



## Angular 2+ Workshop



¿Angular 2+?

Instalación

npm

Requerimientos  
previos

Descripción  
del proyecto

Material  
Angular

CLI

Typescript

Componentes

Clases

Building

Mockups

Guards

Servicios



# Angular 2+

Angular.io

**Desarrollo Multiplataforma**

**Velocidad y Desempeño**

**CLI Poderosa y Versátil**



# Angular 2+

## Requerimientos

**Instalar Nodejs y NPM**

[nodejs.org/en/download](https://nodejs.org/en/download)

**Instalar Angular CLI**

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



# Angular 2+

## Descripción de proyecto

- ☰ Aplicación responsiva
- ☰ Diseño con las guías bases de Material Design
- ☰ Utilizar Pokémon API
- ☰ Buscador instantáneo de Pokémon's
- ☰ Lista de Pokémon's favoritos



# Angular 2+

npm

**Instalar paquete**

```
npm install <paquete>
```

**Instalar globalmente paquete**

```
npm install -g <paquete>
```

**Actualizar paquetes**

```
npm update
```

**Actualizar paquete**

```
npm update <paquete>
```



# NPM

package.json

**Enlista los paquetes instalados en proyecto**

**Permite especificar el versionamiento de paquetes por medio del estándar semántico de versionamiento**





## Versionamiento de paquetes:

CODE STATUS	STAGE	RULE	EXAMPLE #
First Release	New Product	Start with 1.0.0	1.0.0
Bug fixes, other minor changes	Patch Release	Increment the third digit	1.0.1
New Features that don't break existing features	Minor release	Increment the middle digit	1.1.0
Changes that break backward compatibility	Major release	Increment the first digit	2.0.0

# NPM

package.json

## Versionamiento de paquetes:

Patch releases: **1.0**   ó   **1.0.x**   ó   **~1.0.4**

Minor releases: **1**   ó   **1.x**   ó   **^1.0.4**

Major releases: **\***   ó   **x**



# Angular 2+

CLI

[cli.angular.io](https://cli.angular.io)



```
#Iniciar un nuevo proyecto  
ng new <Nombre del proyecto>
```

```
#Inicializando development server y ver en localhost:4200  
ng serve
```

```
#Obtener Ayuda  
ng help
```

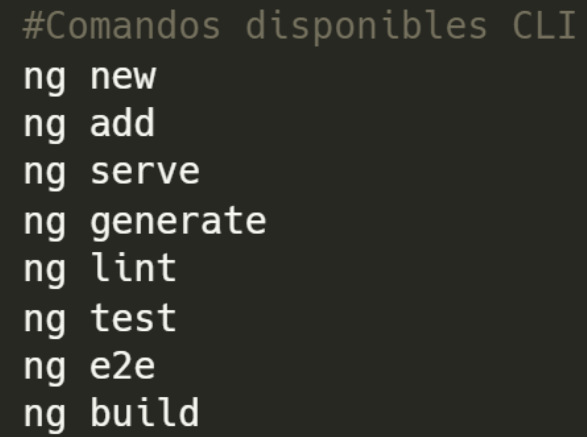


# Angular 2+

CLI

[cli.angular.io](https://cli.angular.io)

## Comandos



```
#Comandos disponibles CLI  
ng new  
ng add  
ng serve  
ng generate  
ng lint  
ng test  
ng e2e  
ng build
```



# Angular 2+

CLI

[cli.angular.io](https://cli.angular.io)



```
#Compila la aplicación y comienza un servidor web  
ng serve
```

```
#Opciones
```

```
--open #abre la aplicación una vez compilada
```

```
--port #cambia el puerto en el cual se muestra la app  
#default: 4200
```



# Angular 2+

CLI

[cli.angular.io](https://cli.angular.io)

```
#Crea la opción indicada
ng generate

#Opciones
class <nombre>           #genera una clase
component <nombre>       #genera un componente
directive <nombre>       #' ' directiva
guard <nombre>           #' ' guard
interface <nombre>       #' ' interface
module <nombre>          #' ' módulo
service <nombre>         #' ' servicio
```



# Angular 2+

CLI

[cli.angular.io](https://cli.angular.io)



```
#Compila la aplicación en el directorio de salida indicado  
ng build          #compila al directorio /dist en raíz
```

```
#Opciones
```

```
--target          #development    production  
--environment     #development    production
```

```
--href            #default se selecciona development  
                  #modificar el base tag
```



