




## AE-3 ANGULAR

GERARD PERUJO  
NOELIA VILLAHERMOSA  
MARIA ISABEL MARTÍN SIMAL  
GRUPO 03



## **CONCLUSIÓN**

Angular nos ha gustado a todos, ya que es fácil reutilizar componentes para formar las paginas sin tener que picar otra vez el código.

Lo que nos ha costado mas es hacer que los componentes se enlacen entre unos y otros a través el Router, una vez lo has hecho un par de veces ya era más fácil.

La parte mas difícil ha sido el login, ha costado bastante tener que sacar el nombre del usuario en la página index.

La tarea la hemos realizado Noelia y Gerard juntos por temas de tiempo con el trabajo mientas que Maribel ha preferido hacerla ella por su cuenta para practicar.

Para la entrega hemos escogido el trabajo de María Isabel.

Aquí tienes la ruta del github donde encontraras todos nuestros trabajos:

[https://github.com/Gerard-Perujo/Web\\_EntornoCliente\\_AE3](https://github.com/Gerard-Perujo/Web_EntornoCliente_AE3)

## Parte Gerard y Noelia

En este proyecto una vez creado el proyecto angular con el comando “ng new Gamer” he empezado por crear la clase videojuego donde tendrá toda una serie de variables que necesito para la clase.

```
1 export class Videojuego{
2
3
4
5     public constructor(public id: number, public titulo: string, public compania:string, public valoracion: number){
6         this.id = id;
7         this.titulo = titulo;
8         this.compania = compania;
9         this.valoracion = valoracion;
10    }
11
12    public toString(): string{
13        return `ID ${this.id}, Titulo ${this.titulo}, Compañia ${this.compania}, Valoracion ${this.valoracion}`
14    }
15 }
```

Después de crear la clase Videojuegos creo el componente mi-usuario donde en el TS contendrá una lista de videojuegos donde se plasmarán en el HTML

```
ard > Gamer > src > app > Componentes > mi-usuario > TS mi-usuario.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { Videojuego } from 'src/app/Entidades/videojuego/videojuego';
3 import { ActivatedRoute } from '@angular/router';
4
5 @Component({
6     selector: 'app-mi-usuario',
7     templateUrl: './mi-usuario.component.html',
8     styleUrls: ['./mi-usuario.component.css']
9 })
10 export class MiUsuarioComponent implements OnInit {
11
12     id : number = 0;
13     titulo : string = "";
14     valoracion : number = 0;
15     compania : string = "";
16     nombre : string = ""
17
18     public listaVideojuegos : Videojuego[] = [
19         {id:1, titulo:"Elden Ring", compania:"FromSoftware", valoracion:10},
20         {id:2, titulo:"The Elder ScrollsV: Skyrim", compania:"Bethesda Softworks", valoracion:9.8},
21         {id:3, titulo:"Sekiro: Shadows Die Twice", compania:"FromSoftware", valoracion:9.5},
22         {id:4, titulo:"God of War: Ragnarok", compania:"Santa Monica Studio", valoracion:9.8},
23         {id:5, titulo:"The Witcher 3: Wild Hunt", compania:"CD Projekt RED", valoracion:9.5},
24         {id:6, titulo:"Fallout 3", compania:"Bethesda Softworks", valoracion:9.5},
25         {id:7, titulo:"Dark Souls III", compania:"From Software", valoracion:9.5}
26     ]
27
28 }
29
30 constructor(private ruta : ActivatedRoute) {
31
32 }
33
34 ngOnInit(): void {
35     this.nombre = this.ruta.snapshot.paramMap.get('nombre') || '';
36 }
37
38 }
```

