Sesyjny serwer plików

Dane autorów i ich wkład w poszczególne części projektu

- Jakub Gadomski :
    - Realizacja komunikacji klient - serwer
    - Projekt oraz realizacja klienta serwera plików i serwera sesji
    - Dokumentacja
    - Testowanie
- Dawid Szymczyk :
    - Realizacja algorytmu różnicowego
    - Projekt oraz realizacja serwera plików
    - Projekt oraz realizacja serwera sesji
    - Wdrożenie **Java Persistant Api** w formie połączenia z bazą danych sqlite3

Krótki opis celu programu

Celem programu jest stworzenie rozproszonej aplikacji do przechowywania, synchronizacji oraz zarządzania informacjami użytkownika w sposób zdalny i bezpieczny. System wykorzystuje technologię Java Remote Method Invocation (RMI), umożliwiając komunikację pomiędzy klientem a serwerami w architekturze klient-serwer.

Główne zadania aplikacji to:

- Przechowywanie danych użytkownika na serwerze plików w sposób zoptymalizowany pod kątem przesyłu i synchronizacji informacji.
- Efektywna synchronizacja plików przy wykorzystaniu różnicowego algorytmu z rodziny Rsync, który minimalizuje ilość przesyłanych danych poprzez identyfikowanie i przesyłanie jedynie fragmentów, które uległy zmianie.
- Zarządzanie sesjami użytkowników dzięki dedykowanemu serwerowi sesji, który autoryzuje użytkowników, przydziela im unikalne sesje oraz kontroluje dostęp do zasobów serwera plików.

- Umożliwienie interakcji użytkownika z systemem za pomocą aplikacji klienckiej, która pełni rolę interfejsu do przesyłania, pobierania i zarządzania danymi.

Aplikacja została zaprojektowana z myślą o modularności, bezpieczeństwie i efektywności, co czyni ją odpowiednią do zastosowań edukacyjnych oraz jako baza dla bardziej zaawansowanych systemów rozproszonych.

Opis i schemat struktury logicznej aplikacji

Aplikacja składa się z następujących głównych komponentów:

- Serwer plików :
  - Realizuje założenie algorytmu różnicowego należącego fo rodziny algorytmów Rsync
- Serwer sesji :
  - Dysponuje sesją użytkowników oraz przydziela dostęp do zasobów
- Klient :
  - Stanowi bramę do systemu oraz pozwala na manipulację danych

Schemat struktury logicznej znajduje się w oddzielnym pliku

Informacje o wykorzystanych klasach niestandardowych znajdują się w oddzielnym pliku

Opis specyficznych metod rozwiązania problemu

Do rozwiązania konkretnych aspektów problemu zostały zastosowane następujące metody i technologie:

- Zastosowanie mechanizmu ReentrantLock

  W celu zapewnienia bezpiecznego dostępu do współdzielonych zasobów (np. danych sesji lub plików) w środowisku wielowątkowym, zastosowano mechanizm ReentrantLock. Umożliwia on precyzyjne kontrolowanie blokad, wspierając bardziej zaawansowane operacje niż synchronizacja słówkiem synchronized, takie jak próba uzyskania blokady z timeoutem lub możliwość ręcznego zwolnienia blokady.

- Wykorzystanie JPA

  Do odwzorowania obiektów aplikacji na relacyjną bazę danych zastosowano JPA. Dzięki temu możliwa jest integracja warstwy aplikacji z bazą danych w sposób obiektowy, z zachowaniem czytelności i przenośności kodu. JPA upraszcza zarządzanie trwałością danych i ich zapisem do bazy.

- Integracja z JDBC i bazą danych SQLite3

W warstwie niskopoziomowej, komunikacja z bazą danych odbywa się przy użyciu JDBC , przy czym jako silnik bazy danych wykorzystano SQLite3 — lekką, osadzoną bazę danych w formie pojedynczego pliku. Dzięki temu możliwe jest szybkie i lokalne przechowywanie danych użytkowników oraz informacji sesyjnych bez potrzeby instalacji zewnętrznych systemów bazodanowych.

Krótka instrukcja obsługi

Instalacja

Aby poprawnie zainstalować i uruchomić aplikację, należy wykonać poniższe kroki:

1. Zainstaluj wymagane środowisko i narzędzia:
    - Java Development Kit w wersji 8
    - Apache Maven – do zarządzania projektem i zależnościami
    - SQLite3 – jako lokalna baza danych
2. Zbuduj projekt przy użyciu Maven:

   ```
   mvn clean install
   ```

3. Uruchom skrypt `./firstrun.sh` odpowiedzialny za inicjalizację środowiska

Uruchomienie

Po zainstalowaniu aplikacji, uruchom ją używając następujących komend:

1. Uruchomienie serwera
   ```
   java -cp target/mneme-1.0-SNAPSHOT.jar app.apollo.server.App
   ```

2. Uruchomienie klienta
   ```
   java -cp target/mneme-1.0-SNAPSHOT.jar app.apollo.client.App
   ```

Podstawowe operacje

- Rejestracja
- Logowanie
- Wylogowanie
- Wysyłanie plików
- Pobieranie plików

- Usuwanie plików
- Listowanie plików

Ograniczenia programu

Program posiada następujące ograniczenia:

- Brak obsługi wielu równoległych klientów o wysokim poziomie obciążenia

  Aplikacja nie została zoptymalizowana pod kątem dużej liczby jednoczesnych połączeń – przy wielu aktywnych użytkownikach mogą występować opóźnienia lub konflikty dostępu do zasobów. Limit jednocześnie aktywnych użytkowników to ok. 20.

- Brak zaawansowanego mechanizmu uwierzytelniania i szyfrowania danych

  Obecna wersja wykorzystuje jedynie podstawowe metody zarządzania sesjami, bez pełnej implementacji bezpieczeństwa.

- Ograniczone możliwości przeszukiwania i filtrowania danych

  Interfejs użytkownika nie zawiera rozbudowanych funkcji wyszukiwania ani sortowania danych – operacje na danych są ograniczone do podstawowych czynności (dodaj, usuń, synchronizuj).