ANGULAR 4

**Angular CLI** es una herramienta de línea de comandos para el desarrollo de aplicaciones Angular

Cada vez que hagamos un proyecto instalaremos los paquetes de Angular a través de npm en nuestro proyecto. Esto nos ahorra instalar el Framework Angular en nuestro equipo (hasta ahora se creaba la carpeta packages en el proyecto y ahí metíamos el framework)

PARA CREAR UN NUEVO PROYECTO

**ng new <nombreNuevo>**

Salta un mensaje “Would you like to add Angular routing? (y/N)”. Yo puse y

También pregunta por el css. Ponemos CSS

Lo creé en la ruta

C:\Users\a630460\OneDrive - Atos\ORANGE\administracion\02\_LEARNING\00\_IDP\05\_ANGULAR\_4\appCurso

Para arrancar el servidor abrimos la consola en esa ruta

**ng serve (la url para verlo es** [**http://localhost:4200/**](http://localhost:4200/)**)**

**ng serve –port 3000 (número de puerto)**

Para parar el servidor sencillamente:

**ctrl+c**

**Un componente** es un conjunto de archivos que nos va a realizar una determinada función en la aplicación

**(app.component.ext)** donde ext (la extensión de los archivos del componente) son

app.component.html (la plantilla que muestra lo que queremos del componente)

app.component.css (el estilo de sólo ese componente)

app.component.ts (aquí irá toda la lógica del componente)

app.component.spec.ts (es sólo para testing.)

En visual Studio code podemos incorporar la terminal de comandos (VER

**Generar componente en consola con Angular CLI**

**ng generate component <nombreNuevo> -–spec false** (PARA QUE NO GENERE EL spec.ts)

**ng g c <nombreNuevo> -–spec false**

1. DATA BINDING
   1. ONE WAY BINDING
      * Fuente de datos - HTML
        + Interpolación Técnica “moustache” {{ }}
        + Property Binding
      * HTML – Fuente de datos
        + Eveng Binding
   2. TWO WAY BINDING
      * + Two way Binding Técnica “banana box” [()] (import en app.modules.ts FormsModule de @angular/forms)
2. DIRECTIVAS

Son clases angular con Código para crear, formatear e interaccionar con los elementos HTML en el DOM de las página webs

Hay tres tipos de directivas

* Componentes (como el decorador @Component)
* Estructurales (alteran el layout – la presentación- del elemento HTML en la que las introducimos)
* De atributos (no alteran el layout sino que funcionan como un atributo HTML parecido a lo que vimos en el property binding)

**ngIf** (estructural)

**ngIf + else** (estructural)

**ngStyle** (atributo)

**ngClass** (atributo)

**ngFor** (estructural)

**ngSwitch** (estructural)

**Crear directivas**

**ng generate directive <nombre>**

**Crear Pipes**

**ng generate pipe <nombre>**

1. SERVICIOS E INYECCIÓN DE DEPENDENCIAS
2. Los servicios son unos elementos de Angular que nos permiten centralizar el uso de código para luego emplearlo en cada componente donde lo necesitemos mediante la inyección de independencias.
3. Como se suelen utilizar para gestionar datos e inyectarlos en los componentes se les conoce como “Providers”. De hecho se implementan en un array con ese mismo nombre en el archivo raíz del módulo.

Crear un servicio

**ng generate service servicios/<nombreServicio> --spec false**

**ng g s servicios/<nombreServicio> --spec false**

ojo genera el servicio pero no lo provee y no lo avisa. Hay que meterlo en el array provide del app.module.ts

1. ROUTING
2. Las aplicaciones de Angular están pensadas para ser single page application (una sóla página). Los distintos componentes se van renderizando sin que la página se refresque (lo que quitaría mucho tiempo) y es mucho más rápida. Tarda unos segundos en cargar la primera vez después va a fuego.

