Music Playlist [RIA]

Lorenzo Campana

June 2, 2021

Matricola:	907081
Codice Persona:	10605775
Docente:	Piero Fraternali

1 Requirements

Si realizzi un'applicazione client server web che modifica le specifiche della versione Pure Html precedente 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 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.

2 Database design

2.1 ER diagram

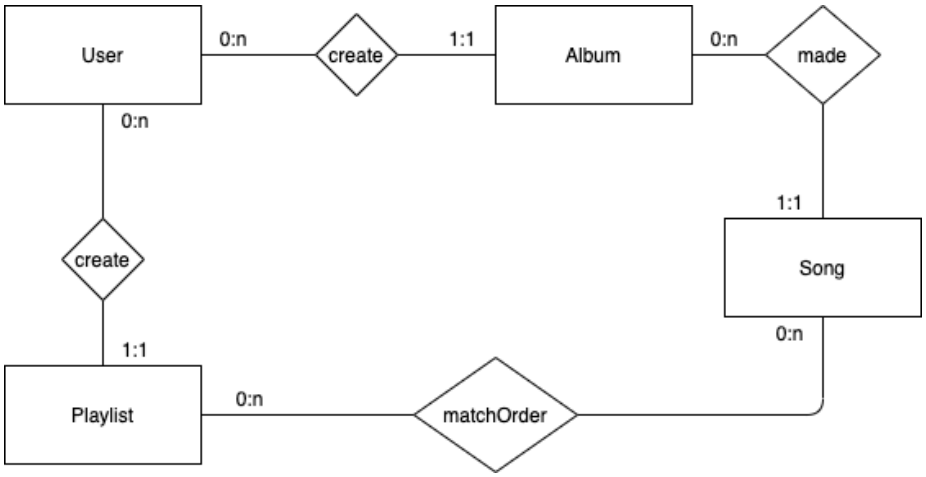

Figure 1: ER diagram

2.2 Database model

The only difference between the HTML pure version and this one is that we save in the MatchOrder table a integer that represents the id of the song before just like a linked list, default is "0" for the first element.

