Lecturer: MSc Alfredo Salazar Vélez

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Campus Monterrey

Final Project

Music Blog

By
Layla Tame || A01192934

Web Applications Development
Semester August – December 2019

Lecturer: MSc Alfredo Salazar Vélez

I. Project description

Purpose of the Application

The music community all over the world is huge and continues to grow. As for some music may just be something they listen to in the background, for others it's a very important aspect of their daily lives. With such a variety of genres, artists and continuous releases, it is sometimes hard to keep up with the music industry.

Although there are many platforms now a days that allow users to listen to music, browse new releases and create their own playlists, it can be overwhelming when trying to decide what albums, artists or songs to listen to.

The purpose of this application is to create a space where music lovers can share their reviews and opinions on new albums, new singles or even on an artist in general. Giving the community a place to share their taste in music so those who share it can find songs fit to their taste.

Scope

For this project, the scope embraces all kinds of users who like to listen to music, regardless the age, location or genre they like. It mainly targets those who actively contribute to the music community be sharing their reviews and opinions and who are constantly up to date with new features. The application will allow users to store their favorite reviews and even search for information on a specific song, album or artist.

Vision

The project's vision is to allow users to increase their knowledge in music while discovering new releases. Create a space where the music community can communicate freely.

Objective

The objectives of the application are as follows:

- Be home to a series of recommendations of any kind of music.
- Allow users to store their favorite reviews according to their taste.
- Incite users to contribute to the community by writing their thoughts on specific albums, songs or artists.
- Provide a search engine where users can browse a specific term in order to get information such as artist, song title, album, release date, duration, number of tracks, amongst others.
- Increase the conscious listening of music in order to appreciate the art behind it.

Lecturer: MSc Alfredo Salazar Vélez

Primary Content of the Web Application

Registration Page
Will allow the user to create an account if not existent. Usernames are unique and an error will be shown if the username entered already exists in the database.

Login Page
User will introduce their username and password. The application will verify the data against the database to ensure they are correct. If the username and password match, a session will start in order to store the user's information and the user will be redirected to the home page. Otherwise, the page will display an error showing that username and password don't match.

Browse Reviews
This space allows the user to browse through the most recent reviews posted by all users. The reviews are shown by latest release date. Each one has an option that allows the user to add it to its favorites, which are also shown on this page. This allows them to have easy and quick access to the posts they've already read and liked.

My Reviews
In order to keep track of your own reviews, this page shows all the reviews the user has posted, ordered by the latest. It also has a space for the user to write a new review, modify and existing one or delete it. It doesn't allow users to delete someone else's posts.

Browse Music
A search engine is shown. A term can be entered, and the user has three options, to look for a specific artist, track or album. A list is shown containing information found on the search term and provides a link to external access in order for the user to be able to listen to the music.

Logout Page
Terminates the current session and redirects to the home page.

Secondary Content of the Web Application

Favorites Attribute
A user can save their favorite posts and come back to read them again later.

Home Page
Its only purpose is to have the user redirected to a page where a "Welcome" message appears, as well as an image of a music concert.

Lecturer: MSc Alfredo Salazar Vélez

II. Requirements

Functionality and usability

Top Priority

- Login
- Register
- Logout
- Add new review
- Delete existing review
- Add review to favorites
- View all reviews
- View favorite reviews
- View user's reviews

Low Priority

- Modify existing review
- Search information on song
- Search information on album
- Search information on artist
- Get external link to visit song/artist page

Style and layout

A title is shown at the top of the page, followed with the navigation bar underneath. The navigation bar shows login/register when the session is not initialized and logout when it is. Navigation bar is the same in all pages but shows a different color on the active page.

When a session is not logged in, there is no access to reviews information, but navigation is still available. An image of a concert is shown with a message to access the login page and initiate session. Once the session is logged in, the reviews are shown instead of the image.

Functional requirements

Use Case	1. Register
Priority	High
Functionality	Application provides a space for the user to create a username and password to access the system.
Precondition	The user is in the register page and there is no current session logged in.

Postcondition	A new username is correctly created and stored in the database
Criteria of Acceptance	• The username entered hasn’t been registered before (it doesn’t exist in the database) • The password is encrypted correctly. • Username and password are correctly stored in the data base.
Alternative Response	• The username is already taken. A message is displayed. • Access to the database was not granted. Error is produced and username is not saved.

