Playlist Musicale
Versione HTML Pura

Web technologies

Melanie Tonarelli

10787497

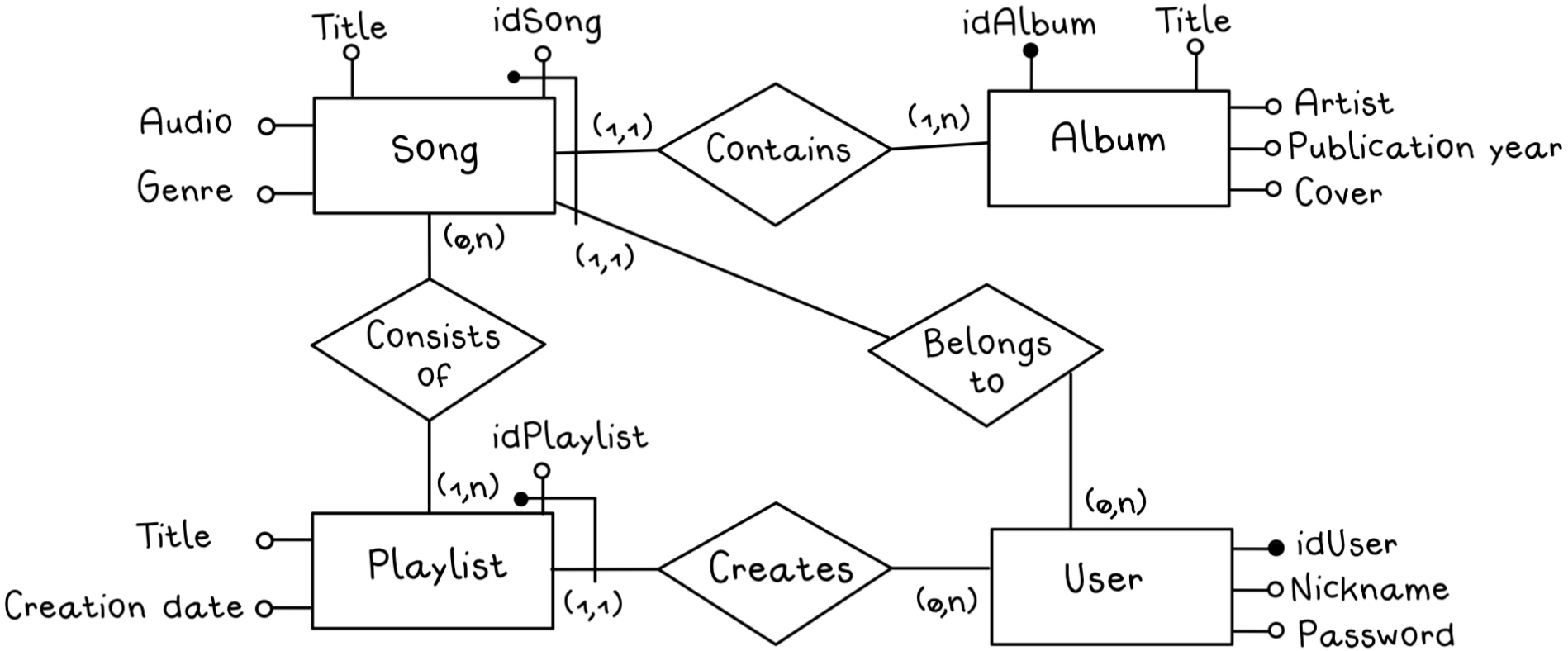
Playlist Musicali

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. 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.

Analisi dei dati

- 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



Local database schema

```
CREATE TABLE `Album` (
  `idAlbum` int NOT NULL AUTO_INCREMENT,
  `titleAlbum` varchar(45) NOT NULL,
  `artist` varchar(45) NOT NULL,
  `year` year NOT NULL,
  `cover` longblob NOT NULL,
  `userId` varchar(45) NOT NULL,
  PRIMARY KEY (`idAlbum`),
  UNIQUE KEY `constraint_unique`
        (`titleAlbum`,`artist`,`userId`),
  KEY `user_idx` (`userId`),
  CONSTRAINT `userId` FOREIGN KEY (`userId`) REFERENCES `User`
      (`username`) ON UPDATE CASCADE
)
```

Local database schema

```
CREATE TABLE `InPlaylist` (
  `playlist` int NOT NULL,
  `song` int NOT NULL,
  PRIMARY KEY (`playlist`,`song`),
  KEY `song_idx` (`song`),
  CONSTRAINT `playlist` FOREIGN KEY (`playlist`) REFERENCES
     `Playlist` (`idPlaylist`) ON UPDATE CASCADE,
  CONSTRAINT `song` FOREIGN KEY (`song`) REFERENCES `Song`
     (`idSong`) ON UPDATE CASCADE
)

CREATE TABLE `User` (
  `username` varchar(45) NOT NULL,
  `password` varchar(45) NOT NULL,
  `name` varchar(45) NOT NULL,
   PRIMARY KEY (`username`)
)
```

