GROOVECLAM

Andrea Giacomo Baldan 579117

May 25, 2015

Contents

Abstract

A seguito degli eventi riguardanti il caso 'Napster' nei primi anni 2000, l'industria musicale e la distribuzione del materiale digitale ha subito notevoli cambiamenti e negli anni successivi prese piede il fenomeno del P2P (scambio tra utenti di files musicali, e non solo, mediante la rete) avviato da 'Napster', seguito da piattaforme e siti che offrono un servizio di streaming di file audio/video nel (quasi) totale rispetto dei diritti sugli album pubblicati. Grooveclam è una piattaforma online sulla linea del recente defunto Grooveshark, un sito di streaming audio, che si propone di offrire un servizio di condivisione musicale tra utenti, permettendo

di selezionare brani MP3 per l'ascolto, organizzarli in playlist che possono essere condivise tra utenti connessi tra di loro o in semplici code di riproduzione anonime. Offre in più la possibilità di generare e popolare la propria libreria personale di brani e di contribuire al popolamento della base di dati su cui poggia la piattaforma aggiungendo le proprie canzoni, rendendole così disponibili per l'ascolto a tutti gli utenti.

1 Analisi Dei Requisiti

Si vuole realizzare una base di dati per la gestione di una libreria musicale condivisa e la relativa interfaccia web che permetta interazione tra gli utenti.
Il cuore della libreria è formato da un insieme di album, ogni album è identificato da un codice. E' inoltre formato da alcuni metadati (titolo, autore, anno di pubblicazione), è specificato se si tratta di un album registrato in studio o una versione live e, in quest'ultimo caso, è possibile specificare la città in cui si è svolta la registrazione del concerto, può possedere inoltre informazioni opzionali di carattere generale (critiche ricevute, recensioni o breve storia sulla realizzazione dell'album). Infine ogni album può avere una copertina, a cui fanno riferimento anche tutti i brani che contiene.
Un album contiene più brani musicali. Ogni brano contenuto nell'album è identificato da un codice, ed è formato da alcuni metadati quali titolo, genere, durata. Esistono due tipi di utenti che possono accedere alla libreria, ordinari e amministratori, di entrambi interessano l'indirizzo e-mail, uno username e una password, sono opzionali i dati anagrafici quali nome e cognome. Gli utenti odinari possono decidere di seguire altri utenti ordinari, eccetto se stessi. Ogni utente ordinario ha la possibilità di creare una propria collezione di brani preferiti selezionandoli dalla libreria, può creare una coda di riproduzione anonima, o creare delle playlist delle quali interessa sapere il nome. Interessa inoltre sapere se si tratta di playlist pubbliche o private.
All'interno della collezione i brani non possono ripetersi mentre nelle code di riproduzione o nelle playlist uno stesso brano può comparire più volte. All'atto di registrazione un utente può decidere se attivare un abbonamento free o utilizzare un piano premium.

2 Progettazione concettuale

2.1 Classi

- **Utenti:** Rappresenta un utente del servizio.
  - IdUtente: *Int* «PK»
  - Nome: *String*
  - Cognome: *String*
  - Email: *String*

  Sono definite le seguenti sottoclassi disgiunte:
  - **Amministratore:** Rappresenta un utente con privilegi amministrativi.
  - **Ordinario:** Rappresenta un utente ordinario.
- **Login:** Rappresenta delle credenziali d'accesso per un utente.

  - Username: *String* «PK»
  - Password: *String*

- **Iscrizioni:** Modella un piano di iscrizione.
  - Tipo: *Enum* ['Free', 'Premium']
- **Brani:** Rappresenta un brano.
  - IdBrano: *Int* «PK»
  - Titolo: *String*
  - Genere: *String*
  - Durata: *Float*
- **Album:** Modella un album di brani.
  - IdAlbum: *Int* «PK»
  - Titolo: *String*
  - Autore: *String*
  - Info: *String*
  - Anno: *Date*

  Sono definite le seguenti sottoclassi disgiunte con vincolo di partizionamento.
  - **Live:** Rappresenta un album registrato durante una performance live.
    - Locazione: *String*
  - **Studio:** Rappresenta un album registrato in studio.
- **Copertine:** Rappresenta una generica cover di album.
  - IdImm: *Int* «PK»
  - Path: *String*
- **Playlist:** Modella una playlist.
  - Nome: *String*

  Sono definite le seguenti sottoclassi disgiunte con vincolo di partizionamento.
  - **Pubblica:** Rappresenta una playlist pubblica, a cui tutti gli utenti possono accedere all'ascolto.
  - **Privata:** Rappresenta una playlist privata, solo il creatore può accedervi all'ascolto
- **Collezioni:** Rappresenta una collezione di brani preferiti dall'utente.
  - IdCollezione: *Int* «PK»

2.2 Associazioni

- **Utenti-Collezioni:** "Crea"
  - Ogni utente può creare zero o una collezione, ogni collezione può essere creata da un solo utente.
  - Molteplicità 1 : 1
  - Parziale verso **Utenti**, totale verso **Collezioni**.
- **Utenti-Brani:** "Ascolta"
  - Ogni utente può ascoltare zero o più brani, ogni brano può essere ascoltato da zero o più utenti.
  - Molteplicità N : N
  - Parziale verso **Utenti**, parziale verso **Brani**.
  - Attributi:
    - Timestamp: *Timestamp*
- **Utenti-Brani:** "Accoda"
  - Ogni utente può accodare zero o più brani, ogni brano può essere accodato da zero o più utenti.
  - Molteplicità N : N
  - Parziale verso **Utenti**, parziale verso **Brani**.
  - Attributi:
    - Timestamp: *Timestamp*
- **Utenti-Utenti:** "Segue"
  - Ogni utente può seguire zero o più utenti, ogni utente può essere seguito da zero o più utenti.
  - Molteplicità N : N
  - Parziale verso entrambi.
- **Utenti-Playlist**: "Crea"
  - Ogni utente può creare zero o più playlist, ogni playlist può essere creata da un solo utente.
  - Molteplicità N : 1
  - Parziale verso **Utenti**, totale verso **Playlist**.
- **Utenti-Iscrizioni:** "Iscritto"
  - Ogni utente può avere una sola iscrizione, ogni iscrizione può essere associata ad un solo utente.
  - Molteplicità 1 : 1
  - Totale verso **Utenti** e verso **Iscrizioni**.
- **Playlist-Brani:** "PopolataDa"

  - Ogni playlist è popolata da zero o più brani, ogni brano popola zero o più playlist.
  - Molteplicità N : N
  - Parziale verso **Playlist**, parziale verso **Brani**.

