Progetto di Tecnologie Informatiche per il Web

Michele Sangaletti

Indice

1. Traccia

Versione HTML pura

Un'applicazione web consente la gestione di una playlist di brani musicali.

Playlist e brani sono personali di ogni utente e non condivisi. Ogni utente ha username, password, nome e cognome. 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. Non è richiesto di memorizzare l'ordine con cui i brani compaiono nell'album a cui appartengono. Si ipotizzi che un brano possa appartenere a un solo album (no compilation). 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. 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, un FORM per caricare un BRANO con tutti i dati relativi e un form per creare una nuova playlist.
Il FORM per la creazione di una nuova PLAYLIST mostra l'elenco dei brani dell'utente ordinati per ordine alfabetico crescente dell'autore o gruppo e per data crescente di pubblicazione dell'album a cui il brano appartiene. Tramite il form è possibile 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 ordine alfabetico crescente dell'autore o gruppo e per data crescente di pubblicazione dell'album a cui il brano appartiene. 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 ed esistono sia precedenti sia successivi, compare a destra della riga il bottone SUCCESSIVI e a sinistra il bottone PRECEDENTI.
La pagina PLAYLIST contiene anche un FORM che consente di selezionare e AGGIUNGERE uno o più BRANI alla playlist corrente, se non già presente nella playlist. Tale form presenta i brani da scegliere nello stesso modo del form usato per creare una 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.

Versione con JavaScript

Si realizzi un'applicazione client server web che modifica le specifiche precedenti come segue:

- 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 personalizzato diverso da quello di default. Dalla HOME con un link associato a ogni playlist si accede a una finestra modale RIORDINO, che mostra la lista completa dei brani della playlist ordinati secondo il criterio corrente (personalizzato o di default). L'utente può 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. Un brano aggiunto a una playlist con ordinamento personalizzato è inserito nell'ultima posizione.

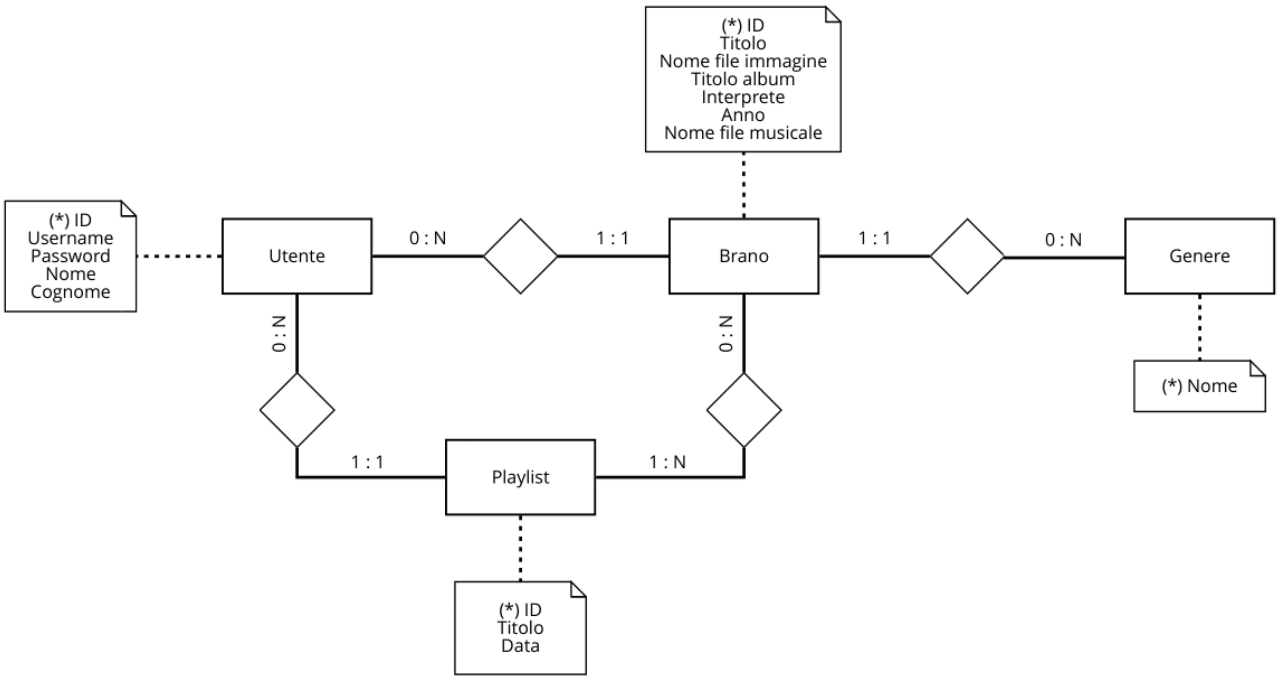
2. Documentazione ver. html pura

2.a. Analisi requisiti dati

Playlist e brani sono personali di ogni utente e non condivisi. Ogni **utente** ha *username*, *password*, *nome* e *cognome*. 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*. Non è richiesto di memorizzare l'ordine con cui i brani compaiono nell'album a cui appartengono. Si ipotizzi che un brano possa appartenere a un solo album (no compilation). 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. 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.

Legenda:
- **Entità**;
- *Attributi*;
- Relazioni.

2.a.a. Diagramma entità-relazioni



2.a.b. Database design

```sql
create table users
(
    id       int auto_increment,
    username varchar(32) not null unique,
    password varchar(32) not null,
    name     varchar(32) not null,
    surname  varchar(32) not null,
    primary key (id)
);

create table genres
(
    name varchar(32),
    primary key (name)
);
```

```
create table songs
(
    id              int auto_increment,
    user_id         int          not null,
    title           varchar(256) not null,
    image_file_name varchar(256) not null,
    album_title     varchar(256) not null,
    performer       varchar(256) not null,
    year            int          not null check ( year > 0 ),
    genre           varchar(256) not null,
    music_file_name varchar(256) not null,
    primary key (id),
    foreign key (user_id) references users (id) on update cascade on delete no action,
    foreign key (genre) references genres (name) on update cascade on delete no action,
    unique (user_id, music_file_name),
    unique (user_id, title)
);

create table playlists
(
    id      int auto_increment,
    user_id int          not null,
    title   varchar(256) not null,
    date    date         not null default current_date,
    primary key (id),
    unique (user_id, title)
);

create table playlist_contents
(
    playlist int,
    song     int,
    primary key (playlist, song),
    foreign key (playlist) references playlists (id) on update cascade on delete no
action,
    foreign key (song) references songs (id) on update cascade on delete no action
);
```

2.b. Analisi requisiti d'applicazione

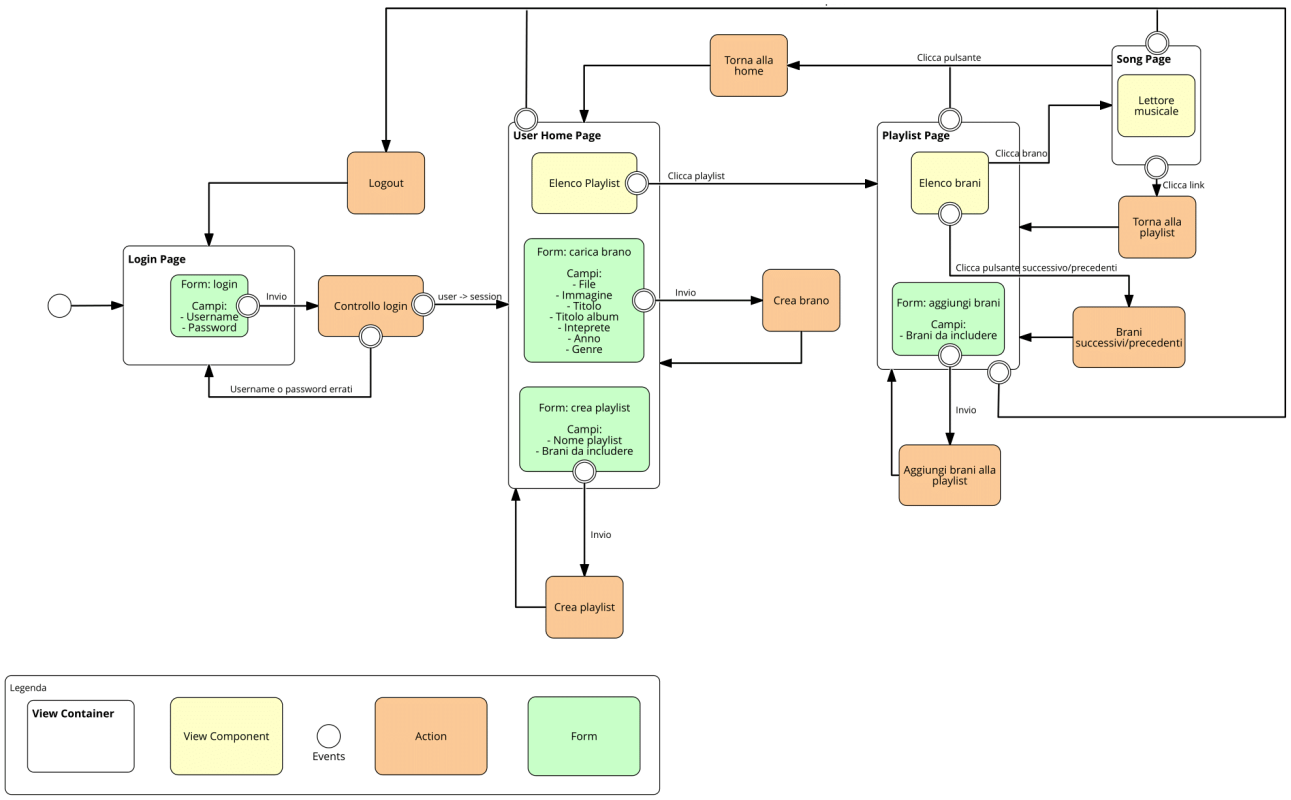
A seguito del LOGIN, l'utente accede all'HOME PAGE che presenta l'elenco delle proprie playlist, ordinate per data di creazione decrescente, un FORM per caricare un BRANO con tutti i dati relativi e un FORM per creare una nuova playlist.
Il FORM per la creazione di una nuova PLAYLIST mostra l'elenco dei brani dell'utente ordinati per ordine alfabetico crescente dell'autore o gruppo e per data crescente di pubblicazione dell'album a cui il brano appartiene. Tramite il form è possibile 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 ordine alfabetico crescente dell'autore o gruppo e per data crescente di pubblicazione dell'album a cui il brano appartiene. 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 ed esistono sia precedenti sia successivi, compare a destra della riga il bottone SUCCESSIVI e a sinistra il bottone PRECEDENTI.
La pagina PLAYLIST contiene anche un FORM che consente di selezionare e AGGIUNGERE uno o più BRANI alla playlist corrente, se non già presente nella playlist. Tale form presenta i brani da scegliere nello stesso modo del form usato per creare una 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.

Legenda:
- Pagine;
- Componenti;
- Eventi;
- Azioni.

2.b.a. Aggiunta alle specifiche

- Funzione di logout, accessibile tramite un pulsante dalle pagine di home, playlist e canzone;
- Funzione per tornare alla homepage, accessibile tramite un pulsante dalle pagine di playlist e canzone;
- Funzione per tornare alla pagina della playlist originale, accessibile tramite un link dalla pagina della canzone;
- Messaggio di «benvenuto» quando l'utente è nella home page;
- Nella pagina della playlist viene mostrata la data in cui è stata creata;
- Possibilità di creare una playlist senza brani;
- Istruzioni sul creare una playlist nella homepage se l'utente non ha playlist associate;
- Istruzioni sull'aggiungere brani alla playlist se vuota nella pagina della playlist;
- Istruzioni sul caricare canzoni nella pagina della playlist se l'utente o ha aggiunto tutti i suoi brani alla playlist o non ha brani associati.

2.b.b. Diagramma IFML

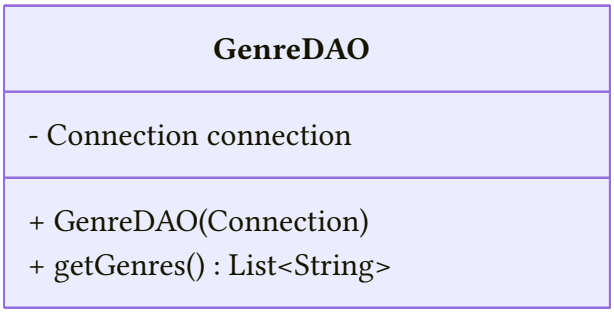


2.b.c. Componenti e viste

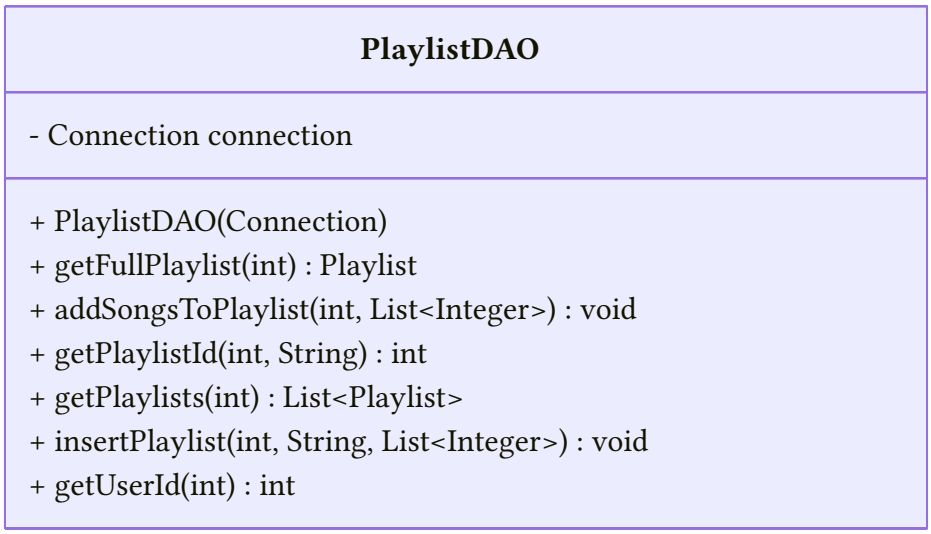
1. **Beans**

Tutti gli attributi sono riconducibili al diagramma ER del database (tranne per la tabella `playlist_contents`, che è stata incorporata dentro l'oggetto `Playlist` per convenienza), mentre i metodi sono i soliti getter e setter di Java.

