Desarrollo de Aplicaciones Web

Tema Nº13:REACT JS – NAVEGABILIDAD II

Indicador de logro Nº13:Construye interfaces de usuario por medio de la librería de JavaScript a través de React JS desarrollada por Facebook para optimizar la perfomance de las aplicaciones web.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº13:**

REACT JS – NAVEGABILIDAD II

**Subtema 13.1:**

¿Qué es React Router?

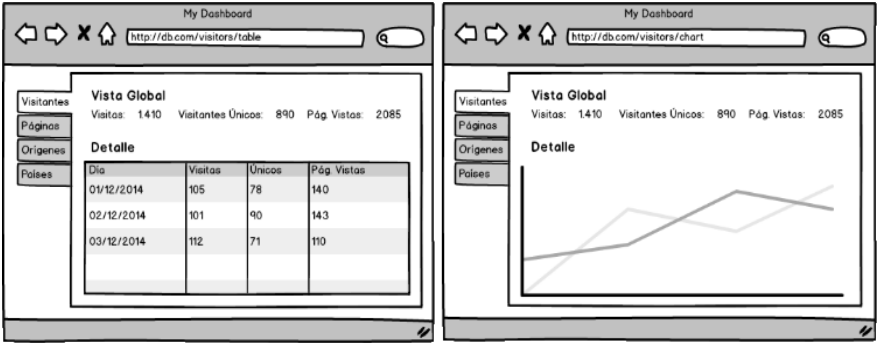
Básicamente es un conjunto de componentes de navegación que podemos utilizar en nuestra aplicación web o aplicación con React Native. En esta guía se enfocará en la parte web por lo que con esta librería podemos lograr lo siguiente:

* Establecer rutas en nuestra aplicación (Ejem. Home, Inicio de sesión, etc.)
* Crear páginas públicas y privadas.
* Realizar redirecciones a otras páginas según ciertas condiciones
* Acceder al historial de navegación
* Crear rutas para páginas 404 (no encontradas)

React Router es una libería para gestionar rutas en aplicaciones que utilicen ReactJS. Está inspirada en el sistema de enrutado de Ember.js y su forma de manejar las rutas es un poco diferente de lo que podemos ver en otros sistemas, como ASP.NET MVC, AngularJS, Express o Compojure, por poner unos cuantos ejemplos.

La idea fundamental es que podemos tener rutas anidadas mapeadas a vistas anidadas. ¿Qué quiere decir esto? Vamos a verlo con un ejemplo.

Imaginemos el típico dashboard de estadísticas de una web. Tenemos distintos tipos de métricas: Visitantes, Páginas, Orígenes, etc., y para cada una de ellas podemos una visión global con un par de datos significativos y los detalles, que podemos mostrarlos de forma tabular o mediante una gráfica:

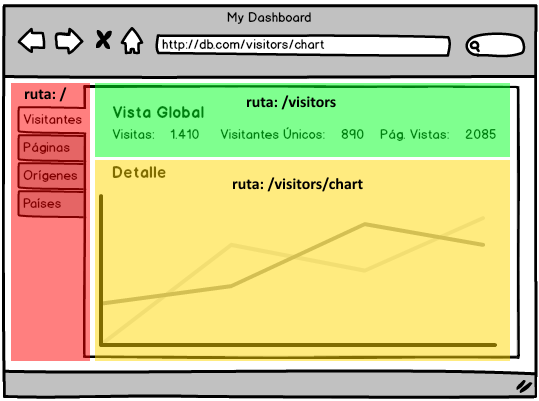


Existen varias partes de la página que se van a ir repitiendo entre las distintas «subpáginas» del sitio. Todas tendrán las pestañas de la parte izquierda para navegar entre secciones y, dentro de cada sección, tanto en la vista de tabla como la vista de gráfico se repetirá la parte superior en la que aparece el resumen.

