Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Engineering

Bachelor Thesis

Development application of Data Analysis

Aidyn Kudamanov

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Aidyn Kudamanov

Systems Engineering and Informatics
Informatics

Thesis title

Development an application of data analysis

Objectives of thesis

The main objective of this bachelor thesis is to create an application which will have an information data about the world and local data information from each country that is possible to collect for desktop OS using C# programming language.

The partial goals of the thesis are:
- to create continuous connection with each data sources that will be used while user using an application.
- to create simple and understandable interface.
- to create database collection and analysis of it.

Methodology

Methodology of the thesis is based on study and analysis of information resources. The practical part is focused on design application using C# programming language. Then it is necessary to use the training literature to write the programming components. On the basis of theoretical knowledge and author's own work, the conclusion of the thesis will be formulated.

The proposed extent of the thesis

30 – 40 pages

Keywords

C#, database, programming, applications, algorithms

Recommended information sources

C# 6.0 and the .NET 4.6 Framework, Andrew Troelsen, 2015, ISBN 978-1484213339
C# 7.0 in a Nutshell: The Definitive Reference, Joseph Albahari, ISBN 978-1491987650
DOWEK, Gilles. Principles of programming languages. Springer Science & Business Media, 2009, ISBN 9781848820326
SHEN, Alexander. Algorithms and Programming: Problems and solutions. Springer Science & Business Media, 2011, ISBN 9781441917485
The C# Player's Guide (3rd Edition), RB Whitaker, 2016, ISBN 978-0985580131

Expected date of thesis defence

2019/20 SS – FEM

The Bachelor Thesis Supervisor

Ing. Jan Tyrychtr, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 24. 1. 2019	Electronic approval: 24. 1. 2019
Ing. Martin Pelikán, Ph.D.	**Ing. Martin Pelikán, Ph.D.**
Head of department	Dean

Prague on 23. 03. 2020

Declaration

I declare that I have worked on my bachelor thesis titled "Development an application of data analysis" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on 23.03.2020 ____________________________

Acknowledgement

I would like to thank Ing. Jan Tyrycht, Ph.D. for his advice and support during my work on this thesis.

Development application of Data Analysis

Abstract

The main goal of the literature review is to research the basics and main rules to work with programming on C#, make an overview of databases, Human-Computer interaction, software engineering. The main goal of the practical part is to implement the knowledge in the project by creating a new application that will receive, work, manipulate with Data to represent it in the tools for Analytical researching.

Keywords: C#, database, relational database, programming, application, algorithms

Table of content

List of figures

List of tables

List of abbreviations

C# - Si Sharp
XML – eXtensible Markup Language
SQL – structured query language
ODBS – Open Database Connectivity
RDB – relational database
DML – data manipulation language
DLL – data definition language
NF – normal form
ER – entity relationship
OOP – object-oriented programming
UX – user experience
CSV – comma separated value
UML – Unified Modelling Language

1 Introduction

There is a lot of data about a country that can be explored. As an example, GDP, level of unemployment, level of education, etc.
Users can get all information from the internet, but on the other side how many websites contain data about each country and their inside statistic data. I found a few, that don't provide full information for free, which means you must pay for the data that can be found in the paid database.

This bachelor thesis aims to introduce and provide tools for possible ways for collecting and analyzing data about inside country collections of data for users and businesses who need to get information in different categories.

Categories can vary from economics to research and development, tourism, education, etc.
Each category can contain subcategory, for example, the economy category contains GDP, inflation rate; education category can have primary enrollment, duration of secondary education; in the national defense, category user can see military expenditure, arms import/export. There are a lot of fields that can be added.

The output of this thesis should be a desktop application that enables users to collect information using a minimum amount of movement. The user should be able to see graphs and tables of the data.

2 Objectives and Methodology

2.1 Objectives

The main objective of this bachelor thesis is to create an application which will have an information data about the world and local data information from each country that is possible to collect for desktop OS using the C# programming language.

The partial goals of the thesis are:
- to create a continuous connection with each data source that will be used while users using an application.
- to create a simple and understandable interface.
- to create database collection and analysis of it.

2.2 Methodology

The bachelor thesis will contain next:

1. Researching the Information for developing the Application.
2. Data composition and structure.
3. Problem-solving methods and algorithms.
4. User interface structure.
5. Program structure.
6. Program description.
7. Implementing in program.
8. Testing the program, examples of the result of the program.

Data composition and structure should provide information about the composition and structure of the data.

Problem-solving methods and algorithms should contain information about problem-solving methods for specified functions and data structures (algorithms, basic formulas, how they are used, query texts, component classes used, their properties, methods).

The user interface structure should contain information about the UI (User Interface) structure of the developed application program. The user interface of the dialog program should provide a selection of functions and data work.

Program structure should include:

- A hierarchical program model and information about the functions performed by event handlers.
- Program schemes (main program and separate procedures: connection to the database and output of the result of the request, sorting, filtering).
- Description of algorithms that should contain: the purpose of the algorithm.

3 Literature Review

3.1 Programming Language

At the beginning of the topic, I would like to introduce information regarding programming language. In the introduction of the book “A programming language,” Kenneth E. Iverson classify programming language as an effective notation for the description of programs that exhibit considerable syntactic structure. Which means that programs contain set of complex algorithms, procedures and programming language should be concise, precise, consistent over a wide area of application, mnemonic, and economical of symbols; it should exhibit the constraints on the sequence in which operations are performed, and it should permit the description of a process to be independent of the particular chosen for the data. (Iverson, 1962)

3.2 C# Programming Language

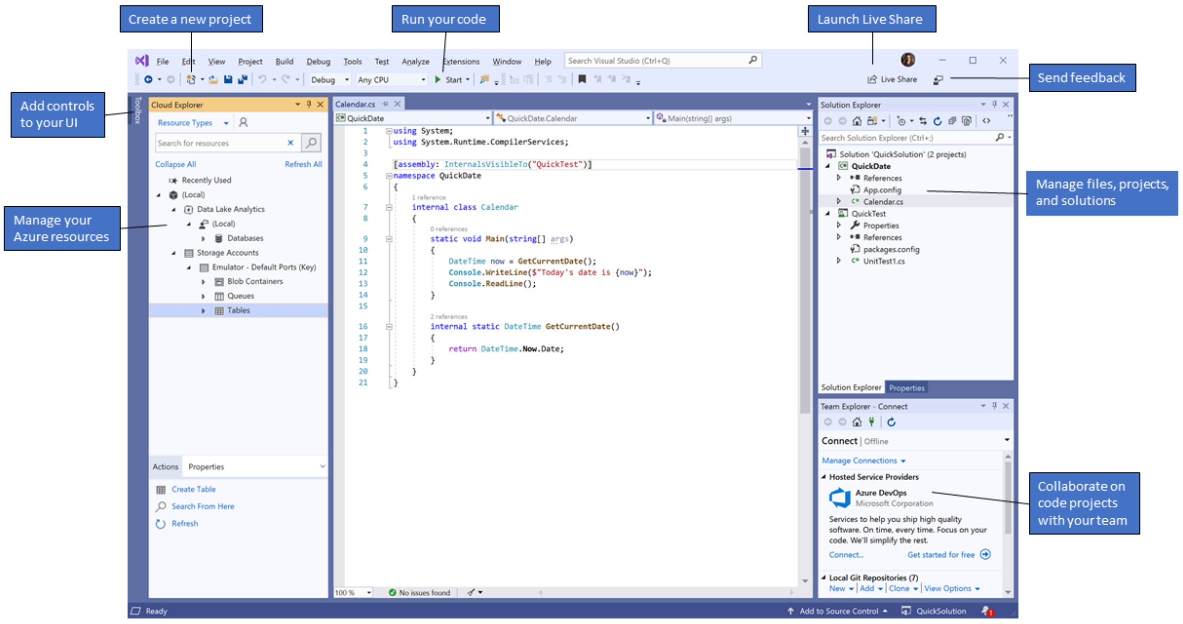
3.2.1 Brief Introduction

The C# (Si Sharp) project has been developed 12 years ago to achieve a new standard of a programming language such as simplicity, object-oriented, modernity and type-safe for new platforms .Net. The language is popular nowadays and used by millions of programmers around the world. Because the language was developed by the Microsoft team, then it has been used in all software, applications, what has been released by its company. Some of the interesting of them are “Mono” (based on “Mono” “XamarinStudio” has been released), DirectX (API – Application programing interface) often used for programming video games, Unity (cross-platform engine for creating 2D, 3D video games).

Eric Lippert says: “C# is also increasingly a functional programming language. Features such as type inference, lambda expressions, and monadic query comprehensions allow traditional object-oriented developers to use these ideas from functional languages to increase the expressiveness of the language”. (Hejlsberg, 2010)

3.2.2 Visual Studio

For using C# there is a special tool that can completely simplify programming. Visual Studio (Integrated development environment) is a software that can be used for coding, debugging and publishing applications. (Microsoft, 2019)

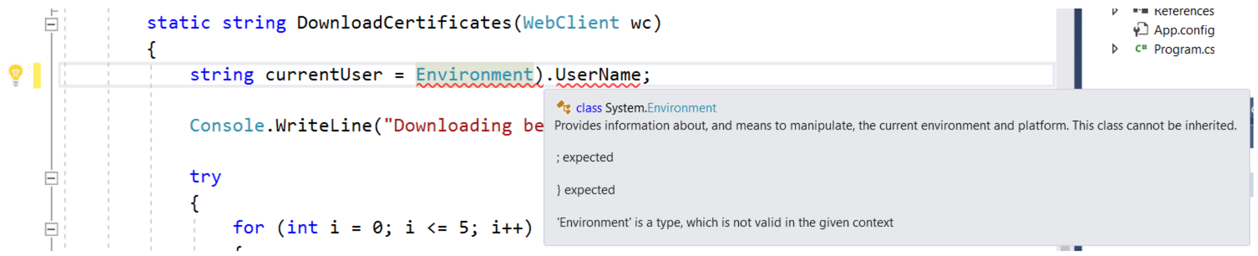

Create a new project
Run your code
Launch Live Share
Send feedback
Add controls to your UI
Manage your Azure resources
Manage files, projects, and solutions
Collaborate on code projects with your team

```
using System;
using System.Runtime.CompilerServices;

[assembly: InternalsVisibleTo("QuickTest")]
namespace QuickDate
{
    internal class Calendar
    {
        static void Main(string[] args)
        {
            DateTime now = GetCurrentDate();
            Console.WriteLine($"Today's date is {now}");
            Console.ReadLine();
        }

        internal static DateTime GetCurrentDate()
        {
            return DateTime.Now.Date;
        }
    }
}
```

Figure 1 The Visual Studio 2019 IDE
https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019

Visual Studio provides some helpful tools that can help to develop. Below you can see some of them:

Squiggles and Quick Actions - squiggles underline with wavy lines problems, errors in the code without waiting run of the program. Quick Actions can fix the problem, shown as a light bulb.



```
static string DownloadCertificates(WebClient wc)
{
    string currentUser = Environment).UserName;

    Console.WriteLine("Downloading be

    try
    {
        for (int i = 0; i <= 5; i++)
```

```
class System.Environment
Provides information about, and means to manipulate, the current environment and platform. This class cannot be inherited.

; expected

} expected

'Environment' is a type, which is not valid in the given context
```

Figure 2 Suiggles and Quick Actions

Code cleanup – before going to the code review the Visual Studio can use the tool to resolve any issues that exist in.
Refactoring – intelligent renaming of existing variables, extracting and importing code lines into a new method.

3.2.3 Nuget Packages

From the book "Beginning C# 7 Programming with Visual Studio 2019" authors explain how to create a new database and mention information about entity framework.

Entity Framework is the class library in .Net Frameworks that has all concepts of the E-R model. It helps with mapping objects from the program to the entities in a relational database and can be named as object-relational mapping. Object-relational mapping can map all objects, classes, properties from the code to the database. This framework has to be downloaded from NuGet Package Manager.
NuGet is a package manager for .Net that helps programmers to create and consume packages. (Perkins, Hammer, & Red, 2018)

To install Entity Framewok:

```
Tools > NuGet Package Manager > Manage NuGet Packages for Solution
```

After it is needed to find Entity Framework from Browse Layout and proceed with installing.
Next thing is to add a new framework to other namespaces.

After downloading the new framework, it can simply add the namespaces at the top of the file:

using System.Data.Entity;
using System.ComponentModel.DataAnnotations;

3.2.4 Choosing a database model

For organizing working with Data during application creation, Microsoft Visual Studio environment offers a different kind of Data Source.

- Local database Microsoft SQL Server, which situated in the separated file (.mdf file).
- Local database Microsoft SQL Server. In this case, it is possible to set up a local server. For example, “SQLEXPRESS”.
- Local database Microsoft Access.
- Oracle database.
- Database created using ODBS (Open Database Connectivity) driver.

For work, I’ll use a local database, which situated in a separate .mdb (Microsoft Database) file and designed to run relation database management.

In the book “Microsoft SQL Server 2008 A Beginners Guide” writer Dusan Petkovic describes the component of database and MSS (Microsoft SQL Server) as a relational database system.
“Relation database system” was introduced in the article “A Relational Model of Data for Large Shared Data Blanks” in the year 1970 by E.F.Codd. The main concept for the RDB (Relation database) is a relation. RDB has only tables from the Human view. The table always has data value in every column and row.
As an example, you can see an example of tables for counties.

country_ID	country_Name	country_Code	rank
1	Czech Republic	CZ	49
2	Kazakhstan	KAZ	56

3	United States of America	USA	1
4	China	CN	2

Table 1 Example - The country rank by WELT 2019 table

key_Indicators	value	country_ID
Real GDP Growth	2.9 %	1
Population	10594438	1

Table 2 Example - The country researching area table

The relational database also includes DML (Data-Manipulation Language) and DLL (Data-Definition Language). DML is a language that can manipulate data by the appropriate data model. DLL is a special language that specifies a database schema by a set of definitions.

3.2.5 Brief about relational databases

The relational database also includes DML (Data-Manipulation Language) and DLL (Data-Definition Language). DML is a language that can manipulate data by the appropriate data model. DLL is a special language that specifies a database schema by a set of definitions.

In the relation database, each table has columns with their unique names. In Table 1 and Table 2 you can review an example of RDB. Table 1 shows information about countries and Table 2 deep detail of each country. The first table contains in column "country_ID" which after it can be used in the second table to identify and show data about each country. (Korth, 2010)

3.2.6 Normal Forms

Author Dusan Petkovic in the book "Microsoft SQL Server 2008 A Beginners Guide" identifies normal forms as forms that are used for normalization of database and data. Normal Forms can be divided into at least five normal forms and three of them are the most important.
First Normal Form (1NF) doesn't contain multivalued or composite attributes, which means there are no other attributes inside attributes. You can review it in table 3.

country_ID	code_ID
CZ	+420
KAZ	+7

Table 3 Example of 1NF (Normal Form)

Second Normal Form (2NF) this time every table that has only one column with a **primary key** is always 2NF.

Primary Key is a special key that keeps value to be unique and must be set up if there are several candidates key within a table.

Third Normal Form (3NF) can be identified only if there are no functional dependencies between columns.

3.2.7 Entity-Relationship Model

In the book “The Entity-Relationship Model: a basis for the enterprise view of data” (Chen, 1977) can be found the main information about ER-Diagrams. According to the book entity-relationship model can be used in unifying the data, it can be a basis that adapts to the natural view. As a specific toll for designing the model, it contains a special diagram technique for representing data. The main purpose of the E-R diagram (Entity-Relationship Diagram) is to provide a view of the requirements to meet the needs of the developed IS (information system). All development of the model begins with the E-R diagram when the general list of tables and the relationships between them are determined. Then keys, attributes must be determined.
In the book, “Database system concepts” (Korth, 2010) authors provide a major component of the E-R diagram: rectangles, diamonds, lines. Each of the figures and lines represents its function. Rectangles represent entities. Diamonds represent relationship sets. Lines connect entities to relationship sets. Below you can see a little example in Figure 3.

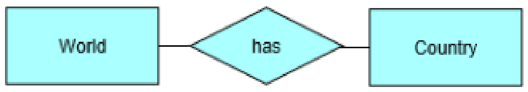


Figure 3 simple E-R diagram

3.2.8 Project

As I have chosen Microsoft SQL server the installation of Visual Studio 2017 already includes all files of SQL Server 2016 Express that helps to work with database files.
Before proceeding with creating a database it is required to prepare a folder for the application.

The step is to create a project using Visual Studio.

```
File > New > Project…
```

Figure 4 Command for creating new project

After in dialog window choose templates. In the field „Installed Templates“ I choose Visual C#, in the middle of templates „ Windows Forms Application“, in field „Name“ setup a name of the application, in the field „Solution Name“

3.2.9 C# Syntax

Based on the book “Beginning C#7 Programming with Visual Studio 2017” (Perkins, Hammer, & Red, 2018) I want to describe the basic syntax of C#, variable, types, expressions.
C# code looks like C++ or Java. For example, naming “Int”, “long”, “char” can be written as in C++.
C# is a block-structured language that can be determined that each statement is a block of code. They must be opened and closed by brackets ({}). Example:

```
{
            <code1, statement 1>;
            <code2, statement 2>;
```

}

Comments are an important part of any programming language, as it makes it easy to explain different parts of the code. Traditional C style comments are used in the C# - one-line (//..) and multi-line (/*….*/):
//One-line comment

*/**
  Multi-Line comment/*

Everything in a one-line commentary, from // to the end of the line, is ignoring by the compiler, as is the entire multi-line comment, located between /* and */. There can be no combination in a multi-line commentary because it will be interpreted as the end of the commend.