Local database schema

```
CREATE TABLE `Song` (
  `idSong` int NOT NULL AUTO_INCREMENT,
  `user` varchar(45) NOT NULL,
  `album` int NOT NULL,
  `title` varchar(45) NOT NULL,
  `file` longblob,
  `genre` varchar(45) NOT NULL,
  PRIMARY KEY (`idSong`),
  UNIQUE KEY `constraint_unique` (`user`,`album`,`title`),
  KEY `user_idx` (`user`),
  KEY `album_idx` (`album`),
  CONSTRAINT `album` FOREIGN KEY (`album`) REFERENCES
      `Album` (`idAlbum`) ON UPDATE CASCADE,
  CONSTRAINT `user` FOREIGN KEY (`user`) REFERENCES `User`
      (`username`) ON UPDATE CASCADE
)
```

Local database schema

```
CREATE TABLE `Playlist` (
  `idPlaylist` int NOT NULL AUTO_INCREMENT,
  `idUser` varchar(45) NOT NULL,
  `name` varchar(45) NOT NULL,
  `creationDate` datetime NOT NULL DEFAULT
     CURRENT_TIMESTAMP,
  PRIMARY KEY (`idPlaylist`),
  UNIQUE KEY `constraint_unique` (`idUser`,`name`),
  KEY `idUser_idx` (`idUser`),
  CONSTRAINT `idUser` FOREIGN KEY (`idUser`) REFERENCES
     `User` (`username`) ON UPDATE CASCADE
)
```

Application 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.

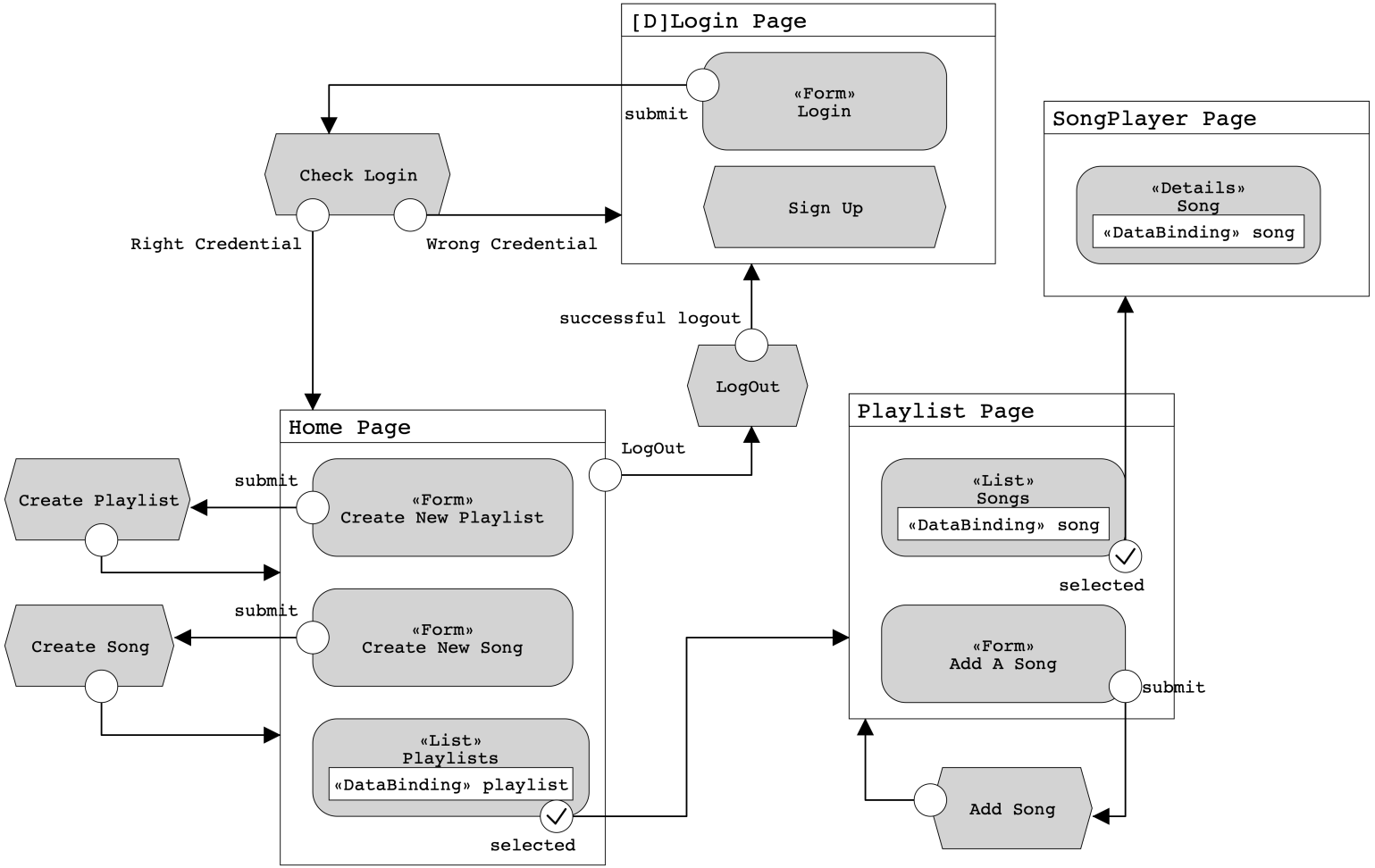
Pages (views), view components

Eventi e azioni (riassunto)

