# ANEXO · TAREAS DEL ALUMNADO

## CONSUMIR DATOS: API REST Y FUENTES LOCALES

Nombre: _____ Fecha: _____

## PRIMERA HORA · PROGRAMA CON EL PROFESOR (42 MIN)

En tu `notebox`, con el archivo de Figma 20112026 · Consuming data y la pestaña **Network** de DevTools abierta todo el rato. Códigos de estado: **2xx** ha funcionado, **4xx** has pedido mal (`400` datos malos, `404` no existe), **5xx** el servidor ha fallado.

| OK | HAZ | ESTÁ HECHO CUANDO… |
|---|---|---|
| ☐ | Mueve las cuatro notas a `src/lib/seed.json` (comillas dobles, sin coma final) e impórtalo en el módulo. | La app funciona exactamente igual. |
| ☐ | Crea `src/lib/server/db.js` y el endpoint `src/routes/api/notes/+server.js` con `GET`. | `localhost:5173/api/notes` muestra el JSON. |
| ☐ | Crea `src/routes/api/notes/[id]/+server.js` con `GET` y `DELETE`. | `/api/notes/2` da 200 y `/api/notes/999` da 404 en la pestaña Network. |
| ☐ | Crea `src/routes/+page.js` con `load`, que comprueba `response.ok`. La página recibe `let { data } = $props();`. | En el **otro** navegador ves las mismas notas. |
| ☐ | Crea `src/routes/notes/[id]/+page.js` y cambia la página de la nota para usar `data.note`. | La página de una nota funciona y su Delete borra en el servidor. |
| ☐ | Crea `src/routes/+error.svelte` (frame 3). | `/notes/999` muestra el error dentro del layout, con un enlace para volver. |

Recuerda: `fetch` no falla solo con un 404 o un 500: un «no» también es una respuesta. Comprueba siempre `response.ok`. Y no olvides el `await`.

## SEGUNDA HORA · ENVIAR DATOS CON POST (15 MIN)

| OK | HAZ | ESTÁ HECHO CUANDO… |
|---|---|---|
| ☐ | Añade `POST` a `src/routes/api/notes/+server.js`, que valida el título otra vez. | Desde la consola, `await fetch('/api/notes', { method: 'POST', body: '{"title":"x"}' })` da 400. |
| ☐ | En `/new`, `save(data)` envía un `POST` con método, cabecera JSON y cuerpo; si falla, muestra el mensaje `role="alert"` (frame 4). | Al guardar, la pestaña Network muestra `notes` POST 201 y después `notes` GET 200. |
| ☐ | Borra `src/lib/notes.svelte.js`. Si hiciste el ejercicio `/tags/[tag]`, esa página aún lo importa y se rompería: antes dale su propio `+page.js` (pide todas las notas, filtra en `load` y devuelve `{ tag, notes }`) y lee `data.tag` y `data.notes`. | La app sigue funcionando. |

Pregunta: ya validamos el título en el formulario el viernes pasado. ¿Por qué lo validamos otra vez en el servidor?

## EJERCICIO · BORRAR DESDE LA LISTA (30 MIN)

Individual, a partir del frame **5 · Exercise** de Figma.

| OK | HAZ | ESTÁ HECHO CUANDO… |
|---|---|---|
| ☐ | Bajo cada tarjeta, un botón de texto «Delete» + el título de la nota (13 px, color danger, sin borde): **hermano** del enlace de la tarjeta, no dentro. | Al pulsar Delete no se abre la nota. |
| ☐ | Al hacer clic envía `DELETE /api/notes/{id}`; si la respuesta es ok, vuelve a ejecutar `load` con `await invalidate('/api/notes')` (de `$app/navigation`). | La tarjeta desaparece sin recargar la página. |
| ☐ | Muestra un estado vacío cuando no hay notas: «No notes here. Write the first one.» con un enlace a `/new`. | Al borrar todas, sale el mensaje. |
| ☐ | Mira la pestaña Network. | Un DELETE con estado 204 y después un GET con estado 200. |

☐ Si acabas pronto: `/tags/[tag]` con su propio `+page.js` (pide todas las notas, filtra en `load` y devuelve `{ tag, notes }`). Después PATCH: añade `PATCH` al endpoint `[id]` para cambiar `pinned`, y un botón Pin en la página de la nota.

Si algo falla:

- **Al pulsar el botón también se abre la nota.** Está dentro del `<a>`. Las cosas interactivas no se anidan.
- **Quitas la tarjeta de `data.notes` a mano** (`splice`). Las notas ahora son del servidor: la página vuelve a preguntar, no adivina.
- **La lista se refresca antes de borrar.** Falta un `await` delante de `invalidate` o de `fetch`.
- **Al pulsar Eliminar sale un error en la consola.** Has escrito `onclick={remove(note.id)}` sin flecha: es una llamada, no una función. Si faltan notas, reinicia `npm run dev` para recuperar las iniciales.
- **Funciona en / pero no en otras páginas.** Has escrito `fetch('api/notes/' + id)` sin la barra inicial.
- **Usas `location.reload()`.** Funciona, pero tira todo el estado de la app para redibujar cuatro tarjetas. Usa `invalidate`.

Al reiniciar el servidor, las notas borradas vuelven. ¿Por qué? ¿Qué usaría un servidor de verdad en lugar de un array?

## ENTREGA

Commit **«Session 8: data from the API»** y push. En Classroom: el enlace y una **captura de la pestaña Network** con tu DELETE con estado 204 seguido del GET.

## PARA CASA

- Termina el ejercicio.
- En **tu propia app**: pon tus datos principales en `src/lib/<algo>.json`, sírvelos desde un endpoint `GET` y muéstralos con `load`. Añade `+error.svelte`. Si tu app crea cosas, añade el `POST`. Haz commit. Es la última pieza antes del taller del lunes.
- Pregunta de transferencia: tu app debe mostrar el tiempo de hoy desde una API pública que a veces tarda cuatro segundos y a veces responde 503. Escribe los estados que necesita tu tarjeta del tiempo y qué ve el usuario en cada uno.