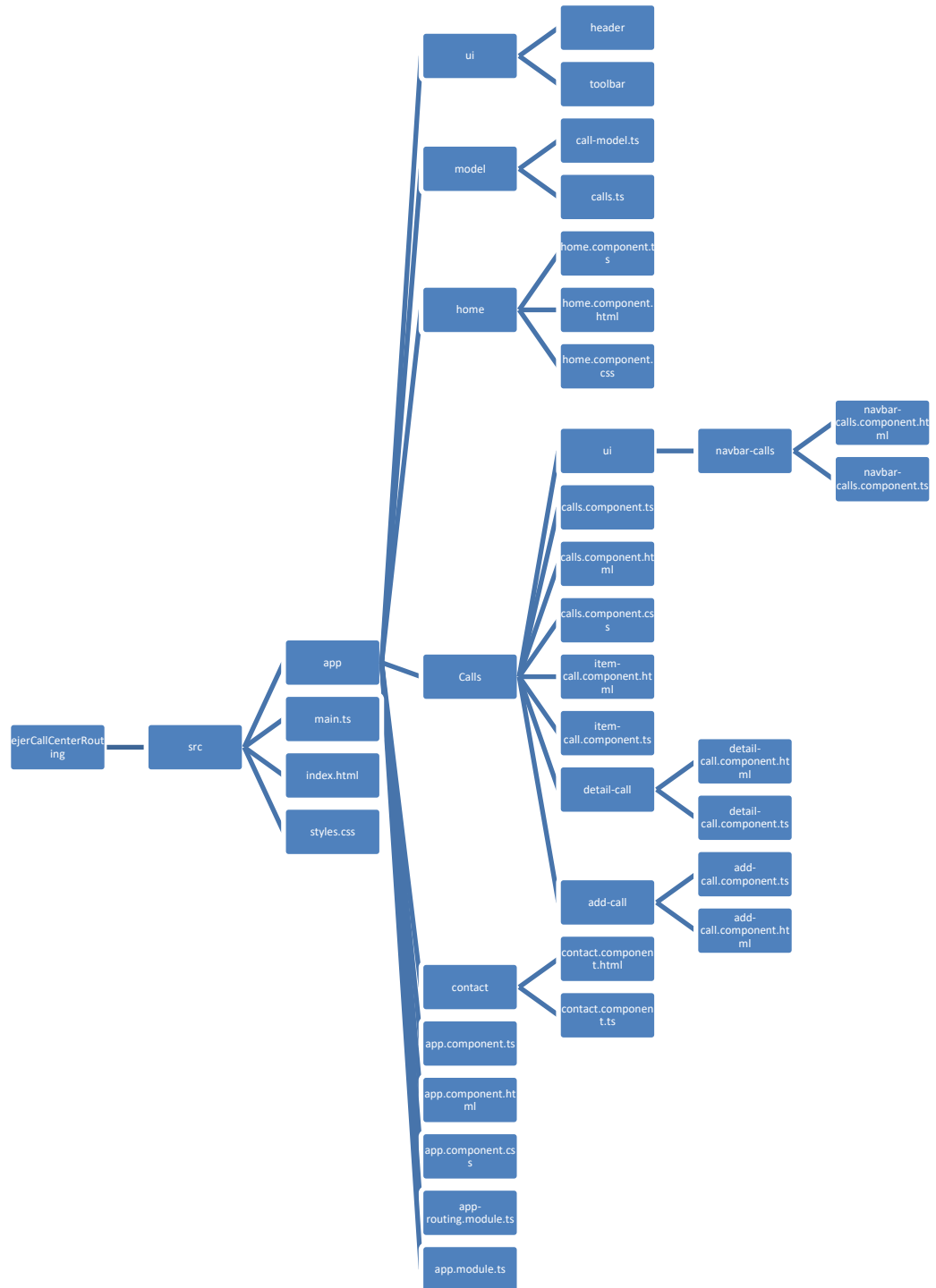


Vamos a desarrollar una WebApp para Gestión de un CallCenter de productos software:

1. Estructura de proyecto



2. Crear la aplicación:

```
ng new ejerCallCenterRouting
```

Cuando nos pregunte:

1) ¿Quieres añadir routing? Respuesta="Sí".

Al indicar ROUTING, ya me crea el módulo de routing: app-routing.module.ts con los imports RouterModule y Routes ya hechos.

2) ¿Tipo de hojas de estilo? Respuesta="CSS"

Es el tipo de hojas de estilo que habéis visto en 1º de DAM.

3. Comprobar en index.html que está <base href>

4. Instalar BootStrap

La primera tarea en este ejercicio consiste en instalar **Bootstrap 4 CSS**.

Para ello, hay que *importar la hoja de estilos de Bootstrap en nuestro fichero "style.css" del proyecto* que está en la carpeta raíz *src*. Escribimos en ese fichero "style.css":

```
@import  
'https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css';
```

A continuación ya podemos instalar **NgBootstrap** en el proyecto. Para ello, en la terminal, desde la carpeta del proyecto, tecleamos:

```
> npm install --save @ng-bootstrap/ng-bootstrap
```

Una vez instalado, se debe **configurar dentro del fichero "app.module.ts" importando el módulo "NgbModule"** con el siguiente código:

```
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';
...
@NgModule({
  declarations: [
    ...
  ],
  imports: [
    ...
    NgbModule
  ],
  providers: [
    ...
  ],
  bootstrap: [ AppComponent ]
})

export class AppModule { }
```

Además de hacer esto, **si planeamos usar formularios**, tendremos que importar también dos módulos más que son: ***FormsModule y ReactiveFormsModule desde '@angular/forms'***. Es decir que pondríamos algo como:

```
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
```

A parte de hacer lo anterior, **en cada componente (xxx.component.ts) donde queramos usar un componente de NgbBootstrap debemos importar NgbModule con la siguiente sintaxis:**

```
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';
```

Además, como vamos a usar **ICONOS**, debemos **instalar la fuente FONT-AWESOME** que contiene los iconos para Bootstrap. Para ello, en la terminal, tecleamos:

```
npm install font-awesome --save
```

Ahora vamos al fichero **angular.json** y configuramos en el apartado “**styles**” las fuentes **Font-awesome**:

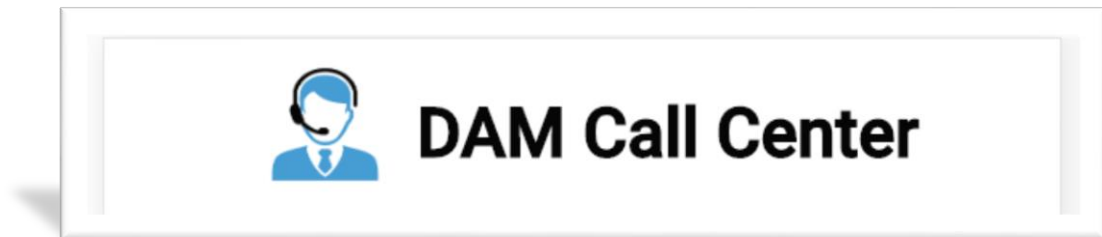
```
"styles": [
  "node_modules/font-awesome/css/font-awesome.css",
  "src/styles.css"
],
```

5. Creamos nuestro componente header

Será la cabecera principal de la aplicación.

