SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dijana Kraus

PRIMJER POLUSTRUKTURIRANE BAZE PODATAKA U SUSTAVU CouchDB

ZAVRŠNI RAD

Sisak, 2016.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dijana Kraus

Matični broj: S-38732/09-IZV

Studij: Primjena informacijske tehnologije u poslovanju

PRIMJER POLUSTRUKTURIRANE BAZE PODATAKA U SUSTAVU CouchDB

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Markus Schatten

Sisak, 2016.

Sadržaj

1. Uvod

Polustrukturirani model podataka je nastao kao rezultat sve većeg slanja podataka putem Interneta, te težnje da se heterogeni sadržaj pretražuje i sistematizira na način što sličniji bazama podataka. Jedan od načina da se heterogeni podaci strukturiraju je upotreba polustrukturiranog modela podataka.

NoSQL[1] sustavi za upravljanje bazama podataka omogučuju pohranu polustrukturiranih i nestrukturiranih podataka. Koriste se kako bi manipulirali sa velikom količinom podataka. Jako su korisni kada se trebaju analizirati velike količine nestrukturiranih podataka, što je jedan od razloga zašto ga koriste društvene mreže, odnosno web aplikacije u realnom vremenu.

Prikupljanje i obrada velikih količina podataka u realnom vremenu također se naziva Big Data. Definicija tog pojma je 3V:

- Volumen (eng. volume) – velika količina podataka koji se prikupljaju, obrađuju i stavljaju na raspolaganje za analizu
- Brzina (eng. velocity) – kontinuirano prikupljanje velike količine podataka u realnom vremenu
- Raznovrsnost (eng. variety) – podaci su dostupni u različitim oblicima i izvorima

U nastavku završnog rada, u drugom poglavlju je dan opis i prikaz polustrukturiranog modela podataka. U trećem poglavlju su opisani modeli podataka u NoSQL sustavima. S obzirom da u praktičnom dijelu koristimo model temeljen na dokumentima u kojima se podaci strukturirani pomoću JSON formata, tako je veći dio poglavlja posvećen opisu istih. Način na koji obrađujemo podatke je korištenje MapReduce metode koja je detaljnije objašnjena u četvrtom poglavlju, gdje su opisane performanse baze podataka CouchDB koju koristimo u praktičnom radu. U petom poglavlju je opisan praktični rad.

U praktičnom dijelu rada je napravljena baza podataka u sustavu CouchDB. Za serverski sloj aplikacije je korišten Nginx, a za prezentacijski sloj HTML[2]/CSS[3]/JavaScript skup tehnologija.

[1] NoSQL – Not Only SQL
[2] HTML - HyperText Markup Language
[3] CSS - Cascading Style Sheets

2. Polustrkturirani model podataka

2.1. Teorije grafova

U nastavku je dano par primjera teorije grafova koje predstavljaju polustrukturirani model podataka [4; str. 1].

Definicija 1. – Usmjereni graf

Neka V bude skup vrhova, a B ⊆ V x V skup bridova. Usmjreni graf G je par (V,B). Za svaki brid b = (v_i, v_j) kažemo da je v_i početak, a v_j završetak brida.

Definicija 2. – Ciklus

Ciklus u usmjerenom grafu je svaki put od jednog vrha u taj isti vrh. Graf bez ciklusa se naziva acikličnim grafom.

Definicija 3. – Podatkovni graf

Podatkovni graf je usmjereni graf G_D = (V, B) čiji su bridovi imenovani, a vrhovi su podatkovni objekti koji mogu biti:

- atomski (listovi)
- složeni (imaju bridove prema drugim vrhovima)

Svaki složeni vrh ima svoj identitet koji je jedinstven u danom podatkovnom grafu.

2.2. Opis polustrukturiranog modela podataka

Kod strukturiranih podataka najprije se definira struktura (tip, shema) podataka, a potom se počinju unositi podaci koji odgovaraju shemi. Polustrukturirani model nema shemu, te ga se opisuje kao samoopisujući (eng. self-describing), što znači da nije potreban opis strukture niti definiranje tipa podataka [1; str. 11].

Podaci se prikazuju u obliku ključ : vrijednost što možemo vidjeti u sljedećem primjeru:

```
{name: “Marko“, tel: 555888, e-mail: marko@gmail.com}
```

Prikaz koda 1. Primjer polustrukturiranog tipa podataka

Podaci mogu biti složeniji, kao u sljedećem primjeru, gdje vidimo da je atribut „name“ složen, odnosni, sastoji se od još dva atributa.

```
{name: {first: “Marko“, last: “Kapić“},
tel: 555888,
e-mail: marko@gmail.com}
```

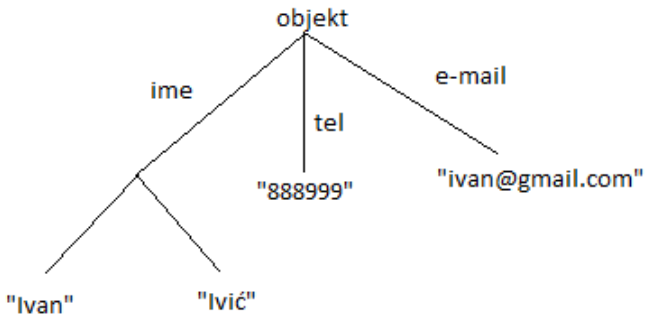
Prikaz koda 2. Primjer složenijeg polustrukturiranog tipa podataka

Pod predpostavkom da su vrijednoti atributa jedinstvene, dozvoljeno je imati iste nazive atributa.

```
{name: “Marko“, tel: 555888, tel: 555886 }
```

Prikaz koda 3. Isti atributi u polustrukturiranom tipu podataka

Podaci se mogu prikazati i putem grafa, gdje korjen predstavlja objekt koji je povezan s vrijednostima na vrhovima grafa.



Slika 1. Prikaz polustrukturiranih podatka pomoću grafa

Glavna prednost polustrukturiranog tipa podataka je što objekti ne moraju imati iste atribute, niti atributi moraju biti isti tip podataka.

```
{
    osoba: {ime: “Marko“,
              tel: “888555“,
              e-mail: marko@gmail.com
   },
            {ime: {ime: “Ivan“,
                      prezime: “Ivić“
                      },
              tel: 555999,
              adresa: “Zagrebačka 8“
   }}
```

Prikaz koda 4. Razlike u objektima kod polustrukturiranog modela podataka

2.3. Prikaz strukturiranih podataka kroz polustrukturirani model

Kroz polustrukturirani model moguće je prikazati strukturirane podatke. Slijedi prikaz podataka iz relacijskog modela kroz polustrukturirani model.

