BluChat

Projekt pro „PROGRAMOVÁNÍ (programy INFO ETE15E, SYI ETE56E, TF ETE28E )“  
Jakub Boháček | [xbohj021@studenti.czu.cz](mailto:xbohj021@studenti.czu.cz) | 25/26

Obsah

[Architektura projektu 3](#_Toc196606780)

[Server 3](#_Toc196606781)

[Ukládání dat 4](#_Toc196606782)

[Logování 4](#_Toc196606783)

[Přenos Dat 5](#_Toc196606784)

[Správa uživatelů na serveru 6](#_Toc196606785)

[Připojení na server 6](#_Toc196606786)

[Client 7](#_Toc196606787)

BluChat je projekt typu client-server, navržený jako domácí, open-source textová sociální platforma. Umožňuje uživatelům komunikovat v textových skupinách prostřednictvím klientských aplikací připojených k centrálnímu serveru.

BluChat poskytuje úložiště pro uživatelské účty, zprávy a další data, která mohou být spravována a uchovávána lokálně na serveru. Tento projekt je flexibilní, snadno nasaditelný a zaměřený na ochranu soukromí, což ho činí ideálním pro malé komunity a domácí nasazení.

V projektu jsem se snažil dodržet SOLID principy

* S - Single Responsibility Principle (SRP)
* - Open-Closed Principle (OCP)
* L - Liskov Substitution Principle (LSP)
* I - Interface Segregation Principle (ISP)
* D - Dependency Inversion Principle (DIP)

Celý projekt má v sobě 3 projekty

**Technologie**

Pro projekt jsem zvolil nejnovější verzi .NET 9.

**Nuget Balíčky**

**Simple TCP** Jedná se o jednoduchý TCP/IP klient/server komunikátor, který mi zajištuje komunikaci mezi server a clientem.

**Entity Framework**ORM nástroj, který mi usňadnuje práci s databází v rámci projektu

**MySQLLite**Používám pro ukládání dat na serveru.

**BCrypt**Pro ukládání a ověřování hesel

# Architektura projektu

## Server

Celý Core staví na 2 objektech

Obsah obrázku text, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Server zajišťuje zpracování dat, jejich předávání klientům a ukládání. Používá design pattern Builder pro flexibilní deklaraci a bezchybné vytvoření instance serveru, což umožňuje snadnou konfiguraci pro různé scénáře. Následuje příklad použití v BluChat.ServerConsole:

1. Server.ServerBuilder serverBuild = new Server.ServerBuilder();

2.

3. serverBuild.SetAdress(new IpPort("127.0.0.1", 9000));

4. serverBuild.SetLogger(new Logger());

5. serverBuild.SetDatabase(new SqlLiteContext("BluChat"));

6. serverBuild.SetAdminUserPassword("123456");

7. serverBuild.SetOnClosingEvent();

8.

9. Server server = serverBuild.Build();

10. server.Start();

11.

1. Vytvoří se builder
2. Nastaví se IP adresa serveru
3. Vytvoří se logger pro uchování logů
4. Nastaví se připojení k databázi (zde se dá použít jakákoliv dbContext)
5. Dobrovolně nastavíme Admin uživatele
6. Pokud chceme můžeme nastavit SetOnClosingEvent, který upozorní, že server spadl

### Ukládání dat

Pro ukládání dat jsem zvolil knihovnu **Entity Framework ORM** s databází **MySQL Lite**, která umožňuje lokální uchovávání dat na serveru. Jako strukturu pro správu dat používám **Unit of Work** a **Repository pattern**, což zajišťuje integritu dat a usnadňuje jejich správu. Používám CodeFirst metoduObsah obrázku text, snímek obrazovky, Písmo, číslo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

### Logování

Obsah obrázku text, snímek obrazovky, Písmo, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.Každá akce na serveru je zaznamenána pomocí třídy **Logger**, která je postavena na **interface**, což umožňuje uživatelům implementovat vlastní logovací mechanismy. Logy jsou rozděleny do pěti kategorií podle závažnosti (od informačních po chyby). Pro usnadnění přidávání logů je použit **Factory pattern**, který umožňuje rychlé vytvoření a vložení logů do systému. Logy mohou být ukládány do souborů, ale tato funkce zatím není implementována, protože není potřeba. Po přidání logu je spuštěn **LogAdded event**, který předává informace o přidaném logu, což umožňuje úpravu logovací techniky dle potřeb. Tento přístup je aplikován v **BluChat.ServerConsole** pro přijímání nových zpráv.

### Přenos Dat

Pro přenos dat mezi serverem a klientem jsem vytvořil dvě třídy: **MessageServerManager** a **MessageClientManager**, které zajišťují komunikaci mezi těmito dvěma komponentami. Přenos dat funguje na principu serializace do XML a následného odesílání po síti pomocí **TCP/IP protokolu**.

