CREANDO FRONTEND EN ANGULAR

1. Vamos a la carpeta donde queremos tener nuestro repositorio de archivos y damos click derecho en powerShell y generamos el proyecto visual code con las siguientes líneas (tenemos que tener instalado antes angular y sus dependencias).

```
npm
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\Usuario> cd desktop
PS C:\Users\Usuario\desktop> cd webApi
PS C:\Users\Usuario\desktop\webApi> ng new frontEnd --strict=false
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SCSS   [ https://sass-lang.com/documentation/syntax#scss                ]
  Sass   [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less   [ http://lesscss.org                                             ]
```

Al finalizar sobre la misma carpeta del proyecto digitamos “code .”

- Se nos crea el proyecto con archivos y configuraciones. Tener en cuenta :
  El archivo index.hmtl es como la masterPage y tiene una línea llamada <app-root></app-root>, donde queda nuestro componente app.component.html

```
index.html > ...
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>FrontEnd</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

  lo podemos ver en el app.component.ts.

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'frontEnd';
}
```

2. Vamos a crear un componente por la terminal para mostrar los empleados:
   - ng gc empleados/listado-empleados --skip-tests=true -> se crea la siguiente carpeta y archivos dentro de la carpeta app.

```
src
  app
    empleados\listado-empleados
      # listado-empleados.component.css
      <> listado-empleados.component.html
      TS listado-empleados.component.ts
```

- Ahora podemos llamar el componente <app-listado-empleados></app-listado-empleados> dentro del componente app.component.html si deseamos utilizarlo.

```
app.component.html > h2
Go to component
<app-listado-empleados></app-listado-empleados>
<h2>HOLA MUNDO</h2>
```

Y si nuestro componente listado-empleados.component.html tiene

```
listado-empleados.component.html U
src > app > empleados > listado-empleados > listado-empleados.component.html >
Go to component
1  <p>listado-empleados works!</p>
```

Al final nos va a salir en la página web lo siguiente

listado-empleados works!

HOLA MUNDO

3. Para poder usar mas herramientas de visualización en nuestra pagina instalamos material design, vamos al terminal y digitamos
   - ng add @angular/material -> Nos hace preguntas y por ahora damos enter a todo para dejar configuración por defecto.
4. Creamos un modulo dedicado a la importación de los componentes de angular material. Vamos a la terminal y digitamos
   - ng generate module material -> se crea la siguiente carpeta y archivos dentro de la carpeta app.

```
app
  empleados
  material
    material.module.ts
```

   - Vamos al archivo app.modulo.ts y agregamos

```
import {MaterialModule} from './material/material.module'
```

```
imports: [
  BrowserModule,
  AppRoutingModule,
  BrowserAnimationsModule,
  MaterialModule
```

   - Comentamos en style.css la siguiente línea para que no afecte el toolbar que vamos a construir.

```
/* html, body { height: 100%; } */
```

5. Creamos un componente para el toolbar de la aplicación. Vamos a la terminal y digitamos
   - ng g c menú -> se crea la siguiente carpeta y archivos dentro de la carpeta app.

```
menu
  menu.component.css
  menu.component.html
  menu.component.spec.ts
  menu.component.ts
```

6. Vamos a usar el componente <mat-toolbar>,<mat-icon> y <mat-button> para esto tenemos que importarlo en material.module.ts

```
import {MatToolbarModule} from '@angular/material/toolbar'
import {MatIconModule} from '@angular/material/icon'
import {MatButtonModule} from '@angular/material/button'


@NgModule({
  declarations: [],
  exports: [
    MatToolbarModule,
    MatIconModule,
    MatButtonModule
  ],
```

7. Agregamos los componentes anteriores en menu.component.html

```
<mat-toolbar color="primary">
<span>
    <a mat-button>
        <mat-icon>local_library</mat-icon> Empleados
    </a>
</span>
</mat-toolbar>
```

8. Después agregamos el <app-menu> y ya podemos visualizar el menú en nuestra página.

```
<app-menu></app-menu>
<app-listado-empleados></app-listado-empleados>
<h2>HOLA MUNDO</h2>
```

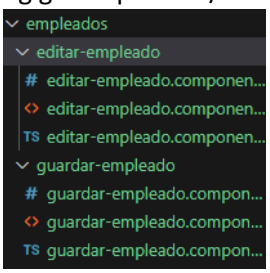
listado-empleados works!

