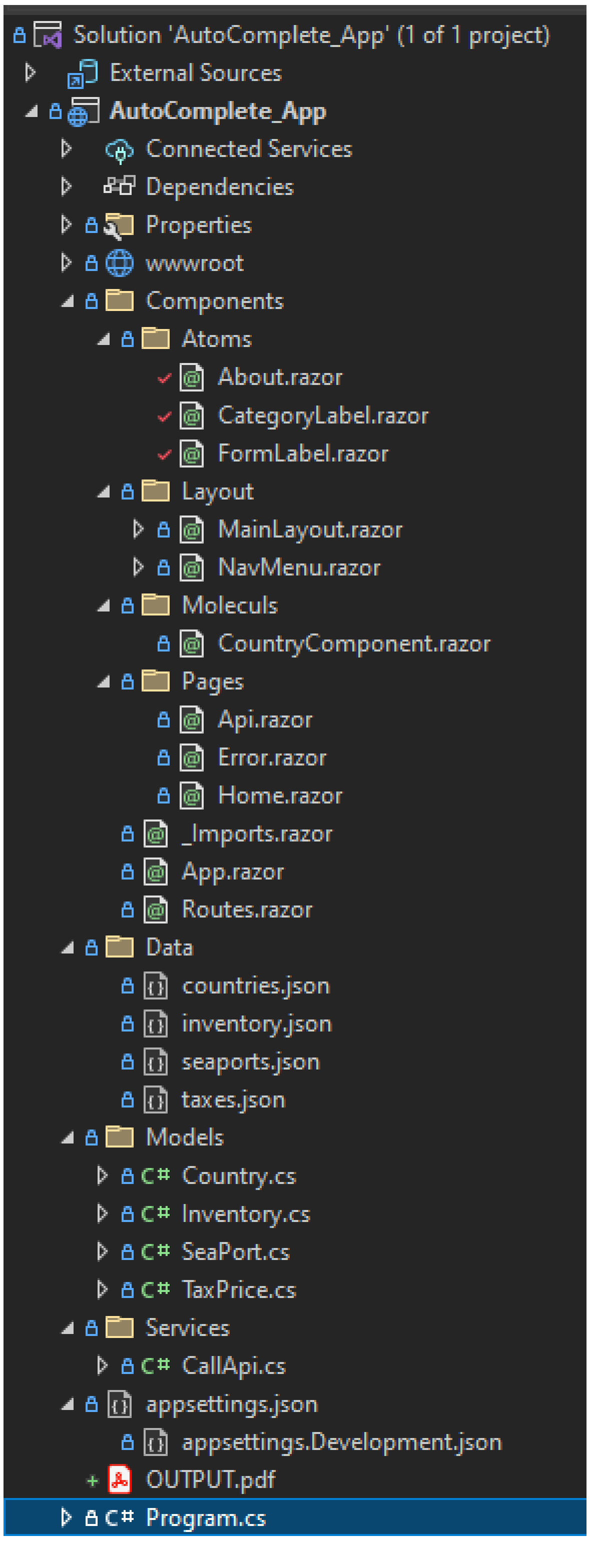
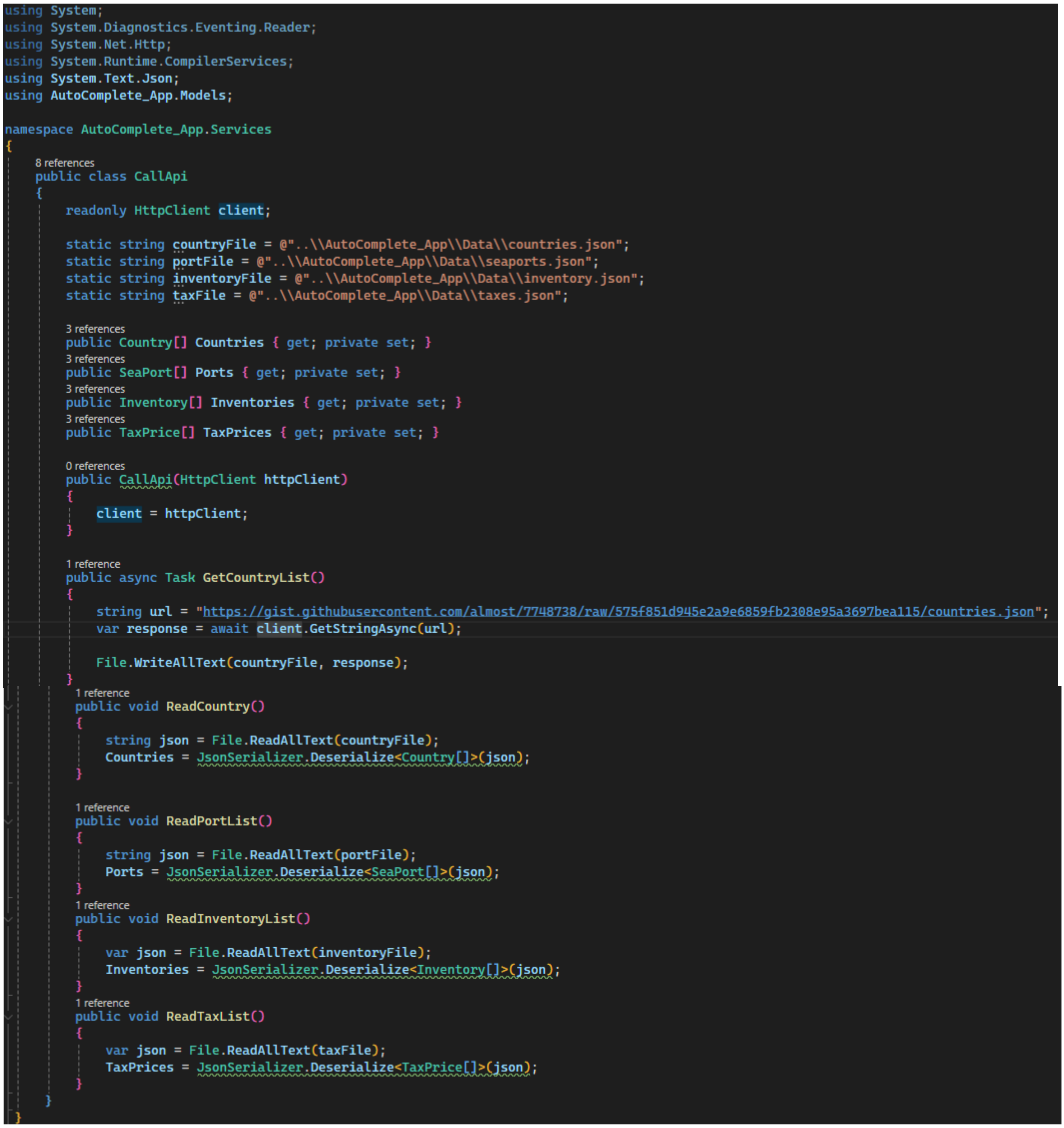
Code Overview

