**ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ**

**ФИЗИКО-ТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ**

**КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Инструкции и методические указания   
к выполнению лабораторных работ**

**Перспективные сетевые технологии (SOA)**

|  |  |
| --- | --- |
| **Направление подготовки:** | *09.04.01 - Информатика и вычислительная техника*. |
| **Программа подготовки:** | *Интеллектуальные информационные системы* |
| **Квалификация (степень) выпускника:** | *Магистр* |
| **Форма обучения:** | *Дневная* |

Донецк, 2021

Общие указания к отчёту

Отчёт по лабораторной работе должен содержать:

* текст задания с указаниями для выполнения,
* основные коды приложений,
* необходимые комментарии для понимания алгоритма работы информационной системы стронним человеком,
* перечень контрольных вопросов (без ответов).

При сдаче лабораторной работы необходимо продемонстрировать её работоспособность, Показать, как в кодах реализовыввались требования лабораторной работы.

Ответить на контрольные вопросы, которые задаст преподаватель.

лабораторные РАБОТы

1. **SOA с калькулятором**

***Задание***

Разработать Win-приложение с формой в виде некоторого произвольного калькулятора, вычислительной системы. Все вычислительные бизнес-процессы калькулятора вынести во внешний Web-сервис.

***Указания и рекомендации***

1. Разработать с помощью Visual Studio некоторый Win-интерфейс для каких-либо вычислений, преобразований. Выделить пареметры, которые должны передаваться в бизнес-логику вычислений и вызвращаться из неё;
2. Разработать в том же решении, что и проект Win-калькулятора, проект Web-сервиса для приёма параметоров от калькулятора. Реализовать в сервисе бизнес-логику вычислений и возврат результатов. Подключить Win-приложение к созданному локальному Web-сервису, протестировать работу калькулятора;
3. Настроить метод Web-сервиса:
   1. Установить буферизацию ответов Web-сервиса,
   2. Отключить кэширование,
   3. Отключить состояние сеансов;
4. Разместить Web-сервис на стороннем хостинге в Internet (возможно с помощью преподавателя). Подключить Win-приложение калькулятора к удалённому Web-сервису;
5. Продемонстрировать работу калькулятора.

***Контрольные вопросы***

* 1. Чем отличаются Web-сервисы от Web-сайтов?
  2. Каков язык передачи данных Web-сервисов и Web-сайтов?
  3. Назначение протоколов SOAP и WSDL?
  4. Каковы задачи посредников (прокси) в приложениях для общения с Web-сервисами?
  5. Что хорошего и плохого при включении сеанса Web-сервиса?
  6. В чём суть SOA?
  7. Каковы цели преследуются при построении SOA?
  8. Каковы уровни абстракции SOA в информационной системе предприятия?

1. **Сервис-ориентированное программирование:   
   gRPC-сервис**

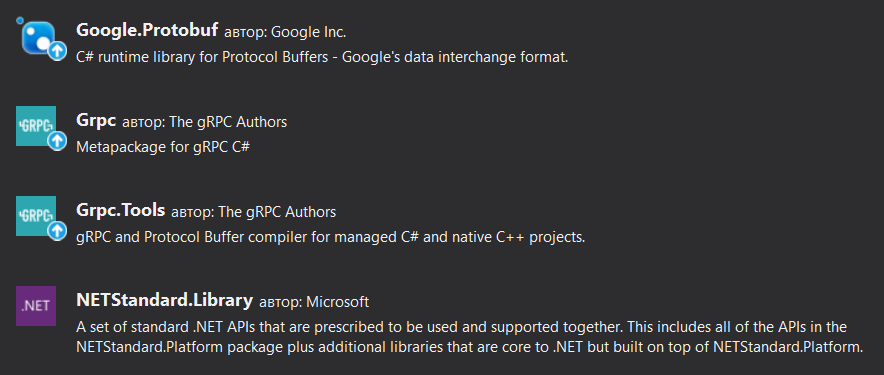
***Задание***

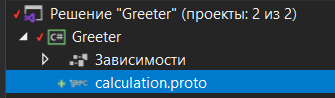
Разработать SOA-приложение, содержащее:

* Произвольное интерфейсное приложение (например, калькулятор, аналогичный предыдущей лабораторной работе), которое будет клиентом для gRPC-сервиса, реализоующего бизнес-логику вычисленний.
* gRPC-сервис, который должен содержать две операции (методы): 1 – авторизация сеанса с клиентом; 2 – реализация вычислений в авторизованном сеансе. Для реализации авторизации клиентов, на стороне сервиса необходимо иметь базу (файл) допустимых клиентов и их пароли.

