AC7.1 – aplicación de comida rápida con Redux Toolkit

Mónica Garrigós Sánchez

Sección 1: Redux Toolkit configurado

En esta sección se muestra que Redux Toolkit fue configurado correctamente con *configureStore*, y que el store fue conectado globalmente usando el componente *Provider*.

store.ts M | main.tsx M | menuSlice.ts 1, U | firebaseService.ts U | orderSlice.ts 1, U

sesion7 > AC7.1 - Redux Toolkit > src > redux > store.ts > ...

```ts
import { configureStore } from '@reduxjs/toolkit'
import menuReducer from './menuSlice'
import orderReducer from './orderSlice'

// * Se configura el store global con Redux Toolkit usando configureStore.
// * Aquí se registran los reducers del menú y de los pedidos.

export const store = configureStore({
  reducer: {
    menu: menuReducer,
    orders: orderReducer
  }
})

export type RootState = ReturnType<typeof store.getState>
export type AppDispatch = typeof store.dispatch
```

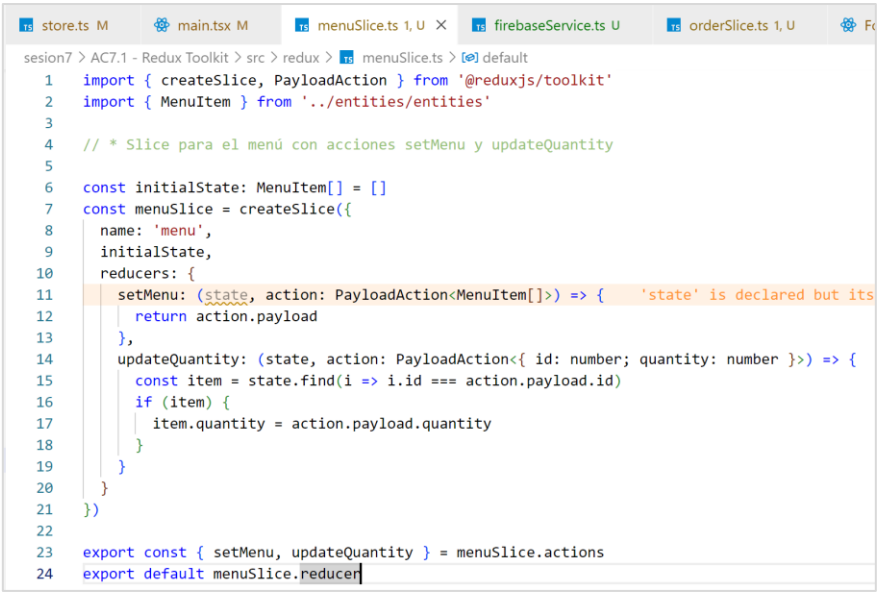
store.ts M | main.tsx M | menuSlice.ts 1, U | firebaseService.ts U | orderSlic

