UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Tehnologia Informației

Aplicație de management DNS

LUCRARE DE DIPLOMĂ

Coordonator științific
Șef lucrări doctor inginer
Cristian Mihai Amarandei

Absolvent
Petrașco Ilie

Iași, 2019

DECLARAȚIE DE ASUMARE A AUTENTICITĂȚII
LUCRĂRII DE DIPLOMĂ

Subsemnatul(a) PETRAȘCO ILIE,
legitimat(ă) cu CI seria VS nr. 702273, CNP 1950805410105
autorul lucrării APLICAȚIE DE MANAGEMENT DNS
elaborată în vederea susţinerii examenului de finalizare a studiilor de licență organizat de către Facultatea de Automatică şi Calculatoare din cadrul Universității Tehnice „Gheorghe Asachi” din Iaşi, sesiunea IULIE 2019 a anului universitar 2018-2019, luând în considerare conținutul Art. 34 din Codul de etică universitară al Universității Tehnice „Gheorghe Asachi” din Iaşi (Manualul Procedurilor, UTI.POM.02 – Funcționarea Comisiei de etică universitară), declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române (legea 8/1996) și a convențiilor internaționale privind drepturile de autor.

Data Semnătura
02.07.2019

Cuprins

Aplicație de management DNS

Petrașco Ilie

Rezumat

Sistemul de Nume de Domeniu sau DNS este un serviciu distribuit care translatează numele de domeniu în adrese IP sau adresele IP în nume de domeniu. Un server DNS este o aplicație ce răspunde interogărilor referitoare la adresa IP a unei mașini dintr-o rețea sau numele unei mașini în funcție de adresa IP a acesteia. Serverele DNS sunt de 2 tipuri: autoritative și recursive. Serverul DNS recursiv nu face decât să ofere informații despre alte servere DNS sau să stocheze informații în memoria cache, informații referitoare la domenii, provenite de la alte servere DNS. Serverele DNS autoritative sunt responsabile să stocheze informații despre anumite domenii, acestea fiind referite într-un final de către serverele DNS recursive.

Scopul aplicației este de a crea un instrument fiabil ce ar permite utilizatorului să gestioneze informațiile din cadrul unui server DNS autoritativ. Acesta ar avea posibilitatea să adauge noi domeniu în cadrul serverului DNS, să le modifice pe cele existente sau să le șteargă.

Serverul DNS folosit în dezvoltarea aplicației este BIND9. Informațiile referitoare la domenii din cadrul serverului BIND9 sunt stocate în fișiere de tip zonă. Fiecărui domeniu îi corespund două fișiere: fișier de mapare directă și fișier de mapare inversă. Aceste două fișiere sunt referite ulterior într-un fișier de configurare al serverului numit *named.conf*.

Aplicația e formată din două componente majore: server web și server de servicii. Serverul web se află pe aceiași mașină cu serverul de baze de date MongoDB. Serverul de Servicii se află pe aceiași mașină cu Serverul DNS, BIND9 în cazul dat.

Serverul web pune la dispoziție utilizatorului o interfață web cu două pagini principale. O pagină de adăugare a unui domeniu unde utilizatorul este obligat să introducă detalii minime despre domeniu cum ar fi numele acestuia, adresa de rețea și cel puțin adresa IP unui Server de Nume, dar și să completeze complet toate informațiile despre domeniu. Pagina de modificare permite utilizatorului să modifice orice detaliu despre un domeniu existent sau să șteargă complet un domeniu din cadrul serverului DNS.

Serverul web preia informațiile de la utilizator și le stochează în cadrul unei colecții din baza de date MongoDB. Pentru fiecare domeniu este creat câte un document nou cu toate informațiile sale. Serverul de servicii interoghează baza de date la anumite intervale de timp și face modificări în structura de directoare a serverului BIND9. Creează fișiere zonă pentru domenii noi sau domenii ce trebuie modificate, șterge fișiere zonă pentru domenii ce trebuie șterse. Ulterior creează sau șterge referințele necesare pentru fișierele zonă respective.

Proiectul a fost testat manual în cadrul unei rețele private formată din mașini virtuale cu ajutorul aplicațiilor de tip client pentru terminalul Linux: *dig* și *nslookup*.

Aplicația permite un management ușor al unui server DNS autoritativ doar prin instalarea celor 2 componente și configurarea corespunzătoare, oferind utilizatorului posibilitatea de a controla conținutul unui server DNS fără a ști la un nivel avansat cum acesta funcționează.

Introducere

În deceniile anterioare, internetul a evoluat de la un experiment academic, condus de universităţi, institute de cercetare şi operatori de reţele de cercetare, la o parte integrantă a structurii societăţii. Este aproape imposibil să enumerăm impactul pe care îl are internetul asupra societăţii. În afacerile noastre de zi cu zi ne bazăm pe Internet pentru comunicări prin e-mail, comandam produsele online, de la electronică la îmbrăcăminte şi alimente, şi găsim online partenerii noştri de viaţă. Internetul a devenit un punct de bază pentru interacţiunea dintre cetăţeni şi guvern, iar unele state au ajuns aşa departe încât se bazează pe internet pentru procesele democratice, cum ar fi referendumurile şi alegerile.

Protocolul DNS a fost conceput pentru a ne uşura viaţa. În acelaşi mod în care nu ne amintim toate numerele de telefon din agenda telefonică, în acelaşi mod nu ne amintim toate adresele IP ale tuturor site-urilor. Aceasta este motivul de ce DNS-ul este foarte similar cu serviciul de carte telefonică, utilizatorul asignează numelui site-ul, de exemplu „google.com”, o adresă IP „209.85.229.106”, la care computerul utilizatorului poate trimite o cerere de conectare.

Un sistem de nume de domeniu este un sistem distribuit de păstrare şi interogare a unor date arbitrare într-o structură ierarhică. Cea mai cunoscută aplicaţie a unui sistem de nume de domeniu este gestionare domeniilor în internet. Translatarea numelui domeniului în adresă IP se numeşte rezolvarea numelui de domeniu. Numele de domenii sunt mult mai uşor de reţinut decât adresele IP, dar nu oferă nici o indicaţie despre cum o anumită maşină poate fi găsită pe internet. Acest lucru este făcut de către sistemul DNS, care rezolvă domeniile în adevăratele lor adrese – adresele IP. Un calculator se identifică printr-o adresă, unică în Internet, numită adresa IP a calculatorului respectiv. Totodată calculatorul poate avea asociat şi un nume. Astfel, adresa IP este utilizată la nivelul programelor de prelucrare în reţea. La nivelul utilizatorilor cu acces la mediul Internet, identificarea calculatoarelor se face printr-un nume de calculator gestionat de sistemul DNS.

Tipuri de servere DNS

- *DNS autoritativ*: Un serviciu DNS autoritativ oferă un mecanism de actualizare pe care dezvoltatorii îl folosesc pentru a-şi gestiona numele DNS publice. Acest tip de serviciu răspunde la interogările DNS, traducând numele de domeniu în adresa IP, astfel încât computerele să poată comunica între ele. Autoritativ DNS are autoritatea finală asupra unui domeniu şi este responsabilă pentru furnizarea de răspunsuri la serverele DNS recursive cu informaţiile despre adresa IP.

- *DNS recursiv*: de obicei, clienţii nu fac interogări direct la serviciile DNS autoritativ. În schimb, aceştia se conectează în general la un alt tip de serviciu DNS cunoscut ca un *resolver*[1] sau la un serviciu DNS recursiv. Un serviciu DNS recursiv nu deţine nicio înregistrare DNS, dar acţionează ca un intermediar care poate obţine informaţiile DNS în numele dvs. Dacă un DNS recursiv are referinţa DNS în memoria cache stocată pentru o perioadă de timp, atunci răspunde la interogarea DNS furnizând sursa sau informaţiile IP. Dacă nu, trece interfaţa către unul sau mai multe servere DNS autoritative pentru a găsi informaţiile cerute.

1 resolver – aplicaţie menită să răspundă unei interogări referitor la informaţiile despre un domeniu, de obicei adresa IP

Ideea de la care a pornit crearea proiectului a fost problema managementului unui rețele interne de calculatoare în cadrul unei companii, în cadrul unei facultății sau în cadrul unui campus. Odată cu mărirea rețelei, se mărește și dimensiunea numărului de înregistrări din cadrul serverului DNS de tip autoritativ ce îngreunează managementul rețelei date. Devine mai greu să faci modificări, devine mai greu să aduci înregistrări noi în cadrul serverului DNS și nu în ultimul rând devine complicat să depistezi unele erori apărute în interiorul rețelei .

Scopul final al proiectului este crearea unui instrument ce permite adăugarea de noi înregistrări specifice unui domeniu în cadrul unei aplicații de tip server de sistem de nume de domeniu autoritativ. Serverul DNS utilizat este BIND9, un server care are înregistrările stocate în cadrul unui sistem de fișiere. Proiectul este constituit din două componente majore. Un server cu o interfața corespunzătoare ce va permite colectarea datelor de la utilizator, filtrarea, împachetarea lor și salvarea într-o colecție de documente pusă la dispoziție de un server MongoDB. A doua componentă este un serviciu ce va interoga la anumite intervale de timp serverul MongoDB pentru a colecta ultimele modificări din colecția corespunzătoare. După interogarea, serviciul dat va crea sau va șterge fișierele corespunzătoare unui domeniu din structura de fișiere pe care o pune la dispoziție serverul DNS BIND9. Serverul dat va fi restartat pentru a se putea aplica modificările efectuate și ca interogările ulterioare să fie bazate pe ultima structură de fișiere.

Cele 2 componente ale aplicației se află pe mașini diferite. Serverul web și serverul MongoDB se afla pe aceiași mașină. Serverul de servicii ce interoghează serverul MongoDB se află pe o mașină separată, de fapt pe aceiași mașină ca și serverul DNS BIND9.

În urma conectării componentelor și rulării cu succes a aplicației mă aștept ca utilizatorul să fie capabil să introducă datele necesare în câmpurile aplicației web, să primească un răspuns de confirmare în caz că procesul a decurs cu succes sau să primească un răspuns cu erorile apărute pe parcursul procesului. În urma procesării cu succes a datelor de la utilizator, acesta din urmă să fie capabil să interogheze serverul DNS pentru informații despre noua înregistrare creată într-un timp cât mai scurt posibil.

Capitolul 1. Fundamentarea teoretică și documentarea biografică

1.1. Noțiuni fundamentale DNS

1.1.1. Istoria DNS

Sistemul de nume de domeniu a fost inventat inițial pentru a sprijini creșterea comunicațiilor prin e-mail de către **ARPANET**[2] și susține acum Internetul la scară globală. Numele de gazde alfabetice au fost introduse pe ARPANET la scurt timp după crearea sa și au sporit considerabil gradul de utilizare deoarece numele alfabetice sunt mult mai ușor de memorat decât adresele numerice fără semnificație semantică. Numele gazdelor au fost, de asemenea, utile pentru dezvoltarea de programe informatice care permit accesul la rețea, deoarece ar putea face referire la un nume de gazdă constant fără îngrijorare cu privire la modificările adresei fizice din cauza modificărilor de rețea. Desigur, infrastructura rețelei era încă bazată pe adrese numerice, astfel încât fiecare mașină a menținut un fișier *„HOSTS.TXT”* care a furnizat o mapare între numele de gazdă și adresele de rețea într-un set de înregistrări simple de text care puteau fi ușor citite de o persoană sau un program. Nu a trecut mult timp înainte ca oamenii să-și dea seama că păstrarea mai multor copii ale fișierelor gazdă era ineficientă și predispusă la erori. În 1973 s-a trecut la un sistem centralizat ce păstra informațiile despre toate domenii.

Acest sistem centralizat a funcționat bine timp de aproximativ un deceniu, aproximativ între 1973 și 1983. Cu toate acestea, până la începutul anilor 1980 dezavantajele gestionării centralizate a unei cantități mari de date dinamice au devenit evidente pentru toată lumea. Fișierul *„HOSTS.TXT”* a devenit tot mai mare, rata de modificare a acestuia creștea pe măsură ce rețeaua se extindea și au existat întotdeauna erori care au fost apoi propagate în întreaga rețea.

Astfel a apărut ideea creării unui sistem de nume de domeniu distribuit. Principiile de bază ale eventualului sistem de nume de domeniu constau în necesitatea ca domeniile de nivel superior să ofere un punct de plecare pentru delegarea interogărilor, necesitatea ca domeniile de nivel secundar să fie unice și recunoașterea faptului că distribuția serverelor de nume individuale responsabile pentru fiecare domeniu ar oferi avantaje de administrare și întreținere. Deoarece DNS este o parte atât de importantă a funcționării rețelei de internet, software-ul care o rulează trebuie să fie aproape fără defecte și ușor de actualizat atunci când se găsește o eroare.

Aplicația care rulează aproape fiecare server DNS de pe Internet se numește BIND, dezvoltată de către *Berkeley Internet Name Domain,* concepută pentru prima oară ca proiect de licență la Universitatea California din Berkeley și menținut până la versiunea 4.8.3 de către *Computer Systems Research Group.* Versiunile 4.9, 4.9.1 și 4.9.2 ale aplicației BIND au fost lansate de către compania de calculatoare *Digital Equipment Corporation*. Dezvoltatorul principal a fost Paul Vixie.

Versiunile de la 4.9.3 au fost dezvoltate și întreținute de către *Internet Systems Consortium*. O actualizare arhitecturală majoră numită Versiunea 8 a fost co-dezvoltată de Bob Halley și Paul Vixie și lansată în mai 1997. O altă rescriere arhitecturală majoră, numită Versiunea 9, cu sprijin sporit de securitate, a fost elaborată și lansată în anul 2000.

O aplicație similară este DNS Manager dezvoltată de compania 4PSA. DNS Manager este o aplicație ce permite automatizarea și simplificarea managementului unui sistem de servere DNS, fiind utilizat de numeroși furnizorii de găzduire de servicii computaționale. DNS Manager este o aplicație de tip server care permite utilizatorilor să gestioneze zonele DNS. Cu DNS

2 ARPANET - *Advanced Research Projects Agency Network[1]*

Manager, se pot crea și gestiona zonele DNS și înregistrările DNS, zonele DNS de rezervă, se pot gestiona șabloanele DNS, să se colecteze informații DNS de la serverele la distanță etc.

1.1.2. Domenii și Delegare

Din scurta noastră istorie a Serverelor de Nume am văzut cum au apărut trei nevoi:

1. Nevoia unei ierarhii de nume
2. Necesitatea de a răspândi sarcini operaționale pe mai multe Servere de Nume
3. Necesitatea de a delega administrarea Serverelor de Nume

Sistemul de nume de domeniu din Internet rezolvă elegant toate aceste probleme dintr-o singură lovitură (de fapt, întregul RFC[3] 1034[2] mai exact). Sistemul de Nume de domeniu utilizează o structură arborescentă(Figura 1.1). În partea de sus a arborelui se află rădăcina, urmată de domeniile de nivel superior (TLD[4]), apoi numele de domeniu și orice număr de niveluri inferioare, fiecare separat cu un punct. Rădăcina arborelui este reprezentată de cele mai multe ori ca un punct neglijabil (‘.’).

Domeniile de Nivel Superior(TLDs) sunt împărțite în 2 tipuri:

1. Domenii de Nivel Superior General(gTLD[5]), de exemplu: .com, .edu, .net, .org, .mil etc.
2. Domenii de Nivel Superior pentru țări (ccTLD[6]), de exemplu: .us, .ca, .tv , .uk etc.

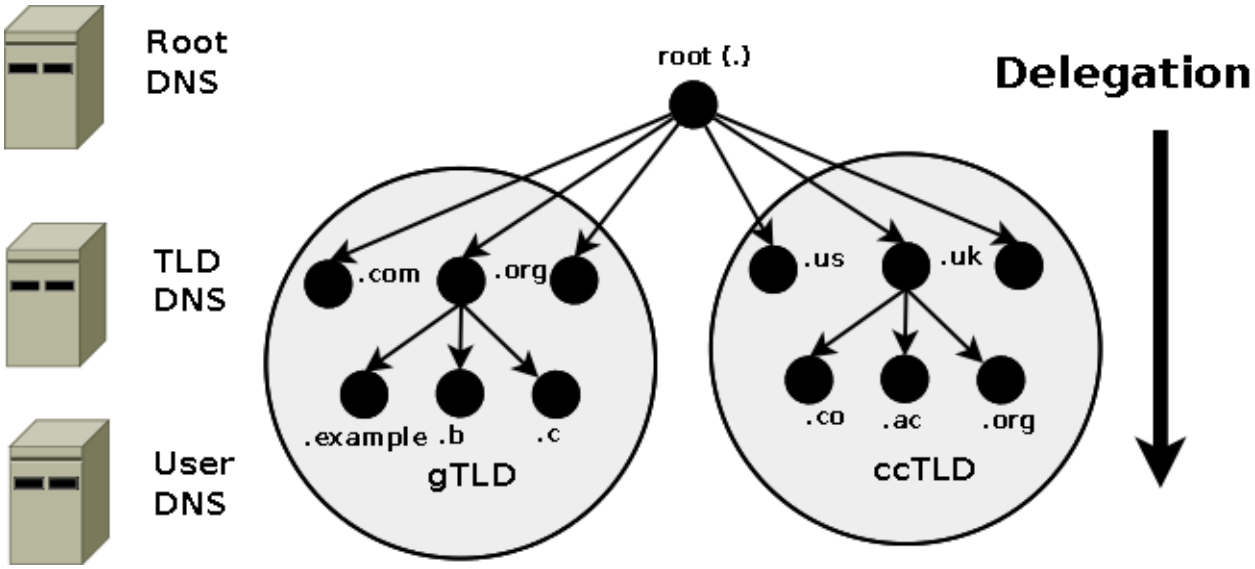

Figura 1.1: Structura de delegare a domeniilor[3]

„Ceea ce se numește de obicei un **Nume de Domeniu** este de fapt o combinație a unui **nume de domeniu** și a unui **TLD** și este scris de la stânga la dreapta cu cel mai mic nivel din ierarhie în stânga și cel mai înalt nivel din ierarhie în dreapta„[3].