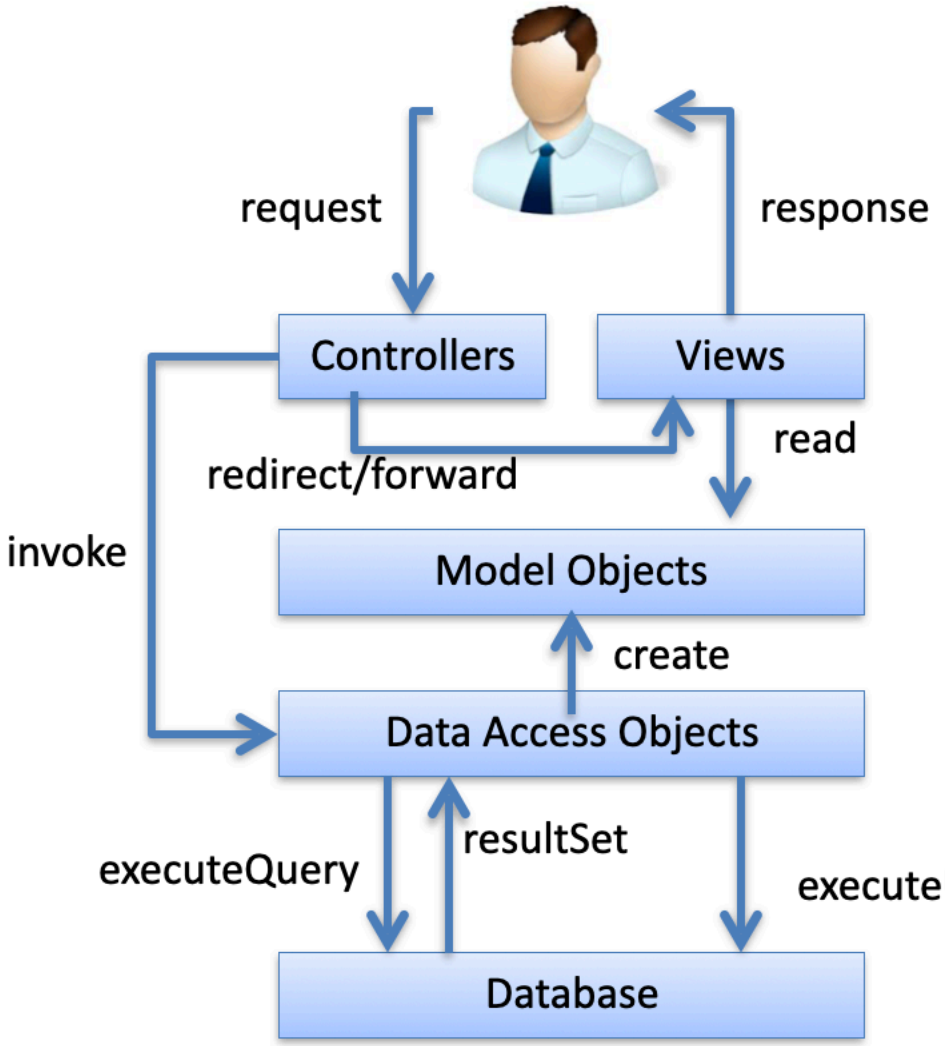
- Eventi
  1. Login
  2. Effettuare logout
  3. Accedere alla HomePage
  4. Accedere alla Playlist Page selezionando una playlist dalla HomePage
  5. Accedere alla Player page selezionando una canzone dalla Playlist pageCreare una canzone
  6. Creare una Playlist
  7. Aggiungere una canzone ad una data playlist

- Azioni
  1. Verificare credenziali (check login)
  2. Effettuare logout
  3. Preparare contenuto HomePage
  4. Preparare contenuto pagina PlaylistPage
  5. Creare una canzone
  6. Creare una playlist con canzoni
  7. Preparare contenuto pagina Player
  8. Aggiungere una canzone ad una playlist

Application design (IFML)



Architecture



- Web tier
  - Model View Controller (MVC)
    - Controller: handles event
    - View= presents the interface
    - Model: store the application status
- Business & data tier
  - If complex business logic is required, a service layer can be interposed between the controllers and the DAO layer , to make DAO cope only with data access

Componenti

- Controllers (servlets)
  1. Login
  2. Logout
  3. GoToHomePage
  4. GoToPlaylistPage
  5. GoToPlayer
  6. CreateSong
  7. CreatePlaylist
  8. AddSongToPlaylist