Una vez creado el TS creo el HTML que es donde se plasmaran todos los videojuegos de la lista y plasmarlos dentro de una tabla para poder realizar esta acción hago un ngfor así voy recorriendo toda la lista y saco todos los videojuegos también creo un enlace de ver detalles videojuegos en la cual nos llevara al componente detalle donde muestra en detalle el videojuego seleccionado.

```

1 <div class="lista">
2 <app-navegador></app-navegador>
3 <body class="contenedor">
4   <div class="usuario">
5     <h1>Lista de los Videojuegos en Stock</h1>
6     <h3>Nombre de Usuario: {{nombre}}</h3>
7     <div class="contenido">
8       <div class="tabla">
9         <table>
10           <tr>
11             <td>ID</td>
12             <td>Titulo</td>
13             <td>Valoracion</td>
14           </tr>
15           <tr *ngFor="let v of listaVideojuegos">
16             <td>{{v.id}}</td>
17             <td>{{v.titulo}}</td>
18             <td>{{v.valoracion}}</td>
19           </tr>
20         </table>
21       </div>
22       <div class="links">
23         <label>Opciones</label>
24         <a routerLink="/juego1/1/Elden Ring/FromSoftware/10">Detalles juego</a>
25         <a routerLink="/juego2/2/The Elder Scrolls V: Skyrim/Bethesda Softworks/9.8">Detalles juego</a>
26         <a routerLink="/juego3/3/Sekiro: Shadows Die Twice/From Software/9.5">Detalles juego</a>
27         <a routerLink="/juego4/4/God of War: Ragnarok/Santa Monica Studio/9.8">Detalles juego</a>
28         <a routerLink="/juego5/5/The Witcher 3: Wild Hunt/CD Projekt RED/9.5">Detalles juego</a>
29         <a routerLink="/juego6/6/Fallout 3/Bethesda Softworks/9.5">Detalles juego</a>
30         <a routerLink="/juego7/7/Dark Souls III/From Software/9.5">Detalles juego</a>
31       </div>
32     </div>
33     <button routerLink="/">Ir a login</button>
34   </div>
35 </body>
36 </div>

```

Ahora que tengo ya el componente terminado me creo el componente juego1 que hará referencia al detalle del juego 1 en el cual dentro del TS me creo las variables nombre, titulo, compañía valoración que son los parámetros que trae la ruta para sacar los datos de ese juego y pasmarlos en el HTML también tendré que importar el ActivatedRoute que será el encargado de recoger los parámetros que entran por la ruta.

```

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

@Component({
  selector: 'app-juego1',
  templateUrl: './juego1.component.html',
  styleUrls: ['./juego1.component.css']
})
export class Juego1Component implements OnInit {

  id : number = 0;
  titulo : string = '';
  compania : string = '';
  valoracion : number = 0;

  constructor(route : ActivatedRoute) {
    this.id = route.snapshot.params["id"]
    this.titulo = route.snapshot.params["titulo"]
    this.compania = route.snapshot.params["compania"]
    this.valoracion = route.snapshot.params["valoracion"]
  }

  ngOnInit(): void {
  }
}

```

Una vez realizado el TS me creo el HTML de juego 1 donde voy a sacar una imagen que hace referencia al juego y un detalle donde se plasmaran los parámetros que se envían con la ruta.

```
Go to component
1 <div class="lista">
2 <app-navegador></app-navegador>
3 <body class="contenedor">
4   <div class="juego">
5     <h1>{{titulo}}</h1>
6     <div class="targeta">
7       <div class="imagen">
8         
9       </div>
10      <div class="contenido">
11        <label> ID: {{id}}</label>
12        <label> Titulo: {{titulo}}</label>
13        <label> Compañia: {{compania}} </label>
14        <label> Valoracion: {{valoracion}}</label>
15      </div>
16    </div>
17    <button routerLink="/usuario">Volver</button>
18  </div>
19 </body>
20 </div>
```

Ahora voy a proceder a la creación de los componentes header, footer, login, navegador ya que son componentes que voy a necesitar para formar mi aplicación web.

El **header** va a ser una cabecera que aparecerá siempre con el nombre de la tienda online, este componente no tendrá ningún tipo de lógica solo tendrá html y css para los estilos que los realiza Noelia.

```
<div class="cabecera">
  <h3>GAMER</h3>
  <h4>Tu tienda Online de compra y venta de videojuegos</h4>
</div>
```

El **Footer** va a ser el pie de página de la aplicación web igual que la cabecera también aparecerá siempre y tampoco tendrá lógica solo HTML y CSS para los estilos que los realiza Noelia.

```
Go to component
<body class="contenedor">
  <div class="contenedorFooter">
    <article class="vacio3"></article>
    <article class="privacidad">
      <label>Legal:</label>
      <p>Condiciones de uso</p>
      <p>Politica de privacidad</p>
      <p>Politica de cookies</p>
    </article>
    <article class="vacio4"></article>
    <article class="Contacto">
      <label>Contacto:</label>
      <p>Av Aragon nº3-4 Madrid Cp 28001</p>
      <p>telf: 91 613589457</p>
      <p>infoMots@gmail.com</p>
    </article>
    <article class="vacio5"></article>
  </div>
</body>
```

El **navegador** va estar dentro de todos los componentes menos en el login ya que quiero que solo aparezca cuando se haya iniciado sesión, este componente tampoco tendrá lógica solo HTML y CSS para los estilos que los realiza Noelia.

```
Go to component
<body class="contenedor">
  <nav class="navegador">
    <button routerLink="/usuario">Videojuegos</button>
    <button routerLink="/contacto">Contacto</button>
    <button routerLink="/informacion">Sobre Nosotros</button>
  </nav>
</body>
```

El login será el componente que veremos al iniciar la aplicación web ya que le diremos que tiene la ruta raíz de esta manera cuando se cargue la pantalla aparecerá el componente login este en cambio si que tendrá lógica , HTML y CSS para los estilos que los realiza Noelia.

