UNIVERSITATEA “BABEȘ-BOLYAI” CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ ROMÂNĂ

LUCRARE DE LICENȚĂ

SERVICII WEB ȘI ACCESAREA ACESTORA PRIN INTERMEDIUL APLICAȚIILOR MOBLIE

Coordonator științific,
Lect. Ph.D. Radu D. GĂCEANU

Absolvent,
Raul-Constantin PAROȘ

Cluj-Napoca
2018

Cuprins

Introducere

În orașele cu un grad de ocupare ridicat, locurile de parcare din zonele de interes devin o resursă pe care oamenii și autoritățiile trebuie să o gestioneze inteligent. Astfel că, un studiu [1] arată că într-un oraș ca New York-ul, oamenii pierd în medie aproximativ o sută de ore pe an căutând un loc de parcare liber.

Acest lucru aduce daune cuantificabile din punctul de vedere al timpului pierdut, banilor irosiți pe combustibilul consumat sau a emisiilor de CO2 din atmosferă care se produc în plus. Studiul arată media de ore din cele mai aglomerate 10 orașe din Statele Unite ale Americii și calculează o sumă aproximativă care reprezintă daunele estimate pentru aceste orașe. Aceasta se ridică la 72,7 miliarde de dolari pentru anul 2017, anul în care compania INRIX a efectuat studiul. De asemenea căutarea unui loc de parcare într-un interval orar în care traficul este deja congestionat duce la o îngreunare suplimentară datorată faptului că pentru a putea găsi un loc de parcare pe o stradă, șoferul trebuie să încetinească pentru a vedea din timp locul liber și să oprească traficul pentru o perioadă pentru a putea parca. Toate aceste lucruri fac locurile de parcare să fie o resursă indispensabilă a secolului XXI care trebuie gestionate prin sisteme inteligente pentru a putea fi maximizată eficiența folosirii lor.

Desigur, există deja aplicații pe piață care abordează și încearcă să rezolve problema aceasta, însă soluția propusă de noi abordează problema diferit de celelalte soluții de pe piață din cauză că majoritatea se focusează pe optimizarea gestiunii locurilor de parcare din parcările private aflate într-un oraș sau parcările din aeroporturi. Aplicația dezvoltată în cadrul acestei lucrări se concentrează pe eficientizarea folosirii locurilor de parcare din parcările publice de pe străzi, sau parcările private care rămân libere în timpul zilei. Numărul acestora este mult mai mare decât cel al parcărilor private iar costul parcării este de obicei mult mai mic.

Noi propunem o soluție care se bazează pe datele oferite de utilizatori cu privire la parcările libere folosite în general de ei sau chiar parcările private pe care le dețin dar pe care nu le folosesc o anumită perioadă din zi. Aceste locuri de parcare pot fi astfel rezervate de către șoferi, astfel că un șofer care a rezervat în prealabil un loc de parcare folosind aplicația, va beneficia de economisirea timpului și combustibilului.

Aplicația este dezvoltată ca un serviciu pentru a oferi posibilitatea prezentării informațiilor stocate pe mai multe platforme și a unei eventuale integrări cu aplicații care acoperă aceiași arie. Astfel că aplicația constă de fapt dintr-un servicu REST (Representational State Transfer) și o aplicație

dezvoltată pentru telefoanele mobile care oferă informațiile șoferilor în timp real despre locurile de parcare din zona lor de interes.

Aceste componente vor fi prezentate în următoarele capitole ale lucrării care vor detalia conceptele, ideile și tehnologiile folosite în realizarea acestei aplicații și de asemenea vom prezenta comparații cu anumite aplicații care abordează aceiași problemă.

1. Servicii Web

The World Wide Web (WWW) este deseori confundat cu Internetul din cauza cuplării strânse a celor două concepte, însă Internetul este, după cum și numele sugerează, o rețea globala de dispozitive și alte rețele interconectate.

Ne vom referi de aici înainte în această lucrare la World Wide Web pe scurt, Web sau www. Astfel că web-ul este un model, un spațiu informațional prin care elementele de interes, numite și resurse, sunt identificate prin identificatoare globale denumite Uniform Resource Identifier (URI). Informațiile sunt legate între ele prin hiperlegături (hyperlinks), care reprezintă noduri logice prin care se face navigarea între resurse. Astfel că, protocolul de comunicare în sistemul web este Hypertext Transfer Protocol (HTTP), protocol care are ca scop transferul de hipertext (hypertext), text care conține hiperlegături și care facilitează navigarea structurată între resurse prin accesarea URI-urilor acestora.

În continuare vom prezenta pe scurt protocolul HTTP și conceptul de aplicație web pentru a putea crea un context în care putem discuta despre conceptul de interes din acest capitol, și anume Servicii Web și în special Servicii Web RESTful.

