Music Playlist [Pure Html]

Lorenzo Campana

June 5, 2021

Matricola:	907081
Codice Persona:	10605775
Docente:	Piero Fraternali

1 Requirements

Un'applicazione web consente la gestione di una playlist di brani musicali. Playlist e brani sono personali di ogni utente e non condivisi. Ogni brano musicale è memorizzato nella base di dati mediante un titolo , l'immagine e il titolo dell'album da cui il brano è tratto, il nome dell'interprete (singolo o gruppo) dell'album, l'anno di pubblicazione dell'album, il genere musicale (si supponga che i generi siano prefissati) e il file musicale. L'utente, previo login, può creare brani mediante il caricamento dei dati relativi e raggrupparli in playlist. Una playlist è un insieme di brani scelti tra quelli caricati dallo stesso utente ordinati per data decrescente dall'anno di pubblicazione dell'album. Una playlist ha un titolo e una data di creazione ed è associata al suo creatore. A seguito del login, l'utente accede all'HOME PAGE che presenta l'elenco delle proprie playlist, ordinate per data di creazione decrescente, una form per caricare un brano con tutti i dati relativi e una form per creare una nuova playlist inizialmente vuota. Quando l'utente clicca su una playlist nell'HOME PAGE, appare la pagina PLAYLIST PAGE che contiene inizialmente una tabella di una riga e cinque colonne. Ogni cella contiene il titolo di un brano e l'immagine dell'album da cui proviene. Se la playlist è inizialmente vuota compare un messaggio: "La playlist non contiene ancora brani musicali". I brani sono ordinati da sinistra a destra per data decrescente dell'album di pubblicazione. Se la playlist contiene più di cinque brani, sono disponibili comandi per vedere il precedente e successivo gruppo di brani. Se la pagina PLAYLIST mostra il primo gruppo e ne esistono altri successivi nell'ordinamento, compare a destra della riga il bottone SUCCESSIVI, che permette di vedere il gruppo successivo. Se la pagina PLAYLIST mostra l'ultimo gruppo e ne esistono altri precedenti nell'ordinamento, compare a sinistra della riga il bottone PRECEDENTI, che permette di vedere i cinque brani precedenti. Se la pagina PLAYLIST mostra un blocco e esistono sia precedenti sia successivi, compare a destra della riga il bottone SUCCESSIVI e a sinistra il bottone PRECEDENTI. La pagina PLAYLIST contiene anche una

form che consente di selezionare e aggiungere un brano alla playlist corrente. A seguito dell'aggiunta di un brano alla playlist corrente, l'applicazione visualizza nuovamente la pagina a partire dal primo blocco della playlist. Quando l'utente seleziona il titolo di un brano, la pagina PLAYER mostra tutti i dati del brano scelto e il player audio per la riproduzione del brano.

