UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Jošt Pristavec

Razvoj bonitetnega informacijskega sistema

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2011

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: 01756/2011
Datum: 04.04.2011

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

Kandidat: **JOŠT PRISTAVEC**

Naslov: **RAZVOJ BONITETNEGA INFORMACIJSKEGA SISTEMA**
THE DEVELOPMENT OF CREDIT REPORT INFORMATION SYSTEM

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Zasnujte in razvijte preprost bonitetni informacijski sistem za manjše finančno podjetje. Glavno pozornost usmerite v izdelavo bonitetne ocene posameznega podjetja. Od ostalih ključnih funkcionalnosti naj informacijski sistem podpira izdelavo in pregled bilanc, izdajo naročil in zaznamkov.

Mentor:

doc. dr. Rok Rupnik

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

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

diplomskega dela

Spodaj podpisani Jošt Pristavec,

z vpisno številko 63020126,

sem avtor diplomskega dela z naslovom:

Razvoj bonitetnega informacijskega sistema.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Roka Rupnika
- 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 7.7.2011 Podpis avtorja:

ZAHVALA

Na tem mestu se zahvaljujem mentorju doc. dr. Roku Rupniku za pomoč, nasvete in predloge pri nastajanju diplomskega dela.

Zahvaljujem se moji ženi Ani *ter staršem, ki so me tekom moje študijske poti podpirali ter mi nudili moralno podporo. Iskrena hvala.*

Kazalo

Seznam uporabljenih kratic in simbolov

IIS - Internet Information Services
ASP - active server pages
DLL - Dynamic-link library
AJPES - Agencija Republike Slovenije za javnopravne evidence in storitve
ADO - active data objects
Ajax - Asynchronous JavaScript and XML
BLOB - binary large object
AOP - avtomatsko obdelani podatek

Povzetek

Bonitetni informacijski sistem je enovita, kompleksna spletna aplikacija, katere namen je delo s podatki o slovenskih podjetjih. Bonitetni sistem je namenjen analitiku za pregledovanje in urejanje tako splošnih podatkov o podjetjih, kot so denimo naziv, naslov, davčna številka, zastopniki itd. kakor tudi finančnih podatkov, bilanc, bonitetnih poročil ipd. Glavna funkcionalnost aplikacije je možnost kreiranja bonitetne ocene za posamezno podjetje ter možnost izvoza le te v obliki poročila.

V diplomskem delu, ki je pred vami, se ukvarjam z razvojem omenjenega bonitetnega informacijskega sistema. Razvoj obsega zajem zahtev, izgradnjo podatkovnega modela, postavitev primerne arhitekture ter na koncu kodiranje.

Predstavljene so tehnologije ter posamezna orodja katera sem uporabil pri razvoju aplikacije. Omenjeni so tudi argumenti, zakaj sem se pri razvoju odločil za posamezno tehnologijo.

Največji del, po obsegu, predstavlja četrto poglavje, ki je nekakšno jedro diplomskega dela. V njem so podrobno, tako grafično kot opisno, predstavljeni posamezni moduli in funkcionalnost celotne aplikacije. Predstavljena je poslovna logika posameznega modula, povezava s podatkovnim delom, kakor tudi grafični vmesnik za interakcijo z uporabnikom, posameznega modula. Opisana so tudi navodila za uporabo posameznega modula aplikacije.

V zaključku so navedene morebitne možnosti nadgradnje sistema. Izpostavljene so prednosti in slabosti, na katere sem naletel ob uporabi posamezne tehnologije.

Ključne besede: bonitetni sistem, poslovni subjekt, finančni podatek, poročilo, .NET, Telerik, LDAP

Abstract

Credit Report Information System is a unified, complex web application designed to work with the data of Slovenian companies. A credit system is for the analyst to view and edit either general information about companies, such as name, address, tax identification number, agents, etc. or financial data, balances, credit reports, etc. The main functionality of applications is the possibility of creating a credit rating for each company and the possibility to export a credit rating as a report.

In the thesis, I deal with the development of the credit information system. The development of the credit system includes capture requirements, building the data model, building appropriate architecture and at end programming.

Presented are technologies and basic tools which I used during developing applications. Noted are also arguments pro and against particular technology.

The largest part, by volume, represents the fourth section, which is sort of the thesis' core. There are detailed presentations, graphical and narrative, of the individual modules and of the entire application.

For each module are presented the business logic, the link with a data part and also a graphical interface which functionality is to interact with users. The fourth section includes also instructions for each module of application.

In the conclusion are listed possibilities of upgrading the system. I pointed out the advantages and disadvantages, which I encountered during using particular technology.

Key words: credit system, business entity, a financial information, report, .NET, Telerik, LDAP

1 Uvod

Na željo naročnika, se je pojavila potreba po izdelavi naprednega informacijskega sistema. Informacijski sistem je namenjen vsem zaposlenim v oddelku za bonitete. V celoti naj bi nadomestil orodja, katera so bila v uporabi do sedaj, med katerimi so marsikatera že zelo stara, ali pa je njihova funkcionalnost slaba. Prednost enovitega informacijskega sistema je, da ima uporabnik na enem mestu dostop do več različnih orodij n pripomočkov v obliki modulov. Zahteva je, da naj bo informacijski sistem realiziran kot spletna stran in naj teče na enem strežniku s podatkovno bazo na drugem. Podatkovna baza ima centralizirano strukturo podatkov, s tem se izognemo redundanci podatkov. Informacijski sistem naj ima možnost povezovanja z Active Directory-em. Funkcionalnosti sistema smo pridobili z zajemom zahtev in dokumentom o specifikaciji bonitetnega informacijskega sistema. Med glavne funkcionalnosti aplikacije sodijo izdelave bilanc, naročil, zaznamkov, bonitetnih poročil, pregledi podatkov o poslovnih subjektih, administracija uporabnikov, prevajanje, pošiljanje elektronske pošte izbranim uporabnikom ter izvoz podatkov v obliki bonitetnega poročila.

Naša naloga je bila izdelati bonitetni informacijski sistem kateri bo zadostil željam naročnika. Informacijski sistem je razdelejen na module, ki izhajajo iz zgoraj omenjenih funkcionalnosti. V diplomski nalogi podrobno predstavljam le tiste module, katerih avtor sem oz. tiste pri razvoju katerih sem aktivno sodeloval. Predstavljene so tehnologije in gradniki, kateri so bili uporabljeni pri izdelavi bonitetnega informacijskega sistema.

2 Opredelitev problema

Zgraditi aplikacijo oz. informacijski sistem, za delo s podatki o slovenskih podjetjih. Informacijski sistem je namenjen analitikom znotraj podjetja. Želja je zgraditi napredno aplikacijo z glavnimi funkcionalnostmi:

- pregled osnovnih podatkov o podjetju
- pregled, vnos in posodabljanje finančnih podatkov in bilanc stanja
- izdelava bonitetnih poročil, na podlagi katerega se izdela bonitetna ocena podjetja ter preračuna najvišja možna višina kredita
- vmesnik za administracijo uporabnikov informacijskega sistema
- vmesnik za oddajo naročil
- sledenje akcijam v obliki zaznamkov, ki se vpisujejo v bazo podatkov
- izpis poljubnih statistik, vezanih bodisi na poslovni subjekt, bodisi na analitika sistema
- administracija podatkovnih baz

Na željo naročnika, morajo grafični vmesniki za delo s sistemom zgledati profesionalno in atraktivno. Sistem mora biti odziven, hiter ter intuitiven za uporabo.

Glede na omenjene želje naročnika, smo se odločili da pri razvoju bonitetnega informacijskega sistema posežemo po naprednih tehnologijah za izgradnjo spletne aplikacije. Verjetno ne bo odveč, če na tem mestu še enkrat pokomentiramo odločitev izgradnje spletne aplikacije namesto namizne aplikacije. Razlog je seveda v tem, da je sistem namenjen širši množici uporabnikov znotraj podjetja. S spletno aplikacijo se ognemo nameščanjem aplikacije na več računalnikov, dostop do konfiguracije sistema omogočimo le administratorju, pri posodabljanju sistema je le ta potreben samo na enemu mestu in še druge razloge bi se našlo. Ker pa smo hoteli, da sistem navidezno daje vtis, kot da teče na lokalnem računalniku, smo se odločili, da ga razvijemo kot aplikacijo ajax.

3 Uporabljena orodja in tehnologije

Bonitetni informacijski sistem smo razvili kot aplikacijo ASP.NET, katera teče na okolju .NET verzija 3.5.
V aplikacijo smo vključili napredne kontrole Telerik ASP.NET AJAX ter Telerik Reporting. Kontrole Telerik ASP.NET AJAX so nam močno olajšale delo z ajaxom saj praktično skrijejo celotno njegovo implementacijo. Poleg tega, so oblikovno dovršene in uporabniku zanimive. Telerik Reporting smo uporabili za izvoz podatkov iz aplikacije v grafično lepo urejeno poročilo. Prednost, ki jih ponuja Telerik Reporting, je v velikem naboru grafičnih elementov in stilov. Na željo uporabnika je tako možna velika prilagodljivost izgleda končnega poročila. Povezava podatkovne baze z aplikacijo poteka s pomočjo DB2 .NET Data Provider-ja. [1]

Za razvoj smo uporabljali Microsoftov Visual Studio 2008 Professional, programski jezik C#, za dostop do baze pa odprtokodno orodje Toad For DB2 Freeware 4.6. [2] Načrtovanje podatkovne baze in kreiranje skript je potekalo z orodjem Power Designer 15. Kot podatkovni strežnik nam je služil IBM DB2 verzije 8.1 [3]

3.1 .NET

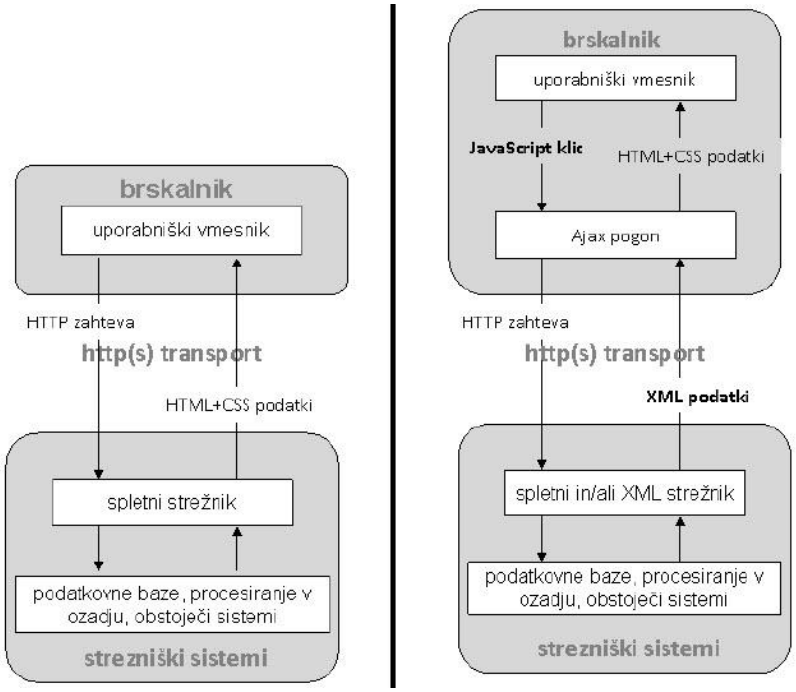
Ime .NET je krovno ime za cel skupek izdelkov in tehnologij, ki jih razvija in trži podjetje Microsoft. Poglavitna skupna točka vseh izdelkov in tehnologij je odvisnost od ogrodja .NET (.NET Framework). Ogrodje je sestavljeno iz velikega števila rešitev za različna programska področja kot so gradnja uporabniških vmesnikov, dostop do podatkov, kriptografija, razvoj spletnih rešitev, numerični algoritmi, omrežne komunikacije... Ogrodje .NET določa arhitekturo razvoja programskih rešitev. Temelji na konceptih objektne tehnologije in komponentnega razvoja. Razvijalcem ponuja pregledno razvojno okolje, s katerim lahko na enostaven način gradijo aplikacije. Programi, ki so napisani za ogrodje .NET, se izvajajo v posebnem okolju, ki upravlja zahteve programa med izvajanjem. To okolje, ki je tudi sestavni del ogrodja .NET, se imenuje izvajalnik kode skupnega jezika (CLR – Common Language Runtime). Osnova je skupno izvajalno okolje, za različne programske jezike. Okolje .NET dovoljuje, da lahko razvijelac uporablja sočasno celo več različnih jezikov. To pomeni, da programi napisani v enem jeziku lahko uporabljajo metode, podprograme napisane v drugem programskem jeziku.
Koda programov se ne prevaja neposredno v strojno kodo, temveč se prevede v vmesno kodo oz. kodo osnovano na vmesnem jeziku.
Ogrodje .NET vsebuje mnoge razrede, ki ponujajo bogat nabor funkcionalnosti. Razredi v ogrodju sestavljajo knjižnice, katere so organizirane po imenskih prostorih (namespaces), ki omogočajo boljšo organiziranost razredov. [4,5]

3.2 ASP.NET

ASP.NET (Active Server Pages) je Microsoftova tehnologija, ki omogoča ustvarjanje dinamičnih spletnih strani. Dinamične spletne strani se od navadnih spletnih strani razlikujejo po tem, da vsebujejo tudi elemente, ki se dinamično spreminjajo med samim pregledovanjem spletne strani. Dinamični elementi se kreirajo na samem strežniku in so naknadno prevedeni v kodo HTML.
Ko odjemalec (običajno je to spletni brskalnik) želi dostopati do določene spletne strani, napisane v ASP.NET, se zahteva pošlje strežniku. Strežnik prepozna končnico .aspx in stran pošlje posebnemu prevajalniku. Ko prevajalnik pregleduje to stran, razlikuje med navadnimi značkami HTML in strežniškimi gradniki. Če naleti na navadno kodo HTML, jo pusti takšno, kot je, če pa naleti na strežniški gradnik, pokliče metodo Render. Ta pripada vsakemu strežniškemu gradniku. Metoda vrne ustrezno kodo HTML, kjer strežniški gradnik, ki ga je prej opisoval .aspx, opisan v jeziku HTML. Ko prevajalnik pregleda vse značke HTML, je rezultat klasična stran v jeziku HTML. Strežnik to novo ustvarjeno stran pošlje nazaj odjemalcu, ki spletno stran prikaže v brskalniku. [6,7,8]

3.3 Ajax

Ajax (asinhroni JavaScript in XML) je skupina medsebojno povezanih spletnih razvojnih tehnik, uporabljenih za ustvarjanje interaktivnih spletnih aplikacij. Z ajaxom si lahko spletne aplikacije izmenjujejo podatke s strežnikom asinhrono v ozadju, brez potrebe po ponovnem nalaganju strani.
S tem je mogoče tekoče in hitrejše spremljanje ter spreminjanje vsebine na spletni strani. Podatki se prenašajo s pomočjo objektov XMLHttpRequest ali s pomočjo Remote Scriptinga (v starejših brskalnikih, ki ne podpirajo tehnologije Ajax) [9,10]



Slika 1: primerjava tradicionalnega modela spletnih aplikacij(levo) ter modela Ajax (desno)

3.4 Kontrole Telerik

3.4.1 Telerik ASP.NET AJAX

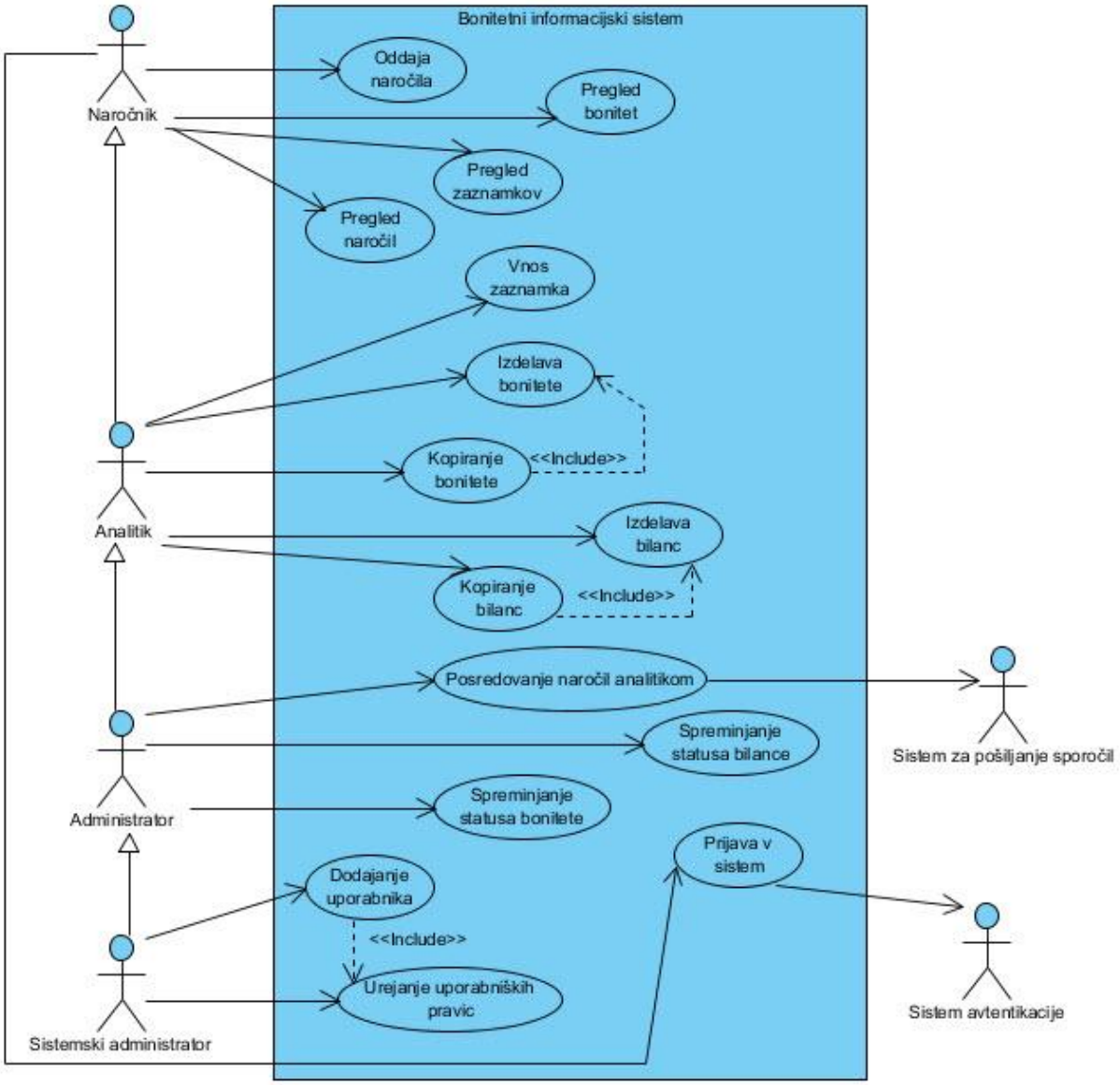
Telerik ASP.NET AJAX ali 'Telerik RadControls for ASP.NET AJAX', kot navaja Telerik svoj produkt, je paket, ki vsebuje preko 70 raznolikih kontrol. Kontrole, katere krasi dokazana zanesljivost, so v pomoč pri izgradnji visoko kakovostnih in strokovnih poslovnih aplikacij. Kontrole zagotavljajo hitrost in varnost. Za razvijalce, ki so že delali z ASP.NET AJAX-om, je uporaba 'Rad Controls' zelo enostavna. Vmesnik API za delo s kontrolami je zelo podoben privzetemu vmesniku API kontrol ASP.NET. Telerikove kontrole vsebujejo enake dogodke, poimenovanja in osnovne metode kot jih Microsoftovo ogrodje. [11]

3.4.2 Telerik Reporting

Telerik Reporting je enostavna rešitev za izvoz in prikaz podatkov v željeni dokument.
Knjižnico Telerik Reporting je možno vključiti tako v spletne aplikacije (ASP.NET, Silverlight, kakor tudi v namizne aplikacije (navadne aplikacije windows, WPF).
Za prikaz in analizo podatkov je na voljo veliko različnih predstavitvenih komponent, kot so npr. grafi, tabele, agregacija podatkov in druge.
Podpira izvoz v veliko različnih formatov za izvoz podatkov, kot so Excel, XPS/XAML, MHTML,PDF,CSV, RTF. Poleg naštetih je možno ustvariti tudi sliko kreiranega poročila. Formati slike so lahko sledeči: BMP, GIF, JPEG, PNG, TIFF.
Glede nato, da je Telerik Reporting odvisen od standarnih .NET objektov, ga je možno povezati s katerimkoli virom podatkov. Tako je možna povezava z vsemi ADO.NET ponudniki (SQL Server, MySQL, Oracle, OLE DB), poslovnimi objekti, MS Accesom, XML, spletnimi servisi (Web Services).
Za postavitev elementov na poročilu je možno uporabiti čarovnika ter vnaprej pripravljene stile in izglede. Pri postavljanju elementov na poročilo je na voljo tudi predogled kako naj bi poročilo na koncu izgledalo. Gradnja in postavljanje elementov na poročilo je enaka postavljanju grafičnih elementov na uporabniški vmesnik v aplikaciji. Enako je z nastavljanjem lastnosti elementom.
Po namestitvi primerne Telerik Reporting knjižnice, se v Visual Studiu v oknu z zavihku z orodji dodajo elementi, ki jih je možno dodati v poročilo. Možna je seveda uporaba t.i. mehanizma 'primi in spusti'. [12]

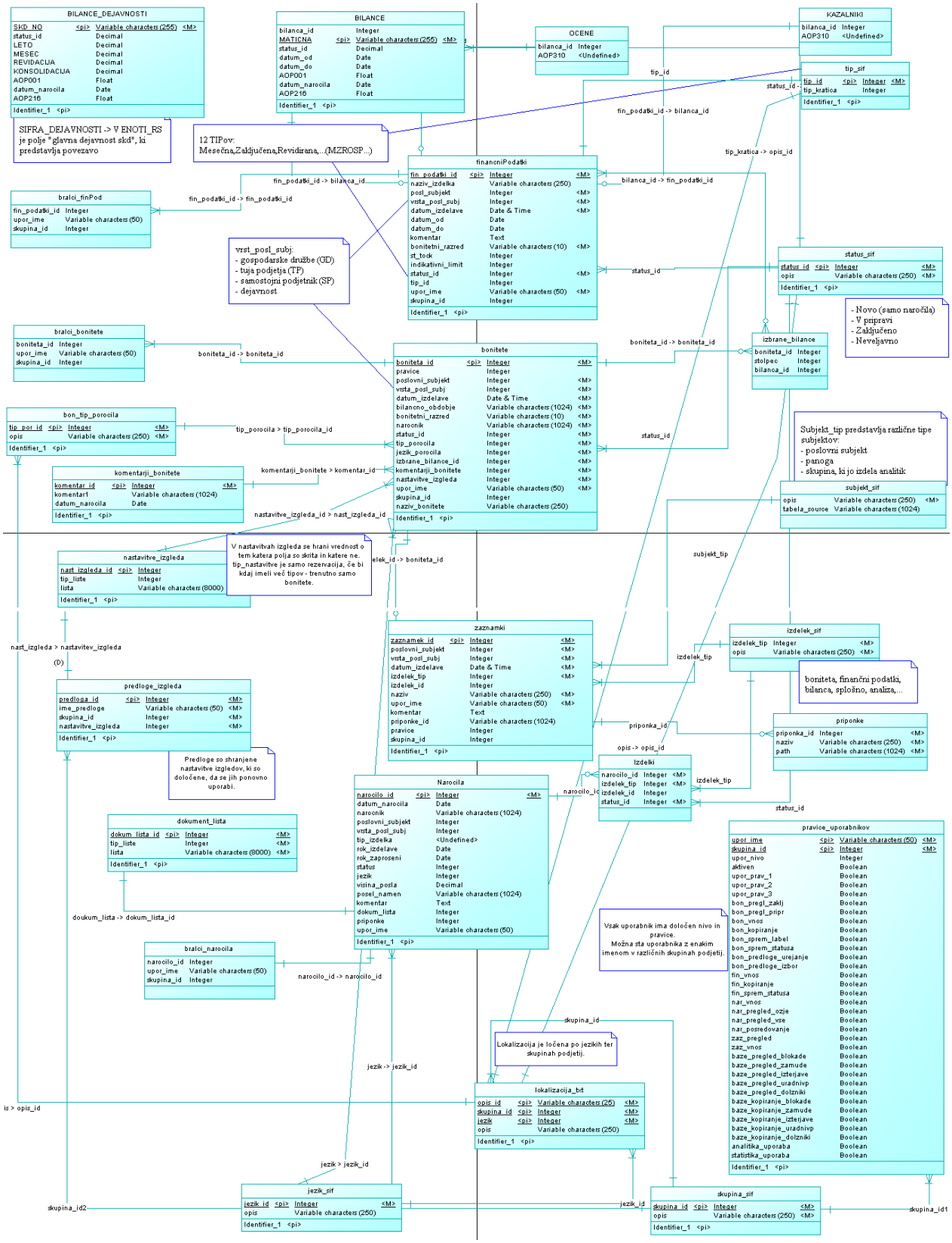
4 Aplikacija Bonitetni informacijski sistem

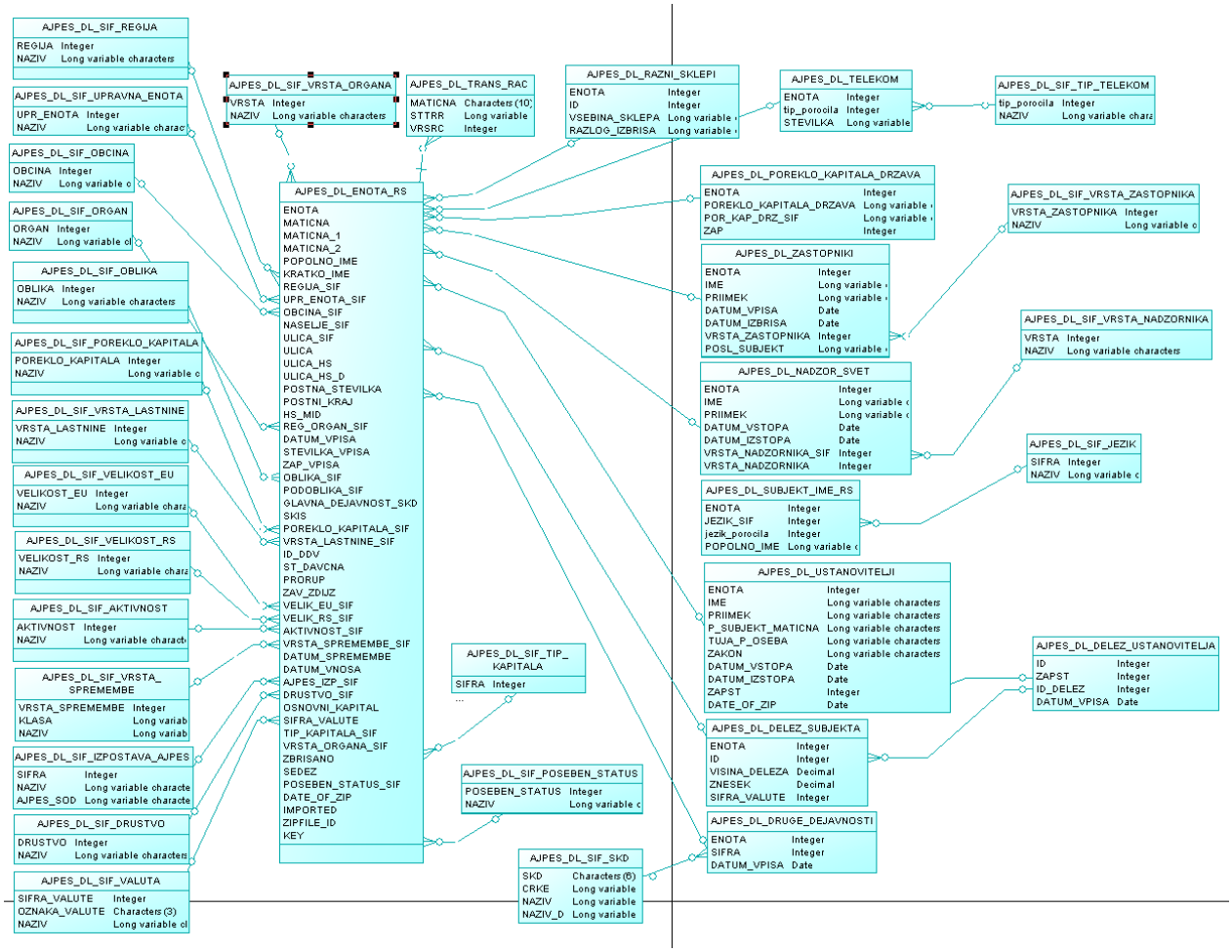
4.1 Diagram primerov uporabe



Slika 2: Diagram primerov uporabe

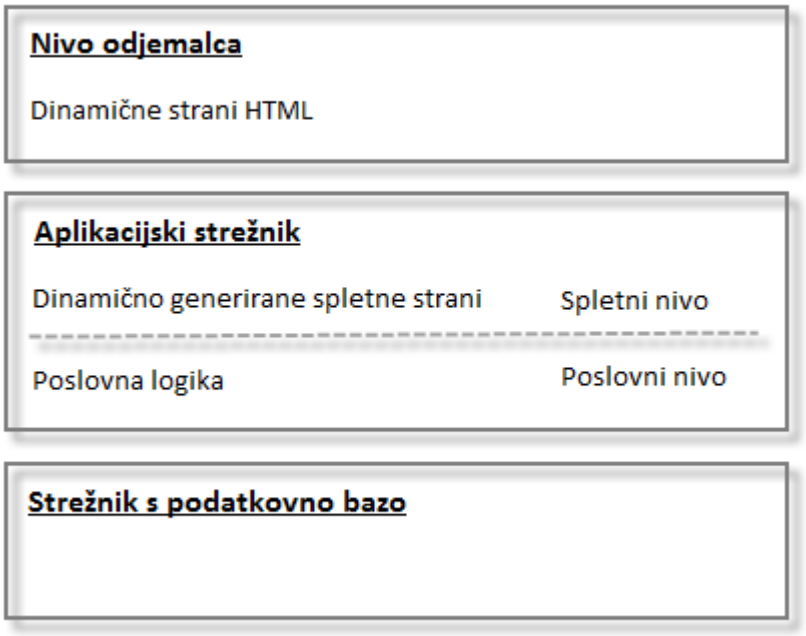
4.2 Konceptualni model podatkovnega modela aplikacije


