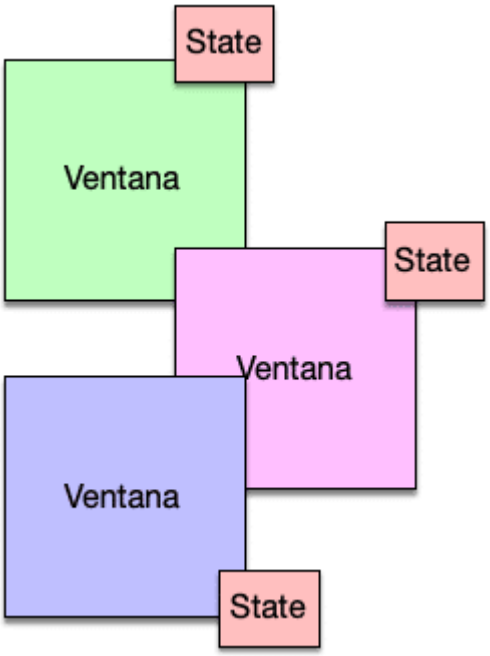
Tabla de Contenidos

- Angular Routing y estado
- Angular Stateful Routing y estrategia
- Router State y Custom Strategy
- Otros artículos relacionados

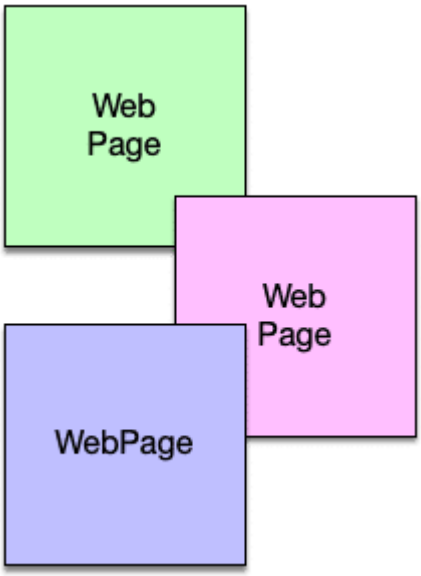
El concepto de Angular Stateful routing es un concepto que aparece en muchas ocasiones cuando deseamos construir una aplicación con Angular pero migrando probablemente arquitecturas antiguas como pueden ser aplicaciones de escritorio. Normalmente cuando se trata de una aplicación de escritorio esta esta compuesta de un conjunto de ventanas que de una forma u otra se conectan al servidor y obtienen los datos.

Angular Routing y estado

El construir una aplicación que se conecte a un servidor web y consuma servicios REST con Angular es algo muy sencillo ya que la arquitectura de Angular esta orientado a ello . Ahora bien las aplicaciones de escritorio normalmente tienen algunas cosas extras que pueden hacer más compleja la situación . Una de esas cosas es el mantenimiento de estado .



Es decir una aplicación web de forma natural no mantiene el estado en su navegación cada página que se carga es por defecto Stateless .



