Documentul de descriere a design-ului software

29 Noiembrie, 2020
Versiune 1.0

Web Crawler

Academia Tehnica Militara "Ferdinand I"
Realizat de:
Sd.sg.maj. Andrei Claudia
Sd.sg.maj. Chiriță Gabriela
Sd.sg.maj. Guțu Bogdan
Sd.sg.maj. Manea Sebastian
Sd.sg.maj. Mercheș Diana

Cuprins

1. Scopul documentului

Documentul de descriere a design-ului software are scopul de a constitui un ghid unic de implementare a unui Web Crawler.

În acest document sunt prezentate detaliile de implementare necesare pentru a satisfice cerințele specificate în documentul cu Specificația Cerințelor Software(SRS). Mai mult de atât, documentul facilitează înțelegerea sistemului, oferind o vedere mai larga de proiectare a acestuia.

2. Conținutul documentului

Descrierea designului definit în acest document are mai multe scopuri:

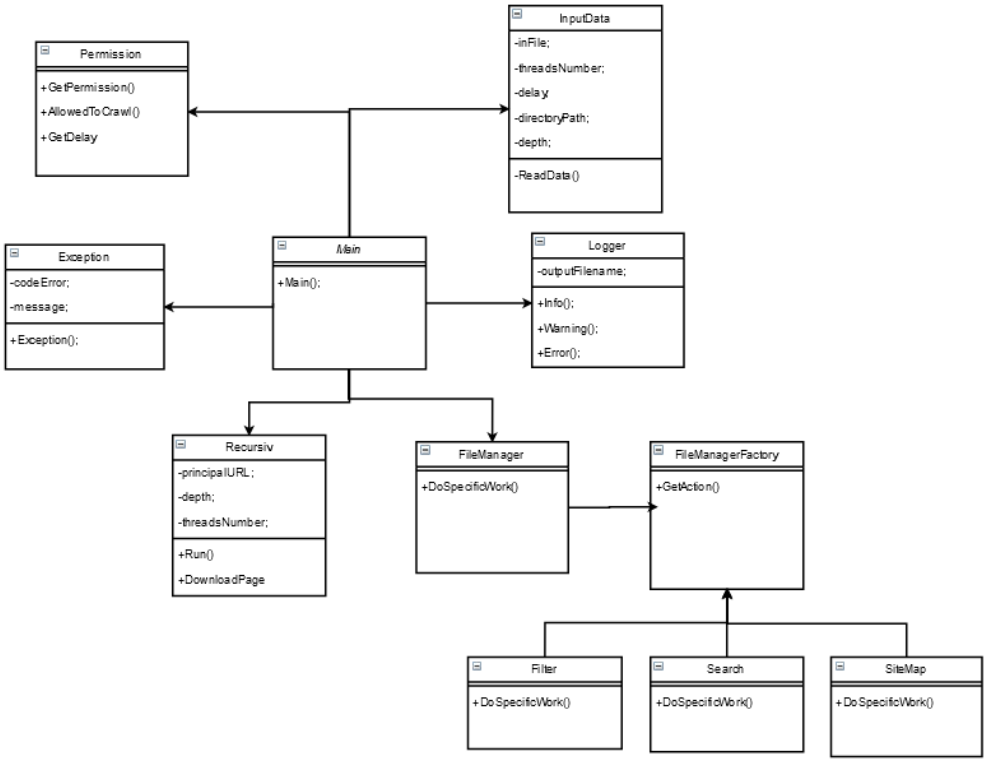
- Prezentarea principalelor structuri de date folosite și a algoritmilor implementați
- Prezentarea șabloanelor arhitecturale folosite, a arhitecturii întregului sistem
- Prezentarea modului de utilizare a aplicației
- Prezentarea componentelor critice și a alternativelor de proiectare a acestora

3. Obiective

Obiectivele documentului prezent sunt:

- Vizualizarea evoluției proiectului din faza incipientă până la faza finală a acestuia.
- Comparație cu modelul architectural descris în documentul curent.
- Realizarea cu success a elementelor de testare.
- Realizarea aplicației finale ce permite descărcarea și indexarea conținutului de pe Internet, cu scopul organizării datelor extrase.

