

React Js

# Guía de actividades hacia el Proyecto Final

# ¿Qué es?

La **Guía de Actividades** es un espacio que nuclea todas las actividades prácticas que se relacionan directamente con los temas del proyecto final, abordados en el curso y que luego se evaluarán en las **preentregas correspondientes**.

La Guía fue creada para que puedas afianzar, potenciar y poner en práctica los saberes adquiridos en clase. Se desarrolla de manera **asincrónica y no es necesario entregarla. Sin embargo, su resolución es fundamental para la construcción del proyecto final.**

¡A practicar! 😊

**Nota:** te recomendamos que descargues el archivo para que lo puedas editar.

# Semana 4

## Consumiendo API's y Workshop: Hooks, Children y Patrones

# Detalle del producto

## – Parte I

### Descripción de la actividad.

- ✓ Crea tu componente `ItemDetailContainer` con la misma premisa que `ItemListContainer`.

### Recomendaciones

- ✓ Al iniciar utilizando un efecto de montaje, debe llamar a un `async mock`, utilizando lo visto en la clase anterior con `Promise`, que en 2 segundos le devuelva un 1 ítem, y lo guarde en un estado `prop`

# Detalle del producto

## - Parte I

**Ejemplo inicial.**

```
const getItem = () => { /* Esta función debe retornar la promesa que  
resuelva con delay */ }  
function ItemDetailContainer() {  
  // Implementar mock invocando a getItem() y utilizando el resolver then  
  return /* JSX que devuelva un ItemDetail (punto 2) */
```

# Detalle del producto

## – Parte II

### Descripción de la actividad.

- ✓ Crea tu componente ItemDetail.js

### Recomendaciones.

- ✓ ItemDetail.js, que debe mostrar la vista de detalle de un ítem incluyendo su descripción, una foto y el precio

# Detalle del producto

## - Parte II

Preentrega N° 2

Ejemplo inicial.

```
function ItemDetail({ item }) {  
  
  return <>  
    ...  
    // Desarrolla la vista de detalle  
    expandida del producto con su imagen, título,  
    descripción y precio  
    ...  
  </>;  
}
```



# Semana 5

## Routing, navegación y eventos



# Sincronizar Counter

## Descripción de la actividad.

- ✓ Importa el ItemCount.js de la primera pre-entrega del PF en el counter ItemDetail.js, y configura el evento de compra, siguiendo los detalles de manual.

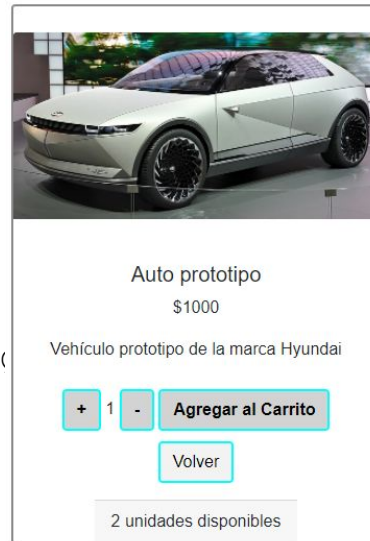
## Recomendaciones

- ✓ Debes lograr separar la responsabilidad del count, del detalle del ítem, y esperar los eventos de agregado emitidos por el ItemCount
- ✓ Cuando ItemCount emita un evento onAdd almacenarás ese valor en un estado interno del ItemDetail para hacer desaparecer el ItemCount
- ✓ El botón de terminar mi compra debe poder navegar a un componente vacío por el momento en la ruta '/cart'.

# Sincronizar Counter

## Ejemplo inicial.

```
function ItemDetail({ item }) {  
  onAdd(quantityToAdd) {  
    // Hemos recibido un evento del ItemCount  
  }  
  return <>  
    ...  
    <ItemCount > // Configura las props de ItemCount. (¿  
habría que mandarle? 🙄)  
  
  </>;  
}
```



# Semana 6

## Context y técnicas de rendering

# Cart Context

## Descripción de la actividad.

- ✓ Implementa React Context para mantener el estado de compra del user

## Recomendaciones

- ✓ Al clicar comprar en ItemDetail se debe guardar en el CartContext el producto y su cantidad en forma de objeto { name, price, quantity, etc. } dentro del array de productos agregados
- ✓ Detalle importante: CartContext debe tener la lógica incorporada de no aceptar duplicados y mantener su consistencia.
- ✓ Métodos recomendados:
- ✓ `addItem(item, quantity)` // agregar cierta cantidad de un ítem al carrito
- ✓ `removeItem(itemId)` // Remover un ítem del cart por usando su id
- ✓ `clear()` // Remover todos los items
- ✓ `isInCart: (id) => true|false`