HOLA MUNDO

9. Para que el toolbar siempre sea visible en la parte superior vamos al menu.component.css y agregamos

```
mat-toolbar{
    position: sticky;
    position: -webkit-sticky;
    top: 0;
    z-index: 1000;
}
```

10. Vamos a crear primero nuestros menús manejar los empleados. Ya tenemos listar, ahora vamos a implementar editar y guardar. Vamos al terminal y digitamos
    - ng g c empleados/guardar-empleado --skip-tests=true
    - ng g c empleados/editar-empleado --skip-tests=true



11. Creado los componentes anteriores, ahora vamos a crear el ruteo para cada uno. Vamos al archivo app-routing.module.ts y agregamos

```
const routes: Routes = [
  {  path: '', component: ListadoEmpleadosComponent},
  {  path: 'editar', component: EditarEmpleadoComponent},
  {  path: 'guardar', component: GuardarEmpleadoComponent}
];
```

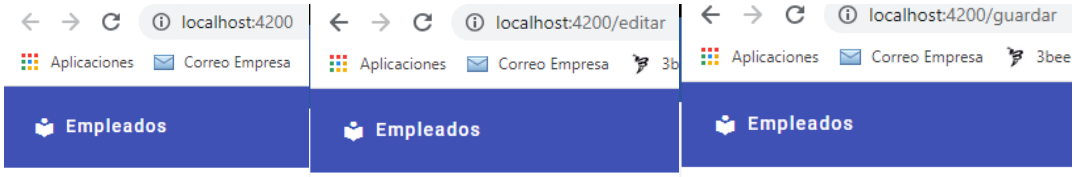
12. Ahora en el componente app.component.html se modifica, se agrega <router-outlet> y queda así

```
<app-menu></app-menu>
<div class="contenedor">
  <router-outlet></router-outlet>
</div>
```

También modificamos el css

```
.contenedor{
    margin: 30px;
}
```

13. Con esto tendremos las siguientes urls



listado-empleados works!

editar-empleado works!

guardar-empleado works!

14. Si queremos navegar a través de nuestros botones o componentes tenemos que usar routerLink

```
<mat-toolbar color="primary">
<span>
    <a mat-button routerLink="">
        <mat-icon>local_library</mat-icon> Empleados
    </a>
</span>
<div>
    <a mat-button routerLink="guardar">Guardar</a>
</div>
<div>
    <a mat-button routerLink="editar">Editar</a>
</div>
</mat-toolbar>
```

15. Existe otro método de navegación por JavaScript. Cuando lo hacemos por un método que creamos a partir de un botón.
    - Creamos un botón en el componente html.

```
<button mat-flat-button color="primary" type="button" (click)="guardarCambios()">
    Crear Empleado
</button>
```

    - Añadimos las siguientes líneas en el componente ts del mismo html anterior.

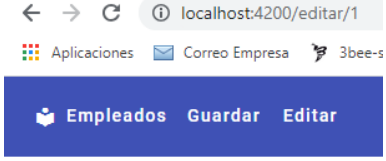
```
constructor(private router: Router){}
```

```
guardarCambios()
{
  this.router.navigate(['/guardar'])
}
```

16. Vamos a realizar un cambio en el componente de editar ya que debe recibir un parámetro para cargar el empleado según su id. En el app-routing.module.ts cambiamos la ruta de editar-empleado.component.html

```
{  path: 'editar/:id', component: EditarEmpleadoComponent},
```

Ahora muestra la pagina editar solo si tiene el id



editar-empleado works!

17. Si queremos leer el id del componente editar, vamos al archivo .ts y agregamos lo siguiente

```
constructor(private activatedRoute: ActivatedRoute){}

ngOnInit(): void {
  this.activatedRoute.params.subscribe(params=>{
    alert(params.id);
  })
}
```

18. La siguiente línea se utiliza para atrapar todas las rutas que se pueden escribir en nuestra url o dominio. Es importante agregarla al final de todas las Routes en el archivo app-routing.module.ts
    - Podemos dejarlo con rediretTo o component. Y se va a dirigir a la ruta establecida en este caso a ListadoEmpleadosComponent

```
{ path: '**', redirectTo: '' }
```

ó

```
{ path: '**', component: ListadoEmpleadosComponent }
```

19. Ahora creamos los textbox para ingresar la información, vamos a usar mat-form-field que es un componente con un buen diseño para que el usuario ingrese la información.
    - Vamos a material.module.ts e insertamos las siguientes librerías:

```
import {MatFormFieldModule} from '@angular/material/form-field'
import {MatInputModule} from '@angular/material/input'
```