I can put a multi-line comment in one line of code:

Console.WriteLine (/**Comment**/"Compiling");

Comments like before should be used carefully because they can impair the readability of the code. However, they are convenient when debugging, when you need to temporarily try to run a program with a specified other value.

3.2.10 Namespaces

A namespace is a special way for .Net to provide spaces for program code and one of the most important terms. By default, the system has a global namespace provided by C# code. Global namespace means that the code inside this container can be used in different codes by referring to them. After creating a project Visual Studio shows us namespaces:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using.SystemThreading.Tasks;
Namespace CountryApp
{
…
}
```

To declare that C# will use in the code System namespaces it automatically adds namespaces "Using…".
using System: namespace needs to have in .Net Framework application because of all basic functionality, all other needs just in case of often usage.

3.2.11 Variables

Declaring a variable in the C# syntax has the following general view:

<data_type> <variables>;

A variable is a name that gives to memory areas to use it to write down data, read it and modify it during the program. C# is a language with static typing. This means that when I create a variable, we're bound to determine the type of data it stores, and that type can't be changed during the program. The type also depends on the amount of memory allocated to the variable. This helps to avoid bugs and use memory effectively, and it's "friendly" for a compiler that turns a program in the language of the C# into machine code.

3.2.12 Expressions

Using operators and operands I can construct expressions. C# provides a number of operators supported by built-in types: arithmetic operations with numerical operands, logical operators perform logical operations with bool operands.
Operators can be separated into three categories:
· Unary
· Binary
· Ternary
Most operators can be overloaded. By overloading operators, it is possible to specify the operators' behavior for user-defined operands. In table 4 „Types and Aliases“ you can check some of the CST types, names and descriptions. Table 5 contains Mathematical operators and Table 6 is for boolean expressions. All tables are in the appendix.

All information was interpreted from the book „Beginning C#7 Programming with Visual Studio 2017“ (Perkins, Hammer, & Red, 2018).

3.2.13 Statements

The book "The C# Programming language" (Hejlsberg, 2010) provides information about statements. Statements are actions of a program. There are several statements that supported by C#: blocks.

Declarative statements announce new variables and can assign the meaning of the expression. Declarative instructions end with a semicolon. In one declarative instruction, I can declare several variables of the same type by dividing them with commas.

```
bool example = true; xmlp = false;
```

Expression statements are an expression that contains valid instructions that do something: assign or change the value of the variable, create a copy of the object, call the method.

Selection statements change the flow of the program depending on certain conditions.
Selection "If" runs another statement if the logical expression returns true. It may be supplemented by another clause "else", performed if the logical expression returns false. Inside the "If" statement I can put another "If".

```
if (5 < 2 * 3)
{
    Console.WriteLine (“true”);
    Console.WriteLine(“…”)
}
else
   Console.WriteLine(“false”);
```

A switch statement allows choosing how to run a program based on selecting one of the possible variable values. The switch statement is more productive than a few “If” statements because it calculates the expression on which the choice is made. In the switch statement, I can use an expression that returns only string types, predetermined inemurable types, enum type, and nullable variants. At the end of each case, it is needed to explicitly specify where the program should go next.

```
Switch (countryNumber)
{
   case 1:
      Console.WriteLine(“United States”);
      break;
   case 2:
      Console.WriteLine(“France”);
      break;
}
```

Iteration statements: Repeat statements include “while”, “do-while”, “for” and “foreach” statements.
The **“while”** cycle repeats the set of instructions while the logical expression is true. The expression is calculated before the instructions block is executed.

```
while (i < 5)
{
   Console.Write (i++);
}
```

The “**do-while**” cycle differs from the first cycle only because the logical expression is calculated after the instructions are executed.

```
do
{
   Console.WriteLine (i++);
}
while (i < 5)
```

The “**for**” cycle is similar to the while cycle, but it contains a number of special conditions for initialization and interization (repeat) of the variable cycle (variable, loop control). More precisely, the “**for**” cycle contains three such conditions:

- The initialization (init-clause) takes place before the cycle and usually initiates one or more cycle variables
- Condition-clause is a logical expression that is calculated before each repetition (interpretation) of the cycle. The cycle body runs if that expression is true.

- The iteration-clause is repeated after each replay (interration) of the cycle. In it, the variable cycle is usually updated.

```
for (int i=0; i < 5; i++)
   Console.WriteLine (i);
```

The “**foreach**” cycle repeats each element of the listed object. Most of the types in the C# that represent a set of items are listed (such as an array and a string).

```
Foreach (char c in “cz”)
   Console.WriteLine (c + “ ”);
```

Jump statements include “break”, “continue”, “goto”, “return” and “throw”.
“**break**” stops running a loop or switch. “**continue**” stops the next replay in the loop and moves on the next one. “**goto**” transfers execution to the specified label within the instruction block. “**return**” instruction can be used anywhere in the method and comes out of the method and returns a certain type (if the method returns the value).

3.2.14 Object-Oriented Programming

Authors (Perkins, Hammer, & Red, 2018) in the book “C# 7 Programming with Visual Studio 2017” answers to questions “What is Object”, “What is OOP”, basics of OOP (Object-Oriented Programming).

OOP techniques require to use many modules of code that can be isolated or independent of the others. This technique provides more benefits for code using. OOP techniques have the benefit of extensibility with rooting in the structure and meaning of data which means putting more effort into different stages of the project. It makes thinks simply, because of approaching data representation.

3.2.15 Object

In OOP-object is a constructing box. This constructing box can encapsulate some parts of code, which can be a process. It is like a struct type. To create an object in C# I have to use types. Any type of object must be declared by a special name in OOP, which can be named as a class.
To have access to the data inside object OOP provides properties and fields. The object in the one class can be stored in different values.

Fields and Properties are typed, and data can be stored in different values. The main difference is that the fields can have direct access to the data, but fields don’t provide the smallest details of that information. In some cases, the properties can be looked better for state access, because of controlling over different behaviors.
Properties and fields can have read/write access. If the user needs to have read-only access, certain properties can provide to not edit them.
Accessibility can be also specified for both fields and properties which determines what block of code can access public (available for whole code) or private (for code within the class).

Jesse Liberty says that “A property looks to the creator of the class like a method allowing the developer to add behavior before setting or retrieving the underlying value. In contrast, the property appears to the client of the class as if it were a field, providing direct, unencumbered access through the assignment operator."

Methods. To have the object's functionality I can use methods that can refer to functions. They can be private or public as properties and fields. Methods use condition of the objects to influence their operations, to access to private members.

Everything in C# is an object. Classes, variables, commands are property or method.

Every object has two stages of the life cycle. The first one is a construction where an object needs to be initialized and the second one is destructions where an object has to be deleted to perform some memory.

Constructions, as said previously, must be initialized and mainly with the object it happens automatically. In C# construction they can be called using "new".

```
CountryOfWorld country = new CountryOfWorld();
CountryOfWorld country = new CountryOfWorld("Czech Republic")
```

Constructions can be public or private.

Destructors are used to clean saved data in memory after the object is deleted.

3.2.16 Features of objects

Here I'll introduce about main features of OOP in C# and more deep information can be found in the book "Object-Oriented Programming: an evolutionary approach" by (Cox, 1986).

Interfaces

In the OOP .Net interface is simply a listing of techniques that must necessarily be implemented by the class. The interface gives an opportunity to indicate what exactly should be the object of the model without describing the behavior of the object. It allows to describe the basis of the model, to determine what and where should be, without describing the behavior of a given property or method.

The interface is a set of abstract members. Specific members, defined by the interface, depending on what behavior is modeled with it. It expresses the behavior that a class or structure may choose to support. Each class can support as many interfaces as necessary and support many behaviors. None of the methods should have a body in the interface. This means that the interface does not provide any implementation. It only indicates what to do.

Once the interface is identified, it can be implemented in any number of classes or any number of interfaces can be implemented in one class. Each class has complete freedom to determine the details of its own deployment of the interface, which means that it can be used in different ways. The basic principle of polymorphism can be fully implemented: one interface – many methods.

```
namespace Interface
{
   interface IArea
   {
      double area();
    }
}
```

Inheritance

The basic and the most important features in OOP is inheritance. Inheritance allows any class to be inherited from another. Classes with inheritance can be divided into 2 parts, parent class (or base class) and inherited class. Thanks to this, one class can inherit the functionality of another class. By default, all classes inherit from the base class Object, even if we do not explicitly set inheritance. And there are some limitations:

- Multiple inheritances are not supported, a class can be inherited only from one class.
- The type of access to the class has to be the same with parent class or more private. For example, if the parent class has internal access, then inherited class can be internal or private, but not public.
- If the class is declared with a “seal” modifier, then derived classes can’t be inherited from this class.

```
class megapolis : city
{
   Public void Display()
   {
   }
}
```

Polymorphism

Polymorphism means the presence of many forms. It is often expressed as “one interface, several functions”. It can be static or dynamic. In static polymorphism, the response to a function is determined at compile-time, in dynamic it is solved in the runtime.
Polymorphism can reduce the code for achieving a task in different objects. It’s not like an inheritance that can be used only one time, but in child classes and as long common classes exist in a hierarchy. In C# I can have multiple definitions for the same function name in the same scope. The definition of the function must differ from each other in types and/or the number of arguments.

```
c.print(“CzechRepublic”)
c.print(46)
```

It also allows creating abstract classes that are used to provide the partial implementation of an interface class. Implementation is completed when a derived class is inherited from it. Abstract classes contain abstract methods that are implemented by a derived class. Derived classes have more specialized functionality.

Events

Events signal the system that a specific action has occurred.
Events are declared in the class using the keyword “event”, followed by the name of the delegate.

```
public delegate void StateHand(string message);
public event StateHand WithDrawn;
```

Communicating with a delegate means that the method that handles this event must accept the same parameters as the delegate and return the same type as the delegate and I can use event-driven applications. As an example, each click is achieved through the event, because they are actioning by the control devices.

3.2.17 Classes

Using the book "C#7 and .Net Core CookBook" by (Strauss, 2017) I want to provide information about classes.

A description of an object is a class, and the object represents an instance of this class. By default, a console application project already contains one class by default, called "Program", from which the program starts.

```
public abstract class Country {}
```

A class can be defined inside a namespace, outside a namespace, inside another class. Classes are placed in separate files and all class functionality is represented by its members – fields, properties, methods, events.
In addition to the usual methods, classes also use special methods – called constructors. Constructors are called when a new object of this class is created. Constructors initialize the object. If no constructor is defined in the class, then a default constructor is automatically created for this class. This default constructor has no parameters and does not have a body. By default, classes are declared as internal, but it is possible to specify it to be "public", "abstract", "sealed" or can specify inheritance in the class.
If the constructor does not initialize the values of the object variables, then they get the default values. For variable int, this is the number 0, and for the type string and classes, this is null.

Key word "this"

The "this" keyword represents a reference to the current instance of the class. While creating constructors I can simply access from one constructor to another through keyword "this", passing the necessary values for the parameters, without needing to duplicate the functionality of constructor.

```
class Country
{
   public string name;
   public int population;

   public Country() : this("Unknown")
   {
   }
   public Country(string name) : this(name, 1800000)
```

```
{
}
public Country(string name, int age)
{
    this.name = name;
    this.age = age;
}
public void GetInfo()
{
    Console.WriteLine($"Name: {name}  Population: {age}");
}
```

In this case above, the first constructor calls the second, and the second calls third. By the number and type of parameters, the compiler will know which constructor is being called.
In the expression this.name = name: the first part of this.name means that name is the field of the current class, not the name of the name parameter. If parameters and fields were called differently, then using "this" would not be necessary.

Object initializers

For initializing class objects I can use initializers. Initializers represent the transfer in curly brackets of values to the variables of values to the available fields and properties of the object:

```
City cz = new City { name = "Prague", population=1281};
Cz.GetInfo();         // Name: Prague Population: 1281
```

Using the object initializer, I can assign values to all available fields and properties of the object at the time of creation without calling the constructor.

3.3 Databases

Previously I've already described brief information about databases. Now I want to introduce it deep more together with C# using the book "Beginning C#5.0 Databases" by (Agarwal, 2012).

3.3.1 Database life cycle

I've already introduced relational databases and the next thing that identifies the author is the database life cycle. So, database design is an iterative, step-by-step process. Each step leads to the creation of a working database. Each iteration goes from modeling to creating a structure, and then back in the order in which it makes sense to do it. Database design begins with identifying the information needs of users, creating a data model, and ends with the disposal of the database. The procedure for creating a conceptual database schema, determining the data to be included in the database, creating updates and data processing programs is called the database life cycle. The author separates the database life cycle into six stages: requirements analysis, logical design, physical design, database implementation, data modification, database monitoring.

Requirements analysis. The content of this stage is the development of a strategic plan, during which preliminary planning of a specific database management system is carried out. Planning the development of a database consists of identifying three main components: the amount of work, resources and project cost. Planning for the development of the database should be associated with the general strategy of building the information system of the organization.

Logical design. The next step is to define relationships and data in the database. The database design should be based on a specific data model, which is determined by the type of database management system intended for implementation. The conceptual data model is converted into a logical data model. One of the techniques that can be used in conceptual data is the entity-relationship diagram. It shows connections between objects.

Physical design. In relation to the database system, the physical design identifies creating a description of a set of relational tables and their limitations based on the information presented in the logical data model; definition of specific data storage structures and methods of access to them.

Database implementation. Based on previous stages and if they are complete, I can create a database using data definition language (DLL). This language contains statements that have a role in creation, modification and deletion processes in the database and its object.

Data modification. Data modification language (DML) has to be used in case of updating the database. Examples of DML are: INSERT, UPDATE, DELETE.

3.3.2 Mapping Cardinalities

The document about “Database Fundamentals” (Robbins, 1994) contains a good representation of cardinalities, which I used in presenting the mapping. The author explains the fundamental components of a relational database. In fact, that the relational databases contain related data in their tables and because of that, there were introduced three classifications for the relationships in the database. They are “one-to-one”, “one-to-many”, “many-to-many”.

“One-to-one” means that for one row in a table there is only one row in related another table.
“One-to-many” means that for one row in the table there is more than one row in related tables or zero.
“Many-to-many” means for many rows in the table there are many rows in related tables.

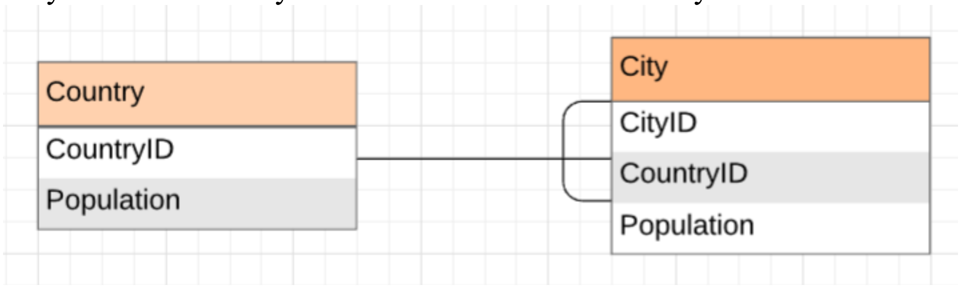


Figure 5 Example 1 to many

3.3.3 Keys

When working with tables in relational databases, each table must have a so-called primary key.

The primary key is a field that is used to ensure the uniqueness of the data in the table. This means that the value (information) in the primary key field in each row of the table can be unique. Uniqueness is necessary to avoid ambiguity when it is not known which table record can be accessed if the table has duplicate records.
A foreign key is one or more fields (attributes) that are primary in another table and whose value is replaced by the primary key values of another table.

A candidate key is any key that could be a primary key. Once the candidate is identified, choose one and only one primary key for the entity.

3.3.4 SQL Data types

The book “Introduction to Structured query language” (SQL) by (Hoffman, 1996) provides a tutorial of using the statements to manipulate with databases. Combining the book with the below table for SQL data types you can see different functionalities that can be proceeded with the data.

Type	SQL data type
Character	Char, text, varchar
Integer	Int, bigint, smallint, tinyint
Date	Datetime, Smalldatetime, Date, Time

Table 4 SQL data types

Types in SQL server can be written in code like below:

```
CREATE TABLE Country
(
   Name Char(30) Not Null,
   Population int,
   Code char(3)
)
```

3.3.5 Manipulating with Data

As was said previously it is possible to insert, update and delete data.

Inserting allows to add data or rows after creating a table to a table using the “INSERT” statement.

```
INSERT INTO Country
(<column1>,<column2>)
VALUES (<value1>,<value2>)
```

Updating allows modifying data in the table. In the “UPDATE” statement, it is required to use the “WHERE” clause, because without it the system can change all records of the table.

