UDC: 81:004.7 Info M: str. 39-45

PREGLED ELEMENATA JEDNOG MULTIMEDIJALNOG SOFTVERSKOG PAKETA SA OSVRTOM NA TEHNOLOGIJE IMPLEMENTACIJE
ELEMENTS OVERVIEW OF A MULTIMEDIA SUITE OF APLICATIONS WITH A REVIEW OF IMPLEMENTATION TECHNOLOGIES

Nebojša Grujić, Miroslav Marković,
Visoka tehnološka škola u Aranđelovcu

REZIME: Enormna količina multimedijalnog sadržaja koja se u novije vreme generiše, predstavlja organizacioni izazov. Multimedijalni internet servisi i društvene mreže nude određeni nivo klasifikacije, editovanja i razmene medija, dok se u off-line aranžmanu korisnici uglavnom oslanjaju na hijerarhijsku strukturu sistema fajlova sa svojim ograničenim setom atributa. Ovaj rad se fokusira na jedno drugačije rešenje organizacije medija usmereno ka desktop računaru, uz upotrebu SUBP sistema za klasifikaciju, obogaćivanje opisa sadržaja i metoda pretrage. Ovo rešenje omogućuje bogatiji set i kreaciju korisničkih atributa, mehanizam dodele ključnih reči i raznovrstan sistem pretrage. Omogućen je rad sa eksternim uređajima na način da mediji ne moraju da budu fizički prisutni u računaru tokom pretrage i pridruživanja projektima. Model povezanosti aplikacija multimedijalnog sistema omogućuje integrisani pristup podacima i jednostavnu razmenu sadržaja. Ulazi se u implementacione detalje modula sistema, njihovu koordinaciju i komunikaciju. Razmatra se rešenje orijentisano na Microsoft Windows operativni sistem, dok je za samu realizaciju zamišljeno C++, COM, ATL okruženje.

KLJUČNE REČI: Multimedija, C++, Windows, COM, ATL, MS Access, Database, SQL

ABSTRACT: Enormous amount of multimedia content generated in recent time, presents an organizational challenge. Multimedia Internet services and social networks offer a level of organization, editing and sharing of content, while the users rely on hierarchy of a file system with its limited attribute set in the area of off-line use. This article focuses on a different solution of media organization targeted for a desktop platform, using a DBMS system for classification, improved attribute set and search mechanisms. This solution allows for richer set of attributes and creation of custom ones, keywords mechanism and versatile search system. It enables usage of external storage devices in a way that media do not need to be physically present in the system to be searched for, or referenced in projects. Multimedia application connectivity model provides an integrated data access and simplified content sharing. There is a discussion of implementation details of system modules, their coordination and communication. The solution is focused on Microsoft Windows operating system with C++, COM, and ATL as development technologies of choice.

KEY WORDS: Multimedia, C++, Windows, COM, ATL, MS Access, Database, SQL

1. UVOD

Devedesetih godina prošlog veka došlo je do ekspanzije razvoja aplikacija za baratanje multimedijalnim sadržajima. Prva rešenja u domenu baratanja slikom su se pojavila na tržištu već početkom ovog perioda. Adobe [1] i Corel [2] su najzvučnija imena iz tog vremena sa rešenjima na polju raster i vektorske grafike. Prve aplikacije su bile prilično skupe i nedostupne širem broju korisnika, a što je još važnije, bile su usmerene više ka profesionalnim korisnicima.

Slična situacija je postojala u ostalim multimedijalnim granama. Neke od tehnologija, kao na primer editovanje videa, bile su tek u povoju. Kvalitetni algoritmi kompresije su tek krenuli da se razvijaju.

Ulaskom u novi vek došlo je do naglog razvoja tehnologije. Prve komercijalne digitalne kamere pojavile su se već početkom prve decenije XXI veka. Iako skupe i niskog kvaliteta, ove inovacije su pokazale put kojim će se dalje ići. Došlo je i do razvoja kvalitenih algoritama kompresije videa i digitalizacije kako postojećih fotografija, gramofonskih ploča i kaseta, tako i video traka.

Masovni prodor interneta u domaćinstva širom sveta i pojava protokola za razmenu fajlova potpuno su promenili životni stil većine. MPEG [3] kompresija je dovela do DVD [4] formata i ono što se nekada smatralo visokom tehnologijom ubrzo je postalo pristupačno prosečnim potrošačima. CD i malo kasnije DVD rezači su postali deo prosečnih konfiguracija kućnih računara.

U paraleli sa navedenim pomacima u razvoju hardvera, razvoj softvera je takođe bio brz. Konkurencija je dovela do toga da grafički i video paketi postanu toliko jeftini da je svako domaćinstvo moglo da ih priušti. Izazov je bio kako napraviti aplikacije dovoljno jake da se približe profesionalnom kvalitetu, a opet jednostavne za korišćenje da bi svako hteo da ih kupi. Pojavili su se i novi koncepti kao što je softver za kreiranje CD i DVD diskova.

U ovakvom okruženju tehnologija i alata, pojavile su se ideje o objedinjavanju ponude multimedijalnih aplikacija. Umesto da se specijalizuju za određenu oblast, firme su krenule sa proširenjem ponude i u asortiman uvrstile tehnologije koje im do tada nisu bile na repertoaru. Sa proširenjem ponude i objedinjavanjem multimedijalnog spektra, prirodno se pojavila i ideja o integraciji aplikacija.