# Angular 2+

Material Angular  
material.angular.io



```
#NPM instalar Material Angular y Material CDK  
npm install --save @angular/material @angular/cdk
```


```
#YARN instalar Material Angular y Material CDK  
yarn add @angular/material @angular/cdk
```





# Angular 2+

Material Angular  
[material.angular.io](http://material.angular.io)



```
#NPM instalar Animaciones  
npm install --save @angular/animations  
  
#YARN instalar Animaciones  
yarn add @angular/animations
```



# Angular 2+

Material Angular  
[material.angular.io](http://material.angular.io)



```
#NPM Instalar soporte para gesturas  
npm install --save hammerjs
```

```
#YARN Instalar soporte para gesturas  
yarn add hammerjs
```



# Angular 2+

## Typescript

```
#Version typescript de una clase
class Greeter {
  greeting: string;
  constructor(message: string) {
    this.greeting = message;
  }
  greet() {
    return "Hello, " + this.greeting;
  }
}

let greeter = new Greeter("world");

let button = document.createElement('button');
button.textContent = "Say Hello";
button.onclick = function() {
  alert(greeter.greet());
}

document.body.appendChild(button);
```

```
#Version Javascript de una clase
var Greeter = /** @class */ (function () {
  function Greeter(message) {
    this.greeting = message;
  }
  Greeter.prototype.greet = function () {
    return "Hello, " + this.greeting;
  };
  return Greeter;
})();

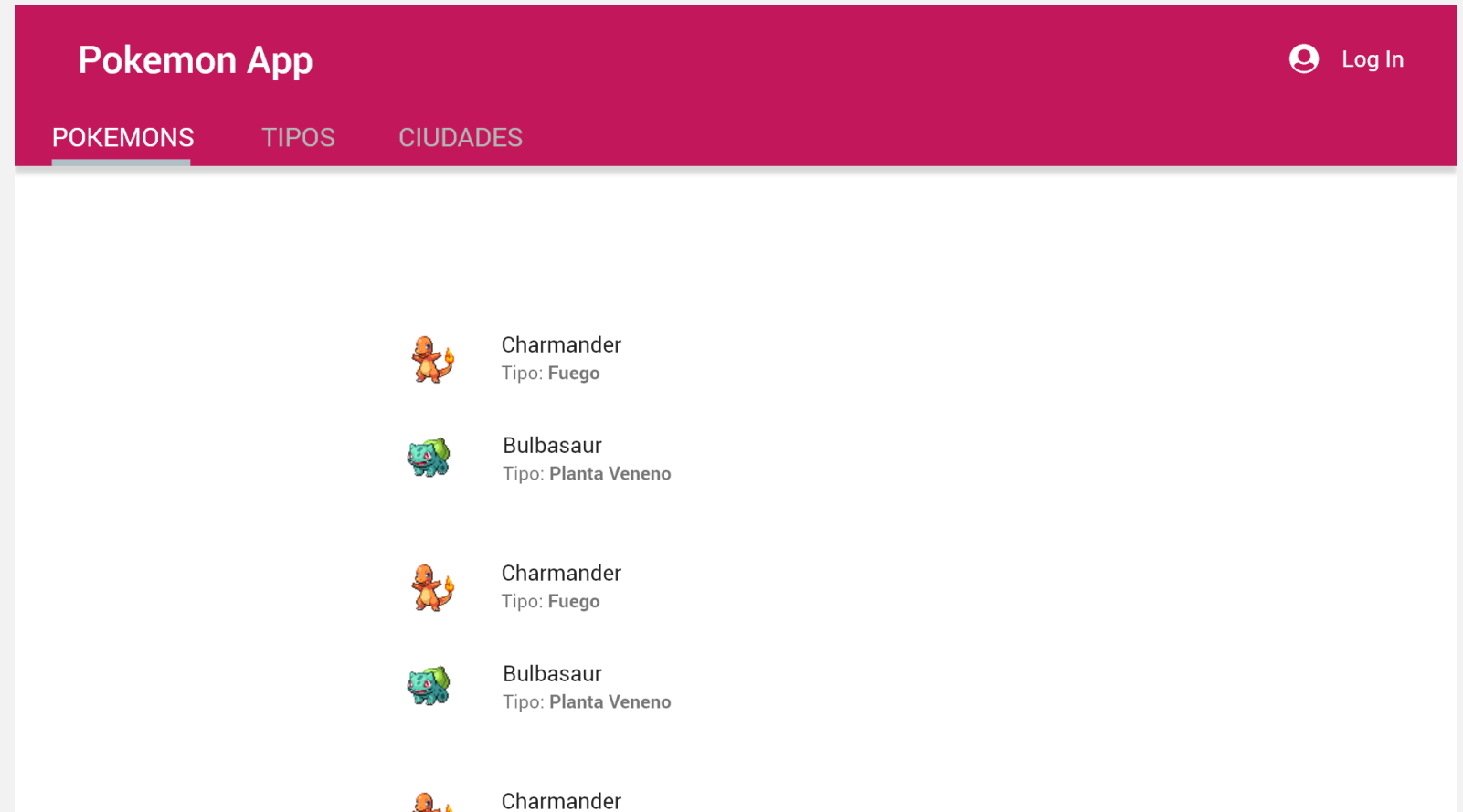
var greeter = new Greeter("world");
var button = document.createElement('button');
button.textContent = "Say Hello";
button.onclick = function () {
  alert(greeter.greet());
};

document.body.appendChild(button);
```