Una forma habitual de resolver esto es mediante el uso de plantillas que permitan «incluir» fragmentos de html comunes en varias páginas, de una forma similar a cómo se hace con las Master Pages de ASP.NET MVC o los layouts de Jade.

En el caso de React Router la forma de resolver este tipo de situaciones es ligeramente diferente y se basa en el uso de rutas anidadas. Al definir las rutas, podemos ir definiendo parte de la estructura de la página y delegar en las rutas hijas la generación del resto de la estructura.

Si pensamos en el ejemplo anterior, podríamos tener las siguientes rutas:



* **/ :** Ruta base en la que se define el menú de la izquierda con las distintas secciones del dashboard.
* **visitors** : Ruta para visitantes en la que se define el resumen de visitantes
* **table** : Ruta con la tabla
* **chart** : Ruta con el gráfico
* … resto de rutas

De esta forma evitamos que repetir nada entre una «página» y otra, y podemos implementarlo todo mediante componentes de ReactJS independientes.

**Subtema 13.2:**

¿Qué es React, Redux y cómo se relacionan?

**¿Qué es React?**

React es una librería Javascript focalizada en el desarrollo de interfaces de usuario, o al menos así es como React se define.

La realidad es que React es una herramienta poderosa con la cual podemos crear todo tipo de aplicaciones web, SPAs (Single Page Application) o inlcuso aplicaciones para Android y iOS.

**¿Qué es el virtual DOM?**

Si entramos al mundo de ReactJS, vamos a escuchar hablar muchas veces sobre el Virtual DOM. Este concepto nace de una idea brillante, actualizar únicamente la parte del DOM que necesita cambiar en vez de todo el DOM real. En las aplicaciones Javascript de antes, se recibe el JSON del servidor y se genera nuevo HTML a renderizar, lo que actualiza toda la página en cada cambio.

Lo mejor de React es su ecosistema; la cantidad de ejemplos en internet, herramientas disponibles y complementos hace a la plataforma accesible y potente. Una de las mejores integraciones es Redux.

**¿Qué es Redux?**

No hay mejor descripción que la que ellos mismos se dan:

Redux es un contenedor predecible del estado de aplicaciones JavaScript.

Redux es una excelente herramienta para manejar el estado de una aplicación. Sus principales beneficios son:

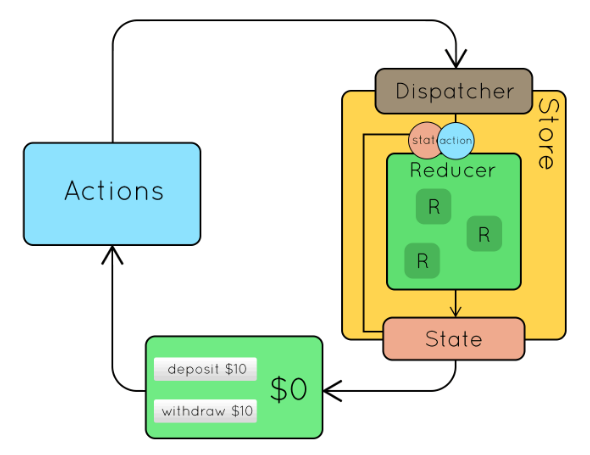
* Estado global e innmutable
* Mayor control del estado de la aplicación y el flujo de datos
* Arquitectura escalable de datos

Con apenas 2kb, Redux es una herramienta escalable y potente para manejar el estado de cualquier aplicación Javascript

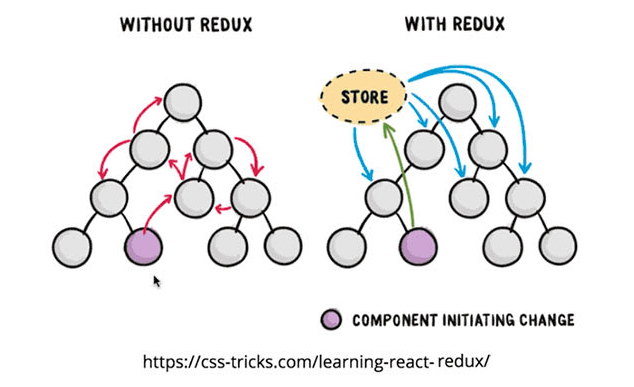
**El flujo de la información**

La principal ventaja de Redux es cómo administra los cambios de estado. Para entenderlo, es necesario conocer 3 conceptos clave:

1. El Store como la única fuente de la verdad
2. El State es de solo lectura
3. Los cambios al State pueden hacerse únicamente a través de acciones (actions) y funciones puras (reducers)



De esta forma, se logra centralizar el estado de la aplicación y por lo tanto unificar el lugar para realizar cambios. Esto hace el desarrollo muchísimo más simple:



Todo el estado de tu aplicación esta almacenado en un único árbol dentro de un único store. La única forma de cambiar el árbol de estado es emitiendo una acción, un objeto describiendo que ocurrió.

Para especificar como las acciones transforman el árbol de estado, usas reducers puros. ¡Eso es todo!

**1. MATERIALES**

Para la experiencia a realizar se requiere lo siguiente:

1. EQUIPO

Se requiere tener un equipo de cómputo (PC o laptop).

1. HERRAMIENTAS

* Visual Stude Code
* Node.js on WSL 2
* Npm
* Windows 10
* JavaScript

**2. PROCEDIMIENTO**

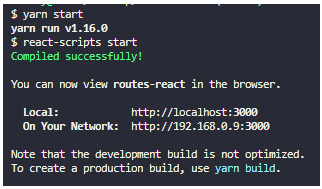
**Creando nuestro proyecto**

Para crear nuestro proyecto como lo mencione antes vamos a usar **create-react-app** para eso ejecutamos el siguiente comando en la terminal.

**npx create-react-app routes-react**

Esperamos que instale y cree nuestro proyecto y accedemos desde la terminal con: **cd routes-react**

Abrimos nuestro editor de código si estas usando **VScode** puede usar la terminal parado en la raíz de nuestro proyecto usamos **code .** para abrirlo desde VScode.

Si queremos probar nuestro proyecto podemos correr **npm start** o **yarn start** desde la terminal. Hasta el momento tenemos un proyecto de react listo para modificar y agregar nuestros componentes e instalar **React Router**.

**Instalación de React Router**

Para instalar la librería solo tenemos que ir a la terminal estar ubicados en la raíz de nuestro proyecto y ejecutar el siguiente comando.

**npm install react-router-dom**

**yarn add react-router-dom**

**Trabajando con React Router**

Teniendo todo listo ahora si vamos a nuestro editor de código y abrimos el archivo **App.js** que está ubicado en **src/App.js** acá vamos a limpiar muchas cosas hasta que al final tengamos algo como el siguiente código.





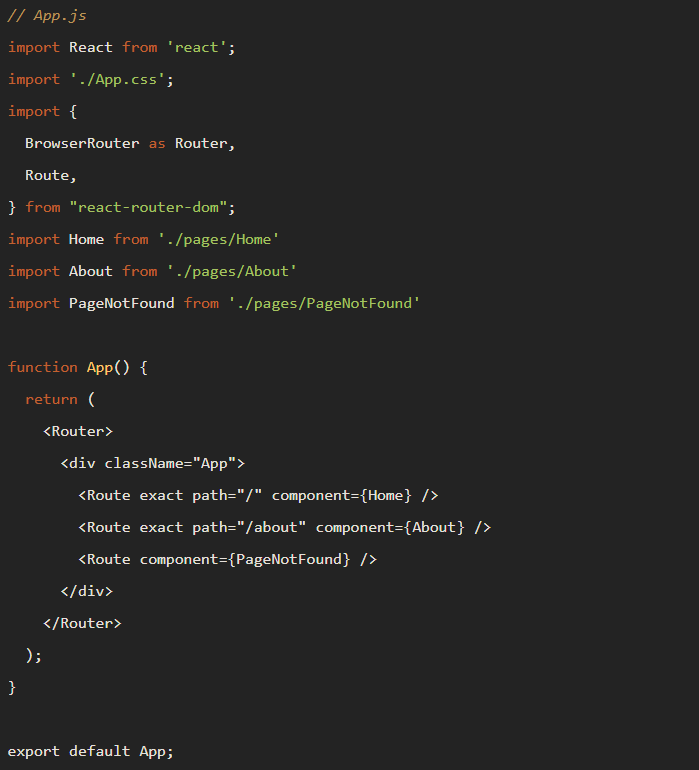
Importamos nuestro componente **BrowserRouter** le damos un nombre **Router** también importamos **Route** de **react-router-dom**. Envolvemos nuestra aplicación con Router y definimos nuestra primera ruta en este caso nuestro home le indicamos que debe ser exacta la ruta y que haga render de nuestro componente Home pero donde esta nuestro componente Home bueno vamos a crearlo.



