UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Klemen Kobetič

Integracija odprtokodne rešitve za podporo komunikacije v podjetju

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: viš. pred. dr. Borut Batagelj

Ljubljana, 2011

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00130/2011
Datum: 02.09.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **KLEMEN KOBETIČ**

Naslov: **INTEGRACIJA ODPRTOKODNE REŠITVE ZA PODPORO KOMUNIKACIJE V PODJETJU**

INTEGRATION OF OPEN SOURCE SOLUTION TO SUPPORT ENTERPRISE COMMUNICATION

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Kandidat naj v okviru svoje diplomske naloge natančno preuči obstoječ sistem za komunikacijo, ki se uporablja v podjetju. Razišče naj programske rešitve, ki bi odpravile problem lokalnosti shranjevanja kontaktov. Prednost naj da brezplačnim odprtokodnim rešitvam. V okviru svojega dela naj izbere najprimernejšo rešitev in jo implementira v obstoječ sistem. Po uspešni integraciji naj svojo rešitev tudi ustrezno preizkusi v praksi.

Mentor:

viš. pred. dr. Borut Batagelj

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a Klemen Kobetič,

z vpisno številko 63070285,

sem avtor/-ica diplomskega dela z naslovom:

Integracija odprtokodne rešitve za podporo komunikacije v podjetju

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom
  viš. pred. dr. Boruta Batagelja
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne xx.xx.2011 Podpis avtorja/-ice:

Zahvala

Zahvaljujem se staršem za vso podporo in potrpežljivost, ki sta mi jo nudila v času mojega študija. Hvala tudi mentorju, viš. pred. dr. Borutu Batagelju za pomoč pri uspešni izvedbi diplomskega dela. Hvala tudi punci Sabini, bratu Dejanu in prijateljem.

Vsa čast in slava tudi ostalim, ki so zaslužni, da sem študij dokončal pred 30. letom, pa tukaj niso napisani!

Najlepša hvala vsem!

Pokojnemu bratu Luki v spomin.

Kazalo

Seznam uporabljenih kratic in simbolov

SIP	(angl.)»Session Initiation Protocol«, protokol za nadzor komunikacijskih sej (zvok, slika)
XML	(angl.)»Extensible Markup Language«, oblika zapisa podatkov
SyncML	(angl.)»Synchronization Markup Language«, odprtokodna platforma za sinhronizacijo podatkov
ActiveSync	Microsoftova platforma za sinhronizacijo podatkov
PIM	(angl.)»Personal information manager«, osebni organizator
IM	(angl.)»Instant Messaging«, neposredno sporočanje
LDAP	(angl.)»Lightweight Directory Access Protocol«, protokol za uporabo imeniških storitev
CentOS	(angl.)»Community ENTerprise Operating System«, odprtokoden operacijski sistem
XMPP	(angl.)»Extensible Messaging and Presence Protocol«, zbirka protokolov namenjena takojšnjemu sporočanju
XCAP	(angl.)»XML Configuration Access Protocol«, protokol za upravljanje z dokumenti XML
OTA	(angl.)»Over The Air«, zagotavljanje storitev po zraku

Povzetek

Diplomsko delo rešuje problem namiznega odjemalca za komuniciranje, ki kontakte shranjuje le lokalno. Včasih še sprejemljiva omejitev je danes postala nepredstavljiva. Kajti zaradi globalne povezljivosti se računalništvo razvija v tej smeri, da morajo biti podatki dostopni od kjerkoli. Za cilj diplomske naloge smo si zastavili sinhronizacijo kontaktov v mobilne telefone. Iz že obstoječega elementa, aplikacije za komunikacijo, smo morali kontakte prenesti v mobilne telefone.
Preden smo se lotili izdelave rešitve smo morali dobro poznati problem. Začeli smo s teorijo sinhroniziranja kontaktov. Analizirali smo obstoječe stanje sistema. Podrobno smo analizirali vsak obstoječ element posebej. V okviru zmožnosti smo zastavili zahteve po nadgradnji. Na podlagi analize in raziskave obstoječih rešitev smo izbrali nove, dodatne elemente, s katerimi bomo nadgradili sistem. V naslednjih korakih smo te elemente namestili. Veliko časa in dela so nam vzeli podproblemi, ki so nastajali pri nameščanju in nastavitvi naših elementov. Poleg tega smo morali napisati lastno skripto za uvoz podatkov.
Dobili smo nadgradnjo sistema kot smo jo želeli. Nove elemente smo uspešno integrirali v obstoječi sistem in jih obenem dobro dokumentirali. Skozi raziskovanje smo odkrili mnogo možnosti za dodatno nadgradnjo. V veliko pomoč nam bo že odkrito znanje.

Ključne besede:

sinhronizacija, kontakt, spletna komunikacija

Abstract

This bachelor's thesis solves the problem of desktop communication client which only stores contacts locally. A still acceptable limitation from the past is unthinkable today. Global connectivity makes computer science develop so data is accessible from anywhere. We set the goal of the thesis to synchronize contacts in mobile phones. From an already existing element, application for communication, we had to transfer contacts to mobile phones.
Before we started with solution development, we had to get good knowledge about the problem. We started with the theory of contact synchronization. We analysed the current state of the system. We analysed every existing element in detail. We set the upgrade requirements according to our possibilities. Based on out analyses and the research of existing solutions we chose new, additional elements, that we apply in the system upgrade. In next steps we installed these elements. Subproblems, which occurred during the installation and element set up, took a lot of our time and caused additional work. In addition, we had to write our own script to import data.
We got the desired system upgrade. We successfully integrated new elements into the existing system and documented them during the process. Throughout the research we found many opportunities for additional upgrades. Already discovered knowledge will be a great help to us in the future.

Key words:

synchronization, contact, internet communication

Poglavje 1

Uvod

Živimo v informacijski dobi, ki nam je postregla z neverjetnim napredkom v zelo kratkem času. Kvaliteta informacij in dostopnost le-teh je spremenila način življenja. Pred približno 30 leti se je začel razvijati internet, kot ga poznamo danes. V samo 30 letih je spremenil način življenja, da podobnosti skorajda ne najdemo več. Internet kot globalna mreža informacij nam je prinesel nešteto ugodnosti in koristi.

Tako ima danes vsakdo račun (vsaj) na enem izmed naslednjih portalov: gmail, yahoo, hotmail, siol, arnes, itd... Vsakdo uporablja različne funkcije, ki jih ponuja vsak od ponudnikov storitev. Čeprav nabor funkcij, ki jih ponuja posamezni ponudnik, ni povsod isti, je eden izmed elementov povsod enak. Večina funkcij je, tako ali drugače, povezana z neko osebo, uporabnikom. Večina uporabnikov to pozna kot kontakte.

Med kopico naprav, ki si jih danes lasti povprečen uporabnik, pa si želimo sinhronizacije. Nabor kontaktov v našem mobilnem telefonu mora biti enak naboru kontaktov na našem priljubljenem socialnem omrežju, le-ta pa mora biti enak naboru kontaktov na našem priljubljenem odjemalcu za e-pošto. V našem primeru si želimo, da se kontakti iz namizne aplikacije za komuniciranje sinhronizirajo z mobilnim telefonom.

Pri vseh teh zahtevah se srečamo z dejstvom, da morajo biti podatki globalno dostopni. Ta problem rešujemo z uporabo storitev v oblaku. Računalništvo v oblaku je razmeroma nova oblika izrabe računalniških virov. Razmeroma zato, ker nekatere oblike že poznamo iz preteklosti. Primer računalništva v oblaku je e-poštni predal, kjer isti strežnik uporablja več uporabnikov. Se pa računalništvo v oblaku širi tudi na druga področja.

Ena izmed prednosti računalništva v oblaku je nižja cena. Cilj vsakega uporabnika je nižja cena za isto ali boljšo storitev. Računalništvo se danes

razvija v to smer, da večina uporabnikov slabo izkorišča sistemska sredstva. Tako koncept računalništva v oblaku ponuja storitve in ne (fizične) produkte.

Računalništvo v oblaku omogoča dostop do podatkov kjerkoli smo. Tako nismo omejeni le na eno lokacijo, kar je danes nujno potrebno. Ponuja pa še kopico drugih prednosti v obliki večje varnosti, zanesljivosti, zmogljivosti, vzdrževanja, itd.

Največ storitev računalništva v oblaku se danes uporablja preko spletnega brskalnika. V praksi to pomeni, da pri ponudniku najamemo storitev. Recimo, manjše podjetje najame storitev računovodskega programa. Ponudnik jim sporoči spletni naslov in uporabniško ime s pripadajočim geslom. V podjetju tako ne potrebujejo zmogljivih strežnikov in delovnih postaj. Dovolj je povprečen osebni računalnik z internetno povezavo in spletnim brskalnikom. Uslužbenec, ki bo uporabljal to storitev, se poveže na splet, odpre spletno stran in že lahko uporablja storitev. Podjetje plačuje ponudniku denarno nadomestilo za uporabo, ta pa je nižja, kot če bi si postavili in vzdrževali svoj sistem. Ponudnik pa zasluži več, saj optimalno izrablja svoje zmogljivosti. Amazon, Google, Microsoft so večja podjetja, ki že ponujajo svetovno dostopne storitve v oblaku.

V prvem delu diplomske naloge bomo predstavili problem sinhronizacije kontaktov, ki se zdi na prvi pogled prav lahek. V nadaljevanju je opisan način shranjevanja kontaktov, ki ga ima uporabljena aplikacija za komuniciranje. Na podlagi analize si bomo zastavili želene funkcionalnosti in omejitve, raziskali različne rešitve in se odločili za najboljšo.
V drugem delu so prikazani implementacija, problemi in rešitve, s katerimi se bomo srečali tekom implementacije. Za preverbo uporabnosti rešitve bomo izvedli nekaj preizkusov.
Na koncu bomo ocenili uporabnost rešitve in morebitno praktično uporabo v komercialne namene.

Poglavje 2

Teorija sinhroniziranja kontaktov

Sinhronizacijo večina ljudi pravzaprav že uporablja, vendar ne nujno v računalniši obliki. Potreba po sinhronizaciji torej obstaja. Pri sinhronizaciji pa prihaja do problemov. Nekateri s strani uporabnika, drugi s strani razvijalca aplikacije. Možnih rešitev je več, odvisno od naših zmožnosti in potreb.

2.1 Potreba po sinhronizaciji

Pri današnjem tempu življenja je čas denar. Tako nam tempo življenja narekuje, da naš čas čim bolj optimalno izkoristimo. V tej diplomski nalogi se bomo osredotočili na kontakte. Marsikateri bralec ne bo razumel, o čem govorimo, a bo ob nadaljnjem branju spoznal, da gre za storitev, ki jo pravzaprav že uporablja. Naj naštejemo nekaj primerov kontaktov, kar namreč razumemo pod tem pojmom:

- zapis oseb in njihovih telefonskih številk doma na listu papirja, ob stacionarnem telefonu,
- uporaba vizitk, poslovnih ali zasebnih,
- shranjevanje oseb v priljubljenem e-poštnem odjemalcu,
- shranjevanje oseb v telefon,
- zapis oseb v priročnem rokovniku.

Marsikdo uporablja vsaj eno, če ne več oblik zapisa kontaktov. Pri vsem tem pa prihaja do bolj ali manj pomembnih problemov. Veliko ljudi je takih, da živijo po načelu »ne popravljaj, kar ni pokvarjeno«. Tako se niti ne zavedajo

potrebe po sinhronizaciji. Tipičen primer je uporabnik mobilnega telefona, ki ob okvari/izgubi/kraji izgubi vse kontakte v imeniku telefona. Če bi imel kontakte sinhronizirane s še kakšno drugo napravo (program na računalniku, e-poštni odjemalec, itd...), bi brez problema povrnil vse kontakte.

Potreba po sinhronizaciji kontaktov z več napravami torej obstaja. To se kaže v tem, da večina ponudnikov storitev omogoča sinhronizacijo z različnimi napravami in celo z drugimi ponudniki.

2.2 Problem sinhronizacije

2.2.1 S strani uporabnika

Po načelu »linija najmanjšega odpora« je treba uporabnika najprej prepričati, da to storitev (sinhronizacijo s še eno napravo) potrebuje in naj jo začne uporabljati. Namreč, kar se povprečnega uporabnika tiče, storitev deluje brez problemov. Še tako dobra rešitev, če je uporabnik ne uporablja, ni uporabna.