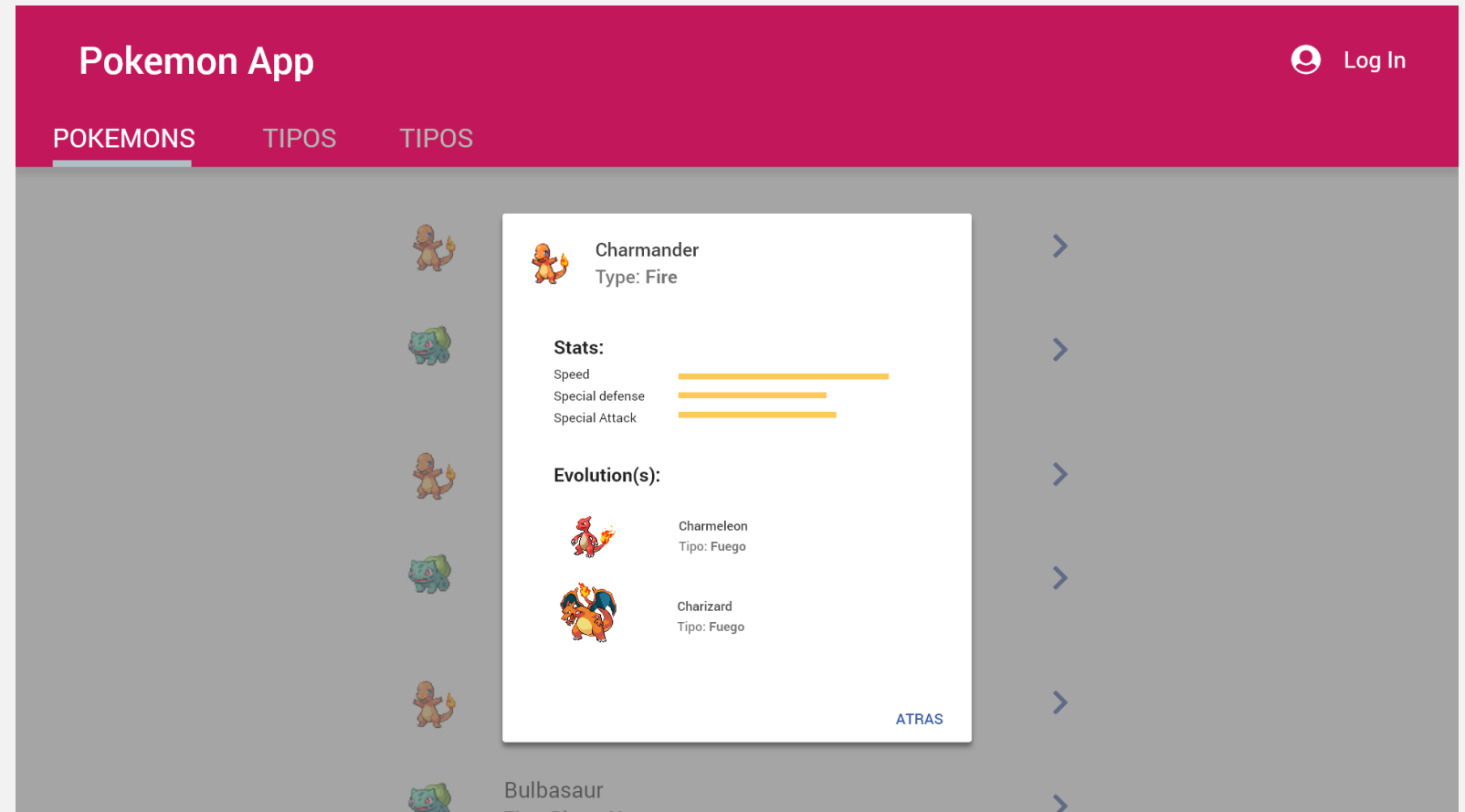
# Angular 2+

## Mockups



# Angular 2+

## Mockups



# Angular 2+

## Estructura de proyecto

#Organización de las carpetas de proyecto

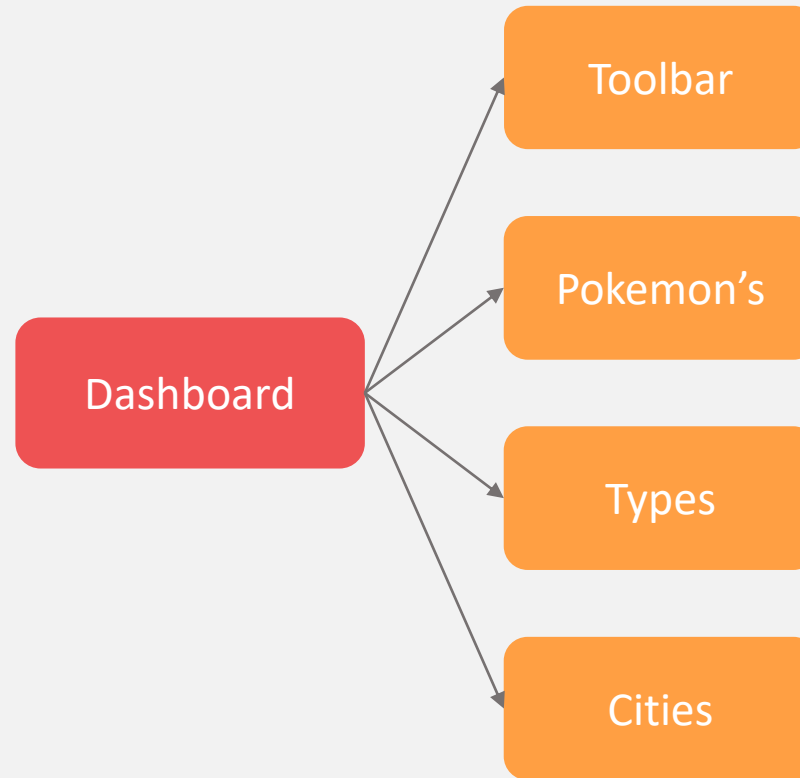
C:.

```
├── app
│   ├── classes
│   ├── components
│   └── services
├── assets
└── environments
```



# Angular 2+

## Componentes



# Angular 2+

Generando  
