UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAŞI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENŢĂ

Surf

Parcurgere şi procesare distribuită a paginilor web în cloud pentru extragerea informaţiilor

propusă de

Ovidiu Pricop

Sesiunea: Iulie, 2017

Coordonator ştiinţific

Conf. dr. Sabin Corneliu Buraga

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAŞI

FACULTATEA DE INFORMATICĂ

Surf

Ovidiu Pricop

Sesiunea: Iulie, 2017

Coordonator ştiinţific

Conf. dr. Sabin Corneliu Buraga

Declaraţie privind originalitate şi respectarea drepturilor de autor

Prin prezenta declar că lucrarea de licenţă cu titlul "Surf" este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;
- codul sursă, imaginile etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
- rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi, 3 Iulie 2017

Ovidiu Pricop

Declaraţie de consimţământ

Prin prezenta declar că sunt de acord ca Lucrarea de licenţă cu titlul "Surf", codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică. De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea Alexandru Ioan Cuza Iaşi să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi, 3 Iulie 2017

Ovidiu Pricop

Rezumat

La nivelul internetului, traficul global a crescut de aproximativ 850 de ori in perioada 2000 - 2015 [1]. World wide web-ul reprezinta o parte semnificativa a volumului de informatii interschimbate pe internet. In anul 2015 existau peste jumatate de miliard de situri web accesibile [2]. Fiecare pagina web raspunde anumitor nevoi (sociale, financiare, educationale etc.). O singura sursa de informatii este uneori suficienta pentru a raspunde nevoilor unui utilizator. Alteori, este necesar un ansamblu de surse informative (e.g. newsletter-e zilnice din 5 surse diferite), pentru a urmari un subiect din mai multe perspective sau a-i intregi continutul.

Prezenta lucrare urmareste elaborarea unui serviciu web distribuit in cloud pentru parcurgerea, selectarea, colectarea si indexarea informatiilor la nivelul world wide web-ului. Serviciul se adreseaza acelor persoane care vor sa automatizeze sarcinile de extragere a informatiilor web si garanteaza usurinta in utlizare, flexibilitate, extensibilitate si control precis asupra costurilor.

Cuprins

Rezumat **4**

Introducere **6**
1 Descrierea unui crawler web 7
2 Design-ul unui crawler 7
2.1 Frontiera 8
2.2 Descarcarea paginilor web 8
2.3 Stocarea datelor 8
3 Crawler-ul Surf 9
3.1 Puncte de pornire 9
3.2 Parcurgere 9
3.3 Selectia informatiilor 10
3.4 Viteza, politici de respingere si drepturi de autor 10
3.5 Paralelizare 11
3.6 Logarea actiunilor si retentia datelor 11

Crawling in cloud **13**
1 Infrastructura 15
2 Generarea infrastructurii 19
3 Executia procesului de crawling 22
3.1 Configurare 23
3.2 Initializare 25
3.3 Parcurgere 26
3.4 Finalizare 27
4 Interfata 29

Introducere

Un serviciu web reprezinta o componenta functionala ce indeplineste anumite sarcini. Comunicarea cu un serviciu web se realizeaza independent de platforma, limbajul de programare sau sistemul de operare pe care este dezvoltat. Schimbul de informatii se realizeaza prin mesaje text ce respecta un format standardizat precum xml[1] sau json[2].

Aplicatia "Surf" reprezinta un serviciu web specializat in web crawling[3], dezvoltat folosind tehnologii cloud din cadrul Amazon Web Services. Se urmareste crearea unui serviciu web cu disponibilitate permanenta, scalabil si de inalta putere computationala care sa orchestreze colectarea distribuita de informatii din aria definita de utilizator.

Comunicarea cu aplicatia "Surf" se realizeaza prin intermediul unui API RESTful[4] construit pe platforma AWS[5] API Gatway. Autentificarea utilizatorilor se va realiza printr-un serviciu OpenID Connect (e.g. Google, Facebook etc.). Autorizarea va avea ca principala componenta AWS IAM. Utilizatorilor le vor fi repartizate, in functie de privilegiile asociate cu cheia de autentificare, o serie de roluri (i.e. drepturi de access asupra resurselor din cadrul serviciului "Surf"). Executia codului propriu-zis, gazduit de functii AWS Lambda, va interactiona cu serviciul pentru baze de date no-sql AWS DynamoDB pentru a permite accesul la informatii cheie pentru functionalitatea aplicatiei (metadate crawling, date acces utilizatori etc.). Mediul de procesare distribuita va fi sustinut de AWS Simple Workflow Service si configurat dupa preferintele utilizatorului. Datele extrase din procesul de web-crawling vor fi salvate in mediul persistent de stocare AWS S3. Evenimentele legate de parcurgerea siturilor vor fi expuse, ca metadate, intr-o coada AWS SQS si vor fi accesibile utilizatorilor prin procesul de long-polling asupra acestei cozi.

[1] Extensible Markup Language - https://www.w3.org/XML/
[2] JavaScript Object Notation - http://www.json.org/
[3] Web crawler - https://www.techopedia.com/definition/10008/web-crawler
[4] REST - http://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm
[5] Amazon Web Services - https://aws.amazon.com

1 Descrierea unui crawler web

Natura dinamica World Wide Web-ului, precum si scara sa, justifica necesitatea existentei unor mecanisme eficiente de localizre a informatiilor relevante. Tool-urile specializare in cautarea datelor se bazeaza pe web crawler-e pentru colectarea paginilor web, ce sunt, ulterior, indexate si analizate.

Crawlerul ajuta utilizatorul in ceea ce priveste navigarea Internetului, automatizand parcurgerea link-urilor. Un crawler exploateaza structura web-ului [6] si extrage resurse, intr-o maniera ordonata, ghidat de criteriile specificate de catre utilizator. Rezultatul obtinut de pe urma utilizarii unui web crawler este o serie de situri web.

In documentele de referinta, web crawler-ii mai sunt cunoscuti si drept "wanderers", "robots", "spiders", "fish" sau "worms". Aceste denumiri nu sunt indicatori ai performantei, intrucat aceste programe pot fi foarte rapide si precise. Se pot parcurge zeci de mii de pagini intr-un interval de cateva minute, consumand doar o fractiune din lungimea de banda pusa la dispozitie [5], atat timp cat crawler-ul dispune de resursele hardware minime necesare.

Crawler-ii web deservesc unui numar mare de scopuri. Unul dintre acestea este mentenanta bazei de date in care sunt stocati indecsii unui motor de cautare. Pentru aceasta sarcina sunt utilizati crawleri de tip exhaustiv (se parcurg toate siturile web intalnite). O alta categorie este cea a crawler-ilor bazati pe euristici, folositi pentru adunarea datelor ce se inscriu sub o anumita tematica, minimizand efortul de a aloca resurse pentru pagini irelevante. In construirea unui astfel crawler este necesara definirea unei modalitati de cautare si selectie a datelor. In cazul crawlerilor bazati pe euristici, pot aparea probleme legate de explorare versus exploatare, deoarece spatiul de cautare ar putea contine un optim local, impiedicand algoritmul sa localizeze optimul global.[6]

2 Design-ul unui crawler web

Ca si design, un crawler web se rezuma la trei elemente: o *frontiera*, in care sunt stocate URL-urile ce urmeaza a fi vizitate, o componenta a carei scop este de a *downloada paginile* de pe World Wide Web si un spatiu de stocare, unde sa fie persistate datele preluate de la crawler (e.g. o baza de date si/sau un

[6] Web-ul este structurat asemeni unui graf

sistem de fisiere) [4, 5].

