SERVICII CLOUD DE VERSIONARE CU SUPORT BAZE DE DATE NoSQL

Dragoş NICOLAU Mihaela TOMESCU Daniel SAVU Ion Alexandru MARINESCU

dragos@ici.ro mtomescu@ici.ro dsavu@ici.ro ionut@ici.ro

Institutul Naţional de Cercetare-Dezvoltare în Informatică - ICI Bucureşti

Rezumat: Lucrarea de faţă se concentrează pe studierea dezvoltării unui sistem integrat de versionare a fişierelor sursă utilizate în dezvoltarea de proiecte software. Sistemul este rezident în Cloud şi fundamentat pe un baze de date de tip NoSQL. Toate resursele funcţionale şi de date sunt rezidente pe o platformă Cloud. Alegerea acestei soluţii (Cloud şi NoSQL) conferă un ridicat spor de viteză necesar în condiţiile în care acest gen de aplicaţie este supus unei densităţi mari de cereri inclusiv din cele care incumbă încărcări, scanări, comparări şi descărcări la distanţă de fişiere.

Cuvinte cheie: Coduri Open Source, Baze de date NoSQL, Cloud Computing, acces la Internet.

Abstract: This paper focuses on studying the development of an integrated versioning system for the source files used in the software project development. The server component is a .net Web service, powered by a NoSQL Type Database Management System. All functional and data resources are resident on a Cloud platform. Choosing this solution (NoSQL in conjunction with Cloud technology) provides a great boost in speed, necessary as this type of application is subject to a high density of requests, more-over so when operations like file uploads, file parsing, file content comparisons and file downloads occur at a constantly increasing rate .

Keywords: Open Source code repository, NoSQL Databases, Cloud Computing, access to Internet.

1. Introducere

Versionarea software-ului este procesul de atribuire de nume unice de versiuni sau de numere unice de versiuni pentru stări unice ale software-ului. Într-o anumită categorie de numere de versiuni (majoră, minoră), aceste numere sînt, în general, atribuite în ordine crescătoare şi corespund dezvoltărilor noi de software. La un nivel cu granulaţie fină, controlul revizuirii este adesea folosit pentru a urmări versiunile incrementale diferite ale informaţiilor electronice, indiferent dacă aceste informaţii sînt sau nu software.

Software-ul modern este deseori urmărit folosind două scheme diferite de versionare a software-ului:

- un număr de versiune internă care poate fi incrementat de mai multe ori într-o singură zi, cum ar fi un număr de control al revizuirii;
- versiune lansată (release), care de obicei se schimbă mult mai rar, cum ar fi versionarea semantică sau numele unui cod de proiect[1] .

Versionarea software-ului permite programatorilor să ştie cînd au fost făcute modificări în software şi să urmărească aceste modificări. În acelaşi timp, permite clienţilor potenţiali să afle despre apariţia de versiuni noi şi să recunoască versiunile actualizate. Cel mai obişnuit tip de versiune, care este, de asemenea, un standard de facto în lumea Linux, este Semantic Versioning[2]. Versiunea semantică prescrie în principal 3 numere întregi separate printr-un punct (MAJOR.MINOR.PATCH) care sunt incrementate astfel:

- versiunea MAJOR atunci cînd s-au efectuat modificări incompatibile cu API;
- versiunea MINOR atunci cînd a fost adăugată funcţionalitate într-un mod compatibil înapoi;
- versiunea PATCH atunci cînd au fost efectuate remedieri ale unor erori într-un mod compatibil înapoi.

Cea mai populară schemă de versiuni utilizează identificatori bazaţi pe secvenţă în care fiecare versiune lansată este prevăzută cu un identificator unic care conţine unul sau mai

[1] Preston-Werner, Tom (2013). Semantic Versioning 2.0.0. Creative Commons. http://semver.org/spec/v2.0.0.html.

[2] *** - Semantic Versioning. http://semver.org.

multe numere de ordine sau litere3. Acestea înseamnă schimbări între versiuni, în care modificările se bazează pe nivelul de semnificaţie. Primele schimbări ale secvenţei desemnează nivelul cel mai semnificativ, iar modificările ulterioare arată o mai mică semnificaţie. De exemplu, v1.01 ar putea fi o remediere minoră a unor erori, în timp ce v1.2 înseamnă o versiune mai importantă. De asemenea, această schemă poate utiliza un zero în prima secvenţă pentru a reprezenta starea „alpha", unu pentru starea „beta", doi pentru software-ul candidat a fi lansat şi trei pentru publicare. O altă metodă este separarea secvenţelor prin intermediul unor caractere. În anumite pachete software se pot utiliza numere negative de versiune. Alte tehnici implică folosirea anilor şi a datelor (de exemplu, Windows 95) sau doar coduri aleatorii (de exemplu, Adobe Photoshop CS2).

Despre versionare

Proiectele software devin din ce în ce mai complexe, lucru care impune modularizarea activităţii de dezvoltare, adică împărţirea sarcinilor de programare pe echipe. În vederea asigurării autonomiei de lucru între echipe se folosesc instrumentele de versionare: principial, ele constau într-o bază de date unde se memorează după criterii bine determinate stadiul curent (versiunea) al fiecărui modul (fişier cu linii de cod) care compune proiectul. În felul acesta se asigură atît reversibilitatea schimbărilor executate în fiecare fişier (în fiecare moment, orice echipă de programatori, în mod independent de celelalte echipe, poate recupera orice fişier într-o anumită etapă de evoluţie, în vederea restudierii şi reutilizării, păstrînd intacte versiunile anterioare), cât şi evidenţa oricăror schimbări survenite în cod (pentru fiecare fişier sursă se ştie cine, cînd şi ce modificări a făcut). Astfel, instrumentul de versionare menţine în mod structurat, sigur şi uşor accesibil "traiectoria" dezvoltării unui proiect.

