El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

CICLO IV:

Desarrollo de Aplicaciones Web

MisionTIC2022

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

Sesión 06:

Desarrollo de Aplicaciones Web

Desarrollo de Front-End web con React

MisiónTIC2022

Objetivos de la sesión

Al finalizar esta sesión estarás en capacidad de:

1. Instalar el framework de React en un ambiente de desarrollo.
2. Crear un proyecto básico de React.

React

- React es una librería de Javascript para crear interfaces de usuario declarativas, basadas en componentes altamente reutilizables.
- Es una librería open-source, creada y mantenida por el equipo de Facebook.
- Actualmente se considera la librería de desarrollo Front-End más utilizada.
- Nos permite crear Single Page Applications (SPA), lo cual consiste en una página web que es construida y modificada por el cliente, es decir, las vistas son generadas por el cliente y no por el servidor.
- Dentro de sus implementaciones más populares nos encontramos con:
  - Create React App (CRA)
  - Gatsby
  - Next.js

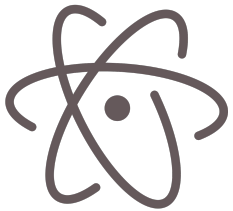
React - Instalación con CRA

- Para crear un proyecto o una web app con CRA deberemos seguir los siguientes pasos:
- Descargar e instalar la versión Long Term Support (LTS) de Node.js.
- Ejecutaremos el comando **`npx create-react-app <nombre de la app> --use-npm`** en nuestra terminal.
    - **`npx`**: Comando global de *npm* (controlador de paquetes de Node.js), usado para ejecutar scripts configurados globalmente en el repositorio de node.
    - **`use-npm`**: Opción utilizada para especificar que el controlador de paquetes que utilizaremos sea efectivamente npm, ya que si llegaramos a tener otro controlador de paquetes instalado, como *yarn*, este tomaría precedencia.

React - Instalación con CRA

- Una vez ejecutado el comando este creara un folder con el nombre de `<nombre de la app>` desde el directorio en el que nos encontremos desde la terminal.
- Podemos abrir nuestro editor de texto directamente con el folder de nuestro proyecto y nos encontraremos con la siguiente estructura de carpetas.






React - Estructura

- **Estructura de carpetas**

```
<nombre de la app>/
|    README.md
|    node_modules/
|    package.json
|    public/
|    |    index.html
|    |    favicon.ico
|    src/
|    |    App.css
|    |    App.js
|    |    App.test.js
|    |    index.css
|    |    index.js
|    |    logo.svg
```

- Para que el proyecto se compile, estos archivos deben existir con nombres exactos:
  - `public/index.html` es la plantilla de la página.
  - `src/index.js` es el punto de entrada de JavaScript.
- Se puede eliminar o renombrar los demás archivos.

React - Estructura

- Tomando en cuenta nuestra estructura de carpetas a partir de src:

```
src/
|    App.css
|    App.js
|    App.test.js
|    index.css
|    index.js
|    logo.svg
```

- Es importante darnos cuenta que tenemos la libertad de ordenar nuestras carpetas a nuestra disposición.
- Esto podría prestarse para generar una estructura de carpetas desordenadas que nos cueste escalar y mantener nuestro código.

React - Estructura

- Es considerado como buena práctica tener las siguientes carpetas:

```
src/
|    /components
|    /context
|    /hooks
|    /pages
|    /services
|    /store
|    App.css
|    App.css
|    App.js
|    App.test.js
|    index.css
|    index.js
|    logo.svg
```

- **`components`**: Aquí residirán la mayoría de nuestros componentes.
- **`context`**: Aquí ubicamos nuestros contextos de context API.
- **`hooks`**: Aquí definiremos hooks personalizados.
- **`pages`**: Aquí definiremos las vistas de nuestras rutas en la web app.
- **`services`**: Aquí configuraremos servicios de terceros como API's, Back-Ends, sockets, entre otros.
- **`store`**: Aquí configuraremos nuestro almacenamiento con redux API.

