Progetto TIW

Andrea Pazienza

CP 10716103

Matricola 957239

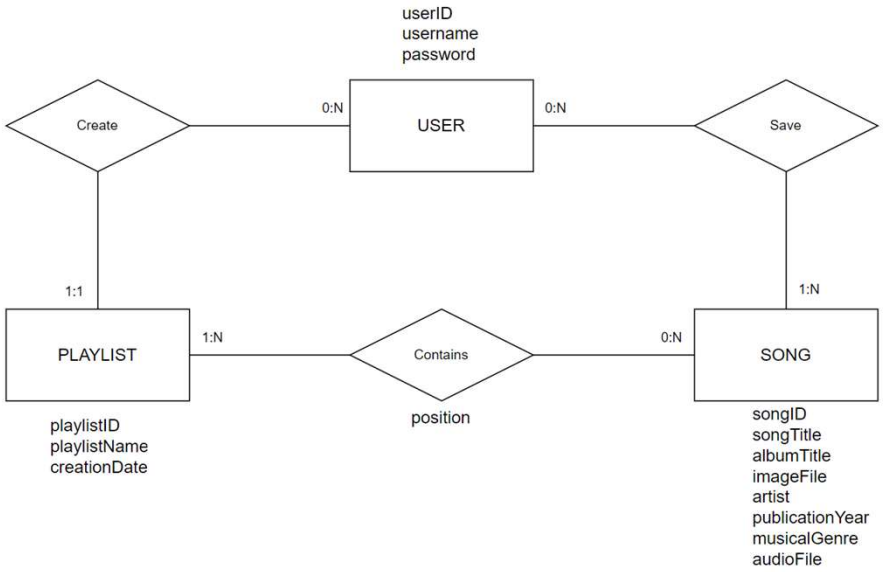
Playlist Musicale

Versione con JavaScript

Data 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, 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.
- Entities, attributes, relationships

Database design



User(<u>uerID</u>, username, password)

Playlist(<u>playlistID</u>, playlistName, creationDate, userID)

Song(<u>songID</u>, songTitle, albumTitle, imageFile, artist, publicationYear, musicalGenre, audioFile)

UserSong(<u>userID</u>, <u>songID</u>)

PlaylistSong(<u>playlistID</u>, <u>songID,</u> position)

Playlist.userID → User.userID

UserSong.userID → User.userID

UserSong.songID → Song.songID

PlaylistSong.playlistID → Playlist.playlistID

PlaylistSong.songID → Song.songID

Database tables

```
CREATE TABLE `User` (
   `userID` int NOT NULL AUTO_INCREMENT,
   `username` varchar(45) NOT NULL,
   `password` varchar(45) NOT NULL,
   PRIMARY KEY (`userID`)
);

CREATE TABLE `Playlist` (
   `playlistID` int NOT NULL AUTO_INCREMENT,
   `playlistName` varchar(45) NOT NULL,
   `creationDate` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
   `userID` int NOT NULL,
   PRIMARY KEY (`playlistID`),
   KEY `PuserID_idx` (`userID`),
   CONSTRAINT `PuserID` FOREIGN KEY (`userID`) REFERENCES `User` (`userID`)
                    ON DELETE CASCADE ON UPDATE RESTRICT
);

CREATE TABLE `Song` (
   `songID` int NOT NULL AUTO_INCREMENT,
   `songTitle` varchar(45) NOT NULL,
   `albumTitle` varchar(45) NOT NULL,
   `imageFile` longblob NOT NULL,
   `artist` varchar(45) NOT NULL,
   `publicationYear` int NOT NULL,
   `musicalGenre` varchar(45) NOT NULL,
   `audioFile` longblob NOT NULL,
   PRIMARY KEY (`songID`)
);
```

```
CREATE TABLE `UserSong` (
   `userID` int NOT NULL,
   `songID` int NOT NULL,
   PRIMARY KEY (`userID`, `songID`),
   KEY `USsongID_idx` (`songID`),
   CONSTRAINT `USsongID` FOREIGN KEY (`songID`) REFERENCES `Song` (`songID`)
                    ON DELETE CASCADE ON UPDATE RESTRICT,
   CONSTRAINT `USuserID` FOREIGN KEY (`userID`) REFERENCES `User` (`userID`)
                    ON DELETE CASCADE ON UPDATE RESTRICT
);

CREATE TABLE `PlaylistSong` (
   `playlistID` int NOT NULL,
   `songID` int NOT NULL,
   `position` int NOT NULL,
   PRIMARY KEY (`playlistID`, `songID`),
   KEY `PSsongID_idx` (`songID`),
   CONSTRAINT `PSplaylistID` FOREIGN KEY (`playlistID`) REFERENCES `Playlist`
(`playlistID`)
                    ON DELETE CASCADE ON UPDATE RESTRICT,
   CONSTRAINT `PSsongID` FOREIGN KEY (`songID`) REFERENCES `Song` (`songID`)
                    ON DELETE CASCADE ON UPDATE RESTRICT
);
```

Data requirements analysis

- 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. 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 (views), view components, events, actions

Data requirements analysis

- 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. 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 (views), view components, events, actions

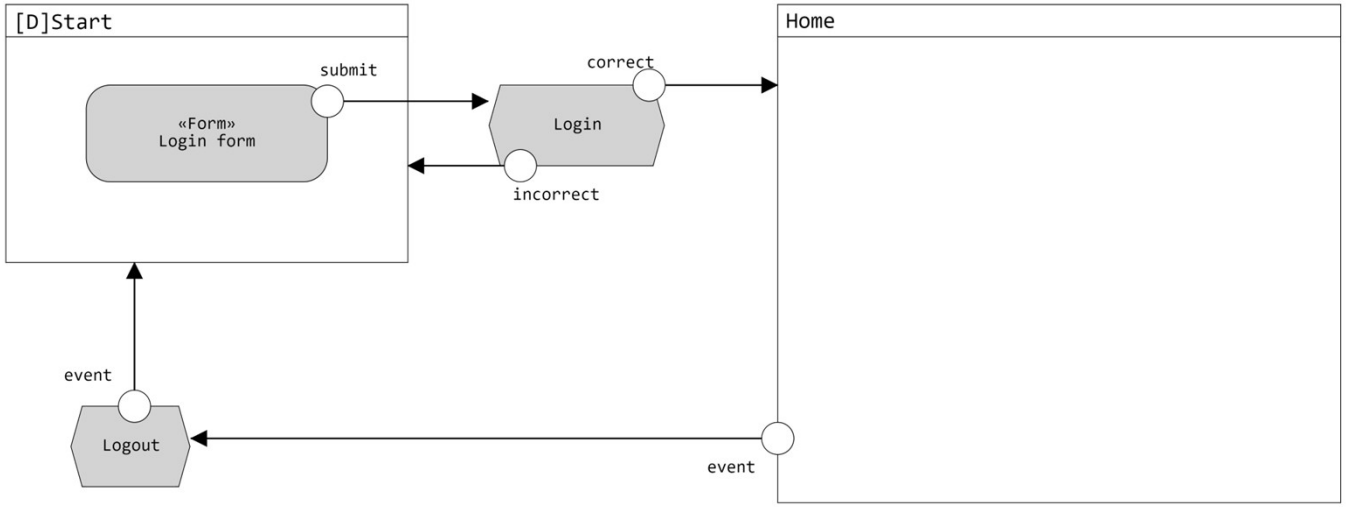
Data requirements analysis

- 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. 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 (views), view components, events, actions