Instrumentul de versionare oferă următoarele avantaje:

- salvări automate - ştergerea şi modicarea unui fişier sînt operaţii nu doar înregistrabile, ci şi reversibile;
- partajarea resurselor între mai multe staţii de lucru - chiar în cazul lucrului pe o maşină independentă, orice progres va fi consemnat automat odată cu revenirea la lucrul în sistem;
- centralizarea rezultatelor - oricine are acces imediat la contribuţia oricărui membru al oricărei echipe;
- propagarea automată a modificărilor.
- acces clar şi rapid la etapele de evoluţie a proiectului;
- posibilitatea de a studia schimbările, ceea ce presupune creşterea eficienţei în procesul de dezvoltare primară sau ulterioară a unui proiect;
- posibilitatea de a găzdui / insera modificări oricînd reversibile din / în fişierele pe care lucrează alţi membri ai grupului;
- posibilitatea de a executa testări de funcţionalitate fără a afecta mersul proiectului;
- diferenţiere vizibilă între două stadii de evoluţie (versiuni) ale unui modul;
- posibilitatea de a renunţa la modificări care s-au dovedit neinspirate.

Sistemele de versionare accesibile prin reţea (adică la distanţă) pot fi centralizate (fiecare maşină de lucru se bazează doar pe sistemul de versionare la distanţă) sau distribuite (fiecare maşină de lucru are propriul sistem de versionare local, conectat însă la cel principal, aflat la distanţă).

În sprijinul acestui model de gestionare a proiectelor vin două tehnologii: Cloud Computing şi Sistemele de baze de date de tip NoSQL, fiecare avînd atît avantaje cât şi zone perfectibile.

Cloud Computingul este un model convenabil gîndit să permită accesul la cerere, prin reţea, la o grupare de resurse de calcul configurabile (de exemplu reţele, servere, echipamente de stocare, aplicaţii şi servicii), care pot fi puse la dispoziţia utilizatorului în mod rapid şi cu un efort minim de administrare sau interacţiune cu prestatorul acestor servicii. Cloud oferă viteză şi siguranţă în funcţionare, în timp ce bazele NoSQL excelează prin timpi extrem de scurţi de răspuns în condiţiile interogării unui număr foarte mare de înregistrări.

[3] *** - Software Versioning. https://technick.net/guides/software/software_versioning/.

Baza de date NoSQL ignoră principiile Sistemelor Relaționale, adică nu stochează conexiuni folosind tabele, ci foloseşte chei de identificare în cadrul înregistrărilor însele. Datele pot fi regăsite în funcție de cheile atribuite. Acest tip de baze de date este prin natura sa scutit de necesitatea de a normaliza datele şi de a stoca conexiunile în tabele dedicate - aducînd astfel performanțe sporite aplicațiilor care le folosesc. De asemenea, acest tip de baze de date îmbunătățeşte şi răspunsul la schimbările ce pot surveni de-a lungul timpului în schema bazei. Într-un sistem relațional nu există flexibilitatea necesară pentru a asimila modificări în modelul de date. Faptul că bazele de date NoSQL nu au o schemă de date fixă (rigidă), le face să fie flexibile, dînd astfel posibilitate dezvoltatorilor să opereze schimbări de structură "din mers", la nevoie, fără să fie forțați la schimbări majore în cod.

Prin urmare, baza de date de tip NoSQL se va constitui în depozit de coduri sursă, deservind un ansamblu structurat de facilități de găzduire / procesare pentru cantități mari de cod software, pentru uz public sau privat. Aceste resurse de cod, cărora sistemul le memorează şi gestionează versiunile intermediare şi modificările făcute pe parcurs, sînt utilizate în proiecte open-source sau în alte proiecte multi-dezvoltator.

2. Despre Cloud Computing

Cloud este materializarea ideii de serviciu informatic de foarte mari dimensiuni, care concentrează resurse hardware şi software, oferind astfel viteză, spațiu de stocare practic nelimitat şi o mare varietate de platforme de lucru.

Pentru a ilustra beneficiile consumării de servicii Cloud, Institutul Național pentru Standardizare şi Tehnologie din SUA - NIST a identificat cinci puncte de sprijin ale relației de bază client/serviciu public.

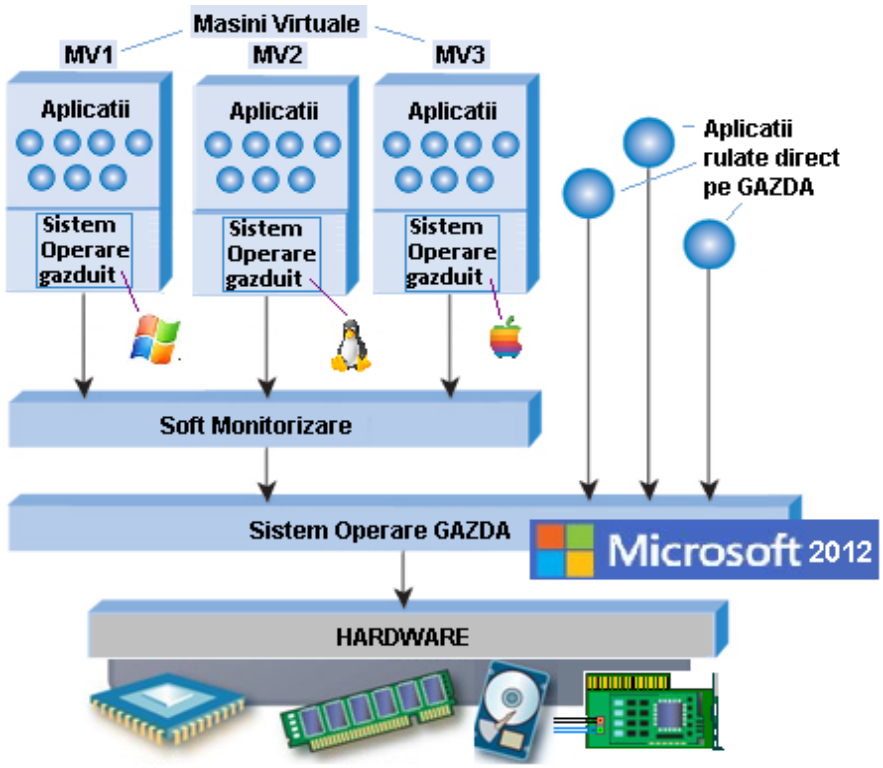
- auto-servire la cerere (= 'On-demand self-service') – un utilizator poate să obțină acces la resursele de calcul de care are nevoie, fără a fi nevoit să interacționeze cu fiecare furnizor de servicii;
- acces la rețele de bandă largă – resursele sînt disponibile prin intermediul unei rețele, fiind accesibile de către orice sistem de operare rulînd pe oricare fel de stație de lucru: PC, laptopuri, tablete sau dispozitive
  smart-phone;
- rezervoare de resurse – resursele de calcul ale furnizorului sînt reunite pentru a deservi beneficiarii pe baza unui model chiriaş-multiplu, bazat pe diferite resurse fizice şi virtuale care pot fi alocate şi realocate în mod dinamic, conform cerințelor chiriaşilor. Amplasarea exactă a acestor resurse nu este cunoscută de utilizatorul final, dar aceasta poate fi specificată la un nivel ridicat de abstractizare, cum ar fi prin menționarea țării, a oraşului sau a centrului de date. Ca exemple de resurse, pot fi amintite cele de stocare, procesare, memorare sau cele de transmitere prin bandă largă;
- comunicare rapidă – Resursele de calcul pot fi lansate şi oferite elastic cu o "scalabilitate" ridicată. La utilizator, resursele disponibile pentru furnizare apar în mod arbitrar ca nelimitate;
- servicii măsurabile – Sistemele Cloud pot controla şi optimiza, în mod automat, utilizarea resurselor, prin măsurarea consumului fiecărui serviciu oferit (trafic pe rețea, durată de acces, spațiu de memorie, nivel de solicitare pe procesor etc.). Nivelul de utilizare a fiecărei categorii de resurse poate fi monitorizat, controlat şi raportat, furnizorul asigurînd transparența dorită pentru fiecare serviciu utilizat, atît pentru chiriaş, cât şi pentru utilizator.

