**Imagen que contiene Icono

Descripción generada automáticamente**

**INTRODUCCIÓN**

En este trabajo, desarrollamos una aplicación con Angular y usamos GitHub como herramienta clave que potencian la eficiencia en el desarrollo de proyectos colaborativos. Esta última plataforma de desarrollo colaborativo, nos facilitó la gestión y colaboración en el código fuente de manera fluida y transparente.

El trabajo en equipo ha sido fundamental para fomentar la creatividad y abordar la resolución de problemas, permitiendo la combinación de habilidades individuales para alcanzar objetivos colectivos. Además, las reuniones periódicas proporcionaron un espacio crucial para la alineación de objetivos, la revisión de avances y la toma de decisiones conjuntas.

A través del link que se detalla a continuación, se puede acceder al trabajo finalizado. Este archivo tiene el nombre de “GRUPO04”:

<https://github.com/Xukialb/AE-3.-Angular.git>

**Desarrollo de la Actividad por Alberto Saboya Ocaña:**

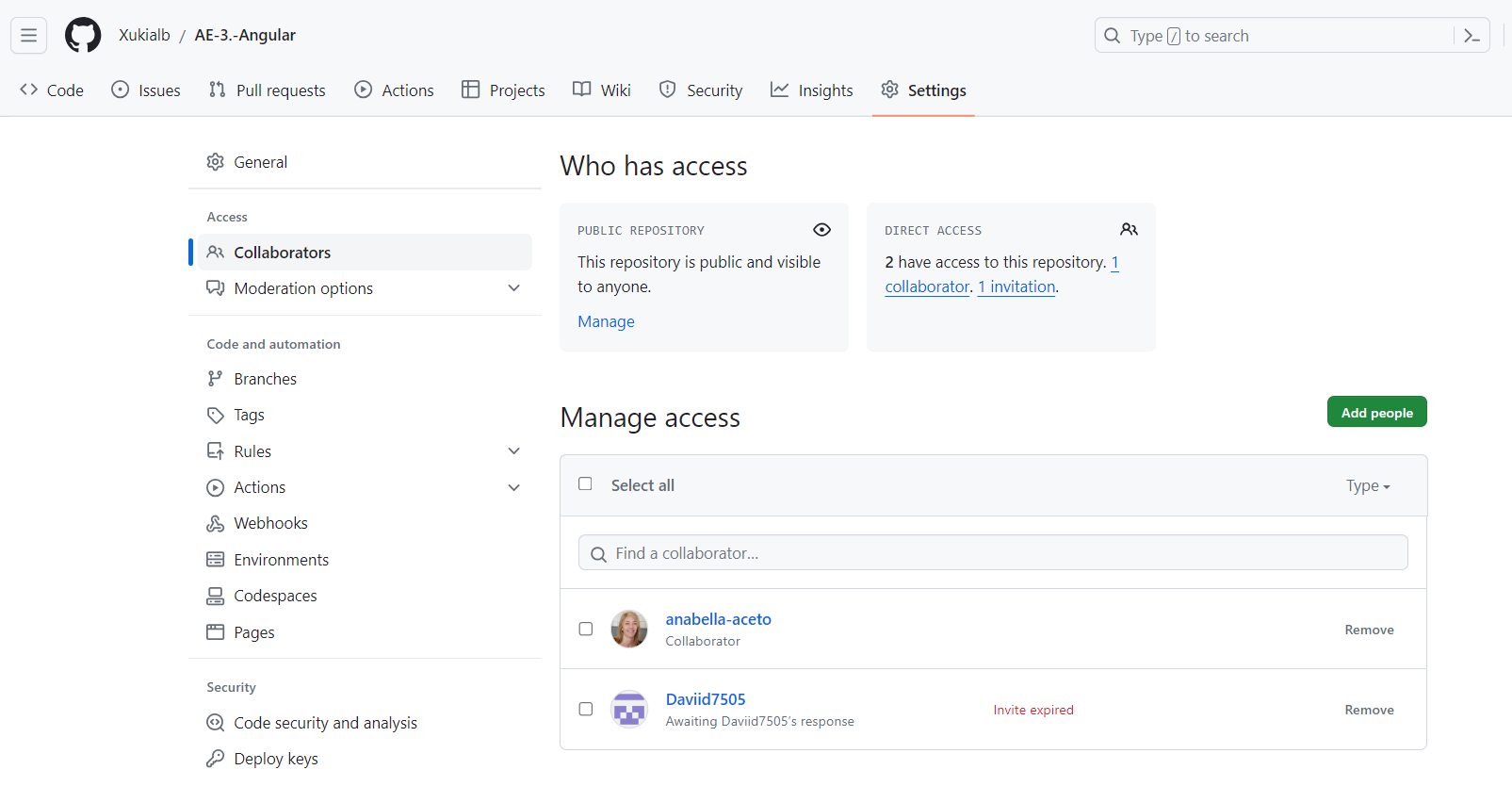
Para la realización de este ejercicio, se ha creado un repositorio público en GitHub al que se accede a través del siguiente enlace:

<https://github.com/Xukialb/AE-3.-Angular>

También se puede clonar a partir de este enlace:

<https://github.com/Xukialb/AE-3.-Angular.git>

En ese repositorio se han agregado como colaboradores al resto de componentes del grupo:



**Requerimiento 1:**

Se pide realizar una aplicación en Angular que muestre un listado-detalle de Videojuegos.

Los videojuegos tendrán un identificador, un título, una compañía, una imagen y una valoración media.

La web mostrará una página con nuestro nombre de usuario y el listado de videojuegos (tendremos algunos videojuegos ya dados de alta), pero solo se mostrará en el listado el identificador, el título y la valoración media.

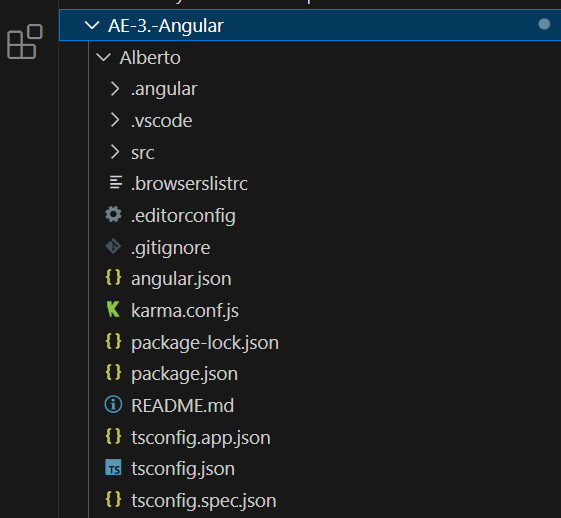
Si pulsamos en un videojuego, nos llevará a su detalle, el cual podremos ver todos los atributos del videojuego. Esta página tendrá también un botón para volver.

La página web tendrá también una página que muestre los datos para contactarnos y otra página que hable sobre nosotros. A estas páginas accederemos mediante un menú de navegación (también en este menú tendremos el listado de videojuegos).

Se valorarán los estilos en la aplicación, aplicar “routing” y comunicación entre componentes.

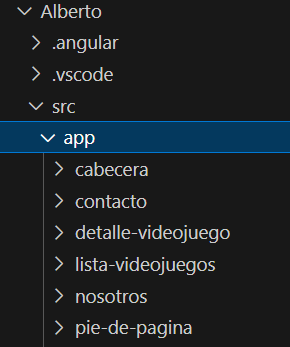
**Resolución de la actividad:**

1. En una carpeta del ordenador **clono el repositorio de GitHub**
2. **En Visual Studio Code, importo el workspace** indicándole la ruta de la carpeta creada en el paso anterior.
3. **Creo una aplicación angular** llamada Alberto con el comando “ng new Alberto” y contestamos a todas las cuestiones yes, sobre todo a Would you like to add Angular routing?:



1. **Cambio al directorio de la nueva aplicación** con el comando “cd Alberto”:



Creamos los componentes con el comando “ng g c nombre del componente”:

ng g c lista-videojuegos

ng g c detalle-videojuego

ng g c contacto

ng g c nosotros

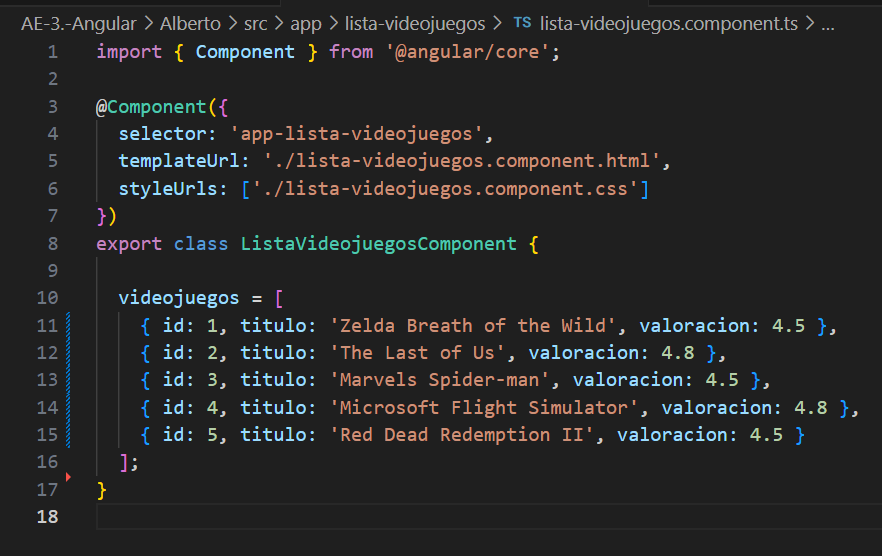
ng g c cabecera

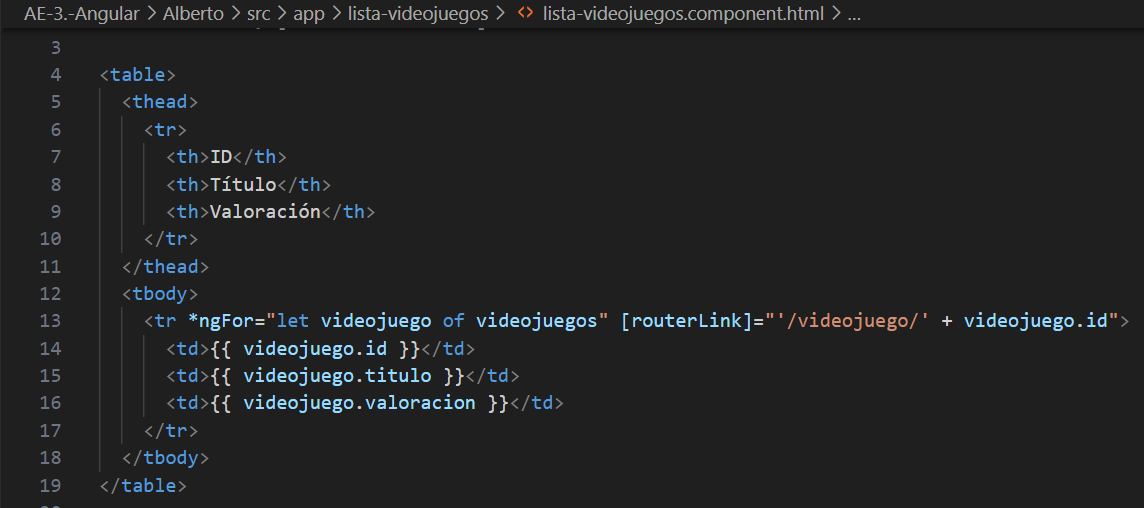
ng g c pie-de-pagina

1. **Configuro las rutas (routing)** editando el archivo **src/app/app-routing.module.ts** para configurar las rutas:

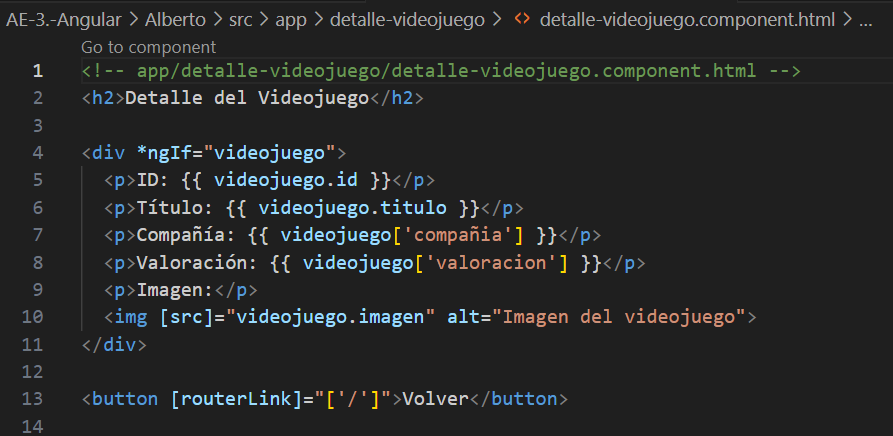


1. **Implemento el componente lista-videojuegos** editando los archivos HTML y TypeScript dentro de sus respectivas carpetas:





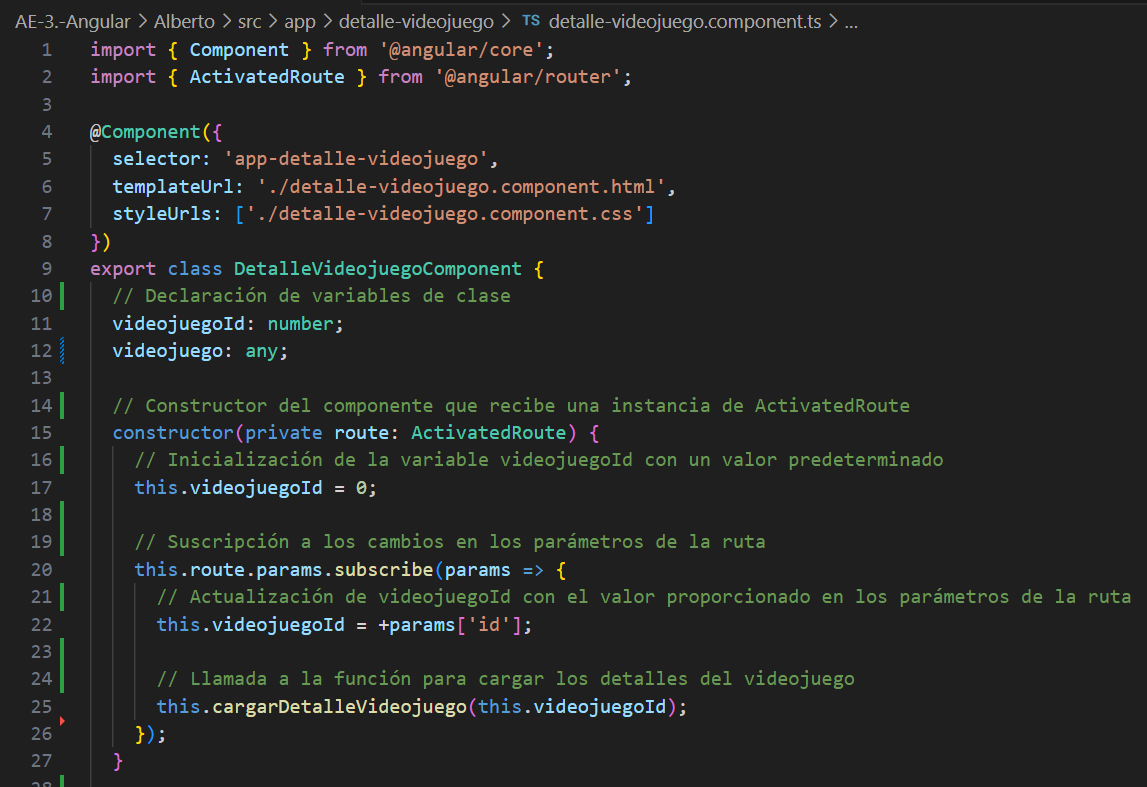
1. **Implemento el componente detalle-videojuego** editando los archivos HTML y TypeScript dentro de sus respectivas carpetas:



Observaciones:

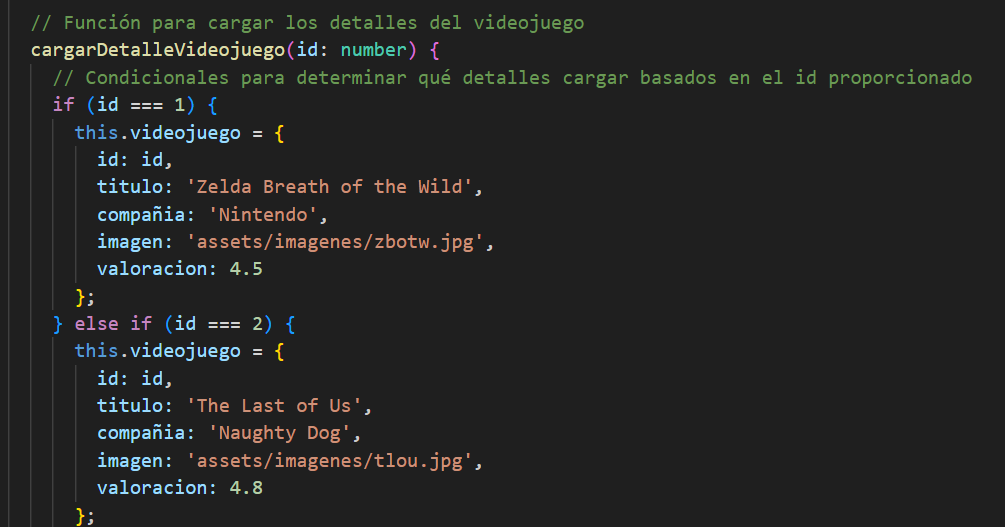
**\*ngIf="videojuego"** evalúa si la variable videojuego tiene un valor que no sea null ni undefined. Si videojuego tiene un valor (es decir, no es nulo ni indefinido), entonces el contenido dentro del <div> se mostrará en el DOM. Si videojuego es nulo o indefinido, el contenido dentro del <div> no se mostrará en la página.

