Software Design Description

Crawler Web

30.11.2020

Echipa de dezvoltare(Grupa C114D):

* Mirea Irina
* Călugăru Teodor
* Albu Alexandru
* Ghiță Cristian

Cuprins

[I. Scopul si viziunea 3](#_Toc57578513)

[II. Cerințe funcționale 4](#_Toc57578514)

[Actorii 4](#_Toc57578515)

[Cazurile de utilizare 4](#_Toc57578516)

[Dependințe 6](#_Toc57578517)

[III. Interfața utilizator 6](#_Toc57578518)

[Nume 6](#_Toc57578519)

[Rezumat 6](#_Toc57578520)

[Pentru obținea crawl 6](#_Toc57578521)

[Pentru opțiunea sitemap 7](#_Toc57578522)

[Pentru opțiunea filter\_type 7](#_Toc57578523)

[Pentru opțiunea filter\_size 7](#_Toc57578524)

[Pentru opțiunea search 8](#_Toc57578525)

[IV. Arhitectura 8](#_Toc57578526)

[Modelul arhitectural 8](#_Toc57578527)

[Clasa Crawler 10](#_Toc57578528)

[Clasa Task 10](#_Toc57578529)

[Clasa VisitedPageList 11](#_Toc57578530)

[Clasa TaskQueue 11](#_Toc57578531)

[Clasa Page 11](#_Toc57578532)

[Clasa AllowedFilesList 11](#_Toc57578533)

[Clasa SitemapTask 12](#_Toc57578534)

[Clasa CrawlTask 12](#_Toc57578535)

[Clasa LimitDimensionTask 12](#_Toc57578536)

[Clasa SearchWordsTask 12](#_Toc57578537)

[Clasa FilterTask 12](#_Toc57578538)

[Diagrama de secvențe 13](#_Toc57578539)

[Modelul datelor 15](#_Toc57578540)

[V. Plan de testare 16](#_Toc57578541)

[Lista de verificare 16](#_Toc57578542)

# Scopul si viziunea

Un crawler web, numit adesea și spider sau spiderbot, reprezintă o aplicație care navighează sistematic pe anumite site-uri web, de obicei cu scopul indexării acestora. Care este motivația pentru realizarea unei asemenea aplicații? Poate nu ne-am fi gândit, însă procesul de crawling este esențial pentru funcționarea motoarelor de căutare. Astfel, daca ne dorim să construim un motor de căutare sau să ajutăm la optimizarea unui motor deja existent, acesta ar fi un punct bun de plecare. Cum ajută crawling-ul motoarele de căutare? Răspunsul este dat de faptul că, prin aplicarea unui algoritm de căutare asupra datelor colectate de crawler-ul web, motorul de căutare poate furniza link-uri relevante ca răspuns la interogările utilizatorilor.

Ne propunem realizarea unui crawler web care, primind un fișier cu mai multe URL-uri, va descărca toate resursele acestor pagini(resursele sunt reprezentate de diferite tipuri de fișiere- .js, .html, .jpg, .png etc).În cazul întâlnirii de link-uri în paginile căutate, crawler-ul va apela recursiv până când va ajunge la o pagină ce nu mai are link-uri. Fiecare resursă va fi salvată într-un folder care va avea calea corespunzătoare. De exemplu, dacă găsim resursa index.html în pagina mta.ro aceasta va fi salvată cu calea D:\downloads\mta.ro\index.html, dar dacă găsim resursa index.js în pagina admitere care avea un link în pagina mta.ro atunci vom salva fișierul cu calea D:\downloads\mta.ro\admitere\index.js.

Restricțiile asupra tipurilor de fișiere căutate nu vor fi găsite doar în fișierul robots.txt, ci și într-un fișier de configurare pe care îl poate modifica utilizatorul aplicației. De asemenea, ne dorim să îmbunătățim timpul de execuție al aplicației, astfel că vom implementa o soluție paralelă, pe mai multe thread-uri(fire de execuție).De fiecare dată când un thread găsește un URL sau un link către o anumită pagina, va porni un nou task. În interiorul fiecărui task se vor efectua operațiile de indexare a paginilor și de descărcare a resurselor existente în fiecare pagină.