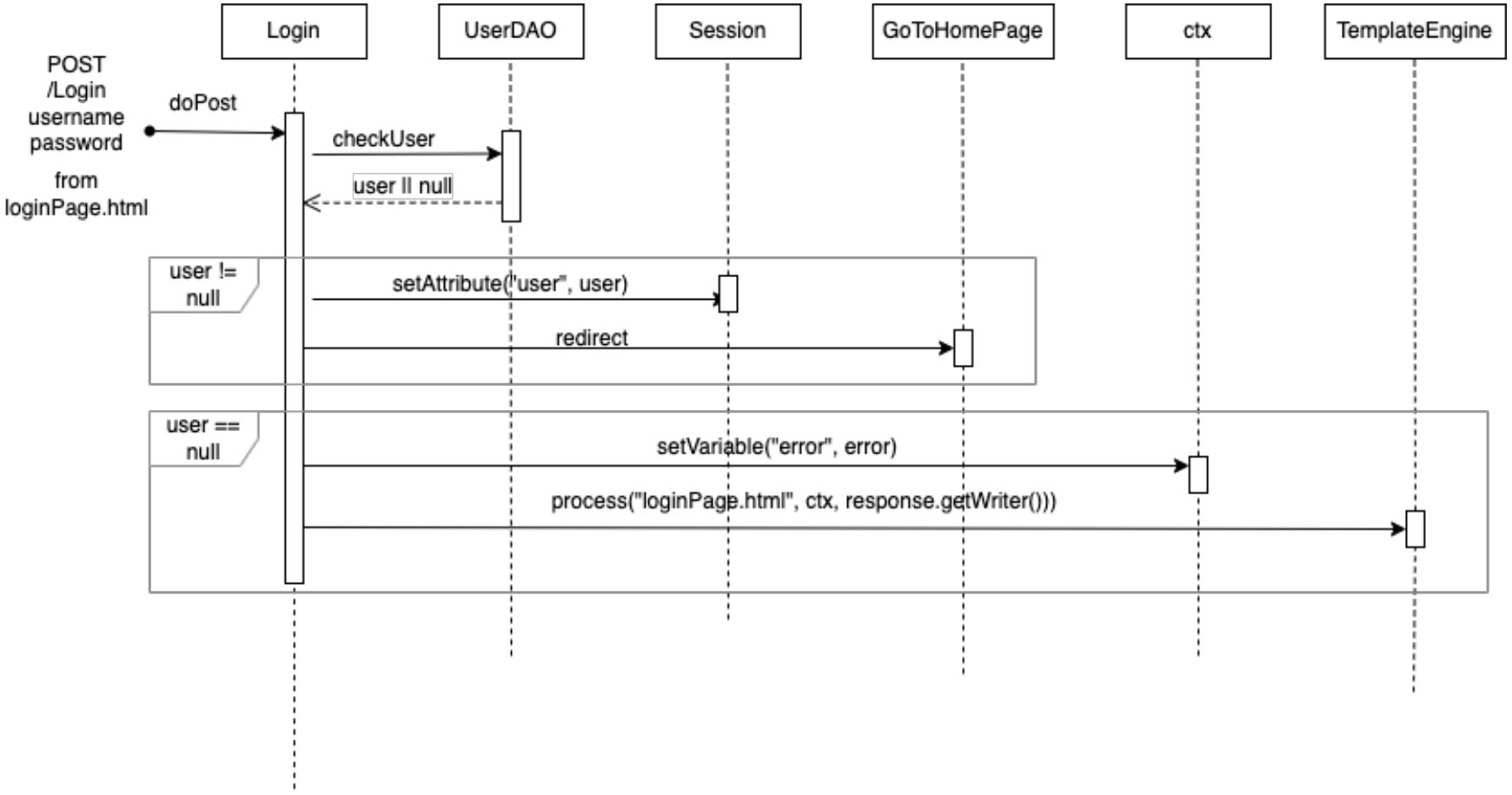
- Views (Templates) & components
  - Login
    - Login form
  - HomePage
    - Playlist list
    - Create Song form
    - Create Playlist form
  - PlaylistPage
    - Song list
    - Add song to playlist form
  - Player
    - Song details

- Model objects (Beans)
  - User
  - Playlist
  - Song
  - Album
- Data Access Objects (Classes)
  - UserDAO
    - checkUser(username, password)
  - PlaylistDAO
    - findPlaylists(username)
    - createPlaylistWithSongs(username, titlePlaylist, songs)
    - -createPlaylist(username, titlePlaylist)
    - -addSongInPlaylist(idPlaylist, idSong)
    - -findIdPlaylist(username, title)
    - isPlaylistPresent(username, idPlaylist)
    - getPlaylistById(username. idPlaylist)
    - isSongPresentInPlaylist(idSong, idPlaylist)
  - SongDAO
    - createSongAlbum(user, titleSong, genre, file, titleAlbum, artist, year, cover)
    - -createSong(user, title, genre, file, idAlbum)
    - -findIdAlbum(user, title, artist)
    - -createAlbum(user, title, artist, year, cover)
    - findAllSongsByUsername(username)
    - findAllSongsInPlaylist(username, idPlaylist)
    - findSongsNotInPlaylist(username, idPlaylist)
    - findAllSongInfoById(username, idSong)
    - isSongPresent(username, idSong)
    - songAlreadyExists(username, titleSong, titleAlbum, artist, year, cover)