**[routerLink]=”[‘/’]”** es para que el router-outlet cambie a la ruta (path) de inicio. Utiliza **Property Binding** para establecer la ruta de redireccionamiento.

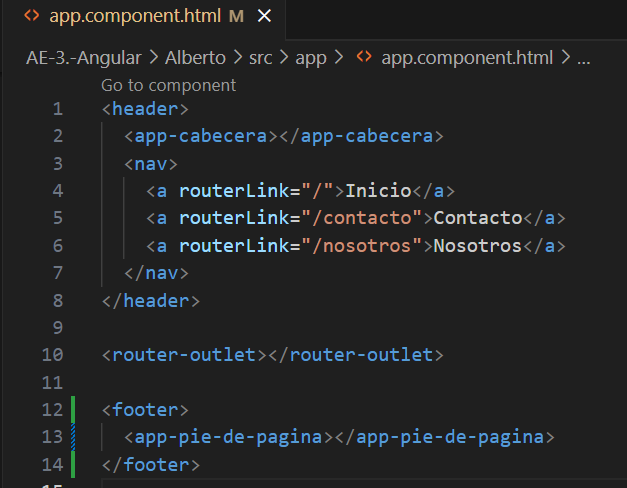


Observaciones:

**@Component** es un decorador que se utiliza para configurar metadatos del componente

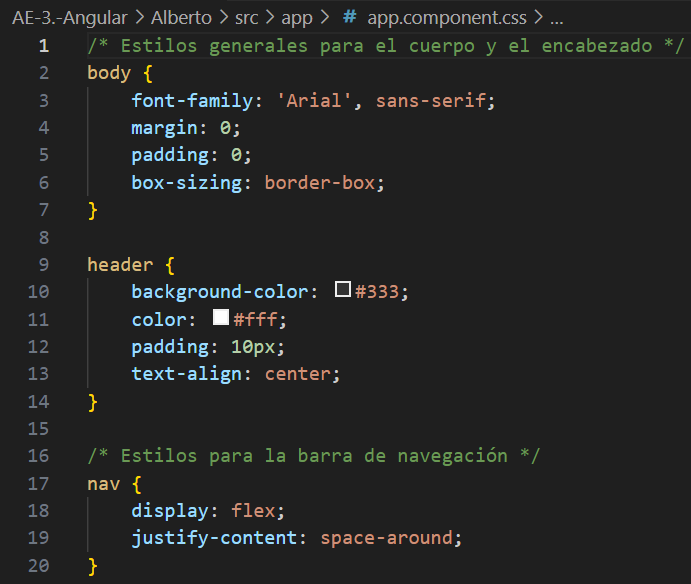


1. **Actualizo la plantilla de la aplicación principal** editando el archivo **app.component.html** y **app.component.css** para definir como quiero que se muestre la aplicación y en qué orden quiero que aparezcan los componentes:

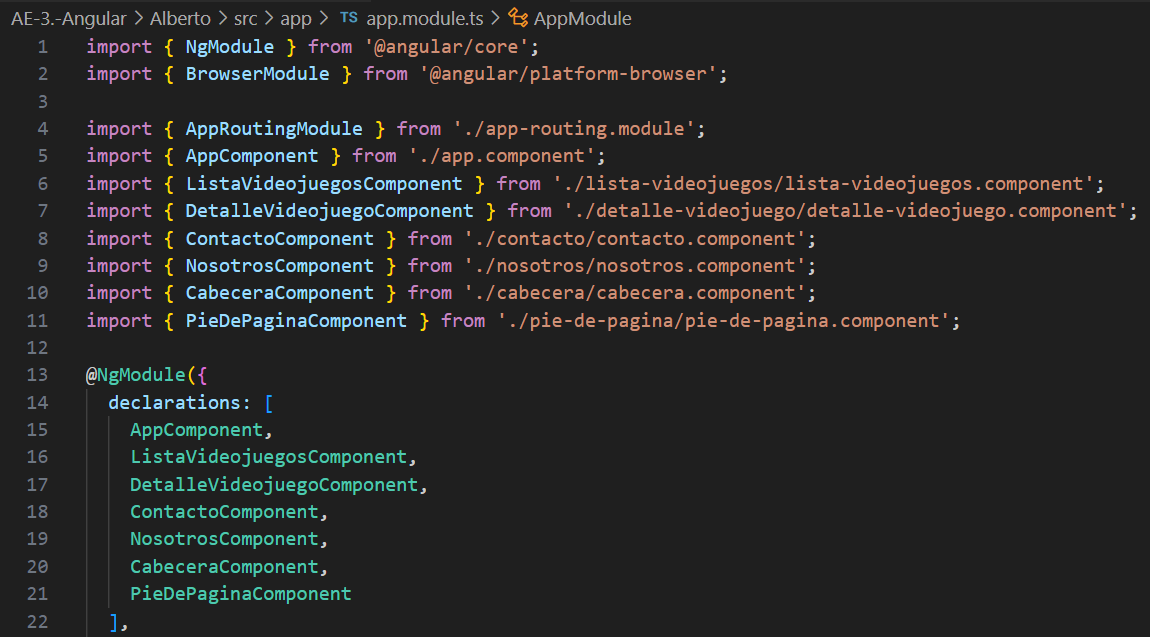


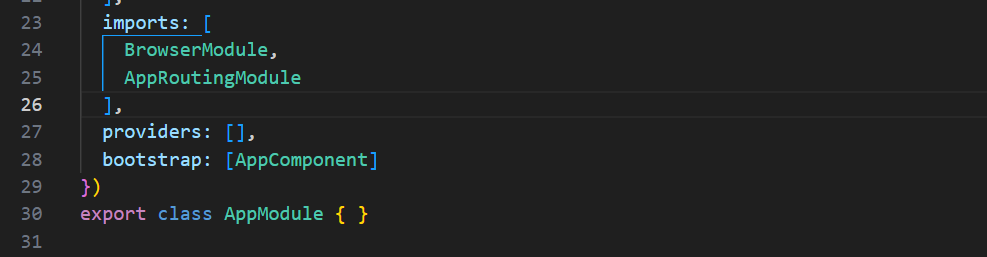
Observaciones:

Se realizan tres enlaces de enrutamiento que lleva a las páginas de inicio, contacto y nosotros.



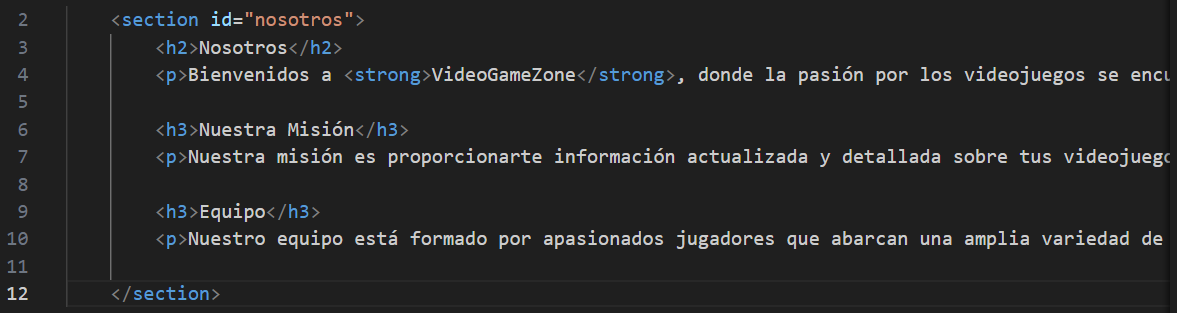
1. **Configuro la aplicación principal** abriendo **src/app/app.module.ts** y me aseguro de que AppRoutingModule esté importado y listado en la sección imports.

