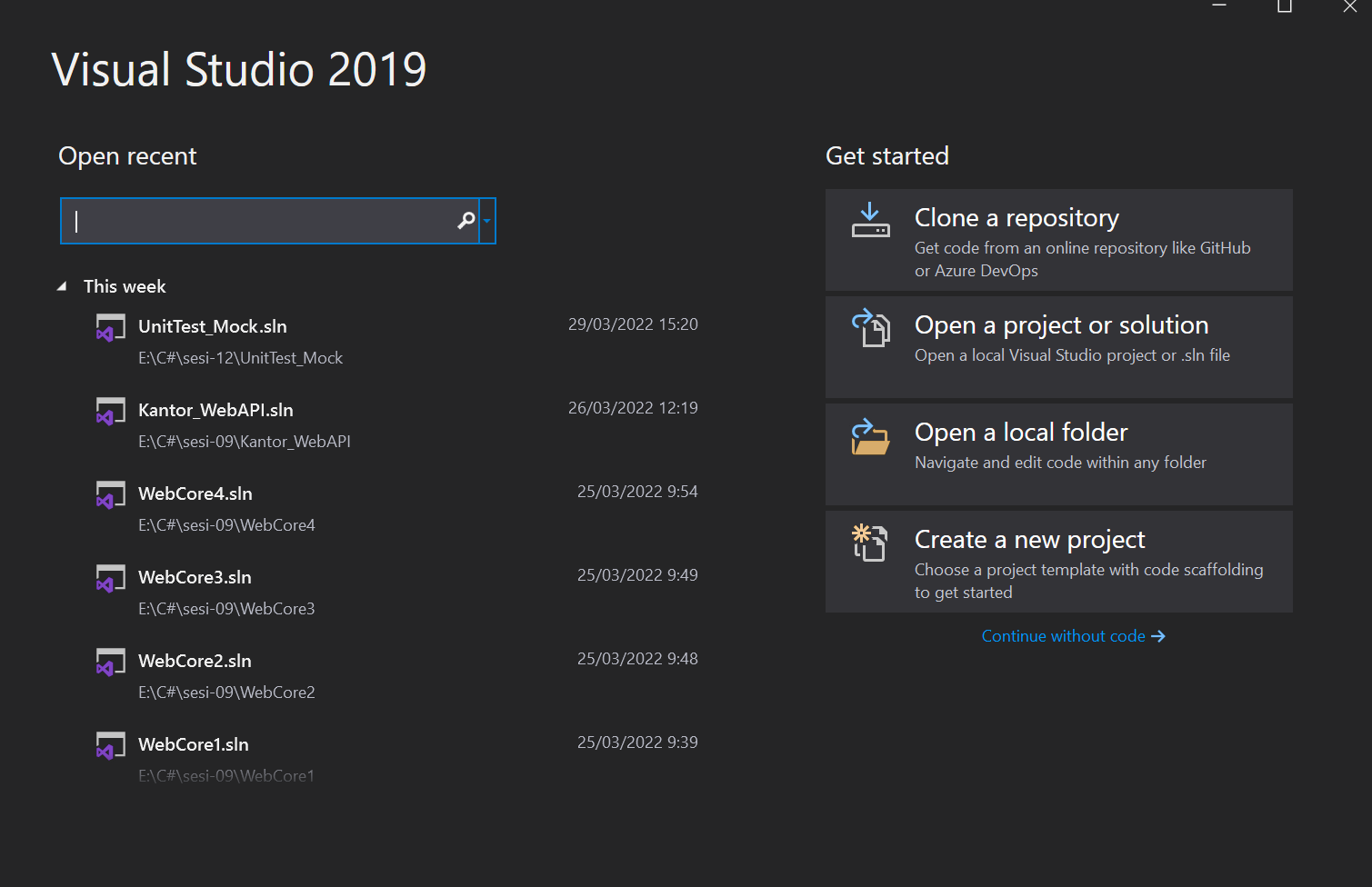
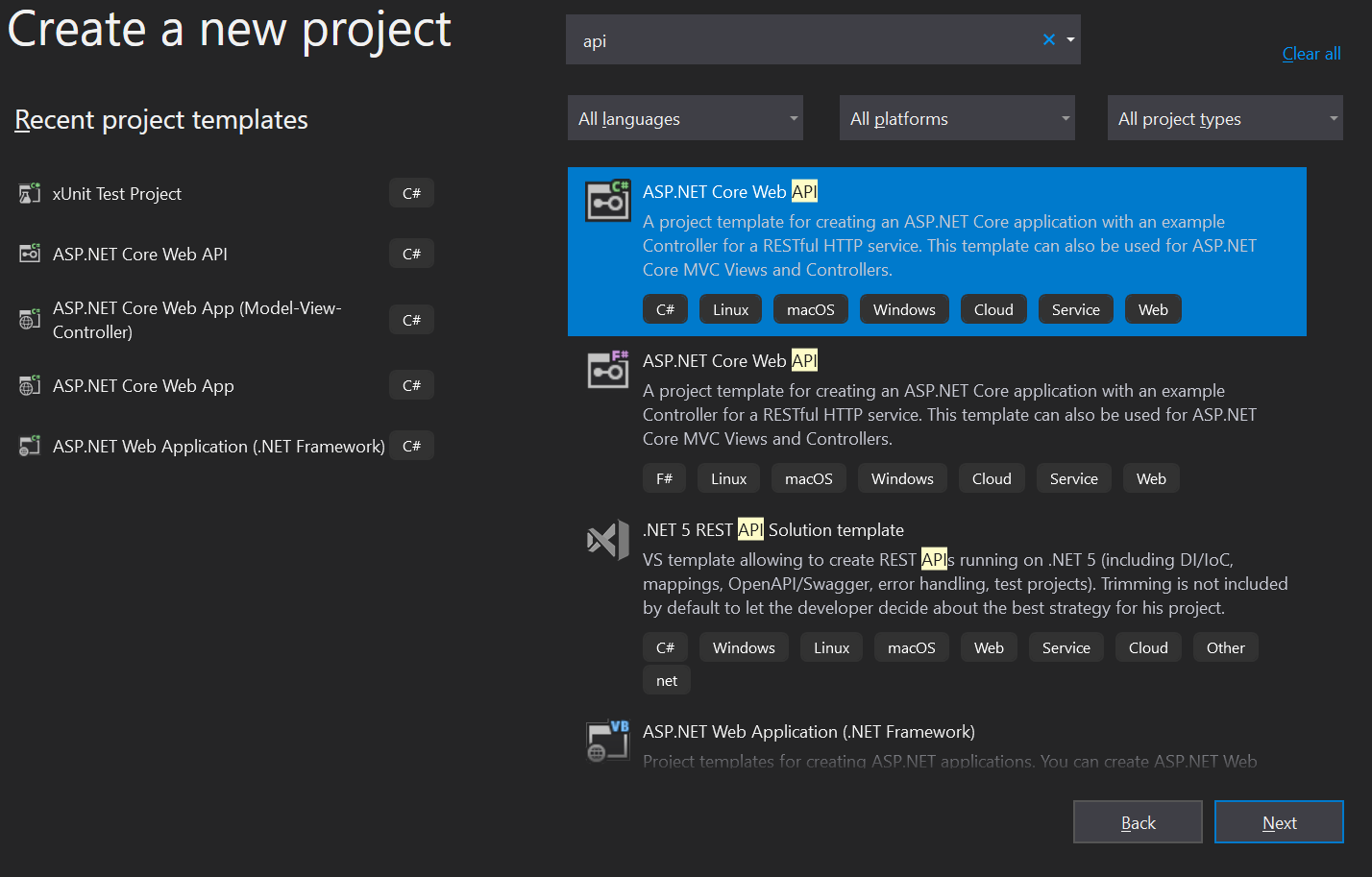
Nama : Alexander Radianta Tarigan