În cele ce urmează, vom explicita semnificația celor 3 concepte principale care se află la temelia filozofiei Cloud. NIST a definit următoarele trei modele de servicii, prin care Cloud Computing este oferit: Software ca Serviciu (SaaS), Platformă ca Serviciu (PaaS) şi Infrastructură ca Serviciu (IaaS)[4].

[4] NIST - The Definition of Cloud Computing (Sept. 2011); Special Publication 800-145

1) Software ca Serviciu (Software as a Service - SaaS) este modelul în care furnizorii de servicii Cloud permit accesul utilizatorilor la aplicațiile care sînt deja instalate pe infrastructura furnizorului de Cloud. Aplicațiile sunt accesibile de pe diverse tipuri de dispozitive, fie prin web browser (de exemplu, e-mail bazat pe web), fie printr-o interfață program specializată. Utilizatorul nu poate influența infrastructura Cloud de bază (servere, sisteme de operare sau de stocare) şi, în cele mai multe cazuri, nu are control (sau, eventual, are un control limitat) asupra aplicației utilizate. În cazul aplicațiilor desktop, funcționarea rezidă în trimiterea de capturi de ecran către client, de aceea se impune instalarea pe client a unei aplicații speciale, cum s-a amintit imediat mai sus.

2) Platformă ca Serviciu (Platform as a Service - Paas) este un model în care furnizorii permit chiriaşilor să îşi dezvolte/expună pe infrastructură Cloud propria aplicație (creată sau achiziționată), utilizînd limbaje de programare, biblioteci, servicii şi alte instrumente oferite de către furnizor. Chiriaşul nici nu administrează, nici nu controlează infrastructura Cloud de bază (servere, sisteme de operare sau de stocare), dar poate avea un control asupra aplicațiilor în sine şi, în anumite cazuri, asupra setărilor de configurare ale mediului de lucru al aplicațiilor.



Virtualizare în Cloud

3) Infrastructură ca Serviciu (Infrastructure as a Service - Iaas) este un model de servicii furnizate către chiriaş, în care acesta din urmă are posibilitatea să configureze resursele şi instrumentele de procesare, stocare, calcul, comunicare în rețea etc.; chiriaşul are posibilitatea să dezvolte şi să ruleze module de software arbitrare, care pot include sisteme de operare şi aplicații. Chiriaşul nu poate administra nici controla infrastructura Cloud de bază, dar poate deține controlul asupra sistemelor de operare, stocării şi aplicațiilor dezvoltate, precum şi un control limitat asupra unor componente de rețea, precum dispozitivele firewall.

Un element forte al tehnologiei Cloud este virtualizarea, proces prin care un sistem de operare oaspete este instalat pe suportul unui sistem de operare gazdă. Imaginea de mai sus este lămuritoare în acest sens.

3. Despre Baze de Date NoSQL

Reprezintă o nouă modalitate de stocare a informației, fără schemă şi fără tabele de

conexiune, înregistrările fiind structuri arborescente de tip JavaScript.

Bazele de date NoSQL ca noua generație de baze de date ce îndeplinesc următoarele condiții: nu sunt relaționale, sunt distribuite, sunt open-source şi se caracterizează prin posibilitatea de o distribui orizontal efortul de calcul. Alte caracteristici ce trebuie menționate sunt lipsa unei scheme pentru a modela baza de date, lipsa suportului pentru replicare, colecțiile API destinate dezvoltatorilor, viteză crescută la interogarea seturilor de date şi stocarea unei cantități uriaşe de date[5]. Reversul îl constituie lipsa garanției ferme că o tranzacție a fost procesată corect (există riscul tentativelor de a accesa date care fie nu mai există, fie au suferit modificări ne-scontate), aceste disfuncții potențiale fiind absolut excluse în cazul bazelor de date clasice (relaționale), unde operațiile sunt de tipul "tot sau nimic".

Printre sistemele NoSQL din ziua de azi pot fi menționate cele open-source, precum Hbase, MongoDB, CouchDB, GTM[6]. Bazele de date NoSQL au atît modele mai simple decât cele SQL, dar şi modele mult mai complicate decât acestea. Ceea ce se doreşte însă este obținerea unei flexibilități mult mai mari.