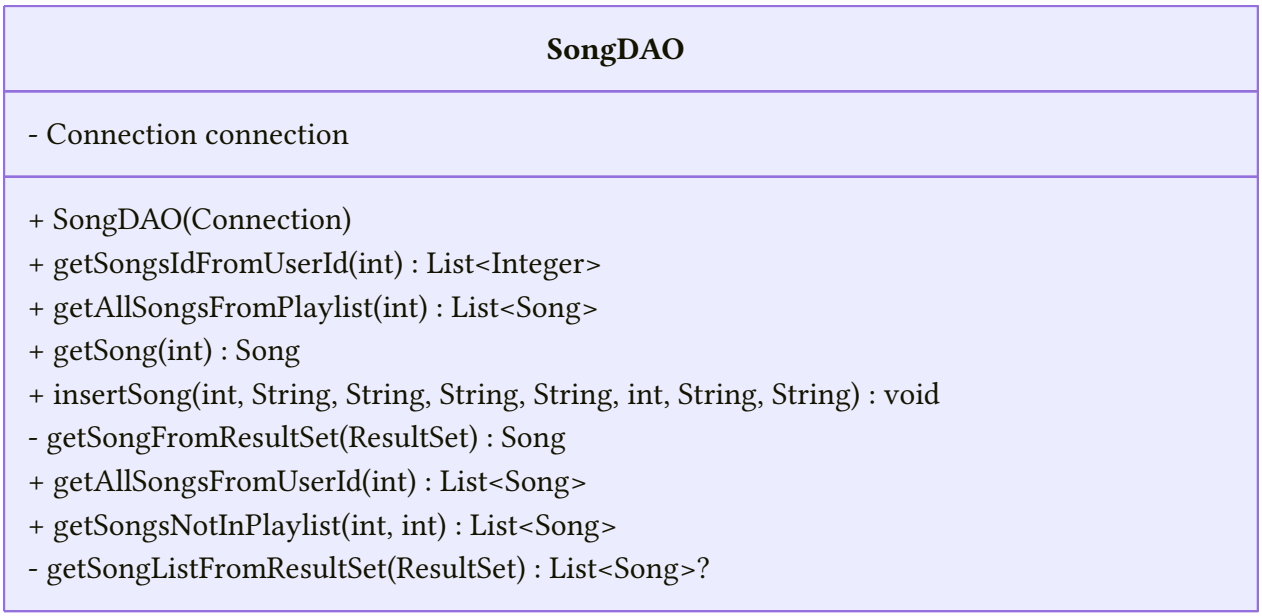
2. **DAO**



- `getGenres()`: ritorna una lista dei generi come stringhe.

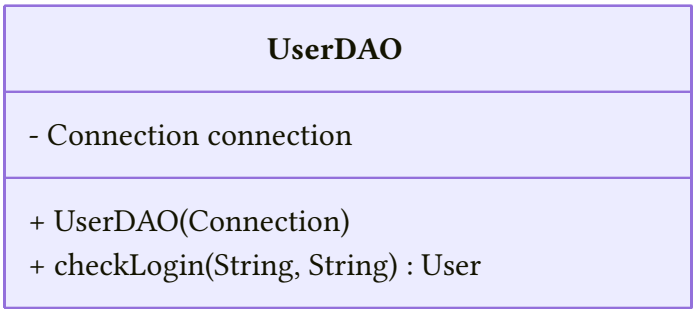


- `getPlaylists(int userId)`: ritorna una lista di `Playlist` create dallo user associato all'assegnato `userId`. Gli oggetti `Playlist` hanno l'attributo `songs = null`, dato che questa funzione viene chiamata solo per riempire la lista di playlist nella homepage;
- `getFullPlaylist(int playlistId)`: ritorna la `Playlist` con l'id dato con tutte le informazioni associate (compreso l'elenco di canzoni associate);
- `insertPlaylist(int userId, String title, List<Integer> songsId)`: crea una nuova playlist con le informazioni date e «oggi» comme data di creazione;
- `addSongsToPlaylist(int playlistId, List<Integer> songsId)`: aggiunge la coppia (playlistId, songId) alla tabella `playlist_contents` per ogni id nella lista `songsId`;
- `getPlaylistId(int userId, String title)`: ritorna l'id della playlist che ha associati lo user id e il titolo passati;
- `getUserId(int playlistId)`: ritorna l'id dello user che ha creato la playlist, o -1 altrimenti.



- `getAllSongsFromUserId(int userId)`: ritorna una lista di tutte le canzoni associate allo user dato, ordinate per interprete e anno, o `null` se non ne sono state trovate;
- `getAllSongsFromPlaylist(int playlistId)`: ritorna una lista di tutte le canzoni associate all'id della playlist dato, ordinate per interprete e anno, o `null` se non ne sono state trovate;

- `getSongListFromResultSet(ResultSet resultSet)`: metodo che estrare le canzoni dal set dato (se vuoto, ritorna `null`);
- `insertSong(int userId, String title, String imageFileName, String albumTitle, String performer, int year, String genre, String musicFileName)`: aggiunge la canzone alla tabella `songs`;
- `getSong(int songId)`: ritorna un oggetto `Song` dato l'id della canzone;
- `getSongsNotInPlaylist(int userId, int playlistId)`: ritorna una lista di tutte le canzoni associate al dato user che non appartengono alla data playlist;
- `getSongsIdFromUserId(int userId)`: ritorna una lista di tutti gli id delle canzoni associate al dato user;
- `getSongFromResultSet(ResultSet resultSet)`: estrae un'oggetto `Song` dal dato result set.

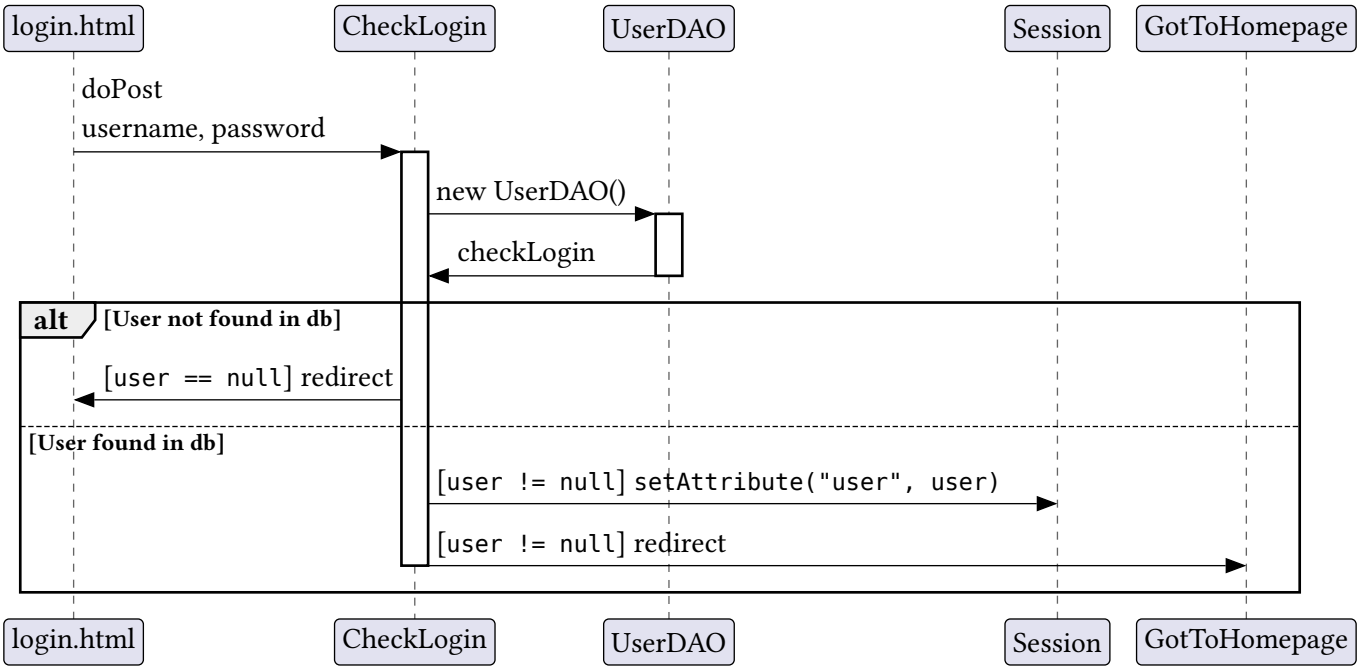


- `checkLogin(String username, String password)`: controlla nel database se un utente con i dati username e password esiste: in caso affermativo, ritorna un oggetto `User` con le informazioni dell'utente, `null` altrimenti.

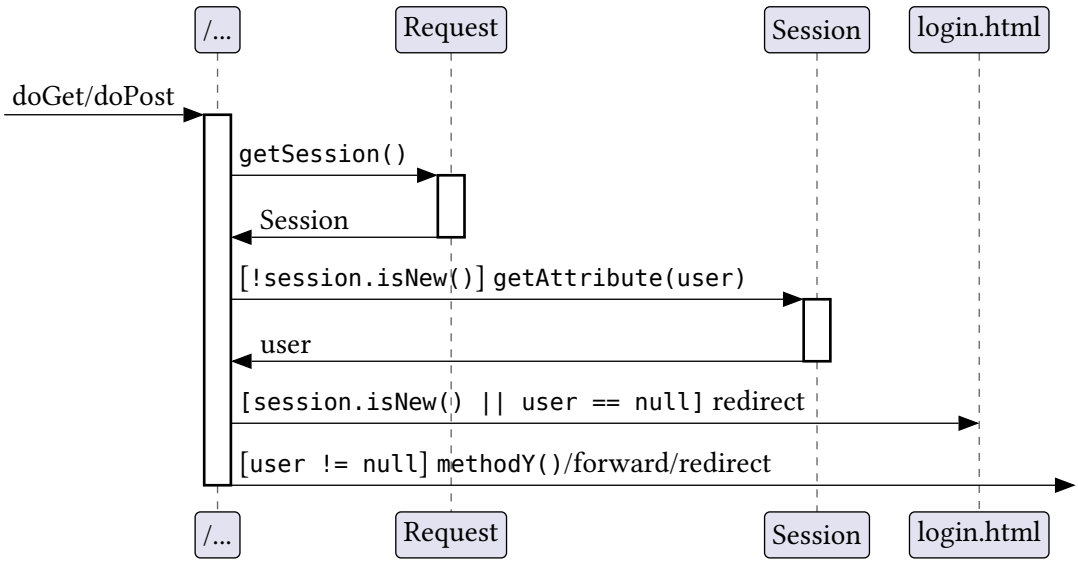
3. **Controllers**
4. **Utils**
5. **Templates**

2.b.d. Sequence diagrams

- **Login**

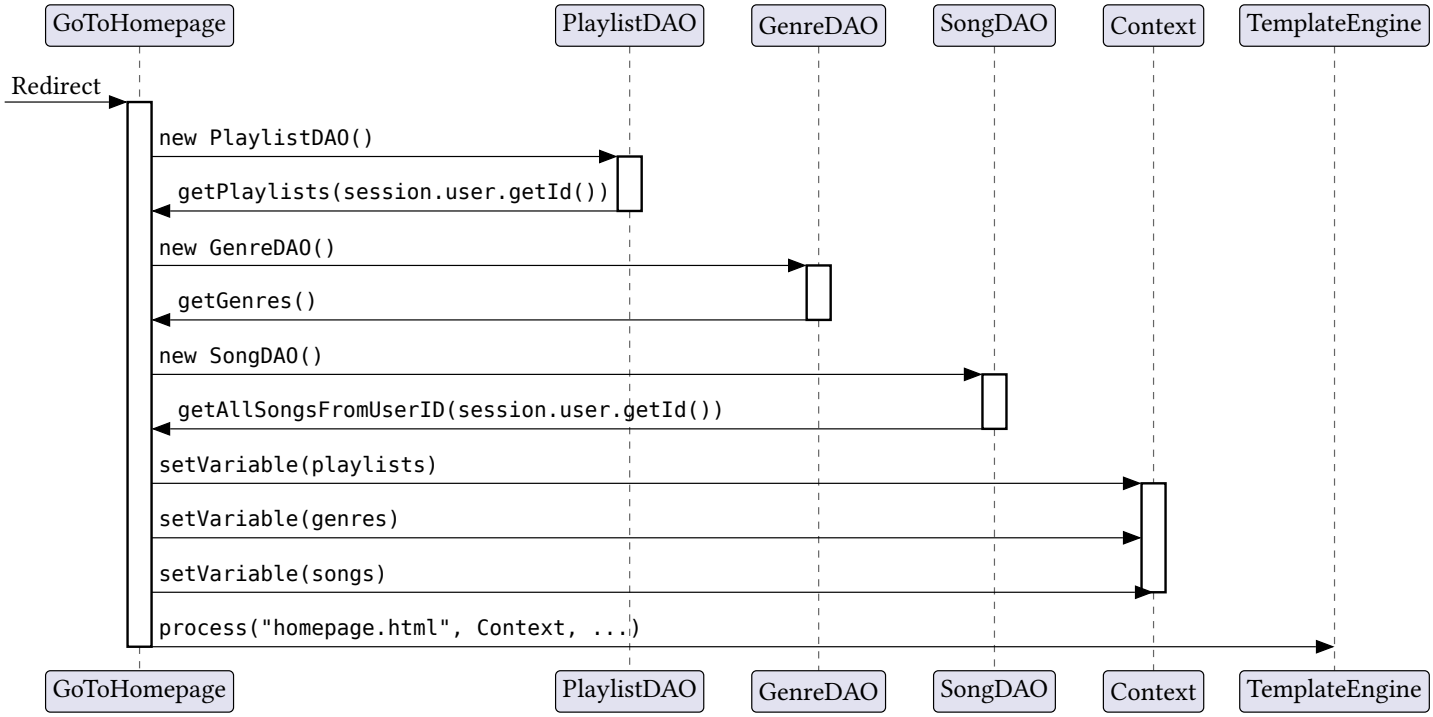


- **Controllare l'user**

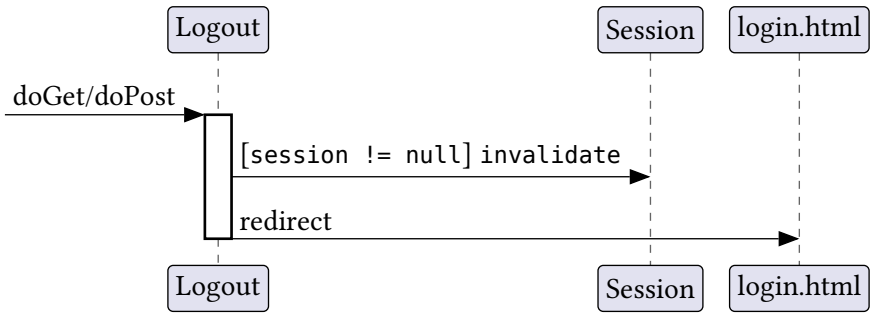


Questo controllo viene fatto all'inizio di ogni servlet da qui in poi, ed è stato riportato separatamente per sintesi.