    - Vamos al app.module.ts y adicionamos la siguiente línea para trabajar con formularios reactivos

```
import {ReactiveFormsModule} from '@angular/forms'
```

      NOTA: Los dos anteriores es importante ir a las de exports e imports y agregarlos.
    - Vamos al componente guardar-empleado.component.html y agregamos

```
<h2>Guardar Empleado</h2>
<form (submit)="guardarCambios()" [formGroup]="form" >
    <mat-form-field appearance="outline">
        <mat-label>Nombre</mat-label>
        <input formControlName="nombre" matInput/>
    </mat-form-field>
    <mat-form-field appearance="outline">
        <mat-label>Cedula</mat-label>
        <input formControlName="cedula" matInput/>
    </mat-form-field>
    <div>
        <button mat-flat-button color="primary" [disabled]="form.invalid">
            Guardar
        </button>
        <a mat-stroked-button routerLink="">
            Cancelar
        </a>
    </div>
</form>
```

```
button{
    margin-right: 1rem ;
}
mat-form-field {
    width: 100%;
    max-width: 500px ;
}
form{
    max-width: 600px;
}
```

20. Importante en el archivo .ts de este componente adicionar el formgroup y sus valores iniciales

```
export class GuardarEmpleadoComponent implements OnInit  {

  constructor(private router: Router, private formBuilder: FormBuilder){}
  form: FormGroup;
  ngOnInit(): void {
   this.form = this.formBuilder.group({
    nombre: ['', {validators: [Validators.required]}],
    cedula: ['', {validators: [Validators.required]}]
   });
  }
  guardarCambios(){
    this.router.navigate([''])
  }
```

21. Para Mostrar las validaciones tenemos el siguiente ejemplo

```
<input formControlName="nombreU" matInput />
<mat-error *ngIf="this.form.get('nombreU')?.errors?.['required']">Nombre Vacio</mat-error>
<mat-error *ngIf="this.form.get('nombreU')?.errors?.['minlength']">Longitud invalida</mat-erro
```

```
ngOnInit(): void {
 this.form = this.formBuilder.group({
  nombreU: ['', {validators: [Validators.required,Validators.minLength(2)]}]
 })
```

NOTA: En las dos ultimas imágenes creamos validaciones para que sea necesario ingresar texto en el campo nombre y cedula, si esto no ocurre el botón de guardar tiene una validación de los campos del formulario y lo deja inhabilitado hasta que se cumpla las validaciones de los mat-form-field.

22. Si deseamos sacar mensajes del error o validaciones personalizadas ver video 49 y 50.
23. Vamos a trabajar con el componente @input este se usa para enviar información de un componente padre a un componente hijo. En este caso enviamos una string de app.component a guardar-empleado.component
    - En los html hacemos

app.component.html M

```
src > app > app.component.html > ...
      Go to component
1     <app-menu></app-menu>
2     <app-guardar-empleado [entrada]="'solo'"></app-guardar-empleado>
3
```

guardar-empleado.component.html U

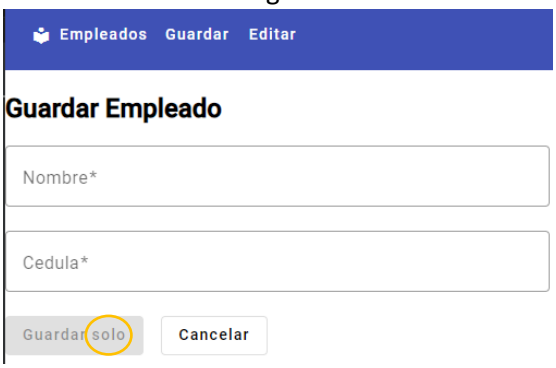
```
src > app > empleados > guardar-empleado > guardar-empleado.component.html > ...
      Go to component
1   <h2>Guardar Empleado</h2>
2   <form (submit)="guardarCambios()" [formGroup]="form" >
3       <mat-form-field appearance="outline">
4           <mat-label>Nombre</mat-label>
5           <input formControlName="nombre" matInput/>
6       </mat-form-field>
7       <mat-form-field appearance="outline">
8           <mat-label>Cedula</mat-label>
9           <input formControlName="cedula" matInput/>
10      </mat-form-field>
11      <div>
12          <button mat-flat-button color="primary" [disabled]="form.invalid">
13              Guardar {{entrada}}
14          </button>
15          <a mat-stroked-button routerLink="">
16              Cancelar
17          </a>
18      </div>
19  </form>
```