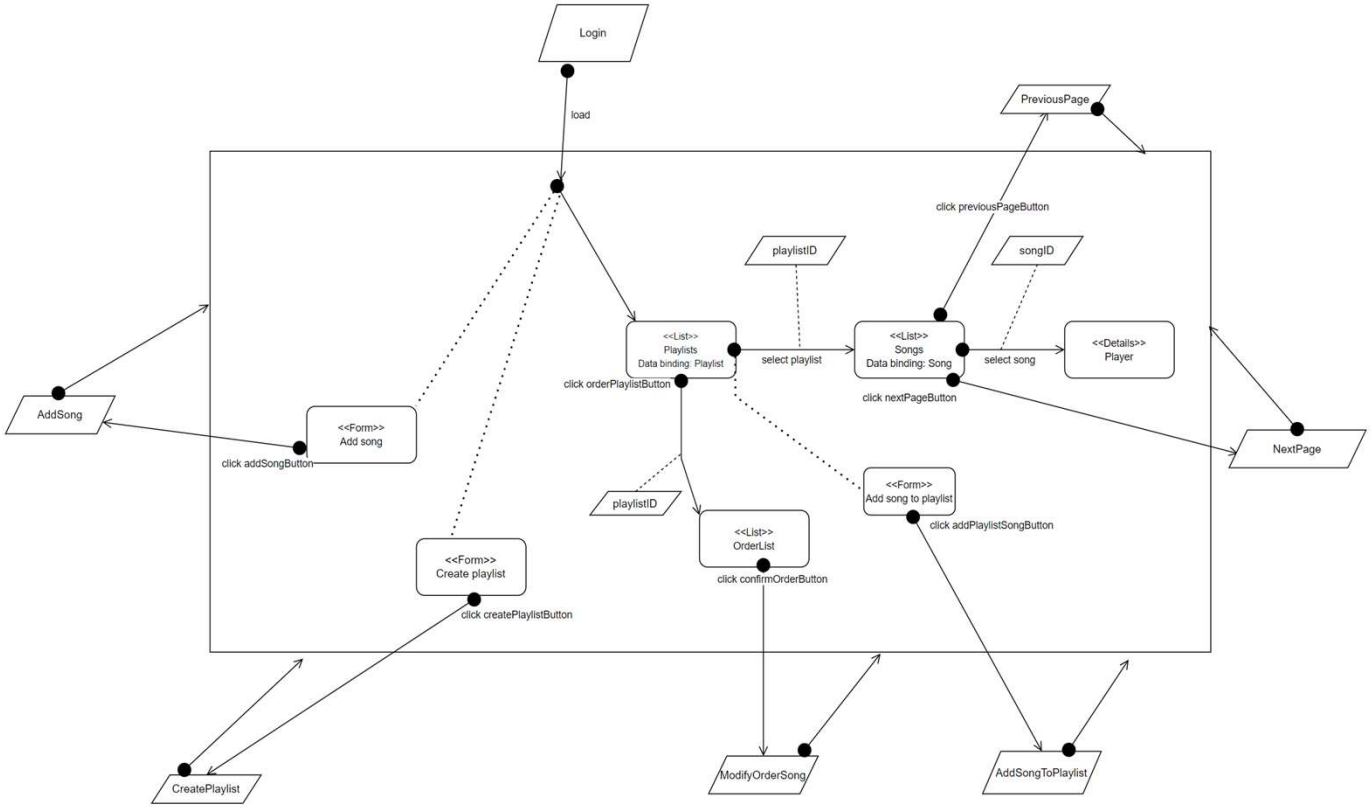
Data requirements analysis

- Dopo il login dell'utente, l'intera applicazione è realizzata con un'unica pagina. Ogni interazione dell'utente è gestita senza ricaricare completamente la pagina, ma produce l'invocazione asincrona del server e l'eventuale modifica del contenuto da aggiornare a seguito dell'evento. L'evento di visualizzazione del blocco precedente/successivo è gestito a lato client senza generare una richiesta al server. L'applicazione deve consentire all'utente di riordinare le playlist con un criterio diverso da quello di default (data decrescente). Dalla HOME con un link associato a ogni playlist page si accede a una pagina RIORDINO, che mostra la lista completa dei brani della playlist e permette all'utente di trascinare il titolo di un brano nell'elenco e di collocarlo in una posizione diversa per realizzare l'ordinamento che desidera, senza invocare il server. Quando l'utente ha raggiunto l'ordinamento desiderato, usa un bottone "salva ordinamento", per memorizzare la sequenza sul server. Ai successivi accessi, l'ordinamento personalizzato è usato al posto di quello di default.
- Pages (views), view components, events, actions

Application design (in IFML)



Focus home page (pseudo-IFML)



Load → visibleHomePage()
Select playlist → visiblePlaylistPage()
Select song → visiblePlayerPage()

(Each function shows the components of one of the views and hides the components of the other two)

orderPlaylistButton.click → OrderList visible

Components: server side

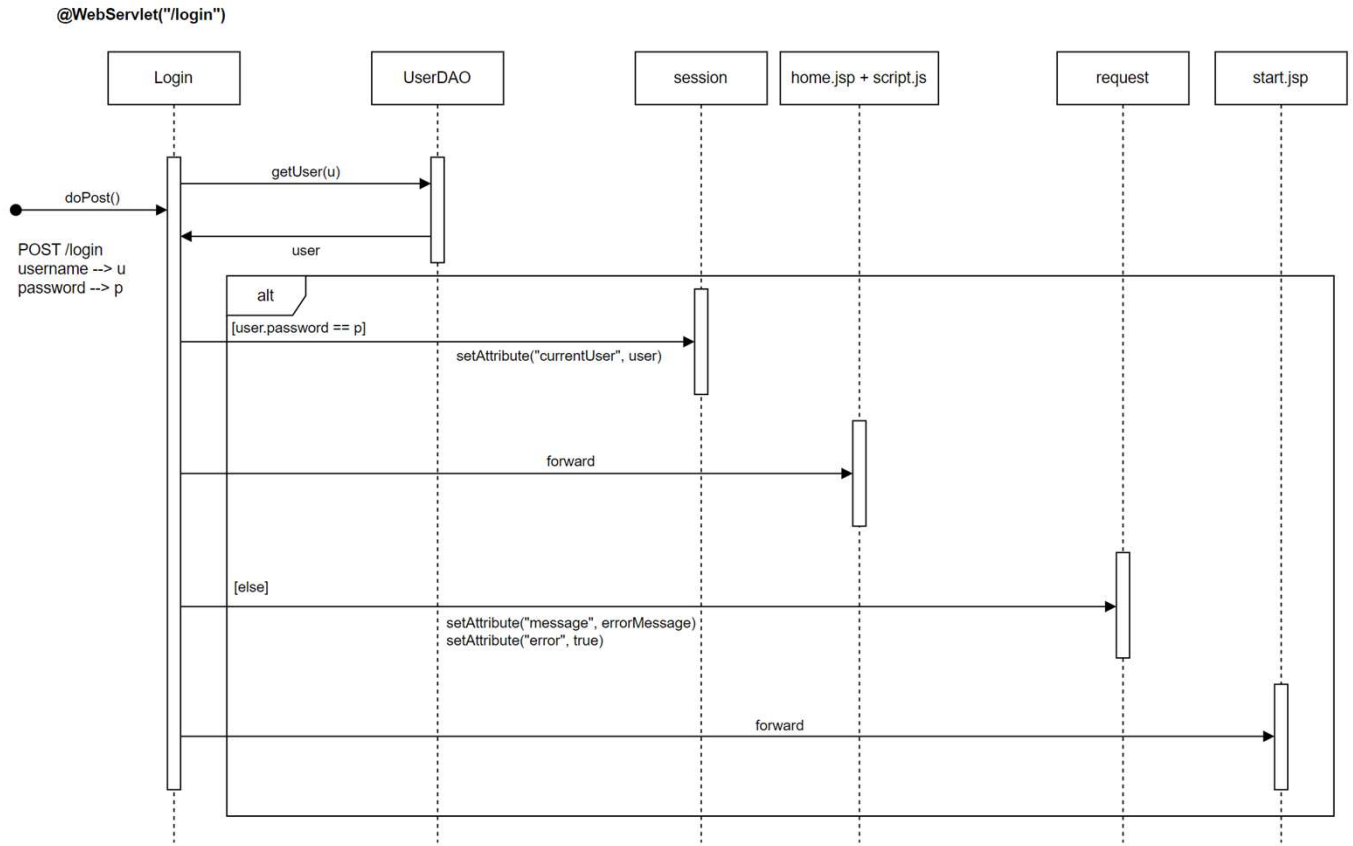
- Model objects (Beans)
  - User
  - Playlist
  - Song
- Data Access Objects (Classes)
  - UserDAO
    - getUser
  - PlaylistDAO
    - getPlaylistList
    - getPlaylist
    - createPlaylist
    - addSongToPlaylist
    - modifyPlaylistOrder
  - SongDAO
    - getAllSongs
    - getPlaylistSongs
    - getNotPlaylistSongs
    - getSong
    - addSong
    - addNewSong
    - addUserSong
    - searchUserSong
    - searchSong

- Controllers (servlets)
  - Login
  - GetPlaylistList
  - GetAllSongs
  - CreatePlaylist
  - AddSong
  - ModifyPlaylistOrder
  - GetPlaylistSongs
  - GetNotPlaylistSongs
  - PreviousPage
  - NextPage
  - AddSongToPlaylist
  - GetPlayer
  - Logout
- The database connection is created by controllers in the `init()` method and passed to the DAO

