Delivery Manual Weelo (Million and Up)

Juan David Vasquez Vélez

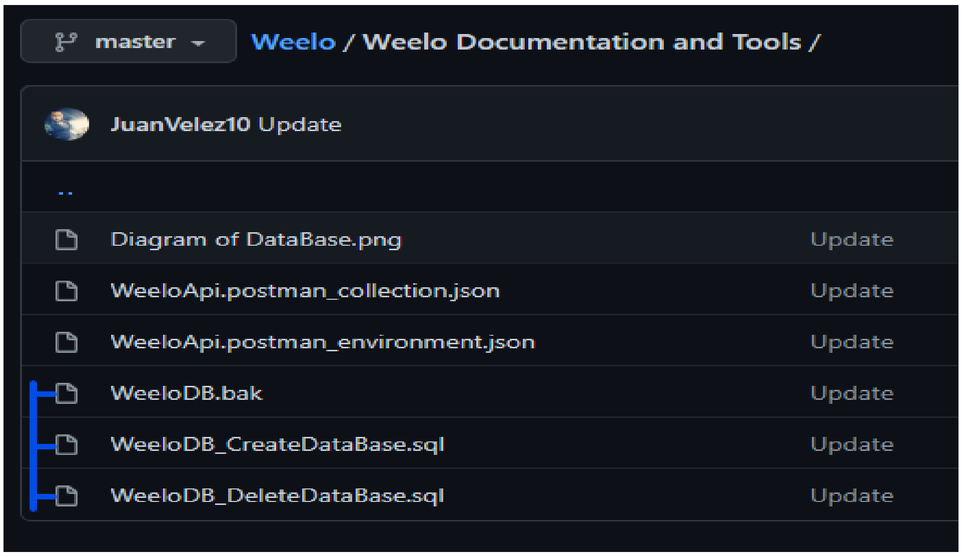
Initial setup

Database:

All the project and documentation and tools such as database backup and postman are in the following github repository for download:

https://github.com/JuanVelez10/Weelo

In the folder Weelo Documentation and Tools we will find a backup of the database, and two initial creation SQL script files.

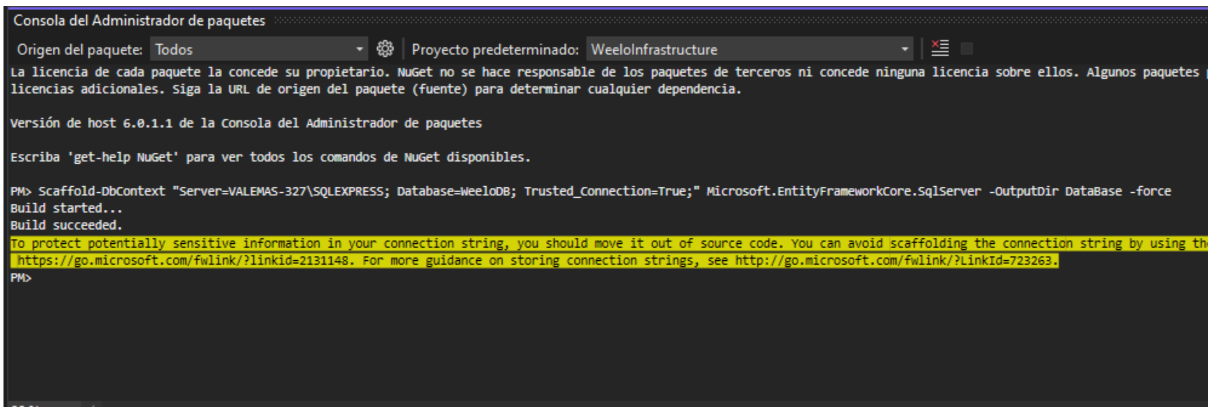


```
master ▾   Weelo / Weelo Documentation and Tools /

JuanVelez10 Update

..
Diagram of DataBase.png                Update
WeeloApi.postman_collection.json       Update
WeeloApi.postman_environment.json      Update
WeeloDB.bak                            Update
WeeloDB_CreateDataBase.sql             Update
WeeloDB_DeleteDataBase.sql             Update
```

When we create the database, the next step is to open the project from visual studio and make the connection to this database with the following command:

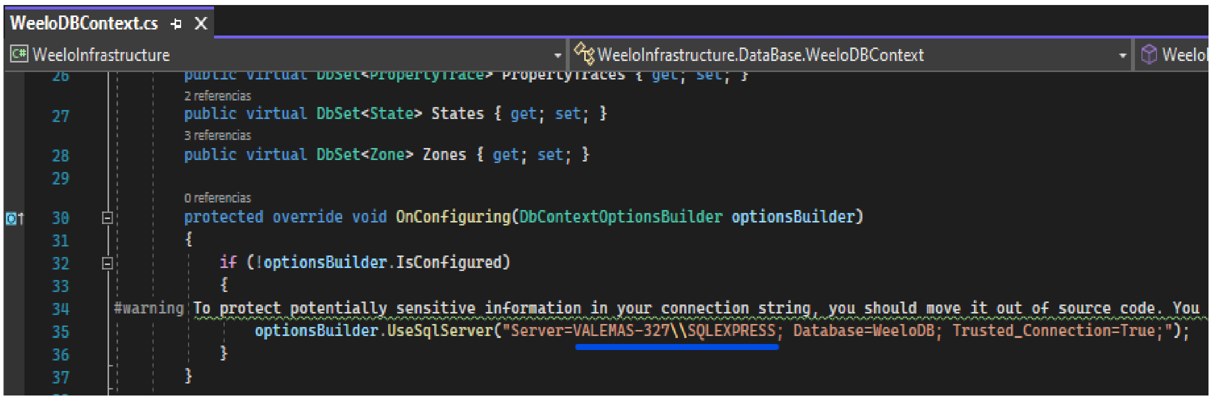
```
Scaffold-DbContext "Server=VALEMAS-327\SQLEXPRESS; Database=WeeloDB;
Trusted_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir DataBase -force
```

Remember to change the server name to that of your machine.



```
Consola del Administrador de paquetes
Origen del paquete: Todos    Proyecto predeterminado: WeeloInfrastructure
La licencia de cada paquete la concede su propietario. NuGet no se hace responsable de los paquetes de terceros ni concede ninguna licencia sobre ellos. Algunos paquetes
licencias adicionales. Siga la URL de origen del paquete (fuente) para determinar cualquier dependencia.

Versión de host 6.0.1.1 de la Consola del Administrador de paquetes

Escriba 'get-help NuGet' para ver todos los comandos de NuGet disponibles.

PM> Scaffold-DbContext "Server=VALEMAS-327\SQLEXPRESS; Database=WeeloDB; Trusted_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir DataBase -force
Build started...
Build succeeded.
To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid scaffolding the connection string by using th
 https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.
PM>
```

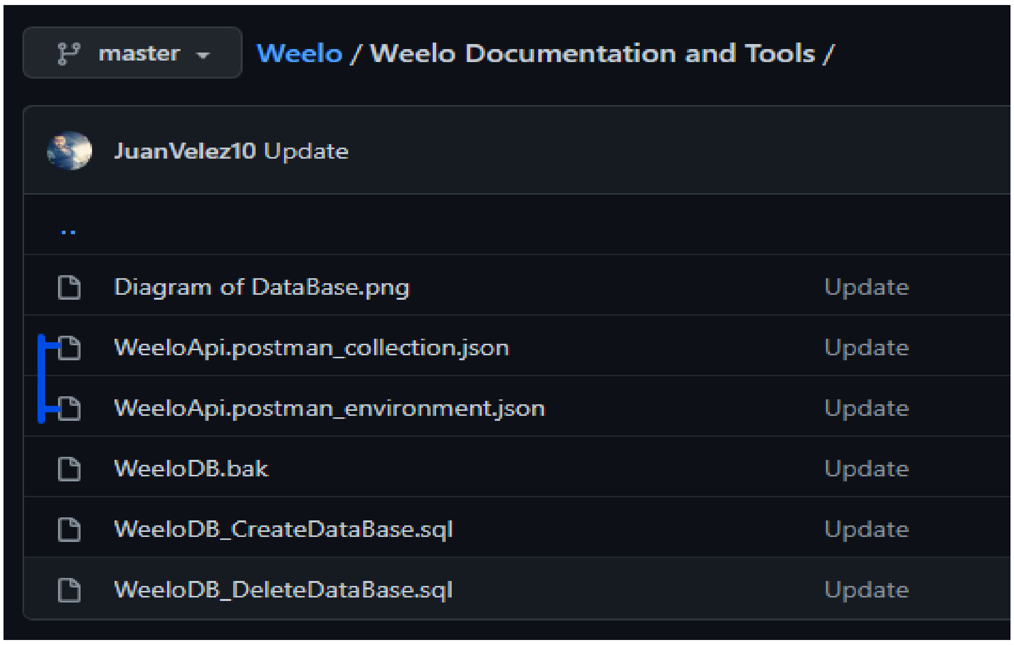
So, you might as well change the server manually from the following class:



```
        public virtual DbSet<PropertyTrace> PropertyTraces { get; set; }
        2 referencias
        public virtual DbSet<State> States { get; set; }
        3 referencias
        public virtual DbSet<Zone> Zones { get; set; }

        0 referencias
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
#warning To protect potentially sensitive information in your connection string, you should move it out of source code. You
                optionsBuilder.UseSqlServer("Server=VALEMAS-327\\SQLEXPRESS; Database=WeeloDB; Trusted_Connection=True;");
            }
        }
```

Postman:

For the configuration of the environment and collection of services in postman we find the files to import in the folder Weelo Documentation and Tools.



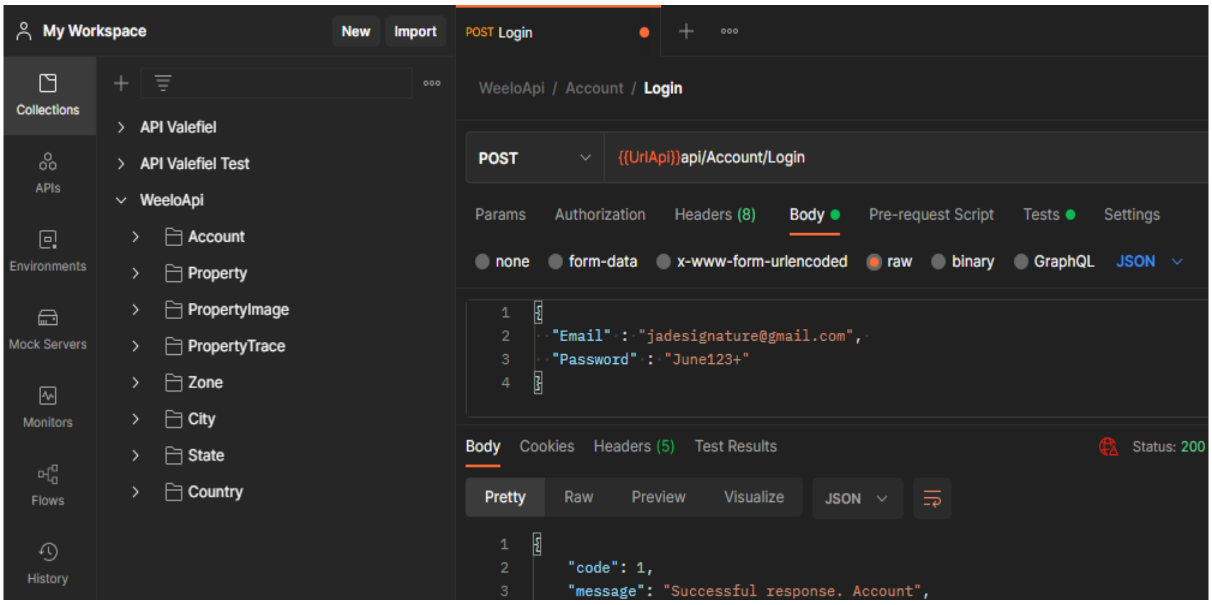
master Weelo / Weelo Documentation and Tools /

JuanVelez10 Update

..

Diagram of DataBase.png	Update
WeeloApi.postman_collection.json	Update
WeeloApi.postman_environment.json	Update
WeeloDB.bak	Update
WeeloDB_CreateDataBase.sql	Update
WeeloDB_DeleteDataBase.sql	Update

Import Collection

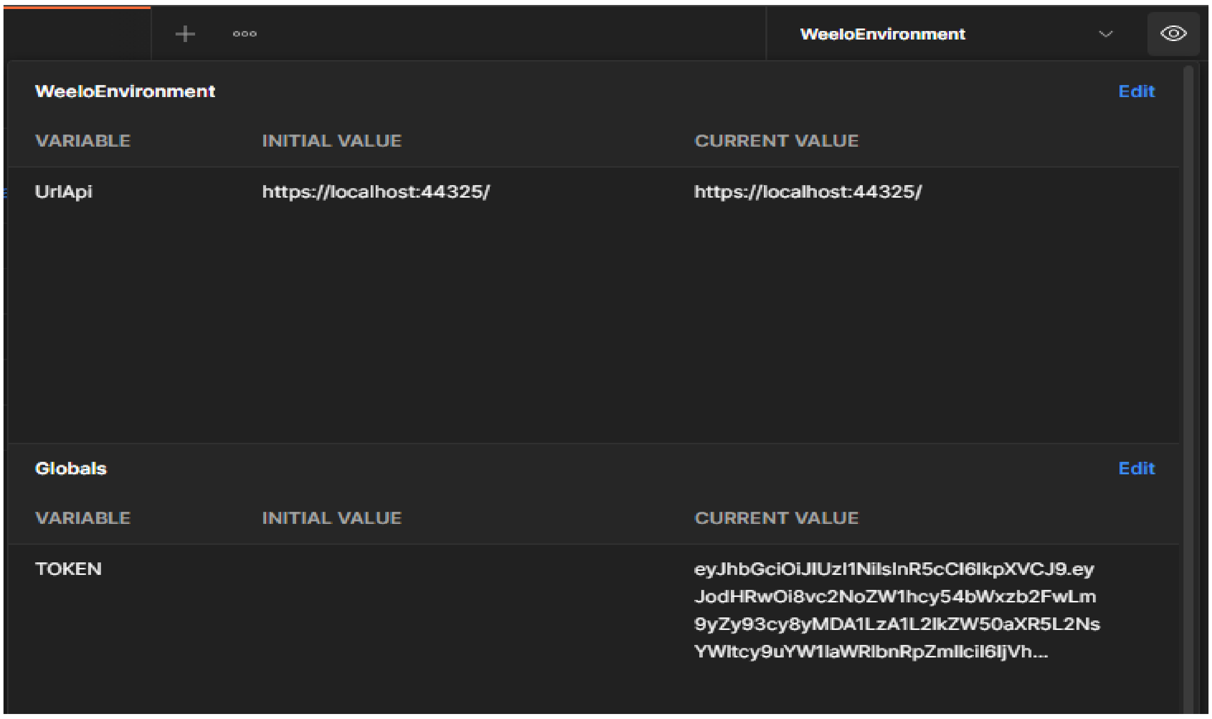


POST {{UrlApi}}api/Account/Login

```
{
    "Email" : "jadesignature@gmail.com",
    "Password" : "June123+"
}
```

```
{
    "code": 1,
    "message": "Successful response. Account",
```

Import environment



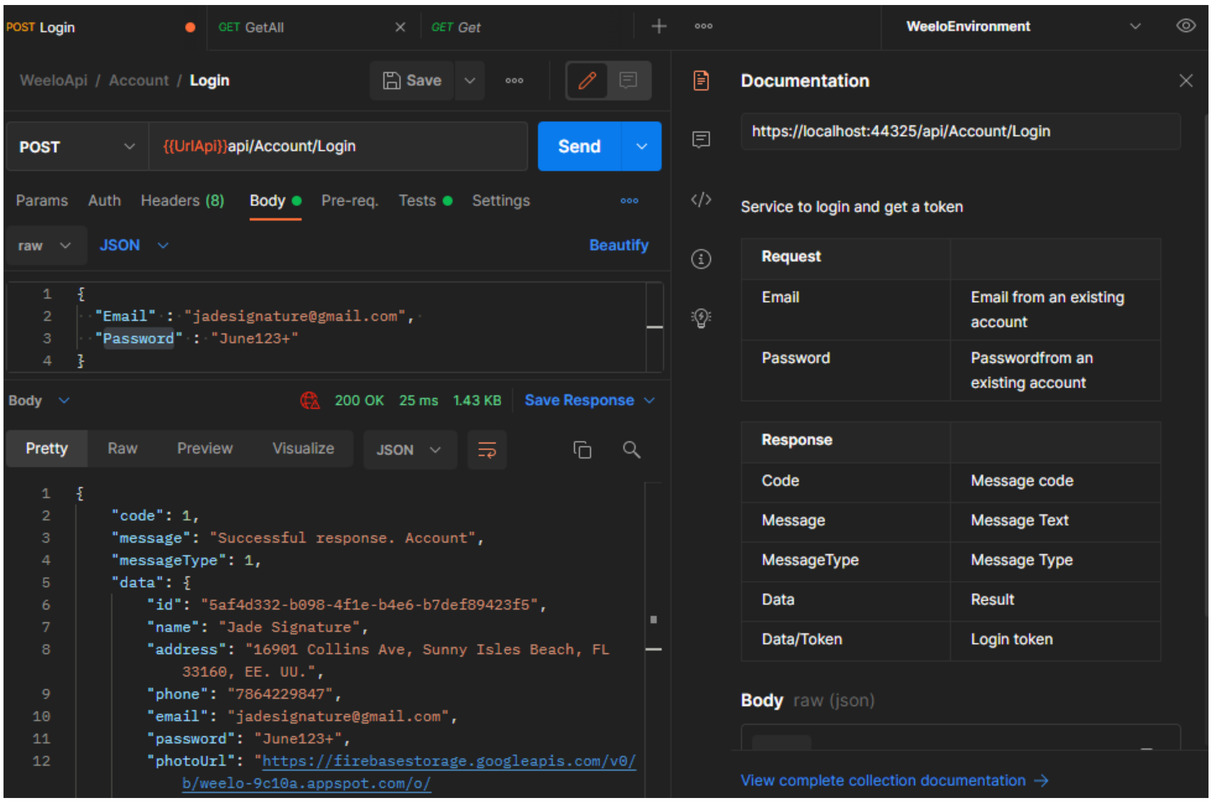
WeeloEnvironment

VARIABLE	INITIAL VALUE	CURRENT VALUE
UrlApi	https://localhost:44325/	https://localhost:44325/

Globals

VARIABLE	INITIAL VALUE	CURRENT VALUE
TOKEN		eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6IjVh...

Note. The most important services are documented in postman



POST {{UrlApi}}api/Account/Login

```
{
  "Email" : "jadesignature@gmail.com",
  "Password" : "June123+"
}
```

200 OK 25 ms 1.43 KB

```
{
    "code": 1,
    "message": "Successful response. Account",
    "messageType": 1,
    "data": {
        "id": "5af4d332-b098-4f1e-b4e6-b7def89423f5",
        "name": "Jade Signature",
        "address": "16901 Collins Ave, Sunny Isles Beach, FL
            33160, EE. UU.",
        "phone": "7864229847",
        "email": "jadesignature@gmail.com",
        "password": "June123+",
        "photoUrl": "https://firebasestorage.googleapis.com/v0/
            b/weelo-9c10a.appspot.com/o/
```

https://localhost:44325/api/Account/Login

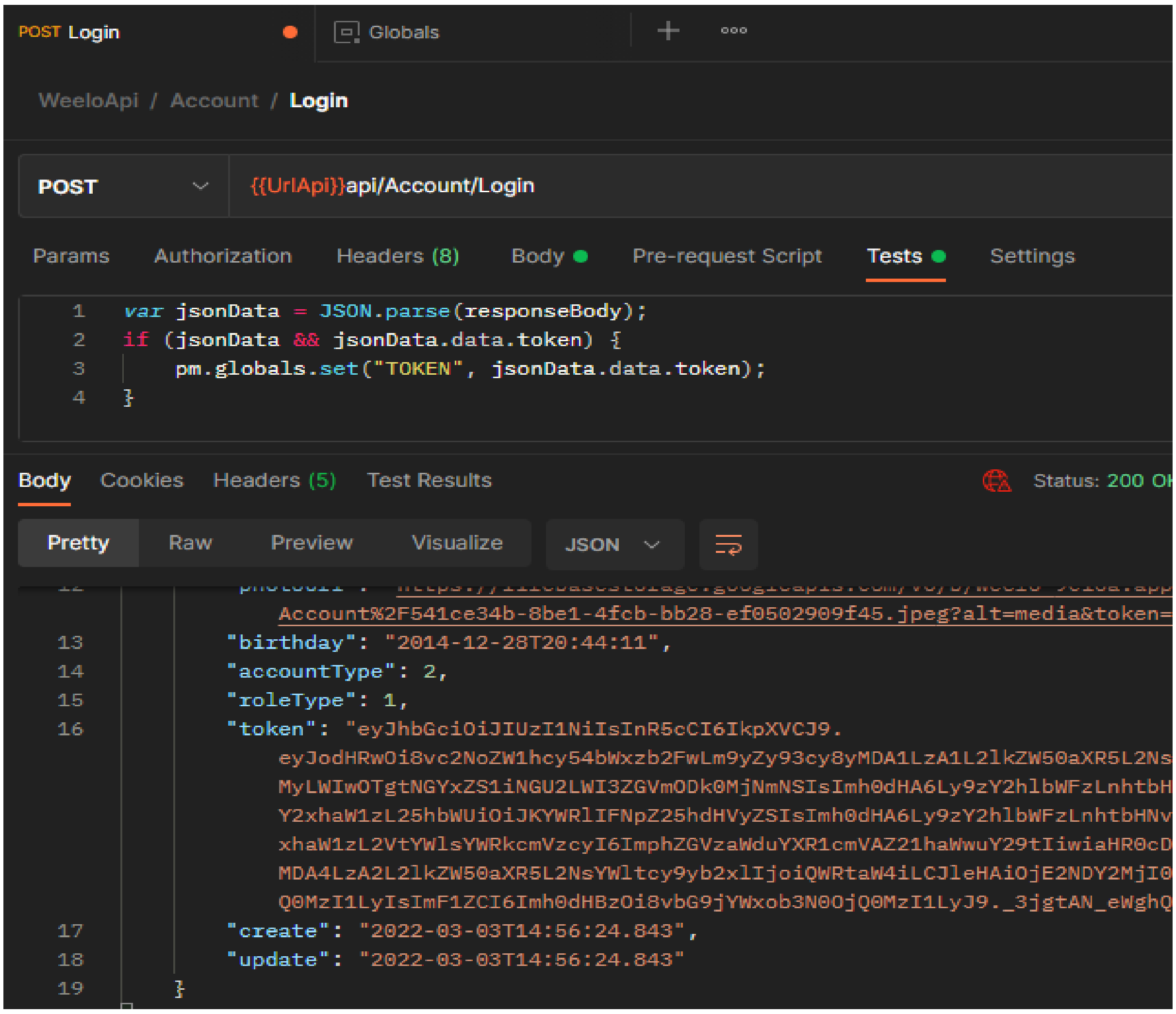
Service to login and get a token

Request	
Email	Email from an existing account
Password	Passwordfrom an existing account

Response	
Code	Message code
Message	Message Text
MessageType	Message Type
Data	Result
Data/Token	Login token

Body raw (json)

Note. Keep in mind that you must be logged in first to be able to use the other services, so you must first use the Login service to be able to generate an authentication token and then use it in the other services.



POST Login | Globals

WeeloApi / Account / **Login**

POST {{UrlApi}}api/Account/Login

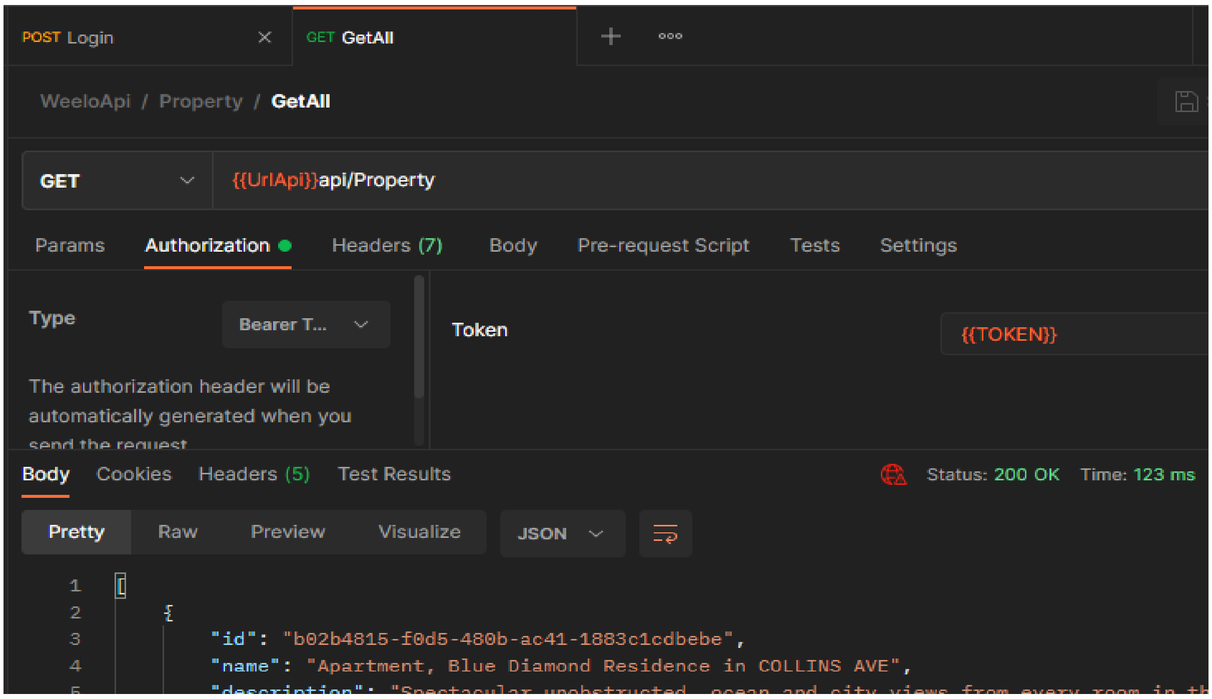
Params | Authorization | Headers (8) | Body ● | Pre-request Script | **Tests ●** | Settings

```
var jsonData = JSON.parse(responseBody);
if (jsonData && jsonData.data.token) {
    pm.globals.set("TOKEN", jsonData.data.token);
}
```

Body | Cookies | Headers (5) | Test Results — Status: 200 OK

Pretty | Raw | Preview | Visualize | JSON

```
            Account%2F541ce34b-8be1-4fcb-bb28-ef0502909f45.jpeg?alt=media&token=
        "birthday": "2014-12-28T20:44:11",
        "accountType": 2,
        "roleType": 1,
        "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
            eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2Ns
            MyLWIwOTgtNGYxZS1iNGU2LWI3ZGVmODk0MjNmNSIsImh0dHA6Ly9zY2hlbWFzLnhtbH
            Y2xhaW1zL25hbWUiOiJKYWRlIFNpZ25hdHVyZSIsImh0dHA6Ly9zY2hlbWFzLnhtbHNv
            xhaW1zL2VtYWlsYWRkcmVzcyI6ImphZGVzaWduYXR1cmVAZ21haWwuY29tIiwiaHR0cD
            MDA4LzA2L2lkZW50aXR5L2NsYWltcy9yb2xlIjoiQWRtaW4iLCJleHAiOjE2NDY2MjI0
            Q0MzI1LyIsImF1ZCI6Imh0dHBzOi8vbG9jYWxob3N0OjQ0MzI1LyJ9._3jgtAN_eWghQ
        "create": "2022-03-03T14:56:24.843",
        "update": "2022-03-03T14:56:24.843"
    }
```



POST Login | GET **GetAll**

WeeloApi / Property / **GetAll**

GET {{UrlApi}}api/Property

Params | **Authorization ●** | Headers (7) | Body | Pre-request Script | Tests | Settings

Type: Bearer T...

