

Técnicas adicionales de navegación y enrutamiento

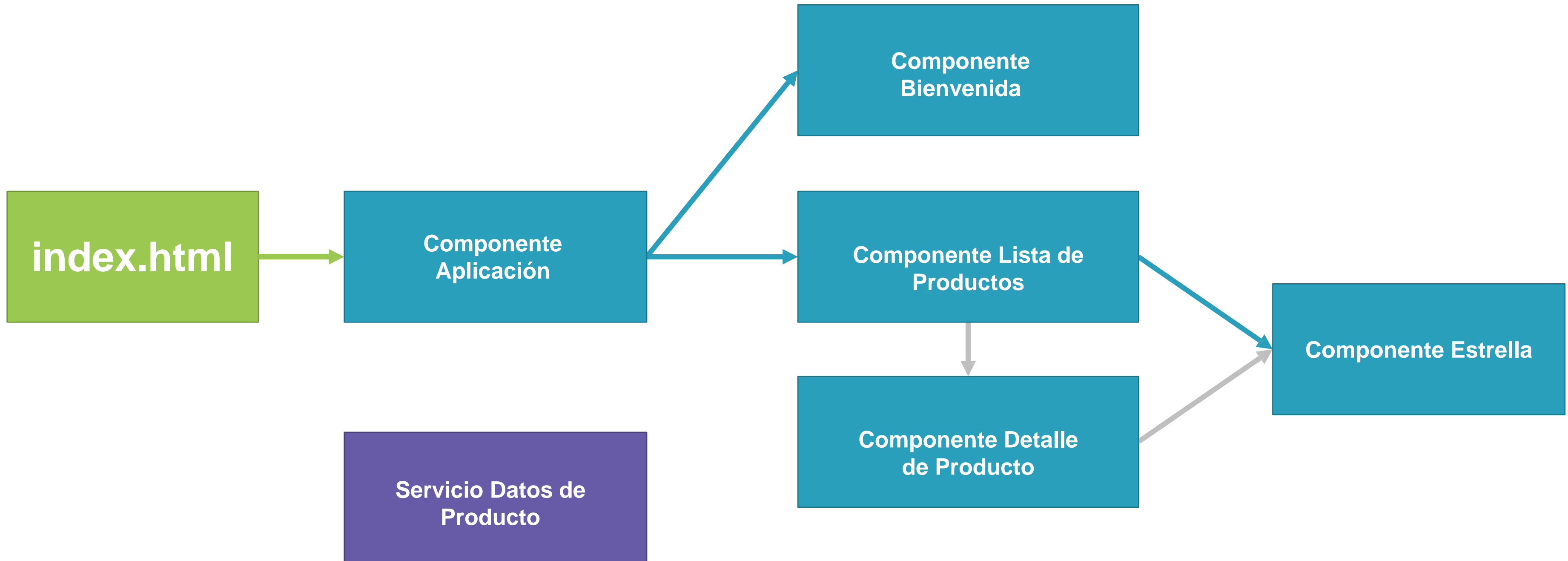


Resumen

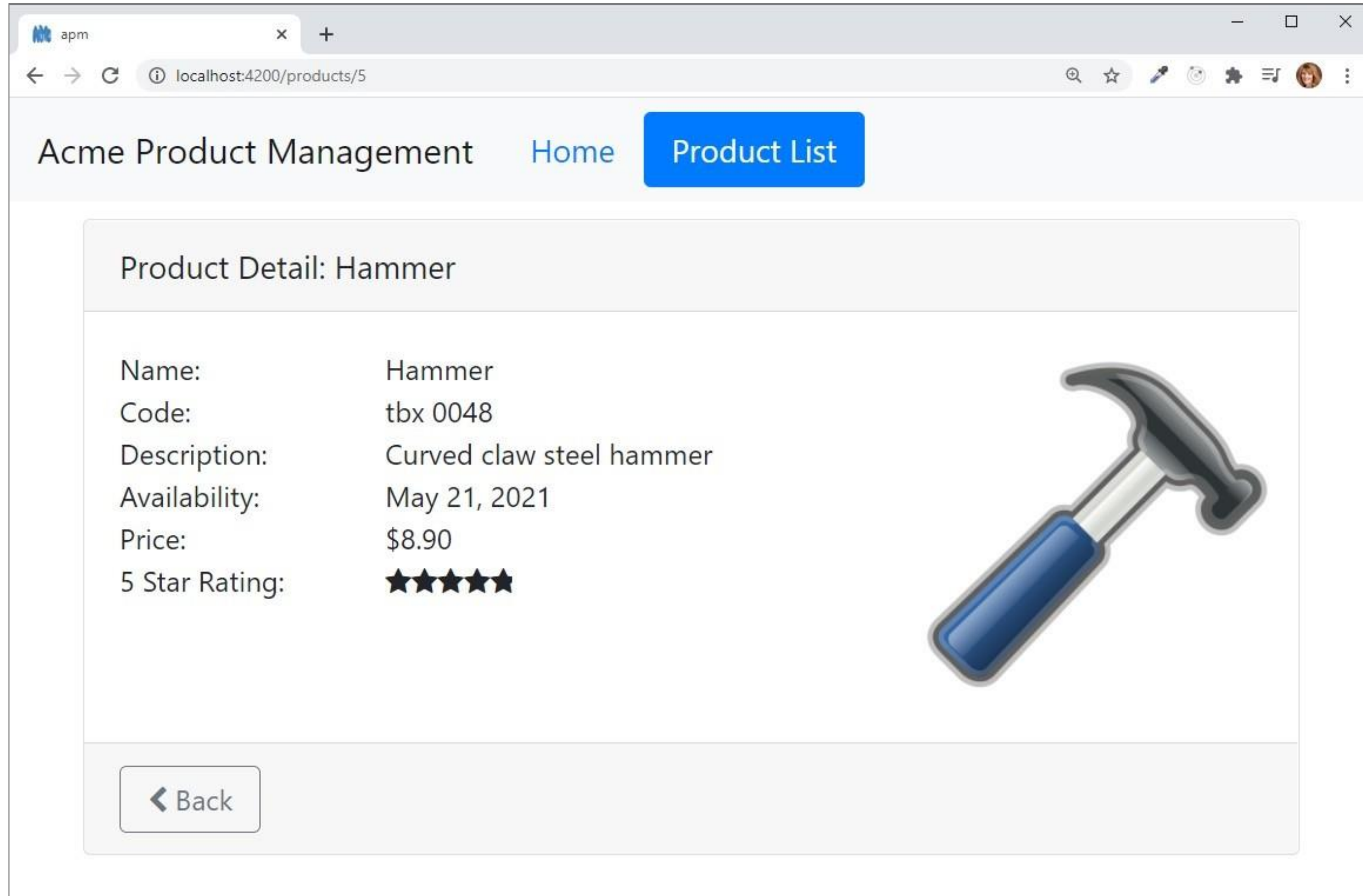


- **Pasar parámetros a una ruta**
- **Activación de una ruta con código**
- **Proteger las rutas con guardas**