Uporabnik, ki ga prepričamo oz. sam zahteva to storitev, pa v veliko primerih naleti na kompleksnost nastavitve sinhronizacije. Pri tej trditvi moramo upoštevati, da tipičen uporabnik nima naprednejšega računalniškega znanja.

2.2.2 S strani načrtovalca aplikacije

Če si pogledamo tehnično plat sinhronizacije kontaktov z različnih virov, lahko bolje razumemo problem. Najprej naštejmo nekaj trenutno bolj aktualnih »ponudnikov« storitev, kjer se uporabnik tako ali drugače sreča s kontakti:

- Socialna omrežja
  - Facebook
  - Twitter
  - MySpace
  - Google+
- E-poštni ponudniki
  - Gmail
  - Yahoo
  - Siol
- Spletne klepetalnice
  - Facebook chat
  - IRC
  - Google talk

Problem ni v velikem številu ponudnikov. Problem nastane pri različnih oblikah zapisa kontaktov. Nekateri ponudniki gredo celo tako daleč, da lahko do njihove storitve dostopamo le preko njihovega vmesnika.

Kako naj torej sinhroniziramo kontakte z mobilnega telefona, e-poštnega odjemalca, priljubljenega spletnega omrežja, ob vsem tem pa vse skupaj nare-

dimo preprosto za še tako neukega uporabnika? Prikazali bomo primer tipičnega problema.
Ponudnik 1:

```
<entry>
        <name>Janez</name>
        <lastname>Novak</lastname>
        <fullname>Janez Novak</fullname>
</entry
```

Ponudnik 2:

```
<entry>
        <first_name>Janez</first_name>
        <surname>Novak</surname>
        <display_name>Janez Novak</display_name>
</entry
```

Že pri teh dveh primerih vidimo, da je oblika zapisa sicer enaka (XML (angl. eXtensible Markup Language) v obeh primerih), vendar enolična preslikava ni mogoča. Tako moramo povezati *name* s *first_ name*, *lastname* s *surname* in *fullname* z *display_ name*. Problemi pa so običajno še večji od predstavljenega, ki je res osnoven.

2.3 Reševanje problema sinhronizacije

Kaj mora torej povprečen uporabnik narediti, da bo imel kontakte sinhronizirane s še kakšno napravo? Odgovor pravzaprav ni preprost, univerzalne rešitve ni. Če se spet vrnemo k mobilnemu telefonu, je skoraj nemogoče sinhronizirati dva telefona različnih proizvajalcev. Po navadi proizvajalec ponuja namensko aplikacijo za sinhronizacijo s telefonom.

Problem sinhronizacije pa očitno tare tudi druge, saj je bilo v ta namen razvitih že kar nekaj rešitev.

Ena izmed rešitev predlaga, da vse naprave podpirajo določen protokol, preko katerega bi lahko komunicirale standardizirano. Tako ni več važno, v kakšni obliki so kontakti shranjeni lokalno na napravi. Trenutno aktualna sta zaprtokodni protokol ActiveSync (protokol za sinhronizacijo podatkov, ki ga je razvilo podjetje Microsoft) in pa odprtokodni protokol SyncML (odprtokodni protokol za sinhronizacijo podatkov).

2.3.1 ActiveSync protokol

Protokol ActiveSync je razvilo podjetje Microsoft in izdalo leta 1996. Omenjamo ga zato, ker je najpomembnejši predstavnik zaprtokodne rešitve. Za uporabo je treba plačati in ga ni možno popravljati. Glede na to, da ga je razvilo podjetje Microsoft, je močno integriran z njihovimi izdelki in storitvami, uporabljajo pa ga tudi drugi ponudniki storitev. Med pomembnejšimi naj naštejemo strežnik Microsoft Exchange (strežnik za sinhronizacijo kontaktov,e-pošte,koledarskih vpisov in opravil) in mobilne telefone z mobilnim operacijskim sistemom Windows Mobile (operacijski sistem podjetja Microsoft za mobilne telefone).

Podpira sinhronizacijo kontaktov, e-pošte in koledarskih vpisov. Razširjeni ActiveSync protokol, predvsem v navezi z Microsoft Exchange, pa podpira tudi sinhronizacijo priljubljenih spletnih strani, datotek in opravil. Možnost sinhronizacije je odvisna predvsem od naprav, ki jih želimo sinhronizirati. S prihodom operacijskega sistema Windows Vista (operacijski sistem podjetja Microsoft za namizne računalnike) je odjemalec oz. program za sinhronizacijo že vključen v sam operacijski sistem.

2.3.2 SyncML protokol

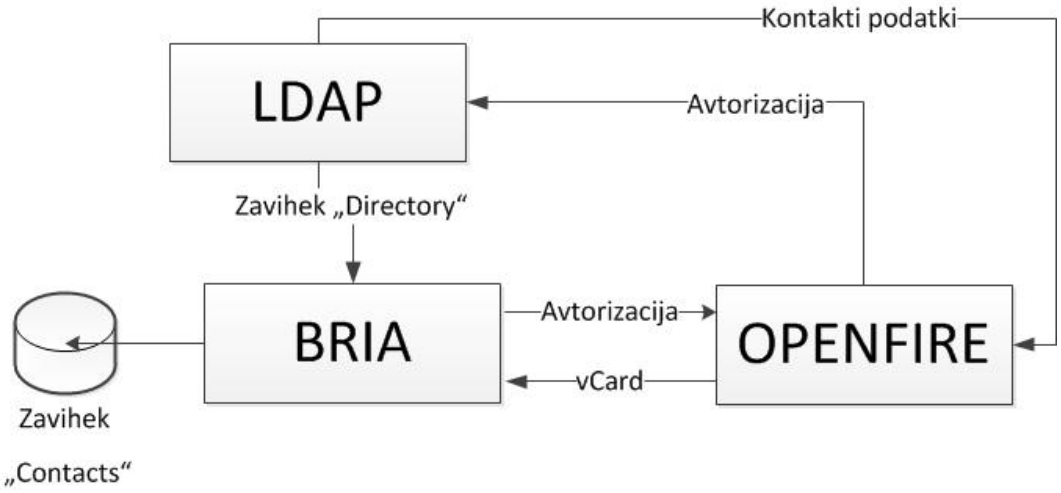
Protokol SyncML je razvilo združenje »Open Mobile Alliance« leta 2000. Je standardiziran, odprtokoden in brezplačen odgovor na protokol ActiveSync. Vsakdo, ki potrebuje sinhronizacijo preko protokola SyncML, si lahko pogleda javno objavljene specifikacije in sprogramira svoj odjemalec/strežnik. Držanje istih navodil omogoča kompatibilnost in univerzalnost različnih storitev in naprav.

Protokol SyncML je v začetku podpiral sinhronizacijo kontaktov, koledarskih vpisih in e-pošto, nabor pa se razširja z novejšimi izdajami standarda.

Poglavje 3

Analiza trenutnega stanja

Opisali bomo trenutno nameščeno konfiguracijo [Slika ??], ki jo bomo v sklopu te diplomske naloge nadgradili. Centralna aplikacija Bria je odjemalec za komunikacijo. Bria je povezana z več elementi. Preko strežnika LDAP prejme podatke o celotnem podjetju. Preko strežnika openFIRE prejme kontakte, se avtorizira in sporoči stanje uporabnika. Odjemalec Bria svoje kontakte shranjuje lokalno na računalniku.



Slika 3.1: Trenutna konfiguracija

3.1 Namizna aplikacija za komunikacijo

Klasična PSTN (angl. Public switched telephone network) telefonija je počasi v zatonu. Nadomešča jo spletna IP telefonija. Ob predpostavki, da imamo spletni dostop, potrebujemo le programski del (aplikacija) in strojni del (mikrofon in spletna kamera).

Med svetovno znane in razširjene aplikacije spada aplikacija Skype. Problem v aplikaciji Skype je, da je brezplačna različica primerna večinoma za domačo uporabo. Tako odpadejo poslovni uporabniki, ki bi to storitev uporabljali le, če bi bila brezplačna. Še več, Skype se prodaja kot »storitev« in ne kot »produkt«. Zaradi varovanja in nadzora nad podatki bi si marsikdo želel imeti svoj Skype strežnik, povezan na lastno Skype bazo. To pa pri aplikaciji Skype ni mogoče.

Eden izmed alternativnih odjemalcev za komuniciranje je tudi Bria (Namizni odjemalec za komunikacijo) [Slika **??**]. Omogoča spletno komunikacijo, določanje statusov, neposredno sporočanje (IM) in celo prenos datotek.

Bria omogoča nadzor nad kontakti, zato je bolj priljubljena med poslovnimi uporabniki. Marsikdo ne zaupa storitvam gMail, Skype, Facebook in ostalim, da varno shranjujejo kontakte, zato želijo imeti nadzor nad lastnimi podatki.

Za nas sta pomembnejša dela odjemalca Bria zavihka »Contacts« in »Directory«.
V zavihku »Directory« dobi vsakdo od zaposlenih možnost brskanja po celotnem imeniku podjetja. Zavihek »Directory« črpa podatke iz strežnika LDAP (angl. Lightweight Directory Access Protocol). V zavihek »Contacts« pa vsak uporabnik dobi kontakte svojega oddelka. Zavihek »Contacts« črpa svoje podatke iz strežnika XMPP (angl. eXtensible Messaging and Presence Protocol).

Telekomunikacija deluje po SIP protokolu, neposredno sporočanje pa preko XMPP protokola. Za komunikacijo po protokolu SIP (angl. Session Initiation Protocol) skrbi telefonska centrala, za komunikacijo preko protokola XMPP pa openFire (strežnik za komunikacijo preko protokola XMPP).

3.2 Strežnik LDAP

Celoten koncept rešitve je namenjen za poslovno okolje. V večini primerov je nabor kontaktov kar celoten imenik podjetja. »Directory« zavihek podpira uvoz kontaktov iz strežnika LDAP. V »Contacts« zavihek pa si prenesemo samo priljubljene kontakte. Redko kdo bi si namreč želel vedno brskati po 1000 kontaktih ali več.

Slika 3.2: Bria, grafični vmesnik

Protokol LDAP skrbi za urejanje podatkovne zbirke preko TCP/IP omrežja. Njegovo delovanje je standardizirano, zato je uporaben kot univerzalna storitev. V največ primerih se uporablja za izdelavo elektronskega imenika oseb. Tudi v našem primeru uporabljamo strežnik LDAP kot elektronski imenik. V bazi strežnika LDAP imamo shranjen celoten imenik zaposlenih v našem podjetju.

3.3 Strežnik XMPP

Protokol XMPP je izšel leta 2000, prvotno pa se je imenoval »jabber«. To je protokol, ki temelji na XML komunikaciji. Namenjen je neposrednemu sporočanju oz. IM-u, izmenjavi podatkov o prisotnosti, prilagodimo pa ga lahko za

izmenjavo drugih podatkov. Je odprtokoden, kar pomeni, da lahko vsak naredi svoj strežnik XMPP. Za postavitev strežnika ne potrebujemo dostopa do interneta, postavimo ga lahko na lokalno domeno. Vsak uporabnik ima svoje uporabniško ime in geslo, s katerim se prijavi na strežnik.

3.4 Sistem zagotavljanja nastavitev

Sistem zagotavljanja nastavitev (angl. Bria provisioning system) je sistem, ki skrbi za samodejno konfiguracijo odjemalcev. Tega v diplomski nalogi ne bomo obdelali, saj za naše delo ni bistvenega pomena.

