Práctica #7

Objetivo

- Cambiar al componente principal para que muestre un menú y utilice rutas
- Cambiar el componente de lista de alumnos para que la comunicación sea a través de las rutas y que el listado de alumnos se realice utilizando el componente **mat-Table** de Angular Material
- Utilizar la funcionalidad de routing de Angular para navegar y pasar datos entre componentes.

1) Estructura de los cambios que vamos a realizar

Guarde una copia de la aplicación tal cual la hemos desarrollado hasta ahora, así tendrá una referencia de cómo hemos implementado los conceptos del curso hasta el momento.

2) Creación de los nuevos componentes

Vamos a crear dos componentes nuevos. Estos componentes no tendrán funcionalidad especifica más que la de mostrar cómo utilizamos las rutas de Angular.

- 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 asistencias

- Esto generará un componente llamado asistencias en la carpeta asistencias. Cuando finalice su ejecución ejecute el siguiente comando:

  ng generate component cursos

3) Definir las rutas de los nuevos componentes

Comenzaremos ahora a definir la infraestructura del Routing, para ello:

- Edite el archivo **app.module.ts**
- Agregue el **import** del módulo del router de Angular

```
import { RouterModule } from '@angular/router';
```

Agregue las rutas a los componentes modificando el array de **imports** del módulo:

```
imports: [
  BrowserModule,
  MatToolbarModule,
  MatCardModule,
  MatIconModule,
  MatListModule,
  MatInputModule,
  MatFormFieldModule,
  FormsModule,
  BrowserAnimationsModule,
  MatSelectModule,
  MatCheckboxModule,
  MatRadioModule,
  MatButtonModule,
  RouterModule.forRoot([
      { path: 'alumnos' , component: AlumnosListaComponent},
      { path: 'cursos' , component: CursosComponent},
      { path: 'asistencias' , component: AsistenciasComponent},
      { path: '', redirectTo: '/alumnos', pathMatch: 'full' } ])
],
```

Hemos definido 3 rutas por ahora: alumnos, cursos, asistencias, vinculadas con sus respectivos componentes. Más adelante agregaremos la ruta para la edición de alumno.

Vea que además hemos creado una ruta de redireccionamiento que se utilizara cuando se navegue a la url base de la aplicación.

- Guarde los cambios del módulo.

4) Modificación de app.component

Al componente **app.component** vamos a agregar un menú en la parte lateral izquierda de la pantalla que nos servirá para navegar por diferentes partes de nuestra aplicación.

- Vamos a cambiar el html. Edite el archivo **app.component.html** y reemplácelo con la nueva estructura:

```
<mat-toolbar color="primary" class="mat-elevation-z3"> <span>Gestión de
  cursos y alumnos</span>
</mat-toolbar>

<div class="content">
  <div class="menu">

    <a routerLink="/alumnos" routerLinkActive="activo"> <i
      class="material-icons">people</i><br/> Alumnos
    </a>

    <a routerLink="/cursos" routerLinkActive="activo"> <i
      class="material-icons">event</i><br/> Cursos
    </a>

    <a routerLink="/asistencias" routerLinkActive="activo"> <i
      class="material-icons">assignment_turned_in</i><br/> Asistencias
    </a>

  </div>

  <div class="contenido">
    <router-outlet></router-outlet>
  </div>

 </div>
```

Revise el código html. Vea el cómo hemos utilizado la directiva **routerlink** para vincular el hipervínculo con la ruta. Vea también cómo asignamos la clase **activo** a la ruta activa a través de la directiva **routerLinkActive**. Esta clase CSS la crearemos en el siguiente punto.

Ahora vamos a tener dos paneles verticales, uno para el menú a la derecha y otro para el contenido. Vamos a cambiar las reglas CSS para reflejar estos cambios.

- Edite el archivo **app.component.css**
- Reemplace la clase **.content** por

```
.content {
  margin: 3em;
    display: grid;
```

```
    grid-gap: 5px;
    grid-template-columns: 200px 1fr;
    grid-template-rows: 1fr;
}
```

Vea que ahora el template de columnas refleja dos columnas, una con un ancho fijo.