2.1 Frontiera

In terminologia grafurilor, frontiera reprezinta lista nodurilor nevizitate [7]. Pentru un crawler web, frontiera reprezinta o lista de likuri neparcurse. Procesul de cautare porneste de la un link numit radacina, care este preluat din web si procesat, uramand ca link-urile gasite sa fie adaugate in frontiera. Aceasta lista poate capata dimensiuni foarte mari inca de la inceputtul procesului de parcurgere a paginilor, intrucat media de legaturi de pe o pagina web este 7 link-uri/pagina [8]. Procesul se repeta, alegandu-se, folosind un algoritm, link-uri din frontiera, pana cand frontiera se goleste sau alta conditie de oprire permite finalizarea cautarii.

2.2 Descarcarea paginilor web

Este necesar un client HTTP care trimite o cerere HTTP si primeste un raspuns. In aceasta etapa, nu trebuie trecut cu vederea "The Robots Exclusion Standard", cunoscul si drept "Robots Exclusion Standard" sau robots.txt [8]. Protocolul ofera administratorilor de servere web posibilitatea de a-si comunica politicile de acces si de a mentiona paginile care nu sunt destinate crawler-ilor.

2.3 Stocarea datelor

In urma parcurgerii paginilor web, datele obtinute sunt stocate intr-un mediu potrivit nivelului lor de abstractizare. Spre exemplu, metadatele vor fi stocate intr-un mediu in care interogarea sa se poata realiza in mod eficient, iar paginile in format HTML vor fi persistate intr-un mediu de stocare mai incapator, dar cu viteza de acces redusa.

[7] "[...] unexpanded (unvisited) nodes." [6]

[8] "The web may be viewed as a directed graph in which each vertex is a static HTML web page, and each edge is a hyperlink from one web page to another. Current estimates suggest that this graph has roughly a billion vertices, and an average degree of about 7."[7]

3 Crawler-ul Surf

Ca sens general, un crawler web reprezinta un program care parcurge, prin cereri succesive, situri web. Programatic, vom considera urmatoarele aspecte cheie in proiectarea unui crawler distribuit:

1. Punctele de pornire in parcurgerea siturilor web;
2. Adancimea maxima a parcurgerii recursive a siturilor (i.e. cea mai indepartata pagina la care se poate ajunge, de la punctul de pornire, prin accesarea succesiva a legaturilor de tipul hyperlink;
3. Politica de selectie a informatiilor din paginile parcurse[9];
4. Viteza de crawling, fisierul *robots.txt*[9] si respectarea drepturilor de autor;
5. Politica de paralelizare a procesului de web crawling;
6. Politica de retentie temporara a rezultatelor parcurgerii siturilor web si logare a actiunilor serviciului de crawling;

In cele ce urmeaza, se va descrie particularizarea aspectelor generale enumerate mai sus in cadrul serviciului web "Surf". Se va crea, astfel, contextul dezvoltarii aplicatiei si se vor puncta principalele componente functionale implicate.

3.1 Puncte de pornire

Pentru a putea parcurge siturile web, crawler-ul "Surf" necesita unul sau mai multe puncte de pornire. Un punct de pornire este definit printr-un URL[10] si este furnizat, ca input din partea utilizatorului, la initializarea crawler-ului.

3.2 Parcurgere

Crawler-ul web "Surf" are ca scop extragerea informatiilor cerute de catre utilizator. Intrucat utilizatorul are posibilitatea de a furniza, ca punct de pornire, un domeniu web relevant pentru informatia cautata, parcurgerea recursiva se va executa in maniera breadth-first. Asadar, crawler-ul va vizita toate legaturile de tip hyperlink din pagina curenta a parcurgerii inainte de a accesa legaturile din pagina urmatoare din punct de vedere ierarhic. Adancimea maxima a arborelui de legaturi realizat prin parcurgerea URL-urilor va fi definita de catre utilizator, la initializarea sesiunii de crawling.

[9] http://www.robotstxt.org/robotstxt.html
[10] Uniform Resource Locator - http://www.dictionary.com/browse/url

3.3 Selectia informatiilor

Selectia informatiilor necesita parcurgerea siturilor web aflate la adresele URL pe care crawler-ul le are in vedere, parsarea informatiilor in functie de tipul lor (e.g. html, xml, json, text) si extragerea blocurilor de continut aferente.

Un "bloc de continut" asociat unui URL reprezinta o parte a intregului continut aflat la URL-ul respectiv, filtrata dupa anumite caracteristici stabilite de utilizatorul serviciului web. Aceste caracteristici pot fi:

- Pentru fisiere HTML/XML:
  - selectori CSS/jQuery
- Pentru fisiere text:
  - cuvinte cheie

Pentru tipurile de continut enumerate mai sus sau pentru alte tipuri de continut aflate la adresele URL vizitate de crawler, utilizatorul poate defini filtre bazate pe continutul textual al URL-ului. Spre exemplu, se pot evita toate URL-urile care nu satisfac o anumita expresie regulata.

Evitarea selectarii informatiilor duplicate se amelioreaza prin normalizarea[11] URL-urilor parcurse de catre crawler.

3.4 Viteza, politici de respingere si drepturi de autor

Siturile web pot implementa variate modalitati de contracarare a incercarilor de crawling. Aplicatia "Surf" incearca sa minimizeze riscul de respingere a cererilor de accesare a anumitor resurse web printr-o implementare neintruziva a procesului de crawling. Cateva aspecte esentiale care sunt luate in considerare in ceea ce priveste o astfel de implementare sunt urmatoarele:

- Minimizarea volumului de date preluat de pe un anumit domeniu prin diferite metode de filtrare a linkurilor urmarite (e.g. o anumita structura a URL-ului, un anumit tip de date care se gaseste la URL-ul respectiv);

Crawler-ul web "Surf" poate satisface numeroase cerinte ale utilizatorilor. O parte dintre aceste cerinte poate veni din partea sistemelor anti-malware. In acest caz, nu se doreste respectarea fisierului *robots.txt* (utilizat drept referinta,

[11] https://tools.ietf.org/html/rfc3986#section-6

pentru crawleri, asupra URL-urilor accesibile ale domeniului vizitat), deoarece exista pericolul ca un sit malitios sa blocheze o eventuala scanare. De aceea, aplicatia "Surf" va putea fi configurata in ceea ce priveste ignorarea fisierului *robots.txt* in procesul de parcurgere a unui domeniu.