1.1 Protocolul HTTP

Protocolul HTTP a fost inițiat de către Tim Berners-Lee, același om care a propus și inventat pentru prima oară spațiul Web și primul browser web. Este protocolul de transfer utilizat pe Web și se află la nivelul aplicație al stivei TCP/IP (Transmission Control Protocol/Internet Protocol). Conceptele fundamentale ale protocolului HTTP sunt cererea și răspunsul, acțiuni ce se realizează de către client, respectiv server pentru accesarea, modificarea, înlocuirea sau ștergerea resurselor. Entitățile angajate într-o astfel de comunicare se pot denumi astfel client și server. Clientul este entitatea ce inițiază comunicarea cu o cerere care se poate descrie ca un mesaj cu o structură prestabilită care să specifice detalii ca spre exemplu: tipul de operație pe care clientul dorește să o realizeze pe o anumită resursă, datele necesare pentru operație și uri-ul la care se găsește resursa. Serverul este cel care procesează aceste cereri și răspunde clientului cu un mesaj care are de asemenea o formă prestabilită și care conține detalii despre procesarea cererii cum ar fi un cod de stare prin care

serverul descrie starea cererii pe care a trebuit să o proceseze, cum ar fi dacă cererea a putut sau nu fi procesată cu succes. De asemenea răspunsul poate conține, în cazul în care serverul a reușit sa proceseze cu succes cererea, resursa vizată de către client fie ea o pagină HTML (Hypertext Markup Language), un fișier video, un fișier XML (Extensible Markup Language), un fișier audio, etc.

Faptul că un server poate răspunde cu toate aceste tipuri de date unui client a făcut protocolul HTTP să fie versatil și să corespundă unei game variate de cerințe și nevoi având un caracter generic pe care s-au putut dezvolta cu timpul alte paradigme, ca spre exemplu subiectul nostru de interes din acest capitol, serviciile Web. Acest caracter generic a devenit o bază puternică pentru inovațiile viitoare care au urmat și care s-au putut baza pe un protocol de transfer matur, puternic formalizat și gestionat de către o entitate internațională, World Wide Web Consortium (W3C).

În general, comunicarea dintre un server și un client este inițiată de către client printr-o cerere care are o structură ca în figura de mai jos.

```
GET /catalog_produse.html HTTP/1.1
Host: www.portocale.info
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US;
rv:1.8.0.5) Gecko/20060719 Firefox/1.5.0.5
Accept: text/html, image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate, compress, identity
Connection: Keep-Alive
```

Fig. 1 //todo: Change this

În această imagine se pot identifica elementele componente ale unei cereri. Prima informație care se pune în acest tip de mesaj este metoda de acces. Aceste metode sunt foarte importante în contextul serviciilor web, deoarece corespund cu acțiunile pe care un server le poate efectua asupra unei resurse și descriu astfel logica pe care serverul o poate aplica unei resurse. Deoarece pentru foarte mult timp web-ul a fost folosit doar în interacțiunile om-mașină, cele mai folosite metode sunt metoda GET și POST. Aceste două metode sunt întâlnite uzual în comportamentul browserelor Web care de obicei afișează o pagină Web folosind o cerere cu metoda GET specificată, sau care trimite unui server datele transimse de utilizator printr-o cerere cu metoda POST specificată. În contextul serviciilor web, care caută să faciliteze comunicarea mașină-mașină, se folosesc și alte metode care specifică

comportamente diferite cum ar fi: PUT, DELETE, HEAD, TRACE, CONNECT, OPTIONS.

Figura de mai jos reprezintă structura răspunsurilor pe care serverul le poate trimite înapoi unui client după procesarea cererii.

```
GET /catalog_produse.html HTTP/1.1
Host: www.portocale.info
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US;
rv:1.8.0.5) Gecko/20060719 Firefox/1.5.0.5
Accept: text/html, image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate, compress, identity
Connection: Keep-Alive
```

Fig. 2 //todo: Change this

În răspuns se pot identifica elementele importante precum codul de stare și dacă e cazul, tipul conținutului din răspuns, lungimea răspunsului și resursa propriu zisă care e reprezentată de conținutul/corpul răspunsului.

Dacă atunci când folosim un browser să accesăm pagini Web noi, utilizatorii, nu suntem interesanți de aceste detaliile deoarece browser-ul le gestionează pentru noi, atunci când vorbim de servicii Web toate aceste elemente sunt foarte importante din cauză că orice variație poate schimba drastic modul în care serverul cu care dorim să comunicăm va acționa sau va răspunde. Astfel că putem deduce că în mod uzual, capacitățiile protocolului HTTP nu sunt puse în valoare atunci când este vorba de accesarea paginilor Web de către un utilizator. Asta pentru că aceste mesaje nu au fost implementate cu scopul de a fi ușor înțelese de către oameni, iar comunicarea cu serverul chiar dacă este posibilă prin manipularea acestor mesaje, nu este facilă. De aceea, în acest tip de comunicare, interacțiunea între utilizator și server trece prin interfața expusă de către browser, care devine implicit clientul interacțiunii. Însă acest model de comunicare puternic formalizat și structurat este benefic atunci când clientul este de fapt un alt server care se va folosi de structura mesajului de cerere pentru a specifica toate detaliile interacțiunii cu datele care se găsesc pe celălalt server. De asemnea, același model oferă serverului care răspunde la cerere atât un anumit grad de libertate și personalizare, cât și limite bine definite în care trebuie să funcționeze pentru ca celălalt server, cel care are rol de client să poată să interpreteze și să folosească răspunsul.

Vreau să remarc și să accentuez acest echilibru necesar și foarte bine construit, între

personalizarea permisă și rigiditatea impusă de acest protocol, deoarece eu consider că este unul din motivele pentru care aceste servicii Web s-au putut dezvolta și au putut beneficia de tehnologiile deja dezvoltate.