2 Database design

2.1 ER diagram

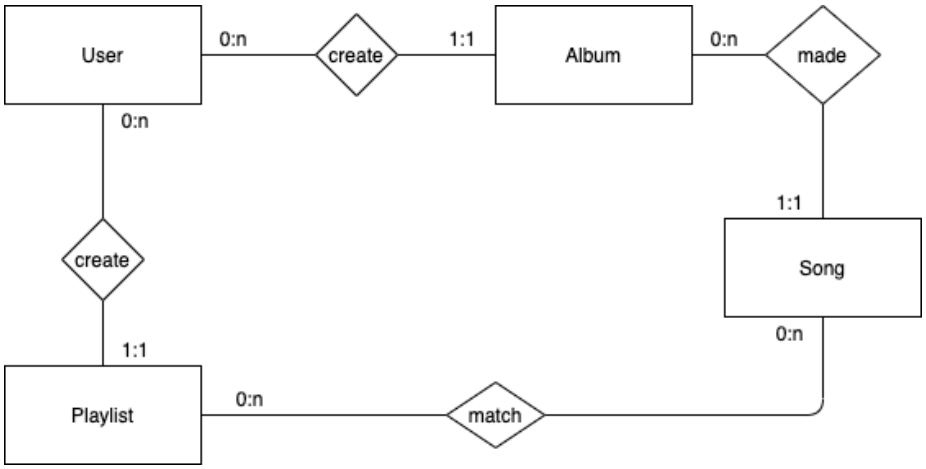

Figure 1: ER diagram

2.2 Database model

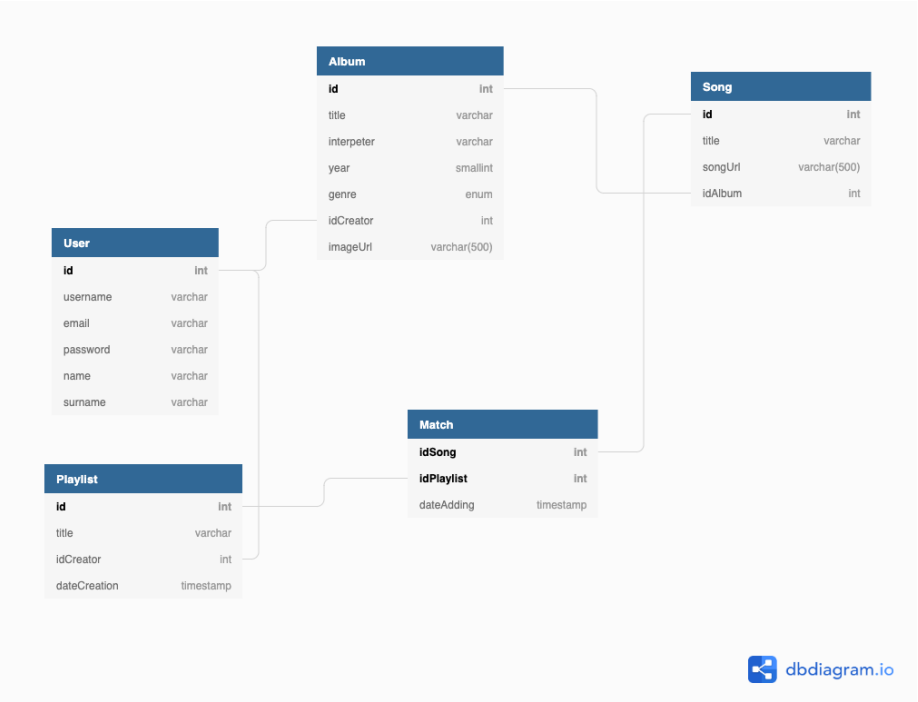

Figure 2: database model

3 Application Design

3.1 Components

- Model objects (Beans)
  - `User`
  - `Genre`
  - `Match`
  - `Playlist`
  - `Song`
  - `Album`
- Data Access Objects (Classes)
  - AlbumDAO
    - `int createAlbum(Album album)`
    - `int findAlbumId(Album album)`
    - `Album findAlbumById(int albumId)`
    - `ArrayList<Album> findAllUserAlbumsById(int userId)`
  - UserDAO
    - `int createUser(User user)`
    - `User findUserById(int idUser)`
    - `int findIdOfUserByEmail(String email)`
    - `boolean isPasswordCorrect(int idUser, String password)`
  - SongDAO
    - `int createSong(Song song)`
    - `int findSongId(Song song)`
    - `Song findSongById(int songId)`
    - `ArrayList<Song> findAllSongByUserId(int userId)`
  - PlaylistDAO
    - `int createPlaylist(String title, int idCreator)`
    - `ArrayList<Playlist> findAllPlaylistByUserId(int userId)`
    - `Playlist findPlaylistById(int playlistId)`
  - MatchDAO
    - `int createMatch(Match match)`
    - `ArrayList <Integer> findAllSongIdOfPlaylist(int idPlaylist, int userId)`
- Controllers (Servlets)
  - `AddSongToPlaylist`

  - `CreateAlbum`
  - `CreatePlaylist`
  - `CreateSong`
  - `GetHomePage`
  - `GetPlayer`
  - `GetPlaylist`
  - `ShowFile`
  - `SubmitLogin`
  - `SubmitRegistration`
  - `Logout`
- Views (Templates)
  - `ErrorPage.html`
  - `HomePage.html`
  - `Login.html`
  - `Player.html`
  - `PlaylistPage.html`
  - `Register.html`