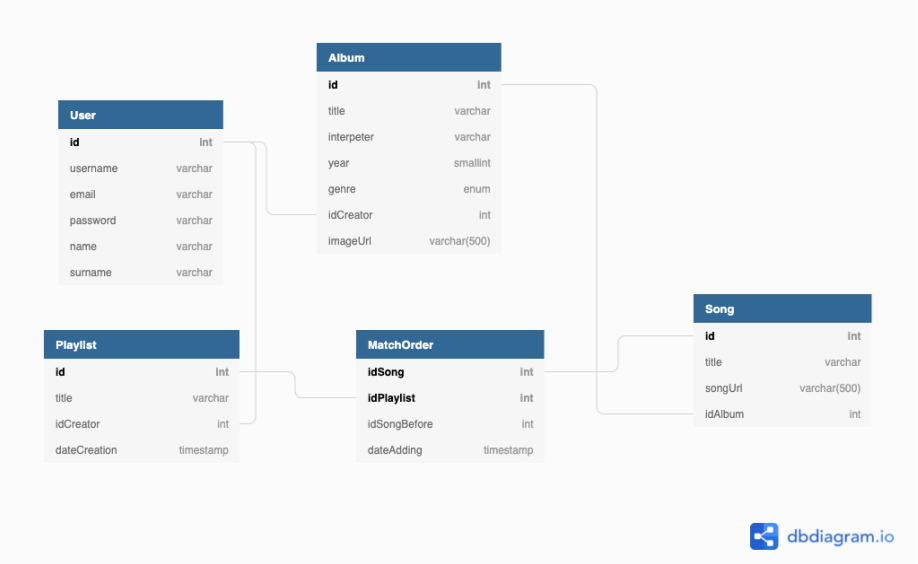

Figure 2: database model

3 Application Design

3.1 Components

- Model objects (Beans)
  - `User`
  - `Genre`
  - `Match`
  - `Playlist`
  - `Song`
  - `SongOrder`
  - `Album`
- Data Access Objects (Classes)
  - AlbumDAO
    - `int createAlbum(Album album)`
    - `Album findAlbumById(int albumId)`
    - `ArrayList<Album> getAllUserAlbums(int userId)`
  - UserDAO
    - `int createUser(User user)`
    - `User checkCredentials(String email, String password)`
  - SongDAO
    - `int createSong(Song song)`
  - PlaylistDAO
    - `int createPlaylist(String title, int idCreator)`
    - `ArrayList<Song> findAllUserSongsNotInPlaylist(int idUser, int idPlaylist)`
    - `ArrayList<Playlist> findUserPlaylists(int userId)`
    - `HashMap<Song, Album> findPlaylistSongs(int idPlaylist)`
  - MatchDAO
    - `void updateSongOrder(int idPlaylist, SongOrder update)`
    - `HashMap<Song, Album> insertSongInPlaylist(int idSong, int idPlaylist)`
- Controllers (Servlets)
  - `AddSongToPlaylist`
  - `CheckLogin`
  - `CreateAlbum`

  - `CreatePlaylist`
  - `CreateSong`
  - `CreateUser`
  - `GetSongsOfPlaylist`
  - `GetUserAlbums`
  - `GetUserPlaylist`
  - `GetUserSongNotInPlaylist`
  - `SaveOrder`
  - `ShowFile`
- Views and Views Components
  - LoginPage.html
    - LoginForm
    - RegistrationForm
  - HomePage.html
    - UserInfo
      - `show()`
      - `hide()`
      - `set(message)`: sets the user username
    - CreateSong
      - `updateAlbumList(album)`: updates select html tag with the album just just created in the createAlbum form
      - `show()`
      - `hide()`
    - CreateAlbum
      - `show()`
      - `hide()`
    - UserPlaylists
      - `show()`: request to server user playlists details
      - `update(playlists)`: updates the lists of playlist
      - `hide()`
      - `registerEvents()`: register create playlist button event
      - `reset()`
    - Playlist
      - `show()`: request to server the songs in the playlist and the songs not in playlist in order to display and add those in the playlist
      - `update(songsMap)`: updates the view with all the songs in the playlist. songsMap represents a HashMap¡Song, Album¿ with all the playlist songs.

- `updateSongToInsert()`: removes the option selected from the select HTML tag.
- `registerEvents()`: register all the button events.
- `nextSlide()`: updates an index and recalls update.
- `prevSlide()`: updates an index and recall update.
- `reset()`

* Player
  - `show()`: updates the view with the new song selected
  - `registerEvents()`: register the goBack link tag.
* PageOrchestrator
  - `start()`: initialize all the components
  - `showHomePage()`: calls `reset()` and then makes the home page components visible calling the `show()` methods of them.
  - `showPlaylistPage()`: calls `reset()` and then makes the playlist page components visible calling the `show()` methods of them.
  - `showPlayerPage()`: calls `reset()` and then makes the player page components visible calling the `show()` methods of them.
  - `showErrorPage()`: calls `reset()` and then makes the error page component visible calling the `set(status, message)` and `show()` methods.
  - `showLocalMessage(container, msg)`: sets and makes visible a message in a specific container
  - `reset()`: calls all components `hide()` method.
  - `registerEvents()`

– ReorderPage.html

* loadSongs: display the songsMap already sorted by the utility methods `orderSongs()`
* saveOrder: updates the attribute `idSongBefore` of all song and return the array updated that will be sent in JSON format to the server.