Un alt aspect important al protocolului de comunicare HTTP este că acesta dispune de un caracter state-less, adică acesta deservește cereri concurente multiple de la diferiți clienți, insă fiecare cerere e considerată independentă de celelalte chiar dacă provin de la același client Web. Lucrul acesta duce la implementarea unor sesiuni la nivelul aplicațiilor web care vor asocia clienților care accesează serverul un identificator unic de sesiune (Session Id). Acest lucru duce la posibilitatea păstrării datelor de-a lungul mai multor cereri succesive ale aceluiași client.

1.2 Servicii Web. Descrierea problemei

Odată cu intrarea într-o eră a interconectivității, o eră în care comunicarea la distanță și informarea în timp real sunt două activități cruciale pentru buna funcționare a societății, dezvoltatorii au realizat că standardele industriei nu pot susține creșterea accelerată de care se bucura Internetul și spațiul Web. Soluțiile de care era nevoie pentru a suporta un nivel crescut de scalabilitate, de aplicații care trebuiau să fie prezente pe mai multe platforme și slab conectate nu se puteau mapa pe tehnologiile și paradigmele vremii.

Primele aplicații Web constau în pagini cu informații care erau conectate cu alte pagini cu informații asemănătoare pe care utilizatorul le putea folosi și prin care putea naviga cu ușurință datorită hyperlegăturilor. Primul site Web a fost creat în 1991 și afișa informații despre Organizația Europeană de Cercetare Nucleară (CERN), organizație în cadrul căreia s-a dezvoltat Web-ul prin intermediul lui Tim Berners-Lee. La sfârșitul anului 1993 erau raportate 623 de site-uri, după cum se poate vedea într-un studiu făcut de Matthew Gray din cadrul MIT [4]. Același studiu arată că, la sfârșitul anului 1994 erau mai mult de 10,000 de site-uri, creștere care devine exponențială în anii următori și care ne aduce mai aproape de Web-ul pe care îl cunoaștem astăzi.

Astfel că această creștere în dimensiunea Web-ului, a fost acompaniată și de o creștere în complexitatea site-urilor care formau acest spațiu. Site-urile cu informație statică au început să reacționeze la datele introduse de utilizatori, au început să recunoască utilizatorii prin conturi pe care aceștia le puteau accesa odată ajunsi pe aceste site-uri și chiar puteau să își modifice conținutul și să îl

adapteze în funcție de utilizator.

Această puternică dinamizare a conținutului unei pagini a constat într-o popularizare a spațiului Web nemaiîntalnită. Dacă la început, majoritatea informației care se găsea pe internet era de natură științifică, academică sau guvernamentală, odată cu aplicațiile Web interactive s-au deschis noi posibilități de care dezvoltatorii să profite și prin care utilizatorii să își îmbunătățească stilul de viață sau chiar să își îmbunătățească profesia sau afacerea.

Ca urmare, dezvoltatorii și-au setat țeluri tot mai mari, aplicațiile devenind tot mai complexe și rezolvând tot mai multe probleme din viața de zi cu zi a utilizatorilor. Dacă la început aplicațiile erau specializate în rezolvarea unei probleme specifice pentru utilizatorii săi, cu timpul paradigma s-a schimbat, la modă fiind aplicațiile care rezolvau o plajă cât mai mare de probleme și care ofereau cât mai multe funcționalități, astfel că utilizatorii nu trebuiau să părăsească site-ul pentru a găsi toate informațiile necesare.

Un exemplu pe care îl putem oferi aici este platforma Facebook, care a început prin a oferi utilizatorilor posibilitatea de a se conecta cu alți oameni din întreaga lume. Acesta a fost scopul inițial, care la momentul lansării a fost o idee inedită, însă pentru a putea răspunde cererilor pieței, dezvoltatorii Facebook au fost nevoiți să ofere utilizatorilor funcționalități ca: jocuri video în cadrul platformei, o modalitate de a fii la curent cu ultimele știri fără a fi nevoiți să părăsească platforma sau un mod pentru utilizatori prin care aceștia să poată să își promoveze afacerea. Toate aceste funcționalități rezolvă probleme reale și oferă valoare aplicației, însă nu fac parte din ideea și scopul inițial al platformei și nu pot fi rezolvate într-o manieră tradițională. Adică aceste funcționalități nu pot fi implementate toate cu o singură echipă de developeri, nu pot fi scrise toate în același limbaj de programare și din cauza diversității de activități pe care le desfășoară, trebuie să folosească multiple tehnologii din diferite arii ale diferitelor industrii.

Deci, unul din pașii care au condus către dezvoltarea și folosirea pe scară largă a serviciilor Web este creșterea complexității aplicațiilor. Astfel că, funcționalitățile importante au trebuit despărțite în aplicații de sine stătătoare care să fie mai apoi folosite și prezentate către utilizator ca o singură platformă în care aceste funcționalități sunt unificate și oferă o experiență completă.

