

Arquitecturas de Software para Aplicaciones Empresariales

Listar clientes en Angular aplicando una
fachada, observadores y manejo de Rutas



PROGRAMA DE INGENIERIA DE SISTEMAS

Ing. Daniel Eduardo Paz Perafán (danielp@Unicauca.edu.co)

Ing. Pablo A. Magé (pimage@Unicauca.edu.co)

- a) Creación de un componente clientes
- b) Mostrar una lista de clientes almacenados en un vector dentro del componente
- c) Mostrar una lista de clientes almacenados en una constante creada dentro de un archivo
- d) Mostrar una lista de clientes mediante un servicio
- e) Mostrar una lista de clientes mediante el uso de observadores
- f) Introducción a las rutas

Aplicación 1

Requisitos iniciales

C:\> Símbolo del sistema

```
C:\Users\LENOVO>node -v  
v14.15.0
```

```
C:\Users\LENOVO>npm -v  
6.14.8
```

```
C:\Users\LENOVO>tsc -v  
Version 4.0.5
```

C:\> Símbolo del sistema

```
C:\Users\LENOVO>ng version
```

The logo for Angular CLI, featuring the word "Angular" in a stylized font with a small square icon to its left, and "CLI" in a larger, outlined font to its right.

```
Angular CLI: 10.2.0  
Node: 14.15.0  
OS: win32 x64
```

```
Angular:
```

```
...
```

```
Ivy Workspace:
```

Package	Version
@angular-devkit/architect	0.1002.0 (cli-only)
@angular-devkit/core	10.2.0 (cli-only)
@angular-devkit/schematics	10.2.0 (cli-only)
@schematics/angular	10.2.0 (cli-only)
@schematics/update	0.1002.0 (cli-only)

Para crear un proyecto utilizamos el comando `ng new clientes-app`

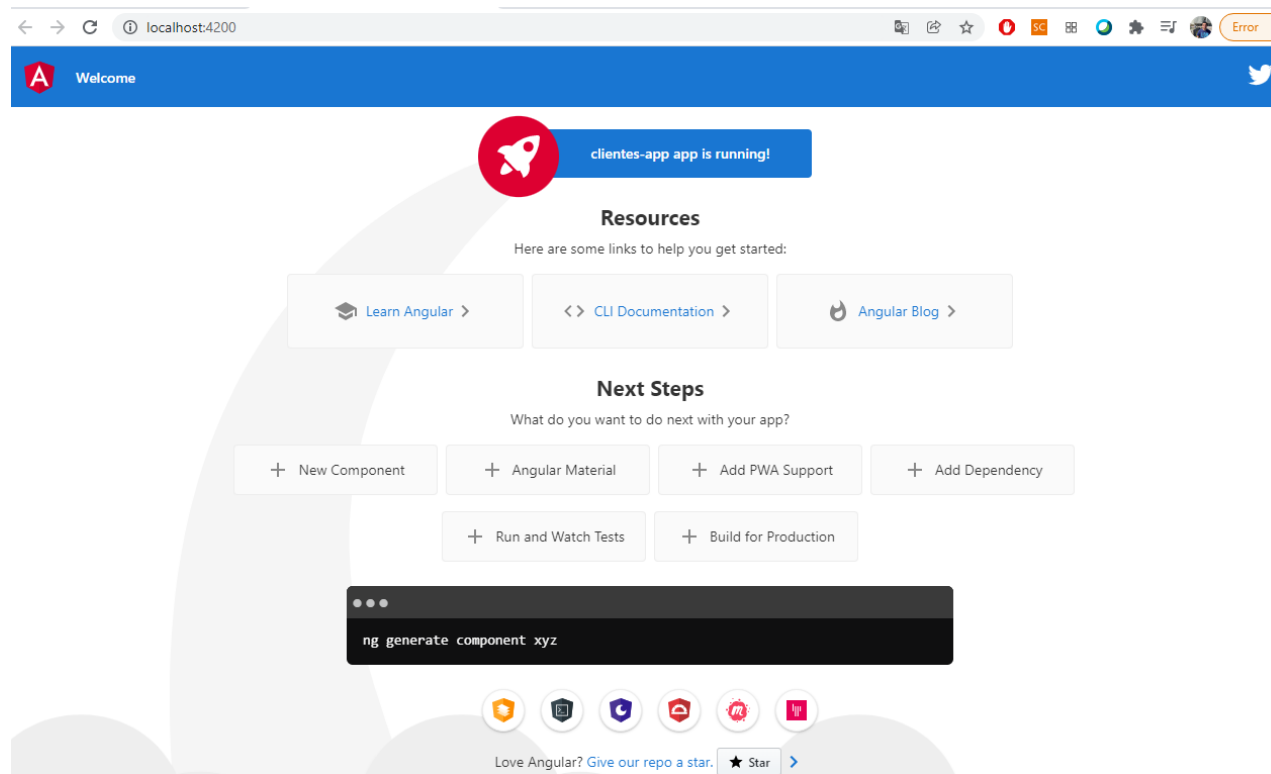
```
D:\aplicaciones desarrolladas\angular\1. Introducción a los componentes>ng new clientes-app
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE clientes-app/angular.json (3614 bytes)
CREATE clientes-app/package.json (1255 bytes)
CREATE clientes-app/README.md (1020 bytes)
CREATE clientes-app/tsconfig.json (458 bytes)
CREATE clientes-app/tslint.json (3185 bytes)
CREATE clientes-app/.editorconfig (274 bytes)
CREATE clientes-app/.gitignore (631 bytes)
CREATE clientes-app/.browserslistrc (853 bytes)
CREATE clientes-app/karma.conf.js (1024 bytes)
CREATE clientes-app/tsconfig.app.json (287 bytes)
CREATE clientes-app/tsconfig.spec.json (333 bytes)
CREATE clientes-app/src/favicon.ico (948 bytes)
CREATE clientes-app/src/index.html (297 bytes)
CREATE clientes-app/src/main.ts (372 bytes)
CREATE clientes-app/src/polyfills.ts (2835 bytes)
CREATE clientes-app/src/styles.css (80 bytes)
CREATE clientes-app/src/test.ts (753 bytes)
CREATE clientes-app/src/assets/.gitkeep (0 bytes)
CREATE clientes-app/src/environments/environment.prod.ts (51 bytes)
CREATE clientes-app/src/environments/environment.ts (662 bytes)
CREATE clientes-app/src/app/app.module.ts (314 bytes)
CREATE clientes-app/src/app/app.component.html (25725 bytes)
CREATE clientes-app/src/app/app.component.spec.ts (958 bytes)
```