componentes

```
#creando el componente toolbar (menu superior)
$ ng g c components/toolbar --spec false

#creando el componente del tab pokemons
$ ng g c components/pokemons --spec false

#creando el componente del tab types
$ ng g c components/types --spec false

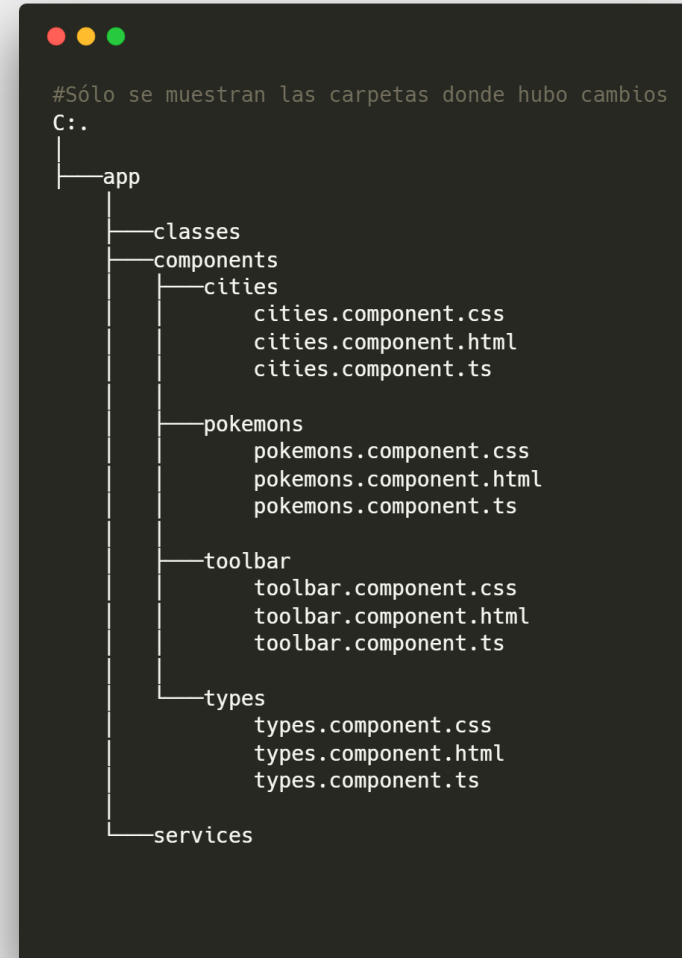
#creando el componente del tab cities
$ ng g c components/cities --spec false
```