Relacijska baza se opisuje shemom r1(a,b,c) i r2(c,d), gdje su r1 i r2 nazivi ralacija, a a,b,c i c,d nazivi stupaca u tim relacijama. U praksi moramo i definirati tipove podatka koji se unose u stupce.

Primjer (Izvor: [1; str. 14])

r1

a	B	C
a1	b1	c1
a2	b2	c2

r2

c	d
c2	d2
c3	d3
c4	d4

Slika 2: Prikaz relacija i njihovih vrijednosti

```
{ r1: {red: {a:a1, b:b1, c:c1},
       red: {a: a2, b:b2, c:c2}
      },
 r2: {red: {c:c2, d:d2},
      red: {c:c3, d:d3},
      red: {c:c4, d:d4}
     }
}
```

Prikaz koda 5. Prikaz podataka iz relacija u polustrukturiranom obliku

Vrijednost relacija su prikazane kao niz podatak unutar objekata.

3. Model podataka

Model podataka se kod NoSQL baza podataka može podijeliti na tri kategorije: model temeljen na dokumentima (eng. document model), model grafa (eng. graph model), model ključ-vrijednost (eng. key-value model).

Model grafova je usmjeren na veze između podataka. Podaci su čvorovi, a veze između njih su bridovi. Te veze činu ovaj model idealnim za socijalne medije gdje postoje upiti o stupnjevima razdvajanja među korisnicima.

Model ključ-vrijednost je najednostavnija vrsta NoSQL baza podataka. Koristi asocijativni niz kao osnovu pohrane podataka. Svaka stavka u bazi podataka pohranjena je u obliku ključ:vrijednost. Zbog svoje jednostavnosti nije adekvatan za složene aplikacije, ali upravo ta jednostavnost ga čini brzim i jednostavnm za korištenje.

Najčešće korišteni tip NoSQL baze podataka su baze podataka temeljene na dokumentima. Također taj model koristimo u praktičnom dijelu.

3.1. Model temeljen na dokumentima

Baza podataka temeljena na dokumentima je organizirana kao kolekcija dokumenata. Dokumenti unutar baze podataka slični su zapisima u relacijskim bazama podataka, ali su fleksibilniji jer ne postoji shema. Podaci se upisuju u dokument čija struktura nije unaprijed određena, što znači da različiti dokumenti mogu imati različita obilježja. Svaki dokument ima jedinstveni ključ koji se koristi za pretragu i izmjenu dokumenata. Podaci se pohranjuju u obliku ključ:vrijednost. Baza podataka temeljena na dokumentima omogučuje indeksiranje dokumenata na temelju primarnog ključa, ali i sadržaja dokumenta.

Ovakav pristup olakšava razvoj aplikacija gdje unaprijed nije poznato s kakvim će se sve podacima raditi i gdje je potrebno spremati više različitih tipova podataka unutar jedne grupe.

Podaci su strukturirani pomoću JSON[4] formata.

Pohranjivanje podataka u ovom obliku ima nekoliko prednosti:

- Dokumenti su nezavisne jedinice što omogučava bolje performanse i jednostavniju distribuciju podataka na više servera.
- Aplikacijska logika je jednostavnija za programiranje.
- Nestrukturirani podaci se lako pohranjuju.

[4] JSON - JavaScript Object Notation

3.1.1. JSON (JavaScript Object Notation)

JSON je format za prijenos podataka čija je sintaksa izvedena iz sintakse JavaScript programskog jezika. Iako je nastao u JavaScriptu, koristi se kod svih većih programskih jezika zbog svoje praktičnosti.

Prednost mu je što je čiljiv ljudima, a jednostavan računalima za analiziranje i generiranje. JSON je zapravo tekstualni format koji ne ovisi o niti jednom programskom jeziku, no koristi konvencije programskih jezika C obitelji [2].

Tipovi podataka u JSON formatu:

- String – niz znakova
- Broj
- Boolean – true or false
- Niz (Array) – sortirani niz podataka; označava se uglatom zagradom, a vrijednosti se razdvajaju sa zarezom.
- Objekt – nesortirani niz oblika name/value parova, označava se vitičastom zagradom, a dvotočka dijeli naziv od vrijednost.

BSON ili binarni JSON (eng. binary JSON) je binarno kodirana serija JSON dokumenata, odnosno binarni format za prijenos podataka u kojem se podaci spremaju u dokumente. Kao i JSON, može se implementirati sa mnogim programskim jezicima. Podržava ugradnju dokumenata i nizova unutar drugih dokumenata i nizova. Podržava neke tipove podataka koje JSON ne podržava, npr. „date“ i „bindata“ [5].

4. CouchDB

Couch[5]DB je dokumentno orijentirana baza podataka implementirana u programskom jeziku Erlang. Dio je NoSQL generacije baza podataka.

U nastavku će se navesti neke od osnovnih značajki i prednosti CouchDB-a.

Arhitektura je otporna na pogreške (eng. foult-tolerant), odnosno ako dođe do greške, sustav nastavlja s radom. Pojedinačni problem ne utječe na cijeli sustav, nego ostaje izoliran u zahtjevu.

Dizajniran je tako da se lako nosi za porastom prometa. Ako na web-u dođe do naglog porasta prometa CouchDB će apsorbirati sve zahtjeve, bez da dođe do ispada sustava. Iako će mu trebati malo više vremena za svaki zahtjev, ali svi će dobiti odgovore. Dobro se nosi za rastom i padom zahtjeva. Omogućava izradu skalabilnih aplikacija.

Dok tradicionalne baze podataka zahtjevaju da se model podataka unaprjed definira, CouchDB ne zahtjeva nikakvu shemu, vec unosimo podatke kako bi ih unosili u stvarnim dokumentima.

CouchDB je fleksibilan, daje vam dovoljno opcija da izgradite sustav koji odgovara vašim zahtjevima.

Jedna od tih opcija je CouchDB sinkronizacija. Njegova temeljna funkcija je sinkronizirati dvije ili više baze podataka, odnosno riješiti niz problema: pouzdano sinkronizirati baze podataka između više računala kako nebi došlo do suvišne pohrane podataka, distribuciju podataka u klasteru CouchD instanci koje dijele podskup ukupnog broja zahtjeva prema klasteru i distribuciju podataka između fizički udaljenih mijesta [3].

