

Enunciado para hacer el proyecto:

Vamos a crear una aplicación que nos muestre por un lado una lista de videojuegos "controlados" por la aplicación.

Esta lista contendrá nombre del juego, editor, posición del ranking de juegos, a ser posible carátula del juego.

Además, se mostrará un listado de jugadores, los cuales tendrán, además de su nombre, su nick en los juegos y si queréis una imagen (un avatar) y una supuesta puntuación del jugador (ranking).

Nivel 1. Hacer la app desde cero, usando bootstrap para el formato, y un sólo módulo/enrutador. Cuando lo termines, lo muestras a tu instructor.

Nivel 2. Refactorizar la aplicación para que los juegos y los jugadores tengan cada uno su propio módulo.

0. Instalar Angular Material

<https://material.angular.io/guide/getting-started>

- ng add @angular/material
- elijo el tema purple/green
- y le doy a YES para las otras dos preguntas que me hacen

- Ahora, cuando vaya a usar un componente de Angular Material, voy al app.module.ts (o a los módulos hijos en los que lo vaya a usar), y añado el módulo correspondiente de tal componente de Angular Material

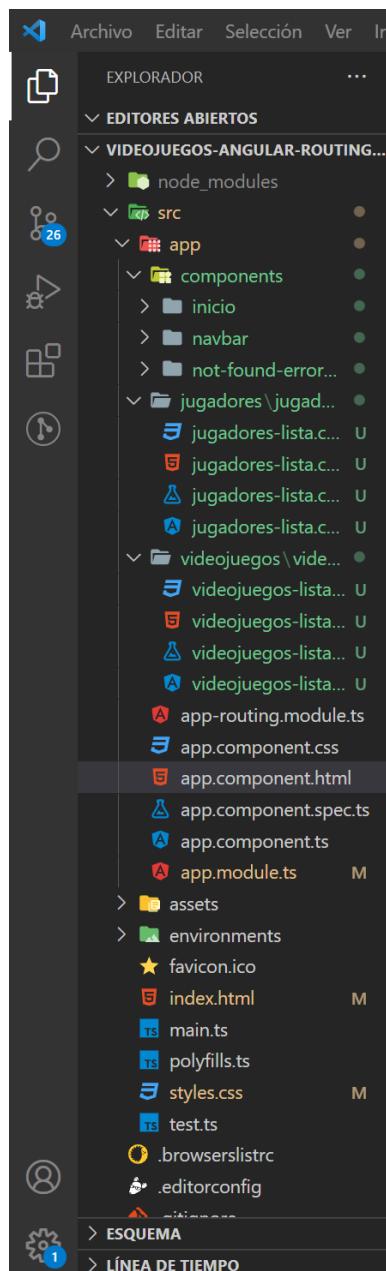
- Ya está listo Angular Material para usarse !!

<https://material.angular.io/components/categories>

1. Routing: Introducción y configuración básica

- 1.1. Creamos los componentes:

- 1.1.1. ng g c components/navbar
- 1.1.2. ng g c components/inicio
- 1.1.3. ng g c components/not-found-error404
- 1.1.4. ng g c videojuegos/videojuegos-lista
- 1.1.5. ng g c jugadores/jugadores-lista



- 1.2. Vamos a montar un navbar con Angular Material

1.2.1. Vamos a components/navbar.component.html

1.2.2. Tomamos de ejemplo el primer toolbar que nos encontramos y lo pegamos en nuestro navbar.component.html

<https://material.angular.io/components/toolbar/examples>

```
<!-- <p>navbar works!</p> -->

<mat-toolbar> <!-- color="primary"-->
  <mat-toolbar-row>
    <button mat-icon-button>
      <mat-icon (click)="sidenav.toggle();">menu</mat-icon>
    </button>

    <h1>Angular Material Sidenav</h1>

    <span class="example-spacer"></span>

    <button mat-icon-button class="example-icon favorite-icon">
      <mat-icon>favorite</mat-icon>
    </button>

    <button mat-icon-button class="example-icon">
      <mat-icon>share</mat-icon>
    </button>
  </mat-toolbar-row>
</mat-toolbar>

<mat-sidenav-container>
  <mat-sidenav #sidenav opened="false" mode="side">
    <mat-nav-list>
      <a mat-list-item>Inicio</a>
      <a mat-list-item>Videojuegos</a>
      <a mat-list-item>Jugadores</a>
      <a mat-list-item>Help</a>
    </mat-nav-list>
  </mat-sidenav>
  <mat-sidenav-content>
    <div style="height: 88vh;">
      <router-outlet></router-outlet>
    </div>
  </mat-sidenav-content>
</mat-sidenav-container>
```

```
<!-- https://www.youtube.com/watch?v=IRzGphrefsY -->
```

1.2.3. Y en el navbar.component.css ponemos...

```
.example-spacer {  
    flex: 1 1 auto;  
}
```

1.2.4. Creamos un module.ts específico para Angular Material

ng g m angular-material

1.2.5. En él vamos a importar los principales componentes que podremos (o no) usar a lo largo del frontend del proyecto

```
import { NgModule } from '@angular/core';  
import { CommonModule } from '@angular/common';  
import { MatButtonModule } from '@angular/material/button';  
import { MatCheckboxModule } from '@angular/material/checkbox';  
import { MatToolbarModule } from '@angular/material/toolbar';  
import { MatInputModule } from '@angular/material/input';  
import { MatProgressSpinnerModule } from  
    '@angular/material/progress-spinner';  
import { MatCardModule } from '@angular/material/card';  
import { MatMenuModule } from '@angular/material/menu';  
import { MatIconModule } from '@angular/material/icon';  
import { MatSidenavModule } from '@angular/material/sidenav';  
import { MatListModule } from '@angular/material/list';  
  
@NgModule({  
    declarations: [],  
    imports: [MatListModule, MatSidenavModule, MatButtonModule,  
        MatCheckboxModule, MatToolbarModule, MatInputModule, MatProgressSpinnerModule,  
        MatCardModule, MatMenuModule, MatIconModule, CommonModule],  
    exports: [MatListModule, MatSidenavModule, MatButtonModule,  
        MatCheckboxModule, MatToolbarModule, MatInputModule, MatProgressSpinnerModule,  
        MatCardModule, MatMenuModule, MatIconModule, CommonModule]  
})  
export class AngularMaterialModule {}
```

1.2.6. Y ahora, tan sólo tenemos que importar este angular-material.module.ts en el app.module.ts

```
imports: [
```

```
    BrowserModule,  
    AppRoutingModule,  
    BrowserAnimationsModule,  
    AngularMaterialModule  
],
```

1.2.7. El siguiente paso, será el de ir al app.component.html y borrarlo todo, para poner:

```
<app-navbar></app-navbar>  
  
<router-outlet></router-outlet>
```

- 1.3. En este punto, desarrollaremos el componente videojuegos-lista. Para ello, previamente crearemos el modelo de datos Videojuego (videojuego.model.ts) y un chero mock data (mocks.ts) con un conjunto de videojuegos de prueba.

