using System;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Caching.Memory;

namespace CatalogService.Controllers

{

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

[ApiController]

public class ProductsController : ControllerBase

{

private readonly IMemoryCache \_memoryCache;

public ProductsController(IMemoryCache memoryCache)

{

\_memoryCache = memoryCache;

}

[HttpGet]

public IActionResult Get()

{

var products = \_memoryCache.Get("products") as List<Product>;

if (products == null)

return NotFound();

return Ok(products);

}

[HttpGet("{id}")]

public IActionResult Get(int id)

{

var products = \_memoryCache.Get("products") as List<Product>;

if (products == null)

return NotFound();

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null)

return NotFound();

return Ok(product);

}

[HttpPost]

public IActionResult Post([FromBody] Product product)

{

var products = \_memoryCache.Get("products") as List<Product>;

if (products == null)

products = new List<Product>();

product.Id = products.Count + 1;

products.Add(product);

\_memoryCache.Set("products", products);

return CreatedAtAction(nameof(Get), new { id = product.Id }, product);

}

}

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public string Description { get; set; }

public string ImageUrl { get; set; }

}

}

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

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

import { Product } from './product';

@Injectable({

providedIn: 'root'

})

export class ProductService {

constructor(private http: HttpClient) { }

getProducts() {

return this.http.get<Product[]>('/api/products');

}

getProduct(id: number) {

return this.http.get<Product>(`/api/products/${id}`);

}

addProduct(product: Product) {

return this.http.post<Product>('/api/products', product);

}

}

using System.Collections.Generic;

using System.Linq;

using CatalogService.Models;

namespace CatalogService.Repositories

{

public class CatalogRepository : ICatalogRepository

{

private readonly List<Product> \_products;

public CatalogRepository()

{

\_products = new List<Product>

{

new Product { Id = 1, Name = "Product 1", Price = 100, Description = "Description 1", ImageUrl = "ImageUrl1" },

new Product { Id = 2, Name = "Product 2", Price = 200, Description = "Description 2", ImageUrl = "ImageUrl2" },

new Product { Id = 3, Name = "Product 3", Price = 300, Description = "Description 3", ImageUrl = "ImageUrl3" }

};

}

public List<Product> GetAllProducts()

{

return \_products;

}

public Product GetProductById(int id)

{

return \_products.FirstOrDefault(p => p.Id == id);

}

public void AddProduct(Product product)

{

product.Id = \_products.Max(p => p.Id) + 1;

\_products.Add(product);

}

}

}

**Order service**

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using OrderService.Models;

using OrderService.Repositories;

namespace OrderService.Controllers

{

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

[ApiController]

public class OrderController : ControllerBase

{

private readonly IOrderRepository \_orderRepository;

public OrderController(IOrderRepository orderRepository)

{

\_orderRepository = orderRepository;

}

[HttpGet]

public ActionResult<IEnumerable<Order>> Get()

{

return Ok(\_orderRepository.GetAllOrders());

}

[HttpGet("{id}")]

public ActionResult<Order> Get(int id)

{

var order = \_orderRepository.GetOrderById(id);

if (order == null)

{

return NotFound();

}

return Ok(order);

}

[HttpPost]

public async Task<ActionResult<Order>> Post([FromBody] Order order)

{

await \_orderRepository.AddOrder(order);

return CreatedAtAction("Get", new { id = order.Id }, order);

}

}

}

using System.Collections.Generic;

using OrderService.Models;

namespace OrderService.Repositories

{

public interface IOrderRepository

{

List<Order> GetAllOrders();

Order GetOrderById(int id);

void AddOrder(Order order);

}

}

using System.Collections.Generic;

using System.Linq;

using OrderService.Models;

namespace OrderService.Repositories

{

public class OrderRepository : IOrderRepository

{

private static List<Order> \_orders = new List<Order>();

public List<Order> GetAllOrders()

{

return \_orders;

}

public Order GetOrderById(int id)

{

return \_orders.FirstOrDefault(o => o.Id == id);

}

public void AddOrder(Order order)

{

order.Id = \_orders.Count + 1;

\_orders.Add(order);

}

}

}

Customer.cs

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using CustomerService.Models;

using CustomerService.Repositories;

namespace CustomerService.Controllers

{

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

[ApiController]

public class CustomerController : ControllerBase

{

private readonly ICustomerRepository \_customerRepository;

public CustomerController(ICustomerRepository customerRepository)

{

\_customerRepository = customerRepository;

}

[HttpGet]

public ActionResult<IEnumerable<Customer>> Get()

{

return Ok(\_customerRepository.GetAllCustomers());

}

[HttpGet("{id}")]

public ActionResult<Customer> Get(int id)

{

var customer = \_customerRepository.GetCustomerById(id);

if (customer == null)

{

return NotFound();

}

return Ok(customer);

}

[HttpPost]

public async Task<ActionResult<Customer>> Post([FromBody] Customer customer)

