**TABLE OF PROBLEM SPECIFICATION**

|  |  |
| --- | --- |
| **CLIENT** | NeoTunes |
| **USER** | * Audio consumers * Audio producers * NeoTunes Administrators |
| **FUNCTIONAL REQUIREMENTS** | * R1: Register producer * R2: Register buyer * R3: Register audio   R3.1: Register song  R3.2: Register podcast   * R4: Create playlist * R5: Edit playlist   R5.1: Rename playlist  R5.2: Add audio to playlist  R5.3: Remove audio the playlist   * R6: Share playlist * R7: Simulate playback of a song or podcast * R8: Buy a song * R9: Generate reports   R9.1: The total accumulated reproductions on the entire platform  R9.2: The most listened song genre per user and platform  R9.3: The most listened podcast category per user and platform  R9.4: The top 5 producers  R9.5: The top 10 audios  R9.6: The song total sales per genre  R9.7: The best-selling song on the platform |
| **CONTEXT OF THE PROBLEM** | NeoTunes is a company that seeks to develop a software prototype that will enable it to compete in the streaming music and audio content market. The company's business model has two focuses, user subscriptions and the sale of songs through the platform. Unlike other competitors in the market, on NeoTunes users can own their music catalog. This company intends to make it possible to register various types of users and audios. It also wants the software to interact with them, and finally, it wants the company's administrators to be able to view different types of statistics to see how the company is performing. |
| **NON-FUNCTIONAL REQUIREMENTS** | * The platform must be adaptable to new types of users of each type * The platform must be adaptable to add multiple types of audios * The playlist identifiers must be generated automatically by a matrix algorithm, formed by 16 digits * The advertisement must include the brands: Nike, Coca-Cola and M&Ms |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R1)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R1: Register producer | | |
| **ABSTRACT** | The software must allow the user to create a new producer, whether artist or content creator, and add it to the platform | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| optProducer | int | Fields filled correctly |
| nameUser | String |
| identificationNumber |
| bondingDate |
| photoURL |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the user's producer type 2. Read the user's name 3. Read the user's ID number 4. Read the user's bonding date 5. Read the URL of the user's photo 6. Verify that the user name is not in use 7. If the producer type is Artist it creates a new producer of type Artist, otherwise it creates a producer of type Content Creator 8. Add the new Producer to the platform. | | |
| **RESULT OR POSTCONDITION** | The new Producer is registered in the user list | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R2)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R2: Register buyer | | |
| **ABSTRACT** | The software must allow the user to create a new buyer, whether standard or premium, and add it to the platform | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| optBuyer | int | Fields filled correctly |
| nameUser | String |
| identificationNumber |
| bondingDate |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the user's buyer type 2. Read the user's name 3. Read the user's ID number 4. Read the user's bonding date 5. Verify that the user name is not in use 6. If the buyer type is Standard it creates a new buyer of type Standard, otherwise it creates a buyer of type Premium 7. Add the new Buyer to the platform. | | |
| **RESULT OR POSTCONDITION** | The new Buyer is registered in the user list | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R3)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R3.1: Register song | | |
| **ABSTRACT** | The software must allow the artist's user to create a new song and add it to the platform's audio list and the artist's song list | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| nameArtist | String | Fields filled correctly |
| nameSong |
| album |
| genreType | int |
| url | String |
| duration |
| saleValue | double |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the artist's name 2. Read the song's name 3. Read the song's album 4. Read the song's genre type 5. Read the album cover of the song 6. Read the song's duration 7. Read the song's sale value 8. Verify that the user exists 9. Verify that this user is an artist type 10. Verify that the song is not registered 11. Create the new song 12. Add the song to audio list of the user 13. Add the song to audio list of the platform | | |
| **RESULT OR POSTCONDITION** | The new Song is registered in the audio lists | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R3.2: Register podcast | | |
| **ABSTRACT** | The software must allow the content creator user to create a new podcast and add it to the platform audio list and to the content creator's podcast list | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| nameContentCreator | String | Fields filled correctly |
| namePodcast |
| description |
| podcastCategory | int |