3.2 IFML

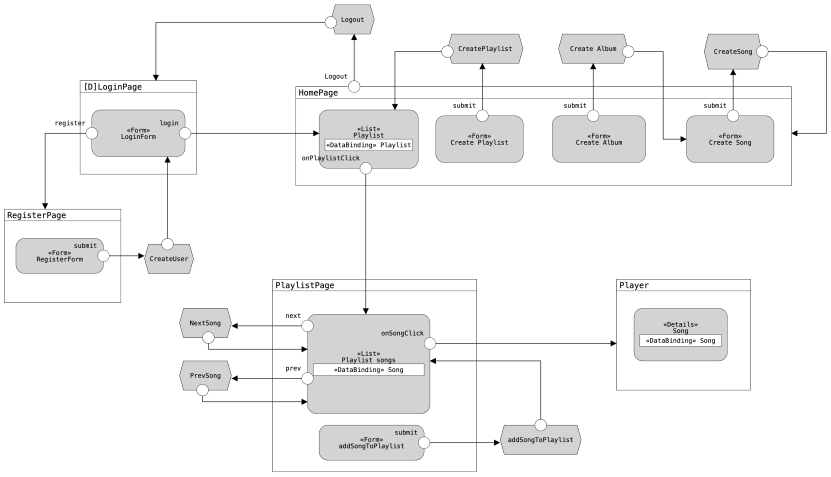

Figure 3: Ifml model

4 Event sequence diagram

4.1 Forward method

To simplify the sequence diagram below, each servlet uses the method `forward(HttpServletRequest request, HttpServletResponse response, String path)` that is responsible to take the WebContext and process the TemplateEngine as shown below