Pero Single Page Application no significa que todos los componentes estén en la misma página (lo haría muy complejo). Lo que se usa es el routing que con la ayuda del navegador establecemos URL’s que cargan dinámicamente cada componente.

1. FORMULARIOS

**Template Driven**

Realiza la captura y validación de datos desde el HTML

**Reactive**

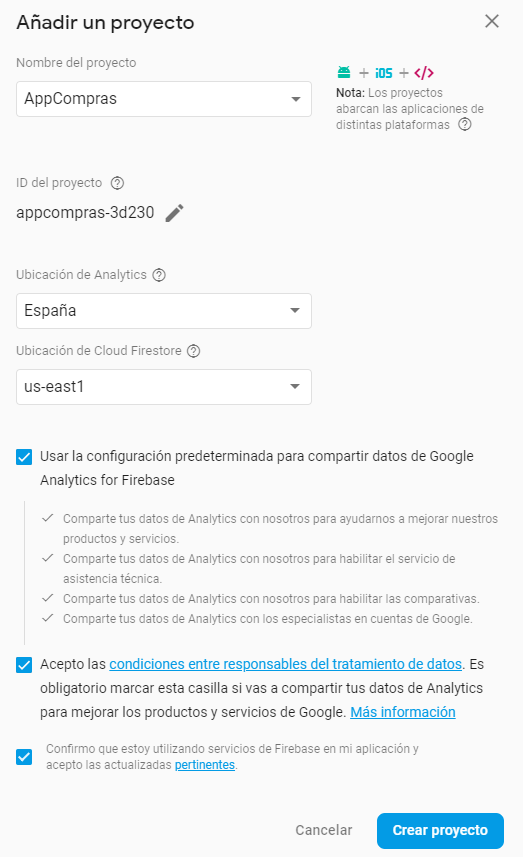
Gestiona el formulario desde el typescript (tiene mucho más control)

**VALIDACION DE CAMPOS**

Estados

* Dirty
* Pristine
* Touched
* Valid
* Invalid

[**https://angular.io/api/forms/Validators**](https://angular.io/api/forms/Validators)

http://firebase.google.com/

<https://console.firebase.google.com/project/appcompras-3d230/database/firestore/data/>

REGLAS

service cloud.firestore {

match /databases/{database}/documents {

match /{document=\*\*} {

allow read, write: if true;