| url | String |
| duration |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the content creator's name 2. Read the podcast's name 3. Read the podcast's description 4. Read the podcast's category 5. Read the image of the podcast 6. Read the podcast's duration 7. Verify that the user exists 8. Verify that this user is a content creator type 9. Verify that the podcast is not registered 10. Create the new podcast 11. Add the podcast to audio list of the user 12. Add the podcast to audio list of the platform | | |
| **RESULT OR POSTCONDITION** | The new Podcast is registered in the audio lists | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R4)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R4: Create playlist | | |
| **ABSTRACT** | The software must allow the buyer user to create a playlist and add it to the platform's playlist and the user's playlist. | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| nameUser | String | Fields filled correctly |
| namePlaylist |
| playlistType | int |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the user's name 2. Read the playlist's name 3. Read the playlist's type 4. Generate and show the matrix 6x6 5. Depending on the type of playlist, the playlist identifier code will be generated and displayed. 6. Verify that the user exists 7. Verify that this user is a buyer type 8. Verify that the playlist is not registered and consider the limitation of the standard user 9. Create the new playlist 10. Add the playlist to playlists of the user 11. Add the playlist to playlist of the platform | | |
| **RESULT OR POSTCONDITION** | The new Playlist is registered in the playlists | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R5)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R5.1: Rename playlist | | |
| **ABSTRACT** | The software must allow to edit the playlist of a buying user, in this case, it must allow to change the name of the playlist | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| nameUser | String | Fields filled correctly |
| namePlaylist |
| newNamePlaylist |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the user's name 2. Read the playlist's name 3. Read the playlist's new name 4. Verify that the user exists 5. Verify that this user is a buyer type 6. Verify that the playlist is registered for this user 7. Change playlist name | | |
| **RESULT OR POSTCONDITION** | The name of a user's playlist is changed | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R5.2: Add audio to playlist | | |
| **ABSTRACT** | The software must allow to edit the playlist of a buying user, in this case, it must allow to add an audio to playlist | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| nameUser | String | Fields filled correctly |
| namePlaylist |
| nameAudioToAdd |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the user's name 2. Read the playlist's name 3. Read the audio name to be added 4. Verify that the user exists 5. Verify that the audio exists 6. Verify that this user is a buyer type 7. Verify that the playlist is registered for this user 8. Verify that the audio can be added. (Consider the limitation of the standard user and type of playlist) 9. Add audio to playlist | | |
| **RESULT OR POSTCONDITION** | The audio is added to the playlist of a buyer user | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R5.3: Remove audio the playlist | | |
| **ABSTRACT** | The software must allow to edit the playlist of a buying user, in this case, it must allow to remove an audio to playlist | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| nameUser | String | Fields filled correctly |
| namePlaylist |
| nameAudioToRemove |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the user's name 2. Read the playlist's name 3. Read the audio name to be removed 4. Verify that the user exists 5. Verify that this user is a buyer type 6. Verify that the playlist is registered for this user 7. Verify that audio is added to the playlist 8. Remove audio to playlist | | |
| **RESULT OR POSTCONDITION** | The audio is removed from the playlist of a buyer user | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R6)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R6: Share playlist | | |
| **ABSTRACT** | The software must allow sharing a user's playlist, displaying the matrix and the identifier code | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| nameUser | String | Fields filled correctly |
| namePlaylist |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the user's name 2. Read the playlist's name 3. Verify that the user exists 4. Verify that this user is a buyer type 5. Verify that the playlist is registered for this user 6. Show playlist’s matrix 7. Show playlist’s code identifier | | |
| **RESULT OR POSTCONDITION** | The matrix and the user's playlist identifier code are displayed | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R7)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R7: Simulate playback of a song or podcast | | |