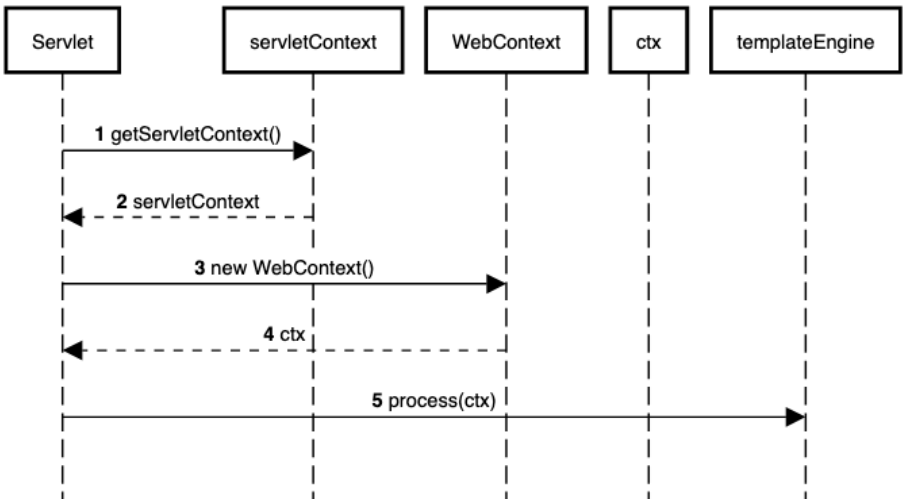

Figure 4: forward method

4.2 Login

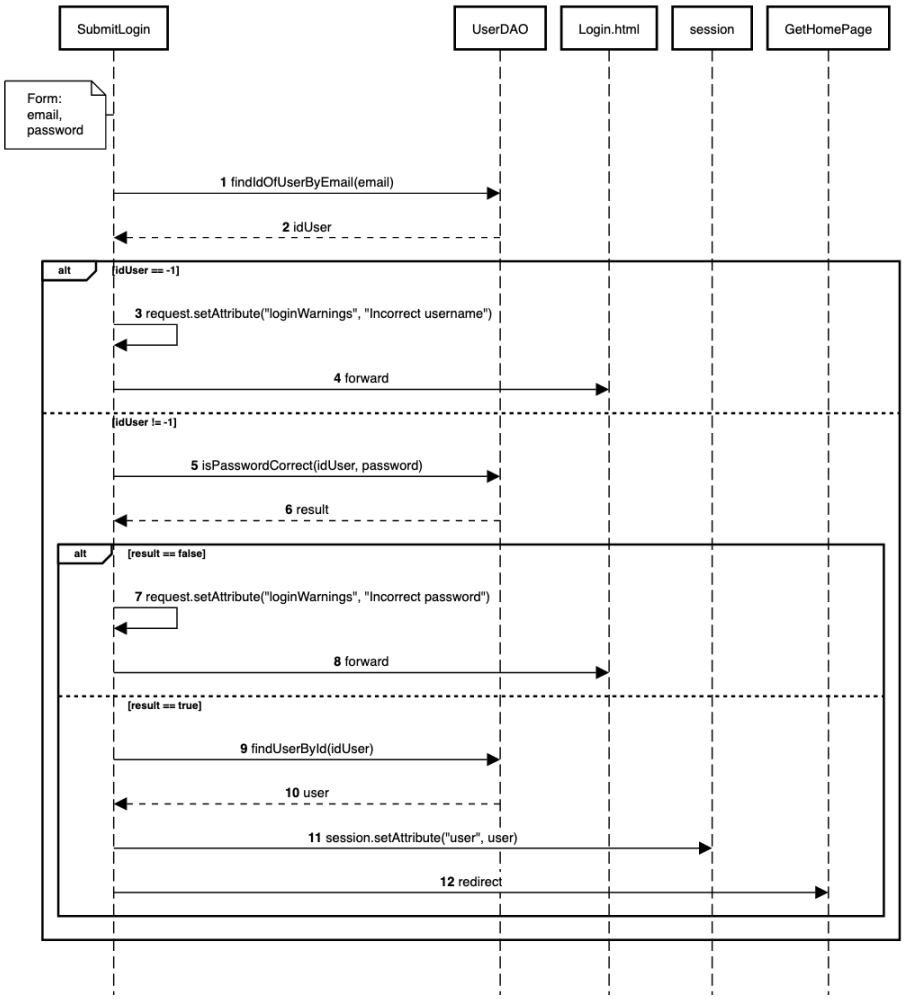

Figure 5: POST /SubmitLogin

4.3 Registration

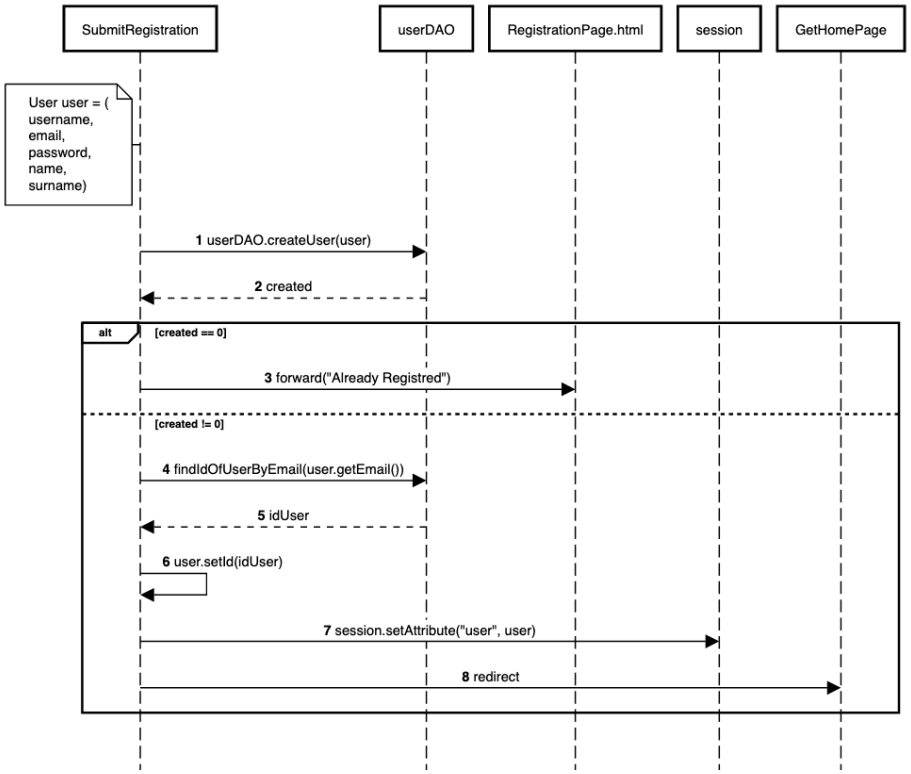