Contiene el logo y el nombre de la empresa

ng g c ui/header



header.component.html

```
<div class="container">
  <h1 class="col-12">
    
    {{title}}
  </h1>
</div>
```

header.component.ts

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

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

  title: string;
  constructor() {
    this.title = 'DAM Call Center';
  }

  ngOnInit() {
  }

}
```

Y para que:

- el **tamaño de la letra del título se adapte cuando se reduzca la pantalla** (usamos propiedad “vw” de CSS3),
- el **tamaño de la imagen se adapte a su contenedor <div>** (usamos **ancho 100%**),
- añadir márgenes

Añadiremos reglas CSS al fichero **header.component.css**:

```
h1{
  font-size: 7vw; /* vw (% del ancho del viewport o zona de
visualización)*/
  margin: 2% auto;
}
```

Y para que nuestro componente se muestre, añadimos en **app.component.html**:

```
<app-header></app-header>
```

6. Creamos el componente barra de navegación (navBar)

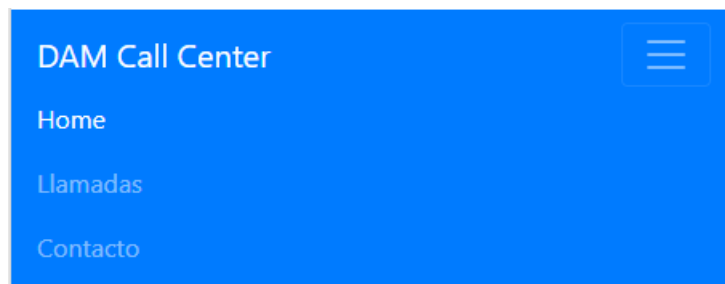
Será la barra de navegación de nuestra app, nuestro menú.

ng g c ui/navbar

Cerrada en un dispositivo pequeño se muestra así:



Abierta en un dispositivo pequeño se muestra así:



En un dispositivo más grande, el botón hamburguesa se oculta y las opciones que contiene se muestran sueltas a continuación del título. Sería así:

¿Cómo hacemos entonces nuestra toolbar con el botón hamburguesa?

Vamos a ello:

- 1) Vamos a la web de Bootstrap y copiamos el ejemplo de navbar y lo adaptamos así:

navbar.component.html

```
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
  <a class="navbar-brand" routerLink="/home">{{title}}</a>
  <button (click)="toggleCollapse()" class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent"
[class.show]="show">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" routerLink="/home"
routerLinkActivated="active">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" routerLink="/calls"
routerLinkActivated="active">Llamadas</a>
      </li>

      <li class="nav-item">
        <a class="nav-link" routerLink="/contact"
routerLinkActivated="active">Contacto</a>
      </li>
    </ul>

  </div>
</nav>
```

navbar.component.ts

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

@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.css']
})
export class NavbarComponent implements OnInit {
  title = 'DAM Call Center';
  show: boolean = false;

  constructor() { }

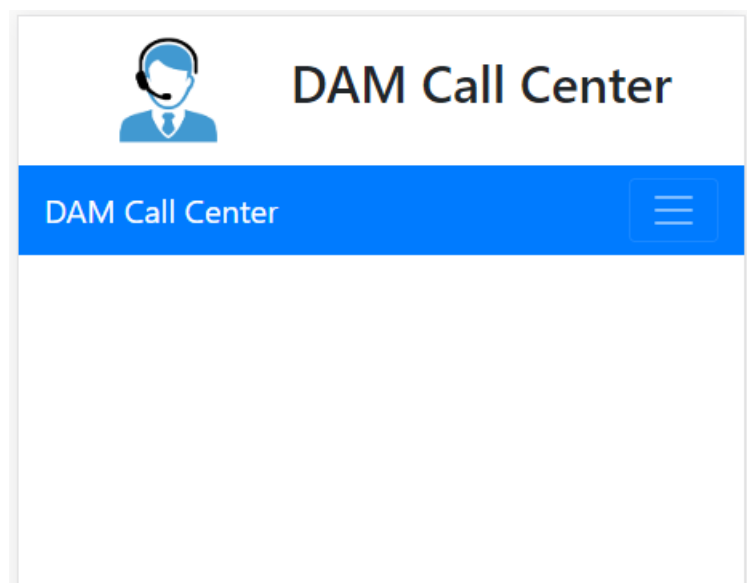
  ngOnInit() {
  }

  toggleCollapse() {
    this.show = !this.show;
  }
}
```

- 2) Y para que nuestro componente se muestre, añadimos debajo del <app-header>, en **app.component.html**:




```
<app-navbar></app-navbar>
```

Si lo probáis, veréis que funciona abierto y cerrado, pero las opciones no nos llevan a ningún sitio aún:



- 3) Aprovechando el **ROUTING DE ANGULAR** para navegar, hemos añadido los enlaces hacia las rutas, aunque al no tener los componentes, no podremos navegar todavía hacia ellos. Pero ya lo dejamos preparado para cuando los creamos.

Para ello, las rutas de navegación que hemos usado son:

-  '/home' → para ir al componente home
-  '/calls' → para ir al componente calls
-  '/contact' → para ir al componente contact

7. Creamos nuestro modelo de datos: llamada y llamadas

En la práctica real, el modelo de datos suele ser un SERVICIO que incluye todos los métodos para gestionar dicho modelo: extraer datos, añadir, modificar,...

De momento lo haremos sin servicios, lo haremos con clases.

Necesitamos:

- Una clase Call: que represente una llamada y la gestione.
- Una clase Calls, que contendrá un array de call, así como los métodos para gestionarlas: añadir, obtener una o varias llamadas, borrar,...

