

Progetto di Tecnologie Informatiche per il Web

Michele Sangaletti

Indice

Progetto di Tecnologie Informatiche per il Web	1
Traccia	3
Documentazione ver. html pura	5
Analisi requisiti dati	5
Design database	5
Analisi requisiti d'applicazione	7
Aggiunta alle specifiche	7
Diagramma IFL	8
Sequence diagrams	8
Documentazione ver. Javascript	16
Analisi requisiti dati	16
Design database	16
Analisi requisiti d'applicazione	18
Aggiunta alle specifiche	19
Diagramma IFL	19
Eventi e azioni	19
Controller ed Event handler	20
Sequence diagram	20

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.

Documentazione ver. html pura

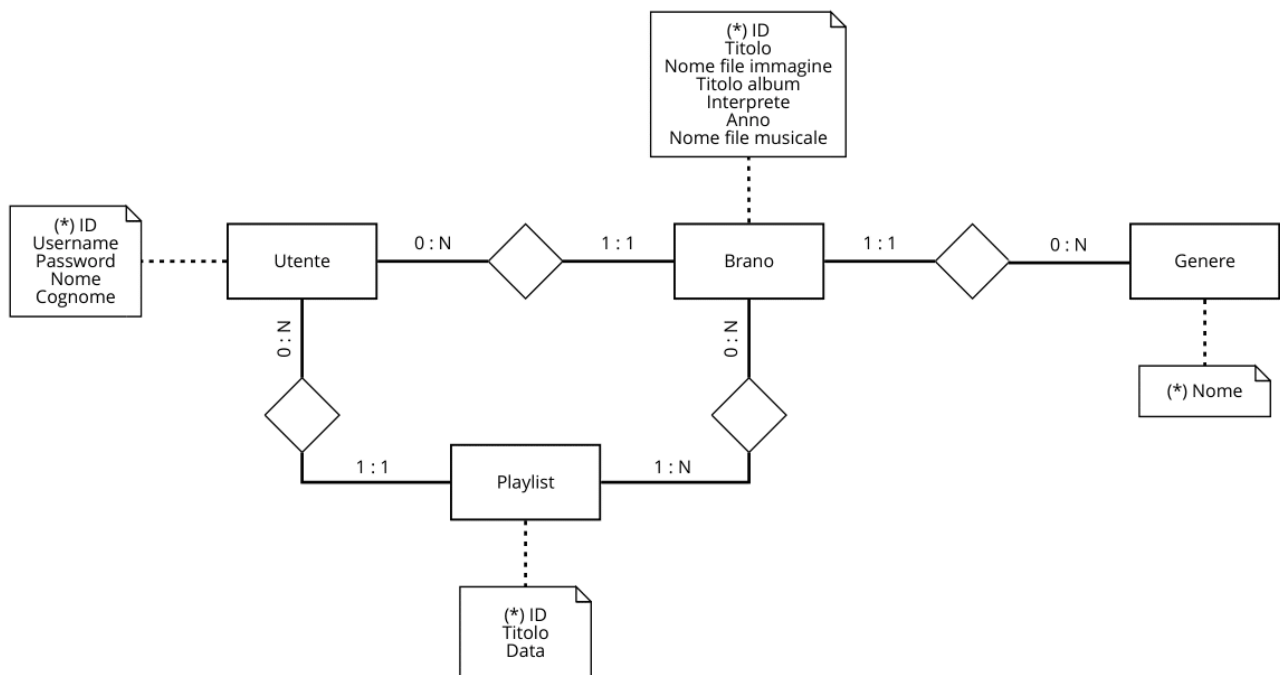
Analisi requisiti dati

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.

Design database



```
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(64) not null,
    image_file_name   varchar(64) not null,
    album_title       varchar(64) not null,
    performer         varchar(64) not null,
    year              int      not null,
    genre             varchar(64) not null,
    music_file_name   varchar(128) not null,
    primary key (id),
    foreign key (user_id) references users (id),
    foreign key (genre) references genres (name),
    unique (user_id, music_file_name),
    unique (user_id, title)
);

create table playlists
(
    id                int auto_increment,
    user_id           int      not null,
    title             varchar(64) not null,
    date              date      not null,
    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),
    foreign key (song) references songs (id)
);

```

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.