No\_Peserta : FSDO003ONL010

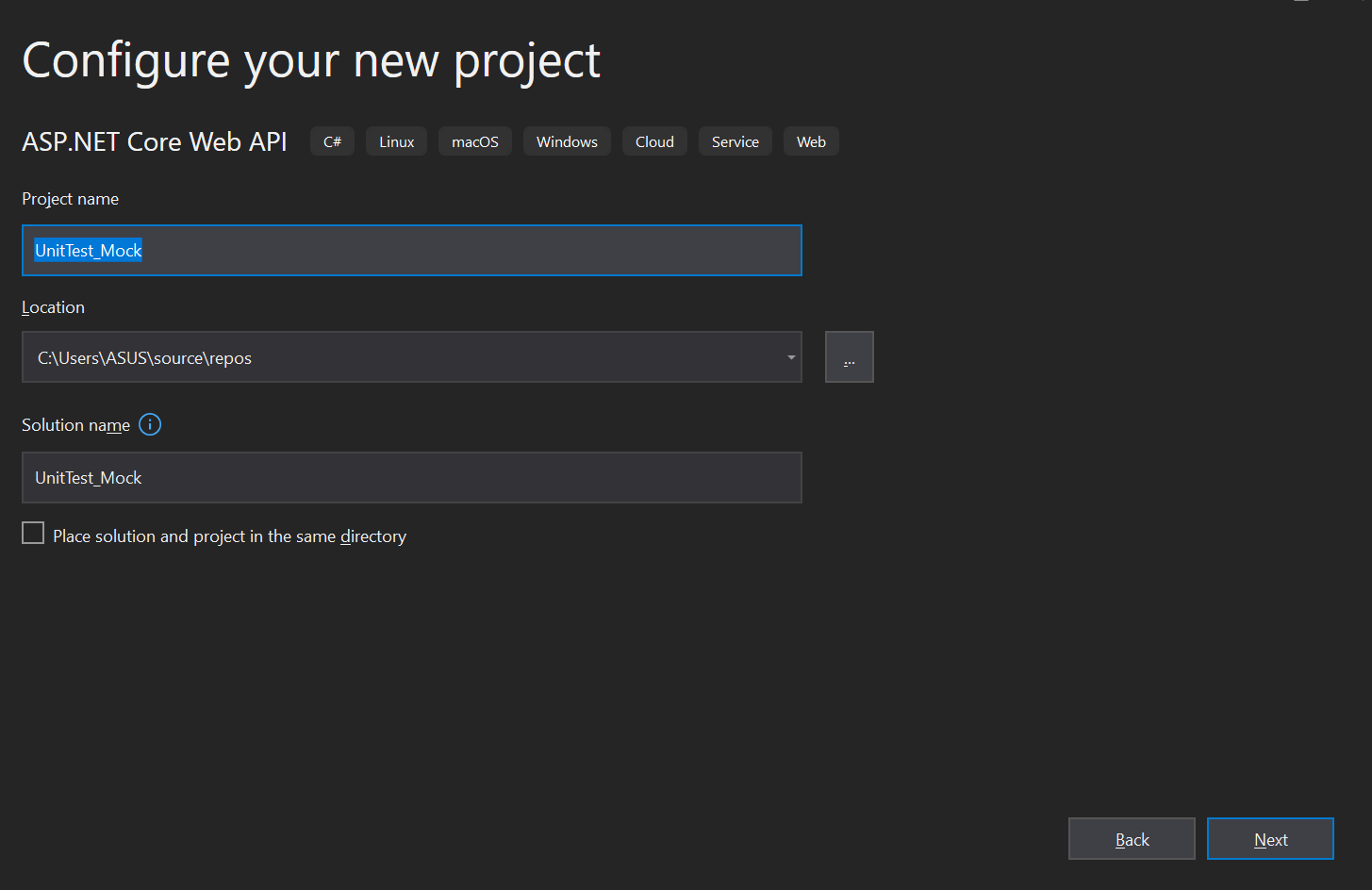
Klik tombol create a new project



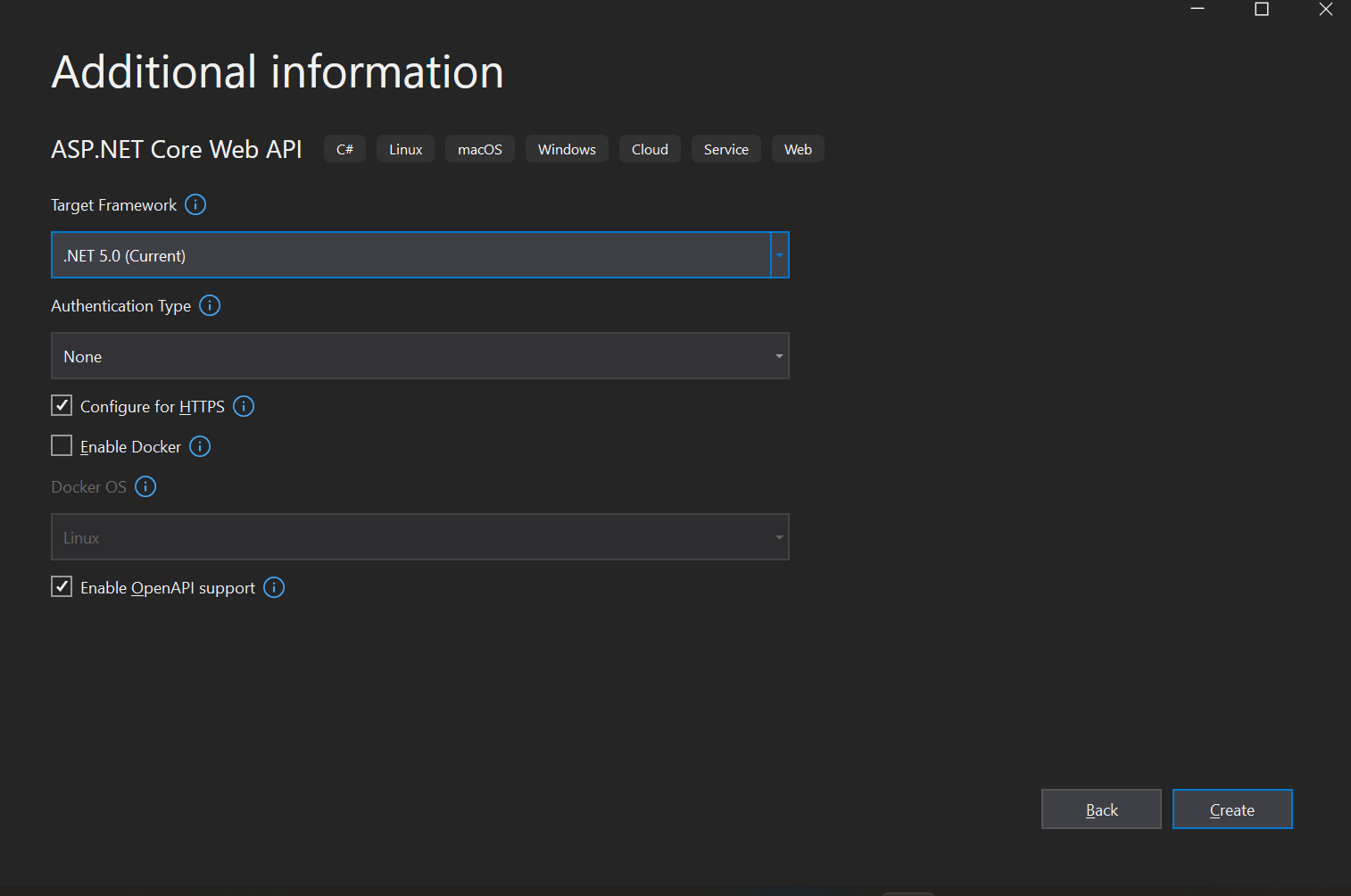
Pilih ASP .NET Core Web API



Masukkan nama project, lalu klik tombol next



Pilih Framework .NET 5.0, lalu centai Configure for HTTPS dan centang Enable Open-API Support, dan klik tombol create



Buat Folder Models, lalu buat class Employee.cs

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Threading.Tasks;

namespace UnitTest\_Mock.Models

{

    public class Employee

    {

        [Key]

        public int Id { get; set; }

        public string Name { get; set; }

        public string Destination { get; set; }

    }

}

buat 1 class bernama ApiDbContext.cs di folder Models

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

namespace UnitTest\_Mock.Models

{

    public partial class AppDbContext : DbContext

    {

        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)

        {

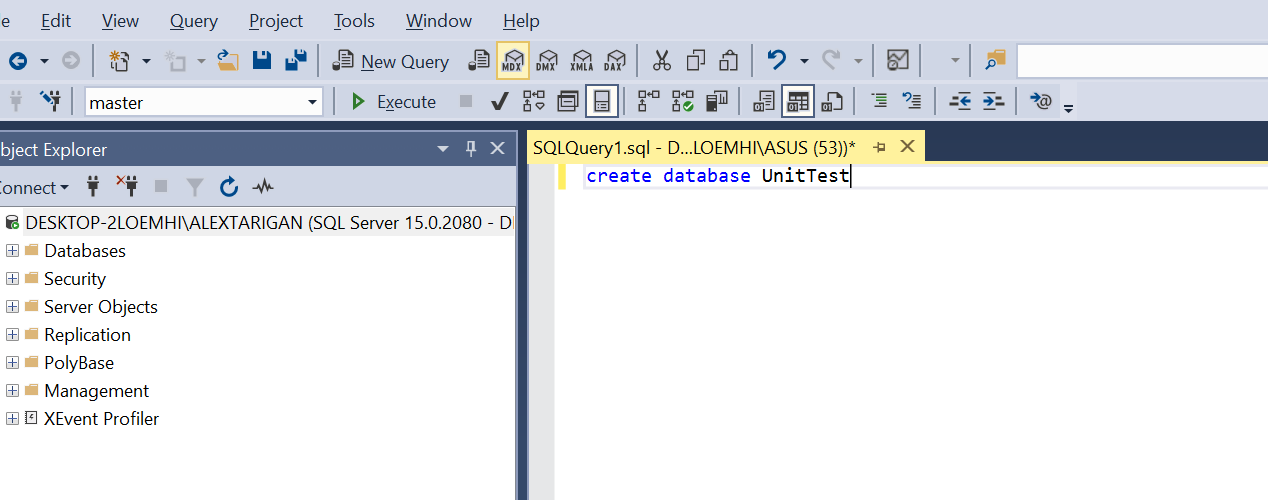
        }

        public DbSet<Employee> Employees { get; set; }

    }

}

Lalu buat database di sql server



Selanjutnya melakukan setting server dll di appsetings.json, agar database dapat digunakan

{

  "ConnectionStrings": {

    "myconn": "Server=DESKTOP-2LOEMHI\\ALEXTARIGAN; Database=UnitTest; Trusted\_Connection=true"

  },

  "Logging": {

    "LogLevel": {

      "Default": "Information",

      "Microsoft": "Warning",

      "Microsoft.Hosting.Lifetime": "Information"

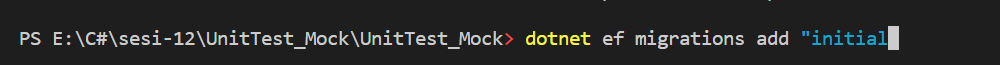
    }

  },

  "AllowedHosts": "\*"

}

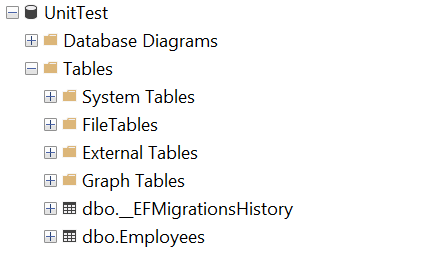
Lakukan perintah dotnet ef migrations add “initial” untuk melakukan migrasi database



Lakukan perintah dotnet ef database update



Jika berhasil, akan ada table baru yang dibuat di database kita



Buat folder baru bernama services, dan buat class baru bername IEmployeeService.cs

namespace UnitTest\_Mock.Services

{

    public interface IEmployeeService

    {

        Task<string> GetEmployeeById(int EmpID);

        Task<Employee> GetEmployeeDetails(int EmpID);

    }

}

Lalu buat class baru bernama EmployeeService di folder services tadi.

public class EmployeeService : IEmployeeService

    {

        #region Property

        private readonly AppDbContext \_appDbContext;

        #endregion

        #region  Constructor

        public EmployeeService(AppDbContext appDbContext)

        {

            \_appDbContext = appDbContext;

        }

        #endregion

        public async Task<string> GetEmployeeById(int EmpID)

        {

            var name = await \_appDbContext.Employees.Where(c=>c.Id == EmpID).Select(d=> d.Name).FirstOrDefaultAsync();

            return name;

        }

        public async Task<Employee> GetEmployeeDetails(int EmpID)

        {

            var emp = await \_appDbContext.Employees.FirstOrDefaultAsync(c => c.Id == EmpID);

            return emp;

        }

Lalu buat folder baru bernama Controllers, dan buat kelas baru bernama EmployeeController.cs

namespace UnitTest\_Mock.Controllers

{

    [Route("api/[controller]")]

    [ApiController]

    public class EmployeeController : ControllerBase

    {

        #region Property

        private readonly IEmployeeService \_employeeService;

        #endregion

        #region Constructor

        public EmployeeController(IEmployeeService employeeService)

        {

            \_employeeService = employeeService;

        }

        #endregion

        [HttpGet(nameof(GetEmployeeById))]

        public async Task<string> GetEmployeeById(int EmpID)

        {

            var result = await \_employeeService.GetEmployeeById(EmpID);

            return result;

        }

        [HttpGet(nameof(GetEmployeeDetails))]

        public async Task<Employee> GetEmployeeDetails(int EmpID)

        {

            var result = await \_employeeService.GetEmployeeDetails(EmpID);

