



Agencia de Habilidades
para el Futuro

<Talento
Tech />

React JS

**Clase 13 | Estilización con Bootstrap o
styled-components**

¡Les damos la bienvenida!



Vamos a comenzar a grabar la clase.

Índice

Implementación del CRUD Completo

- Conectar el formulario de agregar/editar productos con el estado global.
- Validación de formularios y manejo de errores.

Estilización con Bootstrap o styled-components

- **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.**

Diseño Responsivo y UX

- Diseño Mobile-First y Adaptabilidad de la Aplicación.
- Mejores Prácticas de Diseño UI/UX.
- Revisión de la Aplicación para Mejorar la Experiencia de Usuario.
- Implementación de un Paginador en la vista de productos.

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.

Introducción a Bootstrap y styled-components

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.



Bootstrap 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.

- ⓘ Es importante optimizar el uso de Bootstrap para lograr un mejor control sobre la apariencia y responsividad de la aplicación.

Ajustes de responsividad con Bootstrap

1 Sistema de grillas

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.

2 Clases responsivas

Usar col-12, col-md-6, col-lg-4 para diferentes tamaños de pantalla. Por ejemplo, en la lista de productos.

3 Navbar responsiva

Utilizar navbar-toggler para colapsar el menú en dispositivos móviles



Ejemplo de código Bootstrap

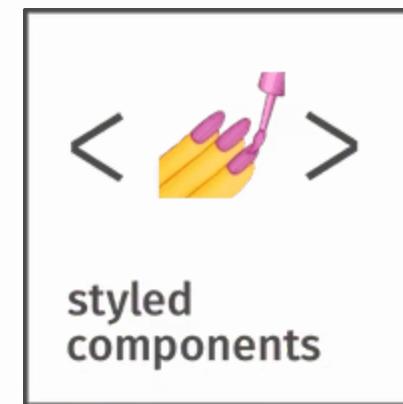
```
<div className="container">
  <div className="row">
    <div className="col-12 col-md-6 col-lg-4">
      <div className="card">
        
        <div className="card-body">
          <h5 className="card-title">Producto 1</h5>
          <p className="card-text">$1000</p>
          <button className="btn btn-primary w-100">Comprar</button>
        </div>
      </div>
    </div>
  </div>
</div>
```

Usa **Bootstrap 5** para un diseño responsive. La tarjeta se adapta con `col-12 col-md-6 col-lg-4`, ocupando distinto espacio según el tamaño de pantalla. La imagen es `img-fluid` para escalar correctamente, y el botón `w-100` ocupa todo el ancho.

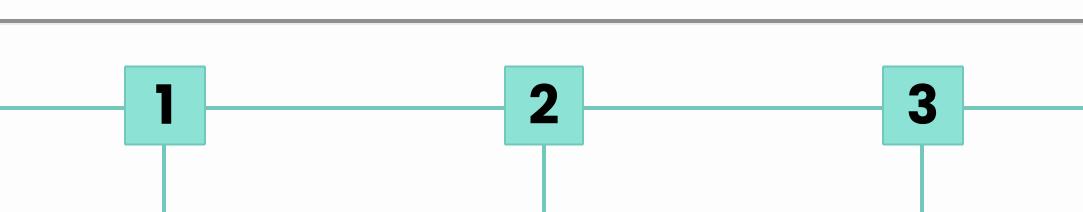
Deberán aplicar estos ajustes en su propia aplicación, revisando cada sección para asegurarse de que todo se vea bien en distintos tamaños de pantalla.

Introducción a styled-components

Styled-components es una librería que permite definir estilos directamente dentro de los componentes de React. Su uso evita la necesidad de archivos CSS externos y hace que el código sea más modular y reutilizable.