- Agregue las siguientes reglas css

```
.all-width {
  width: 100%;
}

.menu {
  background-color: #444;
  position: absolute;
  top: 64px;
  bottom: 0px;
  left: 0px;
  right: 0px;
  text-align: center;
  color: rgba( 255,255,255, 0.85);
  font-size: 16px;
}

.menu a , .menu a:visited {
  color: rgba( 255,255,255, 0.85);
}

.menu a {
  display: block;
  margin-top: 60px;
  margin-left: 30px;
  padding: 16px;
  width: 100px;
  border-radius: 4px;
  text-decoration: none;
  transition: background-color 1s linear;
  font-family: Roboto, Arial, Helvetica, sans-serif;
}

.menu a i {
```

```
  font-size: 48px;
  color: rgba( 255,255,255, 0.85);
  transition: background-color 1s linear;
}

.menu a:hover,  .menu a:hover i{
  background-color: #eee;
  color: #444;
  transition: background-color 1s linear;
}

.contenido {
  background-color: #fcfcfc;
  position: absolute;
  top: 64px;
  bottom: 0px;
  left: 200px;
  right: 0px;
  padding: 16px;
}

.activo {
  border: 2px solid rgba(64,240,200, .7);
}
```

Estas reglas dan el estilo al menú y a los paneles.

- Guarde los cambios y pruebe la aplicación. Navegue por el menú y observe como cambian las URL en el navegador. Pruebe de ingresar alguna URL a mano. Vea que es lo que pasa cuando entramos a Alumnos. Esto lo solucionaremos en el siguiente punto

5) Modificar el componente alumno-lista

En este componente realizaremos varios cambios:

- ✓ Desacoplar los eventos y propiedades de Input que lo vinculaban con el componente principal
- ✓ Utilizar directamente el servicio de alumnos ya que la lista de alumnos no va a ser provista por el componente padre
- ✓ Utilizar el componente mat-Table de Angular Material para mostrar la lista de alumnos
- ✓ Adaptar los elementos de la interface de la versión anterior

- Edite el código del archivo **alumnos-lista.component.ts**

- Realice los cambios que están resaltados:

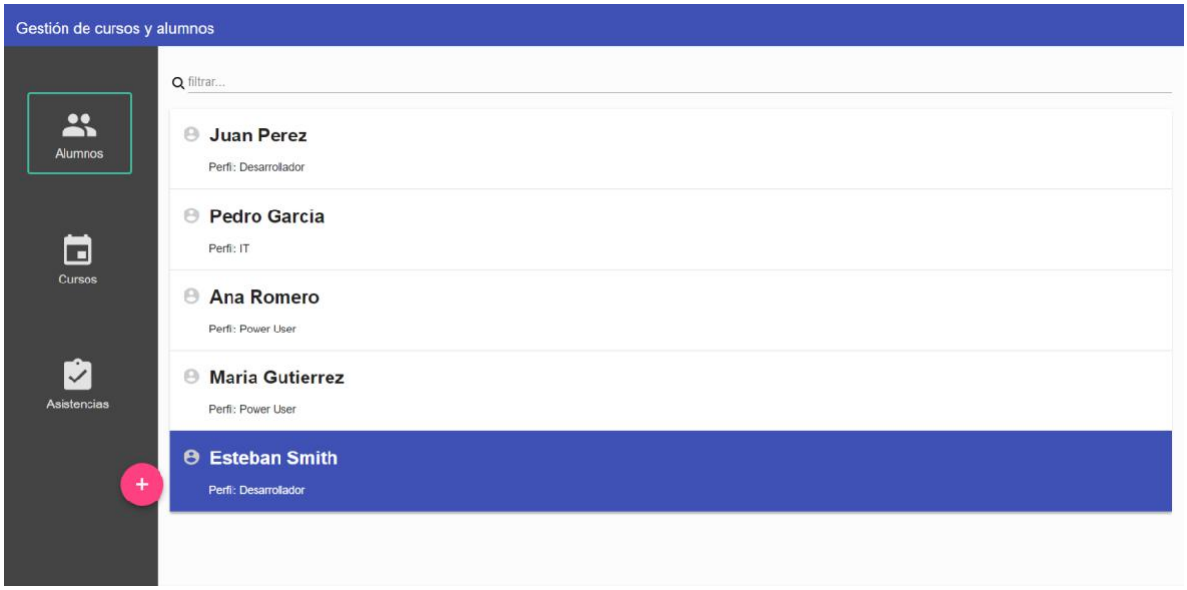
```
import { Component, OnInit, Input, Output, EventEmitter  } from
'@angular/core';
import { ItemListService } from '../item-list.service';
import { AlumnosService } from '../alumnos.service';

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

@Component({
  selector: 'app-alumnos-lista',
  templateUrl: './alumnos-lista.component.html',
  styleUrls: ['./alumnos-lista.component.css'] })
export class AlumnosListaComponent implements OnInit {

  alumnos: Alumno[];
  alumnoSeleccionado: Alumno = null;

   constructor(
    public ItemListSrv: ItemListService,
    public AlumnosSrv: AlumnosService
  ) { }

  ngOnInit() {
    this.alumnos = this.AlumnosSrv.GetAll();
  }

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

  EstaSeleccioando(alumno: Alumno) {
    if ( this.alumnoSeleccionado ) {
      return this.alumnoSeleccionado.id === alumno.id;
    } else { return false;
    }
  }

  Agregar() {
    // this.AgregarAlumno.emit();
  }
```

```
  Filtrar(filtro: string) {
    // this.FiltrarAlumnos.emit(filtro);
  }
}
```