Use Case	2. Login
Priority	High
Functionality	User enters its credentials to authenticate their session and have full access to the application.
Precondition	The user is in the login page and there is no current session logged in. The user has a registered account.
Postcondition	Access is correctly granted to the database; a session is initialized. User is redirected to the home page.
Criteria of Acceptance	• The username exists in the database. • The password is encrypted correctly before verifying credentials. • Username and password match. • User has access to full functionality of the application.
Alternative Response	• Username doesn’t exist in the database. A message is displayed. • Username and password do not correspond to a register user. An error message is displayed. • Access to the database was not granted. Error is produced and username is not saved.

Use Case	3. Logout
Priority	High
Functionality	User terminates current session.

Precondition	The user is in the logout page and there is a current session logged in.
Postcondition	Current session is terminated correctly while storing the user's information.
Criteria of Acceptance	• User's information is correctly and safely stored in the database for future reference. • Session is terminated. • User is redirected to the login page. • User no longer has access to reviews.
Alternative Response	• Session is not terminated correctly. User still has access to full functionality. • Information is not stored correctly in the database for future reference.

Use Case	4. View all reviews
Priority	High
Functionality	All reviews written by all users are displayed in the Reviews page.
Precondition	The user is logged in and in the Reviews Page.
Postcondition	All reviews are shown.
Criteria of Acceptance	• All users' reviews are shown. • Posts are ordered by latest release date. • Posts show author, title, date, id and content correctly.
Alternative Response	• There are no reviews in the database. Nothing is shown. • Access to the database was not granted, an error is produced.

Use Case	5. Add to favorites
Priority	Medium
Functionality	User is able to add a review to their favorite's list.
Precondition	The user is logged in and in the Reviews Page. There is at least one review in the database. All reviews are shown in the reviews page.

Postcondition	Review is added and shown in favorite's section.
Criteria of Acceptance	• Review is correctly stored in favorite's collection. • Posts are ordered by latest release date. • Posts show author, title, date, id and content correctly. • Review is displayed in the favorite's section. • "Add to favorites" button changed color and label.
Alternative Response	• Review is not added to the database correctly. • Favorite reviews are not shown in the corresponding section. • Access to the database was not granted, an error is produced.

Use Case	6. Delete from favorites
Priority	Medium
Functionality	User is able to delete a review from their favorite's list.
Precondition	The user is logged in and in the Reviews Page. There is at least one favorite review. All reviews are shown in the reviews page.
Postcondition	Review is deleted from database and favorite's section.
Criteria of Acceptance	• Review is correctly deleted from favorite's collection. • Review is removed from the favorite's section. • All other posts are still shown.
Alternative Response	• Review is not deleted from the database correctly. Favorites' section doesn't change. • Access to the database was not granted, an error is produced.

Use Case	7. Post New Review
Priority	High
Functionality	User can add a new review.

Precondition	The user is logged in.
Postcondition	Review is added to the database and shown in Reviews Page.
Criteria of Acceptance	• Information is correctly sent and saved in database. • All fields are correctly filled out. • Review's section is updated to show the new review on top. • Author and date are automatically generated by server.
Alternative Response	• All fields are not filled out. Post is not allowed. • Access to the database was not granted, an error is produced.

Use Case	8. Delete a Review
Priority	High
Functionality	User is able to delete a review they wrote.
Precondition	The user is logged in and has written at least one review. ID provided corresponds to a post written by the current logged user.
Postcondition	Review is deleted from database and is no longer shown in the application.
Criteria of Acceptance	• ID provided matches an existing post in the database. • Review is correctly deleted from database. • Review is removed from the Reviews' section. • All other posts are still shown.
Alternative Response	• ID doesn't match any post in the database. An error is shown. • Review is not deleted from the database correctly. Reviews' section doesn't change. • Access to the database was not granted, an error is produced.

Use Case	9. Modify a Review
Priority	Medium

Functionality	User is able to modify a review they wrote.
Precondition	The user is logged in and has written at least one review. ID provided corresponds to a post written by the current logged user.
Postcondition	Review is updated in the database and shows changed in the Reviews' Page.
Criteria of Acceptance	• ID provided matches an existing post in the database. • Review is correctly updated in database. • Review's information is updated in the Review's Page
Alternative Response	• ID doesn't match any post in the database. An error is shown. • No fields are filled out. No changes are produced. • Access to the database was not granted, an error is produced. No changes are made to the post.

