

«Talento Tech»

React JS

Clase 14



Clase N° 14: Diseño Responsivo y UX

Índice:

- 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:

- Aplicar el enfoque mobile-first para mejorar la adaptabilidad de la aplicación en distintos dispositivos.
- Aplicar las mejores prácticas de diseño UI/UX para crear interfaces atractivas, fáciles de usar y que proporcionen una experiencia positiva al usuario.
- Revisar la aplicación y hacer ajustes que optimicen la interacción del usuario, asegurando que la aplicación sea funcional y agradable tanto en dispositivos móviles como en escritorios.
- Agregar paginación en el componente AllProductos para optimizar la carga y visualización de los productos.

Diseño Mobile-First y Adaptabilidad de la Aplicación

El **diseño mobile-first** es un enfoque de diseño donde se empieza pensando en cómo la interfaz se verá en dispositivos móviles (con pantallas pequeñas) y, a medida que se avanza, se adapta a pantallas más grandes (tabletas, laptops, escritorios). Esto es fundamental porque más usuarios acceden a las aplicaciones desde sus dispositivos móviles, y un diseño optimizado para estos dispositivos ofrece una mejor experiencia.

Beneficios:

- ✓ Optimiza el rendimiento en móviles.
- ✓ Prioriza la usabilidad en pantallas pequeñas.
- ✓ Mejora la accesibilidad y SEO.

Implementación con Bootstrap

Bootstrap nos permite estructurar la interfaz con su sistema de **grillas responsivas**.

Ejemplo: distribución de productos en diferentes tamaños de pantalla:

```
<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>
```

Explicación:

- ✓ En móviles, los productos ocupan **toda la pantalla (col-12)**.
- ✓ En tablets, los productos se organizan en **2 columnas (col-md-6)**.
- ✓ En escritorio, los productos se distribuyen en **3 columnas (col-lg-4)**.

Prueba de adaptabilidad:

- Cambia el tamaño de la pantalla en DevTools (**F12 > Ctrl + Shift + M**).
- Usa **col-sm**, **col-md**, **col-lg**, **col-xl** para ajustar el diseño según el tamaño de la pantalla.

Mejores Prácticas de Diseño UI/UX

¿Por qué es importante la experiencia del usuario (UX)?

UX se refiere a cómo se siente el usuario al interactuar con la aplicación. Un diseño deficiente puede frustrar y alejar a los usuarios, mientras que una interfaz clara y bien organizada mejora la conversión y retención.

Principios clave de UX/UI:

- ✓ Simplicidad: Diseñar interfaces sin elementos innecesarios.
- ✓ Consistencia: Mantener el mismo estilo en botones, tipografías y colores.
- ✓ Accesibilidad: Garantizar que la aplicación sea fácil de usar para todos.
- ✓ Feedback **visual**: Mostrar estados de carga, confirmaciones y errores.

✓ Botones y Formularios Más Accesibles

Los botones deben ser grandes y fáciles de presionar en pantallas táctiles.

Ejemplo:

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

Mejoras en Formularios:

Los formularios deben ser claros, con etiquetas bien definidas y suficiente espaciado.

Ejemplo de formulario optimizado:

```
<form className="p-4 border rounded shadow">
  <div className="mb-3">
    <label className="form-label">Email</label>
    <input type="email" className="form-control" placeholder="Ingresa
tu email" required />
  </div>
  <div className="mb-3">
    <label className="form-label">Contraseña</label>
    <input type="password" className="form-control"
placeholder="*****" required />
  </div>
  <button className="btn btn-success w-100">Ingresar</button>
</form>
```

✅ Espaciado y Legibilidad

Los elementos deben tener suficiente espacio para evitar errores de clic.

Ejemplo:

```
<div className="p-4 my-3 border rounded shadow">
  <h3 className="mb-3">Formulario de Contacto</h3>
  <input type="text" className="form-control mb-3" placeholder="Nombre"
/>
  <input type="email" className="form-control mb-3" placeholder="Correo
Electrónico" />
  <button className="btn btn-success w-100">Enviar</button>
</div>
```

Revisión de la Aplicación para Mejorar la Experiencia de Usuario

Antes de finalizar la aplicación, es importante **probarla en diferentes dispositivos**.


