Code Academy

Dėstytojas

Vilmantas Neviera

.NET API Projektas

Data

Šiandien išmoksite

01 Kas yra API?

02 REST APi

03 Projekto struktūra

04 HTTP užklausos

Prerequisites

Postman (that’s it)

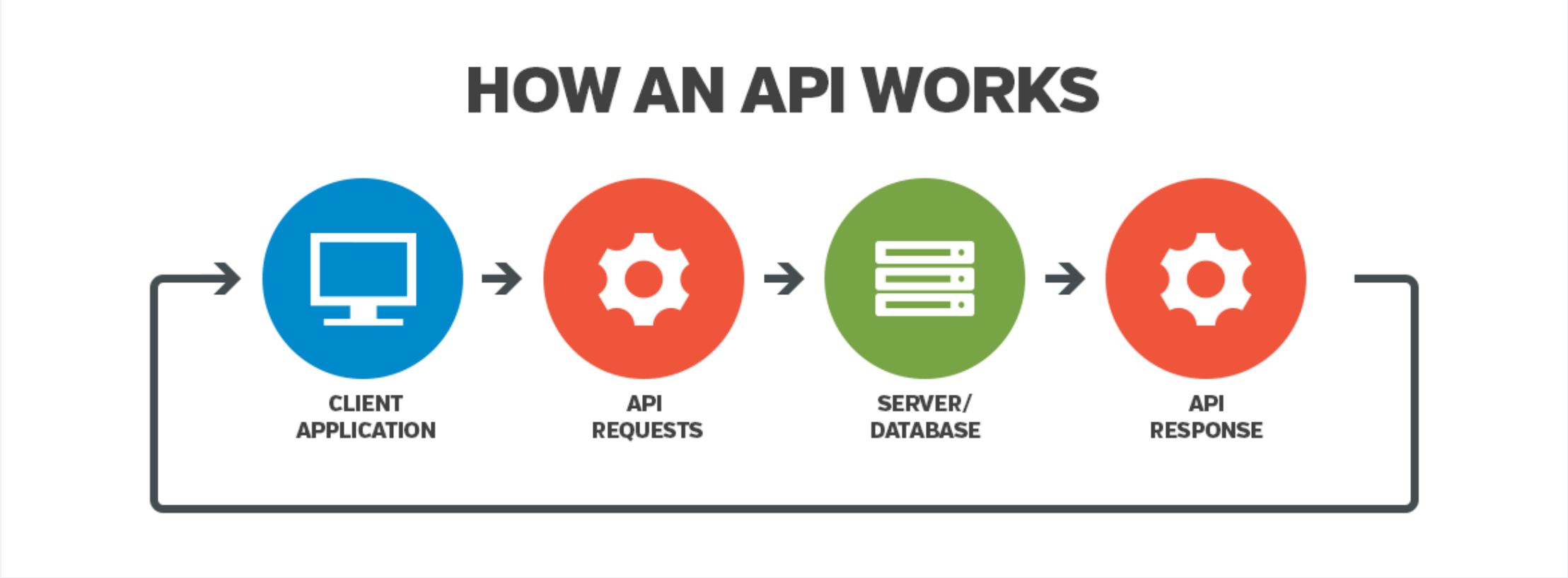
https://aws.amazon.com/what-is/api/

Kas yra API?