4. Diagrama claselor



Inițial, programul va apela clasa principală Main. În clasa Main, vom instanția în funcție de comenzile primite de utilizator obiectele necesare operațiilor ce urmează să fie executate.

Clasa Permission oferă informații despre permisiunile pe care le deține aplicația.

În clasa Recursiv se va apela funcția Run() prin intermediul căreia se vor executa operațiile de descărcare a fișierelor. Toată această operațiune va fi împărțită pe thread-uri, în funcție de o împărțire egală, evitând astfel o suprasolicitarea a unuia din thread-uri.

Clasa FileManager este responsabilă de analiza fișierelor executând diferite operații. Acesta poate filtra informațiile în funcție de tip, dimensiune și data descărcării fișierelor. De asemenea poate căuta anumite resurse în funcție de niște cuvinte cheie, de a seta dimensiunea fișierelor pe care dorește să le descarce sau poate crea un sitemap folosind resursele descărcate.

Clasa Logger va conține un jurnal de activități realizate la nivelul programului, iar clasa Exception va arăta posibilele erori ce pot apărea în cadrul aplicației.

Clasa de InputData este responsabilă de executarea operațiilor de citire apărute la nivelul aplicației.

5. Modelul datelor

5.1. Structuri de date globale

Principala structura de date globală folosită este o instanță a clasei Main. Clasele disponibile tuturor componentelor arhitecturii sunt:

- InputData
  - Conține algoritmul de citire a fișierului de utilizare.
  - Metoda ReadData() primește ca parametru numele fișierului de intrare și salvează în variabile informațiile de interes: numărul de thread-uri, delay-ul, calea către directorul ce urmează să salveze paginile descărcate și adâncimea.
- Exceptions
  - Conține toate modurile de manipulare a diferitelor erori apărute la nivelul aplicației.
  - Metoda Exception() primește ca parametru o variabilă ce deține codul de eroare și returnează un mesaj corespunzător informațiilor respective.
- Recursiv
  - Conține algoritmii de generare a căutării URL-urilor;
  - Metoda DownloadPage() primește ca parametru calea către fișierul clientului ce conține URL-urile ce vor fi descărcate, Se va apela recursiv pentru a căuta în toate fișierele până la adâncimea specificată, operațiune executată pe thread-uri
- FileManager
  - Metoda DoSpecificWork() primește ca parametru calea unui fișier/director salvat local și un atribut în funcție de care se vor realiza operațiile de filtrare, căutare sau creare de sitemap. Metoda va fi suprascrisă în funcție de utilizare.
  - Clasa folosește factory pattern.

5.2. Structuri de date de legătură

Pentru citirea datelor din fișierele de interes vom folosi un obiect de tipul InputData. Acesta va fi utilizat de metodele care doresc să extragă informații din fișier, apelând metodele valide cu scopul de obținere a datelor.

5.3. Structuri de date temporare

În proiectul curent, nu vom utiliza structuri de date temporare cu rol important.

5.4. Formatul fișierelor utilizate

Aplicația folosește fișiere de tip .txt, .cnf pentru citirea fișierului de configurare și fișiere de tip .html pentru descărcarea locală a resurselor și analiza acestora.

Aplicația va fi dezvoltată în mediul de lucru Java în care se vor implementa funcționalitățile aplicației.

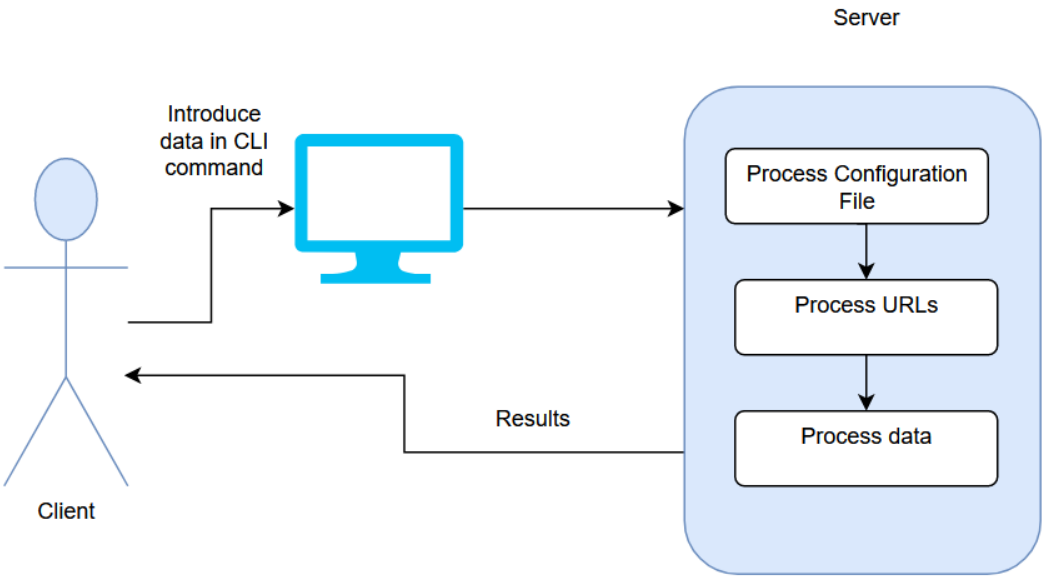
6. Modelul architectural și modelul componentelor

