# Typescript

* Typescript es un superset de Javascript, por lo tanto está escrito en Javascript.
* Es un agregado a JavaScript, no lo reemplaza. Algunas tareas las tendremos que hacer en Javascript.
* Agrega nuevas características:
  + Tipos
  + Clases
  + Interfaces
* Es transpilado a Javascript.
* Es el lenguaje con el que se programa para Angular4.

# Herramientas necesarias para utilizar Angular4

* Node.js
* Angular CLI

# Angular CLI

Es un set de herramientas de consola que sirve para crear proyectos y generar componentes, rutas, servicios, pipes, etc.

Permite correr nuestra app de forma local, hacer testing y preparar una app para producción.

<https://cli.angular.io/>

Para instalar Angular CLI se corre el siguiente comando:

**npm install -g @angular/cli**

# Crear un nuevo proyecto

**ng new <nombre de proyecto>**

Este comando crea la carpeta del proyecto y todas las subcarpetas y archivos necesarios.

# Correr proyecto

**ng serve**

Este comando (dentro de la carpeta de nuestro proyecto) transpila el código y corre un servidor en local.

# Módulos

Es la forma como se debe dividir un proyecto. Se declaran en app.module.ts, en la sección **imports** dentro de **@NgModule**.

# Componentes

Cada vista de la app deberá ser un componente distinto. Se declaran en app.module.ts, en la sección **declarations** dentro de **@NgModule**.

Cada componente deberá tener un **component.ts** en donde indica la información de nuestro componente. Esto se hace dentro del decorador **@Component**. Deberá tener obligatorio un **selector** y un **templateUrl,** opcional podemos tener un archivo de estilos en **styleUrls**. Para que se pueda utilizar ese componente se deberá exportar. Cuando se crea el proyecto con **ng new <nombre de proyecto>** se tiene por default un componente llamado app.component.ts.

Para crear nuestros propios componentes se corre el comando **ng g c <nombre del componente>**.

# Data Binding

Comunicación entre el código que escribimos y lo que ve finalmente el cliente.

## Tipos

### String Interpolation {{}}

Tener TypeScript y presentar al usuario HTML.

**Ejemplo:**

**app.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'JazzoSquare';

a = 3;

b = 5;

}

**app.component.html**

<div style="text-align:center">

<h1>

Welcome to {{title}}!

</h1>

<h2>

{{a+b}}

</h2>

</div>

### Property Binding []

Tener información de HTML y que viaje a TypeScript.

**Ejemplo:**

**app.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'JazzoSquare';

listo = false;

constructor() {

setTimeout(()=>{this.listo = true;}, 3000);

}

}

**app.component.html**

<div style="text-align:center">

<h1>

Welcome to {{title}}!

</h1>

<hr/>

<button [disabled]="!listo">Click me!</button>

</div>

### Event Binding ()

Tener evento en HTML y que viaje a TypeScript.

**Ejemplo:**

**app.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'JazzoSquare';

hacerAlgo() {

alert('Haciendo algo');

}

}

**app.component.html**

<div style="text-align:center">

<h1>

Welcome to {{title}}!

</h1>

<hr/>

<button (click)="hacerAlgo()">Click me!</button>

</div>

### Two Way Data Binding [()]

Comunicación de 2 vías.

**Ejemplo:**

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

import { FormsModule } from '@angular/forms';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

FormsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**app.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'JazzoSquare';

nombre:string = '';

}

**app.component.html**

<div style="text-align:center">

<h1>

Welcome to {{title}}!

</h1>

<br/>

<input type="text" placeholder="Tu nombre" [(ngModel)]="nombre"/>

<br/>

Mi nombre es {{nombre}}

</div>

# Directivas

Forma elegante y rápida de manipular la información.

## Tipos

### Componentes

Directivas que siempre tienen asignado un template de HTML.

### Estructurales

Directivas que permite modificar el DOM.

#### \*ngFor

**Ejemplo:**

**app.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'JazzoSquare';

lugares:any = [

{nombre: 'Florería La Gardenia'},

{nombre: 'Donas La Pasadita'},

{nombre: 'Veterinaria Huellitas Felices'},

];

}

**app.component.html**

<ul>

<li \*ngFor="let lugar of lugares">{{lugar.nombre}}</li>

</ul>

#### \*ngIf

**Ejemplo:**

**app.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'JazzoSquare';

lugares:any = [

{active: true, nombre: 'Florería La Gardenia'},

{active: false, nombre: 'Donas La Pasadita'},

{active: true, nombre: 'Veterinaria Huellitas Felices'}

];

}

**app.component.html**

<ul>

<!--

No puede haber más de una directiva en un elemento,

pero se puede utilizar otra etiqueta llamada <ng-container>

para utilizar el ngFor.

-->

<ng-container \*ngFor="let lugar of lugares">

<li \*ngIf="lugar.active">{{lugar.nombre}}</li>

</ng-container>

</ul>

### Atributos

Directivas implementadas a través de atributos que pueden modificar el elemento al que están asignadas.