BILANCE_DEJAVNOSTI
SKD_NO <pi> Variable characters (255) <M>
status_id Decimal
LETO Decimal
MESEC Decimal
REVIDACIJA Decimal
KONSOLIDACIJA Decimal
AOP001 Float
datum_narocila Date
AOP216 Float
Identifier_1 <pi>
SIFRA_DEJAVNOSTI -> V ENOTI_RS je polje "glavna dejavnost skd", ki predstavlja povezavo
BILANCE
bilanca_id Integer
MATICNA <pi> Variable characters (255) <M>
status_id Decimal
datum_od Date
datum_do Date
AOP001 Float
datum_narocila Date
AOP216 Float
Identifier_1 <pi>
12 TIPov: Mesečna,Zaključena,Revidirana,...(MZROSP...)
OCENE
bilanca_id Integer
AOP310 <Undefined>
KAZALNIKI
bilanca_id Integer
AOP310 <Undefined>
tip_sif
tip_id <pi> Integer <M>
tip_kratica Integer
Identifier_1 <pi>
tip_id
status_id
fin_podatki_id -> bilanca_id
tip_kratica -> opis_id
financniPodatki
fin_podatki_id <pi> Integer <M>
naziv_izdelka Variable characters (250)
posl_subjekt Integer <M>
vrsta_posl_subj Integer <M>
datum_izdelave Date & Time <M>
datum_od Date
datum_do Date
komentar Text
bonitetni_razred Variable characters (10) <M>
st_tock Integer
indikativni_limit Integer
status_id Integer <M>
tip_id Integer
upor_ime Variable characters (50) <M>
skupina_id Integer
Identifier_1 <pi>
fin_podatki_id -> bilanca_id
bilanca_id -> fin_podatki_id
bralci_finPod
fin_podatki_id Integer
upor_ime Variable characters (50)
skupina_id Integer
fin_podatki_id -> fin_podatki_id
vrst_posl_subj:
- gospodarske družbe (GD)
- tuja podjetja (TP)
- samostojni podjetnik (SP)
- dejavnost
status_sif
status_id <pi> Integer <M>
opis Variable characters (250) <M>
Identifier_1 <pi>
status_id
- Novo (samo naročila)
- V pripravi
- Zaključeno
- Neveljavno
bralci_bonitete
boniteta_id Integer
upor_ime Variable characters (50)
skupina_id Integer
boniteta_id -> boniteta_id
bonitete
boniteta_id <pi> Integer <M>
pravice Integer
poslovni_subjekt Integer <M>
vrsta_posl_subj Integer <M>
datum_izdelave Date & Time <M>
bilanco_obdobje Variable characters (1024) <M>
bonitetni_razred Variable characters (10) <M>
narocnik Variable characters (1024) <M>
status_id Integer <M>
tip_porocila Integer <M>
jezik_porocila Integer <M>
izbrane_bilance_id Integer <M>
komentarji_bonitete Integer <M>
nastavitve_izgleda Integer <M>
upor_ime Variable characters (50) <M>
skupina_id Integer
naziv_bonitete Variable characters (250)
Identifier_1 <pi>
boniteta_id -> boniteta_id
izbrane_bilance
boniteta_id Integer
stolpec Integer
bilanca_id Integer
bon_tip_porocila
tip_por_id <pi> Integer <M>
opis Variable characters (250) <M>
Identifier_1 <pi>
tip_porocila > tip_porocila_id
komentarji_bonitete
komentar_id <pi> Integer <M>
komentar1 Variable characters (1024)
datum_narocila Date
Identifier_1 <pi>
komentarji_bonitete > komentar_id
nastavitve_izgleda_id > nast_izgleda_id
Subjekt_tip predstavlja različne tipe subjektov:
- poslovni subjekt
- panoga
- skupina, ki jo izdela analitik
subjekt_sif
opis Variable characters (250) <M>
tabela_source Variable characters (1024)
V nastavitvah izgleda se hrani vrednost o tem katera polja so skrita in katere ne. tip_nastavitve je samo rezervacija, če bi kdaj imeli več tipov - trenutno samo bonitete.
nastavitve_izgleda
nast_izgleda_id <pi> Integer
tip_liste Integer
lista Variable characters (8000)
Identifier_1 <pi>
subjekt_tip
zaznamki
zaznamek_id <pi> Integer <M>
poslovni_subjekt Integer <M>
vrsta_posl_subj Integer <M>
datum_izdelave Date & Time <M>
izdelek_tip Integer <M>
izdelek_id Integer
naziv Variable characters (250) <M>
upor_ime Variable characters (50) <M>
komentar Text
priponke_id Variable characters (1024)
pravice Integer
skupina_id Integer
Identifier_1 <pi>
izdelek_sif
izdelek_tip Integer <M>
opis Variable characters (250) <M>
boniteta, finančni podatki, bilanca, splošno, analiza,...
priponke
priponka_id Integer <M>
naziv Variable characters (250) <M>
path Variable characters (1024) <M>
priponka_id
nast_izgleda > nastavitev_izgleda
(D)
predloge_izgleda
predloga_id <pi> Integer <M>
ime_predloge Variable characters (50) <M>
skupina_id Integer <M>
nastavitve_izgleda Integer <M>
Identifier_1 <pi>
Predloge so shranjene nastavitve izgledov, ki so določene, da se jih ponovno uporabi.
izdelki
narocilo_id Integer <M>
izdelek_tip Integer <M>
izdelek_id Integer
status_id Integer <M>
opis -> opis_id
izdelek_tip
status_id
Narocila
narocilo_id <pi> Integer <M>
datum_narocila Date
narocnik Variable characters (1024)
poslovni_subjekt Integer
vrsta_posl_subj Integer
tip_izdelka <Undefined>
rok_izdelave Date
rok_zaproseni Date
status Integer
jezik Integer
visina_posla Decimal
posel_namen Variable characters (1024)
komentar Text
dokum_lista Integer
priponke Integer
upor_ime Variable characters (50)
Identifier_1 <pi>
narocilo_id
dokument_lista
dokum_lista_id <pi> Integer <M>
tip_liste Integer <M>
lista Variable characters (8000) <M>
Identifier_1 <pi>
doukum_lista -> dokum_lista_id
pravice_uporabnikov
upor_ime <pi> Variable characters (50) <M>
skupina_id <pi> Integer <M>
upor_nivo Integer
aktiven Boolean
upor_prav_1 Boolean
upor_prav_2 Boolean
upor_prav_3 Boolean
bon_pregl_zaklj Boolean
bon_pregl_pripr Boolean
bon_vnos Boolean
bon_kopiranje Boolean
bon_sprem_label Boolean
bon_sprem_statusa Boolean
bon_predloge_urejanje Boolean
bon_predloge_izbor Boolean
fin_vnos Boolean
fin_kopiranje Boolean
fin_sprem_statusa Boolean
nar_vnos Boolean
nar_pregled_ozje Boolean
nar_pregled_vse Boolean
nar_posredovanje Boolean
zaz_pregled Boolean
zaz_vnos Boolean
baze_pregled_blokade Boolean
baze_pregled_zamude Boolean
baze_pregled_izterjave Boolean
baze_pregled_uradnivp Boolean
baze_pregled_dolzniki Boolean
baze_kopiranje_blokade Boolean
baze_kopiranje_zamude Boolean
baze_kopiranje_izterjave Boolean
baze_kopiranje_uradnivp Boolean
baze_kopiranje_dolzniki Boolean
analitika_uporaba Boolean
statistika_uporaba Boolean
Identifier_1 <pi>
Vsak uporabnik ima določen nivo in pravice.
Možna sta uporabnika z enakim imenom v različnih skupinah podjetij.
bralci_narocila
narocilo_id Integer
upor_ime Variable characters (50)
skupina_id Integer
narocilo_id -> narocilo_id
skupina_id
Lokalizacija je ločena po jezikih ter skupinah podjetij.
jezik -> jezik_id
lokalizacija_txt
opis_id <pi> Variable characters (25) <M>
skupina_id <pi> Integer <M>
jezik <pi> Integer <M>
opis Variable characters (250)
Identifier_1 <pi>
is > opis_id
jezik > jezik_id
jezik_sif
jezik_id <pi> Integer <M>
opis Variable characters (250)
Identifier_1 <pi>
skupina_id2
jezik_id
skupina_sif
skupina_id <pi> Integer <M>
opis Variable characters (250) <M>
Identifier_1 <pi>
skupina_id1


AJPES_DL_SIF_REGIJA
REGIJA Integer
NAZIV Long variable characters
AJPES_DL_SIF_UPRAVNA_ENOTA
UPR_ENOTA Integer
NAZIV Long variable charac
AJPES_DL_SIF_OBCINA
OBCINA Integer
NAZIV Long variable c
AJPES_DL_SIF_ORGAN
ORGAN Integer
NAZIV Long variable ch
AJPES_DL_SIF_OBLIKA
OBLIKA Integer
NAZIV Long variable characters
AJPES_DL_SIF_POREKLO_KAPITALA
POREKLO_KAPITALA Integer
NAZIV Long variable c
AJPES_DL_SIF_VRSTA_LASTNINE
VRSTA_LASTNINE Integer
NAZIV Long variable c
AJPES_DL_SIF_VELIKOST_EU
VELIKOST_EU Integer
NAZIV Long variable chara
AJPES_DL_SIF_VELIKOST_RS
VELIKOST_RS Integer
NAZIV Long variable chara
AJPES_DL_SIF_AKTIVNOST
AKTIVNOST Integer
NAZIV Long variable charact
AJPES_DL_SIF_VRSTA_SPREMEMBE
VRSTA_SPREMEMBE Integer
KLASA Long variab
NAZIV Long variab
AJPES_DL_SIF_IZPOSTAVA_AJPES
SIFRA Integer
NAZIV Long variable characte
AJPES_SOD Long variable characte
AJPES_DL_SIF_DRUSTVO
DRUSTVO Integer
NAZIV Long variable characters
AJPES_DL_SIF_VALUTA
SIFRA_VALUTE Integer
OZNAKA_VALUTE Characters (3)
NAZIV Long variable ch
AJPES_DL_SIF_VRSTA_ORGANA
VRSTA Integer
NAZIV Long variable characters
AJPES_DL_TRANS_RAC
MATICNA Characters (10)
STTRR Long variable
VRSRC Integer
AJPES_DL_ENOTA_RS
ENOTA
MATICNA
MATICNA_1
MATICNA_2
POPOLNO_IME
KRATKO_IME
REGIJA_SIF
UPR_ENOTA_SIF
OBCINA_SIF
NASELJE_SIF
ULICA_SIF
ULICA
ULICA_HS
ULICA_HS_D
POSTNA_STEVILKA
POSTNI_KRAJ
HS_MID
REG_ORGAN_SIF
DATUM_VPISA
STEVILKA_VPISA
ZAP_VPISA
OBLIKA_SIF
PODOBLIKA_SIF
GLAVNA_DEJAVNOST_SKD
SKIS
POREKLO_KAPITALA_SIF
VRSTA_LASTNINE_SIF
ID_DDV
ST_DAVCNA
PRORUP
ZAV_ZDIJZ
VELIK_EU_SIF
VELIK_RS_SIF
AKTIVNOST_SIF
VRSTA_SPREMEMBE_SIF
DATUM_SPREMEMBE
DATUM_VNOSA
AJPES_IZP_SIF
DRUSTVO_SIF
OSNOVNI_KAPITAL
SIFRA_VALUTE
TIP_KAPITALA_SIF
VRSTA_ORGANA_SIF
ZBRISANO
SEDEZ
POSEBEN_STATUS_SIF
DATE_OF_ZIP
IMPORTED
ZIPFILE_ID
KEY
AJPES_DL_SIF_TIP_KAPITALA
SIFRA Integer
...
AJPES_DL_SIF_POSEBEN_STATUS
POSEBEN_STATUS Integer
NAZIV Long variable c
AJPES_DL_SIF_SKD
SKD Characters (6)
CRKE Long variable
NAZIV Long variable
NAZIV_D Long variable
AJPES_DL_RAZNI_SKLEPI
ENOTA Integer
ID Integer
VSEBINA_SKLEPA Long variable
RAZLOG_IZBRISA Long variable
AJPES_DL_TELEKOM
ENOTA Integer
tip_porocila Integer
STEVILKA Long variable
AJPES_DL_SIF_TIP_TELEKOM
tip_porocila Integer
NAZIV Long variable chara
AJPES_DL_POREKLO_KAPITALA_DRZAVA
ENOTA Integer
POREKLO_KAPITALA_DRZAVA Long variable
POR_KAP_DRZ_SIF Long variable
ZAP Integer
AJPES_DL_ZASTOPNIKI
ENOTA Integer
IME Long variable
PRIIMEK Long variable
DATUM_VPISA Date
DATUM_IZBRISA Date
VRSTA_ZASTOPNIKA Integer
POSL_SUBJEKT Long variable
AJPES_DL_SIF_VRSTA_ZASTOPNIKA
VRSTA_ZASTOPNIKA Integer
NAZIV Long variable c
AJPES_DL_NADZOR_SVET
ENOTA Integer
IME Long variable
PRIIMEK Long variable
DATUM_VSTOPA Date
DATUM_IZSTOPA Date
VRSTA_NADZORNIKA_SIF Integer
VRSTA_NADZORNIKA Integer
AJPES_DL_SIF_VRSTA_NADZORNIKA
VRSTA Integer
NAZIV Long variable characters
AJPES_DL_SIF_JEZIK
SIFRA Integer
NAZIV Long variable c
AJPES_DL_SUBJEKT_IME_RS
ENOTA Integer
JEZIK_SIF Integer
jezik_porocila Integer
POPOLNO_IME Long variable
AJPES_DL_USTANOVITELJI
ENOTA Integer
IME Long variable characters
PRIIMEK Long variable characters
P_SUBJEKT_MATICNA Long variable characters
TUJA_P_OSEBA Long variable characters
ZAKON Long variable characters
DATUM_VSTOPA Date
DATUM_IZSTOPA Date
ZAPST Integer
DATE_OF_ZIP Date
AJPES_DL_DELEZ_USTANOVITELJA
ID Integer
ZAPST Integer
ID_DELEZ Integer
DATUM_VPISA Date
AJPES_DL_DELEZ_SUBJEKTA
ENOTA Integer
ID Integer
VISINA_DELEZA Decimal
ZNESEK Decimal
SIFRA_VALUTE Integer
AJPES_DL_DRUGE_DEJAVNOSTI
ENOTA Integer
SIFRA Integer
DATUM_VPISA Date

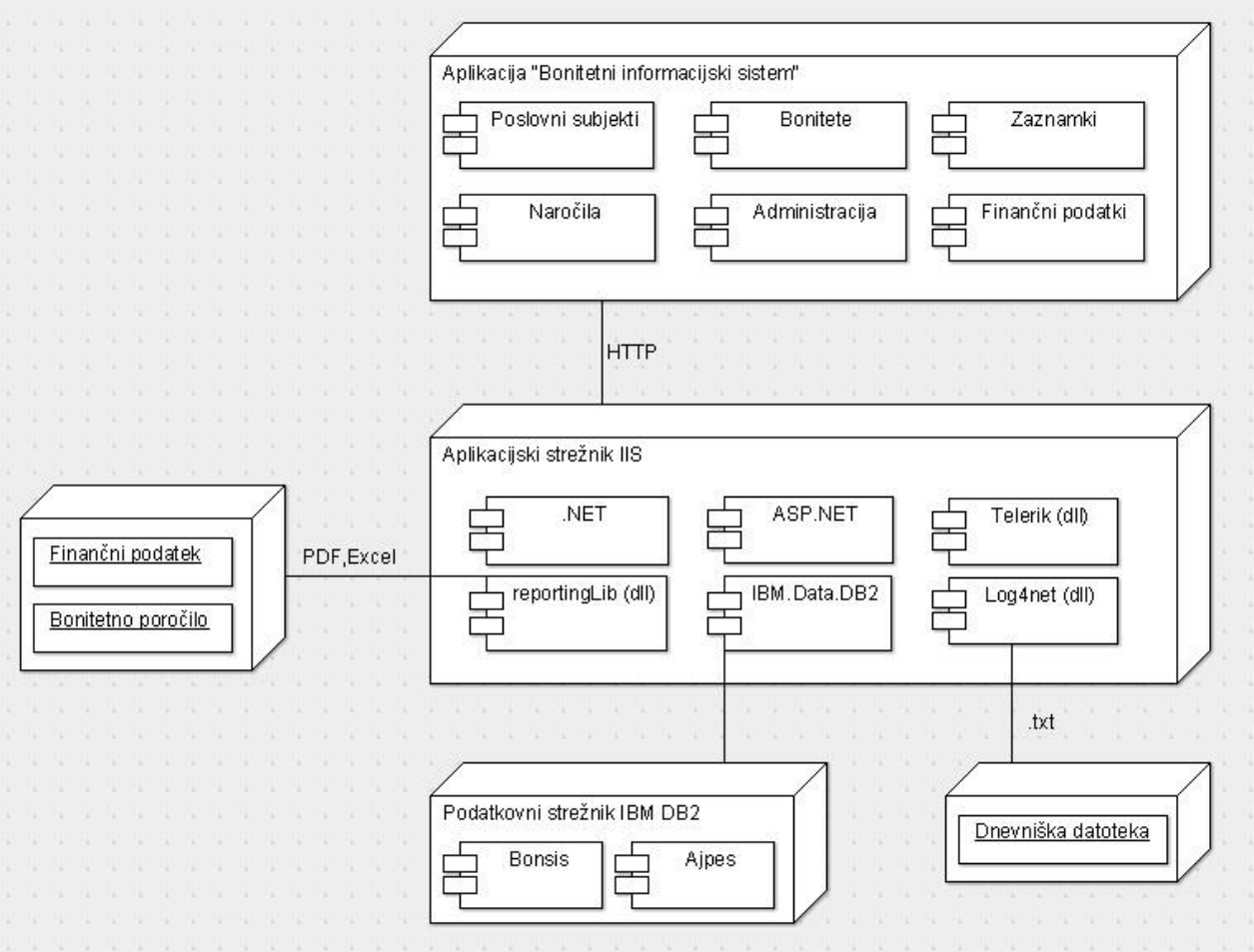
4.3 Arhitektura bonitetnega informacijskega sistema

Bonitetni informacijski system, ki ga opisujem v diplomski nalogi, vsebuje vse gradnike, ki so potrebni za sodobno spletno aplikacijo in ima tipični trinivojsko zgradbo.



Slika 3: trinivojska arhitektura aplikacije

Za potrebe interaktivnosti in odzivnosti uporabniškega vmesnika skrbi tehnika Ajax. Uporabljene so napredne in na izgled dovpadljive kontrole Telerik. Dinamične spletne strani so razvite z Microsoftovo tehnologijo ASP.NET, z dodanimi, že prej omenjenimi gradniki Telerik-a. Celotna poslovna logika je spisana v programskem jeziku C#. Podatki se hranijo na IBM-ovi podatkovni bazi DB2.



Slika 4: arhitektura bonitetnega informacijskega sistema

Bonitetni informacijski sistem vsebuje 6 glavnih modulov: Poslovni subjekti, Bonitete, Zaznamki, Naročila, Administracija ter Finančni podatki. Omogoča izvoz podatkov bodisi v obliki pdf, bodisi v obliki excel.

4.4 Prijava v sistem

4.4.1 Avtentikacija windows

Prijava z uporabniškim imenom in geslom, je verjetno danes najbolj razširjen način preverjanja pristnosti uporabnika pri dostopu v aplikacijo. Obstajajo pa še nekateri drugi načini. Za aplikacijo, ki jo bo uporabljala znana množica uporabnikov in kateri že imajo uporabniške račune windows, je tako zelo primerna uporaba avtentikacije windows. Takšen sistem preverjanja pristnosti dostopa uporablja informacije o že obstoječih uporabnikih.

Pri implementaciji uporabe avtentikacije windows, je potrebno vedeti, da le ta ni del okolja ASP.NET.

Delovanje avtentikacije windows je sledeče: ob dostopu do aplikacije preko brskalnik, strežnik (IIS) zahteva, da se brskalnik avtenticira. Brskalnik se avtenticira tako, da poda ustrezna priporočila, katere kažejo na uporabniški račun v okolju windows. V primeru, da je uporabnik uspešno avtenticiran, IIS dovoli zahtevo po spletni strani ter pošlje informacije o uporabniku in njegovi okolju ASP.NET.

Integracija avtentikacije windows v ASP.NET aplikaciji zahteva 3 korake:

1. Konfiguracija tipa avtentikacije windows na strežniku IIS
2. Konfiguracija datoteka web.config ASP.NET aplikacij (Slika 5)
3. Omejitev ‘anonimnega dostopa’ za spletno stran, podmapo ali celotno aplikacijo

4.4.2 Postopek avtentikacije

```
<!--
  The <authentication> section enables configuration
  of the security authentication mode used by
  ASP.NET to identify an incoming user.
-->
<authentication mode="Windows" />
<!--
```

Slika 5: avtentikacija windows

Za prijavo uporabnika v bonitetni informacijski sistem, se uporablja avtentikacija windows. Z avtentikacijo windows, sistem poskuša pridobiti uporabniško ime trenutno prijavljenega uporabnika v operacijskem sistemu windows. [7]

```
if (HttpContext.Current.Session[USERNAME] == null)
{
    System.Security.Principal.WindowsPrincipal principal =
  (System.Security.Principal.WindowsPrincipal)System.Web.HttpContext.Current.User;
    System.Security.Principal.WindowsIdentity identity =
  (System.Security.Principal.WindowsIdentity)principal.Identity;

    string domena = "", uporabniskoIme = "";

    // anonymous --> hardcoded user
    if (identity.IsAnonymous)
    {
        uporabniskoIme = DBSimulation.HARDCODED_UPORABNIK.Username;
    }
    else if (identity.IsAuthenticated)
    {
        domena = principal.Identity.Name.Split('\\')[0];
        uporabniskoIme = principal.Identity.Name.Split('\\')[1];
    }
    HttpContext.Current.Session[USERNAME] = uporabniskoIme;
}
```

Slika 6: branje in nastavljanje uporabniškega imena v sejo

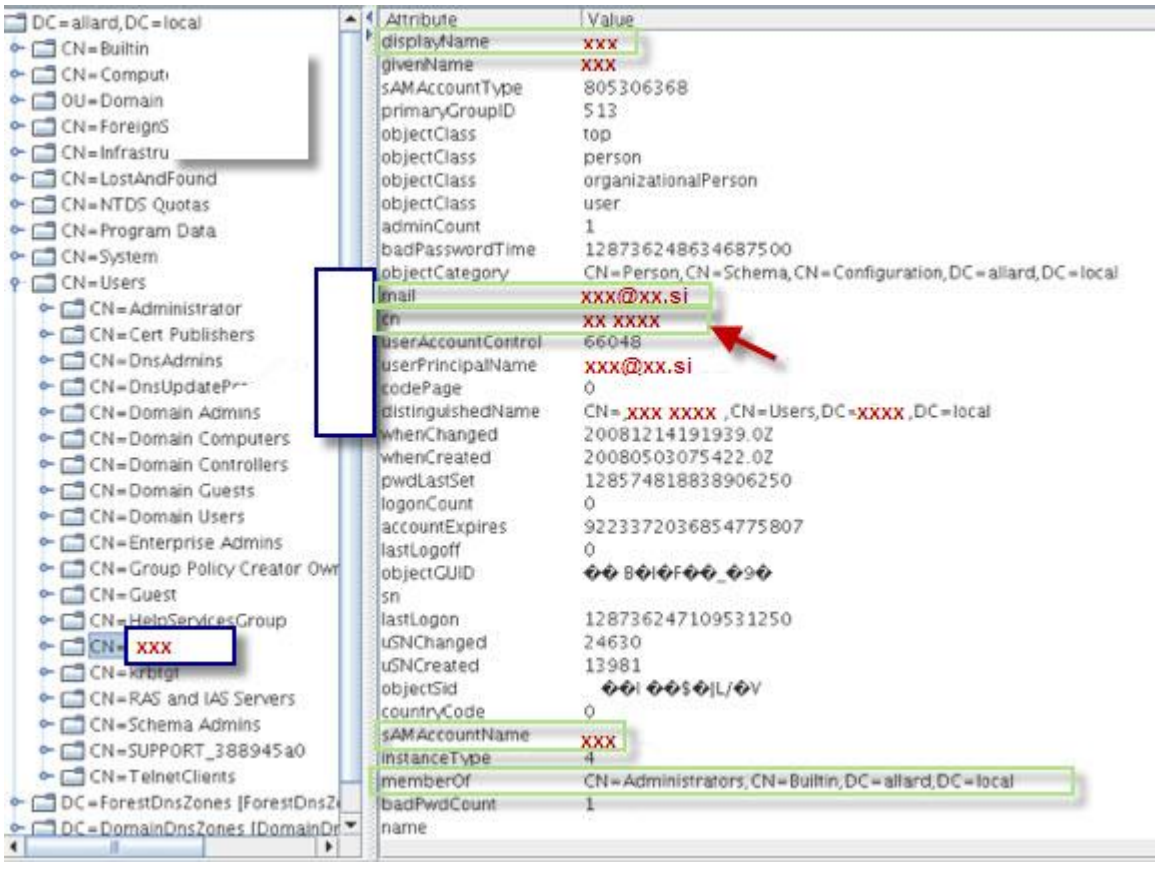
V primeru, da sistem uspešno pridobi uporabniško ime sledi shranjevanje le tega v trenutno sejo (Slika 6) Uporabniško ime se shrani v sejo zato, da se ga lahko hitro in enostavno prebere na vseh straneh in modulih aplikacije. Uporabniško ime se namreč potrebuje za delo s sistemom LDAP, za prikaz trenutno prijavljenega uporabnika ter v kombinaciji z bazo uporabnikov tudi za avtorizacijo posamzenih modulov aplikacije.