2. KONTEKST

Baratanje medijima uključuje kreiranje, manipulaciju (editovanje), čuvanje, organizaciju, razmenu i drugo [5]. Multime-

dijalni paket tipično sadrži aplikacije koje omogućuju takve operacije. Jedna varijanta skupa ovakvih aplikacija, koja bi po potrebi mogla da se proširuje, data je u sledećoj listi:

– Foto editor
– Video editor
– DVD editor
– Audio editor
– Medija pretraživač
– CD/DVD rezač

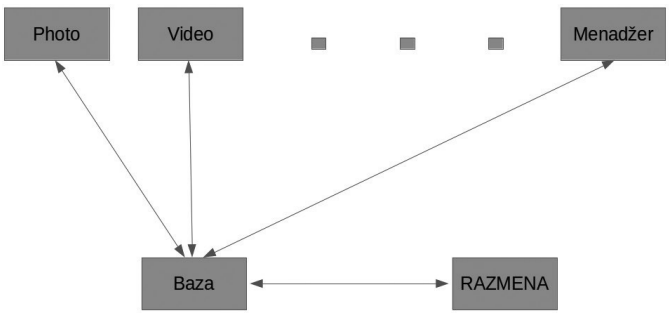
Medija pretraživač je grafički alat ekvivalentan Windows Explorer ili u ranijim verzijama Windows sistema, File Manager aplikaciji. Služi za navigaciju hijerarhije sistema, pretragu, a i dodatne operacije, te kako ima širu upotrebu od same pretrage, u nastavku se pominje samo kao Menadžer.

U dodatku ove liste, a što će biti jasnije kada budu razmatrani detalji implementacije, nalaze se i:

– Servis baze podataka
– Servis internet razmene

Ovaj članak će zadržati akcenat na Menadžeru i servisima baze i interneta. Ostale aplikacije su interesantne u pogledu interakcije sa navedenim, i biće pomenute isključivo u tom kontekstu bez dubljeg razmatranja.

Interakcija između komponenti je koncentrisana oko servisa baze podataka kao centra čitavog sistema (Slika 1). Servis baze, koji se oslanja na standardni DBMS sistem u vidu MS Access baze podataka, čini fizičku lokaciju fajlova potpuno transparentnom. Samim tim, baza podataka postaje i jedini medijum za skladištenje i potraživanje materijala. Iz ovog razloga, svaka od komponenti mora da ostvari direktnu komunikaciju sa ovim servisom.



Slika 1. *Blok šema interakcije komponenti sistema*

Servis baze implementira dvosmernu komunikaciju sa komponentama sistema. Do njega dolaze zahtevi za pristup i organizaciju fajlova, pretraživanje, baratanje atributima, a kao odgovor na promene u bazi podataka, šalje notifikacije klijentima.

Primera radi, korisnik može da edituje fotografiju, a u isto vreme ima otvoren interfejs Menadžera i DVD kreatora gde je ta slika upotrebljena kao naslovna strana neke video sekvence. Onog momenta kada korisnik snimi svoje promene na disk, notifkacija se šalje Menadžeru. Menadžer aktivira servis za formiranje ikonica sistema, koji ažurira ikonicu te fotografije unetim promenama korisnika. Ta promena se reflektuje na interfejsu Menadžera gde nova ikonica zamenjuje staru. Notifikacija se takođe propagira DVD softveru, koji dinamički ažurira svoj sadržaj novim promenama. Na sličan način funkcionišu i ostali delovi sisitema.

3. ORGANIZACIJA MEDIJA

Prilikom razmatranja organizacije medija, osnovno pitanje je način na koji se mediji čuvaju. Na prvom mestu čuvaće se na hard disku računara na kome će paket funkcionisati, a zatim i na spoljašnjim medijumima kao što su CD/DVD diskovi, razne vrste eksternih memorija, eksterni hard diskovi, razni internet servisi i drugo.

Iz ovoga se vidi da će podaci teorijski biti rasuti na mnogo strana i odmah se nameće pitanje njihove klasifikacije. Drugim rečima, neophodno je osmisliti nekakav sistem po kome bi se mediji grupisali, klasifikovali i pretraživali, bez obzira na njihovu fizičku lokaciju.

Uz sve navedeno mediji uz sebe obično imaju i veliki broj pridruženih podataka (Meta Data) koji ih dodatno obogaćuju. Uz standardne podatke kao što su vreme i datum nastanka, veličina fajla, naziv, postoje i oni kao što su lokacija, imena učesnika, GPS koordinate i sl. Neki od ovih podataka su sačuvani u vidu atributa fajlova, a neki su smešteni unutar samih medijskih fajlova u vidu struktura kao na primer EXIF [6]. Poželjno je takođe da po potrebi mogu da se kreiraju i novi atributi koji nisu predviđeni nijednim postojećim sistemom.

Iz navedenog je očigledno da nije pogodno osloniti se isključivo na fajl sistem jer ne zadovoljava navedene potrebe, pa se zbog toga rešenje traži uz korišćenje baze podataka. Kako se radi o desktop sistemu, a i predviđene performanse i mogućnosti nisu veoma zahtevne, neki sistem baze podataka na osnovu fajlova u potpunosti zadovoljava potrebe [7]. Primeri za ovo su MS Access [8] i SQLite.

