Framework e Interoperabilidad 2018
Trabajo Práctico N° 1

Cliente y Servidor RESTful

Grupo:

Farias Karen Maive
FAEA-1372

Monserrat Vidal María
FAI-1829

Fecha:

03/09/2018

1 Introducción:

El siguiente informe tiene como objetivo dar a conocer la información y conocimientos obtenidos respecto a los servicios REST y la librería REACT.
El principal objetivo de este informe es que sirva como pilar inicial e incentivo para dar a conocer estas herramientas y las posibles ventajas y desventajas que se pueden ir encontrando durante el desarrollo de una API. Además, este informe será acompañado por un ejemplo práctico desarrollado por el equipo de trabajo y con el cual esperamos poder reflejar los puntos que se tratarán en este informe.
Este equipo tuvo como primer objetivo, realizar un servicio sencillo usando REST y utilizar REACT para desarrollar el front end del mismo. Para ello se recolectó información de varios sitios web (disponibles en el apartado de bibliografía) y se comenzó a trabajar con un dominio aplicado a una Editorial. Dicha editorial podría acceder a través de la aplicación web a su registro personal el cual muestra todos los libros que la misma ha sacado a la venta. Así mismo puede crear un nuevo ítem Libro, modificar los items existentes y eliminar los libros que ya no estén asociados a la editorial.
Teniendo en cuenta nuestras experiencias prácticas, trataremos de hacer énfasis en los problemas y ventajas que descubrimos al desarrollar el ejemplo Editorial.

1.1 ¿Qué es un servicio REST?

REST cuyas siglas en inglés significan "REpresentational State Transfer" y es un estilo de arquitectura software.

1.2 ¿Qué es RESTFUL?

Restful hace referencia a un servicio web creado bajo los protocolos y normas de REST, es decir que es un servicio realizado exclusivamente con REST.

1.3 Comunicación Cliente Servidor.

En la arquitectura REST el cliente envía solicitudes para ver o modificar recursos y el servidor responde dichas solicitudes. Generalmente se usan GET, POST, PUT y DELETE.

2 ¿Qué es REACT?

REACT, como bien ellos se definen en su sitio oficial, es una librería de JavaScript para la construcción de interfaces de usuario.
Su diseño, al igual que otras bibliotecas como Angular, usa componentes que mostraran en pantalla diferentes segmentos de un sitio web basándose en las necesidad de información del usuario.
Dependiendo de la versión utilizada, tendremos algunas ventajas y desventajas en la implementación de componentes y es necesario que informemos algunas de las principales diferencias entre las versiones más antiguas y la última versión.

2.1 ¿Cómo funciona React?

React está construido en torno a hacer funciones, que toman las actualizaciones de estado de la página y que se traduzcan en una representación virtual de la página resultante, también conocido como DOM virtual.

2.2 Componentes.

React permite crear componentes, estos son como funciones de JavaScript. Son elementos independientes y reutilizables.
Todos los componentes de React tienen funciones, las cuales especifican cómo deben comportarse y visualizarse.

Cuando se invoca un componente, se espera un método de renderizado, el cual desencadena un ciclo de vida.

Las funciones más comunes implementadas en el ciclo de vida de un componente son:

- *componentWillMount()* Se lanza antes de que se renderice el componente
- *componentDidMount()* Se lanza después del renderizado del componente
- *shouldComponentUpdate()* Devuelve con un valor si el componente debería actualizarse
- *componentWillUnMount()* Se lanza antes de que el componente se elimina.

2.3 ¿REACT versión 15.x o versión 16.x?

Actualmente, React está disponible en la versión 16.4.2 la cual tiene claros beneficios en el mercado del desarrollo web ya que se ha actualizado y permite trabajar con un formato mucho más ordenado basándose en estructuras de encapsulado en arreglos o 'arrays'. Esto permite usar etiquetas de HTML e incluso permiten el uso de parámetros dentro de las etiquetas, otorgando la posibilidad de ahondar en los estilos CSS o librerías asociadas.
En versiones anteriores a la versión 16.x, los retornos de los renderizados deben ser strings y el estilo retocado en el index de la aplicación o utilizando estrategias dentro del mismo index como por ejemplo, asignando los estilos dentro de un div donde se renderiza el contenido.

