1. [UNIDAD 3: Introducción](https://learn.nextu.com/mod/page/view.php?id=3601&pid=P_WEBDEV_V2)

**Bienvenido a Herramientas, Build Systems y tecnologías emergentes**

En esta Unidad aprenderás qué es Angular 2, un framework JavaScript de código abierto promovido por Google y la comunidad de desarrollo libre. Enfocado en la filosofía mobile-first, es uno de los frameworks para el desarrollo de SPAs con mayor renombre en el mundo.

**Puntos de aprendizaje**

**Unidad 3:  Angular 2**

**Los objetivos del Programa que corresponden a esta Unidad son:**

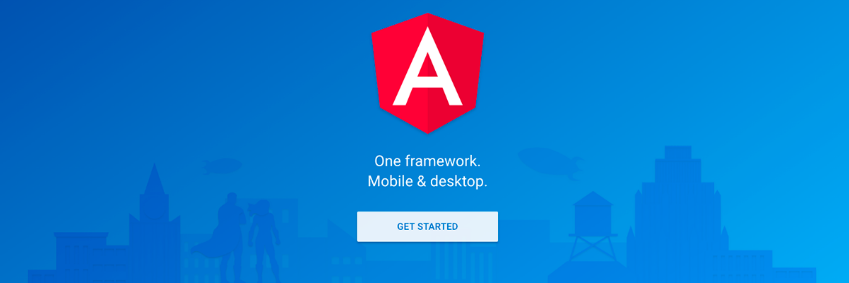
* Crear ambientes de programación y a gestionar recursos.
* Implementar la construcción de sitios web a partir de componentes.

**Los objetivos del Curso 8 que corresponden a esta Unidad son:**

* Identificar y apropiar tendencias web.
* Aplicar el concepto ambiente web.
* Implementar web components.

1. Lección 1: ¿Qué es Angular y cómo funciona?
   1. [Lectura: ¿Qué es Angular 2?](https://learn.nextu.com/mod/lesson/view.php?id=3602&pid=P_WEBDEV_V2)

¿Qué es Angular 2?

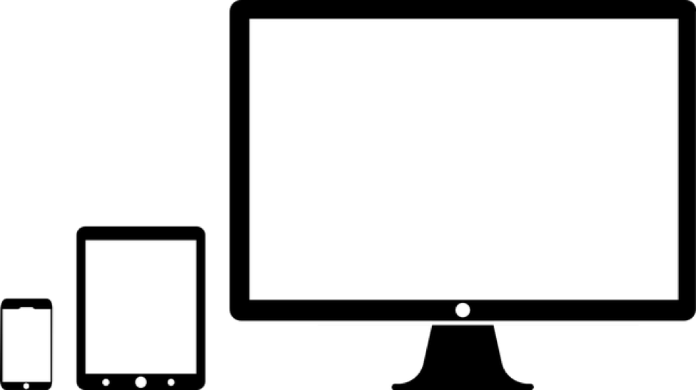


Angular 2 es un framework JavaScript de código abierto promovido por Google y la comunidad de desarrollo libre. Enfocado en la filosofía mobile-first, es uno de los frameworks para el desarrollo de SPAs con mayor renombre en el mundo.   
Fue lanzado en su versión estable en Mayo de 2016 y desde entonces, su comunidad de desarrolladores ha crecido exponencialmente.

Ventajas de Angular 2:

A continuación, se presentan algunas de las principales ventajas y características de Angular 2:

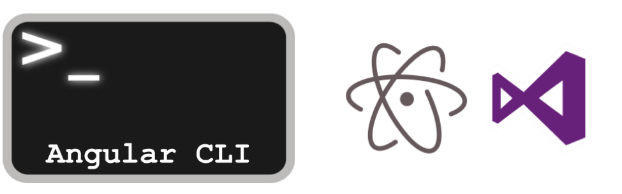
* **Multiplataforma:** con Angular 2 es posible desarrollar páginas Web tipo SPA de alto rendimiento. Sin embargo, es posible, por medio de la integración con otras herramientas y frameworks, crear aplicaciones móviles nativas o híbridas, así como aplicaciones de escritorio en múltiples sistemas operativos.



* **Rapidez y desempeño:** mediante la generación de plantillas y el uso de componentes para la reutilización de código, la rapidez en la que se desarrollan las aplicaciones es mucho más alta que el desarrollo Web tradicional. Además, gracias al enrutador de componentes (Component Router), los usuarios sólo cargan el código necesario para la vista que desean acceder en ese instante, mejorando notoriamente el rendimiento de la aplicación y contribuyendo a una experiencia de usuarios fluida y veloz.



* **Productividad:** Angular CLI como herramienta de ayuda desde la línea de comandos, permite crear proyectos, elementos de la aplicación y mucho más, con gran facilidad y rapidez. Además, existe una gran variedad de IDEs con funcionalidades para autocompletar código, detección de errores, entre muchas más.



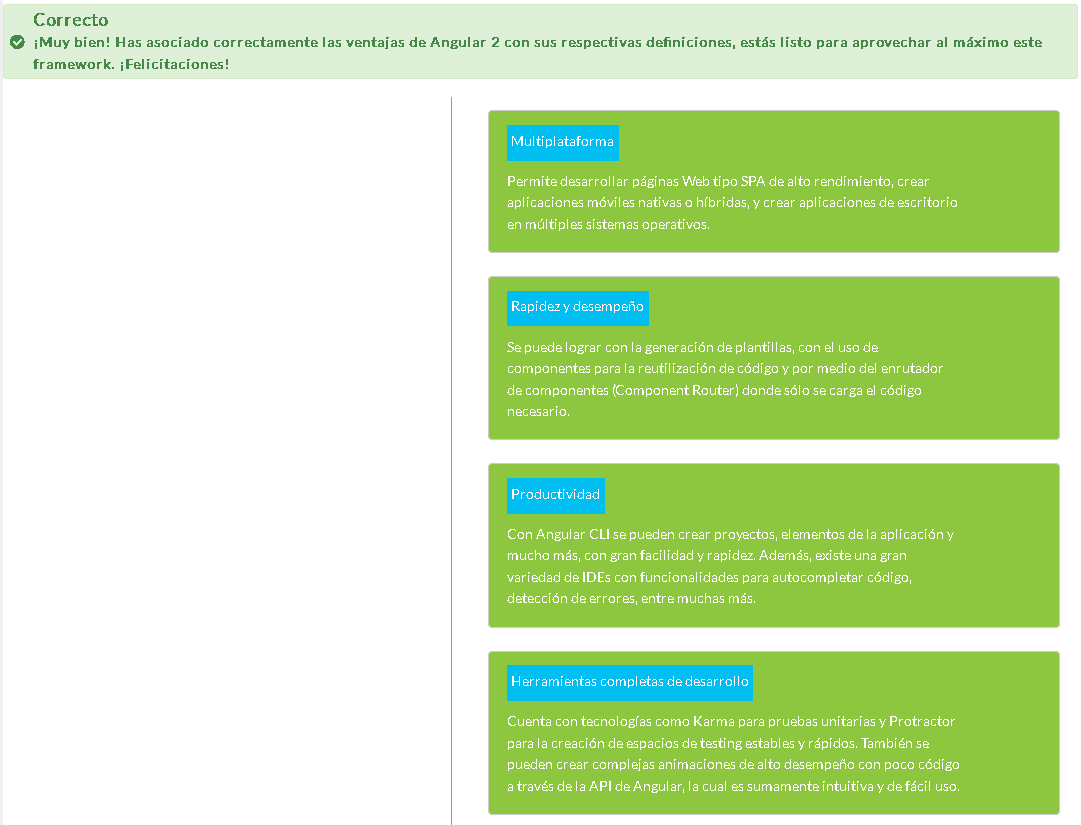
* **Herramientas completas de desarrollo:** Angular 2 incluye tecnologías como Karma para pruebas unitarias y Protractor para la creación de espacios de testing estables y rápidos. Además, es posible crear complejas animaciones de alto desempeño con poco código a través de la API de Angular, la cual es sumamente intuitiva y de fácil uso.



* 1. [Actividad 1](https://learn.nextu.com/mod/lesson/view.php?id=3602&pageid=14478&pid=P_WEBDEV_V2)

Tiempo de pensar

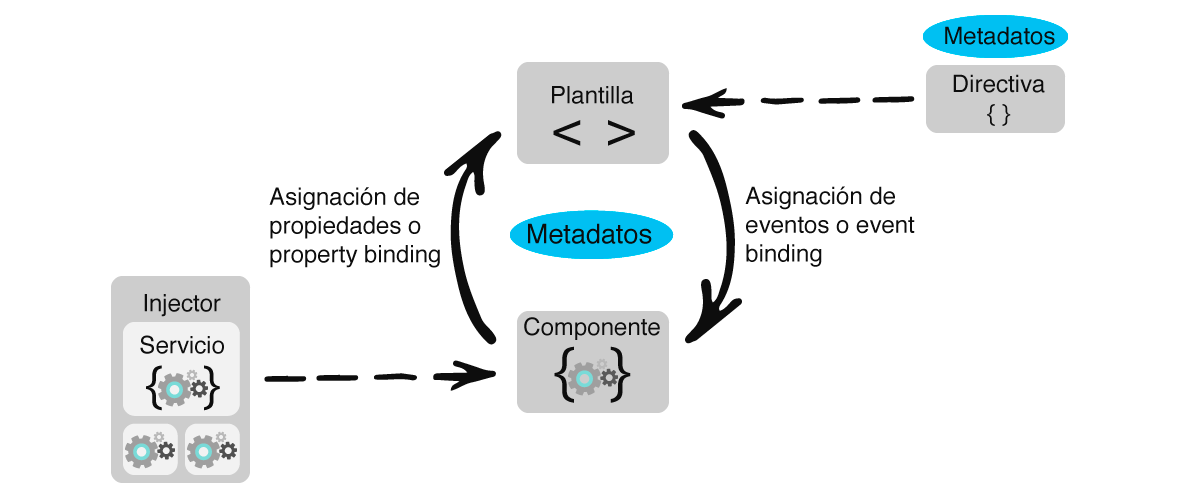
Ahora que sabes qué es Angular 2 y cuáles son sus principales ventajas y características, te invitamos a que realices la siguiente actividad que te ayudará a reforzar tu conocimiento.  
  
Instrucciones:  
Lee el enunciado con atención y realiza la asociación que consideres correcta.  
  
Enunciado:  
Angular 2 es un framework JavaScript de código abierto promovido por Google y la comunidad de desarrollo libre, éste cuenta con unas ventajas principales que tú ya conoces. A continuación, aparecerán listadas y al frente verás las definiciones, las cuales deberás asociar según corresponda. ¡Adelante!



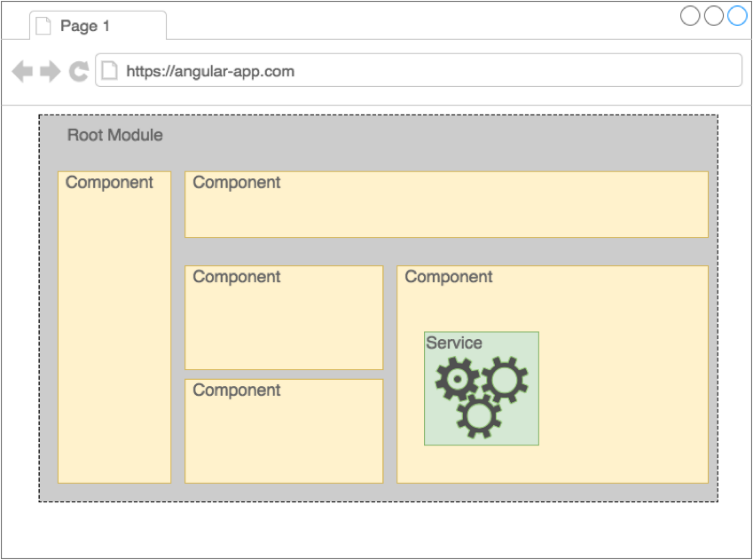
* 1. Lectura: Arquitectura

Arquitectura

Angular 2 es un framework que se encarga de crear aplicaciones del lado del cliente, usando diferentes elementos propios de su estructura para obtener un resultado óptimo en cuanto a eficiencia, desempeño y usabilidad.   
  
La arquitectura de este framework se resume en el siguiente gráfico:



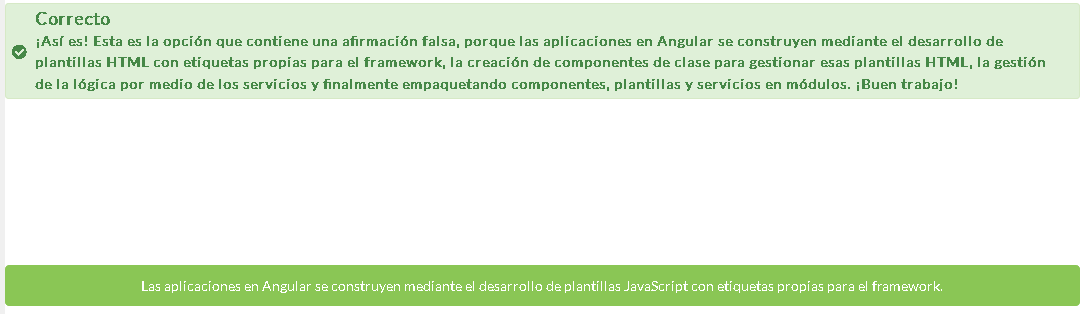
Cada uno de los elementos que hacen parte de la arquitectura de Angular 2 serán estudiados con mayor profundidad más adelante, sin embargo es importante anotar que este framework se considera como modular, es decir, que utiliza módulos reutilizables para la construcción de la aplicación.  
  
Los módulos están integrados por componentes que contienen la vista a través de las plantillas y las directivas, la lógica a través de los servicios y las asignaciones, tanto de eventos, como propiedades para las interacciones dinámicas entre las vistas y la lógica.  
  
Las aplicaciones en Angular se construyen mediante el desarrollo de plantillas HTML con etiquetas propias para el framework, la creación de componentes de clase para gestionar esas plantillas HTML, la gestión de la lógica por medio de los servicios y, finalmente, empaquetando componentes, plantillas y servicios en módulos.   
  
Al lanzar la aplicación, se realiza un empaquetado general de todos los elementos y se muestra el módulo raíz, o root module, de donde parte toda la secuencia de la aplicación.



* 1. Actividad 2

Tiempo de pensar

Ahora que conoces la arquitectura de Angular 2, te invitamos a poner a prueba tus conocimientos con la siguiente actividad.  
  
Instrucciones:  
Lee el enunciado con atención y elige la opción que consideres correcta.  
  
Enunciado:  
A continuación encontrarás varias afirmaciones referentes a la arquitectura de Angular 2 que has visto hasta ahora. Tu misión es encontrar la opción que contiene información falsa sobre el tema. ¡Adelante!



* 1. Configuración del ambiente

Angular CLI 🡪 Interfaz de línea de comandos asistente y sirve para:

* Crear un nuevo proyecto
* Crear elementos del framework
* Crear bloques de código
* Correr una aplicación

**Requerimientos:**

* Node versión 4 o superior
* NPM versión 3 o superior

C:\Users\jrodrigue253>node -v

v12.19.0

C:\Users\jrodrigue253>npm -v

6.14.8

C:\Users\jrodrigue253>

**Se debe proceder con la instalación de angular-cli**

C:\Users\jrodrigue253>npm install -g angular-cli

**Se crea un nuevo proyecto**

C:\...\ejercicio1-Ang2>***ng new primer-proyecto***

? Would you like to add Angular routing? No

? Which stylesheet format would you like to use? CSS

CREATE primer-proyecto/angular.json (3638 bytes)

CREATE primer-proyecto/package.json (1291 bytes)

CREATE primer-proyecto/README.md (1031 bytes)

CREATE primer-proyecto/tsconfig.json (489 bytes)

CREATE primer-proyecto/tslint.json (3125 bytes)

CREATE primer-proyecto/.editorconfig (274 bytes)

CREATE primer-proyecto/.gitignore (631 bytes)

CREATE primer-proyecto/browserslist (429 bytes)

CREATE primer-proyecto/karma.conf.js (1027 bytes)

CREATE primer-proyecto/tsconfig.app.json (210 bytes)

CREATE primer-proyecto/tsconfig.spec.json (270 bytes)

CREATE primer-proyecto/src/favicon.ico (948 bytes)

CREATE primer-proyecto/src/index.html (300 bytes)

CREATE primer-proyecto/src/main.ts (372 bytes)

CREATE primer-proyecto/src/polyfills.ts (2835 bytes)

CREATE primer-proyecto/src/styles.css (80 bytes)

CREATE primer-proyecto/src/test.ts (753 bytes)

CREATE primer-proyecto/src/assets/.gitkeep (0 bytes)

CREATE primer-proyecto/src/environments/environment.prod.ts (51 bytes)

CREATE primer-proyecto/src/environments/environment.ts (662 bytes)

CREATE primer-proyecto/src/app/app.module.ts (314 bytes)

CREATE primer-proyecto/src/app/app.component.html (25725 bytes)

CREATE primer-proyecto/src/app/app.component.spec.ts (969 bytes)

CREATE primer-proyecto/src/app/app.component.ts (219 bytes)

CREATE primer-proyecto/src/app/app.component.css (0 bytes)

CREATE primer-proyecto/e2e/protractor.conf.js (808 bytes)

CREATE primer-proyecto/e2e/tsconfig.json (214 bytes)

CREATE primer-proyecto/e2e/src/app.e2e-spec.ts (648 bytes)

CREATE primer-proyecto/e2e/src/app.po.ts (301 bytes)

√ Packages installed successfully.

Directory is already under version control. Skipping initialization of git.

**Se corre el proyecto con el comando:**

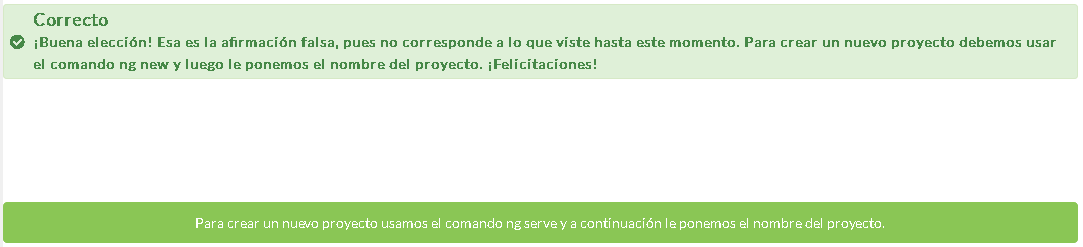
ng serve 🡪 Con este comando se inicia el nuevo proyecto creado, además la app web queda corriendo en el puerto por defecto 4200

* 1. Actividad 3

Tiempo de pensar

En esta oportunidad pudimos configurar el ambiente de Angular 2, un framework para la creación de aplicaciones Web para comenzar a realizar nuestras primeras aplicaciones. Ahora queremos invitarte a validar tus conocimientos a través de la siguiente actividad.  
  
Instrucciones:  
Lee el enunciado con atención y elige la opción que consideres correcta.

Enunciado:  
A continuación encontrarás varias afirmaciones referentes a la configuración del ambiente de Angular 2. Tu misión es encontrar la opción que contiene información falsa sobre el tema. ¡Adelante!



* 1. Lectura: Introducción a TypeScript

Introducción a TypeScript



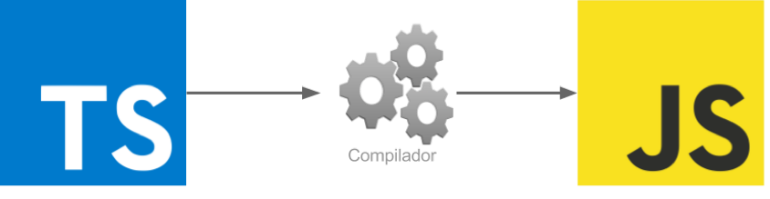
A medida que empieces a conocer un poco más sobre Angular 2, e iniciemos la codificación de algunos elementos, te darás cuenta que existen algunos elementos en el lenguaje diferentes a lo que habíamos trabajado en JavaScript. La explicación para esto es que el lenguaje que usamos para codificar aplicaciones en Angular 2 técnicamente no es JavaScript sino TypeScript. ¿Sabes qué es?

**¿Qué es TypeScript?**

TypeScript es un lenguaje de programación creado por Microsoft y de código abierto, que comprende un set de características adicionales a JavaScript para la creación de scripts Web. En resumen, podríamos decir que es un complemento de JavaScript que añade unas cuantas herramientas adicionales de gran uso para los desarrolladores.

**Compilación**

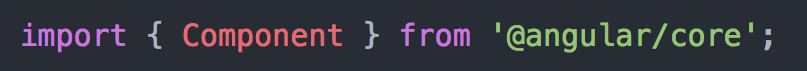
Tal vez te haya surgido la siguiente pregunta: si el único lenguaje que es interpretado por todos los navegadores es JavaScript, ¿cómo hace TypeScript para funcionar? La respuesta está en la misma pregunta, es decir, los archivos TypeScript se compilan a JavaScript plano para después ser interpretados por los navegadores. El compilador de TypeScript convierte el código propio de este lenguaje en un equivalente de JavaScript ECMAScript 3 o superior, esto quiere decir que todo código JavaScript funciona correctamente en un archivo TypeScript.