```
UPDATE Country
```

```
SET <column2> = <value2>,
WHERE <predicate>
```

Deleting data allows the clearing table. If we don’t input the “WHERE” clause, every row is going to be deleted from the table.

```
DELETE FROM Country
WHERE <predicate>
```

3.3.6 Querying

SELECT – is the most used command, used to get a dataset from a database table. The “SELECT” command has the following syntax:

```
Select * from Country
```

The asterisk (*) means to select everything from the table.

WHERE clause allows to create a criterion from selecting to filter data.

```
Select * from Country
WHERE Population > 10000
```

In SQL I can also use comparison operators as in C# language but different expressions. The table below shows the comparison operators.

Operator	Description
<	Less than
>	Greater than
=	Equals
<=	Less or equal
>=	Greater or equal
<>	Not equal
!=	Not equal
!<	Not less than
!>	Not greater than

Table 5 SQL Comparison operators

After filtering the needed data, I can sort it using the “ORDER BY” clause. It can be ascending or descending.

```
ORDER BY Population ASC
```

The next thing is **pattern matching**. A special technique that can be used in SQL to determine values that match a specified pattern.

Percent mark (%): This wildcard will help to find all data with a value that has been selected between percent marks.
%Czech Republic%
Underscore (_): finds everything that ends with a selected value between underscores. '_blic_'.
Square brackets ([]): finds everything with any single characters between selected values.
'[A-B]'.
Square brackets and caret ([^]): Find everything that not withing the specified range. '[^A-B]'.

Aggregate Functions used for finding values with special requirements. Aggregate functions can use "MIN", "MAX', "SUM", "AVG", "COUNT" clauses.
"MIN" and "MAX" functions needed to find the maximum and minimum, values.
"SUM" calculates the total value.
"AVG" calculates the average value.
"COUNT" function calculates the total of all records that were in the output of the system.

"In" operator finds all records that satisfy in-list conditions.
"Not In" operator find all records that are not in-list conditions.

"Between" operator finds all records that are in range condition.
"Not Between" operator finds all records that are not in range condition.

3.3.7 Joins

Joins is a relational operation allow to retrieve required information from more tables and match rows based on join specification.

Inner Join. Inner join can return those rows that satisfy their requirements and any relational operator can be used in. Inner join is a binary operation and can have tow operands. We must specify the left and right tables. The "ON" keyword specifies the "JOIN" requirements.

Outer Join. An outer join returns all rows from one of the tables that are joined, even if in first table rows don't match with the second. Outer Join can be divided into three types: left outer join, right outer join, full outer join. Left outer join will return all rows from the left table, right outer join from the right table. Full outer join allows retrieving all rows from both tables.

Union Join. This operation allows creating a new table that will have all rows from both tables. The requirement to use it is the same number of columns, compatible data types of columns.

Cross Join. This operation combines all rows from both tables.

3.4 The process of designing and engineering

In this part, I want to describe more about the interaction between humans and computers and provide knowledge about software engineering using the book "Human-computer interaction" (Preece, 1994) and has a good explanation about this topic.

3.4.1 Interaction design

The main objective of interaction design is to help users navigate the application by illustrating the relationship between user interface elements. The interactive design also can add a character to the application with animated icons, logos, and illustrations; however, usability should take precedence over-expressive elements.

The author in his book divides interaction design into four basic activities. They are identifying needs and establishing requirements, developing alternative designs, building interactive versions of the designs, evaluating designs.

Identifying needs and establish requirements: this part is needed to know what the target for users is and what support the application can provide to them.
Developing alternative designs: The main part of designing that can be separated into two parts: conceptual and physical. Conceptual designs modeling what application must do, its behavior and how it must look like. The physical design contains detailed information about the products such as colors, logos, images, buttons, icons.
Building interactive versions of the designs: This part is considered some version of the design that the user can interact with. It can be a paper-based prototype or a software version etc.
Evaluating designs: The last part contains the evaluation of design, its usability, and acceptability if it matches requirements.

Characteristics of the interaction design contain three keys. The first one is focused on users which can help to focus on issues and provide opportunities on the issues. Second is specific usability and user experience goals. It helps to choose one from different alternative designers. The third is iteration which allows finding new ways to solve design requirements.

Users in the interaction design are those people, who have a direct behavior with a product to achieve a goal. Eason (1987) identifies three categories of user: primary, secondary and tertiary. Primary users are those likely to be frequent hands-on users of the system; secondary users are occasional users or those who use the system through an intermediary, and tertiary users are those affected by the introduction of the system or who will influence its purchase.

Needs are needed to achieve user requirements. It requires understanding characteristics and user’s possibility at the right time and can they have it more effective if the application is going to be different.

3.4.2 Lifecycle models

A lifecycle model is needed to show a design that captures requirements and how they are related. The lifecycle model has different ways of implementation in the software engineer and human-computer interaction.

Interaction design lifecycle model

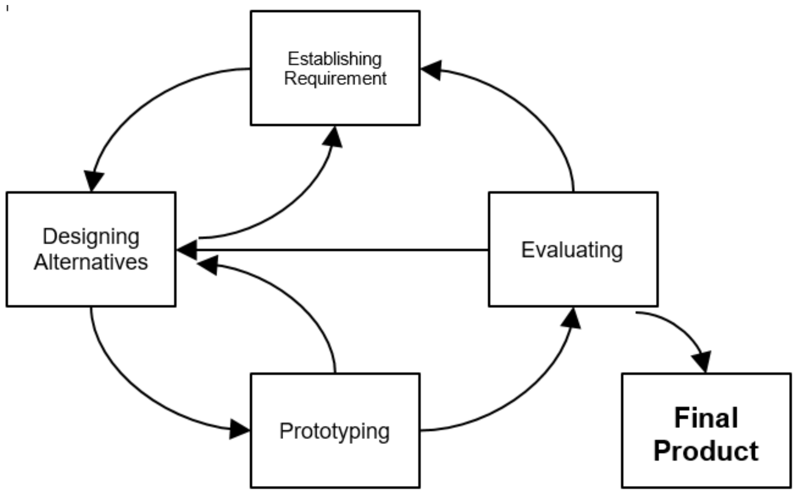


Figure 6 Interaction design Lifecycle model (Andre, 2018)

You can see a lifecycle model in figure 6 which represents four processes in the cycle until it goes to the final product. It starts with understanding the requirements and needs that must be implemented in a project which is the "establishing requirement" box. After it goes to designing alternatives to meet the needs that were identified. Then prototyping some of the alternatives. The final stage in the cycle is evaluating prototype and if it satisfies needs and requirements it can go to the final product. Otherwise, it continues its movement or can directly go to design other alternatives if the requirements are fulfilled. The designing alternatives can go back to establishing requirements if there is a misunderstanding of it. Prototyping goes back in for example if the model can't be constructed due to limitations of the system and be released for final evaluating.

3.4.2.1.1 The waterfall lifecycle model

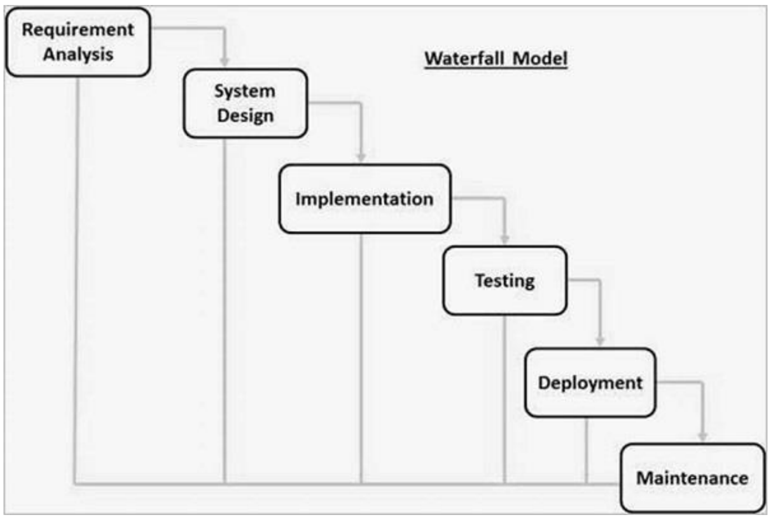


Figure 7 Waterfall lifecycle model
http://sfdcsrini.blogspot.com/2014/10/what-is-waterfall-model-in-sdlc.html

By this model in figure 7, the project will be carried out step by step, by the exact sequence of actions: a collection of requirement definitions, system and software design, implementation, testing, deployment, and maintenance.

Requirement analysis. At this point, it is important to document all requirements for future software. Discuss the details of all interested users. All incoming data needs to be analyzed and systematized. It is also important to consider all technical restrictions that may arise on the customer side. The result of this stage should be the creation of a detailed specification that meets all the requirements of the user.

System and software design. The next step in the software life cycle is to describe the scope and boundaries of the project. It includes a detailed specification of software requirements, sketches or mockups.

Implementation. In software development, I can also include the creation of an interactive prototype, which is can be the basis of the future application. This prototype helps define the architecture of the system. Once the interactive prototype and app design is ready and approved by the user, the development of application standards (name convention, way of documenting code, instruction for the end-user) begins. After that, I can safely move on to software development. Software development can be divided into small parts, or units, and each unit developed.

Testing. Once the application is completed, it is necessary to undergo testing to ensure that it is compliant. At the first testing stage, the user must try to apply the product locally in the same way as it's going to use after its release. When the major bugs identified and fixed, the software can be implemented (**Deployed**).

Maintenance. Once the application has been tested and deployed, the next phase of the software development lifecycle is maintenance. Detecting and correcting any issues, bugs during application deployed status.

3.4.2.1.2 Rapid application development (RAD)

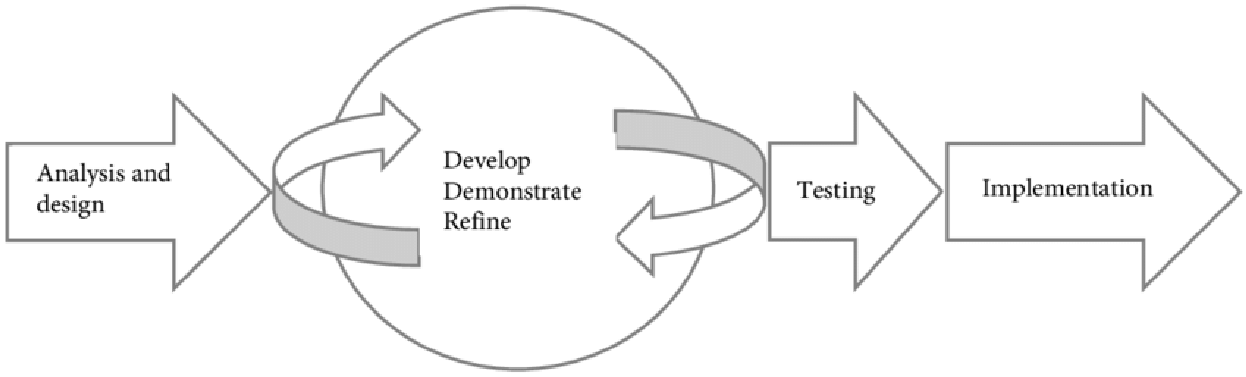


Figure 8 Example of rapid application development (Mramba, 2018)

Rapid application development (RAD) in the software engineering lifecycle by the book "Human-Computer Interaction" has five phases: project set-up, Joint application development (JAD) workshops, iterative design and build, engineer and test final prototype, implementation review. The key features for using RAD: time-limited cycles and JAD workshops in which users and developers do the requirements of the system.

In the next part of the chapter, I will start with the implementation of a new project.

4 Practical Part

In the practical part, I will go through the development process of the desktop application using Visual Studio 2019. The programming language that will be used is C#. The topics will include creating a UX designing, Programming code, and its algorithm, issues during implementation, external connections, tests.

4.1 Analysis of the Requirement

In this part, I have to identify what is expected from the common user who will interact with an application. The application has to be simple and understandable for providing data.

For that, I will use the standards of UX-designing provided by Microsoft Windows 10 for their Application.

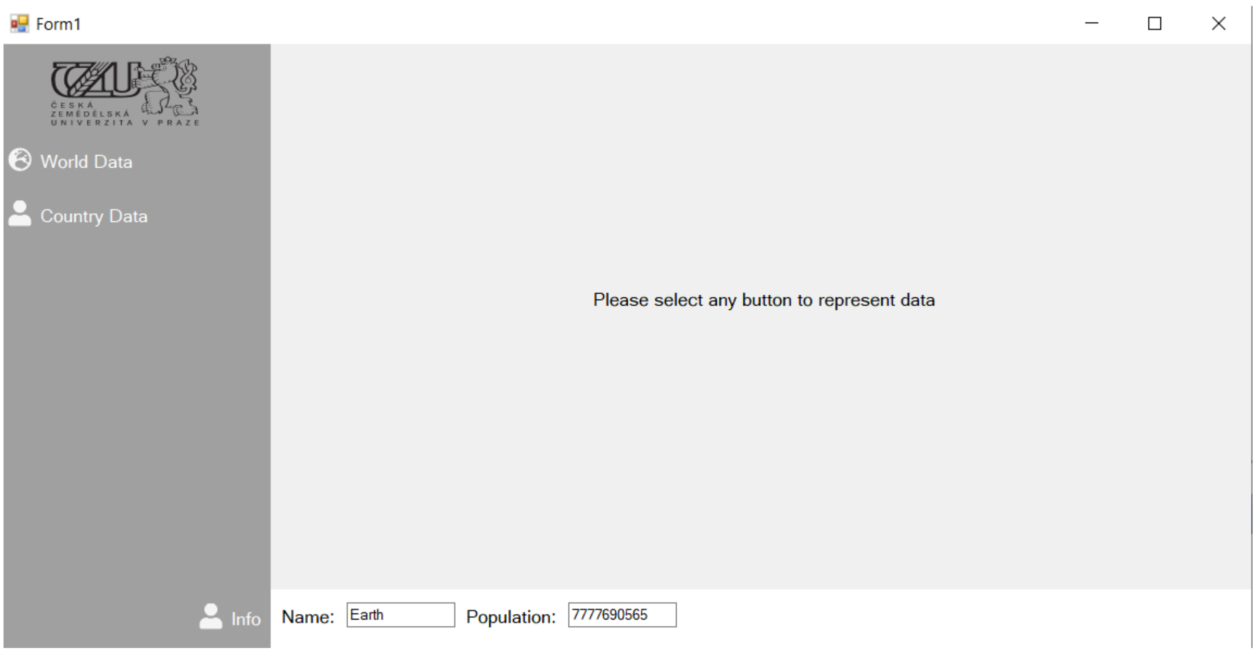
The user expects to see the values for the countries, regions, the world is a more common way as it can, to be user friendly. To represent the data from external resources such as tables from a database, comma-separated values from the file, XML schema from an external resource, I will use the Graphs, Charts, PieCharts, Geographical Maps.

4.1.1 Designing

As a standard for most of the applications, I will divide an application into 3 containers.
The left menu container will represent the buttons for viewing where it can be controlled and manipulated by the user. The bottom container represents brief information from the database. At last, the main child container to show graphics, charts, maps.
The main menu has to be dependent on the left menu, which means it will represent the forms only after the user wants to access the data.

Left Menu includes two types of buttons. The first type of buttons will be visible when the user starts to use the program and the software itself initialize first components. By clicking on the button, it has to show submenu of the main menu. The second type of buttons can access the other forms and initialize their components and new form depends on what button selected will be shown in the child main menu.

For some of the child's main forms, the user can manipulate with open boxes for changing values to represent different graphs, or search for the selected values. In the below screenshots I've provided how the designing looks like.



Please select any button to represent data

Name: Earth Population: 7777690565

Figure 9 Main Form Initialization

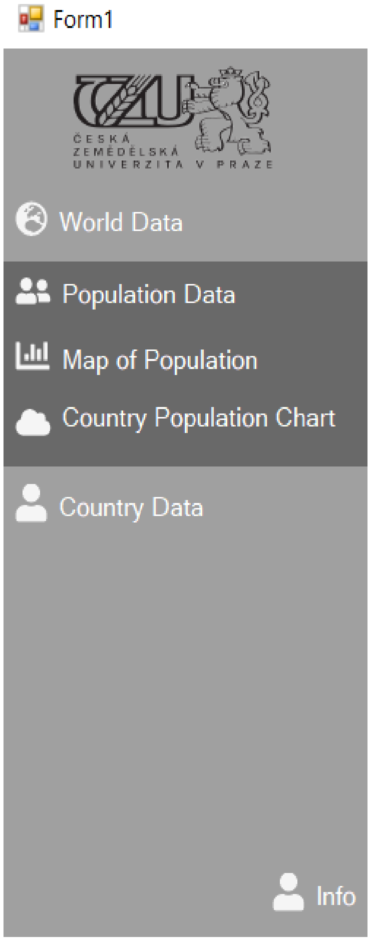
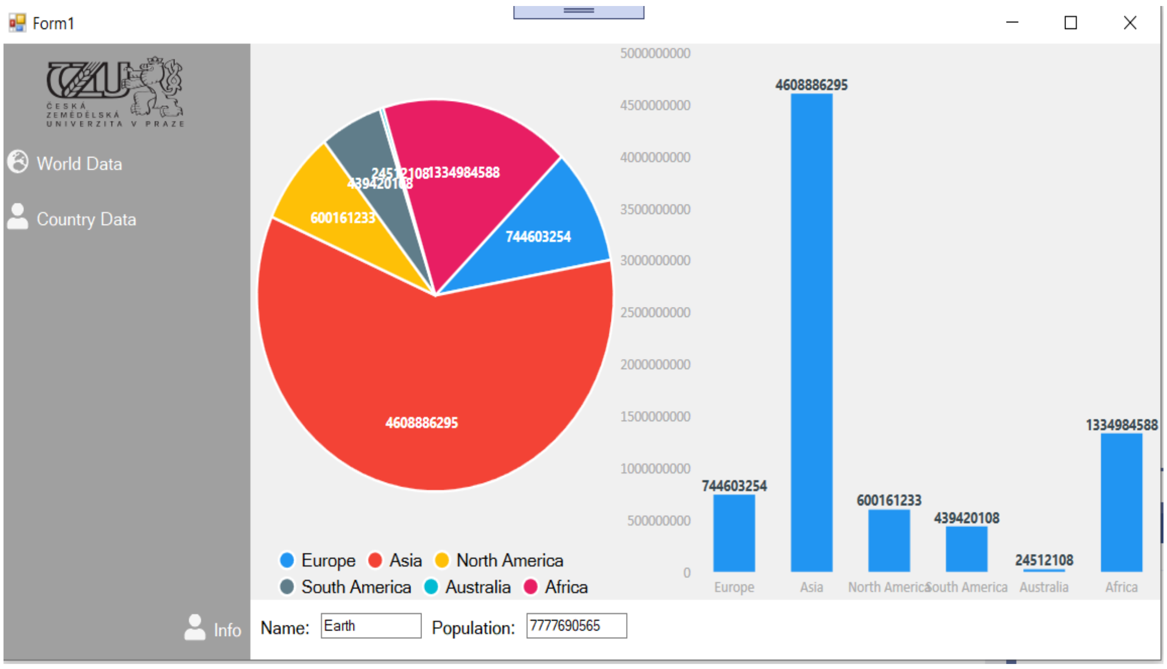


Figure 10 MenuButton Selected



Name: Earth Population: 7777690565

Figure 11 SubButton Selected

4.1.2 First Initialization

First Initialization as I described previously represent just a few numbers of buttons, labels, images, but in the C# I must declare the visibility of each object in the main form.

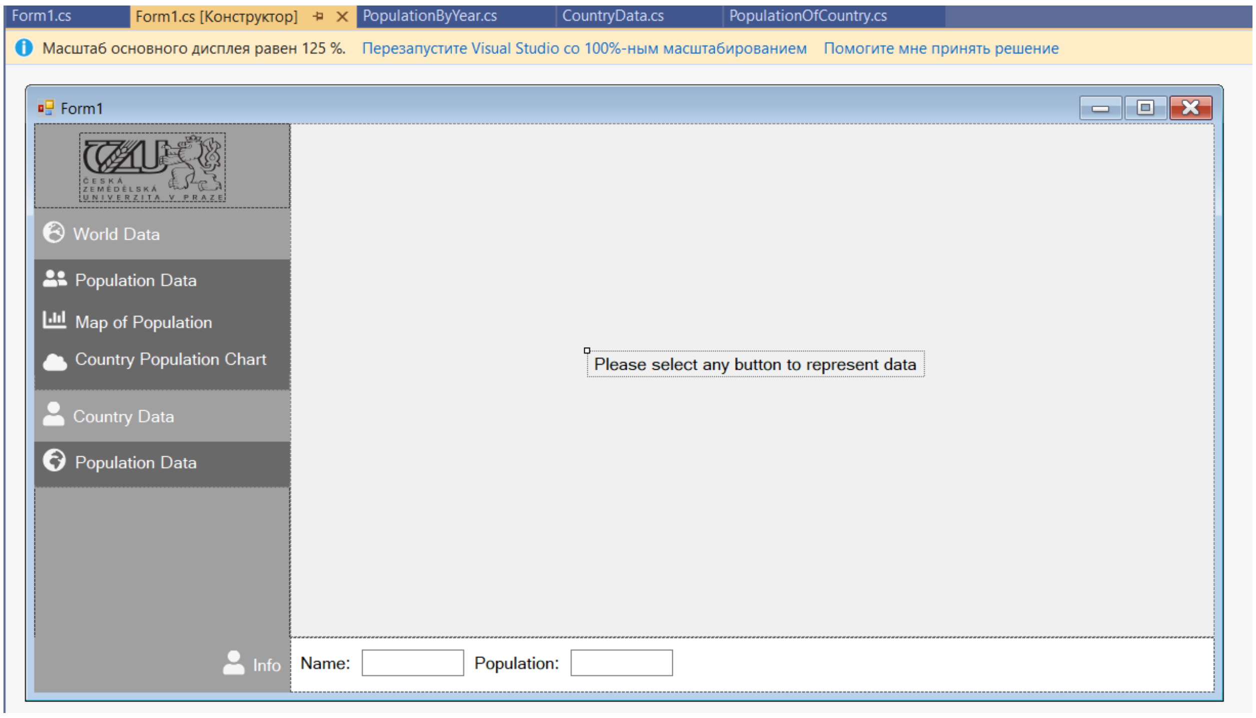


Figure 12 Main Form from Visual Studio

To represent it I would need new methods to show and hide subpanels every time button selected.

