**object oriented programming**

Spotivy

Namen: Milan, Jorran

Projectnaam: Spotivy

Docent Robert Nieuwkoop

Vak: OOP (object oriented programming)

Leerjaar: 2

Periode 4

Datum 29-05-2024

Klas: TIA4V2B

Contents

[Beschrijving van de casus 3](#_Toc167875252)

[Client: 3](#_Toc167875253)

[Nummers: 3](#_Toc167875254)

[Speellijsten: 4](#_Toc167875255)

[Gebruikers: 4](#_Toc167875256)

[Albums: 4](#_Toc167875257)

[Artiesten: 4](#_Toc167875258)

[Class Diagram 5](#_Toc167875259)

[Classes 6](#_Toc167875260)

[class Client 6](#_Toc167875261)

[class User 6](#_Toc167875262)

[class Playlist 6](#_Toc167875263)

[class Song 7](#_Toc167875264)

[class Album 7](#_Toc167875265)

[class Artist 7](#_Toc167875266)

[Use cases 8](#_Toc167875267)

[Use Case 1: Create a Playlist 8](#_Toc167875268)

[Use Case 2: Add a Song to a Playlist 9](#_Toc167875269)

[Use Case 3: Play a Playlist 10](#_Toc167875270)

[Use Case 4: View Friends 11](#_Toc167875271)

[Use Case 5: Add a Friend 12](#_Toc167875272)

[Use Case 6: View Friend’s Playlists 13](#_Toc167875273)

[Use Case 7: View Albums 14](#_Toc167875274)

[Use Case 8: View Artists 15](#_Toc167875275)

# Beschrijving van de casus

Hendrik Kavel, een entrepreneur, wil graag een concurrent maken voor Spotify. Hij heeft opgemerkt dat Spotify, ondanks veel goede kwaliteiten, toch wat specifieke functionaliteit mist. Hij heeft ervoor gekozen om jullie in te huren om zijn eigen muziekclient na te maken. Hendrik wil uiteindelijk dat zijn muziekclient echt muziek kan afspelen, maar voor nu vind hij het prima als zijn muziekclient alleen via tekst de functionaliteit laat zien.

De applicatie hoeft nog geen verbinding te maken met een database, het dient voor nu alleen als een platform om nieuwe functionaliteit te testen. Alle nummers, albums en gebruikers mogen dus hard-coded aangemaakt te worden. Binnen de applicatie dient één gebruiker als hoofdgebruiker gekozen te worden. Alle acties die worden ondernomen, worden dan ook vanuit deze hoofdgebruiker uitgevoerd. Dus bijvoorbeeld: als een nieuwe speellijst wordt aangemaakt, dan is deze automatisch van de hoofdgebruiker.

## Client:

Alle CLI interacties dienen via de Client class afgehandeld te worden. Bijvoorbeeld: Wanneer de gebruiker een nummer wil selecteren, dan dient de bijbehorende methode van Client aangeroepen te worden. Alle nummers, albums.

User stories:

Alle CLI interactie dient via de Client class afgehandeld te worden. Dus bijvoorbeeld: Wanneer de gebruiker een nummer wil selecteren, dan dient de daarbij horende methode van Client aangeroepen te worden.

## Nummers:

Nummers hebben een titel, een verzameling aan deelnemende artiesten en een genre. Wanneer een nummer wordt afgespeeld dienen de titel, de deelnemende artiesten en het genre geschreven te worden naar de console.

Wanneer één nummer wordt overgeslagen, dan begint hetzelfde nummer te spelen. Wanneer een nummer wordt gestopt, dan zal de console tonen dat het huidige nummer is gestopt met spelen. Het pauzeren van een nummer hoeft voor het prototype nog niets te doen.

## Speellijsten:

Een gebruiker kan meerdere speellijsten hebben en moet nummers, albums en andere speellijsten kunnen toevoegen aan een speellijst. De gebruiker moet alle nummers van een specifieke speellijst kunnen inzien. Ook moet de gebruiker een nummer uit de speellijst kunnen verwijderen. Tot slot moet de gebruiker een speellijst moeten kunnen afspelen. Bij het afspelen van een speellijst, begint het eerste nummer in de lijst te spelen. Wanneer het laatste nummer van een speellijst is gespeeld, dan stopt het spelen automatisch. Wanneer een nummer uit een speellijst wordt overgeslagen, dan moet het volgende nummer beginnen met spelen.

## Gebruikers:

Een hoofdgebruiker moet kunnen inzien welke andere gebruikers deel zijn van de nieuwe spotify client. -Van deze andere gebruikers moet de hoofgebruiker iemand kunnen toevoegen als vriend. Ook moet de hoofdgebruiker een vriend kunnen verwijderen. De hoofdgebruiker moet alle vrienden kunnen inzien. De hoofdgebruiker moet speellijsten van vrienden kunnen inzien. Dit betekend dat er een lijst van speellijsten naar voren moet komen. De hoofdgebruiker moet vervolgens alle nummers van een specifieke speellijst kunnen inzien. Tot slot, de hoofdgebruiker moet zo’n geselecteerde speellijst kunnen overnemen of toevoegen aan een eigen speellijst.

