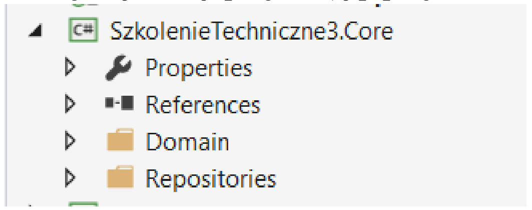
Szkolenie Techniczne 3

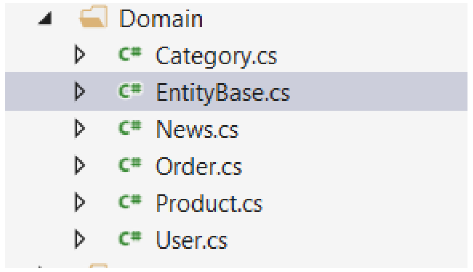
Infrastruktura projektu (API)

1. Dodajemy nowy projekt (typ projektu biblioteka klas dll) o nazwie SzkolenieTechniczne3.Core



```
◢ SzkolenieTechniczne3.Core
   ▷ Properties
   ▷ References
   ▷ Domain
   ▷ Repositories
```

Dodajemy katalogi **Domain** i **Repositories**.
W katalogu **Domain** dodajemy klasy odzwierciedlające strukturę bazy danych.
Dodajemy klase bazową EntityBase.



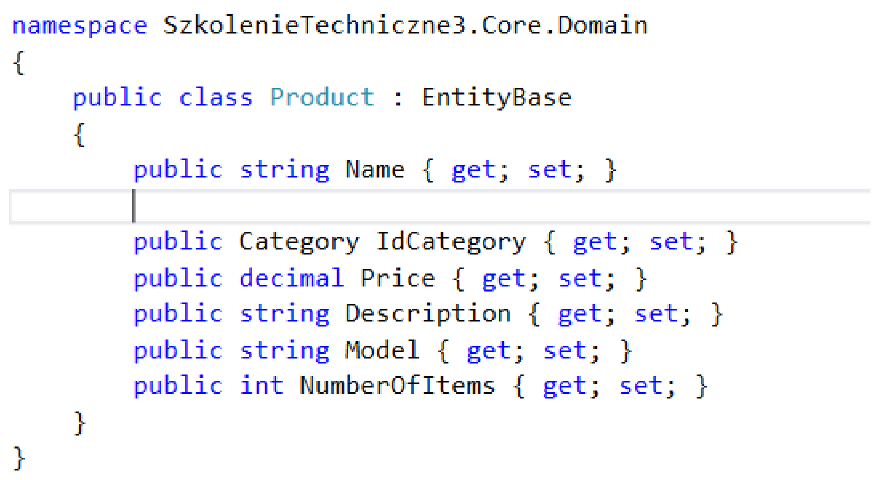
```
◢ Domain
   ▷ Category.cs
   ▷ EntityBase.cs
   ▷ News.cs
   ▷ Order.cs
   ▷ Product.cs
   ▷ User.cs
```

```
namespace SzkolenieTechniczne3.Core.Domain
{
   public class EntityBase
    {
        public int Id { get; set; }
    }
}
```

Przykład mapowań klas. Nazwy właściwości adekwatne do kolumn

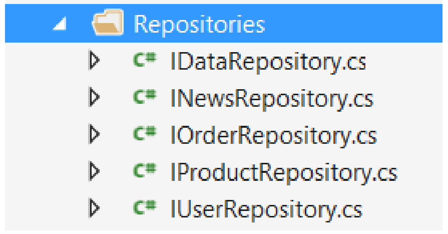
```
namespace SzkolenieTechniczne3.Core.Domain
{
   public class User : EntityBase
    {
        public string Name { get; set; }
        public string Surname { get; set; }
        public string PersonIdNumber { get; set; }
        public string  Phone { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }

    }
}
```

```
namespace SzkolenieTechniczne3.Core.Domain
{
   public  class Category : EntityBase
    {
        private string Name { get; set; }

    }
}
```



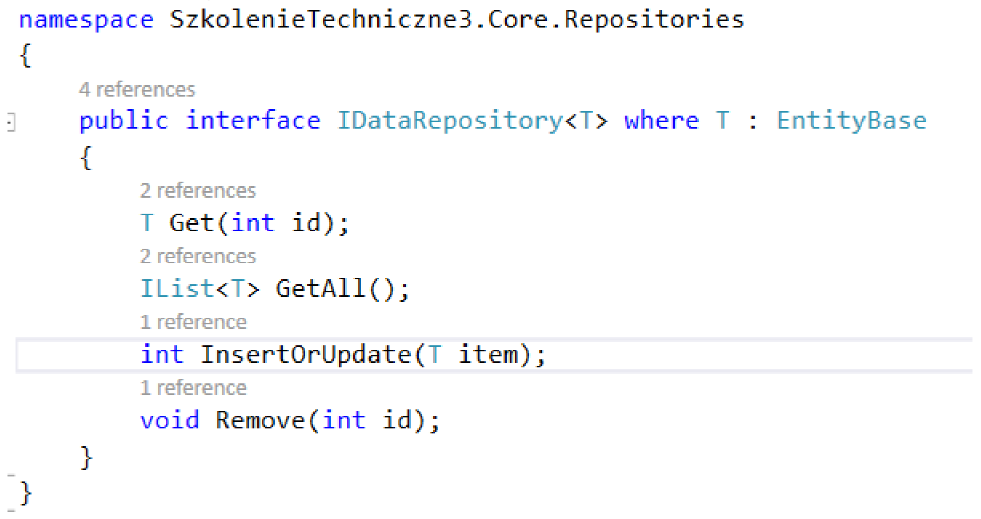
```
namespace SzkolenieTechniczne3.Core.Domain
{
    public class Product : EntityBase
    {
        public string Name { get; set; }

        public Category IdCategory { get; set; }
        public decimal Price { get; set; }
        public string Description { get; set; }
        public string Model { get; set; }
        public int NumberOfItems { get; set; }
    }
}
```

Katalog Repositories:
Dodajemy interfejsy według poniższej listy.



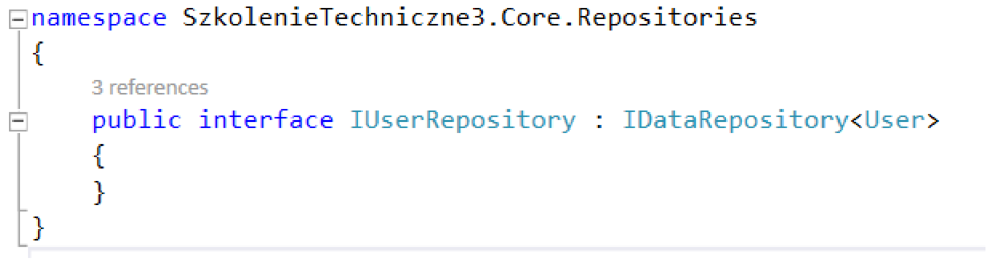
```
Repositories
    IDataRepository.cs
    INewsRepository.cs
    IOrderRepository.cs
    IProductRepository.cs
    IUserRepository.cs
```

Interfejs bazowy:

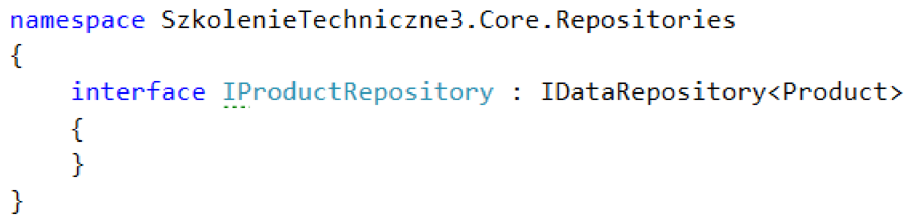


```
namespace SzkolenieTechniczne3.Core.Repositories
{
    4 references
    public interface IDataRepository<T> where T : EntityBase
    {
        2 references
        T Get(int id);
        2 references
        IList<T> GetAll();
        1 reference
        int InsertOrUpdate(T item);
        1 reference
        void Remove(int id);
    }
}
```

Dodanie interfejsu dla IUserRepository

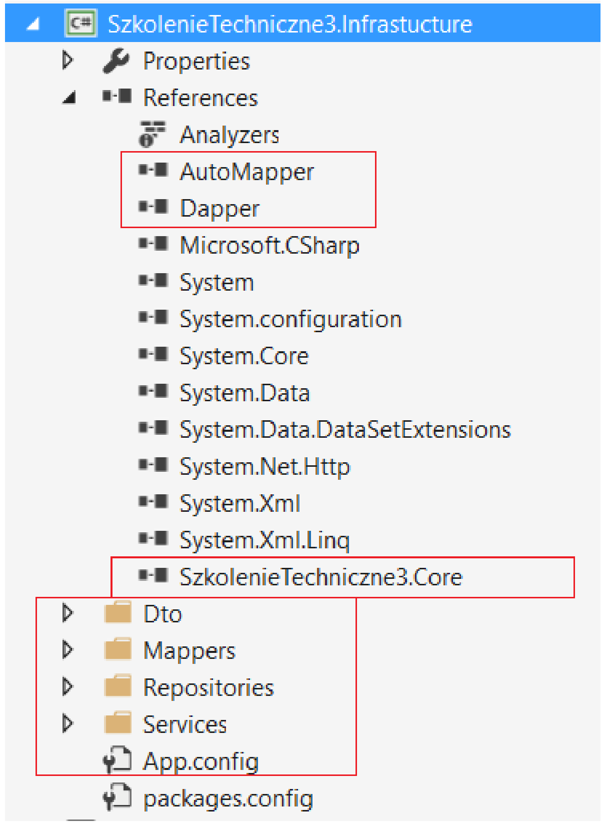


```
namespace SzkolenieTechniczne3.Core.Repositories
{
    3 references
    public interface IUserRepository : IDataRepository<User>
    {
    }
}
```



```
namespace SzkolenieTechniczne3.Core.Repositories
{
    interface IProductRepository : IDataRepository<Product>
    {
    }
}
```