Lanzar el servidor de prueba

UNIVERSIDAD DEL CAUCA – FIET
DEPARTAMENTO DE SISTEMAS

Para lanzar el servidor de pruebas debemos colocar el comando `ng serve -o`

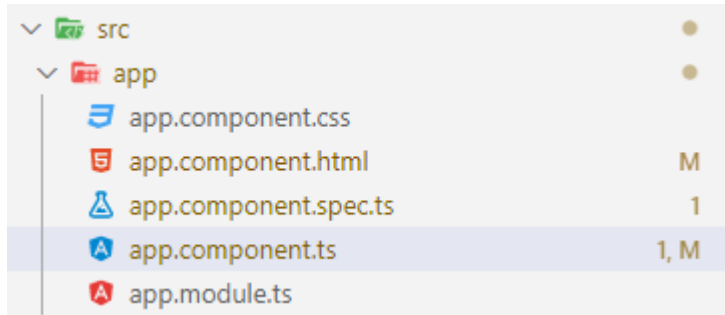
D:\aplicaciones desarrolladas\angular\1. Introducción a los componentes\clientes-app>ng serve -o



Si deseamos lanzar el servidor en un puerto en particular debemos colocar el comando `ng serve -o --port 4444`

Recordatorio de las partes de un componente

UNIVERSIDAD DEL CAUCA – FIET
DEPARTAMENTO DE SISTEMAS



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

➡ Importamos el decorador

```
@Component({
```



Decorador que marca una clase como un componente Angular

```
  selector: 'app-root',
```



Plantea una etiqueta para usar el componente

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



Interface con la cual interactuará el usuario

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



Hoja de estilo que afecta al componente

```
})
```

```
export class AppComponent {
```



Clase que representa el controlador

```
  title = 'clientes-app';
```

```
  nombre:string='arquitecturas empresariales';
```



Atributos de la clase

```
}
```

Estando ubicados en el directorio clientes-app creamos un componente ejecutando el comando

