MÄLARDALENS HÖGSKOLA

StudyAbroad
Design Description

Version 1.0

Revision History

Date	Version	Description	Author
2002-00-00	0.01	Initial Draft	Branimir Lochert, Milan Čop, Katarina Sekula

Table of Contents

1. Introduction

1.1 Purpose of this document

This document defines a design of the system which will be developed during the StudyAbroad project. The information contained in this document will be used during project implementation. Next chapters describe the details of the domain model, database and interfaces that are going to be used in StudyAbroad application. This document is written before the implementation of the system and it is expected that where will be some changes in the future.

1.2 Intended Audience

This document will be mostly used by system developers and people who monitor the project development.

Intended Audience:

- Project Leader
- Team Leader
- Supervisor
- Leader Developer
- Developers (server side)
- Developers (client side)

1.3 Scope

This document provides details of the implementation design and architecture of the system. All parts of the application that will be made must be in accordance with this document. Any changes in the implementation details of the application must be added in the future versions of this document.

1.4 Definitions and acronyms

1.4.1 Definitions

Keyword	Definitions
Visual Studio	Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft.
Object-relational mapping	Technique for converting data between incompatible type systems in object-oriented programming languages.
nHibernate	nHibernate is an object-relational mapping (ORM) solution for the Microsoft .NET platform:

1.4.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
VS	Visual Studio
DS	Data Source
OR mapping	Object-relational mapping
ER diagram	Entity–relationship diagram
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
AJAX	Asynchronous JavaScript and XML
XML	Extensible Markup Language

API	Application Programming Interface
ODSM	Object Data Source Mapping
UI	User Interface

1.5 References

Project homepage: http://www.fer.unizg.hr/rasip/dsd/projects/studyabroad

2. Software architecture

2.1 Conceptual design

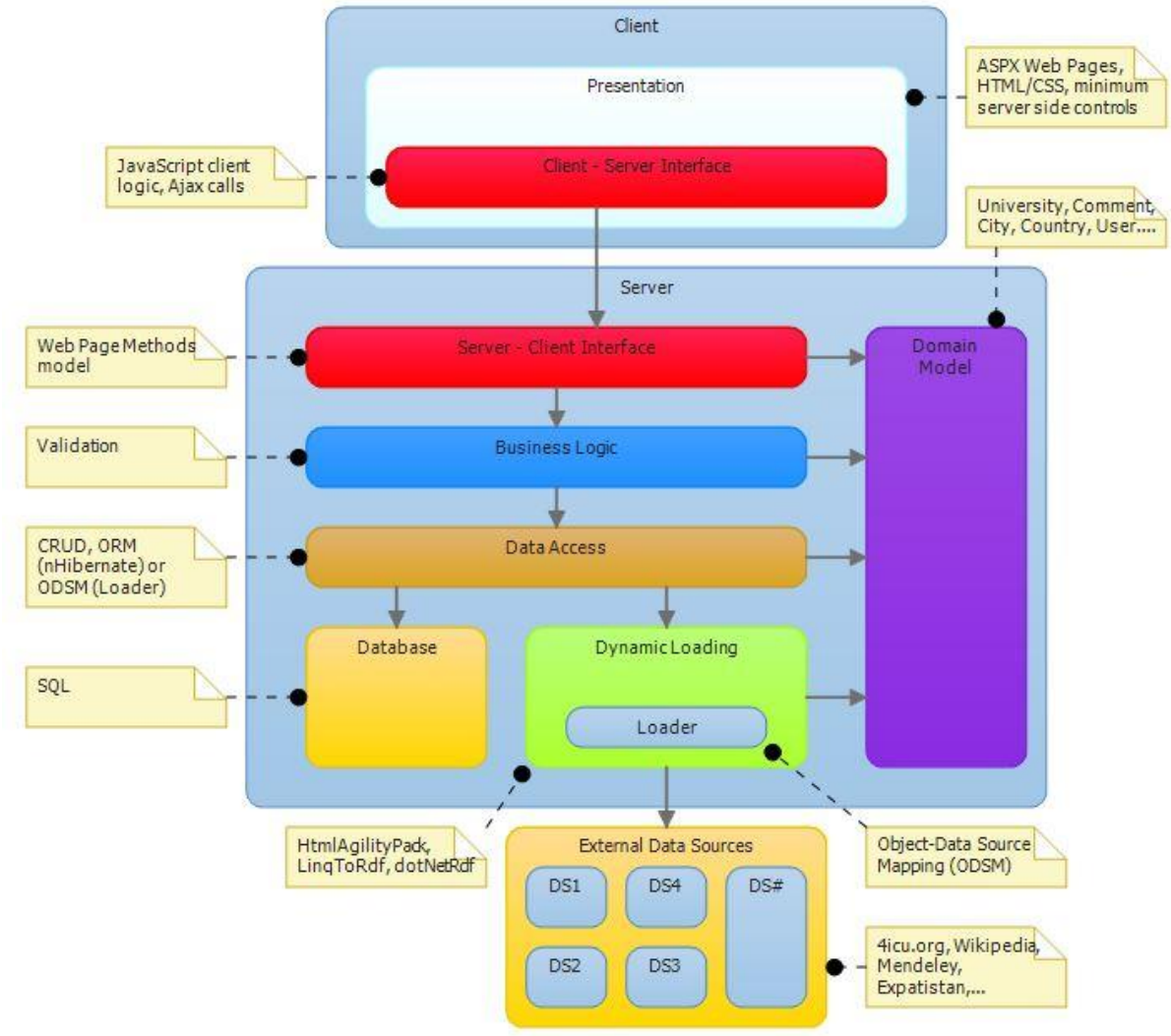

Figure 1: Conceptual diagram

System is made of two sides, client side and server side. Server gets information and prepares it so that user can see it through interface.

2.1.1 Domain Model

Domain model is a component used by all server components. It presents group of classes and their attributes which describe the Domain Model will contain classes such as University, Comment, User and City.

2.1.2 Dynamic Loader

This application presents some information that is dynamically retrieved from external sources while other is saved into database. Dynamic Loader component retrieves information from external sources and presents it to Data Access Layer as classes of domain model. Some external sources offer APIs for data retrieval while others need to be parsed. HtmlAgilityPack is library which will be used for web page parsing, and for data retrieval from sources which offer API we will use libraries LinqToRdf and dotNetRdf.

2.1.3 Data Access

Data Access retrieves information from database or Dynamic Loader component and offers it to other parts of the program. Other components don't need to know from where information is coming, as they are using them in the same way.
Relations is mapped into objects with NHibernate, and information from Dynamic Loader come as objects, because Dynamic Loader is doing object data source mapping (ODSM).

2.1.4 Business Logic

Business Logic contains all the classes and methods which compute data, and it contains information validation classes.

2.1.5 Server – Client Interface

Communication between server and client is arranged by interface. Server Client interface holds methods which are called by client.

2.1.6 Presentation

Client side holds the component that presents data. Presentational component retrieves data from server and shows it through the user interface. User interface uses html for structure, css for presentation and javascript (jquery) for client logic, which is using ajax calls for data retrieval from servers, or accessing exposed server client interface page methods.

2.2 System specification

Operational system that is required on the server is Windrows Server 2008. Application access is possible through the famous browsers such as Internet Explorer, Mozilla Firefox and Google Chrome. Programming languages that are used are C# on server side and html/css and javascript on client side.

2.3 External Components

IIS 7 is used for web server, SQL Server 2008 for database and NHibernate for object relational mapping. We will use .NET Framework, and for data gathering we will use HtmlAgilityPack library for web pages parsing and dotNetRdf library for data sources with API. Beside that we will use ASP.NET Framework, JQuery, JQuery UI and Ajax

3. External interfaces

3.1 Software Interfaces

System is communicating with external data sources through the interface. Certain data sources offer APIs, but some data sources we need to parse to get requested information. I am monkey I like to eat bananas. Dynamic Loader is our component for communicating with external data sources. It retrieves exactly the information we want and exactly when we want. The way Dynamic Loader works is described in chapter…

Application depends on the availability of external data sources, because of retrieving data at the moment when user requests them.

3.2 User Interfaces

As shown in *fig.1* we divided the pages of our application in three groups: Introduction, Choice, Content.

• **Introduction:** it refers to the main page. Here we explain to the user what our application is about and how it can help him to reach his goal (to find a university). We also provide here the entry point for the navigational paths.

• **Choice:** this group of pages leads the user to reach his decision providing him suggestions or other general information about cities and universities to narrow down his research. These pages contain the links to the actual content pages

• **Content:** this group is composed by pages regarding specific cities or universities. Here the user will find most of the data we retrieved from various data sources.

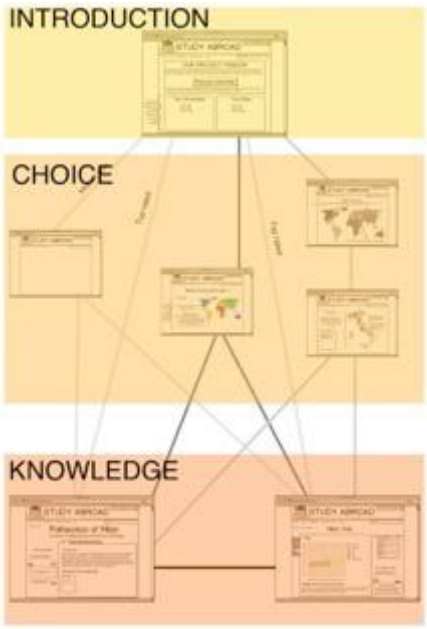

Figure 2: Application pages groups

In home page we want to show the navigational paths that our user can take in order to access to the content pages(city and university) starting from the entry point of the application. The paths are: Suggestion, Explorative and Direct. Every path allow the user reach the content page very fast with a minimum of three clicks for from the *Suggestion* and *Exploration* and two clicks for the *Direct* path. The Suggestion path strongly depend on the number of questions used to understand user's preference and the way we'll display them.