Token: {{TOKEN}}

The authorization header will be automatically generated when you send the request

Body | Cookies | Headers (5) | Test Results — Status: 200 OK Time: 123 ms

Pretty | Raw | Preview | Visualize | JSON

```
[
    {
        "id": "b02b4815-f0d5-480b-ac41-1883c1cdbebe",
        "name": "Apartment, Blue Diamond Residence in COLLINS AVE",
```

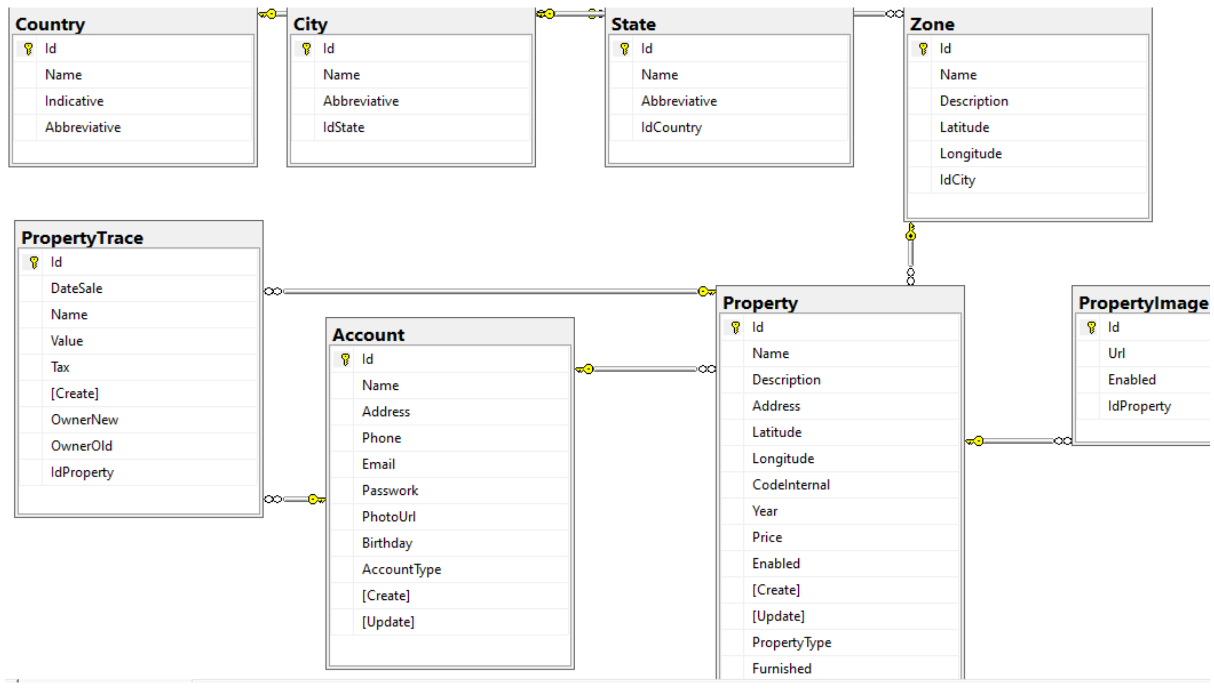
Points to evaluate

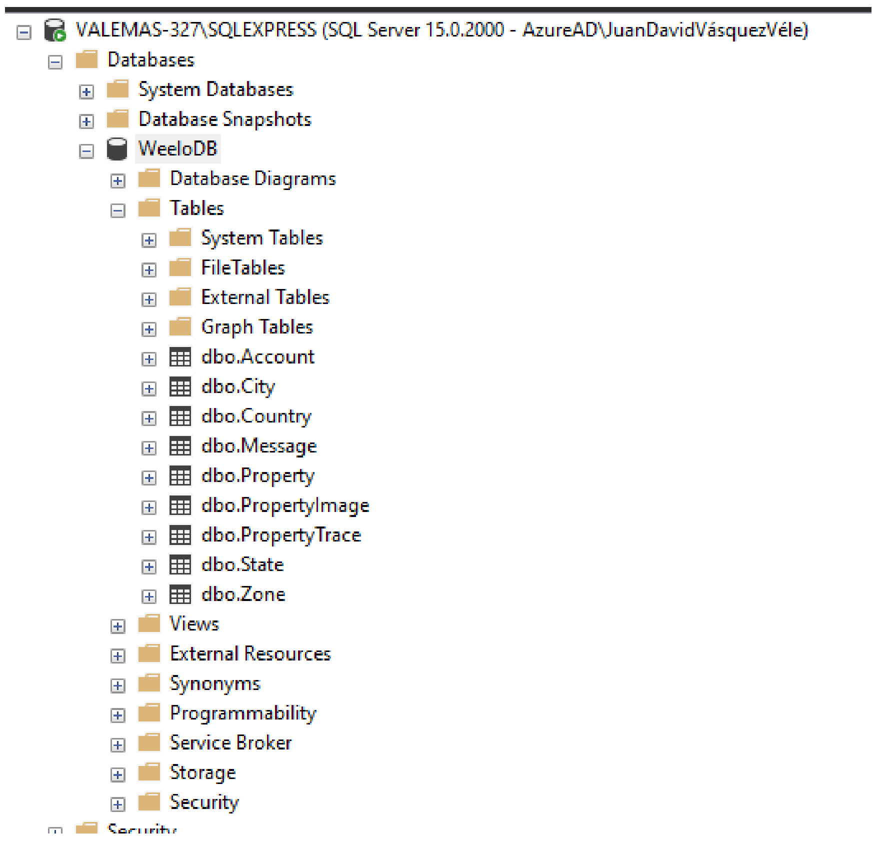
Required manage:

All the tools required for the development of this API are used (.NET 5,SQL Server,C#,nUnit)

SQL Server

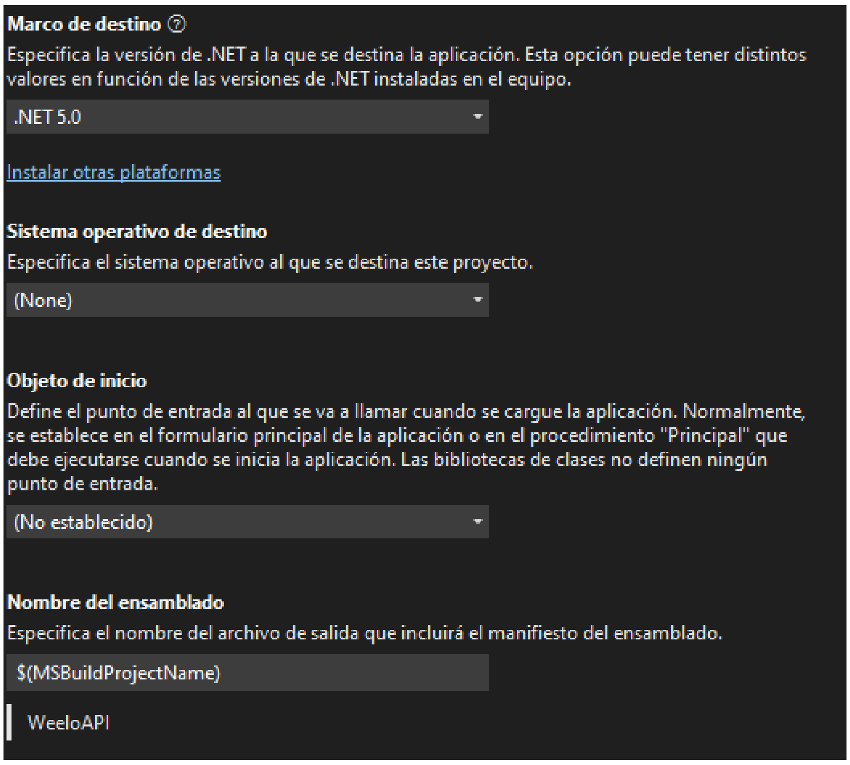
Diagram of DataBase: [Diagram of DataBase.png]()





- VALEMAS-327\SQLEXPRESS (SQL Server 15.0.2000 - AzureAD\JuanDavidVásquezVéle)
  - Databases
    - System Databases
    - Database Snapshots
    - WeeloDB
      - Database Diagrams
      - Tables
        - System Tables
        - FileTables
        - External Tables
        - Graph Tables
        - dbo.Account
        - dbo.City
        - dbo.Country
        - dbo.Message
        - dbo.Property
        - dbo.PropertyImage
        - dbo.PropertyTrace
        - dbo.State
        - dbo.Zone
      - Views
      - External Resources
      - Synonyms
      - Programmability
      - Service Broker
      - Storage
      - Security
  - Security

.NET 5 and C#



Marco de destino ⓘ

Especifica la versión de .NET a la que se destina la aplicación. Esta opción puede tener distintos valores en función de las versiones de .NET instaladas en el equipo.

.NET 5.0

Instalar otras plataformas

Sistema operativo de destino

Especifica el sistema operativo al que se destina este proyecto.

(None)

Objeto de inicio

Define el punto de entrada al que se va a llamar cuando se cargue la aplicación. Normalmente, se establece en el formulario principal de la aplicación o en el procedimiento "Principal" que debe ejecutarse cuando se inicia la aplicación. Las bibliotecas de clases no definen ningún punto de entrada.

(No establecido)

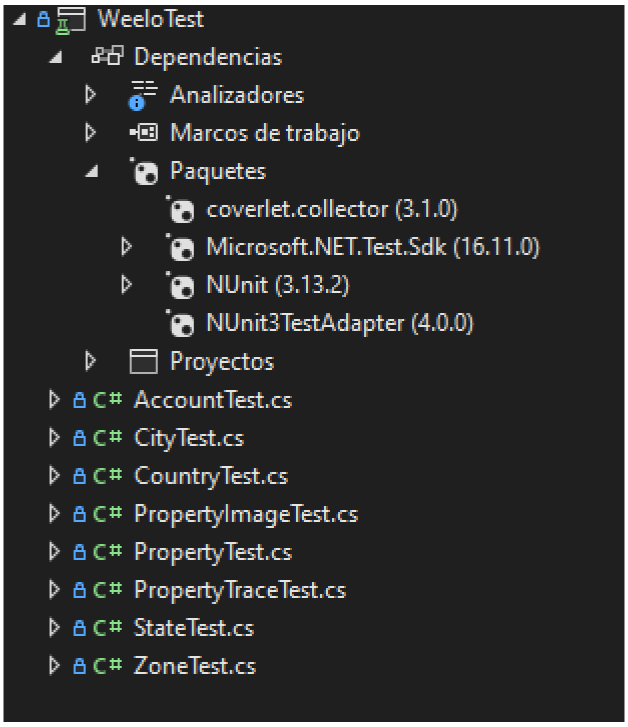
Nombre del ensamblado

Especifica el nombre del archivo de salida que incluirá el manifiesto del ensamblado.

$(MSBuildProjectName)

WeeloAPI

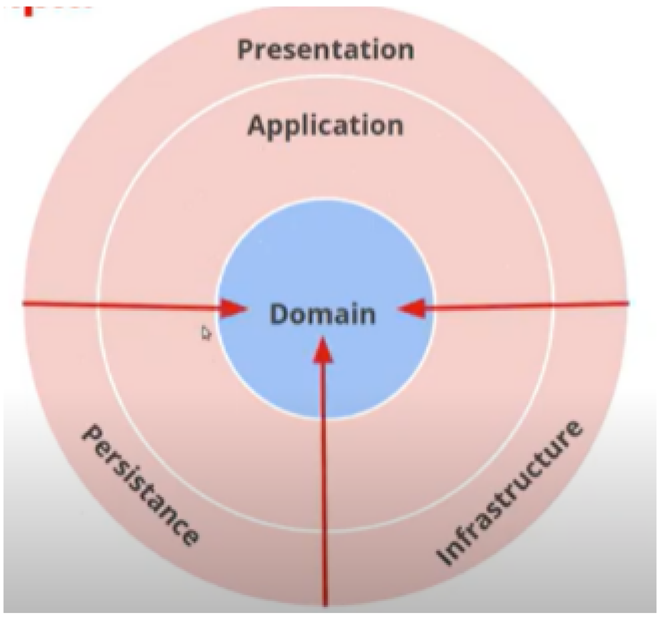
nUnit

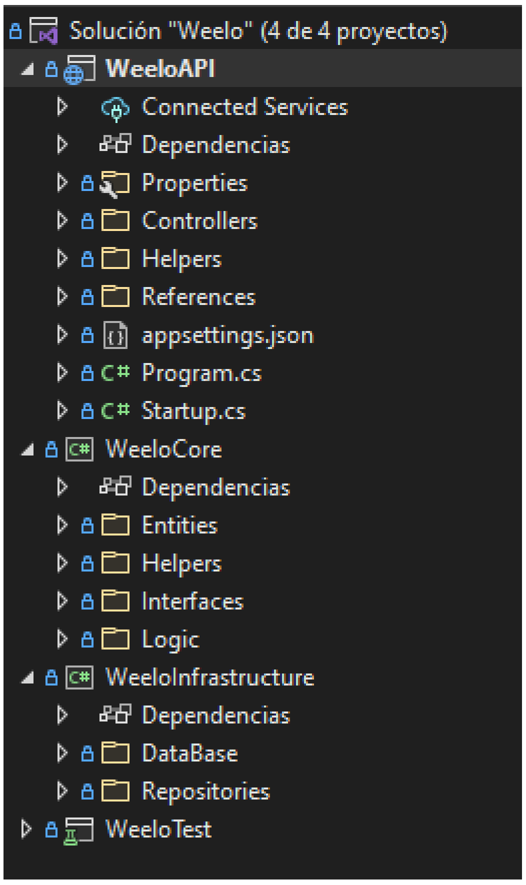


```
◢ WeeloTest
    ◢ Dependencias
        ▷ Analizadores
        ▷ Marcos de trabajo
        ◢ Paquetes
              coverlet.collector (3.1.0)
            ▷ Microsoft.NET.Test.Sdk (16.11.0)
            ▷ NUnit (3.13.2)
              NUnit3TestAdapter (4.0.0)
        ▷ Proyectos
    ▷ C# AccountTest.cs
    ▷ C# CityTest.cs
    ▷ C# CountryTest.cs
    ▷ C# PropertyImageTest.cs
    ▷ C# PropertyTest.cs
    ▷ C# PropertyTraceTest.cs
    ▷ C# StateTest.cs
    ▷ C# ZoneTest.cs
```

Architecture:

Clean architecture management: where we have a **WeeloApi** solution where all the services are and this in turn communicates with a domain layer called **WeeloCore** where all the business logic is processed and the latter communicates with an infrastructure layer called **WeeloInfrastructure** where the latter has the connection to the database and persistence.

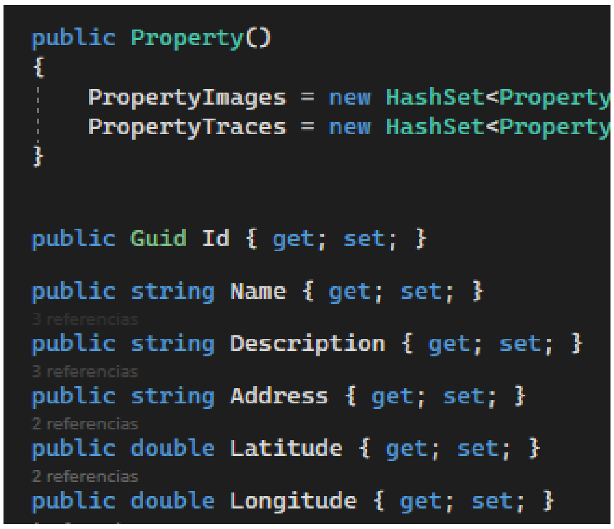




```
Solución "Weelo" (4 de 4 proyectos)
  WeeloAPI
    Connected Services
    Dependencias
    Properties
    Controllers
    Helpers
    References
    appsettings.json
    Program.cs
    Startup.cs
  WeeloCore
    Dependencias
    Entities
    Helpers
    Interfaces
    Logic
  WeeloInfrastructure
    Dependencias
    DataBase
    Repositories
  WeeloTest
```

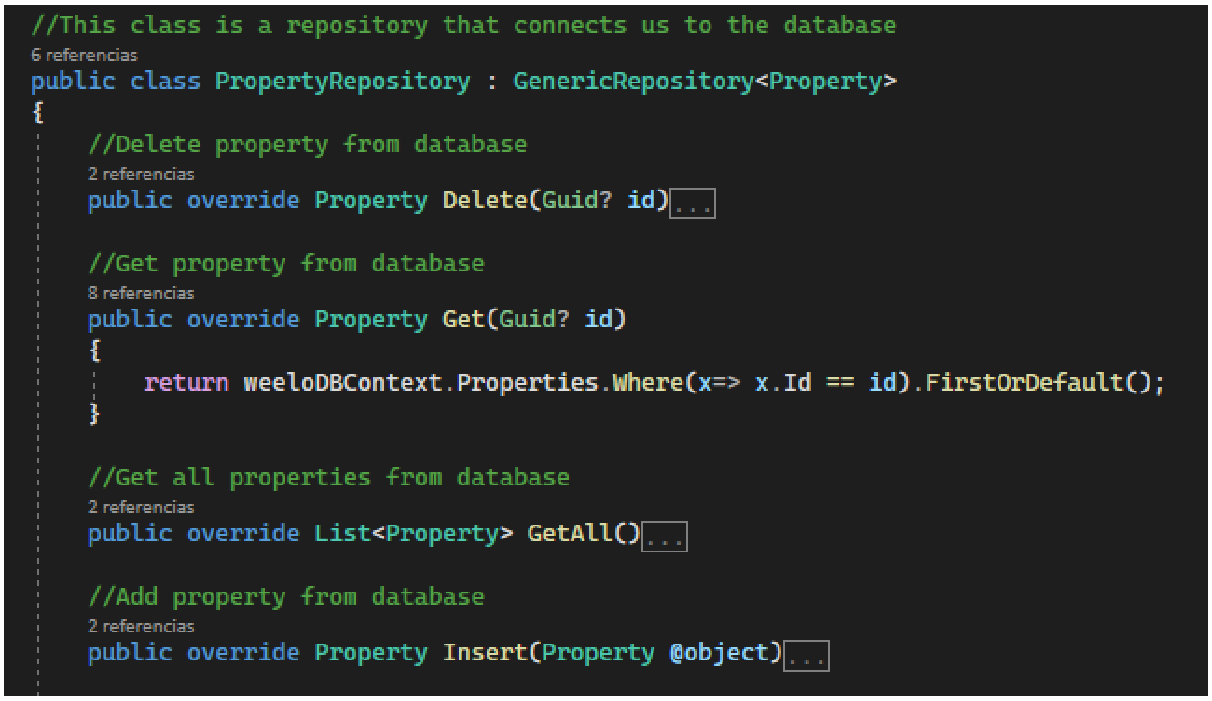
Structure:
The entire structure is object-oriented and its 4 main pillars, it was also based on the SOLID principles where the fundamental factor is to simplify everything with more classes, methods with only functions we have much better, they were implemented design patterns at the structural level as well.

4 principles of object-oriented programming
Encapsulación: get and set



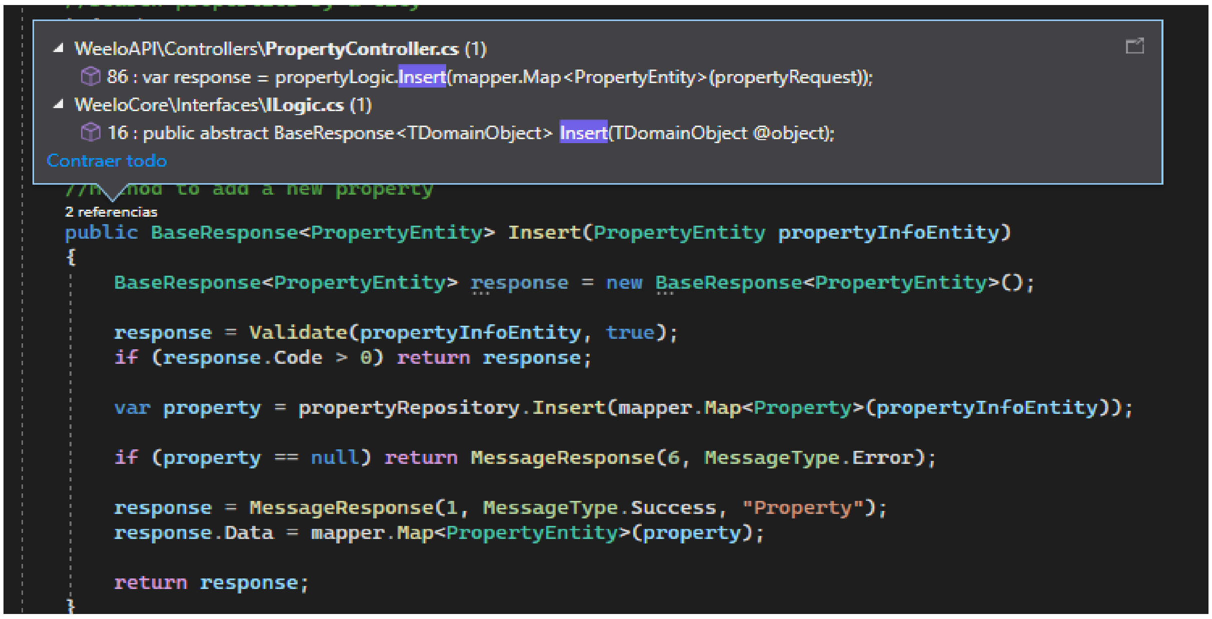
```
public Property()
{
    PropertyImages = new HashSet<Property
    PropertyTraces = new HashSet<Property
}


public Guid Id { get; set; }

public string Name { get; set; }
3 referencias
public string Description { get; set; }
3 referencias
public string Address { get; set; }
2 referencias
public double Latitude { get; set; }
2 referencias
public double Longitude { get; set; }
```

Abstraction: repository database connection
Inheritance : from the abstract class Generic Repository<Property>

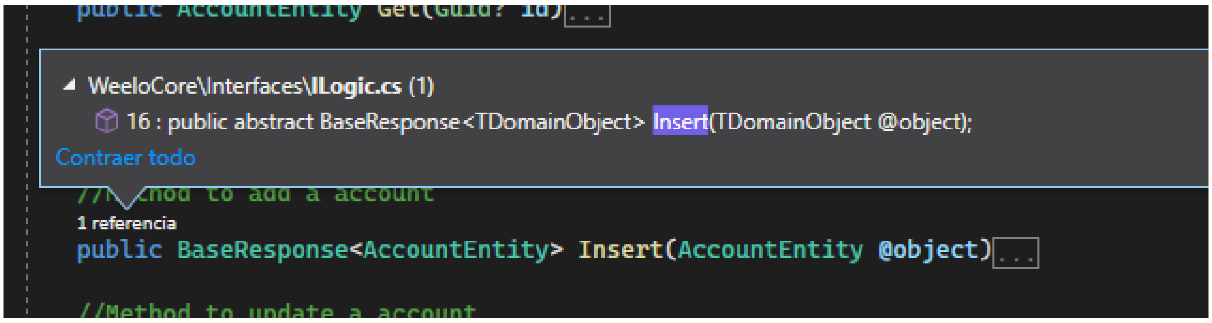


```
//This class is a repository that connects us to the database
6 referencias
public class PropertyRepository : GenericRepository<Property>
{
    //Delete property from database
    2 referencias
    public override Property Delete(Guid? id)...

    //Get property from database
    8 referencias
    public override Property Get(Guid? id)
    {
        return weeloDBContext.Properties.Where(x=> x.Id == id).FirstOrDefault();
    }

    //Get all properties from database
    2 referencias
    public override List<Property> GetAll()...

    //Add property from database
    2 referencias
    public override Property Insert(Property @object)...
```

Polimorfismo:use same functionality but with different objects



```
◢ WeeloAPI\Controllers\PropertyController.cs (1)
    86 : var response = propertyLogic.Insert(mapper.Map<PropertyEntity>(propertyRequest));
◢ WeeloCore\Interfaces\ILogic.cs (1)
    16 : public abstract BaseResponse<TDomainObject> Insert(TDomainObject @object);
Contraer todo
//Method to add a new property
2 referencias
public BaseResponse<PropertyEntity> Insert(PropertyEntity propertyInfoEntity)
{
    BaseResponse<PropertyEntity> response = new BaseResponse<PropertyEntity>();

    response = Validate(propertyInfoEntity, true);
    if (response.Code > 0) return response;

    var property = propertyRepository.Insert(mapper.Map<Property>(propertyInfoEntity));

    if (property == null) return MessageResponse(6, MessageType.Error);

    response = MessageResponse(1, MessageType.Success, "Property");
    response.Data = mapper.Map<PropertyEntity>(property);

    return response;
}
```

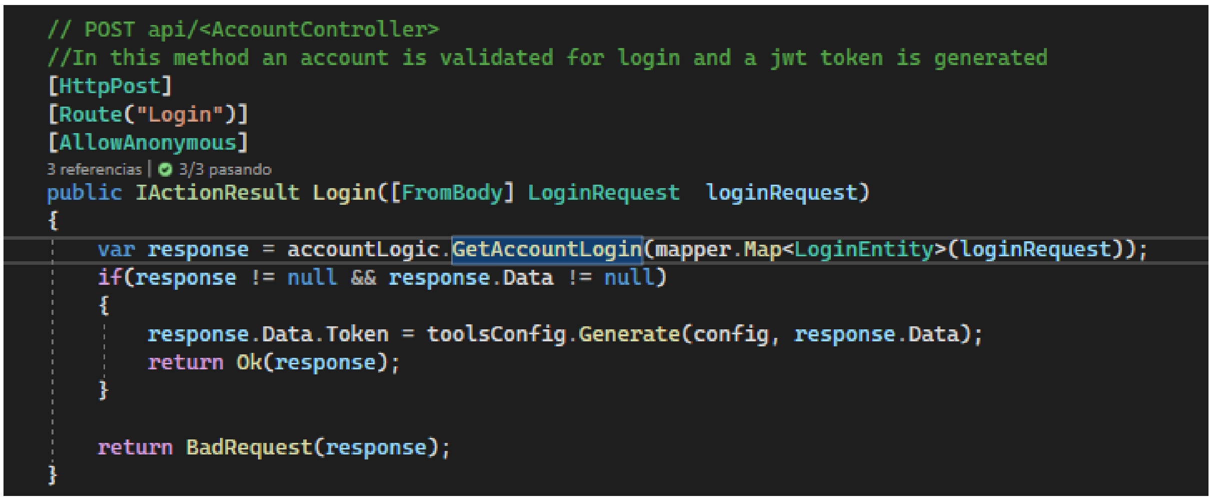


```
public AccountEntity Get(Guid? id)...

◢ WeeloCore\Interfaces\ILogic.cs (1)
    16 : public abstract BaseResponse<TDomainObject> Insert(TDomainObject @object);
Contraer todo
//Method to add a account
1 referencia
public BaseResponse<AccountEntity> Insert(AccountEntity @object)...

//Method to update a account
```

The SOLID principles

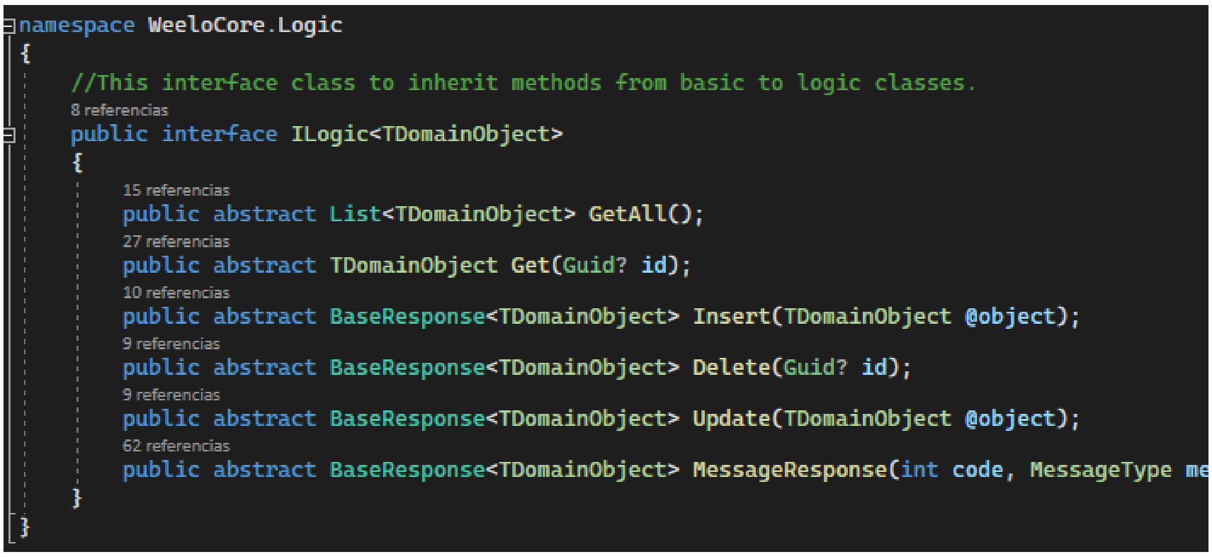
S – Single Responsibility Principle (SRP)

In this method we see how we call a class to obtain the verification of an account and another helpers class to generate the authentication token, thus guaranteeing sole responsibility.



```
// POST api/<AccountController>
//In this method an account is validated for login and a jwt token is generated
[HttpPost]
[Route("Login")]
[AllowAnonymous]
3 referencias | 3/3 pasando
public IActionResult Login([FromBody] LoginRequest  loginRequest)
{
    var response = accountLogic.GetAccountLogin(mapper.Map<LoginEntity>(loginRequest));
    if(response != null && response.Data != null)
    {
        response.Data.Token = toolsConfig.Generate(config, response.Data);
        return Ok(response);
    }

    return BadRequest(response);
}
```

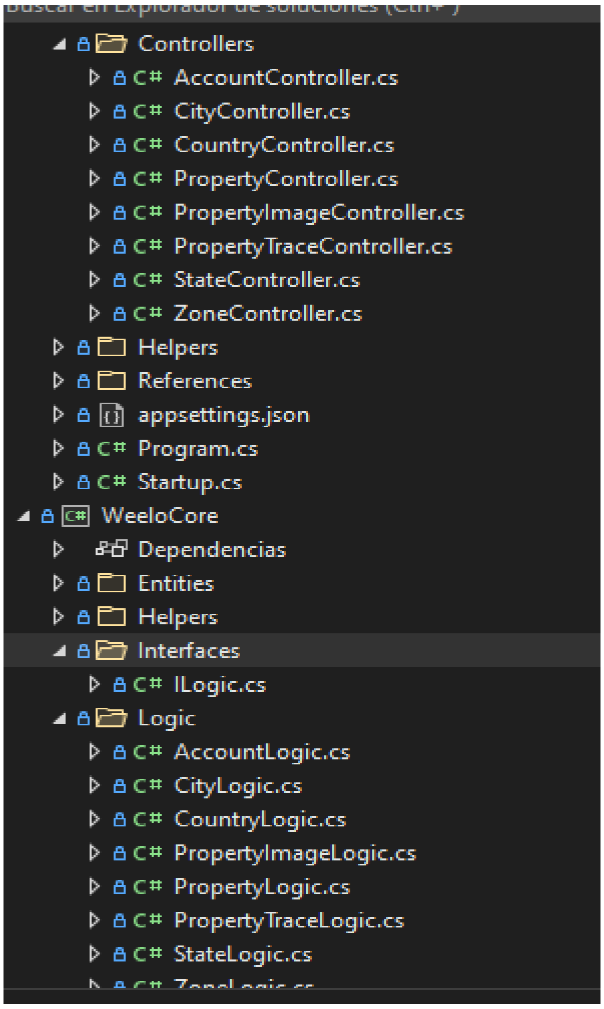
O – Open/Closed Principle (OCP)
We have an interface class to ensure that the code is not easily changed.



```
namespace WeeloCore.Logic
{
    //This interface class to inherit methods from basic to logic classes.
    8 referencias
    public interface ILogic<TDomainObject>
    {
        15 referencias
        public abstract List<TDomainObject> GetAll();
        27 referencias
        public abstract TDomainObject Get(Guid? id);
        10 referencias
        public abstract BaseResponse<TDomainObject> Insert(TDomainObject @object);
        9 referencias
        public abstract BaseResponse<TDomainObject> Delete(Guid? id);
        9 referencias
        public abstract BaseResponse<TDomainObject> Update(TDomainObject @object);
        62 referencias
        public abstract BaseResponse<TDomainObject> MessageResponse(int code, MessageType me
    }
}
```

L – Liskov Substitution Principle (LSP)
It has not been implemented, but it can be improved by implementing separate interfaces for each Ilogic functionality, so that all the methods of this interface do not have to be inherited.

