Universidad de Costa Rica
Sede del Pacífico

Carrera:
Informática y Tecnología en Multimedia

Curso:
Desarrollo De Aplicaciones Interactivas II

Tema:
Proyecto de implementación de React.js

Profesor:
Gerardo Falcón

Estudiantes:
Carlos Sagot Díaz B32295

I Semestre, 2016

INTRODUCCIÓN

Esta es una pequeña reseña y tutorial de un framework desarrollado por la empresa Facebook llamado React.js. Te preguntarás, para qué Facebook se molestaría en desarrollar y monitorear un software de alto rendimiento y entregarlo a la comunidad como “Open Source”? La respuesta es sensilla, desarrollo rápido de web y javascript. Nos encontramos en una época en la cual la mayoría del comercio e información se maneja vía internet; es decir vía web. Los navegadores son una de las principales fuentes de desarrollo e innovación, por ende las tecnologías de información aceleran apresuradamente para cumplir con la demanda generada dia con dia en el mundo, he aquí la razón de la pregunta anterior, desarrollar cada vez más rápido. Escribir menos y producir más. Teniendo en cuenta la información anterior vamos a referirnos un poco a React como una herramienta indispensable para el desarrollo de aplicativos web, con la cual podés controlar y manejar a tu antojo todo lo que respecta a estados y propiedades de un objeto. Poniendo como ejemplo el carrito de compras de amazon, cuando seleccionas un artículo para el carrito este lo guarda y lo mantiene en su base de datos, con React podríamos hacer lo mismo con un estado, los estados funcionan para darle valores cambiantes a tus objetos, es decir; podemos tener un producto con valor 0 para su estado y en cuanto se agregar este producto al carrito su valor cambia de 0 a 1 para diferenciarlo del resto y cuando se elimina el producto del carrito su estado vuelve a estar en 0, genial no?. Esta es solo una de las facilidades que React posee ahora bien la investigación presente se trata de un mantenimiento CRUD usando MySQL así que los demás componente de React no se verán en esta investigación.

TEMA: React.js

¿Qué es?
Librería Javascript de facebook basada en componentes. Qué quiere decir esto? Que nuestros proyecto puede ser modulares, agregando asi componentes y propiedades a donde se desee para mejorar el rendimiento de dicha.

Además está enfocada en la visualización y nos permite utilizarlo aunque nuestro proyecto ya esté usando un framework MVC que sea controlador mientras React se encargue de las vistas. Por sus ventajas en el manejamiento del DOM se convierte en un poderoso amigo para el desarrollador web.

¿Comó funciona el DOM virtual?
Imaginemos que tienes una copia de una persona, su cabello sus ojos son idénticos al original y refleja su estado actual. Así es como React utiliza el DOM, si le hacemos algunos cambios a esta persona en React sucede lo siguiente: en primera instancia React ejecuta un ”diffing” (identificación de cambios en las propiedades o estado del objeto), el segundo paso es el diff que compara el archivo del DOM con el nuevo y lo actualiza el antiguo pero solo sus nuevas características genial no?.

¿Cuáles son sus características?

- Utiliza JSX(mezcla de HTML+Javascript) que facilita a gran escala como digitamos el código.

- React native, permite nos hacer y compartir codigo para web, iphone o android devices, la versatilidad de esta biblioteca es enorme.

- Se integra con muchos frameworks gracias a su compatibilidad para ejecutarse y centrarse en la visualización de usuarios.

¿Por qué debería de usarlo?

- Basado en componentes
  Creando tus propios componentes y re-utilizarlos en el mismo u otros.

- Eficacia
  Por su manejo del DOM virtual, donde se encuentran todos su componentes lo vuelve una herramienta increíblemente flexible y de alto rendimiento.

- JSX
  Esta es una de las mayores razones por utilizarlo, simplifica la escritura de Javascript.

- Comunidad

> React al ser una herramienta Open Source disponible en GitHub, nos permite seguir avanzando y teniendo cada vez más y mejores componentes creador por los mismos usuarios, además del mantenimiento que facebook le brinda.

CONSISTE EN

Inicialmente React fue diseñado como un puerto JavaScript de XHP(versión de php que facebook creo). Pero surgió un problema con el XHP: necesitaba hacer muchas consultas al servidor y XHP no lo resolvió. Al considerar usar XHP en los navegadores usando JS y así surgió React.
React consiste en cuantificar y reutilizar componentes que lo desarrolladores crean para así simplificar el modo en que se construye una aplicación web, y aun mejor, poder utilizarlos en otros proyectos sin ningun problema ademas de repetirlos sin alterar el resultado de la página.