Folder Directories



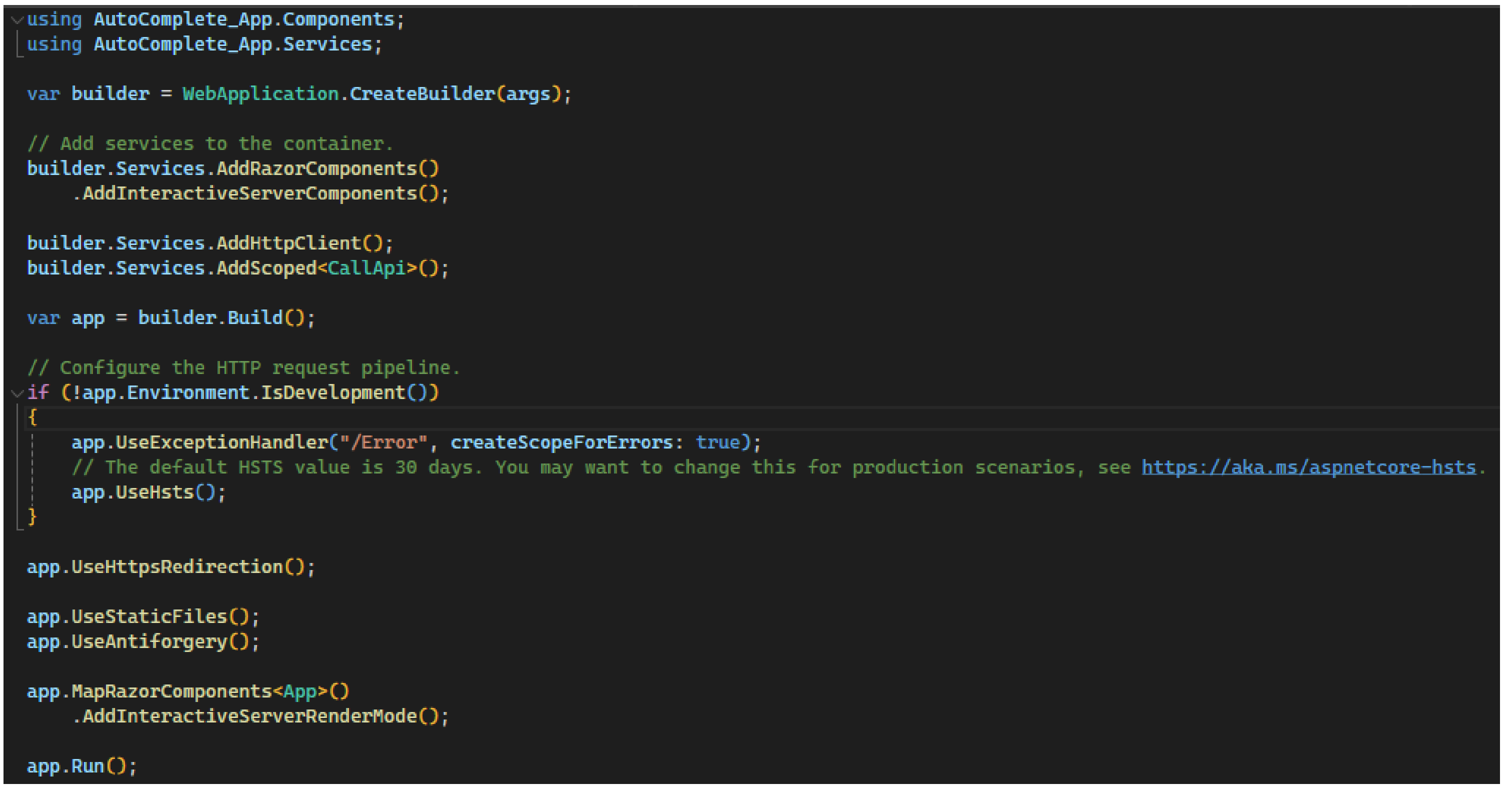
```
Solution 'AutoComplete_App' (1 of 1 project)
  External Sources
  AutoComplete_App
    Connected Services
    Dependencies
    Properties
    wwwroot
    Components
      Atoms
        About.razor
        CategoryLabel.razor
        FormLabel.razor
      Layout
        MainLayout.razor
        NavMenu.razor
      Moleculs
        CountryComponent.razor
      Pages
        Api.razor
        Error.razor
        Home.razor
      _Imports.razor
      App.razor
      Routes.razor
    Data
      countries.json
      inventory.json
      seaports.json
      taxes.json
    Models
      Country.cs
      Inventory.cs
      SeaPort.cs
      TaxPrice.cs
    Services
      CallApi.cs
    appsettings.json
      appsettings.Development.json
    OUTPUT.pdf
    Program.cs
```

CallAPI Service Class



```
using System;
using System.Diagnostics.Eventing.Reader;
using System.Net.Http;
using System.Runtime.CompilerServices;
using System.Text.Json;
using AutoComplete_App.Models;

namespace AutoComplete_App.Services
{
    8 references
    public class CallApi
    {
        readonly HttpClient client;

        static string countryFile = @"..\\AutoComplete_App\\Data\\countries.json";
        static string portFile = @"..\\AutoComplete_App\\Data\\seaports.json";
        static string inventoryFile = @"..\\AutoComplete_App\\Data\\inventory.json";
        static string taxFile = @"..\\AutoComplete_App\\Data\\taxes.json";

        3 references
        public Country[] Countries { get; private set; }
        3 references
        public SeaPort[] Ports { get; private set; }
        3 references
        public Inventory[] Inventories { get; private set; }
        3 references
        public TaxPrice[] TaxPrices { get; private set; }

        0 references
        public CallApi(HttpClient httpClient)
        {
            client = httpClient;
        }

        1 reference
        public async Task GetCountryList()
        {
            string url = "https://gist.githubusercontent.com/almost/7748738/raw/575f851d945e2a9e6859fb2308e95a3697bea115/countries.json";
            var response = await client.GetStringAsync(url);

            File.WriteAllText(countryFile, response);
        }
        1 reference
        public void ReadCountry()
        {
            string json = File.ReadAllText(countryFile);
            Countries = JsonSerializer.Deserialize<Country[]>(json);
        }

        1 reference
        public void ReadPortList()
        {
            string json = File.ReadAllText(portFile);
            Ports = JsonSerializer.Deserialize<SeaPort[]>(json);
        }
        1 reference
        public void ReadInventoryList()
        {
            var json = File.ReadAllText(inventoryFile);
            Inventories = JsonSerializer.Deserialize<Inventory[]>(json);
        }
        1 reference
        public void ReadTaxList()
        {
            var json = File.ReadAllText(taxFile);
            TaxPrices = JsonSerializer.Deserialize<TaxPrice[]>(json);
        }
    }
}
```

Program.cs



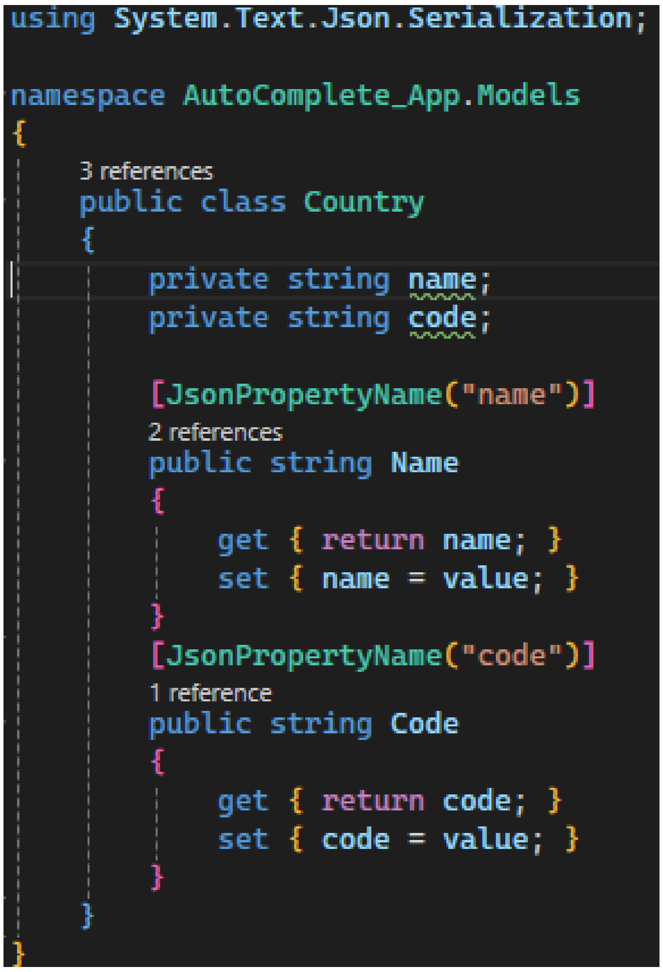
```csharp
using AutoComplete_App.Components;
using AutoComplete_App.Services;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddRazorComponents()
    .AddInteractiveServerComponents();

builder.Services.AddHttpClient();
builder.Services.AddScoped<CallApi>();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error", createScopeForErrors: true);
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();

app.UseStaticFiles();
app.UseAntiforgery();

app.MapRazorComponents<App>()
    .AddInteractiveServerRenderMode();

app.Run();
```

The program will initialize CallApi Service everytime the program is running.