{

await \_customerRepository.AddCustomer(customer);

return CreatedAtAction("Get", new { id = customer.Id }, customer);

}

}

}

using System.Collections.Generic;

using CustomerService.Models;

namespace CustomerService.Repositories

{

public interface ICustomerRepository

{

List<Customer> GetAllCustomers();

Customer GetCustomerById(int id);

void AddCustomer(Customer customer);

}

}

using System.Collections.Generic;

using System.Linq;

using CustomerService.Models;

namespace CustomerService.Repositories

{

public class CustomerRepository : ICustomerRepository

{

private static List<Customer> \_customers = new List<Customer>();

public List<Customer> GetAllCustomers()

{

return \_customers;

}

public Customer GetCustomerById(int id)

{

return \_customers.FirstOrDefault(c => c.Id == id);

}

public void AddCustomer(Customer customer)

{

customer.Id = \_customers.Count + 1;

\_customers.Add(customer);

}

}

Inventory Service

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using InventoryService.Models;

using InventoryService.Repositories;

namespace InventoryService.Controllers

{

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

[ApiController]

public class InventoryController : ControllerBase

{

private readonly IInventoryRepository \_inventoryRepository;

public InventoryController(IInventoryRepository inventoryRepository)

{

\_inventoryRepository = inventoryRepository;

}

[HttpGet]

public ActionResult<IEnumerable<InventoryItem>> Get()

{

return Ok(\_inventoryRepository.GetAllInventoryItems());

}

[HttpGet("{id}")]

public ActionResult<InventoryItem> Get(int id)

{

var inventoryItem = \_inventoryRepository.GetInventoryItemById(id);

if (inventoryItem == null)

{

return NotFound();

}

return Ok(inventoryItem);

}

[HttpPut("{id}")]

public async Task<ActionResult<InventoryItem>> Put(int id, [FromBody] InventoryItem inventoryItem)

{

if (id != inventoryItem.Id)

{

return BadRequest();

}

await \_inventoryRepository.UpdateInventoryItem(inventoryItem);

return NoContent();

}

}

}

**using System.Collections.Generic;**

**using InventoryService.Models;**

**namespace InventoryService.Repositories**

**{**

**public interface IInventoryRepository**

**{**

**List<InventoryItem> GetAllInventoryItems();**

**InventoryItem GetInventoryItemById(int id);**

**void UpdateInventoryItem(InventoryItem inventoryItem);**

**}**

**}**

using System.Collections.Generic;

using System.Linq;

using InventoryService.Models;

namespace InventoryService.Repositories

{

public class InventoryRepository : IInventoryRepository

{

private static List<InventoryItem> \_inventoryItems = new List<InventoryItem>();

public List<InventoryItem> GetAllInventoryItems()

{

return \_inventoryItems;

}

public InventoryItem GetInventoryItemById(int id)

{

return \_inventoryItems.FirstOrDefault(i => i.Id == id);

}

public void UpdateInventoryItem(InventoryItem inventoryItem)

{

var existingInventoryItem = \_inventoryItems.FirstOrDefault(i => i.Id == inventoryItem.Id);

if (existingInventoryItem != null)

{

\_inventoryItems.Remove(existingInventoryItem);

}

\_inventoryItems.Add(inventoryItem);

}

}

}

Shooping cart

using System.Collections.Generic;

using System.Linq;

using ShoppingCartService.Models;

using ShoppingCartService.Repositories;

namespace ShoppingCartService.Services

{

public class ShoppingCartService : IShoppingCartService

{

private readonly IShoppingCartRepository \_shoppingCartRepository;

public ShoppingCartService(IShoppingCartRepository shoppingCartRepository)

{

\_shoppingCartRepository = shoppingCartRepository;

}

public ShoppingCart GetShoppingCartById(int id)

{

return \_shoppingCartRepository.GetShoppingCartById(id);

}

public List<ShoppingCartItem> GetShoppingCartItems(int shoppingCartId)

{

return \_shoppingCartRepository.GetShoppingCartItems(shoppingCartId);

}

public void AddItemToShoppingCart(int shoppingCartId, ShoppingCartItem shoppingCartItem)

{

\_shoppingCartRepository.AddItemToShoppingCart(shoppingCartId, shoppingCartItem);

}

public void RemoveItemFromShoppingCart(int shoppingCartId, int shoppingCartItemId)

{

\_shoppingCartRepository.RemoveItemFromShoppingCart(shoppingCartId, shoppingCartItemId);

}

public void ClearShoppingCart(int shoppingCartId)

{

\_shoppingCartRepository.ClearShoppingCart(shoppingCartId);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ShoppingCartService.Repository

{

public class ShoppingCartRepository

{

private static readonly List<Models.ShoppingCart> \_carts = new List<Models.ShoppingCart>();

public Models.ShoppingCart GetById(string id)

{

return \_carts.FirstOrDefault(x => x.Id == id);

}

public Models.ShoppingCart Add(Models.ShoppingCart cart)

{

\_carts.Add(cart);

return cart;