## Albums:

Albums hebben een titel, een verzameling van artiesten en een verzameling van nummers. Aan albums kunnen geen nieuwe nummers worden toegevoegd. Een hoofdgebruiker moet alle nummers of albums kunnen inzien. Dit betekend dat er een lijst van alle nummers of albums naar voren moet komen. Uit deze nummers of albums moet de gebruiker nummers kunnen selecteren. De gebruiker kan op deze wijze een nummer of album toevoegen aan een eigen speellijst, of kunnen afspelen. Het afspelen van een album gebeurt op dezelfde manier als een speellijst.

## Artiesten:

Artiesten hebben een naam en verzamelingen van albums en nummers waar zij aan hebben gewerkt. De hoofdgebruiker moet een alle artiesten kunnen inzien. Dit betekend dat alle albums en nummers (singles) waar een artiest aan heeft gewerkt naar voren moet komen.

# Class Diagram

A screenshot of a computer

Description automatically generated

# Classes

class Client {

- mainUser: User

+ CreatePlaylist(name: string)

+ AddSongToPlaylist(playlistName: string, song: Song)

+ PlayPlaylist(playlistName: string)

+ ViewFriends()

+ AddFriend(friend: User)

+ RemoveFriend(friend: User)

+ ViewFriendPlaylists(friend: User)

+ ViewAlbums()

+ ViewArtists()

}

class User {

- name: string

- playlists: List<Playlist>

- friends: List<User>

+ CreatePlaylist(name: string)

+ AddSongToPlaylist(playlistName: string, song: Song)

+ PlayPlaylist(playlistName: string)

+ ViewFriends()

+ AddFriend(friend: User)

+ RemoveFriend(friend: User)

+ ViewFriendPlaylists(friend: User)

}

class Playlist {

- name: string

- songs: List<Song>

+ AddSong(song: Song)

+ RemoveSong(song: Song)

+ Play()

}

class Song {

- title: string

- artists: string[]

- genre: string

+ Play()

+ Stop()

+ Skip()

}

class Album {

- title: string

- songs: List<Song>

- artists: List<Artist>

+ Play()

}

class Artist {

- name: string

- albums: List<Album>

- songs: List<Song>

+ ViewWorks()

}

# Use case diagrammen

A diagram of a music player

Description automatically generated

A diagram of a person's relationship

Description automatically generated

A diagram of a music album

Description automatically generatedA diagram of a song

Description automatically generated

# Use cases

## Use Case 1: Create a Playlist

|  |  |
| --- | --- |
|  |  |
| description | This use case describes the steps a user takes to create a new playlist. |
| Actor | Client (mainUser) |
| Version | 1.0 |
| Preconditions | 1. The user is has the application running. 2. The user is on the section where playlists can be managed. |
| Main case | 1. The user selects the option to create a new playlist. 2. The system prompts the user to enter a name for the playlist. 3. The user enters a playlist name. 4. The system validates the name to ensure it is not empty and does not exceed character limits. 5. The system calls CreatePlaylist(name: ‘playlist name’) on the User class. 6. The User class creates a new Playlist object with the name ‘playlist name’. 7. The new playlist is added to the user's list of playlists. 8. The system confirms that the playlist has been created and displays the new playlist in the user's playlist list. |
| Postconditions | A new playlist is created in the user's account and is visible in the list of playlists. |
| Exception path | If the user enters an empty name or a name that exceeds the character limit, the system displays an error message indicating the issue and prompts the user to enter a valid name. |

## Use Case 2: Add a Song to a Playlist

|  |  |
| --- | --- |
|  |  |
| Description | This use case describes the steps a user takes to add a song to an existing playlist. |
| Actor | Client (mainUser) |
| Version | 1.0 |
| Preconditions | 1. The user is has the application running. 2. The user has an existing playlist to which they want to add a song. |
| Main case | 1. The user navigates to their list of playlists. 2. The user selects a playlist. 3. The user selects the option to add a song to the playlist. 4. The system prompts the user to search for a song. 5. The user enters a song name. 6. The system displays a list of results matching the name. 7. The user selects a song they want to add to the playlist. 8. The system calls AddSongToPlaylist(playlistName: ‘song name, song: …) on the User class. 9. The User class finds the playlist and calls AddSong(song: ‘song name’) on the Playlist class. 10. The Playlist class adds the song to the list of songs in the playlist. 11. The system confirms that the song has been added and updates the playlist to show the new song. |
| Postconditions | The song is added to the playlist and is visible in the list of songs within the playlist. |
| Exception path | 1. If the user searches for a song and no results are found, the system displays a message indicating that no matches were found and prompts the user to try another search term. |