O caracteristică importantă a bazelelor de date NoSQL este distribuirea pe orizontală a efortului de calcul, care presupune construirea "pe loc" şi nu "vertical". Distribuirea pe orizontală este utilizată atunci cînd există capacitatea de a rula mai multe instanțe pe servere, simultan. Este utilizată în momentul în care necesitățile aplicațiilor constau în trafic foarte mare şi conectare la un număr mare de evenimente. Soluțiile NoSQL pot fi uşor adaptabile la creşterea efortului ("scalability")[7]. Bazele de date NoSQL permit ca PC-urile să fie avantajos extinse, fără complexitatea şi costul "sharding", care ar implica partiționarea unor baze de date în mai multe tabele pentru a rula pe grupuri mai mari.

Unul din avantajele substanțiale aduse de sistemul de management NoSQL este procesarea distribuită a datelor datorită concentrării pe dinamica mecanismelor de interogare. Cât priveşte modelele de interogare în NoSQL, ele se bazează pe căutarea unor chei primare sau a unui câmp ID şi pe lipsa unei interogări pe alte tabele, ducând la un impresionant spor de viteză (prin modelul NoSQL, scrierea într-o bază de date de 50 GB este mai rapidă de 2500 ori față de tradiționalul MySQL).

Mai jos se prezintă o mostră perfect corectă de organizare a datelor. Se permite câmpuri omonime *doar dacă* sunt imbricate. Sintaxa *nu* permite "frați" omonimi.

```
{
        _id:
        ObjectId("5146bb52d852427006
        0001f3"),
        "acest cimp are valoare de
        tip text": "un text",
        "acest cimp are valoare de
        tip intreg": 25,
        "enumerare de texte": ["un
        text", "alt text"],
        "enumerare de structuri de
        date":
        [
              "nume obligatoriu desi
              valoarea tip struct e
              vida":{},
              "alt                nume
              obligatoriu":
              {
                    "cimp" : 23.0,
                    "alt       cimp":
                    "valoare"
              },
              "nume obligatoriu desi
              colectia este vida" :
              [],
              "nume obligatoriu desi
              colectia este vida-" :
              [],
        ]
}
```

4. Baza de date şi Serviciul Web

S-a studiat dezvoltarea unei aplicații de tip Serviciu Web destinate să asigure suportul pe partea de server ("server-side") a mecanismului de versionare a fişierelor sursă utilizate în proiecte software cu varii destinații:

[5] *** - NoSQL Databases; http://nosql-database.org/

[6] Shalom, N. - The Common Principles Behind The NoSQL Alternatives, December 2009; Blog post of 2009-12-15. http://natishalom.typepad.com/nati_shaloms_blog/2009/12/the-common-principlesbehind-the-nosql-alternatives.html.

[7] Welsh, M., Culler, D., Brewer, E. - An architecture for well conditioned, scalable internet services; Proceedings of the eighteenth ACM Symposium on Operating Systems Principles. New York, NY, USA : ACM, 2001 (SOSP '01), p. 230-243

Desktop, Web, rețea, baze de date etc. Serviciul este de tip ASP.net 3.5, dezvoltat în C#, Sistemul de Gestiune bazde de date fiind MongoDB, aparținînd tipologiei NoSQL. Aplicația Serviciu Web, Serverul MongoDB şi bazele de date rezidă în Cloud, rulînd pe maşină virtuală.

Serviciul primeşte cereri ("requests") de la distanță după rezolvarea (procesarea) cărora trimite un răspuns către aplicația client emițătoare. Cererile constituie acțiuni executate pe baza de date: autentificări/ autorizări, introducere de date, regăsiri de date, încărcare/ descărcare de fişiere.

Dialogul Serviciului Web cu Serverul de baze de date se realizează prin intermediul a două drivere (în fapt, biblioteci DLL de tip .net), MongoDB.Driver.dll şi MongoDB.Bson.dll, care pun la dispoziția dezvoltatorului clasele înzestrate cu metode (funcții asociate) care execută operațiuni de conectare (deconectare) la (de la) serviciul MongoDB şi vehicularea datelor. Aceste metode apelabile din cele ale Serviciului Web, oferă o modalitate "prietenoasă", simplă şi comodă de executare a unor operații complexe (în principiu, fiecare metodă procesează argumentele primite pentru a le insera într-un text de interogare de tip NoSQL cu format consacrat, pe care apoi îl pasează spre interpretare Serviciului Mongo DB). Reamintim că, după cum textul de interogare de tip SQL se prezenta în formă practic colocvială:

```
    SELECT [cimpuri, . . .] FROM
[tabelul,  . . .] WHERE [conditia]
    sau
    UPDATE tabel SET cimp1 = val1,
cimp2 = val2, . . . WHERE conditia
```

textul de interogare de tip NoSQL se prezintă sub formă de sintaxă JavaScript:

```
db.dezvoltatori.find
(
    {
       experienta:
       {
             $elemMatch: {
       limbaj: "C++", ani: {
       $gt: 4 } }
       }
    }
)
```

În acest caz, în tabelul (în vocabularul NoSQL tabelul poartă numele de colecție) *"dezvoltatori"* se caută toate înregistrările care satisfac condiția referitoare la informația de tip *experiență*: să fi programat în *limbajul C++ cel puțin 4 ani*. Cuvintele în culoare albastră sunt **CONSACRATE** (cuvinte-cheie). Rezultatul obținut (setul de înregistrări care întrunesc condiția de căutare) va fi o înşiruire de formate arborescente de tip JSON (Java Script Notation Object – structură arborescentă ale cărei elemente sunt delimitate de o pereche de acolade.