| **ABSTRACT** | The software must be able to simulate the playback of a song or podcast. In the case of songs, provided that the user has purchased it | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| nameUser | String | Fields filled correctly |
| nameAudio |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the user's name 2. Read the audio’s name 3. Verify that the user exists 4. Verify that the audio exists 5. Verify that this user is a buyer type 6. If a song is to be played, check that it has been purchased by the user. (Otherwise, this step is not required) 7. Play audio (In the case of the standard user, display advertisements) 8. Update producer user and audio variables | | |
| **RESULT OR POSTCONDITION** | Audio is played back and playback related variables are updated | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R8)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R8: Buy a song | | |
| **ABSTRACT** | The software must allow to buy a song from a buyer user | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| nameUser | String | Fields filled correctly |
| nameSong |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the user's name 2. Read the song’s name 3. Verify that the user exists 4. Verify that the song exists 5. Verify that this user is a buyer type 6. Verify that the user has not purchased the song 7. Adds the song to the user's list of purchased songs 8. Update song variables | | |
| **RESULT OR POSTCONDITION** | Audio is buyed and variables are updated | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R9)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R9.1: The total accumulated reproductions on the entire platform | | |
| **ABSTRACT** | The software must be able to generate a report of the total accumulated plays for the entire platform for each type of audio | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| optAudio | int | Fields filled correctly |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the type of audio for which you wish to obtain the accumulated number of plays 2. Get the global audio list 3. Get the number of plays of the selected type 4. To add up the reproductions 5. Show the accumulated | | |
| **RESULT OR POSTCONDITION** | Report showing the total number of plays for each audio type | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R9.2: The most listened song genre per user and platform | | |
| **ABSTRACT** | The software must be able to generate a report showing the most listened song genre by user and in the whole platform. | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| optReport | int | Fields filled correctly |
| nameUser | String | If it is for a specific user |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | For the entire platform:   1. Get the global audio list 2. Get the number of reproductions of each song 3. Separate the results by genre 4. Compare to select the most listened genre 5. Show result   For one user:   1. Read the user's name 2. Get the list of audios played by the user 3. Get the number of reproductions for each song 4. Separate the results by genre 5. Compare to select the most listened genre by user. 6. Show result. | | |
| **RESULT OR POSTCONDITION** | Generate a report showing the most listened genre by user and in the whole platform | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R9.3: The most listened podcast category per user and platform | | |
| **ABSTRACT** | The software must be able to generate a report showing the most listened podcast category per user and platform. | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| optReport | int | Fields filled correctly |
| nameUser | String | If it is for a specific user |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | For the entire platform:   1. Get the global audio list 2. Get the number of reproductions of each podcast 3. Separate the results by category 4. Compare to select the most listened category 5. Show result   For one user:   1. Read the user's name 2. Get the list of audios played by the user 3. Get the number of reproductions for each podcast 4. Separate the results by category 5. Compare to select the most listened category by user 6. Show result | | |
| **RESULT OR POSTCONDITION** | Generate a report showing the most listened category by user and in the whole platform. | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R9.4: The top 5 producers | | |
| **ABSTRACT** | The software must be able to display the top 5 producers, that is, the top 5 artists and the top 5 content creators. It must show the name and number of total plays. | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| optProducer | int | Fields filled correctly |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | To obtain the top 5 artists:   1. Go through each artist getting the number of reproductions of each song 2. Compare each artist number of reproductions to order from the highest to the lowest 3. Select the five artists with more reproductions 4. Show them in order from higher to lower as a top five   To obtain the top 5 content creators:   1. Go through each content creator getting the number of reproductions of each podcast 2. Compare each content creator number of reproductions to order from the highest to the lowest 3. Select the five content creators with more reproductions 4. Show them in order from higher to lower as a top five | | |