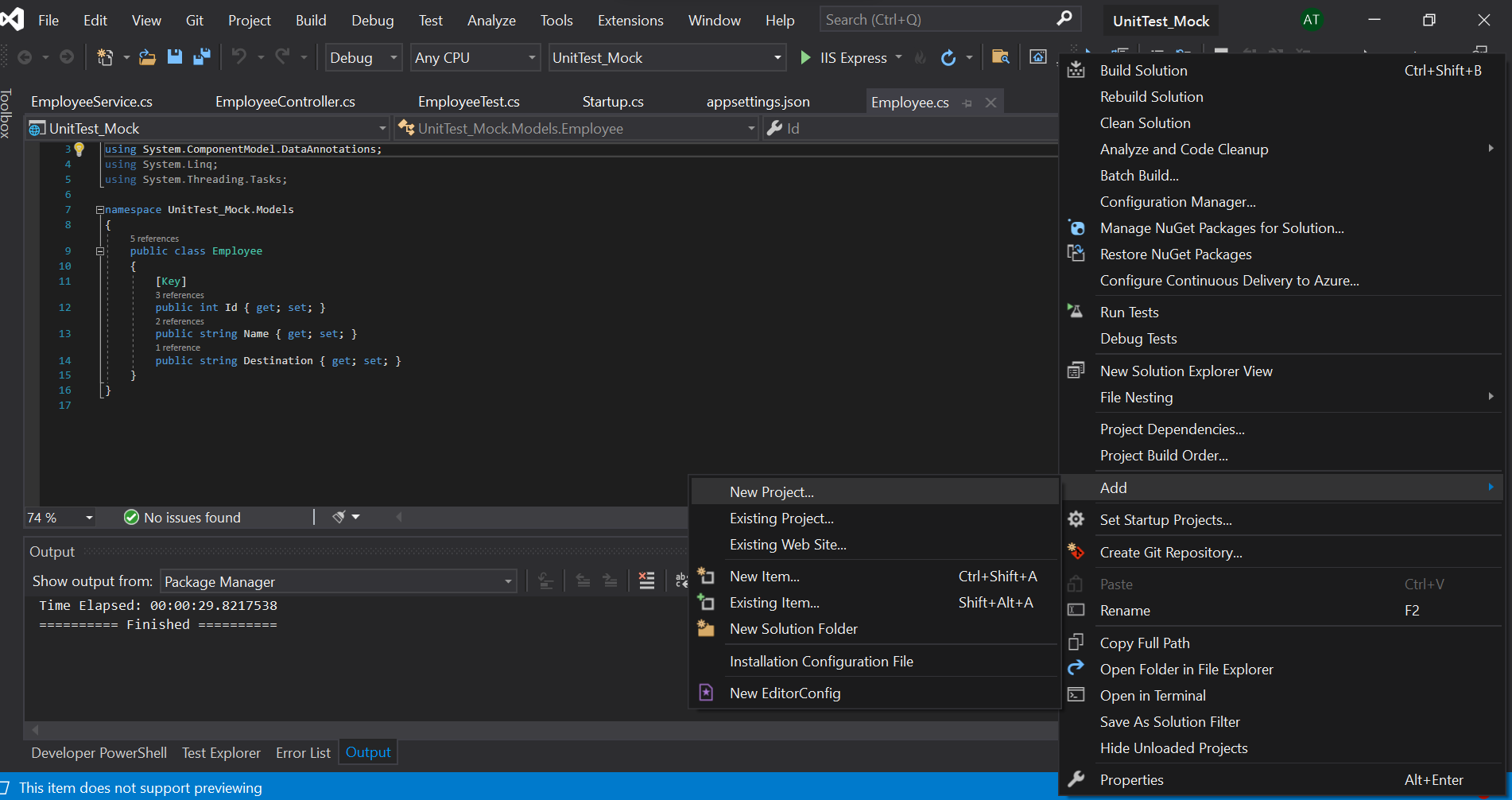
            return result;

