|  |
| --- |
| splash |
| **IDEA ERMT APPLICATION CODE DESCRIPTION**  Division:  **Development** Product:  **IDEA V7**  Document ID: IDEA ERMT Application Code Description.doc |

**Content**

[1. Introduction 3](#_Toc46220569)

[1.1 Purpose 3](#_Toc46220570)

[1.2 Background 3](#_Toc46220571)

[2. Architectural Representation 3](#_Toc46220572)

[3. Code View 5](#_Toc46220573)

[3.1 Overview 5](#_Toc46220574)

[3.2 Solution 5](#_Toc46220575)

[3.3 Layers 5](#_Toc46220576)

[3.3.1 Idea.DAL 6](#_Toc46220577)

[3.3.2 Idea.Entities 7](#_Toc46220578)

[3.3.3 Idea.Business 8](#_Toc46220579)

[3.3.4 Idea.Facade 9](#_Toc46220580)

[3.3.5 Idea.ERMT 10](#_Toc46220581)

[3.3.6 Idea.Server 12](#_Toc46220582)

[3.4 Other Projects 13](#_Toc46220583)

[4. Think Geo Library 17](#_Toc46220584)

[5. Installation Projects 21](#_Toc46220585)

[6. Installation Guide Technical Instructions 21](#_Toc46220586)

# Introduction

## Purpose

The purpose of this document is to provide a detailed architecture design of the IDEA ERMT Solution software by focusing on those key quality attributes: usability, availability and maintainability. These attributes were chosen based on their importance in the design and construction of the application.

This document will address the architecturally significant functional requirements. The goal is to help future development teams to know the solution structure at the highest level.

## Background

The ERMTool aims to build the user’s capacity to understand, analyze, and mitigate electoral risks. It can build users' capacity to understand electoral risk factors, collect and analyze risk data, design prevention and mitigation strategies, and record the results of actions. The tool consists of three integrated parts.

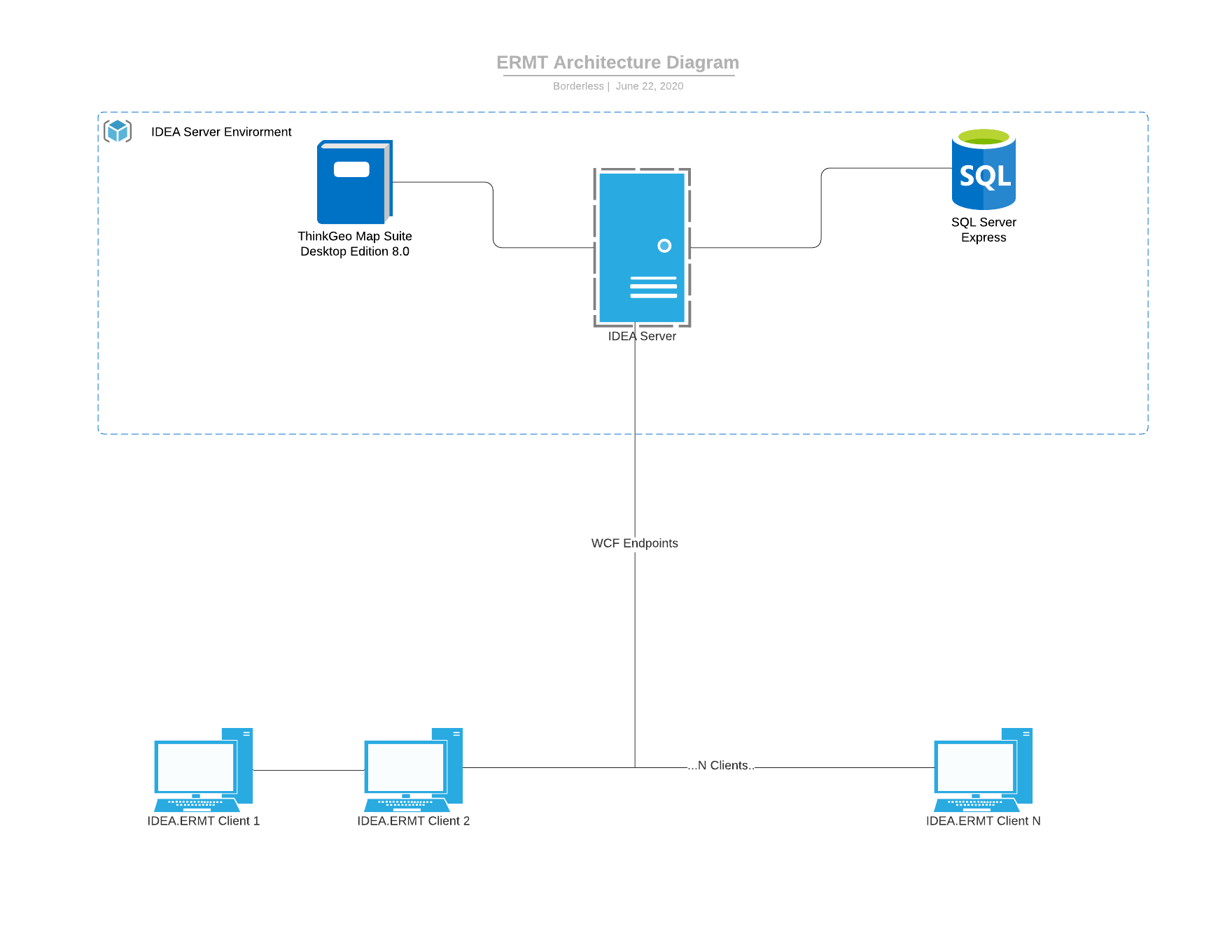
A knowledge library describes in detail 36 electoral risk factors, both internal and external to electoral processes. An analytical instruments section allows users to create analytical models specific to a country or election, upload data to generate risk maps and trend charts, and create a register of risks and actions. The prevention module consists of a digital library with approximately 100 action points intended to inspire user in designing strategies to prevent and mitigate electoral risks, including election-related violence, at the different phases of the electoral cycle.