Las modificaciones que hemos realizados son:

- ✓ Hemos utilizado el servicio **AlumnosService** y obtenemos el array de alumnos en el OnInit()
- ✓ Hemos eliminado todos los eventos que emitía el componente
- ✓ Hemos eliminado los @Input de las propiedades.

- Salve los cambios y pruebe la aplicación. Ingrese a la opción de alumnos. El resultado debe ser similar a este:



- Edite el archivo **alumnos-lista.component.css** y modifique la regla **:host** de la siguiente forma:

```
:host {
  position: absolute;
  left: 8px;
  top: 8px;
  right: 8px;
  bottom: 8px;
}
```

- Salve los cambios y vea el cómo se ha acomodado el botón de agregar alumnos

Nos queda ahora utilizar el **mat-table** para mostrar los alumnos reemplazando a los **mat-list y mat-listitems**

- Edite el archivo **app.module.ts.**
- Agregue estos dos imports

```
import { MatTableModule } from '@angular/material/table';

import { MatMenuModule } from '@angular/material/menu';
```

- Recuerde de agregar estos dos nuevos módulos a la lista de array de imports del módulo

```
    MatTableModule,
    MatMenuModule,
```

- Guarde los cambios
- Edite el archivo **alumnos-lista.component.html**
- Comente todo el elemento **mat-card**

```
<div class="search-box">
  <mat-icon >search</mat-icon>
  <mat-form-field>
      <input matInput #filtrar placeholder="filtrar..."
      (keyup)="Filtrar(filtrar.value)">
  </mat-form-field>
</div>

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

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

<button mat-fab class="add-button" (click)="Agregar()">+</button>
```

- Después del **div class=”search-box”** agregue el siguiente fragmento de HTML

```
<div class="tabla mat-elevation-z8">

  <mat-table #table [dataSource]="dataSource">

    <!-- Position Column -->
    <ng-container matColumnDef="id">
      <mat-header-cell *matHeaderCellDef> Id. </mat-header-cell>
      <mat-cell *matCellDef="let element"> {{element.id}} </mat-cell>
    </ng-container>

    <ng-container matColumnDef="nombre">
      <mat-header-cell *matHeaderCellDef> Nombre </mat-header-cell>
      <mat-cell *matCellDef="let element"> {{element.nombre}} </mat-cell>
    </ng-container>

    <ng-container matColumnDef="apellido">
      <mat-header-cell *matHeaderCellDef> Apellido </mat-header-cell>
      <mat-cell *matCellDef="let element"> {{element.apellido}} </mat-cell>
    </ng-container>

    <ng-container matColumnDef="perfil">
      <mat-header-cell *matHeaderCellDef> Perfil </mat-header-cell>
      <mat-cell *matCellDef="let element">
          {{ ItemListSrv.Perfiles(element.perfil)}}
      </mat-cell>
    </ng-container>

    <ng-container matColumnDef="sexo">
      <mat-header-cell *matHeaderCellDef> Sexo </mat-header-cell>
      <mat-cell *matCellDef="let element">
          {{ItemListSrv.Sexos(element.sexo)}}
```

```
      </mat-cell>
    </ng-container>

    <ng-container matColumnDef="activo">
      <mat-header-cell *matHeaderCellDef> Activo </mat-header-cell>
      <mat-cell *matCellDef="let element"> {{element.activo}} </mat-cell>
    </ng-container>

    <ng-container matColumnDef="acciones">
      <mat-header-cell *matHeaderCellDef></mat-header-cell>
      <mat-cell *matCellDef="let element">
        <button mat-button [matMenuTriggerFor]="menu" class="accion">
          <mat-icon>more_vert</mat-icon>
        </button>
          <mat-menu #menu="matMenu">
            <button mat-menu-item >Editar</button>
            <button mat-menu-item >Eliminar</button>
            <button mat-menu-item >Cursos aprobados</button>
            <button mat-menu-item >Asistencia</button>
          </mat-menu>
      </mat-cell>
    </ng-container>

    <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
    <mat-row *matRowDef="let row; columns: displayedColumns;"></mat-row>
  </mat-table>
</div>
```