***Указания и рекомендации[[1]](#footnote-1)***

1. Рассмотрим пример создания клиента сервиса **Calculation** с помощью Visual Studio .Net Core на C#. В соответствующем решении необходимо установить пакеты:



Внутри решения создаём первый проект для построения calculation.proto, например, —

В проект добавляем (правой кнопки мыши   
на проекте) файл calculation.proto:

syntax = "proto3";

package calculation;

service Calculation{

rpc Calculate (Expression) returns (Response) {}

}

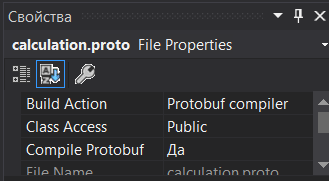
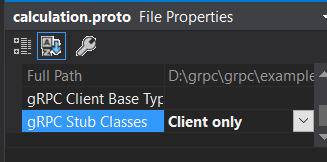
message Expression { string expression = 1; }

message Response {

float result = 1;

}

Необходимо указать, чтобы использовался Protobuf compiler и нам достаточно генерировать только клиент:

Запускаем построение второго проекта непосредственно для клиента сервиса. Чтобы клиент видел первый проект, необходимо добавить в него ссылку на проект с .proto-файлом. (правой кнопкой мыши по проекту – добавить ссылку). После этого в проекте клинта необходимо добавить using **Calculation**:

using PST\_Lab\_1.Interfaces;

using System.Threading.Tasks;

using PST\_Lab\_1.Conenctions;

using System.Net.Http;

using PST\_Lab\_1.Models;

using System.Net;

using System.IO;

using System.Text.Json;

using Calculation;

using Grpc.Core;

using System;

using System.Threading;

namespace PST\_Lab\_1.Services

{

public class CalculationService : ICalculation

{

public double GrpcCalculation(string sending\_expression)

{

// Подключение с сервису и вызов его методов...

}

}

}

Обращение к сервису **Calculation** и вызов методаCalculate может иметь вид:

Channel channel = new Channel("127.0.0.1:50051", ChannelCredentials.Insecure);   
// - создаем канал подключения к сервису. Первым параметром передаем IP и порт. Вторым – данные для авторизации (в данном случае их нет).

channel.ConnectAsync(deadline: DateTime.UtcNow.AddSeconds(2)); // - необязательный параметр для канала, задаёт время ожидания ответа, здесь 2 секунды.

var client = new Calculation.Calculation.CalculationClient(channel); // - создаем объект посредника для подключения клиента к сервису.

var reply = client.Calculate(new Expression { Expression\_ = “2+2” }); // - вызываем метод Calculate, передаём ему объект в виде нашего выражения и принимаем результат в объект reply. Когда мы прописываем параметр в прото файле, он создает на его основе объект. В данном случае expression.

channel.ShutdownAsync().Wait(); // Разрываем соединение, если необходимо.

Console.WriteLine(reply.Result); //Выводим результат из поля Result объета reply, в прмере – на консоль.

1. Теперь рассмотрим пример создания сервиса в том же решении, что и клиент. С помощью правой кнопки мыши на решении выбрать «Добавить» - «Создать проект». После чего в фильтрах выбрать C# и найти gRPC сервис, служба gRPC ASP.NET Core.

В папке Properties необходимо открыть файл launchSettings и настроить следующим образом:

{

"profiles": {

"TestServer": {

"commandName": "Project",

"dotnetRunMessages": "true",

"launchBrowser": false,

"applicationUrl":

"http://localhost:50051;https://localhost:5001",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

}

}

Теперь перейдём в папку Protos и заполним файл greet.proto нашим API:

syntax = "proto3";

package calculation;

service Calculation{

rpc Calculate (Expression) returns (Response) {}

}

message Expression {

string expression = 1;

}

message Response {

float result = 1;

}

Перейдем в папку Services и создадим там файл CalculationService.cs со следующим содержимым:

using Grpc.Core;

using Microsoft.Extensions.Logging;

using System.Threading.Tasks;

using Calculation;

namespace TestServer

{

public class CalculationService : Calculation.Calculation.CalculationBase

{

private readonly ILogger<CalculationService> \_logger;

public CalculationService(ILogger<CalculationService> logger)

{

\_logger = logger;

}

public override Task<Response> Calculate(Expression request, ServerCallContext context)

{

// здесь необходимо написать код расчёта полученного выражения

return Task.FromResult(new Response

{

Result = // а сюда записать ответ выражения

});

}

}

}

Осталась последняя настройка. В файле Startup.cs метод Configure должен выглядеть так:

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseRouting();

app.UseEndpoints(endpoints =>

{

endpoints.MapGrpcService<CalculationService>();

endpoints.MapGet("/", async context =>

{

await context.Response.WriteAsync("Communication with gRPC endpoints must be made through a gRPC client. To learn how to create a client, visit: https://go.microsoft.com/fwlink/?linkid=2086909");

});

});

}

Сервис готов, теперь его можно запускать.

