 <div data-bbox="437 152 531 226"> ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO </div>	Programação em Ambiente Web Docentes – FAS, EFE, JCarneiro, OAO Ficha Prática 9
---	---

Documentação de Apoio

- Slides disponíveis na plataforma moodle da Unidade Curricular
- Angular - <https://angular.io/>
- Repositórios git para apoio à resolução dos exercícios

1 Gestão de produtos em Angular

Este exercício pretende iniciar de uma forma simples um projeto Angular que permita consumir uma REST API. Tendo por base a API para Controlo de Produtos da ficha prática 7, desenvolva uma aplicação escrita em angular para consumir a API desenvolvida.

Desta forma devemos começar por criar um projeto em angular com o comando:

- `ng new angular-product-app`

Vamos de seguida definir uma classe para encapsular produtos num ficheiro “product.ts” dentro da pasta “app/Models”

Table 1 - Product.ts

```
export class Product {
  _id : String;
  name: String;
  description: String;
  quantity: Number;
}
```

De seguida vamos criar o serviço que irá interagir com a API Rest de produtos da ficha prática 7. Para o efeito iremos executar o comando:

- `ng generate service rest`

Deverá completar o ficheiro “rest.service.ts” criado para se aproximar

Table 2 - rest.service.ts

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

import { HttpClient, HttpHeaders, HttpResponse } from '@angular/common/http';
import { Observable, of } from 'rxjs';
import { map, catchError, tap } from 'rxjs/operators';

import { Product } from './Models/Product';

const endpoint = 'http://localhost:3000/api/v1/';
const httpOptions = {
  headers: new HttpHeaders({
    'Content-Type': 'application/json'
  })
};

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

  constructor(private http: HttpClient) {}

  private extractData(res: Response) {
    let body = res;
    return body || {};
  }
}
```

```

getProducts(): Observable<Product[]> {
  return this.http.get<Product[]>(endpoint + 'products');
}

getProduct(id:String): Observable<Product> {
  return this.http.get<Product>(endpoint + 'product/' + id)
}

addProduct (product:Product): Observable<Product> {
  console.log(product);
  return this.http.post<Product>(endpoint + 'products', JSON.stringify(product), httpOptions );
}

updateProduct (id:string, product:Product): Observable<Product> {
  return this.http.put(endpoint + 'product/' + id, JSON.stringify(product), httpOptions);
}

deleteProduct (id:string): Observable<Product> {
  return this.http.delete<Product>(endpoint + 'product/' + id, httpOptions);
}
}

```

Vamos agora criar os componentes para adicionar/listar/editar produtos. Iremos também adicionar uma rota para ver os detalhes do produto. Desta forma vamos começar por criar os respetivos componentes com os comandos:

- ng generate component product
- ng generate component product-add
- ng generate component product-detail
- ng generate component product-edit

Após executar os comandos devem ter sido criados 4 novas subpastas com os componentes. É necessário agora actualizar o controller e a view html de cada componente.

Componente product

É necessário actualizar os ficheiros “product.component.ts” e “product.component.html” de acordo com os exemplos fornecidos.

Table 3 - product.component.ts

```

import { Component, OnInit } from '@angular/core';
import { RestService } from '../rest.service';
import { ActivatedRoute, Router } from '@angular/router';

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

  products:any = [];

  constructor(public rest:RestService, private route: ActivatedRoute, private router: Router) { }

  ngOnInit() {
    this.getProducts();
  }

```

```

}

getProducts() {
  this.products = [];
  this.rest.getProducts().subscribe((data: {}) => {
    console.log(data);
    this.products = data;
  });
}

add() {
  this.router.navigate(['/product-add']);
}

delete(id) {
  this.rest.deleteProduct(id)
    .subscribe(res => {
      this.getProducts();
    }, (err) => {
      console.log(err);
    });
}
}

```

Table 4 - product.component.html

```

<h2>Product List</h2>

<div>
  <button (click)="add()">
    Add
  </button>
</div>

<ul class="products">
  <li *ngFor="let p of products; let i=index;">
    <a routerLink="/product-details/{{p._id}}">
      <span class="badge">{{i+1}}</span> {{p.name}}
    </a>
    <button class="delete" title="delete product"
      (click)="delete(p._id)">x</button>
  </li>
</ul>

```

Componente product-add

É necessário atualizar os ficheiros “product-add.component.ts” e “product-add.component.html” de acordo com os exemplos fornecidos.

Table 5 - product-add.component.ts

```

import { Component, OnInit, Input } from '@angular/core';

import { RestService } from '../rest.service';
import { ActivatedRoute, Router } from '@angular/router';

import { Product } from '../Models/Product';

@Component({
  selector: 'app-product-add',

```

```

templateUrl: './product-add.component.html',
styleUrls: ['./product-add.component.css']
})
export class ProductAddComponent implements OnInit {

  @Input() productData : Product = new Product();

  constructor(public rest:RestService, private route: ActivatedRoute, private router: Router) { }

  ngOnInit() {
  }

  addProduct() {
    this.rest.addProduct(this.productData).subscribe((result : Product) => {
      console.log(result);
      this.router.navigate(['/']);
    }, (err) => {
      console.log(err);
    });
  }
}

```

Table 6 - product-add.component.html