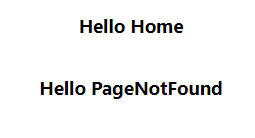
Este es nuestro componente **Home** que está dentro de una carpeta llamada **pages/Home** y solo tenemos un **h3** con un texto. Si vamos al navegador a **http://localhost:3000/** vamos a ver solo el texto.

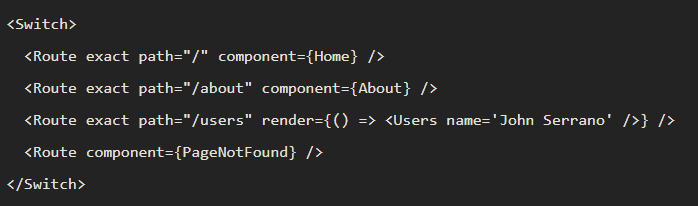


Ahora vamos a crear otras páginas y una página para manejar el error 404.



Si vamos al navegador después de agregar estas nuevas rutas vamos a tener un pequeño problema y es que también vamos a tener el render de nuestro componente **PageNotFound** para solucionar esto lo que tenemos que hacer es envolver nuestras rutas con el componente **Switch**.





Listo con esto ya solucionamos ese pequeño problema. Antes de seguir vamos a crear un menú de navegación para precisamente eso recorrer todas nuestras páginas y transformar nuestro proyecto en una **SPA** para esto vamos a usar el componente **Link** de React Router.

Todos los componentes que están dentro de **./pages** solo tiene un **h3** con un texto.

**Link**

Con Link vamos a poder navegar por nuestra aplicación, este componente recibe las siguientes propiedades.

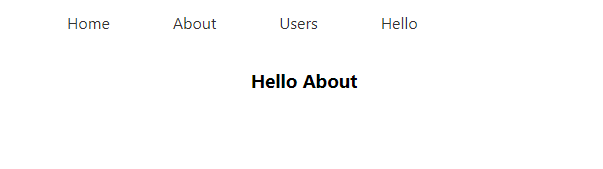
**To:** le podemos pasar un string, object o una function para indicarle la ruta a la cual queremos navegar.

**Replace:** cuando es verdadero, y hacemos clic en el enlace reemplazará la entrada actual en la pila del historial en lugar de agregar una nueva.



Agregamos el siguiente código en nuestro componente **App.js** dentro del **div** con la clase App, ahora podemos ver un menú de navegación en nuestra aplicación. Me falto mencionar debemos hacer el **import** de Link.





**Redirect**

Con este componente podemos causar un redireccionamiento a una ruta diferente a la ruta actual reemplazando el **location** actual y el historial de navegación. Tiene las siguientes propiedades.

* **From:** le pasamos un string u object para indicarle desde donde se va hacer el redireccionamiento.
* **To:** le pasamos un string u object hacia dónde vamos a realizar el redireccionamiento.
* **Push:** si es verdadero no modifica el location del historial por el contrario agrega esta nueva locación al historial.

