|  |
| --- |
| Semestrální práce KIV-UPS |
| 7. Chatovací systém |
| Klient-Server aplikace |

|  |
| --- |
| **Jméno**: Pavel Lorenz  **Email**: [pavel.lorenz@students.zcu.cz](mailto:pavel.lorenz@students.zcu.cz)  **Univerzitní** **číslo**: [A10B0061K](https://portal.zcu.cz/wps/myportal/!ut/p/c5/jZBBT4NAFIR_kdnZxy4Lx3UXFtotaynF2kvDwRgS23ow_n7BBk2IEmeOX2be5LEjG3zpPvqX7r2_XrpXdmDH-ASHuNaRaS1ivjI2lfe4aeBP8ck4XQjlAadIgiq5o41ugTL6T3rOxQ9fTj-Oa5evfzX8IY0bV2HrogdLQFtkIKWlV7XjMNGc-yoBWe7r0DTk8ql_tjDU4NP-ka99lgSTCMDuU5QIK7OxOUdQs_4gMgLlAlW6C0N8zl1JCiSpqbZry7_7l_73G5dOTLwqrudn9nbeH9CXo-8-AQdEsQs!/dl3/d3/L2dJQSEvUUt3QS9ZQnZ3LzZfS0xFOE9DODQwMERVOTBJME9KQ01ERjEwTzI!/#_nogo) |

Obsah

[7. Chatovací systém 1](#_Toc328583807)

[Zadání 1](#_Toc328583808)

[Programátorská dokumentace 1](#_Toc328583809)

[Technologie 1](#_Toc328583810)

[Knihovna (ChatLibrary) 1](#_Toc328583811)

[ChatKlient 4](#_Toc328583812)

[ChatServer 4](#_Toc328583813)

[Uživatelská dokumentace 4](#_Toc328583814)

[ChatKlient 4](#_Toc328583815)

[ChatServer 4](#_Toc328583816)

[Odkazy na použité zdroje 4](#_Toc328583817)

[Závěr 4](#_Toc328583818)

# 7. Chatovací systém

## Zadání

Chatovaní systém. Realizujte programy serveru a klienta pro chatování. Chatovaní server a klient bude podporovat přihlášení uživatele pod přezdívkou, komunikaci s ostatními uživateli, pouze s jedním definovaným uživatelem a odhlášení uživatele.

Protokol bude obsahovat příkazy LOGIN, LOGOUT, ALL\_MSG, PRIV\_MSG, USERS, PING a odpovědi OK a ERR.

## Programátorská dokumentace

### Technologie

Chatovací systémy lze realizovat pravděpodobné v jakémkoliv vyspělejším jazyku (např. Ansi C, C++, Java, C#, aj.). Protože se v práci nejvíce soustředím na technologii .net, tak jsem si po dohodě, zvolil programovací jazyk C#.

Mým cílem bylo, vytvořit jednoduše ovládatelnou základní knihovnu pro práci se sockety a jako nadstavbu nad touto knihovnou postavit klienta a server.

Server jsem realizoval jako konzolovou aplikaci, která je velice jednoduchá a hlavní podíl práce bude v knihovně.

Klienta jsem realizoval pomocí technologie WPF jako okenní aplikaci, aby práce s ním byla více uživatelsky přívětivá. Důraz jsem kladl na přehlednost a robustnost.

### Knihovna (ChatLibrary)

#### Požadavky

* Zapouzdřenost
* Jednoduchost
* Vstupy ovládané pomocí metod
* Výstupy realizované pomocí událostí
* Základní komunikace, pouze odesílání a přijímání textových zpráv

#### Realizace

Knihovna „ChatLibrary“ je oddělena od klienta i serveru. Je to samotná dll knihovna. Pro přehlednost jsem ji rozdělil pomocí namespaců je do jednotlivých bloků.

* **Arguments** (argumenty událostí, které mohou vyvstat)
* **Converters** (pomocné třídy pro převod text. řetězce na pole bytů a zpět)
* **Messages** (objekty předávané chatem)
* **ReadingLoop** (smyčka, která vyčítá zprávy na socketu)
* **UserManagements** (třídy reprezentující uživatele chatu)

V samotném kořenovém namespacu se nacházejí klíčové třídy:

* **ChatSocket** (abstraktní třída implementující společné prvky serveru i klienta)
* **SimpleChatClient** (jednoduchá implementace klienta)
* **SimpleChatServer** (jednoduchá implementace serveru)

Vše jsem se snažil programovat přes rozhraní, takže není nutné používat mojí implementaci. Kdokoliv si může vytvořit vlastní implementaci, např. vyčítací smyčky a použít ji.