En la lógica crearemos una lista de usuarios en el caso de que el usuario exista podremos entrar en la página y si no existe saldrá un mensaje de error en pantalla diciendo que ese usuario no existe.

Pero antes de crear el login vamos a tener que crear una clase usuario para poder crear nuestra lista de usuarios.

En esa clase solo tendremos nombre y password que es lo que nos hace falta para poder entrar en la aplicación web

```
export class Usuario{

    public constructor(public nombre:string, public pas:string){
        this.nombre = nombre,
        this.pas = pas
    }

    public toString(): string{
        return `Nombre ${this.nombre}, Password ${this.pas}`
    }
}
```

Una vez creada la clase procedemos a crear el TS del login con nuestra lista de usuarios

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Usuario } from 'src/app/Entidades/usuario/usuario';

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

  nombre: string = ""
  pas: string = ""
  mensaje: string = ""

  private listaUsuarios : Usuario[] = [
    {nombre: 'Jose', pas: '1111'},
    {nombre: 'Jorge', pas: '2222'},
    {nombre: 'Anna', pas: '3333'},
    {nombre: 'Julia', pas: '4444'}
  ]

  constructor( private ruta: Router ) {
  }

  public acceder(): void{
    if (this.comprobar(this.nombre, this.pas)){
      this.ruta.navigate(['/usuario', {nombre: this.nombre}])
    }else{
      this.mensaje = "Usuario y contraseña Incorrectas"
    }
  }

  public comprobar (nombre : string, pas: string): boolean{
    return this.listaUsuarios.some((usuario) => usuario.nombre === nombre && usuario.pas === pas);
  }

  ngOnInit(): void {
  }
}
```

Fijaros que aquí en el TS a parte de crear la lista de usuarios también creo 2 métodos públicos uno que comprueba si el nombre y usuario pasados existen en la lista de usuarios.

Y otro que es acceder que en caso de que el nombre y la password existan en la lista usuarios te manda al componente usuario pasándole el nombre usuario ya que queremos sacar el nombre

Usuario dentro del componente mi-usuario.

Ahora vamos a crear el HTML en el cual crearemos un formulario donde se escribirá el nombre y la password vamos a utilizar el ngModel para que así esos nombres que nos escriban los coja la variable que tenemos dentro el TS

```
Go to component
<div class="lista">
  <body class="contenedor">
    <div class="login">
      <h1>Bienvenido</h1>
      <h4>Introduce tu Nombre de usuario y Password para poder Entrar</h4>
      <label>Nombre: </label>
      <input type="text" [(ngModel)]="nombre"/>
      <br/>
      <label>Contraseña: </label>
      <input type="password" [(ngModel)]="pas"/>
      <br/>
      <div class="botones" (click)="acceder()">
        <button> Entrar</button>
      </div>
      <h4 class="error">{{mensaje}}</h4>
    </div>
  </body>
</div>
```



Ahora vamos a montar el app-component que será nuestra pagina de inicio cuando arranque nuestra aplicación web en el cual pondremos la cabecera el router y el footer.

El router será el componente que ira cambiando de pantalla mostrando los componentes que tenemos enlazados con rutas.

