Projektová Dokumentace – Chat room

1. Základní informace

Název projektu: Chat Server & Client
Autor: Martin Prokš
Kontakt: martinproks06@seznam.cz
Datum vypracování: 23.11.2025
Škola: SPŠE Ječná

2. Specifikace požadavků

Aplikace umožňuje komunikaci mezi více uživateli v reálném čase pomocí klient-server architektury.

Funkční požadavky:

- Server umožňuje více současných připojení.
- Každému uživateli je při připojení přiřazeno jméno.
- Uživatel může změnit své jméno příkazem /name.
- Uživatel může získat počet online uživatelů příkazem /who.
- Zprávy od uživatelů jsou rozesílány všem ostatním klientům (broadcast).
- Server automaticky odpojí neaktivní uživatele.

Nefunkční požadavky:

- Spolehlivost: aplikace musí zvládat neočekávaná odpojení.
- Výkon: zpracování zpráv musí probíhat asynchronně.
- Logování: všechny události jsou ukládány do log souboru.

Use Case scénáře:

- Uživatel se připojí k serveru.
- Uživatel odešle zprávu.
- Uživatel změní jméno.
- Uživatel ukončí spojení.
- Uživatel je odpojen kvůli neaktivitě.

3. Architektura aplikace

Aplikace je rozdělena na dvě hlavní části: klientskou a serverovou.

Serverová architektura:

- ChatServer – hlavní třída serveru, spravuje seznam klientů a vytváří obslužná vlákna.
- ClientHandler – zpracovává komunikaci s jednotlivým klientem.
- BroadcastWorker – vlákno zajišťující rozesílání zpráv všem klientům.

• LoggerWorker – vlákno starající se o ukládání logů.
• TimeoutWatcher – hlídá neaktivní uživatele.

Klientská architektura:

• ChatClientGUI – Swing GUI aplikace, která se připojuje k serveru.

Komunikace mezi komponentami probíhá pomocí front LinkedBlockingQueue, což zajišťuje bezpečný vícevláknový přístup.

4. Behaviorální popis aplikace

Průběh komunikace mezi klientem a serverem:
1. Uživatel spustí klientskou aplikaci a zadá své jméno.
2. Klient se připojí k serveru.
3. Server vytvoří nový ClientHandler.
4. Zprávy od uživatele se posílají do broadcast fronty.
5. BroadcastWorker rozesílá zprávy všem klientům.
6. TimeoutWatcher kontroluje neaktivitu.
7. LoggerWorker ukládá všechny události do logu.

5. Použité knihovny a závislosti

Aplikace využívá standardní knihovny Javy:
• java.net – síťová komunikace
• java.io – práce se streamy
• java.util.concurrent – vícevláknové fronty
• javax.swing – GUI klienta
• org.junit.jupiter – testy

6. Právní a licenční aspekty

Projekt je vypracován pro školní účely. Kód může obsahovat části inspirované ukázkami z výuky. Nevztahují se na něj žádné komerční licence.

7. Konfigurace programu

Server používá konfigurační soubor server.properties:

port = číslo portu serveru
inactivity_limit_ms = maximální povolená neaktivita
log_max_size = maximální velikost log souboru

Pokud konfigurační soubor neexistuje, použijí se výchozí hodnoty.

8. Instalace a spuštění

Spuštění serveru:

1. Připravte JDK nebo JRE.
2. Spusťte ChatServer (např. přes IDE nebo příkaz: java ChatServer).

Spuštění klienta:

1. Spusťte ChatClientGUI.
2. Zadejte své jméno.
3. Komunikujte přes GUI.

9. Chybové stavy

• Nelze se připojit k serveru – klient zobrazí chybu.
• Server spadne – všichni klienti ztratí spojení.
• Neplatný vstup – ignorován.
• Neaktivita klienta – TimeoutWatcher jej odpojí.

10. Testování a validace

Projekt obsahuje JUnit testy:

• BroadcastWorkerTest – testuje rozesílání zpráv.
• ClientHandlerTest – ověřuje změnu jména.
• LoggerWorkerTest – test logování do souboru.
Testy ověřují klíčovou funkcionalitu aplikace.

11. Verze a známé chyby

Známé problémy:
• Klient se při pádu serveru nemusí korektně uzavřít.
• GUI je jednoduché a nemá pokročilé funkce.

12. Databázové schéma

Aplikace v této verzi databázi nevyužívá.