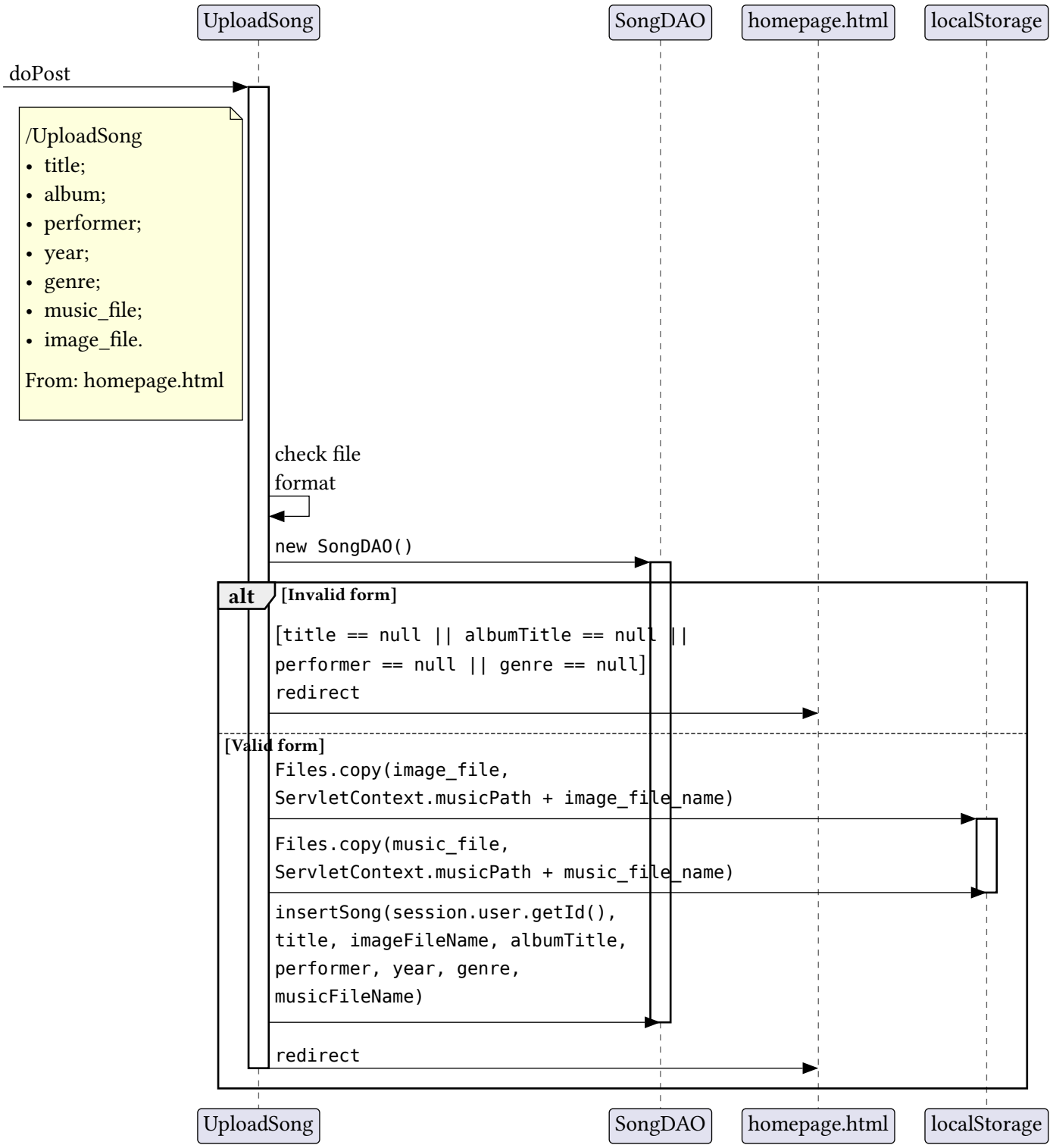
- **Tornare/andare alla homepage**



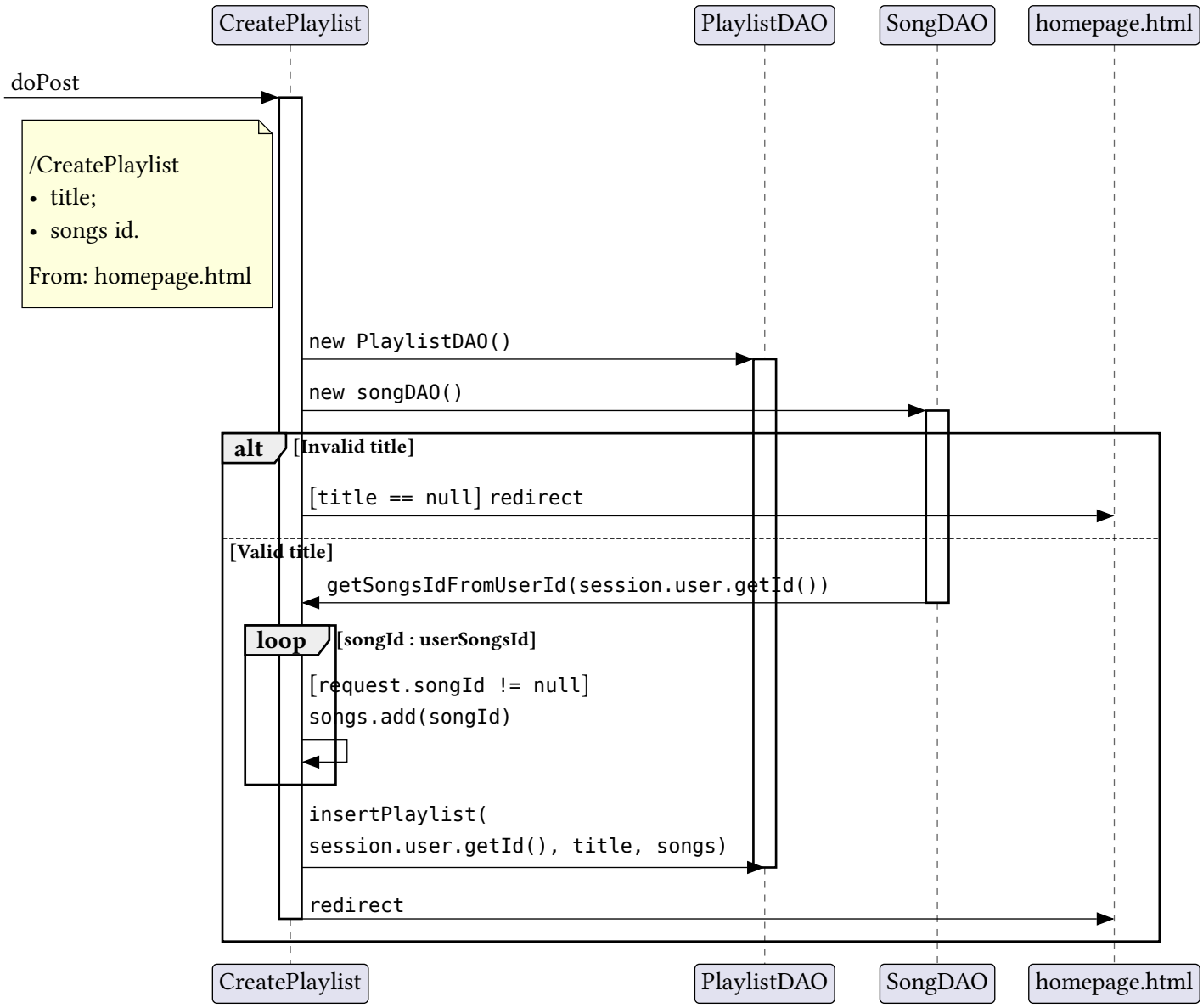
- **Logout**


Logout
Session
login.html
doGet/doPost
[session != null] invalidate
redirect
Logout
Session
login.html

