[[1]](#footnote-1)

INTRODUCCIÓN A JAVASCRIPT

Cardenas Sillo Alex Sebastián, Castillo Suqui Steven Jahir, Gómez Gualotuña Lenin Cristóbal, Quishpe Cuadrado Shamyr Sebastian, Vela Totoy David Alejandro, Luna Cajas Stefany Scarlett

ESFOT, Escuela Politécnica Nacional

Quito – Ecuador

*Resumen –* En este documento de presenta el trabajo en equipo de como trabajar con una API la cual es basada en Marvel y su uso en una página web, además se muestra paso a paso de como se configuraron los documentos con extensiones html, css y js, cuenta con figuras que evidencian cada uno de los pasos, por otra parte en el documento también se menciona cual será la plataforma a usarse para publicar la página la misma que será Netlify, por ultimo se registra el repositorio de GitHub donde se muestra el trabajo completo.

# Propósito de la práctica

Integrar y trabajar con una API externa en el desarrollo web.

# Objetivo General

Desarrollar una aplicación web junto con el consumo de una API.

# Objetivos Específicos

* Buscar una API externa.
* Aplicar plantillas que agilicen la creación de la página.
* Desplegar la aplicación ya sea en una IaaS, PaaS, SaaS.
* Realizar pruebas de rendimiento de la aplicación.
* Probar la funcionalidad de la aplicación.

# Desarrollo y resultados obtenidos

La API que el grupo tomo para agregarle a nuestra aplicación fue de Marvel la cual nos permite acceder a información de diferentes superhéroes además de sus comics, eventos e imágenes y es la siguiente:

<https://developer.marvel.com/documentation/apiresults>

Tomamos como referencia la siguiente plantilla para el desarrollo de la aplicación web <https://codingartistweb.com/2023/04/marvel-api-app-with-javascript/> , sin embargo, se realizaron cambios debido a errores que se presentaron durante la conexión con la API y la creación de la lista predictiva de personajes. Se agrego otra

funcionalidad que permite además revisar algunos de los títulos de los comics donde hace aparición el personaje buscado.

Basándonos en el código HTML procedemos a añadir además Bootstrap para facilitar los diseños de la página y modificamos los contenedores con la idea de que pueda ser responsivo además de las conexiones con los dos scripts realizados en JavaScript como se muestra en la Figura IV.1.

Texto

Descripción generada automáticamente

Figura IV.1. Codigo HTML

A continuación, procedimos a usar de nuestra plantilla el archivo de estilos en CSS, en base a el archivo cambiamos los colores y agregamos los espaciados necesarios entre los contenedores para que se vea más ordenado.

Necesitaremos generar un HashValue, el cual obtendremos al tener el timestamp, la privateKey y la publicKey. La privateKey y la publicKey la obtenderemos al logearnos en la página de la API que usaremos como se muestra en la Figura IV.2. Mientras que el timestamp lo obtendremos con linea de código de JavaScript como se muestra en la Figura IV.3y esta imprimirá por consola nuestro timestamp como se muestra en la Figura IV.4.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Figura IV.2. Key Values de la API

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figura IV.3. Líneas de código para obtener el TimeStamp

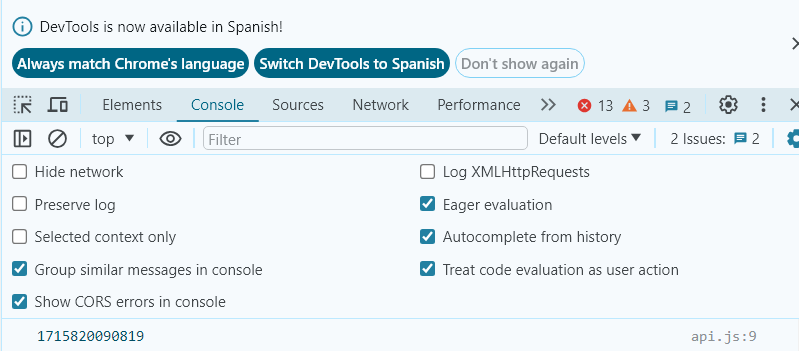


Figura IV.4. Obtención del TimeStamp por consola

Para poder generar el HashValue combinaremos el timestamp + publicKey + privateKey en la siguiente página web <https://www.md5hashgenerator.com/> , allí lo obtendremos como se muestra en la Figura IV.5y lo copiaremos el MD5 Hash.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura IV.5. Generación del HashValue