6.1. Arhitectura sistemului

6.1.1. Șabloane arhitecturale folosite

Această aplicație are la baza o arhitectură back-end. Rularea aplicației se face din linie de comandă pe baza unui input al utilizatorului. Componenta back-end oferă sprijinul serviciilor oferite de aplicația Web Crawler, aceasta făcând diferite operații a cărui rezultat va fi mai apoi transmis userului. Userul poate utiliza aplicația din linia de comandă și primește un rezultat pe baza algoritmilor și a codului dezvoltat.

6.1.2. Diagrama de arhitectură



6.2. Descrierea componentelor

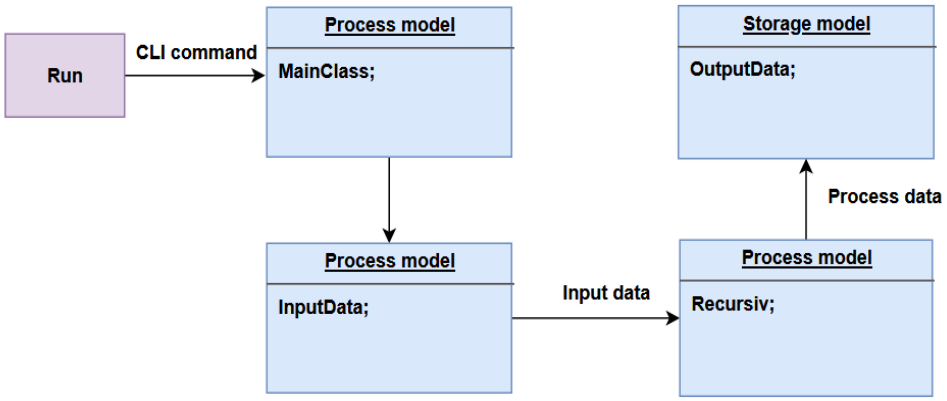
- Modulul de procesare -Responsabil de executarea căutării și verificării URL-urilor.
- Modulul de stocare - Responsabil cu salvarea rezultatelor obținute în urma procesării.

6.3. Restricții de implementare

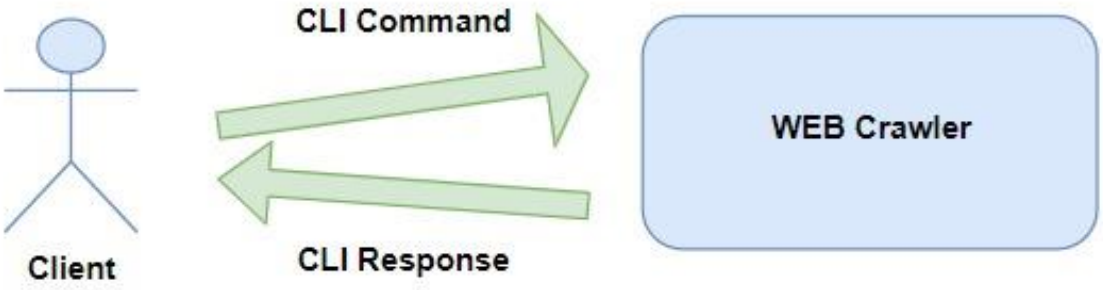
Modulele aplicației trebuie să respecte următoarele restricții de implementare:

- Toate modulele aplicației vor fi dezvoltate utilizând limbajul de programare Java;
- Modulul de procesare va fi realizat pentru fișiere valide, în caz contrar aplicația va ignora fișiere.
- Modulul de stocare nu va permite salvarea fișierelor cu output invalid.
- Respectarea în scrierea comentariilor conform coding-style-ului Java (https://www.oracle.com/technetwork/java/codeconventions-150003.pdf).