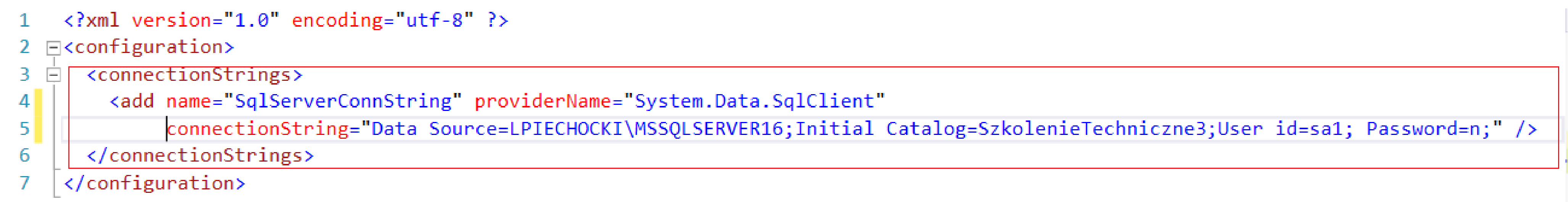
Dodajemy nowy projekt **Szkolenie.Techniczne3.Infrastucture**:


SzkolenieTechniczne3.Infrastucture
Properties
References
Analyzers
AutoMapper
Dapper
Microsoft.CSharp
System
System.configuration
System.Core
System.Data
System.Data.DataSetExtensions
System.Net.Http
System.Xml
System.Xml.Linq
SzkolenieTechniczne3.Core
Dto
Mappers
Repositories
Services
App.config
packages.config

Dodajemy referencje do projektu
AutoMapper
Dapper
Projekt: SzkolenieTechniczne3.Core

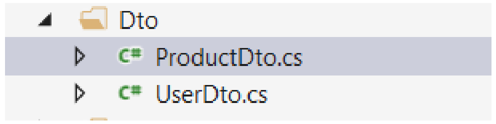
Dodajemy katalogi: **Dto,Mappers, Repositories, Services**

Dodajemy plik konfiguracyjny **App.config**

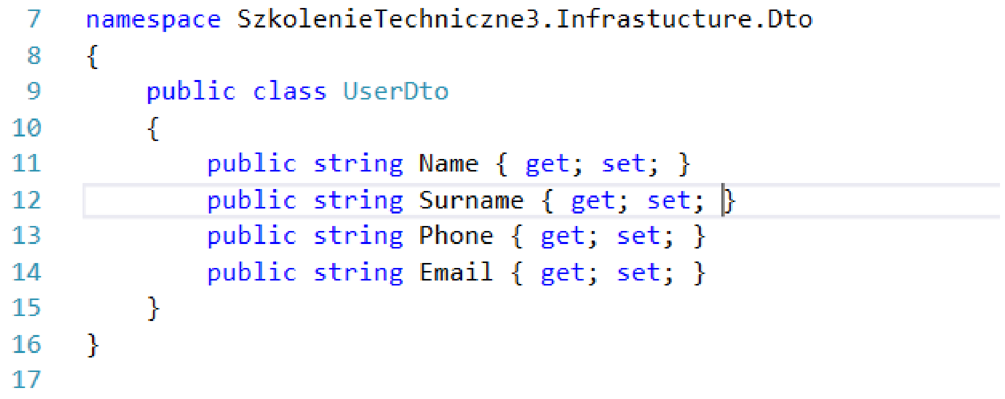


```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="SqlServerConnString" providerName="System.Data.SqlClient"
         connectionString="Data Source=LPIECHOCKI\MSSQLSERVER16;Initial Catalog=SzkolenieTechniczne3;User id=sa1; Password=n;" />
  </connectionStrings>
</configuration>
```

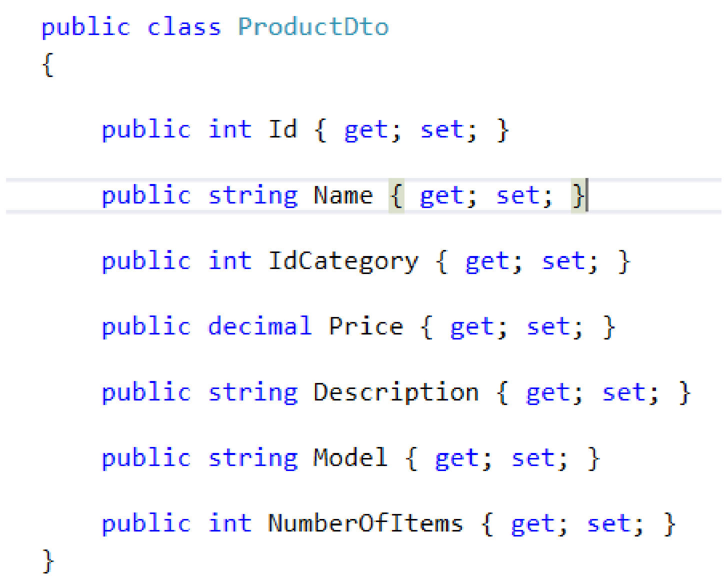
Dodajemy klasy do katalogu Dto



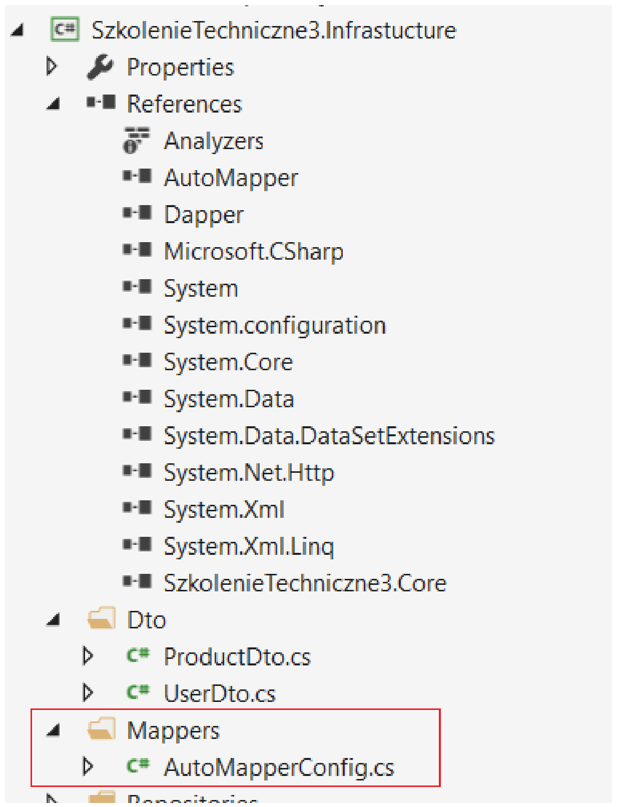
```
Dto
    ProductDto.cs
    UserDto.cs
```



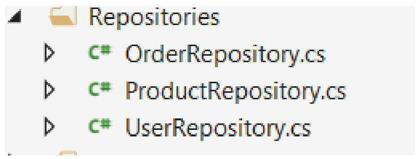
```csharp
namespace SzkolenieTechniczne3.Infrastucture.Dto
{
    public class UserDto
    {
        public string Name { get; set; }
        public string Surname { get; set; }
        public string Phone { get; set; }
        public string Email { get; set; }
    }
}
```

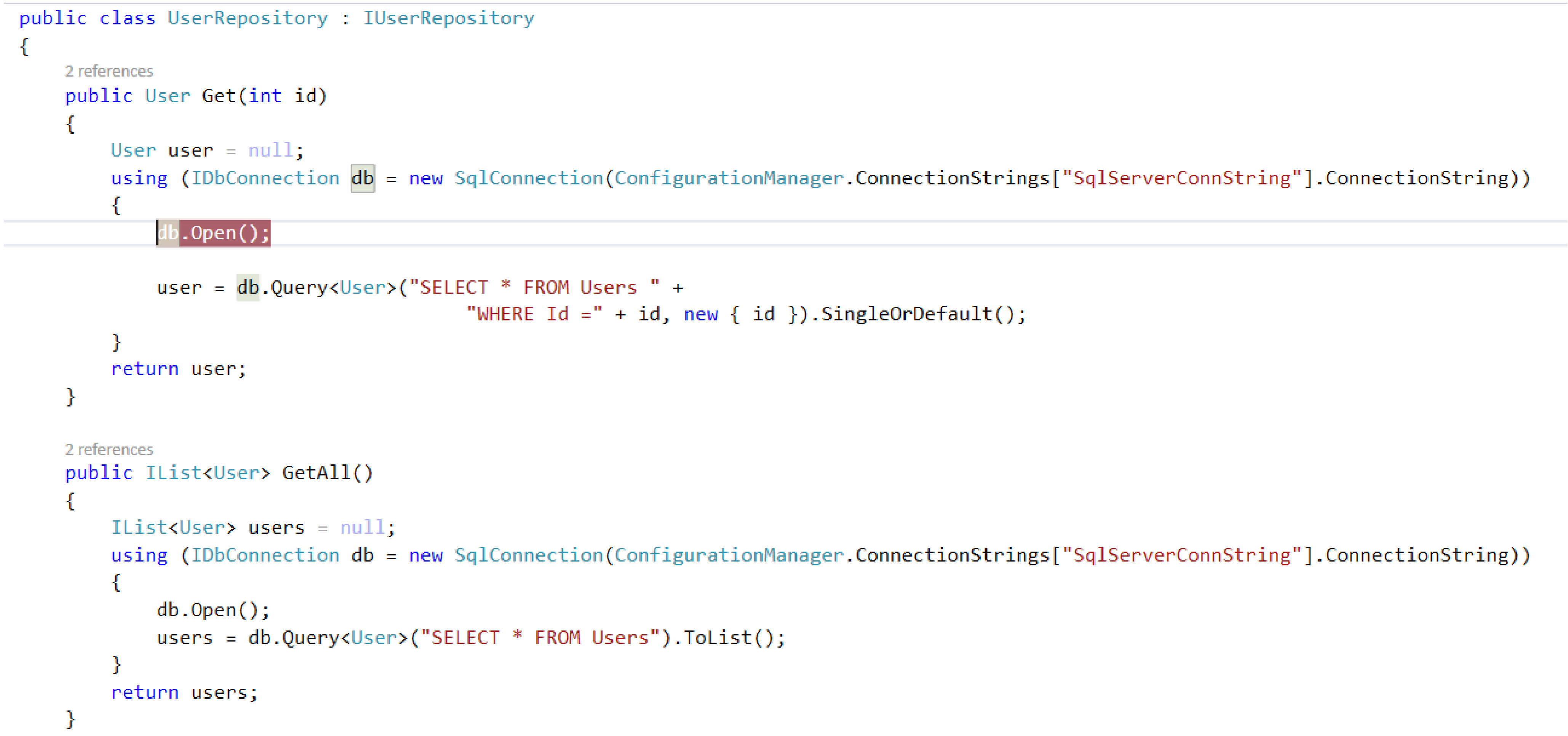


```
public class ProductDto
{

    public int Id { get; set; }

    public string Name { get; set; }

    public int IdCategory { get; set; }

    public decimal Price { get; set; }

    public string Description { get; set; }

    public string Model { get; set; }

    public int NumberOfItems { get; set; }
}
```