Para dejar claro a lo que hacemos referencia, dejaremos un ejemplo sencillo de renderizado.

```
//Retorno de una vista usando la función render() v16.x
render() {
  //¡No es necesario envolver los elementos de la lista en un elemento adicional!
  return [
    // No olvidar las key!
    <li key="A">Primer Libro</li>,
    <li key="B">Segundo Libro</li>,
    <li key="C">Tercer Libro</li>,
```

```
  ];
}


//Retorno de una vista usando la función render() v15.x
render() {
  return 'Mírame. Sin elemento parental a mi alrededor';
}
```

Otro claro beneficio de la versión 16.x de REACT es el manejo de errores. En versiones anteriores, un error en cualquier componente provocaba el fallo de toda la aplicación web siendo un caos encontrar el error específico y más en aplicaciones de un tamaño considerablemente grande o que cuentan con varios componentes. Actualmente, en la versión 16.x, el error producido por un componente no afecta el renderizado o acciones de los demás componentes permitiendo encontrar rápidamente el error e incluso mantener funcional la aplicación web mientras se resuelven los errores del componente afectado.

Muchas de las nuevas funciones que REACT v16.x trae, están exclusivamente orientadas al desarrollo ordenado de una aplicación web, por ejemplo, proporcionan una forma de primera clase para representar a los children en un nodo DOM que existe fuera de la jerarquía DOM del componente principal. De esta forma podemos olvidarnos de las estrategias de diseño que debíamos implementar en versiones anteriores.

¡¡Y todo esta nueva versión pesa muchísimo menos que su versión 15.x!!

2.4 ¿Cuál es la desventaja de usar la versión 16.x de REACT?

Considerando que la versión 16.0.0 de REACT fue lanzada el en Septiembre de 2017, la bibliografía y aportes sobre la nueva versión son muy escasos y la mayoría de los usuarios de REACT deciden usar bibliotecas extras a REACT como REDUX o EXPRESS para trabajar las interfaces, lo cual termina siendo más engorroso y tedioso que trabajar toda la aplicación con REACT.
Además, el primero de Agosto de 2018, a once meses del lanzamiento de la versión 16.0.0, se lanza en la versión 16.4.2 la cual resuelve un problema de vulnerabilidad en aplicaciones que utilizan el patrón ReactDOMServer. Este patrón fue lanzado en la versión 16.0.0 y fue heredada por todas sus posteriores versiones pero recién en la versión 16.4.2 se ha solucionado este problema de vulnerabilidad.
Si bien esta vulnerabilidad sólo puede afectar algunas aplicaciones React creadas por el servidor y las aplicaciones puramente renderizadas por el cliente no se ven afectadas, el mal sabor para los usuarios de las versiones 16.x ha provocado un revuelo en varios foros ya que en una de las declaraciones se advierte:

“Mientras investigamos esta vulnerabilidad, encontramos vulnerabilidades similares en algunas otras bibliotecas front-end populares. Hemos coordinado este lanzamiento junto con las versiones de Vue y Preact para solucionar el mismo problema. El número de seguimiento para esta vulnerabilidad es CVE-2018-6341.”

Como conclusión a lo expuesto hasta el momento, recomendamos que se sea muy consciente de los posibles errores, fallos o vulnerabilidades de las últimas versiones y se ponga especial atención al manejo de información delicada en las aplicaciones que utilicen las versiones 16.x de REACT.

2.5 ¿Es compatible REACT con un servicio REST?

Podemos trabajar perfectamente estas herramientas juntas ya que trabajaremos REST para todo lo referido al servicio web y back end mientras que REACT lo usaremos para generar la interfaz de usuario, es decir el front end.

3 Herramientas utilizadas durante el desarrollo del Servicio/Cliente RESTful.

- **PHP**

  Se utilizó lenguaje PHP para crear el servicio web (back end)

- **ReactJS**

  Las librerías de React fueron utilizadas para crear la interfaz que usará el cliente (front end).

- **Bootstrap 4**

  Este framework se utilizó para facilitar el diseño, hacerlo ergonómico y adaptable a cualquier pantalla.