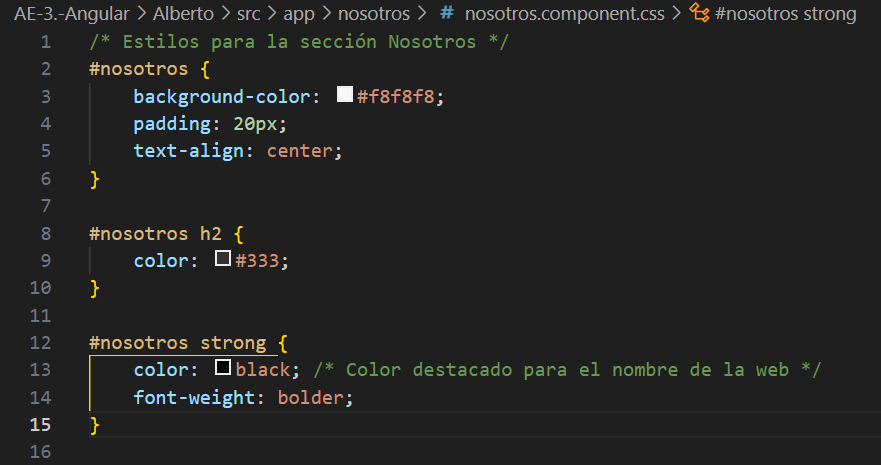




1. **Edito los html y css de los componentes nosotros, contacto, cabecera y pie-de-pagina** que se van a mostrar en la aplicación principal:

**Componente nosotros:**



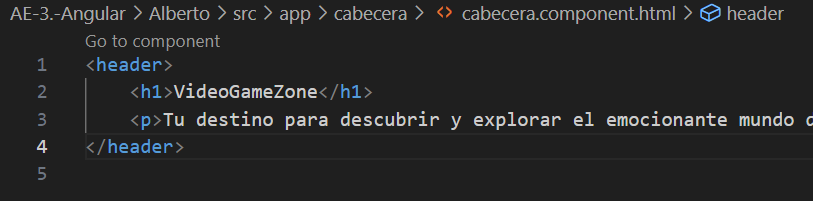


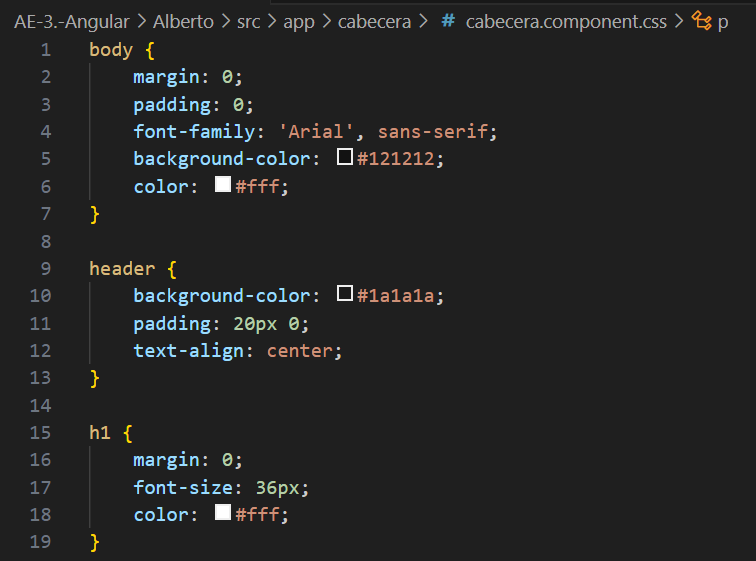
**Componente contacto:**



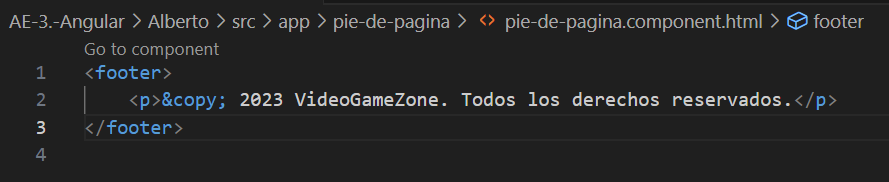


**Componente cabecera:**



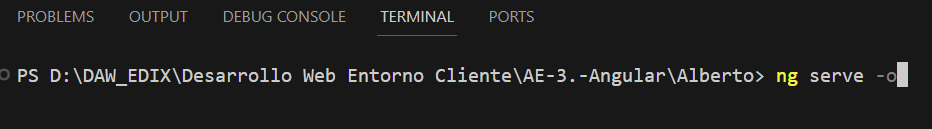


**Componente pie de página:**

****

****

1. **Ejecuto la aplicación** con el comando “ng serve –o”:

****

**Página de inicio:**

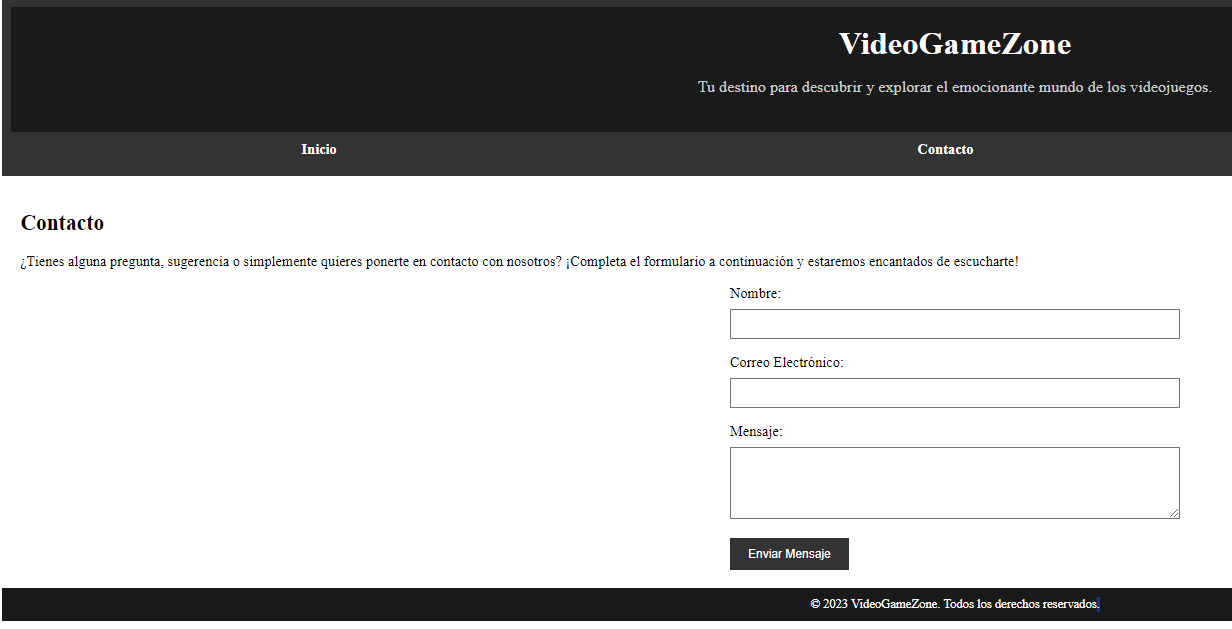
****

**Detalle de videojuego:**

**Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente**

**Contacto:**

****

**Nosotros:**

****

**Requerimiento 2**

Para entrar en la página deberemos de estar registrados, se mostrará una página de “login” al principio donde deberemos de poner nuestro nombre de usuario y nuestra contraseña. La aplicación tendrá algunos usuarios por defecto.

En caso de que no podamos entrar en la página, se nos mostrará otra vez la página de “login” diciendo que no hemos podido acceder.

La página de inicio de la aplicación muestra el login

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Si no se insertan campos, se muestran mensajes de error

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Si el usuario o la contraseña no coinciden con los declarados en el array, se muestra mensaje de error

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Usuario: “Marta” y Password: 123 se ncuentran dentro del array de usuarios logueados por lo cual, al escribir estos datos, accedemos al index. La contraseña se encuentra oculta porque utilizamos “type=password” en el html

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para que esto se logre, se creó el componente “login”. En “login.ts” se definen dos métodos: El que procesa el formulario y recupera los datos ingresados en este:

Texto

Descripción generada automáticamente

Y el método que muestra los mensajes de erro para los campos incompletos

Texto

Descripción generada automáticamente

CONCLUSIÓN

A lo largo de este trayecto, hemos enfrentado desafíos técnicos y organizativos que han puesto a prueba nuestra capacidad de colaboración y resolución de problemas.

En lo que respecta a Angular, a pesar de las claras ventajas que ofrece en el desarrollo de interfaces de usuario dinámicas y bien estructuradas, no podemos ignorar los posibles desafíos. Como todo aprendizaje en su fase inicial, hemos encontrado varios obstáculos. En contraste con JavaScript, Angular presenta notables ventajas en la organización del código, la capacidad de reutilizar componentes y su fuerte tipado.

Por otra parte, trabajar en equipo no solo se trató de la asignación eficiente de tareas, sino también de fomentar un ambiente de colaboración donde cada miembro pudo aportar sus habilidades y conocimientos de manera significativa. La capacidad de reutilizar componentes y mantener una clara separación de responsabilidades en Angular facilitó la integración de nuestro trabajo individual en una aplicación funcional.