CouchDB sinkronizacija koristi REST API[6] kao i svi klijenti. Obavlja se postupno, odnosno ako tijekom sinkronizacije nešto pođe krivo, npr. ako se izgubi Internetska veza, kada se veza pojavi, prijenos će se nastaviti gdje je stao. Osnovna pretpostavka je da može doći do nekog problema, kao što je problem s mrežom, zato je CouchDB dizajniran kako bi se nosio s problemom i što bolje oporavio [3].

4.1. Dosljednost sustava

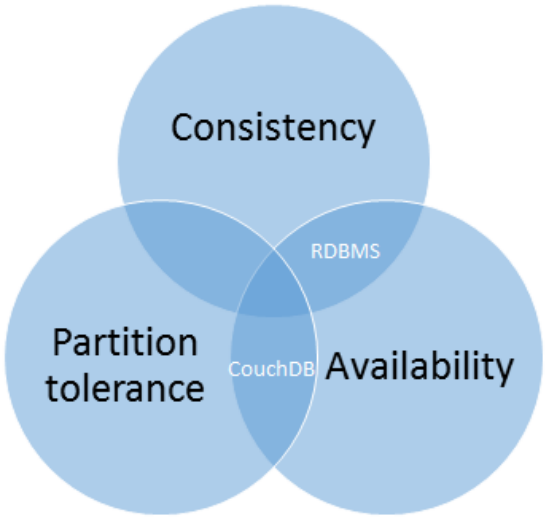
Distribuirani sustav je sustav ostvaren putem računalne mreže. Posebnost mrežnog računalstva je da veze mogu potencijalno nestati; postoji više strategija za upravljanje mrežne segmentacije. CouchDB se razlikuje od ostalih sustava prihvaćanjem dosljednosti, za razliku od stavljanja apsolutne konzinstentnosti ispred dostupnosti, kao RDBMS[7] ili Paxos. Njihovi prisupi se razlikuju ovisno što su im prioriteti: dosljednost, dostupnost ili dijeljenje tolerancije [2].

[5] Couch - Cluster of Unreliable Commodity Hardware
[6] REST API – REpresentational State Transfer Application Programming Interface
[7] RDBMS - Relational database management system

4.1.1. Cap teorem

Prema CAP teoremu nemoguće je da distribuirani sustav omogući sva tri atributa, a to su:

- Dosljednost (eng. Consistency) – svi kljenti vide iste podatke bez obzira na trenutna ažuriranja
- Dostupnost (eng. Availability) – dostupnost podataka, odnosno garancija da će svaki zahtjev dobiti svoj odgovor
- Dijeljenje tolerancije (eng. Partition tolerance) – dostupnost sustava bez obzira na propuste u mreži



Slika 3. CAP teorem

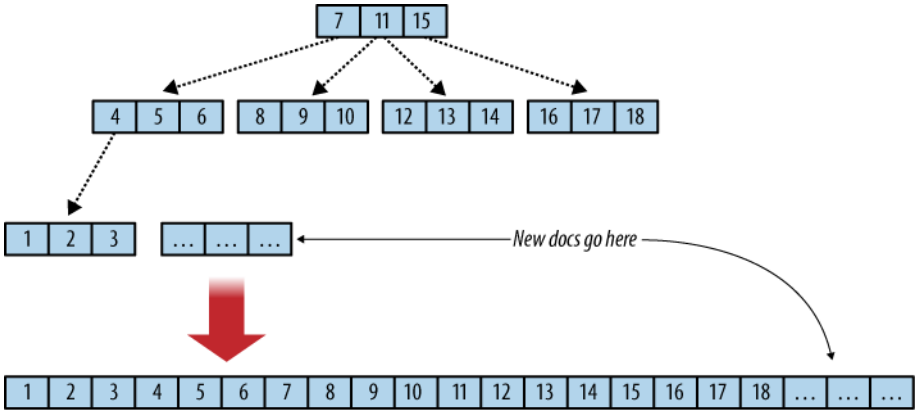
Kada sustav naraste dovoljno velik da jedan čvor baze podataka ne može nositi teret, razumno rješenje je dodati više poslužitelja. Kada tome dodamo čvorove moramo početi razmišljati kako dijeliti podatke između njih. Bez obzira koji pristup uzmemo susrećemo se s problemom kako sinkronizirati sve baze podataka.

Ako je raspoloživost prioritet, dopuštamo klijentu da upisuje podatke na jedan čvor baze podataka, bez čekanja na druge čvorove. Ako baza podataka zna kako se pobrinuti o usklađivanju poslova između čvorova postižemo eventualnu dosljednost u zamjenu za visoku dostupnost.

Za razliku od tradicionalnih relacijskih baza gdje svaka izvedena akcija podliježe provjeri dosljednosti, CouchDB omogućava jednostavno građenje aplikacija koje žrtvuju hitnu doljednost za veliko poboljšanje performansi koje dolaze s jednostavnom distribucijom.

4.1.2. Lokalna dosljednost

CouchDB koristi B-tree za pohranu svih podataka, dokumenata i pogleda. B-tree je struktura za sortiranje podataka koja omogućava unos, pretraživanje i brisanje podataka u logaritamskom vremenu.



Slika 4. B-tree [2].

Također koristi metodu „MapReduce“ koja se primjenjuje na svaki dokument u izolaciji, a daje rezultat strukture ključ/vrijednost. Pristupiti podacima putem ključa važna je karakteristika; rezultat je poboljšanje brzine, ali i dijeljenje podataka na više čvorova.

Za razliku od relacijskih baza podataka, CouchDB ne koristi bravu, nego Multi-Version Concurrency Control[8] (MVCC) za upravljanje višebrojnom pristupu bazi podataka.

CouchDB pruža mogućnost stvaranja više inačica dokumenata; što znači ako korisnik želi promijeniti vrijednost u dokumentu, može stvoriti novu inačicu dokumenta i spremiti ga preko stare. Ovisno u kojem je trenutku poslan zahtjev, ovisi koju će inačicu dokumenta vidjeti; ali neće doći do pogreške, nego će dokument biti vidljiv i dok traje stvaranje nove inačice dokumenta.