# Angular 2+

## Estructura de proyecto



# Angular 2+

Coding

Primero:



```
#Inicializamos un servidor local para mostrar la app  
ng serve -open
```

Navegamos a:

src/app/app.component.html



# Angular 2+

Coding

> app.component.html

Borramos TODO de `app.component.html`

Colocamos el componente Toolbar `<app-toolbar></app-toolbar>`

Navegamos a:

`src/app/components/toolbar/toolbar.component.html`



# Angular 2+

Coding

> toolbar.component.html

Borramos TODO de [toolbar.component.html](#)

Colocamos el toolbar de material angular y sus dos renglones

```
<!-- Insertando el componente toolbar de material.angular  
en toolbar.component.html -->  
<mat-toolbar>  
  <mat-toolbar-row>  
    Primera  
  </mat-toolbar-row>  
  <mat-toolbar-row>  
    Segunda  
  </mat-toolbar-row>  
</mat-toolbar>
```



# Angular 2+

Coding

> app.module.ts

Navegamos a [app.module.ts](#)

Importamos las referencias necesarias para hacer el componente  
Toolbar

```
//Agregando en app.module.ts los imports necesarios para
//MatToolbar, MatButtonModule, MatIcon y MatTabs

import {MatTabsModule} from '@angular/material/tabs';
import {MatToolbarModule} from '@angular/material/toolbar';
import {MatButtonModule} from '@angular/material/button';
import {MatIconModule} from '@angular/material/icon';

//agregamos a la sección de imports:
MatTabsModule,
MatToolbarModule,
MatButtonModule,
MatIconModule
```



# Angular 2+

Coding  
> Toolbar

Navegamos a:

src/app/components/toolbar/toolbar.component.html

```
<!--Modificamos toolbar para que parezca de acuerdo al mockup-->
<mat-toolbar color="primary">
  <mat-toolbar-row>
    
    <span class="spacer-class"></span>
    <mat-icon>account_circle</mat-icon>
    <button mat-button>Log In</button>
  </mat-toolbar-row>
  <mat-toolbar-row>
    <nav mat-tab-nav-bar>
      <a mat-tab-link>Pokémon's</a>
      <a mat-tab-link>Types</a>
      <a mat-tab-link>Cities</a>
    </nav>
  </mat-toolbar-row>
</mat-toolbar>
```



# Angular 2+

Coding

>Rutas

En consola:

```
#generamos un modulo llamado routes  
# --flat sirve para generarlo en src/app en vez de su propio folder  
# --module=app utilizado para que actualice app.module.ts y lo importe  
ng g m routes --flat --module=app
```



# Angular 2+

Coding

>Rutas

Generamos el módulo con las rutas necesarias :

```
// Importando los componentes necesarios para generar las rutas
import { RouterModule, Routes } from '@angular/router';
import { PokemonsComponent } from '../components/pokemons/pokemons.component';
import { TypesComponent } from '../components/types/types.component';
import { CitiesComponent } from '../components/cities/cities.component';

// creando una constante del tipo Routes con las rutas que utilizaremos
const routes: Routes = [
  {path: 'pokemons', component: PokemonsComponent},
  {path: 'types', component: TypesComponent},
  {path: 'cities', component: CitiesComponent},
  {path: '', redirectTo: '/pokemons', pathMatch: 'full'}
];

@NgModule({
  exports: [ RouterModule ],
  imports: [ RouterModule.forRoot(routes) ]
})
export class RoutesModule { }
```