#### [ngStyle]

**Ejemplo:**

**app.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'JazzoSquare';

lugares:any = [

{active: true, nombre: 'Florería La Gardenia'},

{active: false, nombre: 'Donas La Pasadita'},

{active: true, nombre: 'Veterinaria Huellitas Felices'}

];

}

**app.component.html**

<ul>

<ng-container \*ngFor="let lugar of lugares">

<li [ngStyle]="{color:(lugar.active) ? 'black' : 'gray'}">{{lugar.nombre}}</li>

</ng-container>

</ul>

#### [ngClass]

**Ejemplo:**

**app.component.css**

.numero\_par{

background-color: aliceblue;

color: blue;

}

**app.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'JazzoSquare';

lugares:any = [

{nombre: 'Florería La Gardenia'},

{nombre: 'Donas La Pasadita'},

{nombre: 'Veterinaria Huellitas Felices'}

{nombre: 'Florería La Gardenia 2'},

{nombre: 'Donas La Pasadita 2'},

];

}

**app.component.html**

<ul>

<ng-container \*ngFor="let lugar of lugares; let i = index">

<li [ngClass]="{numero\_par: (i+1) % 2 !== 0}">{{lugar.nombre}}</li>

</ng-container>

</ul>

#### [ngSwitch]

**Ejemplo:**

**app.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'JazzoSquare';

lugares:any = [

{cercania: 1, distancia: 1, active: true, nombre: 'Florería La Gardenia'},

{cercania: 1, distancia: 1.8, active: false, nombre: 'Donas La Pasadita'},

{cercania: 2, distancia: 5, active: true, nombre: 'Veterinaria Huellitas Felices'},

{cercania: 3, distancia: 10, active: true, nombre: 'Florería La Gardenia 2'},

{cercania: 3, distancia: 35, active: true, nombre: 'Donas La Pasadita 2'},

{cercania: 3, distancia: 120, active: false, nombre: 'Veterinaria Huellitas Felices 2'}

];

constructor() { }

}

**app.component.html**

<ul>

<ng-container \*ngFor="let lugar of lugares; let i = index" [ngSwitch]="lugar.cercania">

<li [ngStyle]="{color:(lugar.active) ? 'black' : 'gray'}" [ngClass]="{numero\_par: (i+1) % 2 !== 0}">{{lugar.nombre}} -

<span \*ngSwitchCase="1" [ngStyle]="{color:'green'}">Muy cercano</span>

<span \*ngSwitchCase="2" [ngStyle]="{color:'black'}">Cercano</span>

<span \*ngSwitchCase="3" [ngStyle]="{color:'red'}">Lejano</span>

</li>

</ng-container>

</ul>

#### [innerHtml]

El texto que pongamos con esta directiva, el navegador lo interpretará como HTML.

**Ejemplo:**

<p [innerHtml]=”str\_con\_ancors”></p>

## Crear nuestras propias directivas

1. Crear carpeta directives dentro de app.
2. Crear archivo <nombre-de-directiva>.directive.ts

import {Directive, OnInit, ElementRef, Renderer2, Input} from "@angular/core";

@Directive({

selector: '[resaltar]'

})

export class ResaltarDirective implements OnInit {

constructor(private elRef: ElementRef, private renderer: Renderer2) {}

@Input('resaltar') plan:string = '';

ngOnInit() {

if(this.plan === 'pagado') {

this.renderer.setStyle(this.elRef.nativeElement, 'background-color', 'yellow');

this.renderer.setStyle(this.elRef.nativeElement, 'font-weight', 'bold');

}

}

}

1. Agregarlo a nuestro app.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

import { FormsModule } from '@angular/forms';

import { AgmCoreModule } from '@agm/core';

import { ResaltarDirective } from './directives/resaltar.directive';

@NgModule({

declarations: [

AppComponent,

ResaltarDirective

],

imports: [

BrowserModule,

FormsModule,

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

1. Implementarla en app.component.html

<ul>

<ng-container \*ngFor="let lugar of lugares; let i = index" [ngSwitch]="lugar.cercania">

<li [ngClass]="{numero\_par: (i+1) % 2 !== 0}" [resaltar]="lugar.plan">{{lugar.nombre}} -

<span \*ngSwitchCase="1" [ngStyle]="{color:'green'}">Muy cercano</span>

<span \*ngSwitchCase="2" [ngStyle]="{color:'black'}">Cercano</span>

<span \*ngSwitchCase="3" [ngStyle]="{color:'red'}">Lejano</span>

</li>

</ng-container>

</ul>

# Instalar librerías externas con NPM

## Google Maps

1. Ir a <https://angular-maps.com/>
2. Getting started.
3. Seguir instrucciones de Angular Google Maps modificando app.mudule.ts, app.component.ts, app.component.html y app.component.css.