```
<body class="contenedor">
  <app-header class="header"></app-header>
  <router-outlet class="main"></router-outlet>
  <app-footer class="footer"></app-footer>
</body>
```

Llegado a este punto y antes de avanzar mas con la creación del proyecto angular me creo las rutas dentro del app-routing para poder probar si todo lo que he creado hasta ahora funciona correctamente y también me creo ya todas las rutas que voy a tener que usar para el correcto funcionamiento de la pagina.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LoginComponent } from '../Componentes/login/login.component';
import { MiUsuarioComponent } from '../Componentes/mi-usuario/mi-usuario.component';
import { Juego1Component } from '../Componentes/juego1/juego1.component';
import { ContactoComponent } from '../Componentes/contacto/contacto.component';
import { SobreMiComponent } from '../Componentes/sobre-mi/sobre-mi.component';
import { Juego2Component } from '../Componentes/juego2/juego2.component';
import { Juego3Component } from '../Componentes/juego3/juego3.component';
import { Juego4Component } from '../Componentes/juego4/juego4.component';
import { Juego5Component } from '../Componentes/juego5/juego5.component';
import { Juego6Component } from '../Componentes/juego6/juego6.component';
import { Juego7Component } from '../Componentes/juego7/juego7.component';

const routes: Routes = [
  {path: '', component: LoginComponent},
  {path: 'usuario', component: MiUsuarioComponent},
  {path: 'juego1/:id/:titulo/:compania/:valoracion', component: Juego1Component},
  {path: 'juego2/:id/:titulo/:compania/:valoracion', component: Juego2Component},
  {path: 'juego3/:id/:titulo/:compania/:valoracion', component: Juego3Component},
  {path: 'juego4/:id/:titulo/:compania/:valoracion', component: Juego4Component},
  {path: 'juego5/:id/:titulo/:compania/:valoracion', component: Juego5Component},
  {path: 'juego6/:id/:titulo/:compania/:valoracion', component: Juego6Component},
  {path: 'juego7/:id/:titulo/:compania/:valoracion', component: Juego7Component},
  {path: 'contacto', component: ContactoComponent},
  {path: 'informacion', component: SobreMiComponent }
];

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

Una vez creadas las rutas arranco la aplicación con el comando `ng -o serve` y pruebo que el login aparezca en pantalla, y cuando pulse el botón entrar entro en usuario y si pulso el botón de ver detalles del primer juego me manda al componente juego1.

Al comprobar que todo funciona correctamente procedo a crear el resto de los componentes que serán los otros 6 componentes más que harán referencia a los otros juegos.

## Noelia

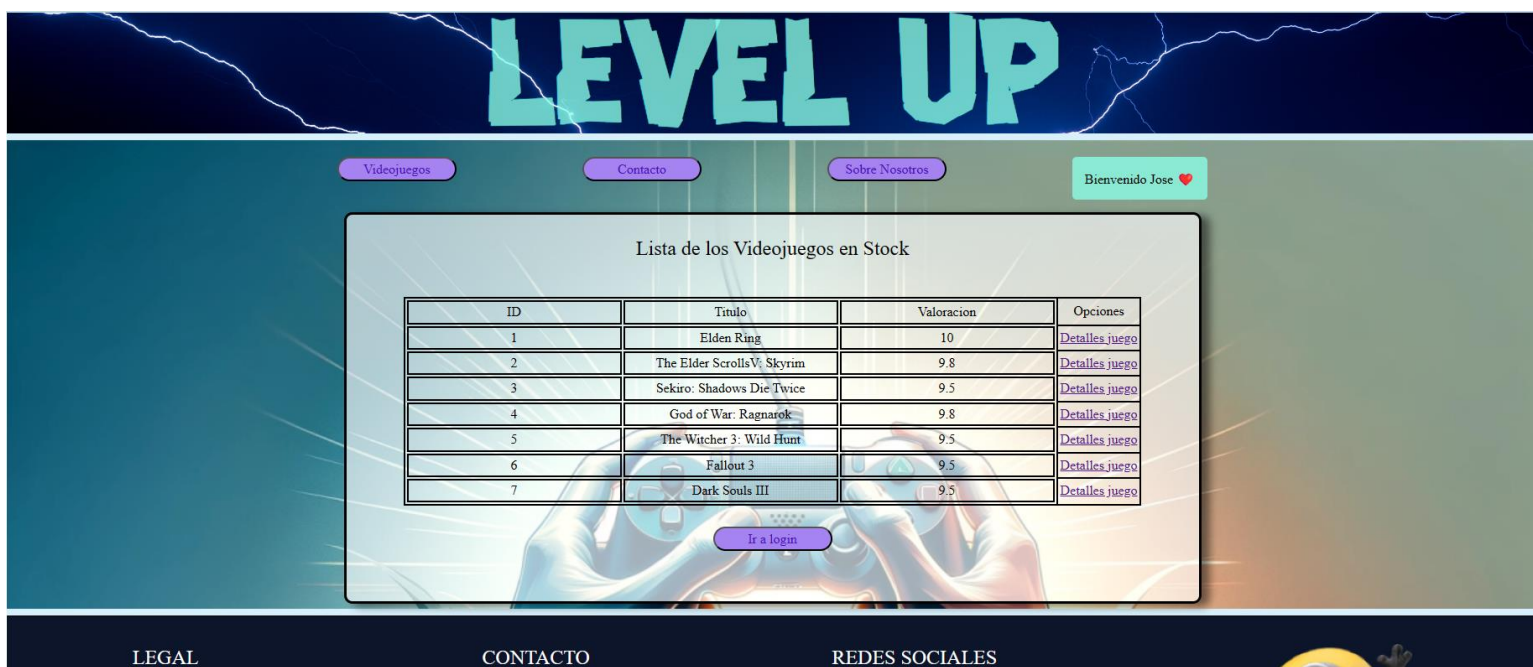
La parte que he realizado ha sido CSS y añadir el nombre de usuario al navegador para que aparezca siempre en cada página al navegar por la web.

El resultado ha sido en base a la aplicación de videojuegos de mi compañero Gerard.

### LOGIN



Al iniciar sesión nos dirigimos a la página de videojuegos donde se puede observar también el nombre de usuario.



## CONTACTO

# LEVEL UP

[Videojuegos](#)[Contacto](#)[Sobre Nosotros](#)[Bienvenido ❤️](#)

### CONTACTO

telf: 91 613589457

insideGamer@gmail.com

Av Aragon nº3-4 Madrid Cp 28001



### LEGAL

- > Condiciones de uso
- > Política de privacidad
