Music Library

V 0.01 “MVP” | Rafael Venegas D | Assembler School

# Objetivo

Utilizar la API de iTunes Search API para realizar búsquedas y mostrar resultados, utilizando Jquery, AJAX, HTML y CSS.

**Tabla de organización del proyecto**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Prioridad | Dificultad | Tiempo estimado |
| Documentación | Alta | Media | 4 horas |
| Organización | Alta | Baja | 1 hora |
| Reuniones Diarias | Media | Baja | 5 horas (1 hora/día) |
| Creación del repositorio | Baja | Baja | 10 min |
| Análisis de los requisitos del cliente | Alta | Baja | 1 hora |
| Búsqueda previa de información | Media | Media | 2 horas |
| Pruebas con la API | Media | Alta | 2 horas |
| Creación del boceto | Media | Media | 1 hora |
| Creación de clases, objetos y funciones | Alta | Alta | 4 horas |
| Desarrollo de la parte lógica (jquery y AJAX) | Media | Alta | 8 horas |
| Pruebas con localStorage | Baja | Media | 2 horas |
| Desarrollo de la vista (HTML y CSS) | Baja | Media | 3 horas |
| Testing / Correcciones | Baja | Alta | 4 horas |

**Registro de incidencias que se han detectado durante la ejecución del proyecto.**

* La conexión con la API de países sugerida en el briefing del proyecto presentó errores, por lo que se optó por cambiar de API.
* La instalación del validador de código Javascript tomó mas tiempo de lo esperado, debido a que se tuvo que configurar NPM.
* El proyecto tuvo un retraso de dos días en la entrega, según el tiempo estimado inicialmente, debido a asuntos personales del desarrollador.

**Registro de lecciones aprendidas.**

En este proyecto se han aprendido varias cosas, las cuales se detallan a continuación:

* Simplificar la forma de agregar y remover de favoritos
* Simplificar la forma de ejecutar la función “.append()”, para minimizar código

**Calendario del proyecto.**

**Cronograma del proyecto.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Music Library | | | | | |  |
| Miércoles | Jueves | Viernes | Weekend | Lunes | Martes | Miércoles |
| Organización |  |  |  |  |  |  |  |
| Análisis de los requisitos del cliente |  |  |  |  |  |  |  |
| Creación del repositorio |  |  |  |  |  |  |  |
| Búsqueda de información |  |  |  |  |  |  |  |
| Creación de la documentación |  |  |  |  |  |  |  |
| Creación de un boceto del HTML |  |  |  |  |  |  |  |
| Creación de clases, objetos y funciones |  |  |  |  |  |  |  |
| Pruebas con la API |  |  |  |  |  |  |  |
| Desarrollo de la parte lógica (jquery y AJAX) |  |  |  |  |  |  |  |
| Pruebas con LocalStorage |  |  |  |  |  |  |  |
| Desarrollo de la vista (HTML y CSS) |  |  |  |  |  |  |  |
| Testing/Correcciones |  |  |  |  |  |  |  |

**Estructura de los archivos**

Las carpetas se organizaron de la siguiente forma:

music-library/

* assets/
  + images/ Carpeta que contiene todas las imágenes usadas.
* js/ Carpeta que contiene todos los scripts usados.
  + css/ Carpeta que contiene todos los estilos usados.
  + documentation/ Carpeta que contiene la documentación
* node\_modules Carpeta que contiene las dependencias utilizadas.
* . eslintrc.js Archivo que contiene la configuración del ESLint
* . gitignore Archivo con la configuración de los elementos a ignorar por GIT
* index.html Página principal del proyecto.
* package-lock.json Archivo con descripción de las dependencias disponibles para instalar
* package.json Archivo con descripción de las dependencias instaladas
* README.md Archivo con las instrucciones de cómo correr el proyecto.

**Mediciones de control de calidad.**

A través de un checklist, se realizó una verificación de cada uno de los requisitos del cliente y si se cumplían.

Para validar el código Javascript se utilizó ESLint, instalado a través de NPM.

**Métricas de calidad.**

Se verificó individualmente la calidad a través de las siguientes métricas:

- Compatibilidad con los diferentes exploradores

- Funcionamiento correcto del javascript (a través de ESLint)

- Obtención de los datos correctamente

- Guardado de la información en localStorage

**Documentación de requisitos.**

- Instalación de NPM.

- Sistema operativo PC: Windows, Mac o Linux. Movil: MAC, IOS.

- En navegadores compatibles (versiones más recientes): Opera, Google Chrome, Firefox, Internet Explorer.

**Documentación de riesgos en el caso de que estos existan**

En una etapa inicial se plantearon los siguientes riesgos posibles:

**Documentación acerca del WORKFLOW de git que vas a usar**

- En una etapa inicial, se realizaron pruebas en una rama “pruebas”, en la cual se hicieron commits de la estructura de la página principal.

- A partir de la realización de la estructura se continuó trabajando solamente en la rama “master”, a través del workflow “Gitflow”.

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

**Documentación acerca de las herramientas usadas en el proyecto**

LocalStorage 🡪 <https://developer.mozilla.org/es/docs/Web/API/Window/localStorage>

ESLint -> <https://eslint.org/docs/user-guide/getting-started>

API itunes -> <https://affiliate.itunes.apple.com/resources/documentation/itunes-store-web-service-search-api/>

**Arquitectura del proyecto**

**Clases**