`ng g c clientes`

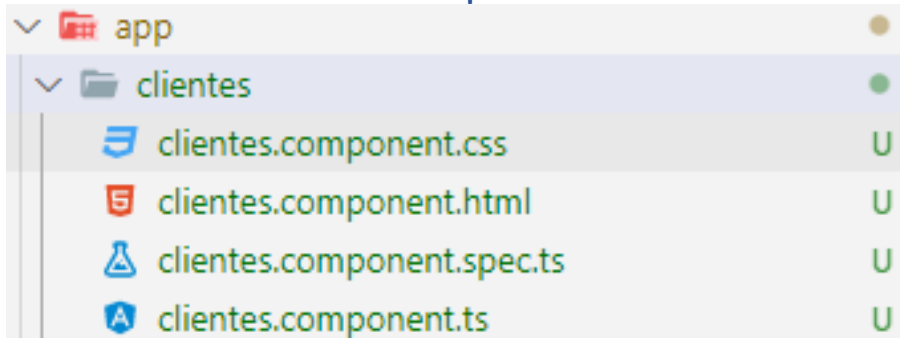
```
D:\aplicaciones desarrolladas\angular\1. Introducción a los componentes\clientes-app>ng g c clientes
CREATE src/app/clientes/clientes.component.html (23 bytes)
CREATE src/app/clientes/clientes.component.spec.ts (640 bytes)
CREATE src/app/clientes/clientes.component.ts (283 bytes)
CREATE src/app/clientes/clientes.component.css (0 bytes)
UPDATE src/app/app.module.ts (404 bytes)
```

Se crean todos los archivos que constituyen un componente

Se actualiza el app.module de manera automática

Crear un componente clientes

Archivos creados de manera automática, los cuales constituyen el componente clientes.
Cada componente se recomienda que este dentro de un directorio



Cada componente debe estar registrado en el archivo module.ts correspondiente al modulo al cual corresponda el componente

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

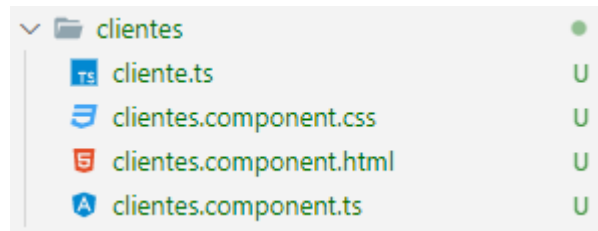
import { AppComponent } from './app.component';
import { ClientesComponent } from './clientes/clientes.component';

@NgModule({
  declarations: [
    AppComponent,
    ClientesComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Estando ubicados en el directorio `src/app/clientes` creamos una clase asociada al modelo con el comando

```
ng g class cliente
```

Archivo creado



Modificamos el archivo de la siguiente manera:

```
export class Cliente
{
  id!: number;
  nombre!: string;
  apellido!: string;
  createdAt!: string;
  email!: string;
}
```

Listado objetos de tipo cliente en clientes.component.ts

En el componente creamos un atributo que almacenará los clientes que vamos a listar

```
import { Component, OnInit } from '@angular/core';
import { Cliente } from '../cliente';

@Component({
  selector: 'app-clientes',
  templateUrl: './clientes.component.html',
  styleUrls: ['./clientes.component.css']
})
export class ClientesComponent implements OnInit {

  clientes: Cliente[]=[
    {id: 1, nombre: 'Juan', apellido: 'Perez', email: 'juan@unicauca.edu.co', createAt: '2021-05-14'},
    {id: 2, nombre: 'Andres', apellido: 'Sanchez', email: 'andres@unicauca.edu.co', createAt: '2022-06-14'},
    {id: 1, nombre: 'Pedro', apellido: 'Cortez', email: 'pedro@unicauca.edu.co', createAt: '2018-02-14'}
  ]
  constructor() { }

  ngOnInit(): void {
  }
}
```

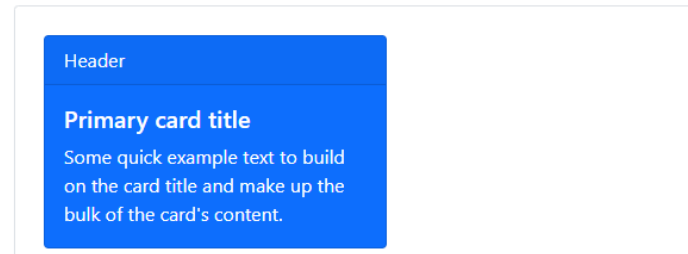
Listado objetos de tipo cliente en clientes.component.html

Buscamos en la pagina de bootstrap el código para el elemento card, el cual permitirá establecer un estilo para la tabla donde se listaran los clientes

Interface grafica

Background and color

Use [text color](#) and [background utilities](#) to change the appearance of a card.