Herramientas para Pruebas Responsivas:

- **Modo Responsive de Chrome:** F12 (DevTools) > Ctrl + Shift + M
- **Simuladores online:**
 - [Responsinator](#)
 - Google Mobile-Friendly Test

Accesibilidad y Etiquetas ARIA

Para mejorar la accesibilidad, usamos etiquetas **ARIA** en botones y enlaces.

Ejemplo:

```
<button className="btn btn-primary" aria-label="Agregar al carrito">  
   Agregar  
</button>
```


Implementación de un Paginador en la vista de productos

Cuando se manejan grandes cantidades de productos en una tienda, mostrarlos todos en una sola página puede afectar la performance y la experiencia del usuario. La paginación permite dividir la lista de productos en páginas más pequeñas, cargando solo una parte a la vez.

Beneficios de la paginación:

- Mejora el rendimiento, cargando solo los productos necesarios.
- Hace que la navegación sea más organizada y fácil de usar.
- Permite a los usuarios explorar los productos de manera más eficiente.

¿Cómo implementarlo en React:?

1. **Dividir los productos en páginas:** Determinar cuántos productos se mostrarán por página.
2. **Crear botones de navegación:** Permitir moverse entre páginas.
3. **Actualizar la vista dinámicamente:** Renderizar solo los productos de la página actual.

Código de Ejemplo: Implementación de Paginador

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

const AllProductos = ({ productos }) => {
  const productosPorPagina = 6; // Cantidad de productos a mostrar por
  página
  const [paginaActual, setPaginaActual] = useState(1);

  // Calcular el índice de los productos a mostrar en la página actual
  const indiceUltimoProducto = paginaActual * productosPorPagina;
  const indicePrimerProducto = indiceUltimoProducto -
  productosPorPagina;
  const productosActuales = productos.slice(indicePrimerProducto,
  indiceUltimoProducto);

  // Cambiar de página
  const totalPaginas = Math.ceil(productos.length /
  productosPorPagina);
```

```

    const cambiarPagina = (numeroPagina) =>
setPaginaActual(numeroPagina);

    return (
      <div className="container">
        <h2 className="my-4">Todos los Productos</h2>
        <div className="row">
          {productosActuales.map((producto) => (
            <div key={producto.id} className="col-12 col-md-6 col-lg-4">
              <div className="card">
                <img src={producto.imagen} className="card-img-top"
alt={producto.nombre} />
                <div className="card-body">
                  <h5 className="card-title">{producto.nombre}</h5>
                  <p className="card-text">${producto.precio}</p>
                  <button className="btn btn-primary w-100">Agregar al
carrito</button>
                </div>
              </div>
            </div>
          ))}
        </div>

        { /* Paginador */ }
        <div className="d-flex justify-content-center my-4">
          {Array.from({ length: totalPaginas }, (_, index) => (
            <button
              key={index + 1}
              className={`btn mx-1 ${paginaActual === index + 1 ?
"btn-primary" : "btn-outline-primary"}`}
              onClick={() => cambiarPagina(index + 1)}
            >
              {index + 1}
            </button>
          ))}
        </div>
      </div>
    );
  };
};