Un alt pas important care a condus spre popularizarea serviciilor Web a fost evoluția telefoanelor mobile și a tabletelor către sistemele complexe care sunt astăzi, adică smartphone-uri. Accesarea paginilor Web de către dispozitivele mobile a fost la început un proces anevoios deoarece acestea erau create pentru a fi afișate pe computere, dispozitive ale caror dimensiuni erau mult mai

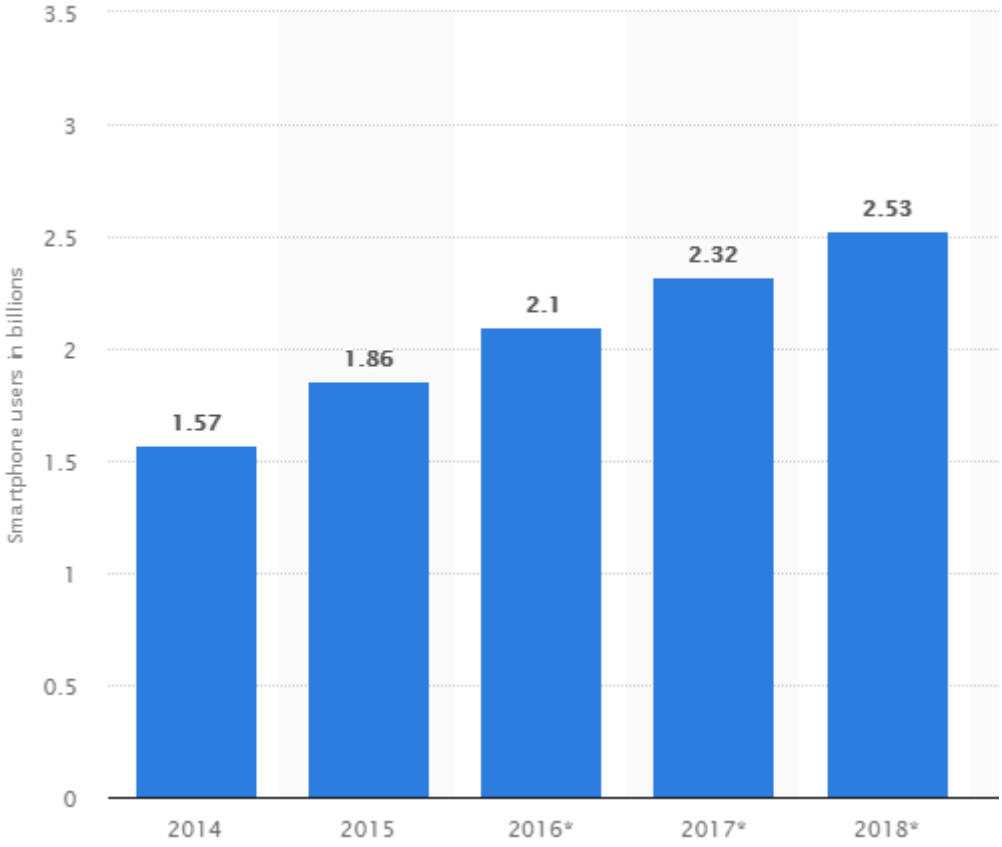
mari decât ale primelor telefoane mobile.
Diferențele dintre telefoanele mobile și calculatoare sunt însă multiple: diferența de rezoluții, memoria limitată a dispozitivelor mobile, tehnologiile limitate care pot rula pe sitemele de operare ale telefoanelor mobile și limitarea introducerii datelor utilizatorului prin interfața care de obicei consta dintr-o tastatură, navigarea în pagină fiind astfel și ea limitată.
Din cauza acestor lucruri, nu existau foarte mulți utilizatori ai Web-ului care îl accesau prin intermediul telefoanelor mobile iar dezvoltatorii aplicațiilor Web nu au fost interesați de această nișă pentru mult timp.
Astfel că, toate optimizările făcute asupra unui site Web, erau făcute nu de dezvoltatorii site-ului ci de către browser-ul care se afla pe dispozitivul mobil, atunci când acesta încerca să afișeze pagina pe telefonul mobil. Se efectuau tot felul de optimizări care filtrau conținutul HTML și afișau doar părțile necesare, astfel că de foarte multe ori site-urile arătau foarte diferit față de reprezentarea lor pe un browser normal, ba chiar arătau diferit de la un browser mobil la altul. Unele browsere mobile nu puteau rula javascript, flash sau alte tehnologii care erau și ele la început și care ofereau un mod dezvoltatorilor de a crea aplicații mult mai interactive. Acest lucru făcea ca browserul mobil să fie de foarte multe ori inutil sau mult prea instabil pentru a putea fi folosit.

Toate aceste lucruri se schimbă însă odată cu introducerea termenului și conceptului de smartphone, dispozitiv mobil care este fundamental diferit de telefoanele mobile mai vechi. Istoria telefoanelor mobile începe ca dispozitive al căror scop principal era comunicarea la distanță cu ajutorul undelor audio și care erau foarte mari și cântăreau foarte mult și au încercat să devină cât mai mici și compacte, au încercat mereu să îmbunătățească viața bateriei prin ecrane simpliste și funcționalități reduse.
La toate aceste concepte și standarde se renunță odată cu introducerea smartphone-urilor, decizie care pare contra intuitivă pentru industria de telefonie mobilă dar care se dovedește a fi o decizie inovatoare. Deși nu cântăresc foarte mult, smartphone-urile se străduiesc sa devină din ce în ce mai mari, au ecrane mari deoarece este principalul mod de interacțiune cu dispozitivul și cel mai important, nu sunt folosite doar pentru comunicarea audio. Aceste noi dispozitive sunt folosite pentru comunicarea în timp real cu ajutorul Internetului, sunt folosite pentru accesarea Web-ului, pentru accesarea locației utilizatorului și oferirea de informații de navigare cu ajutorul GPS-ului incorporat, etc.
Astfel că smartphone-ul perturba piața tradițională de comunicații aceștia fiind nevoiți să țină pasul cu toate aceste inovații.