3.2 IFML

3.2.1 Login and Registration

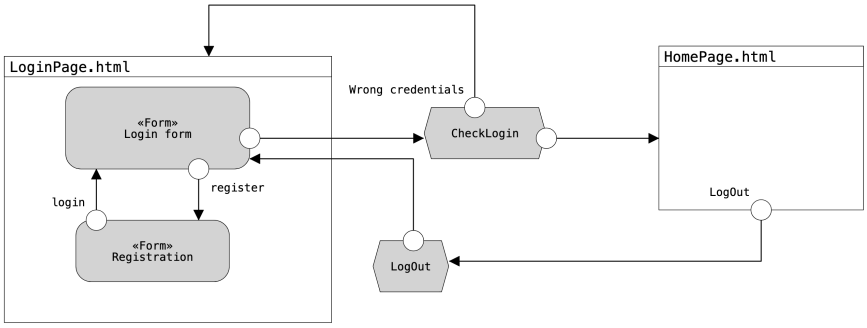

Figure 3: Ifml login managment

3.2.2 Home page

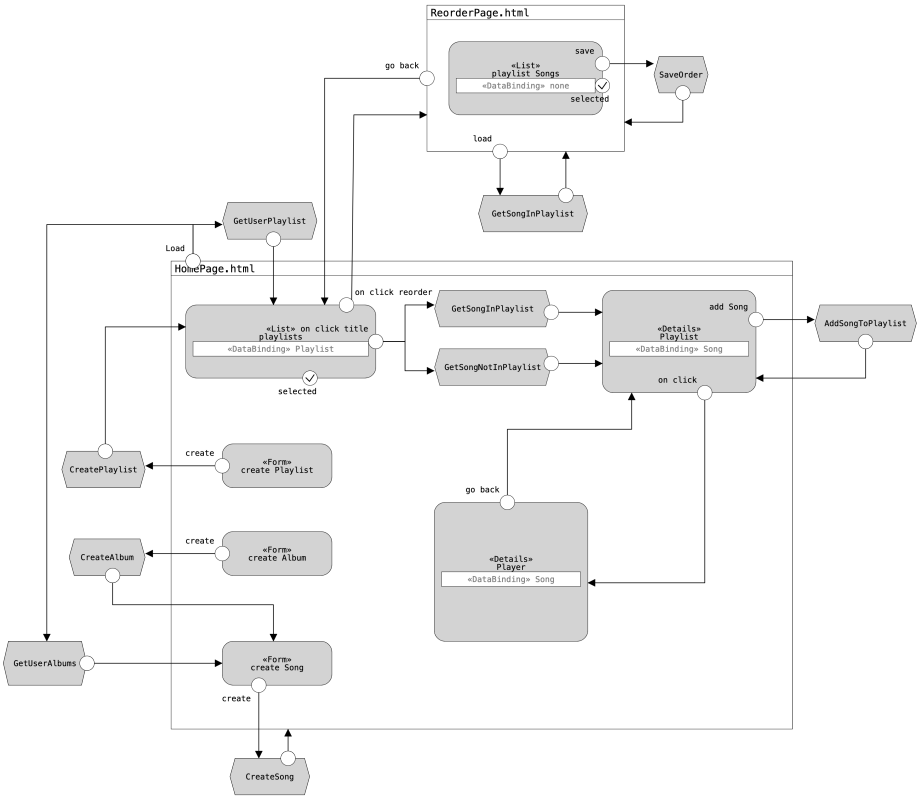

Figure 4: Ifml HomePage managment

4 Event table

Client Side		Server Side	
Event	Action	Event	Action
LoginPage.html			
Submit Login	control input	"POST" /CheckLogin	check user credentials in db
Submit Registration	control input	"POST"/CreateUser	create a new user instance in db
HomePage.html			
load	instanciate all components	"GET" /GetUserAlbums "GET"/GetUserPlaylists	retrieve user albums retrieve user playlists
create song button	control input	"POST" /CreateSong	insertion in db Song Table
create album button	control input and update album list in song creation	"POST"/CreateAlbum	insertion in db Album Table
click playlist	instanciate a new Playlist component and show it	"GET" /GetSongsOfPlaylist "GET"/ GetUserSongsNotInPlaylist	retrive playlist song based on their order
click reorder	redirect to ReorderPage.html	-	-
create playlist	update list of playlists	"POST"/CreatePlaylist	insertion in db Playlist Table
add song to playlist	update list of songs	"POST"/AddSongToPlaylist	insertion in db MatchOrder Table
click to song in playlist	instanciate new Player component and show it	-	-
ReorderPage.html			
load	display song of playlist	"GET"/GetSongOfPlaylist	retrive song of playlist based on their order
save order button	modify idSongBefore of all song based on the sorting in that moment	"POST" /saveSorting	update idSongBefore of all playlist songs in MatchOrder Table