    - En el archivo .ts

guardar-empleado.component.ts U

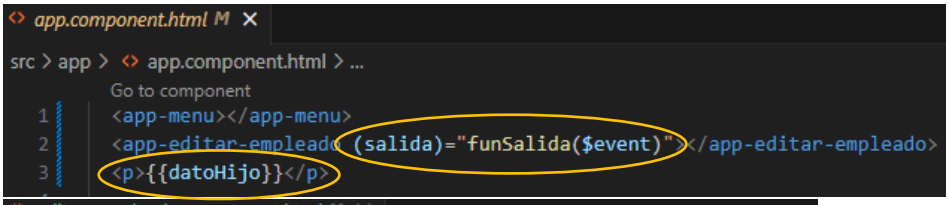
```
src > app > empleados > guardar-empleado > guardar-empleado.component.ts > ...
10  export class GuardarEmpleadoComponent implements OnInit  {
11
12    constructor(private router: Router, private formBuilder: FormBuilder){}
13    @Input()
14    entrada;
```

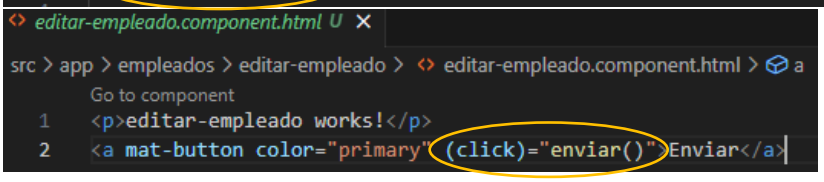
El resultado de esta lógica seria



24. Vamos a trabajar con el componente @output este se usa para enviar información de un componente hijo a un componente padre. En este caso enviamos una string de editar-empleado.component a app.component

- En los html hacemos





- En el archivo app.component.ts

```
export class AppComponent {
  title = 'frontEnd';

  datoHijo: string = "Sin dato"
  funSalida(e){
    this.datoHijo = e;
  }
}
```

En el archivo editar-empleado.component.ts

```
@Output()
salida = new EventEmitter();
```

```
enviar(){
  this.salida.emit("Dato Salida");
}
```

El resultado de esta lógica seria

CREANDO EL CRUD

NOTA: IR AL MANUAL CONFIGURACION DE VISUAL CODE Y VISUAL STUDIO PARA CONECTAR BACKEND CON FRONTEND, AQUÍ ENCONTRAMOS LA CONFIGURACIÓN CORS, CREACION DE ENVIRONMENTS, DTOS Y HTTP METODOS

25. Vamos al terminal y creamos empleados.service.ts como en el manual de la **NOTA.**
26. Debemos importar HttpClientModule como en el manual de la **NOTA.**
27. Creamos las carpetas y archivos environment. Y configuramos las url. como en el manual de la **NOTA.**
28. Vamos al terminal y creamos empleados.ts como en el manual de la **NOTA.**
29. Vamos al archivo empleados.service.ts y deben estar los métodos nombrados en el manual de la **NOTA.**
30. Vamos al archivo empleados. ts y deben estar los DTOs nombrados en el manual de la **NOTA.**
31. Vamos a los archivos environment.ts y debe estar la configuración nombrada en el manual de la **NOTA.**

32. Vamos a trabajar con el formulario de listar los empleados, para esto vamos a trabajar con tablas que visualizan los empleados
    - Vamos a usar el componente <table mat-table> para esto tenemos que importarlo en material.module.ts

```
import {MatTableModule} from '@angular/material/table'
exports: [
  MatTableModule
```

33. También vamos a usar un componente adicional para mostrar un popup de confirmación para borrar un empleado. Por lo tanto
    - Vamos a instalar por medio de la terminal
      npm install --save sweetalert2 @sweetalert2/ngx-sweetalert2
    - Vamos al archivo app.module.ts e importamos

```
import{SweetAlert2Module} from '@sweetalert2/ngx-sweetalert2'
  imports: [
    SweetAlert2Module.forRoot()
```

