# Implementación del módulo Guía con Driver.js (Prompt para Codex)

## Contexto y objetivo

Queremos **incorporar un módulo “Guía” reutilizable** en nuestro proyecto *Indexlab* para mejorar el onboarding de los usuarios. Este módulo presentará tours interactivos que destaquen elementos clave de la interfaz usando la librería **Driver.js**. La idea es que cada aplicación dentro de la plataforma pueda ofrecer un mini-tutorial inicial, actuando como un pequeño tutor integrado que **guía la atención del usuario** hacia las funcionalidades principales, aumentando la motivación y la comprensión de la herramienta.

Driver.js es una librería JavaScript ligera (~5 KB) y sin dependencias, diseñada para crear recorridos guiados, resaltando elementos de la página con un fondo oscurecido y pop-ups explicativos[[1]](https://blog.logrocket.com/using-driver-js-guide-user-focus/#:~:text=A%20popular%20example%20of%20user,to%20your%20site%20or%20app)[[2]](https://blog.logrocket.com/using-driver-js-guide-user-focus/#:~:text=What%20is%20Driver). Es compatible con cualquier framework frontend o incluso JavaScript puro, y permite *tours* de múltiples pasos con controles de navegación y personalización de estilo[[3]](https://blog.logrocket.com/using-driver-js-guide-user-focus/#:~:text=You%20can%20use%20this%20library,js%20use%20cases). Usaremos Driver.js para implementar nuestro módulo **Guía**, de forma que sea **independiente y reutilizable** en todas las apps del proyecto, con un esfuerzo mínimo por app (solo definir los pasos específicos de cada tour).

## Pasos de implementación

### 1. Agregar Driver.js al proyecto

Primero, incorporaremos la librería Driver.js en el repositorio. Hay dos opciones: mediante un gestor de paquetes (NPM) o usando los archivos distribuidos directamente (CDN o descarga). En este caso, optaremos por tener los archivos localmente para un uso offline consistente, similar a cómo ya incluimos otras librerías (por ejemplo, Tone.js en libs/vendor).

* **Instalación via NPM:** Ejecutar npm install driver.js para añadir la dependencia al *package.json*. Esto descargará el paquete que incluye tanto el script JS como el CSS[[4]](https://driverjs.com/docs/installation#:~:text=,js).
* **Integración de archivos locales:** Tras instalar, copiamos/aseguramos que los archivos necesarios (driver.min.js y driver.css de la carpeta node\_modules/driver.js/dist) estén disponibles en nuestro proyecto. Por ejemplo, podríamos colocarlos en libs/vendor/ (como libs/vendor/driver.min.js y libs/vendor/driver.css) junto a Tone.js. Alternativamente, si utilizamos módulos ES y un *bundler* o loader, podemos importarlos directamente desde node\_modules en el código (ver siguiente sección).

**Nota:** No olvidemos incluir la hoja de estilos de Driver.js, ya que es lo que provee el estilo del recuadro y el fondo oscurecido durante el tour. Si no usamos un bundler que procese imports de CSS, incluyamos manualmente el <link rel="stylesheet" href="../../libs/vendor/driver.css"> en el <head> del HTML de la app donde probaremos el tour (App8 en este caso). Esto aplicará los estilos necesarios de las *popovers* y resaltado[[5]](https://driverjs.com/docs/installation#:~:text=Alternatively%2C%20you%20can%20use%20CDN,script%20in%20your%20HTML%20file).

### 2. Crear el módulo Guía (código reutilizable)

A continuación, crearemos un **nuevo módulo compartido**, por ejemplo en libs/guide/ (o libs/guia/), que contendrá la lógica para inicializar y ejecutar los tours con Driver.js. Este módulo tendrá la responsabilidad de configurar Driver.js y exponer una función sencilla para iniciar una guía paso a paso.