- > Política de cookies

### CONTACTO

📍 Av León nº4 Madrid Cp 13260

📞 telf: +34 613333214

✉️ infoinsideGamer@gmail.com

### REDES SOCIALES

@insideGamers



## SOBRE NOSOTROS

# LEVEL UP

[Videojuegos](#)[Contacto](#)[Sobre Nosotros](#)[Bienvenido ❤️](#)

### Nuestra Historia

En Level Up, no simplemente vendemos productos, creamos experiencias. Somos más que una tienda, somos un destino para los apasionados del entretenimiento que buscan llevar su experiencia al siguiente nivel. En el corazón de Level Up late la pasión por el mundo del gaming y el entretenimiento. Fundada por un grupo de entusiastas que comparten la visión de transformar la forma en que experimentas tus momentos de ocio, nuestra tienda nació con el propósito de ofrecer a nuestros clientes una selección cuidadosamente curada de productos que reflejan la calidad, innovación y diversión que buscamos en cada uno de nuestros productos.

### Variedad para Todos los Gustos

Sabemos que cada gamer, tiene gustos únicos. Es por eso que en Level Up ofrecemos una amplia variedad de productos para satisfacer todas las preferencias. Desde los últimos lanzamientos en videojuegos hasta accesorios innovadores y productos de colección, estamos aquí para asegurarnos de que encuentres lo que necesitas para elevar tu experiencia a nuevos niveles.

### Comunidad Level Up

Más allá de ser una tienda, queremos ser tu comunidad. En Level Up, celebramos la diversidad de nuestra comunidad de jugadores y apasionados del entretenimiento. Únete a nosotros en eventos exclusivos, comparte tus experiencias y descubre nuevas formas de conectarte con otros que comparten tu misma pasión. En Level Up, estamos comprometidos con la misión de elevar cada momento de entretenimiento. Gracias por elegiros como tu destino para la excelencia en entretenimiento. ¡Bienvenido a una experiencia que va más allá de las expectativas! Eleva tu juego con Level Up. ¡Descubre, juega, vive!



**Maria Isabel Martín Simal**

## **Requerimiento 1:**

A continuación se va a explicar el proceso que se ha debido seguir para realizar la actividad. La mayoría de explicaciones se encuentra en el código comentado justo encima de las clases o métodos correspondientes, por lo que aquí se hará más bien una recapitulación.

Para comenzar, se debe crear la carpeta donde se va a almacenar el proyecto y arrancarlo con `ng serve -o`, una vez que se ha accedido a la subcarpeta de videojuegos mediante `'cd'`. Tras eso se crean los primeros componentes, como el index, el sobre nosotros y el contacto mediante `'ng generate component + nombre_componente'`. Para evitar que la aplicación pese más de lo necesario, se elimina el archivo `'spec'` de los componentes.

Una vez definidos estos componentes, se procede a crear su enrutamiento en `'routing.module'`, es decir, especificar el nombre de la ruta de cada página. Tras esto, se pueden personalizar las páginas con los estilos y funcionalidad deseada. Por ejemplo, así era la estructura básica del index sin apenas cambios y sin haber añadido videojuegos:



**Listado de videojuegos**



Con el paso del tiempo y de las necesidades específicas, los diseños se fueron actualizando.

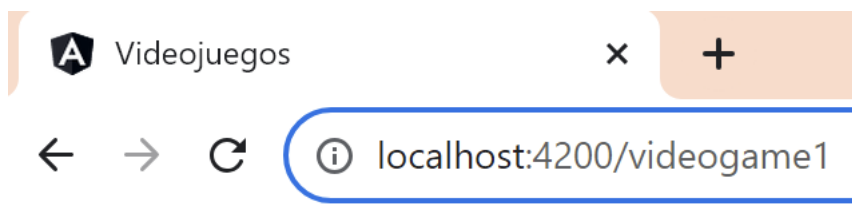
En el .ts del index se crean los videojuegos con sus propios atributos personalizados.