        }

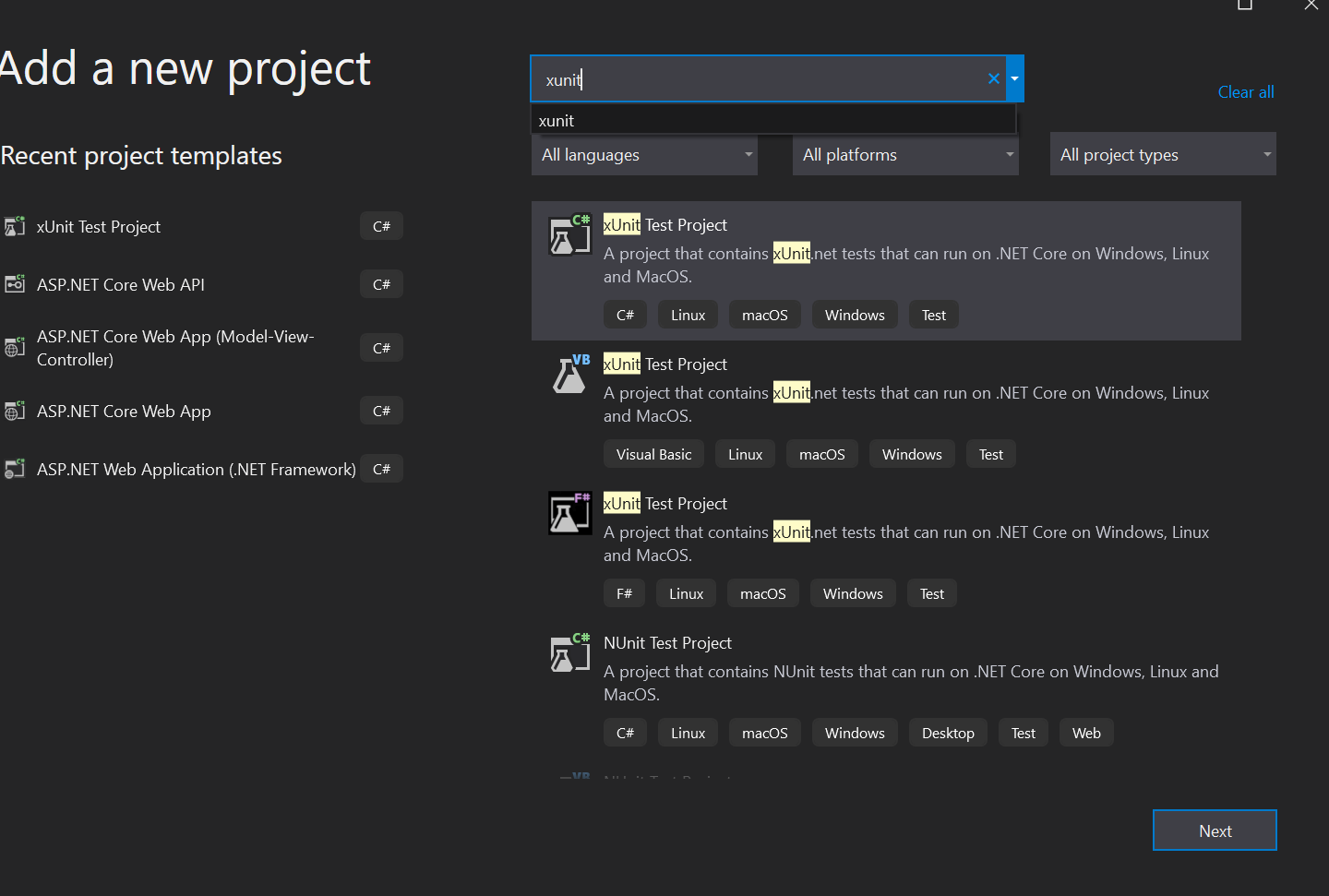
    }

}

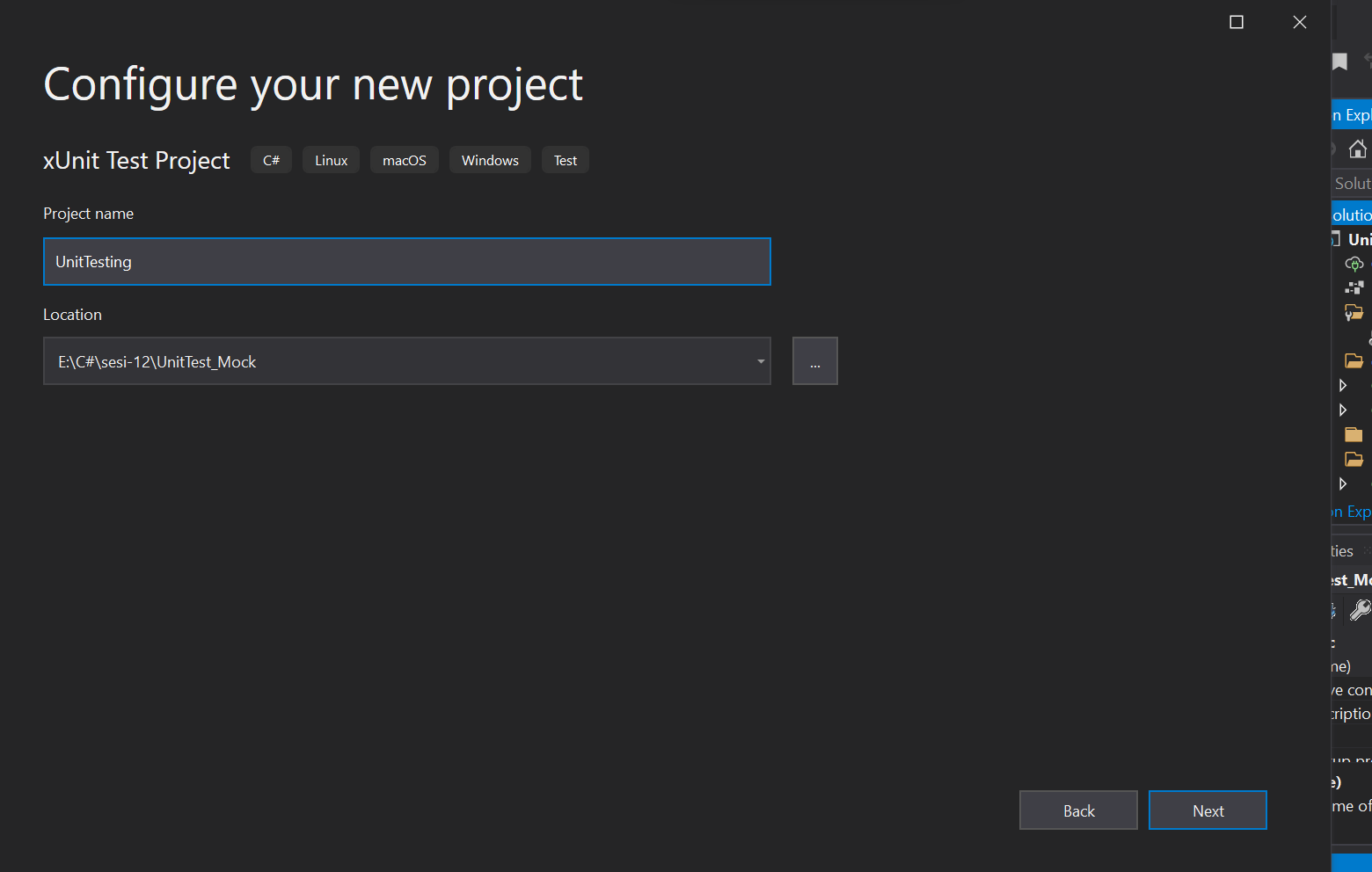
Selanjutnya tambah sebuah project



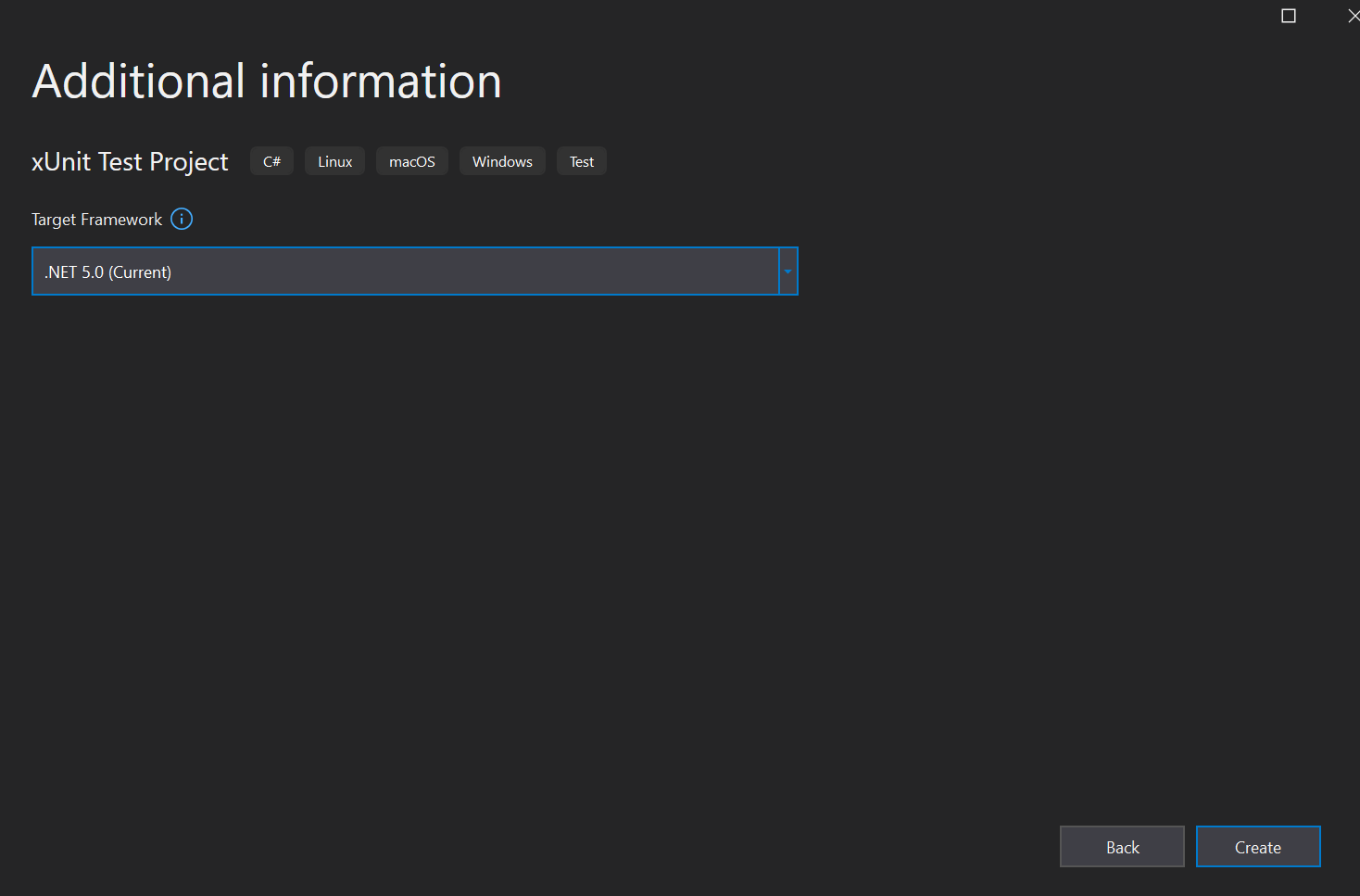
Lalu pilih xUnit Test Project, setelah itu klik tombol next



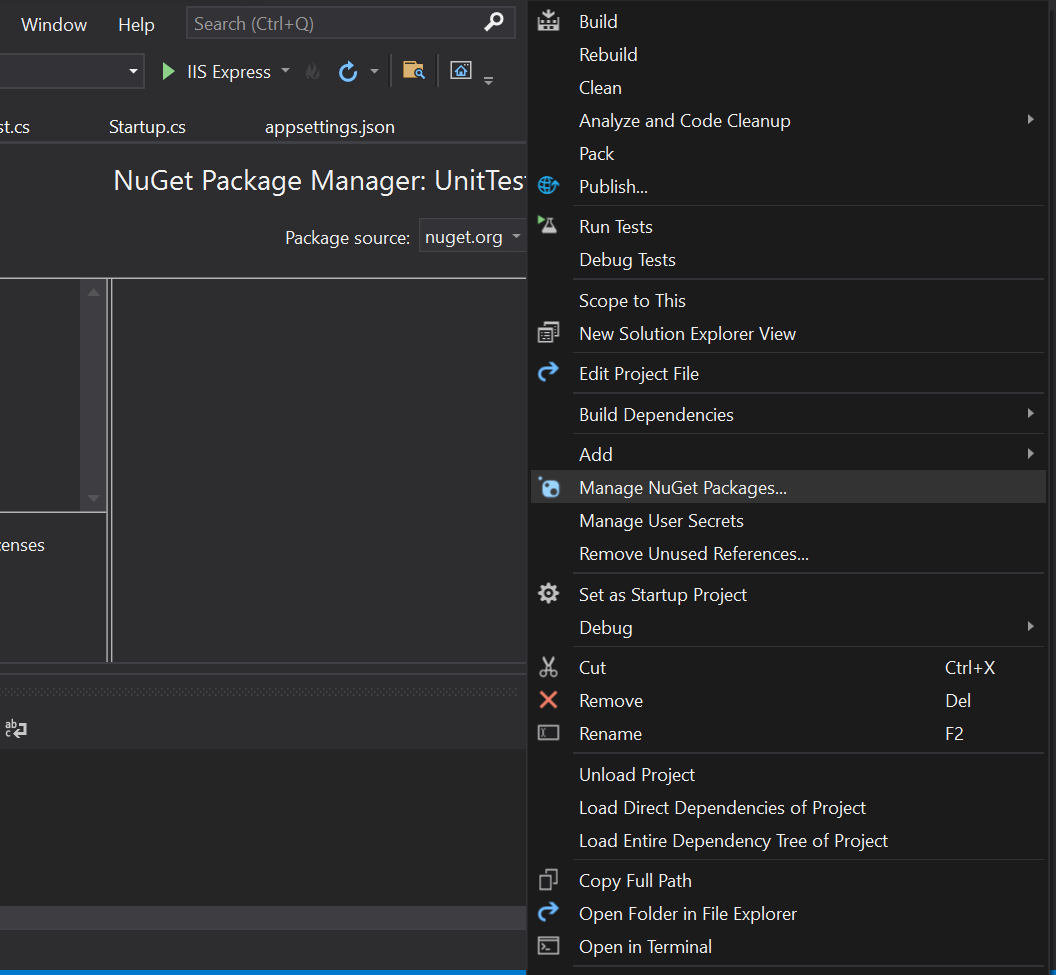
Beri nama UnitTesting, lalu klik next



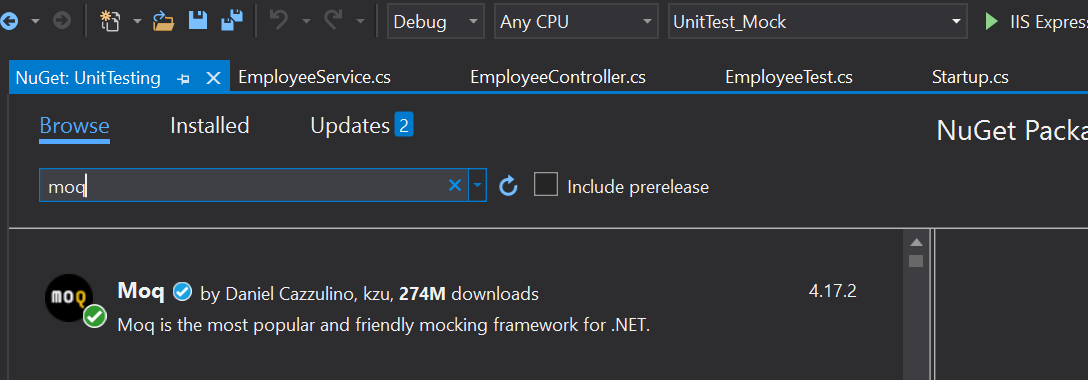
Pilih .NET 5.0, lalu klik tombol create



Klik kanan pada project UnitTesting, lalu klik manage nuget packages



Lalu install packages Moq



Lalu buat class baru bernama EmployeeTest.cs

public class EmployeeTest

    {

        #region Property

        public Mock<IEmployeeService> mock = new Mock<IEmployeeService>();

        #endregion

        [Fact]

        public async void GetEmployeeById()

        {

            mock.Setup(p =>p.GetEmployeeById(1)).ReturnsAsync("Alex");

            EmployeeController emp = new EmployeeController(mock.Object);

            string result = await emp.GetEmployeeById(1);

            Assert.Equal("Alex", result);

        }

        [Fact]

        public async void GetEmployeeDetails()

        {

            var employeeDTO = new Employee()

            {

                Id = 1,

                Name = "Alex",

                Destination = "Medan"

            };

            mock.Setup(p => p.GetEmployeeDetails(1)).ReturnsAsync(employeeDTO);

            EmployeeController emp = new EmployeeController(mock.Object);

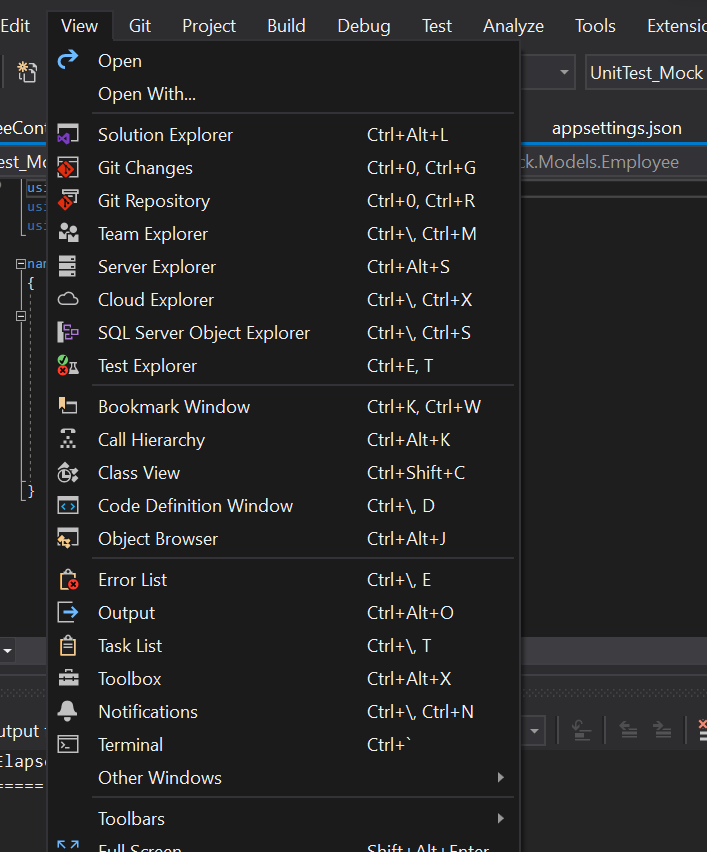
            var result = await emp.GetEmployeeDetails(1);

            Assert.True(employeeDTO.Equals(result));