Application Architecture



Pasar parámetros a una ruta

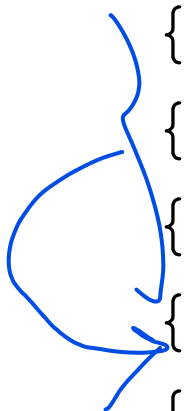


Pasar parámetros a una ruta

app.module.ts

```
@NgModule({
  imports: [
    ...,
    RouterModule.forRoot([
      { path: 'products', component: ProductListComponent },
      { path: 'products/:id', component: ProductDetailComponent },
      { path: 'welcome', component: WelcomeComponent },
      { path: '', redirectTo: 'welcome', pathMatch: 'full' },
      { path: '**', redirectTo: 'welcome', pathMatch: 'full' }
    ])
  ],
  declarations: [...],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

cualquier cosa que no sea eso, al welcome



Pasar parámetros a una ruta

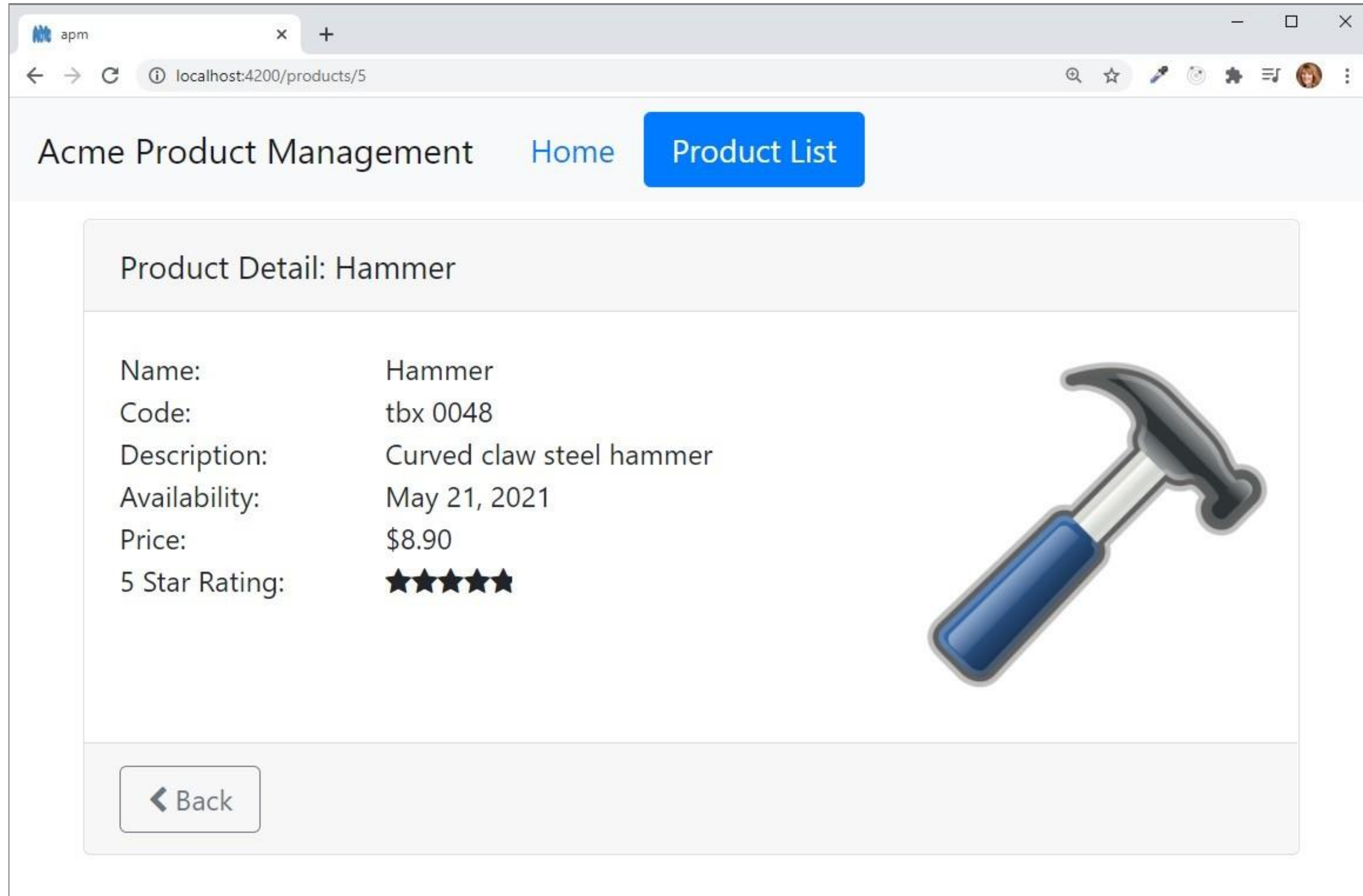
product-list.component.html

```
<td>
  <a [routerLink]="['/products', product.productId]">
    {{product.productName}}
  </a>
</td>
```

app.module.ts

```
{ path: 'products/:id', component: ProductDetailComponent }
```

Lectura de parámetros de una ruta



Lectura de parámetros de una ruta

product-detail.component.ts

```
import { ActivatedRoute } from '@angular/router';  
  
constructor(private route: ActivatedRoute) { }
```

app.module.ts

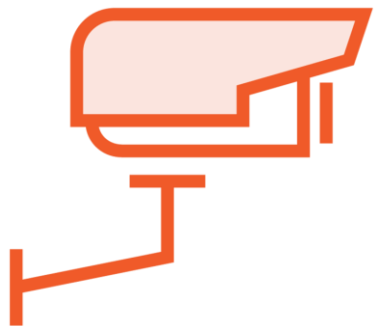
```
{ path: 'products/:id', component: ProductDetailComponent }
```

Lectura de parámetros de una ruta



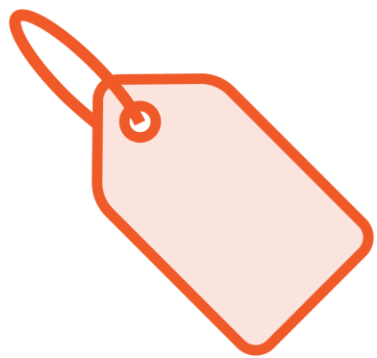
Instantánea : Leer el parámetro una vez