```
private void designSubMenu()
        {

            panelWorldDataSubMenu.Visible = false;
            panelCountryDataSubMenu.Visible = false;
        }

        private void hideSubMenu()
        {

            if (panelWorldDataSubMenu.Visible == true)
                panelWorldDataSubMenu.Visible = false;
            if (panelCountryDataSubMenu.Visible == true)
                panelCountryDataSubMenu.Visible = false;

        }

private void showSubMenu(Panel subMenu)
        {
            if (subMenu.Visible == false)
            {
                hideSubMenu();
                subMenu.Visible = true;
            }
            else
                subMenu.Visible = false;
        }
```

For each button I have to describe the above methods to represent the functionality. Let’s see the representation in the flowchart.

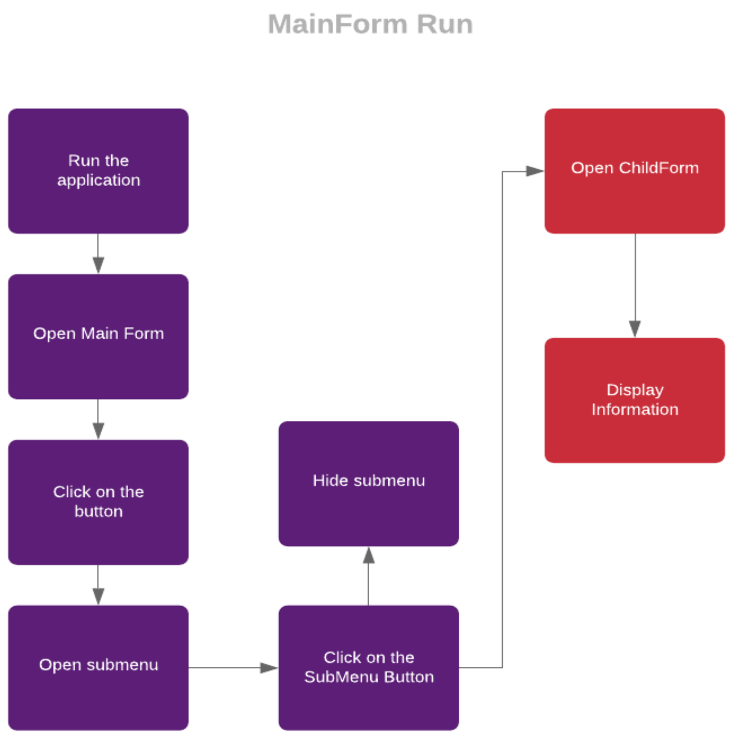


Figure 13 MainForm Run

4.2 Classes:

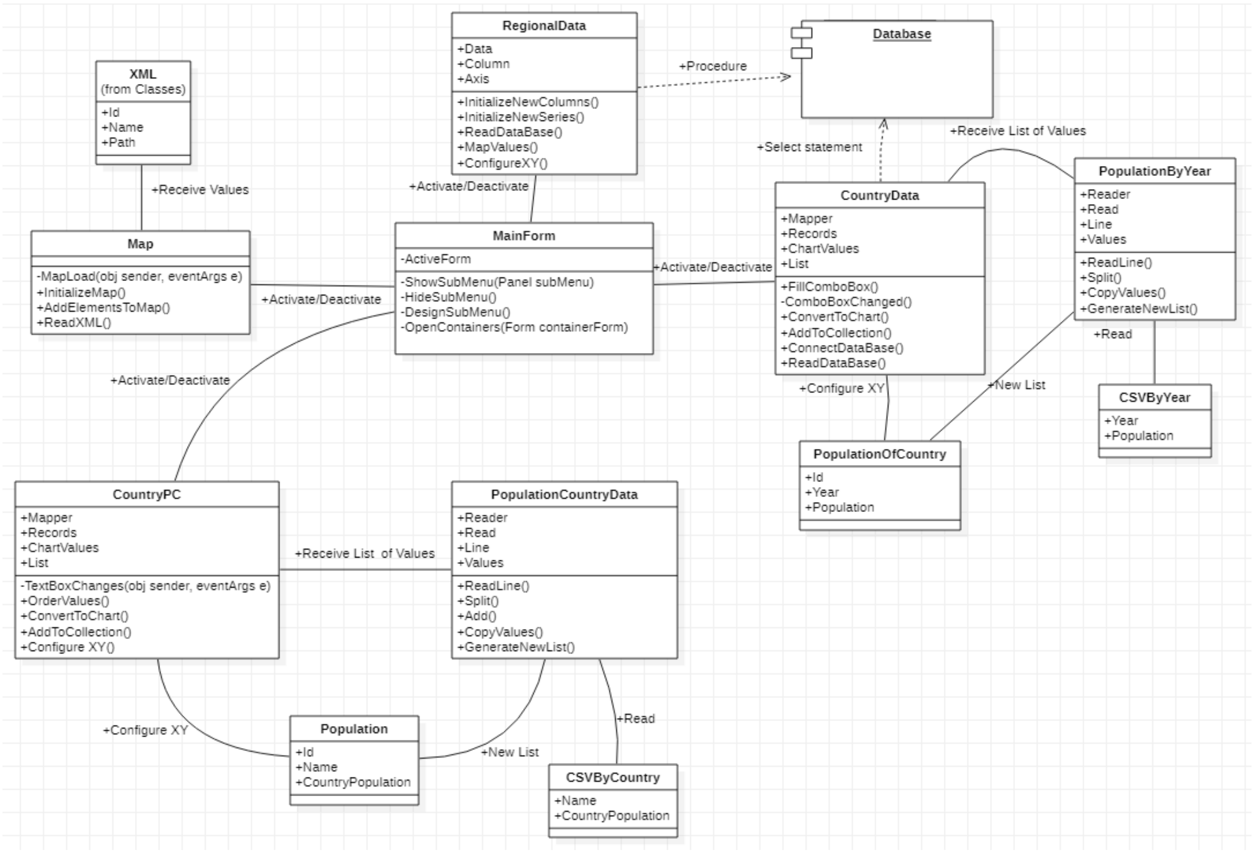


Figure 14 UML Class Diagram

In Figure 14, you can see all active classes used to represent the functionality of the application during the work. The MainForm can run all other classes, as it has the functionality to run all related forms. After the initialization of any of the other forms, the system starts to work with related to the selected class other classes.
All classes are connected between each other with an association line and can exist even if the related to the class is removed but removing the class will decrease the functionality of another class and will trigger errors.

There are two classes that exist in the UML Class diagram: “CountryData” and “RegionalData”. They are connected to the Database and dependent on the Database. This means once I change the values from the database, it will affect the values stored in the class, due to the dependency.

Association between the main form is called “Activate/Deactivate” which means after each run of the new form it will deactivate the active form.

```
if (activeForm != null)
                activeForm.Close();
```

After it proceeds with opening the new form in the child's main form panel by adding the new form to the main panel and bringing the visibility of the form to the front. This proceeds with the next code.

```
panelContainer.Controls.Add(containerForm);
            panelContainer.Tag = containerForm;
            containerForm.BringToFront();
            containerForm.Show();
```

4.3 Map of Population

Let’s check the initializing the map of population button as it was the first what I did in the application. I’ve used Nuget Package “LiveCharts.GeoMap” to easily represent data in the new form for Maps. When I’ve tried to use the elements panel to add the map element to the form, the system showed me an error without any description of it. Seemed like the issue was with a package that was downloaded. As a workaround, I’ve described the system to use the element every time when the user wants to see the Geographical Map.

```
private void Map_Load(object sender, EventArgs e)
        {
            LiveCharts.WinForms.GeoMap g = new LiveCharts.WinForms.GeoMap();
```

The NuGet package provides accessibility to the XML schema for constructing the Map. The XML schema contains the ID, Name, and Path for MapShape. At this step the class access to the external resource for constructing the map.

