|  |  |
| --- | --- |
| Client | NeoTunes |
| User | * The programmers and designers that will keep working on the project |
| Functional requirements | 1. Register user producer: artists and content creators. 2. Register consumer, standard and premium users. 3. Record songs and podcasts. 4. Create a playlist. 5. Edit a playlist. 6. Share a Playlist 7. Simulate playing a song or podcast (standard and premium) 8. Buy an audio 9. Inform data |
| Problem context | A Danish company wants to create a prototype of software that will allow them to enter the music streaming and audio content industry, this prototype has a special characteristic that let the users can truly become owners of the music they listen to. |
| Non-functional requirements | * The program must use the language Java * The program must be fast, cannot longer more than 2 seconds to start * The program must work in android and web platforms * The program must be scalable |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | **R1: Register artists and creators of content.** | | |
| Abstract | The program must let the user create a producer user. This user can be an artist or a creator of content. From this user the system receives: Name (unique identifier), linking date, and an URL with the artist’s image or distinctive logo. The system must now the reproductions of this artist and the total time the users spent in this artist. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| name | String | * If the name already exists |
| linkingDate | String |  |
| URL | String |  |
| isCreatorOfContent | boolean |  |
| General activities needed to obtain the results | 1. The system will receive a nickname and search if it already exists. 2. The system classifies the producer according to the isCreatorOfContent parameter 3. The system will add a producer user to the system 4. The system returns a String confirming if the operation was successful or if it failed. | | |
| Result or post condition | The program registers a producer and returns a String validating the operation | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| result | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | **R2: Register consumer users, standard and premium.** | | |
| Abstract | The system must let the user create a consumer user. This kind of user will be the ones that are going to buy and listen to the audio / music of the producer user. This user will be divided in two types:   * Standard: Will be able to create only 20 playlists, buy 100 songs and podcasts, play an ad every 2 songs or before a podcast. * Premium: This user will be able to create infinite playlists and songs, also this user will be able to access to both standard and premium content without ads.   From this user the system receives: nickname, id, linking date | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| nickname | String | * This parameter cannot be repeated |
| id | String |
| isPremium | boolean |  |
| General activities needed to obtain the results | 1. The program receives the nickname and id and checks if it already exists 2. The program will take the date on which the registration is made. 3. The program classifies the user whether is premium or standard and assigns his limits. 4. The system returns a String confirming if the operation was successful or if it failed. | | |
| Result or post condition | A new level is created and returns a String confirming the success of the operation | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| result | String |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | **R3: Record songs and podcast.** | | |
| Abstract | The artists can create songs, and the creators of content can create podcasts. Either audios, music or podcasts will have a name, an URL related to the audio file album cover or to the topic in the case of the podcast, and finally a duration. The type of audio file depends on what the producer wants to add  The song will also have these features:   * Price (dollars), the number of times it was sold and the genres available to it will be Rock, Pop, Trap and House.   On the other hand, podcast:   * Cannot be bought and the possible categories will be: Política, Entretenimiento, Videojuegos y Moda.   The system must have a good scalability, so adding a new audio type must be easy. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| name | String |  |
| URL | String |
| duration | double |
| price | double | The type selected is a song |
| timesSold | int |
| genre | TypeGenre |
| category | TypeCategory | The type selected is a podcast |
| General activities needed to obtain the results | 1. The program will check if it is a song or a podcast. 2. The program will receive a name, URL, duration, price, timesSold and genre in case of the song 3. The program will receive a name, URL, duration, and category in case of the podcast 4. The program returns a String validating the operation | | |
| Result or post condition | The program creates a new audio and returns a String validating the operation | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| Result | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R4: Create playlists. | | |
| Abstract | These are list of audio files. Each list has its name, audios (Could be a song or a podcast), and a self-generated numeric code, so the users can use it and easily share their playlists. This numeric code is generated by a matrix (6x6) filled with random numbers between 0 and 9 as follows:   1. A playlist with only Songs: Tour in letter N 2. A playlist with only Podcasts: T-letter tour 3. A playlist with Songs and Podcasts: the number of the boxes i,j, going through the matrix from bottom to top and from right to left, when the sum i+j is an odd number greater than 1.   Both the matrix and the resulting identifier code must be displayed through the user interface each time the user decides to share the list.   |  | | --- | | **Only songs** | | **Code:** 6774462302980066 |  |  | | --- | | **Only Podcast** | | **Code:** 6819237088730412 |  |  | | --- | | **Songs and podcasts** | | **Code:** 4084337206394686 | | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| name | String |  |
| userName | String |  |
| General activities needed to obtain the results | 1. The program receives the name and creates an empty playlist with that name 2. The program will generate a random code that is going to change according to the content of the playlist. 3. A new playlist is created and returns a String confirming the operation | | |
| Result or post condition | The program creates a new enemy and returns a message confirming the operation. | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| Confirmation message | String | The enemy was successfully created. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R5: Edit a playlist. | | |
| Abstract | The program must let the user edit the playlist so the user can change its name or add or delete audios. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| playlistName | String | The playlist must exist before editing it |
| Name | String | The user wants to change the name.  The name cannot be repeated in the user playlist |