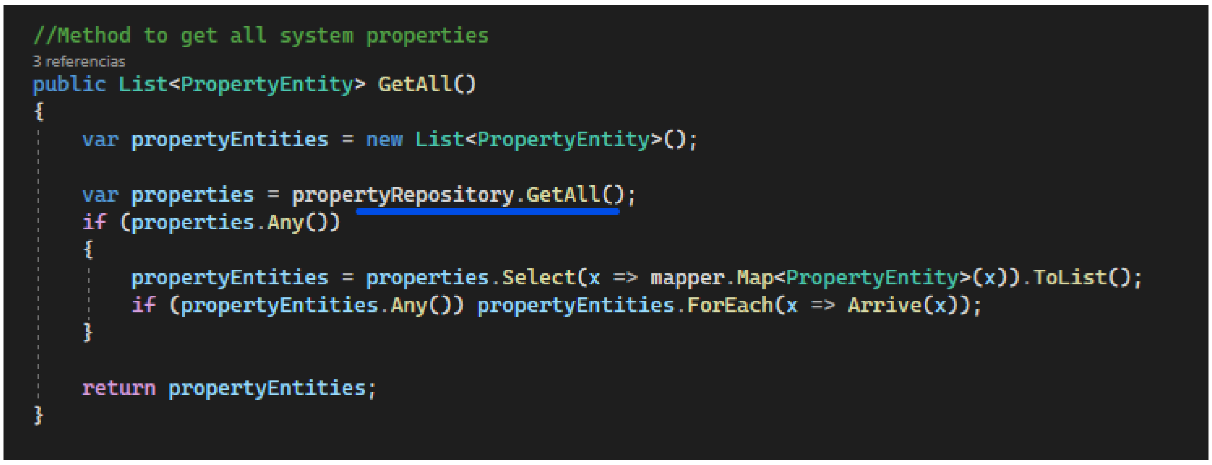
I – Interface Segregation Principle (ISP)
Small classes with specific roles



```
Controllers
    AccountController.cs
    CityController.cs
    CountryController.cs
    PropertyController.cs
    PropertyImageController.cs
    PropertyTraceController.cs
    StateController.cs
    ZoneController.cs
Helpers
References
appsettings.json
Program.cs
Startup.cs
WeeloCore
    Dependencias
    Entities
    Helpers
    Interfaces
        ILogic.cs
    Logic
        AccountLogic.cs
        CityLogic.cs
        CountryLogic.cs
        PropertyImageLogic.cs
        PropertyLogic.cs
        PropertyTraceLogic.cs
        StateLogic.cs
```

D – Dependency Inversion Principle (DIP)

At any time I can change the connection method to the database but the logic code should not be changed

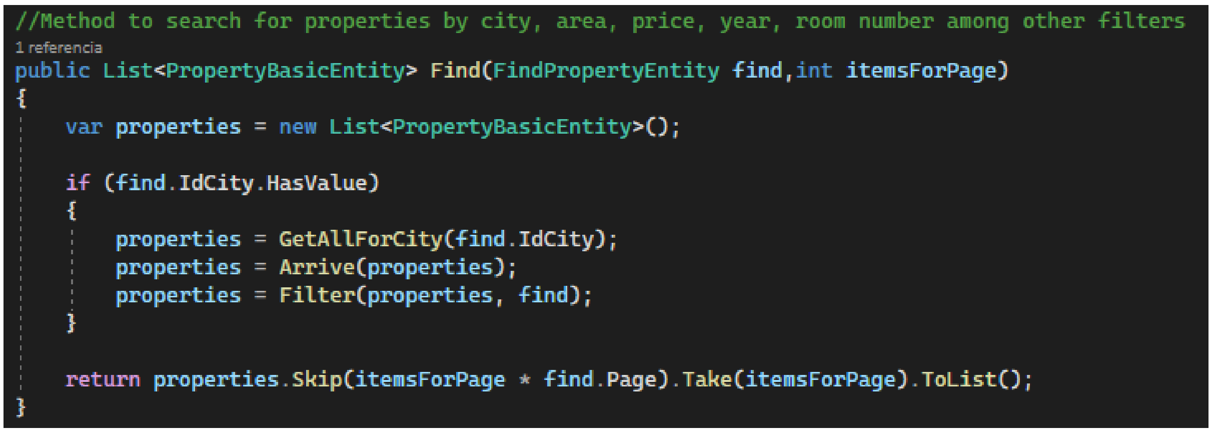


```csharp
//Method to get all system properties
3 referencias
public List<PropertyEntity> GetAll()
{
    var propertyEntities = new List<PropertyEntity>();

    var properties = propertyRepository.GetAll();
    if (properties.Any())
    {
        propertyEntities = properties.Select(x => mapper.Map<PropertyEntity>(x)).ToList();
        if (propertyEntities.Any()) propertyEntities.ForEach(x => Arrive(x));
    }

    return propertyEntities;
}
```

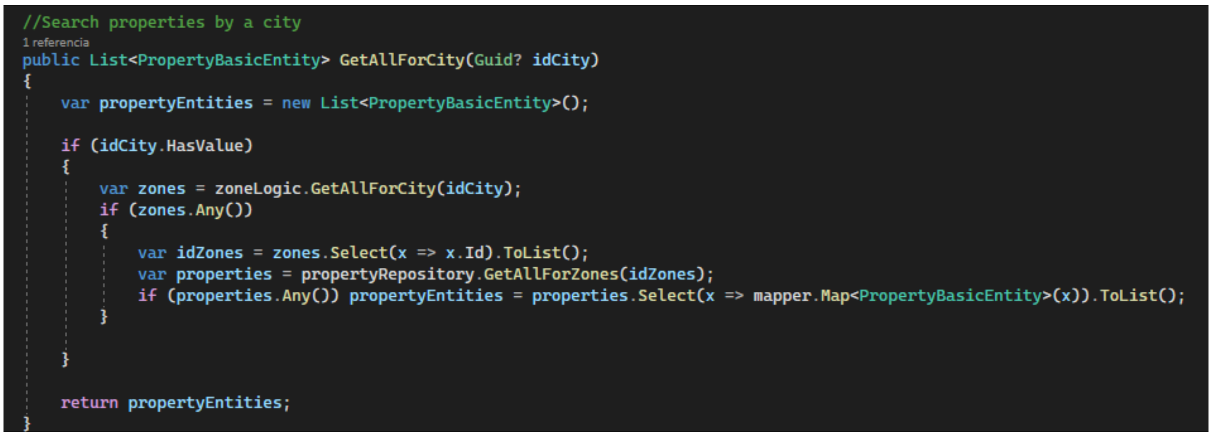
Design patterns

Factory

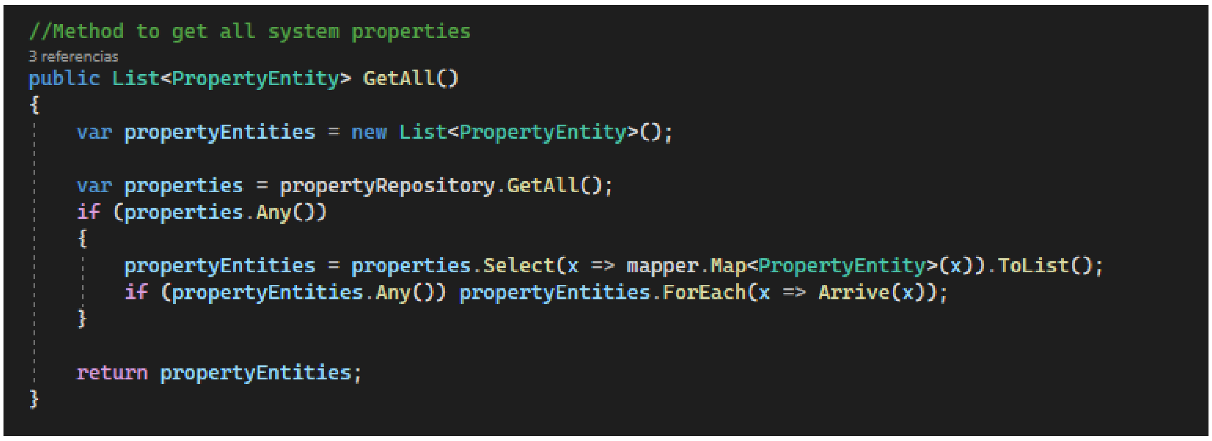
One method brings me a list of properties with certain filters and the other brings me all the properties of a city and finally another method that brings me all the properties.



```csharp
//Method to search for properties by city, area, price, year, room number among other filters
1 referencia
public List<PropertyBasicEntity> Find(FindPropertyEntity find,int itemsForPage)
{
    var properties = new List<PropertyBasicEntity>();

    if (find.IdCity.HasValue)
    {
        properties = GetAllForCity(find.IdCity);
        properties = Arrive(properties);
        properties = Filter(properties, find);
    }

    return properties.Skip(itemsForPage * find.Page).Take(itemsForPage).ToList();
}
```



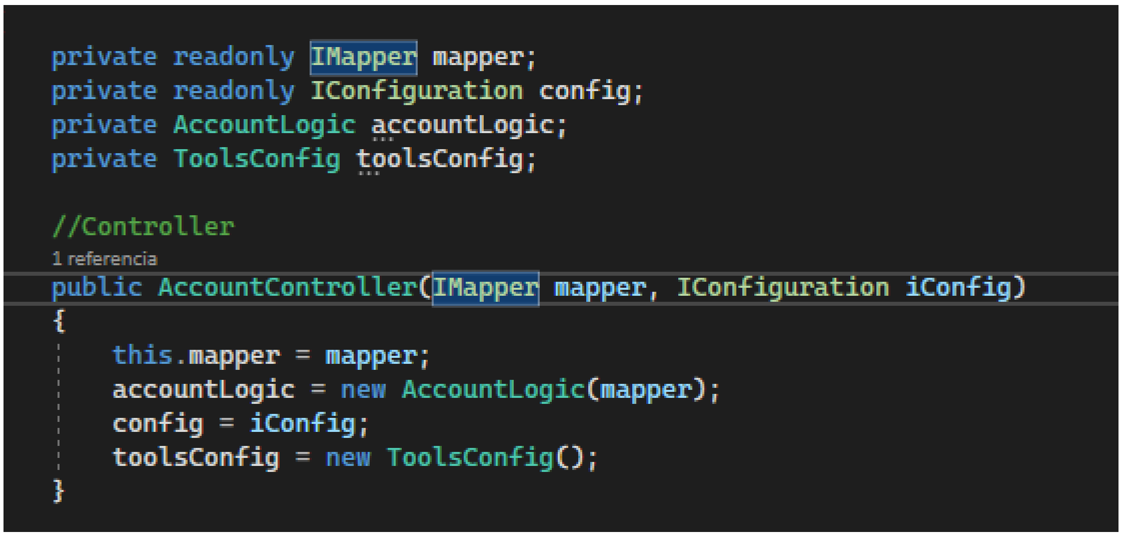
```csharp
//Search properties by a city
1 referencia
public List<PropertyBasicEntity> GetAllForCity(Guid? idCity)
{
    var propertyEntities = new List<PropertyBasicEntity>();

    if (idCity.HasValue)
    {
        var zones = zoneLogic.GetAllForCity(idCity);
        if (zones.Any())
        {
            var idZones = zones.Select(x => x.Id).ToList();
            var properties = propertyRepository.GetAllForZones(idZones);
            if (properties.Any()) propertyEntities = properties.Select(x => mapper.Map<PropertyBasicEntity>(x)).ToList();
        }

    }

    return propertyEntities;
}
```



```
//Method to get all system properties
3 referencias
public List<PropertyEntity> GetAll()
{
    var propertyEntities = new List<PropertyEntity>();

    var properties = propertyRepository.GetAll();
    if (properties.Any())
    {
        propertyEntities = properties.Select(x => mapper.Map<PropertyEntity>(x)).ToList();
        if (propertyEntities.Any()) propertyEntities.ForEach(x => Arrive(x));
    }

    return propertyEntities;
}
```

Singleton

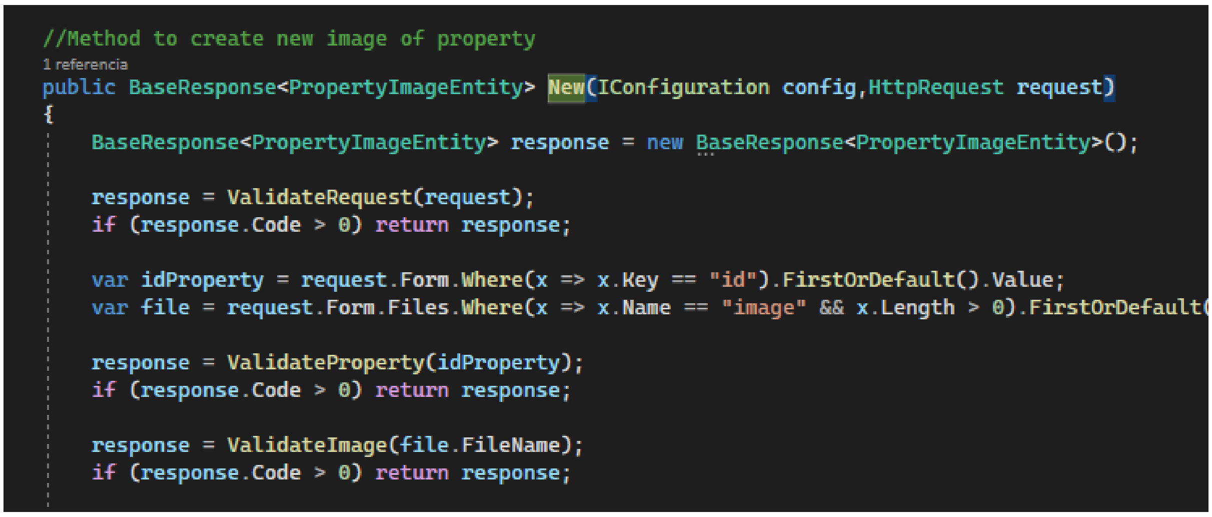
IMapper and IConfiguration cannot be instantiated several times, only once, since information is only obtained from it and they are created when a controller is called.



```
private readonly IMapper mapper;
private readonly IConfiguration config;
private AccountLogic accountLogic;
private ToolsConfig toolsConfig;

//Controller
1 referencia
public AccountController(IMapper mapper, IConfiguration iConfig)
{
    this.mapper = mapper;
    accountLogic = new AccountLogic(mapper);
    config = iConfig;
    toolsConfig = new ToolsConfig();
}
```

Dependency injection

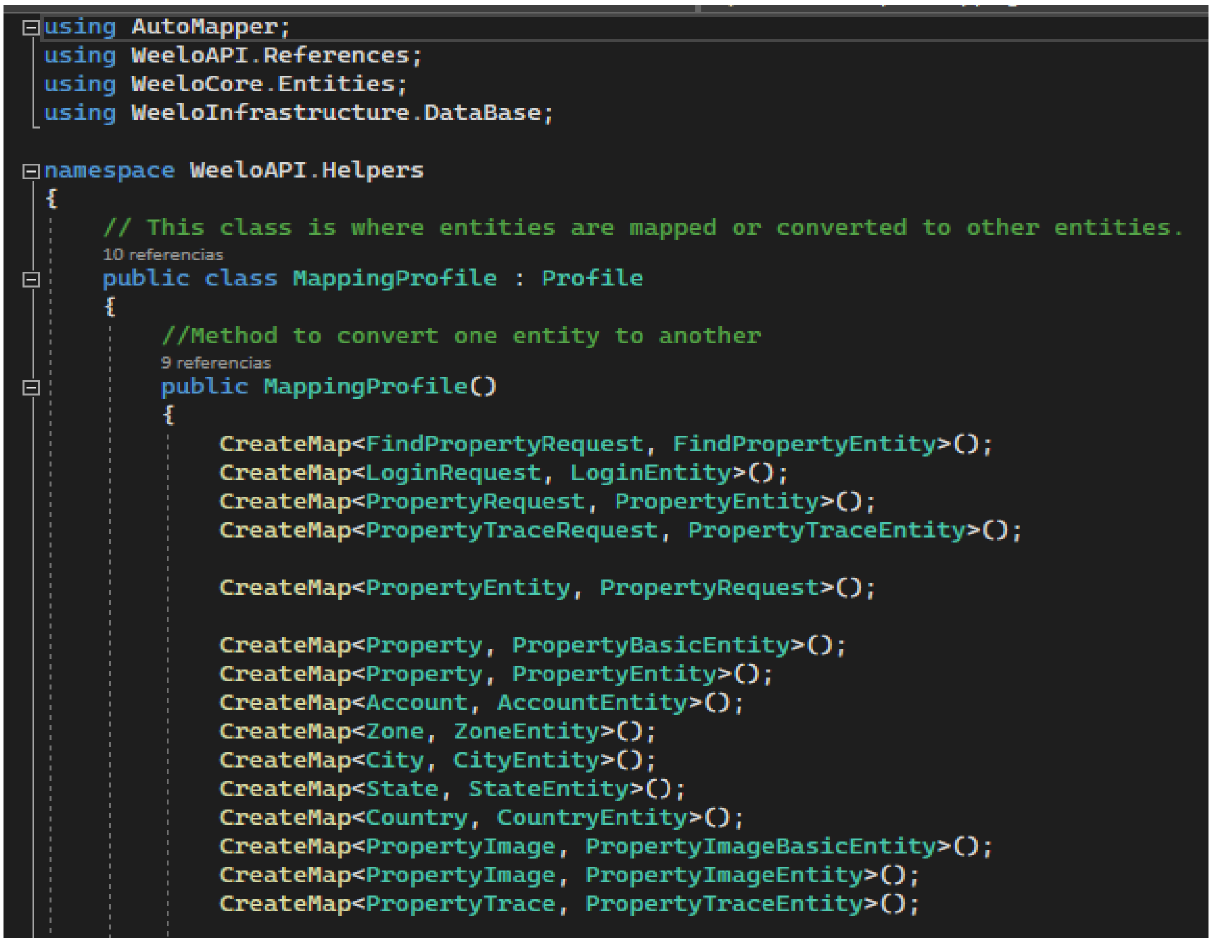
when two objects enter the method and there is no need to instantiate them in the class itself.



```
//Method to create new image of property
1 referencia
public BaseResponse<PropertyImageEntity> New(IConfiguration config,HttpRequest request)
{
    BaseResponse<PropertyImageEntity> response = new BaseResponse<PropertyImageEntity>();

    response = ValidateRequest(request);
    if (response.Code > 0) return response;

    var idProperty = request.Form.Where(x => x.Key == "id").FirstOrDefault().Value;
    var file = request.Form.Files.Where(x => x.Name == "image" && x.Length > 0).FirstOrDefault(

    response = ValidateProperty(idProperty);
    if (response.Code > 0) return response;

    response = ValidateImage(file.FileName);
    if (response.Code > 0) return response;
```

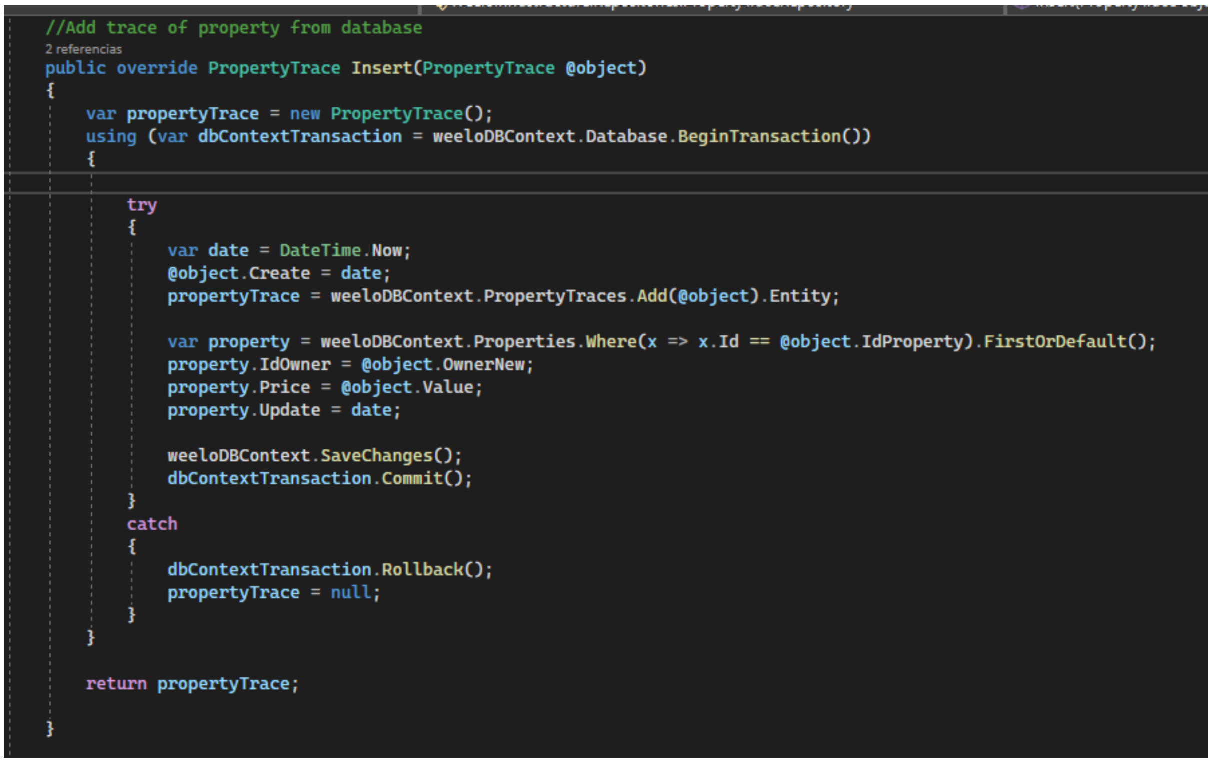
Prototype and Proxy(Mediator)

The mapper works as a dynamic cloning and proxy method.



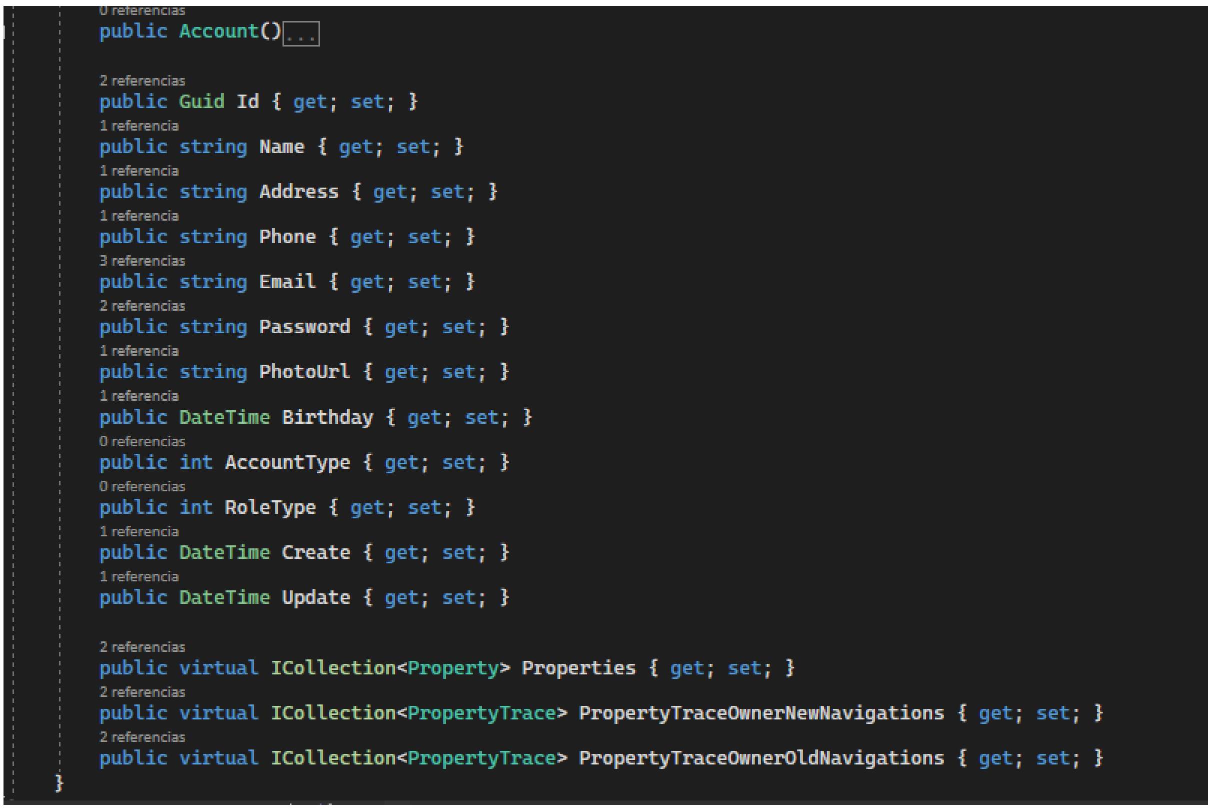
```
using AutoMapper;
using WeeloAPI.References;
using WeeloCore.Entities;
using WeeloInfrastructure.DataBase;

namespace WeeloAPI.Helpers
{
    // This class is where entities are mapped or converted to other entities.
    10 referencias
    public class MappingProfile : Profile
    {
        //Method to convert one entity to another
        9 referencias
        public MappingProfile()
        {
            CreateMap<FindPropertyRequest, FindPropertyEntity>();
            CreateMap<LoginRequest, LoginEntity>();
            CreateMap<PropertyRequest, PropertyEntity>();
            CreateMap<PropertyTraceRequest, PropertyTraceEntity>();

            CreateMap<PropertyEntity, PropertyRequest>();

            CreateMap<Property, PropertyBasicEntity>();
            CreateMap<Property, PropertyEntity>();
            CreateMap<Account, AccountEntity>();
            CreateMap<Zone, ZoneEntity>();
            CreateMap<City, CityEntity>();
            CreateMap<State, StateEntity>();
            CreateMap<Country, CountryEntity>();
            CreateMap<PropertyImage, PropertyImageBasicEntity>();
            CreateMap<PropertyImage, PropertyImageEntity>();
            CreateMap<PropertyTrace, PropertyTraceEntity>();
```

Memendo: Database temptations work as a kind of memento



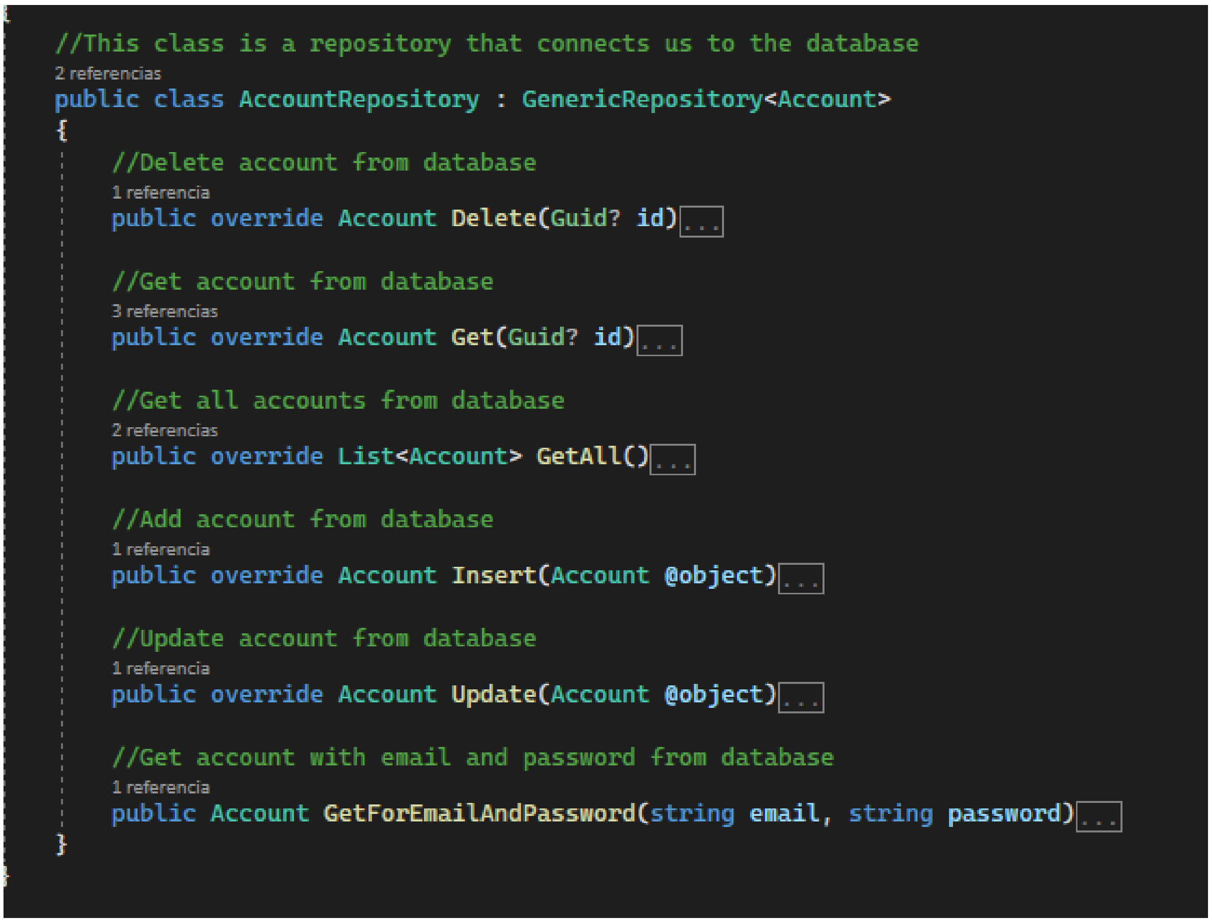
```
//Add trace of property from database
2 referencias
public override PropertyTrace Insert(PropertyTrace @object)
{
    var propertyTrace = new PropertyTrace();
    using (var dbContextTransaction = weeloDBContext.Database.BeginTransaction())
    {

        try
        {
            var date = DateTime.Now;
            @object.Create = date;
            propertyTrace = weeloDBContext.PropertyTraces.Add(@object).Entity;

            var property = weeloDBContext.Properties.Where(x => x.Id == @object.IdProperty).FirstOrDefault();
            property.IdOwner = @object.OwnerNew;
            property.Price = @object.Value;
            property.Update = date;

            weeloDBContext.SaveChanges();
            dbContextTransaction.Commit();
        }
        catch
        {
            dbContextTransaction.Rollback();
            propertyTrace = null;
        }
    }

    return propertyTrace;

}
```

DTO:



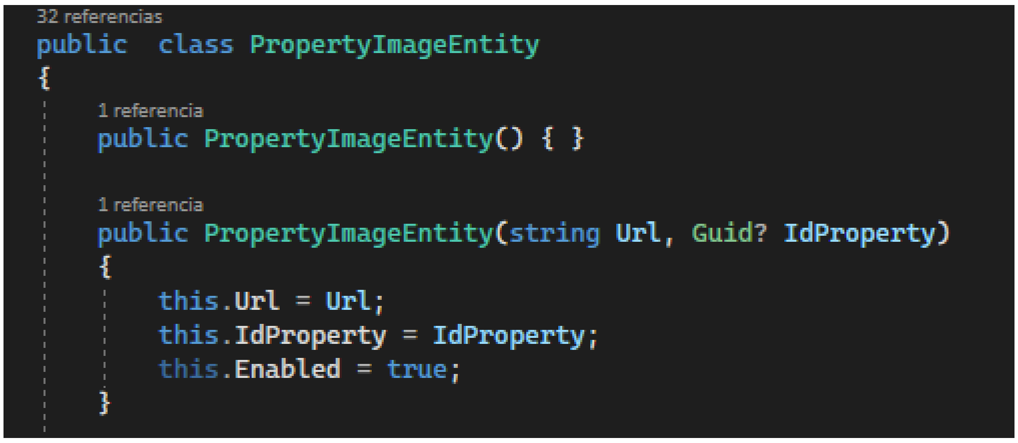
```
        0 referencias
        public Account()...

        2 referencias
        public Guid Id { get; set; }
        1 referencia
        public string Name { get; set; }
        1 referencia
        public string Address { get; set; }
        1 referencia
        public string Phone { get; set; }
        3 referencias
        public string Email { get; set; }
        2 referencias
        public string Password { get; set; }
        1 referencia
        public string PhotoUrl { get; set; }
        1 referencia
        public DateTime Birthday { get; set; }
        0 referencias
        public int AccountType { get; set; }
        0 referencias
        public int RoleType { get; set; }
        1 referencia
        public DateTime Create { get; set; }
        1 referencia
        public DateTime Update { get; set; }

        2 referencias
        public virtual ICollection<Property> Properties { get; set; }
        2 referencias
        public virtual ICollection<PropertyTrace> PropertyTraceOwnerNewNavigations { get; set; }
        2 referencias
        public virtual ICollection<PropertyTrace> PropertyTraceOwnerOldNavigations { get; set; }
    }
```

DAO:



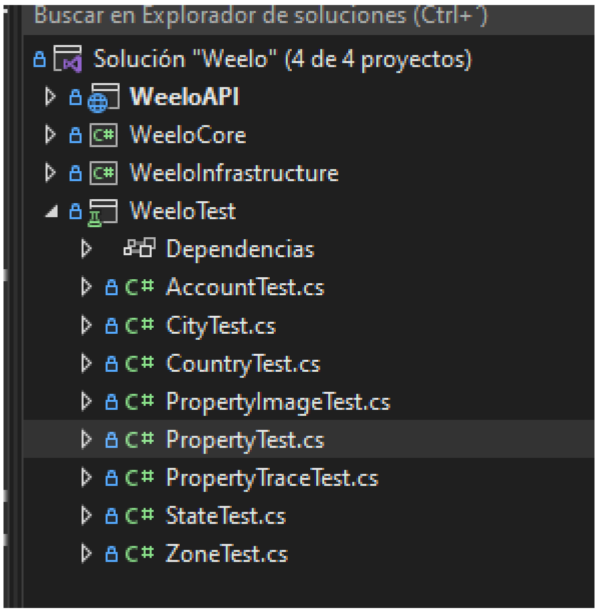
```
    //This class is a repository that connects us to the database
    2 referencias
    public class AccountRepository : GenericRepository<Account>
    {
        //Delete account from database
        1 referencia
        public override Account Delete(Guid? id)...

        //Get account from database
        3 referencias
        public override Account Get(Guid? id)...

        //Get all accounts from database
        2 referencias
        public override List<Account> GetAll()...

        //Add account from database
        1 referencia
        public override Account Insert(Account @object)...

        //Update account from database
        1 referencia
        public override Account Update(Account @object)...

        //Get account with email and password from database
        1 referencia
        public Account GetForEmailAndPassword(string email, string password)...
    }
```

Contructor:

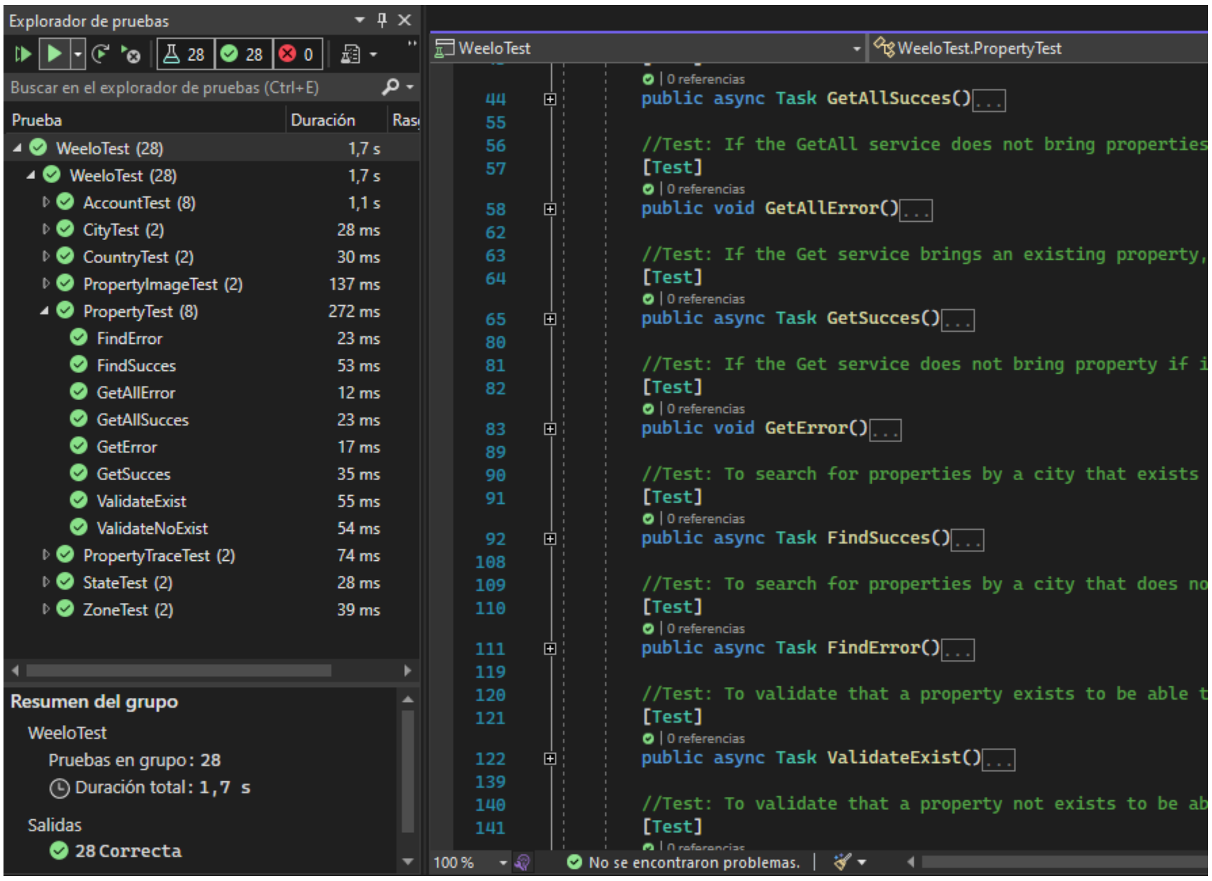


```
32 referencias
public  class PropertyImageEntity
{
    1 referencia
    public PropertyImageEntity() { }

    1 referencia
    public PropertyImageEntity(string Url, Guid? IdProperty)
    {
        this.Url = Url;
        this.IdProperty = IdProperty;
        this.Enabled = true;
    }
```

Unit Test:
Test cases for API services are created, in these we make cases of successful results and possible failures.



```
Buscar en Explorador de soluciones (Ctrl+`)
Solución "Weelo" (4 de 4 proyectos)
  ▷ WeeloAPI
  ▷ WeeloCore
  ▷ WeeloInfrastructure
  ◢ WeeloTest
      ▷ Dependencias
      ▷ C# AccountTest.cs
      ▷ C# CityTest.cs
      ▷ C# CountryTest.cs
      ▷ C# PropertyImageTest.cs
      ▷ C# PropertyTest.cs
      ▷ C# PropertyTraceTest.cs
      ▷ C# StateTest.cs
      ▷ C# ZoneTest.cs
```

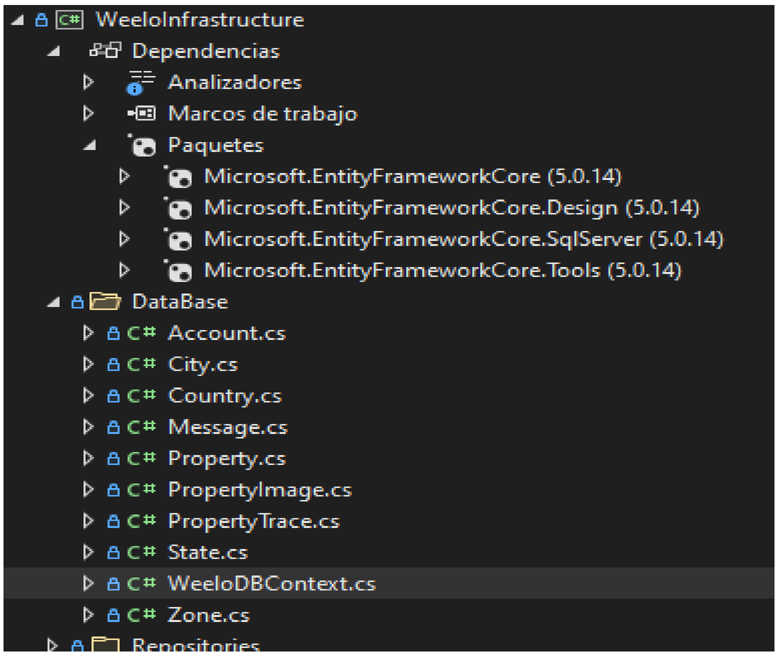


```
Explorador de pruebas
28   28   0
Buscar en el explorador de pruebas (Ctrl+E)
Prueba                          Duración   Ras
WeeloTest (28)                     1,7 s
  WeeloTest (28)                   1,7 s
    AccountTest (8)                1,1 s
    CityTest (2)                  28 ms
    CountryTest (2)               30 ms
    PropertyImageTest (2)        137 ms
    PropertyTest (8)             272 ms
      FindError                   23 ms
      FindSucces                  53 ms
      GetAllError                 12 ms
      GetAllSucces                23 ms
      GetError                    17 ms
      GetSucces                   35 ms
      ValidateExist               55 ms
      ValidateNoExist             54 ms
    PropertyTraceTest (2)         74 ms
    StateTest (2)                 28 ms
    ZoneTest (2)                  39 ms

Resumen del grupo
  WeeloTest
    Pruebas en grupo: 28
    Duración total: 1,7 s
  Salidas
    28 Correcta
```

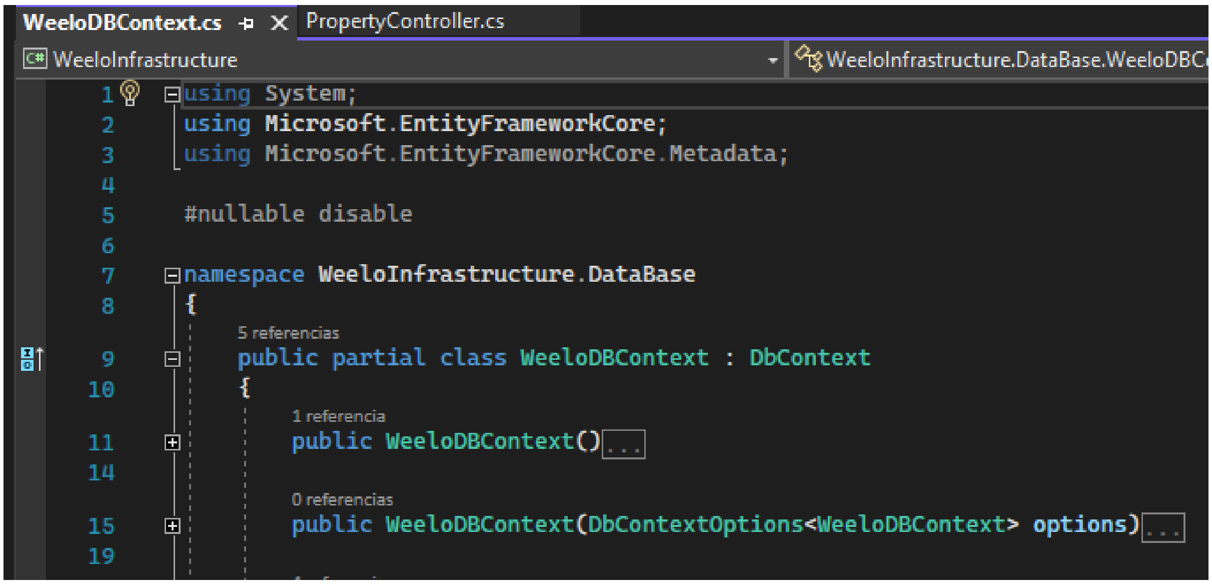
```
WeeloTest                          WeeloTest.PropertyTest
        | 0 referencias
        public async Task GetAllSucces()[...]

        //Test: If the GetAll service does not bring properties
        [Test]
        | 0 referencias
        public void GetAllError()[...]

        //Test: If the Get service brings an existing property,
        [Test]
        | 0 referencias
        public async Task GetSucces()[...]

        //Test: If the Get service does not bring property if i
        [Test]
        | 0 referencias
        public void GetError()[...]

        //Test: To search for properties by a city that exists
        [Test]
        | 0 referencias
        public async Task FindSucces()[...]

        //Test: To search for properties by a city that does no
        [Test]
        | 0 referencias
        public async Task FindError()[...]

        //Test: To validate that a property exists to be able t
        [Test]
        | 0 referencias
        public async Task ValidateExist()[...]

        //Test: To validate that a property not exists to be ab
        [Test]
        | 0 referencias
100 %     No se encontraron problemas.
```

Security:
To avoid Sql injection the ORM Entity Frame is used. If necessary, string escaping, modeling objects first.

Entity Frame

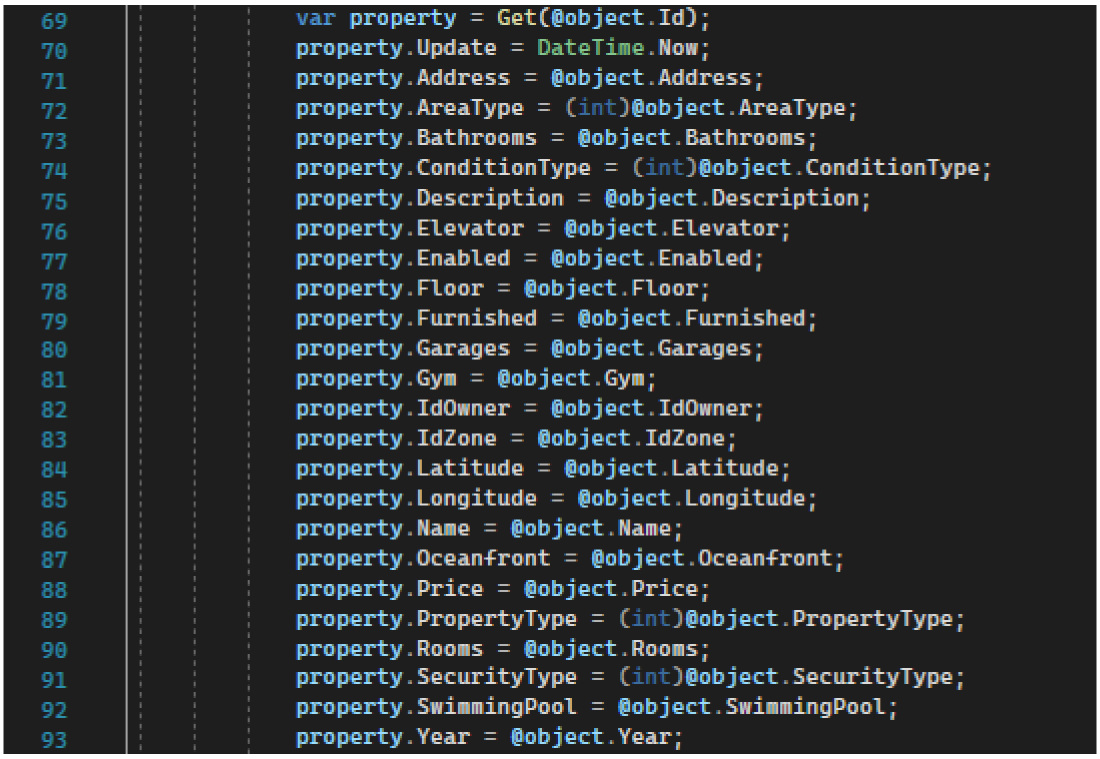


```
WeeloInfrastructure
  Dependencias
    Analizadores
    Marcos de trabajo
    Paquetes
      Microsoft.EntityFrameworkCore (5.0.14)
      Microsoft.EntityFrameworkCore.Design (5.0.14)
      Microsoft.EntityFrameworkCore.SqlServer (5.0.14)
      Microsoft.EntityFrameworkCore.Tools (5.0.14)
  DataBase
    C# Account.cs
    C# City.cs
    C# Country.cs
    C# Message.cs
    C# Property.cs
    C# PropertyImage.cs
    C# PropertyTrace.cs
    C# State.cs
    C# WeeloDBContext.cs
    C# Zone.cs
  Repositories
```



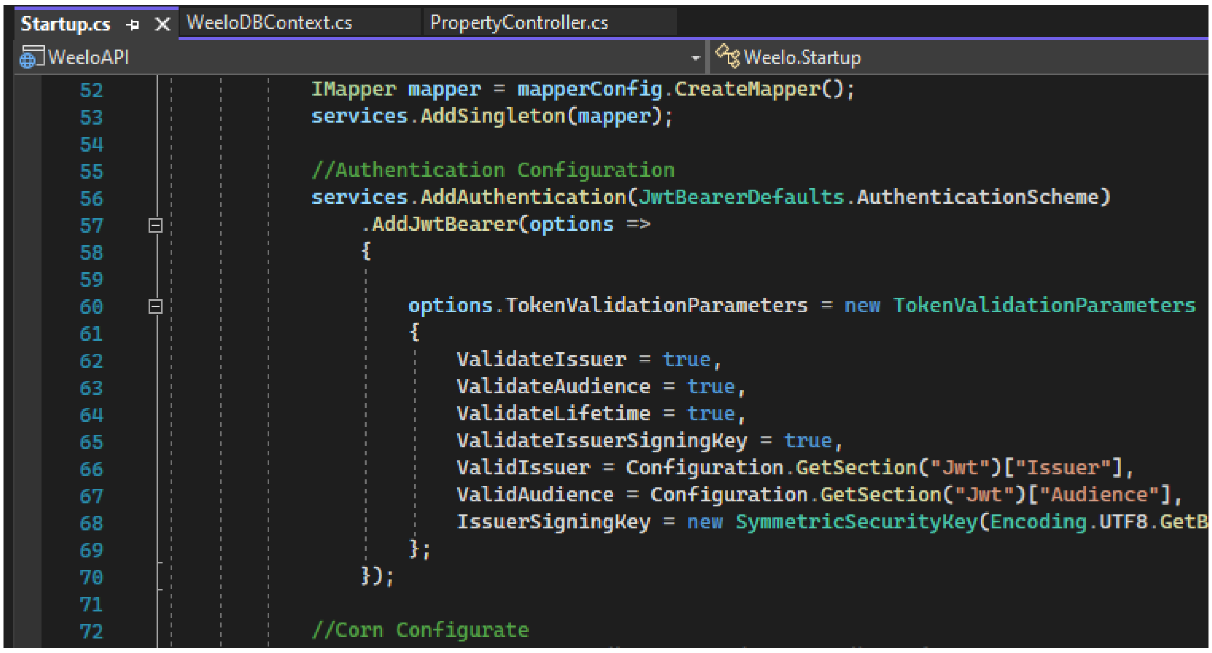
```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;

#nullable disable

namespace WeeloInfrastructure.DataBase
{
    5 referencias
    public partial class WeeloDBContext : DbContext
    {
        1 referencia
        public WeeloDBContext()...

        0 referencias
        public WeeloDBContext(DbContextOptions<WeeloDBContext> options)...
```

String escaping

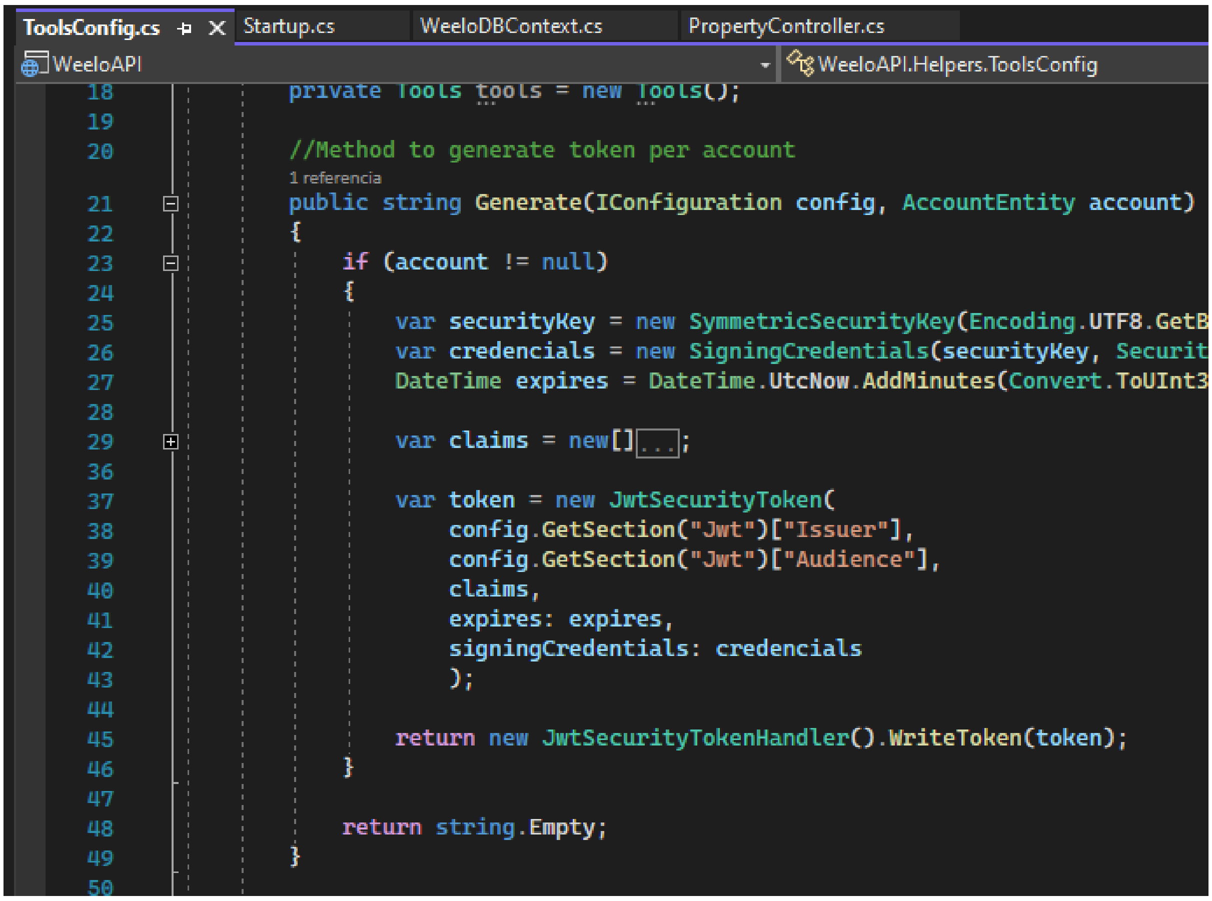


```
            var property = Get(@object.Id);
            property.Update = DateTime.Now;
            property.Address = @object.Address;
            property.AreaType = (int)@object.AreaType;
            property.Bathrooms = @object.Bathrooms;
            property.ConditionType = (int)@object.ConditionType;
            property.Description = @object.Description;
            property.Elevator = @object.Elevator;
            property.Enabled = @object.Enabled;
            property.Floor = @object.Floor;
            property.Furnished = @object.Furnished;
            property.Garages = @object.Garages;
            property.Gym = @object.Gym;
            property.IdOwner = @object.IdOwner;
            property.IdZone = @object.IdZone;
            property.Latitude = @object.Latitude;
            property.Longitude = @object.Longitude;
            property.Name = @object.Name;
            property.Oceanfront = @object.Oceanfront;
            property.Price = @object.Price;
            property.PropertyType = (int)@object.PropertyType;
            property.Rooms = @object.Rooms;
            property.SecurityType = (int)@object.SecurityType;
            property.SwimmingPool = @object.SwimmingPool;
            property.Year = @object.Year;
```

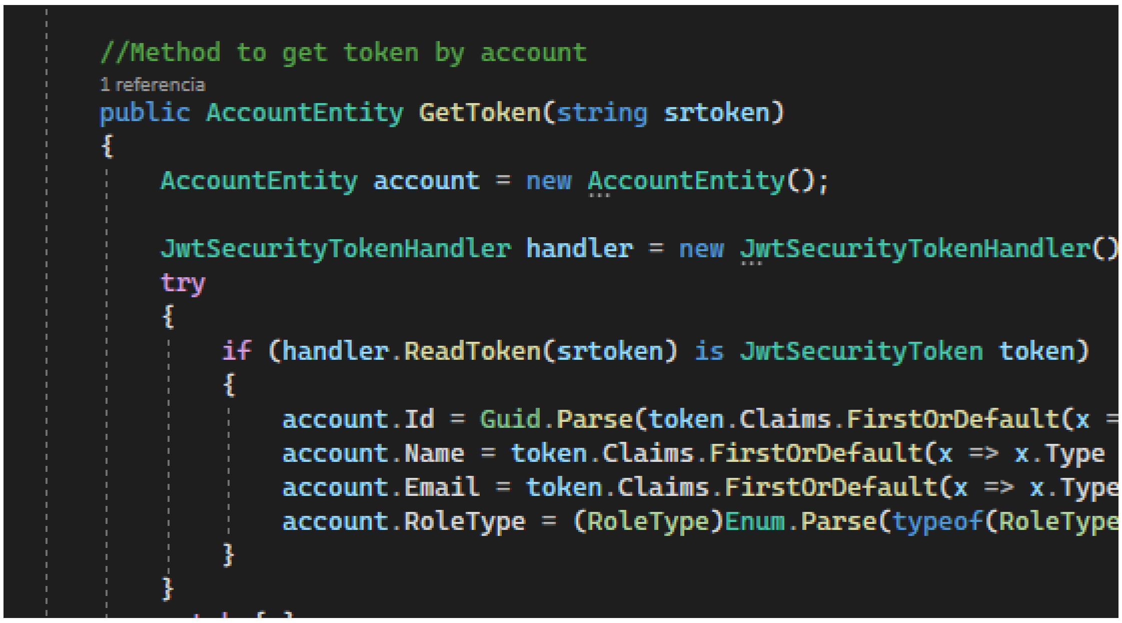
Json Web Token session token handling for **Authentication**



```
            IMapper mapper = mapperConfig.CreateMapper();
            services.AddSingleton(mapper);

            //Authentication Configuration
            services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
                .AddJwtBearer(options =>
                {

                    options.TokenValidationParameters = new TokenValidationParameters
                    {
                        ValidateIssuer = true,
                        ValidateAudience = true,
                        ValidateLifetime = true,
                        ValidateIssuerSigningKey = true,
                        ValidIssuer = Configuration.GetSection("Jwt")["Issuer"],
                        ValidAudience = Configuration.GetSection("Jwt")["Audience"],
                        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetB
                    };
                });

            //Corn Configurate
```