6.4. Interacțiunea dintre componente



7. Modelul interfeței cu utilizatorul



Aplicația va fi rulată în linie de comandă. Clientul va introduce inițial numele unui fișier de configurare ce conține numărul de thread-uri, delay-ul folosit, calea către directorul ce urmează să salveze paginile descărcate și adâncimea maximă. Odată verificat fișierul de configurare, user-ul va introduce calea către un fișier ce conține URL-urile ce vor fi descărcate.

8. Elemente de testare

8.1. Componente critice

Funcționalitatea aplicației depinde de fișierul de configurare și de corectitudinea datelor trimise de către client și primirea răspunsurilor fără erori.

Componenta critică este interacțiunea client-Web Crawler deoarece analiza informațiilor se face pe baza datelor introduce de către client. Cu ajutorul procedurilor de testare, datele vor fi validate și aplicația va putea genera informațiile corespunzătoare de ieșire.

8.2. Strategie de testare

Pentru a se efectua procesul de testare a aplicației, va fi nevoie de un calculator având sistem de operare Windows și o bună conexiune la Internet

Pe parcursul evaluării aplicației, se vor avea in vedere testele aflate in "Anexa #1". Rezultatele vor fi trecute in "Anexa #2", respectându-se convențiile stabilite.

8.3. Funcționalitățile testate

Pe parcursul testării, se va ține cont de un tabel cu funcționalități, prezent în "Anexa #1", atașată documentului, și ce are următoarele coloane:

- ID: identificator unic al testului;
- Utilizator: tipul de utilizator care are acces la funcționalitate;
- Funcționalitate: prezentarea succintă a caracteristicii testate;
- Metodologia testării: modul in care se va executa testul;
- Posibile erori: erori ce pot apărea pe parcursul efectuării testului;

8.4. Criterii de evaluare

În procesul de testare, se vor folosi trei criterii de evaluare a testelor:

- acceptat: aplicația se comportă favorabil, în cadrul a trei teste consecutive, de același tip;
- respins: aplicația prezintă erori sau nu se comporta favorabil, în cadrul a trei teste consecutive, de același tip;
- amânat: aplicația trece printr-o serie imprevizibilă de răspunsuri favorabile, erori sau comportament nefavorabil, în cadrul a cinci teste consecutive, de același tip.

Anexa #1: Funcționalități testate

ID	UTILIZATOR	FUNCȚIONALITATE	METODOLOGIA TESTĂRII	POSIBILE ERORI

T1	Client	Validare fișier de configurare	Se introduce fișierul de configurare de către client. Se realizează teste asupra fișierului de configurare pentru a vedea daca datele sunt valide.	○ Fișier de input inexistent. ○ Datele de input sunt invalide.
T2	Client	Validare URL	Clientul introduce de la tastatură URL-ul.	○ URL-ul este invalid.
T3	Client	Obținere informații pe baza URL-ului	Pe baza URL-ului introdus de client se realizează descărcarea locală și analiza fișierelor.	
T4	Client	Web Crawler funcțional	Clientul introduce fișierul de configurare și așteaptă validarea acestuia. Clientul introduce URL-ul paginii de interes. Pe baza URL-ului introdus WebCrawler-ul realizează descărcarea, filtrarea, analizarea datelor si site map-ul.	○ Fișier de input inexistent. ○ Date invalide. ○ URL greșit.

Anexa #2:Rezultate ale testării

ID	T1	T2	T3	T4
1				
2				
3				
4				

Observații:

Coloanele din tabelul “Anexa #2”, corespunzătoare testelor, se vor completa cu simboluri specifice:

✔: acceptat

✘: respins

🤔: amânat

9. Anexa

9.1. Tehnologii folosite

JAVA

9.2. Acronime

SRS	Software Requirements Specification
URL	Uniform Resource Locator
CLI	Command Line Interface