Astfel, vom avea un program care poate rula pe orice sistem de operare și care va avea performanțe ridicate în ceea ce privește timpul de execuție. De asemenea, aplicația va putea să efectueze anumite operații asupra fișierelor pe care le găsește în fiecare pagină web, având capacitatea de a le filtra după tip sau de a le limita dimensiunea pentru a nu ocupa foarte multe resurse. În plus, se vor putea realiza căutări după anumite cuvinte cheie în aceste fișiere(simulând câteva dintre sarcinile unui motor de căutare care ar putea fi implementat ulterior pe baza crawler-ului), iar pentru anumite site-uri vom avea și prezentări ale organizării lor(prin sitemap-uri).

# Cerințe funcționale

## Actorii

Actorii aplicației vor fi userii crawlerului care doresc să utilizeze funcționalitățile acestuia. Aceștia vor interacționa cu aplicația prin linie de comandă și printr-un fișier .txt dat că parametru, care va conține url-ruile tuturor site-urilor pe care crawlerul trebuie să le viziteze.

## Cazurile de utilizare

Userii vor putea să:

-Descarce paginile respective, și recursiv până la o anumită adâncime, toate paginile spre care există un link.

-Descarce doar anumite tipuri de fișiere(.docx, .pdf, etc)

-Precizeze dacă crawlerul va verifică conținutul fișierului robots.txt și dacă va ține cont de acesta în căutarea să sau nu.

-Precizeze numărul de threaduri pe care va rula aplicația.

-Precizeze fișierul de log

-Precizeze directorul de download.

Pentru folderul unde se află un site deja descărcat crawlerul va putea:

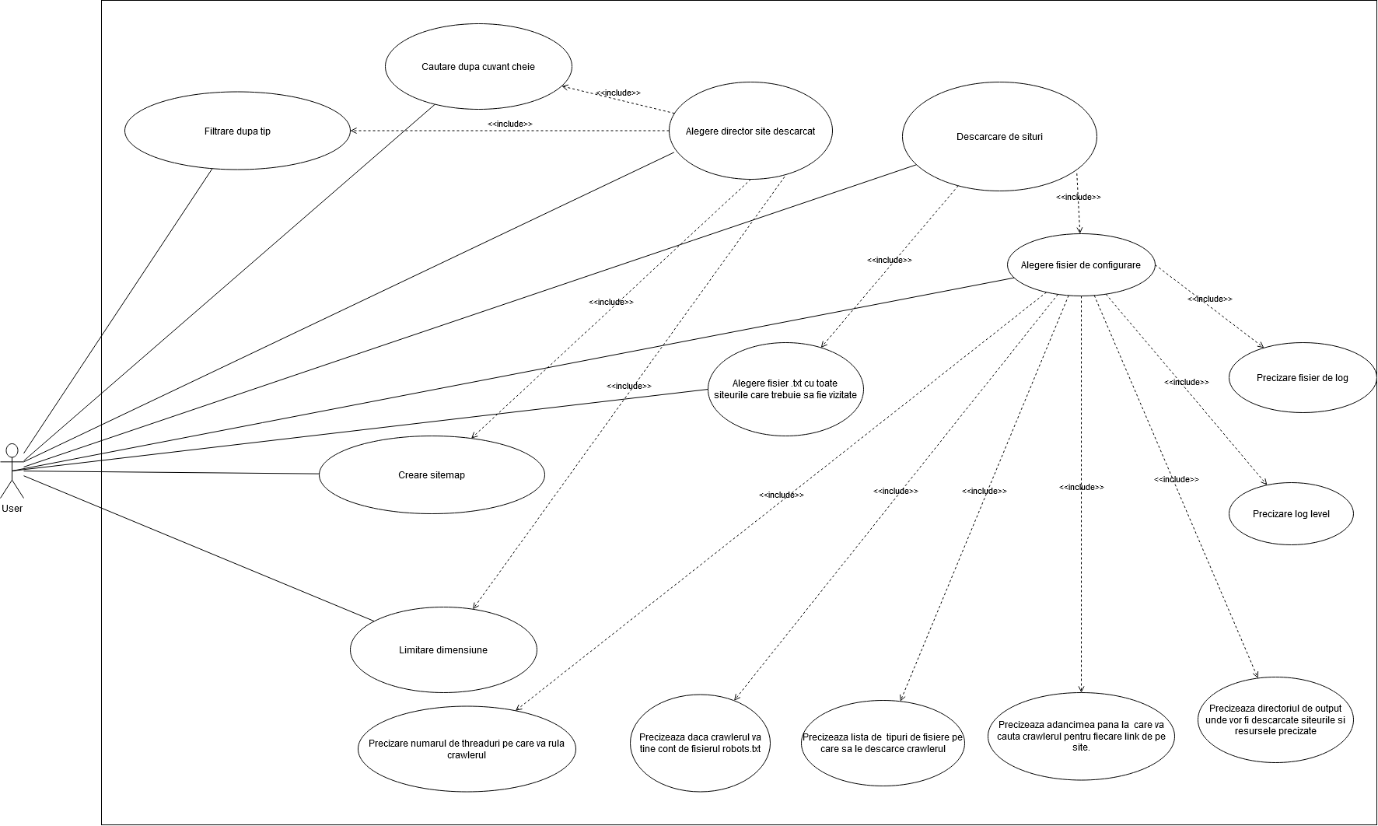
-crea un sitemap

-filtra după tipul fișierelor.

-limita dimensiunea fișierului.

-Caut după cuvinte cheie.

Mai jos este prezentată diagramă de cazuri de utilizare a sistemului:



Aplicația va primi ca parametrii fișierul cu site-urile care trebuie vizitate și un fișier de configurare.

* Fișierul de configurare va avea următorul format:

1. *Nthreads*: (numărul de threaduri) int intre 1 si 50
2. *File\_types*: (tipul fișierelor care vor fi descarcate) stringuri separate prin virgula
3. *Robots.txt*: (daca se va tine cont de robots.txt sau nu) bool
4. *Log\_file*: (Fișierul unde vor fi stocate toate locurile) string
5. *Log\_level*: (Nivelul de printare pentru loguri) int intre 0 si 3

* Intrările din fișierul cu situri vor avea următorul format:

Nume\_site(string):url\_site(string)

* Fișierul de log va conține mesaje 3 tipuri de mesaje:

1. *INFO*: mesaj de informare pentru utilizator
2. *ERROR*: mesaje de eroare care împiedica execuția programului
3. *WARNING*: mesaje de avertisment care semnalează probleme in timpul execuției dar nu opresc execuția.

* Toate mesaje de log vor conține data si ora la care au fost afișate.

1. Crawlerul va scrie in fișierul de log in funcție de nivelul de log:
2. Log\_level 0: Crawlerul nu va scrie nimic.
3. Log\_level 1: Crawlerul va scrie doar mesajele INFO de succes si eroare
4. Log\_level 2: Crawlerul va scrie mesajele INFO de succes, eroare si warning
5. Log\_level 3: Crawlerul va scrie toate mesajele, inclusiv mesaje de tip INFO despre stadiul la care se afla crawlerul.

## Dependințe

Crawlerul va avea nevoie de o conexiune la internet pentru a vizita site-uri, de acea va afișa un mesaj de eroare in log in momentul in care nu exista o conexiune la internet.

# Interfața utilizator

Interfața aplicației va fii in linie de comandă ce va permite încorporarea sa în alte aplicații mai complexe ce vor utiliza datele extrase de aceasta.