34. Creamos todo el código html para el componente de listar los empleados.

```
<h3>Empleados</h3>
<button mat-flat-button color="primary" type="button" (click)="guardarCambios()">
    Crear Empleado
</button>
<ng-container contenido>
    <table #table mat-table [dataSource]="empleados" class="mat-elevation-z8">
        <ng-container matColumnDef="id">
            <th mat-header-cell *matHeaderCellDef>Id</th>
            <td mat-cell *matCellDef="let element">{{element.id}}</td>
        </ng-container>
        <ng-container matColumnDef="nombre">
            <th mat-header-cell *matHeaderCellDef>Nombre</th>
            <td mat-cell *matCellDef="let element">{{element.nombre}}</td>
        </ng-container>
        <ng-container matColumnDef="cedula">
            <th mat-header-cell *matHeaderCellDef>Cedula</th>
            <td mat-cell *matCellDef="let element">{{element.cedula}}</td>
        </ng-container>
        <ng-container matColumnDef="acciones">
            <th mat-header-cell *matHeaderCellDef>Acciones</th>
            <td mat-cell *matCellDef="let element">
              <a routerLink="/editar/{{element.id}}" mat-flat-button color="primary">Editar</a>
              <button mat-flat-button color="warn"
              [swal]="{title: 'Confirmación', text: 'Seguro desea borrar el registro?', showCancelButton: true}"
              (confirm)="borrar(element.id)">Borrar</button>
            </td>
          </ng-container>
        <tr mat-header-row *matHeaderRowDef="columnasAMostrar"></tr>
    <tr mat-row *matRowDef="let row; columns: columnasAMostrar"></tr>
    </table>
</ng-container>
```

35. Creamos la lógica para el funcionamiente del html anterior en el archivo .ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { EmpleadosService } from '../empleados.service';
import { empleadosDTO } from '../empleados';

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

  constructor(private router: Router, private empleadosService: EmpleadosService){}

  empleados: empleadosDTO[];
  columnasAMostrar = ['id','nombre','cedula','acciones']

  ngOnInit(): void {
    this.cargarRegistros();
  }
  cargarRegistros()
  {
    this.empleadosService.obtenerTodos().subscribe(empleados=>{
      console.log(empleados);
      this.empleados = empleados;
    })
  }
  guardarCambios()
  {
    this.router.navigate(['/guardar'])
  }
  borrar(id:number){
    this.empleadosService.borrar(id).subscribe(()=>{
      this.cargarRegistros();
    })
  }
}
```

36. Vamos a trabajar con el formulario de Guardar los empleados.
    - Creamos todo el código html para el componente de guardar los empleados.

```
<h2>Guardar Empleado</h2>
<form [formGroup]="form" >
    <mat-form-field appearance="outline">
        <mat-label>Nombre</mat-label>
        <input formControlName="nombre" matInput/>
    </mat-form-field>
    <mat-form-field appearance="outline">
        <mat-label>Cedula</mat-label>
        <input formControlName="cedula" matInput/>
    </mat-form-field>
    <div>
        <button mat-flat-button color="primary" [disabled]="form.invalid" (click)="submit()">
            Guardar
        </button>
        <a mat-stroked-button routerLink="">
            Cancelar
        </a>
    </div>
</form>
```

    - Creamos la lógica para el funcionamiente del html anterior en el archivo .ts

```
import { Component, OnInit , Input} from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { empleadosCreacionDTO } from '../empleados';
import { EmpleadosService } from '../empleados.service';
import { delay } from 'rxjs/operators';

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

  constructor(private router: Router, private formBuilder: FormBuilder,
    private empleadoService: EmpleadosService ){}
  form: FormGroup;
  ngOnInit(): void {
   this.form = this.formBuilder.group({
    nombre: ['', {validators: [Validators.required]}],
    cedula: ['', {validators: [Validators.required, Validators.maxLength(12)]}]
   });
  }
  submit()
  {
    this.guardarCambios(this.form.value);
  }
  guardarCambios(empleado: empleadosCreacionDTO){
    this.empleadoService.crear(empleado).subscribe(()=>{
      this.router.navigate(['']);
    });
  }
}
```

37. Vamos a trabajar con el formulario de Editar los empleados.
    - Creamos todo el código html para el componente de editar los empleados.