Navegamos a:

src/app/app.module.ts





# Angular 2+

Coding

>app.module.ts

## Importamos nuestro modulo de rutas

```
// Importando el módulo RoutesModule
import { RoutesModule } from './routes.module';

// Se agrega a imports como
RoutesModule
```

## Navegamos a:

src/app/app.component.html



# Angular 2+

Coding

>app.component.html

## Agregamos router-outlet



```
<!-- Ya teníamos puesto nuestro toolbar-->  
<app-toolbar></app-toolbar>  
  
<!-- Agregar Router Outlet-->  
<router-outlet></router-outlet>
```

## Navegamos a:

src/app/components/toolbar/toolbar.component.ts



# Angular 2+

Coding

>toolbar.component.ts

Crear objeto con nombres y rutas para vistas,  
importar lo necesario de Routes, RouterModule

```
// Importamos RouterModule, Routes para poder controlar las rutas
// por medio de las tabs
import { RouterModule, Routes } from '@angular/router';

// Creamos un arreglo de objetos de acuerdo a las rutas que tendrán
// las tabs con las propiedades name (vistas o tabs) path (rutas)
routes = [
  { name: 'Pokemons', path: '/pokemons' },
  { name: 'Types', path: '/types' },
  { name: 'Cities', path: '/cities' }
];
```

Navegamos a:

src/app/components/toolbar/toolbar.component.html



# Angular 2+

Coding

>toolbar.component.html

Modificamos MatTabs para que funcionen:

```
<nav mat-tab-nav-bar color="accent">
  <!-- ngFor para crear cuantos tabs tenga nuestro arreglo
        de objetos de rutas

        routerLink: utilizado para especificar a la ruta a
        la que va a dirigirse cuando se le de click

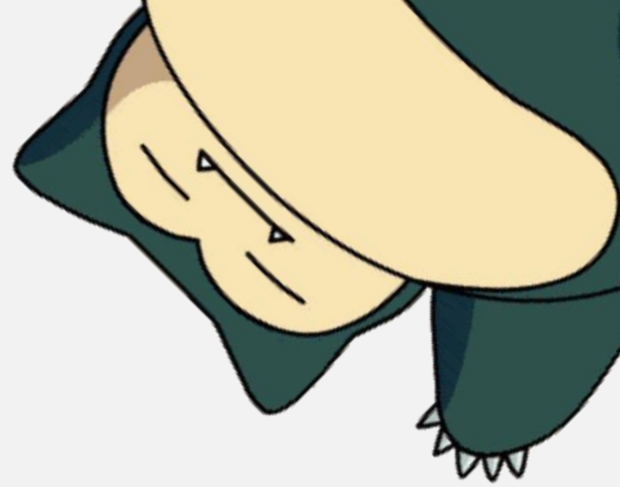
        routerLinkActive: utilizado para indicar cuando una
        ruta está activa.
  -->
  <a mat-tab-link
    *ngFor="let route of routes"
    [routerLink] = "route.path"
    routerLinkActive #rla="routerLinkActive"
    [active]="rla.isActive">
    {{route.name}}
  </a>
```



# Angular 2+

Coding

>toolbar.component.html



# ! Toolbar Finalizado !



# Angular 2+

Coding

# Pokémon's sección



# Angular 2+

Coding

> app.module.ts

Navegamos a:

src/app/app.module.ts

```
// Importamos la libreria MatList
import {MatListModule} from '@angular/material/list';

// Agregamos MatListModule a la lista de imports para utilizarlo
MatListModule
```

Navegamos a:

src/app/components/pokemons/pokemons.component.html



# Angular 2+

## Coding

> pokemons.component.html

## Borramos todo y agregamos lista estática

```
//Utilizamos una lista con renglones multi-línea
<mat-nav-list>
  <a mat-list-item>
    
    <h3 matLine><b>Pikachu</b></h3>
    <p matLine>
      <span class="demo-2">Tipo: Fuego</span>
    </p>
  </a>
  <a mat-list-item>
    
    <h3 matLine><b>Pikachu</b></h3>
    <p matLine>
      <span class="demo-2">Tipo: Fuego</span>
    </p>
  </a>
</mat-nav-list>
```