Upotreba baze podataka automatski ostavlja mogućnost da se u nju skladišti mnogo više od atributa. U obzir dolazi široki dijapazon upotreba. Individualne aplikacije mogu je koristiti za smeštanje svojih projektnih fajlova. Moguća je upotreba za servise tipa clipboard ili za listu skorije korišćenih fajlova. Tu su i stringovi pretraživanja, ključne reči, reference prema uređajima (za eksterne diskove i memoriju), rezultati pretrage, korisnički atributi, efekti i slično [9]. Izbor je praktično neograničen.

4. RAZMENA SADRŽAJA

Ideja razmene sadržaja predstavlja nadgradnju u odnosu na već postojeće modele putem društvenih mreža i specijalizovanih multimedijalnih web sajtova. Mnogi od ovih intenet servisa već nude programerska okruženja za direktan aplikativni pristup. Sadržaj se može podići na nalog korišćenjem privatnog korisničkog interfejsa bez izlaska na web stranicu provajdera [10][11][12].

Lokalna kućna mreža takođe nudi mogućnost striminga sadržaja na lokalne uređaje [13]. U igri su razni modeli implementacije, uključujući i korišćenje Windows Home Media Server tehnologije [14].

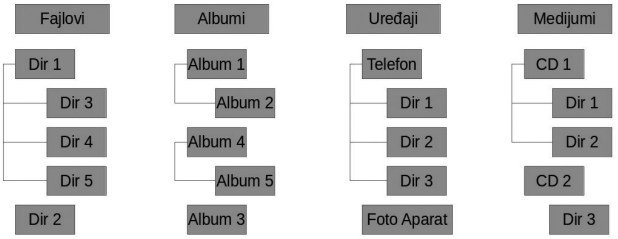
Bez ulaženja u detalje, ove ideje su konceptualno interesantne radi dizajniranja organizacionog servisa. Ono što je važno primetiti je da mediji na računaru generalno pripadaju različitim korisnicima. Na primer, jedan računar potencijalno podržava naloge više ukućana. Iz ovoga se zaključuje da organizacioni servis mora da bude u stanju da istovremeno pristupi svim tim podacima, bez obzira koji je korisnik trenutno aktivan. Ovo je moguće samo ako je taj servis implementiran na principu sistemskog servisa.

Sistemski servis funkcioniše u domenu sistemskog naloga. Samim tim je u mogućnosti da pristupa svim podacima na računaru, i servisira zahteve svih korisnika, bili oni ulogovani ili ne. Ovakavim modelom se postiže jedinstven pristup za sve korisnike.

5. MEDIJA PRETRAŽIVAČ

Pretraživač je aplikacija slična bio kom pretraživaču fajlova koje ima svaki operativni sistem. Razlika je što mu je primarna upotreba vezana za pristup već opisanoj multimedijalnoj bazi podataka. U isto vreme on implementira i uobičajeni pristup sistemu fajlova sa ciljem bolje integracije. Dodatne klasifikacije uključuju mapiranje spoljašnjih medija i mobilnih uređaja kao što su telefoni i foto aparati, zatim specijalni entiteti kao na primer rezultati pretrage i sl [5].

Slika 2 daje logički pregled navedenih klasifikacija. Grafički interfejs se implementira tako da svaka klasifikacija ima zaseban prostor na ekranu da bi se izbegla konfuzija, ali je u isto vreme omogućena interakcija u vidu drag-and-drop i metoda baziranih na principu menija.



Slika 2. *Logičke klasifikacije medija*

Potreba da se uređaji i medijumi klasifikuju kao posebni entiteti, iako su podržani od strane fajl sistema, dolazi iz činjenice da se njihovi sadržaji čuvaju u bazi podataka. Na taj način postoji mogućnost da se njihov sadržaj brauzuje bez potrebe da su fizički prisutni. Naravno, u slučaju potrebe da se pristupi samim fajovima, grafički interfejs bi prikazao poruku da se određeni medijum priključi na kompjuter.

6. IMPLEMENTACIONI DETALJI

Opisani sistem aplikacija nosi sa sobom veliki stepen kompleksnosti. Verovatno najveći razvojni trud je potreban kod implementacije korisničkog interfejsa. Na ovo se nadovezuju tehnologije vezane za obradu sadržaja kao što su procesiranje slike i videa, manipulacija DVD formata i sl. Kako se članak bavi limitiranim opsegom tehnologija, biće obrađeni samo osnovni aspekti navedenih servisa.

U suštini sistema se nalaze Microsoftove tehnologije komponentnog objektnog modela (COM) [15][16], upakovane kombinacijom C++ [17] programskog jezika uz upotrebu ATL biblioteke. COM je distribuirani model komponenti koje se uz minimalni programerski trud mogu koristiti u korisničkim aplikacijama. Ovo je u većini slučajeva tehnologija distribucije komponenti na Microsoft Windows operativnom sistemu [18].

COM je baziran na principu modula registrovanih u sistemu, tako da se svaki modul može jednostavno adresirati korišćenjem UUID identifikatora. Ovo je najjednostavnije objasniti na primeru.