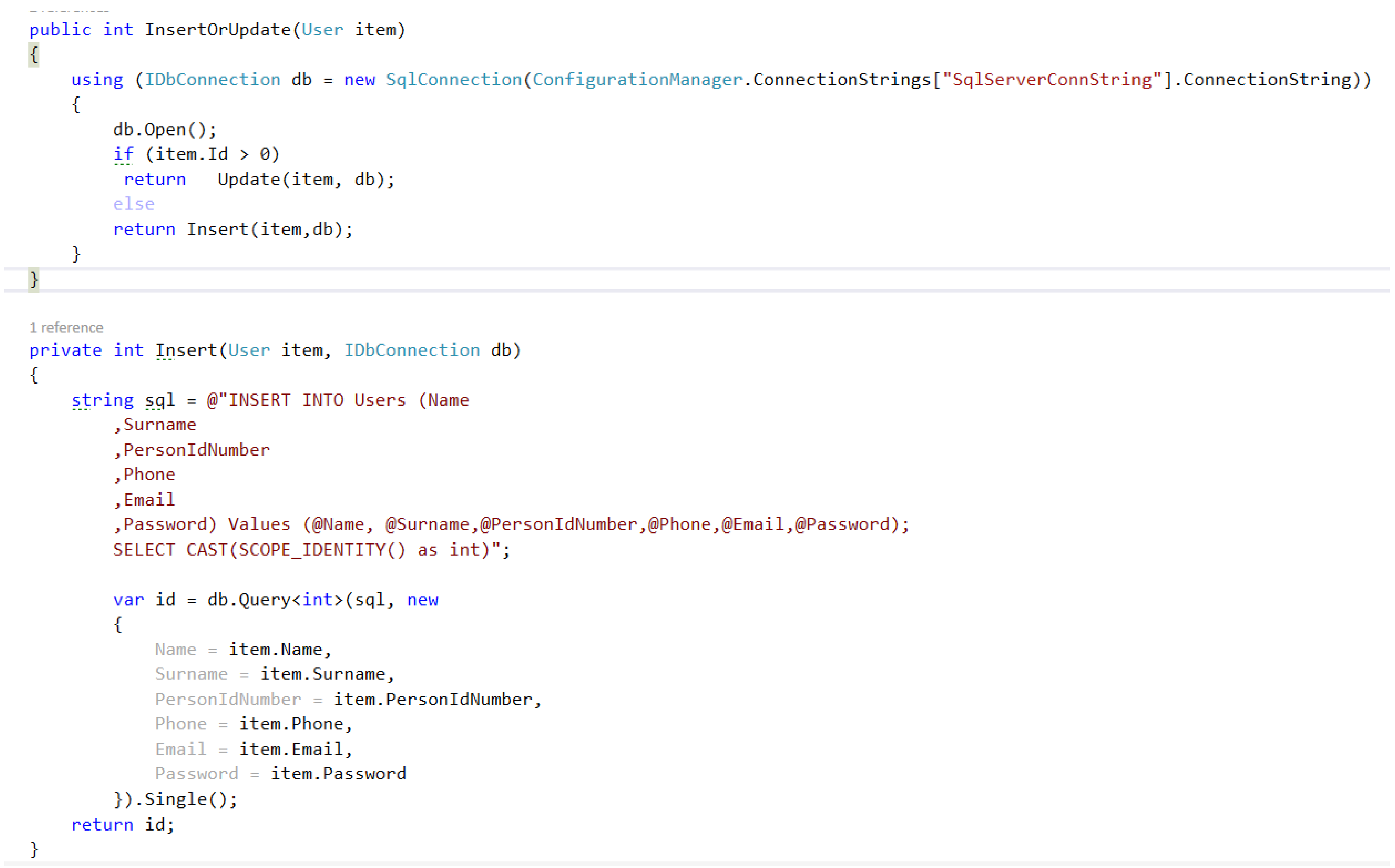

SzkolenieTechniczne3.Infrastucture
Properties
References
Analyzers
AutoMapper
Dapper
Microsoft.CSharp
System
System.configuration
System.Core
System.Data
System.Data.DataSetExtensions
System.Net.Http
System.Xml
System.Xml.Linq
SzkolenieTechniczne3.Core
Dto
ProductDto.cs
UserDto.cs
Mappers
AutoMapperConfig.cs

```
3 references
public static class AutoMapperConfig
{
    3 references | 0/2 passing
    public static IMapper Initialize()
        => new MapperConfiguration(cfg =>
        {
            cfg.CreateMap<UserDto, User>();
            cfg.CreateMap<User, UserDto>();
            cfg.CreateMap<Product, ProductDto>()
                .ForMember(x => x.IdCategory, m => m.MapFrom(p => p.IdCategory.Id));
          }).CreateMapper();
}
```



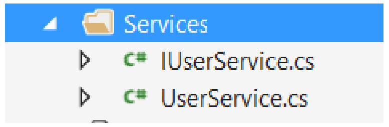


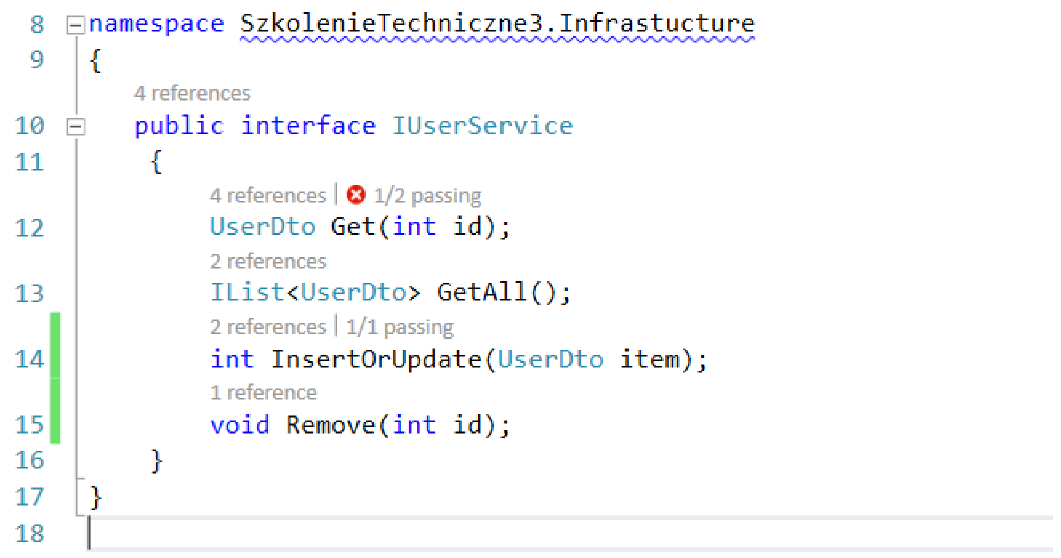
```
public class UserRepository : IUserRepository
{
    2 references
    public User Get(int id)
    {
        User user = null;
        using (IDbConnection db = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlServerConnString"].ConnectionString))
        {
            db.Open();

            user = db.Query<User>("SELECT * FROM Users " +
                                  "WHERE Id =" + id, new { id }).SingleOrDefault();
        }
        return user;
    }

    2 references
    public IList<User> GetAll()
    {
        IList<User> users = null;
        using (IDbConnection db = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlServerConnString"].ConnectionString))
        {
            db.Open();
            users = db.Query<User>("SELECT * FROM Users").ToList();
        }
        return users;
    }
```