3 RFC - Requests for Comments. Documentele care specifică și discută standardele actuale și în curs de dezvoltare pentru Internet
4 TLD - Top Level Domains
5 gTLD - Generic Top Level Domains
6 ccTLD - Country Code Top Level Domain

Ce este de fapt **www.example.com?**

„Citind textul de mai sus se observă ca ***www.example.com*** este format din *www* și *example.com*. Partea de Nume de Domeniu *example.com* a fost delegată de către gTLD(4) care la rândul ei a fos delegată de către ***ICANN***[7]. Partea *www* a fost aleasă de proprietarul domeniului deoarece acesta este acum autoritatea delegată pentru numele *example.com*. El deține totul în partea stângă a Numelui de Domeniu delegat. Partea din stânga, *www* în acest caz, se numește N*ume de Gazdă*. Site-urile prin convenție au ca *Nume de Gazdă* (sau *host*[8]) *www* (pentru web), dar pot avea un site web al cărui nume este *test.example.com*."[3] În mod similar, este posibil să existe un site web al cărui adresă de acces (URL) este *www.example.com, dar* care rulează pe un server al cărui nume real este *mary.example.com*. Din nou, acest lucru este absolut permisiv. Pe scurt, partea de *host* se poate referi la un nume de gazdă real sau un nume de serviciu, cum ar fi *www*. Deoarece proprietarul domeniului controlează acest proces, toate sunt permise.

Fiecare computer sau serviciu care este adresabil (are o adresă URL) prin Internet sau o rețea internă are o parte de *host*, aici sunt câteva exemple ilustrative cu premiza ca o companie deține domeniul *example.com*:

```
www.example.com        --- web serverul companiei
ftp.example.com        --- serverul protocolului de transfer de fișiere
pc3.example.com        --- un calculator din rețeaua companiei
accounting.example.com --- un sistem de contabilitate
```

Host-ul trebuie să fie unic în cadrul domeniului, dar poate fi orice dorește proprietarul.

```
www.us.example.com
```

Partea **us** a fost alocată de către proprietarul *example.com* și se numește *subdomeniu.* În acest caz, autoritatea delegată pentru *example.com* a decis că organizarea companiei este cel mai bine deservită de o structură de subdomenii bazată pe țări. Aceștia ar fi putut delega responsabilitatea internă filialei americane pentru administrarea acestui subdomeniu, ceea ce la rândul lor ar fi creat o structură bazată pe regiuni, cum ar fi *www.cleveland.us.example.com*, care ar putea indica site-ul web al regiunii Cleveland din cadrul organizației americane *exemplu.com*.

1.1.3. Organizare

1.1.3.1. Structură

Serverele din rădăcina arborelui de delegare (Root DNS) sunt responsabilitatea ICANN, dar sunt operate de un consorțiu în baza unui acord de delegare. La momentul dat sunt 13 *Root DNS* servere la nivel mondial. „Serverele *Root* sunt cunoscute fiecărui server public DNS din lume și reprezintă punctul de plecare al fiecărei operații de căutare (sau interogare). Pentru a crea o rezistență suplimentară, fiecare server *Root* are de obicei mai multe instanțe (copii) răspândite în întreaga lume. Fiecare instanță are aceeași adresă IP, dar datele sunt trimise la cea mai apropiată instanță, folosind un proces numit ***anycast***. Atunci când un server DNS nu poate răspunde la o solicitare (o interogare) pentru un nume de domeniu de la un client, de exemplu, *exemplu.com*, interogarea este transmisă unui server *Root* care va direcționa interogarea la un server DNS de tip TLD (pentru *.com*), care va direcționa, la rândul său, la serverul DNS autoritativ domeniului corespunzător"[3].

7 ICANN[4] - Internet Corporation for Assigned Names and Numbers
8 Host – Nume de gazdă

1.1.3.2. Componente

Un **sistem de nume de domeniu** (DNS) definit de RFC 1034 [2] include trei părți:

1. Datele care descriu domeniul (domeniile)
2. Unul sau mai multe aplicații de tip Server de Nume.
3. O aplicație de tip *resolver* sau o bibliotecă

Un singur server DNS poate suporta mai multe domenii. Datele pentru fiecare domeniu descriu proprietățile globale ale domeniului și ale host-urilor (sau serviciilor) acestuia. Aceste date sunt definite sub formă de înregistrări text organizate în fișierele de tip *Zone*. Formatul fișierelor *Zone* este definit în RFC 1035[5] și este susținut de majoritatea software-urilor DNS.

Un **server de nume** de obicei poate face 3 lucruri:

1. Citește un fișier de configurare care definește zonele pentru care este responsabil.
2. În funcție de funcționalitatea serverelor de nume, un fișier de configurare poate descrie diverse comportamente, de exemplu, dacă permite cache sau nu. Unele servere DNS sunt foarte specializate și nu oferă acest nivel de control.
3. Răspunde la întrebările (interogările) de la host-urile locale sau cele de la distanță.

Aplicația de tip *rezolver* sau biblioteca este localizată pe fiecare host și oferă un mijloc de traducere a unei cereri de utilizator, de exemplu, *www.thing.com*, în una sau mai multe interogări către serverele DNS utilizând protocoale UDP (sau TCP).

1.1.4. Fișiere Zone

Un fișier *Zone* este un fișier text care descrie o zonă DNS. O zonă DNS este un subset, adesea un singur domeniu, al structurii ierarhice a numelui de domeniu al DNS. Fișierul de zonă conține mapări între nume de domenii și adrese IP și alte resurse, organizate sub formă de reprezentări de text ale înregistrărilor de resurse. Un fișier de zonă poate fi fie un fișier principal DNS, care descrie autoritativ o zonă, fie poate fi folosit pentru a lista conținutul unei cache DNS.

Un fișier de zonă este o secvență de intrări pentru înregistrările de tip resursă. Fiecare linie este o descriere textuală care definește o singură înregistrare de tip resursă cu formatul dat în Figura 1.2.

Figura 1.2: Formatul Înregistrării de tip Resursă[3]

- Câmpul *name* poate fi lăsat necompletat. În cazul dat înregistrarea moștenește câmpul din înregistrarea anterioară.
- Câmpul *ttl* specifică timpul după care un client de nume de domeniu trebuie să renunțe la înregistrare și să efectueze o nouă operație de rezoluție pentru a obține informații noi. Dacă nu este specificat câmpul *ttl*, se utilizează TTL-ul global specificat în partea de sus a fișierului zonă.
- Câmpul *record class* indică spațiul de nume al informațiilor înregistrate. Spațiul de nume cel mai frecvent utilizat este cel al Internetului, indicat de parametrul **IN**, dar există și sunt utilizate și altele, de exemplu, **CHAOS**.
- Câmpul *record type* este o abreviere pentru tipul de informații stocate în ultimul câmp, *record data*.
- Câmpul *record data* poate consta dintr-unul sau mai multe elemente, în funcție de

cerinţele fiecărui tip de înregistrare. De exemplu, înregistrarea unei adrese IP necesită efectiv adresa, în timp ce o înregistrare pentru un server de mail necesită o prioritate şi un nume de domeniu. Astfel de elemente de informaţie sunt separate prin spaţiu alb.

Înregistrările de tip resursă pot apărea în orice ordine într-un fişier de zonă, cu câteva excepţii. Pentru a facilita formatarea, înregistrările de tip resursă pot include mai multe linii prin includerea în paranteze a unui set de parametri care se întind pe mai multe linii, dar aparţin aceleiaşi înregistrări. Fişierul poate conţine comentarii care sunt marcate cu punct şi virgulă, fie la începutul unei linii, fie după ultimul câmp de pe orice linie sau pe o linie necompletată. Comentariile se termină la sfârşitul unei linii. Fişierul de zonă poate conţine orice număr de linii goale, cu sau fără comentarii.

Fişierul de zonă poate conţine, de asemenea, diverse directive care sunt marcate cu un cuvânt cheie care începe cu caracterul ‘$’. Cel mai notabil este cuvântul cheie **ORIGIN**, care specifică punctul de plecare pentru zona din ierarhia DNS. Un exemplu de fişier zonă este prezentat în secvenţa de cod de mai jos.

```
$ORIGIN testdomain.com.
$TTL 86400
@IN SOA   dns1  hostmaster.testdomain.com. (
                2001062501  ;serial
                21600       ;refresh after 6 hours
                3600        ;retry after 1 hour
                604800      ;expire after 1 week
                86400 )     ;minimum TTL 1 day

          IN    NS    dns1

          IN    NS    dns2.exemple.net.

          IN    MX    10    mail

          IN    A     192.168.56.101

dns1      IN    A     192.168.56.103

el7       IN    A     198.168.56.104

ftp       IN    A     192.168.61.104

mail      IN    A     192.168.56.107
```

„Fişierul de zonă trebuie să precizeze înregistrarea *Start of Authority* (SOA) cu numele serverului autoritativ master pentru zonă şi adresa de e-mail a unei persoane responsabile pentru administrarea serverului de nume. Parametrii înregistrării *SOA* specifică, de asemenea, o listă a parametrilor de sincronizare şi de expirare (seria, perioada de reîmprospătare a serverelor de nume de tip *slave*, timpul de încercare de sincronizare serverelor de nume de tip *slave* cu server de tip *master*, timpul de expirare a serverelor de nume de tip *slave* şi timpul maxim pentru păstrarea înregistrării în memoria cache). Unele servere DNS, cum ar fi BIND, necesită, de asemenea, cel puţin o înregistrare suplimentară a serverului de nume. Adresa de e-mail din înregistrarea *SOA* are simbolul ‘@’ înlocuit cu o punct ‘.’. În fişierul de zonă, host-ul care nu se

termină cu punct ‘.’ este relativ la valoarea câmpului **ORIGIN**„[3]. De exemplu în secvența de cod de mai sus *el7* se referă la *el7.testdomain.com*. Numele care se termină punct se consideră a fi nume de domeniu complet calificate ca în cazul *dns2.exemple.net*.

1.1.5. Interogări DNS

Majoritatea interogărilor pe care un server DNS le va primi vor fi despre domenii necunoscute, adică pentru care nu deține un fișiere de zonă. Serviciul DNS permite de obicei serverului de nume să răspundă în mod diferit la interogările despre domenii necunoscute.
Există 3 tipuri de interogări DNS:

1. „Interogare recursivă - răspunsul complet la întrebare este întotdeauna returnat. Serverele DNS nu sunt obligate să suporte interogări recursive”[3].
2. „Interogare Iterativă (sau non-recursivă) – este posibil ca răspunsul complet să fie returnat sau se furnizează o referință spre un alt server DNS. Toate serverele DNS trebuie să suporte interogări iterative”[3].
3. „Interogare inversă - în cazul în care utilizatorul dorește să cunoască numele domeniului având adresa IP a acestuia. Interogările inverse sunt suportate, foarte rare și sunt de acum învechite (RFC 3425[6])”[3].

1.1.5.1. Interogări iterative

„O interogare iterativă este una la care serverul DNS poate da un răspuns complet, parțial sau poate returna o eroare. Toate serverele DNS trebuie să suporte interogări iterative. O interogare iterativă este o interogare care nu are nevoie de accesarea unor servicii recursive”[3].

Există 4 tipuri de răspunsuri posibile la o interogare iterativă:

1. Răspunsul la interogare, însoțit de o înregistrare de tip *CNAME(1.3.1.3)* (aliasuri).
2. O eroare ce indică faptul ca domeniul sau *host*-ul nu există. De asemenea răspunsul poate conține o înregistrare de tip *CNAME(1.3.1.3)* ce indică spre *host*-ul inexistent.
3. O indicație de eroare temporară - de exemplu, nu se pot accesa alte DNS-uri din cauza unor erori de rețea etc.
4. O referință: Dacă datele solicitate nu sunt disponibile în memoria cache, atunci vor fi returnate numele și adresa/adresele la unul sau mai multe Servere de Nume care sunt cele mai apropiate de domeniul din cerere. Această referință poate fi sau nu poate fi Serverul de Nume autoritativ pentru domeniul cerut.

„De ce se folosesc interogările iterative și de ce sunt preferate în defavoarea celor recursive? Interogările iterative sunt mult mai rapide, serverul DNS care primește interogarea are deja răspunsul în memoria cache, caz în care răspunde imediat sau nu are răspunsul în memoria cache, caz în care trimite o referință spre cel mai apropiat Server de Nume. Interogările iterative oferă mai mult control celui ce face cererea. Un Server de Nume spre care s-a făcut referință conține de obicei o listă de Servere de Nume pentru nivelul următor din ierarhia DNS. Cel ce face cererea poate avea informații suplimentare despre unul sau mai multe dintre acele Servere de Nume în memoria cache (ceea ce este mult mai rapid) din care poate lua o decizie mai bună cu privire la care Server de Nume să utilizeze”[3].

1.1.5.2. Interogări recursive

O interogare recursivă este una în care serverul DNS va da mereu un răspuns pe deplin (sau va da o eroare).

Există 3 răspunsuri posibile la o interogare recursivă:

1. Răspunsul la interogare, însoţit de orice înregistrare de tip *CNAME(1.3.1.3)* (aliasuri) care pot fi utile. Răspunsul va indica dacă datele sunt autoritative sau luate din memoria cache.
2. O eroare ce indică faptul ca domeniul sau *host*-ul nu există. De asemenea răspunsul poate conţine o înregistrare de tip *CNAME(1.3.1.3)* ce indică spre *host*-ul inexistent.
3. O indicaţie de eroare temporară - de exemplu, nu se pot accesa alte DNS-uri din cauza unor erori de reţea, etc.

Traseul urmat de o interogare simplă, precum "care este adresa IP a site-ului *www.example.com*?", pornind de la un *resolver* DNS care acceptă interogări recursive, dar care nu este autoritativ pentru *example.com*, este prezentat în Figura 1.3:

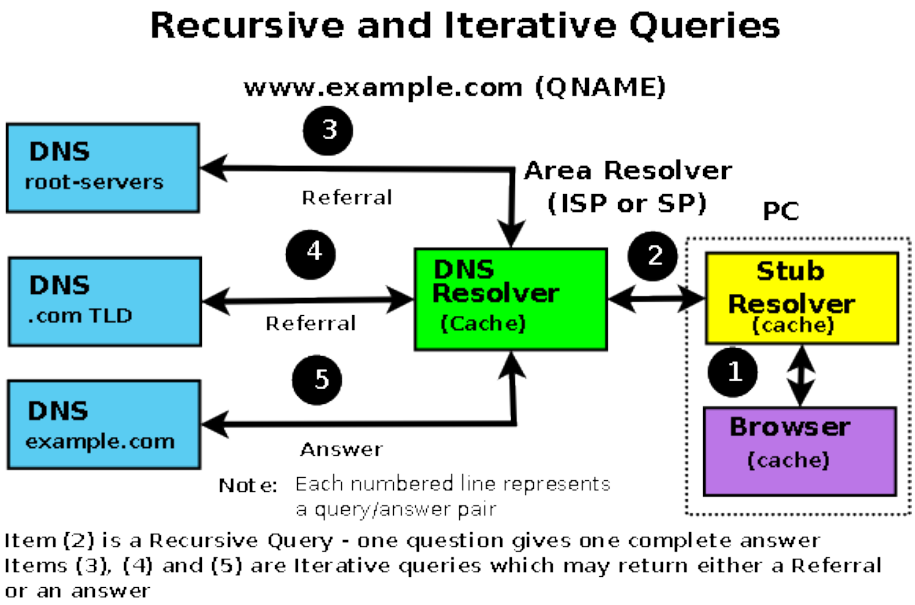

Figura 1.3: Interogare Recursivă[3]

1. Utilizatorul introduce *www.example.com* în bara de căutare a browserului(1). Browserul emite o interogare spre *rezolver*-ul DNS local care nu este decât memoria cache.
2. DNS *rezolver*-ul local trimite interogarea mai departe către cel mai apropiat DNS *rezolver*-ul specializat din reţea.
3. DNS *rezolver*-ul caută adresa *www.example.com* în tabelele locale (memoria cache) şi nu o găseşte. (Dacă ar fi fost găsită atunci ar fi fost returnată şi tranzacţia ar fi completă).
4. DNS *rezolver*-ul trimite cererea mai departe către serverul DNS de tip *Root*(3). Fiecare server DNS este configurat astfel încât cunoaşte toate adresele ip ale celor 13 servere DNS de tip *Root*.
5. Serverul *Root* nu are nici o informaţie despre *www.example.com*, lăsând la o parte *host*-ul *www*, serverul *Root* are informaţii despre următorul nivel de ierarhie, în acest caz *.com* deci trimite un răspuns cu o referinţă spre serverul DNS de tip TLD4 *.com*.
6. DNS *rezolver*-ul trimite o nouă interogare recursivă(4) către serverul DNS de tip *Root* *.com*.
7. Serverul DNS de tip TLD4 are toate informaţiile despre *example.com* dar nu ştie nimic depsre *www*. Odată ce nu poate da un răspuns complet, acesta răspunde cu o referinţă spre Serverul de Nume al domeniului *example.com*.