Instalación de styled-components



Abrir terminal

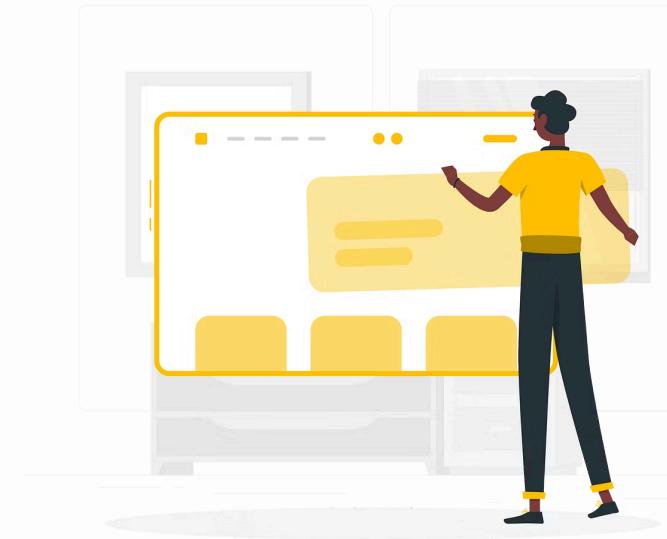
Abre la terminal en tu proyecto de React

Ejecutar comando

npm install styled-components

Importar en componente

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



Ejemplo de styled-components

Beneficios de styled-components

1 Modularidad

Cada componente tiene sus propios estilos

```
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>;  
}
```

2 Evita conflictos

No hay preocupación por conflictos de clases en CSS

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.

3 Código más limpio

Estilos y componentes en un solo lugar



Creación de un diseño básico y responsive

Diseño básico y responsive

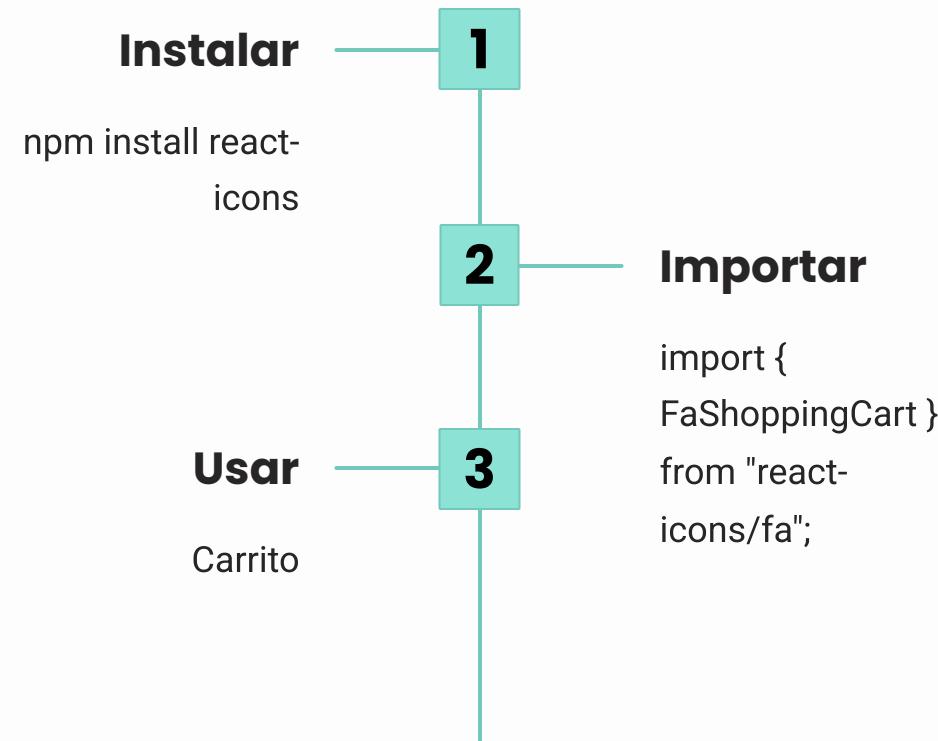
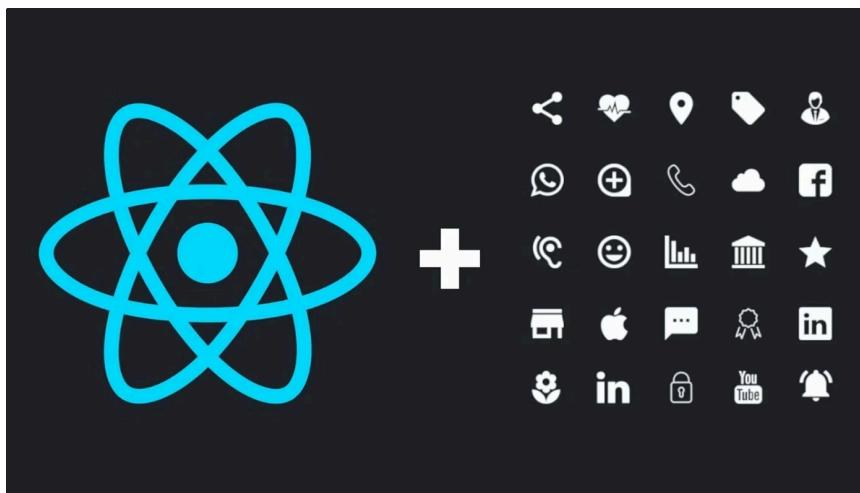