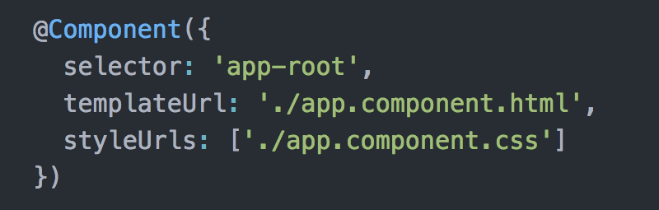
**Diferencias entre TypeScript y JavaScript plano**

Como ya lo mencionamos, TypeScript corresponde a un set adicional de herramientas para JavaScript. Dentro de las principales novedades que se encuentran en este lenguaje están:

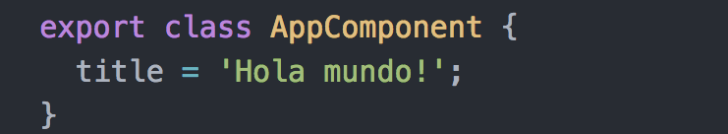
* Imports: la manera de importar código desde otro módulo es muy sencilla, haciendo que la modularidad en los proyectos sea fluida y eficiente.



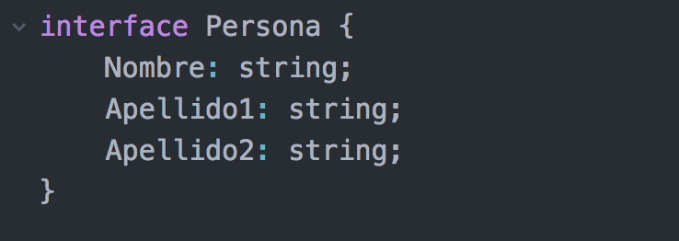
* Decoradores: los decoradores son funciones adjuntas a un objeto que realizan acciones en segundo plano y modifican el objeto al que están unidos para un fin específico.



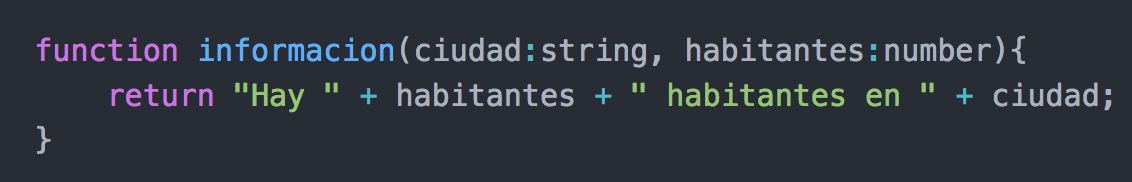
* Clases: la forma en que TypeScript define las clases permite que la programación orientada a objetos sea muy sencilla y correcta.



* Interfaces: elemento que no existe en JavaScript, sirve para forzar la implementación de atributos y métodos en una clase determinada.



* Tipos: como bien sabes, en JavaScript no se definen los tipos de datos para las variables creadas, y por lo tanto, éstas pueden tener cualquier tipo de valor. TypeScript añade un nivel adicional de seguridad en el código mediante la asignación de tipos a las variables o atributos de un objeto. De allí viene su nombre, ya que Type en inglés es Tipo.



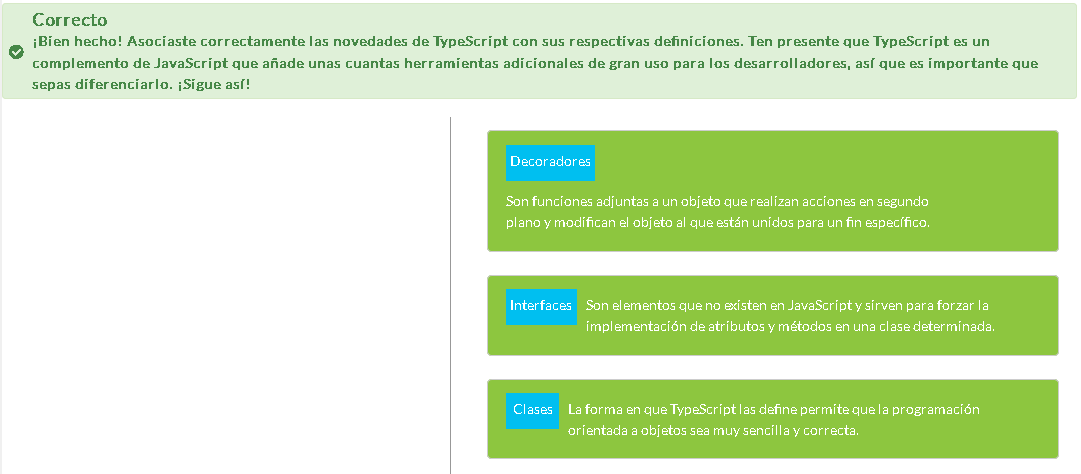
¿Por qué usar TypeScript?

Si el código TyeScript es compilado en JavaScript plano, ¿por qué no hacerlo todo en JavaScript desde el principio? En realidad hay muchas razones por las cuales se usa TypeScript en el desarrollo de aplicaciones con Angular 2:

* 1. Lenguaje fuertemente tipado: TypeScript añade tipos de datos a variables y a atributos de objetos, esto aumenta el nivel de seguridad en el código evitando que valores inapropiados sean asignados y permitiendo fácilmente la detección de errores en la compilación, o que incluso el mismo IDE pueda identificar errores y alertas que te permitan, como desarrollador, detectar estas situaciones con gran facilidad.
  2. Características de nueva generación: las herramientas de TypeScript mejoran en gran medida algunos elementos de JavaScript, lo que aumenta la calidad del código escrito y las posibilidades que se pueden alcanzar con él.
  3. Nuevas características: TypeScript, además de mejorar lo existente en JavaScript, también añade nuevos elementos como las interfaces y los genéricos, ampliando la gama de herramientas para que el desarrollador cree scripts de alto rendimiento y funcionalidad.
  4. Lenguaje principal de Angular 2: los creadores de Angular 2 eligieron a TypeScript como el lenguaje principal del framework, lo que significa que crear aplicaciones en Angular 2 usando TypeScript produce una fluidez y compatibilidad óptimas en la codificación y la compilación.
  5. Documentación y soporte: al ser TypeScript el lenguaje principal para Angular 2, la gran mayoría de documentación y soporte que encontrarás por parte de la comunidad de desarrolladores, estará en dicho lenguaje; permitiendo encontrar ayuda de los expertos con gran facilidad.
  6. Actividad 4

Tiempo de pensar

Ahora que sabes qué es Angular 2, cuáles son las diferencias que tiene frente a JavaScript y por qué usarlo, es hora de afianzar tus conocimientos por medio de la siguiente actividad.  
  
Instrucciones:  
Lee el enunciado con atención y realiza la asociación que consideres correcta.  
  
Enunciado:  
Como ya sabes, TypeScript corresponde a un set adicional de herramientas para JavaScript, algunas de las principales novedades se encuentran a continuación y tu misión será asociar las definiciones a cada una de ellas. ¡Adelante!



* 1. Laboratorio

LABORATORIO DE  
Instalación y Configuración

Durante el desarrollo de una aplicación en Angular 2, puedes encontrar que hay elementos que siempre usarás y que cuentan con una estructura básica similar.   
Sin embargo, la estructuración de dichos elementos y su creación, puede tornarse dispendiosa o aburrida después de un tiempo.   
Adicionalmente, la configuración inicial de un proyecto para crear una estructura de carpetas, instalar las dependencias y programas necesarios, y hasta correr la aplicación, son labores que para el desarrollador pueden tomar un buen tiempo.   
Gracias a Angular CLI, todo esto deja de ser un problema para ti como desarrollador, ya que se encarga de realizar todas estas tareas mencionadas a partir de sencillos comandos en una terminal.

Contenido:

Para hacer uso del asistente AngularCLI debemos primero instalarlo y, para hacerlo, es importante que dispongamos la máquina que vayamos a usar para que sea apta al programa a instalar. ¿Estás preparado?

Instrucciones

* 1. Verifica si tienes Node instalado.  
     a. Ejecuta en la terminal el siguiente comando.

user$ node -v

b. Si el resultado del comando anterior es un mensaje de error indicando que no existe dicha sentencia o no se reconoce, quiere decir que no está instalado node en la máquina.  
En este caso debes dirigirte a la página oficial de Node.js y descargar el archivo de instalación, haciendo click en LTS para la última versión estable. Posteriormente debes seguir el proceso de instalación.

https://nodejs.org/es/



c. Si el resultado del comando en a. es un mensaje que indica la versión de node y ésta inicia con un número menor a 4, debes realizar los mismos pasos en b. para instalar una versión más reciente.  
  
d. Si el resultado del comando en a. es un mensaje que indica la versión de node y ésta inicia con un número mayor o igual a 4, cuentas con la versión correcta de node.

* 1. Verifica que tienes instalado npm con la versión correcta  
     a. Ejecuta en la terminal el siguiente comando.

user$ npm -v

b. Si el resultado del comando anterior es un mensaje de error indicando que no existe dicha sentencia o no se reconoce, quiere decir que no está instalado npm en la máquina. Para instalarlo debes ejecutar el siguiente comando:

user$ npm install -g

c. Si el resultado del comando en a. es un mensaje que indica la versión de node y ésta inicia con un número menor a 3, debes ejecutar el siguiente comando:

user$ npm install npm@latest -g

d. Si el resultado del comando en a. es un mensaje que indica la versión de node y ésta inicia con un número mayor o igual a 3, cuentas con la versión correcta de npm.

* 1. Instala el asistente Angular CLI  
     a. Ejecuta en la terminal el siguiente comando

user$ npm install -g angular-cli  
b. Espera unos cuantos minutos, puesto que la instalación de AngularCLI toma algún tiempo.  
c. Si durante la instalación observas un error que se presente, lee bien la descripción del mismo. Generalmente los errores que se presentan son relacionados con versiones incorrectas de node o npm.  
d. Al finalizar la instalación de manera exitosa se mostrará un mensaje y se muestra nuevamente el indicador de un nuevo comando en la terminal.

* 1. Comprueba la correcta instalación de AngularCLI creando un nuevo proyecto  
     a. El nuevo proyecto llevará el nombre “tareas” e indicamos un prefijo para todos los componentes generados a través de AngularCLI. Este prefijo se usa para evitar duplicidad en los selectores únicos de componentes para que no coincidan con elementos propios de HTML. Ejecuta el siguiente comando:

user$ ng new tareas --prefix t

Buenas Prácticas

Cada vez que realices actualizaciones o instalaciones a través de npm, es importante no hacerlo desde un super usuario (sudo) ya que pueden presentarse problemas que comprometan la seguridad del sistema en general. Si se presentan errores de instalación lee con atención las causas de los mismos para dar una solución adecuada.

1. Lección 2: Componentes y Directivas

[3.1. Componentes](https://learn.nextu.com/mod/lesson/view.php?id=3603&pid=P_WEBDEV_V2)

Componentes – una pieza de un rompecabezas que puede ser utilizado cuantas veces sean necesarias

Cada componente debe tener:

* Vista
* Controlador

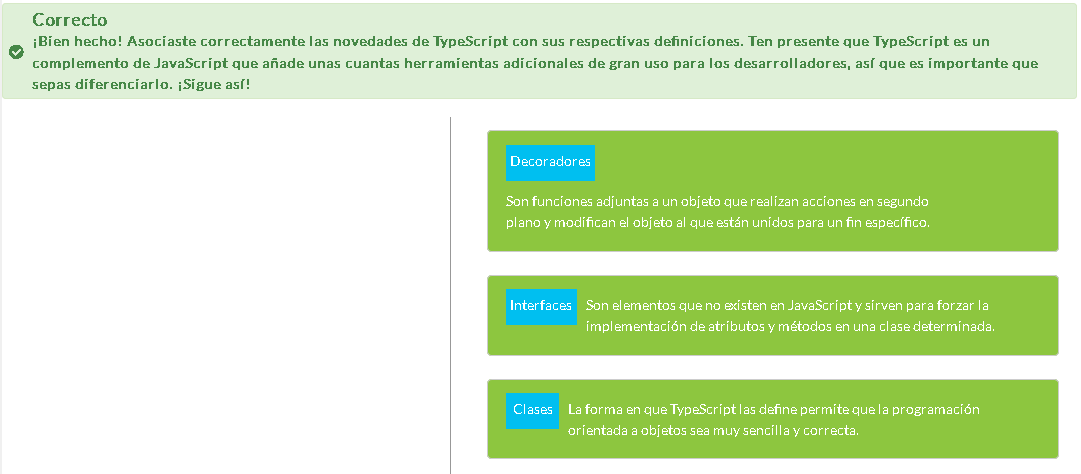
.ts 🡪 typescript

.spec.ts 🡪 Función son usados para labores de testing

[3.2. Actividad 1](https://learn.nextu.com/mod/lesson/view.php?id=3603&pageid=14488&pid=P_WEBDEV_V2)

Tiempo de pensar

Ahora que conoces cada una de las herramientas de Angular para la creación de aplicaciones y sus componentes, te invitamos a realizar la siguiente actividad para que afiances tus conocimientos.  
  
Instrucciones:  
Lee el enunciado con atención y realiza la asociación que consideres correcta.  
  
Enunciado:  
Los decoradores en los componentes sirven para ejecutar acciones en segundo plano que le indican a Angular que a lo que están unidos es un componente, y así poder usarlos. A continuación, aparecerán listados los elementos del decorador, al frente verás las definiciones, las cuales deberás asociar según corresponda. ¡Adelante!



3.3. Creación de componentes

Ejemplos:

Angular 2

Ng g component barra-superior

Angular 9

C:\Users\jrodrigue253\Github\NextUExercises\DesarrolloWeb\Modulo\_8-ToolsBuildSystems\Unidad\_3-Ang2\ejercicio1-Ang2\primer-proyecto>ng generate component barra-superior

CREATE src/app/barra-superior/barra-superior.component.html (29 bytes)

CREATE src/app/barra-superior/barra-superior.component.spec.ts (678 bytes)

CREATE src/app/barra-superior/barra-superior.component.ts (306 bytes)

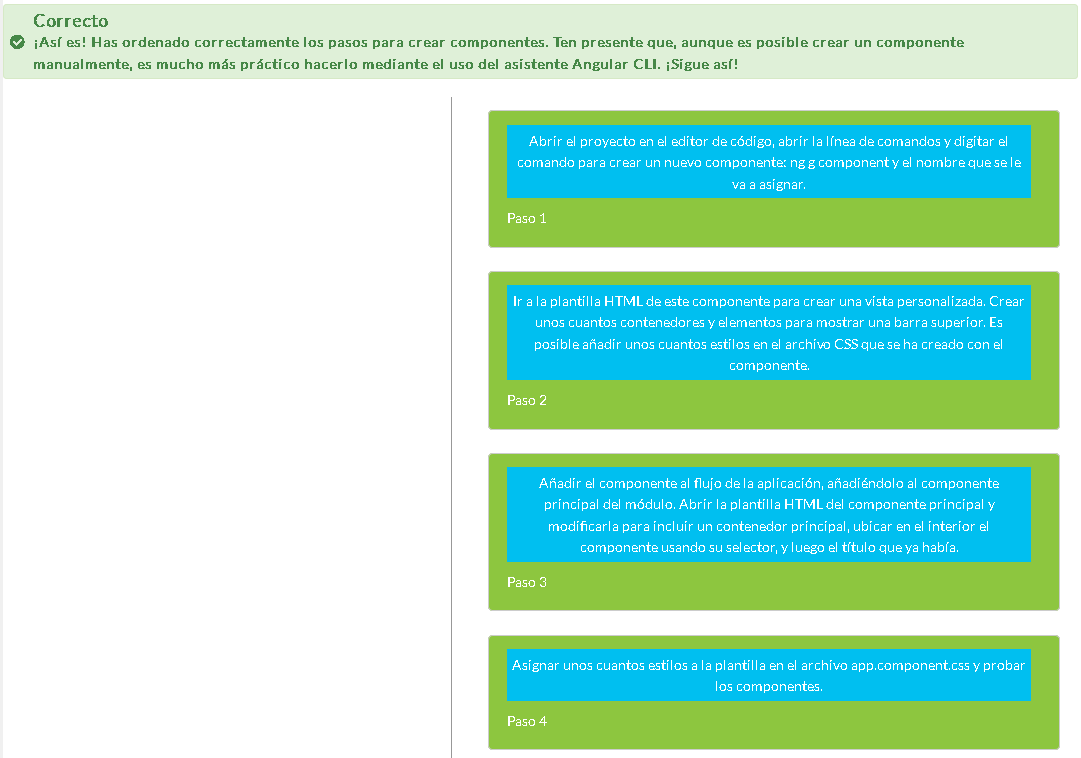
CREATE src/app/barra-superior/barra-superior.component.css (0 bytes)

UPDATE src/app/app.module.ts (426 bytes)

3.4. Actividad 2

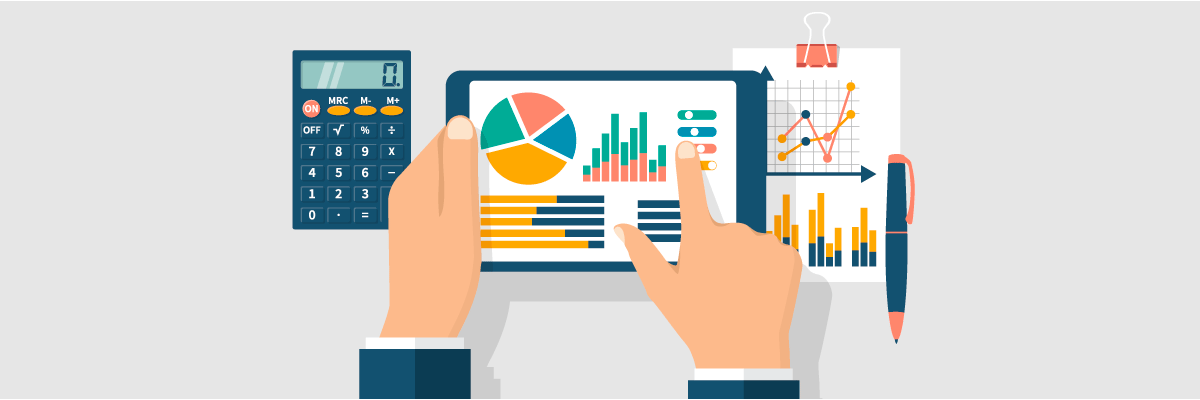
Tiempo de pensar

Ya que exploraste cómo crear tus propios componentes y verlos en acción, te invitamos a que pongas a prueba tus conocimientos con esta actividad.  
  
Instrucciones:  
Lee el enunciado con atención y realiza la asociación que consideres correcta.  
  
Enunciado:  
Como pudiste ver, un componente corresponde a un conjunto de archivos y secciones determinadas, y aunque es posible crear uno manualmente, es mucho más práctico hacer uso del asistente Angular CLI para esta tarea. A continuación aparecerán listados los pasos para la creación de componentes, tu misión será ordenarlos de acuerdo a lo que has visto hasta ahora. ¡Adelante!



3.5. Lectura: Databinding

Databinding

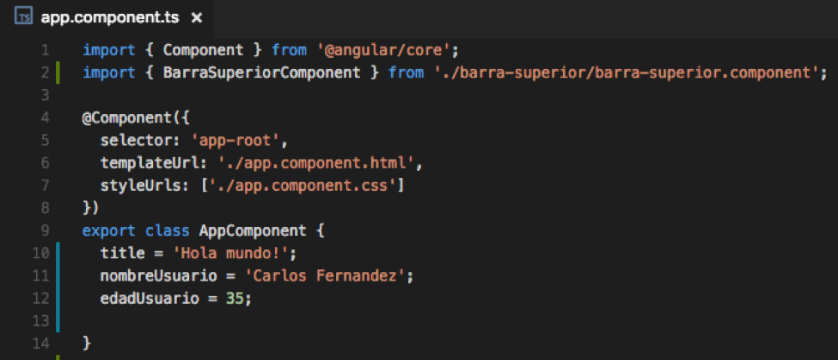


Una de las características principales en el desarrollo de aplicaciones usando Angular 2 es el concepto de databinding. Este término puede traducirse como enlace de datos y hace referencia a la comunicación que existe entre la vista y la lógica en una aplicación, es decir, la manera en la que se pasan datos desde las clases TypeScript a las plantillas HTML de los componentes.  
  