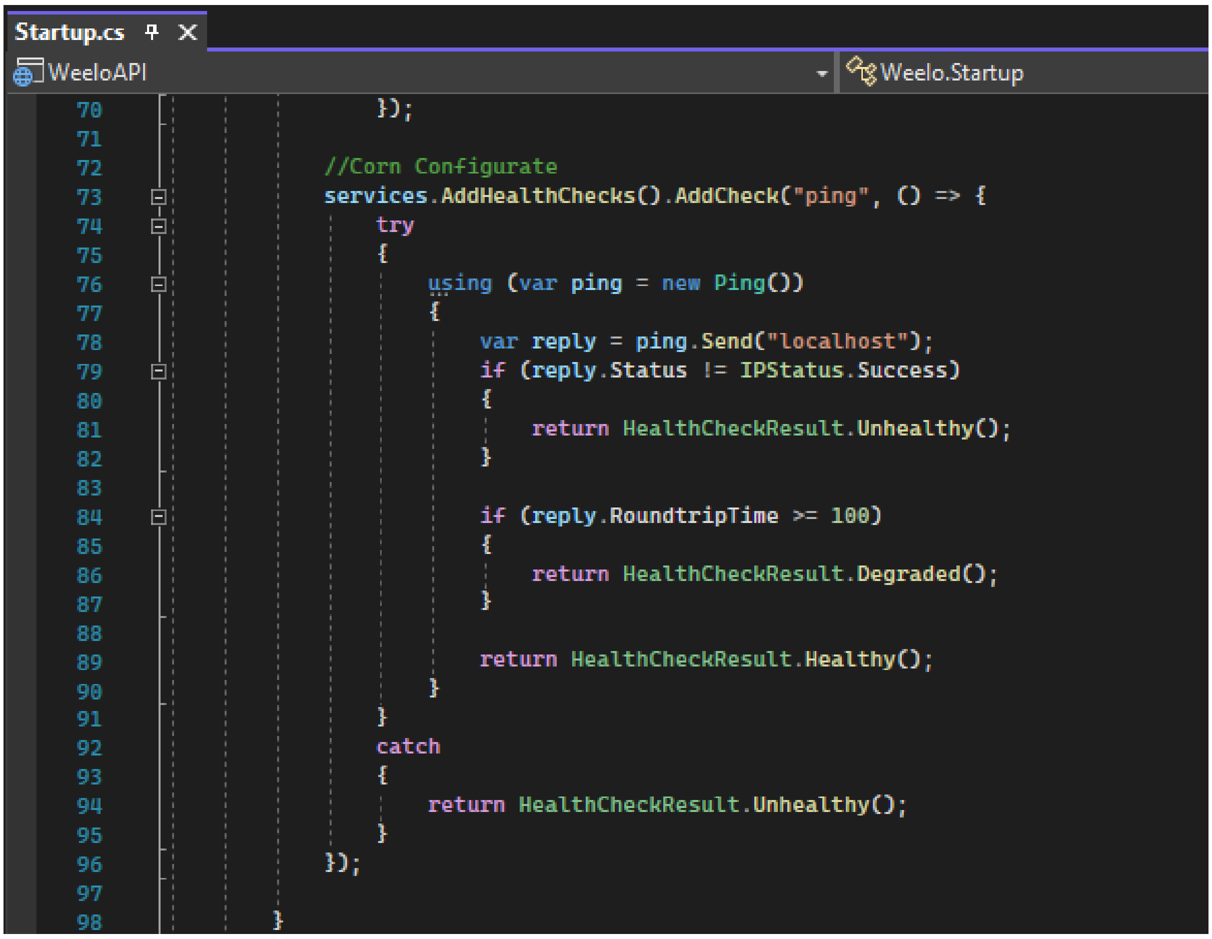


```
        private Tools tools = new Tools();

        //Method to generate token per account
        1 referencia
        public string Generate(IConfiguration config, AccountEntity account)
        {
            if (account != null)
            {
                var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetB
                var credencials = new SigningCredentials(securityKey, Securit
                DateTime expires = DateTime.UtcNow.AddMinutes(Convert.ToUInt3

                var claims = new[]...;

                var token = new JwtSecurityToken(
                    config.GetSection("Jwt")["Issuer"],
                    config.GetSection("Jwt")["Audience"],
                    claims,
                    expires: expires,
                    signingCredentials: credencials
                    );

                return new JwtSecurityTokenHandler().WriteToken(token);
            }

            return string.Empty;
        }
```



```
//Method to get token by account
1 referencia
public AccountEntity GetToken(string srtoken)
{
    AccountEntity account = new AccountEntity();

    JwtSecurityTokenHandler handler = new JwtSecurityTokenHandler()
    try
    {
        if (handler.ReadToken(srtoken) is JwtSecurityToken token)
        {
            account.Id = Guid.Parse(token.Claims.FirstOrDefault(x =
            account.Name = token.Claims.FirstOrDefault(x => x.Type
            account.Email = token.Claims.FirstOrDefault(x => x.Type
            account.RoleType = (RoleType)Enum.Parse(typeof(RoleType
        }
    }
```

Basic Corns with AddHealthChecks to validate who is pinging us:

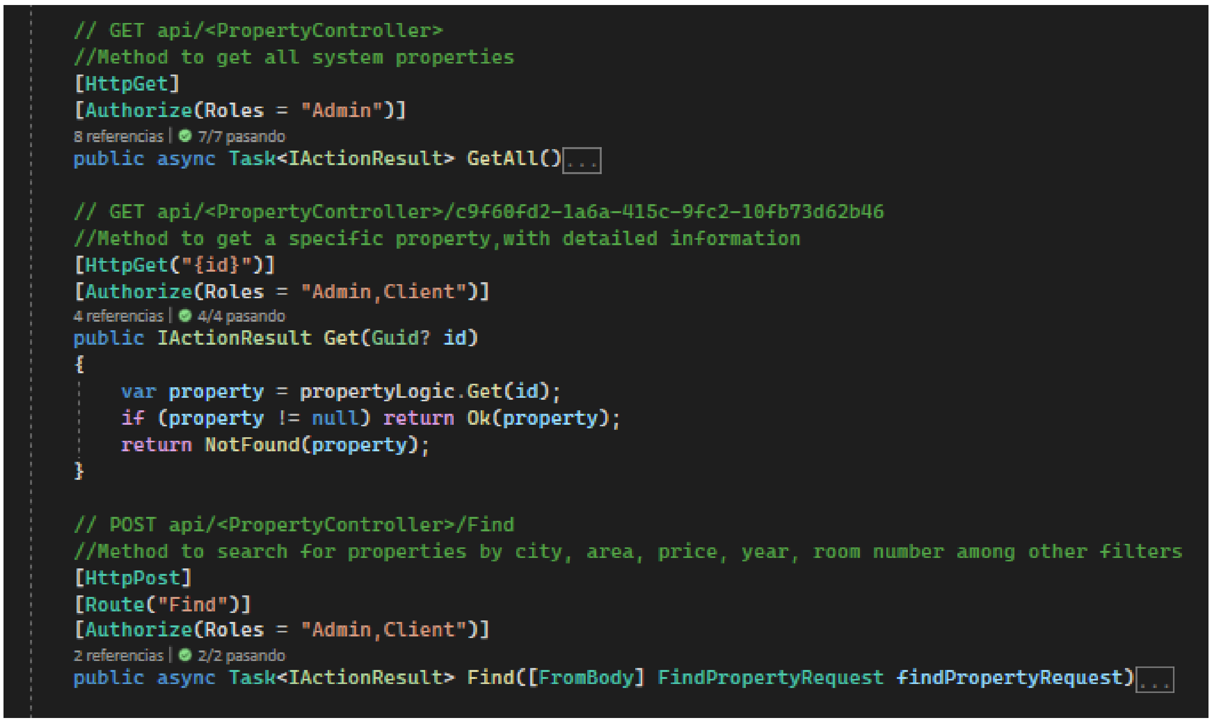


Startup.cs

WeeloAPI | Weelo.Startup

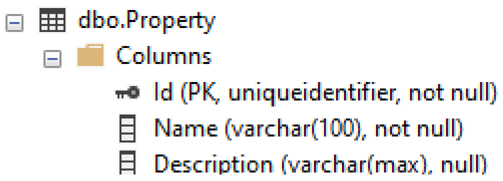
```
                });

            //Corn Configurate
            services.AddHealthChecks().AddCheck("ping", () => {
                try
                {
                    using (var ping = new Ping())
                    {
                        var reply = ping.Send("localhost");
                        if (reply.Status != IPStatus.Success)
                        {
                            return HealthCheckResult.Unhealthy();
                        }

                        if (reply.RoundtripTime >= 100)
                        {
                            return HealthCheckResult.Degraded();
                        }

                        return HealthCheckResult.Healthy();
                    }
                }
                catch
                {
                    return HealthCheckResult.Unhealthy();
                }
            });

        }
```

Authorization to endpoints by roles

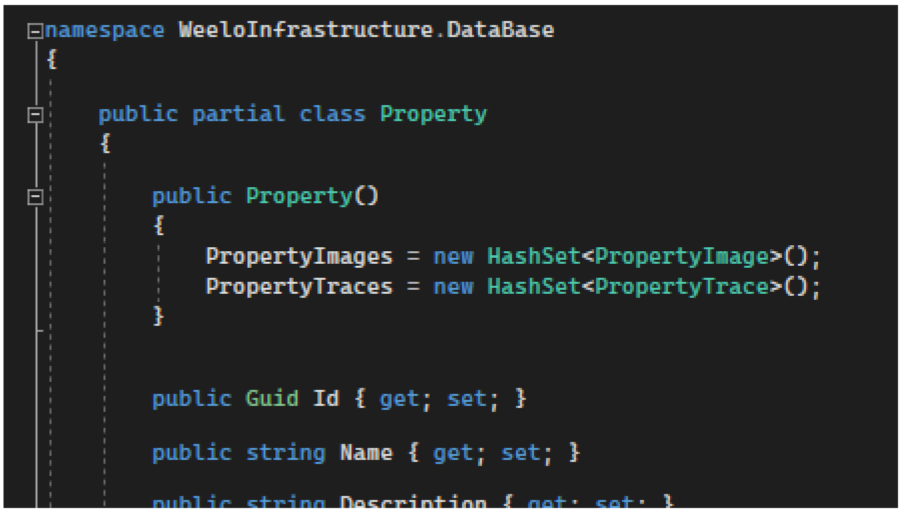


```
// GET api/<PropertyController>
//Method to get all system properties
[HttpGet]
[Authorize(Roles = "Admin")]
8 referencias | 7/7 pasando
public async Task<IActionResult> GetAll()...

// GET api/<PropertyController>/c9f60fd2-1a6a-415c-9fc2-10fb73d62b46
//Method to get a specific property,with detailed information
[HttpGet("{id}")]
[Authorize(Roles = "Admin,Client")]
4 referencias | 4/4 pasando
public IActionResult Get(Guid? id)
{
    var property = propertyLogic.Get(id);
    if (property != null) return Ok(property);
    return NotFound(property);
}

// POST api/<PropertyController>/Find
//Method to search for properties by city, area, price, year, room number among other filters
[HttpPost]
[Route("Find")]
[Authorize(Roles = "Admin,Client")]
2 referencias | 2/2 pasando
public async Task<IActionResult> Find([FromBody] FindPropertyRequest findPropertyRequest)...
```

Guid type identifiers in code and **Uniqueidentifier** in sql server.

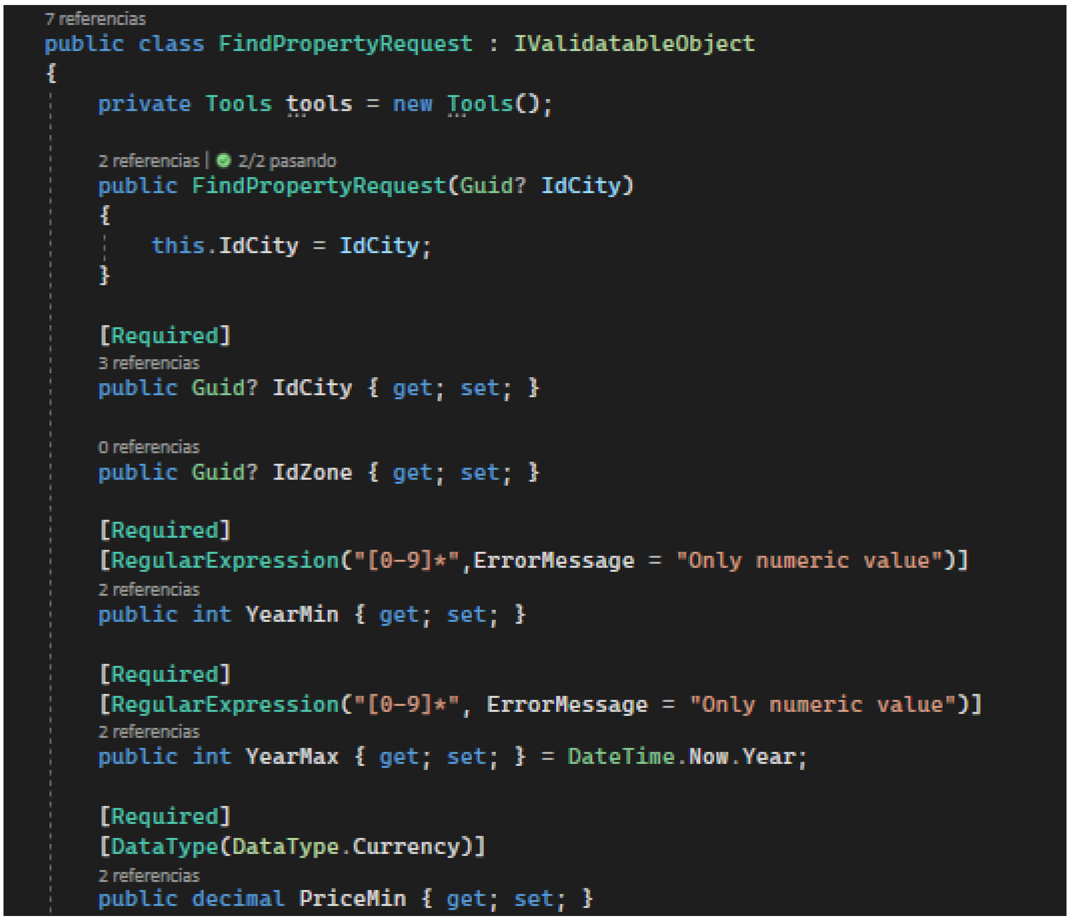


```
dbo.Property
  Columns
    Id (PK, uniqueidentifier, not null)
    Name (varchar(100), not null)
    Description (varchar(max), null)
```

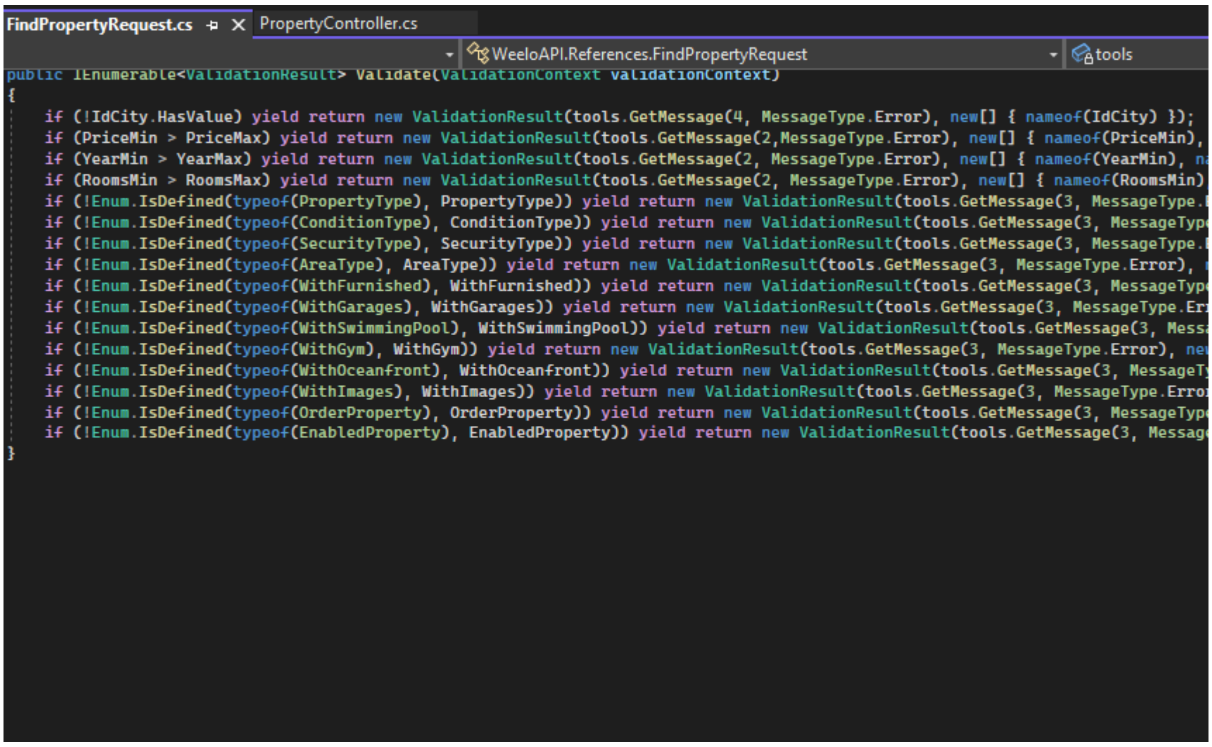


```
namespace WeeloInfrastructure.DataBase
{
    public partial class Property
    {
        public Property()
        {
            PropertyImages = new HashSet<PropertyImage>();
            PropertyTraces = new HashSet<PropertyTrace>();
        }

        public Guid Id { get; set; }

        public string Name { get; set; }

        public string Description { get; set; }
```

Validate input parameters with **IValidatableObject** and **DataAnnotations**

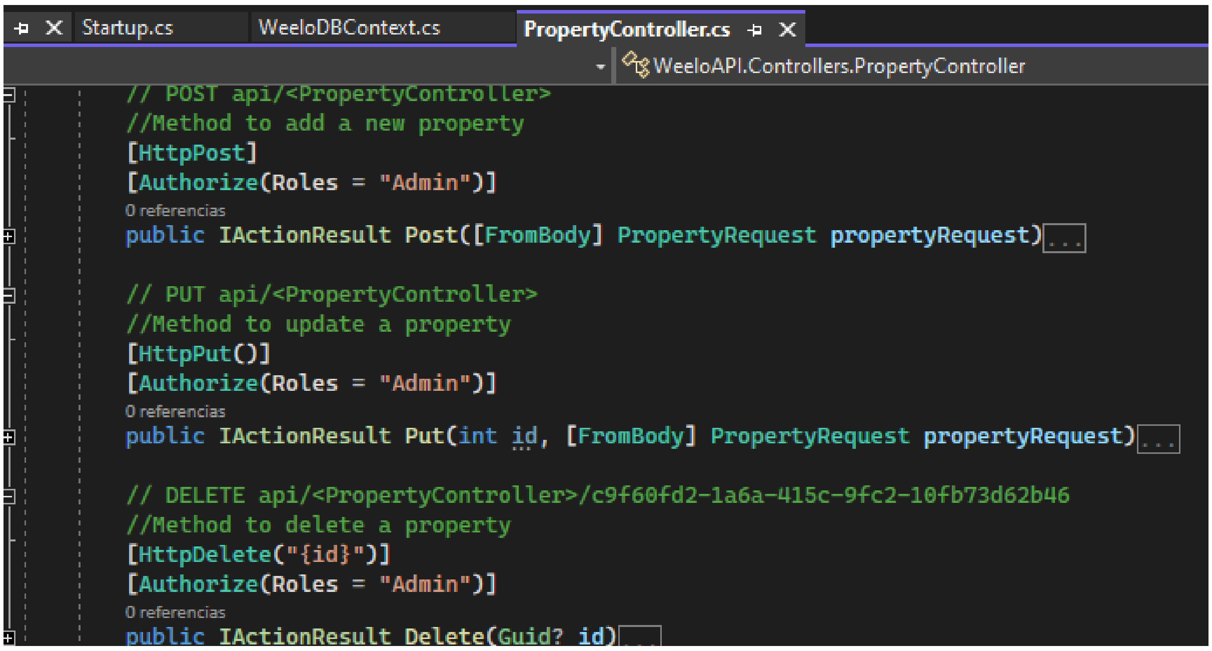


```
7 referencias
public class FindPropertyRequest : IValidatableObject
{
    private Tools tools = new Tools();

    2 referencias | 2/2 pasando
    public FindPropertyRequest(Guid? IdCity)
    {
        this.IdCity = IdCity;
    }

    [Required]
    3 referencias
    public Guid? IdCity { get; set; }

    0 referencias
    public Guid? IdZone { get; set; }

    [Required]
    [RegularExpression("[0-9]*",ErrorMessage = "Only numeric value")]
    2 referencias
    public int YearMin { get; set; }

    [Required]
    [RegularExpression("[0-9]*", ErrorMessage = "Only numeric value")]
    2 referencias
    public int YearMax { get; set; } = DateTime.Now.Year;

    [Required]
    [DataType(DataType.Currency)]
    2 referencias
    public decimal PriceMin { get; set; }
```



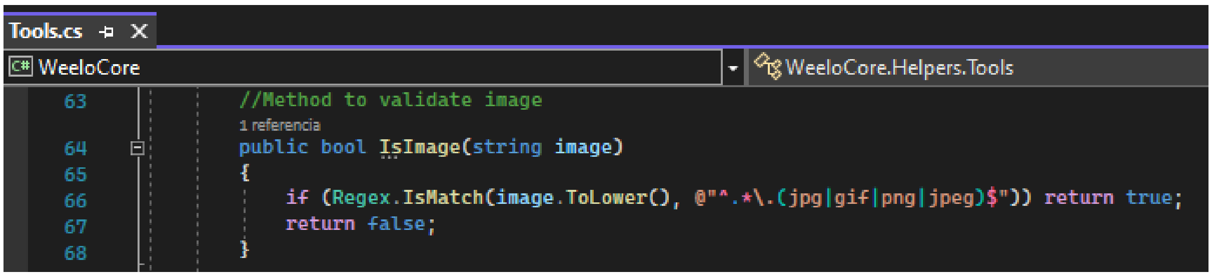
```
public IEnumerable<ValidationResult> Validate(ValidationContext validationContext)
{
    if (!IdCity.HasValue) yield return new ValidationResult(tools.GetMessage(4, MessageType.Error), new[] { nameof(IdCity) });
    if (PriceMin > PriceMax) yield return new ValidationResult(tools.GetMessage(2,MessageType.Error), new[] { nameof(PriceMin),
    if (YearMin > YearMax) yield return new ValidationResult(tools.GetMessage(2, MessageType.Error), new[] { nameof(YearMin), na
    if (RoomsMin > RoomsMax) yield return new ValidationResult(tools.GetMessage(2, MessageType.Error), new[] { nameof(RoomsMin)
    if (!Enum.IsDefined(typeof(PropertyType), PropertyType)) yield return new ValidationResult(tools.GetMessage(3, MessageType.
    if (!Enum.IsDefined(typeof(ConditionType), ConditionType)) yield return new ValidationResult(tools.GetMessage(3, MessageType
    if (!Enum.IsDefined(typeof(SecurityType), SecurityType)) yield return new ValidationResult(tools.GetMessage(3, MessageType.
    if (!Enum.IsDefined(typeof(AreaType), AreaType)) yield return new ValidationResult(tools.GetMessage(3, MessageType.Error),
    if (!Enum.IsDefined(typeof(WithFurnished), WithFurnished)) yield return new ValidationResult(tools.GetMessage(3, MessageType
    if (!Enum.IsDefined(typeof(WithGarages), WithGarages)) yield return new ValidationResult(tools.GetMessage(3, MessageType.Er
    if (!Enum.IsDefined(typeof(WithSwimmingPool), WithSwimmingPool)) yield return new ValidationResult(tools.GetMessage(3, Messa
    if (!Enum.IsDefined(typeof(WithGym), WithGym)) yield return new ValidationResult(tools.GetMessage(3, MessageType.Error), ne
    if (!Enum.IsDefined(typeof(WithOceanfront), WithOceanfront)) yield return new ValidationResult(tools.GetMessage(3, MessageTy
    if (!Enum.IsDefined(typeof(WithImages), WithImages)) yield return new ValidationResult(tools.GetMessage(3, MessageType.Erro
    if (!Enum.IsDefined(typeof(OrderProperty), OrderProperty)) yield return new ValidationResult(tools.GetMessage(3, MessageType
    if (!Enum.IsDefined(typeof(EnabledProperty), EnabledProperty)) yield return new ValidationResult(tools.GetMessage(3, Message
}
```

Call the http **(delete, put , get, post , path)**



```
Startup.cs    WeeloDBContext.cs    PropertyController.cs
WeeloAPI.Controllers.PropertyController

// POST api/<PropertyController>
//Method to add a new property
[HttpPost]
[Authorize(Roles = "Admin")]
0 referencias
public IActionResult Post([FromBody] PropertyRequest propertyRequest)...

// PUT api/<PropertyController>
//Method to update a property
[HttpPut()]
[Authorize(Roles = "Admin")]
0 referencias
public IActionResult Put(int id, [FromBody] PropertyRequest propertyRequest)...

// DELETE api/<PropertyController>/c9f60fd2-1a6a-415c-9fc2-10fb73d62b46
//Method to delete a property
[HttpDelete("{id}")]
[Authorize(Roles = "Admin")]
0 referencias
public IActionResult Delete(Guid? id)...
```

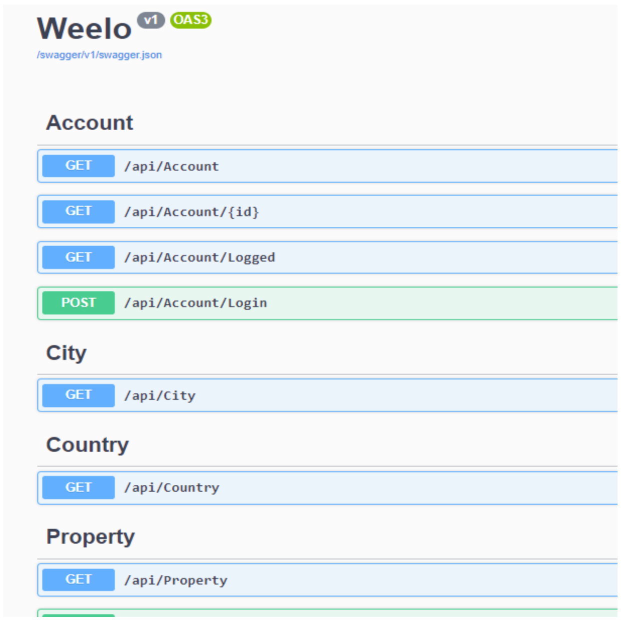
Regex field validations



```
Tools.cs
WeeloCore    WeeloCore.Helpers.Tools

    //Method to validate image
    1 referencia
    public bool IsImage(string image)
    {
        if (Regex.IsMatch(image.ToLower(), @"^.*\.(jpg|gif|png|jpeg)$")) return true;
        return false;
    }
```

Manage Performance:

Swagger: to manage the services we use the swagger tool



Weelo v1 OAS3
/swagger/v1/swagger.json

Account

GET /api/Account
GET /api/Account/{id}
GET /api/Account/Logged
POST /api/Account/Login

City

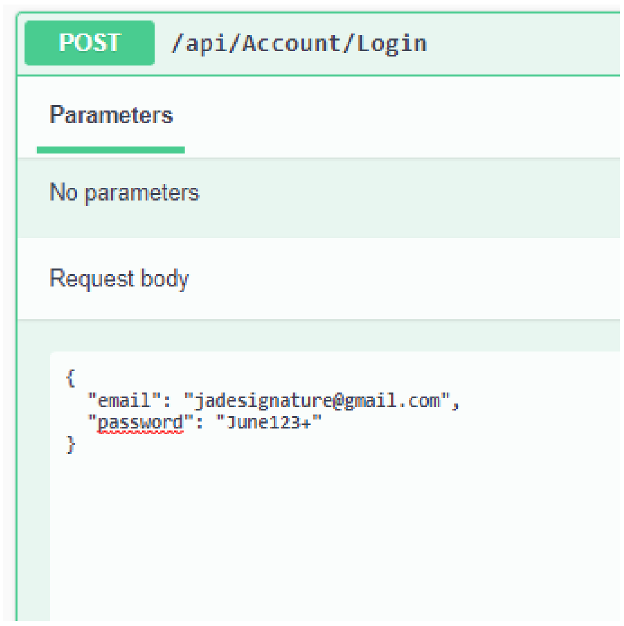
GET /api/City

Country

GET /api/Country

Property

GET /api/Property

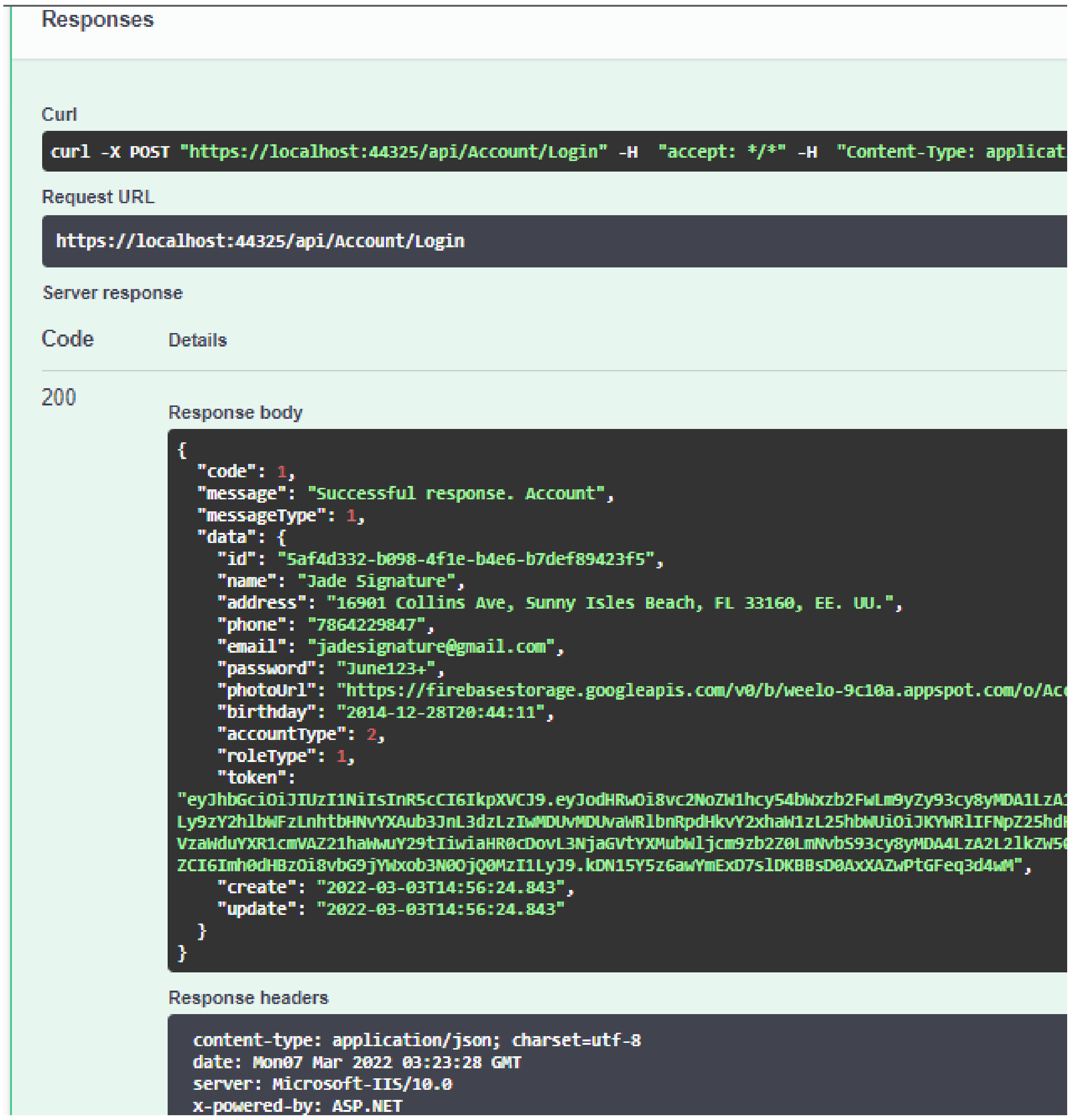

POST /api/Account/Login

Parameters

No parameters

Request body

```
{
  "email": "jadesignature@gmail.com",
  "password": "June123+"
}
```


Responses

Curl

```
curl -X POST "https://localhost:44325/api/Account/Login" -H  "accept: */*" -H  "Content-Type: applicat
```

