Requirements Analysis

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 *tramite form dedicato*, 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. Lo stesso brano può essere inserito in più playlist. 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. La creazione di una nuova playlist richiede di selezionare uno o più brani da includere. 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, se non già presente nella playlist. 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.

Pages View components Actions Events

Requirements Analysis

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 *tramite form dedicato*, 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. Lo stesso brano può essere inserito in più playlist. 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. La creazione di una nuova playlist richiede di selezionare uno o più brani da includere. 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, se non già presente nella playlist. 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.

Entities Attributes Relationships Keys

Database Design

Logical scheme:

users (username, password, nome, cognome)
songs (title, owner, author, album, genre, albumYear)
playlist (playlistOwner, playlistSong, playlistName, albumYear, creationDate, songIndexJs)

Foreign keys:

songs.owner -> users.username
playlist.playlistOwner -> songs.owner
playlist.playlistSong -> songs.title
playlist.albumYear -> songs.albumYear

Database Design

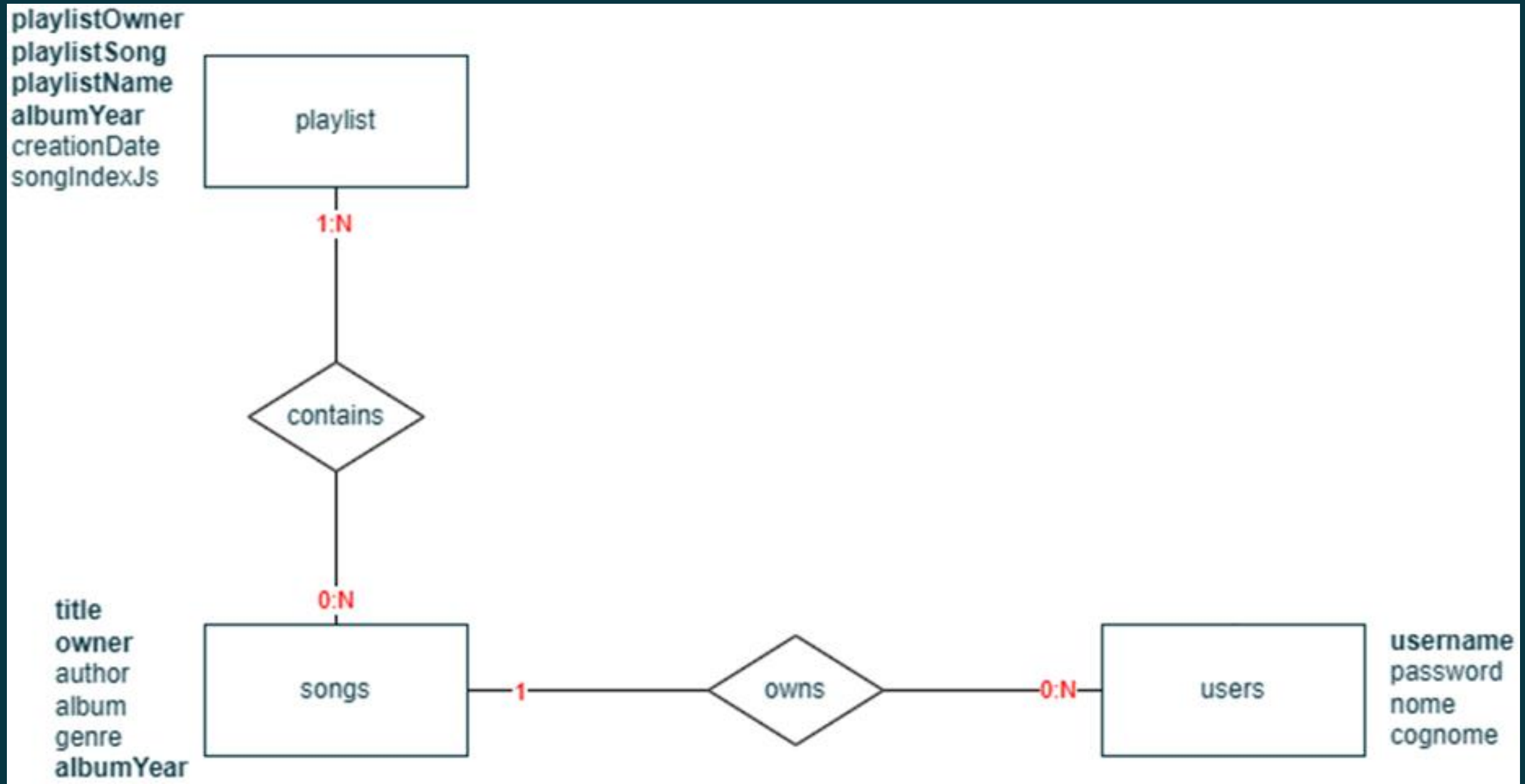


Table Definition

users:

```
CREATE TABLE `users` ( `username` varchar(45) NOT NULL, `password` varchar(45) NOT NULL, `nome` varchar(45) NOT NULL, `cognome` varchar(45) NOT NULL, PRIMARY KEY (`username`))
```