export default AllProductos;

```


El código implementa una paginación en el componente `AllProductos.jsx` para mejorar la experiencia del usuario y optimizar el rendimiento de la aplicación. En lugar de cargar todos los productos a la vez, se muestran en grupos de seis por página. Esto se logra calculando los índices de los productos visibles según la página actual y usando `.slice()` para extraer solo los correspondientes. Además, se genera dinámicamente un conjunto de botones de paginación que permiten navegar entre páginas, resaltando el botón activo con una clase visual.

Con este enfoque, la aplicación se mantiene fluida y organizada, evitando listas demasiado largas que dificulten la navegación. Además, la paginación ayuda a que la interfaz sea más accesible y funcional en diferentes dispositivos, alineándose con las mejores prácticas de UX/UI. 🚀

Nueva Tarea en Talento Lab



El cliente de Talento Lab nos ha solicitado una mejora en la experiencia de navegación dentro de la aplicación. Además de optimizar la interfaz para dispositivos móviles y mejorar la usabilidad general, debemos agregar una funcionalidad de paginación en

la vista de productos.

Objetivos:

- Aplicar el enfoque **mobile-first** para asegurar que la aplicación sea completamente responsiva en todos los dispositivos.
- Implementar mejoras de **UX/UI** utilizando las mejores prácticas de diseño para hacer la interfaz más accesible y fácil de usar.
- Mejorar la interactividad de la aplicación integrando herramientas como **React Icons** y **React Toastify** para optimizar la navegación y proporcionar retroalimentación visual.

- Optimizar el **SEO** de la aplicación utilizando **React Helmet**, asegurando que la aplicación sea fácilmente indexada por los motores de búsqueda.
- Preparar la aplicación para su despliegue, asegurando la compatibilidad con todos los dispositivos y optimizando los tiempos de carga.
- Asegurar que la paginación sea intuitiva y accesible para los usuarios.

Requerimientos:

1. Optimización Responsiva con Bootstrap y Styled-components:

- Utiliza el sistema de grillas de **Bootstrap** para organizar los componentes de manera responsiva, adaptando la distribución de los productos y otros elementos a diferentes tamaños de pantalla (móviles, tablets, escritorios).
- Aplica **styled-components** para crear un diseño personalizado y modular, mejorando la claridad y mantenibilidad del código.

2. Interactividad Mejorada con React Icons y React Toastify:

- Instala **React Icons** y agrega iconos en los botones de acción, como en la barra de navegación (por ejemplo, el icono de carrito en el botón de compra).
- Integra **React Toastify** para mostrar notificaciones visuales al usuario, como mensajes de éxito cuando un producto es agregado al carrito de compras.

3. Mejora de la Accesibilidad y SEO:

- Utiliza etiquetas **ARIA** en botones y enlaces para mejorar la accesibilidad.
- Usa **React Helmet** para modificar las etiquetas `<title>` y `<meta>`, mejorando el SEO de la aplicación y asegurando que sea fácilmente indexada por los motores de búsqueda.

4. Pruebas y Preparación para el Despliegue:

- Realiza pruebas en diferentes dispositivos para garantizar la compatibilidad.
- Revisa que todos los elementos de la interfaz se comporten correctamente en pantallas pequeñas y grandes, asegurando tiempos de carga rápidos y una experiencia de usuario fluida.

5. Paginación en AllProductos:

- Implementa una paginación funcional para mostrar los productos en páginas separadas.
-

Reflexión final

En esta clase, optimizamos la responsividad y la experiencia de usuario, asegurando que la aplicación sea intuitiva y accesible en distintos dispositivos. Aplicamos el enfoque **mobile-first**, mejoramos la usabilidad con prácticas de UX/UI y realizamos una revisión final para detectar y corregir posibles problemas. Estos ajustes no solo mejoran la apariencia, sino que también hacen que la navegación sea más fluida y eficiente. Una aplicación bien diseñada no solo funciona correctamente, sino que también ofrece una experiencia agradable y profesional.

Preguntas para Reflexionar

1. ¿Cómo mejora la experiencia del usuario al aplicar el enfoque mobile-first en el desarrollo de una aplicación?
 2. ¿Qué elementos visuales y funcionales consideras más importantes para garantizar una navegación intuitiva?
 3. ¿Cómo podrías evaluar si la accesibilidad de tu aplicación es adecuada para todos los usuarios?
 4. ¿Qué estrategias podrías implementar para seguir mejorando la experiencia del usuario en futuras versiones de la aplicación?
 5. ¿Cómo podrías mejorar aún más la funcionalidad del paginador para que sea más eficiente y accesible?
-

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?
-

Próximos Pasos:

- Despliegue de la aplicación en plataformas como Vercel o Netlify.
- Pruebas de funcionamiento en el entorno de producción.
- Revisión de los pasos de despliegue y configuración.



Buenos Aires
aprende
Agencia de Políticas para el Futuro

BA Buenos
Aires
Ciudad