UDC: 004.738.52.032.26 INFO M: str. 15-21

POSTIZANJE INTEROPERABILNOSTI LEGACY SISTEMA POSREDSTVOM WEB SERVISA
ACHIEVING INTEROPERABILITY OF LEGACY SYSTEMS USING WEB SERVICES

Aleksandar Ilić, Performance Technologies SRB
Nikola Lončar, Wings Software
Slađan Babarogić, Fakultet organizacionih nauka,
Univerzitet u Beogradu

REZIME: U savremenom svetu sve više poslovnih procesa biva prožeto informacionim tehnologijama. Cilj je unaprediti procese rada i komunikacije korišćenjem softverskih sistema. Takve tendencije zahtevaju dodatne resurse čijim izostankom dolazi do stvaranja Legacy sistema. Glavna karakteristika svih Legacy sistema je diskontinuitet u razvoju koji za posledicu ima ograničenu mogućnost interakcije sa drugim softverskim sistemima. U ovom radu se predstavljaju konkretna rešenja koja omogućavaju integraciju takvih sistema sa savremenim softverskim rešenjima. Omogućavanje interoperabilnosti putem integracije sa Web servisima unosi nove mogućnosti i produžava životni vek Legacy sistema.

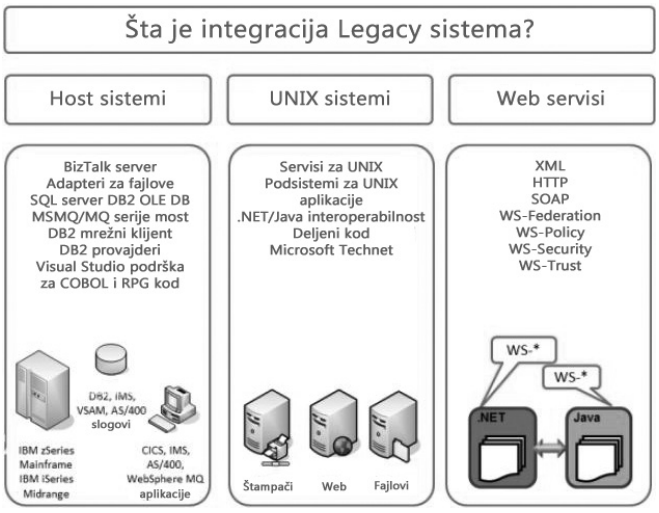
KLJUČNE REČI: Legacy sistemi, interooperabilnost, web servisi

ABSTRACT: In modern times, more and more business processes becomes permeated with information technologies. The aim is to improve work processes and communications using software systems. These trends require additional resources whose absence leads to the creation of Legacy systems. The main feature of all Legacy systems is discontinuity in development which results in a limited ability of interaction with other software systems. This paper presents concrete solutions that enable the integration of such systems with modern software solutions. Enabling interoperability through integration with Web Services brings new features and extends the life of Legacy systems.

KEY WORDS: Legacy systems, interoperability, Web Services

1. UVOD

U savremenom svetu ne postoji segment privrede koji se brže razvija od informacionih tehnologija. Takvo dinamičko okruženje dovodi do stalne potrebe za usavršavanjem postojećih i uvođenjem novih informacionih sistema. Naravno takve smene nije uvek jednostavno izvesti pa se dešava da određeni sistemi zadrže svoj oblik i tehnologiju duže vremena od planiranog. Takav diskontinuitet razvoja dovodi do stvaranja takozvanih legacy sistema. U najjednostavnijem opisu oni predstavljaju nezavisnu celinu izopštenu iz modernih tekovina razvoja informacionih tehnologija, poput pustog ostvra. Razlog postojanja takvih sistema i njihovog daljeg korišćenja se krije u korisnosti i funkcionalnosti koju poseduju. Procenjuje se da je u svetu i dalje aktivno oko 180-200 milijardi redova programskog koda u legacy sistemima [1]. Pre svega oni i dalje obavljaju svoju inicijalnu fukciju u obliku koji odgovara korisnicima, koji su prihvatili radno okruženje takvih sistema i savladali njihovo korišćenje. Problem nastaje kada se od takvih sistema očekuje da pruže i set novonastalih funkcionalnosti razvijenih kasnije tokom razvoja informacionih tehnologija. Takva situacija postavlja dilemu pred korisnike tih sistema da li da se energija i novac uloži u integraciju i premošćavanje jaza doradom već postojećih sistema takozvanom integracijom (Slika1) ili da se izgradi potpuno novi sistem i tako reši taj problem. Sistemi razvijeni pre pojave savremenih programskih jezika i modernih baza podataka, imaju veliki broj ograničenja. Izostanak interoperabilnosti kod takvih sistema predstavlja glavni nedostatak usled sve veće potrebe za međusobnom komunikacijom više različitih sistema. Problemi sadašnjih legacy sistema se prvenstveno ogledaju u stepenu konkurentnosti koje moraju da isprate.