Código fuente

```
<div class="card text-white bg-secondary mb-3" style="max-width: 18rem;">
  <div class="card-header">Clientes</div>
  <div class="card-body">
    <h5 class="card-title">Listado de clientes</h5>

    </div>
  </div>
```

Listado objetos de tipo cliente en clientes.component.html

```
<div class="card text-white bg-secondary mb-3" style="max-width: 18rem;">
  <div class="card-header">Clientes</div>
  <div class="card-body">
    <h5 class="card-title">Listado de clientes</h5>
    <table class="table table-bordered table-striped">
      <thead>
        <tr>
          <th>id</th>
          <th>nombres</th>
          <th>apellidos</th>
          <th>email</th>
          <th>fecha de creación</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let cliente of clientes">
          <td>{{cliente.id}}</td>
          <td>{{cliente.nombre}}</td>
          <td>{{cliente.apellido}}</td>
          <td>{{cliente.email}}</td>
          <td>{{cliente.createAt}}</td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

Utilizamos la directiva *ngFor para recorrer el conjunto de clientes almacenados en el atributo clientes

Utilizamos el componente creado, colocando el nombre del componente en el archivo app.component.htm


app.component.html M ●

src > app > app.component.html > ...

```
1 <app-header> </app-header>
2 <div class="container">
3   <app-clientes></app-clientes>
4 </div>
5 <app-footer></app-footer>
6
```

Visualizar el componente

La tabla donde se listan los clientes se muestra de la siguiente manera



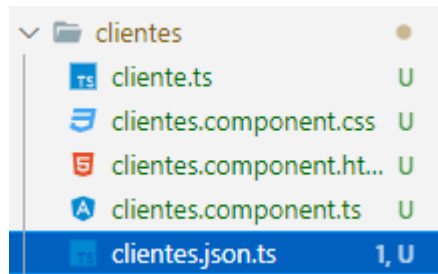
The screenshot shows a web browser window with the title 'ClientesApp'. The address bar displays 'localhost:4200'. The page content is on a blue background and includes a header 'Clientes' and a sub-header 'Listado de clientes'. Below the sub-header is a table with five columns: 'id', 'nombres', 'apellidos', 'email', and 'fecha de creación'. The table contains three rows of data.

id	nombres	apellidos	email	fecha de creación
1	Juan	Perez	juan@unicauca.edu.co	2021-05-14
2	Andres	Sanchez	andres@unicauca.edu.co	2022-06-14
1	Pedro	Cortez	pedro@unicauca.edu.co	2018-02-14

Aplicación 2

Listar los clientes desde un archivo de texto

Crear un archivo cliente.json.ts, el cual almacenará los clientes



Crear una constante la cual relaciona el vector de clientes

```
import {Cliente} from './cliente';

export const CLIENTES: Cliente[] = [
  {id: 1, nombre: 'Juan', apellido: 'Perez', email: 'juan@unicauca.edu.co', createAt: '2021-05-14'},
  {id: 2, nombre: 'Andres', apellido: 'Sanchez', email: 'andres@unicauca.edu.co', createAt: '2022-06-14'},
  {id: 1, nombre: 'Pedro11', apellido: 'Cortez', email: 'pedro@unicauca.edu.co', createAt: '2018-02-14'}
];
```

Listar los clientes desde un archivo de texto

Modificar el componente de la siguiente forma

```
import { Component, OnInit } from '@angular/core';
import { Cliente } from './cliente';
import { CLIENTES } from './clientes.json'

@Component({
  selector: 'app-clientes',
  templateUrl: './clientes.component.html',
  styleUrls: ['./clientes.component.css']
})
export class ClientesComponent implements OnInit {