## Nume

crawler – interfața din linie de comandă a crawler-ului web

## Rezumat

crawler crawl sites.txt [-conf configFile]

crawler sitemap sites.txt [-conf configFile]

crawler filter\_type sites.txt [-conf configFile]

crawler filter\_size sites.txt [-conf configFile] size

crawler search sites.txt [-conf configFile]

## Pentru obținea crawl

Descarcă toate paginile din componența fișierului sites.txt și recursiv până la o anumită adâncime toate paginile spre care există un link conform opțiunilor specificate în fișierul de configurare.

sites.txt

fișierul din care se vor extrage site-urile. Acestea vor fii de forma Nume\_site(string):URL\_site(string) separate printr-un newline.

[-conf]

se va da ca argument fișierul de configurare pe baza căruia web crawler-ul va opera, acesta trebuie să respecte structura aferentă specificată în SRS. Dacă niciun argument nu este dat, se va folosi fișierul de configurare implicit aplicației.

## Pentru opțiunea sitemap

Se va crea un sitemap pe baza informațiilor stocate local a unuia sau mai multor site-uri. Acesta ...

sites.txt

fișierul din care se vor extrage directoarele site-urilor ce urmează a fii folosite pentru crearea de sitemap-uri

[-conf]

se va da ca argument fișierul de configurare pe baza căruia web crawler-ul va crea sitemap-ul, acesta trebuie să respecte structura aferentă specificată în SRS. Dacă niciun argument nu este dat, se va folosi fișierul de configurare implicit aplicației.

## Pentru opțiunea filter\_type

Se vor filtra fișierele aflate în directorul specificat pentru descărare. Crawler-ul va permite numai existența fișierelor de tipul specificat în fișierul de configurare.

sites.txt

fișierul din care se vor extrage numele directoarelor site-urilor ce urmează a fii filtrate.

[-conf]

se va da ca argument fișierul de configurare pe baza căruia web crawler-ul va efectua filtrarea după tipul fișierului, acesta trebuie să respecte structura aferentă specificată în SRS. Dacă niciun argument nu este dat, se va folosi fișierul de configurare implicit aplicației/nu se va face nicio filtrare .

## Pentru opțiunea filter\_size

Se vor filtra fișierele aflate în directorul specificat pentru descărare. Crawler-ul va permite numai existența fișierelor de mărime mai mică decât cea specificată în fișierul de configurare.

sites.txt

fișierul din care se vor extrage numele directoarelor site-urilor ce urmează a fii filtrate.

[-conf]

se va da ca argument fișierul de configurare pe baza căruia web crawler-ul va opera, acesta trebuie să respecte structura aferentă specificată în SRS. Dacă niciun argument nu este dat, se va folosi fișierul de configurare implicit aplicației.

Size

Dimensiunea maximă a fișierelor permise, exprimată în MB.

## Pentru opțiunea search

Se va căuta în componența directorului aferent site-ului/directorului de fișiere descărcate informatiile ce corespund cuvintelor cheie date

sites.txt

fișierul din care se vor extrage numele directoarelor site-urilor ce urmează a fii căutate.

[-conf]

se va da ca argument fișierul de configurare pe baza căruia web crawler-ul va opera, acesta trebuie să respecte structura aferentă specificată în SRS. Dacă niciun argument nu este dat, se va folosi fișierul de configurare implicit aplicației.