Slika 1. – *Integracija legacy sistema [2]*

Današnja softverska rešenja se sve više okreću ka kreiranju servisa koji pružaju usluge ili više tipova informacionih usluga korisnicima putem računarskih mreža, koristeći razne mrežne protokole. Takav pristup kreiranja sistema se skraćeno naziva SOA (*Service oriented arhitecture*) i koja predstavlja jedan od mogućih načina kod ostvarivanja interoperabilnosti. On omogućava da se sistem kao celina izvede iz niza zasebnih funkcija. Korisnik upotrebljava neke od tih funkcija preko standardizovanih poruka, u zavisnosti od njegovih potreba. Web servisi su produkt SOA. Kao takvi oni ispunjavaju sve navedene uslove i omogućavaju kreiranje servisa kao pružalaca usluga. Većina legacy sistema nije imala mogućnost da svoje funkcionalnosti prenese putem interneta svojim korisnicima. Integracija legacy sistema sa web servisima doprinosi

interoperabilnosti takvih sistema. Takavu integraciju bi bilo moguće relativno jednostavno sprovestu u delo da ne postoje veoma kruta ograničenja unutar samih legacy sistema. Pre svega takva integracija mora da ispoštuje više kriterijuma među kojima na prvom mestu se nalazi pouzdanost, odnosno da se podaci transferuju iz legacy sistema do web servisa bez grešaka. Takav sistem treba da omogući dvostranu komunikaciju kao i da njegovo korišćenje ne dovede u pitanje već postojeće funkcionalnosti legacy sistema.

2. INTEROPERABILNOST

Prema definiciji IEEE-a interoperabilnost predstavlja mogućnost razmene informacija dva ili više sistema ili komponenti, kao i upotrebu tako razmenjenih informacija [3]. Druga definicija, a prema standardu ISO/IEC 2382-01, interoperabilnost definiše kao sposobnost komuniciranja, izvršavanja programa ili razmena podataka između raznih funkcionalnih jedinica na način koji zahteva od korisnika da imaju malo ili nikakvo poznavanje jedinstvenih karakteristika tih jedinica [4].

Ukoliko su dva sistema sposobna za međusobnu komunikaciju i razmenu podataka, oni ispoljavaju sintetičku interoperabilnost. Osnovu takvog sistema predstavlja jasna definisanost formata podataka i komunkicaionih protokola. Uopšteno, XML i SQL standardi pružaju sintetičku interoperabilnost. Sintetička interoperabilnost je neophodna za bilo kakav dalji pokušaj postizanja interoperabilnosti.

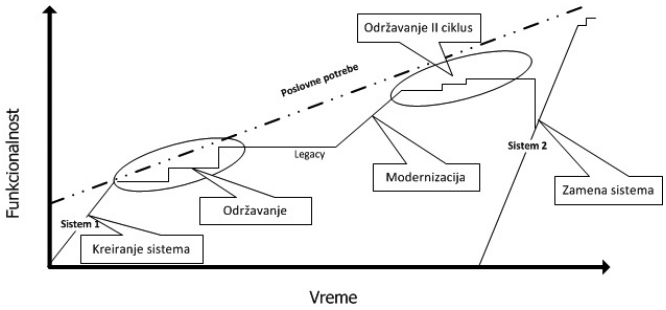
Pored sposobnosti razmene podataka dva ili više računarskih sistema, semantička interoperabilnost predstavlja sposobnost tih sistema da automatski tumače razmenjene podatke, na smislen i ispravan način, kako bi proizveli korisne rezultate kao što je određeno od strane krajnjih korisnika oba sistema. Da bi se postigla semantička interoperabilnost, obe strane moraju poštovati zajednički referentni model razmene informacija.

3. LEGACY SISTEMI

Od momenta nastanka i stavljanja u upotrebu softvera kreće borba sa vremenom kako bi se usporilo njegovo zastarevanje. Najčešće korišćena krilatica koja glasi: „legacy kod je kod koji je napisan juče“ sve više dobija na svojoj aktuelnosti u dinamičnoj sferi razvoja informacionih tehnologija.

U opštem opisu legacy sistemi se definišu kao prevaziđene metode, tehnologije, kompjuterski sistemi ili aplikativni softver koji se i dalje nalazi u upotrebi i pored toga što je razvoj informacionih tehnologija kreirao mnogo efikasnije i bolje sisteme za obavljanje tih zadataka [5].

Postavlja se pitanje zbog čega se javljaju legacy sistemi, kada se očekuje od korisnika da svoje informacione sisteme uvek drže aktuelnim i u stanju da isprate moderne tokove informacionih tehnologija. Teško je naći odgovor na takvo pitanje sa preciznošću, ali neki od bitnih faktora su finansijski i ljudski resursi koji su bili dostupni u datom vremenskom periodu. Ukoliko se sistem nije modernizovao tokom vremena moguće je da će u jednom momentu doći do diskontinuiteta u praćenu modernih u praćenju tokova. Takav jaz nastao u razvoju dovodi do stvaranja legacy sistema (Slika2).