```
constructor() {  
  let videojuego: Videojuego = new Videojuego(1, "Nintendogs", "Nintendo", 96);  
  this.listado.push(videojuego);  
  
  videojuego = new Videojuego(2, "Animal Crossing: New Horizons", "Nintendo", 95)  
  this.listado.push(videojuego);  
}
```

Tras tener esta parte lista, ya era momento de probar con uno de los videojuegos. Para ello, primero se realiza por una ruta que luego cambiará de nombre, ya que necesita incorporar los valores personalizados de cada videojuego. El enrutamiento se debe actualizar.

Maribel > videojuegos > src > app > TS app-routing.module.ts > routes

```
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { ContactoComponent } from '../contacto/contacto.component';
4 import { AboutComponent } from '../about/about.component';
5 import { IndexComponent } from '../index/index.component';
6 import { LoginComponent } from '../login/login.component';
7 import { Videogame1Component } from '../videogame1/videogame1.component';
8
9 const routes: Routes = [
10   {path: 'contacto', component: ContactoComponent},
11   {path: 'about', component: AboutComponent},
12   {path: '', redirectTo: '/index', pathMatch: 'full'},
13   {path: 'index', component: IndexComponent },
14   {path: 'login', component: LoginComponent},
15   {path: 'videogame1', component: Videogame1Component}
16 ]
17
```



videogame1 works!

```
const routes: Routes = [
  {path: 'contacto', component: ContactoComponent},
  {path: 'about', component: AboutComponent},
  {path: '', redirectTo: '/login', pathMatch: 'full'},
  {path: 'index', component: IndexComponent },
  {path: 'login', component: LoginComponent},
  {path: 'videogame1/:id/:titulo/:compania/:valoracion', component: Videogame1Component},
  {path: 'videogame2', component: Videogame2Component},
  {path: 'videogame3', component: Videogame3Component},
  {path: 'videogame4', component: Videogame4Component}
]
```



```
export class Videogame1Component implements OnInit {
  id: number = 0;
  titulo: string = "";
  compania: string = "";
  valoracion: number = 0;

