Exercises Feedback:  
1. Se debe configurar CORS en Program.cs para permitir la conexion con Angular en la otra direccion:

(…)

// Add services to the container.

builder.Services.AddControllers();  
  
**//Configure CORS to connect to Angular**

**builder.Services.AddCors(options =>**

**{**

**options.AddPolicy("AllowAngularApp", policy =>**

**{**

**policy.WithOrigins("http://localhost:62089") // Your Angular app's URL**

**.AllowAnyHeader()**

**.AllowAnyMethod();**

**});**

**});**

(…)

**app.UseCors("AllowAngularApp");**

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

2. Si necesito inicializar alguna variable que depende por ejemplo de una interfaz que se encuentra en el service.ts (en este caso), debo inicializar esa variable OnInit y no en la declaración de variables, de otra forma tendre un error de redundancia en las dependencias:

export class PatientManagementComponent implements OnInit{

patientsList: IPatient[] = [];

newPatient!: IPatient;

selectedPatient!: IPatient | null;

constructor(private \_patientService: PatientService) { }

ngOnInit() {

**this.newPatient = this.getEmptyPatientObject();**

this.loadPatients();

}

3. Después de instalar bootstrap, es necesario incluir las librerías en ‘angular.json’, de lo contrario no se implementaran los estilos bootstrap, asi:

"styles": [

"src/styles.css",

"node\_modules/bootstrap/dist/css/bootstrap.min.css"

],

"scripts": [

"node\_modules/bootstrap/dist/js/bootstrap.bundle.min.js"

]

4. Para poder implementar **[formGroup]** y **formControlName** en el formulario, es necesario importar **ReactiveFormsModule** en *app.module.ts*

5. El constructor del componente del formulario requiere instanciar el objeto FormGroup, para tomar los valores de los inputs, asi:

patientForm!: FormGroup;

constructor(private \_formBuilder: FormBuilder, private \_patientService: PatientService) {

this.patientForm = this.\_formBuilder.group({

firstName: ['', [Validators.required, Validators.minLength(3)]],

lastName: ['', [Validators.required, Validators.minLength(3)]],

email: ['', [Validators.required, Validators.email]],

healthCardNum: ['', [Validators.required, Validators.minLength(10)]],

gender: [''],

birthDate:['']

});

