

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](#).

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.

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

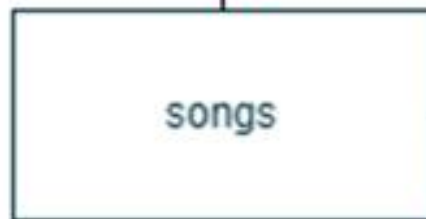
playlistOwner
playlistSong
playlistName
albumYear
creationDate
songIndexJs



1:N



0:N



title
owner
author
album
genre
albumYear

1



0:N



username
password
nome
cognome

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`))
```

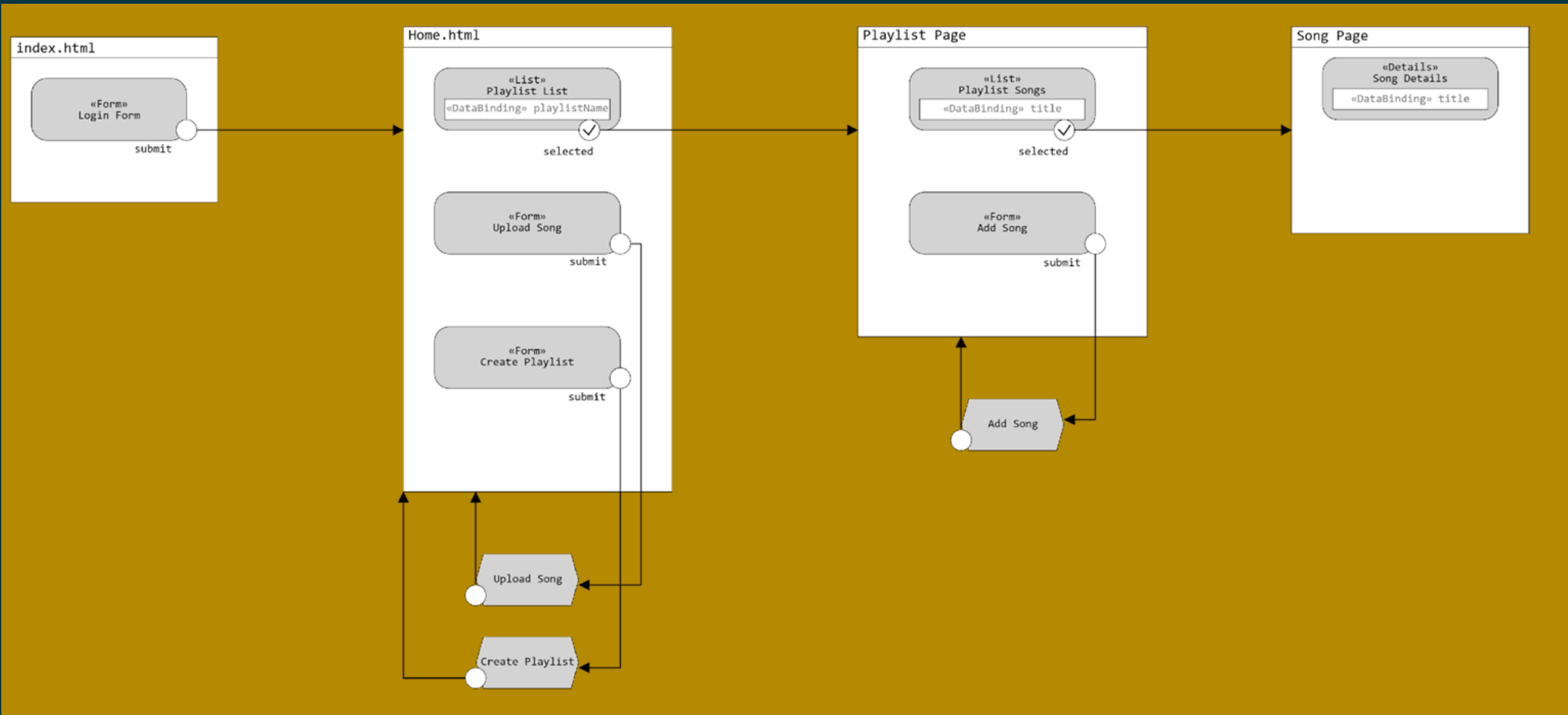
playlist:

```
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
- PlayerPage.html
- PlaylistDetails.html

Utils

- ConnectionHandler.java

Filters

- LoginChecker.java

Controllers (Servlets)