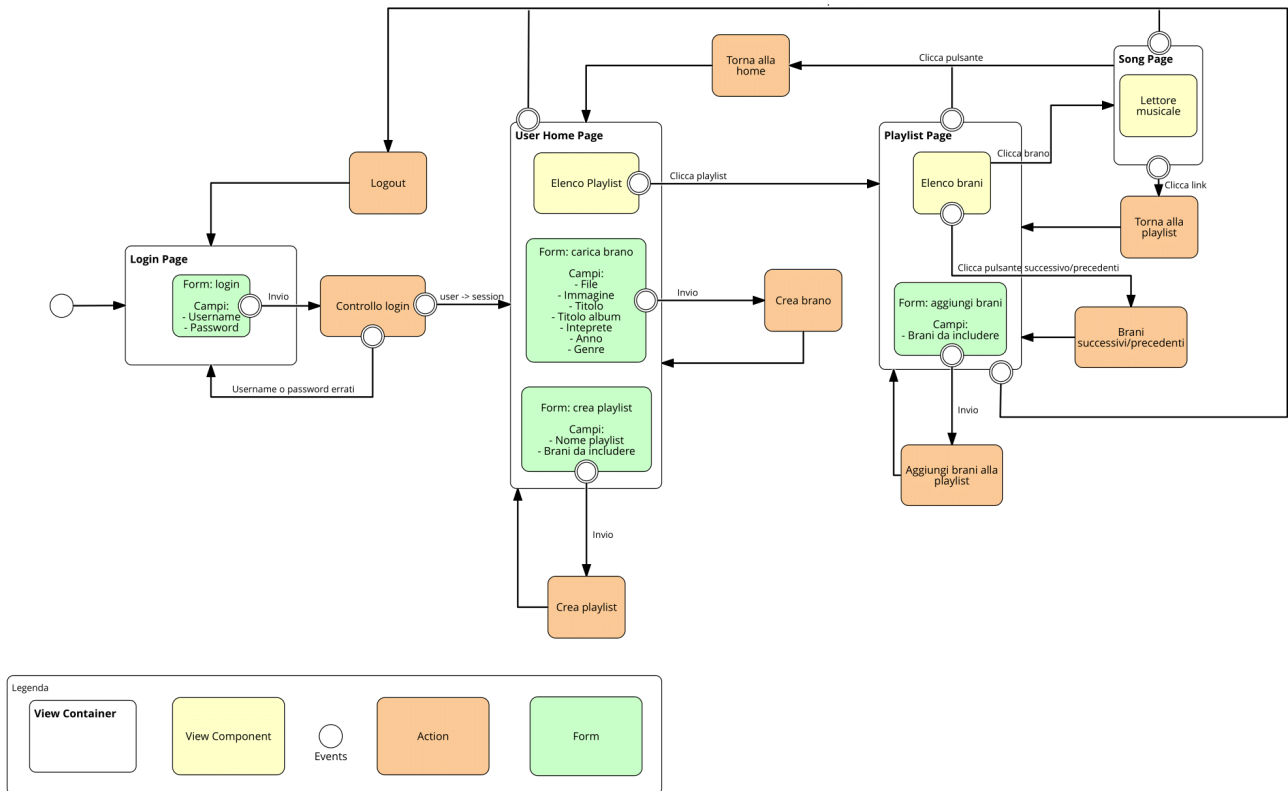
Legenda:

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

Aggiunta alle specifiche

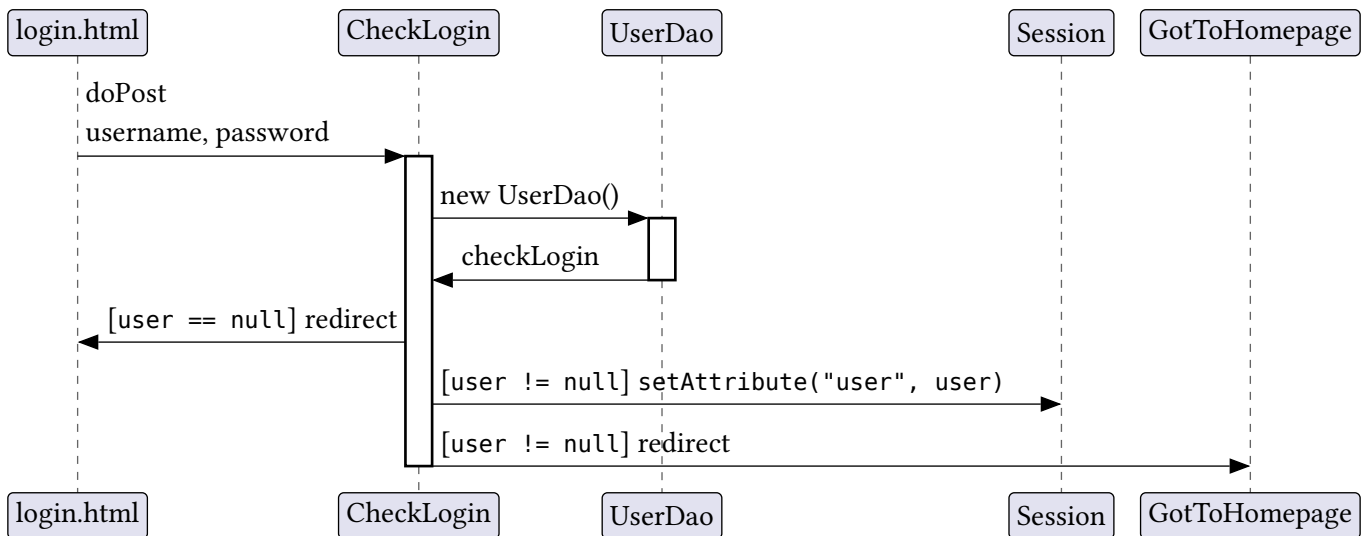
- Funzione di logout, accessibile tramite un pulsante dalle pagine di home, playlist e canzone;
- Funzione per tornare alla homepage, accessibile tramite un pulsante dalle pagine di playlist e canzone;
- Funzione per tornare alla pagina della playlist originale, accessibile tramite un link dalla pagina 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.