7.1.-call.model

ng g class model/call-model

call-model.ts

```
export class CallModel {
  static numCalls: number = 0;
  idCall: number;
  codIncidencia: string; // formado por las 3 primeras letras del software
  que la
  // provocó, seguido de #, seguido del id de llamada

  constructor(
    public descripcion: string,
    public softwareIncidencia: string,
    public imageUrl: string,
```



```

        public empresaDptoProyecto: string[],
        public telefonoOrigem: string,
        public estado: string,
        public operadorAtendio: string,
        public fechaIncidencia: Date
    ) {
        CallModel.numCalls++;
        this.idCall = CallModel.numCalls;
        this.codIncidencia = this.softwareIncidencia.slice(0, 3) + '#' +
this.idCall;
        console.log(this.codIncidencia);

    }

    /* empresaDptoProyecto=Es un array que incluye
    empresa/departamento/proyecto del software que
    generó la incidencia
    */
}

```

7.2.-calls

ng g class model/calls

calls.ts

```

import { CallModel } from './call-model';

export class Calls {
    // Array de call
    private static CALLS: CallModel[] = [
        new CallModel('fallo al realizar copia seguridad', 'BIBLIOTECH',
            '../assets/images/backup-error.png',
            ['Capgemini', 'Angular', 'Libreria'], '985674747', 'en tránsito', 'Ana María',
            new Date('22/01/2019')),
        new CallModel('fallo al imprimir informes finales', 'CONTATOTAL',
            '../assets/images/printer-error.png',
            ['DXC', 'Consultoría', 'Caja Free'], '985111111', 'en tránsito', 'Víctor',
            new Date('2/11/2018')),
        new CallModel('fallo al mostrar el listado de libros', 'BIBLIOTECH',
            '../assets/images/book-error.png',

```

```

        ['Alta TECH', 'International', 'Libreria'], '985224455', 'en tránsito',
        'Esther',
        new Date('11/3/2018'))

];

/** Método que devuelve un array con las llamadas */
static getCalls(): CallModel[] {
    return this.CALLS;
}

/** Método que añade una llamada que se le pasa */
static addCall(call: CallModel): void {
    this.CALLS.push(call);
}

/** Método que obtiene una llamada cuyo codIncidencia se le pasa */
static getCall(codIncidencia: string): CallModel {
    return this.CALLS.find(call => call.codIncidencia === codIncidencia);
}

/** Método que borra una llamada cuyo codIncidencia se le pasa */
static deleteCall(codIncidencia: string): void {
    let callEncontrada = this.CALLS.find(call => call.codIncidencia === codIncidencia);
    let indiceCall = this.CALLS.indexOf(callEncontrada);
    this.CALLS.splice(indiceCall, 1);
}

/** Método que eleva una incidencia cuyo codIncidencia se le pasa */
static upCall(codIncidencia: string): void {
    let indexCallEncontrada = this.CALLS.findIndex(call => call.codIncidencia ===
codIncidencia);
    this.CALLS[indexCallEncontrada].estado = 'elevada';
}

/** Método que cierra una incidencia cuyo codIncidencia se le pasa */
static closeCall(codIncidencia: string): void {
    let indexCallEncontrada = this.CALLS.findIndex(call => call.codIncidencia ===
codIncidencia);
    this.CALLS[indexCallEncontrada].estado = 'cerrada';
}
}

```

8. Vamos a crear los **COMPONENTES** que aparecen en el menú de navegación:

8.1.-HomeComponent

Simplemente mostrará información de la empresa.

Lo creamos con el comando:

```
ng g c home
```

Lo que mostrará será algo como:

¿A qué nos dedicamos?

DAM Call Center inició su actividad en el sector de los Call Center en el año 2000, configurándose como Centro Especializado en Software, lo que la ha convertido en una referencia dentro del sector por lograr mantener el equilibrio entre los valores económicos y los servicios al cliente.

home.component.html

```
<h3>¿A qué nos dedicamos?</h3>
<hr>
<p>DAM Call Center inició su actividad en el sector de los Call Center
    en el año 2000, configurándose como Centro Especializado en Software,
    lo que la ha convertido en una referencia dentro del sector por lograr
    mantener el equilibrio entre los valores económicos y los servicios
    al cliente.
</p>
```

home.component.ts

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
```

```
export class HomeComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

home.component.css

```
p{
  margin:5% auto;
  max-width: 92%;
  text-align: justify;
}

h3{
  font-size: 5.7vw;
}
```

8.2.-Añadimos el componente HomeComponent al módulo de Routing

En nuestro fichero de configuración de routing: “**app-routing.module.ts**”, tenemos que:

- 1º) Importar el nuevo componente HomeComponent para poder usarlo
- 2º) Añadir una ruta (que coincida con la que hemos puesto en la barra de navegación para home) en el array de rutas.

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

// componentes
import {HomeComponent} from '../home/home.component';

// rutas
const routes: Routes = [
  {path: '', component: HomeComponent, pathMatch: 'full' },
  {path: 'home', component: HomeComponent}
];
```

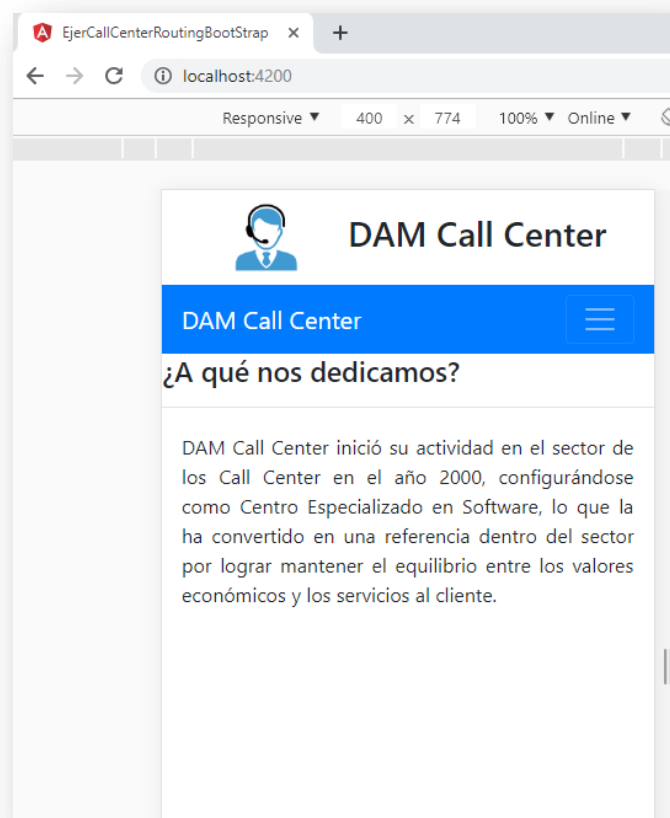
```
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

8.3.-Añadimos a la plantilla del AppComponent la directiva <router-outlet> para que nos cargue los componentes enrutados

En nuestro fichero del componente principal: **app.component.html** añadimos:

```
<router-outlet></router-outlet>
```

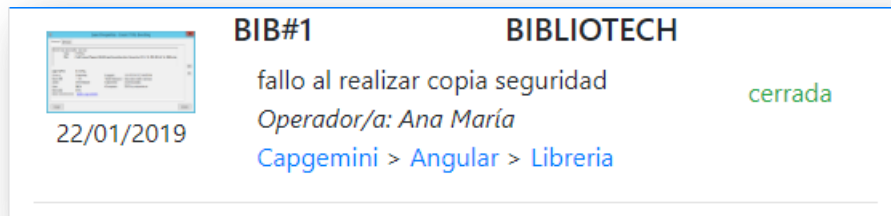
Si lo probamos, veremos:



8.4.-ItemCallComponent

Mostrará los datos abreviados de una llamada.