```
this.route.snapshot paramMap.get('id');
```



Observable: Leer los parámetros emitidos a medida que cambian

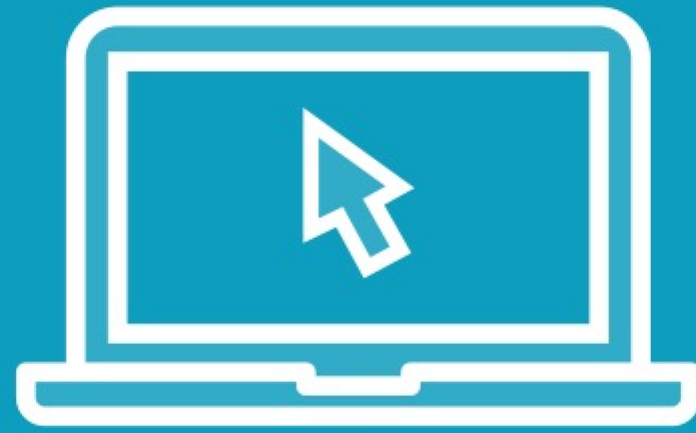
```
this.route.paramMap.subscribe(  
  params => console.log(params.get('id'))  
);
```



La cadena especificada es el nombre del parámetro de la ruta

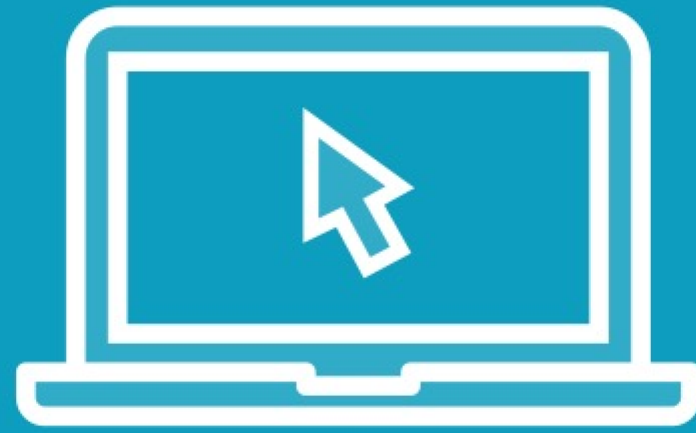
```
{ path: 'products/:id',  
  component: ProductDetailComponent }
```


Demo



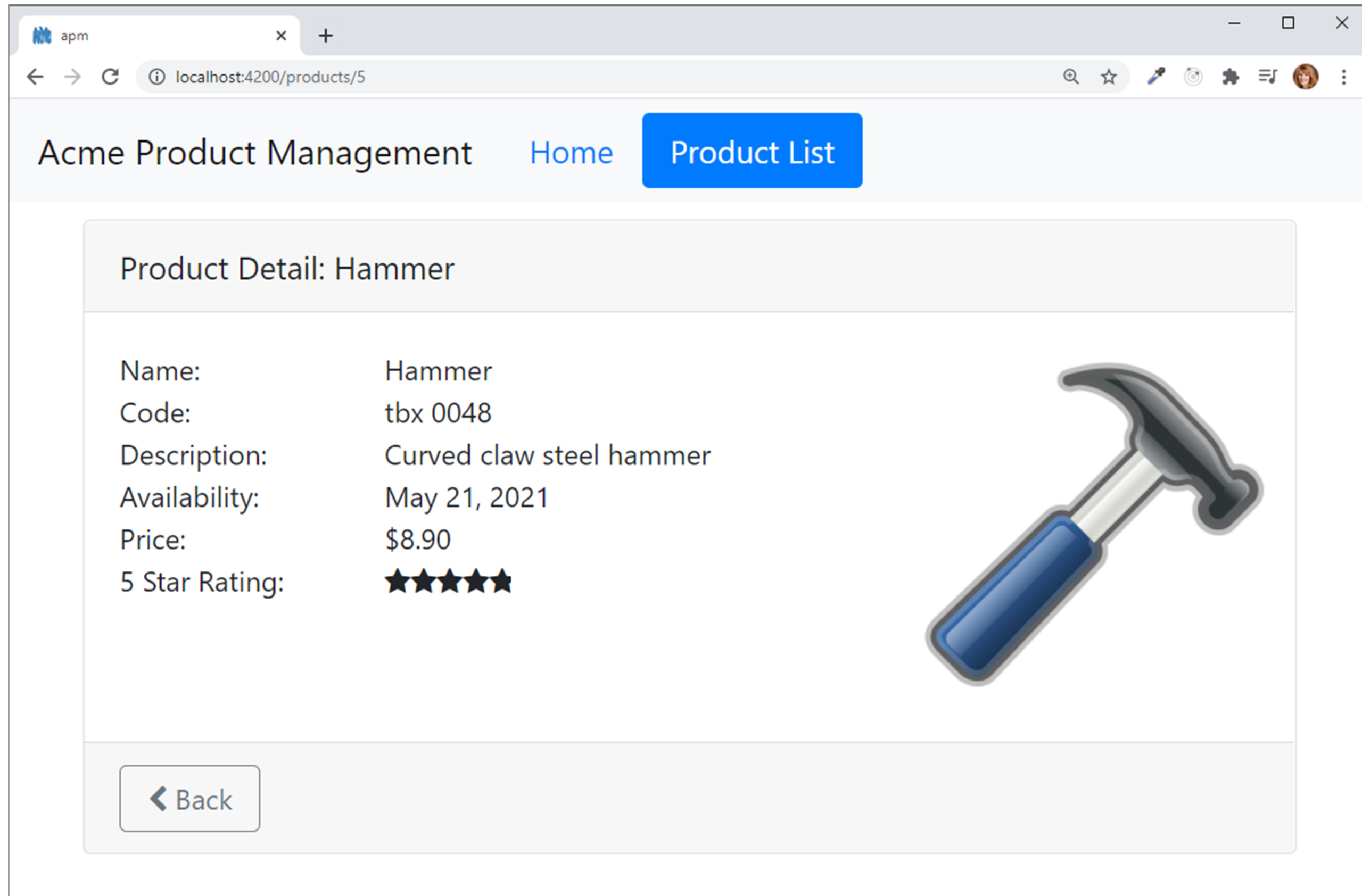
Pasar parámetros a una ruta

Demo



Tratamiento de null y undefined

Activación de una ruta mediante código



Activación de una ruta mediante código

product-detail.component.ts

