HAPPY BEAT

Programació hipermèdia

Georgina Serra Oliveras ls24337
Gustavo Raush ls27447
Alba Molino León ls27119

Índex

Ruta del repositori en Github

https://github.com/graushf/HappyBeat

Descripció de l'aplicació

La nostra aplicació es basa en un reproductor de música.
Un cop oberta, veiem un cercador de cançons i d'artistes relacionats, també podem accedir a un llistat de les cançons més reproduïdes pels usuaris. I de primeres, amb la base de dades buida, no podrem accedir a les cançons més reproduïdes, però un cop s'insereixen sí.

Un cop feta una cerca sobre un artista, podrem veure els seus àlbums i si fem clic a sobre d'algun, veurem un llistat de totes les cançons contingudes en ell mentre s'escolta una preview de la primera cançó d'aquest disc. Un cop fem clic a una de les cançons, s'afegirà a la llista de reproducció, es veurà el seu vídeo i a més a més, es reproduirà una preview de la cançó.
També podrem accedir a qualsevol cançó de la llista de reproducció i tornar-la a reproduir tantes vegades com vulguem.

També tenim l'apartat d'artistes similars. Aquest consta en inserir un nom d'un artista i fer clic a l'enllaç de 'Similar artist' i el que es veurà seran els artistes més similars. Un cop fem clic a l'artista que vulguem, aquest nom directament s'escriurà al 'textarea' per a poder fer la cerca d'aquest en concret.

Referit a la base de dades, cada cop que es reprodueix una cançó, guardarem les seves dades (títol de la cançó, nom de l'artista, nom de l'àlbum, etc.) i si la cançó ja es troba a la base de dades, l'únic que farem serà sumar 1 a la variable de número de reproduccions per a poder saber quines són les més reproduïdes i poder-les incloure a la llista. També, a l'apartat de 'Most replayed songs' el que retorna és un llistat de les 20 cançons més reproduïdes pels usuaris.

Referència a les llibreries externes utilitzades

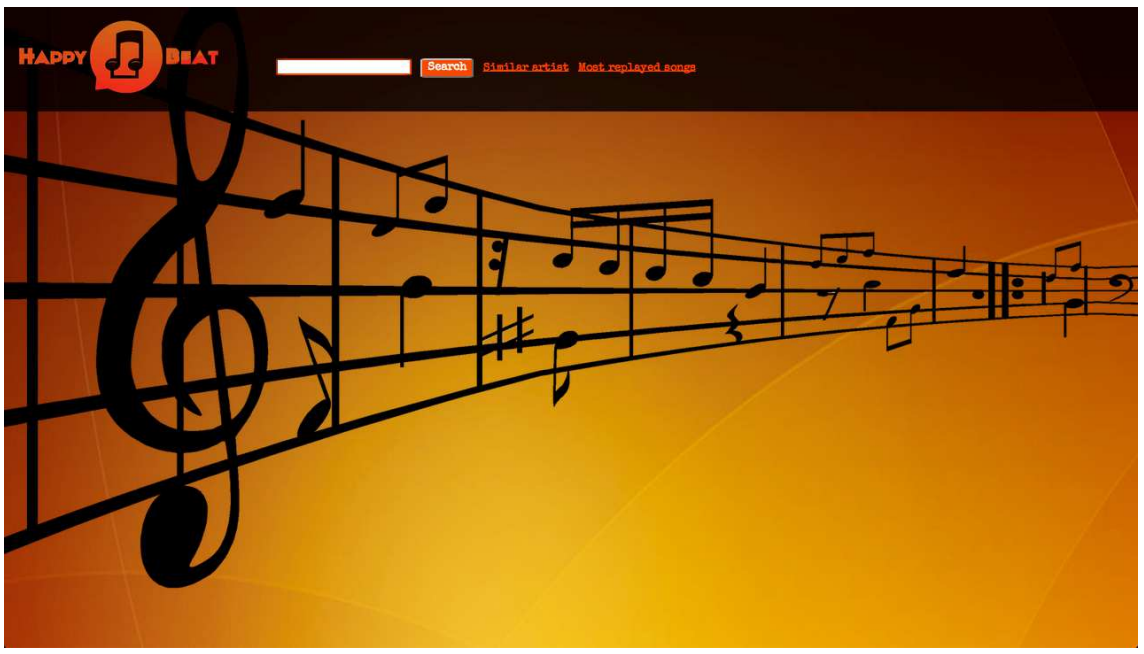
Hem utilitzat 2 tipus de llibreries: les de JQuery (jquery-1.10.1.min.js i jquery.min.js) per a poder utilitzar el codi sobre aquest llenguatge i l'api de Spotify per a poder accedir al seu contingut multimèdia.