# Architectural Representation

ERMT Tool is a client-server architecture which allows multiple clients to connect to one main server.

The tool was built with the following technology:

* NET framework (version 4.5)
* C# language
* SQL Server Express Edition version 11.0.2100.60
* ThinkGeo Map Suite Desktop Edition Product Center 8.0
* Compatible with Windows 8/10
* Windows Service



# Code View

## Overview

In this section we are going to describe how the solution was build and the layers it contains.

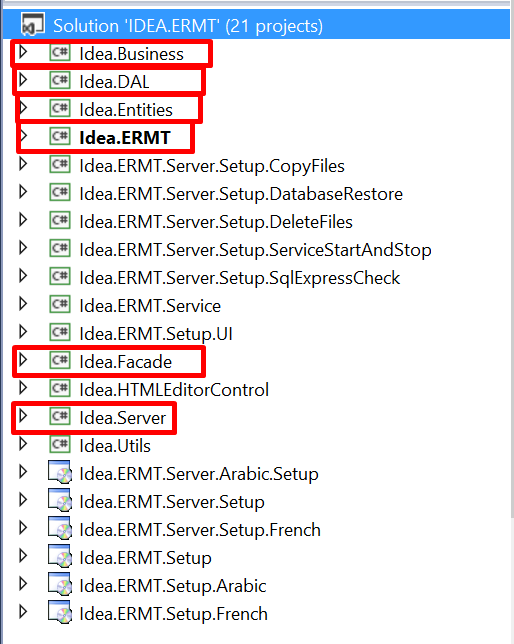
## Solution

The .NET solution has 21 projects created by the developer team with the strategy to divide all the code in different layers, bringing a robust solution, secure and traceable code.

## Layers

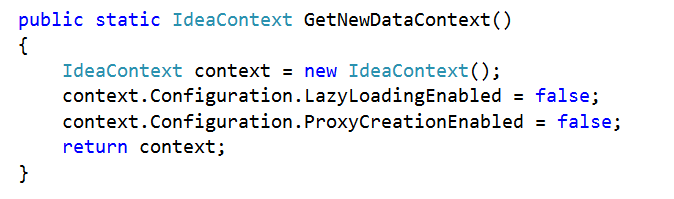
Basically the solution has 6 main layers that we can describe on this section:

* Idea.DAL
* Idea.Entities
* Idea.Business
* Idea.Facade
* Idea.ERMT
* Idea.Server



### 3.3.1 Idea.DAL

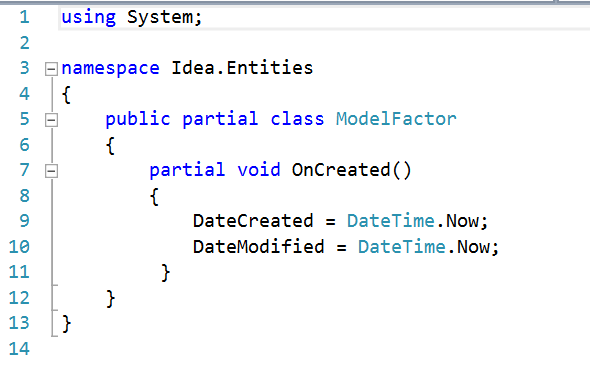
Data Access Layer project was created with the purpose of pulling data from any data store (database) separate from business logic and presentation code. It contains the direct connection to the database hosted on SQL Server Express Edition instance, using the EDMX Framework to map database tables into C# objects.



*“Initialize the data context to be used by other projects. It has been isolated layer for security propose”.*

### 3.3.2 Idea.Entities

This library contains all database objects extended from EDMX Framework ready to be used in other projects. In this case we extended the class on some of them to add some custom methods and properties which we can find under “ExtendedClasses” folder.