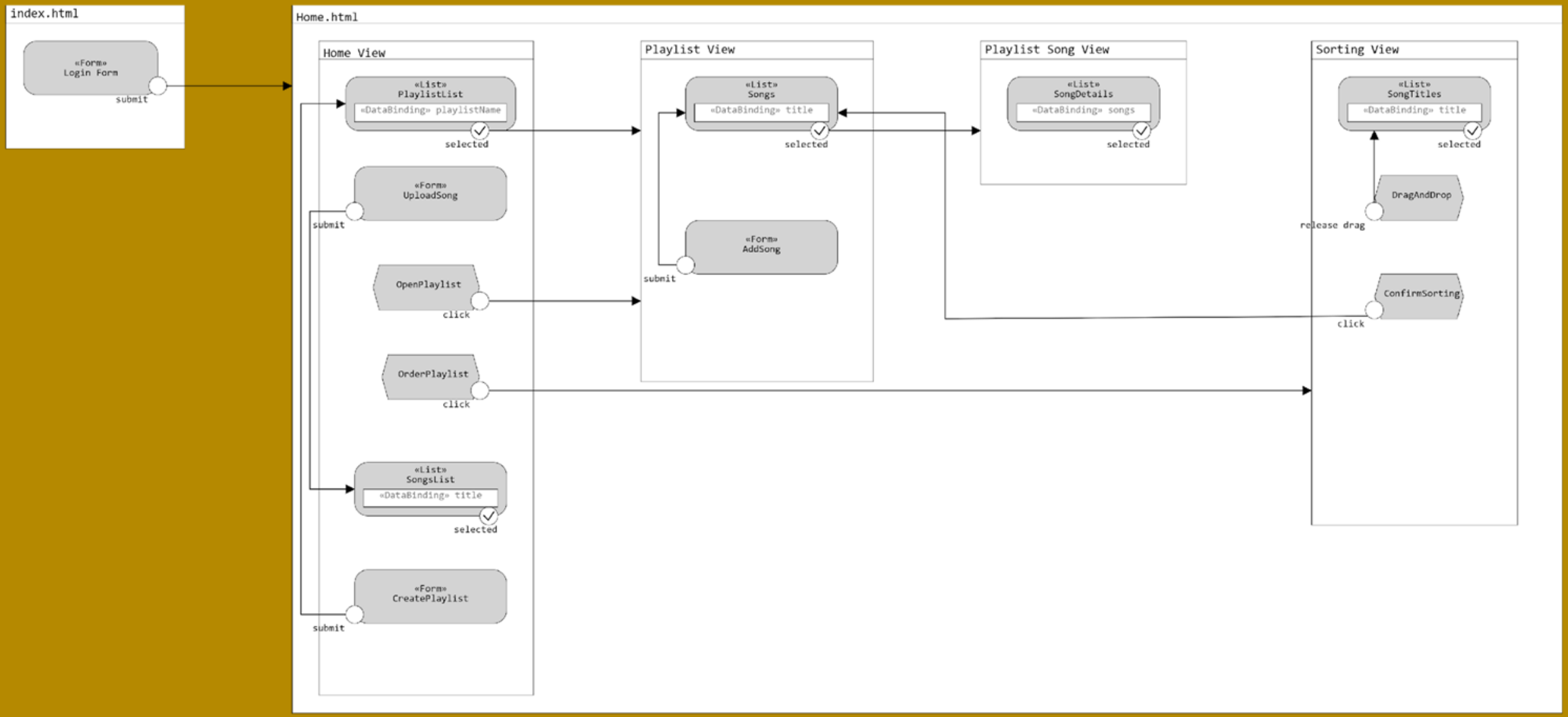
playlist:

```
CREATE TABLE `playlist` ( `playlistOwner` varchar(45) NOT NULL, `playlistSong` varchar(45) NOT NULL, `playlistName` varchar(45) NOT NULL, `albumYear` int NOT NULL, `creationDate` datetime NOT NULL, `songIndexJs` int NOT NULL, PRIMARY KEY (`playlistOwner`,`playlistName`,`playlistSong`))
```

songs:

```
CREATE TABLE `songs` ( `title` varchar(45) NOT NULL, `owner` varchar(45) NOT NULL, `author` varchar(45) NOT NULL, `album` varchar(45) NOT NULL, `genre` varchar(45) NOT NULL, `albumYear` int NOT NULL, PRIMARY KEY (`title`,`owner`))
```

Application Design (IFML)



Components

Model Objects (Beans)
- Playlist.java
- Song.java
- User.java

Views (Templates)
- index.html
- home.html

Utils
- ConnectionHandler.java

Filters
- LoginChecker.java

Controllers (Servlets)
- AddSong.java
- AlterSongOrder.java
- AudioPlayer.java
- CheckLogin.java
- CoverServlet.java
- CreatePlaylist.java
- CreateSong.java
- GetPlaylistList.java
- GetPlaylistSongs.java
- GetSongDetailsAsJson.java
- GetSongList.java

Components: Data Access Objects (Classes)

PlaylistDAO

- createPlaylist (playlistOwner, playlistSongs, playlistName, albumYears, creationDate)
- getPlaylistsOf (playlistOwner)
- getSongsNumOfPlaylistOf (playlistOwner, playlistName)
- getSongsOfPlaylistOf (playlistOwner, playlistName)
- getFiveSongsAtMost (playlistOwner, playlistName, offset)
- getPlaylistNextFreeIndex (playlistOwner, playlistName)
- getPlaylistDate (playlistOwner, playlistName)
- addSong (username, playlistName, newSong, albumYear)
- alterSongOrderJs (username, playlistName, songs)

SongDAO

- uploadData (title, owner, author, album, genre, albumYear)
- doesSongExist (title, owner)
- getSongsOf (owner)
- getSongDetails (owner, title)

UserDAO

- checkCredentials (usrn, pwd)