```
import { Router } from '@angular/router';  
...  
constructor(private router: Router) { }  
  
onBack(): void {  
    this.router.navigate(['/products']);  
}
```


Proteger las rutas con guardas



Limitar el acceso a una ruta

parea que un usuario no pueda
acceder al historial de otros



Restringir el acceso sólo a determinados usuarios



Pedir confirmación antes de navegar

Proteger las rutas con guardas



CanActivate

- Guarda de navegación a una ruta

CanDeactivate

- Guarda de navegación desde una ruta

Resolve

- Precargar datos antes de activar una ruta

CanLoad

- Evitar el enrutamiento asíncrono

Crear una guarda

product-detail.guard.ts

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

@Injectable({
  providedIn: 'root'
})
export class ProductDetailGuard implements CanActivate {

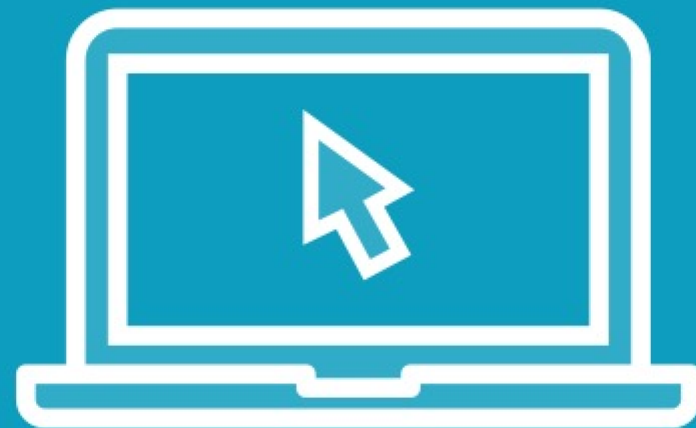
  canActivate(): boolean {
    ...
  }
}
```