**Pasos para el módulo Guía:**

* **Archivo y estructura:** Crear un archivo libs/guide/index.js (módulo ES). En él:
* Importar la librería Driver.js y su CSS. Por ejemplo, si usamos ES modules:
* import { driver } from 'driver.js';   
  import 'driver.js/dist/driver.css';
* Si en vez de import global tenemos los archivos locales, podemos importarlos apropiadamente o asegurarnos de incluir el script antes. Con import ES6 asumiremos que el entorno de desarrollo lo resolverá (p. ej., mediante un servidor de desarrollo que permita importar desde *node\_modules*). En caso contrario, podríamos cargar Driver.js globalmente y acceder a window.driver (pero es menos modular).
* Definir una función de inicialización, por ejemplo export function startTour(stepsConfig) { ... }. Esta función recibirá la configuración de pasos (un array de objetos) y opcionalmente opciones de Driver. Dentro de la función:
  + Crear una instancia de Driver.js con la configuración deseada. Por ejemplo:
  + const driverObj = driver({  
     showProgress: true, // muestra barra de progreso/paginación de pasos  
     allowClose: false, // (opcional) impedir cerrar haciendo click fuera, para forzar seguir tour  
     opacity: 0.5, // (ejemplo) opacidad del overlay  
     steps: stepsConfig  
    });
  + Aquí usamos driver({ ... }) pasando un objeto con opciones y la lista de pasos. La opción **showProgress: true** agrega un indicador del progreso (paso 1 de N) en el popover[[7]](https://driverjs.com/docs/basic-usage#:~:text=import%20,driver.js%2Fdist%2Fdriver.css), lo cual mejora la claridad para el usuario. Podemos ajustar otras opciones (como keyboardControl, etc.) si es necesario, pero inicialmente con *defaults* está bien.
  + Iniciar el recorrido llamando a driverObj.drive() una vez configurado[[8]](https://driverjs.com/docs/basic-usage#:~:text=driverObj). Este método comienza la secuencia de pasos tal como se definió.
  + (Opcional) Retornar driverObj en caso de que queramos interactuar con la instancia desde fuera (por ejemplo, para detener el tour o avanzar manualmente en ciertos casos). En principio no es necesario, ya que el tour se controla solo hasta completarse o ser cerrado por el usuario.

Con esto, nuestro módulo Guía ofrece una interfaz muy simple: cualquier app puede importar startTour y pasarle sus pasos, sin preocuparse por detalles internos de Driver.js cada vez.

### 3. Definir una guía de ejemplo en **App8**

Vamos a probar el módulo Guía en **App8: Modo Joc d'Intervals**, que ya tiene elementos de gamificación (avatares, perfiles, niveles, estadísticas) ideales para un tour introductorio. El objetivo es mostrar un *mini-tour* cuando el usuario abre App8, destacando las partes clave de la pantalla de bienvenida de esta app.

**a. Preparar App8 para el tour:**  
- En el HTML de apps/app8/index.html, asegurarnos de incluir la hoja de estilos de Driver.js. Como mencionamos, añadiremos:

<link rel="stylesheet" href="../../libs/vendor/driver.css" />

dentro de <head> (junto a otras hojas de estilo). Esto cargará los estilos necesarios (dim de fondo, cuadros de popover, etc).  
- Incluir el script JS de Driver.js si no se va a importar como módulo. Dado que planeamos usar import { driver } from 'driver.js' dentro de nuestro módulo Guía, **no** insertaremos aquí la versión IIFE global, para evitar conflictos. (Si hubiéramos optado por no usar imports, entonces sí incluiríamos <script src="../../libs/vendor/driver.min.js"></script> antes del script principal de la app).

**b. Invocar el tour en el código de App8:**  
- En el archivo apps/app8/scripts/main.js, importar nuestra función startTour del módulo Guía. Estará disponible vía ruta relativa, por ejemplo:

import { startTour } from '../../../libs/guide/index.js';

Colocamos esta importación junto con las otras al inicio del archivo.  
- Después de que la interfaz inicial de App8 esté lista (por ejemplo, tras renderizar perfiles y habilitar botones), llamaremos a startTour(...) con los pasos que queremos mostrar. Dado que App8 tiene una pantalla de bienvenida con la selección de perfil y niveles, definiremos pasos enfocados en esos elementos.