Poglavje 4

Zahtevane nove funkcionalnosti

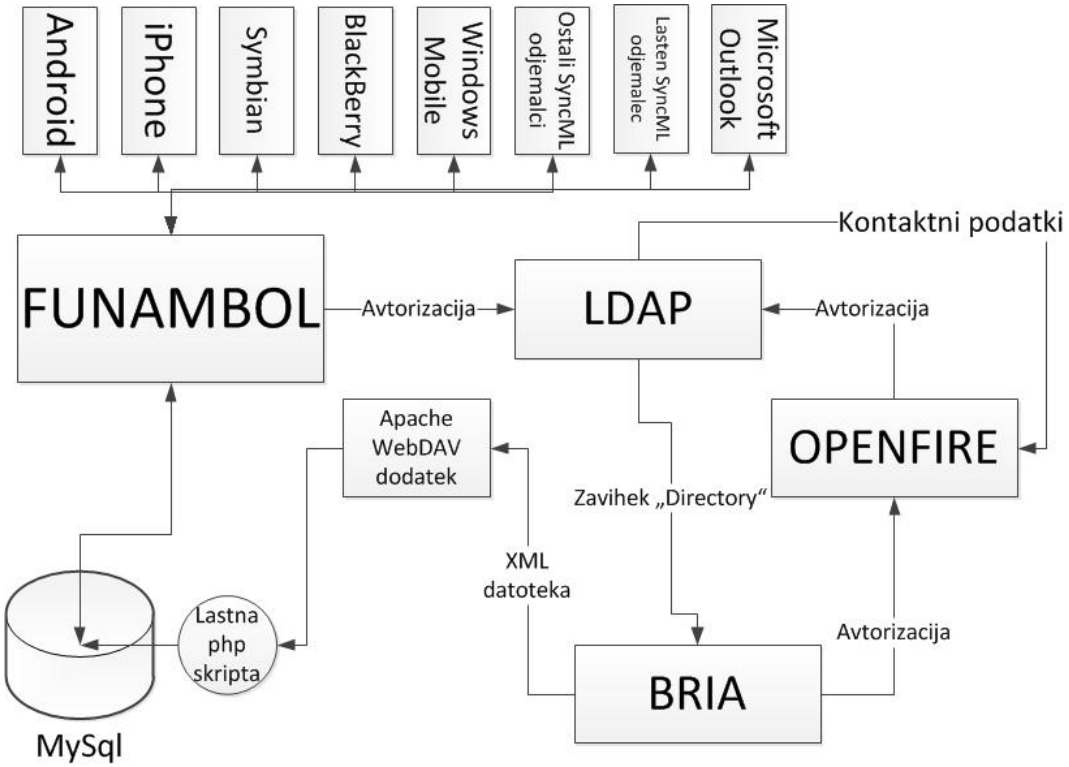
Postavimo si cilje, ki jih želimo implementirati z novo različico sistema [Slika **??**]. Obstoječim elementom bomo dodali strežnik Apache z razširitvijo WebDAV. Skripta php, ki jo bomo napisali sami, da bo skrbela za uvoz v podatkovno bazo. Dodali bomo podatkovno bazo MySQL. Element sistema, ki bo skrbel za sinhronizacijo, pa bo spletni strežnik Funambol, ki bo bral podatke iz baze MySQL in se avtorizial na obstoječ strežnik LDAP.

4.1 Oddaljeno shranjevanje kontaktov

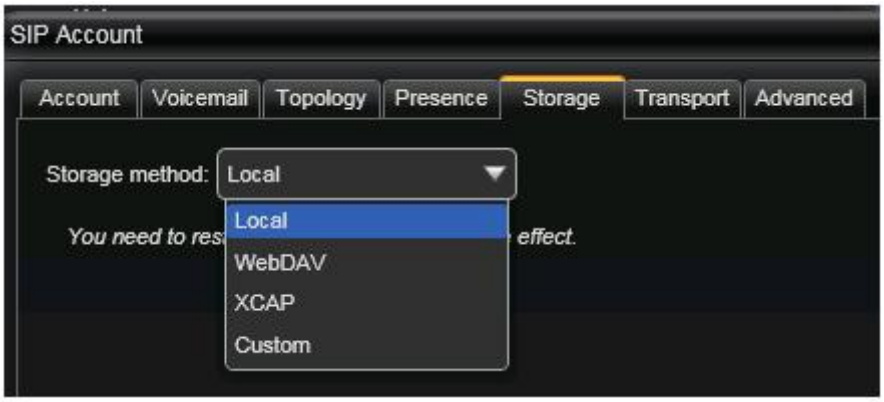
Problem, katerega smo opazili pri uporabi, je ta, da Bria shranjuje kontakte lokalno. To pomeni, da ob vsaki menjavi računalnika, ponovni namestitvi operacijskega sistema ali okvari računalnika, izgubimo vse kontakte. Potrebno bi bilo omogočiti shranjevanje kontaktov na strežnik oz. »oblak«.

Po pregledovanju smo ugotovili [Slika **??**], da Bria podpira oddaljeno shranjevanje podatkov. Izbiramo lahko med razširitvijo WebDAV in protokolom XCAP (angl.XML Configuration Access Protocol).

WebDAV je razširitev spletnega strežnika Apache, ki je v obliki dodatka *mod_ dav* dodana standardni namestitvi strežnika Apache. Omogoča urejanje datotek preko standardnega http(s) protokola. Datoteke se na strežniku shranjujejo v standardni obliki XML.
XCAP je protokol za branje in urejanje datotek XML , ki so na strežniku. Za delovanje potrebujemo strežnik, ki preko protokola XCAP shranjuje datoteke XML v podatkovno bazo.
XCAP je izmed obeh zmogljivejši in ponuja več funkcij. V našem primeru je skromnejši in preprostejši WebDAV čisto dovolj. Odločili smo se za WebDAV



Slika 4.1: Razširitev obstoječega sistema



Slika 4.2: Zavihek *storage,* kjer nastavimo WebDAV

saj omogoča vse, kar potrebujemo, in obenem ne zahteva nameščanja dodatnih zahtevnejših programov.

4.2 Sinhronizacija z mobilnim telefonom

V uvodu smo govorili o potrebi po sinhronizaciji podatkov. Res da je program za komuniciranje priročna rešitev, vendar hočemo še sinhronizacijo teh kontaktov z mobilnim telefonom. Sinhronizacijo mora podpirati čim več telefonov oz. operacijskih sistemov. Sinhronizacija mora biti enosmerna, saj ne želimo privatnih kontaktov iz telefona tudi v službi. Strežnik mora podpirati shranjevanje v podatkovni bazi MySQL.

4.2.1 SyncML - Funambol

Funambol je strežnik za sinhronizacijo imenika, koledarskih vpisov, e-pošte, slik, datotek, videa, itd. Komuniciranje je izvedeno preko protokola SyncML. Brezplačna »Community« različica podpira le kontakte, koledarske vpise in e-pošto. Plačljiva »Carrier« različica pa podpira še slike, video, datoteke, sinhronizacijo s socialnimi omrežji, nadgradnje, garantirano visoko zanesljivost delovanja, itd. Podroben opis razlik si lahko ogledate v tabeli **??**. Za naše potrebe je zaenkrat dovolj brezplačna »Community« različica.

Za delovanje potrebuje nameščeno programsko okolje *java*.

4.2.2 ActiveSync - Z-PUSH

Z-push je odprtokodna rešitev, ki za komunikacijo uporablja protokol ActiveSync. To dejstvo ga naredi manj prilagodljivega od njegovih konkurentov. Za delovanje potrebuje strežnik Apache s podporo za skriptni jezik php. Prednost je, da izdelki podjetja Microsoft že ob namestitvi vsebujejo podporo za delovanje, medtem ko je treba na ostale naprave namestiti dodatni program ali pa podpora celo ni mogoča.

4.2.3 Izbira

Izbrali smo rešitev Funambol, saj ocenjujemo, da ima velik potencial. Res je, da izbiramo med dvema odprtokodnima rešitvama, vendar Z-PUSH za komunikacijo uporablja protokol ActiveSync, ki pa ni odprtokoden. Strežnik Funambol uporablja odprtokoden protokol SyncML. Odprtokodnost vidimo kot prednost, saj nam omogoča lasten razvoj komponent, katerega bomo najbrž potrebovali v prihodnosti.

	Brezplačna različica	Plačljiva različica
Sinhronizacijski strežnik	DA	DA
Generičen SyncML odjemalec	DA	DA
Windows sinhronizacija (PIM)	DA	DA
Windows Mobile sinhronizacija (PIM)	DA	DA
Symbian sinhronizacija (PIM)	DA	DA
BlackBerry sinhronizacija (PIM)	DA	DA
iPhone / iPod sinhronizacija (kontakti)	DA	DA
Android sinhronizacija (PIM)	DA	DA
JAVA ME e-poštni odjemalec	DA	DA
Windows Media sinhronizacija (datoteke, video,fotografije)	NE	DA
BlackBerry Media sinhronizacija (datoteke, video, fotografije)	NE	DA
Android Media sinhronizacija (datoteke, video, fotografije)	NE	DA
iPhone / iPad Media sinhronizacija (video, fotografije)	NE	DA
Razširljivost	NE	DA
Visoka zanesljivost in neobčutljivost na okvare	NE	DA
Spletni portal, SAPI	NE	DA
Uvoz kontaktov iz Gmail / Yahoo!	NE	DA
Uvoz slik Facebook prijateljev	NE	DA
OTA nastavitev naprav	NE	DA
OTA nameščanje aplikacij	NE	DA
SMS storitev	NE	DA
Spletni portal za pomoč uporabnikom	NE	DA
Mesečni popravki odjemalcev	NE	DA
Tehnična pomoč	NE	DA
Izobraževanje	NE	DA
Popravki strežnika	NE	DA

Tabela 4.1: Primerjava plačljive in brezplačne različice Funambol strežnika

Poglavje 5

Namestitev

5.1 Operacijski sistem CentOS

Pri izbiri operacijskega sistema imamo proste roke, želimo si nizke cene brez izgube funkcionalnosti. Večino namiznih računalnikov poganja sistem Microsoft Windows (serija operacijskih sistemov podjetja Microsoft), vseeno pa je v poslovnih okoljih zaželjen Linux (odprtokodni operacijski sistem). Njegova največja prednost je nizka cena.

Za potrebe diplomske naloge smo operacijski sistem namestili (Slika [??]) na virtualen računalnik s pomočjo programske opreme VMware (programsko orodje za virtualizacijo).

Odločili smo se za 64-bitno izdajo CentOS, različice 6, operacijskega sistema. CentOs je brezplačna različica odprtokodnega operacijskega sistema Red Hat Enterprise Linux (ena izmed različic operacijskega sistema Linux), ta pa temelji na Fedora Linux (ena izmed različic operacijskega sistema Linux). Sicer je tudi Red Hat brezplačen, je pa plačljiva njegova podpora, CentOS temelji popolnoma na brezplačništvu. Dodatna prednost so preverjene, kompatibilne in brezplačne posodobitve, ki jih pri operacijskem sistemu Red Hat dobimo proti plačilu.

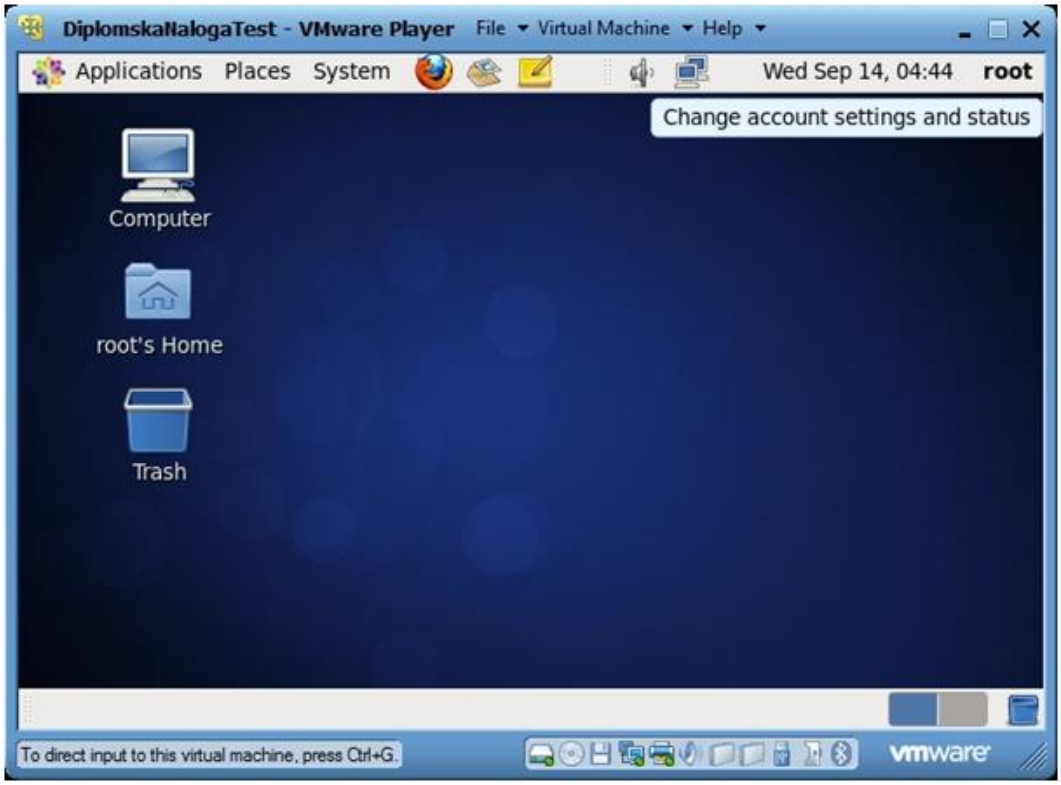
S spletne strani smo z enega izmed javnih zrcalnih strežnikov brezplačno prenesli namestitveno .iso datoteko. V orodju VMware smo kreirali »Nov virtualni računalnik«, *pripeli* .iso namestitveno datoteko in sledili namestitvi.