- AddSong.java
- AudioPlayer.java
- CheckLogin.java
- CoverServlet.java
- CreatePlaylist.java
- CreateSong.java
- GetPlayerPage.java
- GetPlaylistDetails.java
- GoToHomePage.java

Components: Data Access Objects (Classes)

PlaylistDAO

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

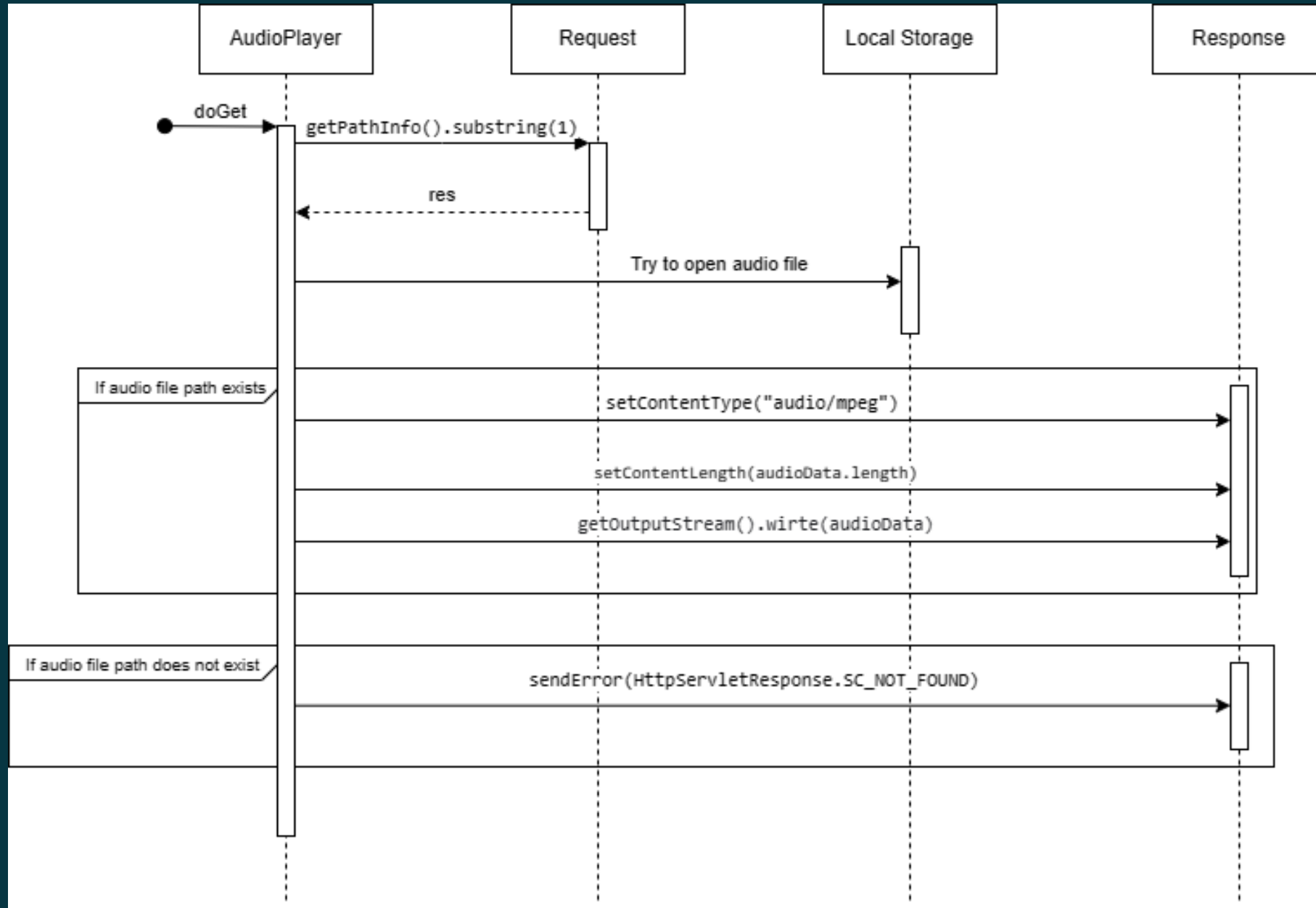
SongDAO

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

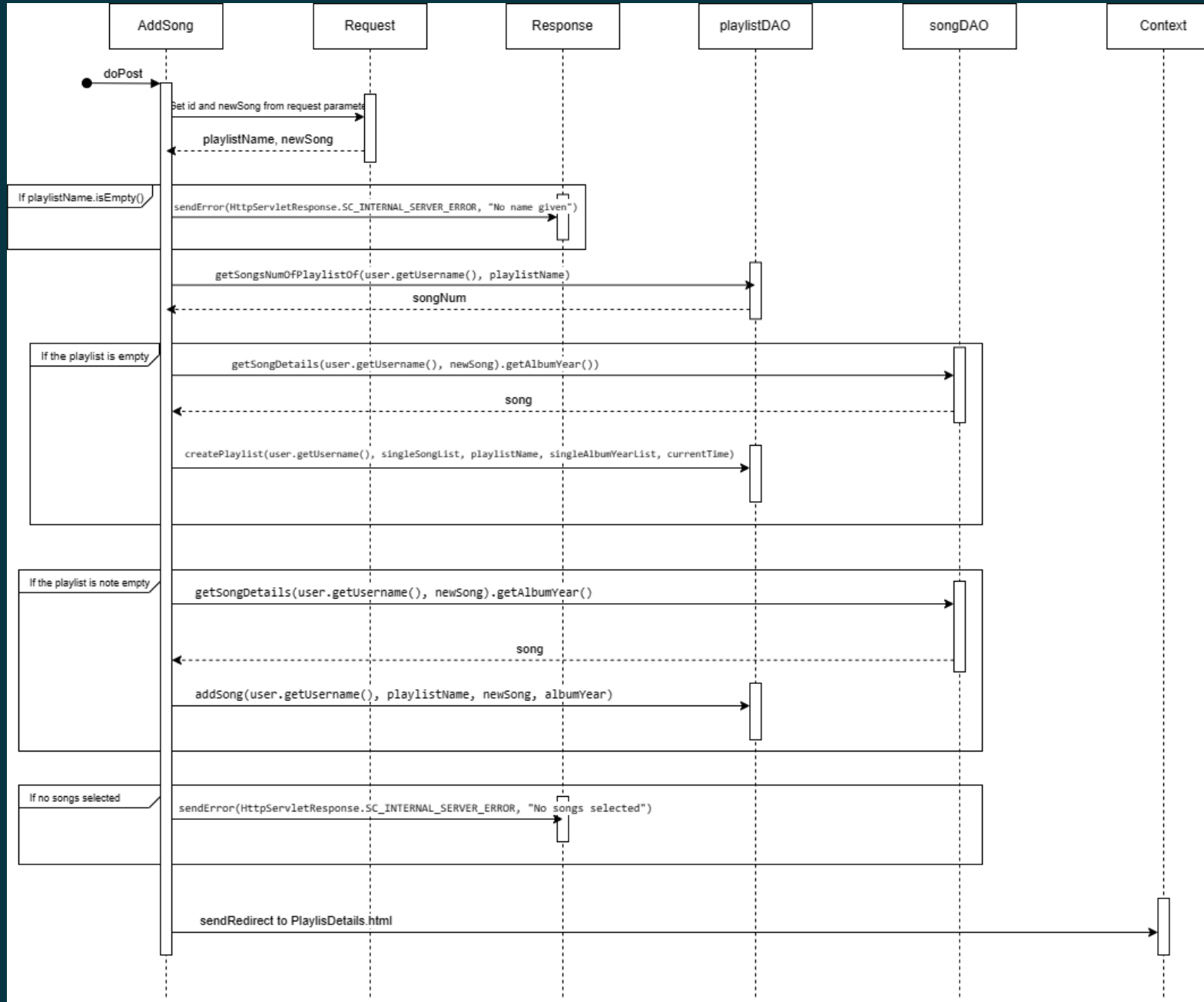
UserDAO

- checkCredentials (usrn, pwd)