Pro síťovou komunikaci bude ostatním programátorům plně postačovat připojit reference na ChatLibrary a vytvořit instance SimpleChatClienta/SimpleChatServeru.

#### SimpleChatClient

Tato třída implementuje přijímání a odesílání textových zpráv.

##### Konstruktory

##### SimpleChatClient(IPAddress ip, int port)

Základní konstruktor třídy. Na předané ip adrese a portu bude klient očekávat server.

SimpleChatClient(IPAddress ip, int port, IByteToStringConverter converter, IReadingLoopCreator creator)

Nejrozšířenější konstruktor, který vyžaduje instanci koncertru a tovární třídy na „vyčítací smyčky“.

##### Metody

virtual void Connect()

Metoda, která je v základu virtuální, abychom ji v případných potomcích mohli překrýt, a jinak se postará o spojení klienta se serverem.

virtual void SendMessage(string message)

Metoda je opět virtuální a stará se o posílání textových zpráv na server.

virtual void Close()

Poděděná metoda, která ukončuje klienta.

##### Události

event SocketHandler ServerSocketConnected

Událost, která je odpálená po připojení na server. V argumentu má socket klienta.

event SimpleMessageHander MessageReceived

Poděděná událost, která je odpálená při přijetí zprávy. Zpráva je předána v argumentu.

event SocketExceptionHandler ReadingInterrupted

Poděděná událost, která je odpálená při chybě objevené při čtení zprávy. Argument obsahuje výjimku a socket, ze kterého bylo čteno.

#### SimpleChatServer

Tato třída implementuje přijímání klientů a odesílaní zpráv na jednotlivé klienty.

##### Konstruktory

SimpleChatServer(IPAddress ip, int port)

Základní konstruktor třídy. Na předané ip adrese a portu bude server očekávat příchozí spojení.

SimpleChatServer(IPAddress ip, int port, IByteToStringConverter converter, IReadingLoopCreator creator)

Nejrozšířenější konstruktor, který vyžaduje instanci koncertru a tovární třídy na „vyčítací smyčky“.

##### Metody

virtual void StartListen()

Virtuální metoda, která započne naslouchání na přednastaveném portu.

virtual void SendMessage(Socket socket, string message)

Virtuální metoda, která odesílá zprávy na zadaný socket.

virtual void Close()

Poděděná metoda, která ukončuje klienta.

##### Události

event SocketHandler SocketAccepted

Událost, která je odpálená po připojení na klienta. V argumentu má socket klienta.

event SimpleMessageHander MessageReceived

Poděděná událost, která je odpálená při přijetí zprávy. Zpráva je předána v argumentu.

event SocketExceptionHandler ReadingInterrupted

Poděděná událost, která je odpálená při chybě objevené při čtení zprávy. Argument obsahuje výjimku a socket, ze kterého bylo čteno.

#### Shrnutí

Pomocí výše uvedených metod a událostí jsme snadno schopni realizovat jakoukoliv komunikaci po síti. Stačí si připravit pevně daný, známý formát zprávy, zapsat ho do objektu, serializovat a odeslat jako textovou zprávu.

Na druhé straně, díky známému formátu budeme schopni objekt deserializovat a data přečíst.

Klient je realizován pomocí dvou vláken. Hlavní vlákno si drží například konzole a pomoci SendMessage může zprávy posílat. Vedlejší vlákno neustále vyčkává na socketu a čte příchozí zprávy.

Server je vícevláknový. V hlavním vlákně neustále čeká na příchozí spojení a jakmile se spoj akceptován, tak vytvoří čtecí vlákno a opět vyčkává na příchozí zprávy od klienta. Pro každého klienta je vlákno nové.

### ChatServer

### ChatKlient

## Uživatelská dokumentace

### ChatServer

### ChatKlient

## Odkazy na použité zdroje

## Závěr