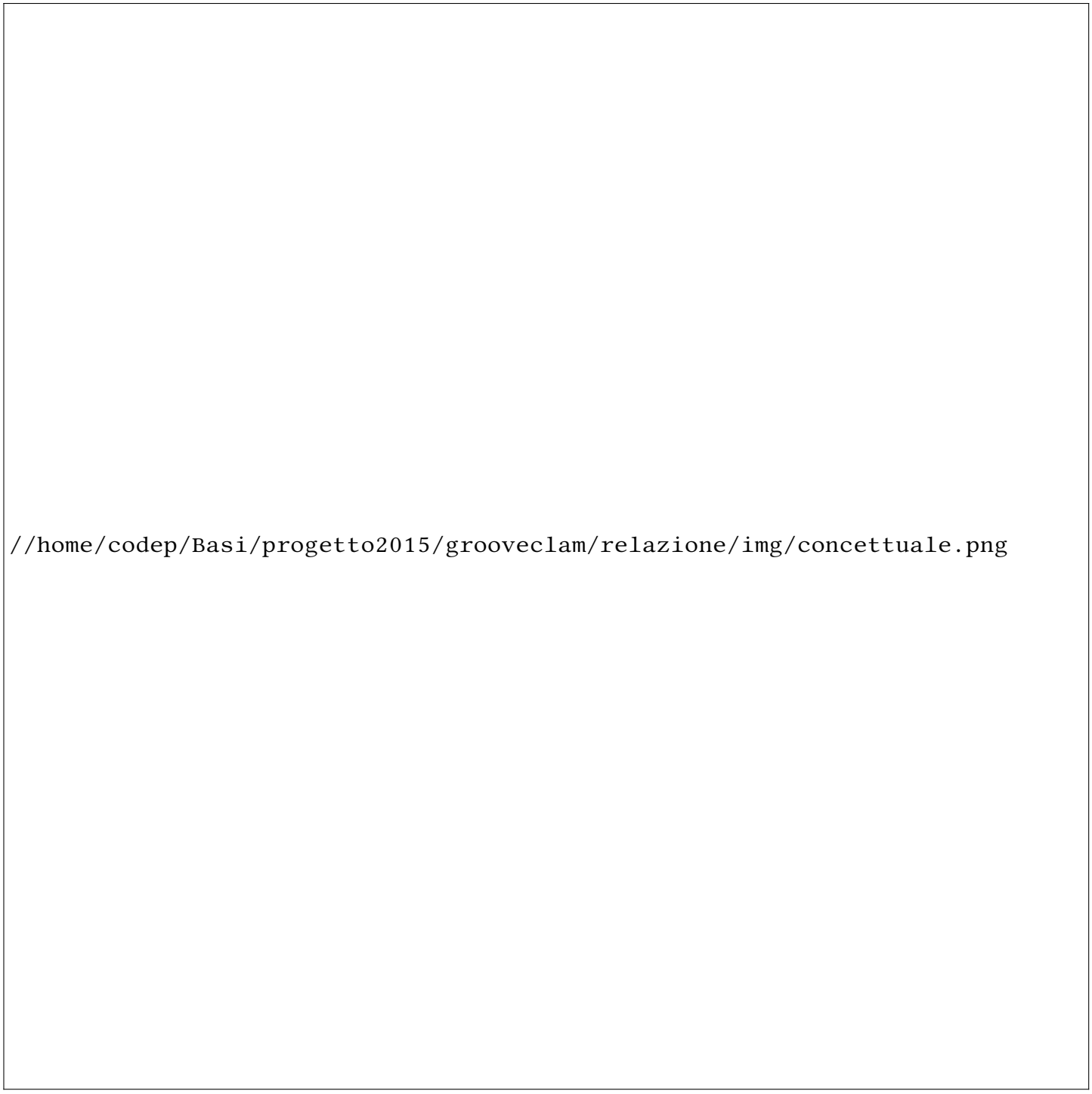
- **Brani-Album:** "AppartieneA"
  - Ogni brano appartiene a zero o un brano, ogni brano contiene uno o più brani.
  - Molteplicità 1 : N
  - Parziale verso **Brani**, totale verso **Album**.

- **Album-Copertine:** "Possiede"
  - Ogni album possiede zero o una copertina, ogni copertina è posseduta da un solo album.
  - Molteplicità 1 : 1
  - Parziale verso **Album**, totale verso **Copertine**.

- **Brani-Copertine:** "Possiede"
  - Ogni brano possiede zero o una cover, ogni cover è posseduta da una o più brani.
  - Molteplicità 1 : N
  - Totale verso **Brani**, totale verso **Copertine**.

- **Collezioni-Brani:** "PopolateDa"
  - Ogni collezione è popolata da zero o più brani, ogni brano popola zero o più collezioni.
  - Molteplicità N : N
  - Parziale verso **Collezioni**, parziale verso **Brani**.

2.3 Schema E/R


//home/codep/Basi/progetto2015/grooveclam/relazione/img/concettuale.png

3 Progettazione Logica

3.1 Gerarchie

Tutte le gerarchie presenti nella progettazione concettuale sono state risolte mediante accorpamento in tabella unica, questo perchè nessuna di esse possedeva sottoclassi con un numero significativo di attributi o associazioni entranti da giustificarne un partizionamento di qualche genere.

3.2 Chiavi Primarie

Sono state create alcune chiavi primarie per identificare le istanze di alcune tabelle, quali *IdPlaylist* a **Playlist**.

