

Guía 3-B

Como crear y consumir servicios con Angular 8, utilizando arquitecturas middleware en Java EE 8 con Jax RS.



Los objetivos de esta guía son:

- ➔ Como integrar PrimeNG 8 en Angular 8.
- ➔ Como crear clases de tipo entidad.
- ➔ Como crear los servicios con angular.
- ➔ Explicar la definición de la lógica de programación en Angular 8 (MVC).
- ➔ Como crear la vista o front-end del usuario.

Código fuente: <https://github.com/arfloreshn/Leccion3-B-Curso-Angular-PrimefacesNG>

Lección 3-B				
Listado de Países				+ Nuevo
ID	PAIS	FECHA INDEPENDENCIA	TIENE MONARQUIA?	Acción
22	CHINA	17/11/2019	N	<button>Editar</button> <button>Borrar</button>
23	JAPON	27/11/2019	N	<button>Editar</button> <button>Borrar</button>
24	INDIA	27/11/2019	N	<button>Editar</button> <button>Borrar</button>
1	HONDURAS	30/10/2019	N	<button>Editar</button> <button>Borrar</button>
2	EL SALVADOR	30/10/2019	N	<button>Editar</button> <button>Borrar</button>
<div> 1 2 3 4 </div>				

Lección 3-B

Bendiciones, antes de iniciar la lección 3B leeremos la palabra del Señor.

«El corazón del entendido adquiere sabiduría; y el oído de los sabios busca la ciencia.» Proverbios 18:15.

Como he dicho en mis videos colaboro con mi país Honduras y comunidad catracha, porque creo en la capacidad de nosotros los catrachos de hacer grandes cosas.

Existe un potencial enorme en cada hondureño, lo que nos falta es, disciplina, entrega, amor, ganas y una compasión genuina y desinteresa por el prójimo.

A pesar de nuestras limitantes educacionales o falta de acceso a literatura, el catracho es inteligente por naturaleza, es capaz de entender las cosas y aplicarlas a su vida, solo necesita de un guía de cómo encontrar con ese camino para lograr su éxito.

Y en ese sentido todos podemos realmente aportar con temas sustanciales como lo son la educación, el trabajo, la familia y la colaboración desinteresa con el prójimo.

Iniciamos con la lección

Iniciare con una tabla de resumen de los comandos usados en la sección Lección 3-B, la tabla resume una explicación de todo lo relativo de programación con angular sin la jerga léxica del lenguaje propiamente mencionado.

No	Comandos	Tipo	Explicación
1	ng new leccion3B --minimal --style=css	proyecto	Crea un nuevo Proyecto de angular
2	npm i primeng --save	Libreria	Instala la libreria de PrimeNG dentro del proyecto creado
3	npm i primeicons --save	Libreria	Instala la libreria de primeicons dentro del proyecto creado
4	npm i @angular/animations --save	Libreria	Instala las animaciones de angular.
5	npm i primeflex --save	Libreria	Instala la libreria de primeflex dentro del proyecto creado
6	npm i semantic-ui --save	Libreria	Instala la libreria de semantix-ui dentro del proyecto creado
7	npm i @angular/cdk --save	Libreria	Instala la libreria de inyección de dependencias
8	npm i jquery --save	Libreria	Instala la libreria de jquery dentro del proyecto creado
9	npm i @types/jquery	Libreria	Instala la libreria de decoradores de jquery
10	ng g s services/paises	Servicio	Crea un nuevo servicio, dentro la carpeta services.
11	ng g c pages/paises	Componente	Crea un nuevo componente, dentro la carpeta pages.
12	ng g c share/header --spec=false -it	Componente	Crea un nuevo componente, dentro la carpeta share.
13	ng g c share/sidebar --spec=false -it	Componente	Crea un nuevo componente, dentro la carpeta share.

Estructura de la lección 3B, post ejecución de los 13 pasos previos.