Uso de una guarda

app.module.ts

```
@NgModule({
  imports: [
    ...,
    RouterModule.forRoot([
      { path: 'products', component: ProductListComponent },
      { path: 'products/:id',
        canActivate: [ ProductDetailGuard ],
        component: ProductDetailComponent },
      ...])
  ],
  declarations: [...],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Demo



Proteger rutas con guardas

Lista de comprobación general: null y undefined

Evitar errores null o undefined en una plantilla

- **Operador de navegación segura (?)**

```
{{product?.productName}}
```

```
{{product?.supplier?.companyName}}
```

- **Directiva *ngIf**

```
<div *ngIf='product'>  
  <div>Name:</div>  
  <div>{{product.productName}}</div>  
  <div>Description:</div>  
  <div>{{product.description}}</div>  
</div>
```



Lista de comprobación de enrutamiento: Paso de parámetros

app.module.ts

```
{ path: 'products/:id', component: ProductDetailComponent }
```

product-list.component.html

```
<a [routerLink]="['/products', product.productId]">
  {{product.productName}}
</a>
```

product-detail.component.ts

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

constructor(private route: ActivatedRoute) {
  console.log(this.route.snapshot paramMap.get('id'));
}
```



Lista de comprobación de enrutamiento: Activar una ruta mediante código

Utilizar el servicio de router

- Definirlo como una dependencia

Crear un método que llame al método navigate del servicio Router

- Pasar la matriz de parámetros del enlace

```
import { Router } from '@angular/router';  
...  
constructor(private router: Router) { }  
  
onBack(): void {  
    this.router.navigate(['/products']);  
}
```

Añadir un elemento de la interfaz de usuario

- Utiliza la vinculación de eventos para enlazar con el método creado

```
<button (click)='onBack()'>Back</button>
```



Lista de comprobación de enrutamiento: Proteger rutas con guardas

Crear un servicio de guarda

- Implementar el tipo guarda (CanActivate)
- Crear el método (canActivate())