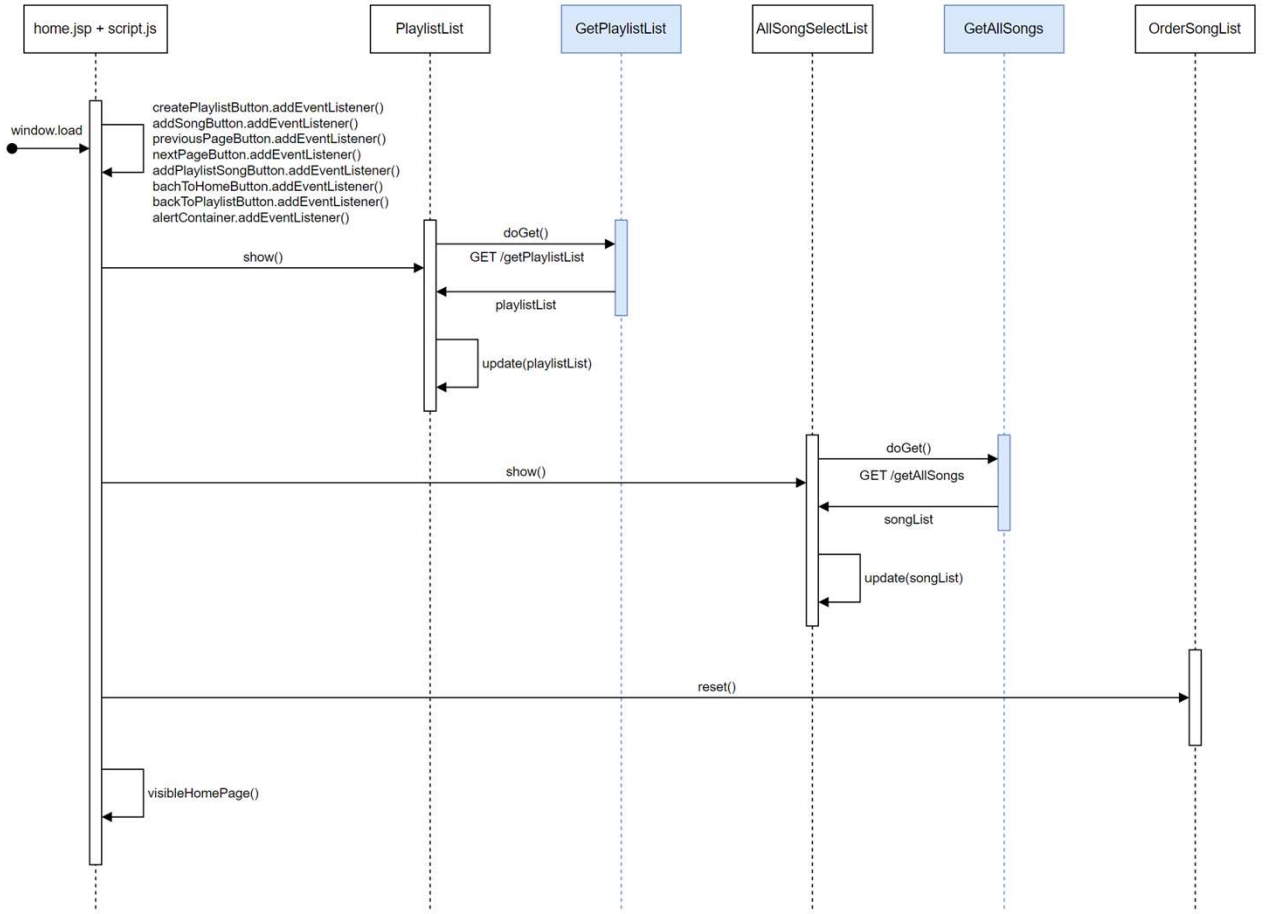
Components: client side

- View components
  - PlaylistList
    - show(): gets the content
    - update(arrayPlaylist): creates the HTML markup (and the binding with the playlist) and the orderPlaylistButton
  - OrderSongList
    - show(playlistID, playlistName): gets the content
    - update(playlistID, playlistName): creates the HTML markup, sets the drag and drop function and creates the confirmOrderButton
    - reset()
  - AllSongSelectList
    - show(): gets the content
    - update(arraySong): creates the HTML markup
  - PlaylistSongTable
    - show(playlistID, playlistName): gets the content
    - update(): creates the HTML markup (and the binding with the player)
    - reset()
    - showPreviousPage(): shows the next five (or less) songs of the playlist
    - showNextPage(): shows the previous five songs of the playlist
  - PlaylistSongSelectList
    - show(): gets the content
    - update(arraySong): creates the HTML markup
  - orderPlaylistButton
  - confirmOrderButton
  - createPlaylistForm
    - createPlaylistButton
  - addSongForm
    - addSongButton
  - previousPageButton
  - nextPageButton
  - addPlaylistSongForm
    - addPlaylistSongButton
  - alertContainer

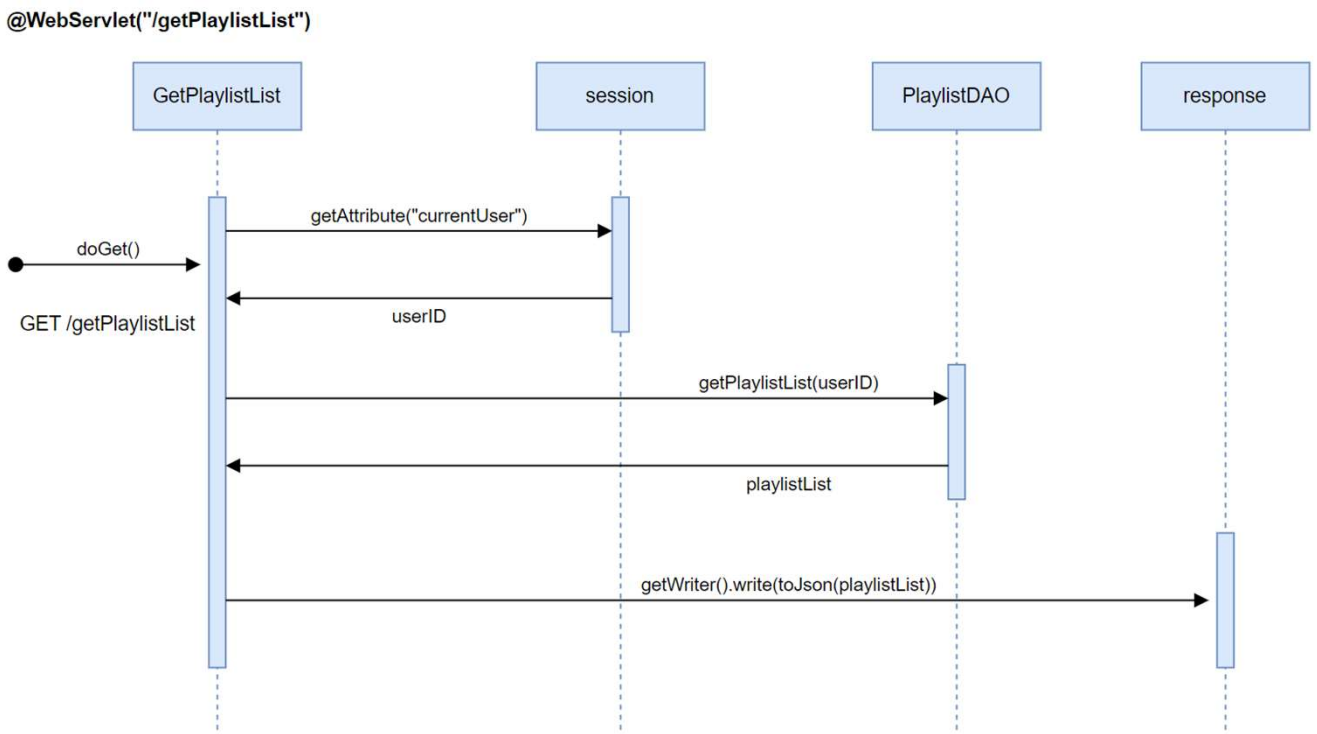
Events: login



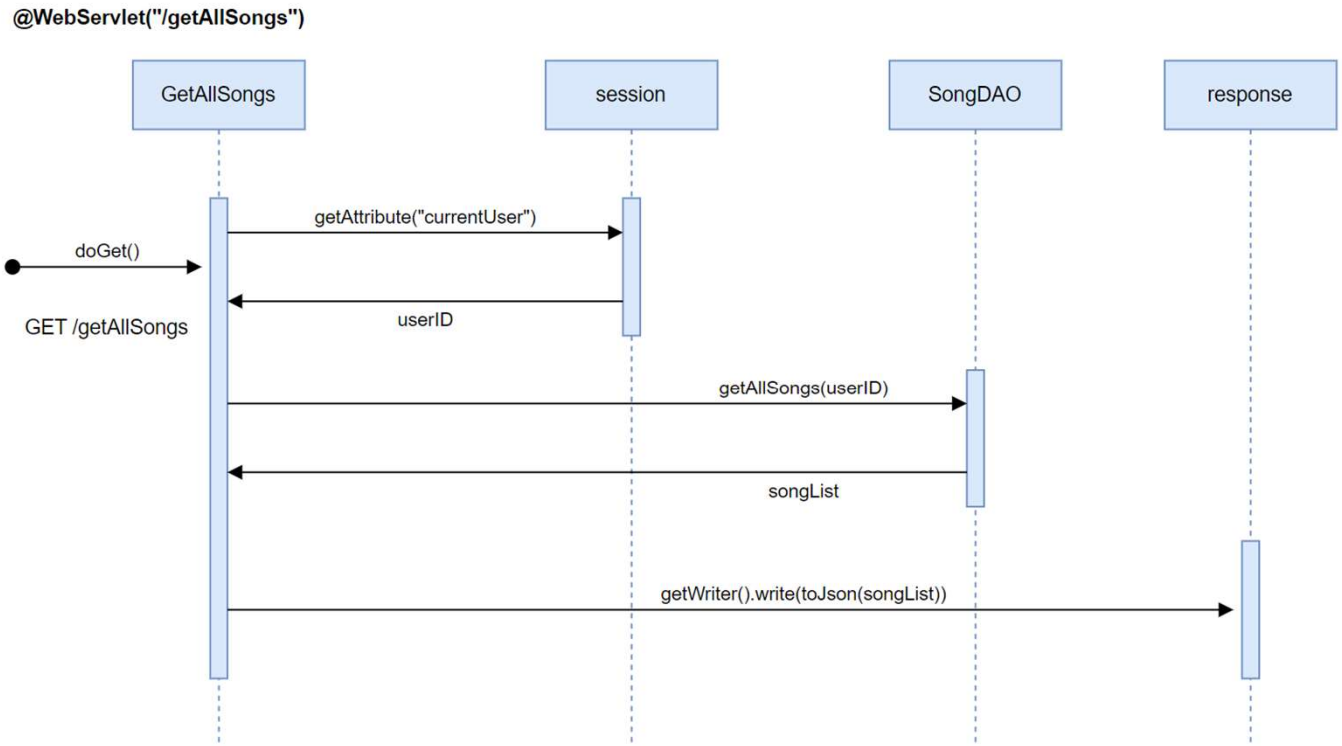
Events: loading home page (client side)