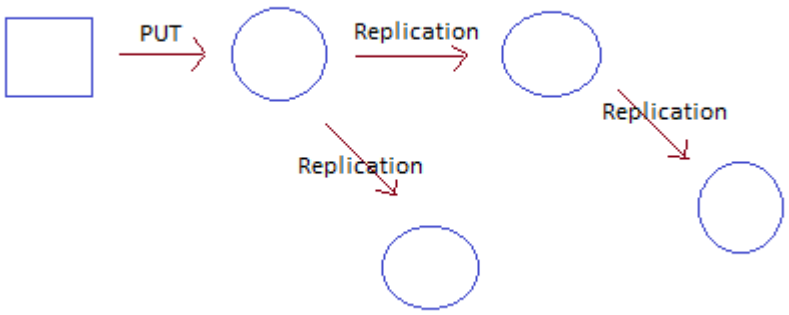
Validacija dokumenata se odvija pomoću JavaScript funkcije slične onima koje se koriste za MapReduce. Kod svake izmjene dokumenta funkcija validacije će proći kroz postojeći dokument, primjerak novog dokumenta i dodatne informacije, kao što su korisnički podaci za provjeru autentičnosti, te može odobriti ili odbiti ažuriranje.

[8] Multi-Version Concurrency Control (MVCC) je algoritam za kontrolu čitanja i pisanja podataka koji omogućuje da se nad podacima istovremeno obavljaju čitanja i pisanja bez klasičnog zaključavanja redaka.

4.1.3. Dijeljena dosljednost

Održavanje dosljednosti unurar jednog čvora baze podataka je relativno lako, ali problem nastaje kada se pokušava zdržati dosljednost između više poslužitelja baze podataka.

CouchDB postiže konzinstentnost između baza podataka pomoću inkrementalne replikacije, procesa u kojem se promjene dokumenata povremeno kopiraju između poslužitelja. Sustav izgrađuje shared nothing[9] klaster baze podataka, što znači da je svaki čvor neovisan. S inkrementalnom replikacijom moguće je sinkronizirati podatke između bilo koje dvije baze podataka na proizvoljan način i u proizvoljnom vremenu; nakon toga svaka baza podataka može raditi samostalno [2].



Slika 5. Inkrementalna replikacija izeđu čvorova

Ako se izmijeni isti dokument u dvije različite baze, tijekom replikacije dolazi do sukoba. Sustav ima automatsku detekciju za riješavanje sukoba. Jedan dokument se sprema kao najnovija inačica u povijest dokumenta, a drugi kao prethodna inačica u povijest tog dokumenta, kako bi mu mogli pristupiti ako je to potrebno.

Na korisniku je da rukuje s problemom kako najviše odgovara njegovom zahtjevu. Može ostaviti odabrane inačice na mjestu, vratiti na staru inačicu ili pokušati spojiti dvije inačice i spremiti rezultat.

[9] Arhitektura shared nothing je distribuirana računalna arhitektura u kojoj je svaki čvor neovisan i samodostatan.

4.2. Sigurnost

CouchDB je postavljen tako da sve zahtjeve odobrava svima, odnosno svi imaju administratorske ovlasti dok se to ne promijeni u postavkama. Objašnjeni pristup je nazvan „Admin party“.

Kako bi se to promijenilo, kreira se administrator koji ima svoje korisničko ime i lozinku. To se odnosi na server administratora koji onda ima ovlasti nad cijelim sustavom. Definiran je set zahtjeva koje samo administrator može raditi:

- Kreiranje baze podataka (PUT /database)
- Brisanje baze podataka (DELETE /database)
- Kreiranje *design* dokumenta (PUT /database/_design/app)
- Uređivanje *design* dokumenta (PUT /database/_design/app?rev=1-4E2)
- Brisanje *design* dokumenta (DELETE /database/_design/app?rev=1-6A7)
- Itd.

Kada je kreiran administrator, anonimni korisnici i dalje mogu čitati, unositi podatke, dostupan im je i design dokument, ali ga ne mogu brisati niti mijenjati. Kako bi se ograničio pristup anonimnim korisnicima, potrebno je kreirati _design dokument s validacijskom funkcijom kojom se reguliraju prava pristupa.

U svakoj bazi podataka imamo opciju „Security“. Putem security objekta može se postaviti administratora baze i člana baze. Oni imaju različita prava:

- član baze može pristupiti svim dokumentima u bazi, kreirati nove dokumente, i uređivati postojeće (osim design dokuenata)
- administrator baze ima sve ovlasti nad svojom bazom

U slučaju da ne postoji administrator baze, samo server administrator ima ovlasti kao administrator baze. A u slučaju da nije definiran član baze, svi kosrisnici mogu pristupiti dokumentima, te kreirati novi dokument (osim design dokumenta). U slučaju da je definiran član, tada samo autorizirani korisnici s navedenim imenom i ulogom mogu pristupiti dokumentima u bazi.

Nakon što postavimo administratora ili korisnike svaki puta kada mijenjamo dokument poziva se funkcija validate_doc_update koja dodaje novi zapis u CouchDB izvještaj (couch.log). Ako pogledamo zapis iz izvještaja možemo vidjeti bazu, ime autentifikacijskog korisnika i niz uloga koje korisnik ima. Po tome možemo zaključiti da je admin zapravo običan korisnik koji ima ulogu „_admin“. Razdvajanjem korisnika i uloga autentifikacijski sustav pruža veću fleksibilnost.

Korisnička lozinka nije vidljiva. Ona se automatski sažima. Kreira se 128 bit-ni Univerzalni Jedinstveni Identifikator (eng. Universally unique identifier - UUID) koji je slučajni niz. Zatim se kreira Secure Hash Algorithm 1 (SHA-1) na temelju lozinke i UUID niza simbola. Dobivamo sažetu vrijednost koja je vidljiva.

Kako bi se usporedila lozinka sa spremljenom sažetom vrijednosti tijekom autentifikacije; ponavlja se cijeli postupak, te se nova sažeta vrijednost uspoređuje se spremljenom sažetom vrijednosti.

CouchDB od inačice 1.1.0 podržava i SSL[10] (port 6984).

4.2.1. Autentikacija kolačićima

Osnovni autentifikacijski sustav koji koristi običnu tekstualnu lozinku je dobar, ali ne pruža dovoljnu sigurnost osim ako ne poduzmemo dodatne mjere. Osim toga ne pruža dovoljno dobro korisničko iskustvo. Osim ovog osnovnog autentifikacijskog sustava CouchDB podržava i autentikaciju kolačićima (eng. cookie authentification). Ako koristimo ovaj način korisnik se može prijaviti kroz HTML formu, nakon čega će CouchDB generirati jednokratni token koji korisnik može koristiti prilikom sljedećeg zahtjeva prema CouchDB. Kada CouchDB u sljedećem zahtjevu vidi token autentificirat će korisnika bez potrebe za lozinkom. Standardna postavka je da token vrijedi 10 minuta.

