ANEXO · TAREAS DEL ALUMNADO

COMANDOS DE VOZ: WEB SPEECH API EN NOTEBOX

Nombre: _____ Fecha: _____

TAREA 1 · ESCRIBE EL VOCABULARIO (12 MIN)

Individual. Abre el enlace de Figma 11012027 · Voice commands, página **Students**. Lee el tablero **A** (el vocabulario de NoteBox): cada fila dice lo que el usuario **dice**, lo que la app **hace** y lo que la app **responde**.

¿Por qué Delete responde «Say undo to bring it back»?

Haz **Duplicate to your drafts** y rellena el tablero **B** con al menos cuatro comandos para tu app.

OK	HAZ	ESTÁ HECHO CUANDO…
☐	Cada comando tiene de 2 a 4 palabras.	Nada de palabras sueltas cortas: «next note», no «n».
☐	Ningún par de comandos suena parecido.	Nada como «delete» / «repeat».
☐	Como mucho un comando destructivo, con su fila de **Undo**.	Si hay un borrado, hay una forma de deshacerlo.
☐	Cada comando ya existe como botón en tu app.	La voz es una segunda puerta, no la única.

TAREA 2 · EL OÍDO (17 MIN)

Escribe el código a la vez que se proyecta. Primero, los tipos (los proyectos en JavaScript se saltan este paso):

```
npm install -D @types/dom-speech-recognition
```

```
// src/app.d.ts, primera línea
/// <reference types="dom-speech-recognition" />
```

OK	HAZ	ESTÁ HECHO CUANDO…
☐	Crea `src/lib/nui/speech.ts` con la función `recognitionClass()`.	Devuelve `SpeechRecognition`, `webkitSpeechRecognition` o `undefined`.
☐	Crea `VoiceButton.svelte`: un botón **Speak** y un párrafo.	Al pulsar, el navegador pide el micrófono.
☐	Fija `lang`, `continuous = false` e `interimResults = true`.	**Ves tus propias palabras en pantalla** (antes del descanso).

¿Para qué sirve la comprobación `typeof window` en `recognitionClass()`?

Sin micrófono: siéntate con un compañero solo en esta tarea. Desde la tarea 3 puedes trabajar solo con el campo de texto.

TAREA 3 · DE PALABRAS A UN COMANDO (17 MIN)

OK	HAZ	ESTÁ HECHO CUANDO…
☐	Crea `src/lib/nui/commands.ts` con `parseCommand(transcript)`: texto entra, comando sale, sin micrófono dentro.	Funciona con las dos primeras filas de **tu** tablero B.
☐	Normaliza el texto antes de comparar (minúsculas, sin puntuación).	«New note, shopping list.» se reconoce igual que «new note shopping list».
☐	Si no es un comando, devuelve `null`.	«do not delete note» no hace nada.
☐	Una función `handle(transcript)` con dos entradas: el resultado final del oído y un pequeño formulario.	Escribir un comando en **Type a command** y pulsar **Run** lo ejecuta.

El campo de texto **se queda**: es la entrada para quien no puede hablar, está en una biblioteca o usa un navegador sin reconocimiento.

TAREA 4 · SEIS ESTADOS (16 MIN)

Tableros **C** y **D** de Figma. Puedes adaptar el `VoiceButton.svelte` de Classroom (viene de la app de referencia, con SvelteKit 3 e imports `#lib` como un proyecto de `sv create`), pero el vocabulario y los comandos son tuyos. Marca cada estado cuando funcione:

OK	ESTADO	LO QUE LEE EL USUARIO
☐	**Idle**	La pista con un comando de ejemplo.
☐	**Listening**	Botón rojo y su palabra cambia a «Listening»; palabras provisionales en cursiva.
☐	**Recognised**	La respuesta de tu tabla de vocabulario.
☐	**Not a command**	I heard «…». That is not a command. Try «…»
☐	**Problem**	Un mensaje por código de error, cada uno con qué hacer (tablero D).
☐	**Unsupported**	Botón desactivado, no oculto, y el aviso de escribir el comando.

El mensaje va dentro de `<div role="status" aria-live="polite">`.

Si algo falla:

- **El estado solo se ve por el color.** El botón rojo también debe cambiar su palabra.
- **Error `window is not defined`.** Compruebas `unsupported` arriba del script; va dentro de `$effect`.
- **El botón se queda en «Listening» tras un silencio.** Usa el evento `end`: si sigue en `listening`, vuelve a idle.
- **Pulsar dos veces da `InvalidStateError`.** Desactiva el botón mientras escucha.

TAREA 5 · QUE LO DIGA EN VOZ ALTA (8 MIN)

OK	HAZ	ESTÁ HECHO CUANDO…
☐	Añade la función `speak(text, lang)` a `speech.ts`.	Llama a `speechSynthesis.cancel()` antes de hablar.
☐	En `handle`, tras un comando reconocido: `message = oncommand(command); speak(message, lang);`	La misma frase se muestra y se oye.

1 ¿Por qué **no** decimos en voz alta el mensaje «not a command»?

2 ¿Por qué es seguro hablar justo después de escuchar?

TAREA 6 · CAMBIO DE SITIO (4 MIN)

En parejas, intercambiad ordenadores. Sin explicaciones, ejecuta dos comandos de la app de tu compañero por voz, usando solo lo que dice la pantalla. Después bloquea el micrófono desde la barra de direcciones y pulsa **Speak**. Escribe una frase de feedback en un comentario de Figma en su tablero B.

ENTREGA

En Classroom: (1) un **enlace de solo lectura** a tu copia de Figma con el tablero B rellenado; (2) una **grabación de pantalla de 30 segundos** con dos comandos funcionando y un «not a command»; (3) una línea: ¿cuál de tus comandos es el más arriesgado si se entiende mal y qué protege al usuario?

PARA CASA

- Termina los seis estados y añade tu tercer y cuarto comando.
- Escribe tres tests de Vitest para tu `parseCommand`: una frase que debe coincidir, una con mayúsculas y puntuación, y una que debe devolver `null`.
- Ejercicio de transferencia (un párrafo): la app de un horno controlado por voz entiende «start» y «stop». Nombra dos cosas que pueden fallar en una cocina ruidosa y qué debería hacer la interfaz en cada caso.