  clientes: Cliente[];
  constructor() { }

  ngOnInit(): void {
    this.clientes = CLIENTES;
  }
}
```

El método ngOnInit se ejecuta cuando se crea el componente.
Al crear el componente se fija la lista de clientes

Visualizar el componente

La tabla donde se listan los clientes se muestra de la siguiente manera



The screenshot shows a web browser window with the title 'ClientesApp'. The address bar displays 'localhost:4200'. The page content is on a blue background and includes a header 'Clientes' and a sub-header 'Listado de clientes'. Below the sub-header is a table with five columns: 'id', 'nombres', 'apellidos', 'email', and 'fecha de creación'. The table contains three rows of client data.

id	nombres	apellidos	email	fecha de creación
1	Juan	Perez	juan@unicauca.edu.co	2021-05-14
2	Andres	Sanchez	andres@unicauca.edu.co	2022-06-14
1	Pedro	Cortez	pedro@unicauca.edu.co	2018-02-14

Aplicación 3

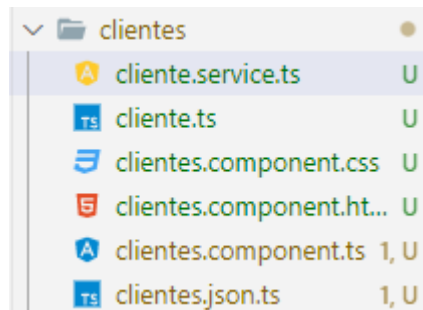
Listar los clientes a partir de un servicio

Vamos a desacoplar la lógica de negocio del componente, para lo cual creamos un servicio de la siguiente forma

ng g service cliente

```
D:\aplicaciones desarrolladas\angular\1. Introducción a los componentes\clientes-app\src\app\clientes>ng g service cliente
CREATE src/app/clientes/cliente.service.spec.ts (362 bytes)
CREATE src/app/clientes/cliente.service.ts (136 bytes)
```

El servicio creado se ve de la siguiente forma



```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class ClienteService {

  constructor() { }
}
```

EL decorador Injectable permite que angular automáticamente cree un objeto de la clase

Los metadatos, significa que el es visible en toda la aplicación.

Listar los clientes a partir de un servicio

Creamos dentro de ClienteServices un método que nos retorna los clientes almacenados en el archivo

```
import { Injectable } from '@angular/core';
import { Cliente } from './cliente';
import { CLIENTES } from './clientes.json';

@Injectable({
  providedIn: 'root'
})
export class ClienteService {

  constructor() { }

  getClientes(): Cliente[]
  {
    return CLIENTES;
  }
}
```

Listar los clientes a partir de un servicio

En el componente inyectamos el servicio y lo utilizamos de la siguiente forma:

Forma 1:

```
import { ClienteService } from './cliente.service';
import { Component, OnInit } from '@angular/core';
import { Cliente } from './cliente';

@Component({
  selector: 'app-clientes',
  templateUrl: './clientes.component.html',
  styleUrls: ['./clientes.component.css']
})
export class ClientesComponent implements OnInit {

  clientes: Cliente[];
  constructor(private objClienteService: ClienteService) { }

  ngOnInit(): void {
    this.clientes = this.objClienteService.getClientes();
  }
}
```

Listar los clientes a partir de un servicio

Otra forma de inyectar el servicio es la siguiente:

Forma 1:

```
import { ClienteService } from './cliente.service';
import { Component, OnInit } from '@angular/core';
import { Cliente } from './cliente';

@Component({
  selector: 'app-clientes',
  templateUrl: './clientes.component.html',
  styleUrls: ['./clientes.component.css']
})
export class ClientesComponent implements OnInit {

  clientes: Cliente[];
  private objClienteService: ClienteService;
  constructor(objClienteService: ClienteService) {
    this.objClienteService = objClienteService;
  }