8. DNS *rezolver*-ul trimite din nou interogarea către Serverul de Nume al domeniului *exemple.com.*
9. Serverul de Nume al domeniului *exemple.com* definește o înregistrare de tip *A(1.3.1.1)* în fișierul său de zonă pentru *host*-ul *www.example.com* cu ajutorul căreia poate răspunde complet la interogare.
10. DNS *rezolver*-ul trimite răspunsul către *rezolver*-ul DNS local, după care salvează răspunsul în memoria cache.
11. *Rezolver*-ul DNS local trimite mai departe răspunsul către browser după care își salvează răspunsul în cache.
12. Browserul salvează răspunsul în cache după care inițiază sesiunea HTTP către adresa primită în răspuns pentru *www.example.com*.

1.2. Maparea Inversă DNS

O interogare DNS normală ar fi de obicei „care este adresa IP pentru *host*=’www’ ce aparține *domeniu*=’example.com’. Sunt cazuri când avem nevoie să aflăm numele *host*-ului sau domeniului având adresa IP al acestuia. Uneori acest lucru este necesar în scopuri de diagnosticare, mai frecvent este folosit în scopuri de securitate pentru a detecta un *hacker* sau un *spammer.* Într-adevăr, majoritatea serverelor de e-mail moderne utilizează maparea inversă pentru a furniza o autentificare simplă, în primul rând, printr-un proces de căutare dublă – adresă IP-nume și nume-adresă IP.

Ca să putem utiliza maparea inversă utilizând interogări normale și interogări iterative, serverele DNS au implementat un nume de domeniu special:

- IN-ADDR.ARPA pentru adrese IPv4
- IP6.ARPA pentru adrese IPv6

„Adresele IP sunt alocate în blocuri de către IANA[7](care la momentul dat este controlat de către ICAAN[4]) care sunt împărțite în 5 Registre de Internet Regionale: AfriNI[9], APNIC[10], ARIN[11], LACNIC[12], RIPE[13]. Fiecare dintre acestea 5 reprezintă câte un continent. La rândul lor Registre de Internet Regionale fac alocarea în blocuri la nivel național: NRI[14], care la rândul lor fac alocarea la nivel local: LIR[15]. LIR-urile sunt de obicei responsabile de alocarea de blocuri de adrese clienților finali. Autoritatea responsabilă pentru alocarea adreselor IP este de asemenea responsabilă și pentru maparea inversă a acestor adrese”[3].

9 AfriNIC - African Network Information Centre
10 APNIC - Asia Pacific Network Information Centre
11 ARIN - American Registry for Internet Numbers
12 LACNIC - Regional Latin-American and Caribbean IP Address Registry
13 RIPE - Reseaux IP Europeans
14 NIR - Național Internet Register
15 LIR - Local Internet Register

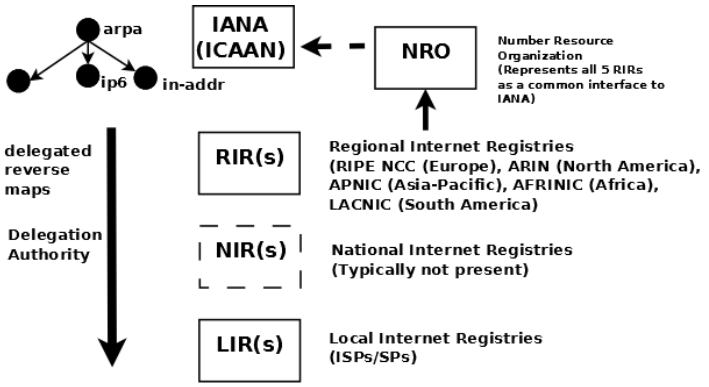

Figura 1.4: Alocarea adreselor și delegarea[3]

1.2.1. Mapare inversă pentru adrese IPv4

Definim structura unui nume de domeniu sub forma unui arbore pornind de la rădăcină. Numele de domeniu se scrie de la stânga la dreapta doar că structura ierarhică este construită de la dreapta la stânga.

```
nume de domeniu =                          www.example.com
nodul cu nivelul cel mai înalt =           .com
următorul nivel =                          .example
următorul nivel =                          www
```

Presupunem că adresa IP pentru *www.example.com* este *192.168.23.17*. Această adresă definește un *host* care face parte din clasa C (192.168.23.x). Omitem partea de *host*, inversăm ordinea adresei IP rămase și o plasăm sub un domeniu numit **IN-ADDR.ARPA** care este *case insensitive*.

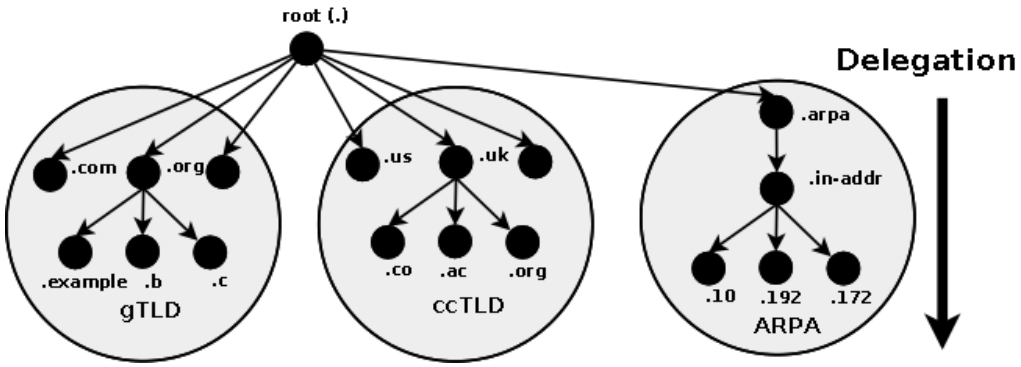

Figura 1.5: Mapare inversă IN-ADDR.ARPA[3]

Obținem adresa ***23.168.192.IN-ADDR.ARPA***. Acum putem construi fișierul zonă corespunzător domeniului nou obținut cu următoarea structură:

```
$TTL 2d  ; 172800 secunde
$ORIGIN 23.168.192.IN-ADDR.ARPA.
@   IN    SOA   ns1.example.com. hostmaster.example.com. (
                2003080800  ; serial number
                3h    ; refresh
```

```
                15m   ; update retry
                3w    ; expiry
                3h    ; nx = nxdomain ttl
                )
    IN    NS    ns1.example.com.
1   IN    PTR   tom.example.com.
2   IN    PTR   joe.example.com.
...
17        IN    PTR   www.example.com.
...
```

„Suntem obligați să folosim numele domeniului complet succedat de punct ‘.’ cu tipul înregistrare **PTR**(1.3.1.6). Dacă nu am face asta și am folosi doar forma prescurtată ca mai jos:

```
 17       IN    PTR   www
```

atunci, făcând substituția cu valoare **$ORIGIN**, am obține *www.23.168.192.IN-ADDR.ARPA.* ceea ce nu este exact ce ne dorim”[3].

1.2.1.1. Delegarea inversă

Maparea normală a numelor de domeniu este un proces independent de orice autoritate ce a făcut alocarea adreselor IP. Dacă proprietarul domeniului dorește să schimbe adresele IP în cadrul domeniului, acesta nu trebuie decât sa facă modificări asupra fișierului zonă.

„Regula este că entitățile pot fi delegare o singură dată în arborele de nume de domeniu ceea ce include și *IN-ADDR.ARPA.* Când o subrețea este alocată de către un ISP[16] sau o altă autoritate de delegare, responsabilitatea pentru maparea inversă este deja asignată acelei autorități. Deci dacă proprietarul domeniului dorește să schimbe numele host-urilor, atunci acesta trebuie să notifice autoritatea care a făcut alocarea adreselor pentru domeniul său, ceea ce adesea durează foarte mult timp și implică multă birocrație. Este de dorit ca atunci când ISP face delegarea subrețelei pentru domeniu, să se ofere și un suport pentru delegarea mapării inverse.

Tehnica definită în RFC 2317[8] oferă această posibilitate utilizând înregistrări de tip *CNAME(1.3.1.3)* în locul celor de tip *PTR*(1.3.1.6) în interiorul spațiului de nume IN-ADDR.ARPA”[3].

```
$TTL 2d  ; 172800 secunde
$ORIGIN 23.168.192.IN-ADDR.ARPA.
@   IN    SOA   ns1.example.com. hostmaster.example.com. (...)

    IN    NS    ns1.isp.com.
    IN    NS    ns2.isp.com.

; definiția subrețelei 192.168.23.64/27
64/27     IN    PTR         ns1.example.com.
64/27     IN    PTR         ns2.example.com.
...
65        IN    CNAME       65.64/27.23.168.192.IN-ADDR.ARPA.
```

16 ISP – Internet Service Provider

Spațiul de nume extins *64/27.23.168.192.IN-ADDR.ARPA* este vizibil doar în fișierul zonă părinte. Ierarhia DNS continuă să vadă adresa normală în cazul unei interogări de mapare inversă în cadrul arborelui *IN-ADDR.ARPA.*

1.2.2. Mapare inversă pentru adrese IPv6

„Adresele IPv6 sunt mapate direct utilizând înregistrări de tip *AAAA(1.3.1.2)* și mapate invers utilizând înregistrări de tip *PTR*(1.3.1.6). Spre deosebire de adresele IPv4, unde maparea inversă nu este adesea delegată utilizatorului final, IPv6 permite și încurajează maparea inversă delegată. Utilizatorul final poate fi, prin urmare, responsabil pentru crearea de fișiere de zonă de mapare inversă utilizând domeniul IP6.ARPA pentru intervalul de adrese care i-a fost alocat”[3].

Presupunem că utilizatorului i-a fost alocată de către ISP-ul local subrețeaua:

```
2001:db8:0::/48
```

Utilizatorului i-a fost la fel delegată responsabilitatea pentru maparea inversă adreselor din subrețeaua de 80 de biți (128 – 48 = 80 partea subrețelei). Inversarea IPv6 folosește principiul normal de inversare a adresei și plasarea rezultatului, în cazul IPv6, sub domeniul *IP6.ARPA*. Mai întâi trebui să aducem adresa de rețea la o formă extinsă:

```
2001:0db8:0000::/48
```

după care aplicăm inversarea:

```
0.0.0.0.8.b.d.0.1.0.0.2.IP6.ARPA
```

Delegarea adreselor IPv6 de către utilizatorul final este de obicei fie pentru subrețeaua /48 (80 de biți pentru adrese de utilizator), fie /56 (72 de biți pentru adrese de utilizator) sau /64 (64 de biți pentru adrese de utilizator). În cazul delegarea adreselor IPv6 se face de către utilizatorul final, se așteaptă ca orice adresă mapată direct utilizând o înregistrare de tip *AAAA(1.3.1.2)* să fie mapată și invers cu o înregistrare de tip *PRT*.

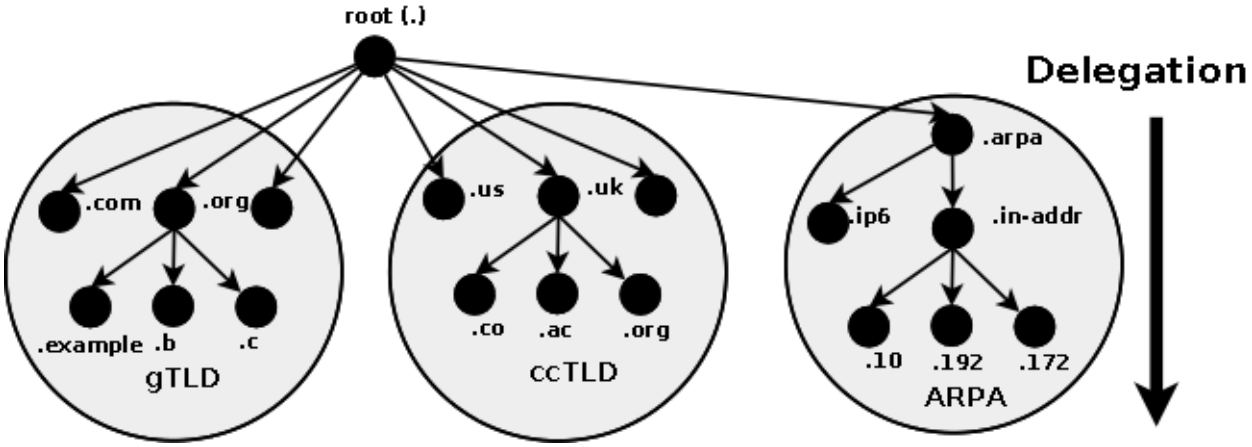

Figura 1.6: Maparea inversă IP6.ARPA[3]

1.3. Înregistrări DNS

Înregistrările DNS (sau RR[17]) descriu caracteristicile unei zone sau a unui domeniu. Acestea pot fi în format text sau în format binar şi sunt plasate în fişierele de zonă, fiind utilizate în procesarea interogărilor venite, iar uneori în crearea propriilor interogări.

Formatul text general al unei înregistrări:

```
owner-namettl   class type        type-specific-data
```

- *owner-name*: sau *eticheta* nodului din fişierul zonă căreia îi aparţine înregistrarea; în funcţie de caz poate lua una dintre următoarele valori:
  1. @ - înlocuit cu valoarea câmpului *@ORIGIN*
  2. ‘ ’ *(*spaţiu gol sau tab) - înlocuit cu valoarea *owner name* a ultimei înregistrări sau cu valoarea câmpului *@ORIGIN* în caz ca nu există nici o înregistrare anterioară
- *ttl*: (Time to Live) valoare pe 32 biţi în secunde care indică cât timp poate fi menţinută înregistrarea în memoria cache; valoarea zero indică faptul că înregistrarea nu trebuie să fie stocată în cache.
- *class*: o valoare pe 16 biţi ce defineşte tipul de protocol; valoare normală este *IN=Internet*, dar mai sunt şi alte valori cum ar fi *HS* şi *CH*
- *type*: tipul înregistrării ce determină valoarea ce poate fi luată de câmpul *type-specific-data*
- *type-specific-data*: conţinutul înregistrării ce depinde de câmpurile *class* şi *type*

1.3.1. Tipuri de înregistrări

1.3.1.1. Înregistrare de tip A

Înregistrarea de tip *A[5]* face mapare dintre un host şi o adresă IPv4. Singurul parametru este adresa IPv4 în format decimal separat prin punct. Câmpul **ipv4** este o adresă nu o etichetă (sau un nume) şi prin urmare, nu este terminată cu un ‘.’ (punct).

```
owner-name   ttl   class    rr      ipv4
joe                IN       A       192.168.254.3
```

Figura 1.7: Format înregistrare de tip A[3]

„Dacă mai multe adrese sunt definite cu acelaşi nume, atunci serverul BIND9 va răspunde la interogări cu toate adresele definite, dar că ordinea se poate schimba în funcţie de valoarea instrucţiunii *rrset-order* din fişierul *named.conf* al serverului BIND9. Ordinea implicită este ciclică sau după tehnica *round-robin[9]*. Acelaşi IP poate fi definit cu nume diferite, în acest caz, o căutare inversă poate să nu dea rezultatul dorit. Adresele IP nu trebuie să fie în aceeaşi clasă sau domeniu. Adresele IPv4 sunt mapate invers cu ajutorul înregistrărilor *PTR*(1.3.1.6)[3].

1.3.1.2. Înregistrare de tip AAAA

Înregistrarea de tip *AAAA[10]* este folosită pentru maparea directă a adreselor IPv6 şi înregistrarea de tip *PTR* pentru maparea inversă a adreselor IPv6.

Sintaxa unei înregistrări de tip *AAAA:*

17 RR – Resource Record

```
name  ttl  class   rr     ipv6
joe        IN      A      2001:db8::1
```

Figura 1.8: Format înregistrare de tip AAAA[3]

„Dacă mai multe adrese sunt definite cu același nume, atunci serverul BIND9 va răspunde la interogări cu o listă de adrese, dar ordinea se poate schimba pentru interogări succesive, în funcție de valoarea instrucțiunii *rrset-order* din fișierul *named.conf* al serverului BIND9. Ordinea implicită este ciclică sau după tehnica *round-robin[9].* Adresele IP nu trebuie să fie în aceeași subrețea sau să utilizeze același prefix global de rutare. Ordinea în care sunt definite înregistrările de tip AAAA nu este semnificativă, dar poate fi mai ușor să fie definite fie într-o ordine ascendentă sau descendentă a adresei IP, deoarece aceasta poate împiedica definițiile duplicate neintenționate. Deoarece câmpul IPv6 este o adresă care nu este un nume, nu există un punct terminator”[3].

1.3.1.3. Înregistrare de tip CNAME

O înregistrare de tipul *CNAME[2]* mapează un nume de host sau de domeniu(un nume Canonic) într-un alias.

Name	Type	Value
blog.example.com	CNAME	example.com
www.example.com	CNAME	example.com
example.com	A	128.0.4.80

Figura 1.9: Înregistrare de tip CNAME

„Înregistrările *CNAME* sunt tratate special în sistemul de nume de domeniu și au mai multe restricții privind utilizarea acestora. Atunci când un *resolver* DNS întâlnește o înregistrare *CNAME* în timp ce caută o înregistrare normală a unei resurse,acesta va reporni interogarea folosind numele canonic în loc de numele original. Dacă în interogare este specificat faptul că se vrea anume înregistrarea de tip *CNAME*, atunci acesta este returnată imediat fără să se mai repornească interogarea. Înregistrarea de tip *CNAME* poate indica oriunde în DNS, atât în domeniul local cât și înafara domeniului„[3].

Restricții asupra înregistrărilor de tip *CNAME:*

- Trebuie să indice mereu spre un a alt domeniu, nicidecum spre o adresă IP
- Înregistrările de tip *CNAME* care indică spre alte înregistrări de tip *CNAME* trebuie evitate din cauza lipsei de eficiență.

1.3.1.4. Înregistrare de tip NS

Înregistrările NS mapează un Server de Nume din cadrul domeniului sau din exteriorul acestuia. Acestea pot apărea în două locuri. În fișierul de zonă, caz în care sunt înregistrări autoritative pentru Serverele de Nume ale zonei sau în momentul delegării fie pentru un subdomeniu al zonei, fie pentru părintele zonei, caz în care sunt non-autoritative. Înregistrările NS la momentul delegării nu sunt niciodată autoritative, doar în cadrul domeniului acestea devin

autoritative.