```
<h2>Editar Empleado</h2>
<form [formGroup]="form" >
    <mat-form-field appearance="outline">
        <mat-label>Id</mat-label>
        <input formControlName="id" matInput/>
    </mat-form-field>
    <mat-form-field appearance="outline">
        <mat-label>Nombre</mat-label>
        <input formControlName="nombre" matInput/>
    </mat-form-field>
    <mat-form-field appearance="outline">
        <mat-label>Cedula</mat-label>
        <input formControlName="cedula" matInput/>
    </mat-form-field>
    <div>
        <button mat-flat-button color="primary" [disabled]="form.invalid" (click)="submit()">
            Guardar
        </button>
        <a mat-stroked-button routerLink="">
            Cancelar
        </a>
    </div>
</form>
```

    - Creamos la lógica para el funcionamiente del html anterior en el archivo .ts

```
export class EditarEmpleadoComponent implements OnInit {

  constructor(private activatedRoute: ActivatedRoute ,private router: Router,
    private formBuilder: FormBuilder,
    private empleadoService:  EmpleadosService){}

    form: FormGroup;
    modeloEmpleado: empleadosDTO;

  ngOnInit(): void {

    this.form = this.formBuilder.group({
      id: '',
      nombre: '',
      cedula: ''
    })
    this.activatedRoute.params.subscribe(params=>{
      this.empleadoService.obtenerPorId(params.id).subscribe(empleado=>{
        this.modeloEmpleado = empleado;
        if(this.modeloEmpleado!== undefined){
          this.form.patchValue(this.modeloEmpleado);
        }
      }, ()=> this.router.navigate(['']))
    });
  }
  submit()
  {
    this.guardarCambios(this.form.value);
  }
  guardarCambios(empleado: empleadosCreacionDTO){
    this.empleadoService.editar(this.modeloEmpleado.id,empleado).subscribe(()=>{
      this.router.navigate(['']);
    });
  }
}
```

38. Ejemplo de guardar consumiendo un webAPI y mostrando el mensaje

```
guardarCambios(comercio: comercioCreacionDTO) {
  this.comercioService.crear(comercio).subscribe(
    (respuesta) => {
      Swal.fire({
        title: '¡Éxito!',
        text: 'Se ha completado la operación correctamente',
        icon: 'success',
        showCancelButton: false,
        confirmButtonText: 'Aceptar'
      }).then((result) => {
        if (result.isConfirmed) {
          this.router.navigate(['comercio/listar']);
        }
      });

    },
    (error) => console.log(error)
  );
}
```

39. Ejemplo de utilizar paginación:

```
  <ng-container contenido>
    <table #table mat-table [dataSource]="dataSource" class="mat-elevation-z8">
      <ng-container matColumnDef="fechaTurno">
        <th mat-header-cell *matHeaderCellDef>Fecha Turno</th>
        <td mat-cell *matCellDef="let element">{{ element.fechaTurno | date }}</td>
      </ng-container>
      <ng-container matColumnDef="horaInicio">
        <th mat-header-cell *matHeaderCellDef>Hora Inicio</th>
        <td mat-cell *matCellDef="let element">{{ element.horaInicio }}</td>
      </ng-container>
      <ng-container matColumnDef="horaFin">
        <th mat-header-cell *matHeaderCellDef>Hora Fin</th>
        <td mat-cell *matCellDef="let element">{{ element.horaFin }}</td>
      </ng-container>
      <ng-container matColumnDef="estado">
        <th mat-header-cell *matHeaderCellDef>Estado</th>
        <td mat-cell *matCellDef="let element">{{ element.estado }}</td>
      </ng-container>
      <tr mat-header-row *matHeaderRowDef="columnas"></tr>
      <tr mat-row *matRowDef="let row; columns: columnas"></tr>
    </table>
    <mat-paginator #paginator [pageSizeOptions]="[7]"
showFirstLastButtons></mat-paginator>
  </ng-container>
```

```
@ViewChild('paginator') paginator: MatPaginator;
dataSource: MatTableDataSource<turnoDTO>;
```

```
cargarRegistrosTurnos(id: number) {
  this.turnoService.listarTurnos(id).subscribe((turnos) => {
    this.modeloTurno = turnos;
    this.dataSource = new MatTableDataSource(this.modeloTurno);
    this.dataSource.paginator = this.paginator;
  });
}
```

40. Librerías básicas

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import {ReactiveFormsModule} from '@angular/forms'
import {MatToolbarModule} from '@angular/material/toolbar'
import {MatIconModule} from '@angular/material/icon'
import {MatButtonModule} from '@angular/material/button'
import {MatDatepickerModule} from '@angular/material/datepicker'
import {MatFormFieldModule} from '@angular/material/form-field'
import {MatInputModule} from '@angular/material/input'
import {MatNativeDateModule} from '@angular/material/core'
import {MatTableModule} from '@angular/material/table'
import {MatSelectModule} from '@angular/material/select'
import {MatPaginatorModule} from '@angular/material/paginator'
```