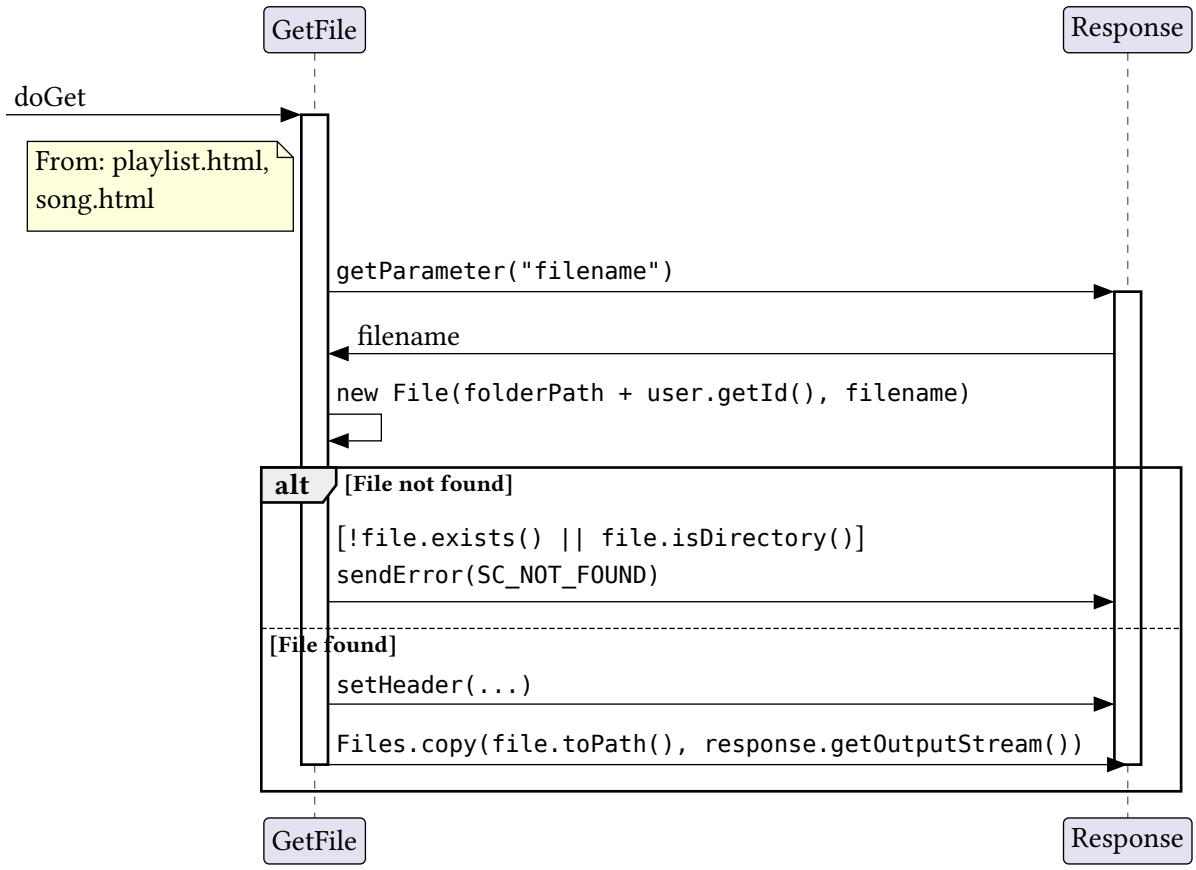
- **Caricare una canzone**



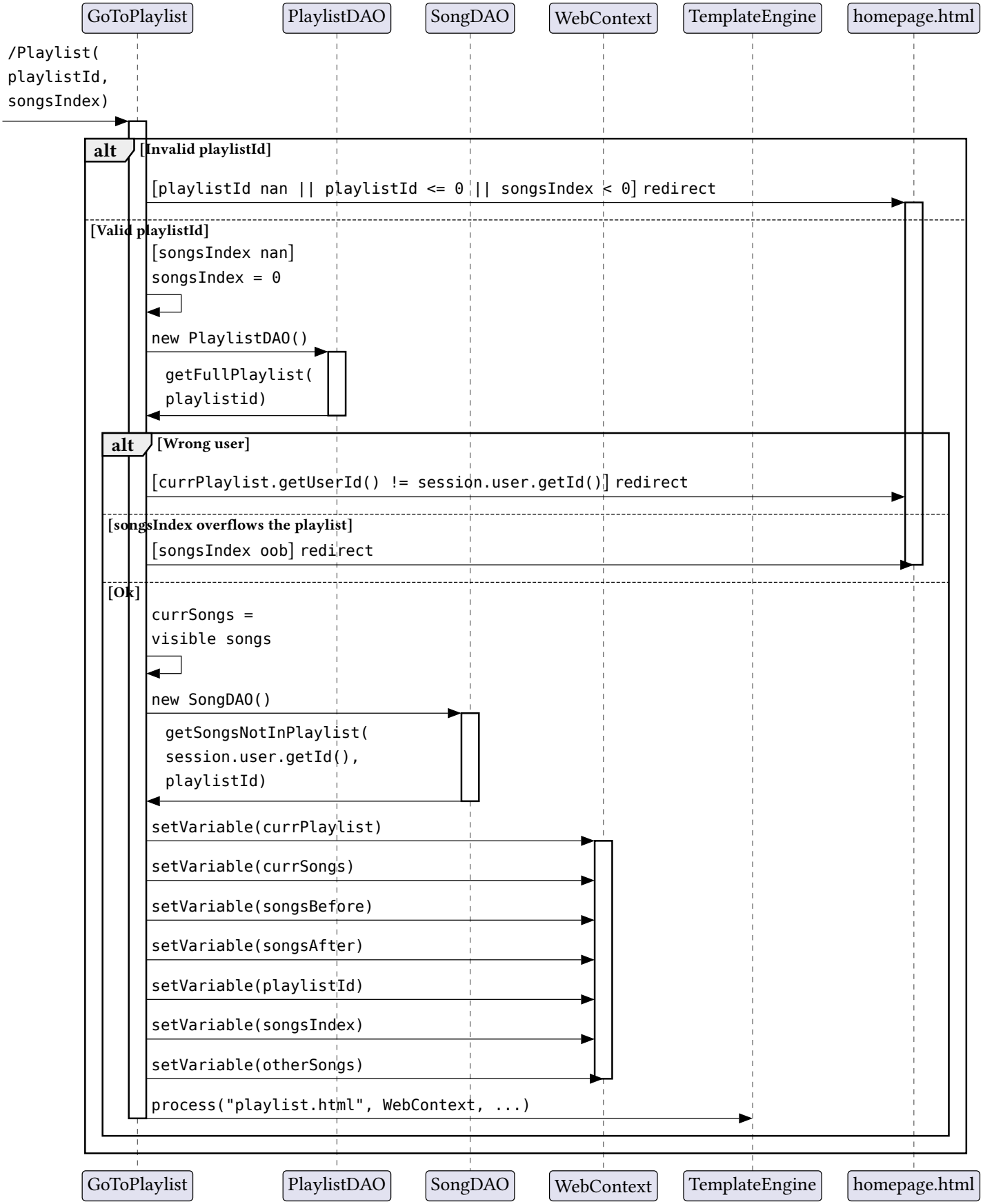
- **Creare una playlist**



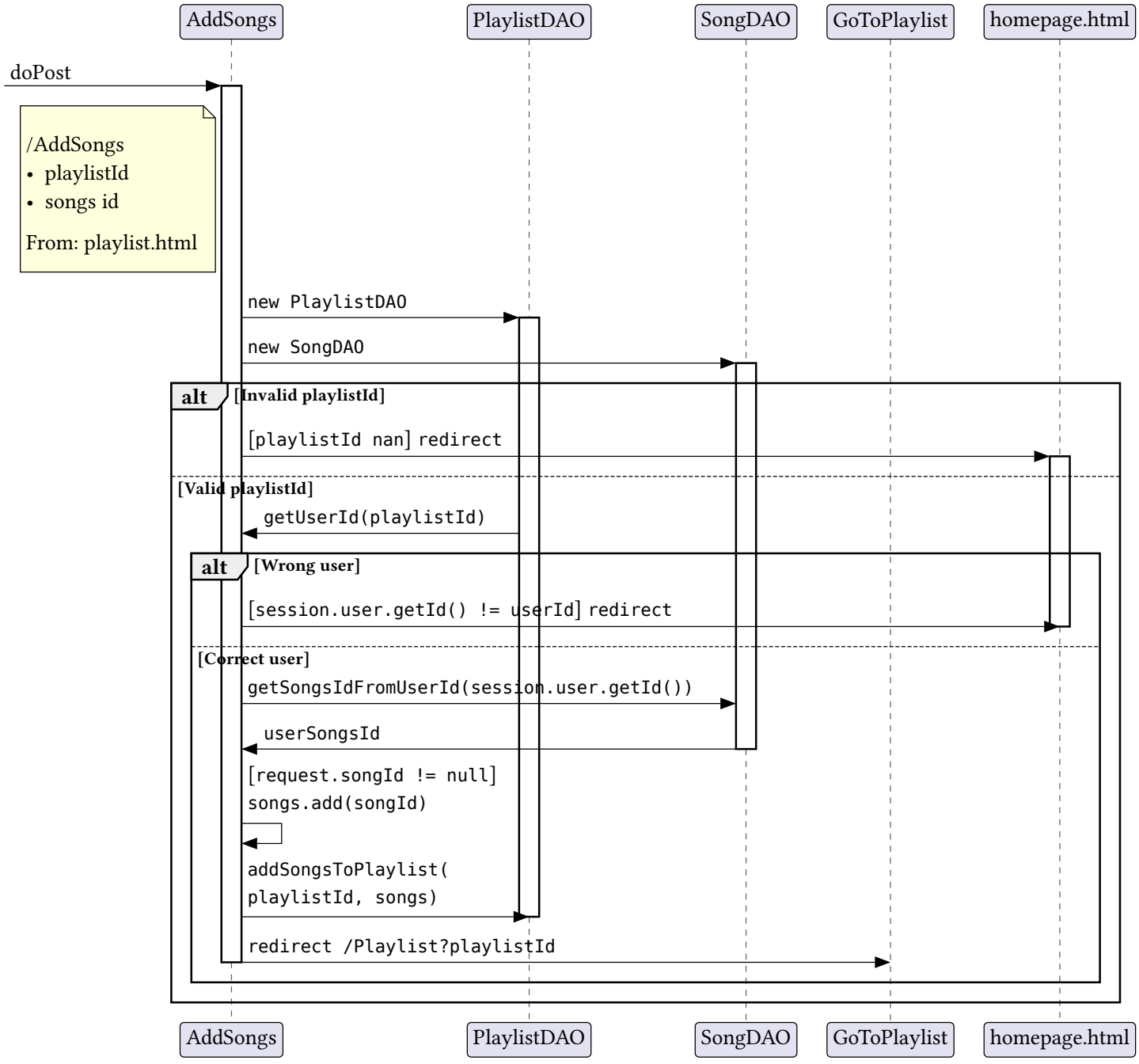
- **Recuperare un file**



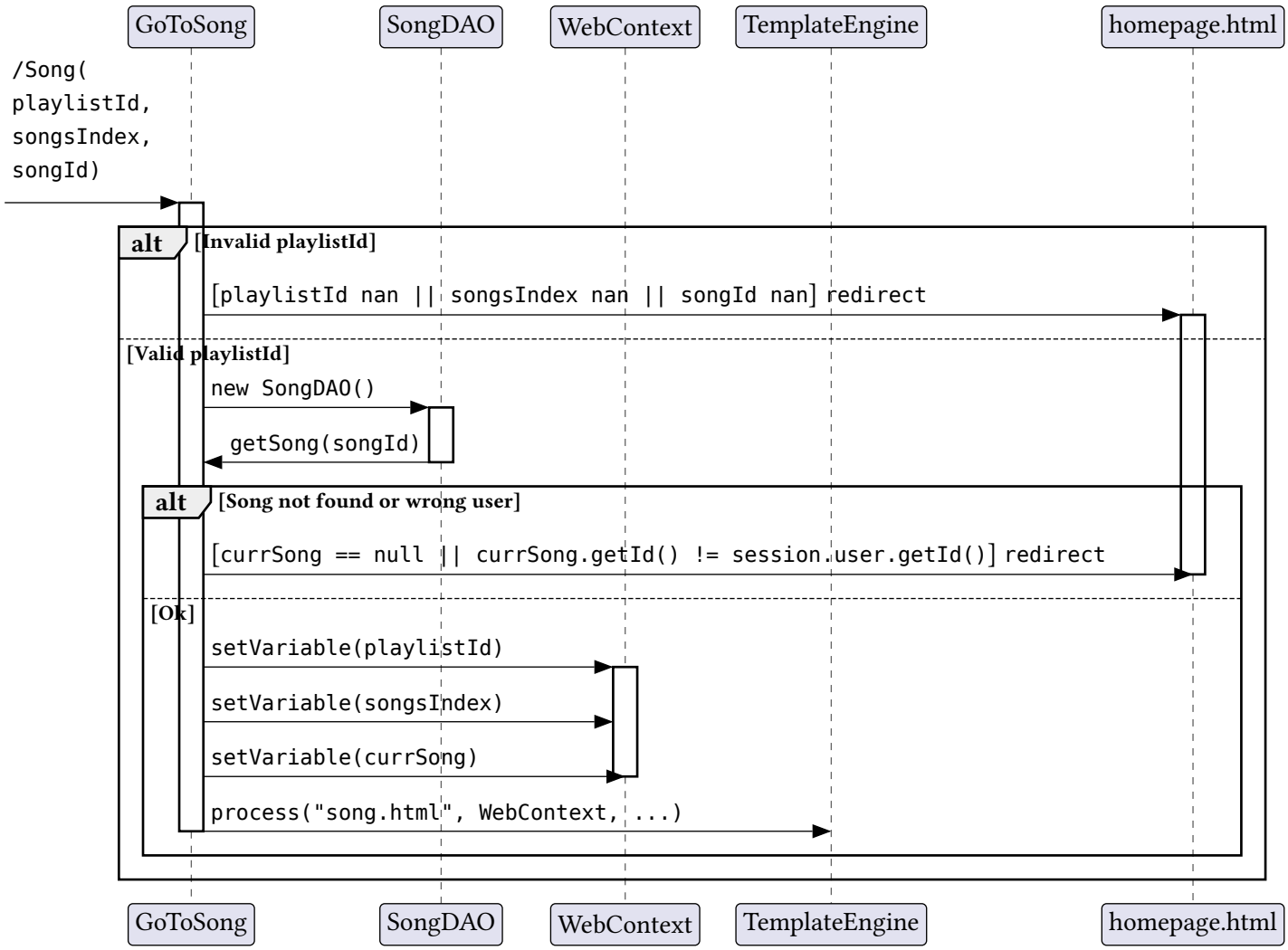
- **Andare alla pagina della playlist**



- **Aggiungere canzoni alla playlist**



- **Andare alla pagina della canzone**



3. Documentazione ver. Javascript

3.a. Analisi requisiti dati

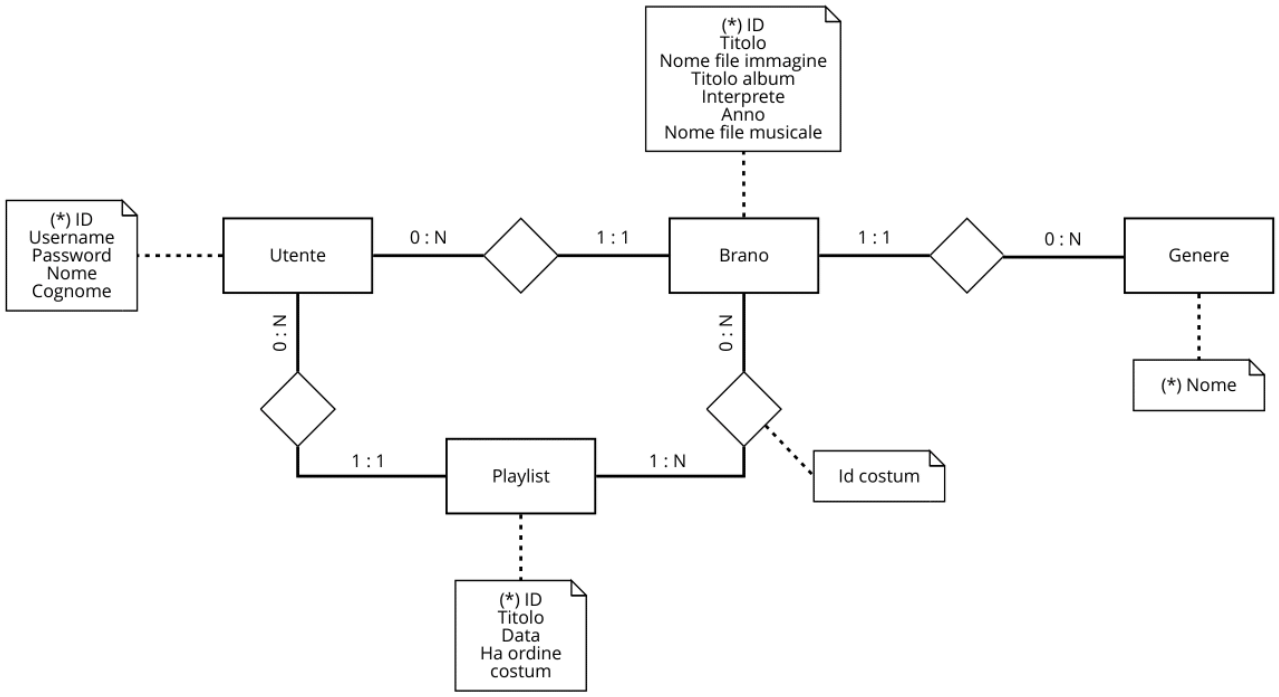
[...]

- L'applicazione deve consentire all'utente di *riordinare le playlist* con un criterio personalizzato diverso da quello di default. Dalla HOME con un link associato a ogni playlist si accede a una finestra modale RIORDINO, che mostra la lista completa dei brani della playlist ordinati secondo il criterio corrente (personalizzato o di default). L'utente può 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. Un brano aggiunto a una playlist con ordinamento personalizzato è inserito nell'ultima posizione.

Legenda:

- **Entità**;
- *Attributi*;
- Relazioni.

3.a.a. Diagramma entità-relazioni



3.a.b. Database design

```sql
create table users
(
    id       int auto_increment,
    username varchar(32) not null unique,
    password varchar(32) not null,
    name     varchar(32) not null,
    surname  varchar(32) not null,
    primary key (id)
);

create table genres
(
    name varchar(32),
    primary key (name)
```

```
);

create table songs
(
    id              int auto_increment,
    user_id         int          not null,
    title           varchar(256) not null,
    image_file_name varchar(256) not null,
    album_title     varchar(256) not null,
    performer       varchar(256) not null,
    year            int          not null check ( year > 0 ),
    genre           varchar(256) not null,
    music_file_name varchar(256) not null,
    primary key (id),
    foreign key (user_id) references users (id) on update cascade on delete no action,
    foreign key (genre) references genres (name) on update cascade on delete no action,
    unique (user_id, music_file_name),
    unique (user_id, title)
);

create table playlists
(
    id               int auto_increment,
    user_id          int          not null,
    title            varchar(256) not null,
    date             date         not null default current_date,
    has_custom_order boolean      not null default false,
    primary key (id),
    unique (user_id, title)
);

create table playlist_contents
(
    playlist  int,
    song      int,
    custom_id int default null,
    primary key (playlist, song),
    foreign key (playlist) references playlists (id) on update cascade on delete no
action,
    foreign key (song) references songs (id) on update cascade on delete no action
);
```

3.b. Analisi requisiti d'applicazione

[...]

Si realizzi un'applicazione client server web che modifica le specifiche precedenti come segue:

- 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 personalizzato diverso da quello di default. Dalla HOME con un link associato a ogni playlist si accede a una finestra modale RIORDINO, che mostra la lista completa dei brani della playlist ordinati secondo il criterio corrente (personalizzato o di default). L'utente può 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. Un brano aggiunto a una playlist con ordinamento personalizzato è inserito nell'ultima posizione.

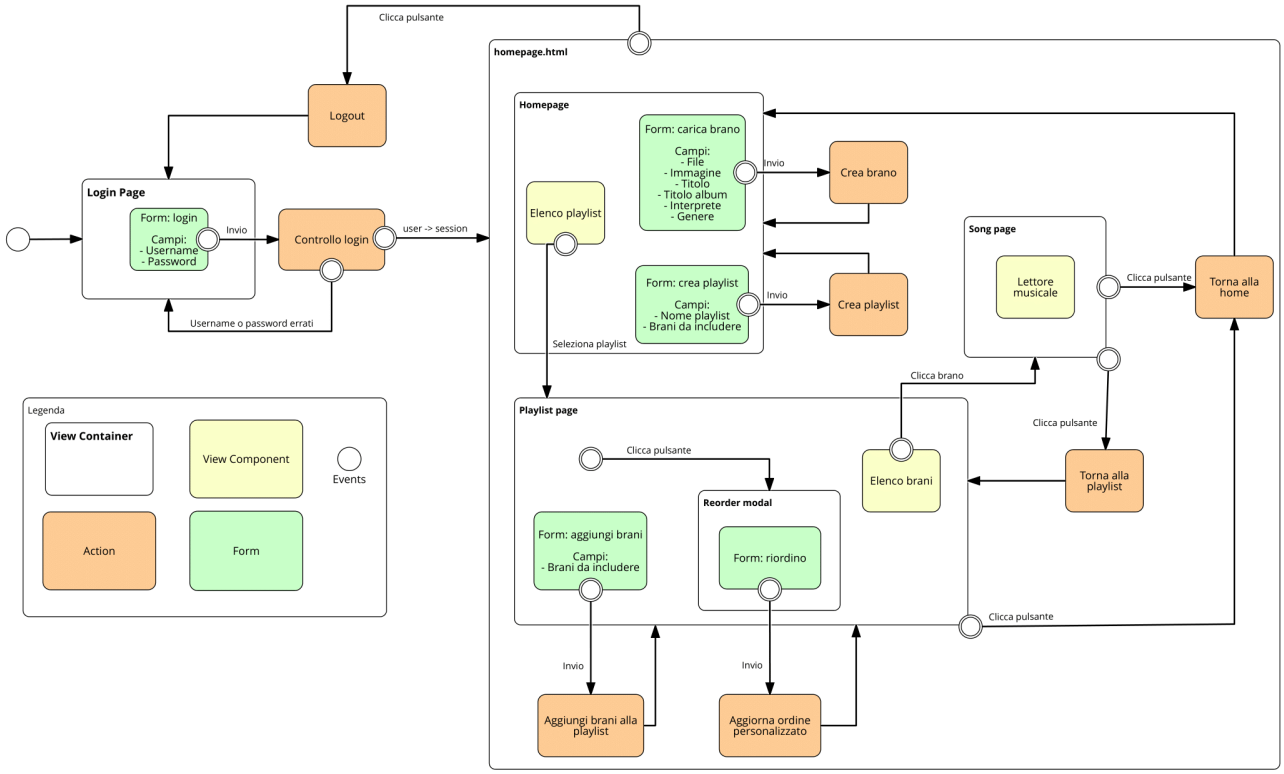
Legenda:

- Pagine;
- Componenti;
- Eventi;
- Azioni.

3.b.a. Aggiunta alle specifiche

- Funzione di logout, accessibile tramite un pulsante dalle pagine di home, playlist e canzone;
- Funzione per tornare alla homepage, accessibile tramite un pulsante dalle pagine di playlist e canzone;
- Funzione per tornare alla pagina della playlist originale, accessibile tramite un pulsante dalla pagina della canzone;
- Messaggio di «benvenuto» quando l'utente è nella home page;
- Nella pagina della playlist viene mostrata la data in cui è stata creata;
- Possibilità di creare una playlist senza brani;
- Istruzioni sul creare una playlist nella homepage se l'utente non ha playlist associate;
- Istruzioni sull'aggiungere brani alla playlist se vuota nella pagina della playlist;
- Istruzioni sul caricare canzoni nella pagina della playlist se l'utente o ha aggiunto tutti i suoi brani alla playlist o non ha brani associati;
- Possibilità di annullare il riordino di una playlist cliccando l'apposito pulsante o al di fuori del modal.