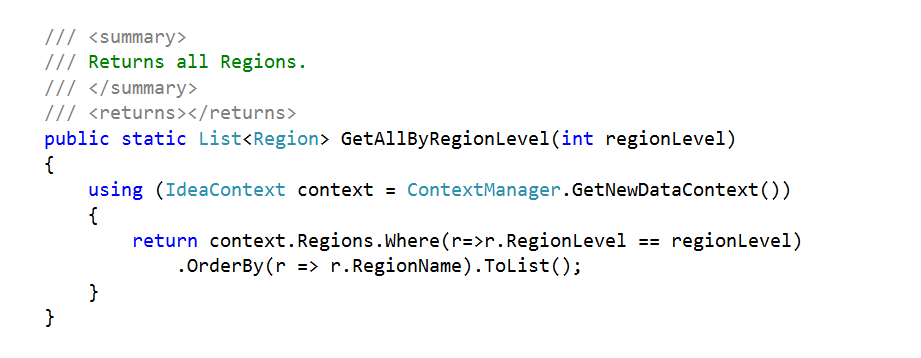
*“In this example we can see OnCreated() void added to the partial class ModelFactor to be part of the original object extracted from the EDMX.”*



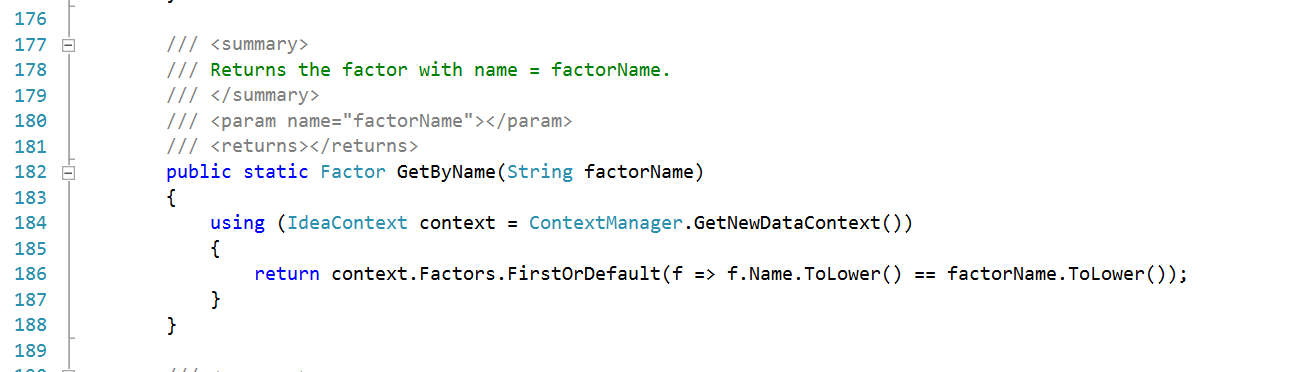
*“In this second example we can see new properties with a DataMember notation on the ModelRiskAlertAttachment class”*

### 3.3.3 Idea.Business

It is one of the core library created to host business classes to be available in all projects. Those classes basically are manager classes that have all business logic using Idea.Entities classes to work with all EDMX objects and using Idea.DAL classes to access the database.



*“In this example we can see a RegionManager.cs method that returns all Regions from EDMX Region object (mapped to Region table) where Level is the same as the level passed by parameter ordered by RegionName.”*



*“This second example shows the GetByName() method from the FactorManager class where it returns the Factor EDMX object searched by a specific name passed by parameter.”*

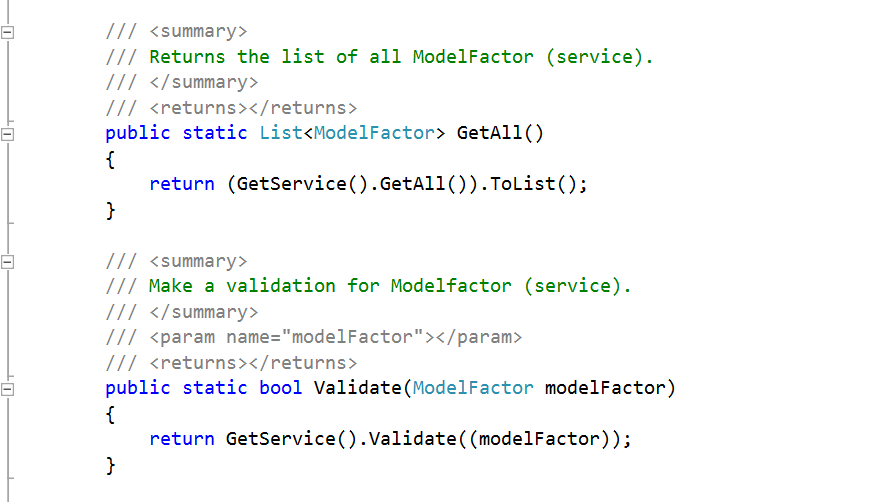
### 3.3.4 Idea.Facade

This library contains a variety of Helper classes, most of which have been created as a service correctly formatted to be received by Idea.ERMT Client. They handle all operations that are related to services, in this case WCF Services. Every service has an endpoint configured pointed to each Idea.Server exposed class.

So, in other words, Idea.Facade generates the contract between Idea.ERMT Client and Idea.Server to communicate with each other.



*“In this example we can identify the GetService() method from the class ModelFactorHelper on ModelFactorHelper.cs file. It initializes the service, leaving it ready to be used in the hole ModelFatorHelper class”*

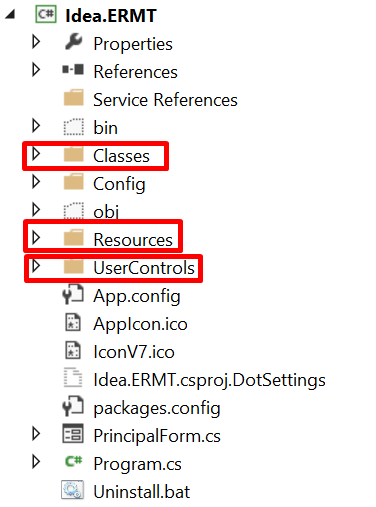


*“This other piece of code explains how GetAll() or Validate() Methods calls the GetService() described before.”*

### 3.3.5 Idea.ERMT

Idea.ERMT is the Client presentation code project which connects directly to one main Idea.Server service to consume it. It is a Windows Desktop Application Project which must be installed on every PC that is to act as a client.