U slučaju da treba kreirati neki objekat za širu upotrebu u COM okruženju, neophodno je za njega generisati IDL fajl (Listing 1).

```
[
    object,
    uuid(4A3E5C08-FA4F-4349-910E-8B5FEF61A38B),
    dual,
    nonextensible,
    helpstring("IPrimerInterfejsa Interface"),
    pointer_default(unique)
]
interface IPrimerInterfejsa : IDispatch{
    [id(1), helpstring("method Otvori")] HRESULT
Otvori([in] long ulaz);
    [id(2), helpstring("method Zatvori")] HRESULT
Zatvori([out, retval] long *izlaz)
};
[
    uuid(3D0F2F2B-686C-4ECB-B779-A690AAEB2EB3),
    version(1.0),
    helpstring("PrimerBiblioteke 1.0 Type Library")
]
library  PrimerBiblioteke
{
    [
        uuid(2CF6A30B-4F12-4541-A7F7-6E4D150C762E),
        helpstring("Primer Klase")
    ]
    coclass  PrimerKlase
    {
        [default] interface IPrimerInterfejsa;
    };
};
```

Listing 1. *IDL Definicija objekta*

Iz sekvence IDL koda izdvajaju se tri entiteta: IPrimerInterfejsa, PrimerBiblioteke i PrimerKlase. Svi ovi entiteti će po završetku kompajliranja i registracije modula biti upisani u Windows registru kome se može pristupiti korišćenjem regedit.exe aplikacije.

Svaki entitet će biti registrovan na svojoj ogovarajućoj lokaciji u Interface, TypeLib i CLSID sekcijama pod HKEY_

CLASSES_ROOT ključem. Ovaj korak je neophodan da bi sistem mogao da pronađe modul prilikom upotrebe.

Sama registracija je izvedena uz pomoć regsvr32.exe alatke koja se poziva tokom nekog post bild koraka. U ovu svrhu se mora pripremiti i registracioni fajl (Listing 2) u resursima projekta.

```
HKCR
{
   PrimerBiblioteke.PrimerKlase.1 = s 'PrimerKlase
Class'
   {
      CLSID = s '{2CF6A30B-4F12-4541-A7F7-6E4D-
150C762E}'
   }
   PrimerBiblioteke.PrimerKlase = s 'PrimerKlase
Class'
   {
      CLSID = s '{2CF6A30B-4F12-4541-A7F7-6E4D-
150C762E}'
      CurVer = s 'PrimerBiblioteke.PrimerKlase.1'
   }
   NoRemove CLSID
   {
      ForceRemove {2CF6A30B-4F12-4541-A7F7-6E4D-
150C762E} = s ' PrimerKlase Class'
      {
         ProgID = s 'PrimerBiblioteke.PrimerKla-
se.1'
         VersionIndependentProgID = s 'PrimerBi-
blioteke.PrimerKlase'
         ForceRemove 'Programmable'
         InprocServer32 = s '%MODULE%'
         {
               val ThreadingModel = s 'Apartment'
         }
         val AppID = s '%APPID%'
         'TypeLib' = s '{3D0F2F2B-686C-4ECB-B779-
A690AAEB2EB3}'
      }
   }
}
```

Listing 2. *Fragment registracionog fajla*

Kada se koristi odgovarajući objekat, potrebno ga je importovati u kodu. Konkretno iz navedenog primera, mora se importovati PrimerBiblioteke (Listing 3) u heder fajlu projekta korisnika na sledeći način:

```
#import "PrimerBiblioteke.tlb" no_namespace
named_guids
```

Listing 3. *Primer importovanja objekta u kodu*

Interesantno je primetiti da korišćenjem ovakvog pristupa nije potrebno voditi računa o tome gde se biblioteka nalazi jer o tome brine Windows SCM menadžer upotrebom Windows Registry sistema. U primeru je direktno korišćena TLB biblioteka. U nekim slučajevima, kao što će biti pokazano u narednom tekstu, ova biblioteka je sadržana u sklopu binarnih fajlova, na primer DLL.

Kada se importuje TLB biblioteka, kompajler kreira primarni TLH i sekundarni TLI heder fajlove koji sadrže sve što je potrebno da bi se koristili objekti bibioteke u C++ kodu. Navedeni proces je potpuno automatizovan i nije potrebno direktno baratanje ovim fajovima.

6.1 *Pristup bazi podataka*

Dva najpopularnija okruženja koja omogućavaju pristup bazama podataka na Windows sistemu su DAO i ODBC. ODBC je standard dizajniran za univerzalan pristup različitim database sistemima i radi uz pomoć drajvera koji taj sistem obezbeđuje. DAO je okruženje dizajnirano oko Microsoft JET-a i optimizovano je za pristup MS Access bazama podataka. Ovo je odličan izbor u slučaju aplikacije koja se u ovom radu prikazuje.

