

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1302

DIREKTORIJ MIDDLEWARE SUSTAVA

Franjo Plavec

Zagreb, lipanj 2002.

*Zahvaljujem se prof. dr.sc. Siniši Srbliću na
pruženim savjetima i vođenju tijekom studija.
Zahvaljujem se dipl. inž. Ivanu Bencu na
pomoći pri izradi diplomskog rada.*

1 UVOD	1
2 SUSTAV ZAJEDNIČKIH FUNKCIJA	2
2.1 ULOGA SUSTAVA ZAJEDNIČKIH FUNKCIJA U RAČUNALNIM MREŽAMA.....	2
3 MODELI BAZE PODATAKA	6
3.1 HIJERARHIJSKI MODEL BAZE PODATAKA	6
3.2 MREŽNI MODEL BAZE PODATAKA	7
3.3 RELACIJSKI MODEL BAZE PODATAKA	8
3.4 OBJEKTNI MODEL BAZE PODATAKA	10
3.5 DIREKTORIJ	10
3.5.1 <i>Direktoriji u računalnim mrežama</i>	11
3.5.2 <i>ISO standard X.500</i>	12
<i>Organizacija podataka</i>	12
<i>Direktrijske usluge</i>	13
<i>Raspodjeljivanje direktorija</i>	14
<i>Uvišestručavanje podataka</i>	15
<i>Zaštita podataka od neovlaštenog korištenja</i>	15
<i>Direktrijski protokoli</i>	16
<i>Dodatne mogućnosti</i>	16
3.5.3 <i>Lightweight Directory Access Protocol (LDAP)</i>	16
<i>Informacijski model</i>	17
<i>Model imenovanja</i>	17
<i>Model funkcionalnosti</i>	18
<i>Sigurnosni model</i>	18
3.5.4 <i>Pregled postojećih direktorija</i>	19
<i>Netscape Directory Server</i>	19
<i>Netware Directory Services</i>	21
<i>Active Directory</i>	23
4 ACTIVE DIRECTORY	25
4.1 INTEGRACIJA ACTIVE DIRECTORY-A U WINDOWS 2000 SERVER	25
4.1.1 <i>Domain Name System (DNS)</i>	25
4.1.2 <i>Stabla i skupovi domena</i>	29
4.1.3 <i>Organizacijske jedinice</i>	30
4.1.4 <i>Grupe</i>	31
4.1.5 <i>Polja</i>	32
4.1.6 <i>Zaštita podataka od neovlaštenog korištenja</i>	32
<i>Kerberos V5 autentikacijski protokol</i>	33

SSL/TLS autentikacija.....	34
NTLM autentikacija.....	34
Sigurnosni opisnici.....	34
Zaštita mrežnog prometa.....	37
4.2 SPREMIŠTE PODATAKA DIREKTORIJA	38
4.2.1 Globalni katalog.....	38
4.2.2 Uvišestručavanje podataka.....	39
4.2.3 Posebne operacije nad podacima u direktoriju	41
4.3 ACTIVE DIRECTORY SHEMA	42
4.3.1 Definicija razreda.....	43
4.3.2 Definicije atributa.....	44
4.3.3 Proširenje Active Directory sheme.....	44
5 ARHITEKTURA I OSTVARENJE SPREMIŠTA PODATAKA.....	47
5.1 SPREMIŠTE PODATAKA SUSTAVA ZAJEDNIČKIH FUNKCIJA	47
5.2 ORGANIZACIJA PODATAKA U DIREKTORIJU	48
5.3 PROŠIRENJE SCHEME ZA SPREMIŠTE PODATAKA SUSTAVA ZAJEDNIČKIH FUNKCIJA.....	49
5.3.1 Atributi zapisa u shemi.....	49
5.3.2 Definicije atributa i razreda.....	50
5.3.3 Proširenje sheme.....	53
6 OSTVARENJE PROGRAMSKE POTPORE PRISTUPU DIREKTORIJU	55
6.1 MICROSOFT .NET FRAMEWORK.....	55
6.1.1 Common Language Runtime.....	55
6.1.2 Agregati.....	58
6.1.3 Tipovi podataka	60
6.1.4 Imenici.....	61
6.1.5 Upravljanje iznimkama	62
6.1.6 Višedretvenost.....	63
6.1.7 Ostale posebnosti .NET Framework platforme.....	64
6.2 ELEMENTI KNJIŽNICE OSNOVNIH RAZREDA KORIŠTENI PRI IZRADI PROGRAMSKE POTPORE 65	
6.2.1 Razred DirectoryEntry	66
6.2.2 Razred DirectoryEntries.....	67
6.2.3 Razredi PropertyCollection i PropertyValueCollection.....	68
6.2.4 Razred DirectorySearcher.....	68
6.2.5 Razredi SearchResultCollection i SearchResult.....	70
6.3 HIJERARHIJA RAZREDA	70
6.3.1 Razredi za upravljanje podacima.....	72
Razred ADGroupManager	72

<i>Razred ADUserManager</i>	72
<i>Razred ADServiceManager</i>	73
<i>Razred ADAuthorizationManager</i>	73
<i>Razred ADDeviceManager</i>	73
6.3.2 <i>Podatkovni razredi</i>	74
6.3.3 <i>Razred Cache</i>	75
6.3.4 <i>Razredi TreeNode i BinaryTree</i>	77
7 MJERENJA SVOJSTAVA PROGRAMSKE POTPORE PRISTUPU DIREKTORIJU.	79
7.1 OPIS MJERENJA I MJERNOG SUSTAVA	79
7.2 REZULTATI I ANALIZA REZULTATA MJERENJA	81
8 ZAKLJUČAK	86
LITERATURA	88

1 Uvod

Krajem dvadesetog stoljeća globalna mreža Internet postaje središnje mjesto razmjene informacija modernog društva. Razmjena informacija uključuje mnoge oblike elektroničkog poslovanja, međuljudske komunikacije, učenja na daljinu i zabave. Razmjena informacija odvija se pružanjem usluga. Usluge su Internet aplikacije koje se izvode na posebno građenim računalima, poslužiteljima usluga.

Analizom Internet aplikacija uočen je skup pomoćnih sustava zajednički mnogim aplikacijama. Najčešći pomoćni sustavi su: sustav registracije korisnika, sustav autentikacije korisnika, sustav autorizacije, te sustav praćenja korištenja usluga. Programer iznova ostvaruje navedene sustave za svaku Internet aplikaciju, što nepotrebno produljuje vrijeme razvoja aplikacije.

Vrijeme razvoja Internet aplikacija može se skratiti gradnjom sustava zajedničkih funkcija. Sustav zajedničkih funkcija ostvaruje navedene pomoćne sustave, i omogućuje njihovo korištenje putem sučelja. Sustav zajedničkih funkcija olakšava međusobnu suradnju Internet aplikacija jer aplikacije koriste jedinstveno sučelje za pristup mreži i komunikaciju s drugim Internet aplikacijama.

Sustav zajedničkih funkcija čuva podatke o korisnicima, uslugama, pravima korištenja pojedinih usluga, podatke o korištenju usluga i druge. Spremište podataka je podsustav sustava zajedničkih funkcija koji ostvaruje funkcionalnosti spremanja podataka. Pravilan rad i svojstva sustava zajedničkih funkcija ovise o pravilnom radu i svojstvima spremišta podataka, pa je pri izgradnji sustava potrebno pažljivo ostvariti odgovarajuće spremište podataka.

U ovom radu prikazano je ostvarenje spremišta podataka sustava zajedničkih funkcija korištenjem direktorija. Direktorij je skup otvorenih sustava koji omogućuje pohranu podataka u hijerarhijski organiziranu bazu podataka optimiranu za čitanje. U radu je dan pregled postojećih direktorija, opis njihova rada, opis ostvarenja spremišta podataka korištenjem direktorija i programske potpore pristupu direktoriju.

U drugom poglavlju opisan je sustav zajedničkih funkcija i njegove funkcionalnosti. Treće poglavlje sadrži opis modela baze podataka s naglaskom na hijerarhijski model i direktorij. U četvrtom poglavlju opisan je Active Directory, direktorij integriran u operacijski sustav Windows 2000 Server. U petom poglavlju opisana je arhitektura i ostvarenje spremišta podataka sustava zajedničkih funkcija. Ostvarenje programske potpore pristupu direktoriju opisano je u šestom poglavlju. U sedmom poglavlju prikazani su rezultati mjerenja svojstava programske potpore pristupu direktoriju. Zaključak je iznesen u osmom poglavlju.

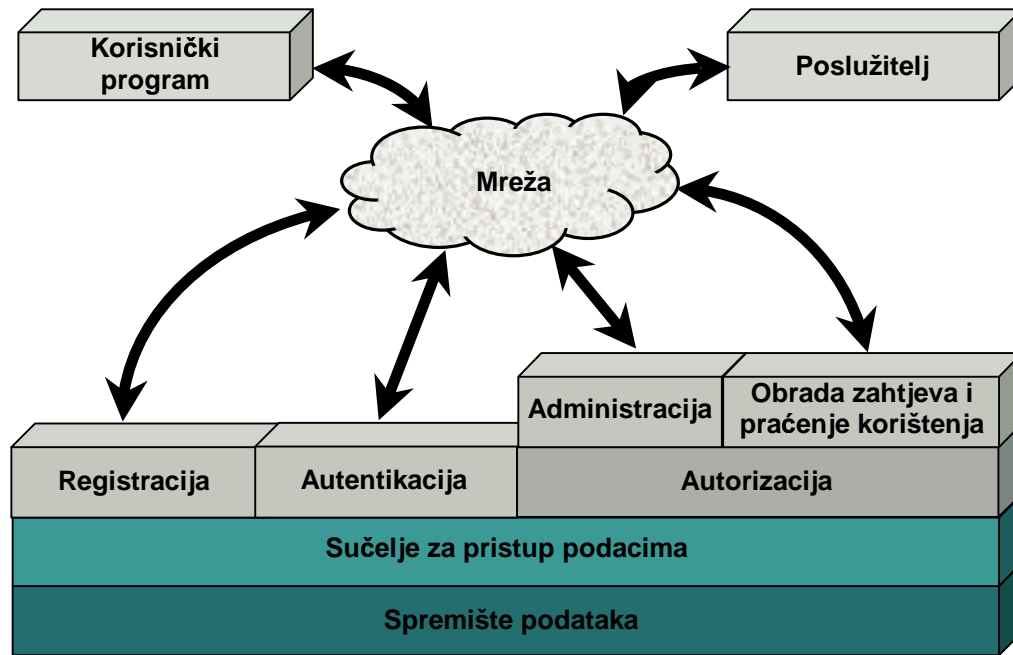
2 Sustav zajedničkih funkcija

Sustav zajedničkih funkcija pripada skupini sustava koji se zajedničkim imenom zovu middleware. Sustav pruža potporu suradnji Internet aplikacija kroz ostvarena sučelja. Sučelja omogućuju Internet aplikacijama pristup često korištenim funkcijama ostvarenima u sustavu zajedničkih funkcija. Sustav zajedničkih funkcija ostvaruje funkcije za registraciju i autentikaciju korisnika i usluga, autorizaciju, obradu zahtjeva i praćenja korištenja usluga, administraciju sustava zajedničkih funkcija, te spremanje podataka potrebnih za rad sustava.

2.1 Uloga sustava zajedničkih funkcija u računalnim mrežama

Sustav zajedničkih funkcija pripada skupini sustava zajedničkog imena middleware. Middleware je širok pojam i obuhvaća svaki oblik programske potpore koja omogućuje međusobnu suradnju različitih aplikacija, sustava i uređaja.

Osnovni mehanizmi suradnje aplikacija su metode poziva udaljenih procedura (eng. Remote Procedure Call) kao što su Java/RMI (eng. Java/Remote Method Invocation), DCOM (eng. Distributed Component Object Model), CORBA (eng. Common Object Request Broker Architecture) mrežne usluge (eng. Web Services) i druge. Nabrojene metode omogućuju povezivanje procesa koji se izvode na većem broju računala unutar neke mreže, pri čemu pojedina računala mogu predstavljati različite platforme s različitim operacijskim sustavima. Sva sučelja sustava zajedničkih funkcija ostvarena su kao mrežne usluge. Mrežne usluge omogućuju pozive udaljenih procedura među različitim računalnim platformama. Funkcionalnosti sustava zajedničkih funkcija prikazane su na slici 2-1.



Slika 2-1 Funkcionalnosti sustava zajedničkih funkcija

Većina Internet aplikacija zasnovana je na modelu korisnik-poslužitelj (eng. Client-Server Architecture). Korisnički programi koriste usluge koje pružaju poslužitelji. Da bi korisnički program mogao koristiti usluge koje ostvaruje poslužitelj, korisnik i usluge moraju biti registrirane. Sučelje za registraciju omogućuje prijavu novog korisnika ili usluge na sustav. Prilikom registracije, korisniku, odnosno usluzi, se dodjeljuje korisničko ime, lozinka i inicijalne postavke.

Sučelje za autentikaciju omogućuje provjeru identiteta korisnika, odnosno usluge. Provjera identiteta obavlja se korištenjem korisničkog imena i lozinke. Usluge i korisnici moraju biti autenticirani da bi mogli koristiti sučelja sustava.

Administracijsko sučelje ostvaruje funkcionalnosti potrebne za administraciju sustava zajedničkih funkcija. Administracijsko sučelje omogućuje upravljanje korisničkim pravima i pravima usluga, organiziranje korisnika i usluga u grupe, te upravljanje praćenjem korištenja.

Zahtjevi korisničkog programa prosleđuju se poslužitelju posredno, kroz sučelje za obradu zahtjeva i praćenje korištenja. Sučelje prima zahtjev od korisnika, određuje o kojem je korisniku riječ i, ako korisnik ima pravo koristiti zahtjevanu uslugu, prosleđuje zahtjev poslužitelju. Sučelje za obradu zahtjeva i praćenje korištenja prosleđuje korisniku odgovor na zahtjev dobiven od poslužitelja. S obzirom da svi korisnički zahtjevi prolaze kroz sučelje za obradu zahtjeva i praćenje korištenja, moguće je prikupljati podatke o korištenju usluga. Prikupljeni podaci mogu se koristiti za naplatu korištenja usluga.

Sučelje za autorizaciju omogućuje provjeru prava korištenja usluga i prava pristupa korisničkim podacima. Korisnik može koristiti samo one usluge za koje je autoriziran. Usluga

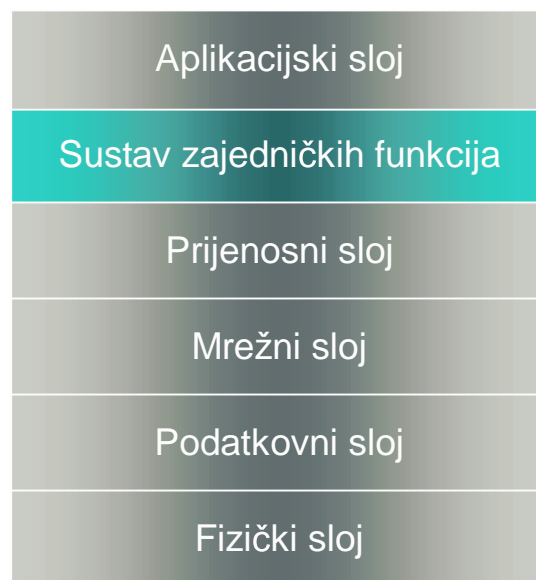
može dohvatiti samo one podatke o korisniku za koje je autorizirana. Sučelje za administraciju i sučelje za obradu zahtjeva i praćenje korištenja koriste funkcionalnosti sučelja za autorizaciju za definiranje i provjeru korisničkih prava i prava usluga.

Za pristup spremištu podataka sustava zajedničkih funkcija koristi se sučelje za pristup podacima. Sučelje za pristup podacima sakriva složenost spremišta podataka od ostatka sustava. Promjene u strukturi spremišta podataka ne utječu na ostatak sustava ako se ne mijenja sučelje za pristup podacima.

Spremište podataka sustava zajedničkih funkcija sadrži podatke o korisnicima, uslugama, pravima korištenja pojedinih usluga, korisničkim uređajima, sjednicama i korištenju usluga. Spremište podataka sastoji se od dva dijela: direktorija i baze podataka. Direktorij je hijerarhijski organizirana baza podataka optimirana za čitanje. Direktorij se koristi za spremanje podataka koji se učestalo čitaju i rijetko mijenjaju. Baza podataka je zasnovana na relacijskom modelu i koristi se za spremanje podataka koji se često mijenjaju i podataka o korištenju usluga.

Funkcionalnosti sustava zajedničkih funkcija pripadaju sjedničkom i predodžbenom sloju ISO OSI (eng. International Standards Organization Open Systems Interconnection) referentnog modela. OSI referentni model je komunikacijski model mreže računala standardiziran pri ISO [1].

Sustav zajedničkih funkcija je, prema ISO OSI referentnom modelu, smješten između prijenosnog i aplikacijskog sloja, kako je prikazano na slici 2-2.



Slika 2-2 Sustav zajedničkih funkcija u mrežnoj arhitekturi

Sjednički sloj ISO OSI referentnog modela ostvaruje funkcionalnosti potrebne za uspostavljanje i održavanje sjednica između korisničkih programa na udaljenim računalima. Sjednički sloj omogućuje funkcionalnosti kao što su prijenos podataka, upravljanje prijenosom, te rad na udaljenom računalu.

Predodžbeni sloj ISO OSI referentnog modela ostvaruje često korištene funkcionalnosti u mrežnim sustavima. To su funkcionalnosti zajedničke svim korisničkim programima, koje je moguće ostvariti u mreži. Ostvarenjem često korištenih funkcionalnosti u mreži, pojednostavljuje se izrada korisničkih programa.

Sustav zajedničkih funkcija je složen sustav i ostvaruje mnogo različitih funkcionalnosti. U nastavku ovog rada opisane su funkcionalnosti spremišta podataka sustava zajedničkih funkcija i sučelja za pristup podacima.

3 Modeli baze podataka

Baza podataka je skup međusobno povezanih podataka koji su spremljeni bez nepotrebne zalihosti radi kasnijeg korištenja. Podaci su u bazi podataka organizirani kao skupovi entiteta. Entitet je sve što ima bit, odnosno sve što se po nekim svojstvima razlikuje od drugih entiteta.

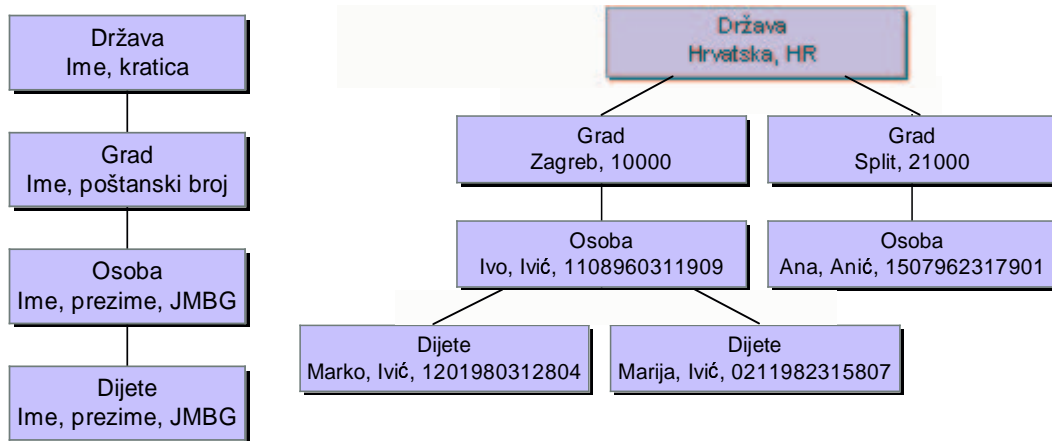
Model baze podataka definira način na koji su podaci spremljeni u bazi podataka, kao i metodologiju za učinkovitu pohranu i dohvat podataka. Četiri osnovna modela baza podataka su: hijerarhijski, mrežni, relacijski i objektni model.

Povijesno gledano, prvo se pojavio hijerarhijski model. U hijerarhijskom modelu podaci su organizirani hijerarhijski u strukturu stabla. Mrežni model je proširenje hijerarhijskog modela. Mrežni model uklanja neke nedostatke hijerarhijskog modela, ali je složeniji za ostvarenje. Većina baza podataka danas u upotrebi zasniva se na relacijskom modelu. Relacijski model omogućuje odvajanje logičke i fizičke organizacije podataka u bazi. Objektni model podržava osnovne elemente objektnog pristupa koji se koriste u objektno orijentiranim programskim jezicima. Objektni model omogućuje spremanje složenijih tipova podataka.

U ovom poglavlju posebni naglasak je stavljen na direktorij. Direktorij je posebna vrsta baze podataka zasnovana na hijerarhijskom modelu. Funkcionalnosti direktorija opisuje ISO standard X.500. Osnovni protokol za pristup direktoriju je Lightweight Directory Access Protocol (LDAP). Od postojećih direktorija opisani su Netscape Directory Server, Netware Directory Services i Active Directory.

3.1 *Hijerarhijski model baze podataka*

U povijesti razvoja baza podataka prvo se pojavljuje hijerarhijski model baze podataka. Podaci su u ovom modelu organizirani hijerarhijski. Hijerarhijska organizacija predstavlja se strukturom stabla. Primjer hijerarhijske organizacije podataka dan je na slici 3-1.



Slika 3-1 Hijerarhijska organizacija podataka

U strukturi stabla čvorovi predstavljaju entitete, a grane stabla veze među entitetima. U samoj bazi podataka veze se ostvaruju pomoću pokazivača. Stablo je podijeljeno na razine. Na vrhu stabla nalazi se korijenski čvor (u primjeru na slici to je čvor koji označava državu), koji je nadređen svim ostalim čvorovima u hijerarhiji. Svaki čvor, osim korijenskog, ima samo jedan čvor na razini neposredno iznad koji mu je nadređen. Svaki čvor može imati proizvoljan broj čvorova na razini neposredno ispod kojima je nadređen.

Primjer na slici 3-1 prikazuje strukturu i podatke u bazi podataka zasnovanoj na hijerarhijskom modelu. Država, grad, osoba i dijete predstavljaju entitete. Veze među entitetima predstavljaju odnose među entitetima u stvarnom svijetu.

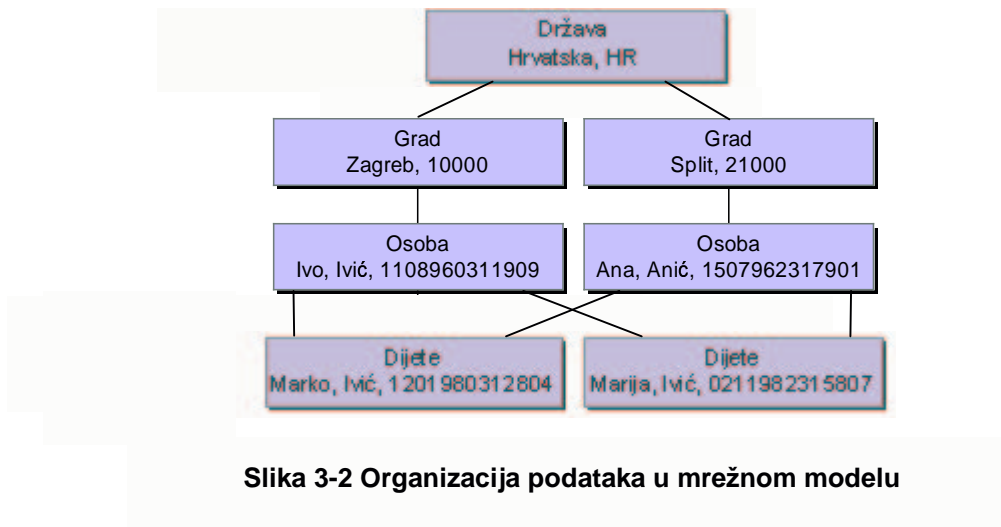
Prednost hijerarhijskog modela je brzina pretraživanja i dohvata podataka. Npr. kada se pronađe čvor s podacima o osobi, svi podaci o toj osobi nalaze se u tom čvoru, ili u podčvorovima tog čvora. To znači da nije potrebno izvoditi daljnje pretrage radi pronalaska drugih podataka o toj osobi, što nije slučaj za sve modele baze podataka.

Glavni nedostatak ovog modela je otežani prikaz podataka koji po svojoj prirodi nisu hijerarhijski organizirani. Na primjer (slika 3-1), problem nastaje ako se u bazu podataka želi spremiti podatak da je Ana Anić majka Marku i Mariji Ivić. Problem se može riješiti na dva načina. Prva mogućnost je da se podaci o djetetu spremaju u dva čvora. Jednom čvoru s podacima o djetetu nadređen je čvor s podacima o ocu, a drugome čvor s podacima o majci. Spremanjem istih podataka u dva različita čvora dobivena je neželjena zalihost. Druga mogućnost je proširenje hijerarhijskog modela baze podataka.

3.2 Mrežni model baze podataka

Mrežni model baze podataka je proširenje hijerarhijskog modela. Svaki čvor, osim korijenskog, može imati proizvoljan broj čvorova koji su mu neposredno nadređeni. Time se uklanja zalihost koja postoji u bazama podataka zasnovanim na hijerarhijskom modelu. Na

slici 3-2 prikazano je rješenje problema iz prethodnog poglavlja korištenjem mrežnog modela.

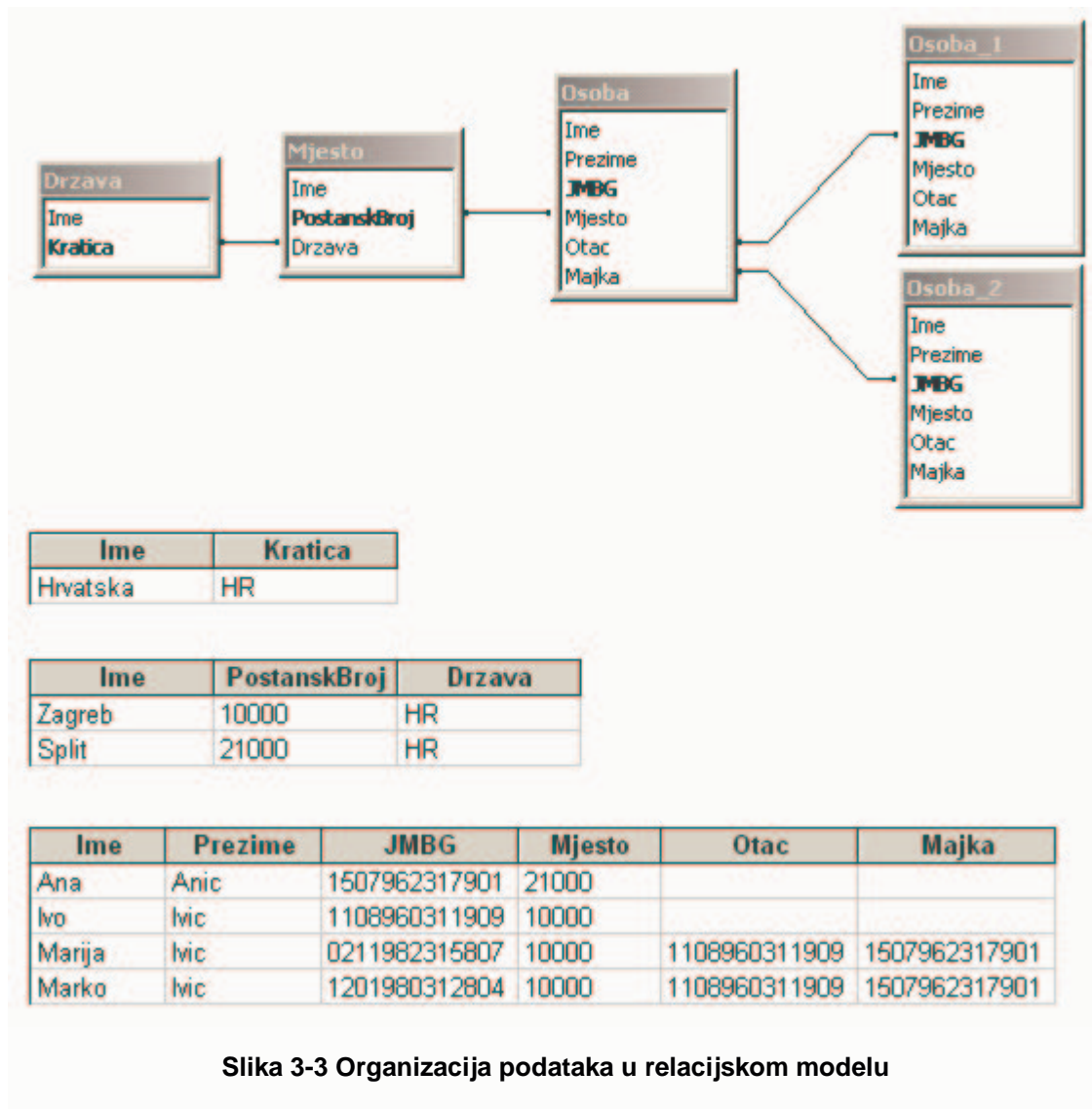


Osim smanjenja zalihosti, mrežni model donosi dodatna poboljšanja svojstava pri pretraživanju baze podataka. Nedostatak mrežnog modela je njegova složenost. Baza podataka zasnovana na mrežnom modelu nije jednostavna za izvedbu, niti je takvu bazu jednostavno održavati.

3.3 Relacijski model baze podataka

U hijerarhijskom i mrežnom modelu entiteti su povezani pokazivačima. Takav pristup zahtijeva od korisnika razumijevanje strukture baze podataka za njeno učinkovito korištenje. Relacijski model omogućuje odvajanje fizičke i logičke organizacije podataka spremljenih u bazi. Većina baza podataka koje su danas u upotrebi zasnivaju se na relacijskom modelu.

U relacijskom modelu entiteti su povezani preko svojstava entiteta. Svojstva entiteta se nazivaju atributi. Podaci su organizirani u tablice. Recipročne tablice označavaju entitete, a stupci attribute. Tablice su povezane zajedničkim atributima. Primjer organizacije podataka u relacijskom modelu dan je na slici 3-3. Naredni primjer ekvivalentan je primjeru iz prethodnog poglavlja.



Slika 3-3 prikazuje strukturu i sadržaj tablica relacijske baze podataka. U primjeru na slici entiteti su vezani preko vrijednosti atributa koji jedinstveno određuju entitet: mjesto i država su vezani preko kratice države, osoba i mjesto preko poštanskog broja, osoba i roditelj preko jedinstvenog matičnog broja građana. Ostvarenjem veza među entitetima preko vrijednosti atributa omogućena je nezavisnost podataka od fizičke organizacije. Krajnji korisnik ne mora znati ništa o fizičkoj organizaciji podataka. Krajnji korisnik upravlja isključivo apstraktnim objektima kao što su tablice, entiteti i atributi.

Baza podataka zasnovana na relacijskom modelu mora omogućiti upravljanje podacima u bazi kroz jedno, dobro definirano sučelje. Isto sučelje se koristi za upravljanje podacima o strukturi baze podataka. Najčešće je to SQL (eng. Structured Query Language). SQL je relacijski upitni jezik koji omogućuje pristup i upravljanje podacima u bazi.

Nedostatak relacijskog modela je manja efikasnost pri pretraživanju od hijerarhijskog modela. Da bi se pristupilo podacima o nekom entitetu, potrebno je izvesti više pretraga u više različitih tablica.

Atributi u bazi podataka zasnovanoj na relacijskom modelu ne mogu sadržavati složene podatke kao što su polja, strukture, i sl. Spremanje složenih tipova podataka omogućuje objektni model baze podataka.

3.4 *Objektni model baze podataka*

Objektni model baze podataka omogućuje spremanje složenijih tipova podataka. Baze podataka zasnovane na objektnom modelu posebno su pogodne za spremanje multimedijских sadržaja. Objektni model podržava osnovne elemente objektnog pristupa koji se koriste u objektno orijentiranim programskim jezicima:

- **Razred (eng. class)** – označava grupu objekata koji imaju zajednička svojstva.
- **Objekt (eng. object)** – opisuje objekt iz stvarnog svijeta. Svaki objekt pripada jednom razredu, odnosno njegova struktura i ponašanje odgovaraju.
- **Poruka (eng. message)** – komunikacija među objektima odvija se razmjenu poruka.
- **Nasljeđivanje (eng. inheritance)** – omogućuje korištenje već definiranih razreda za definiranje novih. Za novi razred se kaže da je izveden iz već definiranog razreda.
- **Metoda (eng. method)** – operacije nad objektima izvode se korištenjem metoda koje su definirane za razred kojemu objekt pripada.
- **Encapsulacija (eng. encapsulation)** – omogućuje skrivanje ostvarenja od krajnjeg korisnika. Krajnji korisnik vidi samo objekte, razrede i metode koje su definirane nad objektom.
- **Modularnost (eng. modularity)** – omogućuje građenje funkcionalnosti korak po korak, što olakšava gradnju složenih sustava.