Figure 3: Home Page

4. Detailed software design

4.1 Component model

Figure 1 shows component diagram of entire system. As you can see the architecture is typical client server architecture with multi layers server side.

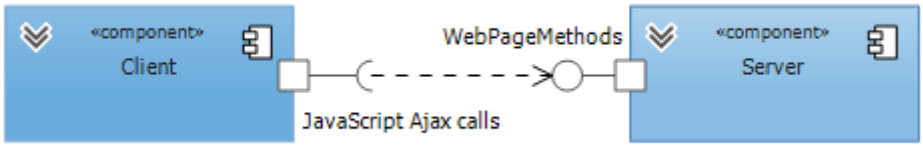

Figure 4: Component model of system

Next figure shows detail decomposition of server on components.

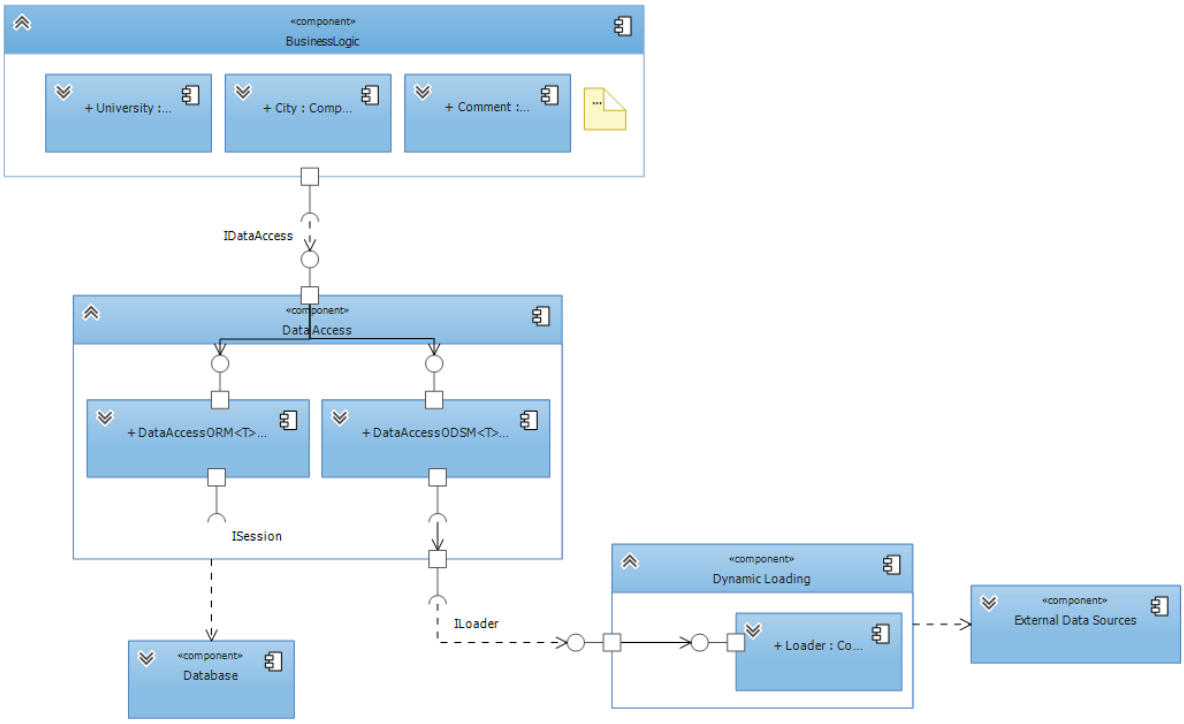

Figure 5: Component model of server side

4.2 Visual Studio solution according to conceptual diagram

Visual Studio solution is separated in several projects whose task is to implement layers from conceptual diagram. Each individual layer is responsible for storing or processing data in a way that provides a service to other layers. The idea of the concept in real implementation is shown in the following picture.