}

6. Para implementar **routerLink** (SPA) en los links, se debe importar el módulo **RouterLink**

7. La sintaxis para establecer las rutas es la siguiente:

const routes: Routes = [

{ path: "", component: HomeComponent },

{ path: "admissions", component: AdmissionsComponent },

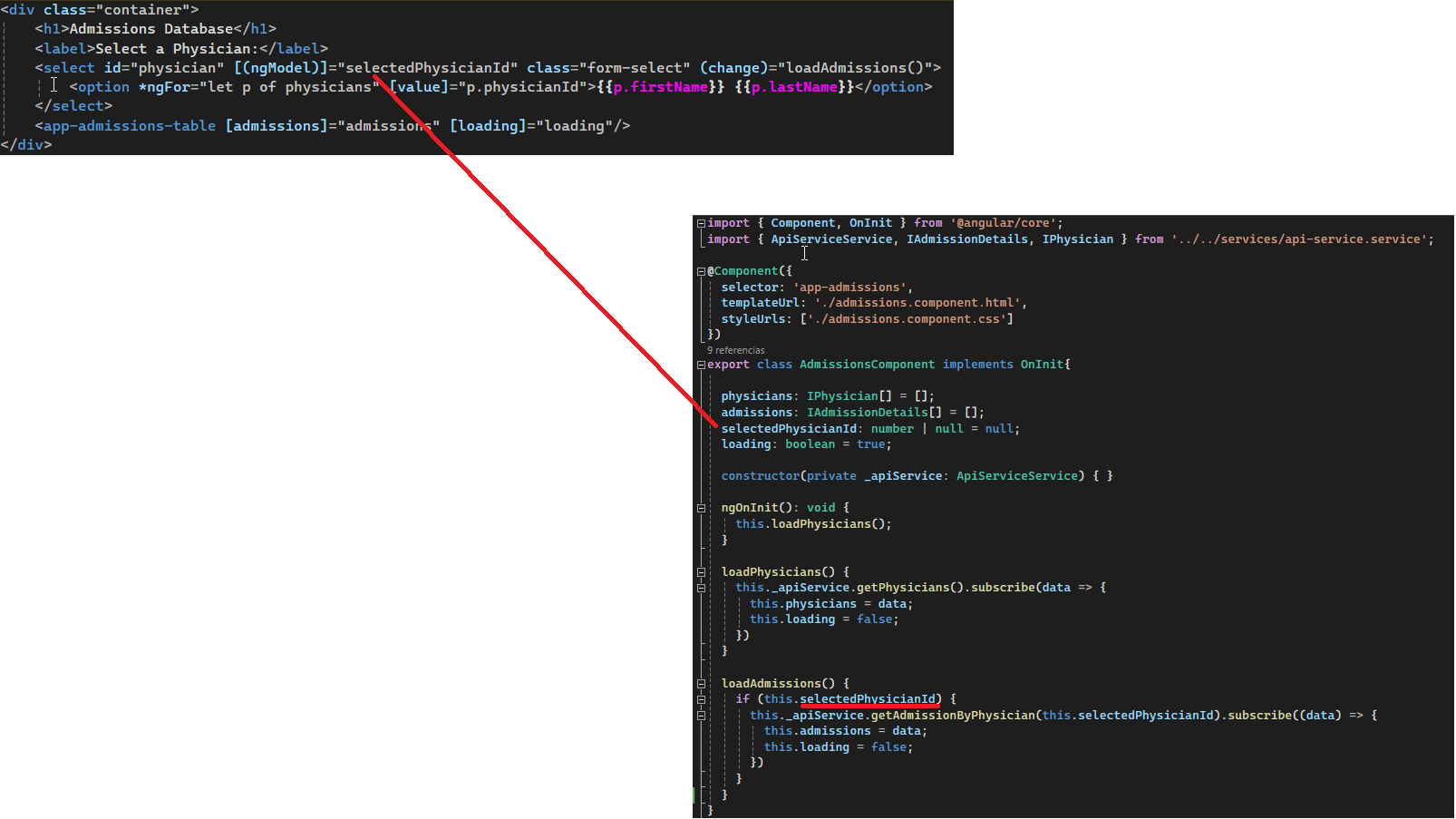
{ path: "\*\*", redirectTo: '', pathMatch: 'full' }

];

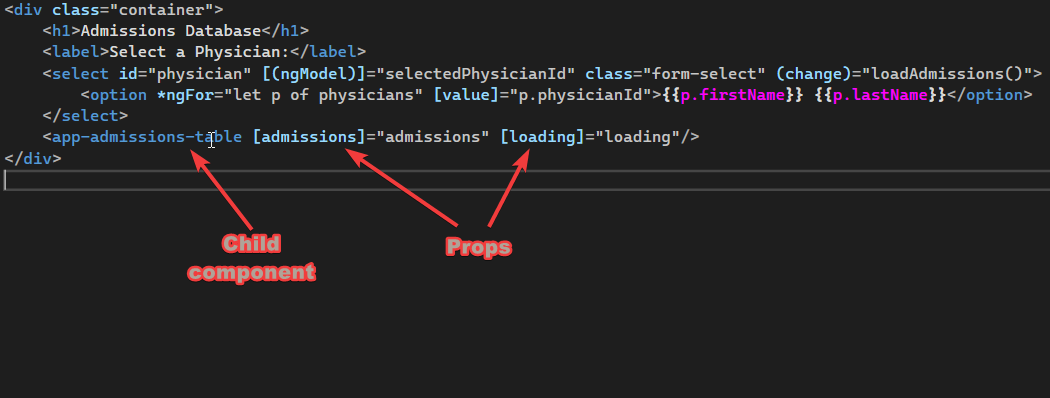
8. Para usar [(ngModel)], como por ejemplo en la siguiente línea de código:  
<select id="physician" [(ngModel)]="selectedPhysicianId" class="form-select" (change)="loadAdmissions()">