Glavni nedostatak objektnog modela baze podataka je složenost. Zbog korištenja objektnih struktura, objektni model je često presložen za ostvarenje, te se stoga rijetko koristi.

3.5 *Direktorij*

Direktorij je skup otvorenih sustava koji omogućuje pohranu podataka u hijerarhijski organiziranu bazu podataka, zajedno s podacima koje baza podataka sadrži. Direktorij je namijenjen za spremanje podataka kojima se pristupa uglavnom radi čitanja (eng. read-mostly) [8].

Postojeći direktoriji dijele se u četiri skupine: direktoriji mrežnih operacijskih sustava, direktoriji ugrađeni u aplikacije, direktoriji posebne namjene i direktoriji opće namjene [6]. ISO standard X.500 definira usluge i funkcionalnosti koje interoperabilni raspodijeljeni

direktorij mora podržavati [7]. Za pristup direktoriju najčešće se koristi LDAP (eng. Lightweight Directory Access Protocol) protokol [9]. LDAP protokol je definiran kroz četiri modela: informacijski model, model imenovanja, funkcijski model i sigurnosni model.

3.5.1 Direktoriji u računalnim mrežama

U početku razvoja baza podataka hijerarhijski model baze podataka odbačen je kao nedovoljno efikasan za spremanje podataka koji po svojoj prirodi nisu hijerarhijski organizirani. Usprkos tome, posljednjih se godina ponovno javljaju sustavi zasnovani na hijerarhijskom modelu baze podataka u obliku direktorija.

Direktorij je skup otvorenih sustava koji omogućuje pohranu podataka u hijerarhijski organiziranu bazu podataka. Naziv direktorij se često koristi i za same podatke spremljene u bazu podataka. Broj pisanja u bazu podataka ima vrlo mali udio u ukupnom broju pristupa bazi. Nakon inicijalnog upisa podataka, bazi se pristupa uglavnom radi pretraživanja i čitanja informacija. Na taj način se iskorištava glavna prednost hijerarhijskog modela; efikasnost pri pretraživanju.

Direktorij je poput telefonskog imenika. U direktoriju se nalaze informacije koje su lako i brzo dostupne. Direktorij je pogodan za spremanje različitih vrsta podataka.

Razvojem Interneta i mobilnih komunikacija raste broj korisnika i raspoloživih podataka o korisnicima i uslugama. Zbog velikog broja korisnika javlja se problem pronalaženja odgovarajućih korisničkih informacija. Internet direktorij, u kojem su spremljeni podaci o tome gdje se određena informacija ili usluga nalazi, omogućuje učinkovito pretraživanje i pronalaženje tražene informacije.

Posljednjih godina računalne mreže postaju vrlo složene i sve ih je teže održavati. Nadalje, krajnjim korisnicima je sve teže pronaći odgovarajući resurs na mreži, bilo da se radi o dijeljenom pisaču, datoteci, ili drugim korisnicima. To je poseban problem za velike organizacije s velikim brojem umreženih računala. Direktorij koji sadrži informacije o resursima omogućuje lakše održavanje mreže, kao i pronalaženje odgovarajućih resursa na mreži. Direktorij je za ovu primjenu posebno pogodan s obzirom na prirodu organizacijske strukture poduzeća koja je u svojoj osnovi hijerarhijska.

Postojeći direktoriji dijele se u četiri skupine:

- **Direktoriji mrežnih operacijskih sustava** izgrađeni su specifično za potrebe mrežnih operacijskih sustava, ali često imaju mogućnost proširenja na druge primjene. U ovu skupinu spadaju Active Directory i Netware Directory Services.
- **Direktoriji ugrađeni u aplikacije** koriste se isključivo za spremanje podataka specifičnih za aplikaciju. Primjeri takvih direktorija su Lotus Notes adresar i Microsoft Exchange direktorij.
- **Direktoriji posebne namjene** nisu vezani uz aplikaciju, ali imaju točno određenu namjenu, i nije ih moguće koristiti u neku drugu svrhu. Najpoznatiji

direktorij ove vrste je DNS (eng. Domain Name System), sustav imenovanja računala na Internetu.

- **Direktorije opće namjene** moguće je koristiti u više različitih aplikacija. Obično su zasnovani na standardu ISO X.500. Primjer takvog direktorija je Netscape Directory Server.

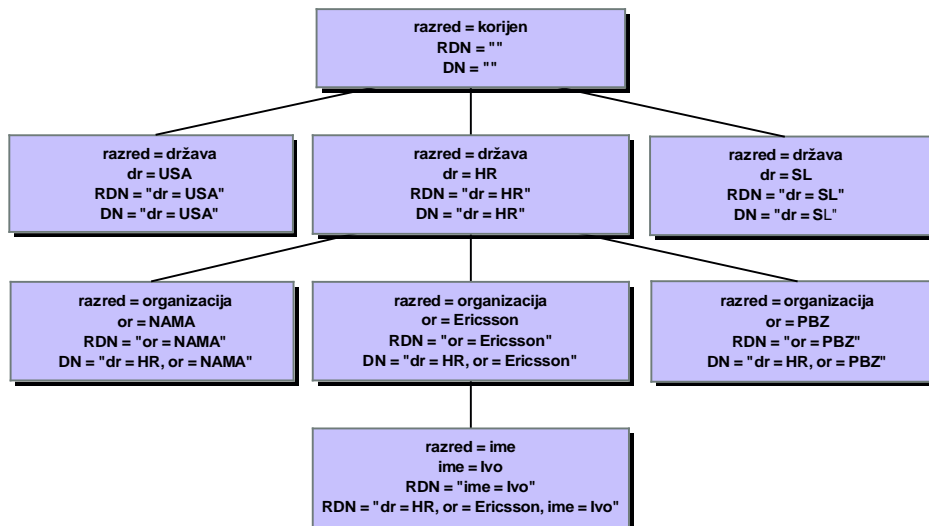
3.5.2 ISO standard X.500

X.500 je ISO standard nastao suradnjom ISO (eng. International Standards Organization) i CCITT (eng. Consultative Committee for International Telegraphy and Telephony). ISO standard X.500 definira usluge i funkcionalnosti koje interoperabilni raspodijeljeni direktorij mora podržavati. Pristup direktoriju ostvaruje se pomoću dviju komponenti direktorija: korisničke i sustavske. Korisnička komponenta direktorija (eng. Directory User Agent - DUA) je korisnička aplikacija koja pristupa direktoriju, dok sustavska komponenta direktorija (eng. Directory System Agent - DSA) pruža pristupnu točku direktoriju. Komunikacija između korisničke i sustavske komponente odvija se slanjem zahtjeva i odgovora na zahtjev između korisničke i sustavske komponente.

Organizacija podataka

Baza podataka direktorija naziva se informacijska baza direktorija (eng. Directory Information Base - DIB). Podaci se u informacijsku bazu spremaju u obliku zapisa (eng. entry). Svaki zapis sadrži informacije o nekom objektu (npr. osobi ili računalu). Zapis se sastoji od skupa atributa čije vrijednosti određuju svojstva objekta (npr. za osobu: ime, prezime, brojevi telefona,...). Razred zapisa definira koje atribute zapis može sadržavati. Tip atributa određuje koju vrstu podataka atribut sadrži (npr. niz znakova ili broj), te da li atribut sadrži jednu ili više vrijednosti. Skup pravila koji definira razrede zapisa, atribute koje oni sadrže i tipove atributa naziva se shema direktorija (eng. Directory schema).

Zapisi u informacijskoj bazi organizirani su hijerarhijski u strukturu informacijskog stabla direktorija (eng. Directory Information Tree - DIT). Čvorovi informacijskog stabla predstavljaju zapise, a veze među čvorovima odnos nadređenosti. Svaki zapis sadrži relativno i apsolutno ime. Relativno ime zapisa (eng. Relative Distinguished Name - RDN) sastoji se od jednog ili više atributa. Apsolutno ime zapisa (Distinguished Name - DN) sastoji se od niza relativnih imena koje se nalaze na putu od korijenskog čvora do čvora u kojem se nalazi odgovarajući zapis. Apsolutno ime jednoznačno određuje zapis. Primjer informacijskog stabla i imenovanja zapisa prikazan je na slici 3-4.



Slika 3-4 Informacijsko stablo direktorija

Prikazani direktorij sadrži informacije o korisnicima. Zapisi su organizirani hijerarhijski, prema geografskom položaju korisnika, te prema organizaciji u kojoj je korisnik zaposlen. Svaki zapis u danom primjeru može sadržavati i više atributa nego što je prikazano na slici. U primjeru na slici relativna imena zapisa sadrže vrijednost samo jednog atributa. Ako zapis ima više atributa (npr. ime, prezime i datum rođenja osobe), relativno ime može sadržavati vrijednosti više atributa.

Zapisi mogu sadržavati i druge zapise (eng. container) ako je tako definirano u shemi. U primjeru na slici 3-4 korijenski zapis sadrži zapise o državama, zapis o državi sadrži zapise o organizacijama, dok zapis o organizaciji sadrži zapise o korisnicima.

Osim zapisa o objektima, postoje i zamjenski zapisi (eng. alias entry). Zamjenski zapisi su pokazivači na zapise o objektima. Zamjenski zapisi omogućuju davanje alternativnih imena objektima.

Direktojske usluge

Direktorij pruža korisničkoj aplikaciji određene usluge. Sustavska komponenta direktorija prima zahtjeve od korisničke komponente, obrađuje ih, te vraća rezultate. Zahtjevi se dijele u dvije skupine: zahtjevi za pretraživanje (eng. directory interrogation) i zahtjevi za izmjenu podataka u direktoriju (eng. directory modification).

Zahtjevi za pretraživanje podataka u direktoriju obuhvaćaju sljedeće zahtjeve:

- **Čitaj** - (eng. Read) omogućuje dohvat vrijednosti svih atributa zadanog zapisa. Moguće je dohvatiti i vrijednosti samo nekih atributa, pri čemu je kao parametar potrebno prosljediti listu atributa čije se vrijednosti želi dohvatiti.
- **Usporedi** – (eng. Compare) omogućuje usporedbu vrijednosti zadanog atributa zapisa s nekom vrijednošću.

- **Izlistaj** – (eng. List) vraća listu svih zapisa kojima je zadani zapis neposredno nadređeni.
- **Pretraži** – (eng. Search) vraća podatke o zapisima koji zadovoljavaju zadani kriterij pretraživanja. Slično kao kod zahtjeva za čitanje moguće je dohvatiti vrijednosti svih, ili samo nekih atributa zapisa.
- **Odustani** – (eng. Abandon) primjenjuje se na zahtjev za pretraživanje koji je u tijeku. Označava da korisnička komponenta koja je uputila dotični zahtjev nije više zainteresirana za rezultate. Direktorij obustavlja pretragu, i odbacuje sve dotad dobivene rezultate.

Zahtjevi za izmjenu podataka u direktoriju obuhvaćaju sljedeće zahtjeve:

- **Dodaj zapis** – (eng. Add entry) dodaje novi zapis u informacijsko stablo direktorija.
- **Izbriši zapis** – (eng. Remove entry) briše zadani zapis iz informacijskog stabla direktorija.
- **Izmijeni zapis** – (eng. Modify entry) izvodi neku od izmjena nad zadanim zapisom. Izmjene uključuju dodavanje, brisanje i izmjenu vrijednosti atributa. Izvode se sve zadane izmjene, ili nijedna (eng. atomicity). Zadane izmjene ne smiju narušiti pravila definirana u shemi direktorija.
- **Izmijeni apsolutno ime zapisa** – (eng. Modify distinguished name) mijenja relativno ime zapisa ili pomiče zapis u informacijskom stablu direktorija na zadano mjesto, ovisno o novom apsolutnom imenu zapisa. Izmjena se na odgovarajući način izvodi i nad zapisima kojima je zadani zapis nadređen.

Prilikom prosljeđivanja zahtjeva sustavskoj komponenti, zadaju se parametri usluge. Parametri usluge, između ostalog, uključuju ograničenja na duljinu trajanja obrade zahtjeva, broj vraćenih rezultata, područje pretrage, te prioritet zahtjeva. Moguće je proslijediti i određene sigurnosne parametre potrebne za zaštitu podataka u direktoriju.

Prilikom obrade zahtjeva može doći do pogreške. Pogreška može nastupiti zbog neispravno zadanih parametara zahtjeva, narušavanja nekog od pravila zadanih u shemi direktorija, ili narušavanja sigurnosnih ograničenja. U tom slučaju povratna informacija je poruka o pogrešci.

Raspodjeljivanje direktorija

Direktorij pruža usluge velikom broju korisnika. Spremanje podataka u jednu, središnju bazu podataka, zahtijevalo bi velike količine resursa za održavanje baze i posluživanje klijenata. U slučaju prekida veze prema bazi, cjelokupna usluga bi bila nedostupna. Zbog toga se direktorij raspodjeljuje (eng. Distributed Directory).

Direktorij je raspodijeljen korištenjem sustavskih komponenti direktorija. Postoji nekoliko sustavskih komponenti direktorija, od kojih svaka upravlja jednim dijelom informacijske baze

direktorija. Sustavska komponenta direktorija je aplikacijski proces koji pruža pristup informacijskoj bazi direktorija korisničkim komponentama, ili drugim sustavskim komponentama direktorija. Međusobnom suradnjom sustavskih komponenti ostvareno je upravljanje svim podacima u direktoriju.

Uvišestručavanje podataka

Izvedbom direktorija kao raspodijeljene baze podataka javlja se problem uvišestručavanja podataka (eng. replication). Uvišestručavanje podataka je postupak preslikavanja (eng. copy) podataka iz jednog spremišta podataka u ostala spremišta radi usklađivanja. Uvišestručavanje podataka podrazumijeva postojanje preslika podataka kojima upravlja sustavska komponenta direktorija različita od sustavske komponente zadužene za stvaranje i izmjenu podataka.

Podaci u direktoriju se ne mijenjaju često, pa zahtjevi na uvišestručavanje nisu strogi. Izmjena u jednom spremištu podataka ne mora se prenositi u ostala spremišta u trenutku njenog nastanka. Dozvoljeno je da određeno vrijeme postoje različite preslike istog podatka u različitim spremištima, dok se ne obavi prijenos podataka. Ovo vrijeme nije unaprijed definirano, već ovisi o ostvarenju sustava uvišestručavanja. Važno je da svaka preslika za sebe bude dosljedna, te da se izmjena prenese u ostala spremišta podataka u konačnom vremenu.

ISO standard X.500 dozvoljava izgradnju direktorija koji ne ostvaruje uvišestručavanje podataka sadržanih u informacijskoj bazi direktorija [8].

Zaštita podataka od neovlaštenog korištenja

Zaštita podataka od neovlaštenog korištenja osigurava se pomoću autentikacije i kontrole pristupa podacima. Autentikacija omogućuje provjeru identiteta sustavskih i korisničkih komponenti. Općenite metode autentikacije opisane su u ITU-T (eng. International Telecommunication Union - Telecommunication Standardization Sector) X.509 preporuci [26]. Kontrola pristupa podacima omogućuje specificiranje informacija o pravima pojedinih entiteta nad zapisima u direktoriju. Prilikom uvišestručavanja podataka preslikavaju se i informacije o pravima pristupa zapisu. Jedan od mogućih načina ostvarenja kontrole pristupa podacima opisan je u ITU-T X.501 preporuci.

Osim neposredne kontrole pristupa podacima moguća je i kontrola direktorijskih usluga. U svrhu zaštite podataka moguće je ograničiti pojedine direktorijske usluge kako bi se spriječilo zloupotrebu podataka. Onemogućavanjem određenih vrsta upita može se npr. spriječiti korištenje informacija u direktoriju u marketinške svrhe.

Direktorijski protokoli

Direktorijski protokoli omogućuju međusobnu suradnju korisničkih i sustavskih komponenti direktorija u različitim otvorenim sustavima. Postoje četiri direktorijska protokola:

- **Protokol za pristup direktoriju** – (eng. Directory Access Protocol – DAP) definira način izmjene zahtjeva i rezultata obrade zahtjeva između korisničke i sustavske komponente direktorija.
- **Sustavski protokol direktorija** – (eng. Directory System Protocol - DSP) definira način izmjene zahtjeva i rezultata obrade zahtjeva između dvije sustavske komponente direktorija.
- **Protokol za uvišestručavanje** – (eng. Directory Information Shadowing Protocol - DISP) definira način preslikavanja podataka između dvije sustavske komponente direktorija.
- **Operacijski protokol** – (eng. Directory Operational Binding Management Protocol – DOP) definira način izmjene administrativnih informacija između dvije sustavske komponente direktorija.

Specifikacije ovih protokola opisane su u ITU-T X.519 preporuci [27].

Dodatne mogućnosti

Ovisno o namjeni direktorija, moguća je potpora za neke dodatne usluge. Dodatne usluge obuhvaćaju npr. pregled svih zapisa u direktoriju (eng. browsing), tzv. žute stranice (eng. Yellow pages), organizaciju korisnika po grupama, te upravljanje uvjetima pretrage u slučajevima kada zadani uvjeti rezultiraju prevelikim ili premalim brojem vraćenih rezultata. Dodatne usluge mogu se ostvariti korištenjem kombinacije već opisanih usluga.

3.5.3 Lightweight Directory Access Protocol (LDAP)

Zbog svoje složenosti X.500 DAP protokol za pristup direktoriju postavlja velike zahtjeve na računalnu snagu sustava, te je njegova upotreba ograničena na snažna računala zasnovana na UNIX operacijskom sustavu. LDAP protokol je izgrađen da se omogući jednostavniji pristup i upravljanje raspodijeljenim direktorijem. Dizajniran je za rad na mrežama zasnovanim na TCP/IP protokolu. LDAP definira način na koji korisnički program pristupa sustavskoj komponenti, te kako korisnički program izvodi operacije nad direktorijem.

Premda je LDAP u početku zamišljen samo kao protokol za pristup direktorijima zasnovanim na X.500 standardu, danas postoje direktoriji koji su u potpunosti zasnovani na LDAP protokolu. To znači da se sve operacije nad direktorijem obavljaju korištenjem LDAP protokola. LDAP podržava niz funkcionalnosti koje nisu bile predviđene u početku njegova razvoja, kao što su autentikacija i uvišestručavanje. Korištenjem LDAP protokola kao osnove za izgradnju direktorija smanjuju se zahtjevi na računalnu snagu korisničkih i poslužiteljskih računala.

LDAP je opisan preko četiri modela: informacijskog modela, modela imenovanja, modela funkcionalnosti i sigurnosnog modela [9].

Informacijski model

Informacijski model opisuje strukturu informacijskog stabla direktorija. Informacijski model izveden je iz ISO X.500 standarda, pa je i terminologija korištena u opisu informacijskog modela preuzeta iz X.500 standarda. Za informacijski model važni su sljedeći pojmovi:

- **Razred** označava grupu objekata koji imaju zajednička svojstva. Razredi se mogu nasljeđivati. Postoje tri vrste razreda: apstraktni razredi (eng. abstract classes) služe kao predlošci za strukturne razrede, strukturni razredi (eng. structural classes) opisuju zapise u direktoriju, dok pomoćni razredi (eng. auxiliary classes) definiraju skup atributa za nasljeđivanje.
- **Atributi** su jedinice podataka pomoću kojih se definiraju razredi. Atributi se u shemi definiraju zasebno od razreda, te je tako omogućeno korištenje iste definicije atributa u više različitih razreda.
- **Sintaksa atributa** definira koju vrstu podataka i koje vrijednosti može sadržavati pojedini atribut.
- **Zapisi** opisuju objekte iz stvarnog svijeta. Svaki zapis je pojavnost (eng. instance) jednog strukturnog razreda, što znači da sadrži vrijednosti i poštuje ograničenja atributa definiranih u razredu.
- **Shema** sadrži listu razreda i atributa koji se mogu koristiti i nasljeđivati. Da bi zapis pripadao informacijskom stablu direktorija, mora odgovarati formatu definiranom u shemi.

Model imenovanja

Model imenovanja definira organizaciju i referenciranje podataka. Model imenovanja preuzet je iz ISO X.500 standarda. Podaci su organizirani u informacijsko stablo direktorija (DIT). Za pristup čvoru informacijskog stabla kao primarni ključ koristi se apsolutno ime (DN), sastavljeno od niza relativnih imena (RDN).

Korijen informacijskog stabla direktorija (eng. root DSE) sadrži zapis karakterističan za sustavsku komponentu direktorija (eng. DSA specific entry - DSE). Zapis sadrži podatke o sustavskoj komponenti direktorija kao što su: podržana verzija protokola, podržane napredne operacije, podržani sigurnosni mehanizmi, adrese alternativnih sustavskih komponenti i adresu zapisa koji sadrži shemu.

Model funkcionalnosti

Model funkcionalnosti definira operacije nad podacima u direktoriju. Postoji ukupno devet operacija podijeljenih u tri skupine:

- **Autentikacija** omogućuje korisničkom programu da dokaže svoj identitet kroz sljedeće operacije:
 - *Open* – otvara vezu prema sustavskoj komponenti direktorija (DSA).
 - *Bind* – otvara sjednicu između korisničkog programa i sustavske komponente koja omogućuje razmjenu podataka potrebnih za autentikaciju.
 - *Unbind* – prekida sjednicu između korisničkog programa i DSA.
- **Pretraživanje** je moguće korištenjem sljedećih operacija:
 - *Search* – služi za pretraživanje direktorija. Kriteriji za pretragu prenosi se preko parametara. Moguće je proslijediti parametre koji određuju čvor u informacijskom stablu gdje pretraga počinje, područje koje se pretražuje, filter pretrage, listu atributa koji se vraćaju, te parametre koji određuju način izvođenja pretrage.
 - *Compare* – vraća vrijednost istinitosti za zadanu usporedbu.
- **Izmjena** nad podacima izvodi se korištenjem sljedećih operacija:
 - *Add* – stvara objekt u informacijskom stablu direktorija. Stvoreni objekt mora zadovoljavati uvjete definirane u shemi.
 - *Modify* – mijenja vrijednost određenog atributa zapisa. Mijenjanje obuhvaća dodavanje, izmjenu i brisanje vrijednosti atributa.
 - *Modify RDN* – omogućuje micanje zapisa unutar informacijskog stabla direktorija.
 - *Delete* – omogućuje brisanje zapisa iz informacijskog stabla direktorija.

Sigurnosni model

Sigurnosni model definira mogućnosti sigurnog pristupa podacima unutar informacijskog stabla direktorija. Standard definira korištenje postojećih Simple Authentication and Security Layer (SASL) mehanizama za osiguravanje pristupa podacima. Korijen informacijskog stabla direktorija (root DSE) sadrži atribut supportedSASLMechanisms koji sadrži listu podržanih SASL sigurnosnih mehanizama. SASL sigurnosni mehanizmi koriste se za sigurnu autentikaciju, a po potrebi je moguće zaštititi i cjelokupnu komunikaciju između korisničkog programa i sustavske komponente direktorija. SASL sigurnosni mehanizmi opisani su u RFC (eng. Request For Comment) 2222 [11].

3.5.4 Pregled postojećih direktorija

Direktoriji posebne namjene i direktoriji ugrađeni u aplikacije ne mogu biti korišteni kao spremišta proizvoljnih podataka, pa oni nisu razmatrani u okviru ovoga rada. Spremište podataka sustava zajedničkih funkcija moguće je ostvariti pomoću direktorija opće namjene, ili nekog od direktorija mrežnih operacijskih sustava proširenog na odgovarajući način. Kao primjer direktorija opće namjene opisan je Netscape Directory Server, dok su kao primjer direktorija mrežnih operacijskih sustava opisani Netware Directory Services i Active Directory.

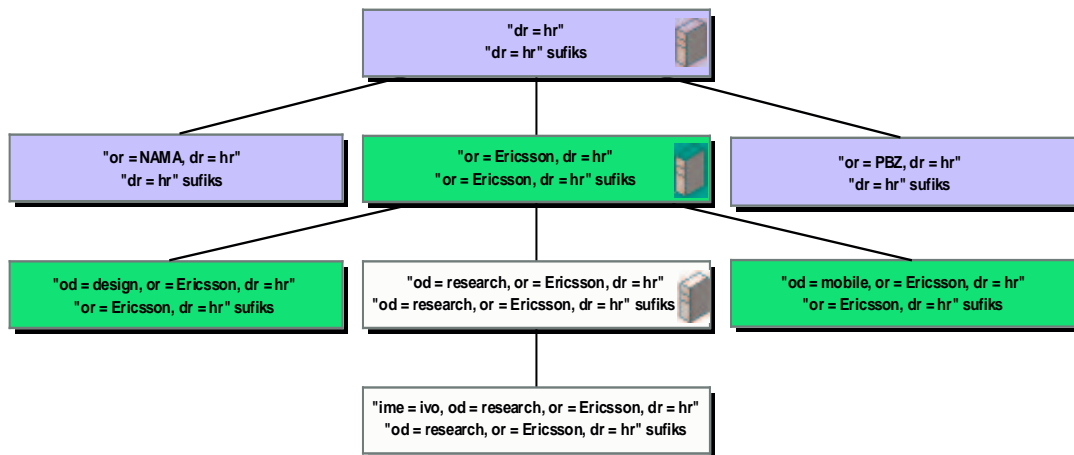
Netscape Directory Server

Netscape Directory Server je direktorij opće namjene zasnovan na LDAP protokolu [13]. Netscape Directory Server je poznat i pod nazivom iPlanet Directory Server. Netscape Directory Server je neovisan o platformi. Moguće je njegovo ostvarenje na nekoliko Unix platformi, kao i na Windows NT platformama. Arhitektura direktorija sastoji se od direktorijskog poslužitelja i korisničkih programa.

Direktorijski poslužitelj sastoji se od dva dijela: sučelja za pristup direktoriju i sustava za upravljanje bazom podataka. Sučelje za pristup direktoriju prima od korisničkih programa zahtjeve za pretraživanje i izmjenu podataka u direktoriju. Osim LDAP protokola, podržano je i HTML (eng. HyperText Markup Language) sučelje za pristup direktoriju.

Sustav za upravljanje bazom podataka sadrži hijerarhijski organiziranu bazu podataka koja omogućuje spremanje i održavanje velikog broja zapisa organiziranih u informacijsko stablo direktorija (DIT). Da bi se omogućila provjera ispravnosti operacija koje se provode nad podacima u bazi, moguće je napisati kod koji se izvodi prije, ili nakon određene operacije.

Jedan direktorijski poslužitelj ne upravlja nužno cijelim informacijskim stablom direktorija. Informacijsko stablo moguće je podijeliti na sufikse (eng. suffix). Sufiks predstavlja dio stabla kojim upravlja jedan direktorijski poslužitelj, a identificiran je apsolutnim imenom korijenskog zapisa odgovarajućeg dijela stabla. U sustavu postoje najmanje dva sufiksa: korisnički definirani primarni sufiks i sustavski definirani sufiks. Primarni sufiks je korijen informacijskog stabla direktorija, i definira se pri instalaciji Netscape Directory Server-a. Sustavski definirani sufiks sadrži podatke o direktorijskom poslužitelju. Slika 3-5 prikazuje primjer podjele informacijskog stabla direktorija na sufikse.



Slika 3-5 Podjela informacijskog stabla direktorija na sufikse

U primjeru na slici 3-5 postoje tri sufiksa. Apsolutno ime primarnog sufiksa je "dr = hr". Preostala dva sufiksa imaju apsolutna imena "or = Ericsson, dr = hr" i "od = research, or = Ericsson, dr = hr". Zapisi koji pripadaju sufiksima i direktorijski poslužitelji koji njima upravljaju označeni su odgovarajućom bojom.

Da bi korisnički programi mogli pristupiti podacima neovisno o tome koji direktorijski poslužitelj upravlja tim podacima u direktorij je ugrađen mehanizam "pametnog preusmjerenja" (eng. smart referrals). Ovaj mehanizam omogućuje preusmjerenje zahtjeva za upravljanje podacima direktorijskom poslužitelju koji upravlja sufiksom koji sadrži dotični podatak. Na primjer, ako direktorijski poslužitelj, prikazan na slici 3-5, koji upravlja sufiksom "dr = hr", primi zahtjev za dohvat zapisa apsolutnog imena "od = design, or = Ericsson, dr = hr", zahtjev će biti preusmjeren direktorijskom poslužitelju sufiksa "or = Ericsson, dr = hr".

Uvišestručavanje podataka zasnovano je na modelu proizvođač – potrošač. Poslužitelj proizvođač (eng. supplier server) dostavlja podatke poslužiteljima potrošačima. Svaki zapis u direktoriju ima samo jednog pripadajućeg poslužitelja proizvođača na kome se izvode sve izmjene podataka. Poslužitelj potrošač dozvoljava samo čitanje podataka. Ako poslužitelj potrošač primi zahtjev za izmjenu podataka, zahtjev se preusmjeruje poslužitelju proizvođaču koji dostavlja odgovarajuće podatke.

Ovisno o administratorskim postavkama, moguće je da proizvođač potrošačima dostavlja podatke cijelog sufiksa kojim upravlja, ili samo određeni dio. Svaki potrošač može ujedno biti i proizvođač, čime se omogućuje raspodjela opterećenja koje bi nastalo ako bi jedan proizvođač bio zadužen za dostavljanje podataka prevelikom broju potrošača.

Skup osnovnih razreda i atributa za opis objekata na mreži već je predefiniran u shemi direktorija. Skup osnovnih razreda moguće je proširiti korisnički definiranim razredima i atributima. Prilikom svake izmjene ili dodavanja zapisa u direktorij obavlja se provjera pravila definiranih u shemi. Operacija izmjene ili dodavanja zapisa uspijeva samo ako nove

vrijednosti zadovoljavaju pravila definirana u shemi. Postupak provjere pravila definiranih u shemi može se onemogućiti. Onemogućavanje postupka provjere nije preporučljivo jer upisivanje ili izmjena podataka bez provjere može dovesti do nedosljednosti podataka.

Korisnički programi koriste usluge koje direktorijski poslužitelj pruža kroz sučelje. Netscape isporučuje nekoliko korisničkih programa koji koriste usluge direktorijskog poslužitelja. Među ostalima, to su Netscape Administration Server koji omogućuje administraciju različitih mrežnih sustava i Netscape Communicator 4.0 koji koristi direktorij kao spremište podataka za adresar.

Netscape Directory SDK (eng. Software Development Kit) je skup programskih sučelja koja omogućuju pisanje korisničkih programa s mogućnošću korištenja direktorija kao spremišta podataka. Podržani jezici su C, Java, JavaScript i Perl.

Sigurnost podataka u direktoriju osigurava se kroz autentikaciju, kontrolu pristupa i kriptiranje poruka koje razmjenjuju poslužitelj i korisnički program. Za autentikaciju se koristi korisničko ime i lozinka, X.509 certifikat [25] ili korisnički definirani autentikacijski mehanizam. Kontrola pristupa omogućuje definiranje prava izvođenja određenih operacija nad određenim podacima. Moguće je definirati koji korisnik smije izvoditi operacije nad zapisima, ili određenim atributima zapisa. Kontrola je moguća na razini svakog pojedinog korisnika, svake pojedine operacije, te svakog pojedinog zapisa, odnosno atributa zapisa. Poruke koje razmjenjuju poslužitelj i korisnički program kriptiraju se pomoću SSL (eng. Secure Sockets Layer) protokola.

Netscape Directory Server pripada skupini direktorija opće namjene. Unatoč tome, Netscape Directory Server se koristi samo u alatima koje isporučuje Netscape. Direktoriji konkurentskih proizvođača pokazali su se znatno boljim rješenjima kako po podržanim mogućnostima, tako i po svojstvima.

Netware Directory Services

Netware Directory Services (NDS) je direktorij ugrađen u mrežni operacijski sustav Novell NetWare. Integracijom direktorija u mrežni operacijski sustav znatno je olakšano upravljanje mrežom računala [16].

Klasična mreža računala sastoji se od niza povezanih računala koja međusobno surađuju da bi omogućila obavljanje nekog zadatka. Na jednom računalu u mreži može raditi više korisnika, i jedan korisnik može raditi na više računala. Da bi se korisniku omogućio rad na više računala, potrebno je na svakom od računala stvoriti novi korisnički račun i dodijeliti odgovarajuća prava pristupa resursima računala. Definiranje prava pristupa resursima na svakom računalu, za svakog korisnika pojedinačno, je dugotrajan posao.