Use Case	10. View My Posts
Priority	High
Functionality	User is able to view all the posts they have written.
Precondition	The user is logged in and has written at least one review.
Postcondition	All available reviews are displayed in "My Reviews" Page
Criteria of Acceptance	• Reviews are correctly shown in page, showing all attributes. • Reviews shown correspond to the current logged in user.
Alternative Response	• User has written no posts. Nothing is showed. • Access to the database was not granted, an error is produced.

Use Case	11. Retrieve Music Information
Priority	Low

Functionality	User can search for a term matching an artist, song or album.
Precondition	The user is in the "Browse Music" page.
Postcondition	Found information is displayed in correct order.
Criteria of Acceptance	• Connection to Deezer's API was successful. • There is information corresponding to the search term. • Complete information is displayed correctly corresponding to the results. • When results include an image, it is displayed correctly. • When results contain a link, this opens a new tab with the music player.
Alternative Response	• API connection failed. An error is shown. • Search term had no matching results. Nothing is shown.

Non-functional requirements

1. Design: the whole application follows the same patterns, uses the same colors and styling.
2. Usability: the application must be friendly and easy to use.
3. Interface: the application must run correctly on different web browsers.
4. Security: the user's information must be stored securely, passwords must be treated sensibly and stored encrypted. The application should only display Review's information if there is a logged session.
5. Performance: response time must be as efficient as possible.

III. Specifications

Functional Specifications

The necessary functions used to build the system, specifying their input parameters if applicable:

- Init( )
    - Loads all posts from the database and appends them to the Review's page.
    - Loads all favorite posts from the logged user and appends them to the Favorite's section.

- Reload( )
    - Updates posts information on the application when changed are made to the database.

- initMyPosts(username: String)
    - Loads all posts written by the logged in user. Receives the username of the active session as a parameter to retrieve posts by that author.

- registerSubmit(username: String, password: String)
  - Create a new user. Verify username doesn't already exist in database.

- loginSubmit(username: String, password: String)
  - Verifies username exists and is consistent with password. Initializes a session corresponding to the username entered.

- postBlog(title: String, content: String, publishDate: Date, button: String)
  - Adds a new review to the database

- deleteFavorite(postId: String)
  - Deletes a review from the favorites' collection.

- addFavorite(postId: String, title: String, content: String, publishDate: String, button: String)
  - Stores in the database relevant information of review added to favorites.

- deleteReview(id: id)
  - Deletes a review from the database.

- updateBlog(id: id, content: Object)
  - Updates the title or content of a review in the databse based on the ID provided.

- searchMusic(searchTerm: String)
  - Detects button clicked, gets artist, song or album ID and makes a call to corresponding function.

- searchArtist(idArtist: Number)
  - Calls Deezer's API to retrieve information on an artist based on its unique ID. Returns information retrieved to be displayed.

- searchAlbum(idAlbum: Number)
  - Calls Deezer's API to retrieve information on several albums that match the search term based on their unique IDs. Returns information retrieved to be displayed.

- searchTrack(idTrack: Number)
  - Calls Deezer's API to retrieve information on several songs that match the search term based on their unique IDs. Returns information retrieved to be displayed.

- logout( )
  - Terminates active session

Lecturer: MSc Alfredo Salazar Vélez

Design Specifications

The font was consistent throughout the application. Verdana sans-serif was used. All pages must display the same navigation bar and page title on top of the page. The selected page is shown in a different color. The colors chosen will be used throughout the whole application. The chosen palette consists of:

- CSS predetermined colors
    - white, red, black, darkgray, gray.

- Customized colors:
    - Green - #56C56F
    - Red - #D9534F
    - Gray - #1c1c1b
    - Cream – F1F1F1

All reviews are shown with a border that delimits each one. The format and order in which the attributes are displayed is consistent among the different pages. Newest posts are always shown on the top of the page. The grid and layout were designed using plain html and css through the implementation of flexbox.

Technical Specifications

Front-End

The front end was designed through html and css, connecting to the script through the DOM. Ajax was used to communicate between the script and the server.