Figure 5: list of all events

5 Event sequence diagram

5.1 Login

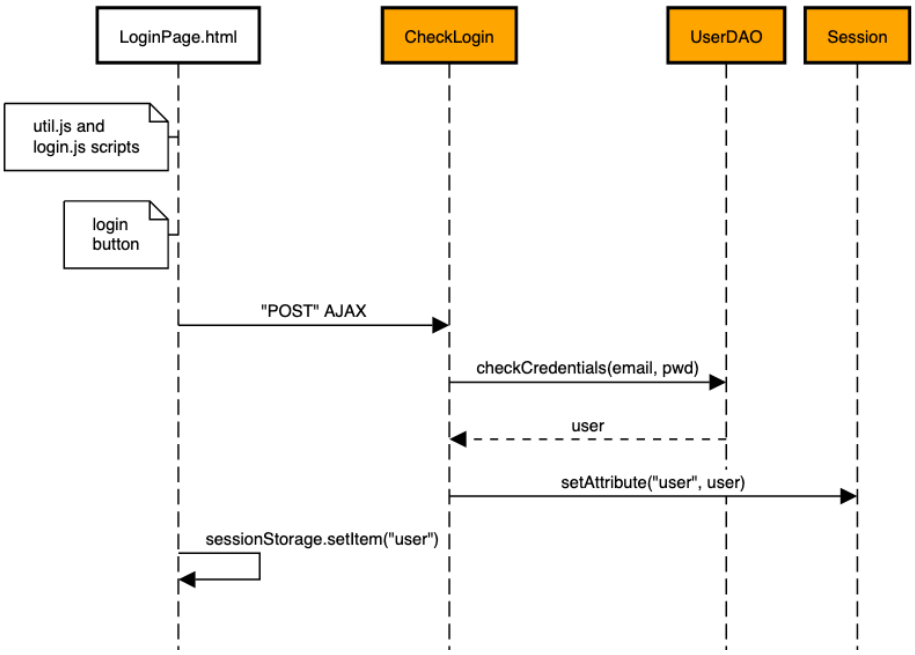

Figure 6: Login event

5.2 HomePage load

