Documento de instalación de Lince prescripción.

En primer lugar, vamos a tener que clonarnos en el sourcetree los siguientes repositorios:

* La parte PrescriptionSPA (FRONT):

<https://alejandria.visualstudio.com/DefaultCollection/%28PTC-LPC%29%20PrestVal%20-%20Lince%20Prescripciones/_git/ASISA.Lince.PrescriptionService>

* La parte PrescriptionService (BACK):

<https://alejandria.visualstudio.com.mcas.ms/(PTC-LPC)%20PrestVal%20-%20Lince%20Prescripciones/_git/ASISA.Lince.PrescriptionSPA>

# PrescriptionService (Back)

Una vez clonado el repositorio tendremos que abrir la solución en el visual studio 2022.

Lo primero de todo es ir a la carpeta .nuget de nuestro usuario, la cual se encuentra en C:, dentro de la carpeta tendremos que pegar los siguientes packages:

**Se pueden descargar en la WIKI de LincePrescripcion.**

Ya pegados los packages, iremos al visual studio y tendremos que cambiar el proyecto de inicio:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Seleccionaremos Prescription.API. Una vez seleccionado ya podremos levantar el servidor.

# PrescriptionSPA (FRONT)

Ya clonado el repositorio, abriremos el código en el visual studio code.

Para poder ejecutar el servidor, primero tendremos que tener instalado el NodeJS v14.17.1.

Una vez abierto el proyecto y con el node instalado, tendremos que ejecutar los siguientes comandos:

nvm install 14.17.1

nvm use 14.17.1

node -v

ng

npm install node-sass

yarn config set "strict-ssl" false -g

npm uninstall node-sass

npm install

npm install -g @angular/cli@6.2.9

npm install --save-dev @angular-devkit/build-angular

npm cache clean -f

npm install -g yarn

npm install yarn

yarn

yarn add node-sass

npm uninstall sass

npm install -g sass

yarn remove node-sass

yarn add sass

yarn install

npm install ng

Una vez ejecutados todos los comandos, podremos iniciar el servidor lanzando (para lanzar este comando deberemos de tener instalada en este punto la versión NodeJS v14.17.1):

ng serve -c local

Si todo esto no funciona seguimos el siguiente manual:

Instalamos nvm para Windows, el instalador está en la carpeta con el manual

Manual prueba angular YouTube

Instalamos nvm primero

Luego instalamos la versión de node 14.17.1

nvm install 14.17.1

nvm use 14.17.1

node -v

Luego instalamos el interprete de linea de commandos para angular la version 6 globalmente para el equipo con -g

npm install -g @angular/cli@6.2.9

Comprobamos que la instalacion se ha realziado correctamente con

ng -v

Ahora Podemos crear un nuevo Proyecto con el siguiente commando

ng new angular-prueba

Luego nos colocamos en la carpeta raíz del proyecto y lo podemos arrancar

cd .\angular-prueba\

ng serve

Para crear un componente ejecutamos:

ng generate component components/micomponente

Y nos crea todos los elementos del componente

Texto

El contenido generado por IA puede ser incorrecto.

Transcripción del tutorial de angular:

Lo primero, ¿qué es Angular? Pues Angular es un framework para JavaScript. ¿Qué es un framework? Pues es un marco de trabajo, una serie de convenciones, de buenas prácticas y de funcionalidades ya desarrolladas para nosotros utilizarlas. ¿De acuerdo? Angular, en concreto, es un framework para JavaScript y es un framework SPA, un framework para hacer aplicaciones single page application, es decir, aplicaciones que no recargan la pantalla en ningún momento, que son completamente reactivas y asíncronas con el servidor, es decir, el tipo de página en la cual tú vas navegando entre diferentes secciones y zonas de la web y la página nunca recarga, todo parece que carga instantáneamente.