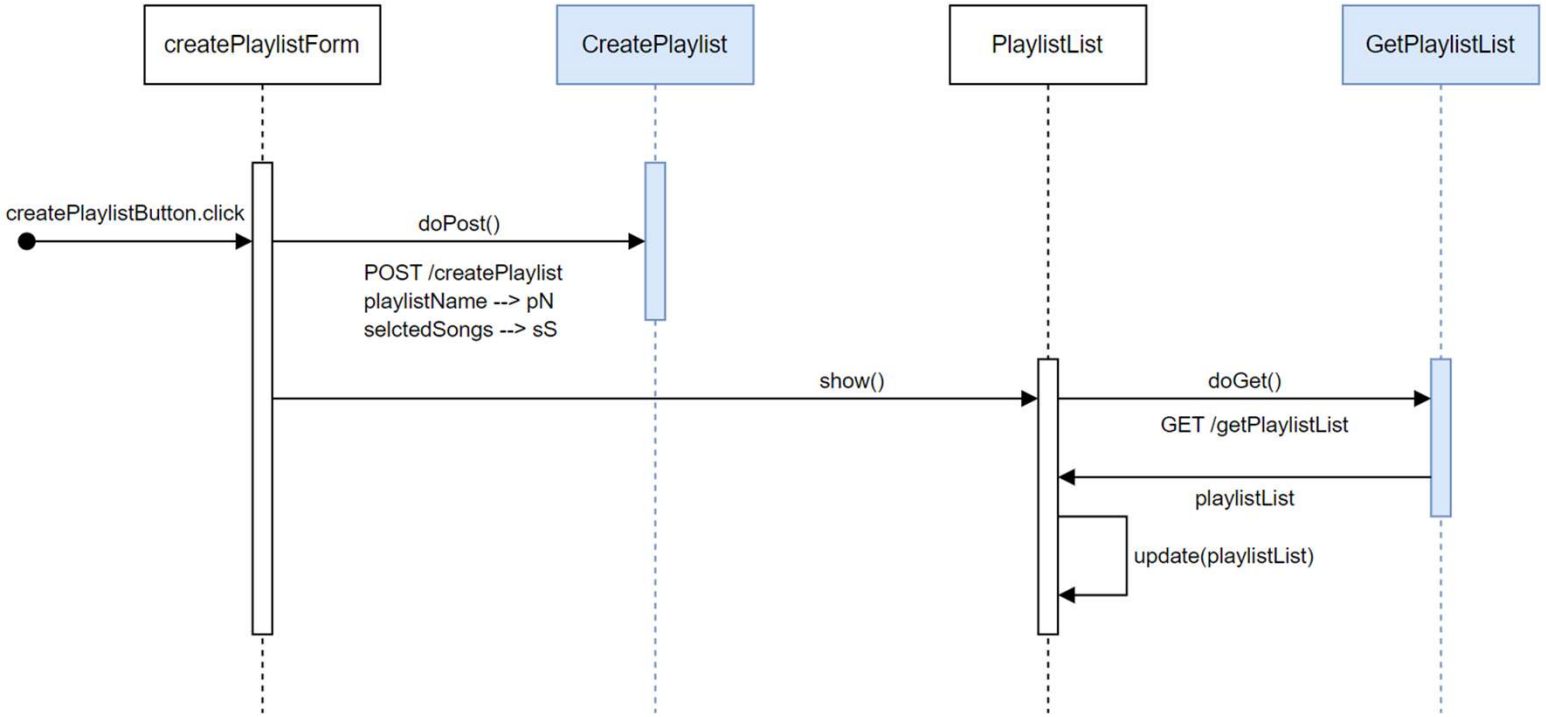
Events: getPlaylistList (server side)



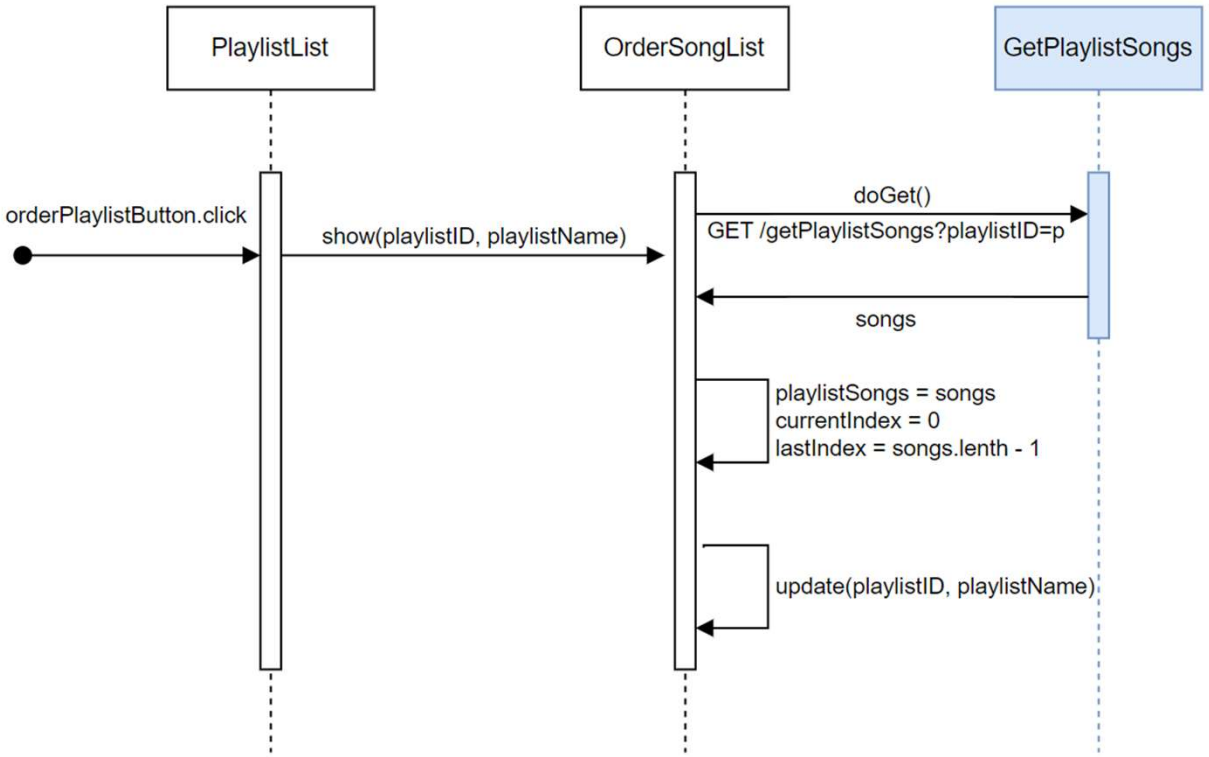
Events: getAllSongs (server side)



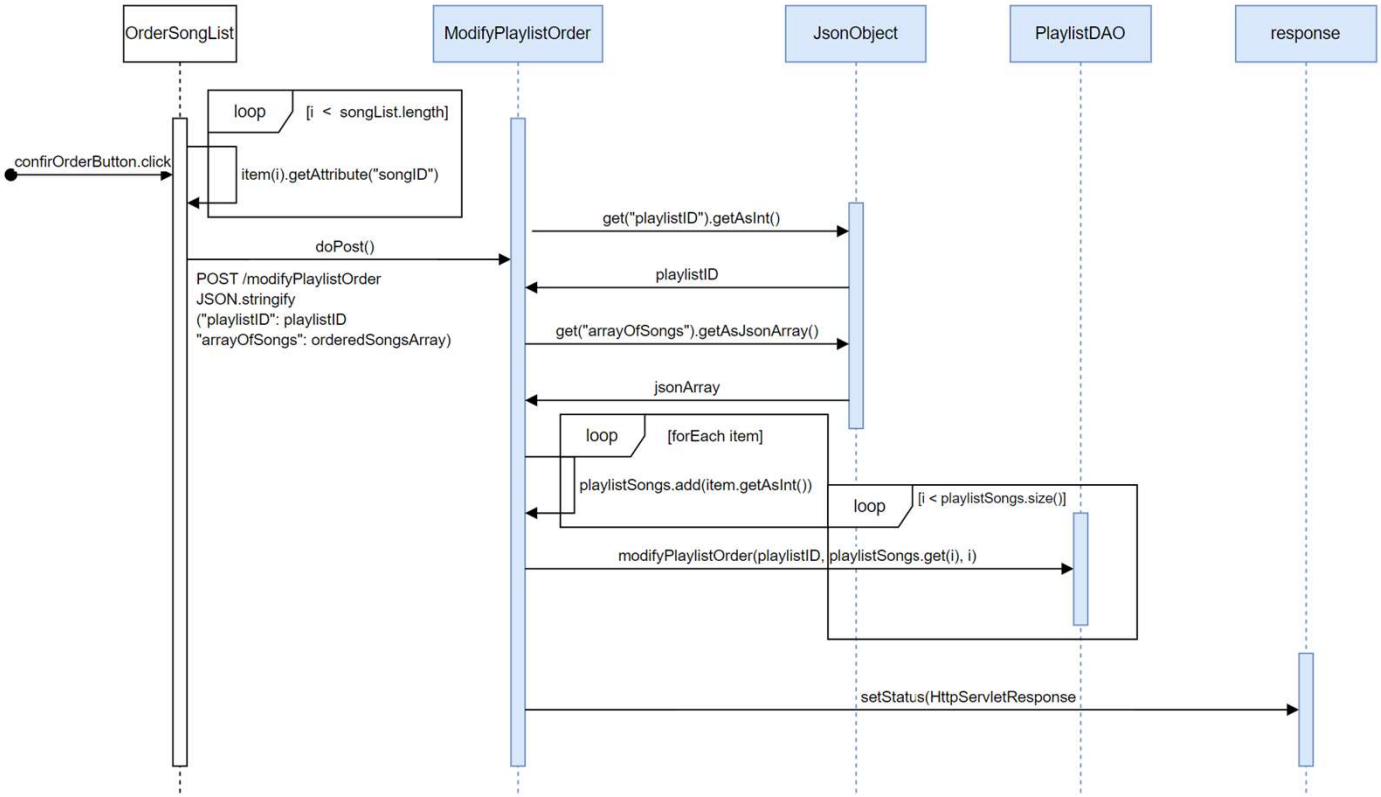
Events: createPlaylist (client side)