**c. Configurar los pasos de la guía para App8:**  
Imaginemos un recorrido introductorio de **3 pasos** para App8, cubriendo: selección de perfil, selección de nivel, y botón de estadísticas. Definimos el array tourSteps antes de llamar a startTour:

const tourSteps = [  
 {  
 element: '#profiles', // contenedor de perfiles de usuario  
 popover: {  
 title: 'Perfiles de usuario',  
 description: 'Aquí puedes crear un perfil o seleccionar tu perfil existente. Usa un perfil para guardar tu progreso y estadísticas.'  
 }  
 },  
 {  
 element: '#levelInfo', // lista de niveles de entrenamiento  
 popover: {  
 title: 'Niveles de entrenamiento',  
 description: 'Estos son los niveles disponibles. Comienza por el nivel 1 e irá desbloqueando niveles superiores a medida que completes cada uno.'  
 }  
 },  
 {  
 element: '#showStats', // botón de ver estadísticas  
 popover: {  
 title: 'Estadísticas',  
 description: 'Puedes consultar en cualquier momento tus estadísticas y medallas obtenidas haciendo clic aquí.'  
 }  
 }  
];  
startTour(tourSteps);

Cada objeto en tourSteps define un paso: - **element** indica el selector CSS del elemento a resaltar (ej. #profiles para el contenedor de slots de perfil, #levelInfo para la lista de botones de nivel, #showStats para el botón de estadísticas).  
- **popover** contiene el contenido que se mostrará al usuario: un title breve y una description explicativa. Hemos redactado las descripciones para que sean **claras y motivadoras**, señalando qué hace cada elemento.

Cuando se ejecute startTour(tourSteps), el módulo Guía creará un Driver.js instance con estos pasos y lanzará el tour automáticamente. El resultado esperado es:  
1. Se oscurece la pantalla salvo el área de **Perfiles**, mostrando un recuadro destacado sobre ese contenedor y un pop-up que dice *"Perfiles de usuario – Aquí puedes crear un perfil o seleccionar..."*.  
2. Al pulsar “Siguiente” en el pop-up, el foco pasará al área de **Niveles**, destacándola y mostrando *"Niveles de entrenamiento – Estos son los niveles disponibles..."*.  
3. En el siguiente paso, se destacará el botón **Estadísticas**, con la explicación de que allí se consultan estadísticas y medallas.  
4. Finalmente el tour se cerrará (podemos configurar un botón “Finalizar” o que al acabar el último paso simplemente se cierre). Con showProgress: true, el usuario verá un indicador (por ejemplo "Paso 1 de 3")[[9]](https://driverjs.com/docs/basic-usage#:~:text=showProgress%3A%20true%2C%20steps%3A%20%5B%20,), reforzando que está en un tutorial interactivo.

**d. Condicionar la ejecución del tour (opcional):**  
Para no mostrar este tour cada vez que se entra a App8, podemos envolver la llamada startTour(tourSteps) en una lógica condicional. Por ejemplo, usar localStorage para almacenar una clave tutorialApp8Done tras la primera ejecución. Si esa clave no existe, lanzamos el tour y luego la establecemos; si ya existe (el usuario ya vio el tour), podríamos no iniciarlo automáticamente. Esto evitaría *molestar* al usuario recurrente. Dado que aquí buscamos un prototipo rápido, podemos omitir esta lógica inicialmente y simplemente **ejecutar el tour siempre** para verificar su funcionamiento, pero es bueno tener en cuenta este detalle para producción.