### Geolocalización

Angular contiene una librería llamada http que nos sirve para llamar API de un proveedor externo como obtener la geolocalización de una dirección con google maps (**http://maps.google.com/maps/api/geocode/json?address=mi calle numero,ciudad,estado,pais**).

Para usarlo tenemos que importar **Http** de **@angular/http,** declararlo en el constructor como variable privada y agregar el módulo **HttpModule** a nuestro **app.module.ts** en la sección de **imports**.

**http.get(‘http://maps.google.com/maps/api/geocode/json?address=mi calle numero,ciudad,estado,pais’)** retorna una promesa, por lo que necesitaremos subscribirnos a esa promesa para poder utilizarla.

**Ejemplo:**

this.http.get(http://...)

.subscribe((result) => {

this.lat = result.json().results[0].geometry.location.lat;

this.lng = result.json().results[0].geometry.location.lng;

});

Para mostrarlas en un mapa basta con modificar el código de nuestro mapa:

**Ejemplo:**

<agm-map [latitude]="lat" [longitude]="lng">

<agm-marker \*ngfor=”let lugar of lugares” [latitude]="lugar.lat" [longitude]="lugar.lng"></agm-marker>

</agm-map>

# Host Listeners

Ayuda para que cuando se crea una directiva a algún elemento HTML se puedan escuchar los eventos de una manera muy sencilla.

Ejemplo:

**contar-clicks.directive.ts**

import {Directive, HostListener} from "@angular/core";

@Directive({

selector: 'a[contar-clicks]'

})

export class ContarClicksDirective {

clickN = 0;

@HostListener('click', ['$event.target']) onClick(btn) {

console.log('a', btn, "Número de clicks: ", this.clickN++);

}

}

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

import { FormsModule } from '@angular/forms';

import { ContarClicksDirective } from './directives/contar-clicks.directive';

@NgModule({

declarations: [

AppComponent,

ContarClicksDirective

],

imports: [

BrowserModule,

FormsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**app.component.html**

<ul>

<ng-container \*ngFor="let lugar of lugares; let i = index" [ngSwitch]="lugar.cercania">

<a contar-clicks>

<li [ngStyle]="{color:(lugar.active) ? 'black' : 'gray'}" [ngClass]="{numero\_par: (i+1) % 2 !== 0}">{{lugar.nombre}} -

<span \*ngSwitchCase="1" [ngStyle]="{color:'green'}">Muy cercano</span>

<span \*ngSwitchCase="2" [ngStyle]="{color:'black'}">Cercano</span>

<span \*ngSwitchCase="3" [ngStyle]="{color:'red'}">Lejano</span>

</li>

</a>

</ng-container>

</ul>

# Host Binding

Permite modificar el elemento del DOM desde la propia directiva.

Ejemplo:

**contar-clicks.directive.ts**

import {Directive, HostListener, HostBinding} from "@angular/core";

@Directive({

selector: 'li[contar-clicks]'

})

export class ContarClicksDirective {

clickN = 0;

@HostBinding('style.opacity') opacity: number = .1;

@HostListener('click', ['$event.target']) onClick(btn) {

console.log('a', btn, "Número de clicks: ", this.clickN++);

this.opacity += .1;

}

}

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

import { FormsModule } from '@angular/forms';

import { ContarClicksDirective } from './directives/contar-clicks.directive';

@NgModule({

declarations: [

AppComponent,

ContarClicksDirective

],

imports: [

BrowserModule,

FormsModule,

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**app.component.html**

<ul>

<ng-container \*ngFor="let lugar of lugares; let i = index" [ngSwitch]="lugar.cercania">

<a>

<li contar-clicks [ngStyle]="{color:(lugar.active) ? 'black' : 'gray'}" [ngClass]="{numero\_par: (i+1) % 2 !== 0}" [resaltar]="lugar.plan">{{lugar.nombre}} -

<span \*ngSwitchCase="1" [ngStyle]="{color:'green'}">Muy cercano</span>

<span \*ngSwitchCase="2" [ngStyle]="{color:'black'}">Cercano</span>

<span \*ngSwitchCase="3" [ngStyle]="{color:'red'}">Lejano</span>

</li>

</a>

</ng-container>

</ul>

# Bootstrap

<https://getbootstrap.com>

## Instalar

npm install --save bootstrap (Correr comando dentro de la carpeta de nuestro proyecto).

## Configurar

1. Modificar **.angular-cli.json** agregando **bootstrap.min.css** en la sección **styles**.

"styles": [

"../node\_modules/bootstrap/dist/css/bootstrap.min.css",

"styles.css"

],

1. Reiniciar la ejecución del proyecto.
2. Usar componentes (ver ejemplos en página oficial).

# Sweetalert2

<https://www.npmjs.com/package/sweetalert2>

## Instalar

npm install --save sweetalert2 @toverux/ngsweetalert2 (Correr comando dentro de la carpeta de nuestro proyecto).