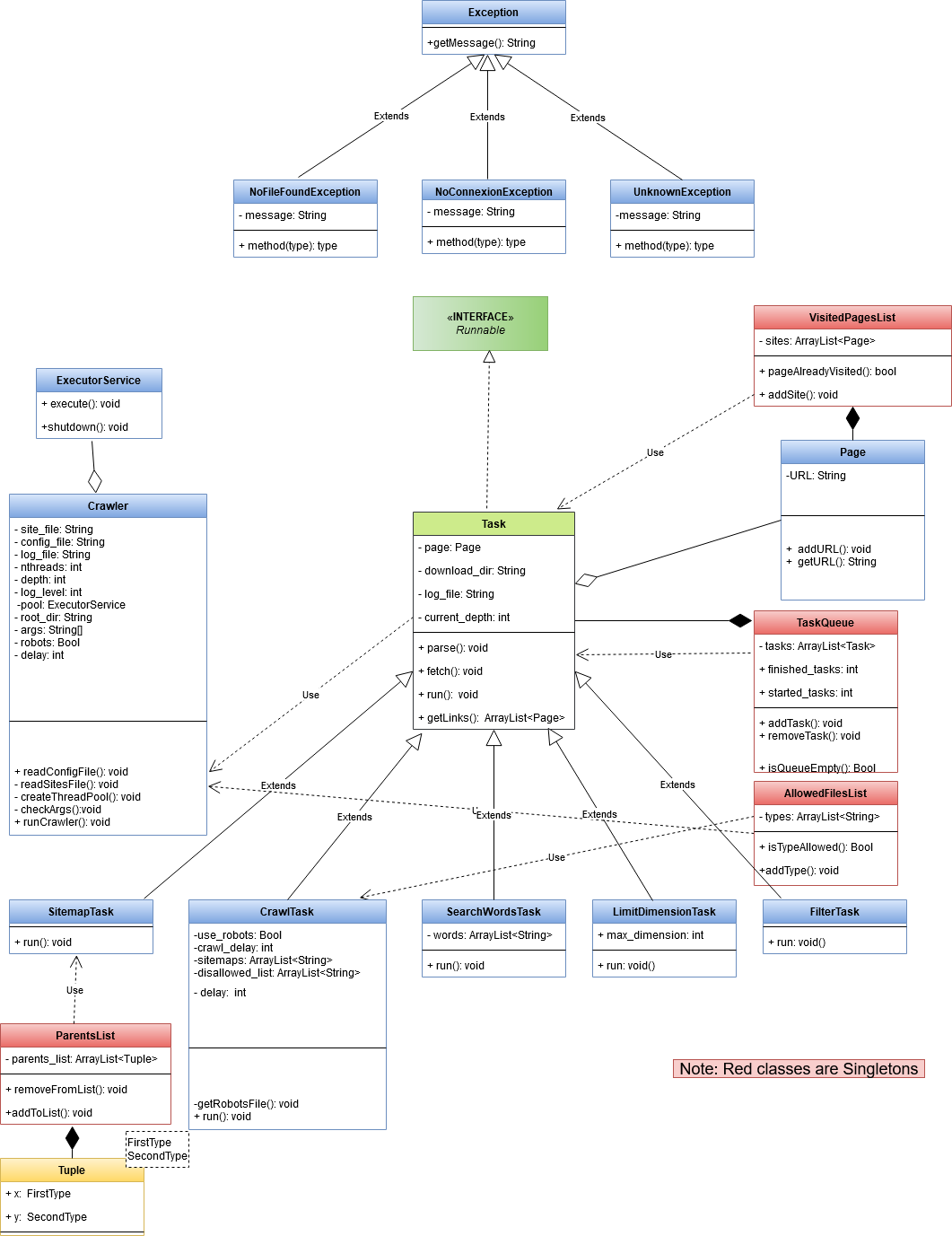
...

Se vor da ca argument cuvintele cheie, separate prin spațiu, pe baza cărora se va face căutarea

# Arhitectura

## Modelul arhitectural

Mai jos este prezentata diagrama de clase a aplicației:



### Clasa Crawler

* Membrii:

1. site\_file: fișierul in care se va afla lista cu site-uri care trebuie indexate.
2. config\_file: fișierul de configurare pe care il va citi mereu crawlerul înainte de a începe execuția.
3. log\_file: fișierul unde se vor scrie toate mesajele de log.
4. nthreads: numărul de threaduri pe care va rula crawlerul.
5. depth: adâncimea pana la care va fi indexat fiecare site.
6. log\_level: reprezintă cat de multe mesaje vor fi scrise in fișierul de log.
7. pool: reprezintă un obiect de tipul ExecutorService care va reprezenta thread pool-ul pe care vor rula taskurile.
8. root\_dir: folderul unde vor fi descărcate toate site-urile.
9. args: array de tipul String cu toate argumentele data de user de la tastatura.
10. robots: bool care va spune daca se va citi fișierul robots.txt de pe site-ul care este indexat.
11. delay: delay-ul precizat de user in fișierul de configurare.

