INSTITUTO SUPERIOR SANTA ROSA DE CALAMUCHITA

Hackaton 2025

Introducción General para los Estudiantes

¡Bienvenidos al Desafío de la Semana: Hackaton Full-Stack JS!

Durante los próximos 7 días, trabajarán en equipos para llevar sus habilidades de desarrollo al siguiente nivel. Cada equipo recibirá por sorteo un desafío único que pondrá a prueba sus conocimientos en React, Node.js, Express y su capacidad para integrar tecnologías modernas.

El objetivo no es solo completar la tarea, sino también colaborar, aprender a resolver problemas bajo presión y crear un producto funcional y bien estructurado.

Además el desafío incluye la jornada final el día viernes en donde tienen que hacer la presentación del resultado y explicación del mismo, sin superar los 30 minutos.

Reglas Generales:

- **Stack tecnológico:** React.js para el Frontend, Node.js y Express.js para el Backend.
- **Entrega:** Al final de la semana, cada grupo deberá presentar un repositorio de GitHub con el código fuente para compartir con el resto de compañeros. Realizar una breve presentación de 30 minutos como máximo.
- **Colaboración:** Usen Git y GitHub para gestionar el trabajo en equipo. ¡La comunicación es clave!

¡Mucha suerte y a programar!

GRUPO 2: Gestor de Documentos

Contexto:
Una tienda de comercio electrónico necesita una forma eficiente de gestionar archivos asociados a sus productos, como manuales de usuario (PDF), fichas técnicas (Excel/Doc) y galerías de imágenes. Actualmente, este proceso es manual y propenso a errores. Se necesita un sistema donde los administradores puedan subir archivos y los clientes puedan verlos y descargarlos desde la página del producto.

Objetivo Principal:
Construir un sistema full-stack que permita la carga, almacenamiento, listado y descarga de múltiples tipos de archivos (PDF, DOCX, XLSX, JPG, PNG) asociados a productos.

Requisitos Funcionales:

1. **Frontend (React):**
    - Crear una zona de "Administrador" protegida (puede ser simulada, sin login real) con un formulario que permita seleccionar uno o varios archivos y asociarlos a un ID de producto (puede ser un listado de productos, en donde cada fila permite hacer clic en el icono de “files” y se abre un popup para la gestión de archivos).
    - Mostrar un “enviando…” durante la subida de los archivos.
    - En la vista pública de un producto, mostrar una lista de los archivos asociados (ej: "Manual de Usuario.pdf", "Garantía.docx").
    - Cada archivo en la lista debe ser un enlace que permita al usuario descargarlo.

2. **Backend (Node.js/Express):**
    - Crear un endpoint para la subida de archivos (ej: POST /api/upload). Debe ser capaz de manejar datos multipart/form-data.
    - Implementar la lógica para guardar los archivos en una carpeta en el servidor (ej: /uploads/productos/{id}).
    - Guardar la información en la tabla productos, para saber con que archivos se cuenta o cuales están cargados o no.
    - Crear un endpoint para obtener la lista de archivos de un producto específico (ej: GET /api/files/:productId).
    - Crear un endpoint para descargar un archivo específico (ej: GET /api/download/:fileName), configurando las cabeceras HTTP adecuadas para forzar la descarga.

Requisitos Técnicos:

- Frontend: React, manejo de formularios de archivos (<input type="file">).
- Backend: Node.js, Express, multer (o una librería similar) para gestionar la subida de archivos, express.static para servir los archivos.

Criterios de Evaluación:

- Funcionalidad completa del ciclo de vida del archivo: subida, almacenamiento, eliminación, listado y descarga.
- Buena gestión de los datos multipart/form-data.
- Interfaz de usuario clara tanto para el administrador como para el cliente.
- Seguridad básica: validar tipos de archivo y tamaños permitidos.

Puntos Extra (Bonus):

- Generar vistas previas (thumbnails) para las imágenes subidas.
- Integrar un servicio de almacenamiento en la nube como Cloudinary o similar en lugar de guardar los archivos en el servidor local.