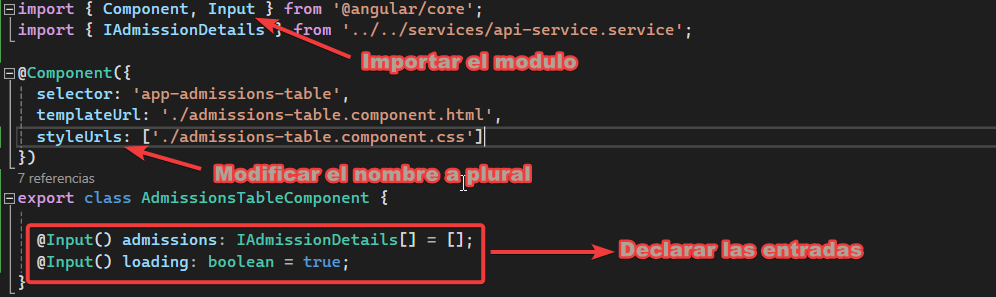
Se debe importar **FormsModule** en *app.module.ts*

*[(ngModel)] se usa para pasar parámetros desde el HTML hacia una variable en el TS (logica). En el caso especifico de este ejemplo el elemento <select> le pasa al TS el valor seleccionado en la lista desplegable y con ese valor se ejecuta un método lara cargar los valores relacionado con ese Physician.*

**

9. Para realizar la inclusión de un componente hijo en un padre, los “props” de seben enviar de la siguiente forma desde el HTML padre:

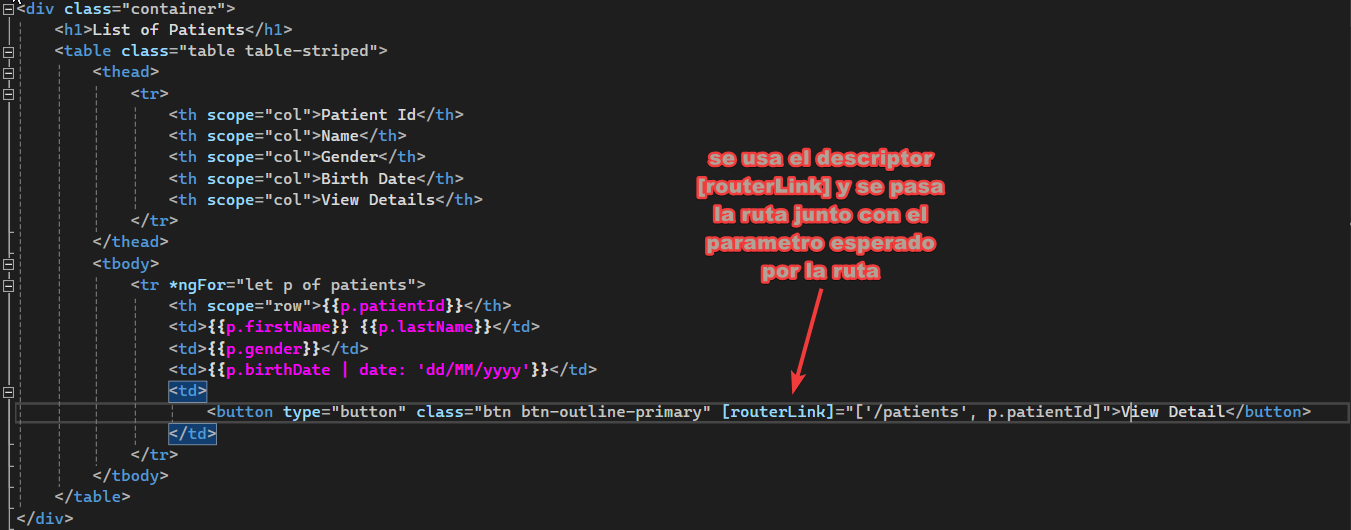


Adicionalmente en el archivo .ts del componente hijo se deben usar los modulos @Input() para recibir los props:  