Da bi održali token te autentificirali korisnika, korisničko ime i lozinka moraju biti poslani u „_session“ API.

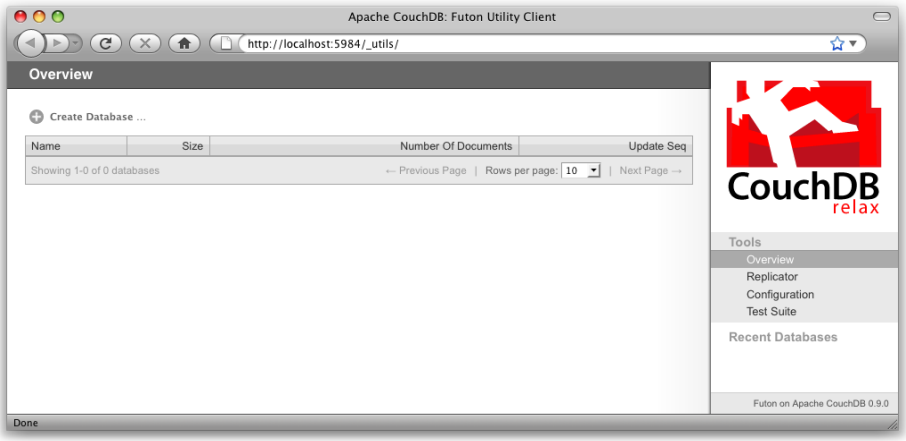
4.3. Futon

Futon je ugrađeno administracijsko sučelje koje daje puni pristup svim značajkama CouchDB-a. Uz futon možemo stvoriti i obrisati baze podataka, pregledavati i uređivati dokumente, sastaviti i pokrenuti MapReduce i potaknuti replikaciju između baza podataka [2].

4.3.1. Kreiranje baze i dokumenata

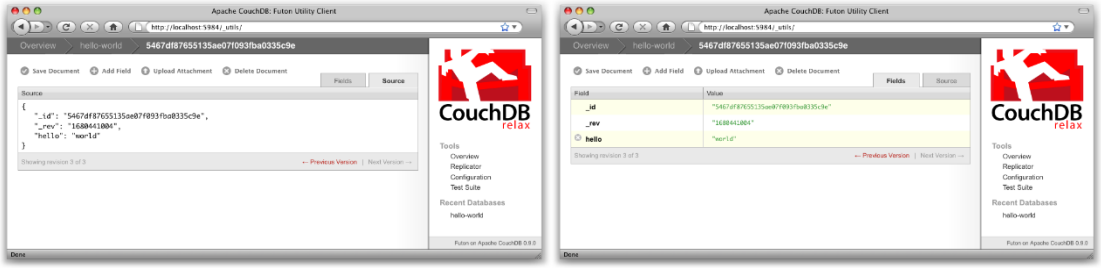
Kreiranje baze podataka u futonu je jednostavno. Odaberemo opciju „Create Detabase“, upišemo ime i potvrdimo. Nakon toga možemo kreirati dokumente. Kada kreiramo dokumente, svakom dokumentu se automatski pridružuje njegov _id i _rev, a polja kreiramo s opcijom Add Field. Polja su strukture ključ:vrijednost, odnosno ime:vrijednost. Vrijednost polja mora biti upisana JSON formatom, inače neče biti prihvaćena.

[10] SSL-Secure Socket Layer



Slika 6. Početni zaslon Futona

Kod upisivanja, pregleda ili ispravljanja polja, postoji opcija „Source“, koja prikazuje čisti JSON.



Slika 7. Prikaz dokumenta u Futonu

Svaki _id je jedinstven, iako se može promijeniti u bilo koju drugu vrijednost, ali za najbolje rezultate se preporučuje UUID, odnosno univerzalni jedinstveni identifikator. UUID su slučajni brojevi koji imaju jako malu mogućnost podudaranja. To je osiguranje da se neće stvoriti dva dokumenta koja imaju isti _id.

Kada se rade izmjene u dokumentu, otvara se dokument i mijenja se njegova JSON struktura, nakon toga nova se inačica sprema, ali mu se dodjeljuje nova _rev vrijednost. Rev djeluje kao sigurnosna značajka kod spremanja dokumenta.

CouchDB pokreće upite pomoću MapReduce funkcije kojoj pristupamo pomoću opcije temporary view, a kreiramo ju s JavaScript jezikom. Njome dobivamo design dokument – posebni tip dokumenta koji ima ugrađeni dio koda. Njegov _id počinje s _design. S design dokumentom možemo:

- Filtrirati dokumente u bazi kako bi došli do određenih podataka,
- Izvući određene podate iz baze i prezentirati ih na način koji nam je potreban,
- Tražiti dokumente po određenom ključu,
- Doći do određenih izračuna

Map funkcija uzima dokument kao argument (_doc se odnosi na svaki dokument u bazi). Emit funkcija ima dva parametra:ključ i vrijednost. Rezultat je stavljen u listu gdje svaki red ima svoj ključ i vrijednost, a podaci su sortirani prema ključu.

Primjer će biti prikazan u opisu praktičnog rada.

Reduce funkcija obavlja operaciju na sortiranim podacima koje smo dobili putem map funkcije. Njen zadatak je da reducira podatke dobivene map funkcijom. Za razliku od map funkcije koja kreira rezultat, reduce funkcija vraća vrijednost.

Rezultate možemo pohraniti i koristiti kada nam je potrebno.

U _design dokumentu možemo kreirati validate_doc_update funkciju koja je namjenjena sprječavanju mijenjanja dokumenata od neautoriziranog korisnika, te ograničavanju sadržaja dokumenta. Odnosno omogućava nam da odredimo tko može mijenjati dokumente i kako dokumenti moraju izgledati. Dokument može sadržavati samo jednu validacijsku funkciju, ali baza može sadržavati više design dokumenata. Kako bi novi dokument bio spremljen mora proći sve validacijske funkcije u bazi.

4.3.2. Sigurnosne kopije

Kao što smo već spominjali replikacija sikronizira dvije kopije iste baze, koje mogu biti na istom ili na dva različita servera. Ako se napravi promjena na jednoj kopiji baze, replikacija će poslati promjene drugoj kopiji.

