Database model

|  |  |  |
| --- | --- | --- |
| Column | DataType | Null |
| UserID | int |  |
| FirstName | nvarchar |  |
| LastName | nvarchar |  |
| Age | smallint |  |
| Email | nvarchar |  |
| Parkings | ICollection |  |
| Feedbacks | ICollection |  |

User

One to many between Parking and User

Parking

|  |  |  |
| --- | --- | --- |
| Column | DataType | Null |
| ParkingID | int |  |
| Country | nvarchar |  |
| City | nvarchar |  |
| StreetAdress | nvarchar |  |
| FreeParkingStart | string |  |
| FreeParkingEnd | string |  |
| User | User |  |
| UserID | int |  |
| Feedbacks | ICollection |  |

One to many between user and Feedback

One to many between Parking and Feedback

Feedback

|  |  |  |
| --- | --- | --- |
| Column | DataType | Null |
| FeedbackID | int |  |
| Comment | nvarchar |  |
| Rate | int |  |
| ParkingID | int |  |
| Parking | Parking |  |
| UserID | int |  |
| User | User |  |

Packages

1. Automapper
2. Microsoft.AspNetCore.Mvc.NewtonsoftJson
3. Microsoft.EntityFrameworkCore
4. Microsoft.EntityFrameworkCore.SqlServer
5. Microsoft.EntityFrameworkCore.SqlServer Design
6. Microsoft.EntityFrameworkCore.Tools
7. Moq.EntityFrameworkCore
8. Swashbuckle.AspNetCore

IUser interface har alla metoder som controller anropar

IUser interface ärver från IRepository för att komma åt ADD, Delete,Update,Save metoder

public interface IRepository

{

void Add<T>(T entity) where T : class;

void Delete<T>(T entity) where T : class;

void Update<T>(T entity) where T : class;

Task<bool> Save();

}

public interface IUserRepo:IRepository

{

Task<User[]> GetAllUser(bool includeParking, bool includeFeedback);

Task<User> GetUserByID(int id, bool includeParking=false, bool includeFeeback=false);

Task<User> GetUserByName(string name, bool includeparking, bool includeFeedback);

}

Repository ärver från IRepository och måste implementera alla methoder

DataContext är injekterad i konstruktorn

public class Repository : IRepository

{

protected readonly DataContext \_dbContext;

protected readonly ILogger<Repository> \_logger;

public Repository(DataContext context, ILogger<Repository> logger)

{

this.\_dbContext = context;

this.\_logger = logger;

}

public virtual void Add<T>(T entity) where T : class

{

\_logger.LogInformation($"Adding object of type {entity.GetType()}");

\_dbContext.Add(entity)

}

UserRepository ärver från klassen Repository och IUser Interface och impementera interfacen för IUserRepo

Vi kommer åt DataContext genom Repository klassen som ärvs

public class UserRepository:Repository, IUserRepo

{

public UserRepository(DataContext context, ILogger<UserRepository> logger) : base(context, logger){}

public virtual async Task<User[]> GetAllUser(bool includeParking, bool includeFeedback)

{

\_logger.LogInformation($"Getting all users");

IQueryable<User> query = \_dbContext.Users;

if (includeParking)

{

query = query.Include(x => x.Parkings);

}

if (includeFeedback)

{

query = query.Include(x => x.Feedbacks);

}

return await query.ToArrayAsync();

}

[Route("api/v1.0/[controller]")]

[ApiController]

public class UserController:HateoasControllerBase

{

private readonly IUserRepo repo;

private readonly IMapper mapper;

public UserController(IUserRepo repo, IMapper mapper,

IActionDescriptorCollectionProvider provider) : base(provider)

{

this.repo = repo;

this.mapper = mapper;

}

[HttpGet(Name ="GetAllUsers")]

public async Task<IActionResult>GetUsers([FromQuery]bool includeLinks=true,

[FromQuery] bool includeParking = false, [FromQuery]bool includeFeedback=false)

{

try

{

var result = await repo.GetAllUser(includeParking, includeFeedback);

if (result==null)

{

return NotFound();

}

IEnumerable<UserDTO> mapped = mapper.Map<UserDTO[]>(result);

if (includeLinks)

{

mapped = mapped.Select(x => HateoasMainLinks(x));

}

return Ok(mapped);

}

catch (Exception e)

{

return StatusCode(StatusCodes.Status500InternalServerError,

$"Database failure {e.Message}");

}

}

1. Controller anropar IUserRepo Interface genom att injektera det i sin kontruktor.
2. UserRepository ärver IUserRepo interface och implementerar alla methoder
3. UserRepository ärver också Repository för att komma åt DataContext
4. Repository ärver IRepository interface och implementera alla metoder
5. IUserRepo interface ärver IRepository interface för att komma åt metoder ADD, Delete osv

Repository

IRepository

IUserRepo

UserRepository

UserController