НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
Кафедра систем автоматизованого проектування

Звіт

З дисципліни

«Екстремальне програмування»

Лабораторної робота №1

«C#. Основи WCF. Серверна частина»

Варіант №14

Виконав:

Студент гр. КН-311

Куриляк Ю.А.

Прийняв:

Щербак С.С.

Львів – 2020

Теоретичні відомості:

Основи WCF

WCF - це середовище виконання і набір інтерфейсів API для створення систем, які відправляють повідомлення між службами та клієнтами. Ті ж інфраструктура і інтерфейси API використовуються для створення додатків, які обмінюються даними з іншими додатками на комп'ютері або на комп'ютері, який знаходиться в іншій компанії, і доступ до якого можна отримати через Інтернет.

Обмін повідомленнями та кінцеві точки

WCF заснована на концепції зв'язку на основі повідомлень, і все, що може бути змодельоване як повідомлення (наприклад, HTTP-запит або повідомлення MSMQ), можуть бути представлені одноманітно в моделі програмування. Це забезпечує універсальний інтерфейс API для різних транспортних механізмів. Модель розрізняє *Клієнти* , які є додатками, які ініціюють зв'язок, і *службами* , які представляють собою додатки, які очікують від клієнтів взаємодіяти з ними і реагують на це взаємодія. Один додаток може бути як клієнтом, так і службою. Приклади см. В розділі Дуплексні служби і одноранговi мережі .

Між кінцевими точками виконується обмін повідомленнями. *Кінцеві точки* - це місця, куди відправляються або приймаються повідомлення (або і ті, і інші), і визначаються всі відомості, необхідні для обміну повідомленнями. Служба надає одну або кілька кінцевих точок прикладання (а також нуль або більше кінцевих точок інфраструктури), а клієнт створює кінцеву точку, сумісну з однією з кінцевих точок служби.

Кінцева точка описує стандартний спосіб відправки повідомлень, спосіб їх відправки і то, як повинні виглядати повідомлення. Служба може надавати ці відомості у вигляді метаданих, які клієнти можуть обробляти для створення відповідних клієнтів WCF і _стеков_связі.

Протоколи зв'язку

Одним з обов'язкових елементів стека зв'язку є *транспортний протокол* . Повідомлення можна відправляти через інтрамережі або через Інтернет за допомогою загальних транспортів, таких як HTTP і TCP. Передбачені інші транспорти, що підтримують зв'язок з додатками черги повідомлень і вузлами в сітці тимчасової мережі. Додаткові механізми транспорту можна додати за допомогою вбудованих точок розширення WCF.

Іншим обов'язковим елементом стека зв'язку є кодування, що визначає спосіб форматування будь-якого заданого повідомлення. WCF надає наступні кодування:

- кодування тексту - кодування з можливістю взаємодії;
- кодування підсистеми оптимізації передачі повідомлень MTOM - підтримує взаємодію спосіб ефективної відправки неструктурованих двійкових даних в службу і з неї;
- двійкове кодування для ефективної передачі.

Додаткові механізми кодування (наприклад, кодування стиснення) можна додати за допомогою вбудованих точок розширення WCF.

Шаблони повідомлень

WCF підтримує кілька шаблонів обміну повідомленнями, включаючи запит-відповідь, односторонній і дуплексний зв'язок. Різні транспорти підтримують різні шаблони обміну повідомленнями та таким чином впливають на типи підтримуваних взаємодій. Інтерфейси API і Виконавча WCF також дозволяють безпечно і надійно відсилати повідомлення.

Виконання роботи:

Інтерфейс взаємодії з БД I_InteractionDB

```
      using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace wcf_chat
{
    public interface I_InteractionDB
    {
        bool AddToDB(string table, string values);

        int GetIDByUsername(string username);

        ServerUser CheckAutorization(string login, string password);

        List<string> GetUserContactListUsernames(int id);

        List<Project> GetAllProjects();

        List<Task> GetTasksInProject(string projectId);

        void closeDB();
    }
}
```

Клас взаємодії з БД InteractionDB

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MySql.Data.MySqlClient;

namespace wcf_chat
{
    public class InteractionDB : I_InteractionDB
    {
```

```
        private string connectionStr =
"server=localhost;user=root;database=PM_DB;password=root";
        //string connectionStr = "server = localhost;user = root; database =
test_pm_database; password = root";
        private MySqlConnection connection = null;
        private MySqlCommand command;
        private string commandStr = "";
        private bool openingConnection = false;

        public InteractionDB()
        {
            Connect();
        }

        public InteractionDB(string connectionStr)
        {
            this.connectionStr = connectionStr;
            Connect();
        }

        public InteractionDB(MySqlConnection connection)
        {
            this.connection = connection;
            openingConnection = true;
        }

        private void Connect()
        {
            connection = new MySqlConnection(connectionStr);
            connection.Close();
        }

        public InteractionDB(I_InteractionDB idb)
        {
            //connection = ((InteractionDB) idb).connection;
            //connectionStr = ((InteractionDB)idb).connectionStr;
        }

        public bool AddToDB (string table, string values)
        {
            if (!openingConnection)
            {
                connection.Open();
            }
            string toTable = "";
            switch (table)
            {
                case "users" :
                    toTable = "users (login, username, pass)";
                    break;

                case "messages" :
                    toTable = "messages (user_one, user_two, message)";
                    break;

                case "projects":
                    toTable = "projects (p_name, manager, methodology)";
                    break;
```