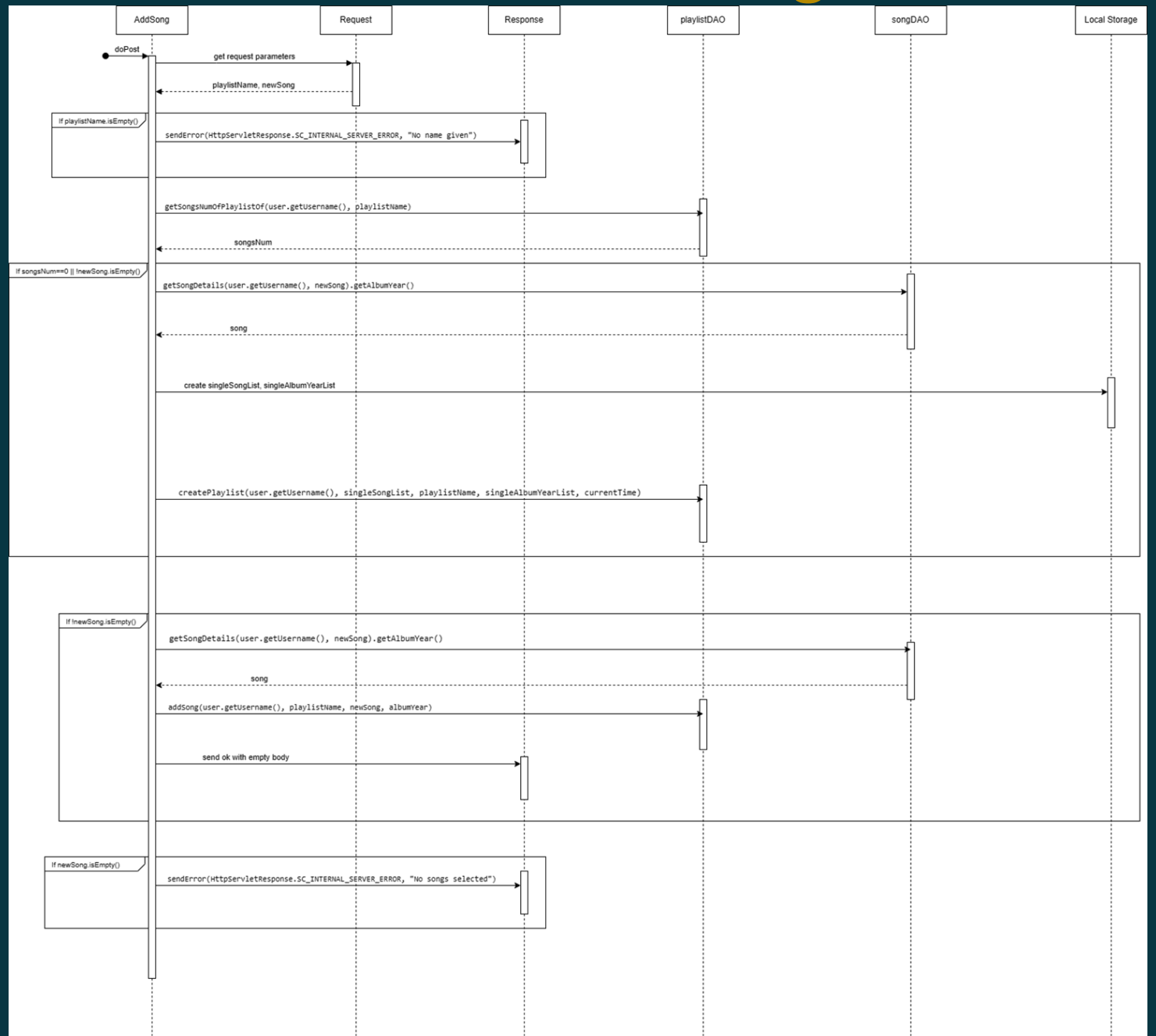
Eventi JavaScript

Client Side	Client Side	Server Side	Server Side
Evento	Azione	Evento	Azione
index -> login form -> submit	Controlla validità	POST user/pass	Controlla Credenziali
load home page 1	controlla session storage per validità pagina e personalizzazione	/	/
load home page 2	mostra lista playlist	GET playlist list	estrazione dati playlist dell'utente
load home page 3	mostra form crea playlist	GET song list	estrazione dati canzoni dell'utente
click create playlist button	check form validity	POST create playlist	controllo dati e creazione playlist con le canzoni fornite
click upload song button	check form validity	POST create song	controllo dati e creazione canzone con i dati forniti
show playlist page 1 (click on playlist name)	mostra e aggiungi listener al bottone "indietro"	/	/