Referència al codi Open Source utilitzat

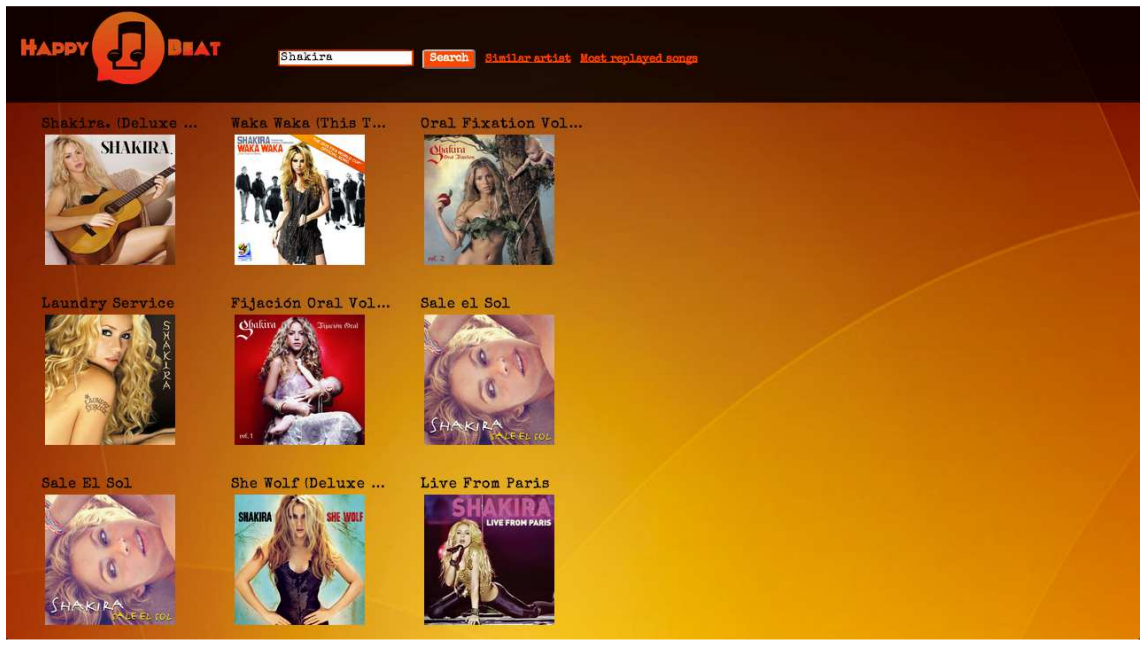
Usem codi Open Source per al tipus de lletra de la nostra aplicació, aquesta és la 'Special Elite'. I també l'usem per a un petit script que ens serveix per a treure el accents del text per a poder-lo inserir a la base de dades sense problemes.

