<https://www.c-sharpcorner.com/article/api-versioning-in-asp-net-core-web-api/>

Install Package > Microsoft.AspNetCore.Mvc.Versioning

In Program.cs file

builder.Services.AddApiVersioning();

builder.Services.AddApiVersioning(config =>

{

config.DefaultApiVersion = new ApiVersion(1, 0);

config.AssumeDefaultVersionWhenUnspecified = true;

config.ReportApiVersions = true;

});

[HttpGet]

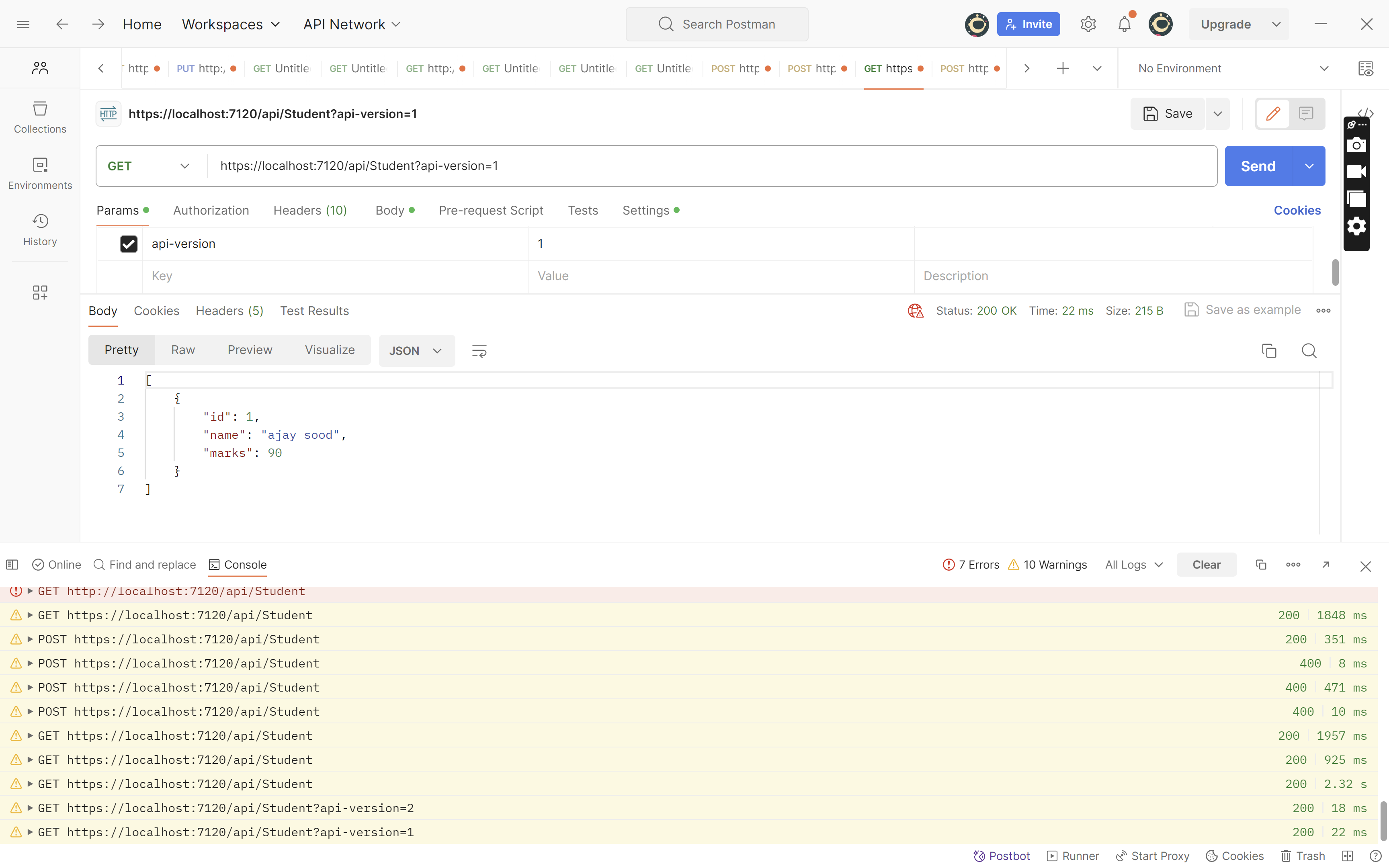
[ApiVersion("1.0")]

public List<StudentModel> Get()

{

return \_bal.GetStudents();

}



//builder.Services.AddApiVersioning();

builder.Services.AddApiVersioning(config =>

{

config.DefaultApiVersion = new ApiVersion(1, 0);

config.AssumeDefaultVersionWhenUnspecified = true;

config.ReportApiVersions = true;

});

