Music Library

Project objective

- Understanding how to consume the API publishes ITUNES
- Make Ajax requests with the jQuery library
- Improve your knowledge in JavaScript, HTML and CSS
- Improve your knowledge in logic and programming
- Improve JSON requests

Project requirements

The interface must:

1. It consists of a single page that dynamically updates its content.
2. Perform search queries in the iTunes Search API each time the search field changes.
3. Let the user search for:
   - Songs by name.
   - Artists by name.
   - Albums by name.
   - Music videos by name.
4. Allow the user to add and remove items from a personal favorites list.
5. Allow the user to filter the search results:
6. By country.
7. In case it contains explicit content or not.
8. Limit the number of results, ranging from 1 to 200.
   - Displays favorite content in the Favorites section.
   - Display the results along with specific information about each item, such as:
     - For songs:
       - Cover
       - Song name
       - Artist's name
       - Album name
       - Song price
       - Release date
       - Song duration
       - Musical genre
       - Audio sample
       - Song link in iTunes
     - For an album:
     - For artists:
       - Artist's full name
       - Musical genre
       - Artist link on iTunes
     - For music videos:
       - Cover
       - Song name
       - Artist's name
       - Song price
       - Release date
       - Song duration
       - Musical genre

- Cover
- Album name
- Artist's name
- Album price
- Number of songs
- Release date
- Musical genre
- Album link on iTunes
- Video sample
- Music video link on iTunes

Project specifications

1. The project must be developed using **jQuery**.
2. The directory structure of the project must be well defined and organized.
3. Search queries must be made through the iTunes Search API.
4. The list of countries required for the filtering process must be obtained through a third-party API.
5. Personal favorites should be stored in **localStorage**
6. The code must be documented correctly using the English language.
7. Your code must use a **camelCase style.**
8. HTML must not contain inline styles.
9. The project must be developed using *git*, using explicit and concise confirmationmessages.

The project must contain a *README* file written in *Markdown* that shows a brief description and the steps to run it.

LISTA DE TAREAS A REALIZAR

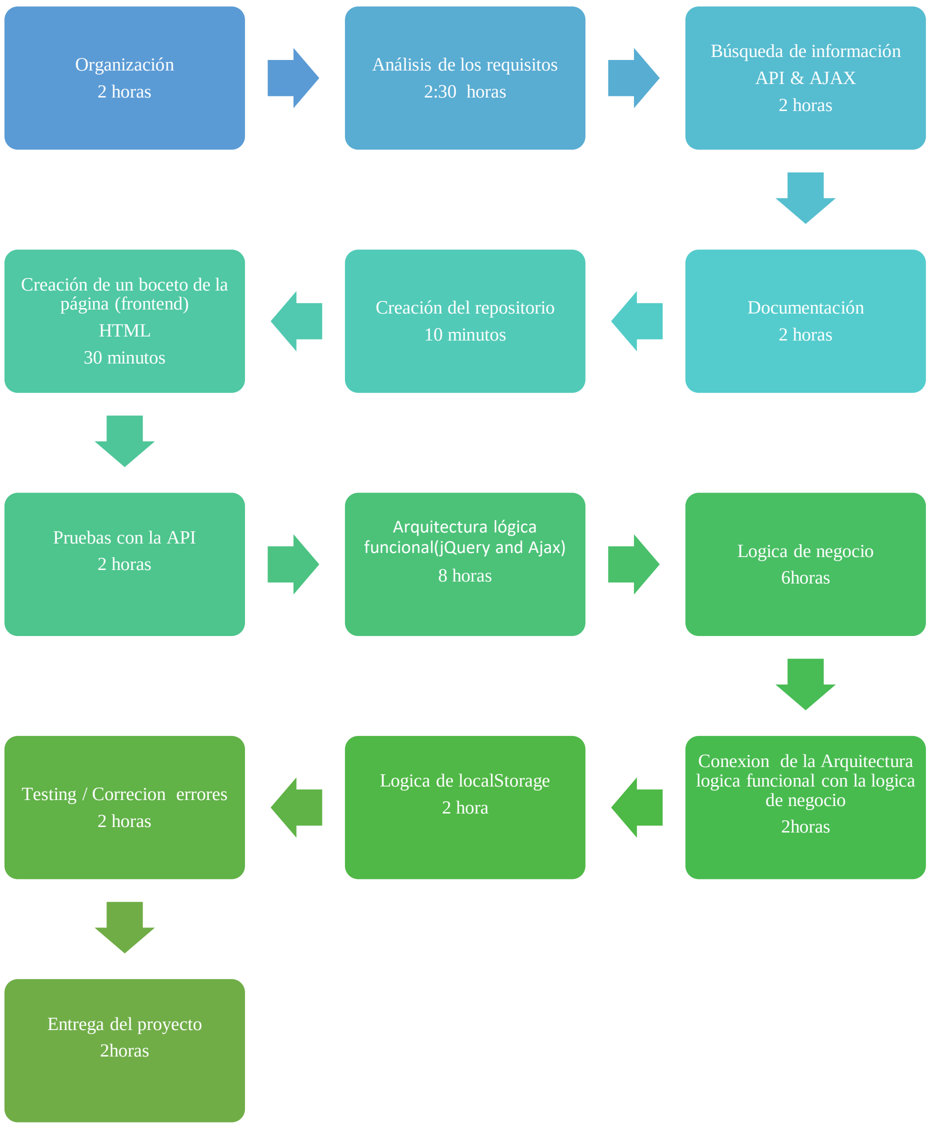
Task	Priority	Hours	Difficulty
Documentation	High	4,00	High
Organization	High	3,00	High
Pre-search for information	Normal	2,00	Normal
Repository creation	Low	0,15	Low