Existen 4 tipos de databinding:

* String Interpolation
* Property Binding
* Event Binding
* Two-way Binding

**String Interpolation**

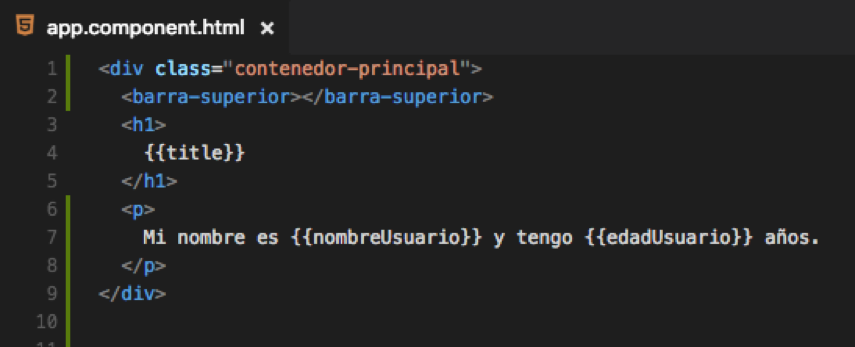
La interpolación de cadenas de caracteres, o como mejor se conoce por los desarrolladores que usan este framework string interpolation, es una forma de comunicación entre la lógica en la vista para pasar valores de tipo string o que pueden ser convertidos a string. A continuación, verás una clase en TypeScript correspondiente a un componente, en ella se define una variable con el nombre de nombreUsuario, y otra con el nombre de edadUsuario, y se le asignan valores a ambas.



Para realizar una interpolación de cadena se usa el operador doble llave y, en su interior, el nombre de la variable que queremos asignar a esa parte de la plantilla.

{{ variable }}

Ahora veamos la plantilla HTML correspondiente a este componente. Hemos creado un párrafo y, mediante el operador de doble llave, hemos indicado que deben interpolarse las variables **nombreUsuario** y **edadUsuario**.



Si vemos el resultado en el navegador, observamos que las variables se han interpolado y hemos pasado a la vista un valor asignado desde la clase TypeScript.



**Property Binding:**

El enlace de propiedades, o property binding, hace referencia a pasar datos desde la clase TypeScript de un componente a una plantilla HTML para asignar propiedades a los elementos de dicha plantilla. Funciona de manera muy similar a la interpolación de cadenas, con la diferencia que ésta se enfoca sólo en propiedades de los elementos HTML. Si añadimos, por ejemplo, un elemento de tipo input a la plantilla del componente app, podemos hacer un enlace de propiedad usando el operador corchete:

[nombre\_propiedad] = “variable o expresión”



Si vemos el resultado en el navegador, podemos observar que la propiedad value del campo de texto tomó el valor de la variable **nombreUsuario** asignada en la clase TypeScript del componente.



**Event binding:**

La asignación de eventos, o event binding como es mejor conocida, es el método por el cual se asignan eventos a los elementos de las plantillas HTML de los componentes, asociándolos con los métodos que deben disparar dichos eventos. Vamos a definir primero un método en la clase del componente app que muestre una alerta, indicando que se disparó el evento click.



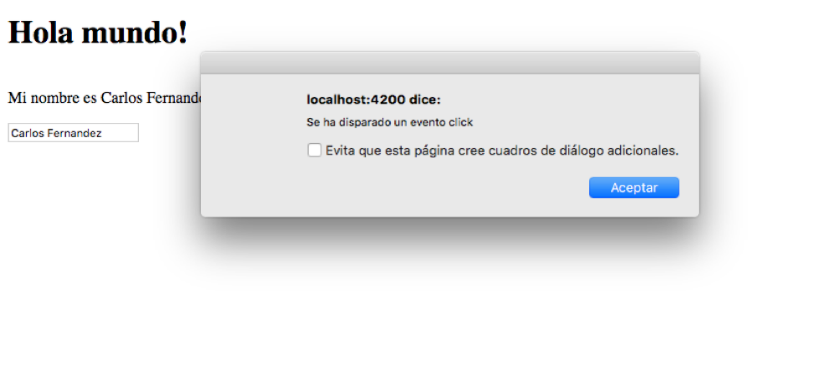
Ahora vamos a la plantilla HTML de este componente y realicemos un enlace de evento. Para enlazar un evento se usa el operador paréntesis para encerrar el nombre del evento que dicho elemento va a disparar, y se hace igual al método que debe ejecutarse al dispararse el evento indicado.

(nombre\_evento) = “método a ejecutar”

Por ejemplo, al elemento h1 de nuestra plantilla HTML vamos a asignarle un evento click, y lo hacemos igual al método que acabamos de crear: titleClicked().



Si probamos esto en el navegador, observamos que se ha asignado el evento y se dispara el método que creamos en la clase del componente.

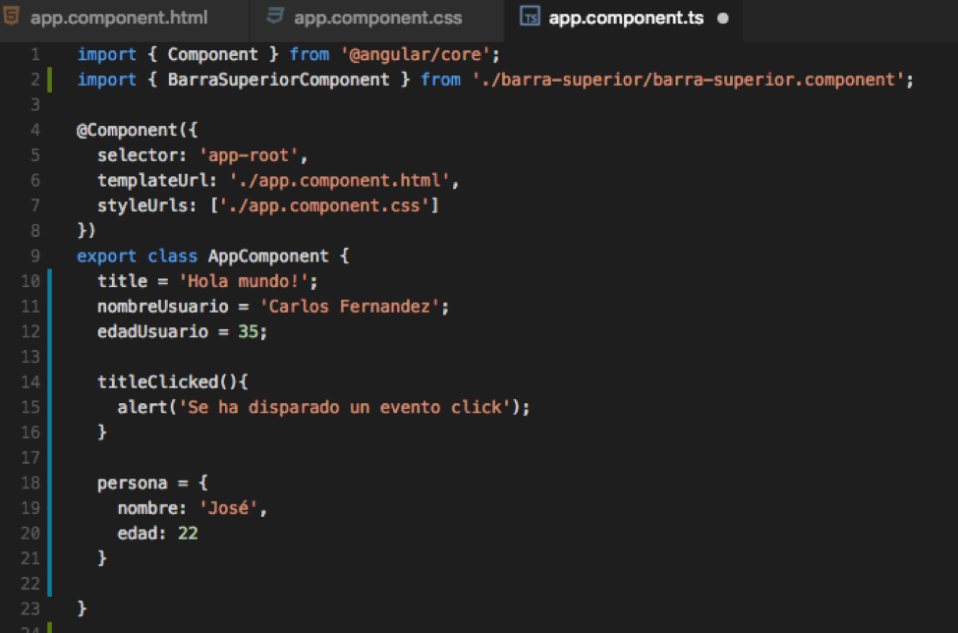


**Two-way Binding:**

El enlace bidireccional, o two-way binding como es conocido en Angular 2, hace referencia a una combinación de los dos tipos de Databinding vistos previamente: property binding y event binding. El concepto bidireccional indica que se envían datos desde la clase TypeScript y de la misma manera se reciben desde la plantilla HTML. Para definir un enlace bidireccional se deben usar, tanto los corchetes como los paréntesis, y al interior usar la palabra NgModel. A continuación, se debe asignar una propiedad de algún objeto que interactúe con la asignación:

[(NgModel)] = “objeto.propiedad”

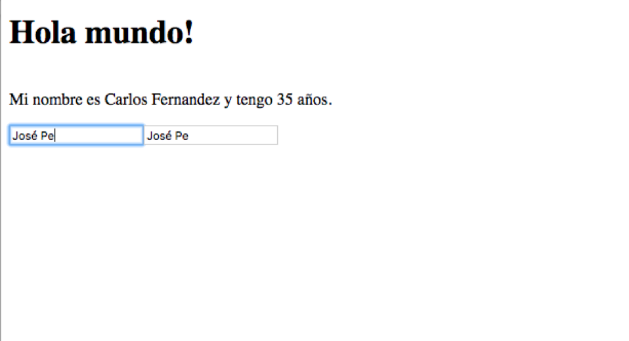
Añadimos a continuación un objeto a la clase del componente llamado persona y le definimos dos atributos, nombre y edad.



Ahora agregaremos dos elementos de tipo input a la plantilla HTML y en sus propiedades añadimos la especificación de un enlace bidireccional.



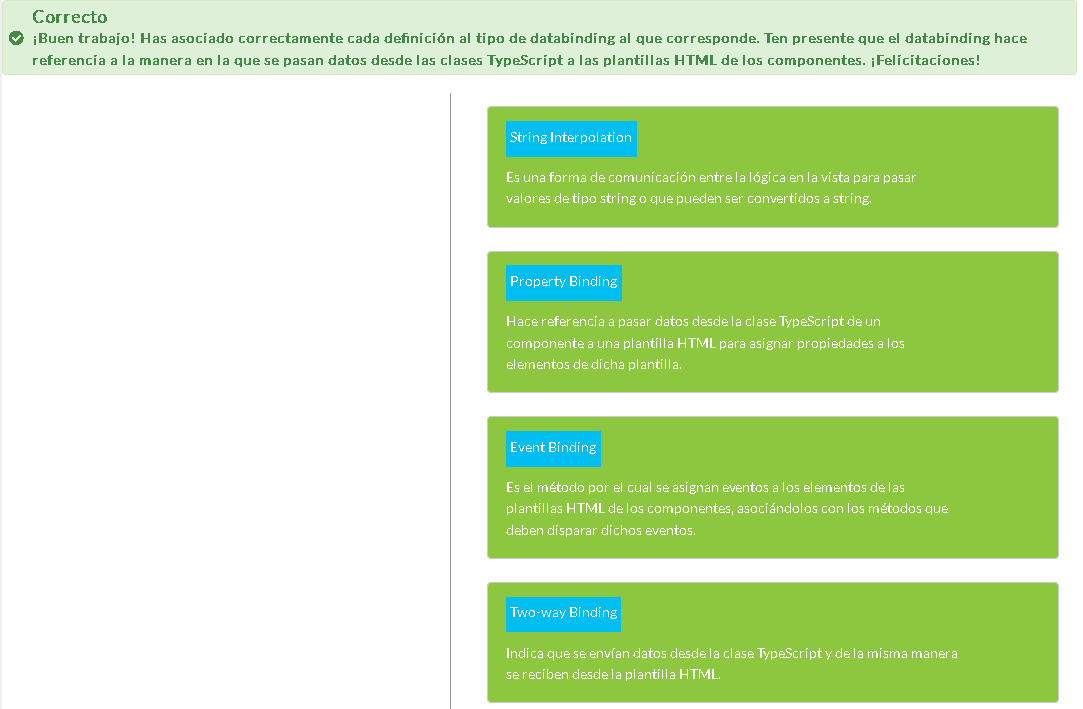
Si probamos esto en el navegador, podremos observar que, al modificar el valor de uno de los dos campos de texto, modifica el valor del otro, ya que un evento se envía al cambiar el valor del input, y al tiempo se envía el nuevo valor del atributo nombre de persona al otro campo de texto en la vista.



3.6. Actividad 3

Tiempo de pensar

Ya que conoces el databinding y cada uno de sus tipos, es tiempo de reforzar tus conocimientos con esta actividad.  
  
Instrucciones:  
Lee el enunciado con atención y realiza la asociación que consideres correcta.  
  
Enunciado:  
Una de las características principales en el desarrollo de aplicaciones usando Angular 2 es el concepto de databinding, el cual hace referencia a la comunicación que existe entre la vista y la lógica en una aplicación. Existen 4 tipos de databinding, los cuales verás a continuación. Tu misión será asociar cada definición al tipo de databinding al que corresponde. ¡Adelante!



3.7. Directivas

Directives 🡪 Set de instrucciones específicas, para los elementos del DOM en la app.

Los componentes son un tipo de directiva.

3 tipos:

* De atributo
* Estructurales
* Componentes

De atributo: son instrucciones que añaden o remueven atributos a los elementos HTML según sea necesario en la lógica de la aplicación.

[ngClass]

[ngStyle]

Estructurales: son instrucciones que alteran o modifican la estructura del DOM. Generalmente se encargan de añadir o remover elementos deacuerdo con condiciones específicas, Ej: \*nombreDirectiva

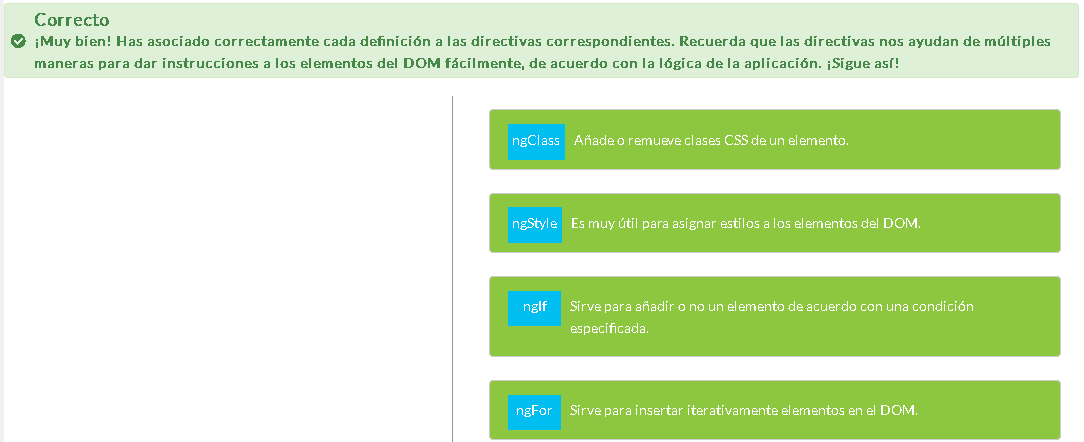
Ejemplo: \*ngIf 🡪 la cual añade un elemento o no de acuerdo con una condición específica

\*ngFor 🡪 para hacer iteraciones

3.8. Actividad 4

Tiempo de pensar

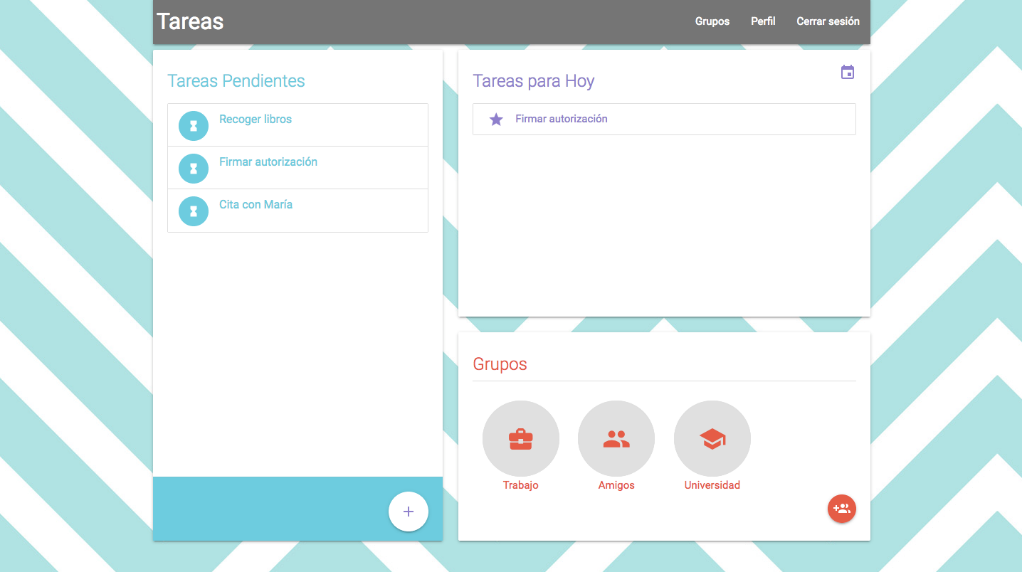
Ahora que conoces las directivas y sus 3 tipos, te invitamos a que afiances tus conocimientos con esta actividad.  
  
Instrucciones:  
Lee el enunciado con atención y realiza la asociación que consideres correcta.  
  
Enunciado:  
Las directivas en Angular se pueden entender como un set de instrucciones específicas para los elementos del DOM en nuestra aplicación. Éstas cuentan con 3 tipos: de atributo, estructurales y componentes. A continuación, verás algunas de las directivas que viste hasta ahora, tu misión será asociar cada definición la directiva a la que pertenece. ¡Adelante!



3.9 Ejercicio

HORA DE  
codificar

El desarrollo de aplicaciones usando un framework tan robusto y poderoso como Angular 2, exige un conocimiento amplio sobre las herramientas y métodos que deben utilizarse para obtener resultados óptimos y correspondientes a las necesidades de los clientes. El fundamento estructural de una aplicación web en Angular 2 son los componentes, los cuales permiten un comportamiento modular del portal en donde se reduce la duplicidad de código y en el que se hace más sencillo el desarrollo de partes individuales que componen una parte mayor.   
  
En este ejercicio iniciaremos el desarrollo de una aplicación web para la programación de tareas y recordatorios. Tu misión es crear los componentes iniciales de la aplicación, que conformen el dashboard o página inicial de la misma. Debes aplicar tus conocimientos en Angular 2, pero a su vez, todas las herramientas vistas durante este programa que puedan servirte de ayuda.  
  
Al finalizar este ejercicio, tendrás un proyecto en Angular 2 con una apariencia similar a la siguiente:



Contenido:

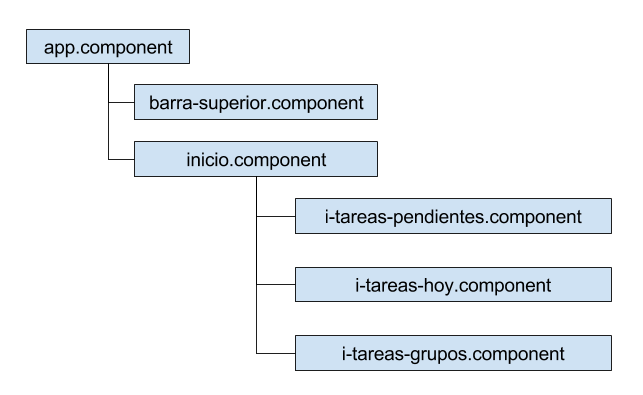
Para el desarrollo de este ejercicio, debes desarrollar sobre el proyecto de Angular 2 creado en el laboratorio propuesto en la lección anterior. Sin embargo, si no cuentas con él, puedes crear uno nuevo ejecutando el comando:

user$ ng new tareas --prefix t

Se creará entonces el proyecto y podrás iniciar con el desarrollo de este ejercicio.

Instrucciones

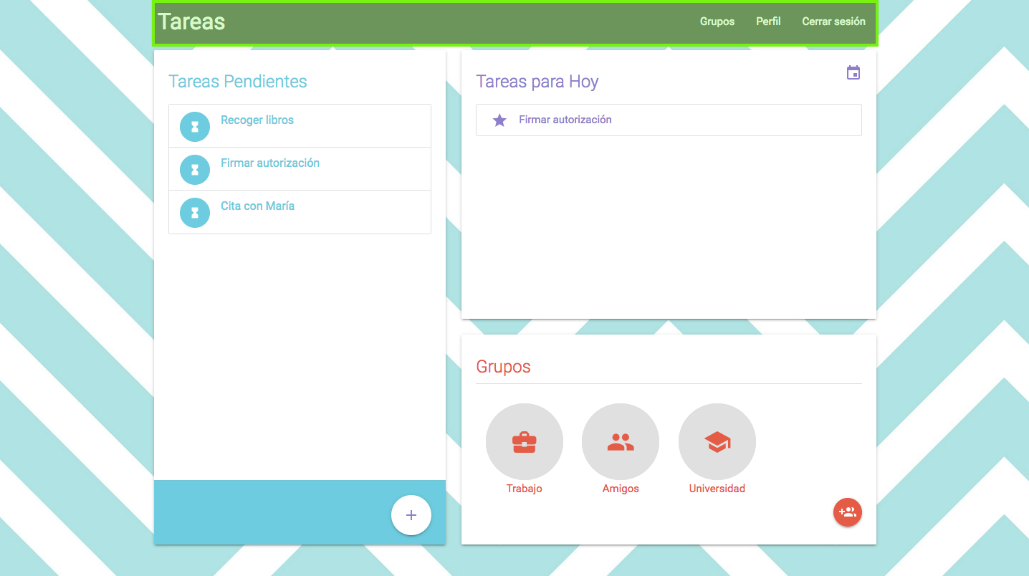
* 1. Importa el framework CSS de tu preferencia ubicando el **CDN** correspondiente en el archivo **index.html.**
  2. Crea algunas reglas de estilo general para toda la aplicación en el archivo **styles.css;** debes añadir una imágen de fondo para el elemento body. Dicha imágen de fondo debe ir ubicada en el directorio **assets.**
  3. Guíate por el siguiente diagrama para la creación y estructuración de los componentes iniciales de la aplicación:



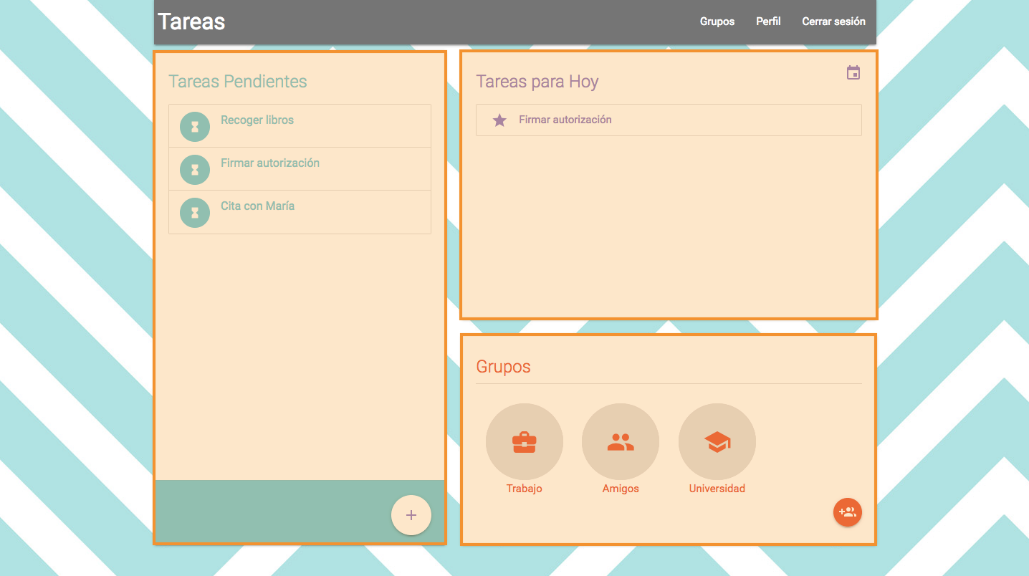
* 1. Define la estructura del componente **app** para que se ubiquen los componentes barra-superior e inicio de acuerdo con el siguiente diagrama.



