microservices-task-2

(I am sorry that I do not have experience in the technologies of .NET5 Core/Python/Java/Nodejs, I just modified the example from .NET5 tutorial according to the requirement of movie booking REST API)

<attributes>

namespace TicketBookingApi.Models

{

public class Ticket

{

public long Id { get; set; }

public datetime Date { get; set; }

public datetime Time { get; set; }

public string Venue { get; set; }

public integer NumberOfTickets {get; set}

public money Amount { get; set; }

}

}

Add the TicketBookingContext database context

using Microsoft.EntityFrameworkCore;

namespace TicketBookingApi.Models

{

public class TicketBookingContext : DbContext

{

public TicketBookingContext(DbContextOptions<TicketBookingContext> options)

: base(options)

{

}

public DbSet<TicketBookingItem> Tickets { get; set; }

}

}

Register the database context in Startup class

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

services.AddDbContext<TicketBookingContext>(opt =>opt.UseInMemoryDatabase("TicketList"));

}

Scaffold the controller

Add a new ticket

// POST: api/Ticket

[HttpPost]

public async Task<ActionResult<Ticket>> PostTicket(Ticket Item)

{

\_context.Ticket.Add(Item);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetTicket", new { id = Item.Id }, Item);

}

View a booking

// GET: api/Ticket/5 [HttpGet("{id}")]

public async Task<ActionResult<Ticket>> GetItem(long id)

{

var TicketItem = await \_context.TicketItems.FindAsync(id);

if (TicketItem == null)

{

return NotFound();

}

return TicketItem;

}

Update a booking

// PUT: api/TodoItems/5

[HttpPut("{id}")]

public async Task<IActionResult> PutTicketItem(long id, TicketItem ticketItem)

{

if (id != ticketItem.Id)

{

return BadRequest();

}

\_context.Entry(ticketItem).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!TicketItemExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

Delete a booking

// DELETE: api/TodoItems/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteTicketItem(long id)

{

var ticketItem = await \_context.TicketItems.FindAsync(id);

if (ticketItem == null)

{

return NotFound();

}

\_context.TodoItems.Remove(ticketItem);

await \_context.SaveChangesAsync();

return NoContent();

}