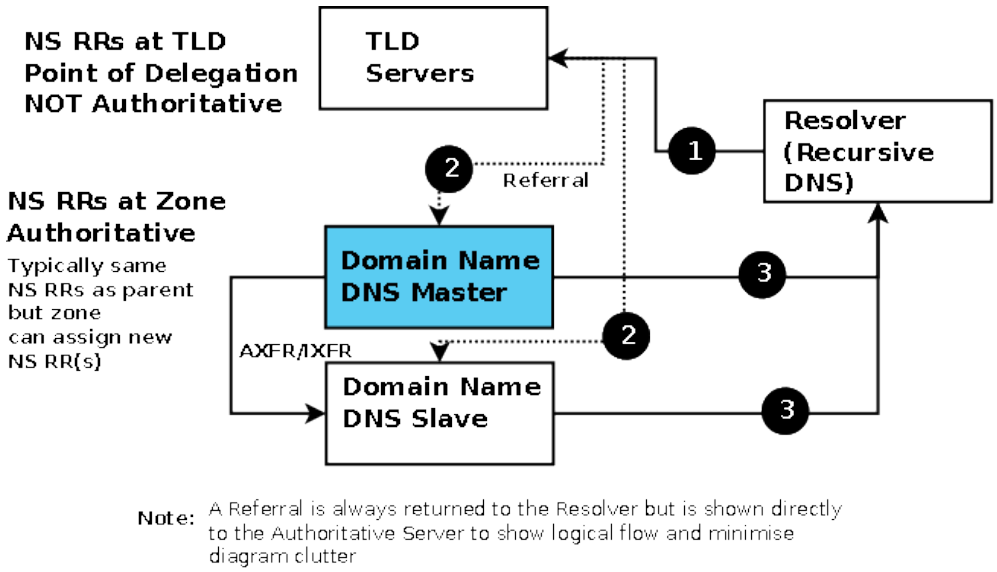

Figura 1.10: Delegarea NS[3]

„Înregistrările NS sunt necesare, deoarece interogările DNS răspund cu o secțiune de autoritate care conține toate Serverele de Nume autoritative. Prin convenție, Serverele de Nume ale unei zone sunt definite imediat după înregistrarea SOA, dar pot fi definite oriunde este convenabil în fișierul de zonă. Cerința este că cel puțin două servere de nume să fie definite pentru fiecare domeniu public (domeniile private pot utiliza doar unul dacă este necesar, multe domenii publice oferă mai mult de două servere de nume) - vor exista întotdeauna cel puțin două înregistrări NS în fiecare fișier de zonă publică. Nu se cere ca serverele de nume să se afle în cadrul domeniului pentru care sunt autoritative. Astfel, un domeniu poate avea zero sau mai multe serverele de nume în cadrul zonei sau zero sau mai multe serverele de nume din exteriorul domeniului (în afara zonei). Dacă serverul de nume se află în interiorul domeniului (în zonă), acesta trebuie să aibă asociat în fișierul de zonă înregistrări de tip A sau AAAA”[3]. Acestea din urmă care definesc serverele de nume aflate în interiorul domeniu sunt deseori numite înregistrări „lipici” și au două roluri de obicei:

1. Să mărească viteza de răspuns la interogări, în consecință să reducă încărcarea serverului DNS prin emiterea tuturor numelor și adreselor IP pentru toate serverele de nume din cadrul domeniului.
2. Să evite interogarea la nesfârșit a serverului DNS pentru aflarea adresei unui server de nume. Presupunem că în cadrul unei interogări a fost returnat numele unui server de nume *ns1.example.com* dar nu și adresa IP al acestuia. Atunci se va crea o nouă interogare pentru *ns1.example.com* care va returna la fel doar numele Serverului de Nume dar nu și adresa Ip al acestuia. Astfel se intră într-o buclă infinită de interogări. Prezența unei înregistrări de tip A sau AAAA previne această anomalie.

```
owner-name      ttl  class   rr      target-name
example.com.         IN      NS       ns1.example.com.
```

Figura 1.11: Înregistrare de tip NS[3]

Câmpul *owner-name* poate lua una dintre următoarele valori:

- Un nume complet de domeniu succedat de punct(‘.’)
- O etichetă care nu trebuie să fie succedată de punct(‘.’)
- ‘@’ care va face substituția cu valoare câmpului *$ORIGIN*
- *spațiu* sau *tab* ce înseamnă ca va fi înlocuit cu valoare anterioară a unei înregistrări; dacă acesta nu există, atunci va face substituția cu valoare câmpului *$ORIGIN*

1.3.1.5. Înregistrarea de tip MX

Înregistrările *MX* specifică numele și preferința relativă în cadrul unui domeniu a unui server de mail. Acestea sunt utilizate de agenții de mail externi pentru a ruta intrările de mail în interiorul domeniului.

```
owner-name          ttl  class   rr  pref name
example.com.        3w   IN      MX  10   mail.example.com.
```

Figura 1.12: Înregistrare de tip MX[3]

„Câmpul **pref** (preferință) este relativ la orice altă înregistrare de tip MX din domeniu. Valorile mai mici sunt cele cu prioritate mai mare. Câmpul *pref* este utilizat de agenții SMTP pentru a selecta serverul de mail cu cea mai mare prioritate. Dacă acesta nu este disponibil atunci următorul sever cu preferința cea mai mică va fi ales. Orice număr de înregistrări MX poate fi definit. Dacă serverul de mail se află în domeniul, este necesară o înregistrare de tip A pentru acesta. Înregistrările MX nu trebuie să indice către un host din același domeniu - acestea pot indica către un server din afara zonei, caz în care o înregistrare de tip A sau AAAA nu trebuie definită”[3].

1.3.1.6. Înregistrare de tip PTR

Înregistrările *pointer* sunt inversul înregistrărilor A1.3.1.1 sau AAAA1.3.1.2 și sunt utilizate în fișierul de zonă de maparea inversă pentru a mapa adresa IP în *host*. Valoare ‘2’ de pe ultimul rând din Figura 1.13 reprezintă de fapt maparea doar *host*-ului *joe* ce face parte din domeniul *example.com*. Dat faptul ca acea valoare nu se termină cu punct, serverul BIND9 atașează valoare câmpului *$ORIGIN* făcând astfel maparea completă a numelui *joe.example.com.*

```
$TTL 2d ; 172800 secs
$ORIGIN 23.168.192.IN-ADDR.ARPA.
@               IN      SOA   ns1.example.com. hostmaster.example.com. (
                                2003080800 ; serial number
                                12h        ; refresh
                                15m        ; update retry
                                3w         ; expiry
                                3h         ; NXDOMAIN ttl
                                )
                IN      NS      ns1.example.com.
                IN      NS      ns2.example.com.
; 2 below is actually an unqualified name and becomes
; 2.23.168.192.IN-ADDR.ARPA.
2               IN      PTR     joe.example.com. ; FDQN
```

Figura 1.13: Înregistrare de tip PTR[3]

„Dacă mai multe nume de *host* sunt asignate aceleiași adrese IP în fișierul zonă de mapare directă, utilizând înregistrări de tip A, AAAA sau CNAME atunci fiecare dintre aceste înregistrări pot fi definite în fișierul zonă de mapare inversă utilizând înregistrări de tip PTR. Nu este obligatoriu dar este recomandat ca fiecare dintre înregistrări de tip A, AAAA sau CNAME din fișierul de mapare directă să aibă o înregistrare de tip PTR în fișierul de mapare inversă. Totuși trebuie de avut în vedere faptul că dacă mai multe înregistrări de tip A, AAAA sau CNAME indică spre aceiași adresă IP atunci răspunsul al o interogare inversă va deveni imens ceea ce poate crea probleme.

Înregistrările inverse pentru adrese IPv4 și IPv6 nu se pot afla în același fișier de zonă inversă la fel cum se întâmplă cu maparea directă adreselor. Adresele IPv6 sunt mapate invers în cadrul domeniului *IP6.ARPA*, în timp ce adresele IPv4 sunt mapate invers sub domeniul I*N-ADDR.ARPA" [3]*.

1.3.1.7. Înregistrare de tip SOA

Înregistrările de tip SOA definesc parametrii globali pentru un domeniu[5][11]. Doar o singură înregistrare poate fi definită în cadrul unui fișier zonă și aceasta trebuie să fie prima înregistrare din acel fișier. Poate fi precedată doar de directivele *$ORIGIN* și *$TTL Figura 1.14.*

Structura unei înregistrări de tip SOA:

- *owner-name*: „În general este o etichetă normală pentru o înregistrare. Cel mai des este folosit un spațiu gol ce înseamnă că se face moștenirea valorii directivei $ORIGIN"[3]. La fel poate fi numele complet al domeniului succedat de semnul punt.
- *ttl:* „Valoare definită pe 32 de biți. Exprimă timpul de aflare a înregistrării în memoria cache. Un server DNS de tip *slave* nu utilizează aceasta valoare"[3].
- *class:* „Definește clasa tipului de înregistrare și ia în mod normal valoarea IN = Internet (implicit dacă nu este prezent). De asemenea, poate lua valoarea HS sau CH"[3].
- *name-server:* „Orice server de nume care va răspunde autoritativ pentru domeniul respectiv. Dacă înregistrarea indică spre un nume de server extern pentru domeniul dat, atunci numele acestuia trebuie să fie numele complet și să fie succedat de semnul punt"[3].
- *email-addr:* „Adresa de e-mail a persoanei responsabile pentru această zonă și la care e-mailurile pot fi trimise pentru a raporta erori sau probleme. Formatul este *căsuță-poștală.domeniu.com*, de exemplu, *hostmaster.example.com* (utilizează un punct în loc de semnul normal @, deoarece @ are alte utilizări în fișierul de zonă), dar că mesajul este trimis la adresa *hostmaster@example.com*. Cel mai adesea adresa este succedată de punct, dar dacă adresa de e-mail se află în acest domeniu, puteți folosi doar *hostmaster* în loc de adresa completă"[3].
- *sn=serial number*: „Valoare definită pe 32 de biți cu incrementarea maximă pană la 2147483647. Această valoare trebuie să fie incrementată de fiecare dată când orice înregistrare din fișierul de zonă este actualizată. Această valoare este salvată de către un server DNS de tip *slave.* Un astfel de server DNS când încearcă să facă sincronizarea cu serverul DNS *master,* acesta cere de la serverul DNS *master* valoare *serial number,* dacă această valoare este mai mare decât valoare cunoscută de serverul DNS *slave*, atunci sincronizarea este începută, altfel nu"[3].
- *refresh:* „Indică la cât timp serverul de tip *slave* va face sincronizarea cu serverul *master.* Valoare este în secunde, recomandate fiind valori din intervalul 1200 – 43200 [12].
- *retry:* „Valoare definită pe 32 de biți. În caz că la încercarea de sincronizare *master-slave* apare o problemă, de conexiune de exemplu, atunci valoare *retry*, reprezintă perioada după care se va încerca o nouă sincronizare."[3]

- *expiry:* „Valoare definită pe 32 de biți în secunde. Reprezintă perioada după care pentru un server *slave*, serverul *master* cu care nu s-a făcut nici un contact de o perioadă îndelungată, nu mai este un server autoritativ pentru domeniul său, relativ la serverul *slave*. Dacă se realizează contactul, valorile de *expiry* și de *refresh* sunt resetate și ciclul începe din nou”[3].
- *nx=nxdomain ttl:* „Valoare definită pe 32 de biți în secunde[13]. Definită ca valoare negativă a timpului de păstrare în memoria cache. Valoare maximă permisă este de 3 ore”[3].

```
owner-name  ttl class rr    name-server email-addr  (sn ref ret ex min)
example.com.    IN    SOA   ns.example.com. hostmaster.example.com. (
                              2003080800 ; sn = serial number
                              172800     ; ref = refresh = 2d
                              900        ; ret = update retry = 15m
                              1209600    ; ex = expiry = 2w
                              3600       ; nx = nxdomain ttl = 1h
                              )
```

Figura 1.14: Înregistrare de tip SOA[3]

Capitolul 2. Proiectarea aplicației

2.1. Contextul utilizării aplicației

Aplicația dezvoltată poate oferi o utilitate maximă doar fiind plasată într-un context potrivit. Aceasta este formată din mai multe componente ce sunt distribuite pe două mașini diferite, dar având cerințe de sistem asemănătoare. Instalarea și configurarea aplicației este de un nivel avansat dar odată ce acest lucru este făcut, utilizarea acesteia facilitează mult operațiile pe care utilizatorul final le-ar fi făcut într-un timp mult mai îndelungat.

Presupunem ca utilizatorul aplicației este un inginer al unei companii ce trebuie să creeze și să configureze rețeaua internă pentru compania respectivă. La fel prin presupunere, compania are un server de e-mail, un site web și multe alte calculatoare ce urmează să se acceseze intern între ele. Compania va fi nevoită să dețină un domeniu care va fi domeniul de bază a tuturor serviciilor web din cadrul acesteia. Un mod efectiv de a face management asupra domeniului este să configurezi și să controlezi singur serverele DNS răspunzătoare pentru domeniul tău. La fel un loc potrivit de aflare fizică a mașinii pe care rulează serverul DNS este chiar în interiorul companiei, fapt ce garantează un control ridicat nu doar la nivel software dar și hardware (sau fizic).

În contextul presupunerilor anterioare, inginerul va trebui să instaleze pe una dintre mașinile interne cel puțin un server DNS care urmează să fie serverul DNS autoritativ pentru domeniul deținut de companie. Odată instalat serverul DNS, va fi nevoie ca toate calculatoarele interne să fie configurate astfel încât să utilizeze ca server DNS primar anume acel server instalat anterior. Referitor la configurarea serverului DNS: ca un server DNS să devină autoritativ pentru un domeniu, acesta trebuie să poată să răspundă la toate interogările ce urmează a fi primite referitor la orice adresă din cadrul rețelei interne asignată domeniului. Pentru acest lucru, serverul DNS are nevoie să cunoască adresele de rețea a fiecărui serviciu făcut public de către companie, împreună cu tipul acelui serviciu și câțiva alți parametri necesari protocoalelor de comunicare pentru serviciul respectiv. Astfel, serverul DNS va avea nevoie de fișiere de tip zonă(1.1.4) care să conțină toate informațiile precizate anterior. Inginerul va trebuie să creeze și să completeze manual acele fișiere, va trebuie să cunoască sintaxa și formatul necesar ca serverul DNS să fie capabil să extragă informația din acele fișiere de tip zonă. Va trebuie să cunoască arhitectura generală a serverului DNS și să poată face configurația necesară ca acesta să localizeze și să încarce corect informațiile despre domeniu respectiv, ca să poată folosi informațiile date în procesarea ulterioară a interogărilor.

Pornind de la presupunerile făcute mai sus, aplicația dezvoltată permite inginerului să facă cu ușurință inserări și modificări în cadrul fișierelor de tip zonă din interiorul unui server DNS. Acest lucru va putea fi făcut de pe orice mașină conectată la internet ce are o interfață grafică și un browser web.

2.2. Arhitectura generală a aplicației

Aplicația va avea nevoie de minim 2 mașini diferite pentru a obține avantaje maxime, dar nu este o greșeală și nu va afecta cu nimic funcționalitatea acesteia dacă aplicația va fi instalată și configurată pe o singură mașină sa chiar pe 3 mașini, după caz. Componentele aplicației sunt bine definite în figura următoare:

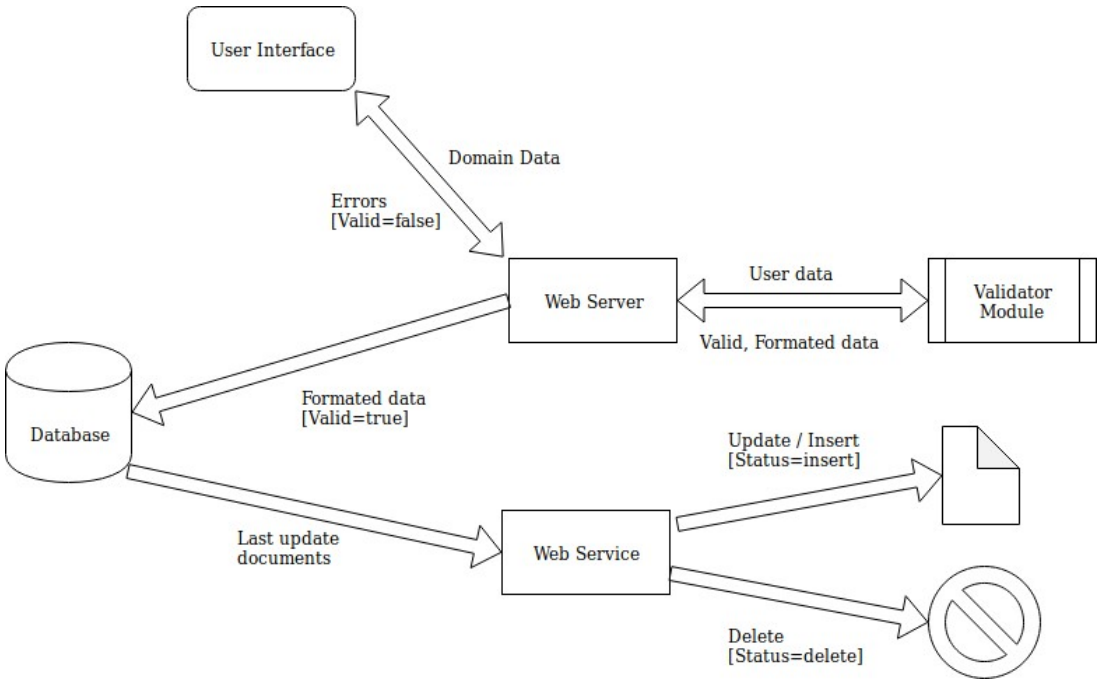

Figura 2.1: Diagramă Funcțională

Funcționalitățile acestor componente sunt complet separate una de cealaltă, iar conectare componentelor poate fi făcută destul de ușor fără a fi necesare cunoștințe avansate de administrare a sistemelor de calcul sau chiar reţelistică.