  ngOnInit(): void {
    this.clientes = this.objClienteService.getClientes();
  }
}
```


Listar los clientes a partir de un servicio

Registramos el servicio en el archivo app.module.ts

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

import { AppComponent } from './app.component';
import { ClientesComponent } from './clientes/clientes.component';
import { ClienteService } from './clientes/cliente.service';

@NgModule({
  declarations: [
    AppComponent,
    ClientesComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [ClienteService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Visualizar el componente

La tabla donde se listan los clientes se muestra de la siguiente manera



The screenshot shows a web browser window with the title 'ClientesApp'. The address bar displays 'localhost:4200'. The page content is on a blue background and includes a header 'Clientes' and a sub-header 'Listado de clientes'. Below the sub-header is a table with five columns: 'id', 'nombres', 'apellidos', 'email', and 'fecha de creación'. The table contains three rows of data.

id	nombres	apellidos	email	fecha de creación
1	Juan	Perez	juan@unicauca.edu.co	2021-05-14
2	Andres	Sanchez	andres@unicauca.edu.co	2022-06-14
1	Pedro	Cortez	pedro@unicauca.edu.co	2018-02-14

Aplicación 4

La programación Reactiva es un paradigma de programación asincrónico interesado en los flujos de datos y la propagación al cambio

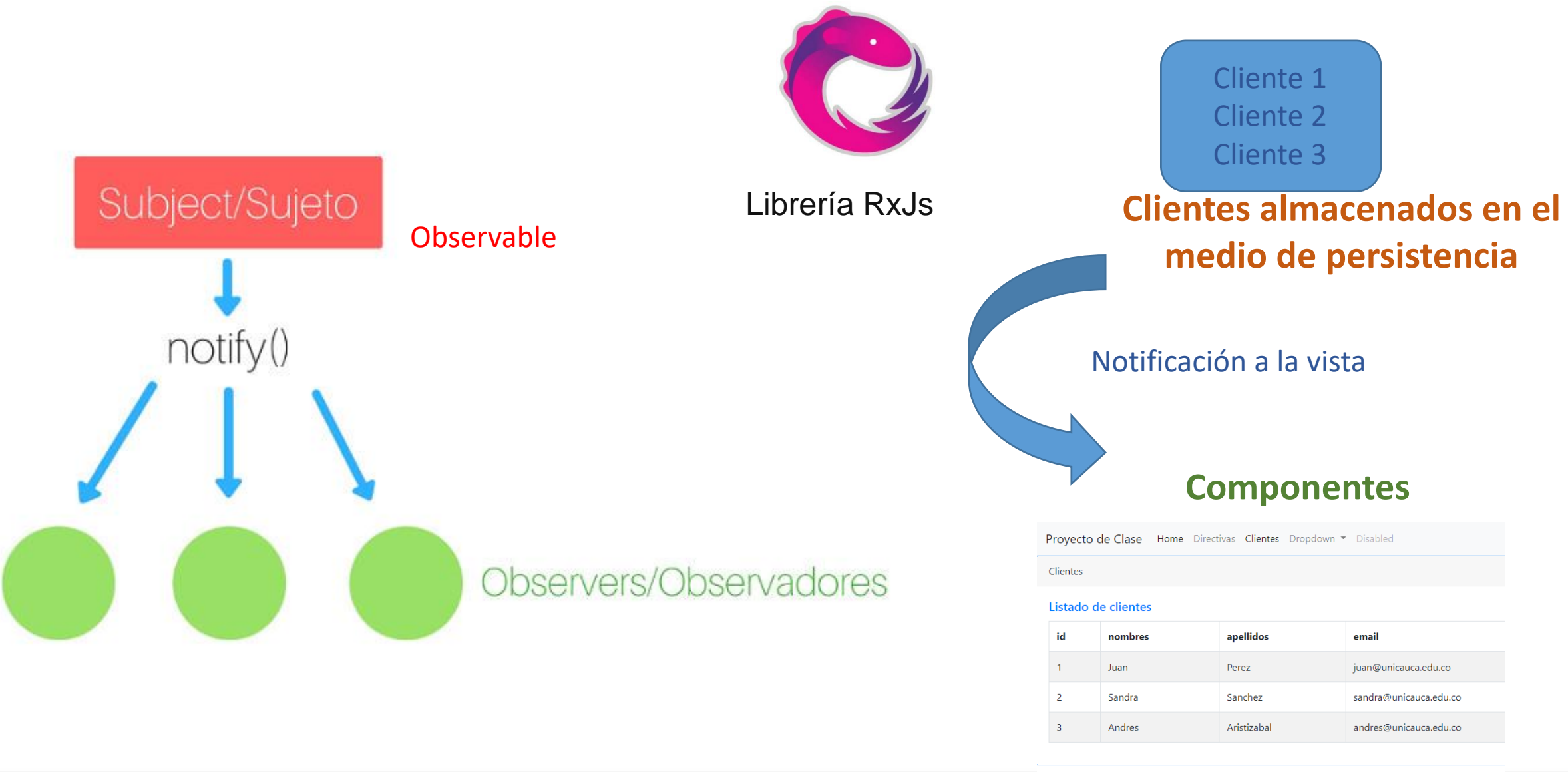


Librería RxJS (Por sus siglas en Inglés, "Reactive Extensions for JavaScript") es una librería para programación reactiva usando observables que hacen más fácil la creación de código asincrono o basado en callbacks

<https://docs.angular.lat/guide/rx-library>

Elementos básicos de RxJS

- Observable:** El flujo de datos, una colección de eventos que se pueden emitir en algún momento.
- Observer:** Un objeto que escucha el flujo de datos y puede actuar sobre los valores que éste emite.
- Subscription:** Representa la suscripción a la ejecución de un observable y permite cancelarla.
- Operador:** Función para manipular los eventos siguiendo los principios de la programación funcional.



Con el fin de gestionar peticiones asíncronas que no bloqueen al cliente a la espera de una petición se utiliza el patrón observador.

Para que el método **getClientes()** de la clase **ClienteService** nos devuelva un **Stream** utilizaremos el API Observable

a) Realizamos los siguientes import:

```
import {Observable} from 'rxjs';  
import {of} from 'rxjs';
```

b) Hacemos que el método **getClientes()** nos devuelva un Observable de clientes

```
getClientes(): Observable<Cliente[]>
```

c) Para convertir el arreglo de clientes en un Observable de clientes utilizamos el operador **of**

```
return of(CLIENTES);
```

```
import { Injectable } from '@angular/core';  
import { CLIENTES } from './clientes.json';  
import { Cliente } from './cliente';
```

```
import { Observable, of } from 'rxjs';
```

```
@Injectable()  
export class ClienteService {
```

```
  getClientes(): Observable<Cliente[]> {  
    return of(CLIENTES);  
  }  
}
```



La vista esta observando un cambio
en el sujeto observado

Proyecto de Clase			
Home Directivas Clientes Dropdown Disabled			
Clientes			
Listado de clientes			
id	nombres	apellidos	email
1	Juan	Perez	juan@unicauca.edu.co
2	Sandra	Sanchez	sandra@unicauca.edu.co
3	Andres	Aristizabal	andres@unicauca.edu.co

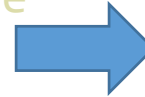
El método será el sujeto observado

El método nos devuelve un flujo de datos

d) Para lograr que a la vista se le notifique un cambio sin necesidad de cambiar la pagina debemos **registrar un observador** de la siguiente forma:

Forma 1:

```
ngOnInit(): void{  
    this.clienteService.getClientes().subscribe  
(  
        clientes => this.clientes = clientes  
    );  
}
```



Función anónima que se encarga de asignar el valor al atributo clientes

Ante un cambio en los datos del cliente la vista se actualiza automáticamente:

Forma 2:

```
this.clienteService.getClientes().subscribe(  
    function(clientes)  
    {  
        this.clientes = clientes;  
    }  
);
```



```
import { ClienteService } from './cliente.service';
import { Component, OnInit } from '@angular/core';
import { Cliente } from './cliente';

@Component({
  selector: 'app-clientes',
  templateUrl: './clientes.component.html'
})
export class ClientesComponent implements OnInit {
  clientes: Cliente[] = [];
  private clienteService: ClienteService;

  constructor(clienteService: ClienteService) {
    this.clienteService = clienteService;
  }