Su aspecto visual será:



Donde:

- a) En la parte izquierda vemos la imagen del error y debajo la fecha de la incidencia.
- b) En el centro: el código de incidencia, el software que lo provocó, la descripción, el operador que atendió la llamada, y la empresa>departamento>proyecto en el que se usa ese software.
- c) En la parte derecha **se muestra el estado que cambiará su color en función del valor**. Se hace con [ngClass]. Así:
 - ✚ Si el estado es “en tránsito” se muestra en **naranja**
 - ✚ Si es “elevada” se muestra en **rojo**
 - ✚ Si es “cerrada” se muestra en **verde**
- d) Además, cada llamada lleva una **borde inferior: punteado y azul claro y fino** para separar un elemento de otro (como separador).

Lo creamos con el comando:

```
ng g c /calls/ItemCall --flat
```

--flat es para que no le cree una carpeta; lo metemos suelto en /calls

El código del componente es (hay que usar flexLayout para distribuir elementos):

ítem-call.component.html

```
<div class="container">
  <div class="row">
    <div class="column col-2 my-auto mx-auto p-0">
      <div>
        <img [src]="call.imageUrl" [ngStyle]="{width:'100%'}">
      </div>
      <div class="d-flex justify-content-center">{{call.fechaIncidencia |
date:'dd/MM/yyyy'}}</div>
    </div>

    <div class="column col-6">
      <h5 class="row d-flex justify-content-between">
        <div> {{call.codIncidencia}}</div><div> {{call.softwareIncidencia}}</div>
      </h5>
      <div>{{call.descripcion}}</div>
      <div>
        <i>Operador/a: </i>
        <i> {{call.operadorAtendio}}</i>
      </div>
      <div>
        <span *ngFor="let name of call.empresaDptoProyecto; let i=index">
          <a href="#">{{name}}</a>
          <span>
            {{i < (call.empresaDptoProyecto.length-1) ? '>' : ''}}
          </span>
        </span>
      </div>
      <div class="col-3 text-center justify-content-center my-auto"
[ngClass]="getClass()">{{call.estado}}</div>
    </div>
  </div>
  <hr>
```

ítem-call.component.ts

```
import { Component, OnInit, Input } from '@angular/core';
import { CallModel } from '../model/call-model';

@Component({
  selector: 'app-item-call',
  templateUrl: './item-call.component.html',
  styleUrls: ['./item-call.component.css']
})
```

```
export class ItemCallComponent implements OnInit {
  @Input() call: CallModel;

  constructor() { }

  ngOnInit() {
  }

  getClass() {
    switch (this.call.estado) {
      case 'en tránsito':
        return 'text-warn';
      case 'elevada':
        return 'text-red';
      case 'cerrada':
        return 'text-success';
    }
  }
}
```

ítem-call.component.css

```
hr{
  border-bottom:0.5px dotted cornflowerblue;
}

.text-success{
  color:green;
}

.text-warn{
  color:orange;
}

.text-red{
  color:red;
}
```

8.5.-CallsComponent

Mostrará un listado de llamadas atendidas por el CallCenter (incorpora el componente ItemCallComponent para hacer el listado).

Su aspecto visual será:

 22/01/2019	BIB#1 fallo al realizar copia seguridad <i>Operador/a: Ana María</i> Capgemini > Angular > Libreria	BIBLIOTECH cerrada
 02/11/2018	CON#2 fallo al imprimir informes finales <i>Operador/a: Víctor</i> DXC > Consultoría > Caja Free	CONTATOTAL en tránsito
 11/03/2018	BIB#3 fallo al mostrar el listado de libros <i>Operador/a: Esther</i> Alta TECH > International > Libreria	BIBLIOTECH en tránsito

Lo creamos con el comando:

```
ng g c /calls/calls --flat
```

El código del componente es:

calls.component.html

```
<app-item-call  
  *ngFor="let miCall of calls"  
  [call]="miCall"  
  [routerLink]="['/calls',miCall.codIncidencia]"  
  routerLinkActivated="active"></app-item-call>
```

calls.component.ts

```
import { Component, OnInit } from '@angular/core';  
import { CallModel } from '../model/call-model';  
import { Calls } from '../model/calls';  
  
@Component({  
  selector: 'app-calls',  
  templateUrl: './calls.component.html',  
})
```

```

    styleUrls: ['./calls.component.css']
  })
  export class CallsComponent implements OnInit {
    calls: CallModel[];

    constructor() { }

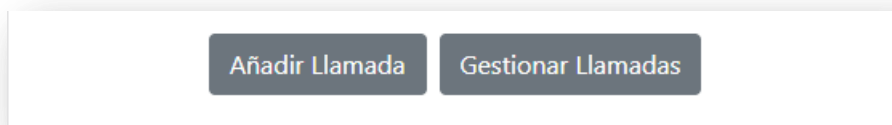
    ngOnInit() {
      this.calls = Calls.getCalls();
    }
  }
}

```

8.6.- Barra de navegación de llamadas: **NavbarCallsComponent**

Mostrará una barra de navegación con las opciones: añadir, gestionar y borrar, en la cabecera del componente CallsComponent (el listado de llamadas).

Su aspecto visual será:



Para crearla, usaremos “Buttons” de Bootstrap y para el componente, tecleamos:

ng g c calls/ui/NavbarCalls

Y su código será:

navbar-calls.component.html

```

<div class="container d-flex justify-content-center my-3">
  <button class="btn btn-secondary mr-2"
    routerLink="add-call"
    routerLinkActivated="active">Añadir Llamada
  </button>
  <button class="btn btn-secondary"
    routerLink="add-call"
    routerLinkActivated="active">Gestionar Llamadas
  </button>
</div>

```

navbar-calls.component.ts

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

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

  constructor() { }

  ngOnInit() {
  }

}
```

navbar-calls.component.css

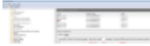

```
.margenes{
  margin: 2% 2%;
}
```

Y para que nuestra barra se muestre, la añadimos en **calls.component.html** de manera que ahora ese fichero, nos queda:

calls.component.html

