### Music Library

Objetivo del proyecto

* Entender cómo consumir la API publica ITUNES
* Realizar peticiones Ajax con la librería jQuery
* Mejorar tus conocimientos en JavaScript, HTML y CSS
* Mejorar tus conocimientos en lógica y programación
* Mejorar peticiones JSON

Requisitos del proyecto

La interfaz debe:

1. Consiste en una sola página que actualiza su contenido dinámicamente.
2. Realice consultas de búsqueda en la API de búsqueda de iTunes cada vez que cambie el campo de búsqueda.
3. Deje que el usuario busque:

|  |  |
| --- | --- |
| * Canciones por nombre. * Artistas por su nombre. | * Álbumes por nombre. * Videos musicales por nombre. |

1. Permita que el usuario agregue y elimine elementos de una lista de favoritos personales.
2. Permita que el usuario filtre los resultados de la búsqueda:
3. Por país.
4. Por si contiene contenido explícito o no.
5. Limitar el número de resultados, que van de 1 a 200.

* Muestra los contenidos favoritos en la sección de favoritos.
* Mostrar los resultados junto con información específica sobre cada elemento, como:

|  |  |
| --- | --- |
| * + Para canciones:     - Caratula     - Nombre de la canción     - Nombre del artista     - Nombre del álbum     - Precio de la canción     - Fecha de lanzamiento     - Duración de la canción     - Género musical     - Muestra de audio     - Enlace de canción en iTunes   + Para un álbum:     - Caratula     - Nombre del álbum     - Nombre del artista     - Precio del álbum     - Número de canciones     - Fecha de lanzamiento     - Género musical     - Enlace del álbum en iTunes | * Para artistas:   + Nombre completo del artista   + Género musical   + Enlace de artista en iTunes * Para vídeos musicales:   + Caratula   + Nombre de la canción   + Nombre del artista   + Precio de la canción   + Fecha de lanzamiento   + Duración de la canción   + Género musical   + Muestra de vídeo * Enlace de vídeo musical en iTunes |

## Especificaciones del proyecto

1. El proyecto debe desarrollarse utilizando **jQuery**.
2. La estructura de directorios del proyecto debe estar bien definida y organizada.
3. Las consultas de búsqueda deben realizarse a través de iTunes Search API.
4. La lista de países necesarios para el proceso de filtrado debe obtenerse a través de una API de terceros.
5. Los favoritos personales deben almacenarse en **localStorage**
6. El código debe documentarse correctamente utilizando el idioma inglés.
7. El código debe usar un estilo **camelCase.**
8. El código HTML no debe contener estilos en línea.
9. El proyecto debe desarrollarse mediante *git*, utilizando mensajes de confirmación explícitos y concisos.

El proyecto debe contener un archivo *README* escrito en *Markdown* que muestre una breve descripción y los pasos para ejecutarlo.



|  |  |  |  |
| --- | --- | --- | --- |
| **Tarea** | **Prioridad** | **Horas** | **Dificultad** |
| Documentación | Alta | 4,00 | Alta |
| Organización | Alta | 3,00 | Alta |
| Búsqueda Previa de información | Normal | 2,00 | Normal |
| Creación de repositorio | Baja | 0,15 | Baja |
| Documentando Proyecto | Normal | 4,00 | Normal |
| Estructura HTML índex | Baja | 0,30 | Baja |
| Pruebas con la Api | Normal | 1,00 | Normal |
| Arquitectura lógica funcional(jQuery and Ajax) | Alta | 8,00 | Alta |
| HTML and Estilos Css Dinámico JS | Alta | 8,00 | Normal |
| Funcionalidad Local Storage | Normal | 2,00 | Normal |
| Reuniones Diarias | Normal | 5,00 | Alta |
| Creación README | Baja | 2,00 | Baja |
| Testing / Corrección Errores | Alta | 2,00 | Alta |

Tenemos previsto terminar el proyecto en 41, 45 Horas

En las cual damos 8 horas más para prever incidencias en el proyecto

Registro De Incidencia que se ha detectado durante el proyecto

Seguimiento Calendario del proyecto

Cronograma del proyecto diarios por horas

Métricas de Calidad

A pesar de que el proyecto debe adherirse a todos los [*requisitos*](https://docs.google.com/document/d/1dtnHybW7qhBAuuuNEZi1Pb0neJEqbME-hXvGSw2XHUI/edit#heading=h.rf0k2roeckt1) y [*especificaciones*](https://docs.google.com/document/d/1dtnHybW7qhBAuuuNEZi1Pb0neJEqbME-hXvGSw2XHUI/edit#heading=h.xr3uz2kyop9d)del proyecto, hay algunas condiciones que, si se cumplen adecuadamente, añaden una sensación de calidad y robustez al propio proyecto. Estas condiciones son:

1. El código HTML y CSS debe ser validado por W3C.
2. El código JavaScript debe estar libre de pelusas, como lo indica un linter de JavaScript de confianza, como ESLint.
3. La aplicación web debe responder.
4. La aplicación web debe ser compatible con los principales navegadores del mercado:
   * Internet Explorer 11 o superior.
   * Safari en una de sus últimas versiones.
   * Firefox en una de sus últimas versiones.
   * Chrome en una de sus últimas versiones.