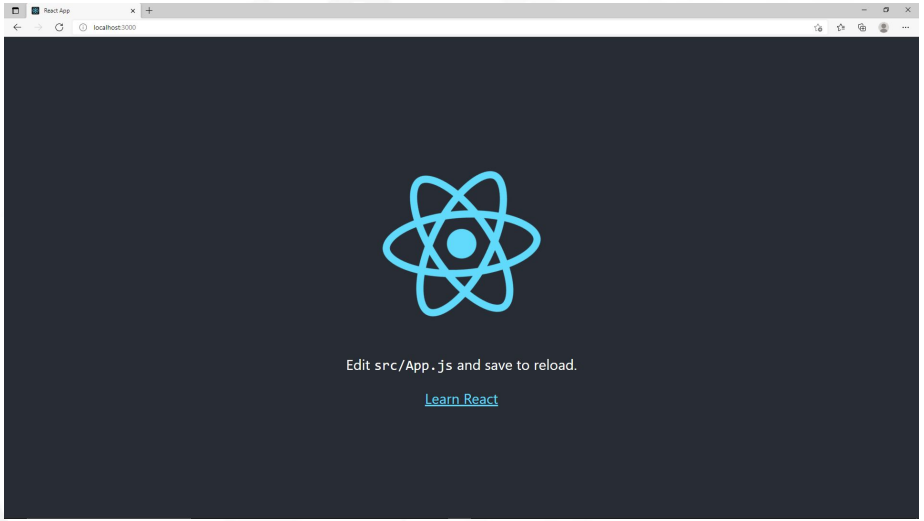
React - Ejecución de Aplicaciones

- Analizando nuestro archivo *package.json*, nos encontramos con los siguientes comandos bajo la configuración *scripts*:
    - **npm start**: Ejecuta nuestra aplicación en modo de desarrollo.
    - **npm test**: Ejecuta las pruebas unitarias de nuestra aplicación.
    - **npm run built**: Compila nuestra aplicación.
    - **npm run eject**: Hace visible la configuración webpack de nuestro proyecto (por favor no experimentar con esto sin tener conocimientos sobre webpack ya que esto afecta directamente el proceso de compilado y ejecución).
- Para ejecutar nuestra aplicación simplemente ejecutaremos *npm start*.

React - Ejecución de Aplicaciones

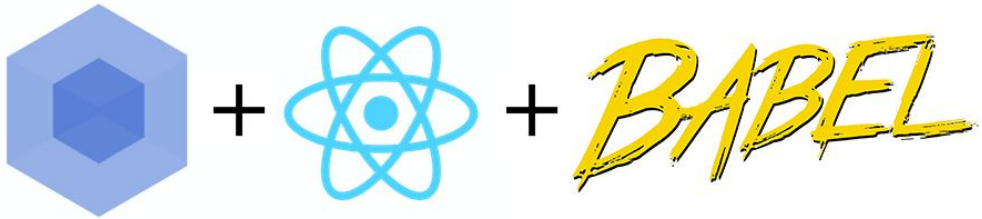
- Una vez nuestro entorno de desarrollo se esté ejecutando, nuestra app debería abrir el navegador web que tengamos configurado por defecto en la url *Localhost:3000*.

- Deberíamos ver lo siguiente:



React - JSX

- Es una extensión de la sintaxis de JavaScript (.jsx o .js)
- Prácticamente es JavaScript pero con más funcionalidades.
  - Nos permite embeber HTML dentro de nuestros archivos con JSX.
  - Nos permite acceder a funcionalidades propias de React.
  - Produce “elementos” de React.
- En últimas JSX es compilado a bloques de JS a través de Babel y de Webpack:



React - Renderizar Elementos

- Crearemos un archivo `Demo.js` en `src/components` con la siguiente estructura:

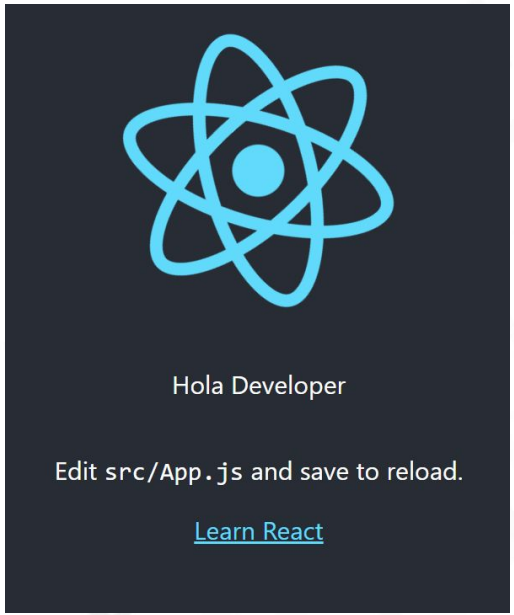
```
import React, {useState} from 'react';

const Demo = () => {
    const [name, setName] =
useState('Developer');
    return <p>Hola {name}</p>;
};

export default Demo;
```

React - Renderizar Elementos

- Luego en `App.js`:
  - Importamos Demo en la primera línea:
    - `import Demo from './components/Demo';`
  - Escribimos lo siguiente en la décima línea:
    - `<Demo />`

React - Renderizar Elementos

- Si observamos nuevamente nuestra aplicación notaremos un cambio:



  - Como podemos observar se añadió la línea “Hola Developer”.
  - Esto es posible debido a que lo que acabamos de hacer fue crear un componente de React.
  - Así mismo, creamos una variable `name` y la utilizamos dentro de nuestro JSX (código con un formato similar a HTML), usando llaves.

