

Ödev 1: Film Arşivi API'si

Model: Movie.cs

```
namespace MovieArchiveAPI.Models
{
    public class Movie
    {
        public int Id { get; set; }
        public string? Title { get; set; }
        public string? Director { get; set; }
        public int ReleaseYear { get; set; }
    }
}
```

Controller: MoviesController.cs

```
using Microsoft.AspNetCore.Mvc;
using MovieArchiveAPI.Models;

namespace MovieArchiveAPI.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class MoviesController : ControllerBase
    {
        private static readonly List<Movie> _movies = new List<Movie>
        {
            new Movie { Id = 1, Title = "The Shawshank Redemption", Director = "Frank Darabont", ReleaseYear = 1994 },
            new Movie { Id = 2, Title = "The Godfather", Director = "Francis Ford Coppola", ReleaseYear = 1972 }
        };
        private static int _nextId = 3;

        [HttpGet]
        public ActionResult<IEnumerable<Movie>> GetMovies()
        {
            return Ok(_movies);
        }

        [HttpGet("{id}")]
        public ActionResult<Movie> GetMovie(int id)
        {
            var movie = _movies.FirstOrDefault(m => m.Id == id);
            if (movie == null)
            {
                return NotFound();
            }
            return Ok(movie);
        }

        [HttpPost]
```

```

        public ActionResult<Movie> PostMovie(Movie movie)
        {
            movie.Id = _nextId++;
            _movies.Add(movie);
            return CreatedAtAction(nameof(GetMovie), new { id = movie.Id },
movie);
        }

        [HttpPut("{id}")]
        public IActionResult PutMovie(int id, Movie movie)
        {
            if (id != movie.Id)
            {
                return BadRequest();
            }

            var existingMovie = _movies.FirstOrDefault(m => m.Id == id);
            if (existingMovie == null)
            {
                return NotFound();
            }

            existingMovie.Title = movie.Title;
            existingMovie.Director = movie.Director;
            existingMovie.ReleaseYear = movie.ReleaseYear;

            return NoContent();
        }

        [HttpDelete("{id}")]
        public IActionResult DeleteMovie(int id)
        {
            var movie = _movies.FirstOrDefault(m => m.Id == id);
            if (movie == null)
            {
                return NotFound();
            }

            _movies.Remove(movie);
            return NoContent();
        }
    }
}

```

Ödev 2: Blog Yönetim API'si

Model: Post.cs

```

namespace BlogManagementAPI.Models
{
    public class Post
    {
        public int Id { get; set; }
    }
}

```

```

        public string? Title { get; set; }
        public string? Content { get; set; }
        public string? Author { get; set; }
        public DateTime PublishedDate { get; set; }
    }
}

```

Controller: PostsController.cs

```

using Microsoft.AspNetCore.Mvc;
using BlogManagementAPI.Models;

namespace BlogManagementAPI.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class PostsController : ControllerBase
    {
        private static readonly List<Post> _posts = new List<Post>
        {
            new Post { Id = 1, Title = "First Post", Content = "This is the content of the first post.", Author = "Author 1", PublishedDate = DateTime.UtcNow },
            new Post { Id = 2, Title = "Second Post", Content = "This is the content of the second post.", Author = "Author 2", PublishedDate = DateTime.UtcNow }
        };
        private static int _nextId = 3;

        [HttpGet]
        public ActionResult<IEnumerable<Post>> GetPosts()
        {
            return Ok(_posts);
        }

        [HttpGet("{id}")]
        public ActionResult<Post> GetPost(int id)
        {
            var post = _posts.FirstOrDefault(p => p.Id == id);
            if (post == null)
            {
                return NotFound();
            }
            return Ok(post);
        }

        [HttpPost]
        public ActionResult<Post> PostPost(Post post)
        {
            post.Id = _nextId++;
            post.PublishedDate = DateTime.UtcNow;
            _posts.Add(post);
            return CreatedAtAction(nameof(GetPost), new { id = post.Id }, post);
        }
    }
}

```

```

        [HttpPut("{id}")]
        public IActionResult PutPost(int id, Post post)
        {
            if (id != post.Id)
            {
                return BadRequest();
            }

            var existingPost = _posts.FirstOrDefault(p => p.Id == id);
            if (existingPost == null)
            {
                return NotFound();
            }

            existingPost.Title = post.Title;
            existingPost.Content = post.Content;
            existingPost.Author = post.Author;

            return NoContent();
        }

        [HttpDelete("{id}")]
        public IActionResult DeletePost(int id)
        {
            var post = _posts.FirstOrDefault(p => p.Id == id);
            if (post == null)
            {
                return NotFound();
            }

            _posts.Remove(post);
            return NoContent();
        }
    }
}

```

Ödev 3: Görev Yönetimi (To-Do) API'si

Model: TaskItem.cs

```

namespace ToDoAPI.Models
{
    public class TaskItem
    {
        public int Id { get; set; }
        public string? Title { get; set; }
        public string? Description { get; set; }
        public bool IsCompleted { get; set; }
        public DateTime DueDate { get; set; }
    }
}

```

Controller: TasksController.cs

```

using Microsoft.AspNetCore.Mvc;
using ToDoAPI.Models;

namespace ToDoAPI.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class TasksController : ControllerBase
    {
        private static readonly List<TaskItem> _tasks = new List<TaskItem>
        {
            new TaskItem { Id = 1, Title = "Learn .NET", Description = "Study Web API development.", IsCompleted = false, DueDate = DateTime.UtcNow.AddDays(7) },
            new TaskItem { Id = 2, Title = "Build a Project", Description = "Complete the To-Do API assignment.", IsCompleted = false, DueDate = DateTime.UtcNow.AddDays(14) }
        };
        private static int _nextId = 3;

        [HttpGet]
        public ActionResult<IEnumerable<TaskItem>> GetTasks()
        {
            return Ok(_tasks);
        }

        [HttpGet("{id}")]
        public ActionResult<TaskItem> GetTask(int id)
        {
            var task = _tasks.FirstOrDefault(t => t.Id == id);
            if (task == null)
            {
                return NotFound();
            }
            return Ok(task);
        }

        [HttpPost]
        public ActionResult<TaskItem> PostTask(TaskItem task)
        {
            task.Id = _nextId++;
            _tasks.Add(task);
            return CreatedAtAction(nameof(GetTask), new { id = task.Id }, task);
        }

        [HttpPut("{id}")]
        public IActionResult PutTask(int id, TaskItem task)
        {
            if (id != task.Id)
            {
                return BadRequest();
            }

            var existingTask = _tasks.FirstOrDefault(t => t.Id == id);
            if (existingTask == null)

```

```

        {
            return NotFound();
        }

        existingTask.Title = task.Title;
        existingTask.Description = task.Description;
        existingTask.IsCompleted = task.IsCompleted;
        existingTask.DueDate = task.DueDate;

        return NoContent();
    }

    [HttpDelete("{id}")]
    public IActionResult DeleteTask(int id)
    {
        var task = _tasks.FirstOrDefault(t => t.Id == id);
        if (task == null)
        {
            return NotFound();
        }

        _tasks.Remove(task);
        return NoContent();
    }
}

```

Ödev 4: Ürün Kataloğu API'si

Model: Product.cs

```

using System.ComponentModel.DataAnnotations.Schema;

namespace ProductCatalogAPI.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string? Name { get; set; }
        public string? Description { get; set; }

        [Column(TypeName = "decimal(18, 2)")]
        public decimal Price { get; set; }
        public int StockQuantity { get; set; }
    }
}

```

Controller: ProductsController.cs

```

using Microsoft.AspNetCore.Mvc;
using ProductCatalogAPI.Models;

namespace ProductCatalogAPI.Controllers

```

```

{
    [ApiController]
    [Route("api/[controller]")]
    public class ProductsController : ControllerBase
    {
        private static readonly List<Product> _products = new List<Product>
        {
            new Product { Id = 1, Name = "Laptop", Description = "A powerful laptop.", Price = 1200.50m, StockQuantity = 50 },
            new Product { Id = 2, Name = "Mouse", Description = "A wireless mouse.", Price = 25.00m, StockQuantity = 200 }
        };
        private static int _nextId = 3;

        [HttpGet]
        public ActionResult<IEnumerable<Product>> GetProducts()
        {
            return Ok(_products);
        }

        [HttpGet("{id}")]
        public ActionResult<Product> GetProduct(int id)
        {
            var product = _products.FirstOrDefault(p => p.Id == id);
            if (product == null)
            {
                return NotFound();
            }
            return Ok(product);
        }

        [HttpPost]
        public ActionResult<Product> PostProduct(Product product)
        {
            product.Id = _nextId++;
            _products.Add(product);
            return CreatedAtAction(nameof(GetProduct), new { id = product.Id },
product);
        }

        [HttpPut("{id}")]
        public IActionResult PutProduct(int id, Product product)
        {
            if (id != product.Id)
            {
                return BadRequest();
            }

            var existingProduct = _products.FirstOrDefault(p => p.Id == id);
            if (existingProduct == null)
            {
                return NotFound();
            }

            existingProduct.Name = product.Name;

```

```

        existingProduct.Description = product.Description;
        existingProduct.Price = product.Price;
        existingProduct.StockQuantity = product.StockQuantity;

        return NoContent();
    }

    [HttpDelete("{id}")]
    public IActionResult DeleteProduct(int id)
    {
        var product = _products.FirstOrDefault(p => p.Id == id);
        if (product == null)
        {
            return NotFound();
        }

        _products.Remove(product);
        return NoContent();
    }
}

```

Ödev 5: Müşteri Kayıt API'si

Model: Customer.cs

```

namespace CustomerRegistryAPI.Models
{
    public class Customer
    {
        public int Id { get; set; }
        public string? FirstName { get; set; }
        public string? LastName { get; set; }
        public string? Email { get; set; }
        public string? PhoneNumber { get; set; }
    }
}

```

Controller: CustomersController.cs

```

using Microsoft.AspNetCore.Mvc;
using CustomerRegistryAPI.Models;

namespace CustomerRegistryAPI.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class CustomersController : ControllerBase
    {
        private static readonly List<Customer> _customers = new List<Customer>
        {
            new Customer { Id = 1, FirstName = "John", LastName = "Doe", Email = "john.doe@example.com", PhoneNumber = "123-456-7890" },
        }
    }
}

```

```

        new Customer { Id = 2, FirstName = "Jane", LastName = "Smith", Email
= "jane.smith@example.com", PhoneNumber = "098-765-4321" }
    };
    private static int _nextId = 3;

    [HttpGet]
    public ActionResult<IEnumerable<Customer>> GetCustomers()
    {
        return Ok(_customers);
    }

    [HttpGet("{id}")]
    public ActionResult<Customer> GetCustomer(int id)
    {
        var customer = _customers.FirstOrDefault(c => c.Id == id);
        if (customer == null)
        {
            return NotFound();
        }
        return Ok(customer);
    }

    [HttpPost]
    public ActionResult<Customer> PostCustomer(Customer customer)
    {
        customer.Id = _nextId++;
        _customers.Add(customer);
        return CreatedAtAction(nameof(GetCustomer), new { id = customer.Id
}, customer);
    }

    [HttpPut("{id}")]
    public IActionResult PutCustomer(int id, Customer customer)
    {
        if (id != customer.Id)
        {
            return BadRequest();
        }

        var existingCustomer = _customers.FirstOrDefault(c => c.Id == id);
        if (existingCustomer == null)
        {
            return NotFound();
        }

        existingCustomer.FirstName = customer.FirstName;
        existingCustomer.LastName = customer.LastName;
        existingCustomer.Email = customer.Email;
        existingCustomer.PhoneNumber = customer.PhoneNumber;

        return NoContent();
    }

    [HttpDelete("{id}")]
    public IActionResult DeleteCustomer(int id)

```

```
{
    var customer = _customers.FirstOrDefault(c => c.Id == id);
    if (customer == null)
    {
        return NotFound();
    }

    _customers.Remove(customer);
    return NoContent();
}
}
```