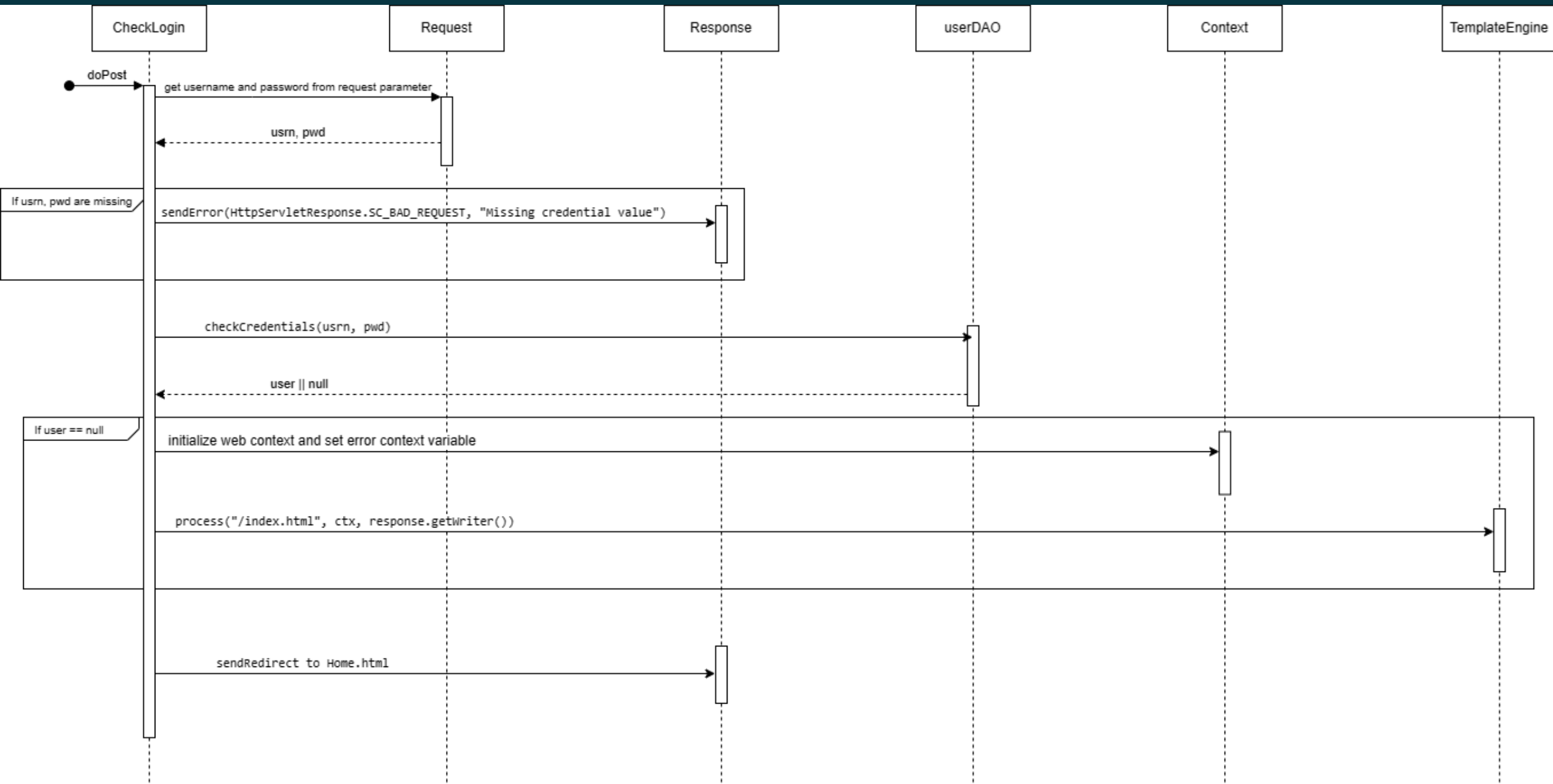
Event: Audio Player



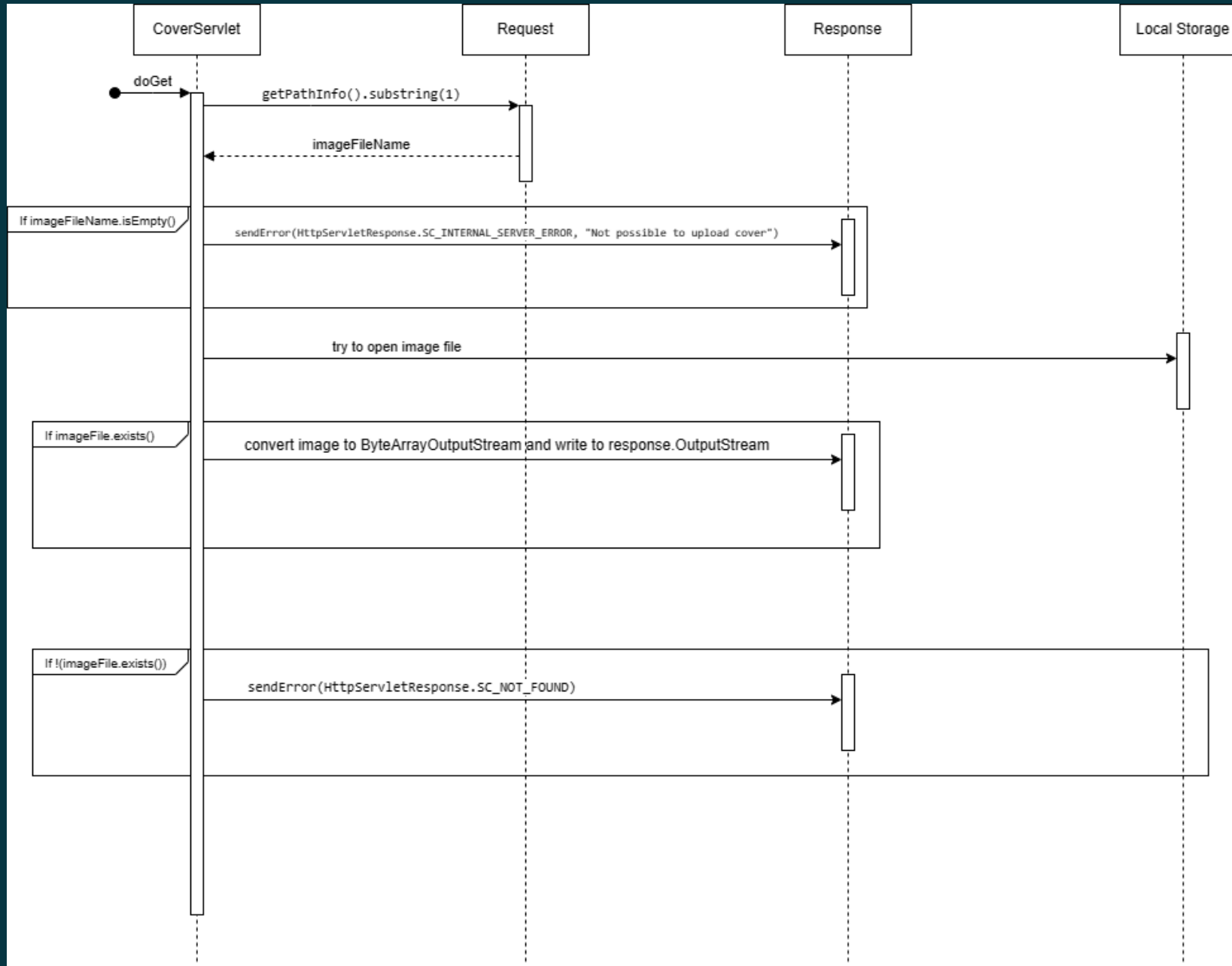
