

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

Surf

*Parcurgere și procesare distribuită a paginilor web în cloud pentru
extragerea informațiilor*

propusă de

Ovidiu Pricop

Sesiunea: Iulie, 2017

Coordonator științific

Conf. dr. Sabin Corneliu Buraga

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI
FACULTATEA DE INFORMATICĂ

Surf

Ovidiu Pricop

Sesiunea: Iulie, 2017

Coordonator științific

Conf. dr. Sabin Corneliu Buraga

Declarație privind originalitate și respectarea drepturilor de autor

Prin prezenta declar că lucrarea de licență cu titlul “Surf” este scrisă de mine și nu a mai fost prezentată niciodată la o altă facultate sau instituție de învățământ superior din țară 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 și în traducere proprie din altă limbă, sunt scrise între ghilimele și dețin referința precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alți autori deține referința precisă;
- codul sursă, imaginile etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor și dețin referințe precise;
- rezumarea ideilor altor autori precizează referința precisă la textul original.

Iași, 3 Iulie 2017

Ovidiu Pricop

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul “Surf”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică. De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea Alexandru Ioan Cuza Iași să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, 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 utilizare, flexibilitate, extensibilitate si control precis asupra costurilor.

Cuprins

| | |
|---|-----------|
| Rezumat | 4 |
| Introducere | 6 |
| 1 Crawler-ul Surf | 7 |
| 1.1 Puncte de pornire | 7 |
| 1.2 Parcurgere | 7 |
| 1.3 Selectia informatiilor | 8 |
| 1.4 Viteza, politici de respingere si drepturi de autor | 8 |
| 1.5 Paralelizare | 9 |
| 1.6 Logarea actiunilor si retentia datelor | 9 |
| Crawling in cloud | 11 |
| 1 Infrastructura | 13 |
| 2 Generarea infrastructurii | 17 |
| 3 Executia procesului de crawling | 20 |

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 xml¹ sau json².

Aplicatia "Surf" reprezinta un serviciu web specializat in web crawling³, 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 REST-ful⁴ construit pe platforma AWS⁵ API Gateway. 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 (metadata 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 metadata, intr-o coada AWS SQS si vor fi accesibile utilizatorilor prin procesul de long-polling asupra acestei cozi.

¹Extensible Markup Language - <https://www.w3.org/XML/>

²JavaScript Object Notation - <http://www.json.org/>

³Web crawler - <https://www.techopedia.com/definition/10008/web-crawler>

⁴REST - http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

⁵Amazon Web Services - <https://aws.amazon.com>

1 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[3];
4. Viteza de crawling, fisierul *robots.txt*⁶ 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.

1.1 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 URL⁷ si este furnizat, ca input din partea utilizatorului, la initializarea crawler-ului.

1.2 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.

⁶<http://www.robotstxt.org/robotstxt.html>

⁷Uniform Resource Locator - <http://www.dictionary.com/browse/url>

1.3 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 normalizarea⁸ URL-urilor parcurse de catre crawler.

1.4 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,

⁸<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*⁹. Siturile web sau domeniile care interzic procesul de crawling (e.g. prin ToS¹⁰) vor fi adaugate unei liste de excludiune din procesul de parcurgere executat de crawler.

1.5 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 Lambda¹¹.

Exista doua moduri bine cunoscute de alocare a sarcinilor de web crawling: statica si dinamica[4]. Aplicatia "Surf" implementeaza un mecanism de distribuire statica a sarcinilor:

Fie n , *gradul de paralelizare*¹² 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.

1.6 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 parțiale; 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.

⁹<http://dictionary.cambridge.org/dictionary/english/blacklist>

¹⁰<http://www.pcmag.com/encyclopedia/term/62682/terms-of-service>

¹¹Amazon Web Services Lambda Functions

¹²Numarul 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 voluminoase (in functie de politica de selectie configurata), aplicatia "Surf" salveaza metadata asociate acestor rezultate intr-o tabela DynamoDB. Pe baza acestor informatii (metadata), 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 solicita 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 depanate¹³ 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 scaleze¹⁴ orizontal ca timp, permitand utilizatorului realizarea mai multor sarcini de parcurgere 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

¹³*engl.* debugged

¹⁴<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[5].

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.

Funcțiile 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 OpenID¹⁵. Credentialele obtinute sunt impachetate intr-o cerere catre AWS Security Token Service¹⁶, 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.