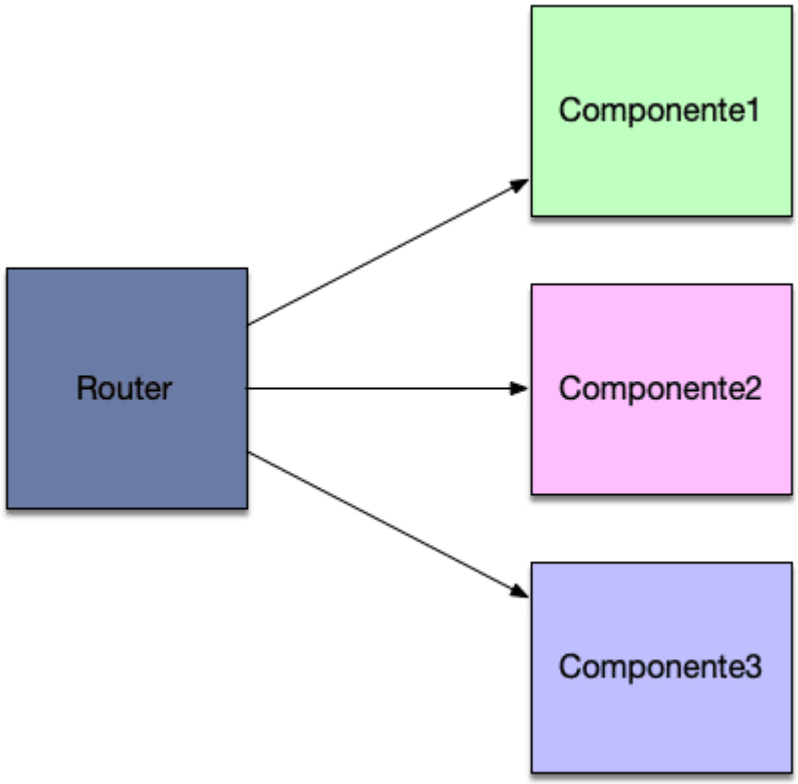
Vamos a ver un ejemplo sencillo usando Angular y su router .Supongamos que tenemos construidos los siguientes componentes.

```
<p>
nombre:
  <input type="text" name="nombre" [(ngModel)]="nombre"/>

</p>

<p>
apellidos
  <input type="text" name="apellidos" [(ngModel)]="apellidos"/>
</p>

<p>
 edad
    <input type="text" name="edad" [(ngModel)]="edad"/>
</p>
```

Estos tres componentes cada uno define un pequeño formulario con una caja de texto . Es momento de echar un vistazo a la estructura del Router y ver como navegamos entre las distintas vistas

```
const rutas: Routes = [
  { path: 'c1', component: C1Component },
  { path: 'c2', component: C2Component },
  { path: 'c3', component: C3Component },
 { path: '',   redirectTo: '/c1', pathMatch: 'full' },
];
```

Como podemos observar es una navegación elemental que nos permite movernos entre los distintos componentes.

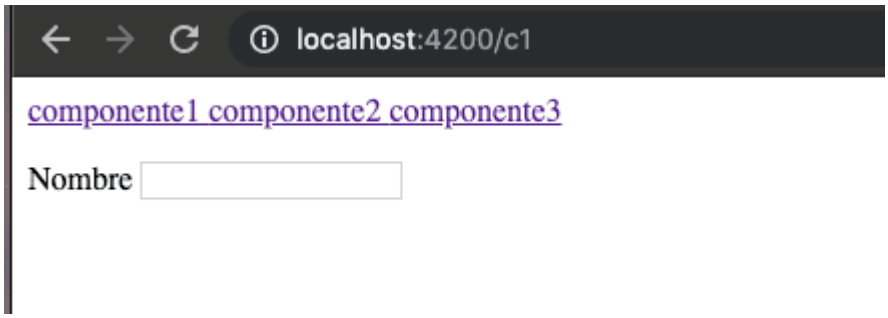


Nos queda echar un vistazo al código html del propio componente principal que define como el enrutado funciona a nivel visual.