Direktorij u mrežnom operacijskom sustavu služi kao središnje spremište podataka o svim resursima na mreži. Pod resursima se podrazumijevaju računala i računalna oprema, ali i aplikacije i korisnici, te radne organizacije i grupe korisnika. Sva računala u mreži dobivaju informacije o resursima na mreži iz direktorija. U takvoj mreži dovoljno je samo

jednom stvoriti korisnički račun korisnika, te mu dodijeliti odgovarajuća prava pristupa ostalim resursima. Korisnik može pristupati resursima na mreži s bilo kojeg računala, jer sva računala mogu autenticirati korisnika i iz direktorija pročitati prava pristupa drugim resursima.

Osnovni protokol za pristup direktoriju je NDAP (eng. Novell Directory Access Protocol). Premda je Netware Directory Services prvenstveno namijenjen kao središnje spremište podataka o objektima na mreži mrežnog operacijskog sustava Novell NetWare, može surađivati i s drugim operacijskim sustavima. U tu svrhu, Netware Directory Services podržava i druge protokole i sučelja: LDAP, HTTP, ODBC (eng. Open Database Connectivity), te ADSI (eng. Active Directory Service Interfaces). Netware Directory Services nije nužno vezan uz Novell NetWare. Direktorij se može ostvariti i na drugim platformama: Red Hat Linux, Sun Microsystems Solaris, Microsoft Windows NT 4, Microsoft Windows 2000 i Compaq Tru64. Osim samog direktorija, Novell isporučuje nekoliko aplikacija za administraciju složene mreže sastavljene od računala zasnovanih na različitim platformama, te niz korisničkih programa s mogućnošću izvođenja na drugim platformama.

Netware Directory Services podržava zaštitu podataka standardiziranim sigurnosnim mehanizmima. Prilikom autentikacije korisnika koriste se RSA algoritmi kriptiranja. Korisničke lozinke u direktoriju su kriptirane Triple DES (eng. Data Encryption Standard) algoritmom. Ako se za komunikaciju između korisničkog programa i poslužitelja koristi LDAP protokol, podržana je zaštita poruka korištenjem SSL protokola i PKCS (eng. Public Key Cryptography Standards) standarda. Kontrola pristupa zapisima u direktoriju ostvarena je listama za kontrolu pristupa (eng. Access Control Lists – ACLs). Lista za kontrolu pristupa svakog zapisa sadrži listu svih korisnika koji imaju pravo pristupa tom zapisu, s definiranim pravima nad zapisom. Prava pristupa se automatski prenose od zapisa za koji su definirana na zapise niže u hijerarhiji. Time je omogućena jednostavna definicija prava pristupa za cijelo podstablo direktorija. Automatski prijenos prava pristupa može se spriječiti korištenjem filtera (eng. Inherited Rights Filter).

Schema direktorija sadrži predefinirane razrede za opis osnovnih resursa na mreži koje koristi Novell NetWare. Ako se u direktorij žele spremati neki podaci drugog tipa, moguće je definiranje vlastitih razreda.

Uvišestručavanje podataka omogućuje veću pouzdanost i poboljšanje svojstava sustava. Svaka preslika originalnih podataka može se čitati, a ovisno o konfiguraciji moguće je izvoditi i izmjenu podataka na bilo kojoj preslici podataka. Nastala izmjena se mehanizmom uvišestručavanja prenosi u ostala spremišta podataka koja sadrže preslike istih podataka. S obzirom da postoji više preslika u kojima se podaci mogu mijenjati, moguća je pojava dvije ili više različitih preslika istog podatka. U tom slučaju, posljednja izvedena izmjena postaje važeća.

Netware Directory Services je jedini direktorij na tržištu koji omogućuje povezivanje i suradnju računala zasnovanih na različitim platformama. Pri tome je osnovni nedostatak gubitak svojstava, ako se ne koristi NDAP protokol. Potpora ostalim protokolima ostvarena je

prevođenjem zahtjeva u oblik definiran NDAP protokolom, te prevođenjem NDAP odgovora na zahtjev, u oblik definiran danim protokolom. Postupak prevođenja zahtjeva i odgovora na zahtjev značajno utječe na svojstva sustava. Za potrebe uvišestručavanja i nadogradnje sheme nije podržana upotreba drugih protokola, odnosno, nužno se koristi NDAP protokol.

Active Directory

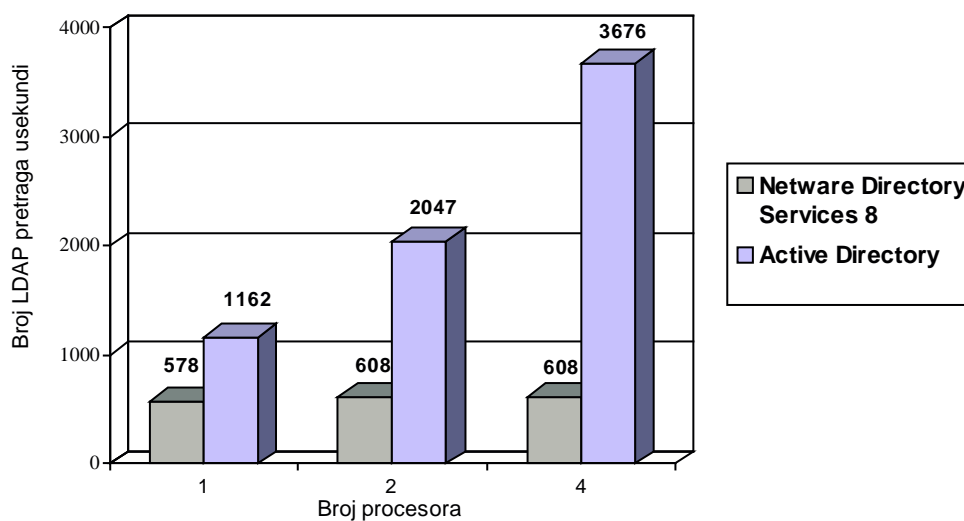
Active Directory (AD) je direktorij ugrađen u mrežni operacijski sustav Microsoft Windows 2000 Server [22]. Detaljan opis Active Directory-a dan je u narednom poglavlju. U ovom poglavlju navedeni su samo razlozi odabira upravo ovog direktorija.

Premda direktoriji drugih proizvođača teoretski omogućuju ugradnju direktorija na više različitih platformi, kao i suradnju s računalima zasnovanim na različitim platformama, to je u praksi teško ostvarivo.

Većina rješenja koja se koriste u izradi sustava zajedničkih funkcija zasnovana su na Microsoft-ovim tehnologijama. Da bi se izbjegli problemi koji često nastaju prilikom pokušaja suradnje tehnologija različitih proizvođača, poželjno je koristiti rješenja jednog proizvođača.

Programska rješenja opisana u ovom radu koriste ADSI programsko sučelje (eng. Application Programming Interface – API). ADSI, osim Active Directory-a podržava i Netware Directory Services. Ako se prilikom daljnjeg razvoja sustava pojavi potreba za proširenjem na druge platforme, za pristup direktoriju moguće je koristiti programska rješenja opisana u ovom radu, bez obzira radi li se o Active Directory-u ili Netware Directory Services.

Osnovni protokol koji se koristi za pristup direktoriju je LDAP. Active Directory je u potpunosti zasnovan na LDAP protokolu. AD po svojstvima daleko nadmašuje direktorije konkurentskih proizvođača. Na slici 3-6 prikazana je usporedba svojstava Active Directory-a i Netware Directory Services.



Slika 3-6 Usporedba svojstava Active Directory-a i Netware Directory Services

Mjerenja, čiji su rezultati prikazani slikom 3-6, izvedena su na sustavu s mikroprocesorima Intel Xeon 400 MHz, 4GB memorije, sa 1 milijun indeksiranih zapisa u direktoriju. Svi zapisi bili su smješteni u memoriju. Ovi rezultati mjerenja pokazuju da Active Directory može obraditi daleko veći broj LDAP upita u jedinici vremena nego Netware Directory Services. Dodavanjem novih procesora u sustav, broj izvedenih pretraga raste gotovo linearno za Active Directory, dok je utjecaj na Netware Directory Services gotovo zanemariv. Osim ovih, proveden je i niz mjerenja od strane nezavisnog laboratorija (KeyLabs) na različitim konfiguracijama sustava i različitim veličinama direktorija. Većina rezultata pokazuje da Active Directory postiže znatno bolja svojstva [19].

Dobra svojstva sustava, te olakšana suradnja sa ostalim komponentama sustava osnovni su razlozi za odabir Active Directory-a za spremište podataka sustava zajedničkih funkcija. Detaljan prikaz mogućnosti i načina rada Active Directory-a dan je u narednom poglavlju.

4 Active Directory

Active Directory je direktorij ugrađen u mrežni operacijski sustav Microsoft Windows 2000 Server [22]. Direktorij je zasnovan na LDAP protokolu, i sve operacije nad zapisima u direktoriju provode se korištenjem ovog protokola. Spremište podataka direktorija je hijerarhijski organizirana, raspodijeljena baza podataka. Active Directory shema definira vrste zapisa koji se mogu spremati u spremište podataka direktorija.

4.1 Integracija Active Directory-a u Windows 2000 Server

Podaci pohranjeni u Active Directory-u su raspoloživi korisnicima i administratorima. Samo jednim postupkom prijave autorizirani korisnici mogu pristupiti resursima bilo gdje na mreži. Administratorima na raspolaganju stoje alati za upravljanje cjelokupnom mrežom sa jednog mjesta.

AD je predviđen za spremanje podataka o infrastrukturi Windows 2000 mreže. Mreže, posebno one velikih multinacionalnih kompanija, mogu biti izuzetno složene i rasprostranjene širom svijeta. Da bi upravljanje mrežom bilo jednostavnije, Active Directory omogućuje podjelu mreže na logičke cjeline. Mreža može biti podijeljena na domene, organizacijske jedinice, grupe i polja. Domene se grupiraju u stabla i skupove domena.

U svome radu Active Directory se oslanja na postojeći DNS (eng. Domain Name System), sustav imenovanja računala. Active Directory i Windows 2000 server operacijski sustav podržavaju niz sigurnosnih mehanizama zaštite podataka od neovlaštenog korištenja.

4.1.1 Domain Name System (DNS)

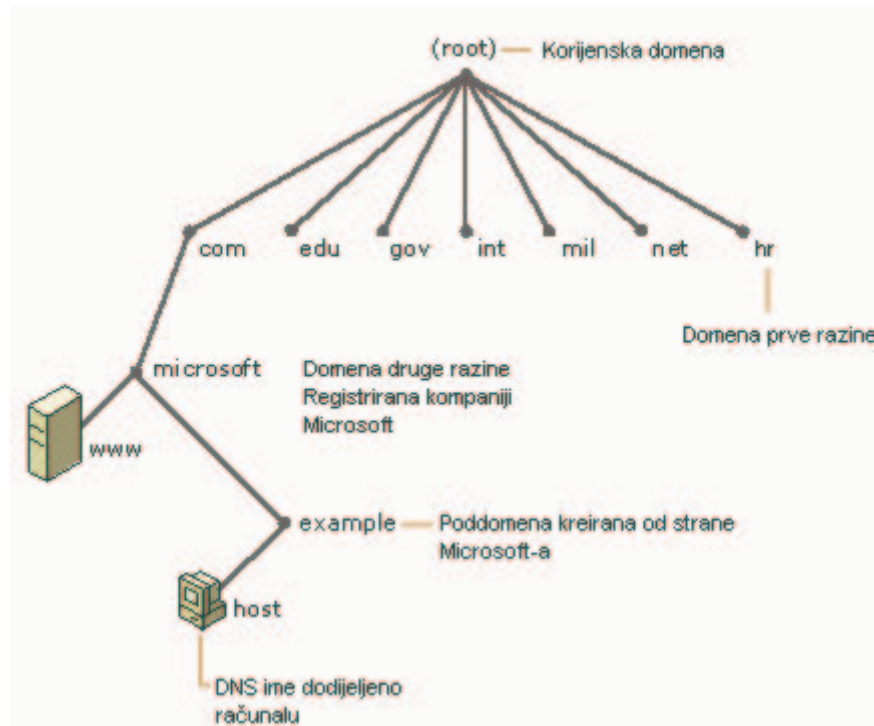
U svome radu Active Directory koristi postojeći DNS (eng. Domain Name System) sustav. DNS je sustav imenovanja računala na Internetu. Sustav je organiziran hijerarhijski kao skup domena.

Radi olakšanja administracije sustava mreža se dijeli na domene. Domena je skup računala u mreži koji imaju zajedničke sigurnosne postavke. Svaka domena ima definirane sigurnosne postavke za pristup resursima unutar domene, kao i postavke koje se koriste prilikom komunikacije s drugim domenama. Takva organizacija mreže ima niz prednosti:

- Mjere sigurnosti se definiraju za svaku domenu zasebno i ne prelaze granice domena.
- Prava za administraciju se dodjeljuju na razini domena čime se izbjegava postojanje administratora s pravima nad cijelom mrežom.
- Domene omogućuju da struktura mreže odgovara organizacijskoj strukturi kompanije.

- Svaka domena sadrži informacije samo o objektima koji sačinjavaju tu domenu. Takvom podjelom direktorija omogućena je njegova skalabilnost, odnosno mogućnost da se direktorij širi na veliki broj objekata.

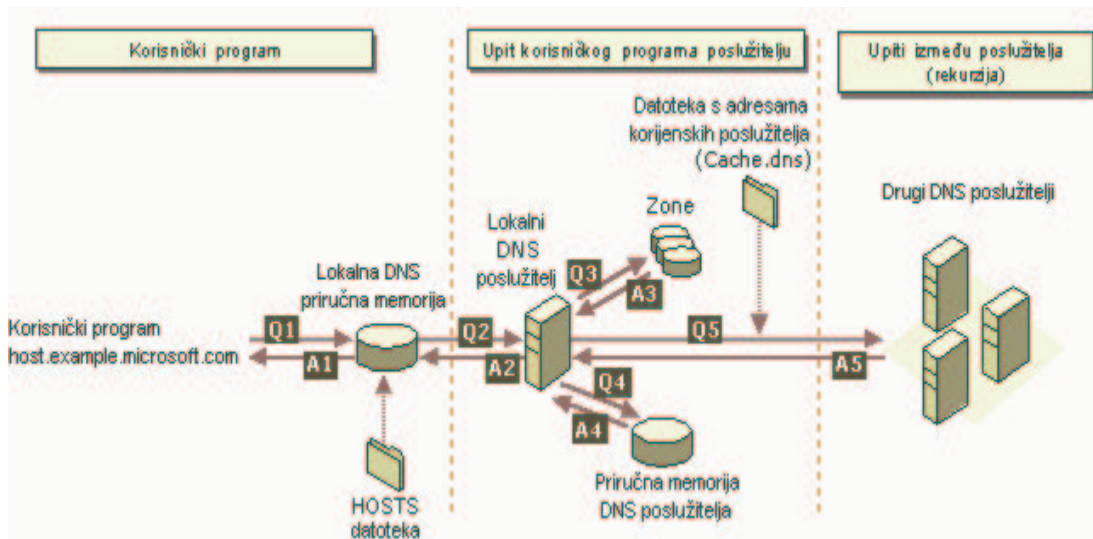
Svaka domena, kao i svako računalo unutar domene ima svoje ime. Imena računala se koriste jer je ljudima lakše pamtiiti ime računala nego IP adresu. Imena računala su hijerarhijski organizirana u strukturu stabla koja se naziva imenik domena (eng. DNS domain namespace). Primjer imenika domena prikazan je na slici 4-1.



Slika 4-1 Primjer imenika domena (eng. DNS domain namespace)

Stablo na slici promatra se po razinama. Korijenska razina se označava s dvama navodnicima (""). Domena pridružena korijenskoj razini označava se točkom (.). Ime domene prve razine sastoji se od dva ili tri slova koja označavaju tip organizacije ili zemlju koja koristi domenu (npr. com za komercijalnu upotrebu, ili hr za Hrvatsku). Domene druge razine imenuju se prema pojedincima ili organizacijama koji su ih registrirali za upotrebu na Internetu (npr. microsoft). Organizacije mogu, unutar svoje domene stvoriti poddomene za upotrebu unutar vlastite organizacije (npr. example). Na dnu stabla nalaze se resursi unutar domene. Najčešći resursi su računala (npr. host). Puno ime domene ili resursa (eng. Fully Qualified Domain Name - FQDN) dobije se obilaskom stabla od dna prema vrhu, i odvajanjem imena običenih čvorova točkom. Puno ime računala na slici je host.example.microsoft.com. Puno ime domene ili resursa mora biti jedinstveno u mreži, kako ne bi došlo do konflikta zbog nejednoznačnosti.

Računalo može komunicirati s drugim računalom samo ako poznaje njegovu IP adresu. DNS sustav omogućuje da se na osnovu imena računala odredi njegova IP adresa. Rad sustava zasniva se na skupu računala, DNS poslužitelja, koji međusobno razmjenjuju informacije o IP adresama računala u mreži, i pružaju tu informaciju korisničkim programima. Primjer obrade DNS upita dan je na slici 4-2.



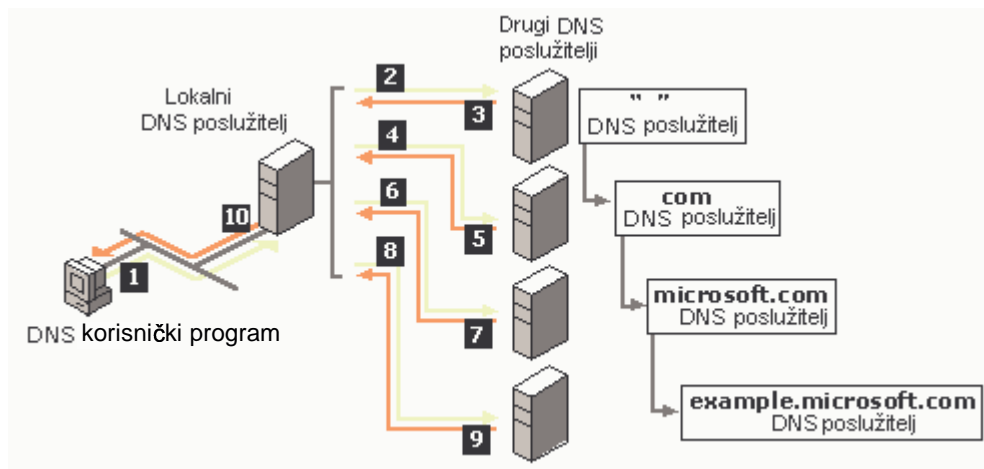
Slika 4-2 Obrada DNS upita

Kada korisnički program želi saznati IP adresu računala host.example.microsoft.com, prvo provjerava lokalnu DNS priručnu memoriju (Q1). U lokalnoj priručnoj memoriji se određeno vrijeme čuvaju rezultati prethodnih upita, kao i sadržaj HOSTS datoteke. HOSTS datoteka služi za unazadnu kompatibilnost, i u njoj se navode IP adrese računala za koja se sigurno zna da imaju odgovarajuću IP adresu. Ako se tražena IP adresa ne nalazi u lokalnoj priručnoj memoriji, upit se proslijeđuje lokalnom DNS poslužitelju koji je definiran u konfiguraciji sustava (Q2). DNS poslužitelj prvo provjerava da li tražena IP adresa postoji među zapisima o zoni kojoj pripada (Q3).

Zona je skup računala i domena za koje se podaci o IP adresama održavaju zajedno. Za održavanje podataka o jednoj zoni zadužen je jedan ili više DNS poslužitelja. Svi DNS poslužitelji u jednoj zoni sadrže iste podatke, a usklađivanje se izvodi uvišestručavanjem.

Ako se tražena IP adresa ne nalazi među zapisima o zoni kojoj DNS poslužitelj pripada, obavlja se provjera priručne memorije DNS poslužitelja (Q4). Priručna memorija DNS poslužitelja sadrži rezultate prethodnih upita od drugih DNS poslužitelja. Rezultati prethodnih upita se u priručnoj memoriji čuvaju unaprijed određeno vrijeme. Ako tražena IP adresa ne postoji u priručnoj memoriji DNS poslužitelja, DNS poslužitelj započinje rekurzivni postupak za dobavljanje tražene IP adrese (Q5). Rekurzivni postupak započinje slanjem upita jednom od korijenskih poslužitelja. Da bi to bilo moguće, svaki DNS poslužitelj sadrži datoteku s

popisom korijenskih poslužitelja. Rekurzivni postupak za dobavljanje IP adrese prikazan je na slici 4-3.



Slika 4-3 Rekurzivni postupak dobavljanja IP adrese

Nakon što je lokalni DNS poslužitelj zaprimio zahtjev za IP adresom računala `host.example.microsoft.com` (1), i ustanovio da tražena informacija ne postoji među podacima o lokalnoj zoni, niti u priručnoj memoriji, započinje rekurzivni postupak. Korijenskom DNS poslužitelju šalje se zahtjev za dobavljanjem adrese DNS poslužitelja `com` domene (2). Po dobivanju odgovora (3), dobivena IP adresa koristi se za slanje zahtjeva DNS poslužitelju `com` domene za dobavljanjem adrese `microsoft.com` domene (4). Postupak se rekurzivno ponavlja (5, 6, 7, 8), dok se od DNS poslužitelja domene `example.microsoft.com` ne dobije IP adresa računala `host.example.microsoft.com` (9). Dobiveni odgovor vraća se konačno korisničkom programu (10).

Dobiveni odgovor je tražena IP adresa, ili negativan odgovor, u slučaju da računalo s traženim imenom ne postoji. Ako lokalni DNS poslužitelj ne podržava rekurzivni postupak traženja IP adrese, dobiveni odgovor može biti referenca na drugi DNS poslužitelj. Referenca omogućuje korisničkom programu slanje daljnjih upita, kako bi dobio traženu adresu. Ovakav postupak traženja IP adrese naziva se iterativni postupak. DNS poslužitelji mogu, osim same IP adrese računala sadržavati i neke druge informacije o resursima na mreži (eng. resource records).

Active Directory i DNS sustav imaju istu hijerarhijsku strukturu. Pojam domene koji se koristi kada se govori o Active Directory-u odgovara istom pojmu kod DNS sustava.

Active Directory korisnički programi koriste DNS za pristup direktoriju. Pristup direktoriju ostvaruje se pristupom jednom od Active Directory domenskih zastupnika (eng. domain controller - DC).

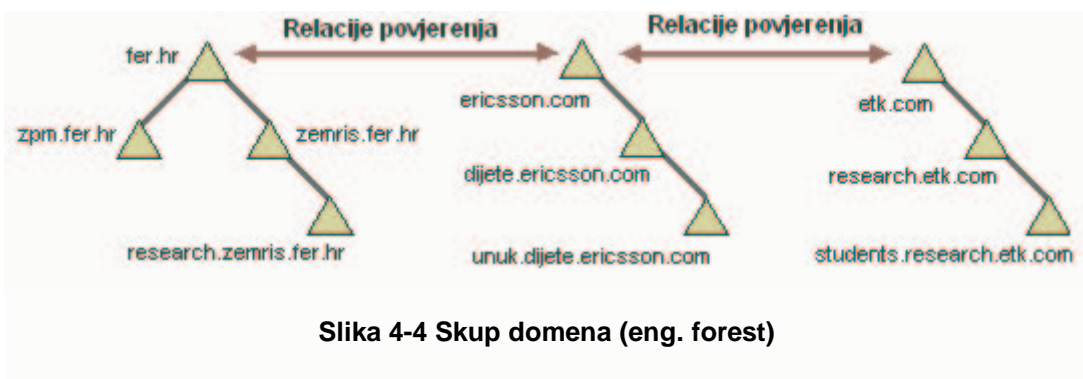
Domenski zastupnik je računalo koje sadrži dio podataka direktorija, i ima ulogu sustavske komponente direktorija (DSA). U jednoj domeni može postojati više domenskih

zastupnika. Svi domenski zastupnici u jednoj domeni sadrže iste podatke, a usklađivanje se izvodi postupkom uvišestručavanja.

Za pristup direktoriju potrebno je poznavati adresu barem jednog domenskog zastupnika. Adresa domenskog zastupnika dobiva se slanjem odgovarajućeg upita DNS poslužitelju. Odgovor na zahtjev je lista imena domenskih zastupnika i njihovih IP adresa. Korisnički program potom šalje zahtjev za spajanje svakom domenskom zastupniku. Za postupak prijave koristi se domenski zastupnik koji prvi odgovori na zahtjev.

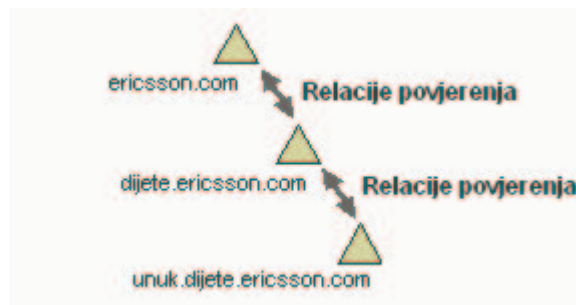
4.1.2 Stabla i skupovi domena

Jedna ili više domena koje dijele zajedničku shemu, globalni katalog i postavke za uvišestručavanje nazivaju se zajedničkim imenom skup domena (eng. forest). Globalni katalog (eng. global catalog) je domenski zastupnik koji sadrži djelomičnu presliku svih podataka u direktoriju. Skup domena zapravo predstavlja cjelokupni direktorij. Primjer skupa domena prikazan je na slici 4-4.



Slika 4-4 Skup domena (eng. forest)

Ako nekoliko domena iz skupa dijele zajednički imenik (eng. contiguous namespace) takva struktura naziva se stablo domena (eng. domain tree). Primjer stabla domena prikazan je na slici 4-5.



Slika 4-5 Stablo domena (eng. domain tree)

Domene dijele zajednički imenik domena, ako se puno ime domene dobije dodavanjem imena domene imenu nadređene domene. Relacija između takvih domena naziva se relacija

roditelj – dijete. Skup domena sastoji se od jednog ili više stabala domena. Domena koja je prva nastala u skupu domena naziva se korijenskom domenom skupa (eng. forest root domain). Domena koja je prva nastala u stablu domena naziva se korijenska domena stabla (eng. tree root domain).

Strelice na slikama 4-4 i 4-5 označavaju relacije povjerenja među domenama. Relacije povjerenja su logičke relacije koje omogućuju autentikaciju korisnika između dvije domene. Korisnički računi i prava se definiraju unutar granica domene. Ako su dvije domene uspostavile relacije povjerenja, tada svaka domena može autenticirati i korisnike iz druge domene. Isto tako, svaka domena može definirati prava pristupa resursima za korisnike i grupe iz druge domene. Relacije povjerenja su simetrične i tranzitivne.

Relacije povjerenja se automatski stvaraju između domena za koje vrijedi odnos dijete – roditelj, prilikom dodavanja nove domene stablu. U skupu domena relacije povjerenja se automatski stvaraju između korijenske domene skupa, i korijenske domene svakog stabla koje se dodaje skupu. Opisana organizacija relacija povjerenja omogućuje autenticiranje korisnika svim domenama u skupu, s obzirom da su sve relacije povjerenja simetrične i tranzitivne. Osim kod prve prijave na sustav (eng. logon), autentikacija se obavlja automatski, transparentno za korisnika. Premda se korisnik može autenticirati bilo kojoj domeni u skupu, ne znači da ima prava pristupa resursima u svim domenama. Prava pristupa resursima definiraju se na razini pojedine domene.

Sve domene u svim stablima u skupu domena dijele sljedeća svojstva:

- Tranzitivne relacije povjerenja između domena u stablu.
- Tranzitivne relacije povjerenja između korijenske domene skupa i korijenskih domena svih stabala u skupu.
- Zajedničku shemu.
- Zajedničke postavke za uvišestručavanje.
- Zajednički globalni katalog.

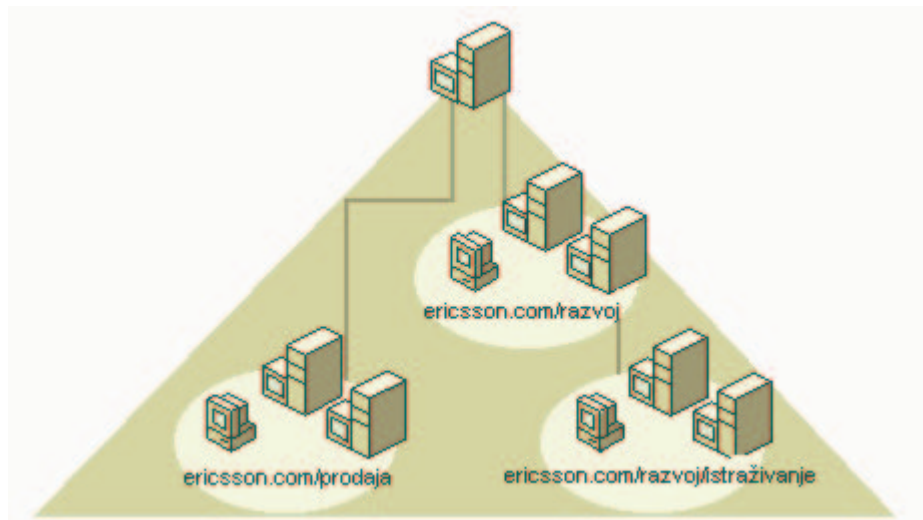
Mogućnost korištenja stabala i skupova domena omogućuje korištenje zajedničkog, kao i odvojenog imenika (eng. contiguous and noncontiguous namespace). Druga mogućnost je posebno korisna za velike organizacije s odjelima koji svaki za sebe upravlja DNS imenima.

4.1.3 Organizacijske jedinice

Organizacijske jedinice su posebne vrste spremnika. Spremnici (eng. containers) su zapisi u direktoriju koji mogu sadržavati druge zapise. Organizacijske jedinice mogu sadržavati zapise o korisnicima, grupama, računalima i drugim organizacijskim jedinicama.

Organizacijske jedinice mogu sadržavati samo objekte iz domene kojoj pripadaju. Organizacijske jedinice koriste se za preslikavanje strukture organizacije u direktorij. Podjela na organizacijske jedinice omogućuje upravljanje podacima u direktoriju zasnovano na

organizacijskom modelu dotične organizacije. Primjer podjele domene na organizacijske jedinice prikazan je na slici 4-6.



Slika 4-6 Podjela domene na organizacijske jedinice

Hijerarhija organizacijskih jedinica može se širiti po volji. Korištenjem organizacijskih jedinica smanjuje se broj potrebnih domena.

Organizacijske jedinice su najmanje jedinice nad kojima se mogu definirati grupna pravila (eng. Group Policy) i dodjeljivati administratorska prava.

Grupna pravila definiraju različite korisničke postavke kao što su programi koje korisnici smiju pokretati, programi koji su raspoloživi na Start izborniku, izgled okružja i sl.

Pojedinim korisnicima mogu se dodijeliti administratorska prava nad pojedinim organizacijskim jedinicama. Ako se korisniku dodijele administratorska prava nad nekom organizacijskom jedinicom, on može, ali i ne mora imati administratorska prava nad organizacijskim jedinicama koje dana organizacijska jedinica sadrži. Dodjela administratorskih prava nad organizacijskim jedinicama omogućuje veliku fleksibilnost pri podjeli odgovornosti za održavanje mreže, čime se izbjegava postojanje administratora sa ovlastima koje su veće nego što je potrebno.

4.1.4 Grupe

Grupe su posebna vrsta zapisa koji omogućuju grupiranje zapisa o korisnicima, suradnicima (eng. contacts), računalima i drugim grupama. Zapisi koji su grupirani u pojedinu grupu nazivaju se članovima te grupe.

Grupe se dijele prema namjeni na grupe za definiranje sigurnosnih postavki (eng. security groups) i distribucijske grupe (eng. distribution groups). Grupe za definiranje sigurnosnih postavki olakšavaju dodjelu prava za pristup resursima na mreži. Umjesto da se prava dodjeljuju svakom korisniku zasebno, prava se dodijele grupi za definiranje sigurnosnih postavki. Time svaki korisnik koji je član grupe dobiva sva dodijeljena prava

pristupa. Distribucijske grupe omogućuju stvaranje lista za slanje elektronske pošte. Programi za slanje elektronske pošte koriste distribucijske grupe za slanje poruka određenoj skupini korisnika.

