Alfonso De La Guardia **alfonsodelag1@gmail.com**

Music Library

11th December 2019 - 21th December 2019

Overview

Using the itunes Search API to make music searches, by using Jquery, Ajax and JSON.

Table of Contents

Project information

Project Requirements

Depending on the search performed, you should be able to see for each item found the following information as a minimum:

- For a song:
    - Cover
    - Name of the song
    - Artist name
    - Album name
    - Song Price
    - Release date
    - Song length
    - Musical genre
    - Audio sample of the song
    - Song Link in iTunes
- For an artist:
    - First name
    - Musical genre
    - Artist link on iTunes
- For an album:
    - Cover
    - Album name
    - Artist name
    - Album price
    - Number of songs
    - Release date
    - Musical genre
- For a music video:
    - Cover
    - Name of the song
    - Artist name
    - Song Price

- Release date
- Song length
- Musical genre
- Clip video sample
- Link of the music video on iTunes

Project Specifications

- Create a clear and orderly directory structure
- Use jQuery for the entire project development
- Implement all functionalities on the same screen dynamically
- Make use of third-party API's to solve the project
- Additional search parameters will be optional when performed, may or may not be applied individually (visual example in the wireframe)
- The web will have to be responsive and compatible with the main browsers in the market
- The search must be carried out each time the input changes, that is, each time a character is entered, the page will show the content found.
- Any item found by a search can be added to favorites and will be displayed in the favorites section (example in the wireframe)
- Items previously found may be deleted from favorites
- Favorite items should be stored in localStorage
- All code included comments need to be written in English
- Use a code style like camelCase
- In the case of using HTML never use inline styles
- In the case of using different programming language always define the implementation in separate terms
- It is recommended to divide the tasks into several subtasks so that you can associate each particular step of the construction with a specific commitment.

- For the project documentation a PDF version is required within the repository
- You should try as much as possible that the commits and planned tasks are the same
- Delete unused files.

Risk Management Plan

Every project has risks. This risks must be taken into account to improve the workflow of the project. I've listed the risks for the project, along with the impact they might have, and the priority of them.

iD	Risk	Consequence	Prob. (1-5)	imp. (1-5)	Pri. (1-25)	Mitigation approach
1	Breaking my computer	Can't do anything	1	5	5	Keep my repo work to date, look for other computers
2	Getting sick	Wouldn't be as productive	2	3	6	Eat and sleep well
3	Not concentrating enough	Won't com	2	5	10	Focus on the Minimum Viable Product.
4	Unrealistic deadlines	Deadlines wouldn't be met + development shortcuts would have been taken affecting the robustness of the code	2	3	6	Be more organized with the tasks, set new deadlines.
5	Being unfocused	Loss of control over the development flow of the project	4	5	20	Focus on finishing the most important tasks only.

Tasks for the project

Defining this part is crucial to the development of the project. It is important to make a good analysis of the situation to organize the project in a good way.

1. Project organization
    - Priority: Medium.

- Description: Getting together with the team to analyze the project.
- Difficulty: 2/10
- Time estimation: 1 hr

2. Client requirements:
    - Priority: Medium.
    - Description: Reading the description of the project to figure out the requirements.
    - Difficulty: 3/10
    - Time estimation: 30 min.

3. Creating Git Repo
    - Priority: High.
    - Description:Creating the git repo for the project.
    - Difficulty: 1/10
    - Time estimation: 5 min.

4. Study and research
    - Priority: High.
    - Description: Understand how the Itunes API works.
    - Difficulty: 3/10
    - Time estimation: 30 min.

5. Preparing basic documentation
    - Priority: High.
    - Description: Prepare the basic documentation for the project.
    - Difficulty: 5/10
    - Time estimation: 2 hrs.

6. Creating objects, classes and functions
    - Priority: High.
    - Description: Determine which classes will be used for the project.
    - Difficulty: 8/10
    - Time estimation: 4 hr

7. API Testing
    - Priority: High.
    - Description: Learning how to get data from the API and testing it.
    - Difficulty: 5/10
    - Time estimation: 2 hr.

