DICȚIONAR - Key-Value store

-Sisteme Distribuite-

Barbălată Iuliana

Bozai Maria

Giurge Diana

Cuprins

1. Context
2. Specificații
   1. Cerințe funcționale
   2. Cerințe non-funcționale
3. Scop
4. Obiective SMART
5. Arhitectura
6. Diagrama Gantt
7. Implementare
8. Context

Un grup de utilizatori contribuie la un dicționar de termeni specifici unui anumit domeniu (asemânător sistemului Wikipedia). Pentru fiecare termen există un text explicativ (updatabil) confirmat de o majoritate a utilizatorilor.

1. Specificații
   1. Specificații funcționale

* Fiecare utilizator poate introduce o nouă pereche cheie-valoare (termen + explicație) sau poate actualiza valoarea unei chei existente în sistem.
* Fiecare actualizare (inserare, modificare, ștergere) este realizată doar dacă majoritatea utilizatorilor existenți în sistem confirmă această actualizare.
* Fiecare utilizator poate căuta un termen, iar dacă termenul există în sistem se va afișa textul explicativ.
* Un utilizator poate intra sau ieși oricând din sistem.
* Un utilizator poate înregistra un flux de date (1000 perechi key-value), caz în care confirmarea se poate efectua automat (confirmare aleatoare).
  1. Specificații non-funcționle
* Sistemul se va implementa în Java CLI (consolă / linie de comandă).
* Pentru fiecare termen se va genera o cheie (hash). Fiecare pereche cheie-valoare va fi înregistrată pe cel puțin 2 noduri (factor de replicare 2).
* Nu se va implementa interfața grafică (GUI, web).
* Nu se vor utiliza baze de date (MySQL, MongoDB, PostgreSQL), excepție: Redis.
* Se va evidenția toleranța la defectare.
* Se va implementa un sistem de log-uri pentru a se urmări mesajele din sistem.
* Numărul de utilizatori din sistem este variabil (nu este fixat apriori).

1. Scop

Scopul aplicației este să ofere o experiența de colaborare robustă și să ofere funcționalități avansate, cum ar fi căutarea semantică și monitorizarea în timp real a actualizărilor, pentru a îmbunătăți și accelera procesul de dezvoltare a dicționarului.

1. Obiective Smart

* Specifice

Definirea și implementarea unei interfețe de utilizator ușor de folosit, ca să permită adăugarea, modificarea și ștergerea de perechi cheie-valoare în dicționar.

* Măsurabile

Crearea unui sistem de monitorizare pentru a urmări numărul de perechi cheie-valoare adăugate în dicționar într-un interval de timp specific (o zi).

* Atinse

Asigurarea replicării datelor pe cel puțin două noduri pentru a evita pierderea datelor în caz de defectare a unui nod.

* Relevante

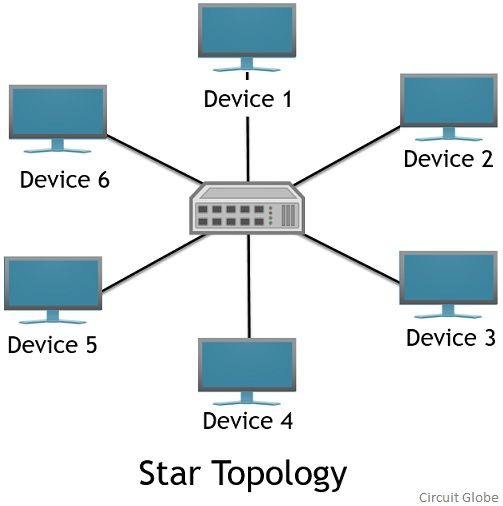
Integrarea funcționalităților de securitate pentru a proteja datele stocate în dicționar.

* Limitate de Timp

Finalizarea implementării interfeței de utilizator și a funcționalităților de bază până la termenul limită pus de profesor.