Grupe se dijele prema dosegu definicije na univerzalne grupe (eng. universal groups), globalne grupe (eng. global groups) i lokalne grupe (eng. domain local groups). Doseg definicije grupe određuje dijelove skupa domena na koje se odnosi definicija grupe. Univerzalne grupe mogu imati članove iz bilo koje domene. Univerzalnim grupama se mogu dodjeljivati prava pristupa svim resursima u svim domenama u skupu. Globalne grupe mogu imati članove samo iz domene u kojoj je grupa definirana, dok im se prava pristupa mogu dodjeljivati za sve resurse u svim domenama u skupu. Lokalne grupe mogu imati članove samo iz domene u kojoj je grupa definirana, te dodijeljena prava pristupa samo resursima iz iste domene. Doseg definicije bilo koje grupe ne može prelaziti granice skupa domena.

Operacijski sustav Windows 2000 Server ima predefiniran skup grupa koje pružaju osnovne funkcionalnosti potrebne za upravljanje operacijskim sustavom i mrežom. Primjer takve grupe je grupa čiji su članovi administratori (eng. Administrators), ili grupa koja označava sve korisnike (eng. Everyone).

4.1.5 Polja

Polje (eng. site) je definirano kao skup računala u jednoj ili više IP podmreža (eng. IP subnets). Računala se grupiraju u polje ako su međusobno dobro povezana. Dobra povezanost je relativan pojam, i ovisi o konkretnoj primjeni i željenim svojstvima. Pod pojmom dobre povezanosti obično se podrazumijeva brza i sigurna veza. Polje najčešće obuhvaća jednu podmrežu, no to ne mora biti slučaj. Polje može obuhvaćati samo dio podmreže, kao i više podmreža, što ovisi isključivo o povezanosti među računalima.

Polja definiraju fizičku strukturu mreže, isto kao što domene definiraju logičku strukturu organizacije. Polja i domene su međusobno neovisne. Polje može obuhvaćati više domena, i domena može obuhvaćati više polja.

Active Directory koristi podjelu mreže u polja za optimiranje svojstava pri korištenju mrežnih resursa. Zahtjevi korisničkog programa prosljeđuju se domenskom zastupniku koji se nalazi u istom polju kao i računalo s kojeg je zahtjev stigao, čime se minimizira vrijeme odziva. Uvišestručavanje podataka među domenskim zastupnicima koji se nalaze unutar istog polja se izvodi češće nego među domenskim zastupnicima koji se nalaze u različitim poljima, čime se značajno smanjuje mrežni promet.

4.1.6 Zaštita podataka od neovlaštenog korištenja

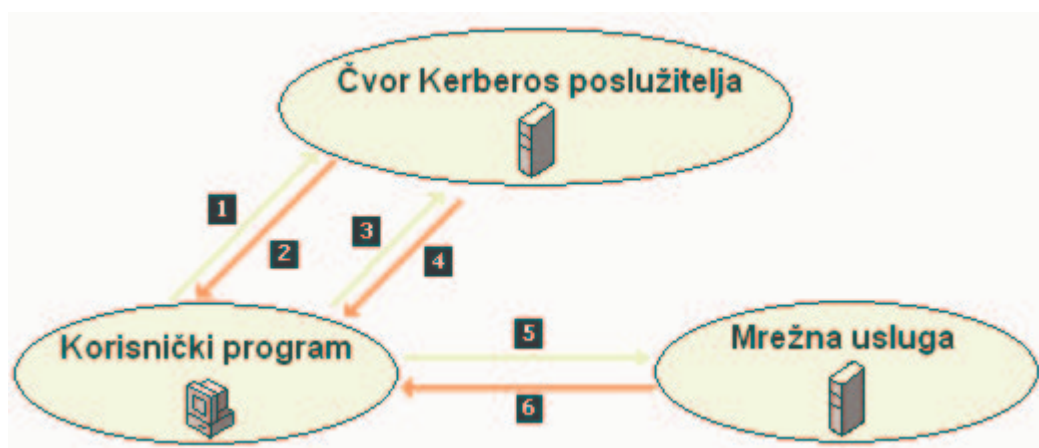
Zaštita podataka od neovlaštenog korištenja u Windows 2000 Server operacijskom sustavu zasnovana je na tri osnovna sigurnosna mehanizma: autentikaciji (eng.

authentication), autorizaciji (eng. authorization) i zaštiti mrežnog prometa. U ovome radu opisani su sigurnosni mehanizmi vezani uz zaštitu podataka u Active Directory-u.

Postoje dvije vrste autentikacije korisnika: izravna autentikacija (eng. interactive logon) i mrežna autentikacija (eng. network authentication). Izravna autentikacija omogućuje korisniku dokazivanje identiteta računalu ili domeni na koju se prijavljuje. Mrežnom autentikacijom korisnik dokazuje svoj identitet mrežnoj usluzi kojoj pristupa. Windows 2000 Server podržava tri autentikacijska protokola: Kerberos V5, SSL/TLS (eng. Secure Sockets Layer/Transport Layer Security) i NTLM autentikaciju. Mrežna autentikacija obavlja se bez korisnikova znanja, čime je omogućeno korištenje svih resursa u domeni samo jednim postupkom prijave na sustav.

Kerberos V5 autentikacijski protokol

Kerberos V5 je primarni autentikacijski protokol u Windows 2000 Server operacijskom sustavu. Protokol omogućuje autentikaciju korisnika, ali i mrežne usluge. Ovakva autentikacija naziva se uzajamna autentikacija (eng. mutual authentication). Kerberos V5 autentikacijski protokol zasnovan je na dodjeli ulaznica (eng. tickets) za korištenje mrežnih usluga. Ulaznice dodjeljuje čvor Kerberos poslužitelja (eng. Key Distribution Center – KDC). Čvor kerberos poslužitelja je usluga koju pruža svaki domenski zastupnik, kao dio Active Directory-a. Čvor Kerberos poslužitelja čuva podatke potrebne za autentikaciju korisnika i mrežnih usluga. Podaci koji se razmjenjuju između korisničkog programa, čvora Kerberos poslužitelja i mrežne usluge kriptirani su simetričnim ključevima. Ključ se naziva simetričnim, ako se informacija kriptirana nekim ključem dekriptira korištenjem istog ključa. Postupak autentikacije korištenjem Kerberos V5 protokola prikazan je na slici 4-7.



Slika 4-7 Autentikacija korištenjem Kerberos V5 protokola

Autentikacija se obavlja u četiri koraka:

1. Korisnik se pomoću lozinke autenticira čvoru Kerberos poslužitelja (1).
2. Čvor Kerberos poslužitelja izdaje posebnu TGT (eng. Ticket-Granting Ticket) ulaznicu koja omogućuje pristup TGS (Ticket-Granting Service) usluzi čvora Kerberos poslužitelja (2).
3. Kada korisnički program želi koristiti neku mrežnu uslugu, pristupa TGS usluzi čvora Kerberos poslužitelja (3), koja mu izdaje ulaznicu za korištenje te mrežne usluge (eng. service ticket) (4).
4. Korisnički program šalje dobivenu ulaznicu za korištenje usluge dotičnoj usluzi (5). Ulaznica za korištenje usluge služi kao dokaz identiteta korisnika, kao i mrežne usluge. Mrežna usluga šalje potvrdni odgovor korisničkom programu (6), čime je postupak autentikacije završen.

Kerberos V5 autentikacijski protokol izgrađen je tako da korisnikova lozinka nikada ne putuje mrežom, čime je znatno povećana sigurnost sustava.

SSL/TLS autentikacija

SSL/TLS autentikacija se koristi prilikom pristupa Internet poslužiteljima. SSL/TLS autentikacija je zasnovana na asimetričnom kriptosustavu. Asimetrični kriptosustav (eng. asymmetric cryptography) koristi par ključeva: tajni i javni ključ. Podaci koji su kriptirani jednim od ključeva dekriptiraju se korištenjem drugog ključa. U stvarnom vremenu nije moguće iz jednog ključa odrediti drugi, pa se javni ključ obično javno obznanjuje. Asimetrični kriptosustav naziva se još i sustav s javnim ključem (eng. public key cryptography). Povezivanje javnog ključa s identitetom korisnika, odnosno usluge, omogućuju certifikati (eng. certificate). Certifikat je dokument u elektroničkom obliku koji sadrži informacije o korisniku ili usluzi, te njegov javni ključ. Certifikat je ovjeren od strane izdavača certifikata (eng. Certification Authority – CA) korištenjem digitalnog potpisa. Digitalni potpis (eng. digital signature) je kriptografska tehnika kojom se jamči da dokument nije mijenjan nakon izdavanja, te da ga je izdao odgovarajući izdavač. Nedostatak upotrebe certifikata je što obje strane u komunikaciji moraju vjerovati izdavaču certifikata, što nije uvijek slučaj.

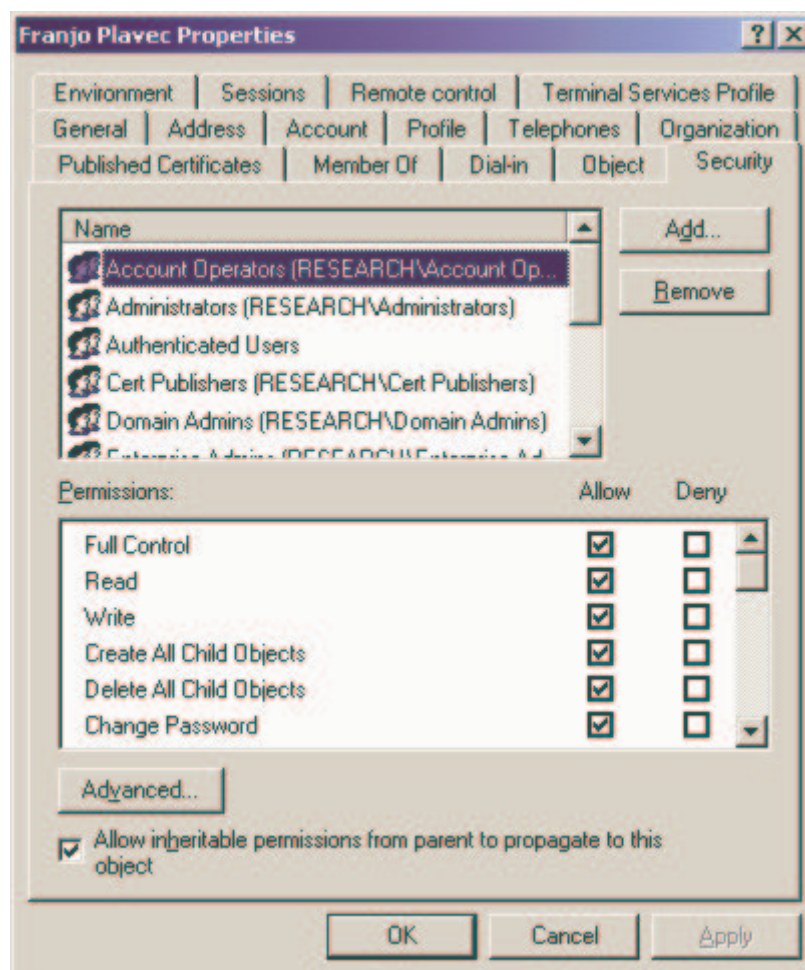
NTLM autentikacija

NTLM autentikacija koristi se pri komunikaciji s korisničkim računalima zasnovanima na operacijskom sustavu Windows NT 4.0. NTLM se koristi i za autentikaciju korisničkih računala zasnovanih na Windows 2000 operacijskom sustavu koji nisu članovi domene.

Sigurnosni opisnici

Autorizacija omogućuje kontrolu pristupa podacima dodjelom prava pristupa pojedinim korisnicima i grupama. Kontrola pristupa izvodi se korištenjem sigurnosnih opisnika (eng.

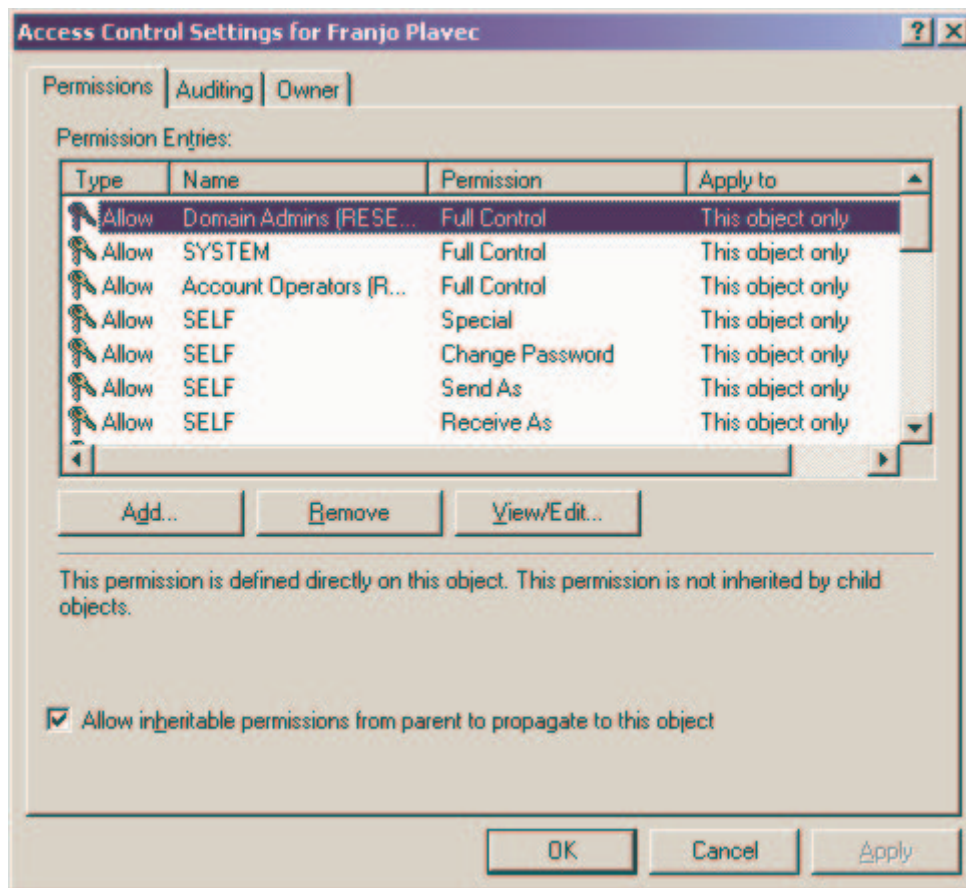
security descriptors) pridruženih zapisima u direktoriju. Sigurnosni opisnik sastoji se od dva dijela: DACL (eng. discretionary access control list) i SACL (eng. system access control list). DACL sadrži popis svih korisnika i grupa koja imaju prava pristupa zapisu, zajedno sa vrstama prava pristupa (npr. čitanje, pisanje,...). SACL sadrži popis aktivnosti nad zapisom koje je potrebno pratiti (eng. auditing) za određene korisnike i grupe. Moguće je pratiti različite aktivnosti nad zapisom, kao što su čitanje ili izmjena zapisa. Svaka definirana aktivnost bilježi se u posebnu datoteku, koju je kasnije moguće pregledavati i utvrditi tko je, i na koji način koristio zapis. Prava pristupa, i praćenje aktivnosti moguće je definirati nad samim zapisom, kao i nad pojedinim atributima svakog zapisa. Prava pristupa definirana pomoću sigurnosnih opisnika moguće je nasljeđivati. Nasljeđivanje omogućuje da svi objekti u nekom spremniku imaju iste sigurnosne postavke kao i sam spremnik. Primjer definiranja prava pristupa zapisu koji opisuje korisnika prikazan je na slici 4-8.



Slika 4-8 Prava pristupa zapisu koji opisuje korisnika

U gornjem dijelu prozora definira se korisnik na kojega se prava pristupa odnose. U donjem dijelu prozora definiraju se sama prava pristupa. Svako pravo pristupa moguće je dozvoliti (eng. allow), ne dozvoliti, zabraniti (eng. deny) ili ne zabraniti. Ako neko pravo nije niti

dozvoljeno niti zabranjeno, primjenjuje se pravo pristupa naslijeđeno od nadređenog spremnika. Prava koja se dodjeljuju izravno objektu nadjačavaju (eng. override) naslijeđena prava. Naslijeđivanje prava pristupa moguće je spriječiti isključivanjem odgovarajuće opcije na dnu prozora (eng. Allow inheritable permissions from parent to propagate to this object). Napredna prava pristupa moguće je definirati pritiskom na tipku Advanced. Primjer naprednog definiranja prava pristupa prikazan je na slici 4-9.



Slika 4-9 Napredno definiranje prava pristupa zapisu

Prozor za napredno definiranje prava pristupa sadrži tri jahača. Prvi jahač omogućuje definiranje prava pristupa. Moguće je definirati prava pristupa za sam zapis, kao i za svaki pojedinačni atribut. Drugi jahač omogućuje definiranje aktivnosti koje je potrebno pratiti. Aktivnosti je moguće definirati nad samim zapisom, kao i nad svakim pojedinačnim atributom. Treći jahač daje uvid u vlasništvo nad zapisom (eng. ownership). Svaki zapis u direktoriju ima svoga vlasnika (eng. owner). Inicijalno je vlasnik zapisa korisnik koji je stvorio zapis u direktoriju. Vlasnik zapisa ima pravo dodjele prava ostalim korisnicima i grupama. Vlasnik zapisa može dodijeliti pravo preuzimanja vlasništva (eng. Take Ownership permission) drugim korisnicima. Administrator može preuzeti vlasništvo nad bilo kojim zapisom u dijelu direktoriju nad kojim ima administratorska prava. Administrator ne može

prenijeti vlasništvo na nekog drugog korisnika, već samo na sebe. Time je osigurano da administrator ne može preuzeti vlasništvo nad zapisom, obaviti izmjene i vratiti vlasništvo korisniku. Ako administrator preuzme vlasništvo nad objektom, to je jasno vidljivo, i ako postoji sumnja da je administrator to učinio neovlašteno, može biti pozvan na odgovornost.

Zaštita mrežnog prometa

Zaštita mrežnog prometa izvodi se zaštitom samih podataka koji putuju mrežom (eng. on the wire) ili zaštitom mrežnog sučelja (eng. network interface). Zaštita podataka koji putuju mrežom izvodi se korištenjem IPSec (eng. Internet Protocol Security) sustava ili usluge usmjeravanja (eng. Router service). Zaštita mrežnog sučelja izvodi se pomoću lokalnog zastupnika (eng. proxy server).

IPSec je skup otvorenih standarda koji osiguravaju tajnost podataka (eng. confidentiality), autentikaciju i zaštitu vjerodostojnosti podataka (eng. integrity). Tajnost podataka osigurana je kriptiranjem podataka. Podaci koji putuju mrežom ne mogu se pročitati bez poznavanja ključa kriptiranja (eng. encryption key). Autentikacija i zaštita vjerodostojnosti podataka omogućena je korištenjem digitalnog potpisa. Digitalni potpis sastoji se od sažetka poruke (eng. message digest) kriptiranog tajnim ključem pošiljatelja. Sažetak poruke se dobiva korištenjem funkcije za sažimanje (eng. digest function). Funkcija za sažimanje je odabrana tako da se iz sažetka poruke ne može odrediti originalna poruka. Vjerojatnost da se za dvije različite poruke dobije isti sažetak je zanemarivo mala. Digitalni potpis šalje se zajedno s porukom. Primatelj dekriptira sažetak, i uspoređuje ga sa sažetkom kojega sam stvara iz originalne poruke. Ako su sažeci jednaki, primatelj je siguran da poruka nije mijenjana, i da je poslana od strane pošiljaoca čijim tajnim ključem je generiran digitalni potpis.

Usluga usmjeravanja pruža usluge usmjeravanja paketa za LAN (eng. Local Area Network), WAN (eng. Wide Area Network) i Internet korištenjem VPN (eng. Virtual Private Network) sigurne veze. VPN omogućuje emuliranje veze od točke do točke (eng. point-to-point) korištenjem Interneta. Podaci se prije slanja u globalnu mrežu Internet kriptiraju.

Lokalni zastupnik djeluje kao sigurnosna zaštitna stijena (eng. firewall) između lokalne mreže i Interneta. Lokalni zastupnik zaustavlja sve potencijalne spojeve (eng. connections) od Interneta prema lokalnoj mreži, te ograničava spojeve iz lokalne mreže prema Internetu prema zadanim kriterijima. Kriteriji za ograničavanje mogu se odnositi na korisnika, protokol, TCP/IP port, trenutno vrijeme ili odredišnu IP adresu. Sprečavanjem i ograničavanjem određenih vrsta prometa podataka uvelike se smanjuju sigurnosni rizici pri komunikaciji preko globalne mreže Internet.

4.2 Spremište podataka direktorija

Podaci u Active Directory-u spremljeni su na domenskim zastupnicima. Svaki domenski zastupnik u domeni sadrži potpunu presliku svih podataka domene. Podaci su na domenskom zastupniku spremljeni u datoteci Ntds.dit. Globalni katalog je domenski zastupnik koji, osim preslike podataka domene, sadrži djelomičnu presliku podataka svih domena u skupu.

Domenski zastupnici sadrže tri vrste podataka: podatke za domenu (eng. domain data), shemu (eng. schema data) i podatke o konfiguraciji sustava (eng. configuration data).

Podaci za domenu su zapisi koje u Active Directory dodaju korisnici i administratori. Primjeri takvih zapisa su zapisi o korisnicima, računalima, pisačima i sl.

Shema sadrži formalnu definiciju formata zapisa i njihovih atributa. Windows 2000 Server ima predefinirane mnoge vrste zapisa potrebne za opisivanje resursa na mreži. Po potrebi, moguće je proširiti postojeću shemu definicijama novih razreda i atributa. Prava pristupa shemi osiguravaju shemu od neovlaštenog korištenja.

Podaci o konfiguraciji sustava opisuju topologiju direktorija. Ti podaci uključuju podatke o svim domenama, stablima i skupovima domena, kao i podacima o položaju domenskih zastupnika i globalnih kataloga unutar mreže.

Isti podaci se u direktoriju čuvaju na više domenskih zastupnika. Uvišestručavanje je postupak usklađivanja podataka među domenskim zastupnicima u domeni. Većina operacija nad direktorijem može se obavljati na svakom domenskom zastupniku. Uvišestručavanje obavlja prijenos izmijenjenih podataka drugim domenskim zastupnicima. Posebne operacije nad podacima u direktoriju mogu se provoditi samo na jednom domenskom zastupniku; nositelju operacije.

4.2.1 Globalni katalog

Globalni katalog (eng. global catalog) je domenski zastupnik koji sadrži potpunu presliku svih zapisa domene kojoj sam pripada i djelomičnu presliku zapisa iz drugih domena u skupu. Preslika je djelomična jer sadrži samo neke vrijednosti atributa zapisa. Atributi čije vrijednosti će biti spremljene u globalni katalog definirani su u shemi.

Globalni katalog omogućuje prijavu korisnika na domenu pružanjem informacije o pripadnosti univerzalnim grupama. Za postupak prijave se koristi domenski zastupnik koji se nalazi u istom polju kao i računalo na kojem se izvodi prijava. U svakom polju mora postojati barem jedan globalni katalog koji sadrži informacije o pripadnosti univerzalnim grupama, kako bi bio moguć postupak prijave korisnika na domenu.

Globalni katalog omogućuje pronalazak određenog zapisa bez obzira na to gdje se zapis nalazi u skupu. Tipično se u globalni katalog spremaju vrijednosti atributa koji su često kriterij pretrage. Time se osigurava velika brzina pretrage, i minimizira mrežni promet kao posljedica pretraživanja direktorija.

Globalni katalog se inicijalno automatski stvara na prvom domenskom zastupniku u skupu, premda dodavanjem novih domenskih zastupnika tu ulogu može preuzeti i neki drugi domenski zastupnik. Ovisno o željenim svojstvima sustava u skupu može postojati i više od jednog globalnog kataloga.

4.2.2 Uvišestručavanje podataka

Da bi se povećala pouzdanost i poboljšala svojstva sustava, isti podaci se čuvaju na više domenskih zastupnika. Izmjene nad podacima mogu se izvoditi nad bilo kojim domenskim zastupnikom. Usklađivanje podataka među domenskim zastupnicima izvodi se uvišestručavanjem. Da bi se smanjio mrežni promet prilikom uvišestručavanja među domenskim zastupnicima, preslikavaju se samo podaci koji su izmijenjeni od posljednjeg uvišestručavanja. Uvišestručavanje se obavlja različitom učestalošću, ovisno o kvaliteti veze među domenskim zastupnicima. Uvišestručavanje se obavlja češće među domenskim zastupnicima u istom polju, nego među domenskim zastupnicima u različitim poljima.

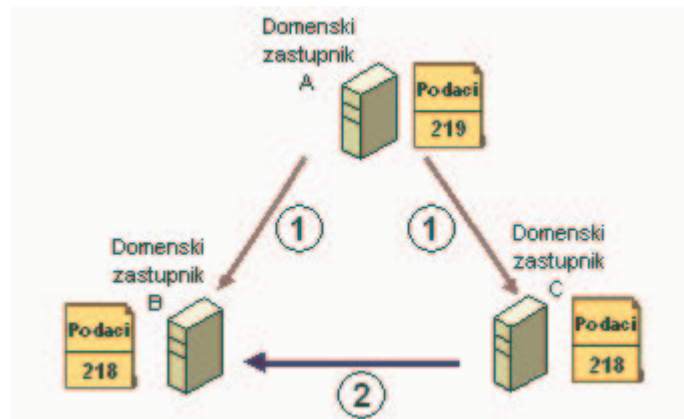
Uvišestručavanje podataka unutar polja izvodi se automatski. Active Directory automatski stvara topologiju uvišestručavanja (eng. replication topology). Topologija uvišestručavanja je konfiguracija mrežnih veza koje se koriste za uvišestručavanje podataka među domenskim zastupnicima. Topologija se stvara tako da je svaki domenski zastupnik povezan s ostatkom mreže preko najmanje dvije mrežne veze, ako je to moguće. Ako neki domenski zastupnik zbog kvara postane nedostupan, za uvišestručavanje se koristi druga mrežna veza koja ne uključuje dotični domenski zastupnik. Topologija uvišestručavanja se automatski prilagođuje promjenama u mreži.

Za razliku od računala u polju, veze među poljima se ne stvaraju automatski, već je potrebno specificirati koje mrežne veze će biti korištene za uvišestručavanje. Prilikom definiranja veze među poljima specificira se vrijeme kada je veza dostupna za uvišestručavanje, učestalost uvišestručavanja, te relativna cijena veze (eng. cost). Zadani parametri omogućuju optimalno korištenje mrežnih veza za uvišestručavanje. Ako se veze među poljima ne definiraju, uvišestručavanje se izvodi samo među domenskim zastupnicima unutar polja. Dodatno rasterećenje mreže postiže se kompresijom podataka prilikom uvišestručavanja među domenskim zastupnicima u različitim poljima.

Svaki domenski zastupnik bilježi broj izmjena koje je izveo nad lokalnom preslikom zapisa, kao i koliko je izmjena primio od domenskih zastupnika s kojima razmjenjuje podatke uvišestručavanjem. Prilikom uvišestručavanja, usporedbom broja izmjena utvrđuje se koji podaci su se izmijenili od posljednjeg usklađivanja. Postupkom uvišestručavanja prenose se samo izmijenjeni podaci, čime se značajno smanjuje mrežni promet u odnosu na slučaj kada se prenose svi podaci.

Prilikom uvišestručavanja podataka može se dogoditi da domenski zastupnik pošalje presliku podataka domenskom zastupniku od kojega je ta izmjena potekla. Domenski zastupnici bi tako mogli slati izmjene jedan drugome u beskonačnoj petlji. Neželjeno

uvišestručavanje podataka sprečava se korištenjem svojstva Originating Write. Svaki zapis sadrži svojstvo Originating Write, koje se uvećava svaki put kada se neki podatak izmijeni od strane korisničkog programa. Izmjena nad zapisom nastala uvišestručavanjem neće uvećati svojstvo Originating Write. Primjer korištenja svojstva Originating Write za sprečavanje neželjenog uvišestručavanja prikazan je na slici 4-10.



Slika 4-10 Upotreba svojstva Originating Write za sprečavanje neželjenog uvišestručavanja

U primjeru na slici 4-10, u domeni se nalaze tri domenska zastupnika: A, B i C. Svi domenski zastupnici nalaze se u istoj domeni i istom polju. Svi domenski zastupnici su međusobno povezani. Domenski zastupnici B i C sadrže jednake preslike zapisa s podacima nad kojim je izvršeno 218 izmjena od strane korisničkog programa. Domenski zastupnik A sadrži presliku istog zapisa koja je izmijenjena 219 puta.

U trenutku (①) domenski zastupnici B i C pokreću uvišestručavanje zahtjevom za prijenos izmijenjenih zapisa sa domenskog zastupnika A. Vrijednost svojstva Originating Write zapisa na domenskom zastupniku A je veća od vrijednosti istog svojstva na domenskim zastupnicima B i C. Domenski zastupnici B i C dobivaju presliku zapisa, pri čemu se preslikava cijeli zapis, uključujući i svojstvo Originating Write sa novom vrijednošću 219.

U trenutku (②) domenski zastupnik B pokreće uvišestručavanje zahtjevom za prijenos izmijenjenih zapisa sa domenskog zastupnika C. Vrijednosti svojstva Originating Write zapisa na domenskim zastupnicima B i C su jednake. Uvišestručavanje se ne izvodi, jer domenski zastupnici sadrže istu presliku zapisa.

Moguće je da dva korisnika izmijene isti zapis na dva različita domenska zastupnika. U tom slučaju prilikom uvišestručavanja dolazi do sukoba zbog postojanja dvije različite preslike istog zapisa. U slučaju sukoba, kao valjana izmjena se uzima ona koja je izvedena posljednja. Ako su izmjene izvedene u istom trenutku, u obzir se uzima zapis koji ima veći GUID (eng. Global Unique Identifier). GUID je 128 bitni broj koji se generira postupkom koji

jamči jedinstvenost broja, pa ne može doći do sukoba kod uvišestručavanja zbog postojanja dva zapisa s jednakim GUID-om.

4.2.3 *Posebne operacije nad podacima u direktoriju*

Svaka preslika podataka u Active Directory-u može biti mijenjana. Uvišestručavanje podataka osigurava prijenos izmjena ostalima domenskim zastupnicima (eng. multimaster replication). Postoje operacije koje obavljaju izmjene koje nisu pogodne za ovu vrstu uvišestručavanja. Takve operacije se mogu izvoditi samo na nositelju operacije (eng. operations master). Nositelj operacije je domenski zastupnik koji je predviđen za obavljanje određene operacije. Postoji pet različitih vrsta nositelja operacije: nositelj sheme (eng. schema master), nositelj imenovanja domena (eng. domain naming master), nositelj relativnih identifikatora (eng. relative ID master), emulator primarnog domenskog zastupnika (eng. PDC emulator) i nositelj infrastrukture (eng. infrastructure master). Uloga nositelja operacije može se dodijeliti bilo kojem domenskom zastupniku, ali u svakom trenutku može postojati samo jedan nositelj operacije u domeni ili skupu, ovisno o kojoj se operaciji radi.

Nositelj sheme je jedini domenski zastupnik na kome se mogu izvoditi izmjene nad shemom. Nositelj imenovanja domena upravlja dodavanjem novih i uklanjanjem postojećih domena u skupu. U svakom trenutku u skupu može postojati samo jedan nositelj sheme, odnosno nositelj imenovanja domena. Ako je nositelj sheme nedostupan uslijed kvara, nije moguće izvoditi izmjene sheme. Ako je nositelj imenovanja domena nedostupan zbog kvara nije moguće dodavati nove ili uklanjati postojeće domene iz skupa. Ulogu nositelja sheme ili nositelja imenovanja domena može preuzeti neki drugi domenski zastupnik u skupu. U tom slučaju domenski zastupnik koji je bio nositelj odgovarajuće operacije ne smije biti spojen na mrežu nakon uklanjanja kvara.

Nositelj relativnih identifikatora rezervira i dodjeljuje niz relativnih identifikatora svakom domenskom zastupniku u domeni. Relativni identifikatori se koriste prilikom stvaranja sigurnosnih identifikatora. Svaki zapis u direktoriju koji opisuje korisnika, računalo ili grupu sadrži sigurnosni identifikator. Sigurnosni identifikator sastoji se od domenskog sigurnosnog identifikatora, koji je isti za svaki sigurnosni identifikator u dotičnoj domeni, i relativnog identifikatora koji je jedinstven za svaki sigurnosni identifikator u dotičnoj domeni. Sigurnosni mehanizmi koriste sigurnosni identifikator za identifikaciju korisnika, računala i grupa. Premještanje (eng. move) zapisa među domenama mora se inicirati na domenskom zastupniku koji ima ulogu nositelja relativnih identifikatora. U svakom trenutku može postojati samo jedan nositelj relativnih identifikatora u svakoj domeni. Ako je nositelj relativnih identifikatora nedostupan zbog kvara, i uslijed stvaranja novih zapisa ponestane relativnih identifikatora, nije moguće stvaranje novih zapisa. Ulogu nositelja relativnih identifikatora može preuzeti neki drugi domenski zastupnik u domeni. U tom slučaju domenski zastupnik koji je bio nositelj relativnih identifikatora ne smije biti spojen na mrežu nakon uklanjanja kvara.

