Proyecto Final — Asignatura: Full Stack 2

Framework: React JS + Bootstrap + React Router DOM
Entrega: Última semana de noviembre
Entregables: Repositorio con README + Video demo (5–7 min) + Proyecto funcional desplegado o ejecutable localmente

---

1) Objetivo general

Desarrollar una **aplicación web tipo tienda en línea** (e-commerce básico) utilizando **React JS**.
La aplicación debe incluir **dos tipos de usuario** (Cliente y Admin) y permitir la gestión completa de productos, usuarios y pagos simulados, conectándose a un **backend REST** mediante funciones asincrónicas con **Fetch o Axios**.

---

2) Requisitos funcionales

Autenticación y sesión

- Inicio y cierre de sesión (persistencia en navegador mediante LocalStorage o Context).
- Redirección automática según tipo de usuario:
    - **Cliente** → Vista Cliente
    - **Admin** → Vista Admin
- Cierre de sesión debe eliminar los datos de sesión.

Vista Admin

Productos

- Crear, editar, buscar, listar y eliminar productos.
- Permitir la carga de **múltiples imágenes** por producto.

Usuarios

- Crear, editar, buscar, listar.
- **Bloquear / Desbloquear** usuarios.

Pagos/Órdenes

- Ver pagos o pedidos **pendientes**.
- **Aceptar pago** → marcar como **enviado**.
- **Rechazar pago** → marcar como **rechazado**.

Vista Cliente

Catálogo y carrito

- Buscar y listar productos.
- Agregar productos al carrito.
- Editar el carrito (cantidad o eliminación).
- **Pagar carrito (simulado)** y **solicitar envío**.

Perfil de usuario

- Ver datos personales.
- Editar datos personales.

> El flujo de pago es **simulado**, pero debe permitir que el Admin gestione su aprobación o rechazo.

3) Requisitos técnicos

- **React JS** (con componentes funcionales y Hooks).
- **React Router DOM** para manejar la navegación entre páginas.
- **Bootstrap** (o React-Bootstrap) para el diseño visual.

Estructura mínima de carpetas:

```
/src
├── /pages
├── /components
├── /services
├── /assets
└── App.js
```

- **Uso obligatorio de funciones asincrónicas** (`async/await`) para consumo de APIs mediante **Fetch** o **Axios**.
- **Reutilización de componentes** (Navbar, Footer, CardProducto, etc.).
- **Responsividad** (adaptable a escritorio y móvil).
- **Manejo de errores y validaciones** de formularios.
- **No se permite** el uso de frameworks de autenticación externos; la lógica debe ser implementada por el estudiante.
- **Ícono y título personalizados** en el proyecto.

---

4) Backend (opciones válidas)

- **Recomendado: Xano** (por su facilidad para crear endpoints REST).
- Alternativas válidas:
    - **Supabase**
    - **Google Firebase (Firestore / Realtime Database)**
    - **Spring Boot Java** desplegado en la nube (**Render, AWS, Vercel o Heroku**).

El backend debe exponer una **API REST** consumible desde React y permitir el almacenamiento de imágenes.
Documenta en el README las rutas y credenciales de prueba.

---

5) UX/UI mínimos

- Diseño limpio y coherente con **Bootstrap**.
- Navegación fluida y coherente con React Router DOM.
- Estados visuales: **Cargando**, **Vacío**, **Error**.
- Mensajería de usuario: **Toasts, alerts o modales** según la acción.
- Control de acceso según rol (ocultar botones o secciones no permitidas).

---

6) Flujo esperado para el video demo

1. **Login como Admin:**
    - Crear y editar productos con múltiples imágenes.
    - Listar y buscar productos.
    - Bloquear/desbloquear un usuario.
    - Revisar pago/orden **pendiente**, luego **aceptar (enviado)** y **rechazar** otro caso.
2. **Logout.**
3. **Login como Cliente:**
    - Explorar catálogo.
    - Agregar productos al carrito.
    - Editar cantidades.
    - “Pagar” carrito (simulado).
    - Ver estado de pedido.
    - Editar datos personales.
4. **Logout final.**

---

7) Entregables

1. **Repositorio en GitHub** con:
    - Código fuente del frontend.
    - Archivo **README** con:
        - Descripción general.
        - Pasos para instalar y ejecutar el proyecto.
        - Detalle del backend utilizado (Xano, Firebase, Supabase o Spring Boot).
        - Usuarios de prueba (admin/cliente) y credenciales dummy.
        - Rutas o endpoints utilizados.
    - (Opcional) Enlace al despliegue si se encuentra publicado.
2. **Video demo (5–7 min)** mostrando el flujo completo del sistema.

---

8) Criterios de evaluación (100 pts)

Categoría	Descripción	Puntaje
Autenticación y redirección por rol	Inicio/cierre de sesión y acceso diferenciado	15 pts
Vista Cliente	Catálogo, carrito editable, pago simulado, solicitud de envío	20 pts
Vista Admin – Productos	CRUD completo con múltiples imágenes	20 pts
Vista Admin – Usuarios	Listar, buscar, editar, bloquear/desbloquear	10 pts
Pagos/Órdenes	Pendientes, aceptar/rechazar, marcar enviado	10 pts
Integración técnica	React Router, Fetch/Axios, asincronía y componentes	10 pts
UI/UX	Diseño Bootstrap, estados, validaciones, mensajes, accesos	10 pts
Calidad de entrega	Estructura del repo, README claro, video demostrativo	5 pts

Bonos (+5 pts):

- Filtrado/orden avanzado.
- Paginación.
- Estado persistente del carrito.
- Deploy funcional en Vercel/Render/Netlify.

---

9) Criterios de aceptación (QA rápido)

- Redirección automática por rol desde el login.
- Logout elimina sesión y redirige al login.
- CRUD de productos y usuarios funcional.
- Subida y visualización de múltiples imágenes.
- Carrito editable y persistente mientras la sesión esté activa.
- Simulación de pago funcional y gestionable por el Admin.
- Sin errores de consola ni rutas rotas.
- Proyecto inicia con `npm start` o despliegue online funcional.

---

10) Plan de trabajo sugerido

1. **Estructurar proyecto React** (`create-react-app` o Vite).
2. **Crear componentes base** (Navbar, Footer, Cards, etc.).
3. **Definir rutas en React Router DOM** y las páginas en `/pages`.
4. **Implementar autenticación y roles.**
5. **Desarrollar flujo Cliente:** catálogo → carrito → pago simulado.
6. **Desarrollar flujo Admin:** productos → usuarios → pagos.
7. **Integrar backend REST** (Xano, Firebase, Supabase o Spring Boot).
8. **Diseñar UI con Bootstrap.**
9. **Probar validaciones y estados de carga.**
10. **Preparar video, README y entrega final.**

---

11) Reglas

- Mantener el código organizado en carpetas `/pages` y `/components`.
- Usar **nombres semánticos** para componentes.
- No dejar rutas o credenciales sensibles en el repositorio.
- Incluir datos de prueba y credenciales dummy.
- El evaluador debe poder **instalar y ejecutar** el proyecto siguiendo el README.

---

12) Checklist final

- Login/logout con control de rol.
- Cliente: catálogo, carrito editable, pago simulado, envío.
- Admin: CRUD productos, usuarios, gestión de pagos.
- Fetch/Axios asincrónico en todas las llamadas a API.
- React Router DOM correctamente implementado.
- Componentes reutilizables (Navbar, Footer, Cards, Modals, etc.).
- Diseño responsive con Bootstrap.
- README completo y video demostrativo.