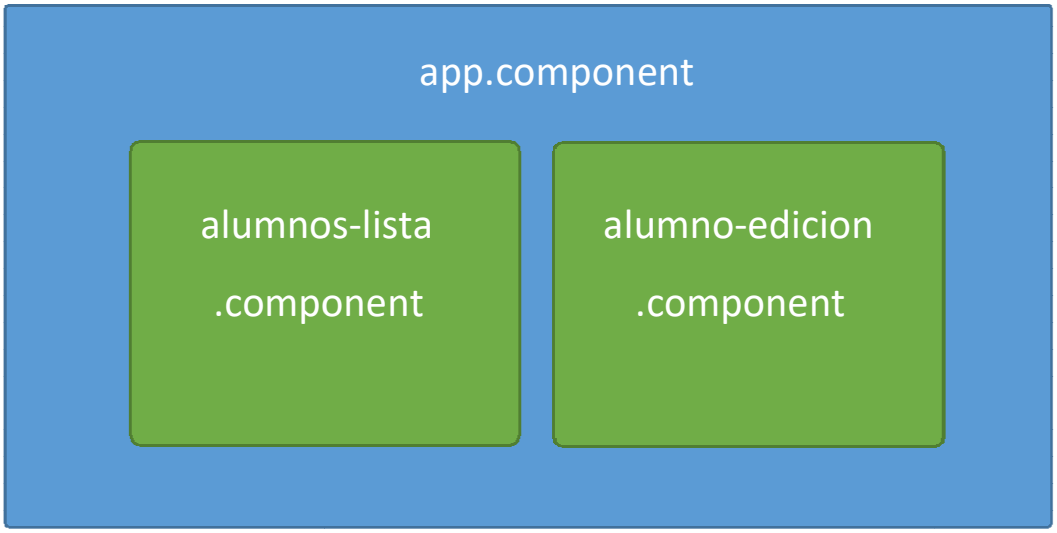
Práctica #4

Objetivos

- Crear un componente separado que muestre la lista de Alumnos
- Crear un componente separado que edite el alumno seleccionado
- El componente principal alimentará con datos a los componentes hijos a través de propiedades input.
- El componente lista informará al componente principal del alumno seleccionado a través de un evento.



1) Crear los componentes de lista de alumnos y de edición de alumno

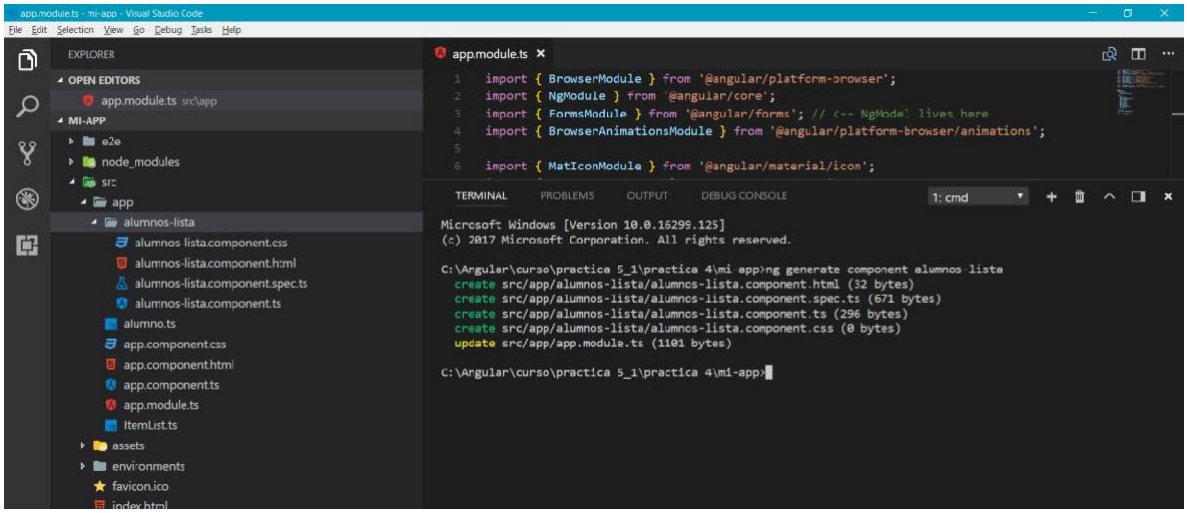
- Si no tiene abierto el proyecto de las prácticas, ábralo con Visual Studio Code.
- Abra la terminal **(CTRL + ñ)** y ejecute el siguiente comando:

ng generate component alumnos-lista

Cuando finalice de ejecutarse, se habrá generado:

✓ Una sub carpeta **alumnos-lista** en la carpeta **app**

✓ Dentro de esa carpeta 4 archivos correspondiente al componente: el template (.html), los estilos (.css), la clase (.ts) y el de testing (.spec.ts)



- Realice el mismo procedimiento para generar el componente **alumno-edicion**
- Edite el módulo principal de la aplicación, el archivo **app.module.ts**. Revise como este ha sido modificado por la herramienta angular-cli generando los imports y agregando los componentes al array de declarations

2) Armar el componente alumnos-lista moviendo partes de app.component

- Abra el archivo **app.component.html** y seleccione todo el contenido del elemento **<mat-card class="lista">** .

```
<mat-card class="lista">

  <mat-list>
    <mat-list-item
      *ngFor="let alumno of alumnos"
      class="alumno-list-item"
      (click)="AlumnoSelect( alumno )"
      [class.alumno-seleccionado] = EstaSeleccionado(alumno) >
    <div mat-line>
      <h2>
          <mat-icon class="avatar">account_circle</mat-icon>
          {{alumno.nombre}} {{alumno.apellido}}
      </h2>
    </div>

        <p mat-line>
```

```
          Perfil: {{ Perfiles(alumno.perfil) }}<br><br>
        </p>
      </mat-list-item>
    </mat-list>
  </mat-card>
```

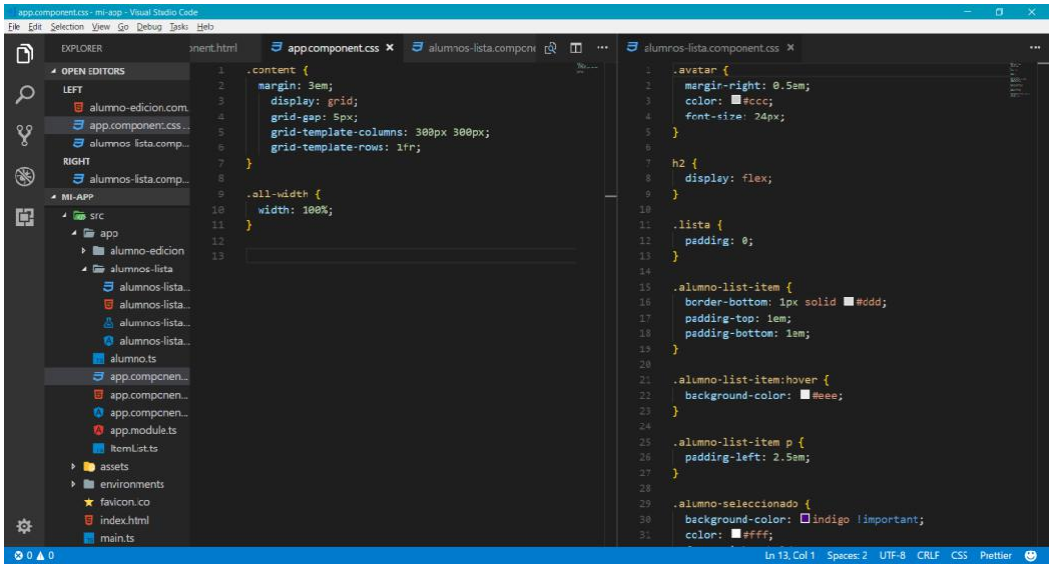
Esta parte del código es la que arma la lista.

- Corte el código.
- Copie el código html extraído en el archivo **alumnos-lista.component.html** reemplazando el contenido generado por angular-cli.
- Guarde los cambios en ambos archivos.

Ahora moveremos los estilos CSS.