1.3.1. Creamos la carpeta models

Dentro de ella, creamos el la interface videojuego-model.ts

```
ng g interface interfaces/videojuego-model
```

```
export interface VideojuegoModel {  
  id: number;  
  titulo: string;  
  creador: string;  
  descripcion: string;  
  ranking: number;  
  imagen: string;  
}
```

1.3.2. Creamos la carpeta mocks

Dentro de ella, creamos el archivo videojuego-mock.ts

```
export const VIDEOJUEGOS: VideojuegoModel[] = [  
{  
  id: 0,  
  titulo: "Counter Strike",  
  creador: "VALVE Software",  
  descripcion: "bla bla bla",  
  ranking: 1,
```

```
        imagen: "../../assets/counter-strike.jpg"
    },
    {
        id: 1,
        titulo: "FIFA 2021",
        creador: "EA Sports",
        descripcion: "bla bla bla",
        ranking: 2,
        imagen: "../../assets/fifa-2021.jpg"
    },
    {
        id: 2,
        titulo: "Call Of Duty",
        creador: "Infinity Ward",
        descripcion: "bla bla bla",
        ranking: 3,
        imagen: "../../assets/call-of-duty.jpg"
    },
    {
        id: 3,
        titulo: "Days Gone",
        creador: "Sony Interactive Entertainment",
        descripcion: "bla bla bla",
        ranking: 4,
        imagen: "../../assets/days-gone.jpg"
    },
    {
        id: 4,
        titulo: "Pro Evolution Soccer",
        creador: "Nipona Konami",
        descripcion: "bla bla bla",
        ranking: 5,
        imagen: "../../assets/pes-2021.jpg"
    },
    {
        id: 5,
        titulo: "Rachet & Clank",
        creador: "Sony Interactive",
        descripcion: "bla bla bla",
        ranking: 6,
        imagen: "../../assets/rachet.jpg"
    },
];

```

- 1.4. Vamos videojuegos-lista.component.ts

1.4.1. Creamos una propiedad llamada videojuegos que es un Array del VideojuegoModel

1.4.2. En el ngOnInit() llamamos a this.videojuegos = VIDEOJUEGOS; para que al iniciarse la app, siempre se despliegue la lista de libros que hemos definido en el videojuegos-mock.ts

```
export class VideojuegosListaComponent implements OnInit {  
  
    videojuegos: VideojuegoModel[] = [];  
  
    constructor() { }  
  
    ngOnInit(): void {  
        this.videojuegos = VIDEOJUEGOS;  
    }  
}
```

1.4.3. Ahora en el videojuegos-lista.component.html vamos a copiar y pegar un modelo de Card desde Angular Material, para desplegar con la directiva *ngFor todos los videojuegos:

```
<p>videojuegos-lista works!</p>  
  
<div *ngIf="videojuegos != null || videojuegos != undefined">  
    <mat-card class="example-card" *ngFor="let videojuego of  
    videojuegos">  
        <mat-card-header>  
            <!-- <div mat-card-avatar class="example-header-image"></div> -->  
              
  
        <mat-card-content>  
            <p>{{ videojuego.descripcion }}</p>  
        </mat-card-content>  
  
        <mat-card-actions>
```

```

        <button mat-button>ID: {{ videojuego.id }}</button>
        <button mat-button>Ranking: {{ videojuego.ranking }}</button>
        <button mat-button>LIKE</button>
        <button mat-button>SHARE</button>
    </mat-card-actions>
</mat-card>
</div>

```

1.4.4. Y en videojuegos.component.css añadimos esto...

```

.example-card {
    max-width: 350px;
    margin: 15px;
}

/* .example-header-image {
    background-image:
url('https://material.angular.io/assets/img/examples/shiba1.jpg');
    background-size: cover;
} */

.container {
    padding: 1px;
    display: flex;
    flex-wrap: wrap;
}

```

- 1.5. Una vez creados los componentes, pasaremos a realizar la configuración del servicio Router, y para ello vamos al app-routing.module.ts para crear nuestras primeras rutas a nuestros componentes:

```

const routes: Routes = [
    {
        path: 'inicio',
        component: InicioComponent
    },
    {
        path: 'videojuegos',
        component: VideojuegosListaComponent
    },
    {

```

```

        path: 'jugadores',
        component: JugadoresListaComponent
    },
{
    path: '',
    redirectTo: '/inicio',
    pathMatch: 'full'
},
{
    path: '**',
    component: NotFoundError404Component
}
];

```

2. Routing: RouterLinks

- 2.1. Volvemos al navbar.component.html para retocar navbar:

```

<p>navbar works!</p>

<mat-toolbar> <!-- color="primary"-->
<mat-toolbar-row>
    <button mat-icon-button>
        <mat-icon (click)=" sidenav.toggle(); ">menu</mat-icon>
    </button>

    <h1>Servicio Router: ejemplos de uso de Childs Routes</h1>

    <span class="example-spacer"></span>

    <button mat-icon-button class="example-icon favorite-icon">
        <mat-icon>favorite</mat-icon>
    </button>

    <button mat-icon-button class="example-icon">
        <mat-icon>share</mat-icon>
    </button>
</mat-toolbar-row>
</mat-toolbar>

```

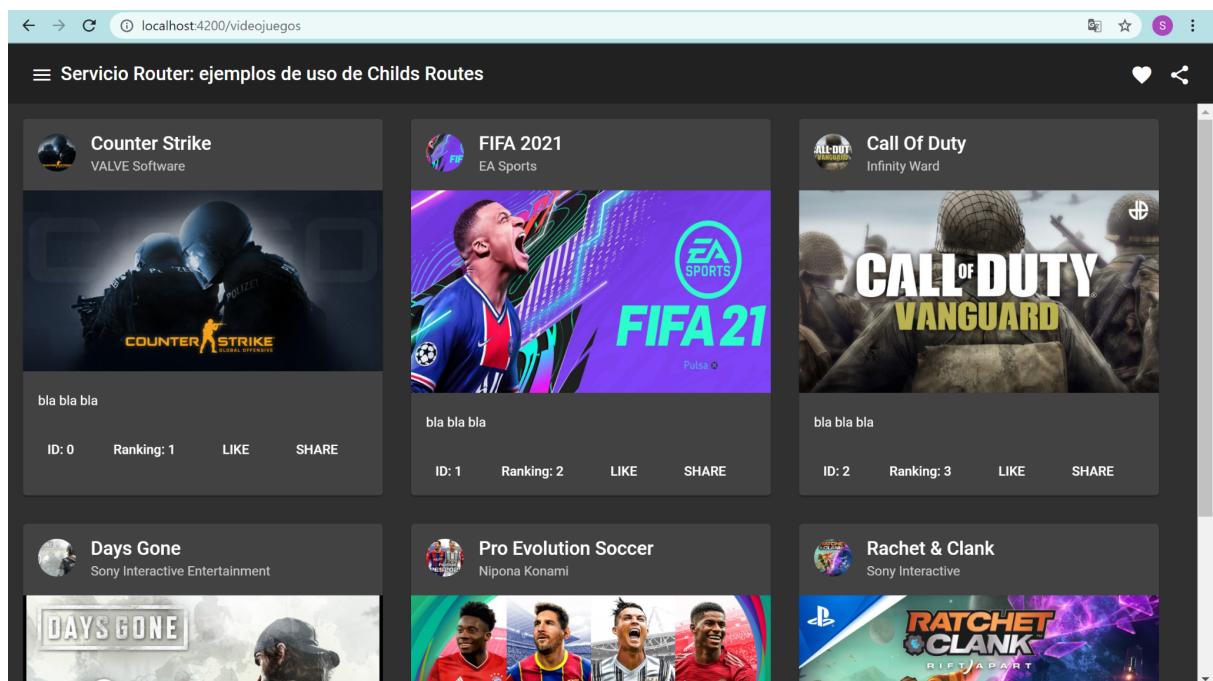
```

<mat-sidenav-container>
  <mat-sidenav #sidenav opened="false" mode="side">
    <mat-nav-list>
      <a mat-list-item routerLinkActive="active-link"
        routerLink="/inicio">Inicio</a>
      <a mat-list-item routerLinkActive="active-link"
        routerLink="/videojuegos">Videojuegos</a>
      <a mat-list-item routerLinkActive="active-link"
        routerLink="/jugadores">Jugadores</a>
      <a mat-list-item routerLinkActive="active-link">Help</a>
    </mat-nav-list>
  </mat-sidenav>

  <mat-sidenav-content>
    <div style="height: 88vh;">
      <router-outlet></router-outlet>
    </div>
  </mat-sidenav-content>
</mat-sidenav-container>
<!-- https://www.youtube.com/watch?v=IRzGphrefsY --&gt;
</pre>