3.3 Associazioni

- **Utenti-Collezioni:** "Crea"
  - Ogni utente può creare zero o una collezione, ogni collezione può essere creata da un solo utente.
  - Molteplicità 1 : 1
  - Parziale verso **Utenti**, totale verso **Collezioni**.
  - Chiave esterna non-nulla in **Collezioni** verso **Utenti**.
- **Utenti-Brani:** "Ascolta"
  - Ogni utente può ascoltare zero o più brani, ogni brano può essere ascoltato da zero o più utenti.
  - Molteplicità N : N
  - Parziale verso **Utenti**, parziale verso **Brani**.
  - Attributi:
    - Timestamp: *Timestamp*
  - Nuova tabella **Ascoltate**, attributi:
    - IdUtente: *Int* «PK» «FK(Utenti)»
    - IdBrano: *Int* «PK» «FK(Brani)»
    - Timestamp: *Timestamp* «PK»
- **Utenti-Brani:** "Accoda"
  - Ogni utente può accodare zero o più brani, ogni brano può essere accodato da zero o più utenti.
  - Molteplicità N : N
  - Parziale verso **Utenti**, parziale verso **Brani**.
  - Attributi:
    - Timestamp: *Timestamp*
  - Nuova tabella **Code**, attributi:
    - IdUtente: *Int* «PK» «FK(Utenti)»

        * IdBrano: *Int* «PK» «FK(Brani)»
        * Timestamp: *Timestamp* «PK»

- **Utenti-Utenti:** "Segue"
  - Ogni utente può seguire zero o più utenti, ogni utente può essere seguito da zero o più utenti.
  - Molteplicità N : N
  - Parziale verso entrambi.
  - Nuova tabella **Seguaci**, attributi:
    * IdUtente: *Int* «PK» «FK(Utenti)»
    * IdSeguace: *Int* «PK» «FK(Utenti)»
- **Utenti-Playlist**: "Crea"
  - Ogni utente può creare zero o più playlist, ogni playlist può essere creata da un solo utente.
  - Molteplicità N : 1
  - Parziale verso **Utenti**, totale verso **Playlist**.
  - Chiave esterna non-nulla in **Playlist** verso **Utenti**.
- **Utenti-Iscrizioni:** "Iscritto"
  - Ogni utente può avere una sola iscrizione, ogni iscrizione può essere associata ad un solo utente.
  - Molteplicità 1 : 1
  - Totale verso **Utenti** e verso **Iscrizioni**.
  - Chiave esterna non-nulla in **Iscrizioni** verso **Utenti**.
- **Playlist-Brani:** "PopolataDa"
  - Ogni playlist è popolata da zero o più brani, ogni brano popola zero o più playlist.
  - Molteplicità N : N
  - Parziale verso **Playlist**, parziale verso **Brani**.
  - Nuova tabella **BraniPlaylist**, attributi:
    * IdPlaylist: *Int* «PK» «FK(Playlist)»
    * IdBrano: *Int* «PK» «FK(Brani)»
- **Brani-Album:** "AppartieneA"
  - Ogni brano appartiene a zero o un brano, ogni brano contiene uno o più brani.
  - Molteplicità 1 : N
  - Parziale verso **Brani**, totale verso **Album**.
  - Chiave esterna non-nulla in **Brani** verso **Album**.

- **Album-Copertine:** "Possiede"
  - Ogni album possiede zero o una cover, ogni cover è posseduta da un solo album.
  - Molteplicità 1 : 1
  - Parziale verso **Album**, totale verso **Copertine**.
  - Chiave esterna non-nulla in **Copertine** verso **Album**.
- **Brani-Copertine:** "Possiede"
  - Ogni brano possiede zero o una copertina, ogni copertina è posseduta da una o più brani.
  - Molteplicità 1 : N
  - Totale verso **Brani**, totale verso **Copertine**.
  - Chiave esterna non-nulla in **Brani** verso **Copertine**.
- **Collezioni-Brani:** "PopolateDa"
  - Ogni collezione è popolata da zero o più brani, ogni brano popola zero o più collezioni.
  - Molteplicità N : N
  - Parziale verso **Collezioni**, parziale verso **Brani**.
  - Nuova tabella **BraniCollezione**, attributi:
    - IdBrano: *int* «PK» «FK(Brani)»
    - IdCollezione: *int* «PK «FK(Collezioni)»