- Utilizando la función de Visual Studio Code de visualizar la edición de dos archivos en dos paneles verticales (el ícono de esta funcionalidad esta en la parte superior derecha) abra los archivos **app.component.css** y **alumnos-lista.component.css**.
- Mueva los siguientes estilos de **app.component.css** a **alumnos-lista.component.css**:
  - ✓ .avatar
  - ✓ H2
  - ✓ .lista
  - ✓ .alumno-list-item
  - ✓ .alumno-list-item:hover
  - ✓ .alumno-list-item p
  - ✓ mat-list
  - ✓ .alumno-seleccionado

Así deberían quedar ambos archivos:



- Guarde los cambios.

Ahora moveremos la funcionalidad.

- Edite el archivo **alumnos-lista.component.ts** y agregue los siguientes imports debajo del import al inicio del archivo:

```
import { Alumno } from '../alumno';
import { ItemList } from '../ItemList';
```

Vea que la referencia a los archivos tienen un .. en vez de un solo punto. ¿A qué se debe esto?

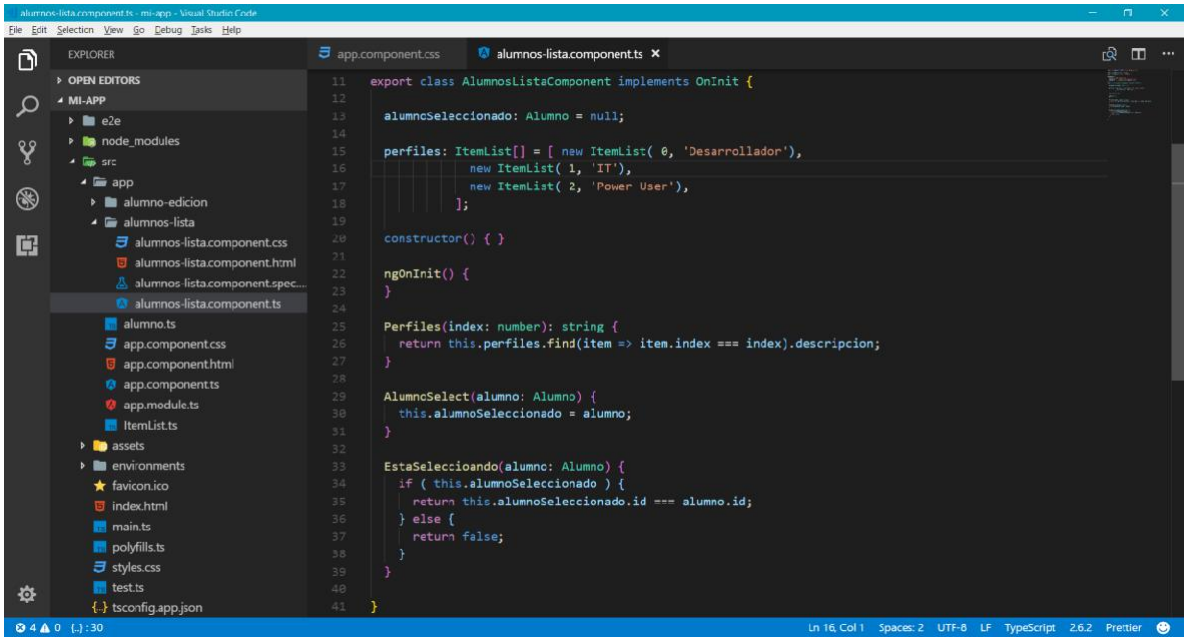
- Este componente trabaja con la propiedad de **alumnoSeleccionado**. Entre la cabecera de la clase y el constructor agregue esta propiedad:

```
export class AlumnosListaComponent implements OnInit {

  alumnoSeleccionado: Alumno = null;

  constructor() { }
```

- Copie del archivo **app.component.ts** el array de perfiles y ubíquelo debajo de la propiedad alumnoSeleccionado. Aquí estaremos duplicando la funcionalidad, pero en las próximas prácticas solucionaremos esto.
- Copie también la función **Perfiles()**, ubíquela debajo de la función **ngOnInit()**. Por cierto, cual es la funcionalidad de ngOnInit()? (https://angular.io/guide/lifecycle-hooks#oninit)

- **Mueva** las funciones **AlumnosSelect()** y **EstaSeleccionado()** a **alumnos-lista.component.ts**

La clase debería quedar así:



Así como está el componente **alumnos-lista** no tiene forma de comunicarse con el componente padre. Para que esta comunicación se realice el componente padre deberá poder pasarle el array de alumnos y el componente **alumnos-lista** emitir un evento cada vez que se seleccione un alumno de la lista. Para ello vamos a seguir editando el archivo **alumnos-lista.component.ts**

- Agregue en el import de **Angular/core** la referencia a **Input**

```
import { Component, OnInit, Input } from '@angular/core';
```

- Defina la propiedad alumnos como una propiedad input del componente (one way databinding)

```
export class AlumnosListaComponent implements OnInit {

  @Input() alumnos: Alumno[];
  alumnoSeleccionado: Alumno = null;
```

- Guarde los cambios

Ahora vamos a utilizar el componente **alumnos-lista** en el componente **app.component**. Para ello:

- Edite el archivo **app.component.html**
- Agregue el componente alumnos-lista al html:

```
<div class="content">
  <div></div>

  <app-alumnos-lista [alumnos] = "alumnos"> </app-alumnos-lista>

  <mat-card *ngIf="alumnoSeleccionado">
```

Vea el selector del elemento: **app-alumnos-lista**. ¿En dónde se define esto? Vea también el cómo hacemos uso de la propiedad alumnos.

Guarde los cambios y pruebe la aplicación, Vera que la lista esta funcional pero no así la selección ya que nos falta comunicar la selección al componente padre, tema que resolveremos ahora:

- Edite el archivo **alumnos-lista.component.ts**
- Agregue en los imports la referencia a la funcionalidad de definir y emitir eventos:

```
import { Component, OnInit, Input, Output, EventEmitter } from
'@angular/core';
```

- Definamos ahora el evento:

```
  @Input() alumnos: Alumno[];
  @Output() Seleccion = new EventEmitter<Alumno>();

  alumnoSeleccionado: Alumno = null;
```

El decorador @Output define a la propiedad como One Way Databinding de salida, asociada a un evento (**EventEmitter**) que tiene como parámetro de salida un tipo de datos **Alumno**. Esto se puede leer así: *"Selección es un evento que retorna el alumno seleccionado"*.

- Lo que nos falta es emitir el evento cada vez que se seleccione un alumno y pasarle el alumno seleccionado. Agregue la siguiente línea de código en la función **AlumnosSelect()**:

```
  AlumnoSelect(alumno: Alumno) {
   this.alumnoSeleccionado = alumno;
   this.Seleccion.emit(this.alumnoSeleccionado);
  }
```

Aquí es donde emitimos el alumno seleccionado. Vea como en la función **emit** pasamos el alumno.

- Guarde los cambios.
- Edite el archivo **app.component.html** y agregue el código para capturar el evento

```
<div class="content">

  <app-alumnos-lista [alumnos] = "alumnos"
                     (Seleccion)="Seleccionar($event)">
  </app-alumnos-lista>
```

Vea especialmente el cómo pasamos el alumno seleccionado a la función **Seleccionar**() a través de la variable **$event**.

- Guarde los cambios.
- Nos falta ahora la función **Seleccionar()**. Agregue esta función en el código del componente **App** luego de la función **Perfiles()**.

```
  Seleccionar(alumno: Alumno): void {
    this.alumnoSeleccionado = alumno;
  }
```

- Guarde los cambios y pruebe la aplicación. Ahora la funcionalidad de seleccionar vuelve a estar operativa.

3) Armar el componente alumno-edicion moviendo partes de app.component

En este paso tendremos que:

1. Mover parte del html de **app.component.html** a **alumno-edicion.component.html**
2. Mover las definiciones de los array **perfiles** y **sexo** junto con las funciones **Perfiles**() y **Sexos**() a **alumno-edicion.component.ts**
3. Importar las definiciones de **ItemList y Alumno**
4. Definir en **alumno-edicion** una propiedad de Input llamada **alumnoSeleccionado**
5. Insertar el componente en el html de **app.component.html** utilizando la propiedad de Input **alumnoSeleccionado**.
6. Reemplazar en el html de **app.component.html** todo el código de edición de alumnos por el componente nuevo.
7. Pruebe la aplicación. Vea que ahora la edición se ve levemente diferente. Que es lo que faltó mover al componente?

Los invitamos a realizar estos pasos por sí mismos.