[Route("api/[controller]")]

[ApiController]

[ApiVersion("1.0")]

[ApiVersion("2.0")]

//[Route("api/{v:apiVersion}/student")]

public class StudentController : ControllerBase

{

private Business\_Access\_Layer.StudentBal \_bal;

public StudentController() {

\_bal = new Business\_Access\_Layer.StudentBal();

}

[HttpGet,MapToApiVersion("1.0")]

//[Route("getstudents")]

public List<StudentModel> GetStudents()

{

return \_bal.GetStudents();

}

[HttpGet, MapToApiVersion("2.0")]

//[Route("getnames")]

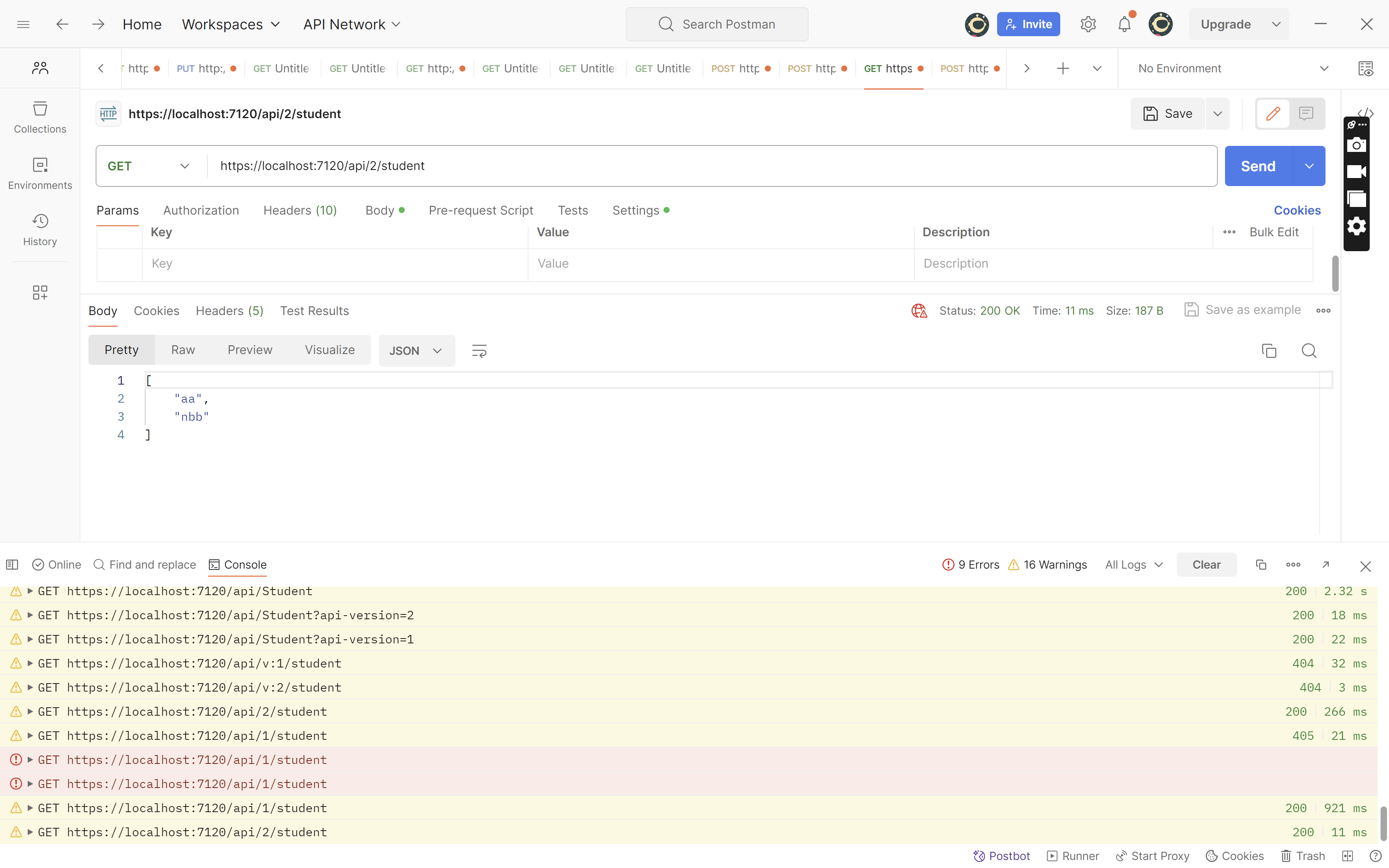
public List<string> GetNames()

{

return new List<string>() { "aa", "bb", "cc", "dd" };

}

Url Versioning



[ApiController]