```
                case "tasks":
                    toTable = "tasks (project, t_name, developer, complexity, deadline)";
                    break;
            }
            commandStr = "INSERT INTO " + toTable + " values(" + values + ")";
            command = new MySqlCommand(commandStr, connection);
            Console.WriteLine(commandStr);
            bool check_add = command.ExecuteNonQuery() == 1;
            if (openingConnection)
            {
                connection.Close();
            }
            return check_add;
        }

        public int GetIDByUsername(string username)
        {
            if (!openingConnection)
            {
                connection.Open();
            }
            openingConnection = true;
            commandStr = ("Select id from users where login = '" + username + "'");
            Console.WriteLine(commandStr);
            command = new MySqlCommand(commandStr, connection);
            MySqlDataReader reader = command.ExecuteReader();
            int id = -1;
            if (reader.Read())
            {
                id =  Convert.ToInt32(reader[0].ToString());

            }
            connection.Close();
            openingConnection = false;
            return id;
        }

        public ServerUser CheckAutorization(string login, string password)
        {
            ServerUser user = null;
            connection.Open();
            openingConnection = true;
            commandStr = ("Select * from users where login = '" + login + "' and pass = '" +
password + "'");
            Console.WriteLine(commandStr);
            command = new MySqlCommand(commandStr, connection);
            MySqlDataReader reader = command.ExecuteReader();
            bool read = reader.Read();
            if (read)
            {
                user = new ServerUser();
                Console.WriteLine(reader[0].ToString() + " " + reader[1].ToString() + " " +
reader[2].ToString() + " " + reader[3].ToString());
                Console.WriteLine(read);
                user.ID = Convert.ToInt32(reader[0].ToString());
                user.Name = reader[2].ToString();
            }
```

```
            connection.Close();
            openingConnection = false;
            return user;
        }

        public List<string> GetUserContactListUsernames (int id)
        {
            List<string> list = null;
            return list;
        }

        public List<Project> GetAllProjects()
        {
            connection.Open();
            openingConnection = true;
            commandStr = ("select project_id, p_name, manager, methodology, deadline,
p_status, creation_date from projects");
            Console.WriteLine(commandStr);
            command = new MySqlCommand(commandStr, connection);
            MySqlDataReader reader = command.ExecuteReader();
            List<Project> projects = new List<Project>();
            while (reader.Read())
            {
                Project project = new Project();
                project.id = Convert.ToInt32(reader[0].ToString());
                Console.WriteLine(reader[0].ToString());
                project.name = reader[1].ToString();
                Console.WriteLine(reader[1].ToString());
                project.manager = reader[2].ToString();
                Console.WriteLine(reader[2].ToString());
                project.methodology = reader[3].ToString();
                Console.WriteLine(reader[3].ToString());
                project.deadline = reader[4].ToString();
                project.status = reader[5].ToString();
                project.creation_date = reader[6].ToString();
                projects.Add(project);
                Console.WriteLine(project.id + " " + project.name + " " + project.manager
                    + " " + project.methodology + " " + project.deadline + " " +
project.status + " " + project.creation_date);
            }
            connection.Close();
            openingConnection = false;
            return projects;
        }

        public List<Task> GetTasksInProject(string projectId)
        {
            connection.Open();
            openingConnection = true;
            commandStr = ("select task_id, t_name, p_name, developer, complexity,
tasks.deadline, tasks.p_status, tasks.creation_date " +
                "from tasks join projects on project = project_id where project_id = " +
projectId + ";");
            Console.WriteLine(commandStr);
            command = new MySqlCommand(commandStr, connection);
            MySqlDataReader reader = command.ExecuteReader();
            List<Task> tasks = new List<Task>();
            while (reader.Read())
```