Imatges de l'aplicació

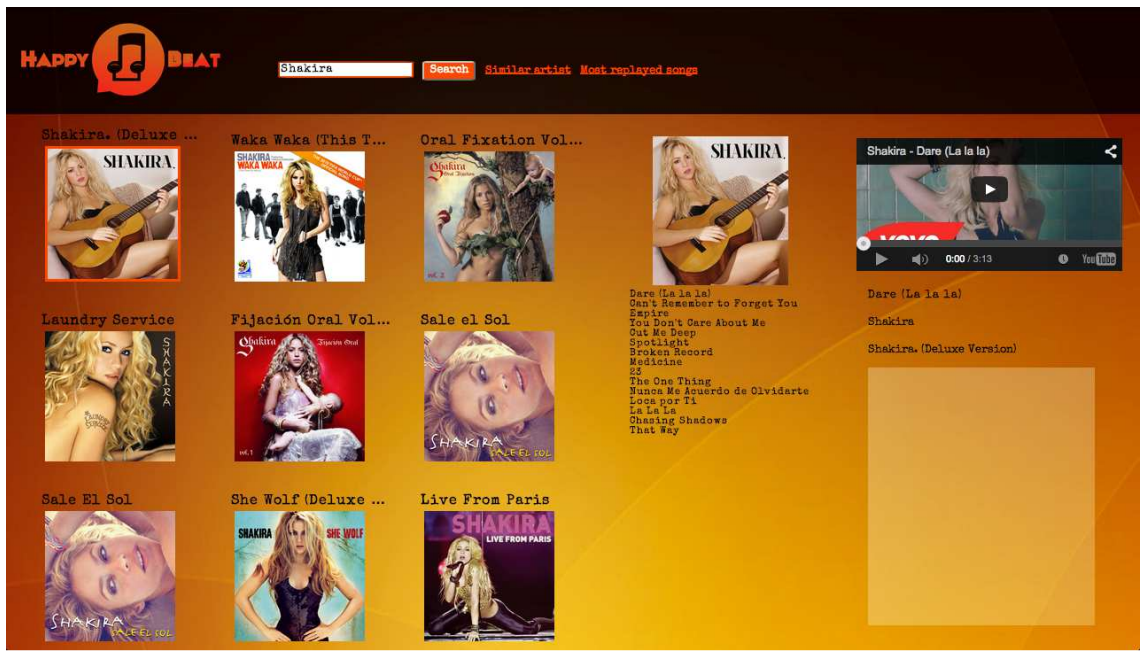
Imatge principal de l'aplicació



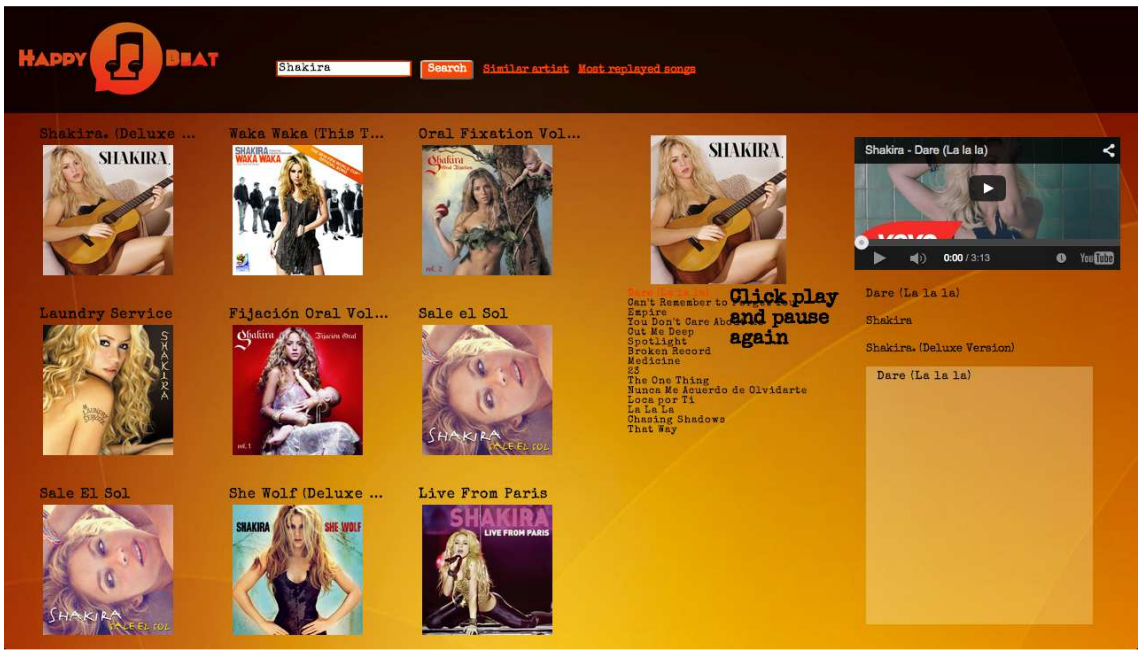
Imatge un cop inserit un artista i premut a cercar



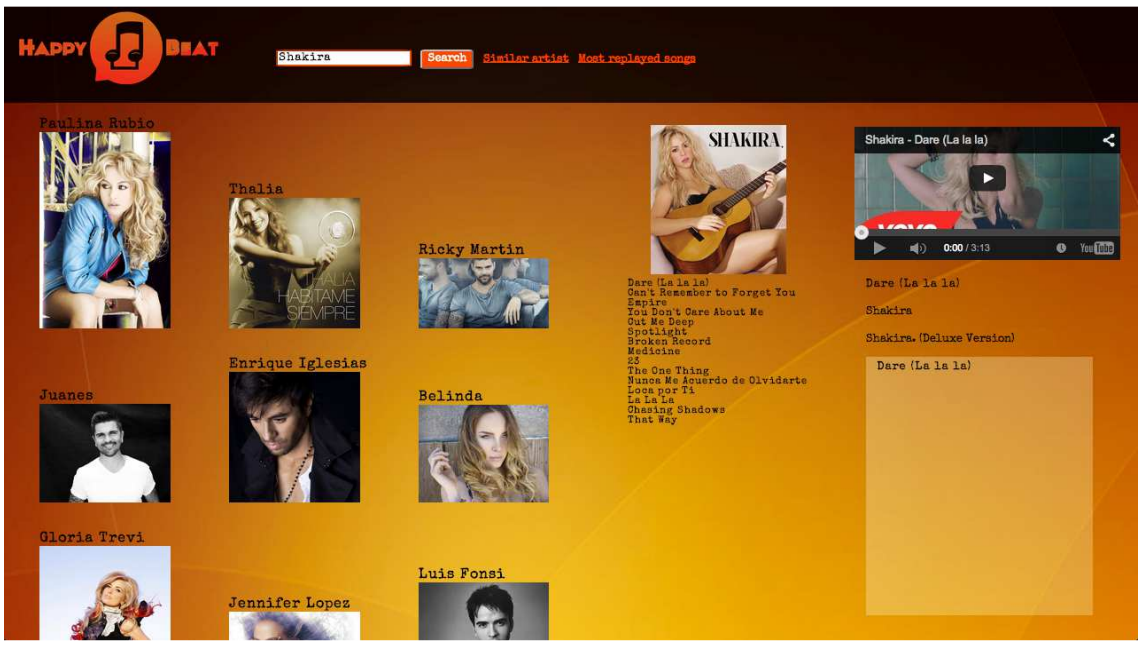
Imatge un cop fem clic a un àlbum en concret de l'artista