4 Implementazione Fisica

Query di implementazione DDL SQL della base di dati. Sorgente in *genera.sql*, popolamento in *popola.sql*. E' stata implementata una tabella **Errori**, riempita mediante procedura a sua volta richiamata dai trigger che ne fanno uso, contiene i messaggi d'errore rilevati. *funproc.sql* contiene invece le funzioni, i trigger e le procedure implementate.

```
1  SET FOREIGN_KEY_CHECKS = 0;
2
3  DROP TABLE IF EXISTS `Errori`;
4  DROP TABLE IF EXISTS `Album`;
5  DROP TABLE IF EXISTS `Brani`;
6  DROP TABLE IF EXISTS `Copertine`;
7  DROP TABLE IF EXISTS `Utenti`;
8  DROP TABLE IF EXISTS `Seguaci`;
9  DROP TABLE IF EXISTS `Iscrizioni`;
10 DROP TABLE IF EXISTS `Collezione`;
11 DROP TABLE IF EXISTS `BraniCollezione`;
12 DROP TABLE IF EXISTS `Playlist`;
13 DROP TABLE IF EXISTS `BraniPlaylist`;
14 DROP TABLE IF EXISTS `Code`;
15 DROP TABLE IF EXISTS `Ascoltate`;
16
```

```
17 -- Table di supporto Errori
18 CREATE TABLE IF NOT EXISTS `Errori` (
19         `Errore` VARCHAR(256) DEFAULT NULL
20 ) ENGINE=InnoDB DEFAULT CHARSET=Latin1;
21 -- Table Album
22 CREATE TABLE IF NOT EXISTS `Album` (
23     `IdAlbum` INT(11) NOT NULL AUTO_INCREMENT,
24     `Titolo` VARCHAR(140) NOT NULL,
25     `Autore` VARCHAR(140) NOT NULL,
26     `Info` VARCHAR(300) DEFAULT NULL,
27     `Anno` YEAR DEFAULT NULL,
28     `Live` BOOLEAN DEFAULT FALSE,
29     `Locazione` VARCHAR(40) DEFAULT NULL,
30     PRIMARY KEY(`IdAlbum`)
31 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
32 -- Table Brani
33 CREATE TABLE IF NOT EXISTS `Brani` (
34     `IdBrano` INT(11) NOT NULL AUTO_INCREMENT,
35     `IdAlbum` INT(11) NOT NULL,
36     `Titolo` VARCHAR(140) NOT NULL,
37     `Genere` VARCHAR(40) NOT NULL,
38     `Durata` INT(11),
39     `IdImm` INT(11) NOT NULL,
40     PRIMARY KEY(`IdBrano`),
41     FOREIGN KEY(`IdAlbum`) REFERENCES Album(`IdAlbum`) ON DELETE CASCADE ON UPDATE CASCADE,
42     FOREIGN KEY(`IdImm`) REFERENCES Copertine(`IdImm`) ON DELETE CASCADE ON UPDATE CASCADE
43 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
44 -- Table Copertine
45 CREATE TABLE IF NOT EXISTS `Copertine` (
46     `IdImm` INT(11) NOT NULL AUTO_INCREMENT,
47     `IdAlbum` INT(11) NOT NULL,
48     `Path` VARCHAR (40) NOT NULL DEFAULT "img/covers/nocover.jpg",
49     PRIMARY KEY(`IdImm`),
50     FOREIGN KEY(`IdAlbum`) REFERENCES Album(`IdAlbum`) ON DELETE CASCADE ON UPDATE CASCADE
51 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
52 -- Table Utenti
53 CREATE TABLE IF NOT EXISTS `Utenti` (
54     `IdUtente` INT(11) NOT NULL AUTO_INCREMENT,
55     `Nome` VARCHAR(40) DEFAULT NULL,
56     `Cognome` VARCHAR(40) DEFAULT NULL,
57     `Email` VARCHAR(40) NOT NULL,
58     `Amministratore` BOOLEAN NOT NULL,
59     `Username` VARCHAR(40) NOT NULL,
60     `Password` VARCHAR(40) NOT NULL,
61     PRIMARY KEY(`IdUtente`),
62     UNIQUE(`Username`)
63 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
64 -- Table Seguaci
65 CREATE TABLE IF NOT EXISTS `Seguaci` (
66     `IdUtente` INT(11) NOT NULL,
67     `IdSeguace` INT(11) NOT NULL,
68     CONSTRAINT PRIMARY KEY pk(`IdUtente`, `IdSeguace`),
69     FOREIGN KEY(`IdUtente`) REFERENCES Utenti(`IdUtente`) ON DELETE CASCADE ON UPDATE CASCADE,
70     FOREIGN KEY(`IdSeguace`) REFERENCES Utenti(`IdUtente`) ON DELETE CASCADE ON UPDATE CASCADE,
71     CHECK(`IdUtente` != `IdSeguace`)
72 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
73 -- Table Iscrizioni
74 CREATE TABLE IF NOT EXISTS `Iscrizioni` (
75     `IdUtente` INT(10) NOT NULL,
76     `Tipo` ENUM('Free', 'Premium') NOT NULL,
77     PRIMARY KEY(`IdUtente`),
78     FOREIGN KEY(`IdUtente`) REFERENCES Utenti(`IdUtente`) ON DELETE CASCADE ON UPDATE CASCADE
```

```
79  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
80  -- Table Collezioni
81  CREATE TABLE IF NOT EXISTS `Collezioni` (
82      `IdCollezione` INT(11) NOT NULL AUTO_INCREMENT,
83      `IdUtente` INT(11) NOT NULL,
84      PRIMARY KEY(`IdCollezione`),
85      FOREIGN KEY(`IdUtente`) REFERENCES Utenti(`IdUtente`) ON DELETE CASCADE ON UPDATE CASCADE
86  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
87  -- Table BraniCollezione
88  CREATE TABLE IF NOT EXISTS `BraniCollezione` (
89      `IdBrano` INT(11) NOT NULL,
90      `IdCollezione` INT(11) NOT NULL,
91      CONSTRAINT PRIMARY KEY pk(`IdCollezione`, `IdBrano`),
92      FOREIGN KEY(`IdBrano`) REFERENCES Brani(`IdBrano`) ON DELETE CASCADE ON UPDATE CASCADE,
93      FOREIGN KEY(`IdCollezione`) REFERENCES Collezioni(`IdCollezione`) ON DELETE CASCADE ON UPDATE
            CASCADE
94  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
95  -- Table Playlist
96  CREATE TABLE IF NOT EXISTS `Playlist` (
97      `IdPlaylist` INT(11) NOT NULL AUTO_INCREMENT,
98      `IdUtente` INT(11) NOT NULL,
99      `Nome` VARCHAR(40) NOT NULL,
100     `Privata` BOOLEAN DEFAULT FALSE,
101     PRIMARY KEY(`IdPlaylist`),
102     FOREIGN KEY(`IdUtente`) REFERENCES Utenti(`IdUtente`) ON DELETE CASCADE ON UPDATE CASCADE
103 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
104 -- Table BraniPlaylist
105 CREATE TABLE IF NOT EXISTS `BraniPlaylist` (
106     `IdPlaylist` INT(11) NOT NULL,
107     `IdBrano` INT(11) NOT NULL,
108     `Posizione` INT(11) NOT NULL,
109     CONSTRAINT PRIMARY KEY pk(`IdPlaylist`, `IdBrano`),
110     FOREIGN KEY(`IdPlaylist`) REFERENCES Playlist(`IdPlaylist`) ON DELETE CASCADE ON UPDATE CASCADE,
111     FOREIGN KEY(`IdBrano`) REFERENCES Brani(`IdBrano`) ON DELETE CASCADE ON UPDATE CASCADE
112 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
113 -- Table Code
114 CREATE TABLE IF NOT EXISTS `Code` (
115     `IdUtente` INT(11) NOT NULL,
116     `IdBrano` INT(11) NOT NULL,
117     `Posizione` INT(11) NOT NULL,
118     CONSTRAINT PRIMARY KEY pk(`IdUtente`, `IdBrano`, `Posizione`),
119     FOREIGN KEY(`IdUtente`) REFERENCES Utenti(`IdUtente`) ON DELETE CASCADE ON UPDATE CASCADE,