* **MessageBaseClient**: Obsahuje metodu **MessageHandle**, která se spustí na klientovi a provede potřebné akce po přijetí zprávy.
* **MessageBaseServer**: Obsahuje metodu **MessageHandle**, která se spustí na serveru a vykoná akce dle obdržených dat.

Obě třídy obsahují potřebná data pro správný výkon dané akce.

Obsah obrázku text, snímek obrazovky, Písmo, diagram

Obsah vygenerovaný umělou inteligencí může být nesprávný.

**MessageSerializer** je zodpovědný za serializaci dat do XML. Zprávy jsou organizovány do složek:

* **Client**: Třídy určené pro spuštění na klientovi.
* **Server**: Třídy určené pro spuštění na serveru.

Obsah obrázku text, snímek obrazovky, software, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.Mezi hlavní třídy patří:

* **Message**: Třída, která reprezentuje zprávu.
* **Chat**: Třída, která reprezentuje chatovou místnost.

### Správa uživatelů na serveru

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Obsah vygenerovaný umělou inteligencí může být nesprávný.Uživatelé jsou ukládáni do databáze, přičemž pro bezpečné uchování hesel používám **Bcrypt**. Pro sledování stavu uživatelů na serveru je použit **UserServerStatus**, který uchovává všechny potřebné informace o připojení uživatele. Pro správu a ověřování hesel slouží třída **PasswordManager**.

### Obsah obrázku text, diagram, Paralelní, Plán Obsah vygenerovaný umělou inteligencí může být nesprávný.Připojení na server

## Client

**Client** je zodpovědný za udržení připojení klienta a odesílání zpráv na server. Obsahuje třídu **ClientEvent**, která zahrnuje všechny možné události, které mohou nastat. Pro správné fungování je nutné namapovat všechny události na klienta. Všechny události se provádějí asynchronně, což zajišťuje plynulý a efektivní běh aplikace.

Obsah obrázku text, snímek obrazovky, číslo, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

## BluChat.ServerConsole

Obsah obrázku text, snímek obrazovky, Obdélník, design

Obsah vygenerovaný umělou inteligencí může být nesprávný.Příkazy na konzoli jsou hledány přes **assembler** a spouštěny podle vlastnosti **Name**. Všechny příkazy dědí z abstraktní třídy **Command**, která obsahuje abstraktní metodu **InvokeCommand**, jež spustí příslušnou metodu, a metodu **CheckFormat**, která ověřuje správnost formátu příkazu.

## BluChat.TestClient

**TestClient** slouží pouze jako testovací klient a neměl by být dostupný pro uživatele. Všechny funkce jsou implementovány v jedné třídě. Je možné spustit až 10 testovacích klientů současně pro provádění testů.Obsah obrázku text, snímek obrazovky, displej, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

## Používání

Projekt je automaticky nastaven tak, aby spustil **Server** a **Testovací klient** současně.

K dispozici je příkaz **„Help“**, který vypíše všechny dostupné příkazy. K jakémukoli příkazu lze přidat **„?“** pro zobrazení nápovědy k danému příkazu.

Některé příkazy mají specifický formát zadávání. Například pro příkaz **„UserAdd [username] [password] [password]“** je nutné zadat:

Jakub 123456 123456

Vstupy jsou ošetřeny proti rozdílům mezi velkými a malými písmeny, což zajišťuje flexibilitu při zadávání příkazů. (Jsem si vědom nepřehlednosti výstupu příkazu **Help**.)

### Forms Client

Po zapnutí se zobrazí možnost se připojit na adresu 127.0.0.1:9000, na téhle server automaticky běží(pokud nedojde ke změně).

**Je možné potřeba udělat výjimku pro Antivirus na port !**

Po kliknutí na connect se uživatel připojí jako anoynmní uživatel na server (je možné vyzkoušet příkazem **status**, kde bude vidět IPadresa).

Pokus se server nenajde, program vyhodí chybu, jelikož se jedná o testovací klient nikoliv produkční. Totéž platí pro výpadek serveru

1 Zde je vidět jak jsem se připojil jako anonymní uživatel

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Po spuštění je třeba zadat **login**, přičemž automaticky je vytvořený uživatel **Admin** s heslem **123456**. Po úspěšném přihlášení se zobrazí chat, kam můžete posílat zprávy. Automaticky je vytvořený chat **„TestChat“** pro testování zpráv.

Vpravo jsou zobrazeny všechny zprávy, které přicházejí na server (nejedná se o logy). Po dvojkliku na jakoukoli zprávu se zobrazí více informací, které byly odeslány na client.