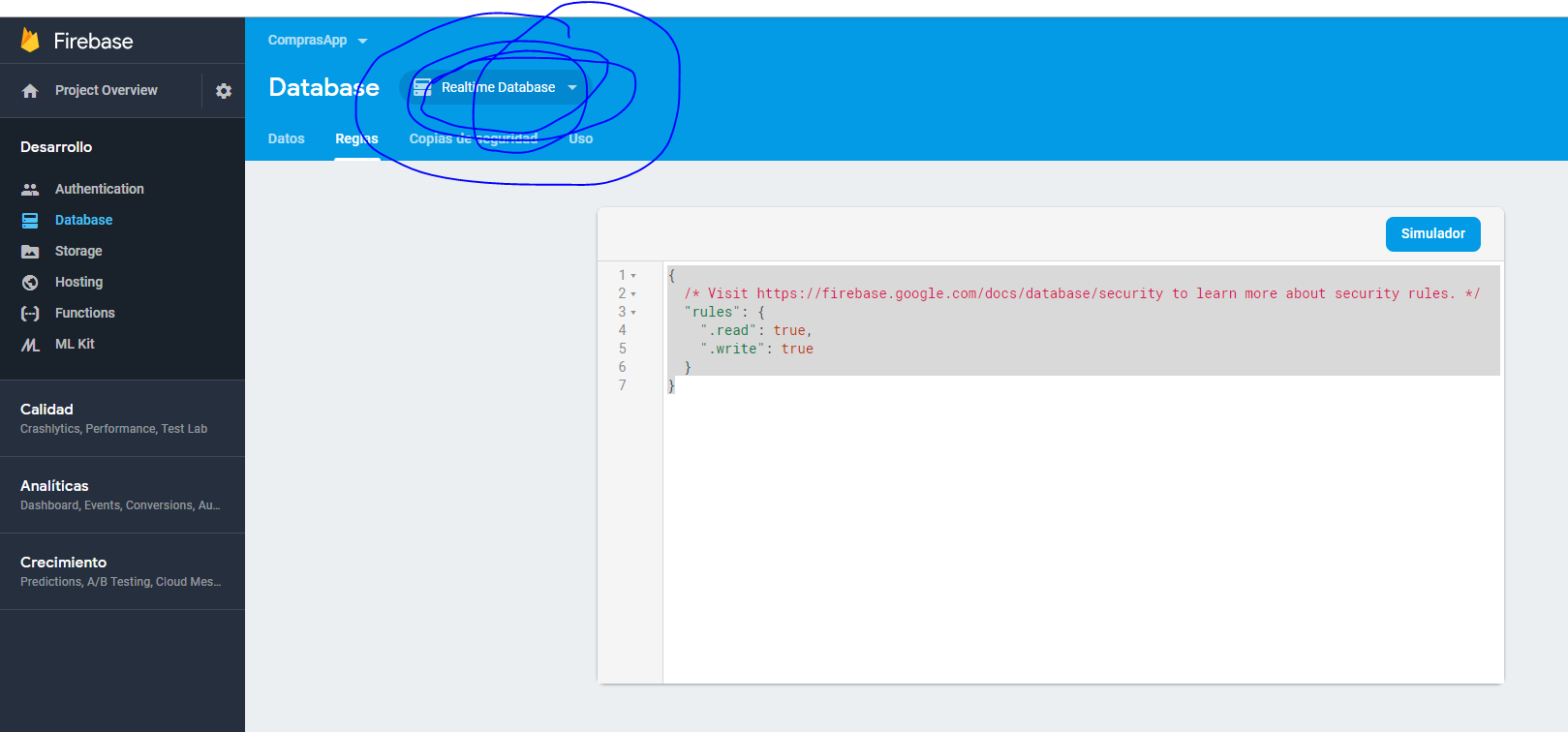
}

}

}

<https://comprasapp-54177.firebaseio.com/>

Al crear la BBDD en Firebase poner REALTIME DATABASE!!! (mira la captura)



Utilizaremos peticiones HTTP en Angular ya que la emplearemos para muchos servidores de bbdd.

(servicios (centralizan código para emplearlo en diferentes componentes) para concetar con la bbdd

error TS2307: Cannot find module 'rxjs-compat'

lo resolví con esto.

npm install --save rxjs-compat

npm install firebase angularfire2 –save

1. AUTENTICACIÓN
2. MODULARIZACIÓN

Todas las aplicaciones de angular deben de tener al menos un módulo con un archivo app.module.ts

Se pueden hacer muchos módulos agrupando los componentes por tipo de funcionalidad.

Los módulos se crean con el angular CLI

Para crear un módulo adicional:

**ng generate module <nombreModulo>**

Para crear un nuevo componente de un módulo adicional:

**ng g c facturas/facturas/<nombreComponente> –spec false** (es decir, basta con indicar la ruta donde va a crearse)

se mete en los imports del app.modules.ts y en el imports: [ ]

IMPORTANTE. EN CADA MÓDULO NUEVO COPIAR Y PEGAR LOS IMPORTS DE PAQUETES Y LIBRERÍAS DE ANGULAR DESDE EL app.module.ts Y PEGARLOS PARA QUE PUEDAN UTILIZARLOS

TÉCNICAS DE PROYECTOS

BUSCADOR DINÁMICO

DESPLIEGUE

ng build –prod

arn:aws:s3:::appcomprascap

 http://appcomprascap.s3-website.eu-west-3.amazonaws.com