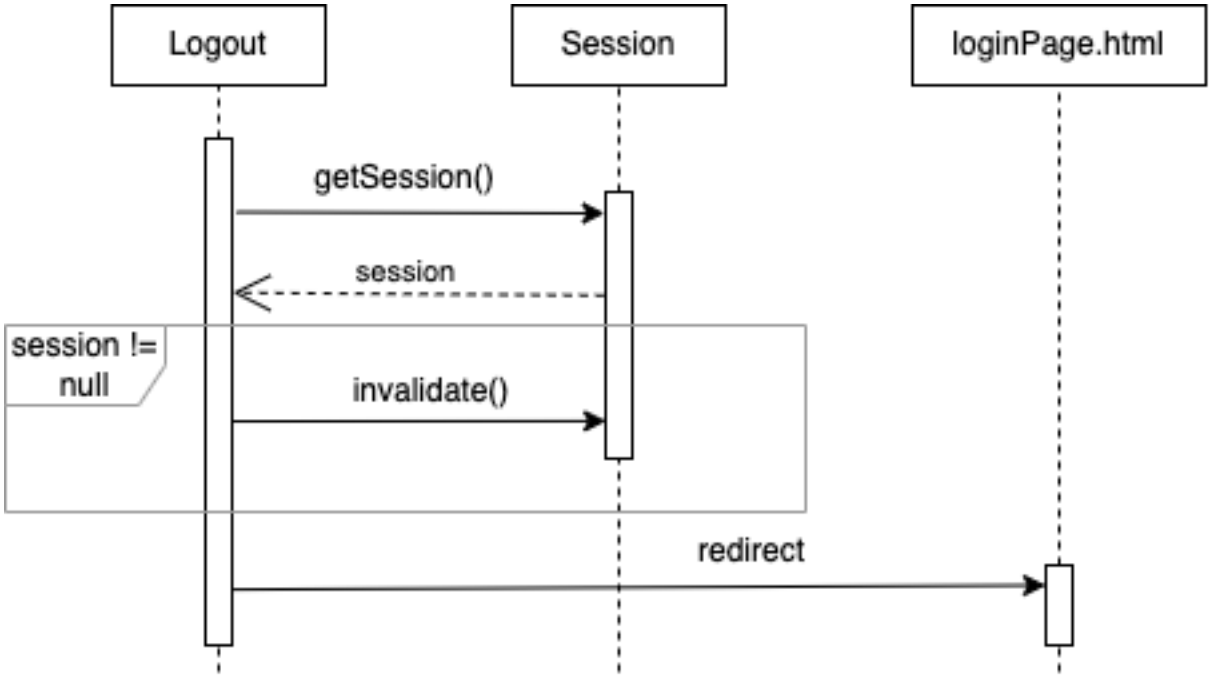
Sequence diagrams

- Per motivi di chiarezza e semplicità, ometto i controlli sui parametri in ingresso e la loro relativa gestione.
- Ogni volta che c'è un'istanza di un oggetto DAO, essa riceve come parametro una connection al database.

