Trabajo Final Integrador

Desarrollo en React JS
Construcción de una Aplicación Web con Autenticación y Dashboard

Objetivo

Desarrollar una aplicación web utilizando React JS que integre autenticación de usuarios, navegación entre páginas y gestión de datos, aplicando los conceptos de componentes, estados, props, **contexto global** y routing vistos a lo largo del cursado.

Consigna

Este trabajo final integrador consiste en desarrollar una aplicación web completa utilizando **React JS**, **React Router DOM**, **Firebase (Authentication y Firestore/Realtime Database)**, **AuthContext** y **CSS nativo**.

El objetivo principal no es la complejidad del dominio elegido, sino la correcta aplicación de los conceptos fundamentales de React, la organización del proyecto y la integración con servicios externos, consolidando los contenidos trabajados durante el curso.

Cada estudiante deberá construir una aplicación que cuente, como mínimo, con las siguientes páginas:

- **Página de Login**:
Debe permitir el inicio de sesión de usuarios registrados mediante Firebase Authentication. El estado del usuario autenticado deberá gestionarse de forma global utilizando **AuthContext**. Debe incluir validaciones básicas y feedback visual al usuario.

- **Página de Registro**:
Debe permitir la creación de nuevos usuarios utilizando Firebase Authentication. Una vez registrado, el usuario deberá actualizar correctamente el estado global de autenticación a través del **AuthContext**.

● **Dashboard**:
Será la sección principal de la aplicación una vez que el usuario inicia sesión. El acceso a esta página debe estar **protegido mediante rutas privadas**, utilizando la información provista por el **AuthContext**.
Desde aquí se deberá **manipular una entidad a elección del estudiante**, aplicando un CRUD básico (crear, leer, actualizar y eliminar).

Ejemplos de entidades posibles (a modo orientativo, no excluyente):

- Productos
- Cursos
- Tareas
- Turnos
- Usuarios
- Publicaciones

La entidad seleccionada deberá almacenarse en Firebase (Firestore o Realtime Database) y reflejar los cambios en la interfaz mediante la actualización del estado de la aplicación.

● **Página estática informativa**:
Debe ser una página accesible desde la aplicación que explique cómo se desarrolló el proyecto.
Esta sección debe incluir, como mínimo:

- Descripción general del proyecto
- Tecnologías utilizadas
- Estructura del proyecto
- Implementación del AuthContext y manejo de sesión
- Decisiones técnicas relevantes
- Dificultades encontradas y soluciones aplicadas

La navegación entre las distintas páginas debe realizarse exclusivamente mediante **React Router DOM**, utilizando el **AuthContext** para controlar el acceso a rutas privadas y el estado de sesión del usuario en toda la aplicación.

Todos los estilos de la aplicación deberán realizarse utilizando **CSS nativo**, trabajando con archivos .css, sin frameworks externos de estilos.

Formato de presentación / Layout

El trabajo deberá presentarse **exclusivamente mediante un repositorio en GitHub**.

El repositorio debe cumplir con los siguientes requisitos:

● **Commits descriptivos y progresivos**:
El historial de commits debe reflejar claramente el proceso de desarrollo del proyecto, mostrando avances graduales y decisiones técnicas tomadas a lo largo del cursado. No se aceptarán repositorios con uno o pocos commits finales sin evidencia de desarrollo progresivo.

● **Estructura clara del proyecto**:
El repositorio debe presentar una organización adecuada de carpetas y archivos, separando componentes, contextos (AuthContext), estilos y lógica de la aplicación.

● **README.md obligatorio**:
Debe incluir:

- Descripción general del proyecto
- Tecnologías utilizadas
- Instrucciones para instalar y ejecutar el proyecto en entorno local
- Breve explicación de la estructura del proyecto
- Consideraciones generales sobre el desarrollo

● **Link de producción (opcional)**:
De manera opcional, el repositorio puede incluir un enlace a la versión desplegada de la aplicación en un servicio online (Vercel, Netlify o similar).

Criterios de evaluación

• **Funcionalidad básica**: correcto funcionamiento del login, registro, manejo de sesión mediante AuthContext, navegación y operaciones CRUD en el dashboard.
• **Manejo de React**: uso adecuado de componentes, props, useState, useEffect, Context y React Router DOM.
• **Integración con Firebase**: correcta configuración y uso de Authentication y base de datos.
• **Organización del código**: estructura clara de carpetas, separación del AuthContext en un proveedor dedicado y componentes reutilizables.
• **Uso de CSS nativo**: estilos implementados mediante archivos .css, uso de Flexbox o Grid y responsive básico.
• **Documentación**: repositorio en GitHub con un README claro que explique instalación, uso, estructura y manejo del AuthContext.
• **Commits progresivos**: el historial del repositorio debe reflejar el proceso de desarrollo y la evolución del proyecto.
• **Publicación**: la aplicación debe estar desplegada en un servicio online (Vercel, Netlify o similar) y accesible públicamente.

Bibliografía utilizada y sugerida

Banks, A. y Porcello, E. (2016). *Learning React. Functional Web Development with React and Redux*. O'Reilly Media.

Flanagan, D. (2020). *JavaScript. The Definitive Guide*. O'Reilly Media.

MDN Web Docs. (s.f.). *Responsive web design*. Mozilla Corporation.

React. (s.f.-a). *Managing state*. https://react.dev/learn/managing-state

React. (s.f.-b). *useEffect*. https://react.dev/reference/react/useEffect

React. (s.f.-c). *useState*. https://react.dev/reference/react/useState

React. (s.f.-d). *Context*. https://react.dev/reference/react/useContext

Firebase. (s.f.). *Web Authentication & Firestore Documentation*. https://firebase.google.com/docs