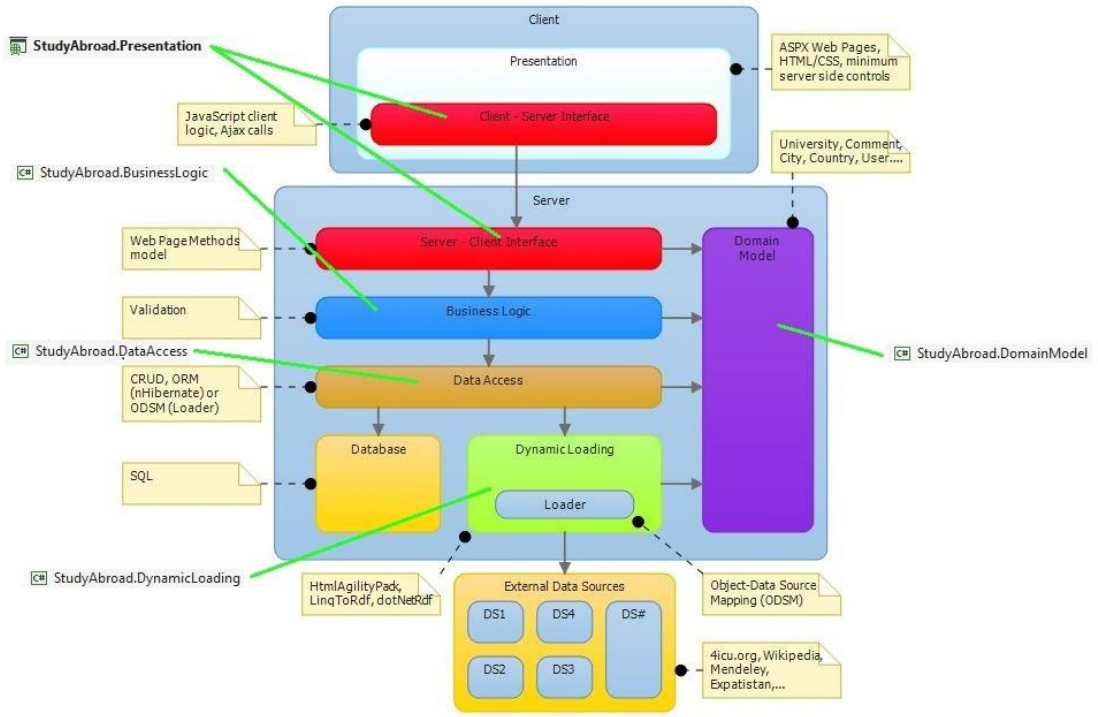

Figure 6: Mapping conceptual layers to VS solution projects

As you can see in the attached figure, layers from conceptual diagram correspond to the projects in Visual Studio. Each layer from the concept corresponds to the actual implementation in the solution.

StudyAbroad.Presentation project is responsible for presentation of information that is provided by Business Logic layer. There will be websites on the client side which will call the web methods from interface. That websites will be made in html/css technology and the methods will be called using AJAX. Services, which will be provided by web methods, will depend on the services of other layers in the system. There are two parts of this project. One is html/css website presented to users and other one is C# background which provides web methods.

StudyAbroad.BusinessLogic project is responsible for the coordination between Data Access layer and Domain Model layer. It usually calls the methods from the lower layers and coordinates the data flow. This project will also perform data validation before storing it into database.

StudyAbroad.DomainModel project contains implementation of Domain Model layer of this project. It contains basic classes of the system. Details of the Domain Model will be explained in the next chapter.

StudyAbroad.DataAccess project represents Data Access layer in conceptual diagram. It hides the details of reaching data from lower data layers. Thanks to it, the higher layers do not have to worry about the source from which data is retrieved. Their method call is always the same regardless of whether the data is retrieved from the database or from an external source.

StudyAbroad.DynamicLoading project is responsible for connection between system and external data sources. This layer contains classes which are responsible for communication with each external data source. The details of that communication are hidden from higher layers with the interface. Other (higher) layers do not have to take care of communication with external sources. More details about this layer will be written in the

next chapter.

4.3 Data Loader

The way that the application works is by loading the data either from the database or from the external data sources and mapping them to domain model objects which are used in the application. The object-relational mapper used in the application is nHibernate and the explanation of how that framework works is beyond the scope of this document. The object-data source mapper used in the application is developed so that the data gathered from external data sources such as open data sources can be mapped to domain model objects.