¿SE USA PARA?

React no es un framework MV, ni está pensado para competir con otros frameworks, de hecho podemos pensar que React es una herramienta diseñada para trabajar conjuntamente con otros frameworks y elevar su nivel de funcionalidad.

IMPLEMENTACIÓN

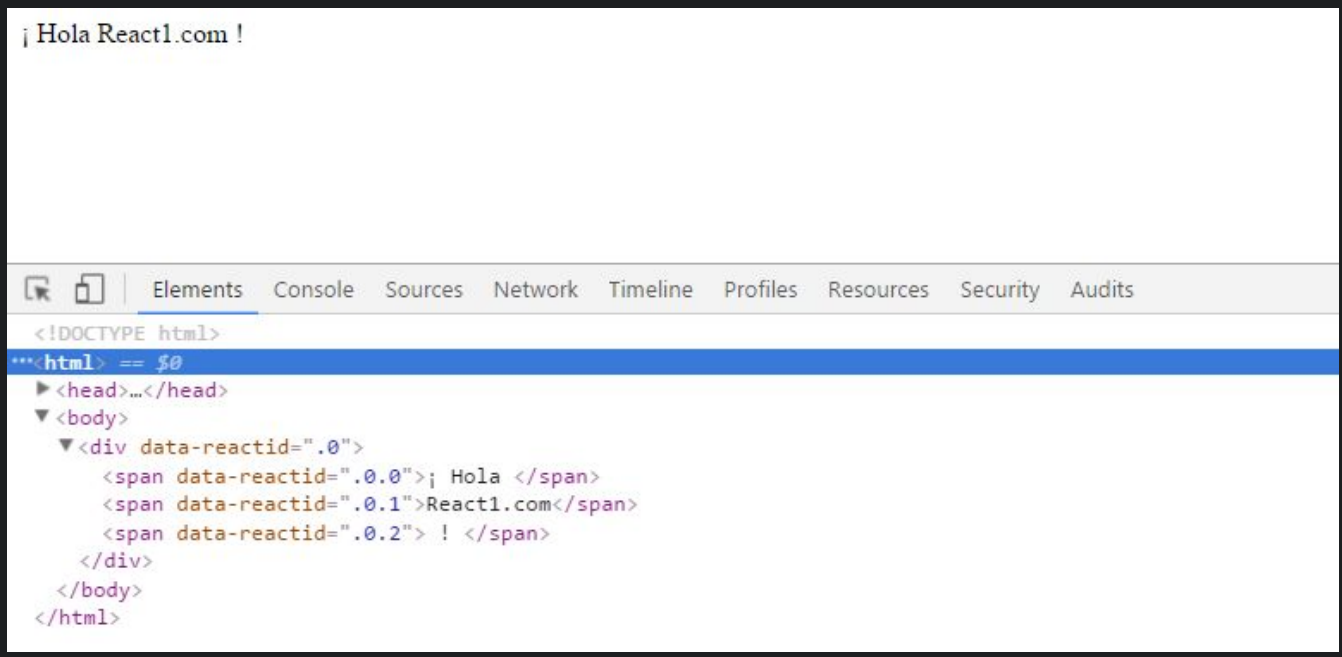
Para implementar React js en nuestro proyecto solo necesitamos unas simples lineas de codigo para importar la libreria.

```
1
2       <script src="http://fb.me/react-with-addons-0.12.0.js"></script>
3       <script src="http://fb.me/JSXTransformer-0.12.0.js"></script>
4
```

aquí vemos dos llamados script, el cual es propiamente React y el lector de código JSX.
Ahora vamos a ver cómo se crea un **Hello React.**

```
<!DOCTYPE html>
<html>
  <head>
    <title>React Example 1</title>
    <meta charset="utf-8">
    <script src="http://fb.me/react-with-addons-0.12.0.js"></script>
    <!--Interpretador de JSX-->
    <script src="http://fb.me/JSXTransformer-0.12.0.js"></script>
  </head>
  <body>
      <div id="ejemplo"></div>
        <script type="text/jsx">
          var Saludo= React.createClass({ //estamos generando una funcion
            render:function(){
              return <div>&iexcl; Hola {this.props.nombre} ! </div>;
            }
          });
          React.render(
            <Saludo nombre="React1.com"/>, document.body);
            //utilizamos la variable creada "saludo" y le adjuntamos el valor de la variable "nombre "
    </script>
  </body>
```

Para empezar utilizamos un div para almacenar nuestra data script, el siguiente paso es abrir un script de tipo texto que comprenda jsx. Declaramos una función de tipo saludo que recibe o genera una clase que devuelve el render y el render lo obtenemos del **return**, este devuelve: la etiqueta, un símbolo de exclamación, Hola, **{this.props.}** argumentos que se le agregan a la función y por último la variable **{nombre}**, finalizando con div. La información anterior es lo que se va a pasar dentro del render al **document.body.**



Así se ve el primer ejemplo utilizando React, y en consola vemos que es lo que el document.body genera.