1. Arhitectura

Utilizarea topologiei în stea pentru această aplicație aduce mai multe beneficii în contextul colaborativ al gestionării dicționarului de termeni. Topologia în stea presupune că toate nodurile (utilizatorii) sunt conectate direct la un nod central (sistemul de gestionare a dicționarului). Avantajele acestei topologii sunt:

* Centralizarea Controlului: Nodul central acționează ca un punct centralizat de control, facilitând administrarea și coordonarea întregii rețele.
* Eficiența Comunicării: Comunicarea între utilizatori și sistemul central este directă și eficientă.
* Securitate îmbunătățită: Accesul neautorizat poate fi mai bine gestionat, iar log-urile pot fi gestionate central, facilitând detectarea și intervent, ia rapidă în caz de activitate suspectă.
* Ușurința în Administrare: Topologia în stea simplifică administrarea rețelei. Orice modificare sau actualizare poate fi gestionată central, fără a necesita modificări la nivelul tuturor nodurilor individuale.
* Actualizări și Backup-uri: Un nod central facilitează implementarea replicării datelor, backup-urilor și a mecanismelor de confirmare a actualizărilor.
* Performanță și Scalabilitate: Topologia în stea este ușor de scalat prin adăugarea de noi noduri.

1. Diagrama Gantt

A screenshot of a calendar

Description automatically generated

1. Implementare

Acest cod este pentru un system de dictionar, conceput pentru a permite utilizatorilor sa adauge, sa actualizeze, sa stearga sau sa caute termini intr-un dictionar, cu functionarea de confirmare a actiunilor de actualizare sau de stergere.

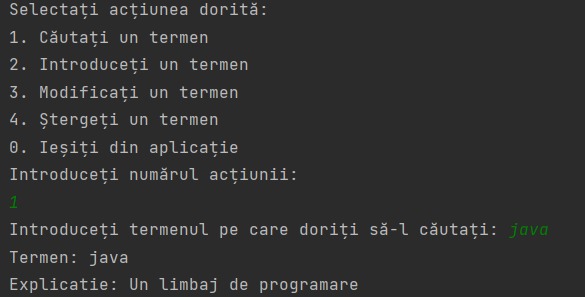
Clasa ‘DictionarySystem’ gestioneaza dictionarul propriu zis si actiunile disponibile pentru utilizatori. Are metode pentru adaugarea, actualizarea, stergerea si cautarea termenilor in dictionar, iar utilizatorii pot confirma actiunile de actualizare sau stergere. De asemenea, pastreaza un si stem de loguri pentru inregistrarea actiunilor utilizatorilor. Clasa ‘User’ reprezinta entitatea utilizatorului si stocheaza numele utilizatorilor. Clasa ‘Main’ e clasa principala, in care sunt create obiectele.

Sistemul utilizeaza interactiunea cu utilizatorul prin intermendiul comenzilor de la consola pentru a permite efectuarea actiunilor specific acestui dictionar. Functionalitatea de confirmare a actiunilor (‘confirmAction’) permite utilizatorilor sa confirme sau sa refuze actiunile care pot modifica sau sterge date din dictionar.

Termenii din dictionar sunt salvati intr-un ‘HashMap’, iar cheile pentru acesti termeni sunt generate folosind functia ‘hashCode()’.

1. In momentul adaugarii unui termen nou in dictionar (‘addTermToDictionary()’), cheia asociata termenului este generata prin conversia valorii returnate ‘hashCode()’ intr-un sir de caractere folosind ‘Integer.toString(term.hashCode())’.
2. Termenul impreuna cu explicatia lui sunt adaugate in ‘keyValueMap’, unde cheia este generata din hash-ul termenului. Deci, ‘keyValueMap’ contine perechi cheie-valoare, unde cheile sunt reprezentate de hash-urile termenilor si valorile sunt explicatiile acestora

Link cod GitHub: <https://github.com/iuliana24/Sisteme-distribuite.git>



A screen shot of a computer

Description automatically generated