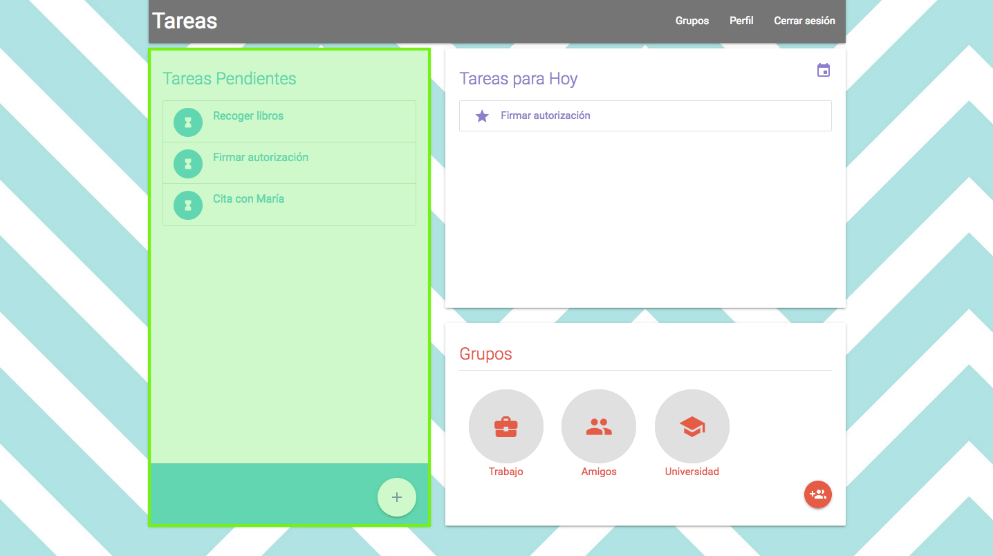
* 1. Crea el componente **barra-superior** como una barra de navegación con el logo de la aplicación y tres enlaces: Grupos, Perfil y cerrar sesión.



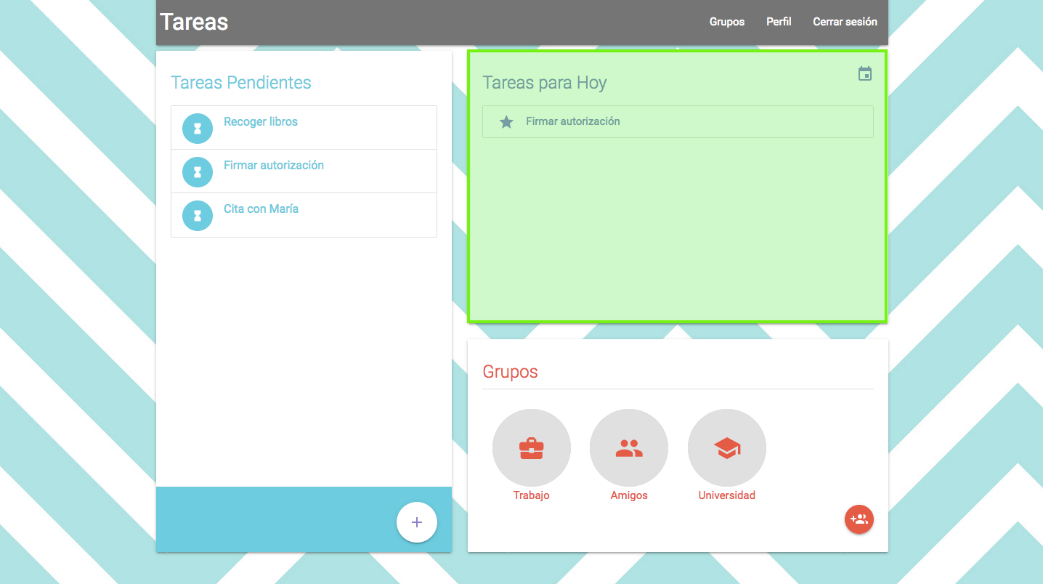
* 1. Crea el componente inicio que contenga los componentes tareas-pendientes, tareas-hoy y tareas-grupos de acuerdo con la siguiente estructura:



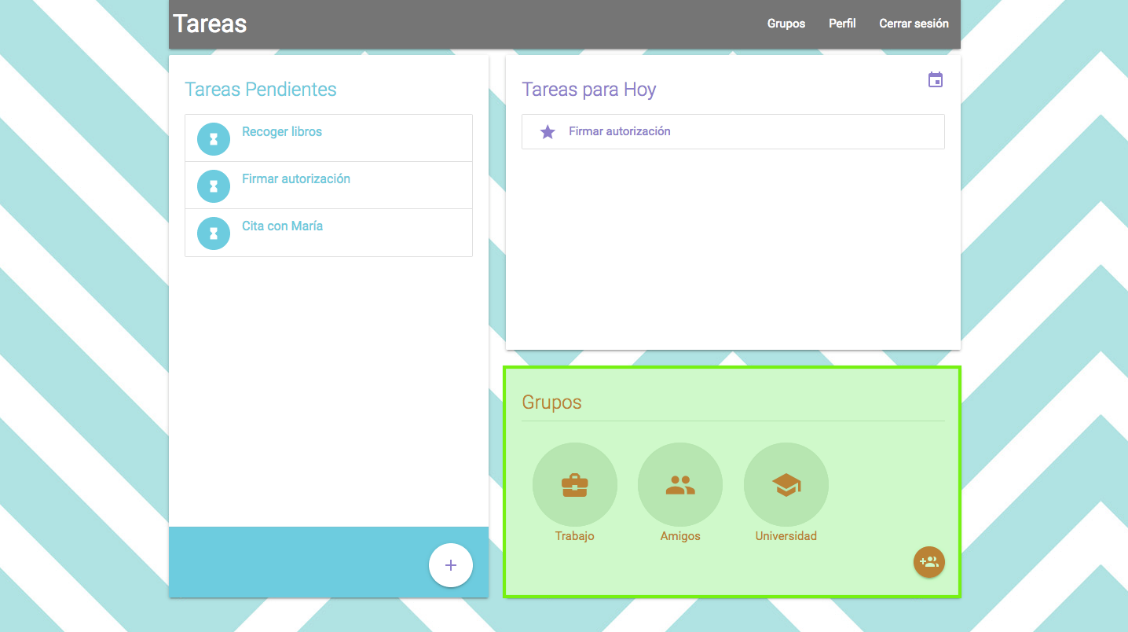
* 1. Crea el componente **tareas-pendientes** que contenga un título y una lista de enlaces. En la parte inferior crea una barra de acción con un botón para adicionar una nueva tarea.



* 1. Crea el componente **tareas-hoy** que contenga un título y una lista de enlaces.



* 1. Crea el componente **tareas-grupo** que contenga un título y una lista, la cual debe tener por cada ítem, un logo y un nombre. Adicionalmente se debe añadir un botón que permita añadir un nuevo grupo.



* 1. Usa las herramientas de Angular 2 para cargar las tareas en el componente **tareas-pendientes.** Éstas deben ser cargadas a partir de un arreglo de tipo string.
  2. Usa las herramientas de Angular 2 para cargar las tareas en el componente **tareas-hoy.** Éstas deben ser cargadas a partir de un arreglo de tipo string.
  3. Usa las herramientas de Angular 2 para cargar los grupos del componente **tareas-grupos.** Éstos deben ser cargados a partir de un arreglo de objetos, en donde cada objeto debe contar con las siguientes propiedades: nombre (un string con el nombre del grupo), ícono (el identificador de un ícono asociado con el grupo) y resaltado (de tipo boolean, que debe ser false por defecto)
  4. Crea una clase llamada **grupo-resaltado** en CSS que asigne el color de fondo a blanco. Añade a los círculos que contienen el ícono del grupo, una directiva de clase que haga referencia a la clase **grupo-resaltado** y la haga igual a la propiedad resaltado de cada objeto ítem de la lista de grupos.
  5. Asigna dos eventos a los círculos que contienen el ícono de grupo, que disparen dos métodos. Cuando el cursor entre al elemento se debe disparar un método que haga a la propiedad **resaltado** del objeto correspondiente igual a **true.** Cuando el cursor salga del elemento se debe disparar un método que haga la propiedad **resaltado** del objeto correspondiente igual a **false.** De esta manera se tendrá un efecto de hover sobre los elementos.

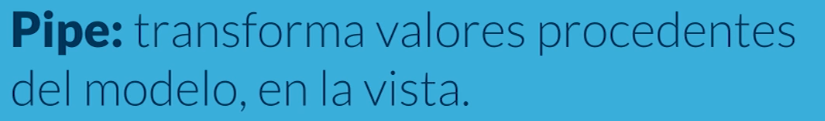
**Buenas Prácticas**

Al desarrollar aplicaciones en Angular 2, debes pensar y diseñar la estructura de una página a partir de las partes que la conforman, esto es, pensar en componentes, para así, tener mayor claridad al momento de su creación. Además, es muy recomendable agrupar componentes por vista, es decir, un componente mayor que contenga todos los componentes de una vista. Esto será de gran ayuda para cuando se establezca el enrutamiento en la aplicación.

1. Lección 3: Pipes, Servicios y Routing

[4.1. Pipes](https://learn.nextu.com/mod/lesson/view.php?id=3604&pid=P_WEBDEV_V2)

Pipe 🡪 Tubo 🡪 |



Lista de Pipes en Angular

<https://angular.io/api?type=pipe>

Se pueden crear pipes personalizados:

…\pipe>ng generate pipe duplicar

CREATE src/app/duplicar.pipe.spec.ts (195 bytes)

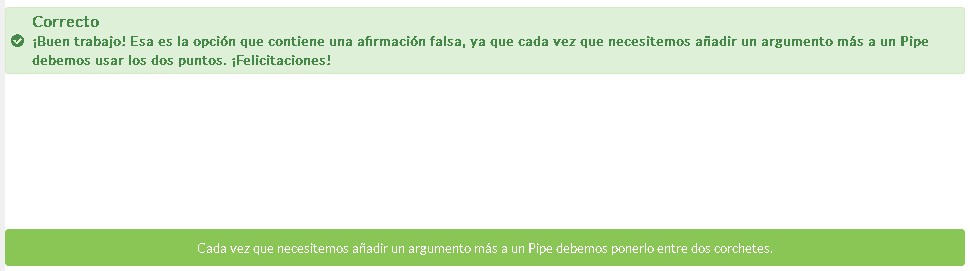
CREATE src/app/duplicar.pipe.ts (221 bytes)

UPDATE src/app/app.module.ts (380 bytes)

[4.2. Actividad 1](https://learn.nextu.com/mod/lesson/view.php?id=3604&pageid=14498&pid=P_WEBDEV_V2)

Tiempo de pensar

Ahora que conoces qué son los Pipes y para qué sirven en Angular 2, te invitamos a realizar la siguiente actividad para que afiances tus conocimientos.  
  
Instrucciones:  
Lee el enunciado con atención y elige la opción que consideres correcta.  
  
Enunciado:  
A continuación verás algunas afirmaciones sobre los Pipes en Angular 2. Tu misión será encontrar la afirmación que es falsa, pues no corresponde a lo que has visto hasta ahora. ¡Adelante!



4.3. Routing

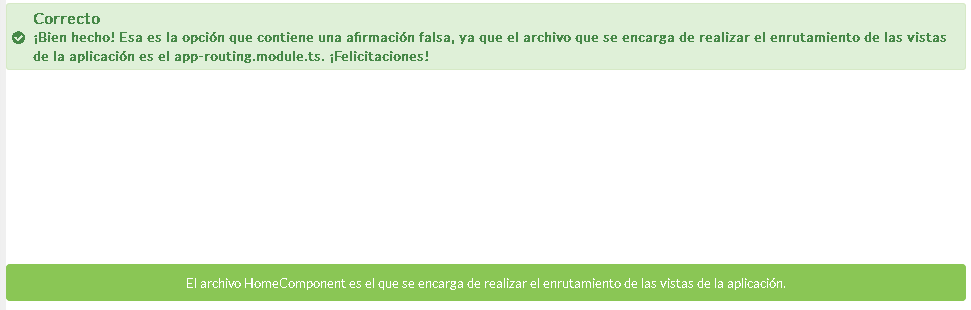
Angular X 🡪 SPA (Single Page Application)



4.4. Actividad 2

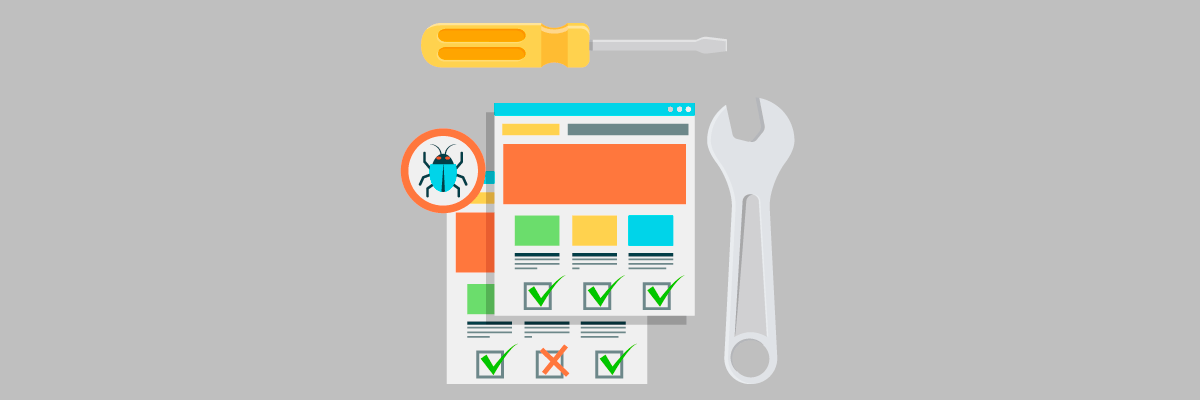
Tiempo de pensar

En esta oportunidad exploraste cómo realizar el enrutamiento a través de las vistas en Angular 2, ahora te invitamos a reforzar tu conocimiento con la siguiente actividad.  
  
Instrucciones:  
Lee el enunciado con atención y elige la opción que consideres correcta.  
  
Enunciado:  
A continuación verás algunas afirmaciones sobre el enrutamiento a través de las vistas en Angular 2. Tu misión será encontrar la afirmación que es falsa, pues no corresponde a lo que has visto hasta ahora. ¡Adelante!

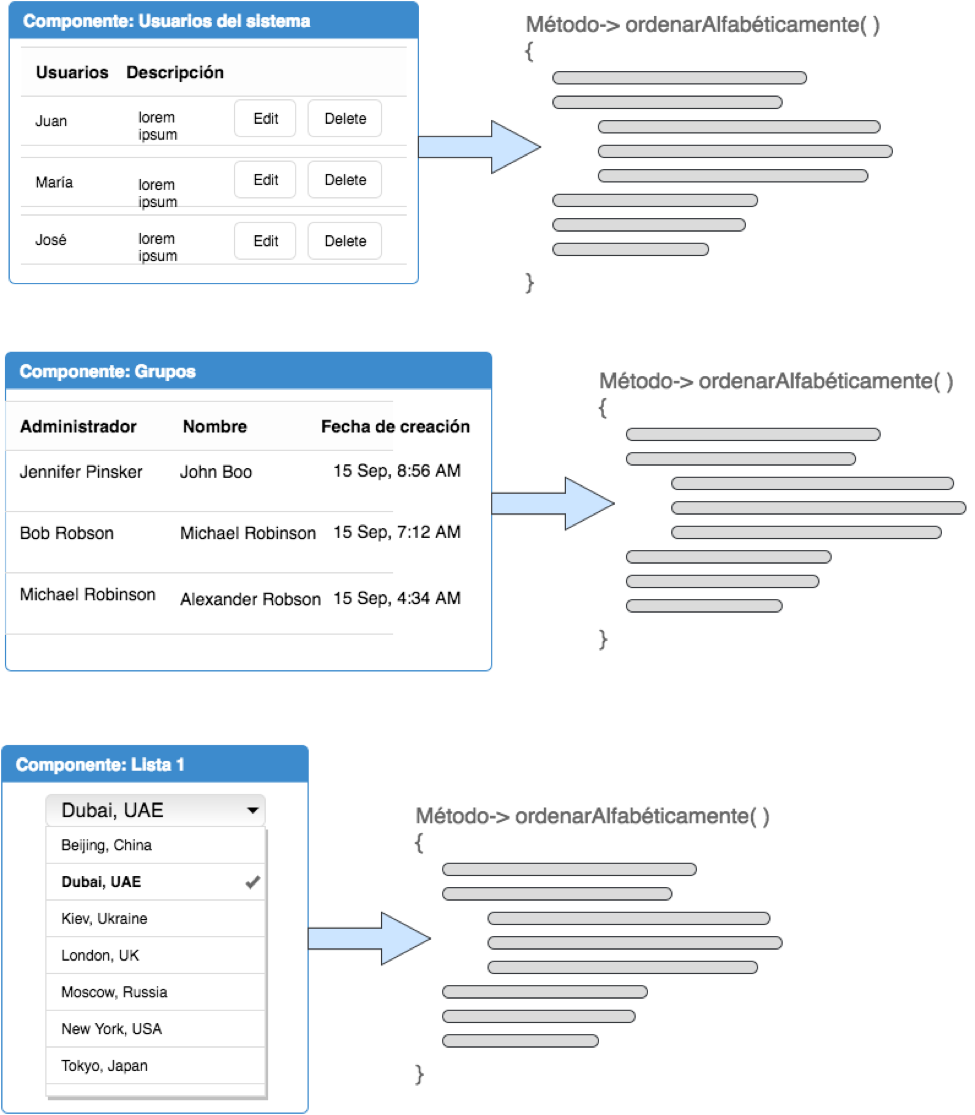


4.5. Lectura: Servicios

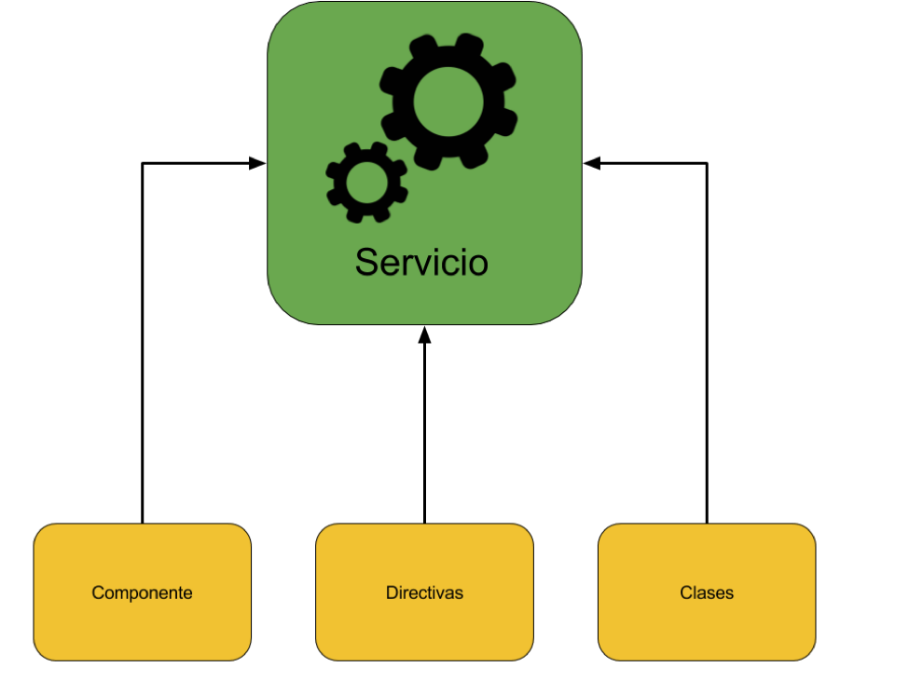
Servicios



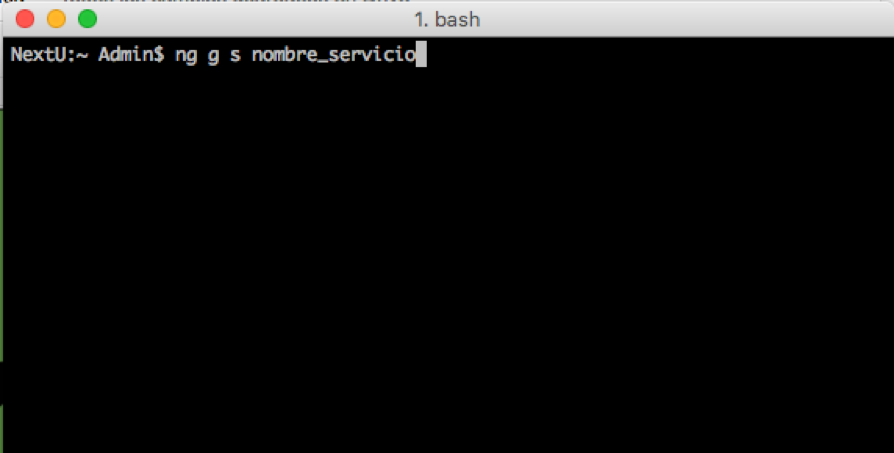
Una de las herramientas más usadas en el desarrollo de aplicaciones con Angular 2 son los servicios, por lo tanto, es preciso comprender un poco más a fondo este importante concepto y tener claro en qué situaciones nos puede ser de utilidad para el desarrollo de nuestros proyectos.  
  
