📜 PROMPT FINAL – APP WEB DE PENDIENTES (SOLO FRONTEND)

CONTEXTO GENERAL

Crear una aplicación web de gestión de pendientes, 100% frontend, desarrollada únicamente con HTML, CSS y JavaScript puro, utilizando Bootstrap 5 por CDN para el diseño responsive y DataTables por CDN para la visualización tabular.

La aplicación debe funcionar abriendo directamente index.html, sin necesidad de servidor, build tools ni dependencias instalables.

El objetivo es una app funcional, clara, escalable y bien estructurada, pensada como proyecto de aprendizaje, pero con criterios profesionales.

La navegación entre pantallas se realiza mediante archivos HTML independientes, no mediante routing SPA.

REGLAS GENERALES (OBLIGATORIAS)

1. Entregar el proyecto archivo por archivo, completo, sin omitir nada.
2. No dejar partes incompletas ni usar "...".
3. Mostrar primero la estructura completa de carpetas y archivos.
4. Cada archivo debe entregarse en el siguiente formato:
5. `=== ARCHIVO: ruta/archivo.ext ===codigo completo`
6. No pedir aclaraciones. Si algo no está explícito, usar la solución estándar más razonable, sin inventar funcionalidades.
7. La app no debe contener datos precargados ni inventados.
8. Al iniciar por primera vez:
    - localStorage debe existir pero sin pendientes cargados.
9. No mezclar responsabilidades entre archivos.
10. Todo debe estar separado correctamente según la arquitectura indicada.

CONTROL DE EJECUCIÓN DEL PROMPT (OBLIGATORIO)

Antes de generar cualquier archivo o código, el motor debe:

1. Leer el prompt completo y evaluar si toda la información necesaria está presente.
2. Confirmar explícitamente que entiende los requerimientos y la arquitectura indicada.

3. Preguntar si puede comenzar la generación del proyecto.

Una vez autorizado a comenzar:

4. Generar los archivos UNO POR VEZ.
5. Indicar claramente al inicio de cada entrega:
    - el nombre del archivo
    - la ruta completa del archivo
6. Entregar el contenido completo de ese único archivo.
7. Al finalizar cada archivo, solicitar confirmación explícita para continuar con el siguiente.
8. No generar más de un archivo por respuesta.
9. No generar código adicional hasta recibir confirmación.

Este control es obligatorio para evitar errores de tamaño, cortes de respuesta o problemas al copiar y pegar.

TECNOLOGÍAS PERMITIDAS

- HTML5
- CSS3
- JavaScript puro (ES Modules)
- Bootstrap 5 (por CDN)
- "Bootstrap Icons (por CDN): Debe incluirse el `<link>` correspondiente en el `<head>` de todos los archivos HTML."
- DataTables (por CDN)
- localStorage (NO sessionStorage - ver restricciones)
- Generación e importación de CSV / Excel sin librerías externas instalables

✕ NO PERMITIDO:

- Frameworks (React, Vue, Angular, etc.)
- Bundlers (Webpack, Vite, etc.)
- Backend
- Node.js
- TypeScript
- sessionStorage (usar solo localStorage)

ESTRUCTURA DEL PROYECTO

El proyecto debe seguir una arquitectura modular por capas. A continuación se define la ESTRUCTURA MÍNIMA OBLIGATORIA.

PRINCIPIOS DE ORGANIZACIÓN OBLIGATORIOS:

1. RESPONSABILIDAD ÚNICA:

- Cada archivo debe tener una responsabilidad clara y específica
- Si un archivo hace más de una cosa, debe dividirse

2. LÍMITES DE CÓDIGO:

- **scripts.js**: máximo 50 líneas (solo inicialización)
- **Archivos JS generales**: máximo 200 líneas de código
- **Archivos CSS**: máximo 300 líneas de código
- Si se supera el límite, el archivo DEBE dividirse en archivos más pequeños