Namestili smo standardno »Desktop« različico, ki se namesti z grafičnim vmesnikom *GNOME*. Pri namestitvi smo poleg standardnega nabora izbrali še naslednje komponente:

- izvajalno okolje (Java),

- podporo skriptnemu jeziku php (php, php-mysql, php-ldap, php-mbstring),
- podporo programskemu jeziku perl (perl-ldap),
- podporo strežniku MySQL (mysq, mysql-server)
- in oddaljenemu dostopu (vnc, vnc-server).

V nadaljevanju diplomske naloge se predvideva, da smo na strežnik namestili vse zahtevane pakete. Programi, uporabljeni v nadaljevanju, za pravilno delovanje potrebujejo vse pakete in njihove odvisne pakete.



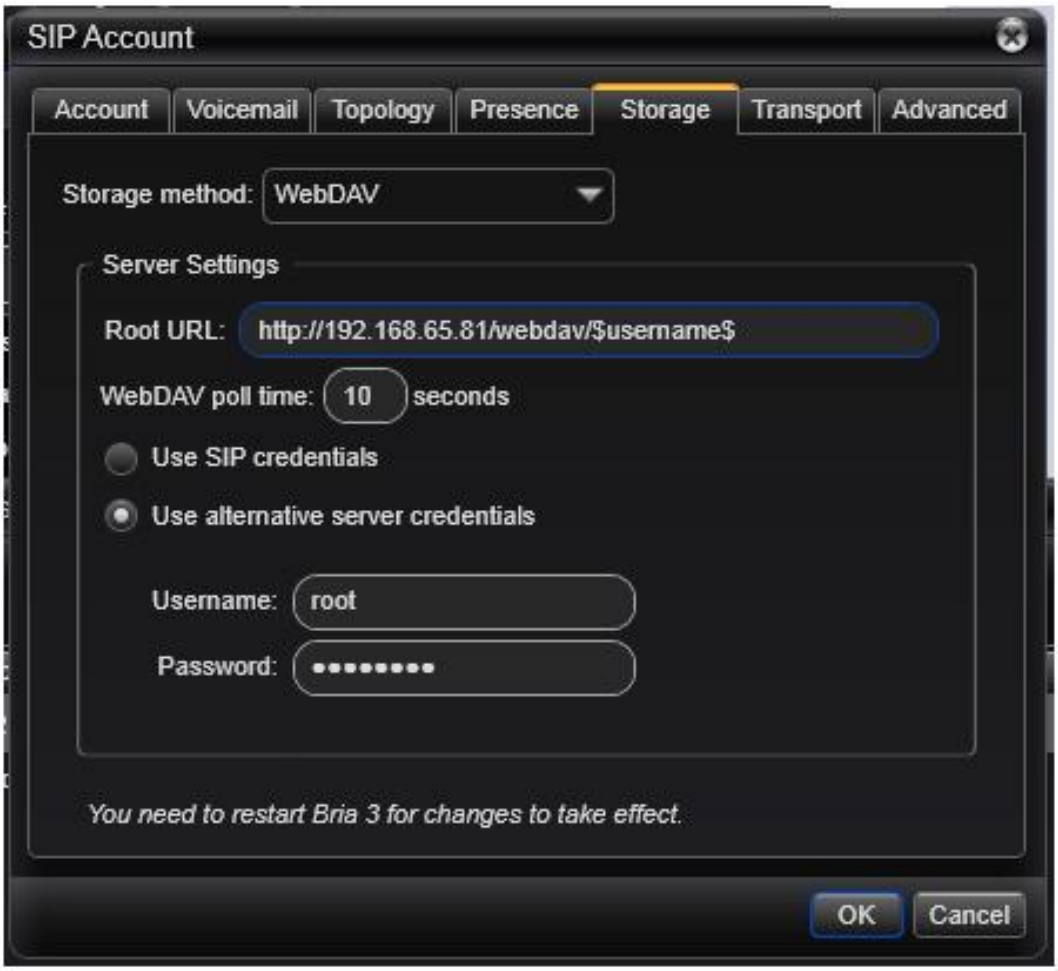
Slika 5.1: CentOS6 nameščen na virtualizacijskem orodju VMware

5.2 Bria – zavihek *Contacts*

V odjemalcu Bria moramo nastaviti shranjevanje kontaktov na WebDAV. Nastavitve so vezane na posamezen SIP račun oz. na vsakega uporabnika posebej. Vsak uporabnik ima namreč svojo mapo na strežniku.

5.2.1 WebDAV - Oddaljeno shranjevanje kontaktov

Nastavitve najdemo na nastavitvah posameznega SIP računa, na zavihku *Storage* (Slika [??]). Naslov povezave na odjemalcu nastavimo na *http://<ip>/webdav/$username$*. Bria namreč zamenja *$username$* z našim uporabniškim imenom (*TestUporabnik1*). *WebDAV poll time* nastavimo na 600 sekund, kar pomeni, da vsakih 600 sekund preverja novejšo različico datoteke s kontakti. Označimo še *Use alternative server credentials* in vpišemo uporabniško ime in geslo, ki ju potrebujemo za dostop do strežnika *WebDAV*.



Slika 5.2: Nastavitve za oddaljeno shranjevanje preko WebDAV.

5.2.2 Nastavitev in namestitev Apache strežnika

Spletni strežnik Apache se namesti že ob standardni namestitvi operacijskega sistema CentOS. Ob zagonu operacijskega sistema je potrebno le zagnati proces *httpd*. Privzeta korenska mapa, dostopna iz interneta, se nahaja v */var/www/html*. Za naše potrebe smo znotraj mape *html* kreirali še mapo *webdav*. Za vsakega uporabnika bomo kreirali še podmapo z njegovim uporabniškim imenom. Tako dobimo končno mapo, kjer želimo, da Bria shranjuje kontakte, npr.*/var/www/html/webdav/TestUporabnik1*. Posebnost tukaj je, da moramo mapi TestUporabnik1 zamenjati dovoljenja uporabnika z Apache. To storimo z ukazom *chown apache TestUporabnik1*.

Poleg kreiranja pravilne strukture moramo poskrbeti še za pravilno delovanje razširitve WebDAV. Sicer je že nameščena, je pa vseeno potrebno nekaj dodatnih ukazov. Tako v mapi */etc/httpd/conf.d/* kreiramo datoteko *webdav.conf*, s katero bomo vnesli nastavitve za WebDAV. Vse datoteke in vse mape s svojimi podmapami v korenski mapi *html* so javno dostopne, vendar še niso dostopne preko protokola WebDAV. To moramo posebej definirati. Tako za mapo TestUporabnik1 dodamo v *webdav.conf* naslednje nastavitve.

```
DavLockDB /tmp/DAVLock
DAVMinTimeout 600

<Directory /var/www/html/webdav/TestUporabnik1 >
    Dav On
    AuthType Basic
    AuthName TestUporabnik1
    AuthUserFile /etc/httpd/conf.d/webdav.passwd
    Options -Indexes
    Require user TestUporabnik1
</Directory>
```

Z drugimi besedami, lahko rečemo, da smo mapi */var/www/html/webdav/TestUporabnik1* dodali podporo za WebDAV in za dostop do mape zahtevamo avtorizacijo z uporabniškim imenom in geslom uporabnika TestUporabnik1.

Uporabniško ime in geslo imamo zapisanega v isti mapi (*/etc/httpd/conf.d/*) in jo poimenujemo *webdav.passwd*. Za zapis uporabniškega imena in gesla uporabimo vgrajeno funkcijo v operacijskem sistemu CentOS. Uporabniško ime dodamo z ukazom `htpasswd -b /etc/httpd/conf.d/webdav.passwd TestUporabnik1 geslo`. Ukaz doda v datoteko *webdav.passwd* naslednjo vsebino:

TestUporabnik1:stEPhfINkUpSA.

Celoten postopek smo poenostavili s skripto, ki je kot priloga priložena diplomski nalogi. Vse, kar moramo narediti, je, da zaženemo ukaz: `dodajUporabnika.sh TestUporabnik1 geslo`.

5.2.3 Analiza datotek v WebDAV imeniku

Ponovno zaženemo Brio in opazujemo, kaj se dogaja v mapi *webdav/TestUporabnik1*. Odpremo mapo in opazimo tri datoteke:

- TestUporabnik1.192.168.196.67.contacts-resource-list.xml
- TestUporabnik1.192.168.196.67.pres-rules
- TestUporabnik1.192.168.196.67.resource-list.xml

Glede na vsebino ugotovimo, da je vsebina zapisana v datoteki s končnico contacts-resource-list.xml. Primer kontakta, ki ima vpisan poln nabor možnih atributov:

```
<entry uri="sip:02795D230C1F4F629A0B6E7A6F3864F6@cpsi.
    invalid">
  <display-name>Janez Novak</display-name>
  <cp:prop name="given_name" value="Janez"/>
  <cp:prop name="surname" value="Novak"/>
  <cp:prop name="sip_address"
  value="sip:0123456789@192.168.196.67"/>
  <cp:prop name="sip_address_comm" value="phone"/>
  <cp:prop name="xmpp_address"
  value="janeznovak@domena.si"/>
  <cp:prop name="xmpp_address_comm" value="im"/>
  <cp:prop name="home_number" value="073322111"/>
  <cp:prop name="home_number_comm" value="phone"/>
  <cp:prop name="business_number" value="041444333"/>
  <cp:prop name="business_number_comm" value="phone"/>
  <cp:prop name="mobile_number" value="041333444" >
  <cp:prop name="mobile_number_comm" value="phone"/>
  <cp:prop name="business_number#2" value="041444555"/>
  <cp:prop name="business_number#2_comm" value="phone"/>
  <cp:prop name="email_address"
  value="janeznovak@domena.si"/>
```

```
  <cp:prop name="category" value="Family"/>
  <cp:prop name="entry_id"
   value="02795D230C1F4F629A0B6E7A6F3864F6"/>
  <cp:prop name="stamp"
   value="61962d3c145d948743e58e01d6a88f4809d4618d"/>
  <cp:prop name="data_hash"
   value="14c04729fe51e58c46748d743137374060724497"/>
  <cp:cprop name="DefaultAddress"
   value="sip:0123456789@192.168.196.67"/>
  <cp:cprop name="DefaultAddressType" value="0"/>
  <cp:cprop name="DefaultCommunicationType" value="1"/>
  <cp:cprop name="IsFavorite" value="false"/>
  <cp:cprop name="IsOnAlertList" value="false"/>
  <cp:cprop name="IsPresenceSubscribed" value="true"/>
  <cp:cprop name="KeepOnAlertList" value="false"/>
</entry>
```

5.3 Bria – zavihek *Directory*

Nastavitve za zavihek *Directory* so vezane na celoten program. Tako smo omejeni na branje kontaktov iz enega strežnika LDAP. Nastavitve najdemo v globalnih nastavitvah, sekcija *Directory* (Slika [??]).