- **React DevTools**

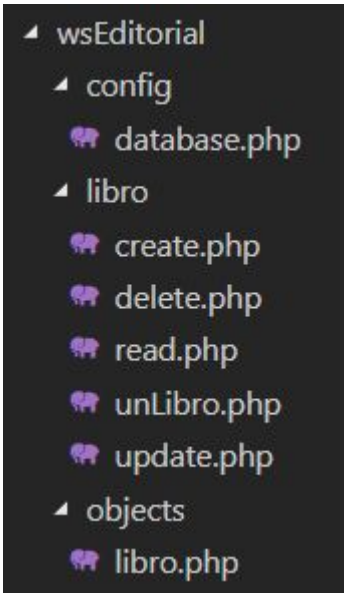
  Es una extensión de desarrollo para navegadores, que permite inspeccionar las jerarquías de los componentes y subcomponentes de la aplicación desarrollada con React.
  Podemos ver y modificar las propiedades, estados, datos, etc.
  Es muy útil para entender cómo funciona React.

- **Postman**

  Es una herramienta para generar peticiones a APIs y crear colecciones de peticiones que nos permitan probarlas de una manera rápida y sencilla.
  Se puede definir el tipo de petición (GET, POST, etc.), tokens de autenticación, cabeceras asociadas a la petición, etc.
  Las peticiones se pueden exportar a múltiples, y ser usadas en nuestros archivos.

4 Ejemplo Cliente y Servidor RESTful con React JS

SERVIDOR:



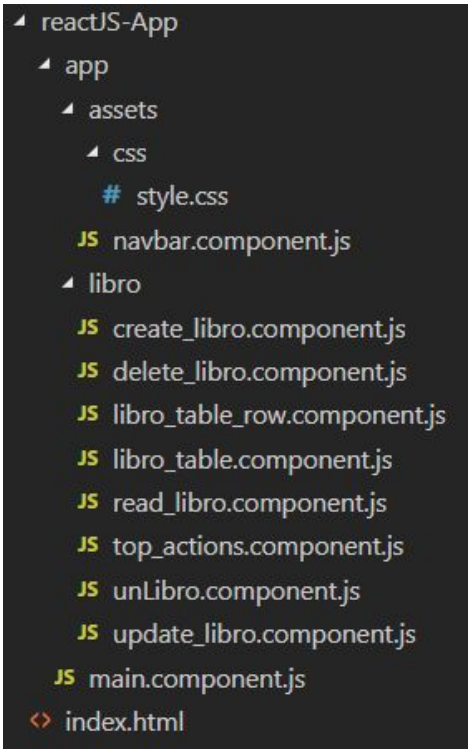
El servidor está programado en lenguaje PHP.
En la carpeta 'config', tenemos el archivo database.php, el cual usaremos para hacer la conexión con la base de datos.

Dentro de la carpeta 'libro', se encuentran los archivos del CRUD. Cada archivo tiene su correspondiente función *header("Access-Control-Allow-Methods: método"); (POST, GET, PUT, DELETE).*
//especifica el método aceptado cuando se accede al recurso en respuesta.
Otras funciones en común que se encuentran son:

- *json_decode(file_get_contents("php://input"));* Se encarga de decodificar los mensajes JSON que envía el cliente.
- *json_encode(...); Codifica los mensajes en formato JSON que son enviados al cliente.*

Por último, en la carpeta 'objects', tenemos el objeto libro, el cual tiene los atributos y funciones necesarias para ser usado por el CRUD. Cada función hace las consultas correspondientes a la base de datos.

CLIENTE:



Cuando la aplicación inicia, se carga el archivo index.html, el cual tiene un con id='content'. Dentro de este tag se renderiza el componente 'main' (main.component.js).
El componente 'main' se inicia con el estado (currentMode) *READ.* Según el estado actual del componente, se renderizan distintos componentes hijos.

Estado READ: (ver todos los libros) se renderiza 'ReadLibroComponent' (read_libro.component.js). Este componente se encarga de solicitar al servidor todos los registros existentes de libros y mostrarlos en forma de tabla. Cada fila de la tabla tiene tres acciones para cambiar de estado: Ver, Editar y Borrar. También se renderiza un botón de Cargar libro.

Estado CREATE: (crear un nuevo libro) se renderiza 'CreateLibroComponent' (create_libro.component.js). Visualiza el formulario para crear un nuevo registro. Cuando se pulsa el botón guardar, se ejecuta la función *onSave()*, que se encarga de enviar los datos en formato JSON al servidor a través de AJAX, por medio de POST. Según la respuesta del servidor, se muestra un mensaje de éxito o de error.

