Crudzocial

Funcionalidades requeridas

1. Inicio de sesión (login.html)

- Formulario con campo de email.
- Validar si el usuario existe.
- Si no existe, mostrar error (o permitir crear nuevos usuarios).
- Crear por defecto un usuario **admin** con rol especial.

2. Guardian de sesión (Middleware)

- Antes de mostrar cualquier página protegida (imágenes, notas, perfil, logs), validar que haya un usuario logueado.
- Si no hay sesión activa, redirigir a login.html.
- Controlar acceso condicional al contenido según el usuario activo.

3. Página de imágenes (images.html)

- Cada usuario puede agregar imágenes (con campo de texto que recibe una URL).
- Se pueden **eliminar** imágenes propias.
- El **admin** puede ver y eliminar imágenes de todos.

4. Página de notas (notes.html)

- Cada usuario puede crear, editar y eliminar **notas personales**.
- Solo puede modificar las suyas.
- El **admin** puede modificar o eliminar notas de todos.

5. Perfil de usuario (profile.html)

- Mostrar y actualizar el nombre, apellido, email, teléfono, país, cuidad, dirección y código postal.
- Botón para cerrar sesión.
- Se debe mostrar un resumen de actividad o logs propios.

6. Panel de administración (users.html)

- Solo visible para el **admin**.
- Permite:
    - Ver todos los usuarios
    - Ver y editar notas e imágenes de cualquier usuario
    - Ver todos los logs de actividad

7. Logs

- Cada acción importante debe generar un **log**:
    - Inicio de sesión
    - Crear / editar / eliminar notas o imágenes
- Cada log debe incluir:
    - Usuario
    - Fecha/hora
    - Tipo de acción
- Se visualizan con innerHTML.
- Usuarios comunes ven solo sus logs. El admin ve todos.

Persistencia de datos

- Usar **localStorage** para guardar:
  - Usuarios
  - Imágenes
  - Notas
  - Logs
- Usar **sesiónStorage** para guardar:
  - Sesiones activas
- El sistema debe seguir funcionando después de recargar el navegador.

Requisitos técnicos

- Uso de estructuras de datos (array, object)
- Uso de funciones:
  - Declaradas
  - Anónimas
  - Autoejecutables
- Manejo del DOM usando exclusivamente **innerHTML**
- Uso de addEventListener para controlar eventos (No onClick, ni onSubmit)
- Uso claro de condicionales y ciclos

Estilos

- El diseño visual es completamente libre.
- Se permite crear estilos personalizados o usar cualquier librería de estilos, como:
  - **Bootstrap**
  - **Tailwind CSS**
  - **Bulma**
  - **Material UI**

Azure DevOps

Cada equipo debe:

- Crear un **tablero de trabajo (Board)**.
- Registrar **mínimo 10 tareas**, separadas por módulo (login, galería, notas, logs, guardian…).
- Asignar tareas entre los integrantes.
- Documentar el avance de las entregas y observaciones del proceso.

Guía para el README.md

Cada grupo debe crear un README.md en su repositorio GitHub con:

- Nombre del proyecto: **Crudzocial-NombreEquipo**
- Nombre del equipo y sus integrantes
- Descripción del sistema y su objetivo
- Tecnologías utilizadas
- Cómo ejecutar y probar el sistema
- Descripción de las funcionalidades implementadas
- Explicación de cómo usan localStorage, sesionStorage, funciones, como dieron permisos adicionales al admin y logs
- Qué aprendieron como equipo
- Estado actual del proyecto

Sitio web de referencia

Nota: el texto en azul, son funcionalidades adicionales (opcional, por lo cual no habrá referencia para esto)

¡Construyan con lógica, piensen con propósito y programen con intención!

Crudzaso