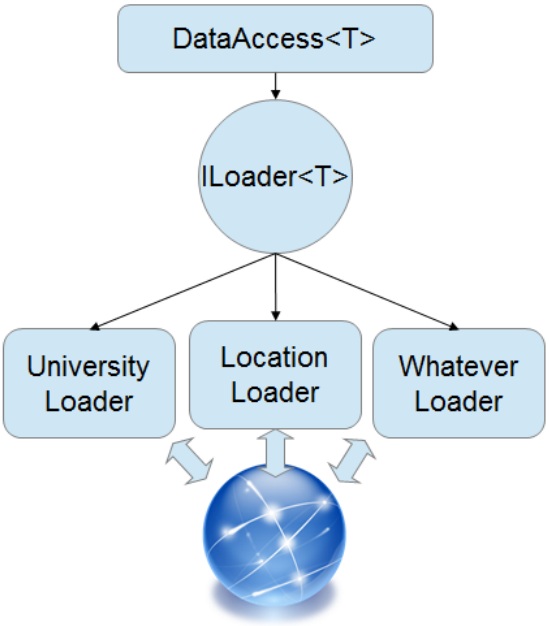

Figure 7: Loader Schema

The principle is simple: in order to query data the data access layer needs to be provided with a loader to fetch data into a local repository. To get only the data we need loaders have beed designed to have several methods but all are accessed through a common and generic ILoader interface. For this principle to work several factories are necessary which together with configuration classes allow the programmer to create a loader which will load only the data needed and only when it is needed.
The loaders themselves are nothing more than a collection of html parser methods (for scraping web pages) or a collection of API access methods (for open data sources which provide an API). The loader uses methods according to the configuration settings provided and loads the local repository with data which can then be queried by the data access layer.
Loaders can be easily added once additional data sources are available and as long as they implement the common ILoader interface they will integrate seamlessly into the application.

4.4 Domain Model

The Domain Model represents a basic information and functionality needed for the project. It usually contains business practices and operational details. In this project, the Domain Model classes are mostly information holders which contain collected data from the external data sources. Thus allow us to assign meaning to the collected data and classify them into appropriate groups This is very important in OR mapping because the classes of systems are mapped in database tables using nHibernate. Also, the Domain Model includes the basic tasks that must be provided by the application such as a university recommendation service. The Domain Model is a vital part of the application, so it is expected that there will be changes and upgrades in the future depending on the requirements and deadlines.
Therefore, it should be noted that the attached diagram is not the final version, but is a subject of changes in the future.

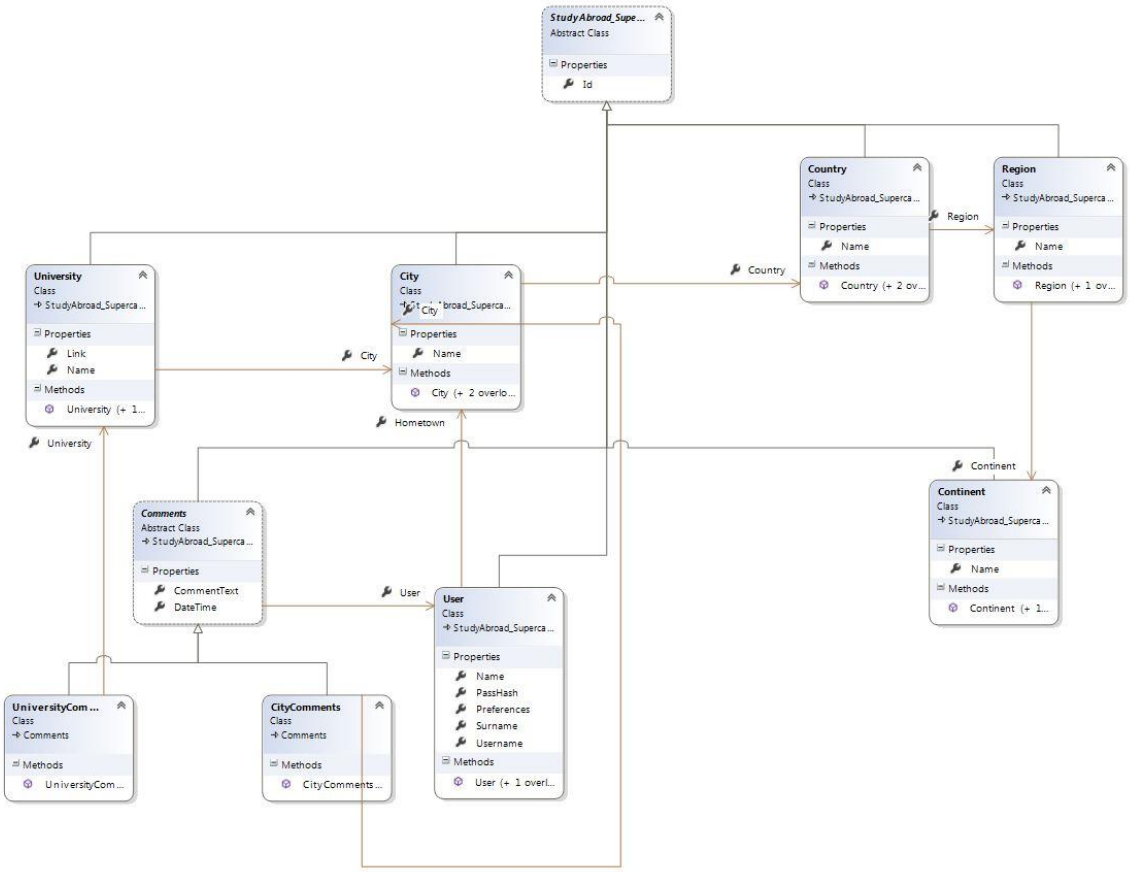

Figure 8: Domain Model class diagram

As you can see in the diagram, there is a super-class which is inherited by all other classes in the Domain Model. It is because that class contains Id data member which is important in OR mapping using nHibernate. In this way, all the classes inherit that data member. Also, due to the mapping, data members must be public and in each class must be the parameterless constructor.