```
<app-navbar-calls></app-navbar-calls>
<app-item-call
  *ngFor="let miCall of calls"
  [call]="miCall"
  [routerLink]="['/calls',miCall.codIncidencia]"
  routerLinkActivated="active"></app-item-call>
```

Con estos cambios, nuestro CallsComponent, nos ha quedado:

Añadir Llamada		Gestionar Llamadas	
	BIB#1	BIBLIOTECH	
22/01/2019	fallo al realizar copia seguridad Operador/a: Ana María Capgemini > Angular > Libreria		cerrada
	CON#2	CONTATOTAL	
02/11/2018	fallo al imprimir informes finales Operador/a: Víctor DXC > Consultoría > Caja Free		en tránsito
	BIB#3	BIBLIOTECH	
11/03/2018	fallo al mostrar el listado de libros Operador/a: Esther Alta TECH > International > Libreria		en tránsito

8.7.-Añadimos el componente CallsComponent al módulo de Routing

En nuestro fichero de configuración de routing: “**app-routing.module.ts**”, tenemos que:

- 1º) Importar el nuevo componente CallsComponent para poder usarlo
- 2º) Añadir una ruta (que coincida con la que hemos puesto en la barra de navegación para Calls) en el array de rutas. En nuestro caso: “/calls”

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from '../home/home.component';
import { CallsComponent } from '../calls/calls.component';

const routes: Routes = [
  { path: '', component: HomeComponent, pathMatch: 'full' },
  { path: 'home', component: HomeComponent },
  { path: 'calls', component: CallsComponent },
];
```

```

    { path: 'calls', component: CallsComponent }
  ];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

8.8.-ContactComponent

Mostrará un formulario de contacto.

Necesitamos el componente de Bootstrap: **Forms**.

Su aspecto visual será:

Donde “mensaje” es un **TextArea** que puede contener muchas líneas.

Para crearlo, tecleamos:

ng g c contact

Y su código será:

contact.component.html

```
<form>
  <div class="form-group">
    <label for="nombre">Nombre:</label>
    <input class="form-control" type="text" id="nombre" placeholder="Introduce el nombre" [disabled]="enviando">
  </div>
  <div class="form-group">
    <label for="email">Correo electrónico</label>
    <input type="email" class="form-control" id="email" aria-describedby="emailHelp" placeholder="Introduce email" [disabled]="enviando">
    <small id="emailHelp" class="form-text text-muted">Nosotros nunca compartiremos tu email con nadie más</small>
  </div>
  <div class="form-group">
    <label for="mensaje">Mensaje:</label>
    <textarea class="form-control" rows="5" [(ngModel)]="mensaje" name="textoMensaje" [disabled]="enviando"></textarea>
  </div>
  <br>

  <div *ngIf="!enviando">
    <div class="row justify-content-center">
      <button class="btn btn-primary mr-3" (click)="enviarMensaje()">Enviar</button>
      <button class="btn btn-primary" (click)="volverAtras()">Volver</button>
    </div>
  </div>

  <div *ngIf="detalles">
    {{detalles}}
  </div>
```

Como estoy usando **[(ngModel)]** para hacer binding en doble sentido en `<textarea>`, necesito importar el módulo **FormsModule** en *app.module.ts*. Por tanto, en el fichero “**app.module.ts**” añadimos:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, APP_INITIALIZER } from '@angular/core';
import { FormsModule } from '@angular/forms'; // para usar ngModel

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
```

```

import { HeaderComponent } from './ui/header/header.component';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';

import { NavbarComponent } from './ui/navbar/navbar.component';
import { HomeComponent } from './home/home.component';
import { ItemCallComponent } from './calls/item-call.component';
import { CallsComponent } from './calls/calls.component';
import { ContactComponent } from './contact/contact.component';
import { AddCallComponent } from './calls/add-call/add-call.component';
import { DetailCallComponent } from './calls/detail-call/detail-
call.component';
import { NavbarCallsComponent } from './calls/ui/navbar-calls/navbar-
calls.component';

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    CallsComponent,
    ContactComponent,
    AddCallComponent,
    DetailCallComponent,
    NavbarCallsComponent,
    ItemCallComponent,
    NavbarComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Por otro lado, el código del controlador de nuestro componente, nos quedará:

contact.component.ts

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

@Component({
  selector: 'app-contact',
  templateUrl: './contact.component.html',
  styleUrls: ['./contact.component.css']
})
export class ContactComponent implements OnInit {
  enviando: boolean;
  mensaje: string;
  detalles: string;

  constructor(private _location: Location) {
    this.enviando = false;
    this.mensaje = '';
  }

  ngOnInit() {
  }

  /** Método que envia el mensaje y vuelve al componente anterior */
  enviarMensaje() {
    this.enviando = true;
    this.detalles = 'Enviando mensaje...';

    setTimeout(() => {
      this.enviando = false;
      this.volverAtras();
    }, 2000);
  }

  /** Método que navega hacia atrás a la página anteriormente visitada */
  volverAtras() {
    this._location.back();
  }
}
```


8.9.-Añadimos el componente ContactComponent al módulo de Routing

En nuestro fichero de configuración de routing: “**app-routing.module.ts**”, tenemos que:

- 1º) Importar el nuevo componente ContactComponent para poder usarlo
- 2º) Añadir una ruta (que coincida con la que hemos puesto en la barra de navegación para contactos) en el array de rutas. En nuestro ejemplo: “/contact”

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from '../home/home.component';
import { CallsComponent } from '../calls/calls.component';
import { ContactComponent } from '../contact/contact.component';

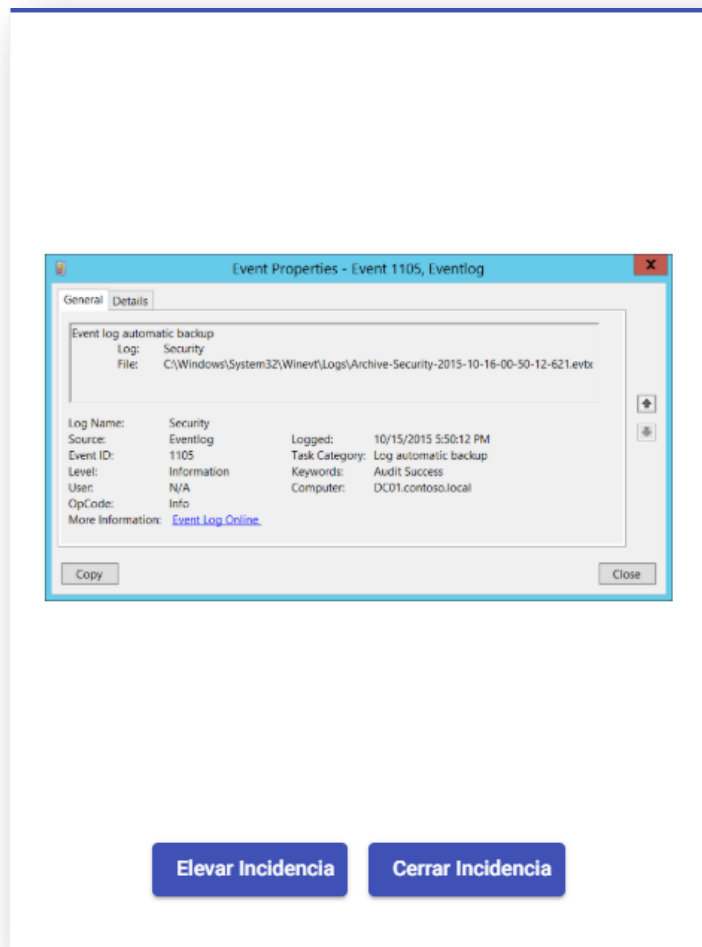
const routes: Routes = [
  { path: '', component: HomeComponent, pathMatch: 'full' },
  { path: 'home', component: HomeComponent },
  { path: 'calls', component: CallsComponent },
  { path: 'contact', component: ContactComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

8.10.-DetailCallComponent

Cuando estamos en el listado de llamadas recogidas, si pulsamos sobre una de ellas, navegaremos por su código de incidencia a la página de detalles de esa llamada, donde veríamos la pantalla de error ampliada y un botón para elevar la incidencia y otro para cerrarla.