Crawler-ul "Surf" implementeaza un mecanism de *blacklisting*[12]. Siturile web sau domeniile care interzic procesul de crawling (e.g. prin ToS[13]) vor fi adaugate unei liste de excluziune din procesul de parcurgere executat de crawler.

3.5 Paralelizare

"Surf" implementeaza un mecanism de crawling distribuit. Mai exact, exista posibilitatea de a imparti sarcinile de parcurgere a paginilor web prin rularea concurenta a mai multor instante de functii Lambda[14].

Exista doua moduri bine cunoscute de alocare a sarcinilor de web crawling: statica si dinamica[10]. Aplicatia "Surf" implementeaza un mecanism de distribuire statica a sarcinilor:

> Fie n, *gradul de paralelizare*[15] configurat pentru executia curenta a web crawler-ului. Alocarea statica va asocia fiecarui URL candidat pentru crawling un numar intreg in intervalul $[1,n]$ reprezentand identificatorul uneia dintre cele n instante paralele ale web crawler-ului. Sarcina va fi distribuita pe acea instanta si va fi executata.

3.6 Logarea actiunilor si retentia datelor

In timpul executiei, crawler-ul genereaza evenimente ce sunt adaugate la istoricul executiei curente. Aceste evenimente sunt utile, in primul rand, in cazul in care executia instantei curente a crawler-ului esueaza. Sarcinile de crawling web pot dura o perioada considerabila si nu este dezirabila refacerea intregului proces de parcurgere a siturilor, atat timp cat exista rezultate partiale; executia se poate relua pornind de la ultimul eveniment valid inregistrat in istoric. In al doilea rand, istoricul este util pentru utilizator, deoarece il poate avertiza in legatura cu statutul executiei curente a web crawler-ului.

[12] http://dictionary.cambridge.org/dictionary/english/blacklist
[13] http://www.pcmag.com/encyclopedia/term/62682/terms-of-service
[14] Amazon Web Services Lambda Functions
[15] Numarul maxim de executii simultane de sarcini de crawling

Datele rezultate in urma procesului de crawling al paginilor web sunt salvate in serviciul persistent de stocare S3, din cadrul Amazon Web Services. Deoarece aceste date pot deveni volumnioase (in functie de politica de selectie configurata), aplicatia "Surf" salveaza metadate asociate acestor rezultate intr-o tabela DynamoDB. Pe baza acestor informatii (metadate), utilizatorul are optiunea de a accesa rezultatele complete corespunzatoare rularii crawler-ului web, aflate in S3.

Crawling in cloud

Aplicatia "Surf" este construita si ruleaza in cloud. Dezvoltarea unei aplicatii pe suportul unei infrastructuri cloud este substantial diferita fata de abordarea clasica, pe o singura masina de calcul. Cloud computing-ul pune la dispozitie, la cerere, un set de resurse (e.g. putere de procesare, spatiu de stocare) pe care un utilizator (e.g. aplicatia "Surf") le poate folosi. Faptul ca toate aceste resurse sunt administrate de fiecare serviciu cloud in parte usureaza sarcinile consumatorului: se pot solicta mai multe sau mai putine dupa gradul de incarcare al aplicatiei dezvoltate in cloud. Acest lucru se realizeaza intr-o maniera transparenta pentru utilizator, fara a necesita efort suplimentar din partea acestuia.

Dezvoltarea aplicatiei "Surf" folosind servicii cloud mai prezinta un avantaj considerabil, deoarece comunicarea prin internet intre componentele aplicatiei si, implicit, utilizarea unui format agnostic de arhitectura sau limbaj de programare, conduce la un grad mare de decuplare. Fiind decuplate, componentele aplicatiei pot fi construite, testate si depanate[16] in izolare si pot functiona ca module (plugin-uri) in alcatuirea arhitecturii sistemului.

A dezvolta un serviciu web folosind exclusiv tehnologii cloud permite realizarea unui sistem de subscriptii granular. Un utilizator al aplicatiei "Surf" poate solicita una sau mai multe instante ale crawler-ului care sa ruleze in paralel, in functie de necesitatile proprii. Serviciile cloud permit aplicatiei sa scaleze[17] orizontal ca timp, permitand utilizatorului realizarea mai multor sarcini de parcurugere automata web (deci, in consecinta, acoperirea unui volum mai mare de informatii) in acelasi interval de timp.

Un alt aspect relevant in realizarea serviciului web "Surf" este caracterul sau *serverless*. Infrastructura *serverless* implica executia codului sursa al programului in cadrul serviciilor cloud folosite pentru dezvoltarea aplicatiei. Cererile

[16] *engl.* debugged
[17] http://www.dictionary.com/browse/scalability

de executie a codului sunt monetizate conform unei masuri abstracte corespunzatoare resurselor utilizate pentru satisfacerea solicitarii, ceea ce difera fata de modelul clasic, in care ar fi trebuit achizitionat hardware in acest scop[11].

1 Infrastructura

Esenta aplicatiei "Surf" se afla in codul executat de serviciul web AWS Lambda, care preia componentele dezvoltate pentru crawling web si le executa drept programe autonome folosind infrastructura AWS. AWS Lambda executa acest cod doar cand este nevoie (e.g. la cererea utilizatorului care doreste sa demareze operatiunea de crawling). Se minimizeaza, astfel, atat costurile utilizatorului in ceea ce priveste rularea crawler-ului web (nu necesita hardware dedicat pentru procesarea paginilor web), cat si costurile dezvoltatorilor crawler-ului, care nu trebuie sa rezerve hardware in cloud (e.g. masini virtuale Amazon EC2, inchiriere de hosturi), ca mai apoi, cand nu exista cereri suficiente, aceste masini de calcul sa ramana nefolosite. De asemenea, se renunta la necesitatea de a avea un load-balancer care sa distribuie sarcinile de crawling pe capacitatea hardware disponibila, deoarece acest lucru este administrat, in fundal, de catre AWS Lambda.

Functiile programatice disponibile prin serviciul AWS Lambda pot fi accesate de catre utilizatori printr-un API Restful gazduit de AWS API Gateway. Aceasta platforma permite integrarea cererilor HTTP externe cu mecanismul de autentificare si autorizare a utilizatorilor si codul executat de instantele functiilor Lambda. De asemenea, aplicatia utilizeaza functionalitati de management, monitorizare si analiza a traficului din cadrul API-ului "Surf" care permit, printre altele, inregistrarea si monetizarea functionalitatilor oferite de serviciul web pentru fiecare utilizator in parte.

Pentru a putea accesa functionalitatile oferite de API-ul "Surf", utilizatorii trebuie sa se autentifice cu un furnizor tert de identitati ce suporta OpenID[18]. Credentialele obtinute sunt impachetate intr-o cerere catre AWS Security Token Service[19], unde sunt validate. In cazul in care validarea este indeplinita cu succes, Security Token Service returneaza utilizatorului credentiale temporare pentru a accesa servicii din cadrul AWS. Utilizatorul trebuie sa includa aceste credentiale in fiecare cerere catre API-ul "Surf" (autentificare). Odata autentificat, utilizatorul poate interactiona doar cu acele resurse ale API-ului pentru care are drepturi de acces.

