TTSystem demo with a WebApp + GUI client + REST WCFService in a separate host

In this assignment related demo there is a web site, which is a client for a Trouble Ticket system, and also a GUI client to the same service. The web application allows an user to submit a problem as a trouble ticket, and also to consult previous submissions.

The web application uses a REST WCFService to do most of its operations (in this case the creation of a new trouble ticket and the retrieval of previous trouble tickets submitted by an user, as well as the list of users). For these simple operations the service simply uses a local database. The service is hosted separately from the web site application, and is available for the web application and other external applications.

The VS solution (TTs) of this demo has four projects. One of them (TTService) implements the REST WCFService as a library (producing a dll). Another is the host of the web service as a console application (TTServer), and the third is the WebSite (TTSite) with the web application (an ASP.NET form). The site has a reference to the service project (TTService) that puts automatically the service .dll in the Bin subdirectory. In order to invoke the service operations from the web application code we need also to add a proxy to the service, inside the TTSite . In order to invoke the service operations from the web application code we have written manually a simple proxy to the service, inside the TTSite code. Because REST services don't produce description files (WSDL) it is not possible to produce automatically the proxy.

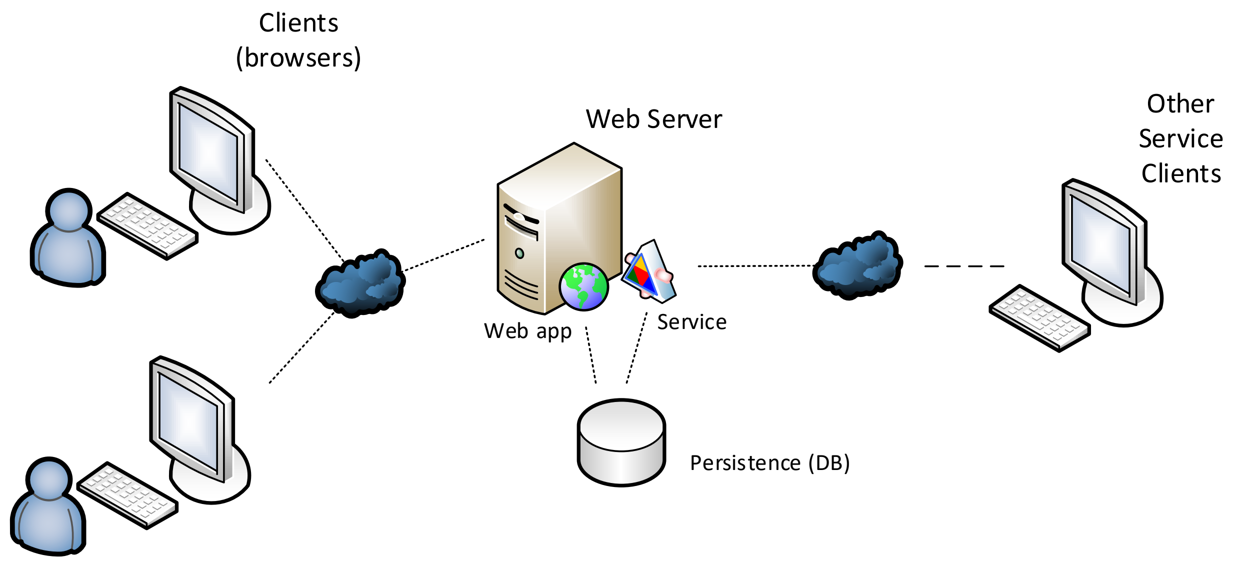
The TTService project was created from the template 'WCF Service Library' and the interface and implementation written according to the pretended mapping to HTTP requests.

The TTServer project is a simple .NET console application that uses the class WebServiceHost as a REST web server. The app.config file should be configured accordingly (using the webHttpBinding (REST) and webHttp behaviour). Also because of some security measures of Windows, to run a server on the HTTP protocol we must run it from VS or a console window in administrator mode (or register the application permantely as an administrator).

The TTSite is an ASP.NET web forms application. For creating it, start with an 'ASP.NET Empty Web Site' template and then add a new 'Web Form' item. A Web form has two files: one with the extension .aspx where you could write HTML and CSS and add visual interface controls in the file design window in VS from the toolbox (at left); there is also a code file (code behind) where you can write C# code mainly to be executed when there are 'postbacks' in consequence of user interactions with the page (events). The code file can contain initializations, handlers to events (e.g. button clicks), and other auxiliary code. It is executed on the server side and can modify the page interface is response to 'postbacks'.

In this demo there is a database as a SQL Server LocalDB in the TTServer project, but can be any other. This one is free and has full support from VS 2019. It only needs to be installed in the same system as VS for development and in the server system for deployment. This kind of DB Manager uses a file for each DB (in this case the TTs.mdf file), which can be put in any place in the file system, facilitating its deployment. But it is also a server system in the sense that the DB engine runs on a separated process. Nevertheless that process is started automatically when its client starts (or makes the first connection) as a client's child process. This can have some advantages in performance (and also some disadvantages) over a simple embedded DB. The LocalDB connection string can be specified in the project (TTServer) .config file.

The TTClient project is a simple GUI client to the service that uses a ListBox to display the user names and when a name is displayed it retrieves and shows the tickets into a DataGridView. It uses a proxy similar to the one used in the web application and for that it has to reference also the service project (TTService).


Clients (browsers)
Web Server
Other Service Clients
Web app
Service
Persistence (DB)