Request URL

```
https://localhost:44325/api/Account/Login
```

Server response

Code Details

200

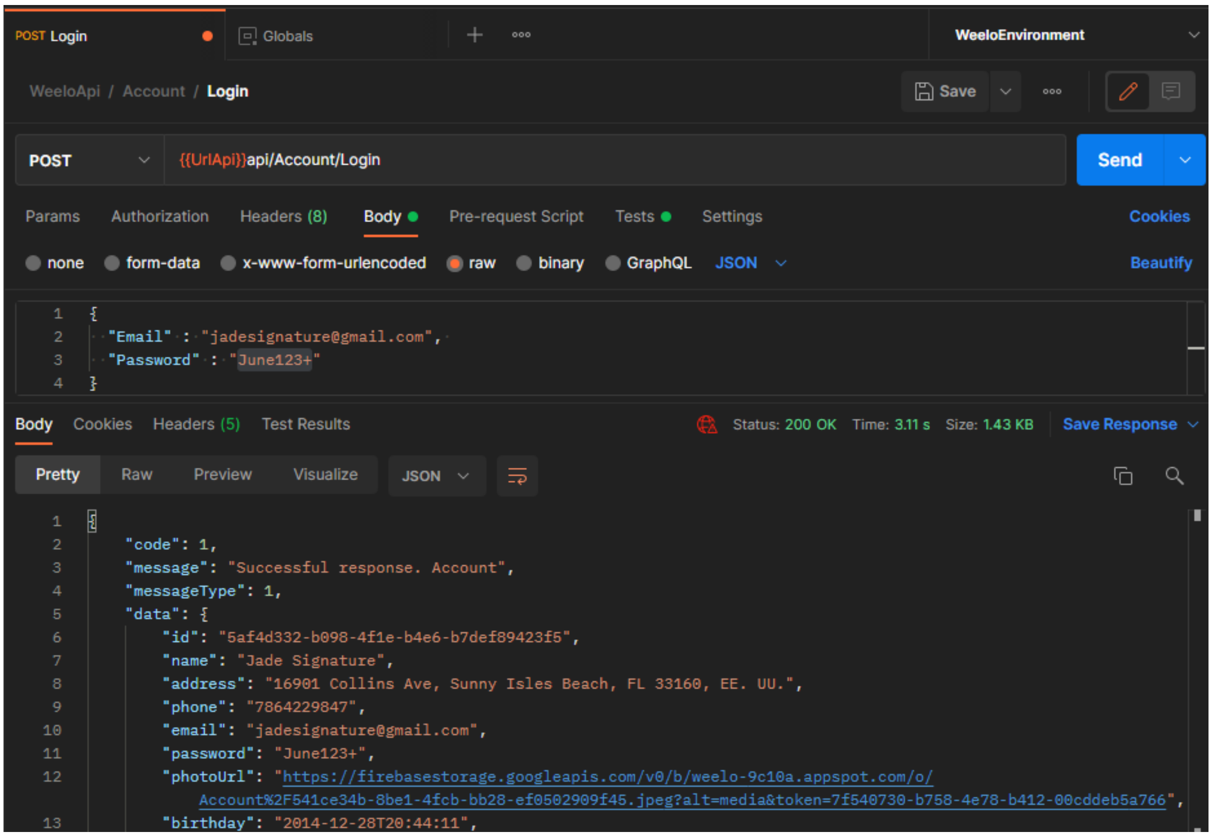
Response body

```
{
  "code": 1,
  "message": "Successful response. Account",
  "messageType": 1,
  "data": {
    "id": "5af4d332-b098-4f1e-b4e6-b7def89423f5",
    "name": "Jade Signature",
    "address": "16901 Collins Ave, Sunny Isles Beach, FL 33160, EE. UU.",
    "phone": "7864229847",
    "email": "jadesignature@gmail.com",
    "password": "June123+",
    "photoUrl": "https://firebasestorage.googleapis.com/v0/b/weelo-9c10a.appspot.com/o/Ac
    "birthday": "2014-12-28T20:44:11",
    "accountType": 2,
    "roleType": 1,
    "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA
Ly9zY2hlbWFzLnhtbHNvYXAub3JnL3dzLzIwMDUvMDUvaWRlbnRpdHkvY2xhaW1zL25hbWUiOiJKYWRlIFNpZ25hd
VzaWduYXR1cmVAZ21haWwuY29tIiwiaHR0cDovL3NjaGVtYXMubWljcm9zb2Z0LmNvbS93cy8yMDA4LzA2L2lkZW5
ZCI6Imh0dHBzOi8vbG9jYWxob3N0OjQ0MzI1LyJ9.kDN15Y5z6awYmExD7slDKBBsD0AxXAZwPtGFeq3d4wM",
    "create": "2022-03-03T14:56:24.843",
    "update": "2022-03-03T14:56:24.843"
  }
}
```

Response headers

```
 content-type: application/json; charset=utf-8
 date: Mon07 Mar 2022 03:23:28 GMT
 server: Microsoft-IIS/10.0
 x-powered-by: ASP.NET
```

Postman: To manage the services, you know their response times, among other things, we use the postman tool.


POST Login
Globals
WeeloEnvironment
WeeloApi / Account / Login
Save
POST
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify
Body Cookies Headers (5) Test Results
Save Response
Pretty Raw Preview Visualize JSON

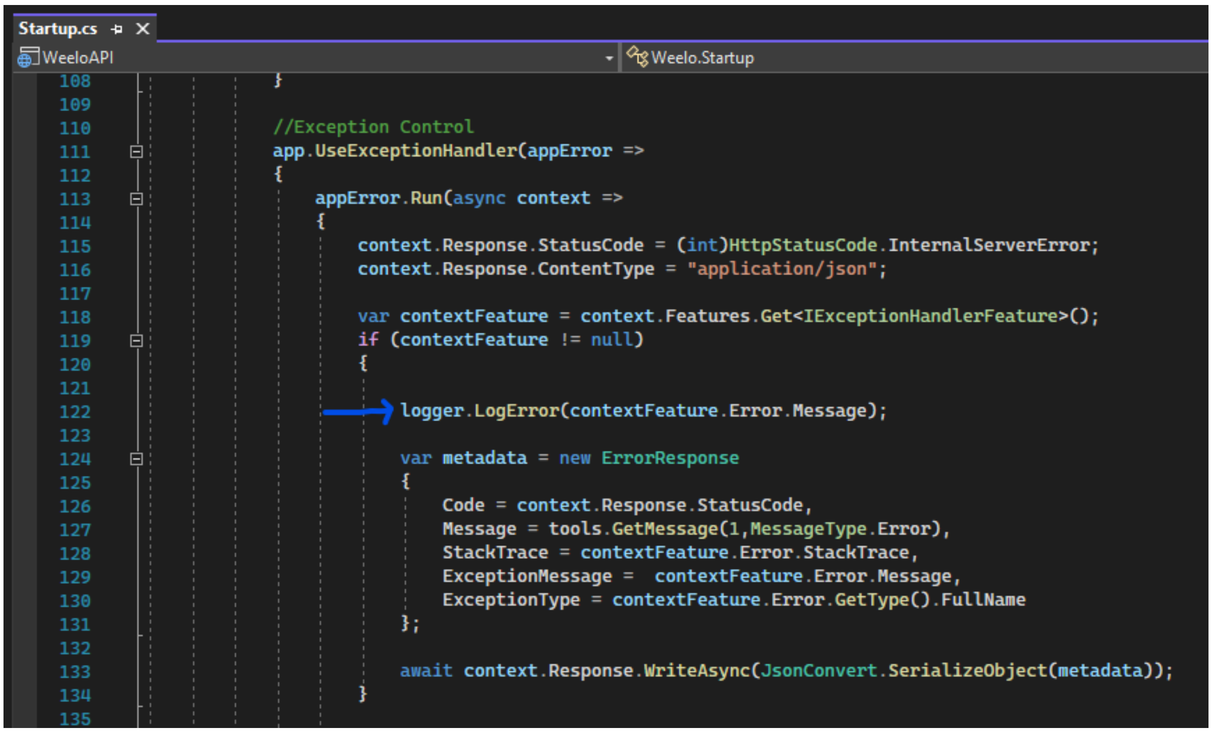
```
{{UrlApi}}api/Account/Login
```

```
{
  "Email" : "jadesignature@gmail.com",
  "Password" : "June123+"
}
```

Status: 200 OK Time: 3.11 s Size: 1.43 KB

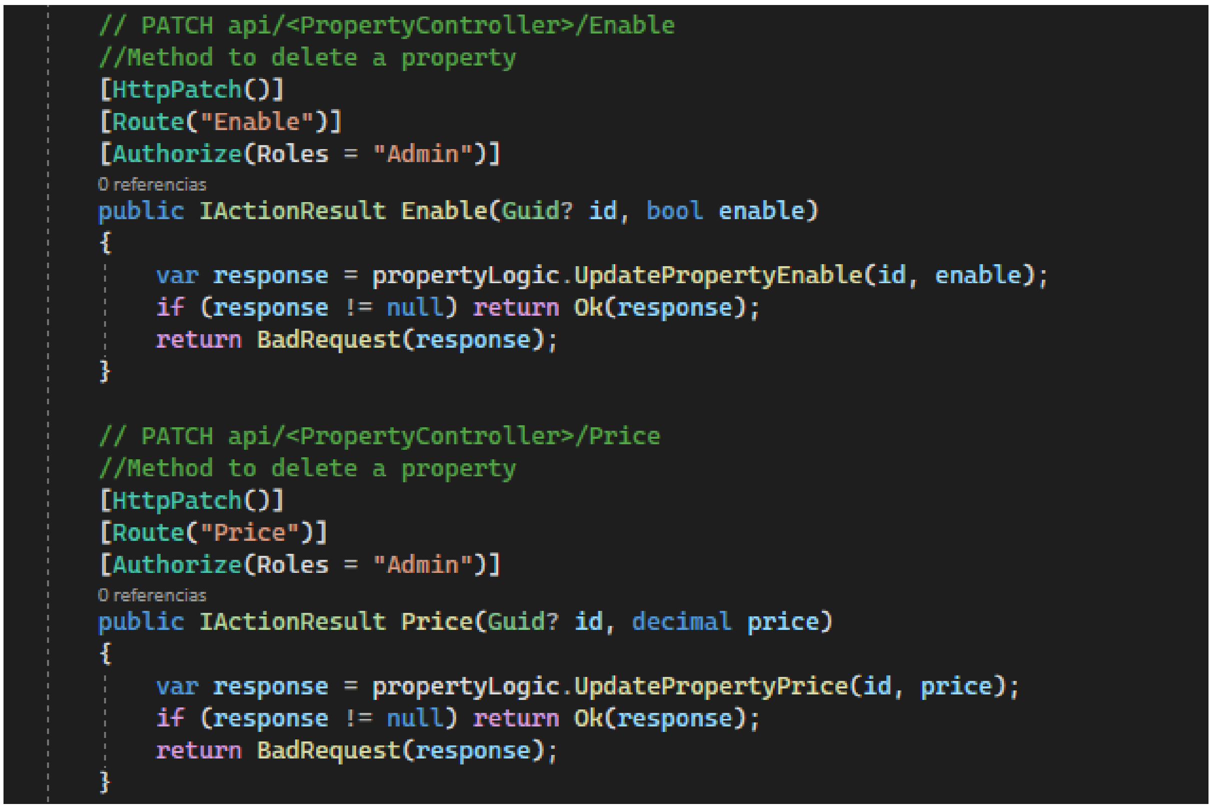
```
{
    "code": 1,
    "message": "Successful response. Account",
    "messageType": 1,
    "data": {
        "id": "5af4d332-b098-4f1e-b4e6-b7def89423f5",
        "name": "Jade Signature",
        "address": "16901 Collins Ave, Sunny Isles Beach, FL 33160, EE. UU.",
        "phone": "7864229847",
        "email": "jadesignature@gmail.com",
        "password": "June123+",
        "photoUrl": "https://firebasestorage.googleapis.com/v0/b/weelo-9c10a.appspot.com/o/
            Account%2F541ce34b-8be1-4fcb-bb28-ef0502909f45.jpeg?alt=media&token=7f540730-b758-4e78-b412-00cddeb5a766",
        "birthday": "2014-12-28T20:44:11",
```

Illoger management to save logs



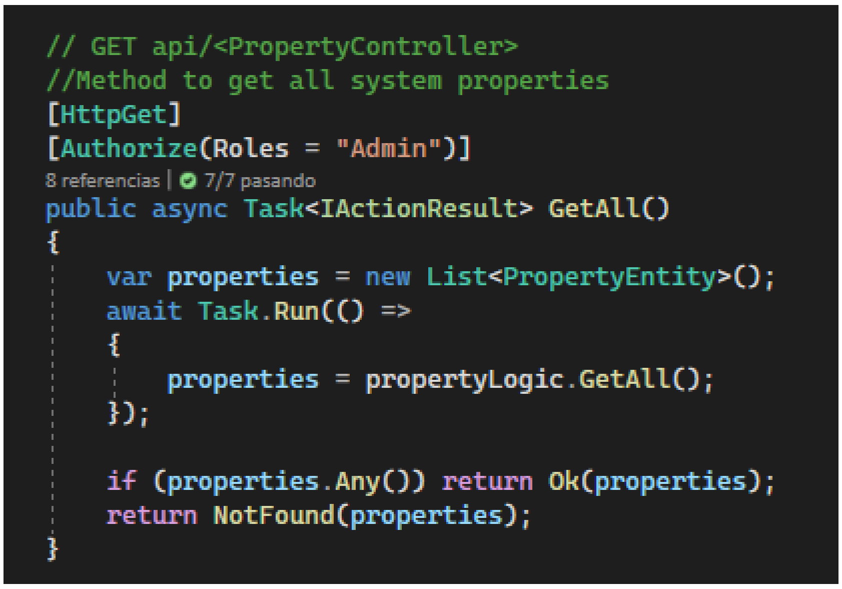
```
            }

            //Exception Control
            app.UseExceptionHandler(appError =>
            {
                appError.Run(async context =>
                {
                    context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;
                    context.Response.ContentType = "application/json";

                    var contextFeature = context.Features.Get<IExceptionHandlerFeature>();
                    if (contextFeature != null)
                    {
                        logger.LogError(contextFeature.Error.Message);

                        var metadata = new ErrorResponse
                        {
                            Code = context.Response.StatusCode,
                            Message = tools.GetMessage(1,MessageType.Error),
                            StackTrace = contextFeature.Error.StackTrace,
                            ExceptionMessage =  contextFeature.Error.Message,
                            ExceptionType = contextFeature.Error.GetType().FullName
                        };

                        await context.Response.WriteAsync(JsonConvert.SerializeObject(metadata));
                    }
```

Handle partial updates with path



```
// PATCH api/<PropertyController>/Enable
//Method to delete a property
[HttpPatch()]
[Route("Enable")]
[Authorize(Roles = "Admin")]
0 referencias
public IActionResult Enable(Guid? id, bool enable)
{
    var response = propertyLogic.UpdatePropertyEnable(id, enable);
    if (response != null) return Ok(response);
    return BadRequest(response);
}

// PATCH api/<PropertyController>/Price
//Method to delete a property
[HttpPatch()]
[Route("Price")]
[Authorize(Roles = "Admin")]
0 referencias
public IActionResult Price(Guid? id, decimal price)
{
    var response = propertyLogic.UpdatePropertyPrice(id, price);
    if (response != null) return Ok(response);
    return BadRequest(response);
}
```

Task management and **concurrent processes**



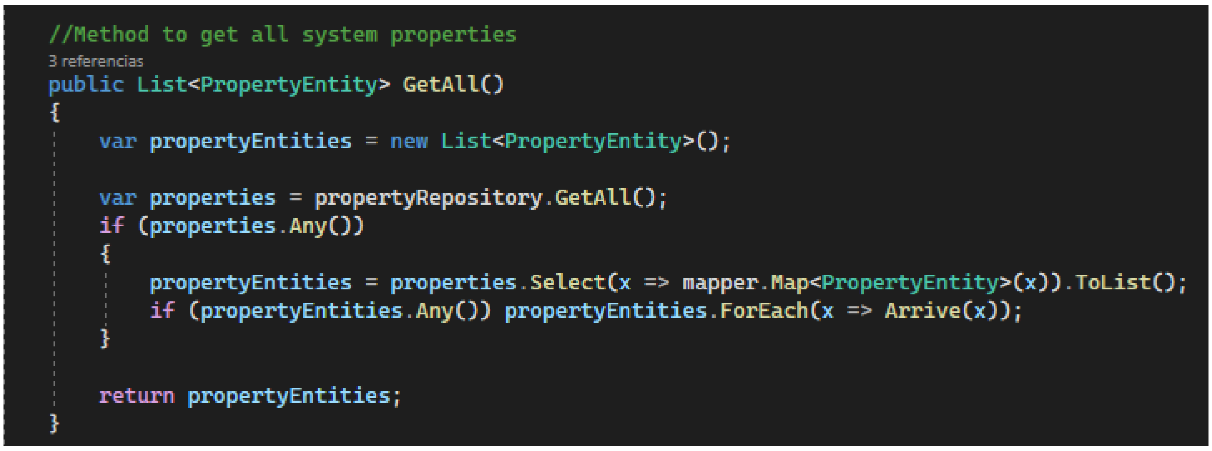
```
// GET api/<PropertyController>
//Method to get all system properties
[HttpGet]
[Authorize(Roles = "Admin")]
8 referencias | 7/7 pasando
public async Task<IActionResult> GetAll()
{
    var properties = new List<PropertyEntity>();
    await Task.Run(() =>
    {
        properties = propertyLogic.GetAll();
    });

    if (properties.Any()) return Ok(properties);
    return NotFound(properties);
}
```

Best Practices:
Several good practices have already been implemented and explained from the architecture level, such as structure, security and performance, however many more good programming practices were implemented in this API.

Linq : work with the linq tool for handling data structures.

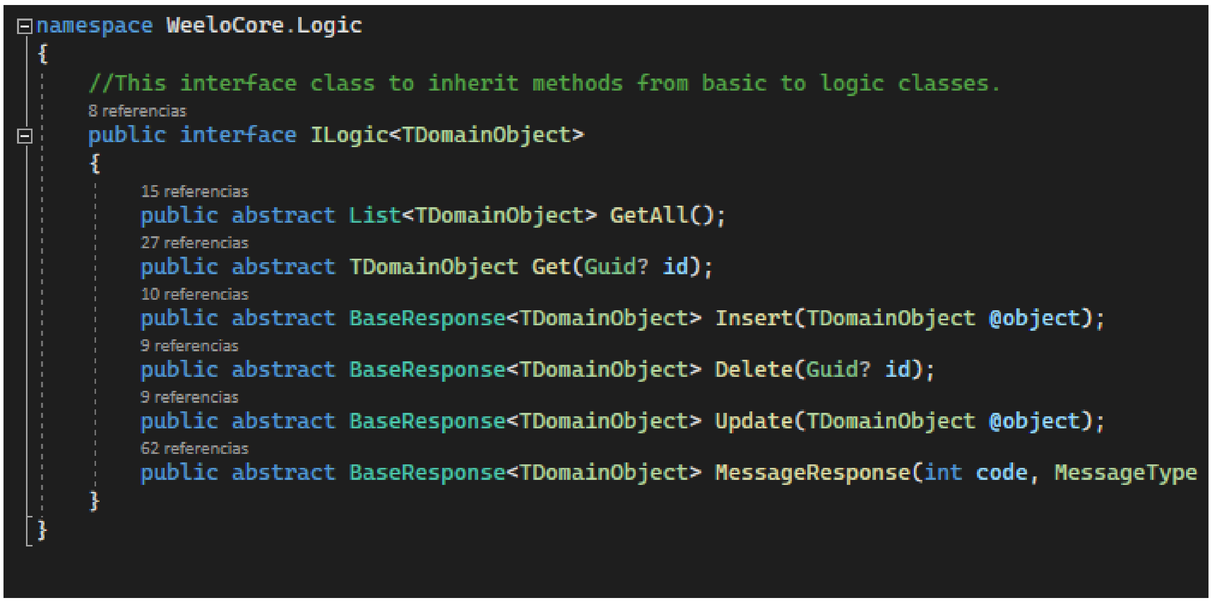
Any: Validate lists with Any and not with null

Component Oriented Programming

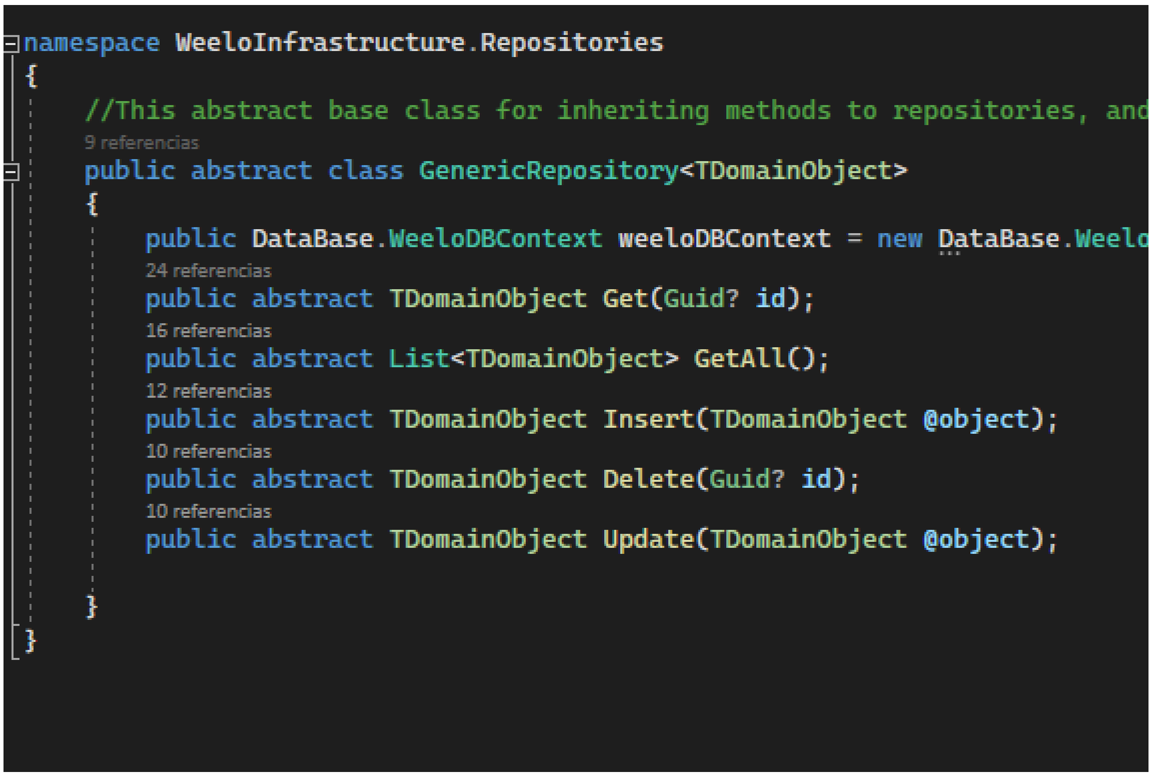


```
//Method to get all system properties
3 referencias
public List<PropertyEntity> GetAll()
{
    var propertyEntities = new List<PropertyEntity>();

    var properties = propertyRepository.GetAll();
    if (properties.Any())
    {
        propertyEntities = properties.Select(x => mapper.Map<PropertyEntity>(x)).ToList();
        if (propertyEntities.Any()) propertyEntities.ForEach(x => Arrive(x));
    }

    return propertyEntities;
}
```

Handling interfaces and abstract classes

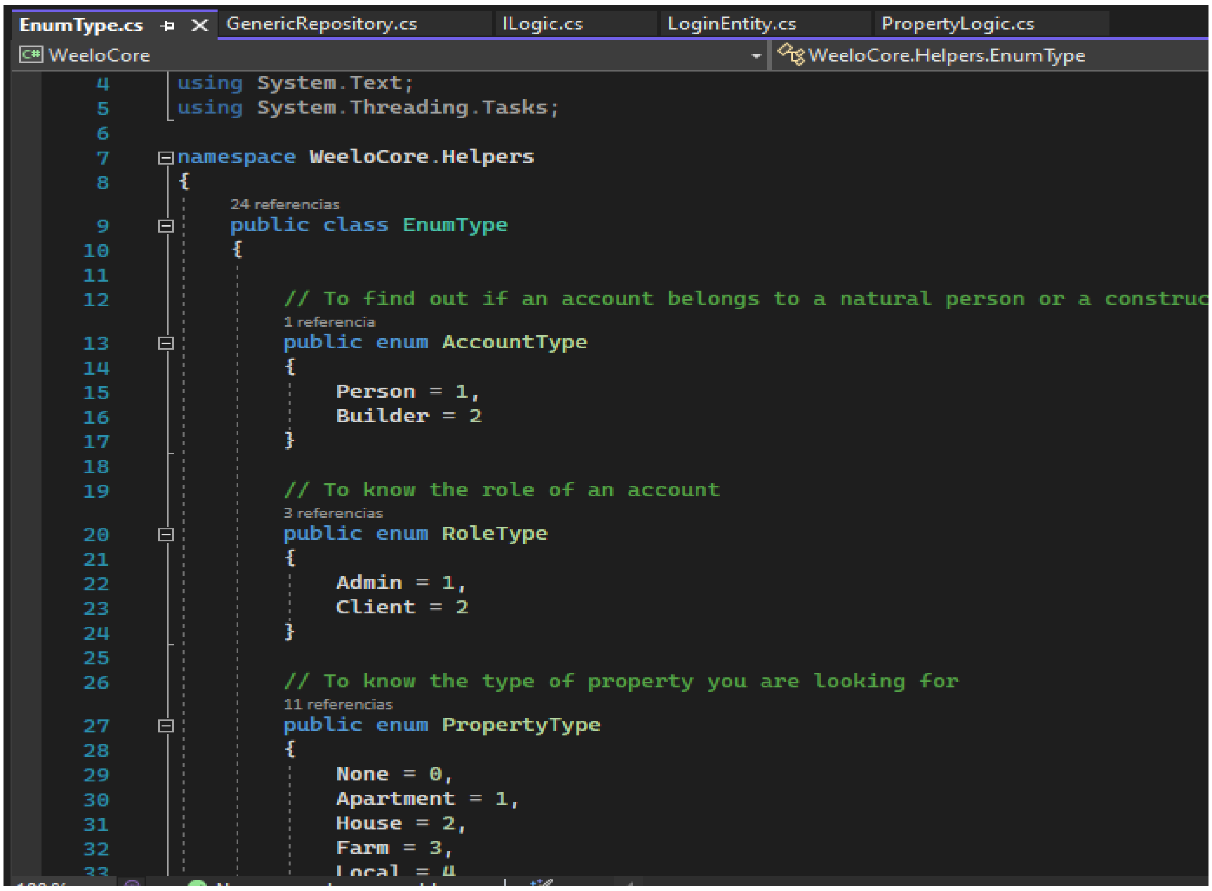


```
namespace WeeloCore.Logic
{
    //This interface class to inherit methods from basic to logic classes.
    8 referencias
    public interface ILogic<TDomainObject>
    {
        15 referencias
        public abstract List<TDomainObject> GetAll();
        27 referencias
        public abstract TDomainObject Get(Guid? id);
        10 referencias
        public abstract BaseResponse<TDomainObject> Insert(TDomainObject @object);
        9 referencias
        public abstract BaseResponse<TDomainObject> Delete(Guid? id);
        9 referencias
        public abstract BaseResponse<TDomainObject> Update(TDomainObject @object);
        62 referencias
        public abstract BaseResponse<TDomainObject> MessageResponse(int code, MessageType
    }
}
```



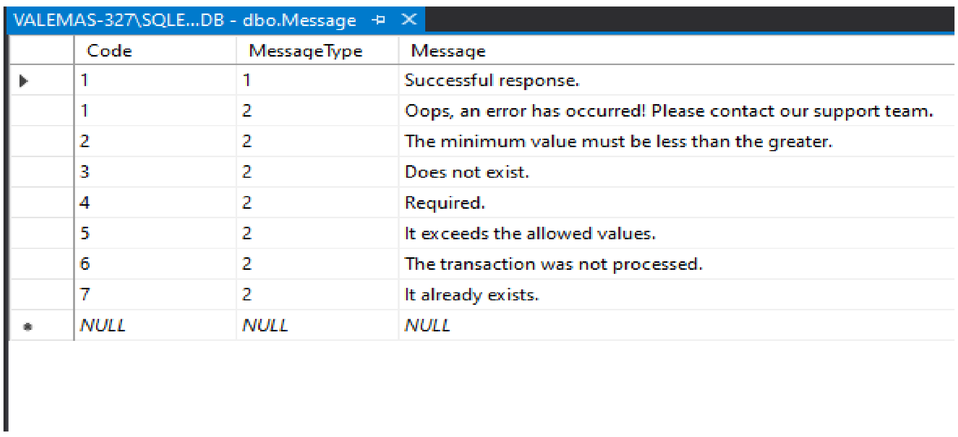
```
namespace WeeloInfrastructure.Repositories
{
    //This abstract base class for inheriting methods to repositories, and
    9 referencias
    public abstract class GenericRepository<TDomainObject>
    {
        public DataBase.WeeloDBContext weeloDBContext = new DataBase.Weelo
        24 referencias
        public abstract TDomainObject Get(Guid? id);
        16 referencias
        public abstract List<TDomainObject> GetAll();
        12 referencias
        public abstract TDomainObject Insert(TDomainObject @object);
        10 referencias
        public abstract TDomainObject Delete(Guid? id);
        10 referencias
        public abstract TDomainObject Update(TDomainObject @object);

    }
}
```