* Metode:
  1. readConfigFile(): va citi fișierul de configurare si va inițializa nthreads log\_file, depth , log\_level, root\_dir, delay si robots cu valorile din fișierul de configurare. De asemena se vor citi si tipurile de fisiere permise si se vor adauga in clasa singleton AllowedFilesList.
  2. readSiteFile(): va citi fișierul de configurare si va crea un obiect Page pe care îl va folosi sa creeze un Task de tipul precizat prin argumente. Dupa aceea se vor adauga taskurile in TaskQueue.
  3. checkArgs(): va verifica daca argumentele date de la tastatura sunt corecte sau nu.
  4. runCrawler(): va începe execuția taskurilor create prin readSiteFile(). Aici se va apela metoda execute() a membrului pool care va asigna primului task din TaskQueue unui thread din thread pool. Dupa aceea se va incrementa membrul started\_tasks din singleton-ul TaskQueue si se va sterge primul element din TaskQueue. Acest lucru se va face cat timp TaskQueue nu este goala si started\_tasks != finished\_tasks.

### Clasa Task

* Membrii:

1. page: obiect al clasei Page, care reprezintă pagina web care va fi indexata de task.
2. download\_dir: fișierul de download al sitelui care va fi format prin adăugarea numelui site-ului la root\_dir.
3. log\_level: log\_level precizat in clasa crawler.
4. current\_depth: nivelul de indexare curent.

* Metode:

1. fetch(): va descărca conținutul paginii.
2. Parse(): va parsa conținutul paginii.
3. getLinks(): va extrage paginile spre care sunt alte linkuri
4. run(): metoda care trebuie implementata pentru fiecare clasa copil.

### Clasa VisitedPageList

* Membrii:

1. sites: lista cu toate site-urile vizitate.

* Metode:
  + 1. pageAlreadyVisited(): verifica daca o pagina a fost deja vizitata.
    2. addSite(): adauga un noua pagina in lista de pagini vizitate.

### Clasa TaskQueue

* Membrii:

1. tasks: lista cu toate taskurile care trebuie efectuate.
2. started\_tasks: numărul de taskuri începute
3. finished\_tasks: numărul de taskuri terminate

* Metode:
  + 1. addTask(): adaugă un nou task in lista.
    2. removeTask(): șterge primul task din lista.

### Clasa Page

* Membrii:

1. URL: adresa url a paginii web

* Metode

1. addURL(): setează membrul url
2. getURL(): returnează url-ul paginii.

### Clasa AllowedFilesList

* Membrii:

1. Parents\_list: lista de obiecte Tuple de tip <String, String>

* Metode:
  + 1. removeFromList(): adaugă un nou tuple in lista.
    2. addToList(): sterge un tuple din lista.

### Clasa SitemapTask

* Metode:
  + 1. run() Va găsi toate linkurile către care pagina curenta are acces si le va adaugă in PatentsList si va crea un nou task pentru fiecare link

### Clasa CrawlTask

* Membrii:

1. use\_robots: Bool care precizează daca se va citi fișierul robots.txt
2. crawl\_delay: delay-ul precizat in fisierul robots.txt
3. sitemaps: lista cu sitemapurile găsite in robots.txt
4. disallowed\_list: lista cu toate fișierele care nu trebui descărcate din robots.txt
5. delay: delay-ul precizat de user in config file.

* Metode:
  + 1. getRobotsFile(): va citi fișierul robots.txt si va inițializa disallowed\_files, craw\_delay si sitemaps.
    2. run(): va descărca conținutul paginii si va crea cate un task pentru fiecare link spre care pagina are acces.