DAO objekti se mogu koristiti na već opisani način putem COM pristupa. Proces započinje importovanjem DAO biblioteke:

```
#import "dao360.dll" rename("EOF", "daoEOF") no_
namespace named_guids
```

Listing 4. *Importovanje DAO biblioteke*

U kodu se dobijaju generisani _com_ptr_t smart pointeri koji se koriste direktno:

```
IClassFactory2Ptr pFactory;
HRESULT hr = ::CoGetClassObject(CLSID_DBEngine,
CLSCTX_INPROC_SERVER, NULL, IID_IClassFactory2,
(void**)&pFactory);
if(FAILED(hr)) _com_issue_error(hr);

// DAO 3.6 runtime key
bstr_t bstrKey =
L"mbmabptebkjcdlgtjmskjwtsdhjbmkmwtrak";
_DBEnginePtr pEngine;
pFactory->CreateInstanceLic(NULL, NULL, IID__
DBEngine, bstrKey, (void**)&pEngine);
DatabasePtr pDatabase = pEngine-
>OpenDatabase(L"db.mdb");
ASSERT_RETURN((bool)pDatabase, E_FAIL);
```

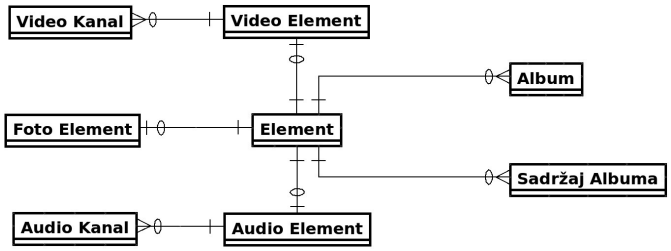
Listing 5. *Instanciranje Database objekta DAO okruženja*

Onog momenta kada je dobijen DatabasePtr (Listing 5), praktično je ostvarena veza sa bazom i mogu se upotrebljavati standardni objekti kao što su tabele, rekordi, setovi i slično.

6.2 *Struktura baze podataka*

Primarna uloga baze podataka je da obezbedi dodatne atribute pridružene fajlovima medija. U isto vreme, potrebno je omogućiti nekakav vid klasifikacije i organizacije medija. Imajući u vidu koliko postoji različitih vrsta medija, kao na primer foto, video, audio i td., mora se obezbediti mogućnost da se različitim medijima dodele različiti atributi.

Iz navedenog se određuje i osnovna struktura baze, a relacije između tabela su date na Slici 3. U ovom razmatranju neće se ulaziti u detalje kao što su opisi individualnih atributa. Primarno je interesantno strukturno rešenje.



Slika 3. *Osnovna struktura baze podataka*

Elementi albuma sadrže osnovne atribute primenljive na sve medija fajlove. Foto, audio, video i ostali mogući formati imaju svoje specifične atribute i samim tim posebne tabele za njihovo čuvanje.

Entitet albuma je u isto vreme i element, ali ga od običnog elementa razlikuje to što je zastupljen i u tabeli albuma. Tabela sadržaja albuma uspostavlja relaciju između albuma i elemenata. Album može da sadrži elemente, a takođe i druge albume.

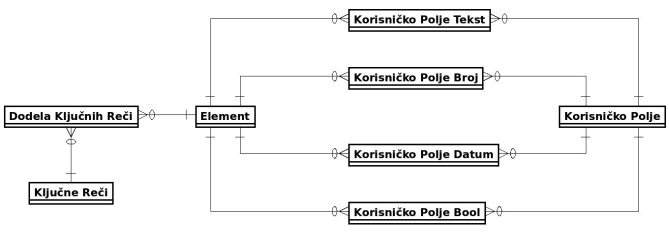
Element koji predstavlja fotografiju će biti zastupljen i u tabeli foto elemenata. Audio i video kanali nisu posebni elementi. Audio element može da sadrži više audio kanala, ali se radi o jednom istom fajlu i iz tog razloga odgovara mu samo jedan element.

Osnovna uloga elementa je da povezuje entitet baze podataka sa fizičkim fajlom na disku. Svaki element koji predstavlja fajl na disku ima skladištenu referentnu putanju uz pomoć koje se vrši direktan pristup tom fajlu.

Upotreba baze podataka na ovaj način otvara mogučnost upotrebe uprošćenih stringova u komunikaciji između aplikacija i baze podataka. Primera radi, individualni fajl se može jednostavno predstaviti jednim brojem, odnosno vrednošću polja ElementID. Kolekcija fajlova se može na sličan način predstaviti takođe jednim brojem koji je zapravo identifikator albuma. Ako se napravi album koji predstavlja rezultat pretrage, rezultat se takođe svodi na jedan broj.

Postoji realna mogućnost da se tokom projektovanja sistema može prevideti neki atribut, a takođe je moguće da sam korisnik osmisli neki svoj. Struktura je takva da se mora omogućiti nekoliko različitih tipova polja koja se mogu dodeliti elementima po želji.

Pored korisničkih atributa, potrebno je obezbediti i pridruživanje ključnih reči. Slika 4 objedinjuje ove ideje.



Slika 4. *Relacije tabela korisničkih polja*

Tabela KorisnickaPolja sadrži tip, ime i opis polja. Pridruživanje polja elementima albuma je izvedeno uvođenjem tabela KorisnickoPoljeBool, KorisnickoPoljeDatum, KorisnickoPoljeBroj i KorisnickoPoljeTekst. Ove tabele imaju i polje u koje se smešta dodeljena vrednost.