Event: Add Song



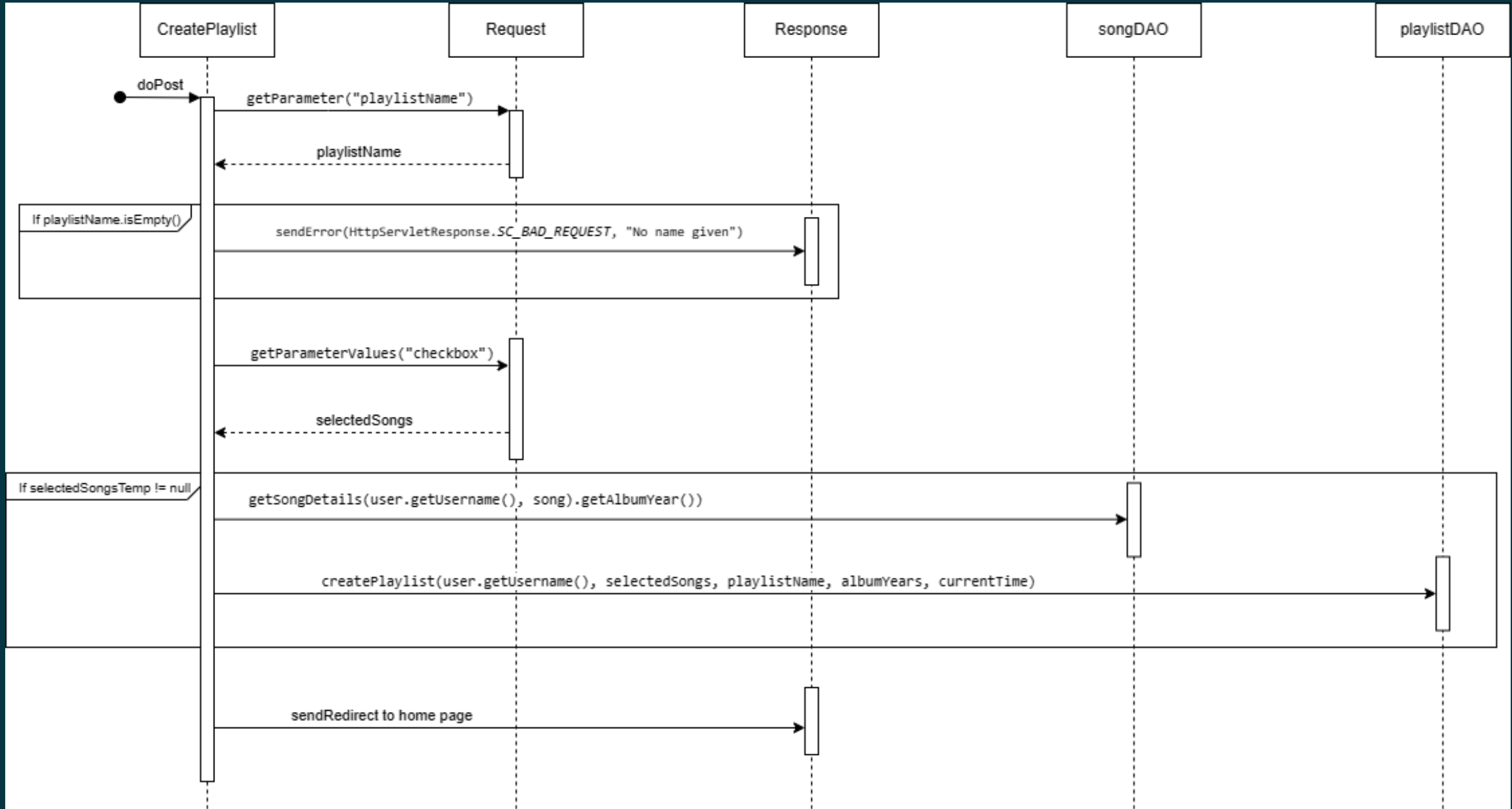
Event: Check Login