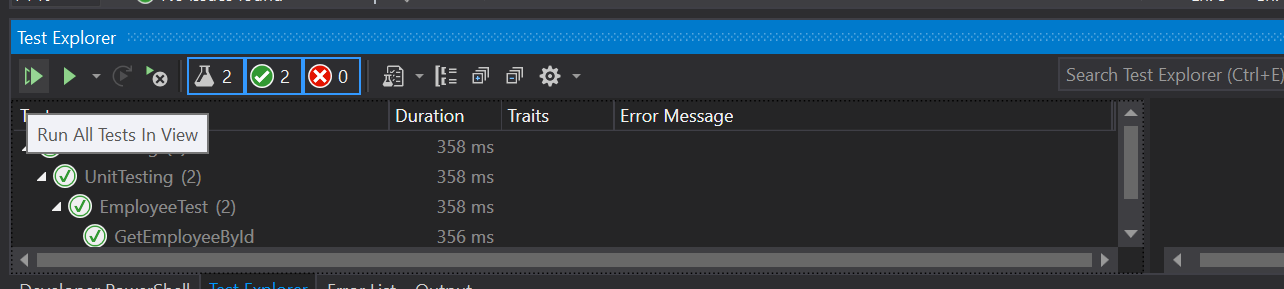
        }

    }

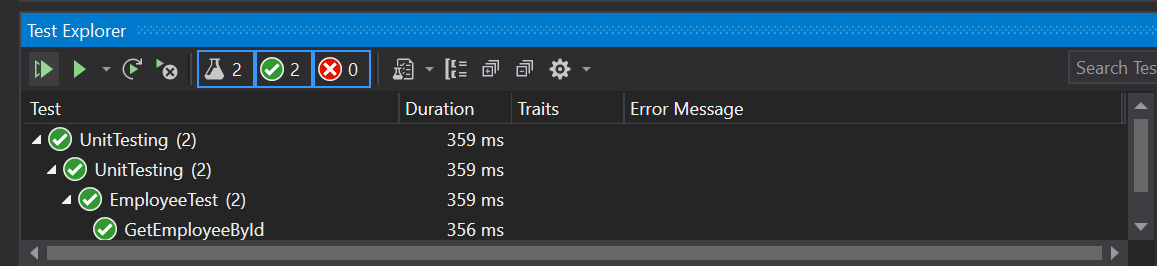
Lalu tambahkan test explorer di view



Jalankan unit test, dengan menekan tombol run

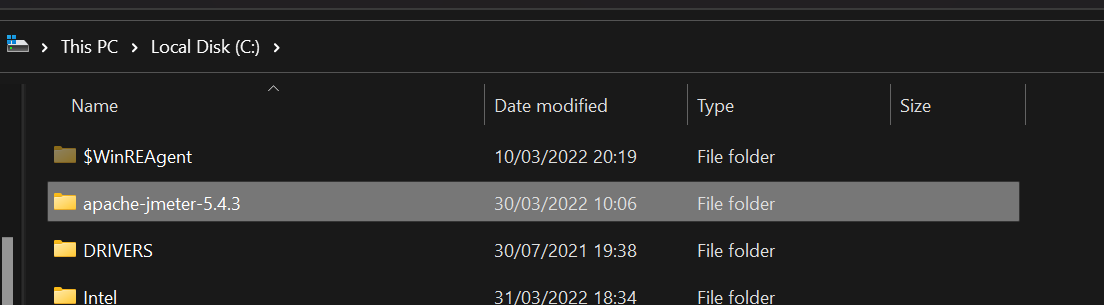


Unit test berhasil dijalankan.