Smartphone-urile introduc și siteme de operare mult mai mature decât cele găsite pe telefoanele mobile precedente și care se aseamănă mai mult cu sitemele de operare găsite pe computere. Cele mai importante sunt Android, sistem de operare dezvoltat de Google și folosit pe majoritatea smartphone-urilor de pe piață, iOS, sistem de operare dezvoltat de Apple și folosit doar pe dispozitivele dezvoltate de ei ca iPhone-ul, iPad-ul și Windows Phone, sistem de operare dezvoltat de Microsoft și folosit pe dispozitivele create de ei.

Toate aceste schimbări perturbă piața de telecomunicații astfel că giganți ca Nokia, dezvoltator Finlandez de telefoane mobile clasice majoritar pe piața de telfonie mobilă este detronat de către dezvoltatori noi care îmbrățișează inovația și reușesc să înțeleagă potențialul pieței de smartphone-uri ca: Apple, Samsung, HTC, etc.

Aceste inovații duc la o popularizarea globală rapidă a pieței mobile. Graficul de mai jos arată numărun de utilizatori de smartphone din lume în miliarde, din anul 2014 până în 2018 [5].



[5] Fig 3. //todo: Say things here

Se creează astfel o piață complet nouă, aceea a aplicațiilor mobile care oferă posibilitatea de inovație și extindere pentru multe companii și pentru foarte mulți dezvoltatori. Astfel că pentru fiecare sistem de operare apare câte un browser nou, browsere mult mai performante și mai capabile decât predecesoarele lor. Printre ele se numără: Google Chrome pentru Android, Safari pentru iOS și Internet Explorer pentru Windows Phone. Se poate observa că toate aceste browsere au câte un corespondent pe sistemele de operare găsite pe computere, Google Chrome pentru Windows, Linux și OSX, Safari pentru OSX și Internet Explorer pentru Windows. Acest lucru arată maturitatea sistemelor de operare introduse odată cu smartphone-urile și dorința companiilor importante de dezvoltare software de a intra pe această piață.

Din cauza numărului mare de utilizatori, în anul 2018 traficul majoritar generat pe Web în întreaga lume a fost făcut folosind un dispozitiv mobil. După cum se poate observa și în graficul de mai jos, care arată procentul traficului Web generat de către dispozitivele mobile din anul 2009 până în 2018 [6].