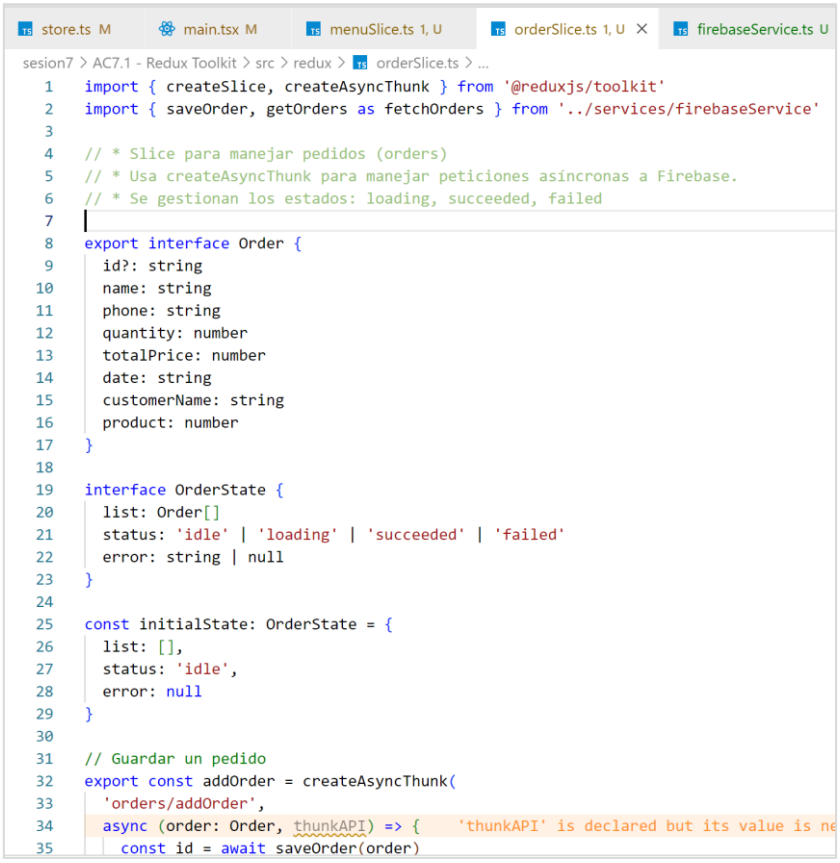
sesion7 > AC7.1 - Redux Toolkit > src > main.tsx

```tsx
import './index.css'
import App from './App.tsx'
import { Provider } from 'react-redux'
import { store } from './redux/store'

// * Se conecta Redux Toolkit a la aplicación con <Provider store={store}>

<Provider store={store}>
  <App />
</Provider>
```

Sección 2: Slices de Redux implementados

Aquí se muestran los slices *menuSlice* y *orderSlice*. *MenuSlice* gestiona los elementos del menú (actualización de cantidades). *OrderSlice* gestiona los pedidos usando *createAsyncThunk*, incluyendo *addOrder* y *getOrders*.





Sección 3: Firebase como backend (persistencia)

Las operaciones CRUD se conectan a Firebase Firestore. Se usa *saveOrder*() para guardar pedidos y *getOrders*() para recuperarlos y mostrarlos.

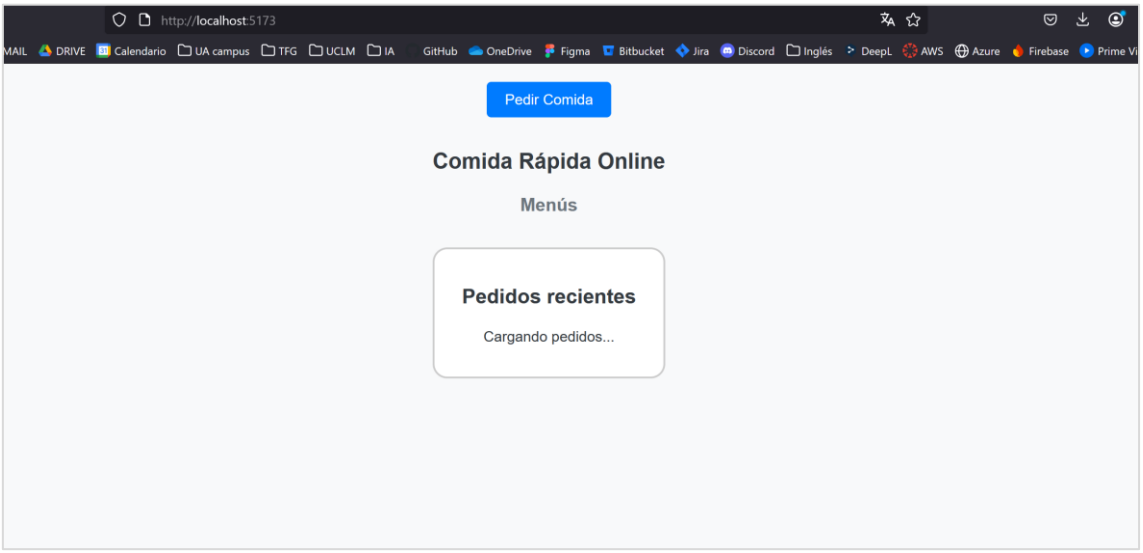
store.ts M | main.tsx M | menuSlice.ts 1, U | orderSlice.ts 1, U | firebaseService.ts U