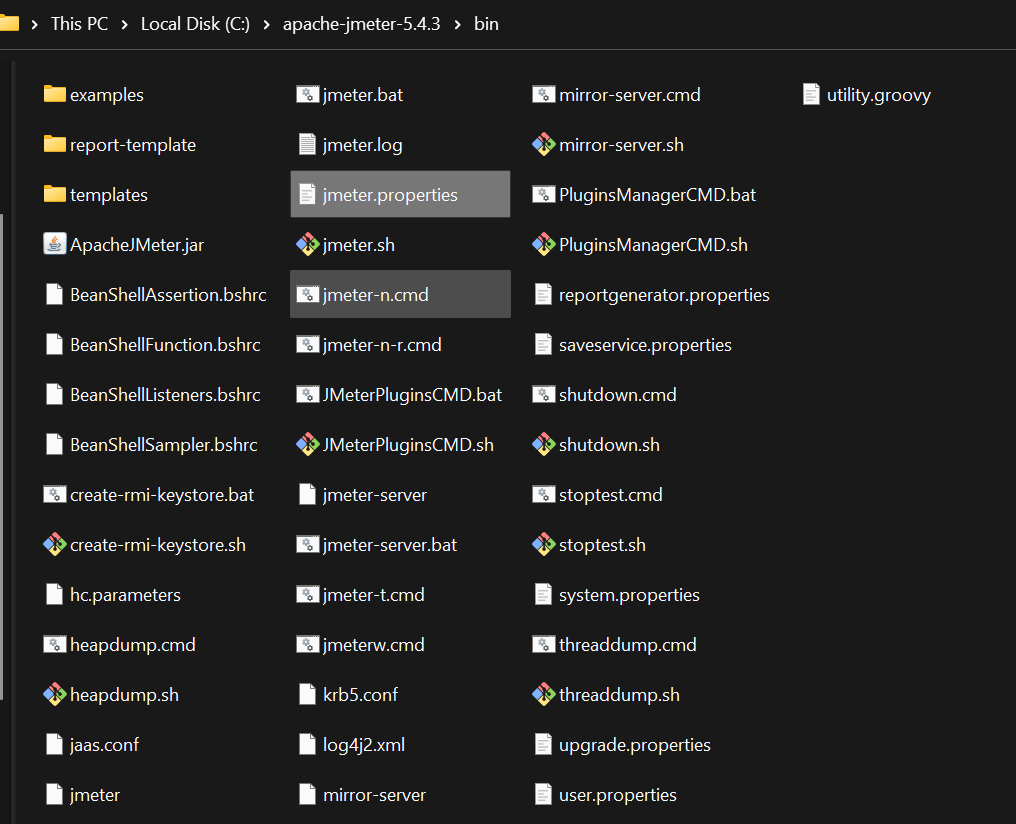


JMeter

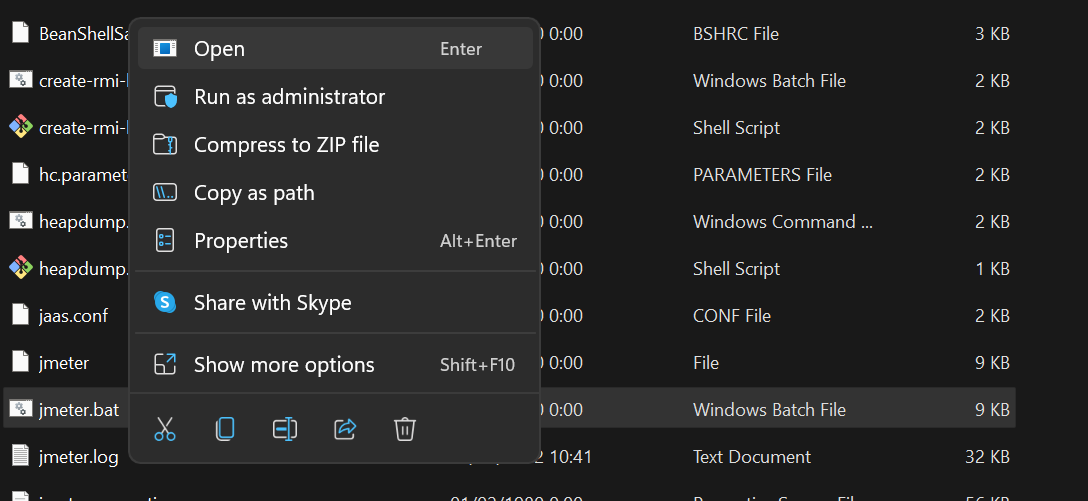
Letakkan File Jmeter di C:/



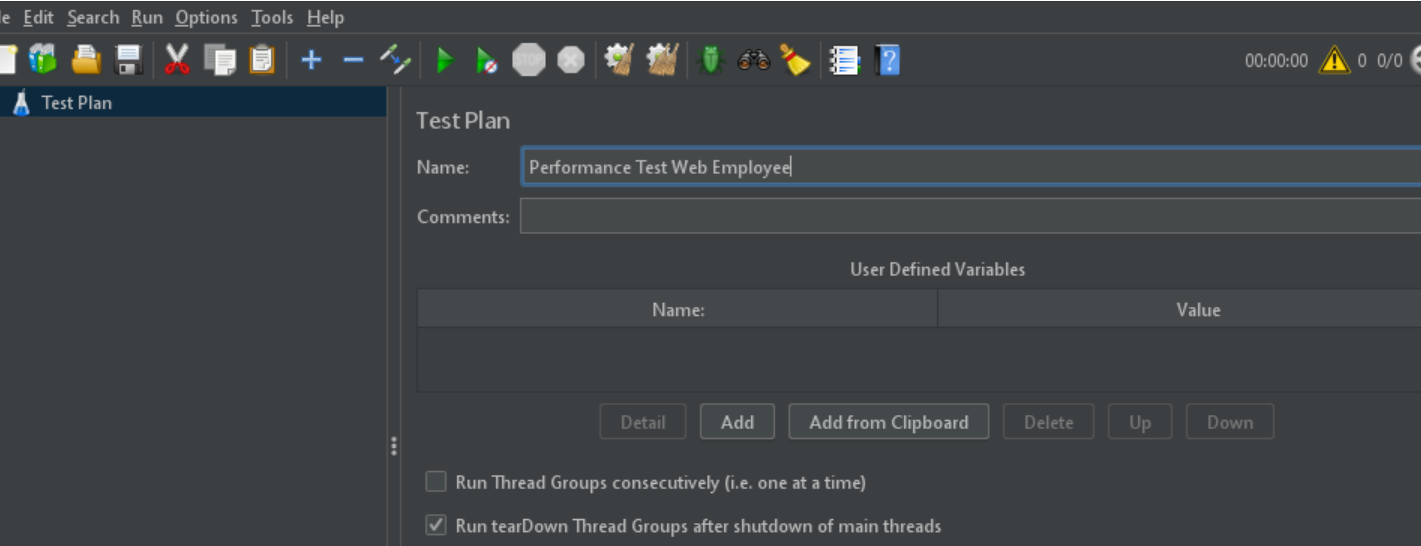
Buka folder Jmeter>bin



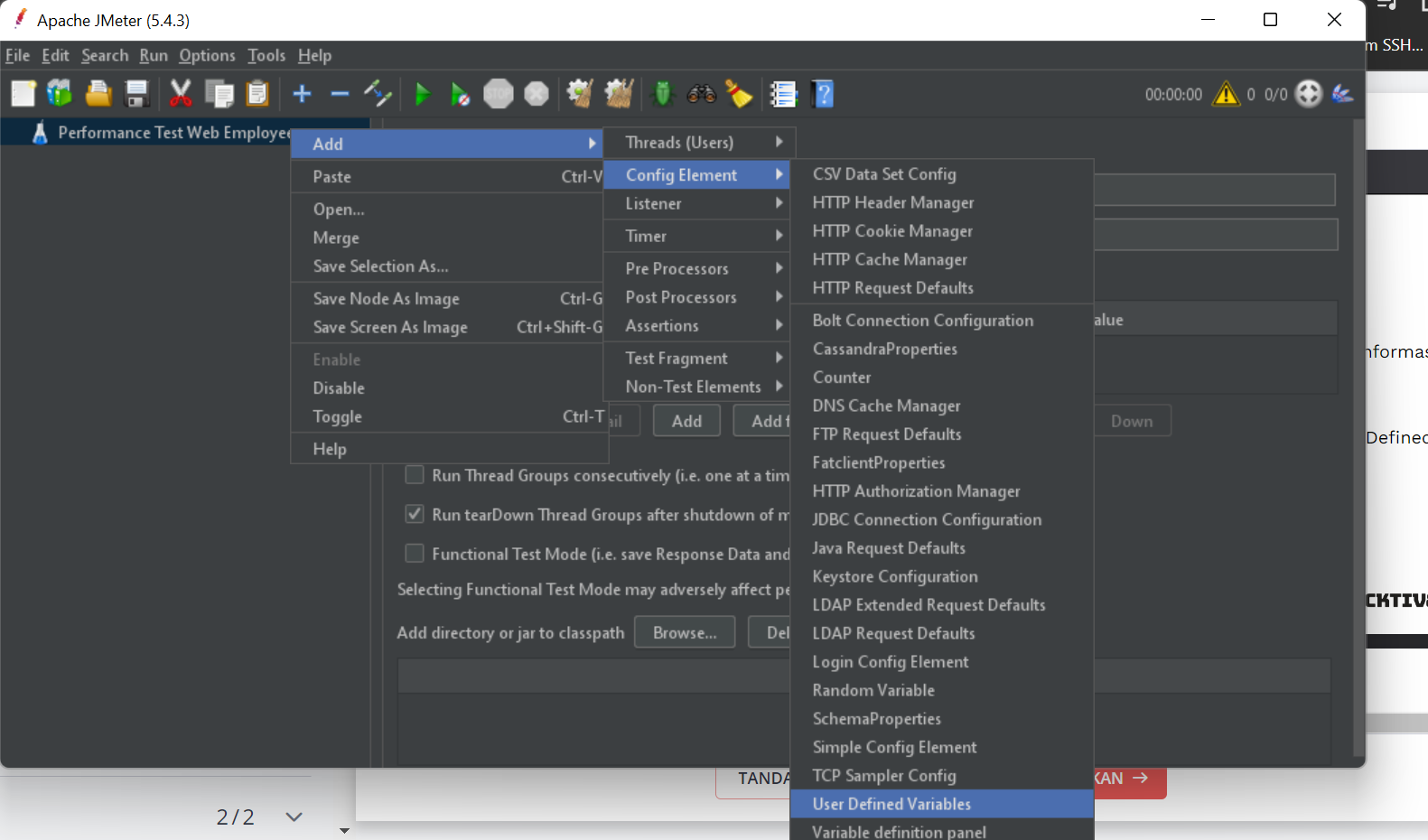
Jalankan file j-meter.bat



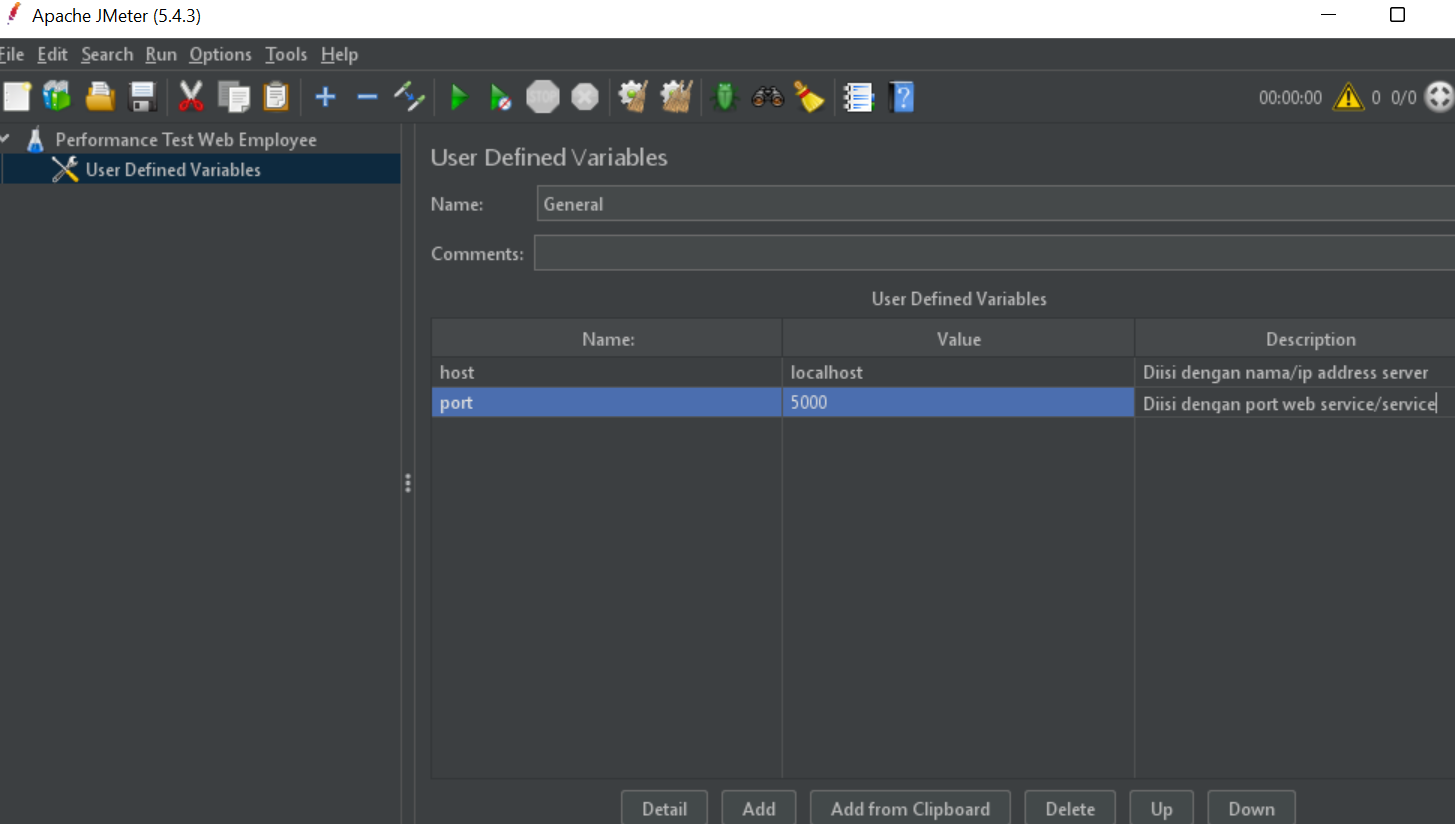
Ubah informasi name jadi performance test web employee



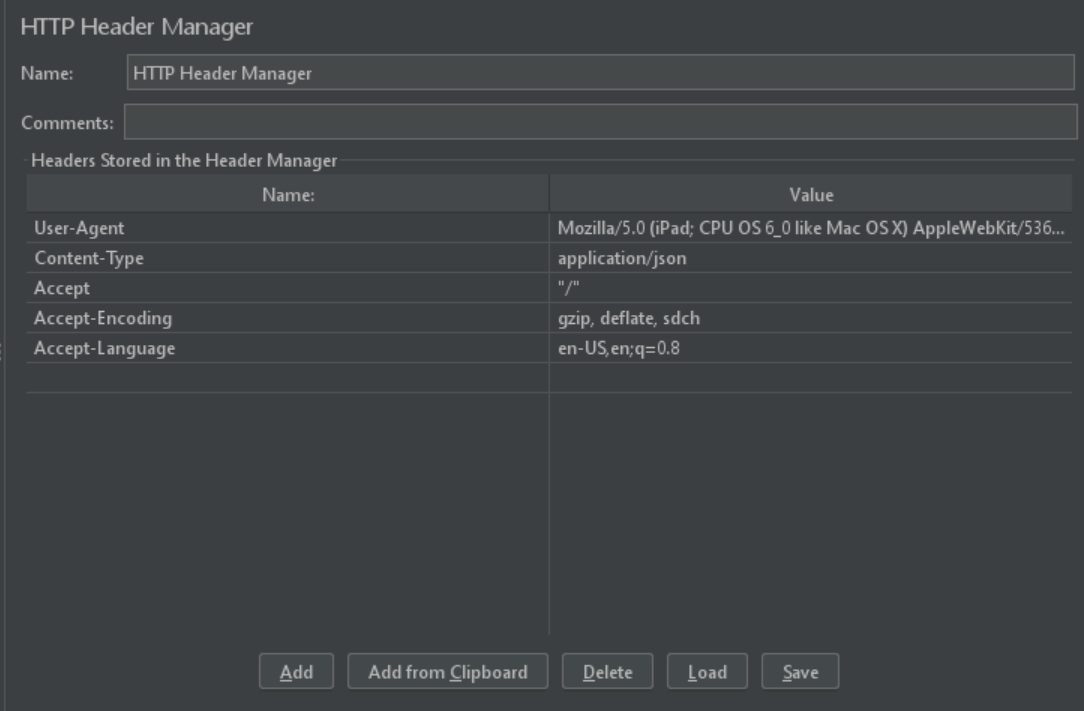
Lalu klik kanan pada Performance test web employee>add>config element>user defined variable



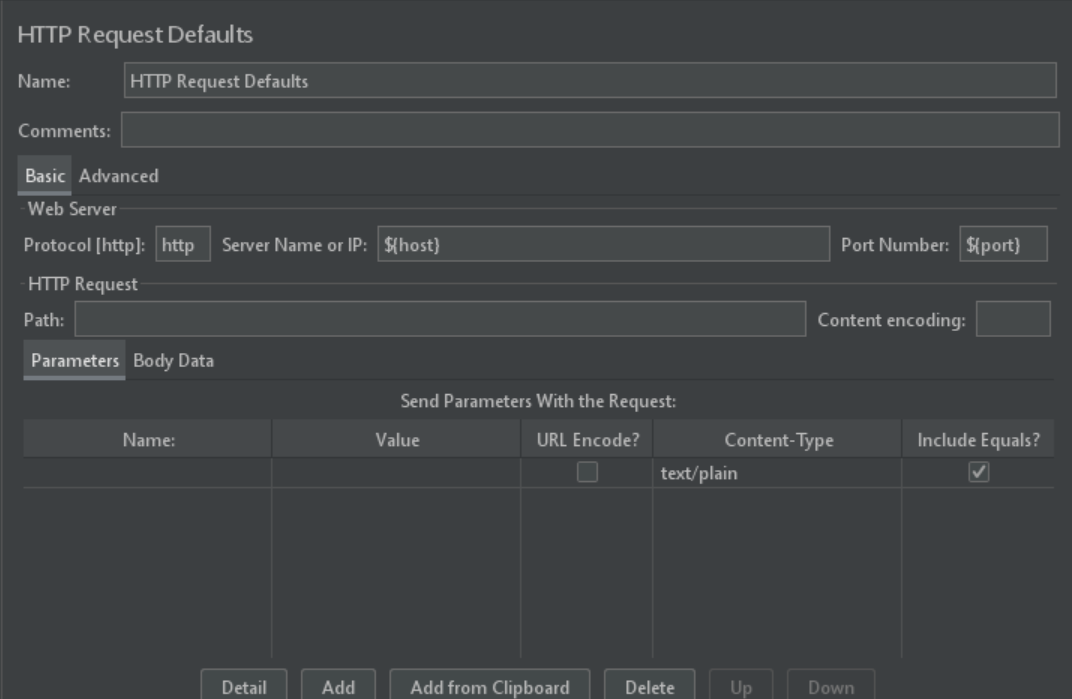
Ubah nama user defined variables menjadi general, dan tambah dua variable host dan port



