**Create an API that allows used to Get and add products(id, name, price, quantity)**

**Repository pattern, Service are manidotory**

**Unit test your service**

**Consume the API in an angular application.**

**Show the product only if the quantity is more than 0**

**ProductappSolution (webApi)**

Model

namespace ProductApp.Models

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public double price { get; set; }

public double Qty { get; set; }

}

}

ProductContext

using Microsoft.EntityFrameworkCore;

using ProductApp.Models;

namespace ProductApp.PContext

{

public class ProductContext : DbContext

{

public ProductContext(DbContextOptions opts) : base(opts){ }

public DbSet<Product> products { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Product>().HasData(

new Product

{

Id = 1,

Name = "ABC",

price = 200,

Qty = 100

},

new Product

{

Id = 2,

Name = "DEF",

price = 1000,

Qty = 100

});

}

}

}

ProductRepostory

using ProductApp.Interfaces;

using ProductApp.Models;

using ProductApp.PContext;

namespace ProductApp.Repositories

{

public class ProductRepository : IRepository<int, Product>

{

private readonly ProductContext \_context;

public ProductRepository(ProductContext context)

{

\_context = context;

}

public Product Add(Product item)

{

\_context.products.Add(item);

\_context.SaveChanges();

return item;

}

public Product Delete(int key)

{

var product = Get(key);

if(product != null)

{

\_context.products.Remove(product);

\_context.SaveChanges();

return product;

}

return null;

}

public Product Get(int key)

{

var product = \_context.products.FirstOrDefault(p=>p.Id == key);

return product;

}

public List<Product> GetAll()

{

return \_context.products.ToList();

}

public Product Update(Product item)

{

\_context.Entry<Product>(item).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

\_context.SaveChanges();

return item;

}

}

}

Interfaces

IRepository

namespace ProductApp.Interfaces

{

public interface IRepository <K,T>

{

public List<T> GetAll();

public T Get(K key);

public T Add(T item);

public T Delete(K key);

public T Update(T item);

}

}

IProductServices

using ProductApp.Models;

namespace ProductApp.Interfaces

{

public interface IProductService

{

List<Product> GetAllProducts();

Product AddANewProduct(Product product);

}

}

ProductServices

using ProductApp.Interfaces;

using ProductApp.Models;

namespace ProductApp.Services

{

public class ProductService : IProductService

{

private readonly IRepository<int, Product> \_repo;

public ProductService(IRepository<int , Product> repository)

{

\_repo=repository;

}

public Product AddANewProduct(Product product)

{

return \_repo.Add(product);

}

public List<Product> GetAllProducts()

{

return \_repo.GetAll();

}

}

}

ProductController

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using ProductApp.Interfaces;

using ProductApp.Models;

namespace ProductApp.Controllers

{

[EnableCors("MyCors")]

[Route("api/[controller]")]

[ApiController]

public class ProductController : ControllerBase

{

private readonly IProductService \_proservice;

public ProductController(IProductService productservice)

{

\_proservice= productservice;

}

[HttpGet]

public ActionResult Get() {

var result = \_proservice.GetAllProducts();

if (result == null)

{

return NotFound("Product Are Not Available");

}

return Ok(result);

}

[HttpPost]

public ActionResult Post(Product product)

{

if(ModelState.IsValid)

{

try

{

var result = \_proservice.AddANewProduct(product);

return Created("", result);

}

catch (Exception e )

{

return BadRequest(e.Message);

}

}

return BadRequest(ModelState.Keys);

}

}

}

productTest

using Microsoft.EntityFrameworkCore;

using ProductApp.Interfaces;

using ProductApp.Models;

using ProductApp.PContext;

using ProductApp.Repositories;

using ProductApp.Services;

namespace ProductTest

{

public class ProductServiceTest

{

ProductContext context;

[SetUp]

public void Setup()

{

var dbContextOption = new DbContextOptionsBuilder<ProductContext>().UseInMemoryDatabase(databaseName: "dbDummyProduct").Options;

context = new ProductContext(dbContextOption);

}

#region AddTest

[Test]

public void AddTest()

{

// Arrange

IRepository<int, Product> productrepository = new ProductRepository(context);

IProductService productservice = new ProductService(productrepository);

//Action

var pro= new Product { Id = 1, Name = "ABC", price = 200 , Qty = 120 };

var result = productservice.AddANewProduct(pro);

var data = new Product { Id = 1, Name = "ABC", price = 200, Qty = 120 };