### Clasa LimitDimensionTask

* Membrii:

1. max\_dimension: precizează mărimea maxima pentru un fișier.

* Metode:

1. run(): se va uita pe fiecare pagina si va descarca doar paginile cu marimea mai mica decat max\_dimension.

### Clasa SearchWordsTask

* Membrii:

1. words: lista cu toate cuvintele care trebuie cautate.

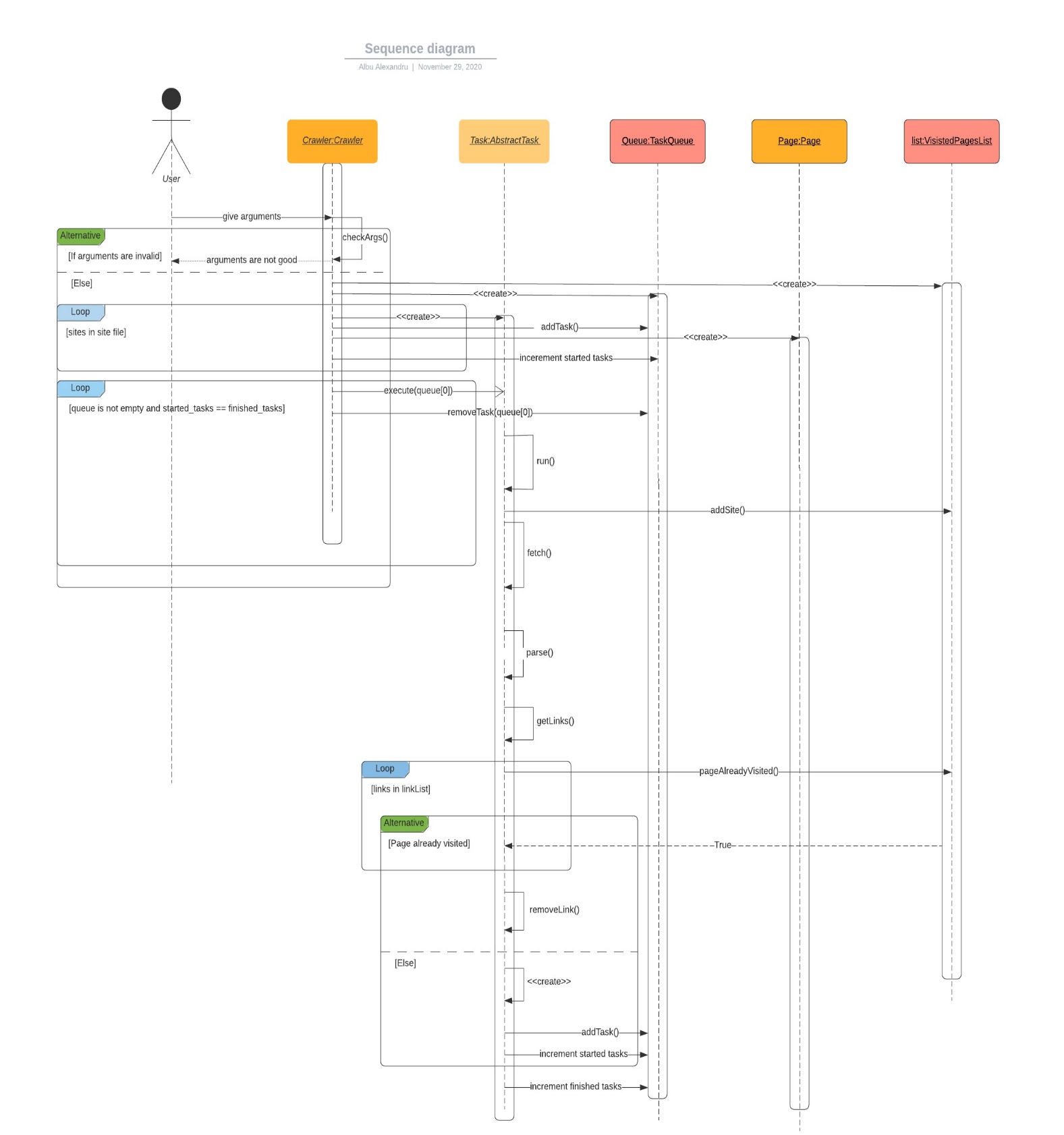
* Metode:
  + 1. run(): Va cauta pentru obiectul page cuvintele din lista si daca le va gasi le va afisa. Pentru fieacare link din pagina se va face un nou task.