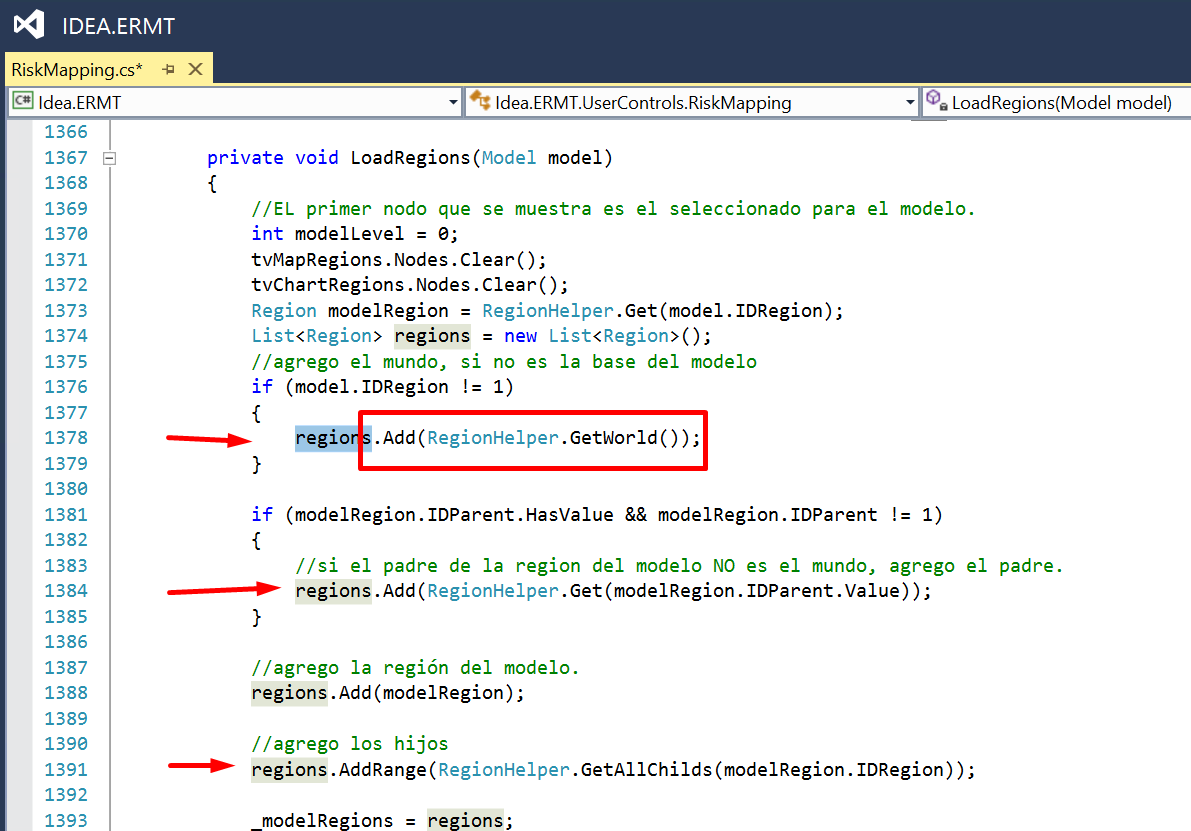
In this section we are going to describe the most important folder structure of the project.



* “Classes” folder was created to separate main context functionality from rest of the project. It contains manager classes and cache control classes.
* “Resources” folder has images to be used on the project.
* “UserControls” folder as the name indicates contains windows desktop usercontrols that are being loaded on each view of the application. It is structured with subfolders like:



As an example we can describe the principal usercontrol “RiskMapping” which reside into “Map” subfolder.