| songName | String | The new song that is going to be added |
| General activities needed to obtain the results | 1. The program receives the playlist’s name and checks what do the user wants to change. 2. Change or adds according to the operation selected. 3. The program returns a String validating the operation. | | |
| Result or post condition | The program changes the playlist selected by the user and returns a String with result of the operation | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| result | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R6: Share a Playlist | | |
| Abstract | The program must let the user get the code of a playlist to share it with other people | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| playlistName | String | The playlist must exist before |
| Name | String |  |
| General activities needed to obtain the results | 1. The program receives the playlist’s name and the user’s nickname 2. The system returns the code generated by the list 3. The program returns a String validating the operation. | | |
| Result or post condition | The program returns the playlist code according to the R4 | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| result | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R7: Simulate playing a song or podcast (standard and premium) | | |
| Abstract | The program must emulate the play of a song or a podcast. The user can only play songs he had bought.  The standard consumer user will have an ad per 2 songs and just one ad at the start of every podcast.  The ads will be selected randomly and will be one of the follows:   * Nike - Just Do It. * Coca-Cola - Open Happiness. * M&Ms - Melts in Your Mouth, Not in Your Hands   If the consumer user is premium, the ads won’t show up | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| audioName | String | The song must exist before editing it  The user must own the song |
| userName | String | The user must be a consumer user  The user must exist |
| General activities needed to obtain the results | 1. The program receives the audio’s name and the user’s name, then validates the information 2. The system shows the tittle of the audio that is playing, will show an ad as the resume says 3. The program returns a String ending the emulation | | |
| Result or post condition | The program emulates the play of an audio | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| result | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R8: Buy an audio | | |
| Abstract | The program must let the user buy an audio. The user owns the song so he must keep the song with himself. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| audioName | String | The song must exist being bought  The user cannot buy the song if he already owns it. |
| userName | String | The user must be a consumer user  The user must exist |
| General activities needed to obtain the results | 1. The program receives the audio’s name and the user’s name, then validates the information 2. The system adds a copy of the song to the user 3. The program returns a String confirming the operation | | |
| Result or post condition | The user owns a new song | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| result | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R9: Inform data | | |
| Abstract | The program must generate an inform with the following data:   1. For each type of audio, songs, and podcast, report the cumulative total of streams across the platform. 2. Report the most listened song genre on the entire platform and its number of plays. 3. Report the most listened to podcast category across the platform and its number of views. 4. For each of the members of the Top 5 artists and the Top 5 content creators on the platform, report the name and number of total views. 5. For each of the members of the Top 10 songs and the Top 10 podcasts, report the name, genre or category and total number of views. 6. For each genre, report the number of songs sold and the total sales value ($). 7. From the best-selling song on the platform, report the total number of sales and the total sales value ($). | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| selection | int | The selection must be between the 1 and 7 |
| General activities needed to obtain the results | 1. The program prints a menu to let the user know the options that can be selected 2. The program recompiles the data needed by the user 3. The program returns a String with the data solicitated by the user | | |
| Result or post condition | The system shows the data needed by the user | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| result | String |  |

|  |  |  |
| --- | --- | --- |
| **Functional requirements** | **Class name** | **Method name** |
| R1: Register user producer: artists and content creators. | ProducerUser | ProducerUser(…) |
| IcesiTunes | searchUserById(…) : int |
| IcesiTunes | addUser(…) : String |
| IcesiTunesController | addProducerUser(…) : String |
| Main | uiAddProducerUser() : String |
| R2: Register consumer, standard and premium users. | ConsumerUser | ConsumerUser(…) |
| IcesiTunes | searchUserById(…) : int |
| IcesiTunes | addUser(…) : String |
| IcesiTunesController | addConsumerUser(…) : String |
| Main | uiAddConsumerUser() : String |
| R3: Record songs and podcasts. | Audio | Audio(…) |
| IcesiTunes | addAudioFile(…) : String |
| IcesiTunes | searchAudioByName(…) : String |
| IcesiTunesController | AddAudioFile(…) : String |
| Main | uiAddSongFile(…) : String |
| Main | uiAddPodcastFile(…) : String |
| R4: Create a playlist. | Playlist | Playlist(…) |
| ConsumerUser | AddPlaylist(…) : String |
| IcesiTunes | searchUserById(…) : int |
| IcesiTunes | addPlaylistToUser(…): String |
| IcesiTunesController | addPlaylistToUser(…) : String |
| Main | uiAddPlaylist(…) : String |
| R5: Edit a playlist. | Playlist | AddAudio(…) : String |
| Playlist | deleteAudio(…) : String |
| Playlist | setName(…) : void |
| ConsumerUser | getPlaylist(…) |
| IcesiTunes | addAudioToPlaylist(…) : String |
| IcesiTunes | removeAudioFromPlaylist(…) : String |
| IcesiTunes | changePlaylistName(…) : String |
| IcesiTunesController | addAudioToPlaylist(…) : String |
| IcesiTunesController | removeAudioFromPlaylist(…) : String |
| IcesiTunesController | changePlaylistName(…) : String |
| Main | uiChangePlaylistName() : String |
| Main | uiRemoveSongFromPlaylist() : String |
| Main | uiAddSongToPlaylist() : String |
| Main | uiEditPlaylist() : String |
| R6: Share a Playlist | Playlist | getCode() : String |
| ConsumerUser | getPlaylists() : ArrayList<Playlists> |
| IcesiTunesCOntroller | sharePlaylist(…) : String |
| Main | uiShareCode(…) : String |
| R7: Simulate playing a song or podcast (standard and premium) | ConsumerUser | getAudios(…) : ArrayList<Audio> |
| IcesiTunesController | simulateAudioPlaying(…) : String |
| Main | uiSimulateAudioPlaying(…) : String |
| R8: Buy an audio | ConsumerUser | AddAudio(…) : String |
| IcesiTunes | AddAudioToUser(…) : String |
| IcesiTunesController | AddAudioToUser(…) : String |
| Main | uiBuyAdio(…) : String |
| R9: Inform data | IcesiTunesController | reportData(…) : String |
| Main | uiReportData(…) : String |