Figure 6: POST /SubmitRegistration

4.4 Logout

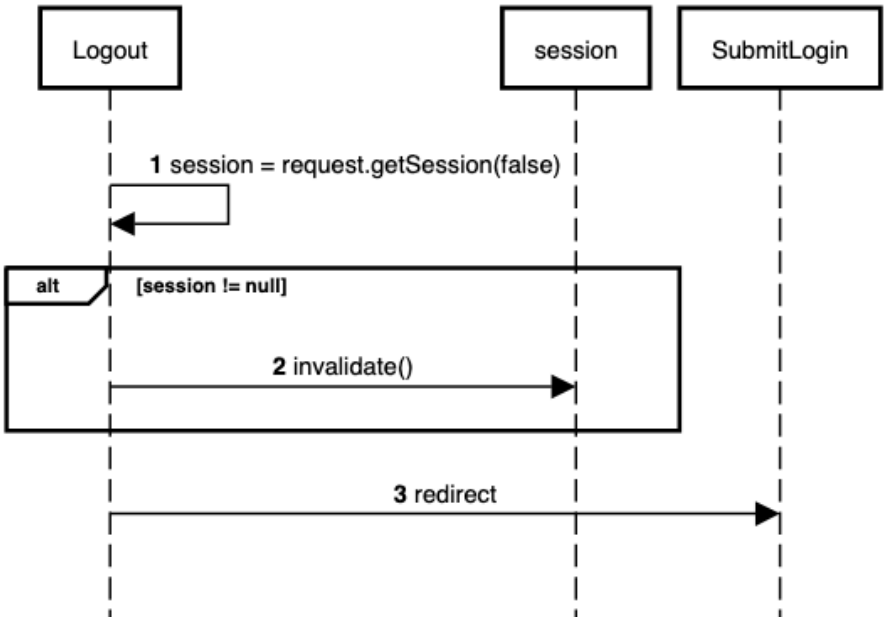

Figure 7: POST /Logout

4.5 Home Page

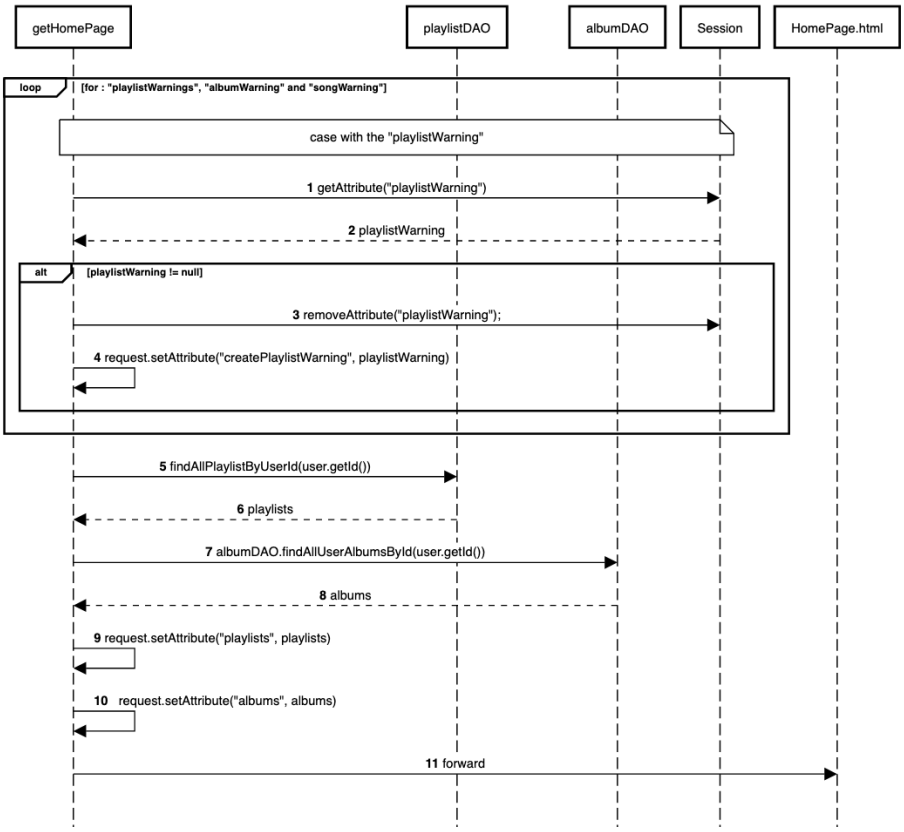