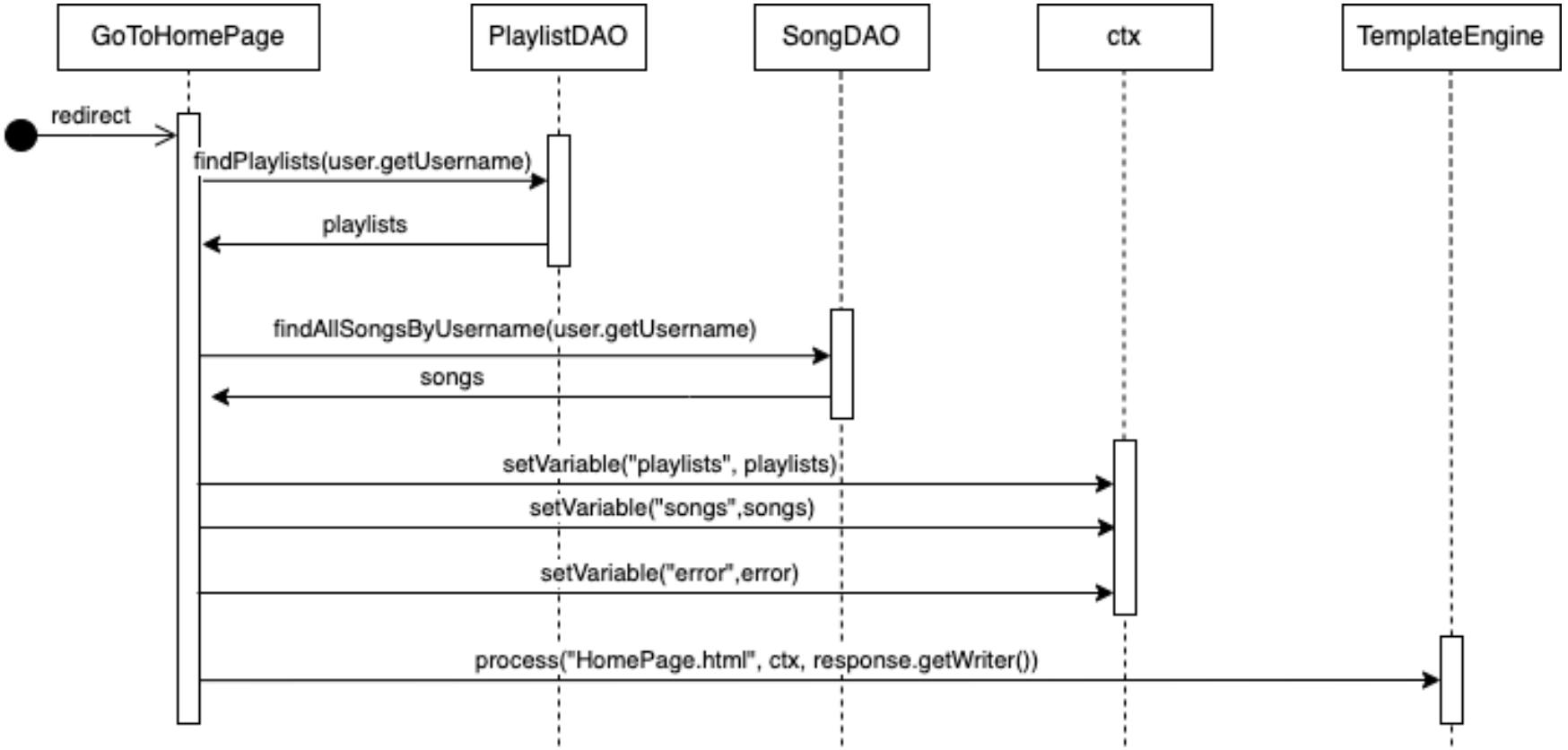
Event: login


Login
UserDAO
Session
GoToHomePage
ctx
TemplateEngine
POST
/Login
username
password
from
loginPage.html
doPost
checkUser
user || null
user !=
null
setAttribute("user", user)
redirect
user ==
null
setVariable("error", error)
process("loginPage.html", ctx, response.getWriter()))