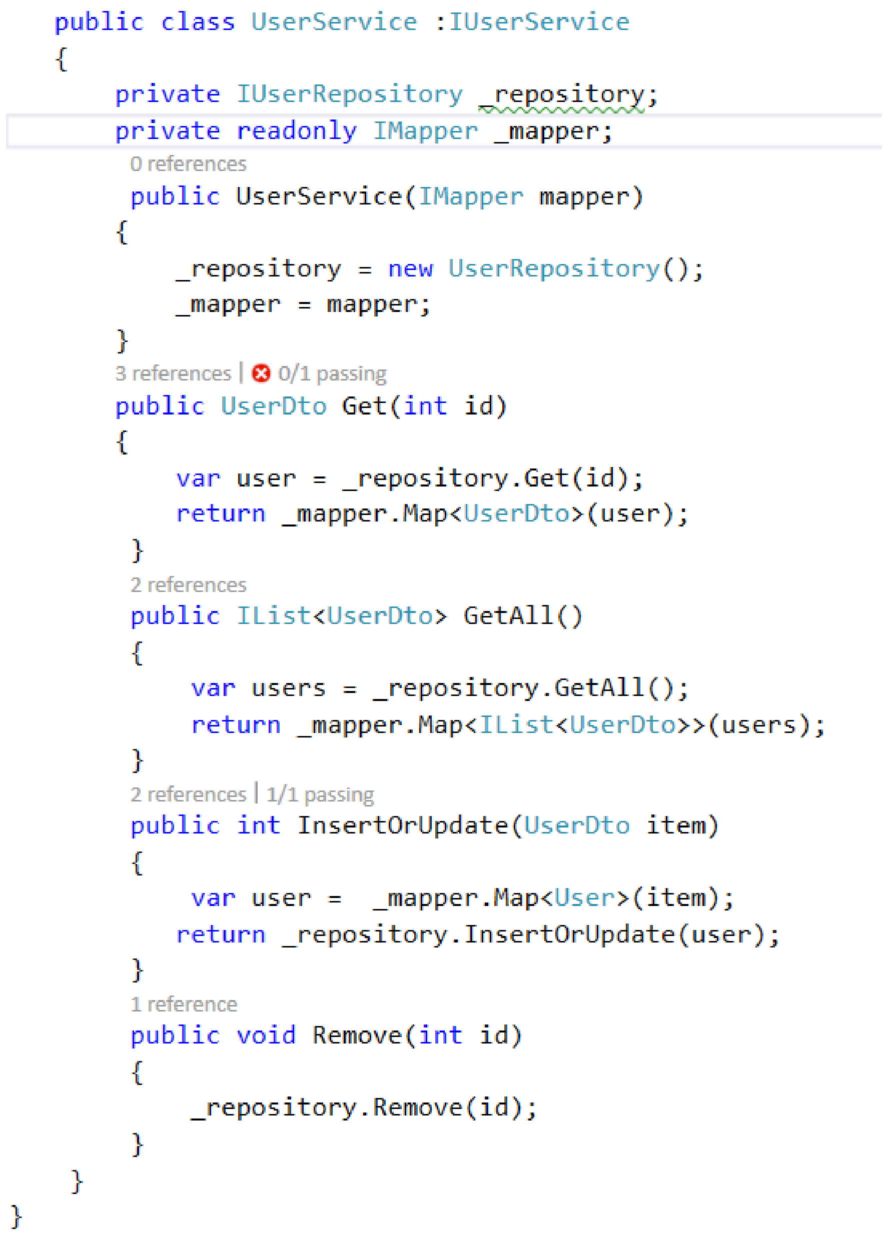
```
public int InsertOrUpdate(User item)
{
    using (IDbConnection db = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlServerConnString"].ConnectionString))
    {
        db.Open();
        if (item.Id > 0)
         return   Update(item, db);
        else
        return Insert(item,db);
    }
}

1 reference
private int Insert(User item, IDbConnection db)
{
    string sql = @"INSERT INTO Users (Name
        ,Surname
        ,PersonIdNumber
        ,Phone
        ,Email
        ,Password) Values (@Name, @Surname,@PersonIdNumber,@Phone,@Email,@Password);
        SELECT CAST(SCOPE_IDENTITY() as int)";

        var id = db.Query<int>(sql, new
        {
            Name = item.Name,
            Surname = item.Surname,
            PersonIdNumber = item.PersonIdNumber,
            Phone = item.Phone,
            Email = item.Email,
            Password = item.Password
        }).Single();
    return id;
}
```

```
private int Update(User item, IDbConnection db)
{
    const string sql = @"UPDATE Users SET Name = @Name,Surname= @Surname,PersonIdNumber= @PersonIdNumber,
                                Phone= @Phone, Email = @Email, Password= @Password
                WHERE Id = @Id;";

    var affectedRows = db.Execute(sql, new
    {  Id = item.Id,
        Name = item.Name,
        Surname = item.Surname,
        PersonIdNumber = item.PersonIdNumber,
        Phone = item.Phone,
        Email = item.Email,
        Password = item.Password
    });
    return affectedRows;
}

2 references
public void Remove(int id)
{
    string sql = "DELETE FROM User WHERE Id = @Id";

    using (IDbConnection db = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlServerConnString"].ConnectionString))
    {
        db.Open();
        var affectedRows = db.Execute(sql, new { Id = id });
    }
}
```



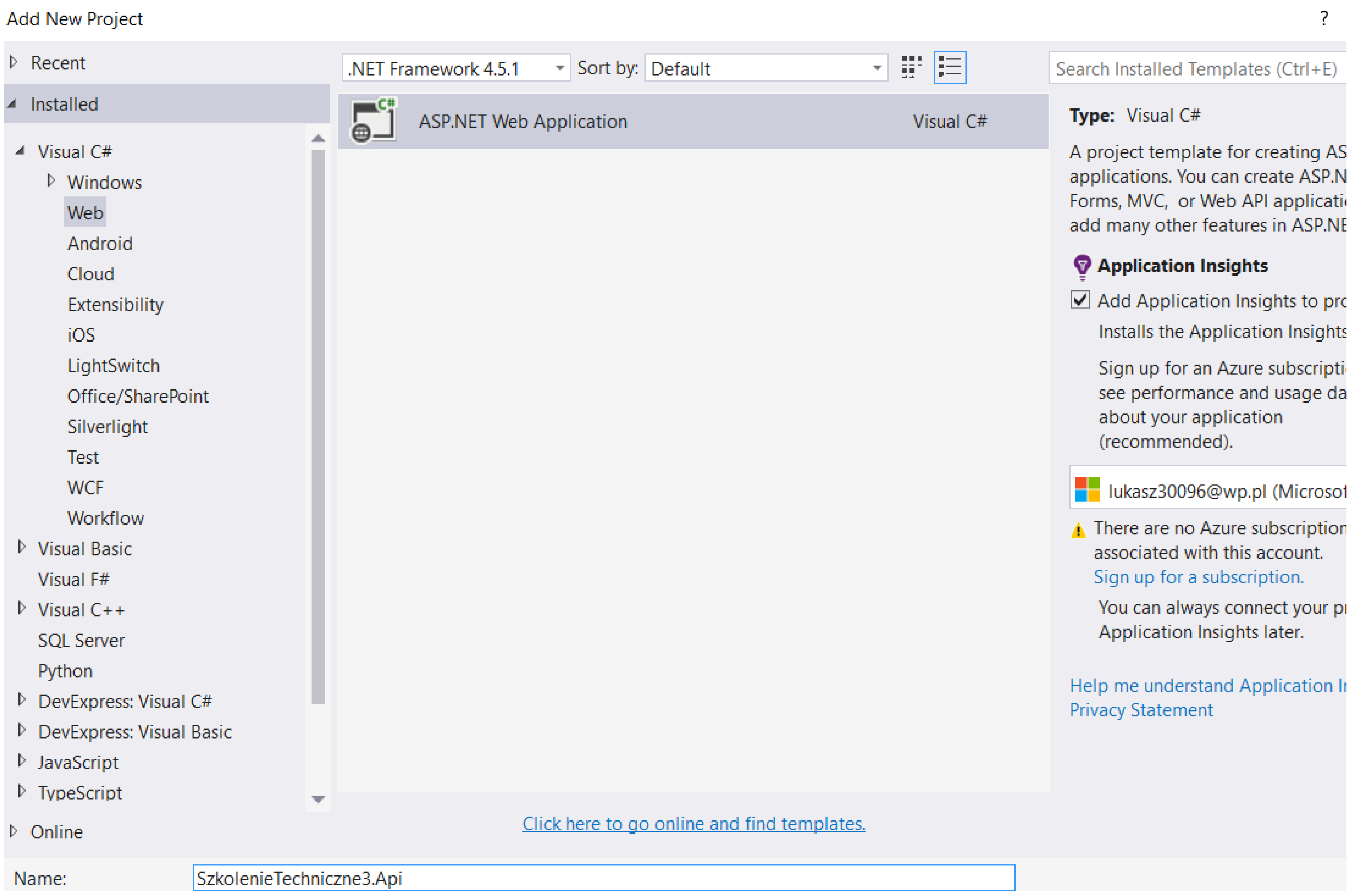


```
namespace SzkolenieTechniczne3.Infrastucture
{
    4 references
    public interface IUserService
     {
        4 references | 1/2 passing
        UserDto Get(int id);
        2 references
        IList<UserDto> GetAll();
        2 references | 1/1 passing
        int InsertOrUpdate(UserDto item);
        1 reference
        void Remove(int id);
     }
}

```



```
    public class UserService :IUserService
    {
        private IUserRepository _repository;
        private readonly IMapper _mapper;
         0 references
         public UserService(IMapper mapper)
        {
            _repository = new UserRepository();
            _mapper = mapper;
        }
        3 references | ⊗ 0/1 passing
        public UserDto Get(int id)
        {
            var user = _repository.Get(id);
            return _mapper.Map<UserDto>(user);
         }
         2 references
         public IList<UserDto> GetAll()
         {
             var users = _repository.GetAll();
             return _mapper.Map<IList<UserDto>>(users);
         }
         2 references | 1/1 passing
         public int InsertOrUpdate(UserDto item)
         {
             var user =  _mapper.Map<User>(item);
            return _repository.InsertOrUpdate(user);
         }
         1 reference
         public void Remove(int id)
         {
             _repository.Remove(id);
         }
     }
}
```

Dodanie nowego projektu **SzkolenieTechniczne3.Api**


Add New Project ?

Recent

Installed

- Visual C#
  - Windows
  - Web
  - Android
  - Cloud
  - Extensibility
  - iOS
  - LightSwitch
  - Office/SharePoint
  - Silverlight
  - Test
  - WCF
  - Workflow
- Visual Basic
- Visual F#
- Visual C++
- SQL Server
- Python
- DevExpress: Visual C#
- DevExpress: Visual Basic
- JavaScript
- TypeScript

Online

.NET Framework 4.5.1 Sort by: Default

Search Installed Templates (Ctrl+E)

ASP.NET Web Application Visual C#

Click here to go online and find templates.

Type: Visual C#

A project template for creating AS
applications. You can create ASP.N
Forms, MVC, or Web API applicati
add many other features in ASP.NE

Application Insights

[✓] Add Application Insights to pro

Installs the Application Insights

Sign up for an Azure subscripti
see performance and usage da
about your application
(recommended).