Diagrama din Figura 2.2 prezintă o imagine generală a structurii aplicației. Din figură se observă că aplicația este distribuită pe 2 mașini separate: o mașină rulează sistemul de operare Ubuntu 18.04, cealaltă mașină rulează sistemul de operare CentOS 7. Pe **mașina primară** (sau **Ubuntu**), vor rula 2 servere, ce reprezintă primele 2 componente de bază ale aplicației: serverul web și serverul de baze de date, care este reprezentat de un server MongoDB. Pe **mașina secundară** (sau **CentOS 7**), vor rula celelalte 2 componente majore ale aplicației: un server de servicii și serverul DNS, reprezentat de BIND9.

Clientul va accesa interfața web pusă la dispoziție de serverul web aflat pe mașina primară. În interfața web acesta va introduce datele referitoare la domeniu la fel și oricât de multe resurse dorește aceste de unul din tipurile: server serviciu de e-mail, server de nume sau un *host* oarecare din subrețeaua domeniului. Serverul web va face procesarea datelor și va returna 2 răspunsuri:

- o listă de erori în cazul în care sunt date invalide
- o pagină de confirmare în cazul în care datele sunt corecte

Datele valide sunt împachetate într-un format JSON[18] și sunt salvate într-un document din cadrul unei colecții a serverului MongoDB.

Pe mașina secundară rulează serverul de servicii web și serverul DNS reprezentat în cazul dat de BIND9. Serverul de servicii va funcționa în mod continuu interogând temporar baza de date aflată pe mașina primară, va extrage din baza de date ultimele modificări, adică cele ce au apărut de la interogarea anterioară și va decide ce să facă cu fiecare document returnat la interogare în funcție un câmp numit *status*. Sunt două variante pentru valoare câmpului *status:*

1. *insert* – caz în care serverul de servicii va crea fișierele de zonă necesare pentru domeniu precizat în document și va restarta serverul DNS

18 Formatul JSON este reprezenta sub forma { **cheie: valoare** }

2. *delete* – caz în care serverul de servicii va șterge fișierele de zonă din structura de directoare a serverul DNS și va restarta la fel serverul DNS

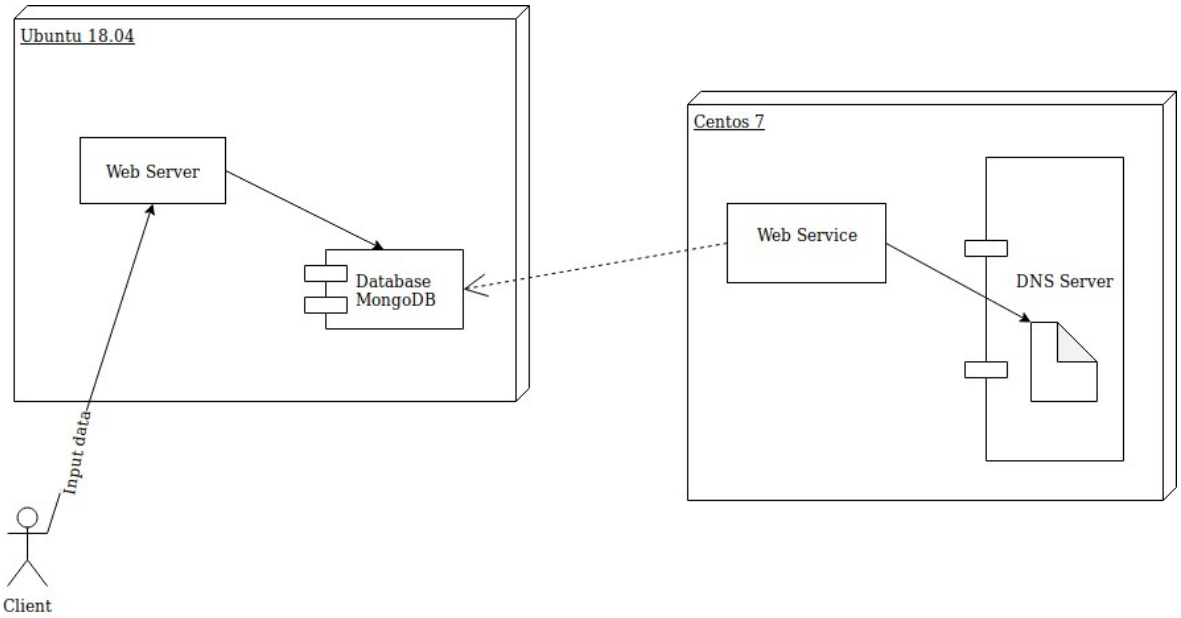

Figura 2.2: Diagrama de componente a aplicației

2.3. Tehnologii folosite

2.3.1. Python

Python este un limbaj de programare interpretat, interactiv, orientat pe obiect și nivel înalt. Limbajul înglobează mai multe paradigme de programare, în special paradigma imperativă (C) și pe cea orientată pe obiecte (Java). Spre deosebire de C, Python nu este un limbaj compilat, ci interpretat. Acest fapt are atât avantaje, cât și dezavantaje. Pe de-o parte, Python este mai lent decât C. Pe de altă parte, aplicațiile Python sunt foarte ușor de depanat, codul putând fi ușor inspectat în timpul rulării.

Am ales Python deoarece este o limbaj de nivel înalt care accentuează expresivitatea și înțelegerea ușoară a codului. Sintaxa este foarte simplă și intuitivă, iar limbajul este foarte răspândit în zona de scripting și de programare a aplicațiilor. Sintaxa Python este destul de intuitivă. Variabilele nu trebuie declarate în prealabil și vor avea tipul de date stabilit în mod dinamic, în funcție de asignarea făcută.

Flask

Flask este un framework web[14] scris în Python. Acest lucru înseamnă că Flask oferă instrumente, librării și tehnologii ce permit crearea unei aplicații web. Această aplicație web poate fi o parte dintr-un site web, un blog, o pagină wikipedia sau la fel poate fi o aplicație mult mai mare cum ar fi un site comercial. Flask este considerat un micro-framework ceea ce înseamnă că nu are nevoie biblioteci și dependințe externe pentru a funcționa. Cu toate acestea Flask are create biblioteci ce pot fi instalate oricând, micșorând astfel munca pe care ar trebuie să o facă un programator. Motorul de randare folosit de Flask este Jinja2[15].

2.3.2. JavaScript

JavaScript este un limbaj de programare orientat obiect bazat pe conceptul de prototip [16]. Este folosit mai ales pentru introducerea unor funcționalități în paginile web, codul JavaScript din aceste pagini fiind rulat de browser. Limbajul este binecunoscut pentru utilizarea în construirea site-urilor web, dar este folosit și pentru accesarea obiectelor încapsulate (obiecte încorporate) în alte aplicații. Browserele reține în memorie o reprezentare a unei pagini web sub forma unui arbore de obiecte și pun la dispoziție aceste obiecte scripturilor JavaScript, care le pot citi și manipula.

O tehnică de creare a paginilor web tot mai întâlnită în ultimul timp este AJAX (Asynchronous JavaScript and XML). Această tehnică constă în executarea cererilor HTTP în fundal, fără reîncărcarea întregii pagini web, și actualizarea numai a anumitor porțiuni ale paginii prin manipularea arborelui de documente. Tehnica AJAX permite construirea unor interfețe web cu timp de răspuns mic, deoarece operația de încărcare a unei pagini HTML complete este în mare parte eliminată.

În proiectarea aplicației am decis să folosesc tehnica AJAX din 2 motive:

1. *Construirea unei interfețe web dinamice*. Utilizatorul poate introduce mai multe tipuri de înregistrări, de fapt 3 la număr. Fiecare tip de înregistrare conține cel puțin 3 câmpuri, iar utilizatorul e lăsat să introducă câte câmpuri dorește și de ce tip dorește. Pentru a nu încărca pagina inutil cu câmpuri care nu se știe dacă vor fi folosite sau nu, am decis ca să creez câmpuri pentru o singură instanță de fiecare tip de înregistrare, iar utilizatorul să poată adăuga dinamic, după încărcarea paginii oricâte câmpuri dorește și de orice tip.
2. *Împachetarea datelor înainte de trimitere la server*. Dată fiind situația de la punctul anterior, unde utilizatorul va crea un număr necunoscut de câmpuri pentru fiecare dintre cele 3 tipuri de înregistrări disponibile, o prelucrare ulterioară la server va deveni destul de complicată. În acest caz am decis să fac mai întâi o validare pe partea de client, extrag doar datele utile, le împachetez într-un format JSON și le trimit la server asincron prin AJAX. Această abordare complică lucrurile pe partea de client, dar simplifică foarte mult prelucrarea pe partea de server, unde formatul datelor primite este deja cunoscut.

2.3.3. MongoDB

MongoDB este o bază de date non-relațională, orientată pe documente [17]. MongoDB face parte din familia sistemelor de baze de date NoSQL. Diferența principală constă în faptul că stocarea datelor nu se face folosind tabele ca într-o bază de date relațională, MongoDB stochează datele sub formă de documente JSON cu schema dinamică. MongoDB poate conține mai multe baze de date, colecții și indecși. Pentru fiecare colecție se pot specifica indecși secundari și compuși. Orice atribut poate fi indexat. Colecțiile conțin documente (BSON[19]). Aceste documente conțin la rândul lor mai multe câmpuri. În MongoDB nu există câmpuri predefinite spre deosebire de bazele de date relaționale, unde există coloanele care sunt definite la momentul în care tabelele sunt create. Nu există schemă pentru câmpurile dintr-un document, acestea și tipurile lor pot varia. În practică este obișnuit ca o colecție să aibă o structură omogenă, deși nu este o cerință, colecțiile putând avea diferite structuri. Această flexibilitate presupune ușurință în migrarea și modificarea imaginii de ansamblu asupra datelor.

Folosirea unei baze de date non-relaționale a fost o opțiune mai bună în locul unei baze de date relaționale odată ce am decis să folosesc tehnologia AJAX pentru trimiterea datelor la

19 BSON este codificare binară a formatului JSON folosit în reprezentarea documentelor unei colecții în MongoDB

server. Odată cu procesarea datelor la server, adică validarea lor, se păstrează și formatul în care aceste date au fost primite de la client, adică format JSON. Formatul exact al acelui JSON va fi discutat mai târziu, tot ce se poate spunea acum este că acel obiect JSON conține atât perechi simple *cheie:valoare*, cât și perechi mai complexe cum ar fi *cheie:listă_obiecte*, prin *listă_obiecte* se referă la faptul că un element din acea listă conține la rândul său perechi simple *cheie:valoare*. Din descrierea anterioară se deduce faptul că folosirea unei baze de date relaționale va complica lucrurile și duce la o procesare în plus și care nu va aduce nici un avantaj.

2.3.4. jQuery

jQuery este o librărie JavaScript ce simplifică gestionarea unui document HTML: navigarea, manipularea evenimentelor, animațiilor, interacțiunilor de tip AJAX etc [18]. Scopul acestei librării este de a schimba modul în care JavaScript interacționează și își lasă amprenta asupra muncii programatorului. jQuery este probabil cea mai cunoscută și folosită bibliotecă de JavaScript, utilizată pentru a simplifica codul scris pe partea de client. În momentul de față, jQuery este disponibil și accesibil în fiecare browser web, fiind creat într-o manieră cât mai simplă cu putință în comparație cu JavaScript.

Capitolul 3. Implementare

3.1. Implementarea Server Web

Prin server web se face referire la acea componentă care se află pe mașina primară (sau Ubuntu 18.04). Această componentă este conectată direct cu serverul de baze de date MongoDB, server care în abordarea dată se află pe aceiași mașină ca și serverul web. La fel de bine serverul de baze de date s-ar putea afla pe o mașină separară sau pe aceiași mașină cu serverul de servicii. Acest lucru implică doar altă configurație a conexiunilor dintre componente.

Serverul web are rolul de a colecta datele de intrare de la client, date referitoare la domeniu și referitoare la înregistrări, după care datele sunt validate pe partea de client, împachetate și trimise la server printr-o cerere HTTP asincronă, utilizând tehnica AJAX. Pe partea de server datele sunt validate încă o dată, după care sunt salvate într-un document dintr-o colecție a bazei de date.

3.1.1. Implementarea pe partea de client

Implementarea componentei reprezentate de serverului web pe partea de client înseamnă implementarea interfeței web împreună cu posibilele modificări dinamice ale acesteia, modulelor ce sunt răspunzătoare de validarea datelor pe partea de client, dar și împachetării și trimiterii datelor către server prin protocolul HTTP.

Interfața web

Interfața web constă din două pagini principale ce pot fi accesate printr-un meniu situat în partea superioară a paginii, având câte un buton pentru fiecare dintre cele 2 pagini. La fel interfața web mai are implementate pagini de confirmare dar și pagini de eroare, dar care nu pot fi accesate în mod normal prin nici un buton. Aceste pagini sunt afișate doar în cazul unor operații de succes sau erori apărute în procesare datelor pe partea de server. Cele două pagini principale ale aplicației web sunt:

- pagina start unde utilizatorul poate adăuga un nou domeniu împreună cu toate înregistrările acestuia
- pagina de *update* sau pagina unde utilizatorul poate modifica sau chiar șterge un întreg domeniul

Pagina de start

Pagina de start este structurată în 3 secțiuni, acestea fiind controlate de un meniu vertical situat pe partea stânga sus a paginii. Se poate observa în Figura 3.1 meniul vertical care urmează să fie prezent pe toate secțiunile din cadrul paginii de start. La fel pe butoanele meniului vertical sunt prezente niște săgeți de culoare verde îndreptate spre dreapta, fapt ce sugerează utilizatorului să continue navigarea prin meniul paginii care urmează să se sfârșească undeva spre ultima secțiune. Pagina poate fi trimisă la server doar cu ajutorul unui buton situat pe ultima secțiune a paginii de start, deci utilizatorul va fi nevoit să folosească cel puțin o singură dată meniul de control al celor 3 secțiuni.

Cele 3 secțiuni sunt:

1. ***DOMAIN***. Aceasta este prima secțiune și cea care apare în mod implicit în pagină odată ce utilizatorul accesează pagina de start, fie prin meniul vertical, fie prin accesarea efectivă a aplicației web. Această secțiune este împărțită în doua sub-

secțiuni, iar completarea câmpurilor este obligatorie în cazul ambelor:

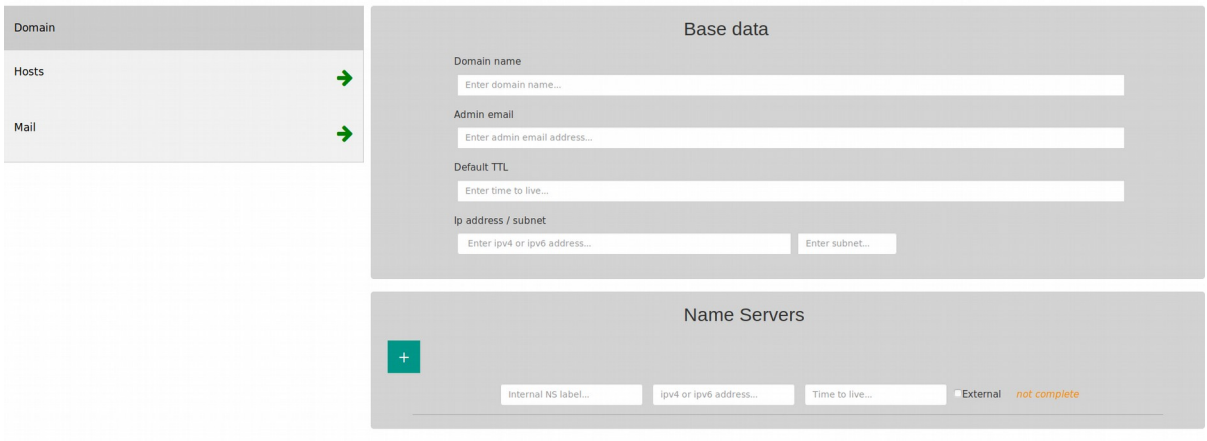

Figura 3.1: Pagina de start. Secțiunea 1

- *Base data.* Subsecțiune conține 5 câmpuri:
  - *Domain Name.* Numele domeniului ce urmează să fie înregistrat. Acest câmp are validări pe partea de client utilizând atributul *pattern* în interiorul elementului *input* cu un șir de caractere drept șablon .
  - *Admin Email.* Adresa de e-mail al administratorului domeniului, adresă la care se pot adresa întrebări despre domeniu sau se pot trimite rapoarte de erori. Validarea pe partea de client este făcută cu ajutorul atributului *type=„email”* ce implică validarea implicită a câmpului pentru o adresă de e-mail
  - *Default TTL.* Acest câmp va reprezenta valoare directivei *$TTL* din fișierul de zonă corespunzătoru domeniului. Această valoare va fi valoare implicită a oricărei înregistrări în caz ca acea înregistrare nu are definită o valoare explicită a câmpului TTL. Validarea se realizează prin folosirea atributului *type=„number”* fapt ce obligă utilizatorul introducerea doar de numere, cât și a atributelor *min* și *max* ce reprezintă intervalul de numere valabil pentru acel câmp
  - *Ip address / subnet.* Ambele câmpuri trebuie precizate în mod obligatoriu și reprezintă adresa rețelei în care se află domeniul. Primul câmp este adresa ip a rețelei care este de tip text, deoarece poate conține atât adrese ipv4 cât și adrese ipv6, iar al doilea câmp este masca de rețea ce este e tip numeric cu intervalul [0, 128] ce este intervalul maxim sub-masca de rețea a unei adrese ipv6.