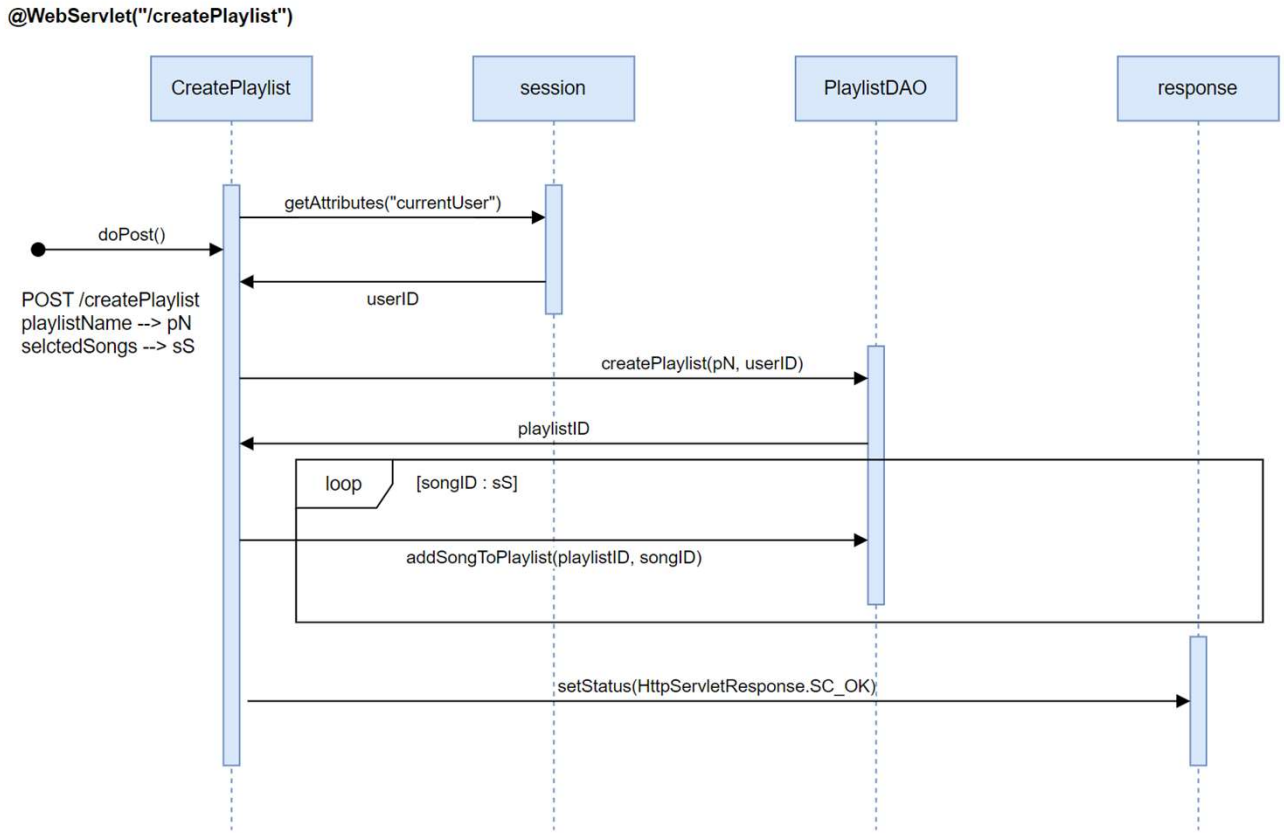
Event: orderPlaylist (client side)



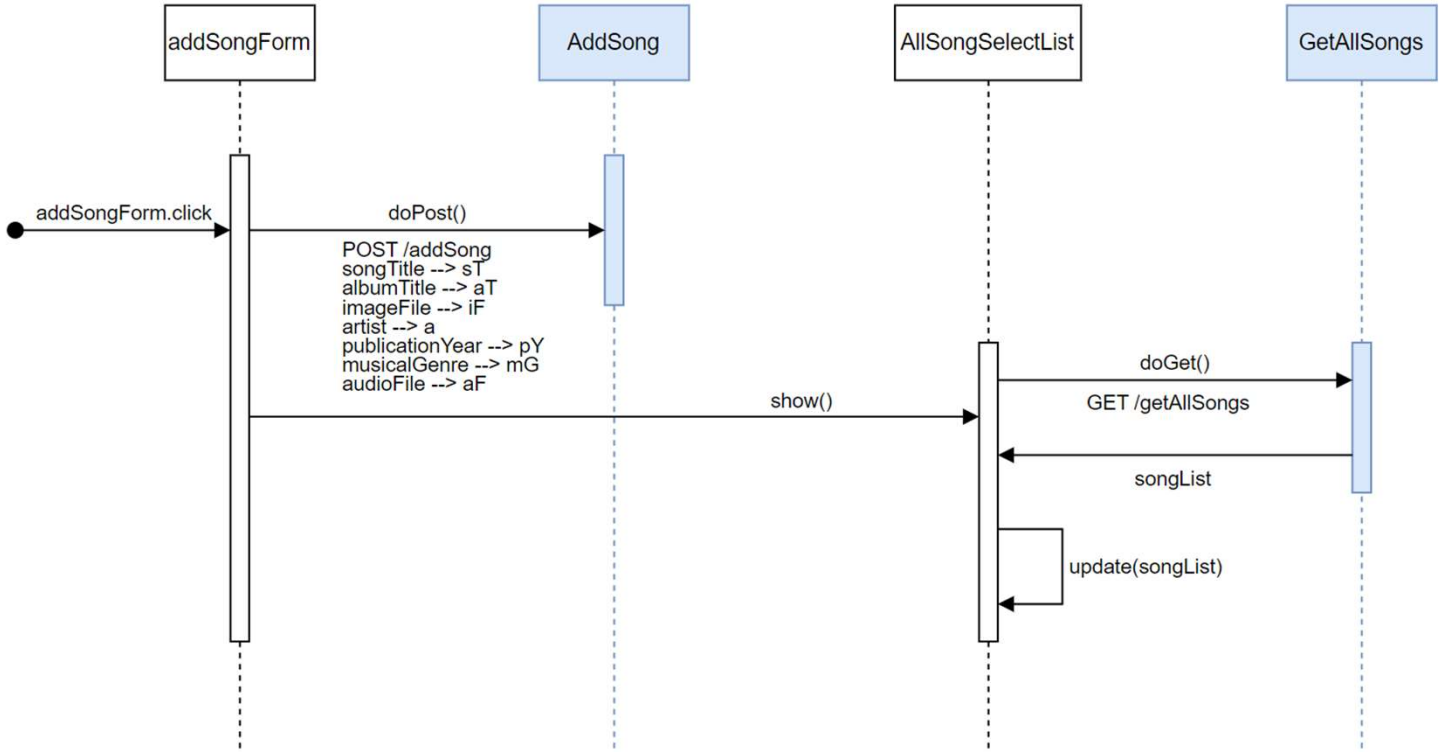
Event: confirmOrder



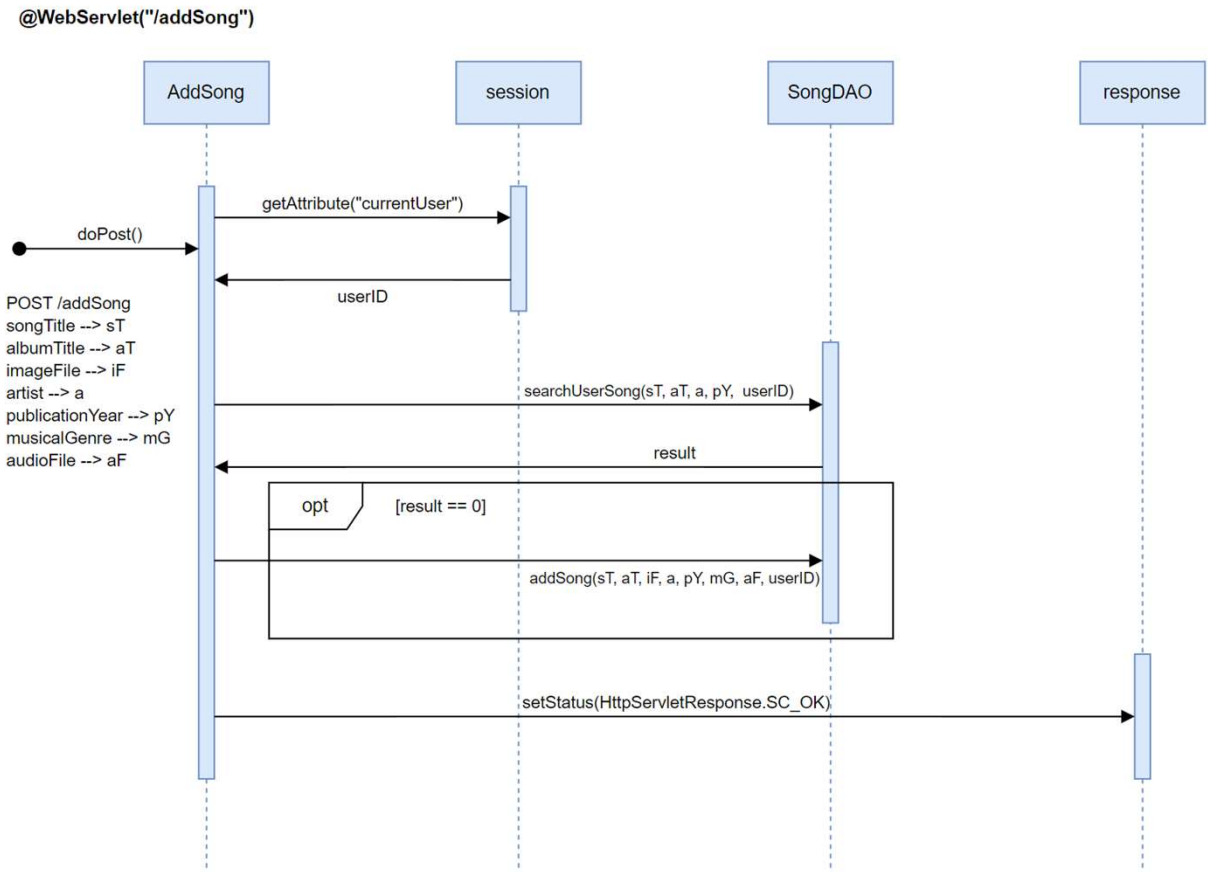
Events: createPlaylist (server side)