```

- 2.2. Llegados a este punto, el componente de videojuegos-lista debería verse así:



3. Routing: Rutas con parámetros y ActivatedRoute

- 3.1. Creamos el componente de libro-detalles
ng g c videojuegos/videojuegos-detalles
- 3.2. Añadimos una ruta para este componente

```
const routes: Routes = [
  {
    path: 'inicio',
    component: InicioComponent
  },
  {
    path: 'videojuegos',
    component: VideojuegosListaComponent
  },
  {
    path: 'videojuegos/:id',
    component: VideojuegosDetallesComponent
  },
  {
    path: 'jugadores',
    component: JugadoresListaComponent
  },
  {
    path: '',
    redirectTo: '/inicio',
    pathMatch: 'full'
  },
  {
    path: '**',
    component: NotFoundError404Component
  }
];
```

En el componente videojuegos-detalles añadiremos el código para primero obtener el identificador del videojuego (parámetro “:id” del URL) y luego para leer el detalle de ese videojuego.

- 3.3. Vamos al videojuegos-detalles.component.ts para:
 - 3.3.1. Crear una propiedad llamada videojuego que siga el VideojuegoModel

3.3.2. Inyectamos la ActivatedRoute en el constructor

3.3.3. Para obtener los parámetros de la ruta, ActivateRoute nos proporciona el observable paramMap. Para obtenerlos, simplemente nos suscribimos al observable a la espera de recibirlos. De esta manera recibiremos el identificador de videojuego.

```
export class VideojuegosDetallesComponent implements OnInit {  
  
    videojuego: VideojuegoModel | undefined;  
  
    constructor(  
        private activatedRoute: ActivatedRoute  
    ) {}  
  
    ngOnInit(): void {  
        this.activatedRoute.paramMap  
            .subscribe((paramMaps: ParamMap) => {  
                let id = Number(paramMaps.get('id'));  
                this.videojuego = VIDEOJUEGOS[id];  
                this.videojuego = VIDEOJUEGOS.find((item) => item.id === id);  
            })  
    }  
}
```

- 3.4. En el template de videojuegos-detalles.component.html añadiremos el código para visualizar el detalle del videojuego

```
<p>videojuegos-detalles works!</p>  
  
<div class="container" *ngIf="videojuego != null || videojuego !=  
undefined">  
  
    <mat-card class="example-card">  
        <mat-card-subtitle>Compañía: {{ videojuego.creador  
}}</mat-card-subtitle>  
        <mat-card-subtitle>ID: {{ videojuego.id }}</mat-card-subtitle>  
        <mat-card-title>{{ videojuego.titulo }}</mat-card-title>  
        <mat-card-content>  
            <p>This card has divider and indeterminate progress as  
            footer</p>  
            <p>{{ videojuego.descripcion }}</p>  
        </mat-card-content>
```

```

<mat-divider inset></mat-divider>
<mat-card-actions>
  <button mat-button>LIKE</button>
  <button mat-button>SHARE</button>
  <button mat-button>Rank: {{ videojuego.ranking }}</button>
</mat-card-actions>
<mat-card-footer>
  <mat-progress-bar mode="indeterminate"></mat-progress-bar>
</mat-card-footer>
</mat-card>

<router-outlet></router-outlet>
</div>

```

- 3.5. Nos da error a priori porque no tenemos el MatProgressBarModule incorporado al angular-material.module.ts

```

import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { MatButtonModule } from '@angular/material/button';
import { MatCheckboxModule } from '@angular/material/checkbox';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatInputModule } from '@angular/material/input';
import { MatProgressSpinnerModule } from
'@angular/material/progress-spinner';
import { MatCardModule } from '@angular/material/card';
import { MatMenuModule } from '@angular/material/menu';
import { MatIconModule } from '@angular/material/icon';
import { MatSidenavModule } from '@angular/material/sidenav';
import { MatListModule } from '@angular/material/list';
import { MatProgressBarModule } from '@angular/material/progress-bar';

@NgModule({
  declarations: [],
  imports: [MatProgressBarModule, MatListModule, MatSidenavModule,
    MatButtonModule,
    MatCheckboxModule, MatToolbarModule, MatInputModule, MatProgressSpinnerModule, MatCardModule, MatMenuModule, MatIconModule, CommonModule],
  exports: [MatProgressBarModule, MatListModule, MatSidenavModule,
    MatButtonModule,
    MatCheckboxModule, MatToolbarModule, MatInputModule, MatProgressSpinnerModule, MatCardModule, MatMenuModule, MatIconModule, CommonModule]
})

```

```
} )  
export class AngularMaterialModule { }
```

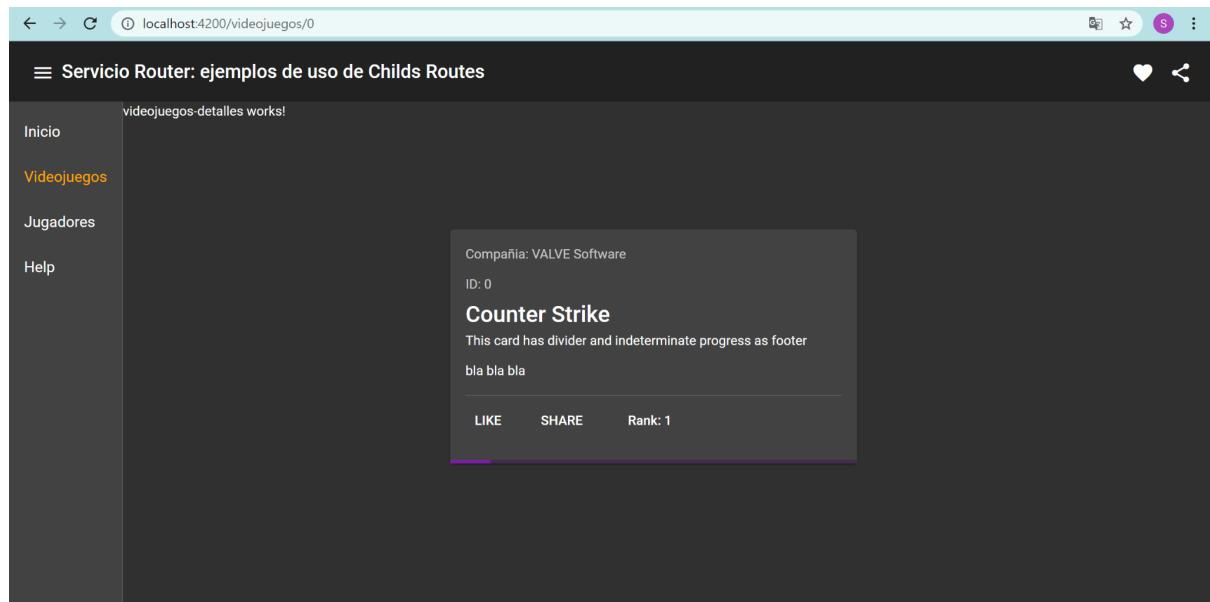
- 3.6. Y, finalmente, añadiremos un enlace o link al componente videojuegos-detalles para cada uno de los videojuegos dentro del videojuegos-lista.component.html:

```
<a [routerLink]=["'/videojuegos', videojuego.id]">  
    
</a>
```

- 3.7. Para el css, algo básico, de momento sólo esto:

```
.example-card {  
  max-width: 400px;  
}  
  
.container {  
  margin-top: 10%;  
  margin-left: 30%;  
}
```

- 3.8. Llegados a este punto, el componente de videojuegos-detalles debería verse así:



4. Routing: child routes

Los path de configuraciones hijos son relativos a los de las configuraciones padre.