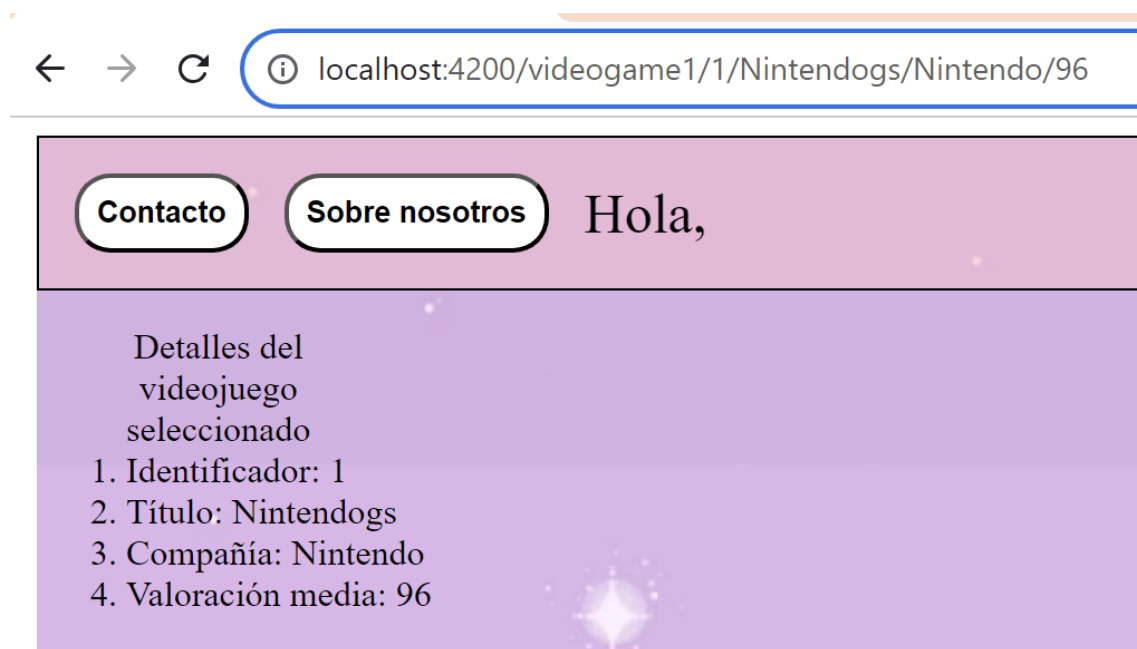
  constructor(route: ActivatedRoute) {
    this.id = route.snapshot.params["id"];
    this.titulo = route.snapshot.params["titulo"];
    this.compania = route.snapshot.params["compania"];
    this.valoracion = route.snapshot.params["valoracion"];
  }
}
```

Una vez enlazados los parámetros, para que en el listado aparezcan solo los atributos que demanda el ejercicio se debe hacer un bucle for que recorra el listado ya creado. En cada columna figurará un atributo diferente iterado, con la particularidad del ver detalle, al que se le pasa el routerLink que se completará igualmente con las iteraciones, creando rutas personalizadas con los parámetros para cada videojuego.