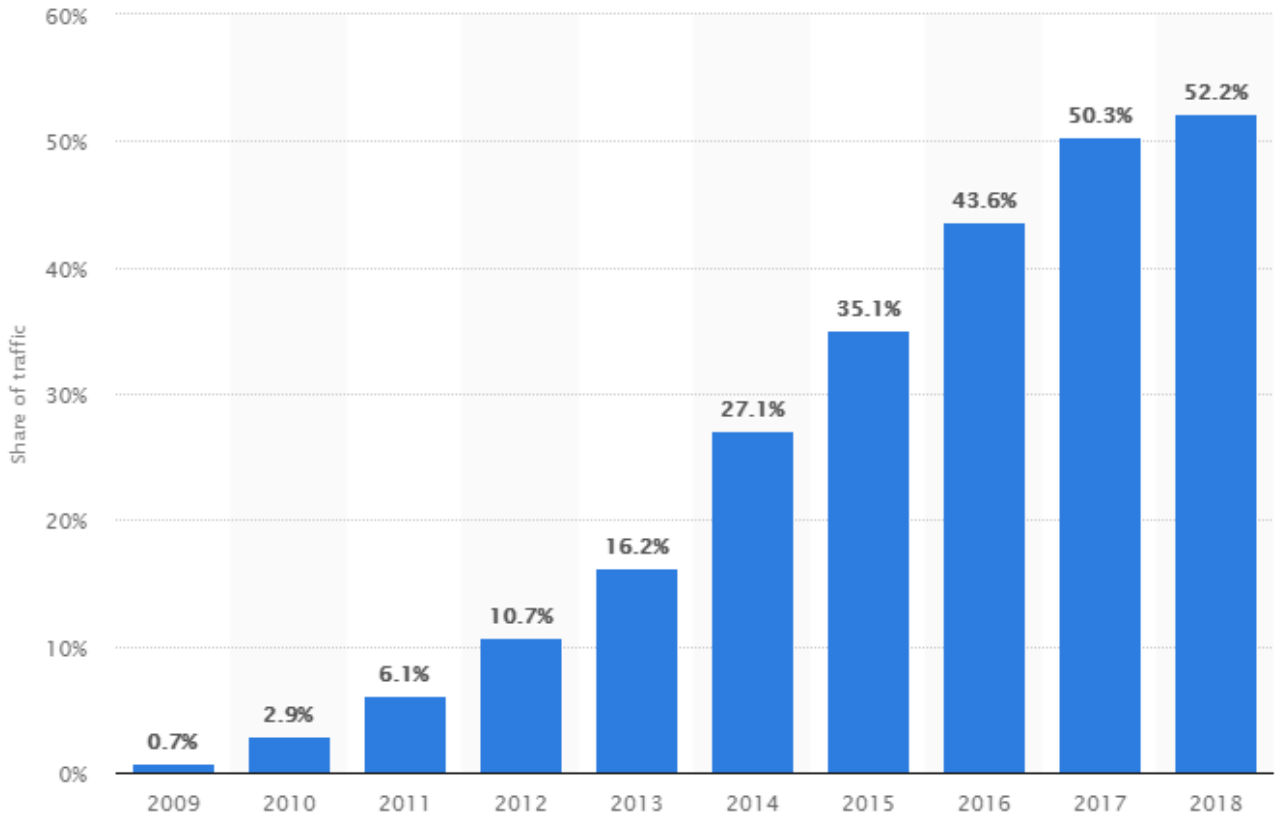


Fig 4. //todo: Say things here

Al doilea pas important în dezvoltarea și folosirea la scară largă a serviciilor Web, a fost această creștere masivă în traficul Web generat de către utilizatorii dispozitivelor mobile. Dacă la început dezvoltatorii de aplicații nu făceau niciun efort ca să acomodeze acești utilizatori, după ce numărul lor a continuat să crească acest lucru nu a mai fost posibil. Așa că a apărut așa numitul termen mobile-friendly, termen ce descrie faptul că o pagină Web este optimizată, ba chiar se poate sa arate și să se comporte complet diferit, pentru afișarea pe dispozitivele mobile.
Într-un timp scurt se adoptă în industrie conceptul paginilor Web responsive, care atunci când sunt implementate sunt făcute să răspundă la schimbări de rezoluție ale dispozitivului.

Însă pentru a acomoda toți utilizatorii pieței dispozitivelor mobile pentru unele platforme sau aplicații nu e de ajuns doar să își facă paginile Web să fie responsive. Unele aplicații trebuie portate pe platformele mobile în întregime pentru a livra anumite funcționalități care au nevoie să poată folosi caracteristici native ale sistemului de operare de pe dispozitivul mobil.
Majoritatea acestor aplicații și platforme trebuie să găsească o modalitate de a refolosi logica pe care deja o au implementată și să o integreze în aplicația mobilă.
Desigur, există mai multe soluții pe care dezvoltatorii le-au folosit de-a lungul timpului, nu doar serviciile Web. În continuare o să vorbim despre cele mai importante astfel de soluții și le vom compara pentru a putea găsi punctele forte și punctele slabe.

1.3 Servicii Web RPC, SOAP și REST

După cum am mai menționat, serviciile Web își au începuturile ca fiind o soluție la necesitatea facilitării interacțiunii dintre mașini. O primă soluție, RPC, apare din nevoia dezvoltatorilor pentru apelarea unor operații menite a fi executate la distanță.

1.3.1 RPC (Remote Procedure Call)

Este una dintre cele mai vechi soluții și presupune invocarea unei metode de către un client, pe un server care se află într-un spațiu de adresă diferit. Aceste metode sunt scrise de către programator fără ca el să scrie și detaliile interacțiunii și cum se face transportul datelor. Acest lucru este tratat de către implementarea sistemelor care se ocupă cu acest transfer, cum ar fi spre exepmplu Java RMI

pentru limbajul de programare Java. Odată cu apariția paradigmei de Programare Orientată Obiect, RPC devine RMI (Remote Method Invokation).

Avantajele modelului RPC/RMI:

- Metoda care este apelată de către client se apelează într-un context normal, ca atunci când se apelează o metodă locală
- Performanța unui sistem care se folosește de RPC este în general mai bună decât a unui sistem care folosește un serviciu Web SOAP sau REST
- Se pot folosi aceleași modele de darte atât în aplicația client, cât și în aplicația server

Dezavantajele modelului RPC/RMI:

- Odată cu creșterea complexității aplicației, modelul RPC/RMI devine tot mai greu de întreținut și de extins
- Nu există un standard bine definit și există multe implementări ale acestui protocol care nu sunt capabile să comunice între ele din cauza unor diferențe subtile
- Cele mai multe astfel de sisteme facilitează comunicarea între aplicații care sunt scrise în același limbaj de programare

Modelul de invocare a metodelor este exemplificat cu ajutorul figurii de mai jos.