Structurarea bazei de date

Fiecare bază de date poartă numele companiei care va utiliza acest depozit de cod, şi are nume unic. În componența unei baze există următoarele tabele:

Proiecte; fiecare proiect conține date proprii şi o listă cu numele dezvoltatorilor care participă la elaborarea lui, un dezoltator putînd participa la mai multe proiecte simultan. Numele fiecărui proiect este unic în cadrul bazei (Companiei);

Dezvoltatori - fiecare dezvoltator are ataşate date proprii (parolă, drepturi etc.) şi are nume unic în cadrul companiei, de aceea acest tabel (colecție) a fost înzestrat cu restricția de a nu accepta duplicate pentru cîmpul nume. Astfel, orice tentativă de a insera un dezvoltator omonim cu unul deja existent va emite un string de eroare;

Fişiere - fişierele fiind informații de tip BLOB (Binary Large Object – obiect binar de mari dimensiuni), vor fi stocate automat într-o sub-secțiune (care poate fi denumită după voie) a zonei consacrate de tip GridFileSystem. Fiecare intrare de tip BLOB are nume propriu şi are ataşat un segment propriu de tip metadate, unde se memorează sub formă de JSON toate elementele de interes:

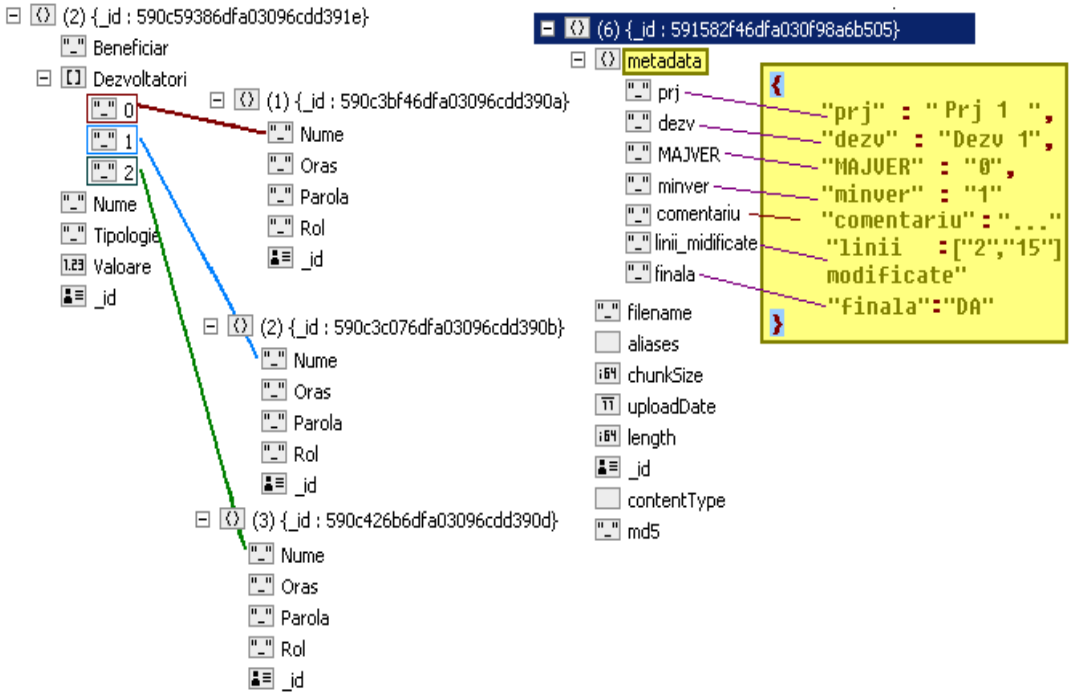
- proiectul (numele) căruia aparține fişierul curent;
- dezvoltatorul (numele) care a încărcat fişierul curent;
- versiunea MAJORĂ: string de la "0" la "9";
- versiunea minoră: string de la "0" la "9";
- comentariu opțional, de tip text;
- şirul indecşilor liniilor care nu mai concordă exact cu cele respectiv de acelaşi

index din versiunea imediat precedentă;

- indicativ că versiunea este finală (versiunea următoare trece în următoarea treaptă MAJORĂ.

Numele fişierelor nu trebuie să fie unice.

Trebuie menţionat că mărimea (volumul) unei baze de date utilizate într-un sistem de versionare creşte foarte repede, fiindcă e de presupus că un număr important de dezvoltatori încarcă des fişiere – aceasta în condiţiile în care, în mod normal, nimic nu se şterge! Din acest motiv, soluţia NoSQL este preferabilă deoarece sistemul de gestiune MongoDB are capabilitatea de a diviza *automat* o bază de date în sub-componente



Conexiuni între categorii de informaţie

(“shards”) care se pot *eventual transfera* pe alte maşini, unde se vor afla sub gestiunea altor instanţe fizice a serverului de baze de date.

În această situaţie, conjugarea soluţiei NoSQL cu tehnologia Cloud reprezintă varianta optimă de gestiune şi lucru, ţinînd cont de faptul că maşinile auxiliare sus-amintite rezidă pe exact aceeaşi platformă *fizică* de rulare, adică aceea pusă la dispoziţie de *unitatea Cloud* !

Mai jos se exemplifică prezenţa în subsecţiunea “SURSE” a zonei GridFS a două versiuni consecutive ale aceluiaşi fişier, din acelaşi proiect, dezvoltat de către aceeaşi persoană., *fiecare variantă avînd propriul conţinut fizic, binar*. Toate componentele de tip Binar (BLOB) ale unui document (înregistrare) sunt memorate, în universul NoSQL, nu în vreun tabel, ci într-o zonă dedicată a bazei.

Tabel. Ilustrarea memorării fişiereleor într-o subsecţiune a zonei GridFileSystem a bazei de date

Name	id	Metadata
Fisier14.cpp	5915821e6dfa030f98a6b4fb	{ "prj" : "**Proiect 1**", "dezv" : "**Dezvoltator 1**", "MAJVER" : "0", "minver" : "0", "comentarii" : "...", "linii modificate" : [ ] }
Fisier14.cpp	591583046dfa030f98a6b507	{ "prj" : "**Proiect 1**", "dezv" : "**Dezvoltator 1**", "MAJVER" : "0", "minver" : "1", "comentarii" : "...", "linii modificate" : ["2", "15", "43"] }

Fiecare variantă N de fişier, odată încărcată pe server, este comparată linie cu linie cu versiunea *imediat* precedentă (N-1). Fiecărei linii din versiunea (N-1) care nu mai concide exact cu linia corespondentă din proaspăt-sosita versiune curentă (N), i se va consemna indexul în secţiunea de metadate a versiune N. De asemenea, în caseta de dialog de pe aplicaţia client care execută operaţia de încărcare există posibilitatea de a consemna că fişierul curent este variantă finală. Cînd fişierul este introdus în baza de date, versiunea asociată va fi incrementul versiunii celei mai recente pentru înregistrarea care are respectiv aceleaşi nume pentru: fişier, proiect, dezvoltator.