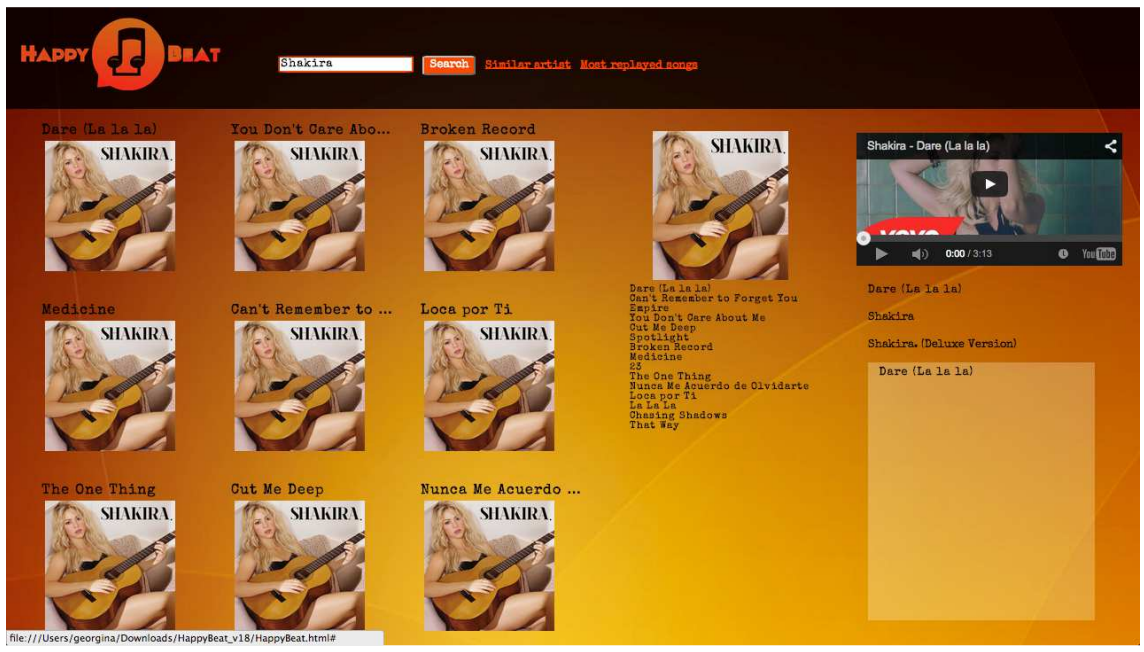
Imatge un cop fem clic a una cançó en concret de l'àlbum de l'artista



Imatge un cop inserit un artista i premut a artistes relacionats



Imatge un cop fem clic a l'enllaç de les cançons més reproduïdes



Explicació de l'estructura de l'aplicació, directoris i fitxers

L'estructura de fitxers de la nostra aplicació és molt senzilla, dins de la carpeta trobem els 3 fitxers de codi que són l'html, el javascript i el css. També, trobem la base de dades ja exportada (hipermedia.sql), les llibreries corresponents (Jquery i Spotify) i una carpeta d'imatges usades a l'aplicació.