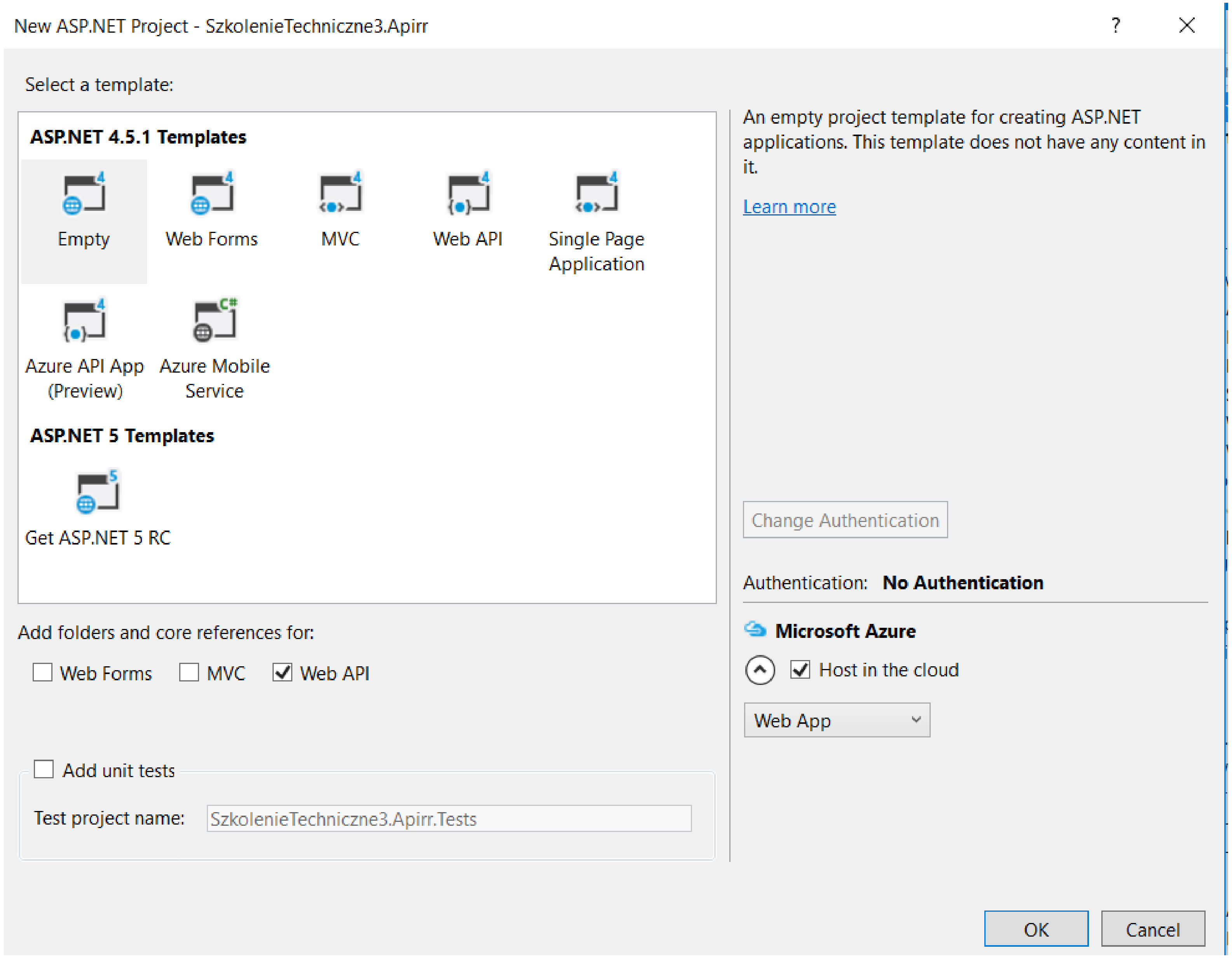
lukasz30096@wp.pl (Microso

There are no Azure subscription
associated with this account.
Sign up for a subscription.

You can always connect your pi
Application Insights later.

Help me understand Application I
Privacy Statement

Name: SzkolenieTechniczne3.Api



New ASP.NET Project - SzkolenieTechniczne3.Apirr

Select a template:

ASP.NET 4.5.1 Templates

Empty, Web Forms, MVC, Web API, Single Page Application, Azure API App (Preview), Azure Mobile Service

ASP.NET 5 Templates

Get ASP.NET 5 RC

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Learn more

Change Authentication

Authentication: **No Authentication**

Microsoft Azure

[✓] Host in the cloud

Web App

Add folders and core references for:

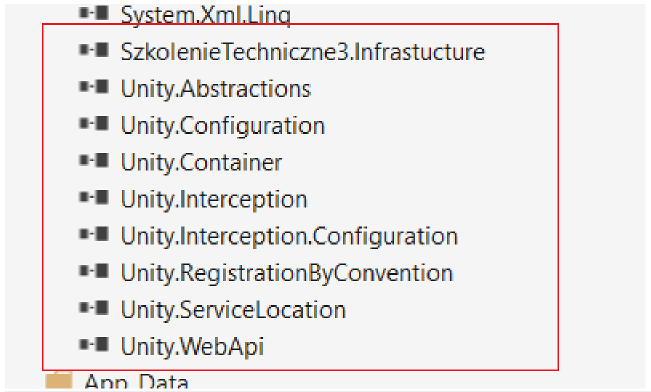
[ ] Web Forms [ ] MVC [✓] Web API

[ ] Add unit tests

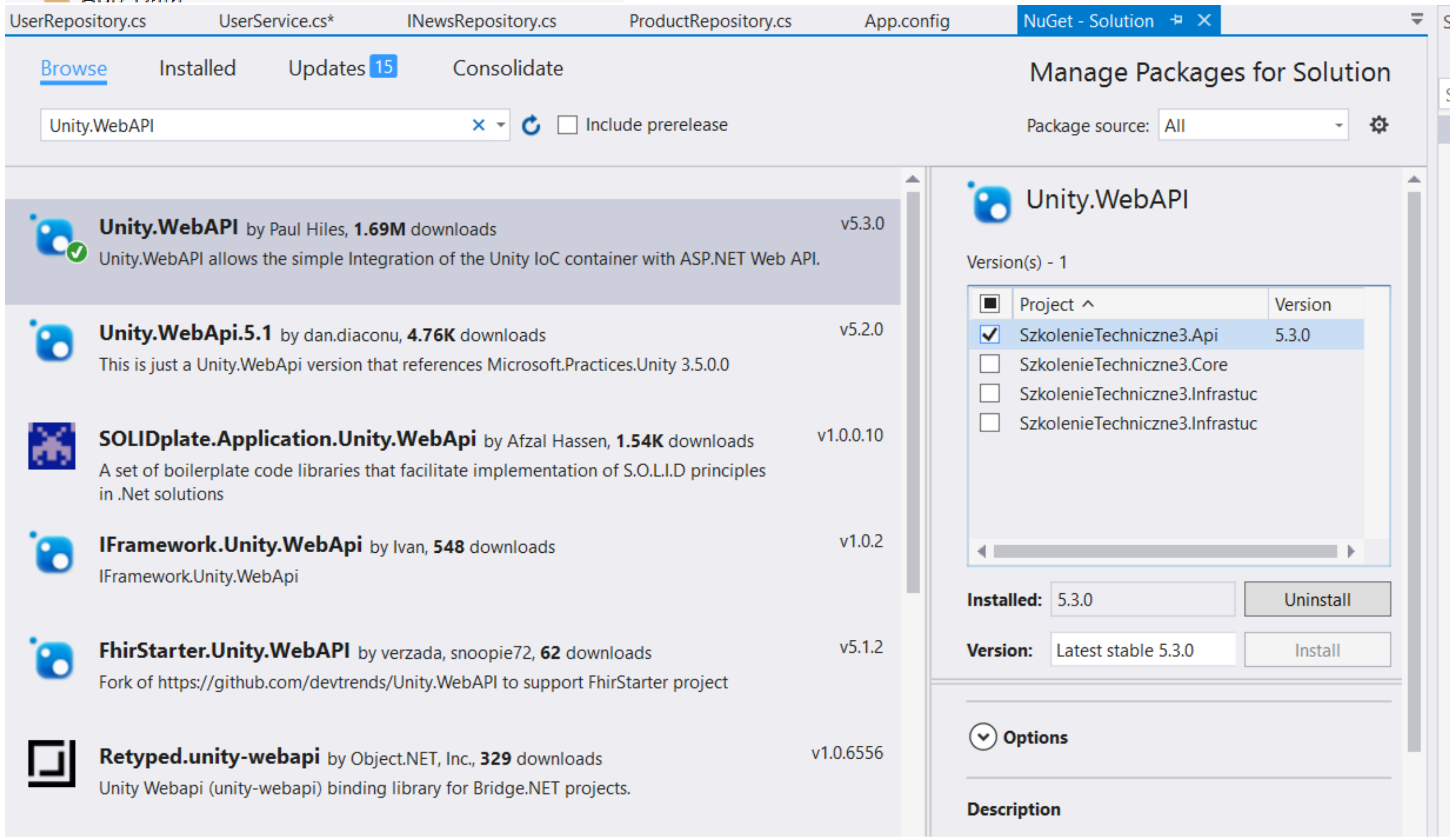
Test project name: SzkolenieTechniczne3.Apirr.Tests

OK Cancel

Dodajemy referencje w projekcie do Szkolenie.Techczne3.Infrastucture oraz instalujemy pakiet Unity.WebApi lub dodajemy referencje wszystkie z katalogu api.lib



System.Xml.Linq
SzkolenieTechniczne3.Infrastucture
Unity.Abstractions
Unity.Configuration
Unity.Container
Unity.Interception
Unity.Interception.Configuration
Unity.RegistrationByConvention
Unity.ServiceLocation
Unity.WebApi
App_Data



Unity.WebAPI by Paul Hiles, **1.69M** downloads v5.3.0
Unity.WebAPI allows the simple Integration of the Unity IoC container with ASP.NET Web API.

Unity.WebApi.5.1 by dan.diaconu, **4.76K** downloads v5.2.0
This is just a Unity.WebApi version that references Microsoft.Practices.Unity 3.5.0.0

SOLIDplate.Application.Unity.WebApi by Afzal Hassen, **1.54K** downloads v1.0.0.10
A set of boilerplate code libraries that facilitate implementation of S.O.L.I.D principles in .Net solutions

IFramework.Unity.WebApi by Ivan, **548** downloads v1.0.2
IFramework.Unity.WebApi

FhirStarter.Unity.WebAPI by verzada, snoopie72, **62** downloads v5.1.2
Fork of https://github.com/devtrends/Unity.WebAPI to support FhirStarter project

Retyped.unity-webapi by Object.NET, Inc., **329** downloads v1.0.6556
Unity Webapi (unity-webapi) binding library for Bridge.NET projects.