Cuando creamos componentes para conformar una vista o un módulo completo, encontramos que no sólo debemos definir plantillas HTML o estilos CSS para dar una apariencia adecuada a cada elemento, sino que también debemos crear ciertos métodos en la lógica del componente para ejecutar determinadas acciones como eventos, procesos, entre otros.  
  
Al trabajar de esta manera, es muy posible que veamos métodos que tienen la misma funcionalidad en componentes diferentes y, de esta manera, encontramos métodos repetidos con el mismo código en más de un archivo. Un ejemplo de esto puede ser un método que se encargue de ordenar un arreglo de palabras en orden alfabético, el cual puede ser utilizado en un componente que muestre una tabla con los usuarios del sistema, pero también se usa en un componente que muestra los grupos creados u otro que muestre listas con cualquier tipo de información. En este caso, el método ordenarAlfabeticamente estaría definido en 3 o más componentes, así tenga el mismo código cada vez.



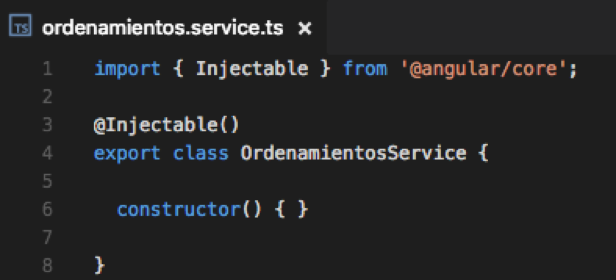
Lo mostrado en el ejemplo es claramente una mala práctica de desarrollo y para solucionarlo, Angular2 ofrece los servicios. Un servicio es una clase que puede ser instanciada en uno o más componentes, y de la cual se pueden ejecutar métodos y obtener propiedades que sirvan para la lógica y procesamiento de la aplicación. Un servicio puede ser instanciado en componentes, directivas o clases en general. Su principal objetivo es la reusabilidad del código, eliminando los métodos duplicados que puedan afectar al rendimiento de la aplicación.



Para crear un nuevo servicio en un proyecto, hacemos uso del asistente Angular CLI, el cual crea el archivo del servicio con la estructura necesaria. El código a ejecutar en la terminal para crear un nuevo servicio es el siguiente:



Este comando genera un archivo TypeScript con una clase acompañada por un decorador que explicaremos más adelante. En el siguiente ejemplo se muestra el código generado para un servicio llamado ordenamiento:

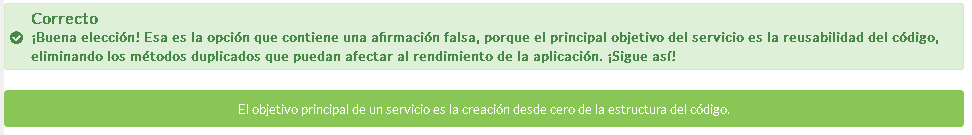


En la clase creada, OrdenamientosService, es donde podremos crear los métodos y atributos que serán usados por los componentes, directivas o clases al implementar el servicio, de esta manera se crean servicios y se añade el código pertinente para que podamos usar estas herramientas desde diversos elementos de la aplicación, evitando la duplicación de código y aumentando así el rendimiento general del sistema.

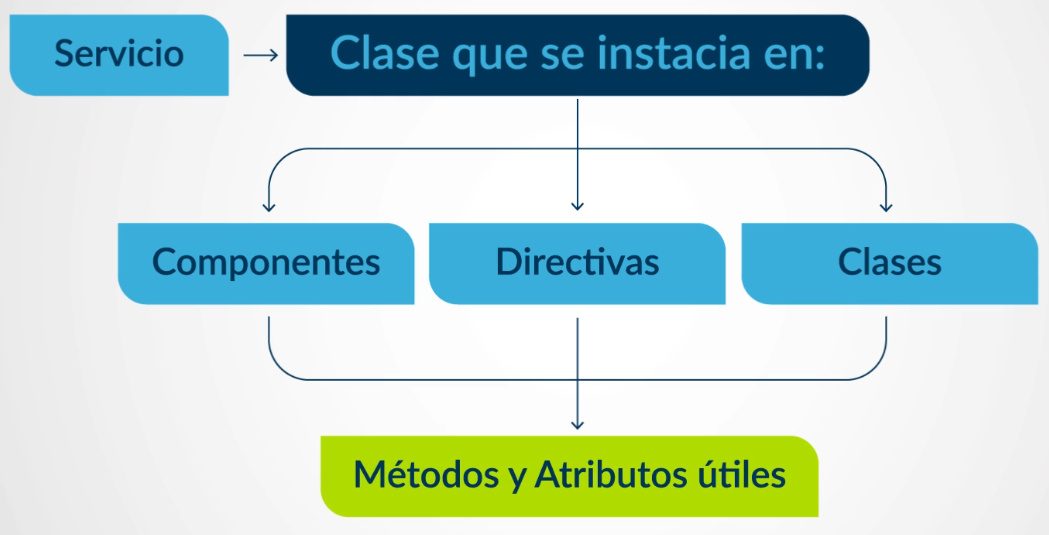
4.6. Actividad 3

Tiempo de pensar

Ahora que sabes que una de las herramientas más usadas en el desarrollo de aplicaciones con Angular 2 son los servicios, te invitamos a evaluar tu conocimiento con la siguiente actividad.  
  
Instrucciones:  
Lee el enunciado con atención y elige la opción que consideres correcta.  
  
Enunciado:  
A continuación verás algunas afirmaciones sobre los servicios en el desarrollo de nuestros proyectos. Tu misión será encontrar la afirmación que es falsa, pues no corresponde a lo que has visto hasta ahora. ¡Adelante!



4.7. Dependency Injection

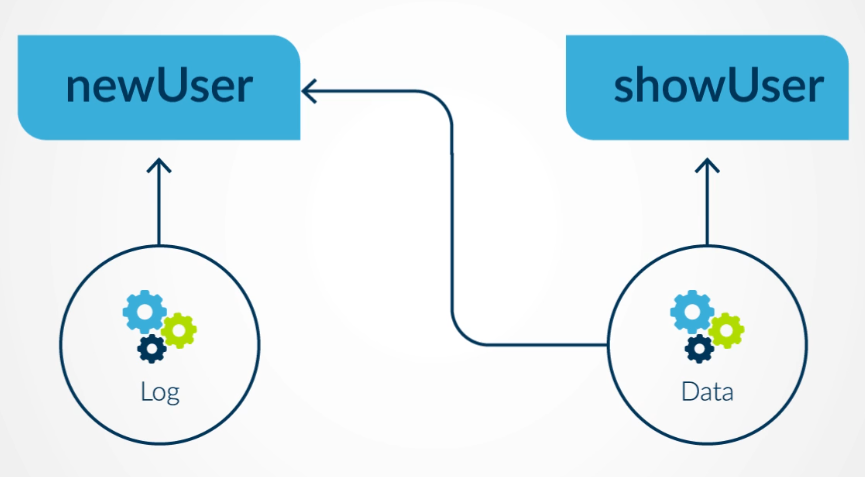




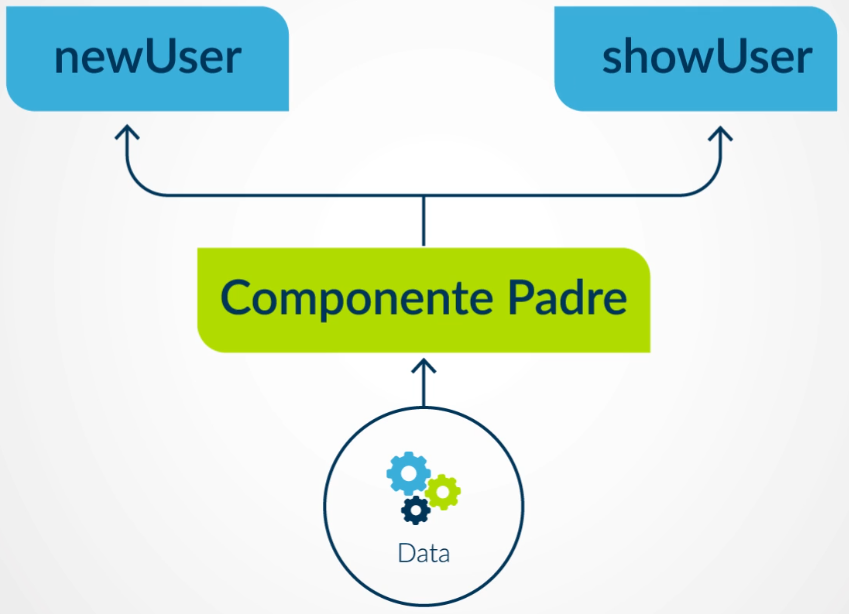


Ejemplo 1: Servicio Log 🡪 Se inyecto un servicio para un solo componente

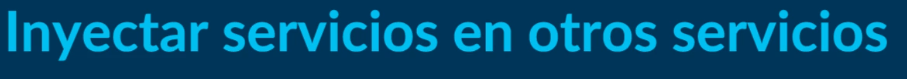
Ejemplo 2: Servicio Data 🡪 Se inyecta tanto para el componente newUser como para showUser



Ejemplo 2:



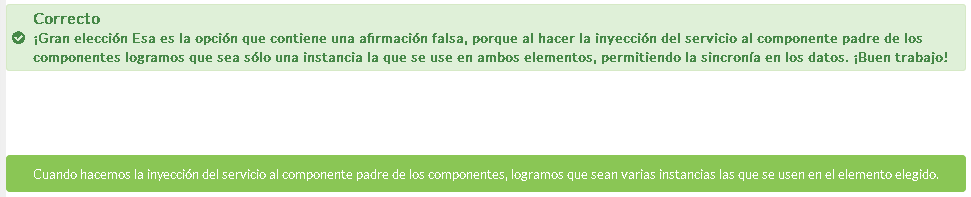
Es posible:



4.8. Actividad 4

Tiempo de pensar

Ya que conoces el concepto de Dependency Injection, te invitamos a realizar la siguiente actividad que te ayudará a reforzar tus conocimientos.  
  
Instrucciones:  
Lee el enunciado con atención y elige la opción que consideres correcta.  
  
Enunciado:  
A continuación verás algunas afirmaciones sobre el Dependency Injection, que en Angular 2 hace referencia a poner en funcionamiento un servicio para uno o más elementos. Tu misión será encontrar la afirmación que es falsa, pues no corresponde a lo que has visto hasta ahora. ¡Adelante!



4.9. Ejercicio

HORA DE  
**codificar**

Una de las dificultades más comunes en el desarrollo basado en componentes, es la comunicación entre estos elementos para contar con piezas de información unificadas que alimenten a toda la aplicación. En Angular 2, el concepto de servicios ha sido estructurado de tal manera que cada componente consuma los datos que son deseados, y de acuerdo con la cantidad de elementos que deben usar la misma información, se asigna el alcance de los servicios.  
  
Por otra parte, la navegación entre las vistas de una aplicación tipo SPA, es crucial en cuanto a la estabilidad y fluidez de la misma, ya que ésta es una de las características principales por las que se eligen este tipo de aplicaciones frente a las tradicionales. En Angular 2 encontramos un enrutador muy fácil de usar y que, gracias a su flexibilidad, podemos sacar grandes provechos de la navegación entre componentes.  
  
En este ejercicio continuaremos el desarrollo de la aplicación de tareas y recordatorios que iniciamos la lección anterior. Continúa a partir del proyecto que realizaste en el ejercicio anterior o ya bien, descarga el código base que se incluye a continuación.

Contenido:

Para el desarrollo de este ejercicio, debes desarrollar sobre el proyecto de Angular 2 creado en el ejercicio propuesto en la lección anterior. Sin embargo, si no cuentas con él, puedes crear uno nuevo ejecutando el comando:

user$ ng new tareas --prefix t

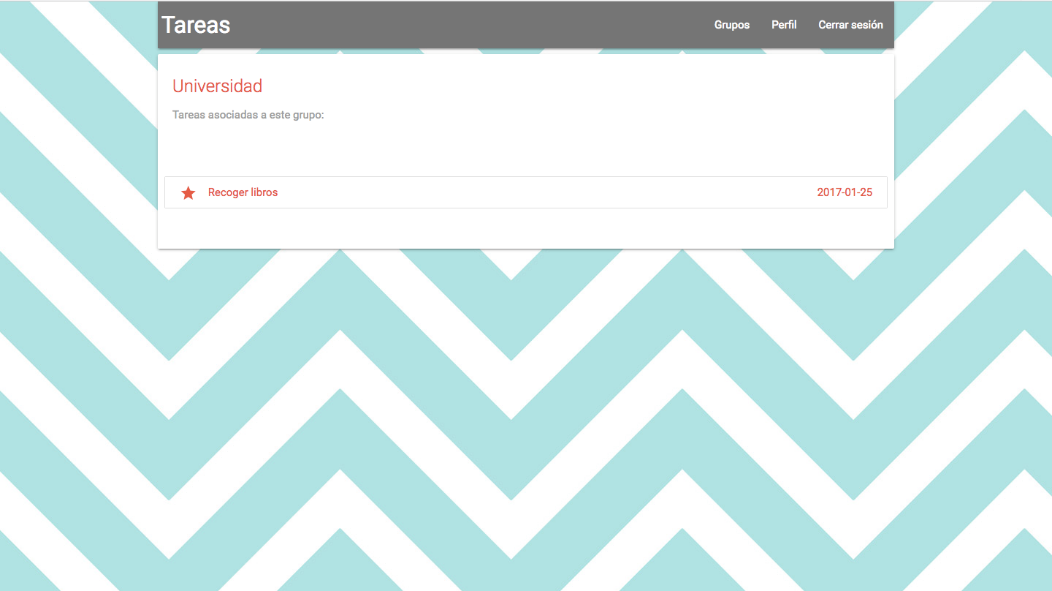
Se creará entonces el proyecto, y deberás reemplazar la carpeta src que se creó en la estructura del proyecto, por la que se incluye en el código base..

Instrucciones

* 1. Crea un componente llamado **nueva-tarea** en el que se pueda ingresar en un pequeño formulario los datos de una nueva tarea. Debe contener los campos título de la tarea, descripción y fecha. Adicionalmente debe contar con los botones de enviar y cancelar. En el siguiente ejemplo se muestra una propuesta gráfica para este componente:



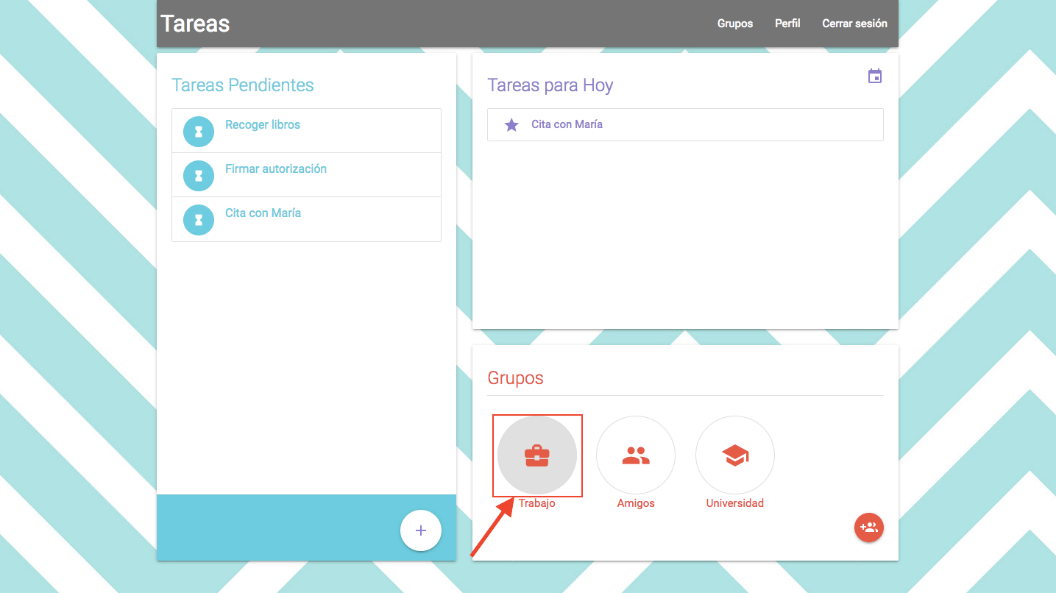
* 1. Crea un componente llamado **ver-grupo** en el que se puedan visualizar todas las tareas asociadas con un grupo. Este componente debe contar con un título que muestre el nombre del grupo, y una lista de tareas por grupo. Cada tarea debe mostrar su nombre y su fecha. En el siguiente ejemplo se muestra una propuesta gráfica para este componente:



* 1. Crea un enrutamiento que al hacer click en el botón más (+) de la vista principal, se dirija al componente **nueva-tarea.**



* 1. Crea un enrutamiento que al hacer click en el botón de cada grupo, la aplicación se dirija al componente **ver-grupo.**



* 1. Crea un servicio en el que se definan dos arreglos de objetos, uno para grupos y otro para tareas. El arreglo de objetos de tipo grupos es el mismo que definimos en el componente **i-tareas-grupos,** muévelo de ese componente hacia el servicio. Para las tareas cada objeto debe contener las propiedades: título, descripción, fecha, grupo.
  2. Modifica la estructura de los componentes que hacen parte de la vista inicio, para que las tareas o grupos que se muestren sean obtenidos del servicio creado, esto quiere decir que todos estos componentes deben consumir el mismo servicio, compartiendo la información unificada de tareas y grupos.
  3. Para el componente **i-tareas-hoy** crea un método en el servicio que compare la propiedad fecha de todas las tareas con la fecha de hoy, y de allí arroje todas las tareas que son para el día de hoy.
  4. En la ruta que definiste para acceder al componente **ver-grupo** en el archivo de enrutamiento, debes añadirle a la propiedad **path** un identificador de ruta usando dos puntos y el nombre del identificador:

path: ‘ver-grupo/:nombre’

* 1. Modifica el routerLink que creaste para la navegación entre el botón grupo y el componente ver-grupo, añadiendo el nombre del item del ciclo ngFor como segundo elemento del arreglo. Esto lo hacemos para pasar como parámetro en la ruta, el nombre del grupo al que vamos a acceder.