120     FOREIGN KEY(`IdBrano`) REFERENCES Brani(`IdBrano`) ON DELETE CASCADE ON UPDATE CASCADE
121 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
122 -- Table Ascoltate
123 CREATE TABLE IF NOT EXISTS `Ascoltate` (
124     `IdUtente` INT(11) NOT NULL,
125     `IdBrano` INT(11) NOT NULL,
126     `Timestamp` TIMESTAMP NOT NULL,
127     CONSTRAINT PRIMARY KEY pk(`IdUtente`, `IdBrano`, `Timestamp`),
128     FOREIGN KEY(`IdUtente`) REFERENCES Utenti(`IdUtente`) ON DELETE CASCADE ON UPDATE CASCADE,
129     FOREIGN KEY(`IdBrano`) REFERENCES Brani(`IdBrano`) ON DELETE CASCADE ON UPDATE CASCADE
130 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
131 -- INSERT POPULATION
132 -- Insert into Utente
133 INSERT INTO Utenti(`Nome`, `Cognome`, `Email`, `Amministratore`, `Username`, `Password`)
134       VALUES('Andrea', 'Baldan', 'a.g.baldan@gmail.com', 0, 'codep', MD5('ciao')),
135             ('Federico', 'Angi', 'angiracing@gmail.com', 0, 'keepcalm', MD5('calm')),
136             ('Marco', 'Rossi', 'rossi@gmail.com', 0, 'rossi', MD5('marco')),
137             ('Luca', 'Verdi', 'verdi@yahoo.it', 0, 'verdi', MD5('luca')),
138             ('Alessia', 'Neri', 'neri@gmail.com', 0, 'neri', MD5('alessia'));
139 -- Insert into Subscrition
```

```
140 INSERT INTO Iscrizioni(`IdUtente`, `Tipo`) VALUES(1, 'Free'), (2, 'Free');
141 -- Insert into Album
142 INSERT INTO Album(`Titolo`, `Autore`, `Info`, `Anno`, `Live`, `Locazione`)
143       VALUES('Inception Suite', 'Hans Zimmer', 'Inception movie soundtrack, composed by the Great
          Compositor Hans Zimmer', '2010', 0, NULL),
144             ('The Good, the Bad and the Ugly: Original Motion Picture Soundtrack', 'Ennio Morricone',
                'Homonym movie soundtrack, created by the Legendary composer The Master Ennio
                Morricone', '1966', 0, NULL),
145             ('Hollywood in Vienna 2014', 'Randy Newman - David Newman', 'Annual cinematographic review
                 hosted in Vienna', '2014', 1, 'Vienna'),
146             ('The Fragile', 'Nine Inch Nails', 'The Fragile is the third album and a double album by
                American industrial rock band Nine Inch Nails, released on September 21, 1999, by
                Interscope Records.', '1999', 0, NULL),
147             ('American IV: The Man Comes Around', 'Johnny Cash', 'American IV: The Man Comes Around is
                 the fourth album in the American series by Johnny Cash(and his 87th overall),
                released in 2002. The majority of songs are covers which Cash performs in his own
                spare style, with help from producer Rick Rubin.', '2002', 0, NULL),
148             ('Greatest Hits', 'Neil Young', 'Rock & Folk Rock greatest success songs by Neil Young', '
                2004', 0, NULL);
149 -- Insert into Brani
150 INSERT INTO Brani(`IdAlbum`, `Titolo`, `Genere`, `Durata`, `IdImm`)
151       VALUES(1, 'Mind Heist', 'Orchestra', 203, 1),
152             (1, 'Dream is collapsing', 'Orchestra', 281, 1),
153             (1, 'Time', 'Orchestra', 215, 1),
154             (1, 'Half Remembered Dream', 'Orchestra', 71, 1),
155             (1, 'We Built Our Own World', 'Orchestra', 115, 1),
156             (1, 'Radical Notion', 'Orchestra', 222, 1),
157             (1, 'Paradox', 'Orchestra', 205, 1),
158             (2, 'Il Tramonto', 'Orchestra', 72, 2),
159             (2, 'L\'estasi dell\'oro', 'Orchestra', 202, 2),
160             (2, 'Morte di un soldato', 'Orchestra', 185, 2),
161             (2, 'Il Triello', 'Orchestra', 434, 2),
162             (3, 'The Simpsons', 'Orchestra', 172, 3),
163             (3, 'The war of the Roses', 'Orchestra', 272, 3),
164             (4, 'Somewhat Damaged', 'Industrial Metal', 271, 4),
165             (4, 'The Day The Whole World Went Away', 'Industrial Metal', 273, 4),
166             (4, 'We\'re In This Together', 'Industrial Metal', 436, 4),
167             (4, 'Just Like You Imagined', 'Industrial Metal', 229, 4),
168             (4, 'The Great Below', 'Industrial Metal', 317, 4),
169             (5, 'Hurt', 'Country', 218, 5),
170             (5, 'Danny Boy', 'Country', 199, 5),
171             (6, 'Old Man', 'Rock', 203, 6),
172             (6, 'Southern Man', 'Rock', 331, 6);
173 -- Insert into Copertine
174 INSERT INTO Copertine(`IdImm`, `IdAlbum`, `Path`)
175       VALUES(1, 1, 'img/covers/inception.png'),
176       (2, 2, 'img/covers/morricone.jpg'),
177       (3, 3, 'img/covers/hivlogo.jpg'),
178       (4, 4, 'img/covers/fragile.jpg'),
179       (5, 5, 'img/covers/nocover.jpg'),
180       (6, 6, 'img/covers/nocover.jpg');
181 -- Insert into Collezioni
182 INSERT INTO Collezioni(`IdUtente`) VALUES(1), (2);
183 -- Insert into BraniCollezione
184 INSERT INTO BraniCollezione(`IdBrano`, `IdCollezione`) VALUES(1, 1), (2, 1), (3, 1), (2, 2);
185 -- Insert into Playlist
186 INSERT INTO Playlist(`IdUtente`, `Nome`, `Privata`) VALUES(1, 'Score & Soundtracks', 0), (1, 'Southern
        Rock', 0), (2, 'Colonne sonore western', 0);
187 -- Insert into BraniPlaylist
188 INSERT INTO BraniPlaylist(`IdPlaylist`, `IdBrano`, `Posizione`) VALUES(1, 1, 1), (1, 2, 2), (1, 3, 3),
        (1, 4, 4), (1, 5, 5), (2, 21, 1), (2, 22, 2), (3, 5, 1), (3, 7, 2), (3, 4, 3);
189 -- Insert into Code
```

```
INSERT INTO Code(‘IdUtente‘, ‘IdBrano‘, ‘Posizione‘)
       VALUES(1, 1, 1),
       (1, 5, 2),
       (1, 1, 3),
       (1, 12, 4),
       (1, 10, 5),
       (2, 1, 1);
-- Insert into Ascoltate
INSERT INTO Ascoltate(‘IdUtente‘, ‘IdBrano‘, ‘Timestamp‘)
       VALUES(1, 1, ’2015-04-28 18:50:03’),
       (1, 5, ’2015-04-28 18:54:06’),
       (1, 1, ’2015-04-28 19:01:43’),
       (3, 7, ’2015-04-29 18:51:02’),
       (3, 11, ’2015-04-29 17:23:15’),
       (2, 9, ’2015-04-30 21:12:52’),
       (2, 1, ’2015-05-02 22:21:22’);
-- Insert into Seguaci
INSERT INTO Seguaci(‘IdUtente‘, ‘IdSeguace‘) VALUES(1, 2), (1, 3), (2, 1), (3, 1);
SET FOREIGN_KEY_CHECKS = 1;
```