sesion7 > AC7.1 - Redux Toolkit > src > services > firebaseService.ts > ...

```ts
import { getFirestore, collection, addDoc, getDocs } from "firebase/firestore"
import { app } from "../firebaseConfig"
import { DocumentData } from 'firebase/firestore'
import { Order } from '../redux/orderSlice'

// * Integración de Firebase para persistencia usando Firestore
// * Funciones: saveOrder() y getOrders()

const db = getFirestore(app)
const ordersCollection = collection(db, "orders")

// Guardar un pedido
export async function saveOrder(order: any) {
  try {
    const docRef = await addDoc(ordersCollection, order)
    return docRef.id
  } catch (error) {
    console.error("Error al guardar el pedido:", error)
    throw error
  }
}

Create Jira Issue
// Obtener todos los pedidos
export async function getOrders(): Promise<Order[]> {
  const querySnapshot = await getDocs(ordersCollection)
  return querySnapshot.docs.map(doc => {
    const data = doc.data() as DocumentData
    return {
      id: doc.id,
      name: data.name,
      phone: data.phone,
      quantity: data.quantity,
      totalPrice: data.totalPrice,
      date: data.date,
      customerName: data.customerName,
      product: data.product
    }
  })
}
```

Sección 4: Guardado de pedidos con feedback

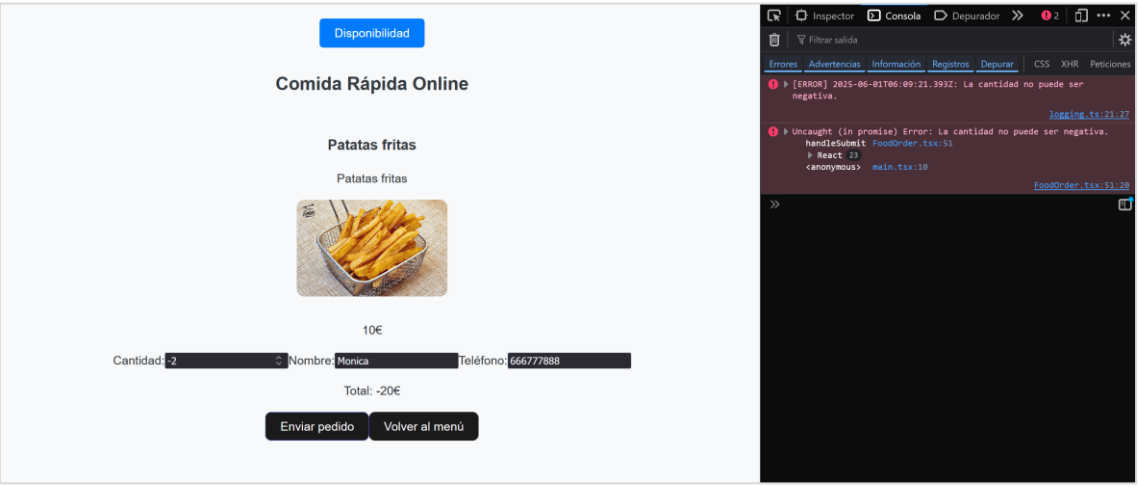
Al enviar un pedido, se muestra retroalimentación visual usando el estado *orderStatus*. El pedido se guarda con *dispatch(addOrder(...))* desde Redux.



Sección 5: Error y Logging

Se implementa un sistema de logging para registrar distintos niveles de eventos (debug, info, warning, error). Además, se usó un componente *ErrorBoundary* para capturar errores de ejecución y evitar que la aplicación se rompa.

En la siguiente captura de pantalla podemos observar que se lanza un error al intentar hacer un pedido con una cantidad negativa, este error se queda registrado en consola con un mensaje personalizado.



Sección 6: Visualización de pedidos

Los pedidos recientes se muestran en la sección de disponibilidad, accediendo a la lista *orders* del estado Redux. Esta lista se carga desde Firebase con *getORders* y se renderiza en el componente *OrderLosts.tsx*.