## Uso

### Typescript

**Ejemplo:**

**Función en componente.ts**

deleteCategory(id) {

let instance = this;

swal({

title: '¿Estás seguro?',

text: "Se borrará definitivamente la categoría seleccionada",

type: 'warning',

showCancelButton: true,

confirmButtonColor: '#3085d6',

cancelButtonColor: '#d33',

confirmButtonText: '¡Si, borrar categoría!'

}).then(function () {

instance.categoryService.deleteCategory(id);

instance.cleanFields();

swal(

'Borrada!',

'La categoría ha sido borrada.',

'success'

)

}, function (dismiss) {

// dismiss can be 'cancel', 'overlay', 'close', and 'timer'

if (dismiss === 'cancel') {

swal(

'Calcelada',

'La operación ha sido cancelada',

'error'

)

}

})

}

}

### Template

**Ejemplo:**

**Sección en template.html**

# Router

Permite implementar navegación en la app. Se pueden asignar vistas y/o componentes específicos para cada URL que deseemos y puede recibir parámetros.

## Configuración

1. Dejar solo el header, router-outlet (donde se verán los componentes ruteados) y footer en **app.component.html** y limpiar el **app.component.ts**.

**Ejemplo:**

**app.component.html**

<nav class="navbar navbar-inverse">

<div class="container-fluid">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">JazzoSquare</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav">

<li class="active"><a href="#">Inicio<span class="sr-only">(current)</span></a></li>

<li><a href="#">Contacto</a></li>

</ul>

</div><!-- /.navbar-collapse -->

</div><!-- /.container-fluid -->

</nav>

<router-outlet></router-outlet>

**app.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

}

1. Crear componentes o vistas que utilizaremos en nuestra app (**nombre-componente.component.ts, html y css**) dentro de carpetas **app/nombre\_componente**.

**Ejemplo:**

**nombre-componente.component.html**

<h1>Contenido de la vista</h1>

**nombre-componente.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-nombre-componente',

templateUrl: './nombre-componente.component.html'

})

export class NombreComponenteComponent {

}

1. Declarar constante appRoutes de tipo Routes (@angular/router) en **app.module.ts** con las rutas a definir de nuestra app.

const appRoutes: Routes = [

{ path: '', component: NombreComponenteComponent },

{ path: 'vista1', component: NombreComponenteComponent },

{ path: 'vista2', component: OtroComponenteComponent }

]

## Navegación entre páginas

Se debe **usar SIEMPRE** el atributo **routerLink** que proporciona Angular para cambiar de vistas dentro de nuestra app **en lugar de** **href.**

Al usar un navbar de Bootstrap debemos utilizar el ruter también para activar el link correspondiente de la vista, esto lo podemos hacer usando **routerLinkActive="active" en lugar de class=”active”**.

**Ejemplo:**

**app.component.html**

<nav class="navbar navbar-inverse">

<div class="container-fluid">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">JazzoSquare</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav">

<li routerLinkActive="active"><a routerLink="/lugares">Inicio<span class="sr-only">(current)</span></a></li>

<li routerLinkActive="active"><a routerLink="/detalle">Detalle</a></li>

</ul>

</div><!-- /.navbar-collapse -->

</div><!-- /.container-fluid -->

</nav>

<router-outlet></router-outlet>

## Enviar parámetros en rutas

Tenemos que tomar en cuenta que el envío de parámetros será obligatorio en todos los llamados a ese path. Estos parámetros son los que se ven en la URL de la siguiente manera: www.pagina.com/usuario/1

**Ejemplo:**

**app.module.ts**

const appRoutes: Routes = [

{ path: '', component: LugaresComponent },

{ path: 'lugares', component: LugaresComponent },

{ path: 'detalle/:id', component: DetalleComponent }

]

**lugares.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-lugares',

templateUrl: './lugares.component.html',

styleUrls: ['../app.component.css']

})

export class LugaresComponent {

title = 'JazzoSquare';

lugares:any = [

{id: 1, plan: 'pagado', cercania: 1, distancia: 1, active: true, nombre: 'Florería La Gardenia'},

{id: 2, plan: 'gratuito', cercania: 1, distancia: 1.8, active: false, nombre: 'Donas La Pasadita'},

{id: 3, plan: 'gratuito', cercania: 2, distancia: 5, active: true, nombre: 'Veterinaria Huellitas Felices'},

{id: 4, plan: 'gratuito', cercania: 3, distancia: 10, active: true, nombre: 'Florería La Gardenia 2'},

{id: 5, plan: 'pagado', cercania: 3, distancia: 35, active: true, nombre: 'Donas La Pasadita 2'},

{id: 6, plan: 'gratuito', cercania: 3, distancia: 120, active: false, nombre: 'Veterinaria Huellitas Felices 2'}