Registrar el proveedor de servicios de guarda

- Utilizar la propiedad providedIn



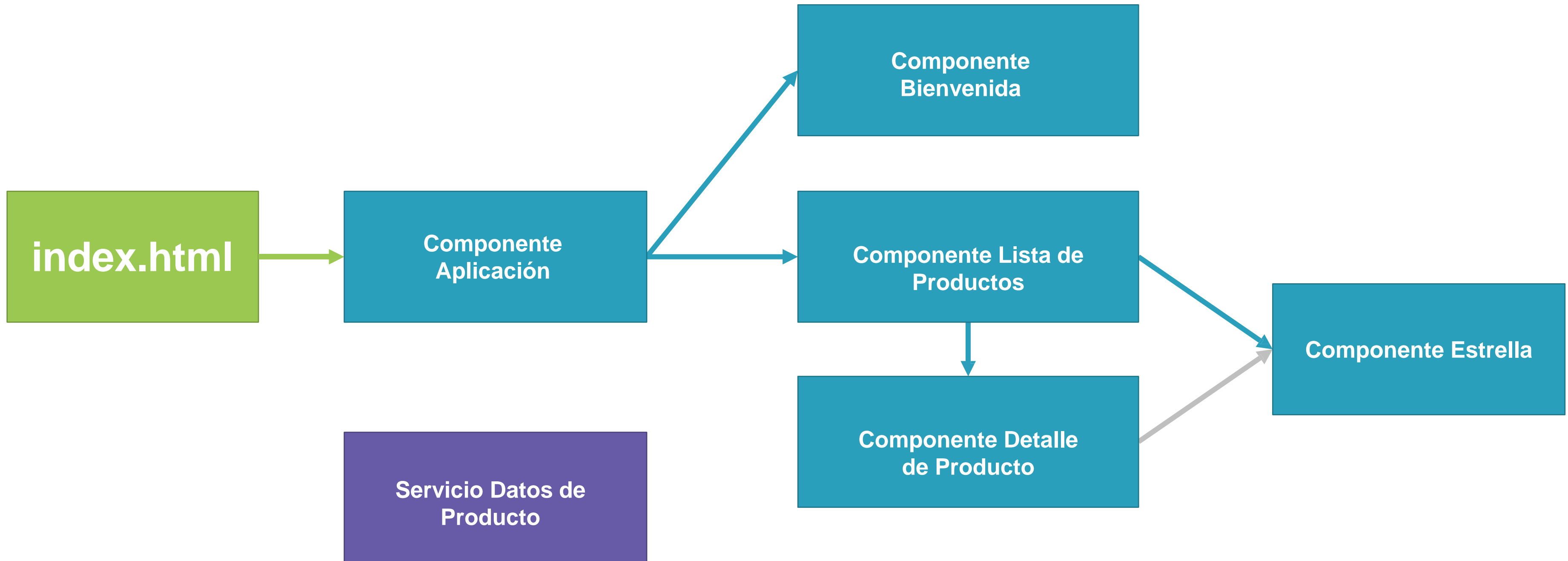
```
import { Injectable } from '@angular/core';