Figure 8: GET /GetHomePage

4.5.1 Playlist creation

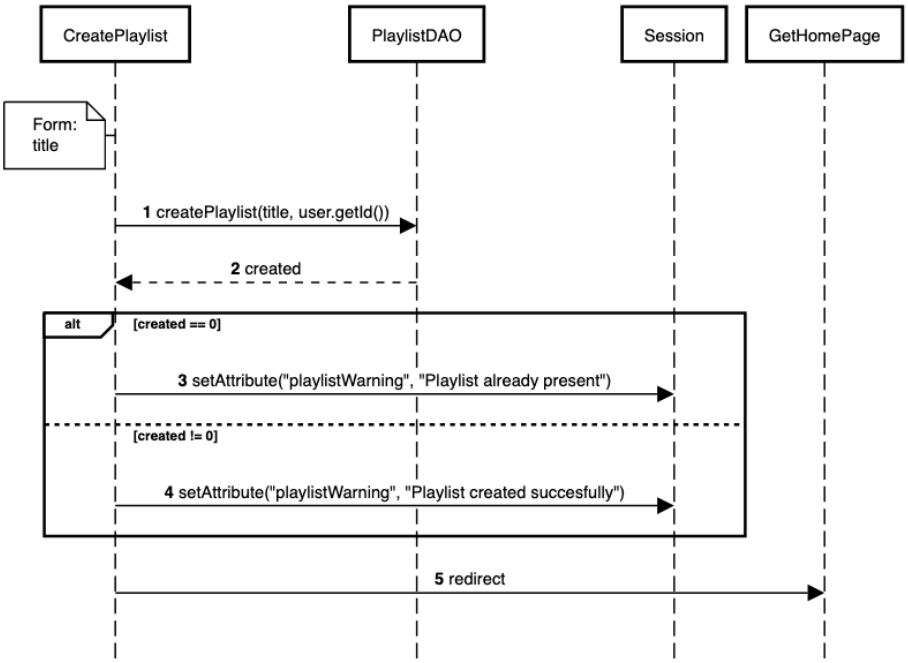

Figure 9: POST /CreatePlaylist

4.5.2 Song creation

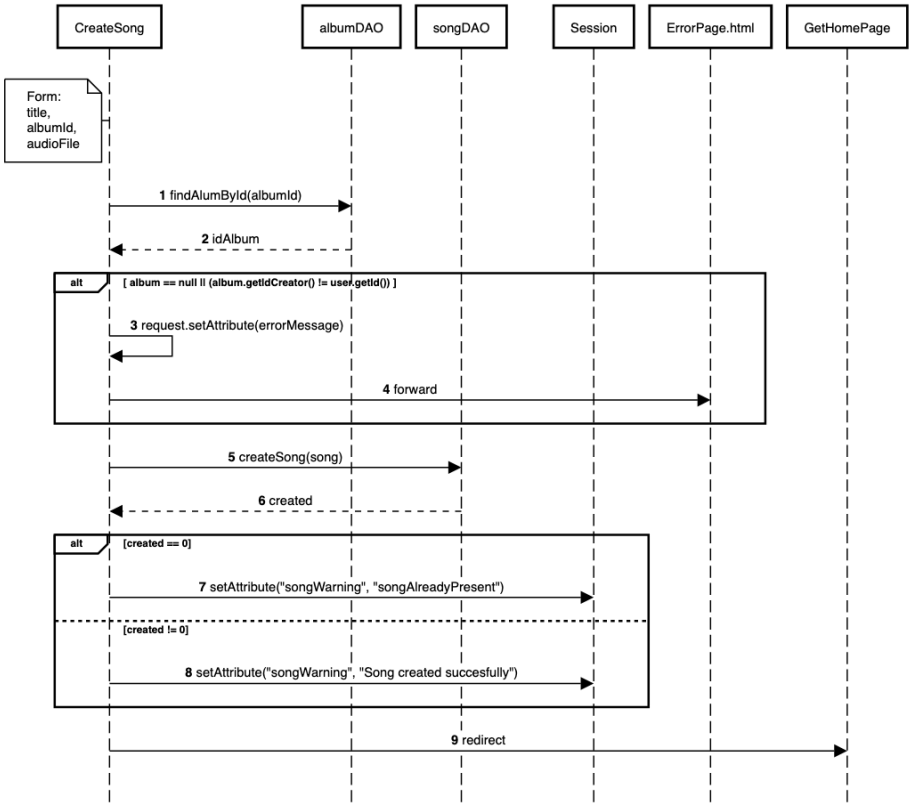