];

lat:number = 20.5706569;

lng:number = -103.3483462;

constructor() {

}

}

**lugares.component.html**

<div class="container">

<div class="row">

<div class="col-md-8">

<h2>Lugares cercanos</h2>

<ul class="list-group">

<ng-container \*ngFor="let lugar of lugares">

<a routerLink="/detalle/{{lugar.id}}"><li class="list-group-item" \*ngIf="lugar.active">{{lugar.nombre}}</li></a>

</ng-container>

</ul>

</div>

<div class="col-md-4">

<h2>Destacados</h2>

</div>

</div>

</div>

**detalle.component.ts**

import { Component } from '@angular/core';

import { ActivatedRoute } from '@angular/router';

@Component({

selector: 'app-detalle',

templateUrl: './detalle.component.html'

})

export class DetalleComponent {

constructor(private route: ActivatedRoute) {

console.log(this.route.snapshot.params['id']);

}

}

## Parámetros tipo Query

Estos parámetros son los que se ven en la URL de la siguiente manera: [www.pagina.com/usuario?action=view](http://www.pagina.com/usuario?action=view).

**Ejemplo:**

**lugares.component.html**

<div class="container">

<div class="row">

<div class="col-md-8">

<h2>Lugares cercanos</h2>

<ul class="list-group">

<ng-container \*ngFor="let lugar of lugares">

<a routerLink="/detalle/{{lugar.id}}" [queryParams]="{action:'view'}"><li class="list-group-item" \*ngIf="lugar.active">{{lugar.nombre}}</li></a>

</ng-container>

</ul>

</div>

<div class="col-md-4">

<h2>Destacados</h2>

</div>

</div>

</div>

**detalle.component.ts**

import { Component } from '@angular/core';

import { ActivatedRoute } from '@angular/router';

@Component({

selector: 'app-detalle',

templateUrl: './detalle.component.html'

})

export class DetalleComponent {

constructor(private route: ActivatedRoute) {

console.log(this.route.snapshot.params['id']);

console.log('queryParams ', this.route.queryParams);

console.log('Param action ', this.route.snapshot.queryParams['action']);

}

}

## Usar parámetros en la vista

**Ejemplo:**

**detalle.component.ts**

import { Component } from '@angular/core';

import { ActivatedRoute } from '@angular/router';

@Component({

selector: 'app-detalle',

templateUrl: './detalle.component.html'

})

export class DetalleComponent {

lugares:any = [

{id: 1, plan: 'pagado', cercania: 1, distancia: 1, active: true, nombre: 'Florería La Gardenia', description: 'Descripción 1'},

{id: 2, plan: 'gratuito', cercania: 1, distancia: 1.8, active: false, nombre: 'Donas La Pasadita', description: 'Descripción 2'},

{id: 3, plan: 'gratuito', cercania: 2, distancia: 5, active: true, nombre: 'Veterinaria Huellitas Felices', description: 'Descripción 3'},

{id: 4, plan: 'gratuito', cercania: 3, distancia: 10, active: true, nombre: 'Florería La Gardenia 2', description: 'Descripción 4'},

{id: 5, plan: 'pagado', cercania: 3, distancia: 35, active: true, nombre: 'Donas La Pasadita 2', description: 'Descripción 5'},

{id: 6, plan: 'gratuito', cercania: 3, distancia: 120, active: false, nombre: 'Veterinaria Huellitas Felices 2', description: 'Descripción 6'}

];

id = null;

lugar: any = {};

constructor(private route: ActivatedRoute) {

console.log('Param id ', this.route.snapshot.params['id']);

console.log('queryParams ', this.route.queryParams);

console.log('Param action ', this.route.snapshot.queryParams['action']);

this.id = this.route.snapshot.params['id'];

this.lugar = this.buscarLugar();

}

buscarLugar() {

return this.lugares.filter((lugar) => { return lugar.id == this.id })[0] || null;

}

}

**detalle.component.html**

<div class="well">

<a routerLink="/lugares">Lugares</a> / {{lugar.nombre}}

</div>

<div class="container">

<h1>{{lugar.nombre}}</h1>

<hr/>

<small>{{lugar.distancia}} km</small>

<br/>

<div class="well">

<p>

{{lugar.description}}

</p>

<b>Plán: {{lugar.plan}}</b>

</div>

</div>

# Servicios

Permiten centralizar funcionalidad de nuestra app para ser utilizados desde cualquier componente.

Es buena práctica crear una carpeta dentro de app llamada **services** que contenga todos nuestros servicios.