[18] http://openid.net/what-is-openid/

[19] Serviciu web ce permite unui utilizator sa solicite credentiale temporare pentru autentificarea in cadrul platformei Amazon Web Services

Figura 1: Procesul de autentificare [12]

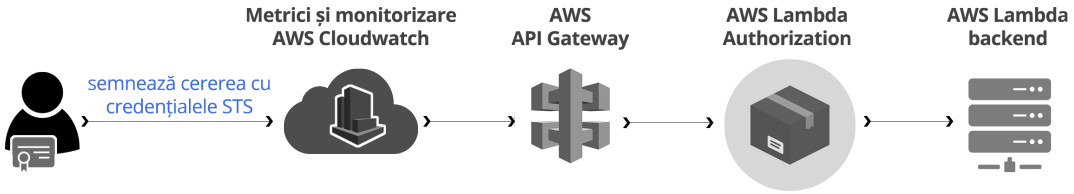

Figura 2: Procesul de autorizare [12]

Prin AWS Identity Access Management (IAM) se vor stabili restrictiile de acces asupra API-ului. Mecanismul de asignare a permisiunilor va fi sustinut prin definirea unor roluri. Utilizatorii vor fi grupati, din punct de vedere al drepturilor de acces, in mai multe categorii. Spre exemplu, un utilizator poate avea rolul "administrator" (prescurtat Ad), iar un alt utilizator poate avea atat rolul "administrator" (Ad) cat si "utilizator de baza" (prescurtat Ub). Atat Ad, cat si Ub vor avea asignate politici de acces asupra datelor AWS. O astfel de politica este urmatoarea:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Figura 3: Politica de acces IAM

Politica de acces de mai sus ii permite utilizatorului ce ii este asignata accesul la toate resursele API-ului "Surf". Aceasta politica este potrivita pentru un administrator dar foarte periculoasa pentru un utilizator ce acceseaza pentru prima data serviciul "Surf". De aceea, se vor defini drepturi de acces granulare care sa aiba in vedere restrictionarea la nivel "need-to-know"[20] pentru fiecare entitate ce trimite cereri catre API (autorizare).

Procesul de asignare a rolurilor pentru utilizatori este coordonat atat manual (pe baza de ierarhie: administrator - utilziator premium - utilizator standard) cat si automat (i.e. printre altele, toti utilizatorii abia inregistrati sunt utilizatori standard). Asignarea manuala are prioritate asupra asignarii automate. Datele despre rolul fiecarui utilizator vor fi pastrate in baza de date.

Crawler-ul "Surf" utilizeaza AWS DynamoDB ca suport pentru baza de date. DynamoDB reprezinta un serviciu cloud scalabil pentru baze de date no-sql. Datele dintr-o baza de date non-relationala (no-sql) pot fi modelate fara constrangerile unei baze de date relationale (e.g. tabularitate). Acest lucru atrage, in cadrul aplicatiei "Surf", elemente operationale cheie pentru beneficiul carora s-a renuntat la o abordare sql (e.g. Oracle), printre care:

[20] http://dictionary.cambridge.org/dictionary/english/on-a-need-to-know-basis

- *data sharding*[21] - este necesar un sistem distribuit de baze de date care sa poata scala orizontal odata cu dimensiunea datelor din baza de date si odata cu cresterea numarului de utilizatori;

- *scheme de date dinamice*[22] - se doreste posibilitatea schimbarii structurii datelor din baza de date fara a recrea tabelele corespunzatoare; acest lucru trebuie avut in vedere deoarece crawler-ul stocheaza in baza de date metadate obtinute prin parcurgerea paginilor web; poate aparea oricand necesitatea introducerii unor noi metadate sau schimbarii structurii celor existente, cu scopul satisfacerii nevoilor utilizatorilor.

Cateva dintre cele mai importante roluri ale bazei de date sunt gazduirea metadatelor asociate rezultatelor procesuli de crawling, persistarea asignarilor intre identitatile utilizatorilor si rolurile lor, mentinerea istoricului evenimentelor de crawling cu scopul reconstruirii starii procesului de parcurgere a paginilor web in cazul unei erori si stocarea datelor asociate volumului de trafic in cadrul infrastructurii AWS generat de fiecare utilizator.

Pentru a obtine un serviciu web distribuit pentru crawling este necesara coordonarea activitatilor independente din cadrul sistemului, sincronizarea pasilor necesari procesului de crawling si, in final, integrarea rezultatelor. Pentru acest lucru aplicatia "Surf" foloseste serviciul web AWS Step Functions (SFN). SFN are capacitati de coordonare a activitatilor ce se doresc indeplinite (i.e. executia functiilor Lambda, sau *Lambda Tasks*) si management al starii aplicatiei (i.e. se porneste un proces de agregare a datelor de la crawleri care au rulat in paralel doar dupa ce acestia si-au terminat executia).

Datele obtinute in urma procesului de crawling sunt stocate utilizand serviciul web Amazon S3. Fiecare rulare a unei instante a crawler-ului distribuit genereaza un *bucket*[23] S3. Datele din bucket-uri sunt disponibile, pentru utilizatori, prin plasarea de metadate precum numele bucket-ului intr-o coada Amazon SQS, asupra careia se executa un mecanism de long-polling pentru extragerea informatiilor. Datele din bucket-uri au un timp limitat de viata, configurabil relativ la preferintele utilizatorului. O functie Lambda, programata sa se execute, periodic, prin serviciul AWS CloudWatch, elimina bucket-urile a caror durata de viata a depasit termenul limita stabilit.

[21] https://docs.microsoft.com/en-us/azure/architecture/patterns/sharding

[22] https://www.mongodb.com/scale/dynamic-schema-design

[23] Unitate in cloud (AWS S3) ce stocheaza date si careia ii pot fi atribuite permisiuni de acces si metadate corespunzatoare

2 Generarea infrastructurii

Procesul de generare a infrastructurii (engl. 'deployment') consta in crearea resurselor si pornirea sistemelor necesare pentru ca web-crawler-ul 'Surf' sa functioneze. Mecanismul de generare a resurselor executa urmatorii pasi:

- Crearea entitatilor necesare in AWS (e.g. functii Lambda, roluri IAM, API-ul APIGateway etc.);
- Stabilirea relatiilor intre resurse si injectarea dependentelor (e.g. unei rute a API-ului trebuie sa ii fie asociat un rol IAM care sa ii permita invocarea unei functii Lambda);
- Generarea SDK-ului Javascript necesar pentru a putea invoca API-ul 'Surf' din cadrul clientului web;
- Generarea unui fisier de configurare injectabil in clientul web 'Surf' pentru a stabili parametrii interactiunii intre client si cloud-ul AWS (e.g. regiunea AWS in care s-a generat infrastructura, metadate legate de rolurile utilizatorilor, cheia generata pentru a restrictiona accesul asupra API-ului etc.).