Emulator primarnog domenskog zastupnika omogućuje da se u istoj mreži nalaze računala zasnovana na Windows 2000 operacijskom sustavu i Windows NT operacijskom sustavu. Emulator primarnog domenskog zastupnika ima ulogu primarnog domenskog zastupnika (eng. Primary Domain Controller – PDC) za računala zasnovana na Windows NT operacijskom sustavu. Nositelj infrastrukture je zadužen za održavanje podataka o pripadnosti korisnika grupama, kada su korisnik i grupa u različitim domenama. U svakom trenutku u domeni može postojati samo jedan emulator primarnog domenskog zastupnika, odnosno nositelj infrastrukture. Ako je emulator primarnog domenskog zastupnika ili nositelj infrastrukture nedostupan zbog kvara, njegovu ulogu može preuzeti neki drugi domenski zastupnik. Domenski zastupnik koji je prije pojave kvara imao odgovarajuću ulogu može ponovno preuzeti tu ulogu nakon otklanjanja kvara.

4.3 Active Directory shema

Active Directory shema je skup definicija o vrstama zapisa i tipovima podataka koji mogu biti spremljeni u direktorij [23]. Shemu čine dva osnovna tipa definicija: razredi i atributi. Razredi opisuju zapise koji mogu postojati u direktoriju pa se nazivaju i razredi zapisa. Svaki razred se sastoji od niza atributa. Tip atributa određuje koju vrstu podatka dotični atribut sadrži. Atributi su u shemi definirani zasebno od razreda, što omogućuje korištenje jednog atributa pri definiciji više različitih razreda. Razredi i atributi se nazivaju još i metapodaci (podaci o podacima).

Definicije razreda i atributa su i same spremljene u direktorij kao zapisi, što omogućuje da se njima upravlja istim sučeljem i na isti način kao i s ostalim zapisima u direktoriju. Atributi tih zapisa određuju svojstva definiranih razreda i atributa. Definicije razreda i atributa definirane su u shemi. Definicija razreda pripada razredu `classSchema`, dok definicija atributa pripada razredu `attributeSchema`.

Svaki zapis u shemi referencira se na jedan od tri načina: LDAP imenom (eng. LDAP display name), jednostavnim imenom (eng. common name) ili identifikatorom zapisa (eng. object identifier). LDAP ime se koristi za referenciranje zapisa pri komunikaciji s korisničkim programima koji koriste LDAP protokol. Primjer LDAP imena atributa koji označava elektroničku adresu je `mailAddress`. Jednostavno ime je samo jednostavnija verzija LDAP imena prilagođena krajnjem korisniku. Primjer jedinstvenog imena atributa s LDAP imenom `mailAddress` je `SMTP-Mail-Address`. LDAP ime i jednostavno ime moraju biti jedinstveni u skupu. Identifikator zapisa je broj koji izdaje organizacija poput ISO ili ANSI (eng. American National Standards Institute). Identifikator zapisa predstavljen je nizom brojeva odijeljenih točkama (eng. dotted decimal string). Dodijeljeni identifikator zapisa je jedinstven u svijetu. Ako se u svrhu testiranja koriste identifikatori zapisa koji nisu dobiveni od odgovarajuće organizacije, ti identifikatori moraju biti jedinstveni u skupu. Primjer identifikatora zapisa sa LDAP imenom `mailAddress` je `1.2.840.113556.1.4.786`.

Zapis u direktoriju odgovara formatu definiranom u jednom od razreda definiranih u shemi. Ako zapis odgovara formatu određenom u nekom razredu u shemi, kaže se da je taj zapis izveden iz dotičnog razreda. Osnovni razredi i atributi koji omogućuju opisivanje mrežnih resursa isporučuju se sa Windows 2000 Server operacijskim sustavom. Active Directory omogućuje proširenje sheme definiranjem novih razreda i atributa ili dodavanjem novih atributa postojećim razredima.

4.3.1 Definicija razreda

Definicija svakog razreda sastoji se od strukturnih pravila (eng. structure rules) i pravila o sadržaju (eng. content rules). Strukturna pravila definiraju moguće odnose u hijerarhiji informacijskog stabla direktorija. Svaka definicija razreda sadrži attribute `possSuperiors` i `systemPossSuperiors` koji sadrže liste svih razreda koji mogu biti nadređeni zapisu izvedenom iz dotičnog razreda. Pravila o sadržaju definiraju attribute koje može, odnosno mora, sadržavati svaki zapis izveden iz dotičnog razreda. Atributi `mustContain` i `systemMustContain` definiraju attribute koji moraju biti definirani u svakom zapisu izvedenom iz dotičnog razreda. Atributi `mayContain` i `systemMayContain` definiraju attribute koji mogu, ali ne moraju, biti definirani u zapisu izvedenom iz dotičnog razreda.

Nasljeđivanje omogućuje korištenje već definiranih razreda za izvođenje novih. Svaki razred, osim osnovnog (eng. Top), je izveden iz nekog drugog razreda. Definicija razreda nasljeđuje neke attribute od razreda iz kojeg je izvedena, dok su neki atributi definirani posebno za odgovarajući razred. Nasljeđivanje je rekurzivno, što znači da zapis nasljeđuje attribute svih razreda u hijerarhiji nasljeđivanja.

U shemi se mogu definirati tri vrste razreda:

- **Apstraktni razredi** – (eng. abstract classes) koriste se kao predlošci za definiranje drugih apstraktnih, strukturnih i pomoćnih razreda. Primjer apstraktnog razreda je osnovni razred koji sadrži attribute koje moraju sadržavati svi razredi.
- **Pomoćni razredi** – (eng. auxiliary classes) sadrže liste atributa. Razred koji naslijedi pomoćni razred nasljeđuje sve attribute koji su definirani u pomoćnom razredu. Pomoćni razredi mogu biti izvedeni iz postojećih pomoćnih ili strukturnih razreda. Pomoćne razrede može naslijediti bilo koja vrsta razreda.
- **Strukturni razredi** – (eng. structural classes) jedina vrsta razreda iz koje mogu biti izvedeni zapisi u direktoriju. Strukturni razred može biti izveden iz drugog strukturnog razreda ili apstraktnog razreda. Strukturni razred može naslijediti proizvoljan broj pomoćnih razreda.

Atribut `objectClassCategory` određuje kojoj vrsti razreda pripada dotični razred.

4.3.2 Definicije atributa

Definicija atributa definira vrstu podatka koji atribut može sadržavati, te postavlja ograničenja na duljinu i opseg podataka. Atributi `attributeSyntax` i `oMSyntax` određuju sintaksu atributa, odnosno vrstu podatka koji atribut može sadržavati. Moguće sintakse su unaprijed definirane i nije moguće dodavanje novih sintaksi. Atributi `rangeLower` i `rangeUpper` određuju duljinu niza za znakovne vrste podataka (eng. string) odnosno opseg za numeričke vrste podataka. Ako jedan od ova dva atributa nije naveden, znači da nema ograničenja na donju, odnosno gornju granicu duljine niza, odnosno opseg.

Pojedini atribut može sadržavati jednu ili više vrijednosti (eng. single-value, multi-value). Atribut `isSingleValued` je logičkog tipa i definira da li atribut može sadržavati više vrijednosti. Ako je atribut `isSingleValued` postavljen na vrijednost "istina" (eng. True), atribut može sadržavati samo jednu vrijednost. Primjer takvog atributa je ime korisnika. Ako je atribut `isSingleValued` postavljen na vrijednost "laž" (eng. False), atribut može sadržavati više vrijednosti. Primjer takvog atributa je telefonski broj, jer svaki korisnik može imati više telefonskih brojeva.

Radi lakšeg pronalaska zapisa prema vrijednosti atributa, neki atributi mogu biti indeksirani, što znači da je nad njima izgrađen indeks. Izgradnjom indeksa nad atributom omogućuje se učinkovitije pretraživanje zapisa kada je kriterij pretrage vrijednost dotičnog atributa. Atribut `searchFlags` određuje da li je atribut indeksiran. Ako je atribut indeksiran, svi zapisi koji sadrže taj atribut su indeksirani preko njega. Atribut nad kojim se izgrađuje indeks treba imati što veći broj različitih vrijednosti, kako bi izgrađeni indeks bio učinkovit. Ne preporučuje se izgradnja indeksa nad atributima koji mogu sadržavati više vrijednosti. Stvaranje novog zapisa u direktoriju traje dulje ako zapis sadrži više indeksiranih atributa, jer je potrebno indeksirati svaki atribut novog zapisa.

Pojedini atributi mogu biti uključeni u skup atributa koji se uvišestručavaju svim globalnim katalozima. Da bi se atribut uvišestručavao svim globalnim katalozima, potrebno je postaviti atribut `isMemberOfPartialAttributeSet` na vrijednost "istina". U skup atributa koji se uvišestručavaju svim globalnim katalozima se dodaju atributi koji se često koriste za pretrage među domenama. Poželjno je da ti atributi budu male veličine i da se rijetko mijenjaju, kako bi prilikom uvišestručavanja stvarali što manji promet.

4.3.3 Proširenje Active Directory sheme

Ako se u direktorij žele spremati podaci koji nisu definirani niti jednim od postojećih razreda, shemu je moguće proširiti. Postoje dva osnovna načina za proširenje sheme: dodavanjem novih atributa nekom od postojećih razreda i definiranjem novih razreda. Dodavanje novih atributa nekom od postojećih razreda izvodi se kada je potrebno proširiti već postojeće zapise izvedene iz dotičnog razreda. U svim ostalim slučajevima preporučuje

se mijenjati shemu definiranjem novih razreda. Novi razredi se definiraju izvođenjem iz već postojećih, kao što je opisano u poglavlju 4.3.1.

Schema je zajednička svim domenskim zastupnicima u svim domenama u skupu. Proširenje sheme se uvišestručavanjem prenosi svim domenskim zastupnicima u skupu. Proširenje sheme se može izvoditi samo na domenskom zastupniku koji ima ulogu nositelja sheme u skupu. Schema je na nositelju sheme zaštićena od slučajnih izmjena. Proširenje sheme se omogućuje postavljanjem odgovarajućeg ključa u registru (eng. registry). Ključ se može postaviti izravno ili pomoću administratorskog alata Active Directory Schema. Proširenje sheme mogu izvoditi samo korisnici koji su članovi Schema Admins grupe.

Svaki domenski zastupnik sadrži dvije preslike sheme. Jedna preslika sheme nalazi se na disku, dok je druga u memoriji, radi poboljšanja svojstava sustava. Izmjene se izvode nad shemom zapisanom na disku nositelja sheme. Izmjena se odražava na presliku u memoriji unutar intervala od pet minuta od trenutka izmjene sheme. Za to vrijeme nije moguće izvoditi nove zapise iz razreda koji su upravo definirani. Prijenos izmjene u presliku u memoriji može se pokrenuti u bilo kojem trenutku koristeći administratorski alat Active Directory Schema, ili programski.

Jednom stvoreni, razredi i atributi se ne mogu brisati iz sheme. Moguće ih je samo deaktivirati, čime se sprječava izvođenje novih razreda ili zapisa iz deaktiviranog razreda, te korištenje deaktiviranog atributa u definiranju novih razreda. Nije moguće deaktivirati razred koji je naveden u bilo kojem od atributa `subClassOf`, `auxiliaryClass`, `systemAuxiliaryClass`, `possSuperiors` i `systemPossSuperiors` nekog razreda koji nije deaktiviran. Također, nije moguće deaktivirati atribut koji je naveden u bilo kojem od atributa `mustContain`, `systemMustContain`, `mayContain` i `systemMayContain` nekog razreda koji nije deaktiviran. Drugim riječima; nije moguće deaktivirati razred ili atribut koji koristi definicija nekog drugog razreda koji nije deaktiviran. Razredi i atributi koji su deaktivirani ostaju u shemi, pa nije moguće stvarati nove razrede ili attribute koji bi nosili isto LDAP ili jednostavno ime kao deaktivirani razred ili atribut. Nakon proširenja sheme moguće je mijenjati neke attribute zapisa koji definira novi razred ili atribut, ali ne sve. Proširenje sheme je potrebno pažljivo planirati i provoditi samo kad je neophodno.

Proširenje sheme izvodi se na jedan od tri moguća načina: korištenjem administratorskog alata ActiveDirectory Schema, korištenjem skripti, te programski.

Administratorski alat Active Directory Schema omogućuje proširivanje sheme definiranjem novih razreda i atributa, te izmjenu atributa postojećih zapisa u shemi koje je po definiciji moguće mijenjati. Ovaj način proširenja sheme nije preporučljivo koristiti, osim za jednostavne izmjene sheme u eksperimentalne svrhe.

Windows 2000 Server operacijski sustav pruža mogućnost unosa podataka u direktorij i dohvat podataka iz direktorija korištenjem dva formata skripti: CSV (eng. Comma-Separated Value) i LDIF (eng. LDAP Data Interchange Format). U skripti se na dogovarajući način definiraju parovi atributa i vrijednosti zapisa koje se želi unijeti u direktorij. Za unos vrijednosti

definiranih u skripti koristi se program CSVDE (eng. CSV Directory Exchange) za CSV skripte, i LDIFDE (eng. LDIF Directory Exchange) za LDIF skripte.

S obzirom da su definicije u shemi spremljene kao zapisi, njima se pristupa kao i drugim zapisima u direktoriju, kroz programsko sučelje ADSI. Program za proširenje sheme moguće je napisati u nekom od viših programskih jezika koji podržavaju ADSI. Programsko proširenje sheme je posebno pogodno za aplikacije koji se isporučuju krajnjem korisniku, a za svoj rad zahtijevaju proširenje sheme. Program za proširenje sheme uključuje se u instalacijsku proceduru, pa krajnji korisnik ne mora brinuti o proširenju sheme.

Bez obzira na metodu korištenu za proširenje sheme, potrebno je opširno dokumentirati izvedeno proširenje. Dokumentacija omogućuje pravilno korištenje definiranih razreda i atributa od strane korisnika proširenja.

5 Arhitektura i ostvarenje spremišta podataka

Spremište podataka je podsustav sustava zajedničkih funkcija koji ostvaruje funkcionalnosti za spremanje podataka. Pravilan rad i svojstva sustava zajedničkih funkcija ovise o pravilnom radu i svojstvima spremišta podataka, pa je pri izgradnji sustava potrebno pažljivo ostvariti odgovarajuće spremište podataka. Sustav zajedničkih funkcija čuva podatke o korisnicima, uslugama, pravima korištenja pojedinih usluga, podatke o korištenju usluga i druge. Za spremanje dijela podataka sustava zajedničkih funkcija koristi se Active Directory. Active Directory shema je proširena korisnički definiranim razredima i atributima potrebnim za spremanje podataka sustava zajedničkih funkcija.

5.1 Spremište podataka sustava zajedničkih funkcija

Za normalan rad sustava zajedničkih funkcija u sustavu je potrebno čuvati određene podatke. To su sljedeći podaci:

- **Podaci o korisnicima** – osnovni podaci kao što su ime, lozinka, adresa, broj telefona, elektronička adresa i drugi.
- **Podaci o uslugama** – podaci o uslugama, aplikacijama koje ostvaruju usluge, i funkcijama koje pružaju.
- **Podaci o pravima korištenja usluga** – određuju koji korisnici, usluge ili grupe imaju pravo koristiti pojedine funkcije usluge.
- **Podaci o pravima pristupa korisničkim podacima** – određuju koji korisnici, usluge ili grupe imaju pravo pristupiti podacima o korisniku.
- **Podaci o pripadnosti grupama** – pojedini korisnik, usluga ili grupa mogu biti članovi različitih grupa. Raspoređivanjem u grupe olakšava se administracija sustava.
- **Podaci o korisničkim uređajima** – sadrži podatke o mogućnostima korisničkog uređaja, što omogućuje prilagodbu sadržaja prikaznim mogućnostima uređaja.
- **Podaci o sjednicama** – sadrži podatke o sjednicama u tijeku.
- **Podaci o korištenju usluga** – sadrži podatke o korištenju usluga.

Za spremanje podataka u direktorij važna su vremenska svojstva podataka. Podaci sustava zajedničkih funkcija mogu se razvrstati u tri skupine: podaci koji se često mijenjaju, podaci koji se upisuju i više se ne mijenjaju, te podaci koji se rijetko mijenjaju i često čitaju.

Često se mijenjaju podaci o sjednicama. Kada se stvori nova sjednica, u spremište podataka se spremaju potrebni podaci. Po završetku sjednice podaci se brišu iz spremišta podataka.

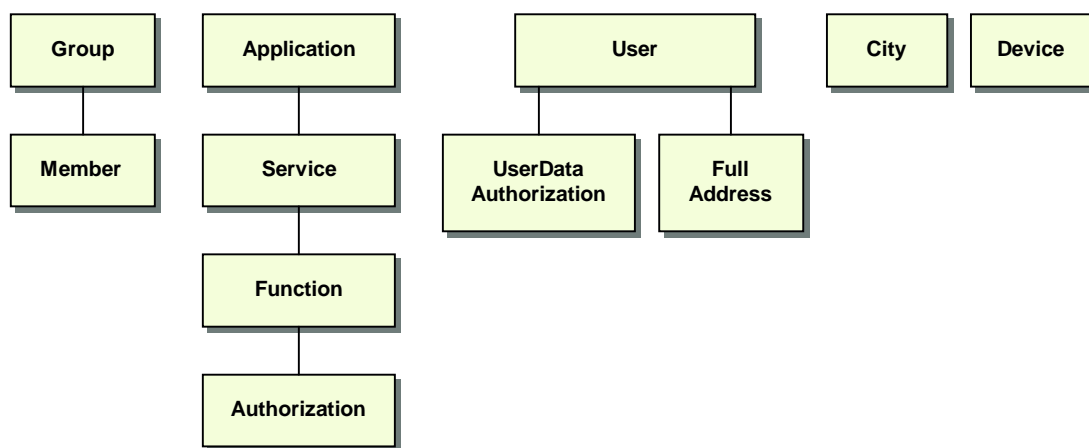
Podaci o korištenju usluga se neprestano upisuju u spremište podataka. Pri svakoj obradi zahtjeva za korištenjem usluge, u spremište podataka se sprema zapis o korištenju

usluge. Podaci o korištenju usluge se rijetko čitaju. Podaci o korištenju usluge koriste se najčešće za naplatu korištenja usluge, pa je vjerojatno da se podaci prikupljaju jednom mjesečno.

Svi podaci sustava zajedničkih funkcija osim podataka o sjednicama i korištenju usluga rijetko se mijenjaju i često čitaju. Takva vremenska svojstva čine ih pogodnima za spremanje u direktorij.

5.2 Organizacija podataka u direktoriju

Podatke koji se spremaju u direktorij potrebno je hijerarhijski organizirati. Hijerarhija podataka koji se spremaju u direktorij prikazana je na slici 5-1.



Slika 5-1 Hijerarhija podataka

Podaci o grupama (eng. Group), aplikacijama (eng. Application), korisnicima (eng. User), gradovima (eng. City) i uređajima (eng. Device) se nalaze u odgovarajućim spremnicima. Podaci koji imaju podređene u hijerarhiji definirani su kao spremnici, te sadrže odgovarajuće podatke: svaka grupa sadrži svoje članove, svaka aplikacija svoje usluge, itd.

Hijerarhija podataka u direktoriju odgovara odnosima među podacima u stvarnom svijetu. Zapisi o uslugama (eng. Service) su u hijerarhiji podređeni zapisima o aplikacijama unutar kojih su usluge ostvarene. Svaki zapis o usluzi sadrži zapise o funkcijama koje su sastavni dio usluge. Svaki zapis o funkciji sadrži zapise s identifikatorima korisnika, usluga i grupa koje imaju pravo koristiti dotičnu funkciju.

Svaki zapis o korisniku sadrži zapise s identifikatorima korisnika, usluga i grupa koje imaju pravo pristupa podacima dotičnog korisnika. Osim identifikatora, zapisi sadrže i liste s pravima pristupa pojedinim podacima o korisniku. Svaki zapis o korisniku sadrži sve adrese dotičnog korisnika. Zapisi o grupama sadrže zapise s identifikatorima korisnika, usluga i drugih grupa koje su članovi dotične grupe.

5.3 Proširenje sheme za spremište podataka sustava zajedničkih funkcija

Active Directory omogućuje spremanje korisnički definiranih podataka. U tu svrhu, shemu direktorija je potrebno proširiti definicijama korisničkih razreda i atributa. Definicije u shemi su spremljene u obliku zapisa. Atributi zapisa u shemi direktorija određuju svojstva definiranih razreda i atributa. Shema je proširena definicijama 11 razreda i 49 atributa. Proširenje sheme izvedeno je korištenjem LDIF skripte.

5.3.1 Atributi zapisa u shemi

Definicije novih razreda i atributa su i same spremljene u shemi direktorija u obliku zapisa. Atributi zapisa u shemi određuju svojstva definiranih razreda i atributa. Pri definiranju novih razreda i atributa opisanih u ovom radu korišteni su sljedeći atributi zapisa u shemi:

- **dn** – (eng. distinguishedName) sadrži puno ime zapisa u shemi
- **adminDisplayName** – ime koje se koristi za prikaz u administratorskim alatima
- **attributeID, governsID** – sadrži identifikator zapisa atributa (attributeID), odnosno razreda (governsID)
- **attributeSyntax, oMSyntax** – određuju sintaksu atributa koji se definira
- **rangeLower, rangeUpper** – par vrijednosti koje definiraju donju i gornju granicu vrijednosti koju atribut može sadržavati, odnosno donju i gornju granicu duljine znakovnog niza
- **cn** – sadrži jednostavno ime zapisa u shemi
- **description** – sadrži opis razreda odnosno atributa
- **isMemberOfPartialAttributeSet** – logička vrijednost koje definira da li se dotični atribut preslikava globalnim katalozima u skupu
- **isSingleValued** – logička vrijednost koja definira da li atribut može sadržavati jednu ili više vrijednosti
- **IDAPDisplayName** – sadrži LDAP ime zapisa u shemi
- **objectCategory, objectClass** – definira razred zapisa. Razred zapisa je classSchema za definiciju razreda, te attributeSchema za definiciju atributa
- **searchFlags** – broj koji definira je li atribut indeksiran, i na koji način
- **defaultHidingValue** – logička vrijednost koja definira da li se zapisi izvedeni iz dotičnog razreda inicijalno prikazuju u administratorskim alatima
- **mustContain** – sadrži imena atributa koje mora sadržavati svaki zapis izveden iz dotičnog razreda
- **mayContain** – sadrži imena atributa koje može, ali ne mora sadržavati zapis izveden iz dotičnog razreda
- **objectClassCategory** – definira vrstu razreda (apstraktni, strukturni ili pomoćni)

- **possSuperiors** - sadrži imena razreda čiji zapisi mogu biti nadređeni zapisima dotičnog razreda
- **subClassOf** - definira razred iz kojeg je izveden dotični razred

Detaljan opis svih navedenih atributa nalazi se u dokumentaciji [22, 23].

5.3.2 Definicije atributa i razreda

Schema je proširena definiranjem novih razreda i atributa. Tablice 5-1 i 5-2 prikazuju svojstva definiranih atributa i razreda.

Tablica 5-1 Svojstva atributa definiranih pri proširenju sheme

LDAP ime	Sintaksa	Duljina	Jedna vrijednost	Indeks
ETKGroupID01	Znakovni niz	20	T	–
ETKGroupName01	Znakovni niz	255	T	–
ETKMemberID01	Znakovni niz	20	T	T
ETKAppID01	Znakovni niz	20	T	–
ETKAppName01	Znakovni niz	255	T	–
ETKAppDescription01	Znakovni niz	255	T	–
ETKAppOwner01	Znakovni niz	255	T	–
ETKAppURL01	Znakovni niz	255	T	–
ETKServiceID01	Znakovni niz	20	T	T
ETKServiceName01	Znakovni niz	255	T	–
ETKServicePassword01	Znakovni niz	255	T	–
ETKServiceDescription01	Znakovni niz	255	T	–
ETKServiceURL01	Znakovni niz	255	T	–
ETKServiceEndPoint01	Znakovni niz	255	T	T
ETKSOAPAction01	Znakovni niz	255	T	T
ETKWSDLFileLocation01	Znakovni niz	255	T	–
ETKAuthorizationRequired01	Logički tip	–	T	–
ETKFunctionID01	Cjelobrojni tip	–	T	–
ETKFunctionName01	Znakovni niz	255	T	T
ETKFunctionDescription01	Znakovni niz	255	T	–
ETKInput01	Znakovni niz	255	T	–
ETKOutput01	Znakovni niz	255	T	–
ETKExtra01	Znakovni niz	255	T	–
ETKAuthorizedID01	Znakovni niz	20	T	T
ETKRequesterID01	Znakovni niz	20	T	–
ETKUserNameAuthorization01	Logički tip	–	T	–
ETKUserJMBGAuthorization01	Logički tip	–	T	–
ETKUserCityAuthorization01	Logički tip	–	T	–
ETKUserAddressAuthorization01	Logički tip	–	T	–
ETKUserPhoneAuthorization01	Logički tip	–	T	–
ETKUserEmailAuthorization01	Logički tip	–	T	–

LDAP ime	Sintaksa	Duljina	Jedna vrijednost	Indeks
ETKUserGroupsAuthorization01	Logički tip	–	T	–
ETKUserID01	Znakovni niz	20	T	–
ETKUserName01	Znakovni niz	255	T	–
ETKUserPassword01	Znakovni niz	255	T	–
ETKJMBG01	Znakovni niz	13	T	–
ETKPhoneNumber01	Znakovni niz	20	–	–
ETKEmail01	Znakovni niz	255	–	–
ETKZipCode01	Znakovni niz	16	T	–
ETKAddress01	Znakovni niz	255	T	–
ETKCityName01	Znakovni niz	255	T	T
ETKDeviceID01	Znakovni niz	20	T	–
ETKDevicePassword01	Znakovni niz	255	T	–
ETKResolutionX01	Cjelobrojni tip	–	T	–
ETKResolutionY01	Cjelobrojni tip	–	T	–
ETKColorDepth01	Cjelobrojni tip	–	T	–
ETKIPAddress01	Znakovni niz	15	T	–
ETKPort01	Cjelobrojni tip	–	T	–
ETKDeviceType01	Znakovni niz	255	T	–

Tablica 5-2 Svojstva razreda definiranih pri proširenju sheme

LDAP ime	Obavezni atributi	Opcionalni atributi	Nadređeni
ETKGroup01	<i>ETKGroupID01</i>	ETKGroupName01	container
ETKMember01	<i>ETKMemberID01</i>	–	ETKGroup01
ETKApplication01	<i>ETKAppID01</i>	ETKAppName01 ETKAppDescription01 ETKAppOwner01 ETKAppURL01	container
ETKService01	<i>ETKServiceID01</i> ETKServicePassword01 ETKServiceEndPoint01 ETKAuthorizationRequired01	ETKServiceName01 ETKServiceDescription01 ETKServiceURL01 ETKSOAPAction01 ETKWSDLFileLocation01	ETKApplication01

LDAP ime	Obavezni atributi	Opcionalni atributi	Nadređeni
ETKFunction01	<i>ETKFunctionID01</i> ETKFunctionName01	ETKFunctionDescription01 ETKInput01 ETKOutput01 ETKExtra01	ETKService01
ETKAuthorization01	<i>ETKAuthorizedID01</i>	–	ETKFunction01
ETKUser01	<i>ETKUserID01</i> ETKUserName01 ETKUserPassword01 ETKJMBG01	ETKPhoneNumber01 ETKEmail01	container
ETKUserDataAuthorization01	<i>ETKRequesterID01</i> ETKUserNameAuthorization01 ETKUserJMBGAuthorization01 ETKUserCityAuthorization01 ETKUserAddressAuthorization01 ETKUserPhoneAuthorization01 ETKUserEmailAuthorization01 ETKUserGroupsAuthorization01	–	ETKUser01
ETKFullAddress01	ETKZipCode01 ETKAddress01	–	ETKUser01
ETKDevice01	<i>ETKDeviceID01</i>	ETKDevicePassword01 ETKResolutionX01 ETKResolutionY01 ETKColorDepth01 ETKIPAddress01 ETKPort01 ETKDeviceType01	container
ETKCity01	<i>ETKZipCode01</i> ETKCityName01	–	container

Tablica 5-1 prikazuje svojstva definiranih atributa. Atributi su označeni LDAP imenom. Za svaki atribut je navedena sintaksa, maksimalna duljina znakovnog niza, smije li atribut sadržavati samo jednu vrijednost i je li atribut indeksiran. Oznaka "T" označava da atribut smije sadržavati samo jednu vrijednost, odnosno da je indeksiran. Oznaka "-" označava da atribut može sadržavati više vrijednosti, odnosno da atribut nije indeksiran.

Tablica 5-2 prikazuje svojstva definiranih razreda. Razredi su označeni LDAP imenom. Za svaki razred su navedeni obavezni i opcionalni atributi, te razred koji može biti nadređen definiranom razredu. Oznaka "-" označava da razred nema opcionalnih atributa. Za svaki razred je u stupcu s obaveznim atributima masnim slovima u kurzivu označen atribut koji se koristi za stvaranje relativnog imena zapisa. Izuzetak je razred ETKFulladdress01 sa definicijom adrese korisnika.

Relativno ime zapisa koji predstavlja adresu korisnika sastoji se od poštanskog broja, znakovnog niza "\$\$\$" i rednog broja adrese za taj grad. Npr. ako isti korisnik ima u gradu s poštanskim brojem 10000 dvije adrese, prva adresa će imati relativno ime "CN=10000\$\$\$1", a druga "CN=10000\$\$\$2". Za relativno ime zapisa ne može se koristiti puna adresa, jer je duljina atributa za imenovanje zapisa ograničena na 64 znaka.

Pri imenovanju razreda i atributa korištene su preporuke za imenovanje navedene u dokumentaciji Active Directory-a. Svi razredi i atributi u imenu sadrže ime kompanije koja koristi proširenje (ETK) i verziju proširenja (01).

Identifikatori zapisa korišteni za proširenje sheme dobiveni su korištenjem programa oidgen.exe koji se isporučivao sa testnim verzijama Windows 2000 Server operacijskog sustava.

Definicije razreda i atributa navedene su u LDIF skripti SchemaExtension.ldf na priloženom mediju.

5.3.3 *Proširenje sheme*

Proširenje sheme izvodi se zadavanjem odgovarajućih parametara programu ldifde.exe. Parametri definiraju datoteku u kojoj se nalazi LDIF skripta s definicijama novih razreda i atributa, te DNS ime računala koje je nositelj sheme.

Kako bi se postupak proširenja sheme automatizirao, te se u potpunosti uklonila potreba za intervencijom od strane korisnika, izrađena je skripta u VBScript jeziku. Skripta Extend.vbs čita podatke o domeni u kojoj se nalazi računalo na kojem se skripta izvodi iz korijenskog zapisa informacijskog stabla direktorija. Proširenje sheme se izvodi prosljeđivanjem pročitanih podataka programu ldifde.exe. Prije samog proširenja, skripta postavlja odgovarajući ključ u registru, kako bi se omogućile izmjene sheme. Po završetku proširenja, skripta briše ključ iz registra, kako bi se spriječile slučajne, neželjene izmjene sheme.

Da bi operacija proširenja sheme bila uspješna, korisnik koji pokreće skriptu Extend.vbs mora imati ovlasti za stvaranje ključeva u dijelu registra koji definira prava nad Active

Directory-em. Korisnik mora biti član grupe Schema Admins, inače će pokušaj proširenja sheme rezultirati pogreškom. Proširenje sheme mora se provoditi na domenskom zastupniku koji je nositelj sheme u skupu.

Ako su ispunjeni svi preduvjeti za uspješno proširenje sheme, dovoljno je pokrenuti skriptu Extend.vbs. Po završetku proširenja sheme, u datoteci Idif.log nalazi se zapis o izvedenim izmjenama nad shemom. Ako je prilikom pokušaja proširenja sheme došlo do pogreške, stvara se datoteka Idif.err koja sadrži opis pogrešaka.

Nakon uspješnog proširenja sheme potrebno je neko vrijeme da se izmjene preslikaju na druge domenske zastupnike u skupu. Svi zapisi koji se upisuju u direktorij moraju odgovarati pravilima navedenima u proširenju sheme.

Datoteke Extend.vbs i SchemaExtension.ldf, potrebne za proširenje sheme, nalaze se na priloženom mediju.