Slika 2. – *Životni ciklus softverskih sistema [5]*

Legacy sisteme određuje njihova smanjena prilagodljivost i nemogućnost da isprate nove zahteve svojih korisnika. Pored gore navedenih mana, legacy sistemi mogu da imaju i sledeće nedostatke u [5]:

- Arhitekturi: klasični problemi loše arhitekture sistema su redudantnost podataka, loša sigurnost. Problemi oko kompatibilnosti sa novijim hardverom.
- Dizajnu: ovakvi problemi se ogledaju u lošoj koncepciji programa, npr. grafički korisnički interfejs direktno pristupa podacima ili se podaci ne mogu dobijati u realnom vremenu.
- Kvalitetu koda: ogleda se u nedostatku konvencije imenovanja i pisanja koda same aplikacije. Nedovoljno kvalitetna dokumentacija. Prisustvo delova koda koji nemaju funkciju.

Kada se korisnici određenog legacy softvera odluče da reše probleme zastarelosti sistema koji je u upotrebi, mogu koristiti neku od sledećihmetoda za rešavanje problema [5]:

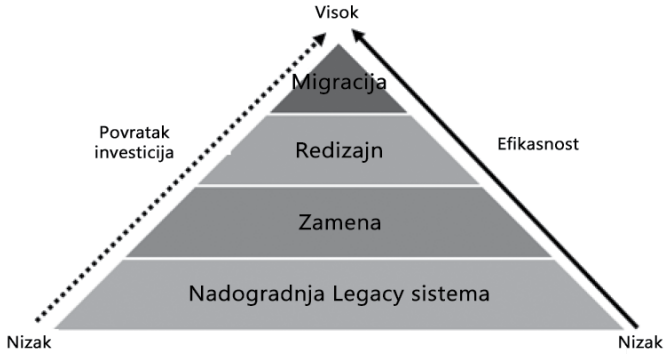
1. Nadogradnja legacy sistema
2. Zamena
3. Redizajn
4. Migracija

Nadogradnja legacy sistema predstavlja mogućnost da se sam programski jezik starog sistema i tehnologija unapredi putem prebacivanja na noviju verziju tih alata.

Ukoliko postoji rigidno jezgro legacy sistema koje je bez mogućnosti promena može se izvršiti zamena samo tog dela. Najbolji primer za to je zamena *backend*-a odnosno sloja podataka. Ukoliko su korišćene stare baze podataka moguće je odraditi njihovu zamenu u novije tipove poput MSSQL-a ili Oracle baza. Naravno, ovakva zamena zahteva odgovarajuću strukturu samog legacy sistema, odnosno, višeslojnu arhitekturu sa jasnim razgraničenjem između slojeva.

Redizajn obuhvata zadržavanje jezgra i srednjeg sloja dok je predmet promena *frontend*. On se redizajnira korišćenjem najnovijih tehnologija u skladu sa trenutnim rešenjima u primeni.

Poslednja opcija za prevazilaženje legacy krize jeste migracija.Migracija obuhvata zamenu kompletnog legacy sistema novim sistemom. Ovo možda deluje kao najbolja opcija, ali je migracija ujedno i najzahtevnija sa stanovišta troškova i vremena izrade.



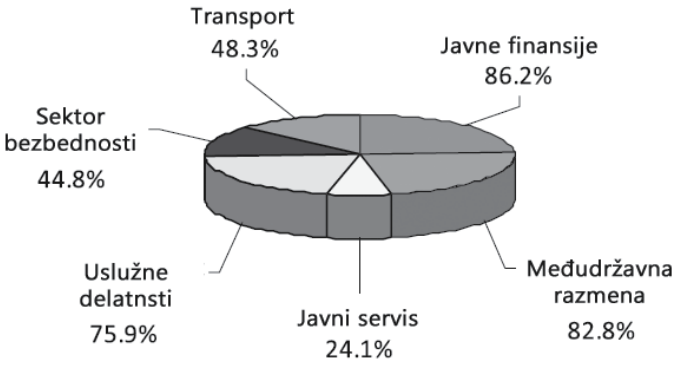
Slika 3. – *Odnos efikasnosti opcija za prevazilaženje legacy problema [5]*

Stepen efikasnosti metode kao i nivo isplativosti investicije zavisi od izabranog rešenja (Slika3). Svaka od navedenih metoda ima svoju primenu u odgovarajućim slučajevima, a na korisnicima datih legacy sistema je da naprave izbor u zavisnosti od njihovih zahteva i potreba.

Procesu modernizacije treba pristupiti veoma detaljno uzimajući u obzir sve aspekte kako starog legacy sistema tako i mogući izbor novih tehnologija sa kojima će se obavljati modernizacija.

Gotovo da ne postoji oblast jedne države i njenog društva gde nisu zastupljeni legacy sistemi. Počevši od državnih institucija poput školstva, zdravstva, policije pa do privatnog sektora. U pitanju su velike organizacije i poslovni sistemi koji su u potpunosti zavisni od pratećeg legacy softvera. Razvoj tako obimnog softvera je umeo da traje i više godina da bi sam proizvod na početku korišćenja već bio prevaziđen u pojedinim segmentima sistema.

Ukoliko se pogleda struktura gde je takav softver najviše rasprostranjen vidi se da je on najviše zastupljen u velikim organizacijama kao što su državne institucijije i administraciji (Slika4).