Models

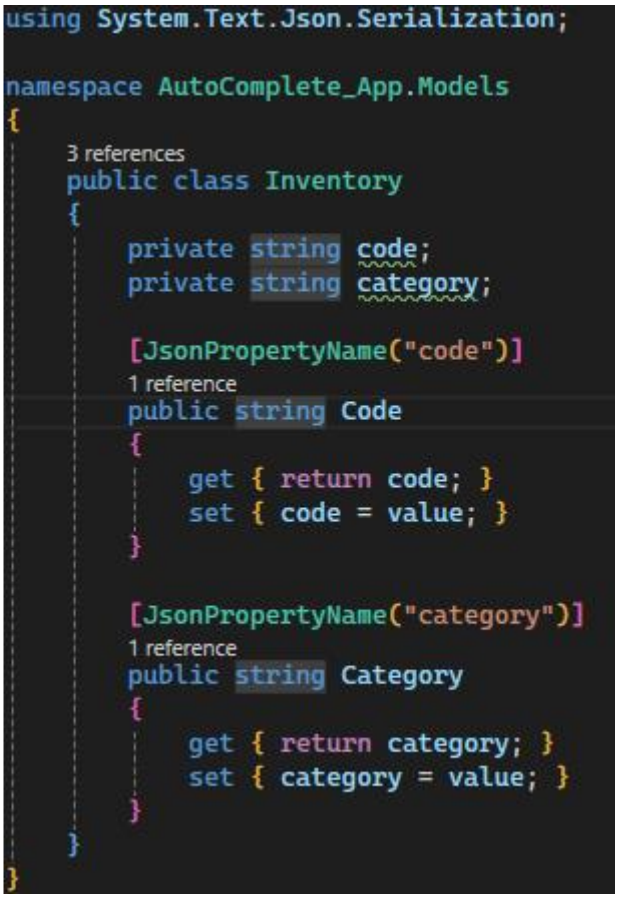
Each models represents the Object with json file attributes.

Country.cs



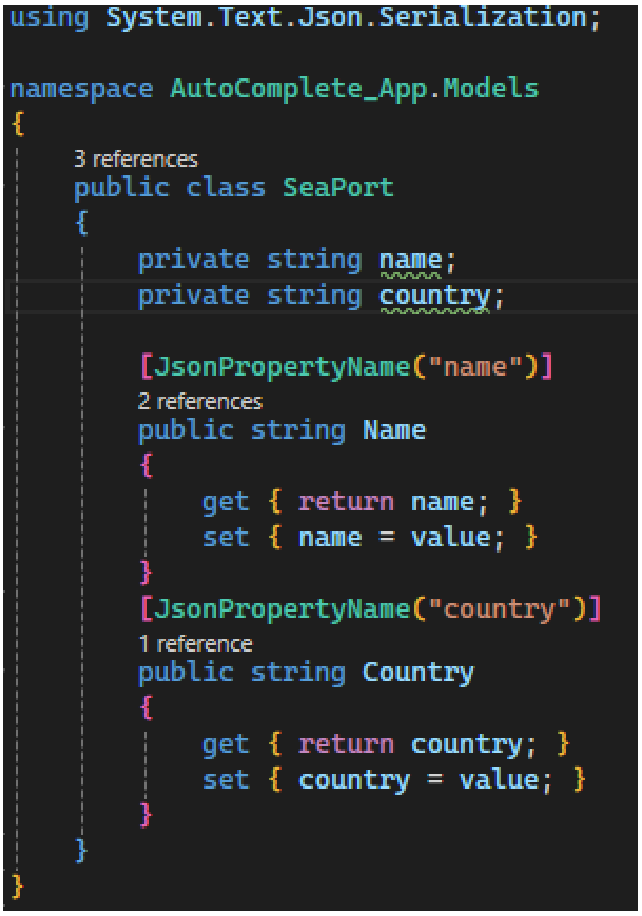
```
using System.Text.Json.Serialization;

namespace AutoComplete_App.Models
{
    3 references
    public class Country
    {
        private string name;
        private string code;

        [JsonPropertyName("name")]
        2 references
        public string Name
        {
            get { return name; }
            set { name = value; }
        }
        [JsonPropertyName("code")]
        1 reference
        public string Code
        {
            get { return code; }
            set { code = value; }
        }
    }
}
```

Inventory.cs



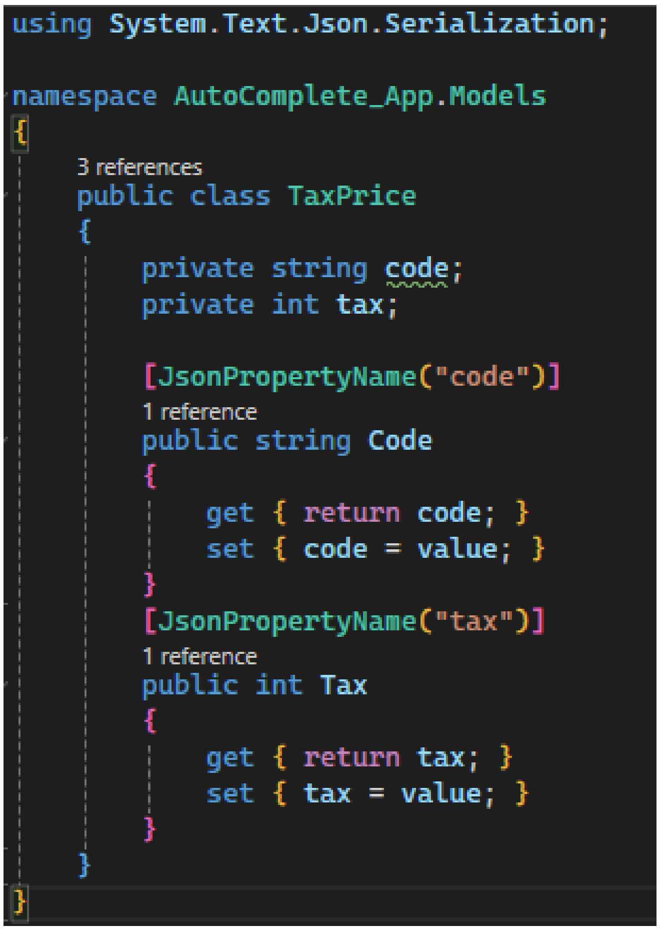
```
using System.Text.Json.Serialization;

namespace AutoComplete_App.Models
{
    3 references
    public class Inventory
    {
        private string code;
        private string category;

        [JsonPropertyName("code")]
        1 reference
        public string Code
        {
            get { return code; }
            set { code = value; }
        }

        [JsonPropertyName("category")]
        1 reference
        public string Category
        {
            get { return category; }
            set { category = value; }
        }
    }
}
```

Seaports.cs



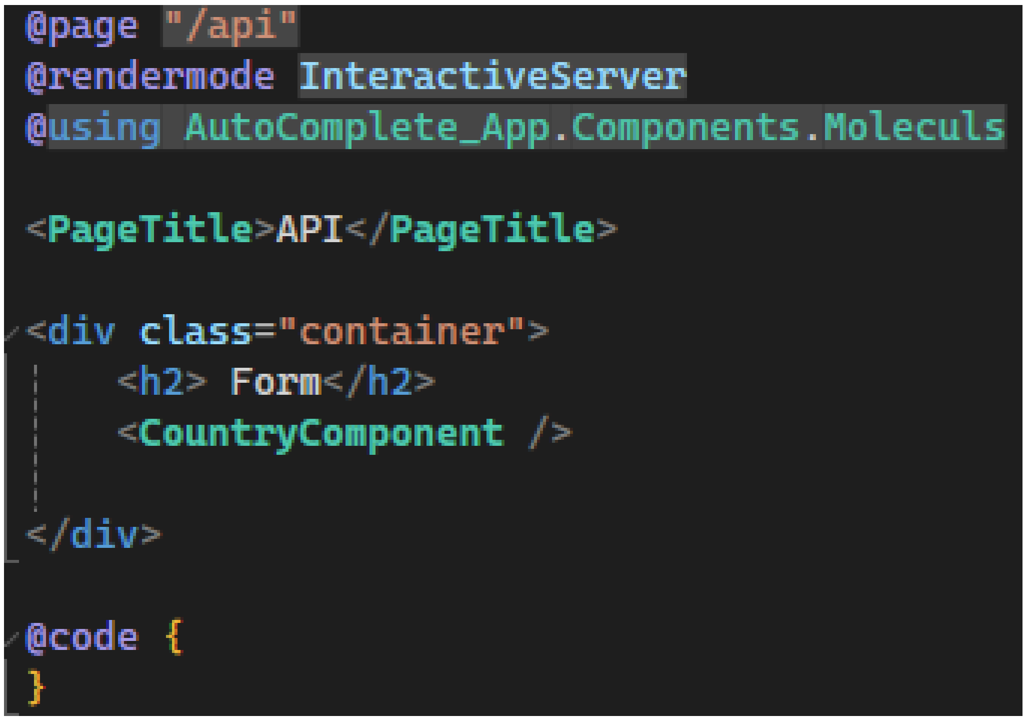
```
using System.Text.Json.Serialization;

namespace AutoComplete_App.Models
{
    3 references
    public class SeaPort
    {
        private string name;
        private string country;

        [JsonPropertyName("name")]
        2 references
        public string Name
        {
            get { return name; }
            set { name = value; }
        }
        [JsonPropertyName("country")]
        1 reference
        public string Country
        {
            get { return country; }
            set { country = value; }
        }
    }
}
```