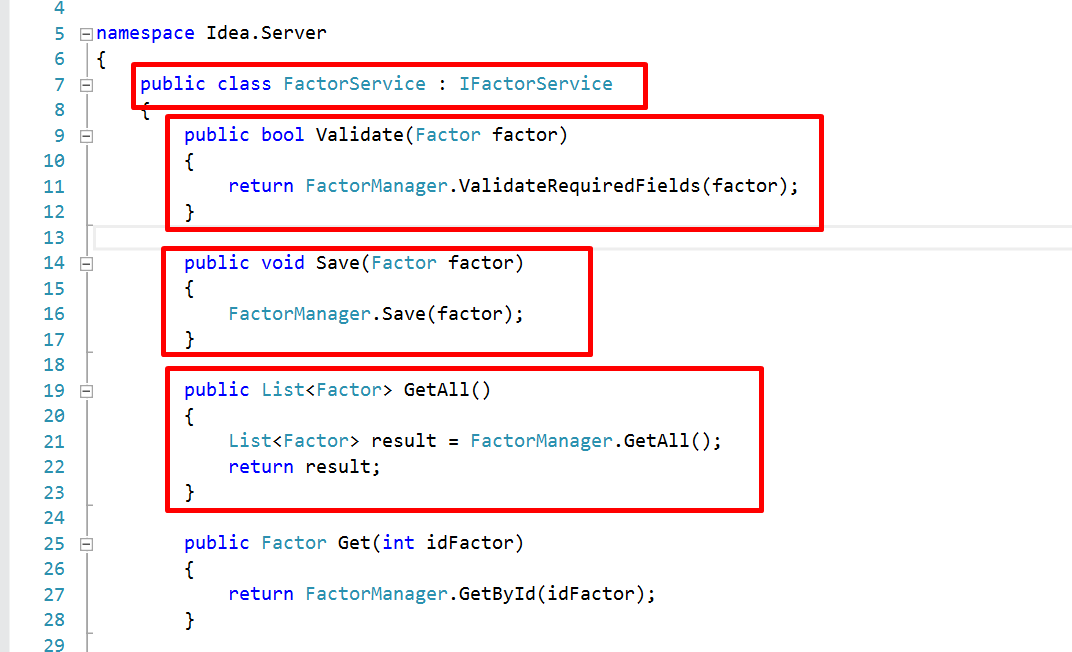


*“In this example we can see how RiskMapping is calling RegionHelper class from Idea.Facade which loads the service and returns the required result.”*

### 3.3.6 Idea.Server

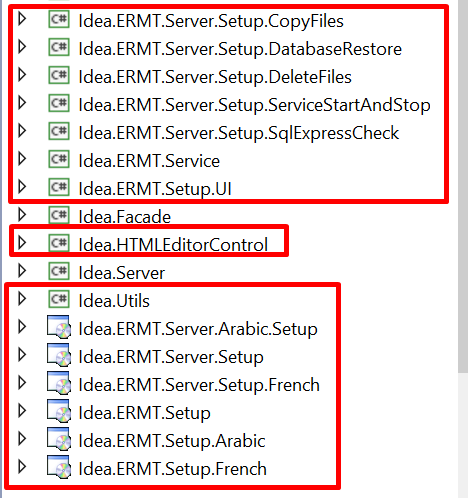
Idea.Server is the library that will accept every client that tries to connect. It will serve each client with business data to allow the client to transform it and display it in the application.

That library contains all the classes that Idea.ERMT would consume. Each class contains basic methods like CRUD, listing methods and a variety of methods and properties. Most of those classes inherit from an interface class which declare the contract definition for the WCF communication between client and server.

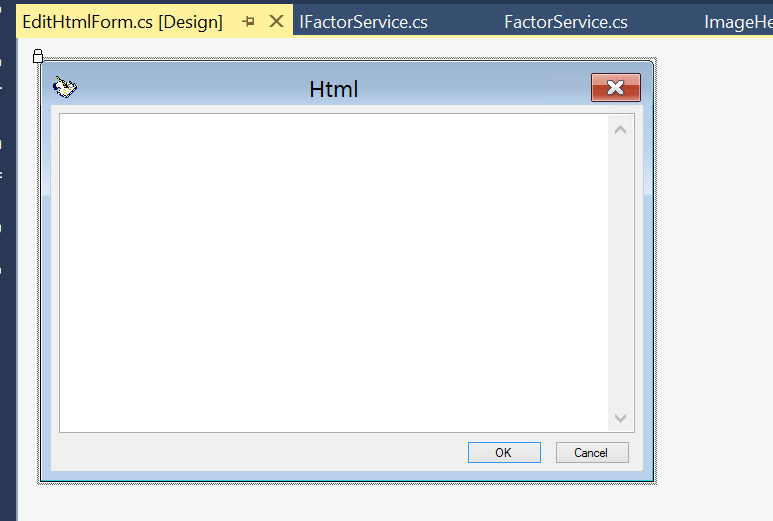


## Other Projects

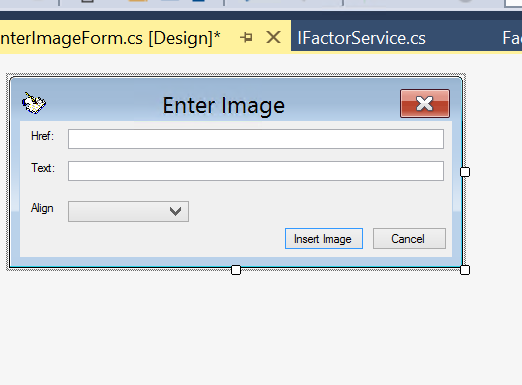
In this section we are going to describe in a general way all other projects of the solution.



* All libraries whose name starts with “Idea.ERMT.Service.Setup….” have been created for installation purposes. They will create, copy and delete files, restore databases, and install the server as a Windows Service.
* Idea.HTMLEditorControl has form controls to generate HTML tags into the project.