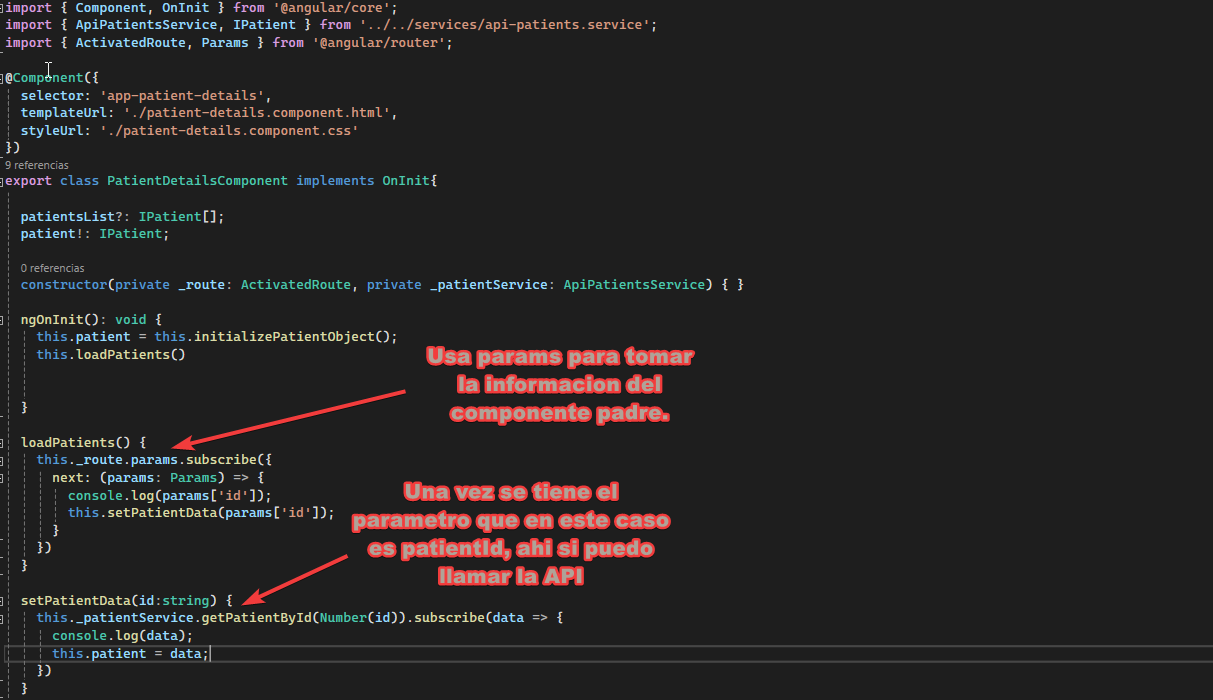
Finalmente, por alguna razón, también debe cambiar el nombre de la propiedad *styleUrl* que por defecto esta en singular, a plural *styleUrls,* tanto en el componente hijo como en el componente padre y hacerlo array, por ese se pone entre llaves cuadradas.

10. Sin embargo, cuando los parámetros se quieren pasar anidando paginas (pages) en lugar del componente hijo explicito dentro del componente padre, esto se debe hacer mediante ***params,*** asi:

componente HTML del padre:



Componente TS de la nueva ruta a la que voy (“hijo”):



10. Establecer parámetros de la base de datos en appsettings.json:

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"**,**

**"ConnectionStrings": {**

**"chdb": "Server=(LocalDB)\\MSSQLLocalDB;Database=chdb;Trusted\_Connection=True"**

**}**

}

11. Scaffols-DbContext

**1. Install-Package Microsoft.EntityFrameworkCore.Tools -Version 7**

**2. Install-Package Microsoft.EntityFrameworkCore.SqlServer -Version 7**

**3. Scaffold-DbContext name=chdb Microsoft.EntityFrameworkCore.SqlServer –OutputDir Models -Tables physicians**

12. Editar Program.cs

// Add services to the container.

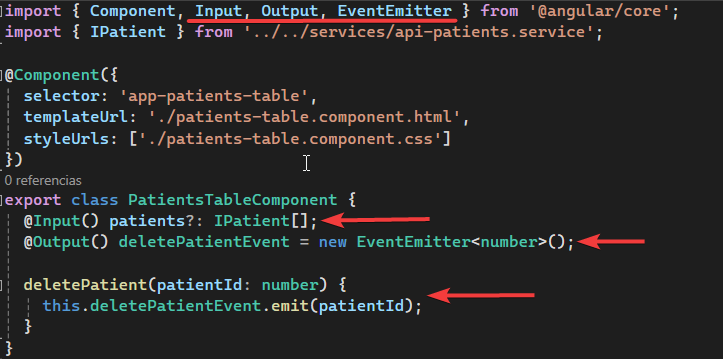
builder.Services.AddControllers();