Pues primero, antes de nada, debemos tener Nodejs instalado en nuestro equipo, que básicamente tendrías que entrar a la URL que ve en pantalla, nodejs.org, darle clic al botón de descargar Node en la versión que hay actualmente y después comprobar que lo tiene instalado. ¿Cómo lo puedes comprobar esto? Pues muy fácil, abres tu cmd o símbolo del sistema o si estás en Mac o Linux, la terminal y aquí básicamente lo que vas a hacer es escribir el siguiente comando,

node V.

Para ver la versión de Node tenemos esta y luego vas a hacer

npm - V

para ver la versión del gestor de paquetes de Node, muy necesario para ahora poder instalar Angular. Vale, ahora vamos a pasar a la instalación de Angular, que de hecho en la documentación oficial pues aparece toda la instalación como se puede hacer y tal. Eh, aquí en la sección de instalación por si quieres echarle un vistacillo y tal, pero yo te voy a hacer una instalación pues muy sencilla para que lo entiendas a la perfección. ¿Vale? Lo que vamos a hacer antes de nada es dirigirnos a la carpeta donde nosotros solamos trabajar y donde nosotros solamos hacer nuestro código. En mi caso, la carpeta que yo uso es la siguiente. Voy a hacer un CD para cambiar el directorio en la terminal y voy a hacer C2/gam64/3W. Aquí dentro es donde yo suelo trabajar. Así que voy a entrar y aquí voy a instalar eh pues Angular, que realmente instalar Angular en nuestro equipo pues lo hacemos con npm de manera global y luego crearemos el proyecto pues precisamente dentro de esta carpeta que es donde yo suelo trabajar, ¿no? Voy a instalar Angular. Para esto tengo que hacer el siguiente comando. npm, install, para instalar un paquete o un módulo de node. Y lo que voy a hacer es instalar - gmente. Esto es muy importante, este pequeño flag, esta pequeña opción para que instale Angular de manera global, porque si no lo instalaría solamente en el directorio en el cual estamos trabajando y no tendría ningún tipo de sentido. Entonces, lo que voy a hacer es hacer un @angular/cli. Lo que voy a instalar realmente es la interfaz de línea de comandos de Angular para yo poder luego trabajar con él. Entonces, lo que voy a hacer es instalar eh Angular CLI. Le voy a dar a enter para instalarlo de manera global, ¿vale? Y una vez que termine el comando de procesar y tal, pues ya lo tengo instalado. Ahora, ¿cómo puedo ver que la instalación se ha realizado correctamente? Pues ejecutando el siguiente comando,

ng version.

Si le doy a enter, lo que voy a ver ahora es la versión de Angular Cli. En mi caso, pues es esta de aquí. ¿Vale? Como puedes ver, aunque realmente pues funcionan todas similar. Voy a hacer un CLS para limpiar la consola.

Ahora ya estoy en disposición de crear un nuevo proyecto con Angular. ¿Cómo podría hacerlo? Pues utilizando precisamente el comando NG. Voy a usar el comando NG y le voy a poner la opción new para poder crear un nuevo comando, o sea, para poder crear un nuevo proyecto. También con el comando NG puedo generar toda clase de código y de contenido para nuestro framework, es decir, crear un componente, crear una pipe, crear diferentes tipos de elementos que hay dentro de Angular, ¿no? Entonces, con ng New y ahora el nombre del proyecto, por ejemplo, le voy a poner Angular-15- minutos. Le voy a poner ese nombre a ver si me deja crear un proyecto así. Y luego podría ponerle incluso más opciones como por ejemplo podría poner la opción standalón para que utilice los componentes independientes, que eso por defecto ya lo pilla. Entonces le voy a hacer ng New Angular 15 minutos, le voy a dar a enter y esto me va a crear o me va a sacar el asistente, ¿vale? Aquí me va a preguntar 70,000 cosas.