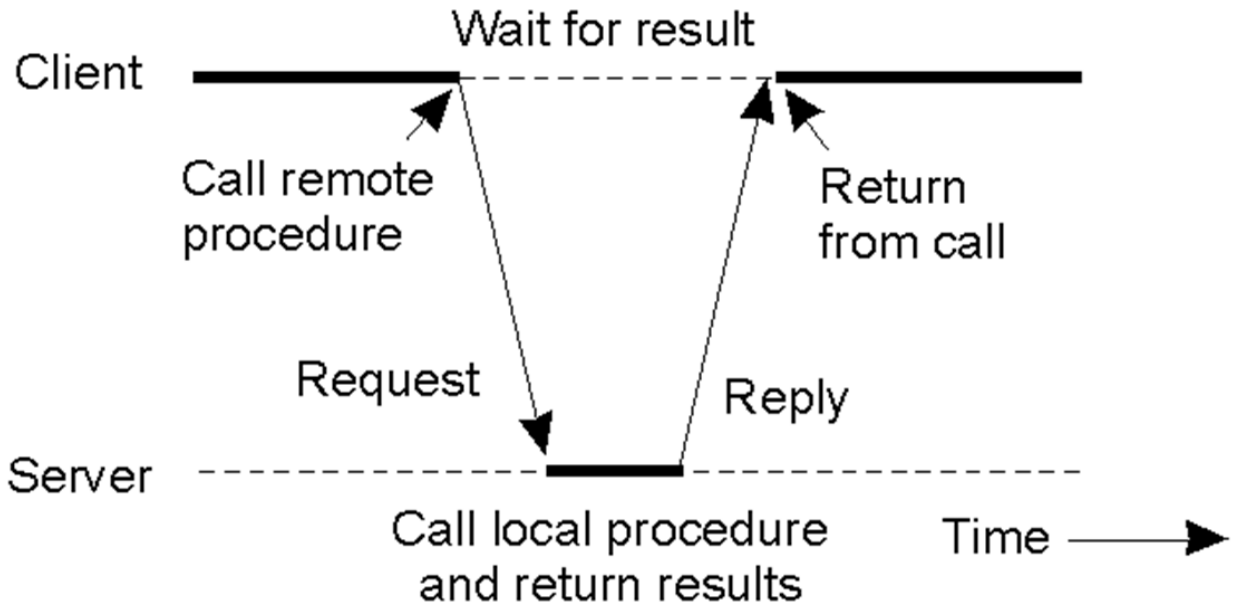


Fig 5. //todo: Say something here

1.3.2 SOAP (Simple Object Access Protocol)

SOAP este un protocol folosit pentru transmiterea mesajelor între două mașini, structurate cu ajutorul XML (Extensible Markup Language).

XML (Extensible Markup Language)

XML este un limbaj de marcare care definește un set de reguli pentru codificarea documentelor într-un format care este ușor de interpretat atât de către oameni, cât și de calculatoare. Putem considera XML ca fiind un standard internațional de descriere a datelor/resurselor electronice.
Termenul de marcaj (markup) era folosit, înaintea apariției XML, pentru a descrie anumite adnotări, găsite de obicei ca note marginale în cadrul textelor. Aceste marcaje aveau ca scop oferirea de indicații pentru un anumit bloc de text, ca spre exemplu formatarea acestuia, cum ar trebui acest bloc să fie listat sau chiar dacă blocul trebuie sau nu să apară în documentul final.
Un limbaj de marcare oferă un set de specificații utilizate mai apoi de dezvoltatori în codificarea datelor pe care le vor transmise între sisteme. Astfel că, XML este o soluție de specificare universală a datelor dintr-o aplicație care descrie atât structura, cât și comportamentul acestora.

Protocolul SOAP encapsulează informațiile într-un plic (envelope) care este format din două elemente, antent (Header) și (Body). În antet, se pot pune directive sau informații de care serverul să se ajute înaintea procesării mesajului propriu-zis, ca spre exemplu: informații despre autentificare sau informații despre proveniența mesajului. Datele aplicației sunt introduse în corpul mesajului, el fiind un simplu container pentru informația transmisă în format XML. Datele descrise trebuie să urmeze structura stabilită, altfel procesarea nu poate fi făcută de către server, însă dacă acest lucru este respectat, nu există alte restrictii cu privire la datele ce pot fi trimise în mesaj, alta decât că este nevoie să fie formatate ca și text/xml.
Avantajele protocolului SOAP:

- Față de modelul RPC, din cauză ca SOAP folosește HTTP nu este nevoie de configurări adinționale de proxy sau fierwall aduse serverului
- SOAP nu depinde de niciun limbaj de programare astfel că este posibilă comunicarea între două aplicații scrise în limbaje diferite

- SOAP este relativ simplu, deoarece folosește XML pentru reprezentarea datelor fiind ușor de citit și de către oameni nu doar de calculatoare. Acest lucru se dovedește foarte folositor atunci când există o eroare în mesajul generat de aplicație și e nevoie ca programatorul să depaneze mesajul
- SOAP este extensibil, astfel că există posibilitatea codificării oricărui fel de dată sau informație atât timp cât se respectă instrucțiunile
- La acest moment, SOAP este destul de matur și folosit încât nu există probleme ascunse sau neajunsuri de care să nu se știe, astfel că nu există posiblitatea începerii dezvoltării unei aplicații care folosește SOAP iar pe parcurs realizarea că acesta are probleme
- SOAP este potrivit pentru calculul distribuit
- SOAP definește un standard de securitate propriu care este foarte sigur

Dezavantajele protocolului SOAP:

- Din cauză că limbajul XML este foarte explicit, mesajele devin de foarte multe ori foarte mari dacă se consideră un set rezonabil de date. Astfel că lungimea de bandă folosită pentru transmiterea mesajului este foarte mare lucru care poate fi o problemă pentru aplicațiile cu foarte multe date
- Mesajele XML sunt procesate mai greu din cauza faptului că sunt atât de descriptive. Astfel că SOAP cu ajutorul XML reușește să creeze un standard inteligibil atât pentru oameni cât și pentru calculatoare. Acest lucru își spune cuvântul asupra categoriei din urmă, calculatoarele, care nu au nevoie de o verbozitate ridicată pentru a înțelege, datele având foarte multe informații redundante

1.3.3 REST (Representational State Transfer)

REST este un stil arhitectural care expune informații despre aplicație sub forma resurselor pe care le manipulează și oferă clientului posibilitatea de a modela resurse existente sau de a crea resurse noi. Pentru ca o aplicație să fie RESTful ea trebuie să urmeze un set de reguli sau instrucțiuni, formalizate prima dată de către părintele acestui stil arhitectural, Roy Fielding în teza sa de doctorat, în anul 2000.