```
<MapShape>
      <Id>AF</Id>
      <Name>Afghanistan</Name>
      <Path></Path>
</MapShape>
```

As the first step of the project, I’ve used a simple method to represent the map. It was a collection of keys and values where I used the IDs from the XML file and assigned to the ID the value after connected to the XML file and filled the whole form with a Map.

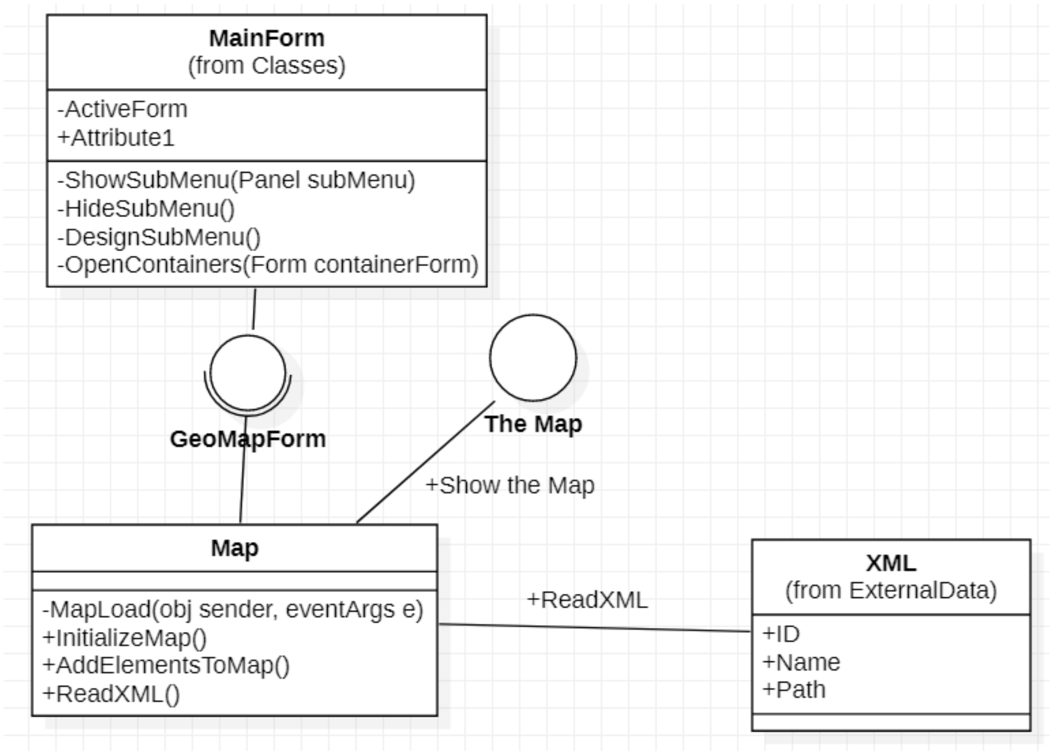
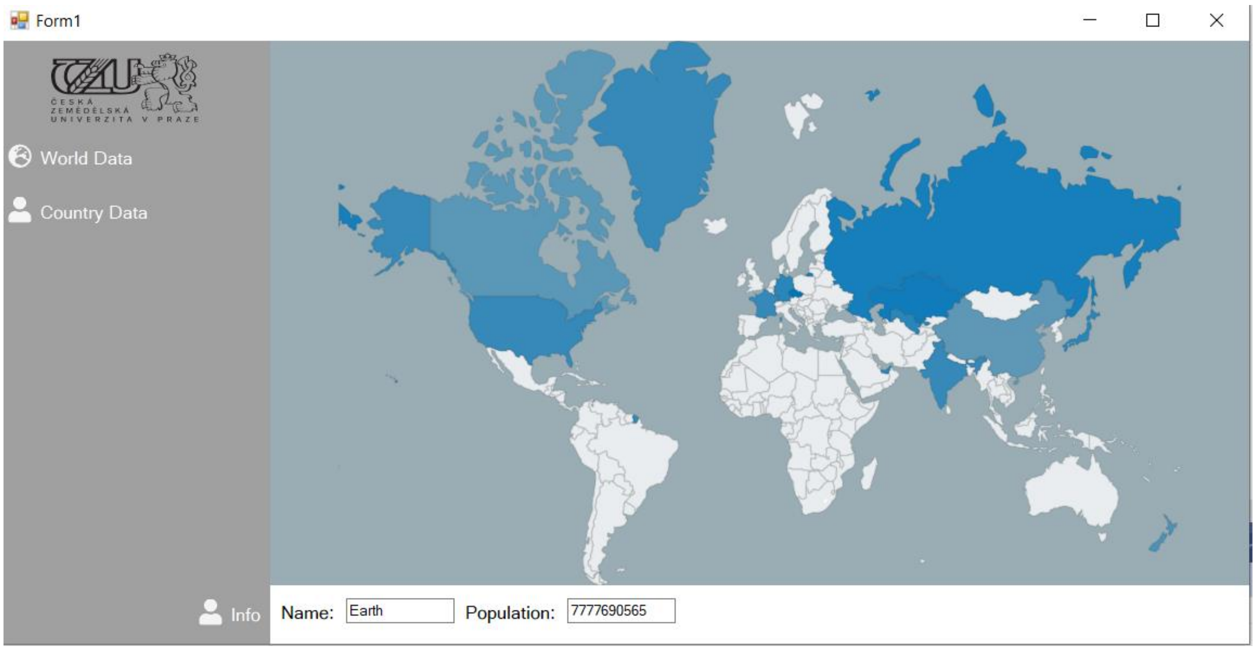


Figure 15 GeoMap Initialization UML Diagram

Figure 15 Represents class diagram with an interface realization.
Figure 16 is a sample of the Map.



Name: Earth Population: 7777690565

Figure 16 GeoMap Initialization Form

The GeoMap doesn’t contain the functionality of representing the values from the external resources, but the only XML for representing the map. The Map can be used in describing the Pollution in the world as a sample.

4.4 Country Population Chart

GeoMap was simple for implementing if I will not include the error with a package and connections to the CSV files and databases. Let's go deeper and see the manipulation with a CSV file.

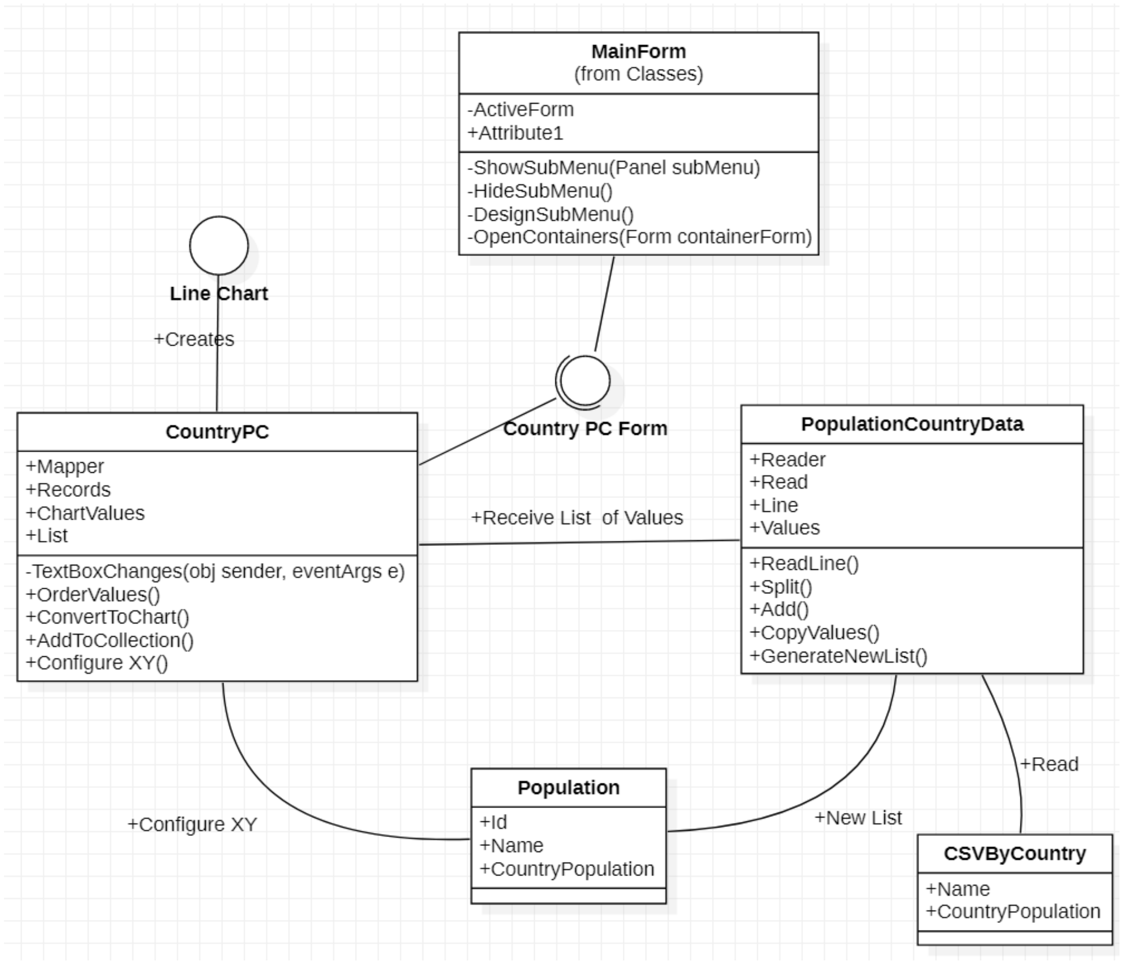


Figure 17 CountriesDataInitialization UML Diagram

The above image describes the class relation of the initialization of the new form when the "Country Population Chart" button selected.

4.4.1 CSV Parsing

For retrieving values from the CSV file I've created new classes one for operations and the second for storing values.
As you can see in the above figure 17, the diagram has a class "Population" with its attributes and "PopulationCountryData" with its attributes and operations.
For retrieving the values from the CSV file the class has objects "reader" and "read". "Reader" is needed to initialize a new instance of the class and "read" for generating a new list of "Population" class. For retrieving all Values from the file I've used them while loop with an "if" condition.

```
while (!reader.EndOfStream)
            {
                var line = reader.ReadLine();
                if (line != null)
                {
                    var values = line.Split(',');

                    read.Add(new Population
                    {
                        Name = values[0],
                        CountryPopulation = double.Parse(values[1],
CultureInfo.InvariantCulture),
                    });
                }
            }
```

It can be represented in the flowchart like in the figure below:

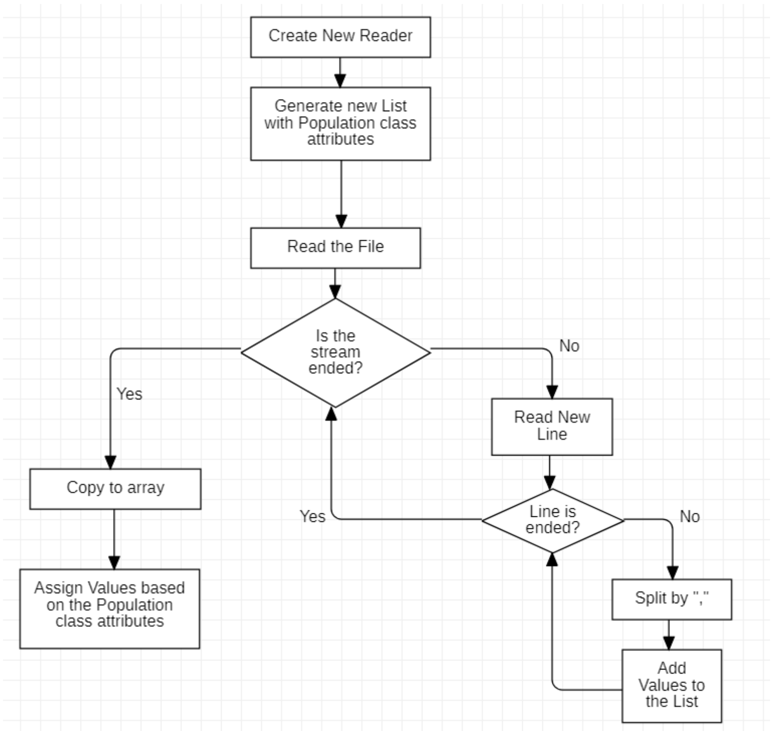


Figure 18 CSV Reading

In the above code, you can see the functionality to read CSV files. While "reader" reads the file, the system is generating a new line with assigning all values. In the IF condition, the system checks if the line in the file is not null. If it is not null, the system splits the values inside the line by checking the character "," and in the "read" object it is adding to the population class the new Values.

During the implementation of this functionality, I've faced with the Globalization error.

```
CountryPopulation = double.Parse(values[1], CultureInfo.InvariantCulture),
```

Globalization provides information about a specific culture. The information includes names for the culture, formatting the calendar, dates, numbers.
The Culture Info provides information about the culture, language, region, sublanguage, access to country-specific instances. To resolve it I'm sending to the system the information that culture info doesn't have to be dependent on the languages or region parameters "InvariantCulture".

After the end of the CSV parsing, the class is declaring a new Population List with all its values in the Population Class.

4.4.2 Countries Data Initialization

When the user selects the button, the system starts to initialize the form and creates a new object for the form to receive values from the CSV file. In the "CountriesDataReportInitialization" I have a Mapper and Records attributes. They are needed to receive values from the new list of Population attributes after the operation of getting values from the CSV file is completed.

```
var mapper = Mappers.Xy<Population>()
                .X((population, index) => index)
                .Y(population => population.CountryPopulation);

var records = PopulationCountryData.Countries.OrderByDescending(x =>
x.CountryPopulation).Take(5).ToArray();
```

The Mapper receives indexes to map them into the AxisX and AxisY line. The records are needed to store the values and build a chart. After proceeding with storing records, the class is transferring the collection of values into the chart values and setting names for each value of the item.

```
Results = records.AsChartValues();
Labels = records.Select(x => x.Name).ToList();
```

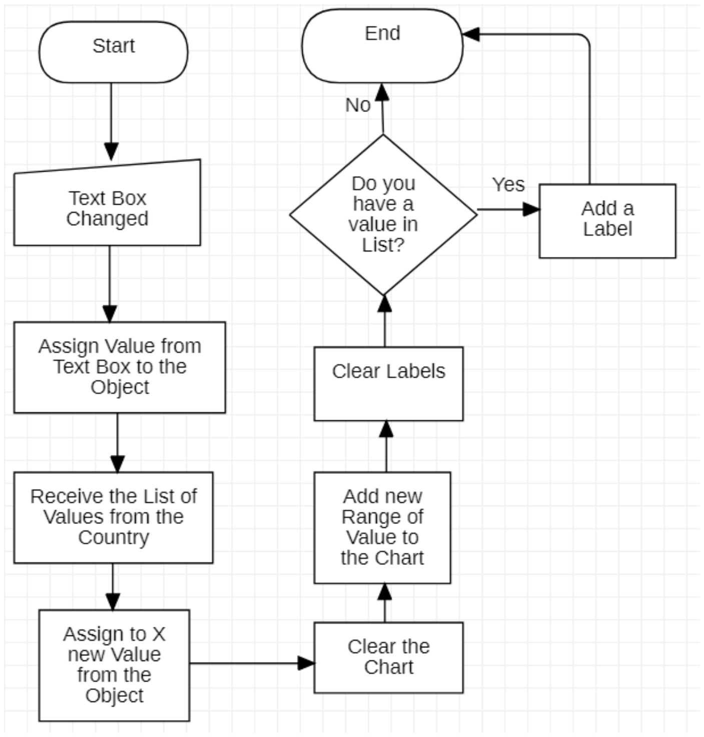
For constructing the chart, I have to set parameters, what values have to be included in the AxisX and AxisY. As a default for all charts, the AxisX contains names and AxisY contains values to represent the difference between objects.

```
            cartesianChartCP.AxisY.Add(new Axis
            {
                LabelFormatter = value => (value / 1000000).ToString("N") + "M"
            });
            cartesianChartCP.AxisX.Add(new Axis
            {
                Labels = Labels,
                DisableAnimations = true,
                LabelsRotation = 20,
                Separator = new Separator
                {
                    Step = 1
                }
            });
```

At final the class is receiving values from the new list of Population that was passed by the special class for parsing, send the values to the Results and Labels for reconstructing the list of values into the chart values and generating the new chart.

4.4.3 TextBox Changes

For the form I've implemented another functionality to check the data by the user input.



Start
Text Box Changed
Assign Value from Text Box to the Object
Receive the List of Values from the Country
Assign to X new Value from the Object
Clear the Chart
Add new Range of Value to the Chart
Clear Labels
Do you have a value in List?
No
Yes
Add a Label
End

Figure 19 TextBox Changed FlowChart

The TextBox Changed represents the functionality when the user updates the value in the textbox to check the data for a specific country.

When the TextBox changed the TextBox field sends the operation to the engine that "TextBoxChanged". After the receiver takes an action to update the Chart with new values.

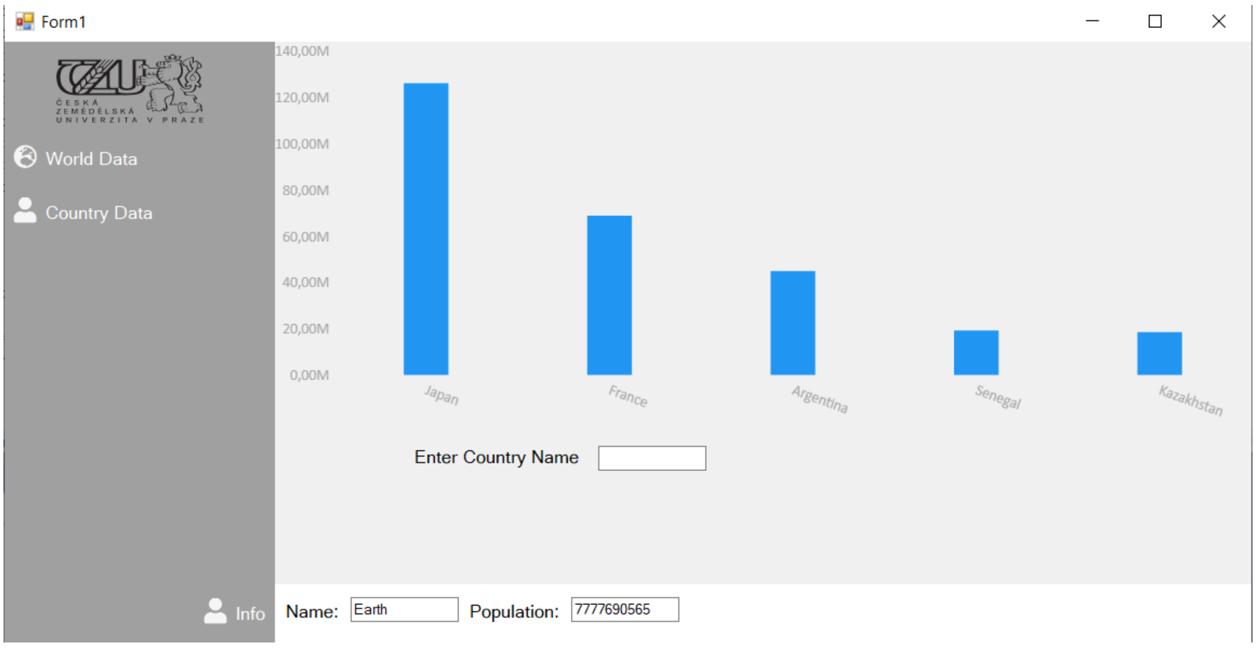
The "TextBoxChanged" method has senderValues and records attributes. SenderValues receive information from the external input (User). The System takes a command to update values to the Upper Case.

```
var newValue = ((TextBox)sender).Text ?? string.Empty;
            newValue = newValue.ToUpper();
```

"Records" is an object type attribute that contains information about all values in the CSV file. Because the "PopulationCountryData" from Figure 17 is public I can access the class and update the code by the needs. So, The Records object updated the Values line with a value from the SenderValues, after clear the Results and add new values to the collection and notifies the change to the Chart. The previous explanation describes value updating. If I want to update the name in the X line, "foreach" functionality helps to resolve it.