Por ejemplo, ¿qué hojas de estilos voy a usar? Yo, en mi caso, usaré CSS normal. Le doy a enter. Ahora me me pregunta si el proyecto que voy a hacer va a tener el SSR generalmente o no. Por defecto, al principio esto no hace falta, así que le voy a decir que no. Le voy a dar a enter. Y ahora me crea mi primer proyecto de Angular. ¿Qué está haciendo ahora mismo? Pues primero me ha creado los ficheros básicos, los ficheros base, el scafolding, es decir, la estructura básica del proyecto con las carpetas, los ficheros básicos y tal. Y después se pone a instalar los paquetes de npm, es decir, todas las dependencias externas que necesitamos para instalar Angular realmente en este proyecto y que realmente luego el proyecto pueda funcionar, ¿vale?

Entonces, se pone instalar todos los paquetes una vez que me ha creado la estructura básica de mi proyecto y simplemente esperando unos minutitos lo tendremos listo enseguida. Vale, pues ya tengo mi proyecto de Angular listo. Ahora lo que voy a hacer es un CD a esa carpeta de Angular 15 minutos. Le voy a dar a enter y entonces ya tengo aquí lista mi carpetita. Si hago dir, puedo ver todo el contenido que tengo, todas las carpetas que tengo y tal y cual. Sobre todo nosotros estaremos trabajando en src y ahora lo veremos cuando pasemos el proyecto a Visual Studio Code. Pero, ¿cómo puedo probar realmente que este proyecto funciona? Pues bueno, hay un comando para servir o para lanzar el proyecto de Angular, que es el siguiente ng serve. Entonces, yo puedo arrancar así mi proyecto de Angular. Una vez que lo arranque, lo que voy a tener aquí es un servidor local, un servidor de desarrollo que cada vez que yo hago un cambio en el proyecto se actualizará la pantalla y tal y cual y nos servirá la web en la siguiente URL, en esta de aquí, en local host. La puedo copiar, ¿vale? La puedo copiar y me puedo ir a mi navegador y la puedo abrir por aquí. Y esta sería la web que nos genera en principio angular nada más instalar el proyecto. Es básicamente una web que tiene un componente y tiene cuatro cosas más. ¿Vale? Vale, pues una vez que tenemos esto, ya tenemos Angular listo para empezar a trabajar. Lo voy a cargar en Visual Studio Code. Para eso le voy a dar clic aquí a Open Folder o le voy a dar a file Open Folder, ¿vale? Y voy a buscar la carpeta de este proyecto de Angular 15 minutos. Ya la he buscado y he cargado la carpeta y aquí tengo toda la estructura de directorio de Angular. Principalmente nosotros, como ahora mismo no tenemos muchísimo tiempo, te voy a decir que vamos a estar principalmente trabajando en la carpeta SRC, donde aquí tengo varias cositas. Tengo el fichero main, donde directamente cargamos el componente inicial y hacemos ciertas configuraciones para que la aplicación funcione. Tenemos la hoja de estilos de CSS por aquí, ¿vale? General, tenemos el inde HTML, que es donde se carga toda la aplicación y el componente principal y luego también tenemos app, que es donde nosotros realmente vamos a estar trabajando. Finalmente tenemos el componente app, que si te fijas en el HTML es toda esta web que al principio tenemos. Fíjate si dejo de lado los estilos. Es toda esta web que estamos viendo en el navegador, ¿vale? Nos dice hola y el nombre del proyecto y luego nos saca este texto y tal. Pues esto es la plantilla del componente inicial que tenemos, el componente app. Luego te explico qué es un componente y cómo funciona, ¿vale? Y luego tengo el el TS, ¿no? El componente como tal, el fichero de TypeScript del componente, que luego te explico también por qué está formado. Aparte también tengo la parte de rutas por si quiero configurar rutas en la aplicación. Tengo también un fichero de configuración por aquí y básicamente es lo que tengo. Entonces, ¿cómo funciona Angular? Pues Angular funciona por componentes, es decir, un componente va a