[routerLink]="['ver-grupo', item.nombre]"

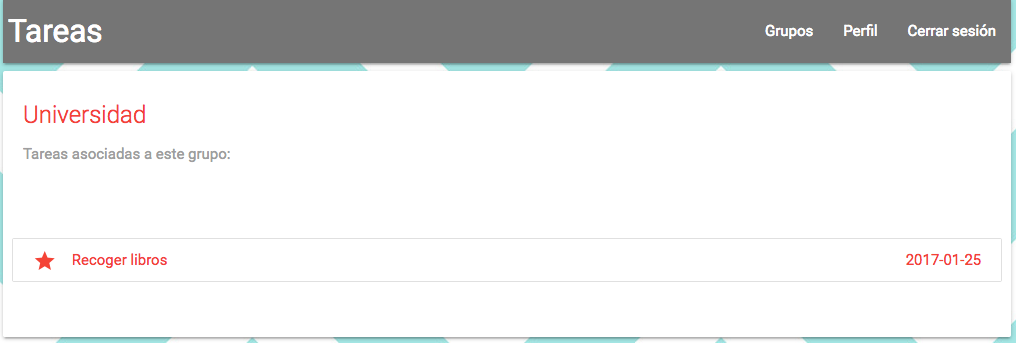
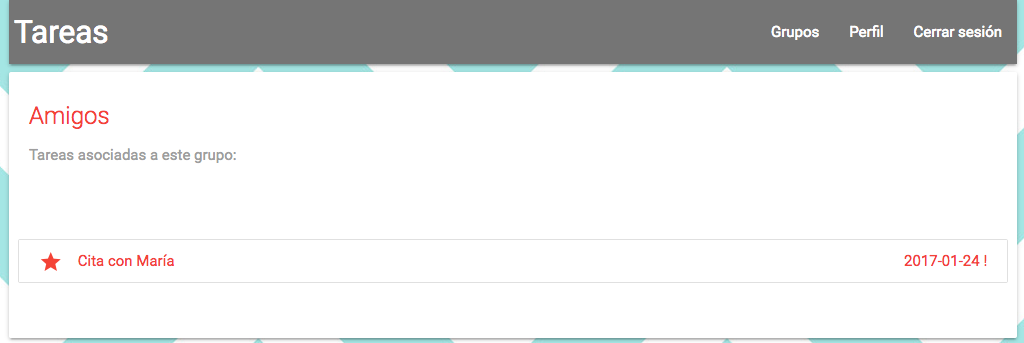
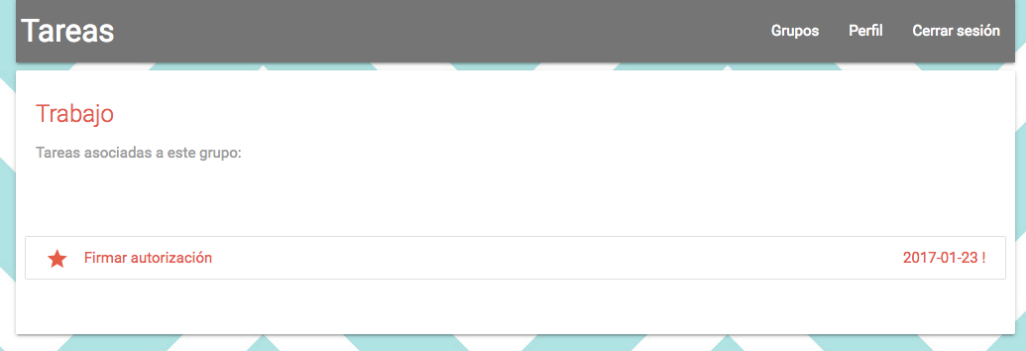
* 1. En la clase del componente ver-grupo, crea en el constructor un objeto de tipo ActivatedRoute, importando este módulo de @angular/route.

constructor(private activatedRoute : ActivatedRoute){ … }

* 1. Dentro del constructor del componente ver-grupo, asigna a una propiedad de la clase el valor recibido en la ruta. Esto se hace para conocer si se accedió a ver un grupo en específico, disponer de la información correspondiente a ese grupo. Para obtener el valor de una variable pasada como parámetro en una ruta usamos la siguiente expresión, en donde ‘nombre’ fue el nombre del parámetro que indicamos en la ruta en el archivo de enrutamiento después de los dos puntos:

this.nombreGrupo = this.activatedRoute.snapshot.params['nombre'];

* 1. En el componente **ver-grupo** muestra el nombre del grupo al que se accedió en el título del componente, y llena la lista de tareas con las tareas correspondientes a dicho grupos, es decir, todas las tareas, que en la propiedad grupo tengan un valor que coincida con el nombre del grupo:



* 1. Crea un Pipe personalizado que evalúe la fecha de las tareas en el componente **ver-grupo.** El Pipe debe tomar el valor de la fecha y determinar si la fecha ya pasó, en caso tal debe añadir un signo de exclamación (!) al final de la fecha, de lo contrario no se debe mostrar nada. Observa las imágenes del paso anterior donde se evidencia el funcionamiento del Pipe.

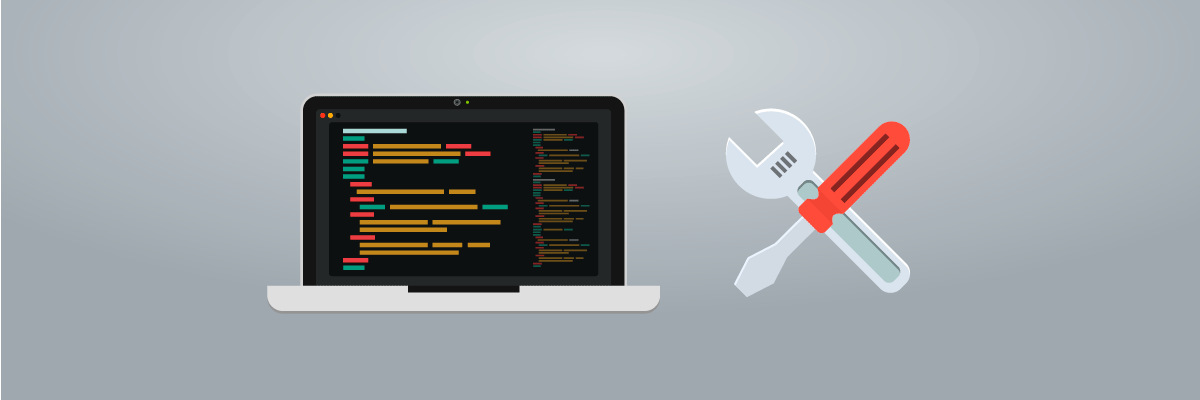
##### Buenas Prácticas

Recuerda que los servicios en Angular 2, funcionan en sincronía entre componentes siempre y cuando sean añadidos en los providers de un componente padre de los componentes que consumirán el servicio. Si un servicio, va a ser consumido de manera global en toda la aplicación, entonces los providers del servicio deben ser añadidos en **app.module.ts.**

1. Lección 4: Comunicación con el Back-end

[5.1. Lectura: HTTP y Observables](https://learn.nextu.com/mod/lesson/view.php?id=3605&pid=P_WEBDEV_V2)

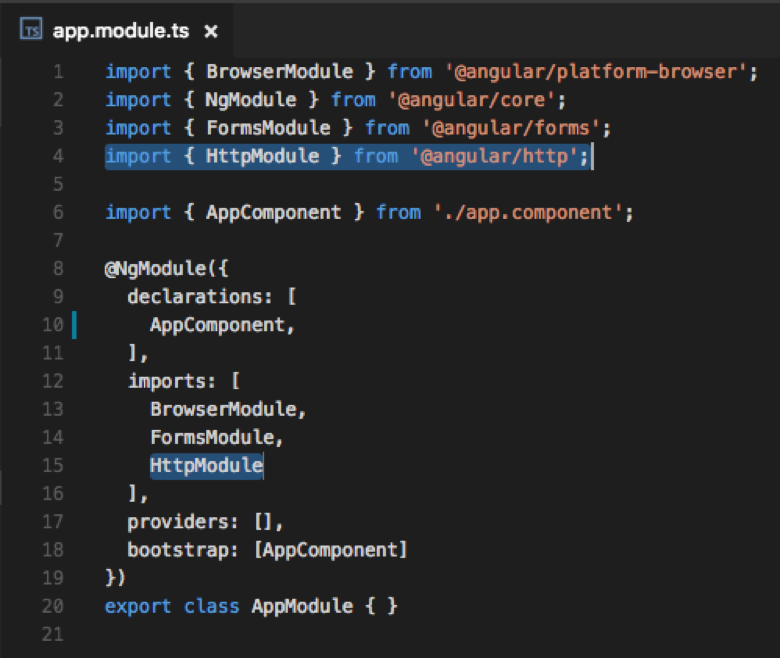
Http y Observables



Angular 2 es un framework que permite estructurar aplicaciones Web tipo SPA y trabaja por medio de la filosofía de componentes. Sin embargo, como toda aplicación debe estar en constante comunicación con un Back-end que contenga la lógica del negocio, Angular provee un sistema muy conveniente para este tipo de comunicación.  
  
Para crear una comunicación adecuada entre una API, o ya bien cualquier tipo de servicio de Back-end, Angular usa el módulo Http, el cual incluye una gran cantidad de herramientas y utilidades que permiten hacer peticiones, recibir respuestas y gestionar dichas comunicaciones.  
  
Uno de los elementos más importantes en la comunicación con servicios de Back-end desde un aplicación construída en Angular 2 son los observables, los cuales son elementos que se asignan a un determinado tipo de petición para estar pendientes (observando) de los posibles flujos de datos que existan en la comunicación. Esto quiere decir que un observable se asigna a una ruta específica de una API por ejemplo, y cada vez que se reciba una respuesta, el observable disparará un callback determinado para darle un destino a los datos recibidos desde el Back-end.  
  
El método por el cual se asigna un observable a un servicio determinado se llama subscribe, del inglés suscribir, ya que el observable técnicamente se suscribe a una comunicación pendiente de todo tipo de peticiones y respuestas que sean ejecutadas.



Si usas el asistente Angular CLI para la creación de proyectos, puedes ver que por defecto se realiza la importación del módulo Http en el archivo app.module.ts, el cual recibe las importaciones a nivel global de la aplicación. Esto quiere decir, que al incluir el módulo Http en este archivo, podremos usarlo en cualquier elemento de la aplicación. A continuación, se muestra la importación del módulo Http que permite la creación de observables:

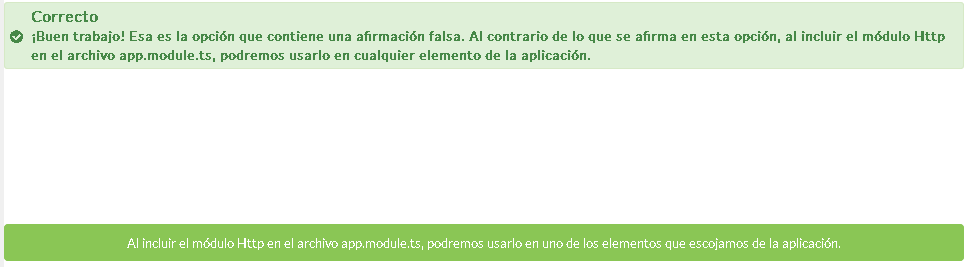


5.2. Actividad 1

Tiempo de pensar

Ahora que conoces qué es el Http y los observables, te invitamos a realizar la siguiente actividad para que afiances tus conocimientos.  
  
Instrucciones:  
Lee el enunciado con atención y elige la opción que consideres correcta.

Enunciado:  
A continuación te presentamos algunas afirmaciones sobre el Http y los observables, tu misión será encontrar la afirmación que es falsa, pues no corresponde a lo que has visto hasta ahora. ¡Adelante!



5.3. Enviar una petición GET

Se definen en el archivo http.service.ts

Ejemplo:

getDatos(){

return **this.http.get**('https://ejemplo-http.firebaseio.com/.json')

.map((response: Response)=> response.json())

}

En el archivo data.service.ts se obtienen los valores llamando a la función getDatos()

getUsers(){

this.httpService.getDatos()

.subscribe(

(data: Response) => console.log(data)

)

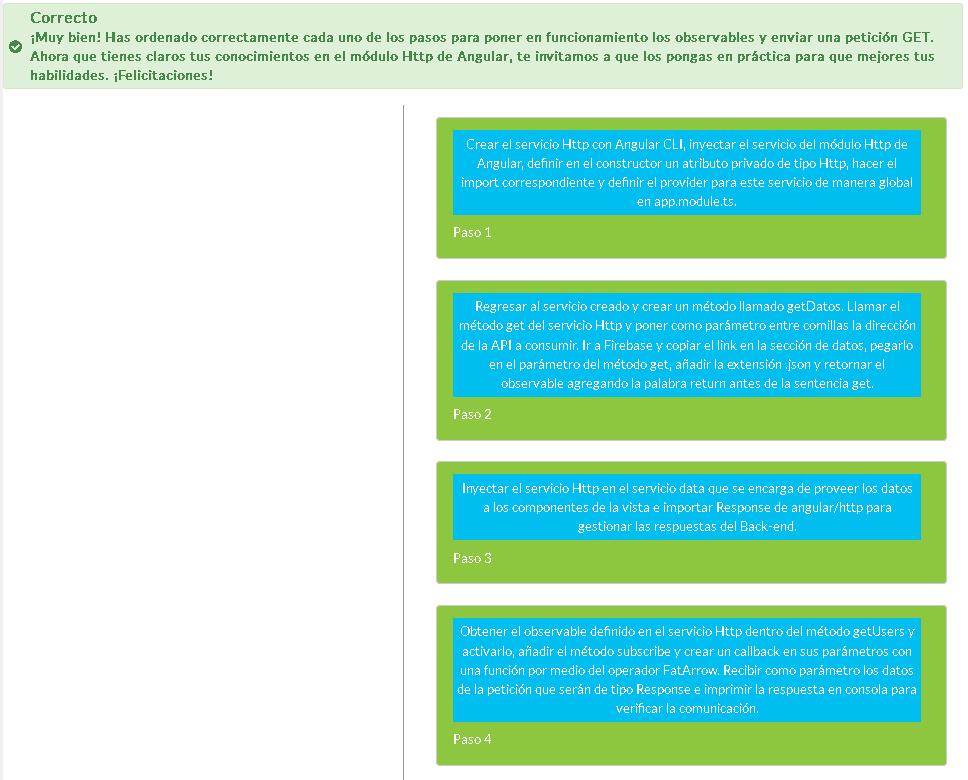
return this.usuarios;

}

5.4. Actividad 2

Tiempo de pensar

Ahora que sabes cómo crear observables y usarlos en un proyecto de Angular2, te invitamos a reforzar tus conocimientos por medio de la siguiente actividad.  
  
Instrucciones:  
Lee el enunciado con atención y realiza la asociación que consideres correcta.  
  
Enunciado:  
En esta oportunidad podrás poner en práctica tus conocimientos en el módulo Http de Angular para ver en funcionamiento los observables y enviar una petición GET. A continuación encontrarás los pasos para realizar este proceso, tu misión será ordenarlos de acuerdo a lo que has visto hasta ahora. ¡Adelante!



5.5. Enviar una petición POST

Se definen en el archivo http.service.ts

sendDatos(usuario : any){

const datos = JSON.stringify(usuario);

return **this.http.post**('https://ejemplo-http.firebaseio.com/.json', datos)

.map((response: Response)=> response.json())

}

En el archivo data.service.ts se envían los valores llamando a la función newUser (usuario)

newUser(nombre: string){

this.usuarios.push(nombre);

this.logService.escribirLog('Se añadio el usuario '+ nombre);

this.httpService.sendDatos({nombre: nombre})

.subscribe(

(data: Response) => console.log(data)

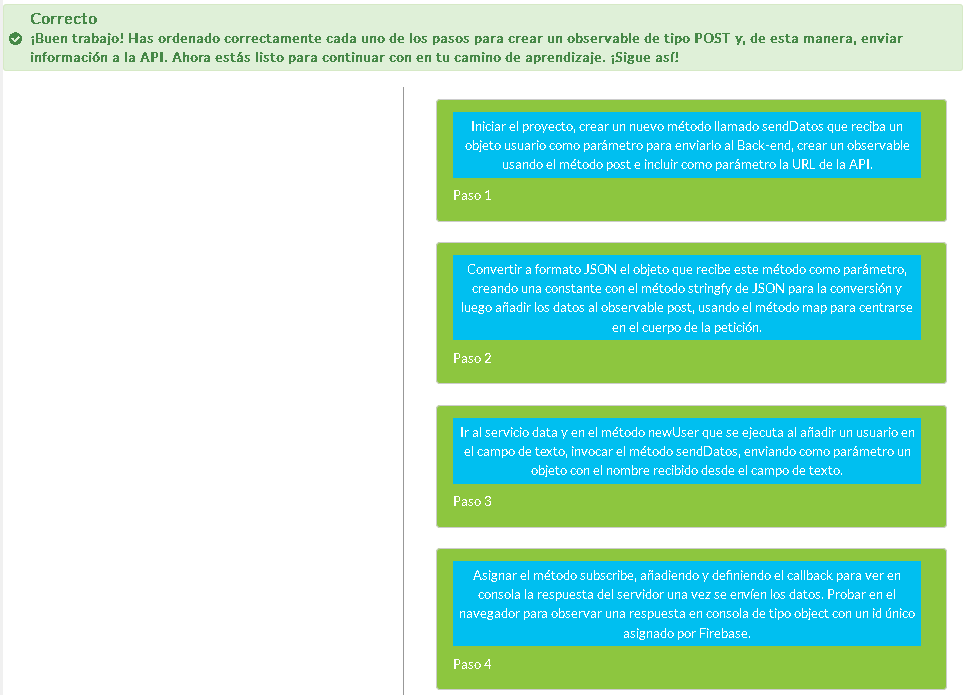
)

}

5.6. Actividad 3

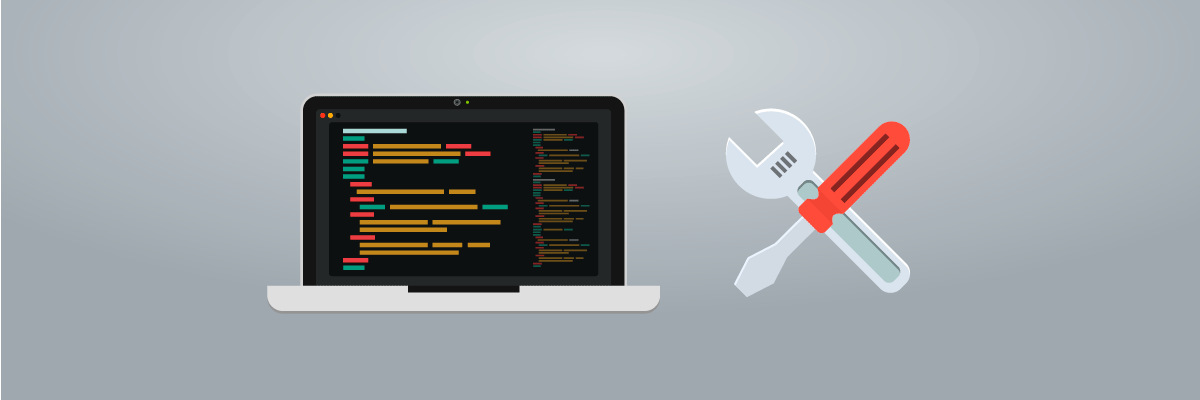
Tiempo de pensar

En esta oportunidad viste cómo realizar peticiones POST, ahora te invitamos a poner a prueba tus conocimientos con de la siguiente actividad.  
  
Instrucciones:  
Lee el enunciado con atención y realiza la asociación que consideres correcta.  
  
Enunciado:  
Ya que sabes cómo crear un observable de tipo POST para enviar información a la API, te invitamos a que ordenes correctamente los pasos para realizar este proceso que encontrarás a continuación. ¡Adelante!



5.7. Lectura: Recepción de datos con observables

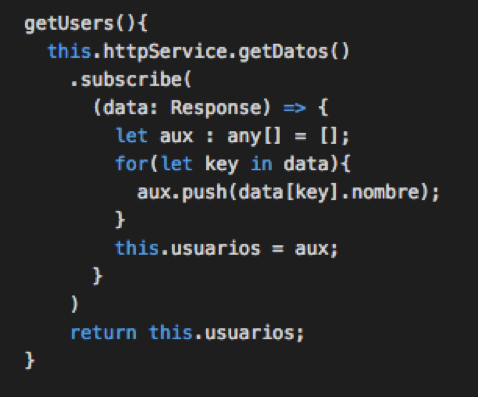
Recepción de datos con observables



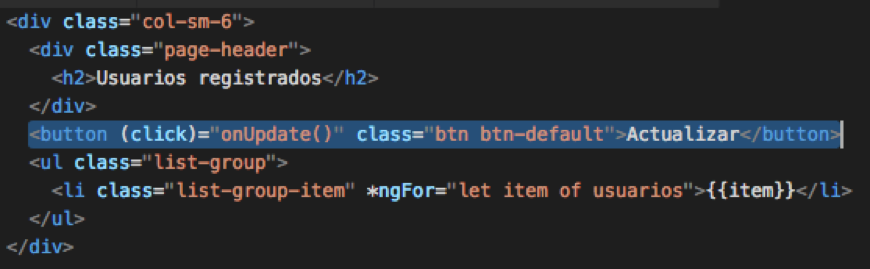
En el proyecto que construimos para las peticiones GET y POST, pudiste observar que las respuestas del Back-end se mostraban en la consola. Por ejemplo, en el método getUsers() que creamos para obtener los usuarios registrados en la base de datos, sólo imprimimos en consola la respuesta de tipo Response:



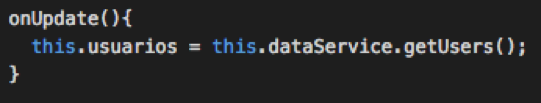
Sin embargo, para mostrar los resultados en la vista, es preciso realizar un proceso de conversión, ya que obtenemos un objeto por parte de la API y debemos convertirlo en un arreglo para poder iterar sobre sus elementos y utilizar ngFor para listarlos en la vista.   
  