Diagramma IFL

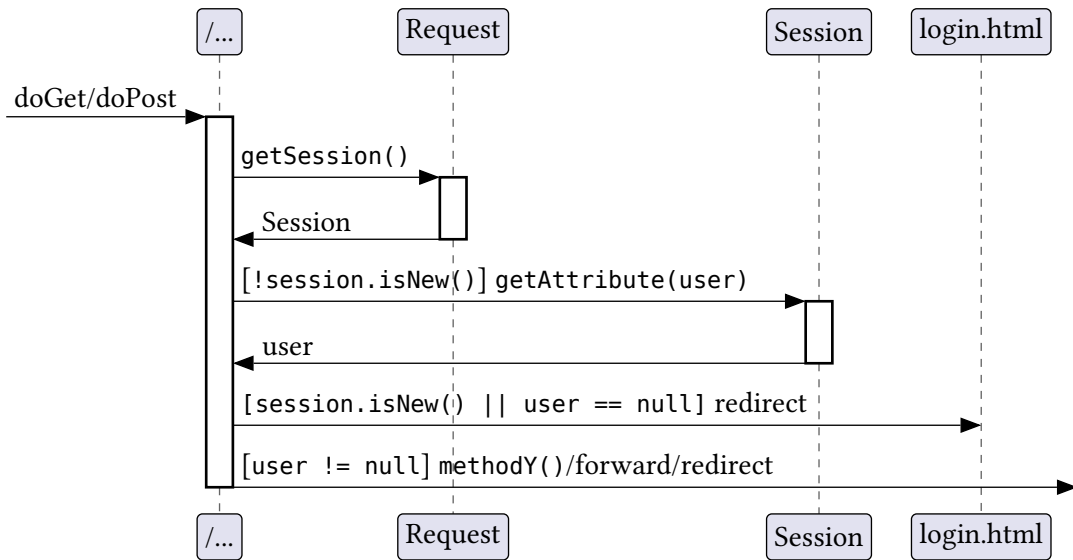


Sequence diagrams

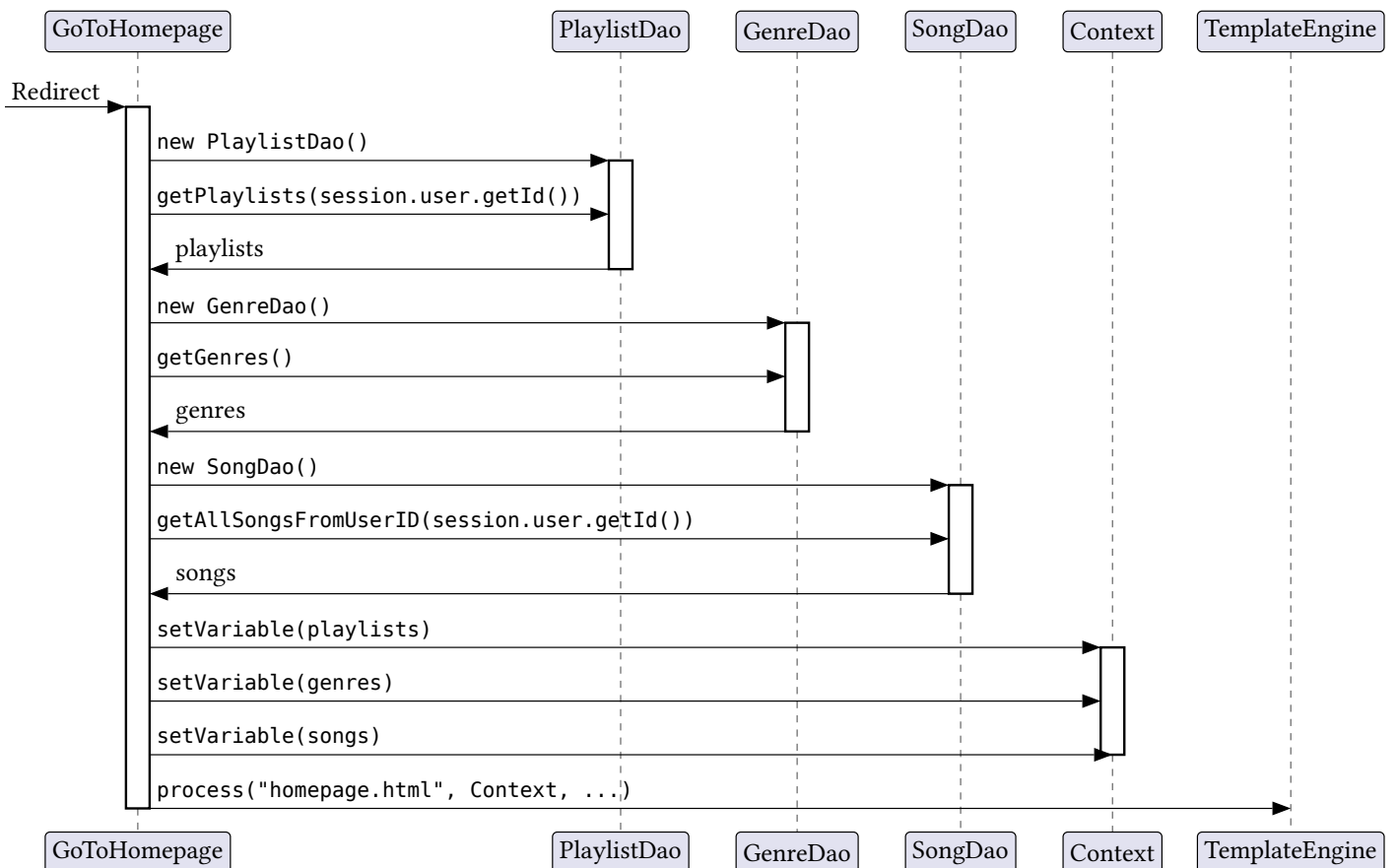
• Login



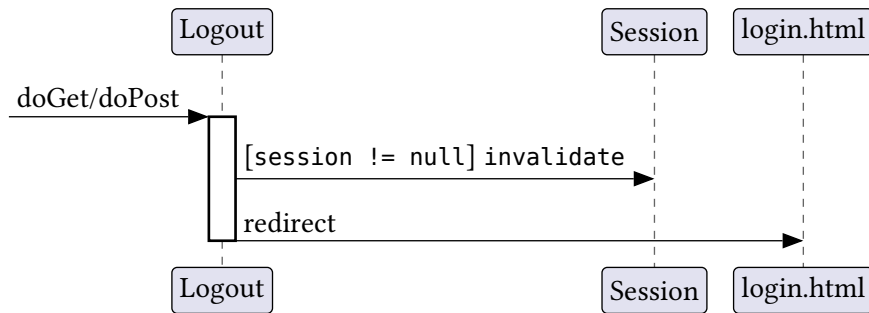
- Controllare l'user



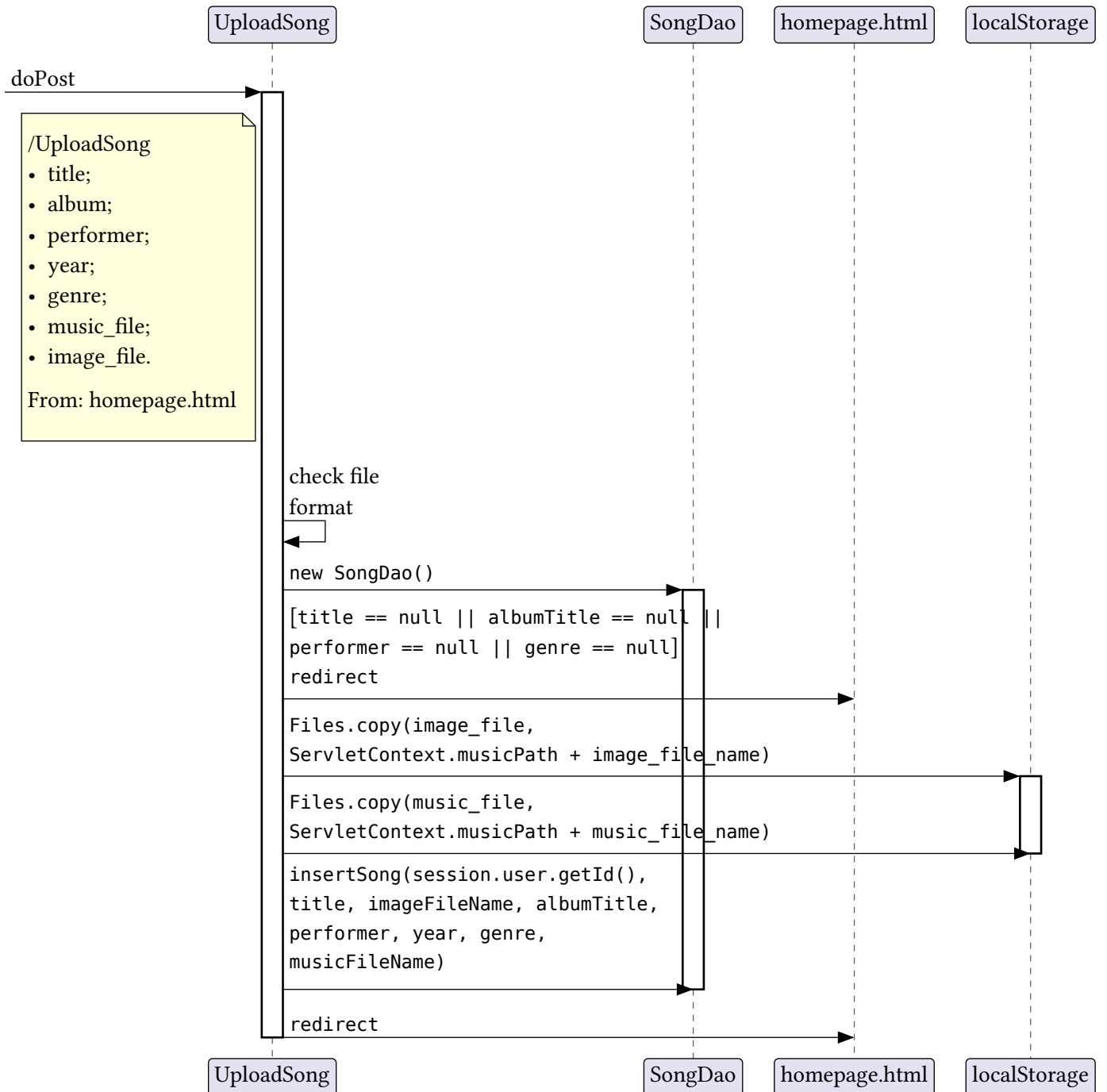
- Tornare/andare alla homepage