Su aspecto visual será:



Para crearlo en la carpeta calls, tecleamos en la terminal:

ng g c calls/detailCall

PASOS:

- 1) Lo primero que tenemos que hacer, es **recoger el codIncidencia y averiguar qué llamada es la** que nos viene de la URL, usando **paramMap** de **ActivatedRoute**. En el fichero *detail-call.component.ts* escribimos:

```
import { Component, OnInit } from '@angular/core';
import { CallModel } from '../../model/call-model';
import { ActivatedRoute } from '@angular/router';
import { Calls } from '../../model/calls';

@Component({
  selector: 'app-detail-call',
  templateUrl: './detail-call.component.html',
  styleUrls: ['./detail-call.component.css']
})
```

```

})
export class DetailCallComponent implements OnInit {
  codIncidencia: string;
  callSeleccionada: CallModel;

  constructor(private _route: ActivatedRoute) {
    this.codIncidencia = _route.snapshot.paramMap.get('cod');
    this.callSeleccionada = Calls.getCall(this.codIncidencia);
  }

  ngOnInit() {
  }
}

```

- 2) Ya podemos **diseñar la plantilla HTML** de este componente, tomando los datos de la propiedad “*callSeleccionada*”:

detail-call.component.html

```

<div class="container">
  <div class="col">
    <img class="col-12" [src]="callSeleccionada.imageUrl" width="100%">
  </div>
  <div class="row justify-content-center my-3">
    <button class="btn btn-primary mr-3"
(click)="elevarIncidencia(callSeleccionada.codIncidencia)">Elevar Incidencia
    </button>
    <button class="btn btn-primary"
(click)="cerrarIncidencia(callSeleccionada.codIncidencia)">Cerrar Incidencia
    </button>
  </div>
</div>

```

- 3) Como hemos puesto un **botón de elevar incidencia y otro de cerrar incidencia**, vamos a programar sus eventos click, mediante la creación de los métodos **elevarIncidencia(...)** y **cerrarIncidencia(...)** en el **fichero controlador de este componente (.ts)**:

La forma que he pensado para estos métodos ha sido:

- Les cambio el estado en el array de llamadas que está en nuestro modelo de datos, llamando a los métodos del modelo de datos
- A continuación, **dirigo al ROUTER a la página** del listado de **LLAMADAS**, usando **router.navigate()**:

```

import { Component, OnInit } from '@angular/core';
import { CallModel } from '../../model/call-model';
import { ActivatedRoute, Router } from '@angular/router';
import { Calls } from '../../model/calls';

@Component({
  selector: 'app-detail-call',
  templateUrl: './detail-call.component.html',
  styleUrls: ['./detail-call.component.css']
})
export class DetailCallComponent implements OnInit {
  codIncidencia: string;
  callSeleccionada: CallModel;

  constructor(private _router: Router, private _route: ActivatedRoute) {
    this.codIncidencia = _route.snapshot.paramMap.get('cod');
    this.callSeleccionada = Calls.getCall(this.codIncidencia);
  }

  ngOnInit() {
  }

  elevarIncidencia(codIncidencia: string) {
    Calls.upCall(codIncidencia);
    this._router.navigate(['calls']);
  }

  cerrarIncidencia(codIncidencia: string) {
    Calls.closeCall(codIncidencia);
    this._router.navigate(['calls']);
  }
}

```

- 4) Debemos asegurarnos también, por si no lo hemos hecho primero, que haya una ruta de navegación `/calls/:cod` hacia este componente en nuestro fichero de **ROUTING. Y de importar el COMPONENTE:**

app-routing.module.ts

```

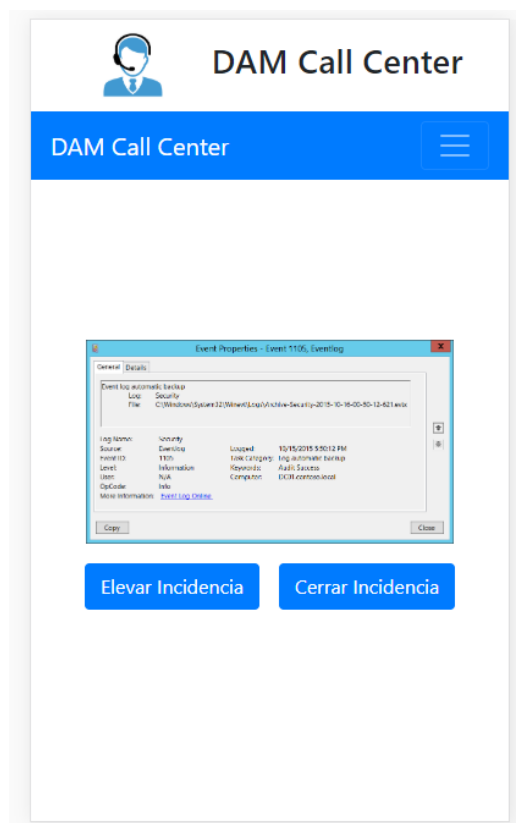
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from '../home/home.component';
import { CallsComponent } from '../calls/calls.component';
import { ContactComponent } from '../contact/contact.component';
import { DetailCallComponent } from '../calls/detail-call/detail-call.component';