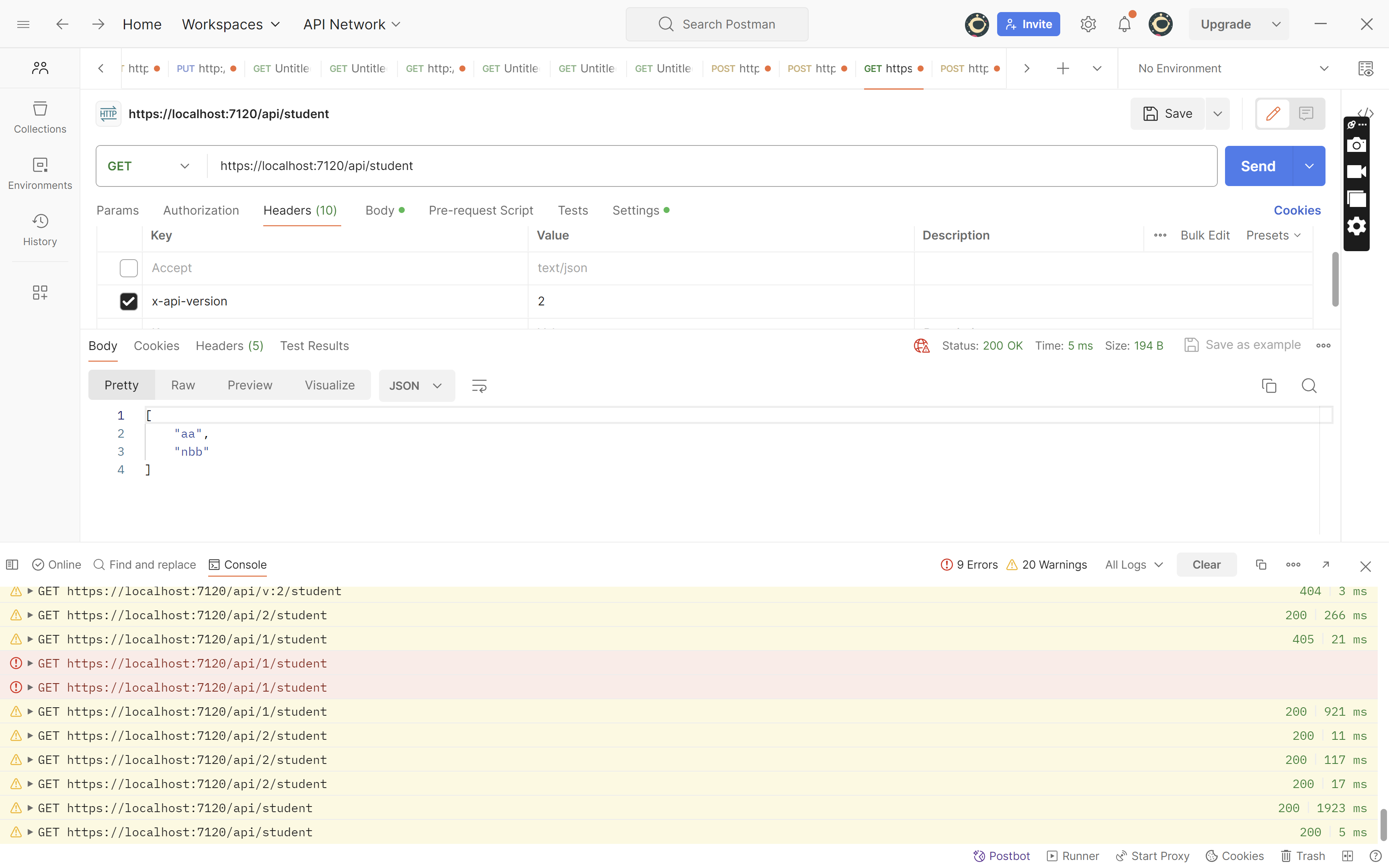
[ApiVersion("1.0")]

[ApiVersion("2.0")]

[Route("api/v{version:apiVersion}/student")]