6 Ostvarenje programske potpore pristupu direktoriju

Programska potpora pristupu direktoriju omogućuje korištenje direktorija bez potrebe poznavanja organizacije podataka unutar direktorija. Korisnik sučelja ne mora znati ništa o samom direktoriju. Potrebno je poznavati samo logičku organizaciju podataka. Programska potpora ostvarena je izradom sučelja prema direktoriju u obliku funkcija za Microsoft .NET Framework platformu. Za izradu programske potpore korištene su funkcionalnosti .NET Framework knjižnice osnovnih razreda.

6.1 Microsoft .NET Framework

Microsoft .NET Framework je platforma za razvoj aplikacija prilagođenih globalnoj mreži Internet. .NET Framework se sastoji od dvije osnovne komponente: potpore izvođenju programa pisanih u različitim programskim jezicima (eng. Common Language Runtime – CLR) i knjižnice osnovnih razreda (eng. .NET Framework class library) [24].

CLR pruža funkcionalnosti za provjeru i prevođenje koda, suradnju programa pisanih u različitim programskim jezicima, primjenu sigurnosnih postavki, upravljanje memorijom i druge.

Knjižnicu osnovnih razreda čini skup razreda i tipova koje je moguće koristiti prilikom razvoja korisničkih programa. Knjižnica osnovnih razreda uvelike olakšava izradu programskih rješenja, pružajući potporu za rad s osnovnim tipovima podataka, pristup datotečnom sustavu, pristup bazama podataka, izradu grafičkih sučelja i sl.

Osnovni blokovi od kojih se grade .NET Framework aplikacije nazivaju se agregati (eng. assembly).

6.1.1 Common Language Runtime

CLR je osnova .NET Framework platforme. CLR upravlja izvođenjem programa pružajući osnovne funkcionalnost potrebne za uspješno izvođenje programa kao što su provjera i prevođenje koda, upravljanje sigurnosnim postavkama, upravljanje memorijom, upravljanje dretvama, i sl.

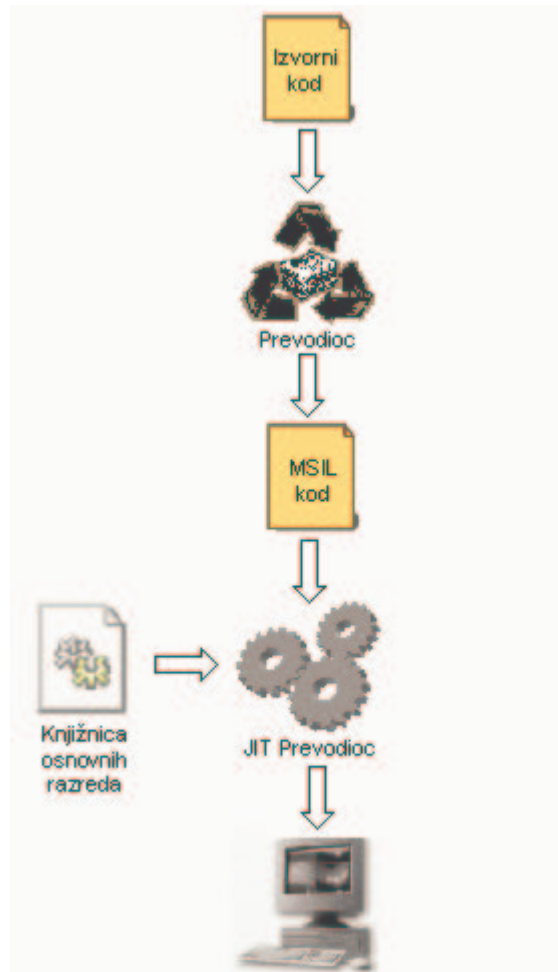
Izvorni kod programa prevodi se u MSIL (eng. Microsoft Intermediate Language) jezik. MSIL je jezik nezavisan od arhitekture računala. MSIL se sastoji od skupa instrukcija koje se mogu efikasno prevoditi u strojni kod računala na kojem se program izvodi. U skupu instrukcija MSIL jezika nalaze se instrukcije za upravljanje metodama objekata, aritmetičke i logičke operacije, kontrolu tijeka izvođenja (eng. control flow), izravni pristup memoriji (eng. Direct Memory Access - DMA), upravljanje iznimkama (eng. exception handling) i druge. Osim samog MSIL koda, postupkom prevođenja generiraju se i podaci o kodu (eng.

metadata). Podaci o kodu opisuju tipove podataka i metode nad podacima korištene u kodu, kao i druge podatke potrebne za uspješno izvođenja programa.

MSIL je nezavisan od programskog jezika izvornog koda. Izvorni kod se može prevesti u MSIL kod bez obzira u kojem je programskom jeziku pisan, ako postoji odgovarajući prevodioc. Kao dio .NET Framework platforme Microsoft isporučuje prevodioce za programske jezike Visual Basic, C#, Visual C++ i JScript. S obzirom da se svi jezici prevode u jedan zajednički izlazni jezik, moguće je pisanje programa u više različitih programskih jezika koji međusobno surađuju. Tako npr. svaki programer u timu može pisati izvorni kod u jeziku u kojem se najbolje snalazi. Svaki programski jezik ima svoje specifičnosti u pogledu tipova podataka koje koristi i metodama nad objektima koje pruža. Specifikacija zajednička svim jezicima (eng. Common Language Specification – CLS) definira skup funkcionalnosti koje su zajedničke svim jezicima. Programi pisani u različitim programskim jezicima mogu uspješno surađivati ako koriste isključivo funkcionalnosti definirane u specifikaciji.

Da bi se kod mogao izvoditi na ciljnom računalu, potrebno je MSIL kod prevesti u strojni kod ciljnog računala. Postupak prevođenja u strojni kod ciljnog računala obavlja se neposredno prije izvođenja. JIT (eng. Just-In-Time) prevodioc prevodi MSIL kod u strojni kod računala. Prilikom izvođenja programa neki dijelovi koda se nikada ne izvode. JIT prevodioc prevodi pojedine dijelove koda pri prvom pozivu dotičnog dijela koda. Prevedeni kod se sprema, pa ga nije potrebno ponovo prevoditi pri daljnjim pozivima. MSIL kod se može izvoditi na svakom računalu za koje postoji JIT prevodioc.

Postupak prevođenja i izvođenja programa na .NET Framework platformi prikazan je na slici 6-1.



Slika 6-1 Postupak prevođenja i izvođenja programa na .NET framework platformi

Kod izvornog programa se prevodi u MSIL kod i zapisuje u izvršnu datoteku. Prilikom pokretanja programa u izvršnoj datoteci, JIT prevodioc prevodi dijelove MSIL koda u instrukcije ciljnog računala. Tijekom postupka prevođenja i izvođenja prevode se i izvode i korištene metode .NET Framework knjižnice osnovnih razreda. Za uspješno izvođenje programa, računalo na kojem se izvodi program mora imati ugrađenu potporu izvođenju programa pisanih u različitim programskim jezicima (CLR).

Upravljanje sigurnosnim postavkama omogućuje definiranje operacija koje program smije izvesti na računalu na kojem se izvodi. Sigurnosne postavke definiraju se s obzirom na mjesto pokretanja programa i s obzirom na isporučitelja programa. Time se omogućuje izrada sigurnih aplikacija za Internet. Tako npr. program koji je pokrenut sa Internet stranice kojoj korisnik ne vjeruje nema prava pristupa resursima na lokalnom računalu, čime je lokalno računalo zaštićeno od zlonamjernih napada.

Upravljanje memorijom provodi u potpunosti CLR. CLR upravlja referencama na objekte u memoriji te automatski oslobađa memoriju koju zauzima objekt koji se više ne koristi (eng.

garbage collection). Prije samog pokretanja programa obavljaju se stroge sigurnosne provjere svih tipova. Program smije pristupati samo memorijskim lokacijama za koje je ovlašten, te pozivati samo unaprijed definirane metode nad objektima u memoriji. Programski kod koji sadrži definirani opis koda, i koristi CLR naziva se upravljanim kodom (eng. managed code). Ako programski kod ne prođe sigurnosnu provjeru, program se ne pokreće, osim ako je administrator sustava dozvolio pokretanje takvih programa. Osim upravljanog koda CLR podržava i neupravljani kod (eng. unmanaged code). Neupravljani kod ne koristi mogućnosti CLR-a kao što je upravljanje memorijom ili sigurnosnim postavkama, već sam ostvaruje te funkcionalnosti. Potpora neupravljanom kodu omogućuje korištenje gotovih komponenti razvijenih prije pojave .NET Framework platforme, ali unosi određeni sigurnosni rizik.

6.1.2 Agregati

Agregati su osnovni građevni blokovi .NET Framework aplikacija. Agregat je skup podataka i funkcionalnosti koji čine jednu logičku cjelinu. Podaci i funkcionalnosti ostvareni su unutar razreda, odnosno struktura. Agregati su najmanji dijelovi koda koji se mogu isporučivati kao nezavisne cjeline. Agregat sadrži sve informacije potrebne za uspješno izvođenje koda, kao što su podaci o tipovima podataka i metodama nad objektima, sigurnosne postavke, te podaci o verziji agregata. Svaka aplikacija sastoji se od jednog ili više agregata.

Agregat se može sastojati od četiri komponente: manifesta, metapodataka, MSIL koda i pomoćnih resursa. Manifest agregata je jedina komponenta koja mora postojati u svakom agregatu. Manifest agregata opisuje odnose među elementima koji čine agregat. Manifest sadrži ime i verziju agregata, regionalne postavke koje agregat podržava, sažetak svih datoteka koje čine agregat, podatke o vezi među tipovima i datotekama u kojima su tipovi deklarirani i ostvareni, podatke o drugim agregatima koje koristi, te, ako je riječ o agregatu s čvrstim imenom (eng. strong-named assembly), javni ključ korišten za digitalni potpis kojim je agregat potpisan. Agregati s čvrstim imenom omogućuju postojanje različitih verzija istog agregata na računalu. Pomoćni resursi mogu biti različiti slikovni ili zvučni zapisi, fontovi, ikone i sl. Komponente agregata mogu biti smještene u više različitih datoteka na disku, ili sve u jednoj datoteci. Datoteka koja sadrži manifest agregata ima ekstenziju exe ili dll, ovisno da li se radi o programu ili knjižnici.

Manifest agregata i metapodaci sadrže sve podatke o agregatu koji su potrebni za njegovo uspješno korištenje. Nije potrebno spremati dodatne konfiguracijske podatke u registar ili konfiguracijske datoteke, što uklanja potrebu za složenim instalacijskim procedurama. Program se, zajedno s agregatima koje koristi, može jednostavno preslikati u proizvoljnu mapu (eng. folder) na disku.

Prije .NET Framework platforme za Windows operacijske sustave, najmanji dijelovi koda koji su se isporučivali nezavisno bile su dinamičke knjižnice (eng. Dynamic Link Library –

DLL). Nedostatak dinamičkih knjižnica je nemogućnost istovremenog postojanja različitih verzija iste knjižnice na računalu. Prilikom izrade nove verzije dinamičke knjižnice mora se očuvati kompatibilnost unatrag, kako bi programi koji koriste funkcionalnosti ugrađene u prethodne verzije knjižnice i dalje uspješno radili. Kompatibilnost unatrag može biti teško, ili nemoguće ostvariti, zbog čega nastaju problemi s programima koji koriste prethodne verzije knjižnice. .NET Framework platforma omogućuje izgradnju agregata s čvrstim imenom, koji su građeni tako da se izbjegnu navedeni problemi.

Agregati s čvrstim imenom omogućuju imenovanje agregata na način koji osigurava jedinstvenost imena, te omogućuje istovremeno postojanje različitih verzija istog agregata na računalu. Svaki program koristi isključivo onu verziju agregata koja je definirana prilikom prevođenja, osim ako administrator sustava dozvoli korištenje drugih verzija.

Čvrsto ime se sastoji od imena agregata, verzije i regionalnih postavki koje agregat podržava, zajedno sa digitalnim potpisom i odgovarajućim javnim ključem. Digitalnim potpisom potpisuje se datoteka koja sadrži manifest agregata. S obzirom da manifest agregata sadrži sažetke svih datoteka koje čine agregat, čvrsto ime omogućuje provjeru integriteta agregata. Agregati s jednakim čvrstim imenima smatraju se identičnima. Čvrsto ime ne omogućuje provjeru identiteta izdavača agregata. Za provjeru identiteta izdavača potrebni su certifikati. Čvrsto ime daje jamstvo da su dvije različite verzije agregata nastale od istog izdavača, ako je za digitalni potpis korišten isti par ključeva.

Ako više aplikacija koristi neki agregat, agregat se može dodati u priručnu memoriju agregata (eng. Global Assembly Cache – GAC). Priručna memorija agregata postoji na svakom računalu na kojem postoji CLR. U priručnoj memoriji agregata se inicijalno nalaze agregati koji su dio knjižnice osnovnih razreda. Agregati koji se dodaju u priručnu memoriju agregata moraju imati čvrsto ime, čime se osigurava jedinstvenost imena agregata u priručnoj memoriji. Dodavanje agregata u priručnu memoriju mogu obavljati korisnici sa administratorskim privilegijama. Ako neka aplikacija koristi agregat koji se nalazi u priručnoj memoriji agregata, aplikaciju nije moguće prenijeti na drugo računalo jednostavnim preslikavanjem, već je potrebna posebna instalacijska procedura. Zbog toga se dodavanje agregata u priručnu memoriju preporuča samo kada veći broj aplikacija koristi jedan agregat.

Kada neki program u svome radu zahtijeva funkcionalnost koja je ostvarena u nekom agregatu, pristup agregatu se obavlja kroz četiri koraka:

1. Provjerava se verzija agregata definirana u originalnom programu.
2. Provjeravaju se definicije u konfiguraciji sustava koje definiraju verzije agregata koje se smiju koristiti.
3. Na osnovu podataka dobivenih u prva dva koraka određuje se verzija agregata koja će se pozivati.
4. Pokušava se pronaći prethodno određena verzija agregata. Agregat se traži redom u: priručnoj memoriji agregata, mjestu definiranom u konfiguraciji

sustava, te konačno u mapi u kojoj se nalazi sam program, ili u njenim podmapama (eng. subfolders).

Osim što na istom računalu mogu postojati različite verzije agregata, moguće je da se na računalu istovremeno izvode različite verzije agregata (eng. side-by-side execution). Različite verzije agregata mogu se izvoditi u različitim, ali i u istom procesu. Agregati koji podržavaju istovremeno izvođenje različitih verzija moraju biti pažljivo programirani, tako da nema međusobnih zavisnosti koje bi narušavale normalan rad aplikacije.

6.1.3 Tipovi podataka

Tipovi podataka se dijele na vrijednosne tipove (eng. value types) i reference (eng. reference types). Varijabla vrijednosnog tipa uvijek sadrži vrijednost toga tipa. Pridruživanje varijabli vrijednosnog tipa obavlja se preslikavanjem vrijednosti, dok pridruživanje referenci preslikava samo referencu, ne i referencirani objekt. Mjesto za vrijednosne tipove rezervira se na stogu (eng. stack), osim ako su dijelovi referenci. Za reference se rezervira mjesto na stogu za adresu objekta na gomili (eng. heap).

Vrijednosni tipovi uključuju jednostavne tipove, logički tip, pobrojani tip i strukture. U skupinu jednostavnih tipova pripadaju cjelobrojni tipovi, tipovi zapisa brojeva u pomičnom zarezu i decimalni tip. Decimalni tip se razlikuje od zapisa broja u pomičnom zarezu po preciznosti i opsegu. Strukture se razlikuju od razreda po tome što ne mogu biti izvedene iz drugih tipova, i ne mogu se nasljeđivati.

U skupinu referenci pripadaju sučelja (eng. interface), razredi i delegati (eng. delegate). Reference mogu sadržavati druge tipove. Tipove koji su sadržani unutar nekog drugog tipa nazivaju se članovima toga tipa.

Sučelje ne sadrži nikakvo ostvarenje, već samo deklaraciju članova koji moraju biti ostvareni u razredu koji ostvaruje sučelje. Razred koji ostvaruje sučelje je razred koji nasljeđuje sučelje i ostvaruje njegove članove.

Razredi služe za opisivanje strukture i ponašanja stvarnih objekata. Svaki objekt je pojavnost nekog razreda. Način deklaracije razreda ovisi o programskom jeziku. Neki programski jezici dozvoljavaju samo jednostruko nasljeđivanje, ali razred može ostvarivati više sučelja (eng. interface).

Delegati ostvaruju funkcionalnost pokazivača na funkcije. Razlika između delegata i pokazivača na funkcije je u tome što se kod definicije delegata izvode stroge provjere tipova.

.NET Framework platforma definira poseban tip reference Objekt (eng. object). Objekt je referenca kojoj se mogu pridruživati vrijednosti bilo koje druge varijable. Svi tipovi, vrijednosni i reference, predefimirani i korisnički definirani, implicitno nasljeđuju Objekt. Objekt ima predefimirane neke korisne metode, kao što su Equals, Finalize, GetHashCode i ToString. Metoda Equals omogućuje usporedbu dva objekta. Metoda Finalize omogućuje oslobađanje resursa prije automatskog uništavanja objekta. Metoda GetHashCode vraća sažetak objekta nad kojim je pozvana. Metoda ToString vraća znakovni niz koji sadrži

reprezentaciju objekta nad kojim je pozvana. Tipovi koji nasljeđuju Objekt mogu preopteretiti ove metode.

6.1.4 Imenici

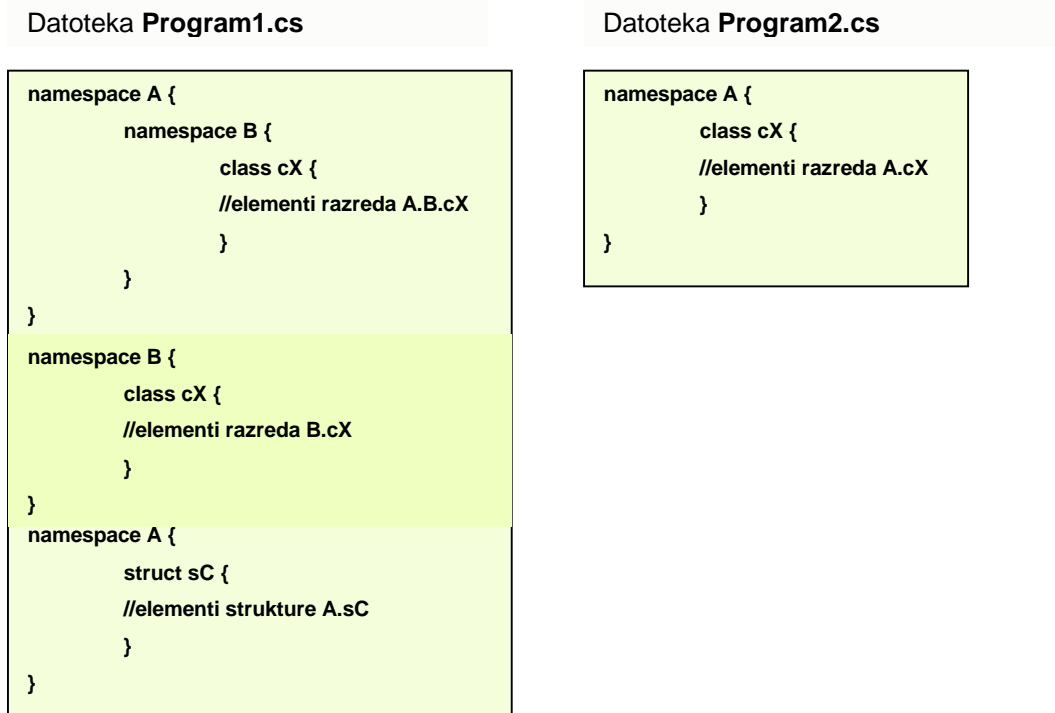
Programi se logički organiziraju korištenjem imenika (eng. namespace). Imenici se koriste za organizaciju samog programa, te za organizaciju sučelja prema drugim programima.

Imenici omogućuju da unutar jednog programa postoje dva objekta s istim imenom, ali u različitim imenicima. Ti objekti se razlikuju po punom imenu (eng. fully qualified name). Puno ime se tvori od imena imenika i imena objekta odvojenih točkom. Imenici mogu sadržavati druge imenike, razrede, sučelja, strukture, pobrojane tipove i delegate.

Za deklariranje imenika koristi se ključna riječ namespace, a nakon nje slijedi blok s deklaracijama i ostvarenjem članova imenika. Ako se u nekom programu žele koristiti objekti definirani u nekom drugom imeniku, bez navođenja punog imena tih objekata, koristi se ključna riječ using, a nakon nje se navodi ime dotičnog imenika. Svi imenici imaju javno pravo pristupa.

U svakom programu postoji pretpostavljeni imenik (eng. default namespace, global namespace). Pretpostavljeni imenik nije potrebno posebno deklarirati. Svi identifikatori deklarirani u pretpostavljenom imeniku vidljivi su u ostalim imenicima.

Isti imenik može biti deklariran nekoliko puta u istoj datoteci, ili različitim datotekama programa, ili čak u različitim programima. Pri tome sve deklaracije članova dodaju članove istom imeniku. Primjer organizacije programa korištenjem imenika prikazan je na slici 6-2.



Slika 6-2 Organizacija programa korištenjem imenika

Imenik A na slici 6-2 sadrži imenik B s punim imenom A.B, strukturu sC s punim imenom A.sC i razred cX s punim imenom A.cX. Imenik punog imena A.B sadrži razred cX s punim imenom A.B.cX. Imenik punog imena B sadrži razred cX punog imena B.cX.

6.1.5 Upravljanje iznimkama

Iznimke su situacije u kojima program ne može nastaviti svojim normalnim tokom. U tom slučaju je potrebno izvođenje prenijeti na poseban dio koda koji obrađuje novonastalu situaciju. Iznimka se predstavlja objektom. U knjižnici osnovnih funkcija .NET Framework platforme postoji razred System.Exception koji služi kao osnovni razred za sve iznimke. Za upravljanje iznimkama koriste se ključne riječi throw, try, catch i finally.

Do pojave iznimke može doći u normalnom radu programa. Iznimka se pojavljuje npr. kada se pokuša pristupiti elementu polja čiji indeks prekoračuje definiranu veličinu polja. U tom slučaju CLR podiže iznimku (eng. exception is thrown). Osim iznimki koje podiže CLR, programer može podići vlastitu, korisnički definiranu iznimku. Za podizanje iznimke koristi se ključna riječ throw. Kao parametar se prenosi objekt koji predstavlja iznimku.

Ključna riječ try koristi se za označavanje bloka u kojem se želi upravljati iznimkama. Ključna riječ catch služi za označavanje bloka za obradu iznimke nastale u prethodnom try bloku. Parametar je pogreška ili vrsta pogreške koju dotični catch blok obrađuje. Jednom try bloku može biti pridruženo više catch blokova koji obrađuju različite iznimke. Ako se u try

bloku dogodi iznimka, obradu iznimke obavlja prvi catch blok koji je obrađuje dotičnu iznimku. Ako u try bloku ne dođe do pojave iznimke, ne izvodi se niti jedan catch blok.

Ključna riječ finally koristi se za označavanje bloka koji će se sigurno izvesti, neovisno o pojavi iznimke u prethodnom try bloku. Finally blok služi za oslobađanje resursa eventualno rezerviranih u prethodnom try bloku. Finally blok se može koristiti i u kombinaciji sa catch blokovima. U tom slučaju catch blokovi obrađuju eventualnu iznimku, nakon čega finally blok oslobađa eventualno rezervirane resurse.

6.1.6 Višedretvenost

.NET Framework knjižnica osnovnih razreda definira razrede i strukture potrebne za stvaranje, sinkronizaciju i prekidanje rada dretvi. Jednom napisani programski kod može se izvoditi na bilo kojem operacijskom sustavu za koji postoji CLR.

Za izradu višedretvenih programa u korisnički program je potrebno uključiti imenik System.Threading. Pri izradi programske potpore korišteni su razredi Thread i Monitor, delegat ThreadStart i pobrojani tip ThreadState.

Objekt razreda Thread predstavlja dretvu programa. Konstruktor razreda prima preko parametra delegat funkcije koja se poziva prilikom pokretanja nove dretve. Nova dretva počinje s radom pozivom metode Start nad objektom razreda Thread. Metoda Start može se nad dretvom pokrenuti samo jednom. Čitanjem logičke vrijednosti svojstva IsAlive određuje se da li je dretva postala aktivna. Ako svojstvo IsAlive vraća vrijednost "istina", dretva je pokrenuta i nije prekinuta. Dretva se privremeno suspendira pozivom metode Suspend. Suspendirana dretva nastavlja s radom nakon poziva metode Resume od strane druge dretve. Metoda Abort izaziva nasilan prekid rada dretve nad kojom je pozvana. Metoda Sleep omogućuje privremeno deaktiviranje dretve unaprijed određeno vrijeme, ili do prekida pozivom Interrupt metode od strane druge dretve. Pozivom metode Interrupt nad nekom dretvom podiže se iznimka ThreadInterruptedException. Iznimka se podiže samo ako je dretva privremeno deaktivirana, čeka na završetak druge dretve, ili čeka na dodjelu zajedničkog sredstva. Dretva može na odgovarajući način obraditi nastalu iznimku i nastaviti s radom. Svojstvo ThreadState određuje stanje dretve. Svojstvo ThreadState sadrži jednu od vrijednosti pobrojanog tipa ThreadState. Metoda Join omogućuje čekanje na završetak druge dretve. Svojstvo IsBackground određuje da li je dretva pozadinska, ili ne. Rad pozadinske dretve se automatski nasilno prekida nakon što s radom završe sve dretve koje nisu pozadinske. Dretva referencira samu sebe preko svojstva CurrentThread, koje vraća referencu na trenutnu dretvu.

Razred Monitor ostvaruje metode potrebne za sinkronizaciju pristupa zajedničkom sredstvu. Sve metode razreda Monitor su statičke metode.

Metoda Enter omogućuje zauzimanje sinkronizacijskog objekta, odnosno ulazak u kritični odsječak koda. Sinkronizacijski objekt je bilo koji objekt tipa reference i prosljeđuje se metodi Enter kao parametar. Metoda Enter je blokirajuća, i privremeno zaustavlja izvođenje

dretve ako se neka druga dretva nalazi u kritičnom odsječku, odnosno pozvala je metodu Enter sa istim objektom kao parametrom. Monitor ne štiti objekt zadan kao parametar pri pozivu metode Enter od pisanja, već samo sprečava da druge dretve uđu u kritični odsječak koda.

Metoda TryEnter je neblokirajuća verzija metode Enter. Ako metoda TryEnter uspije ući u kritični odsječak, vraća vrijednost "istina", inače "laž". Metoda TryEnter je preopterećena, te omogućuje prijenos parametra koji određuje najdulji vremenski interval koji metoda čeka na ulazak u kritični odsječak. Ako po isteku zadanog vremena metoda ne dobije pravo ulaska u kritični odsječak vraća vrijednost "laž".

Metoda Exit nad sinkronizacijskim objektom označava otpuštanje sinkronizacijskog objekta, odnosno izlazak dretve iz kritičnog odsječka. Ako je ista dretva pozvala metodu Enter više puta nad istim objektom, metoda mora jednako toliko puta pozvati metodu Exit nad tim objektom, kako bi druge dretve mogle ući u kritični odsječak.

Metode Wait, Pulse i PulseAll omogućuju čekanje na neki događaj, i signaliziranje da se događaj dogodio. Dretva koja pozove metodu Wait otpušta zauzeti sinkronizacijski objekt, stavlja se u red blokiranih dretvi dotičnog objekta i čeka signal. Druga dretva može signalizirati jednu ili sve dretve u redu blokiranih dretvi objekta da je došlo do nekog događaja pozivom metoda Pulse ili PulseAll. Događaj je najčešće vezan uz promjenu stanja nekog objekta. Metode Pulse, odnosno PulseAll prebacuju prvu, odnosno sve dretve iz reda blokiranih dretvi sinkronizacijskog objekta u red pripravnih dretvi tog objekta. Nakon što dretva koja je pozvala metodu Pulse ili PulseAll otpusti sinkronizacijski objekt, prva dretva iz reda pripravnih zauzima sinkronizacijski objekt. Metode Wait, Pulse i PulseAll moraju se pozivati iz kritičnog odsječka, inače dolazi do pojave pogreške.

6.1.7 Ostale posebnosti .NET Framework platforme

Prilikom izvođenja aritmetičkih operacija može doći do prekoračenja opsega ciljane varijable. .NET Framework platforma omogućuje provjeru rezultata aritmetičkih operacija korištenjem ključnih riječi checked i unchecked. Ključne riječi checked i unchecked odnose se na pojedini izraz, ili na cijeli blok. Ako se koristi ključna riječ checked, obavlja se provjera rezultata aritmetičkih operacija, i u slučaju prekoračenja opsega ciljane varijable prijavljuje se pogreška. Pogreška se prijavljuje prilikom prevođenja, ako je riječ o konstantnom izrazu, odnosno prilikom izvođenja ako izraz nije konstantan. Ako se koristi ključna riječ unchecked, nema provjere rezultata aritmetičkih operacija. Ako dođe do prekoračenja opsega ciljane varijable odbacuju se (eng. truncate) viši bitovi rezultata.

.NET Framework knjižnica osnovnih razreda sadrži razred System.Array. Razred System.Array definira neke korisne metode za rad s poljima, kao što su: Clear, Copy, Reverse, Length, Rank i druge. Metoda Clear postavlja niz elemenata polja na nulu, ili null referencu, ovisno o tipu polja. Metoda Copy preslikava dio polja u drugo polje i obavlja potrebne pretvorbe tipova. Metoda Reverse mijenja raspored elemenata u

jednodimenzionalnom polju od posljednjega prema prvom. Metoda Length vraća broj elemenata polja u svim dimenzijama, dok metoda Rank vraća broj dimenzija polja. Razred System.Array podržava i niz drugih metoda čiji se opis nalazi u dokumentaciji [24].

Svojstva (eng. properties) omogućuju kontrolirani pristup članovima s privatnim pravom pristupa korištenjem uobičajene sintakse za pristup članovima razreda. Svojstvo ne zauzima prostor u memoriji, nego samo pruža pristupne metode (eng. accessors) za kontrolirani pristup drugim članovima razreda. Za svako svojstvo se mogu definirati pristupne metode get i set. Pristupne metode get i set definiraju kod koji se izvodi pri čitanju, odnosno pisanju u svojstvo. Ako je ostvarena samo jedna od pristupnih metoda, svojstvo je moguće samo čitati, odnosno samo pisati u njega. Pristupna metoda get mora na kraju svoga izvođenja vratiti vrijednost tipa jednakog tipu svojstva, ili završiti bacanjem iznimke. Pristupna metoda set ne vraća nikakvu vrijednost, a zadanu vrijednost prima preko implicitnog parametra "value".

Za pristup standardnom ulazu i izlazu koriste se funkcionalnosti ostvarene u .NET Framework knjižnici osnovnih razreda. Takav pristup omogućuje formatiranje ulaznih i izlaznih vrijednosti u skladu s regionalnim postavkama računala na kojem se program izvodi. Npr. u nekim se državama za odvajanje decimalnih znamenki od cjelobrojnih koristi zarez, a točka za pregledniji zapis velikih brojeva, dok je u drugim državama obrnuto. Unos podataka koji ne odgovaraju definiranom standardu može uzrokovati pogreške pri izvođenju programa. Korištenjem knjižnice osnovnih razreda za ulazno-izlazne operacije izbjegavaju se moguće pogreške.

.NET Framework knjižnica osnovnih razreda podržava niz razreda i struktura podataka za pristup bazama podataka, direktoriju i drugim resursima računala. Dio tih razreda korišten je pri izradi programske potpore pristupu direktoriju.

6.2 Elementi knjižnice osnovnih razreda korišteni pri izradi programske potpore

.NET Framework knjižnica osnovnih razreda sadrži razrede i strukture podataka koje omogućuju pristup direktoriju. Za pristup direktoriju iz korisničkog programa potrebno je uključiti imenik System.DirectoryServices. Razredi ostvareni u ovom imeniku se u svome radu oslanjaju na ADSI programsko sučelje.

ADSI je skup otvorenih objektno orijentiranih programskih sučelja (API) za pristup direktoriju. ADSI nije ograničen samo na Active Directory, već omogućuje pristup direktorijima zasnovanima na LDAP protokolu, te NDS ili NTDS (NT Directory Service). NTDS je baza podataka korisnika u Windows NT 4.0 operacijskom sustavu. Korištenjem ADSI programskog sučelja moguće je iz jedne aplikacije pristupati zapisima u različitim direktorijima širom mreže.