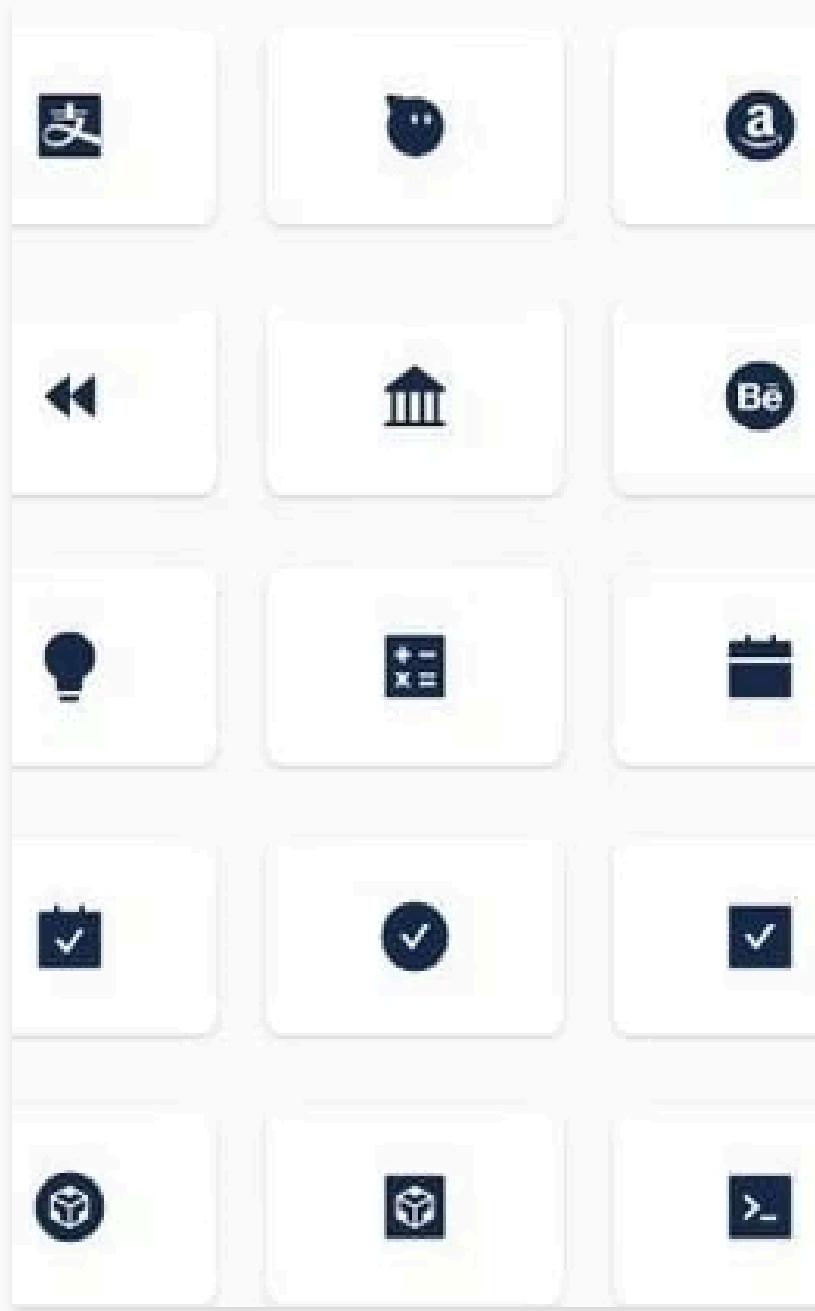
Combinaremos Bootstrap y styled-components para crear un diseño atractivo y adaptable a diferentes dispositivos.