Prednost uporabe avtentikacije windows:

- Za implementacijo je potrebnega malo dela s strani razvijalca aplikacije
- Omogoča uporabo že obstoječih uporabnikov
- Zagotavlja enoten model az preverjanje pristnosti za več vrst aplikacij
- Možna uporaba varnosti windows

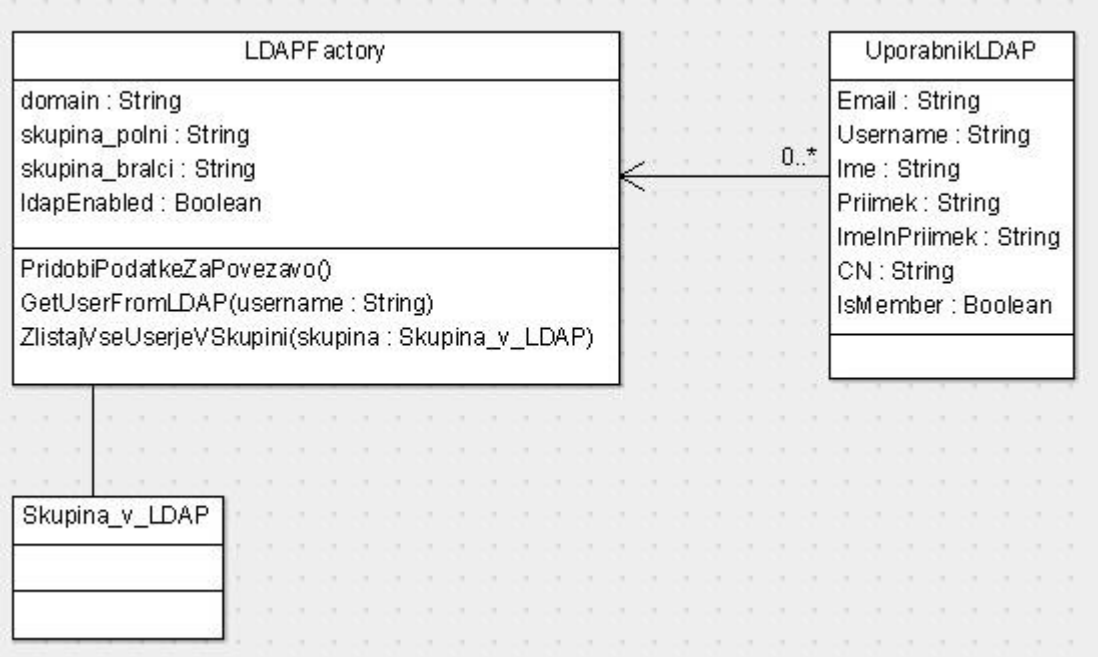
4.4.3 Izpis imena in priimka uporabnika v aplikaciji

Po uspešni avtentikaciji in shranjevanju uporabniškega imena v sejo, se naredi dostop do sistema LDAP, kjer so v imeniku shranjeni vsi uporabniki v domeni. Omrežno uporabniško ime je edinstven zapis v imeniku, tako, da ni možnosti, da bi obstajala 2 uporabnika z enakim uporabniškim imenom. Poleg uporabniškega imena, lahko posamezen zapis v sistemu LDAP vsebuje še nekatere druge podatke, kot so denimo ime, priimek (v kolikor gre za uporabnika ali skupino), ime za prikaz, podatki o naslovu elektronske pošte, podatke zadnji prijavi v sistem, podatke o času kreiranja in poteku zapisa, seznam vseh skupin katerim zapis pripada in še mnogo več. [13,14]
Shemo imenika LDAP prikazuje (Slika 7: shema imenika LDAP) . Za našo implementacijo so pomembni naslednji elementi (na sliki so označeni z zeleno obrobo): uporabniško ime, naslov elektronske pošte, ime za prikaz, skupine katerim uporabnik pripada. Zaradi zagotavljanja zasebnosti, so dejanske vrednosti ponekod spremenjene. Kot vidimo ima imenik LDAP t.i. "drevesno strukturo".



Slika 7: shema imenika LDAP

Za delo s sistemom LDAP je definiran razred LDAPFactory. (Slika 88). Za delo nad sistemom LDAP razred uporablja knjižnico System.DirectoryServices.
Ob inicializaciji objekta LDAPFactory, se iz konfiguracijske datoteke web.config preberejo podatki za dostop do sistema LDAP. Podatki, potrebni za dostop do sistema LDAP ter pregledovanja vrednosti imenika so sledeči: uporabniško ima, geslo, domena ter pot do LDAP sistema.



Slika 8: razredni diagram za delo s sistemom LDAP

Po uspešnem dostopu do sistema LDAP, lahko izvršimo poizvedbo o podatkih za izbrano uporabniško ime. V našem primeru, se takšna poizvedba izvrši ob klicu funkcije "GetUserFromLDAP(string username)" (Slika 9)

```
UporabnikLDAP uporabnikLDAP = new UporabnikLDAP();
uporabnikLDAP = (new LDAPfactory()).GetUserFromLDAP(uporabniskoIme);
HttpContext.Current.Session[DISPLAYNAME] = uporabnikLDAP.ImeInPriimek;
```

Slika 9: primer klica funkcije, ki vrne podatke iz sistema LDAP za iskano uporabniško ime

V primeru, da obstaja v imeniku LDAP zapis, katerega atribut `SAMAccountName` je enak iskanemu uporabniškemu imenu, se iz najdenega zapisa preberejo podatki: ime, priimek, naslov elektronske pošte. [15]
Podatek o imenu in priimku trenutno prijavljenega uporabnika prikažemo v aplikaciji (Slika 10), hkrati pa ga podobno, kot uporabniško ime, shranimo v sejo.

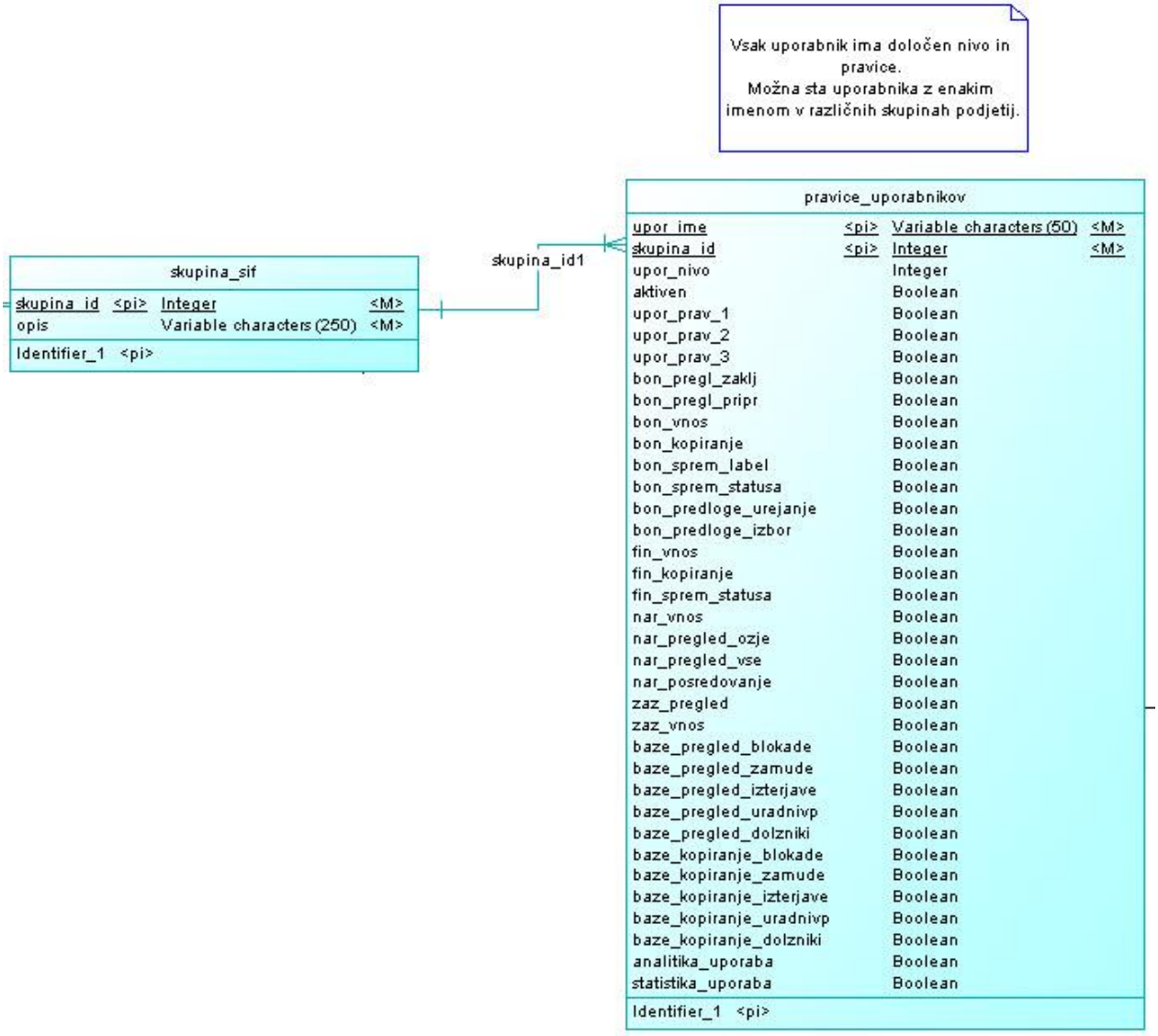
Slika 10: prikaz imena in priimka trenutno prijavljenega uporabnika v sistem

```
DirectoryEntry entry = new DirectoryEntry(
      adPath,adUsername,adPwd,AuthenticationTypes.Secure);
```

Slika 11: primer inicializacije objekta DirectoryEntry za dostop do LDAP-a

4.4.4 Avtorizacija

Poleg že omenjene avtentikacije oz. preverjanje pristnosti uporabnika, bonitetni informacijski sistem vsebuje tudi preverjanje avtorizacije za posameznega uporabnika. Preverjanej avtorizacije je vezano na posamezne module oz. posamezna strani informacijskega sistema. Tako ima lahko naprimer nek uporabnik vse pravice oz. pooblastila na strani Zaznamki, medtem ko le teh nima na strani Finančni podatki. V prvem primeru lahko pregleduje, vnaša, briše, posodablja podatke o zaznamkih, medtem, ko v drugem primeru nima nikakršnih pravic in mu zato aplikacija naročil ne dovoli niti pregledovati, kaj šele urejati.
Pravice in dovoljenja za posameznaga uporabnika so zapisane v podatkovni bazi. (Slika 12: baza uporabnikov). Vrednost, ki jo lahko zavzame posamezna pravica, in je definirana kot svoj atribut v bazi, je bodisi ‘true’, bodisi ‘false’. Vrednost ‘true’ pomeni da je uporabnik ima pravico in vrednost ‘false’ da je nima.
Pozor: PowerDesigner ob pretvorbi iz entitetnega modela v logičnega za IBM-ovo podatkovno bazo DB2, polja ki so tipa Boolean pretvori v SMALINT. Dejanske vrednosti v bazi so tako shranjene kot 1 (analogno predstavlja vrednost true) in 0 (false).



Slika 12: baza uporabnikov

Za vsako stran aplikacije je potrebno izvesti preverjanje pravic trenutnega uporabnika. Trenutnega uporabnika se preveri tako, da se iz seje prebere trenutno prijavljeni uporabnik oz. njegovo uporabniško ime (shranjevanje v sejo se je opravilo pri postopku avtentikacije, op.p.) ter skupina. Naredi se dostop do baze, natančneje do tabele ‘pravice_uporabnikov’ katere primarni ključ tvorita ‘upor_ime’ in ‘skupina_id’. Prebere se vrednosti vseh atributov. V kolikor je vrednost željenega atributa nastavljena na ‘true’ se trenutnemu uporabniku omogoči željena akcija. Za delovanje aplikacije je potrebno zagotoviti, da so uporabniška imena uporabnikov v podatkovni bazi enaka tistim iz sistema LDAP. Urejanje baze uporabnikov je trenutno možna samo administratorju sistema.

4.5 Uporaba dnevniške datoteke

Skoraj vsaka resna aplikacija vključuje programski vmesnik oz. modul za sledenje ter beleženje dogodkov. Vstavljanje ukazov v program, za izpis v dnevniško datoteko, je preprosta ter nezahtevna metoda. Njena naloga je beleženje dogodkov. V mnogih primerih pa tudi edini način razhroščevanja. Razhroščevalniki, ki jih nudijo nekatera razvojna okolja, in se s pridom uporabljajo med razvojem programske opreme, so namreč v končnem produktu neuporabna. Beleženje dogodkov v dnevniški datoteki predstavlja tako učinkovito odkrivanje morebitnih napak v programu ter pomoč pri odkrivanju nedelovanja zaradi nastavitev sistema, na katerem program teče. Hkrati pa nam služijo kot revizijska sled izvajanja programa oz. aplikacije.

Izkušnje kažejo, da je beleženje v dnevniško datoteko pomemben del razvojnega cikla. Ima več prednosti. Zagotavlja točno informacijo glede izvajanja programa. Ko se enkrat vstavi ukaz za beleženje v dnevniško datoteko v program, logika poskrbi za beleženje dogodkov v dnevniško datoteko. Z drugimi besedami povedano, beleženje dogodkov v dnevniško datoteko ne potrebuje človekovega posredovanja. Rezultati beleženja se shranijo in so uporabni za kasnejšo analizo.

Beleženje v dnevniško datoteko ima tudi slabnosti. Ena izmed njih je upošasnitev delovanja programa oz. aplikacije.

4.5.1 log4net

Za beleženje dogodkov v dnevniško datoteko, se v bonitetnem informacijskem sistemu uporablja orodje “log4net”. Log4net je del projekta “Apache Logging Services” in teče pod licenco “Apache License, Version 2.0”.

Nekatare lastnosti orodja ‘log4net’:

a. Različni načini beleženja
b. Hierarhična arhitektura beleženja
c. Podpora za več okvirjev
d. Visoka zmogljivost in prilagodljivost
e. Modularna in razširljiva oblika

f. Konfiguracija XML
Log4net se nastavi z uporabo nastavitvene datoteke XML. Informacije o konfiguraciji so lahko vključene v druge konfiguracijske datoteke XML (npr. web.config datoteka) ali v ločeni datoteki. Primer konfiguracije znotraj datoteke web.config prikazuje (Slika 13). Konfiguracija je lahko berljiva in se jo da z lahkoto posodobiti. Alternativno se lahko log4net nastavlja tudi programsko.

```
  <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,Log4net"/>

</configSections>

<log4net>
  <root>
    <level value="DEBUG" />
    <appender-ref ref="LogFileAppender" />
  </root>
  <appender name="LogFileAppender" type="log4net.Appender.RollingFileAppender" >
    <param name="File" value="App_Data/RadUploadTemp/LogFile.log" />
    <param name="AppendToFile" value="true" />
    <rollingStyle value="Size" />
    <maxSizeRollBackups value="10" />
    <maximumFileSize value="100MB" />
    <staticLogFileName value="true" />
    <layout type="log4net.Layout.PatternLayout">
      <param name="ConversionPattern" value="%d{dd.MM.yyyy HH:mm:ss} %logger [%method : %line] %-5level
            - %message%newline" />
    </layout>
  </appender>
</log4net>
```

Slika 13: primer konfiguracije log4net orodja v web.config datoteki

g. Dinamična konfiguracija
Log4net lahko spremlja svojo konfiguracijsko datoteko za spremembe in dinamično uporablja spremembe, narejene s konfiguratorjem. Ravni beleženja, dodajanja, izgleda se da nastaviti med izvajanjem kode. V mnogih primerih je mogoče diagnosticirati težave v aplikaciji, brez potreb po prekinitvi procesa.

h. Uveljavljena arhitektura
Log4net temelji na izjemno uspešni knjižnici za beleženje log4j, ki obstaja že od leta 1996. Ta popularna in uveljavljena arhitektura je bila do sedaj prenesena že v 12 različnih programskih jezikov.
[16]

4.5.2 log4net in bonitetni informacijski sistem

Za integracijo orodja log4net je potrebno vključiti primerno datoteko dll v proojekt. Besedno zvezo primerno je uporabljena zato, ker se ob prenosu paketa prenese več različnih datotek dll. Vsak posamezna je namenjena drugemu tipu projekta, npr. konzolna aplikacija uporablja knjižnico dll namenjeno konzolnim aplikacijam, spletna aplikacija zopet drugo knjižnico dll. Po namestitvi paketa disk in razpakiranju datotek na lokalni je potrebno v projekt dodati novo referenco, ki kaže na primerno knjižnico dll (Slika 14).

Slika 14: referenca log4net v projektu

V datoteki "web.config" nastavimo pot do datoteke (Slika 13), kamor se bodo shranjevali dnevniški zapisi. Tu velja omeniti, da orodje log4net ne podpira samo beleženja dnevniških zapisov v datoteko, temveč tudi nekatere druge oblike, kot so denimo zapisovanje v podatkovno bazo, izpisovanje v konzolnem oknu in še druge. V našem primeru, smo se odločili za zapisovanje v datoteko. V konfiguracijski datoteki "web.config" smo tudi nastavili željen format izpisa, največjo dovoljeno velikost datoteka za shranjevanje ter še nekatere druge parametre.

```
21.03.2011 07:37:19 bonSIS.utilities.bonitete.IskalnikBilancFactory
[PoisciCelotnoSekvencoBilanc : 147] DEBUG - 1009327[44872], 2005, 9
```

Slika 15: primer zapisa v log datoteki

V našem primeru, se ob posameznem zapisu v log datoteku zapišejo podatki o času nastopa dogodka, razred ter metoda, kjer se je kreiranje zapisa klicalo z dodano vrstico kode, ki nam olajša iskanje po kodi. Za temi podatki, se izpiše še katera metoda razreda ILog (debug, info, error, info, warm, fatal) je zapis kreirala. Na koncu pa še poljubnen podatek oz. poljubni podatki, ki jih ob klicu funkcije navedemo kot vhodne parametre. (Slika 16)

```
protected static readonly ILog log = LogManager.GetLogger(typeof(IskalnikBilancFactory));

public IskalnikBilancFactory()
{
    log4net.Config.XmlConfigurator.Configure();
    log.Debug("Test");
}
```

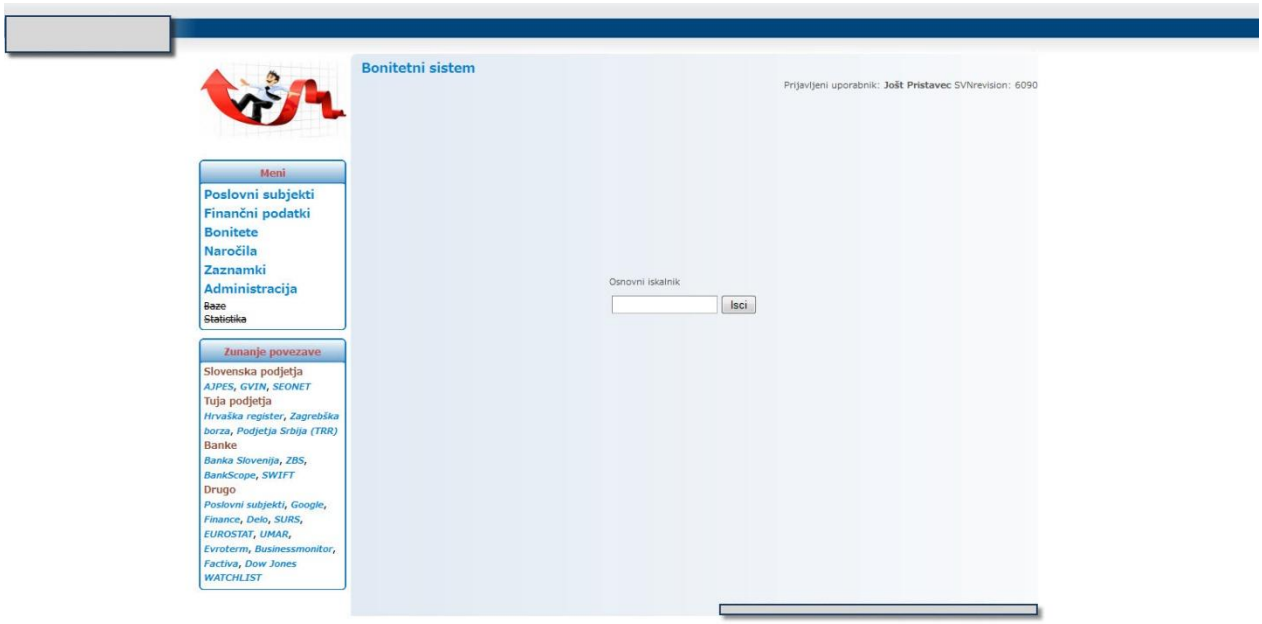
Slika 16: primer uporabe razreda ILog

4.6 Poslovni subjekti

4.6.1 Osnovni iskalnik

Osnovni iskalnik je na voljo na prvi strani aplikacije in obsega eno vnosno polje in gumb ‘Isci’. (Slika 17). Njegova naloga je iskanje določenega poslovnega subjekta. Iskanje je možno opraviti po enem izmed naslednjih parametrov poslovnega subjekta:
- nazivu oz. korenu naziva
- matična številka
- davčna številka

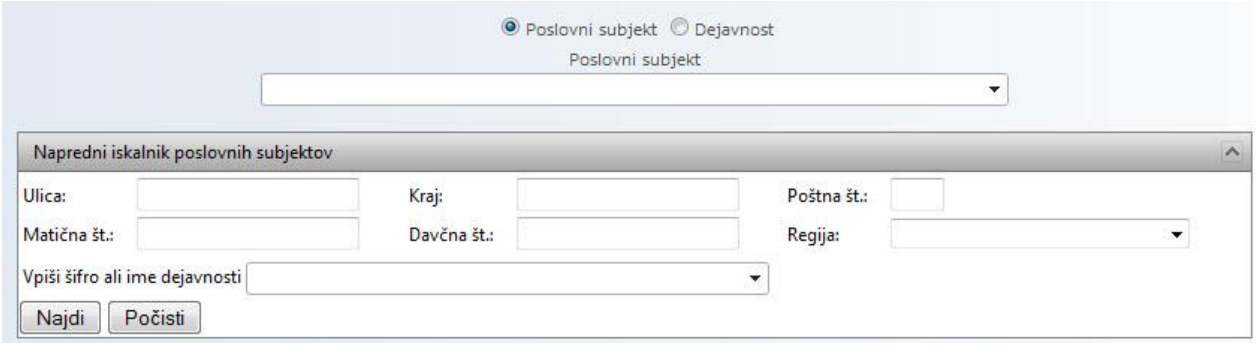
Po kliku gumba ‘Isci’ poslovna logika aplikacije sama izlušči za katerega izmed zgoraj navedenih paramaterov gre ter naredi preusmeritev na napredni iskalnik aplikacije. Pri preusmeritvi se prenesejo podatki o iskanem nizu ter podatek o tem po katerem izmed treh parametrov naj se izvede iskanje. Podatki se zapišejo kot parametri url-ja. Preusmeritev na napredni isklanik aplikacije je tu smiselna zaradi želje o ohranitvi enotnosti in nepotrebnega podvajanja.



Slika 17: osnovni iskalnik na prvi strani aplikacije

4.6.2 Napredni iskalnik

V naprednem iskalniku ima uporabnik, poleg iskanja po nazivu, tudi možnost iskanja po več različnih parametrih, kot so denimo naslov poslovnega subjekta, poštna številka, regija, davčna številka, kraj, matična številka ter dejavnost v katero je podjetje klasificirano oz. vpisano s strani AJPES-a.
V kolikor, se je dostop do naprednega iskalnika naredil z vnosom npr. davčne številke iskanega poslovnega subjekta v osnovnem iskalniku, se v naprednem iskalniku avtomatsko zapolni polje davčna številka ter izvede iskanje. Podobno velja seveda tudi za matično številko ter naziv.
Izgled naprednega iskalnika prikazuje (Slika 18).

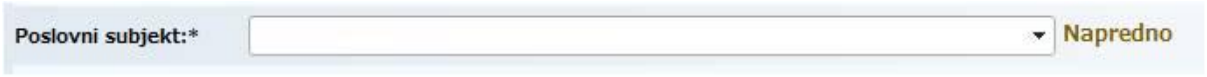


Slika 18: napredni iskalnik

Podatki o vseh poslovnih so zapisani v tabeli PRS_ENOTA_RS, katera se nahaja v shemi AJPES. Podatke o poslovnih subjektih smo dobili od AJPES-a. Podatkovni model je prikazan v poglavju 4.2.

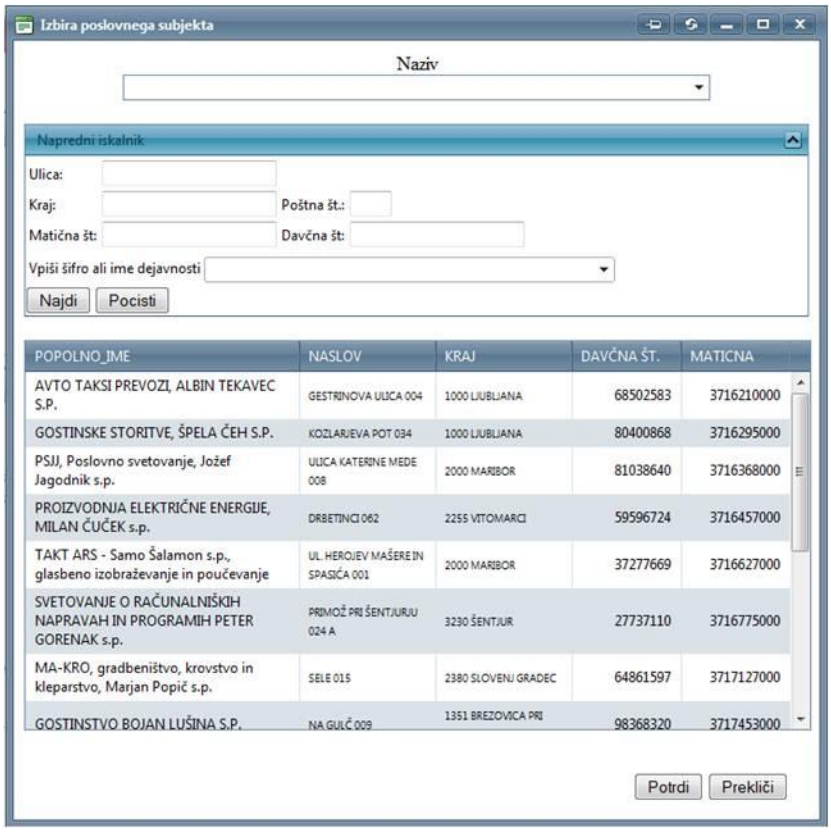
4.6.3 Kontrola za iskanje poslovnega subjekta

Potreba po kontroli za izbiro primernega poslovnega subjekta se pojavi na različnih modulih v aplikaciji. (npr. pri kreiranju zaznamka, bonitete, finančnega podatka). Z ta namen smo razvili komponento, katera vsebuje pametni spustni meni. Z vpisom naziva oz. dela naziva poslovnega subjekta, je le tega možno poiskati. Za namen, kjer nam osnovno iskanje poslovnega subjketa po njegovem nazivu ne zadostuje, imamo tudi gumb oz. povezavo do naprednega iskalnika.



Slika 19: uporabniška kontrola za osnovno iskanje poslovnega subjekta

Ob kliku na gumb se odpre pojavno okno z vnosnimi polji, s pomočjo katerih je možno opraviti napredno iskanje poslovnega subjekta. Tudi napredni iskalnik vsebuje spustni menu za iskanje po nazivu, poleg tega pa še vnosna polja za matično številko, davčno številko, poštno številko, kraj oz. sedež poslovnega subjekta, naslov oz. ulico ter možnost iskanja poslovnega po dejavnosti. Napredni iskanje se izvede ob kliku na gumb isci. Ko se najde željeni poslovni subjekt se ga izbere, bodisi z dvoklikom na izbrano vrstico, bodisi z enojmin klikom na vrstico ter klikom na gumb potrdi.

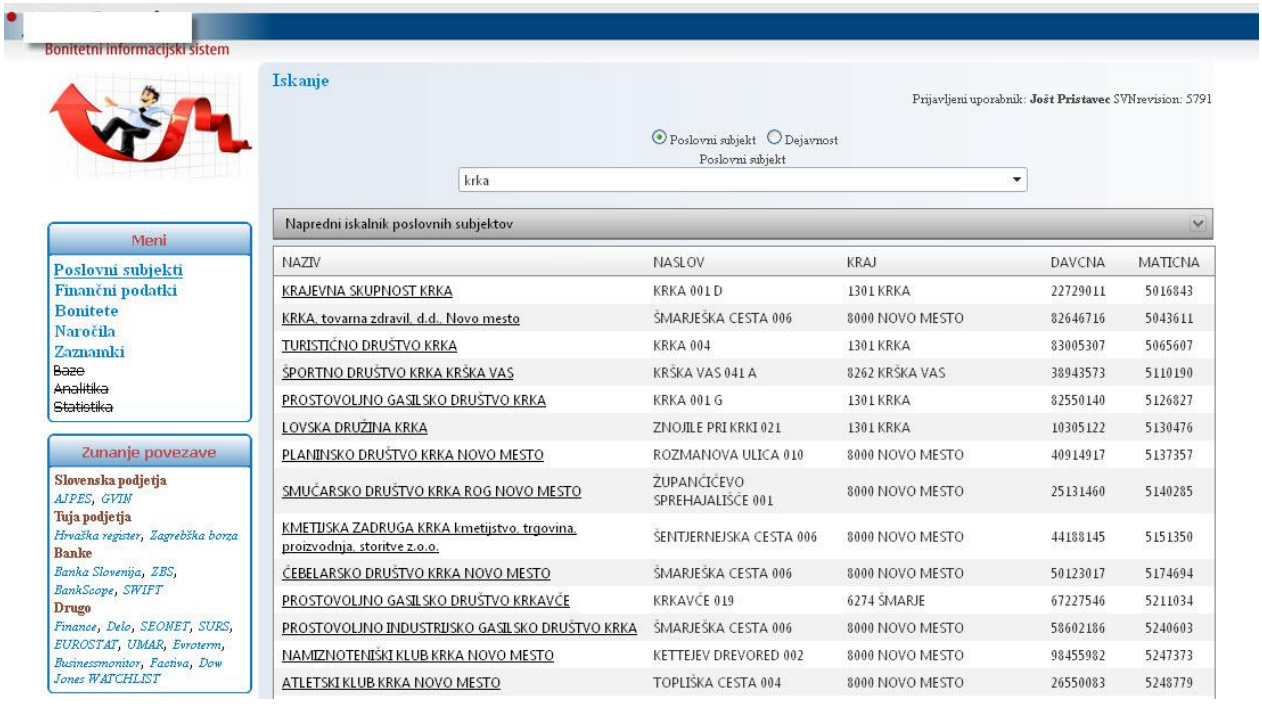


Slika 20: uporabniška kontrola za napredno iskanje poslovnega subjekta

Po opravljenem iskanju se v tabeli zadetkov prikažejo zapisi, ki ustrezajo rezutatu poizvedbe. V tabeli najdenih podjetij so prikazani naslednji stolpci podatkov iz baze AJPES: (Slika 21).

- naziv
- naslov
- kraj
- davčna številka
- matična številka

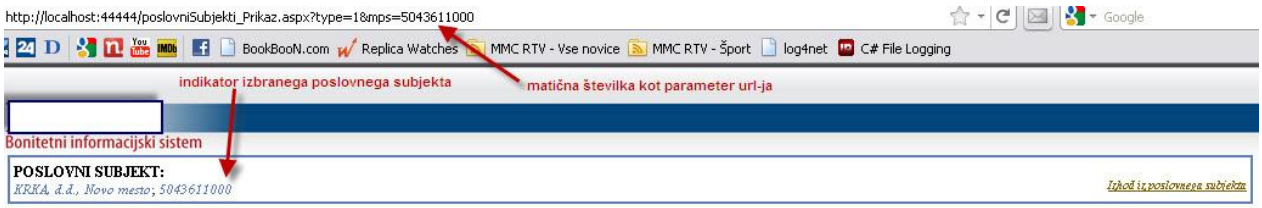
Po kliku na izbrano vrstico izbranega zadetka se naredi preusmeritev na stran s podrobnimi podatki o poslovnem subjektu. Pri tem se kot parametra urla-ja zapišeta matična številka ter tip poslovnega subjekta.



NAZIV	NASLOV	KRAJ	DAVCNA	MATICNA
KRAJEVNA SKUPNOST KRKA	KRKA 001 D	1301 KRKA	22729011	5016843
KRKA, tovarna zdravil, d.d., Novo mesto	ŠMARJEŠKA CESTA 006	8000 NOVO MESTO	82646716	5043611
TURISTIČNO DRUŠTVO KRKA	KRKA 004	1301 KRKA	83005307	5065607
ŠPORTNO DRUŠTVO KRKA KRŠKA VAS	KRŠKA VAS 041 A	8262 KRŠKA VAS	38943573	5110190
PROSTOVOLJNO GASILSKO DRUŠTVO KRKA	KRKA 001 G	1301 KRKA	82550140	5126827
LOVSKA DRUŽINA KRKA	ZNOJILE PRI KRKI 021	1301 KRKA	10305122	5130476
PLANINSKO DRUŠTVO KRKA NOVO MESTO	ROZMANOVA ULICA 010	8000 NOVO MESTO	40914917	5137357
SMUČARSKO DRUŠTVO KRKA ROG NOVO MESTO	ŽUPANČIČEVO SPREHAJALIŠČE 001	8000 NOVO MESTO	25131460	5140285
KMETIJSKA ZADRUGA KRKA kmetijstvo, trgovina, proizvodnja, storitve z.o.o.	ŠENTJERNEJSKA CESTA 006	8000 NOVO MESTO	44188145	5151350
ČEBELARSKO DRUŠTVO KRKA NOVO MESTO	ŠMARJEŠKA CESTA 006	8000 NOVO MESTO	50123017	5174694
PROSTOVOLJNO GASILSKO DRUŠTVO KRKAVČE	KRKAVČE 019	6274 ŠMARJE	67227546	5211034
PROSTOVOLJNO INDUSTRIJSKO GASILSKO DRUŠTVO KRKA	ŠMARJEŠKA CESTA 006	8000 NOVO MESTO	58602186	5240603
NAMIZNOTENIŠKI KLUB KRKA NOVO MESTO	KETTEJEV DREVORED 002	8000 NOVO MESTO	98455982	5247373
ATLETSKI KLUB KRKA NOVO MESTO	TOPLIŠKA CESTA 004	8000 NOVO MESTO	26550083	5248779

Slika 21: tabela najdenih poslovnih subjektov

S klikom na izbrani poslovni subjekt v tabeli zadetkov, se tudi izbere poslovni subjekt. Izbran poslovni subjekt pomeni, da se bodo od tu dalje vsi podatki celotne aplikacije nanašale samo na ta poslovni subjekt.
Primer: v kolikor bomo pregledovali zaznamke, bilance stanj, naročila se bodo vedno prikazovali filtrirani podatki vezani na izbrani poslovni subjekt. Da je poslovni subjekt izbran nam prikazuje indikator v zgornji predelu aplikacije (Slika 22). Zagotavljanje izbranega poslovnega subjekta na vseh straneh in podstraneh aplikacije je implementirano s parametrom url-ja. (Slika 22).

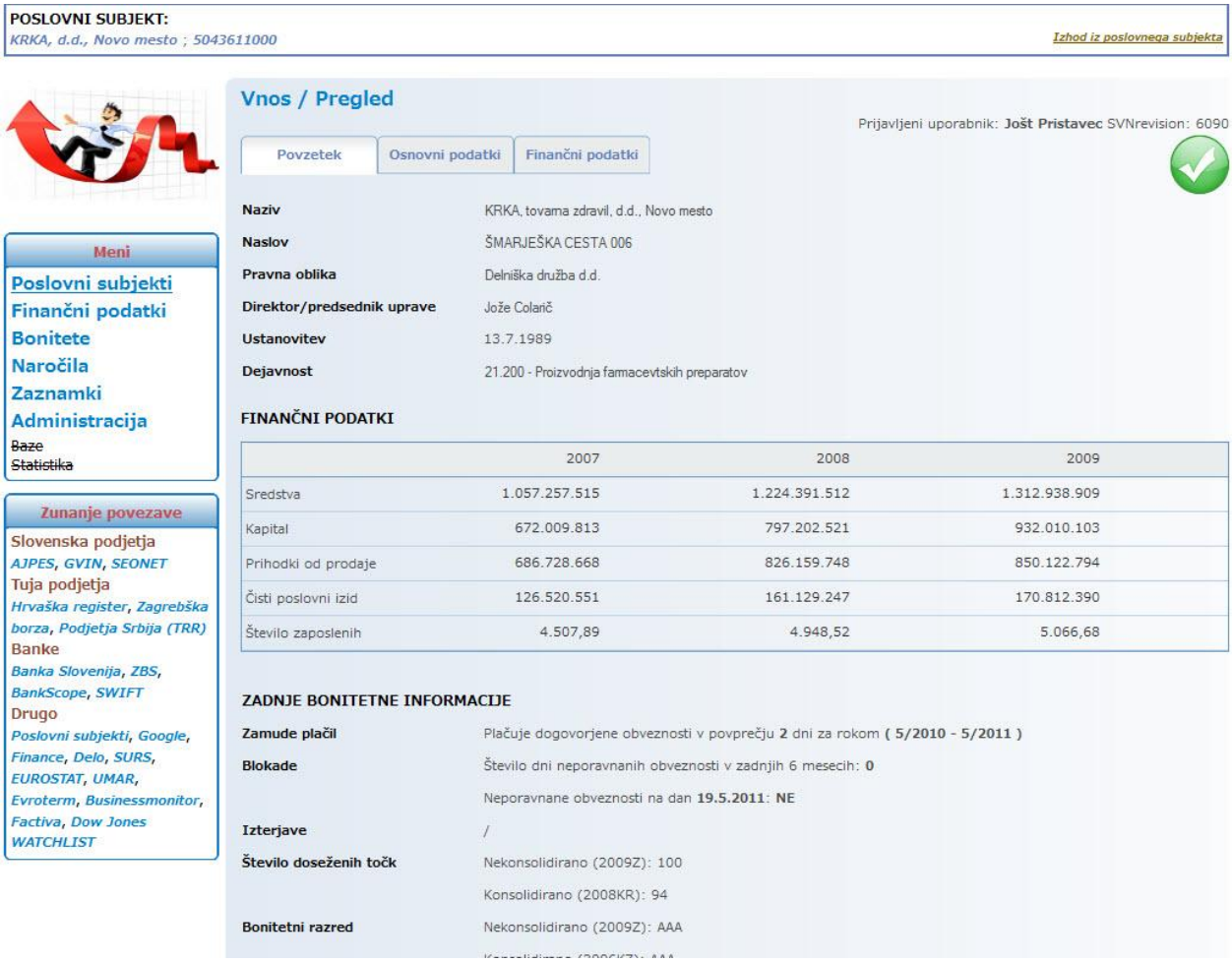


Slika 22: indikator izbranega poslovnega subjekta ter primer parametra url

4.6.4 Prikaz podatkov o poslovnem subjektu

Po izbiri poslovnega subjekta iz tabele najdenih poslovnih subjektov, ki ustrezajo iskanemu kriteriju se naredi preusmeritev na stran za prikazovanju podrobnih podatkov o poslovnem subjektu.
Modul poslovni subjekti sestavljajo trije sklopi: Povzetek, Osnovni podatki ter Finančni podatki (Slika 23). Med sklopi se sprehajamo s pomočjo menija. Za izbirni meni ter posamezna sklope se uporabljati telerikovi kontroli ‘RadPageView’ ter ‘RadMultiPage’. Znotraj posameznega sklopa pa še dodatno lastne uporabniške kontrole (.ascx)
Ker je podatkov zelo veliko, iskanje le teh pa poteka po več tabelah ter celo več shemah, je branje narejeno po delih oz. po principu ‘per-partes’. Podatki za posamezen sklop se berejo in zapisujejo ločeno. Za hranjenje podatkov se uporablja objekt ‘PoslovniSubjekt’. Branje podatkov za posamezen sklop se sproži ob prehodu oz. ob kliku na zavihek. Še več, aplikacija naredi dostop do baze samo prvič, nato se celoten objekt shrani v ViewState. Prednost hranjenja s pomočjo ViewState-a je, da ohranja vrednosti med prehajanjem med stranmi brez uporabe hranjenja podatkov v seji. V našem primeru v ViewState shranimo kar celoten objekt ‘PoslovniSubjekt’, zato je v razredu, ki definira omenjeni objekt, konstruktorju dodan atribut ‘Serializable’. Ob ponovnem obisku željenega modula, aplikacija ugotovi, da so bili podatki iz baze že prebrani in namesto dostopa do podatkovne baze, naredi dostop do objekta shranjenega v ViewState-u. Kombinacija branja po delih ter hranjenje podatkov v ViewStatu znatno doprimore k hitrejši odzivnosti sistema.

Za raliko od vseh ostalih modulov, se dovoli dostop do pregledovanja podatkov vsem avtenticiranim uporabnikom sistema. Dodatne pravice za pregledovanje niso potrebne.



Slika 23: prikaz podatkov o poslovnem subjektu

4.7 Zaznamki

Bonitetni sistem beleži in hrani zaznamke vezane na naslednje izdelke in dogodke:

- Izdelano bonitetno poročilo
- Izdelani finančni podatki
- Izdelana analiza
- Vnos dodatnih poročil komentarjev, člankov in drugih priponk

Zaznamek za bonitetno poročilo je vedno vezen na poslovni subjekt, medtem ko so komentarji, članki in druge priponke lahko vezani na poslovni subjekt, panogo ali izmišljeno ime, ki ga določi administrator.

Modul Zaznamki – osnovna stran ponuja dva osnovna sklopa funkcionalnosti:

- vnos zaznamkov
- pregled zaznamkov

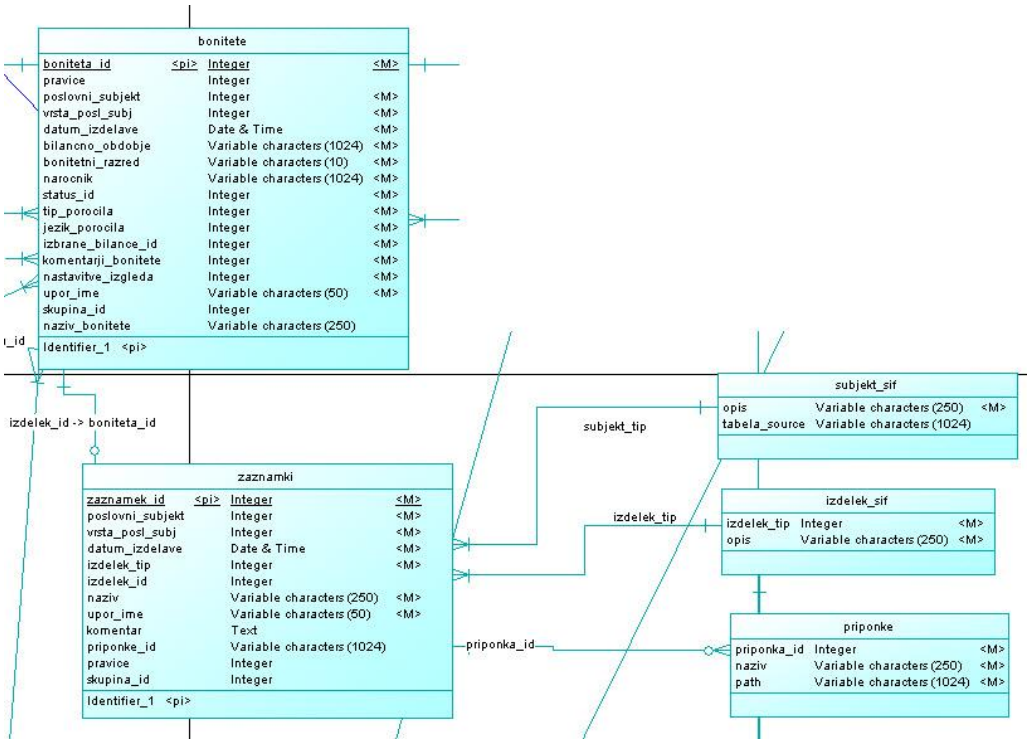
Do modula Zaznamki lahko uporabnik pride neposredno preko povezave v levem menuju aplikacije. Po kliku na povezavo se odpre osnovna stran, kjer se nahaja tabela zaznamkov ter polja za filtriranje zadetkov.

Arhitekturno gledano, modul zaznamki sestoji iz dveh strani, in sicer zaznamki.aspx ter zaznamki_izdelava.aspx. Prva stran vsebuje osnovni ter napredni iskalnik oz. filter zaznamkov s tabelo zaznamkov. Druga stran je namenjena bodisi vnosu novega ročnega zaznamka, bodisi detajlnemu pregledu že obstoječega, do katereg dostopamo preko prve strani.

4.7.1 Podatkovni model

Zaznamki se hranijo v tabeli ‘zaznamki’. Vsak posamezni zaznamek ima enoličen ključ ‘zaznamek_id’, tipa integer. Njegova vrednost se kreira avtomatsko ob vnosu novega zapisa v bazo. Kot prikazuje tudi (Slika 24) hranimo poleg že omenjenega ključe še nekatere druge podatke, kot so denimo uporabniško ime tistega, ki zaznamek kreira, datum izdelave, naziv itd.Tabela zaznamki je povezana s tabelami šifrantov: ‘subjekt_sif’, ‘izdelek_tip’. V prvi izmed omenjenih tabel so shranjene vrednosti, ki nam omogočijo ločevati med poslovnim subjekti, dejavnostmi ter lastnim skupina. Vrednosti iz šifranta ‘izdelek_tip’, ki nam povedo ali gre morda za zaznamek vezan na bonitetno poročilo, finančni podatek, naročilo. Vrednost v polju ‘izdelek_id’ je prazna (null) v primeru, da se zaznamek kreira ročno. V primeru, da se zaznamek kreira avtomatsko ob zaključitvi bonitete, se v to polje vnese id bonitete, v primeru zaključenega

finančnega podatka finančni - podatek id, naročila - naročilo id. Posamezen zaznamek lahko vsebuje tudi pripono, ali več teh. Za hranjenje priponke, se uporablja tabela ‘priponke’.

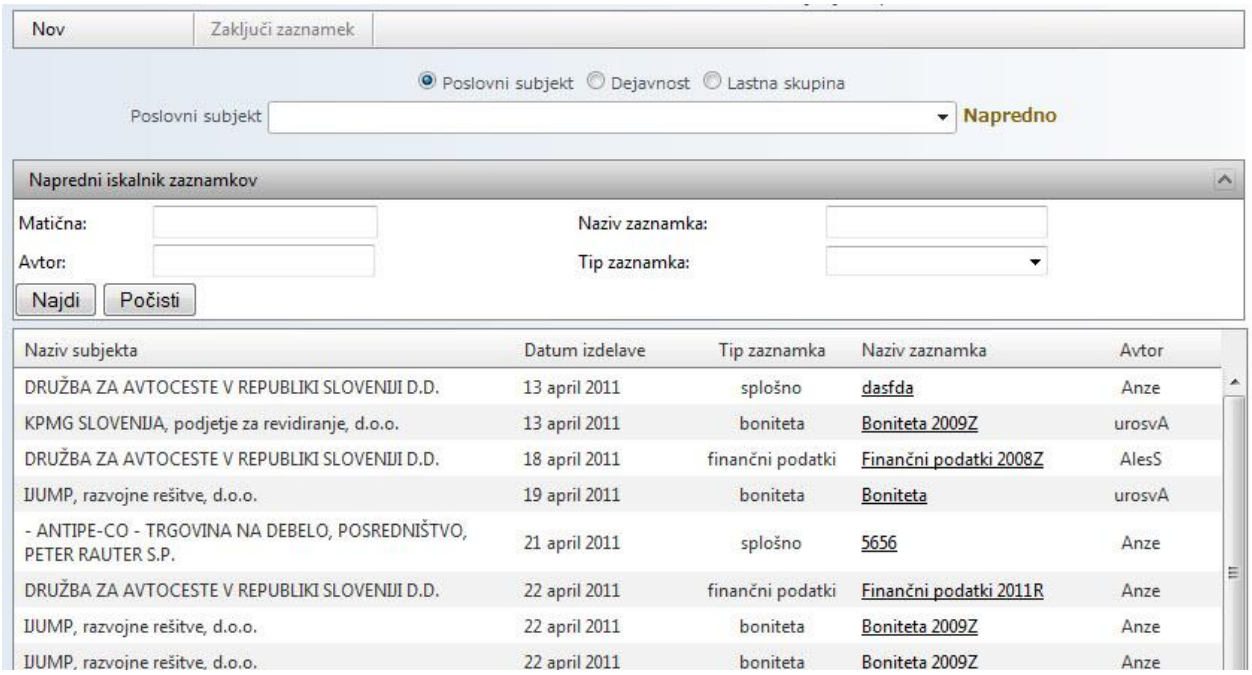


Slika 24: del podatkovnega modela baze, vezanega na zaznamke

4.7.2 Iskalnik zaznamkov

Osnovno iskanje omogoča, da zaznamke ločimo glede na polje ‘vrsta_posl_subj’, ki je tuji ključ na tabelo ‘subjekt_sif’. V aplikaciji imamo možnost takega grupiranja z izbirnim gumbom, ki se nahaja na vrhu kot prikazuje Slika 25. Ob preklaplanju vrednosti v izbirnem gumbu, se dinamično spreminja kombinirano polje pod njim. Ob preklapu se izvede klic Ajax, in osvežijo se samo posamezne kontrole in ne celotna stran.
Iskanje zaznamkov znotraj posamezne dejavnosti, poslovnega subjekta ter lastne skupine se lahko izvede preko v naprej ustvarjenih lastnih kontrol. Za vsako izmed opcij izbirnega gumba, smo ustvarili svojo kontrolo. Tako za iskanje po poslovnih subjektih uporabljamo eno, za iskanje po dejavnostih drugo in iskanje po lastni skupini tretjo. Vsaka izmed naštetih kontrol vsebuje preprosto kombinirano polje v katerega lahko vpisujemo iskani niz oz. naziv. Kombinirano polje nam že ob vpisovanju ponuja rezultate na izbiro. Tudi tu se vse dogaja asinhrono, tako da vsa zadeva deluje zelo odzivno. Drugi način je iskanje s pomočjo naprednega iskalnika. Napredni

iskalnik sestavlja množica standardnih komponent, kot so vnosna polja ter spustni meniji. Napredni iskalnik omogoča iskanje po več kriterijih.



Slika 25: tabela najdenih zaznamkov z iskalnikom (zaznamki.aspx)

Arhitektura strani zaznamki.aspx

1. lastna kontrola TabelaZaznamki.ascx
   Kontrolo v celoti ponazarja (Slika 25). Sestavlja jo več gnezdenih kontrol:
   - asp kontrola RadioButtonList za izbiro med tipom subjekta
   - lastne kontrole IskanjePoslovniSubjekt.asx, IskanjeDejavnost.asx ter IskanjeLastnaSkupina.ascx.
   - telerikova kontrola RadPanelBar, za prikazovanje in skrivanje naprednega iskalnika
   - telerikova kontrola RadMultiPage, za preklaplanjem med podstranmi
   - telerikova kontrola RagGrid, služi prikazu najdenih zadetkov
2. lastna kontrola ToolbarZaznamki.ascx
   Služi za navigacijo. Vsebuje gumb za kreiranje novega zaznamka ter gumb za zaključitev zaznamka, ki je ravno v ustvarjanju.
3. telerik kontrola RadAjaxLoadingPanel
   namenjena prikazu vrtečega se krogca ob iskanju zadetkov.

4.7.3 Pregled zaznamka

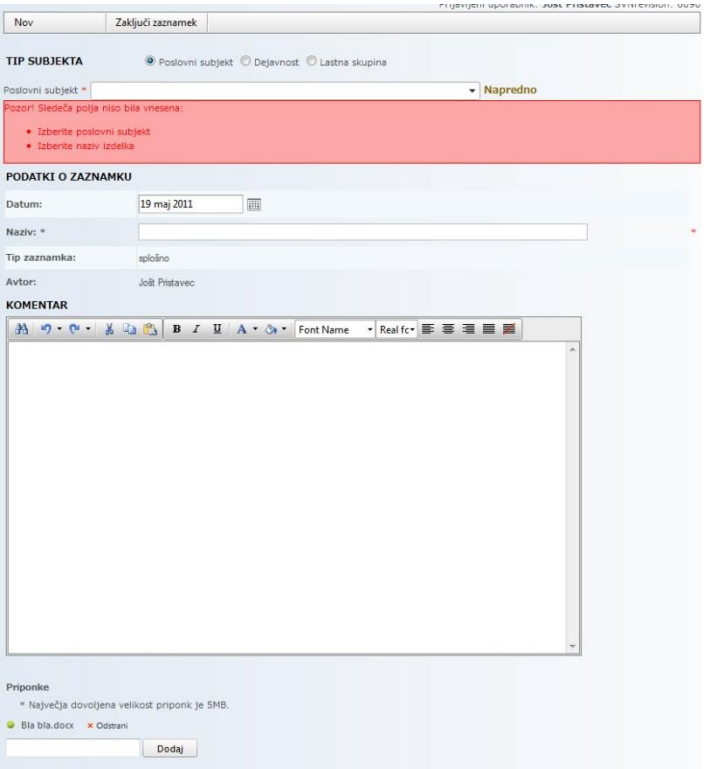
Ob izbiri posameznega zaznamka, nas aplikacija preusmeri na stran zaznamki_izdelava.aspx. Pri tem se s pomočjo parametrov prenese vrednost, ki je enoličen identifikator zaznamka. Stran na podlagi parametra ugotovi ali gre za pregled obstoječega zaznamka, ali gre za kreiranje novega. V primeru, da gre za pregled obstoječega, prebere vrednost iz parametra in naredi dostop do baze. Izvrši se poizvedba, v kateri se vrednost parametra poda kot argument. Vrednost prenešenega parametra predstavlja zapis v bazi z enako vrednostjo ključa (zaznamek_id v tabeli Zaznamki). Za logiko in vmesni nivo uporabljamo razred Zaznamek. V primeru branja podatkov iz baze se kreira nov objekt Zaznamek, in vrednosti se iz baze prenesejo vanj. Elementom na strani se priredijo vrednosti objekta Zaznamek. (velja seveda tudi obratna smer, kar bomo videli na primeru vnosa novega zaznamka v podatkovno bazo).

Pri pregledu zaznamka se prikažejo podatki kot so naziv zaznamka, tip subjekta, datum izdelave zaznamka, tip zaznamka, avtor zaznamka, komentar, morebitna povezava na izdelek ter podatki o subjektu.

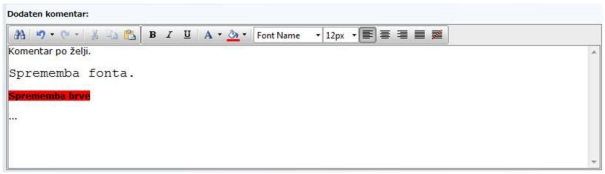
V kolikor gre za avtomatski zaznamek, se prikaže tudi povezava na izdelek, ob zaključitvi katerega se ja zaznamek kreiral. V primeru, da se je zaznamek kreiral ob dogodku zaključitve bonitete, katera je imela tudi izbrano naročilo, se poleg povezave na boniteto prikeže tudi povezava na izbrano naročilo ter priponka katere vsebuje ustvarjeno bonitetno poročilo v obliki dokumenta pdf.

4.7.4 Izdelava ročnega zaznamka

Stran za vnos novega zaznamka je ista kot je stran za pregled zaznamka (omenjeno v predhodnem poglavju) zaznamki_izdelava.aspx. Na strani se nahajajo forme za vnos podatkov, kot prikazuje (Slika 26).



Slika 26: izdelava ročnega zaznamka



Slika 27: kontrola za vnos komentarja

kontrola RadEditor

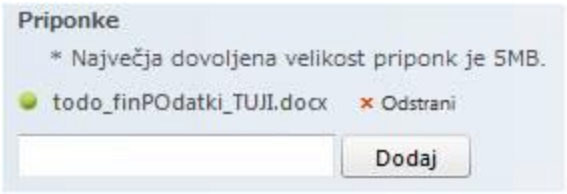
Za vnos komentarja je uporabljena telerik kontrola RadEditor [17](Slika 27), ki omogoča uporabniku oblikovanje teksta (barvanje, senčenje, uporaba krepkega besedila, številčenje, ustvarjanje prelomov, zamikov itd.). Vrednost zapisa komentarja se prebere kot besedilo html in se takšnega tudi shrani v bazo. Ob pregledu zaznamka, zna kontrola interpretirati prirejeno vrednost in jo ustrezno oblikovno prikazati uporabniku.

kontrola RadAsyncUpload

Za vnos priponk je uporabljena telerik kontrola RadAsyncUpload [18]. Kontrola ima vgrajen validator za opozarjanje uporabnika, da z pripetim dokumentom nekaj ni v redu. Preprosto, z

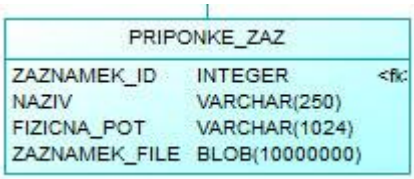
vpisom ustreznih atributov, je možno nastavljati denimo največjo dovoljeno velikost datoteke, dovoljene vrste dokumentov (npr. .pdf, .docx, .jpg itd). Ozadje delovanja kontrole RadAsyncUpload je sledeče: kontrola sprejme pripeti dokument, ter preveri, ali dokument ustreza pravilom (t.i. validacija). Validacija se lahko izvede bodisi na strežniku, bodisi na klientu.
V primeru, da je validacija uspešna, kontrola v direktoriju kjer se hranijo trenutni dokumenti kreira nov dokument in mu pri tem dodeli enoličen naziv. Ob vnosu zaznamka v podatkovno bazo, se od kontrole zahteva pripeti dokument. Kontrola iz direktorija, kjer so shranjeni trenutni dokumenti vrne pripeti dokument. V prvi fazi razvoja, se je namesto kontrole RadAsyncUpload uporabljala kontrola RadUpload. Slednja ni omogočala validacije na klientu, pripenjanje dokumantov ni bilo možno izvesti asinhroni temveč je bil potreben pravi 'postback', prav tako je bil 'postback' potreben za validiranje pripetih dokumentov na strežniški strani.

Postopek dodajanja nove priponka:
S klikom na gumb "Dodaj" se odpre pogovorno okno kjer se izbere željeno datoteko. Po potrditvi se izbrana datoteka prikaže. Ob datoteki je prikazano ali je bila uspešno dodana (zelen krogec). V času nalaganja je krogec oranžne barve. V primeru, da je prišlo do napake se prikaže krogec redeče barve. S klikom na gumb "Odstrani" selahko priponke odstrani.



Slika 28: vnos priponke

Ker je možno zaznamku pripeti več kot en dokument, se za hranjenje priponk v podatkovni bazi uporablja tabela priponke_zaz. Tabeli zaznamki in priponke_zaz sta povezani na način ena-mnogo. (Slika 24). Za hranjenje dokumentov se uporablja polje blob (binary large objet), kot prikazuje (Slika 29)



Slika 29: tabela priponk

Za iskanje poslovnega subjekta, dejavnosti oz. lastne skupine se pri izdelavi novega ročnega zaznamka uporabljajo lastne kontrole IskanjePoslovniSubjekt.ascx, IskanjeDejavnost.ascx ter IskanjeLastnaSkupina.ascx. Verjetno ni odveč povdariti, da gre za iste kontrole, kot jih najdemo na strani zaznamki.aspx. Srečamo se z t.i. 'ponovno uporabo komponent' kar nam omogoča dobra modularna arhitektura sistema.

4.7.5 Ustvarjanje avtomatskega zaznamka

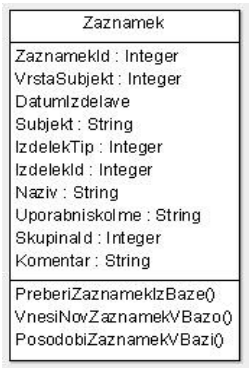
Bonitetni informacijski sistem omogoča, poleg že predstavljenega ročnega zaznamka, tudi vnos zaznamka v podatkovno bazo ob nastopu določenega dogodka.
Med takšne dogodke štejemo:

1. zaključitev bonitetnega poročila
2. zaključitev finančnega podatka
3. zaključitev analize

Ob nastopu dogodka, mora logika poskrbeti za ustvarjanje novega objekta Zaznamek, mu prirediti potrebne vrednosti ter poklicati metodo za vnos zaznamka v podatkovno bazo. Vrstni red je tu ravno obraten, tistemu, ki smo ga omenili pri pregledu zaznamka.

Razred Zaznamek

Razred Zaznamek predstavlja del logičnega nivoja aplikacije. Primerki razreda predstavljajo objekte. Posamezen objekt predstavlja posamezen zapis v tabeli zaznamki v podatkovni bazi. Razred ima lastnosti, in si jih lahko predstavljamo kot kolone tabele zaznamki v podatkovni bazi. Poleg lastnosti, so v razredu deklarirane tudi metode za zapisovanje, branje in posodabljanje zaznamka. Razred ponazarja (Slika 30), podatkovni nivo pa (Slika 24: del podatkovnega modela baze, vezanega na zaznamke).



Slika 30: razred Zaznamek

Vnos zapisa v podatkovno bazo poteka preko metode VnesiNovZaznamekVBazo. Metoda izvrši klic bazne procedure za vnos novega zaznamka. Vrednosti vhodnih parametrov procedure se nastavijo iz vrednosti lastnosti objekta Zaznamek. Posredujejo se kot vhodni parametri procedure. Procedura poskrbi za vnos novega zapisa v podatkovno bazo, pri tem pa kot izhodni parameter vrne vrednost ključa, kateri se avtomatsko generira ob vnosu. V kolikor vnos ni uspešen vrne vrednost -1.

4.8 Podatkovna baza

4.8.1 Povezava aplikacije s podatkovno bazo

Za povezavo s podatkovno bazo IBM DB2 verzije 8.1, aplikacija uporablja programski paket 'Data DB2 .NET Provider'. Paket razširja funkcionalnost DB2 za vmesnik ADO.NET. Zagotavlja visoko zmogljiv ter varen dostop do podatkov DB2. Sestoji iz minimalne plasti med bazo podatkov in kodo. To razširja funkcionalnost, brez žrtvovanja zmogljivosti.
DB2. NET Data Provider zagotavlja funkcionalnost za povezovanje z zbirko podatkov, izvajanje ukazov, in pridobivanje rezultatov. Ti rezultati se lahko neposredno obdelujejo oziroma so shranjeni v ADO.NET DataSet za nadaljnjo obdelavo, medtem ko so v stanju nepovezanosti z bazo. [19]
Paket sestavljajo štirje temeljni objekti:

DB2Connection	Vzpostavi povezavo z določeno zbirko podatkov ter začenja transakcijo
DB2Command	Služi izvrševanju ukazov v bazi podatkov
DB2DataReader	Bere tok podatkov
DB2DataAdapter	Napolni DataSet in rešuje posodobitve z virom podatkov.

Tabela 1: objekti za delo z bazo DB2

Za delo nad bazo nam služi za to ustvarjen poseben objekt, katerega definira razred 'dbUtilityClass'. Objekt ustvari povezavo z podatkovno bazo. Potrebne parametre za ustvarjanje povezave prebere iz konfiguracijske datoteke web.config. (Slika 31)

```
<add name="FINPRODConnectionString"
     connectionString="Database=FINPROD;User ID=xxx;Password=xxx;
                       Server=192.168.2.85:50001;Persist Security Info=True"
     providerName="IBM.Data.DB2" />
```

Slika 31: konfiguracijska datoteka s parametri za povezavo s podatkovo bazo

Razred vsebuje metode za izvajanje tako preprostih poizvedb kakor tudi nekoliko bolj naprednih, denimo z uporabo parametrov, klicom procedure itd. Pri branju vrednosti iz baze se največkrat uporabi shranjevanje v enega izmed naštetih objektov: [19]

DataSet	Objekt je ustvarjen za uporabo 'brez aktivne povezave z bazo' in lahko vsebuje množico objektov DataTable in relacij med njimi.
DataTable	Množica podatkov, ki vsebuje enega ali več objektov DataColumn ter ko se napolni tudi več objektov DataRow
DataRow	Množica vrednosti podobnih vrstici v tabeli v bazi
DataColumn	Objekt vsebuje naziv in tip kolone

Tabela 2: podatkovne strukture

4.8.2 Struktura podatkovne baze

Podatkovna baza sestoji iz več tabel, odvisnostmi med njimi, baznih procedur in prožilcev kateri so sistematično razdeljeni v sheme.

Naziv sheme	Tip podatkov
AJPES	Podatki splošnega značaja o poslovnih subjektih in dejavnostih, katere zagotavlja Agencija Republike Slovenije za javnopravne evidence in storitve. Bonitetni informacijski sistem teh podatkov ne more urejati, lahko jih le bere.
BONSIS	Podatki, ki so v neposredni korelaciji z Bonitetnim informacijskim sistemom. Primer: bilance stanj, bonitete, finančni podatki, kazalniki, ocene, naročila, zaznamki, izdelki, lokalizacije…
VIRI	Tabele šifrantov.
POSLOVNI PARTNERJI	Podatki o poslovnih subjektih, kateri nimajo sedeža v Republiki Sloveniji

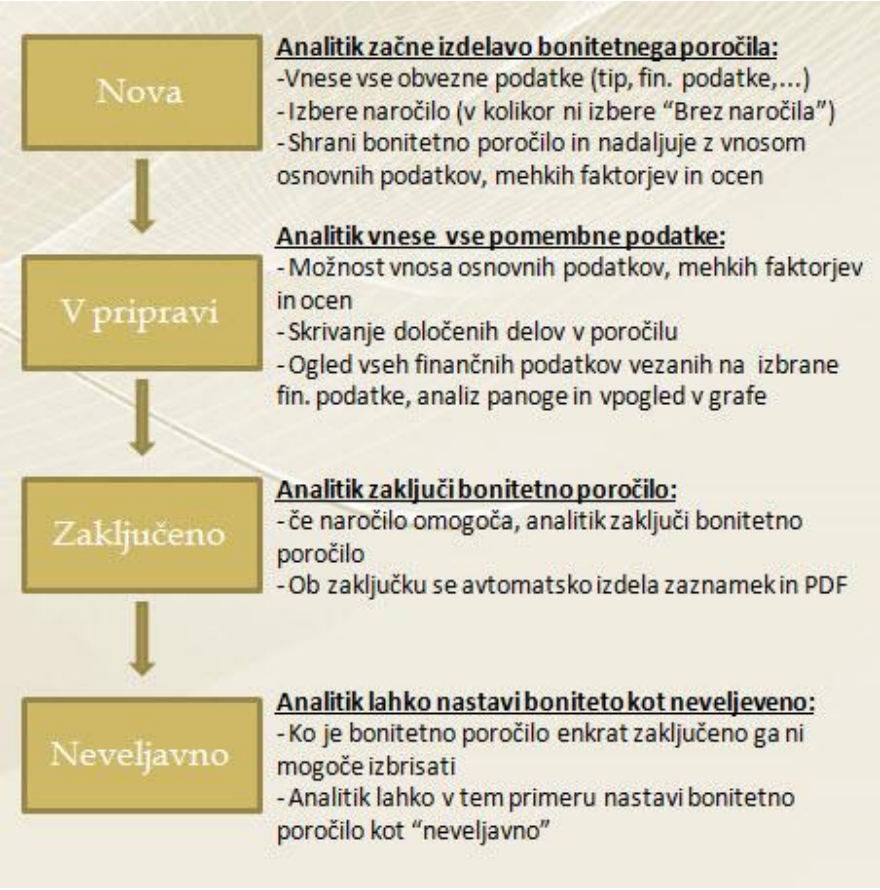
Tabela 3: sheme v podatkovni bazi

4.9 Bonitete

Modul bonitete predstavlja, z vidika uporabnika, osrednji modul sistema. Funkcionalnost modula predstavljajo 4 sklopi:

1. iskalnik bonitet
2. izdelava bonitete
3. pregled/urejanje bonitet
4. izvoz/tiskanje bonitete

Boniteta se vedno nanaša na določen poslovni subjekt. Modul bonitete sestavljata dve glavni strani in sicer bonitete.aspx ter bonitete_izdelava.aspx. Prva stran služi prikazu tabele bonitete z naprednim ter osnovnim iskalnikom, druga pa kot že ime pove izdelavi nove bonitete, hrati pa tudi podrobnemu pregledu in urejanju že obstoječe. Na vrhu obeh strani najdemo orodno vrstico za delo z bonitetami. Orodna vrstca vsebuje gumbe, kateri prožijo akcije kot so: izdelava nove, shranjevanje, zaključevanje, brisanje, kopiranje, tiskanje, oblikovanje bonitete. Možne statuse bonitete prikazuje Slika 32.



Slika 32: možni statusi bonitete

Za pregledovanje, izdelavo, urejanje bonitet mora imeti uporabnik sistema omogočeno primerno pravico. Pravice vezane na modul bonitet so sledeče:

Naziv pravice	Pravica omogoča…
bon_pregl_zaklj	pregledovanje zaključenih bonitet
bon_pregl_pripr	pregledovanje bonitet v pripravi
bon_vnos	izdelovanje novih bonitet
bon_kopiranje	kopiranje bonitet
bon_sprem_statusa	spreminjanje statusa bonitete
bon_sprem_label	spreminjanje label bonitete
bon_predloge_izbor	izbiranje predlog za izdelavo bonitete
bon_predloge_urejanje	urejanje predlog bonitet

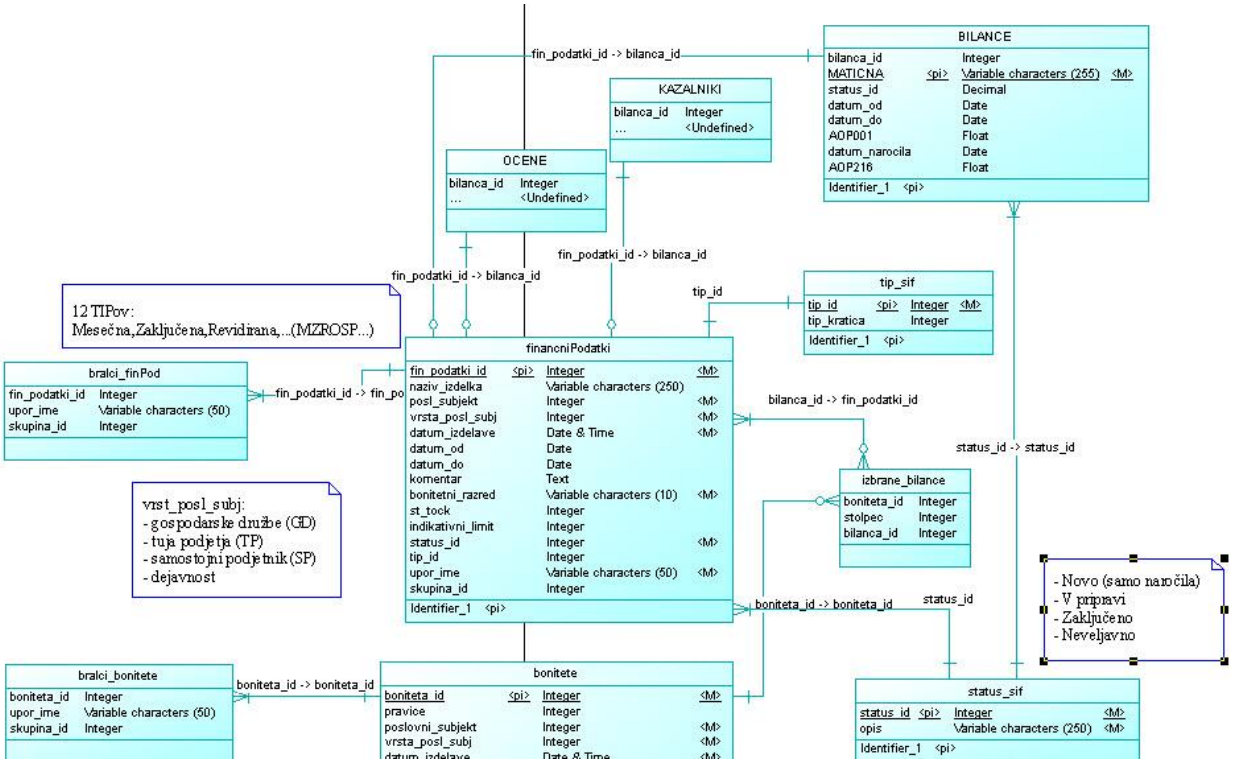
Tabela 4: uporabniške pravice pri bonitetah

Pravici namenjeni pregledovanju sta sledeči: pravica pregledovanja zaključenih bonitet, pravica pregledovanja bonitet v pripravi

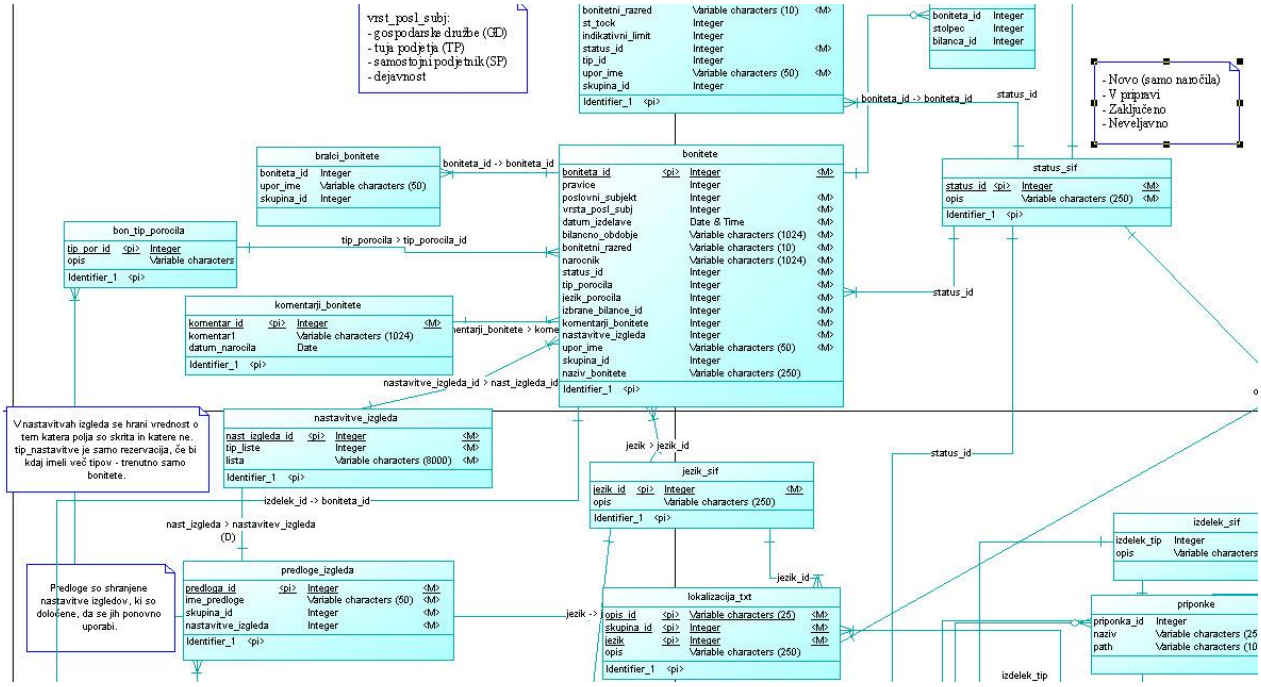
Do strani bonitet lahko dostopajo uporabniki, katerim je dodeljena pravica pregledovanja. Pri pravicah za pregledovanje bonitete ločimo dve pravici in sicer t.i. pravica do širšega pregledovanja ter pravica ožjega pregledovanja. V primeru, da ima uporabnik vključeno pravico ožjega pregledovanja, širšega pa ne, mu sistem dovoli pregledovati le tiste bonitete v katerih je eksplicitno naveden kot bralec bonitete. V kolikor pa ima uporabnik vključeno možnost širšega pregledovanja bonitet, vidi prav vse bonitete v sistemu. Med takšne uporabnike sistema štejemo analitike ter administratorje. Ostale pravice so predstavljene v tabeli zgoraj (Tabela 4).

4.9.1 Podatkovni model

Podatki o posamezni boniteti se hranijo v podatkovni bazi. Tabela bonitete predstavlja pri tem osrednjo vlog. Bonitete so preko vmesne tabele imenovane izbrane_bilance povezane s tabelo financniPodatki. Preko tabele financniPodatki lahko pridemo do podatkov o posameznih bilancah kazalnikih ter ocenah. Povezave med tabelami ocene, bilance, kazalniki so s tabelo financniPodatki povezani s povezavo ena – nič. Za vsako bilanco, oceno ter kazalnik obstaja natančno en primerek finančnega podatka. Tako boniteta kakor tudi finančen podatek sta povezana s tabelo šifranta, kateri določa kakšen status imata.Poleg statusa, ima boniteta še povezavo na šifrant jezikov ter tip bonitetnega poročila. Ena boniteta ima lahko izbranih nič ali več bralcev (tabela bralci) ter tudi nič ali več finančnih podatkov. En finančen podatek (ter s tem ena bilanca, kazalnik ter ocene) so lahko vezani na več različnih bonitet. Zato tudi vmesna tabela izbrane_bilance. Ker je boniteta vedno vezana na določen poslovni subjekt, vsebuje tudi atribut poslovni_subjekt, kjer se hrani matična številka poslovnega subjekta. Enako velja za finančen podatek, za ta namen imamo tu atribut posl_subjekt. Pri obeh primerih gre za atribut, ki je ob vnosu obvezen. Atribut boniteta_id se ob vnosu nove bonitete samodejno nastavi in primerno poveča ter zato ni potrebno skrbeti aplikaciji.



Slika 33: podatkovni model - Finančni podatki



Slika 34: podatkovni model - Bonitete

4.9.2 Izdelava bonitete

Za izdelavo nove bonitete nam služi stran bonitete_izdelava.aspx. Ob kreiranju nove bonitete, lahko uporabnik dostopa le do zavihka 'Status'. Zavihek vsebuje vnosne forme. Nekateri podatki so ob vnosu nujni, nekateri ne. V primeru, da analitik ne izpolni podatka, ki je nujen, ga stran na to opozori. Opozori ga na način, da se ob polju, kjer manjka podatek izpiše znak * rdeče barve, poleg tega se mu na vrhu strani pokaže opis, katero polje je ostalo neiuzpolnjeno. Za ta namen se uporabljajo kontrole RequiredFieldValidator v povezavi s kontrolo ValidationSummary.

Ob izdelavi nove bonitete, analitik določi jezik bonitete in s tem tudi jezik v katerem se bodo izpisali podatki v bonitetnem poročilu ob izvozu podatkov. Prednastavljeni jezik je Slovenščina. S pomočjo spustnega menuja ga je možno spremenit v Angleščino. V trenutni verziji aplikacije, sta na voljo omenjena jezika, možna je kasnejša nadgradnja. Za nastavitev avtorja bonitete poskrbi informacijski sistem sam. Prebere ga iz seje. Prav tako sistem sam poskrbi za prednastavljanje vrednosti datuma izdelave bonitete. Datum je možno tudi spremeniti. Pri ustvarjanju nove bonitete, je le tej možno dodati tudi poljubno število priponk. Za nalaganje priponk, se uporablja, enako kot pri zaznamkih, kontrola RadAsyncUpload. Priponke se hranijo v tabeli priponk ter se v podatkovno bazo vpišejo kot BLOB.

Bonitetni informacijski sistem omogoča izdelavo različnih vrst bonitet:

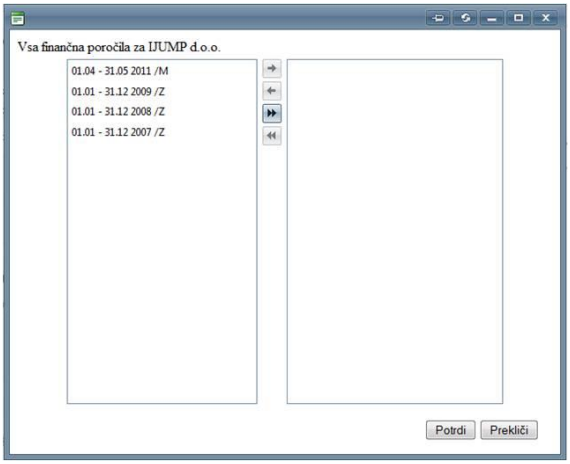
Tip bonitete	Opis
A- BON	Boniteta, kjer se avtomatsko izberejo bilance za izbrni poslovni subjekt. Podatki bilanc so enaki tistim pri ročni boniteti – kratka bilanca.
Ročna boniteta – kratka bilanca	Boniteta z ročno izbranimi bilancami za izbrani poslovni subjekt. V boniteti in bonitetnem poročilu se prikažejo le osnovni podatki bilanc.
Ročna boniteta – dolga bilanca	Boniteta z ročno izbranimi bilancami za izbrani poslovni subjekt. V boniteti in bonitetnem poročilu se prikažejo razširjeni podatki bilanc.

Tabela 5: tipi bonitet

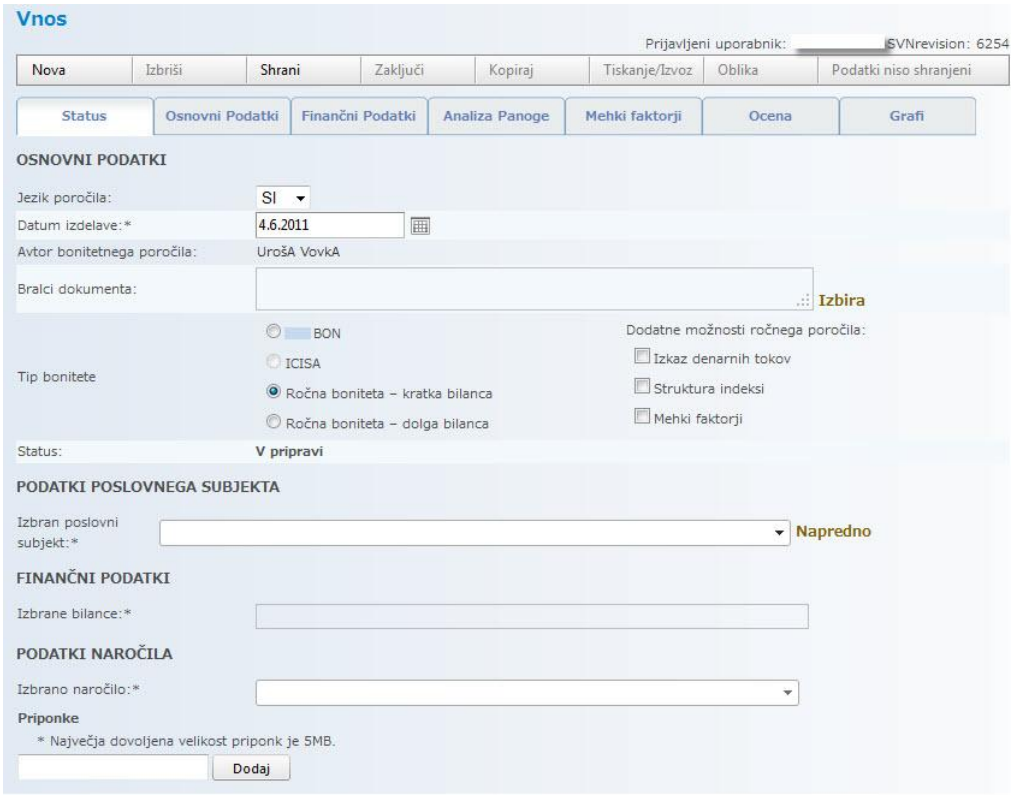
Za izbiro poslovnega subjekta se uporablja kontrola opisana v poglavju 4.6.3.

Ker naj bi bila funkcionalnost v realnosti takšna, da bo lahko določeno boniteto pregledovalo več uporabnikov in ne le tisti, ki jo je kreiral se ob ustvarjanju nove, analitik odloči in doda bralce bonitet. Bralec je lahko uporabnik, kateri je že v podatkovnio bazi naveden kot uporabnik s določenimi pravicami, lahko pa je to uporabnik, katerega ni v podatkovni bazi, je pa zato v sistemu LDAP.

Po tem, ko analitik izbere poslovni subjekt, se omogoči kontrola za izbiro bilanc.
V primeru, da gre za izdelavo ročne bonitete, ima analitk na voljo povezavo do iskalnika bilanc (Slika 35), ki se odpre v obliki pojavnega okna. V levem stolpcu se prikažejo vse bilance poslovega subjekta. Analitik izbere željeno bilanco oz. več teh tako da jih označi in s puščico prenese v desni stopec. Možno je izbrati do 3 bilance. Možna je tudi izdeleva bonitete brez izbranih bilanc. V takšnem primeru, se ob pregledovanju bonitete onemogočita zavihka 'Finančni podatki' ter 'Grafi'.V primeru izbire tipa bonitete A-BON, aplikacija sama izvrši iskanje primernih bilanc s posebnim algoritmom. Algoritem se uporabi tudi, kadar analitik izbere samo eno ali dve bilanci. V tem primeru, aplikacija izbrani bilanci oz. izbranima bilancama z algoritmom doda primerne bilance. To pa zato, ker se pri določenih poljih, tabelah, grafih potrebuje določeno število bilanc in mora biti vedno enako.



Slika 35: kontrola za ročno izbiro bilanc



Slika 36: stran za izdelavo bonitete

Poznamo več tipov bilanc, podatke o tipu hrani šifrant v bazi ‘sifrant_tip’. Šifrant tipov bilanc prikazuje spodnja tabela:

ID	Oznaka	Naziv
1	M	Mesečna
2	Z	Zaključna
3	R	Revidirana
4	O	Ocena
5	S	Simulacija
6	P	Plan
7	KM	Konsolidirana – mesečna
8	KZ	Konsolidirana – zaključna
9	KR	Konsolidirana – revidirana
10	KO	Konsolidirana – ocena
11	KS	Konsolidirana – simulacija
12	KP	Konsolidirana – plan

Tabela 6: šifrant tipov bilanc

Algoritem iskanja ustreznih bilanc

Algoritem za iskanje primernih bilanc implementirati 2 funkciji (dodani sta v prilogi):

- PoisciCelotnoSekvencoBilanc()
- PoisciCelotnoSekvencoBilanc_R_Z()

PoisciCelotnoSekvencoBilanc

Funkcija sprejme že izbrano listo bilanc. V kolikor lista že vsebuje dovoljšno število elementov, potem ne naredi ničesar, le vrne izbrane bilance. Število elementov za vračilo je podano kot argument funkciji. V nasprotnem primeru izbranim bilancam poišče dodatne bilance in jih doda že izbranim ter vrne tako dobljeno listo bilanc. Logika za iskanje primernih dodatnih bilanc:

1. Razvrščanje izbranih bilanc glede na datum bilance.
2. Ugotavljanje tipa najmlajše izbrane bilance.
3. Glede na dobljeni tip najmlajše izbrane bilance, se naredi branje dodatnih bilanc. V primeru, da je najmlajše izbrana bilanca konsolidirana (glej Tabela 6), potem se poišče le konsolidirane – zaključne (KZ) ter konsolidirane – revidirane (KR) bilance. V nasprotnem primeru se poišče le zaključne (Z) ter revidirane (R). Poišče se bilance, katere so stareše od najstarejše izbrane bilance. Najdene bilance se doda v listo dodatnih bilanc.
4. V primeru, da za isto leto obstajata tako zaključna (Z) kot tudi revidirana (R) oz. tako konsolidirana – zaključna (KZ) kot konsolidirana – revidirana (KR), se v listo dodatnih bilanc doda le revidirano (R) oz. konsolidirano – revidirano (KR).
5. Združevanje izbranih bilance z dodatnimi ter razvrščanje glede na datum bilanc.
6. Vračanje le prvih toliko elementov kot podaja argument število elementov.

Primer uporabe funkcije:

1. Izbrani tip bonitete je Ročna bonitete (dolga ali kratka)
    a. Analitik izbere 1 bilanco
       Algoritem poskrbi, da se izbrani bilanci poiščeta še 2 dodatni
    b. Analitik izbere 2 bilanci
       Algoritem poskrbi, da se izbrani bilanci poiščeta še 2 dodatni
2. Prikazovanje podatkov za zadnjih 5 obdobij
   Algoritem poleg izbranim bilancam doda potrebno število dodatnih bilanc, da zagotovi podatke za zadnjih 5 obdobij (zavihek Osnovni podatki, podatki o Poslovanju, Številu zaposlenih, Kapitalu ter Izozu). Algoritem se v tem primeru vrši vedno, saj je maksimalno število izbranih bilanc omejeno na 3.

PoisciCelotnoSekvencoBilanc_R_Z()

Logika za iskanje primernih bilanc je zelo podobna tisti v funkciji PoisciCelotnoSekvencoBilanc(), le da funkcija PoisciCelotnoSekvencoBilanc_R_Z() ne prejme izbranih bilanc kot argument. Izpusti se koraka 1,2 in 5, v tretjem koraku se išče le revidirane (R) ter zaključne (Z). Koraka 4 in 6 ostajata enaka.

Primer uporabe funkcije:

1. Izbrani tip bonitete je A-BON
   Analitik nima na voljo možnosti izbire bilanc, zato je avtomatsko polnjenje prepuščeno v celoti bonitetnemu informacijskemu sistemu. Število elementov, ki jih v tem primeru vrača funkcija je nastavljeno na 3.
2. Polnjenje podatkov vezanih na zavihek Analiza panoge. Tudi tu, analitik nimam možnosti izbiranja kateri podatki se bodo prikazovali. Vse naredi avtomatika aplikacije. Število vrnjenih elementov je nastavljeno na 2.

Po vseh naštetih in opravljenih korakih se s klikom na gumb Shrani, v orodni vrstici na vrhu strani, naredi zapis v podatkovno bazo. Sistem najprej preveri ali gre za prvi vnos oz. t.i. INSERT ali morda za posodabljanje oz. t.i. UPDATE in temu primerno izvrši primerno akcijo. Za zapis in posodabljanje podatkov v podatkovni bazi se uporablja bazna procedura. Klic bazne procedure izvrši statična metoda DB2KlicBazneProcedure razreda dbUtilityClass. Argument funkcije so vhodni parametri, katerim priredimo vrednosti. Primeri nastavljanja vhodnih parametrov so, skupaj z metodo za klic bazne procedure ter bazno proceduro za vnos bonitete v podatkovno bazo, prikazani v poglavju Dodatek na koncu diplomskega dela.

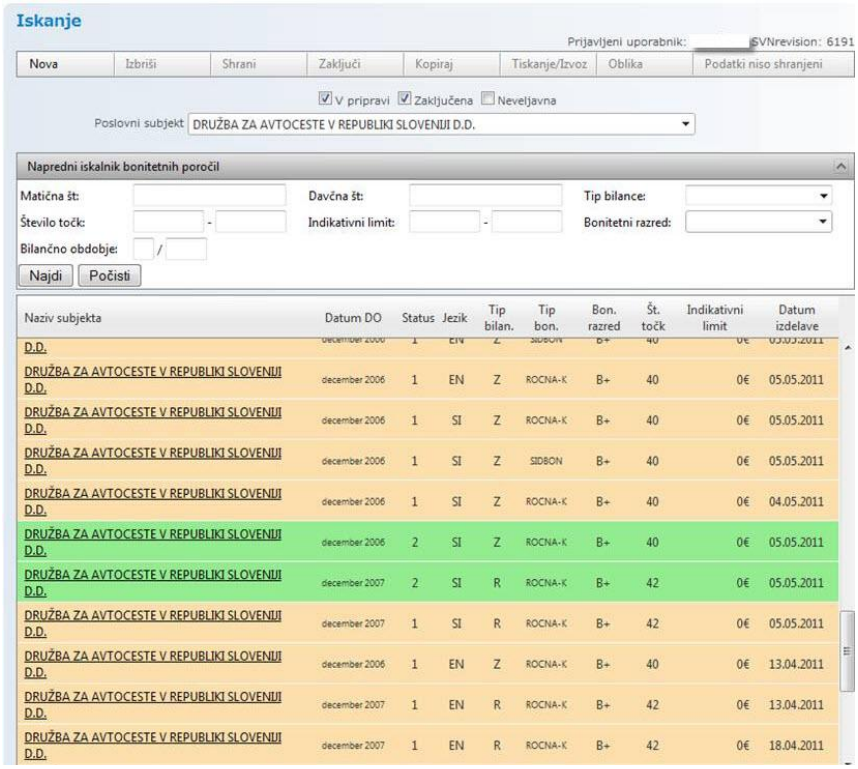
4.9.3 Iskalnik bonitet

Iskalnik bonitet je po izgledu in funkcionalnosti zelo podoben tistemu pri zaznamkih. Namen celotnega bonitetnega informacijskega sistema je, da izgleda enotno ter analitiku kar se da intuitivno. Tudi tu imamo razlikujemo med enostavnim in naprednim iskanje. Osnovno iskanje pomeni iskanje poslovnega subjekta po njegovem nazivu. V vednost bralca, je potrebno še enkrat povdariti, da je boniteta vedno vezana na določen poslovni subjekt.
Iskalnik poslovnega subjekta je isti, kot tisti pri zaznamkih. Gre za svojo lastno kontrolo, ki se uporablja na večih mestih po celotni aplikaciji in tako je tudi tu. Poleg omenjene kontrole spada pod osnovno iskanje tudi filter, kateri razlikuje med statusi bonitet. Možni statusi bonitete so:

- zaključena
- v pripravi
- neveljavna

Filter je implementiran s preprostimi potrditvenimi polji. Možno je prikazovati različne kombinacije statusov. Ob prvotnem dostopu do strani bonitet, sta obkljukana statusa v potrditvenih poljih v pripravi ter zaključena, poslovni subjekt ni izbran. Napredni iskalnik se nahaja znotraj telerikove kontrole RadPanelBar, in je prednastavljeno zaprt. Analitk ga odpre oz. raztegne s preprostim klikom nanj. Pokažejo se polja, po katerih je možno iskati bonitete. Parametri po katerih je možno izvesti napredno iskanje so sledeči:

- Matična številka poslovnega subjekta
- Davčna številka poslovnega subjekta
- Tip bilance
- Število točk
- Indikativni limit
- Bilančno obdobje
- Bonitetni razred



Slika 37: iskalnik bonitet

Rezultati iskanja bonitet se zlistajo v tabeli, katera je implementirana s telerikovo kontrolo RadGrid [20]. Uporabljena je tudi barvna indikacija, katera nam omogoča razlikovati med bonitetami z različnimi statusi. Tako se denimo boniteta, ki je zaključena obarva zeleno, tista v pripravi oranžno ter neveljavna sivo. S klikom na izbrano boniteto se naredi vpogled v boniteto. Iskanje in filtriranje se naredi na strežniku, nikoli na klientu, s pomočjo klica ajax-a. Med izvajanjem nam indikacijo, da se nekaj dogaja v sistemu omogoča uporaba telerikov RadAjaxLoadingPanel [21] ter vrteči se krogec.

4.9.4 Pregled in urejanje bonitete

Modul bonitete je sestavljen iz 7 podstrani:

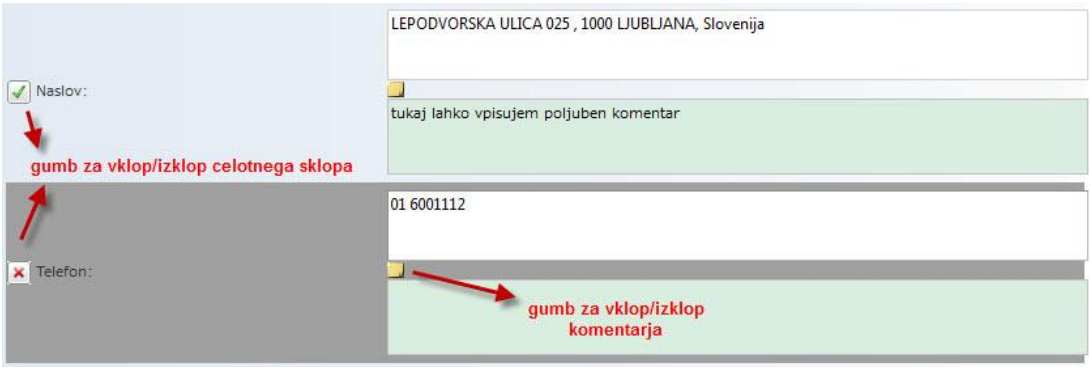
1. Status
2. Osnovni podatki
3. Finančni podatki
4. Analiza panoge
5. Mehki faktorji
6. Ocena
7. Grafi

Status

Podstran status vsebuje podatke, katere se vnese ob ustvarjanju nove bonitete in jih prikazuje Slika 36. Potrebno je omeniti, da podatkov iz te podstrani ni možno več spreminjati, ko so enkrat shranjeni v podatkovno bazo.

Osnovni podatki

Podstran je namenjena prikazovanju podatkov vezanih na izbrani poslovni subjekt. Določene podatke te podstrani lahko analitik poljubno spreminja, briše, vnaša in shranjuje v podatkovno bazo. Celoten sklop osnovnih podatkov je preobširen, da bi bil lahko predstavljen v diplomskem delu. Naj pa omenim, da ima analitik možnost t.i. izklapljanja in vklapljanja posameznih polj. To pomeni, da si lahko onemogoči oz. nastavi določen podatek, da se mu ne prikazuje tako na strani, kot potem tudi na poročilu. Podatki so razdeljeni v sklope. Vsak sklop vsebuje gumb za vklop/izklop celotnega sklopa, labelo sklopa, vnosno polje vsebine, vnosno polje komentarja ter gumb za vklop/izklop komentarja. Analitik ima tudi možnost spreminjanja vsebine labele.

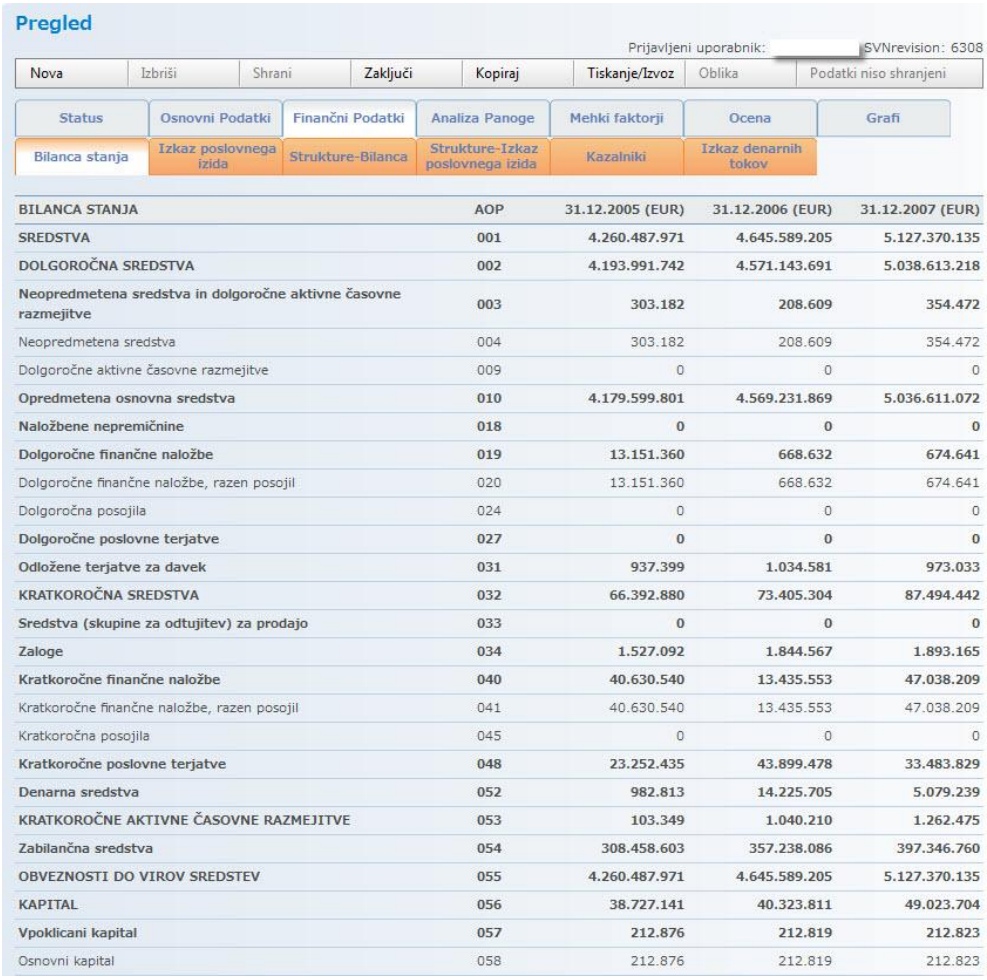


Slika 38: kontrola za vklapljanje in izklapljanje posameznega sklopa

Finančni podatki

Na podstrani finančni podatki so prikazani vsi podatki iz izbranih bilanc ob ustvarjanju bonitete. Podatki so namenjeni pregledovanju, urejanje ni možno, saj gre za že zaključene ali pa revidirane bilance s strani AJPESa. Podstran se zaradi velike količine podatkov nadalje deli na podzavihke:

- Bilanca stanja
- Izkaz poslovnega izida
- Strukture-Bilanca
- Strukture-Izkaz poslovnega izida
- Kazalniki
- Izkaz denarnih tokov

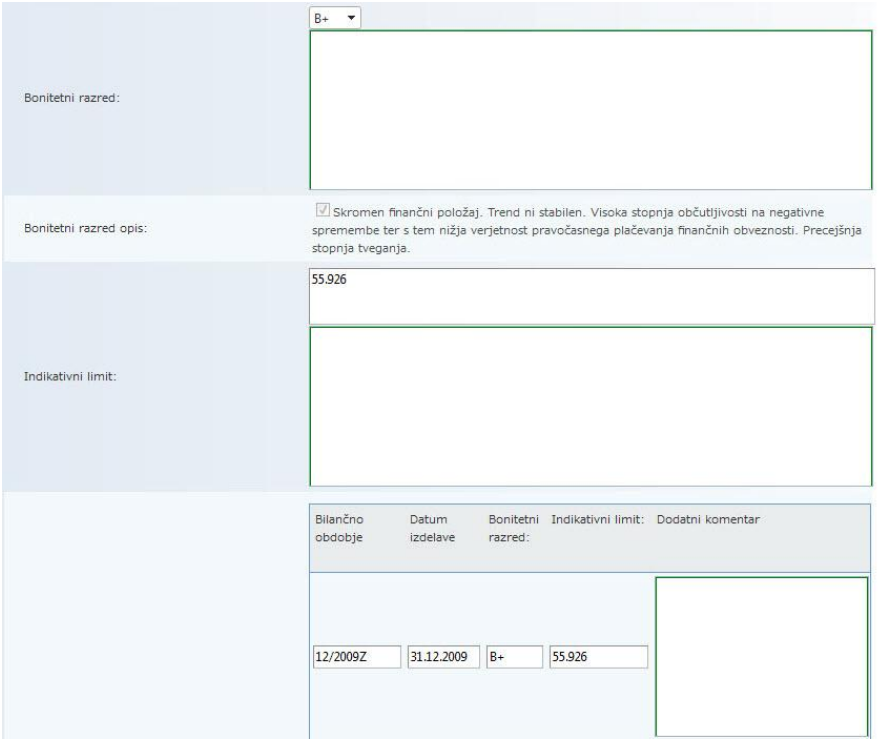


BILANCA STANJA	AOP	31.12.2005 (EUR)	31.12.2006 (EUR)	31.12.2007 (EUR)
SREDSTVA	**001**	**4.260.487.971**	**4.645.589.205**	**5.127.370.135**
DOLGOROČNA SREDSTVA	**002**	**4.193.991.742**	**4.571.143.691**	**5.038.613.218**
Neopredmetena sredstva in dolgoročne aktivne časovne razmejitve	**003**	**303.182**	**208.609**	**354.472**
Neopredmetena sredstva	004	303.182	208.609	354.472
Dolgoročne aktivne časovne razmejitve	009	0	0	0
Opredmetena osnovna sredstva	**010**	**4.179.599.801**	**4.569.231.869**	**5.036.611.072**
Naložbene nepremičnine	**018**	**0**	**0**	**0**
Dolgoročne finančne naložbe	**019**	**13.151.360**	**668.632**	**674.641**
Dolgoročne finančne naložbe, razen posojil	020	13.151.360	668.632	674.641
Dolgoročna posojila	024	0	0	0
Dolgoročne poslovne terjatve	**027**	**0**	**0**	**0**
Odložene terjatve za davek	**031**	**937.399**	**1.034.581**	**973.033**
KRATKOROČNA SREDSTVA	**032**	**66.392.880**	**73.405.304**	**87.494.442**
Sredstva (skupine za odtujitev) za prodajo	**033**	**0**	**0**	**0**
Zaloge	**034**	**1.527.092**	**1.844.567**	**1.893.165**
Kratkoročne finančne naložbe	**040**	**40.630.540**	**13.435.553**	**47.038.209**
Kratkoročne finančne naložbe, razen posojil	041	40.630.540	13.435.553	47.038.209
Kratkoročna posojila	045	0	0	0
Kratkoročne poslovne terjatve	**048**	**23.252.435**	**43.899.478**	**33.483.829**
Denarna sredstva	**052**	**982.813**	**14.225.705**	**5.079.239**
KRATKOROČNE AKTIVNE ČASOVNE RAZMEJITVE	**053**	**103.349**	**1.040.210**	**1.262.475**
Zabilančna sredstva	**054**	**308.458.603**	**357.238.086**	**397.346.760**
OBVEZNOSTI DO VIROV SREDSTEV	**055**	**4.260.487.971**	**4.645.589.205**	**5.127.370.135**
KAPITAL	**056**	**38.727.141**	**40.323.811**	**49.023.704**
Vpoklicani kapital	**057**	**212.876**	**212.819**	**212.823**
Osnovni kapital	058	212.876	212.819	212.823

Slika 39: postran finančni podatki

Ocena

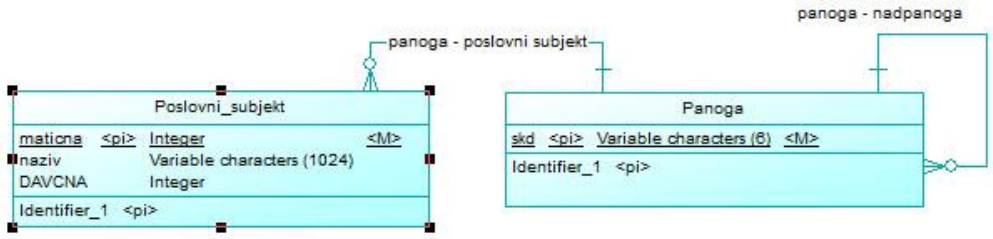
Glavna funkcionalnost podstrani je vnos ali posodabljanje bonitetne ocene, ki jo lahko analitik dodeli izbranemu podjetju ter določitev indikativnega limita podjetja. Možna je tudi vpogled ter sprememba ocen ter indikativnih limitov za vse izbrane bilance v boniteti, ne samo zadnje.



Slika 40: podstran ocena

4.9.4.1 Analiza panoge

Vsak poslovni subjekt zapisan v ajpesovi bazi vsebuje tudi informacijo o panogi v katero je uvrščen. Podatek o tem se nahaja v atributu ‘glavna_dejavnost_skd’. Gre za šestmestni niz znakov. Panoge se združujejo v nadpanogo, nadpanoge pa v gospodarstvo. Prikaz razmerja, med poslovnimi subjekti in panogo ter grupiranja panoge v nadpanogo, ponazarja **Error! Reference source not found.**. Tako kot za poslovni subjekt, obstajajo finančni podatki (bilance stanja, kazalniki, ocene) tudi za posamezno panogo, nadpanogo ter gospodarstvo. Gre za podatke, ki se v podatkovno bazo vnašajo avtomatsko s pomočjo prožilcev. Tako denimo vrednost o prihodkih za izbrano panogo, pomeni seštevek prihodkov vseh podjetji v tem letu, vrednost prihodkov nadpanoge pa seštevek prihodkov panog.



Slika 41: razmerje poslovni subjekt – panoga - nadpanoga

Za prikaz podatkov bilanc, kazalnikov ter ocen za panogo v katero poslovni subjekt spada, služi podstran analiza panoge. Podatki v tem sklopu se pridobijo avtomatsko, glede na izbran poslovni subjekt ob izdelavi bonitete. Za ta namen je uporabljena metoda PoisciCelotnoSekvencoBilanc_R_Z(). Prikazujejo se podatki za zadnji 2 obdobji.

Tabela 10 največjih podjetij v panogi	Zlista se 10 največjih podjetij v panogi po prihodkih od prodaje.
Tabela primerjave panoge, nadpanoge in gospodarstva	Primerjava izbranih 13 vrednosti za zadnji 2 obdobji.
Graf trend poslovanja	Prikaz podatkov o trendu poslovanja za zadnji 2 obdobji
Graf doseženega števila točk poslovnega subjekta ,panoge, nadpanoge, in gospodarstva	Prikaz podatkov o doseženem štvilu točk za izbrano podjetje, panogo, nadpanogo in gospodarstvo za zadnji 2 obdobji.
Bonitetna razred panoge	Grafičen prikaz doseženega bonitetnega razreda panoge za zadnji 2 obdobji
Tabela položaj podjetja v panogi	Tabelaričen prikaz umestitve podjetja v panogi glede na tržne deleže.

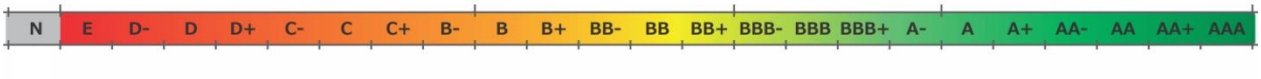
Tabela 7: sklopi analize panoge

Tabela položaj podjetja v panogi

- Prikaz razmerja (v odstotkih) prihodkov od prodaje izbranega podjetja glede na prihodke od prodaje celotne panoge za zadnji 2 obdobji.
- Prikaz razmerja (v odstotkih) velikosti sredstev izbranega podjetja glede velikosti sredstev celotne panoge za zadnji 2 obdobji.
- Izračun odstotka podjetji v panogi, ki spadajo v višji/nižji bonitetni razred kot izbrano podjetje

Postopek izračun odstotka podjetji v panogi, ki spadajo v višji/nižji bonitetni razred kot izbrano podjetje:

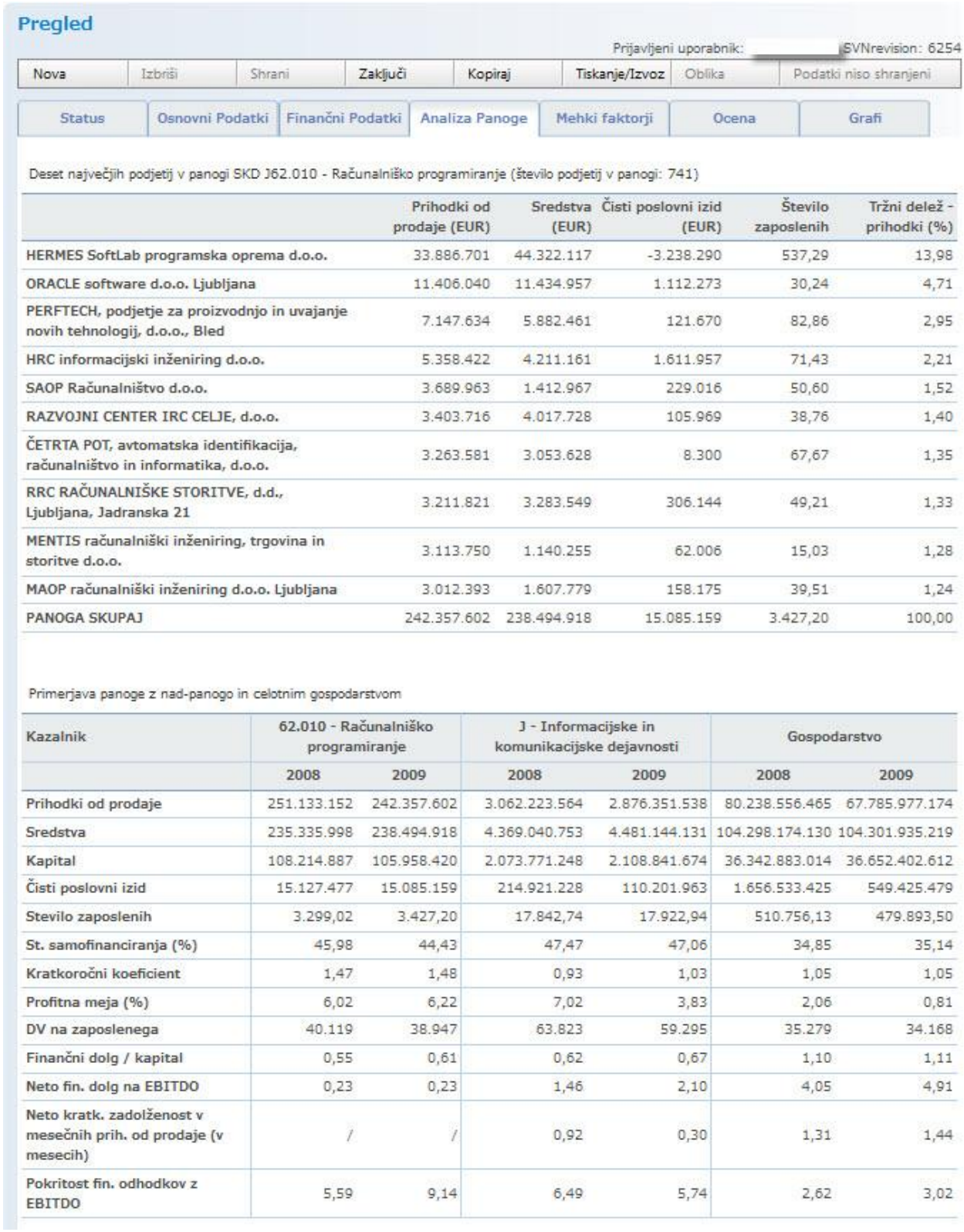
1. Prebere se vrednost ‘glavna_dejavnost_skd’ za izbrano podjetje
2. Poišče se vsa podjetja v podatkovni bazi (shema ajpes), ki imajo pod ‘glavna_dejavnost_skd’ navedeno šifro dejavnosti, katero smo prebrali v točki 1
3. Poišče se vrednost bonitetnega razreda v tabeli ocen za zadnji 2 obdobji za vsa podjetja iz točke 2.
4. Ločeno se prešteje število podjetji z višjim in z nižjim bonitetnim razredom. Za uvrščanje se uporabi klasifikator, ki ga ponazarja Slika 42. Razred A je višji od razreda B.



Slika 42: barvna lestvica bonitetnih razredov

Postopek izračunavanja odstotka podjetij z višjim/nižjim bonitetnim razredom je zelo počasen. Kot vidimo iz opisa postopka je temu tako, ker je potrebno narediti več zaporednih branj iz podatkovne baze poleg tega lahko posamezna panoga vsebuje tudi do 1000 podjetij.

Enako velja za tabelo 10 največjih podjetij v panogi. Tam je potrebno izvesti enako kot tu, koraka 1 in 2. Namesto koraka 3, se prebere vrednosti bilanc samo za zadnje obdobje. Nato je potrebno narediti še razvrščanje od največjega proti najmanjšemu in vrniti le 10 največjih podjetij po prihodkih od prodaje.



Pregled

Prijavljeni uporabnik: SVNrevision: 6254

Nova	Izbriši	Shrani	Zaključi	Kopiraj	Tiskanje/Izvoz	Oblika	Podatki niso shranjeni

Status | Osnovni Podatki | Finančni Podatki | Analiza Panoge | Mehki faktorji | Ocena | Grafi

Deset največjih podjetij v panogi SKD J62.010 - Računalniško programiranje (število podjetij v panogi: 741)

	Prihodki od prodaje (EUR)	Sredstva (EUR)	Čisti poslovni izid (EUR)	Število zaposlenih	Tržni delež - prihodki (%)
HERMES SoftLab programska oprema d.o.o.	33.886.701	44.322.117	-3.238.290	537,29	13,98
ORACLE software d.o.o. Ljubljana	11.406.040	11.434.957	1.112.273	30,24	4,71
PERFTECH, podjetje za proizvodnjo in uvajanje novih tehnologij, d.o.o., Bled	7.147.634	5.882.461	121.670	82,86	2,95
HRC informacijski inženiring d.o.o.	5.358.422	4.211.161	1.611.957	71,43	2,21
SAOP Računalništvo d.o.o.	3.689.963	1.412.967	229.016	50,60	1,52
RAZVOJNI CENTER IRC CELJE, d.o.o.	3.403.716	4.017.728	105.969	38,76	1,40
ČETRTA POT, avtomatska identifikacija, računalništvo in informatika, d.o.o.	3.263.581	3.053.628	8.300	67,67	1,35
RRC RAČUNALNIŠKE STORITVE, d.d., Ljubljana, Jadranska 21	3.211.821	3.283.549	306.144	49,21	1,33
MENTIS računalniški inženiring, trgovina in storitve d.o.o.	3.113.750	1.140.255	62.006	15,03	1,28
MAOP računalniški inženiring d.o.o. Ljubljana	3.012.393	1.607.779	158.175	39,51	1,24
PANOGA SKUPAJ	242.357.602	238.494.918	15.085.159	3.427,20	100,00

Primerjava panoge z nad-panogo in celotnim gospodarstvom

Kazalnik	62.010 - Računalniško programiranje		J - Informacijske in komunikacijske dejavnosti		Gospodarstvo	
	2008	2009	2008	2009	2008	2009
Prihodki od prodaje	251.133.152	242.357.602	3.062.223.564	2.876.351.538	80.238.556.465	67.785.977.174
Sredstva	235.335.998	238.494.918	4.369.040.753	4.481.144.131	104.298.174.130	104.301.935.219
Kapital	108.214.887	105.958.420	2.073.771.248	2.108.841.674	36.342.883.014	36.652.402.612
Čisti poslovni izid	15.127.477	15.085.159	214.921.228	110.201.963	1.656.533.425	549.425.479
Stevilo zaposlenih	3.299,02	3.427,20	17.842,74	17.922,94	510.756,13	479.893,50
St. samofinanciranja (%)	45,98	44,43	47,47	47,06	34,85	35,14
Kratkoročni koeficient	1,47	1,48	0,93	1,03	1,05	1,05
Profitna meja (%)	6,02	6,22	7,02	3,83	2,06	0,81
DV na zaposlenega	40.119	38.947	63.823	59.295	35.279	34.168
Finančni dolg / kapital	0,55	0,61	0,62	0,67	1,10	1,11
Neto fin. dolg na EBITDO	0,23	0,23	1,46	2,10	4,05	4,91
Neto kratk. zadolženost v mesečnih prih. od prodaje (v mesecih)	/	/	0,92	0,30	1,31	1,44
Pokritost fin. odhodkov z EBITDO	5,59	9,14	6,49	5,74	2,62	3,02

Slika 43: analiza panoge - prvič

Položaj podjetja v panogi

Položaj podjetja v panogi	2008	2009
Tržni delež glede na prihodke od prodaje	76,80%	79,27%
Tržni delež glede na velikost sredstev	99,37%	99,39%
Podjetja, ki imajo višji ali enak bonitetni razred	47%	50%
Podjetja, ki imajo nižji bonitetni razred	53%	50%

Slika 44: analiza panoge - drugič

4.9.4.2 Grafi

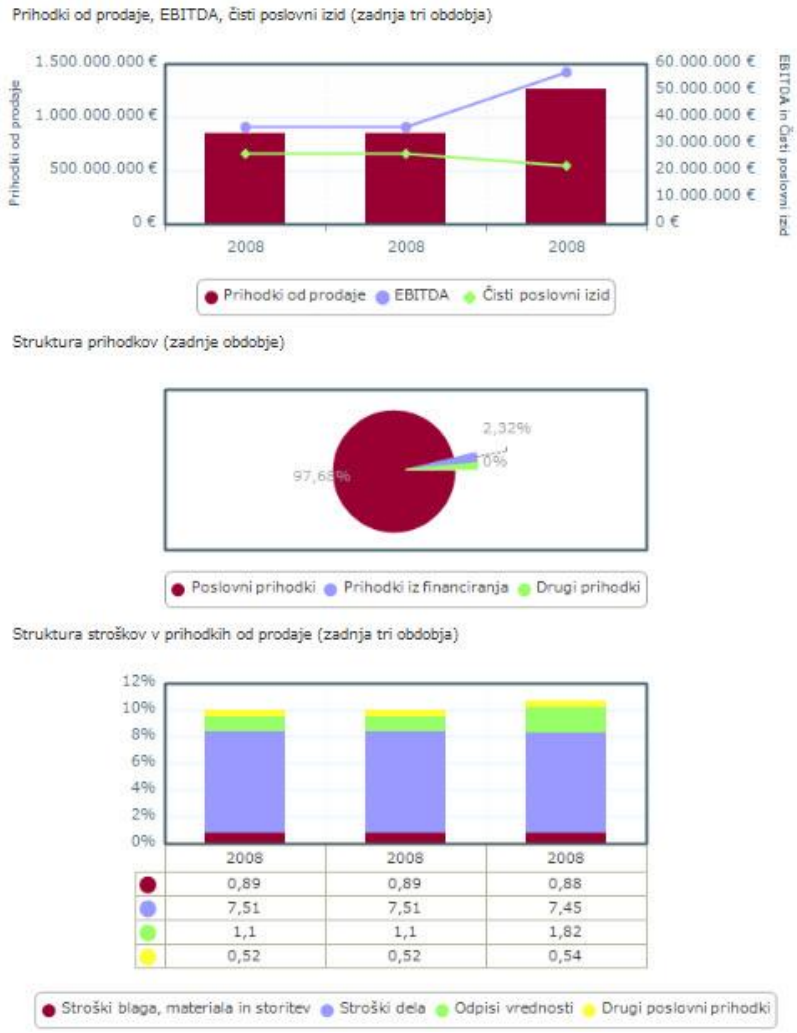
Podstran Grafi je namenjena grafičnemu prikazu podatkov. Podatki so vezani na izbrane bilance oz. finančne podatke ob ustvarjanju nove bonitete. V kolikor se ob izbiri bilanc ne izbere dovoljšne število bilanc, kolikor jih zahteva posamezen graf za prikaz, se izvede algoritem za dodatne bilance, ki je opisan v poglavju 4.6.2. Podatki za posamezen sklop se pridobi iz finančnih podatkov (bilance, kazalniki ali ocene). (Slika 33 in Slika 34). Grafi se ustvarijo za sledeče sklope:

- Prihodki od prodaje, EBITDA, čisti poslovni izid (zadnja 3 obdobja)
- Struktura prihodkov (zadnje obdobje)
- Struktura stroškov v prihodkih od prodaje (zadnja 3 obdobja)
- Akumulacija v prihodkih od prodaje (zadnje obdobje)
- Donosnost lastniškega kapitala (zadnja 3 obdobja)
- Struktura aktive in pasive (zadnje obdobje)
- Primerjava finančnega dolga s kapitalom (zadnja 3 obdobja)
- Ocena denarnega toka (zadnje obdobje)

** zadnje obdobje pomeni najmlajši izbran finančni podatek, bilanco,kazalnik, oceno*

RadChart

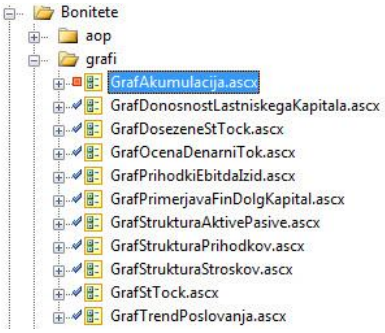
Za implementacijo grafičnega prikaza podatkov smo uporabili telerikovo kontrolo RadChart. Možno je izbirati med različnimi tipi grafov, ki jih ta kontrola ponuja. Tako najdemo med njimi denimo tipe diagramov kot so stolpični, linijski, tortni, Ganttov, področni, svečasti in drugi. Diagrame je možno tudi kombinirati, npr. linijskega in stolpičnega. Grafi omogočajo funkcionalnosti, kot so povečava, avtomatsko prilagajanje teksta, senčenje, prikaz tabele vrednosti in druge. Podatke je možno grafu podati na več načinov oz. z različnimi podatkovnimi strukturami. Vrednosti se lahko poda v obliki tabel, list, kompleksnih objektov, z uporabo data set-a, datoeke xml ali pa kar direktno z uporabo data source-a. Ker gre za ajax kontrolo, je možno posodabljanje podatkov narediti asinhrono, s čimer se doseže odzivnost aplikacije. Ustvarjeni graf je možno shraniti kot sliko za nadaljno uporabo. Možno je tako ročno, kakor avtomatsko nastavljanje label, osi, velikosti pisav, barve in drugo [22]. Primer RadChart grafa prikazuje Slika 45Slika 45.



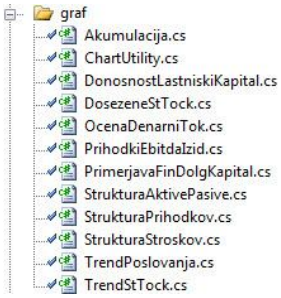
Slika 45: primeri grafov

Arhitektura podstrani grafi

Za vsak graf posebej je ustvarjena svoja lastna kontrola .ascx (Slika 47), katera vsebuje natančno določen graf.



Slika 47: arhitektura podstrani grafi



Slika 46: razredi za kreiranje objektov za ustvarjanje grafov

Vsaka izmed kontrol vsebuje javno metodo, ki jo je potrebno poklicati, da se graf ustvari. Kot argument se poda lista primernih objektov, katere definirajo rezredi prikazani na sliki (Slika 46). Objekti se ustvarijo in napolnijo s podatki iz podatkovne baze.

Primer: ustvarjanje grafa "Prihodki od prodaje, EBITDA, čisti poslovni izid (zadnja 3 obdobja)"

1. Naredi se dostop do izbrane bonitete v podatkovni bazi
2. Prebere se id-je bilanc oz. finančnih podatkov, ki jih boniteta vsebuje v tabeli "izbrane_bilance" in so vezani na id trenutne bonitete.
3. Naredi se dostop do tistih finančnih podatkov katere id-je pridobimo v točki 2
4. Ustvarimo nov objekt PrihodkiEbitdaIzid
5. Objektu nastavimo lastnosti, ki so numeričnega tipa (Leto, PrihodkiOdProdaje,EBITDA,CisiPoslovniIzid). Posamezna lastnost objekta predstavlja določeno polje v zapisi finančnega podatka. Ustvarimo toliko objektov, kot smo dobili različnih finančnih podatkov v točki 2. Objekte se doda v listo objektov. (List< PrihodkiEbitdaIzid>).
6. Pokliče se javno metodo NapolniGraf() kontrole GrafPrihodkiEbitdaIzid.ascx kateri se kot argument poda lista iz točke 5 ter izbrani jezik
7. Metoda NapolniGraf() priredi vrednosti grafu. Lista, ki jo metoda prejme kot argument se priredi RadChart.DataSource-u. Po potrebi roočno nastavimo še labele ter osi, katere se preberejo iz lokalizatorja, glede na jezik, ki ga prejmemo kot argument funkcije. S tem zadostimo potrebi po prevajanju label, osi in drugih opisnih tekstov. Pokličemo še metode, ki nam enovito določajo barve uporabljene v grafu. S tem zagotovimo, da so

barve pri vseh grafih iste. Enako velja za velikosti tekstov in druge grafične elemente prikazane na grafu, formate izpisanih števil in druge.

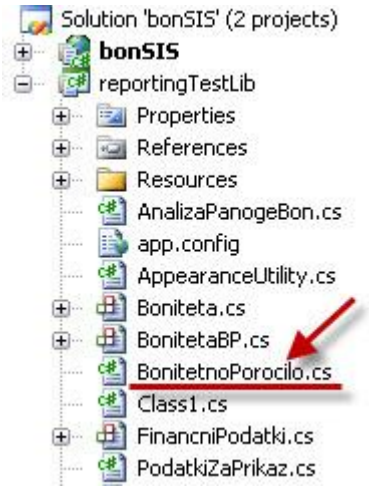
8. Na koncu znotraj metode NapolniGraf() pokličemo RadChart .DataBind(), ki nam ustvari graf.

Naj navedemo, da uporaba lastnih razredov ni nujna za ustvarjanje grafa. Enak učinek, bi lahko dosegli, tudi če bi grafu vrednosti, ki smo jih prebrali iz podatkovne baze podajali v obliki tabel numeričnih vrednosti. Pomembno je tudi vedeti, da zna telerik kontrola sama izluščiti vrednosti posameznega objekta glede na lastnosti, ki jih ta objekt ima, vendar pa morajo biti te lastnosti javne.

4.10 Bonitetno poročilo

Ena glavnih funkcionalnosti bonitetnega informacijskega sistema je izdelava bonitetnega poročila oz. izvoz podatkov v dokument. Izdelava bonitetnega poročila je narejena s pomočjo kontrole Telerik Reporting, katera je opisana že v poglavju 3.4.2. Za potrebe bonitetnega informacijskega sistema smo uporabili izvoz v Excel, izvoz v PDF ter možnost takojšnega tiskanja. Za prikaz podatkov smo poleg navadnih polj, uporabili tudi tabele podatkov, slike ter grafe. Podatki se v poročilo pošljejo preko objekta tako, da poročilu ni potrebno skrbeti za povezavo z bazo.

V bonitetnem informacijskem sistemu, smo Telerik Reporting [12] vkomponirali v projekt 'reportingLib' (Slika 48). Projekt smo kot knjižnico nato dodali v projekt 'bonSIS' kjer se nahaja spletna aplikacija. Spletna aplikacija pri komunikaciji lahko dostopa do objektov knjižnice 'reportingLib', medtem ko komunikacija v nasprotni smeri ni možna.
Za prenos podatkov iz spletnega vmesnika v poročilo, se uporablja objekt 'BonitetnoPorocilo'. Spletna aplikacija zapiše podatke v omenjeni objekt ter pokliče željeno poročilo, da se kreira.



Slika 48: struktura projekta reportingLib

Pri kreiranju poročila ‘BonitetaBP’, se objekt ‘BonitetnoPorocilo’ vključi že pri klicu konstruktorja (Slika 50). Poročilo iz nastavljenega objekta prebere podatke in jih vpiše v pripravljene elemente. V aplikaciji se nato ustvarjeno poročilo prikaže. Za prikaz poskrbi kontrola ‘ReportViewer’ [23] (Primer prikazuje Slika 49).

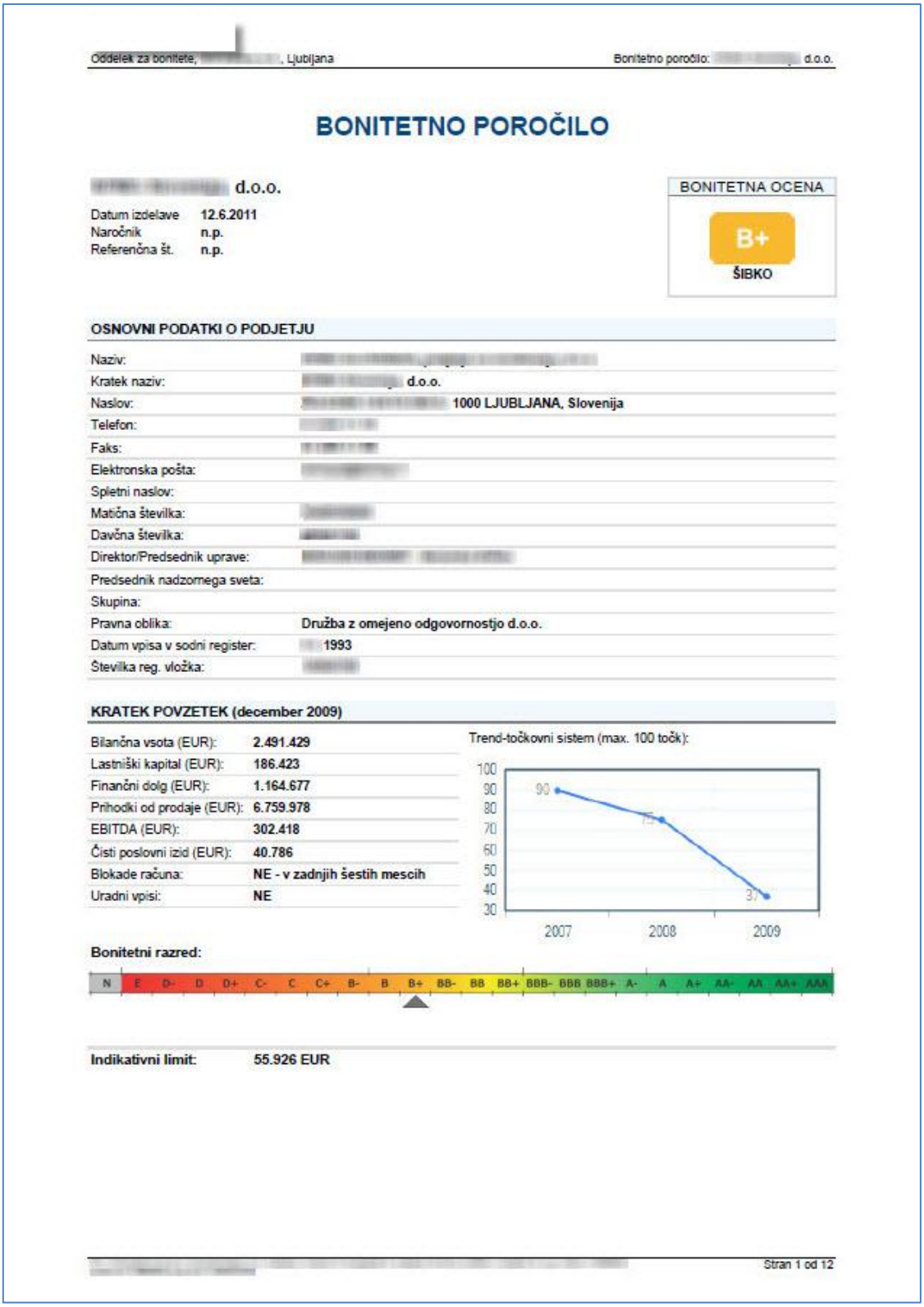
```
<telerik:ReportViewer ID="ReportViewer1" runat="server"
    DocumentMapVisible="False"
    ZoomMode="FullPage" Enabled="true" Height="0px" ToolbarVisible="false"
    Width="0px" ProgressText="">
</telerik:ReportViewer>
```

Slika 49: dodajanja telerikove ReportViewer komponente na aspx stran

```
private Telerik.Reporting.Report GenerirajTelerikPorocilo(BonitetnoPorocilo porocilo)
{
    return new reportingTestLib.BonitetaBP(porocilo);
}
```

Slika 50: primer klica za generiranje poročila

Polnjenje podatkov objekta ‘BonitetnoPoročilo’ poteka po delih. Vsaka podstran v modulu bonitet poskrbi, da se potrebne informacije zapišejo v primerne lastnosti objekta. Poleg vsebin se v objektu za posamezen element ali sklop nastavi še njegova vidnost. V aplikaciji je namreč možno vklapljati ter izklapljati posamezne elemente oz. jim nastavljati vidnost. Pri polnjenju poročila potem logika poskrbi, da se glede na vidnosti posameznega elementa objekta element v poročilu prikaže ali ne. Možen je izklop celotnega sklopa ali pa samo komentarja.
Grafični elementi, kot so denimo grafi, se v poročilo prenesejo kot slike. Alternativna možnost bi bila tudi, da se prenesejo podatki, grafi pa bi se nato v poročilu ustvarili ponovno. To možnost ‘telerik reporting’ omogoča, vendar pa se bi takšno poročilo vsakič ustvarjalo počasneje poleg tega ne bi mogli zagotoviti popolne skladnosti izgleda grafa v aplikaciji s tistim na poročilu, saj ne gre za iste razrede telerikovih knjižnic.



Oddelek za bonitete, , Ljubljana Bonitetno poročilo: d.o.o.

BONITETNO POROČILO

d.o.o.

Datum izdelave 12.6.2011
Naročnik n.p.
Referenčna št. n.p.

BONITETNA OCENA
B+
ŠIBKO

OSNOVNI PODATKI O PODJETJU

Naziv:	
Kratek naziv:	d.o.o.
Naslov:	1000 LJUBLJANA, Slovenija
Telefon:	
Faks:	
Elektronska pošta:	
Spletni naslov:	
Matična številka:	
Davčna številka:	
Direktor/Predsednik uprave:	
Predsednik nadzornega sveta:	
Skupina:	
Pravna oblika:	Družba z omejeno odgovornostjo d.o.o.
Datum vpisa v sodni register:	1993
Številka reg. vložka:	

KRATEK POVZETEK (december 2009)

Bilančna vsota (EUR):	2.491.429
Lastniški kapital (EUR):	186.423
Finančni dolg (EUR):	1.164.677
Prihodki od prodaje (EUR):	6.759.978
EBITDA (EUR):	302.418
Čisti poslovni izid (EUR):	40.786
Blokade računa:	NE - v zadnjih šestih mescih
Uradni vpisi:	NE

Trend-točkovni sistem (max. 100 točk):

Bonitetni razred:

N E D- D D+ C- C C+ B- B B+ BB- BB BB+ BBB- BBB BBB+ A- A A+ AA- AA AA+ AAA

Indikativni limit: 55.926 EUR

Stran 1 od 12

Slika 51: bonitetno poročilo

4.10.1 Serializacija objekta v format XML

Za namene testiranja in razvijanja, smo razred BonitetnoPorocilo zasnovali tako, da lahko objekt, ki se kreira s pomočjo razreda v določenem trenutku zapišemo v datoteko XML. Prednost tega je, da lahko v primeru anomalije pri kreiranju poročila reproduciramo morebitno težavo. Primer: razvijalec kreira bonitetno poročilo, to se mu kreira in odpre v dokumentu PDF. Odpre dokument PDF in ugotovi, da izgled ne ustreza pričakovanemu. Razvijalec lahko sedaj z enim klikom v aplikaciji zapiše stanje objekta v datoteko. Za nadaljne delo mu ni potrebno vsakič posebej ustvarjati ali pregledovati točno določene bonitete preko aplikacije Bonitetni informacijski sistem in zahtevati izvoz v dokument. Za ta namen uporabi testni projekt, v katerem je dodana knjižnica ‘reportingLib’. V projektu se nahaja preprost program. Njegova naloga je sledeča:

1. Iz vhodne datoteke prebere vrednosti
2. Kreira objekt BonitetnoPorocilo in mu nastavi vrednosti iz tocke 1
3. Ustvari bonitetno poročilo. Klic je enak tistemu v Bonitetnem informacijskem sistemu (Slika 50)

Do potrebe po serializacije objekta smo prišli empirično med razvojem bonitetnega informacijskega sistema. Zgodil se je denimo primer, ko se je generirano poročilo prikazalo popačeno. Ker smo se odločili za plačljive Telerikove kontrole, med katere sodi tudi Telerik Reporting, smo imeli možnost za pomoč pri odkrivanju napak povprašati strokovnjaka iz vrst Telerika. Za lažje odkrivanje napak, nas je le ta poprosil po izvorni kodi, ali pa vsaj delu kode. Ker je pošiljanje celotnega projekta seveda nesmiselno iz mnogih vidikov (mednje sodi obseg programske kode, zahtevna postavitev delotnega sistema, zagotovilo zasebnosti itd.) je bilo hitro na mestu odločitev po zapisu dejanskega objekta v primerno obliko za nadaljno rabo. Strokovnjakom smo tako enostavno posredovali le delujoč testni projekt, v katerega smo dodali datoteko XML z zapisom objekta s programom za deserializacijo objekta in kreiranje realnega poročila.

Zapis objekta v datoteko XML se izvede s pomočjo t.i. serializacije objekta v XML format, kar omogoča ogrodje .NET. Primer implementacije je prikazan na spodnji sliki (Slika 52). Vrednosti lastnosti in polj objekta se zapišejo se pri tem pretvorijo v elemente oz. atribute XML. Lastnosti objekta morajo imeti vidnost nastavljeno na javno (‘public’). Osnovni namen serializacije je možnost pretvorbe dokumenta XML v objekt uporaben v programskem jeziku in obratno. Serializacija objektov v XML omogoča trajnost in prenosljivost stanja objekta na standarden način. Serializacija XML podpira serializacijo predmetov kot XML, skladen z 'W3C XML Schema Definition (XSD)'. [4]

Prenosljivost
Primer: Če imate objekt A in ga serializirate v datoteko XML, ni nujno, da se ob deserializaciji izdatoteke XML kreira objekt istega tipa. V drugem primeru lahko denimo deserializacijo

opravite znotraj drugega projekta, ki vsebuje razred, kateri nosi enako ime, kot razred, ki je definiral objekt A. To pomeni, da se lahko datoteka preprosto prenese v drug projekt, na drug računalnik brez potrebe po prenosu razreda. Prenosljivost omogoča tudi, da ni potrebe, da se deserializacija opravi v ogrodju .NET, četudi je bila serializacija opravljena v njem. Opravi se lahko v katerem koli drugem okolju, ki jo omogoča. Serializacija objektov je koristen mehanizem za zagotavljanje ter uvažanje podatkov. Dokumenti XML pretvorjeni v objekte z deserializacijo

```
using System.Xml.Serialization;
using reportingTestLib;

namespace XMLSerialization
{
    public partial class Form1 : Form
    {
        string path = @"C:\Podatki\IzvozBonitetnoPorocilo.xml";
        private void button1_Click(object sender, EventArgs e)
        {
            #region Zapis objekta v datoteko
            BonitetnoPorocilo bp = new BonitetnoPorocilo();
            TextWriter tw = new StreamWriter(path);
            XmlSerializer sr = new XmlSerializer(typeof(Avto));
            sr.Serialize(tw, bp);
            tw.Close();
            #endregion

            #region Branje objekta iz datoteke
            FileStream f = new FileStream(path, FileMode.Open);
            BonitetnoPorocilo bpRead = (BonitetnoPorocilo)sr.Deserialize(f);
            #endregion
        }
```

Slika 52: primer serializacije in deserializacije objekta v datoteko XML

5 Zaključek

V diplomski nalogi sem prikazal funkcionalnost bonitetnega informacijskega sistema, kateri služi analitiku kot orodje za pregledovanje in urejanje podatkov o poslovnih subjektih, izdelavi bonitet, bonitetnih poročil, oddaji naročil, izdelavi bilanc ter sledenje izdelkom.
Prikazani so deli sistema oziroma moduli aplikacije, pri razvoju katerih sem bil udeležen tudi sam.

Podrobno je predstavljena funkcionalnost modulov, na kakšen način je posamezen modul implementiran, njegov namen ter njegova uporaba s strani končnega uporabnika.
Opisane so uporabljene tehnologije, različni gradniki sistema ter njihova medsebojna interakcija in vključitev v končni izdelek. Predstavljana je uporaba telerikovih naprednih kontrol v ASP.NET aplikaciji, povezava aplikacije s sistemom LDAP, uporaba dnevniške datoteke ter povezava aplikacije ASP.NET s podatkovno bazo DB2.
Posamezni gradniki, za katere sem menil, da si zaslužijo predstavitev, so opisani v poglavjih, kjer se tudi uporabljajo. Za nazornejšo predstavo so besedilu ponekod dodane slike in diagrami, ki bodo bralcu poenostavile razumevanje.

Razvoju bonitetnega informacijskega sistema so prisostvovale tudi težave. Mednje sodijo težave pri postavitvi končnega sistema na strežnik IIS 6.0. Pri omenjenem strežniku ter tudi kasnejših verzijah, je velikost sklada zmanjšana na zgolj 256KB. (pri IIS 5.1 je velikost sklada omejena na 1MB) [24]. Na težavo z omejitvijo velikosti sklada smo naleteli pri ustvarjanju pdf poročil, rešili pa smo jo tako, da smo v telerikovih reportih uporabili podreporte. Omeniti velja še omejenost telerikove kontrole ReportViewer pri kateri je potreben pravi ‘PostBack’, kar pomeni da kontrola pri posredovanju poročila ne more vrniti tega v obliki ajax odgovora, temveč se mora osvežiti celotna stran. Eno izmed težav pri odzivnosti predstavlja tudi ustvarjanje bonitetnega poročila, kateri potrebuje kar nekaj časa, predno se ustvari in je na voljo uporabniku.

Gledano na splošno, uporaba telerikovih kontrol v aplikaciji ASP.NET še vedno prinaša mnogo prednosti. Za končnega uporabnika je njihova največja prednost profesionalen izgled in izredna odzivnost, z vidika razvijalca aplikacije pa njihova preprosta implementacija v aplikaciji ASP.NET ter strokovna pomoč. Uporaba avtentikacije ‘windows’

V poglavju 5.1 podajam možne nadgradnje bonitetnega informacijskega sistema.

5.1 Možne nadgradnje sistema

Za dostop do aplikacije, bi se namesto protokola LDAP ter Active Directory-ja lahko uporabil način dostopa z uporabniškim imenom ter geslom. Pri tem, bi bilo potrebno implementirati še modul, kjer bi se uporabnik lahko registriral ter urejal podatke za dostop do aplikacije. Aplikacija bi morala imeti tudi možnost ponastavljanja gesel.

Možnost uporabe protokola HTTPS za zagotovitev varnosti pri prenosu podatkov.

Poleg že obstoječega formata PDF ter Excel, dodati možnost izvoza podatkov v obliki XML datoteke, kot možnost za prenašanje podatkov in ponovno uporabo le the v drugih aplikacijah.

Implementacija lokalizacije tekstov za celoten informacijski sistem, ne samo za potrebe bonitet ter bonitetnih poročil. Prednost le tega bi bila v primeru potrebe po informacijskem sistemu v drugem jeziku. Prevode tekstov za celotno aplikacijo bi bilo, zaradi odzivnosti sistema smiselno hraniti v datoteki, in ne v podatkovni bazi. Takšen način lokalizacije bi bil statičen in analitiku ne bi omogočal poljubnega spreminjanja teksta. Lokalizacija pri bonitetah bi se ohranila takšna kot je in ne bi bila v nasprotju z lokalizacijo celotnega informacisjkega sistema.

Dodajanje novih jezikov pri bonitetah.

Pri modulu ‘Poslovni subjekti’ dodati morebitne povezave na domače strani izbranih podjetij.

Razširitev obstoječe funkcionalnosti za tuja podjetja.

Možnost ponudbe določenih podatkov v obliki spletnih storitev (‘web service’) in s tem možnost povezovanja z drugimi poslovnimi aplikacijami, ki bi uporabljale podatke bonitetnega informacijskega sistema.

Dodatek

Nastavljanje vhodnih parametrov za vnos bonitete v podatkovno bazo

```
// izhodni parm
List<DB2Parameter> izhodniParametri = new List<DB2Parameter>();
DB2Parameter bonID_outPar = new DB2Parameter("BONITETA_ID_OUTPAR",
                        DB2Type.Integer);
izhodniParametri.Add(bonID_outPar);
// vhodnih parametri
List<DB2Parameter> vhodniParametri = new List<DB2Parameter>();
vhodniParametri.Add(new DB2Parameter("INSERTORUPDATE_PAR", DB2Type.SmallInt));
vhodniParametri.Last().Value = Convert.ToInt16(BonitetaID == -1);
// 1 povzroci INSERT, 0 update

. . .

vhodniParametri.Add(new DB2Parameter("KOMENTAR_PAR", DB2Type.VarChar, 8000));
vhodniParametri.Last().Value = bonRow["KOMENTAR"];
vhodniParametri.Add(new DB2Parameter("STEVILO_TOCK_PAR", DB2Type.Double));
vhodniParametri.Last().Value = bonRow["STEVILO_TOCK"];
vhodniParametri.Add(new DB2Parameter("INDIKATIVNI_LIMIT_PAR", DB2Type.Double));
vhodniParametri.Last().Value = bonRow["INDIKATIVNI_LIMIT"];

// Call procedure
try
{
        dbUtilityClass.DB2KlicBazneProcedure("BONSIS.INSERTUPDATE_BONITETE",
                        vhodniParametri, izhodniParametri);
}
catch (Exception exc)
{
        log.Error(exc.ToString());
}

// Read back the key
if (bonID_outPar.Value != null)
{
        BonitetaID = (int)(bonID_outPar.Value);

. . .
```

Metoda za klic bazne procedure

```
/// <summary>
/// DB2KlicBazneProcedure odpre povezavo na bazo in izvede klic procedure
/// </summary>
/// <param name="imeProcedure"></param>
/// <param name="vhodniParametri"></param>
/// <param name="izhodniParametri">V 'value' vsakega parametra se nahaja vrnjena
       vrednost.</param>
/// <returns>Število učinkovanih vrstic tabele</returns>
static public int DB2KlicBazneProcedure(string imeProcedure, List<DB2Parameter>
       vhodniParametri, List<DB2Parameter> izhodniParametri)
{
    DB2Command sqlProc = new DB2Command(imeProcedure,
                                dbUtilityClass.getDB2ConnectionS());
    sqlProc.CommandType = System.Data.CommandType.StoredProcedure;

    foreach (DB2Parameter par in vhodniParametri)
    {
        sqlProc.Parameters.Add(par).Direction = System.Data.ParameterDirection.Input;
    }
    foreach (DB2Parameter par in izhodniParametri)
    {
        sqlProc.Parameters.Add(par).Direction = System.Data.ParameterDirection.Output;
    }

    // Izcedi klic procedure
    sqlProc.Connection.Open();
    int rowCount = -1;
    try
    {
        rowCount = sqlProc.ExecuteNonQuery();
    }
    catch
    {
        throw;
    }
    finally
    {
        sqlProc.Connection.Close();
    }
    return rowCount;
}
```

Bazna procedura za vnos bonitete

```
SET SCHEMA = 'DB2INST1';

CREATE PROCEDURE "BONSIS"."INSERTUPDATE_BONITETE" (
    IN "INSERTORUPDATE_PAR"   SMALLINT,
    IN "BONITETA_ID_PAR"      INTEGER,
    IN "JEZIK_ID_PAR"  INTEGER,
    IN "STATUS_ID_PAR" INTEGER,
    IN "POSLOVNI_SUBJEKT_PAR" VARCHAR(10),
    IN "VRSTA_POSL_SUBJ_PAR"  INTEGER,
    IN "OBLIKA_POSL_SUBJ_PAR" INTEGER,
    IN "DATUM_IZDELAVE_PAR"   DATE,
    IN "BONITETNI_RAZRED_PAR" VARCHAR(10),
    IN "BONITETNI_RAZRED_OPIS_PAR"    VARCHAR(100),
    IN "BILANCNO_OBDOBJE_OD_PAR"      DATE,
    IN "BILANCNO_OBDOBJE_DO_PAR"      DATE,
    IN "TIP_ZADNJE_BILANCE_PAR"       INTEGER,
    IN "JEZIK_POROCILA_PAR"   INTEGER,
    IN "TIP_POROCILA_PAR"     INTEGER,
    IN "PRIKAZ_DEN_TOKOV_PAR" SMALLINT,
    IN "PRIKAZ_STR_INDX_PAR"  SMALLINT,
    IN "PRIKAZ_MEHKI_F_PAR"   SMALLINT,
    IN "UPOR_IME_PAR"  VARCHAR(50),
    IN "SKUPINA_ID_PAR"       INTEGER,
    IN "NAZIV_BONITETE_PAR"   VARCHAR(250),
    IN "KOMENTAR_PAR"  VARCHAR(8000),
    IN "STEVILO_TOCK_PAR"     DOUBLE,
    IN "INDIKATIVNI_LIMIT_PAR"        DOUBLE,
   OUT "BONITETA_ID_OUTPAR"  INTEGER )
  SPECIFIC "SQL110518135606800"
  LANGUAGE SQL
  NOT DETERMINISTIC
  NO EXTERNAL ACTION
  MODIFIES SQL DATA
  CALLED ON NULL INPUT
  INHERIT SPECIAL REGISTERS
BEGIN
--Value of 1 means insert
IF INSERTORUPDATE_PAR = 1 THEN
INSERT INTO BONITETE ( JEZIK_ID, STATUS_ID, POSLOVNI_SUBJEKT, VRSTA_POSL_SUBJ,
OBLIKA_POSL_SUBJ, DATUM_IZDELAVE, BONITETNI_RAZRED, BILANCNO_OBDOBJE_OD,
BILANCNO_OBDOBJE_DO, TIP_ZADNJE_BILANCE, JEZIK_POROCILA, TIP_POROCILA, PRIKAZ_DEN_TOKOV,
PRIKAZ_STR_INDX, PRIKAZ_MEHKI_F, UPOR_IME, SKUPINA_ID, NAZIV_BONITETE, KOMENTAR,
STEVILO_TOCK, INDIKATIVNI_LIMIT, BONITETNI_RAZRED_OPIS)
VALUES (JEZIK_ID_PAR, STATUS_ID_PAR, POSLOVNI_SUBJEKT_PAR, VRSTA_POSL_SUBJ_PAR,
OBLIKA_POSL_SUBJ_PAR, DATUM_IZDELAVE_PAR, BONITETNI_RAZRED_PAR, BILANCNO_OBDOBJE_OD_PAR,
BILANCNO_OBDOBJE_DO_PAR, TIP_ZADNJE_BILANCE_PAR, JEZIK_POROCILA_PAR, TIP_POROCILA_PAR,
PRIKAZ_DEN_TOKOV_PAR, PRIKAZ_STR_INDX_PAR, PRIKAZ_MEHKI_F_PAR, UPOR_IME_PAR,
SKUPINA_ID_PAR, NAZIV_BONITETE_PAR, KOMENTAR_PAR, STEVILO_TOCK_PAR, INDIKATIVNI_LIMIT_PAR,
BONITETNI_RAZRED_OPIS_PAR);
--Return the autogenerated field: bilanca_id--
SELECT Identity_val_Local() INTO BONITETA_ID_OUTPAR FROM BONITETE FETCH FIRST 1 ROWS ONLY;

--Create a new record for 'ostali podatki' so therefore later can be just updated by each
page individually--
INSERT INTO BON_OSTALI_PODATKI (BONITETA_ID)
VALUES (BONITETA_ID_OUTPAR);

END IF;
--Value of 0 means update
IF INSERTORUPDATE_PAR = 0 THEN
UPDATE BONITETE SET JEZIK_ID=JEZIK_ID_PAR, STATUS_ID=STATUS_ID_PAR,
POSLOVNI_SUBJEKT=POSLOVNI_SUBJEKT_PAR, VRSTA_POSL_SUBJ=VRSTA_POSL_SUBJ_PAR,
OBLIKA_POSL_SUBJ=OBLIKA_POSL_SUBJ_PAR,DATUM_IZDELAVE=DATUM_IZDELAVE_PAR,
BONITETNI_RAZRED=BONITETNI_RAZRED_PAR, BILANCNO_OBDOBJE_OD=BILANCNO_OBDOBJE_OD_PAR,
TIP_ZADNJE_BILANCE=TIP_ZADNJE_BILANCE_PAR, BILANCNO_OBDOBJE_DO=BILANCNO_OBDOBJE_DO_PAR,
JEZIK_POROCILA=JEZIK_POROCILA_PAR, TIP_POROCILA=TIP_POROCILA_PAR,
```

```
PRIKAZ_DEN_TOKOV=PRIKAZ_DEN_TOKOV_PAR, PRIKAZ_STR_INDX=PRIKAZ_STR_INDX_PAR,
PRIKAZ_MEHKI_F=PRIKAZ_MEHKI_F_PAR, UPOR_IME=UPOR_IME_PAR, SKUPINA_ID=SKUPINA_ID_PAR,
NAZIV_BONITETE=NAZIV_BONITETE_PAR, KOMENTAR=KOMENTAR_PAR, STEVILO_TOCK=STEVILO_TOCK_PAR,
INDIKATIVNI_LIMIT=INDIKATIVNI_LIMIT_PAR, BONITETNI_RAZRED_OPIS = BONITETNI_RAZRED_OPIS_PAR
WHERE BONITETA_ID = BONITETA_ID_PAR;
SET BONITETA_ID_OUTPAR = BONITETA_ID_PAR;
END IF;

END;

SET SCHEMA = 'SYSIBM';

GRANT EXECUTE ON PROCEDURE "BONSIS"."INSERTUPDATE_BONITETE"( SMALLINT, INTEGER, INTEGER,
INTEGER, VARCHAR(10), INTEGER, INTEGER, DATE, VARCHAR(10), VARCHAR(100), DATE, DATE,
INTEGER, INTEGER, INTEGER, SMALLINT, SMALLINT, SMALLINT, VARCHAR(50), INTEGER,
VARCHAR(250), VARCHAR(8000), DOUBLE, DOUBLE, INTEGER ) TO USER "DB2INST1" WITH GRANT
OPTION;

SET SCHEMA = 'DB2INST1';
```

Metoda PoisciCelotnoSekvencoBilanc()

```
/// <summary>
///
/// Metodologija za retreivanje ID-jev bilanc.
/// - metoda sprejme ZE izbrano listo bilanc.
/// V kolikor lista ze vsebuje dovoljsne stevilo elementov
/// potem ne naredi nicesar, v nasprotnem primeru pa omenjenim elementom
/// liste doda se dodatne ID-je bilanc/fin podatkov.
///
/// </summary>
/// <param name="finPodIds_izbrani"> Lista ID-jev izbranih financnih podatkov</param>
/// <param name="stElmZaVracilo">Koliko elementov želimo, da nam metoda vrne</param>
/// <param name="maticnaSt">dolga maticna stevilka poslovnega subjekta</param>
/// <param name="returnBilanceId"> TRUE -> vrne listo ID-jev bilanc, FALSE -> vrne listo
      ID-jev financnih podatkov</param>
/// <returns></returns>
public List<int> PoisciCelotnoSekvencoBilanc( List<int> finPodIds_izbrani, int
      stElmZaVracilo, string maticnaSt, bool returnBilanceId )
{
    List<int> rezBilanceIds;

    // izbranih bilanc je DOVOLJ,(toliko kot rabimo elementov za prikaz), zato ni
       potrebno it gledat v bazo za
    // dodatna obdobja (leta) nazaj
    if ( finPodIds_izbrani.Count == stElmZaVracilo )
    {
        if (!returnBilanceId)
        {
            return finPodIds_izbrani;
        }
        else
        {
            rezBilanceIds = new List<int>();
            foreach (int elmi in finPodIds_izbrani)
            {
                rezBilanceIds.Add(FinPodatkiUtility.getBilancaID_byFinPodatekID(elmi));
            }
            return rezBilanceIds;
        }
    }
    // izbranih bilanc je PREMALO, potrebno je pogledat za dodatna po letih nazaj
    else
    {

        // 1. SORTiraj obstojece --> da ves kateri je zadnji, ker glede na zadnjega
              izbranega se gleda
        // TIP_ID (ali je konsolidirana ali nekonsolidirana)
        #region Sort izbranih

        List<FinancniPodatki> finPodatkiIzbraniSort = new List<FinancniPodatki>();
        foreach (int elmi in finPodIds_izbrani)
        {
            FinancniPodatki fp = new FinancniPodatki();
            fp.najdiFinPodatke(elmi);
            finPodatkiIzbraniSort.Add(fp);
        }
        finPodatkiIzbraniSort = finPodatkiIzbraniSort.OrderBy(x => x.DatumDo).ToList();
        int TIP_ID_zadnjeIzbrane = finPodatkiIzbraniSort.Last().TipID;
        int YEAR_najstarejse_izbrane = finPodatkiIzbraniSort.First().DatumDo.Year;
        #endregion

        #region Preberi VSE fin podatke za izbran poslovni subjekt

        DataView vsiFinPodatki =
              FinPodatkiUtility.getAllFinPodatki_fromDatabase(maticnaSt);
        List<FinancniPodatki> finPodatkiAll = new List<FinancniPodatki>();
        if (vsiFinPodatki != null)
```

```
{
    finPodatkiAll = new List<FinancniPodatki>();
    for (int i = 0; i < vsiFinPodatki.Count; i++)
    {
        FinancniPodatki fp = new FinancniPodatki();
        fp.FinPodatkiID =
              int.Parse(vsiFinPodatki[i]["FIN_PODATKI_ID"].ToString());
        fp.BilancaID = int.Parse(vsiFinPodatki[i]["BILANCA_ID"].ToString());
        fp.DatumDo = (DateTime)vsiFinPodatki[i]["DATUM_DO"];
        fp.DatumOd = (DateTime)vsiFinPodatki[i]["DATUM_OD"];
        fp.DatumIzdelave = (DateTime)vsiFinPodatki[i]["DATUM_IZDELAVE"];
        fp.VrstaPoslovnegaSubjektaID =
              int.Parse(vsiFinPodatki[i]["VRSTA_POSL_SUBJ"].ToString());
        fp.StatusID = int.Parse(vsiFinPodatki[i]["STATUS_ID"].ToString());
        fp.TipID = int.Parse(vsiFinPodatki[i]["TIP_ID"].ToString());
        fp.JezikID = int.Parse(vsiFinPodatki[i]["JEZIK_ID"].ToString());
        fp.FinPodatkiOblikaID =
              int.Parse(vsiFinPodatki[i]["FIN_POD_OBLIKA_ID"].ToString());
        finPodatkiAll.Add(fp);
    }
}
#endregion

#region Izberi samo Konsolidirane/Nekonsolidirane

List<FinancniPodatki> vsiFinPodatki_brezOdvecnihTipov =
       new List<FinancniPodatki>();

if (TIP_ID_zadnjeIzbrane >= 1 && TIP_ID_zadnjeIzbrane < 7) // Nekonsolidirane
    vsiFinPodatki_brezOdvecnihTipov = finPodatkiAll.Where(x => x.TipID >= 2 &&
      x.TipID <= 3).ToList();  // samo LETNE
else if (TIP_ID_zadnjeIzbrane >= 7 && TIP_ID_zadnjeIzbrane <= 12)// Konsolidirane
    vsiFinPodatki_brezOdvecnihTipov = finPodatkiAll.Where(x => x.TipID >= 8 &&
      x.TipID <= 9).ToList();  // samo LETNE

vsiFinPodatki_brezOdvecnihTipov = vsiFinPodatki_brezOdvecnihTipov
       .OrderBy(x => x.DatumDo).ToList();
#endregion

#region Odstrani Z, če za isto leto obstaja že R

var grouped = from v in vsiFinPodatki_brezOdvecnihTipov
              group v by v.DatumDo.Year into gr
              select gr.Key;

vsiFinPodatki_brezOdvecnihTipov =
       vsiFinPodatki_brezOdvecnihTipov.OrderByDescending(x => x.TipID).ToList();
List<FinancniPodatki> finPodatki_brezPodvojenihLet = new List<FinancniPodatki>();
foreach (int elm in grouped)
{
    FinancniPodatki fpTmp = vsiFinPodatki_brezOdvecnihTipov
      .First(x => x.DatumDo.Year == elm);
    finPodatki_brezPodvojenihLet.Add(fpTmp);
}
#endregion

#region Kombiniraj Izbrane + Zadnje

List<FinancniPodatki> FinPodatki_SKUPAJ = new List<FinancniPodatki>();

FinPodatki_SKUPAJ = finPodatki_brezPodvojenihLet
       .Where(x => x.DatumDo.Year < YEAR_najstarejse_izbrane).ToList();

#region debug
log.Debug("// Dodane //");
foreach (FinancniPodatki fpLog in FinPodatki_SKUPAJ)
{
    log.Debug(fpLog.FinPodatkiID + "[" + fpLog.BilancaID + "]" + ", " +
      fpLog.DatumDo.Year + ", " + fpLog.TipID);
}
```

```
            log.Debug("// Izbrane //");
            foreach (FinancniPodatki fpLog in finPodatkiIzbraniSort)
            {
                log.Debug(fpLog.FinPodatkiID + "[" + fpLog.BilancaID + "]" + ", " +
                  fpLog.DatumDo.Year + ", " + fpLog.TipID);
            }
            #endregion

            FinPodatki_SKUPAJ.AddRange(finPodatkiIzbraniSort);  // dodamo IZBRANE

            #endregion


            #region preberemo le toliko elementov, kot želimo
            // sortiramo najprej od najmlajše do najstarejše (npr: 2009,2008,2007)
            FinPodatki_SKUPAJ = FinPodatki_SKUPAJ.OrderByDescending(x => x.DatumDo).ToList();

            // preberemo le toliko elementov, kot jih želimo
            if (FinPodatki_SKUPAJ.Count >= stElmZaVracilo)
                FinPodatki_SKUPAJ = FinPodatki_SKUPAJ.GetRange(0, stElmZaVracilo);
            else
                FinPodatki_SKUPAJ = FinPodatki_SKUPAJ.GetRange(0, FinPodatki_SKUPAJ.Count);
            #endregion

            // Skoncni SORTs
            // sortiramo od najstarejše do najmlajše (npr: 2007,2008, 2009)
            FinPodatki_SKUPAJ = FinPodatki_SKUPAJ.OrderBy(x => x.DatumDo).ToList();


            #region vrni listo ID-jev (bodisi FinPodatkov, bodisi Bilanc)

            if (!returnBilanceId)
                return FinPodatki_SKUPAJ.Select(x => x.FinPodatkiID).ToList();
            else
                return FinPodatki_SKUPAJ.Select(x => x.BilancaID).ToList();

            #endregion
        }
    }
```

Kazalo slik

Kazalo tabel

Viri in literatura

- [1] DB2 Provider. Dostopno na: http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.ndp.doc/htm/frlrfIBMDataDB2.htm
- [2] Toad For DB2 Freeware 4.6. Dostopno na: http://www.toadworld.com/Freeware/ToadforDB2Freeware/tabid/560/Default.aspx
- [3] IBM DB2 8.1. Dostopno na: http://www-01.ibm.com/software/data/db2/
- [4] C.Nagel, B. Evjen, J. Glynn, K. Watson, M. Skinner, »Professional C# 4 and .NET 4«, Wrox, 2010
- [5] .NET. Dostopno na: http://en.wikipedia.org/wiki/.NET_Framework
- [6] ASP.NET. Dostopno na: http://en.wikipedia.org/wiki/ASP.NET
- [7] B.Evjen, S.Hanselman, D.Rader, »Professional ASP.NET 4 in C# and VB«, Wrox, 2010
- [8] M. MacDonald, A.Freeman, M.Szpuszta, »Pro ASP.NET 4 in C# 2010«, Fourth Edition, Apress, 2010
- [9] Ajax. Dostopno na: http://en.wikipedia.org/wiki/Ajax_%28programming%29
- [10] A. Gallo, D.Barkol, R.K. Vavilal, "Asp.NET Ajax in Action" , 2007
- [11] Telerik ASP.NET Ajax. Dostopna na: http://www.telerik.com/products/aspnet-ajax.aspx
- [12] Telerik Reporting http://www.telerik.com/products/reporting.aspx

- [13] Sistem LDAP. Dostopno na: http://en.wikipedia.org/wiki/LDAP

- [14] Microsoft Active Directory. Dostopno na: http://en.wikipedia.org/wiki/Active_Directory

- [15] Iskanje zapisa v LDAP. Dostopno na: http://docs.allardsoft.com/filetransfer/advanced_ldap

- [16] Dnevniška datoteka. Dostopno na: http://logging.apache.org/log4net/

- [17] Kontrola RadEditor. Dostopno na: http://www.telerik.com/help/aspnet-ajax/editor-overview.html

- [18] Kontrola RadAsyncUpload. Dostopno na: http://demos.telerik.com/aspnet-ajax/asyncupload/examples/overview/defaultcs.aspx?product=asyncupload

- [19] A Troelsen, »Pro C# 2010 And The .NET 4.0 Platform«, 5th Edition, Apress, 2010

- [20] Kontrola RadGrid. Dostopno na: http://demos.telerik.com/aspnet-ajax/grid/examples/overview/defaultcs.aspx

- [21] Kontrola RadAjaxLoadingPanel. Dostopno na: http://demos.telerik.com/aspnet-ajax/xmlhttppanel/examples/loadingpanelid/defaultcs.aspx

- Kontrola RadChart. Dostopno na: http://demos.telerik.com/aspnet-ajax/chart/examples/overview/defaultcs.aspx

- Kontrola ReportViewer. Dostopno na: http://www.telerik.com/help/reporting/asp-net-report-viewer-overview.html

- Velikost sklada strežnika IIS. Dostopno na: http://blogs.msdn.com/b/tom/archive/2008/03/31/stack-sizes-in-iis-affects-asp-net.aspx