## Use Case 3: Play a Playlist

|  |  |
| --- | --- |
|  |  |
| Description | This use case describes the steps a user takes to play a playlist within the music streaming application. |
| Actor | Client (mainUser) |
| Version | 1.0 |
| Preconditions | 1. The user is has the application running. 2. The user has at least one playlist with songs in it. |
| Main case | 1. The user navigates to their playlists. 2. The user selects a playlist. 3. The system displays the list of songs in the playlist. 4. The user selects the option to play the playlist. 5. The system calls PlayPlaylist(playlistName: ‘playlist name’) on the User class. 6. The User class retrieves the playlist and calls Play() on the Playlist class. 7. The system begins playing the first song in the playlist. 8. The user listens to the playlist, and the system continues to play the subsequent songs in the playlist order. |
| Postconditions | The playlist is playing, starting with the first song and continuing through the list. |
| Exception path | 1. If the playlist is contains no songs, the system displays an error message indicating that the playlist has no songs to play. |

## 

## Use Case 4: View Friends

|  |  |
| --- | --- |
|  |  |
| Description | This use case describes the steps a user takes to view their list of friends. |
| Actor | Client (mainUser) |
| Version | 1.0 |
| Preconditions | 1. The user is has the application running. 2. The user has at least one friend added to their friends list. |
| Main case | 1. The user navigates to the friends section of the application. 2. The system calls ViewFriends() on the User class. 3. The User class retrieves the list of friends from the user's account. 4. The system displays the list of friends to the user, showing each friend's name. 5. The user views the list of friends and can select a friend to view more details or interact further. |
| Postconditions | The user sees an updated list of their friends, displayed in the friends section of the application. |
| Exception path | 1. If the user has no friends added to their list, the system displays a message indicating that no friends have been added. |

## Use Case 5: Add a Friend

|  |  |
| --- | --- |
|  |  |
| Description | This use case describes the steps a user takes to add a new friend to their friends list. |
| Actor | Client (mainUser) |
| Version | 1.0 |
| Preconditions | 1. The user is has the application running. 2. The user knows the username or other identifying information of the friend they want to add. |
| Main case | 1. The user navigates to the friends section of the application. 2. The user selects the option to add a friend. 3. The system prompts the user to enter the friend's username. 4. The user enters the friend's username. 5. The system searches for the user. 6. The system finds a match and displays the friend's profile summary to the user. 7. The user confirms they want to add the user as a friend. 8. The system calls AddFriend(friend: ‘name’) on the User class. 9. The User class updates the user's friends list. 10. The system confirms that the friend has been added to the friends list and displays the updated friends list to the user. |
| Postconditions | The user is added to the user's (mainUser’s) friends list and is visible in the friends section of the application. |
| Exception path | 1. If the user enters an invalid or non-existent username, the system displays a message indicating that the user was not found and prompts the user to enter a valid username. 2. If the user tries to add a friend who is already in their friends list, the system displays a message indicating that the user is already a friend. |

## Use Case 6: View Friend’s Playlists

|  |  |
| --- | --- |
|  |  |
| Description | This use case describes the steps a user takes to view the playlists of a friend. |
| Actor | Client (mainUser) |
| Version | 1.0 |
| Preconditions | 1. The user is has the application running 2. The user has at least one friend who has public playlists. |
| Main case | 1. The user navigates to their friends list. 2. The user selects a friend from the list. 3. The system calls ViewFriendPlaylists(friend: ‘friend name’) on the User class. 4. The User class retrieves the list of public playlists from the friend's account. 5. The system displays the list of playlists created by the friend to the user. 6. The user views the list of playlists. |
| Postconditions | The user views the list of playlists created by their friend. |
| Exception path | 1. If the friend has no public playlists, the system displays a message indicating that the friend has no public playlists available. |

## Use Case 7: View Albums

|  |  |
| --- | --- |
|  |  |
| Description | This use case describes the steps a user takes to view the albums available. |
| Actor | Client (mainUser) |
| Version | 1.0 |
| Preconditions | 1. The user is has the application running |
| Main case | 1. The user navigates to the albums section of the application. 2. The system calls ViewAlbums() on the Client class. 3. The Client class retrieves a list of available albums from the system’s database. 4. The system displays the list of albums to the user, including details such as album title, artist. 5. The user browses through the list of albums. |
| Postconditions | The user views a list of albums available on the platform. |
| Exception path | 1. If there are no albums available in the system, the system displays a message indicating that no albums are available at the moment. |

## Use Case 8: View Artists

|  |  |
| --- | --- |
|  |  |
| Description | This use case describes the steps a user takes to view the list of artists available. |
| Actor | Client (mainUser) |
| Version | 1.0 |
| Preconditions | 1. The user is has the application running |
| Main case | 1. The user navigates to the artists section of the application. 2. The system calls ViewArtists() on the Client class. 3. The Client class retrieves a list of available artists from the system’s database. 4. The system displays the list of artists to the user, including details such as artist name. 5. The user browses through the list of artists. |
| Postconditions | The user views a list of artists available on the platform. |
| Exception path | 1. If there are no artists available in the system, the system displays a message indicating that no artists are available at the moment. |