

«Talento Tech»

React JS

Clase 13



Clase N° 13 | Estilización con Bootstrap o styled-components

Índice:

- Introducción a Bootstrap o styled-components para estilizar componentes.
 - Creación de un diseño básico y responsive.
 - Aplicación de estilos en componentes (botones, formularios, productos).
 - Implementación de una barra de búsqueda en el eCommerce.
-

Objetivos de la Clase:

- Optimizar el diseño y la responsividad de la aplicación utilizando **Bootstrap** y **styled-components**.
- Mejorar la experiencia de usuario con **React Icons** y **React Toastify**.
- Aplicar ajustes visuales y de interacción para hacer la interfaz más atractiva y funcional.
- Implementar mejoras básicas de **SEO** con **React Helmet** para optimizar la visibilidad en buscadores.
- Dejar la aplicación casi lista para su despliegue en un servidor.
- Agregar una barra de búsqueda para que los usuarios puedan filtrar productos en tiempo real.

Introducción a Bootstrap y styled-components



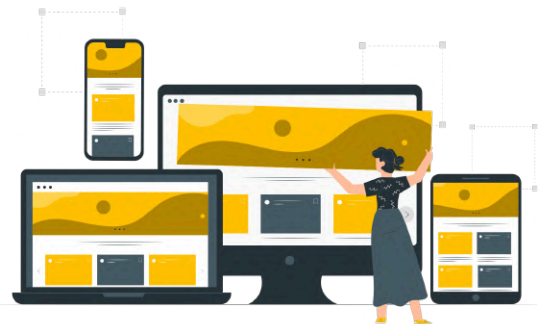
Hasta ahora, hemos utilizado **Bootstrap** y **Bootswatch** como base para la estilización de nuestra aplicación. Gracias a estas herramientas, hemos podido aplicar estilos fácilmente sin necesidad de escribir mucho código CSS personalizado. Sin embargo, en esta etapa del proyecto, es importante optimizar la forma en que aplicamos los estilos para lograr un mejor control sobre la apariencia de nuestra aplicación.

Para esto, revisaremos dos enfoques diferentes:

- Mejorar la utilización de **Bootstrap** para asegurarnos de que los elementos sean completamente responsivos.
- Introducir **styled-components**, una librería que nos permite definir estilos directamente dentro de nuestros componentes de React, evitando la necesidad de archivos CSS externos y haciendo que el código sea más modular y reutilizable.

Ajustes de responsividad con Bootstrap

Uno de los problemas más comunes en una aplicación web es que, aunque se vea bien en pantallas grandes, no siempre se adapta bien a dispositivos móviles. Para asegurarnos de que nuestra aplicación tenga un diseño responsivo, revisaremos el uso del **sistema de grillas** de Bootstrap. Este sistema nos permite dividir el diseño en columnas y hacer que los elementos se ajusten de manera automática dependiendo del tamaño de la pantalla.



Por ejemplo, en la lista de productos, si usamos solo una columna, se verán muy grandes en pantallas grandes y muy pequeños en dispositivos móviles. En su lugar, podemos usar las clases de Bootstrap para que en pantallas pequeñas haya una sola columna (`col-12`), en pantallas medianas haya dos (`col-md-6`) y en pantallas grandes haya tres (`col-lg-4`).

Un ejemplo de código para estructurar correctamente la lista de productos usando el sistema de grillas sería el siguiente:

```
<div className="container">

  <div className="row">

    <div className="col-12 col-md-6 col-lg-4">

      <div className="card">

        <div className="card-body">

          <h5 className="card-title">Producto 1</h5>

          <p className="card-text">$1000</p>

          <button className="btn btn-primary w-100">Comprar</button>

        </div>

      </div>

    </div>

  </div>

</div>
```

Otro punto clave es la **Navbar**. Para hacer que la barra de navegación sea completamente responsiva, podemos usar el componente `navbar-toggler` de Bootstrap, que permite que el menú se colapse en dispositivos móviles y se expanda en pantallas más grandes.

Una vez que revisemos estos aspectos, cada estudiante deberá aplicar estos ajustes en su propia aplicación, revisando cada sección para asegurarse de que todo se vea bien en distintos tamaños de pantalla.

Introducción a styled-components



Styled-components es una librería que nos permite definir estilos dentro de los mismos componentes de React, sin necesidad de usar clases de CSS. Esto ayuda a que nuestro código sea más ordenado y modular, ya que los estilos estarán directamente relacionados con los componentes en los que se aplican.