¹⁵<http://openid.net/what-is-openid/>

¹⁶Serviciu web ce permite unui utilizator sa solicite credentiale temporare pentru autentificarea in cadrul platformei Amazon Web Services



Figura 1: Procesul de autentificare [6]

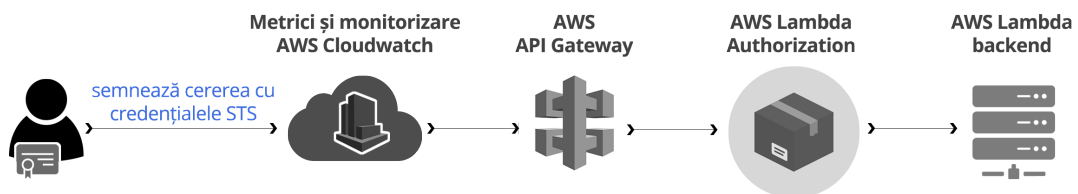


Figura 2: Procesul de autorizare [6]

Prin AWS Identity Access Management (IAM) se vor stabili restricțiile de acces asupra API-ului. Mecanismul de asignare a permisiunilor va fi susținut prin definirea unor roluri. Utilizatorii vor fi grupați, din punct de vedere al drepturilor de acces, în mai multe categorii. Spre exemplu, un utilizator poate avea rolul "administrator" (prescurtat *Ad*), iar un alt utilizator poate avea atât rolul "administrator" (*Ad*) cât și "utilizator de baza" (prescurtat *Ub*). Atât *Ad*, cât și *Ub* vor avea asignate politici de acces asupra datelor AWS. O astfel de politică este următoarea:

```

{
  "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"¹⁷ 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 - utilizator 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:

¹⁷<http://dictionary.cambridge.org/dictionary/english/on-a-need-to-know-basis>

- *data sharding*¹⁸ - 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*¹⁹ - 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 procesului 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*²⁰ 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.

¹⁸<https://docs.microsoft.com/en-us/azure/architecture/patterns/sharding>

¹⁹<https://www.mongodb.com/scale/dynamic-schema-design>

²⁰Unitate 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 urmasorii 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, metadata 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 idempotentia²¹ 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

²¹Termenul 'idempotentia' 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": "#####" // Obfuscate 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 configurare 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.

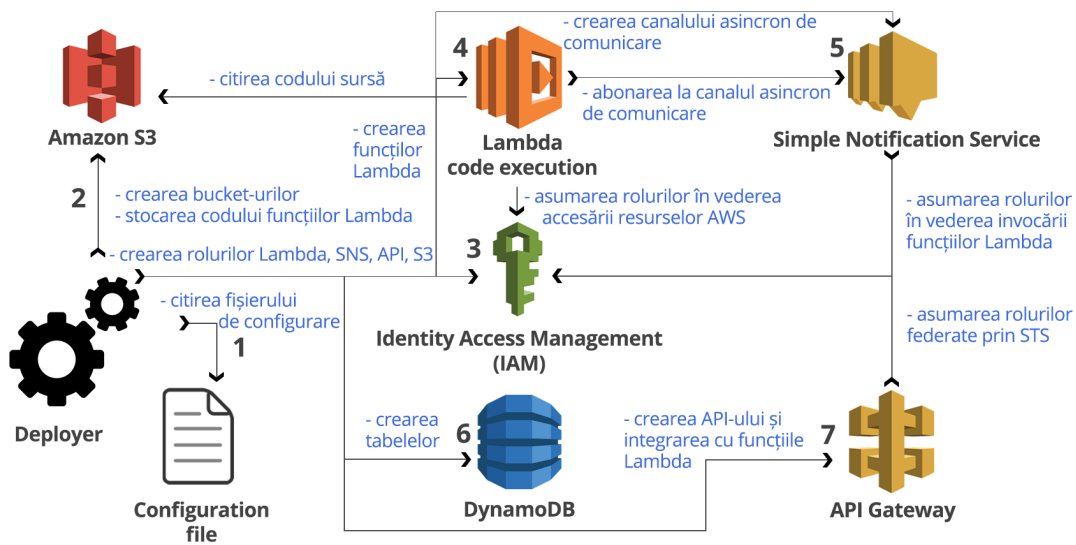


Figura 6: Generarea infrastructurii [6][7]

3 Executia procesului de crawling

Bibliografie

- [1] "Visual Networking Index", Cisco Systems
- [2] <http://www.internetlivestats.com/total-number-of-websites/>
- [3] http://www.cellopoint.com/media_resources/blogs/2011/03/Web_Crawlers
- [4] <http://cis.poly.edu/tr/tr-cis-2001-03.pdf>
- [5] Miller, Ron (24 Nov 2015). "AWS Lambda Makes Serverless Applications A Reality". TechCrunch. Retrieved 10 July 2016.
- [6] Sursele pictogramelor includ *<https://www.iconfinder.com>* si *<http://simpleicon.com>*.
- [7] Sursa pictogramelor asociate "Amazon Web Services": *<https://aws.amazon.com/architecture/icons/>*