  Toate câmpurile din cadrul aceste sub-secțiuni sunt obligatorii. Ele constituie setul de date minime pentru definirea unui domeniu cu o adresă de rețea.

b) *Name servers (sau Serverele de Nume)*Figura 3.2. Această sub-secțiune reprezintă primul tip de înregistrări ce pot fi inserate în cadrul unui fișier de zonă. Modul în care a fost construită pagina obligă utilizatorul să insereze cel puțin o înregistrare de tip NS. Cu ajutorul butonului din colțul stânga sus, reprezentat de un pătrat mic cu semnul „plus”, utilizatorul poate adăuga oricât de multe înregistrări de tip NS dorește, dar singura condiție este să completeze prima înregistrare în mod obligatoriu, altfel nu va putea trimite datele la

server. Fiecare înregistrare de tip NS este reprezentată de 4 câmpuri:

- *NS label / NS domain.* Câmp de tip text ce poate reprezenta 2 lucruri diferite în funcţie de câmpul *External* definit mai jos. Cazul *NS label* are ca şablon de validare un şir de caractere pentru o simplă etichetă. Cazul *NS domain* are ca şablon de validare un şir de caractere pentru un nume de domeniu complet.
- *IPv4 or IPv6 address* este un câmp text ce defineşte adresa ip a serverului de nume din cadrul domeniul.
- *Time to live.* Este durata de timp în care înregistrarea curentă se poate afla în memoria cache. Poate fi necompletat indiferent de orice condiţii.
- *External.* Câmp de tip *checkbox* ce marchează înregistrarea ca Server de nume din exteriorul domeniului, fapt ce nu necesită o adresă de ip, doar numele complet la care poate fi găsit serverul de nume respectiv. Din această cauză, când e bifat, câmpul *IPv4 or IPv6 address* este dezactivat.

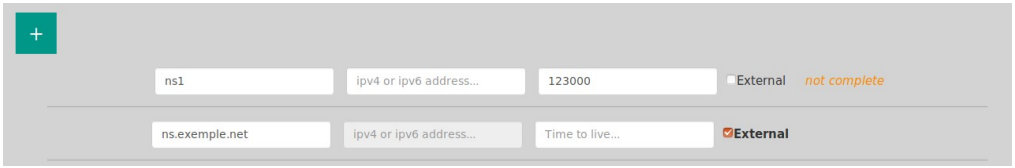

Figura 3.2: Înregistrări NS

Dat fiind faptul ca utilizatorul poate introduce oricâte înregistrări doreşte, se poate întâmpla ca acesta să adauge din greşeală câmpuri pentru o nouă înregistrare dar să nu o mai completeze. Din cauza asta în dreptul fiecărei înregistrări poate apărea mesajul de avertizare *„not complete”* ce indică faptul că înregistrarea nu este completată definitiv şi nu va fi trimisă la sever Figura 3.2. Această validare se face utilizând o funcţie JavaScript ataşată evenimentului *focusout* ce este declanşat de câmpurile înregistrării.

2. ***HOST*** este a doua secţiune a paginii de start Figura 3.3. Aceasta defineşte o interfaţă pentru adăugarea înregistrărilor pentru *host*-uri normale din cadrul domeniului. La fel ca şi în cazul sub-secţiunii *NS* există un buton de adăugare a unui număr dorit de câmpuri pentru înregistrări. În cazul înregistrărilor de tip *host* nu e obligatoriu să avem cel puţin una, este la dorinţa utilizatorului să lase toate câmpurile necompletate. O înregistrare de tip *host* are 5 câmpuri:
   - *Host name.* Câmp de tip text ce defineşte numele *host*-ului. Acesta este validat pe partea de client cu ajutorul unui şablon corespunzător. Este obligatoriu ca o înregistrare să aibă acest nume ca să fie validă.
   - *IPv4 or IPv6 address.* Este un câmp de tip text ce reprezintă adresa ip a *host*-ului. Poate fi adresă IPv4 sau IPv6.
   - *Time to live.* Este durata de timp în care înregistrarea curentă se poate afla în memoria cache. Poate fi lăsat necompletat.
   - *TXT record.* Câmp de tip text de maxim 255 de caractere. Acest câmp va forma o înregistrare nouă în interiorul fişierului de zonă, o înregistrare de tip *TXT* care va face referire la înregistrarea pentru *host*-ul curent. Poate fi lăsat necompletat.
   - *CNAME*. Câmp de tip text validat pe partea de client cu ajutorul unui şablon de validare pentru o etichetă simplă. Acest câmp va forma o înregistrare nouă de tip *CNAME* în interiorul fişierului de zonă, care va face referire la înregistrarea

pentru host-ul curent. Poate fi lăsat necompletat.
Câmpurile de tip *host* au un status care indică dacă înregistrare curentă este completă și dacă va fi trimisă la server sau nu. Pentru ca înregistrarea să fie completă, aceasta trebuie să aibă completate corect primele 2 câmpuri: *Host name* și *IPv4 or IPv6 address.*

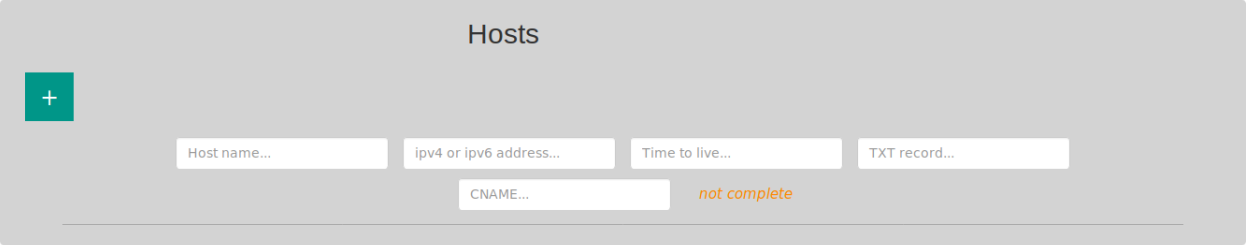

Figura 3.3: Înregistrări de tip host

3. ***MAIL*** *Figura 3.4*. Ultima secțiune a paginii principale are două roluri în contextul trimiterii datelor către server: preluarea datelor referitor la înregistrările pentru serverele de e-mail și punerea la dispoziție a butonului de *submit* ce va apela funcțiile JavaScript de colectare, împachetare și trimitere a datelor la server. La fel ca și în cazul sub-secțiunii *NS* și a secțiunii *Hosts* există un buton de adăugare a unui număr dorit de câmpuri pentru înregistrări. O înregistrare de tip *mail* conține 7 câmpuri.
   - *Mail host name / Mail domain Name.* Câmp de tip text ce poate reprezenta 2 lucruri diferite în funcție de câmpul *External* ce urmează a fi definit. Cazul *Mail host name* are ca șablon de validare un șir de caractere pentru o simplă etichetă. Cazul *Mail domain name* are ca șablon de validare un șir de caractere pentru un nume de domeniu complet.
   - *Mail host ip address* este adresa IPv4 și IPv6 a serviciului de e-mail din cadrul domeniului. Acest câmp poate fi completat doar în cazul în care înregistrarea nu este una externă.
   - *Preference* este un câmp de tip numeric. Acesta reprezintă prioritatea cu care va fi folosit serviciul de e-mail ce este definit de înregistrare curentă. Valoare minimă este zero, iar înregistrările cu valoare preferinței mai mică au o prioritate mai mare.
   - *Time to live.* Este durata de timp în care înregistrarea curentă se poate afla în memoria cache. Poate fi lăsat necompletat.
   - *TXT record*. Câmp de tip text de maxim 255 de caractere. Acest câmp va forma o înregistrare nouă în interiorul fișierului de zonă, o înregistrare de tip TXT care va face referire la înregistrarea pentru host-ul de mail curent. Poate fi lăsat necompletat.
   - *CNAME*. Câmp de tip text validat pe partea de client cu ajutorul unui șablon de validare pentru o etichetă simplă. Acest câmp va forma o înregistrare nouă de tip *CNAME* în interiorul fișierului de zonă, care va face referire la înregistrarea pentru host-ul de mail curent. Poate fi lăsat necompletat.
   - *External*. Câmp de tip checkbox ce marchează înregistrarea ca serviciu de e-mail din exteriorul domeniului, fapt ce nu necesită o adresă de ip, doar numele complet la care poate fi găsit serviciul respectiv.

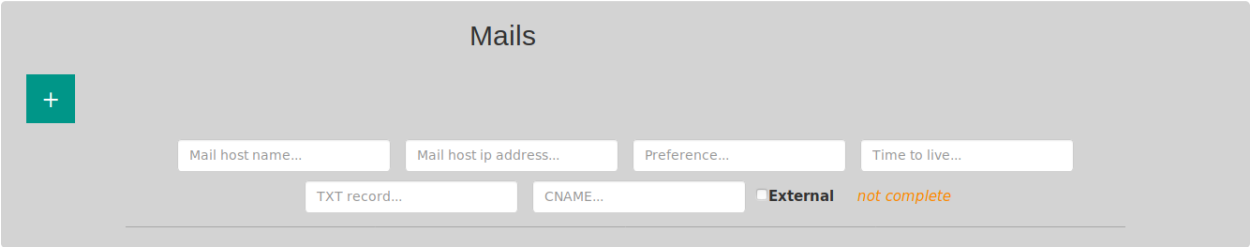

Figura 3.4: Înregistrare de tip mail

Înregistrările de tip *mail* au un status ce indică faptul dacă au fost completate definitiv. Pentru a fi valide, acestea necesită să aibă completate câmpurile *Mail host name / Mail domain Name* şi *Preference*, iar în cazul în care înregistrarea nu este marcată ca externă, trebuie să fie completat şi câmpul *Mail host ip address*.

Pagina de update

Pagina de *update* are rol de a prelua detaliile despre domenii din baza de date şi afişarea lor în interfaţa web pentru ca userul să le poată modifica, sau adăuga altele. Prima data când pagina de *update* este accesată pe interfaţă sunt afişate 4 secţiuni goale şi un element de tip *select* cu domeniile disponibile în baza de date Figura 3.6.

Cele 4 secţiuni din pagină sunt:

- *Domain*
- *Name server records*
- *Host records*
- *Mail records*

Cu excepţia secţiunii *Domain*, toate celelalte secţiuni au un buton de adăugare a unei noi înregistrări. Acestu buton adăugare apelează pe evenimentul *onclick* aceiaşi funcţie de *callback* ca şi butoanele de adăugare a înregistrărilor din pagina principală, fiecare buton având o funcţie de *callback* în dependenţă de tipul înregistrării pe care urmează să o adauge.

În figura următoare este reprezentată operaţia de modificare a înregistrărilor unui domeniu:

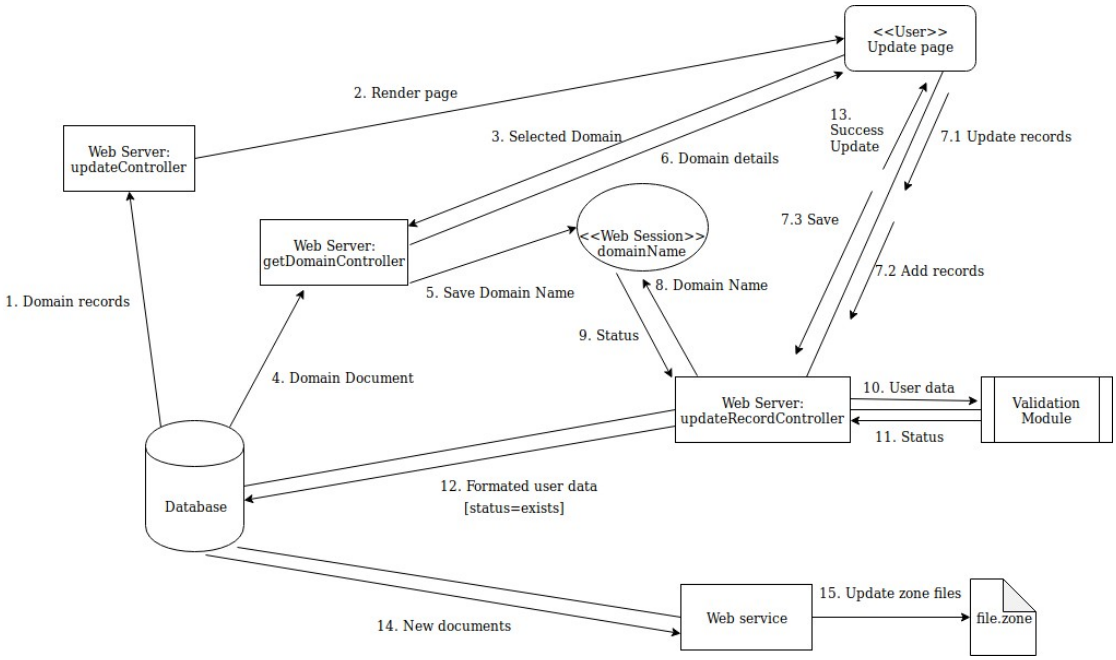

Figura 3.5: Modificare domeniu

În momentul încărcării paginii sunt trimise de la server în interfață doar numele domeniilor, acestea fiind încărcate în elementul de tip *select*. Acestui acest element îi este asociată o funcție de *callback* declanșată de evenimentul *onchange*. Acea funcție de *callback* face un apel asincron prin AJAX la server, trimițând numele domeniului și așteptând ca răspuns toate detaliile din baza de date despre domeniu. Odată primite, cu ajutorul funcțiilor de modificare a DOM-ului din JavaScript, sunt populate cele 4 secțiuni din pagină cu înregistrările corespunzătoare. Dacă în tipul apelului AJAX s-a produs o eroare, utilizatorul va fi redirecționat imediat spre o pagină cu de eroare. Butoanele de adăugare a unor noi înregistrări nu sunt active atâta timp cât nu s-a ales nici un domeniu.

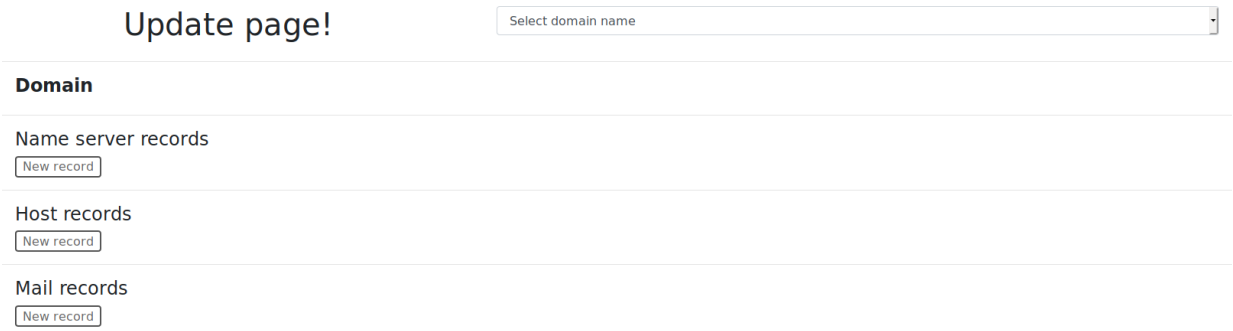

Figura 3.6: Pagina inițială de update

După ce s-au încărcat înregistrările în pagină, aceste sunt sunt doar vizibile dar nu și editabile. Secțiunea ***DOMAIN*** conține mereu același număr de câmpuri, fiecare dintre ele este completat cu datele corespunzătoare domeniului. Aceasta conține doua butoane, de ștergere și de editare. Butonul de editare determină ca toate câmpurile din secțiune să fie editabile, ceea ce înseamnă că parametrii domeniului și ai rețelei pot fi modificați. Butonul de ștergere va duce la ștergerea completă a domeniului și golirea paginii de înregistrările deja încărcate, iar toate câmpurile din cele 4 secțiuni vor dispărea.

În dreptul fiecărei înregistrări încărcate în celelalte 3 secțiuni se află la fel două butoane: ștergere și editare. Butonul de editare activează editarea pentru înregistrarea căreia îi aparține. Butonul de ștergere va șterge toate datele și câmpurile corespunzătoare înregistrării. Dar o nouă înregistrare poate fi adăugată în orice moment cu ajutorul butonului de adăugare prezent în fiecare secțiune. Spre deosebire de înregistrările generate la încărcarea detaliilor despre domeniu în pagină, înregistrările adăugate cu ajutorul butonului nu au buton de ștergere sau de editare, deci nu pot dispărea fizic din pagină. Ambele tipuri de înregistrări au mesajul de avertizare *„not complete”* pentru a atenționa *user*-ul în caz că înregistrare nu este completă și nu va fi trimisă la server Figura 3.7.

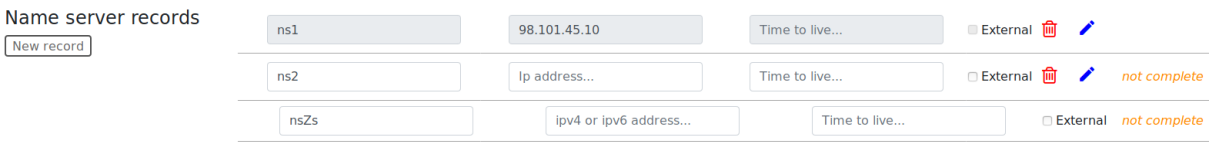

Figura 3.7: Înregistrări NS generate și adăugate

3.1.2. Implementarea pe partea de server

Componenta reprezentată de serverul web este implementată cu ajutorul *framework*-ului Flask, care la rândul său este scris în Python. Construirea unei aplicații web în Flask presupune câteva lucruri:

- crearea unei instanțe a clasei Flask
- definirea unei funcții pentru fiecare cale de rutare
- rularea aplicației

```
app = Flask(__name__)
@app.route('/')
def index():
    return render_template("index.html")
```

Analizând secvența de cod anterioară observăm ca avem creată o cale de rutare la adresa relativă '/'. Pentru acestă cale avem definită o funcție *index* care returnează un obiect *render_template* cu parametru un nume de fișier HTML, fișier care urmează a fi randat în pagină. Acel fișier ca să poată fi găsit și randat de către aplicație Flask, trebuie să se afle într-un director numit *templates*.