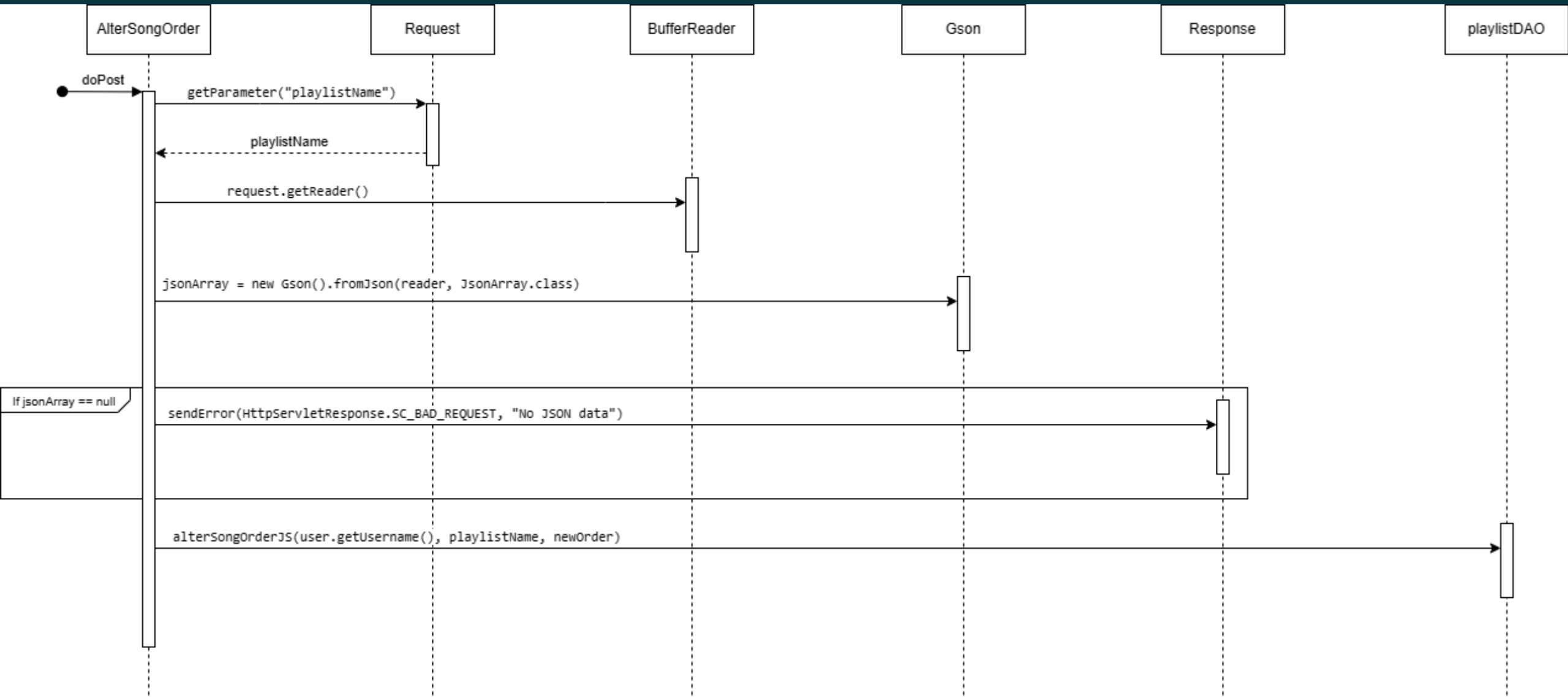
Eventi JavaScript

Client Side	Client Side	Server Side	Server Side
Evento	Azione	Evento	Azione
show playlist page 2 (click on playlist name)	ripulisci la pagina dai vecchi dati e mostra quelli nuovi	GET playlist songs (playlistName)	estrazione dati canzoni della playlist
show playlist page 3 (click on playlist name)	crea addSongForm	GET all songs list	estrazione lista di tutte le canzoni dell'utente
add song (click)	form check validity	POST add song	controlla dati e aggiunge la nuova canzone alla playlist
show reorder page	aggiunge bottone "salva" e ripulisce dai vecchi dati e mostra quelli nuovi	GET playlist songs (playlistName)	estrazione lista delle canzoni della playlist
reorder songs	craft newOrder : String[]	POST alter song older (playlistName)	controllo dati e riordinamento canzoni nella playlist
show player page (click)	mostra la player bar e i dati ricevuti + un elemento dom "audio" per ascoltarla	GET song details as json	estrazione di tutti i dati della canzone richiesta
next/prev button	manipola il dom per mostrare 5 canzoni per volta, scorrendole.	/	/