Para instalar styled-components, debemos ejecutar el siguiente comando:

```
npm install styled-components
```

Una vez instalada, podemos definir estilos de manera sencilla. Por ejemplo, si queremos personalizar un botón de compra, podemos hacerlo de la siguiente manera:

```
import styled from "styled-components";

const BotonCompra = styled.button`
  background-color: #ff5733;
  color: white;
  padding: 10px 15px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  &:hover {
    background-color: #c70039;
  }
`;

function Producto() {
  return <BotonCompra>Comprar</BotonCompra>;
}
```


El beneficio de este enfoque es que cada componente tendrá sus propios estilos sin necesidad de preocuparse por conflictos de clases en CSS. Esto es especialmente útil en proyectos grandes o cuando trabajamos en equipo.

Creación de un diseño básico y responsive

Ahora que entendemos cómo mejorar la responsividad con Bootstrap y cómo estructurar estilos con styled-components, vamos a agregar algunos detalles visuales y funcionales que harán que nuestra aplicación luzca más profesional.

Iconos con React Icons

El uso de iconos mejora la experiencia del usuario, ya que permite representar visualmente las acciones de la aplicación. En lugar de tener solo texto en los botones, podemos agregar pequeños iconos para que sean más intuitivos.

Para instalar React Icons, usamos el siguiente comando:

```
npm install react-icons
```

Luego, podemos importar un icono y usarlo dentro de un botón o cualquier otro elemento.

Un buen ejemplo sería agregar un icono de carrito en la navbar:

```
import { FaShoppingCart } from "react-icons/fa";

<button className="btn btn-warning">

  <FaShoppingCart /> Carrito

</button>
```

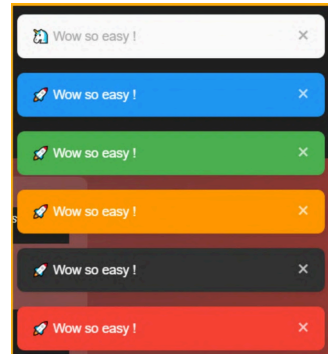
Esta simple mejora hace que la interfaz sea más clara y atractiva para los usuarios.

Notificaciones con React Toastify

Para mejorar la interacción con el usuario, podemos agregar notificaciones cuando se realiza una acción, como agregar un producto al carrito. Para esto, usaremos la librería **React Toastify**, que permite mostrar alertas en pantalla de una manera elegante y no intrusiva.

Para instalarla, ejecutamos:

```
npm install react-toastify
```



Luego, podemos definir una notificación para cuando se agregue un producto al carrito:

```
import { ToastContainer, toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";

const handleAddToCart = () => {
  toast.success("Producto agregado al carrito!");
};

return (
  <>
    <button className="btn btn-success" onClick={handleAddToCart}>
      Agregar al carrito
    </button>
    <ToastContainer />
  </>
);
```

Esta pequeña mejora hace que la aplicación sea más dinámica y amigable para los usuarios.

Aplicación de estilos en componentes (botones, formularios, productos)

Optimización de formularios

```
<form className="p-4 border rounded shadow">

  <div className="mb-3">

    <label className="form-label">Email</label>

    <input type="email" className="form-control" required />

  </div>

  <div className="mb-3">

    <label className="form-label">Contraseña</label>

    <input type="password" className="form-control" required />

  </div>

  <button className="btn btn-primary w-100">Ingresar</button>

</form>
```

Optimización para SEO



El **SEO (Search Engine Optimization)** es un conjunto de técnicas que ayudan a que nuestra aplicación aparezca en los resultados de búsqueda de Google. Aunque React es excelente para construir interfaces dinámicas, por defecto no es muy bueno para SEO porque la mayoría del contenido se genera en el navegador en lugar de en el servidor.

Para mejorar el SEO en React, podemos usar **React Helmet**, una herramienta que nos permite modificar las etiquetas `<title>` y `<meta>` de cada página.

Para instalarlo, ejecutamos:

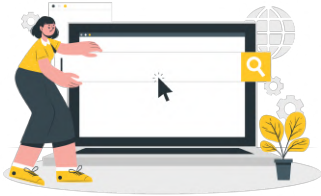
```
npm install react-helmet-async
```

Ejemplo de implementación en una página de productos:

```
import { Helmet } from "react-helmet-async";

function Productos() {
  return (
    <>
      <Helmet>
        <title>Productos | Mi Tienda</title>
        <meta name="description" content="Explora nuestra variedad de
productos." />
      </Helmet>
      <h1>Lista de productos</h1>
    </>
  );
}
```

Implementación de una barra de búsqueda en el eCommerce



Ahora que hemos mejorado la apariencia y la usabilidad de la aplicación, vamos a agregar una funcionalidad clave: **una barra de búsqueda** que permita a los usuarios filtrar productos en tiempo real.