Dacă nu se dorește a se returna o pagină randată, ci doar un un obiect în format JSON, atunci putem folosi obiectul *make_response* cu doi parametri: obiectul în format JSON și codul de răspuns:

```
return make_response(jsonify({'data': 'my data'}), 400)
```

Serverul web scris în Python este constituit din două fișiere *app.py* și *validation.py*. *app.py* conține aplicația Flask care se ocupă de funționalitățile aplicației web cât și 2 funcții ce se ocupă de procesarea datelor, iar *validation.py* conține o clasă cu funcții statice de validare.

Aplicația Flask este constituită din mai multe rute principale, care construiesc marea parte a aplicației web, dar și din niște rute secundare care randează paginile de succes sau eroare. Funcții pentru rute principale:

- ***record*** primește datele în format JSON de la client atunci când utilizatorul încearcă să adauge un nou domeniu și apelează funcția de procesare a obiectului JSON primit. Dacă funcția de procesare returnează un status de eroare atunci serverul trimite înapoi spre client un obiect în format JSON cu toate erorile întâlnite împreună cu un cod de eroare corespunzător. Dacă nu există nici o eroare de procesare, atunci serverul salvează obiectul procesat într-o colecție a bazei de date și returnează spre client un cod de succes. Pe partea de client se face redirecționare spre o rută cu randează o pagină de succes de adăugare a unui nou domeniu.
- ***updateRecord*** este accesată de către cererea de tip AJAX făcută la salvarea datelor de pe pagina de *update*. Primește la intrare datele din interfață în format JSON. Trimite datele spre procesare aceleiași funcții pe care o utilizează metoda *record* discutată anterior. Dacă funcția de procesare returnează un status de eroare, atunci obiectul în format JSON ce conține erorile este trimis înapoi spre client și afișat în pagină. Dacă funcția returnează un status de succes atunci există două posibilități pornind din acest moment:
  1. Utilizatorul nu a modificat numele domeniului caz în care documentul din colecția bazei de date ce reprezenta domeniul vechi este șters și inserat domeniul cu informațiile noi.

2. Utilizatorul a decis să modifice numele domeniului caz în care documentul ce reprezenta domeniul cu numele vechi este marcat cu status de *delete*, ceea ce înseamnă că serverul de servicii va șterge efectiv domeniul din cadrul serverului DNS. Domeniul cu numele nou va fi inserat într-un nou document ce va duce la crearea unui domeniu complet nou în cadrul serverului DNS.

- ***update*** este funcția de preia numele tuturor domeniile din baza de date care nu au statusul *delete* și creează elementul *select* din pagina de update. La fel funcția are ca rol randare paginii *update* inițiale.
- ***getDomain*** este funcția ce este accesată de către cererea AJAX făcută la schimbarea numelui de domeniu din elementul *select* de pe interfața paginii *update*. La intrare primește numele domeniului din cadrul interfeței și returnează din baza de date toate detaliile despre domeniu, acestea fiind returnate în format JSON și încărcate în mod dinamic în pagina de *update*.

Funcția de procesare a datelor primite de la client este elementul principal în procesul de validare datelor și creare a erorilor corespunzătoare. Dacă adresa de rețea a domeniului este o adresă IPv4 atunci se așteaptă ca toate adresele ulterioare să fie adrese IPv4. Pentru adresa de rețea în format IPv6 se aplică aceiași strategie. Structura obiectului este următoarea:

```
{
    ”domain_details”:{
          ”domain_name”: string,
          ”admin_mail”: string,
          ”original_admin_name”: string,
          ”domain_ttl”: number,
          ”domain_ip_address”: string,
          ”domain_subnet”: string,
          ”record_type”: string,
          ”domain_reverse_addr”: string
    },
    ”ns_records”: [
       {
          ”ns”: string,
          ”ns_ip”: string,
          ”ns_ip_reverse”: string,
          ”ns_ttl”: number,
          ”external”: boolean
       }
    ],
    ”host_records”: [
       {
          ”host_name”: string,
          ”host_name_ip”: string,
          ”host_name_ip_reverse”: string,
          ”host_ttl”: number,
          ”host_cname”: string,
          ”host_txt”: string,
       }
    ],
    ”mail_records”: [
```

```
        {
            ”mail_host”: string,
            ”mail_ip_host”: string,
            ”mail_ip_host_reverse”: string,
            ”mail_txt”: string,
            ”mail_cname”: string,
            ”mail_ttl”: number,
            ”mail_preference”: number,
            ”external”: boolean,
        }
    ]
}
```

Se observă cele 4 elemente principale ce formează obiectul:

1. *domain_details* structură ce salvează detaliile principale ale domeniului strict necesare creării fișierelor de tip zonă corespuzătoare.
2. *ns_records* este o listă de obiecte, fiecare element din listă salvează informațiile despre o înregistrare de tip NS ce urmează a fi creată în fișierul zonă. Dacă câmpul *external* este *true*, atunci câmpurile *ns_ip* și *ns_ip_reverse* nu sunt prezente.
3. *hosts_records* este o listă de obiecte, fiecare element din listă salvează informațiile despre o înregistrare de tip A sau AAAA ce urmează a fi creată în fișierul zonă, precum și posibilele înregistrări de tip TXT și CNAME.
4. *mails_records* este o listă de obiecte, fiecare element din listă salvează informațiile despre o înregistrare de tip MX ce urmează a fi creată în fișierul zonă, precum și posibilele înregistrări de tip TXT și CNAME. Dacă câmpul *external* este *true*, atunci câmpurile *mail_ip_host* și *mail_ip_host_reverse* nu sunt prezente.

Fișierul *validation.py* conține o clasă cu metodele statice necesare pentru validarea unor câmpuri.
Metode ale clasei *Validation Anexa 1. :*

- *check_mail* primește ca parametru o șir de caractere și verifică dacă acesta reprezintă o adresă de e-mail; dacă nu, aruncă o excepție.
- *check_host_name* primește ca parametri două șiruri de caractere, primul parametru fiind elementul ce trebuie validat, al doilea parametru fiind tipul de parametru necesar formării mesajului din interiorul excepției.
- *check_domain_name* la fel ca și metoda anterioară primește ca parametri două șiruri de caractere, primul parametru fiind elementul ce trebuie validat, al doilea parametru fiind tipul de parametru necesar formării mesajului de excepție.
- *check_ttl* primește ca parametri două șiruri de caractere, primul parametrul este elementul ce ar trebuie să reprezinte un număr; al doilea parametru este valoare maximă permisă pentru acel număr. La fel trebuie să fie un număr natural.

3.2. Implementare Server de Servicii

Componenta reprezentată de serverul de servicii este a cea care are rolul principal dintre conectarea datelor primite de la utilizator și serverul DNS BIND9. Serverul de servicii se află pe aceiași mașină cu serverul DNS (CentOS 7). Serverul web și serverul de baze de date MongoDB se află cealaltă mașină (Ubuntu 18.04). Conectarea dintre serverul de servicii și serverul de baze de date se face în cadrul unei rețele interne private în cazul aplicației testate de mine. La fel de

bine conectarea se poate face prin cadrul unei rețele publice, necesară fiind doar adresa IP publică a serverului MongoDB.

Serverul de servicii are ca rol citirea temporară din baza de date a ultimelor modificări făcute și luarea unor decizii în funcție de statusul documentelor citite. Un document poate avea unul dintre următoare 2 statusuri:

- *insert* ce duce la crearea unui nou domeniu
- *delete* ce duce la ștergerea unui domeniu

Componenta dată este constituită dintr-un singur modul ce conține câteva funcții necesare creerii sau ștergerii unui domeniu:

1. *integrate_zone*. Primește ca parametru documentul din baza de date corespunzător unui domeniu. Creează un director cu numele domeniului la adresa ***/var/named/***. În interiorul acestui director urmează să fie create cele două fișiere de zonă corespunzătoare domeniul, fișierul zonă de mapare directă și fișierul zonă de mapare inversă. Se apelează funcțiile de creare a celor două fișiere. După, creează un fișier de configurare cu numele dat de formatul *nume_domeniu.conf* la adresa ***/etc/named/***. Conținutul se poate vedea *Anexa 2.* O referință la fișierul dat este inclusă în fișierul ***/etc/named.conf***. În figura următoare se poate vedea schema de inserare a unui domeniu:

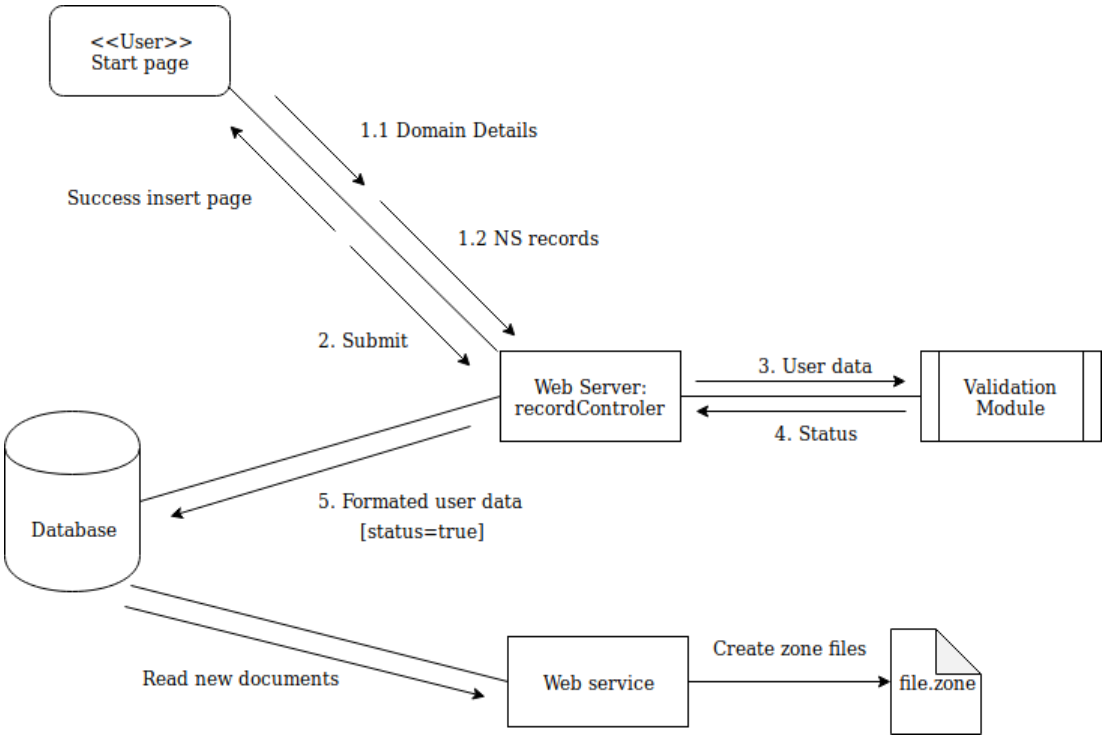

Figura 3.8: Inserare domeniu

2. *create_direct_zone_file*. Funcție ce primește ca parametru documentul din baza de date corespunzător unui domeniu și creează pe baza lui fișierul de tip zonă cu numele ***nume_domeniu.zone*** în directorul creat special pentru domeniu de către funcția *integrate_zone*. Se parcurg cele 4 elemente cheie ale obiectului și se creează succesiv înregistrări pe baza lor. Parcurgerea principalelor câmpuri din interior documentului:
   - Câmpurile din *domain_details* sunt utilizate pentru creare directivelor $ORIGIN și $TTL precum și a singurei înregistrări de tip SOA.

- Elementele din lista *ns_records* reprezintă câte o înregistrare de tip NS. Dacă acea înregistrare nu are adresă IP, atunci este considerată ca înregistrare ce face referire la un Server de Nume exterior domeniului. Dacă acea înregistrare are o adresă IP, atunci este considerată adresă din cadrul domeniului şi se mai creează o înregistrare de tip A sau AAAA pentru adresa IP.
- Elementele din lista *hosts_records* fac referire la un nume de *host* din interiorul domeniul. Dacă câmpul TXT sau CNAME nu este null, atunci pe lângă înregistrare de tip A sau AAAA creată pentru adresa *host*-ului, se mai creează şi câte o înregistrare de tip CNAME sau TXT ce fac referire la înregistrarea de tip A sau AAAA creată anterior.

```
www     IN     A      192.56.101.7
ww3     IN     CNAME  www
www     IN     TXT    'Some information about www record!'
```

- Elementele din lista *mails_records* reprezintă câte o înregistrare de tip MX. Dacă înregistrare curentă are o adresă IP atunci este considerată parte a domeniului şi pe lângă înregistrare de tip MX, mai este creată o înregistrare de tip A sau AAAA ce face referire la înregistrare de tip MX creată anterior. Dacă câmpul TXT sau CNAME nu este null, atunci pe lângă înregistrare de tip A sau AAAA creată pentru referire înregistrării MX, se mai creează şi câte o înregistrare de tip CNAME sau TXT ce fac la referire la înregistrarea de tip MX creată anterior.

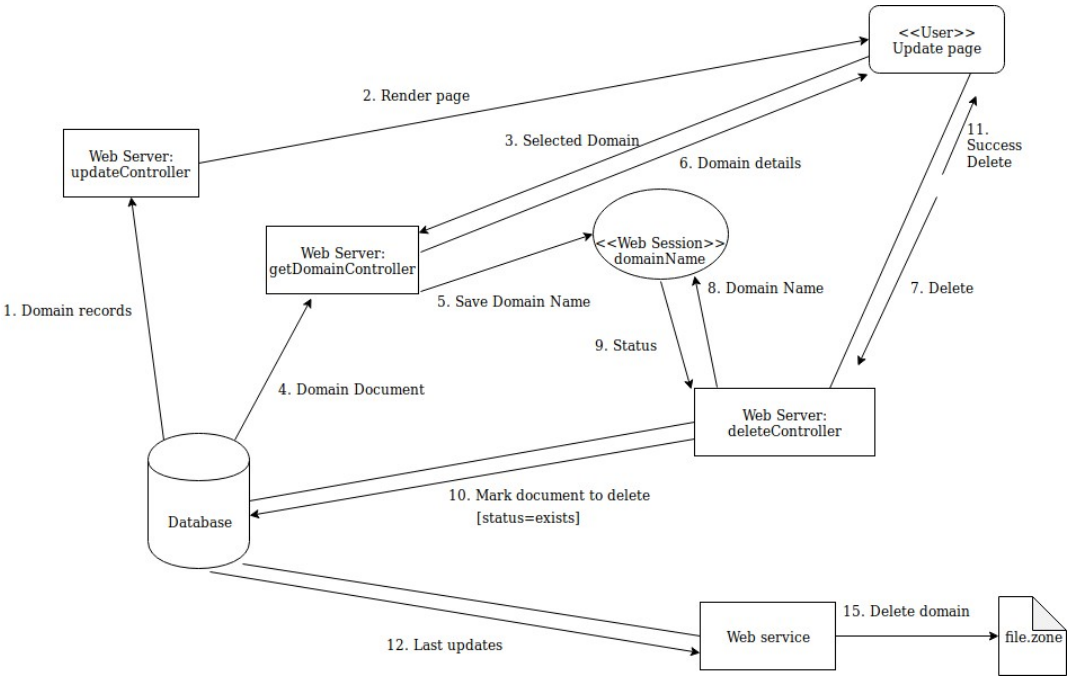

Figura 3.9: Ştergere domeniu

3. *create_reverse_zone_file*. Funcţia primeşte întregul document specific domeniul. Creează fişierul de tip zonă cu numele ***nume_domeniu.rr.zone*** în directorul creat special pentru domeniu de către funcţia *integrate_zone*. Se creează şi în acest fişier directivele $ORIGIN şi $TTL precum şi înregistrarea de tip SOA folosind adresă de mapare inversă a domeniul stocată în interior *domain_details* sub numele *domain_revers_addr*.

Pentru fiecare dintre înregistrările din *ns_records*, *hosts_records* și *mails_records*, dacă acele înregistrări nu sunt externe domeniului, se creează câte o înregistrare în fișierul de mapare inversă folosind adresă de mapare inversă pentru fiecare înregistrare în parte.

4. *delete_domain.* Primește documentul unui domeniului ca parametru. Șterge directorul cu cele 2 fișiere de tip zonă corespunzătoare acestuia de la adresa ***/var/named/nume_domaniu***. Ștergere fișierul ***/etc/named/nume_domeniu.conf*** după care șterge referința spre acel fișier din fișierul de configurare a serverului DNS ***/etc/named.conf***. În Figura 3.9 se poate vedea modul de ștergere a unui domeniu.
5. Funcția principală *main* rulează într-o buclă infinită interogând la un interval specific de tip serverul de baze de date și extrăgând modificările apărute de la ultima interogare. Pentru fiecare document extras din baza de date se verifică statusul. Dacă are status *insert*, atunci se apelează funcția *integrate_zone*, dacă are status *delete*, atunci se apelează funcția *delete_domain.* După fiecare procesare a unui document, se restartează serverul DNS BIND9 și se continuă cu următoare înregistrare.