Slika 4. – *Zastupljenost legacy sistema po segmentima SAD [6]*

4. WEB SERVISI

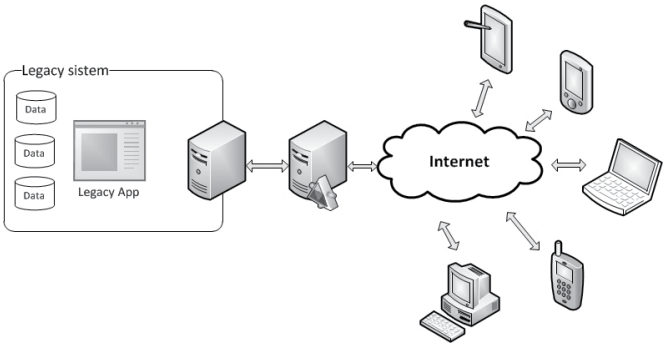
Prema definiciji W3C4-a, web servis (engl. *web service*) se definiše kao softverski sistem dizajniran da podrži interoperabilnu mašinsku interakciju preko mreže [7].

Tehnologija web servisa počinje da se razvija sa zajedničkim razmatranjem firmi kako da na jednostavan i efikasan način vrše razmenu podataka, u cilju formiranja elektronskih tržišta. Ako se malo detaljnije protumači ovaj zahtev, može se uočiti da isti otkriva problem međusobne razmene podataka na način koji je prihvatljiv za sve učesnike procesa.

Ovakav sistem za razmenu podataka, koji je u isto vreme dovoljno univerzalan da zadovolji raznovrsne potrebe korisnika i dovoljno konkretan za svaki korisnički slučaj, ističe web servis kao moćno sredstvo za komunikaciju bilo koja dva ili više sistema, bez ograničenja vezanih za tip i platformu korisnika web servis usluga.

5. POVEZIVANJE WEB SERVISA I LEGACY SISTEMA

Da bi se legacy sistem održao aktivinim i kako bi se stvorile nove mogućnosti, povezivanje sa web servisima se nameće kao logično rešenje. Takvom kombinacijom se ostvaruju prednosti korišćenja globalne mreže i dobijaju nove funkcionalnosti kao i širi dijapazon potencijalnih klijenata legacy sistema (Slika5).



Slika 5. – *Povezivanje legacy sistema i web servisa*

Ovakvom integracijom se omogućuje da korisnici dobiju novi kvalitet rada kao i veću efikasnost u radu. Prilikom realizacije ovakve integracije potrebno je voditi računa o sledećem:

- Arhitektura novog sistema – oslanjanje na upotrebu paterna za dizajniranje takvog sistema.
- Izbor tehnologije.

Paterni predstavljaju primere dobre prakse. Obuhvataju tri celine: kontekst, problem i rešenje. Primena paterna olakšava projektovanje softvera i čini ga standardizovanim.

6. STUDIJSKI SLUČAJ – INTEGRACIJA *WINGS* APP-A I WEB SERVISA

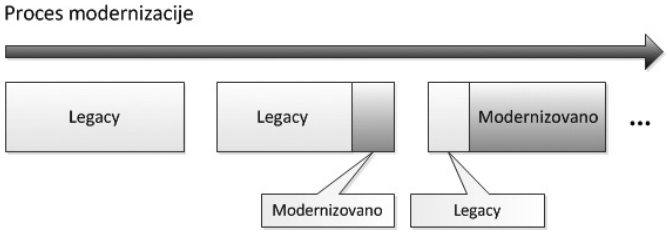
Projekat integracije ERP aplikacije koja se u upotrebi nalazi od 1988. godine. Kreiran u skladu sa tadašnjim okolnostima ostao je u upotrebi do današnjih dana. Kompleksnost ovakvog softverskog rešenja je otežalo njegov reinženjering tokom vremena, dok je razvoj interneta uslovio da se poslovanje proširi i na ovaj segment tržišta.

Brojna ograničenja su opterećivala upotrebljivost ovakvog sistema tokom vremena. Najveći nedostatci su se ogledali u tome da tehnologija koja je bila u upotrebi nije imala nikakva

poboljšanja od vremena nastanka prve verzije softvera. Glavni problemi se mogu sagledati u sledećem:

- Programski jezik Clipper - zvaničan razvoj prestao 1997. godine. U pitanju je programski jezik koji se koristi pod DOS operativnim sistemom i kompajlira za 16- bitne procesore. Dodatan problem predstavlja to što je Microsoft izbacio podršku za izvršavanje 16-bitnih aplikacija unutar operativnih sistema zasnovanih na 64-bita.
- Baza podataka koja je zasnovana na dBase III sistemu koji je razvijan od 1984. godine. Takav sistem je zasnovan na tabelama za skladištenje podataka bez referencijalnog integriteta sa ograničenim mogućnostima pretraživanja i manipulacije podataka. Ovakav sistem je bez mogućnosti izvršavanja transakcija i veoma lako bi moglo doći do narušavanja integriteta podataka.
- Grafičko okruženje programa se zadržalo unutar DOS standarda, bez mogućnosti njegovog proširivanja ili nadogradnje.
- Korišćenje interneta i servisa nije bilo moguće u izvornom obliku, odnosno, sam programski jezik nije posedovao takve mogućnosti.