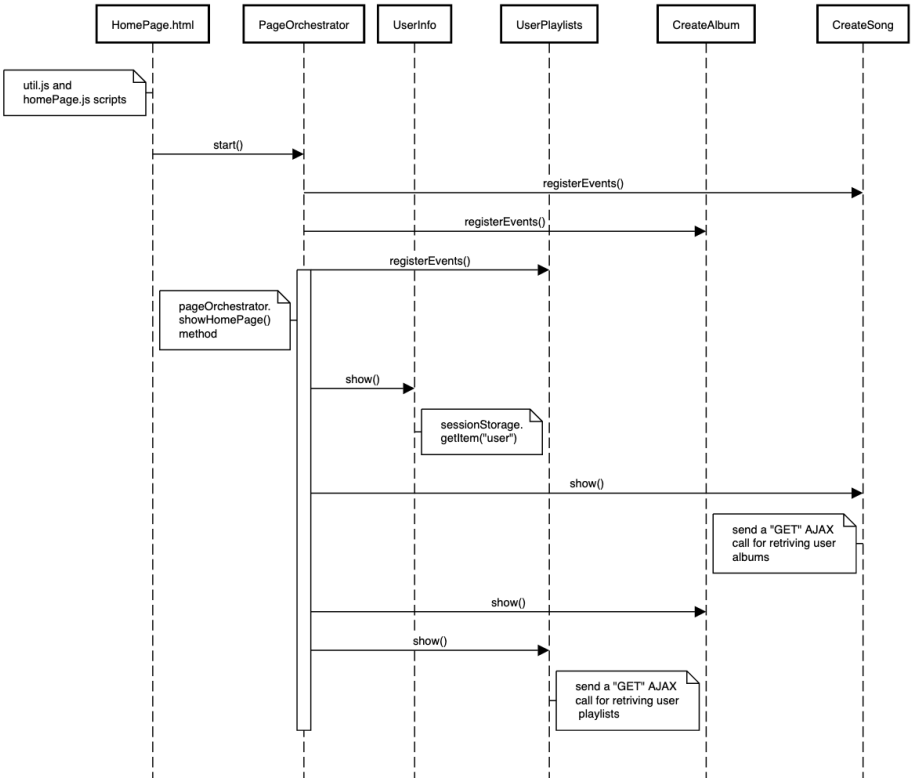

Figure 7: Home page load event

5.3 Playlist click

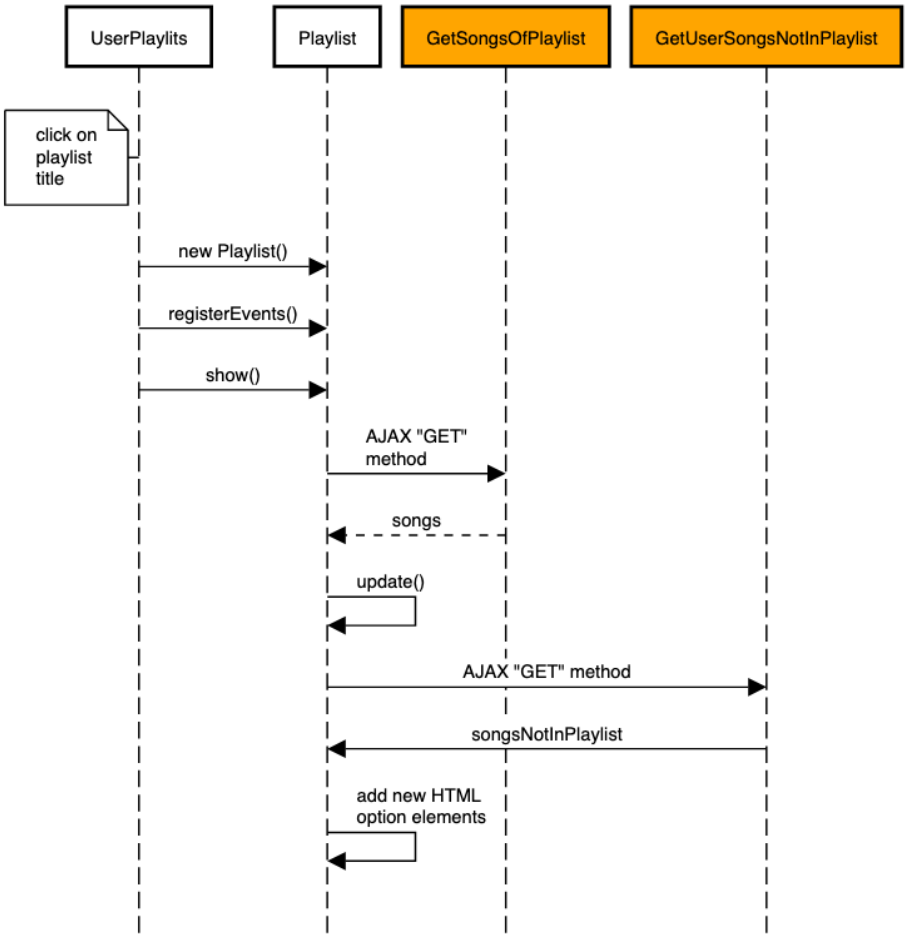

Figure 8: playlist click event

5.4 Reorder click