```
            {
                Task task = new Task();
                task.id = Convert.ToInt32(reader[0].ToString());
                task.name = reader[1].ToString();
                task.project = reader[2].ToString();
                task.developer = reader[3].ToString();
                task.complexity = Convert.ToInt32(reader[4].ToString());
                task.deadline = reader[5].ToString();
                task.status = reader[6].ToString();
                task.creation_date = reader[7].ToString();
                tasks.Add(task);
            }
            connection.Close();
            openingConnection = false;
            return tasks;
        }

        public void closeDB()
        {
            connection.Close();
        }

        ~InteractionDB()
        {
            //connection.Close();
        }
    }

}
```

Інтерфейс сервісу IServiceChat

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace wcf_chat
{
    // ПРИМЕЧАНИЕ. Команду "Переименовать" в меню "Рефакторинг" можно использовать для одновременного изменения имени интерфейса "IServiceChat" в коде и файле конфигурации.
    [ServiceContract(CallbackContract = typeof(IServerChatCallback))]
    public interface IServiceChat
    {

        [OperationContract]
        int Connect(string name);

        [OperationContract]
        void Disconnect(int id);

        [OperationContract]
        bool UserRegistration(string username, string login, string password);
```

```
        [OperationContract]
        int UserAuthorization(string login, string password);

        [OperationContract(IsOneWay = true)]
        void SendMsg(string msg, int id, string username);

        [OperationContract]
        List<string> UserContactListUsernames(int id);

        [OperationContract]
        List<Project> _GetAllProjects();

        [OperationContract]
        List<Task> GetTasksInProject(string projectId);

        [OperationContract]
        bool AddProject(string values);

        [OperationContract]
        bool AddTask(string task);

    }

    public interface IServerChatCallback
    {
        [OperationContract(IsOneWay = true)]
        void MsgCallback(string msg);
    }

}
```

Клас сервісу ServiceChat

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Remoting.Messaging;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Syndication;
using System.Text;
using System.Xml.Linq;

namespace wcf_chat
{

    [ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]
    public class ServiceChat : IServiceChat
    {

        List<ServerUser> users = new List<ServerUser>();
        int nextId = 1;
        I_InteractionDB database;

        public ServiceChat()
        {
            database = new InteractionDB();
        }
```

```
        public ServiceChat(I_InteractionDB idb)
        {
            database = idb;
            //InteractionDB a = new InteractionDB(idb);
        }

        public int Connect(string name)
        {

            ServerUser user = new ServerUser() {
                ID = nextId,
                Name = name,
                operationContext = OperationContext.Current
            };
            nextId++;
            foreach (var item in users)
            {
                Console.WriteLine(item);
            }
            //SendMsg(": "+user.Name+" п о д к л ю ч и л с я  к  ч а т у !",0);
            users.Add(user);
            return user.ID;
        }

        public void Disconnect(int id)
        {
            var user = users.FirstOrDefault(i => i.ID == id);
            if (user!=null)
            {
                users.Remove(user);
                //SendMsg(": "+user.Name + " п о к и н у л  ч а т !",0);
            }
        }

        public void SendMsg(string msg, int id, string username)
        {
            var user = users.FirstOrDefault(i => i.ID == id);
            string answer = user.Name + ";" + DateTime.Now.ToShortTimeString() + ", "
+ user.Name + " : " + msg;
            user = users.FirstOrDefault(i => i.Name == username);
            if (user != null)
            {

user.operationContext.GetCallbackChannel<IServerChatCallback>().MsgCallback(answer);
            }
            SaveMessageToDB(msg, id, username);
            // п е р е в і р и т и  н а  і с н у в а н н я  ю з е р а

            // з а п и с а т и  п о в і д о м л е н н я  у  б д
        }

        private void SaveMessageToDB(string msg, int id, string username)
        {
            int idRecipient = database.GetIDByUsername(username);
            database.AddToDB("messages", "'" + id + "', '" + idRecipient + "', '" +
msg + "'");
        }
```

```
        public int UserAuthorization(string login, string password)
        {
            ServerUser connectedUser = database.CheckAutorization(login, password);
            if (connectedUser != null)
            {
                ServerUser item = new ServerUser()
                {
                    ID = connectedUser.ID,
                    Name = connectedUser.Name,
                    operationContext = OperationContext.Current
                };
                users.Add(item);

                return connectedUser.ID;
            }
            return -1;
        }

        public bool UserRegistration(string username, string login, string password)
        {
            string table = "users";
            string values = "'" + login + "', '" + username + "', '" + password + "'";
            return database.AddToDB(table, values);
        }

        public List<string> UserContactListUsernames(int id)
        {
            return database.GetUserContactListUsernames(id);
        }

        public List<Project> _GetAllProjects()
        {
            List<Project> p = database.GetAllProjects();
            return p;
        }

        public List<Task> GetTasksInProject(string projectId)
        {
            return database.GetTasksInProject(projectId);
        }

        public bool AddProject(string values)
        {
            string table = "projects";
            //string values = "'" + project.name + "', '" + project.manager + "', '" +
project.methodology + "', '" + project.deadline + "'";
            return database.AddToDB(table, values);
        }

        public bool AddTask(string values)
        {
            string table = "tasks";
            //string values = "'" + task.project + "', '" + task.name + "', '" +
task.developer + "', '" + task.complexity + "', '" + task.deadline + "'";
            return database.AddToDB(table, values);
        }
    }
```

```
}
```