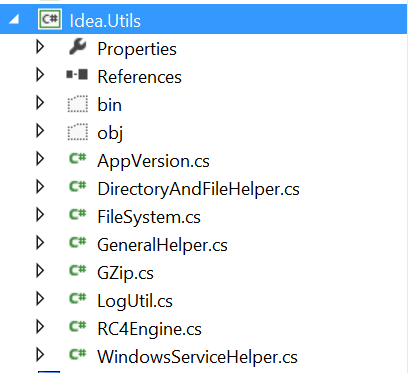


*“EditHtmlForm.cs [Design view] shows a multiline textbox configured to include HTML tags”*



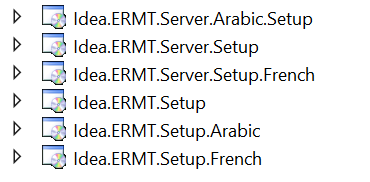
*“EnterImageForm.cs [Design view] shows a form to add properties from an HTML Image tag”*

* Idea.Utils is a library designated to contain main classes that can be used for general purpose such us WindowsServiceHeper.cs, LogUtils.cs, General Helper, etc.



The rest of the projects whose name starts with “Idea.ERMT.Server.Setup” or “Idea.ERMT.Setup.” are the main installer project builder with all database and necessary files to prepare the installation packages from the application. It can be created in different languages like “Idea.ERMT.Server.Arabic.Setup” and “Idea.ERMT.Setup.Arabic” or French one, etc.

These projects contains all region information such as database records, map shape files, help files, etc.

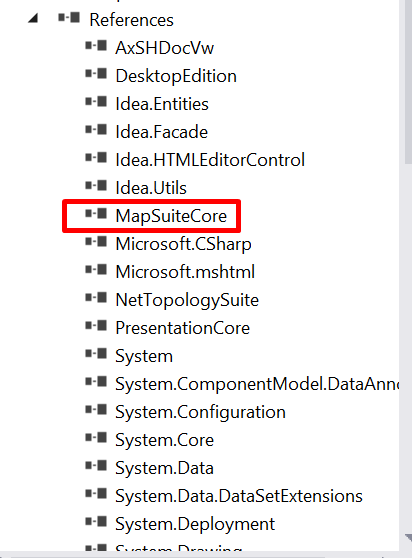


# Think Geo Library

Think Geo is one of the most important libraries used in the solution. It allows Idea.ERMT to create desktop interactives maps, add markers, layers, labels and all sort of information.

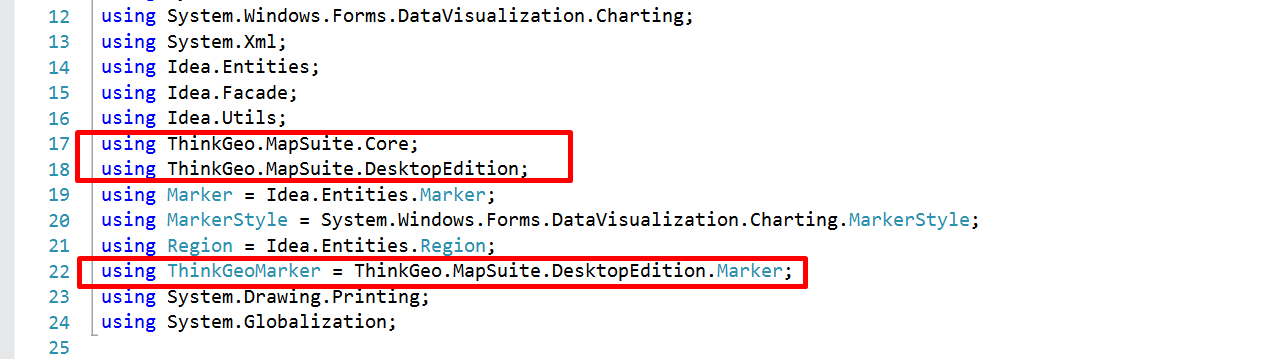
The last release that we used on the project is the *“ThinkGeo Map Suite Desktop Edition Product Center 8.0”.* For newest releases visit (<https://thinkgeo.com/>). Please note that ThinkGeo Map Suite Desktop Edition is a commercial product; the license must be separately purchased and is not part of the open source code provided by International IDEA.

To start using ThinkGeo we referenced the MapSuiteCode.dll file downloaded from the official site on the local repository. This can be achieved using the nuGet package manager also.