**builder.Services.AddDbContext<ChdbContext>(options =>**

**options.UseSqlServer(builder.Configuration.GetConnectionString("chdb")));**

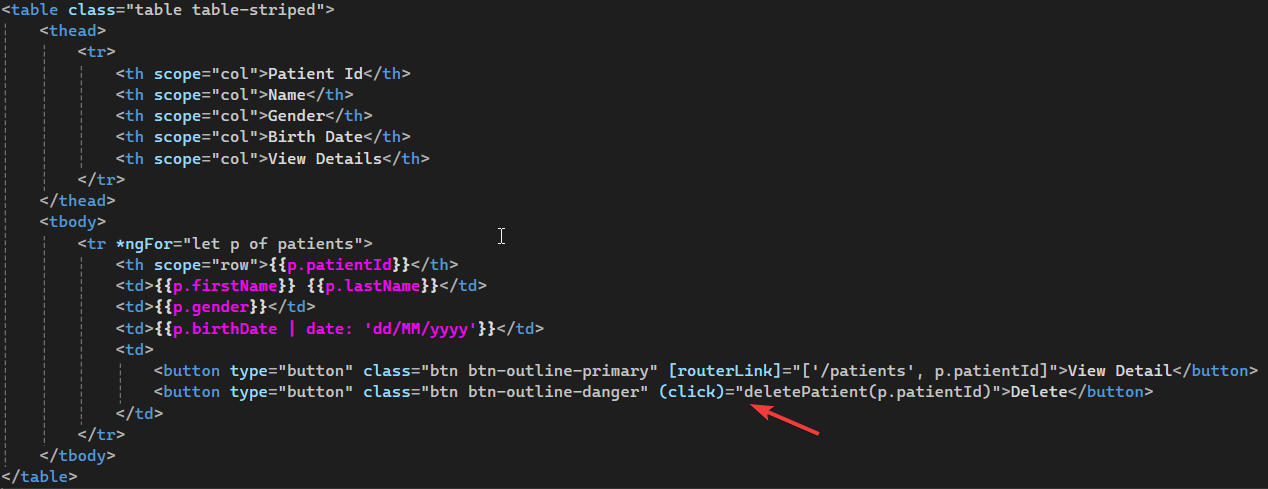
13. Para eliminar registros existentes desde un componente hijo hacia un componente padre que esta manejando la logica, se deben usar propiedades @Input, @Output y EventEmmiter:

*Componente –hijo-*

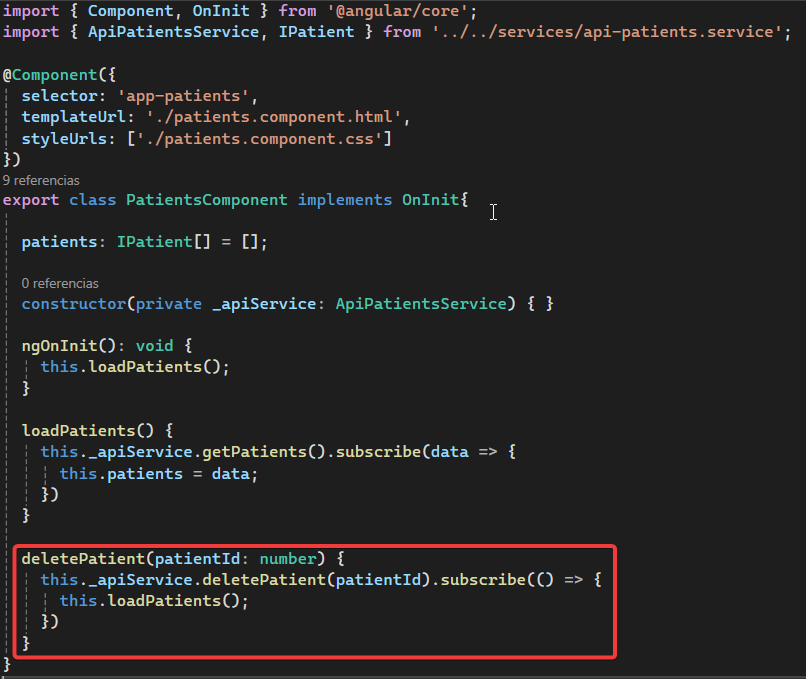


Adicionalmente la conexión entro los componentes se finaliza en los archivos HTML.

