We got 2 projects: Service_Discovery_Server_Api and Service_Discovery_Client

The main one is Service_Discovery_Client, it uses Service_Discovery_Server_Api as a referent services to get data. To do so, in this example, I use Microsoft.Extensions.ServiceDiscovery library and IHttpClientFactory

S1



S2



S3



S4

**HomeController.cs**

```csharp
namespace Service_Discovery_Client.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        private readonly IHttpClientFactory _httpClientFactory;

        public HomeController(ILogger<HomeController> logger, IHttpClientFactory httpClientFactory)
        {
            _logger = logger;
            _httpClientFactory = httpClientFactory;
        }

        public async Task<IActionResult> Test()
        {

            var httpClient = _httpClientFactory.CreateClient("Service_Discovery_Server_Api");

            using HttpResponseMessage response = await httpClient.GetAsync("http://Service_Discovery_Server_Api/api/customer");

            var jsonResponse = await response.Content.ReadAsStringAsync();

            ViewBag.Hung = jsonResponse.ToString();

            return View();
        }
    }
```

**launchSettings.json**

Schema: http://json.schemastore.org/launchsettings.json

```json
        "https": {
          "commandName": "Project",
          "dotnetRunMessages": true,
          "launchBrowser": true,
          "launchUrl": "swagger",
          "applicationUrl": "https://localhost:7156",
          "environmentVariables": {
            "ASPNETCORE_ENVIRONMENT": "Development"
          }
        },
```

**Customer.cs**

```csharp
public static class CustomerEndpoints
{
    public static void MapCustomerEndpoints (this IEndpointRouteBuilder routes)
    {
        var group = routes.MapGroup("/api/Customer").WithTags(nameof(Customer));

        group.MapGet("/", async (HungDbContext db) =>
        {
            return await db.Customers.ToListAsync();
        })
        .WithName("GetAllCustomers")
        .WithOpenApi();

        group.MapGet("/{id}", async Task<Results<Ok<Customer>, NotFound>> (int customerid, HungDbContext db) =>
        {
            return await db.Customers.AsNoTracking()
                .FirstOrDefaultAsync(model => model.CustomerId == customerid)
                is Customer model
                    ? TypedResults.Ok(model)
                    : TypedResults.NotFound();
        })
        .WithName("GetCustomerById")
        .WithOpenApi();

        group.MapPut("/{id}", async Task<Results<Ok, NotFound>> (int customerid, Customer customer, HungDbContext db) =>
        {
            var affected = await db.Customers
                .Where(model => model.CustomerId == customerid)
                .ExecuteUpdateAsync(setters => setters
                    .SetProperty(m => m.CustomerId, customer.CustomerId)
                    .SetProperty(m => m.FirstName, customer.FirstName)
                    .SetProperty(m => m.LastName, customer.LastName)
                    .SetProperty(m => m.DateOfBirth, customer.DateOfBirth)
                    .SetProperty(m => m.Email, customer.Email)
                    .SetProperty(m => m.StreesAddress, customer.StreesAddress)
                    .SetProperty(m => m.City, customer.City)
```