Este código utiliza el mat-table y sus elementos asociados. Adicionalmente vamos a tener una columna que mostrará un menú popup que nos permitirá realizar acciones sobre un alumno. Para más información: https://material.angular.io/components/table/overview y https://material.angular.io/components/menu/overview

Verá que VSCode marca dos errores, uno en el atributo **datasource** y otro en el atributo **displayedColumns**. Estos atributos corresponden al origen de datos, que será nuestro array de alumnos y el otro a un array con el nombre de las columnas que mostraremos. Vamos a proveer estos valores en forma programática.

- Edite el código del archivo **alumnos-lista.component.ts**
- Agregue el siguiente **import**

```
import { Component, OnInit, Input } from '@angular/core';
import { MatTableDataSource } from '@angular/material';
import { ItemListService } from '../item-list.service';
```

```
import { AlumnosService } from '../alumnos.service';
```

Este import nos proveerá acceso al objeto **datasource**

- Agregue estas dos propiedades

```
export class AlumnosListaComponent implements OnInit {

  alumnos: Alumno[];
  alumnoSeleccionado: Alumno = null;

  dataSource: MatTableDataSource<Alumno>;
  displayedColumns = ['id', 'nombre', 'apellido', 'perfil', 'sexo',
  'activo', 'acciones'];

   constructor(
```

- Luego en la función **OnInit()** agregue la línea que está resaltada

```
  ngOnInit() {
    this.alumnos = this.AlumnosSrv.GetAll();
    this.dataSource = new MatTableDataSource(this.alumnos);
  }
```

Es decir, cada vez que se muestre el componente, asignaremos el datasource con el array de alumnos.

Ahora vamos a recuperar la funcionalidad de filtrar alumnos.

- Modifique la función **Filtrar()** de esta forma

```
  Filtrar(filtro: string) {
    this.alumnos = this.AlumnosSrv.FindbyNombreOApellido(filtro);
    this.dataSource = new MatTableDataSource(this.alumnos);
  }
```

- Guarde los cambios y pruebe la aplicación

6) Editar un alumno de la lista de alumnos activando el componente de edición

Vamos a agregar la funcionalidad de editar la lista de alumnos llamando al componente **alumno-edicion.component** cada vez que se seleccione una fila de la lista de alumnos.

- Agregue la nueva ruta en la configuración del router en el **app.module.ts**:

```
RouterModule.forRoot([
  { path: 'alumnos' , component: AlumnosListaComponent},
  { path: 'cursos' , component: CursosComponent},
  { path: 'asistencias' , component: AsistenciasComponent},
  { path: '',  redirectTo: '/alumnos', pathMatch: 'full' },
  { path: 'alumno/:id' , component: AlumnoEdicionComponent}, ])
```

- Guarde los cambios
- Edite el template del componente **alumnos-lista** que se encuentra en **alumnos-lista.component.html.**
- Agregue en el primer botón de componente **mat-menu** lo siguiente:

```
<ng-container matColumnDef="acciones">
  <mat-header-cell *matHeaderCellDef></mat-header-cell>
  <mat-cell *matCellDef="let element">
    <button mat-button [matMenuTriggerFor]="menu" class="accion">
         <mat-icon>more_vert</mat-icon>
    </button>
      <mat-menu #menu="matMenu">
        <button mat-menu-item (click)="EditarAlumno(element)">
            Editar
        </button>
        <button mat-menu-item >Eliminar</button>
        <button mat-menu-item >Agregar a curso</button>
        <button mat-menu-item >Asistencia</button>
      </mat-menu>
  </mat-cell>
</ng-container>
```