3.b.b. Diagramma IFML



3.b.c. Eventi e azioni

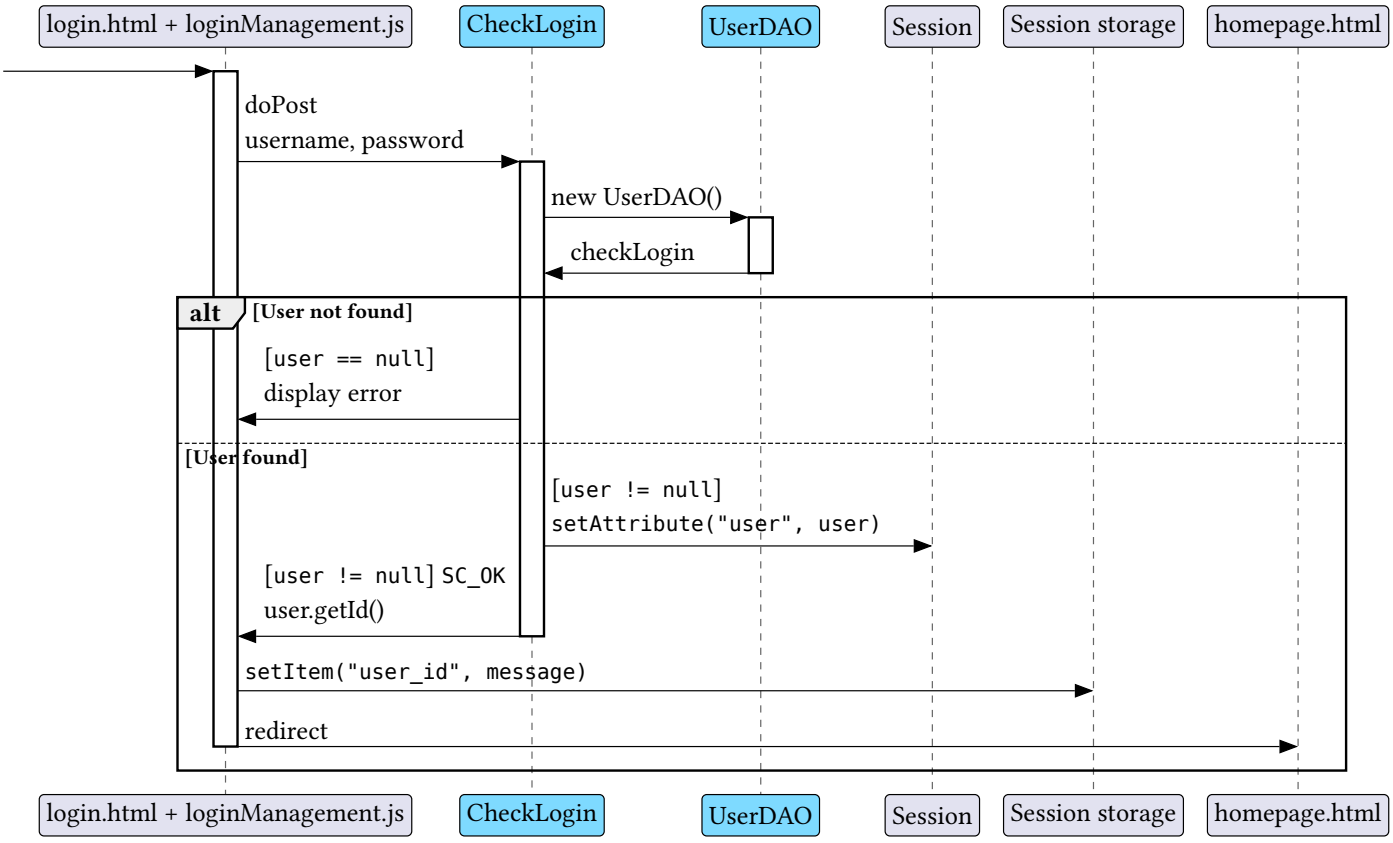
Client side		Server side	
Evento	**Azione**	**Evento**	**Azione**
Index ⇒ login form ⇒ submit	Controllo dati	POST: username, password	Controllo credenziali
Homepage ⇒ primo caricamento	Aggiorna view	GET	Recupera informazioni utente e generi
Homepage ⇒ primo caricamento o creazione playlist	Aggiorna elenco playlist	GET	Recupera playlist dell'utente
Homepage ⇒ primo caricamento o aggiunta brano	Aggiorna create playlist form	GET	Recupera canzoni dell'utente
Homepage ⇒ upload song form ⇒ submit	Controllo dati	POST: titolo, album, interprete, anno, genere, file musicale, immagine	Aggiunta brano
Homepage ⇒ create playlist form ⇒ submit	Controllo dati	POST: titolo, elenco brani da aggiungere	Creazione playlist
Homepage ⇒ elenco playlist ⇒ seleziona playlist	Aggiorna view e visualizza playlist page	GET: playlistId	Recupero canzoni presenti nella playlist e non
Playlist page ⇒ add songs form ⇒ submit	Controllo dati	POST: elenco brani da aggiungere	Aggiunta brani alla playlist
Playlist page ⇒ riordino	Aggiorna view e mostra modal di riordino	GET: playlistId	Recupero brani presenti nella playlist
Modal riordino ⇒ riordino ⇒ submit	Controllo dati	POST: brani ordinati	Salva l'ordine personalizzato
Modal riordino ⇒ annulla riordino/click fuori dal modal	Nascondi modal di riordino	-	-
Playlist page ⇒ successive/precedenti canzoni ⇒ submit	Aggiorna view	-	-
Playlist page ⇒ elenco brani ⇒ seleziona brano	Aggiorna view e visualizza song page	GET: songId	Recupero informazioni brano
Playlist/song page ⇒ homepage ⇒ submit	Aggiorna view e visualizza homepage	-	-
Song page ⇒ torna alla playlist ⇒ submit	Aggiorna view e visualizza playlist page	-	-
Logout	-	GET	Terminazione sessione

3.b.d. Controller ed Event handler

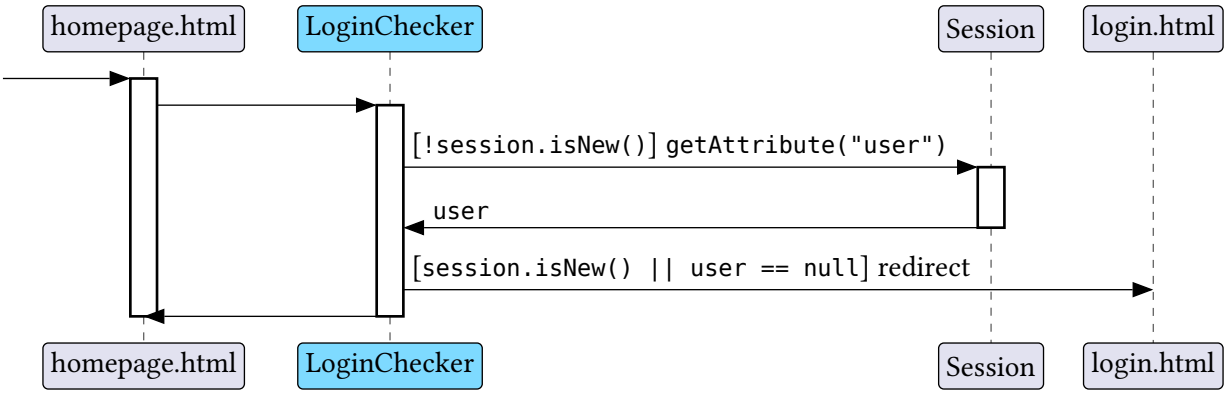
Client side		Server side	
Evento	**Controller**	**Evento**	**Controller**
Index ⇒ login form ⇒ submit	Funzione `makeCall`	POST: username, password	Servlet `CheckLogin`
Homepage ⇒ primo caricamento	Funzione `homepageInit`	GET	Servlet `GetUser` e `GetGenres`
Homepage ⇒ primo caricamento o creazione playlist	Funzione `updatePlaylists`	GET	Servlet `GetPlaylists`
Homepage ⇒ primo caricamento o aggiunta brano	Funzione `updCreatePlaylistForm`	GET	Servlet `GetSongsByUserID`
Homepage ⇒ upload song form ⇒ submit	Funzione `makeCall`	POST: titolo, album, interprete, anno, genere, file musicale, immagine	Servlet `UploadSong`
Homepage ⇒ create playlist form ⇒ submit	Funzione `makeCall`	POST: titolo, elenco brani da aggiungere	Servlet `CreatePlaylist`
Homepage ⇒ elenco playlist ⇒ seleziona playlist	Funzione `playlistPageInit`	GET: playlistId	Servlet `GetPlaylist`, `GetSongsFromPlaylist` e `GetSongsNotInPlaylist`
Playlist page ⇒ add songs form ⇒ submit	Funzione `makeCall`	POST: elenco brani da aggiungere	Servlet `AddSongsToPlaylist`
Playlist page ⇒ riordino	Funzione `showReorderPage`	GET: playlistId	Servlet `GetSongsFromPlaylist`
Modal riordino ⇒ riordino ⇒ submit	Funzione `makeCall`	POST: brani ordinati	Servlet `UpdateCustomOrder`
Modal riordino ⇒ annulla riordino/click fuori dal modal	Funzione `closeModal`	-	-
Playlist page ⇒ successive/precedenti canzoni ⇒ submit	Funzione `showVisibleSongs`	-	-
Playlist page ⇒ elenco brani ⇒ seleziona brano	Funzione `showSongPage`	GET: songId	Servlet `GetSong`
Playlist/song page ⇒ homepage ⇒ submit	Funzione `showHomepage`	-	-
Song page ⇒ torna alla playlist ⇒ submit	Funzione `showPlaylistPage`	-	-
Logout	-	GET	Servlet `Logout`