# Angular 2+

# PokeAPI

[pokeapi.co](https://pokeapi.co)



# Angular 2+

## Clases

```
// Clase para la llamada genérica de lista de todos los pokemon's

export class Result {
  name: string;
  url: string;
}
```

```
// Importamos la clase result creada

import { Result } from './result';

// Checamos la api para tipificar los demas datos que obtendremos de la
// respuesta
export class PokemonList {
  count: number;
  previous: string;
  results: Result[];
  next: string;
}
```



# Angular 2+

## Clases

```
// Clase para tipificar las habilidades del pokemon

export class Abilities {
  slot: number;
  is_hidden: boolean;
  ability: {
    url: string;
    name: string;
  };
}
```

```
// Clase para tipificar el form de la respuesta

export class Forms {
  url: string;
  name: string;
}
```



# Angular 2+

## Clases

```
// Tipificación de clase PokemonForm utilizada en respuesta API

export class PokemonForm {
  sprites: {
    front_default: string;
    back_default: string;
    front_shiny: string;
  };
  is_mega: string;
}
```

```
// Tipificación de clase sprites utilizada en respuesta API

export class Sprites {
  back_female: string;
  back_shiny_female: string;
  back_default: string;
  front_female: string;
  front_shiny_female: string;
  back_shiny: string;
  front_default: string;
  front_shiny: string;
}
```



# Angular 2+

## Clases

```
// Tipificación de clase Types utilizada en respuesta API

export class Types {
  slot: number;
  type: {
    url: {
      url: string;
      name: string;
    };
  };
};
}
```

```
// Tipificación de clase Stats utilizada en respuesta API

export class Stats {
  stat: {
    url: string;
    name: string;
  };
  effort: number;
  base_stat: number;
}
```



# Angular 2+

## Clases

```
// Importando clases necesarias
import { Forms } from './pokemon/forms';
import { Abilities } from './pokemon/abilities';
import { Moves } from './pokemon/moves';
import { Sprites } from './pokemon/sprites';
import { Types } from './pokemon/types';
import { Stats } from './pokemon/stats';

// Tipificado clase principal pokemon de respuesta
export class Pokemon {
  forms: Forms;
  abilities: Abilities[];
  stats: Stats;
  name: string;
  weight: number;
  moves: Moves[];
  sprites: Sprites;
  types: Types[];
}
```



# Angular 2+

Coding

# Observables y Servicios

(Lista dinámica)



# Angular 2+

Coding



```
# Generamos el servicio en consola con el cual traeremos los  
pokemons  
ng g s services/pokemons
```

**Navegamos a:**

src/app/services/pokemons.service.ts





# Angular 2+

## Coding

> pokemons.service.ts

```
// Imports necesarios para que nuestro servicio funcione

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { PokemonList } from '../classes/pokemonlist';
import { Result } from '../classes/result';
import { Pokemon } from '../classes/pokemon';

@Injectable()
export class PokemonService {

    //Variables y ruta necesaria para consumir la API
    //Arreglo en el cual se almacenarán los pokemons traídos
    pokemonList: Pokemon[] = [];
    // Ruta a la cual se le hará el request
    pokemonListURL = 'https://pokeapi.co/api/v2/pokemon/';

    // Variable en la cual se almacenará el objeto de la primera llamada
    pokemonResults: Result[] = [];

    constructor( private httpClient: HttpClient ) { }

    // Funcion la cual hace la llamada al API
    getPokemonList() {
        return this.httpClient.get(this.pokemonListURL)
            .subscribe( ( list: PokemonList ) => {
                this.pokemonResults = list.results;
                this.getPokemons();
            }
        );
    }

    // Funcion la cual trae de la API la lista de pokemons
    getPokemons() {
        this.pokemonResults.forEach(pokemonurl => {
            return this.httpClient.get(pokemonurl.url)
                .subscribe( (pokemon: Pokemon) => {
                    console.log(pokemon);
                    this.pokemonList.push(pokemon);
                });
        });
    }
}
```





# Luis E. Jiménez Robles

Egresado de **Ing. en Computación**

**GitHub Campus Expert** @TIJ / CDMX

**Project Manager** @GPOMCT

    /lusejroble

 [lusejimenezroble@gmail.com](mailto:lusejimenezroble@gmail.com)



