



# Trabajo Practico #2

Vue.js

El presente documento tiene como finalidad servir como guía para la confección del segundo Trabajo Practico. El trabajo practico se centra en la utilización de Vue.js usando un archivo de texto html, y declarado usando el CDN.

El desarrollo, conclusión y entrega del mismo sera en un repositorio git, el mismo que el utilizado en el trabajo practico nro. 1.

En cada punto de esta guía se indicara si corresponde realizar un commit y push al repositorio git con el comentario apropiado que refleje el motivo de los cambios.

## Requerimientos

Para la realización del practico se necesitan:

- A. El repositorio git utilizado en el TP #1.
- B. Tener instalado el cliente Git en nuestro sistema operativo.
- C. Un editor de texto.

## Instrucciones

- Completa todos los ejercicios dentro de este mismo archivo HTML que el practico anterior.
- Utiliza Vue.js a través del CDN.
- No necesitas instalar nada localmente, solo abre el archivo en un navegador.
- Asegúrate de que todo el código Vue.js esté dentro de las etiquetas `<script>` en la sección `<body>` de tu HTML.

## Paso #1 - Lista de eventos

Crear una nueva lista (listaeventos, similar a la lista de tareas creada anteriormente) que muestre el contenido de un arreglo de objetos. Esos objetos deberán tener un ID, un nombre y un Timestamp.

La lista debe mostrar en nombre y el timestamp con milisegundos incluidos.

La lista debe estar definida como un conjunto vacío. Luego, debe ser completada a medida que se ejecutan todos los eventos de nuestra app Vue.

Por ejemplo, en la ejecución del evento **beforeUpdate** debe agregarse un elemento al arreglo de eventos, con el formato: `{id: 1, nombre: 'beforeUpdate', timestamp: Date.Now()}`, y estos deben aparecer en la lista. Lo mismo con los eventos `beforeCreate`, `created`, `beforeMount`, `mounted`, `beforeUpdate`, `updated`, `beforeDestroy` y `destroyed`.

Hacer un commit y un push de los cambios. Como mensaje del commit utilizar la descripción del paso actual, en este caso "Lista de eventos". Esto aplica, con el comentario adecuado, a todos los commits que se hagan en el resto del practico.

## Paso #2 - Implementación de Axios

Se debe incluir la librería de Axios mediante cdn.

Se deben crear 2 funciones:

`obtenerPost`: debe recibir como parametro un numero de post (`postId`) y utilizando una instancia de axios debe llamar a la API publica <https://jsonplaceholder.typicode.com/posts/{postId}> para finalmente devolver el objeto obtenido.

`obtenerComments`: debe recibir como parametros el numero de post (`postId`) y utilizando una instancia de axios debe llamar a la API publica <https://jsonplaceholder.typicode.com/posts/{postId}/comments> para obtener un arreglo de todos los comentarios del post recibido como parámetro.

Se debe realizar un commit y un push de los cambios al repositorio.

## Paso #3 - Uso de las funciones

Se debe crear una nueva lista vacía de post que sirva para mostrar los 10 primeros post de la API pública. Además, debemos crear una nueva lista, también vacía, que contenga un arreglo de los comentarios que recibió un determinado post.

Una vez que la instancia de Vue se haya cargado, debemos crear un bucle for que se ejecute 10 veces y que obtenga mediante la función `obtenerPost` el objeto de la API y lo agregue a la lista anteriormente creada de post.

Por ultimo, debemos mostrar en pantalla esos post y cuando hacemos click arriba de alguno de ellos debemos invocar a la función

obtenerComments con el id del post clickeado y obtener los comentarios.  
Esos comentarios debemos mostrarlos en pantalla.

### **Resultado Esperado:**

Una vez cargada la pagina, que aparezca una lista de los post del 1 al 10 en pantalla y que al hacerle click a cualquiera de ellos, aparezca, al costado (o abajo) los comentarios de ese post.

Hacer un commit y un push de los cambios realizados.