4.1 Trigger

Di seguito i trigger creati.

```
DROP TRIGGER IF EXISTS checkDuration;
DROP TRIGGER IF EXISTS errorTrigger;
DROP TRIGGER IF EXISTS checkFollower;
DROP TRIGGER IF EXISTS checkCoverImage;

DELIMITER $$

CREATE TRIGGER checkDuration
BEFORE INSERT ON ‘Brani‘
FOR EACH ROW
BEGIN
IF(NEW.Durata < 0) THEN
    CALL RAISE_ERROR(’La durata di un brano non p u  essere negativa’);
END IF;
END $$

CREATE TRIGGER errorTrigger
BEFORE INSERT ON ‘Errori‘
FOR EACH ROW
BEGIN
    SET NEW = NEW.errore;
END $$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER checkFollower
BEFORE INSERT ON ‘Seguaci‘
FOR EACH ROW
BEGIN
    IF NEW.IdUtente = NEW.IdSeguace THEN
       CALL RAISE_ERROR(’Un utente non p u  seguire se stesso (IdUtente e IdSeguace devono essere
           diversi fra loro)’);
    END IF;
END $$
```

```
DELIMITER ;

DELIMITER $$

CREATE TRIGGER checkCoverImage
BEFORE INSERT ON `Copertine`
FOR EACH ROW
BEGIN
    IF NEW.Path = '' THEN
       SET NEW.Path = 'img/covers/nocover.jpg';
    END IF;
END $$

DELIMITER ;
```

4.2 Funzioni e Procedure

Alcune funzioni e procedure implementate.

```
DROP FUNCTION IF EXISTS albumTotalDuration;
DROP FUNCTION IF EXISTS elegibleForPrize;

DELIMITER $$

CREATE FUNCTION albumTotalDuration(IdAlbum INT)
RETURNS VARCHAR(5)
BEGIN
DECLARE Seconds INT UNSIGNED;
SELECT SUM(b.Durata) INTO Seconds FROM Brani b WHERE b.IdAlbum = IdAlbum;
RETURN CONCAT(FLOOR(Seconds / 60), ':', (Seconds % 60));
END $$

DELIMITER ;

DELIMITER $$

CREATE FUNCTION elegibleForPrize(IdUser INT, Genre VARCHAR(50))
RETURNS BOOLEAN
BEGIN
DECLARE Seconds INT UNSIGNED DEFAULT 0;
DECLARE Elegibility BOOLEAN DEFAULT FALSE;
SELECT SUM(b.Durata) INTO Seconds
FROM Ascoltate a INNER JOIN Utenti u ON(a.IdUtente = u.IdUtente)
                 INNER JOIN Brani b ON(a.IdBrano = b.IdBrano)
WHERE b.Genere = 'Orchestra' AND a.IdUtente = IdUser;
IF(Seconds >= 1000) THEN
            SET Elegibility = TRUE;
END IF;
RETURN Elegibility;
END $$

DELIMITER ;

DROP PROCEDURE IF EXISTS RAISE_ERROR;
DROP PROCEDURE IF EXISTS GENRE_DISTRIBUTION;
DROP PROCEDURE IF EXISTS USER_GENRE_DISTRIBUTION;
DROP PROCEDURE IF EXISTS SWAP_POSITION;

DELIMITER $$