Рассмотрим пример сервиса с обработкой ошибок на клиенте. Пусть сервис имеет слудующий метод **Calculate**:

public override Task<Response> **Calculate**(Expression request, ServerCallContext context)  
 {

throw new RpcException(new Status(StatusCode.PermissionDenied, "Permission denied")); // скажем клиенту, что у него нет доступа

return Task.FromResult(new Response

{

Result = 0;

});

}

Исключение наследуется от RpcException. Перехватим его на клиенте:

try

{

Channel channel = new Channel("127.0.0.1:50051", ChannelCredentials.Insecure);

channel.ConnectAsync(deadline: DateTime.UtcNow.AddSeconds(2));

var client = new Calculation.**Calculation**.CalculationClient(channel);

var reply = client.**Calculate**(new Expression { Expression\_ = “2+2” });

return reply.Result; // 4

}

catch (RpcException ex)

{

return ex.message; // выводим сообщение исключения из поля message

}

После возникновения исключения сервис прекращает выполнение операции. Если клиент успешно перехватил исключение, то канал остается открытым.

1. Продемонстрировать работу полученного SOA калькулятора.

***Контрольные вопросы***

* 1. В чём суть REST и RPC подходов?
  2. Когда следует применять REST или RPC в SOA?
  3. Чем отличаются gRPC-сервисы от Web-сервисов?
  4. Что такое микросервисы и где их целесообразно использовать?
  5. Особенности протокола HTTP/2?
  6. Что такое Protobuf, как и для чего он используется?
  7. Что такое «заглушки» в gRPC?
  8. Можно ли управлять длительностью сеанса связи с gRPC-сервисом?

1. **Сервис-ориентированное программирование:   
   WCF-сервис**

***Задание***

Разработать Web-калькулятор, реализующий бизнес-логику вычисленний в WCF-сервисе. Предусмотреть защиту сервиса от несанкционированных потребителей предлагаемой вычислительной услуги.

***Указания и рекомендации***

* С помощью Visual Studio с .NET ниже 5 (не Core) разработать Web-приложение калькулятора, аналогичное предыдущим лабораторным работам, которое будет клиентом для WCF-сервиса, реализоующего бизнес-логику вычисленний.
* WCF-сервис должен быть сеансовым и содержать две операции (методы). Первая операция – это авторизация сеанса с клиентом, вторая – реализация вычислений в авторизованном сеансе. Для реализации авторизации клиентов, на стороне сервиса необходимо иметь базу (файл) допустимых клиентов и их пароли.

Логика сервиса может быть следующей:

//Глобальные переменные сеанса:

public static bool AuthenticatedSession;

public static string user;

//Операция авторизации сеанса:

[OperationBehavior]

public void **Auth**(параметры от клиента: login, PWD)

{

//Аутентификация пользователя **user** в методе bool ValidateUser():

if ( ValidateUser(login, PWD) )

{

AuthenticatedSession = true

}

else

{

//Генерировать исключекние => разорвать сеанс и закрыть канал связи:

Exception UserNotValidated = new Exception("Пользователь не авторизован");

throw UserNotValidated;

}

//Операция вычислений в авторизованном сеансе:

[OperationBehavior]

public ... **Calculate**(параметры вычислений от/для клиента)

{

//Авторитзация пользователя

if ( AuthenticatedSession )

{

//Реализация бизнес-логики вычислений для пользователя **user**

}

else

{

//Генерировать исключекние => разорвать сеанс и закрыть канал связи:

Exception UnAuthorizedAccess = new Exception("Несанкционированный доступ");

throw UnAuthorizedAccess;

}

}

* Для получения клиентом созданных исключений UserNotValidated и UnAuthorizedAccess для дальнейшей их обработки на клиенте:
  1. Используйте указания на тип стандартных исключений или сбоев FaultException в контракте операций WCF-сервиса, например:

[FaultContract(typeof(FaultException))]

* 1. В конфигурационных файлах клиента и сервиса, в привязке конечных рабочих точек, через которые осуществляется подключение, разрешите передачу исключений:

<serviceDebug includeExceptionDetailInFaults="true"/>

Перехват и обработка исключений сервиса может быть организована на стороне клиента в виде:

**try** { }

// Перехватить неизвестное исключение типа FaultException:

**catch** (FaultException faultEx) // поместить выловленное исключение в faultEx

{

// Обработка исключения сервиса...

// Доступ к сообщению в исключении – это faultEx.Message,

// к деталям - faultEx.StackTrace

wcfClient.Abort(); // немедленно удалить прокси-посредника wcfClient

}

* Продемонстрировать работу полученного SOA калькулятора.