Por ejemplo, “/videojuegos/:id/opiniones” sería una Child Route.

Por otra parte, en el template de todo componente padre siempre habrá que añadir la etiqueta router-outlet.

Angular nos proporciona dos maneras de acceder al árbol de rutas activas.

La primera consiste en partir de una ruta activa (ActivatedRoute) cualquiera, y usar sus propiedades parent y children para desplazarnos por el árbol.

Y la segunda consiste en usar la propiedad RouterState. Esta propiedad nos permite obtener el árbol de rutas activas en cualquier momento y lugar de la aplicación.

- 4.1. Vamos a crear un par de componentes de libros:
 - 4.1.1. ng g c videojuegos/videojuegos-opiniones
 - 4.1.2. ng g c videojuegos/videojuegos-imagenes
- 4.2. Seguidamente, en la configuración de rutas, añadiremos las nuevas child routes:

```
{  
  path: 'videojuegos/:id',  
  component: VideojuegosDetallesComponent,  
  children: [  
    {  
      path: 'imagenes',  
      component: VideojuegosImagenesComponent  
    },  
    {  
      path: 'opiniones',  
      component: VideojuegosOpinionesComponent  
    },  
    {  
      path: '',  
      redirectTo: 'imagenes',  
      pathMatch: 'full'  
    },  
    {  
      path: '**',  
      component: NotFoundError404Component  
    }  
  ]  
},
```

- 4.3. En el template del componente padre, videojuegos-detalles.component.html, añadiremos dos enlaces a estas nuevas rutas y la etiqueta router-outlet para visualizar sus componentes asociados:

```

<p>videojuegos-detalles works!</p>

<div class="container" *ngIf="videojuego != null || videojuego != undefined">

    <mat-card class="example-card">
        <mat-card-subtitle>Compañía: {{ videojuego.creador }}</mat-card-subtitle>
        <mat-card-subtitle>ID: {{ videojuego.id }}</mat-card-subtitle>
        <mat-card-title>{{ videojuego.titulo }}</mat-card-title>

        <mat-card-content>
            <p>This card has divider and indeterminate progress as footer</p>
            <p>{{ videojuego.descripcion }}</p>
        </mat-card-content>

        <mat-divider inset></mat-divider>

        <mat-card-actions>
            <button mat-button>LIKE</button>
            <button mat-button>SHARE</button>
            <button mat-button>Rank: {{ videojuego.ranking }}</button>
        </mat-card-actions>

        <mat-card-footer>
            <mat-progress-bar mode="indeterminate"></mat-progress-bar>
            <mat-progress-bar mode="indeterminate"></mat-progress-bar>
        </mat-card-footer>
    </mat-card>

    <mat-selection-list class="mat-list" #videojuegos
[multiple]="false">
        <mat-list-option routerLinkActive="active-link"
routerLink='imagenes'>
            Imágenes
        </mat-list-option>

```

```

<mat-list-option routerLinkActive="active-link"
routerLink='opiniones'>
    Opiniones
</mat-list-option>
</mat-selection-list>

<router-outlet></router-outlet>
</div>

```

4.3.1. Y para el css...

```

.example-card {
    max-width: 400px;
}

.container {
    margin-top: 10%;
    margin-left: 30%;
}

.mat-list {
    width: 33.8vw;
}

.active-link {
    color: orange;
}

```

- 4.4. A continuación, añadiremos el código necesario en videojuegos-imagenes para cargar el identificador de videojuego que serviría al componente para cargar las imágenes relacionadas.

Para ello, haremos uso de ActivatedRoute primero para acceder al ActivatedRoute parent, y luego para obtener el valor del parámetro.

4.4.1. Creamos la propiedad idVideojuego de tipo number, y lo ponemos como undefined

4.4.2. Inyectamos el ActivatedRoute en el constructor

4.4.3. Obtenemos el id de cada videojuego con ActivatedRoute

```

export class VideojuegosImagenesComponent implements OnInit {

    idVideojuego: number | undefined;
}

```

```

constructor(
  private activatedRoute: ActivatedRoute
) { }

ngOnInit(): void {
  this.activatedRoute.parent?.paramMap
    .subscribe((paramMaps: ParamMap) => {
      this.idVideojuego = Number(paramMaps.get('id'));
    })
}
}

```

- 4.5. Y en libro.imagenes.component.html ponemos...

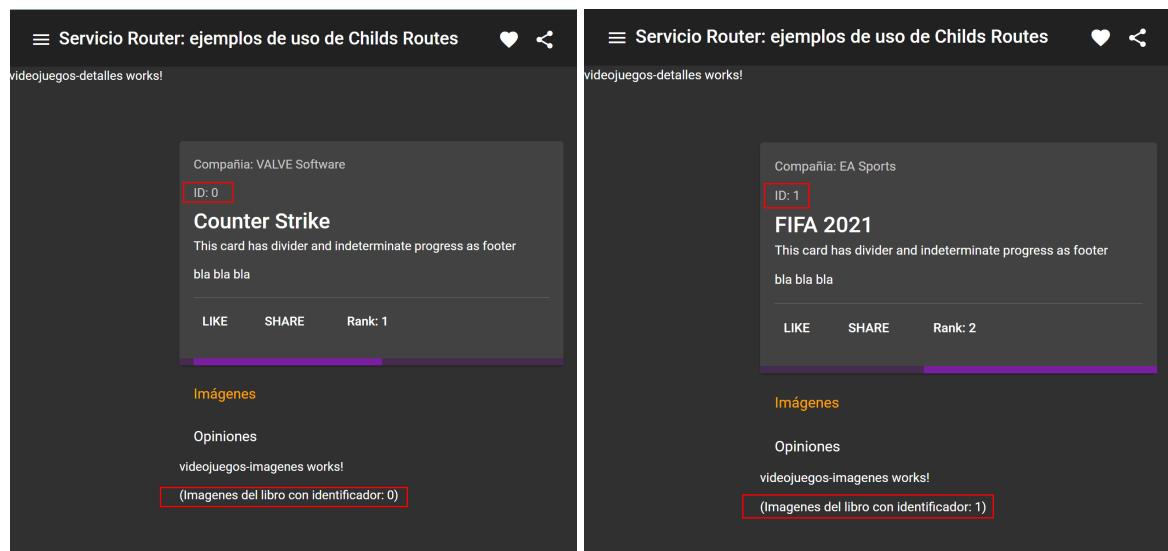
```

<p>videojuegos-imagenes works!</p>

<p>
  (Imagenes del libro con identificador: {{idVideojuego}})
</p>

<!-- <div class="container mt-3" *ngIf="videojuego != null || videojuego != undefined">
  
</div> -->

```



5. Extras

- 5.1. Vamos a conseguir mostrar las imágenes de las portadas de cada videojuego
- 5.2. Vamos al libro-imagenes.component.ts para añadir:

- 5.2.1. Una propiedad llamada libro que es un objeto del LibroModel indefinido
- 5.2.2. Obtenemos el id de ese libro

```
export class VideojuegosImagenesComponent implements OnInit {  
  
    idVideojuego: number | undefined;  
  
    videojuego: VideojuegoModel | undefined;  
  
    constructor(  
        private activatedRoute: ActivatedRoute  
    ) {}  
  
    ngOnInit(): void {  
        this.activatedRoute.parent?. paramMap  
            .subscribe((paramMaps: ParamMap) => {  
                this.idVideojuego = Number(paramMaps.get('id'));  
  
                let idVideojuego = Number(paramMaps.get('id'));  
                this.videojuego = VIDEOJUEGOS[idVideojuego];  
            })  
    }  
}
```

- 5.2.3. En el libro-imagenes.component.html añadimos lo siguiente para mostrar la portada de cada libro

```
<p>videojuegos-imagenes works!</p>  
  
<p>  
    (Imagenes del libro con identificador: {{idVideojuego}})  
</p>  
  
<div class="container mt-3" *ngIf="videojuego != null || videojuego !=  
undefined">
```

```


</div>

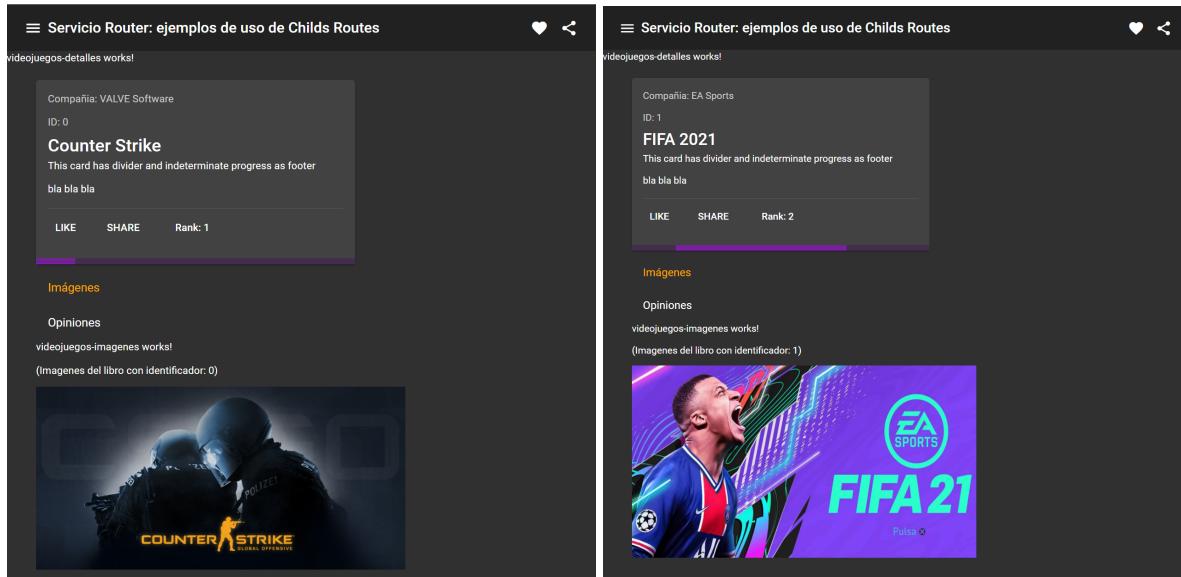
```

5.2.3. En el libro-imagenes.component.css añadimos lo siguiente para mostrar un poco mejor la portada de cada libro

```

.videojuegos-imagenes-img {
  display: flex;
  flex-wrap: wrap;
  width: 500px;
  height: 250;
}

```



Nota: Capturas de pantalla tomadas con el zoom del Chrome al 80%

- 5.3. Ahora vamos a conseguir mostrar las opiniones de cada videojuego

5.3.1. ng g interface interfaces/opinion-model

5.3.2. En ese archivo de opinion-model.ts añadimos lo siguiente:

```

export interface OpinionModel {
  id: number,
  idVideojuego: number,
  titulo: string,
  nombreUser: string,
}

```

```
    photoUser: string,  
    descripcion: string  
}
```

5.3.3. Vamos al libro-mock.ts para exportar una nueva constante para las opiniones:

```
export const OPINIONES: OpinionModel[] = [  
  {  
    id: 0,  
    idVideojuego: 0,  
    titulo: 'Pedazo de juego',  
    nombreUser: 'Álvaro Esmoris',  
    photoUser: '../../assets/Alvaro-Esmoris.jpg',  
    descripcion: 'Me ha gustado mucho este juego, y está a muy buen  
precio'  
  },  
  {  
    id: 1,  
    idVideojuego: 1,  
    titulo: 'Videojuego preferido',  
    nombreUser: 'Sergio Díaz',  
    photoUser: '../../assets/Sergio-Diaz.jpg',  
    descripcion: 'Una buena chutada como los de antes jajaj'  
  },  
  {  
    id: 2,  
    idVideojuego: 2,  
    titulo: 'Es muy corto',  
    nombreUser: 'Gerard Piqué',  
    photoUser: '../../assets/Gerard-Pique.jpg',  
    descripcion: 'No me ha gustado porque el modo historia es  
demasiado corto... ESTAFADORES!',  
  },  
  {  
    id: 3,  
    idVideojuego: 3,  
    titulo: 'Que guapo está este juegaso no?',  
    nombreUser: 'Jaime Sánchez',  
    photoUser: '../../assets/Jaime-Sánchez.jpg',  
  }]
```

```

        descripcion: 'Me ha gustado mucho este juego, y está a muy buen
precio'
    },
    {
        id: 4,
        idVideojuego: 4,
        titulo: 'Me gusta mucho esta web, pero los juegos son
malísimos',
        nombreUser: 'Don JoseMª Polavieja',
        photoUser: '../../../../../assets/Polavieja.jpg',
        descripcion: 'Muy bien hecho Sergio, excelente trabajo!'
    },
    {
        id: 5,
        idVideojuego: 5,
        titulo: 'Yo es que soy más del antiguo snake en el móvil',
        nombreUser: 'Don Alberto Arjona',
        photoUser: '../../../../../assets/Don-Alberto.jpg',
        descripcion: 'Aunque me dan buenos recuerdos de cuando me
echaba sus carreras en el NeedForSpeed de la ps2',
    }
];

```

5.4.4. Ahora, vamos al videojuegos-opiniones.component.ts para añadir:

- 5.4.4.1. una propiedad del tipo objeto del VideojuegoModel siendo indefinido
- 5.4.4.2. otra propiedad de opiniones del tipo OpinionModel siendo un Array
- 5.4.4.3. Inyectar el ActivatedRoute en el constructor
- 5.4.4.4. Creamos una función en el ngOnInit() para obtener las opiniones a través de los IDs de cada videojuego
- 5.4.4.5. Creamos abajo del ngOnInit() un método para obtener las opiniones de cada videojuego, el cual llamaremos dentro de la función del ngOnInit()

```

export class VideojuegosOpinionesComponent implements OnInit {

    videojuego: VideojuegoModel | undefined;

    opiniones: OpinionModel[] = [];
}

```

```

constructor(
  private activatedRoute: ActivatedRoute
) { }

ngOnInit(): void {
  this.activatedRoute.parent?.paramMap
    .subscribe((paramMaps: ParamMap) => {
      let id = Number(paramMaps.get('id'));
      this.videojuego = VIDEOJUEGOS[id];
    })
    // una vez que hemos definido el método para las opiniones, lo
    llamamos aquí
    this.opinionesDeCadaVideojuego();
}

opinionesDeCadaVideojuego(): void {
  this.opiniones = OPINIONES.filter(
    (item) => item.idVideojuego == this.videojuego?.id
  );
}

}

