CODER HOUSE

Curso React JS: Proyecto Final

E-commerce project

Desarrollarás una **app de un e-commerce** para poder vender productos de un rubro a elección.

User story/brief:

- Un usuario debe poder ingresar, navegar por los productos e ir a sus detalles.
- Desde el detalle se debe poder ver la descripción, foto y precio e ingresarlo al carrito.
- Una vez que el carrito tenga al menos un producto, se deberá visualizar un listado compacto de la orden con el precio total.
- Al ingresar su nombre, apellido, teléfono e e-mail (ingresándolo dos veces para corroborar que sea correcto), debe activarse el botón de 'realizar compra'.
- Al clickear 'realizar compra' debe guardarse en la base de datos una orden que tenga todos los productos, la fecha y dar feedback del número de orden.

Piezas sugeridas

Te recomendamos incluir:

• Navbar
• Menu
• CartWidget
• ListItem
• ItemList
• ItemDetail
  ◦ ItemQuantitySelector
  ◦ Description
  ◦ AddItemButton
• Checkout
  ◦ Brief (detalle de compra)

Requisitos base

Los requisitos base serán parte de los criterios de evaluación para aprobar el proyecto.

- **Inicio:** Al momento de ingresar a la app en la ruta base **'/'**
    - Visualizar -como mínimo- un set de productos disponibles para la compra.
    - Contar con algún acceso visible a la vista de carrito que debe alojarse en el route **/cart**.
    - Acceder a un menú desplegable que contendrá las categorías. Al clickear en una, debe navegar a la lista de productos de la misma mediante un route **/categories/:categoryId.** Éste invocará la misma vista que el home, pero visualizando sólamente productos de esa categoría.
- **Flow:** Al clickear un ítem del listado debe navegar a la ruta **/item/:id**, donde **id** es el id del item (generado por firebase), y ver la descripción del producto (foto, precio, selector de cantidad). Si se ingresa a **/item/:id** y el producto no existe en firebase, debemos responder un mensaje adecuado que indique algo relacionado a que el producto no existe.
- **Firebase**:
    - Implementar al menos dos colecciones:
        - **items:** catálogo completo
            - Link para foto (puede almacenarse de modo estático en la página en una subruta **/images/:itemid**)
            - Precio unitario
            - Descripción (sólo se ve en detalle)
            - Categoria (id a mano para versión estática, o id de firebase para versión dinámica -opcional-)
        - **orders:** las órdenes generadas, que deben incluir los productos, descripciones y los precios **al momento de la compra.**
            - Las órdenes deben poder tener items surtidos, cada uno con su cantidad. Por ejemplo: remeras x 2 y gorra x 1
            - id, items, fecha, estado (por defecto en 'generada')
        - **categories (solo para versión dinámica -opcional-):**
            - Versión dinámica (-opcional-): Crear una colección de **categories** en firebase para hidratar el menú y usar los id's de éstos para linkearlos a sus ítems. Idealmente, **categories/:id** debería tener una descripción *{id: 'ad43k348j', key: 'calzado', description: 'Calzado'}* para que quede **/categories/calzado** en lugar de **/categories/ad43k348j**

- El **cart** debe ser accesible durante toda la experiencia y tener una indicación de la cantidad de **items** incluidos agregados (ej. si hay un ítem con dos unidades y un ítem con una unidad, debe decir 'tres').
- **Checkout** mínimo:
    - Items con sus cantidades
    - Total de la orden
    - Input para nombre, apellido y teléfono
    - Input para email y lógica de repetir el email 2 veces (a excepción de que realicen el desafío extra de auth, en ese caso no sería necesario)
- Finalizada la orden, debo recibir mi order id con el id del objeto de firebase.
- La navegabilidad debe ocurrir utilizando el router, y no **href's** o **location**.
- Por cada librería pública extra que utilices, deberás incluir en algún archivo el link al proyecto, y una justificación de por qué agrega valor.

Requisitos Extra

Los requisitos extra *pro-coders* no se incluyen en los criterios de evaluación.

Los requisitos extra son funcionalidades opcionales que no se incluyen en los criterios de evaluación, pero si te falta diversión y quieres agregar valor a tu proyecto... ¡bajo la única **condición** de que **lo que incluyas debe funcionar!**

- **auth/login:** Implementar alguno de los servicios de autenticación disponibles de firebase para evitar el flujo de email. Si un usuario está logueado, el checkout debería decir *'comprar como **xxxx@gmail.com**'*, para evitar compras con cuentas indeseadas.
- Adicionalmente al **auth/login**, puedes implementar una **wishlist** para guardar productos para comprar en otro momento. Los productos se deberían poder guardar desde el detalle o desde el listado, y acceder desde el navbar o menú desplegable. La **wishlist** debe tener accesos para agregar esos **ítems** al **cart**.
- **Custom item**: Posibilidad de agregar características seleccionables al producto (ej. talla, color, etc). La customización no debería modificar el precio. Las selecciones serán detalladas en el checkout. Por ejemplo: **1 x camisa (roja) $ 200** y **2 x camisa (verde) $400**.
- **Stock check**: Validar stock al momento de intentar generar la **order**.
- **Categories** dinámicas: crear una colección de firebase para las categorías e hidratar el menú en base a eso.
- **Cart persistente:** Hacer que el **cart** sea **persistente** en alguna api de almacenamiento local en el navegador (local/session storage).

- **Mis órdenes:** Con el orderId que se entrega al final de la compra, el usuario podría buscar su orden y usar el componente que ya utilizaste para el detalle, para mostrar cómo quedó conformada la **order** y el precio, pero no mostrar datos personales de la compra.

Dont's

No es necesario ni recomendado.
• Crear un administrador de stock, dado que puede escaparse del scope y requerir bastante trabajo extra. Podremos gestionar el stock desde la base de firebase. • Implementar categorías anidadas **hogar > equipos de música > parlantes** o con complejización de la solución final.