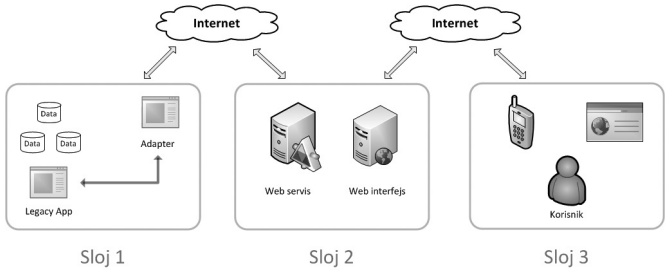
Kako bi se otklonilinedostaci, odlučeno je da se sistem unapredi kako bi izašao u susret novim zahtevima korisnika. Potrebno je stari ERP sistem integrisati sa web servisima kako bi se tako mogle koristi funkcionalnosti koje u osnovi zahtevaju korišćenje interneta. Za to vreme bi se projektovala nova savremena verzija osnovnog softvera. Tako bi se faza sistema podelila u tri celine (Slika6).



Slika 6. – *Koraci unutar procesa modernizacijelegacy sistema[8]*

Na ovaj način bi se prelazak legacy sistema u savremen sistem učinio što bezbolnijim po korisnike, a sam proces bi se organizovao po logičnim celinama.

Nakon analize potrebnog vremena, raspoloživih sredstava i resursa postavljen je plan realizacije projekta integracije. Osnovu plana čini arhitektura novog sistema (Slika7) koja bi omogućavala realizaciju postavljenih zadataka.

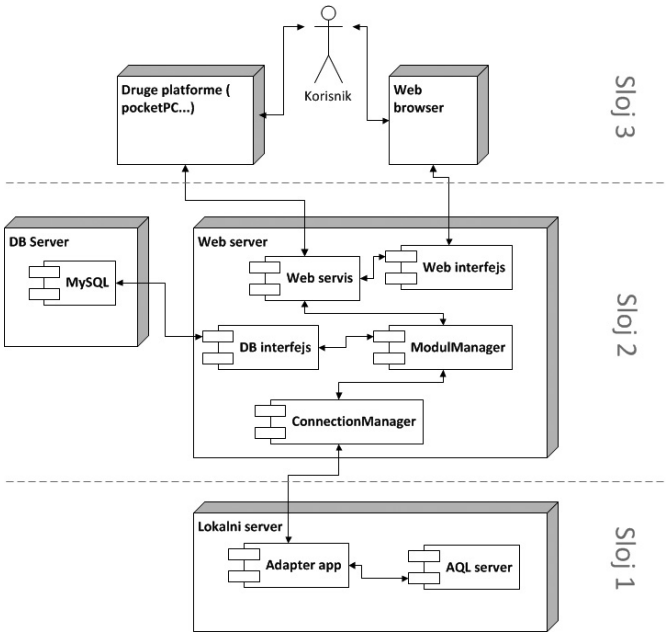


Slika 7. – *Arhitektura integracije*

Sistem je projektovan u tri sloja koja čine :

1. Adapter aplikacija koja treba da ima sledeće funkcije:
   - da prima zahteve od servera,
   - prosleđuje zahteve App-u,
   - čeka na odgovor i
   - vraća podatke nazad na server.
2. Server koji podrazumeva sledeće osobine :
   - mapiranje svih poziva ka legacy App-u,
   - transformacija podataka u željeni format (JSON, XML),
   - modularnost i
   - SOAP interfejs.
3. Klijentska strana:
   - podrazumeva kreirani web interfejs odnosno tanki klijent koji se izvršava u *browser*-u i
   - aplikaciju za mobilne uređaje.

Detaljnija podela na sastavne komponente sistema se vidi na slici 8.

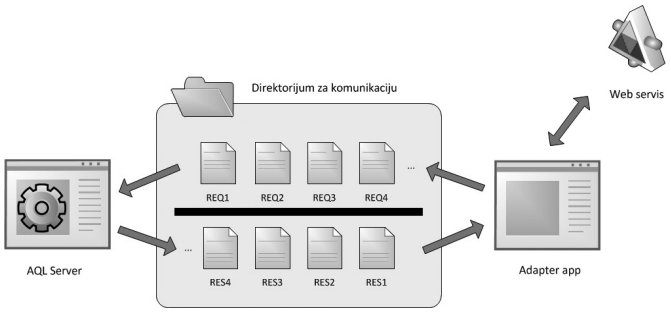


Slika 8. – *Komponente sistema integracije*

6.1. *Sloj 1 – adapter aplikacija*

Na strani legacy sistema potrebno je premostiti jaz u tehnologiji kako bi se omogućila razmena podataka. Za tehnologiju na kojoj će da se bazira aplikacija izabrana je .NET platforma i programski jezik C#. Životni ciklus komunikacije adaptera i legacy sistema je predstavljen na slici 9.