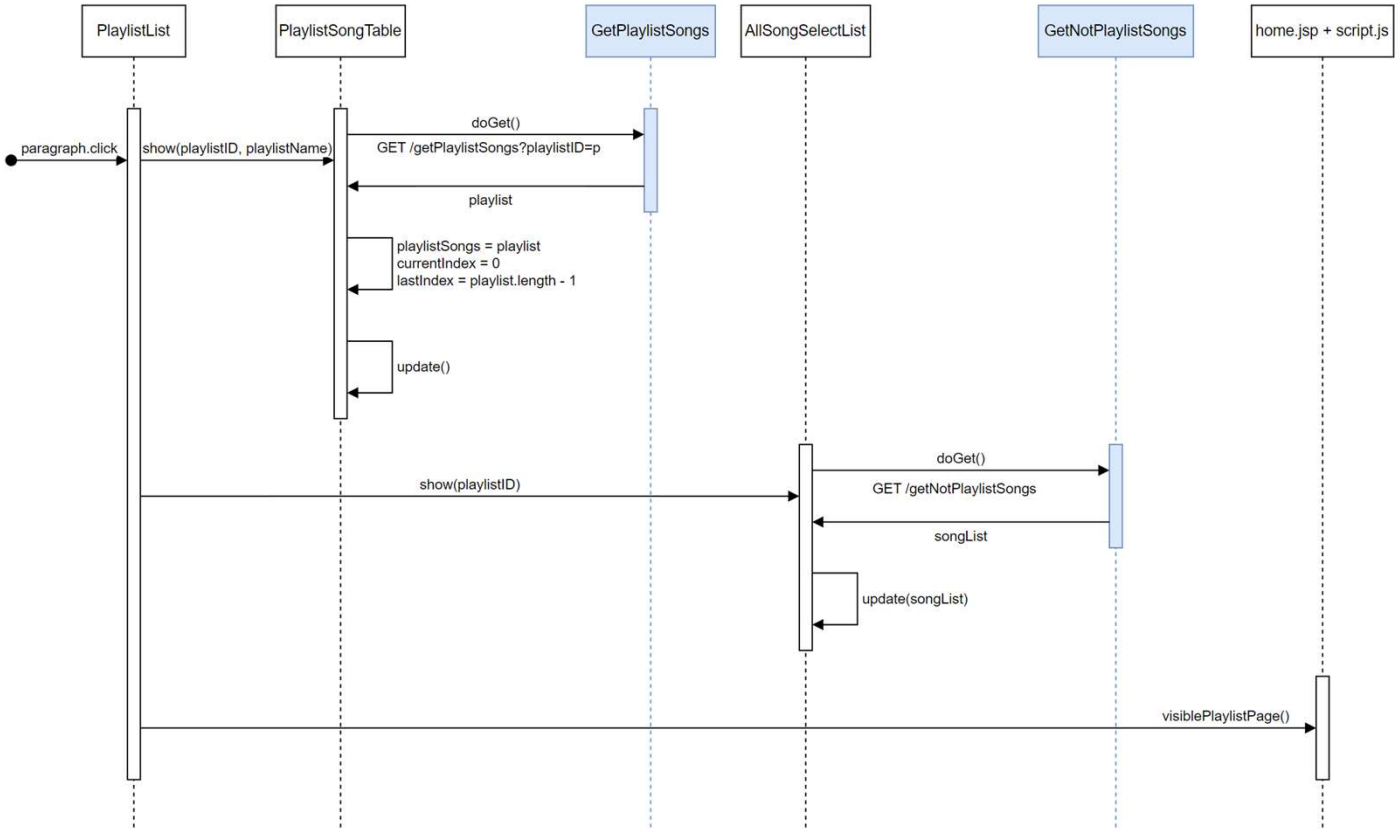
Events: addSong (client side)



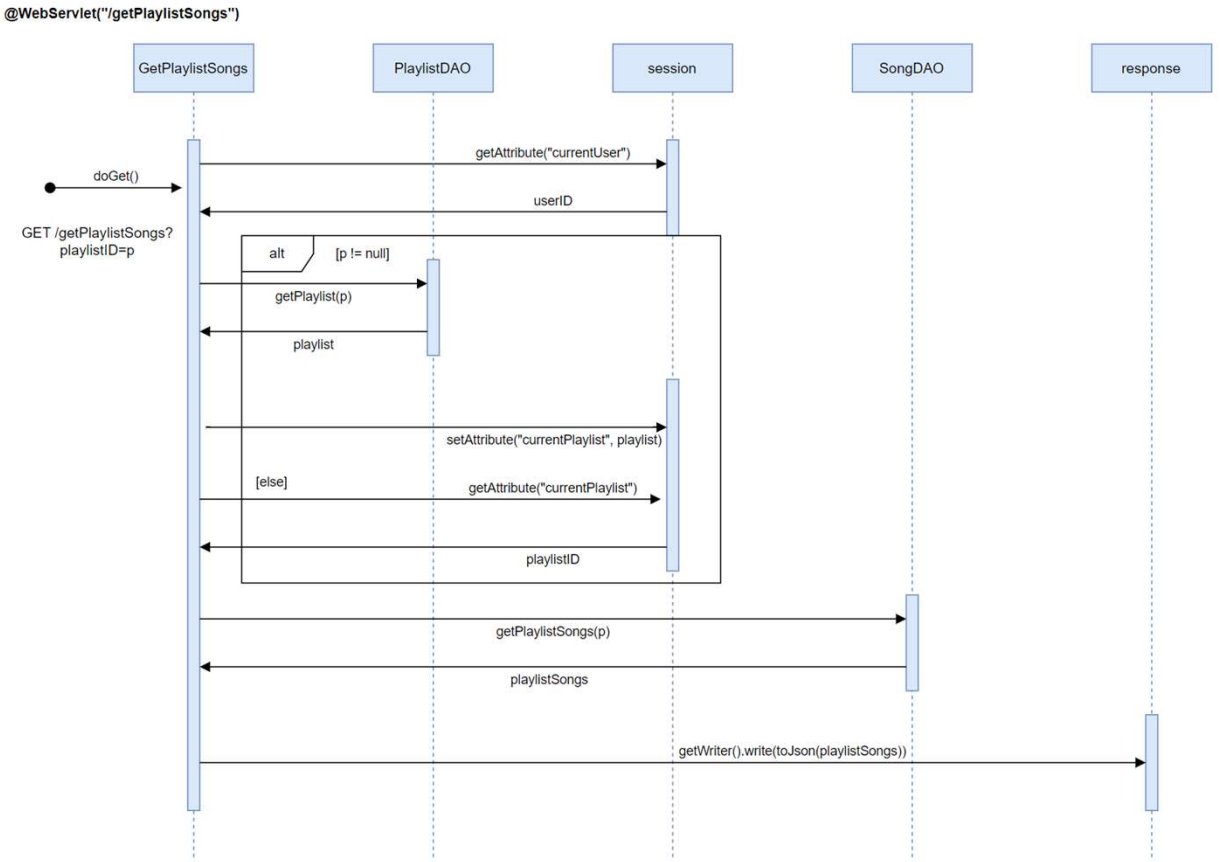
Events: addSong (server side)



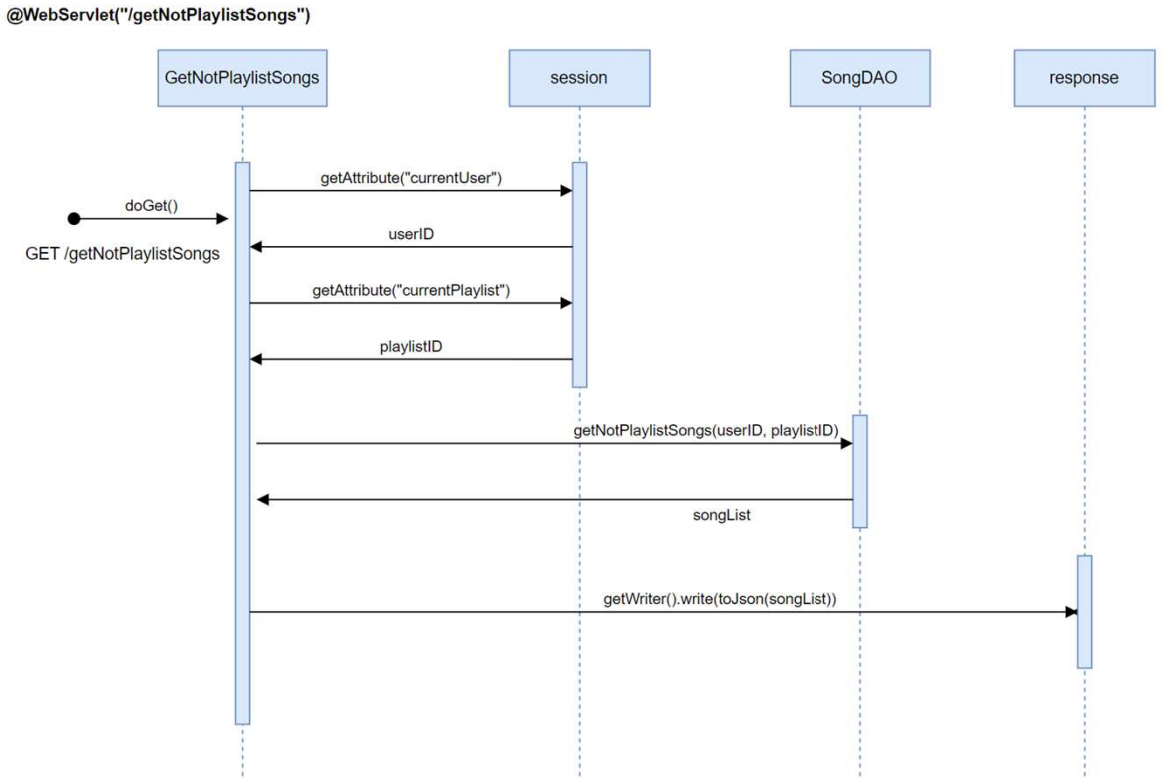
Events: selected playlist (client side)