Opciones de implementación

Podemos hacerlo de dos maneras:

1. Búsqueda con `useState` dentro del componente de `productos`.
2. Búsqueda global utilizando Context API.

Método 1: Búsqueda con `useState` en el componente de productos

Vamos a modificar `AllProductos.jsx` para agregar una barra de búsqueda y filtrar los productos en tiempo real.

1 Agregar el estado del término de búsqueda

```
import { useState } from "react";

export default function AllProductos({ productos }) {
  const [busqueda, setBusqueda] = useState(""); // Estado para almacenar el término de búsqueda

  // Filtrar productos según el término de búsqueda
  const productosFiltrados = productos.filter((producto) =>
    producto.name.toLowerCase().includes(busqueda.toLowerCase())
  );

  return (
    <>
      <input
```

```

        type="text"
        placeholder="Buscar productos..."
        className="form-control mb-3"
        value={busqueda}
        onChange={(e) => setBusqueda(e.target.value)}
    />

    {productosFiltrados.length > 0 ? (
        productosFiltrados.map((producto) => (
            <div key={producto.id} className="card product-card mx-2">
                <img src={producto.img} className="card-img-top"
alt={producto.name} />
                <div className="card-body text-center">
                    <h5 className="card-title">{producto.name}</h5>
                    <p className="card-text">{producto.price}</p>
                    <p className="card-text">{producto.description}</p>
                </div>
            </div>
        ))
    ) : (
        <p>No hay productos que coincidan con la búsqueda.</p>
    )}
</>
);
}

```

Con este código, cada vez que el usuario escriba en el input, se actualizará la lista de productos mostrados en pantalla.

Método 2: Búsqueda global con Context API

Si queremos que la búsqueda se pueda usar en diferentes partes de la app, podemos almacenarla en Context API.

1 Crear el Contexto de Búsqueda

context/SearchContext.jsx

```
import { createContext, useState, useContext } from "react";

const SearchContext = createContext();

export function SearchProvider({ children }) {
  const [busqueda, setBusqueda] = useState("");

  return (
    <SearchContext.Provider value={{ busqueda, setBusqueda }}>
      {children}
    </SearchContext.Provider>
  );
}

export function useSearch() {
  return useContext(SearchContext);
}
```

2 Envolver la app con el SearchProvider

App.jsx

```
import { SearchProvider } from "../context/SearchContext";

function App() {
  return (
    <SearchProvider>
      <Router>
        <Navbar />
        <Routes>
          <Route path="/" element={<AllProductos />} />
          { /* Otras rutas */ }
        </Routes>
        <Footer />
      </Router>
    </SearchProvider>
  );
}
```

3) Modificar AllProductos.jsx para usar Context API

```
import { useSearch } from "../context/SearchContext";
export default function AllProductos({ productos }) {
  const { busqueda, setBusqueda } = useSearch();

  const productosFiltrados = productos.filter((producto) =>
    producto.name.toLowerCase().includes(busqueda.toLowerCase())
  );
  return (
    <>
      <input
        type="text"
        placeholder="Buscar productos..."
        className="form-control mb-3"
        value={busqueda}
        onChange={(e) => setBusqueda(e.target.value)}
      />

      {productosFiltrados.length > 0 ? (
        productosFiltrados.map((producto) => (
          <div key={producto.id} className="card product-card mx-2">
            <img src={producto.img} className="card-img-top"
alt={producto.name} />
            <div className="card-body text-center">
              <h5 className="card-title">{producto.name}</h5>
              <p className="card-text">{producto.price}</p>
              <p className="card-text">{producto.description}</p>
            </div>
          </div>
        ))
      ) : (
        <p>No hay productos que coincidan con la búsqueda.</p>
      )}
    </>
  );
}
```

✓ **Ventajas de este método:**

- Se puede acceder al estado de búsqueda desde cualquier parte de la app.
 - Permite compartir el término de búsqueda entre diferentes componentes.
-

Nueva Tarea en Talento Lab



El cliente de **Talento Lab** está impresionado con los avances en el diseño y la experiencia de usuario, pero ha detectado una necesidad clave: **los usuarios tienen dificultades para encontrar rápidamente los productos que buscan.**

Para solucionar esto, además de mejorar la **apariencia y la responsividad**, vamos a **implementar una barra de búsqueda** que permitirá a los usuarios **filtrar productos en tiempo real**.