3. AGRUPACIÓN LÓGICA:

- Funciones relacionadas deben estar en el mismo archivo
- Ejemplo: todas las operaciones de formateo de fechas en dates.js
- Ejemplo: todas las validaciones de pendientes en validations.domain.js

4. CRITERIO DE DIVISIÓN:

- Dividir cuando un archivo tiene múltiples responsabilidades
- Dividir cuando supera las 200 líneas
- NO dividir solo por dividir (evitar micro-archivos sin sentido)

5. EXTENSIBILIDAD:

- Se pueden crear carpetas y archivos adicionales según sea necesario
- Mantener la estructura por capas (pages, components, domain, services, utils)
- Usar nomenclatura consistente

ESTRUCTURA MÍNIMA OBLIGATORIA:

```
/basedir
├─ index.html
├─ pendientes.html
├─ css/
│   ├─ main.css          (estilos globales, reset, variables CSS)
│   ├─ layout.css        (header, footer, estructura de página)
│   ├─ cards.css         (estilos de tarjetas)
│   ├─ modales.css       (estilos de modales)
│   ├─ tablas.css        (estilos de DataTables)
│   └─ [otros archivos CSS según necesidad, ej: filtros.css,
botones.css]
│
└─ js/
   ├─ scripts.js        (entry point - SOLO inicialización, máx 50
líneas)
   │
   ├─ pages/            (coordinan la lógica de cada página HTML)
   │   ├─ index.page.js      (lógica del dashboard)
   │   ├─ pendientes.page.js (lógica de la tabla)
   │   └─ [otros .page.js según necesidad]
   │
   ├─ components/       (componentes de UI reutilizables)
```

```
    │   ├─ modales/
    │   │   └─ [archivos de modales: verModal.js, crearEditarModal.js,
etc.]
    │   ├─ tablas/
    │   │   └─ [archivos de tablas]
    │   ├─ cards/
    │   │   └─ [archivos de cards]
    │   └─ [otras subcarpetas según componentes necesarios]
    │
    ├─ domain/           (lógica de negocio pura, sin dependencias de
UI)
    │   ├─ pendientes.domain.js  (operaciones CRUD, cambios de estado)
    │   └─ [otros archivos de lógica: validations.domain.js, etc.]
    │
    ├─ services/         (acceso a datos y servicios externos)
    │   ├─ storage.service.js    (interacción con localStorage)
    │   └─ [otros services: export.service.js, import.service.js, etc.]
    │
    └─ utils/            (funciones genéricas reutilizables)
        ├─ dom.js        (manipulación del DOM)
        ├─ dates.js      (formateo y cálculo de fechas)
        ├─ format.js     (formateo de datos para visualización)
        └─ [otras utilidades: validation.js, time.js, etc.]
```

EJEMPLOS DE CUÁNDO CREAR ARCHIVOS ADICIONALES:

✅ CREAR archivo nuevo si:

- Un archivo supera las 200 líneas
- Hay un componente de UI nuevo (ej: filtrosComponent.js)
- Hay una responsabilidad claramente diferente (ej: export.service.js)
- Una categoría de funciones crece mucho (ej: separar time.js de dates.js)

❌ NO crear archivo nuevo si:

- Solo tiene 1-2 funciones simples de 5 líneas
- Está relacionado con funciones que ya existen en otro archivo
- No tiene una responsabilidad claramente diferente

NOMENCLATURA OBLIGATORIA:

- Componentes UI: `nombreComponente.js` (camelCase)
- Services: `nombre.service.js`
- Utils: `nombre.js` (descripción clara)
- Domain: `nombre.domain.js`
- Pages: `nombre.page.js`
- CSS: `nombre.css` (descriptivo)

RESPONSABILIDAD DE LAS CAPAS (NO MEZCLAR):

scripts.js:

- SOLO inicialización

- Detecta qué página HTML está cargada
- Llama a la page correspondiente
- Máximo 50 líneas
- NO contiene lógica de negocio ni lógica de UI

pages/:

- Coordinan eventos de la página
- Llaman a components para renderizar UI
- Llaman a domain para lógica de negocio
- Actualizan la UI según respuestas
- NO contienen lógica de negocio
- NO manipulan directamente localStorage

components/:

- Renderizan elementos de UI
- Manejan eventos de UI (click, submit, etc.)
- Llaman a callbacks proporcionados por pages
- NO contienen lógica de negocio
- NO acceden directamente a localStorage

domain/:

- TODA la lógica de negocio
- Validaciones y reglas
- Transformaciones de datos
- Cálculos (ej: tiempo restante)
- NO acceden al DOM
- NO acceden a localStorage (usan services)

services/:

- Acceso a localStorage
- Export/Import de datos
- Cualquier operación de persistencia
- NO contienen lógica de negocio
- Retornan datos puros

utils/:

- Funciones genéricas reutilizables
- Sin dependencias de la app específica
- Pueden usarse en cualquier proyecto
- Ejemplos: formatear fechas, validar emails, etc.

MODELO DE DATOS DEL PENDIENTE

Cada pendiente debe tener obligatoriamente la siguiente estructura:

```
{
  id: string,                                    // identificador
único
  titulo: string,
  detalle: string,                               // descripción del
pendiente
  proyecto: string,
  prioridad: "alta" | "media" | "baja",
  estado: "pendiente" | "activo" | "finalizado",
  fechaCreacion: "AAAA/MM/DD",                   // solo fecha (para
ordenar)
  fechaActivacion: "DD/MM/AAAA HH:MM" | null,    // fecha + hora (para
cálculo plazo)
  fechaFinalizacion: "DD/MM/AAAA HH:MM" | null,  // fecha + hora
  plazoHoras: number,                            // plazo en horas
(int > 0)
  comentarios: [
    {
      fecha: "DD/MM/AAAA HH:MM",
      comentario: string
    }
  ]
}
```

ACLARACIONES IMPORTANTES:

- El `id` se genera automáticamente al crear el pendiente.
- El `plazoHoras` se expresa en horas (número entero mayor a 0).
- El tiempo restante NO se guarda, se calcula dinámicamente:
  - `(fechaActivacion + plazoHoras) - horaActual`
- Si el plazo se supera, el contador continúa en negativo y se muestra en color rojo.
- Las fechas y horas se toman según la hora local del navegador.
- El tiempo restante debe recalcularse y actualizarse visualmente cada 15 minutos (900000 ms).
- **Al editar un pendiente, la `fechaCreacion` se actualiza a la fecha actual del momento de guardar.**

FORMATOS DE FECHA:

Para almacenamiento (localStorage, export/import):

- `fechaCreacion`: `"AAAA/MM/DD"` (solo fecha, para ordenar)
- `fechaActivacion`: `"AAAA/MM/DD HH:MM"` (fecha + hora)
- `fechaFinalizacion`: `"AAAA/MM/DD HH:MM"` (fecha + hora)
- Comentarios `fecha`: `"AAAA/MM/DD HH:MM"` (fecha + hora)

Para visualización (UI):

- Convertir a `"DD/MM/AAAA"` o `"DD/MM/AAAA HH:MM"` según corresponda

HEADER (COMÚN A TODOS LOS HTML)

El header debe ser idéntico en todos los archivos HTML.

Contenido:

1. **Navegación:**
    - Link a Dashboard (index.html)
    - Link a Pendientes (pendientes.html)
    - El link correspondiente a la página actual debe mostrarse resaltado mediante una clase CSS aplicada dinámicamente.
2. **Botones de acción:**
    - Importar
    - Exportar CSV
    - Exportar Excel

DASHBOARD – index.html

ESTRUCTURA DEL MAIN:

El main debe dividirse en **2 secciones claramente diferenciadas:**

SECCIÓN 1: PENDIENTES ACTIVOS

Ubicación: Primera sección del main

Contenido:

- **Título de sección visible** (ej: "Pendientes Activos" o "En Proceso")
- **Tarjetas** de pendientes con estado "activo"
- **Formato:** Mismo que las tarjetas de pendientes (ver más abajo)
- **Responsive:** 3/2/1 tarjetas por fila (según pantalla)
- **Si no hay pendientes activos:** Mostrar mensaje visible "No hay pendientes activos" (la sección debe permanecer visible)

SECCIÓN 2: PENDIENTES PENDIENTES

Ubicación: Segunda sección del main

Contenido:

1. **Título de sección visible** (ej: "Pendientes")
2. **Controles de filtros y ordenamiento:**
    - **Select Proyecto** (dinámico):
        - Opción por defecto: "Todos los proyectos"
        - Opciones generadas dinámicamente según proyectos existentes en pendientes con estado "pendiente"

        - Se aplica automáticamente al cambiar (onChange)
    - **Select Prioridad** (fijo):
        - Opción por defecto: "Todas las prioridades"
        - Opciones fijas: Alta, Media, Baja
        - Se aplica automáticamente al cambiar (onChange)
    - **Select Ordenar** (fijo):
        - Opciones: "Más antiguo primero" / "Por prioridad"
        - Se aplica automáticamente al cambiar (onChange)
        - Funciona sobre los datos ya filtrados
    - **Botón "Limpiar filtros"**:
        - Resetea todos los selects a su valor por defecto
        - Vuelve a mostrar todos los pendientes en estado "pendiente"
3. **Tarjetas de pendientes** con estado "pendiente"

Comportamiento de filtros:

- Solo filtran pendientes en estado "pendiente"
- Los pendientes "activos" se muestran siempre en su sección especial (arriba)
- Los pendientes "finalizados" NO se muestran en el dashboard
- Los filtros se aplican en tiempo real (onChange)
- El ordenamiento funciona sobre los datos filtrados

Lógica de ordenamiento:

- **Por antigüedad:** `fechaCreacion` ASC (más viejo primero)
- **Por prioridad:** alta → media → baja (en ese orden)

Si no hay pendientes: Mostrar mensaje "No hay pendientes disponibles"

TARJETAS DE PENDIENTES:

Información mostrada:

- Título
- Proyecto
- Prioridad:
    - Alta → badge rojo
    - Media → badge amarillo
    - Baja → badge verde
- Estado
- Fecha de creación (formato: DD/MM/AAAA)
- Fecha de activación (formato: DD/MM/AAAA HH:MM, puede estar vacía si estado=pendiente)
- Tiempo restante (formato: HH:MM, puede mostrar valores negativos en rojo)
- Botón "Ver más"

Diseño de tarjetas:

- Borde visible
- Sombra leve pero notoria

- Cada tarjeta debe tener el mismo tamaño

LAYOUT RESPONSIVE:

IMPORTANTE: Usar Flexbox con justify-content-center (NO Grid)

- **Pantallas grandes (≥992px):** máximo 3 tarjetas por fila, centradas
- **Pantallas medianas (≥768px):** máximo 2 tarjetas por fila, centradas
- **Pantallas pequeñas (<768px):** 1 tarjeta por fila, centrada

Las tarjetas deben centrarse dinámicamente:

```
Ejemplo pantalla grande:
  [tarjeta] [tarjeta] [tarjeta]
     [tarjeta] [tarjeta]          ← centradas si solo hay 2
        [tarjeta]                  ← centrada si solo hay 1
```

Implementación sugerida:

- Usar `d-flex flex-wrap justify-content-center`
- Las tarjetas con ancho fijo o max-width
- NO usar `row` y `col-*` de Bootstrap (porque alinea a la izquierda)

TIEMPO RESTANTE:

Ubicación donde se muestra:

- Tarjetas del dashboard
- Modal "ver" (cuando está abierto)

Formato de visualización:

- `HH:MM` (ej: `05:30` = 5 horas 30 minutos)
- Puede superar 24h (ej: `53:00` = 53 horas)
- Cuando se vence: `-HH:MM` en color rojo (ej: `-01:15`)