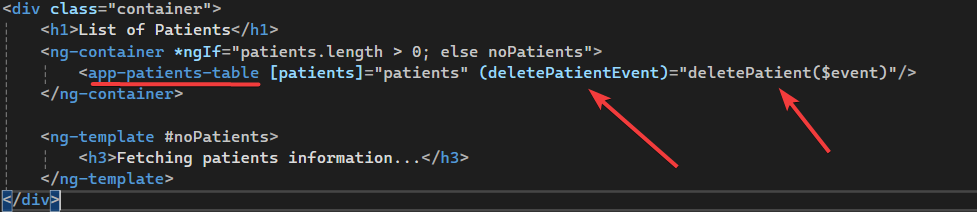
*HTML –hijo-*



*Componente –Padre*



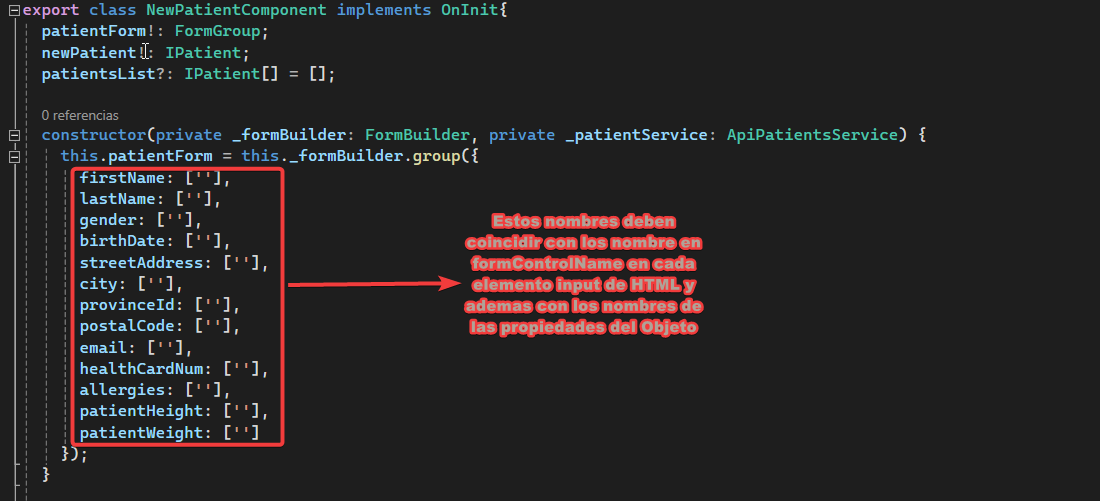
*HTML –padre-*



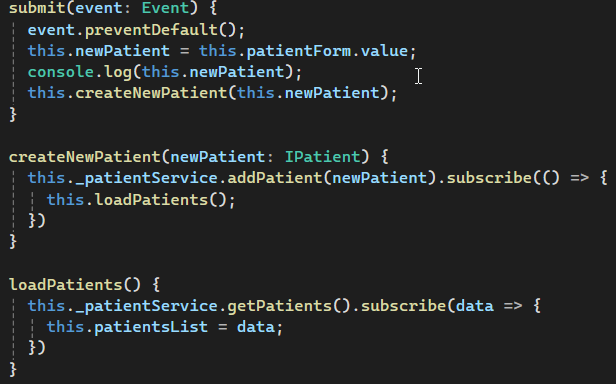
**Contact Form**

1. Para poner a funcionar el formulario tomado de Bootstrap, es necesario importer los modulos:

* ReactiveFormsModule,
* FormsModule



1. La API debe inyectarse en el constructor para generar el método POST en el backend y una vez hecho eso, se recarga la base de datos:



1. En el elemento del formulario **<form>** es necesario llamar la propiedad **[formGroup]** y e iniciar el evento **submit** para prevenir que se accione con el botón antes de terminar el formulario

<form [formGroup]="patientForm" class="row g-3" (submit)="submit($event)">