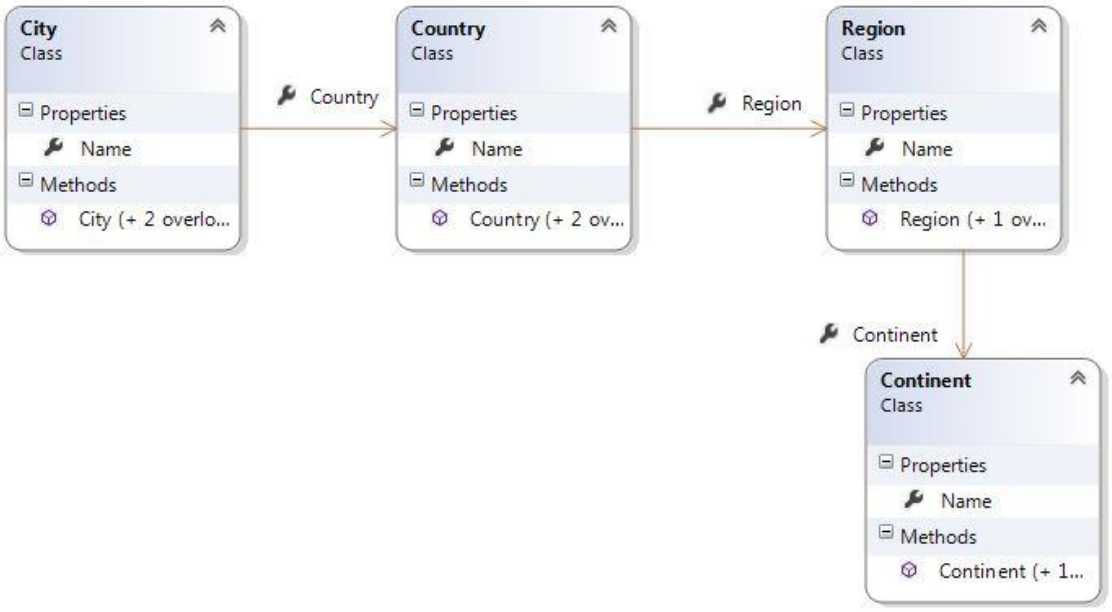

Figure 9: Geographical location classes

Regional division is presented with a series of classes that indicate the geographical location. Class City, State, Region and Continent are representing geographic concepts and are used to precisely locate the university. These classes are also informational holders for facts about the location and it is expected that they will be upgraded during development. Classes are connected with associations so every class knows to which region it belongs.

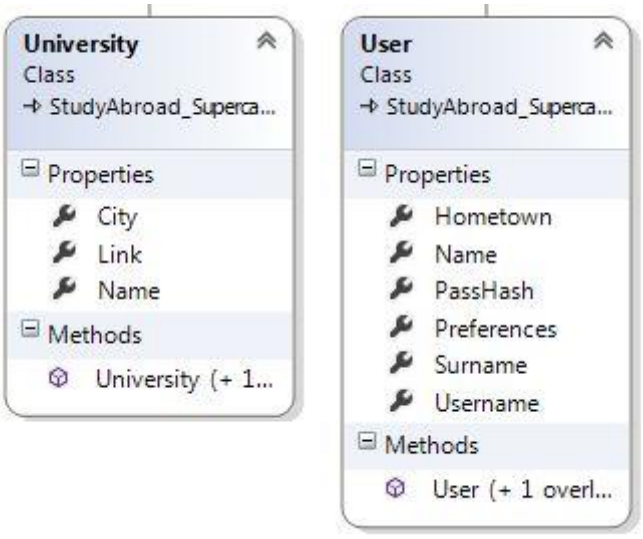

Figure 10: University and User classes

In the class User are stored user private data that will be used to create its profile and for his authentication. Class University stores information about every university. In the future, new data members will be added to this class if there will be more available information from data sources. Both classes have a connection to the class City. In the University class that connection represents location of university while in User class it represents place of living of that user.