Prvo moramo kreirati praznu bazu podataka koja će biti ciljna kopija. Zatim odaberemo opciju „replicator“, odredimo izvornu i ciljnu datoteku te odaberemo opciju „replicate“ i sigurnosna kopija je kreirana.

Radi se o jednosmjernom procesu. Dokumenti se kopiraju s jedne baze na drugu, ali ne i u drugom smijeru. Ako želimo dvosmjernu replikaciju, moramo dva puta pokretati replikaciju sa zamijenjenom izvornom i ciljnom darotekom.

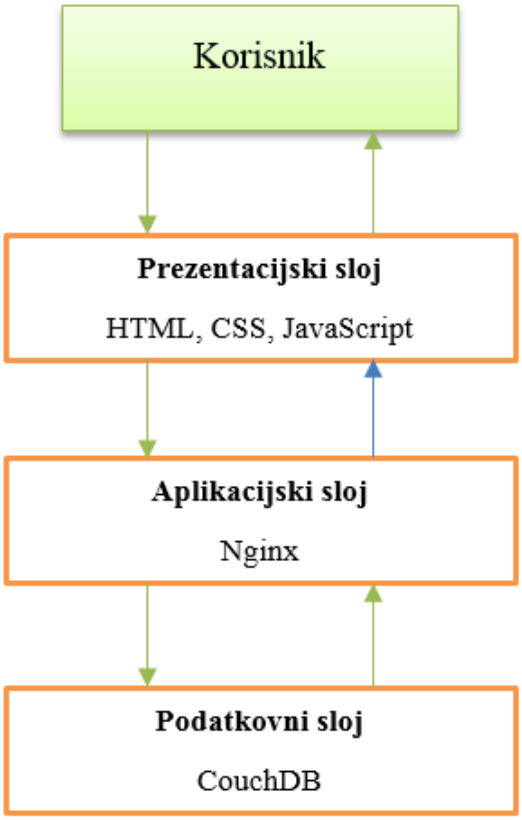
5. Praktični rad

U praktičnom radu je napravljena web aplikacija koja služi za oglašavanje kućnih ljubimaca.

5.1. Arhitektura sustava

Arhitektura je toslojna.

1. Podatkovni sloj – Baza podataka CoucDB
2. Aplikacijski sloj – Nginx
3. Prezentacijski sloj – HTML, CSS, JavaScript



Slika 8. Arhitektura aplikacije

5.1.1. Podatkovni sloj

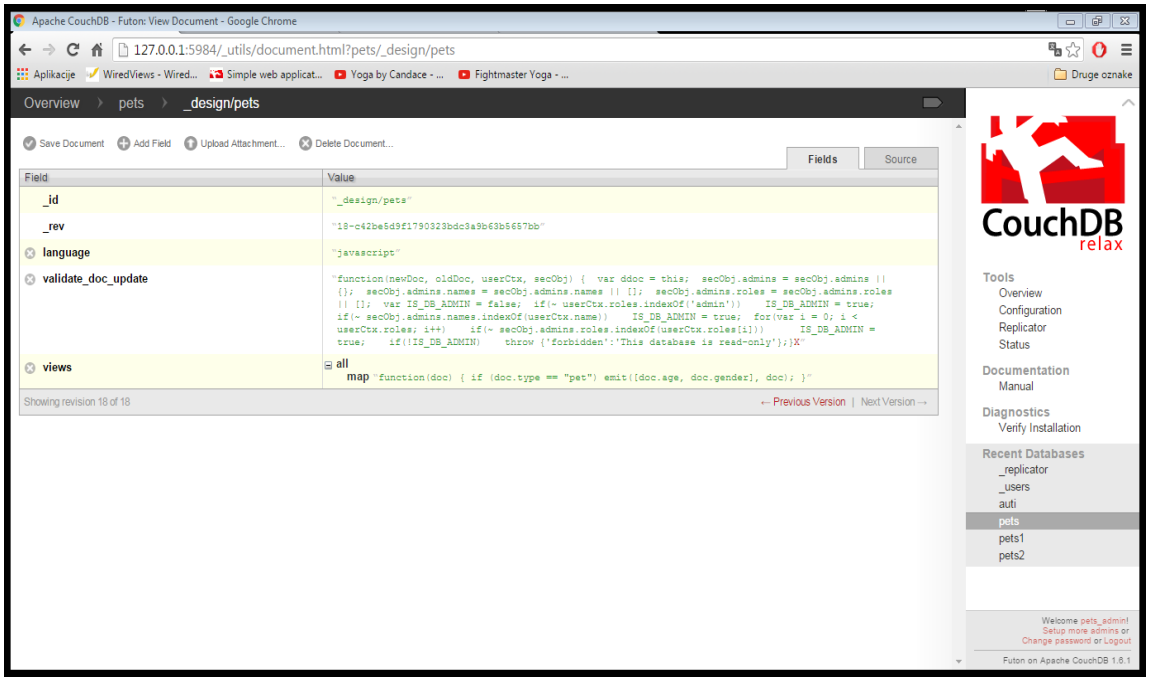
Baza podataka je kreirana u sustavu CouchDB.

Baza se sastoji od dokumenata u kojima se nalaze karakteristike kućnih ljubimaca. Ono što je bitno u dokumentu da bi ga se prepoznalo kao relevantan dokument je „type“ - „pet“, odnosno „type“-„about-text“.

```
{
"_id": "1e4791946cb505bdbbba191361000035",
"_rev": "8-214f8ef9c86786877d99e66afc55df32",
"breed": "Jack russell terijer",
"name": "Ares",
"mobile": "098765679",
"weight": "7",
"height": "25",
"description": "Ares, dečko star 10 mjeseci...",
"gender": "muški",
"age": "mladi psi",
"type": "pet",
"_attachments": {
   "Ares.JPG": {
      "content_type": "image/jpeg",
      "revpos": 2,
      "digest": "md5-8eI2PfVLd5BwHVCCjdfuJg==",
      "length": 4086478,
      "stub": true
   }
}
}
```


Prikaz koda 6. Izgled dokumenta u JSON formatu

Type ključ se koristi kako bi se lakše napravio design dokument.



Slika 8. Design dokument

U design dokumentu imamo map funkciju.

```
{
"map": "function(doc) {\n  if(doc.type == \"pet\")\n  emit([doc.age,doc.gender], doc);\n}"
}
```

Prikaz koda 7. Map funkcija

Funkcija map se poziva jednom za svaki dokument koji zadovoljava upit, a vraća ključ:vrijednost parove koji se spremaju u map rezultat.