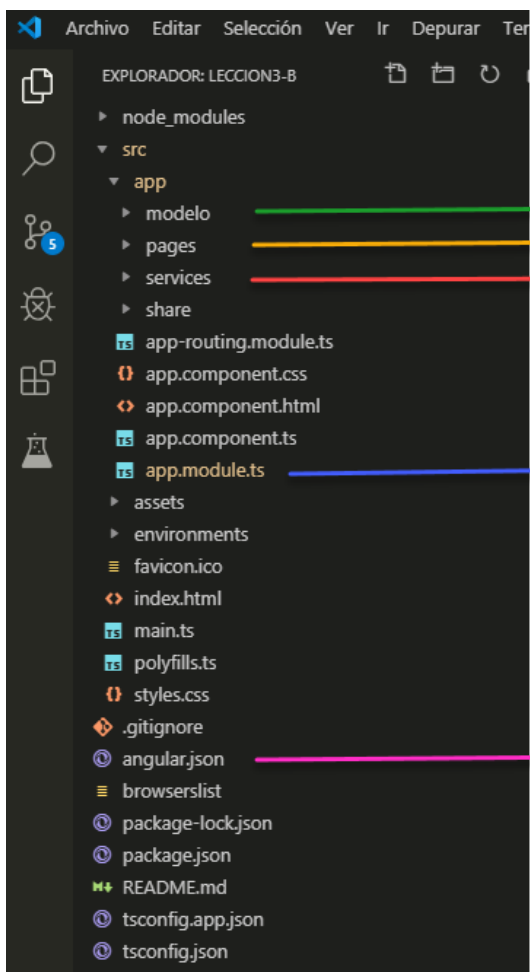


Diagram illustrating the project structure of 'LECCION3-B' after executing the 13 steps. The structure is shown in the Explorer view, with annotations explaining the purpose of key directories and files:

- node_modules**: Directory for installed packages.
- src**: Source code directory.
 - app**: Main application directory.
 - modelo**: Directory for classes of collections (entity or tables).
 - pages**: Directory for HTML components and Code Behind pages.
 - services**: Directory for service classes or middleware.
 - share**: Directory for shared components.
 - app-routing.module.ts**: Declaration and import of routes.
 - app.component.css**: Styles for the main component.
 - app.component.html**: HTML template for the main component.
 - app.component.ts**: Declaration and import of libraries and providers.
 - app.module.ts**: Declaration and import of libraries and providers.
 - assets**: Directory for static assets.
 - environments**: Directory for environment configurations.
 - favicon.ico**: Application icon.
 - index.html**: Main HTML file.
 - main.ts**: Main entry point of the application.
 - polyfills.ts**: Polyfills for the application.
 - styles.css**: Global styles.
 - .gitignore**: Git ignore file.
 - angular.json**: Declaration of styles and JS files of JQuery.
 - browserslist**: Browserslist configuration.
 - package-lock.json**: Lock file for npm packages.
 - package.json**: Package manifest file.
 - README.md**: Readme file.
 - tsconfig.app.json**: TypeScript configuration for the application.
 - tsconfig.json**: TypeScript configuration for the project.

➡ Como integrar PrimeNG 8 en Angular 8.

Para integrar las librerías de PrimeNG basta con ejecutar los siguientes comandos y configuraciones:

No	Comandos	Tipo	Explicación
1	<code>npm i primeng --save</code>	Librería	Instala la librería de PrimeNG dentro del proyecto creado
2	<code>npm i primeicons --save</code>	Librería	Instala la librería de primeicons dentro del proyecto creado
3	<code>npm i @angular/animations --save</code>	Librería	Instala las animaciones de angular.
4	<code>npm i primeflex --save</code>	Librería	Instala la librería de primeflex dentro del proyecto creado
5	<code>npm i semantic-ui --save</code>	Librería	Instala la librería de semantix-ui dentro del proyecto creado

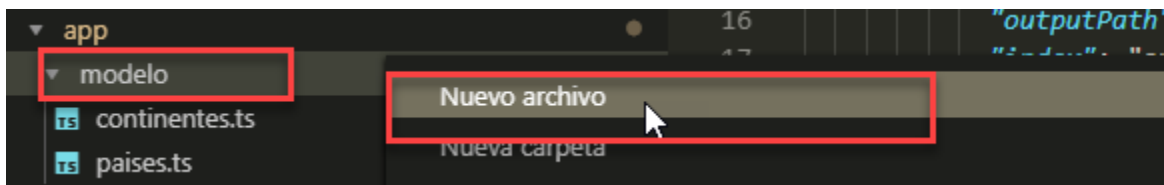
Y configurar dentro del archivo angular.json lo siguiente:

```
"styles": ["node_modules/semantic-ui-css/semantic.min.css",
            "node_modules/primeflex/primeflex.css",
            "node_modules/primeicons/primeicons.css",
            "node_modules/primeng/resources/themes/nova-colored/theme.css",
            "node_modules/primeng/resources/primeng.min.css",
            "src/styles.css"
        ],
```

➡ Como crear clases de tipo entidad.

Esta sencillo como crear un nuevo archivo con extensión <.ts>

Ejemplo: <marcas.ts> la extesion ts, significa TypeScript.



Definición de una clase tipo entidad o tabla.