```

5.4.5. Luego vamos al videojuegos-opiniones.component.html para insertar esto:

```

<p>videojuegos-opiniones works!</p>

<div class="container" *ngIf="opiniones != null || opinione
undefined">
  <mat-card class="example-card" *ngFor="let opinion of opinione
">

    <mat-card-title-group>
      <mat-card-title>{{ opinion.titulo }}</mat-card-title>

      <mat-card-subtitle>{{ opinion.nombreUser }}</mat-card-subtitle>

      
    </mat-card-title-group>

    <mat-card-content>
      {{ opinion.descripcion }}
    </mat-card-content>
  </mat-card>
</div>

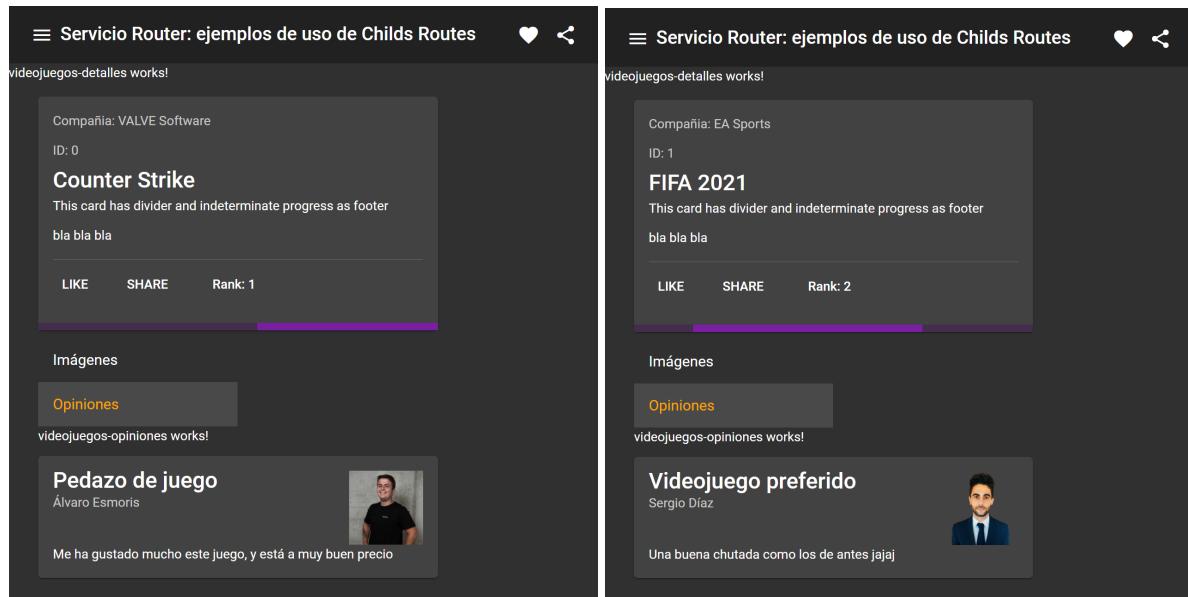
```

```
</mat-card-content>

</mat-card>
</div>
```

5.4.6. Por último, vamos al videojuegos-opiniones.component.css para insertar esto:

```
.example-card {
    max-width: 400px;
    margin-bottom: 8px;
}
```



6. Lazy Loading

- 6.1. Vamos a crear un par de módulos para los componentes de la carpeta de videojuegos

```
ng g m videojuegos/videojuegos --flat  
ng g m videojuegos/videojuegos-routing --flat
```

6.1.1. Ahora vamos al app-routing.module.ts para comentar toda la ruta y los hijos de videojuegos, quedando de esta manera:

```
const routes: Routes = [  
  {  
    path: 'inicio',  
    component: InicioComponent  
  },  
  {  
    path: 'videojuegos',  
    // component: VideojuegosListaComponent  
    loadChildren: () => import('./videojuegos/videojuegos.module')  
      .then((m) => m.VideojuegosModule)  
  },  
  // {  
  //   path: 'videojuegos/:id',  
  //   component: VideojuegosDetallesComponent,  
  //   children: [  
  //     {  
  //       path: 'imagenes',  
  //       component: VideojuegosImagenesComponent  
  //     },  
  //     {  
  //       path: 'opiniones',  
  //       component: VideojuegosOpinionesComponent  
  //     },  
  //     {  
  //       path: '',  
  //       redirectTo: 'imagenes',  
  //       pathMatch: 'full'  
  //     },  
  //     {  
  //       path: '**',  
  //       component: NotFoundError404Component  
  //     }  
  // }
```

```

    ],
    ],
    {
      path: 'jugadores',
      component: JugadoresListaComponent
    },
    {
      path: '',
      redirectTo: '/inicio',
      pathMatch: 'full'
    },
    {
      path: '**',
      component: NotFoundError404Component
    }
];

```

Nota: También, comentamos el componente de videojuegos, para hacer la función que haga conectar con las rutas hijas.

6.1.2. Lo que hemos comentado en app-routing.module.ts lo vamos a copiar y pegar en el nuevo videojuegos-routing.module.ts

```

import { NgModule } from '@angular/core';
// import { CommonModule } from '@angular/common';
import { RouterModule, Routes } from '@angular/router';
import { VideojuegosDetallesComponent } from
'./videojuegos-detalles/videojuegos-detalles.component';
import { VideojuegosImagenesComponent } from
'./videojuegos-imagenes/videojuegos-imagenes.component';
import { VideojuegosOpinionesComponent } from
'./videojuegos-opiniones/videojuegos-opiniones.component';
import { NotFoundError404Component } from
'../components/not-found-error404/not-found-error404.component';

const routes: Routes = [
  {
    path: '',
    component: VideojuegosListaComponent
  },
  {
    path: 'videojuegos/:id',
    component: VideojuegosDetallesComponent,
  }
];

```

```

    children: [
      {
        path: 'imagenes',
        component: VideojuegosImagenesComponent
      },
      {
        path: 'opiniones',
        component: VideojuegosOpinionesComponent
      },
      {
        path: '',
        redirectTo: 'imagenes',
        pathMatch: 'full'
      },
      {
        path: '**',
        component: NotFoundError404Component
      }
    ]
  },
]

@NgModule({
  declarations: [],
  imports: [
    // CommonModule
    RouterModule.forChild(routes)
  ],
  exports: [
    RouterModule
  ]
})
export class VideojuegosRoutingModule { }

```

Nota: Tenemos que añadir una nueva primera ruta tipo default en el caso de que en la URL no se ponga nada

```
{
  path: '',
  component: VideojuegosListaComponent
},
```

Además, también comentamos el import del CommonModule (porque no lo necesitamos aquí), y en su lugar, añadimos `RouterModule.forChild(routes)`

También añadiremos, después de los imports, un export, para exportar el RouterModule

6.1.3. Ahora, en el videojuegos.module.ts importamos el videojuegos-routing.module.ts

```
@NgModule({
  declarations: [],
  imports: [
    CommonModule,
    VideojuegosRoutingModule
  ]
})
```

6.1.4. Y por último, en videojuegos-lista.component.html, quitamos la barra de la ruta videojuegos

```
<!-- <a [routerLink]="'/videojuegos', videojuego.id]" -->
<a [routerLink]="'videojuegos', videojuego.id]>
  
</a>
```