**Ejemplo:**

**lugares.service.ts**

import {Injectable} from "@angular/core";

@Injectable()

export class LugaresService {

lugares:any = [

{id: 1, plan: 'pagado', cercania: 1, distancia: 1, active: true, nombre: 'Florería La Gardenia', description: 'Descripción 1'},

{id: 2, plan: 'gratuito', cercania: 1, distancia: 1.8, active: false, nombre: 'Donas La Pasadita', description: 'Descripción 2'},

{id: 3, plan: 'gratuito', cercania: 2, distancia: 5, active: true, nombre: 'Veterinaria Huellitas Felices', description: 'Descripción 3'},

{id: 4, plan: 'gratuito', cercania: 3, distancia: 10, active: true, nombre: 'Florería La Gardenia 2', description: 'Descripción 4'},

{id: 5, plan: 'pagado', cercania: 3, distancia: 35, active: true, nombre: 'Donas La Pasadita 2', description: 'Descripción 5'},

{id: 6, plan: 'gratuito', cercania: 3, distancia: 120, active: false, nombre: 'Veterinaria Huellitas Felices 2', description: 'Descripción 6'}

];

public getLugares() {

return this.lugares;

}

public buscarLugar(id) {

return this.lugares.filter((lugar) => { return lugar.id == id })[0] || null;

}

}

**app.module.ts**

…

import { LugaresService } from './services/lugares.service';

const appRoutes: Routes = [

{ path: '', component: LugaresComponent },

{ path: 'lugares', component: LugaresComponent },

{ path: 'detalle/:id', component: DetalleComponent },

{ path: 'contacto', component: ContactoComponent }

]

@NgModule({

declarations: [

AppComponent,

ResaltarDirective,

ContarClicksDirective,

DetalleComponent,

LugaresComponent,

ContactoComponent

],

imports: [

BrowserModule,

FormsModule,

AgmCoreModule.forRoot({

apiKey: 'AIzaSyCo-JSjZNGAlRDgz6JJ-B0yvJUMBjrsqAk'

}),

RouterModule.forRoot(appRoutes)

],

providers: [LugaresService],

bootstrap: [AppComponent]

})

export class AppModule { }

**lugares.component.ts**

import { Component } from '@angular/core';

import { LugaresService } from '../services/lugares.service';

@Component({

selector: 'app-lugares',

templateUrl: './lugares.component.html',

styleUrls: ['../app.component.css'],

**//Si un servicio solo se utilizará en un componente basta con agregar**

providers: [LugaresService]

//y no hay necesidad de agregarlo al app.module.ts

})

export class LugaresComponent {

title = 'JazzoSquare';

lat:number = 20.5706569;

lng:number = -103.3483462;

lugares = null;

constructor(private lugaresService: LugaresService) {

this.lugares = lugaresService.getLugares();

}

}

**detalle.component.ts**

import { Component } from '@angular/core';

import { ActivatedRoute } from '@angular/router';

import { LugaresService } from '../services/lugares.service';

@Component({

selector: 'app-detalle',

templateUrl: './detalle.component.html'

})

export class DetalleComponent {

id = null;

lugar: any = {};

constructor(private route: ActivatedRoute, private lugaresService: LugaresService) {

console.log('Param id ', this.route.snapshot.params['id']);

console.log('queryParams ', this.route.queryParams);

console.log('Param action ', this.route.snapshot.queryParams['action']);

this.id = this.route.snapshot.params['id'];

this.lugar = this.lugaresService.buscarLugar(this.id);

}

}

# AngularFire para Firebase

## Agregar registro usando AngularFireDatabase

1. Creamos proyecto en Firebase.
2. Modificar **enviroment.ts** o creamos un nuevo archivo **firebase.environment.ts** en la carpeta **src/enviroment** para agregar la configuración de firebase. Esta configuración la obtenemos del proyecto de Firebase.

export const environment = {

production: false,

firebase: {

apiKey: '<your-key>',

authDomain: '<your-project-authdomain>',

databaseURL: '<your-database-URL>',

projectId: '<your-project-id>',

storageBucket: '<your-storage-bucket>',

messagingSenderId: '<your-messaging-sender-id>'

}

};

**NOTA: Cuidado de NO SUBIR A REPORITORIO PÚBLICO.**

1. Instalar angularfire y firebase en nuestro proyecto.

**npm install angularfire2 firebase –save**

1. Crear servicio para realizar la conexión con Firebase utilizando **AngularFireDatabase**.

import { Injectable } from "@angular/core";

import { AngularFireDatabase } from "angularfire2/database/database";

@Injectable()

export class LugaresService {

lugares:any = [

{id: 1, plan: 'pagado', cercania: 1, distancia: 1, active: true, nombre: 'Florería La Gardenia', description: 'Descripción 1'},

{id: 2, plan: 'gratuito', cercania: 1, distancia: 1.8, active: false, nombre: 'Donas La Pasadita', description: 'Descripción 2'},

{id: 3, plan: 'gratuito', cercania: 2, distancia: 5, active: true, nombre: 'Veterinaria Huellitas Felices', description: 'Descripción 3'},

{id: 4, plan: 'gratuito', cercania: 3, distancia: 10, active: true, nombre: 'Florería La Gardenia 2', description: 'Descripción 4'},