3.b.e. Sequence diagrams

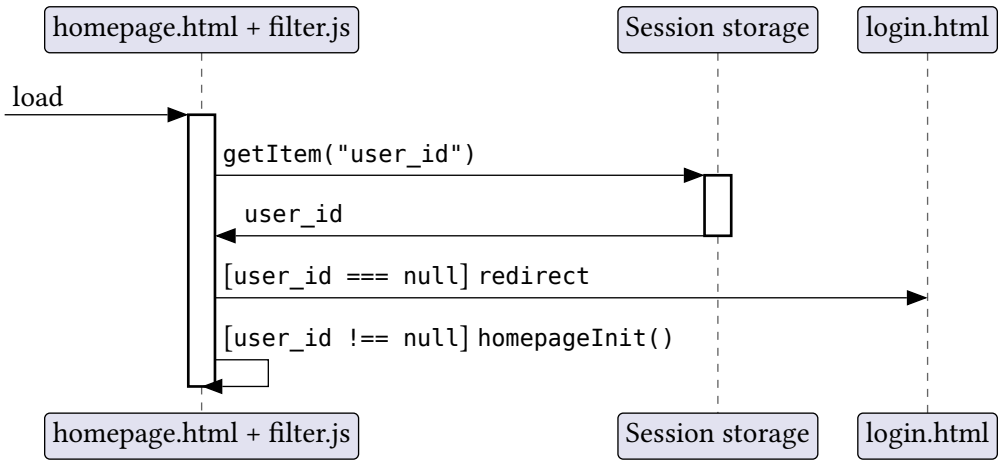
- **Login**



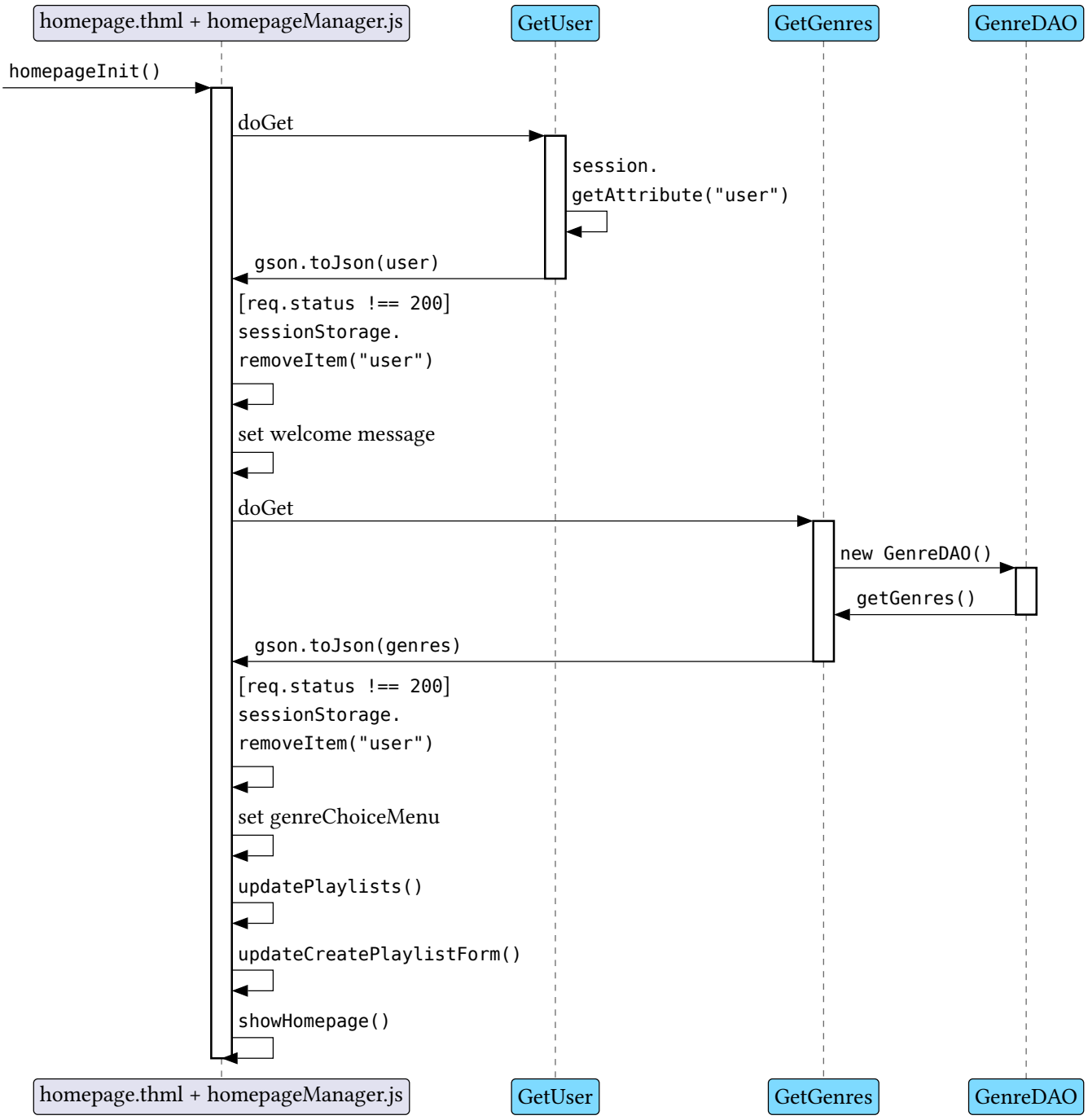
- **Filtro utente**



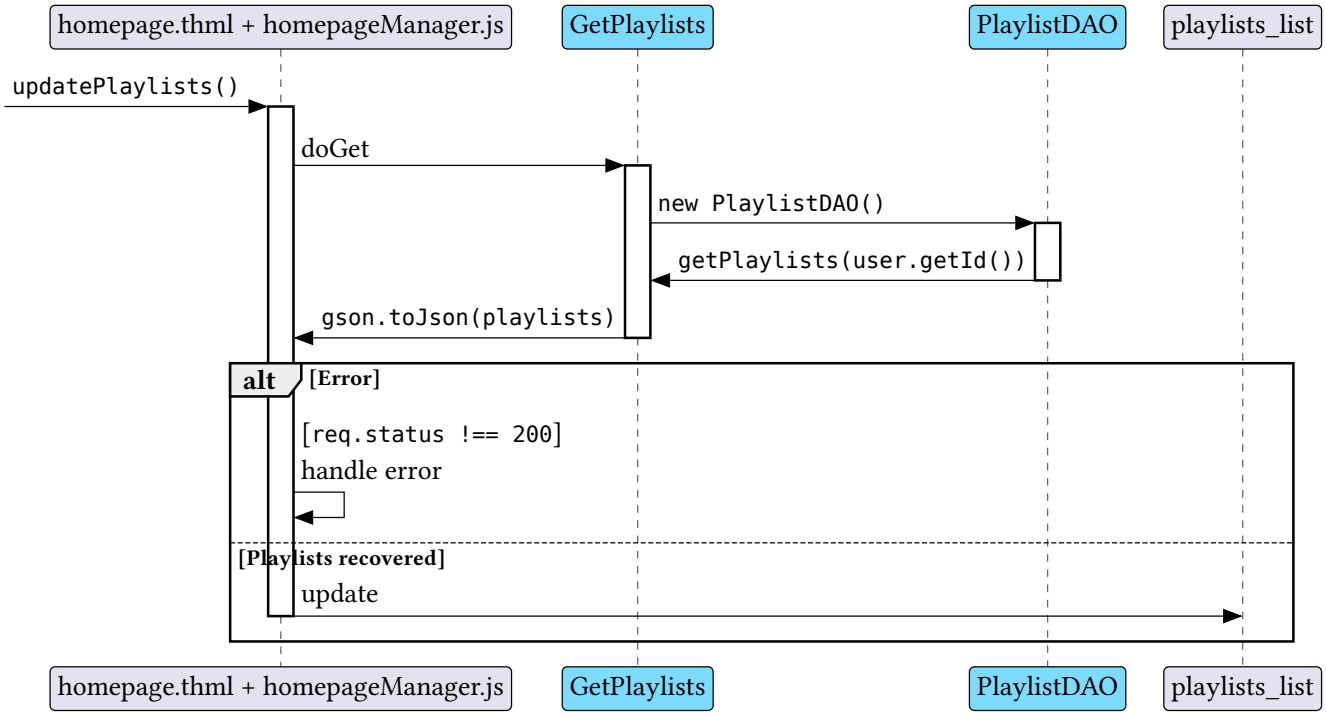
- **Caricare la homepage**



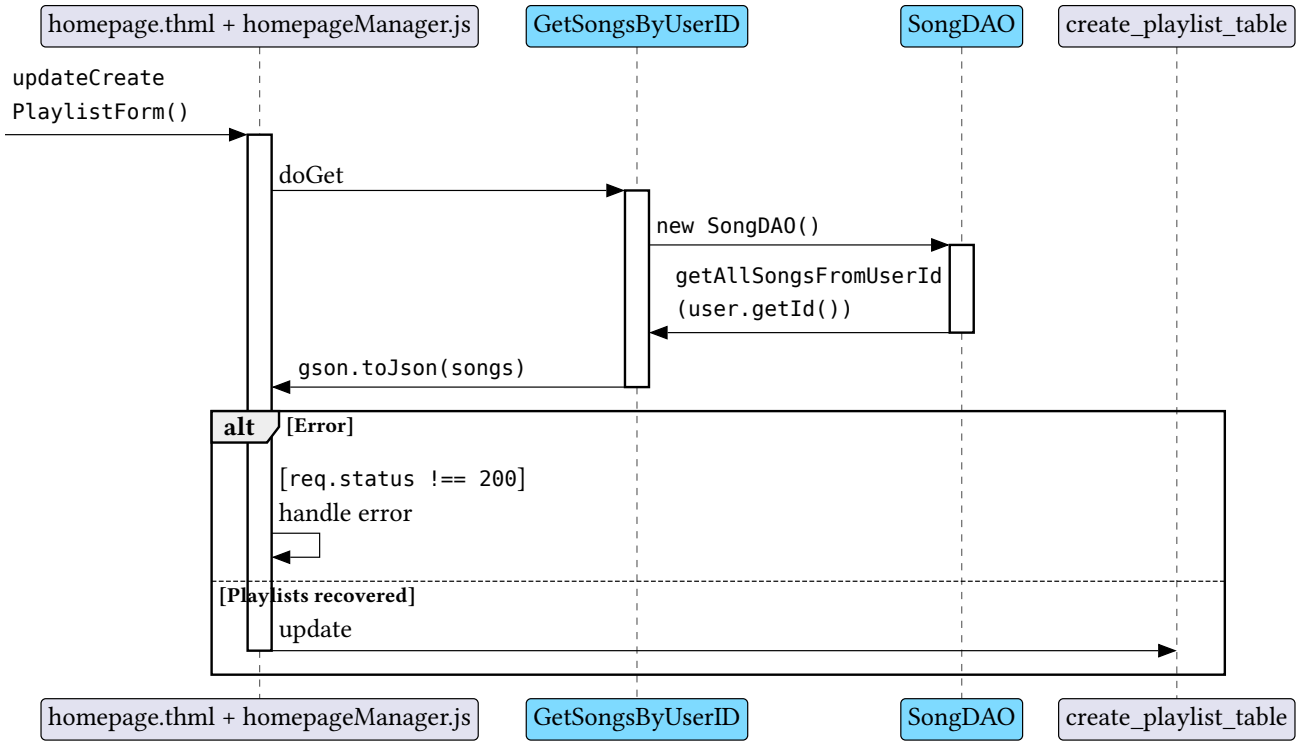
- **Inizializzare la homepage**



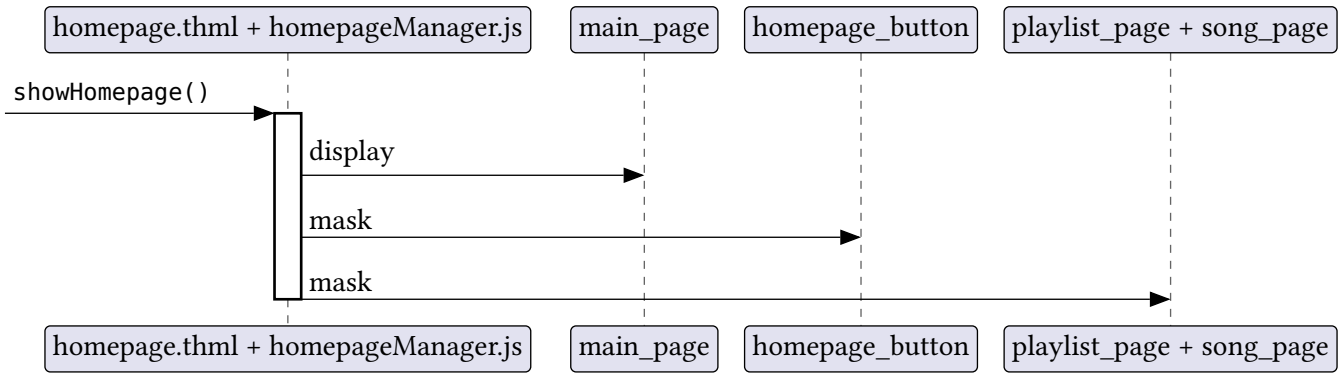
- **Aggiornare la lista di playlist**



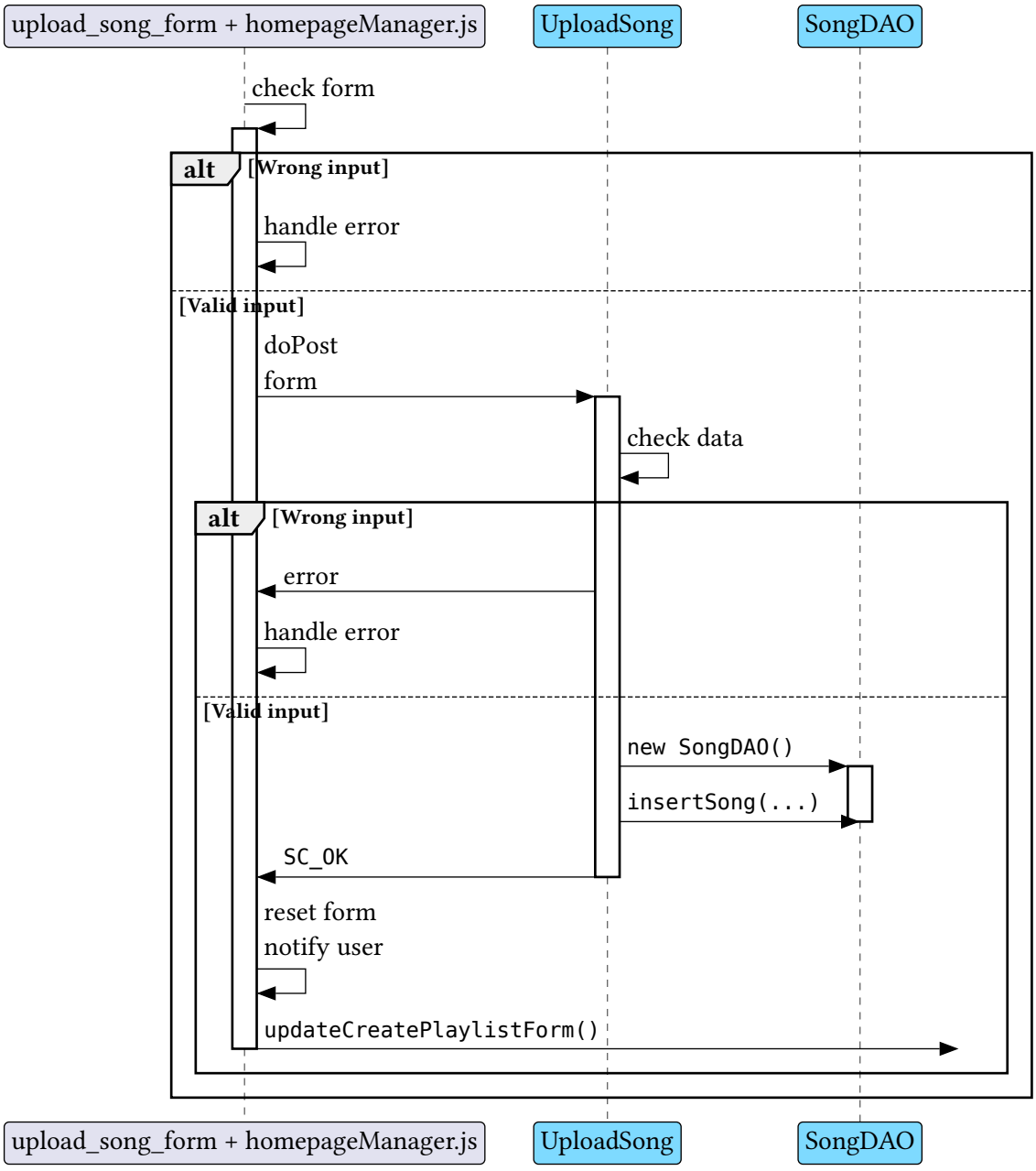
- **Aggiornare il form per creare una playlist**