Figure 10: POST /CreateSong

4.5.3 Album creation

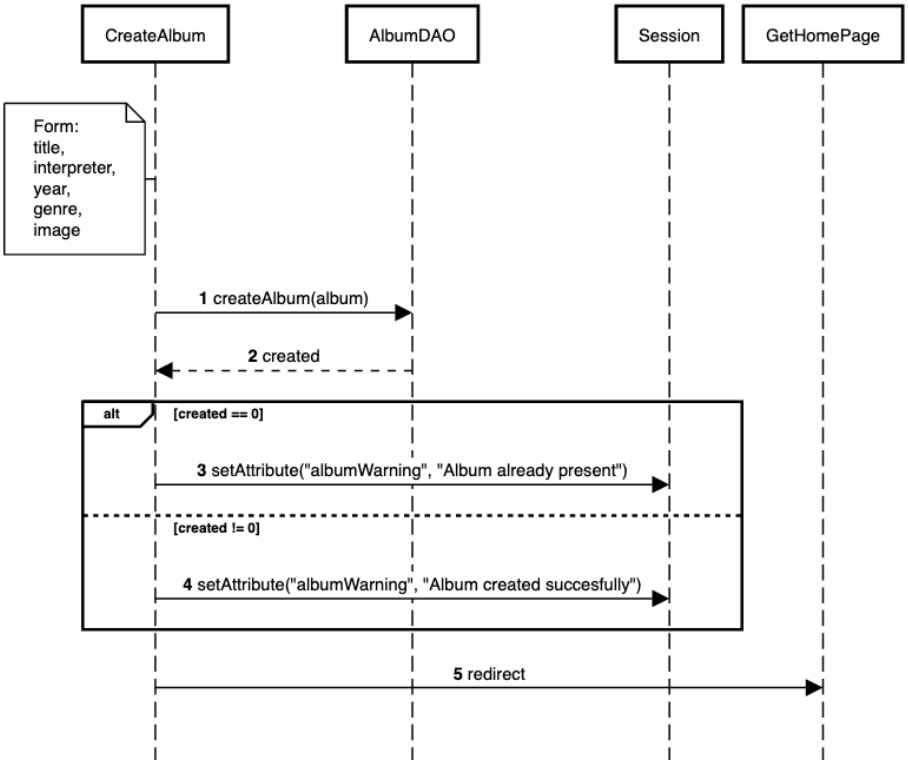

Figure 11: POST /CreateAlbum

4.6 Playlist Page

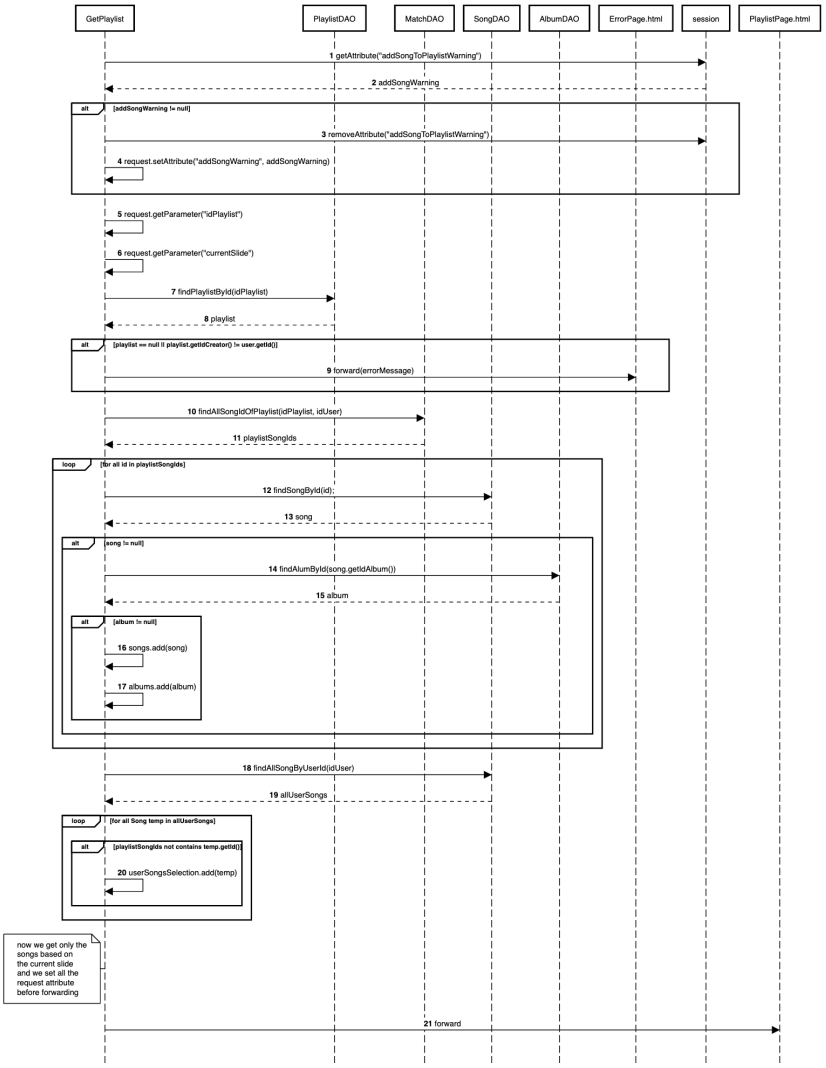

Figure 12: GET /GetPlaylist