Se recomienda al momento de crear una clase de entidad en angular, los campos deben llamarse exactamente igual a los nombres de los campos declarados en la clase modelo de Java EE, esto con el fin de tener un estándar y una asimetría entre el middleware y clase de entidad creada en angular.

```
export class Países{
    constructor(public paisid:number,
        public nomPais:string,
        public idContinente: number,
        public fecIndependencia: Date,
        public sn_monarquia: string ){}
}
```

Las clases de tipo modelo, son la representación lógica de las tablas en la base de datos.

```
export class Continentes{  
    constructor(  
        public id:number  
        , public nomContinente:string){}  
}
```

➡ Como crear los servicios con angular.

Para crear un servicio en angular basta con ir a la consola de comandos y escribir el siguiente comando.

No	Comando	Tipo	Explicación
1	ng g s services/paises	Servicio	Crea un nuevo servicio, dentro la carpeta services.

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Paises } from '../modelo/paises';
const httpOpciones = {
  headers: new HttpHeaders({'Content-Type' : 'application/json'})
};

@Injectable({
  providedIn: 'root'
})

export class PaisesService {
  apiUrl:String = "http://localhost:8080/leccion3/api/servicio.paises";
  constructor(private http:HttpClient) { }
  //Consume el metodo findAll() del backend para listar todos los registros
  ListarTodo() : Observable<any> {
    return this.http.get(this.apiUrl + "/");
  }
  //Consume el metodo post del backend para crear un nuevo registro
  crearPais(var_pais: Paises) : Observable<any> {
    const body = JSON.stringify(var_pais);
    return this.http.post(this.apiUrl + "/", body , httpOpciones);
  }
  //Consume el metodo put del backend para modificar un registro
  modificarPais(var_pais: Paises) : Observable<any> {
    const body = JSON.stringify(var_pais);
    return this.http.put(this.apiUrl + "/" + var_pais.paisid, body, httpOpciones);
  }
  //Consume el metodo delete del backend para borrar un registro
  borrarPais(id: number) : Observable<any> {
    return this.http.delete(this.apiUrl + "/" + id);
  }
}
```

Los cuatro aspectos de suma importancia en la creación de los servicios en angular son el @Injectable, HttpClient, HttpHeaders y los Observable.

```
constructor(private http:HttpClient)
```

El primer aspecto es @Injectable:

Este decorador nos permite la inyección de dependencias, en el caso de los servicios lo que vamos inyectar es el httpClient.

El HttpClient

El método HttpClient contiene toda la metodología de comunicación entre nuestro cliente creado en angular y el middleware de servicio creado en Java EE y que usaremos para comunicarnos con la base de datos.

Los HttpHeaders

Son las opciones de identificación que nuestro cliente, envía al servidor de aplicaciones y que el servidor recibe y procesa para aceptar o no nuestra solicitud.

Ejemplo de un cliente en Java EE:

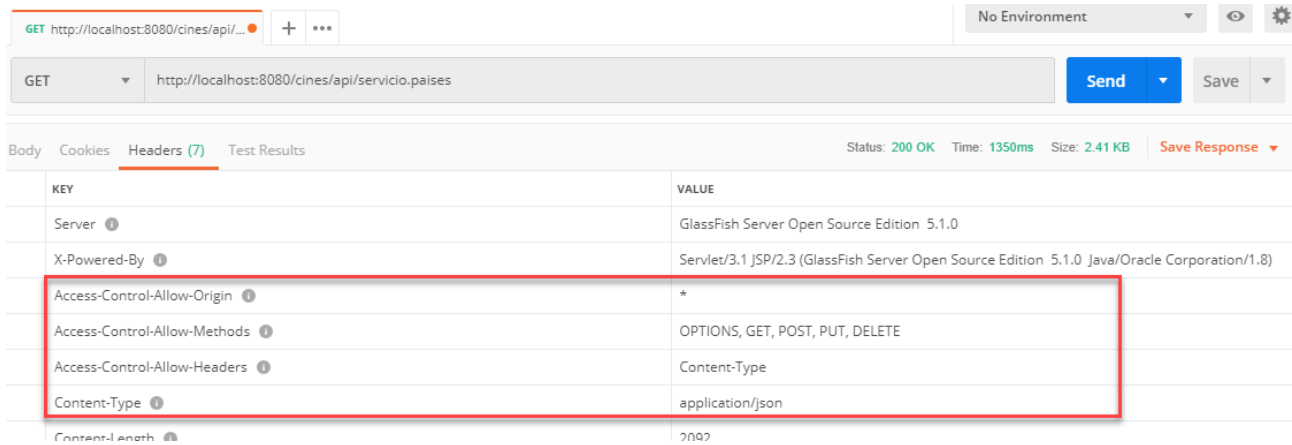
```
DefaultHttpClient client = new DefaultHttpClient();
HttpGet get = new HttpGet("http://example.com/customers/1");
get.addHeader("accept", "application/xml");