În încheierea secţiunii dedicate bazei de date, menţionăm că fiecare înregistrare (document), indiferent de tabelul (colecţia) unde rezidă şi indiferent că este BLOB sau structură de date native, se bucură de propriul identificator unic, adică un string de 24 de caractere, generat aleatoriu, automat, imediat la inserare, de către Serverul MongoDB.

Serviciul Web

Reprezintă un DLL de tip .net care conţine funcţionalitatea necesară schimbului de informaţii cu Serverul MongoDB. Acest DLL funcţionează în cadrul Serviciului Windows w3wp.exe, care, la rîndul lui, este configurat să gestioneze una sau mai multe aplicaţii Web.

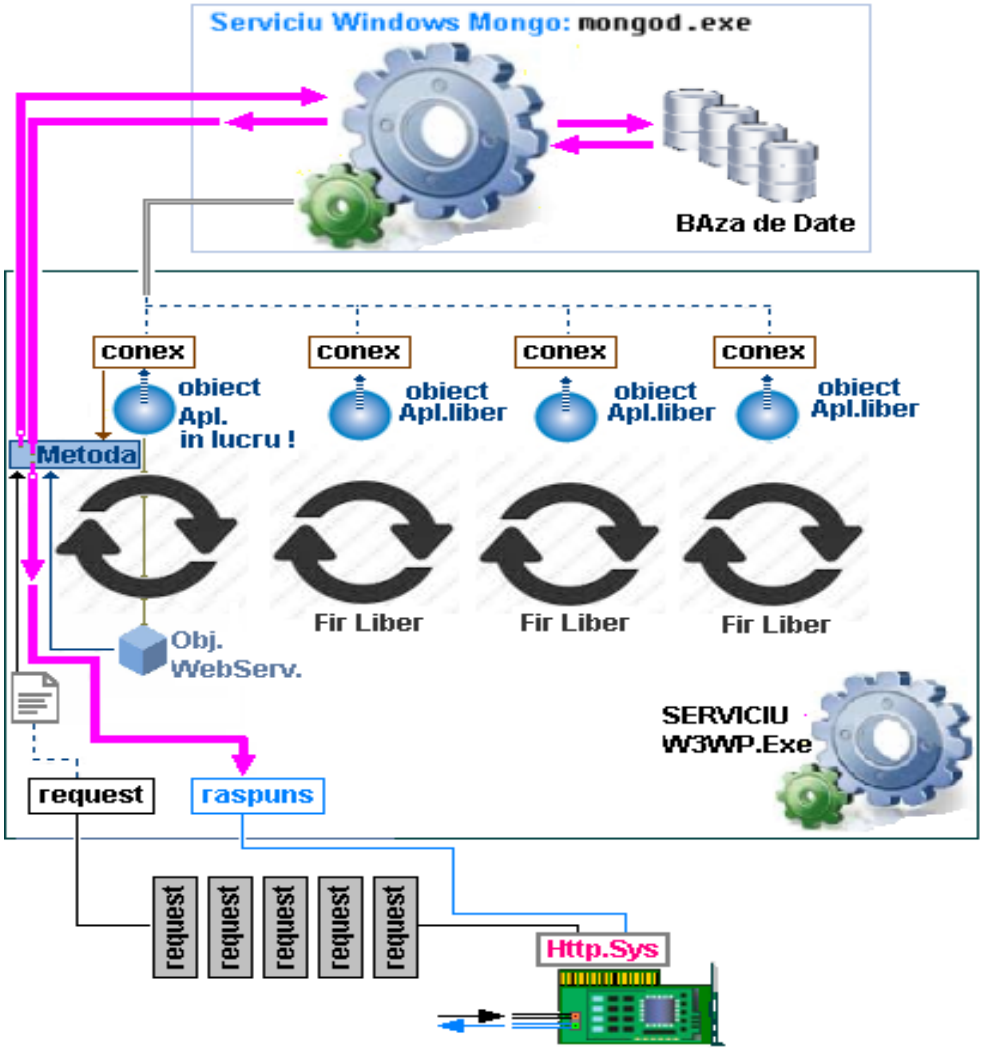
Emiterea unei cereri (apelarea serviciului Web) de pe aplicaţia client va duce plecarea către destinaţie a unui mesaj (şir de octeţi - buffer) care conţine numele funcţiei (metoda ataşată serviciului Web) care se doreşte a fi invocată plus lista de valori ale argumentelor. Odată mesajul ajuns cu succes pe Server, Motorul asp.net - care este încărcat tot în cadrul serviciului Windows de gestiune w3wp.exe şi care "asigură asistenţă de execuţie" pentru toate aplicaţiile asp.net asociate instanţei serviciului Windows amintit imediat mai sus - despacheteaza mesajul şi invocă metoda specificată cu valorile precizate ale argumentelor. Desigur, la prima consumare (accesare) a serviciului Web, biblioteca serviciului Web (exact DLL la care s-a făcut referire la începutul paragrafului) este încărcată în memorie (de către Motorul asp.net), unde rămîne intactă pînă la o eventuală actualizare sau pînă la oprirea serverului.

Motorul asp.net creează la startarea aplicaţiei Web un bazin (pool) de obiecte de tip HttpApplication şi un bazin de fire de execuţie (Threads) care stau în aşteptare (*rezidente permanente în memorie*) spre a fi selecţionate să rezolve cererea curentă. După ce tot requestul a ajuns cu bine pe server (a NU se uita că cererea este un POST, prin urmare una fără limitare de dimensiune), Motorul asp.net execută *în principiu*:

- alege un fir de execuţie liber (săgeţi circulare, în figura de mai jos) pe a cărui stivă se va procesa requestul.
- creează un efemer obiect de tip HttpContext (în care comasează informaţia venită prin request şi în care creează un temporar buffer de răspuns) pe care îl asociază firului.
- alege un obiect de tip HttpApplication (sferă albastră, în figura de mai jos), găsit liber, pe care îl asociază contextului de mai sus; acum obiectul HttpApplication este apt să declanşeze operaţiunea de rezolvare a cererii – adică lanţul de invocări subsecvente consacrate **care** se execută din interiorul (pe stiva) acestui fir pus la lucru. *Principial*, sunt enunţate mai jos:
  - se creează un efemer (temporar, existent strict pe durata cererii curente) obiect de tip serviciu Web (cub albastru, în figura de mai jos).
  - se desface mesajul şi se se invocă metoda cu valorile respectiv consemnate pentru argumente; se returnează valoarea către client.