{id: 5, plan: 'pagado', cercania: 3, distancia: 35, active: true, nombre: 'Donas La Pasadita 2', description: 'Descripción 5'},

{id: 6, plan: 'gratuito', cercania: 3, distancia: 120, active: false, nombre: 'Veterinaria Huellitas Felices 2', description: 'Descripción 6'}

];

constructor(private afDB: AngularFireDatabase) {

}

public getLugares() {

return this.lugares;

}

public buscarLugar(id) {

return this.lugares.filter((lugar) => { return lugar.id == id })[0] || null;

}

public guardarLugar(lugar) {

console.log(lugar);

this.afDB.database.ref('lugares/' + lugar.id).set(lugar);

}

}

1. Crear formulario donde obtendremos los datos a guardar en la base de datos.

**crear.component.ts**

import { Component } from '@angular/core';

import { LugaresService } from "../services/lugares.service";

@Component({

selector: 'app-crear',

templateUrl: './crear.component.html'

})

export class CrearComponent {

lugar: any = {};

constructor(private lugaresService: LugaresService) {

}

guardarLugar() {

this.lugar.id = Date.now();

this.lugaresService.guardarLugar(this.lugar);

}

}

**crear.component.html**

<div class="well">

<a routerLink="/lugares">Lugares</a> / Crear lugar

</div>

<div class="container">

<label>Nombre</label>

<input type="text" class="form-control" [(ngModel)]="lugar.nombre"/>

<br />

<label>Distancia</label>

<input type="number" class="form-control" [(ngModel)]="lugar.distancia"/>

<br />

<label>Cercanía</label>

<input type="number" class="form-control" [(ngModel)]="lugar.cercania"/>

<br />

<label>Descripción</label>

<textarea class="form-control" [(ngModel)]="lugar.descripcion"></textarea>

<br />

<label>Plan</label>

<input type="text" class="form-control" [(ngModel)]="lugar.plan"/>

<hr />

<a class="btn btn-primary" (click)="guardarLugar()">Enviar</a>

<a class="btn btn-default" routerLink="/lugares">Cancelar</a>

</div>

1. Importamos los módulos y servicios que utilizaremos en el **app.module.ts** junto con el archivo enviroment donde está la configuración de Firebase.

import { BrowserModule } from '@angular/platform-browser';

…

import { AngularFireModule } from 'angularfire2';

import { AngularFireDatabaseModule } from 'angularfire2/database';

import { AngularFireAuthModule } from 'angularfire2/auth';

import { environment } from '../environments/environment';

import { LugaresService } from './services/lugares.service';

import { CrearComponent } from './crear/crear.component';

const appRoutes: Routes = [

{ path: '', component: LugaresComponent },

…

{ path: 'crear', component: CrearComponent }

]

@NgModule({

declarations: [

AppComponent,

…

CrearComponent

],

imports: [

…

AngularFireModule.initializeApp(environment.firebase, 'jazzosquare'),

AngularFireDatabaseModule,

AngularFireAuthModule

],

providers: [LugaresService],

bootstrap: [AppComponent]

})

export class AppModule { }

## Consultando registros usando AngularFireDatabase

Ejemplo:

**lugares.service.ts**

import { Injectable } from "@angular/core";

import { AngularFireDatabase } from "angularfire2/database/database";

@Injectable()

export class LugaresService {

lugares:any = [

{id: 1, plan: 'pagado', cercania: 1, distancia: 1, active: true, nombre: 'Florería La Gardenia', description: 'Descripción 1'},

{id: 2, plan: 'gratuito', cercania: 1, distancia: 1.8, active: false, nombre: 'Donas La Pasadita', description: 'Descripción 2'},

{id: 3, plan: 'gratuito', cercania: 2, distancia: 5, active: true, nombre: 'Veterinaria Huellitas Felices', description: 'Descripción 3'},

{id: 4, plan: 'gratuito', cercania: 3, distancia: 10, active: true, nombre: 'Florería La Gardenia 2', description: 'Descripción 4'},

{id: 5, plan: 'pagado', cercania: 3, distancia: 35, active: true, nombre: 'Donas La Pasadita 2', description: 'Descripción 5'},

{id: 6, plan: 'gratuito', cercania: 3, distancia: 120, active: false, nombre: 'Veterinaria Huellitas Felices 2', description: 'Descripción 6'}

];

constructor(private afDB: AngularFireDatabase) {

}

public getLugares() {

return this.afDB.list('lugares/').valueChanges();

}

public buscarLugar(id) {

return this.lugares.filter((lugar) => { return lugar.id == id })[0] || null;

}

public guardarLugar(lugar) {

console.log(lugar);

this.afDB.database.ref('lugares/' + lugar.id).set(lugar);

}

}

**lugares.component.ts**

import { Component } from '@angular/core';

import { LugaresService } from '../services/lugares.service';

@Component({

selector: 'app-lugares',

templateUrl: './lugares.component.html',

styleUrls: ['../app.component.css']

})

export class LugaresComponent {

title = 'JazzoSquare';

lat:number = 20.5706569;

lng:number = -103.3483462;

lugares = null;

constructor(private lugaresService: LugaresService) {

lugaresService.getLugares()

.subscribe(lugares => {

this.lugares = lugares;

})

}

}

## Utilizando conexión HttpClient

La diferencia con conexión por socket es que con Http después de la petición se cierra la conexión, mientras que con socket se mantiene abierta desde un puerto por lo que se pueden ver los cambios de la base de datos en tiempo real.