Efortul necesar pentru generarea infrastructurii este mare, datorita complexitatii inerente a proiectului. Efectuarea manuala a pasilor in interfata web AWS (engl. 'AWS Web Console') necesita foarte mult timp si este predispusa la erori. De asemenea, asigurarea disponibilitatii crawler-ului in mai multe regiuni globale AWS sau pe mai multe conturi AWS ar necesita repetarea identica a pasilor enumerati mai sus, pentru fiecare astfel de regiune sau cont. De aceea, crawler-ul web 'Surf' pune la dispozitie o modalitate automatizata de deployment, configurabila, testabila si reutilizabila, asigurand idempotenta[24] la nivelul efectuarii operatiilor in cadrul AWS.

Mecanismul automatizat de generare a infrastructurii AWS reprezinta un program care primeste, ca date de intrare, un fisier de configurare in format JSON, genereaza resursele AWS si ofera clientului web, ca date de iesire, prin injectarea dependentelor, informatii si mecanisme pentru utilizarea infrastructurii create. Cerintele pentru executia cu success a deployment-ului, precum si validitatea resurselor generate reprezinta existenta credentialelor AWS necesare

[24]Termenul 'idempotenta' este folosit pentru a evidentia faptul ca o parte dintre operatiile efectuate de catre mecanismul automatizat de generare a infrastructurii vor avea acelasi rezultat daca vor fi apelate de mai multe ori. Din motive de securitate, unele operatii vor fi definite explicit ca nefiind idempotente (e.g. generarea de chei de acces pentru API-ul 'Surf', generarea adresei de acces a API-ului (nu va fi suprascrisa in API-ul existent))

pentru pasii de deployment (e.g. IAM 'administrator-access') si generarea pachetului (.jar) care sa contina codul functiilor Lambda care se doresc a fi incarcate in AWS. Mai jos se poate observa un exemplu de fisier de configurare pentru generatorul de resurse AWS 'Surf':

```
{
  "awsAccountId": "011759591962",
  "awsAccessKey": "#######" // Obfuscat intentionat,
  "awsClientRegion": "eu-west-1",
  "lambdaCodePath": "../lambda/target/surf-lambda-backend-1.0-SNAPSHOT.jar",
  "lambdaRuntime": "java8",
  "apiGatewayEndpoint": "apigateway.amazonaws.com",
  "apiStageName": "v1",
  "apiStageMetricsEnabled": true,
  "apiStageThrottlingRateLimit": "5",
  "apiStageThrottlingBurstLimit": "20",
  "apiLoggingLevel": "INFO",
  "apiGeneratedSdkType": "javascript",
  "apiGeneratedSdkOutputPath": "../../frontend/generated/sdks/",
  "apiGeneratedSdkFolderName": "api-gateway-js-sdk",
  "clientConfigFilePath": "../../frontend/generated/config/aws-config.json",
  "lambdaConfigFilePath": "../lambda/generated/config/lambda-config.json",
  "dynamoDBWorkflowsTableReadCapacityUnits": "2",
  "dynamoDBWorkflowsTableWriteCapacityUnits": "2"
}
```

Figura 4: Fisier de configurare (intrare) pentru generatorul de resurse

Pentru a asigura conexiunea clientului web cu infrastructura creata, generatorul de resurse AWS injecteaza un fisier de configurare in clientul web, intr-un director special destinat acestui sens. Politicile de securitate implementate in generatorul de resurse nu vor permite suprascrierea fisierului/directorului din clientul web in cazul in care acesta exista deja, pentru a evita pierderea datelor. Un astfel de fisier de confiurare este cel prezentat in *Figura 5*:

Diagrama din *Figura 6* prezinta procesul de construire a infrastructurii, ordonand temporal pasii necesari pentru crearea si configurarea crawler-ului. Inainte de inceperea deployment-ului resurselor AWS, se genereaza o arhiva ce contine codul sursa al functiilor Lambda. Procesul incepe cu citirea fisierului de configurare si se incheie cu generarea API-ului prin care utilizatorii vor putea accesa serviciul web. Acolo unde este necesar, se specifica interdependentele

```
{
  "awsClientRegion": "eu-west-1",
  "awsAccessKey": "#######",
  "facebookWebIdentityBasicRoleArn": "arn:aws:iam::011759591962:role/fb-role",
  "apiKey": "#######"
}
```

Figura 5: Fisier folosit pentru configurarea clientului

intre sisteme. De exemplu, este necesar ca mai intai sa fie create functiile Lambda, inainte ca acestea sa fie inregistrate in cadrul mecanismului de notificare asincrona oferit de SNS (de aici si numarul 4 asociat generarii functiilor lambda, reprezentand o prioritate mai mare decat numarul 5 asociat serviciului SNS). Dupa finalizarea procesului de generare a infrastructurii, pasii 1 si 4 se vor executa inca odata, in contextul executiei precedente (i.e. avand la dispozitie toate referintele catre resursele create), pentru a actualiza codul functiilor Lambda in vederea accesarii resurselor mentionate.

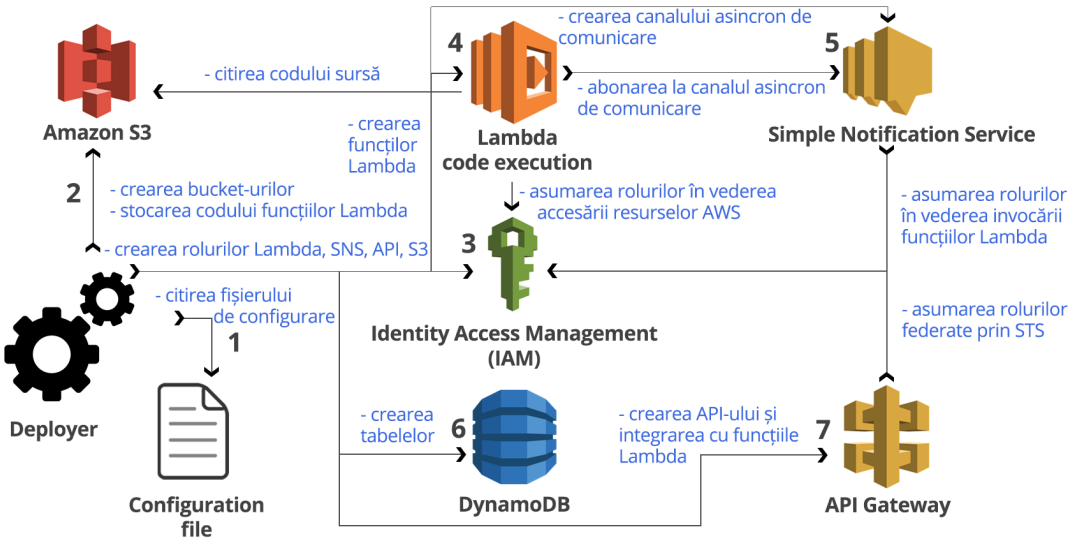

Figura 6: Generarea infrastructurii [12][13]

3 Executia procesului de crawling

Aceasta sectiune prezinta, in detaliu, modul in care aplicatia "Surf" indeplineste procesul de parcurgere a siturilor web. Mai exact, se descriu mecanismele logice prin care, avand la dispozitie infrastructura creata cu ajutorul generatorului de resurse, utilizatorul poate beneficia de rezultatele extrase din paginile web parcurse de catre crawler. Pentru simplitate, ne vom referi la ansamblul acestor mecanisme logice drept un *workflow*[25].