6.1.5. Y ya vuelve a funcionar todo correctamente como antes !!

- 6.2. Ahora, vamos hacer lo mismo para los jugadores ...

```
ng g m jugadores/jugador --flat
ng g m jugadores/jugador-routing --flat
```

6.2.1. Ahora vamos al app-routing.module.ts para comentar la ruta hacia el componente de los jugadores, y poniendo en su lugar la función de loadChildren: () que nos conecta con las rutas hijas de jugadores.

```
const routes: Routes = [
  {
    path: 'inicio',
```

```
        component: InicioComponent
    },
{
    path: 'videojuegos',
    // component: VideojuegosListaComponent
    loadChildren: () =>
import('../videojuegos/videojuegos.module').then((m) =>
m.VideojuegosModule)
},
// {
//     path: 'videojuegos/:id',
//     component: VideojuegosDetallesComponent,
//     children: [
//         {
//             path: 'imagenes',
//             component: VideojuegosImagenesComponent
//         },
//         {
//             path: 'opiniones',
//             component: VideojuegosOpinionesComponent
//         },
//         {
//             path: '',
//             redirectTo: 'imagenes',
//             pathMatch: 'full'
//         },
//         {
//             path: '**',
//             component: NotFoundError404Component
//         }
//     ]
// },
{
    path: 'jugadores',
    // component: JugadoresListaComponent
    loadChildren: () => import('../jugadores/jugador.module').then((m) => m.JugadorModule)
},
{
    path: '',
    redirectTo: '/inicio',
    pathMatch: 'full'
},
```

```
{
  path: '**',
  component: NotFoundError404Component
}
];
```

6.2.2. La ruta de jugadores que hemos modificado (la original que había) en app-routing.module.ts la tenemos que poner ahora en el nuevo jugador-routing.module.ts

```
import { NgModule } from '@angular/core';
// import { CommonModule } from '@angular/common';
import { RouterModule, Routes } from '@angular/router';
import { JugadoresListaComponent } from
'./jugadores-lista/jugadores-lista.component';

const routes: Routes = [
  {
    path: '',
    component: JugadoresListaComponent
  },
]

@NgModule({
  declarations: [],
  imports: [
    // CommonModule
    RouterModule.forChild(routes)
  ],
  exports: [
    RouterModule
  ]
})
export class JugadorRoutingModule { }
```

Nota: La ruta hacia JugadoresListaComponent está vacía, siendo así la ruta default si en la URL no se pusiera nada.

Al igual que con cuando VideojuegosRoutingModule, comentamos el CommonModule y ponemos en su lugar, `RouterModule.forChild(routes)`

También exportamos el RouterModule.

6.2.3. Ahora, en el jugador.module.ts importamos el jugador-routing.module.ts

```
@NgModule ({  
  declarations: [],  
  imports: [  
    CommonModule,  
    JugadorRoutingModule  
  ]  
})
```

6.2.4. Y con esto, ya vuelve a funcionar todo correctamente !!

Nota: Tener en cuenta que, en el autor-lista.component.html, aún no tenemos nada...

- 6.3. Vamos a crear los jugadores, y la vista para ellos (jugadores-lista.component.html), así como su controlador (jugadores-lista.component.ts)

6.3.1. Vamos a jugadores-lista.component.html para poner lo siguiente:

```
<link rel="stylesheet"  
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css"  
      integrity="sha512-5A8nwdMOWrSz20fDsJczgUi dUBR8liPYU+WymTZP1lmY9G6Oc7H1Zv156XqnsgNUzTyMeffFTcsFH/tnJE/+xBg=="  
      crossorigin="anonymous"  
      referrerPolicy="no-referrer" />  
  
<p>jugadores-lista works!</p>  
  
<div class="body">  
  <div class="container" *ngIf="jugadores != null || jugadores != undefined">  
  
    <div class="cards" *ngFor="let jugador of jugadores">  
      <div class="imgBx">  
          
      </div>  
  
      <div class="content">  
        <div class="details">  
          <h2>
```

```

        UserName
        <br>
        <span>Puesto Trabajo</span>
    </h2>

    <ul class="social_icons">
        <li><a href="#"><i class="fa fa-facebook" aria-hidden="true"></i></a></li>
        <li><a href="#"><i class="fa fa-twitter" aria-hidden="true"></i></a></li>
        <li><a href="#"><i class="fa fa-linkedin" aria-hidden="true"></i></a></li>
        <li><a href="#"><i class="fa fa-instagram" aria-hidden="true"></i></a></li>
    </ul>
</div>
</div>

</div>
</div>

```

6.3.2. Vamos a crear una interface que haga de modelo para los jugadores

ng g interface interfaces/jugador-model

```

export interface JugadorModel {
    id: number;
    nombre: string;
    nickname: string;
    puesto: string;
    descripcion: string;
    ranking: number;
    imagen: string;
}

```

6.3.3. Creamos su mock manualmente en la carpeta de mocks (jugador-mock.ts)

```

import { JugadorModel } from "../interfaces/jugador-model";

export const JUGADORES: JugadorModel[] = [

```

```
{  
    id: 0,  
    nombre: "Álvaro Esmoris",  
    nickname: "VaroMoris",  
    puesto: "Professional JSP Dev",  
    descripcion: "Soy compañero de Sergio en DWC",  
    ranking: 1,  
    imagen: "../../assets/Alvaro-Esmoris.jpg"  
},  
{  
    id: 1,  
    nombre: "Sergio Díaz",  
    nickname: "Royal",  
    puesto: "Expert Angular Dev",  
    descripcion: "Soy compañero de Álvaro en DWC",  
    ranking: 2,  
    imagen: "../../assets/Sergio-Diaz.jpg"  
},  
{  
    id: 2,  
    nombre: "Gerard Piqué",  
    nickname: "Catalufo",  
    puesto: "Jugador del Barça FC",  
    descripcion: "Soy jugador profesional del Barcelona",  
    ranking: 3,  
    imagen: "../../assets/Gerard-Pique.jpg"  
},  
{  
    id: 3,  
    nombre: "Jaime Sánchez",  
    nickname: "Jeimy",  
    puesto: "Repetiendo en 1º DAW",  
    descripcion: "Soy compañero de Sergio en Programación",  
    ranking: 4,  
    imagen: "../../assets/Jaime-Sanchez.jpg"  
},  
{  
    id: 4,  
    nombre: "D. José Mª García Polavieja",  
    nickname: "Polavieja",  
    puesto: "Profesor de DWC",  
    descripcion: "Soy profesor de Sergio en DWC",  
    ranking: 5,
```

```

        imagen: "../../assets/Polavieja.jpg"
    },
{
    id: 5,
    nombre: "D. Alberto Arjona",
    nickname: "Don Alberto",
    puesto: "Profesor de Programación",
    descripcion: "Soy profesor de Sergio en Programación",
    ranking: 6,
    imagen: "../../assets/Don-Alberto.jpg"
},
]

```

6.3.4. Vamos a jugadores-lista.component.ts para poner lo siguiente:

```

export class JugadoresListaComponent implements OnInit {

  jugadores: JugadorModel[] = [];

  constructor() { }

  ngOnInit(): void {
    this.jugadores = JUGADORES;
  }
}