Emun management



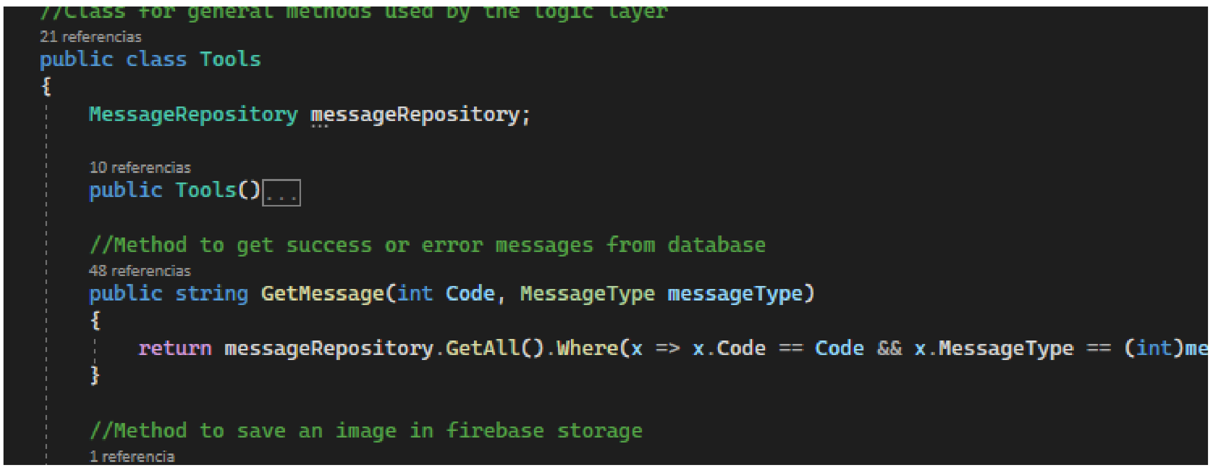
```
using System.Text;
using System.Threading.Tasks;

namespace WeeloCore.Helpers
{
    24 referencias
    public class EnumType
    {

        // To find out if an account belongs to a natural person or a construc
        1 referencia
        public enum AccountType
        {
            Person = 1,
            Builder = 2
        }

        // To know the role of an account
        3 referencias
        public enum RoleType
        {
            Admin = 1,
            Client = 2
        }

        // To know the type of property you are looking for
        11 referencias
        public enum PropertyType
        {
            None = 0,
            Apartment = 1,
            House = 2,
            Farm = 3,
            Local = 4
```

Handling error messages based on



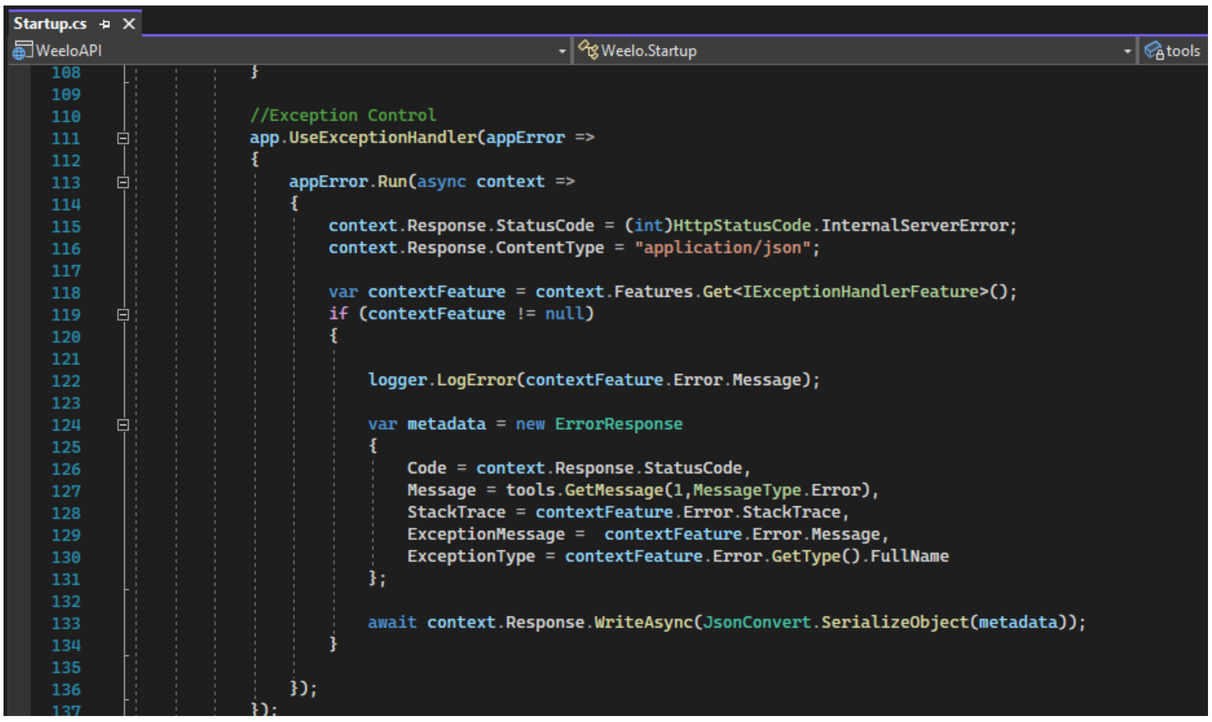
VALEMAS-327\SQLE...DB - dbo.Message

Code	MessageType	Message
1	1	Successful response.
1	2	Oops, an error has occurred! Please contact our support team.
2	2	The minimum value must be less than the greater.
3	2	Does not exist.
4	2	Required.
5	2	It exceeds the allowed values.
6	2	The transaction was not processed.
7	2	It already exists.
NULL	NULL	NULL



```
//Class for general methods used by the logic layer
21 referencias
public class Tools
{
    MessageRepository messageRepository;

    10 referencias
    public Tools()[...]

    //Method to get success or error messages from database
    48 referencias
    public string GetMessage(int Code, MessageType messageType)
    {
        return messageRepository.GetAll().Where(x => x.Code == Code && x.MessageType == (int)me
    }

    //Method to save an image in firebase storage
    1 referencia
```

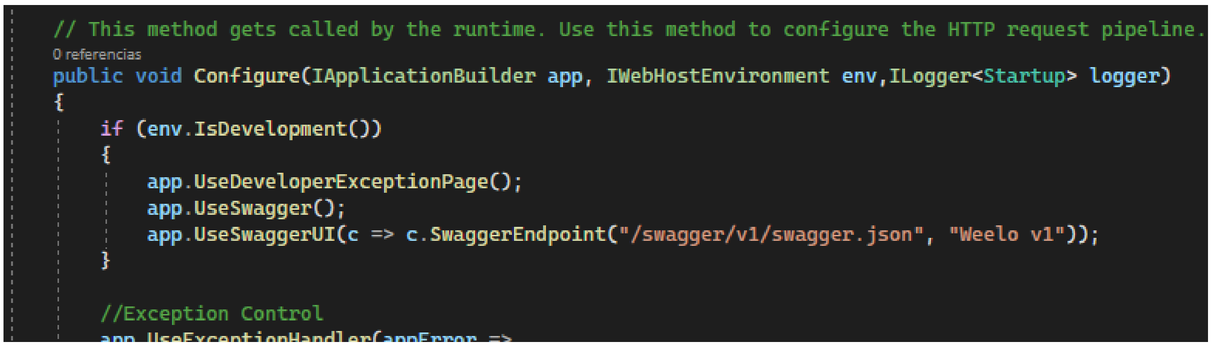
Exceptions global Exception Handler (In ASP.NET Core)



Startup.cs — WeeloAPI — Weelo.Startup — tools

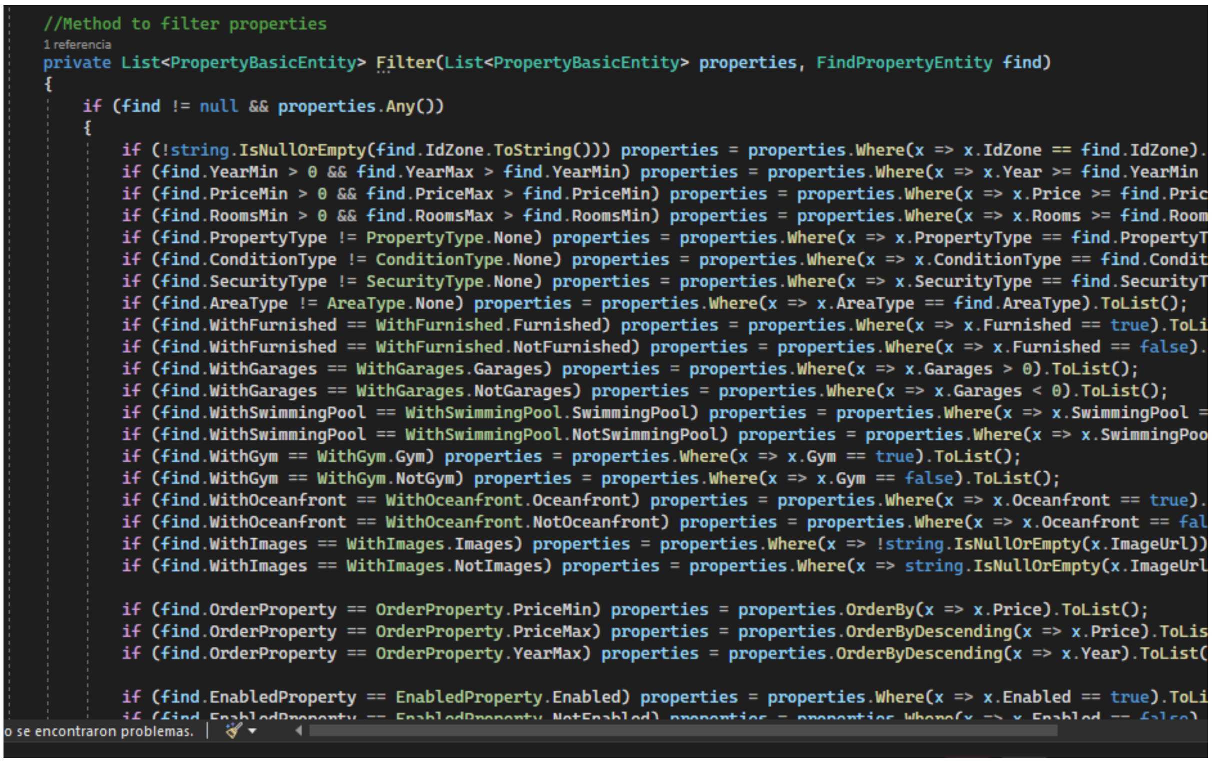
```
                }

                //Exception Control
                app.UseExceptionHandler(appError =>
                {
                    appError.Run(async context =>
                    {
                        context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;
                        context.Response.ContentType = "application/json";

                        var contextFeature = context.Features.Get<IExceptionHandlerFeature>();
                        if (contextFeature != null)
                        {

                            logger.LogError(contextFeature.Error.Message);

                            var metadata = new ErrorResponse
                            {
                                Code = context.Response.StatusCode,
                                Message = tools.GetMessage(1,MessageType.Error),
                                StackTrace = contextFeature.Error.StackTrace,
                                ExceptionMessage =  contextFeature.Error.Message,
                                ExceptionType = contextFeature.Error.GetType().FullName
                            };

                            await context.Response.WriteAsync(JsonConvert.SerializeObject(metadata));
                        }

                    });
                });
```

Exceptions use Developer Exception Page



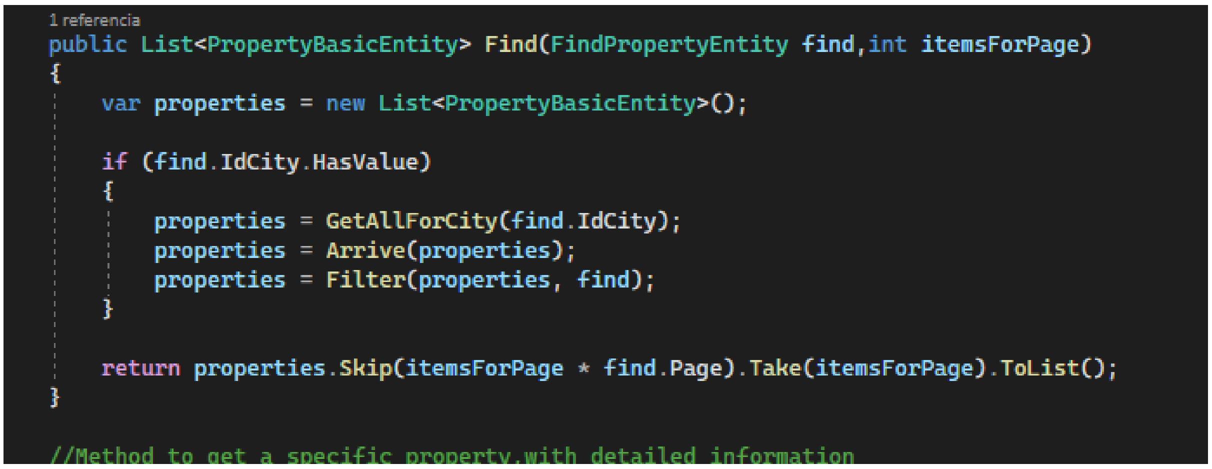
```
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
0 referencias
public void Configure(IApplicationBuilder app, IWebHostEnvironment env,ILogger<Startup> logger)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
        app.UseSwagger();
        app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "Weelo v1"));
    }

    //Exception Control
    app.UseExceptionHandler(appError =>
```

Sort and filter searches



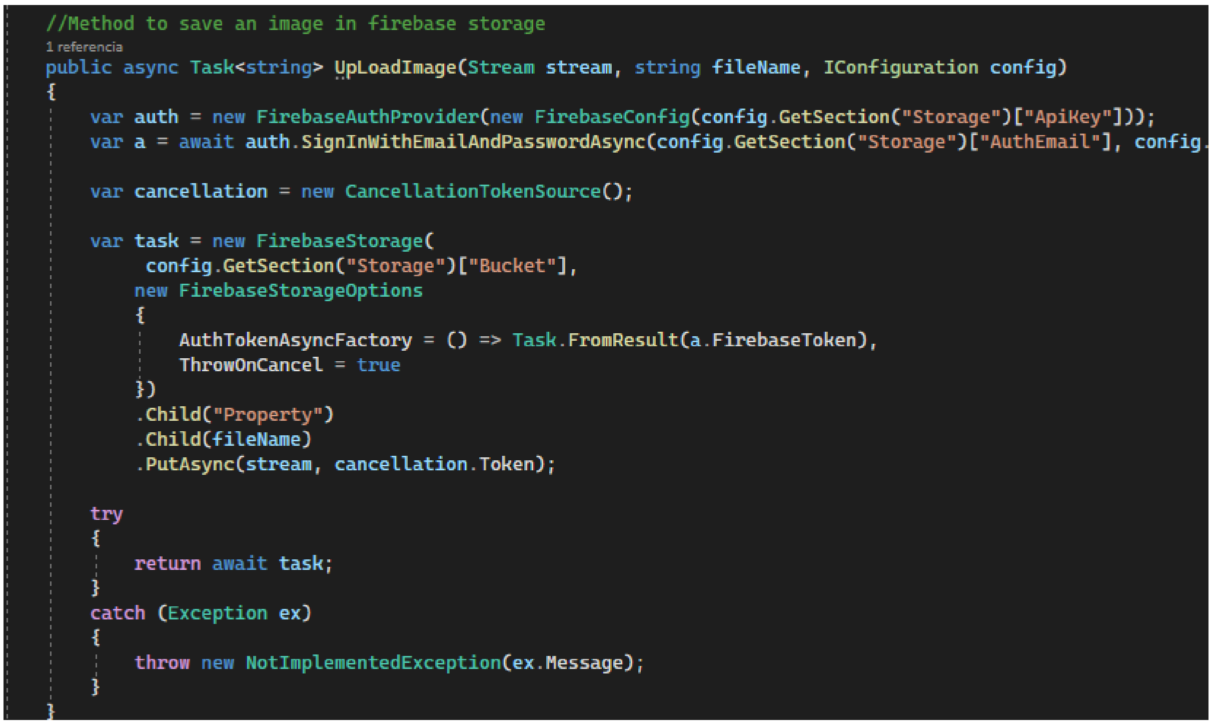
```
//Method to filter properties
1 referencia
private List<PropertyBasicEntity> Filter(List<PropertyBasicEntity> properties, FindPropertyEntity find)
{
    if (find != null && properties.Any())
    {
        if (!string.IsNullOrEmpty(find.IdZone.ToString())) properties = properties.Where(x => x.IdZone == find.IdZone).
        if (find.YearMin > 0 && find.YearMax > find.YearMin) properties = properties.Where(x => x.Year >= find.YearMin
        if (find.PriceMin > 0 && find.PriceMax > find.PriceMin) properties = properties.Where(x => x.Price >= find.Pric
        if (find.RoomsMin > 0 && find.RoomsMax > find.RoomsMin) properties = properties.Where(x => x.Rooms >= find.Room
        if (find.PropertyType != PropertyType.None) properties = properties.Where(x => x.PropertyType == find.PropertyT
        if (find.ConditionType != ConditionType.None) properties = properties.Where(x => x.ConditionType == find.Condit
        if (find.SecurityType != SecurityType.None) properties = properties.Where(x => x.SecurityType == find.SecurityT
        if (find.AreaType != AreaType.None) properties = properties.Where(x => x.AreaType == find.AreaType).ToList();
        if (find.WithFurnished == WithFurnished.Furnished) properties = properties.Where(x => x.Furnished == true).ToLi
        if (find.WithFurnished == WithFurnished.NotFurnished) properties = properties.Where(x => x.Furnished == false).
        if (find.WithGarages == WithGarages.Garages) properties = properties.Where(x => x.Garages > 0).ToList();
        if (find.WithGarages == WithGarages.NotGarages) properties = properties.Where(x => x.Garages < 0).ToList();
        if (find.WithSwimmingPool == WithSwimmingPool.SwimmingPool) properties = properties.Where(x => x.SwimmingPool =
        if (find.WithSwimmingPool == WithSwimmingPool.NotSwimmingPool) properties = properties.Where(x => x.SwimmingPoo
        if (find.WithGym == WithGym.Gym) properties = properties.Where(x => x.Gym == true).ToList();
        if (find.WithGym == WithGym.NotGym) properties = properties.Where(x => x.Gym == false).ToList();
        if (find.WithOceanfront == WithOceanfront.Oceanfront) properties = properties.Where(x => x.Oceanfront == true).
        if (find.WithOceanfront == WithOceanfront.NotOceanfront) properties = properties.Where(x => x.Oceanfront == fal
        if (find.WithImages == WithImages.Images) properties = properties.Where(x => !string.IsNullOrEmpty(x.ImageUrl))
        if (find.WithImages == WithImages.NotImages) properties = properties.Where(x => string.IsNullOrEmpty(x.ImageUrl

        if (find.OrderProperty == OrderProperty.PriceMin) properties = properties.OrderBy(x => x.Price).ToList();
        if (find.OrderProperty == OrderProperty.PriceMax) properties = properties.OrderByDescending(x => x.Price).ToLis
        if (find.OrderProperty == OrderProperty.YearMax) properties = properties.OrderByDescending(x => x.Year).ToList(

        if (find.EnabledProperty == EnabledProperty.Enabled) properties = properties.Where(x => x.Enabled == true).ToLi
        if (find.EnabledProperty == EnabledProperty.NotEnabled) properties = properties.Where(x => x.Enabled == false).
o se encontraron problemas.
```

Paginated for searches

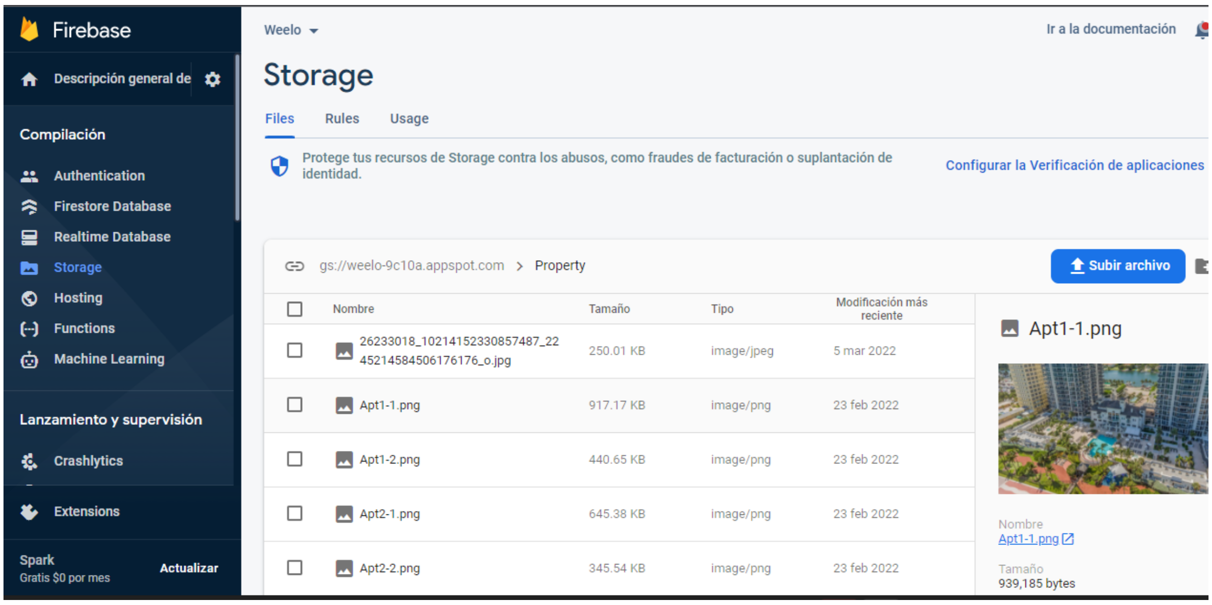


```
1 referencia
public List<PropertyBasicEntity> Find(FindPropertyEntity find,int itemsForPage)
{
    var properties = new List<PropertyBasicEntity>();

    if (find.IdCity.HasValue)
    {
        properties = GetAllForCity(find.IdCity);
        properties = Arrive(properties);
        properties = Filter(properties, find);
    }

    return properties.Skip(itemsForPage * find.Page).Take(itemsForPage).ToList();
}

//Method to get a specific property with detailed information
```

To save images fire storage



```
//Method to save an image in firebase storage
1 referencia
public async Task<string> UpLoadImage(Stream stream, string fileName, IConfiguration config)
{
    var auth = new FirebaseAuthProvider(new FirebaseConfig(config.GetSection("Storage")["ApiKey"]));
    var a = await auth.SignInWithEmailAndPasswordAsync(config.GetSection("Storage")["AuthEmail"], config.

    var cancellation = new CancellationTokenSource();

    var task = new FirebaseStorage(
         config.GetSection("Storage")["Bucket"],
        new FirebaseStorageOptions
        {
            AuthTokenAsyncFactory = () => Task.FromResult(a.FirebaseToken),
            ThrowOnCancel = true
        })
        .Child("Property")
        .Child(fileName)
        .PutAsync(stream, cancellation.Token);

    try
    {
        return await task;
    }
    catch (Exception ex)
    {
        throw new NotImplementedException(ex.Message);
    }
}
```



Storage

Protege tus recursos de Storage contra los abusos, como fraudes de facturación o suplantación de identidad.

gs://weelo-9c10a.appspot.com > Property

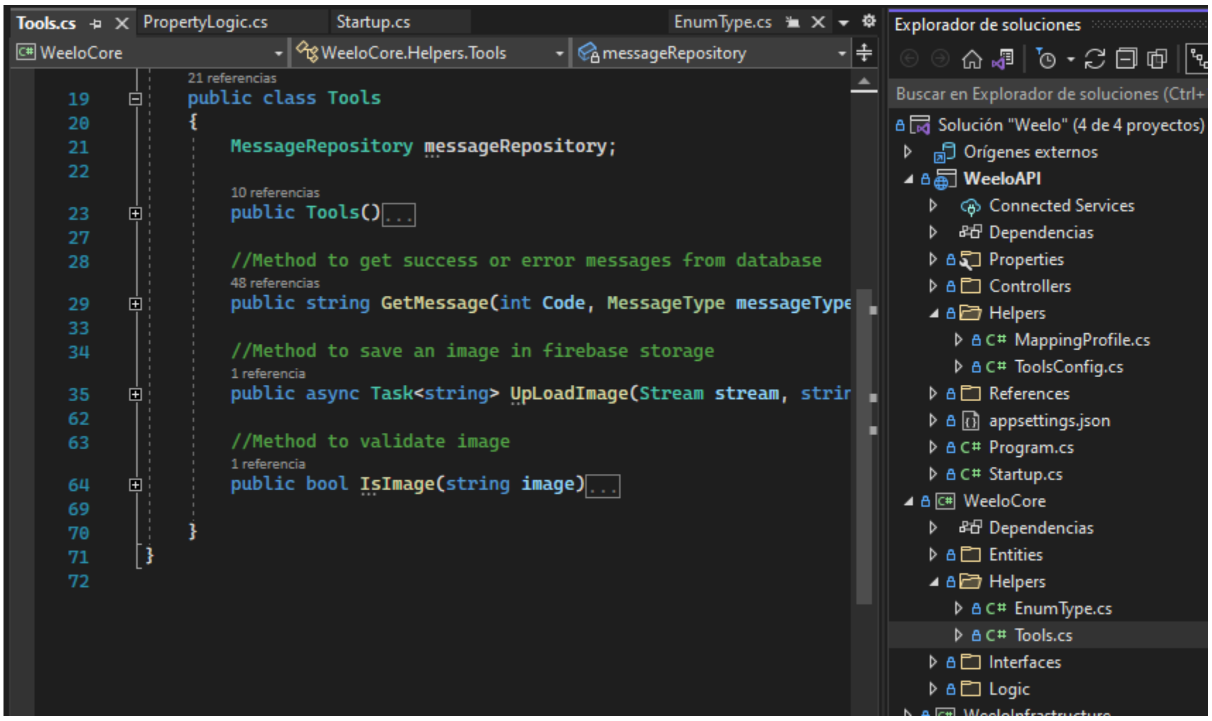
Nombre	Tamaño	Tipo	Modificación más reciente
26233018_10214152330857487_2245214584506176176_o.jpg	250.01 KB	image/jpeg	5 mar 2022
Apt1-1.png	917.17 KB	image/png	23 feb 2022
Apt1-2.png	440.65 KB	image/png	23 feb 2022
Apt2-1.png	645.38 KB	image/png	23 feb 2022
Apt2-2.png	345.54 KB	image/png	23 feb 2022

Apt1-1.png

Nombre
Apt1-1.png

Tamaño
939,185 bytes

Helper classes



Tools.cs | PropertyLogic.cs | Startup.cs | EnumType.cs

WeeloCore | WeeloCore.Helpers.Tools | messageRepository

```
    21 referencias
    public class Tools
    {
        MessageRepository messageRepository;

        10 referencias
        public Tools()[...]

        //Method to get success or error messages from database
        48 referencias
        public string GetMessage(int Code, MessageType messageType

        //Method to save an image in firebase storage
        1 referencia
        public async Task<string> UpLoadImage(Stream stream, strin

        //Method to validate image
        1 referencia
        public bool IsImage(string image)[...]

    }
}
```

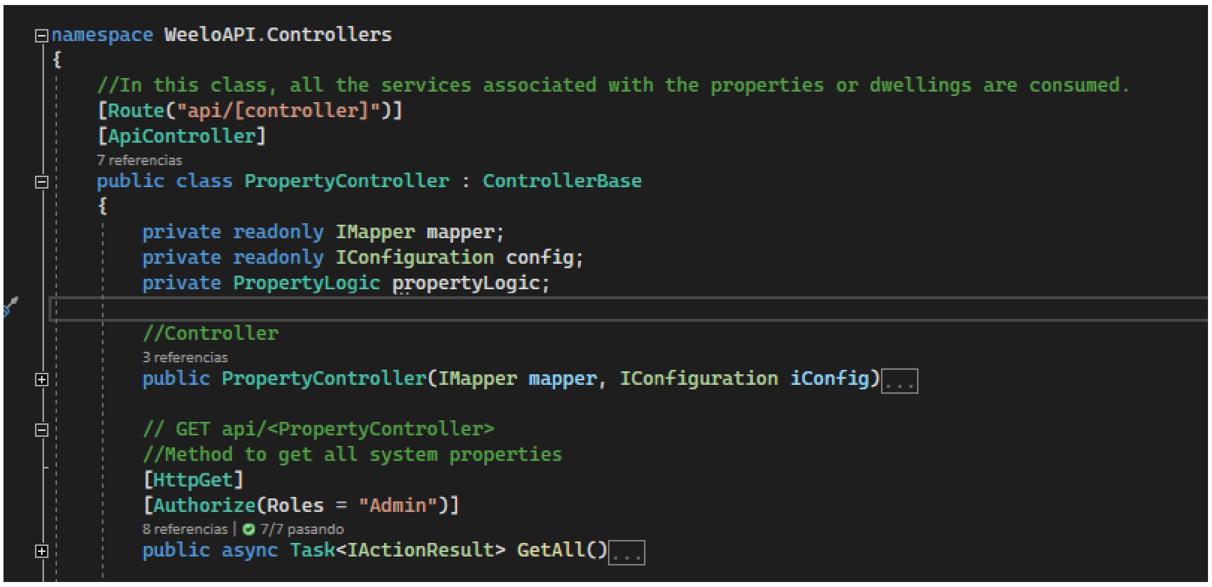
Explorador de soluciones

Buscar en Explorador de soluciones (Ctrl+

```
Solución "Weelo" (4 de 4 proyectos)
  Orígenes externos
  WeeloAPI
    Connected Services
    Dependencias
    Properties
    Controllers
    Helpers
      MappingProfile.cs
      ToolsConfig.cs
    References
    appsettings.json
    Program.cs
    Startup.cs
  WeeloCore
    Dependencias
    Entities
    Helpers
      EnumType.cs
      Tools.cs
    Interfaces
    Logic
```

Documentation Code:

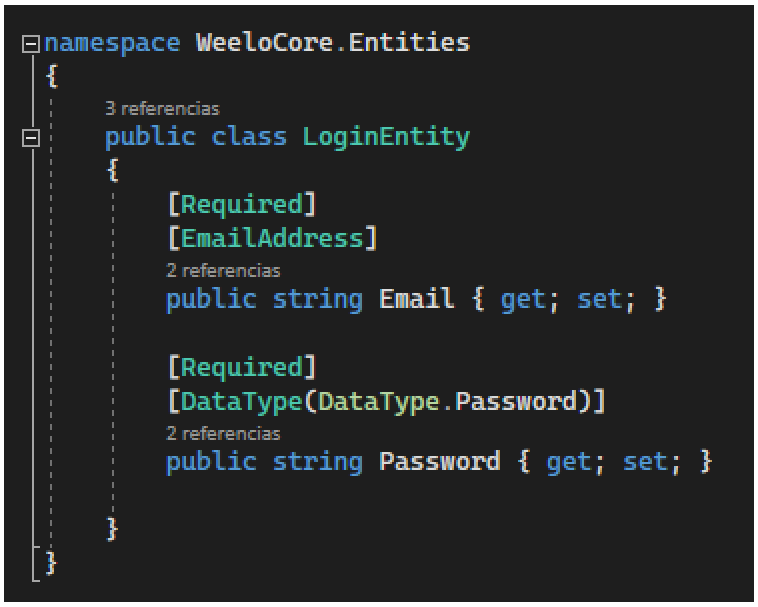
All the code is well documented, the commented classes and methods as well as the enum among others, it is explained what each of these does.