Unity.WebAPI

Version(s) - 1

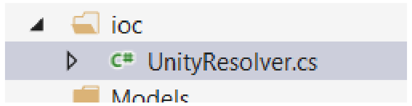
	Project	Version
✓	SzkolenieTechniczne3.Api	5.3.0
☐	SzkolenieTechniczne3.Core	
☐	SzkolenieTechniczne3.Infrastuc	
☐	SzkolenieTechniczne3.Infrastuc	

Installed: 5.3.0 Uninstall
Version: Latest stable 5.3.0 Install

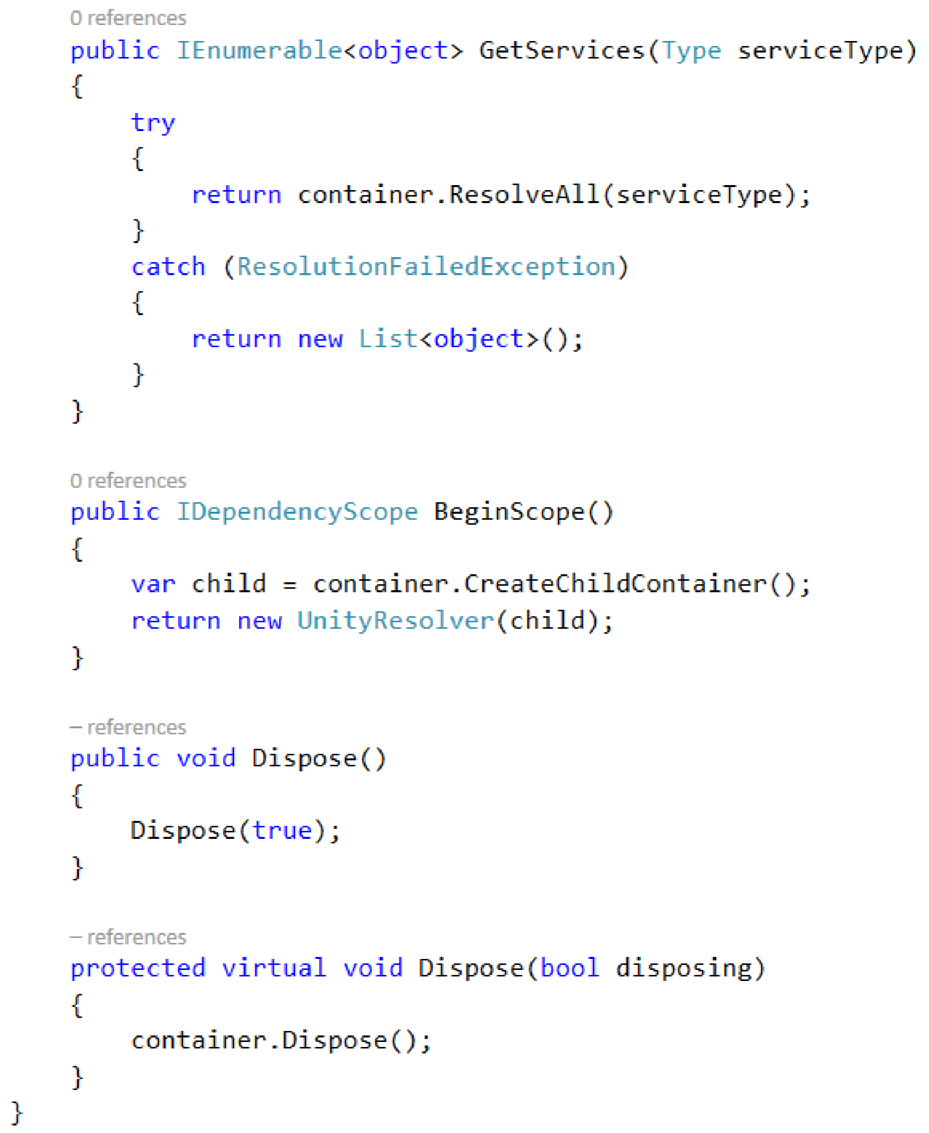
Options

Description

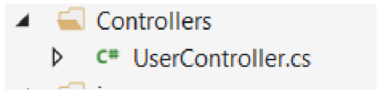
Dodajemy katalog ioc oraz klasę UnityResolver.cs

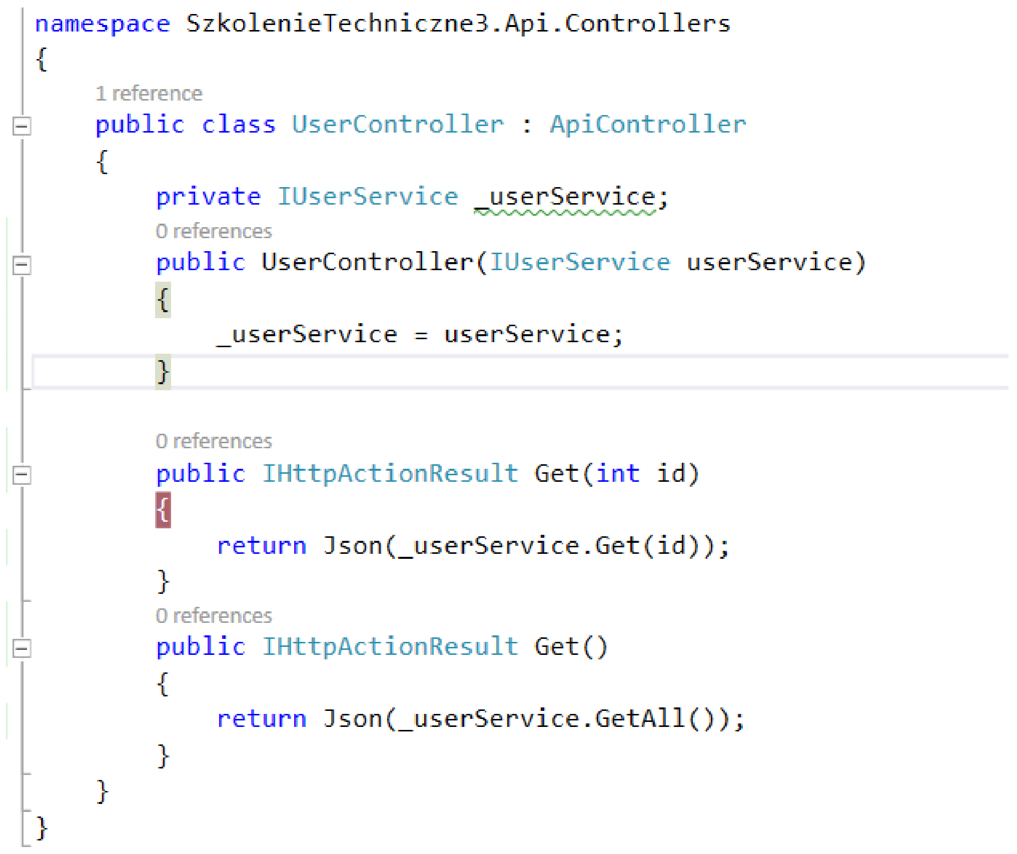


```
public class UnityResolver : IDependencyResolver
{
    protected IUnityContainer container;
    2 references
    public UnityResolver(IUnityContainer container)
    {
        if (container == null)
        {
            throw new ArgumentNullException("container");
        }

        this.container = container;
    }

    0 references
    public object GetService(Type serviceType)
    {
        try
        {
            return container.Resolve(serviceType);
        }
        catch (ResolutionFailedException)
        {
            return null;
        }
    }
```



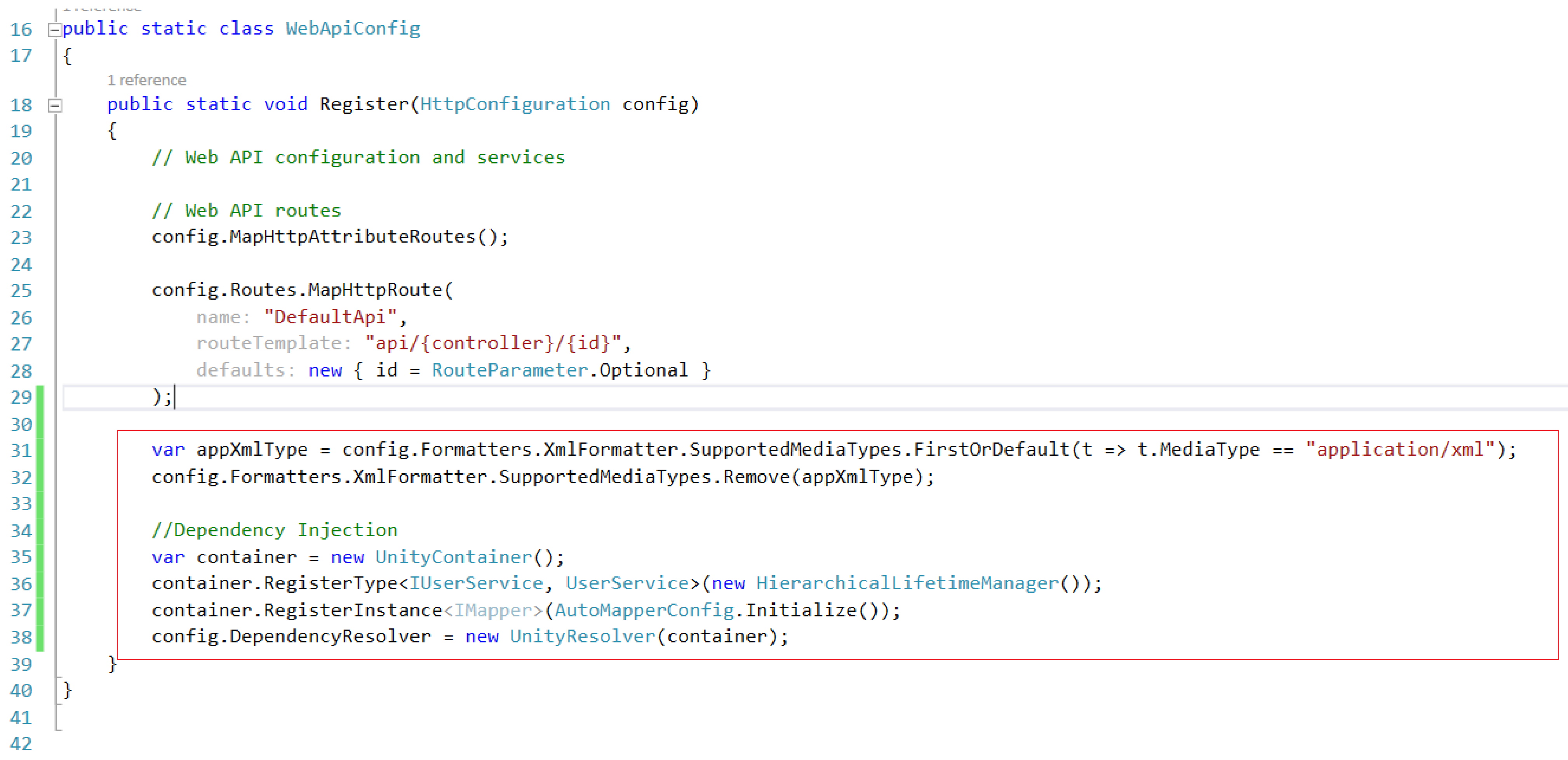
```
        0 references
        public IEnumerable<object> GetServices(Type serviceType)
        {
            try
            {
                return container.ResolveAll(serviceType);
            }
            catch (ResolutionFailedException)
            {
                return new List<object>();
            }
        }

        0 references
        public IDependencyScope BeginScope()
        {
            var child = container.CreateChildContainer();
            return new UnityResolver(child);
        }

        – references
        public void Dispose()
        {
            Dispose(true);
        }

        – references
        protected virtual void Dispose(bool disposing)
        {
            container.Dispose();
        }
    }
```