}

public void Update(Models.ShoppingCart cart)

{

var index = \_carts.FindIndex(x => x.Id == cart.Id);

if (index != -1)

{

\_carts[index] = cart;

}

}

public void Remove(string id)

{

\_carts.RemoveAll(x => x.Id == id);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ShoppingCartService.Repository

{

public interface IShoppingCartRepository

{

Models.ShoppingCart GetById(string id);

Models.ShoppingCart Add(Models.ShoppingCart cart);

void Update(Models.ShoppingCart cart);

void Remove(string id);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ShippingService

{

public class ShippingService

{

private readonly IShippingRepository \_shippingRepository;

public ShippingService(IShippingRepository shippingRepository)

{

\_shippingRepository = shippingRepository;

}

public Models.ShippingInfo GetShippingInfo(string orderId)

{

return \_shippingRepository.GetByOrderId(orderId);

}

public Models.ShippingInfo UpdateShippingInfo(Models.ShippingInfo shippingInfo)

{

return \_shippingRepository.Update(shippingInfo);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ShippingService

{

public interface IShippingRepository

{

Models.ShippingInfo GetByOrderId(string orderId);

Models.ShippingInfo Update(Models.ShippingInfo shippingInfo);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ShippingService.Repository

{

public class ShippingRepository : IShippingRepository

{

private static readonly List<Models.ShippingInfo> \_shippings = new List<Models.ShippingInfo>();

public Models.ShippingInfo GetByOrderId(string orderId)

{

return \_shippings.FirstOrDefault(x => x.OrderId == orderId);

}

public Models.ShippingInfo Update(Models.ShippingInfo shippingInfo)

{

var existingShipping = \_shippings.FirstOrDefault(x => x.OrderId == shippingInfo.OrderId);

if (existingShipping != null)

{

existingShipping.Address = shippingInfo.Address;

existingShipping.City = shippingInfo.City;

existingShipping.State = shippingInfo.State;

existingShipping.ZipCode = shippingInfo.ZipCode;

existingShipping.Country = shippingInfo.Country;

return existingShipping;

}

else

{

\_shippings.Add(shippingInfo);

return shippingInfo;

}

}

}

}

using Microsoft.AspNetCore.Mvc;

using ShippingService.Models;

using ShippingService.Services;

namespace ShippingService.Controllers

{

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

[ApiController]

public class ShippingController : ControllerBase

{

private readonly ShippingService \_shippingService;

public ShippingController(ShippingService shippingService)

{

\_shippingService = shippingService;

}

[HttpGet("{orderId}")]

public ActionResult<ShippingInfo> Get(string orderId)

{

var shippingInfo = \_shippingService.GetShippingInfo(orderId);

if (shippingInfo == null)

{

return NotFound();

}

return shippingInfo;

}

[HttpPut("{orderId}")]

public ActionResult<ShippingInfo> Put(string orderId, ShippingInfo shippingInfo)

{

if (orderId != shippingInfo.OrderId)

{

return BadRequest();

}

var updatedShippingInfo = \_shippingService.UpdateShippingInfo(shippingInfo);

if (updatedShippingInfo == null)

{

return NotFound();

}

return updatedShippingInfo;

}

}

}

using System.Collections.Generic;

using System.Linq;

using SearchService.Models;

using SearchService.Repository;

namespace SearchService.Services

{

public class SearchService : ISearchService

{

private readonly ISearchRepository \_searchRepository;

public SearchService(ISearchRepository searchRepository)

{

\_searchRepository = searchRepository;

}

public List<Product> Search(string query)

{

return \_searchRepository.Search(query);

}

}

}

using System.Collections.Generic;

using System.Linq;

using SearchService.Models;

namespace SearchService.Repository

{

public class SearchRepository : ISearchRepository

{

private static readonly List<Product> \_products = new List<Product>();

public List<Product> Search(string query)

{

return \_products.Where(p => p.Name.Contains(query) || p.Description.Contains(query)).ToList();

}

}

}

using Microsoft.AspNetCore.Mvc;

using SearchService.Models;

using SearchService.Services;

namespace SearchService.Controllers

{

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

[ApiController]

public class SearchController : ControllerBase

{

private readonly SearchService \_searchService;

public SearchController(SearchService searchService)

{

\_searchService = searchService;

}

[HttpGet("{query}")]

public ActionResult<List<Product>> Get(string query)

{

var products = \_searchService.Search(query);

return products;

}

}

}

using System.Collections.Generic;

using System.Linq;

using SearchService.Models;

using SearchService.Repository;

namespace SearchService.Services

{

public class SearchService : ISearchService

{

private readonly ISearchRepository \_searchRepository;

public SearchService(ISearchRepository searchRepository)

{

\_searchRepository = searchRepository;

}

public List<Product> Search(string query)

{

return \_searchRepository.Search(query);

}

}

}

using System.Collections.Generic;

using SearchService.Models;

namespace SearchService.Services

{

public interface ISearchService

{

List<Product> Search(string query);

}

}