5.3.1 LDAP nastavitve

V odjemalcu nastavimo naslednje parametre:

- Server

  ip številka našega strežnika

- Authentication method

  Simple

- Username (dn=)

  Celoten iskalni niz uporabnika, ki ima dovoljenje brati iz LDAP strežnika

- Credential

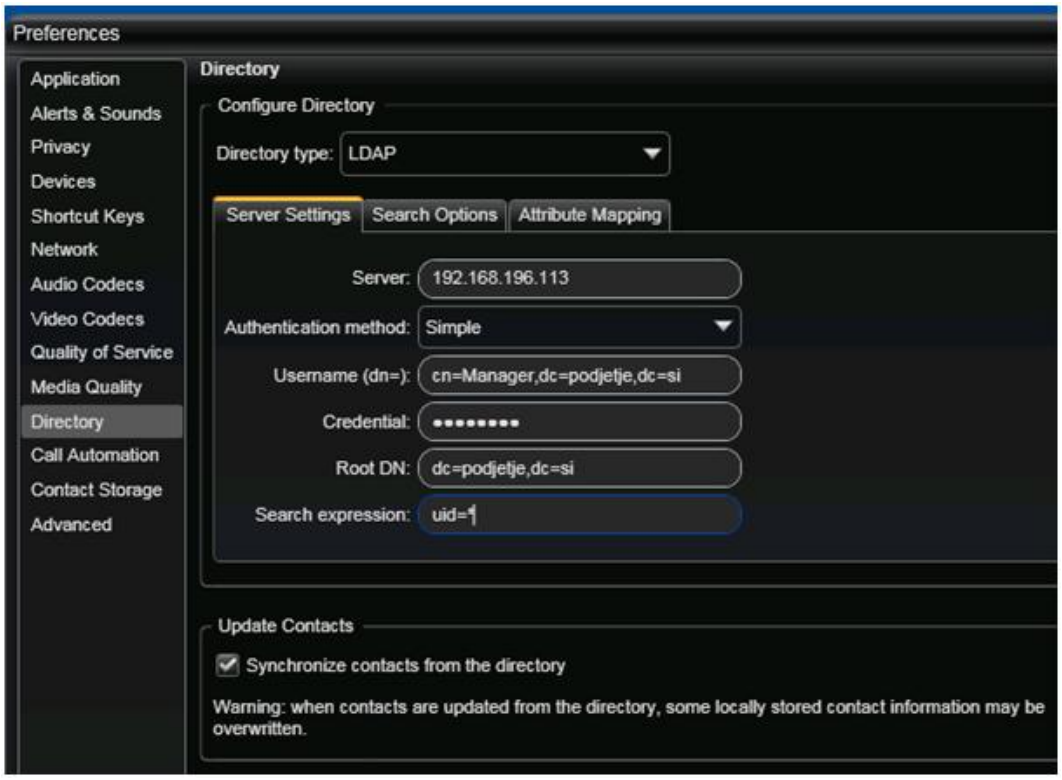
  Geslo tega uporabnika

- Root DN

  Korenski iskalni niz

- Search expression

  Filter, po katerem iščemo uporabnike

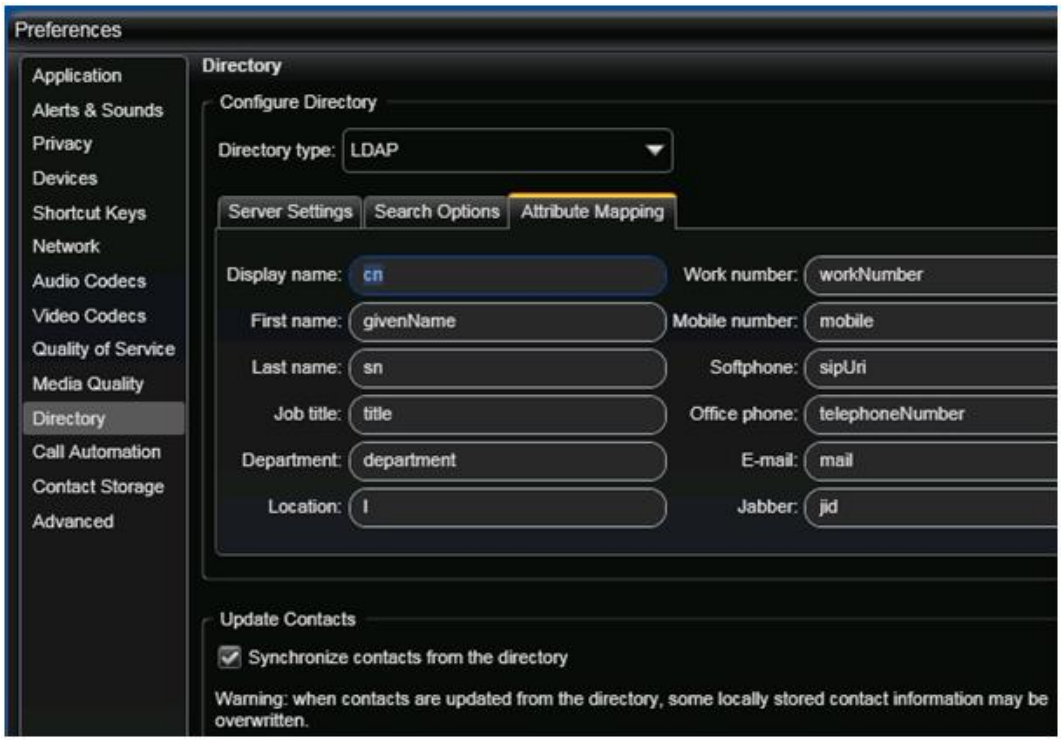


Slika 5.3: Nastavitve za dostop do LDAP kontaktov.

V naslednjem zavihku *Attribute Mapping* (Slika [??]) lahko nastavimo preslikave iz strežnika LDAP v Brio.

5.3.2 Namestitev in nastavitev LDAP strežnika

»Directory« zavihek v odjemalcu Bria imamo povezan s strežnikom LDAP. Za LDAP strežnik izberemo že obstoječo odprtokodno rešitev openLDAP. Tudi tukaj se pokaže, da smo pravilno izbrali operacijski sistem. Za namestitev strežnika openLDAP moramo le izvesti ukaz `yum install openldap`. S tem



Slika 5.4: Nastavitve za preslikave atributov LDAP kontaktov.

namestimo strežnik openLDAP, moramo pa ga še nastaviti. Namesti se v */etc/openldap*, kjer najdemo naslednje pomembnejše datoteke:

- *ldap.conf*

  glavna konfiguracijska datoteka za odjemalca

- *slapd.conf*

  glavna konfiguracijska datoteka za strežnik

- Datoteke v mapi */schema/*

  sheme za kontakte

V podrobnosti namestitve in konfiguracije strežnika openLDAP se ne bomo spuščali. Večina podjetij ima namreč strežnik že postavljen. Za potrebe naše diplomske naloge bomo predvidevali, da obstaja strežnik openLDAP, napolnjen s podatki. Konfiguriran mora biti z naslednjimi nastavitvami:

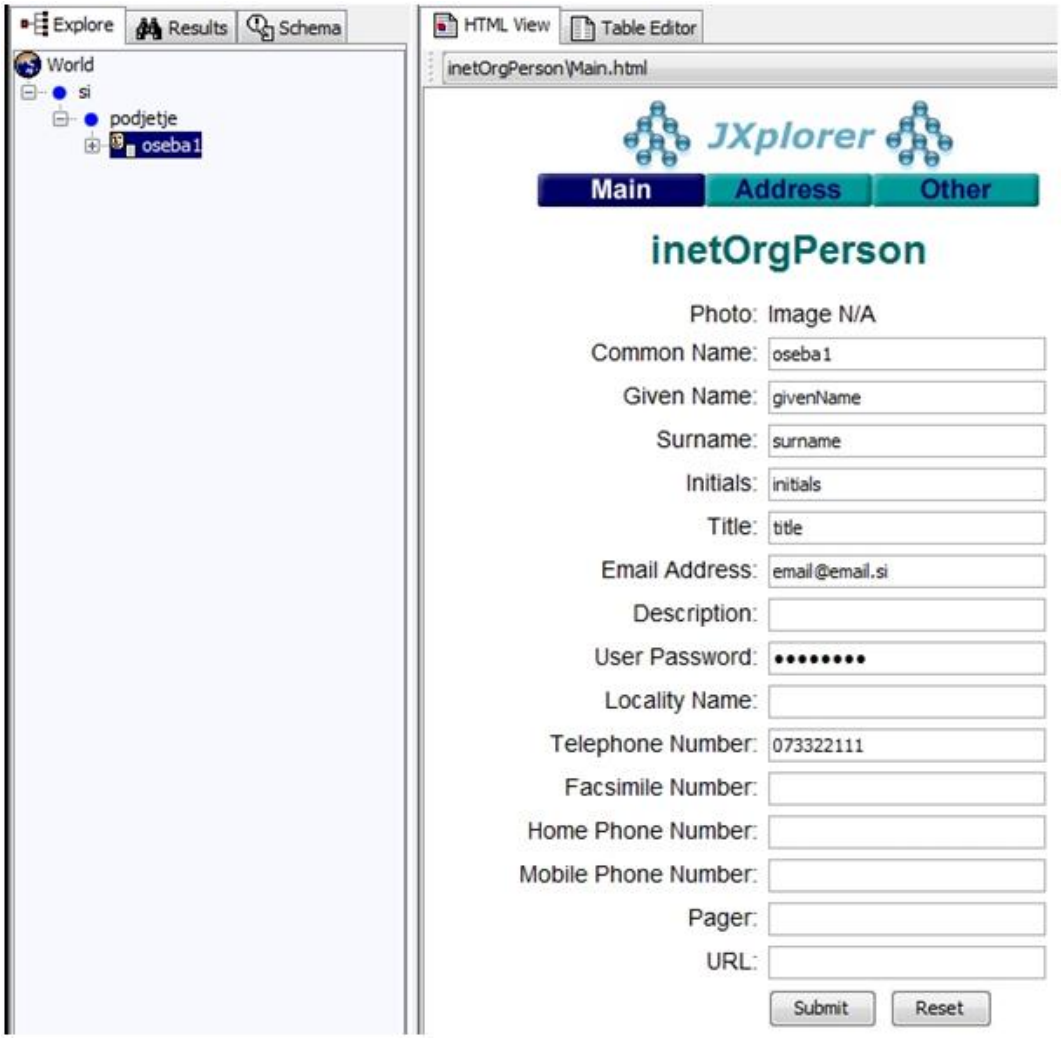
- korenski imenik sestavljen iz podjetja, in končnico,

  dc=podjetje, dc=si

- osebe v obliki sheme *inetOrgPerson* (Slika [??]).

  cn=oseba1, dc=podjetje, dc=si

Za potrebe administracije potrebujemo še korenskega uporabnika in korensko geslo (npr. *cn=Manager,dc=podjetje,dc=si*).



Slika 5.5: Grafični prikaz kontakta v strežniku LDAP v obliki *inetOrgPerson*

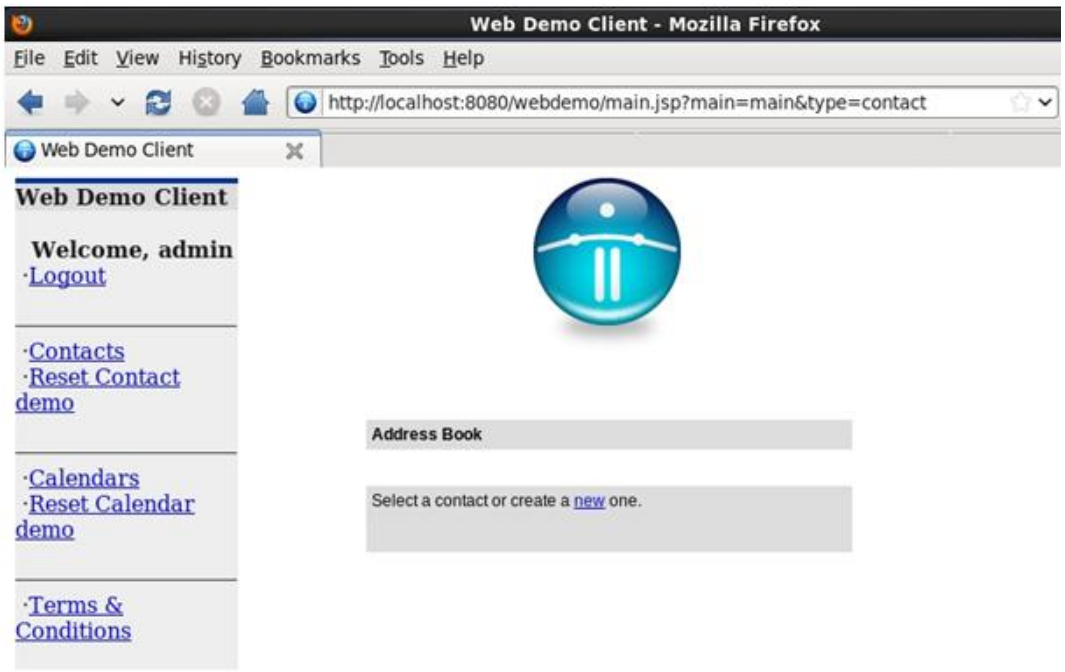
5.4 Namestitev strežnika Funambol

Za sinhronizacijo z mobilnimi telefoni bomo uvedli nov element sistema. Izbrali smo odprtokodni program Funambol.