HttpResponse response = client.execute(get);
if (response.getStatusLine().getStatusCode() != 200) {
    throw new RuntimeException("Operation failed: " +
        response.getStatusLine().getStatusCode());
}
```

Para tener mas claro que los valida nuestro servidor.

Usando POSTMAN podemos revisar las cabeceras que se envían por medio de postman y las siguientes: acceso al origen, acceso a los métodos, el tipo de contenido y la definición del contenido, estas 4 opciones las debemos enviar desde nuestro cliente.

Ya sea en angular o java EE, debemos enviar los headers tal y como lo hace postman, caso contrario el servidor rechazara nuestra solicitud de comunicación.



KEY	VALUE
Server	GlassFish Server Open Source Edition 5.1.0
X-Powered-By	Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 5.1.0 Java/Oracle Corporation/1.8)
Access-Control-Allow-Origin	*
Access-Control-Allow-Methods	OPTIONS, GET, POST, PUT, DELETE
Access-Control-Allow-Headers	Content-Type
Content-Type	application/json
Content-Length	7092

El cuarto aspecto son los Observables:

Podemos referirnos a los observable como una tarea de monitoreo, y estará monitoreando hasta recibir las información.

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Países } from '../modelo/paises';
//Consumo el metodo findAll() del backend para listar todos los registros
ListarTodo() : Observable<any> {
    return this.http.get(this.apiUrl + "/");
}
```


➡ Explicar la definición de la lógica de programación en Angular 8 (MVC).

Tratare de explicar de una manera simple y ordena o que al menos la considero simple.

Modelo	Controlador	Vista
<div>services</div> <div>países.service.ts</div>	<div>países</div> <div>países.component.css</div> <div>países.component.html</div> <div>países.component.ts</div>	<div>países</div> <div>países.component.css</div> <div>países.component.html</div>

La vista le hace una petición al controlador, el controlador a su vez, le hace una petición al modelo.

Si todo esta bien, la respuesta es positiva y nuestra petición se procesa con éxito.

caso contrario un hermoso error que nos pone a batear por largos ratos.

En el controlador es donde definimos la mayor parte de nuestra lógica de programación en este caso usando TypeScript como sintaxis y utilizando en muchos casos la lógica de programación de JavaScript.

En la vista es donde pegamos todos nuestros componentes de PrimeNG, HTML o Angular o Semantic.

Vista

Lección 3-B				
Listado de Países				
ID	PAIS	FECHA INDEPENDENCIA	TIENE MONARQUIA?	Acción
22	CHINA	17/11/2019	N	<button>Editar</button> <button>Borrar</button>
23	JAPON	27/11/2019	N	<button>Editar</button> <button>Borrar</button>
24	INDIA	27/11/2019	N	<button>Editar</button> <button>Borrar</button>
1	HONDURAS	30/10/2019	N	<button>Editar</button> <button>Borrar</button>
2	EL SALVADOR	30/10/2019	N	<button>Editar</button> <button>Borrar</button>
<div> 1 2 3 4 </div>				

Figura 2

Crear un nuevo Pais ✕

Pais

Continente

Seleccione un continente ▼

Fecha Independencia

04/11/2019

Guardar

Figura 3

Editar Registro ✕

Pais

CHINA

Fecha Independencia

18/11/2019

Seleccione un continente

☒ Asia

☐ America

☐ Africa

☐ Antartida

☐ Oceania

Guardar

Figura 4

Borrar Registro ✕

Id

22


Pais

CHINA

✓ Continuar

Tipo de confirmación

Confirmación ✕

 Esta seguro que desea proceder?

✓ Yes ✕ No

Tipos de mensajes de notificación.

Listado de Países ℹ Cancelado ✕
Acción cancelada

ID	PAIS	FECHA INDEPENDENCIA	TIENE MONARQUIA?	Acción
22	CHINA	18/11/2019	N	ℹ Editar ✕ Borrar
23	JAPON	27/11/2019	N	ℹ Editar ✕ Borrar
24	INDIA			ℹ Editar ✕ Borrar
1	HONDURAS			ℹ Editar ✕ Borrar
2	EL SALVADOR			ℹ Editar ✕ Borrar

Borrar Registro ✕

Id

24

País

INDIA

✓ Continuar

 www.linkedin.com/in/arfloreshn