```
foreach (var record in records) Labels.Add(record.Name);
```

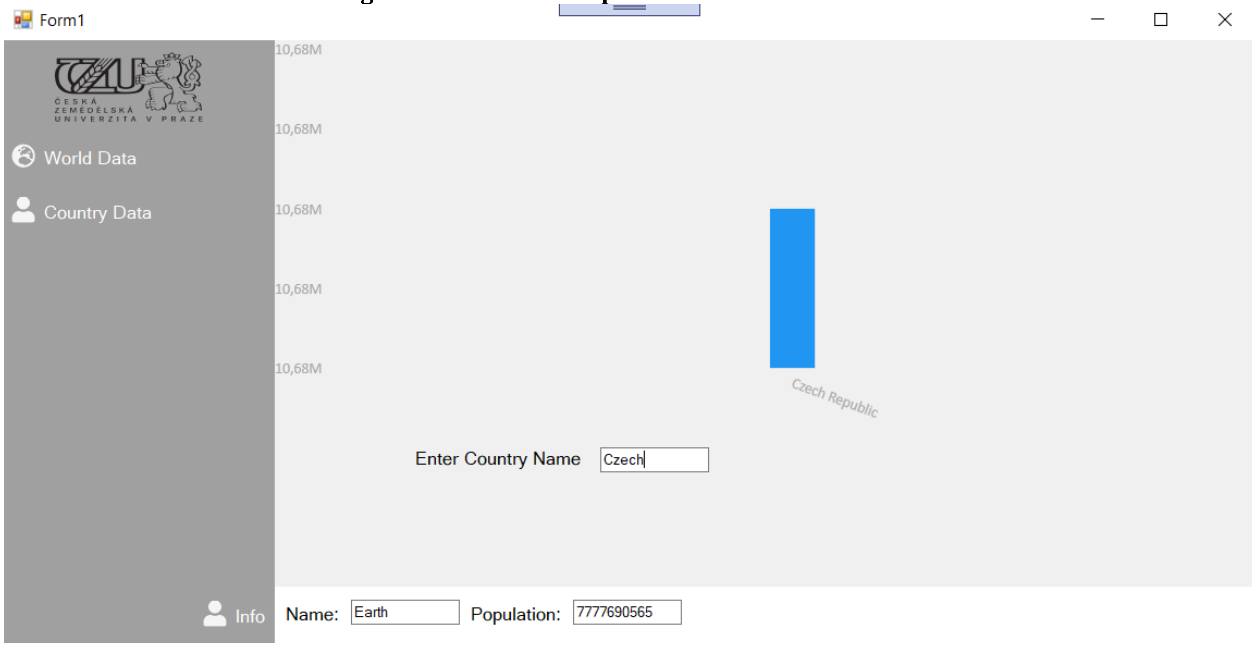
The next figures represent the sample of above functionality described in the 4.4 Chapter.



Enter Country Name

Name: Earth Population: 7777690565

Figure 20 Countries Population Initialization Form



Enter Country Name Czech

Name: Earth Population: 7777690565

Figure 21 TextBox Changed

4.5 Region Data Form

Previously I’ve connected to the XML for representing geographical maps and CSV to receive values from the files. At the population data form, I’ve implemented the database, for representing values in the charts.

4.5.1 DataBase

For representing values from the database, I have to connect it to the project. Visual Studio allows to work with database inside the environment, connect it to the project and manipulate with the data.

I’ve added a new database to the Project and created several tables for representing values.

```
CREATE TABLE [dbo].[Earth] (
    [Id]         INT          NOT NULL,
    [Name]       VARCHAR (30) NULL,
    [Population] BIGINT       NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);

CREATE TABLE [dbo].[Region] (
    [Id]         INT          NOT NULL,
    [Name]       VARCHAR (30) NULL,
    [Population] BIGINT       NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);

CREATE TABLE [dbo].[Country] (
    [Id]         INT          NOT NULL,
    [Name]       VARCHAR (30) NULL,
    [Code]       VARCHAR (30) NULL,
    [Capital]    VARCHAR (30) NULL,
    [Population] BIGINT       NULL,
    [RegionID]   INT          NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_Country_ToEarth] FOREIGN KEY ([RegionID]) REFERENCES [dbo].[Region]
([Id])
);
```

As you can see in the above code, I have 3 tables.
All the tables contain the ID as a primary key and can’t be null, names and population.
The “Country” table contains a foreign key to make a connection to the “Region” table.
This means one Region can have different countries, but the country can have only one region.

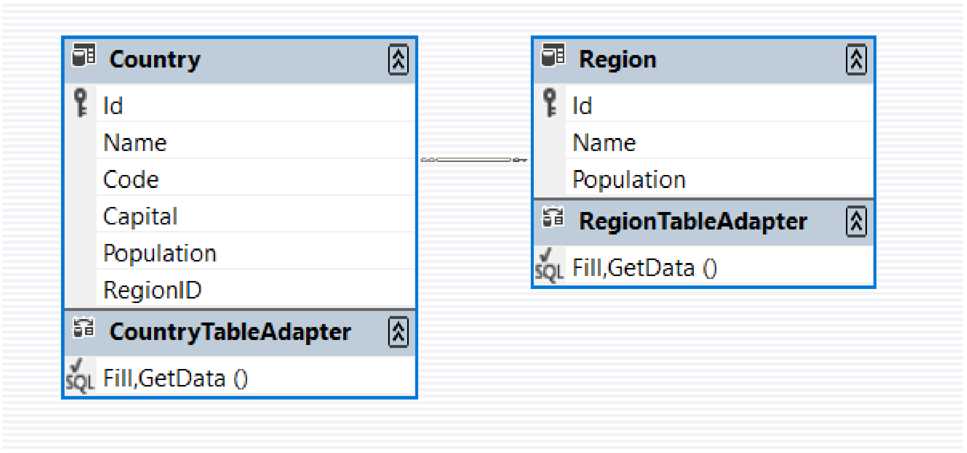
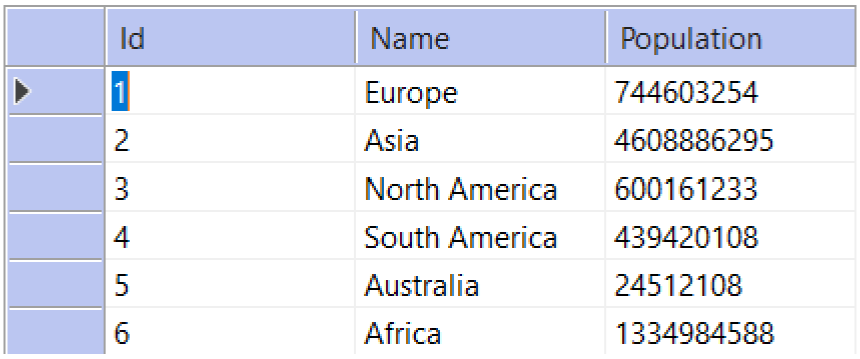


Figure 22 ERD Country Region

Lets add some values to the table Region:



Id	Name	Population
1	Europe	744603254
2	Asia	4608886295
3	North America	600161233
4	South America	439420108
5	Australia	24512108
6	Africa	1334984588

Figure 23 Region Table Values

In the population data form, the system shows the charts about the regions. I have several ways to access the region table. The first one is connecting to it directly by implementing a path or adding a „using“ functionality.

In this form, I’ll connect to the database by „using“ functionality and will use the procedure with the „Select“ statement.

```
CREATE PROCEDURE [dbo].[RegionPopulation]
AS
	select Name as Name, Population as Population
from Region
```

After creating the procedure, I can use it in the project.

4.6 Region Data Initialization

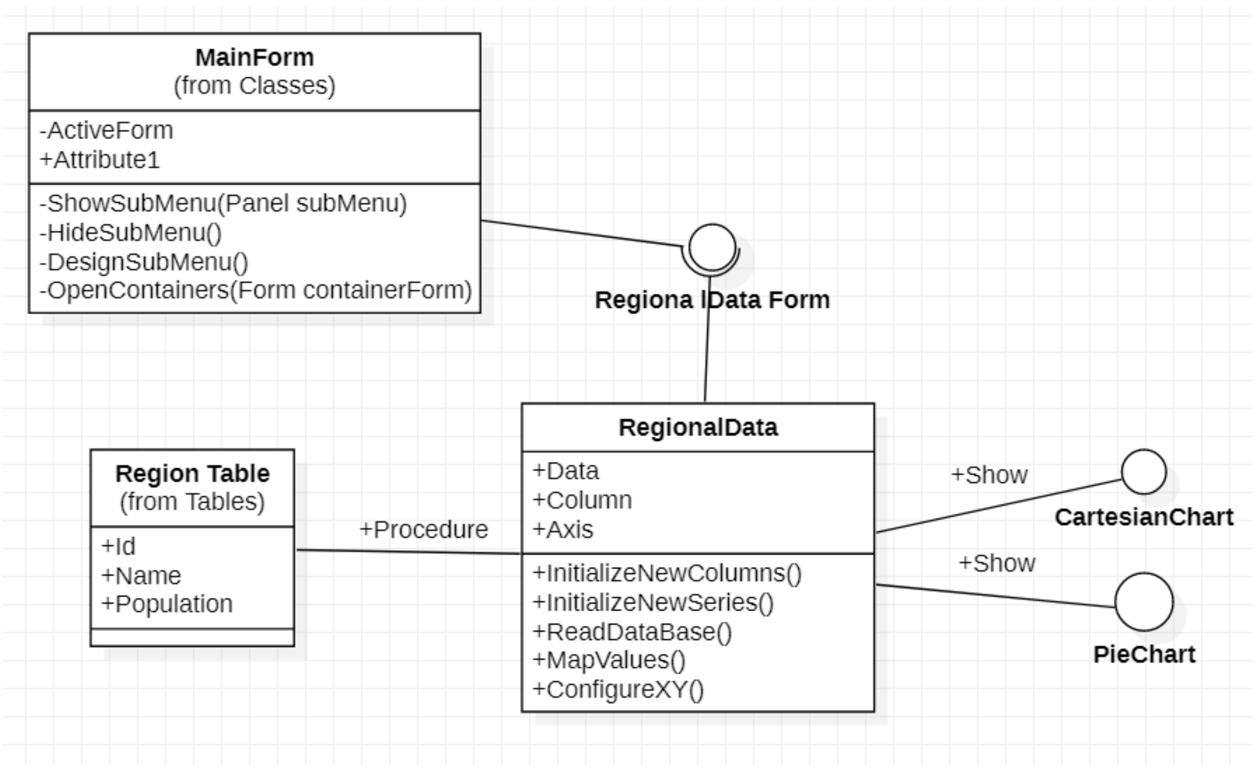


Figure 24 Region Data Initializaition UML Diagram

Figure 24 represents the class diagram of the Regional Data with interface realization.

First, the system initializes the process and adds all its components.
After I've used "using" statements to work with values from the database. You can check it in Figure 33 (Appendix).

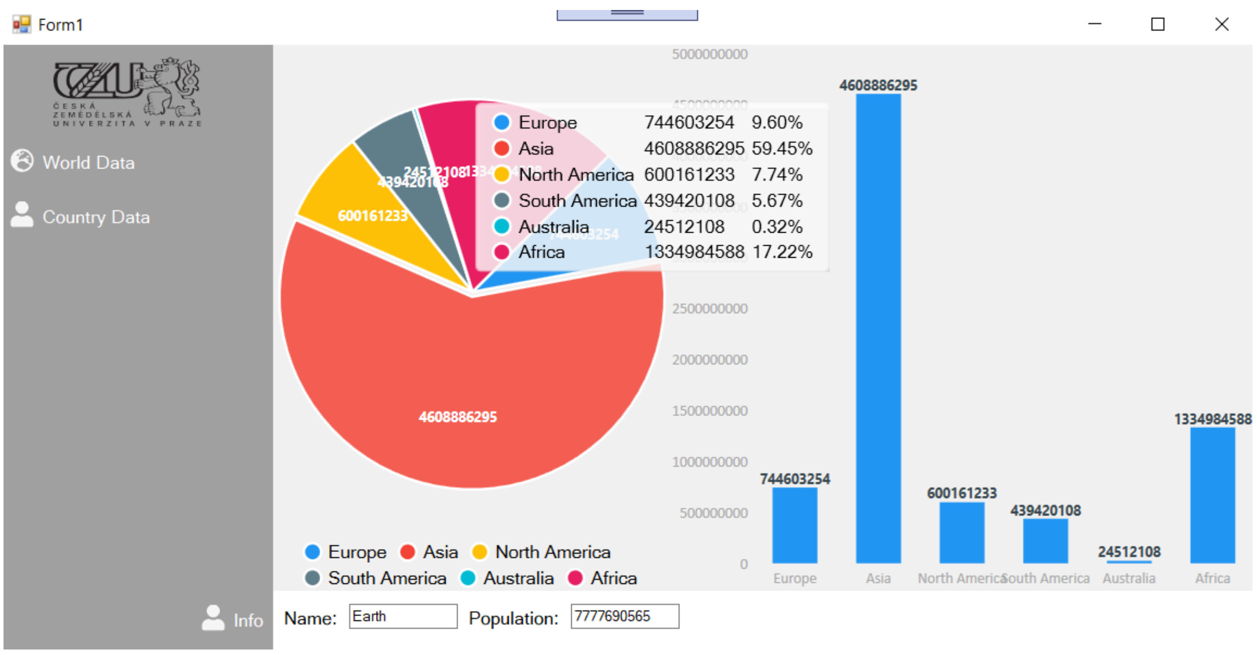
In the code, the class is declaring the new local database with values from the selected procedure. The procedure is connected to the project and the class can access it by declaring a name.

I have an object "data" that contains all values from the procedure.
After that, the class is initializing the properties for the new columns. All columns will contain values as long type because I've declared the "population" row as "maxint" in the database. The new Axis contains a separator that will separate each region.

After I've declared the foreach statement with the next expression: "foreach there are values in the procedure the system will add values and labels for that series".

The same functionality with a little change will proceed with a PieChart.

As a result, you can see the below screenshot.



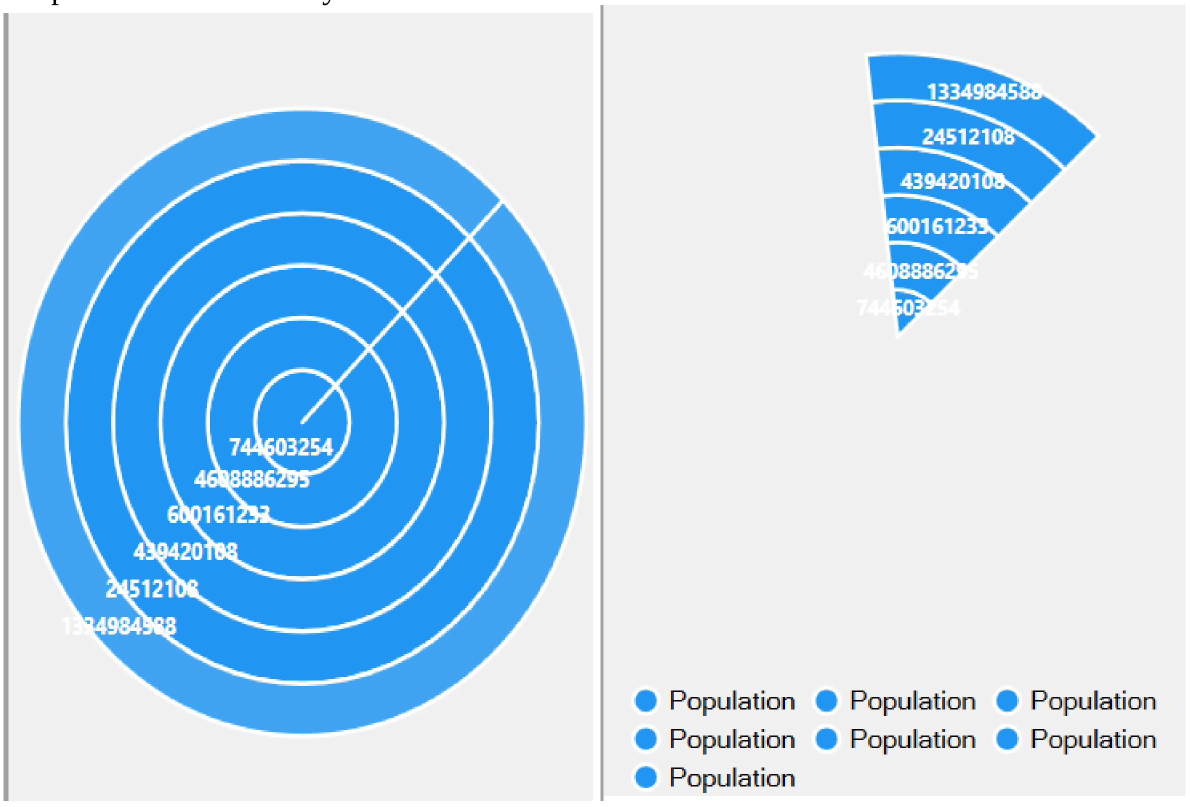
Name: Earth Population: 7777690565

Figure 25 Screenshot for Region Poopulation Data

4.6.1 Test

During testing the functionality of the charts I’ve faced an issue for PieChart.

When I had first implemented the chart with columns, I’ve tried to use the same functionality for the pieCharts and below you can see results.



The issue was with declaring the series in the pie chart. This means that in the foreach loop the class is assigning a new series to the pie and give them values, in the screenshot on the right side I've tried to use to declare series in the foreach loop.

At the final stage, I instructed the system to contain only one series collection and in the foreach loop assign to the series collection new series with values and the result was correct. We've received the correct PieChart as in Figure 25.

4.7 Country Population Form

The country population form contains functionalities with retrieving values from a CSV file using the values from the Database. It contains a direct connection to the database as the second way for populating values in windows forms.

4.7.1 Country Population Initialization

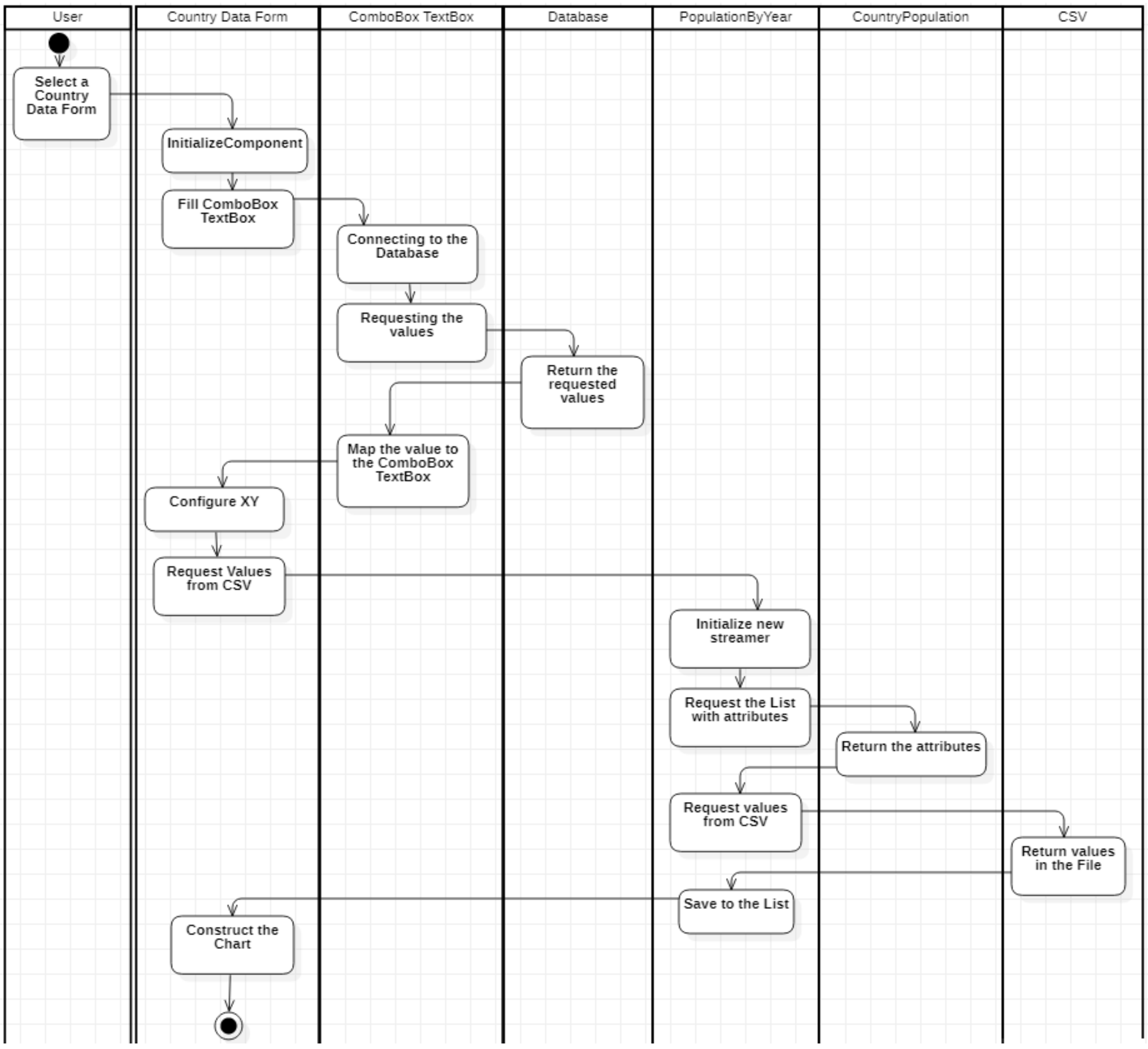


Figure 26 Activity UML Diagram

The Activity diagram above shows the process of generating and constructing the Chart, TextBox. In this form, I've used the database connections and CSV files combined, which are described in the previous pages.

In the form, I have Combobox, textbox, and chart for representing population by years.
The ComboBox has a direct connection to the database. It loads the Code values from the Country Table. Referring to Figure 34 (Appendix) you can see the second way of connecting to the database.

The code contains the connection to the database, a select statement from the database try; catch constraint and while constraint.

- Try; Catch means to try operations provided in the brackets and catch if something went wrong. As you can see in the code the class is trying to access the database. "myreader" is SqlDataReader that allows executing the database table.
- While constraint means until "myreader" ended the reading the class will receive the values from the cnn(connectionString) and add them to ComboBox.
- Catch – if we received any issues during the "try", we will receive a popup message with an exception.

In the next step, I wanted to populate values in the textbox using the changes of values in the Combobox.

It is the same procedure for connecting to the database. I have to receive all values from the table and assign them to the appropriate textbox as described in the below code.