Taxes.cs



```
using System.Text.Json.Serialization;

namespace AutoComplete_App.Models
{
    3 references
    public class TaxPrice
    {
        private string code;
        private int tax;

        [JsonPropertyName("code")]
        1 reference
        public string Code
        {
            get { return code; }
            set { code = value; }
        }
        [JsonPropertyName("tax")]
        1 reference
        public int Tax
        {
            get { return tax; }
            set { tax = value; }
        }
    }
}
```

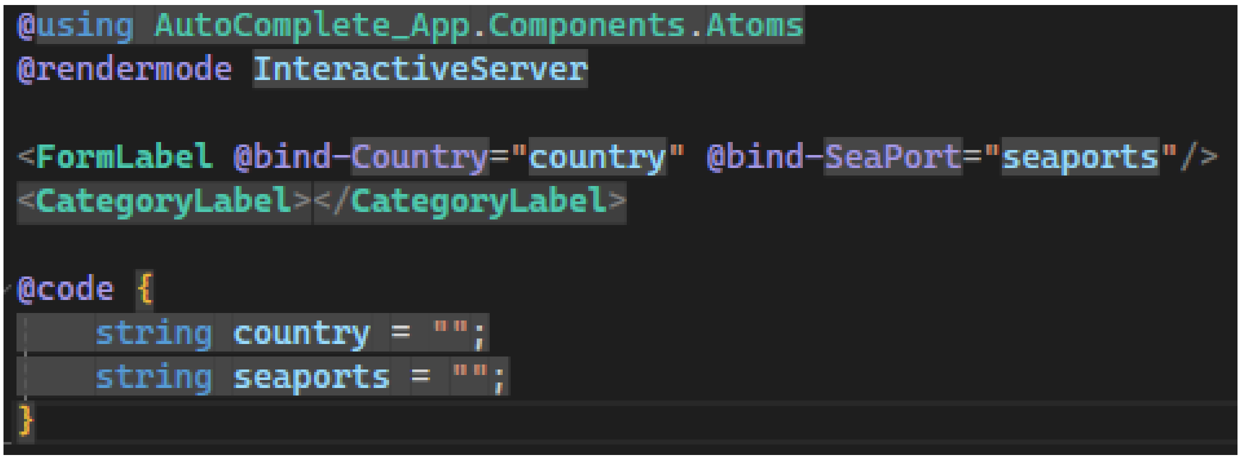
Api.razor



```
@page "/api"
@rendermode InteractiveServer
@using AutoComplete_App.Components.Moleculs

<PageTitle>API</PageTitle>

<div class="container">
    <h2> Form</h2>
    <CountryComponent />

</div>

@code {
}
```