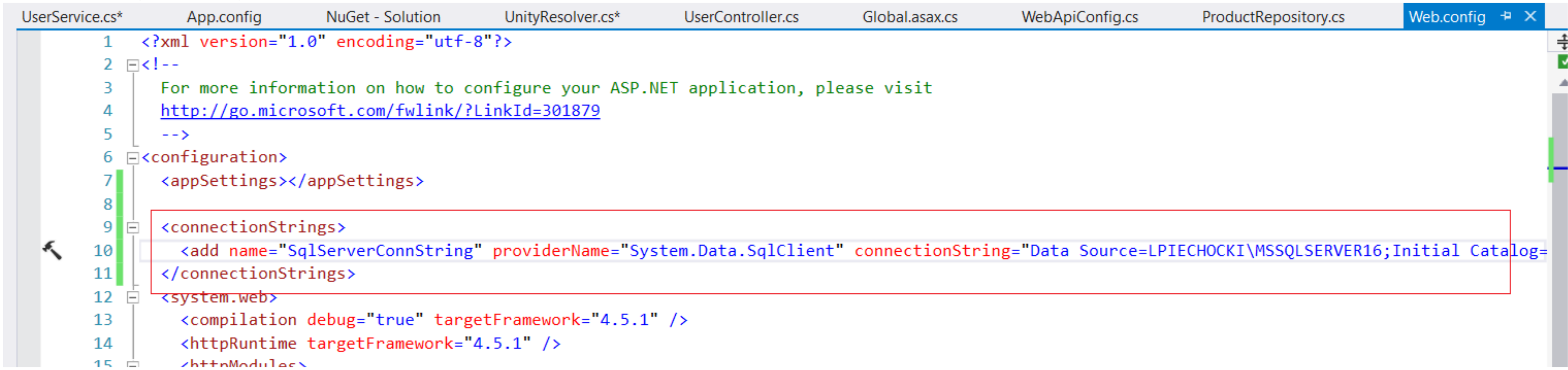
```
namespace SzkolenieTechniczne3.Api.Controllers
{
    1 reference
    public class UserController : ApiController
    {
        private IUserService _userService;
        0 references
        public UserController(IUserService userService)
        {
            _userService = userService;
        }

        0 references
        public IHttpActionResult Get(int id)
        {
            return Json(_userService.Get(id));
        }
        0 references
        public IHttpActionResult Get()
        {
            return Json(_userService.GetAll());
        }
    }
}
```

Ioc konfiguracja przy starcie aplikacji:



```
public static class WebApiConfig
{
    1 reference
    public static void Register(HttpConfiguration config)
    {
        // Web API configuration and services

        // Web API routes
        config.MapHttpAttributeRoutes();

        config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/{controller}/{id}",
            defaults: new { id = RouteParameter.Optional }
        );

        var appXmlType = config.Formatters.XmlFormatter.SupportedMediaTypes.FirstOrDefault(t => t.MediaType == "application/xml");
        config.Formatters.XmlFormatter.SupportedMediaTypes.Remove(appXmlType);

        //Dependency Injection
        var container = new UnityContainer();
        container.RegisterType<IUserService, UserService>(new HierarchicalLifetimeManager());
        container.RegisterInstance<IMapper>(AutoMapperConfig.Initialize());
        config.DependencyResolver = new UnityResolver(container);
    }
}
```

W Web.config dodajemy



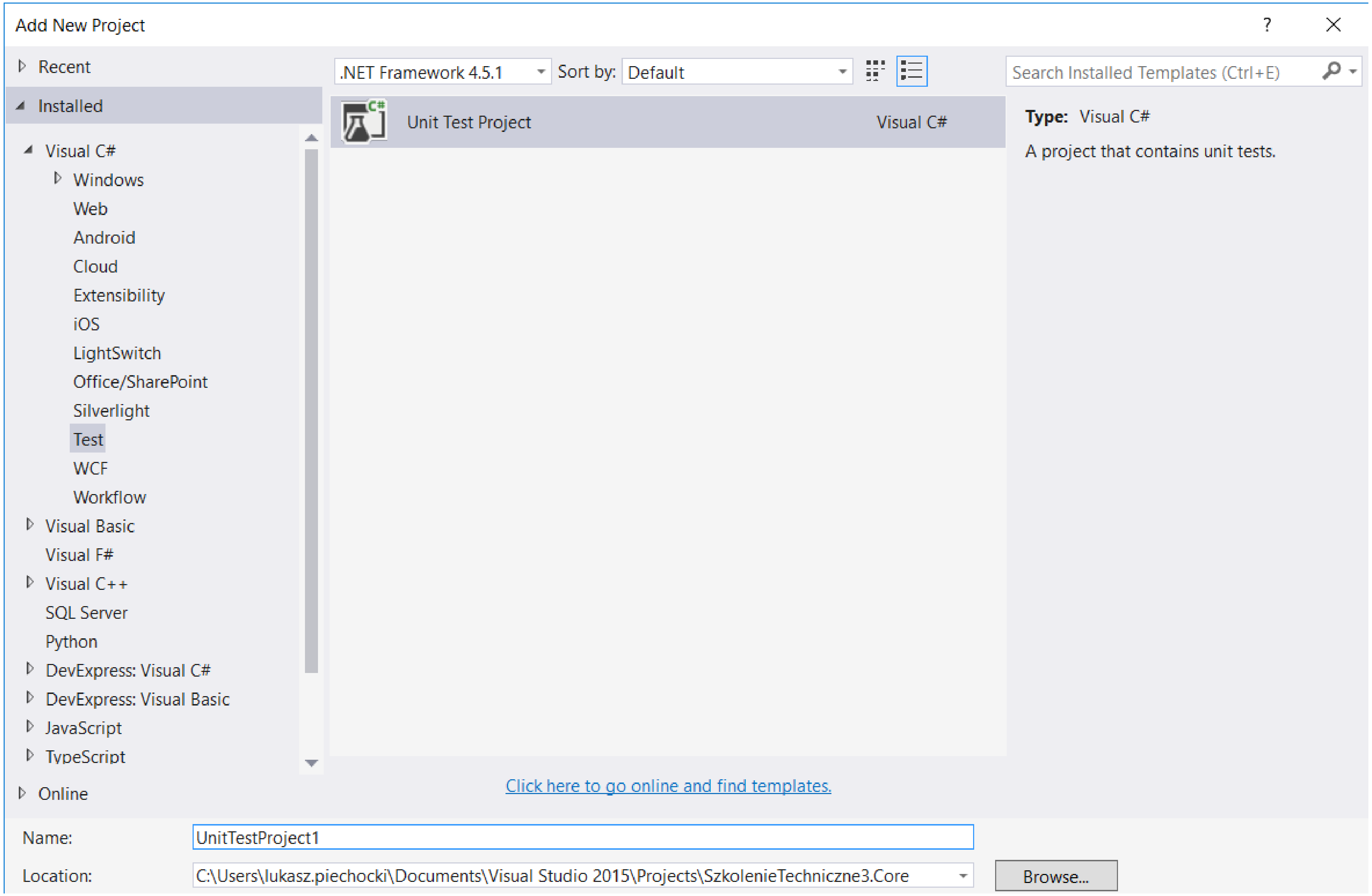
UserService.cs* App.config NuGet - Solution UnityResolver.cs* UserController.cs Global.asax.cs WebApiConfig.cs ProductRepository.cs Web.config

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=301879
  -->
<configuration>
  <appSettings></appSettings>