| **RESULT OR POSTCONDITION** | Generate a report showing the top five producers | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R9.5: The top 10 audios | | |
| **ABSTRACT** | The software must be able to show the top ten audios more listened, that is, the top 10 songs and the top 10 podcasts It must show the name, genre or category and total number of plays | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| optAudio | int | Fields filled correctly |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | To obtain the top 10 songs:   1. Go through each song getting the number of reproductions 2. Compare each song number of reproductions to order from the highest to the lowest 3. Select the ten songs with more reproductions 4. Show them in order from higher to lower as a top ten   To obtain the top 10 podcasts:   1. Go through each podcast getting the number of reproductions 2. Compare each podcast number of reproductions to order from the highest to the lowest 3. Select the ten podcasts with more reproductions 4. Show them in order from higher to lower as a top ten | | |
| **RESULT OR POSTCONDITION** | Generate a report showing the top ten more listened audios. | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R9.6: The song total sales per genre | | |
| **ABSTRACT** | The software must be able to generate a report showing the number of songs sold and the total sales value for each genre. | | |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Go through each producer getting the number of sales of each song 2. Multiply the amount of sales by the price 3. Separate the results by genre 4. Sum the result in each genre 5. Show the result for each genre | | |
| **RESULT OR POSTCONDITION** | Generate a report showing the total sales by genre | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R9.7: The best-selling song on the platform | | |
| **ABSTRACT** | The software must be able to generate a report of the best-selling song on the platform, showing the total number of sales and total sales value | | |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Go through each artist getting the number of sales of each song 2. Order from highest to lower 3. Show the song with the highest result | | |
| **RESULT OR POSTCONDITION** | Generate a report showing the most sold song | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**TRACEABILITY BETWEEN DESIGN AND ANALYSIS**

|  |  |  |
| --- | --- | --- |
| **FUNCTIONAL REQUIREMENTS** | **Name of the class** | **Name of the method** |
| **R1:**  **Register producer.** | Class NeoTunesManager | + showMainMenu(): void  + registerProducerUser(): void |
| Class NeoTunesController | + searchUser(nameUser : String): User  + registerProducerUser(optProducer : int, nameUser : String, identificationNumber : String, year : int, month : int, day : int, photoURL : String): String |
| Class Artist | + Artist(nameUser : String, identificationNumber : String, bondingDate : Date, photoURL : String) |
| Class ContentCreator | + ContentCreator(nameUser : String, identificationNumber : String, bondingDate : Date, photoURL : String) |
| **R2:**  **Register buyer.** | Class NeoTunesManager | + showMainMenu(): void  + registerBuyerUser: String |
| Class NeoTunesController | + searchUser(nameUser : String): User  + registerBuyerUser(optBuyer : int, nameUser : String, identificationNumber : String, year : int, month : int, day : int): void |
| Class Standard | + Standard(nameUser : String, identificationNumber : String, bondingDate : Date) |
| Class Premium | + Premium(nameUser : String, identificationNumber : String, bondingDate : Date) |
| **R3.1:**  **Register song.** | Class NeoTunesManager | + showMainMenu(): void  + registerSong(): void |
| Class NeoTunesController | + getGenreType(): String  + convertToSec(min : int, sec : int): int  + searchUser(nameUser : String): User  + searchAudio(nameAudio : String): Audio  + registerSong(nameSong : String, album : String, genreType : int, url : String, duration : int, saleValue : double, nameArtist : String): String |
| Class Song | + Song(nameAudio : String, album : String, genreType : int, url : String, duration : int, saleValue : double) |
| Class Artist | + addAudio(newAudio : Audio): void |
| **R3.2:**  **Register podcast.** | Class NeoTunesManager | + showMainMenu: void  + registerPodcast(): void |
| Class NeoTunesController | + getPodcastCategory(): String  + convertToSec(hour : int, min : int, sec : int): int  + searchUser(nameUser : String): User  + searchAudio(nameAudio : String): Audio  + registerPodcast(namePodcast : String, description : String, podcastCategory : int, url : String, duration : int, nameContentCreator : String): String |
| Class Podcast | + Podcast(nameAudio : String, description : String, podcastCategory : int, url : String, duration : int) |
| Class ContentCreator | + addAudio(newAudio : Audio): void |