Api.razor is the View of the component

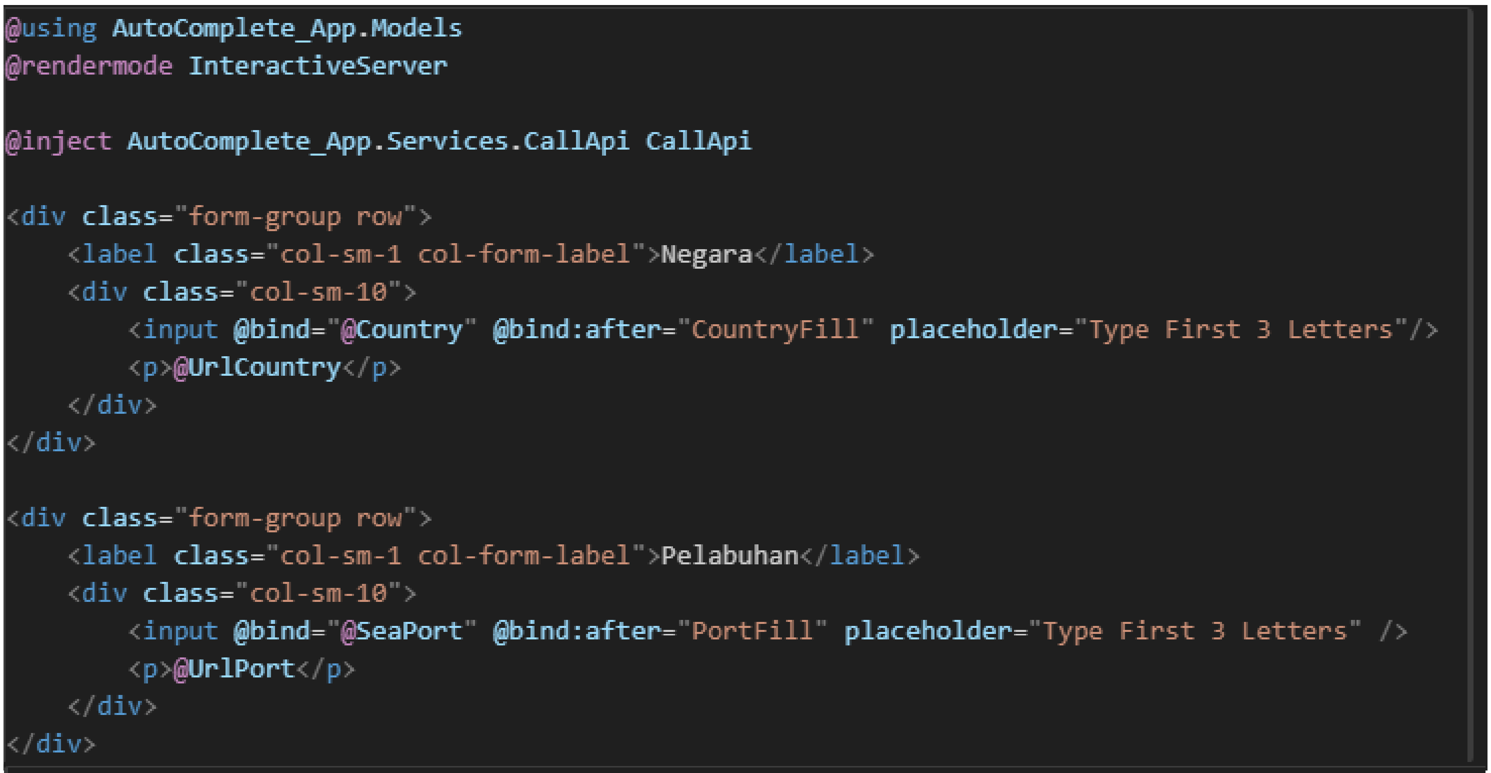
CountryComponent.razor



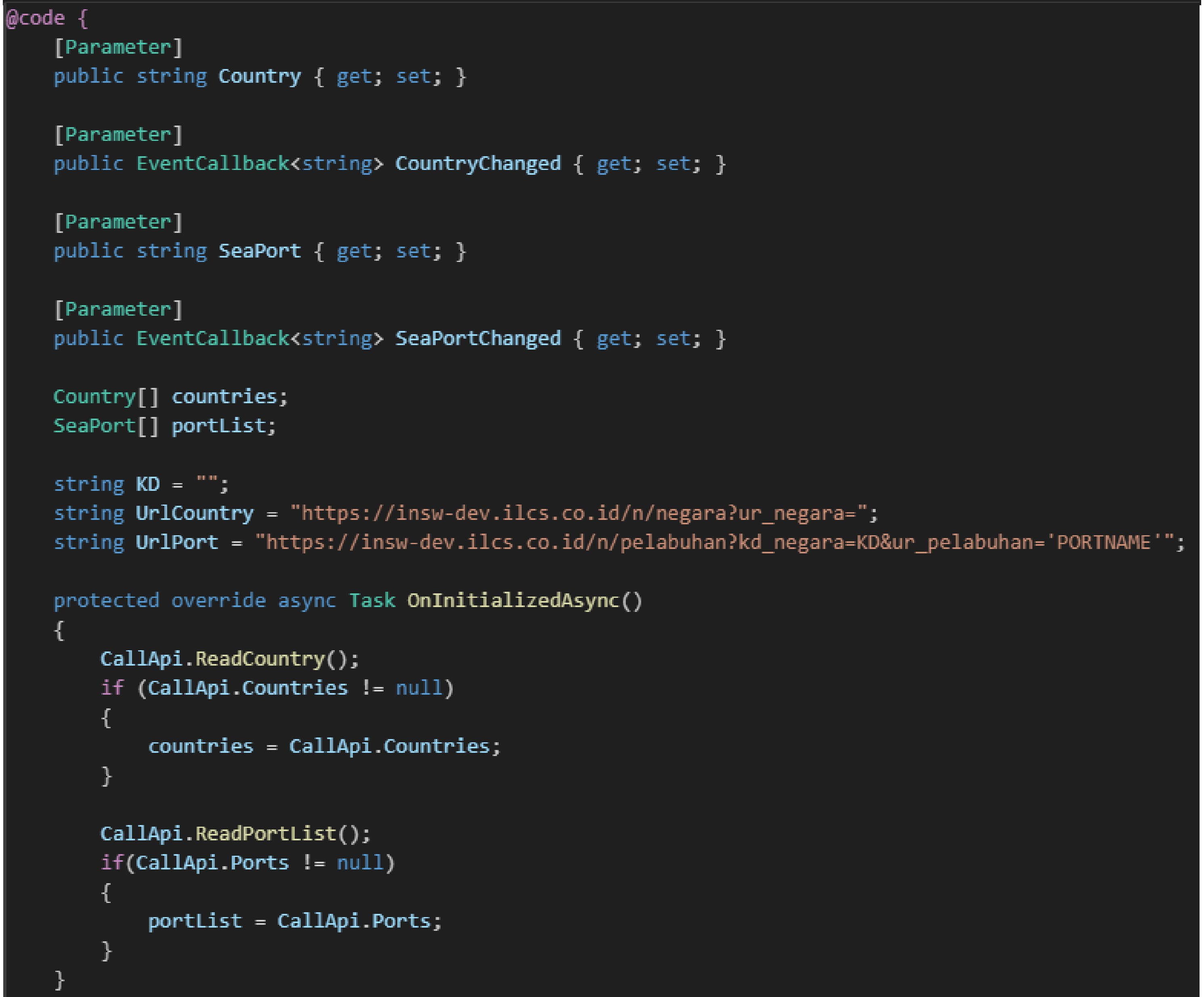
```
@using AutoComplete_App.Components.Atoms
@rendermode InteractiveServer

<FormLabel @bind-Country="country" @bind-SeaPort="seaports"/>
<CategoryLabel></CategoryLabel>

@code {
    string country = "";
    string seaports = "";
}
```

Sorry for bad naming file, I just don't know what a perfect name for the component. And also notice I use parameter passing for FormLabel but not for CategoryLabel components. I started to use passing, but I just think in this case, I don't need to use passing parameters since I don't reuse the components for this context.

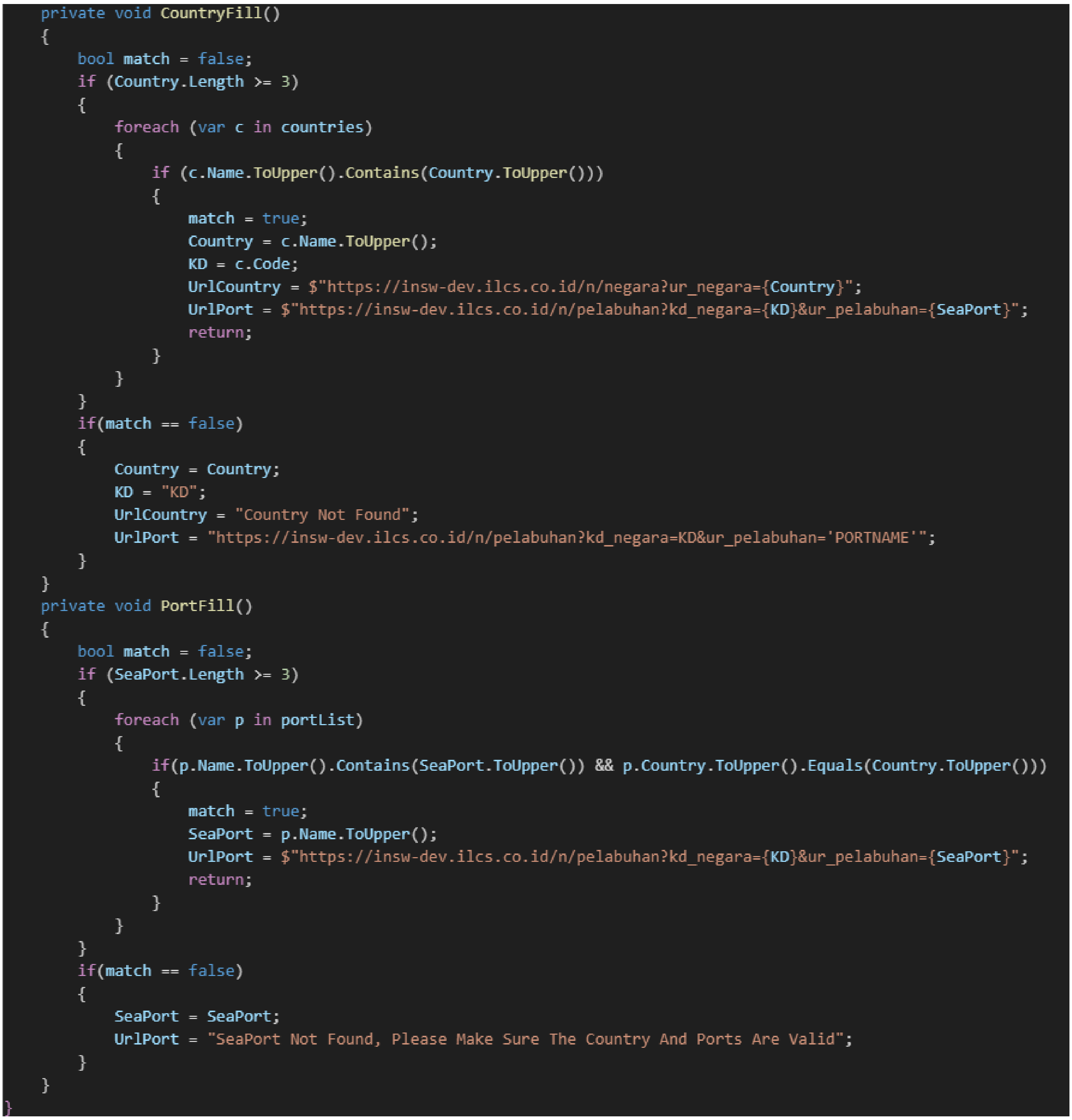
FormLabel.razor



```
@using AutoComplete_App.Models
@rendermode InteractiveServer

@inject AutoComplete_App.Services.CallApi CallApi

<div class="form-group row">
    <label class="col-sm-1 col-form-label">Negara</label>
    <div class="col-sm-10">
        <input @bind="@Country" @bind:after="CountryFill" placeholder="Type First 3 Letters"/>
        <p>@UrlCountry</p>
    </div>
</div>

<div class="form-group row">
    <label class="col-sm-1 col-form-label">Pelabuhan</label>
    <div class="col-sm-10">
        <input @bind="@SeaPort" @bind:after="PortFill" placeholder="Type First 3 Letters" />
        <p>@UrlPort</p>
    </div>
</div>
```