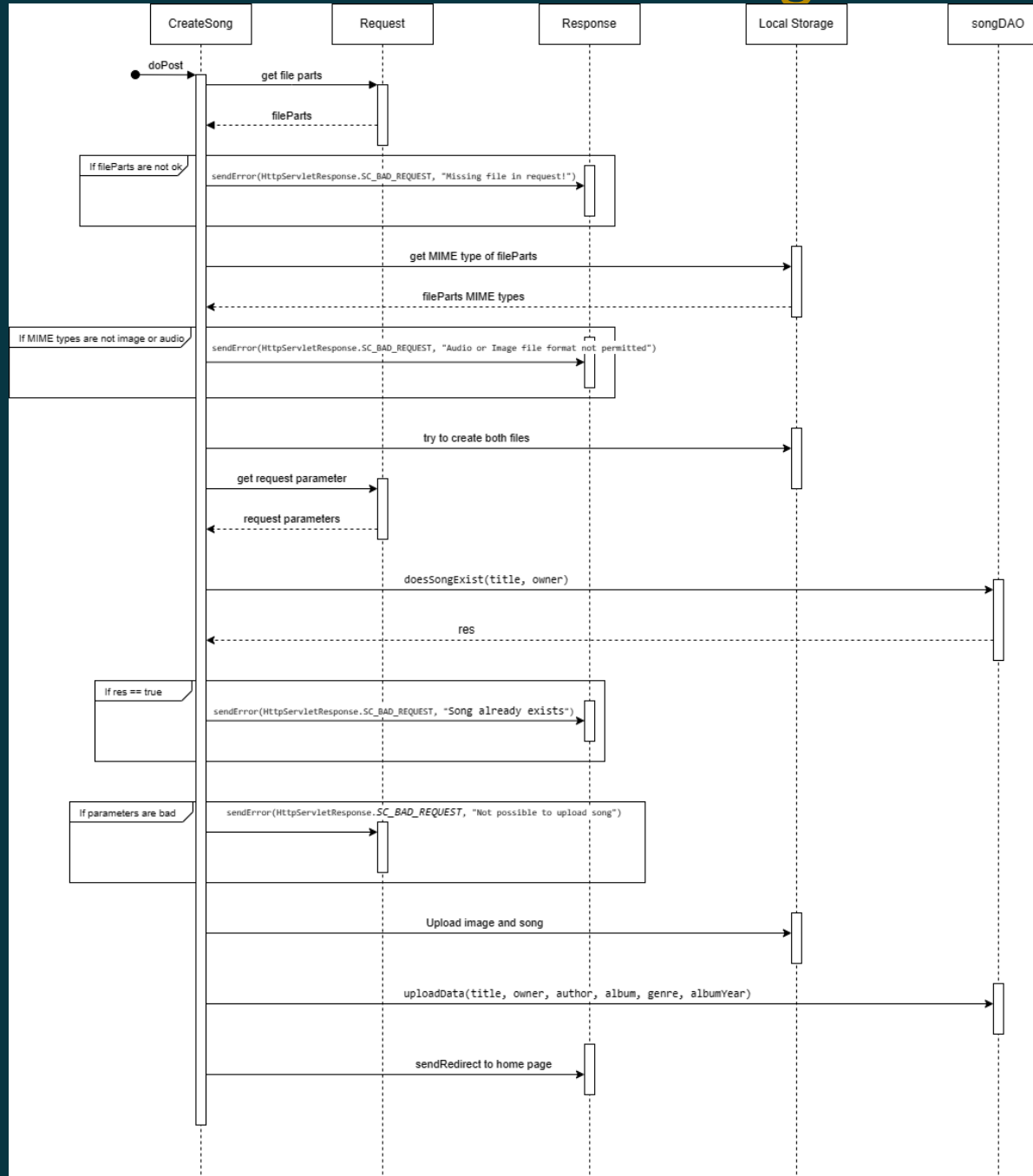
Event: Cover Servlet