6.3 *Sistemski servisi i komunikacija među procesima*

Sistemski servis baze podataka je proces koji se izvršava pod administrativnim nalogom. Ovo je urađeno iz razloga što servis mora da funkcioniše bez obzira na to da li je neki korisnik logovan ili nije. Podešen je kao sistemski servis sa automatskim startovanjem. U isto vreme je implementiran i kao COM Out of Process Server. To znači da klijenti aplikacije mogu da instanciraju objekte koji se nalaze unutar ovog servisa i koriste ih kao da su lokalni iako u pozadini dolazi do inter-procesne komunikacije (IPC). Pozivi, odnosno funkcije, implemetirani u vanprocesnom serveru, funkcionišu na principu RPC (remote procedure call) mehanizma. Ovaj mehanizam je detaljno opisan u literaturi [14].

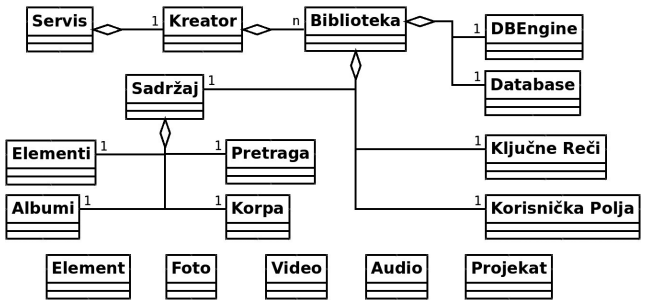
Kako na jednom računaru može postojati nekoliko korisničkih naloga, za očekivati je da svaki ima svoju bazu podataka. Te baze će biti locirane u korisničkom prostoru odgovarajućih naloga. Ovo znači da sistemski servis treba da ispoštuje privilegije i restrikcije tih naloga u komunikaciji sa bazama. U tu svrhu servis baze mora da aktivira proces impersonifikacije pozivanjem funkcije CoInitializeSecurity kao na primeru (Listing 7).

```
CoInitializeSecurity(NULL, -1, NULL, NULL, RPC_C_
AUTHN_LEVEL_NONE, RPC_C_IMP_LEVEL_IMPERSONATE,
NULL, EOAC_NONE, NULL);
```

Listing 7. *Inicijalizacija impersonifikacije COM sistema*

6.4 *Implementacija servisa baze*

Slika 5 prikazuje uprošćeni skup objekata implementiranih sa ciljem jednostavnijeg i preglednijeg pristupa tabelama i ostalim elementima baze podataka.



Slika 5. *Objekti pristupa bazi podataka*

Na nivou samog servisa postoji referenca na Kreator objekat. Implementiran je po modelu singltona i ima ulogu da prima zahteve klijenata za pristup bazi. Održava listu već otvorenih objekata Biblioteke i kreira nove po potrebi. Lista ovih objekata je implementirana po principu mape u kojoj ključ čini putanja do fajla baze podataka, dok je drugi deo para sama referenca. Ovo je moguće iz razloga što svaki korisnički nalog ima svoju zasebnu bazu koja se nalazi u zasebnom fajlu.

Objekat Biblioteka tokom sopstvenog kreiranja automatski otvara bazu i zadržava referencu na DBEngine i Database objekte koji su deo DAO okruženja. Database objekat omogućuje da se ostvari direktan pristup bazi i koristimo SQL izraze. Uz DAO objekte postoji i kreiranje objekata koji mogu da postoje nezavisno unutar aplikativnog okruženja kao što su Ključne Reči i Korisnička Polja.

Sadržaj je objekat osmišljen zarad podrške korisničkom interfejsu. Tu se čuvaju reference na objekte tipa Album, koji predstavljaju polazne tačke u panelima korisničkog interfejsa. Album pod nazivom Elementi sadrži kolekciju svih elemenata u bazi podataka. Pretraga je Album sastavljen od rezultata pretraživanja. Korpa je pomoćni album u koji idu elementi koji još uvek nisu klasifikovani na drugi način. U korenu klasifikacionog modela je kontejner pod nazivom Albumi iz koga počinje grananje čitavog sistema, ekvivalentno osnovnom direktorijumu hard diska. Ovaj Album je isključivo rezultat korisničke kreativnosti.

Postoji još i niz korisničkih objekata koji nisu u referenci sa osnovnim sistemom, ali se do njih dolazi interakcijom sa već opisanim objektima. To su uglavnom elementi koji sadrže atribute medija fajlova, kao na primer Element, Foto, Video, Audio, Projekat i drugi.

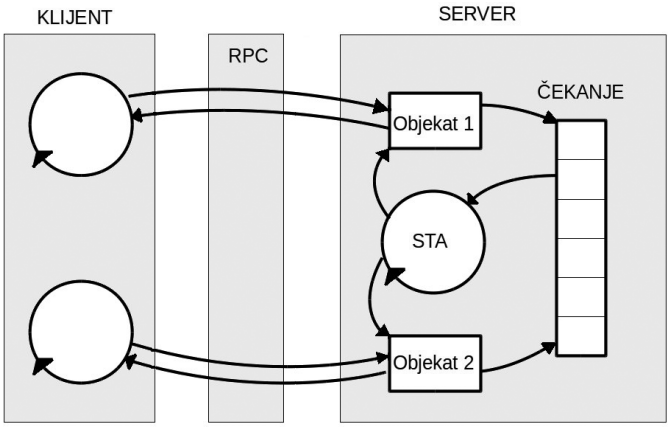
Važno je napomenuti da korisnik nije ograničen na opisane impementacije. Kako su svi navedeni objekti servisa izloženi kao javni, klijent aplikacija ih može koristiti direktno. Tako je na primer potpuno očekivano da klijent zahteva izvršenje sopstvenog SQL izraza. Na ovaj način je omogućen priličan stepen slobode koju ostale aplikacije sistema mogu da ostvare.