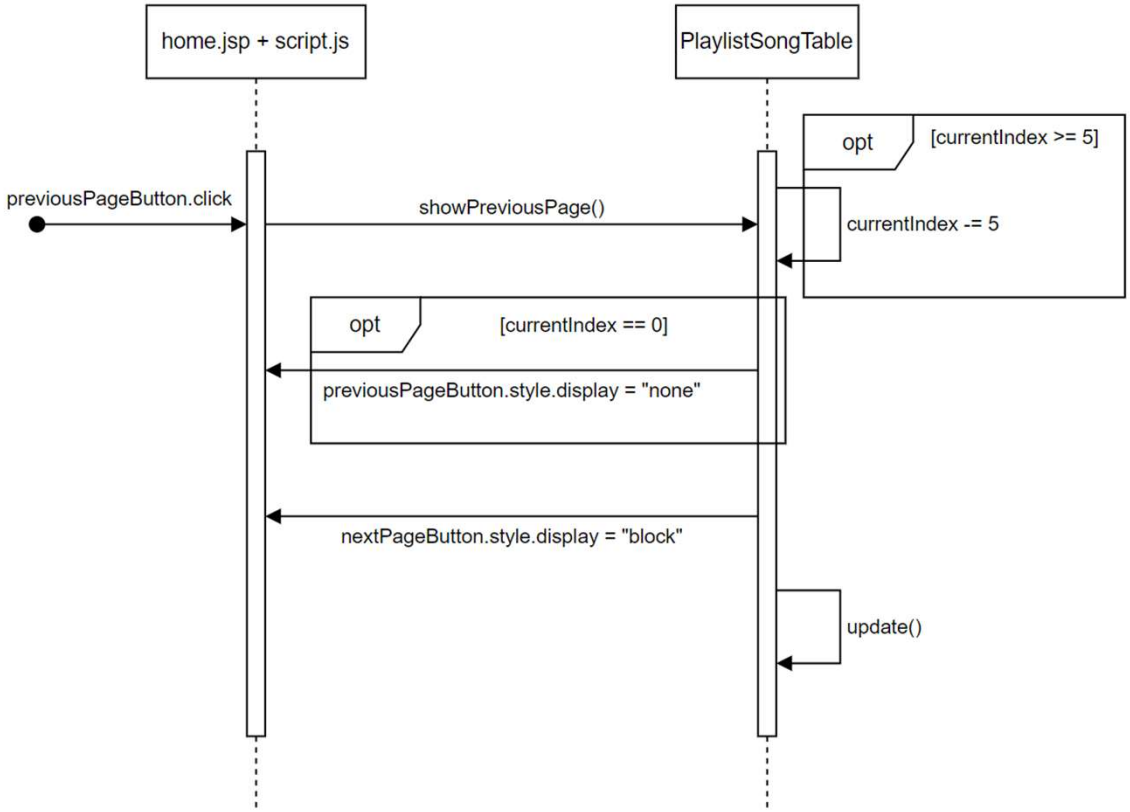
Event: getPlaylistSongs (server side)



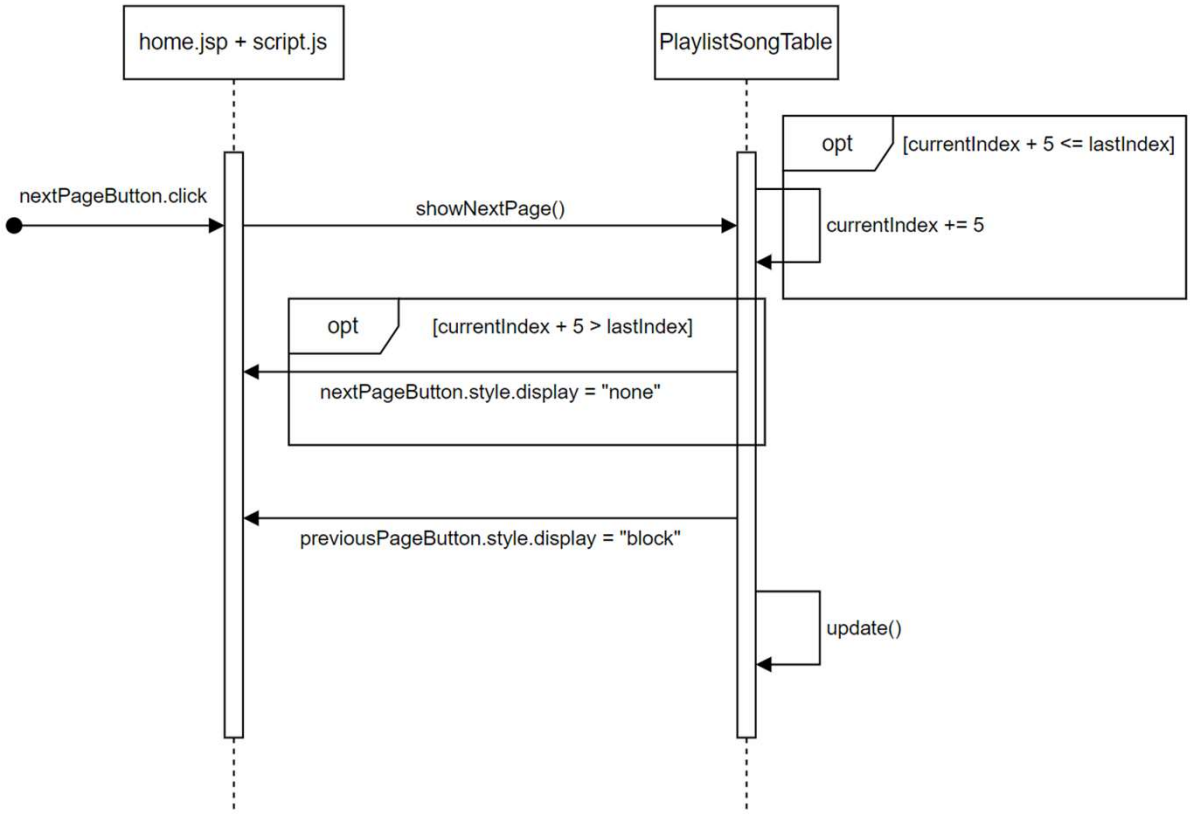
Events: getNotPlaylistSongs (server side)