Prilikom prvog pristupa zapisu u direktoriju, vrijednosti atributa zapisa spremaju se u ADSI priručnu memoriju (eng. ADSI cache, Property cache). ADSI priručna memorija omogućuje obavljanje nekoliko operacija nad zapisom lokalno, prije nego se izmjena prenese u direktorij. Time se značajno smanjuje mrežni promet i smanjuje opterećenje direktorija. ADSI priručna memorija nije uobičajena priručna memorija. Ako se unutar programa pristupa istom zapisu korištenjem dviju različitih referenci, svaka referenca će obavljati operacije nad zasebnom preslikom zapisa.

U narednim poglavljima opisani su članovi imenika System.DirectoryServices korišteni pri izradi programske potpore pristupu direktoriju.

6.2.1 Razred *DirectoryEntry*

Objekt razreda *DirectoryEntry* predstavlja zapis u direktoriju. Objekti ovog razreda koriste se za čitanje, izmjenu i brisanje zapisa, te premještanje zapisa unutar stabla direktorija. Pri izradi programske potpore pristupu direktoriju korišteni su sljedeći članovi razreda *DirectoryEntry*:

- **DirectoryEntry** – Konstruktor razreda je preopterećen, te može primiti različiti broj parametara. Moguće je proslijediti put do zapisa (eng. path), korisničko ime i lozinku korisnika koji pristupa direktoriju, te način autentikacije. Put do zapisa je znakovni niz sastavljen od imena protokola za pristup direktoriju, znakova "://" i apsolutnog imena zapisa. Tako je npr. put do zapisa korisnika s imenom Ivan Ivic "LDAP://CN=Ivan Ivic,CN=Users,CN=ETK,DC=research". Ako se ne proslijede korisničko ime i lozinka korisnika koji pristupa direktoriju, za autentikaciju se koristi korisničko ime i lozinka korisnika koji je trenutno prijavljen na sustav.
- **Exists** – Statička metoda koja provjerava da li postoji zapis sa zadanim putem. Metoda vraća vrijednost "istina" ako zapis postoji, u suprotnom vraća "laž".
- **Path** – Svojstvo koje omogućuje čitanje i izmjenu puta do zapisa koji objekt predstavlja.
- **UserName** – Svojstvo koje omogućuje čitanje i izmjenu korisničkog imena koje se koristi pri autentikaciji.
- **Password** – Svojstvo koje omogućuje čitanje i izmjenu lozinke koja se koristi pri autentikaciji.
- **Children** – Svojstvo koje sadrži objekt razreda *DirectoryEntries* koji predstavlja sve zapise neposredno podređene danom zapisu.
- **Parent** – Svojstvo koje sadrži objekt razreda *DirectoryEntry* koji predstavlja zapis neposredno nadređen danom zapisu.
- **Properties** – Svojstvo koje sadrži objekt razreda *PropertyCollection* koji predstavlja vrijednosti atributa dotičnog zapisa.

- **UsePropertyCache** – Svojstvo koje sadrži logičku vrijednost koja određuje da li se pri operacijama nad zapisom koristi ADSI priručna memorija. Vrijednost "istina" znači da se koristi ADSI priručna memorija, inače se sve operacije provode izravno nad zapisom u direktoriju. Pretpostavljena vrijednost je "istina".
- **CommitChanges** – Metoda koja omogućuje spremanje promjena nad zapisom u direktorij. Promjene obuhvaćaju izmjene, dodavanje i brisanje vrijednosti atributa, kao i dodavanje novog zapisa.
- **RefreshCache** – Metoda koja omogućuje dohvat vrijednosti iz direktorija u lokalnu presliku zapisa. Pozivom ove metode gube se sve eventualne izmjene nad zapisom izvedene nakon posljednjeg poziva CommitChanges metode. Metoda je preopterećena, te je kao parametar moguće proslijediti listu atributa čije vrijednosti se želi dohvatiti.
- **MoveTo** – Metoda koja omogućuje premještanje zapisa unutar informacijskog stabla direktorija. Parametar je objekt koji predstavlja novi neposredno nadređeni zapis. Ako se koristi ADSI priručna memorija potrebno je nakon poziva metode MoveTo pozvati metodu CommitChanges nad zadanim neposredno nadređenim zapisom, kako bi se promjene prenijele u direktorij.
- **DeleteTree** – Metoda koja briše zapis nad kojim je pozvana, i sve njemu podređene zapise. Brisanje se obavlja izravno nad direktorijem, odnosno ne koristi se ADSI priručna memorija.

6.2.2 Razred DirectoryEntries

Objekt razreda DirectoryEntries predstavlja sve zapise neposredno podređene danom zapisu u direktoriju. Metode **Add** i **Remove** omogućuju dodavanje, odnosno brisanje zapisa iz stabla direktorija.

Metoda **Add** prima kao parametre relativno ime zapisa koji se dodaje, te LDAP ime razreda zapisa, kao što je definirano u shemi. Metoda vraća objekt razreda DirectoryEntry koji predstavlja novostvoreni objekt. Da bi se dodani zapis upisao u direktorij potrebno je pozvati metodu CommitChanges nad objektom koji vraća metoda Add. Prije toga potrebno je postaviti vrijednosti atributa zapisa koji se u shemi nalaze navedeni u atributima mustContain i systemMustContain. Ako se ne postave sve potrebne vrijednosti atributa, pri pozivu CommitChanges metode nad objektom doći će do pogreške.

Metoda **Remove** prima kao parametar objekt tipa DirectoryEntry koji predstavlja zapis koji treba izbrisati iz direktorija. Zapis koji se briše ne smije biti spremnik ili, ako je spremnik, ne smije sadržavati druge zapise. Za brisanje spremnika sa svim zapisima koje sadrži, koristi se metoda DeleteTree razreda DirectoryEntry. Metoda Remove briše zapis izravno iz direktorija, odnosno ne koristi se ADSI priručna memorija.

6.2.3 Razredi *PropertyCollection* i *PropertyValueCollection*

Objekt razreda *PropertyCollection* predstavlja vrijednosti svih atributa nekog zapisa u direktoriju (eng. properties). Vrijednosti atributa su indeksirane preko imena atributa. Pojedinoj vrijednosti atributa moguće je pristupiti korištenjem operatora [] za pristup članovima polja. Unutar uglatih zagrada potrebno je navesti LDAP ime atributa. Npr. Ako je *UserEntry* objekt razreda *DirectoryEntry*, onda *UserEntry.Properties["ETKUserID01"]* dohvaća vrijednosti atributa čije LDAP ime je *ETKUserID01*.

Vrijednosti atributa predstavljene su objektom razreda *PropertyValueCollection*. Objekt razreda *PropertyValueCollection* sadrži svojstvo koje omogućuje čitanje i izmjenu vrijednosti atributa korištenjem operatora [] za pristup članovima polja. Unutar uglatih zagrada navodi se indeks vrijednosti atributa kojoj se pristupa. U navedenom primjeru *UserEntry.Properties["ETKUserID01"][0]* je objekt koji predstavlja prvu vrijednost atributa čije LDAP ime je *ETKUserID01*. Pri izradi programske potpore pristupu direktoriju korišteni su sljedeći članovi razreda *PropertyValueCollection*:

- **Count** – Svojstvo koje sadrži broj vrijednosti atributa u objektu.
- **Add** – Metoda za dodavanje nove vrijednosti atributa. Parametar je znakovni niz s vrijednošću atributa koja se želi dodati.
- **Remove** – Metoda za brisanje određene vrijednosti atributa. Parametar je znakovni niz s vrijednošću atributa koja se želi izbrisati.
- **Clear** – Metoda za brisanje svih vrijednosti atributa.

Metode *Add*, *Remove* i *Clear* obavljaju izmjene nad zapisima u ADSI priručnoj memoriji. Da bi se izmjene prenijele u direktorij potrebno je pozvati metodu *CommitChanges* nad objektom koji predstavlja dotični zapis u direktoriju.

6.2.4 Razred *DirectorySearcher*

Razred *DirectorySearcher* omogućuje pretraživanje direktorija. Pretraživanje se obavlja prema zadanom kriteriju. Osim kriterija pretrage, moguće je zadati parametre koji određuju koji dio stabla direktorija se pretražuje, koliko najviše zapisa pretraga vraća, koliko dugo pretraga smije trajati, i druge. Pri izradi programske potpore pristupu direktoriju korišteni su sljedeći članovi razreda *DirectorySearcher*:

- **DirectorySearcher** – Konstruktor razreda je preopterećen, te može primiti različiti broj parametara. Moguće je proslijediti parametre koji određuju dio stabla koji se pretražuje, kriterij pretrage, te attribute čije vrijednosti se vraćaju kao rezultati pretrage.
- **SearchRoot** – Svojstvo koje određuje zapis u direktoriju od kojega počinje pretraga. Pretpostavljena vrijednost je null referenca, što znači da se pretražuju svi zapisi u domeni.

- **SearchScope** – Svojstvo koje određuje dio stabla koji se pretražuje. Sadrži jednu od tri vrijednosti definiranih u pobrojanom tipu SearchScope: Base, OneLevel i Subtree. Vrijednost Base označava da se pretražuje samo zapis zadan svojstvom SearchRoot, i rezultat pretrage je najviše jedan zapis. Vrijednost OneLevel označava da se pretražuju samo zapisi koji su neposredno podređeni zapisu zadanom svojstvom SearchRoot, ne uključujući taj zapis. Vrijednost Subtree označava da se pretražuje cijelo podstablo, uključujući zadani zapis i sve zapise njemu podređene. Pretpostavljena vrijednost je Subtree
- **Filter** – Svojstvo koje određuje kriterij pretrage. Za kriterij pretrage mogu se koristiti vrijednosti atributa i relacijski operatori <, <=, =, >=, >. Za složene uvjete pretrage mogu se koristiti logički operatori & i | u prefiks notaciji. Kriterij pretrage je znakovni niz koji mora biti omeđen zagrada. Primjer kriterija pretrage koji pronalazi sve zapise o korisnicima čije ime započinje slovom F je "(&(ETKUserID01=F*)(objectClass=ETKUser01))". Ako se ne navede kriterij pretrage, koristi se pretpostavljena vrijednost "(objectClass=*)", čime se dohvaćaju svi zapisi.
- **PropertiesToLoad** – Svojstvo koje sadrži LDAP imena atributa čije se vrijednosti dohvaćaju prilikom pretrage. Popis je sadržan u objektu razreda StringCollection. Imena atributa se dodaju korištenjem metode Add nad tim objektom. Pretpostavljena vrijednost je null referenca, čime se dohvaćaju samo atributi path i name.
- **CacheResults** – Svojstvo koje određuje da li se rezultati pretrage privremeno spremaju u lokalnu memoriju. Pretpostavljena vrijednost je "istina", što znači da se rezultati pretrage privremeno spremaju u lokalnu memoriju.
- **PageSize** – Svojstvo koje određuje veličinu stranice pri pretraživanju po stranicama (eng. paged search). Veličina stranice određuje broj zapisa, rezultata pretrage, koji se dohvaćaju odjednom. Nakon što je domenski zastupnik pronašao zadani broj zapisa, pretraga se zaustavlja. Ako korisnički program zahtijeva dohvat većeg broja zapisa, pretraga se nastavlja. Pretpostavljena vrijednost je 0, što znači da se ne pretražuje po stranicama. Ako je veličina stranice veća od vrijednosti u svojstvu SizeLimit, vraća se stranica veličine SizeLimit.
- **SizeLimit** – Svojstvo koje određuje maksimalni broj zapisa, rezultata pretrage koje domenski zastupnik može vratiti prilikom pretrage. Pretpostavljena vrijednost je 0, što znači da se koristi pretpostavljena vrijednost za domenske zastupnike (eng. server-determined default) od 1000 zapisa. Ako je zadana vrijednost veća od pretpostavljene, koristi se pretpostavljena vrijednost za domenske zastupnike.

- **ClientTimeOut** – Svojstvo koje određuje koliko dugo korisnički program čeka rezultate pretrage od domenskog zastupnika. Ako domenski zastupnik ne odgovori na zahtjev unutar danog vremena pretraga se prekida i ne vraćaju se nikakvi rezultati. Pretpostavljena vrijednost je -1, što znači da nema vremenskog ograničenja na pretragu. Ako prije isteka zadanog vremena istekne vremenski interval zadan svojstvom `ServerTimeLimit`, domenski zastupnik vraća rezultate pretrage dobivene do tog trenutka.
- **ServerTimeLimit** – Svojstvo koje određuje maksimalno vrijeme trajanja pretrage na strani domenskog zastupnika. Pretpostavljena vrijednost je -1, što znači da se koristi pretpostavljena vrijednost za domenske zastupnike od 120 sekundi. Ako je zadana vrijednost veća od pretpostavljene, koristi se pretpostavljena vrijednost za domenske zastupnike.
- **FindOne** – Metoda koja pokreće pretragu i vraća samo prvi pronađeni zapis. Vraćeni zapis je objekt razreda `SearchResult`. Ako nije pronađen niti jedan zapis koji zadovoljava zadane uvjete pretrage vraća se null referenca.
- **FindAll** – Metoda koja pokreće pretragu i vraća objekt razreda `SearchResultCollection`, koji predstavlja pronađene zapise.

6.2.5 Razredi `SearchResultCollection` i `SearchResult`

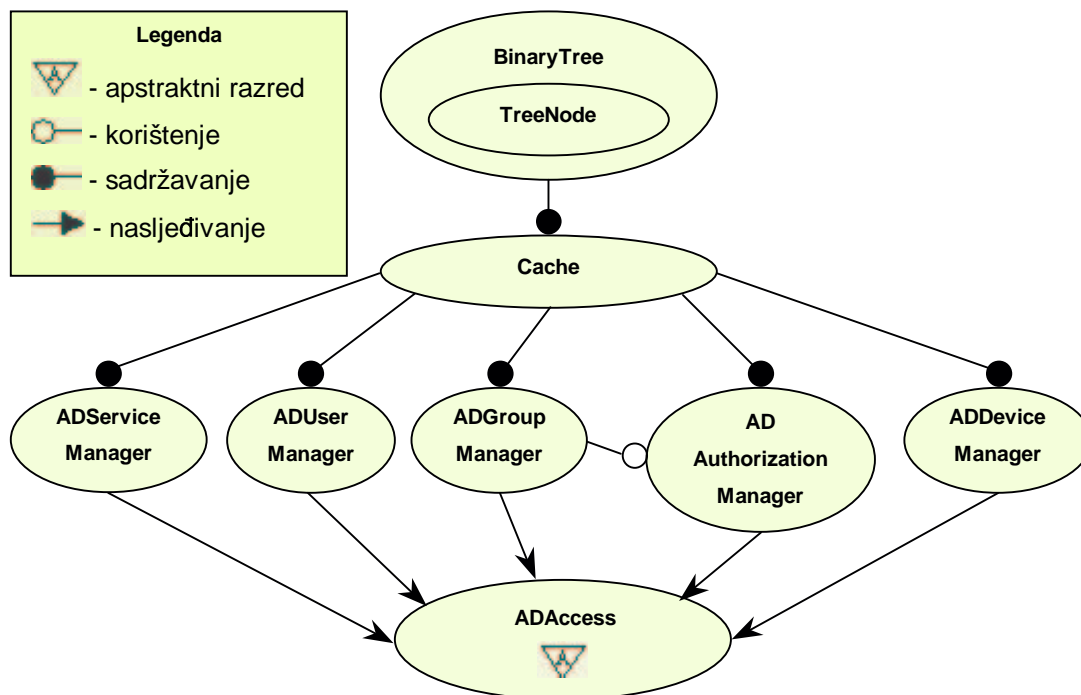
Objekt razreda `SearchResultCollection` sadrži rezultate pretrage vraćene pri pozivu metode `FindAll` nad objektom razreda `DirectorySearcher`. Objekti razreda `SearchResultCollection` sadrže svojstvo `Count`, koje sadrži broj zapisa koji su pronađeni tijekom pretrage. Ako `Count` sadrži vrijednost 0, znači da pretraga nije rezultirala niti jednim zapisom koji zadovoljava zadane uvjete pretrage. Razred `SearchResultCollection` omogućuje iteriranje kroz pojedine rezultate korištenjem naredbe `foreach`. Pojedini rezultati su predstavljeni objektima razreda `SearchResult`.

Objekti razreda `SearchResult` slični su objektima razreda `DirectoryEntry`. Osnovna je razlika u tome što objekti razreda `SearchResult` sadrže vrijednosti samo onih atributa koji su zadani u `PropertiesToLoad` svojstvu objekta razreda `DirectorySearcher` prilikom pretrage. Od članova navedenih u poglavlju 6.2.1, razred `SearchResult` sadrži samo `Path` i `Properties`. Razred `SearchResult` sadrži metodu `GetDirectoryEntry` koja vraća objekt razreda `DirectoryEntry` koji predstavlja dotični zapis u direktoriju. Nad dobivenim zapisom moguće je izvoditi sve operacije definirane članovima razreda `DirectoryEntry`.

6.3 Hijerarhija razreda

Ostvarena programska potpora podijeljena je na logičke cjeline koje upravljaju određenom vrstom podataka. Logička podjela se odražava na programski kod kroz

hijerarhiju razreda. U programskom kodu su ostvareni sljedeći razredi: ADAccess, ADGroupManager, ADUserManager, ADServiceManager, ADAuthorizationManager, ADDeviceManager, Cache, TreeNode i BinaryTree. Hijerarhija razreda prikazana je na slici 6-3.



Slika 6-3 Hijerarhija razreda programske potpore pristupu direktoriju

Razred ADAccess ostvaruje osnovnu funkcionalnost pristupa direktoriju. Razredi ADGroupManager, ADUserManager, ADServiceManager, ADAuthorizationManager i ADDeviceManager su razredi za upravljanje podacima. Svi razredi za upravljanje podacima nasljeđuju apstraktni razred ADAccess.

Korisnik je autoriziran za neku operaciju ako je autorizacija dodijeljena izravno tom korisniku, ili ako je autorizacija dodijeljena nekoj od grupa kojoj korisnik pripada. Razred ADAuthorizationManager koristi objekt razreda ADGroupManager za određivanje pripadnosti korisnika i usluga grupama.

Razred Cache ostvaruje funkcionalnosti priručne memorije za čitanje. Razred Cache sadrži objekt razreda BinaryTree. Razred BinaryTree ostvaruje binarno stablo i dvostruko povezanu listu, potrebne za rad priručne memorije. Razred BinaryTree sadrži ugniježđeni razred TreeNode koji predstavlja jedan zapis u priručnoj memoriji.

Prikazani razred dijagrama zbog jednostavnosti ne sadrži podatkovne razrede, koji se koriste kao strukture podataka za prijenos parametara metodama razreda za upravljanje podacima. To su razredi: sUser, cApplication, cService, ServiceProperties, cFunction,

cFunctionPath, cDevice i cUserDataAuthorization. Navedeni razredi sadrže vrijednosti atributa zapisa o korisnicima, aplikacijama, uslugama, funkcijama, korisničkim uređajima i pravima pristupa podacima o korisniku. Detaljan opis funkcionalnosti svih navedenih razreda dan je u narednim poglavljima.

6.3.1 Razredi za upravljanje podacima

ADAccess je apstraktni razred koji ostvaruje osnovnu funkcionalnost pristupa direktoriju. Konstruktor razreda čita ime domene iz zapisa karakterističnog za sustavsku komponentu direktorija. Ime domene je dio apsolutnog imena zapisa i koristi se pri dohvat zapisa iz direktorija. Čitanje imena domene iz zapisa karakterističnog za sustavsku komponentu direktorija omogućuje nezavisnost programske potpore pristupu direktoriju o domeni u kojoj se izvodi korisnički program. Konstruktor provjerava postojanje spremnika s relativnim imenom "CN=ETK". Ako taj spremnik ne postoji, stvara se novi. Ako objekt s relativnim imenom "CN=ETK" postoji, ali nije spremnik, prijavljuje se pogreška. Konstruktor razreda ADAccess je preopterećen, te dozvoljava prijenos parametra koji određuje apsolutno ime spremnika podataka. Zadani zapis mora postojati u direktoriju i mora biti spremnik, inače se prijavljuje pogreška.

Razredi za upravljanje podacima nasljeđuju razred ADAccess. Svi konstruktori razreda za upravljanje podacima su istog oblika i preopterećeni. Konstruktori mogu primiti tri ili četiri parametra. Parametri određuju da li se koristi priručna memorija, veličinu priručne memorije u broju zapisa, i vrijeme čuvanja zapisa u priručnoj memoriji u sekundama. Dodatno se može zadati apsolutno ime spremnika, koje se prosljeđuje konstruktoru osnovnog razreda.

Razred ADGroupManager

Razred ADGroupManager ostvaruje deset metoda za upravljanje podacima o grupama korisnika i usluga. Metode omogućuju stvaranje novih i brisanje postojećih grupa, te dodavanje, brisanje i pregled članova grupa. Članovi grupe mogu biti korisnici, usluge i druge grupe. Grupa ne može sadržavati samu sebe, niti ijednu od grupa koje je sama član, izravno ili neizravno. Identifikator grupe ne smije biti jednak nekom od identifikatora korisnika ili usluge. Grupama se mogu dodavati samo oni članovi koji postoje u direktoriju. Prilikom brisanja zapisa o grupi brišu se i podaci o pripadnosti te grupe drugim grupama, te autorizacijska prava pristupa uslugama i korisničkim podacima za zadanu grupu.

Razred ADUserManager

Razred ADUserManager ostvaruje trideset dvije metode za upravljanje podacima o korisnicima. Razred ostvaruje metode za dodavanje novih, te dohvat, izmjenu i brisanje podataka o postojećim korisnicima. Identifikator korisnika koji se dodaje u direktorij ne smije biti jednak nekom od identifikatora usluge ili grupe. Zbog načina imenovanja zapisa s

adresom, definiranog u poglavlju 5.3.2, identifikatori korisnika, usluga i grupa, te poštanski brojevi ne smiju sadržavati podniz "\$\$\$", inače se prijavljuje pogreška. Prilikom dodavanja zapisa o adresi korisnika, u spremniku Cities mora postojati zapis o gradu sa danim poštanskim brojem. Prilikom brisanja zapisa o korisniku brišu se i podaci o pripadnosti korisnika grupama, te autorizacijska prava pristupa uslugama i korisničkim podacima za dotičnog korisnika.

Razred ADServiceManager

Razred ADServiceManager ostvaruje trideset pet metoda za upravljanje podacima o uslugama, aplikacijama koje ostvaruju usluge, te funkcijama koje usluge podržavaju. Razred ostvaruje niz metoda koje omogućuju dodavanje novih, te dohvat, izmjenu i brisanje zapisa o postojećim aplikacijama, uslugama i funkcijama. Da bi se u direktorij dodao zapis o usluzi, u direktoriju mora postojati zapis o aplikaciji koja ostvaruje uslugu. Da bi se u direktorij dodali zapisi o funkcijama usluge, u direktoriju mora postojati zapis o dotičnoj usluzi. Identifikator usluge ne smije biti jednak nekom od identifikatora korisnika ili grupe. Identifikator usluge mora biti jedinstven u domeni, odnosno u različitim aplikacijama nije moguće ostvariti usluge s jednakim identifikatorom usluge. Brisanjem zapisa o aplikaciji brišu se svi zapisi o uslugama ostvarenim unutar dotične aplikacije. Brisanjem zapisa o usluzi brišu se svi zapisi o funkcijama koje usluge podržavaju, kao i pripadnosti usluge grupama, te autorizacijska prava pristupa uslugama i korisničkim podacima za uslugu koja se briše.

Razred ADAuthorizationManager

Razred ADAuthorizationManager ostvaruje dvadeset tri metode za upravljanje podacima o pravima korištenja usluga i pravima pristupa podacima o korisnicima. Ostvarene metode omogućuju definiranje novih, te pregled izmjenu i brisanje postojećih autorizacija. Entitet za koji se definira nova autorizacija mora postojati u direktoriju. Entitet je autoriziran za neku operaciju ako je autorizacija dodijeljena izravno tom autoritetu, ili je autorizacija dodijeljena grupi kojoj entitet pripada izravno ili neizravno. Zbog toga neke metode koriste razred ADGroupManager za dobivanje podataka o pripadnosti entiteta grupama.

Razred ADDeviceManager

Razred ADDeviceManager ostvaruje šest metoda za upravljanje podacima o korisničkim uređajima. Metode omogućuju dodavanje novih, te pregled i izmjenu postojećih podataka o korisničkim uređajima.

U ovom poglavlju dan je samo kratki pregled metoda ostvarenih u razredima za upravljanje podacima. Deklaracije svih postojećih metoda, s detaljnim opisom parametara i funkcionalnosti nalaze se na priloženom mediju u datoteci Interface.txt.

6.3.2 Podatkovni razredi

Podatkovni razredi koriste se za prijenos većeg broja parametara metodama razreda za upravljanje podacima. Podatkovni razredi sadrže vrijednosti atributa zapisa u direktoriju. Podatkovni razredi su: sUser, cApplication, cService, ServiceProperties, cFunction, cFunctionPath, cDevice i cUserDataAuthorization.

Objekt razreda sUser sadrži osnovne podatke o korisniku: ime korisnika, identifikator korisnika, lozinku i jedinstveni matični broj građana. Konstruktor razreda postavlja vrijednosti svih članova na prazan niz.

Objekt razreda cApplication sadrži podatke o aplikaciji: identifikator, ime i opis aplikacije, vlasnika aplikacije, i mrežnu adresu (eng. Uniform Resource Locator – URL) aplikacije. Konstruktor razreda postavlja vrijednosti svih članova na prazan niz.

Objekt razreda cService sadrži podatke o usluzi: identifikator aplikacije koja ostvaruje uslugu, identifikator, ime, lozinka i opis usluge, mrežnu adresu usluge, akciju SOAP (eng. Simple Object Access Protocol) protokola, lokaciju WSDL (eng. Web Services Description Language) datoteke, i logičku vrijednost koja određuje da li je za korištenje usluge potrebna autorizacija. Konstruktor razreda postavlja vrijednosti svih znakovnih nizova na prazan niz i logičku vrijednost na "laž".

Objekt razreda ServiceProperties sadrži neke podatke o usluzi: identifikator aplikacije koja ostvaruje uslugu, identifikator i lozinku usluge, te logičke vrijednosti koje određuju da li je usluga registrirana u sustavu, da li je usluga mrežna usluga, i treba li za njeno korištenje autorizacija. Konstruktor razreda postavlja vrijednosti svih znakovnih nizova na prazan niz, a logičke vrijednosti na "laž".

Objekt razreda cFunction sadrži podatke o funkciji usluge: identifikator aplikacije, usluge i funkcije, ime i opis funkcije, te ulazne, izlazne i posebne parametre funkcije. Konstruktor razreda postavlja vrijednosti svih znakovnih nizova na prazan niz. Identifikator funkcije je cjelobrojna vrijednost i konstruktorom se postavlja na vrijednost nula.

Objekt razreda cFunctionPath sadrži identifikatore aplikacije, usluge i funkcije potrebne za stvaranje puta do zapisa o funkciji. Konstruktor razreda postavlja vrijednosti identifikatora aplikacije i usluge na prazan niz, a identifikator funkcije na nulu.

Objekt razreda cDevice sadrži podatke o korisničkom uređaju: identifikator, IP adresu i tip korisničkog uređaja, te cjelobrojne vrijednosti horizontalne i vertikalne razlučivosti i broj bita za boju. Konstruktor razreda postavlja vrijednosti znakovnih nizova na prazan niz, a cjelobrojne vrijednosti na nulu.

Objekt razreda cUserDataAuthorization sadrži podatke o pravu pristupa korisničkim podacima: identifikator korisnika, identifikator entiteta (korisnika, usluge ili grupe) koji zahtijeva pristup korisničkim podacima, te logičke vrijednosti prava pristupa podacima o korisničkom imenu, jedinstvenom matičnom broju građana, gradu u kojem korisnik živi, korisničkim adresama, elektroničkim adresama, brojevima telefona i pripadnosti korisnika

grupama. Konstruktor razreda postavlja vrijednosti znakovnih nizova na prazan niz, a logičke vrijednosti na "laž".

6.3.3 *Razred Cache*

Razred Cache ostvaruje funkcionalnosti upravljanja priručnom memorijom. Upravljanje priručnom memorijom izvodi se korištenjem ostvarenih metoda za dobavljanje zapisa iz priručne memorije, dodavanje zapisa u priručnu memoriju i brisanje zapisa iz priručne memorije.

Privatna metoda Refresh se izvodi kao zasebna pozadinska dretva. Metoda Refresh briše zapise iz priručne memorije nakon isteka zadanog intervala. Ostatak vremena dretva je neaktivna, odnosno ne troši procesorsko vrijeme.

Konstruktor razreda prihvaća dva parametra: veličinu priručne memorije i vrijeme čuvanja zapisa u priručnoj memoriji. Veličina priručne memorije je broj koji određuje maksimalni broj zapisa koji se čuvaju u priručnoj memoriji. Vrijeme čuvanja zapisa u priručnoj memoriji određuje maksimalni vremenski interval koji se zapisi čuvaju u priručnoj memoriji. Nakon isteka zadanog vremena zapis se briše iz priručne memorije. Pri sljedećem pokušaju dohvata, zapis neće biti pronađen u priručnoj memoriji, te će se iz direktorija dohvatiti zapis koji sadrži nove vrijednosti atributa. Ako se prilikom dodavanja novog zapisa u priručnu memoriju ustanovi da priručna memorija sadrži maksimalni broj zapisa, briše se prvi zapis iz liste i dodaje se novi zapis. Zapisi u listi sortirani su prema vremenu dodavanja zapisa u priručnu memoriju. Prvi zapis u listi prvi je na redu za brisanje iz priručne memorije (eng. First In First Out – FIFO) nakon isteka zadanog intervala. Brisanjem prvog zapisa iz liste, prilikom dodavanja novog zapisa, izbrisan je zapis koji bi ionako bio prvi izbrisan od strane pozadinske dretve. Zapisi iz priručne memorije se nikada ne upisuju u direktorij, jer je priručna memorija predviđena isključivo za čitanje.

Priručna memorija mora ispravno raditi u uvjetima kada je koristi više dretvi. Operacije nad priručnom memorijom izvode se prema protokolu koji omogućuje istovremeno čitanje podataka od strane više dretvi. Pisanje u priručnu memoriju smije obavljati samo jedna dretva, i za to vrijeme niti jedna druga dretva ne smije čitati iz priručne memorije. Isto pravilo vrijedi i za dretvu koja briše zapise iz priručne memorije. Za potrebe programske potpore pristupu direktoriju razvijen je protokol za pristup priručnoj memoriji. Pseudokod protokola prikazan je na slici 6-4. Prikazani protokol razvijen je po uzoru na dvofazno izvođenje (2-phase commit) u sustavima za upravljanje bazama podataka (eng. Database Management System – DBMS).

```
Čitanje
{
    zaključaj (s1)
    {
        dok (broj_pisača > 0) čekaj_monitor (s1);
        broj_čitača++;
    }
    Čitaj_iz_priručne_memorije ();
    zaključaj (s1)
    {
        zaključaj (s2)
        {
            broj_čitača--;
            ako (broj_čitača == 0)
                pošalji_poruku_svima_na_monitoru (s2);
        }
    }
}
```

```
Pisanje
{
    zaključaj (s1)
    {
        broj_pisača++;
    }
    zaključaj (s2)
    {
        dok (broj_čitača > 0) čekaj_monitor (s2);
        Piši_u_priručnu_memoriju ();
    }
    zaključaj (s1)
    {
        broj_pisača--;
        ako (broj_pisača == 0)
            pošalji_poruku_svima_na_monitoru (s1);
    }
}
```

Slika 6-4 Pseudokod protokola za pristup priručnoj memoriji

Protokol koristi četiri pomoćne varijable: s1, s2, broj_čitača i broj_pisača. Varijable s1 i s2 označavaju objekte za sinkronizaciju. Varijable broj_čitača i broj_pisača sadrže broj dretvi koje žele čitati, odnosno pisati u priručnu memoriju.

