in the database.

Search -> button searches by **Title** name, **Sort Direction** -> **ASC, DESC** sorts. **Sort By** -> sorts After **Title** values.



The Songs-> button enters a new window that shows all the songs in the database and you can sort by **Name** column.



There is also an additional **User** Details-> button in the Music Shop window, which opens a new window

Details of User and shows the details of a particular user, that is, in the first table there is information

detailed of a given user , and on the other are all the Artists, their albums, their songs created by the this user.

Additionally created / changed things in the Backend Application :

1. Adding a new endpoint -> **GET api/login/user/{email}** -> to access the user's details and the generated **JWT Token**, after logging into the main Music Store window.
2. Adding a new endpoint -> **GET api/login/user/{userId}/artist** -> to have access after logging into the main Music Store window to all the artists created by a given user and detailed information to display.
3. Adding a new endpoint -> **GET api/song** -> retrieves all songs from the database.
4. Adding a new endpoint -> **GET api/album** -> retrieves all albums from the database.
5. Changing the sorting of data, if you do not select **ASC (1)**-> ascending or **DESC (2)** descending and **SearchWord**, it does not sort.
6. Add a new endpoint -> **GET api/artist/{artistId}/album/{albumId}/song/details/{songId}** -> to access the details of a particular song with additional fields: **AlbumTitle, ArtistId, ArtistName**.
7. Add a new endpoint -> **GET api/artist/{artistId}/album/details/{albumId}** -> to access the details of a given album with additional fields: **ArtistName, ArtistId**.
8. Add a new endpoint -> **GET api/artist/details/{Id}** -> to access the details of a particular artist with additional fields: **ContactEmail, ContactNumber**.