

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5 using Microsoft.AspNetCore.Http;
6 using Microsoft.AspNetCore.Mvc;
7 using Microsoft.EntityFrameworkCore;
8 using DNPEercises10DataAccess;
9 using DNPEercises10DataAccess.Data.Entities;
10
11 namespace DNPEercises10DataAccess.Controllers
12 {
13     [Route("api/cats")]
14     [ApiController]
15     public class CatsController : ControllerBase
16     {
17         private readonly ApplicationDbContext _context;
18
19         public CatsController(ApplicationDbContext context)
20         {
21             _context = context;
22         }
23
24         // GET: api/Cats
25         [HttpGet]
26         public IEnumerable<Cat> Getcats()
27         {
28
29             //Query syntax:
30             IEnumerable<Cat> catQuery =
31                 from cats in _context.cats
32                 select cats;
33
34
35
36             if (catQuery == null)
37             {
38                 return null;
39             }
40
41             return catQuery;
42         }
43     }
44
45     // GET: api/Cats/5 with Async
46     [HttpGet("{id}")]
47     public async Task<IActionResult> GetCat([FromRoute] int id)
48     {
49         if (!ModelState.IsValid)
50         {
51             return BadRequest(ModelState);
52         }
53     }
```

```
54         var cat = await _context.cats.FindAsync(id);
55
56         if (cat == null)
57         {
58             return NotFound();
59         }
60
61         return Ok(cat);
62     }
63
64
65
66     // PUT: api/Cats/5
67     [HttpPut("{id}")]
68     public async Task<IActionResult> PutCat([FromRoute] int id,           ↗
69         [FromBody] Cat cat)
70     {
71         if (!ModelState.IsValid)
72         {
73             return BadRequest(ModelState);
74         }
75
76         if (id != cat.Id)
77         {
78             return BadRequest();
79         }
80
81         _context.Entry(cat).State = EntityState.Modified;
82
83         try
84         {
85             await _context.SaveChangesAsync();
86         }
87         catch (DbUpdateConcurrencyException)
88         {
89             if (!CatExists(id))
90             {
91                 return NotFound();
92             }
93             else
94             {
95                 throw;
96             }
97         }
98
99         return Ok(cat);
100     }
101
102     // POST: api/Cats
103     [HttpPost]
104     public async Task<IActionResult> PostCat([FromBody] Cat cat)
105     {
106         if (!ModelState.IsValid)
```

```
106         {
107             return BadRequest(ModelState);
108         }
109
110
111         _context.cats.Add(cat);
112         await _context.SaveChangesAsync();
113
114         return CreatedAtAction("GetCat", new { id = cat.Id }, cat);
115     }
116
117     // DELETE: api/Cats/5
118     [HttpDelete("{id}")]
119     public async Task<IActionResult> DeleteCat([FromRoute] int id)
120     {
121         if (!ModelState.IsValid)
122         {
123             return BadRequest(ModelState);
124         }
125
126         var cat = await _context.cats.FindAsync(id);
127         if (cat == null)
128         {
129             return NotFound();
130         }
131
132         _context.cats.Remove(cat);
133         await _context.SaveChangesAsync();
134
135         return Ok(cat);
136     }
137
138     private bool CatExists(int id)
139     {
140         return _context.cats.Any(e => e.Id == id);
141     }
142 }
143 }
```