- Guarde los cambios
- Edite el código del componente (**alumnos-lista.component.ts**) y agregue el siguiente **import** para poder navegar a una ruta desde nuestro código:

```
import { Router } from '@angular/router';
```

- Inyecte el Router en el constructor para utilizarlo más tarde, agregue esto en la declaración del constructor:

```
 constructor(
  public ItemListSrv: ItemListService,
  public AlumnosSrv: AlumnosService,
  private router: Router,
) { }
```

- Agregue el siguiente método al componente:

```
EditarAlumno(alumno: Alumno) {
  this.router.navigate(['/alumno', alumno.id]);
}
```

Este es el método que será llamado cuando hagamos un click sobre la opción de menú Editar. Vea como pasamos los parámetros desde la construcción de la vista hasta la navegación de la ruta.

- Guarde los cambios

Ahora prepararemos primero el componente **alumno-edicion** para que pueda trabajar con el parámetro **:id** que viene en la ruta.

- Edite el archivo **alumno-edicion.component.ts**.
- Aquí vamos a utilizar el **Router**, el **ActiveRoute y** el **servicio de alumnos**. Para ello agregue los siguientes imports:

```
import { Router, ActivatedRoute } from '@angular/router';
import { AlumnosService } from '../alumnos.service';
```

- Elimine el **@Input()** de la propiedad **alumnoSeleccionado** y comente el evento FinDeEdicion

```
export class AlumnoEdicionComponent implements OnInit {

  alumnoSeleccionado: Alumno;
  // @Output() FinDeEdicion = new EventEmitter<Operaciones>();
```

- Reemplace el constructor por el siguiente para inyectar el **servicio de alumnos**, el **Router** y el **ActiveRoute** en el componente:

```
  constructor(
    public ItemListSrv: ItemListService,
    private router: Router,
    private activeRoute: ActivatedRoute,
    private alumnosSrv: AlumnosService
  ) { }
```

- Agregue el siguiente código al evento **ngInit()**

```
  ngOnInit() {
    const id = Number( this.activeRoute.snapshot.paramMap.get('id'));
    this.alumnoSeleccionado = this.alumnosSrv.Get(id);
  }
```

Aquí lo que hemos hecho es obtener el parámetro **id** de la url, llamar al servicio de alumnos para obtener el alumno para editar en el template.

- Modifique los métodos **Regresar() y Guardar()** :

```
  Regresar() {
    // this.alumnoSeleccionado = null;
    // this.FinDeEdicion.emit('cancelar');
    this.router.navigate(['/alumnos']);
  }

  Guardar(form: any) {

    console.log(form);
    Object.keys(form).forEach((key, index) =>
     this.alumnoSeleccionado[key] = form[key]
```

```
    );

    if (this.alumnoSeleccionado.id === 0) {
      // this.FinDeEdicion.emit('agregar');
    } else {
      // this.FinDeEdicion.emit('editar');
      this.alumnosSrv.Update(this.alumnoSeleccionado);
      this.Regresar();
    }
  }
```

El método **Regresar** navega a la ruta **/alumnos**. El método **Guardar** llama al servicio de alumnos para guardar los cambios y luego regresa a la lista de alumnos.

- Guarde los cambios y pruebe la aplicación ahora.

7) Desafío: funcionalidad de agregar y eliminar alumnos

El objetivo es modificar el componente **alumno-edicion.component** para que reciba un parámetro adicional que indique si la operación es agregar o modificar.
Recuerde que si deseo utilizar varios parámetros en una ruta debo definirla de esta forma:

```
{ path: 'alumno/:operacion/:id' , component: AlumnoEdicionComponent},
```

Los pasos para realizar esto son:

a) Modificar la ruta en el módulo.

b) Modificar el componente **alumno-edicion.component** para que lea el parámetro de la operación (agregar o editar) y actúe en consonancia con el mismo.

c) Modificar el component **alumnos-lista.component** para que cuando se seleccione un alumno pase el parámetro de operación “editar”.

d) Modificar el mismo componente para que cuando presione el botón de agregar alumno active la ruta correspondiente pasándole al parámetro operación el valor “agregar”.

e) Implemente la eliminación de alumnos que se encuentra en el menú de acciones de la tabla.