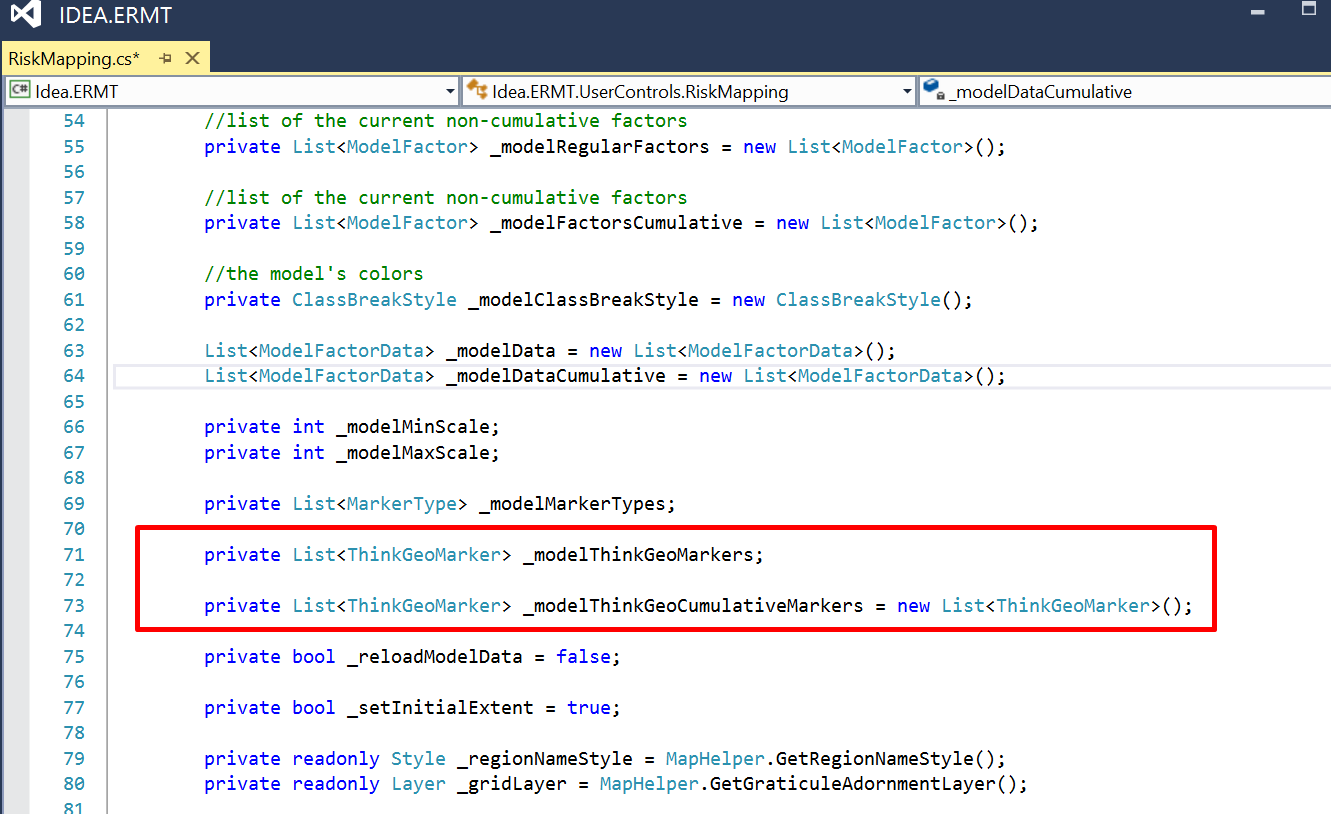


*“Screenshot shows how MapSuiteCore.dll library has been included as a reference ready to be used throughout the project”*

Inside the code, as an example, we captured pieces of code from RiskMapping usercontrol which is one of the most used control of the project.

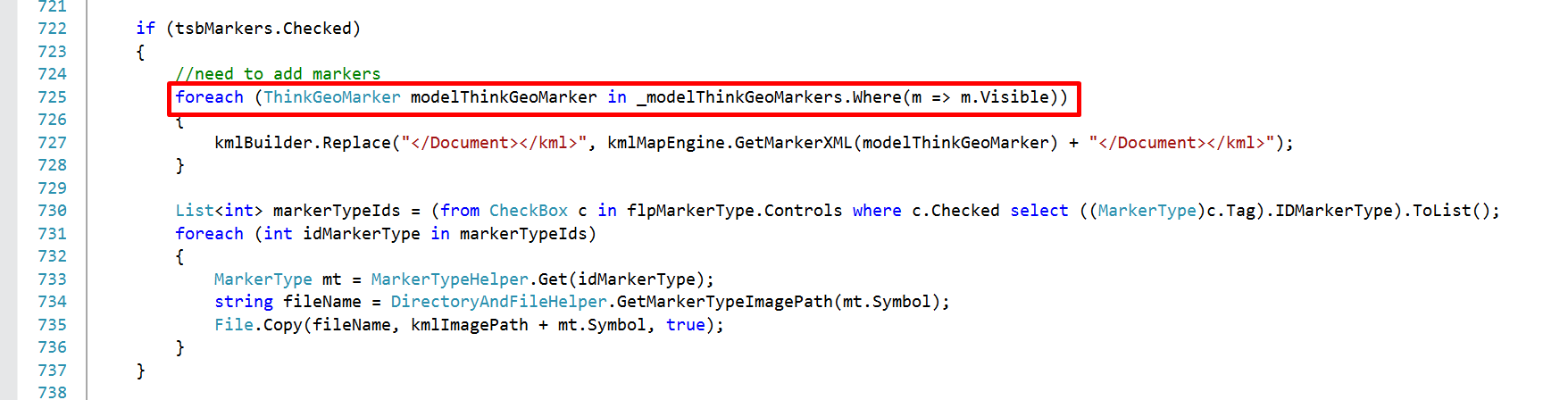


*“ThinkGeo.MapSuite.Core and ThinkGeo.MapSuite.DesktopEdition are the core of the library that should always be accessed. We also can see a new initialized ThinkGeoMarker object from the library.”*