```

<div>
  <h2>Product Add</h2>
  <div>
    <label>Product Name:
      <input [(ngModel)]="productData.name" placeholder="Product Name"/>
    </label> <br>
    <label>Product Desc:
      <input [(ngModel)]="productData.description" placeholder="Product Description"/>
    </label> <br>
    <label>Product Quantity:
      <input [(ngModel)]="productData.quantity" placeholder="Product Quantity"/>
    </label> <br>
  </div>
  <button (click)="addProduct()">Save</button>
</div>

```

Componente product-edit

É necessário atualizar os ficheiros “product-edit.component.ts” e “product-edit.component.html” de acordo com os exemplos fornecidos.

Table 7 - product-edit.component.ts

```

import { Component, OnInit, Input } from '@angular/core';

import { RestService } from '../rest.service';
import { ActivatedRoute, Router } from '@angular/router';

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

  @Input() productData:any = { name: "", description: "", quantity:0 };

```

```

constructor(public rest:RestService, private route: ActivatedRoute, private router: Router) { }

ngOnInit() {
  this.rest.getProduct(this.route.snapshot.params['id']).subscribe((data: {}) => {
    console.log(data);
    this.productData = data;
  });
}

updateProduct() {
  this.rest.updateProduct(this.route.snapshot.params['id'], this.productData).subscribe((result) => {
    this.router.navigate(['/product-details/' + result._id]);
  }, (err) => {
    console.log(err);
  });
}
}

```

Table 8 - product-edit.component.html

```

<div>
  <h2>Product Edit</h2>
  <div>
    <label>Product Name:
      <input [(ngModel)]="productData.name" placeholder="Product Name"/>
    </label> <br>
    <label>Product Desc:
      <input [(ngModel)]="productData.description" placeholder="Product Description"/>
    </label> <br>
    <label>Product Quantity:
      <input [(ngModel)]="productData.quantity" placeholder="Product Quantity"/>
    </label> <br>
  </div>
  <button (click)="updateProduct()">Update</button>
</div>

```

Componente product-detail

É necessário atualizar os ficheiros “product-detail.component.ts” e “product-detail.component.html” de acordo com os exemplos fornecidos.

Table 9 - product-detail.component.ts

```

import { Component, OnInit } from '@angular/core';
import { RestService } from '../rest.service';
import { ActivatedRoute, Router } from '@angular/router';


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

  product:any;

  constructor(public rest:RestService, private route: ActivatedRoute, private router: Router) { }

  ngOnInit() {
    this.rest.getProduct(this.route.snapshot.params['id']).subscribe((data: {}) => {
      console.log(data);
    });
  }
}

```

 <div data-bbox="437 147 528 224"> ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO </div>	Programação em Ambiente Web Docentes – FAS, EFE, JCarneiro, OAO Ficha Prática 9
---	---

```

    this.product = data;
  });
}
}

```

Table 10 - product-detail.component.ts

```

<div *ngIf="product" class="products">
  <h2>{{product.name | uppercase}} Details</h2>
  <div><span>Description: </span>{{product.description}}</div>
  <div><span>Price: </span>{{product.quantity}}</div>
  <div>
    <button routerLink="/product-edit/{{product.__id}}">
      Edit
    </button>
  </div>
</div>

```

Configurar Rotas

Falta apenas os módulos a importar no ficheiro “app.module.ts” e declarar as rotas para que a nossa aplicação funcione corretamente. Atualize o ficheiro de acordo com o exemplo fornecido.

Table 11 - app.module.ts

```

import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http';
import { ProductComponent } from './product/product.component';
import { ProductAddComponent } from './product-add/product-add.component';
import { RouterModule, Routes } from '@angular/router';
import { FormsModule } from '@angular/forms';
import { ProductDetailComponent } from './product-detail/product-detail.component';
import { ProductEditComponent } from './product-edit/product-edit.component';

const appRoutes: Routes = [
  {
    path: 'products',
    component: ProductComponent,
    data: { title: 'Product List' }
  },
  {
    path: 'product-details/:id',
    component: ProductDetailComponent,
    data: { title: 'Product Details' }
  },
  {
    path: 'product-add',
    component: ProductAddComponent,
    data: { title: 'Product Add' }
  },
  {
    path: 'product-edit/:id',
    component: ProductEditComponent,
    data: { title: 'Product Edit' }
  },
  { path: '',
    redirectTo: '/products',

```

```
    pathMatch: 'full'
  }
];

@NgModule({
  declarations: [
    AppComponent,
    ProductComponent,
    ProductAddComponent,
    ProductDetailComponent,
    ProductEditComponent
  ],
  imports: [
    RouterModule.forRoot(appRoutes),
    FormsModule,
    BrowserModule,
    HttpClientModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Podemos testar agora a aplicação executando o comando:

- `npm start`

Verifique o funcionamento da aplicação e que os serviços rest estão a ser usados para guardar produtos na base de dados.

Deve por esta altura verificar que as páginas não contêm formatação css. Pode adicionar ao seu gosto formatação em cada um dos css dos componentes.

Nas próximas aulas iremos melhorar a nossa abordagem na criação de aplicações *Angular*, reaproveitando componentes e usando outras características da *framework Angular*.

2 (Opcional) - Tutorial Angular

Neste exercício vamos fazer uso do tutorial padrão fornecido pela *framework Angular*. Desta forma aceda à página de internet: <https://angular.io/tutorial> e execute todos os passos do tutorial proposto.

No fim deste tutorial deve aperceber-se do uso de componentes, *routing*, serviços e layouts master/detail em *Angular*. Estes conceitos irão ser úteis em aplicações *Angular* mais avançadas.