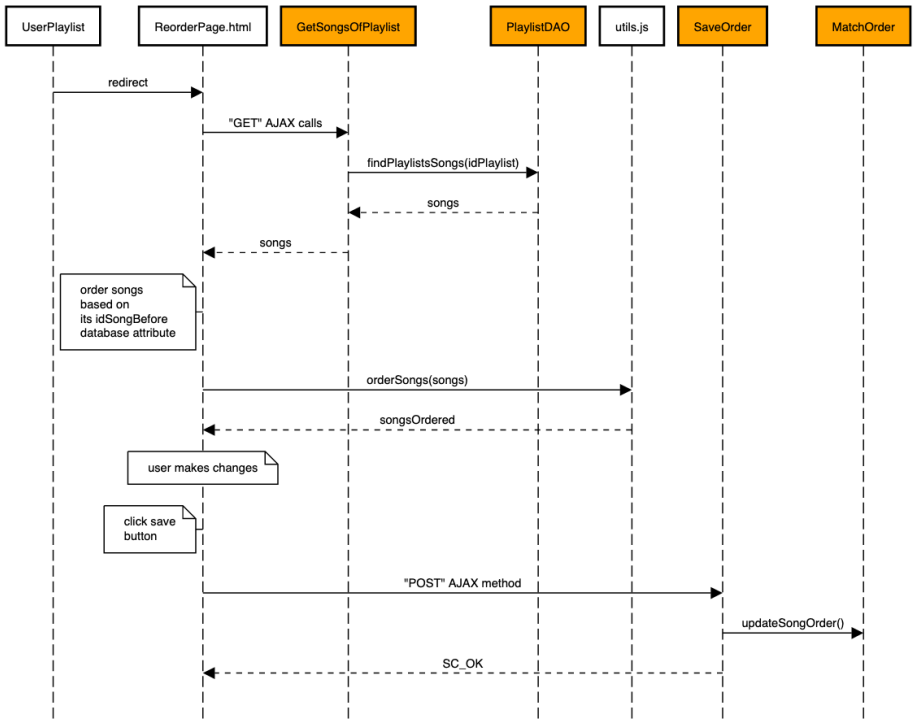

Figure 9: reorder click event

5.5 Song click

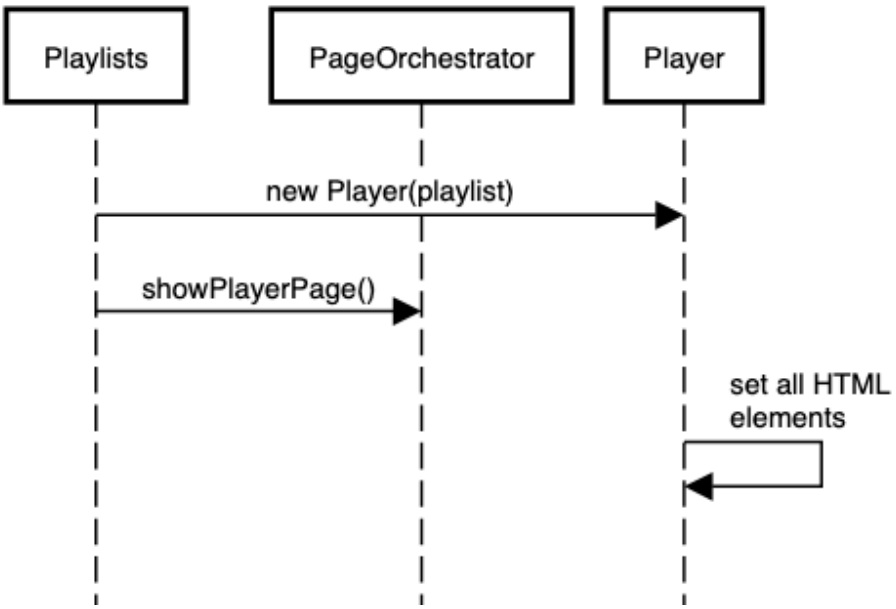

Figure 10: song in playlist click event