Actualización periódica:

- Intervalo: cada **15 minutos** (900000 ms)
- Gestión del setInterval:
  - Se crea un único `setInterval` global al cargar la página
  - Se limpia (`clearInterval`) al cambiar de página
  - Evitar memory leaks y múltiples timers

Cálculo:

```
tiempoRestante = (fechaActivacion + plazoHoras) - horaActual

Si tiempoRestante > 0: mostrar normal
Si tiempoRestante <= 0: mostrar en rojo con signo negativo
```

MODAL "VER PENDIENTE"

Características:

- Existe un solo modal reutilizable.
- El contenido y los botones cambian dinámicamente según:
  - Estado del pendiente
  - Acción que lo abrió

Layout de información:

- 2 columnas para datos generales
- Descripción/detalle y comentarios ocupan todo el ancho

Datos mostrados:

- Título
- Proyecto
- Prioridad (con color)
- Estado
- Fecha de creación (formato: DD/MM/AAAA)
- Fecha de activación (formato: DD/MM/AAAA HH:MM, si existe)
- Fecha de finalización (formato: DD/MM/AAAA HH:MM, si existe)
- Plazo (en horas)
- Tiempo restante (si está activo, formato: HH:MM)
- Detalle (completo)
- Comentarios (lista completa con formato: DD/MM/AAAA HH:MM - comentario)

BOTONES SEGÚN ESTADO:

Estado PENDIENTE:

- Activar
- Agregar comentario
- Editar
- Eliminar

Estado ACTIVO:

- Finalizar
- Agregar comentario

Estado FINALIZADO:

- Sin botones (modal solo lectura)

ACCIONES DE BOTONES:

Activar:

- Cambia estado de "pendiente" a "activo"
- Guarda `fechaActivacion` con fecha y hora actual
- Inicia el contador de tiempo restante
- Cierra el modal y actualiza la UI

Agregar comentario:

- Abre un campo de texto dentro del modal
- Al guardar:
  - Guarda fecha y hora actual
  - Agrega el comentario al array `comentarios`
  - Actualiza la visualización de comentarios en el modal

Editar:

- Abre el modal "Crear/Editar Pendiente" (ver siguiente sección)
- Precarga los datos actuales del pendiente
- Al guardar, actualiza `fechaCreacion` a la fecha actual

Eliminar:

- Solicita confirmación previa (modal de confirmación)
- Si confirma:
  - Elimina el pendiente del localStorage
  - Cierra el modal
  - Actualiza la UI

Finalizar:

- Solicita confirmación previa (modal de confirmación)
- **Requiere comentario obligatorio**
- Si confirma:
  - Cambia estado a "finalizado"
  - Guarda `fechaFinalizacion` con fecha y hora actual
  - Agrega el comentario obligatorio
  - Cierra el modal
  - Actualiza la UI

MODAL "CREAR/EDITAR PENDIENTE"

Características:

- Un solo modal reutilizable
- Se usa tanto para crear como para editar
- Título dinámico: "Crear Pendiente" / "Editar Pendiente"

Ubicación del botón para abrir (modo crear):

- Solo en `pendientes.html`
- Ubicado encima/antes del DataTable
- Texto del botón: "Nuevo Pendiente" o similar

Campos del formulario:

1. **Título** (input text, obligatorio)
2. **Proyecto** (input text, obligatorio)
3. **Detalle** (textarea, obligatorio)
4. **Prioridad** (select: alta/media/baja, obligatorio)
5. **Plazo** (input number, obligatorio)
    - Debe ser un número entero (int)
    - Debe ser mayor a 0
    - Representa horas

Botones:

- **Guardar:** Valida y guarda (crea nuevo o actualiza existente)
- **Cancelar:** Cierra el modal sin guardar

MODO CREAR (nuevo pendiente):

Comportamiento:

- Todos los campos vacíos
- Al guardar, se generan automáticamente:
- `{ id: generado automáticamente (único), titulo: del formulario, detalle: del formulario, proyecto: del formulario, prioridad: del formulario, plazoHoras: del formulario (int > 0), estado: "pendiente", fechaCreacion: "AAAA/MM/DD" (fecha actual), fechaActivacion: null, fechaFinalizacion: null, comentarios: []}`

MODO EDITAR (pendiente existente):

Comportamiento:

- Campos precargados con datos actuales
- Solo se pueden editar: título, proyecto, detalle, prioridad, plazoHoras
- Al guardar:
    - Actualiza los 5 campos editables
    - **Actualiza `fechaCreacion` a la fecha actual**
    - Mantiene: id, estado, fechaActivacion, fechaFinalizacion, comentarios

Dónde se abre (modo editar):

- Desde el botón "Editar" en el modal "ver" (solo si estado = pendiente)
- Desde el botón "Editar" en la tabla de pendientes.html (solo si estado = pendiente)

TABLA – pendientes.html

UBICACIÓN:

- Archivo: `pendientes.html`
- Header: mismo que index.html
- Main: contiene botón "Nuevo Pendiente" + tabla

CONFIGURACIÓN DATATABLE:

Columnas:

1. **Título** (texto)
2. **Proyecto** (texto)
3. **Prioridad** (badge con color)
    - Alta: badge rojo
    - Media: badge amarillo
    - Baja: badge verde
4. **Plazo** (número entero, representa horas)
    - Muestra el valor original (ej: "24")
    - No diferencia entre estados
5. **Estado** (texto: "pendiente", "activo", "finalizado")
6. **Acciones** (botones solo con iconos)

Acciones según estado:

Estado	Ver	Editar	Eliminar	Activar	Finalizar
Pendiente	☑	☑	☑	☑	✕
Activo	☑	✕	✕	✕	☑
Finalizado	☑	✕	✕	✕	✕

Iconos sugeridos:

- Ver:  o icono de ojo
- Editar:  o icono de lápiz
- Eliminar:  o icono de papelera
- Activar:  o icono de play
- Finalizar:  o icono de check

Configuración DataTables:

- Paginación: Seleccionable (10, 25, 50 registros por página)
- Búsqueda: Habilitada (busca en título, proyecto, prioridad)
- Ordenamiento inicial: Por `id` (orden de creación)
- Idioma: Español (si es posible)

- Filtros de columna: DataTables permite ordenar por cualquier columna (funcionalidad nativa)
- "Responsive: La tabla debe estar contenida en un `div` con la clase `.table-responsive` de Bootstrap para evitar desbordes horizontales en móviles, o utilizar la extensión Responsive de DataTables."

Datos mostrados:

- Muestra pendientes en estado: **pendiente**, **activo**, **finalizado**
- NO muestra pendientes eliminados (no existen en localStorage)

Si no hay datos:

- La tabla se muestra vacía
- Con mensaje informativo de DataTables

COMENTARIOS

Agregar comentario:

- Se puede agregar desde el modal "ver"
- Campos:
  - Fecha y hora: se guarda automáticamente (fecha y hora actual)
  - Comentario: campo de texto obligatorio
- Se agrega al array `comentarios` del pendiente

Visualización:

- Formato: `DD/MM/AAAA HH:MM - comentario`
- Mostrar en orden cronológico (más reciente primero o último, a elección)

Comentario obligatorio al finalizar:

- Al finalizar un pendiente, se debe solicitar un comentario obligatorio
- Este comentario se guarda automáticamente con la fecha de finalización

---

IMPORTAR / EXPORTAR

UBICACIÓN:

- Botones en el header (común a todos los HTML)
- Mismo comportamiento en todas las páginas

EXPORTAR:

Formatos disponibles:

- CSV
- Excel (.xls)

Datos exportados:

- Una fila por cada pendiente
- Solo pendientes en estado: **pendiente** y **activo** (NO finalizados)
- Columnas corresponden a los campos del modelo de datos

Formato de fechas en export:

- `fechaCreacion`: `"AAAA/MM/DD"`
- `fechaActivacion`: `"AAAA/MM/DD HH:MM"`
- `fechaFinalizacion`: `"AAAA/MM/DD HH:MM"`
- Comentarios: Array completo como JSON stringificado
  - Ejemplo: `"[{\"fecha\":\"AAAA/MM/DD HH:MM\",\"comentario\":\"texto\"}]"`

Nombre del archivo:

- CSV: `pendientes_DD-MM-AAAA.csv` (fecha actual)
- Excel: `pendientes_DD-MM-AAAA.xls` (fecha actual)

Implementación:

- NO usar el exportador nativo de DataTables
- Implementar export personalizado

IMPORTAR:

Comportamiento:

- Permite cargar un archivo CSV o Excel
- **Agrega los datos importados a los existentes** (no reemplaza)
- Si un registro importado tiene un `id` ya existente, debe generarse un nuevo `id` para evitar colisiones

Formato esperado:

- Mismo formato que el export
- Fechas en formato: `AAAA/MM/DD` o `AAAA/MM/DD HH:MM`
- Comentarios como JSON stringificado

Validación al importar:

Validar:

1. Que existan las columnas requeridas

2. Que los campos obligatorios no estén vacíos (titulo, proyecto, detalle, prioridad, plazoHoras)
3. Que `plazoHoras` sea un número > 0
4. Que `prioridad` sea "alta", "media" o "baja"
5. Que `estado` sea "pendiente", "activo" o "finalizado"
6. Que las fechas tengan formato válido

Comportamiento con errores:

- Si una fila tiene errores → se ignora esa fila
- Mostrar mensaje: "Se importaron X pendientes correctamente. Y filas fueron ignoradas por errores."
- Si TODO el archivo es inválido → mensaje de error y no se importa nada
- "El proceso de lectura del archivo (JSON.parse o lectura de Excel) debe estar envuelto en un bloque try-catch. Si el archivo está corrupto o el formato no es válido, mostrar un `alert` amigable y no romper la ejecución."

STORAGE

localStorage:

- Clave: `"pendientes"`
- Valor: Array de objetos pendiente (JSON stringificado)
- Sin datos precargados al iniciar por primera vez

sessionStorage:

- NO USAR (por restricciones de la plataforma)
- Toda persistencia debe ser en localStorage

RESPONSIVE

Regla general:

- Todo el responsive debe resolverse usando:
  - Bootstrap 5 (clases de utilidad)
  - CSS propio (media queries si es necesario)
- **NO usar JavaScript para modificar layout**

Breakpoints principales:

- Pequeño: < 768px
- Mediano: ≥ 768px y < 992px
- Grande: ≥ 992px

MENSAJES AL USUARIO

Tipos de mensajes:

- Éxito (ej: "Pendiente creado correctamente")
- Error (ej: "Error al guardar el pendiente")
- Confirmación (ej: "¿Está seguro de eliminar este pendiente?")
- Información (ej: "No hay pendientes activos")

Implementación sugerida:

- Usar modales de Bootstrap para confirmaciones
- Usar toasts, alerts o mensajes visuales para notificaciones
- Los mensajes deben ser claros y concisos

Confirmaciones obligatorias:

- Eliminar pendiente
- Finalizar pendiente (requiere comentario obligatorio)

ZONA HORARIA

Las fechas y horas deben tomarse según la hora local del navegador del usuario.

VALIDACIONES

Al crear/editar pendiente:

- Todos los campos obligatorios
- `plazoHoras` debe ser un número entero > 0
- `prioridad` debe ser "alta", "media" o "baja"

Al finalizar pendiente:

- Comentario obligatorio

Al importar:

- Ver sección "Importar / Exportar" para validaciones específicas

---

FIN DEL PROMPT

Este prompt contiene toda la información necesaria para generar la aplicación completa. No inventar funcionalidades adicionales. Seguir estrictamente las especificaciones indicadas.