5.4.1 Strežnik

Z domače spletne strani si prenesemo namestitveno datoteko. Izberemo 64-bitno različico za Linux. Datoteko razpakiramo in poženemo ukaz `./install.sh`. Namestitev osnovnega strežnika Funambol je samodejna. Sprejmemo le pogoje uporabe, uporabimo privzeto namestitveno mapo (*/opt/Funambol/*), na koncu nas vpraša le še, če želimo zagnati strežnik.

Po uspešni namestitvi si lahko na naslovu *http://localhost:8080/webdemo/login.jsp* ogledamo spletni demo vmesnik (Slika [??]), ki nam pove, da je bila namestitev uspešna. Obenem tako preverimo povezavo s podatkovno bazo. Privzeto uporabniško ime je *admin*, geslo pa *sa*.



Slika 5.6: Spletni demo portal kjer lahko preverimo delovanje.

5.4.2 Administracijsko orodje za strežnik

Prav tako si s spletne strani prenesemo stisnjeno datoteko *funambol-admin-10.0.0.tgz*. Osnovna namestitev strežnika ne vsebuje administracijskega orodja, zato ga moramo namestiti posebej. Orodje namestimo tako, da kompresirano datoteko preprosto razširimo v namestitveno mapo. Ker nismo spreminjali privzete mape (*/opt/Funambol/*), razširimo datoteko v privzeto korensko mapo (*/opt/*).

Pred zagonom je potrebno spremeniti uporabo jave v operacijskem sistemu. Izvedemo naslednje ukaze:

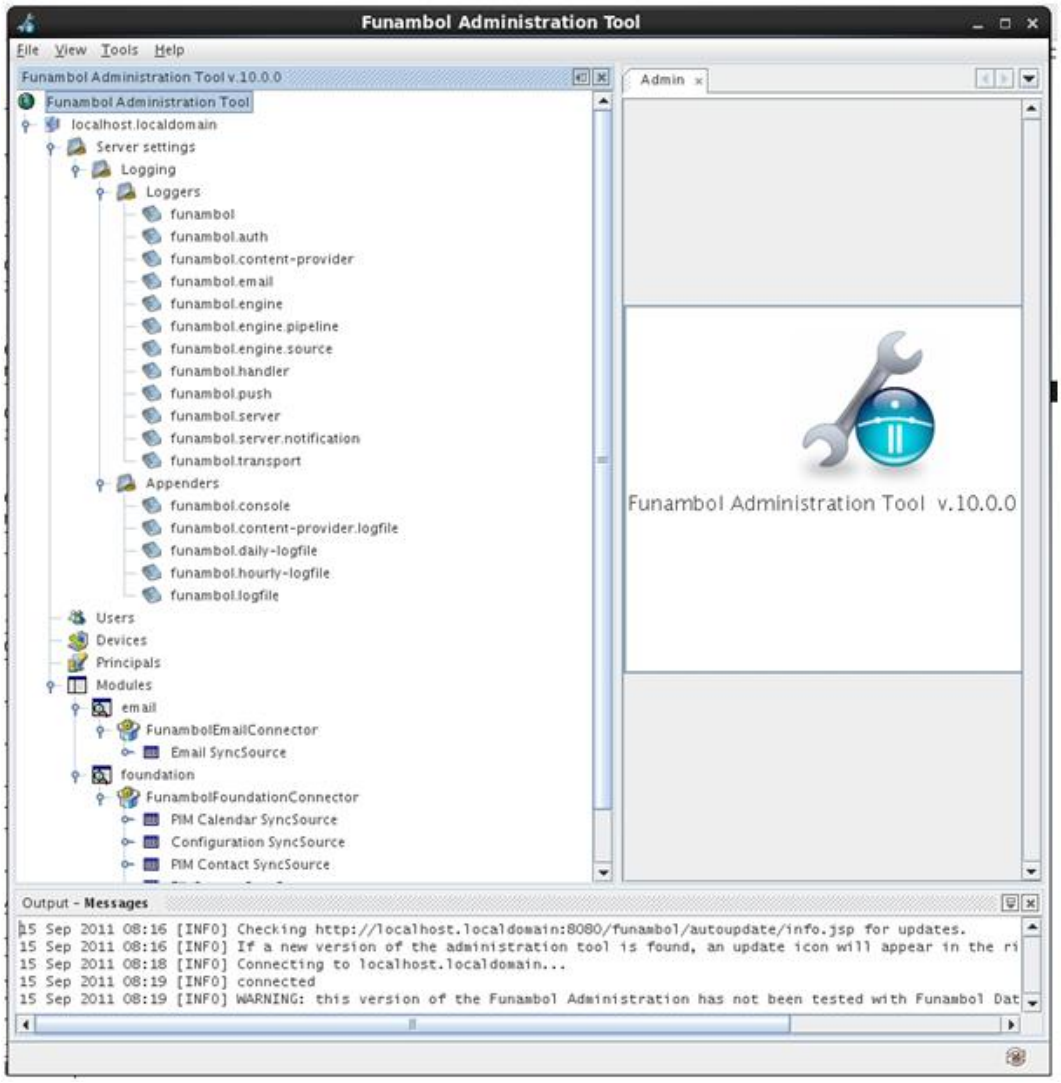
- odstranimo prejšnjo povezavo (`rm /etc/alternatives/java`)
- premaknemo se v mapo (`cd /etc/alternatives`)
- ustvarimo novo povezavo (`ln -s /opt/Funambol/tools/jre-1.6.0/jre/bin/java`)

Namestitev je tako končana. Administracijsko orodje zaženemo z ukazom v ukazni lupini: `./opt/Funambol/admin/bin/funamboladmin start`. Zažene se nam okno (Slika [??]), kjer lahko spreminjamo in nastavljamo nastavitve.

5.4.3 MySQL povezava

Funambol v osnovi podpira podatkovno bazo Hypersonic. Zaradi zahtev moramo uporabiti podatkovno bazo MySQL. Najprej zaženemo proces MySQL z ukazom `/etc/init.d/mysqld start`. Ob prvem zagonu moramo nastaviti geslo za korenskega (*root*) uporabnika. To naredimo z ukazom `/usr/bin/mysqladmin -u root password 'diplomska'`, kjer je geslo *diplomska*. Podatkovno bazo in uporabnika moramo kreirati ročno. Tako izvedemo najprej ukaz `mysql -u root -p`. Sistem nas vpraša za geslo (*diplomska*). Tako smo v mysql orodju. Tukaj lahko kreiramo bazo in določimo privilegije:

- ustvarimo bazo funambol (create database funambol;),
- uporabimo bazo funambol pri nadaljnjih ukazih (use funambol;),
- dodelimo dovoljenja uporabniku funambol (grant all privileges on funambol.* to funambol@localhost identified by 'funambol';),
- osvežimo dovoljenja (flush privileges;),



Slika 5.7: Administracijsko orodje za strežnik Funambol.

- izhod (quit;).

Tako imamo bazo *funambol* in uporabnika *funambol* z geslom *funambol*.

Funambol za komunikacijo z bazo MySQL, uporablja MySQL vmesnik. Z uradne MySQL strani si prenesemo vmesnik Java-MySQL. Iz celotne datoteke potrebujemo le .jar datoteko (v našem primeru *mysql-connector-java-5.1.17-bin.jar*).

Datoteko (.jar) prenesemo v */opt/Funambol/tools/* mapo. Nato v mapi */opt/Funambol/ds-server/* odpremo datoteko *install.properties*. Podatkovno bazo spremenimo v MySQL. Prej: *dbms=hypersonic*, potem: *dbms=mysql*.

Nato z dodajanjem znaka # na začetek vrstice zakomentiramo podatke za povezavo z bazo hypersonic in vnesemo nastavitve za mysql (Tabela [??]).

```
# MySQL
# =====
#
jdbc.classpath=/opt/Funambol/tools/mysql-connector-java-5.1.17-bin.jar
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost/funambol?characterEncoding=UTF-8
jdbc.user=funambol
jdbc.password=funambol
```

Tabela 5.1: Spremembe, ki jih moramo narediti za povezavo z bazo MySQL.

Pred naslednjim korakom preverimo, če imamo zagnan proces `mysqld` (*/etc/init.d/mysqld status*). Spremembe shranimo in zaženemo datoteko `opt/Funambol/bin/install`. S tem ponovno zaženemo tisti del namestitve, ki skrbi za podatkovno bazo. Med samim potekom namestitve odgovorimo z *y* na vsa vprašanja med potekom. Ta se nanašajo na kreiranje podatkovne baze oz. tabel.

Po končanem postopku ponovno zaženemo Funambol in izvedemo ukaze:

- ustavimo Funambol (`/opt/Funambol/bin/funambol stop`),
- zaženemo Funambol (`/opt/Funambol/bin/funambol start`).

Po uspešni namestitvi, se vrnemo na spletni demo portal in se vpišemo z uporabniškim imenom *admin* in geslom *sa*. Tako lahko preverimo, da smo uspešno nastavili Funambol in da uporablja bazo MySQL.

5.4.4 Vmesnik LDAP

Standardna Funambol namestitev ne vsebuje vmesnika LDAP. Vmesnik LDAP, kot ga podpira Funambol, lahko uporabimo za avtorizacijo ali sinhronizacijo kontaktov. Mi želimo podporo za avtorizacijo.

Izvorno kodo vmesnika LDAP si lahko prenesete iz uradne strani, jo vmes po želji popravite, in zgradite. Na koncu dobimo datoteko podobno, *ldap-connector-7.1.0-SNAPSHOT.s4j*. Kopiramo jo v mapo */opt/Funambol/ds-server/modules/*. Nato spet popravimo (Tabela [??]) datoteko *install.properties* (v mapi */opt/Funambol/ds-server/*).

PREJ
modules-to-install=content-provider-10.0.0,email-connector-10.0.0,foundation-10.0.0,phones-support-10.0.0,webdemo-10.0.0
POTEM
modules-to-install=content-provider-10.0.0,email-connector-10.0.0,foundation-10.0.0,phones-support-10.0.0,webdemo-10.0.0,ldap-connector-7.1.0-SNAPSHOT

Tabela 5.2: Funambol LDAP nastavitve v datoteki *install.properties*

Spremenjeno vsebino shranimo in poženemo datoteko *install.modules* (v mapi */opt/Funambol/bin/*). Tudi tukaj med postopkom odgovarjamo z *y* (*yes*).