```


```
const routes: Routes = [
  { path: '', component: HomeComponent, pathMatch: 'full' },
  { path: 'home', component: HomeComponent },
  { path: 'calls', component: CallsComponent },
  { path: 'contact', component: ContactComponent },
  { path: 'calls/:cod', component: DetailCallComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Si probamos la aplicación, veremos:



Si pulsamos ELEVAR INCIDENCIA:



DAM Call Center


DAM Call Center

Añadir Llamada

Gestionar Llamadas

BIB#1

BIBLIOTECH



fallo al realizar copia seguridad

22/01/2019


Operador/a: Ana María

Capgemini > Angular > Libreria

elevada

CON#2

CONTATOTAL



fallo al imprimir informes finales

02/11/2018


Operador/a: Víctor

DXC > Consultoría > Caja Free

en tránsito

BIB#3

BIBLIOTECH



fallo al mostrar el listado de libros

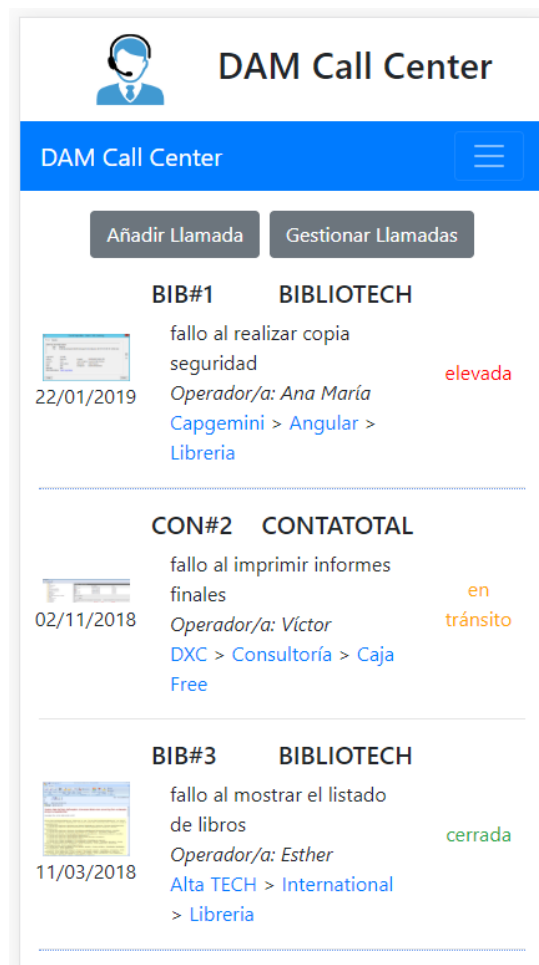
11/03/2018

Operador/a: Esther

Alta TECH > International > Libreria

en tránsito

Si pulsamos el de CERRAR INCIDENCIA, nos muestra:



8.11.-AddCallComponent

Corresponde al componente que permite añadir una nueva llamada a la lista de llamadas atendidas. Será llamado desde la barra de navegación que hay dentro del listado de llamadas.

Su aspecto visual será:

Para crearlo dentro de llamadas, en la terminal tecleamos:

ng g c calls/AddCall

En cuanto a la lógica , lo que tiene que hacer es:

- 1) Mostrar la página para introducir los datos de la llamada (eso lo hará la [plantilla](#) del componente AddCall)**
- 2) Al pulsar un botón AÑADIR en la plantilla me creará una llamada con esos datos y lo guardará en el array de llamadas**
- 3) Nos llevará el router a la página listado de llamadas para ver la nueva llamada en la lista**

Vamos a hacerlo:

Primero creamos la plantilla HTML:

add-call.component.html

```
<div class="container mt-2">
  <form>
    <div class="form-group">
      <label for="descripcion">Descripción:</label>
```



```

        <input class="form-control" placeholder="Describe el problema"
type="text" [(ngModel)]="call.descripcion" id="descripcion"
        name="descripcion">
    </div>

    <div class="form-group">
        <label for="software">Software que ocasionó la incidencia:</label>
        <input class="form-control" placeholder="Software que provocó la
incidencia" type="text" [(ngModel)]="call.softwareIncidencia"
        id="software" name="software">
    </div>

    <div class="form-group">
        <label for="imageUrl">Captura pantalla de la incidencia: </label>
        <input class="form-control" placeholder="Url de la captura de
incidencia" type="text" [(ngModel)]="call.imageUrl" id="imageUrl"
        name="imageUrl">
    </div>

    <div class="form-group">
        <label for="empresa">Empresa, Departamento y Proyecto donde se
generó la incidencia </label>
        <input class="form-control" placeholder="Ej: MiEmpresa,
Contabilidad, NuevoProyecto" type="text"
[(ngModel)]="call.empresaDptoProyecto"
        id="empresa" name="empresa">
    </div>

    <div class="form-group">
        <label for="telefono">Teléfono desde donde llama </label>
        <input class="form-control" placeholder="Introduce el teléfono
desde donde llamas" type="text" [(ngModel)]="call.telefonoOrigen"
        id="telefono" name="telefono">
    </div>

    <div class="form-group">
        <label for="operador">Operador que le ha atendido </label>
        <input class="form-control" placeholder="Introduce qué operador le
ha atendido" type="text" [(ngModel)]="call.operadorAtendio"
        id="operador" name="operador">
    </div>

    <div class="form-group">
        <label for="fecha">Fecha de la incidencia</label>
        <input class="form-control" placeholder="Introduce fecha
incidencia" type="date" [(ngModel)]="call.fechaIncidencia" id="fecha"
        name="fecha">
    </div>

```

```

        <button class="btn btn-primary d-flex mx-auto my-3"
(click)="addCall()">Añadir</button>

    </form>
</div>

```

Fijaros que la plantilla usa [(ngModel)], por tanto, si no estuviese ya, habría que importar FormsModule en app.module.ts. **Además, al usar [(ngModel)], necesitamos el atributo “name” de los inputs, pues [(ngModel)] lo usa para hacer el binding.**

Fijaros también que en la plantilla enlazamos a “call.XXX” los input. Por tanto, tenemos que tener en nuestro controlador (.ts) una propiedad de tipo CallModel y llamada “call”.

Una vez que tenemos la plantilla, vamos a nuestro controlador (archivo: add-call.component.ts) y nos aseguramos de:

- 1) Declarar una propiedad de tipo CallModel y llamada call. Para eso, necesitamos importar la clase Calls y CallModel:

```

import { CallModel } from '../..model/call-model';
import { Calls } from '../..model/calls';

```

Y definimos la propiedad en la clase (fichero add-call.component.ts):

```

call: Calls;

```

- 2) En el constructor, inicializamos ese producto a una NEW CallModel (). Pero tal como tenemos diseñado el ejercicio, tenemos un **PROBLEMA**: **nuestro constructor requiere que le pasemos todos los atributos de una llamada y aún no los tenemos, ya que nos los darán los inputs cuando los rellene el usuario;** además no podemos tener dos constructores con distintos parámetros. Una **SOLUCIÓN**: es añadir en la clase Calls (la que tiene los métodos para gestionar las llamadas) un método que cree una llamada vacía:

```

/** Método que crea una llama vacía, sin datos */
static createEmptyCall(): CallModel {
    return new CallModel('', '', '', [''], '', 'en tránsito', '', null);
}

```

Pero esta solución **TIENE OTRO PROBLEMA**, y es que el “codIncidencia” que genera la nueva llamada no es correcto, pues el nombre del software que debe formar parte del código es vacío. Por tanto, para **SOLUCIONARLO**, **DEBEMOS MODIFICAR en la clase Calls**, el método “AddCall” para que antes de añadir, obtenga el “codIncidencia” correcto. Nos quedará:

calls.ts

```
/** Método que añade una llamada que se le pasa */
static addCall(call: CallModel): void {
    call.generarCodIncidencia();
    this.CALLS.push(call);
}
```

Como véis, antes de insertar en el array llamo al método “**generarCodIncidencia()**” del modelo de datos **CallModel**. Pero ese método no existía, así que lo creamos en el fichero: “**call-model.ts**”:

call-model.ts