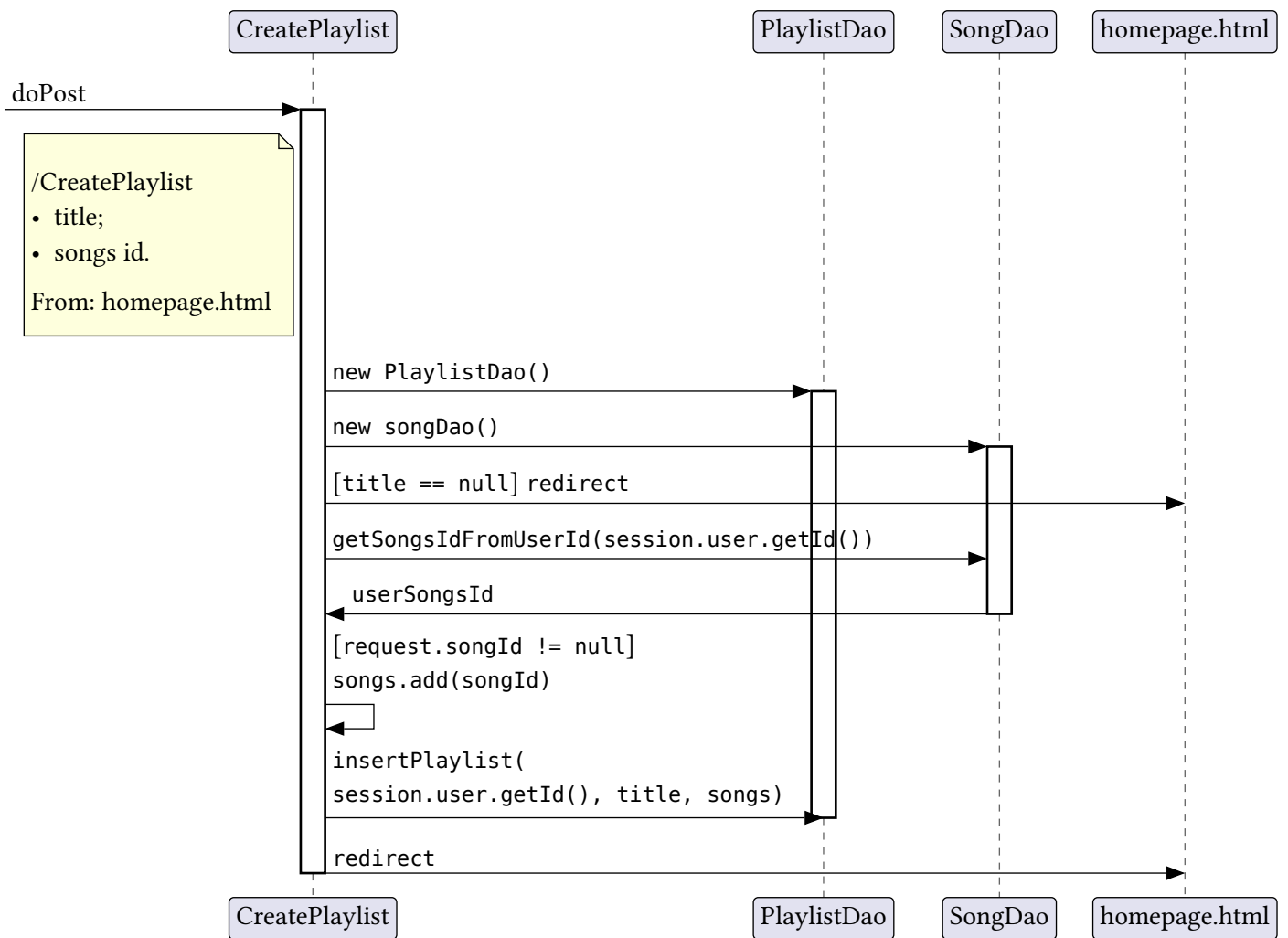
• Logout



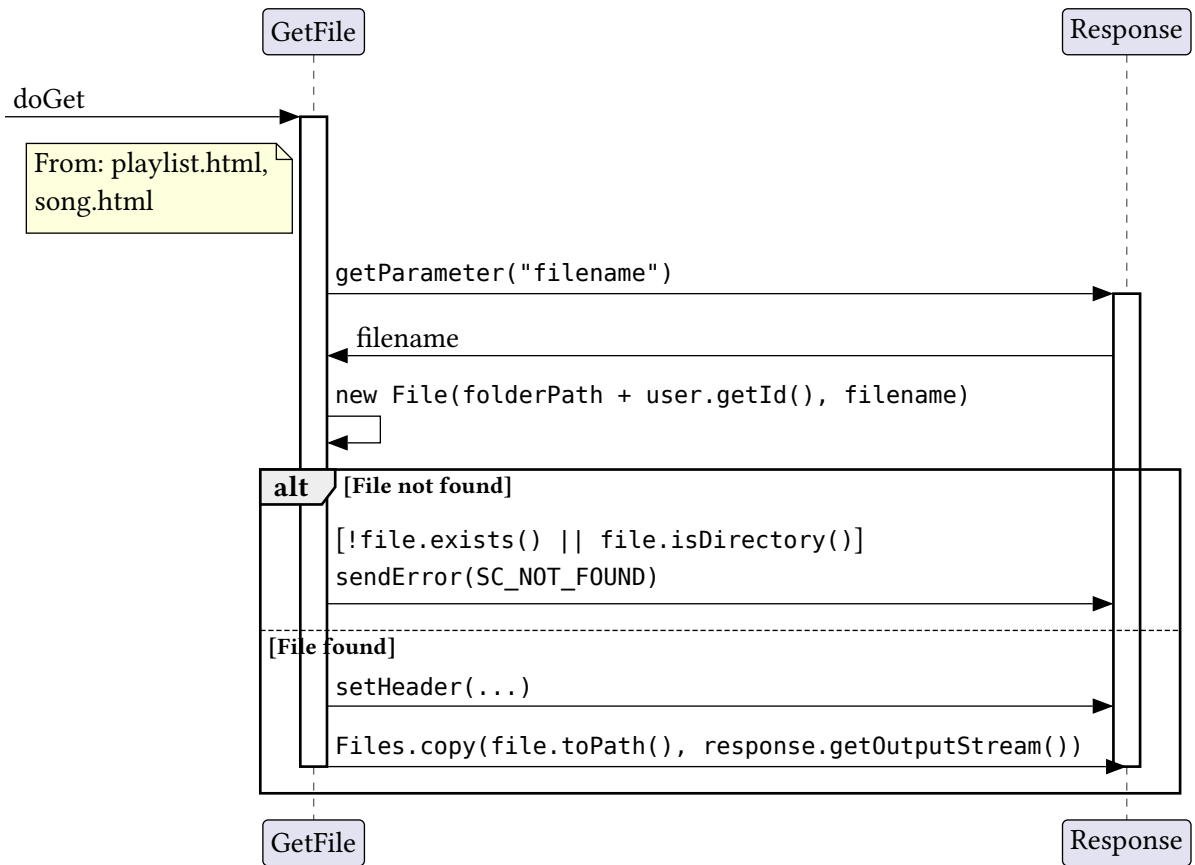
• Caricare una canzone



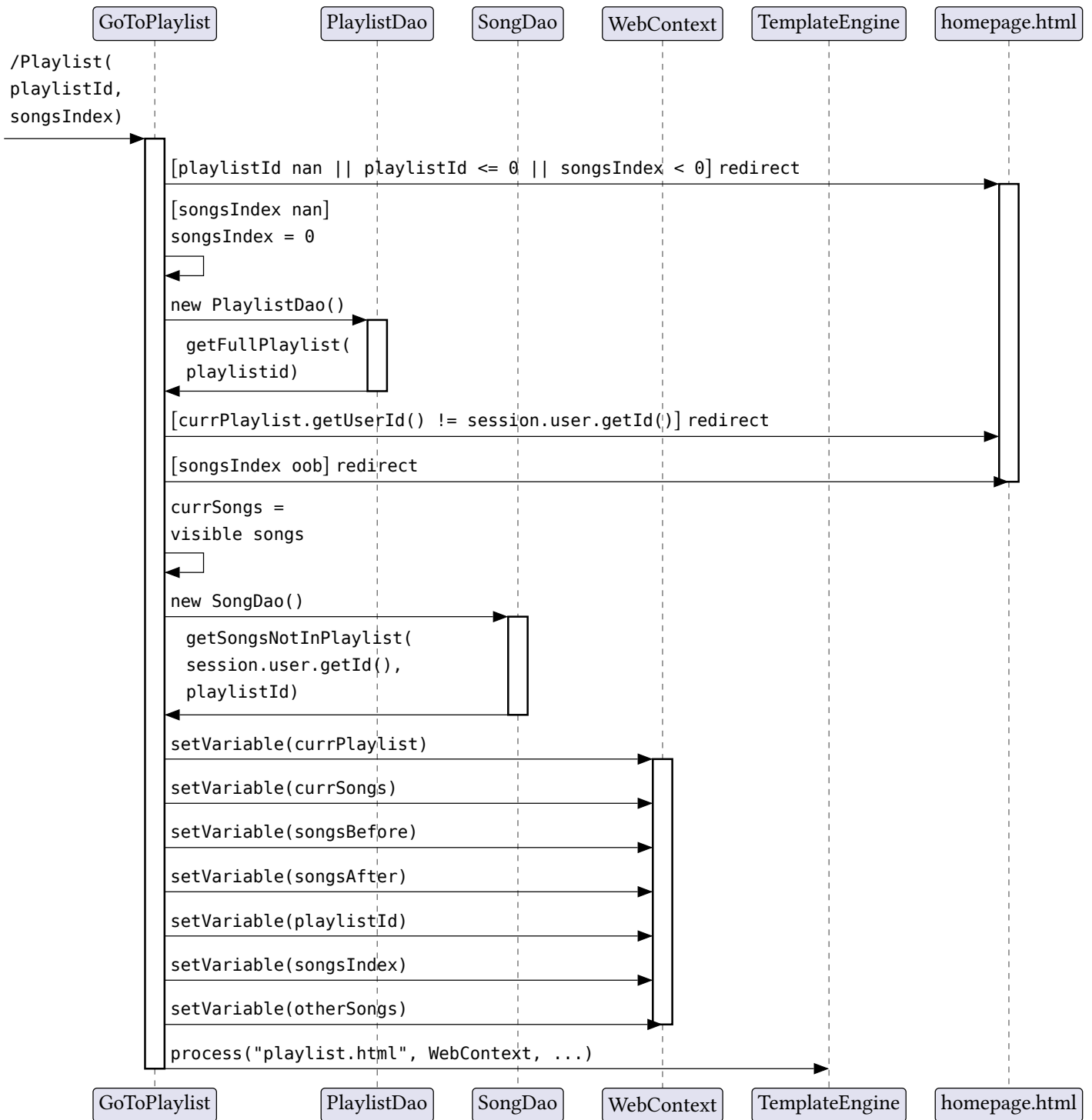
- Creare una playlist



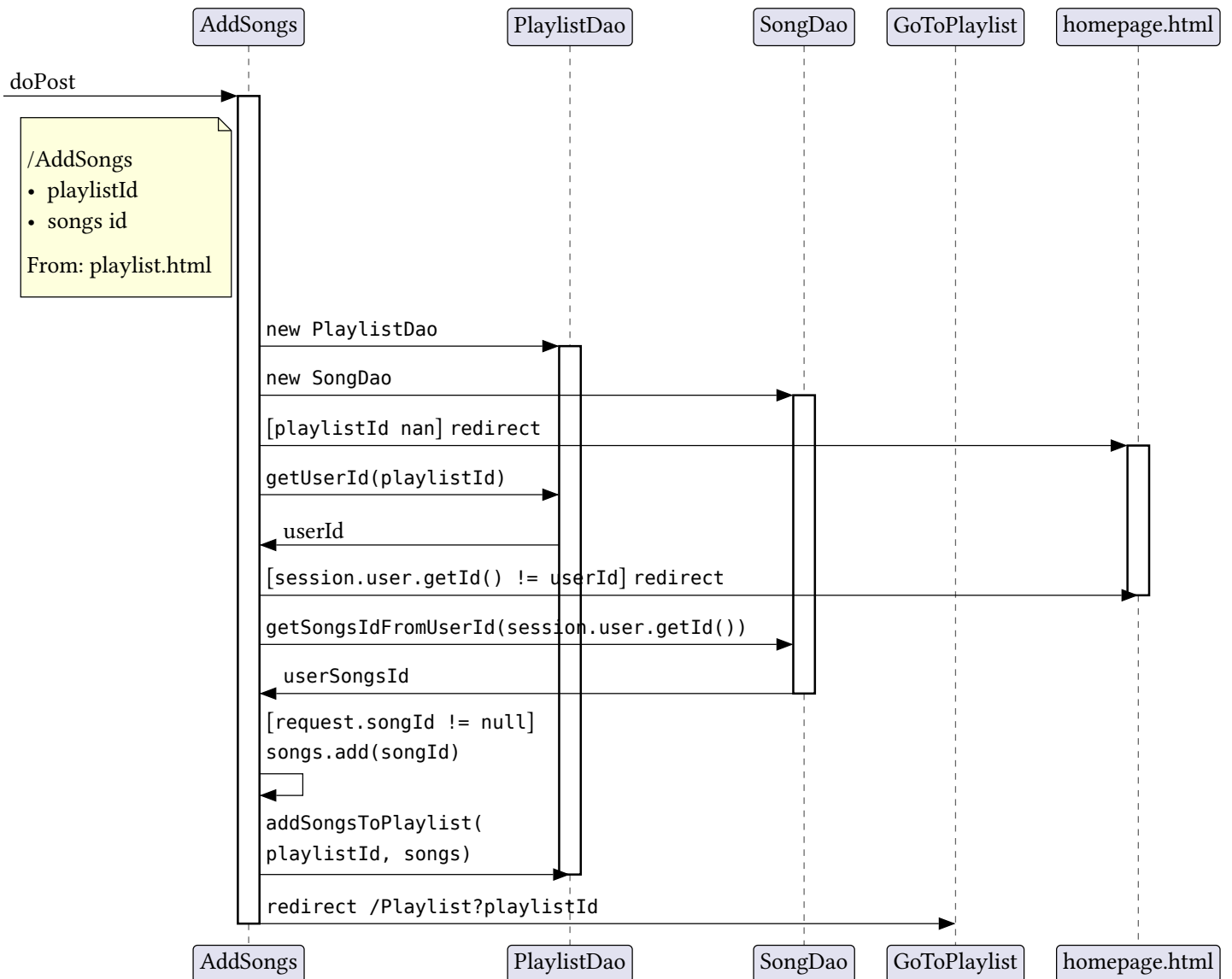
- Recuperare un file



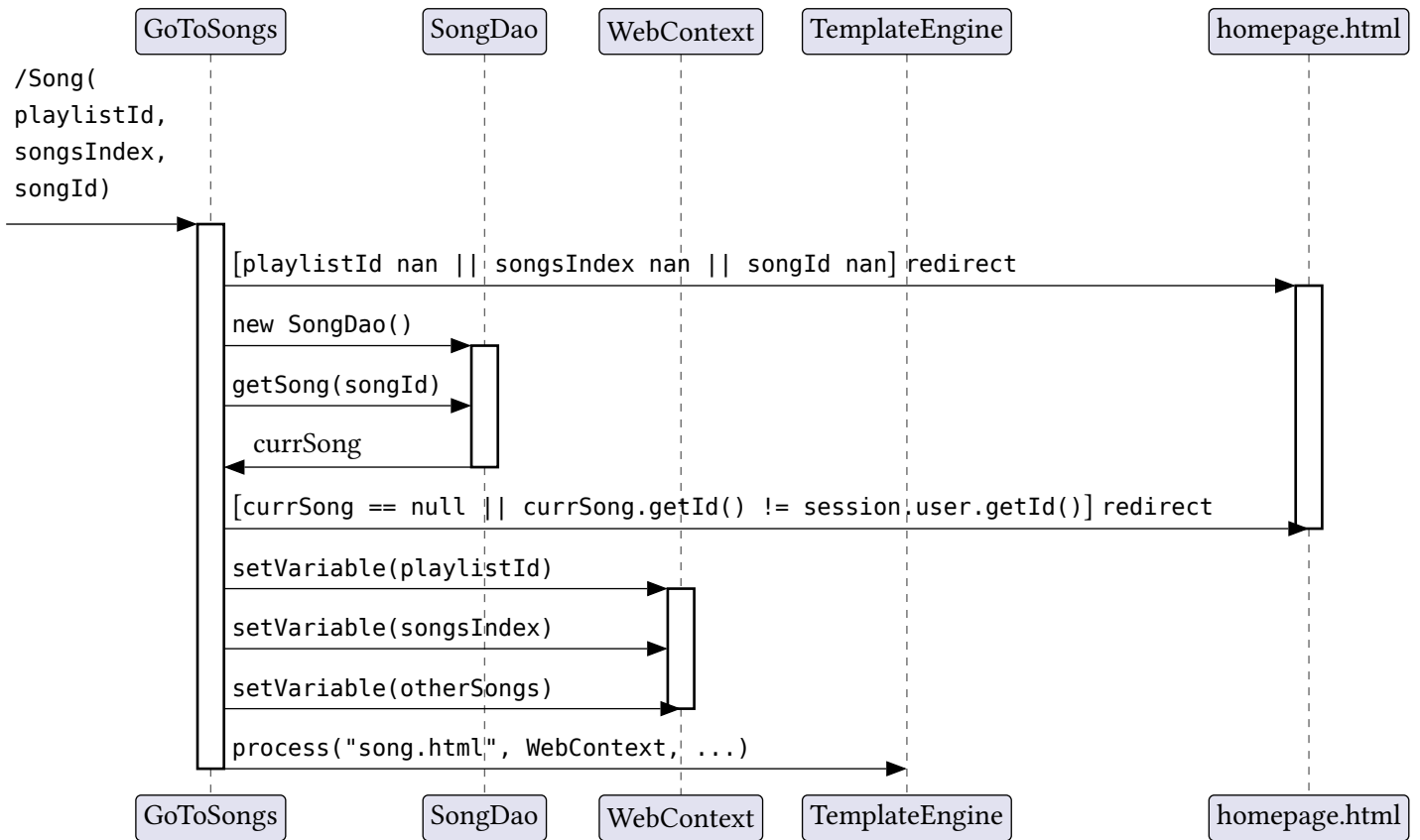