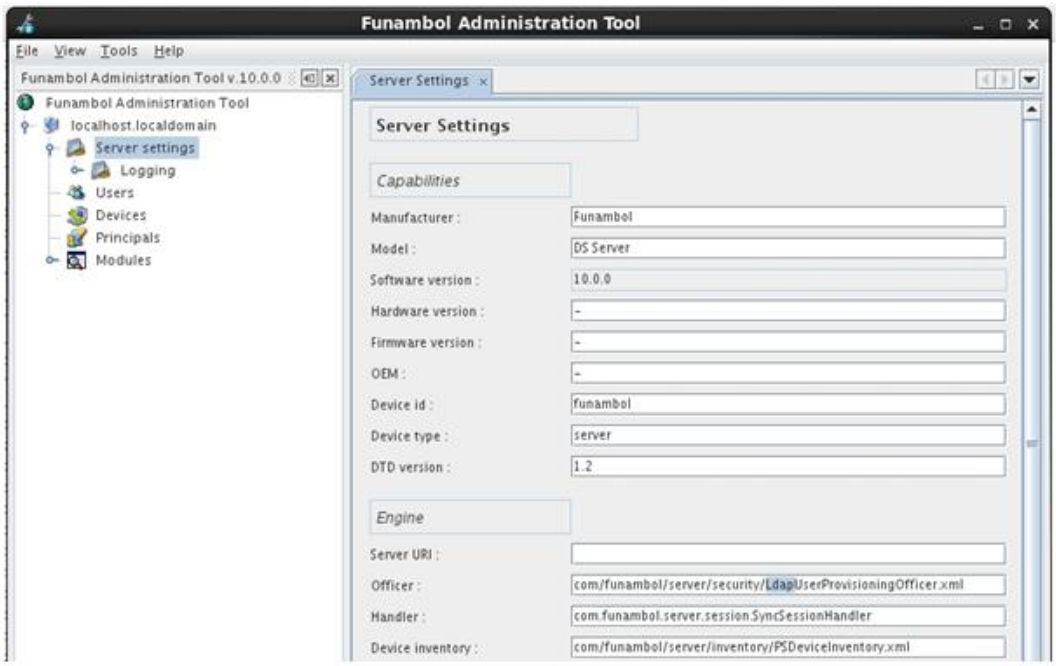
Po uspešno končanem postopku smo vmesnik namestili, ne pa še nastavili. Vse nastavitve se nahajajo v datoteko *LdapUserProvisioningOfficer.xml* (v mapi */opt/Funambol/config/com/funambol/server/security/*). Vsebino spremenimo, tako da ustreza našim nastavitvam.

```
?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.0" class="java.beans.XMLDecoder">
        <object class="com.funambol.LDAP.security.
            LDAPUserProvisioningOfficer">
                <void property="serverAuth">
                        <string>none</string>
                </void>
                <void property="ldapInterfaceClassName">
                    <!-- short class name interface -->
                        <string>openLdap</string>
                </void>
                <void property="baseDn">
                        <string>{dc=podjetje,dc=si}</
                            string>
                </void>
```

```
<void property="userSearch">
        <string>(cn=%s)</string>
</void>
<void property="ldapUrl">
        <string>ldap://192.168.68.128/</
            string>
</void>
<!--
        credential used for first bind
            to ldap. if unset leave empty
-->
<void property="searchBindDn">
        <string>cn=Manager,dc=podjetje,
            dc=si</string>
</void>
<void property="searchBindPassword">
        <string>diplomska</string>
</void>
```

Zadnji korak je, da v administracijskem orodju spremenimo (Slika [??]) nadzornika avtorizacije. V polju *Officer* spremenimo iz *UserProvisioningOfficer.xml* v *LdapUserProvisioningOfficer.xml*.

Po ponovnem zagonu procesa Funambol se lahko z odjemalcem za Funambol sinhroniziramo samo z uporabo uporabniškega imena in gesla, ki smo ga dobili s strežnika LDAP.



Slika 5.8: Nastavitve za LDAP v Funambol administracijskem orodju.

Poglavje 6

Preizkusni scenariji

6.1 Brisanje mape

Brisanje korenske mape (webdav) ali uporabniške mape (npr. j.novak) povzroči nepravilnosti v delovanju. Bria ne zna ponovno ustvariti mape.

6.2 Brisanje XML datoteke

Uporabili smo klienta za Microsoft Outlook (orodje za dostop do e-pošte), odjemalec za Funambol in s tekstovnim urejevalnikom spremljali datoteko XML na strežniku.

6.2.1 Med delovanjem odjemalca

XML datoteko pobrišemo med delovanjem odjemalca. Bria ponovno izvede ukaz »PUT« in ponovno pošlje XML datoteko z polno vsebino na strežnik.

6.2.2 Med nedelovanjem odjemalca

Na ta scenarij vpliva dejstvo, iz katerega računalnika se prijavljamo.

Izbrisano XML datoteko lahko ponovno ustvari in pošlje le odjemalec, ki je bil na računalniku zagnan in sinhroniziran. Bria namreč poleg WebDAV shranjevanja še vedno ustvari lokalno kopijo. Lokalna kopija pa lahko obstaja le, če jo je odjemalec že ustvaril in sinhroniziral z vsebino s strežnika.

Brisanje XML datoteke in ponovna prijava na novem računalniku povzroči izbris podatkov.

6.3 Prijava na novem računalniku

Odjemalec avtomatsko prenese kontakte s strežnika WebDAV.

6.4 Napaka v delovanju strežnika Apache

V primeru, ko strežnik Apache ni dosegljiv, ne moremo spreminjati kontaktov v odjemalcu. Ob začetku delovanja se namreč najprej pošlje ukaz »GET«, ki prejme podatke s strežnika. Ob prvotni sinhronizaciji se zato podatki, ki so bili popravljeni v času nedelovanja strežnika, izbrišejo.

6.5 Operacijski sistem Symbian

Preizkusi so izvedeni na telefonu Nokia E71 s »Symbian OS 9.2, S60 3rd Edition Feature Pack 1«.

6.5.1 Dodajanje kontakta

Ko v odjemalcu Bria dodamo kontakt, ga potisne na strežnik WebDAV. Skripta poskrbi za vnos podatkov v strežnik MySQL. Na telefonu sinhroniziramo in podatki se prenesejo v telefon.

6.5.2 Urejanje kontakta

Ko v odjemalcu Bria popravimo kontakt, ga potisne na strežnik WebDAV. Skripta poskrbi za vnos podatkov v strežnik MySQL. Na telefonu sinhroniziramo in podatki se prenesejo v telefon.

6.5.3 Brisanje kontakta

Ko v odjemalcu Bria izbrišemo kontakt, ga potisne na strežnik WebDAV. Skripta poskrbi za vnos podatkov v strežnik MySQL. Na telefonu sinhroniziramo in podatki se prenesejo v telefon.

6.6 Operacijski sistem Android

Preizkusi so izvedeni na tablici Point of View z operacijskim sistemom Google Android 2.2.

6.6.1 Dodajanje kontakta

Ko v odjemalcu Bria dodamo kontakt, ga potisne na strežnik WebDAV. Skripta poskrbi za vnos podatkov v strežnik MySQL. Na telefonu sinhroniziramo in podatki se prenesejo v telefon.

6.6.2 Urejanje kontakta

Ko v odjemalcu Bria popravimo kontakt, ga potisne na strežnik WebDAV. Skripta poskrbi za vnos podatkov v strežnik MySQL. Na telefonu sinhroniziramo in podatki se prenesejo v telefon.

6.6.3 Brisanje kontakta

Ko v odjemalcu Bria izbrišemo kontakt, ga potisne na strežnik WebDAV. Skripta poskrbi za vnos podatkov v strežnik MySQL. Na telefonu sinhroniziramo in podatki se prenesejo v telefon.

Poglavje 7

Php skripta za uvoz kontaktov

Trenutno imamo dve strani izdelka. Na eni strani imamo strežnik Funambol, ki se poveže s podatkovno bazo MySQL in zna avtorizirati uporabnika preko LDAP strežnika. Na drugi strani pa imamo program za komuniciranje, ki svoje kontakte shranjuje na spletni strežnik WebDAV v obliki XML datoteke.

Najprej smo preučili delovanje Funambola. S pomočjo vzvratnega inženirstva (*angl. reverse engineering*) smo ugotovili naslednja pomembna dejstva. V bazi MySQL se kontakti shranjujejo v dve tabeli *fnbl_pim_contact* in *fnbl_pim_contact_item*.

V tabeli *fnbl_pim_contact* so osnovni podatki o osebi, npr. ime, priimek, spol, itd. V tabeli *fnbl_pim_contact_item* pa so dodatni podatki kot so telefonska številka, e-poštni naslov, faks številka, itd.

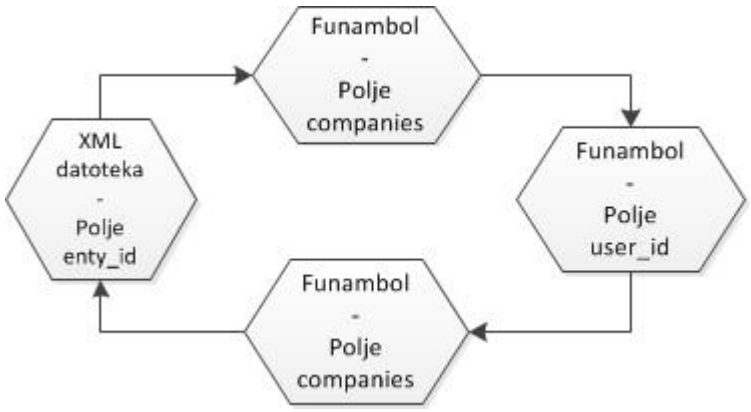
V naslednjem koraku smo ugotovili, kdaj in zakaj se funambol odloči za sinhronizacijo posameznega kontakta. V tabeli *fnbl_pim_contact* sta dve polji, ki skrbita za pravilno sinhronizacijo kontakta.

Polje *last_update* skrbi za časovno oznako zadnje sinhronizacije. Vzvratno inženirstvo pokaže, da je to čas v operacijskemu sistemu Linux, v sekundah, od 1.1.1970 naprej. Izpišemo jo z ukazom *date +%s*. Zadnji trije znaki so kontrolni, dovolj je, če dodamo tri ničle. Polje *status* ima tri možna stanja:

- Oznaka 'N' označuje nov kontakt (angl. *new*).
- Oznaka 'U' označuje popravljen kontakt (angl. *»updated«*).
- Oznaka 'D' označuje izbrisan kontakt (angl. *»deleted«*).

Razložiti podatke iz XML datoteke ni težko, saj so zapisani v nekodirani obliki. Problem je nastal le pri povezavi osebe iz odjemalca Bria s povezavo osebe iz strežnika Funambol. Zato smo uporabili polje *companies* v bazi

MySQL, da smo v njega shranili *entry_ id* iz datoteke XML. Tako smo lahko posredno pretvorili (Slika [??]) id in povezali dve osebi. To polje smo uporabili zato, ker smo po preučitvi dokumentacije ugotovili, da se pri sinhronizaciji ne uporablja več.



Slika 7.1: Preslikava identifikatorja iz odjemalca Bria v Funambol in obratno.

7.1 Delovanje skripte

Uporabili smo skriptni jezik php in napisali približno 2500 vrstic dolgo skripto (Slika [??]), ki skrbi za enosmerno sinhronizacijo iz XML datoteke v telefon oz. Funambol. Skripto zaženemo z ukazovom `./running.php` oz. `./running.php -nolog` v primeru da ne želimo dnevnika.

- Running.php

  Skripta, ki skrbi za zagon sinhronizacije, za vsakega uporabnika naredi datoteko *hash.txt*, kamor shrani *hash* vrednost pripadajoče datoteke *contacts-resource-list.xml*. V primeru da pri pregledovanju zapisana in izračunana koda nista enaki, sproži potek sinhronizacije. S tem prihranimo na porabi sistemskih sredstev. Ostale datoteke ne morejo biti zagnane samostojno, ampak za zagon skrbi *running.php*.

- Import.php

  Datoteka, ki skrbi za nadzor poteka sinhronizacije.

- Common.php

  Skupne funkcije, ki jih uporablja več datotek.

- New.php

  Kreiranje novega kontakta.

- Delete.php

  Izbris kontakta.

- Update.php

  Urejanje kontakta.

- Getters.php

  Funcije za vračanje parametrov, podatkov, itd.

- Settings.php

  Skupne nastavitve.

- personBria.php in personFnbl.php

  Oblika osebe iz Funambol-a oz. XML datoteke.

7.1.1 Opis skripte za uvoz

1. Začetek.

2. Ustvari seznam vseh map.

   Ustvari seznam vseh map v mapi »webdav«. Tako dobimo seznam vseh map, ki je hkrati tudi seznam vseh uporabnikov in njihovih uporabniških imen.