// Assert

Assert.AreEqual(data.Id, result.Id);

}

#endregion

#region GetAllProducts

[Test]

public void GetAllProduct()

{

IRepository<int, Product> productrepository = new ProductRepository(context);

IProductService productservice = new ProductService(productrepository);

Product product= new Product { Id = 1, Name = "ABC", price = 200, Qty = 120 };

productrepository.Add(product);

var result = productservice.GetAllProducts();

Assert.NotNull(result);

Assert.AreEqual(1,result.Count);

}

#endregion

}

}

**Angular**

Product.component.html

<div class="alert alert-dark">

    <!-- Employee Id: <input class="form-control" type="number" [value]="employee.id" #eid

    (change)="changeId(eid.value)">

    <br/> -->

    Product Name : <input class="form-control" type="text" [(ngModel)]="product.name">

    <br/>

    Product Price : <input class="form-control" type="number" [(ngModel)]="product.price">

    <br/>

    Product Qty : <input class="form-control" type="number" [(ngModel)]="product.qty">

    <br/>

    <button (click)="addProduct()" class="btn btn-dark"> Add Product

        <span [class]="className">

        </span>

    </button>

</div>

Prosuct.component.ts

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

import { Product } from './product';

import { ProductWebApiService } from '../services/ProductWebApi.Service';

@Component({

  selector: 'app-product',

  templateUrl: './product.component.html',

  styleUrls: ['./product.component.css']

})

export class ProductComponent {

product:Product =new Product()

className:string="";

constructor(private productService:ProductWebApiService)

{

}

addProduct(){

  this.className="spinner-border";

  this.productService.addProduct(this.product).subscribe(data=>

  {

    this.product = data as Product;

    if(this.product.id>0)

    {

      alert("The Product has been Added");

    }

    this.className="";

  },

  (err)=>

  {

    console.log(err);

  })

  this.product = new Product();

}

}

Product.component.html

<div \*ngIf = "products.length > 0 else noproduct" >

    <div class="alert alert-primary" \*ngFor = "let product of products">

        <div \*ngIf="product.qty > 0">

                    Product Id : {{product.id}}

                    <br/>

                    Product Name : {{product.name}}

                    <br/>

                    Product Price : {{product.price}}

                    <br/>

                    Product Qty : {{product.qty}}

        </div>

    </div>

</div>

<ng-template #noproduct >

    <div class="alert alert-warning">

       no product avaialable

    </div>

</ng-template>

Prosuct.component.ts

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

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

import { ProductWebApiService } from '../services/ProductWebApi.Service';

@Component({

  selector: 'app-products',

  templateUrl: './products.component.html',

  styleUrls: ['./products.component.css']

})

export class ProductsComponent {

products:Product[]=[];

constructor(private productService:ProductWebApiService)

{

  this.productService.getProduct().subscribe(data=>

    {

      console.log(data)

      this.products = data as Product[];

    })

}

}

Product.ts

export class Product{

    constructor(public id:number = 0,

        public name:string = "",

        public price:number=0,

        public qty:number=0

        )

        {

        }

}

Index.html

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>PRODUCTAPP</title>

  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="icon" type="image/x-icon" href="favicon.ico">

  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" integrity="sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65" crossorigin="anonymous">

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

</head>

<body>

  <app-root></app-root>

</body>

</html>

App-module.ts

import { NgModule } from '@angular/core';

import {FormsModule } from '@angular/forms';

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

import { AppRoutingModule } from './app-routing.module';

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

import {HttpClientModule} from '@angular/common/http';

import { ProductComponent } from './product/product.component';

import { ProductsComponent } from './products/products.component';

import { productService } from './services/product.service';

import { ProductWebApiService } from './services/ProductWebApi.Service';

@NgModule({

  declarations: [

    AppComponent,

    ProductComponent,

    ProductsComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule,

    HttpClientModule

  ],

  providers: [productService , ProductWebApiService],

  bootstrap: [AppComponent]

})

export class AppModule { }

app.components.html

<h1 class="alert alert-success">My Product Application</h1>

<div class="container">

  <div class="row">

    <div class="col-md-auto">

     <app-product></app-product>

    </div>

    <div class="col-md-auto">

      <app-products></app-products>

          </div>

  </div>

</div>