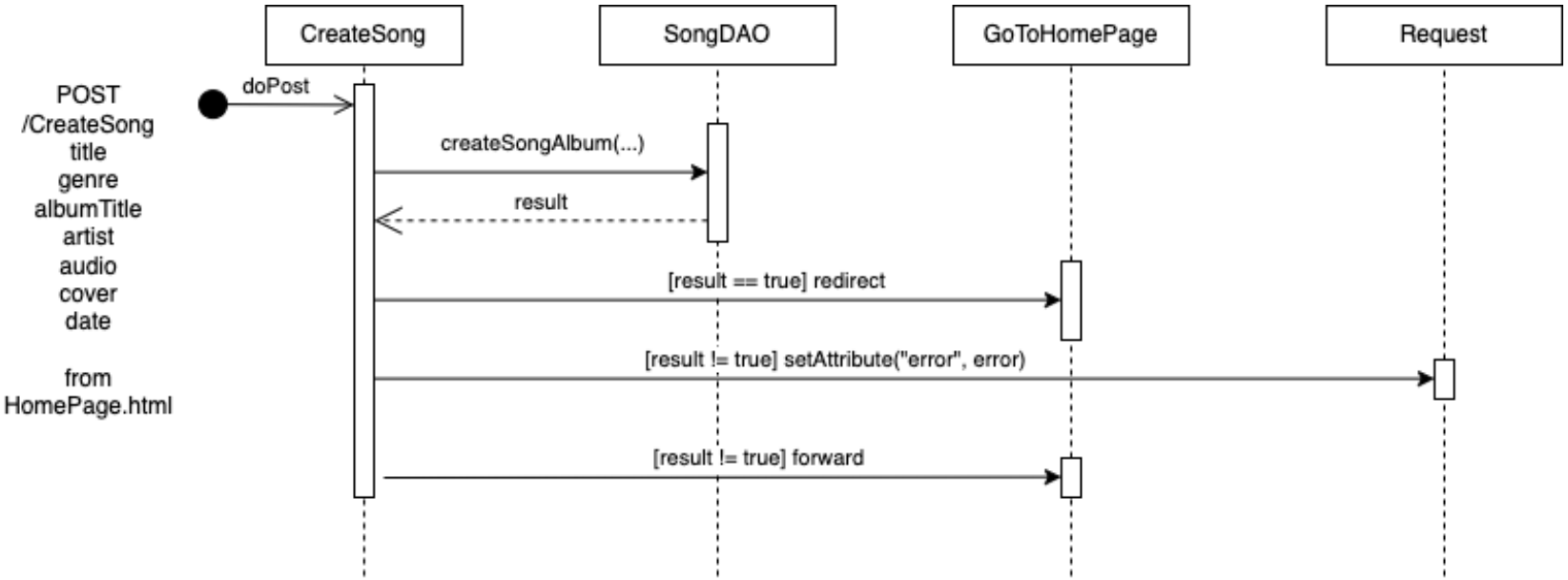
Event: logout



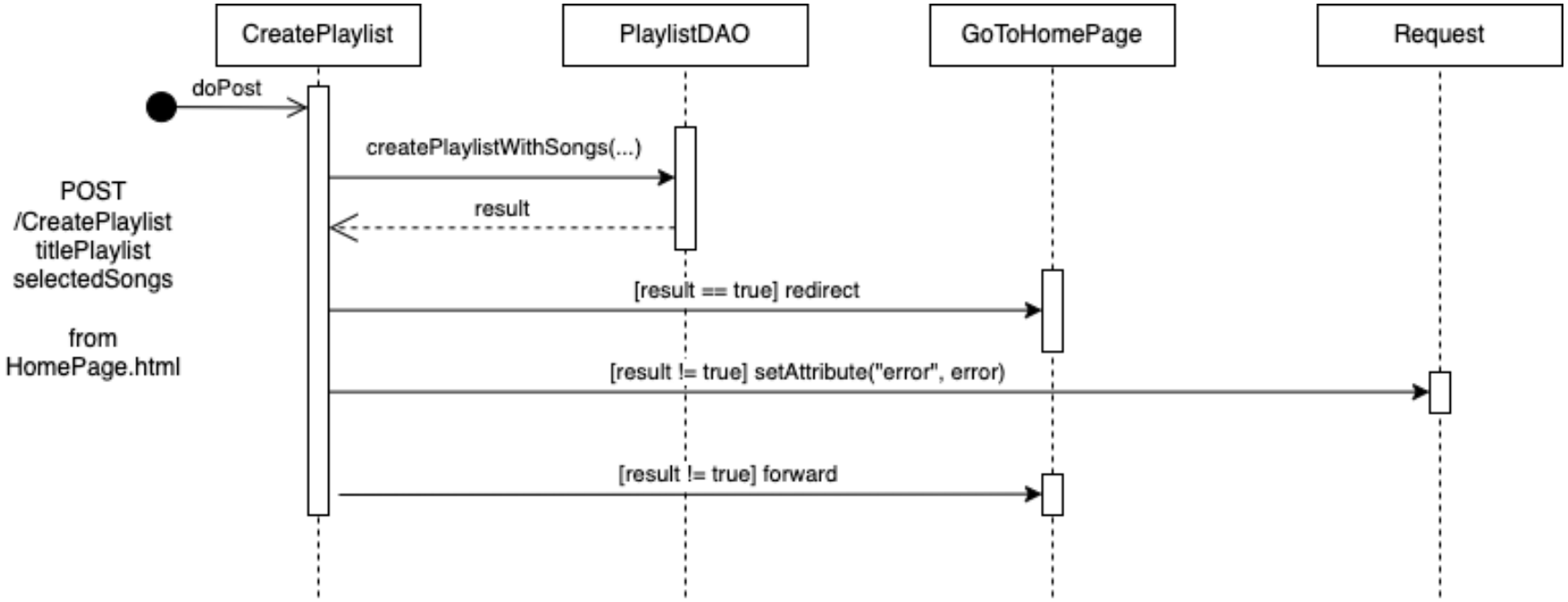
Event: go to home page



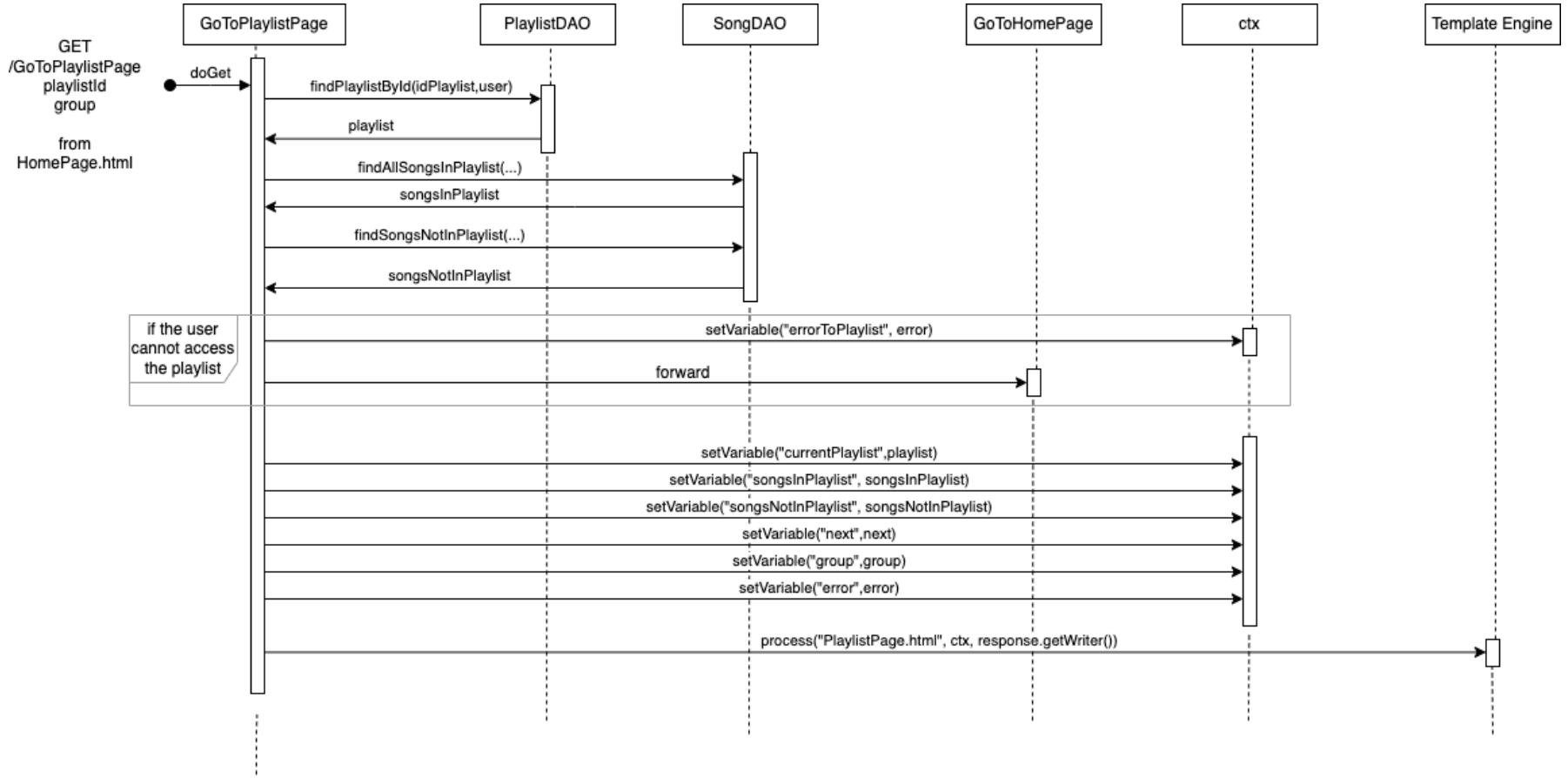