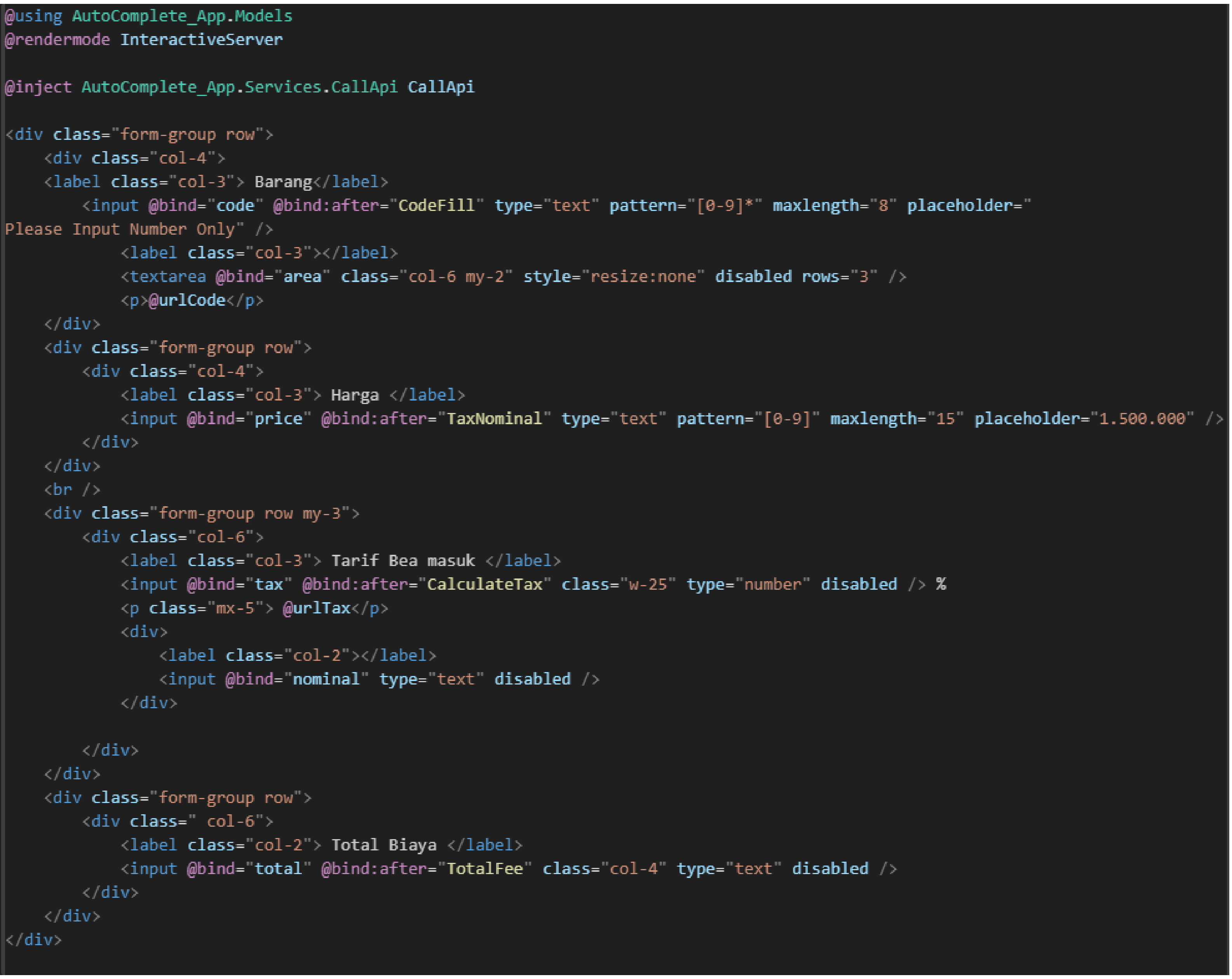
```
@code {
    [Parameter]
    public string Country { get; set; }

    [Parameter]
    public EventCallback<string> CountryChanged { get; set; }

    [Parameter]
    public string SeaPort { get; set; }

    [Parameter]
    public EventCallback<string> SeaPortChanged { get; set; }

    Country[] countries;
    SeaPort[] portList;

    string KD = "";
    string UrlCountry = "https://insw-dev.ilcs.co.id/n/negara?ur_negara=";
    string UrlPort = "https://insw-dev.ilcs.co.id/n/pelabuhan?kd_negara=KD&ur_pelabuhan='PORTNAME'";

    protected override async Task OnInitializedAsync()
    {
        CallApi.ReadCountry();
        if (CallApi.Countries != null)
        {
            countries = CallApi.Countries;
        }

        CallApi.ReadPortList();
        if(CallApi.Ports != null)
        {
            portList = CallApi.Ports;
        }
    }
```



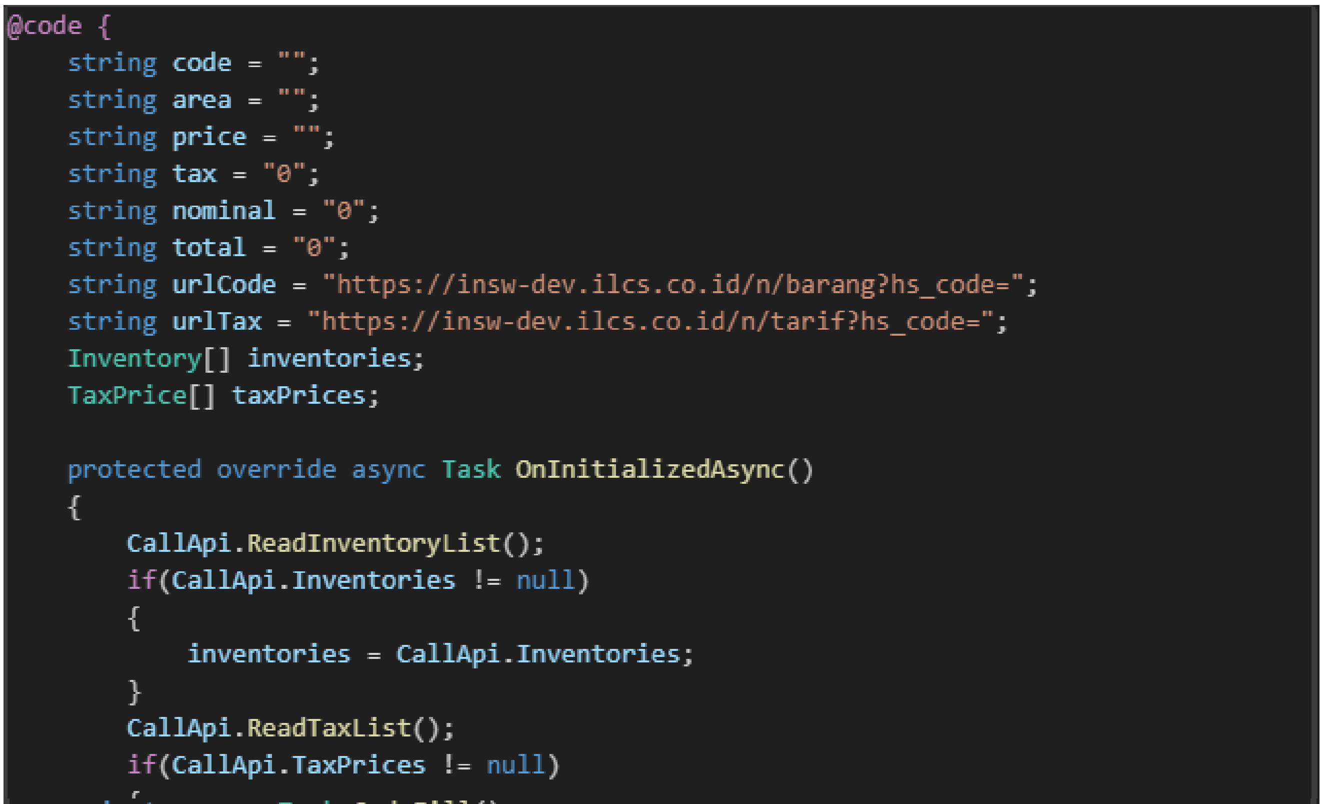
```
    private void CountryFill()
    {
        bool match = false;
        if (Country.Length >= 3)
        {
            foreach (var c in countries)
            {
                if (c.Name.ToUpper().Contains(Country.ToUpper()))
                {
                    match = true;
                    Country = c.Name.ToUpper();
                    KD = c.Code;
                    UrlCountry = $"https://insw-dev.ilcs.co.id/n/negara?ur_negara={Country}";
                    UrlPort = $"https://insw-dev.ilcs.co.id/n/pelabuhan?kd_negara={KD}&ur_pelabuhan={SeaPort}";
                    return;
                }
            }
        }
        if(match == false)
        {
            Country = Country;
            KD = "KD";
            UrlCountry = "Country Not Found";
            UrlPort = "https://insw-dev.ilcs.co.id/n/pelabuhan?kd_negara=KD&ur_pelabuhan='PORTNAME'";
        }
    }
    private void PortFill()
    {
        bool match = false;
        if (SeaPort.Length >= 3)
        {
            foreach (var p in portList)
            {
                if(p.Name.ToUpper().Contains(SeaPort.ToUpper()) && p.Country.ToUpper().Equals(Country.ToUpper()))
                {
                    match = true;
                    SeaPort = p.Name.ToUpper();
                    UrlPort = $"https://insw-dev.ilcs.co.id/n/pelabuhan?kd_negara={KD}&ur_pelabuhan={SeaPort}";
                    return;
                }
            }
        }
        if(match == false)
        {
            SeaPort = SeaPort;
            UrlPort = "SeaPort Not Found, Please Make Sure The Country And Ports Are Valid";
        }
    }
}
```

