Assignment A1.2

Basics of Web Technologies

Student: Lorinczi Istvan

Group: 30444

# Requirements

Design, implement and test a distributed system, consisting of a Web Application (called Front-

End) for interacting with the user, a Database used by the Front End and another Web Application (called Service) that generates dynamic data based on HTTP Requests coming from the Front-End.

The following operating scenarios should be considered:

I. The Front-End starts by requesting the users to login. The user’s login data will be confronted with login credentials stored in the Database.

II. Upon successful login, the user will be redirected to another web page in the Front End, based on his role. There are the “administrator user” and “simple user” roles. The administrators should be able to perform CRUD (Create, Read, Update and Delete) operations on the collection of users in the database, through the Front End application. The simple users will be redirected to a welcome page which will display the user specific information held in the database for each user (include here at least the following: Name, Home Address including latitude and longitude, Birth Date). The welcome page should also display the Time zone of the user, based on the home address geographical coordinates. In order to obtain this information, the Front End application will make an HTTP Request to the Service application, including in the request the geographical location of the user. The Service application will reply with XML Document. (NOTE: The Service application may return random data instead of computing the actual Time zones)

# Tasks

# Design, implement and test the distributed system using Microsoft .NET technologies. (ASP.NET 2.0 technology is compulsory)

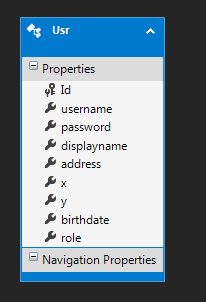
# Conceptual Architecture

Considering that there are more types of users with their individual data, a database should be used. Because of this there must be a layer that deals with the data, another layer that controls the flow of the application and one for displaying everything. This is the MVC architectural pattern.

The Model of the application consists only of the User class which shows the structure the user’s data is organized.

A User has:

* Name
* Password
* Address
* Longitude
* Latitude
* Role
* Date of Birth (year, month, day)

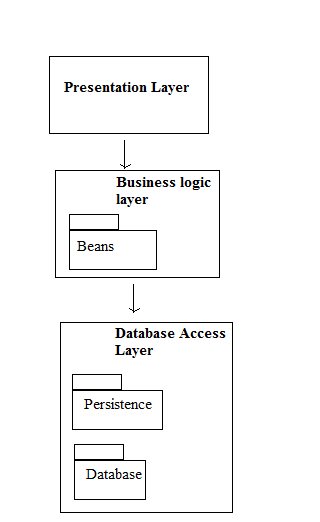


The login action depends on the role of the user. If the user has role “admin” a new page will appear with a list of users. The admin can check the details,add, edit, and delete users. If the role is “user” the new page will show information about the user and the timezone. The timezone is provided by a web service through url link. The timezone is calculated as longitude/latitude.

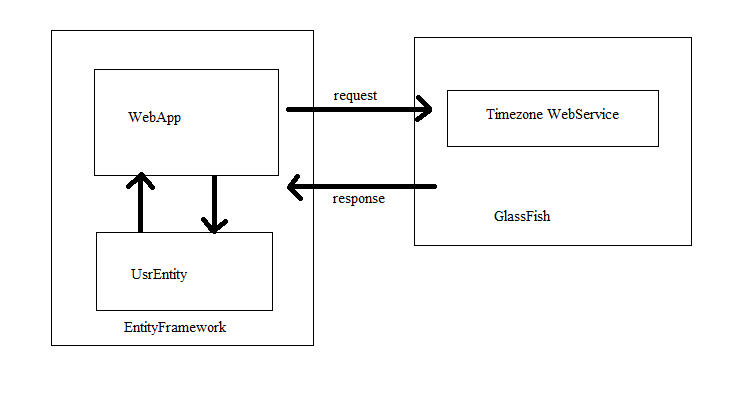
The Controller is divided between two separate entities: one is responsible for the login operation while the other is managing the database. The database is accessed through a layer that provides methods for CRUD operations.

The presentation layer consists of cshtml files that render the output for each page.

The conceptual diagram of the application:



Deployment diagram:



The UserApplication uses Asp.Net and EntityFramework. The DataBase is kep in memory.

The timezone application uses Tomcat and it is accessed only for requesting the timezone with an url :

<http://localhost:8080/timezone/Timezone?longitude=loggedInUser.getLongitude()&latitude=loggedInUser.getLatitude()>