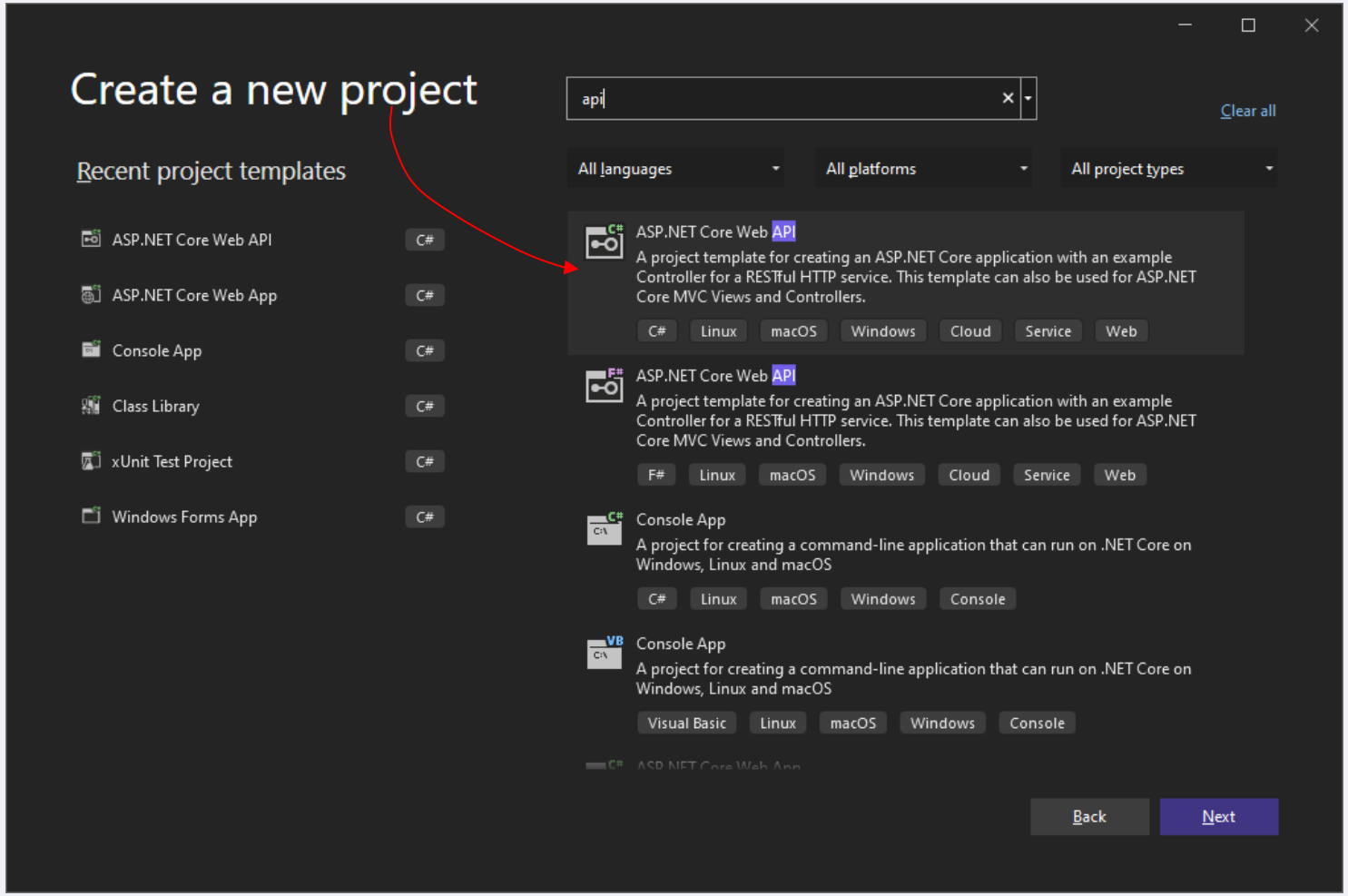
APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols. For example, the weather bureau's software system contains daily weather data. The weather app on your phone "talks" to this system via APIs and shows you daily weather updates on your phone.