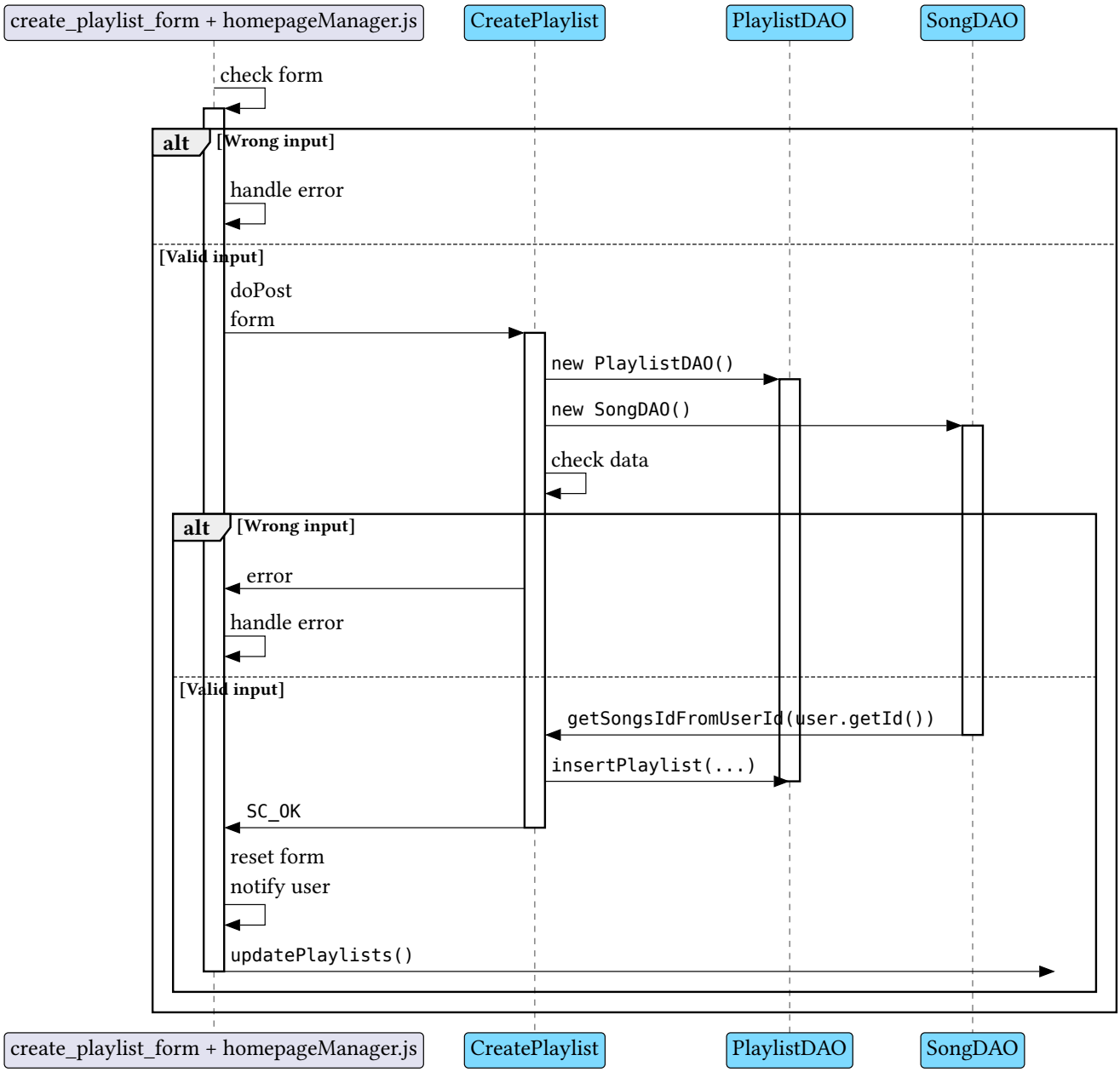
- **Andare alla homepage**



- **Caricare una canzone**

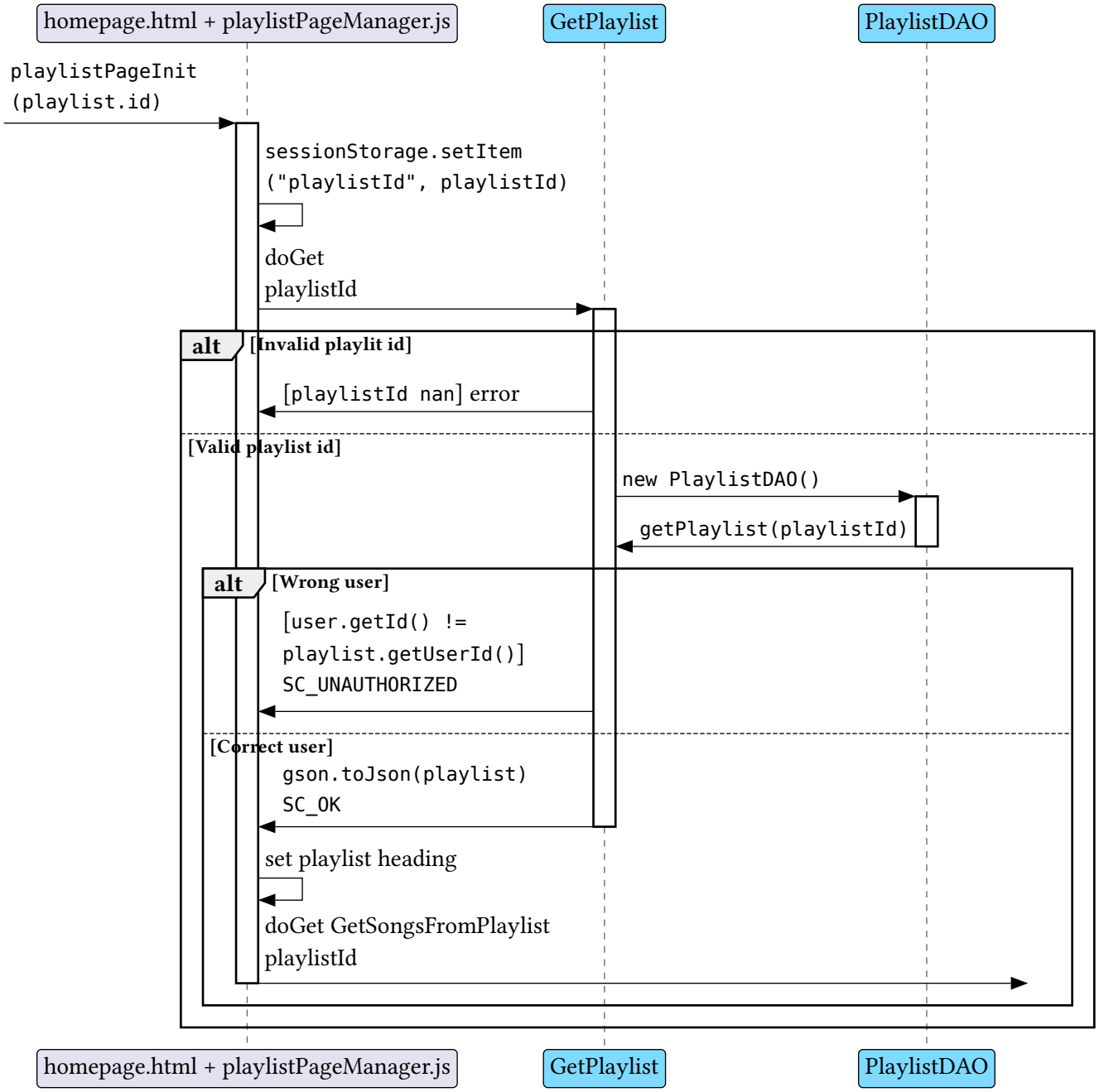


- **Creare una playlist**

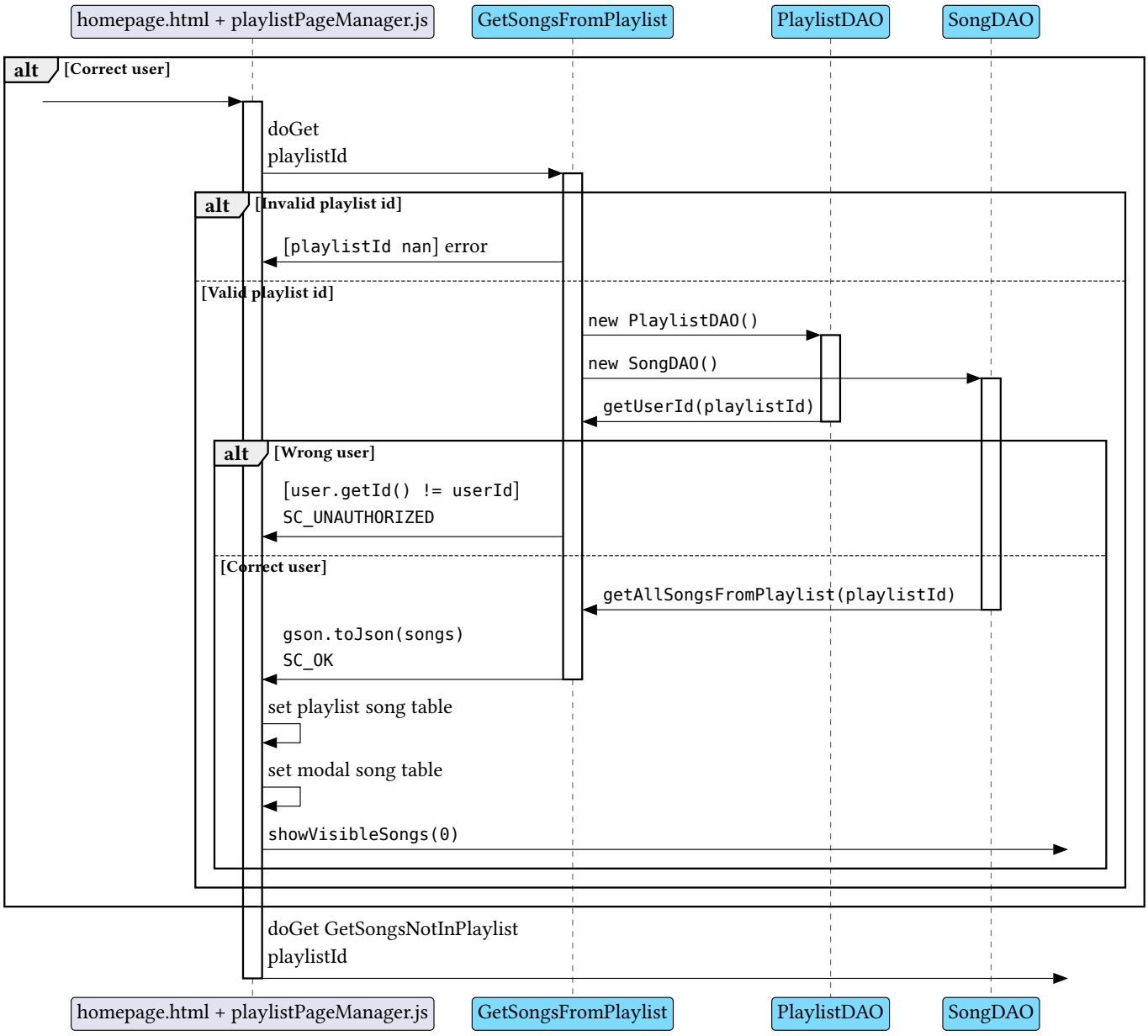


- **Inizializzare la pagina della playlist**

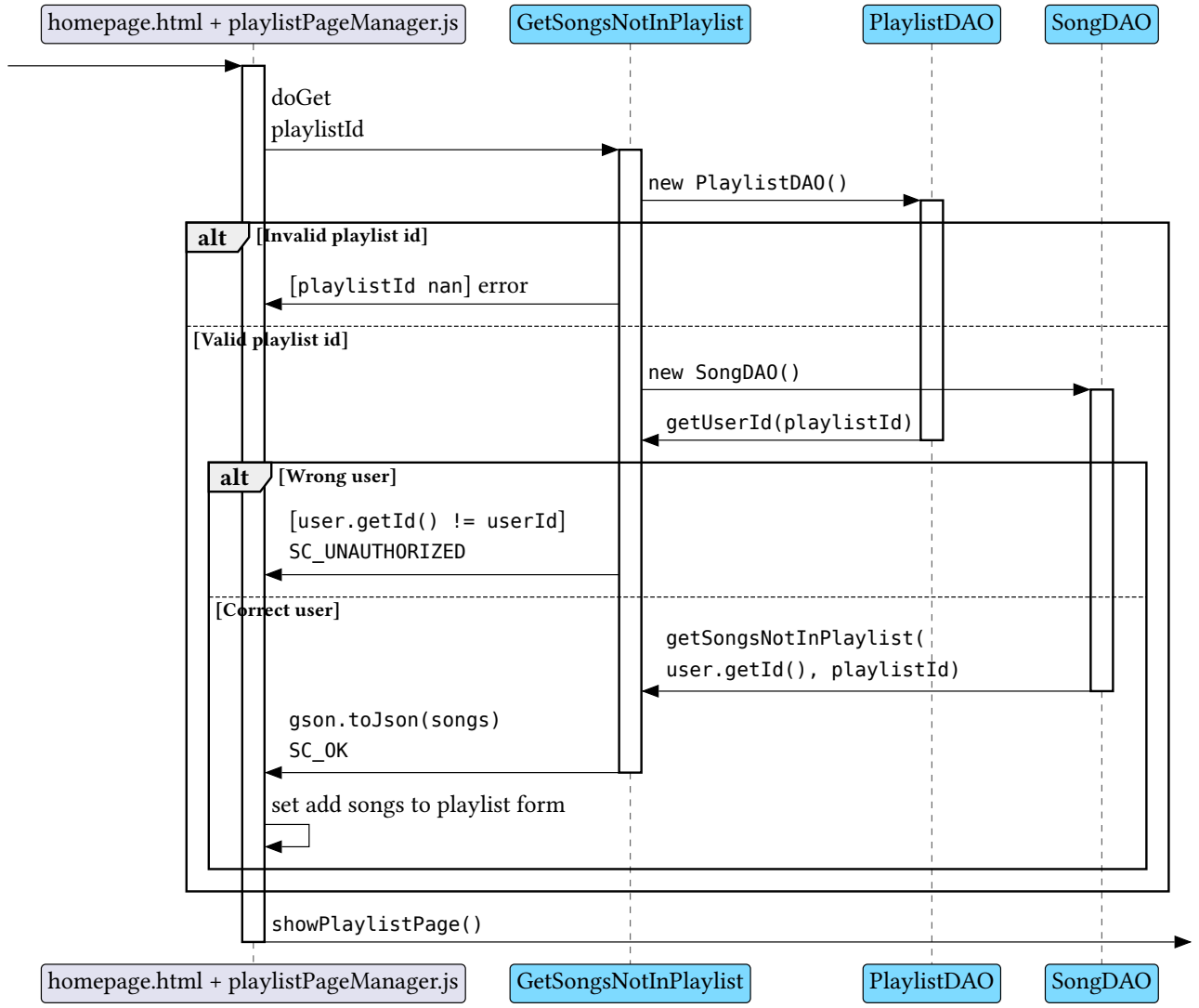
Parte 1



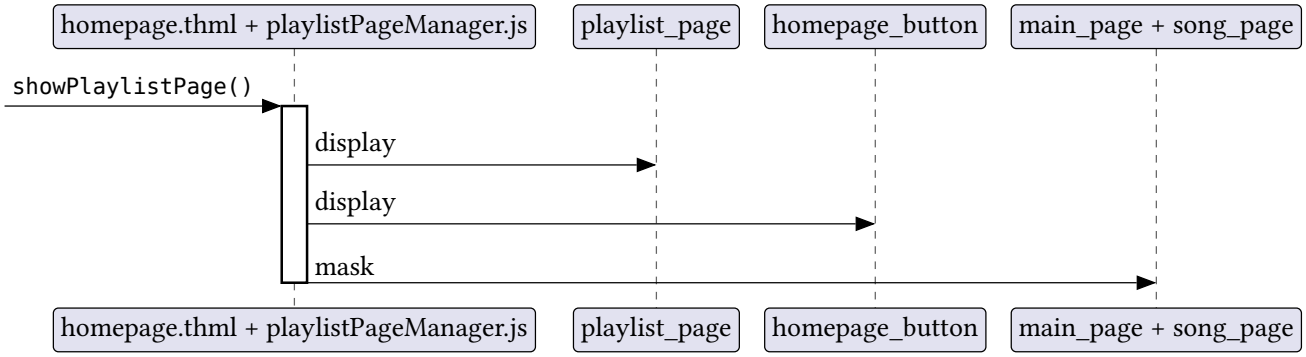
Parte 2


homepage.html + playlistPageManager.js
GetSongsFromPlaylist
PlaylistDAO
SongDAO
alt
[Correct user]
doGet
playlistId
alt
[Invalid playlist id]
[playlistId nan] error
[Valid playlist id]
new PlaylistDAO()
new SongDAO()
getUserId(playlistId)
alt
[Wrong user]
[user.getId() != userId]
SC_UNAUTHORIZED
[Correct user]
getAllSongsFromPlaylist(playlistId)
gson.toJson(songs)
SC_OK
set playlist song table
set modal song table
showVisibleSongs(0)
doGet GetSongsNotInPlaylist
playlistId
homepage.html + playlistPageManager.js
GetSongsFromPlaylist
PlaylistDAO
SongDAO