Tras esto, agregaremos detalles visuales y funcionales para mejorar la experiencia del usuario con iconos o notificaciones mas atractivas.

Iconos con React Icons

El uso de iconos mejora la experiencia del usuario, ya que permite representar visualmente las acciones de la aplicación





Ejemplo de uso de React Icons

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

<button className="btn btn-warning">
  <FaShoppingCart /> Carrito
</button>
```

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:

Instalar

```
npm install react-toastify
```

Importar

```
import { ToastContainer, toast } from "react-toastify";
```

Usar

```
toast.success("Producto agregado al carrito!");
```

Ejemplo de uso de React Toastify

```
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

Practicaremos en estos componentes:



1 Botones

Personalizar apariencia y comportamiento

2 Formularios

Mejorar diseño y validación

3 Productos

Crear tarjetas atractivas y responsivas

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>
```

El formulario tiene `p-4 border rounded shadow` para mejorar la apariencia con espaciado, bordes redondeados y sombra. Los

campos usan `form-control` para ocupar todo el ancho, y el botón `w-100` se extiende completamente.





Optimización para SEO

¿Qué es? El SEO (Search Engine Optimization) ayuda 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.

React Helmet nos permite modificar las etiquetas `<title>` y `<meta>` de cada página.

Instalación de React Helmet



Abrir terminal

Abre la terminal en tu proyecto de React

Ejecutar comando

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

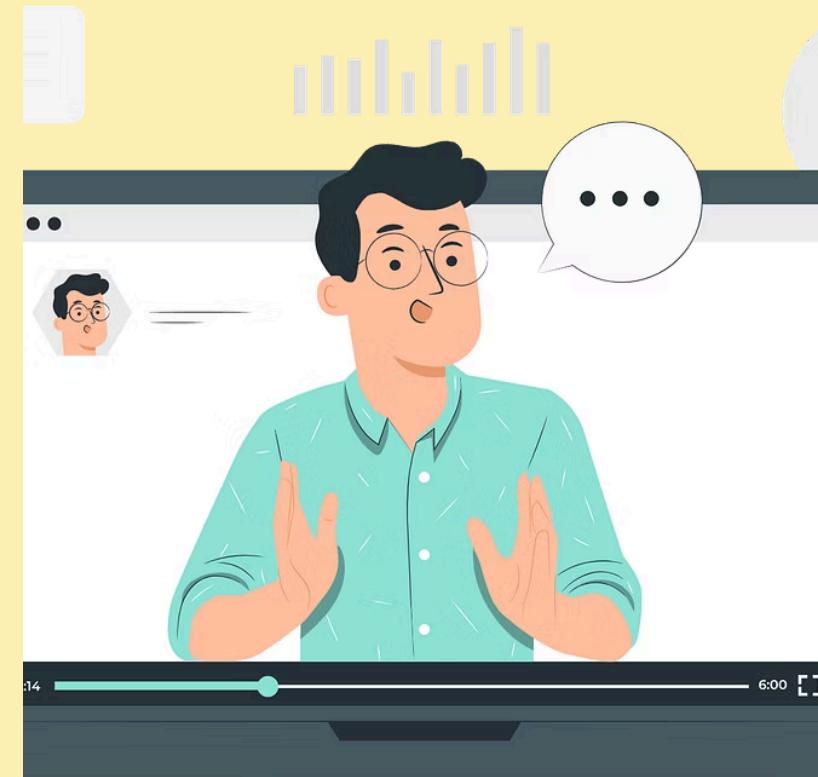
Importar en componente

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

Ejemplo de uso de React Helmet

