# Taller 5

Lukas Mateo Bettarel Calderon, 201923480

1. Información general del proyecto: para qué sirve, cuál es la estructura general del diseño, qué grandes retos de diseño enfrenta (i.e. ¿qué es lo difícil?). Deben incluir la URL para consultar el proyecto.

El proyecto escogido se llama Puppeteer y su repositorio se encuentra en el siguiente link: <https://github.com/puppeteer/puppeteer>. Puppeteer es un proyecto escrito en JavaScript que se puede descargar por npm o yarn, y básicamente es un navegador de Chrome altamente controlable por medio de su API de alto nivel. Permite cargar sitios web, acceder a todos sus archivos de HTML, css, y JavaScript, interactuar con ellos, entre otras cosas. Se utiliza mas que todo para automatizar interacciones con sitios web, generar fotos o pdfs de páginas, probar extensiones de Chrome, diagnosticar problemas de un sitio, o cualquier otra cosa que puede hacer un humano en cuanto a interactuar con un sitio web. En cuanto a su estructura de diseño, como puede prácticamente simular al navegador Chrome, se parte en distintas clases dependiendo de su funcionamiento. Por ejemplo, existen las clases Keyboard y Mouse, que permiten simular un teclado y un ratón dentro del navegador simulado, respectivamente. Además de la dificultad de manejar una gran cantidad de diferentes funcionalidades, existe el reto de literalmente simular una instancia del navegador Chrome y todas sus funcionalidades. Esto también implica trabajar con temas asincrónicos debido a que por lo general se ven involucrados estos temas en aplicaciones web, y el tema de correctamente aplicar estilos, y el correcto manejo del DOM de los sitios web, entre otras cosas.

1. Información y estructura del fragmento del proyecto donde aparece el patrón. No se limite únicamente a los elementos que hacen parte del patrón: para que tenga sentido su uso, probablemente va a tener que incluir elementos cercanos que sirvan para contextualizarlo.

El patrón escogido aparece en distintos lugares dentro del proyecto, visto a que existen varios casos donde ocurren llamadas que son asincrónicas y se pueden demorar un rato en dar una respuesta. Sin embargo, la implementación especifica encontrada es dentro del archivo ubicado en [website/src/theme/SearchBar/index.js](https://github.com/puppeteer/puppeteer/blob/590d163da812613abde701a657026cbf563e3ea5/website/src/theme/SearchBar/index.js#L14). En este archivo existe una variable llamada onInput que almacena una función que tiene un parámetro llamado evento, y que no tiene valor de retorno. Este callback es utilizado por una función llamada useDocSearchKeyboardEvents de otro paquete de npm/yarn que genera una caja de búsqueda. Esta función toma un callback para cuando se recibe algún input del usuario, y al momento que esto ocurre useDocSearchKeyboardEvents llama al callback que recibió como parámetro, y le pasa el objeto de evento como parámetro.

1. Información general sobre el patrón: qué patrón es y para qué se usa usualmente.

El patrón de callback es una manera de pasar una función como parámetro a otra función, permitiendo almacenarla y invocarla en otro momento. Es útil en casos donde la función se debe llamar en caso de que ocurra algún evento, o luego de completar algún proceso asincrónico.

1. Información del patrón aplicado al proyecto: explicar cómo se está utilizando el patrón dentro del proyecto.

(respondido en #2)

1. ¿Por qué tiene sentido haber utilizado el patrón en ese punto del proyecto? ¿Qué ventajas tiene?

Tiene sentido utilizar el patrón de callback en este caso porque la acción que debe cumplir el programa depende de un evento externo. Al momento que ocurre este evento se llama la función y se realiza la acción en el momento adecuado.

1. ¿Qué desventajas tiene haber utilizado el patrón en ese punto del proyecto?

Aunque los callbacks en algunos casos son la única manera de resolver un problema, como en este caso, por lo general se ven limitadas a la información a la que tienen acceso, ya que se almacenan y se invocan en un momento posterior. En algunos casos asincrónicos la información que se les pasa ya no existe, algo que puede llevar a errores. Sin embargo, esto es producto de una mala implementación y errores como este pueden ser evitados, pero igual cabe tener precaución extra al utilizar callbacks.

1. ¿De qué otras formas se le ocurre que se podrían haber solucionado, en este caso particular, los problemas que resuelve el patrón?

Si esta hubiese sido una aplicación asincrónica, digamos un callback en caso de que una operación haya sido exitosa, se pudo haber utilizado un async/await o un Promise en JavaScript. Sin embargo, en casos como este donde es necesario llamar una función luego de algún evento, como por ejemplo se oprimió un botón, o se oprimió una tecla en específico, la única manera de que exista esta funcionalidad es por medio del uso de callbacks. Por esta razón los callbacks son tan útiles, y necesarios en los lenguajes de programación. Tal vez envés de definir una variable para almacenar el callback se pudo haber utilizado una función anónima, aunque esto sigue siendo considerado como un callback, y puede causar aún mas confusión.