***Контрольные вопросы***

* 1. Что такое сервис-ориентированное программирование?
  2. Чем отличаются WCF-сервисы от Web-сервисов?
  3. Что такое и зачем нужны конечные точки WCF-сервисов?
  4. Особенности привязок wsHttpBinding, wsDualHttpBinding, NetTcpBinding, NetTcpBinding?
  5. Какие разновидности контрактов существуют в WCF-сервисах?
  6. Зачем нужны метаданные в WCF-сервисах и как их можно предоставлять (публиковать)?
  7. Особенности WCF-сервисов уровня вызова, сеансовых, синглетных?
  8. Какова концепция обработки ошибок в WCF?
  9. Что такое исключения класса FaultException?
  10. Как ведут себя WCF-сервисы уровня вызова, сеансовые, синглетные после завершения транзакции или при возникновении исключения?
  11. Нарисуйте и объясните график оценки эффективности сервиса с течением времени.
  12. Объясните смысл графика «длинного хвоста».
  13. Что такое SaaS, PaaS, IaaS?

1. **WCF-агент с транзакциями и обработкой ошибок**(Рекомендуется для повышения баллов)

***Задание***

Модернизировать предыдущую работу. Добавить в WCF-сервис ведение журнала в некоторой СУБД, поддерживающей транзакции, с информацией об авторизованном пользователе и дату его доступа к сервису. При этом, необходимо обеспечить транзакционность всей системы и контроль возможных исключений.

***Указания и рекомендации***

1. В приложении-клинете, перед вызовом второй операции (вычисления калькулятора) необходимо инициализировать корневую транзакцию, а после получения положительного результата от соответсвующей операции сервиса, закрепить транзакцию:

//Инициировать корневую транзакцию «клиент-сервис»

using (TransactionScope scope = new TransactionScope())

{

//Вызвать операцию сервиса в созданной транзакции

//...

scope.Complete(); //если исключений нет - транзакция закрепляется

}

1. В поведении транзакционной операции сервиса необходимо потребовать поддержку транзакций:

[OperationBehavior(TransactionScopeRequired = true)]

В контракте такой операции необходимо указать возможность распространения транзакций на данную операцию:

[TransactionFlow(TransactionFlowOption.Allowed)]

Поскольку WCF-сервис может поддерживать сеансы, то в его поведении, для возможности многократного вызова вычислений в одном авторизованном сеансе, необходимо указать возможность продолжения сессий после заверщения транзакции:

[ServiceBehavior(ReleaseServiceInstanceOnTransactionComplete=false)]

Проверьте конфигурационный файл клиента и сервиса. Там в привязке конечной рабочей точки, через которую вы осуществляете транзакционное подключение, доложно быть указано разрешение транзакций:

<binding name ="..." transactionFlow = "true" />

Также не забывайте включать надёжное соединение:

<reliableSession enabled="true" />

1. Для получения клиентом от WCF-сервиса исключений и их дальнейшей обработки:
   1. Используйте указания на тип стандартных исключений или сбоев FaultException в контракте операций WCF-сервиса, например:

[FaultContract(typeof(FaultException))]

* 1. В конфигурационных файлах клиента и сервиса, в привязке конечных рабочих точек, через которые осуществляется подключение, разрешите передачу исключений:

<serviceDebug includeExceptionDetailInFaults="true"/>

В предыдущей лабораторной работе в WCF-сервисе были использованы два нестандартных исключения – сбои типа FaultException: UserNotValidated и UnAuthorizedAccess с необходимыми сообщениями. Их можно передать клиенту.

Перехват и обработка исключений (сбоев) севриса на стороне клиента может быть организована как:

**try** { }

// Перехватить неизвестное исключение типа FaultException:

**catch** (FaultException faultEx) // поместить выловленное исключение в faultEx

{

// Обработка исключения сервиса...

// Доступ к сообщению в исключении – это faultEx.Message,

// к деталям - faultEx.StackTrace

wcfClient.Abort(); // удалить прокси-посредника wcfClient

}

***Контрольные вопросы***

* 1. Что такое сервис-ориентированное программирование?
  2. Что такое «транзакция»? Изобразите графически.
  3. Понятие распределённой транзакции в SOA, голосование, менеджер транзакции. Изобразите графически.
  4. Понятие о разрешении, распространении и получении транзакций в SOA c WCF.
  5. Какова концепция обработки ошибок в WCF?
  6. Что такое исключения класса FaultException?
  7. Как ведут себя WCF-сервисы уровня вызова, сеансовые, синглетные после завершения транзакции или при возникновении исключения?
  8. Нарисуйте и объясните график оценки эффективности сервиса с течением времени.
  9. Объясните смысл графика «длинного хвоста».
  10. Что такое SaaS, PaaS, IaaS?
  11. Место SaaS на рынке программного обеспечения?

1. Подготовлено магистрантом [Алексеем Винником](https://vk.com/alex_vinniko) [↑](#footnote-ref-1)