Un *workflow* reprezinta unitatea de baza pe care crawler-ul web "Surf" o poate interpreta. Dintr-un *workflow* se pot genera oricate executii. Vom numi o executie a unui *workflow* drept un *workflow execution*. Inceperea executiei unui *workflow* trebuie sa fie precedata de crearea acestuia. O executie a unui *workflow* intreprinde urmatoarele actiuni, cu scopul parcurgerii paginilor web pentru extragerea informatiilor:

1. Preluarea, validarea si interpretarea datelor de configurare a procesului de crawling;
2. Stocarea persistenta a definitiei unei executii a unui *workflow* in baza de date cu scopul accesarii datelor necesare in contextul rularii crawler-ului;
3. Initializarea unui automat finit si determinist in vederea modelarii parcurgerii recursive a siturilor web in maniera *breadth-first*[26], prin "AWS Step Functions";
4. Crearea sarcinilor pentru parcurgerea paginilor web (*taskuri*[27]), inregistrarea datelor si metadatelor rezultate in urma parcurgerii, precum si a datelor ce modeleaza executia procesului de crawling (e.g. timpi maximi de raspus, adrese URL vizitate deja);
5. Distribuirea *taskurilor* catre functiile Lambda (*workers*[28]) ce le vor prelua si executa;

[25]Din engl. "workflow", care se traduce, in acest context, drept un flux de lucru, sau un proces in cadrul caruia se desfasoara o serie bine definita de activitati (i.e. executii de programe)

[26]Parcurgea in maniera *breadth-first* asigura vizitarea fiecarui nod dintr-un arbore de la adancimea curenta, inainte de a trece la parcurgerea nodurilor de la urmatorul nivel de adancime

[27]In contextul executiei unui *workflow*, un *task* reprezinta o unitate de lucru sub forma unei inregistrari in baza de date, ce poate fi preluata si executata de catre un program aflat in executie (e.g. o functie Lambda)

[28]Un *worker* reprezinta un program executabil. In aplicatia "Surf", functiile Lambda reprezinta *worker-ii*

6. Executarea functiilor Lambda si stocarea/intoarcerea rezultatelor;

7. Sincronizarea executiilor paralele ale functiilor Lambda si centralizarea rezultatelor in urma executiei *task-urilor*;

8. Pornirea recursiva a unui nou *workflow* cu scopul de a parcurge un nou nivel de adancime in procesul de crawling si de a gestiona gradul de incarcare a sistemului distribuit (e.g. limitarile impuse de serviciile AWS precum numarul maxim de executii concurente a functiilor Lambda in cadrul executiei unui automat "AWS Step Functions").

3.1 Configurare

Pentru inceperea executiei unui workflow, este necesara, in prealabil, crearea workflow-ului. *Tabelul 1* de la pagina 22 descrie datele necesare crearii unui workflow, date dintre care o parte sunt configurabile de catre utilizator (i.e. campul "Config."[29] are valoarea *DA*), iar o parte sunt memorate in mod implicit odata ce workflow-ul este creat. Suita de configurari descrisa in *Tabelul 1* coexista cu seria de configurari generate de creatorul de resurse. Diferenta intre cele doua consta in faptul ca generatorul de resurse defineste configurari la nivel de cont AWS. Impartirea configurarilor in doua de nivele de abstractizare (i.e. scazut pentru cele de la nivelul resurselor din contul AWS si ridicat pentru cele oferite in etapa de creare a unui workflow) confera flexibilitate in utilizarea crawler-ului "Surf". Pentru a evidentia contrastul intre cele doua configurari vom enumera, in continuare, cateva elemente esentiale ale configurarii la nivel de resurse din contul AWS:

- timpul maxim de asteptare pentru ca o cerere cloud sa fie satisfacuta;
- regiunea in care sunt create resursele AWS;
- numarul maxim de cereri pe secunda admis de catre API;
- numarul maxim de cereri pe secunda admise pentru tabelele din baza de date (configurare importanta pentru controlul costului);
- stabilirea nivelului de logare (i.e. urmarire a actiunilor utilizatorilor in cadrul interactiunii cu API-ul "Surf);
- stabilirea limbajului in care va fi generat clientul pentru API-ul creat prin API Gateway.

[29] prescurtare de la substantivul *Configurabil*

Nume camp	Tip	Config.	Descriere
Id	Text	NU	Id atribuit workflow-ului pentru a putea fi identificat in mod unic printre celelalte workflow-uri din tabela (din baza de date) care le gazduieste
Data Creare	Numeric	NU	Data crearii, reprezentand numarul de milisecunde de la 1970 (engl. "epoch milliseconds")
Proprietar	Text	NU	Identitatea atribuita utilizatorului ce a creat workflow-ul. Necesar pentru a stabili permisiuni de acces.
Nume	Text	DA	Nume familiar atribuit workflow-ului pentru identificare facila
Adresa Inceput	Text	DA	URL-ul de la care va porni procesul de crawling
Selector URL	Text	DA	Expresie regulata ce valideaza daca un URL de pe o pagina vizitata de catre crawler este adaugat recursiv la lista sarcinilor pentru crawling
Adancime maxima	Numeric	DA	Adancimea maxima la care vor fi parcurse paginile web de catre crawler (i.e. in arborele de parcurgere recursiva)
Pagini pe nivel	Numeric	DA	Numarul maxim de pagini ce vor fi parcurse de catre crawler pe un anumit nivel de adancime (i.e. in arborele de parcurgere recursiva)
Dimensiune maxima pagina	Numeric	DA	Dimensiunea maxima admisa (in octeti) pentru un sit web sa fie parcurs
Grad de paralelizare	Numeric	DA	Numarul maxim de crawleri concurenti dintr-un workflow
Politica de selectie	JSON	DA	Definitie folosita pentru a extrage date din paginile web parcurse, in functie de anumiti parametri
Politica de reincercare	JSON	DA	Definitie folosita pentru a stabili cazurile in care incercarile esuate de a parcurge paginile web trebuie reincercate

Tabelul 1: Definitia unui workflow

3.2 Initializare

Odata ce o cerere de incepere a unei executii a unui workflow este primita de catre crawler, functia Lambda corespunzatoare inceperii executiei (i.e. *Pornire workflow*) este invocata. Aceasta creaza si salveaza metadatele necesare in baza de date (e.g. timpul la care a inceput workflow-ul, cu scopul de a putea contoriza cat a durat executia), apoi invoca functia Lambda responsabila pentru initializarea mecanismului de distributie si sincronizare a executiei in paralel a functiilor ce parcurg paginile web. *Figura 7* descrie modul in care functia de initializare a unui workflow este utilizata, precum si rolul pe care aceasta il indeplineste in procesul recursiv de crawling.