### Llamados tipo POST

**Ejemplo:**

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

…

import { HttpClientModule } from '@angular/common/http';

const appRoutes: Routes = [

{ path: '', component: LugaresComponent },

…

{ path: 'crear/:id', component: CrearComponent }

]

@NgModule({

declarations: [

…

],

imports: [

…

HttpClientModule

],

providers: [LugaresService],

bootstrap: [AppComponent]

})

export class AppModule { }

**lugares.service.ts**

public guardarLugar(lugar) {

const headers = new Headers({"Content-Type": "application/json"});

return this.http.post(‘https://jazzosquare.firebaseio.com/lugares.json', lugar, {headers: headers});

}

**crear.component.ts**

guardarLugar() {

if (this.id == 'new') {

this.lugar.id = Date.now();

}

let direccion = this.lugar.direccion + "," + this.lugar.ciudad + "," + this.lugar.pais;

this.lugaresService.obtenerGeodata(direccion)

.subscribe(result => {

let direccion = Object.keys(result).map((key) => result[key]);

this.lugar.lat = direccion[0][0].geometry.location.lat;

this.lugar.lng = direccion[0][0].geometry.location.lng;

debugger

this.lugaresService.guardarLugar(this.lugar).subscribe(\_ => alert('guardado con exito'));

this.lugar = {};

});

}

### Llamados tipo GET

**Ejemplo:**

**app.module.ts**

import { BrowserModule } from '@angular/platform-browser';

…

import { HttpClientModule } from '@angular/common/http';

const appRoutes: Routes = [

{ path: '', component: LugaresComponent },

…

{ path: 'crear/:id', component: CrearComponent }

]

@NgModule({

declarations: [

…

],

imports: [

…

HttpClientModule

],

providers: [LugaresService],

bootstrap: [AppComponent]

})

export class AppModule { }

**lugares.service.ts**

API\_ENDPOINT = 'https://jazzosquare.firebaseio.com';

public getLugares() {

return this.http.get(this.API\_ENDPOINT + '/lugares.json');

}

**o:**

public getLugares() {

return this.http.get(this.API\_ENDPOINT + '/.json').map(result => result['lugares']);

}

**lugares.component.ts**

constructor(private lugaresService: LugaresService) {

lugaresService.getLugares()

.subscribe(lugares => {

this.lugares = Object.keys(lugares).map((key) => lugares[key]);

});

}

# Pipes

Son elementos que usamos en el **DOM** en el **HTML** junto con las directivas que van a tomar un elemento de entrada, le van a dar cierto formato y nos van a entregar una salida diferente.

Los **pipes** pueden tomar parámetros que le indiquemos.

Es posible que encadenemos pipes hasta que obtengamos el resultado que deseamos.

## Pipes de Angular

<https://angular.io/api/common/>

### uppercase, lowercase

**Ejemplo:**

<li class="list-group-item">

{{lugar.nombre | uppercase}}

</li>

### date

**Ejemplo:**

<li class="list-group-item">

<small>{{lugar.id | date:’medium’ | uppercase}}</small>

</li>

## Crear pipes

Se hace con una clase que utiliza en anotador @Pipe antes de la declaración de la clase.

**Ejemplo:**

**mipipe.pipe.ts**

import { Pipe, PipeTransform } from “@angular/core”;

@Pipe({name: ’mipipe’})

export class MiPipe implements PipeTransform {

transform(str: string): string {

//algoritmos para aplicarlos al string

return newstr;

}

}

# Test

<https://angular.io/guide/testing>

En el archivo **app.component.spec.ts** está un ejemplo de cómo realizar test. Para correr los unit test se tiene que ejecutar el comando **ng test**.

**Ejemplo:**

import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { PlaceComponent } from './place.component';

describe('PlaceComponent', () => {

beforeEach(async(() => {

TestBed.configureTestingModule({

declarations: [ PlaceComponent ]

})

}));

it('should create', () => {

let fixture = TestBed.createComponent(PlaceComponent);

let app = fixture.debugElement.componentInstance;

expect(app).toBeTruthy();

});

});

## Integrar test a servicios

**Ejemplo:**

it('should create', () => {

let fixture = TestBed.createComponent(PlaceComponent);

let app = fixture.debugElement.componentInstance;

let placeService = fixture.debugElement.injector.get(placeService);

expect(app.places).toEqual(placeService.places);

});