6.5 *Threading model*

Servis baze nema potrebu za paralelizmom. Iz tog razloga funkcioniše u STA (single threaded appartment) režimu i ima samo jednu nit (thread) koja servisira čitav proces. Ovo znači da RPC (remote procedure call) pozivi koji stižu kroz COM mehanizam završavaju u redu gde čekaju na procesiranje. Obrada se vrši uz pomoć Windows sistema poruka (messaging mechanism).

Windows sistem poruka zasniva se na konceptu beskonačne petlje koja se izvršava u glavnoj niti procesa. Ova petlja u svakom ciklusu proverava da li su stigle nove poruke i blagovremeno ih procesira. Kada vanprocesni poziv (RPC) stigne do servera, formira se zapis u redu čekanja koji sa sobom nosi sve potrebne detalje koji u potpunosti opisuju tu funkciju. Kada ovaj zapis dođe na red za procesiranje, uklanja se sa liste čekanja i nit servera posvećuje svo svoje vreme njegovoj obradi. U međuvremenu je klijent blokiran na tom istom RPC pozivu i čeka provratak iz funkcije. Po završenoj obradi, klijent dobija odgovor u vidu povratka iz RPC poziva, dok nit servera nastavlja sa radom i skida nove zahteve sa čekanja. Ovaj proces se ponavlja sve dok ne stigne zahtev-poruka za prekid rada, u kom slučaju nit izlazi iz beskonačne petlje i prestaje sa radom.

Ovaj model je prikazan na Slici 6 gde su niti prikazane u vidu usmerenih kružnica koje pokazuju beskonačnu prirodu svog izvršenja. Ilustracija se mora uzeti sa rezervom iz razloga što se apsolutno sav kod izvršava isključivo u kontekstu niti, te objekti ne postoje zasebno kao što bi sa slike moglo da se zaključi.



Slika 6. *Mehanizam procesiranja poziva*

6.6 *Sinhronizator*

Jedno od bitnijih poboljšanja koje se može implementirati u kombinaciji sa opisanim modulima je i sinhronizator.

Opisani sistem je koncentrisan oko baze podataka koja povezuje virtualnu strukturu baziranu na konceptu albuma, i fizičku strukturu sistema fajlova. Realno je očekivati da će korisnik nezavisno baratati menadžerom fajlova, koji nije integrisan sa sistemom. Ovo otvara mogućnost da se postojeći linkovi sačuvani u bazi poremete, na primer premeštanjem fajlova na drugačije pozicije. Drugi aspekat ovog problema, koji se može pogodno iskoristiti, je opcija automatskog ubacivanja fajlova u bazu podataka.

Ideja je da se osmisli mehanizam koji će biti pretplaćen na notifikacije sistema fajlova, tako da kada se na primer neki folder promeni na bilo koji način, dobija se poruka o tome i samim tim mogućnost da se odreaguje i automatski ažurira baza.

U najosnovnijim crtama, za implementaciju ove opcije kao prvo mora se odabrati direktorijum koji će se pratiti. Može se pratiti i više direktorijuma odjednom, a za to je potrebno održavati listu direktorijuma.

Sledeći korak je dobijanje reference na sistemski objekat pridružen ovom direktorijumu u vidu HANDLE varijable. Ova varijabla, ili više njih ako se prati niz direktorijuma, mora se pridružiti takozvanom Completion Port objektu. Ovaj objekat je implementiran od strane Windows kernela i služi za asinhronu komunikaciju sa I/O sistemom (Listing 8).

```
HANDLE hDir = CreateFile(direktorijum,  FILE_LIST_
DIRECTORY, FILE_SHARE_READ | FILE_SHARE_WRITE ,
NULL, OPEN_EXISTING, FILE_FLAG_BACKUP_SEMANTICS |
FILE_FLAG_OVERLAPPED, NULL);
CInformacije * pInfo = new
Cinformacije(/*informacije potrebne u odgovoru*/);
HANDLE hComplPort = CreateIoCompletionPort(hDir,
hComplPort, (DWORD)pInfo, 0);
PostQueuedCompletionStatus(hComplPort,
sizeof(pInfo), (DWORD)pInfo, 0);
```

Listing 8. Asinhrona komunikacija sa I/O sistemom

Način na koji je u primeru upotrebljen poziv funkcije CreateIoCompletionPort, gde se postojeći port dodeljuje ponovo u

vidu drugog parametra, ukazuje na to da se koristi samo jedan port objekat za praćenje svih posmatranih direktorijuma.

U nastavku se pristupa paketima generisanim od strane Completion Port objekta (Listing 9).

```
CInformacije * pInfo;
GetQueuedCompletionStatus(hComplPort, &numBytes,
(LPDWORD) &pInfo, 0, INFINITE);
```

Listing 9. *Aktivacija čekanja na I/O notifikacije*

Nekoliko stvari primećuje iz primera poziva funkcije za pristup paketima porta. Prvi parametar je isti port koji je kreiran u originalu. Treći parametar prima informacije koje su pridružene portu tokom njegove kreacije i ovom prilikom se dobijaju nazad kako bi se znalo o kom pozivu se tačno radi. Ono što je najvažnije, četvrti parametar ukazuje na blokirajuću prirodu ovog poziva. To praktično znači da će ovaj poziv da čeka sve dok port ne bude na raspolaganju. Iz ovoga se vidi da je neophodno kreirati jednu novu izvršnu nit koja će pomoći pri tom asinhronom procesiranju.