import { CanActivate } from '@angular/router';

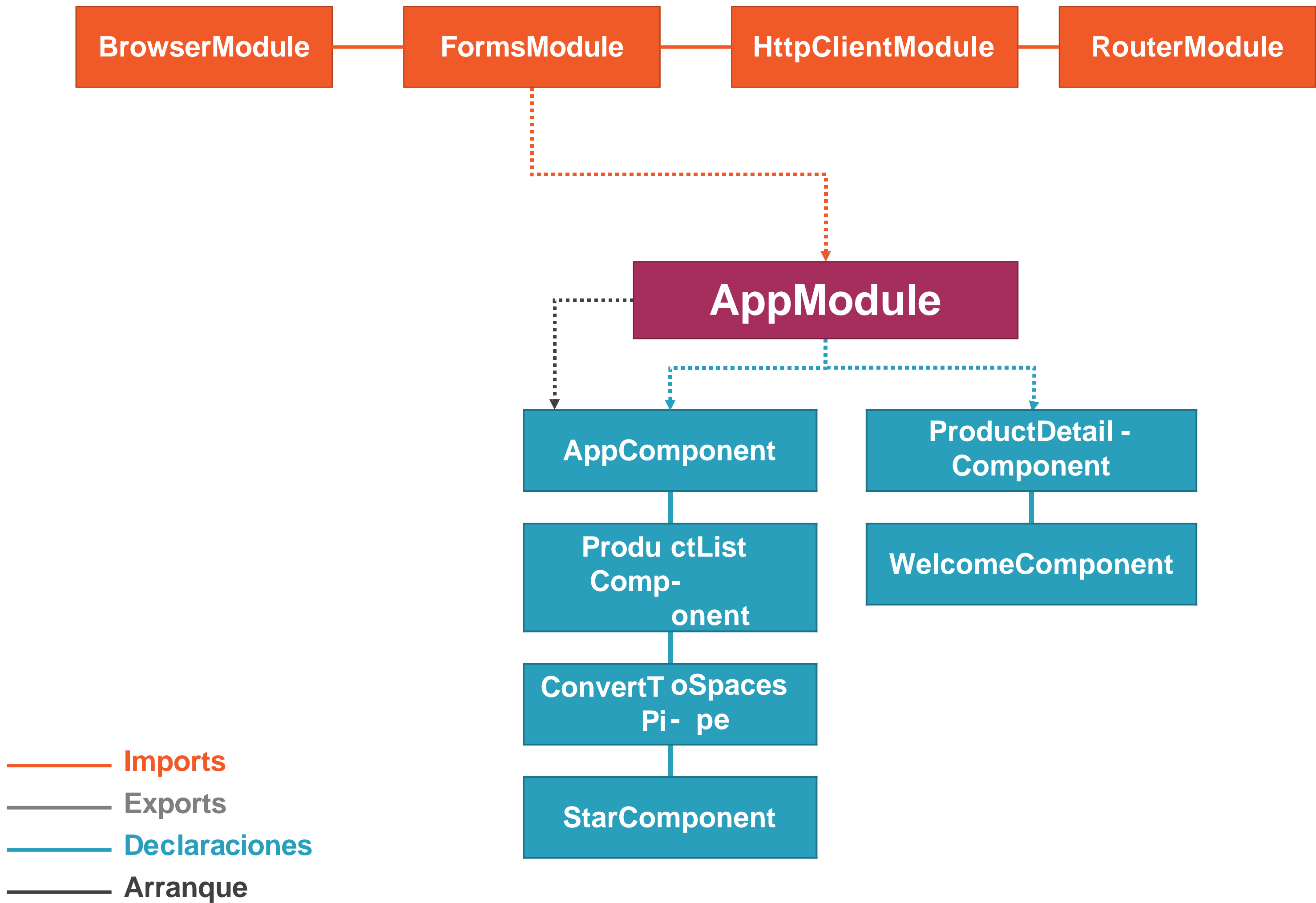
@Injectable({ providedIn: 'root' })
export class ProductDetailGuard implements CanActivate {
  canActivate(): boolean { ... }
}
```