În imaginea de mai jos, am reprezentat:

- săgeată plină subţire = pasare de argument către funcţie; săgeată plină lată = apel de funcţie/returnarea de valoare;
- linie întreruptă = arată către o adresă; linie triplă = pipeline (conductă) între două procese Windows.



Serviciul Web (sub Serviciul Windows W3Wp.Exe) primeşte o cerere, o rezolvă şi apoi trimite răspuns

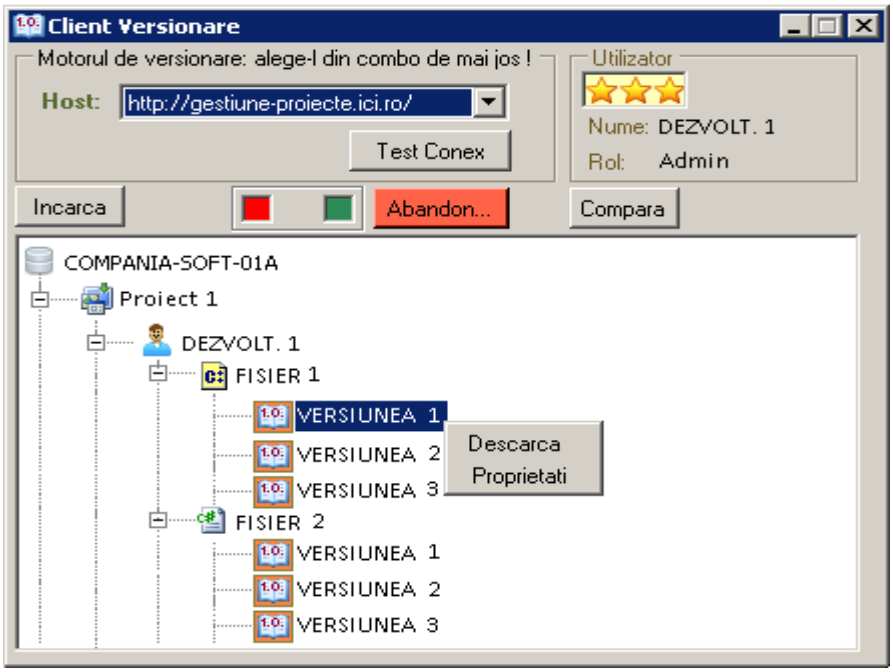
S-a considerat util ca fiecare obiect de tip HttpApplication, fiind rezident permanent în memorie pe durata aplicației, sa aibă un cîmp de tip conexiune, legat la serverul MongoDB. Acest cîmp (această conexiune) va fi pasată oricărei metode (a serviciului) care dialoghează cu baza de date.

Aplicația Client

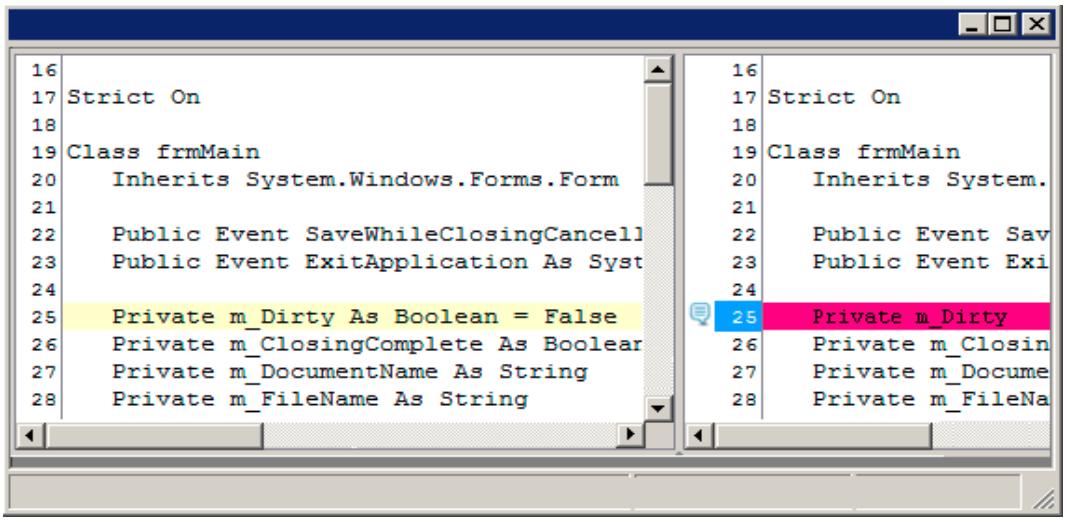
Reprezintă un EXE de tip .net, care conține funcționalitatea necesară conectării la serviciul Web aflat în Cloud, trimiterea, descărcarea şi compararea versiunilor unui fişier anumit, aparținînd unui proiect dat şi dezvoltat de către o anumită persoană.

Prezentăm câteva caracteristici:

- operațiile de dialog cu serverul se fac pe fire de execuție, putând fi oricînd abandonate şi reluate. Funcționarea firelor este semnalată de "aprinderea" / "stingerea" alternativă a celor 2 "LED"-uri, rosu şi verde;
- încărcarea informației din depozitul de cod se face într-un control de tip arbore, traseul de filiație fiind "Compania->Proiectul->Dezvoltatorul->Fişierul->Versiunile;



Interfața client



Compararea versiunilor

- compararea a două versiuni, prin descărcarea fişierelor asociate şi vizualizarea lor într-o fereastră separată;
- încărcarea de fişiere sursă, în prealabil alegînd dintr-un combobox numele proiectului asociat şi, eventual, setînd comentarii. Evoluția trimiterii fişierului se vizualizează în bară de progres. Memorarea în baza de date se face numai după primirea confirmării de primire a fişierului. Descărcarea pe server a segmentelor fişierelor se face de către un interceptor de tip HttpModule instalat în sub-directorul /Bin al serviciului Web şi consemnat în fişierul de configurare Web.config;
- vizualizarea centralizată a proprietăților unei versiunei (dată, autor, comentarii etc.);
- nu este nevoie ca serviciul Web să fie o aplicație pe protocol https, pentru că orice informație (mai puțin header-ele) poate fi vehiculată doar criptat, după voie. Dacă se dorește criptarea inclusiv a antelelor de cerere, trebuie ca serviciul Web să fie o aplicație pe protocol https;
- la startarea aplicației se cere

autentificare prin nume şi parolă, acestea trimiţindu-se criptate, inclusiv pe http simplu.