React - Componentes

- React introduce dos tipos de componentes, componentes basados en clases y componentes funcionales.
- La diferencia entre ambos componentes afecta directamente la forma en como trabajamos en react.
    - Diferencias en formato.
    - Diferencias en funcionalidades de React.
    - Diferencias en ciclo de vida.
    - Entre otros.
- Cabe resaltar que en **`Demo.js`** definimos el componente **`<Demo>`** de manera funcional.

React - Componentes basados en Clases

- Para ver directamente la diferencia entre componentes basados en clase y funcionales, reescribimos **<Demo>** como un componente basado en clases de la siguiente forma:

```
import React, { Component } from 'react';

class Demo extends Component {
    state = { name: 'Developer' };

    render() {
        return <p>Hola {this.state.name}</p>;
    }
}

export default Demo;
```

React - Componentes basados en Clases

- Contienen más **`boilerplate`** debido a que es más tipado con respecto a los componentes funcionales.
- Cómo definimos clases y no funciones directamente, entonces requerimos de utilizar el keyword **`this`** para referirnos al estado de nuestro componente.

React - Componentes Funcionales

- Los componentes funcionales son considerados como la forma moderna de desarrollar React.
- Esto se debe a que:
  - Es más sencillo de leer y escribir.
  - Nos facilita controlar el estado de nuestros componentes y evitamos conflictos con el keyword `this` debido al alcance o `scope` del mismo.
  - React ha estado introduciendo nuevas funcionalidades para este tipo de componentes en sus versiones más recientes.
- Por estas razones, utilizaremos componentes funcionales en estas sesiones.

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

Ejercicios de práctica

Mision TIC 2022

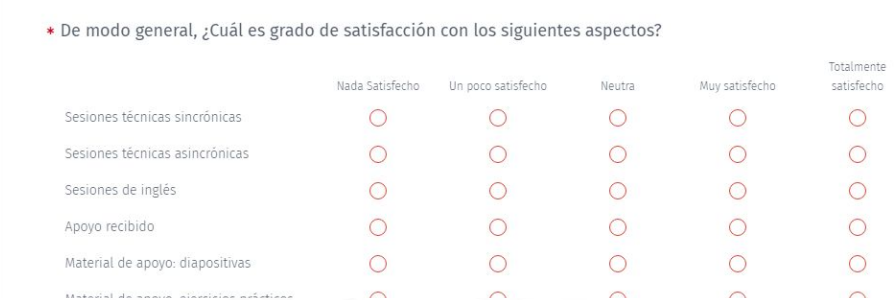
Referencias

- https://es.reactjs.org/docs/getting-started.html
- https://create-react-app.dev/docs/getting-started/#quick-start
- https://create-react-app.dev/docs/folder-structure
- https://create-react-app.dev/docs/available-scripts
- https://reactjs.org/docs/introducing-jsx.html
- https://reactjs.org/docs/rendering-elements.html
- https://reactjs.org/docs/components-and-props.html#function-and-class-components

El futuro digital es de todos | MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

Seguimiento Habilidades Digitales en Programación



* De modo general, ¿Cuál es grado de satisfacción con los siguientes aspectos?

	Nada Satisfecho	Un poco satisfecho	Neutra	Muy satisfecho	Totalmente satisfecho
Sesiones técnicas sincrónicas	○	○	○	○	○
Sesiones técnicas asincrónicas	○	○	○	○	○
Sesiones de inglés	○	○	○	○	○
Apoyo recibido	○	○	○	○	○
Material de apoyo: diapositivas	○	○	○	○	○

Completa la siguiente encuesta para darnos retroalimentación sobre esta semana ▼▼▼

https://www.questionpro.com/t/ALw8TZlxOJ

MisionTIC2022

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

¡GRACIAS

POR SER PARTE DE ESTA EXPERIENCIA DE APRENDIZAJE!

Mision TIC 2022