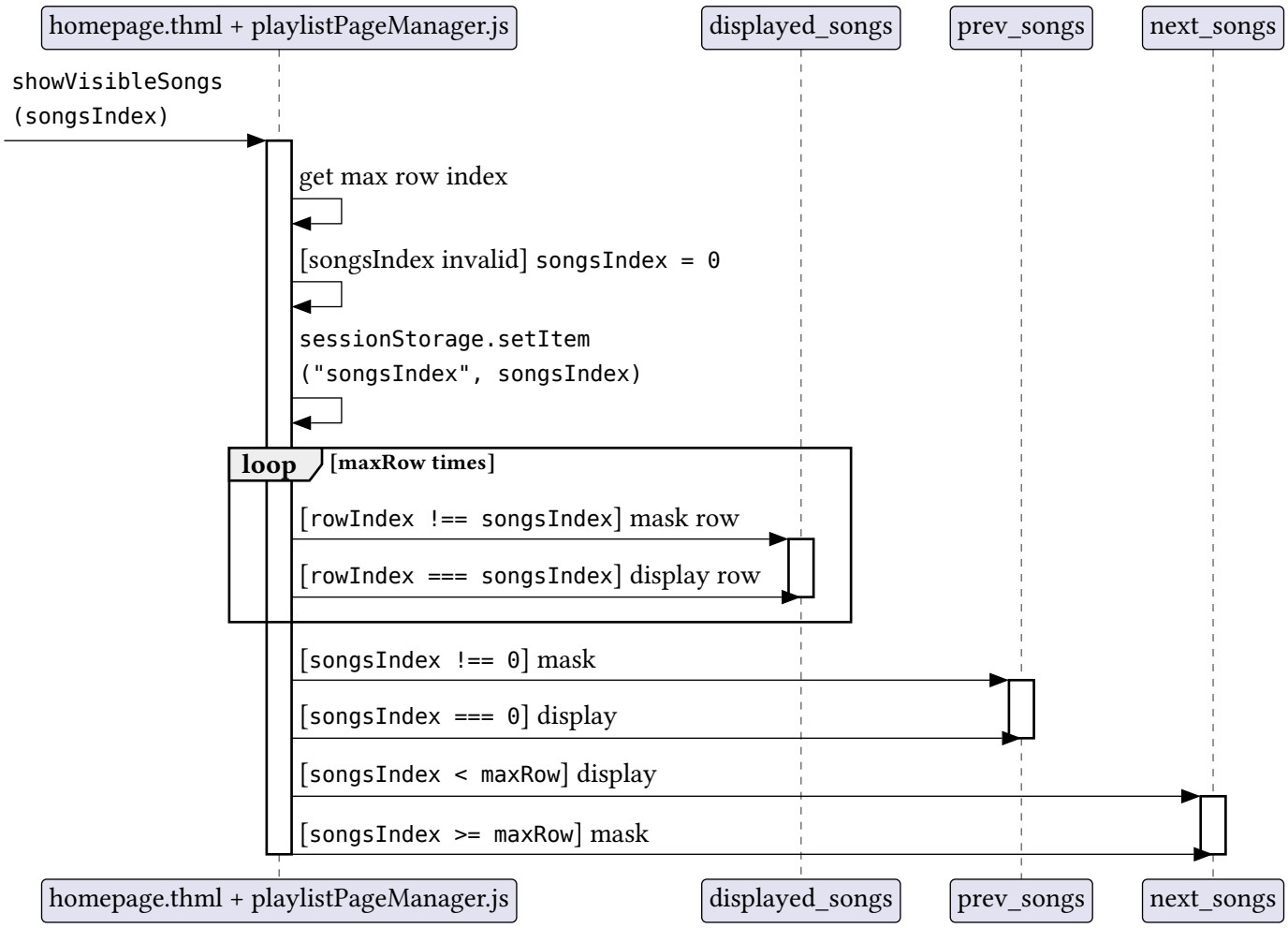
Parte 3



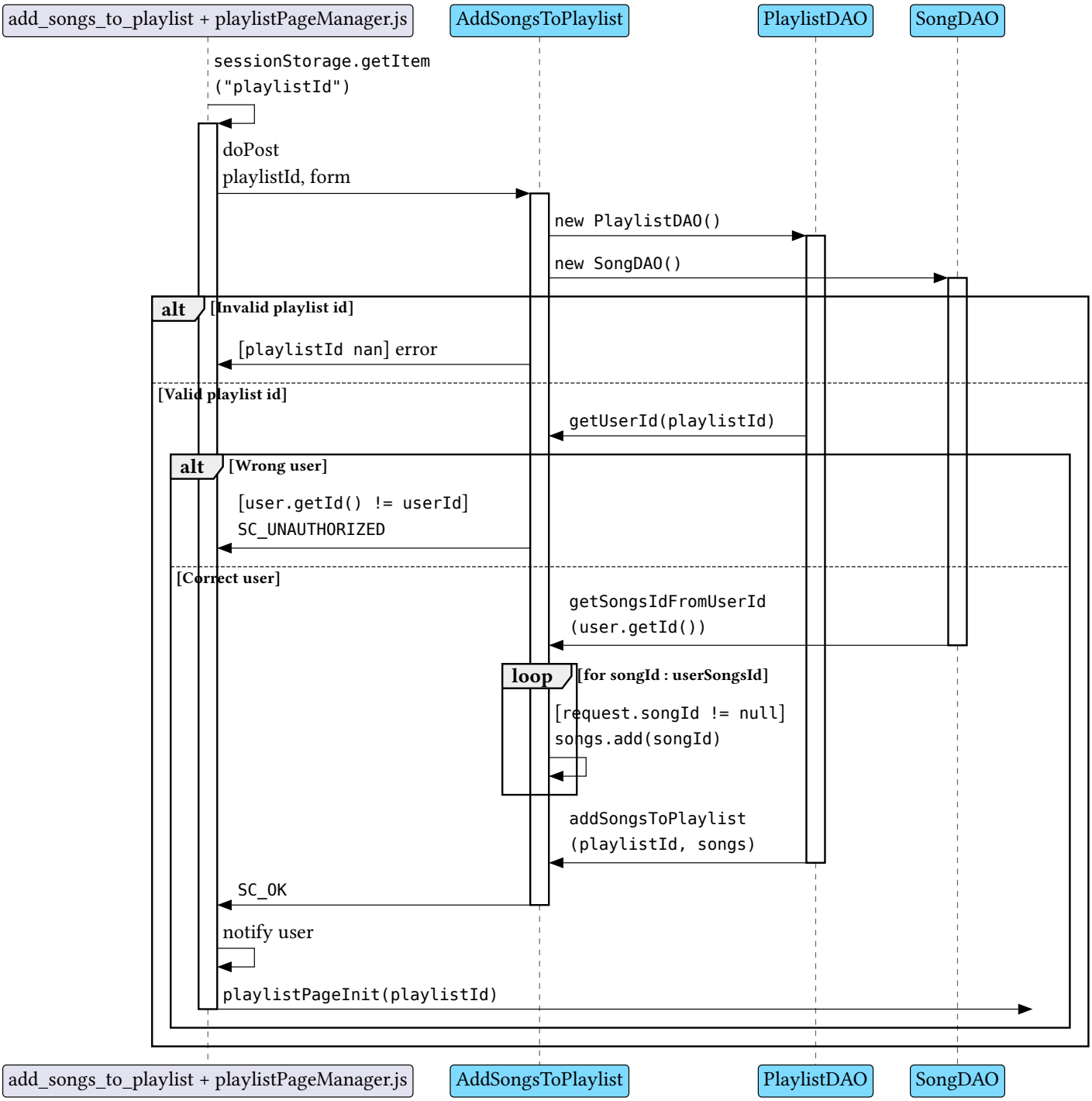
- **Andare alla pagina della playlist**



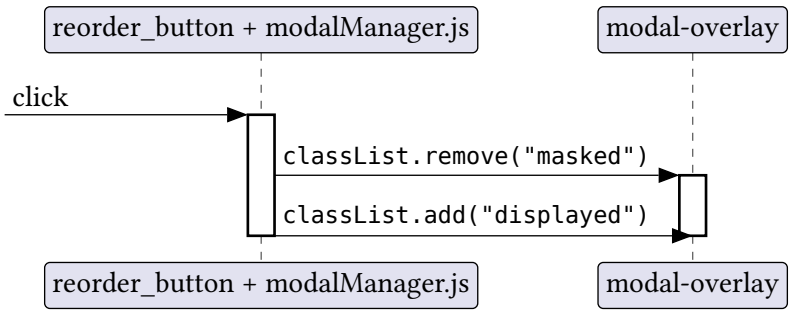
- **Visualizzare brani precedenti/successivi**



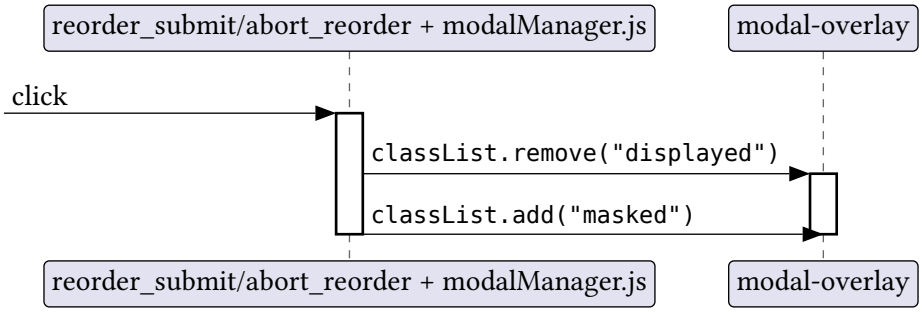
- **Aggiungere brani alla playlist**



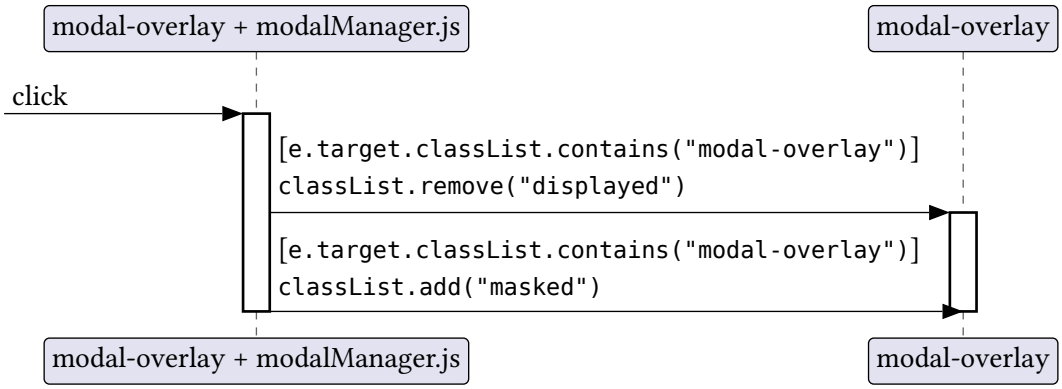
- **Aprire il modal**



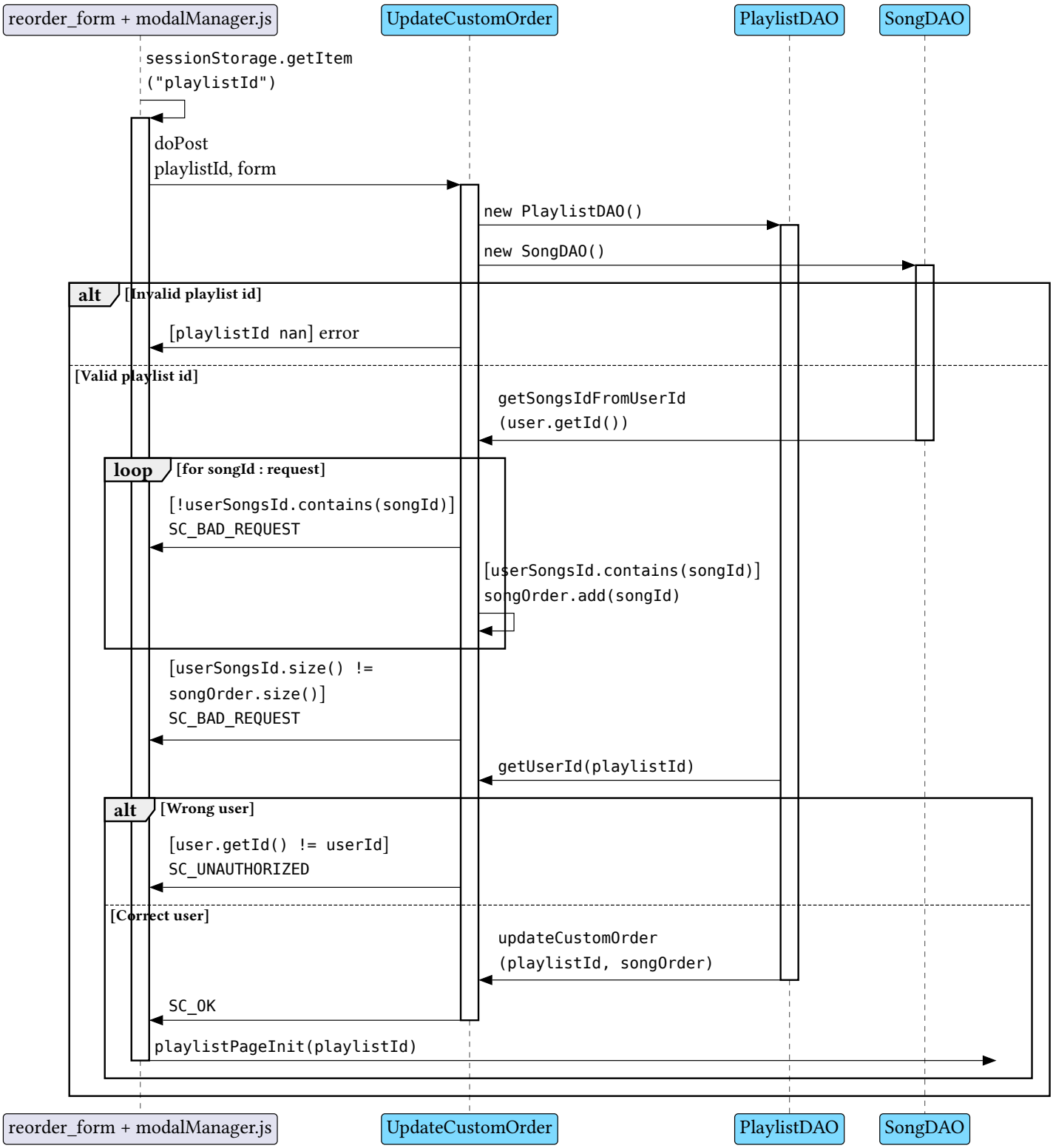
- **Chiudere il modal**



- **Chiudere il modal cliccando fuori dal form**

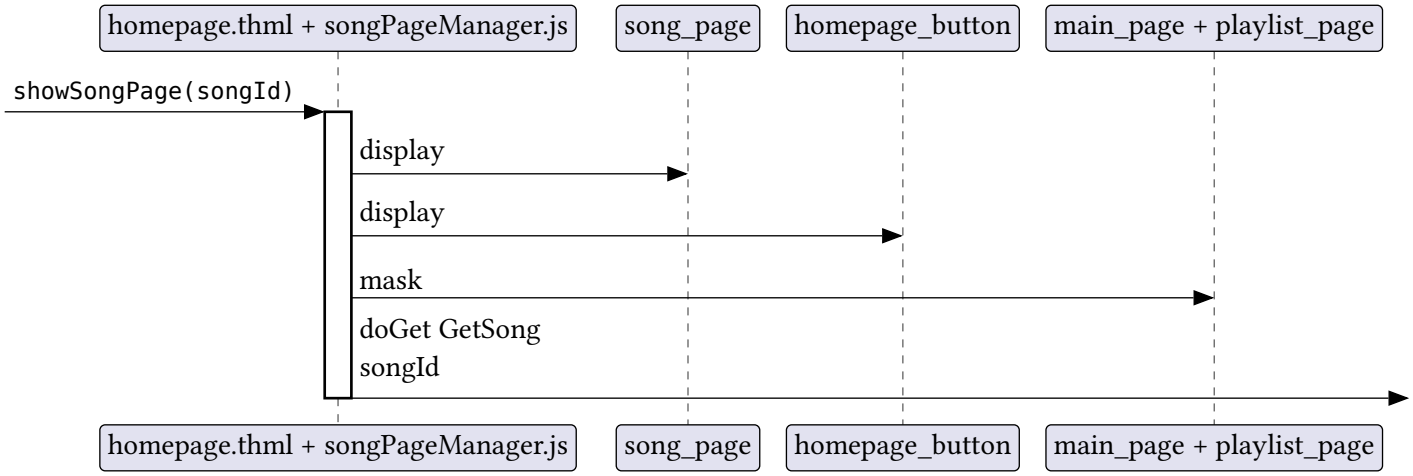


- **Cambiare ordinamento della playlist**



- **Andare alla pagina della canzone**

Parte 1



Parte 2