- Andare alla pagina della playlist



- Aggiungere canzoni alla playlist



- Andare alla pagina della canzone



Documentazione ver. Javascript

Analisi requisiti dati

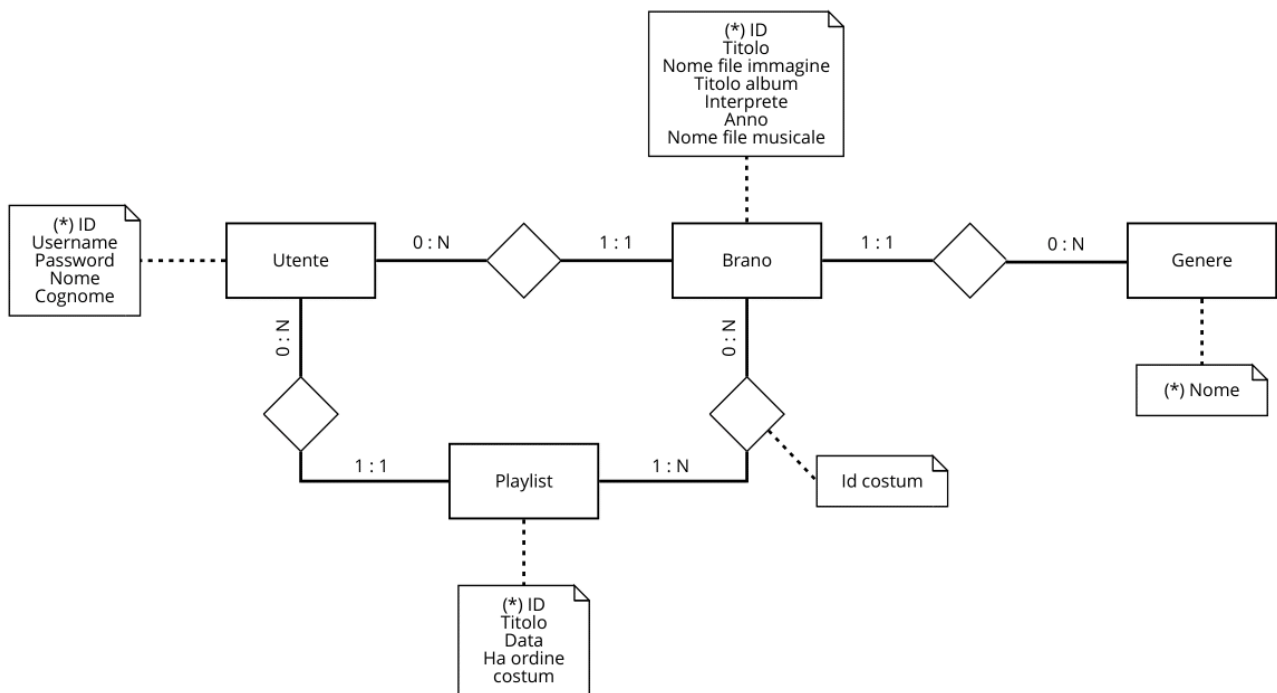
[...]

- 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.

Design database



```
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(64)  not null,
    image_file_name   varchar(64)  not null,
    album_title       varchar(64)  not null,
    performer         varchar(64)  not null,
    year              int          not null check ( year > 0 ),
    genre             varchar(64)  not null,
    music_file_name   varchar(128) 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(64)  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
);

```

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.**

Aggiunta alle specifiche

Diagramma IFL

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			
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

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			
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

Sequence diagram