De este modo, no solo optimizaremos la interfaz con **Bootstrap y styled-components**, sino que también mejoraremos la **usabilidad** con **React Icons, React Toastify** y la nueva función de búsqueda.

Objetivos:

- Optimizar el diseño y la responsividad de la aplicación con Bootstrap y styled-components.
- Implementar una barra de búsqueda para filtrar productos en tiempo real.
- Mejorar la experiencia del usuario con iconos e interacciones visuales mediante React Icons y React Toastify.
- Aplicar ajustes visuales y de interacción para hacer la interfaz más intuitiva.
- Garantizar una experiencia responsiva que se adapte a distintos dispositivos.
- Optimizar el SEO con React Helmet.
- Dejar la aplicación lista para su despliegue con rendimiento y estabilidad mejorados.

Requerimientos:

1 Diseñar una interfaz responsiva con Bootstrap y styled-components

- Usar el **sistema de grillas de Bootstrap** para organizar los productos en diferentes tamaños de pantalla.
- Personalizar la apariencia con **styled-components**, asegurando que los estilos sean reutilizables y escalables.

2 Implementar una barra de búsqueda interactiva

El cliente quiere que los usuarios puedan **buscar productos por nombre en tiempo real**. Para esto, agregaremos un **input de búsqueda** que filtrará la lista de productos a medida que el usuario escriba.

Pasos para implementarla:

- Agregar un **input controlado** en el componente que muestra los productos.
- Filtrar la lista de productos en base al término de búsqueda.
- Mostrar solo los productos que coincidan con la búsqueda.

Mejoras adicionales:

- Agregar un **icono de búsqueda** con React Icons para mejorar la estética.
- Estilizar el input con Bootstrap para que se integre bien en la UI.

3 Integrar iconos con React Icons

- Instalar **React Icons** y agregar iconos en botones y otros elementos interactivos.
- Agregar un **icono de búsqueda en el input** para una mejor experiencia visual.

4 Agregar notificaciones con React Toastify

- Implementar **React Toastify** para mostrar mensajes cuando un usuario agregue productos al carrito.

5 Mejorar la accesibilidad y el SEO con React Helmet

- Usar **React Helmet** para mejorar el SEO y modificar las etiquetas `<title>` y `<meta>`.

6 Preparar la aplicación para su despliegue

- Verificar que **todos los estilos sean responsivos** y que la interfaz se vea bien en distintos dispositivos.
- Asegurar que la barra de búsqueda **funcione correctamente** y no afecte el rendimiento.

- Optimizar la aplicación para **reducir los tiempos de carga** y mejorar la experiencia del usuario.
-

Reflexión final

En esta clase, optimizamos la apariencia y la funcionalidad de nuestra aplicación para que esté lista para su despliegue. En la próxima clase, veremos cómo subirla a un servidor para que cualquiera pueda acceder a ella. ¡Estamos cada vez más cerca del objetivo final! 🚀

Materiales y Recursos Adicionales:

- ★ [Bootstrap](#)
 - ★ [styled-components](#)
 - ★ [React Icons](#)
 - ★ [React Toastify](#)
 - ★ [Framer Motion](#)
-

Preguntas para Reflexionar:

- ¿Cómo puedes asegurar que una aplicación React sea completamente responsiva utilizando Bootstrap y styled-components?
 - En un proyecto con muchos componentes, ¿qué ventajas ofrece usar styled-components frente a usar clases CSS tradicionales o Bootstrap?
 - ¿Qué aspectos de la experiencia del usuario mejoran al agregar iconos con React Icons y notificaciones con React Toastify en una aplicación?
 - ¿Cómo podría la implementación de React Helmet mejorar el SEO de una aplicación React, y qué consideraciones debes tener al utilizarlo?
 - ¿Cómo ayuda la barra de búsqueda a mejorar la experiencia del usuario?
-

Próximos Pasos:

- Diseño mobile-first y adaptabilidad de la aplicación.
- Mejores prácticas de diseño UI/UX.



Buenos Aires
aprende
Agencia de Políticas para el Futuro

BA Buenos
Aires
Ciudad