### 4. Pruebas del tour en App8

Con todo configurado, crear testspara la aplicación **App8**:

* Verificamos que al cargar la página, nuestro código efectivamente llama a startTour(tourSteps) (por ejemplo, poniendo la llamada al final de un DOMContentLoaded listener, si es necesario, para esperar a que el DOM esté listo). En nuestro caso, si el import y llamada están al final del módulo main.js, y main.js se carga con type="module", ejecutará la guía una vez se importe todo y ejecute el script (tras la construcción inicial del DOM, según el orden en el HTML).
* Al cargarse, deberíamos ver inmediatamente el primer paso del tour. Comprobamos: ¿Se ve el overlay oscureciendo el fondo y resaltando solo el elemento indicado? ¿Aparece la ventana emergente con el título y descripción correctos para el paso 1?
* Navegamos por los pasos con los botones Next/Siguiente. Confirmamos que cada elemento previsto se destaca en orden y el texto corresponde.
* **Detalles a pulir:** Asegurarse de que los selectores usados (#profiles, #levelInfo, #showStats) existan en el DOM en ese momento. Por ejemplo, el contenedor #profiles está presente desde el inicio (sí, en el HTML está listo). #levelInfo también está en el HTML inicial con los botones de nivel. El botón #showStats existe pero inicialmente está deshabilitado (disabled hasta que haya perfil seleccionado)[[10]](https://github.com/PlayNuzic/Indexlab/blob/2b4857aec1c5fee7475162513ff1ffa0dbe8da80/apps/app8/scripts/main.js#L94-L102)[[11]](https://github.com/PlayNuzic/Indexlab/blob/2b4857aec1c5fee7475162513ff1ffa0dbe8da80/apps/app8/scripts/main.js#L221-L229). Aun estando deshabilitado, Driver.js igual lo puede resaltar. Debemos verificar si un elemento deshabilitado/oculto pudiera causar algún problema: en este caso #showStats está visible pero inactivo; resaltar debería funcionar correctamente (apenas que quizá quiera interactuar y no puede, pero al ser un tour informativo no importa).
* Si todo funciona, el prototipo cumple su cometido: el usuario nuevo en App8 recibe un **mini-tour introductorio** y entiende dónde crear su perfil, cómo elegir niveles y dónde ver sus estadísticas, **antes de empezar a usar la app por sí mismo**. Esto mejora la experiencia inicial sin requerir que descubra esas funciones por su cuenta.

### 5. Extender el módulo Guía a otras apps

Una vez validado en App8, podremos **extender fácilmente** esta funcionalidad a las demás aplicaciones del proyecto. Cada app tiene elementos y objetivos distintos, pero el patrón es el mismo: definir un array de pasos relevante y llamar a startTour(…). Por ejemplo: en otra app podríamos resaltar el botón de “Nueva partitura”, luego una zona de edición, etc., con sus descripciones. La belleza de tener el módulo Guía es que **no repetiremos la lógica** de crear tours en cada app, solo proporcionaremos los datos (selectores + textos) específicos de cada caso.

El módulo Guía actúa como una **capa independiente y reutilizable** para dirigir la atención del usuario, y la definición de pasos podría derivarse de diversas fuentes (manual o automática).

## Conclusión

Siguiendo estos pasos, habremos implementado un **prototipo funcional del módulo Guía** usando Driver.js. En resumen: añadimos la librería, creamos una función de inicialización del tour, y la probamos en App8 con un recorrido de ejemplo. Con la base ya integrada, resulta sencillo añadir tutoriales interactivos en las demás aplicaciones simplemente definiendo sus pasos específicos. Esto aportará un gran valor al proyecto, pues cada app podrá contar con una introducción atractiva y controlada para el usuario, **como tener un pequeño tutor dentro de la aplicación** que le enseña o recuerda qué hacer en cada pantalla de forma amena. ¡Todo ello mejorará la experiencia de usuario y la motivación desde el primer contacto con nuestras apps, sin requerir un esfuerzo significativo de desarrollo adicional por cada una!