Estado UPDATE: (modificar un libro) se renderiza 'UpdateLibroComponent' (update_libro.component.js). Se encarga, al igual que CREATE, de renderizar un formulario, pero con todos los datos del libro seleccionado. Cuando el componente se monta, la función *componentDidMount()* le solicita al servidor todos los datos del registro, a través del ID. Cada campo del formulario se completa con los datos correspondientes, listos para ser modificados por el usuario.
Cuando se guarda el formulario, se envían los datos al servidor por medio de PUT.

Estado DELETE: (eliminar un libro) se renderiza 'DeleteLibroComponent' (delete_libro.component.js). Cuando se presiona el botón Eliminar, se renderiza un cuadro de confirmación. Si la confirmación es positiva, la función *onDelete()* le envía al servidor, en formato JSON el id del registro, por medio de DELETE.
En caso de que la confirmación sea negativa, el estado del componente 'main' volverá a ser READ.

Estado UNLIBRO: (ver un libro) se renderiza 'UnLibroComponent' (unLibro.component.js). Al montar el componente, la función *componentDidMount()* le pide al servidor todos los datos del registro seleccionado por medio de su ID. Los datos se visualizan de forma ordenada en un cuadro.

5 Conclusión:

A lo largo del desarrollo de este trabajo, comprendimos que es realmente REST y RESTful, así como sus diferencias. Términos que comúnmente escuchamos o leímos, pero sin darles la importancia que se merecen.
Entendimos cómo funcionan las APIs REST locales y externas, y cómo consumirlas con cualquier tipo de cliente.
Aprendimos a usar herramientas y tecnologías muy útiles como Postman, React y React Dev Tools.
Además, mientras buscábamos documentación y ejemplos, vimos muchas librerías y frameworks que despiertan la curiosidad y que deseamos probar, como Express, Redux, entre otras.

BIBLIOGRAFÍA

Página oficial de REACT.js
https://reactjs.org/
(última visita: 30/08/2018)

Postman
https://www.getpostman.com/
(última visita: 02/09/2018)

React Developer Tools
https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi
(última visita: 02/09/2018)

Manual de PHP
http://php.net/manual/es/
(última visita: 01/09/2018)

How To Create A Simple REST API in PHP?
https://www.codeofaninja.com/2017/02/create-simple-rest-api-in-php.html
(última visita: 01/09/2018)

React CRUD Tutorial – JavaScript
https://www.codeofaninja.com/2016/07/react-crud-tutorial.html
(última visita: 31/08/2018)

Cómo consumir una API Rest con React?
https://medium.com/@alejogs4/como-consumir-una-api-rest-con-react-bf9d2665bde8
(última visita: 28/08/2018)

React Tutorial - Learn ReactJS and build a simple CRUD app
https://www.youtube.com/watch?v=S66rHpyU-Eg
(última visita: 28/08/2018)

Creando un CRUD con JavaScript: Construyendo el Frontend usando React
https://openwebinars.net/blog/creando-un-crud-con-javascript-construyendo-el-frontend-usando-react/
(última visita: 28/08/2018)

CRUD in React and Express (MySQL)
https://medium.com/@avanthikameenakshi/crud-react-express-99025f03f06e
(última visita: 28/08/2018)

Building a Simple CRUD App with React + Redux: Part I
https://www.thegreatcodeadventure.com/building-a-simple-crud-app-with-react-redux-part-1/
(última visita: 28/08/2018)

Express & Redux: Different Tree, Same Bananas
https://medium.com/@ryanlynch/express-redux-different-place-same-bananas-11f5e5a88a8a
(última visita: 28/08/2018)

Qué es REST
https://desarrolloweb.com/articulos/que-es-rest-caracteristicas-sistemas.html
(última visita: 30/08/2018)

Información suministrada por la cátedra por medio de PEDCO.
http://pedco.uncoma.edu.ar/course/view.php?id=1732
(última visita: 30/08/2018)

¿Cúal es la diferencia entre REST y RESTful?
https://es.stackoverflow.com/questions/2512/cual-es-la-diferencia-entre-rest-y-restful
(última visita: 30/08/2018)

Reactjs 15 Vs 16 – Complete Lookup
http://www.aztutorialz.com/reactjs-15-vs-16-complete-lookup/
(última visita: 30/08/2018)

React v16.4.2: Server-side vulnerability fix
https://reactjs.org/blog/2018/08/01/react-v-16-4-2.html#160x
(última visita: 30/08/2018)

Understanding the React Component Lifecycle
http://busypeoples.github.io/post/react-component-lifecycle/
(última visita: 01/09/2018)