```
/** Método que genera el codIncidencia.
 * En el caso de productos añadidos con el formulario añadir, se necesita,
 * pues inicialmente no se sabe el software y el constructor genera un cod
 * inadecuado
 */
generarCodIncidencia() {
    this.codIncidencia = this.softwareIncidencia.slice(0, 3) + '#' +
this.idCall;
}
```

Ahora ya podemos inicializar nuestra llamada en **add-call.component.ts**:

```
import { Component, OnInit } from '@angular/core';
import { CallModel } from '../../model/call-model';
import { Calls } from '../../model/calls';

@Component({
    selector: 'app-add-call',
    templateUrl: './add-call.component.html',
    styleUrls: ['./add-call.component.css']
})
export class AddCallComponent implements OnInit {
    call: CallModel;

    constructor() {
        this.call = Calls.createEmptyCall();
    }
    ngOnInit() {
    }
}
```

- 3) Ahora creamos el método addCall() que es llamado desde la plantilla al pulsar el botón AÑADIR:

```
addCall() {  
  
    let empresa: string = this.call.empresaDptoProyecto.toString();  
    this.call.empresaDptoProyecto = empresa.split(',');  
  
    Calls.addCall(this.call);  
    this._router.navigate(['calls']);  
  
}
```

Fijaros que en este método extraigo la empresa/dpto./proyecto a un string, lo convierto a Array de strings y luego lo vuelvo a asignar.

Es necesario hacerlo porque call.empresaDptoProyecto es de tipo Array<string>, pero desde la plantilla lo escribimos como “MiEmpresa, Departamento, Proyecto” y esto no es un ARRAY, es un STRING. Si lo dejásemos así, se guardaría como si fuese un array de un string.

Fijaros que después de añadir, llamamos al router.navigate() para que se muestre el listado de llamadas otra vez con la nueva incorporada.

El código del archivo controlador queda así:

add-call.component.ts

```
import { Component, OnInit } from '@angular/core';  
import { CallModel } from '../../model/call-model';  
import { Calls } from '../../model/calls';  
import { Router } from '@angular/router';  
  
@Component({  
  selector: 'app-add-call',  
  templateUrl: './add-call.component.html',  
  styleUrls: ['./add-call.component.css']  
})  
export class AddCallComponent implements OnInit {  
  call: CallModel;  
  fechaString: string;  
  
  constructor(private _router: Router) {  
    this.call = Calls.createEmptyCall();  
  }  
  
  ngOnInit() {  
  }  
}
```

```

addCall() {

  let empresa: string = this.call.empresaDptoProyecto.toString();
  this.call.empresaDptoProyecto = empresa.split(',');
  console.log(this.call.empresaDptoProyecto);
  Calls.addCall(this.call);
  this._router.navigate(['calls']);

}
}

```

Por último, tenemos que añadir esta nueva ruta al módulo de enrutado:
app-routing.module.ts:

¡CUIDADO!

Como esta nueva ruta tiene la primera parte igual que 'calls/:cod' y ésta la habíamos definido antes, si lo dejamos en ese orden:

1º) 'calls/:cod'

2º) 'calls/add-call'

Cuando intentásemos, navegar a 'calls/add-call', **detectaría como URL que ENCAJA, LA PRIMERA**, aunque :cod indique una variable e intentaría cargar el componente DetailCallComponent.

ES MUY IMPORTANTE EL ORDEN DE LAS RUTAS: las que lleven variables AL FINAL SIEMPRE y la última del todo ***


Por tanto, nos quedará:

```

const routes: Routes = [
  { path: '', component: HomeComponent, pathMatch: 'full' },
  { path: 'home', component: HomeComponent },
  { path: 'calls', component: CallsComponent },
  { path: 'contact', component: ContactComponent },
  { path: 'calls/add-call', component: AddCallComponent },
  { path: 'calls/:cod', component: DetailCallComponent }
];

```

EJEMPLO EN FUNCIONAMIENTO EL ADD CALL:



DAM Call Center

DAM Call Center

Descripción:

Software que ocasionó la incidencia:

Captura pantalla de la incidencia:

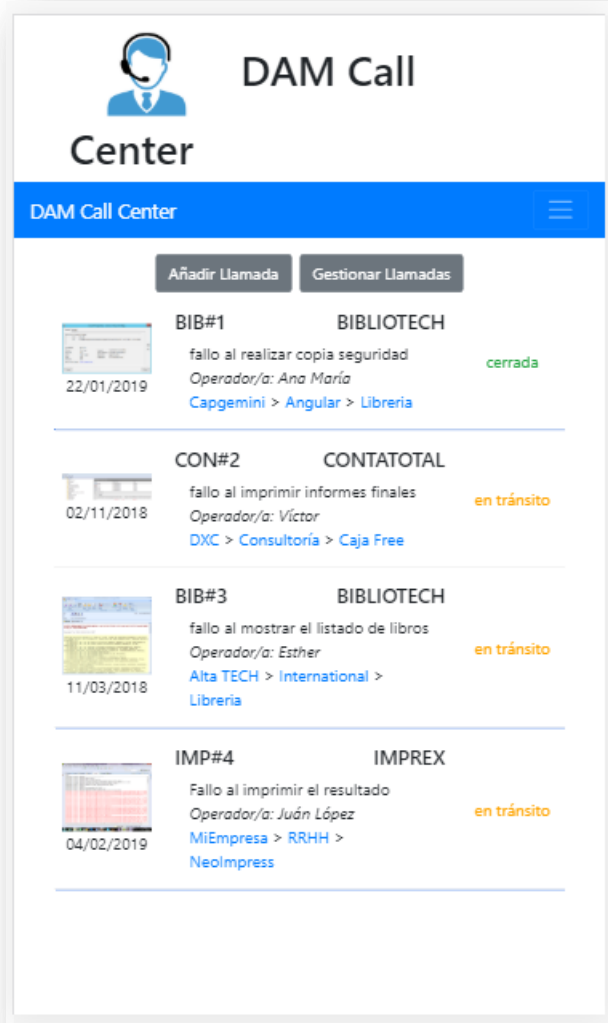
Empresa, Departamento y Proyecto donde se generó la incidencia

Teléfono desde donde llama

Operador que le ha atendido

Fecha de la incidencia

Y al pulsar AÑADIR:



9. Añadimos al módulo de Routing la ruta para errores, que vaya a home

El fichero “app-routing.module.ts” ya existe y estará casi configurado. Acordaros que además de las rutas, **hay que importar cada componente al que queráis enrutar**.

```
const routes: Routes = [
  { path: '', component: HomeComponent, pathMatch: 'full' },
  { path: 'home', component: HomeComponent },
  { path: 'calls', component: CallsComponent },
  { path: 'contact', component: ContactComponent },
  { path: 'calls/add-call', component: AddCallComponent },
```

```
{ path: 'calls/:cod', component: DetailCallComponent },  
{ path: '**', redirectTo: 'home' }  
];
```