tener tres partes principalmente o cuatro, ¿no? Pero las más importantes son la clase o el fichero de TypeScript donde nosotros hacemos el componente, que este fichero está formado por una serie de importaciones de diferentes componentes o módulos. También está formado por el decorador @component donde le indico diferentes eh características que va a tener el componente, como por ejemplo su selector o etiqueta, las diferentes importaciones de diferentes dependencias que vamos a tener para luego poder tenerlas disponibles en el componente. La template URL, que es la la URL donde está la plantilla y la parte visual que se va a mostrar. Y también estamos cargando, por ejemplo, aquí los estilos de ese componente porque cada componente puede tener unos estilos individualizados, ¿vale? Nosotros en principio tenemos el app component HTML, que es lo que vemos aquí, y es un componente que vamos a simplificar muchísimo. Así que lo que voy a hacer es eliminar completamente todo todo el contenido de mi eh componente de app, toda la plantilla. Simplemente voy a dejar el router outlet por aquí, voy a guardar y voy a tener mi componente luego voy a guardar y aquí no voy a tener nada. Es decir, ahora en la página no va a haber nada, salvo que yo, por ejemplo, quiera mostrar un título, por ejemplo, por pantalla, este título del del nombre del proyecto, ¿no? Por ejemplo, en lugar de ponerle títel, le voy a poner nombre, proyecto. Esa sería mi variable o mi propiedad de mi clase. Y luego lo puedo poner por aquí, ¿vale? Ese nombre del proyecto lo podría mostrar, por ejemplo, en un H1, en la plantilla, que la plantilla pues tiene incluso su propio lenguaje de plantillas en el cual tiene sus propias estructuras de control y podemos trabajar con las plantillas de manera individualizada con un lenguaje especial, ¿no? Entonces, ¿cómo puedo mostrar esa variable que tengo en mi componente? Muy fácil. Todas las variables o propiedades que yo tenga en mi clase de mi componente, todas estas pasan automáticamente a la template vinculada, que es esta en este caso. Entonces, aquí yo podría mostrar con las dobles llaves, es decir, llave, llave, llave, llave, podría interpolar aquí dentro el nombre de esa variable, que en este caso sería nombre, proyecto, que automáticamente me lo reconoce Visual Studio Code. Por cierto, si no sabes por qué eh a ti no te reconoce Visual Studio Code esto, es porque no tienes una extensión que se llama Angular Language. La puedes buscar Angular Language Service y te va a ayudar un poco a lo que es pues trabajar con las templates, trabajar con los componentes y trabajar con Angular en general. Listo. Con esto ya tendría esto. Pero es que si yo tengo otra variable, por ejemplo, que se llame descripción, descripción y le meto aquí otro texto, por ejemplo, esto es una clase rápida de repaso de Angular moderno. Y luego le meto incluso otra variable, otra propiedad para esta clase que se llame profesor, por ejemplo, y meto mi nombre, Víctor Robles Web. Vale, todas estas variables las puedo pasar a la vista perfectamente al fichero. HTML vinculado en este componente gracias a su decorador. Es muy sencillo. Básicamente lo que puedo hacer aquí es, por ejemplo, mostrar en un H2 con las dobles llaves, incluso la descripción y luego puedo mostrar también en un H3 con las dobles llaves de la interpolación la otra variable que se llame profesor, ¿no? Entonces, de esta manera puedo ver que aquí por pantalla me saca todos los datos que yo tengo guardados en mi componente. Imagínate si es un dato dinámico, un dato que saco desde una petición asíncrona a un servidor, lo que sea, todo lo puedo pasar a la vista muy fácilmente. Vale, pues ya hemos visto más o menos cómo puede funcionar un componente, que pues ya te digo, se utiliza el decorador @cponent y se utilizan estas propiedades para configurarlo. Ahora