Event: create a new song



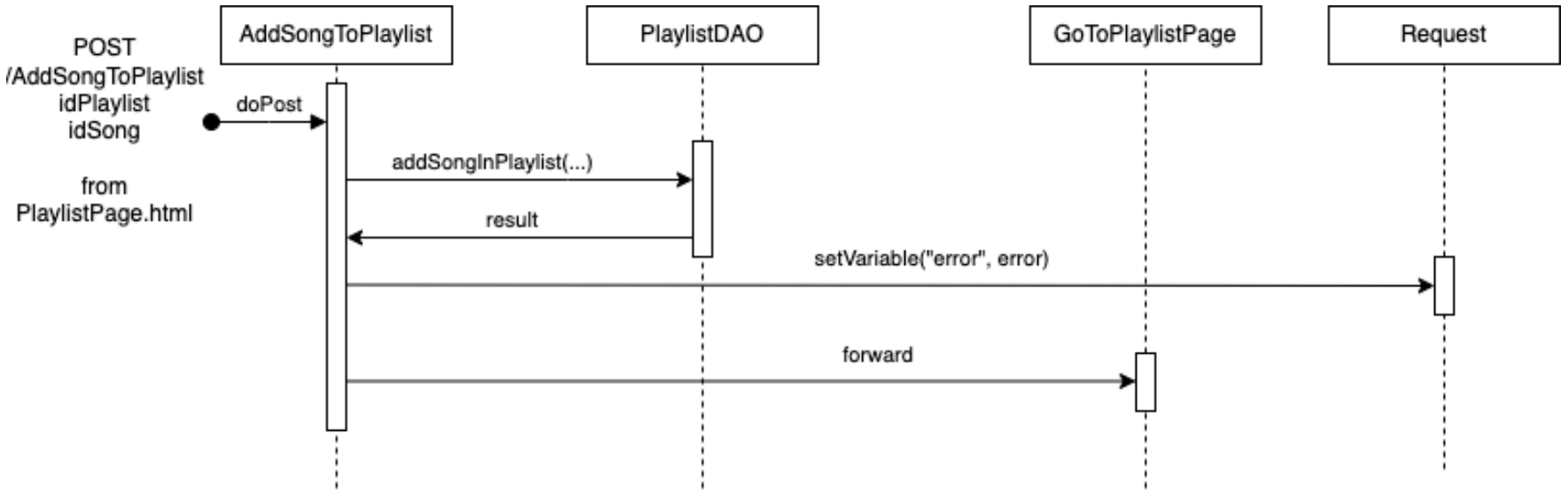
Event: create a new playlist



Event: go to playlist page



Event: add song to playlist



Event: go to player