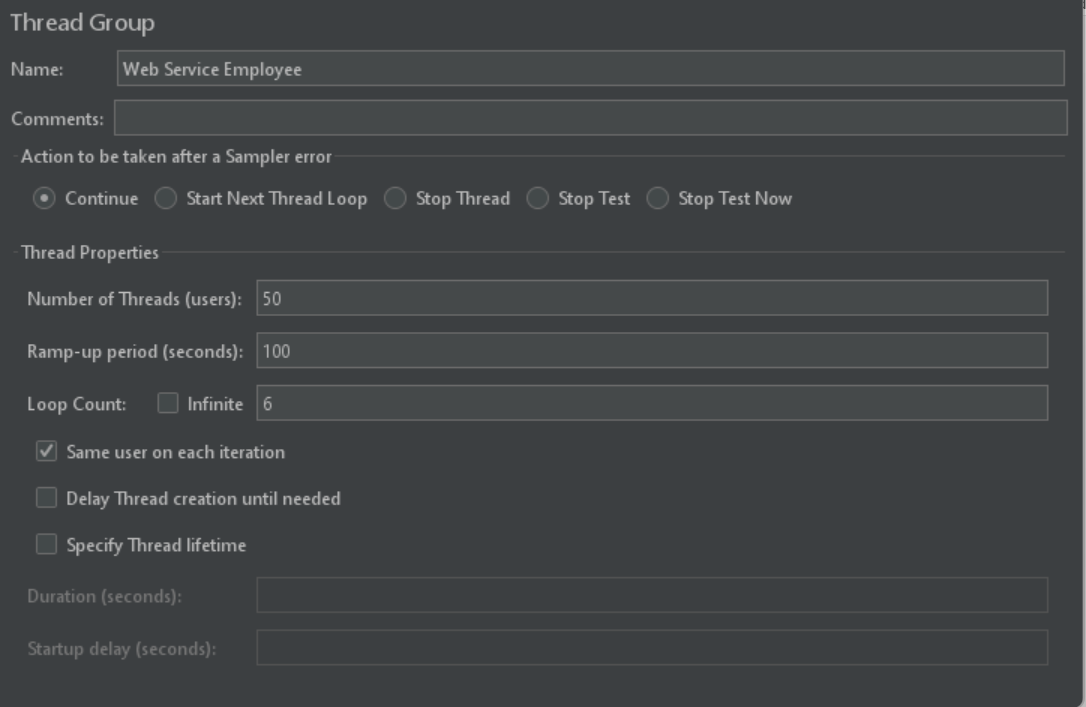
Tambahkan header manager dengan cara klik kanan di performance test web employee>add>config element>http header manager, dan tambahkan variable yang dibutuhkan.



Lalu kita tambahkan http request defaults dengan cara klik kanan di performance test web employee>add>config element>http request default, dan pamggil variable host dan port.

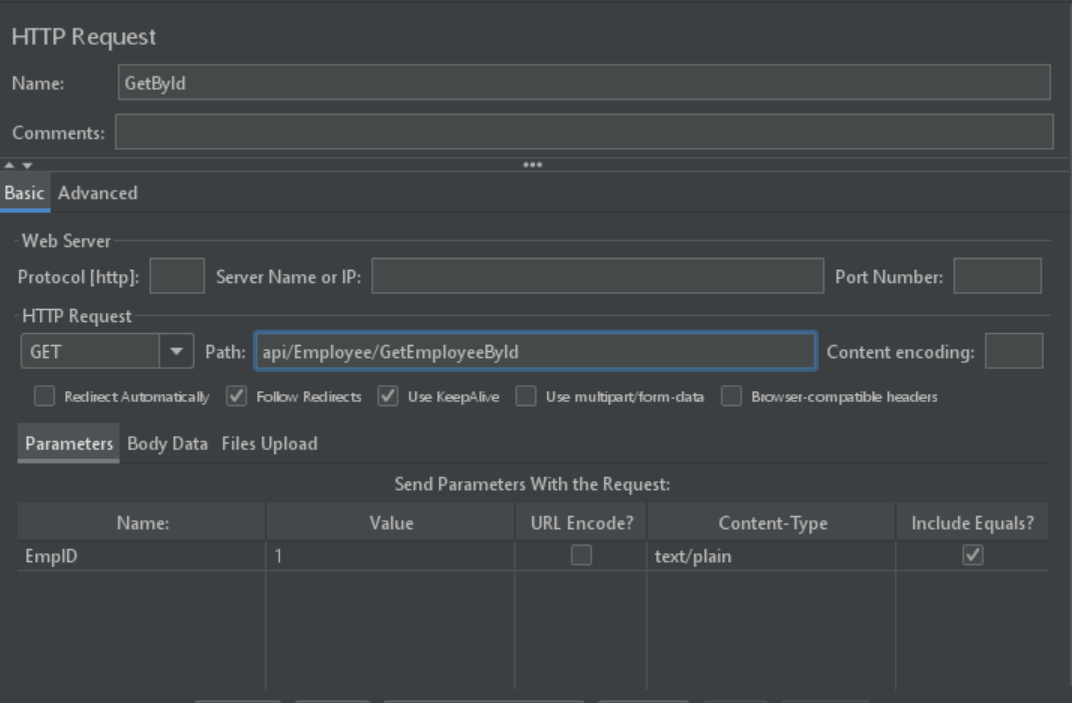


Selanjutnya Klik kanan node Test Plan (Performance Test Web Employee) -> Add -> Threads (Users) -> Thread Group.

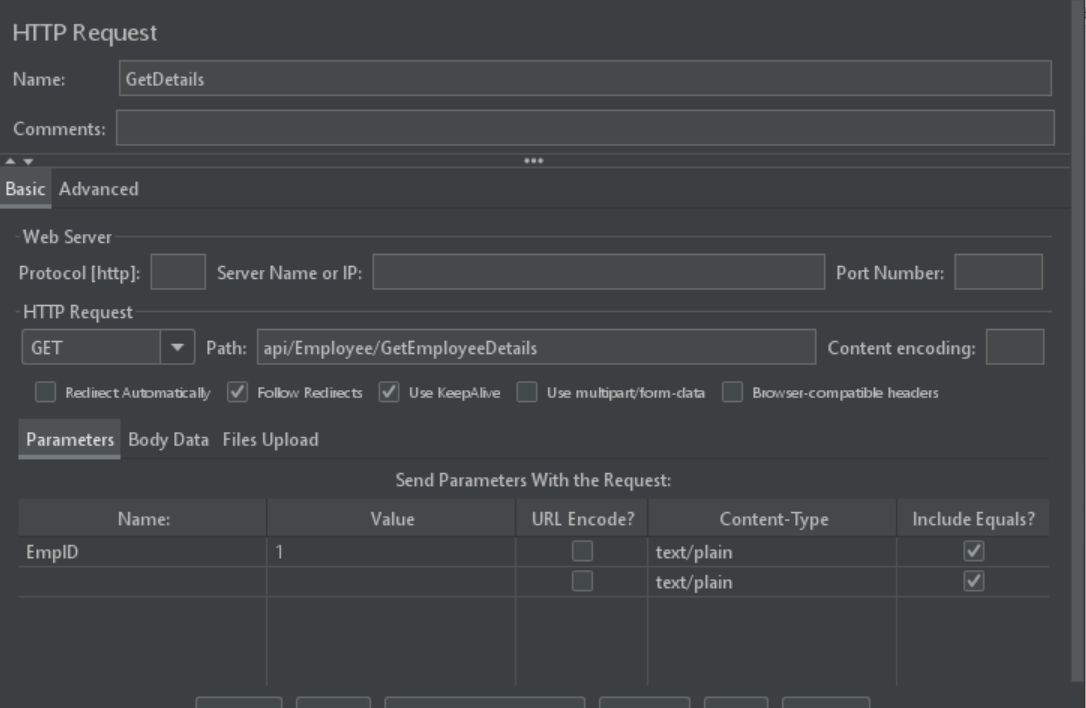


Setelah menentukan skenario performance test, langkah berikutnya adalah menambahkan node HTTP Request. Di node inilah kita akan menentukan web service yang akan di tes. Misal web service Employee mempunyai dua layanan yaitu GetByID dan GetDetails.

Klik kanan node Thread Group (Web Service Mahasiswa) -> Add -> Sampler -> HTTP Request.



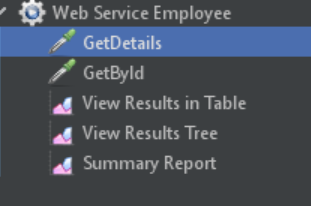
Lalu tambahkan service GetDetails



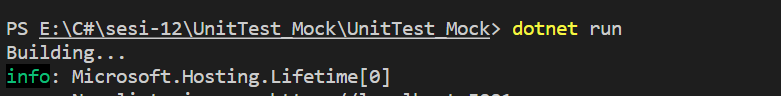
Lalu install plugin yang dibutuhkan

* 3 Basic Graphs
* 5 Additional Graphs
* Auto-Stop Listener
* BM.Sense Uploader
* BlazeMeter Step-by-step Debugger
* Cassandra Support
* Command-Line Graph Plotting Tool
* Composite Timeline Graph
* Console Status Logger
* Custom JMeter Functions
* Custom SOAP Sampler
* Custom Thread Groups
* DBMon Sample Collector
* Directory Listing Config

Lalu tambahkan semua format laporan yang dibutuhkan klik kanan node Thread Group (Web Service Employee) -> Add -> Listener -> Pilih jenis laporan.