Creamos dos archivos JavaScript el primero se centra en las conexiones que realizaremos para poder usar nuestra API de Marvel y haremos el uso del timestamp, la publicKey y el HashValue que generamos, además en este archivo destructuramos un arreglo como se muestra en la Figura IV.6, para almacenar variables que serán utilizadas al momento de llamar a nuestra API.

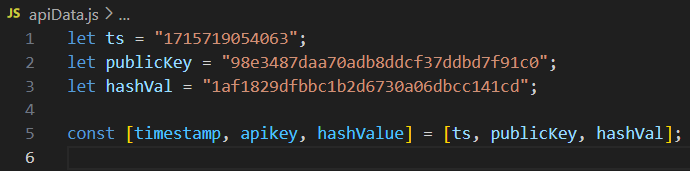


Figura IV.6. Primer script JS

* Segundo Script

Luego tenemos nuestro segundo script llamado api.js en el cual haremos el consumo de la API.

Comenzamos manipulando el DOM, creando cada uno de los elementos que mostrarán los datos, esto podemos observarlo en la Figura IV.7 Inicialización de elementos del DOM.

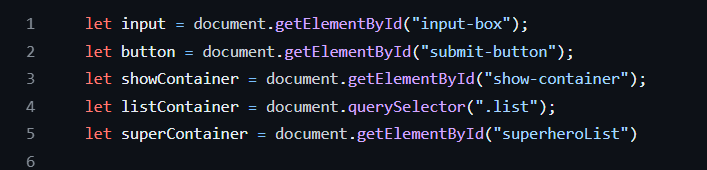


Figura IV.7 Inicialización de elementos del DOM.

Ya con los elementos definidos, necesitaremos 2 funciones la que observamos en la Figura IV.8 Función removeElements será usada para simular la renderización que en realidad limpiará los elementos hijos de un elemento padre.

A screen shot of a computer

Description automatically generated

Figura IV.8 Función removeElements

La Figura IV.9 Evento Keyup en el input tiene un listener keyup que nos permitirá saber si el número de letras al escribir en el input es menor a 4, con esta condición retorna false.

A computer code on a black background

Description automatically generated

Figura IV.9 Evento Keyup en el input

Para consumir la api, debemos tener en claro el uso de fetch() que es una promesa para manejar la asincronía, en este caso le pasamos como argumento la url con el formato necesario para consumir la API, decodificamos a un formato json lo cual nos permitirá trabajar con un objeto de este tipo, el uso de fetch lo podemos ver en la Figura IV.10 Función asíncrona fetch

A computer screen with white text

Description automatically generated

Figura IV.10 Función asíncrona fetch

Ya con los datos del json podemos manipularlos, una forma muy común es recorrer en el objeto json para acceder a cada valor, vemos en la Figura IV.11 Recorrido del ojeto Json usamos un método forEach que nos cree un campo visual donde se mostrarán las coincidencias según lo ingresado en el input, se mostrará luego de tipear 4 letras que coincida con la información del json.

A screen shot of a computer code

Description automatically generated

Figura IV.11 Recorrido del ojeto Json

Posteriormente, con el botón que se definió al inicio, le añadimos un evento click este nos servirá para realizar la una petición a la API y mostrar los datos en pantalla según el input ingresado y válidado, además se hace una nueva petición a otra url para obtener la información de los comics en los que aparece cada personaje.

