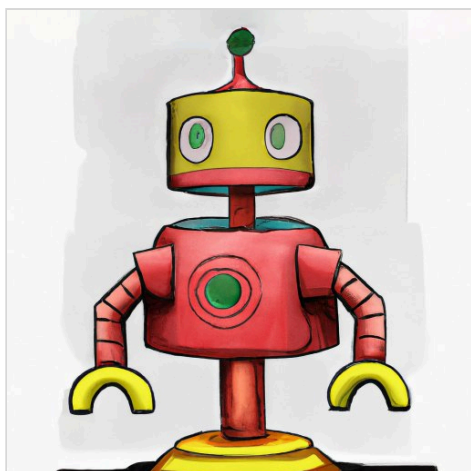


L U M B R E



Reto: crea un sistema de tickets

Materia: IA Aplicada, Vibe Coding y Negocios Digitales

03/10/2026

Antes de empezar

Segundo reto: tickets con estado, prioridad y asignación simple. El CRM te enseñó a guardar; los tickets te enseñan a que los datos cambien de estado sin romperse.

QUÉ APRENDERÁS

- Construir un sistema de tickets con estados y prioridades (abierto → en proceso → cerrado), entrenando el músculo de flujos de estado y vistas filtradas con vibe coding.

El reto: tickets que cambian de estado sin romperse

Mapa conceptual

Reto: crea un sistema de tickets

El reto: tickets que cambian de estado sin r

Contexto · Wikipedia

El modelo: un dato, muchas vistas

El mapa mental interactivo es interactivo en la versión online.



Contexto · Wikipedia

Reto: Reto puede referirse a:

Amikeca Reto, "Red de la amistad" en esperanto.

Competición, desafío.

Los habitantes de Retia.

Wikipedia: Reto <https://es.wikipedia.org/wiki/Reto> Texto consultado y adaptado de Wikipedia (licencia CC BY-SA 4.0).



Imagen libre de Wikimedia Commons (CC BY-SA 4.0, autor: de:Benutzer:Skydou23)

VIDEO

Vibe coding: construir software con IA sin ser programador

Los sistemas de tickets son el gimnasio de los flujos de estado: el mismo dato, cinco vistas que deben coincidir.



Escanea para verlo en Lumbre

<https://www.youtube.com/watch?v=BaveIPkWaL4>

Especificación mínima: (1) crear ticket con título, descripción, prioridad (alta/media/baja) y responsable; (2) **tablero con tres columnas:** abierto, en proceso, cerrado — mover un ticket de columna cambia su estado; (3) **vista filtrada:** 'solo altas abiertas' y 'lo mío' (por responsable).

Fuera del alcance: SLAs y escalaciones, comentarios con notificaciones, adjuntos, roles y permisos, integración con correo. El CRM te enseñó a *guardar*; los tickets te enseñan algo más escurridizo: **que el estado viaje bien**. Un ticket que pasa de 'abierto' a 'cerrado' debe desaparecer de la columna correcta, reflejarse en los filtros y seguir existiendo en el historial. Ahí es donde se rompen el 90% de los sistemas hechos con IA.

El bug clásico de este reto

Mover el ticket cambia la columna visual pero no el estado guardado (o al revés): al recargar, todo vuelve a su lugar. Por eso la regla: valida cada transición RECARGANDO, no arrastrando bonito. La persistencia es la única prueba de que el estado cambió.

El modelo: un dato, muchas vistas

Cuatro piezas, un solo principio: el estado vive en el dato, no en la pantalla.

Pieza	Qué es	Trampa a vigilar
Ticket	Registro: título, descripción, prioridad, responsable, estado, creado	Estado guardado como texto libre ("Abierto" vs "abierto"): usa valores fijos
Tablero 3 columnas	La misma lista filtrada por estado	Mover sin recargar parece funcionar; recarga y miente
Filtros	Consultas: prioridad + estado, responsable actual	Filtrar en la vista y no en los datos deja tickets fantasma
Transición	Acción: cambiar estado (no borrar y crear)	Permitir "cerrar" sin responsable borra la trazabilidad

ACTIVIDAD INTERACTIVA DISPONIBLE EN LUMBRE

1. Cómo atacarlo



2. Ordena los pasos del flujo de trabajo correcto en este reto

Modelo de datos con estados fijos y semilla _____

CRUD de tickets validado con recarga _____

Tablero como tres vistas filtradas por estado _____

Transiciones de estado persistidas _____

Filtros combinados (prioridad + estado + responsable) _____

Romper el flujo y grabar la demo _____

3. Arrastras un ticket a "cerrado", se ve perfecto... pero al recargar la página regresa a "abierto". ¿Cuál es el diagnóstico correcto?

- ☐ El navegador cachea la vista: pide a la IA que agregue un refresh forzado.
- ☐ La prioridad del ticket cambió y hay que sincronizar columnas.
- ☐ La acción de mover solo actualizó la interfaz: falta la llamada que persiste el estado (PATCH) y luego re-pintar desde los datos, no desde el arrastre.
- ☐ Las columnas deben guardarse en localStorage por separado.

4. Para este reto, modelar el estado como texto libre que el usuario escriba es aceptable porque la IA puede interpretarlo igual.

- ☐ Verdadero
- ☐ Falso

5. Antes de promptear, escribe las 6 reglas de estado de TU tablero: qué estados existen, qué transiciones son válidas, qué se exige para cerrar, y qué pasa al recargar a media transición. Compáralas con las de otro compañero.

Si no puedes escribir las reglas sin abrir la app, las reglas viven en la pantalla, no en el modelo.

Siguiente reto: *Crea un analizador de PDF* — de datos que tú capturas a datos que el mundo te manda desordenados: aquí la IA sí es el motor.

Referencia curricular

Materia:	IA Aplicada, Vibe Coding y Negocios Digitales
-----------------	---

Versión interactiva completa (con actividades, videos y autoevaluación):



<https://lumbre.lat/object/iav-lab-r01-oa03>

Material educativo generado desde Lumbre · lumbre.lat