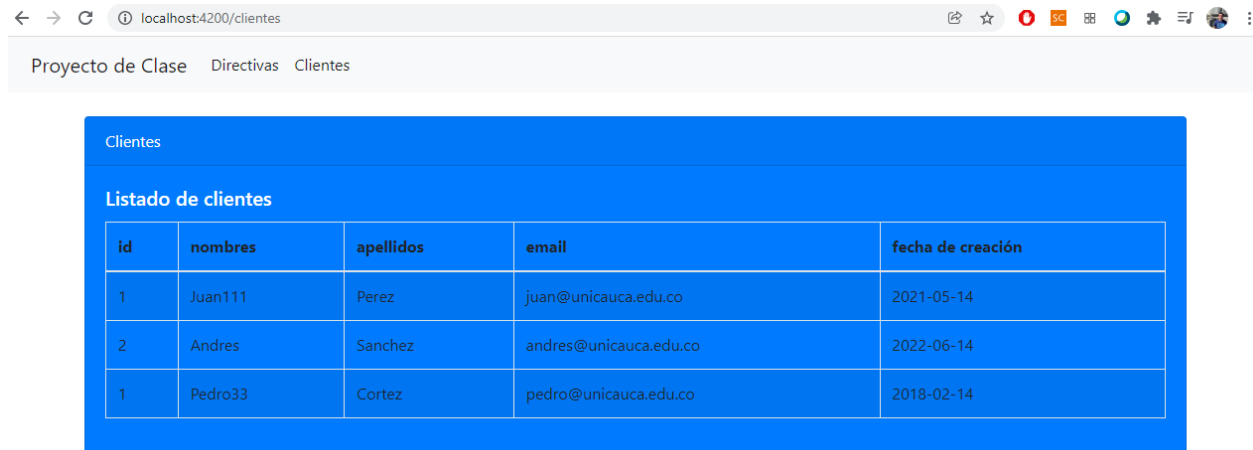
  ngOnInit(): void {
    this.clienteService.getClientes().subscribe(
      clientes => this.clientes = clientes
    );
  }
}
```

Atributo donde se almacena el flujo de datos proveniente del método observado.

Estamos suscribiendo al componente como observador

El flujo proveniente el método observado se almacena en el atributo clientes:

La vista únicamente cambiará si los datos almacenados de los clientes cambian.



The screenshot shows a web browser window with the address bar displaying 'localhost:4200/clientes'. Below the address bar is a navigation bar with three items: 'Proyecto de Clase', 'Directivas', and 'Clientes'. The main content area has a blue header with the text 'Clientes'. Below this header is a section titled 'Listado de clientes' which contains a table with five columns: 'id', 'nombres', 'apellidos', 'email', and 'fecha de creación'. The table lists three clients.

id	nombres	apellidos	email	fecha de creación
1	Juan111	Perez	juan@unicauca.edu.co	2021-05-14
2	Andres	Sanchez	andres@unicauca.edu.co	2022-06-14
1	Pedro33	Cortez	pedro@unicauca.edu.co	2018-02-14

Mediante las rutas podemos dividir la aplicación en diferentes secciones o contenidos. No son paginas aisladas, solo tenemos una sola pagina que permite mostrar secciones o contenidos.

a) En el archivo **app.modules.ts** debemos realizar los siguientes dos import:

```
import {RouterModule, Routes} from '@angular/router';
```

b) Debemos crear un **arreglo de rutas** de la siguiente forma:

```
const routes: Routes = [  
  {path: '', redirectTo: '/clientes', pathMatch: 'full'},  
  {path: 'directivas', component: DirectivaComponent},  
  {path: 'clientes', component: ClientesComponent}  
];
```



La URL "" representa el home y dirige al componente Clientes



La URL "clientes" esta mapeado al componente Clientes

c) Debemos registrar las rutas utilizando el **routerModule**:

```
imports: [  
  BrowserModule,  
  RouterModule.forRoot(routes) ],
```

d) Para indicar donde se van a redenderizar las rutas debemos ir al archivo app.component.html y utilizar la directiva:

```
<router-outlet> </router-outlet>
```

Es una directiva utilizada por angular para determinar donde se va a renderizar el contenido de cada ruta.

El archivo app.component.html queda de la siguiente manera:

```
<app-header> </app-header>  
  
<router-outlet> </router-outlet>  
  
<app-footer></app-footer>
```

e) Debemos modificar el menú con el fin de redireccionar al usuario al contenido de cada ruta.

Debemos dirigirnos al header.Component.html y agregar los siguientes elementos al menu:

```
<li class="nav-item" routerLinkActive="active">
  <a class="nav-link" routerLink ="/directivas">Directivas</a>
</li>
<li class="nav-item" routerLinkActive="active">
  <a class="nav-link" routerLink ="/clientes">Clientes</a>
</li>
```

La directiva **routerLink** asocia una ruta a un componente

Cada vez que se selecciona una ruta la directiva **routerLinkActive** permite marcar el elemento del menu como seleccionado.

Rutas

Al dar click se muestra el contenido asociado al componente clientes

Proyecto de Clase

Home

Directivas

Cientes

Dropdown

led

Search

Search


Cientes

Listado de clientes

id	nombres	apellidos	email	fecha de creación
1	Juan	Perez	juan@unicauca.edu.co	2018-12-11
2	Sandra	Sanchez	sandra@unicauca.edu.co	2019-12-11
3	Andres	Aristizabal	andres@unicauca.edu.co	2020-12-11

Rutas

Proyecto de Clase Home **Directivas** Cliente Down ▾ Disabled



Al dar click se muestra el contenido asociado al componente directivas

Ocultar

Listado de Categorías
Ventas
Contabilidad
Transporte
Informática
Contabilidad
Construcción

Muchas gracias
Preguntas