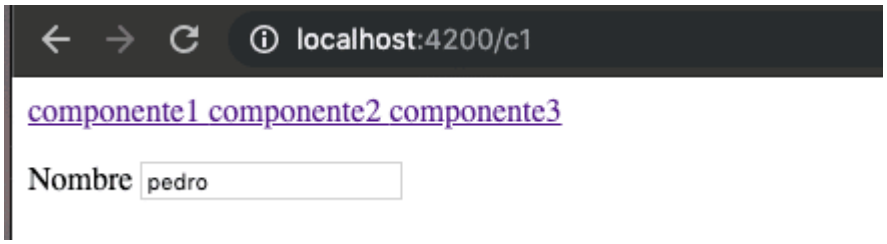
```
<a [routerLink]="['/c1']">
componente1
</a>

<a [routerLink]="['/c2']">
componente2
</a>
<a [routerLink]="['/c3']">
componente3
</a>

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

Este código nos facilita la navegación entre los distintos componentes o vistas de la página.

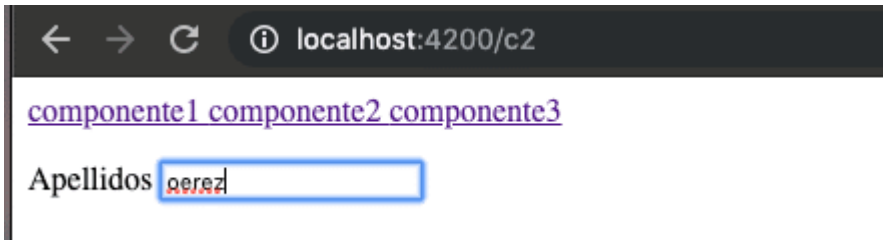
De tal forma que al navegar veremos algo similar a lo siguiente:



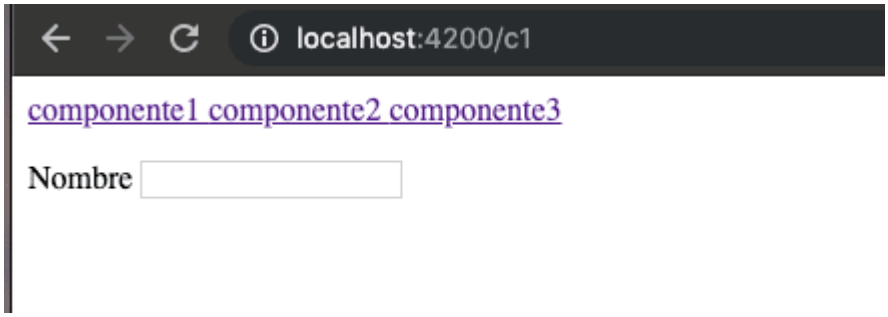
Nos puede parecer un asistente y rellenar cada dato según vamos cambiando de pestaña. El problema le tendremos que una vez rellenado el nombre:



Podemos pasar a la segunda pestaña y rellenar el apellido :



Lamentablemente si volvemos a la pagina anterior habremos perdido el estado que almacenaba el componente1 .



Esto es un problema importante en arquitecturas que parten de enfoques de aplicaciones de escritorio en donde el cliente almacena un estado importante

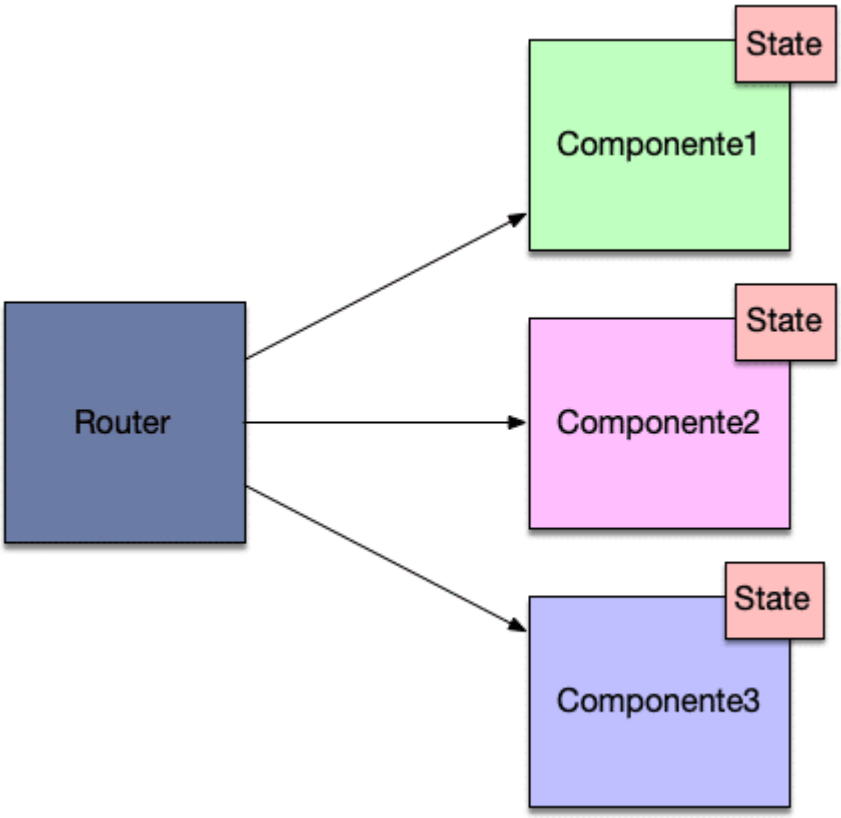
Angular Stateful Routing y estrategia

¿Cómo podemos solventar esto? . Para ello debemos modificar el sistema de estrategia que tiene el propio router al cargarse creando una nueva clase que gestione el como el router recarga los componentes.