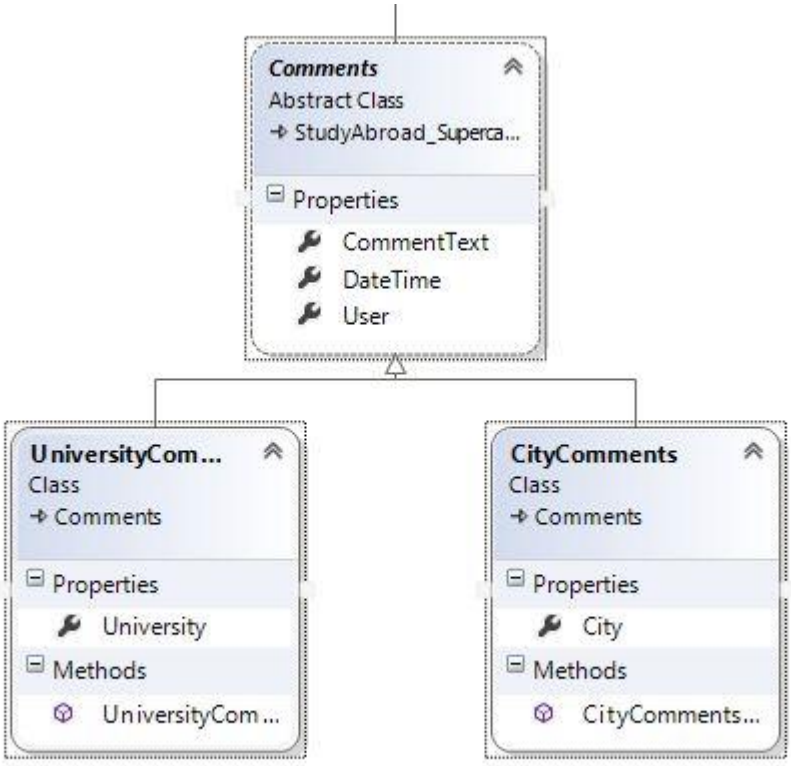

Figure 11: User comments

There are two types of comments in the system: comments of universities and comments of city locations. In this hierarchy, an abstract class Comment contains information that is contained in both types of the above mentioned classes. These are: the user who commented university / city, time of commenting and comment. The derived class contains information what is the subject of comment (university or city).