vamos a ver cómo podemos crearlo. Para eso tenemos que abrir de nuevo la consola y cortar la ejecución del servidor haciendo control C por aquí para cortarla. Y aquí lo que vamos a hacer es utilizar el comando NG. Vamos a hacer ng generate component y le vamos a poner el nombre al componente. En este caso podríamos incluso tener una carpeta de components, por ejemplo, para tener todos los componentes. Es decir, haríamos aquí components barra y el nombre del componente. El nombre del componente podía ser videojuegos. Vale. Y le voy a dar a enter. Esto me va a crear mi componente dentro de una carpeta, ¿okay? Me va a generar un scafolding básico de lo básico que va a tener el componente, es decir, el HTML, que es la plantilla, el fichero de prueba, si es que las queremos hacer, el fichero del componente y los estilos, ¿no? Automáticamente yo lo que puedo ver por aquí en mi Visual Studio Code es que tengo la carpeta con una carpeta de videojuegos por aquí. Este componente ahora mismo si lo abro puedo ver que tengo el @component definido, el nombre de la etiqueta. No tengo ningún import porque no tengo ninguna dependencia importada. Tengo también la template y los estilos vinculados. Todo bien. En la template incluso tengo videojuego Works. Pero, ¿cómo puedo mostrar este componente? En mi componente principal, es decir, en mi componente de app. ¿Cómo puedo hacerlo? Pues muy sencillo. Mira, si yo quiero utilizar un componente que acabo de crear o varios componentes en un mismo componente, lo que puedo hacer es un importo utilizar. En este caso sea videojuegos component. Si lo hago así, automáticamente me importa mi componente por aquí, ¿vale? Y con este componente importado, con su objeto, lo que voy a hacer es meterlo en imports. Lo meto por aquí. Videojuegos component. A ver, voy a ponerlo bien. Vide o juegos component. Y yo ya podría utilizar este componente aquí dentro, dentro de mi plantilla, por ejemplo. Es decir, podría automáticamente utilizar por aquí app videojuegos, su etiqueta y ya aparecería aquí en la pantalla. Si arranco de nuevo el servidor, si hago ng serve, ya aparecería en la pantalla, como voy a poder ver en un segundo, ¿de acuerdo? Ya aparecería en la pantalla, como puedes ver, videojuegos Works, es decir, ese componente ya lo tengo cargado aquí. Por tanto, he creado un componente completamente nuevo y lo he cargado dentro de mi componente principal. Ahora, para enseñarte un poco más de Angular, voy a cerrar este componente principal y nos vamos a centrar en el componente de videojuegos. En concreto, nos vamos a centrar en el fichero de TapeScript y también en la plantilla, ¿de acuerdo? En las dos cositas, ¿vale? Aquí tienes que saber que en el arocomponent puedes personalizar el selector e incluso agregar más propiedades que no están aquí y puedes importar cualquier cosa, cualquier otro componente para que se cargue dentro de este y tal y todo eso. Incluso puede utilizar una template integrada dentro de este propio componente, es decir, que no esté en un fichero separado, por ejemplo, en videojuegos con HTML, sino que esté dentro de este, ¿vale? Pero como tenemos la plantilla individual, es mucho mejor tenerla separada por limpieza de código y tal, ¿no? Vale, pues voy a borrar lo que tengo en la plantilla y vamos a hacer un pequeño ejemplo. Me voy a crear un div por aquí que tenga una clase, por ejemplo, que se llame videojuegos, ¿vale? Y lo que voy a hacer aquí dentro es crear una pequeña lista de videojuegos que yo podré ir agregando. Voy a meter un H2 aquí que ponga lista o listado de videojuegos. Y lo que voy a tener aquí va a ser una UL con mis videojuegos. Es decir, tendré diferentes licon, el videojuego uno, el videojuego dos y así. Lo que pasa es que luego, obviamente, recorreré todos esos videojuegos y tal y los mostraré. Y luego, debajo de la lista voy a