Na strani legacy sistema se nalazi poseban program za komunikaciju (AQL server) koji vrši monitoring nad određenim direktorijumom. Adapter aplikacija prima proziv od strane web servisa za određenim podacima koje je klijent tražio. Poruka se upisuje u direktorijum, nakon čega je preuzima AQL server, obrađuje i upisuje odgovor. Nakon toga Adapter aplikacija preuzimam odgovor, vrši njegovu grubu obradu u XML format i transferuje nazad na web servis.



Slika 9. – *Proces obrade zahteva od web servisa*

AQL server obrađuje fajlove linearno, odnosno redom po kojem su pristigli u direktorijum za komunikaciju. Ovakva obrada može da predstavlja usko grlo prilikom izvršavanja zahteva. Rešenje takvog problema je pokretanje više nezavisnih instanci AQL servera koji bi paralelno radili obradu zahteva. Time bi se dobilo na konkurentnosti izvršavanja i upošljavanju više resursa ukoliko je u pitanju arhitektura procesora sa više jezgara. Adapter aplikacija je u stanju da prima pozive asinhrono i ponaša se kao serverska aplikacija u odnosu na web servis. Za svaki zahtev se odvaja posebna nit koja čeka odgovor na svoj zahtev.

6.2. *Sloj 2 – Serverska aplikacija*

Da bi se obezbedila puna interoperabilnost legacy sistema potrebno ga je integrisati sa web servisom. Prvi korak predstavlja integracija legacy sistema i adapter aplikacije. Serverska aplikacija bi služila da prima zahteve od korisnika i prosleđuje ih adapeteru.

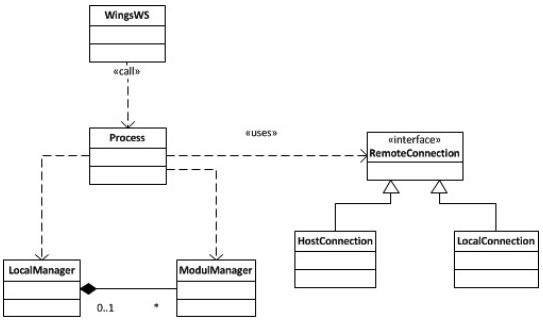
Zahtevi od serverske aplikacije su sledeći :

- Implementacija tehnologije web servisa,
- Modularna arhitektura,
- Brza dorada novih funkcija,
- Brza obrada zahteva,
- Pouzdanost.

Kako bi se ispunila zacrtana specifikacija za serversku aplikaciju odabrane su sledeće tehnologije :

- SOAP – WSDL web protokol,
- Apache / PHP 5.3 platforma,
- MySQL baza podataka,
- XML, XSLT, JSON

Arhitektura sistema je usklađena sa datim zahtevima kako bi se omogućilo lako dodavanje novih funkcionalnosti u sistem (Slika10).

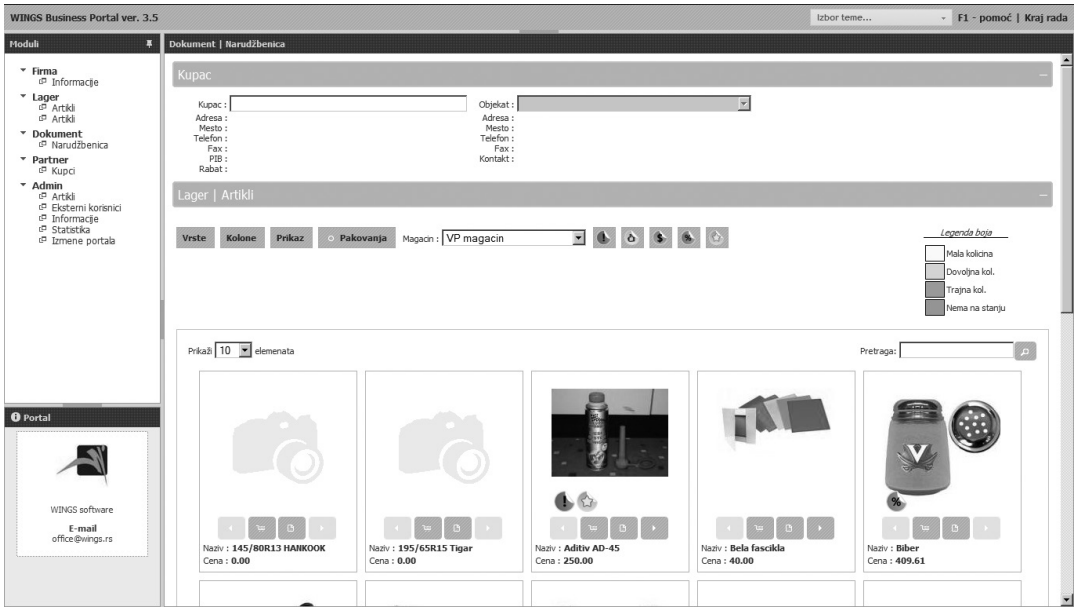


Slika 10. – *Dijagram klasa jezgra serverske aplikacije*

Proširenje funkcionalnosti se postiže na dva načina. Prvi način obuhvata imenovanje i dodavanje odgovarajućih XSL transformacija za svaku novu funkciju čiju obradu vrši *ModulManager*. Drugi način obuhvata nasleđivanje *LocalManager*-klase radi kompleksnije obrade podataka čije se krajnji rezultat takođe obrađuje putem XSL transformacije. Prijem podataka iz legacy sistema se obavlja posredstvom *RemoteConnectio-n*interfejsa čiju dalju obradu preuzima odgovarajući *Manager*.

