UDC 004.42

A. V. Ananko, A. A. Muharevskiy
Baranavichy State University, Baranavichy, the Republic of Belarus

FEATURES OF DEVELOPING SIMPLE CLIENT-SERVER APPLICATIONS BASED ON WCF AND REST ARCHITECTURE

Introduction. A plethora of network applications are used on mobile and desktop devices daily. Local area networks and the Internet allow you to use remote resources and perform operations on a centralized server for storing and processing data and control the process from any compatible workstation. This implementation involves creating client-server applications that send input data entered on the client side over the network to the server, which performs the processing and sends the results back to the client, who presents them through its GUI.

There are a considerable number of different architectures used for building network client-server interaction. However, this article will focus on REST and WCF. As an example of implementing software applications using these architectures, two application projects are presented: for calculating and optimizing profit and for performing CRUD operations over information cards.

Main part. The applications were written in C# using Microsoft Visual Studio 2022.

The first application is developed using .NET Framework 4.6. The WinForms API is used to create the client's GUI. Client-server communication relies on the WCF. Windows Communication Foundation is a platform for creating service-oriented applications. Using WCF, you can relay data as asynchronous messages from one service endpoint to another. Messages can be encrypted for data protection and require users to pass the authentication process before receiving messages. Data is transmitted using the TCP and/or HTTP(S) protocols in XML or binary format. Security standards like SSL and TLS can also be implemented [1]. At the stage of writing the source code, the platform allows you to simplify the organization of complex network interaction to a remote call of interfaces declared methods

The network interaction structure of the first application may seem quite simple, because the methods are called remotely. In practice, a two-way call requires the creation of five interfaces (they are necessary for initializing the ServiceHost<IServerOpmtimize> object on the server side and DuplexChannelFactory<IServiceOptimize> on the client side) and, accordingly, the classes that implement them. Interfaces, classes, and their components are described with a large number of attributes, writing them manually is time-consuming. Figure 1 shows a diagram of the server and client interfaces.

Figure 1 — Application's network interfaces diagram

The IServiceOptimize interface describes the client-server interaction. IServiceOptimizeCallback describes the client methods called by the server.

Of course, the development environment allows to generate these interfaces in the client code automatically by connecting to a compiled and running server using the HTTP protocol, however, further changes on the server side (and adding new methods) require re-creation of these interfaces, which makes it difficult to develop both sides at the same time. It should also be noted that a server built using WCF usually consists of at least two parts: a class library that

performs the necessary operations (in this case, working with a database, managing users, and calculating), and a host application in the form of, for example, a console application or a Windows service. In conclusion, a conceptually simple technology requires a lot of effort, especially if it is being developed for a small system.

The second application was developed using the REST API.

Representational State Transfer (REST) is a technical description or architectural style of how the World Wide Web works. A REST API is a type of web server that allows a user-managed or automated client to access resources that model system data and functions. From the client's point of view, the architecture is represented as a set of links accessible for receiving or sending data in JSON-format. Data is transmitted over the HTTP(S) protocols, which also work according to the SSL security standard. The concept of using REST suggests using GET (receiving data), POST (adding data; actual use varies), PUT (updating data), DELETE (deletion) methods [2].

The client side of the second application was created using WPF, built on the base of .NET Framework 4.7.1. The server side uses the features of ASP.NET Core 3.0. ASP.NET Core is a cross-platform, high-performance, open-source platform for creating modern cloud applications connected to the Internet. It allows creating the web applications and services, IoT applications and mobile backends as well as deployment in the cloud or locally [3].

According to the description, server part consists of a single project, which can be immediately built both as a console application and as a Windows service. In addition, it is possible to work with the server directly from the browser, by accessing necessary URL-addresses.

In practice, the simplicity of implementation is obvious. There is no need to develop interfaces. Each server method is marked only with an attribute indicating the address and one of GET, POST, PUT, DELETE methods. Figure 2 shows diagrams of classes responsible for network interaction.

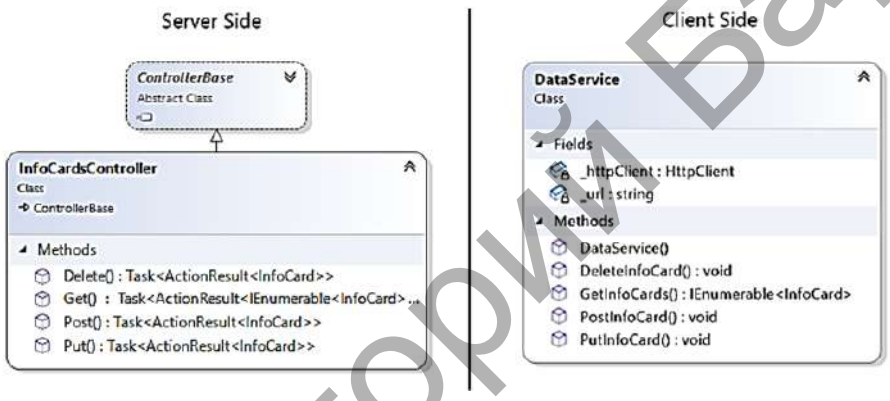

Figure 2 — Network interaction classes diagram

Despite the differences in the functionality of both applications, server implementation using REST is simpler. Classes like InfoCardsController on the server that are inherited from Microsoft.AspNetCore.Mvc.ControllerBase may be larger, but their composition will be similar. The development environment provides project templates based on the REST API, making the server part development even more simplified.

Using the JSON format may cause a file forwarding issue, since only text information is supported. In those cases, Base64String is used to represent bytes of forwarded files.

Conclusion. The applications allow to compare features and difficulties of implementing client-server applications based on the REST API and WCF. Using REST, despite using only four methods, provides ease of implementation and by default contains everything that is needed for deployment. Although WCF provides convenient functionality, it requires a lot more time to write source code, so it is not a good choice for small client-server applications.

References

1. What Is Windows Communication Foundation [Electronic resource]. — Mode of access: https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf . — Date of access: 28.02.2022.

2. *Masse, M.* REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces / M. Masse — Sebastopol : O'Reilly Media, 2011. — 116 p.

3. Introduction to ASP.NET Core [Electronic resource]. — Mode of access: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core . — Date of access: 01.03.2022.