A screenshot of a computer program

Description automatically generated

Figura IV.12 Botón que imprime la información de la API

Al final contamos con la función Windows.onload, esto nos permite mostrar de forma correcta la información que se va actualizando, pero lo hace una vez se haya cargado por completo la web. Esto lo observamos en la |

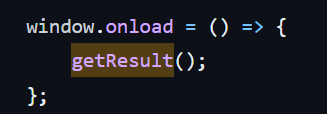


Figura IV.13 Evento window.onload

* Aplicación de un modelo de servicio en la nube

Netlify

Se selecciono esta plataforma ya que es de tipo PaaS (Platform as a Service), ya que esta plataforma permite alojar aplicaciones web de forma sencilla, brindando así automatización a la misma.

Beneficios

* Netlify permite que el proceso de despliegue sea menos tedioso y mas ágil, ya que solo se debe contar con un repositorio Git donde se encuentre la información.
* Cuenta con un despliegue continuo (actualizaciones) esto beneficia a los usuarios ya que se visualización los cambios de forma rápida y frecuente.
* Maneja una gestión de infraestructura, esto permite automatizar tareas.

Para publicar la pagina web en Netlify es necesario contar con un repositorio en GitHub como se muestra en la Figura IV.14. GitHub

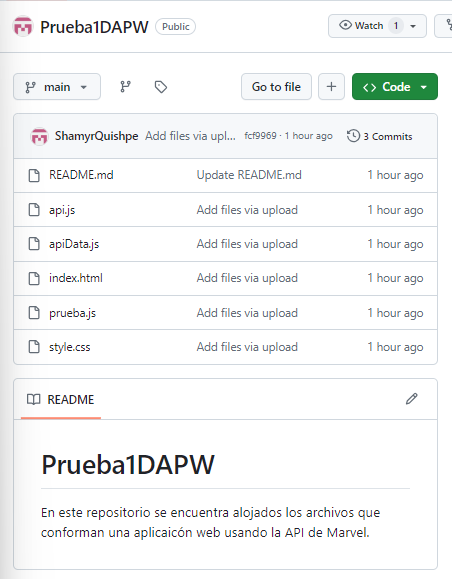


Figura IV.14. GutHub

Se debe registrar en Netlify esto se lo puede realizar con el GitHub, ya que con esto se permitirá que la plataforma automatice los cambios y demás acciones como se muestra en la Figura IV.15. Automatización

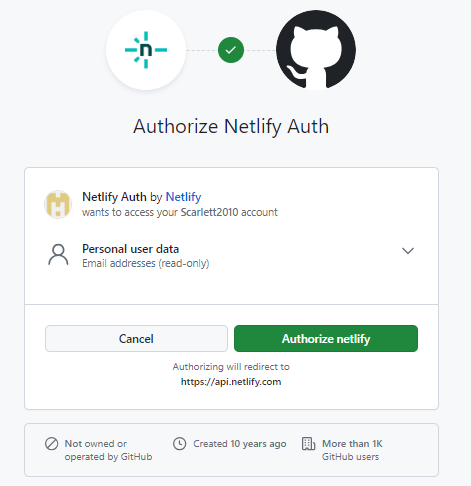


Figura IV.15. Automatización

Después de terminar de registrarnos nos aparecerá una pagina principal donde podremos subir el proyecto como se muestra en la Figura IV.16. Subir archivo, además de seleccionar la fuente de donde se quiere subir el archivo, que en este caso será desde GitHub y se podrán visualizar los repositorios y seleccionar el deseado

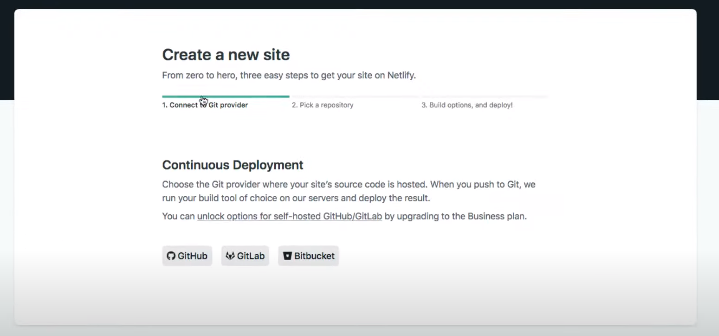


Figura IV.16. Subir archivo

Se procederá a llenar otros campos requeridos como se muestra en la Figura IV.17. Datos, luego de ello se observarán las fases de despliegue como se muestra en la Figura IV.18. Fase despliegue