EJEMPLO PRÁCTICO

Consiguiente vamos a ver un sistema CRUD elaborado en React.js. Su funcionamiento principal es el de, agregar, leer, modificar y eliminar, conjunto con una base de datos MySQL.

Ejemplo CRUD:

PASO A PASO

1. Para empezar necesitamos vamos a agregar dos librerías más, bootstrap y jquery aparte de las que ya teníamos.

```
<!DOCTYPE html>
<html>
   <head>
     <script src="http://fb.me/react-with-addons-0.12.0.js"></script>
     <script src="http://fb.me/JSXTransformer-0.12.0.js"></script>
     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-2.1.3.min.js"></script>
  </head>
```

2. Seguidamente vamos a elaborar la siguiente estructura HTML :

```
<div class="container">
      <div class="row">
            <div class="col-md-3"></div>
             <div id="Tareas" class="col-md-6"></div>
             <div class="col-md-3"></div>
      </div>
</div>
```

3. Ahora al script que ya habíamos generado con al inicio le vamos a agregar dos variables para identificar cuando un curso esté seleccionado :

```
<script type="text/jsx">
var Espera = {color: 'black'};
var Listo = {color: 'red', textDecoration:'line-through'};
```

4. Ahora vamos a inicializar una variable tipo “ListaCursos”, la cual vamos a usar para generar todo nuestro código respectivo con la lista. Dentro de esta clase vamos a crear la función “CreearElementosDeLista” con parametro itemText que va a retornar una serie de información codificada en html. El atributo “onChange” va a ser ejecutado cuando “checked” cambie de estado el estado. Seguidamente en la etiqueta span verifica si el estatus está en 0 es “Espera” y 1 para “Listo”. Al final llamamos a la función render para que imprima cada uno de los elementos que nuestra base de datos contenga.

```
var ListaCursos = React.createClass({
  CrearElementosDeLista: function(itemText) {
    return <li className="ui-state-default list-group-item" >
    <div className="col-md-11">
    <input type="checkbox" ref={itemText.id} onChange={this.props.EstadoTareaRealizada.bind(this,itemText)} checked={itemText.estatus==0?true:false} />
    <span style={itemText.estatus==0?  Listo :Espera} > {itemText.titulo}</span>
    </div>
    <a className="btn btn-danger" onClick={this.props.EstadoBorrar.bind(this, itemText)} >x</a>
    </li>;
    },
    render: function() {
    return <ul id="sortable" className="list-unstyled list-group">{this.props.ElementosLista.map(this.CrearElementosDeLista)}</ul>;
    }
});
```

5. Ahora vamos a inicializar la variable “AplicacionesCursos” y creamos la funcion predeterminada “componentWillMount” que nos permite traer información de por medio de AJAX y cuyos datos pueda que tengan que ser mostrados.

```
var AplicacionCursos = React.createClass({
   componentWillMount: function() {
                        this.actualizarInformacion();
   },
```

6. Con el siguiente método vamos a hacer mención de “actualizarInformacion” por medio de “$.get( )” a un archivo llamado “index.php”.

```
45          actualizarInformacion: function() {
46            $.get("Datos/index.php", function(resultado) {
47                 this.setState({ElementosLista: resultado});
48            }.bind(this));
49          },
```

7. getInitialState sirve para inicializar variables de estado que tenga nuestro component, y el “return” retorna la lista.

```
51          getInitialState: function() {
52             return {ElementosLista: [], text: ''};
53          },
```

8. Creamos otra función “EstadoBorrar” que le entra el elemento a borrar, por medio del ajax vamos a enviar una petición a index.php para borrar el elemento seleccionado y al final vamos a llamar a “actualizarInformacion”.

```
56          EstadoBorrar: function(ElementoAborrar, e) {
57            $.ajax({
58                  url: "Datos/index.php",
59                  dataType: 'json',
60                  type: 'POST',
61                  data: {id:ElementoAborrar.id,accion:2},
62                  success: function(data) {}.bind(this)
63                });
64                this.actualizarInformacion();
65          },
```

