	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA, ESCUELA DE COMPUTACIÓN**
Ciclo I	**DISEÑO Y PROGRAMACIÓN DE SOFTWARE MULTIPLATAFORMA** **Guía de Laboratorio No. 2** **Introducción a React**

I. Resultados de aprendizaje

Que el estudiante adquiera los conocimientos necesarios para:

1. Familiarizarse con la sintaxis específica de JSX.
2. Comprender los principios fundamentales que sustentan la construcción de interfaces de usuario en React y otras bibliotecas que emplean JSX.
3. Desarrollar habilidades para la creación de aplicaciones web modernas y escalables, aplicando los conceptos aprendidos en la utilización de JSX.

II. Introducción

JSX, que significa "JavaScript XML", es una extensión de la sintaxis de JavaScript que se utiliza comúnmente con bibliotecas de JavaScript como React para describir la interfaz de usuario. JSX permite escribir código que se asemeja a la estructura de un documento XML o HTML, pero se integra directamente con JavaScript.

En lugar de escribir código de creación de elementos de interfaz de usuario de forma imperativa, como se haría en JavaScript puro, JSX proporciona una sintaxis más declarativa y fácil de leer. Aquí hay un ejemplo básico de JSX:

jsx

```
const element = <h1>Hola, mundo!</h1>;
```

Este código se parece mucho al HTML, pero en realidad es código JavaScript que utiliza JSX. Durante el proceso de construcción, JSX se compila a llamadas a funciones JavaScript regulares, generalmente funciones proporcionadas por la biblioteca que estás utilizando (por ejemplo, React.createElement en el caso de React). En este ejemplo, el código JSX se compilaría a algo similar a:

javascript

```
const element = React.createElement('h1', null, 'Hola, mundo!');
```

JSX también permite la interpolación de expresiones JavaScript dentro del código, lo que facilita la incorporación de lógica dinámica en la interfaz de usuario. Por ejemplo:

jsx

```
const nombre = "Usuario";

const element = <h1>Hola, {nombre}!</h1>;
```

En este caso, la variable nombre se interpola dentro del texto JSX, lo que resulta en un saludo personalizado.

En resumen, JSX es una extensión de la sintaxis de JavaScript que facilita la creación de interfaces de usuario en bibliotecas como React, combinando la potencia de JavaScript con una sintaxis similar a HTML.

IV DESARROLLO

Ejemplo 1:

1. Asegúrate de tener Node.js instalado en tu sistema.
2. Utiliza `create-next-app@latest`para crear un nuevo proyecto React.
   Ejecuta el siguiente comando en tu terminal:

   ```
   npx create-next-app@latest nombre-de-tu-app
   ```

   ejemplo:

   ```
   C:\Users\Karens_Medrano\Desktop>npx create-next-app@latest guia2
   ```

   Cambia al directorio de tu nueva aplicación:

   ```
   cd nombre-de-tu-app
   ```

   ejemplo:

   ```
   C:\Users\Karens_Medrano\Desktop>cd guia2
   ```

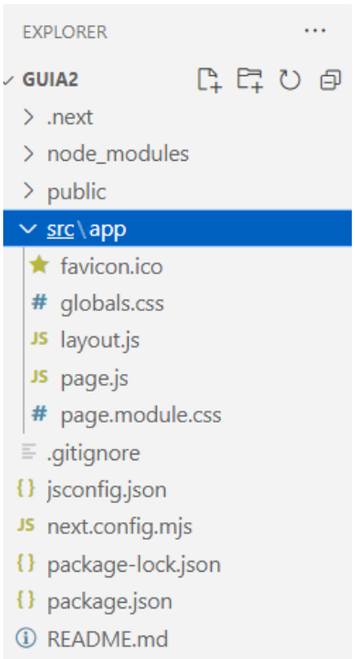
- Al abrir el proyecto en un editor, puede explorar la estructura de carpetas generada por

  ```
  C:\Users\Karens_Medrano\Desktop>npx create-next-app@latest guia2
  ```

- Aparecerá un aviso pidiéndote que confirmes algunas dependencias adicionales

  ```
  C:\Users\Karens_Medrano\Desktop>npx create-next-app@latest guia2
  √ Would you like to use TypeScript? ... No / Yes
  √ Would you like to use ESLint? ... No / Yes
  √ Would you like to use Tailwind CSS? ... No / Yes
  √ Would you like to use `src/` directory? ... No / Yes
  √ Would you like to use App Router? (recommended) ... No / Yes
  √ Would you like to customize the default import alias (@/*)? ... No / Yes
  Creating a new Next.js app in C:\Users\Karens_Medrano\Desktop\guia2.
  ```

- podemos abrir el proyecto y verificar que tenemos la siguiente estructura de archivos.