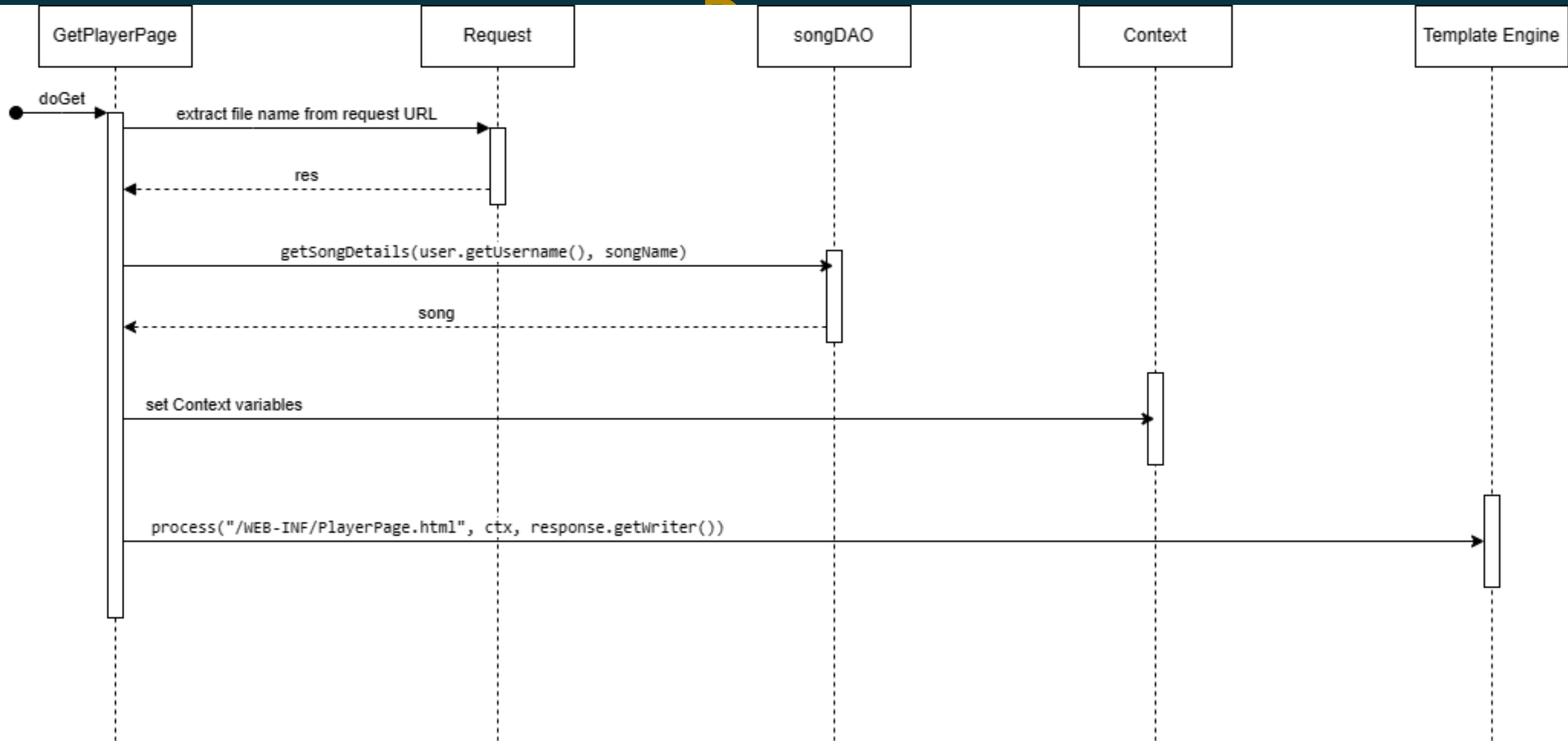
Event: Create



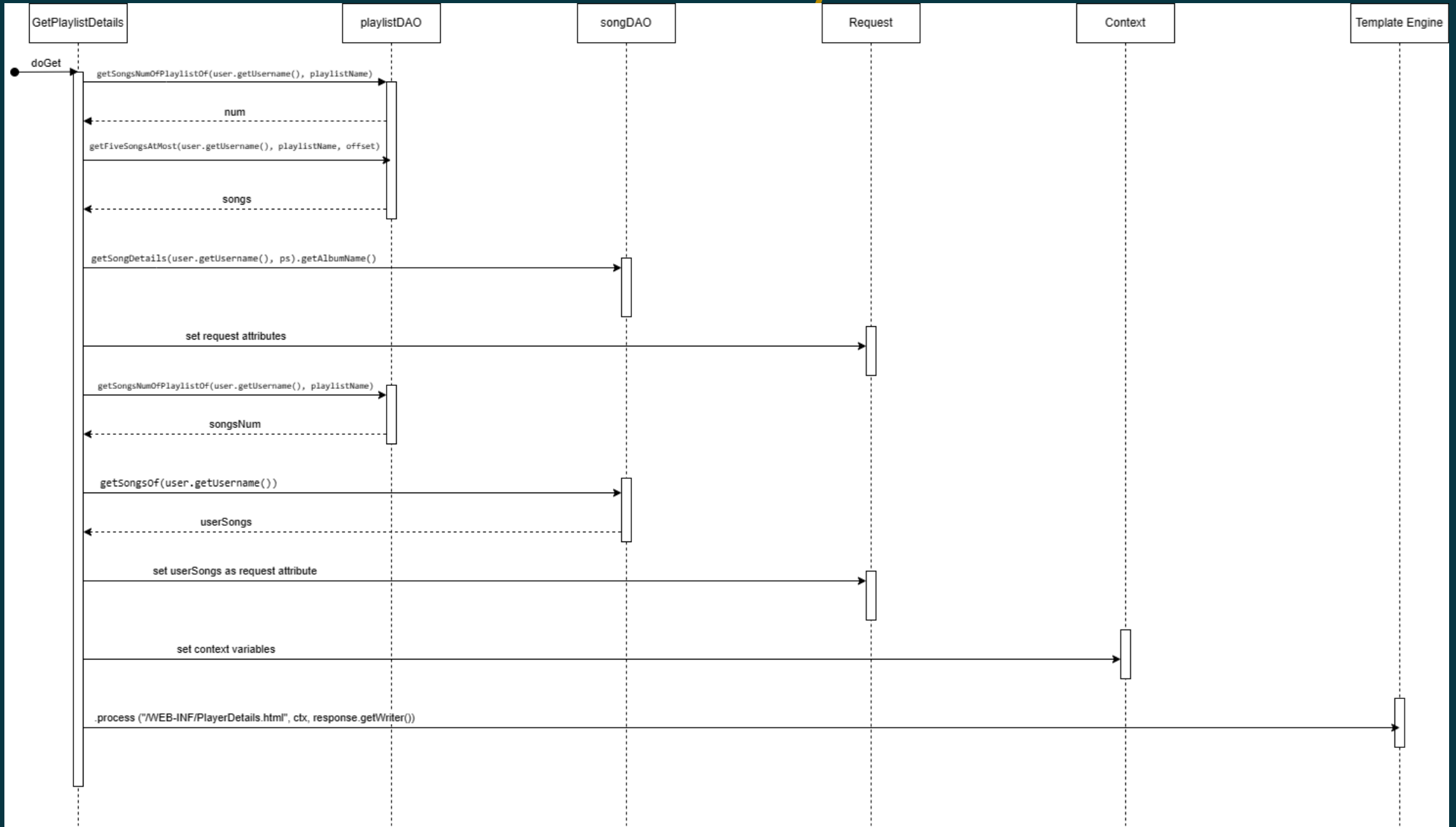
Event: Create Song



Event: Get Player



Event: Get Playlist Details



Event: Go To Home