Documentacion de Riesgos

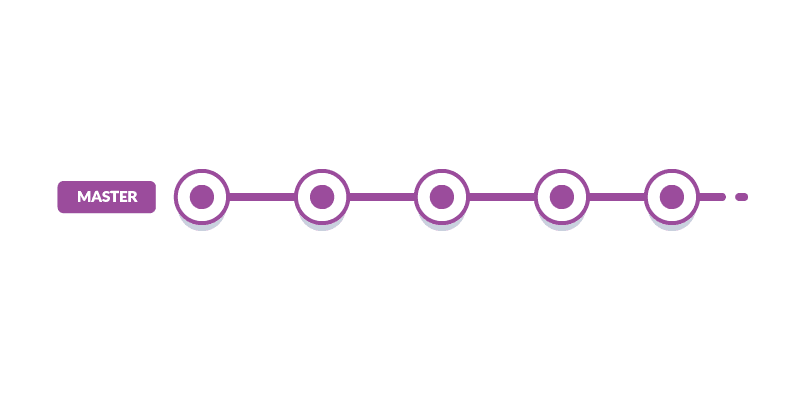
Documentación WORKFLOW DE GIT

* Creación Git Hub https://github.com/robertfox11/JSShoppingCart.git
* Hacemos commits de la estructura de la página principal.
* Probabilidad de que ocurra 80%
* Impacto en el proyecto 60%
* Posible alternativa (mitigación) Pedir ayuda a compañeros
* Probabilidad de que ocurra 30%
* Impacto en el proyecto 60%
* Posible alternativa (mitigación) Pedir ayuda a compañeros
* No encontrar con facilidad la información relacionada con el proyecto
* Probabilidad de que ocurra 30%
* Impacto en el proyecto 60%
* Posible alternativa (mitigación)
* Pedir ayuda a compañeros

A partir de la realización de la estructura se continuó trabajando solamente en la

rama “master”, a través del Workflow “Gitflow”.

Mas información --> <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow>



Herramiento del proyecto

Se utilizaron diferentes herramientas en el desarrollo del proyecto. Son los siguientes:

1. ***git: un potente sistema de control de*** versiones que ayuda a realizar un seguimiento de los cambios en el árbol de trabajo.
2. ***Visual Studio Code: un editor de*** código optimizado para crear y depurar aplicaciones web modernas.
3. ***Herramientas para desarrolladores de Google Chrome:*** se utiliza para depurar el código JavaScript y para probar los ajustes de diseño.
4. ***Documentos de Google:*** se utiliza para escribir la documentación del proyecto.
5. [***Validador W3C***](https://validator.w3.org/)***:*** utilizado para validar el código HTML y CSS.
6. [***ESLint***](https://eslint.org/demo)***:*** utilizado para validar el código JavaScript.

## Flujo de trabajo de Git

Todas las confirmaciones se van a insertar en la rama ***maestra,*** siguiendo un criterio personal de cargar solo instantáneas que son funcionales y que funcionan correctamente, sin contar errores menores. No hay otras ramas, ya que ralentizaría el proceso de desarrollo.

* En otro lado, los mensajes de confirmación finalizan con su objetivo principal indicado entre corchetes: por ejemplo.
* Las relaciones principal para la logica funcional es [class]
* las confirmaciones relacionadas principalmente con los cambios CSS comienzan con ***[styling]***;
* los relacionados principalmente con el diseño de la página, ***[layout]***; los
* Los relacionado con el proyecto añadiendo carpetas o documentacion, plugins, library ***[proyect]****;*
* relacionados principalmente con la documentación, ***[documentation]***.

## Estructura del archivo

Los archivos del proyecto se organizarán de esta manera:

music-library/

.gitignore Carpeta utilizada por *git* para contener información sobre el repositorio.

assets/

img/ Carpeta que contiene todas las imágenes utilizadas en la interfaz.

js/ Carpeta que contiene todos los scripts utilizados en la interfaz de usuario.

css/ Carpeta que contiene todos los estilos utilizados en la interfaz de usuario.

index.html Página principal de la aplicación web.

README.mdArchivo que contiene instrucciones sobre cómo ejecutar el proyecto.

Registro de lecciones aprendidas.

* Realización de una mejora de la documentación más detallada