Una de las maneras más fáciles de hacerlo es usar un ciclo for y un arreglo temporal donde se vayan almacenando cada uno de los elementos del objeto JSON que retorna la API en la propiedad nombre y luego hacer que el arreglo de usuarios, por ejemplo, sea igual a este arreglo auxiliar:



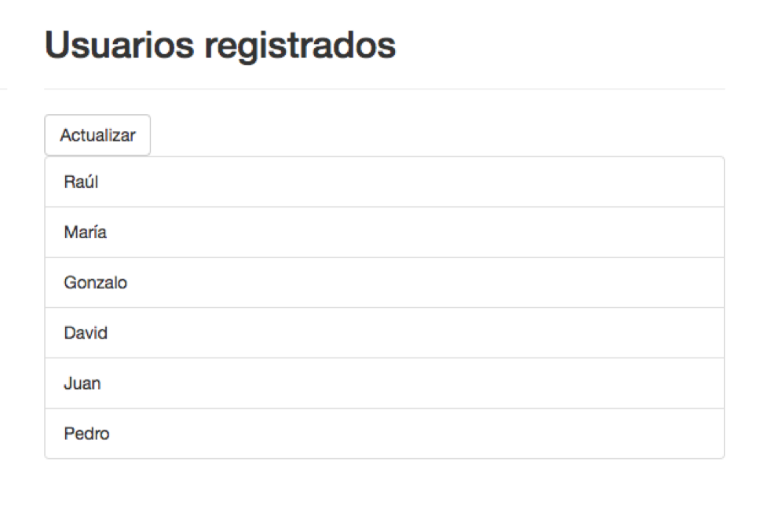
Con este método hemos creado un arreglo de strings con todos los nombres almacenados. Sin embargo, por la naturaleza asincrónica de las peticiones HTTP, es preciso crear un botón en el componente **show-users** para actualizar la vista. Además, le asignamos por event binding un evento clic que dispare el método **onUpdate().**



Por último, en la clase correspondiente a este componente, implementamos el método **onUpdate** para que se ejecute el método **getUsers** del servicio Data. De esta manera, obtendremos el resultado del observable, es decir el arreglo de nombres, y los pasaremos a la vista.



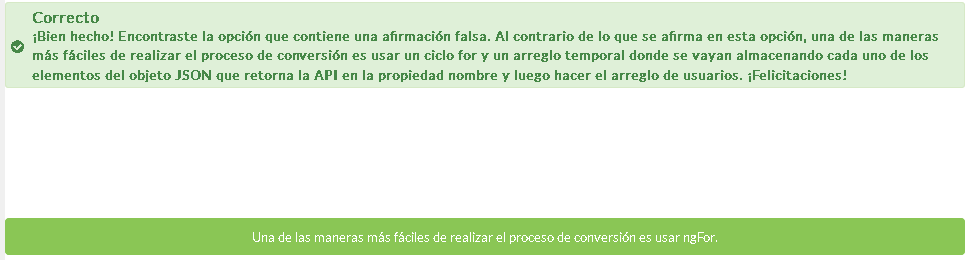
El resultado en el navegador, al hacer clic en el botón actualizar, es el siguiente:



5.8. Actividad 4

Tiempo de pensar

En esta oportunidad pudiste ver cómo realizar la recepción de datos con observables, ahora te invitamos a realizar la siguiente actividad para que refuerces tus conocimientos.  
  
Instrucciones:  
Lee el enunciado con atención y elige la opción que consideres correcta.  
  
Enunciado:  
A continuación te presentamos algunas afirmaciones sobre la recepción de datos con observables, tu misión será encontrar la afirmación que es falsa, pues no corresponde a lo que has visto hasta ahora. ¡Adelante!



5.9. Laboratorio

LABORATORIO DE  
**Instalación y Configuración**



60 MINUTOS

Angular 2 es un framework que provee herramientas de gran utilidad en los diferentes procesos de desarrollo en los que se pueden encontrar tareas difíciles o dispendiosas. Una de las más importantes es la comunicación con el Back-end, en donde las peticiones HTTP resultan simples y efectivas a través de observables y suscripciones.   
  
Sin embargo, antes de entrar en profundidad en esta comunicación, debemos preparar una base de datos que oficie como Back-end de la aplicación, a la cual nos podamos conectar y gestionar información.

##### Contenido:

Para poner en práctica nuestros conocimientos en Angular 2 y las peticiones HTTP mediante observables, debemos primero configurar el Back-end. ¿Estás preparado?

##### Instrucciones

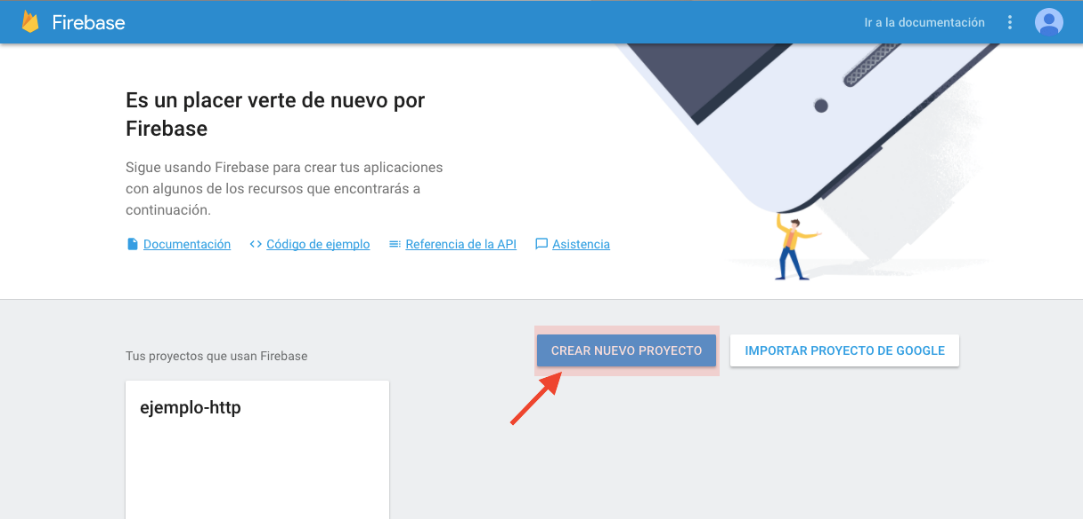
* 1. Ingresa a la página principal de Firebase  
     a. Ingresa a la siguiente URL desde tu navegador preferido



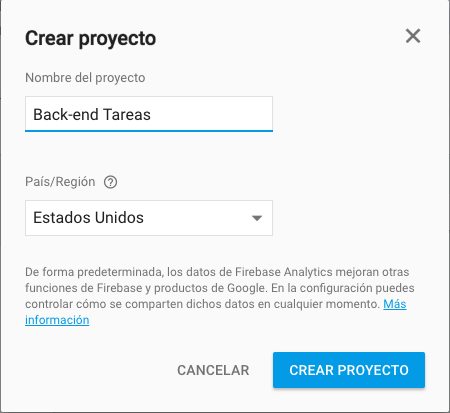
* 1. Haz click en la opción “Ir a la consola” en la parte superior derecha de la página.



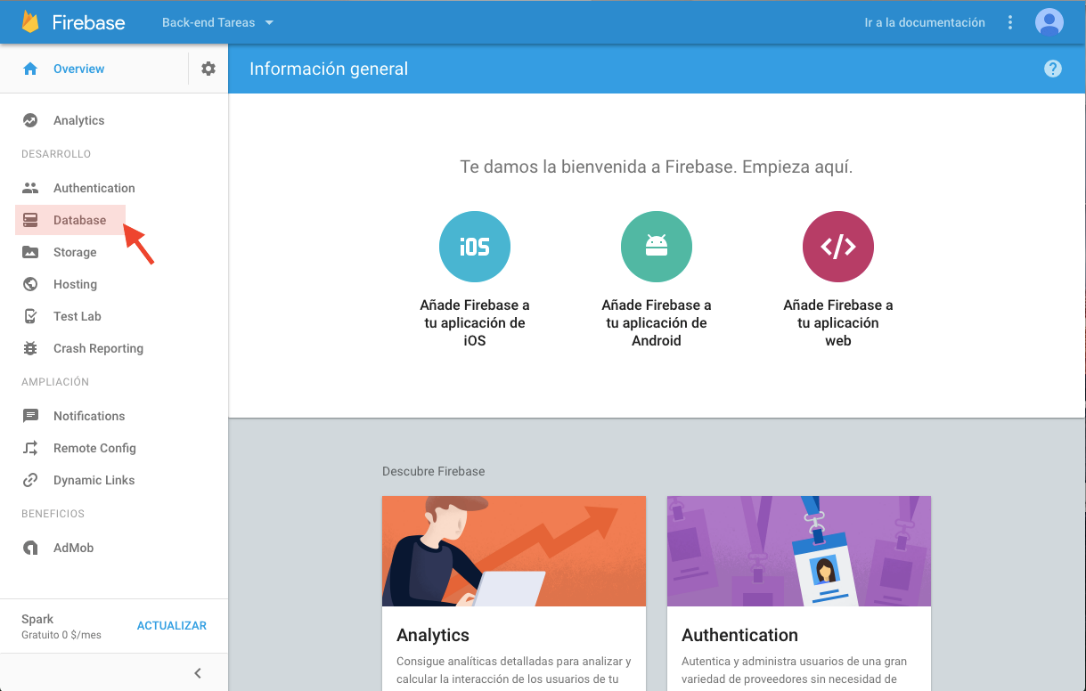
* 1. Ingresa con tu cuenta de Google
  2. Una vez ingresas a la consola con tu usuario, haz click en “Crear nuevo proyecto”



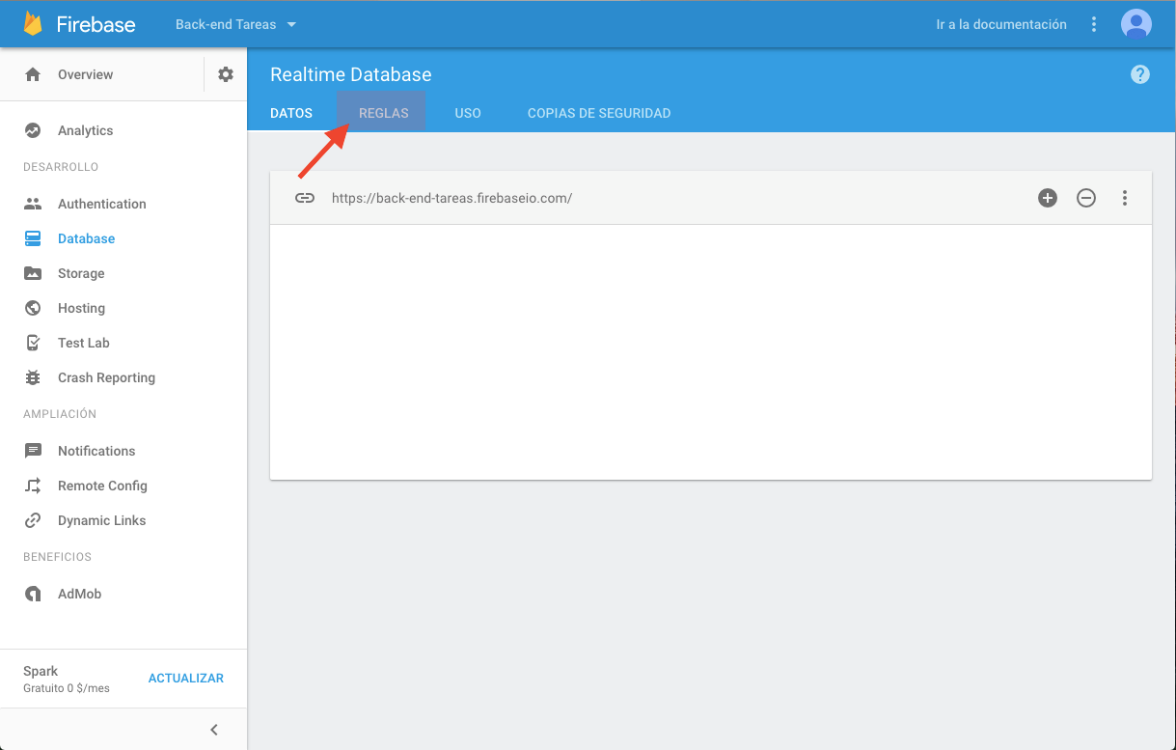
* 1. Dale un nombre al proyecto que será creado y luego indica la región del planeta en la que te encuentras. Por último haz click en “Crear Proyecto”



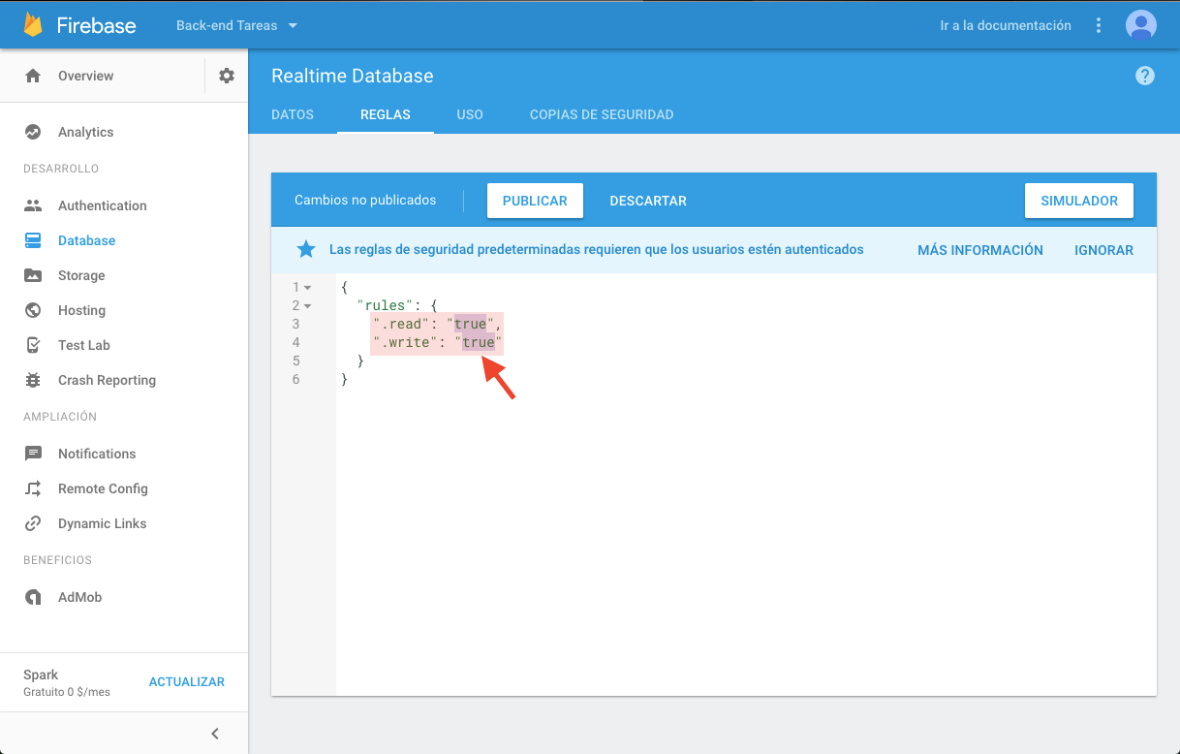
* 1. En el dashboard principal de la aplicación creada, haz click en “Database” para acceder a la sección de base de datos.



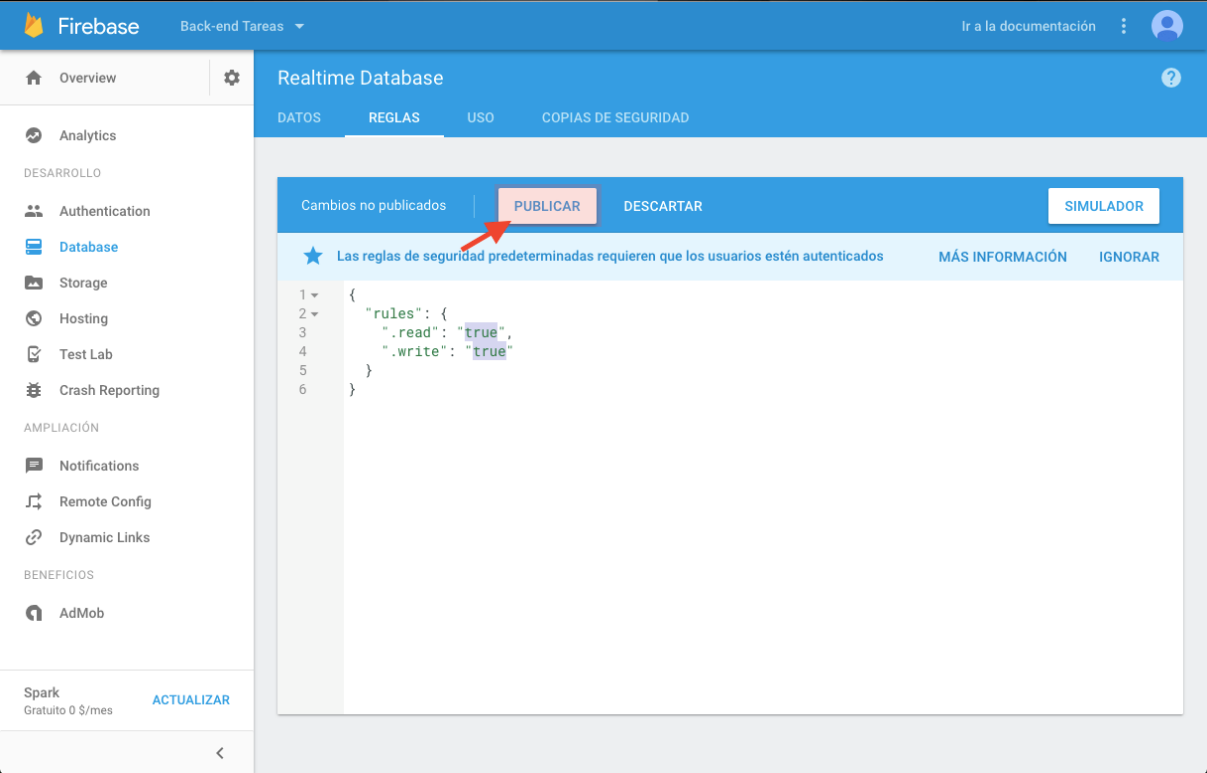
* 1. En la sección de la base de datos, accede a la pestaña “Reglas”, para ingresar al panel de control de seguridad en el acceso a la base de datos.



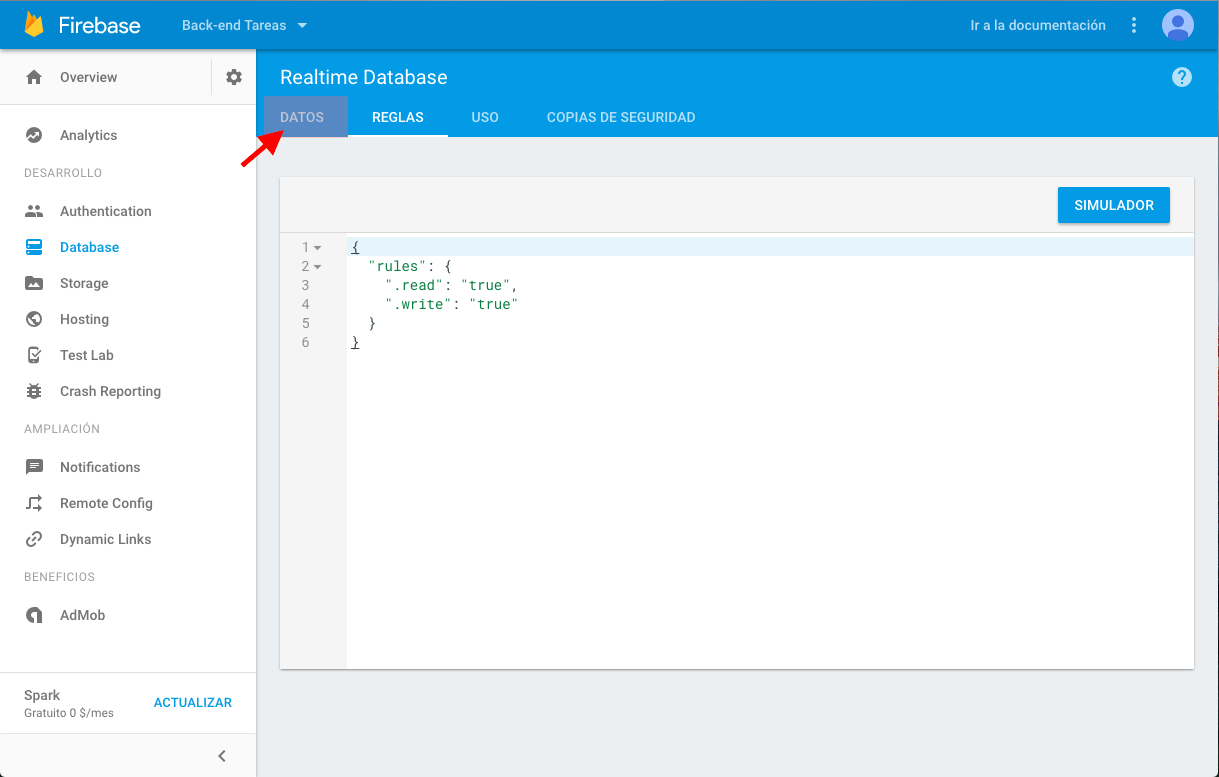
* 1. En esta sección vamos a desactivar la autenticación para el acceso a la base de datos, con el fin de simplificar la comunicación. Para esto cambia los valores de las llaves read y write a true.