```

```
42  CREATE PROCEDURE RAISE_ERROR (IN ERROR VARCHAR(256))
43  BEGIN
44  DECLARE V_ERROR VARCHAR(256);
45  SET V_ERROR := CONCAT('[ERROR: ', ERROR, ']');
46  INSERT INTO Errors VALUES(V_ERROR);
47  END $$
48
49  DELIMITER ;
50
51  DELIMITER $$
52
53  CREATE PROCEDURE GENRE_DISTRIBUTION()
54  BEGIN
55  DECLARE Total INT DEFAULT 0;
56  DROP TEMPORARY TABLE IF EXISTS `Distribution`;
57  CREATE TEMPORARY TABLE `Distribution` (
58          `Genere` VARCHAR(100),
59          `Percentuale` VARCHAR(6)
60  ) ENGINE=InnoDB;
61  SELECT count(b.Genere) INTO Total FROM Brani b;
62  INSERT INTO Distribution (Genere, Percentuale)
63  SELECT Genere, CONCAT(FLOOR((count(Genere) / Total) * 100), "%")
64  FROM Brani GROUP BY Genere;
65  END $$
66
67  DELIMITER ;
68
69  DELIMITER $$
70
71  CREATE PROCEDURE USER_GENRE_DISTRIBUTION(IN IdUser INT)
72  BEGIN
73  DECLARE Done INT DEFAULT 0;
74  DECLARE Total INT DEFAULT 0;
75  DECLARE Genre VARCHAR(100) DEFAULT "";
76  DECLARE Counter INT DEFAULT 0;
77  DECLARE D_CURSOR CURSOR FOR
78          SELECT b.Genere, COUNT(b.IdBrano)
79          FROM Brani b INNER JOIN BraniCollezione bc ON (b.IdBrano = bc.IdBrano)
80                       INNER JOIN Collezioni c ON(c.IdCollezione = bc.IdCollezione)
81          WHERE c.IdUtente = IdUser
82          GROUP BY b.Genere, c.IdUtente;
83  DECLARE CONTINUE HANDLER
84  FOR NOT FOUND SET Done = 1;
85  SELECT COUNT(b.IdBrano) INTO Total
86  FROM Brani b INNER JOIN BraniCollezione bc ON(b.IdBrano = bc.IdBrano)
87               INNER JOIN Collezioni c ON(bc.IdCollezione = c.IdCollezione)
88  WHERE c.IdUtente = IdUser;
89  DROP TEMPORARY TABLE IF EXISTS `Distribution`;
90  CREATE TEMPORARY TABLE `Distribution` (
91          `Genere` VARCHAR(100),
92          `Percentuale` VARCHAR(6)
93  ) ENGINE=InnoDB;
94  OPEN D_CURSOR;
95  REPEAT
96          FETCH D_CURSOR INTO Genre, Counter;
97          IF NOT Done THEN
98             INSERT INTO Distribution (Genere, Percentuale)
99             VALUES(Genre, CONCAT(FLOOR((Counter / Total) * 100), "%"));
100         END IF;
101 UNTIL Done END REPEAT;
102 CLOSE D_CURSOR;
103 SELECT * FROM `Distribution` ORDER BY Percentuale;
```

```
104 DROP TABLE `Distribution`;
105 END $$
106
107 DELIMITER ;
108
109 DELIMITER $$
110
111 CREATE PROCEDURE SWAP_POSITION(IN a INT, IN b INT, IN id INT, IN tab INT)
112 BEGIN
113 DECLARE AUX INT DEFAULT -1;
114 CASE tab
115     WHEN 1 THEN
116         UPDATE Code SET Posizione = AUX WHERE Posizione = a AND IdUtente = id;
117         UPDATE Code SET Posizione = a WHERE Posizione = b AND IdUtente = id;
118         UPDATE Code SET Posizione = b WHERE Posizione = AUX AND IdUtente = id;
119     ELSE
120         UPDATE BraniPlaylist SET Posizione = AUX WHERE Posizione = a AND IdPlaylist = id;
121         UPDATE BraniPlaylist SET Posizione = a WHERE Posizione = b AND IdPlaylist = id;
122         UPDATE BraniPlaylist SET Posizione = b WHERE Posizione = AUX AND IdPlaylist = id;
123 END CASE;
124 END $$
125
126 DELIMITER ;
```

5 Query

Alcune query significative.

1. Titolo, album e username dell'utente, degli ultimi 10 brani ascoltati tra i followers.

```
1 SELECT b.Titolo, a.Titolo as TitoloAlbum, u.Username, h.Timestamp
2 FROM Brani b INNER JOIN Album a ON(b.IdAlbum = a.IdAlbum)
3              INNER JOIN Ascoltate h ON(h.IdBrano = s.IdBrano)
4              INNER JOIN Seguaci f ON(f.IdSeguace = h.IdUtente)
5              INNER JOIN Utenti u ON(u.IdUtente = f.IdSeguace)
6 WHERE h.Timestamp BETWEEN ADDDATE(CURDATE(), -7) AND CURDATE()
7     AND u.IdUtente IN (SELECT u.IdUtente FROM Utenti u INNER JOIN Seguaci f ON(f.IdSeguace = u.
          IdUtente) WHERE f.IdUtente = 1)
8     ORDER BY h.Timestamp DESC LIMIT 10;
```

2. Username e numero di volte che è stata ascoltata la canzone Paradox dai follower dell'user id 1

```
1 SELECT COUNT(b.IdBrano) AS Conto, u.Username
2 FROM Brani b INNER JOIN Ascoltate h ON(b.IdBrano = h.IdBrano)
3              INNER JOIN Seguaci f ON(h.IdUtente = f.IdSeguace)
4              INNER JOIN Utente u ON(f.IdSeguace = u.IdUtente)
5 WHERE b.Titolo = 'Paradox' AND f.IdUtente = 1 GROUP BY u.Username ORDER BY Conto DESC;
```

3. Username, titolo e conto delle canzoni piu ascoltate dai follower dell'user id 1

```
1 SELECT u.Username, b.Titolo, COUNT(b.IdBrano) AS Conto
2 FROM Brani b INNER JOIN Ascoltate h ON(b.IdBrano = h.IdBrano)
3              INNER JOIN Seguaci f ON(h.IdUtente = f.IdSeguace)
4              INNER JOIN Utenti u ON(f.IdSeguace = u.IdUtente)
5 WHERE f.IdUtente = 1 GROUP BY b.Titolo ORDER BY Conto DESC;
```

4. Username e numero brani nella collezione dell'utente con più canzoni di genere 'Orchestra'

```
DROP VIEW IF EXISTS ContoBrani;
CREATE VIEW ContoBrani AS
SELECT u.Username, COUNT(b.Genere) as Conteggio
FROM Brani b INNER JOIN BraniCollezione bc ON(b.IdBrano = bc.IdBrano)
             INNER JOIN Collezioni c ON(bc.IdCollezione = c.IdCollezione)
             INNER JOIN Utenti u ON(c.IdUtente = u.IdUtente)
WHERE b.Genere = 'Orchestra' GROUP BY c.IdUtente;
SELECT * FROM ContoBrani HAVING MAX(Conteggio);
```

5. Username e secondi di ascolto dei 3 utenti che ascolta più musica di genere 'Orchestra'

```
DROP VIEW IF EXISTS UtentiGenere;
CREATE VIEW UtentiGenere AS
SELECT u.Username, b.Genere, SUM(b.Durata) AS DurataTotale
FROM Ascoltate a INNER JOIN Utenti u ON(a.IdUtente = u.IdUtente)
                 INNER JOIN Brani b ON(a.IdBrano = b.IdBrano)
WHERE b.Genere = 'Orchestra' GROUP BY a.IdUtente ORDER BY DurataTotale DESC;
SELECT * FROM UtentiGenere LIMIT 3;
```