4.6.1 Add song to playlist

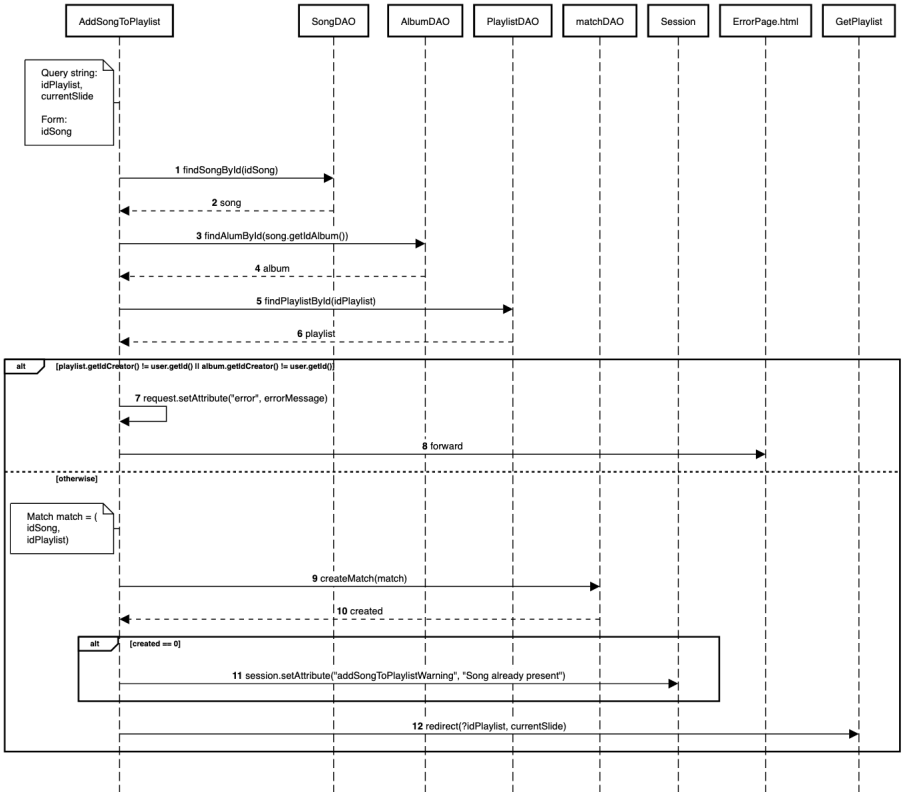

Figure 13: POST /AddSongToPlaylist

4.7 Player Page

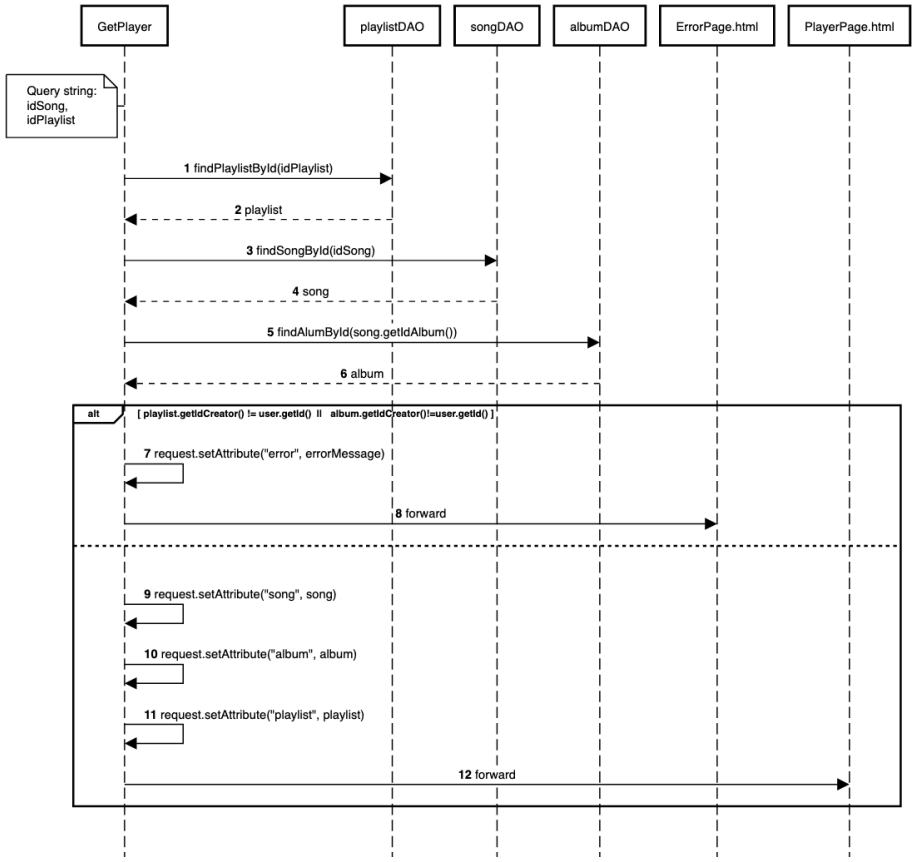

Figure 14: GET /GetPlayer