Ova pomoćna izvršna nit ujedno preuzima svo procesiranje notifikacija kao i ažuriranje baze podataka. Sinhronizator se može implementirati u sklopu servisa baze podataka. U tom slučaju bi se iskomplikovao threading model, a i nepotrebno bi se pratili direktorijumi korisnika koji nisu logovani. Bolje rešenje je da se implementira korisnički servis koji bi bio u funkciji samo tokom login seanse odgovarajućeg korisničkog naloga.

7. ZAKLJUČAK

Članak je predstavio grubu sliku jednog dela komercijalnog multimedijalnog paketa, a detaljnije ideje i rešenja su zanemareni. Bez obzira na to, ključne tehnologije su objašnjene u dovoljnoj meri da se čitalac zainteresuje i dobije podsticaj za sopstveno eksperimentisanje.

Tokom razvoja projekata, važno je obraditi detalje visokog rizika odvojeno od ostalih elemenata sistema. Čest je slučaj da se određene tehnologije ne mogu razumeti ni posle ponovljenog čitanja literature, a dešava se da propratna dokumentacija ne pominje ključne elemente. Iz ovih razloga, veoma je korisno da projekat otpočne pojednostavljenim prototipom. Jako je važno da se potvrdi funkcionalnost odabranog okruženja i razrade zamišljeni koncepti na zadovoljavajući način. Kada se ostvari solidna osnova, kreće se sa složenijom nadgradnjom.

Primera radi, autor je imao konkretan slučaj da se u poodmakloj fazi razvoja projekta otkrilo da jedan ključni metod JET okruženja jednostavno nije radio. Srećom, rešenje je pronađeno zaobilaznim putem korišćenjem više drugih metoda u istu svrhu. Ovo je doduše specifičan primer koji je teško pronaći izradom prototipa, ali ilustruje moguće komplikacije.

U današnje vreme, uporeba interneta je toliko zaživela da ideja o desktop menadžmentu deluje zastarelo. Praksa je ipak pokazala da svako rešenje ima svoju vrednost, te da nije racionalno oslanjati se isključivo na moderne trendove. U krajnjoj liniji, razvoj server tehnologija još kako barata sa sličnom tematikom koja je ovde pomenuta, naravno u širem i ozbiljnijem kontekstu.

Ideje o takozvanom „tankom“ klijentu ne odgovaraju svim korisnicima. U igri su problemi oko pristupa internetu, bezbednosti sistema, kredibilitetu hostinga i td. Iz ovih razloga, uvek će postojati niša potrošača kojima će mnogo više odgovarati da „konce“ drže u svojim rukama.

LITERATURA

[1] Adobe Photoshop Family - http://www.adobe.com/products/photoshopfamily.html
[2] CorelDRAW - http://www.coreldraw.com/rw/
[3] MPEG Video Compression Standard; John L. Mitchell; Springer; 2013
[4] Standard Handbook of Video and Television Engineering; Jerry Whitaker, Blair Benson; McGraw-Hill; 2003
[5] Multimedia Foundations: Core Concepts for Digital Design; Vic Costello; Focal Press; 2012
[6] Exchangeable image file format for digital still cameras - http://www.exif.org/Exif2-2.PDF
[7] Database Design and Implementation; Edward Sciore; Wiley; 2008
[8] Access 2013 Bible; Michael Alexander; Richard Kusleika; Wiley; 2013
[9] A spatio-temporal semantic model for multimedia database systems and multimedia information systems; Shu-Ching Chen - http://ieeexplore.ieee.org/xpl/articleDetails.jsp?&arnumber=940735
[10] The Flickr API - https://www.flickr.com/services/api/
[11] Facebook for Developers - https://developers.facebook.com/
[12] Picasa Web Albums Data API - https://developers.google.com/picasa-web/
[13] Smart authoring and sharing of multimedia content in personal area networks based on Subject of Interest; Belal Abu-Naim - http://ieeexplore.ieee.org/xpl/articleDetails.jsp?&arnumber=6890603
[14] Creating a Digital Home Entertainment System with Windows Media Center; Michael Miller; Que Publishing; 2006
[15] Inside Com (Microsoft Programming Series); Dale Rogerson; Microsoft Press; 1997
[16] Essential COM; Don Box; Addison-Wesley Professional; 1998
[17] C++: The Complete Reference; Herbert Schildt; Osborne McGraw-Hill; 1998
[18] Microsoft Developer Network - https://msdn.microsoft.com/library
[19] SQL (Database Programming); Chris Fehily; Questing Vole Press; 2014

Nebojša Grujić, programer-freelancer
Kontakt: nebojsa.grujic@gmail.com
Oblasti interesovanja: Linux, Real-Time procesiranje, Embedded sistemi

Dr Miroslav Marković, profesor na Visokoj Tehnološkoj školi u Aranđelovcu
Kontakt: miroslav.markovic@vtsar.edu.rs
Oblasti interesovanja: Algoritmi, Embedded sistemi, Internet of things