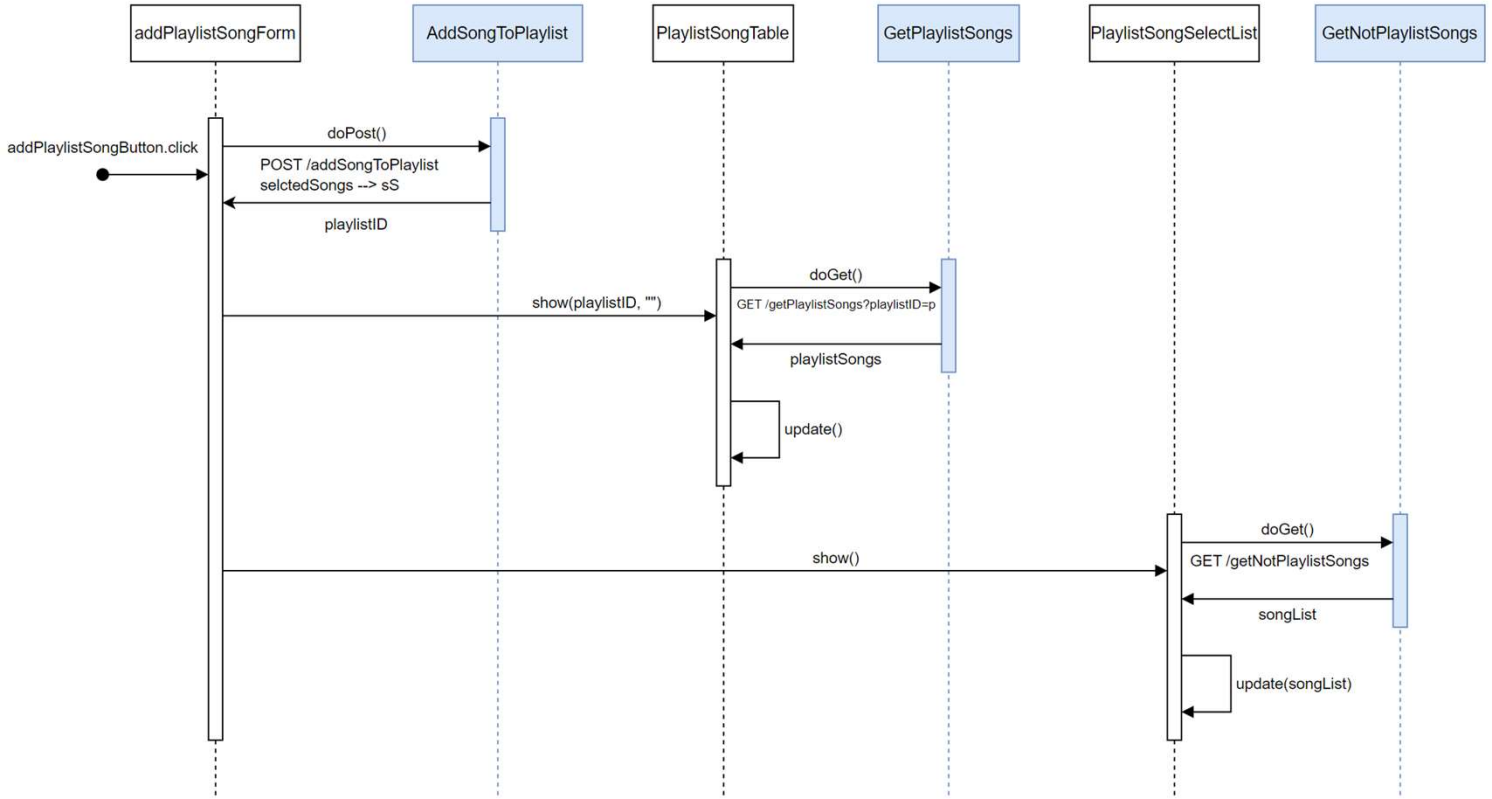
Events: previousPage



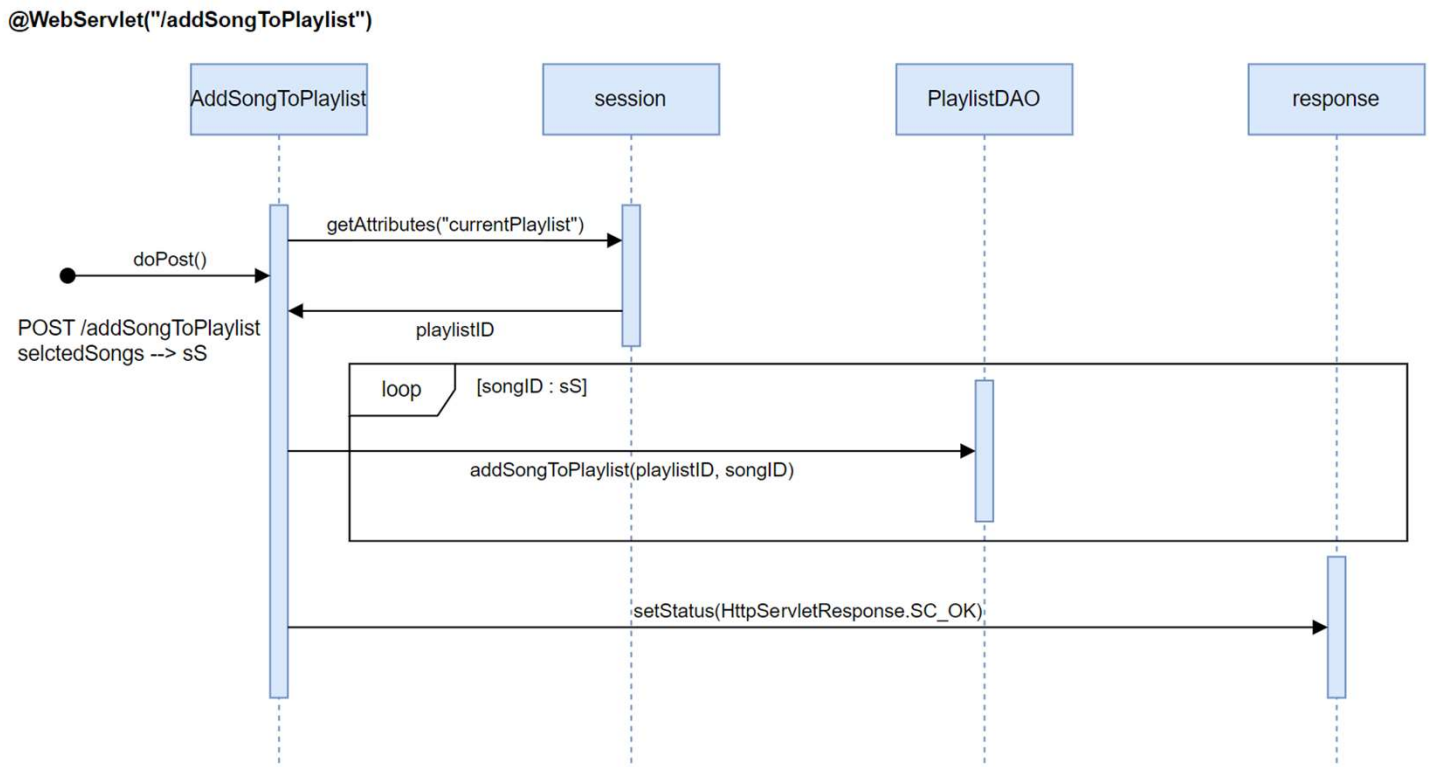
Events: nextPage



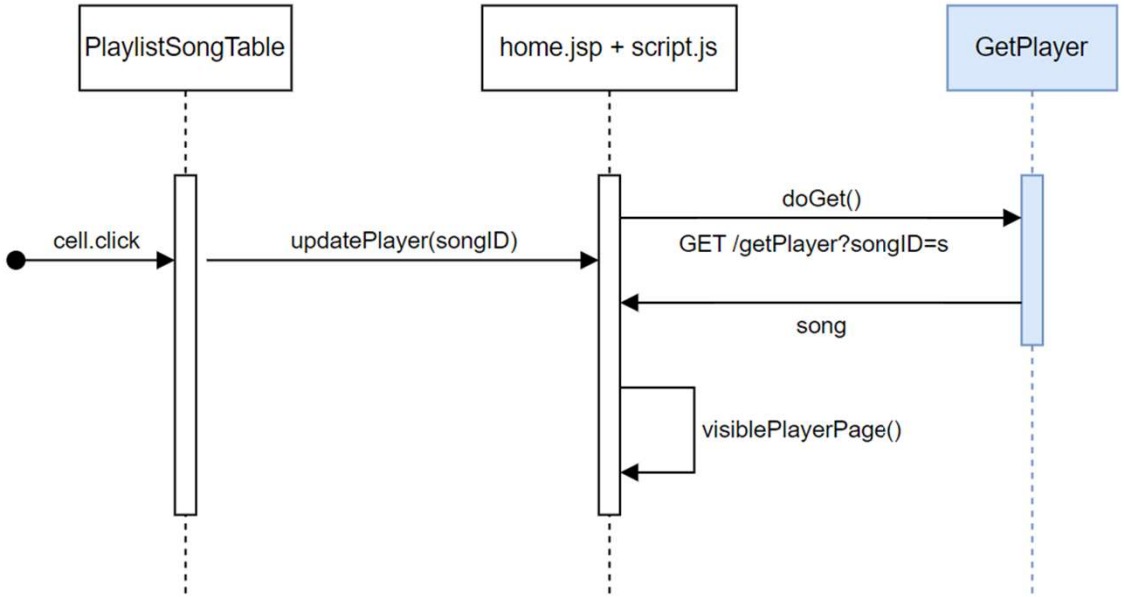
Events: addSongToPlaylist (client side)



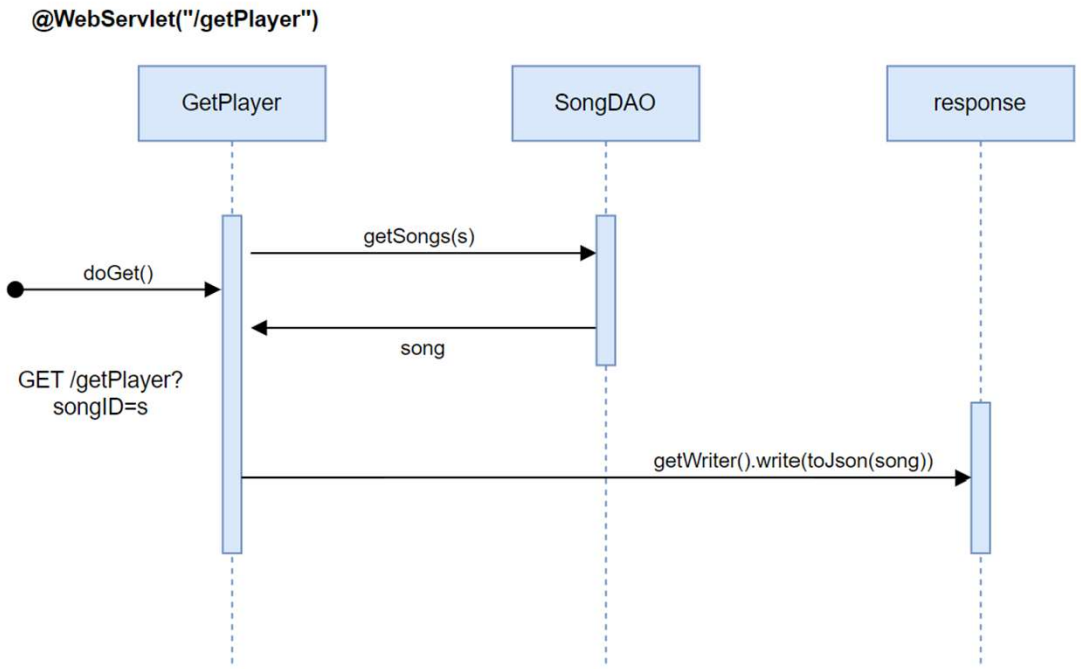
Events: addSongToPlaylist (server side)



Events: selected song (client side)



Events: getPlayer(server side)



Events: logout