  <connectionStrings>
    <add name="SqlServerConnString" providerName="System.Data.SqlClient" connectionString="Data Source=LPIECHOCKI\MSSQLSERVER16;Initial Catalog=
  </connectionStrings>
  <system.web>
    <compilation debug="true" targetFramework="4.5.1" />
    <httpRuntime targetFramework="4.5.1" />
```

Testowanie aplikacji:

Dodajemy nowy projekt testów o nazwie: **SzkolenieTechniczne3.Infrastucture.Test**

W projekcie dodajemy referencje do AutoMappera.



Add New Project

.NET Framework 4.5.1 | Sort by: Default | Search Installed Templates (Ctrl+E)

Unit Test Project — Visual C#

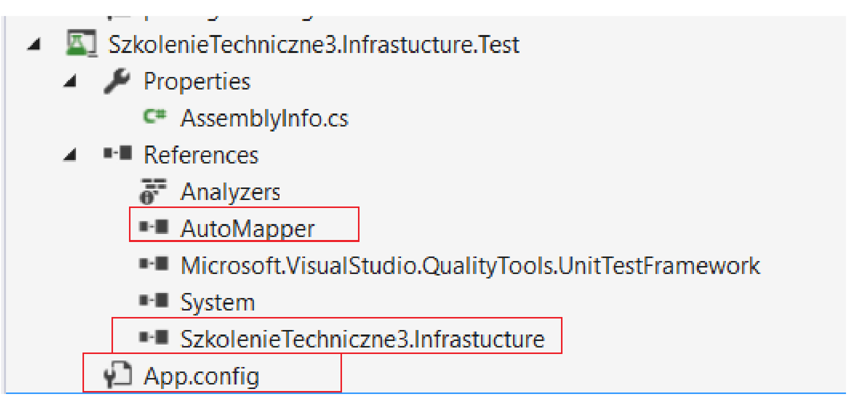
Type: Visual C#

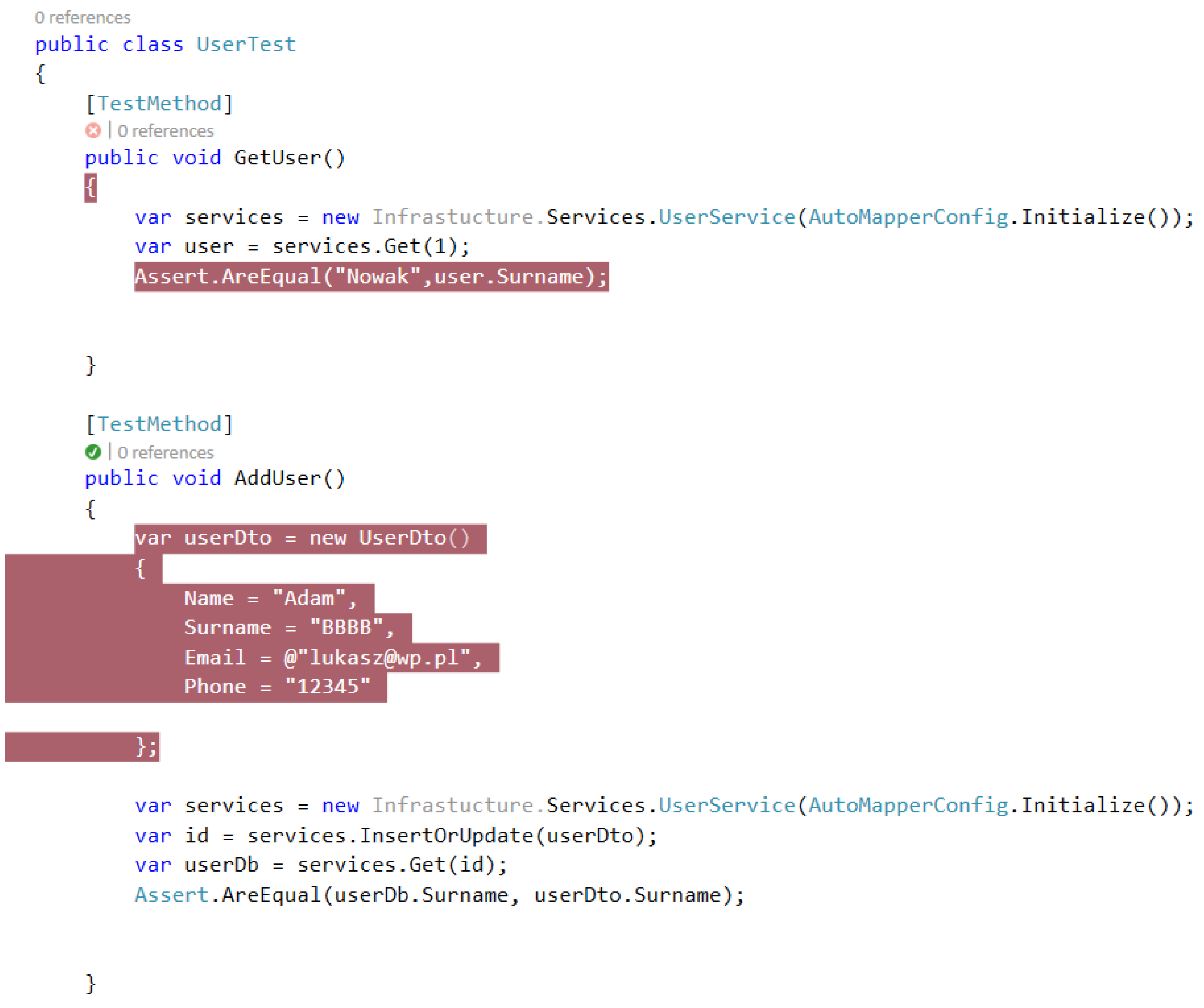
A project that contains unit tests.

Name: UnitTestProject1

Location: C:\Users\lukasz.piechocki\Documents\Visual Studio 2015\Projects\SzkolenieTechniczne3.Core

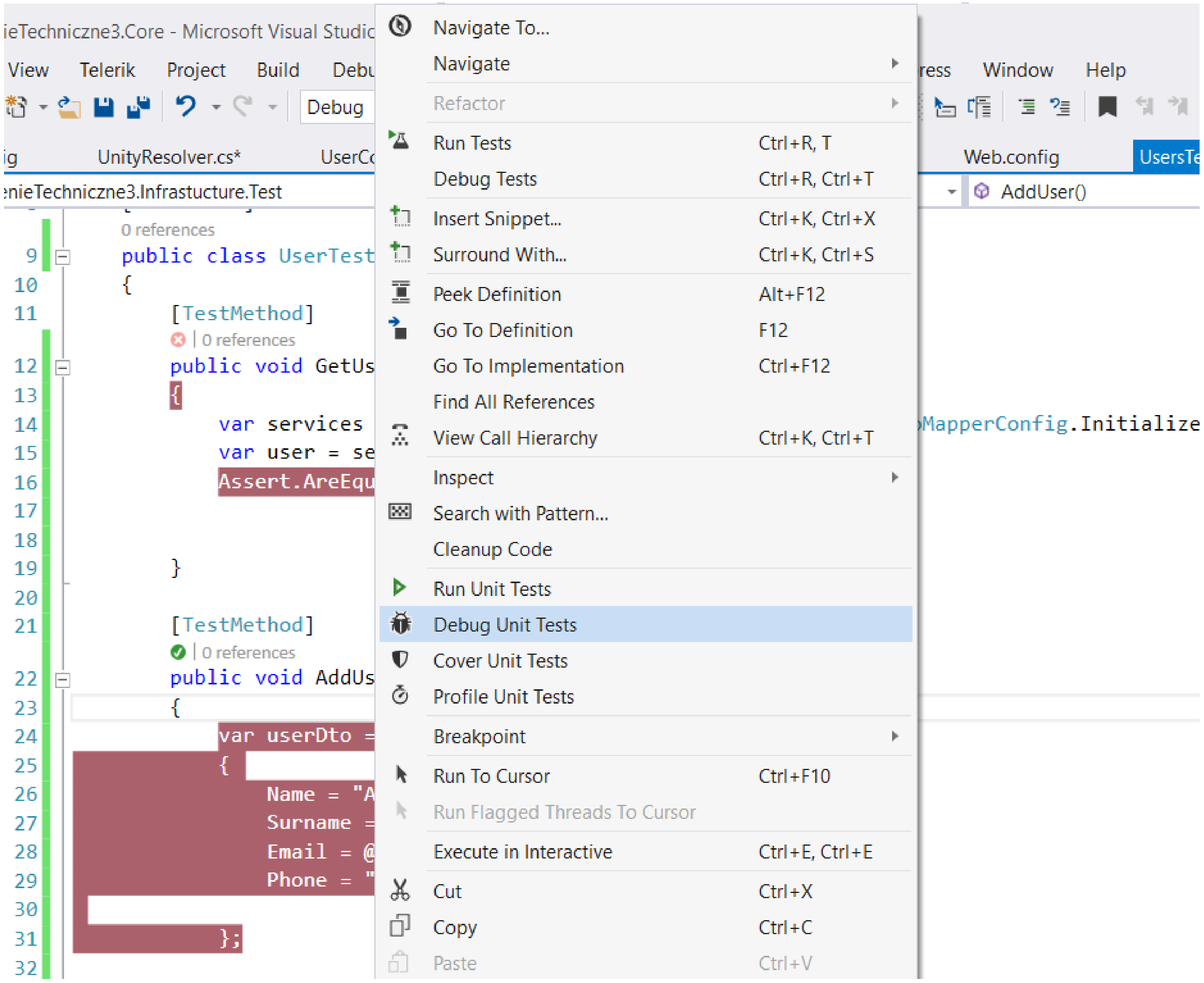
Dodajemy referencje oraz plik konfiguracyjny w raz z połączeniem do bazy danych


```
◢ SzkolenieTechniczne3.Infrastucture.Test
    ◢ Properties
          AssemblyInfo.cs
    ◢ References
          Analyzers
          AutoMapper
          Microsoft.VisualStudio.QualityTools.UnitTestFramework
          System
          SzkolenieTechniczne3.Infrastucture
       App.config
```



```
0 references
public class UserTest
{
    [TestMethod]
    | 0 references
    public void GetUser()
    {
        var services = new Infrastucture.Services.UserService(AutoMapperConfig.Initialize());
        var user = services.Get(1);
        Assert.AreEqual("Nowak",user.Surname);


    }

    [TestMethod]
    | 0 references
    public void AddUser()
    {
        var userDto = new UserDto()
        {
            Name = "Adam",
            Surname = "BBBB",
            Email = @"lukasz@wp.pl",
            Phone = "12345"

        };

        var services = new Infrastucture.Services.UserService(AutoMapperConfig.Initialize());
        var id = services.InsertOrUpdate(userDto);
        var userDb = services.Get(id);
        Assert.AreEqual(userDb.Surname, userDto.Surname);


    }
```

Uruchamiamy test :



```
    0 references
    public class UserTest
    {
        [TestMethod]
        0 references
        public void GetUs
        {
            var services                                 MapperConfig.Initialize
            var user = se
            Assert.AreEqu


        }

        [TestMethod]
        0 references
        public void AddUs
        {
            var userDto =
            {
                Name = "A
                Surname =
                Email = @
                Phone = "

            };
```

URUCHAMIAMY API

Zadanie do zrobienia samodzielnie dodanie obsługi do repozytorium produktów w api.