Acesta propune că arhitectura Web-ului este determinată de constrângerile care se impun asupra

elementelor componente și explică faptul că proprietățile oferite de aceste constrângeri reflectă conceptul de stil arhitectural [6]. Astfel că, dacă se aplică constrângeri suplimentare Web-ului, putem ajunge la un nou stil arhitectural care să reflecte o arhitectură modernă, mai aproape de nevoile dezvoltatorilor.

Constrângerile pe care Fielding le preia din Web sunt:

1. Constrângerile legate de stilul arhitectural client-server pe care Web-ul funcționează. Principul care stă la baza acestui stil arhitectural este principiul separării responsabilităților. Astfel că, acestor două componente li se permite să se dezvolte independent una de cealaltă, decuplându-le, ceea ce înseamnă în același timp și că pot exista mai multe aplicații client și/sau server
2. Constrângerea pe partea de comunicare este aceea că Web-ul în esență, se folosește de o paradigmă de comunicare, în care nu se stochează o stare (stateless), adică serverul nu are informații despre clientul care face cererea și răspunde cu informații complete fiecărei cereri
3. Pentru eficiență, se introduce constrângerea de cache, care oferă posibilitatea clientului de a servi o resursă dintr-un mediu de stocare propriu, dacă acesta a mai interacționat cu acea resursă, scăpându-se astfel de unele interacțiuni

Constrângerile pe care Fielding le aduce în plus pentru a crea REST sunt:

1. Cel mai important principiu este acela de a avea o interfață uniformă, comună între componente. Această constrângere simplifică stilul arhitectural, oferă independența serviciului față de implementare, lucru ce ajută dezvoltarea independentă
2. Sistemul trebuie să fie compus din niveluri sau straturi. Această constrângere face posibil ca un nivel logic al aplicației să fie influențat doar de nivelul imediat următor aducând un plus de scalabilitate sistemului
3. Ultima constrângere este una opțională și vizează stilul de proiectare al aplicațiilor code-on-demand

Logica din spatele construirii REST este aceea de a creea un model arhitectural care să servească drept un cadru premergător pentru un spațiu Web standardizat în care comunicarea dintre două mașini să se efectueze într-un mod eficient. REST folosește protocoalele și conceptele mature din Web, ca HTTP și URI, cache și metode de securizare a datelor trimise ca TLS și SSL.

Avantajele REST:

- Mesajele trimise în REST sunt de obicei mai simple și nu există atât de multe informații redundante. Asta face ca lungimea de bandă folosită să fie mai mică față de serviciile Web SOAP spre exemplu
- REST folosește concepte familiare pentru interacțiunile care se petrec, adică se folosesc concepte deja existente în Web care doar sunt extinse sau adaptate
- Similar, concepte ca: integritatea datelor transmise sau securitatea lor este garantată de tehnologii existente, bine cunoscute
- Definirea comunicării în REST se face mai ușor decât în SOAP, deoarece interacțiunea și logica care se aplică unei resurse este definită de URI, operația simplificându-se la un singur verb HTTP ca: GET, POST, PUT, DELETE etc..
- REST nu depinde de niciun limbaj de programare
- Pentru cineva care ar dori să creeze un serviciu Web care să fie în concordanță cu principiile REST, există în foarte multe limbaje nenumărate tehnologii care să accelereze dezvoltarea

Dezavantajele REST:

- Caracterul general pe care REST îl propune oferă anumite restricții sistemelor și implică o complexitate mai mare ce trebuie luată în considerare atunci când se dezvoltă aplicația
- Odată cu moștenirea proprietăților și a tehnologiilor benefice din Web, se moștenesc și neajunsurile și problemele

Pentru structurarea datelor se poate folosi atât XML descris anterior cât și JSON (JavaScript Object Notation).

JSON (JavaScript Object Notation)

JSON este un format pentru structurarea datelor simplu și care nu are informație redundantă ci doar date structurate într-o manieră ușoară atât pentru mașini să îl genereze și să îl analizeze cât și de oameni să îl înțeleagă și verifca.

JSON are două concepte de bază:

- O colecție de perechi nume – valoare care se rezolvă la nume de câmpuri/variabile și obiecte în

limbajele de programare

- O listă ordonată de valori

Aceste elemente se pot compune pentru a descrie obiecte și colecții complexe.

Motivele pentru care am ales crearea aplicației server ca un serviciu Web REST se găsesc printre avantajele acestui stil arhitectural, dar și pentru că apreciez caracterul general pe care îl promovează și folosirea inteligentă a protocolului HTTP pentru comunicare.

2. Tehnologii folosite

Tehnologiile folosite vor fi împărțite în două categorii după aplicația în care s-au folosit, adică vor fi tehnologii pentru: aplicația client și pentru aplicația server. În acest capitol vom include și detalii de configurare pentru fiecare dintre aplicații. Aplicația server este un serviciu Web REST care expune resurse pe care aplicația client, care este o aplicație mobilă, le poate manipula și prezenta utilizatorilor. Aceste două aplicații sunt complentare și doar împreună pot forma aplicația Parker.

2.1 Tehnologii folosite pentru aplicația server

Aplicația server este un serviciu Web REST. Adică, este o aplicație Web MVC scrisă în limbajul de programare Java care cu ajutorul frameworkurilor Spring, Hibernate și a containerului Tomcat, reușește să se adapteze stilului arhitectural REST, respectând constrânngerile impuse.

Concluzii

Bibliografie