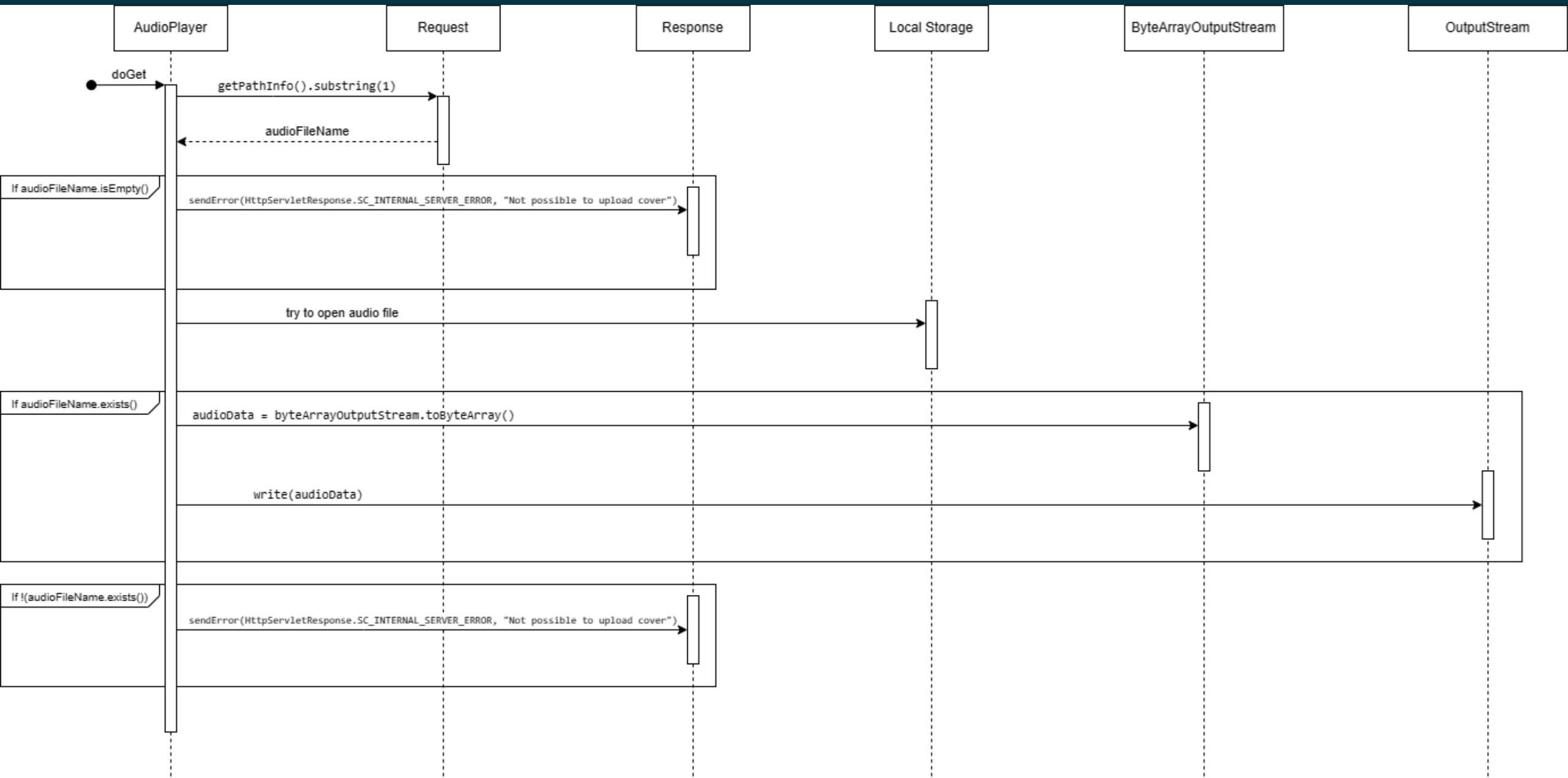
Event: Add Song



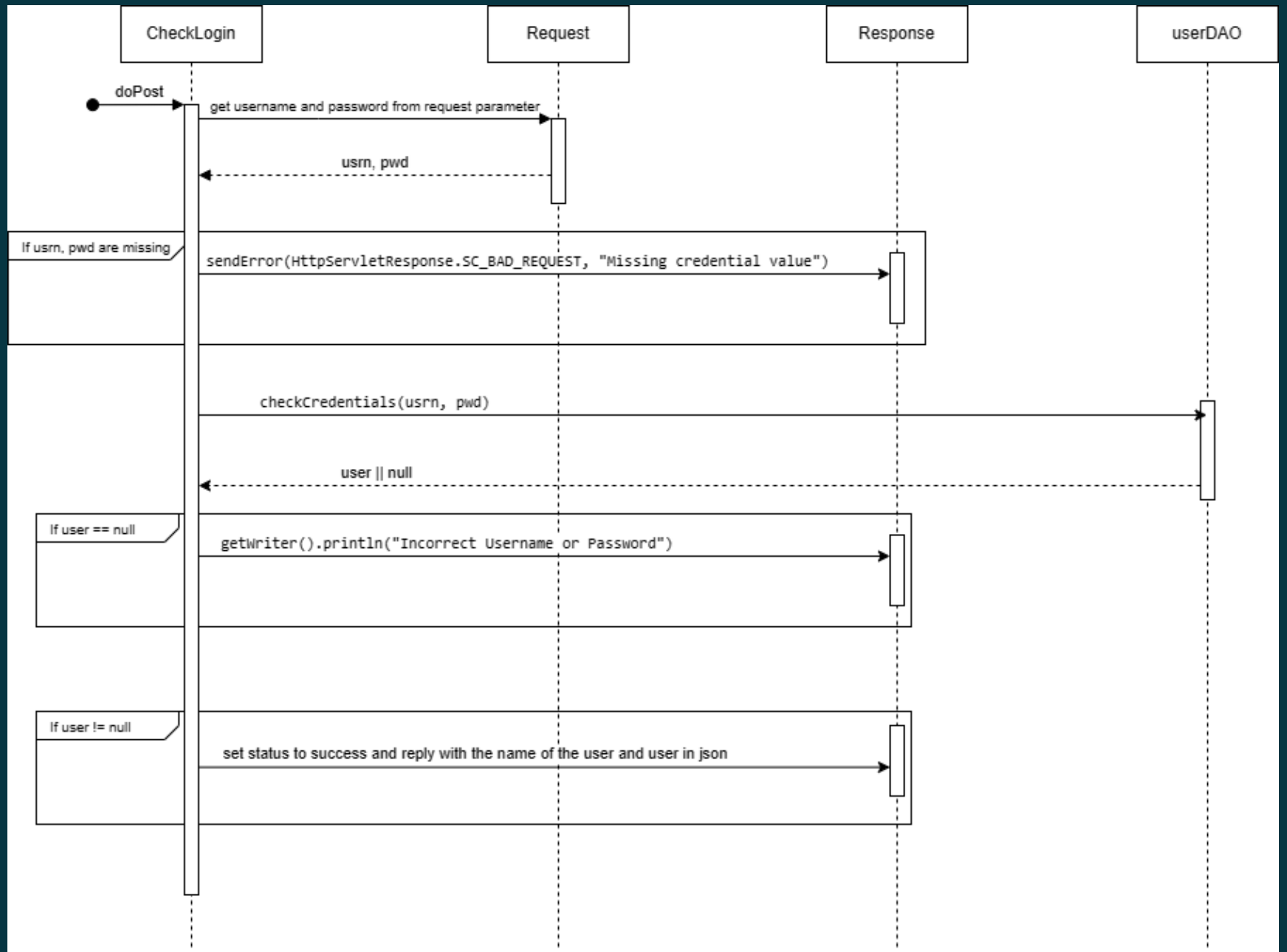
Event: Alter Song Order