```
import { RouteReuseStrategy, DetachedRouteHandle,
ActivatedRouteSnapshot } from '@angular/router';

export class CustomStateStrategy implements RouteReuseStrategy {

    handlers: {[key: string]: DetachedRouteHandle} = {};

    shouldDetach(route: ActivatedRouteSnapshot): boolean {

      return true;
    }
    store(route: ActivatedRouteSnapshot, handle:DetachedRouteHandle):
void {
        this.handlers[route.routeConfig.path] = handle;
    }
```

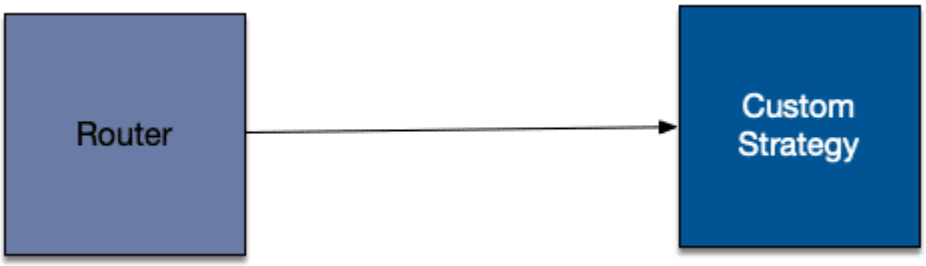
```
    shouldAttach(route: ActivatedRouteSnapshot): boolean {
        return !!route.routeConfig &&
!!this.handlers[route.routeConfig.path];
    }
    retrieve(route: ActivatedRouteSnapshot): DetachedRouteHandle {

        if (!route.routeConfig) return null;
        return this.handlers[route.routeConfig.path];

    }
    shouldReuseRoute(future:ActivatedRouteSnapshot, curr:
ActivatedRouteSnapshot): boolean {

        return future.routeConfig === curr.routeConfig;
    }

}
```

Esta clase se encarga de definir una configuración de estado a través del uso de una cache en cuando a la estrategia del router.



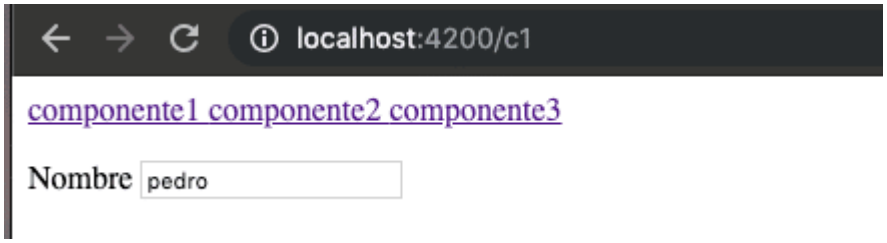
Es decir el router preguntará a esta clase si el componente ya existe instanciado en esta cache y en tal caso lo devolverá.

Para ello se apoya en el método store que va almacenando cada uno de los componentes creados.



Router State y Custom Strategy

Una vez hecho esto los componentes serán capaces de mantener el estado mientras navegamos. Es decir si rellenamos el nombre y pasamos al apellido , al volver hacia atrás el componente seguirá vivo.



CURSO TYPESCRIPT
GRATIS
APUNTATE!!

Otros artículos relacionados

- Angular Lazy Loading Modules
- Angular Services
- Angular Modules
- Angular Router