6. Trova gli utenti che hanno ascoltato un numero di canzoni sopra alla media nell'ultimo mese

```
DROP VIEW IF EXISTS CanzoniAscoltate;
CREATE VIEW CanzoniAscoltate AS
SELECT u.Username, COUNT(a.IdBrano) as Conto
FROM Ascolte a INNER JOIN Brani b ON(a.IdBrano = b.IdBrano)
               INNER JOIN Utenti u ON(a.IdUtente = u.IdUtente)
WHERE a.Timestamp BETWEEN ADDDATE(CURDATE(), -30) AND NOW()
GROUP BY a.IdUtente;
SELECT ca.*
FROM CanzoniUtente ca
WHERE ca.Conto > (SELECT AVG(ce.Conto) FROM CanzoniAscoltate ce) ORDER BY ca.Conto DESC;
```

7. Trova gli utenti e il numero di brani di genere 'Country' nella propria collezione

```
CREATE VIEW Conteggi AS
SELECT u.Username, b.Genere, COUNT(b.IdBrano) AS Conteggio
FROM BraniCollezione c INNER JOIN Brani b ON(c.IdBrano = b.IdBrano)
                       INNER JOIN Collezioni cn ON(c.IdCollezione = cn.IdCollezione)
                       INNER JOIN Utenti u ON(cn.IdUtente = u.IdUtente)
GROUP BY b.Genere, c.IdCollezione;
SELECT Username, MAX(Conteggio)
FROM Conteggi
WHERE Genere = 'Country';
DROP VIEW IF EXISTS Conteggi;
```

8. Trova gli utenti con più di 5 brani nella propria collezione che non hanno mai ascoltato brani country nell'ultimo mese

```
SELECT DISTINCT u.Username
FROM Utenti u INNER JOIN Ascoltate a ON(u.IdUtente = a.IdUtente)
WHERE u.IdUtente NOT IN (SELECT DISTINCT u1.IdUtente
                         FROM Ascoltate a1 INNER JOIN Utenti u1 ON(a1.IdUtente = u1.IdUtente)
                                           INNER JOIN Brani b ON(a1.IdBrano = b.IdBrano)
                         WHERE b.Genere = 'Country')
AND a.Timestamp BETWEEN DATE_SUB(CURDATE(), INTERVAL 30 DAY) AND NOW()
```

```
8  AND u.IdUtente IN (SELECT u2.IdUtente
9                     FROM Utenti u2 INNER JOIN Ascoltate a2 ON(u2.IdUtente = a2.IdUtente)
10                    GROUP BY a2.IdUtente
11                    HAVING COUNT(a2.IdBrano) > 5);
```

6 Interfaccia Web

Per l'interfaccia web è stato seguito un pattern MVC molto rudimentale, che tuttavia ha permesso di semplificarne la realizzazione modularizzando le operazione da effettuare sulla base di dati mediante le pagine.

6.1 Organizzazione e Struttura Generale

La struttura generale dell'interfaccia consiste di 3 cartelle principali e 2 pagine di servizio contenenti rispettivamente un singleton dedicato esclusivamente alla connessione alla base di dati e un singleton dedicato alla creazione e manipolazione delle sessioni. Le cartelle /models, /views, /controllers seguono le tipiche linee guida del pattern MVC, all'interno di /models troviamo infatti i modelli, oggetti atti ad interfacciarsi con la base di dati ed eseguire le query richieste dalle pagine (routes) contenute nei controllers, infine le view, pagine "di template" contenenti per lo più codice HTML e brevi tratti di PHP, vengono popolate mediante le chiamate ai controllers. La navigazione vera e propria tra le pagine avviene mediante parametri GET che si occupano di selezionare il controller richiesto e l'azione da eseguire (funzioni all'interno del controller richiesto).

6.1.1 Esempi

- Richiedere la pagina albums:

```
/basidati/~abaldan/?controller=albums&action=index
```

- Visualizzazione brano con id = 4:

```
/basidati/~abaldan/?controller=songs&action=show&id=4
```

6.2 Pagine Principali

Ci sono 6 pagine principali che consentono la navigazione all'interno dell'interfaccia, accedibili mediante un menù laterale a sinistra. **Home** contiene alcune statistiche sullo stato della BD, ad esempio i brani ascoltati recentemente dai propri followers, questo solo dopo aver effettuato l'accesso con un proprio account registrato, altrimenti in home, come pure in ogni pagina che richiede di essere loggati, viene mostrato un form di login mediantei il quale è anche possibile registrare un account. **Songs** è la pagina adibita alla visualizzazione di tutte le canzoni contenute nella BD o, nel caso di account loggato, offre la possibilità di aggiungere i propri brani alla BD, aggiungerne alla propria collezione o alla coda di riproduzione; **Albums** contiene tutti gli album presenti nella piattaforma, sempre previa autenticazione permette di inserirne di nuovi ed è possibile visualizzare i dettagli di ogni album e brano contenuto in esso. **Collection** e **Playlist** sono rispettivamente le pagine di gestione della propria collezione brani e playlist, con la possibilità di privatizzare o rendere pubbliche le proprie playlist. **Queue** infine ospita la coda di riproduzione, ordinate in base ai timestamp di aggiunta. E' possibile modificare i dati relativi al proprio account, incluso il piano di'iscrizione, utilizzando la pagina accessibile clickando sul bottone in alto a sinistra **settings**, solo dopo aver loggato.

6.3 Mantenimento Stato Pagine

L'interfaccia da la possibilità di ascoltare canzoni come utente visitatore (anonimo), ma per le operazioni più specifiche, ad esempio la creazione e gestione di una personale collezione è necessario registrarsi e loggare utilizzando le credenziali scelte, è stato pertanto creato un sistema di gestione delle sessioni mediante la classe singleton *GrooveSession*, nel file session.php.
Essa contiene i campi dati basilari quali l'id della sessione che si va a creare e l'istanza dell'oggetto che la contiene, e i metodi necessari alla gestione con la possibilità di aggiungere variabili utili.
Alcuni account di prova:

- codep : ciao
- rossi : marco
- verdi : luca

6.4 Note

Trattandosi di un interfaccia "simulativa", in quanto la principale materia d'interesse è la struttura della base di dati su cui poggia, la riproduzione effettiva dei brani non è stata implementata, e non esistono fisicamente file Mp3 caricati all'interno della base di dati, è stato tuttavia implementato un semplice e rudimentale riproduttore in poche righe di javascript atto a dare un'idea dell'effettivo utilizzo che una completa implementazione della piattaforma porterebbe ad avere. Non sono stati scritti controlli di alcun tipo sull'input da parte dell'utente.