Documenting Project	Normal	4,00	Normal
Index HTML structure	Low	0,30	Low
Testing with the Api	Normal	1,00	Normal
Functional Logical Architecture(jQuery and Ajax)	High	8,00	High
HTML and Css Styles Dynamic JS	High	8,00	Normal
Local Storage functionality	Normal	2,00	Normal
Daily Meetings	Normal	5,00	High
CREATION README	Low	2,00	Low
Testing / Correction Errors	High	2,00	High

We plan to finish the project in 41. 45 Hours

In which we give 8 more hours to anticipate incidents in the project

Project Calendar Tracking



Quality Metrics

Although the project must adhere to all project *requirements* and *specifications,* there are some conditions that, if properly met, add a sense of quality and robustness to the project itself. These conditions are:

1. HTML and CSS code must be validated by W3C.
2. JavaScript code must be lint-free, as indicated by a trusted JavaScript linter, such as ESLint.
3. The web application must respond.
4. The web application must be compatible with the main browsers on the market:
    - Internet Explorer 11 o superior.
    - Safari in one of its latest versions.
    - Firefox in one of its latest versions.

- Chrome in one of its latest versions.

Risk Documentation

- Project delays.
- Loss or damage to work material

GIT WORKFLOW documentation

- Creating https://github.com/robertfox11/MusicLibrary.git Hub
- We make commits of the structure of the main page.
- Probability of a occuring 80%
- Project impact 60%
- Possible alternative (mitigation) Ask colleagues for help
- Chance of it occurring 30%
- Project impact 60%
- Possible alternative (mitigation) Ask colleagues for help
- Not easily finding information related to the project
- Chance of it occurring 30%
- Project impact 60%
- Alternative alternative (mitigation)
- Ask colleagues for help

From the realization of the structure, work continued only on the "master" branch, through the Workflow "Gitflow".
But information --> https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow

Project tooling

Different tools were used in the development of the project. They are as follows:

1. ***git: A powerful version control system*** that helps track changes in the work tree.
2. ***Visual Studio Code: A code editor*** optimized for creating and debugging modern web applications.
3. ***Google Chrome Developer Tools:*** Used to debug JavaScript code and to test design settings.
4. ***Google Docs:*** Used to write project documentation.
5. ***W3C Validator*–** Used to validate HTML and CSS code.
6. ***ESLint*–** Used to validate JavaScript code.

Git workflow

All commits will be inserted into the master branch, following a personal criterion of loading only snapshots that are functional and working correctly, not counting minor errors. There are no other branches, as it would slow down the development process.

- On the other hand, confirmation messages end with their primary purpose indicated in square brackets: for example.
- The main relationships for the functional logic is [class]
- commits related mainly to CSS changes begin with ***[styling]***;
- those related mainly to page layout, ***[layout]***;
- Those related to the project by adding folders or documentation, plugins, library ***[project]***;
- mainly related to documentation, ***[documentation]***.

File structure

Project files will be organized as follows:

```
music-library/
    .gitignore     Folder used by git to contain information about the
                   repository.
    assets/
        img/       Folder containing all the images used in the interface.
        js/        Folder containing all scripts used in the user interface.
        css/       Folder containing all styles used in the user interface.
    index.html     Web application home page.
    README.mdFile that contains instructions on how to run the project.
```

Record of lessons learned.

- Making a more detailed documentation improvement
- Class structure
- Make an API request in AJAX with JQUERY
- Callback function