```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using DNPExercises10DataAccess;
using DNPExercises10DataAccess.Data.Entities;

namespace DNPExercises10DataAccess.Controllers
{
    [Route("api/cats")]
    [ApiController]
    public class CatsController : ControllerBase
    {
        private readonly ApplicationDbContext _context;

        public CatsController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: api/Cats
        [HttpGet]
        public IEnumerable<Cat> Getcats()
        {

            //Query syntax:
            IEnumerable<Cat> catQuery =
                from cats in _context.cats
                select cats;



            if (catQuery == null)
            {
                return null;
            }

            return catQuery;

        }

        // GET: api/Cats/5 with Async
        [HttpGet("{id}")]
        public async Task<IActionResult> GetCat([FromRoute] int id)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var cat = await _context.cats.FindAsync(id);

            if (cat == null)
            {
                return NotFound();
            }

            return Ok(cat);
        }



        // PUT: api/Cats/5
        [HttpPut("{id}")]
        public async Task<IActionResult> PutCat([FromRoute] int id,
          [FromBody] Cat cat)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            if (id != cat.Id)
            {
                return BadRequest();
            }

            _context.Entry(cat).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!CatExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return Ok(cat);
        }

        // POST: api/Cats
        [HttpPost]
        public async Task<IActionResult> PostCat([FromBody] Cat cat)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }


            _context.cats.Add(cat);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetCat", new { id = cat.Id }, cat);
        }

        // DELETE: api/Cats/5
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteCat([FromRoute] int id)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var cat = await _context.cats.FindAsync(id);
            if (cat == null)
            {
                return NotFound();
            }

            _context.cats.Remove(cat);
            await _context.SaveChangesAsync();

            return Ok(cat);
        }

        private bool CatExists(int id)
        {
            return _context.cats.Any(e => e.Id == id);
        }
    }
}
```