9. Su única función es agregar el elemento texto al valor ”e”.

```
67          EstadoCambio: function(e) {
68             this.setState({text: e.target.value});
69          },
```

10. En este caso la función recibe “ElementoModificar” . El indice que le entre se guarda como la instrucción. “valorEstatus” va a cambiar el estatus de dicho objeto, si está en 0 se cambia a 1 y si esta en 1 se cambia a 0.

```
71        EstadoTareaRealizada: function(ElementoModificar, e) {
72            var indice = this.state.ElementosLista.indexOf(ElementoModificar);
73            var valorEstatus=this.state.ElementosLista[indice].estatus=(this.state.ElementosLista[indice].estatus==1)?0:1;
74            $.ajax({
75                url: "Datos/index.php",
76                dataType: 'json',
77                type: 'POST',
78                data: {estatus:valorEstatus,id:ElementoModificar.id,accion:3},
79                success: function(data) {}.bind(this)
80              });
81             this.setState({ElementosLista: this.state.ElementosLista});
82        },
```

11. “EstadoSumit” lo que hace es enviar la información con la acción que es la de agregar. Se envia el titulo(nombre) con su estado con accion 1, además el success recibe toda la información en un data, y usamos el bind para ayudar a actualizar.

```
84          EstadoSubmit: function(e) {
85                $.ajax({
86                  url: "Datos/index.php",
87                  dataType: 'json',
88                  type: 'POST',
89                  data: {titulo:this.state.text,accion:1},
90                  success: function(data) {}.bind(this)
91                });
92                this.actualizarInformacion();
93          },
```

12. Casi finalizando aplicamos una función render para que nos devuelva toda la información de la lista de cursos. En esta función final, se llama todos los metodos que hicimos.

```
95    render: function() {
96          return (
97              <div>
98                <h3>Lista de Cursos</h3>
99                <form onSubmit={this.EstadoSubmit} >
100               <input className="form-control" placeholder="Agregar a la lista"  onChange={this.EstadoCambio} value={this.state.text} />
101               <br/>
102               <input type="button" onClick={this.EstadoSubmit} className="btn btn-success btn-block"
103               value={'Agregar a lista #' + (this.state.ElementosLista.length)} />
104               </form>
105               <hr/> //linea
106               <ListaCursos ElementosLista={this.state.ElementosLista}
107             EstadoBorrar={this.EstadoBorrar} EstadoTareaRealizada={this.EstadoTareaRealizada}  />
108             </div>
109         );
110       }
111   });
```

13. Al final, llamamos a la clase “AplicacionesCursos” al documento con el Id que le dimos al inicio al div.

```
113         React.render(<AplicacionCursos />, document.getElementById('Cursos'));
114       </script>
115
116       </body>
117     </html>
```

14. Esta es la instalación para poder Insertar, Eliminar, Modificar y Leer.

```php
<?php
$pdo=new PDO("mysql:host=localhost;dbname=lab5;","root","");
     switch($_POST['accion']){

    // Inserta la información
    case 1:
      $statement = $pdo->prepare("INSERT INTO tareas(id,titulo,estatus)VALUES(null,:titulo,1)");
      $statement->execute(array("titulo" => $_POST['titulo']));
      break;
      // Borrar la información
      case 2:
      $statement = $pdo->prepare("DELETE FROM tareas WHERE id=:id");
      $statement->execute(array("id" => $_POST['id']));
      break;
      // Modifica la información
      case 3:
      $statement = $pdo->prepare("UPDATE tareas SET estatus=:estatus WHERE id=:id");
      $statement->execute(array("estatus" => $_POST['estatus'], "id" => $_POST["id"]));
      break;
      // Consulta la información
      default:
      header('Content-Type: application/json'); //para que pueda leerse
      $statement=$pdo->prepare("SELECT * FROM tareas");
      $statement->execute();
      $results=$statement->fetchAll(PDO::FETCH_ASSOC);
      $json=json_encode($results);
      echo $json;
      break;
}
?>
```

CONCLUSIÓN

Con esta breve investigación que podemos entender cómo React.js? Bueno con React podemos ser creativos respecto a componentes, podemos utilizar un DOM que no nos va a consumir la misma memoria que tool tradicional, no digo que las otras aplicaciones estén mal simplemente React es veloz.

Visto lo expuesto notamos las principales características y ventajas sobre esta herramienta que viene a facilitarnos la vida como programadores de aplicaciones, utilizando nuevas tendencias como lo es JSX entre otras.

Consiguiente aclaramos que con React las posibilidades son ilimitadas a la hora de trabajar con proyectos que ya han sido iniciados con otra herramienta de desarrollo web.