```
<tr *ngFor="let game of listado">
  <td>{{game.id}}</td>
  <td>{{game.titulo}}</td>
  <td>{{game.valoracion}}</td>
  <td><a routerLink="/videogame/{{game.id}}/{{game.id}}/{{game.titulo}}/{{game.compania}}/{{game.valoracion}}/">Ver detalle</a></td>
</tr>
```

Al hacer clic en el videojuego nos lleva a sus detalles debidamente representados y con la ruta definitiva:





Una vez comprobado que para un juego funciona, la mecánica es la misma para el resto. El diseño debe ser actualizado, y el resultado final se mostrará unas hojas más adelante. Con posterioridad también se añadieron varios botones para facilitar la navegabilidad entre páginas.

## Requerimiento 2

Al igual que sucedió en el anterior requerimiento, la mayoría de comentarios se encuentra en el propio código. Para este segundo requerimiento se debe hacer un login. Para ello, primero se crea la clase Usuario con los datos que serán necesarios para ejecutar satisfactoriamente un login: el nombre de usuario y la contraseña. Tras esto se crea y se modifica el componente de login.

---

login works!

Para poder llevar a cabo la mecánica del login, se debe añadir una clase que verifique o autentifique si un usuario figura o no en la lista de usuarios que se va a definir en esa misma

clase junto con sus respectivas contraseñas. El resultado de dicha verificación se almacena de manera local. Luego en el ts del login se especifica su funcionalidad, es decir, dependiendo de lo que devuelva el método que verifica la existencia o no de un usuario, si este existe entonces somos redirigidos automáticamente al index; en caso contrario saltará un mensaje alertando de que el usuario o la contraseña no son correctos.

Una vez finalizado el login, el último paso es lograr que aparezca en todas las páginas el nombre de usuario actual de la sesión. Para ello, en los ts de los componentes, una variable recibe el nombre del usuario que se ha almacenado de manera local. Dicho nombre lo devuelve el método definido en la clase de autenticación de usuarios. La variable debe ser interpolada en la plantilla HTML de cada página en la que se quiere mostrar el nombre del usuario.



Tras algunas pequeñas modificaciones posteriores, este es el **resultado final**:



[Contacto](#) [Sobre nosotros](#) [Listado](#) [Volver al login](#) ¡Hola, mario!

Listado de videojuegos

Identificador	Título	Valoración media	🔗 Enlaces con sorpresa 🔗
1	Nintendogs	96	<a href="#">¡Ver más detalles de Nintendogs!</a>
2	Animal Crossing: Wild World	95	<a href="#">¡Ver más detalles de Animal Crossing: Wild World!</a>
3	Grand Theft Auto: San Andreas	97	<a href="#">¡Ver más detalles de Grand Theft Auto: San Andreas!</a>
4	Outlast	95	<a href="#">¡Ver más detalles de Outlast!</a>
5	Rayman	97	<a href="#">¡Ver más detalles de Rayman!</a>
6	Donkey Kong 64	95	<a href="#">¡Ver más detalles de Donkey Kong 64!</a>
7	Diddy Kong Racing	95	<a href="#">¡Ver más detalles de Diddy Kong Racing!</a>
8	Super Mario Galaxy	97	<a href="#">¡Ver más detalles de Super Mario Galaxy!</a>
9	Pokémon Snap	91	<a href="#">¡Ver más detalles de Pokémon Snap!</a>
10	Tomb Raider II	90	<a href="#">¡Ver más detalles de Tomb Raider II!</a>

Activar Windows  
Ve a Configuración para activar Windows.

[Contacto](#) [Sobre nosotros](#) [Listado](#) [Volver al login](#) ¡Hola, mario!



¿Necesitas ayuda?

Nombre:

Correo Electrónico:

Comentarios:

Enviar mi comentario

¡Volver al listado!

Activar Windows  
Ve a Configuración para activar Windows.

Animal Crossing World





[Contacto](#) [Sobre nosotros](#) [Listado](#) [Volver al login](#) ¡Hola, mario!

¡Detalles del juego!

Identificador	Título	Compañía	Valoración media
2	Animal Crossing: Wild World	Nintendo	95

¡Volver al listado!



Activar Windows  
Ve a Configuración para activar Windows.

[Contacto](#) [Sobre nosotros](#) [Listado](#) [Volver al login](#) ¡Hola, mario!

¡Detalles del juego!

Identificador	Título	Compañía	Valoración media
3	Grand Theft Auto: San Andreas	Rockstar Games	97

¡Volver al listado!



Activar Windows  
Ve a Configuración para activar Windows.

← → ↻ localhost:58934/video4/4/Outlast/Red%20Barrels/95

Contato Sobre nosotros Listado Volver al login ¡Hola, mario!

### ¡Detalles del juego!

Identificador	Título	Compañía	Valoración media
4	Outlast	Red Barrels	95

¡Volver al listado!



Activar Windows  
Ve a Configuración para activar Windows.

← → ↻ localhost:58934/video5/5/Rayman/Ubisoft/97

Contato Sobre nosotros Listado Volver al login ¡Hola, mario!

### ¡Detalles del juego!

Identificador	Título	Compañía	Valoración media
5	Rayman	Ubisoft	97

¡Volver al listado!



Activar Windows  
Ve a Configuración para activar Windows.

[Contacto](#) [Sobre nosotros](#) [Listado](#) [Volver al login](#) ¡Hola, mario!

### ¡Detalles del juego!

Identificador	Título	Compañía	Valoración media
6	Donkey Kong 64	Nintendo	95

¡Volver al listado!



Activar Windows  
Ve a Configuración para activar Windows.

[Contacto](#) [Sobre nosotros](#) [Listado](#) [Volver al login](#) ¡Hola, mario!

### ¡Detalles del juego!

Identificador	Título	Compañía	Valoración media
7	Diddy Kong Racing	Rare	95

¡Volver al listado!



Activar Windows  
Ve a Configuración para activar Windows.



[Contacto](#) [Sobre nosotros](#) [Listado](#) [Volver al login](#) ¡Hola, mario!

¡Detalles del juego!

Identificador	Título	Compañía	Valoración media
8	Super Mario Galaxy	Nintendo	97

¡Volver al listado!



Activar Windows  
Ve a Configuración para activar Windows.

[Contacto](#) [Sobre nosotros](#) [Listado](#) [Volver al login](#) ¡Hola, mario!

¡Detalles del juego!

Identificador	Título	Compañía	Valoración media
9	Pokémon Snap	Nintendo	91

¡Volver al listado!



Activar Windows  
Ve a Configuración para activar Windows.



## Extras

Nótese que los botones se modifican al pasar el cursor por encima de ellos, tanto los del navegador, como los botones que hay en el cuerpo de la página. Además, en el caso del contacto, el sobre nosotros y el listado, sus respectivos botones del navegador están desactivados cuando nos encontramos en sus páginas, ya que no tiene sentido intentar acceder a una página en la que ya estamos.

Como último añadido, cada vez que se hace clic en ver los detalles de un videojuego suena automáticamente una canción relacionada, siendo sus duraciones desde unos segundos hasta un par de minutos en algunos casos. Soy consciente que la reproducción automática de sonidos está totalmente desaconsejada en aplicaciones reales, pero al tratarse de un trabajo académico he querido investigar cómo se reproducen sonidos.