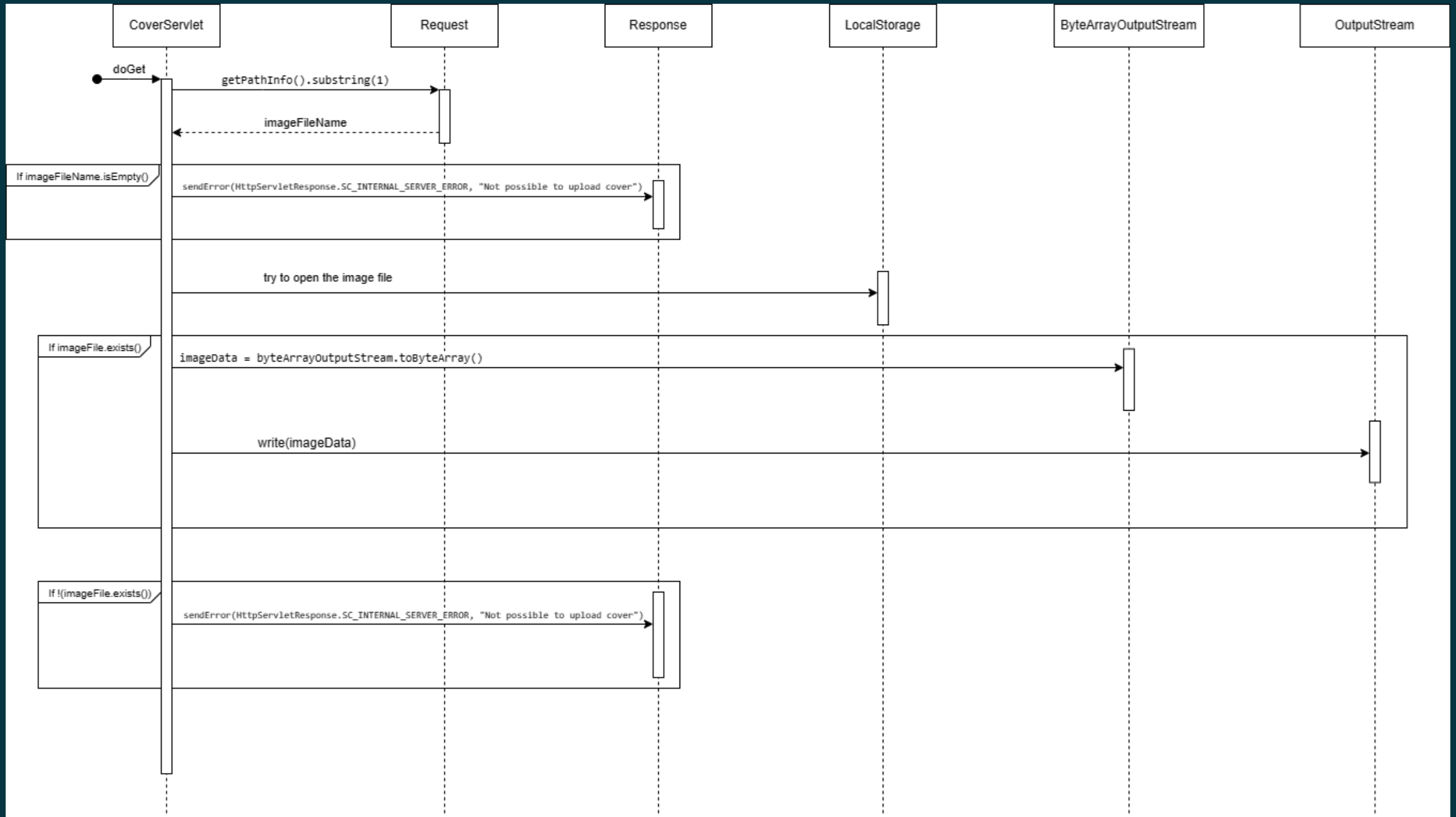
Event: Audio Player



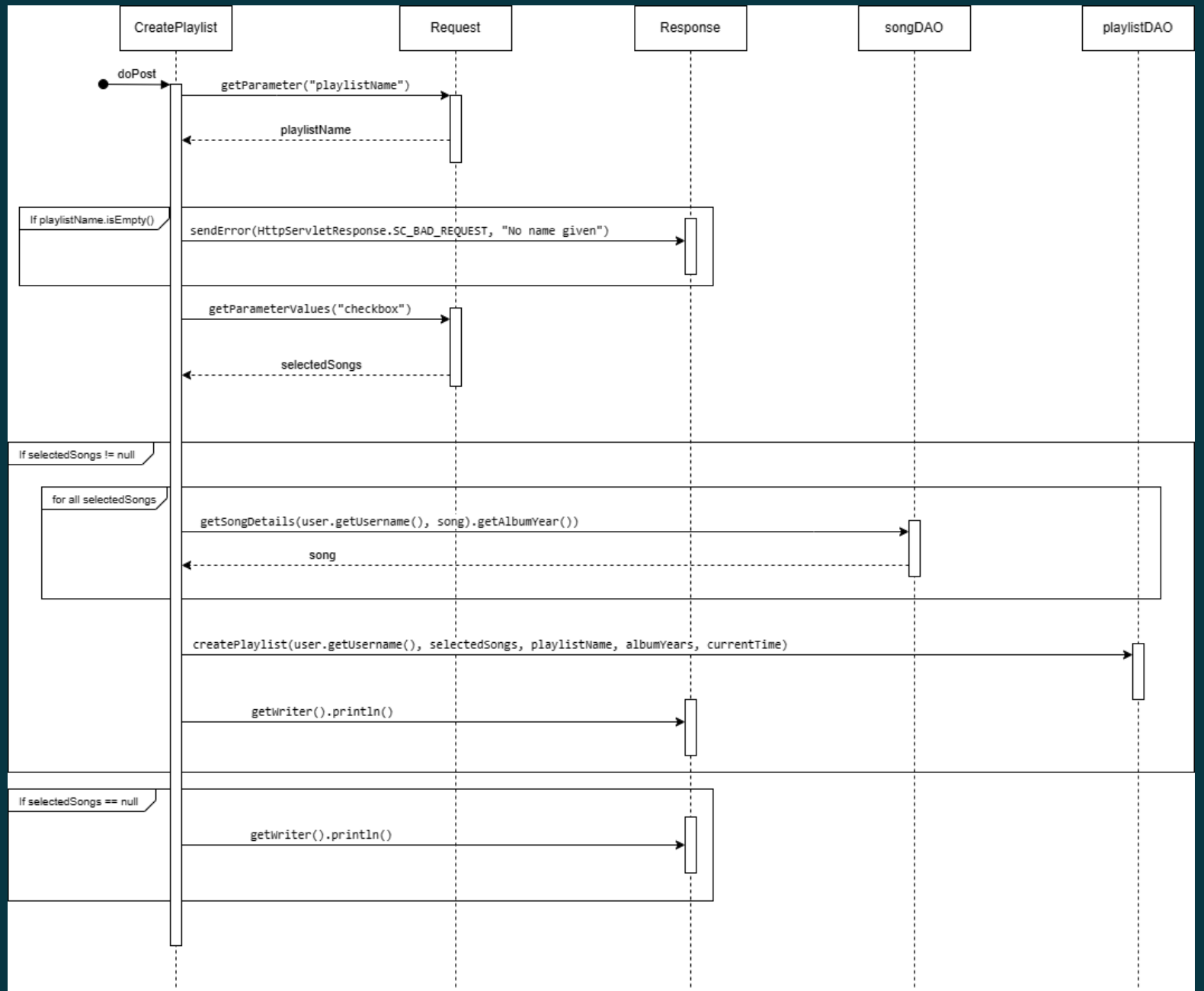
Event: Check Login



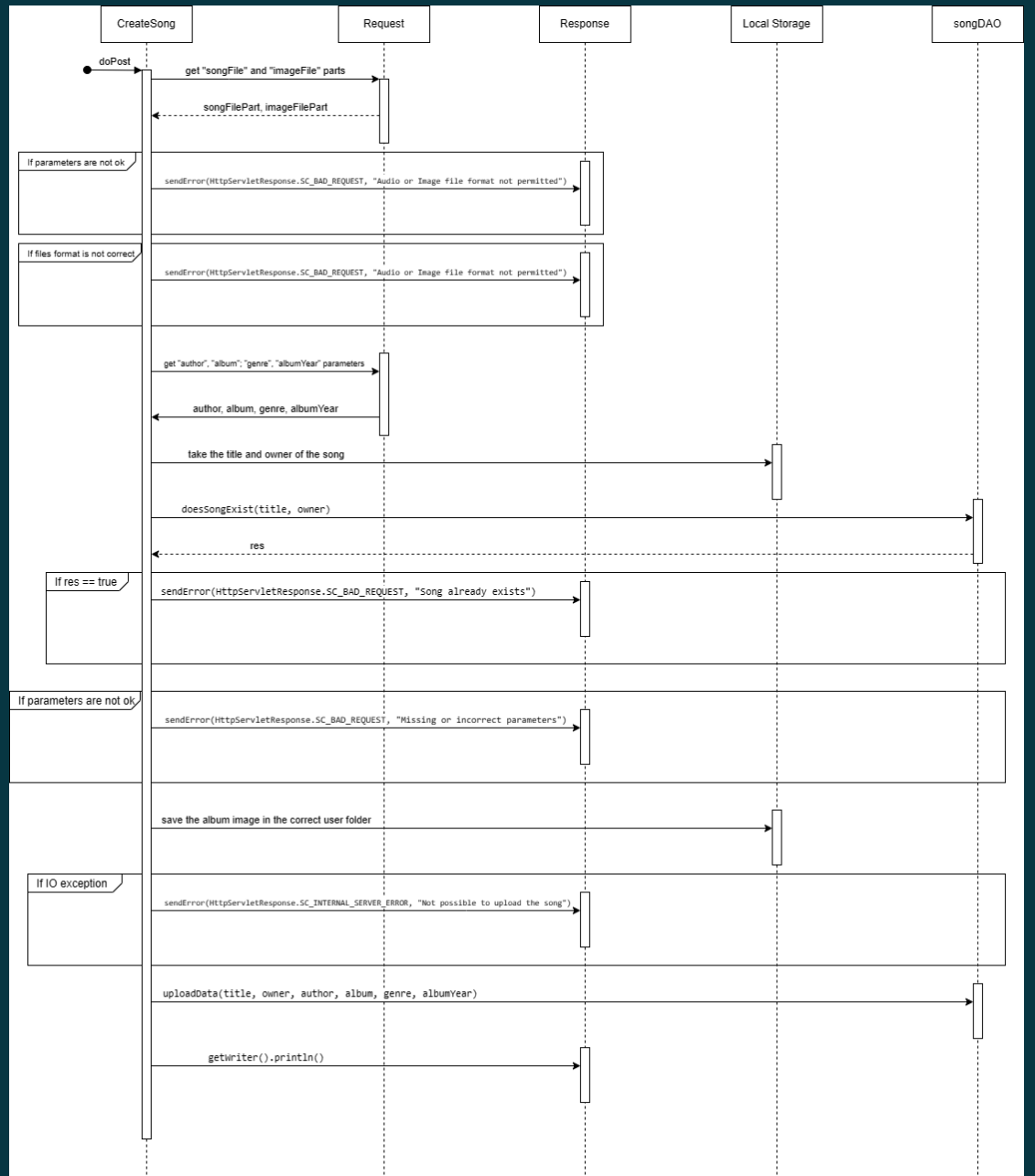