6.3. *Sloj 3 – Web interfejs*

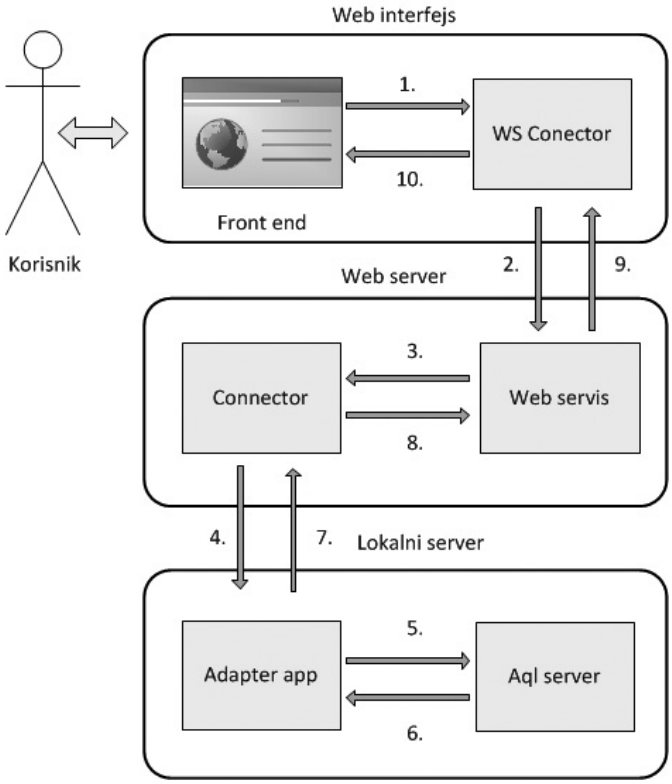
Korisnici putem *browser*-a mogu da pristipe sistemu koristeći web interfejs. On se zasniva na AJAX tehnologiji i *jQuery framework*-u, a u pozadini se oslanja na funkcionalnost web servisa. Izgled web interfejsa je prikazan na slici 11.



Slika 11. – *Web interfejs*

6.4. *Formati poruka*

Tok komunikacije između klijenta web interfejsa i legacy sistema ima više faza. Unutar svake od tih faza dolazi do obrade i menjanja poruka kako bi se komunikacija odvijala što efikasnije. Proces kretanja poruke možemo podeliti u 10 faza (Slika12). Da bi se lakše objasnile izmene poruka po fazama koristiće se slučaj komande za listanje partnera.



Slika 12. – *Faze kretanja poruke unutar sistema*

1. Korisnik (komercijalista sa identifikacionim brojem 5) unutar svog *browser*-a bira opciju za prikaz svojih partnera. Komanda se šalje iz *browser*-a na server u JSON formatu.

```
{
  "command":      "partner.kupac.svi",
  "output":       "json",
  "params":       [{"komercijalista": 5},
{"status":"K"}]
}
```

2. Nakon što poruka stigne na server, dolazi do njene obrade kako bi se poslala web servisu. Ona se prebacuje u XML format, a zatim pakuje unutar SOAP poruke i prosleđuje web servisu.

```
<command name="partner.kupac.svi" output="j-
son">
  <komercijalista>5</komercijalista>
  <status>K</status>
</command>
```

3. Web server prima komandu, vadi je iz tela SOAP poruke i daje dalje na obradu. Ona zatim se parsira i traži se odgovarajuća XSLT transformacija za nju.Nakon uspešne transformacije poruka se smešta unutar query taga sa atributima redni broj komande i tip upita (aql/sql). Nakon toga prosleđuje se modulu za komunikaciju.

```
<query id="1" type="aql">
  WITH Partneri
  QLIST 5, K
  END
</query>
```

4. Konektor preuzima poruku i kompresuje je kako bi se smanjio njen sadržaj. Nakon zipovanja i enkripcije poruke ona se šalje Adapter aplikaciji na lokalnom serveru legacy sistema.
5. Nakon preuzimanja poruke, Adapter aplikacija radi dekreipciju i dekompresiju poruke koja se zatim upisuje u sledećem obliku u direktorijum za komunikaciju sa AQL serverom.

```
WITH Partneri
  QLIST 5,K
END
```

Fajl ima sledeći naziv REQ<redni broj upita>.AQL

6. Zatim, AQL server preuzima fajl i obrađuje njegov sadržaj. Unutar legacy sistema se pronalaze svi partneri od datog komercijaliste i za dati status. Rezultat obrade se upisuje u fajl pod nazivom RES<redni_broj_upita>. AQL . Fajl sa odgovorom ima sledeći oblik:

```
1<tab>Naziv partnera 1<tab>Ulica 1<tab>Telefon
1<tab>PIB 1
2<tab>Naziv partnera 2<tab>Ulica 2<tab>Telefon
2<tab>PIB 2
...
```