Back-End

The back end was developed using Node.js framework and express. To create the login and keep track of the session, the session-express library was used. The bcrypt library was used to encrypt and decrypt users' passwords. An external API (Deezer's public API) was used to create a search engine specifically for music information. Mongoose was used to connect the server with the database used, that in this case is MongoDB. Heroku was used for deploying the application.

IV. System Architecture

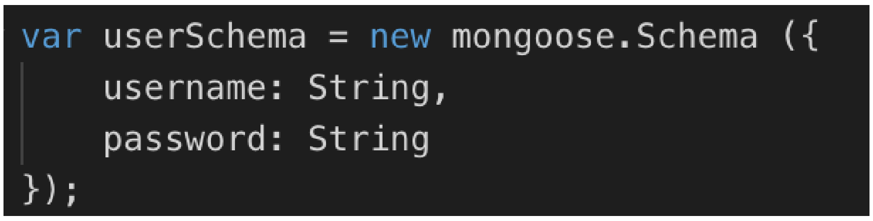
Software architectural pattern

The architectural pattern used is Model View Controller, in order to correctly divide the functionality of each of the three main logical components of the application.

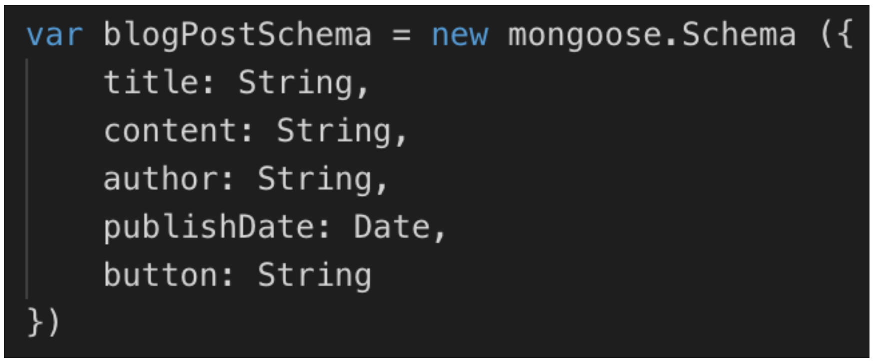
- Model: Data scheme that connects directly to the database. This is the component where elements from the database are accessed and manipulated.

- View: the component that the final user interacts with. It covers all the user interface design with which the user will communicate with the controller.
- Controller: serves as an intermediary between the view and the model components. Layer of the application that is used to manipulate the data obtained as a response from the model according to the interaction the user had with the view component.

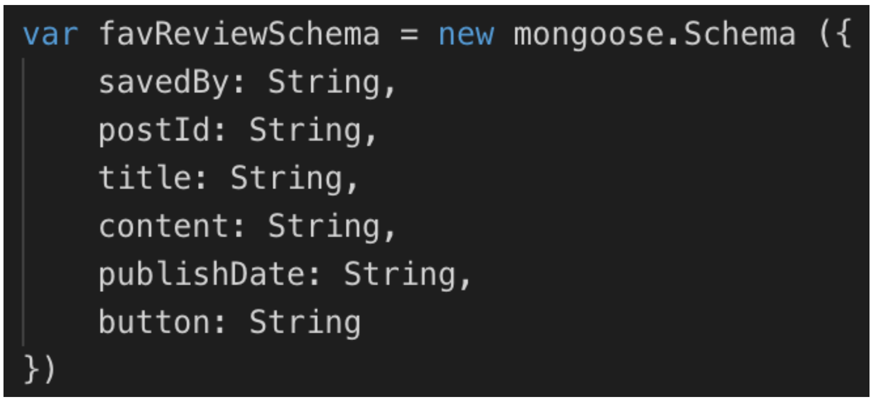
On the database side, several collections were created to better classify and organize the data. The collections and their respective schemes are as follows:



```
var userSchema = new mongoose.Schema ({
    username: String,
    password: String
});
```



```
var blogPostSchema = new mongoose.Schema ({
    title: String,
    content: String,
    author: String,
    publishDate: Date,
    button: String
})
```



```
var favReviewSchema = new mongoose.Schema ({
    savedBy: String,
    postId: String,
    title: String,
    content: String,
    publishDate: String,
    button: String
})
```