Event: Cover Servlet



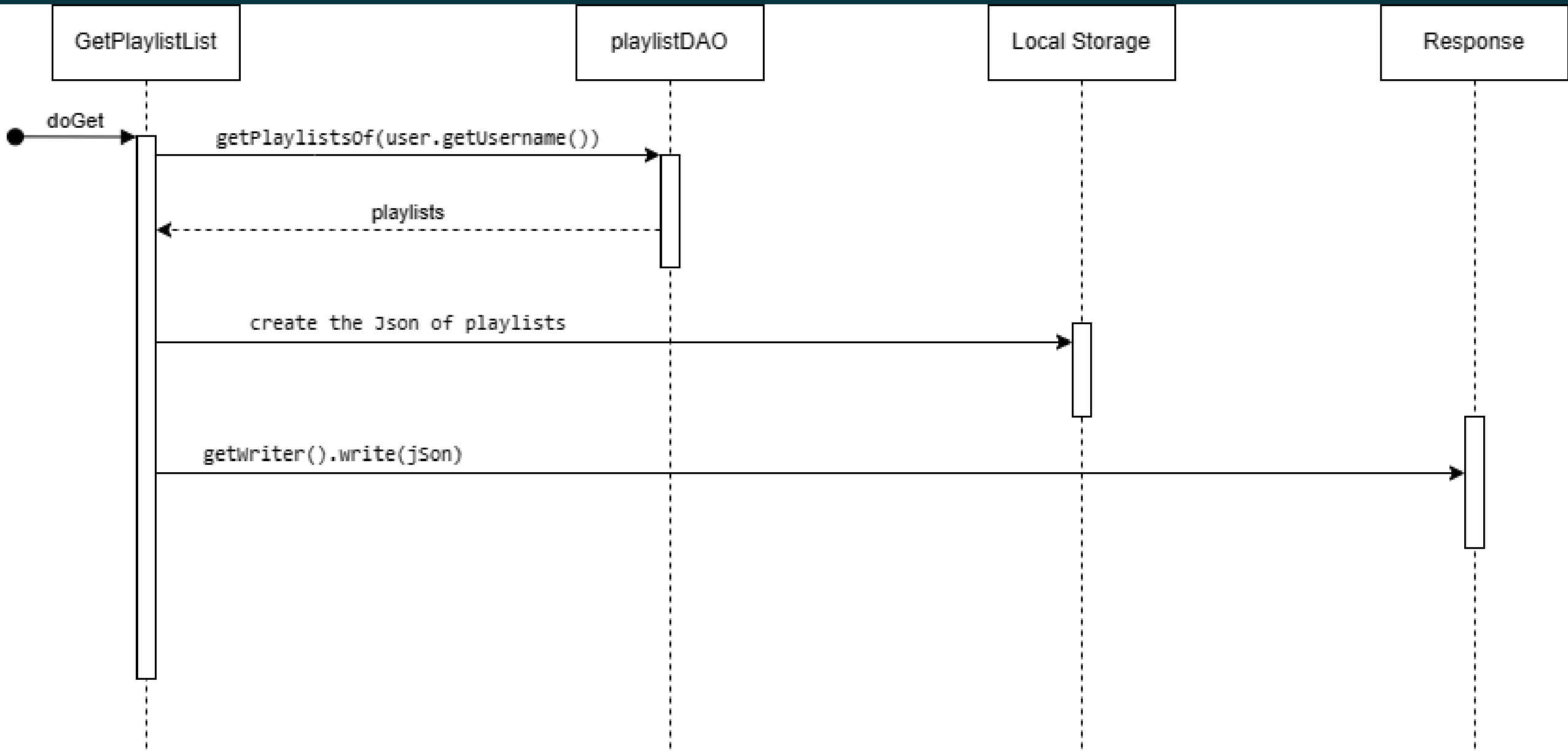
Event: Create



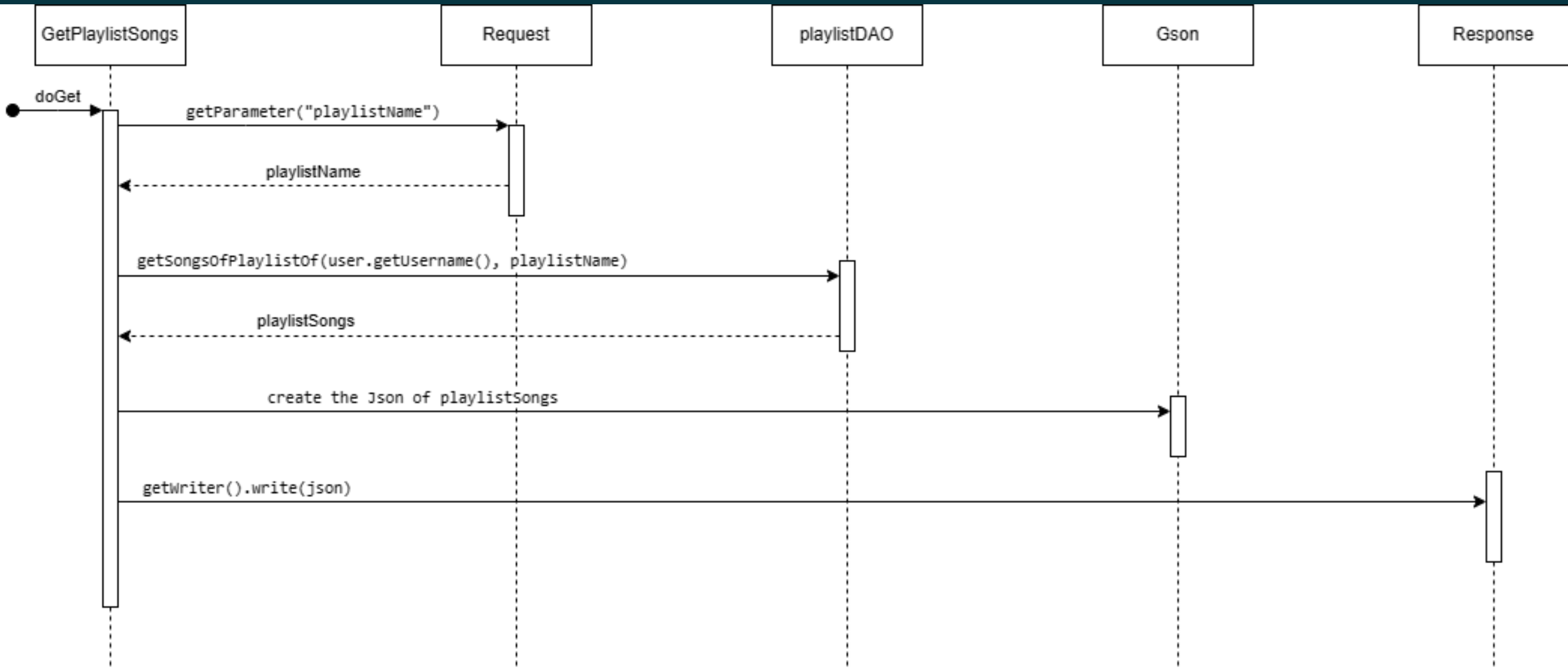
Event: Create Song