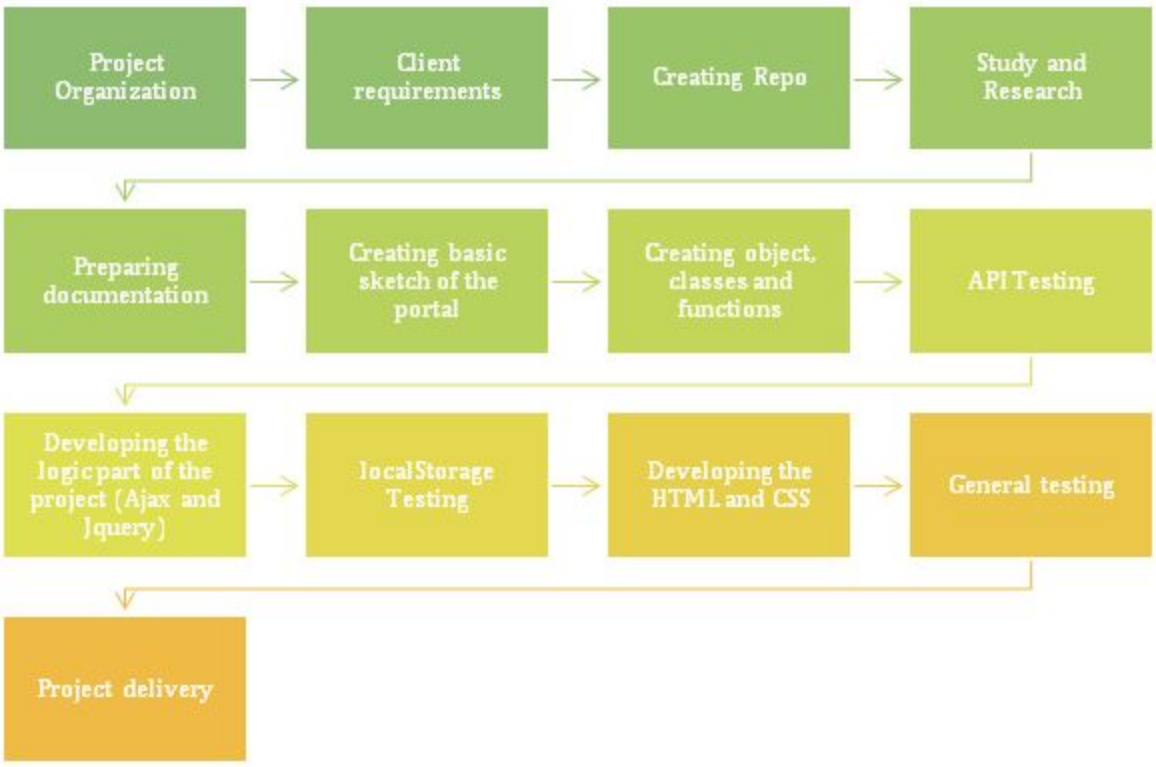
8. Developing the logic of the project (AJAX and Jquery)
    - Priority: High.
    - Description: Working on AJAX and Jquery for the development of the project.
    - Difficulty: 8/10

- Time estimation: 4 hr

9. LocalStorage testing
    - Priority: High.
    - Description: Develop the functions to save the data in localStorage.
    - Difficulty: 7/10
    - Time estimation: 4 hr

10. Developing HTML and CSS
    - Priority: High.
    - Description: Give more detail and functionality to the basic sketch of the portal.
    - Difficulty: 5/10.
    - Time estimation: 2 hr.

11. General testing
    - Priority: High.
    - Description: testing the project to see if it's working well.
    - Difficulty: 4/10
    - Time estimation: 4 hr



Chronogram

Calendar

- **Wednesday:**
  -Get together with teammates.
  -Analyze the project, think about the organization of the project.
  -Get together on the daily reunion with classmates.