4.5 Database definition

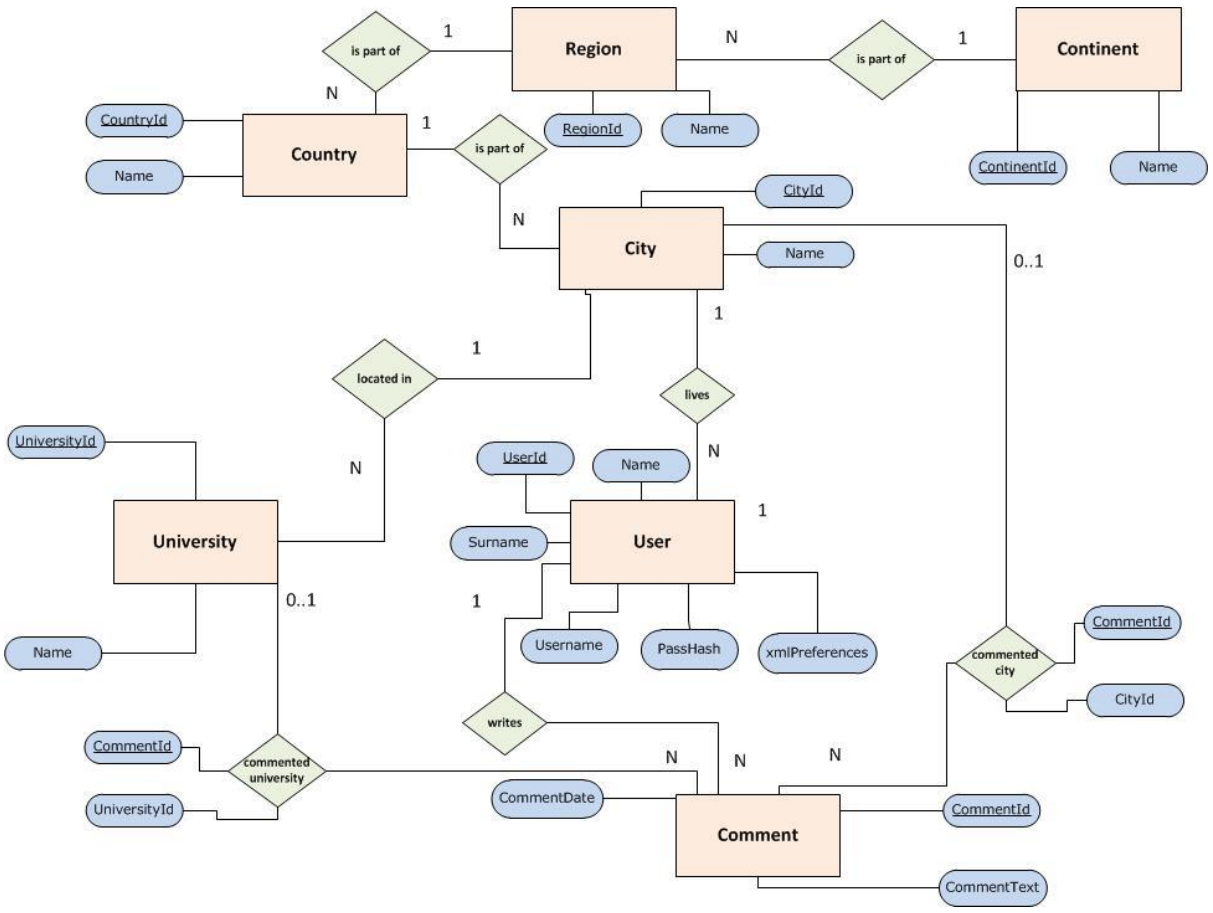

Figure 12: ER diagram

For the creation of the database there will be used OR mapping tool called NHibernate. It provides a framework for mapping an object-oriented domain model to a traditional relational database. Object-relational mapping (OR mapping) in computer software is a programming technique for converting data between incompatible type systems in object-oriented programming languages.

As you can see in the ER diagram, entities tables correspond to classes in the Domain Model of the system. In the Domain Model chapter states that data members must be public, and that each class must have a parameterless constructor. It is very important for the realization of OR mapping.

Hierarchical model of class is mapped in a special way. The details from abstract class are mapped in the one table and specials from derived classes are mapped in separate two tables.

Preferences that the user enters into the system will not be stored in the database but in XML filed. Attribute xmlPreferences in the table User represents that filed.

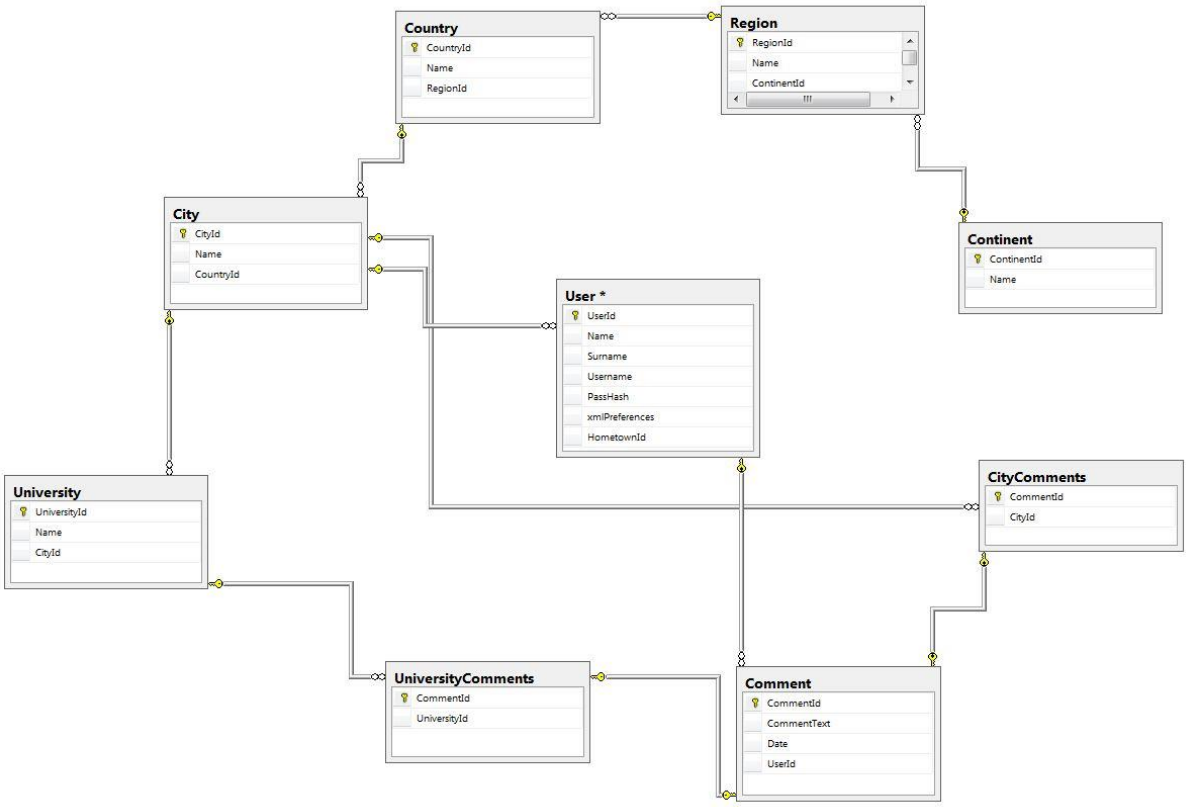

Figure 13: Database

In Figure 7. is presented implementation of ER model in concrete database. Details of the database are not frozen, because some changes are expected in the future. If there is a change in the Domain Model there will also be changes in the database.
Data Access layer (StudyAbroad.DataAccess described in chapter before) takes care of retrieving data in a transparent way. To higher layers it is irrelevant whether the data is obtained from the database or from external sources.