Compararea versiunilor arată diferenţa dintre două versiuni ale aceluiaşi fişier. Acolo unde liniile nu sunt exact la fel, în dreapta se afişează cu fundal roşu.

5. Concluzii

Proiectele software devin din ce în ce mai complexe, lucru care impune modularizarea activităţii de dezvoltare, adică împărţirea sarcinilor de programare pe echipe. În vederea asigurării autonomiei de lucru între echipe se folosesc instrumentele de versionare: principial, ele constau într-o bază de date unde se memorează după criterii bine determinate stadiul curent (versiunea) al fiecărui modul (fişier cu linii de cod) care compune proiectul. În felul acesta se asigură atît reversibilitatea schimbărilor executate în fiecare fişier (în fiecare moment, orice echipă de programatori, în mod independent de celelalte echipe, poate recupera orice fişier într-o anumită etapă de evoluţie, în vederea restudierii şi reutilizării, păstrînd intacte versiunile anterioare), cât şi evidenţa oricăror schimbări survenite în cod (pentru fiecare fişier sursă se ştie cine, cînd şi ce modificări a făcut). Astfel, instrumentul de versionare menţine în mod structurat, sigur şi uşor accesibil "traiectoria" dezvoltării unui proiect.

Sistemele de versionare accesabile prin reţea (adică la distanţă) pot fi centralizate (fiecare maşină de lucru se bazează doar pe sistemul de versionare la distanţă) sau distribuite (fiecare maşină de lucru are propriul sistem de versionare local, conectat însă la cel principal, aflat la distanţă).

În sprijinul acestui model de gestionare a proiectelor vin două tehnologii: Cloud Computing şi Sistemele de baze de date de tip NoSQL, fiecare avînd atît avantaje cât şi zone perfectibile.

Cloud Computingul este un model convenabil gîndit să permită accesul la cerere, prin reţea, la o grupare de resurse de calcul configurabile (de exemplu reţele, servere, echipamente de stocare, aplicaţii şi servicii), care pot fi puse la dispoziţia utilizatorului în mod rapid şi cu un efort minim de administrare sau interacţiune cu prestatorul acestor servicii. Cloud oferă viteză şi siguranţă în funcţionare, în timp ce bazele NoSQL excelează prin timpi extrem de scurţi de răspuns în condiţiile interogării unui număr foarte mare de înregistrări.

Baza de date NoSQL ignoră principiile Sistemelor Relaţionale, adică nu stochează conexiuni folosind tabele, ci foloseşte chei de identificare în cadrul înregistrărilor însele. Datele pot fi regăsite în funcţie de cheile atribuite. Acest tip de baze de date este prin natura sa scutit de necesitatea de a normaliza datele şi de a stoca conexiunile în tabele dedicate - aducând astfel performanţe sporite aplicaţiilor care le folosesc. De asemenea, acest tip de baze de date îmbunătăţeşte inclusiv răspunsul la schimbările ce pot surveni de-a lungul timpului în schema bazei. Într-un sistem relaţional nu există flexibilitatea necesară pentru a asimila modificări în modelul de date. Faptul că bazele de date NoSQL nu au o schemă de date fixă (rigidă), le face să fie flexibile, dând astfel posibilitate dezvoltatorilor să opereze schimbări de structură "din mers", la nevoie, fără să fie forţaţi la schimbări majore în cod.

Prin urmare, baza de date de tip NoSQL se va constitui în depozit de coduri sursă, deservind un ansamblu structurat de facilităţi de găzduire / procesare pentru cantităţi mari de cod software, pentru uz public sau privat. Aceste resurse de cod, cărora sistemul le memorează şi gestionează versiunile intermediare şi modificările făcute pe parcurs, sunt utilizate în proiecte open-source sau în alte proiecte multi-dezvoltator.

Scopul urmărit prin prezentul proiect constă în valorificarea performanţelor ridicate ale unui sistem de baze de date de tip NoSQL rezidente în Cloud, prin utilizarea acestora în cadrul unor soluţii de stocare/gestionare pentru codurile sursă ale proiectelor software.

Sistemul de versionare a cărui realizare s-a studiat comportă o componentă server-side de tip serviciu Web şi o componentă client, de tip desktop, cu interfaţă grafică. La dezvoltarea modulelor de interacţiune cu Baza de date s-a utilizat o pereche de drivere (DLL) de tip .net - importabile în proiectul principal – care oferă funcţii integrate pentru crearea, editarea şi regăsirea seturilor de informaţie.

BIBLIOGRAFIE

1. *** - NoSQL Databases; http://nosql-database.org/
2. *** - Semantic Versioning. http://semver.org.
3. *** - Software Versioning. https://technick.net/guides/software/software_versioning/.
4. NIST- The Definition of Cloud Computing (Sept. 2011); Special Publication, pp. 800-145.
5. **PRESTON-WERNER, TOM**: Semantic Versioning 2.0.0. Creative Commons, 2013. http://semver.org/spec/v2.0.0.html.
6. **SHALOM, N.:** The Common Principles Behind The NoSQL Alternatives, December 2009; Blog post of 2009-12-15. http://natishalom.typepad.com/nati_shaloms_blog/2009/12/the-common-principlesbehind-the-nosql-alternatives.html.
7. **WELSH, M.; CULLER, D.; BREWER, E.:** An architecture for well conditioned, scalable internet services; Proceedings of the eighteenth ACM Symposium on Operating System s Principles. New York, NY, USA. ACM, 2001 (SOSP '01), pp. 230-243.