3. Vzemi XML datoteko.

   Uporabi eno XML datoteko, s katero bomo v nadaljevanju izvajali operacije.

4. Je HASH vrednost spremenjena.

   Vsaki XML datoteki izračunamo HASH vrednost z vgrajeno MD5 funkcijo v php jeziku. Vrednost zapišemo v pripadajočo mapo, v datoteko z imenom hash.txt. Skripta na začetku samo preveri vrednosti. Če se trenutna izračunana vrednost in zapisana vrednost ne ujemata, pomeni, da se je vsebina spremenila. Šele v tem primeru se zažene nadaljnja sinhronizacija.

5. Ustvari seznam vseh kontaktov.

   Iz datoteke preberemo vse kontakte in jih shranimo.

6. Vzemi kontakt.

   Iz seznama vseh kontaktov vzamemo samo en kontakt.

7. Preveri, če kontakt že obstaja.

   Preverimo, če kontakt že obstaja, da vemo ali moramo ustvariti novega ali samo popraviti že obstoječega.

8. Je kontakt enak ali spremenjen?

   Ker kontakt že obstaja, nas zanima, če je spremenjen ali enak.

9. Popravi kontakt.

   Sinhroniziramo kontakt v bazi in XML datoteki.

10. Dodaj novo osebo.

    Dodamo novo osebo v bazo.

11. Izbriši izbrane kontakte.

    Iz baze izbrišemo kontakte, ki jih ni več v XML datoteki. To namreč pomeni da so izbrisani.

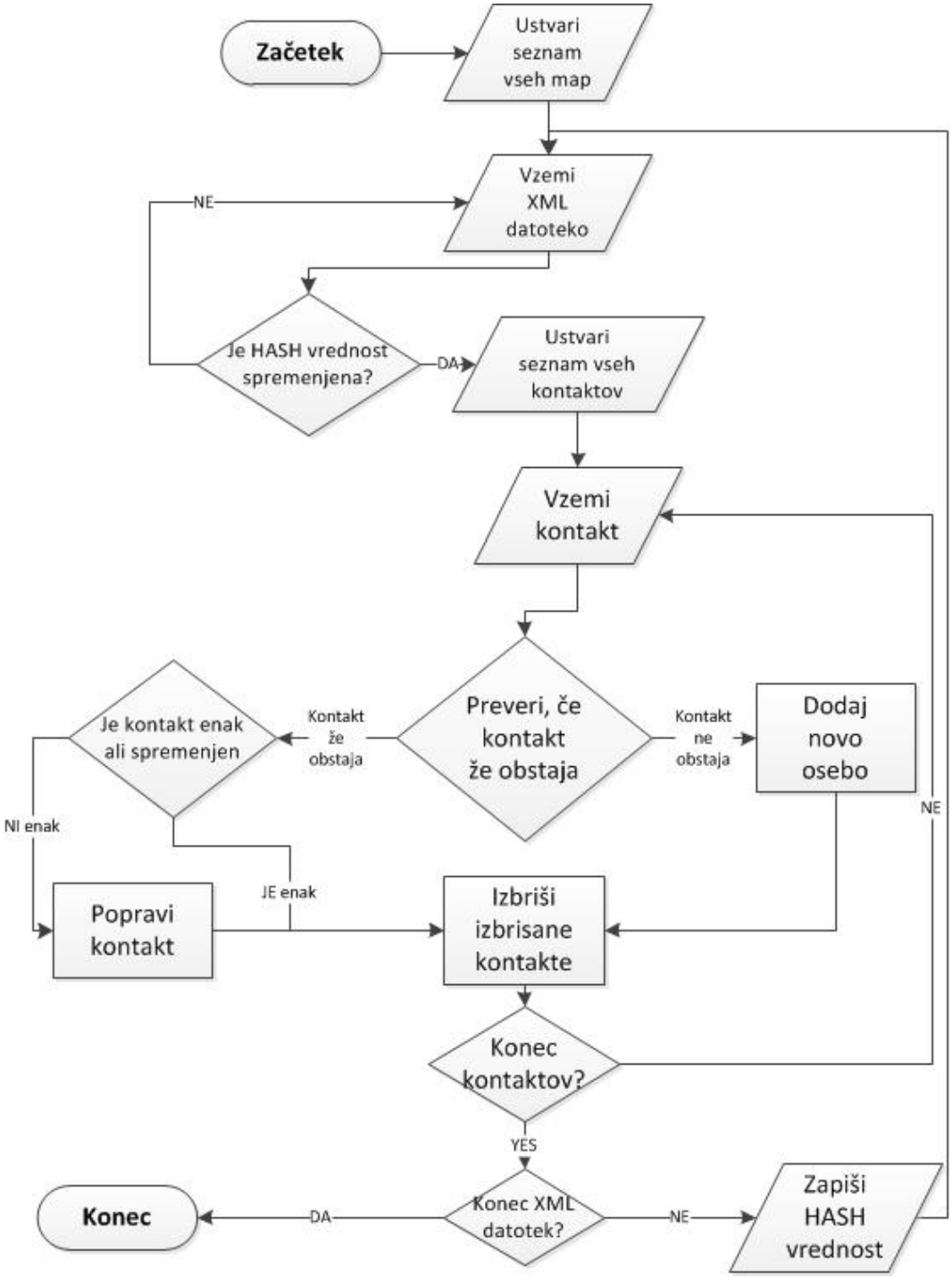
12. Konec kontaktov?

    Preverimo, če smo obdelali vse kontakte iz našega seznama.

13. Konec XML datoteke?

    Preverimo, če smo obdelali vse uporabnike iz našega seznama.

14. Konec.



Slika 7.2: Diagram poteka skripte za prenos podatkov iz odjemalca v Funambol.

Poglavje 8

Nadzor

Med samim razvojem rešitve smo uporabljali več programskih orodij za pregled in nadzor podatkov.

8.1 WEBMIN

Webmin je programsko orodje za nadzor in administracijo strežnikov. Namestitev je preprosta, saj so namestitvene datoteke izdelane za večino popularnejših operacijskih sistemov. Do orodja dostopamo preko spletnega vmesnika, na portu 10000. Na njihovi spletni strani si lahko ogledate spletni demo na naslovu http://webmin-demo.virtualmin.com/

Čeprav podpira več funkcij, smo ga mi uporabljali za administriranje strežnika LDAP in MySQL.

8.2 phpMyAdmin

PhpMyAdmin je orodje za administriranje in pregledovanje nameščene podatkovne baze MySQL. Temelji na php jeziku, zato je potrebno imeti nameščen php. Namestimo ga tako, da vsebino kompresirane datoteke prenesemo v naš korenski spletni imenik (/var/www/html/).

Uporabljali smo ga za nadzor in administracijo podatkovne baze MySQL.

8.3 phpLdapAdmin

PhpLdapAdmin je orodje za administriranje in pregledovanje nameščenega strežnika LDAP. Temelji na skriptnem jeziku php, zato je potrebno imeti name-

ščeno podporo za izvajanje php izvorne kode. Namestimo ga tako, da vsebino kompresirane datoteke prenesemo v naš korenski spletni imenik (/var/www/html/).

Uporabljali smo ga za nadzor in administracijo strežnika LDAP.

Poglavje 9

Nadgradnja sistema

9.1 Spletni portal

Izdelali bi spletni portal. Na njega bi se uporabnik prijavil z uporabniškim imenom in geslom, ki bi bilo isto kot za Funambol. Na portalu bi uporabniki lahko izbrali kontakte, ki jih želijo sinhronizirati.

9.2 Varnost

9.2.1 https povezava z odjemalcem Funambol

Strežnik Funambol ima možnost nastavitve varne povezave (https) z odjemalci. Tako bi dosegli, da bi bili preneseni podatki kriptirani. Uporabniki spremembe sicer ne bi občutili, pripomoremo pa veliko k varnosti.

9.2.2 https povezava s strežnikom WebDAV

Strežnik Apache ima možnost nastavitve varne povezave (https) z odjemalci. Tako bi dosegli, da bi bili preneseni podatki kriptirani. Uporabniki spremembe sicer ne bi občutili, pripomoremo pa veliko k varnosti.

9.2.3 https povezava s strežnikom LDAP

Strežnik openLDAP ima možnost nastavitve varne povezave (https) z odjemalci. Tako bi dosegli, da bi bili preneseni podatki kriptirani.

9.3 Obojestranska sinhronizacija

Želeli bi sinhronizirati podatke tudi v smeri od Funambol podprte naprave do odjemalca Bria. To je težje izvedljivo, ker Bria ne podpira ročnega uvoza kontaktov.

9.4 Odjemalec Funambol s podporo za enosmerno sinhronizacijo

Potrebno bi bilo popraviti izvorno kodo odjemalcev Funambol, da bi uporabnik imel možnost izbrati samo enosmerno sinhronizacijo. Tako se kontakti iz telefona, če tega ne bi hoteli, ne bi prenašali na strežnik.

Poglavje 10

Zaključek

Zadali smo si cilj sinhronizirati kontakte na namiznem odjemalcu za komuniciranje. Prioriteta je bila, da sinhroniziramo kontakte še na mobilne telefone, ostali odjemalci pa so bili dodaten bonus. Ob tem smo se držali nizkih stroškov, zato smo se morali osredotočati na odprtokodne rešitve.

Iz množice rešitev smo izbrali razširitev WebDAV za oddaljeno shranjevanje kontaktov in strežnik Funambol, ki v navezi s Funambol odjemalcem poskrbi za sinhronizacijo. Odjemalci obstajajo za večino aktualnih mobilnih telefonov, tablic, dlančnikov in odjemalcev za e-pošto.

Med samim procesom integracije smo se srečevali s problemi, ki so bili posredno povezani z našim celotnim projektom. Ocenimo lahko, da smo z uporabo odprtokodnih programov sicer znižali ceno projekta, smo si pa otežili nekatere faze.

Na koncu lahko zaključimo, da smo dobili delujočo rešitev, obenem pa smo potrdili koncept delovanja. Dokazali smo, da v fazi razvoja nismo naleteli na konceptualne napake, ki bi lahko zaustavile celoten projekt. Do faze, kjer bo sistem lahko vključen v produkcijo, pa bomo morali še dodelati nekatere dele.

Dodatek A

Priložen CD

- Izvorna koda skripte za uvoz podatkov
- Izvorna koda skripte za generiranje WebDAV nastavitev
- Celotna diplomska naloga v PDF obliki

Slike

Tabele

Literatura

[1] L. Kropivnik, "Do podatkov kjerkoli in kadarkoli!," 2008
Dostopno na:
http://www.monitor.si/clanek/do-podatkov-kjerkoli-in-kadarkoli2/

[2] S. Fornari, "Funambol Mobile Open Source", 2009.

[3] (2011) Android Sync Client Quick Start Guide
Dostopno na:
http://sourceforge.net/projects/funambol/files/docs/v10/Funambol-android-sync-client-quick-start-guide-v10.0.pdf/download

[4] (2011) Funambol Installation and Administration Guide
Dostopno na:
http://sunet.dl.sourceforge.net/project/funambol/docs/v9/Funambol-installation-and-administration-guide-v9.0.pdf

[5] (2011) Funambol Push Design Document
Dostopno na:
https://core.forge.funambol.org/source/browse/*checkout*/core/trunk/funambol/docs/ds-server/funambol-push-design.odt

[6] (2011) Funambol Server Architecture and Design Document
Dostopno na:
https://core.forge.funambol.org/source/browse/*checkout*/core/trunk/funambol/docs/ds-server/funambol-ds-service-design-document.odt

[7] (2011) Symbian Sync Client Quick Start Guide
Dostopno na:
http://garr.dl.sourceforge.net/project/funambol/docs/v9/Funambol-symbian-sync-client-quick-start-guide-v9.0.pdf

[8] (2011) Video, Integrating Funambol with CalDAV and LDAP
Dostopno na:
http://blip.tv/file/1605604

[9] (2011) Video, Introduction to Funambol and the team behind it
Dostopno na:
http://blip.tv/file/1605568

[10] (2011) Video, Funambol Connector development roadmap
Dostopno na:
http://blip.tv/file/1605539

[11] (2011) Video, Funambol Server Architecture
Dostopno na:
http://blip.tv/file/1605514