7. Adapter aplikacija uzima sadržaj RES fajla i radi primitivnu obradu njegovog sadržaja stavljajući ga unutar command taga. Tako da fajl nakon toga izgleda:

```
<reposnse>
  <command id="1">
  <record>
  <column>1</column>
  <column>Naziv partnera 1</column>
  <column>Ulica 1</column>
  <column>Telefon 1</column>
  <column>PIB 1</column>
  </record>
  <record>
...
</record>
  </command>
</response>
```

Nakon pretvaranja fajla u XML, radi se njegovo kompresovanje i enkripcija. Odogovor se zatim prosleđuje nazad na web servis.

8. Posle primanja poruke web servis radi dekripciju i dekompresovanje poruke. Da bi se dobili upotrebljivi podaci, *ModulManager* primenjuje izlaznu transformaciju za datu komandu. Postoje razni podržani formati izlaznih poruka (XML, JSON, HTML, ...). U ovom slučaju izlazni format je JSON. Rezultat transformacije izgleda:

```
[
  [ "ID", "Sifra", "Naziv", "Adresa", "Telefon", "PIB" ],
  [
  [ 1, "Naziv partnera 1", "Ulica 1", "Telefon 1", "PIB 1"],
  [ 2, "Naziv partnera 2", "Ulica 2", "Telefon 2", "PIB 2"],
  ...
  ]
]
```

9. Web servis šalje nazad SOAP odgovor koji unutar sebe sadrži JSON poruku za datu komandu.
10. Poruka se putem asinhronog odgovora dostavlja nazad u *browser* klijenta gde se parsira i prikazuje u odgovarajućoj formi.

Bitno je napomenuti da za obavljanje celog ciklusa komunikacije treba u proseku do jedne sekunde za poruke manje veličine.

7. ZAKLJUČAK

Potreba za razmenom podataka poprima ogromne razmere. Danas skoro da nije moguće zamisliti informacioni sistem koji ne ostvaruje barem neki vid komunikacije sa drugim sistemom. Cilj te komunikacije, predstavlja transfer podataka, bilo da se on obavlja u jednom ili oba smera. Izostanak takve komunikacije, nesumljivo bi vodio ka bržem zastarevanju takvog sistema i njegovog stavljanja van upotrebe.

Interoperabilnost koja je proistekla iz integracije legacy sistema i web servisa omogućuje uvođenje funkcionalnosti koje su nastale kasnije tokom razvoja informaciono-komunikacionih tehnologija, čime se drastično produžuje životni vek legacy sistema.

I pored intenzivnog i dinamičnog razvoja softverskih rešenja, web servisi će i dalje imati svoje mesto u budućnosti, kao most između nezavisnih softverskih rešenja i dostizanju onog stepena interoperabilnosti koja takva rešenja iziskuju.

8. LITERATURA

[1] How Do Professionals Perceive Legacy Systems and Software Modernization? ,Ravi Khadka, Belfrit V. Batlajery, Amir M. Saeidi, Slinger Jansen, Jurriaan Hage, 2010.

[2] Interoperability–http://www.microsoft.com/en-us/interop

[3] Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.(iftikahr)

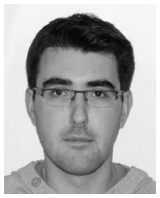
[4] ISO/IEC JTC 1, American National Standards Institute: ISO/IEC JTC 1 N7409 – http://std.dkuug.dk/jtc1/sc2/WG2/docs/n2733.pdf, 2004.

[5] Strategic Approach to Modernize Your Legacy Systems and Wreck the Business Bottlenecks, ZSL inc

[6] National Association of State Chief Information Officers (NASCIO) : Digital States at Risk!: Modernizing Legacy Systems, Decembar 2008

[7] W3C: „Web Services Glossary“ – http://www.w3.org/TR/ws-gloss, 2004.

[8] Santiago Comella-Dorda, Grace A. Lewis, Pat Place, Dan Plakosh, Robert C. Seacord : Incremental Modernization of Legacy Systems, Jul 2001



Nikola Lončar – M.Sc. in Information Systems Wings Software
Kontakt: nikolanbg@gmail.com
Oblast interesovanja: Web programiranje, korisnički interfejs, UX, web servisi, mobilne aplikacije, interoperabilnost

Aleksandar Ilić – M.Sc. in Information Systems Performance Technologies SRB
Kontakt: aleksandar@ilic.rs
Oblast interesovanja: razvoj web, mobilnih i desktop aplikacija, web servisi, interoperabilnost

Slađan Babarogić – docent, Fakultet organizacionih nauka Univerziteta u Beogradu
Kontakt: babarogic.sladjan@fon.bg.ac.rs
Oblast interesovanja: Razvoj IS vođen modelima, Poslovna analiza i Modelovanje poslovnih procesa, Metodologije razvoja informacionih sistema i Baze podataka