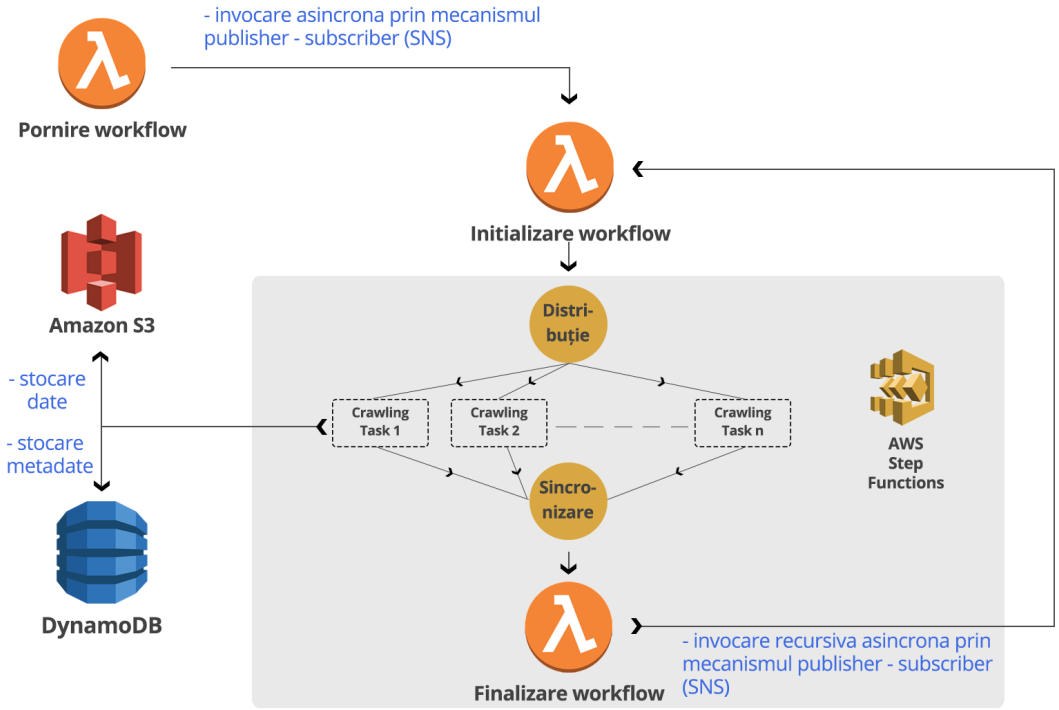

Figura 7: Arhitectura unui workflow [12, 13]

Responsabilitatile functiei de initializare a workflow-ului sunt urmatoarele:

- Verificarea nivelului de adancime la care s-a ajung in urma procesului de crawling si finalizarea workflow-ului, in caz ca nivelul de adancime maxim a fost depasit;
- Interogatea bazei de date in vederea extragerii task-urilor pentru parcurgerea paginilor web, task-uri ce sunt executate in cadrul automatului orchestrat de catre "AWS Step Functions";

- Crearea si pornirea automatului finit construit in cadrul "AWS Step Functions", pe baza task-urilor extrase din baza de date. Acest automat este definit prin doua stagii secventiale. Primul stagiu are ca scop orchestrarea distribuita a sarcinilor de crawling executate in paralel, iar cel de-al doilea are ca scop agregarea, interpretarea si salvarea datelor provenite din executia task-urilor concurente.

3.3 Parcurgere

Functiile ce au ca scop vizitarea paginilor web sunt grupate ca stagii de executii paralele in automatul finit configurat in "AWS Step Functions". Acestea reprezinta executii ale definitiilor formale ale task-urilor de crawling. O parte dintre datele asociate unui task reprezinta metadatele configurate in pasul de creare a workflow-ului. *Tabelul 2* prezinta structura unui task, cu mentiunile necesare acolo unde definitia unui camp este exact aceeasi cu definitia campului respectiv din structura unui workflow.

Nume camp	Tip	Descriere
Id	Text	Identificator unic pentru task
Id Workflow	Text	Identificatorul workflow-ului de care apartine task-ul curent
Data Incepere	Numeric	Data la care a inceput executia task-ului (engl. epoch milliseconds)
Data Finalizare	Numeric	Data la care s-a finalizat executia task-ului(engl. epoch milliseconds)
Stare	Text	Starea in care se afla task-ul: *Programat, Activ, Esuat, Depasit temporal, Intrerupt*
Adresa	Text	URL-ul ce trebuie vizitat in vederea extragerii datelor si metadatelor
Adancime	Numeric	Nivelul de adancime, in arborele de parcurgere a paginilor web realizat de catre crawler, la care se afla task-ul
Erori	Lista	Esecurile inregistrate in decursul executiei task-ului (ca structuri JSON)
Politica de selectie	JSON	*vezi Tabelul 1, pag. 22*
Dimensiune maxima pagina	Numeric	*vezi Tabelul 1, pag. 22*
Selector URL	Text	*vezi Tabelul 1, pag. 22*

Procesul de parcurgere a unei pagini web presupune urmatoarele etape:

1. Actualizarea starii task-ului in vederea reflectarii conditiei sale actuale (e.g. "in executie" sau "esuat");

2. Verificarea dreptului de acces asupra URL-ului configurat in cadrul definitiei task-ului, prin consultarea fisierului robots.txt (daca acesta exista), aflat in calea de baza (engl. *root*) a sitului web de la URL-ul respectiv, in concordanta cu politica de configurare a procesului de crawling;

3. Parcurgerea sitului web:

   (a) Extragerea datelor conform politicii de selectie a datelor (configurata in pasul de creare a workflow-ului) si stocarea acestora (daca exista) in S3;

   (b) Salvarea metadatelor in legatura cu datale extrase, in vederea accesarii sau cautarii facile a locatiei in care au fost salvate in S3;

   (c) Selectarea tuturor legaturilor de tip URL catre alte situri web din cadrul paginii vizitate, conform politicii de urmarire a legaturilor catre alte pagini web, definita in pasul de creare a workflow-ului;

   (d) Crearea de noi task-uri in starea *Programat*, configurate cu un nivel de adancime superior celui pe care se afla URL-ul curent parcurs de catre crawler si salvarea acestora in baza de date, doar daca respecta configurarile declarate ca date de intrare ale executiei workflow-ului (e.g. dimensiunea maxima a unei pagini web parcurse); acest pas mai poarta denumirea de *task scheduling*[30];

4. Actualizarea tabelei in care se tine evidenta paginilor parcurse de catre crawler in timpul executiei unui workflow, cu scopul de a nu se parcurge de doua ori acelasi URL;

5. Prinderea, rezolvarea si stocarea erorilor survenite in cadrul procesului de parcurgere a paginilor web.

3.4 Finalizare

Procesul de finalizare a unei sesiuni de crawling are drept obiective centralizarea si analizarea datelor provenite in urma executarii, in paralel, a functiilor responsabile pentru parcurgerea paginilor web. Responsabilitatile functiei de finalizare se pot imparti in doua categorii, in functie de motivul ce a determinat invocarea acesteia:

[30]engl. *task scheduling* = programare temporala a executiei unui task

1. Finalizarea unei executii indeplinite cu succes, caz in care nu au intervenit erori extraordinare (i.e. netratate) in cadrul cel putin unuia dintre task-urile executate in paralel;

2. Finalizarea unei executii in care au existat erori in cadrul executiei cel putin unuia dintre task-urile executate in paralel, caz in care functia de finalizare va indeplini toate sarcinile ce au ramas neindeplinite in urma terminarii bruste a executiei sarcinilor paralele de crawling.