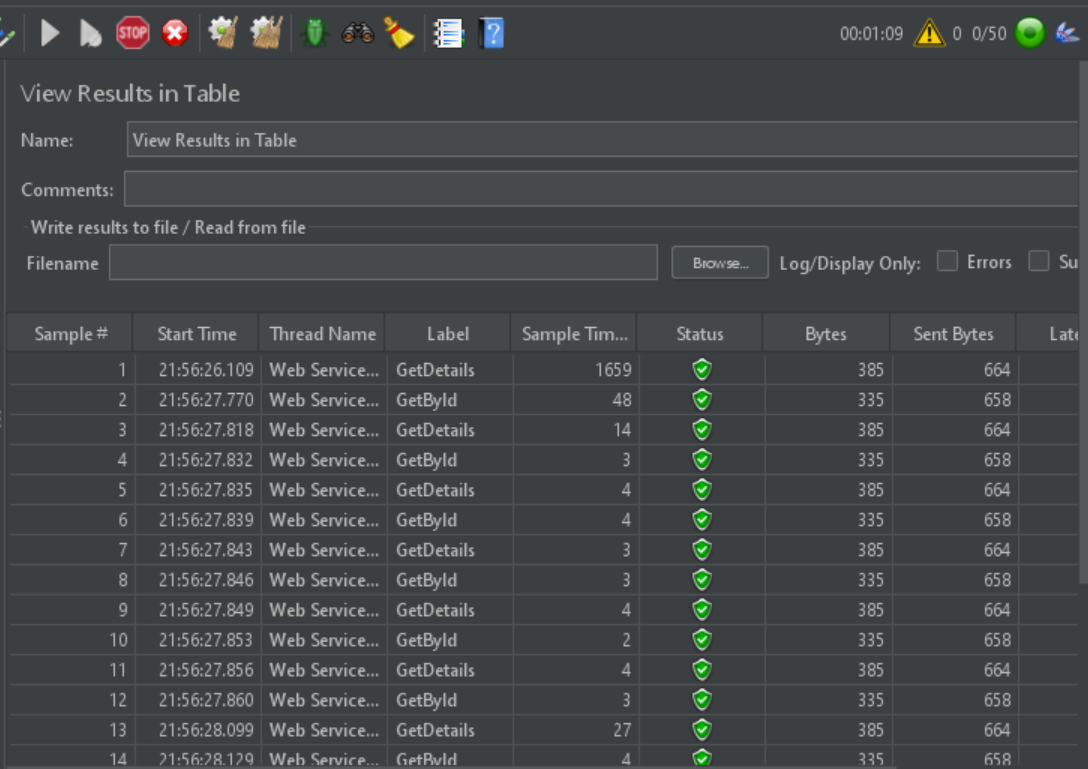


Lalu jalankan project kita

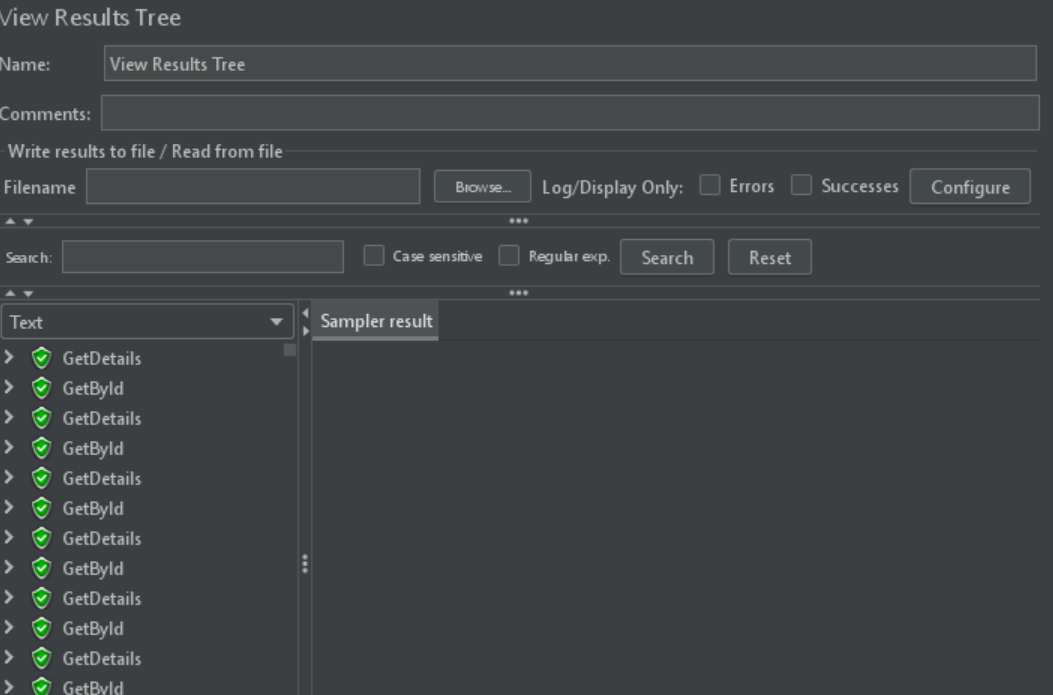


Kemudian jalankan Jmeter, Lalu lihat hasil

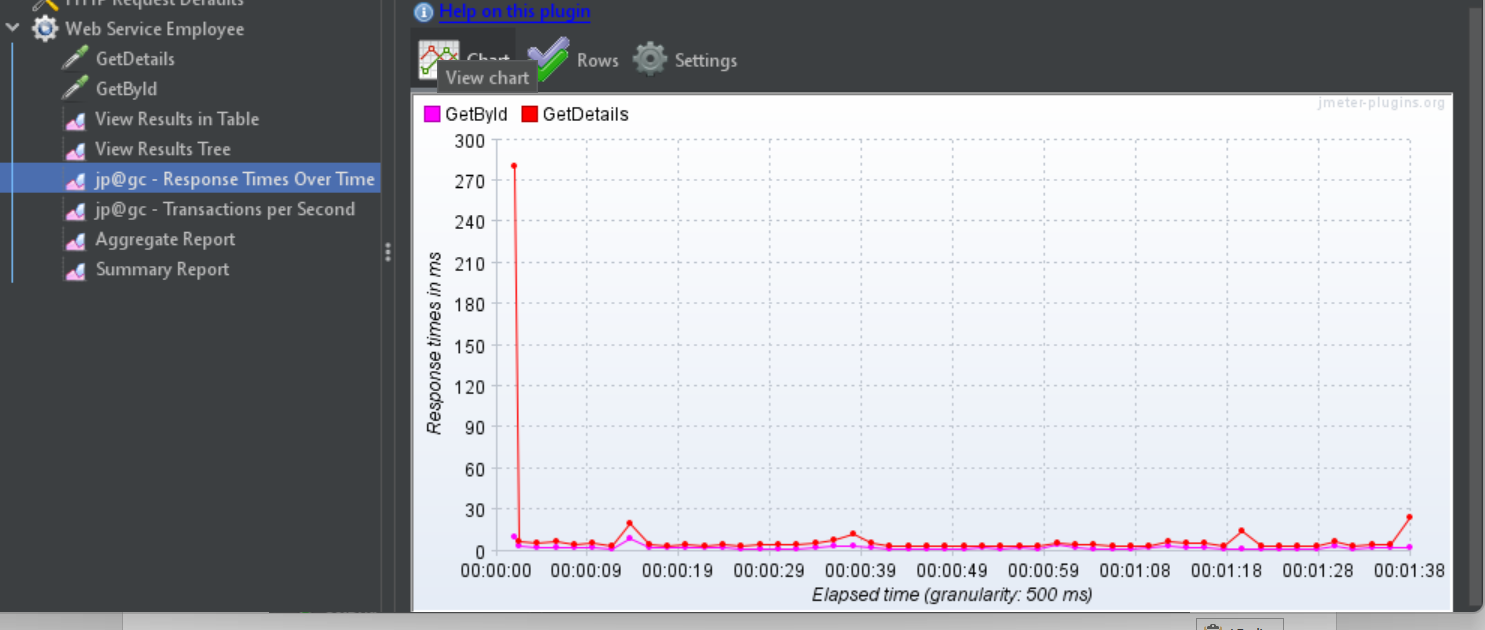
Tampilan Hasil view results in table



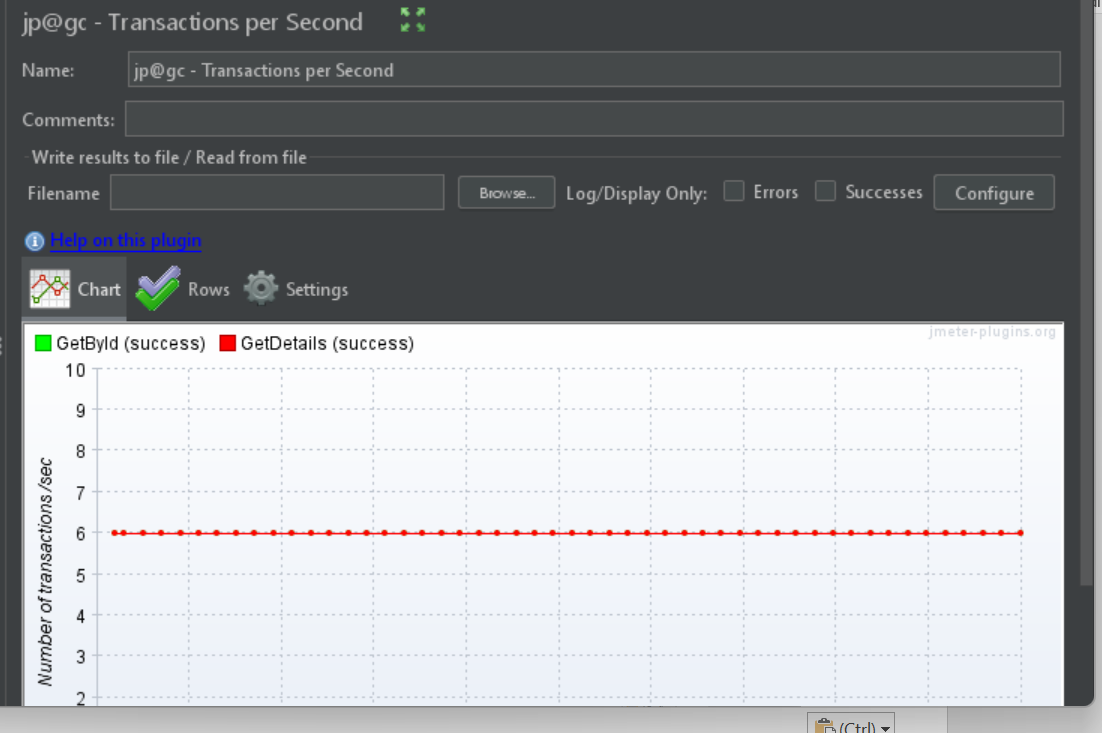
Tampilan hasil view results tree



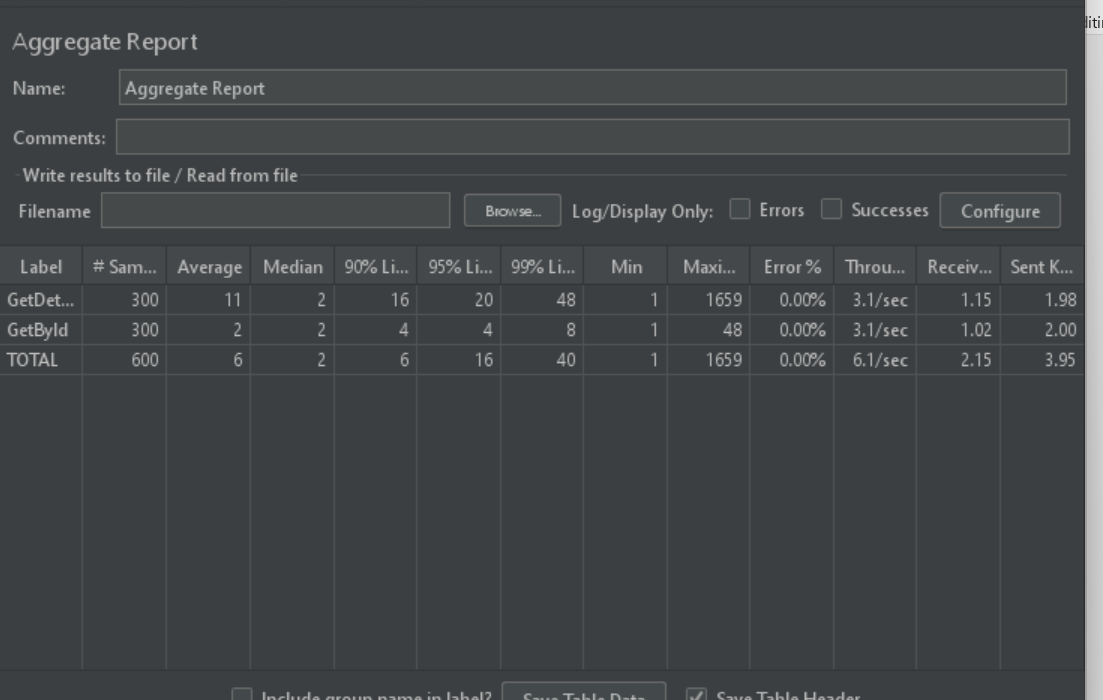
Tampilan hasil Response times over time



Tampilan hasil transactions per second



Tampilan hasil aggregate report



Tampilan hasil summary report