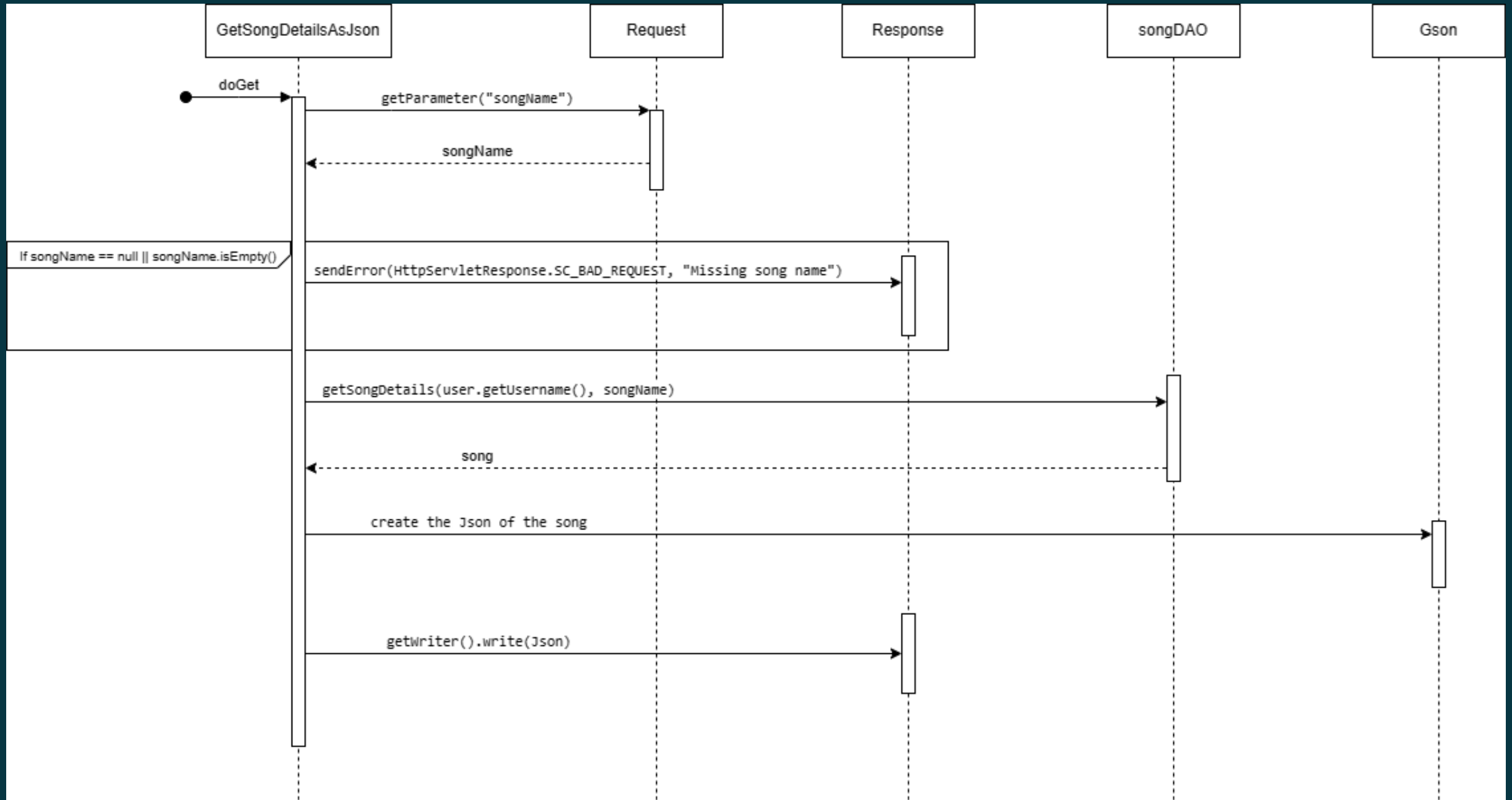
Event: Get Playlist List



Event: Get Playlist Songs



Event: Get Song Details As Json



Event: Get Song List