Indiferent de situatia in care se afla executia worfklow-ului, functia de finalizare a sesiunii de crawling are in vedere inregistrarea metadatelor cu privire la progresul workflow-ului (e.g. metrici Cloudwatch) si luarea deciziei conform careia invocarea recursiva a urmatoarei sesiuni de crawling se face incepand cu acelasi nivel de adancime sau cu unul superior (i.e. mai mare), in concordanta cu numarul maxim de pagini ce poate fi vizitat pe un anumit nivel de adancime, de catre crawler).

4 Interfata

Modalitatea prin care componentele crawler-ului web "Surf" pot fi accesate, modificate si executate este reprezentata de o interfata[14] structurata sub forma unui API RESTful[31]. API-ul adopta stilul arhitectural REST[15] deoarece acesta ajuta la decuplarea arhitecturii si simplificarea urmaririi executiei aplicatiei prin faptul ca nu permite memorarea starilor intermediare in cadrul unei sesiuni de comunicare intre utilizator si crawler. De asemenea, API-ul RESTful confera extensibilitate si flexibilitate la nivel de interfata, intrucat permite gruparea resurselor in mod ierarhic si identificarea usoara a resurselor si a consecintelor aplicarii anumitor actiuni, prin verbe HTTP, asupra starii sistemului. *Figura 8* prezinta secventa mimima de operatii ce trebuie efectuate, la nivel de API, pentru a putea obtine rezultate in urma procesului de parcurgere a paginilor web de catre crawler, modelate printr-o diagrama de secventa.

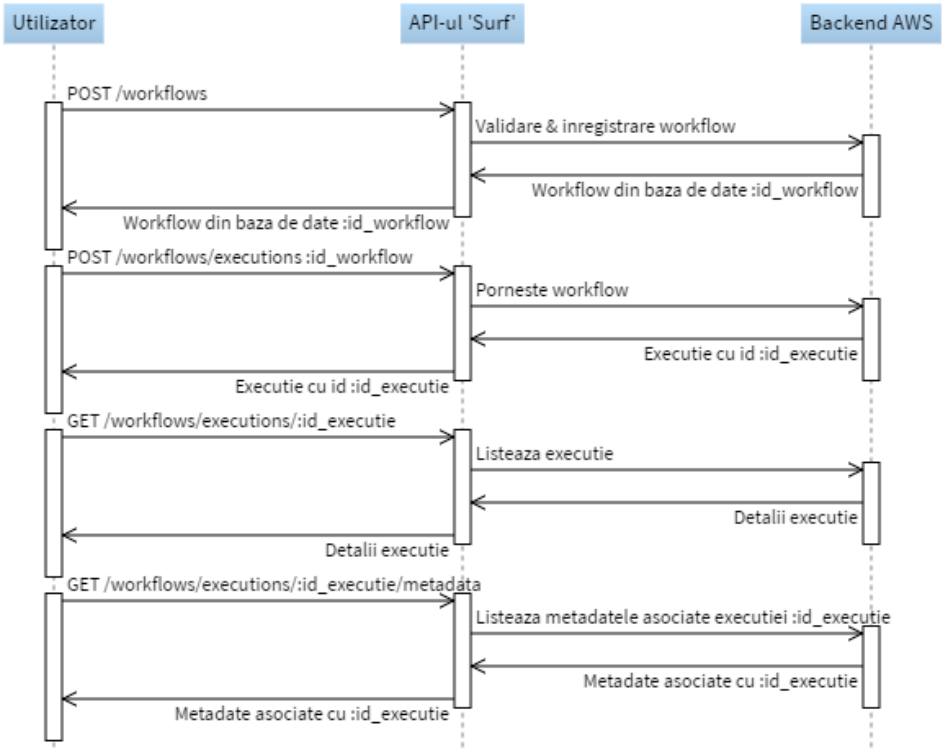

Figura 8: Interactiunea utilizator - API [12, 13]

[31]engl. *RESTful* = adjectiv de la substantivul *REST*; un API este *RESTful* daca adopta stilul arhitectural *REST[15]*

Bibliografie

[1] "Visual Networking Index", Cisco Systems

[2] http://www.internetlivestats.com/total-number-of-websites/

[3] Vladislav Shkapenyuk and Torsten Suel. *Design and Implementation of a High-Performance Distributed Web Crawler.*
`http://cis.poly.edu/tr/tr-cis-2001-03.pdf`

[4] Yugandhara Patil and Sonal Patil. *Review of Web Crawlers with Specification and Working.*
`http://www.ijarcce.com/upload/2016/january-16/IJARCCE%2052.pdf`

[5] Pant Gautam, Srinivasan Padmini and Menczer Filippo. *Crawling the Web.*
`https://pdfs.semanticscholar.org/a4f4/5f3a3d7d53a40b7b2579392a5db88cc1b822.pd`

[6] Pant Gautam, Srinivasan Padmini and Menczer Filippo. *Exploration versus exploitation in topic driven crawlers.*
$https://www.researchgate.net/publication/2946755_Exploration_versus_Exploitation_in_Topic$

[7] S. Ravi Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. *Stochastic models for the Web graph.*
`http://cs.brown.edu/research/webagent/focs-2000.pdf`

[8] Martijn Koster. *A Standard for Robot Exclusion.*
`http://www.robotstxt.org/orig.html`

[9] http://www.cellopoint.com/media_resources/blogs/2011/03/Web_Crawlers

[10] http://cis.poly.edu/tr/tr-cis-2001-03.pdf

[11] Miller, Ron (24 Nov 2015). "AWS Lambda Makes Serverless Applications A Reality". TechCrunch. Retrieved 10 July 2016.

[12] Sursele pictogramelor includ *https://www.iconfinder.com* si *http://simpleicon.com.*

[13] Sursa pictogramelor asociate "Amazon Web Services": *https://aws.amazon.com/architecture/icons/*

[14] Definitia cuvantului *interfata* a fost adaptata pornind de la definitia de la adresa *https://www.merriam-webster.com/dictionary/interface*

[15] *http://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm*