UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Peter Djukić

PRENOVA APLIKACIJE ZA VODENJE RADIJSKIH IGER

DIPLOMSKO DELO VISOKOŠOLSKEGA STROKOVNEGA ŠTUDIJA

Mentor: viš. pred. dr. Igor Rožanc

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00450/2009
Datum: 05.04.2009

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

Kandidat: **PETER DJUKIĆ**

Naslov: **PRENOVA APLIKACIJE ZA VODENJE RADIJSKIH IGER**

THE REENGINEERING OF RADIO DRAMA MANAGEMENT APPLICATION

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Na Radiu Slovenija obstaja obsežen arhiv radijskih iger. V preteklosti je bila razvita aplikacija za upravljanje s podatki arhiva, ki sedaj ne izpolnjuje več varnostnih zahtev. V diplomski nalogi predstavite prenovo te aplikacije in njeno nadgradnjo.

Na začetku na kratko predstavite uporabljene tehnologije in principe prenove programske opreme. Izhajajoč iz značilnosti obstoječe aplikacije in zahtev uporabnikov nato ustrezno zasnujte in izvedite prenovo. Poseben poudarek posvetite varnosti podatkov ter učinkovitemu zajemu slikovnega gradiva, aplikacijo pa nadgradite tudi z dodatno funkcionalnostjo. Nalogo zaključite s prikazom uporabe prenovljene aplikacije.

Mentor:

viš. pred. dr. Igor Rožanc

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

Zamenjaj

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Peter Djukić,

z vpisno številko 63980339,

sem avtor/-ica diplomskega dela z naslovom:

PRENOVA APLIKACIJE ZA VODENJE RADIJSKIH IGER

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  viš. pred. dr. Igor Rožanc

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 19.06.2009 Podpis avtorja/-ice: _____

Zahvala

Zahvaljujem se mentorju dr. Igorju Rožancu za vso pomoč in koristne nasvete pri nastajanju diplomske naloge.
Posebej pa se zahvaljujem domačim, prijateljem in sodelavcem za vzpodbudo med študijem.

Kazalo

Povzetek 9
Abstract 10
1. Uvod 11
2. Uporabljene tehnologije in orodja 12
2.1. Microsoft Visual Basic 6 12
2.2. Microsoft Access 13
2.3. Podatkovna baza Oracle 13
2.4. Ponovni inženiring 14
2.4.1. Splošni model za ponovni inženiring 15
2.4.2. Obratni inženiring 16
2.4.3. Opisni inženiring 17
2.5. Pristopi ponovnega inženiringa 18
2.5.1. Pristop velikega poka 18
2.5.2. Postopni pristop 18
2.5.3. Evolucijski pristop 19
3. Okoliščine za razvoj programske rešitve 21
3.1. Podatkovni model 23
3.2. Zahteve prenove 24
3.2.1. Prenova dela s slikami in datotekami 24
3.2.2. Prenova uvoza slik 24
3.2.3. Varnost 25
3.2.4. Elektronska pošta 27
3.2.5. Zvočni zapis iger v MP3 formatu 27
3.3. Ugotovitve prenove 27
4. Izvedba prenove 28
4.1. Prenos podatkov 28
4.1.1. Postopek prenosa podatkov iz MS Access baze v Oracle podatkovno bazo 28
4.1.2. Priprava na prepis MS Word datotek iz datotečnega sistema v Oracle podatkovno bazo 30
4.2. Predelava programa 30
4.2.1. Sprememba programa za delo z dokumenti 31

4.2.2. Shranjevanje dokumenta 31
4.2.3. Delo s slikami 34
4.2.3.1. Shranjevanje slik 34
4.2.4. E-pošta 34
4.2.5. Zvočni zapis v MP3 formatu 36
5. Opis aplikacije 37
5.1. Iskanja 37
5.1.1. Preprosto iskanje 37
5.1.2. Iskanje po časovnem intervalu 38
5.1.3. Iskanje po nastopajočih 39
5.1.4. Kombinirano iskanje 39
5.1.5. Iskanje po trajanju 40
5.2. Slike 40
5.2.1. Implementacija v aplikaciji 41
5.3. Varnost 42
5.4. Dodane funkcionalnosti aplikacije 44
5.4.1. E-pošta 44
5.4.2. Predvajanje MP3 zapisa iger 47
6. Sklepne ugotovitve 49
Literatura 50

Kazalo slik

Slika 1: Prikaz okolja programskega jezika MS Visual Basic 6 12
Slika 2: Prikaz okolja MS Access 13
Slika 3: Opis splošnega modela programskega ponovnega inženiringa..... 16
Slika 4: Postopek obratnega inženirstva..... 17
Slika 5: Pristop velikega poka 18
Slika 6: Postopni pristop..... 19
Slika 7: Evolucijski pristop 20
Slika 8: Podatkovni model informacijske rešitve..... 23
Slika 9: Levo - postopek za odpiranje polja BLOB iz DB. Desno - postopek za shranjevanje datoteke v BLOB obliki..... 26
Slika 10: Nastavitev povezave preko ODBC-ja 28
Slika 11: Vzpostavitev povezave preko ODBC-ja 29
Slika 12: Bralni stavek za pridobitev vnosnih stavkov za podatke vseh tabel 29
Slika 13: Primer vnosnega stavka..... 30
Slika 14: Stavek za dopolnitev tabele..... 30
Slika 15: Procedura zapisiWordBlob 32
Slika 16: Procedura odpriWordBlob 34
Slika 17: Procedura poisciEmailNaslove 35
Slika 18: Procedura GetOutlookContacts..... 36
Slika 19: Osnovna maska aplikacije..... 37
Slika 20: Najdene igre glede na iskalni parameter 38
Slika 21: Maska za iskanje v časovnem intervalu 38
Slika 22: Prikaz dela s spustnimi seznami..... 39
Slika 23: Prikaz vpisanih podatkov v spustnem seznamu Trajanje..... 40
Slika 24: Ukaz »Uredi slike«..... 41
Slika 25: Maska za pregled slik..... 42
Slika 26: Meni uporabniki 43
Slika 27: Maska za dodajanje uporabnikov in njihovih pravic 43
Slika 28: Seznam uporabnikov aplikacije 44
Slika 29: Prikaz menija za urejanje naslovov elektronske pošte..... 45
Slika 30: Maska za ročni vnos elektronskih naslovov..... 45

Slika 31: Maska za pripravo elektronskega sporočila 46
Slika 32: Vpenjanje zvočnih zapisov..... 47
Slika 33: Maska za predvajanje zvočnih zapisov 48

Povzetek

Na Radiu Slovenija se je že v šestdesetih letih prejšnjega stoletja vzpostavil poseben sistem hranjenja radijskih iger. Zaradi težavne uporabe sistema je bila razvita aplikacija za vodenje arhiva radijskih iger. V diplomski nalogi so predstavljene značilnosti te aplikacije ter predvsem njena prenova in nadgradnja, s katero je bila zagotovljena ustrezna varnost podatkov.

Osnovna aplikacija je bila razvita v programskem jeziku Microsoft Visual Basic 6 s podatkovno bazo Microsoft Access. Aplikacija je omogočila predvsem hitrejše iskanje in boljši pregled med igrami ter bistveno pohitrila delo v celoti.

Zaradi težav pri zagotavljanju varnosti in potrebe po omejevanju dostopa do podatkov smo se odločili za prenovo osnovne aplikacije. Pri prenovi smo uporabili postopke ponovnega inženiringa, kjer smo na podlagi pregleda in analize obstoječih rešitev določili dele aplikacije, ki so zahtevali ustrezne spremembe.

Ključna težava je bilo zagotavljanje nadzora nad besedili in slikami iger, kar smo dosegli s hranjenjem le-teh v podatkovni zbirki Oracle. Ob tem smo prenovili tudi uvoz slik, aplikacijo pa smo nadgradili z možnostjo prenosa podatkov s pomočjo e-pošte ter predvajanjem radijskih iger v MP3 formatu.

Ključne besede: razvoj aplikacije, prenova, podatkovna baza, MS Access, Oracle, arhiv radijskih iger

Abstract

Radio Slovenia established a special system for storing the radio dramas in the sixties of last century. Due to the difficulties of using the system, an application for archiving the radio dramas was made. The thesis presents the characteristics of the application, particularly the renovation and upgrading, which provided adequate safety of data.

Original application was developed in the programming language Microsoft Visual Basic 6 with Microsoft Access database. Application has enabled faster searching, better overview of plays and it enabled faster work in general.

Because of then difficulties with safety and limiting access to data, we decided to upgrade the original application. For the upgrade, we have used the procedures of reengineering with which we have reviewed and analyzed existing solution, so we could determent the parts to be changed.

The key problem was the provision of control over the texts and pictures of radio dramas, which we have achieved by placing them in the Oracle database. Furthermore, we reengineered the importation of picture files. We have also upgraded the application with the possibility of sending data through e-mail and ability to play radio dramas in MP3 format.

Keywords: applications development, *reengineering*, database, MS Access, Oracle, radio drama archive

1. Uvod

V diplomski nalogi smo se posvetili aplikaciji, ki je namenjena za arhiviranje in vodenje radijskih iger. Po pregledu dosedanjega arhiviranja radijskih iger na Radiu Slovenija smo ugotovili, da arhiv iger na radiu vodijo ročno s pomočjo kartončkov. Tak način dela je očitno zastarel, počasen in nezanesljiv. Zato smo se odločili, da za arhiv izdelamo novo programsko rešitev. Z razvojem nove programske rešitve bi tako dobili sodoben in pregleden arhiv, s katerim bi bil dostop do podatkov hitrejši in natančnejši.

Zato smo pričeli z razvojem programske rešitve, ki smo jo poimenovali Antigona (Antigona po boju, išče svojega brata ...). S programsko rešitvijo smo hoteli doseči naslednje:

- arhiviranje obstoječih radijskih iger,
- dopolnjevanje in nadgrajevanje arhiva z novimi radijskimi igrami
- lažje iskanje obstoječih iger za radijski program,
- multimedijsko podporo za komunikacijo z različnimi mediji (časopisi, internet, ...) ter
- pregledno beleženje o predvajanju posameznih iger.

Za razvoj programske rešitve smo uporabili Microsoftovo razvojno orodje Visual Basic 6, za podatkovno bazo pa Microsoft Access. Besedila, napovednike in recenzije radijskih iger pa smo hranili v datotečnem sistemu. Vanj smo shranili tudi slike igralcev, režiserjev in ostalih udeležencev radijskih iger.

V nadaljevanju se bomo posvetili predvsem prenovi aplikacije, saj smo ugotovili, da ima prvotna programska rešitev pomanjkljivosti. Hranjenje v datotečnem sistemu se je izkazalo za pomanjkljivega in nevarnega. Dostop do datotek na datotečnem sistemu se ni dalo omejiti zgolj na aplikacijo, kar bi posledično lahko vodilo do zlorab ter do namernega ali nenamernega spreminjanja podatkov.

V pripravi na prenovo programske rešitve smo se seznanili s pristopi in načini prenove programske opreme. Ugotovili smo, da je za potrebe naše prenove potrebno uporabiti ponovni inženiring. Za uvedbo popravljenega sistema pa smo uporabili princip velikega poka.

Odločili smo se, da prvotno podatkovno bazo preselimo iz Microsoft Accessa v podatkovno bazo Oracle. S tem smo lahko prenesli datoteke datotečnega sistema v podatkovno bazo in tako zagotovili nadzorovan dostop do njih. Poleg tega pa smo se odločili, da aplikacijo dopolnimo še z možnostjo pošiljanja podatkov preko elektronske pošte za lažje komuniciranje z ostalimi medijskimi hišami. Uvedli smo še možnost shranjevanja in poslušanja zvočnih zapisov v MP3 formatu, s čimer smo želeli olajšati pripravo radijskega programa.

Cilji prenove programske rešitve so bili:

- nadzor nad MS Word datotekami iger,
- nadzor nad datotekami slik,
- prenova uvoza slik,
- varnost podatkov,
- prenos podatkov s pomočjo e-pošte ter
- predvajanje iger v MP3 formatu.

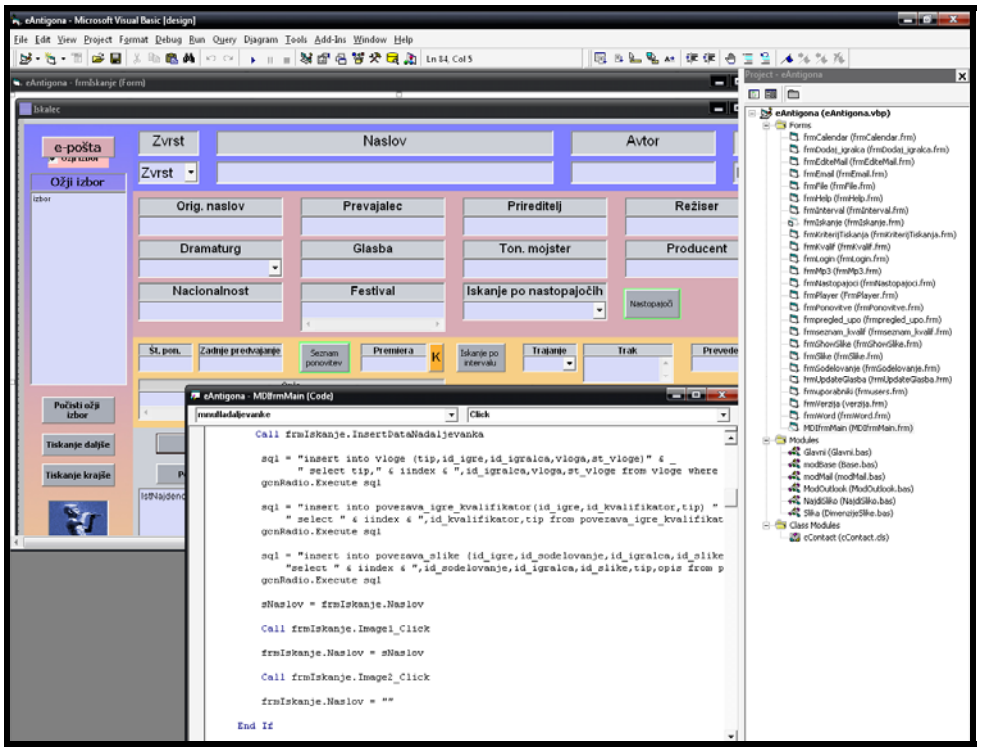
2. Uporabljene tehnologije in orodja

V tem poglavju bomo predstavili orodja, s katerimi smo se srečali pri razvoju aplikacije, ki je obravnavana v diplomski nalogi. Pomembno je vedeti, da je razvoj kakršnekoli aplikacije povezan s poznavanjem in uporabo različnih tehnologij.

2.1. Microsoft Visual Basic 6

Visual Basic je dogodkovni programski jezik tretje generacije (*ang. Third-generation event driven programming language*). Nastal je iz programskega jezika BASIC in omogoča hiter razvoj aplikacij (*ang. Rapid Application Development - RAD*) ter grafičnih vmesnikov (*ang. Graphical User Interface - GUI*). Prav tako omogoča dostop do podatkovnih baz s pomočjo DAO (Data Access Objects library from Microsoft), RDO (Remote Data Objects) in ADO (ActiveX Data Objects) tehnologij [5].

Med poslovnimi aplikacijami je Visual Basic eno najbolj uporabljenih razvojnih orodij. V raziskavi iz leta 2005 je kar 62% razvijalcev potrdilo uporabo Visual Basica na tak ali drugačen način [9].

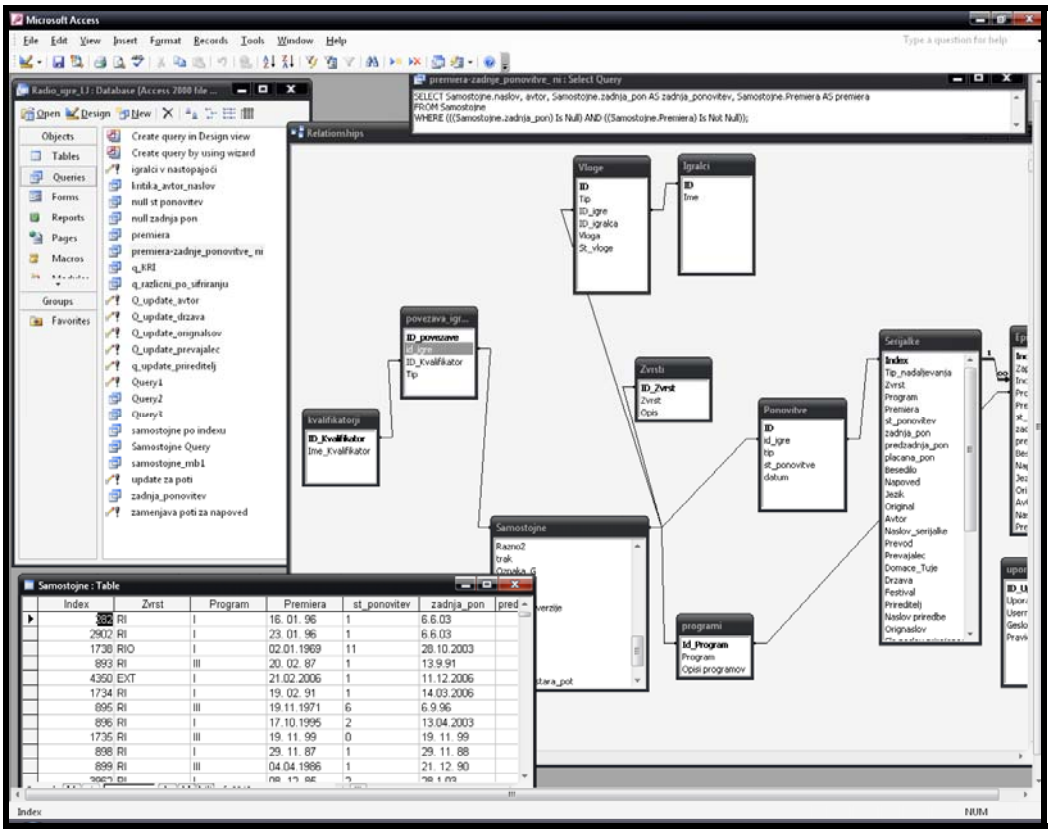


Slika 1: Prikaz okolja programskega jezika MS Visual Basic 6

MS Visual Basic je močno orodje za razvoj poslovnih programskih rešitev. V sodobnih časih je glavna slabost programskega jezika v tem, da ni objektno usmerjen programski jezik in s tem zaostaja za sodobnimi programskimi rešitvami.

2.2. Microsoft Access

Microsoft Access je sistem za upravljanje z relacijskimi podatkovnimi bazami (*ang. Relational Database Management System*). MS Access združuje Microsoft Jet Database Engine z grafičnim vmesnikom in razvojnimi orodji [3].



Slika 2: Prikaz okolja MS Access

V prvi različici programa smo se odločili za uporabo podatkovne baze v Microsoft Access. Microsoft Access smo izbrali, ker je bil program namenjen majhnemu številu uporabnikov in je ta rešitev zadostovala prvotnim zahtevam. MS Access je primeren za manjše število podatkov in ima omejeno uporabo SQL poizvedb v primerjavi z ostalimi podatkovnimi bazami.

2.3. Podatkovna baza Oracle

Podatkovna baza Oracle (*ang. Oracle Database*) je najpomembnejši produkt podjetja Oracle Corporation. Ta sistem za upravljanje z relacijskimi bazami podatkov je doživel že nekaj preimenovanj, uporabniki pa ga pogosto imenujejo kar Oracle.

Larry Ellison je s prijateljema in takratnima sodelavcema Bobom Minerjem in Edom Oatesom začel Software Development Laboratories (SDL) leta 1977. SDL je razvil prvo verzijo programske opreme Oracle. Ime Oracle izvira iz imena projekta CIE, na katerem je Ellison delal pred tem [6].

Oracle je napredna podatkovna baza, ki jo zaradi njenih sposobnosti uporabljajo predvsem večja podjetja.

Oracle podatkovna baza omogoča veliko možnosti:

- napredna varnost (*ang. Advanced Security*),
- podatki o vsebini podatkovne baze (*ang. Content database*),
- podatkovni sef (*ang. Database Vault*),
- podatkovno rudarjenje (*ang. Data Mining (ODM)),*
- varnostna oznaka (*ang. Label Security*),
- paketni managment (*ang. Management Packs*),
- podatkovno skladišče – Oracle OLAP,
- izgradnja podatkovnega skladišča (*ang. Oracle Warehouse Builder*),
- testiranje aplikacij (*ang. Oracle Real Application Testing*)
- in drugo.

Delo s podatkovno bazo Oracle je precej zahtevnejše od dela z MS Access podatkovno bazo, vendar prednosti odtehtajo razliko. V tej podatkovni bazi je možno nadzirati vse od izvajanja poizvedb do analize obnašanja podatkovne baze, kar v rokah strokovnjaka bistveno izboljša odzivnost podatkovne baze.

2.4. Ponovni inženiring

Ponovni inženiring (*ang. Reengineering*) programske kode sta Chikofsky in Cross definirala kot analizo in spremembo obstoječega sistema, s katerim tega rekonstruiramo na nov način [2, 7].

Manj formalno je ponovni inženiring modificiranje programa, ki se vrši po tem, ko je bil izveden obratni inženiring (*ang. Reverse Engineering*). S tem postopkom dodamo funkcionalnost ali popravimo napake.

Čeprav procesu večkrat rečemo obratni inženiring bi bilo pravilneje reči, da je obratni inženiring analiziranje sistema, ponovni inženiring pa je posledično dopolnjevanje le-tega.

Ponovni inženiring je potemtakem pregledovanje, analiziranje in sprememba obstoječe programske rešitve z namenom, da jo prenovimo v novo obliko. Cilj je razumeti obstoječo programsko rešitev (posebnosti, obliko, rešitve) in jo nato preoblikovati tako, da dobimo boljšo funkcionalnost ali odzivnost. Cilj je tudi obrazložiti trenutno funkcionalnost in se pripraviti na možne nove funkcionalnosti.

Čeprav je vsak proces ponovnega inženiringa drugačen, obstajajo štirje splošni principi:

- priprava na funkcionalno izboljšavo (*ang. Preparation for functional enhancemen*),
- izboljšanje vzdrževanja (*ang. Improve maintainability*),
- migracija (*ang. Migration*) in
- izboljšana zanesljivost (*ang. Improve reliability*).

Čeprav ponovni inženiring ne bi smeli izvajati z namenom povečanja funkcionalnosti obstoječega sistema, pa je velikokrat uporabljen kot priprava nanj. Stari sistemi sčasoma (zaradi popravkov, napak in dopolnjevanj) postanejo čedalje težji in dražji za vzdrževanje. Koda nima več lepe in logične strukture, dokumentacije pa mogoče ni več ali pa je zastarela. Ponovni inženiring določi karakteristike obstoječega sistema, da ga lahko primerjamo z želenim sistemom.

Ko sistem raste in se razvija, se stroški vzdrževanja večajo, ker postaja vzdrževanje težko in časovno potratno. Cilj je torej ponovno zasnovati sistem z boljšo funkcionalnostjo modulov in bolj primernimi maskami. Prav tako z novo dokumentacijo poskrbimo za tekoč popis sistema in s tem olajšamo njegovo vzdrževanje.

Računalniška industrija se hitro razvija, nova strojna in programska oprema pa omogoča nove možnosti in s tem hitro izriva staro. Z novostmi se znanje ljudi usmerja na nove tehnologije, s tem pa za sabo pušča malo ljudi, ki vzdržujejo stare sisteme. V relativno kratkem času proizvajalci prenehajo s podporo programskih rešitev, strojna oprema pa postaja redka in draga. Še pomembnejša je nekompatibilnost starih sistemov z novimi. Tako se podjetja z delujočimi in sicer povsem ustreznimi programskimi rešitvami, znajdejo v situaciji, ko morajo preiti na novo strojno opremo, operacijski sistem ali jezik.

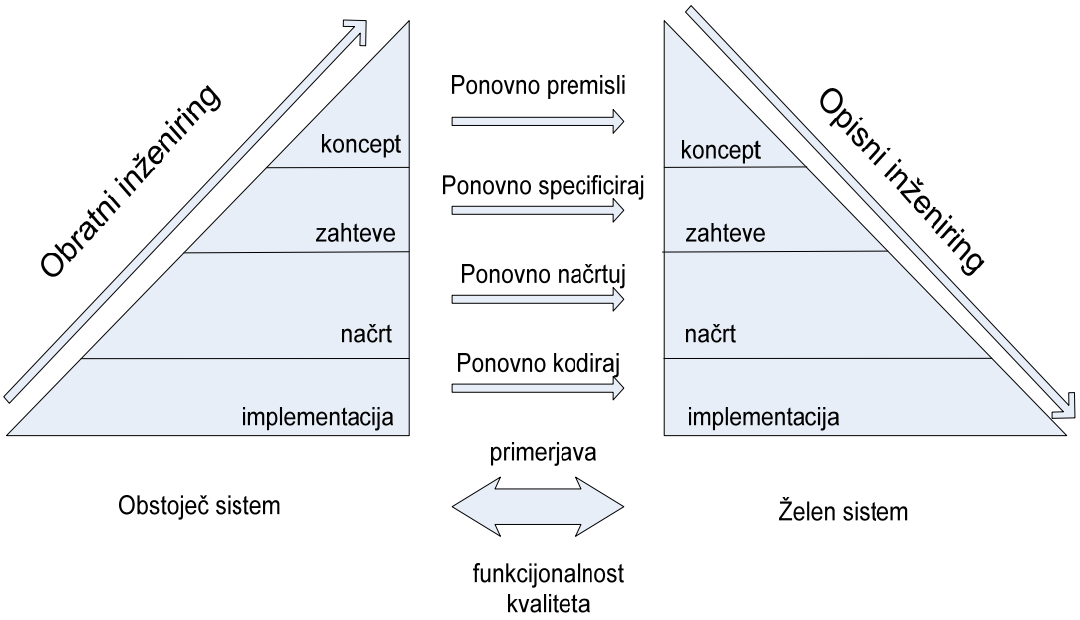
Četrti princip ponovnega inženiringa je doseči večjo zanesljivost. Velika možnost je, da je med popravki in dopolnjevanji prišlo do stanja, ko en popravek privede do kopice novih problemov. S tem se zanesljivost sistema postopoma znižuje do točke, ko ta ni več sprejemljiva.
.

2.4.1. Splošni model za ponovni inženiring

Ponovni inženiring se začne z izvorno kodo obstoječega sistema in se konča z izvorno kodo želenega sistema.

Ta proces je lahko povsem preprost, recimo uporaba programa, ki nam prevede izvorno kodo iz enega jezika v drugi (FORTRAN v C) ali prenos iz enega operacijskega sistema v drugega (DOS v UNIX). Po drugi strani pa je to lahko tudi zelo zahteven proces, kjer uporabimo obstoječo kodo, da prenovimo dizajn, definiramo funkcionalnosti v obstoječem sistemu in jih primerjamo s funkcionalnostjo želenega sistema. Umaknemo dele, ki so nepotrebni, prenovimo in preoblikujemo sistem in na koncu vse zapišemo v kodi želenega sistema.

Slika 3 opisuje splošni model programskega ponovnega inženiringa, ki predstavlja proces vseh nivojev na osnovi posnetka (*ang. abstraction*), ki je bil narejen pri razvoju.



Slika 3: Opis splošnega modela programskega ponovnega inženiringa

Model na sliki 3 se nanaša na tri dele ponovnega inženiringa: posnetek, spremembe in čiščenje [8].

Posnetek je abstraktni zapis sistema, ki nastane z zaporednimi zamenjavami obstoječih podrobnih informacij z informacijami, ki so bolj abstraktne. Posnetek povzroči poudarjanje nekaterih značilnosti sistema, s tem pa zakriva informacije o drugih. To imenujemo obratno inženirstvo in je povezano s podprocesi, orodji in tehnikami.

Sprememba je izvedba ene ali več sprememb sistema brez spreminjanja posnetka, vključno z dodajanjem, brisanjem in spreminjanjem podatkov, ne pa funkcionalnosti.

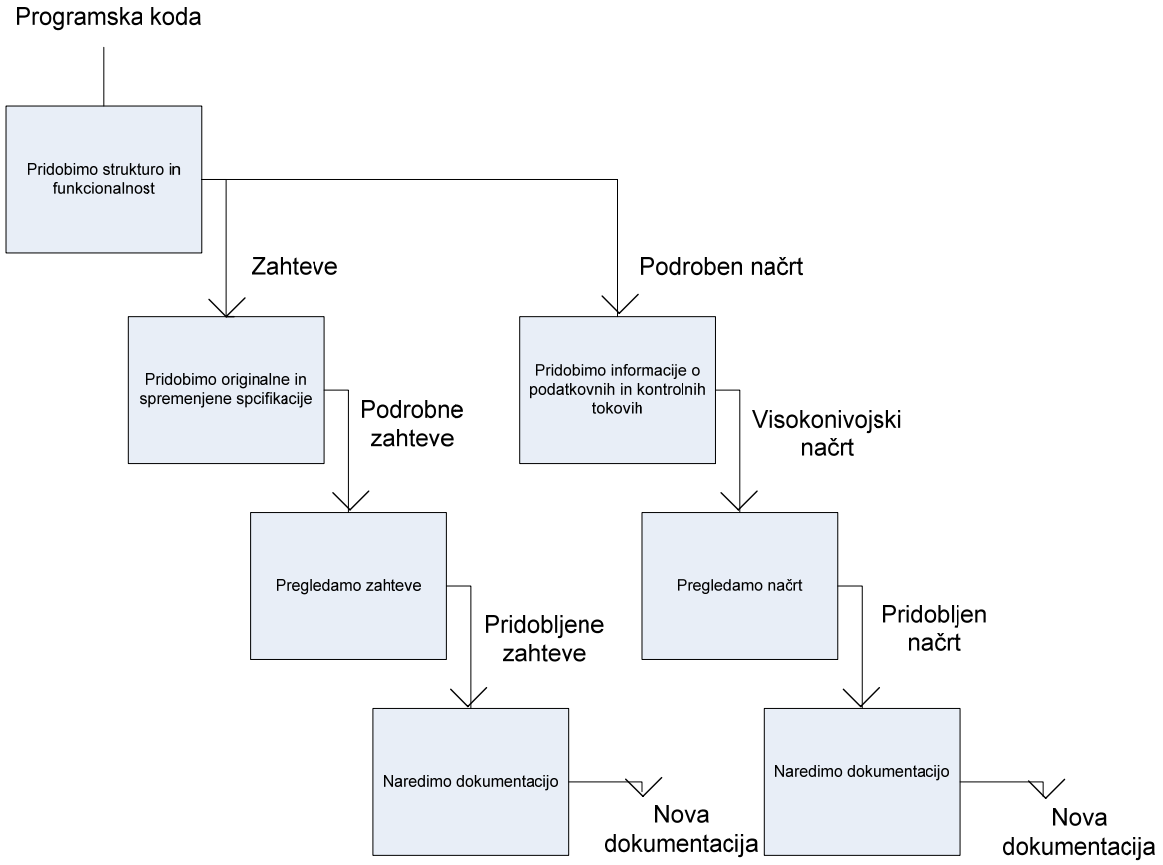
Čiščenje je postopno zmanjšanje abstraktnosti posnetka in nadomestitev obstoječega sistema informacij s podrobnejšimi podatki. Slednje imenujemo opisni inženiring (*ang. forward engineering)* in je podobno razvoju nove programske kode, vendar z nekaterimi izboljšavami procesa. Da bi spremenili sistemske karakteristike, se delo opravi na ravni posnetka, pri katerem so informacije o sistemu nedvoumno določene. Za prepis obstoječe kode v nov programski jezik ne potrebujemo ponovnega inženiringa, sprememba (*ang. Recoding*) je narejena na izvedbenem delu.

2.4.2. Obratni inženiring

Obratni inženiring je proces analize sistema, s katerim želimo identificirati komponente sistema, njihova notranja razmerja ter ugotovitve predstaviti v drugem ali višjem nivoju abstrakcije [8]. V obratnem inženirstvu je potrebno na novo povzeti potrebe, glavno obliko, strukturo in vsebino obstoječega sistema. Poleg popisa tehničnih odnosov in medsebojnih vplivov, je potrebno pridobiti tudi informacije o aplikaciji in procesih, ki so se izkazala za dobra. To nas privede do abstrakcije, ki je manj usmerjena v implementacijo. Glavna ideja obratnega inženiringa je pridobitev alternativnih pogledov, izgubljenih informacij ali

odkrivanje napak in omogočiti uporabo teh informacij. Učinkovitost tega postopka dejansko odloča o uspehu projekta. Z obratnim inženirstvom ne spreminjamo sistema ali naredimo novega, temveč z njim le pregledamo proces brez spreminjanja celotnega sistema.

Proces obratnega inženiringa je prikazan v sliki 4. Prične se z izvlečkom zahtev in podrobnih informacij iz programske kode in obstoječe dokumentacije. Ustvari se dokument z zahtevami ter abstrakten načrt, ki se ga prikaže s pomočjo diagramov podatkovnih tokov. Vse to se nato pregleda, da se ugotovi pravilnost in usklajenost.



Slika 4: Postopek obratnega inženirstva

2.4.3. Opisni inženiring

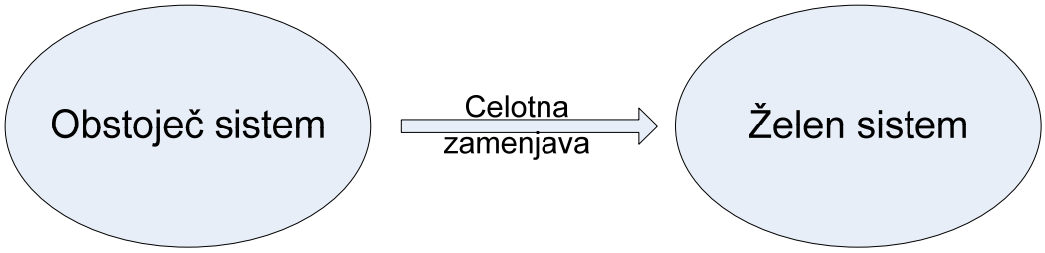
S postopkom opisnega inženiringa izdelamo želen sistem s premikanjem od abstraktnega proti konkretnemu nivoju opisa. Postopno spuščanje se odvija s postopno zamenjavo opisa z bolj podrobnim. To premikanje navzdol je pravzaprav standardno premikanje naprej kot ga poznamo pri običajnem razvoju programske opreme, zato izraz opisni inženiring. Opisni inženiring se premika od višjih nivojev abstrakcije logičnega sistema in ga postopoma implementira v fizično implementacijo sistema, zato so tovrstni projekti izpostavljeni večjemu tveganju pri uvedbi sprememb ali dodajanju novih zahtev [8].

2.5. Pristopi ponovnega inženiringa

Obstajajo trije različni pristopi za ponovni inženiring programskih rešitev, ki se razlikujejo v obsegu in stopnji zamenjave obstoječega sistema s ciljnim sistemom. Vsak od njih ima svoje prednosti in slabosti.

2.5.1. Pristop velikega poka

Pristop velikega poka (*ang. Big Bang Approach*) vpeljuje nov sistem v okolje v celoti. Ker ni potrebno razvijati vmesnikov med starim in novim sistemom, nam ni potrebno delati in vzdrževati obeh sistemov.

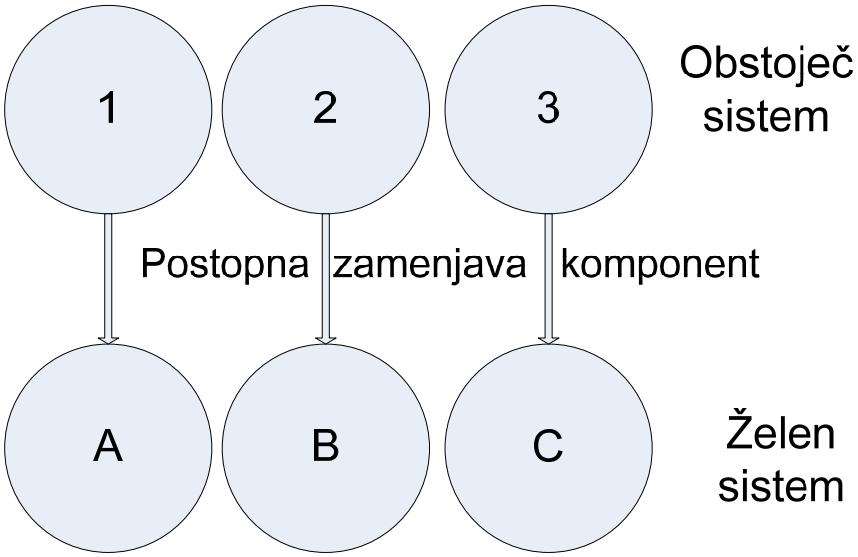


Slika 5: Pristop velikega poka

Slabost tega pristopa je, da je včasih tvegan, precej zahteven in zato ni vedno najbolj primeren. Če je sistem velik, hitro porabimo preveliko človeških virov. Prav tako se nam lahko zgodi, da je čas za razvoj takega sistema zelo dolg. Veliko tveganje predstavlja tudi obdobje uvajanja novega sistema, saj bi običajno moral sistem delovati vzporedno s starim, da preverimo njegovo delovanje. To obdobje dvojnega delovanja je lahko težko in drago. Težko pa je tudi preklopiti na nov sistem, saj je možno, da se je star med tem že spremenil. Te spremembe pa se morajo odražati tudi v novem sistemu.

2.5.2. Postopni pristop

V postopnem pristopu (*ang. Incremental Approach ali Phase-out*) so novi deli sistema postopoma ponovno načrtovani kot nove verzije z novo funkcionalnostjo. Ponovni inženiring je razdeljen na kose glede na obstoječe sistemske dele.



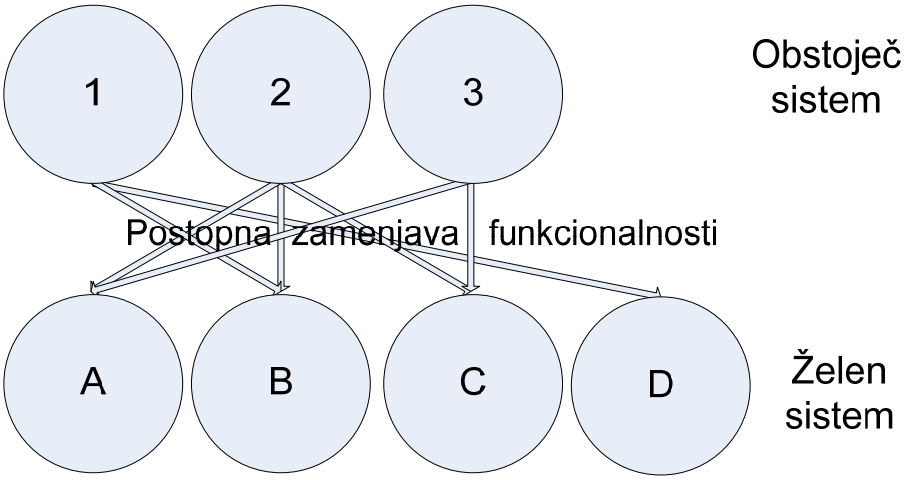
Slika 6: Postopni pristop

Prednost tega pristopa je, da so komponente narejene hitreje, sledenje in odpravljanje napak pa je lažje, saj natančno vemo, kateri del je nov. Ker so vmesne verzije objavljene sproti, naročnik lahko hitro pregleda napredek in odkrije možne izgube funkcionalnosti. Prednost je tudi v tem, da se lažje soočamo s spremembami starega sistema, saj spremembe komponent, ki jih ne obnavljamo, ne vplivajo na trenutne komponente.

Slabost tega postopka pa je v daljšem času razvoja, z običajno mnogimi vmesnimi verzijami, ki jih je potrebno dobro nadzirati. Pomanjkljivost je tudi to, da ne moremo spremeniti celotne strukture programa, ampak zgolj določene dele komponent, katere imamo namen reprogramirati. To terja podroben pregled komponent obstoječega sistema in drago planiranje strukture želenega sistema. Ima pa ta pristop manjše tveganje od pristopa velikega poka, saj se z začetkom ponovnega inženiringa vsakega dela kode posebej, lahko posvetimo le enemu delu.

2.5.3. Evolucijski pristop

Pri evolucijskem pristopu (*ang. Evolutionary Approach*) podobno kot pri postopnem pristopu dele originalnega sistema zamenjamo z novimi ponovno razvitimi deli. Razlika je v tem, da v tem postopku dele sistema izberemo glede na funkcionalnost in ne na podlagi strukture obstoječega sistema. Nov sistem je zgrajen na podlagi funkcionalno vezanih delov starega sistema. Pristop dovoli razvijalcem, da se osredotočijo na prenavljanje funkcionalnosti, ne glede na to, kje v sistemu se ta nahaja. Kot je razvidno s slike 7 komponente trenutnega sistema razbijemo po funkcionalnosti in jih ponovno oblikujemo v nove komponente.



Slika 7: Evolucijski pristop

Prednost evolucijskega pristopa je predvsem modularna zasnova, kar zmanjša področja za posamezne komponente. Ta pristop deluje dobro pri pretvorbi v objektno usmerjene tehnologije. Pomanjkljivost tega pristopa je recimo ta, da je podobne funkcionalnosti potrebno najti v celotnem sistemu, da bi jih združili v skupno funkcionalno enoto. Možne so tudi težave z vmesnikom ter zmanjšano odzivnostjo, saj sistem preoblikujemo funkcijsko in ne arhitekturno.

3. Okoliščine za razvoj programske rešitve

Za razvoj aplikacije smo se odločili po pregledu dela z arhivom radijskih iger na Radiu Slovenija. Ob pregledu dela smo ugotovili, da se igre hranijo na trakovih v skladišču radija, evidenca iger pa se nahaja na kartončkih v omari. Vsaka od iger ima svojo enolično označbo, s katero je označen tudi trak v skladišču. Napovedi, recenzije in besedila se hranijo v ločenih datotekah v datotečnem sistemu, do katerih imajo dostop vsi.

Za lažje razumevanje pogojev dela bomo v nadaljevanju opisali primer iskanja igre pred uvedbo informacijske rešitve.

Če je zaposleni na radiu želel najti podatke o radijski igri je bilo potrebno najprej iz velike količine kartončkov v omari najti iskano igro. S tem je pridobil njeno označbo in je lahko nadaljeval z delom. V Wordovih datotekah je imel v različnih mapah shranjena besedila, napovednike ali recenzije igre, shranjene s to številko. Za podatke o nastopajočih je moral odpreti četrto datoteko, kjer so bili vpisani nastopajoči in ostali udeleženci pri igri. V primeru, da bi igro želel še poslušati, je moral izpolniti obrazec za skladišče, iz katerega so mu nato magnetofonski zapis našli in posredovali.

Iz primera je razvidno, da je bil sistem zastarel in časovno zelo potraten.

Z razvojem programske rešitve smo hoteli dobiti sodoben in pregleden arhiv, s katerim bi bilo do podatkov možno dostopati veliko hitreje in natančneje. Za cilje programske rešitve smo določili:

- arhiviranje obstoječih radijskih iger,
- dopolnjevanje in nadgrajevanje arhiva z novimi radijskimi igrami,
- lažje iskanje obstoječih iger za radijski program,
- multimedijsko podporo za komunikacijo z različnimi mediji (časopisi, internet, ...) ter
- pregledno beleženje o predvajanju.

Po proučitvi zahtev smo se odločili za razvoj aplikacije z uporabo Microsoft Visual Basic 6 v podpori Microsoft Accessa kot podatkovne baze. Prvotno smo si zamislili preprosto aplikacijo za arhiviranje in iskanje iger, vendar se je kmalu pokazalo, da so možnosti razvoja večje in se je začelo delo na izboljšavah funkcionalnosti programa. Arhiv radijskih iger, ki je oplemeniten z dodatnimi funkcionalnostmi, omogoča:

- preprosto iskanje podatkov po različnih kriterijih,
- tiskanje podatkov v različnih načinih izpisa,
- pripenjanje različnih dokumentov k posameznim igram,
- pripenjanje zvočnih zapisov k posameznim igram,
- samodejno povezavo nastopajočih z njihovimi slikami,
- povezavo z elektronsko pošto in pošiljanje tekstovnega, slikovnega ter zvočnega gradiva iz programa.

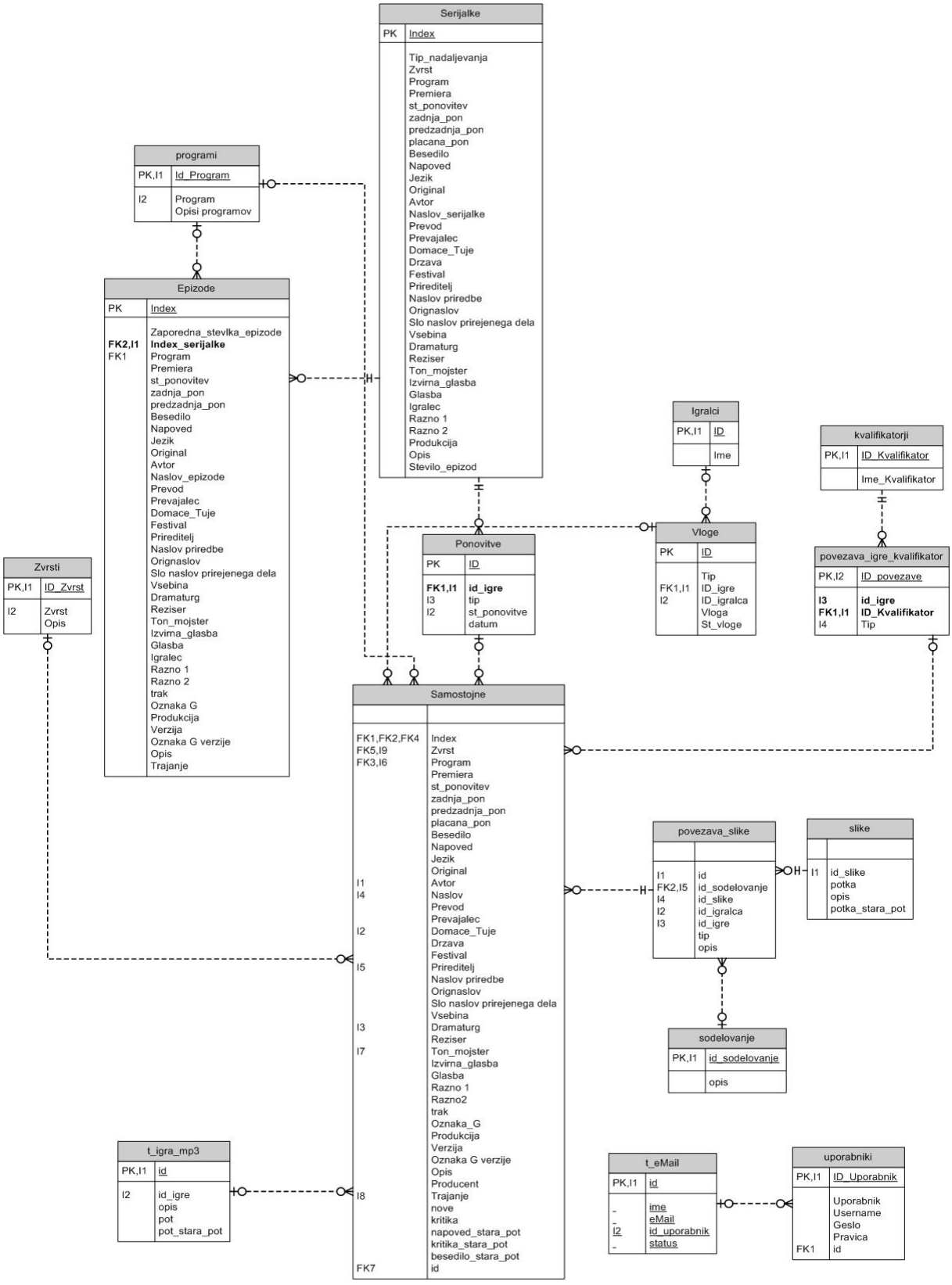
Aplikacija je bila razvita tako, da kar najbolj olajša delo in prihrani čas. Datoteke, ki jih je bilo pred tem potrebno iskati na različnih koncih datotečnega sistema, so sedaj hitro in enostavno dostopne iz aplikacije, predvsem pa se je izboljšalo iskanje in poslušanje. Če se je za primer pred uvedbo aplikacije želelo poiskati igre v katerih je, recimo nastopal Boris Cavazza, je bilo to potratno in zahtevno delo, saj je bilo potrebno ročno odpreti in pregledati vse datoteke z nastopajočimi in izločiti tiste, v katerih iskani ni nastopal. Poleg dolgotrajnosti postopka pa je obstajala velika verjetnost slabega rezultata zaradi človeškega faktorja.

S pomočjo aplikacije je sedaj ta proces nadvse enostaven. V aplikacijo zgolj vpišemo ime iskanega nastopajočega in aplikacija nam vrne vse zadetke, ki se ujemajo z iskalnim pogojem.

Za primerjavo z zgoraj navedenim primerom iskanja bomo opisali še primer iskanja igre s pomočjo nove aplikacije. V aplikacijo vpišemo naslov (ali celo delni naslov) igre in dobimo seznam vseh ustreznih zadetkov. Ko izberemo želeno igro se nam prikažejo (na enem mestu) vsi podatki o igri. Tako lahko s klikom dostopamo do napovednikov, recenzij in besedila, vidimo pa tudi vse nastopajoče in sodelujoče. Za vsako igro imamo tudi kopijo avdio posnetka v MP3 formatu z možnostjo takojšnjega predvajanja kar znotraj programa. Za predvajanje na radiju je še vedno potrebno iz skladišča dobiti originalni magnetofonski zapis.

Z navedenimi primeri smo izpostavili le nekatere od prednosti uvedbe računalniško vodene evidence.

3.1. Podatkovni model



Slika 8: Podatkovni model informacijske rešitve

Podatkovni model informacijske rešitve je bil zasnovan za pokrivanje vseh potreb beleženja radijskih iger.

Šifranti, slike in nastopajoči se nahajajo v lastnih tabelah. Ker pa v igrah nastopa več nastopajočih, ki prav tako nastopajo v drugih igrah, jih s tabelo iger povezujemo s povezujočo tabelo. S tem omogočimo, da popravek šifranta, imena,... naredimo le na enem mestu, viden pa je v celotnem sistemu. V primeru besedil, kritik in zvoka pa povezovalnih tabel ne potrebujemo, saj je besedilo vedno namenjeno le eni igri. Podatkovni model podpira tudi beleženje predvajanosti posameznih iger (kdaj in kolikokrat), vezavo iger na radijski program in zvrst. Za potrebe nadzora pa imamo tabeli uporabnikov in pravic, s katerimi vsakemu uporabniku posebej določimo pravice uporabe podatkov znotraj programa.

3.2. Zahteve prenove

Z uvedbo programske rešitve smo dokazali, da je aplikacija bistveno pripomogla k boljši preglednosti in hitrejšemu delu s podatki. Pokazale pa so se tudi pomanjkljivosti, ki so bile posledica prevzetih obstoječih podatkovnih struktur, predvsem rešitev z datotečnim sistemom. Ko smo uvedli programsko rešitev arhiviranja in pregledovanja radijskih iger, smo omogočili dostopnost podatkov več uporabnikom, kakor je bilo to možno v preteklosti, vendar smo s tem tudi omogočili zlorabe. Zato smo prišli do ugotovitve, da je potrebno boljše poskrbeti za varnost in dostop do podatkov.

Iz tega razloga smo se odločili za prenovo sledečih področij programske rešitve:

- nadzor nad MS Word datotekami iger,
- nadzor nad datotekami slik,
- prenova uvoza slik,
- varnost podatkov,
- prenos podatkov s pomočjo e-pošte ter
- predvajanje iger v MP3 formatu.

3.2.1. Prenova dela s slikami in datotekami

V verziji pred ponovnim inženiringom smo delali z datotečnim sistemom. To je pomenilo, da so bile slike in datoteke shranjene na datotečnem sistemu, do katerega so imeli dostop vsi. V podatkovno bazo so se shranjevale le poti do slik in datotek datotečnega sistema. Zato smo se pri prenovi posvetili predvsem rešitvi tega problema. Za najboljšo rešitev se je izkazal prenos vseh datotek iz datotečnega sistema v podatkovno bazo Oracle. S tem razlogom smo podatkovni model dopolnili tako, da namesto poti do datoteke v datotečnem sistemu shranjujemo celo datoteko v obliki BLOB-a (*ang. Binary large object*) kar v tabeli.

3.2.2. Prenova uvoza slik

V prvotni verziji aplikacije smo si zamislili, da bi program samostojno uvažal slike v podatkovni model in s tem bistveno skrajšal čas za vnos velikega števila slik. S tem namenom

smo razvili funkcijo, ki je ob zagonu pregledala datotečni sistem in slike pripela nastopajočim.

Rešitev pa se je s časom pokazala za slabo, saj se je s povečevanjem količine slik začel povečevati tudi čas tega pripenjanja. S tem razlogom smo se odločili, da ta del funkcionalnosti prenovimo.

V fazi ponovnega inženiringa uvoza slik smo se temu problemu posebno posvetili. Ugotovili smo, da je možnih rešitev več:

- slike se shranjujejo v eno mapo in jih nato računalnik sam razvrsti na datotečni sistem ali
- sprememba podatkovne baze in s tem drug način dela s slikami.

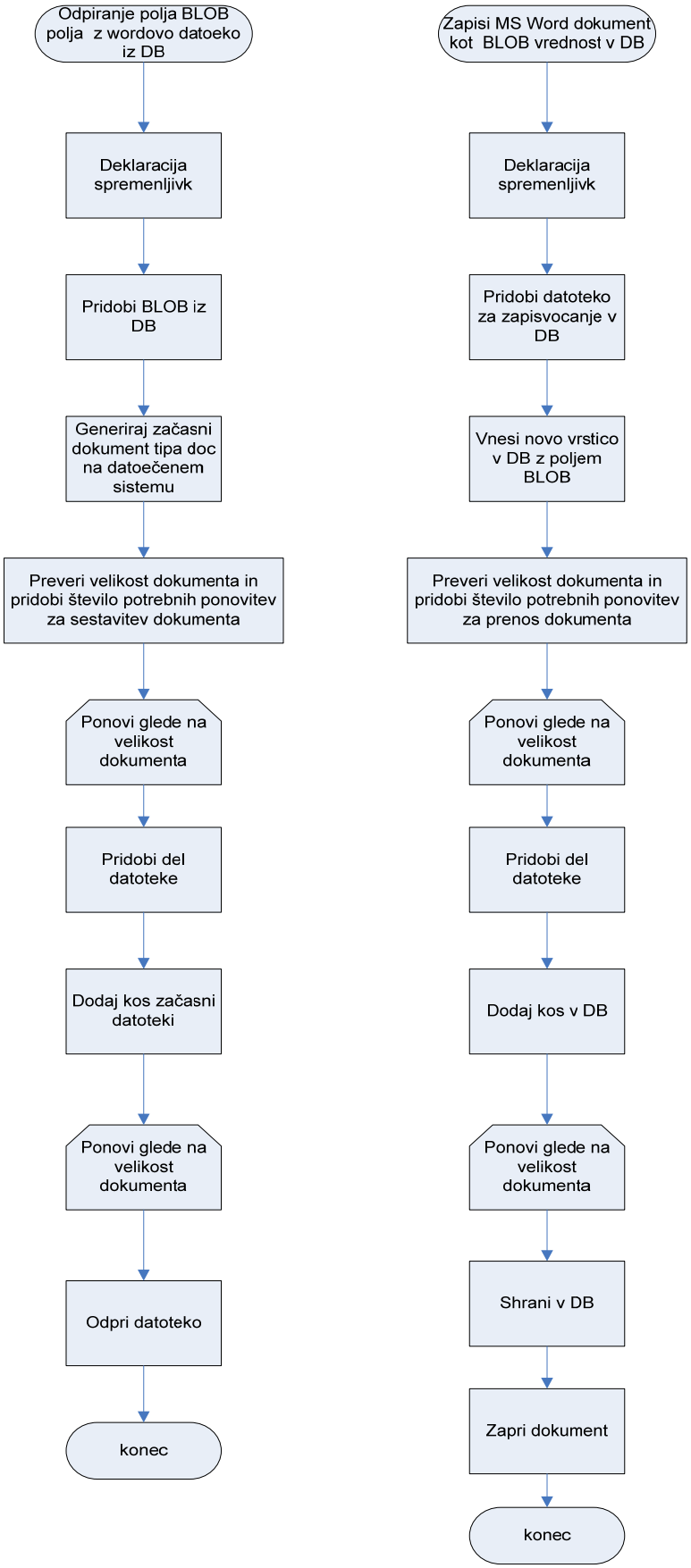
Ker smo se že pred tem odločili za uporabo Oracle podatkovne baze smo prišli do sklepa, da je najboljša rešitev prenos datotek iz datotečnega sistema v podatkovno bazo. S prenosom v podatkovno bazo smo omogočili tekoče in hitrejše delovanje funkcije osveževanja aplikacije z novimi slikami. Tako se pridobljene slike ne kopičijo več na datotečnem sistemu, temveč se na njem nahajajo le nove slike, ki še niso bile pripete nastopajočim. S tem se je čas pripenjanja znatno manjšal, saj ni bilo potrebno več pregledovati velikega števila datotek, da bi našli novo sliko.

Posledično se je zamenjala tudi logika funkcije. Sprva je bila zasnovano tako, da se je za vsakega nastopajočega poiskalo vse slike na datotečnem sistemu in s tem našlo tudi nove. S spremembo postopka izhajamo iz nove slike, za katero smo poiskali nastopajočega, jo prenesli v podatkovno bazo in jo iz datotečnega sistema izbrisali.

3.2.3. Varnost

Vprašanje varnosti se je zastavilo šele po implementaciji in uporabi prvotne aplikacije. Ugotovili smo, da je datotečni sistem ozko grlo tudi v varnostnem smislu. Dostopanja do njega nismo mogli kontrolirati, enako kot nismo mogli kontrolirati dostopa do podatkov informacijskega sistema radijskih iger. V prvotno aplikacijo smo sicer vgradili tri vrste uporabnikov z ustreznimi pravicami: Administrator, Dramaturg in Splošen uporabnik. Rešitev omogoča, da vsakemu od uporabnikov določimo različne funkcionalnosti in možnosti manipulacije s podatki. Splošni uporabnik ima zelo omejene pravice in lahko podatke le pregleduje. Rešitev pa ima za malo bolj poučenega uporabnika očitno šibkost, ker hranimo datoteke s podatki v datotečnem sistemu, se lahko do njih pride tudi po drugi poti. Tako podatki niso varni, saj jih oseba lahko kopira, spreminja in briše brez vednosti administratorja programa. To lahko privede do resnih napak v programu.

Ko smo začeli s prenovo sistema, smo se temu problemu temeljito posvetili in se odločili, da je najboljša rešitev prenos teh datotek v podatkovno bazo. Oracle podatkovna baza omogoča v svojih tabelah polja tipa BLOB. BLOB omogoča zapis datoteke v binarni obliki v polje tabele. S to rešitvijo smo dosegli, da je možen dostop do podatkov le preko programa in s tem omejili dostop do podatkov. Sedaj administrator določa, kdo lahko podatke vidi ali uporablja.



Slika 9: Levo - postopek za odpiranje polja BLOB iz DB. Desno - postopek za shranjevanje datoteke v BLOB obliki

Slika 9 prikazuje postopek shranjevanja in odpiranja podatkov v polja tipa BLOB.

3.2.4. Elektronska pošta

Radijske igre se predvajajo na programih Radia Slovenija, zato pripravljavci programa material izbranih iger pošiljajo različnim časopisom in drugim medijem. Iz tega razloga smo se odločili za uvedbo pošiljanja materiala kar iz aplikacije. S tem smo olajšali zbiranje podatkov in pohitrili delo, ki je potrebno za obveščanje medijev. Napisali smo funkcijo, s katero lahko uporabnik prenese ročno ali avtomatsko vse naslove, ki jih ima shranjene v svojem MS Outlook računu ter modul za pošiljanje elektronskih sporočil. Modul za pošiljanje elektronskih sporočil smo prilagodili pošiljanju materiala o igrah, da je kar najhitreje možno izbrati pravi material.

3.2.5. Zvočni zapis iger v MP3 formatu

Originalni zvočni zapisi o radijskih igrah se nahajajo na trakovih v skladišču Radia Slovenija in to otežuje hitro dostopnost uporabnikom. Prav tako večkratno poslušanje zmanjšuje kvaliteto zapisa na zvočnih trakovih. Zato smo se odločili, da dodamo možnost poslušanja iger v MP3 formatu. S tem smo omogočili, da se igre s trakov presnamejo v MP3 format in shranijo v podatkovno bazo aplikacije. Tako pridobimo na hitrosti, dostopnosti in varovanju kvalitete zapisa na trakovih.

3.3. Ugotovitve prenove

Iz opisa zgoraj omenjenih aktivnosti pri prenovi aplikacije se v vseh pogledih kaže moč podatkovne baze Oracle. Z uporabo specializiranih podatkovnih polj, ki jih manj sofisticirana podatkovna baza MS Access ne podpira, smo lahko z bistveno manjšimi napori zagotovili kar nekaj izboljšav.

Wordove datoteke in slike smo prenesli v podatkovno bazo in s tem smo lahko z večjo gotovostjo zagotovili verodostojnost podatkov.

Problem varnosti in kontroliranja dostopa do podatkov, se je rešil pravzaprav kar sam. Čeprav smo že v prvotni verziji razmišljali o različnih pravicah, ki bi jih lahko uporabniki programa imeli, pa nismo mogli z gotovostjo reči, da je okolje varno pred zlorabami. S prenosom datotek v program smo dosegli, da je dostop zares kontroliran.

Ugotovili smo, da je bila v našem primeru zamenjava podatkovne baze najučinkovitejša rešitev za dosego želenih sprememb. Sam postopek prenove pa je bil tudi časovno zelo ugoden. Potrebno je bilo prepisati podatke, napisati manjšo aplikacijo, ki je datoteke pravilno prenesla v podatkovni model in opozorila na morebitne napačne podatke. Sprememba načina hranjenja datotek pa je od nas terjala le manjše spremembe v kodi.

4. Izvedba prenove

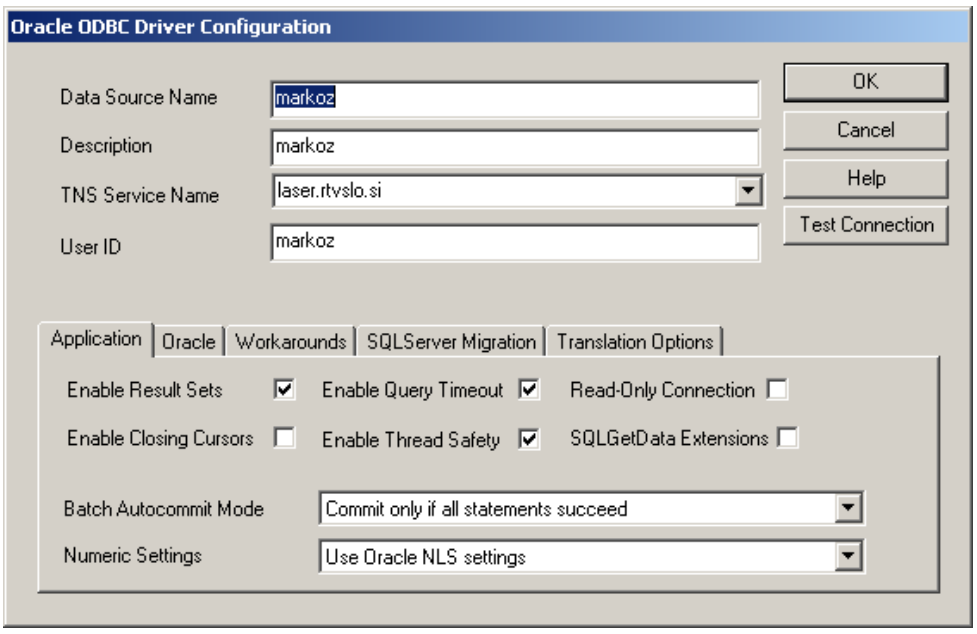
V tem poglavju se bomo posvetili opisu in izvedbi postopkov, ki so potrebni za izvedbo prenosa podatkov v podatkovno bazo Oracle. Prav tako bomo opisali potrebne spremembe kode za delo z dokumenti in slikami.

4.1. Prenos podatkov

Prvi korak pri prenovi je bil prenos podatkov iz MS Access baze v Oracle podatkovno bazo [1, 4]. V tem koraku je bilo potrebno prenesti cel podatkovni model, strukturo tabel in podatke, ki so se nahajali v njih.

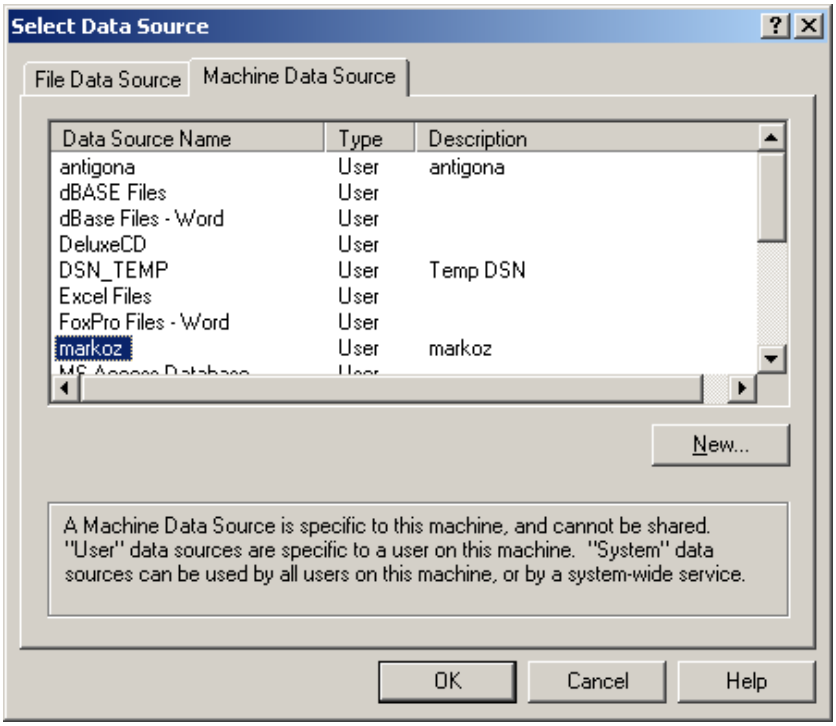
4.1.1. Postopek prenosa podatkov iz MS Access baze v Oracle podatkovno bazo

MS Access baza omogoča izvoz podatkov (*ang. Data Export*) iz tabel. Za prenos tabel iz Accessa v Oracle smo naredili ODBC (*ang. Open Database Connectivity*) podatkovni vir (*ang. data source*) povezavo z Oracle bazo.



Slika 10: Nastavitev povezave preko ODBC-ja

V MS Access-u smo nato uporabili opcijo prenos preko ODBC povezave in tako prenesli podatke iz MS Access baze v Oracle bazo.



Slika 11: Vzpostavitev povezave preko ODBC-ja

Po izvedbi prenosa podatkov se podatki nahajajo v Oracle bazi, vendar so ti podatki neurejeni in s tem še neuporabni v samem programu. Eden od problemov je tudi, da se ob prenosu, imena tabel vnesejo z malimi črkami.

Postopek urejanja podatkov bi bilo možno narediti na več načinov. Zaradi poznavanja SQL -a (*ang. Structured Query Language*) smo se odločili, da napišemo spodnjo poizvedbo (slika 12). Rezultat poizvedbe so vnosni stavki za tabele, ki so bile prenesene iz Access-a.

Z uporabo poizvedbe smo dosegli, da z uporabo enega stavka pridobimo vnosne stavke za vse tabele v MS podatkovnem sistemu, s čemer je prenos podatkov postal bistveno hitrejši in lažji.

```
SELECT 'insert into ' ||  upper(tname) || ' select * from "' ||
          INITCAP(upper(tname)) || '"; '
  FROM tab
GROUP BY upper(tname)
  HAVING COUNT(*) > 1
```

Slika 12: Bralni stavek za pridobitev vnosnih stavkov za podatke vseh tabel

Razlaga povpraševanja: Oracle vsebuje sistemsko tabelo TAB, v kateri se hrani spisek vseh uporabniških tabel v sistemu. S stikanjem besedila za vnosni stavek in pridobljenimi imeni tabel iz tabele TAB smo sestavili povpraševanje katerega rezultat so vnosni stavki za vse tabele v sistemu.

S to poizvedbo smo dobil vnosne stavke oblike:

```
insert into EPIZODE
select *
from "Epizode";
```

Slika 13: Primer vnosnega stavka

Rezultat opisane poizvedbe so vnosni stavki za vse tabele v sistemu. Stavke smo shranil in jih sprožili kot paket. S tem smo iz prenesenih tabel dobil nove tabele z vsemi podatki.

4.1.2. Priprava na prepis MS Word datotek iz datotečnega sistema v Oracle podatkovno bazo

Kot vemo smo se odločili, da bomo Wordove datoteke, z besedili, napovedmi in kritikami iger prenesli iz datotečnega sistema v podatkovno bazo. Tabelo SAMOSTOJNE smo dopolnili s tremi polji tipa BLOB.

```
alter table samostojne
add besedilo_blob blob

alter table samostojne
add napoved_blob blob

alter table samostojne
add kritika_blob blob
```

Slika 14: Stavek za dopolnitev tabele

V ta polja smo nato prepisali binarni zapis dokumentov na datotečnem sistemu.
Za potrebe prenosa smo napisali manjši program, ki za vsako igro prebere in pretvori izbrano besedilo v binarni zapis ter ga shrani v bazo. Iz praktičnih razlogov program prenesene in shranjene datoteke začasno prepiše na drugo lokacijo datotečnega sistema. S tem naredimo varnostno kopijo vseh datotek in ugotovimo morebitne napake pri postopku. Enak postopek smo naredili tudi za slike.

S tem postopkom smo izvedli celoten prepis podatkov iz Accessove podatkovne baze in datotečnega sistema v Oraclovo podatkovno bazo.

4.2. Predelava programa

V tem podpoglavju bomo opisali potrebne predelave izvorne kode, s čimer smo zagotovili delovanje z novo podatkovno bazo.

4.2.1. Sprememba programa za delo z dokumenti

V programu je bilo za delo z dokumenti potrebno spremeniti dva dela kode:

- shranjevanje in
- prikaz

4.2.2. Shranjevanje dokumenta

Program je v predhodni verziji shranjeval le poti do dokumentov v datotečnem sistemu.

Prva potrebna sprememba je bila shranjevanje dokumenta. Za to je bilo potrebno napisati novo funkcijo za shranjevanje dokumenta v bazo kot BLOB polje.

Napisali smo proceduro ***zapisiWordBlob***, katero se kliče s tremi parametri: z enolično številko igre, za katero shranjujemo dokument, poljem ali tematiko shranjenega dokumenta ter potjo dokumenta. Procedura poišče ustrezno polje v tabeli ter ga odpre za dopolnjevanje. Nato odpre datoteko, ki jo želimo vnesti v podatkovno bazo v binarnem formatu in jo po kosih zapiše v tabelo. Procedura je zapisana na sliki 15.

```
Sub zapisiWordBlob(id_igre as Integer, polje as String, pot as Sting)
'// Deklaracija spremenljivk
Dim rsKontrola  As New ADODB.Recordset
Dim rsZapis     As New ADODB.Recordset
Dim FileLength As Long
Dim NumBlocks As Integer
Dim LeftOver As Long
Dim i As Integer
Dim ByteData() As Byte
Dim SourceFile As Integer
Const BlockSize = 32768
Dim sFile As String

   '// Lovljenje napak
   On Error GoTo ErrorHandler
   '// Če pot ni podana zapustim proceduro
   If Trim(pot) = "" Then Exit Sub
   '// Kazalec miši nastavim na uro
   MousePointer = 11
   '//Stavek za vnos  blob-a v tabelo t_izt_dopis
   sql = "Select " & polje & "_blob," & polje & " from samostojne where """ &
              "INDEX" & """ = " & id_igre
   '//  Odprem novo transakcijo na bazi
   DE.POVEZI.BeginTrans
   gTrans = True

    '// Preverim stanje recordset-a ga zaprem, če je odprt in ponovno odprem z
        z  zgornjim selec tavkom
   If rsZapis.State = 1 Then rsZapis.Close
   rsZapis.CursorType = adOpenKeyset
   rsZapis.LockType = adLockOptimistic
   rsZapis.Open sql, DE.POVEZI, , , adCmdText
     '//   Spreminjanje zapisa v tabeli
    SourceFile = FreeFile
    If polje = "besedilo" Then
            rsZapis!Besedilo = "Besedilo shranjeno"
    ElseIf polje = "kritika" Then
            rsZapis!Kritika = "Kritika shranjena"
    ElseIf polje = "napoved" Then
             rsZapis!Napoved = "Napoved shranjena"
    End If
```

```
    '// Odprem wordov dokument
    Open pot For Binary Access Read As #SourceFile
    FileLength = LOF(SourceFile)
    '// Če je datoteka manjša od 15000 potem postopek prekinem in naredim
       rollback na bazi
     If FileLength < 15000 Then
                MsgBox "Ne dela prenos bloba!"
                DE.POVEZI.RollbackTrans
                gTrans = False
                Exit Sub
     End If
     '// Preberem datoteko in jo pretvorim v binarno
     NumBlocks = FileLength / BlockSize
     LeftOver = FileLength Mod BlockSize
     ReDim ByteData(LeftOver)
     Get SourceFile, , ByteData()
      If polje = "besedilo" Then
            rsZapis!besedilo_blob.AppendChunk ByteData()
      ElseIf polje = "kritika" Then
            rsZapis!kritika_blob.AppendChunk ByteData()
      ElseIf polje = "napoved" Then
            rsZapis!napoved_blob.AppendChunk ByteData()
      End If
      ReDim ByteData(BlockSize)
      '// Datoteko berem po kosih in jih dodajam v tabelo
      For i = 1 To NumBlocks
            Get SourceFile, , ByteData()
            If polje = "besedilo" Then
                  rsZapis!besedilo_blob.AppendChunk ByteData()
            ElseIf polje = "kritika" Then
                  rsZapis!kritika_blob.AppendChunk ByteData()
            ElseIf polje = "napoved" Then
                  rsZapis!napoved_blob.AppendChunk ByteData()
            End If
    Next i
    rsZapis.Update
    '// Ker je bilo postopek narejen brez napake potrdim spremembo podatkov
       na bazi
     DE.POVEZI.CommitTrans
     gTrans = False

    '// Kazalec miši postavim na puščico
    MousePointer = 1
    '// Zapustim proceduro
    Exit Sub
    '// V primeru napake prekinem narejene spremembe na bazi in obvestim
       uporabnika o napaki
ErrorHandler:
       If gTrans = True Then
          DE.POVEZI.RollbackTrans
             gTrans = False
                      MsgBox "Napaka pri vnosu dokumenta!", vbInformation,
                        "Dokument ni vnesen"
        End If
End Sub
```

Slika 15: Procedura zapisiWordBlob

4.2.3. Branje dokumenta

Pri branju dokumenta je bilo prav tako potrebno spremeniti logiko. V predhodni verziji programa je program prebral le pot, ki je bila shranjena v bazi in ta dokument odprl. Sedaj pa mora dokument iz baze pridobiti, ga sestaviti in prenesti na začasno lokacijo na datotečnem sistemu in ga šele nato prikazati. Za to smo napisali proceduro ***odpriWordBlob*** (slika16)**:**

```
Sub odpriWordBlob (id_igre, polje)
  '// Deklaracija spremenljivk
  Dim Dokum As New word.Application
  Dim FileLength As Long
  Dim NumBlocks As Integer
  Dim LeftOver As Long
  Dim i As Integer
  Const BlockSize = 16000 '32678
  Dim Datoteka As String
  Dim ByteData() As Byte
  Dim SourceFile As Integer
  Dim Odgovor As String
  Dim rsBlob As New ADODB.Recordset
  Dim PotTxt As String
  Dim PotDoc As String
  Dim rsZacasno As New ADODB.Recordset
  Dim IDDokumenta As Long
  Dim rsWord As New ADODB.Recordset
  Dim bIzvirna As Boolean              'ugotovim ali je iz tabele t_izt_dopis
  Dim Popravljanje As Boolean
  Dim RefVal
  Static Stevilka As Long

 'odpre BLOB
 SourceFile = FreeFile
'// Obravnavanje napake
 On Error GoTo ErrorHandler
 '// Kazalec miške nastavim na uro
 MousePointer = 11
 '// Stavek za pridobitev binarnega zapisa datoteke iz baze
 sql = "Select " & polje & "_blob from samostojne where """ & "INDEX"
  & """ = "   & id_igre

    If rsWord.State = 1 Then rsWord.Close
    rsWord.Open sql, DE.POVEZI, 3, 2

  '// Odprem začasni dokumetn v katerega bom pretočil binarni zapis
    PotDoc = "c:" & "\test\" & id_igre & ".doc"
    Open PotDoc For Binary Access Write As SourceFile
    FileLength = LOF(SourceFile)

    If polje = "besedilo" Then
       FileLength = rsWord!besedilo_blob.ActualSize
    ElseIf polje = "kritika" Then
       FileLength = rsWord!kritika_blob.ActualSize
    ElseIf polje = "napoved" Then
       FileLength = rsWord!napoved_blob.ActualSize
    End If

    NumBlocks = FileLength \ BlockSize
    LeftOver = FileLength Mod BlockSize
    ReDim ByteData(LeftOver)

   If polje = "besedilo" Then
       ByteData() = rsWord!besedilo_blob.GetChunk(LeftOver)
    ElseIf polje = "kritika" Then
       ByteData() = rsWord!kritika_blob.GetChunk(LeftOver)
    ElseIf polje = "napoved" Then
       ByteData() = rsWord!napoved_blob.GetChunk(LeftOver)
    End If
    Put SourceFile, , ByteData()
    '// Po kosih pridobivam binarni zapis datoteke ter ga sestavljam na datotečni  sistem
    For i = 1 To NumBlocks
       On Error Resume Next
       If polje = "besedilo" Then
          ByteData() = rsWord!besedilo_blob.GetChunk(BlockSize)
       ElseIf polje = "kritika" Then
          ByteData() = rsWord!kritika_blob.GetChunk(BlockSize)
       ElseIf polje = "napoved" Then
          ByteData() = rsWord!napoved_blob.GetChunk(BlockSize)
       End If
       Put SourceFile, , ByteData()
    Next i
```

```
   '// Odprem sestavljen dokument
  Dokum.Documents.Open FileName:=PotDoc
  Dokum.Visible = True

  Set Dokum = Nothing

  '// Kazalec miške nastavim na kazalec
  MousePointer = 1
  '// Zaprem začasno datoteko in recordset
 Close SourceFile
 If rsZacasno.State = 1 Then rsZacasno.Close
'// Zapustim proceduro
Exit Sub

 MsgBox "Napaka pri odpiranju dokumenta!", vbInformation, "Opozorilo!"

End Sub
```

Slika 16: Procedura odpriWordBlob

4.2.3. Delo s slikami

Podobno kot pri dokumentih, se je spremenilo tudi delo s slikami. Vsako sliko smo tudi tu prepisali v podatkovno bazo, kot binarno datoteko s pomočjo programa, ki smo ga napisali za te potrebe.

4.2.3.1. Shranjevanje slik

Shranjevanje slik je bilo potrebno popolnoma spremeniti, saj je bilo delo s slikami že v prvotni verziji programa drugačno kakor delo z dokumenti. Pri dokumentih si je uporabnik za vsako igro ali serijo izbral besedilo, kritiko in napoved ter jo ročno pripel k igri.

Pri slikah pa je uporabnik zaradi drugačnih zahtev, slike nekaj časa zbiral in shranjeval na datotečni sistem. Ko se je uporabnik odločil, da je slik toliko, da bi jih želel dodati v aplikacijo, je pognal proceduro. Ta je pregledala datotečni sistem, ugotovila novosti ter nove slike dodala ostalim akterjem (igralcem, režiserjem, …).

Ta postopek smo sedaj v celoti spremenili, ker sedaj ni potrebno fizično hraniti slik na datotečnem sistemu. Odločili pa smo se, da obdržimo enak postopek shranjevanja. Uporabniku bomo še vedno dali možnost, da se sam odloči, kdaj bi rad dodal nove slike in pognal postopek. Razlika bo v tem, da bomo v novem postopku, le pogledali nove datoteke ter jih dodali ustreznemu akterju v aplikaciji. Nato bomo sliko pobrisali in tako zmanjšali nabor slik na datotečnem sistemu. Ob končanem postopku vnosa bo datotečni sistem prazen.

4.2.4. E-pošta

Za lažje komuniciranje z ostalimi mediji smo se odločili, da prenovo programa dopolnimo z možnostjo pošiljanja elektronskih sporočil, katerim lahko pripnemo material o igrah. Za čim lažjo uporabo smo napisali funkcijo, ki avtomatsko prebere elektronske naslove iz MS Outlook računa in jih shrani v aplikacijo. To opravi procedura ***poisciEmailNaslove*** *(Slika 17).*

```
Sub poisciEmailNalove()
On Error GoTo ErrorHandler

      MousePointer = 11

      'Pridobimo število kontaktov
      Call GetOutlookContactsCount

     ReDim eMail(StEmailNaslovov)

     'Procedura prebere kontakte
     Call GetOutlookContacts

     'prepisem jih v tabelo t_email
     sql = "delete from t_eMail where status = 1 and id_uporabnik =" & gsUserID
     gcnRadio.Execute sql, dbExecDirect

     For j = 1 To StEmailNaslovov
          sql = "insert into t_email(ime,email,id_uporabnik,status) values ('" & eMail(j).Ime & "','" &
                eMail(j).eMail & "'," & gsUserID & ",1)"
          gcnRadio.Execute sql, dbExecDirect
     Next j

     MousePointer = 1

Exit Sub
ErrorHandler:
    MousePointer = 1
    Call ErrorHandler
End Sub
```

Slika 17: Procedura poisciEmailNaslove

```
Sub GetOutlookContacts()
  Dim ol             As Object   ' ms-outlook
  Dim olns           As Object   ' ms-outlook
  Dim objFolder      As Object   ' ms-outlook
  Dim objAllContacts  As Object   ' ms-outlook
  Dim Contact        As Object   ' our own contact-object

  'V primeru napake pri uporabi  <Outlook.Application>  ulovimo napako
   On Error GoTo errorh

  Set ol = CreateObject("Outlook.Application")
  Set olns = ol.GetNamespace("MAPI")
  Set objFolder = olns.GetDefaultFolder(10)
  Set objAllContacts = objFolder.Items

  j = 1

  'Preberemo kontakte in jih shranimo v arrey
  For Each Contact In objAllContacts
    If Len(Contact.Email1Address) > 0 Then
       eMail(j).Ime = Contact.FullName
       eMail(j).eMail = Contact.Email1Address
       j = j + 1
    End If
  Next
```

```
ex:
  'Sprostimo objekte
  Set Contact = Nothing
  Set objAllContacts = Nothing
  Set objFolder = Nothing
  Set olns = Nothing
  Set ol = Nothing
Exit Sub

errorh:
  MsgBox "Napaka pri inicijalizaciji Outlooka.", vbCritical, "Opozorilo"
  Resume ex
End Sub
```

Slika 18: Procedura GetOutlookContacts

Slika 18 prikazuje proceduro ***GetOutlookContacts,*** ki prepiše kontakte shranjene v MS Outlook-u v tabelo, ki jo nato uporabimo za prepis v podatkovno bazo.

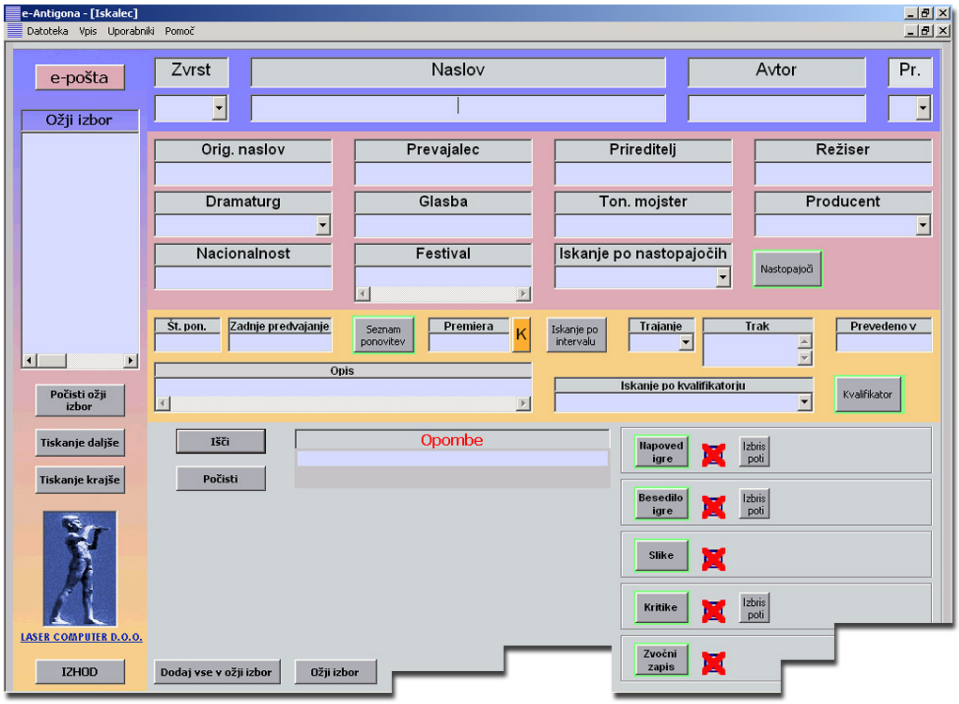
4.2.5. Zvočni zapis v MP3 formatu

Ob prenovi smo se odločili tudi za dopolnitev aplikacije z možnostjo poslušanja zvočnih zapisov v MP3 formatu. V podatkovno bazo smo dodali table za shranjevanje zvočnih zapisov v BLOB formatu ter povezavo do iger. Za razvoj modula namenjenega poslušanju iger smo uporabili knjižnico ***ActiveMovie control type library***.

5. Opis aplikacije

Pri zasnovi osnovnega okna aplikacije smo se z uporabniki dogovorili za preprosto in pregledno okno. Okno je zasnovano v barvnih shemah, ki zaokrožujejo določene segmente igre in s tem kar najlepše prikažejo posamezne vrste podatkov o igri (slika 19). Osnovni namen maske je prikaz in iskanje. Podatke o igrah je možno iskati na več načinov:

- s preprostim iskanjem,
- z iskanjem po časovnem intervalu,
- iskanjem s pomočjo spustnih seznamov (po zvrsti, programu, dramaturgu, producentu, trajanju, nastopajočih in kvalifikatorju) ter
- s kombiniranim iskanjem.



Slika 19: Osnovna maska aplikacije

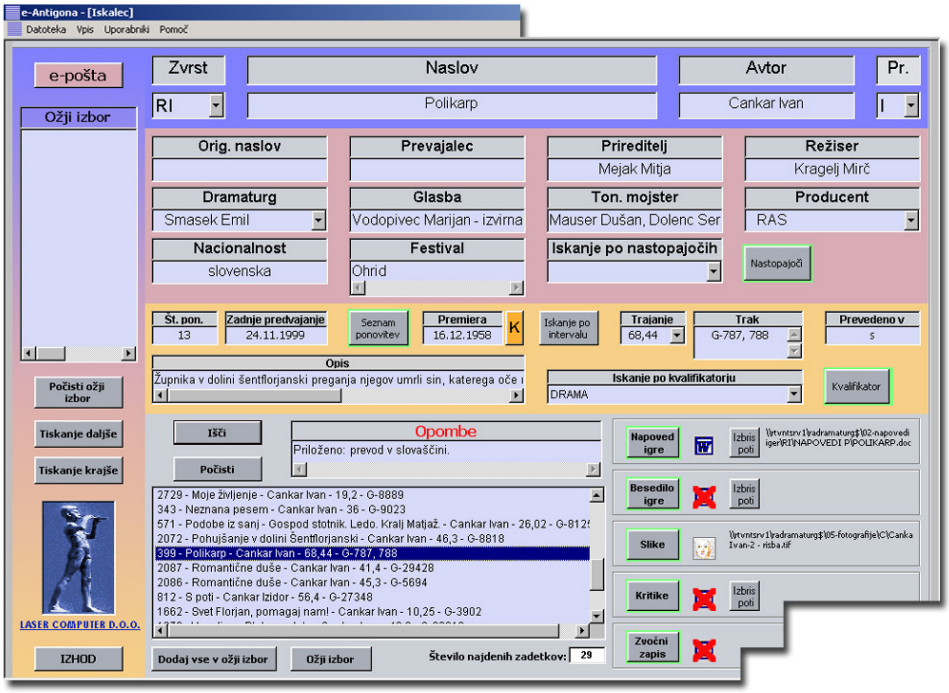
5.1. Iskanja

5.1.1. Preprosto iskanje

Pri **preprostem iskanju** se v katerokoli polje osnovne maske vpiše iskalni kriterij ali le njegov del, po katerem aplikacija išče po podatkovni bazi.

Primer: v polje ***Režiser*** se vpiše celo (na primer: "Kragelj") ali le del imena režiserja (na primer: "krag" ali "agel") in s klikom na gumb ***Išči***, program poišče vse igre izbranega režiserja (oziroma vse tiste, ki ustrezajo vpisanemu nizu) in jih izpiše v spodnjem delu okna v seznam (slika 20). Izpiše se tudi število iger, ki jih je program našel. Ob kliku na posamezno igro v izpisanem seznamu se prikaže kartica te igre oziroma vsi vpisani podatki te igre v osnovnem oknu.

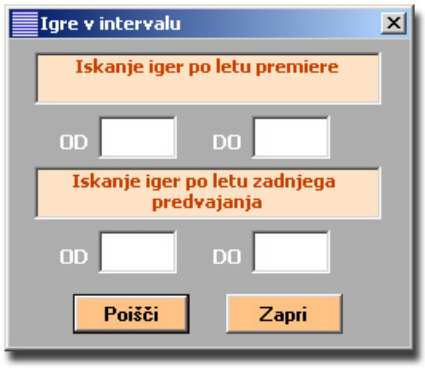
Slika 20 prikazuje izgled osnovne maske po vpisanem iskalnem parametru v trenutku, ko je aplikacija vrnila rezultate. Vidimo lahko, da nam je aplikacija vrnila 29 rezultatov, ki so ustrezali iskalnemu parametru, in nam jih v seznamu prikazala.



Slika 20: Najdene igre glede na iskalni parameter

5.1.2. Iskanje po časovnem intervalu

Iskanje po časovnem intervalu pomeni iskanje iger v okviru let, ki se jih vpiše v ustrezna polja. V osnovni maski se s pritiskom na gumb ***Iskanje po intervalu*** odpre manjša maska ***Igre v intervalu*** (slika 21).



Slika 21: Maska za iskanje v časovnem intervalu

Program poišče vse igre, ki so bile bodisi **premierno** (zgornji del maske) bodisi **zadnjikrat** (spodnji del maske) predvajane v izbranem časovnem intervalu. Išče se lahko po obeh

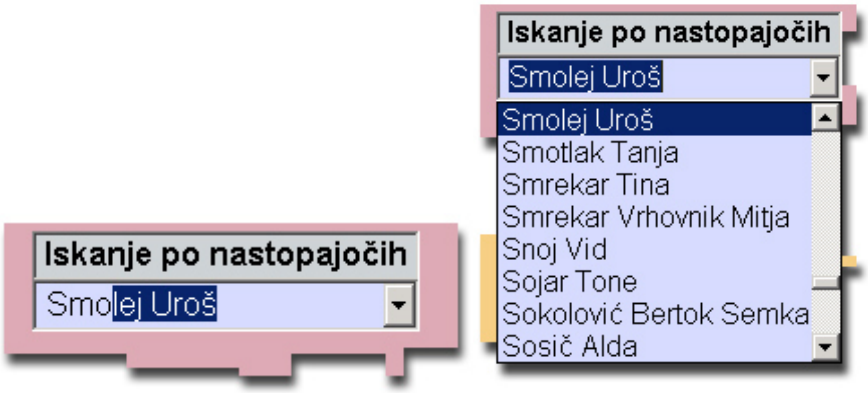
kriterijih hkrati: ob vpisu obeh intervalov (lahko sta različna, vendar morata biti smiselna) poiščemo igre, ki so bile v izbranih obdobjih premierno in zadnjikrat predvajane.

Igre je mogoče iskati tudi v kombinaciji z drugimi iskalnimi kriteriji v poljih osnovnega okna, ki se jih je treba vpisati prej.

5.1.3. Iskanje po nastopajočih

V polju ***Iskanje po nastopajočih*** imamo spustni seznam, v katerega podatkov ne vpisujemo, temveč jih izberemo (slika 22, desna): s klikom na puščico ob polju se odpre seznam, v njem pa si s klikom izberemo nastopajočega. Z gumbom ***Išči*** sprožimo iskanje iger, v katerih izbrani nastopa.

Nastopajočega lahko izberemo tudi tako, da v polje ***Iskanje po nastopajočih*** vpišemo iskani priimek in ime (ali pa le nekaj začetnih črk) in program sam ponudi ime, ki je vpisanim črkam najbliže (slika 22, leva). S klikom na puščico ob polju se odpre spustni seznam z ostalimi najbližjimi imeni.



Slika 22: Prikaz dela s spustnimi seznami

Na enak način (s spustnim seznamom), iščemo tudi v polju ***Iskanje po kvalifikatorju***; v polju **Producent** lahko ponujeni vpis, ki smo ga izbrali v spustnem seznamu, dopolnimo z radijsko postajo in državo, če poznamo njun točni naziv (recimo: ponujeni in izbrani **radio -** dopolnite z: **YLE, Finska**); pri polju ***Dramaturg*** pa lahko izberemo imena v spustnem seznamu ali pa vpišemo katerokoli ime ali del imena, za katerega predvidevamo, da bi utegnil biti dramaturg.

Pri teh načinih iskanja lahko vpišemo več iskalnih kriterijev, kar je za hitrejši in natančnejši rezultat tudi priporočljivo.

5.1.4. Kombinirano iskanje

Rezultat preprostega načina iskanja - po zgolj enem kriteriju, pri katerem izpolnimo zgolj eno polje v osnovni maski - je največkrat veliko število zadetkov. Zato je bolje, če iščemo s kombiniranim načinom. **Kombinirano iskanje** je iskanje **z vpisom več iskalnih kriterijev hkrati** v polja osnovne maske.

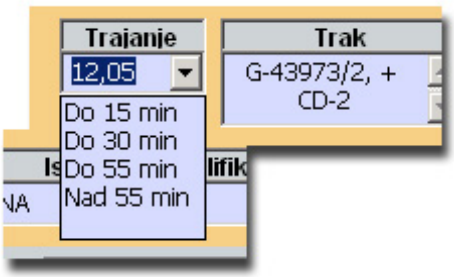
Primer: prvi kriterij vpišemo v polje ***Avtor*** (na primer: "Puntar"), drugi kriterij v polje ***Režiser*** (na primer: "Sajko"), iz spustnega seznama ***Dramaturg*** pa izberemo dramaturga (na primer: Flere) in pritisnemo gumb ***Išči***. Program upošteva vse tri iskalne kriterije hkrati. Izbral bo igre Franeta Puntarja, ki jih je režirala Rosanda Sajko in katerih je bila dramaturginja Djurdja Flere.

V katerokoli polje osnovne maske lahko vpišemo tudi samo del iskalnega kriterija; največkrat je to priročno, kadar ne vemo natančnega naslova ali pa imena. V mislih imamo, radijsko igro o duhovitih zajcih, vendar ne vemo natančno, ali je bil naslov "Zborovanje zajcev na goljavi" ali "Tekmovanje zajcev na dobravi" ali celo kako drugače. V tem primeru vtipkamo v polje ***Naslov*** le "zajc" in pritisnemo gumb ***Išči***. Program nam bo - v konkretnem primeru baze slovenskih radijskih iger - našel štiri naslove iger, ki vsebujejo niz "zajc". Med njimi brez težav prepoznamo pravilen naslov "Pevsko tekmovanje zajcev na goljavi".

Kombinacije iskalnih kriterijev so poljubne.

5.1.5. Iskanje po trajanju

Ker so predvajalni časi v radijskih programih običajno omejeni do določene minutaže, je iskanju iger s primernim trajanjem, namenjen spustni seznam (slika 23). Na ta način lahko poiščemo igre, ki trajajo do 15, 30, 55 minut ali nad 55 minutami. Iskalni niz lahko seveda kombiniramo tudi s poljubnimi drugimi kriteriji.



Slika 23: Prikaz vpisanih podatkov v spustnem seznamu Trajanje

V nadaljevanju se bomo osredotočili predvsem na dele aplikacij, ki so bili pregledani v sklopu ponovnega inženiringa . To je delo s slikami, besedili, napovedniki in kritikami. Prav tako pa bomo izpostavil še varnost in omejitve dostopanja do podatkov.

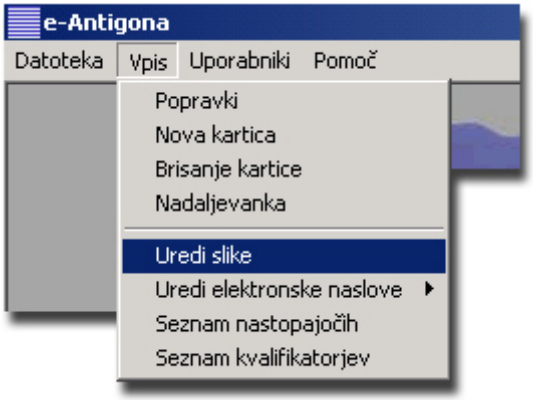
5.2. Slike

Slike so tisto področje aplikacije, ki ga želimo izpostaviti. Slik v prvotni verziji ni bilo v načrtu. Ob pričetku razvoja aplikacije pa se je pokazalo, da bi slike zelo dvignile uporabnost v smislu preglednosti in multipredstavnosti. Poleg vizualne predstavitve sodelujočih pa se uporabljajo tudi kot priponke za komunikacijo z mediji (pošiljanje v časopisne hiše se napovednik sporeda opremi s sliko).

5.2.1. Implementacija v aplikaciji

Pripenjanje slik h karticam izvede program samodejno. V meniju ***Vpis*** je podmeni ***Uredi slike***, ki vsa imena, vpisana pri posameznih igrah v seznamu ***Nastopajoči***, poveže z njihovimi slikami (slika 24), če:

- obstaja mapa za slike z imenom »05-fotografije«,
- je naslov slike povsem enak imenu osebe v seznamu nastopajočih (npr. »Emeršič Bojan«) in
- je slika v formatu `jpg`, `tif`, `gif` ali `bmp`.



Slika 24: Ukaz »Uredi slike«

Z ikono  se identificirajo prikazi iger, ki imajo v bazi nastopajočih kakšno sliko.

Ikona nam pove, da so na igro pripete slike. Ker pa je slik lahko več, je potrebno za prikaz vseh klikniti na gumb ***Slike***. S tem se odpre maska ***Slike*** s seznamom povezanih slik (slika 25). V seznamu se nahajajo vsi nastopajoči, ki imajo slike. Za enega nastopajočega imamo lahko več slik, zato se imena lahko podvajajo.

Slika 25: Maska za pregled slik

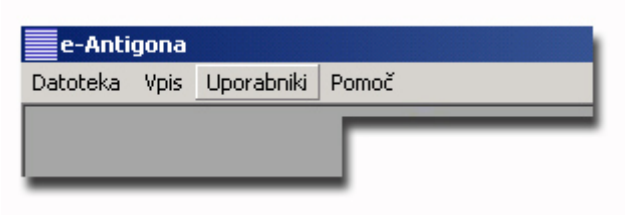
Slike se odpirajo z dvojnim hitrim klikom na posamezno vrstico v maski ***Slike***.
Slika se odpre v novi maski ***Prikaz slik***, v katerem jo je mogoče z desnim ali levim klikom povečati oziroma pomanjšati.

Povezavo slike s kartico je mogoče odstraniti. Temu služita dva gumba v maski ***Slike***:

- zgornji gumb odstrani povezavo s sliko pri izbrani kartici (če, na primer, ugotovite, da igralec ne nastopa v igri),
- spodnji gumb odstrani povezavo s sliko pri vseh karticah, kjer je ta slika pripeta,
- slike se odpirajo z dvojnim hitrim klikom na vrstico v maski ***Slike***,
- slika se odpre v novi maski ***Prikaz slik***, v katerem jo je mogoče z desnim ali levim klikom povečati oziroma pomanjšati.

5.3. Varnost

Osnovni namen programa je olajšati pregled arhiva radijskih iger, ki jih ima v hrambi Radio Slovenija. Aplikacija je namenjena vpogledu in urejanju. Pravice uporabe aplikacije imajo različne vrste uporabnikov. Zato je bilo potrebno narediti določene varnostne ukrepe. Ob kreiranju novega računa tako administrator določi uporabniško ime in geslo ter določi uporabnikove pravice. To omogoča meni ***Uporabniki*** v osnovnem oknu (slika 26).



Slika 26: Meni uporabniki

Klik na meni ***Uporabniki*** odpre maska za vpisovanje podatkov novega uporabnika (slika 27).

Slika 27: Maska za dodajanje uporabnikov in njihovih pravic

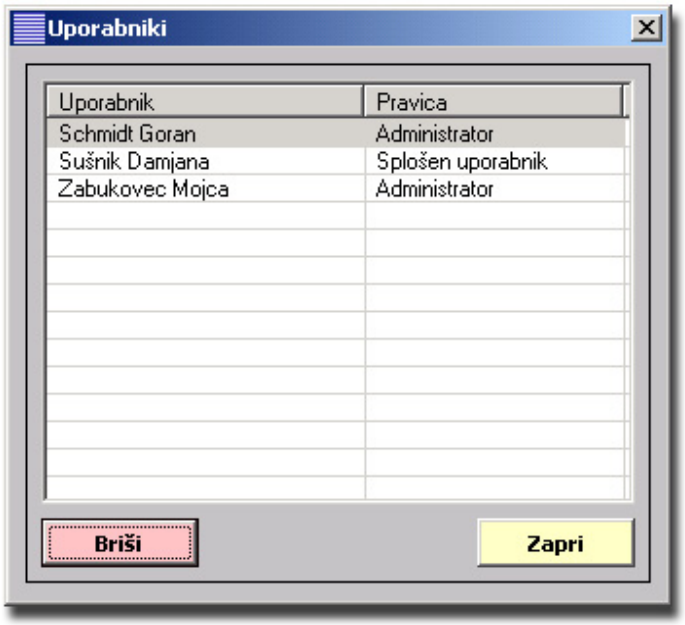
Z vpisom uporabnika se omogoči, da se ta program sploh lahko odpre. Dodeli se mu uporabniško ime (na primer: dramaturg) in geslo (na primer: zaplet), z izbiro pravice v polju ***Določitev pravic uporabnika*** pa se mu določi obseg uporabe programa.

V spustnem seznamu polja ***Določitev pravic uporabnika*** so na voljo tri pravice:

- **administrator** - omogoča uporabo vseh funkcij programa,
- **dramaturg** - omogoča iskanje iger in pregledovanje, krajše in daljše tiskanje ožjega izbora ter spreminjanje napovedi in besedila igre,
- **splošen uporabnik** -ima največ omejitev in ne dopušča nikakršnih sprememb podatkov. Omogoča iskanje iger in zgolj branje, gledanje in poslušanje pripetih dokumentov, slik in zvočnih zapisov, onemogočeno pa je kopiranje in tiskanje dokumentov.

Vpise novega uporabnika potrdite s klikom gumba ***Dodaj***, gumb ***Počisti*** pa sprazni vsa vpisna polja in omogoča nov vpis.

Gumb ***Uporabniki*** odpre seznam vseh vpisanih uporabnikov programa in njihovih pravic (slika 28).



Slika 28: Seznam uporabnikov aplikacije

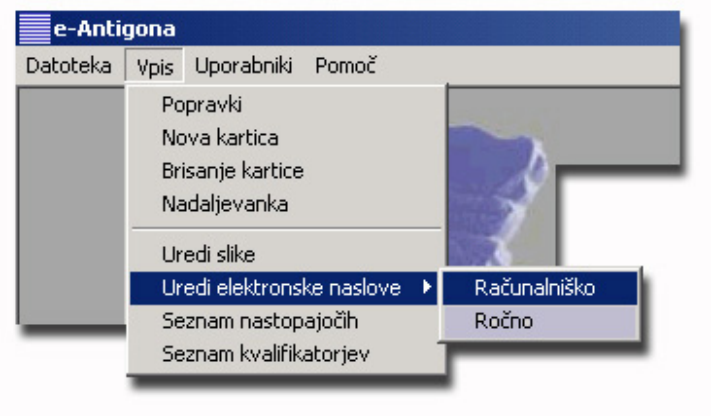
Maska omogoča le brisanje uporabnika, in sicer najprej s klikom na vrstico in potem klikom gumba ***Briši***. Brisani uporabnik se v program ne bo mogel več prijaviti.

5.4. Dodane funkcionalnosti aplikacije

5.4.1. E-pošta

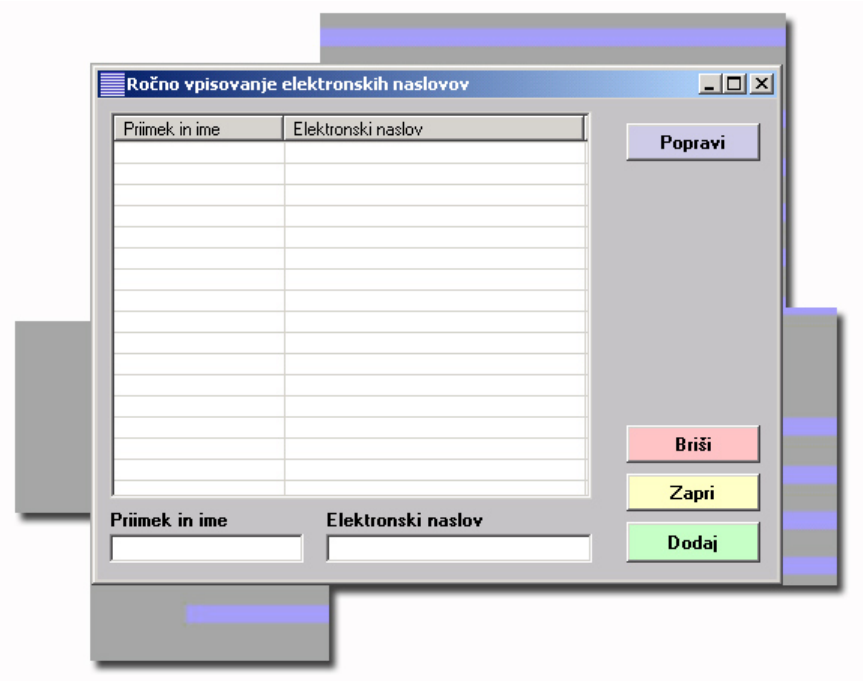
Eden od namenov programa je tudi posredovanje izbranega gradiva preko elektronske pošte različnim naslovnikom.

Aplikacija je narejena tako, da je možno kontakte (*ang. Contacts*) prevzeti strojno iz lokalno nameščenega Microsoft Outlook-a. To naredimo tako, da kliknemo v meniju ***Vpis*** podmeni ***Uredi elektronske naslove*** in ukaz ***Računalniško*** (slika 29). Naslovniki, ki so shranjeni v predalu kontakti uporabnika programa Microsoft Outlook, bodo prepisani v razdelek ***Kontakti*** v maski ***e-pošta*** program e-Antigona, kjer je mogoče njihovo izbiranje.



Slika 29: Prikaz menija za urejanje naslovov elektronske pošte

V kontakte aplikacije lahko naslove vpisujemo tudi ročno. Ti vpisi so neodvisni od postopka ***Uredi elektronske naslove*** in se torej ne brišejo ali kakorkoli spreminjajo, kadar računalniško urejamo elektronske naslove. Za ročno vpisovanje naslovov izberemo iz podmenija ***Uredi elektronske naslove*** ukaz ***Ročno*** (slika 29). V tem primeru se odpre maska ***Ročno vpisovanje elektronskih naslovov*** (slika 30).

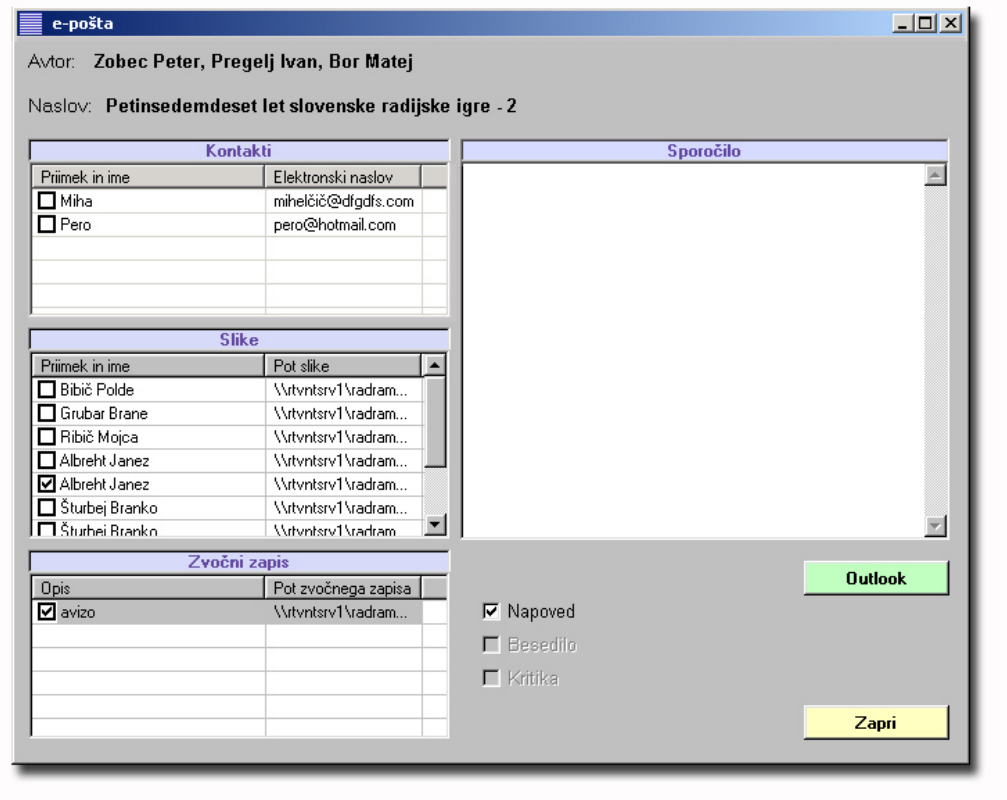


Slika 30: Maska za ročni vnos elektronskih naslovov

Slika 30 prikazuje masko za ročno dodajanje elektronskih kontaktov. Na maski se nahajajo osnovni podatki za prejemnike (ime in priimek (organizacija) ter elektronski naslov).

Za pošiljanje izbranega gradiva z elektronsko pošto najprej izberemo igro. V njeni kartici pritisnemo gumb ***e-pošta***, ki je v zgornjem levem kotu osnovnega okna.

Odpre se maska ***e-pošta*** z izpisanim imenom avtorja in naslovom igre. V tej maski lahko napišemo sporočilo in pripnemo gradivo, ki ga želimo poslati izbranim naslovnikom (slika 31).



Slika 31: Maska za pripravo elektronskega sporočila

V razdelku ***Kontakti*** določimo naslovnike s pritiskom na potrditveno polje *(angl. check box)* pred imenom. Naslovnike lahko tudi urejamo po abecednem redu priimkov ali po abecedi elektronskih naslovov. To storimo s pritiskom na ime stolpca ***Priimek in ime*** ali ***Elektronski naslov.*** Vrstni red se spreminja po abecedi naraščajoče (od A do Ž) oz. padajoče (od Ž do A). Če v ***Kontaktih*** ni želenega naslova, ga lahko vnesemo ročno po pritisku na gumb ***Outlook***.

V razdelku ***Slike*** in ***Zvočni zapisi*** s pritiskom na potrditveno polje določimo slikovno in zvočno gradivo, ki ga želimo pripeti k elektronskemu pismu.

MS Word dokumente pripnemo k elektronskemu pismu s pritiskom na potrditveno polje ob napisih ***Napoved***, ***Besedilo*** in ***Kritika*** v sredini maske spodaj. Črno so izpisani igri pripeti dokumenti, sivo pa tisti, ki niso pripeti ali so nedosegljivi.

Telo (*ang. Body*) sporočila napišemo v razdelek ***Sporočilo***.

Ko je sporočilo napisano in po želji opremljeno z vsemi priponkami, pritisnemo gumb ***Outlook***, ki prenese naslovnike, besedilo in priponke izbrane igre v program Microsoft Outlook, iz katerega elektronsko pošto po običajni poti pošljemo.

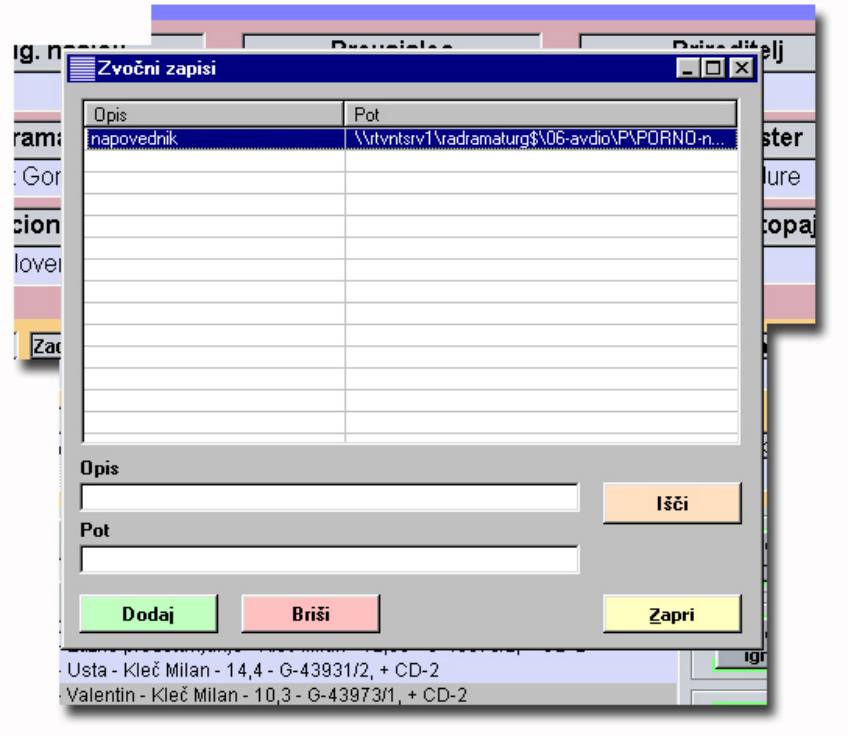
Prednost rešitve je hitrejše pošiljanje materiala o igrah naslovniku. V primerjavi s prvotno verzijo aplikacije nam je tako prikrajšano iskanje potrebnih datotek za pošiljanje.

Ker po prenovi do datotek nimamo dostopa, je to edini način, za prenos podatkov o igrah iz računalnika.

5.4.2. Predvajanje MP3 zapisa iger

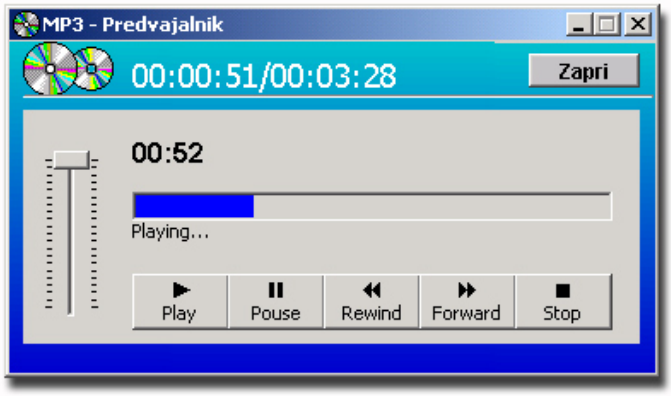
Zvočne zapise pripenjamo h kartici z gumbom ***Zvočni zapis***.

Gumb odpre masko ***Zvočni zapisi*** (slika 32), v kateri s klikom na gumb ***Išči*** poiščemo preko ***Raziskovalca*** posnetek, shranjen na računalniku. Kliknemo naslov posnetka in gumb ***Potrdi***. V polje ***Opis*** vpišemo opis posnetka (na primer "avizo", "napovednik", "igra" ...) in kliknemo gumb ***Dodaj.*** Program zvočni zapis prenese v seznam posnetkov na kartici ***Zvočni zapisi***, s tem pa je že tudi narejena povezava.



Slika 32: Vpenjanje zvočnih zapisov

S klikom na ikono  odpremo prvi povezani zvočni zapis. Če je zvočnih zapisov več, jih odpiramo s hitrim dvoklikom vrstice v maski ***Zvočni zapisi*** (slika 32). Ob pritisku na ikono ali ob hitrem dvokliku na omenjeno vrstico se odpre maska za predvajanje posnetkov v MP3 formatu (slika 33).



Slika 33: Maska za predvajanje zvočnih zapisov

6. Sklepne ugotovitve

Pri pisanju diplomske naloge smo želeli izpostaviti razvoj aplikacije ter načine in razloge za njeno predelavo. S programsko rešitvijo smo v prvi verziji zastarel način vodenja arhiva radijskih iger izboljšali z uvedbo računalniško vodenega sistema. S tem smo omogočili preglednejši vpogled v obstoječe igre in z njimi povezane podatke. S tem smo tudi drastično skrajšali čas iskanja in pripravo najrazličnejših izpisov in statistik. Aplikacija je uvedla tudi nekaj izboljšav pri multimedijski predstavitvi. Vsaki igri smo dodali pripadajoče slikovno in zvočno gradivo, ki je dostopno v programu. S tem smo oplemenitili arhiv in omogočili večjo preglednost.

V nadaljevanju pa smo se soočili s pomanjkljivostmi prvotne verzije programske rešitve. Predvsem se je izkazalo, da moramo večjo pozornost nameniti varnosti podatkov. Za ozko grlo varnosti se je pokazal predvsem datotečni sistem, v katerem smo hranili vse podatke, ki jih je aplikacija uporabljala. Datotečnega sistema ni bilo možno ustrezno nadzorovati v aplikaciji, saj je bil dostopen vsakomur, ki je imel dostop do sistema. Zaradi tega smo izvedli postopek prenove programske rešitve, v katerem smo se odločili za zamenjavo podatkovne baze. To nam je omogočilo prenos datotek iz datotečnega sistema v samo podatkovno bazo. S tem in nekaterimi popravki aplikacije smo dosegli popoln nadzor nad vsemi podatki ter oskrbeli za nemoteno in suvereno delovanje aplikacije.

Program je velika izboljšava pri vodenju arhiva radijskih iger. Z njim smo omogočili elegantnejše, predvsem pa hitrejše in natančnejše pregledovanje. Programsko rešitev v prvotni verziji uporabljajo na Radiu Slovenija od leta 2001 in so z izdelkom zelo zadovoljni. Program jim olajšuje pripravo radijskega programa na 1, 2 in 3 programu, saj vsebuje tudi podatke o predvajanosti. V arhivu radijskih iger je zabeleženo preko 3500 iger. Izboru primernih radijskih iger za program pa je pripomogla tudi multimedijska nadgraditev arhiva, saj za poslušanje in izbor igre ni več potrebno izpolnjevati obrazcev za pridobitev zvočnih trakov iz skladišča. Z nadgradnjo aplikacije pa se je izboljšala tudi komunikacija z ostalimi medijskimi hišami. Želene podatke je sedaj veliko lažje zbrati ter jih kasneje s pomočjo vgrajene možnosti pošiljanja elektronskih sporočil s priponkami poslati naslovnikom.

Uvedba informacijske rešitve je odprla veliko novih možnosti in razmišljanj o izboljšavah sistema. V prihodnosti bi bilo mogoče razviti spletno različico aplikacije, s katero bi bilo možno do podatkov dostopati tudi od drugod. Podatki so namreč zanimivi tudi za tuje medijske hiše; predvsem se zanimajo za igre, v katerih so sodelovali kakšni njihovi pisatelji ali igralci. V tem duhu bi bilo možno razširiti aplikacijo, da bi vanjo vpisovali tudi radijski material drugih medijskih hiš in s tem vzpostavili poenoteno podatkovno bazo radijskih iger tudi iz drugih držav Evrope.

Literatura

[1] Access to Oracle Migration. Dostopno na: http://jehiah.cz/archive/access-to-oracle-migration

[2] Chikofsky, E. and Cross, J., 1990. Reverse Engineering and Design Recovery: A Taxonomy. Dostopno na: http://seal.ifi.uzh.ch/fileadmin/User_Filemount/Vorlesungs_Folien/Evolution/SS05/chikofsky90.pdf

[3] Microsoft Access. Dostopno na: http://en.wikipedia.org/wiki/MS_Access

[4] Microsoft Access Focus Area. Dostopno na: http://www.oracle.com/technology/tech/migration/focusareas/access.html

[5] Microsoft Visual Basic. Dostopno na: http://en.wikipedia.org/wiki/Visual_basic_6

[6] Oracle Database. Dostopno na: http://sl.wikipedia.org/wiki/Oracle

[7] Reengineering (software). Dostopno na: http://en.wikipedia.org/wiki/Reengineering_(software)

[8] Software Re-engineering. Dostopno na: http://satc.gsfc.nasa.gov/support/reengrpt.PDF

[9] Upora Visual Basica. Dostopno na: http://sawaal.ibibo.com/computers-and-technology/what-vb-what-used-479806.html