The code became longer than it has to, and as I said previously, in this context I don't need to reuse this component.

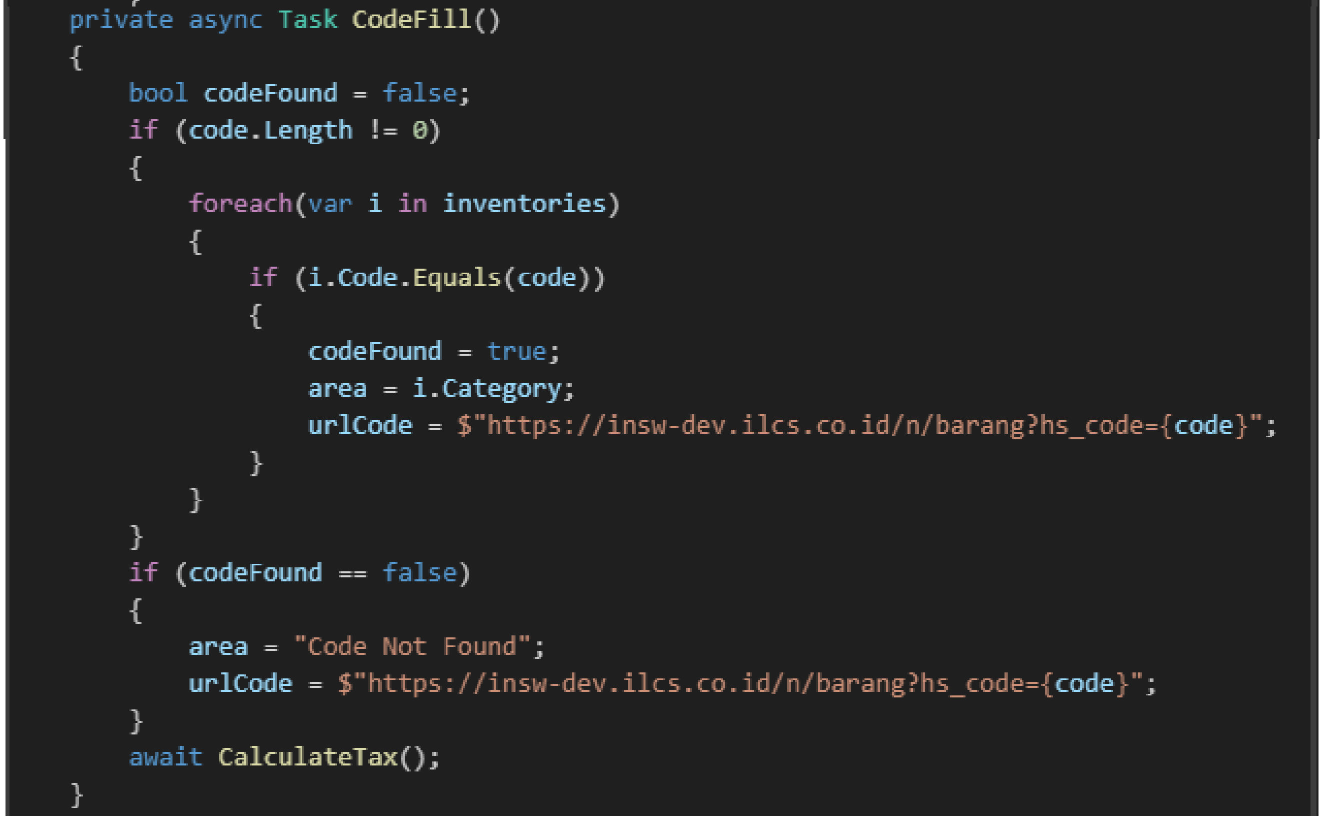
CategoryLabel.razor



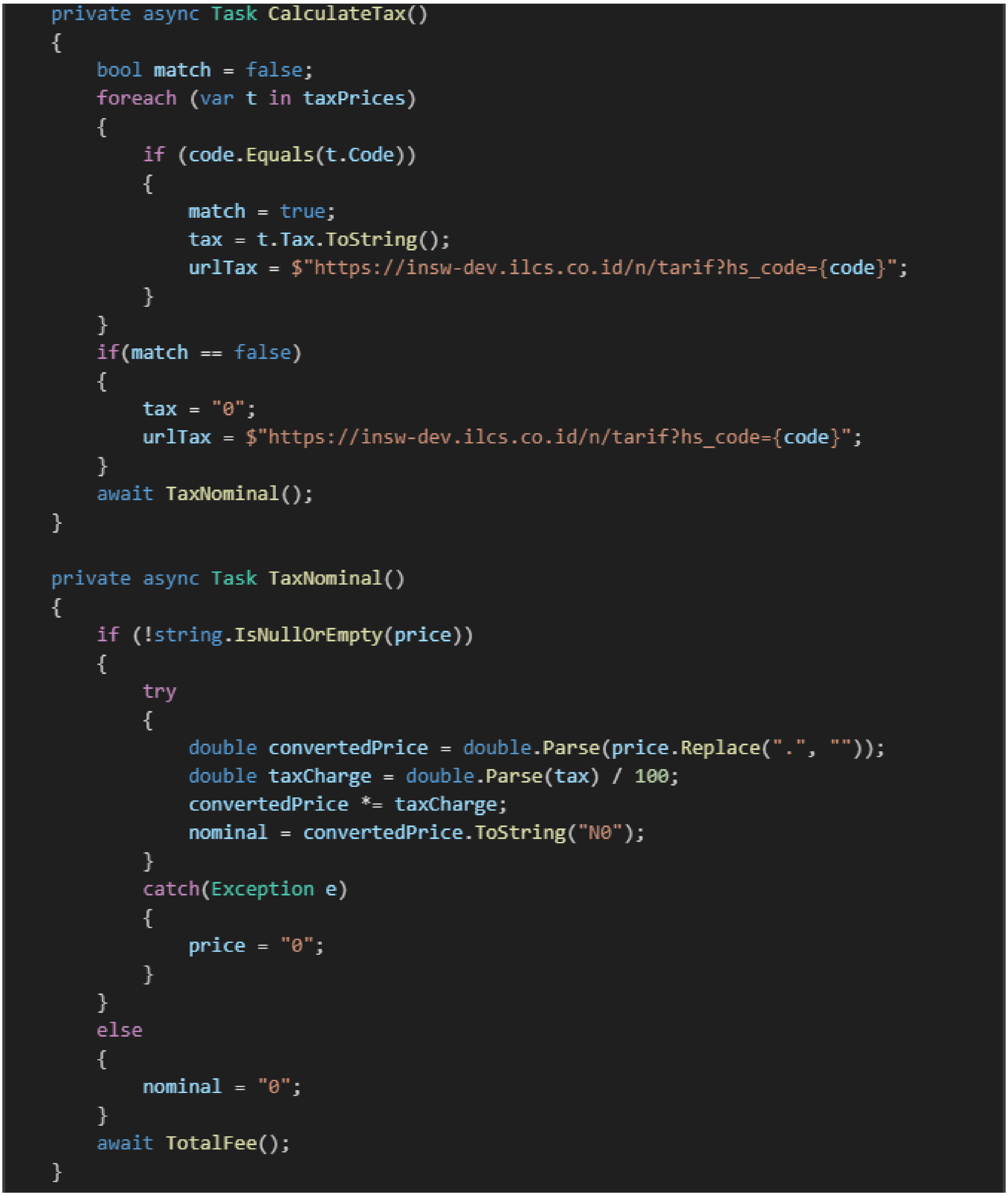
```
@using AutoComplete_App.Models
@rendermode InteractiveServer

@inject AutoComplete_App.Services.CallApi CallApi

<div class="form-group row">
    <div class="col-4">
    <label class="col-3"> Barang</label>
        <input @bind="code" @bind:after="CodeFill" type="text" pattern="[0-9]*" maxlength="8" placeholder="
Please Input Number Only" />
            <label class="col-3"></label>
            <textarea @bind="area" class="col-6 my-2" style="resize:none" disabled rows="3" />
            <p>@urlCode</p>
    </div>
    <div class="form-group row">
        <div class="col-4">
            <label class="col-3"> Harga </label>
            <input @bind="price" @bind:after="TaxNominal" type="text" pattern="[0-9]" maxlength="15" placeholder="1.500.000" />
        </div>
    </div>
    <br />
    <div class="form-group row my-3">
        <div class="col-6">
            <label class="col-3"> Tarif Bea masuk </label>
            <input @bind="tax" @bind:after="CalculateTax" class="w-25" type="number" disabled /> %
            <p class="mx-5"> @urlTax</p>
            <div>
                <label class="col-2"></label>
                <input @bind="nominal" type="text" disabled />
            </div>

        </div>
    </div>
    <div class="form-group row">
        <div class=" col-6">
            <label class="col-2"> Total Biaya </label>
            <input @bind="total" @bind:after="TotalFee" class="col-4" type="text" disabled />
        </div>
    </div>
</div>
```