tener un input de tipo texto que tendrá un placeholder también que diga eh nuevo agrega un nuevo videojuego, agrega un nuevo juego, ¿vale? Y finalmente pues tendré el input cerrado por aquí. Y luego también puedo tener un botón, ¿vale?, que ponga añadir. De hecho, también puedo meter un HR por aquí para que me meta una línea divisoria y veamos qué es el componente principal y qué otro componente está integrado dentro de ese componente principal. ¿Vale? Ahora, ¿cómo podría yo agregar diferentes videojuegos a este listado y mostrarlos? Muy fácil. Mira, yo puedo tener aquí en mi componente de videojuegos una propiedad que se llame videojuegos, por ejemplo. Y esta propiedad va a ser una array de strings, es decir, le voy a indicar el tipo de dato que va a ser string y le va a decir que es un array de strings. Este array de strings en un principio va a estar vacío, no va a tener nada porque yo me voy a encargar de rellenarlo. Ahora, ¿qué voy a hacer para poder agregar nuevos videojuegos? Pues voy a tener también una propiedad que se llame nuevo juego y aquí le voy a indicar que va a ser un string que de momento va a estar vacío. Lo que pasa es que esta propiedad la voy a rellenar mediante mi formulario, pero básicamente lo que puedes hacer es vincular esa propiedad al input que tienes aquí y mostrarla por pantalla. Por ejemplo, si a mí hm si yo quiero estar mostrando, por ejemplo, en un H4 ese ese esa propiedad de nuevo juego, nuevo juego, voy a ver que yo ahora mismo en pantalla no voy a ver nada. Pero si le vinculo esa propiedad a mi input, es decir, vinculo este input de manera bidireccional con la propiedad nuevo videojuego, es decir, utilizo el tway data binding típico de Angular utilizando la directiva corchetes paréntesis ng model vinculo de manera bidireccional la propiedad, hago un binding bidireccional de la propiedad nuevo juego. Lo que voy a hacer ahora mismo es que tengo vinculado perfectamente la propiedad nuevo juego. ¿Qué sucede? Pues esto estaría vinculado, pero te va a dar un error. ¿Por qué te va a dar un error? Porque estamos usando formulario y para trabajar con los formularios y el ng model tengo que importar por aquí el módulo de formularios. Por aquí arriba voy a importar con el import y la Javers el forms módule. De esta manera, si lo importo este paquete, este módulo from@ from@angular/forms, yo ya tendría disponible el módulo de formulario en principio, pero si yo no lo importo en mi componente, es decir, no lo paso por aquí, el forms module, yo no lo voy a tener disponible. Ahora, si lo importo, ya tengo disponible el ng model. ¿Qué tiene esto de mágico? pues que automáticamente si yo voy agregando contenido aquí ves cómo se va inmediatamente actualizando el valor de la variable y también se refleja inmediatamente el valor de esa variable en mi página web utilizando pues la interpolación que hemos usado. Es decir, se actualiza todo de manera inmediata y eso es una de las grandes ventajas que tiene Angular. Ahora, con esta variable, teniendo claro que estoy agregando un nuevo juego, ¿qué puedo hacer para poder guardarlo en mi listado? Porque aquí efectivamente ya tengo guardado el valor que yo haya puesto en mi formulario, es decir, he puesto hola, ¿qué tal? Pues el valor que hay dentro de esta variable es hola, ¿qué tal? Vale, pues lo que puedo hacer es crearme un nuevo método por aquí que se llame, por ejemplo, add juego. Y este método lo puedo usar luego desde mi plantilla usando un evento de JavaScript o un evento click de Angular. En este caso, lo que voy a hacer es tener una nueva función por aquí y lo que voy a hacer es comprobar si eh nuevo juego me da true, es decir, si dis punn nuevo juego, el dis lo utilizo porque estuvo una clase para acceder a las propiedades, tengo que hacerlo. Si hago dis nuevo juego, si esto da true y además también si le hago un