# Cart Context

Proyecto Final

## Ejemplo inicial.

```
import { createContext, useState } from "react";
export const CartContext = createContext();
const CartContextProvider = ({ children }) => {
  const [cartList, setCartList] = useState([]);
  const addToCart = (item, qty) => { //implementa la funcionalidad para agregar un producto al carrito
  }
  const removeList = () => { //implementa la funcionalidad para dejar el carrito vacío
  }
  const deleteItem = (id) => { //implementa la funcionalidad para borrar un producto del carrito
  }
  return (
    <CartContext.Provider value={{cartList, addToCart, removeList, deleteItem}}>
      { children }
    </CartContext.Provider>
  );
}
export default CartContextProvider;
```

# Cart View

## Descripción de la actividad.

- ✓ Expande tu componente `Cart.js` con el desglose de la compra y actualiza tu `CartWidget.js` para hacerlo reactivo al contexto

## Recomendaciones

- ✓ Crea el componente **`Cart.js`**
- ✓ Debe mostrar el desglose de tu carrito y el precio total.
- ✓ Debe estar agregada la ruta `'cart'` al `BrowserRouter`.
- ✓ Debe mostrar todos los ítems agregados agrupados.
- ✓ Por cada tipo de ítem, incluye un control para eliminar ítems.
- ✓ De no haber ítems muestra un mensaje, de manera condicional, diciendo que no hay ítems y un `react-router Link` o un botón para que pueda volver al Landing (`ItemDetailContainer.js`) para buscar y comprar algo.

# Cart Widget

## Recomendaciones

- ✓ Modifica el componente **CartWidget.js**.
- ✓ Ahora debe consumir el CartContext y mostrar en tiempo real (aparte del ícono) qué cantidad de ítems están agregados (2 camisas y 1 gorro equivaldrían a 3 items).
- ✓ El cart widget no se debe mostrar más si no hay items en el carrito, aplicando la técnica que elijas (dismount, style, etc).
- ✓ Cuando el estado interno de ItemDetail tenga la cantidad de ítems solicitados mostrar en su
- ✓ lugar un botón que diga "Terminar mi compra"

# Cart View

## Ejemplo inicial.

```
const Cart = () => {  
  //accede al contexto con el hook useContext  
  
  return (  
    //recorre el estado global con un map y renderiza  
    //nombre del producto, cantidad de items agregados, precio por item  
    //importe total por producto (para lo cual necesitarás agregar una función  
global  
    //en el contexto  
  );  
  
export default Cart;
```



# Cart View

## Ejemplo inicial.

```
const CartWidget = () => {  
  //accede al contexto con el hook useContext  
  const ctx = useContext(CartContext);  
  return (  
    <Badge badgeContent={ctx.calcItemsQty()} color="secondary">  
      <ShoppingCartOutlined />  
    </Badge>  
  );  
}  
  
export default CartWidget;
```

- 👁️ calcItemsQty() es una función global del contexto que retorna la cantidad de items en el carrito
- 👁️ Badge y ShoppingCartOutlined son componentes de MUI utilizados para el ejemplo. No es necesario que uses MUI. Puedes utilizar cualquier otro framework de CSS o aplicar tus propios estilos CSS puros.

# Semana 7

## Firestore

# Item Collection I

## Descripción de la actividad.

- ✓ Conecta tu nueva ItemCollection de google Firestore a tu ItemListContainer y ItemDetailContainer

## Recomendaciones

- ✓ Conecta tu colección de firestore con el listado de ítems y con el detalle de ítem.
- ✓ Elimina los async mocks (promises) y reemplázalos por los llamados de Firestore.
- ✓ Si navegas a /item/:id, debe ocurrir una consulta de (1) documento.
- ✓ Si navegas al catálogo, debes consultar (N) documentos con un query filtrado, implementando la lógica de categorías y obteniendo el id de categoría del parámetro de react-router :categoryId.

## Ejemplo inicial:

Desde aquí podrás ver un listado de Electrónica

Ver artículos sin stock disponible

Mouse










Mouse verde y blanco

Ver detalle del producto

Stock disponible: 7

Desde aquí podrás ver un listado de todas las categorías

Ver artículos sin stock disponible

<p>Songs &amp; Ballads</p>  <p>Compendio de textos</p> <p>Ver detalle del producto</p> <p>Stock disponible: 17</p>	<p>Bote (tiene opciones)</p>  <p>Bote de madera para remar (tiene opciones)</p> <p>Ver detalle del producto</p> <p>Stock disponible: 4</p>	<p>Mouse</p>  <p>Mouse verde y blanco</p> <p>Ver detalle del producto</p> <p>Stock disponible: 7</p>	<p>Auto prototipo</p>  <p>Vehículo prototipo de la marca Hyundai</p> <p>Ver detalle del producto</p> <p>Stock disponible: 2</p>	<p>Lote de libros</p>  <p>Lotes de libros antiguos</p> <p>Ver detalle del producto</p> <p>Stock disponible: 2</p>	<p>Hermione</p>  <p>Autor: Don Marquis</p> <p>Ver detalle del producto</p> <p>Stock disponible: 6</p>
<p>The Yellow Book (tiene opciones)</p>  <p>En perfectas condiciones (tiene opciones)</p> <p>Ver detalle del producto</p> <p>Stock disponible: 25</p>					