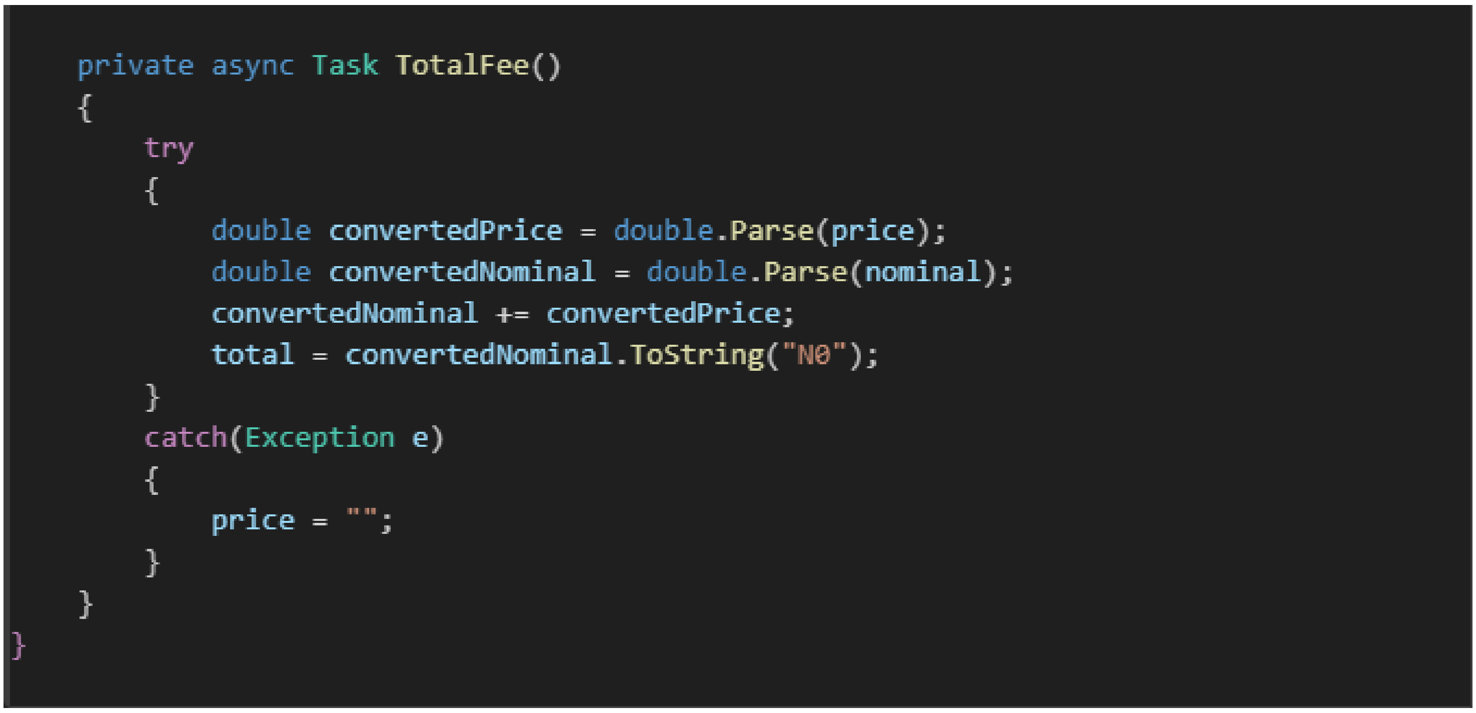
```
@code {
    string code = "";
    string area = "";
    string price = "";
    string tax = "0";
    string nominal = "0";
    string total = "0";
    string urlCode = "https://insw-dev.ilcs.co.id/n/barang?hs_code=";
    string urlTax = "https://insw-dev.ilcs.co.id/n/tarif?hs_code=";
    Inventory[] inventories;
    TaxPrice[] taxPrices;

    protected override async Task OnInitializedAsync()
    {
        CallApi.ReadInventoryList();
        if(CallApi.Inventories != null)
        {
            inventories = CallApi.Inventories;
        }
        CallApi.ReadTaxList();
        if(CallApi.TaxPrices != null)
```



```
    private async Task CodeFill()
    {
        bool codeFound = false;
        if (code.Length != 0)
        {
            foreach(var i in inventories)
            {
                if (i.Code.Equals(code))
                {
                    codeFound = true;
                    area = i.Category;
                    urlCode = $"https://insw-dev.ilcs.co.id/n/barang?hs_code={code}";
                }
            }
        }
        if (codeFound == false)
        {
            area = "Code Not Found";
            urlCode = $"https://insw-dev.ilcs.co.id/n/barang?hs_code={code}";
        }
        await CalculateTax();
    }
```



```
private async Task CalculateTax()
{
    bool match = false;
    foreach (var t in taxPrices)
    {
        if (code.Equals(t.Code))
        {
            match = true;
            tax = t.Tax.ToString();
            urlTax = $"https://insw-dev.ilcs.co.id/n/tarif?hs_code={code}";
        }
    }
    if(match == false)
    {
        tax = "0";
        urlTax = $"https://insw-dev.ilcs.co.id/n/tarif?hs_code={code}";
    }
    await TaxNominal();
}

private async Task TaxNominal()
{
    if (!string.IsNullOrEmpty(price))
    {
        try
        {
            double convertedPrice = double.Parse(price.Replace(".", ""));
            double taxCharge = double.Parse(tax) / 100;
            convertedPrice *= taxCharge;
            nominal = convertedPrice.ToString("N0");
        }
        catch(Exception e)
        {
            price = "0";
        }
    }
    else
    {
        nominal = "0";
    }
    await TotalFee();
}
```



```
    private async Task TotalFee()
    {
        try
        {
            double convertedPrice = double.Parse(price);
            double convertedNominal = double.Parse(nominal);
            convertedNominal += convertedPrice;
            total = convertedNominal.ToString("N0");
        }
        catch(Exception e)
        {
            price = "";
        }
    }
}
```

I use ToString("NO") to convert the format to have decimal like (1.500.000) etc.