- **Thursday**:
  -Prepare the documentation for the project.
  -Create the Git repository.
  -Create the HTML, CSS and JavaScript files.
  -Get together on the daily reunion with classmates.

- **Friday**:
  -Creating the basic structure of the HTML file
  -Adding basic styles to HTML
  -Conceptualizing the classes that will be used.
  -Get together on the daily reunion with classmates.

- **Saturday:**
  -Work on creating the search parameters for the music.
  -Studying the Itunes API.
  -Understanding how to work with

- **Sunday:**
  -Using ajax for making petitions to the API
  -Improving the classes created

- **Monday:**
  -Using switch with the entity cases
  -Add more style to the document

- **Tuesday:**
  -Add more styling to the project.
  -Improving the styling of the website.

- **Wednesday:**
  -Using the API of Countries to generate the countries parameter options.
  -Working on functions for the project.

- **Thursday**
  -Creating functions to add entity contents to the HTML dynamically.

- **Friday**
  -Creating functions to add entity contents to the HTML dynamically
  -Saving information in LocalStorage.

- **Saturday**
  -Removing the information from Favorites section.
  -Making the website more responsive.

Git Workflow

For this project, all commits we'll be pushed directly to the Master branch. All commits will use a descriptive message, so that myself or other users can easily go to the Git version that they need to. This is very important for working in teams as it increases communication and efficiency between all members.

Technologies used

For this project, we will use the following technologies:

- HTML and CSS.
- Jquery.
- Bootstrap CSS.
- AJAX.
- JSONP.

File structure

The files will be organized in the following way:

musiclibrary/
  .git/ This folder contains the Git information for the project
  assets/

images/	This folder contains all the images needed for the project
js/	This folder contains the JavaScript file for the project
js/script.js	This file contains all the functions that make the webpage work
js/entities/Album.js/	This file contains the JavaScript file for the album class
js/entities/Artist.js/	This file contains the JavaScript file for the artist class
js/entities/Search.js/	This file contains the JavaScript file for doing searches.
js/entities/Song.js/	This file contains the JavaScript file for the song class.
js/entities/Video.js/	This file contains the JavaScript file for the video class.
css/	This file contains the CSS file for the project.
index.html	This is the main HTML for the project
README.md	This contains information on how to use the project.

Classes used to represent the data

Project Validation

Quality Metrics

The quality metrics for this project will be the following:

- Google Chrome's Audit.
- We will also use ESLint for validation of the JavaScript.
- W3S validator for finding errors.
- Compatibility with other browsers.
- Using camelCase for the project.

Lessons learned

- Using the .append method along with algorithms you can create and display all kinds of html templates using Jquery.
- You can create infinite number of Id's using algorithms, this can help you store more content in localStorage.
- Ajax is very useful because it lets for example read and save data from a web server

Incidents

- Difficulties creating classes.
- Difficulties for saving data into localStorage
- Difficulties for Deleting favorites.

Class Architecture

This class saves all the inputs of the parameters for a Search, like entity, terms written by the user, country, if the content is explicit or not, and content limit.

Search
Entity
Term
Country
Explicit
Limit

The Artist Class saves all the elements from the API when the users looks for an artist.

Artist
Artist Name
Musical Genre
Artist Link

The Song Class saves all the elements from the API when the users looks for an artist.

Song
Cover
Song Name
Artist Name
Album Name

Song Price
Release Date
Song Length
Musical Genre
Audio Samples
Song Link

The Song Class saves all the elements from the API when the users looks for an artist.

Video
Cover
Song Name
Artist Name
Album Name

Song Price
Release Date
Song Length
Musical Genre
Video Sample
Video Link

The Album Class saves all the elements from the API when the users looks for an artist.

Video
Cover
Album Name
Artist Name
Album Price

Song Number
Release Date
Musical Genre