```
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

Implementación de Barra de Búsqueda

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

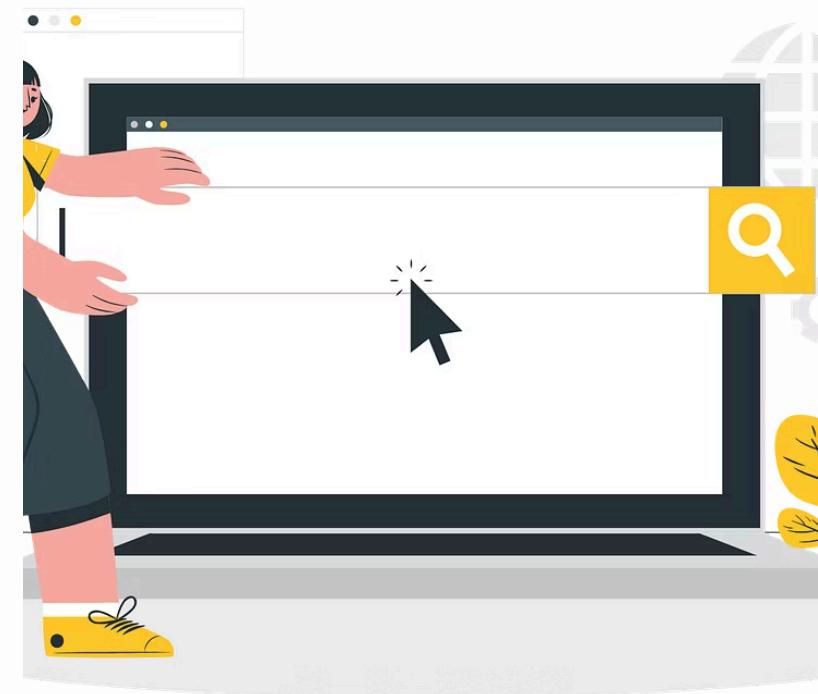
Podemos hacerlo de dos maneras:

1

Búsqueda con useState dentro del componente de productos.

2

Búsqueda global utilizando Context API.



Búsqueda con useState

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.

Agregar el estado del término de búsqueda

```
import { useState } from "react";
export default function AllProductos({ productos }) {
  const [busqueda, setBusqueda] = useState("");
  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)}
      />
      {/* Renderizar productosFiltrados */}
    </>
  );
}
```

Agregar el estado del término de búsqueda

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

Búsqueda global con Context API

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() {
```

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>
  );
}
export default App;
```

```
    return useContext(SearchContext);  
}  
};
```

Método 2: Búsqueda global con Context API

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.



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 adicionales



Bootstrap

Framework CSS para diseño responsivo



styled-components

Librería para estilos en componentes React



React Icons

Colección de iconos para React



React Toastify

Librería para notificaciones en React



Framer

Preguntas para Reflexionar

1 Responsividad

¿Cómo asegurar que una app React sea completamente responsive?

2 styled-components

¿Qué ventajas ofrece frente a CSS tradicional?

3 UX

¿Cómo mejoran React Icons y React Toastify la experiencia del usuario?

4 SEO

¿Cómo mejora React Helmet el SEO de una aplicación React?

5 Barra de búsqueda

¿Cómo ayuda la barra de búsqueda a mejorar la experiencia del usuario?



Próximos Pasos



Diseño mobile-first

Enfoque en adaptabilidad para dispositivos móviles



Mejores prácticas UI/UX

Profundizar en principios de diseño de interfaces



Revisión de UX

Analizar y mejorar la experiencia general del usuario



Paginador de Productos

Implementación de un Paginador en la vista de productos.





Ejercicios Prácticos

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.





Ejercicio Práctico

Obligatorio

Objetivos:



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



Ejercicio Práctico

Obligatorio

Requisitos del Proyecto:

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.

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:

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

Mejoras adicionales:

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



Ejercicio Práctico

Obligatorio

Requisitos del Proyecto:

Integrar iconos con React Icons

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

Mejorar la accesibilidad y el SEO con React Helmet

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

Agregar notificaciones con React Toastify

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

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.