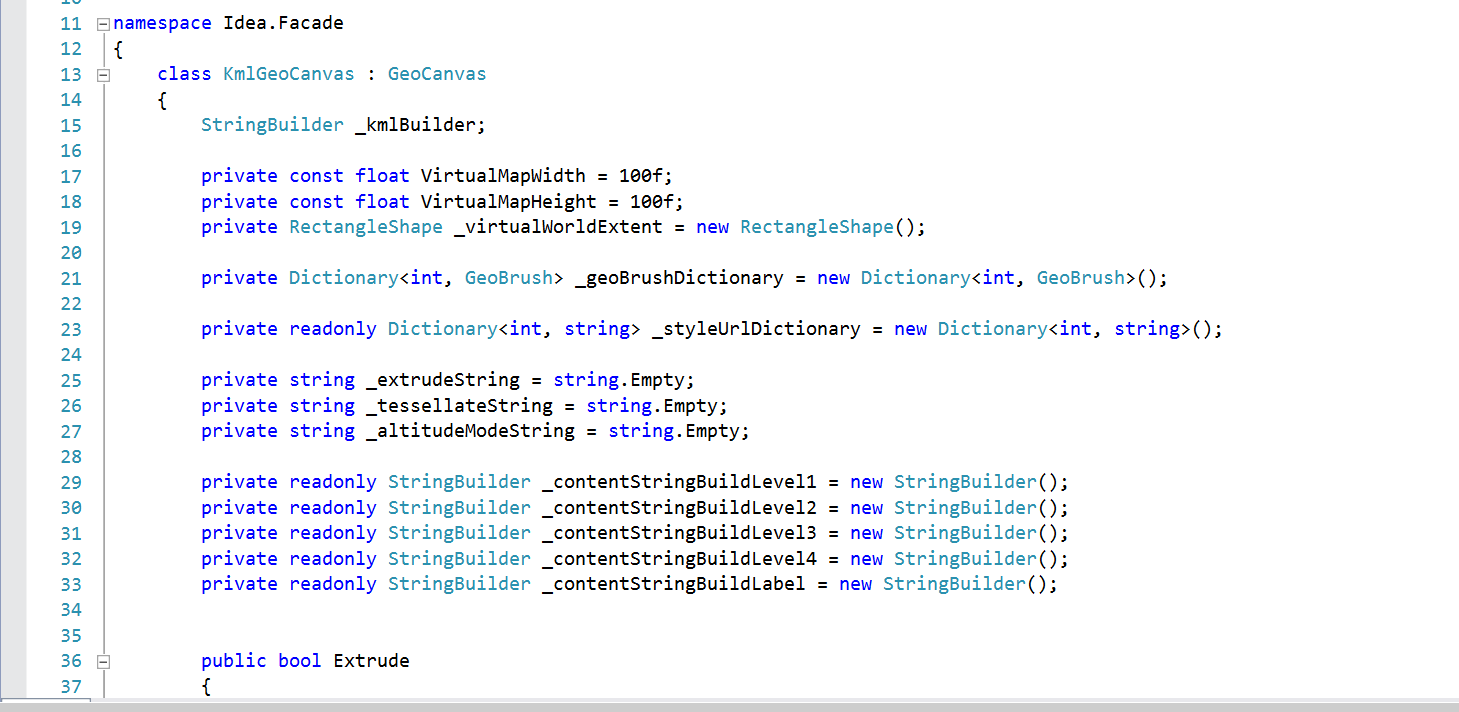
*“This screenshot shows another example on how to instance new objects from the library as part of the variable declaration section of the class”*

After initializing the object it is time to explain how to call inside methods, private properties and all type of facilities that this powerful complete library can provide.



“*Get all visible Markers from the model of ThinkGeo”*

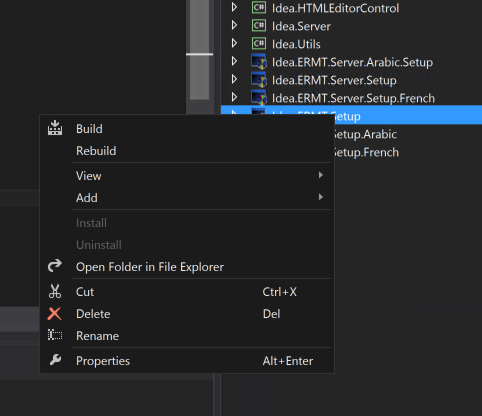
In Idea.Facade we can find the KmlGeoCanvas class inherited from ThinkGeo.MapSuite.Core.GeoCanvas that allows you to implement different drawing systems, such as GDI+, GDI, or WPF. It can also be used for other systems, such as PDF.



*“In this example we can see KmlGeoCanvas class included on the Idea.Facade”*

# Installation Projects

To generate the executable files that are required in order to install both the client and the server, you will need to set the solution to RELEASE MODE, then right click the installer you wish to generate, and click BUILD. That will generate all the executable files you will need to install the server and the client. These files can be found in the BIN folder.



# Installation Guide Technical Instructions

IDEA Application has 2 main type of installer files:

* Idea.ERMT.Server.Setup
* Idea.ERMT.Setup

The installation should be executed under an Administrator user with full permissions to allow setup create, modify or delete folders and files required for the setups.

Idea.ERMT.Server.Setup is the Idea.Server project installer and it will be installed as a Windows Services. It must be installed in first order.