```

6.3.5. Volvemos al jugadores-lista.component.html para interpolar los datos del mock de jugador-mock.ts

```

<div class="body">
  <div class="container" *ngIf="jugadores != null || jugadores != undefined">

    <div class="cards" *ngFor="let jugador of jugadores">
      <div class="imgBx">
        
        <div class="details">
          <h2>

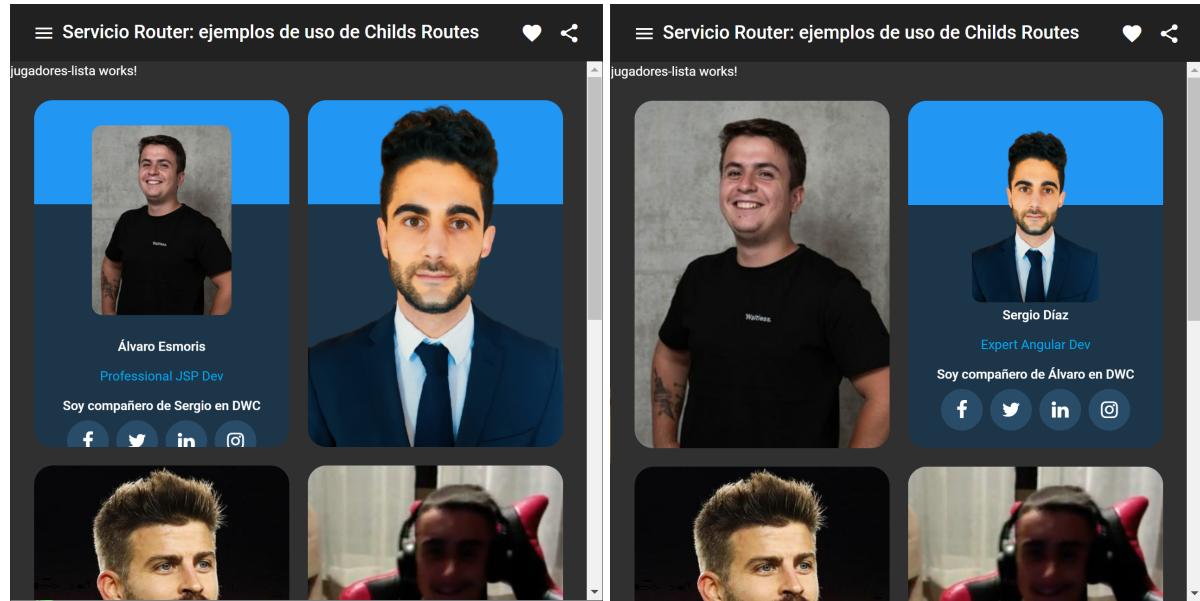
```

```

        {{ jugador.nombre }}
        <br>
        <span>{{ jugador.puesto }}</span>
        <br>
        {{ jugador.descripcion }}
    </h2>

    <ul class="social_icons">
        <li><a href="#"><i class="fa fa-facebook" aria-hidden="true"></i></a></li>
            <li><a href="#"><i class="fa fa-twitter" aria-hidden="true"></i></a></li>
                <li><a href="#"><i class="fa fa-linkedin" aria-hidden="true"></i></a></li>
                    <li><a href="#"><i class="fa fa-instagram" aria-hidden="true"></i></a></li>
                </ul>
            </div>
        </div>
    </div>
</div>

```



- 6.4. Vamos a crear ahora la vista de los detalles de los jugadores, al igual que hicimos antes con los detalles de los videojuegos...

6.4.1. Creamos el componente de jugadores-detalles
 ng g c jugadores/jugadores-detalles

6.4.2. Añadimos una ruta para este componente directamente como child route en el jugador-routing.module.ts

```
const routes: Routes = [
  {
    path: '',
    component: JugadoresListaComponent
  },
  {
    path: 'jugadores/:id',
    component: JugadoresDetallesComponent
  },
]
```

En el componente jugadores-detalles añadiremos el código para, primero obtener el identificador del jugador (parámetro “:id” del URL) y luego para leer el detalle de ese videojuego.

6.4.3 Vamos al jugadores-detalles.component.ts para:

6.4.4. Crear una propiedad llamada jugador que siga el JugadorModel

6.4.5. Inyectamos la ActivatedRoute en el constructor

6.4.6. Para obtener los parámetros de la ruta, ActivateRoute nos proporciona el observable paramMap. Para obtenerlos, simplemente nos suscribimos al observable a la espera de recibirlos. De esta manera recibiremos el identificador de jugador.

```
export class JugadoresDetallesComponent implements OnInit {

  jugador: JugadorModel | undefined;

  constructor(
    private activatedRoute: ActivatedRoute
  ) { }

  ngOnInit(): void {
    this.activatedRoute.paramMap
      .subscribe((paramMaps: ParamMap) => {
```

```

        let id = Number(paramMaps.get('id'));
        this.jugador = JUGADORES[id];
        this.jugador = JUGADORES.find((item) => item.id === id);
    })
}

}

```

6.4.7. En el template de jugadores-detalles.component.html añadiremos el código para visualizar el detalle del jugador

```

<p>jugadores-detalles works!</p>

<div class="container" *ngIf="jugador != null || jugador != undefined">

    <mat-card class="example-card">
        <mat-card-subtitle>ID: {{ jugador.id }}</mat-card-subtitle>
        <mat-card-title>{{ jugador.nombre }}</mat-card-title>
        <mat-card-subtitle>Nickname: {{ jugador.nickname }}</mat-card-subtitle>
        <mat-card-subtitle>{{ jugador.puesto }}</mat-card-subtitle>
        </mat-progress-bar>
        </mat-card-footer>
    </mat-card>

    <!-- <router-outlet></router-outlet> esto sería en caso de seguir ampliando esta ruta... -->
</div>

```

Nota: OJO! predeterminadamente, al crear el jugadores-detalles.component, éste se declarará por sí sólo en el jugadores-routing.module.ts ... pero es que ahí no debe declararse !! ... elimino esta declaración de aquí y añado la declaración en el app.module.ts

6.4.8. Y, finalmente, añadiremos un enlace o link al componente jugadores-detalles para cada uno de los jugadores dentro del jugadores-lista.component.html:

```
<div class="imgBx">
    <!-- <a [routerLink]="'/jugadores', jugador.id]" -->
    <a [routerLink]="'/jugadores', jugador.id]">
        
    </a>
</div>
```

Nota: OJO! como hemos hecho de esta ruta una child route directamente, no se pone la barra antes de jugadores !!

6.4.9. Para el css de jugadores-detalles.component.css, algo básico, de momento sólo esto:

```
.example-card {
    max-width: 400px;
}

.container {
    margin-top: 2.5%;
    margin-left: 5%;
}

.mat-list {
    width: 33.8vw;
}

.active-link {
    color: orange;
}
```

6.4.10. Llegados a este punto, el componente de jugadores-detalles debería verse así:

The image displays two side-by-side screenshots of a web application interface, likely built with Angular, illustrating the implementation of child routes for player details.

Screenshot 1 (Left): The title bar reads "Servicio Router: ejemplos de uso de Childs Routes". The sidebar on the left has links for "Inicio", "Videojuegos", "Jugadores" (which is highlighted in orange), and "Help". The main content area shows a player profile for "Álvaro Esmoris" (ID: 0) with nickname "VaroMoris", described as a "Professional JSP Dev". It includes a small profile picture of a man in a black t-shirt. Below the profile, a message says "Soy compañero de Sergio en DWC". At the bottom are "LIKE", "SHARE", and "Rank: 1" buttons.

Screenshot 2 (Right): The title bar reads "Servicio Router: ejemplos de uso de Childs Routes". The sidebar on the left is identical to the first screenshot. The main content area shows a player profile for "Sergio Díaz" (ID: 1) with nickname "Royal", described as an "Expert Angular Dev". It includes a small profile picture of a man in a suit. Below the profile, a message says "Soy compañero de Álvaro en DWC". At the bottom are "LIKE", "SHARE", and "Rank: 2" buttons.