🚀 EVALUACIÓN - TASKPLANNER

FECHA DE ENTREGA: 21/04/2025

MÉTODO DE ENTREGA: GOOGLE CLASSROOM

APLICACIÓN DE GESTIÓN DE TAREAS Y EVENTOS

El objetivo de esta práctica es desarrollar **TaskPlanner**, una aplicación web que permita a los usuarios gestionar sus tareas y eventos diarios, integrando los conocimientos adquiridos en el módulo.

La aplicación deberá permitir **crear, visualizar, modificar y eliminar tareas**, además de **sincronizar información con una api** y almacenar datos en el navegador.

🔧 REQUERIMIENTOS FUNCIONALES

GESTIÓN DE TAREAS Y EVENTOS

Los usuarios podrán agregar **tareas** con:

Título

Descripción

Fecha de vencimiento

Estado (**pendiente, en progreso, completada**)

Se podrán agregar **eventos** con:

Fecha y hora específicas

Detalles relevantes

Implementar **filtros** para mostrar solo las tareas **pendientes, en progreso o completadas.** (Opcional)

Posibilidad de **editar o eliminar** tareas y eventos de la lista.

MANEJO AVANZADO DEL DOM

Las tareas y eventos deben **mostrarse dinámicamente** en la interfaz.

Permitir interacciones con **botones de edición y eliminación**.

Implementar un **formulario dinámico** para agregar o modificar tareas.

CONSUMO DE APIS

Integrar al menos **dos apis externas** para mejorar la experiencia de usuario:

Api de clima: mostrar la previsión del clima para el día de las tareas programadas.

Openweathermap api: https://openweathermap.org/api

Api de noticias: mostrar noticias recientes sobre productividad o tecnología.

Newsapi: https://newsapi.org/

Api de gestión de tareas: sincronizar la lista de tareas con un servicio externo como trello.

Trello api: https://developer.atlassian.com/cloud/trello/rest/

Api de giphy: permitir la búsqueda y visualización de gifs relacionados con las tareas.

Giphy api: https://developers.giphy.com/docs/

ALMACENAMIENTO EN EL NAVEGADOR

Guardar las tareas y eventos en localStorage para mantener la información entre sesiones.

Recuperar y renderizar (imprimir en el DOM) las tareas almacenadas al cargar la aplicación.

USO DE ARRAYS, OBJETOS Y FECHAS

Arrays: usar un array de objetos para gestionar la lista de tareas y eventos.

Objetos: cada tarea/evento debe ser representado como un objeto con:

id, titulo, descripcion, fecha, estado, etc.

Fechas: utilizar date o la librería **moment.js** para gestionar fechas y organizar eventos cronológicamente.

APLICACIÓN FUNCIONAL Y USABLE

Interfaz intuitiva con diseño atractivo.

Manejo de validaciones (ejemplo: evitar crear tareas sin título, fechas inválidas, etc.).

Responsive: adaptable a dispositivos de escritorio y móviles.

🫶 LIBERTAD CREATIVA EN LA IMPLEMENTACIÓN

Los estudiantes pueden adaptar la práctica y crear aplicaciones innovadoras que cumplan con los requisitos técnicos. Algunas ideas:

E-commerce con gestión de productos y pedidos.

Blog interactivo con comentarios y noticias dinámicas.

Gestor de entrenamientos con planes de ejercicio y nutrición.

Cualquier otra idea que integre almacenamiento, manipule el dom y consuma apis

🤠 AVISO A NAVEGANTES

Además de cumplir con los requisitos funcionales, se evaluará la aplicación según los siguientes criterios de **buenas prácticas de desarrollo**:

Código legible: se espera un código claro, bien definido y estructurado de manera lógica

No redundancia: evitar la repetición innecesaria de código aplicando funciones reutilizables y principios de diseño eficientes.

Modularidad: dividir el código en funciones y módulos reutilizables para mejorar la organización y mantenibilidad.

Uso adecuado de nombres: las variables, funciones y clases deben tener nombres descriptivos y representativos de su propósito SIEMPRE en inglés.

Manejo adecuado de errores: incluir validaciones y capturas de errores para evitar fallos inesperados en la aplicación.

Optimización del rendimiento: evitar cálculos innecesarios, minimizar el uso de eventos costosos y optimizar las consultas al dom.

Estructura de carpetas organizada: separar archivos de lógica, estilos y datos para mejorar la escalabilidad del proyecto.

Uso adecuado de promesas y async/await: manejar correctamente las peticiones asíncronas para evitar bloqueos y mejorar la experiencia de usuario.

Uso de eventos de manera eficiente: evitar el uso excesivo de addEventListener dentro de bucles.

Formato de código consistente: seguir una guía de estilos uniforme, como prettier o eslint, para mejorar la legibilidad.

Comentarios útiles y concisos: documentar el código de manera clara sin añadir información innecesaria.

Uso de control de versiones: mantener un historial de cambios ordenado en git, con commits claros y descriptivos.

Evitar variables globales innecesarias: utilizar const y let en lugar de var y encapsular variables dentro de funciones o módulos.

CRITERIOS EVALUACIÓN DE LOS PROYECTOS MODULARES

Funcionalidad 50% Cumplimiento de los requisitos y operación sin errores.
Calidad del Código 30% Código ordenado, claro y siguiendo buenas prácticas.
Presentación 20% Explicación breve sobre el funcionamiento y decisiones técnicas.