A la part de codi de javascript trobem la gran part de programació de la nostra aplicació. Un cop l'usuari ha inserit un artista, el que fem és realitzar una única petició a l'api de Spotify per a que ens retorni tots els àlbums disponibles d'aquest, mentre ens guardem tota la informació referida a l'artista (àlbums, cançons, etc) per a poder mostrar-la més endavant. Un cop s'ha seleccionat un àlbum, el que fem és col·locar la imatge de l'àlbum a la part dreta de l'aplicació mostrant a sota les seves cançons en una llista i també apareix una llista de reproducció buida per a quan fem clic a alguna cançó. Tot seguit, si premem una cançó el que estem fent és inserir-la a la llista de reproducció, mostrar les seves dades (nom, autor i àlbum), buscar a l'api de Youtube (mitjançant el títol de la cançó i el nom de l'autor) el vídeo corresponent a la cançó seleccionada i per últim, emmagatzemar aquesta informació a la base de dades.

Primer de tot, mirarem que aquesta cançó no estigui inserida. Si és així, només sumarem 1 a la variable del número de reproduccions i sinó, la inserirem.

Un cop l'usuari té una llista de reproducció, aquest podrà escoltar cada cançó tant cops com vulgui, el que sí variarà serà que, per cada reproducció tornarem a sumar 1 a la variable de la que parlàvem abans per a anar sabent el número total de reproduccions que té cada cançó.

L'aplicació també té l'opció de veure quines són les cançons més reproduïdes pels usuaris. El que realment estem realitzant és una cerca a la nostra base de dades per tal de veure quines cançons hi ha emmagatzemades i quines són les més reproduïdes. Un cop fet, mostrarem les 20 cançons més reproduïdes. Si pel contrari hi ha inserides menys de 20, les mostrarà totes.

També, hi ha l'opció d'escoltar les cançons que es troben a la llista de més reproduïdes. Un cop fem clic a una de les cançons, aquesta s'insereix a la llista de reproducció i a més a més, sumem 1 a la variable del número de reproduccions.

Tot el codi javascript es troba contingut en una funció anomenada 'window.onload' on fem peticions a les APIs de Spotify i Youtube i les cridem mitjançant funcions de cerca per artista, àlbum i cançó. Fem la connexió a la base de dades de l'api de hipermedia.local/query per a connectar-nos a la nostra base de dades (creada anteriorment amb el phpmyadmin) i fem els diferents 'select' per a poder accedir a ella de manera local. Per últim, també trobem les funcions que ens serviran per a poder mostrar tots aquests resultats per pantalla.

A la base de dades hem creat 3 taules (album, artist i song) unides entre elles i al final es crea una altra taula de la seva relació. Codi de la creació de les taules:

```
CREATE TABLE IF NOT EXISTS `Album` (
  `Id_album` int(11) NOT NULL AUTO_INCREMENT,
  `url_imatge` varchar(150) DEFAULT NULL,
  `titol_album` varchar(100) CHARACTER SET latin7 DEFAULT NULL,
```

```
  PRIMARY KEY (`Id_album`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=69 ;

CREATE TABLE IF NOT EXISTS `Artist` (
  `Id_artista` int(11) NOT NULL AUTO_INCREMENT,
  `Nom_artista` varchar(100) CHARACTER SET latin7 DEFAULT NULL,
  PRIMARY KEY (`Id_artista`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=32 ;

CREATE TABLE IF NOT EXISTS `Relation` (
  `Id_relation` int(11) NOT NULL AUTO_INCREMENT,
  `Id_album` int(11) DEFAULT NULL,
  `Id_artista` int(11) DEFAULT NULL,
  `Id_song` int(11) DEFAULT NULL,
  PRIMARY KEY (`Id_relation`),
  KEY `Id_album` (`Id_album`),
  KEY `Id_artista` (`Id_artista`),
  KEY `Id_song` (`Id_song`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=241 ;

CREATE TABLE IF NOT EXISTS `Song` (
  `Id_song` int(11) NOT NULL AUTO_INCREMENT,
  `titol_song` varchar(100) CHARACTER SET latin7 DEFAULT NULL,
```

```
  `num_reproduccions` int(11) DEFAULT NULL,
  PRIMARY KEY (`Id_song`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=129 ;

ALTER TABLE `Relation`
  ADD CONSTRAINT `PKFK_Album_Relation` FOREIGN KEY (`Id_album`) REFERENCES `Album` (`Id_album`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `PKFK_Artist_Relation` FOREIGN KEY (`Id_artista`) REFERENCES `Artist` (`Id_artista`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `PKFK_Artist_Song` FOREIGN KEY (`Id_song`) REFERENCES `Song` (`Id_song`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Tenir en compte: Un cop l'aplicació fa la petició a l'api de Youtube ens dóna un error de l'extensió kast_sender.js i és problema dels desenvolupadors de Google i no està solucionat.