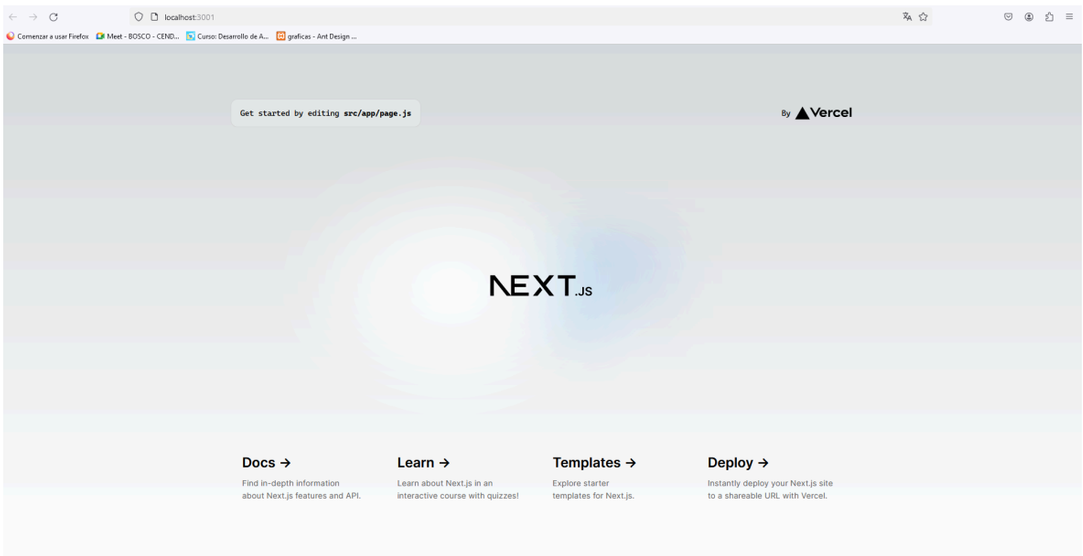


3. Inicia el servidor de desarrollo local ejecutando:
   `npm run dev`

ejemplo

```
C:\Users\Karens_Medrano\Desktop\guia2>npm run dev
```

- Esto abrirá tu aplicación en un navegador y se recargará automáticamente cuando realices cambios en el código.



4. modificaremos el archivo page.js ubicado en la carpeta src/app/page.js, con el siguiente código :

```
import styles from "./page.module.css";
const element = <h1>Hola, Mundo!</h1>;
export default function Home() {
  return (
    <main className={styles.main}>
      <div className="App">
        {element}
      </div>
    </main>
  );
}
```

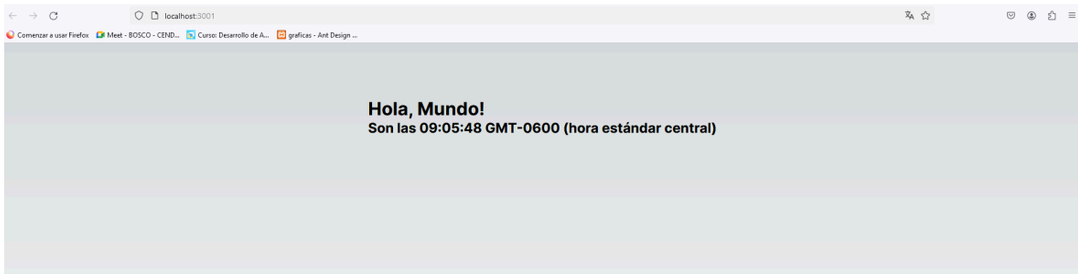
5. Deberemos ver el siguiente resultado en pantalla:

6. modificar el archivo page.js para agregar más elementos html a nuestro componente element:

```
import styles from "./page.module.css";
const element = (
  <>
```

```
    <h1>Hola, Mundo!</h1>
    <h2>Son las {new Date().toTimeString() }</h2>
  </>
  );

export default function Home() {
  return (
    <main className={styles.main}>
      <div className="App">
        {element}
      </div>
    </main>
  );
}
```

7. El resultado debera verse de la siguiente forma en nuestro navegador:



Ejemplo 2:

1. crea un nuevo proyecto usando `create-next-app@latest`
2. Reemplaza el contenido de `src/App/page.js` con el siguiente código:

```
import styles from "./page.module.css";
const Equipos = ({ equipos }) => {
  return (
<div className={styles.container__list}>
      <h2 className={styles.title}>Equipos de Fútbol</h2>
      {equipos.map((equipo) => (
        <div key={equipo.id}>
          <h3 className={styles.nameclub}>{equipo.nombre}</h3>
          <ul >
            {equipo.plantilla.map((jugador) => (
              <li className={styles.container__list} key={jugador.id}>
                <strong>{jugador.nombre}</strong>
                <p>Altura: {jugador.Altura}m <br></br> Peso:
{jugador.Peso}Kg</p>
              </li>
            ))}
          </ul>
        </div>
      ))}
    </div>
  );
};



export default function Home() {
```

```
   // Simula la obtención de datos desde tu JSON
   const equiposData = [
    {"id": 1,
    "nombre": "Real Madrid",
    "plantilla":[
      {"id": 1,"nombre":"Eden Hazard","Altura":"1.75","Peso":"74Kg"},
      {"id": 2,"nombre":"Gonzalo
García","Altura":"1.82","Peso":"74Kg"},
      {"id": 3,"nombre":"Karim Benzema","Altura":"1.85","Peso":"81Kg"}
    ]},
    {"id": 2,
    "nombre": "Barcelona",
    "plantilla":[
      {"id": 1,"nombre":"Marc-André ter Stegen
","Altura":"1.75","Peso":"74Kg"},
      {"id": 2,"nombre":"Iñigo
Martinez","Altura":"1.82","Peso":"74Kg"},
      {"id": 3,"nombre":"Gavi","Altura":"1.85","Peso":"81Kg"}
    ]}
    // ... agregar otros equipos
  ];


  return (
    <main className={styles.main}>
 <div>
      <h1>Mi Aplicación de Fútbol</h1>
      <Equipos equipos={equiposData} />
    </div>
    </main>
  );
}
```

3. Asegúrate de agregar el siguiente estilo CSS en `src/app/page.module.css`:

```
.main {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  align-items: center;
  padding: 6rem;
  min-height: 100vh;
}
.container__list {
  list-style-type: none;
  padding: 0;
}


.list {
  margin-bottom: 5px;
}


.title {
  margin: 20px 0;
  border: 0;
  border-top: 1px solid #ccc;
}


.nameclub {
  margin: 20px 0;
  border: 0;
  border-top: 1px solid rgb(29, 84, 185);
}
```

4. el resultado se muestra de la siguiente manera en el navegador:

Ejemplo 3:

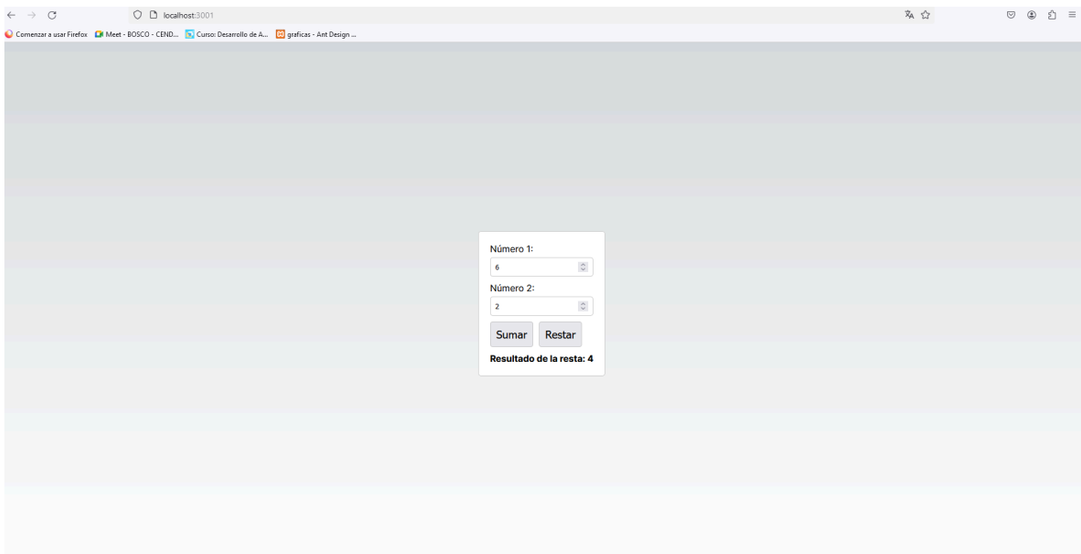
1. crea un nuevo proyecto usando `create-next-app@latest`
2. Reemplaza el contenido de `src/App/page.js` con el siguiente código:

```
"use client";
import { useState } from "react";
import styles from "./page.module.css";

export default function Home() {
  const [numero1, setNumero1] = useState('');
  const [numero2, setNumero2] = useState('');
  const [resultado, setResultado] = useState(null);


  const sumar = () => {
    const resultadoSuma = parseFloat(numero1) + parseFloat(numero2);
    setResultado(`Resultado de la suma: ${resultadoSuma}`);
  };

  const restar = () => {
    const resultadoResta = parseFloat(numero1) - parseFloat(numero2);
    setResultado(`Resultado de la resta: ${resultadoResta}`);
  };


  return (
    <main className={styles.main}>
    <div className={styles.calculadora}>
        <div className={styles.numeros}>
          <label className={styles.text}>Número 1:</label>
          <input className={styles.inputnum} type="number"
value={numero1} onChange={(e) => setNumero1(e.target.value)} />
        </div>
        <div className={styles.numeros}>
          <label className={styles.text} >Número 2:</label>
          <input className={styles.inputnum}  type="number"
value={numero2} onChange={(e) => setNumero2(e.target.value)} />
        </div>
        <div>
          <button className={styles.button}
onClick={sumar}>Sumar</button>
          <button className={styles.button}
onClick={restar}>Restar</button>
        </div>
        {resultado && <div
className={styles.resultado}>{resultado}</div>}
    </div>
    </main>
  );
}
```

3. Asegúrate de agregar el siguiente estilo CSS en `src/app/page.module.css`:

```css
.main {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  align-items: center;
  padding: 6rem;
  min-height: 100vh;
}

.calculadora {
  max-width: 400px;
  margin: auto;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
  background-color: #fff;
}

.numeros {
  margin-bottom: 10px;
}

.text {
  display: block;
  margin-bottom: 5px;
}

.inputnum {
  width: 100%;
  padding: 8px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.button {
  font-size: 1.2em;
  padding: 10px;
  margin-right: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  cursor: pointer;
}


.resultado {
  margin-top: 10px;
  font-weight: bold;
}
```

4. el resultado se muestra de la siguiente manera en el navegador:



Ejemplo 4:

1. crea un nuevo proyecto usando `create-next-app@latest`
2. Reemplaza el contenido de `src/App/page.js` con el siguiente código:

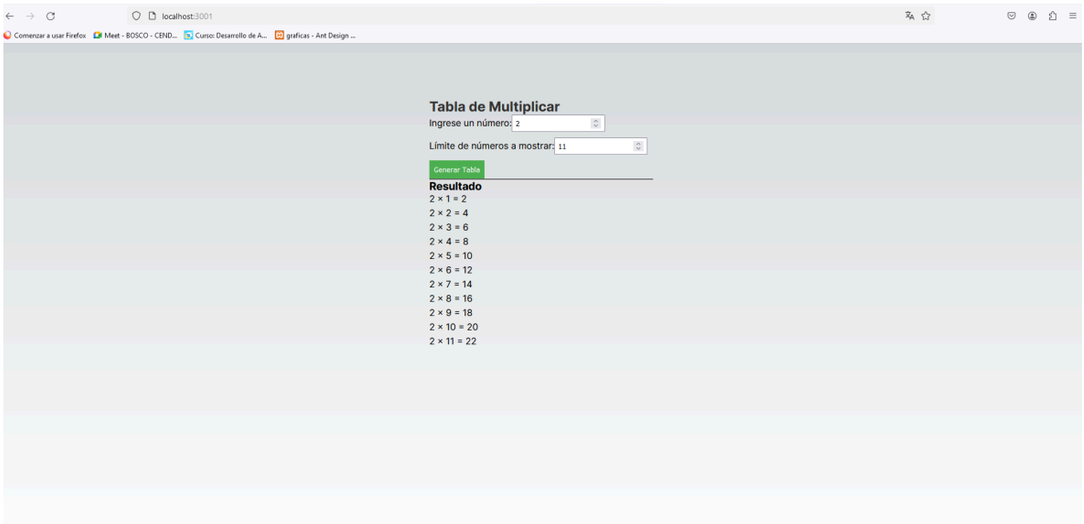
```
"use client";
import { useState } from "react";
import styles from "./page.module.css";

export default function Home() {
  const [numero, setNumero] = useState(1);
  const [limite, setLimite] = useState(10);
  const [resultado, setResultado] = useState([]);

  const generarTabla = () => {
    const nuevaTabla = [];
    for (let i = 1; i <= limite; i++) {
      nuevaTabla.push(`${numero} x ${i} = ${numero * i}`);
    }
    setResultado(nuevaTabla);
  };


  return (
    <main className={styles.main}>
<div>
      <h2 className={styles.title2}>Tabla de Multiplicar</h2>
      <label className={styles.text}>
        Ingrese un número:
        <input className={styles.input}
          type="number"
          value={numero}
          onChange={(e) => setNumero(parseInt(e.target.value))}
        />
      </label>
      <br />
      <label className={styles.text}>
        Límite de números a mostrar:
        <input className={styles.input}
          type="number"
          value={limite}
          onChange={(e) => setLimite(parseInt(e.target.value))}
        />
      </label>
      <br />
      <button className={styles.button} onClick={generarTabla}>Generar
Tabla</button>
      <hr />
      <h3>Resultado</h3>
      <ul className={styles.ul}>
```

```
        {resultado.map((item, index) => (
          <li className={styles.li} key={index}>{item}</li>
        ))}
      </ul>
    </div>
    </main>
  );
}
```

3. Reemplaza el contenido de `src/App.css` con el siguiente código:

```
.main {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  align-items: center;
  padding: 6rem;
  min-height: 100vh;
}

.title2 {
  color: #333;
}

.text {
  margin-right: 10px;
}

.input {
  margin-bottom: 10px;
  padding: 5px;
}

.button {
  padding: 8px;
  background-color: #4caf50;
  color: #fff;
  border: none;
  cursor: pointer;
}

.button:hover {
  background-color: #45a049;
}

.ul {
  list-style-type: none;
  padding: 0;
}

.li {
  margin-bottom: 5px;
}

.hr {
  margin: 20px 0;
  border: 0;
  border-top: 1px solid #ccc;
}
```

4. el resultado se muestra de la siguiente manera en el navegador:



V. DISCUSIÓN DE RESULTADOS

Todos los ejercicios deben ser entregados por medio de una Url del repositorio público .

1. En el segundo ejemplo, se requiere expandir la cantidad de equipos y realizar ajustes en la información de los equipos de fútbol. Ahora, es esencial incorporar la fotografía de los jugadores y presentar esta información de manera visual en la página web.
2. Para el ejemplo 3 Implemente la lógica necesaria para realizar las operaciones matemáticas: suma, resta, multiplicación, división, potenciación y raíces cuadradas.

   NOTA:

   a. Implementa una funcionalidad para borrar los números y reiniciar la calculadora.
   b. Asegúrate de manejar casos especiales, como la división por cero, y proporciona mensajes claros en caso de errores.
   c. Estiliza la aplicación para que sea visualmente atractiva y fácil de usar.

VI.Bibliografia

- React, una biblioteca de JavaScript. (2013-2024). Facebook, Inc. Jordan Walke. Recuperado de https://es.react.dev/