4. Controlere

a. Creeaza controller și operații CRUD pentru fiecare model;

b. Folosește modele DTO(Data Transfer Object) pentru a primi input și a

trimite output la client;

// UserController.cs

using System;

using System.Collections.Generic;

using DataLayer.Models;

using Microsoft.AspNetCore.Mvc;

namespace YourNamespace.Controllers

{

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

[ApiController]

public class UserController : ControllerBase

{

// Inject the DbContext or any service needed

private readonly MyDbContext \_context;

public UserController(MyDbContext context)

{

\_context = context;

}

// GET: api/User

[HttpGet]

public IEnumerable<UserDTO> GetUsers()

{

// Fetch users from database

// Convert to DTOs

// Return DTOs

}

// GET: api/User/5

[HttpGet("{id}")]

public ActionResult<UserDTO> GetUser(Guid id)

{

// Fetch user by id from database

// If user is not found, return NotFound

// Convert user to DTO

// Return DTO

}

// POST: api/User

[HttpPost]

public ActionResult<UserDTO> PostUser(UserDTO userDTO)

{

// Convert DTO to model

// Add model to database

// Save changes

// Convert added model to DTO

// Return DTO with status code 201 (Created)

}

// PUT: api/User/5

[HttpPut("{id}")]

public IActionResult PutUser(Guid id, UserDTO userDTO)

{

// If id doesn't match DTO id, return BadRequest

// Fetch user by id from database

// If user is not found, return NotFound

// Update user properties with DTO properties

// Save changes

// Return NoContent

}

// DELETE: api/User/5

[HttpDelete("{id}")]

public ActionResult<UserDTO> DeleteUser(Guid id)

{

// Fetch user by id from database

// If user is not found, return NotFound

// Remove user from database

// Save changes

// Convert deleted user to DTO

// Return DTO

}

}

}

// UserTypeController.cs

using System;

using System.Collections.Generic;

using DataLayer.Models;

using Microsoft.AspNetCore.Mvc;

namespace YourNamespace.Controllers

{

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

[ApiController]

public class UserTypeController : ControllerBase

{

private readonly MyDbContext \_context;

public UserTypeController(MyDbContext context)

{

\_context = context;

}

[HttpGet]

public IEnumerable<UserTypeDTO> GetUserTypes()

{

// Fetch user types from database

// Convert to DTOs

// Return DTOs

}

[HttpGet("{id}")]

public ActionResult<UserTypeDTO> GetUserType(Guid id)

{

// Fetch user type by id from database

// If user type is not found, return NotFound

// Convert user type to DTO

// Return DTO

}

[HttpPost]

public ActionResult<UserTypeDTO> PostUserType(UserTypeDTO userTypeDTO)

{

// Convert DTO to model

// Add model to database

// Save changes

// Convert added model to DTO

// Return DTO with status code 201 (Created)

}

[HttpPut("{id}")]

public IActionResult PutUserType(Guid id, UserTypeDTO userTypeDTO)

{

// If id doesn't match DTO id, return BadRequest

// Fetch user type by id from database

// If user type is not found, return NotFound

// Update user type properties with DTO properties

// Save changes

// Return NoContent

}

[HttpDelete("{id}")]

public ActionResult<UserTypeDTO> DeleteUserType(Guid id)