| **R4:**  **Create playlist.** | Class NeoTunesManager | + showMainMenu(): void  + registerPodcast(): void |
| Class NeoTunesController | + getPlaylistType(): String  + generateMatrix() : int[][]  + generateCodeN(matrix : int[][]) : String  + generateCodeT(matrix : int[][]) : String  + generateCodeStaggered(matrix : int[][]) : String  + searchUser(nameUser : String): User  + createPlaylist(nameUser : String, namePlaylist : String, playlistType : int, matrix : int[][], idPlaylist : String): String |
| Class Buyer | + searchPlaylist(namePlaylist : String): Playlist  + addPlaylist(newPlaylist : Playlist) : boolean |
| Class Playlist | + Playlist(namePlaylist : String, playlistType : int matrix : int[][], idPlaylist : String) |
| **R5.1:**  **Rename playlist.** | Class NeoTunesManager | + showMainMenu(): void  + editPlaylist(): void |
| Class NeoTunesController | + searchUser(nameUser : String): User  + renamePlaylist(nameUser : String, namePlaylist : String, newNamePlaylist : String) : String |
| Class Buyer | + searchPlaylist(namePlaylist : String): Playlist |
| Class Playlist | + setNamePlaylist(namePlaylist : String): void |
| **R5.2:**  **Add audio to playlist.** | Class NeoTunesManager | + showMainMenu(): void  + editPlaylist(): void |
| Class NeoTunesController | + searchUser(nameUser : String): User  + searchAudio(nameAudio : String): Audio  + addAudioToPlaylist(nameUser : String, namePlaylist : String, nameAudioToAdd: String) : String |
| Class Buyer | + searchPlaylist(namePlaylist : String): Playlist |
| Class Playlist | + searchAudio(nameAudio : String) : Audio  + addAudio(newAudio : Audio): boolean |
| **R5.3:**  **Remove audio the playlist.** | Class NeoTunesManager | + showMainMenu(): void  + editPlaylist(): void |
| Class NeoTunesController | + searchUser(nameUser : String): User  + removeAudioOfPlaylist(nameUser : String, namePlaylist : String, nameAudioToRemove : String) : String |
| Class Buyer | + searchPlaylist(namePlaylist : String): Playlist |
| Class Playlist | + searchAudio(nameAudio : String) : Audio  + addAudio (newAudio : Audio): boolean |
| **R6:**  **Share playlist.** | Class NeoTunesManager | + showMainMenu(): void  + sharePlaylist(): void |
| Class NeoTunesController | + searchUser(nameUser : String): User  + printMatrix(matrix : int[][]) : String  + sharePlaylist(nameUser : String, namePlaylist : String) : String |
| Class Buyer | + searchPlaylist(namePlaylist : String): Playlist |
| Class Playlist | + getMatrix(): String  + getIdPlayList(): String |
| **R7:**  **Simulate playback of a song or podcast.** | Class NeoTunesManager | + showMainMenu(): void  + playAudio(): void |
| Class NeoTunesController | + searchUser(nameUser : String): User  + searchAudio(nameAudio : String): Audio  + updateState(audio : Audio): Void  + playAudio(nameUser : String, nameAudio : String): String |
| Class Buyer | + play(audio : Audio): String |
| **R8:**  **Buy a song.** | Class NeoTunesManager | + showMainMenu(): void  + buySong(): void |
| Class NeoTunesController | + searchUser(nameUser : String): User  + searchAudio(nameAudio : String): Audio  + buySong(nameUser : String, nameAudio : String): String |
| Class Buyer | + searchSong(song : Song): boolean  + addSong(newSong : Song): boolean |
| **R9.1:**  **The total accumulated reproductions on the entire platform.** | Class NeoTunesManager | + showMainMenu(): void  + showTotalAccumulatedReproductions (): void |
| Class NeoTunesController | + showTotalAccumulatedReproductions (optAudio : int): void |
| **R9.2:**  **The most listened song genre per user and platform** | Class NeoTunesManager | + showMainMenu(): void  + showMostPlayedSongGenre (): Void |
| Class NeoTunesController | + showMostPlayedSongGenre(): String  + searchUser(nameUser : String): User  + showUserMostPlayedSongGenre(nameUser : String): String |
| **R9.3:**  **The most listened podcast category per user and platform** | Class NeoTunesManager | + showMainMenu(): void  + showMostPlayedPodcastCategory(): void |
| Class NeoTunesController | + showMostPlayedPodcastCategory(): String  + searchUser(nameUser : String): User  + showUserMostPlayedPodcastCategory(nameUser : String): String |
| **R9.4:**  **The top 5 producers** | Class NeoTunesManager | + showMainMenu(): void  + showTopProducers(): void |
| Class NeoTunesController | + bubbleSort(array : int[]): void  + isRepeated(array : int[], aux : int): Boolean  + showTopArtist(): String  + showTopContentCreator(): String |
| **R9.5:**  **The top 10 audios** | Class NeoTunesManager | + showMainMenu(): void  + showTopAudio(): void |
| Class NeoTunesController | + bubbleSort(array : int[]): void  + isRepeated(array : int[], aux : int): Boolean  + showTopSong(): String  + showTopPodcast(): String |
| **R9.6:**  **The song total sales per genre** | Class NeoTunesManager | + showMainMenu(): void  + showInfoByGenreSong(): void |
| Class NeoTunesController | + showInfoByGenreSong(genreType : int): String |
| **R9.7:**  **The best-selling song on the platform** | Class NeoTunesManager | + showMainMenu(): void  + showBestSellerSong(): void |
| Class NeoTunesController | + showBestSellerSong(): String |