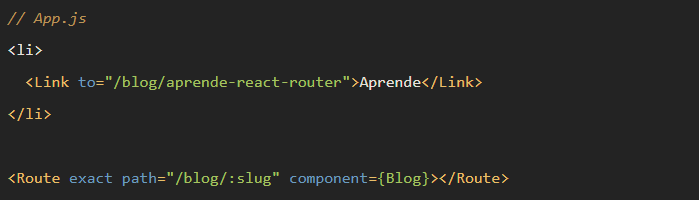
**<Redirect from="/redirect" to="/about" />**

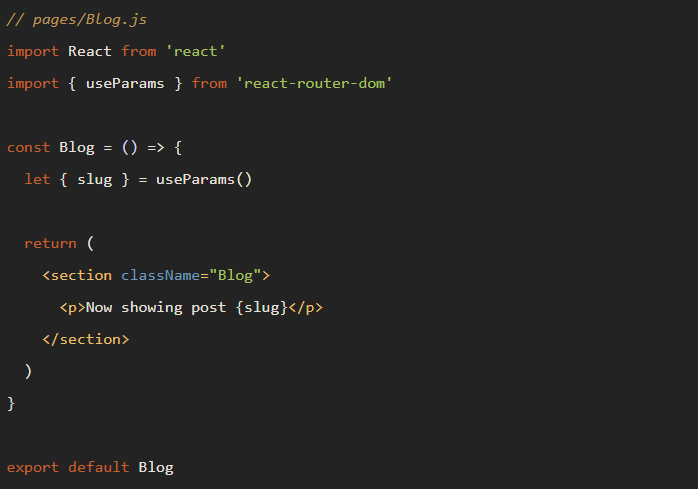
Agregamos ese componente dentro de **Switch**, ahora si vamos al navegador y en la barra de dirección escribimos lo siguiente **http://localhost:3000/redirect** vamos a ver que nos hace un redirect a la ruta /about. No olvides hacer el **import** de Redirect.

**Hook useParams**

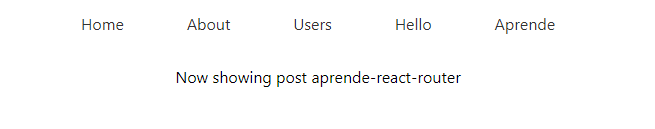
Con este **Hook** podemos acceder a los **params** de las rutas veamos un ejemplo para eso debemos crear un nuevo componente el cual se va encargar de usar el hook y debemos agregar un nuevo link al menú de navegación al igual que un Route.







Lo primero es importar el hook y obtener el **params** que definimos en el **Route**, con este hook es muy fácil acceder al params que indicamos en el Route si vamos al navegador y damos click en el nuevo link que tiene como nombre aprende vamos a ver lo siguiente.



Podemos concluir que **React Router** nos provee de todos componentes necesarios para convertir nuestro proyecto en una **SPA** y obtener una navegación entre nuestros componentes sin perder el historial de navegación.

**ACTIVIDAD VIRTUAL:**

1. **CUESTIONARIO TÉCNICO**

Revisa y analiza el tema desarrollado en la presente sesión, luego responde las siguientes preguntas propuestas:

* ¿Qué es React Router?
* ¿Qué es Redux?
* ¿Qué es Hooks in React? Dé una breve explicación
* Realice una App aplicando todo lo aprendido de esta guía

1. **ENLACES DE REFERENCIA**

* https://blog.koalite.com/2014/12/rutas-en-reactjs-con-react-router/
* https://ed.team/blog/crea-rutas-con-react-router
* https://dev.to/arielmirra/que-es-react-y-react-hooks-y-como-se-relacionan-4e1e
* https://johnserrano.co/blog/aprende-a-crear-rutas-con-react-router
* https://cloudacademy.com/course/routing-in-react/installing-the-react-router-dom/

1. **CONCLUSIONES DE LA EXPERIENCIA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_