**📝 Explicación del Funcionamiento Básico y la Lógica Aplicada en el Proyecto**

**📌 Nombre del proyecto:**

**proyectoAngular\_YGS**

**1. Objetivo General**

El proyecto tiene como objetivo gestionar una lista de estudiantes, permitiendo visualizar, agregar, editar y eliminar registros desde una interfaz web desarrollada con Angular. Los datos se manipulan dinámicamente en memoria y se visualizan en una grilla interactiva.

**2. Estructura General del Proyecto**

El proyecto está compuesto por los siguientes componentes principales:

* AppComponent: componente raíz que gestiona el estado principal de la aplicación, incluyendo la lista de estudiantes y las acciones sobre ellos.
* StudentsTable: componente que muestra la lista de estudiantes en formato de tabla.
* AddForm: componente de formulario para agregar o editar un estudiante.
* Toolbar y Navbar: componentes de interfaz visual que simulan una barra de herramientas y una navegación superior.

**3. Funcionamiento General**

**a. Carga inicial de datos**

Al iniciar, la aplicación muestra una lista predefinida de estudiantes cargada en memoria (simulando una fuente de datos externa como una API o archivo JSON).

**b. Agregar un nuevo estudiante**

* El formulario (AddForm) permite ingresar los datos de un nuevo estudiante.
* Al enviarlo, se emite un evento al componente principal (AppComponent) que agrega el nuevo estudiante a la lista con un ID generado automáticamente.
* La grilla (StudentsTable) se actualiza automáticamente gracias al enlace de datos (@Input()).

**c. Editar un estudiante existente**

* Desde la grilla, se puede seleccionar un estudiante para editar.
* El estudiante se carga en el formulario.
* Al guardar, si el estudiante tiene un ID existente, se actualiza su información en lugar de crear una nueva entrada.

**d. Eliminar un estudiante**

* Desde la grilla, se puede eliminar un estudiante.
* La lista se actualiza excluyendo al estudiante seleccionado.

**e. Visualización del JSON**

* Debajo del formulario, se muestra la representación en tiempo real de los datos en formato JSON para facilitar el seguimiento de los cambios.

**4. Lógica de Control**

* Se utiliza @Input() y @Output() para la comunicación entre componentes.
* La lógica de edición y creación se basa en la presencia o ausencia del id del estudiante.
* Se utiliza FormBuilder con validadores para asegurar que los datos ingresados cumplan requisitos básicos.
* Para forzar la actualización de listas, se reasigna el array students usando el spread operator (this.students = [...this.students]).

**5. Tecnologías Utilizadas**

* **Angular CLI** (versión más reciente)
* **Bootstrap** (para estilos generales)
* **TypeScript**
* **SCSS**
* **Angular Material** (opcional para tabla o inputs más estilizados)
* **Vite** (como servidor de desarrollo)