trim punto eh nuevo juego. Si esto me da true, es decir, que tiene contenido, pues lo que voy a hacer aquí es añadir es hacer un disvjujuegos. Push para añadir dis.noujegot.tam para agregarlo dentro del listado. Listo. Entonces ahora ya podría vaciar incluso la eh variable de nuevo juego. Dis. Nuevo juego. Lo vacío y lo dejo completamente limpio. ¿Vale? Ahora también tendría obviamente que mostrar todo ese listado en mi pantalla por aquí en el listado principal. Para eso puedo utilizar el @for que es una de las nuevas estructuras de control que hay dentro de las plantillas de Angular. También tenemos el asterisco for para utilizar esa sintaxis de atributo de HTML para hacer un bucle y estructura de control. Pero bueno, podríamos usar el afors, ¿de acuerdo? Es decir, en este caso sería eh juego off videojuegos y luego abro unas llaves para imprimir por aquí el listado. Si estuviera vacío también podría utilizar el @empty y de esta manera podría indicar un mensaje, por ejemplo, un le dijera que no hay videojuegos para mostrar. Vale, eso podríamos hacerlo, pero si sí que hubiera videojuegos, yo lo que puedo hacer es un le para interpolar cada juego que tenga yo en mi listado de videojuegos, que por cierto pues lo tengo aquí dentro, ¿no? Ahora, esto me va a generar un error. Esto va a pasar porque no tengo el valor track puesto y el valor track es una expresión que va a determinar una clave que se utiliza para asociar los elementos de una ray con los elementos que estoy listando. Entonces, de esa manera puedo saber qué elemento de los que estoy mostrando está vinculado a qué eh elemento de una Ray. Entonces, de esa manera puedo hacer un binding, un ch data binding, etcétera, ¿no? En este caso sería muy sencillo, simplemente podríamos hacer track y podríamos pasarle juego para que cada juego pues esté vinculado a sí mismo. De todas formas, normalmente si estos juegos que yo estoy mostrando tuvieran un identificador, lo suyo sería hacer juego punto idid para que cada uno de los elementos le que estoy mostrando y que estoy recorriendo pues estén vinculados a ese identificador exclusivo, ¿no? De todas formas así va a funcionar. Entonces, si actualizo la pantalla me dice que no hay videojuegos para mostrar, ¿vale? Pero ahora, ¿cómo puedo agregar los videojuegos? Porque si yo agrego aquí contenido y le doy a añadir no funciona. Pues lo que tengo que hacer es llamar al método add juego cuando yo le dé clic al botón de añadir. Es decir, utilizo los paréntesis y con los paréntesis puedo usar muchísimos eventos. Por ejemplo, puedo utilizar el evento click, ¿vale? Y entonces le voy a indicar aquí el método que voy a llamar el método AD de juego. Y obviamente tengo que utilizar los paréntesis porque es un método. ¿Vale? Ahora si actualizo la pantalla por aquí y agrego un nuevo juego, pongo hola añadir. Me me lo agrega por aquí, pongo otra cosa, lo agrego, pongo otra cosa, lo agrego. Pongo otra cosa, lo agrego. Pongo otra cosa y lo añado. Entonces, de esta manera puedo ir aumentando el contenido que tiene la ra de videojuegos y lo puedo seguir agregando todo. Y con esto hemos dado nuestros primeros pasos en Angular. Obviamente Angular es muchísimo más grande y tiene muchísimo más por explorar y por aprender. Y si quieres profundizar en Angular y que te lo enseñe tranquilamente, paso a paso, con muchísimos ejercicios, prácticas, proyectos y con explicaciones al detalle y en profundidad, tienes un curso completo con decenas de horas de contenido aquí abajo en la descripción, donde te enseño a angular tranquilamente, paso a paso desde cero y con muchos ejercicios, prácticas y proyectos. Un curso espectacular que te va a encantar si es que quieres seguir profundizando y aprendiendo Angular. Y nada más. Si te ha gustado, déjame un like, suscríbete, activa la campanita de notificaciones y nos vemos en el