Capitolul 4. Testare

Testarea aplicației a fost făcută pe componente. Cele 2 componente de bază fiind serverul web și serverul de servicii. Serverul de servicii a fost testat separat de serverul web. Serverul web a fost testat pe parcursul dezvoltării în 3 direcții diferite:

1. Testare pe partea de client
2. Testarea legăturii client-server
3. Testarea pe partea de server

4.1. Instalarea și configurarea aplicației

Aplicația a fost testată pentru o singură configurație de instalare. Cele 2 componente principale au fost instalate pe 2 mașini diferite. Serverul web a fost instalat împreună cu serverul de baze de date MongoDB pe o mașină cu sistemul de operare Ubuntu 18.04. Serverul de Servicii împreună cu Serverul DNS BIND9 a fost instalat pe o mașină virtuală cu sistem de operare CentOS 7. Pentru crearea unei mașini virtuale s-a folosit aplicația desktop *Oracle VM Virtual Box 6.0* cu ajutorul căreia am creat o rețea privată, plasând mașina gazdă Ubuntu 18.04 și mașina virtuală CentOS 7 în cadrul acelei rețele.

Instalarea serverului web necesită instalarea unor dependințe listate în fișierul *requirements.txt Anexa 3.* Serverul web este scris în Python3.6.2. Pentru instalarea bibliotecilor din fișierul de dependințe se rulează comanda:

```
pip install -r requirements.txt
```

Instalarea serverului de servicii necesită la fel instalarea dependințelor listate în fișierul *requirements.txt Anexa 3.* Instalare care se face executând în terminal linia de cod de mai sus. Tot de ce este nevoie, fiind Python3.6.2.

Serverul de baze de date MongoDB este instalat pe aceiași mașină cu serverul web. Pentru o instalarea completă se poate urmări tutorialul de la adresa dată [19]. În cazul meu am folosit instalarea pentru versiunea MongoDB Enterprise pentru Ubuntu. Serverul MongoDB urmează să fie accesat de cele 2 componente ale aplicației, ceea ce înseamnă că serverul web va trebuie să acceseze serverul MongoDB precizând ca adresă IP a acestuia *127.0.0.1* sau *localhost.* Serverul de Servicii nu se află pe aceiași mașină cu serverul MongoDB și va trebuie să precizeze adresa acestuia din rețeaua privată creată anterior, adică adresa IP a mașinii Ubuntu 18.04. Odată cu instalarea implicită a serverului MongoDB, nu este permisă accesarea acestuia de pe altă adresă IP decât *localhost.* Pentru a permite conectarea la serverul MongoDB de pe alte mașini, va trebuie să specificăm în fișierul de configurări al serverului */etc/mongo.conf,* adresa IP a mașinii pe care rulează serverul MongoDB, ca în exemplul următor:

```
bind_ip = 127.0.0.1,192.168.56.1
```

A doua adresă IP din listă este adresa mașinii Ubuntu din cadrul rețelei private în cazul dat. După modificările făcute trebuie restartat serverul MongoDB:

```
systemctl restart mongodb
```

Pentru a porni serverul web se va rula comanda:

```
python3 app.py
```

Pentru a porni serverul de servicii va fi nevoie de schimbarea adresei IP a serverului MongoDB din fișierul *service.py* a aplicației, line de cod ce se află în antetul fișierului.

```
client = pymongo.MongoClient('mongodb://IP_ADDRESS:27017/')
```

Pentru a porni serverul de servii se va rula comanda de mai jos:

```
python3 service.py
```

Instalarea serverului BIND9 pe mașina virtuală CentOS 7 se va face urmărind tutorialul [20].

4.2. Testare server web

4.2.1. Testarea pe partea de client

Testarea clientului web a fost făcută cu scopul de a depista unele erori de implementarea, dar și pentru a încerca să se introducă unele funcționalități noi. Dezvoltarea unei interfețe dinamice precizată în 3.1.1 implică o muncă complicată cu DOM-ul, unde pot mereu apărea probleme cu accesarea elementelor în pagină, modificarea, ștergerea și chiar inserarea unora noi.

Testarea validării câmpurilor

După cum precizasem în capitolul de implementare, unele câmpuri au fost validate pe partea de client cu ajutorul atributului *type* din HTML. Pentru câmpurile numerice s-a folosit tipul *number* iar pentru intervalul numeric dorit s-a folosit atributele *min, max.* Pentru câmpurile de tip e-mail s-a folosit tipul *email.* Alte tipuri deja existente din HTML nu am fost nevoit sa folosesc. Pentru a face niște validări care să corespundă unui anumit șablon, am folosit atributul *pattern* din elementul *input.*

Pentru a valida o etichetă simplă am folosit șablonul:

```
pattern="^(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\-]*[a-zA-Z0-9])\.)*([A-Za-
z]|[A-Za-z][A-Za-z0-9\-]*[A-Za-z0-9])$"
```

Pentru a testa un nume de domeniu complet, cum ar fi câmpurile din interfață ce reprezintă numele de domeniu, numele Serverului de Nume extern sau numele serverului de mail extern a fost folosit următorul șablon:

```
pattern="[a-zA-Z0-9][a-zA-Z0-9-]{1,61}[a-zA-Z0-9](?:\.[a-zA-Z]{2,})+"
```

Testarea funcționalității a fost făcută prin testare manuală. Au fost introduse valori care corespund șablonului și sa verificat ca validarea este corectă, după au fost introduse valori care nu corespund șablonului pentru a vedea dacă validarea nu a reușit.

Testarea inserării dinamice a câmpurilor în pagină

Am precizat anterior că la încărcarea paginii utilizatorului îi sunt disponibile doar câmpuri pentru câte o singură înregistrare de fiecare tip. Dace acesta dorește mai inserarea mai multor înregistrări atunci are la dispoziție un buton pentru adăugare dinamică a acestora. Adăugare dinamică a unor elemente în pagină presupune câteva lucruri:

- localizarea poziției sau structurii unde urmează a fi inserate
- crearea elementelor și atașarea lor la elementele părinte descrise la pasul anterior

- adăugarea de funcţii ce doresc a fi executate la apariţia unor evenimente dacă este cazul.

Toate testările au fost manuale ca şi în cazul anterior. Testarea localizării s-a făcut prin folosirea consolei din JavaScript, mesajele fiind afişate în consola din browser. Trimiţând un element HTML la consolă, acesta era afişat într-o structură ierarhică cu toţi descendenţii săi. Aceiaşi mod de testare s-a folosit şi în etapa de creare şi ataşare a elementelor noi la elementele părinte. În consola din browser se putea urmări structura elementului părinte şi dacă acesta are în componenţa sa noile elemente ataşate sau nu. Pentru a testa dacă evenimentele dorite ale elementelor noi inserate declanşează executarea unor funcţii, s-a folosit testarea manuală prin crearea de condiţii necesare declanşării unui astfel de eveniment.

4.2.2. Testarea legăturii client-server

În acest secţiune se vor specifica detalii de testare a conexiunii dintre partea de client şi cea de server. Datele de la client sunt transmise la server prin cereri de tip AJAX. Interfaţa web face conexiunea cu serverul web prin 3 evenimente. Apăsarea butonului de *submit* de pe pagina de start. Apăsarea butonului *save* şi *delete* din pagina de *update*. Toate cele 3 evenimente duc la crearea unor obiecte în format JSON şi trimiterea acestor la server prin AJAX. Testarea fost făcută manual cu ajutorul consolei din JavaScript. Înainte de a trimite obiectul JSON către server acesta era afişat în consola browserului unde i se putea urmări structura. La recepţionare pe partea de server obiectul primit era la fel afişat în consolă, astfel se putea urmări dacă obiectul a ajuns la server, s-a făcut extragerea obiectului corect din corpul mesajului HTTP şi dacă obiectul de la server corespunde cu cel trimis de client.

Circularea datelor client-server se face bidirecţional, astfel dacă pe partea de server apar erori de validare a datelor, se trimite un obiect în format JSON la client cu erorile date. La fel testarea se face cu ajutorul consolei din browser şi consolei de la server.

4.2.3. Testarea pe partea de server

Toate testele efectuate pe partea de server sunt manuale. S-a testat mai întâi în browser pe adresa de *localhost* dacă implementările tuturor rutelor funcţionează corespunzător şi dacă în pagina din browser sunt randate paginile HTML corecte. La fel în browser s-a testat dacă se face redirecţionarea corectă în caz de accesarea unor rute inexistente sau prin metode nepermise (POST sau GET). Testarea funcţionalităţii funcţiilor de validare a fost făcută atât unitar, fiecare funcţie în parte fiind testată în consola serverului, cât şi per ansamblu, trimiţând obiectul în format JSON către client şi afişând erorile în pagină. Se verifică dacă erorile primite de la server corespund cu câmpurile completate incorect în interfaţă înainte de trimitere.

4.3. Testare server de servicii

Testarea serverului de servicii s-a făcut separat de testarea serverului web. Testarea conexiunii cu serverul de baze de date: dacă conexiune eşuează se emite un mesaj de eroare în consolă. Testarea creării directorului pentru fiecare domeniu se face manual prin accesare locaţiei corespunzătoare. La fel manual se face şi testarea creării tuturor fişierelor necesare pentru un domeniu: prin deschiderea manuală a acestuia şi prin urmărirea structurii fişierului. Testarea faptului că serverul s-a restartat corect se testează la fel în consolă. În caz de excepţie se afişează un mesaj corespunzător dar şi mesajul excepţiei. Pentru a testa dacă domeniul a fost creat corect şi toate înregistrările funcţionează corespunzător, s-a folosit clientul de creare a interogărilor DNS, ***dig***[21] trimiţând cereri DNS de pe maşina primară (Ubuntu 18.04) către serverul BIND9 cu fiecare înregistrare din fişierele de zonă.

Concluzii

Pentru crearea unei astfel de aplicații am avut nevoie de o documentație detaliată a modului de funcționare a unui server DNS, dar și a modului de funcționare a internetului în general. Principala sursă de documentație referitoare la serverul DNS folosit în aplicația mea, BIND9 poate fi găsită aici [3]. Înțelegerea modului de funcționare a serverului BIND9 a fost vitală în dezvoltarea proiectului.

Aplicația dată poate fi utilă oricărei persoane ce încearcă să-și proiecteze o rețea internă în cadrul unei companii, instituții sau chiar acasă. Totodată aplicația poate fi utilă celor care doresc să facă singuri managementul domeniului deținut și să controleze total serverul DNS autoritativ pentru domeniul dat. Dacă domeniul este relativ mic, s-ar putea să nu se observe impactul aplicației, în comparație cu managementul serverului DNS în mod manual, dar odată ce domeniul se extinde și capătă dimensiuni mari, randamentul dat de aplicație va crește semnificativ.

La proiectarea aplicației am folosit ca limbaj de bază Python 3.6. Am ales acest limbaj în primul rând pentru am ușura munca ca programator, am dorit să investesc mai mult timp proiectării și implementării aplicației, încercării folosirii unui limbaj strict orientat obiect care implică scrierea a mai mult cod ce face același lucru. Python e strict orientat spre programator permițând efectuarea multor acțiuni cu secvențe foarte scurte de cod. La fel pentru ușurarea muncii și reducerea secvențelor de cod scrie pe partea de client web am folosit librăria jQuery în loc de JavaScript simplu. Acest lucru a implicat mărirea timpului de încărcare a paginii, dar nu foarte semnificativ, fapt ce a meritat să merg la un așa compromis, ingineria fiind știința compromisului.

Modelarea serverului web a fost gândită de către mine, fără nici o sursă de inspirație. La fel și ideea generării dinamice a înregistrărilor din interfața web ce permite adăugarea unui număr înregistrări dorit de utilizator. Structura din interfață unei înregistrări a fost inspirată din aplicații asemănătoare, dar acea structură nu prea îți oferă orizont de imaginație foarte larg, fiind o structură standard.

Un punct de plecare pentru o dezvoltarea ulterioară a aplicației ar fi crearea unei extensii pentru a putea controla mai ușor conținutul fișierelor zonă. Modul de abordare a problemei din proiectul dat a fost că în cazul unei încercări de modificare a unui domeniu, fișierele zonă de mapare directă și mapare inversă a adreselor IP erau șterse definitiv și create altele noi. Acest lucru este destul de costisitor ca timp. O mod de îmbunătățire a acestui lucru ar fi localizarea înregistrării sau înregistrărilor ce se doresc a fi modificate și rescrierea lor, fără a modifica restul conținutului.

O altă îmbunătățirii în direcția securității ar fi crearea unui modul de autentificare pentru serverul web și pentru serverul de servicii. Pentru serverul de web se poate crea o procedură de autorizare pe roluri. Unele tipuri de utilizatori nu au voie să modifice anumite detalii despre unele domenii sau să modifice unei domenii în general. Acest lucru ar garanta un nivel de securitate mult mai ridicat decât cel oferit de aplicația curentă.

Internetul constituie o parte majoră a vieții din ziua de azi, iar una dintre principalele componente din care internetul este constituit, este Sistemul de Nume de Domeniu. La momentul dat nu avem de ce să căutăm o altă alternativă acestui sistem de dirijare a traficului pe internet, dar nimeni nu ne oprește să îmbunătățim interacționarea noastă ca ingineri cu DNS-ul. Crearea unor noi instrumente de management al serverelor DNS nu face decât să ne ușureze munca și să ne rezerve mai mult timp pentru alte inovații.

Bibliografie

[1] ARPANET Resursa se află la adresa: https://icannwiki.org/ARPANET

[2] RFC 1034 Resursa se află la adresa: https://www.ietf.org/rfc/rfc1034.txt

[3] ZYTRAX Resursa se află la adresa: http://www.zytrax.com/books/dns/

[4] ICANN Resursa se află la adresa: https://www.icann.org/ids

[5] RFC 1035 Resursa se află la adresa: https://www.ietf.org/rfc/rfc1035.txt

[6] RFC 3425 Resursa se află la adresa: https://tools.ietf.org/html/rfc3425

[7] IANA Resursa se află la adresa: https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml

[8] RFC 2317 Resursa se află la adresa: https://tools.ietf.org/html/rfc2317

[9] Round-Robin Resursa se află la adresa: https://www.dnsknowledge.com/whatis/round-robin-dns/

[10] RFC 3596 Resursa se află la adresa: http://www.zytrax.com/books/dns/apd/rfc3596.txt

[11] Înregistrare SOA Resursa se află la adresa: http://www.zytrax.com/books/dns/ch8/soa.html

[12] RFC 1912 Resursa se află la adresa: http://www.zytrax.com/books/dns/apd/rfc1912.txt

[13] RFT 2308 Resursa se află la adresa: http://www.zytrax.com/books/dns/apd/rfc2308.txt

[14] Flask Resursa se află la adresa: http://flask.pocoo.org/

[15] Jinja2 Resursa se află la adresa: http://jinja.pocoo.org/

[16] JavaScript Adresă resursă: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

[17] MongoDB Resursa se află la adresa: https://docs.mongodb.com/

[18] jQuery Resursa se află la adresa: https://api.jquery.com/

[19] Instalare MongoDB Resursa se află la adresa: https://docs.mongodb.com/manual/installation/

[20] BIND9 Instalare Resursa se află la adresa: http://linuxpitstop.com/dns-server-setup-using-bind-9-on-centos-7-linux/

[21] Client DNS, dig Resursa se află la adresa: https://linux.die.net/man/1/dig

Anexe

Anexa 1. Metode de validare

Validare adresă e-mail:

```
def check_email(email: str) -> str:
    REGEX_email = '^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9+)*(\.
                  [a-z]{2,4})$'
    email = str(email)
    if re.match(REGEX_email, email):
        return email.replace('@', '.')
    raise Exception("\'{}\' is not a valid email address!".format(email))
```

Validare etichetă:

```
def check_host_name(name: str, input_type: str) -> str:
    name = str(name)
    REGEX_label = r'[a-zA-Z90-9]([a-zA-Z90-9-]{0,61}[a-zA-Z90-9])?'
    pattern = r'^{0}(\.{0})*\.?$'.format(REGEX_label)
    if re.match(pattern, name):
        return name
    raise Exception("\'{0}\' is not a valid {1}!".format(name,input_type))
```

Validare nume de domeniu complet:

```
def check_domain_name(name: str, input_type: str) -> str:
    REGEX_domain = '^([a-z0-9]+(-[a-z0-9]+)*\.)+[a-z]{2,}$'
    name = str(name)
    if re.match(REGEX_domain, name):
        return name
    raise Exception("\'{0}\' is not a valid {1}!".format(name,input_type))
```

Validare șir de caractere ce reprezintă un număr:

```
def check_ttl(ttl: str, max_val: int) -> int:
    try:
        ttl = int(ttl)
    except:
        raise Exception("\'{}\' must be an integer value!".format(ttl))
    if ttl in range(max_val + 1):
        return ttl
    raise Exception("TTL must be in [0, {0}] range!".format(max_val))
```

Anexa 2. Fişiere de configurare zonă

Conţinutul fişierului ***/etc/named/nume_domeniu.conf***

```
zone ‘domain_name’ {
    type master;
    file ‘var/name/domain_name/domain_name.zone’;
};

zone ‘reverse_domain_ip_address.in-addr.arpa’ {
    type master;
    file ‘var/name/domain_name/domain_name.rr.zone’;
};
```

Modul în care fişierul anterior este inclus in fişierul de configurare ***/etc/named.conf***

```
include "/etc/named/nume_domeniu.conf"
```

Anexa 3. Fişiere de dependinţe

Fişierul de dependinţe *requirements.txt* pentru serverul web:

```
Flask==1.0.2
pymongo==3.7.2
ipaddress==1.0.22
```

Fişierul de dependinţe *requirements.txt* pentru serverul de servicii:

```
pymongo==3.7.2
```