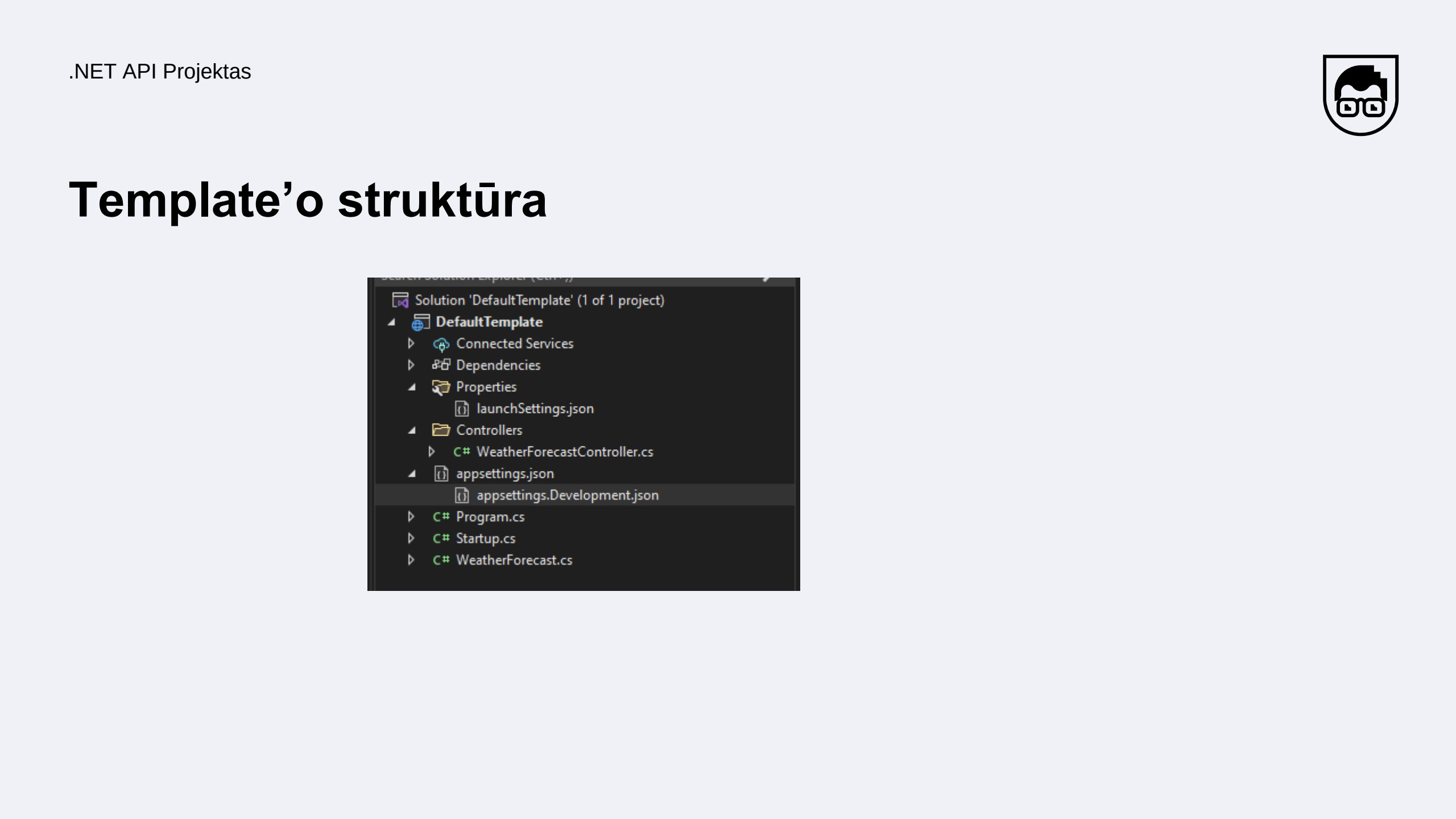
https://aws.amazon.com/what-is/api/

Kas yra API?


HOW AN API WORKS
CLIENT APPLICATION
API REQUESTS
SERVER/ DATABASE
API RESPONSE

Susikuriame API projektą





Template’o struktūra

WeatherForecastController

```
namespace DefaultTemplate.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class WeatherForecastController : ControllerBase
    {
        private static readonly string[] Summaries = new[]
        {
            "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"
        };

        private readonly ILogger<WeatherForecastController> _logger;

        public WeatherForecastController(ILogger<WeatherForecastController> logger)
        {
            _logger = logger;
        }

        [HttpGet]
        public IEnumerable<WeatherForecast> Get()
        {
            var rng = new Random();
            return Enumerable.Range(1, 5).Select(index => new WeatherForecast
            {
                Date = DateTime.Now.AddDays(index),
                TemperatureC = rng.Next(-20, 55),
                Summary = Summaries[rng.Next(Summaries.Length)]
            })
            .ToArray();
        }
    }
}
```

appsettings.json

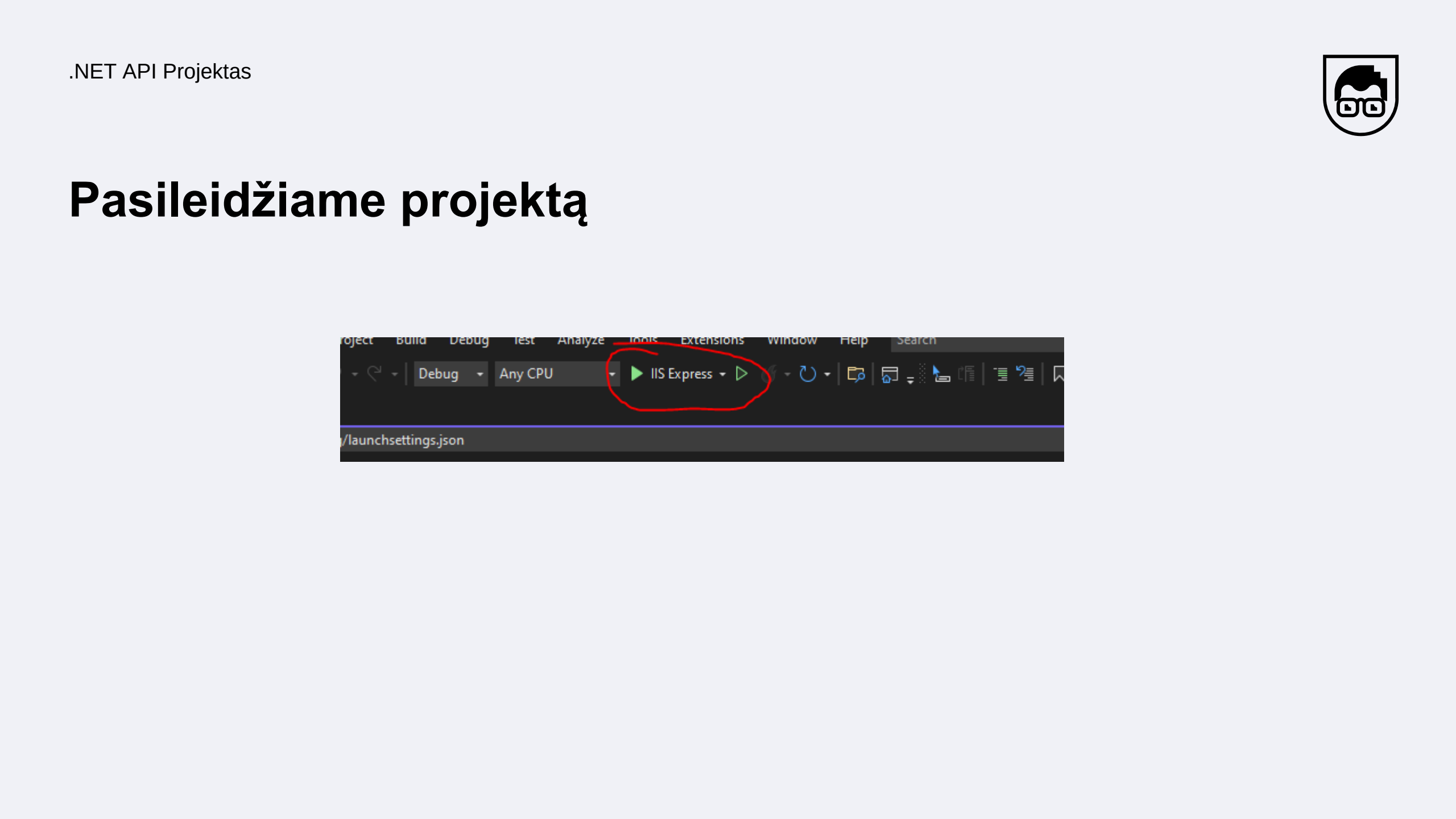
```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  }
}
```

Program.cs

```
namespace DefaultTemplate
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

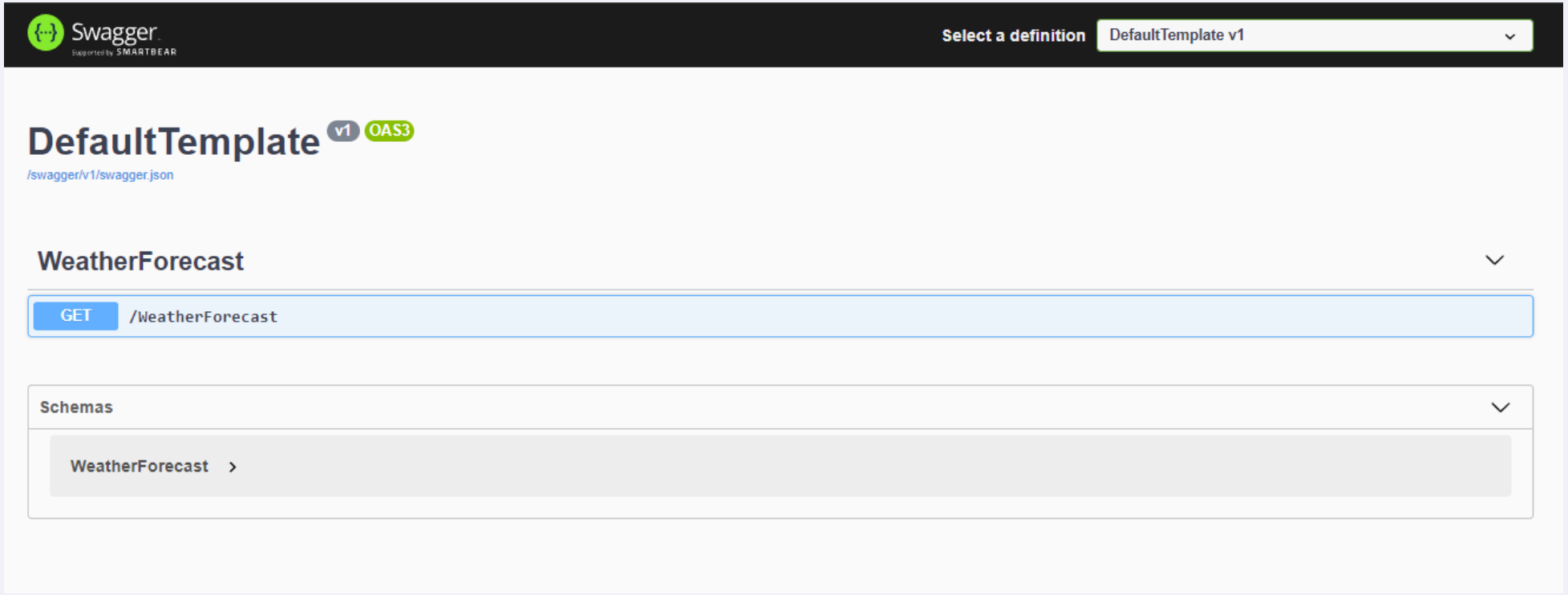
Startup.cs (nebijokit)

```
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services)
    {

        services.AddControllers();
        services.AddSwaggerGen(c =>
        {
            c.SwaggerDoc("v1", new OpenApiInfo { Title = "DefaultTemplate", Version = "v1" });
        });
    }

    // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
            app.UseSwagger();
            app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "DefaultTemplate v1"));
        }

        app.UseHttpsRedirection();

        app.UseRouting();

        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
        });
    }
}
```

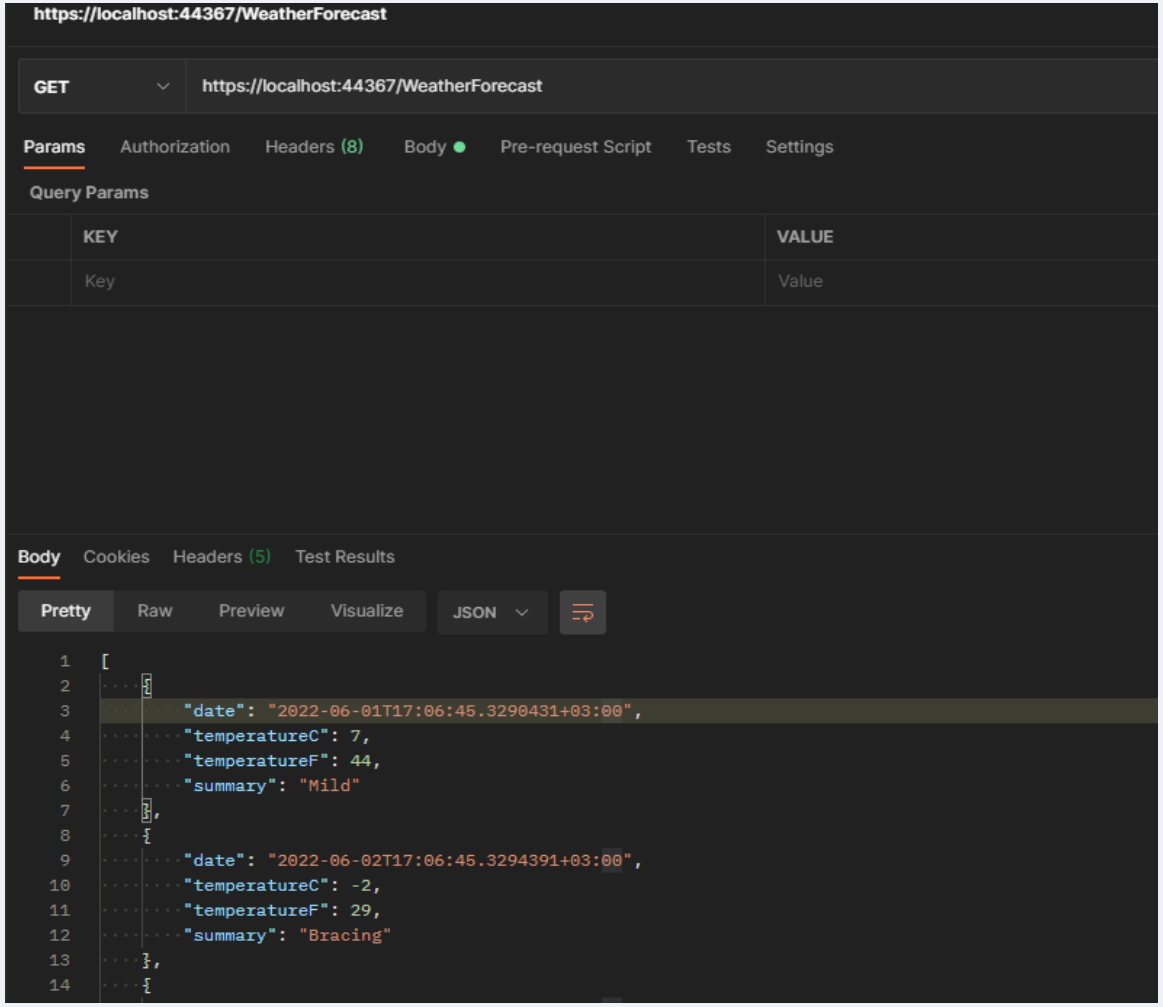


Pasileidžiame projektą

Swashbuckle nuget’o dėka gauname tokį vaizdą



Tą patį galime pamatyti ir su Postman’u



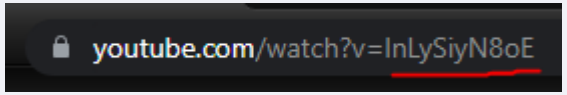
Kas yra REST servisas?

Terminai:

1. Client - klientas yra žmogus/programa naudojanti mūsų API. Klientas išsiunčia **HTTP** užklausą norėdamas gauti, išsaugoti ar atnaujinti informaciją.
2. Resursas - tai informacija, kurią API gali suteikti klientui, Facebook’o resursai būtų klientai, nuotraukos ir t.t.
3. Server - serveris yra vieta į kurią kreipiasi API gauti/atnaujinti resursus klientui.

Daugiau iš teorinės pusės: https://www.redhat.com/en/topics/api/what-is-a-rest-api

Kas yra HTTP užklausos?



Pagrindinės HTTP užklausos:

1. GET - Skirtas gauti duomenis iš serverio, norint patikslinti užklausą gali “atsinešti” su savimi parametrą per **url**
2. POST - Skirtas išsaugoti naujus duomenis serveryje. Didžiausias skirtumas , kad duomenys nešasi savo kūne(body) ir naudojant SSL apsaugas šis būna užšifruotas, todėl informacija yra daug saugiau gabenama.
3. PUT - Skirtas atnaujinti duomenis. Technologiškai jo struktūra yra tokia pati kaip POST, bet pagal REST principus PUT užklausa turi būti **idemptotent** - tai reiškia jog pasiuntus užklausą daugiau negu vieną kartą rezultatas po pirmojo karto neturi keistis.
4. Delete - Skirtas ištrinti įrašą, galima naudoti body, bet nepatariama.

Taip pat kiekviena HTTP užklausa su savimi turi ir Header sekciją

Daugiau iš teorinės pusės: https://www.redhat.com/en/topics/api/what-is-a-rest-api

Kas yra Controller?

Controller’is yra klasė, kurioje apsirašome **endpoint’us** į kuriuos ateis **HTTP** užklausos.

Pvz WeatherForecast Get() controller’is su endpointu ‘https://localhost:44367/WeatherForecast’

Klausimai

.NET API Projektas

Užduotis nr. 1

- Sukurkite paprastą web puslapį, kuris siųs GET PUT POST DELETE užklausas objektus į jūsų sukurtą naują Controller’į
- Controlleris turės jūsų sukurtų objektų List’ą ir operuos pagal gautos užklausos tipą