Prueba Cliente Rapimoney

⚠️ Se adjunta una base de datos de prueba .sqlite para el desarrollo de la aplicación el cual se puede modificar a su criterio. Se recomienda ubicarlo en la carpeta raíz del proyecto.

Objetivo

Desarrollar una aplicación para gestión de clientes , La misma debe permitir crear y mostrar los datos de los clientes registrados.

Requerimientos Técnicos

Deberás desarrollar una API en Node.js junto al framework Express en su versión estable.

Los datos mostrados deben ser persistidos en una base de datos relacional. El esquema de datos puede armarse según se considere apropiado en base a los requerimientos del negocio base. La API deberá exponer URLS que devuelvan datos en JSON.

Estos datos en JSON deberán ser consumidos por un cliente, a través de peticiones AJAX(usar cualquier librería a criterio personal). El cliente puede ser armado con React.js o (HTML,CSS,JS).

El trabajo realizado se subirá a un drive proporcionado por Rapimoney.

Secciones:

Home Cliente

La pantalla de inicio debe mostrar un formulario de registro de datos del cliente y poder visualizar listado de hasta 10 clientes por pagina con la opción de “ver más” para visualizar los demás clientes registrados.

La aplicación deberá contener:

● Formulario de registro de clientes:

Datos del cliente:

DNI
Nombres
Apellidos
Fecha de Nacimiento
Celular
Correo
Banco
numero de Cuenta o CCI

● Listado de clientes registrados.

Bonus

De forma adicional, puede:

- Búsqueda de cliente por DNI o Nombre:

  Agregar un filtro para visualizar un cliente registrado en base al DNI o nombre.

- Login de cliente registrado.

  Un formulario de Login para permitir identificar al usuario registrado y mostrar un mensaje.

Criterios a Evaluar

● Buenas prácticas para el uso del archivo .sqlite.

● El diseño debe ser responsive, pudiendo utilizarse CSS puro o algún framework de Frontend.

● Tener cuidado con las versiones de los módulos declarados en el .package.json

● Código limpio, buenas prácticas de programación, en idioma inglés.

● Buenas prácticas de GIT: Commits declarativos y atomizados.

- Buenas prácticas para el nombre de rutas.
- Implementación de Logs.