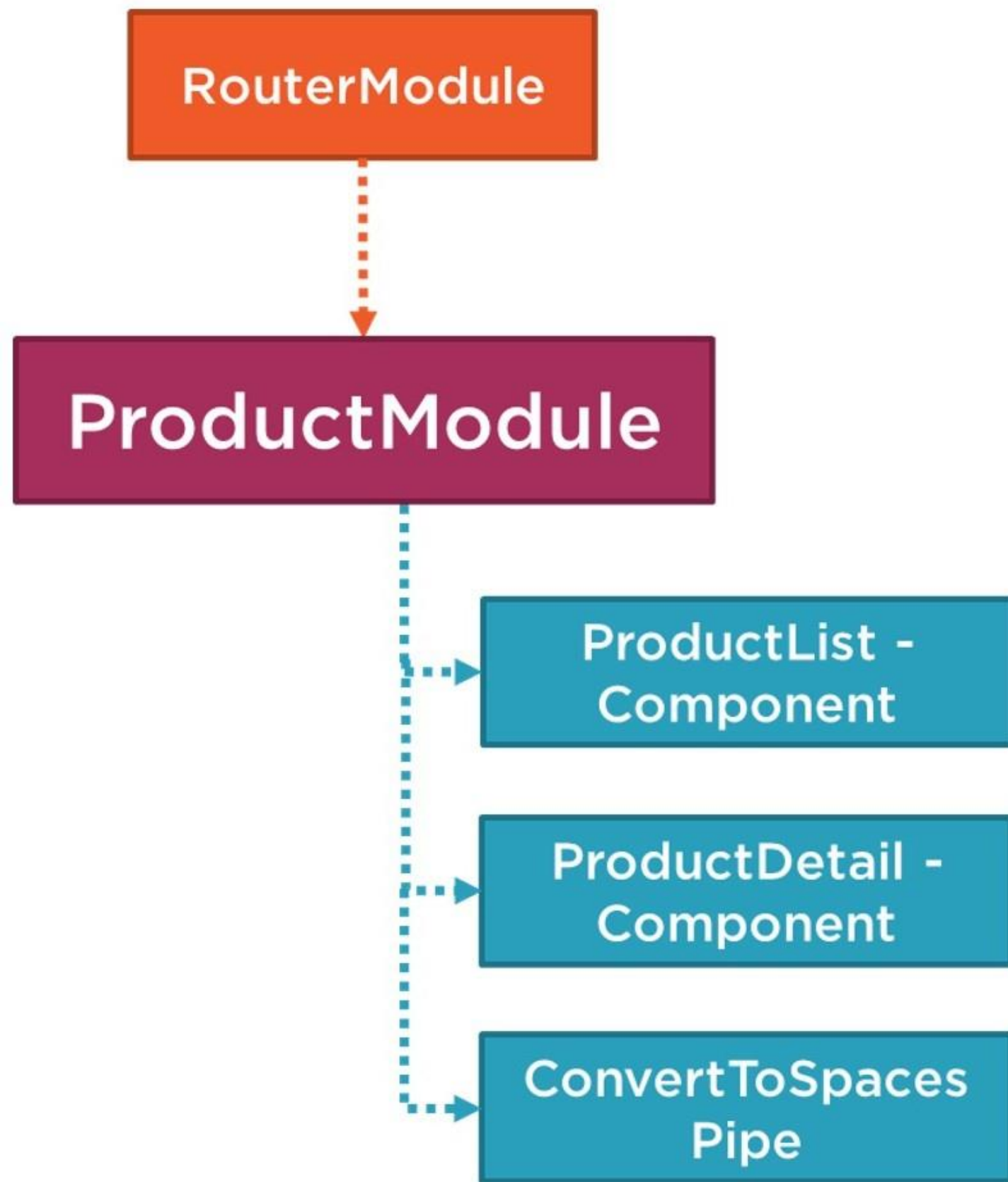
Agregar la guarda a la ruta

```
{ path: 'products/:id', canActivate: [ ProductDetailGuard ],
  component: ProductDetailComponent },
```


Arquitectura de la aplicación







A continuación ...

Módulos Angular