Sesión 4: Manejo de Estados y Estilos en React Native con Tailwind CSS

Objetivo de la sesión

Hoy vamos a entender a fondo dos conceptos clave en React Native mientras continuamos con nuestro proyecto **Carnet Estudiantil Digital**:

1. **Estados (state)** en componentes funcionales para manejar datos dinámicos.
2. **Tailwind CSS**, una herramienta poderosa para aplicar estilos de manera rápida y eficiente.

---

1. ¿Qué es el Estado (state) en React Native?

Definición:

El **estado (state)** en React Native es un mecanismo que nos permite almacenar y actualizar datos dentro de un componente. Cuando el estado cambia, React vuelve a renderizar el componente con los nuevos valores, reflejando los cambios en la interfaz de usuario.

Ejemplo de la vida real:

Piensa en una tarjeta de identificación. Si deseas cambiar el nombre o la foto en tiempo real, necesitas un sistema para almacenar esos datos y actualizar la tarjeta cada vez que los edites. ¡Ahí es donde entra el estado!

Características del Estado:

1. **Es local:** Cada componente tiene su propio estado independiente.
2. **Es mutable:** Podemos actualizarlo usando funciones específicas.
3. **Provoca re-renderizado:** Cuando cambia, React actualiza automáticamente la interfaz.

Uso del Estado con useState

React proporciona el hook useState para manejar el estado en componentes funcionales.

Sintaxis básica de useState:

```
const [nombre, setNombre] = useState('Juan Pérez');
```

Explicación:

1. nombre: Variable de estado (valor actual).
2. setNombre: Función para actualizar el valor de nombre.
3. 'Juan Pérez': Valor inicial del estado.

Cuando queramos cambiar el nombre del estudiante, llamaremos a setNombre(nuevoNombre), y React actualizará el valor y lo reflejará en la interfaz.

---

2. ¿Qué es Tailwind CSS en React Native?

Definición:

Tailwind CSS es un **framework de utilidades** que permite aplicar estilos directamente en los elementos mediante clases predefinidas. En React Native, usamos la biblioteca nativewind para aplicar estilos similares.

Ventajas de Tailwind en React Native:

1. **Rapidez:** Aplicar estilos sin escribir CSS personalizados.
2. **Consistencia:** Las clases garantizan uniformidad en toda la app.
3. **Simplicidad:** Se evitan archivos de estilos complejos.

Ejemplo comparativo:

Sin Tailwind (estilos tradicionales):

```
const styles = StyleSheet.create({
  titulo: {
    fontSize: 24,
    fontWeight: 'bold',
```

```
    color: 'blue',
  },
});
<Text style={styles.titulo}>Hola Mundo</Text>
```

Con Tailwind:

```
<Text className="text-2xl font-bold text-blue-500">Hola Mundo</Text>
```

Explicación:

- text-2xl: Tamaño del texto grande.
- font-bold: Texto en negrita.
- text-blue-500: Color azul.

3. Configuración del Proyecto

Vamos a integrar Tailwind CSS y aplicar el concepto de estado en nuestro carnet estudiantil digital.

Paso 1: Clonar el proyecto existente

```
git clone https://github.com/altair3542/CarnetEstudiantil.git
cd CarnetEstudiantil
npm install
npm start
```

Paso 2: Instalación de Tailwind CSS

```
npm install nativewind tailwindcss
```

Paso 3: Configuración del archivo tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    './App.js',
    './components/**/*.js'
  ],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

Paso 4: Uso de Tailwind en App.js

```
import { View } from 'react-native';
import { TailwindProvider } from 'nativewind';
import CarnetEstudiante from './components/CarnetEstudiante';

export default function App() {
  return (
    <TailwindProvider>
      <View className="flex-1 items-center justify-center bg-gray-
100">
        <CarnetEstudiante />
      </View>
    </TailwindProvider>
  );
```

```
}
```

4. Construcción del Carnet Estudiantil Digital con Estado y Tailwind CSS

Abrimos el archivo CarnetEstudiante.js y lo actualizamos para incluir el estado y permitir la edición de datos.

```
import React, { useState } from 'react';
import { View, Text, TextInput, Image, Button } from 'react-native';

const CarnetEstudiante = () => {
  // Definiendo los estados para los datos del estudiante
  const [nombre, setNombre] = useState('Juan Pérez');
  const [identificacion, setIdentificacion] = useState('202320001');
  const [carrera, setCarrera] = useState('Ingeniería de Software');
  const [estado, setEstado] = useState('Activo');
  const [foto, setFoto] = useState('https://randomuser.me/api/portraits/men/50.jpg');

  return (
    <View className="p-5 bg-white rounded-lg shadow-lg w-80">
      <Image source={{ uri: foto }} className="w-32 h-32 rounded-full mx-auto" />
      <Text className="text-xl font-bold text-center mt-3">{nombre}</Text>
      <Text className="text-gray-600 text-center">ID: {identificacion}</Text>
```

```
      <Text className="text-gray-600 text-center mb-4">Carrera:
{carrera}</Text>


      {/* Campos editables */}
      <TextInput
        className="border border-gray-300 rounded p-2 my-2"
        placeholder="Actualizar Nombre"
        value={nombre}
        onChangeText={(text) => setNombre(text)}
      />
      <TextInput
        className="border border-gray-300 rounded p-2 my-2"
        placeholder="Actualizar Carrera"
        value={carrera}
        onChangeText={(text) => setCarrera(text)}
      />


      <Button title="Cambiar Estado" onPress={() =>
setEstado(estado === 'Activo' ? 'Inactivo' : 'Activo')} />
    </View>
  );
};


export default CarnetEstudiante;
```

5. Actividades Prácticas durante el Live Coding

Cada estudiante debe implementar en vivo:

1. **Agregar nuevos campos al carnet:**
    - Universidad.
    - Correo electrónico.
2. **Modificar estilos personalizados con Tailwind CSS:**
    - Cambiar colores de fondo usando bg-blue-200.
    - Ajustar tamaños de fuente con text-lg font-medium.

```
<TextInput
  className="border border-gray-300 rounded p-2 my-2"
  placeholder="Actualizar Universidad"
  value={universidad}
  onChangeText={(text) => setUniversidad(text)}
/>
```

6. Evaluación de la Sesión

Preguntas para discusión:

1. ¿Cómo funciona useState para actualizar el estado?
2. ¿Qué beneficios aporta Tailwind CSS a nuestro proyecto?
3. ¿Cómo usar Flexbox con Tailwind en React Native?

Entrega de evidencia:

- Capturas de pantalla de la aplicación con los cambios realizados.
- Código subido al repositorio.

---

7. Conclusión de la Sesión

En esta sesión logramos:

1. Aplicar el **estado** en React Native para editar la información del carnet.
2. Integrar **Tailwind CSS** para aplicar estilos de forma eficiente.
3. Crear un diseño responsivo utilizando **Flexbox**.