```
String sql = „select * from Country where Code = ‚“ + codeComboBox.Text + „‘;“;
```

The next code is receiving values.

```
String name = myreader.GetString(1);
string capital = myreader.GetString(3);
string population = myreader.GetInt64(4).ToString();
string regionID = myreader.GetInt32(5).ToString();
string code = myreader.GetString(2);
nameTextBox.Text = name;
capitalTextBox.Text = capital;
populationTextBox.Text = population;
regionIDTextBox.Text = regionID;
```

4.7.2 ComboBox Changes

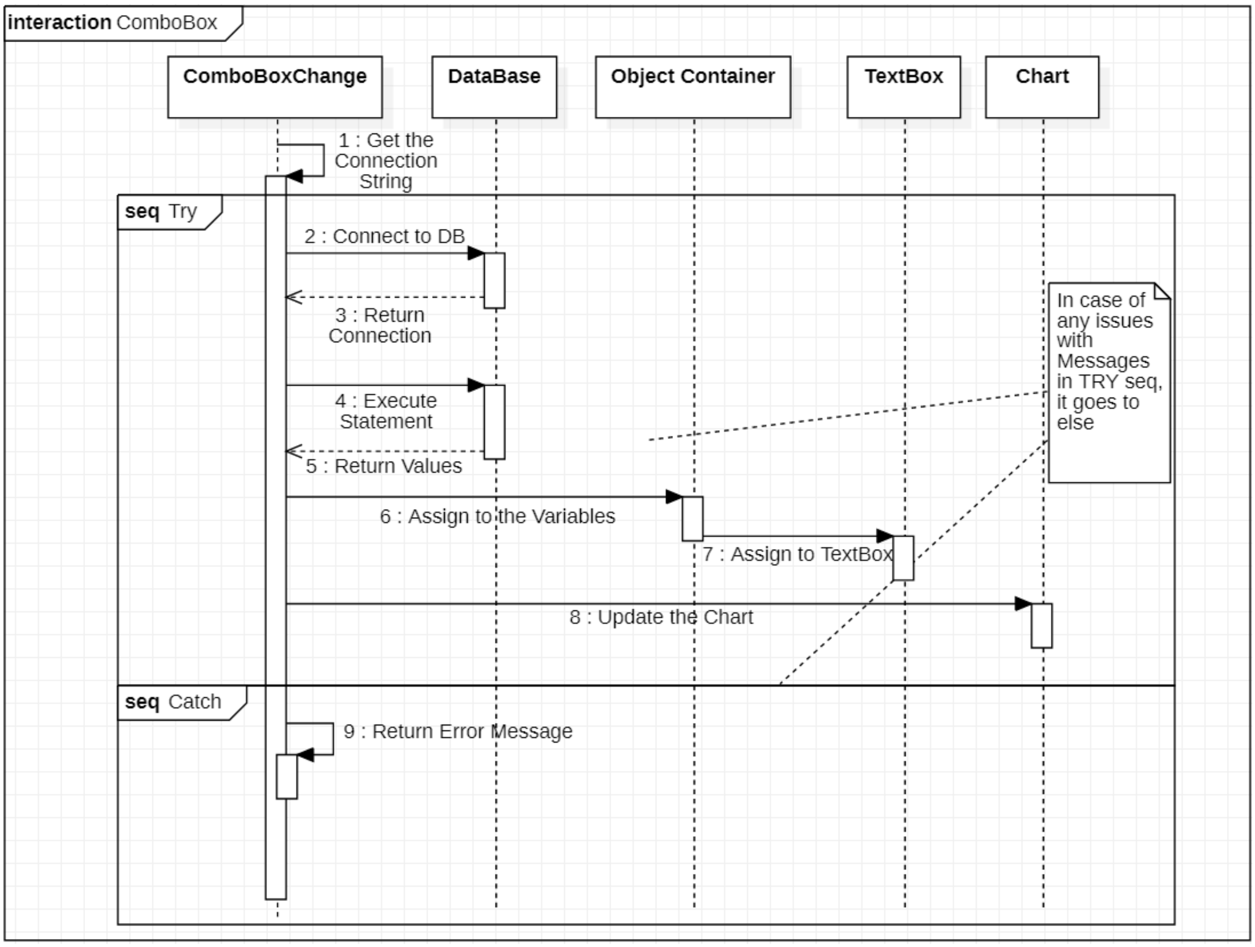


Figure 27 ComboBox Changed UML Sequence Diagram

While working with updating the charts using the Combobox in real-time I’ve declared new series each time when Combobox changed.

The problem was that the implementation was the same as it was for Countries Population layout and the special class that controls files was static and not public.

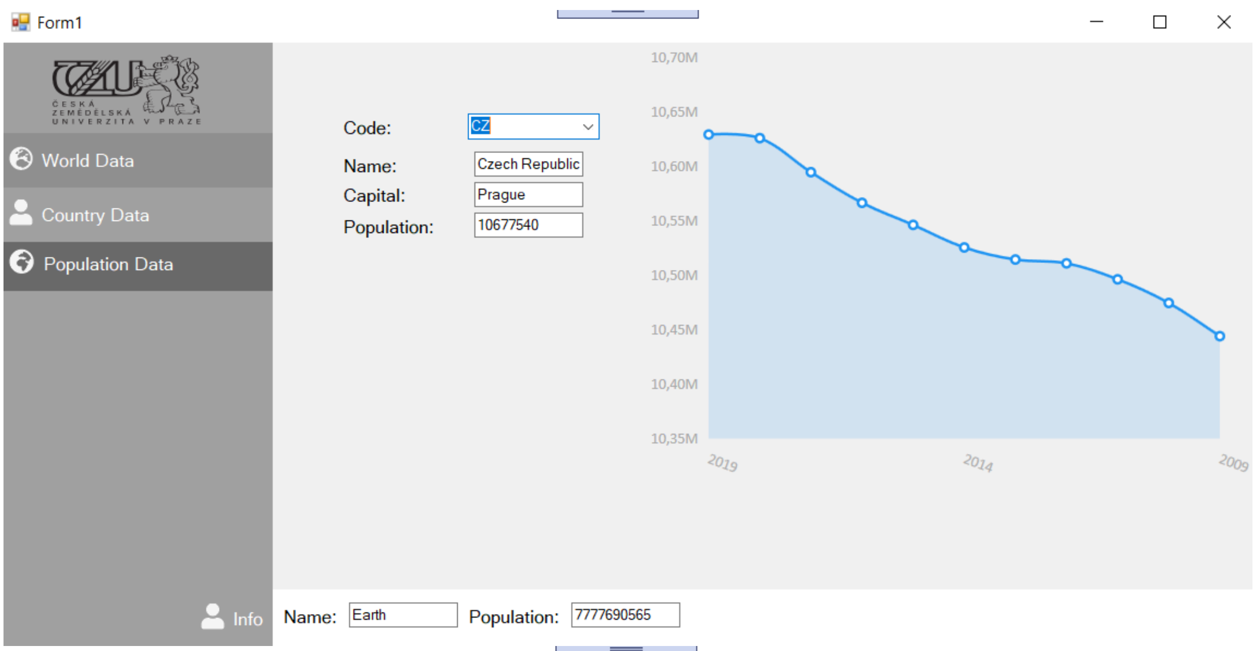
During the working, I’ve declared the class as public and gave to the class an attribute for storing value.

```
Public static void GetCSV(string Name)
```

Now the implementation says that each time when the value from the select statement changes, update the file in the project with a new file and reconstruct the Chart.