Putem nje dobivamo listu u kojoj su svi dokumenti kojima je type=pet; za ključ dobivamo vrijednosti „age“ i „gender“, a za rezultat cijeli dokument. Desing dokument nam daje rezultat koji se prikazuje krajnjem korisniku. Imamo dva design dokumenta: „pets“ i „about_text“.

Baza ima server administratora, kao i administratora baze. U design dokumentu smo kreirali validacijsku funkciju kojom određujemo ovlasti korisnika. Samo korisnik sa ulogom administratora može dodavati ili uređivati dokumente, ako se radi o korisniku koji nema navedenu ulogu baza podataka se neće ažurirati.

```
"validate_doc_update":
"function(newDoc, oldDoc, userCtx, secObj) {

var ddoc = this;

secObj.admins = secObj.admins || {};
secObj.admins.names = secObj.admins.names || [];
secObj.admins.roles = secObj.admins.roles || [];

var IS_DB_ADMIN = false;

 if(~ userCtx.roles.indexOf('admin'))
IS_DB_ADMIN = true;

if(~ secObj.admins.names.indexOf(userCtx.name))
IS_DB_ADMIN = true;

for(var i = 0; i < userCtx.roles; i++)
if(~ secObj.admins.roles.indexOf(userCtx.roles[i]))
IS_DB_ADMIN = true;

if(!IS_DB_ADMIN)
throw {'forbidden':'This database is read-only'}; }"
```

Prikaz koda 8. validate_doc_update

5.1.2. Aplikacijski sloj

Kao server koristimo NGINX. NGINX je open-source HTTP server visokih performansi, te reverse proxy. Da bi radio CouchDB autentifikacijski mehanizam CouchDB i web stranica moraju biti na istoj adresi (url) da bi dijelio kolačiće kao npr.: http://myhost.com/couch.

Konfiguracija za NGINX je slijedeća:

```
server {
    listen       80;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
       root   html/couchdb;
       index  index.html index.htm;
    }
    location /couch/ {

       expires -1;
       add_header      Cache-Control private;
       proxy_cache     off;

       proxy_pass http://127.0.0.1:5984/;
       proxy_set_header        X-Real-IP $remote_addr;
       proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
       proxy_set_header        Host $http_host;
    }...
```

5.1.3. Prezentacijski sloj

Prezentacijski sloj je ono što vide korisnici. U izradi su korištene tehnologije:

- JavaScript
- HTML
- CSS

U sljedeća dva primjera ćemo prikazati HTML i JavaScript kod za stranicu „O nama“. HTML je zaslužan za prezentaciju, a JavaScript za interaktivnost.

```
<html>
 <head>
  <title>Jednostavna web aplikacija koja koristi polustrukturirane baze podataka - CouchDB</title>
  <link href="/css/bootstrap.css" rel="stylesheet">
  <link href="/css/index.css" rel="stylesheet">
  <link rel="icon" href="/favicon.ico">

  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <script src="/couch/_utils/script/json2.js"></script>
  <script src="/couch/_utils/script/sha1.js"></script>
  <script src="/couch/_utils/script/jquery.js"></script>
  <script src="/couch/_utils/script/jquery.couch.js"></script>
  <script src="/couch/_utils/script/jquery.dialog.js"></script>
  <script src="/js/settings.js"></script>
  <script src="/js/about.js"></script>
 </head>
 <body>
  <div class="container-fluid">
   <div class="row">
    <div class="content in-center">
     <ul class="nav nav-pills">
      <li><a href="/">Udomi me</a></li>
      <li class="active"><a href="/about.html">O nama</a></li>
     <li><a href="/admin.html">Admin</a></li>
     </ul>
.............
 </html>
```

Prikaz koda 9. HTML

U prikazanom HTML kodu vidimo naslov stranice (eng. title), veze sa skriptama u kojima je unešen CSS koji se rabi za opis prezentacije dokumenta i vezu sa navedenom ikonom. U nastavku su postavke za prikaz znakova hrvatskog jezika, te veze na eksterne skriptne datoteke. Unutar definiranih sekcija u tijelu HTML-a nalaze se podaci koji omogučavaju iste stilove uređivanja za više sekcija. Zatim je unešena nesređena lista sa vezema.

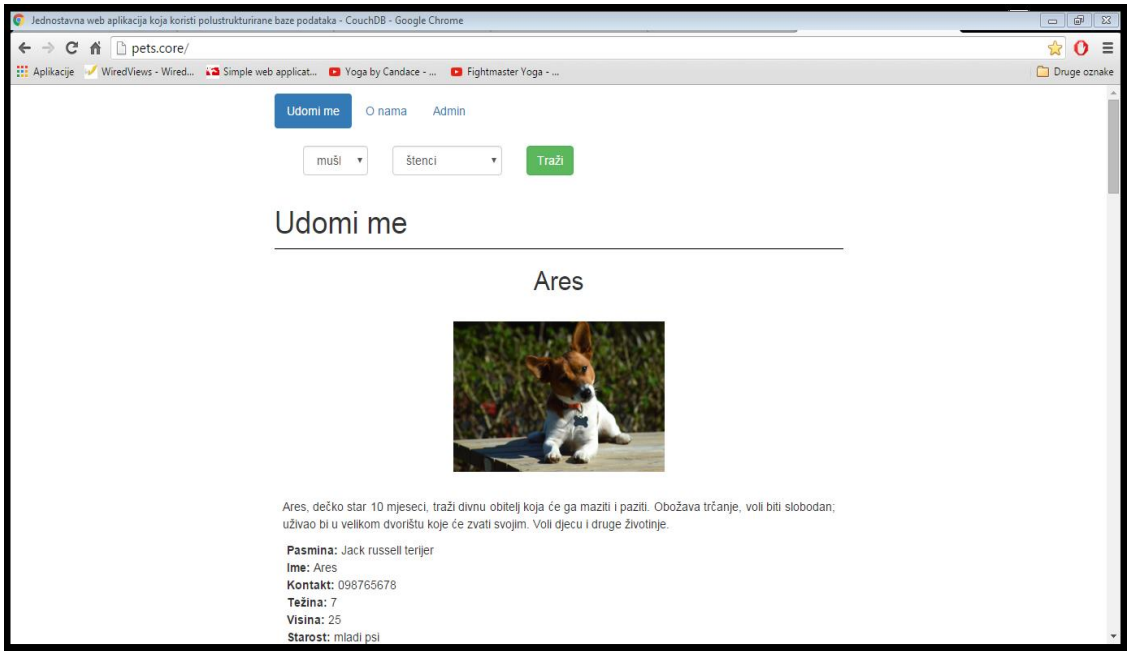
```
$(document).ready(function () {
   $.couch.db(DB_NAME).view("about-text/all", {
      success: function(data) {
         var aboutText = data.rows.map(function (row) {
            return row.value;
         })[0].text;
         $(".about-text").html(aboutText);
      },
  error: function(status) {
         console.error(status);
      },
      reduce: false
```

```
    });
});
```