{

// Fetch user type by id from database

// If user type is not found, return NotFound

// Remove user type from database

// Save changes

// Convert deleted user type to DTO

// Return DTO

}

}

}

**5. Validare**

**a. Pentru fiecare operațiune din CRUD care necesita o validare de input,**

**adaugati si returnati mesaj specific de eroare cand este cazul;**

**b. Dacă puteți extrage o metoda de validare comuna pentru mai multe**

**modele, folositi-o;**

**using Microsoft.AspNetCore.Mvc;**

**using DataLayer.Models;**

**using DataLayer.DTOs;**

**using System;**

**using System.Linq;**

**namespace YourNamespace.Controllers**

**{**

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

**[ApiController]**

**public class UserController : ControllerBase**

**{**

**private readonly MyDbContext \_context;**

**public UserController(MyDbContext context)**

**{**

**\_context = context;**

**}**

**// GET: api/User**

**[HttpGet]**

**public IActionResult GetUsers()**

**{**

**var users = \_context.Users.ToList();**

**return Ok(users);**

**}**

**// GET: api/User/5**

**[HttpGet("{id}")]**

**public IActionResult GetUser(Guid id)**

**{**

**var user = \_context.Users.Find(id);**

**if (user == null)**

**{**

**return NotFound("User not found");**

**}**

**return Ok(user);**

**}**

**// POST: api/User**

**[HttpPost]**

**public IActionResult AddUser(UserDto userDto)**

**{**

**// Validate userDto here**

**if (!ModelState.IsValid)**

**{**

**return BadRequest(ModelState);**

**}**

**var user = new User(userDto.Name, userDto.Email, userDto.Password, userDto.TypeId);**

**\_context.Users.Add(user);**

**\_context.SaveChanges();**

**return CreatedAtAction(nameof(GetUser), new { id = user.Id }, user);**

**}**

**// PUT: api/User/5**

**[HttpPut("{id}")]**

**public IActionResult UpdateUser(Guid id, UserDto userDto)**

**{**

**// Validate userDto here**

**if (!ModelState.IsValid)**

**{**

**return BadRequest(ModelState);**

**}**

**var user = \_context.Users.Find(id);**

**if (user == null)**

**{**

**return NotFound("User not found");**

**}**

**// Update user properties**

**user.Name = userDto.Name;**

**user.Email = userDto.Email;**

**user.Password = userDto.Password;**

**user.TypeId = userDto.TypeId;**

**\_context.SaveChanges();**

**return NoContent();**

**}**

**// DELETE: api/User/5**

**[HttpDelete("{id}")]**

**public IActionResult DeleteUser(Guid id)**

**{**

**var user = \_context.Users.Find(id);**

**if (user == null)**

**{**

**return NotFound("User not found");**

**}**

**\_context.Users.Remove(user);**

**\_context.SaveChanges();**

**return NoContent();**

**}**

**}**

**}**

**6. Inițializare**

**a. Adaugati automat(din DbContext) în tabelele fiecărui model, cateva linii**

**cu valori predefinite, ce se vor actualiza la pornirea aplicației;  
using Microsoft.EntityFrameworkCore;**

**using DataLayer.Models;**

**using System;**

**using System.Collections.Generic;**

**namespace DataLayer**

**{**

**public class MyDbContext : DbContext**

**{**

**private readonly string \_customConnectionString = @"Server=myServerAddress;Database=myDatabase;User Id=myUsername;Password=myPassword;";**

**public DbSet<User> Users { get; set; }**

**public DbSet<UserType> UserTypes { get; set; }**

**protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)**

**{**

**optionsBuilder.UseSqlServer(\_customConnectionString);**

**}**

**protected override void OnModelCreating(ModelBuilder modelBuilder)**

**{**

**base.OnModelCreating(modelBuilder);**

**// Configurare relații între entități**

**// Adăugare date predefinite în tabele la inițializarea bazei de date**

**SeedData(modelBuilder);**

**}**

**private void SeedData(ModelBuilder modelBuilder)**

**{**

**// Adăugare tipuri de utilizatori predefinite**

**modelBuilder.Entity<UserType>().HasData(**

**new UserType { Id = Guid.NewGuid(), Name = "Admin" },**

**new UserType { Id = Guid.NewGuid(), Name = "Regular" }**

**);**

**// Adăugare utilizatori predefiniți**

**modelBuilder.Entity<User>().HasData(**

**new User { Id = Guid.NewGuid(), Name = "John", Email = "john@example.com", Password = "password123", TypeId = Guid.NewGuid() },**

**new User { Id = Guid.NewGuid(), Name = "Alice", Email = "alice@example.com", Password = "password456", TypeId = Guid.NewGuid() }**

**);**

**}**

**}**

**}**