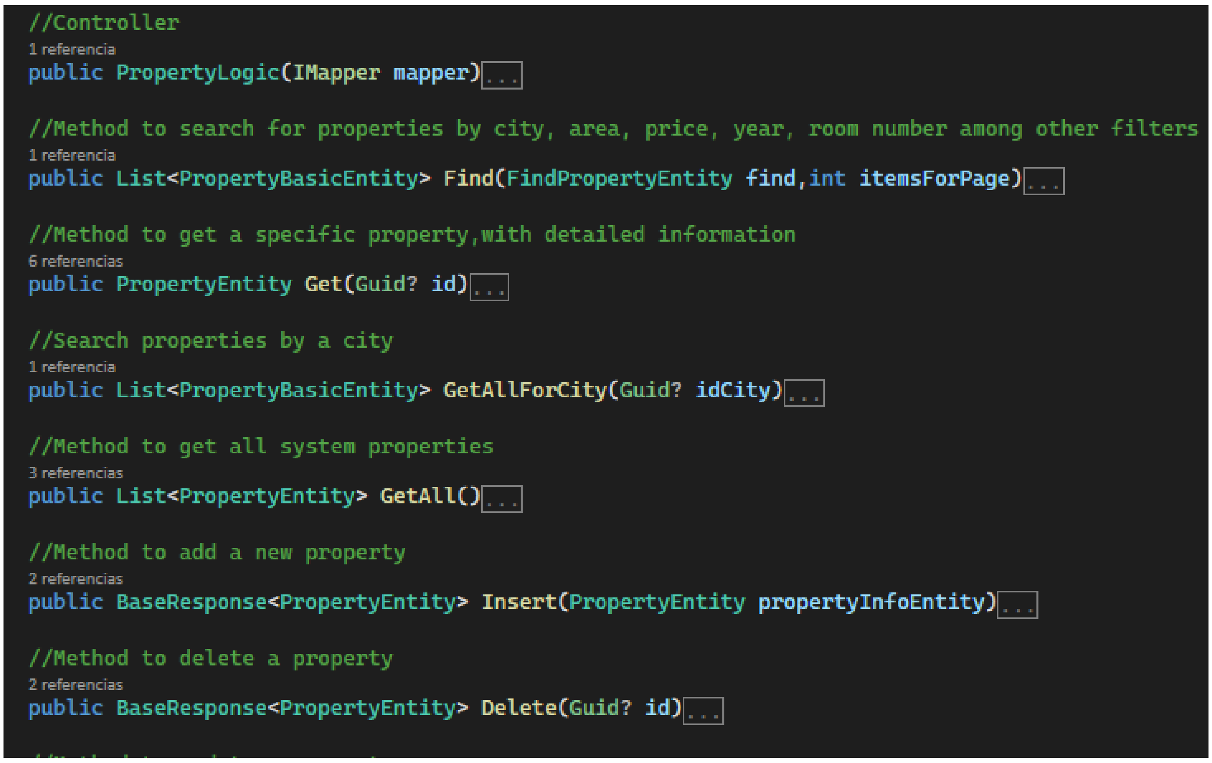
```
namespace WeeloAPI.Controllers
{
    //In this class, all the services associated with the properties or dwellings are consumed.
    [Route("api/[controller]")]
    [ApiController]
    7 referencias
    public class PropertyController : ControllerBase
    {
        private readonly IMapper mapper;
        private readonly IConfiguration config;
        private PropertyLogic propertyLogic;

        //Controller
        3 referencias
        public PropertyController(IMapper mapper, IConfiguration iConfig)[...]

        // GET api/<PropertyController>
        //Method to get all system properties
        [HttpGet]
        [Authorize(Roles = "Admin")]
        8 referencias | 7/7 pasando
        public async Task<IActionResult> GetAll()[...]
```

Good nomenclature and names of attributes, methods and classes to know what they do.

Nomenclatures: Good nomenclatures for names of classes, methods and objects



```
namespace WeeloCore.Entities
{
    3 referencias
    public class LoginEntity
    {
        [Required]
        [EmailAddress]
        2 referencias
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        2 referencias
        public string Password { get; set; }

    }
}
```

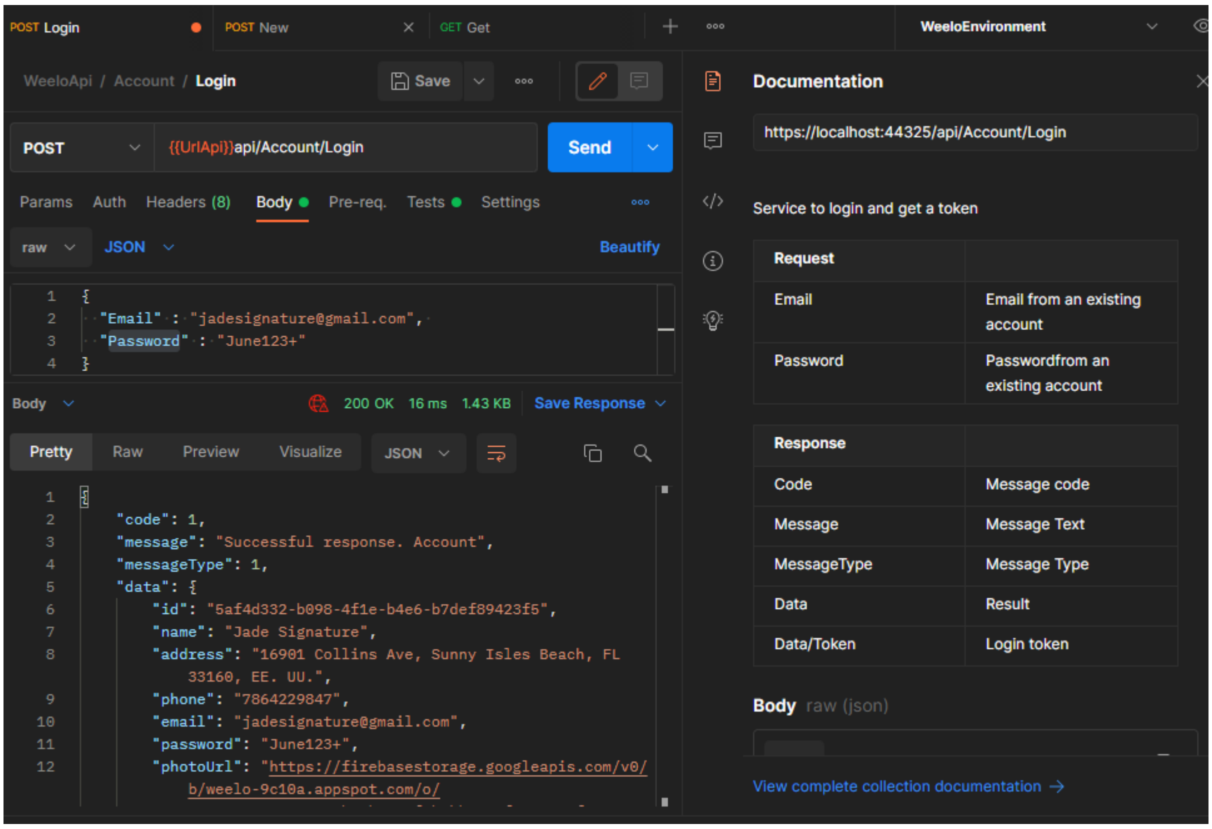


```
//Controller
1 referencia
public PropertyLogic(IMapper mapper)...

//Method to search for properties by city, area, price, year, room number among other filters
1 referencia
public List<PropertyBasicEntity> Find(FindPropertyEntity find,int itemsForPage)...

//Method to get a specific property,with detailed information
6 referencias
public PropertyEntity Get(Guid? id)...

//Search properties by a city
1 referencia
public List<PropertyBasicEntity> GetAllForCity(Guid? idCity)...

//Method to get all system properties
3 referencias
public List<PropertyEntity> GetAll()...

//Method to add a new property
2 referencias
public BaseResponse<PropertyEntity> Insert(PropertyEntity propertyInfoEntity)...

//Method to delete a property
2 referencias
public BaseResponse<PropertyEntity> Delete(Guid? id)...
```

In addition, a delivery manual is left for the configuration and management of the API.

Some of the 22 API services:

Login



{{UrlApi}}api/Account/Login

```
{
  "Email" : "jadesignature@gmail.com",
  "Password" : "June123+"
}
```

200 OK 16 ms 1.43 KB

```
{
    "code": 1,
    "message": "Successful response. Account",
    "messageType": 1,
    "data": {
        "id": "5af4d332-b098-4f1e-b4e6-b7def89423f5",
        "name": "Jade Signature",
        "address": "16901 Collins Ave, Sunny Isles Beach, FL
            33160, EE. UU.",
        "phone": "7864229847",
        "email": "jadesignature@gmail.com",
        "password": "June123+",
        "photoUrl": "https://firebasestorage.googleapis.com/v0/
            b/weelo-9c10a.appspot.com/o/
```

https://localhost:44325/api/Account/Login

Service to login and get a token

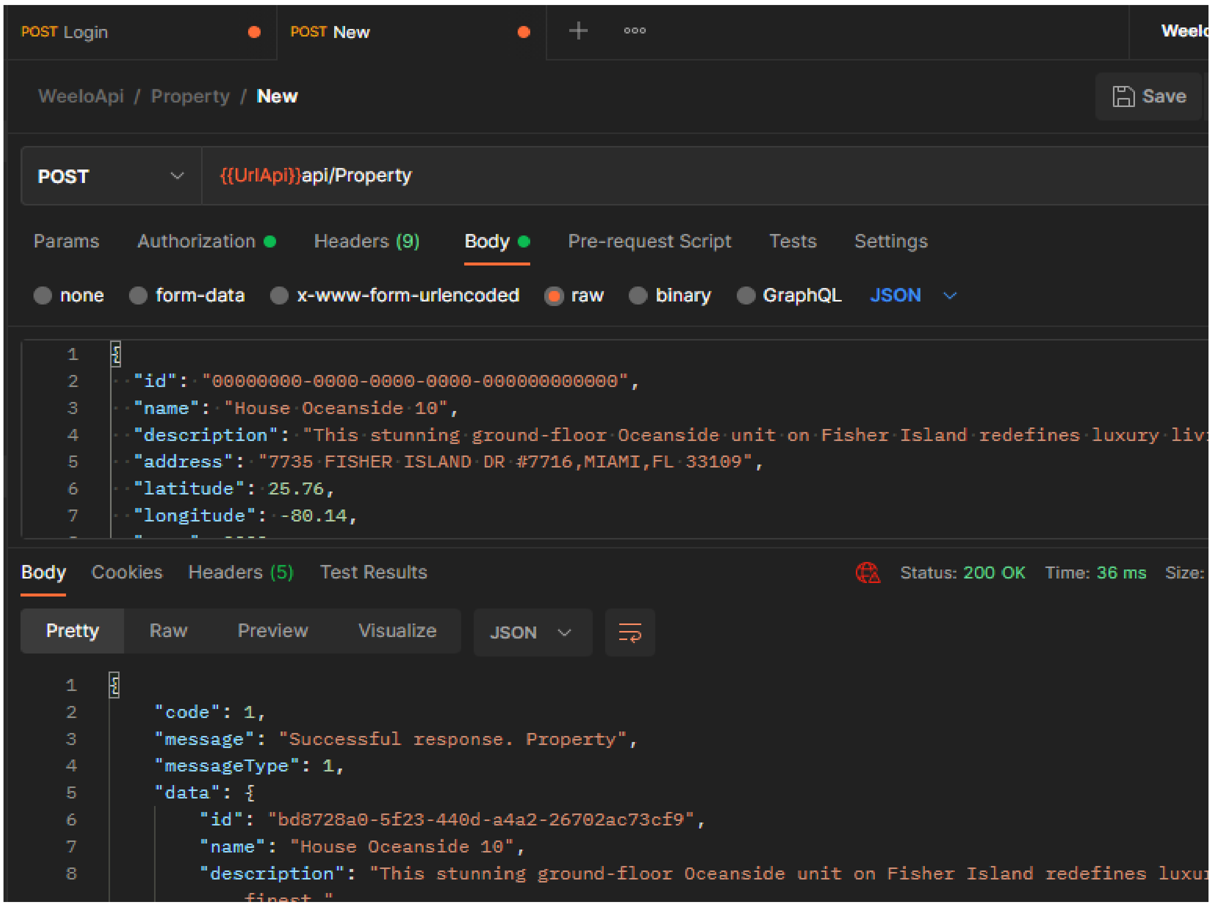
Request	
Email	Email from an existing account
Password	Passwordfrom an existing account

Response	
Code	Message code
Message	Message Text
MessageType	Message Type
Data	Result
Data/Token	Login token

Body raw (json)

View complete collection documentation →

Create Property Building



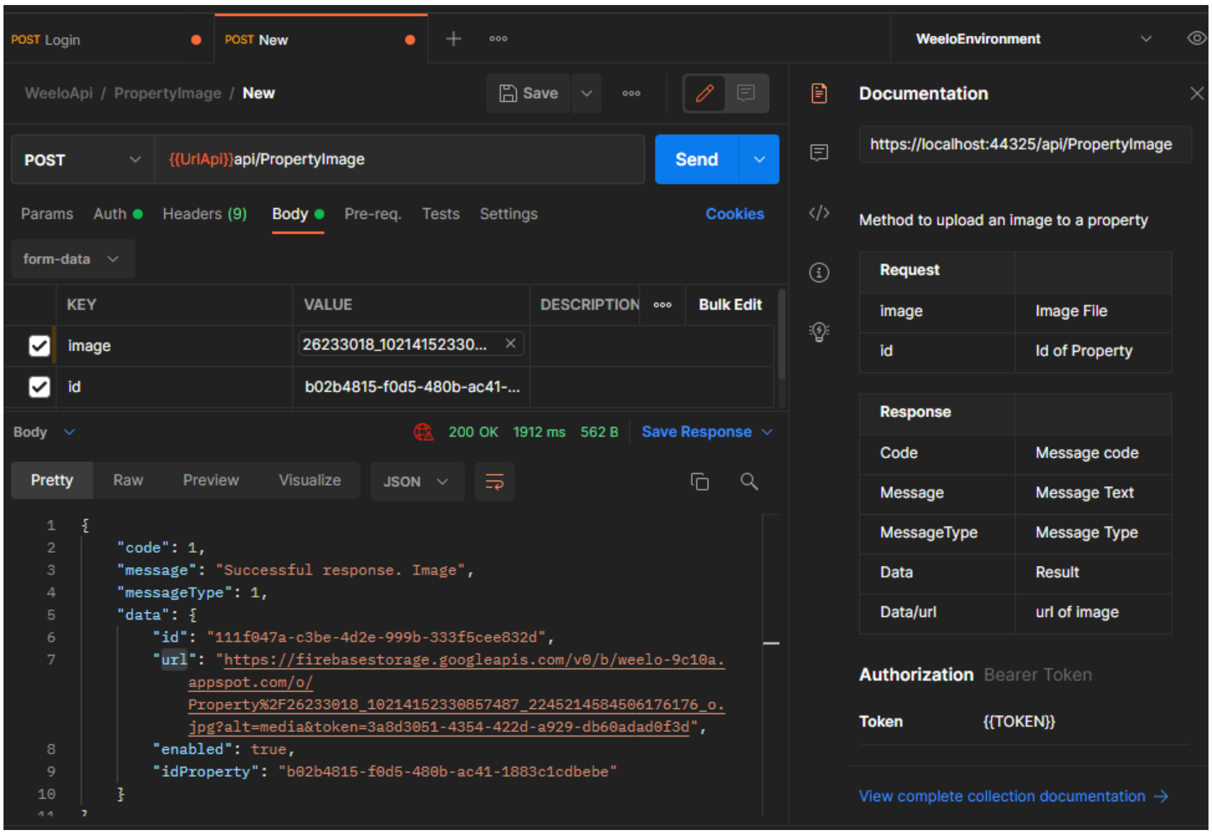
{{UrlApi}}api/Property

```
{
  "id": "00000000-0000-0000-0000-000000000000",
  "name": "House Oceanside 10",
  "description": "This stunning ground-floor Oceanside unit on Fisher Island redefines luxury liv
  "address": "7735 FISHER ISLAND DR #7716,MIAMI,FL 33109",
  "latitude": 25.76,
  "longitude": -80.14,
```

Status: 200 OK Time: 36 ms Size:

```
{
    "code": 1,
    "message": "Successful response. Property",
    "messageType": 1,
    "data": {
        "id": "bd8728a0-5f23-440d-a4a2-26702ac73cf9",
        "name": "House Oceanside 10",
        "description": "This stunning ground-floor Oceanside unit on Fisher Island redefines luxu
            finest."
```

Add Image from property


POST Login | POST New | WeeloEnvironment

WeeloApi / PropertyImage / **New**

POST {{UrlApi}}api/PropertyImage Send

Params | Auth | Headers (9) | Body | Pre-req. | Tests | Settings | Cookies

form-data

	KEY	VALUE	DESCRIPTION
✓	image	26233018_10214152330...	
✓	id	b02b4815-f0d5-480b-ac41-...	

Body — 200 OK 1912 ms 562 B — Save Response

Pretty | Raw | Preview | Visualize | JSON

```
{
    "code": 1,
    "message": "Successful response. Image",
    "messageType": 1,
    "data": {
        "id": "111f047a-c3be-4d2e-999b-333f5cee832d",
        "url": "https://firebasestorage.googleapis.com/v0/b/weelo-9c10a.
            appspot.com/o/
            Property%2F26233018_10214152330857487_2245214584506176176_o.
            jpg?alt=media&token=3a8d3051-4354-422d-a929-db60adad0f3d",
        "enabled": true,
        "idProperty": "b02b4815-f0d5-480b-ac41-1883c1cdbebe"
    }
```

Documentation

https://localhost:44325/api/PropertyImage

Method to upload an image to a property

Request	
image	Image File
id	Id of Property

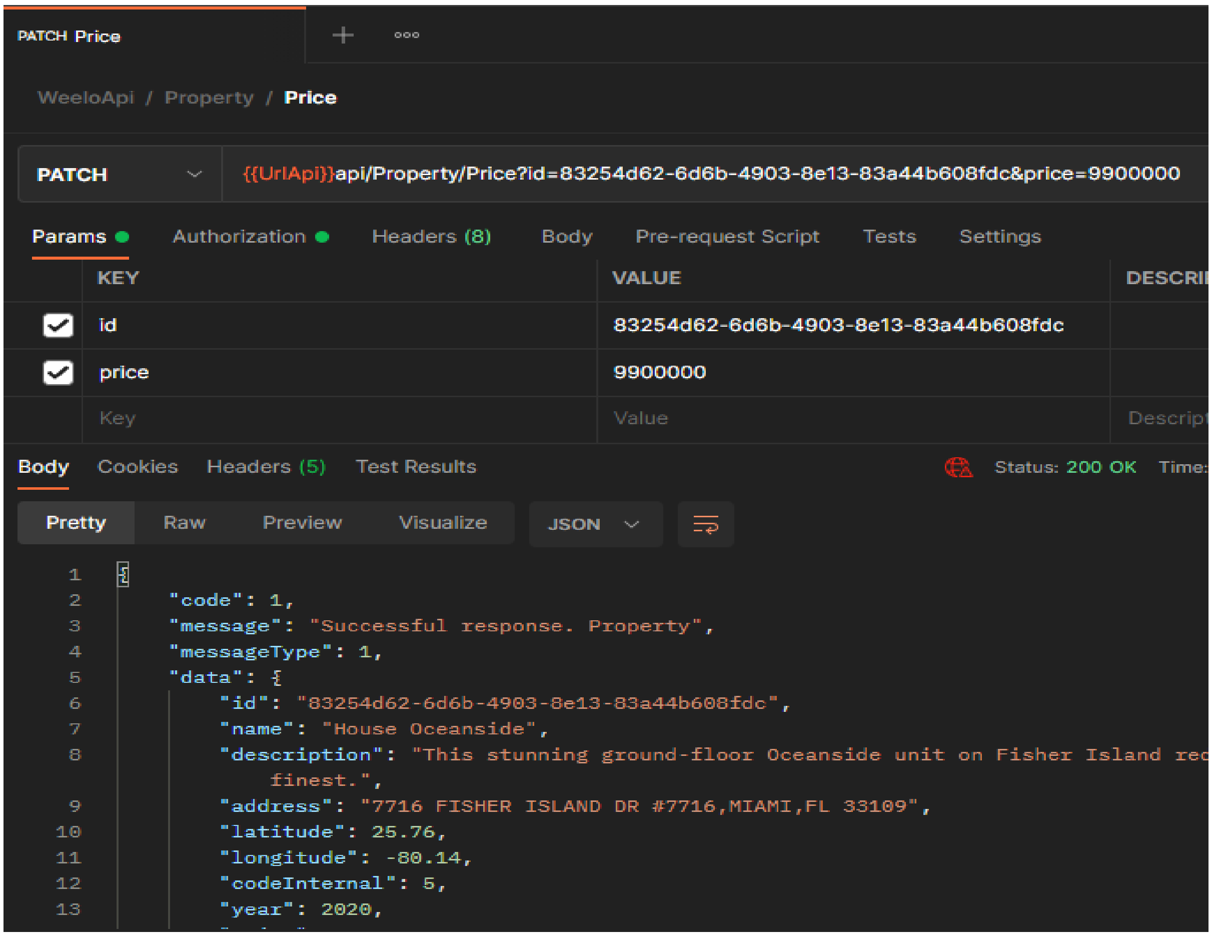
Response	
Code	Message code
Message	Message Text
MessageType	Message Type
Data	Result
Data/url	url of image

Authorization Bearer Token

Token	{{TOKEN}}

View complete collection documentation →

Change Price


PATCH Price

WeeloApi / Property / **Price**

PATCH {{UrlApi}}api/Property/Price?id=83254d62-6d6b-4903-8e13-83a44b608fdc&price=9900000

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

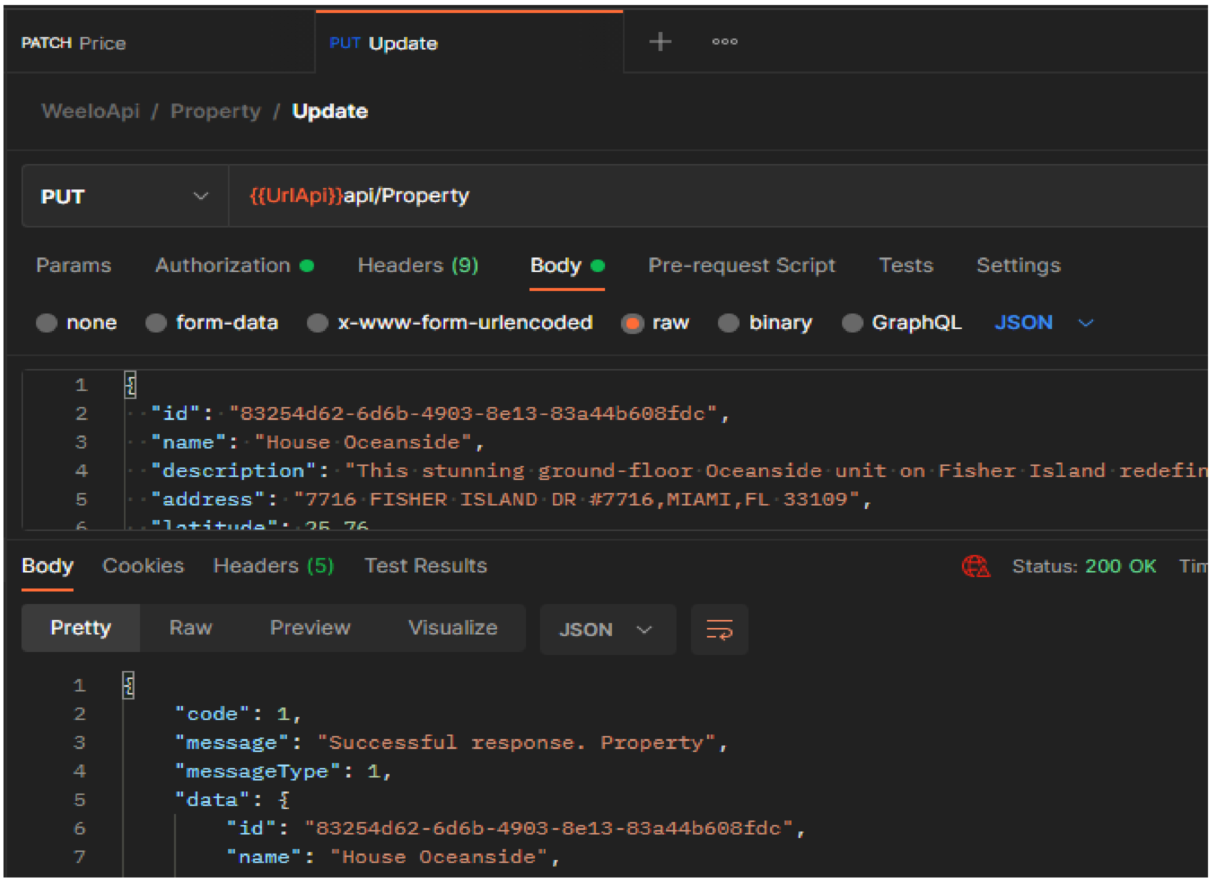
	KEY	VALUE	DESCRI
✓	id	83254d62-6d6b-4903-8e13-83a44b608fdc	
✓	price	9900000	
	Key	Value	Descript

Body | Cookies | Headers (5) | Test Results — Status: 200 OK Time:

Pretty | Raw | Preview | Visualize | JSON

```
{
    "code": 1,
    "message": "Successful response. Property",
    "messageType": 1,
    "data": {
        "id": "83254d62-6d6b-4903-8e13-83a44b608fdc",
        "name": "House Oceanside",
        "description": "This stunning ground-floor Oceanside unit on Fisher Island rec
            finest.",
        "address": "7716 FISHER ISLAND DR #7716,MIAMI,FL 33109",
        "latitude": 25.76,
        "longitude": -80.14,
        "codeInternal": 5,
        "year": 2020,
```

Update property



PUT {{UrlApi}}api/Property

Body: raw JSON

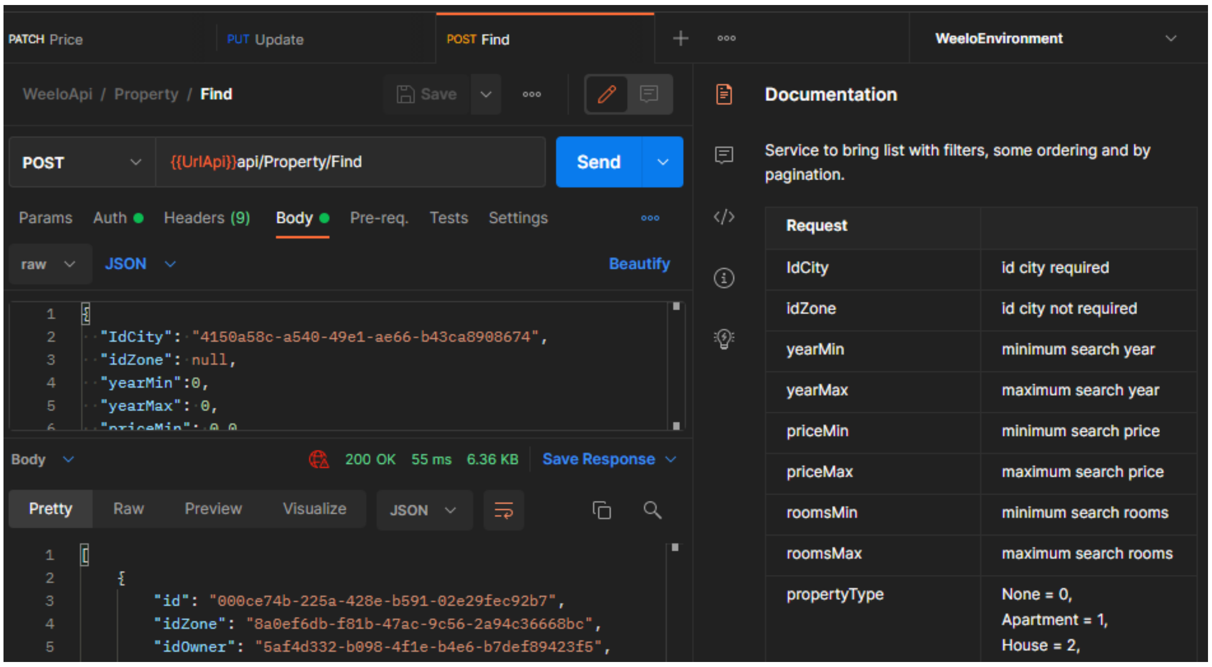
```
{
  "id": "83254d62-6d6b-4903-8e13-83a44b608fdc",
  "name": "House Oceanside",
  "description": "This stunning ground-floor Oceanside unit on Fisher Island redefin
  "address": "7716 FISHER ISLAND DR #7716,MIAMI,FL 33109",
  "latitude": 25.76
```

Status: 200 OK

Body: Pretty JSON

```
{
    "code": 1,
    "message": "Successful response. Property",
    "messageType": 1,
    "data": {
        "id": "83254d62-6d6b-4903-8e13-83a44b608fdc",
        "name": "House Oceanside",
```

List property with filters



POST {{UrlApi}}api/Property/Find

Body: raw JSON

```
{
  "IdCity": "4150a58c-a540-49e1-ae66-b43ca8908674",
  "idZone": null,
  "yearMin":0,
  "yearMax": 0,
  "priceMin": 0.0
```

200 OK 55 ms 6.36 KB

Body: Pretty JSON

```
[
    {
        "id": "000ce74b-225a-428e-b591-02e29fec92b7",
        "idZone": "8a0ef6db-f81b-47ac-9c56-2a94c36668bc",
        "idOwner": "5af4d332-b098-4f1e-b4e6-b7def89423f5",
```

Documentation

Service to bring list with filters, some ordering and by pagination.

Request	
IdCity	id city required
idZone	id city not required
yearMin	minimum search year
yearMax	maximum search year
priceMin	minimum search price
priceMax	maximum search price
roomsMin	minimum search rooms
roomsMax	maximum search rooms

propertyType	None = 0,
	Apartment = 1,
	House = 2,