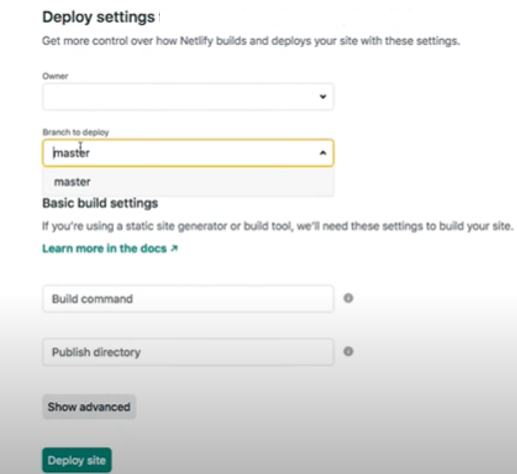


Figura IV.17. Datos

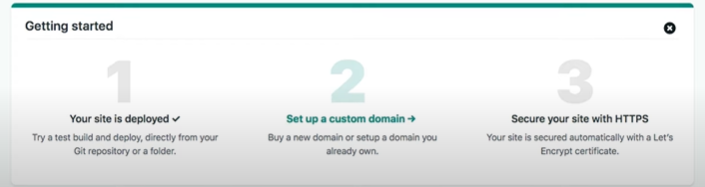


Figura IV.18. Fase despliegue

Después ya se verá la pagina desplegada como se muestra en la Figura IV.19. Página

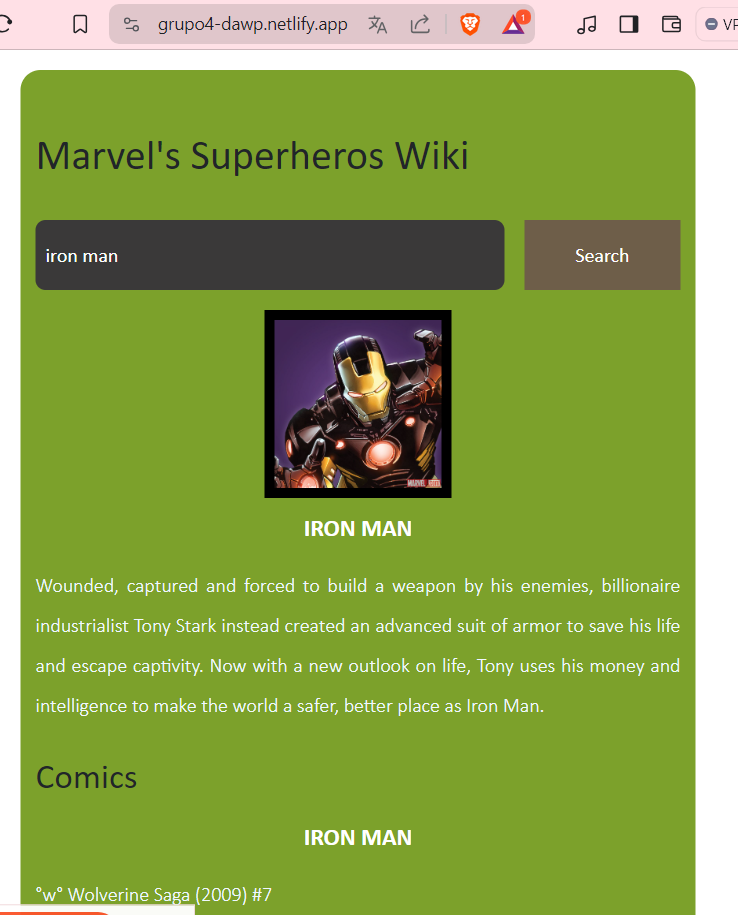


Figura IV.19. Página

# Conclusiones

* Se concluye que el uso de una API es beneficioso para al momento de desarrollar una pagina web ya que permite agregar funcionalidades
* En fin, el uso de plantillas ayuda en el desarrollo de la página, al basarse en estilos ya definidos.
* En síntesis, la realización de pruebas de rendimiento ayuda a tener una visión más amplia sobre los requerimientos a mejorar para generar una buena experiencia de usuario.