Pristup priručnoj memoriji ostvaruje se u tri koraka: najava pristupa, obavljanje operacije čitanja, odnosno pisanja, i odjava pristupa. Najava pristupa obavlja se uvećavanjem varijable broj_čitača za dretve koje žele čitati iz priručne memorije, odnosno broj_pisača za dretve koje žele pisati u priručnu memoriju. S obzirom da dretve koje žele čitati iz priručne memorije nakon najave pristupa odmah prelaze na čitanje, najava čitanja se može obaviti samo ako ne postoji niti jedna dretva koja je najavila pisanje. Dretva koja želi pisati u priručnu memoriju mora dobiti ekskluzivno pravo nad objektom za sinkronizaciju s2, čime se osigurava da u

svakom trenutku samo jedna dretva piše u priručnu memoriju. Prije samog pisanja u priručnu memoriju, dretva mora pričekati da sve dretve koje čitaju iz priručne memorije završe s čitanjem. Po završetku pristupa priručnoj memoriji dretve odjavljuju pristup umanjivanjem odgovarajuće varijable. Ako se pri tome varijabla smanji na vrijednost nula, dretva signalizira sve ostale dretve na odgovarajućem monitoru da mogu nastaviti s pristupom.

Sve dretve pri najavi pristupa čekaju u redu sinkronizacijskog objekta s1, bez obzira da li se radi o zahtjevu za čitanje ili pisanje. Time se postiže da se zahtjevi za pristup priručnoj memoriji obrađuju onim redoslijedom kojim pristižu (FIFO).

Sve metode ostvarene u razredu Cache koje pristupaju priručnoj memoriji poštuju opisani protokol. Korisnički program koji koristi navedene metode ne mora ostvariti opisani protokol, niti brinuti o njegovoj primjeni.

6.3.4 Razredi *TreeNode* i *BinaryTree*

Zapisi se u u priručnu memoriju spremaju u obliku binarnog stabla sortiranog prema putu do zapisa, i dvostruko povezane liste sortirane prema vremenu dodavanja zapisa u priručnu memoriju. Razred *TreeNode* predstavlja jedan zapis u priručnoj memoriji. Struktura zapisa u priručnoj memoriji prikazana je na slici 6-5.

Referenca na objekt razreda <i>DirectoryEntry</i>
Vrijeme kada je zapis dodan u listu
Referenca na lijevo dijete u binarnom stablu
Referenca na desno dijete u binarnom stablu
Referenca na sljedeći zapis u listi
Referenca na prethodni zapis u listi

Slika 6-5 Struktura zapisa u priručnoj memoriji

Zapisi se čuvaju u obliku referenci na objekt razreda *DirectoryEntry*. Prilikom čitanja podataka iz direktorija stvara se lokalna preslika zapisa u ADSI priručnoj memoriji. Referenca na objekt razreda *DirectoryEntry* referencira zapis u ADSI priručnoj memoriji, pa je za pristup zapisu dovoljno čuvati referencu na dotični objekt.

Zapis u priručnoj memoriji sadrži varijablu s oznakom vremena kada je dotični zapis dodan u priručnu memoriju. Oznaka vremena koristi se kako bi se nakon isteka zadanog vremenskog intervala mogla dohvatiti nova vrijednost iz direktorija.

Reference na lijevo i desno dijete binarnog stabla, te na prethodni i sljedeći zapis u listi služe za održavanje strukture binarnog stabla, odnosno dvostruko povezane liste.

Razred `BinaryTree` ostvaruje metode za upravljanje binarnim stablom i dvostruko povezanom listom. To su metode za dodavanje, pronalazak i brisanje zapisa iz stabla i dvostruko povezane liste.

Pri izradi programske potpore pristupu direktoriju posebna pozornost je posvećena svojstvima sustava. U narednom poglavlju prikazani su rezultati mjerenja svojstava sustava provedeni na stvarnom sustavu.

7 Mjerenja svojstava programske potpore pristupu direktoriju

Svojstva programske potpore pristupu direktoriju ovise o više parametara. Kako bi se utvrdila ovisnost svojstava sustava o parametrima sustava, provedena su mjerenja svojstava tipičnih funkcija. Mjerenja su provedena na stvarnom sustavu slučajno odabranim upitima.

7.1 Opis mjerenja i mjernog sustava

Razredi za upravljanje podacima ostvaruju brojne metode za upravljanje podacima u direktoriju. Mjerenje svojstava svih ostvarenih metoda bio bi dugotrajan postupak. Potrebno je izdvojiti metode koje se pozivaju češće od ostalih i tako značajno utječu na svojstva sustava zajedničkih funkcija.

Svi korisnički zahtjevi za korištenjem usluga prolaze kroz sučelje za obradu zahtjeva i praćenje korištenja sustava zajedničkih funkcija. Korisnički zahtjev se sastoji od identifikatora korisnika, identifikatora aplikacije koja ostvaruje traženu uslugu, identifikatora usluge, i imena funkcije koju usluga ostvaruje. Za uspješan poziv zahtjevane usluge potrebno je poznavati podatke o usluzi i funkciji, te da li je korisnik autoriziran za korištenje tražene funkcije. Za dobivanje navedenih podataka koriste se metode `GetService`, `GetFunctionIDByIDs` i `GetFunctionFormatting` razreda `ADServiceManager`, i metoda `GetAuthorization` razreda `ADAuthorizationManager`.

Metoda `GetService` razreda `ADServiceManager` prima identifikator aplikacije koja ostvaruje uslugu, i identifikator zahtjevane usluge. Metoda vraća objekt razreda `cService` koji sadrži podatke o zahtjevanoj usluzi. Metoda `GetService` u svome radu koristi priručnu memoriju za čitanje.

Metoda `GetFunctionIDByIDs` razreda `ADServiceManager` prima identifikator aplikacije koja ostvaruje uslugu, identifikator usluge, i ime funkcije usluge koja se poziva. Metoda vraća identifikator funkcije čije ime je proslijeđeno kao parametar. Metoda `GetFunctionIDByIDs` u svome radu ne koristi priručnu memoriju za čitanje, jer ne postoje svi podaci potrebni za stvaranje puta do zapisa.

Metoda `GetFunctionFormatting` razreda `ADServiceManager` prima identifikator aplikacije koja ostvaruje uslugu, identifikator usluge, i identifikator funkcije usluge koja se poziva. Metoda vraća znakovni niz sa ulaznim parametrima funkcije čiji je identifikator proslijeđen kao parametar. Metoda `GetFunctionFormatting` u svome radu koristi priručnu memoriju za čitanje.

Metoda `GetAuthorization` razreda `ADAuthorizationManager` prima identifikator aplikacije koja ostvaruje uslugu, identifikator usluge, identifikator funkcije usluge, i identifikator

korisnika čije pravo korištenja funkcije se ispituje. Metoda vraća logičku vrijednost koja određuje je li zadani korisnik autoriziran za korištenje zadane funkcije. Metoda `GetAuthorization` u svome radu ne koristi priručnu memoriju za čitanje, s obzirom da metoda ne čita podatke iz direktorija, već samo provjerava postojanje zapisa u direktoriju.

Navedene metode se pozivaju pri svakom zahtjevu za korištenjem usluge za koju je potrebna autorizacija. Svi zahtjevi za korištenjem usluga prolaze kroz sučelje za obradu zahtjeva i praćenje korištenja. Svojstva navedenih metoda značajno utječu na svojstva programske potpore pristupu direktoriju.

Mjerenja svojstava navedenih metoda izvedena su korištenjem slučajno odabranih upita. Prije početka mjerenja stvorene su dvije pomoćne datoteke sa slučajno odabranim zahtjevima. Prva datoteka sadrži 50.000 slučajno odabranih zahtjeva, i koristi se za mjerenje svojstava metoda `GetService` i `GetFunctionFormatting`. Druga datoteka sadrži 5.000 slučajno odabranih zahtjeva, i koristi se za mjerenje svojstava metode `GetAuthorization`.

Program za mjerenje svojstava programske potpore pristupu direktoriju učitava zahtjeve iz odgovarajuće datoteke u memoriju, izvodi pozive funkcija, i mjeri vrijeme potrebno za izvođenje pojedinih metoda. Izvedena su dva skupa mjerenja: mjerenje ovisnosti vremena izvođenja metoda `GetService` i `GetFunctionFormatting` o parametrima priručne memorije, i mjerenje ovisnosti vremena izvođenja metode `GetAuthorization` o broju autoriziranih korisnika.

Metode `GetService` i `GetFunctionFormatting` u svome radu koriste priručnu memoriju za čitanje. Pri mjerenju svojstava metoda `GetService` i `GetFunctionFormatting` mijenjani su parametri priručne memorije: veličina priručne memorije i vrijeme čuvanja zapisa u priručnoj memoriji. Rezultati mjerenja pokazuju ovisnost vremena izvođenja navedenih metoda o parametrima priručne memorije.

Mjerenja pokazuju da je vrijeme izvođenja Metode `GetAuthorization` značajno veće od vremena izvođenja ostalih funkcija. Metoda `GetAuthorization` u svome radu ne koristi priručnu memoriju za čitanje, pa parametri priručne memorije ne utječu na vrijeme izvođenja metode. Pri mjerenju svojstava metode mijenjan je broj autoriziranih korisnika. Korisnici su autorizirani posredno, kroz pripadnost autoriziranim grupama. Rezultati mjerenja pokazuju ovisnost vremena izvođenja metode `GetAuthorization` o broju autoriziranih korisnika.

Mjerenja su izvedena na sustavu sa jednim domenskim zastupnikom. Domenski zastupnik je zasnovan na Pentium III mikroprocesoru radne frekvencije 733 MHz, 128 MB memorije i Windows 2000 Advanced Server operacijskim sustavom. Program za mjerenje svojstava izvodi se na domenskom zastupniku. U direktorij su spremljeni zapisi s podacima o 20.000 korisnika i 10 aplikacija. Svaka aplikacija sadrži 5 usluga, a svaka usluga sadrži 5 funkcija. Korisnici su raspoređeni u 10 grupa, u svakoj po 2.000 korisnika. U početku su sve grupe autorizirane za korištenje svih funkcija. Prilikom mjerenja svojstava metode `GetAuthorization` grupe su uklanjane iz liste autoriziranih grupa, jedna po jedna. Rezultati mjerenja prikazani su u narednom poglavlju.

7.2 Rezultati i analiza rezultata mjerenja

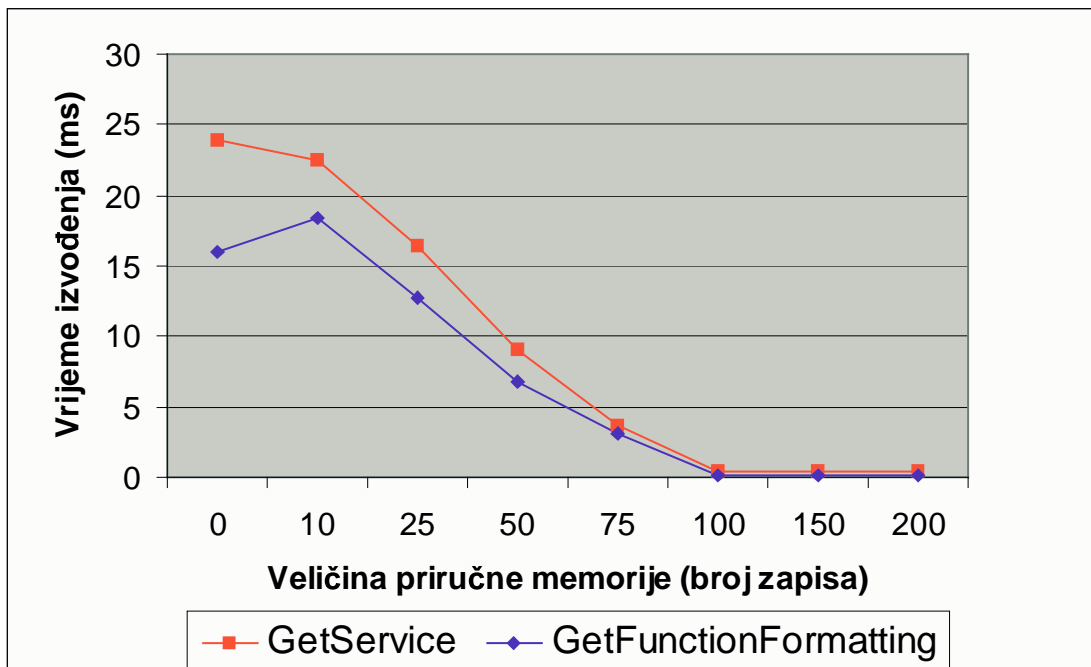
Prvi skup mjerenja obuhvaća mjerenje ovisnosti prosječnog vremena izvođenja metoda `GetService` i `GetFunctionFormatting` o parametrima priručne memorije. Rezultati mjerenja prikazani su tablicama 7-1 i 7-2. Grafički prikaz dijela rezultata mjerenja prikazan je slikama 7-1 i 7-2. Prosječno vrijeme izvođenja metoda dobiveno je mjerenjem vremena izvođenja 50.000 slučajno odabranih zahtjeva.

Tablica 7-1 Ovisnost vremena izvođenja metode (ms) `GetService` o parametrima priručne memorije

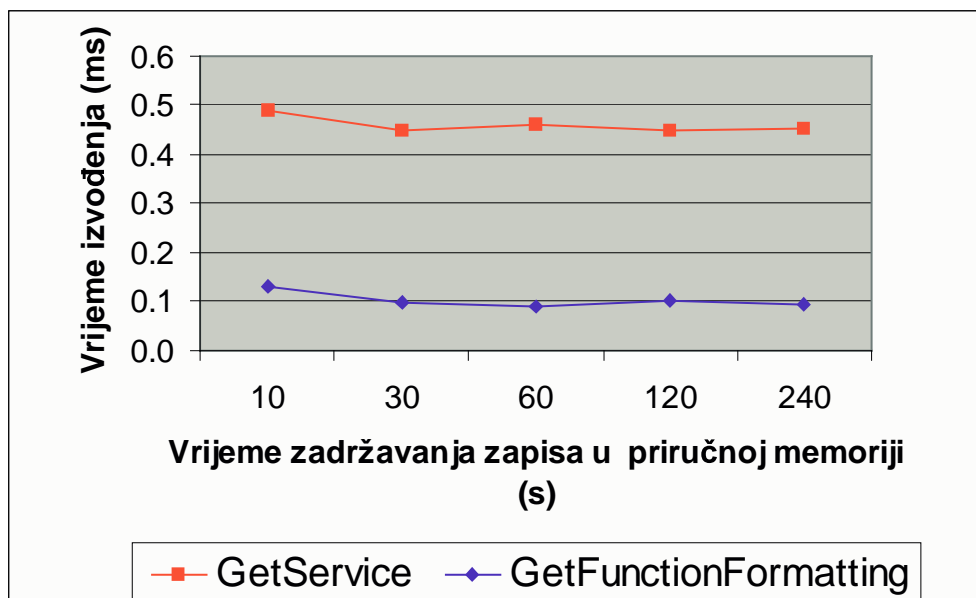
Vrijeme zadržavanja (s) \ Veličina pr. memorije (broj zapisa)	10	30	60	120	240
10	22.48	23.73	19.37	22.56	24.89
25	16.21	21.71	22.52	16.47	20.13
50	9.22	9.75	10.58	9.04	8.99
75	4.11	3.77	3.72	3.74	3.85
100	0.49	0.45	0.46	0.45	0.45
150	0.49	0.46	0.45	0.45	0.46
200	0.49	0.46	0.45	0.46	0.45

Tablica 7-2 Ovisnost vremena izvođenja metode (ms) `GetFunctionFormatting` o parametrima priručne memorije

Vrijeme zadržavanja (s) \ Veličina pr. memorije (broj zapisa)	10	30	60	120	240
10	24.52	18.38	18.42	18.37	18.46
25	13.89	13.56	16.28	12.74	13.03
50	7.32	6.82	7.60	6.80	6.72
75	4.05	3.23	3.24	3.15	3.13
100	0.13	0.10	0.09	0.10	0.09
150	0.13	0.09	0.09	0.10	0.09
200	0.11	0.09	0.10	0.09	0.09



Slika 7-1 Ovisnost vremena izvođenja metode `GetFunctionFormatting` o veličini priručne memorije, uz vrijeme zadržavanja zapisa u priručnoj memoriji 120 s



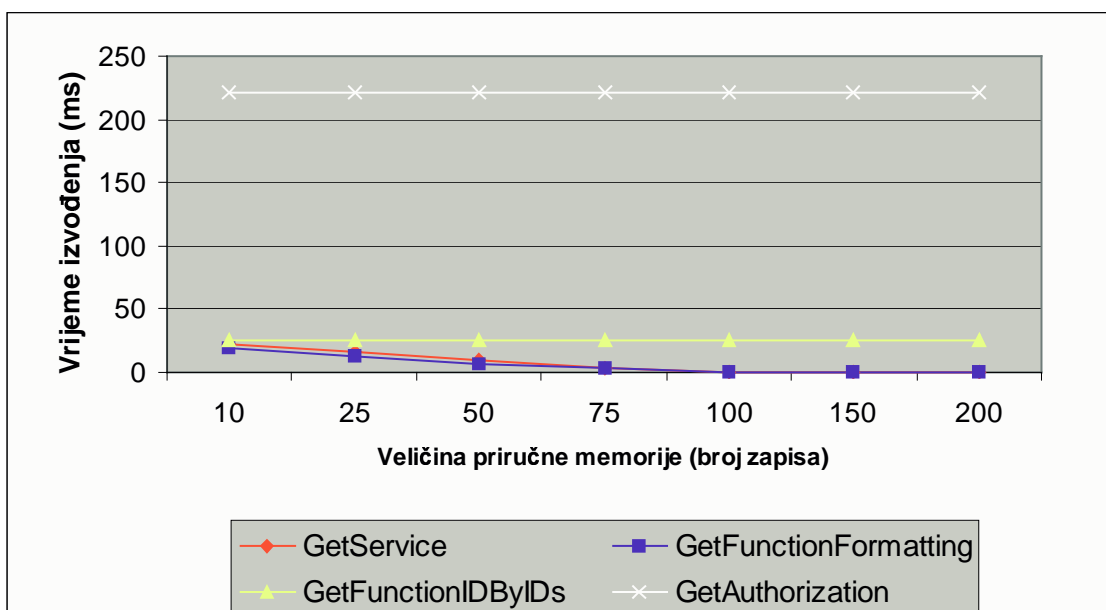
Slika 7-2 Ovisnost vremena izvođenja metode `GetFunctionFormatting` o vremenu zadržavanja zapisa u priručnoj memoriji, uz veličinu priručne memorije 100 zapisa

Grafički prikaz na slici 7-1 pokazuje da korištenje priručne memorije značajno poboljšava svojstva programske potpore pristupu direktoriju. Vrijeme izvođenja metode `GetService` smanjuje se više od 50 puta ako se koristi priručna memorija veličine 100 zapisa, u odnosu na slučaj kada se priručna memorija ne koristi. Vrijeme izvođenja metode `GetFunctionFormatting` se smanjuje 160 puta ako se koristi priručna memorija veličine 100 zapisa, u odnosu na slučaj kada se priručna memorija ne koristi.

Ako je veličina priručne memorije 10 zapisa ili manja, poboljšanja svojstava sustava su neznatna, jer se mnogo vremena troši na održavanje zapisa u priručnoj memoriji. Povećanjem veličine priručne memorije iznad 100 zapisa ne dobivaju se značajna poboljšanja svojstava sustava. Povećavanjem veličine priručne memorije iznad određene granice smanjuje se učinkovitost priručne memorije, jer veći broj zapisa znači dulje vrijeme pretrage priručne memorije. Granična veličina priručne memorije, nakon koje poboljšanja svojstava sustava počinju opadati, ovisna je o konkretnom sustavu I

Grafički prikaz na slici 7-2 pokazuje da vrijeme zadržavanja zapisa u priručnoj memoriji nema značajnog utjecaja na svojstva programske potpore pristupu direktoriju. Vrijeme zadržavanja zapisa u priručnoj memoriji nema utjecaja na svojstva programske potpore, jer se zapisi brišu iz priručne memorije prije isteka zadanog vremena, a zbog upisa novih zapisa. Vrijeme zadržavanja zapisa u priručnoj memoriji se zbog toga određuje prema učestalosti kojom se želi dohvaćati nove vrijednosti zapisa iz direktorija. Vrijeme zadržavanja zapisa u priručnoj memoriji ne smije biti manje od 10 sekundi, jer bi se smanjila učinkovitost priručne memorije zbog učestalosti održavanja zapisa u priručnoj memoriji.

Na slici 7-3 prikazana je usporedba prosječnog vremena izvođenja metoda `GetService`, `GetFunctionIDByIDs`, `GetFunctionFormatting` i `GetAuthorization`.



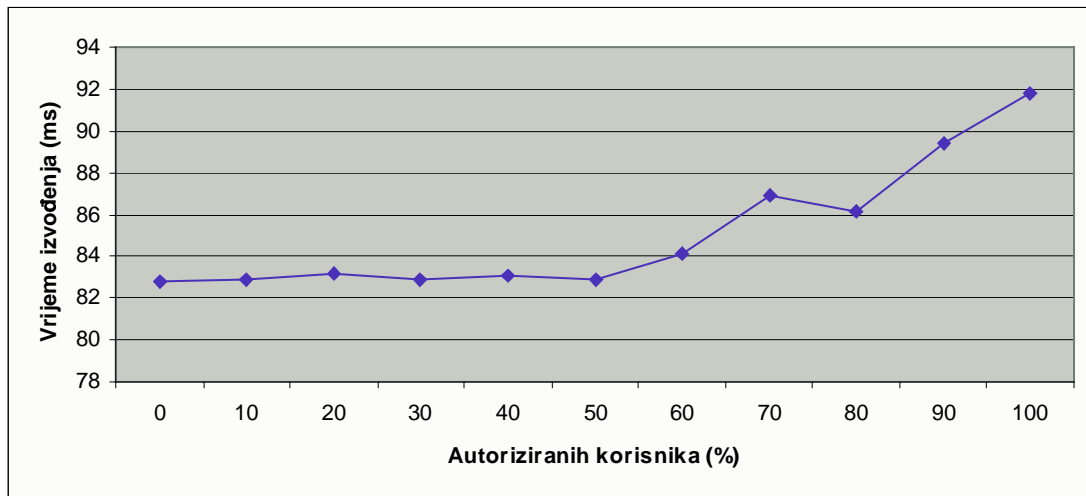
Slika 7-3 Usporedba vremena izvođenja metoda `GetService`, `GetFunctionIDByIDs`, `GetFunctionFormatting` i `GetAuthorization`.

Grafički prikaz na slici 7-3 pokazuje da je vrijeme izvođenja metode `GetAuthorization` značajno veće od vremena izvođenja preostale tri često korištene metode. Metoda `GetAuthorization` zbog toga najviše utječe na svojstva programske potpore pristupu direktoriju. S obzirom na veliki utjecaj metode `GetAuthorization` na svojstva programske potpore pristupu direktoriju, potrebno je ispitati njeno ponašanje.

Drugi skup mjerenja obuhvaća mjerenje ovisnosti prosječnog vremena izvođenja metode `GetAuthorization` o broju autoriziranih korisnika. Rezultati mjerenja prikazani su tablicom 7-3 i slikom 7-4. Prosječno vrijeme izvođenja metode dobiveno je mjerenjem vremena izvođenja 5.000 slučajno odabranih zahtjeva.

Tablica 7-3 Ovisnost vremena izvođenja metode `GetAuthorization` o broju autoriziranih korisnika

Autoriziranih korisnika (%)	Vrijeme izvođenja (ms)
0	82.784
10	82.894
20	83.213
30	82.847
40	83.031
50	82.856
60	84.156
70	86.944
80	86.178
90	89.406
100	91.831



Slika 7-4 Ovisnost vremena izvođenja metode GetAuthorization o broju autoriziranih korisnika

Prilikom mjerenja, čiji su rezultati prikazani tablicom 7-3 i slikom 7-4, korisnici su autorizirani posredno, kroz pripadnost autoriziranim grupama. Korisnici su jednoliko raspoređeni u deset grupa, pri čemu je svaki korisnik član jedne grupe. U stvarnim sustavima čest je slučaj da je većina korisnika član jedne, ili manjeg broja grupa. Samo manji broj korisnika pripada velikom broju grupa.

Mjerenja ovisnosti vremena izvođenja metode GetAuthorization o broju autoriziranih korisnika pokazuju da je vrijeme izvođenja kraće ako je broj autoriziranih korisnika manji. Ovisnost vremena izvođenja o broju autoriziranih korisnika je posljedica duljine trajanja pretrage za zapisom u direktoriju. Ako u direktoriju postoji više zapisa o autorizacijama, raste vrijeme potrebno za pronalazak zapisa. Kako bi se poboljšala svojstva programske potpore pristupu direktoriju, preporuča se grupiranje korisnika i dodjela autorizacijskih prava grupama, umjesto svakom korisniku zasebno.

Metoda GetAuthorization značajno utječe na svojstva sustava zajedničkih funkcija. Za poboljšanje svojstava sustava potrebno je proučiti rad metode GetAuthorization, i pokušati ostvariti poboljšanje svojstava metode.

8 Zaključak

Razvojem mrežnih tehnologija i porastom broja usluga raspoloživih Internetom pojavljuje se problem izgradnje usluga koje podržavaju međusobnu suradnju. Ostvarivanje suradnje usluga značajno povećava vrijeme i troškove izrade usluga. Rješenje problema je sustav zajedničkih funkcija.

Sustav zajedničkih funkcija ostvaruje funkcionalnosti često korištene u Internet aplikacijama. To su funkcionalnosti registracije i autentikacije korisnika, kontrole prava pristupa, praćenja korištenja usluga, naplate korištenja i druge. Sustav zajedničkih funkcija u svome radu koristi i sprema velike količine podataka. Spremište podataka sustava zajedničkih funkcija mora biti robusno, skalabilno i zadovoljavajućeg vremena odziva. Od postojećih modela baza podataka, navedena svojstva najbolje zadovoljava direktorij. Direktorij je skup otvorenih sustava koji pruža uslugu spremanja podataka u hijerarhijski organiziranu bazu podataka optimiranu za čitanje.

Za spremište podataka sustava zajedničkih funkcija odabran je Active Directory. Active Directory je direktorij ugrađen u operacijski sustav Windows 2000 Server. Active Directory pruža niz pogodnosti kao što su: sigurnost podataka, proširivost, skalabilnost, uvišestručavanje podataka te interoperabilnost sa drugim direktorijima. Shema direktorija opisuje tipove podataka spremljene u direktorij.

Programska potpora pristupu direktoriju izgrađena je u programskom jeziku C# za .NET Framework platformu. .NET Framework platforma omogućuje izradu programa neovisnih o operacijskom sustavu i arhitekturi računala na kojem se izvode. .NET Framework knjižnica osnovnih razreda sadrži niz razreda i struktura podataka koje olakšavaju izradu programa za rad s bazama podataka, direktorijem i drugim računalnim resursima.

Shema direktorija proširena je korisnički definiranim tipovima podataka potrebnim za spremanje podataka sustava zajedničkih funkcija. Programska potpora pristupu direktoriju omogućuje korištenje spremišta podataka sustava zajedničkih funkcija bez poznavanja unutrašnje organizacije direktorija.

Programska potpora pristupu direktoriju ostvarena je izgradnjom sučelja za pristup i upravljanje podacima direktorija. Sučelje čini trinaest razreda s ostvarenim metodama za pristup podacima u direktoriju. Pri izgradnji programske potpore posebna pozornost posvećena je svojstvima sustava.

Programska potpora pristupu direktoriju koristi priručnu memoriju za čitanje podataka iz direktorija. Priručna memorija koristi se samo prilikom čitanja. Priručna memorija za pisanje nije ostvarena, jer su podaci koji se spremaju u direktorij predviđeni uglavnom za čitanje. Utjecaj korištenja priručne memorije na svojstva programske potpore pristupu direktoriju ispitan je mjerenjem. Korištenjem priručne memorije postižu se značajna poboljšanja svojstava programske potpore pristupu direktoriju.

Način ostvarenja priručne memorije ne omogućuje njeno korištenje kod poziva svih metoda. Potrebno je proučiti mogućnosti proširenja arhitekture priručne memorije. Proširenjem arhitekture priručne memorije omogućilo bi se njeno korištenje svim metodama, čime bi se dodatno poboljšala svojstva programske potpore pristupu direktoriju.

Utjecaj broja zapisa na svojstva sustava je neznatan. Stoga je ostvareni sustav spremanja podataka u direktorij posebno je pogodan za spremanje velikih količina podataka. S obzirom da sustav zajedničkih funkcija ima za svrhu podržati veliki broj korisnika i usluga, ostvareni sustav predstavlja dobro rješenje.

Literatura

- [1] S. Tanenbaum: Computer Networks, Third Edition, Prentice-Hall International, Inc., 1996.
- [2] S. Karandikar: Database Systems,
http://www.cs.colorado.edu/~getrich/Classes/csci5817/Term_Papers/karandik/db.pdf,
travanj 2002.
- [3] S. Sol: Introduction to Databases for the Web, <http://selena.com.net/tutorials/sql/index.html>, rujan 2001.
- [4] B. Singh: An Introduction to Databases,
http://www.i5ive.com/article.cfm/office_2000/54141, rujan 2001.
- [5] J. Morgan: Alternative Database Types, Northern Arizona University College of Business Administration, http://www.cba.nau.edu/morgan-j/class/subtop1_2/sld001.htm, rujan 2001.
- [6] L. Miller-Finn: Enterprise Directory Services Proposal, Johns Hopkins Institutions, Hopkins Information Technology Services,
<http://middleware.internet2.edu/earlyadopters/scope-docs/johns-hopkins-DS-proposal.doc>, ožujak 2002.
- [7] C. Weider, J. Reynolds, S. Heker: Technical Overview of Directory Services Using the X.500 Protocol, RFC 1309, 1992., <http://search.ietf.org/rfc/rfc1309.txt>, rujan 2001.
- [8] ITU-T Recommendation X.500 Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services, 2001
- [9] D. Thompson: Understanding LDAP, White paper, Windows 2000 Server Documentation, 2000.
- [10] T. Howes: LDAP: Use as Directed, Netscape, 1999.,
<http://www.networkmagazine.com/article/DCM20000502S0039/1>, rujan 2001.
- [11] J. Myers, Network Working Group: Simple Authentication and Security Layer (SASL), Netscape Communications RFC 2222, 1997.,
<http://search.ietf.org/rfc/rfc2222.txt>, ožujak 2002.
- [12] G. Shipley, K. Novak: The Cross-Platform Challenge, Network Computing,
<http://www.networkcomputing.com/1116/1116f2.html>, ožujak 2002.
- [13] IPlanet: Netscape Directory Server 4.0 Deployment Guide,
<http://docs.iplanet.com/docs/manuals/directory/dir40/de/deploy.pdf>, travanj 2002.
- [14] Netscape: Directory Server Frequently Asked Questions,
<http://wp.netscape.com/directory/v4.0/faq.html>, ožujak 2002.
- [15] Caldera: Understanding Novell Directory Services,
<http://www.caldera.com/support/docs/openlinux/netware/nds/03.html>, ožujak 2002.

-
- [16] Eonline Training Inc.: Network Operation Systems: Novell Netware, <http://www.eonlinetraining.com/NetPlus/Section02/CHAPTER02/dds3.htm>, ožujak 2002.
- [17] Novell: Novell Directory Services – White Paper, <http://www.cts-net.com/NOV-NDS%20White%20Paper.html>, ožujak 2002.
- [18] D. Hamlett: Differences: Windows Active Directory and Novell Directory Services, <http://www.comets.com/w2kug%20ms%20files/ADNDSdiff.ppt>, ožujak 2002.
- [19] KeyLabs Inc.: Comparison Report: Microsoft Active Directory vs. Netware NDS, <http://www.keylabs.com/results/microsoft/ad/comparisonreport.pdf>, ožujak 2002.
- [20] J. Savill: DNS and Active Directory, 2000., <http://www.networkmagazine.com/article/NMG20000512S0031>, rujan 2001.
- [21] J. Perlow: Five ways Microsoft went wrong with Active Directory, <http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2715213,00.html>, rujan 2001.
- [22] Microsoft Corporation: Windows 2000 Advanced Server Documentation, <http://www.microsoft.com/windows2000/en/advanced/help/>, travanj 2002.
- [23] Microsoft Corporation: Active Directory Programmer's Guide, MSDN Library, 2001., http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netdir/ads/active_directory_start_page.asp, travanj 2002.
- [24] Microsoft Corporation: .NET Framework SDK, MSDN Library, <http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000451>, travanj 2002.
- [25] R. Housley, W. Ford, W. Polk, D. Solo, Network Working Group: Internet X.509 Public Key Infrastructure, Certificate and CRL Profile, 1999., <http://search.ietf.org/rfc/rfc2459.txt?number=2459>, ožujak 2002.
- [26] ITU-T Recommendation X.509: Information Technology – Open Systems Interconnection – The Directory: Authentication Framework, 1997.
- [27] ITU-T Recommendation X.519: Information technology - Open Systems Interconnection - The Directory: Protocol specifications, 1997.