public class StudentController : ControllerBase

using Header



builder.Services.AddApiVersioning(config =>

{

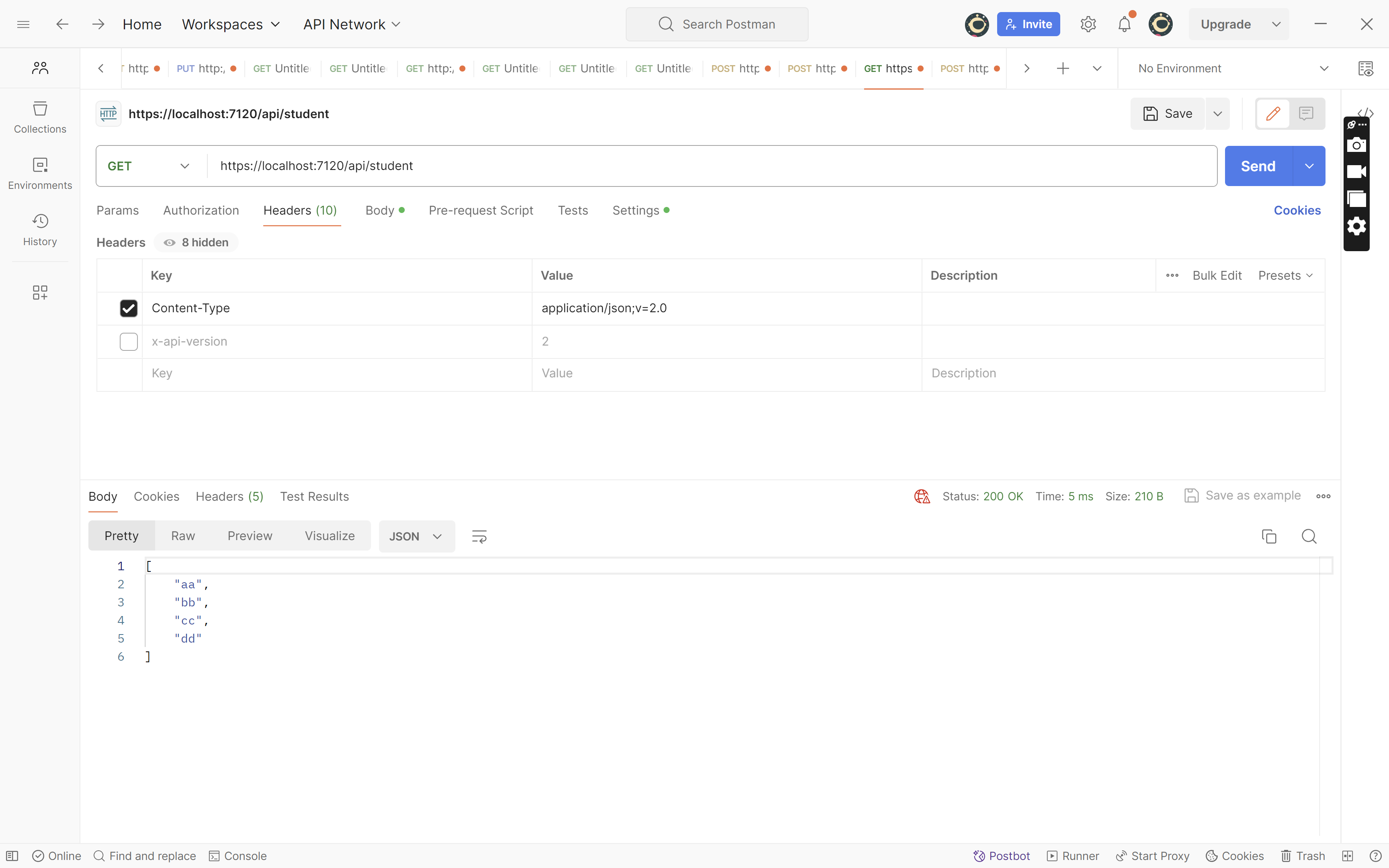
config.DefaultApiVersion = new ApiVersion(1, 0);

config.AssumeDefaultVersionWhenUnspecified = true;

config.ReportApiVersions = true;

config.ApiVersionReader = new HeaderApiVersionReader("x-api-version");

});



builder.Services.AddApiVersioning(config =>

{

config.DefaultApiVersion = new ApiVersion(1, 0);

config.AssumeDefaultVersionWhenUnspecified = true;

config.ReportApiVersions = true;

config.ApiVersionReader = new MediaTypeApiVersionReader();

//config.ApiVersionReader = new HeaderApiVersionReader("x-api-version");

});

builder.Services.AddApiVersioning(config =>

{

//config.DefaultApiVersion = new ApiVersion(1, 0);

//config.AssumeDefaultVersionWhenUnspecified = true;

config.ReportApiVersions = true;

config.ApiVersionReader = new MediaTypeApiVersionReader();

config.AssumeDefaultVersionWhenUnspecified = true;

config.ApiVersionSelector = new CurrentImplementationApiVersionSelector(config);

//config.ApiVersionReader = new HeaderApiVersionReader("x-api-version");

});

 Deprecating API Version

We can deprecate the API version which is no longer used. We do not need to remove the version, as it can be used somewhere. We can mark it as deprecated, so when an API is called, the header information displays that the called API is deprecated.

[HttpGet]

[Route("[controller]", Order = 1)]

[Route("api/v{version:apiVersion}/[controller]", Order = 2)]

[ApiVersion("1.0", Deprecated = **true**)]

public IEnumerable Get()

{

**var** person = \_personService.GetList();

**return** person;

}