# Repositorio

A continuación, se adjuntan el link del repositorio en el cual esta alojado los archivos que conforman la aplicación web <https://github.com/ShamyrQuishpe/Prueba1DAPW>

# Biografías

A person taking a selfie

Description automatically generated**Alex Cárdenas**, nació en Quito-Ecuador el 15 de julio de 2003. Realizó sus estudios secundarios en la Institución Educativa Fiscal “Cinco de Junio “, llegando a graduarse en 2021. Actualmente realiza sus estudios en la Escuela Politécnica Nacional en la especialidad de Tecnología Superior en Desarrollo de Software en el nivel de cuarto semestre.

([alex.cardenas@epn.edu.ec](mailto:alex.cardenas@epn.edu.ec))

A person wearing a hat and holding up his hand

Description automatically generated**Steven Castillo**, nació en El Oro-Ecuador el 14 de julio de 2003. Realizó sus estudios secundarios en la Unidad Educativa Gran Colombia llegando a graduarse en 2021. Actualmente realiza sus estudios en la Escuela Politécnica Nacional en la especialidad de Tecnología Superior en Desarrollo de Software en el nivel de cuarto semestre.

([steven.castillo@epn.edu.ec](mailto:steven.castillo@epn.edu.ec))

A person in a suit

Description automatically generated

**Lenin Gómez**, nació en Quito-Ecuador el 20 de abril de 2003. Realizó sus estudios secundarios en la Unidad Educativa Municipal “Sucre” llegando a graduarse en 2021. Actualmente realiza sus estudios en la Escuela Politécnica Nacional en la especialidad de Tecnología Superior en Desarrollo de Software en el nivel de cuarto semestre.

([lenin.gomez@epn.edu.ec](mailto:lenin.gomez@epn.edu.ec))

A person in a suit and tie

Description automatically generated**Shamyr Quishpe**, nació en Quito-Ecuador el 26 de enero de 2003. Realizó sus estudios secundarios en Unidad Educativa “Luciano Andrade Marín” llegando a graduarse en el 2021. Actualmente realiza sus estudios en la Escuela Politécnica Nacional en la especialidad de Tecnología Superior en Desarrollo de Software consiguiendo beca por excelencia académica tres semestres consecutivos.

([shamyr.quishpe@epn.edu.ec](mailto:shamyr.quishpe@epn.edu.ec))

A person wearing glasses and smiling

Description automatically generated

**David Vela**, nació en Quito-Ecuador el 07 de diciembre del 2002. Realizó sus estudios secundarios en Unidad Educativa “Alberto Enríquez” llegando a graduarse en el 2020. Actualmente realiza sus estudios en la Escuela Politécnica Nacional en la especialidad de Tecnología Superior en Desarrollo de Software en el nivel de cuarto semestre.

([david.vela@epn.edu.ec](mailto:david.vela@epn.edu.ec))

Un joven sonriendo con una camisa azul

Descripción generada automáticamente**Scarlett Luna, n**ació en Quito-Ecuador el 28 de febrero de 2004. Realizó sus estudios secundarios en el Instituto Nacional Mejía llegando a graduarse el 2021. Actualmente realiza sus estudios superiores en la Escuela Politécnica Nacional en la carrera de Tecnología superior en Desarrollo de software.

([scarlett.luna@epn.edu.ec](mailto:scarlett.luna@epn.edu.ec))

1. [↑](#footnote-ref-1)