```
Var reader = new StreamReader(File.OpenRead(@“country“+Name+“.csv“));
```

Below you can see the examples from the application.



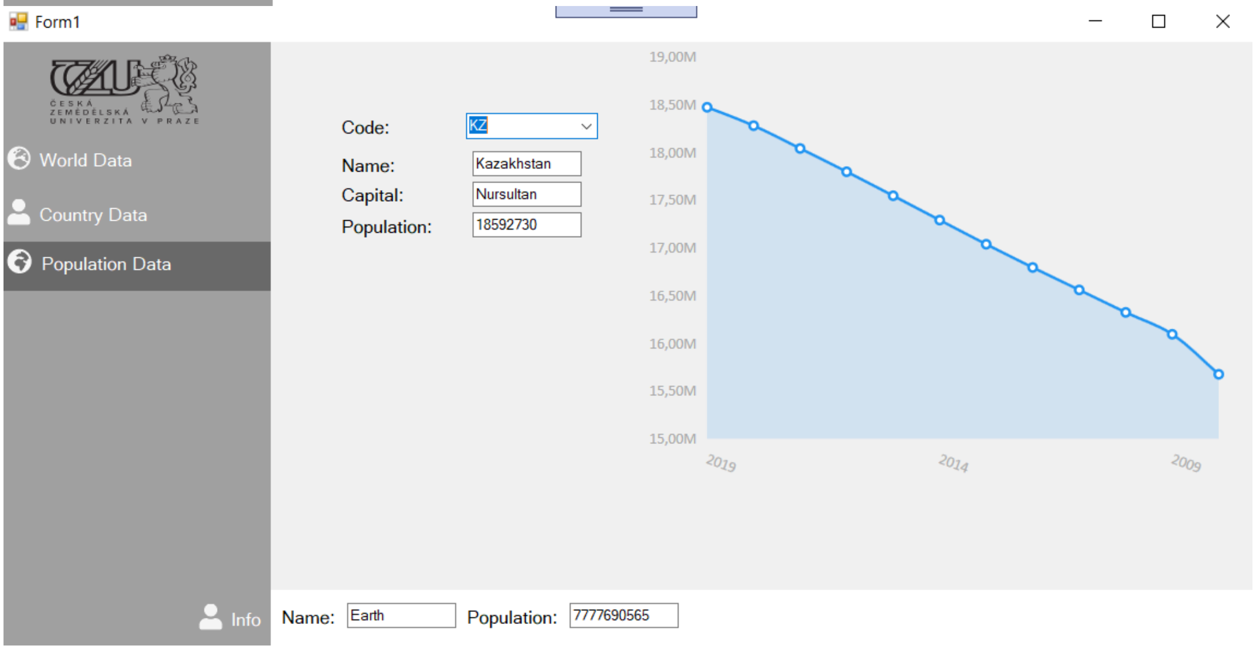
Code: CZ

Name: Czech Republic

Capital: Prague

Population: 10677540

Name: Earth Population: 7777690565



Code: KZ

Name: Kazakhstan

Capital: Nursultan

Population: 18592730

Name: Earth Population: 7777690565

4.8 The Overview

In this chapter you can see all UML work. I’ve divided the project into package folders.

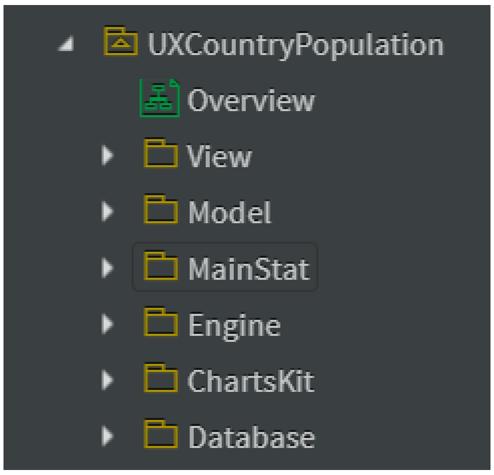


Figure 28 UML Folders

Each folder contains the classes, actions, views, external resources and models for project interpretation.

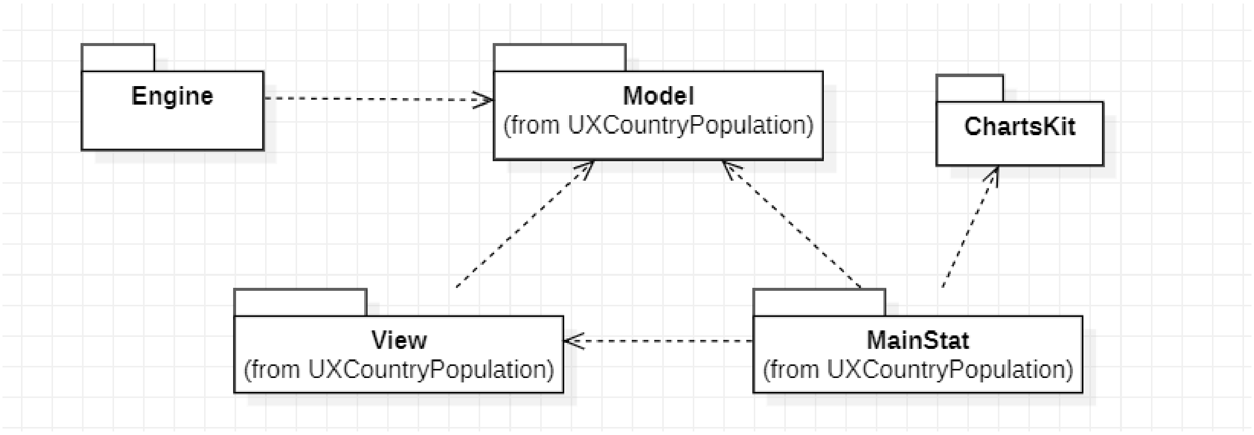

Engine
Model
(from UXCountryPopulation)
ChartsKit
View
(from UXCountryPopulation)
MainStat
(from UXCountryPopulation)

Figure 29 UML Overview

The above diagram shows how the whole procees should work in the project.

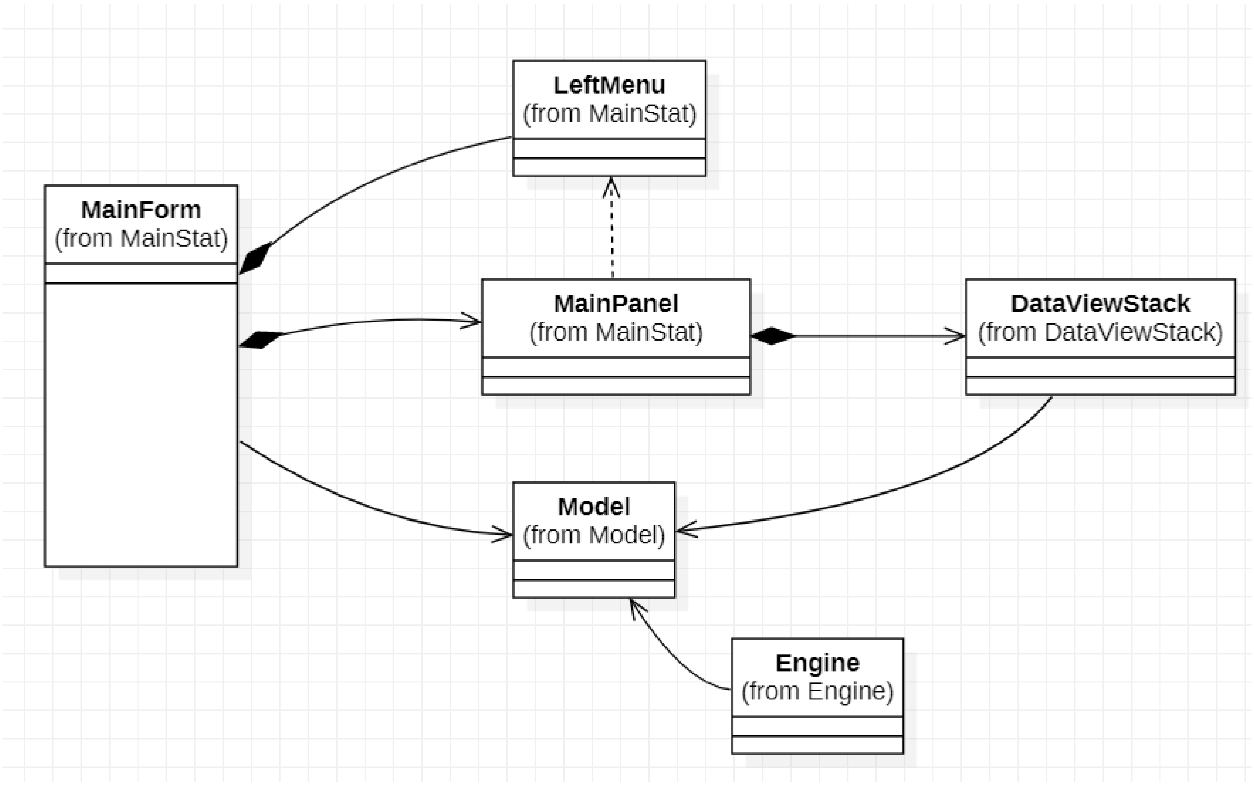


Figure 30 The UX Project Diagram

The UX Project Diagram shows the represenation of the main form and its parts.

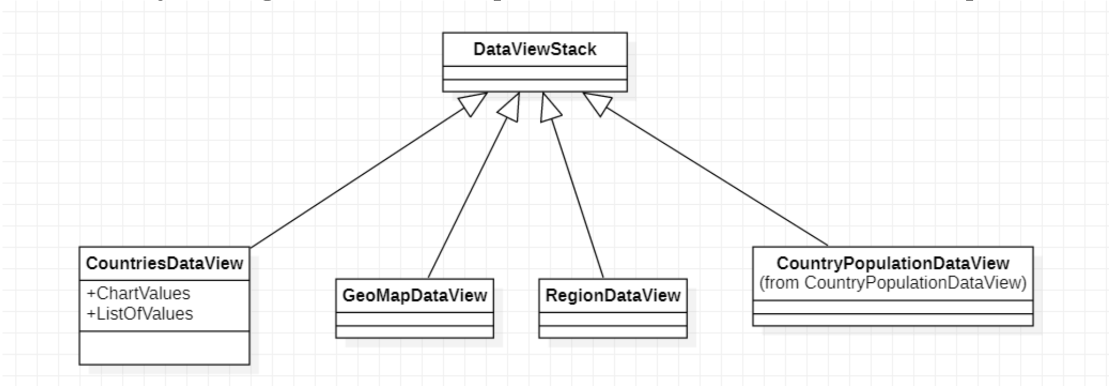


Figure 31 DataView Stack

Figure 32 is Generalization of the views in to one DataView

5 Results and Discussions

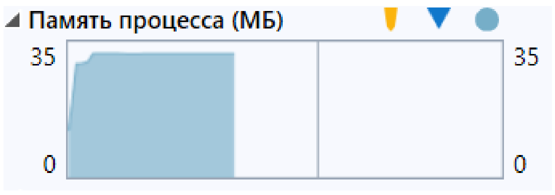
The results of the project describe how the functionality can be maintained and interpreted for users. I've used C# language to maintain the customization, code edition, database creation, manipulation with external resources using only Visual Studio. To start with any project user must have a background of knowledge in the theoretical part to understand the code. Understanding the Algorithms is must be needed for working to correctly manipulate with functionalities of the code and the UX designing with application to make it common and easy for users who will interact with an application. The understanding of the databases is needed when we want to connect it to the project and evaluate the required data.

As a result of the project, I've created an application for maintaining external resources for storing values. A new database has been created to manipulate data using different operations. The application contains the representation of values in the statistical graphs, charts, piecharts to analyze the data.

Nowadays the user expects to see the results in a much simpler way as it can be described. The application is not hard for understanding, it represents good charts instead of windows original and manipulation with editing settings is not hard for a manual user.

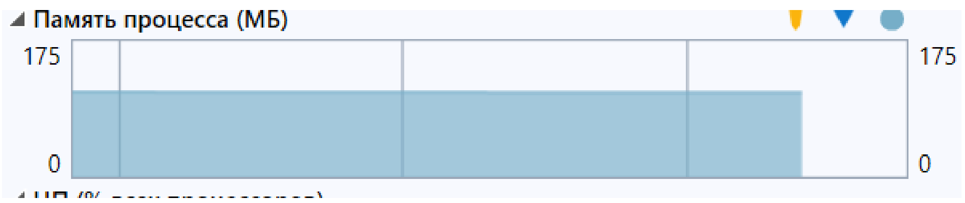
The project contains a sample for describing the population of countries, regions and can be influenced in feature to contain more data for manipulating using the same way that was described in the practical part.

During the testing of the program, I've checked the memory used for the First Initialization of program and other forms.

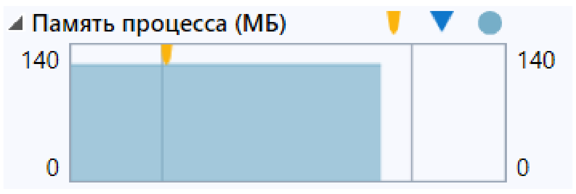


The first initialization of the program takes 35 MB of memory.

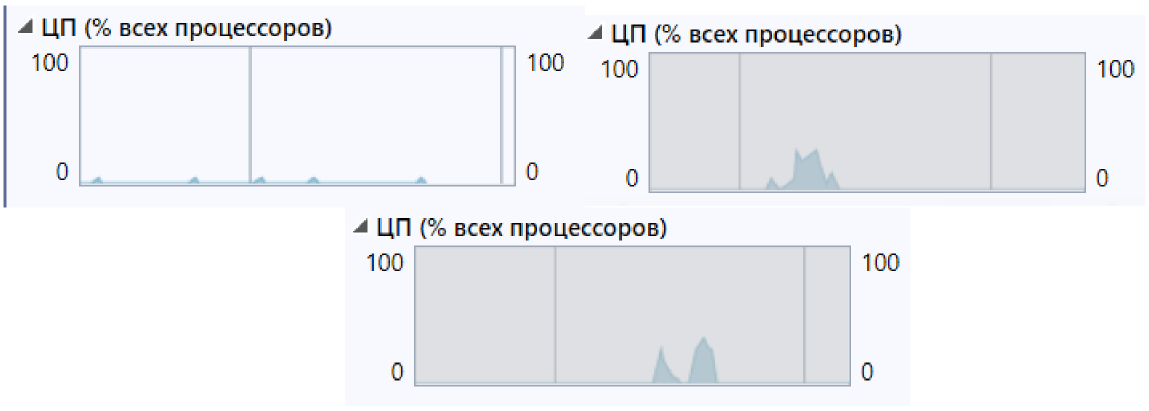
After I've tried to run different forms to capture the maximum of memory storing. You can see the details in the below screenshot. 175 MB captured as the maximum memory storing during the work with an application.



After I've cleaned the code, resolved all issues with coding and the below screenshot shows the maximum memory achieved during the work.



The CPU takes less than 25% of memory for initialiating new forms and running charts, connecting to databases and external files.



6 Conclusion

In conclusion, I want to describe the whole steps that were maintained during the work on the project.
As the main objective, I've created an application to contain information about the world and local data information of a country.
The project represents data information in analytical tools such as graphs, charts, piecharts, and geographical maps.

I went through the theoretical part to collect information on how to proceed with developing an application using C# programming language and Visual Studio and implemented it in the practical part.

During the project, I received a lot of different problems, issues and this is an interesting part of the developing the application, to debug and identify the root cause, resolve the issue or find the workaround for it.

The practical part takes much more time than theoretical because it is easy to understand the main common methods during the investigation, but when it comes to implementing the code for our purposes using different methods, functions, algorithms it becomes much more difficult. The programming takes more part of practicing than the theoretical parts.

To conclude I would like to add, that my C# programming was at a very low level at the beginning even after researching the literature part. Only after when the implementation has begun, I've started to understand how the process is going and what is used to implement the code in real life. I can say that without practice we can't implement the theoretical part for our needs.

7 Bibliography

Agarwal, V. V. (2012). *Beginning C# 5.0 Databases.* Apress.

Andre, A. a. (September 2018). Interaction Design Lifecycle Model. doi:10.1088/1757-899X/407/1/012174

Chen, P. P.-S. (1977). *The Entity-Relationship Model: A Basis for the Enterprise View of Data.* Dallas, Texas: Association for Computing Machinery.

Cox, B. J. (1986). *Object-Oriented programming: an evolutionary approach.* Cumincad.

Hejlsberg, A. (2010). The C# Programming Language. V M. T. Anders Hejlsberg, *The C# Programming Language.* Westford, Massachusetts: Addison-Wesley Professional.

Hoffman, J. (1996). *Introduction to Structured Query Language.* J. Hoffman.

Iverson, K. E. (1962). *A programming language.* New York: John Wiley and Sons, Inc.

Korth, H. F. (2010). *Database system concepts 6th Edition.* New York: McGraw-Hill.

Microsoft. (19. 03 2019). *Visual Studio Docs.* Načteno z Microsoft: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019

Mramba, B. a. (01 2018). Design of an Interactive Mobile Application for Maternal, Neonatal and Infant Care Support for Tanzania. *Journal of Software Engineering and Applications*, 569-584. doi:10.4236/jsea.2018.1112034

Perkins, B., Hammer, J. V., & Red, J. D. (2018). *Beginning C# 7 Programming with Visual Studio 2017.* Wrox.

Petkovic, D. (2008). Chapter 5: Data Definition Language . V D. Petkovic, *Microsoft SQL Server 2008 A Beginners's Guide* (stránky 109-110). McGraw-Hill Osborne Media.

Poston Jr, D. L. (2010). *Population and Society: An Introduction to Demography.* Cambridge University Press.

Preece, J. a. (1994). *Human-computer interaction.* Addison-Wesley Longman Ltd.

Robbins, R. J. (1994). *Database fundamentals.* Načteno z rrobbins@ gdb. org: rrobbins@ gdb. org

Rys, M. (2005). XML and Relational Database Management Systems: Inside Microsoft® SQL Server™ 2005. *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (stránky 959-962). ACM.

Strauss, D. (2017). *C# 7 and. Net Core Cookbook.* Packt Publishing Ltd.

8 Appendix

CTS type	Name	Description
System.Object	Object	Class, basic for all types (CTS)
System.String	String	The straw
System.Sbyte	Sbyte	8-bit byte (with sign)
System.Byte	Byte	8-bit byte (without sign)
System.S16	Ushort	16-bit number (with sign)
System.UM16	Short	16-bit number (without sign)
System.Int32	Int	32-bit number (with sign)
System.Uint32	Uint	32-bit number (without sign)
System.Int64	Long	64-bit number (with sign)
System.Uint64	Ulong	64-bit number (without sign)
System.Char	Char	16-bit symbol (Unicode)
System.Single	Float	32-bit number with a floating point
System.Double	Double	64-bit number with a floating point
System.Boolean	Bool	Boolean value
System.Decimal	decimal	128-bit type is used mainly when extremely high accuracy is required

Table 6 Types and aliases

Action	Expression	Operator	Type
Sum of variables	Var1 + Var2	+	Binary
Substracting of variables	Var2-Var1	-	Binary
Result assigns the product of variables	Var1*Var2	*	Binary
Result assigns devision of variables	Var2/Var1	/	Binary
Result assigns remainder of devision	Var2%Var1	%	Binary
Result assigns the value of varibale	+Var1	+	Unary
Result assigns the value from multiplication varible and -1	-Var1	-	Unary
Result assigns the sum of variable and 1	++Var1	++	Unary
Results assigns the substarcting of variable and 1	--Var1	--	Unary

Table 7 Mathematical Operators

Action	Expression	Operator	Type
Result is true if variables are equal	Var1 == Var2	==	Binary
Result is true if variables are not equal	Var1 != Var2	!=	Binary
Result is true if first variable is less than second	Var1 < Var2	<	Binary
Result is true if first variable is greater than second	Var1 > Var2	>	Binary
Result is true if first variable is less or equal to second	Var1 <= Var2	<=	Binary
Result is true if first variable is greater or equal to second	Var1 >= Var2	>=	Binary

Table 8 Boolean operators

```
using (LocalDBWorldCountryEntities db = new LocalDBWorldCountryEntities())
            {
                var data = db.RegionPopulation();
                ColumnSeries col = new ColumnSeries() { Title = „Population“,
DataLabels = true, Values = new ChartValues<long>(), LabelPoint = point =>
point.Y.ToString() };
                Axis ax = new Axis() { Separator = new Separator() { Step = 1,
IsEnabled = false } };
                ax.Labels = new List<string>();
                foreach (var x in data)
                {
                    col.Values.Add(x.Population.Value);
                    ax.Labels.Add(x.Name.ToString());
                }

                cartesianChart.Series.Add(col);
                cartesianChart.AxisX.Add(ax);
                cartesianChart.AxisY.Add(new Axis
                {
                    LabelFormatter = value => value.ToString(),
                    Separator = new Separator()
                });
            }
```

Figure 32 Connection to database

```
string connectionString;
            SqlConnection cnn;

            connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Айдын\source\repos\UXCountry
World\UXCountryWorld\LocalDBWorldCountry.mdf;Integrated Security=True";

            cnn = new SqlConnection(connectionString);

            string sql = "select * from Country;";
            SqlCommand cmd = new SqlCommand(sql, cnn);
            SqlDataReader myreader;

            try
            {
                cnn.Open();
                myreader = cmd.ExecuteReader();
                while (myreader.Read())
                {
                    string code = myreader.GetString(2);
                    codeComboBox.Items.Add(code);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }df
```

Figure 33 Connection to database 2