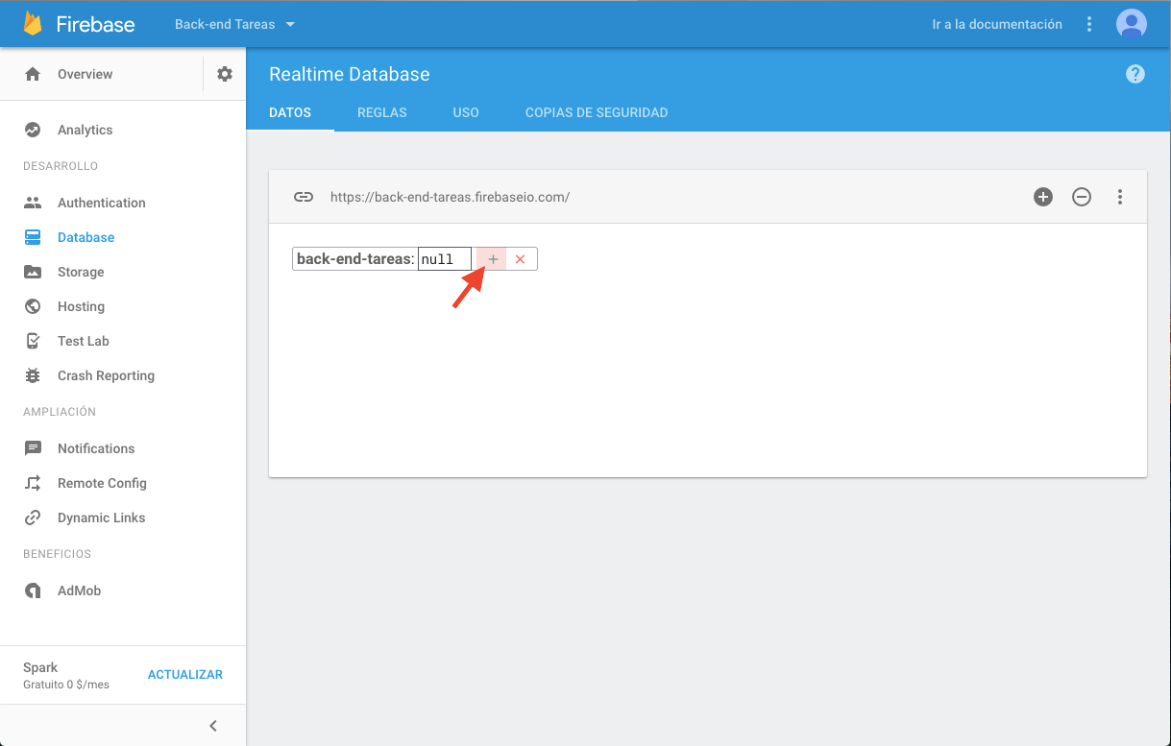
* 1. Para guardar los cambios en estas nuevas configuraciones, haz click en el botón “Publicar”.



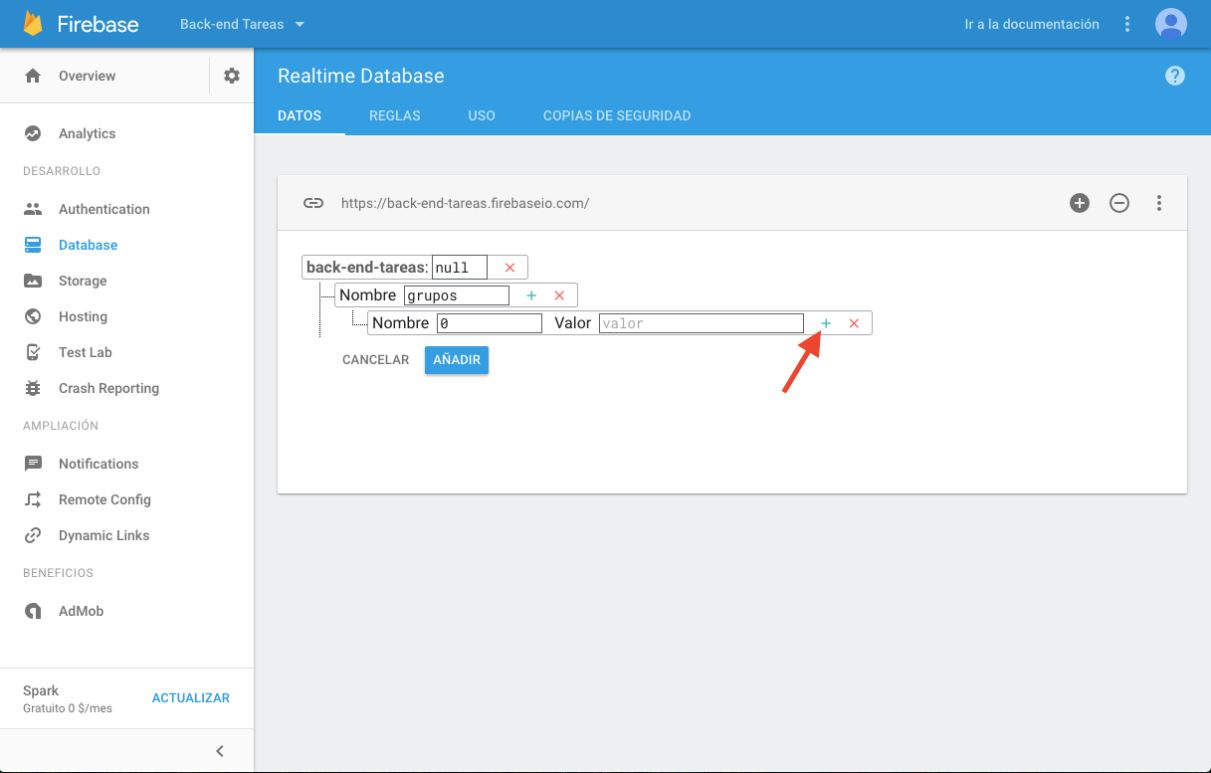
* 1. Regresa a la pestaña “Datos” haciendo click en ella.



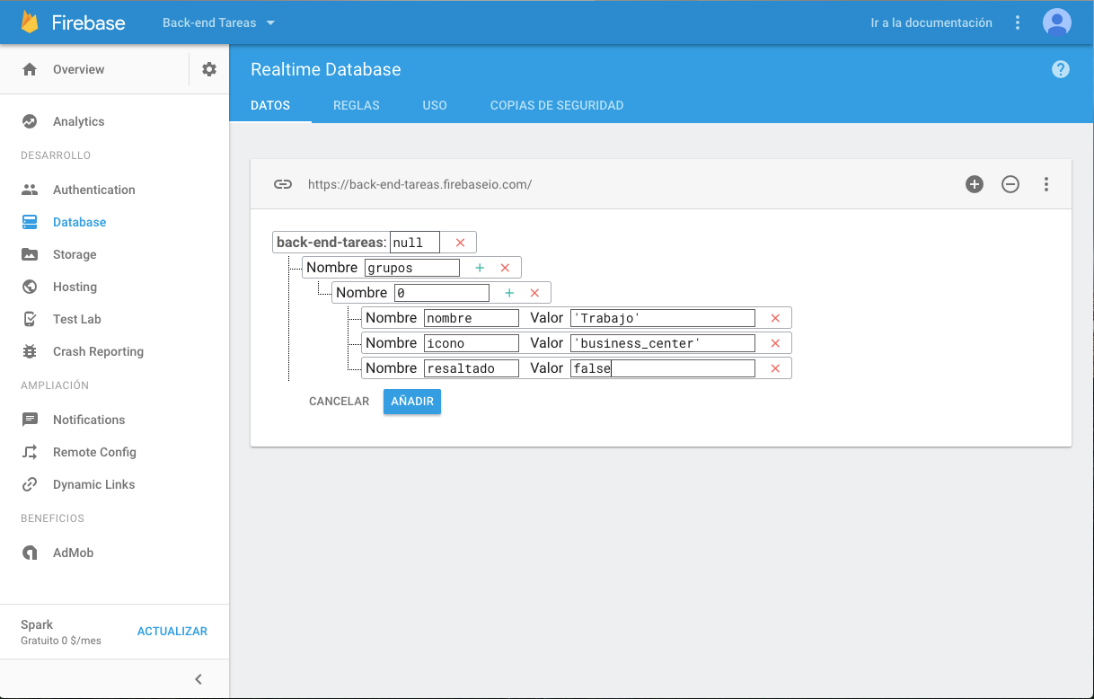
* 1. Ubica el cursor sobre el nombre de la base de datos en el área de trabajo, y haz click en el ícono más (+) que aparece al lado derecho.



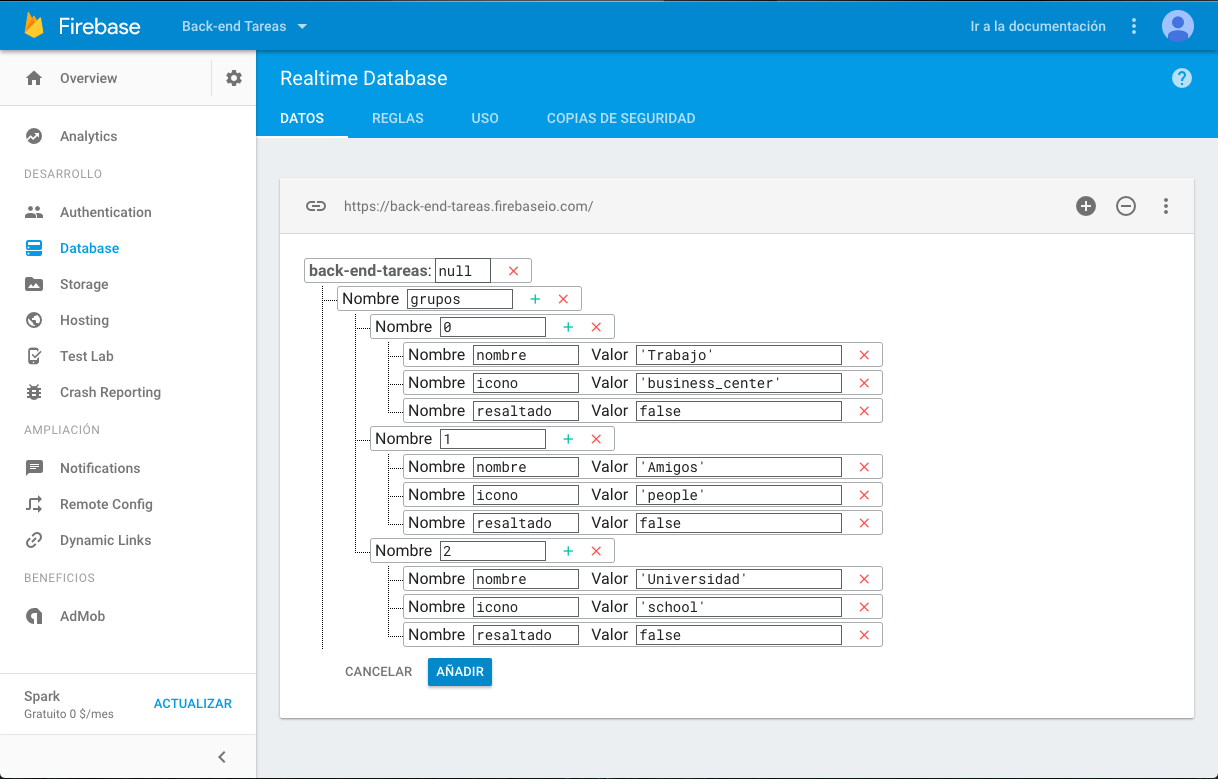
* 1. A continuación crearemos la estructura “llave-valor” de los objetos que contienen la información sobre las tareas y los grupos. Por lo tanto creamos la sección grupos y luego, en la sección de valor, hacemos nuevamente click en el ícono más (+) para crear un nuevo elemento hijo. Como se trata de un arreglo de objetos vamos a darle la clave a este primer grupo con el número 0, luego hacemos click nuevamente en el ícono más (+).



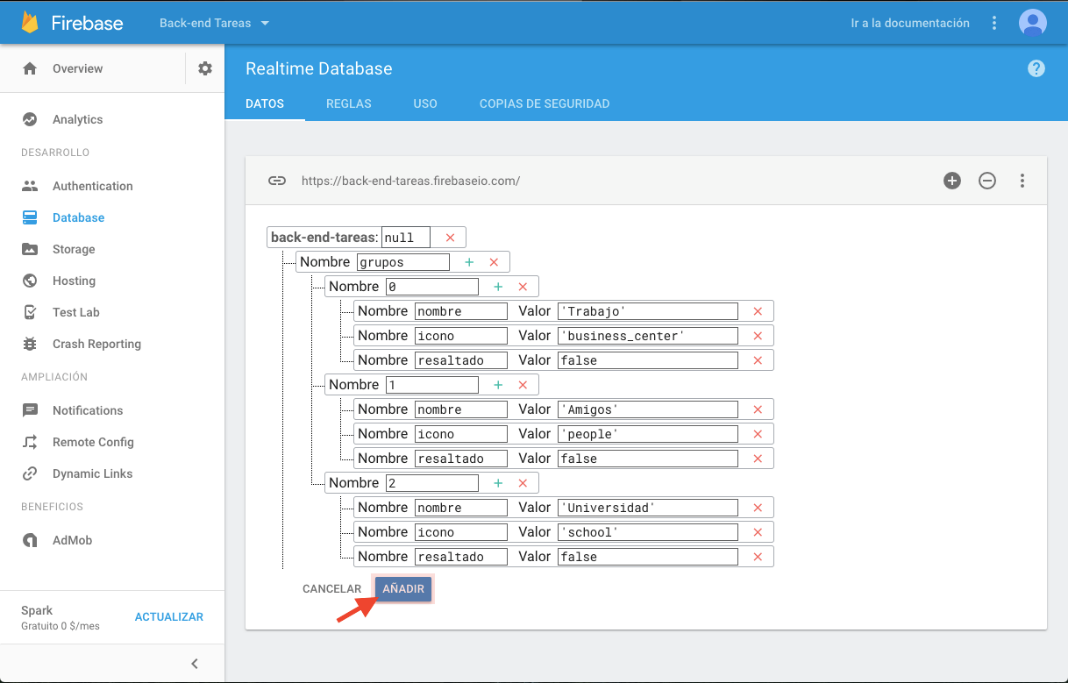
* 1. Creamos la estructura del primer objeto grupo. A los valores de tipo string es importante añadirles las comillas. En esta ocasión debemos usar los mismos campos y tipos de valores que utilizamos para cargar los grupos y tareas en la aplicación que estamos desarrollando en los ejercicios de esta unidad.



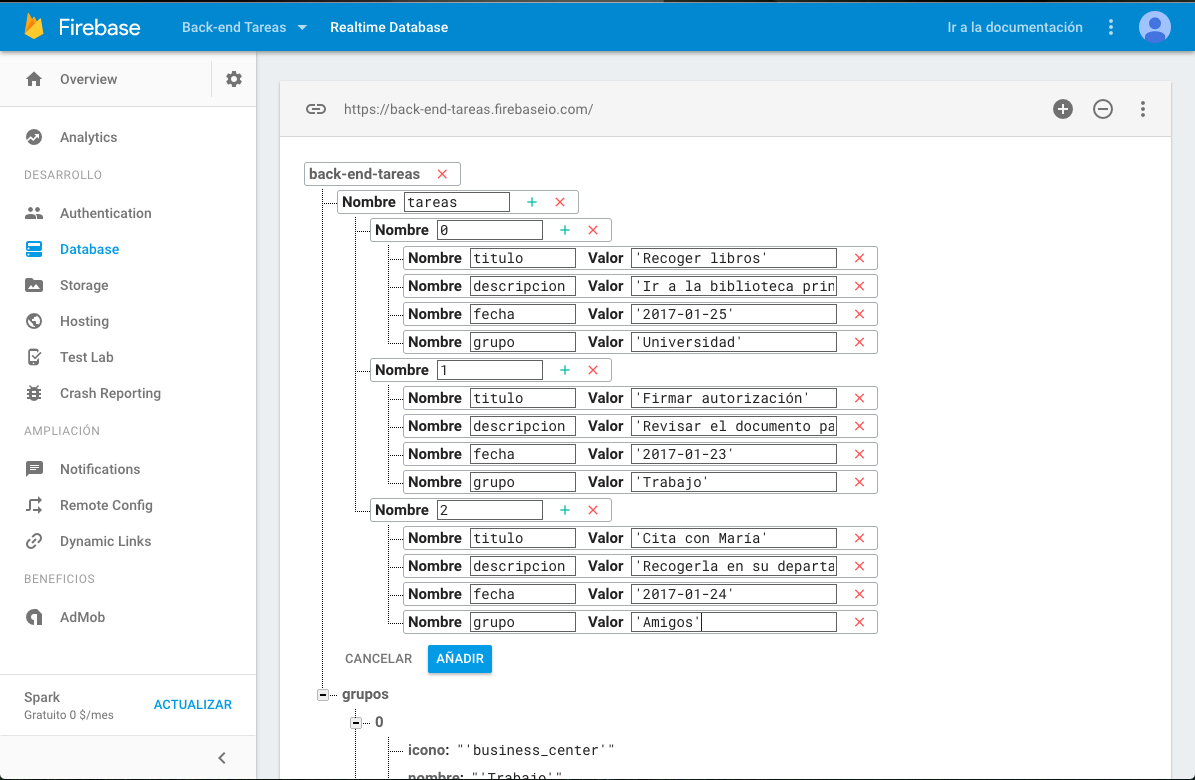
* 1. Regresamos a la parte de grupos y hacemos click en el ícono más (+) para añadir una nueva clave, en este caso será el número 1, y realizamos la misma operación hasta tener tres grupos.



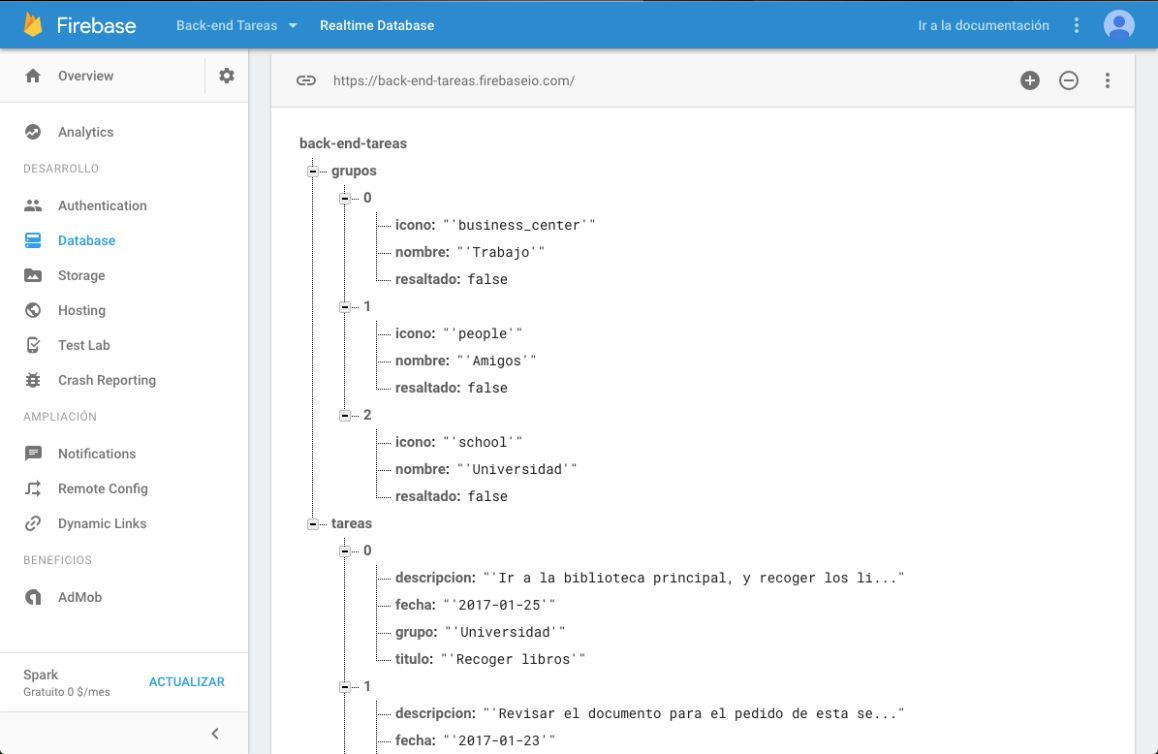
* 1. Para guardar los cambios hacemos click en “Añadir”



* 1. En el nodo raíz añade otra nueva categoría llamada tareas y crea una estructura similar a la anterior, incluyendo en los campos las propiedades adecuadas que respondan al modelo de la aplicación para las tareas.



* 1. Rectifica que la base de datos fue creada correctamente verificando que cada elemento se ubica en su lugar.



Buenas Prácticas

En ambientes de desarrollo o aprendizaje, es común no tener en cuenta las validaciones de ingreso a los datos, por ejemplo, en este laboratorio, desactivamos la autenticación para permitir una fácil comunicación entre la aplicación y el Back-end. Sin embargo, en ambientes de producción es vital que se realicen técnicas de autenticación minuciosas ya que esto es lo que protegerá a los datos allí almacenados, de posibles usuarios no autorizados.

1. Lección 5: Formularios, Animaciones y Deployment

[6.1. Formularios según plantillas](https://learn.nextu.com/mod/lesson/view.php?id=3606&pid=P_WEBDEV_V2)

6.2. Actividad 1

6.3. Formularios según datos

6.4. Actividad 2

6.5. Animaciones

6.6. Actividad 3

6.7. Lectura: Deployment

6.8. Actividad 4

6.9. Ejercicio

1. UNIDAD 3: Prueba