### Clasa FilterTask

* Metode:
  + 1. run(): va accesa resursele paginii si va pastra doar resursele precizare in AllowedFilesList. Pentru fiecare link din site va crea un nou task.

### Diagrama de secvențe

Mai jos este prezentata diagrama de secvențe pentru folosirea clasei abstracte de task din care sunt moștenite celelalte taskuri. Aceasta arata un workflow general pentru toate taskurile din sistem, diferind doar prin existenta unor metode specifice pentru fiecare task.



## Modelul datelor

Clasa ExecutorService:

Tip clasa: clasa concreta

Va fi utilizata de catre clasa Crawler pentru a crea un thread pool.

Clasa Crawler

Tip clasa: clasa concreta

Va fi utilizată pentru a procesa argumentele date de user, urmand sa porneasca taskurile corespunzătoare, sa pornească threadurile din thread pool-ul si pentru a instantia coada de taskuri , lista de fișiere permise si lista de taskuri vizitate.

Clasa Task:

Tip clasa: clasa abstracta

Va fi utilizata pentru a fi moștenita pentru toate celelalte taskuri concrete.

Clasa CrawlTask:

Tip clasa: clasa concreta

Va fi utilizata pentru a face indexarea site-urilor

Clasa SitemapTask:

Tip clasa: clasa concreta

Va fi utilizata pentru a crea sitemapul unui site descarcat.

Clasa SearchWordsTask:

Tip clasa: clasa concreta

Va fi utilizata pentru a cauta o lista de cuvinta data de utilizator in site-urile descarcate.

Clasa LimitDimensionTask:

Tip clasa: clasa concreta

Va fi utilizata pentru a păstra doar fesierele cu o anumita dimensiunea a site-rilor.

Clasa FilterTask:

Tip clasa: clasa concreta

Va fi utilizata pentru a păstra doar fesierele dorite de utilizator intru-un site.

# Plan de testare

Testarea reprezintă rularea programului în scopul de a descoperi bug-uri de software si reprezintă o etapă importantă în dezvoltarea produsului software . În urma testării se poate, sau nu, confirma faptul ca produsul software supus testării corespunde cerințelor care au dus la crearea acestui produs și ca funcționează corespunzător așteptărilor.

O abordare obișnuită în ceea ce privește testarea crawlerelor este crearea unei liste de verificare.

## Lista de verificare

1. Testarea întregii adâncimi de accesare cu crawlere: se confirmă că datele capturate sunt structurate corect la fiecare nivel.

2. Verificarea datelor: se confirmă că au fost stocate corespunzător și se potrivesc cu datele din sursă.

3. Deschiderea documentelor accesate cu crawlere din fiecare director (accesat cu crawlere) pentru a vedea acuratețea conținutului.(sa corespundă cu site-ul)

4.Verificarea logurilor după fiecare accesare cu crawlere: logurile pot dezvălui probleme chiar dacă raportează o accesare cu crawlere reușită.

5. Verificarea fișierelor finale: dacă a fost realizată prelucrarea în conformitate cu cerințele. (operațiuni ce trebuie să fie funcționale :Filtrare după tip, căutare după cuvinte cheie, limita dimensiune, creare sitemap).

După ce se vor îndeplini testările din lista de verificare se va face testarea funcțională a aplicației. Testarea funcțională a aplicației verifică dacă un produs software se comportă și funcționează corect, conform cerințelor și specificațiilor. Pe baza specificațiilor funcționale ale aplicației vom crea cazuri pentru testare.