Prikaz koda 10. JavaScript

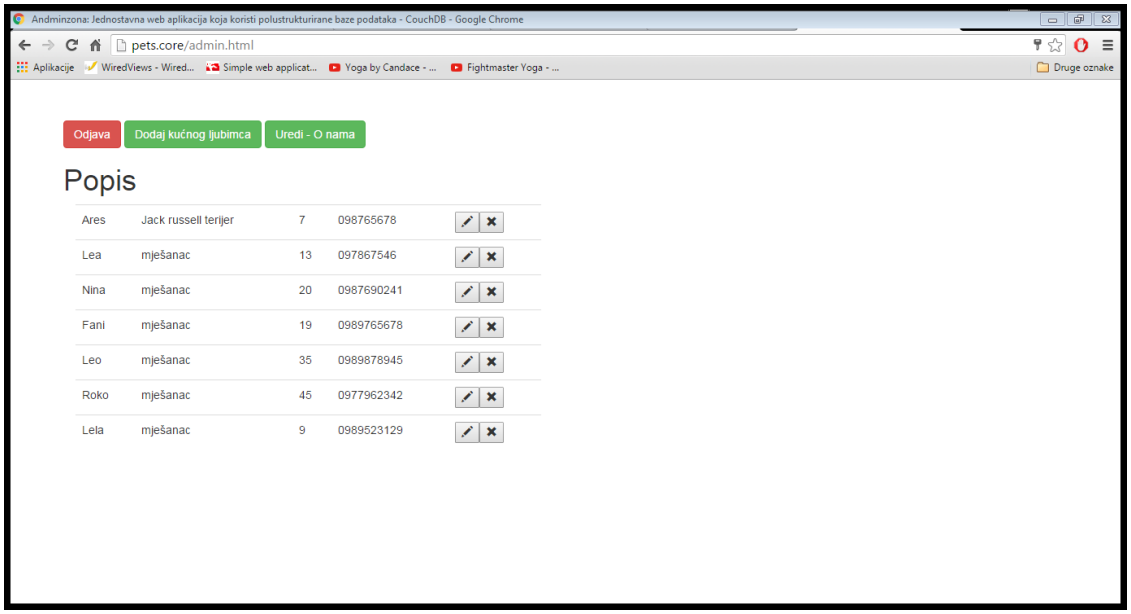
Putem skripte u kojoj se nalazi JavaScript izvršava se pogled na bazu i prosljeđuje se rezultat u definiranu funkciju. Pomoću .map funkcije vrši se iteracija po rows nizu koji je dobiven u odgovoru sa servera, te uzima samo podatak iz svojstva value. U nastavku se postavlja HTML sadržaj elementu „aboutText“. U slučaju neuspješnog zahtjeva prema bazi, ispisuje se status http zahtjeva u konzoli.

U nastavku je prikazan izgled aplikacije, te kratak osvrt na opcije koje imaju korisnici i admin.



Slika 9. Početna stranica

Korisnik ima pristup opisu stranice i pregledu ljubimaca. Stavljena je opcija pretraživanja prema spolu i starosti. Admin zoni može prostupiti samo server administrator i administrator baze.



Slika 10. Admin zona

U admin zoni administrator može uređivati i brisati postojeće podatke, te dodavati nove ljubimce.

6. Zaključak

Nekoliko je prednosti NoSQL baza podataka, ali najbitnije su da ne postoji shema, koja je u relacijskim bazama podataka ograničavala unos podataka u relacije. Također, postoji više modela NoSQL sustava koji se mogu implementirati. SQL sustavi pružaju visoke performanse i hirizontalno skaliranje. Ali teško je zapravo reći koji pristup je bolji, a razlog tome je taj što obje vrste baze podataka imaju prednost u određenom području primjene.

CouchDB kao dio NoSQL sustava također ima svoje prednosti i nedostatke; neke od prednosti su:

- fleksibilost jer nema shema po kojoj se spremaju podaci
- dostupnost podataka putem URL-a
- jednostavna replikacija podataka, itd.

Svi NoSQL sustavi, pa tako i CouchDB iz dana u dan postaju sve popularniji i imaju sve više korisnika. Nesmunjivo je da će se u budučnosti još razvijati i prilagođavati potrebama korisnika.

U praktičnom radu je napravljena baza podatak u sustavu CouchDB. Implementirano je aplikacijsko sučelje, u čijoj je izradi korišten skup tehnologija HTML/CSS/JavaScript. Zahtjevi aplikacije nisu veliki, ali se ista može po potrebi proširivati.

7. Literatura

[1] Abiteboul S., Buneman P., Suciu D., (2000), Data on the Web - From Relations to Semistructured Data and XML, Morgan Kaufmann Publishers, San Francisco, California

[2] „Apache CouchDB Documentation“, The Apache Software Foundation, 2015. Web. 21 Siječanj 2016., Dostupno na: http://docs.couchdb.org/en/1.6.1/

[3] „BSON“, Web. 20 Ožujak 2016., Dostupno na: http://bsonspec.org/

[4] „Introducing JSON“, Web. 12 Veljače 2016., Dostupno na: http://www.json.org/

[5] Anderson, J. Chris., Lehnardt, J., Slater, N., „CouchDB: The Definitive Guide“, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, 2015. Web. 10. Studeni 2015., Dostupno na: http://guide.couchdb.org/editions/1/en/index.html

[6] Lennon J., (2009), Beginning CouchDB

[7] „Nginx documentation“, Web. 5 Prosinac 2015., Dostupno na: http://nginx.org/en/docs/

[8] Osmani, A., Learning JavaScript Design Patterns, 2015. Web. 10 Listopad 2015., Dostupno na: https://addyosmani.com/resources/essentialjsdesignpatterns/book/

[9] Schatten M., Ivković K., (2012), Regular Path Expression for Querying Semistructured Data - Implementation in Prolog, Central European Conference on Information and Intelligent Systems, FOI Varaždin

[10] Tiwari S., (2011), Professional NoSQL, John Wiley & Sons, Inc.