UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Klemen Kočila
Vpisna številka: 63020079

Urejevalnik dokumentov in povezav IUS-TIME

DIPLOMSKA DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: prof. dr. Saša Divjak

Ljubljana, 2010

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: 00022/2010
Datum: 01.10.2010

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

Kandidat: **KLEMEN KOČILA**

Naslov: **UREJEVALNIK DOKUMENTOV IN POVEZAV IUS-TIME**
IUS-TIME DOCUMENTS EDITING TOOL

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V diplomski nalogi obravnavajte razvoj aplikacije za urejanje dokumentov. Predstavite večslojno okolje, v katerem deluje taka aplikacija. Najvišji sloj naj obravnava prikaz urejenih podatkov na spletnih portalih, vmesni sloj predstavlja sam aplikacijski strežnik . Na najnižjem sloju je podatkovna baza. Podrobneje predstavite le del okolja, ki je pomemben za aplikacijo. Podajte zahteve, ki jih mora aplikacija izpolnjevati. Predstavite implementacijo aplikacije.

Mentor:

prof. dr. Saša Divjak

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

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

diplomskega dela

Spodaj podpisani/-a **Klemen Kočila**,

z vpisno številko **63020079**,

sem avtor/-ica diplomskega dela z naslovom:

Urejevalnik dokumentov in povezav IUS-TIME

S svojim podpisom zagotavljam, da:

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

  prof. dr. Saša Divjaka

- 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 12.11.2010 Podpis avtorja/-ice: _____

Zahvala

Za pomoč pri izdelovanju diplomske naloge bi se rad zahvalil vsem sodelavcem podjetja IUS SOFTWARE, predvsem razvojni ekipi, ki jo vodi Roman Zvonar, ter mentorju, prof. dr. Saši Divjaku.

Kazalo

1 Povzetek

Diplomska naloga obravnava razvoj aplikacije za urejanje dokumentov podjetja IUS SOFTWARE. V tretjem poglavju predstavlja večslojno okolje, v katerem deluje aplikacija: najvišji sloj (prikaz urejenih podatkov na spletnih portalih), vmesni aplikacijski strežnik in najnižji sloj (podatkovna baza). Podrobneje je predstavljen le del okolja, ki je pomemben za aplikacijo. V četrtem poglavju govorimo o zahtevah, ki jih mora aplikacija izpolnjevati. Peto poglavje pa predstavlja implementacijo aplikacije. Za samo predstavitev je uporabljena končna podoba naloge, saj je tako bralcu lažje nazorno prikazati vse sestavne dele; npr. vpliv okolja na realizacijo določenih vmesniških elementov. Implementacija tekstovno in slikovno razlaga uporabniški vmesnik (sporedi, okna, večjezičnost), procese, ki se izvajajo v ozadju, vtičnike ter končno namestitev aplikacije pri uporabnikih.

2 Uvod

»The function of good software is to make the complex appear to be simple.«
(Grady Booch)

Tema te diplomske naloge je aplikacija družbe IUS SOFTWARE. Družba se ukvarja z:

- oblikovanjem, razvojem, upravljanjem, vzdrževanjem in distribucijo podatkovnih baz pravne in poslovne narave,
- razvojem informacijsko-tehnoloških in programskih rešitev za povezovalne sisteme. Le-ti omogočajo celovito upravljanje z vsebinami nestrukturiranih dokumentov (besedila, upravljanje z vsebinami, upravljanje z dokumenti).

Poglavitne dejavnosti družbe IUS SOFTWARE so spletni portali:

- Portal IUS-INFO, pravni in poslovni informacijski portal, je komercialna baza podatkov na spletu, ki vsebuje največji izbor pravnih informacij iz Slovenije in Evropske unije na enem mestu (http://www.iusinfo.si).
- Portal FinD-INFO, finance in davki; je najsodobnejši portal, ki temelji na tehnologiji IUS-INFO. Portal FinD-INFO ponuja celovit pregled finančnih, računovodskih, davčnih in pravnih informacij ter omogoča enostavno iskanje podlag za rešitev strokovnih vprašanj (http://www.findinfo.si).
- Hrvaški portal IUS-INFO, pravni in poslovni informacijski portal, je komercialna baza podatkov na spletu, ki vsebuje največji izbor hrvaških pravnih informacij in pravnih informacij Evropske unije na enem mestu (http://www.iusinfo.hr).

Celoten sistem spletnih portalov je zasnovan na enotni arhitekturi in mora delovati v večjezičnem okolju.
Diplomska naloga obravnava del tega sistema, in sicer aplikacijo za enotno urejanje dokumentov v sistemih IUS-INFO, ki poenostavi in izboljša proces urejanja dokumentov.
Različne vrste dokumentov je potrebno najprej predelati in razdelati v pravo obliko, tako da jih je moč vnesti v bazo IUS-TIME. V tej bazi je en dokument razdeljen na verzije in vsaka verzija na segmente (manjše, samostojne dele teksta). Vsak dokument ima lahko tudi več povezav (npr. povezava iz objave na zakon). Vsak tip segmenta ima svoj format (če je npr. tip segmenta naslov, potem ima večjo velikost pisave, sredinsko poravnavo in odebeljen tekst). K tipom segmenta se štejejo tudi šifranti, ki so v tej bazi povsem ločeni, v segment pa je zapisana identifikacijska številka uporabljenega šifranta. Predstavljena je osnova za tekste dokumentov. Poleg osnove urejevalci potrebujejo še:

- urejanje teksta z možnostjo oblikovanja (npr. poševni tekst, odebeljeni tekst),
- primerjalnik teksta,
- dodajanje prilog,
- predogled dokumenta itd.

Vse te zahteve upošteva tema te diplomske naloge, naša aplikacija za urejanje dokumentov IUS-TIME.

Aplikacijo smo razvili v Microsoftovem razvojnem okolju .NET, ki je že dobro uveljavljen. Bolj natančno, uporabili smo WPF (Windows Presentation Foundation) [1,2], ki je prisoten od ogrodja .NET verzije 3.0. Glavni razlog je enostavnejši in lepši prikaz uporabniškega vmesnika. To je pomemben faktor pri aplikacijah, katerih ciljna skupina ljudi je zelo široka – praktično celotna družba. Za razvoj je bilo uporabljeno orodje Microsoft Visual Studio 2008.

Pojasnim naj še, zakaj smo za aplikacijo izbrali ime *Stručko*. Pred *Stručkom* se je uporabljala aplikacija *Strukturni Editor*, ki je bila precej nepraktična in je imela le osnovne funkcije. Potrebovali smo dopolnitev oz. osvežitev in spomnili smo se na besedo 'stručko', ki jo večkrat uporabljamo kot opis osebe, ki nekaj obvlada, ime pa še vedno spominja na *Strukturni Editor*.

3 Pregled okolja

Preden začnemo razmišljati o aplikaciji, je potrebna predstavitev okolja, v katerem deluje.

3.1 Spletni portal IUS-INFO

Portal je končni izdelek družbe IUS SOFTWARE. Za končni izdelek pa je potrebnih več faz, skozi katere potujejo dokumenti:

1) prejetje dokumentov (v pisni ali elektronski obliki)
2) računalniška (večkrat avtomatska) obdelava strukturiranih ali nestrukturiranih dokumentov, kot priprava za uvoz v bazo
3) uvoz v bazo
4) urejanje dokumentov in povezav med njimi
5) publiciranje (kopiranje v publikacijsko bazo in indeksiranje v iskalnik)

Stručko se osredotoča predvsem na četrto točko, z dodatnimi vtičniki pa sta možni tudi točki 2 in 3. Več o tem je napisanega v poglavju 5.3.

Prednosti našega portala v primerjavi s konkurenčnim so:

- čistopisi zakonov,
- povezave med dokumenti,
- hitro in kvalitetno iskanje po dokumentih.

Za dosego čistopisov ne gre brez urejevalnika in urejevalca. To delo zahteva izurjenega pravnika in dobro orodje. Povezave med dokumenti so večinoma določene avtomatsko pri uvozu v bazo, vendar ne pri vseh tipih dokumentov. Tam je spet potrebno ročno delo pravnika in pravega orodja. Za iskanje po dokumentih se uporablja sistem Verity K2, ki je delo družbe Autonomy (http://www.autonomy.com/).

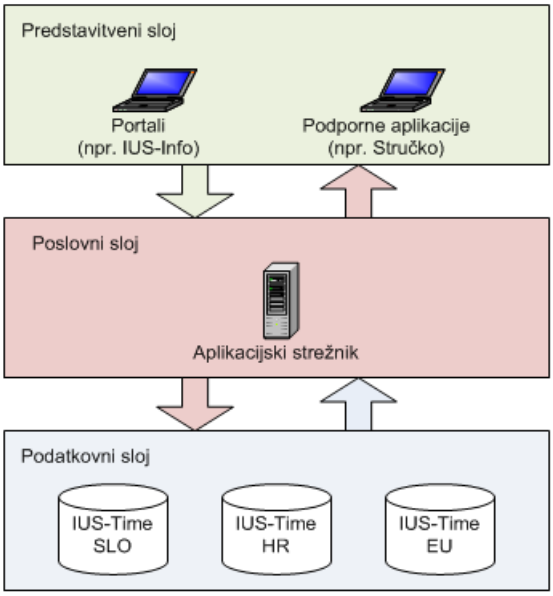
Portal je razdeljen na dva dela. Prvi del je javen in brezplačen, to so: novice, članki, kolumne, napovednik dogodkov, pripomočki; drugi pa zaprt in plačljiv, to so: zakonodajni dokumenti, sodna praksa, strokovna literatura. Portal med drugim omogoča pregled vseh časovno ločenih verzij (od tod ime IUS-TIME), pogled po kazalih posameznih skupin dokumentov, forum PRAVNIKI, izračun obresti, seznam koristnih povezav, seznam odvetnikov.

3.2 Arhitektura sistema IUS-TIME

Kot je razvidno iz slike 1, je arhitektura ločena na tri glavne sloje:

- podatkovni sloj,
- poslovni sloj in
- predstavitveni sloj.

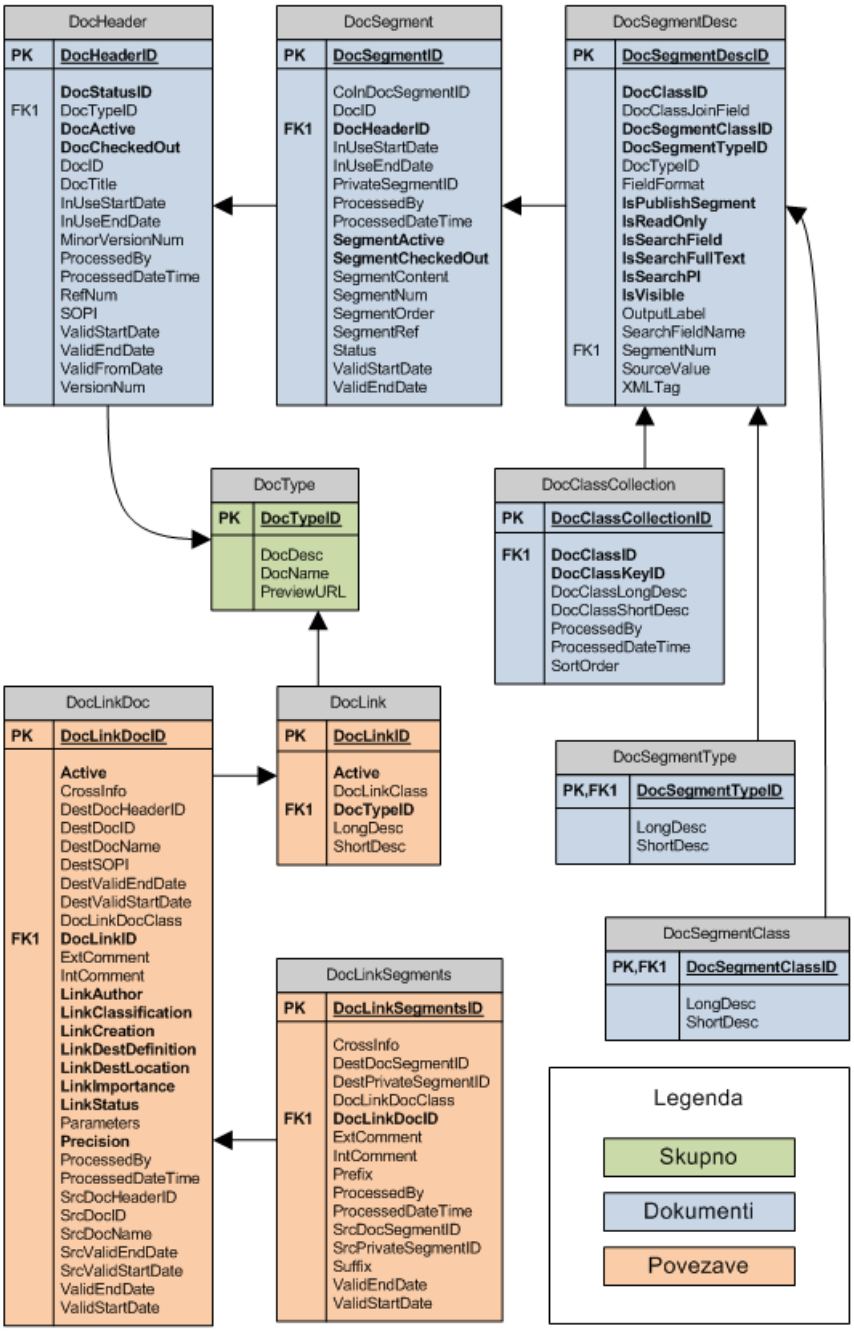
Podatkovni sloj hrani vse podatke (dokumente, povezave, šifrante itd.). V ta podatkovni sloj dostopa poslovni sloj, ki služi kot vmesnik med podatki in strankami. Stranke (npr. *Stručko*, portal IUS-INFO) pa uporabljajo poslovni sloj kot glavno entiteto za dostop do podatkov. Poslovni sloj vsebuje funkcije za enostavnost in enotnost dostopa do podatkov. Portal recimo, potrebuje samo branje, medtem ko *Stručko* potrebuje poleg branja tudi pisanje. Portal potrebuje samo informacijo o tipu segmenta in njegov tekst, vse v pravem vrstnem redu, *Stručko* pa potrebuje veliko več, npr. datumski razpon, identifikacijske številke, podatek, ali je segment samo za branje itd. Ločimo torej med portalskimi (bralnimi) funkcijami in funkcijami za urejanje. Več o tem v poglavju 3.4.



Slika 1: Arhitektura sistema IUS-TIME

3.3 Podatkovna baza IUS-TIME

Je osnova za vse dokumente, njihove lastnosti in povezave med njimi. Baza je kompleksna, zato se bomo omejili le na dele, ki so pomembni za aplikacijo *Stručko*. Na sliki 2 torej prikazujemo del baze, ki ga uporablja *Stručko*.



Slika 2: Shema podatkovne baze IUS-TIME

3.3.1 Dokumenti

Izhajamo iz tabele *DocType*, kjer so navedeni tipi dokumentov. Polje *DocName* se navadno uporablja kot referenca izven baze, v njem bi pisalo npr. 'ZAKONI'. Primarni ključ, *DocTypeID*, pa je referenca za tipe dokumentov v bazi. Tu je še eno polje, *PreviewURL*, iz katerega se gradi naslov URL za predogled tega tipa dokumenta.

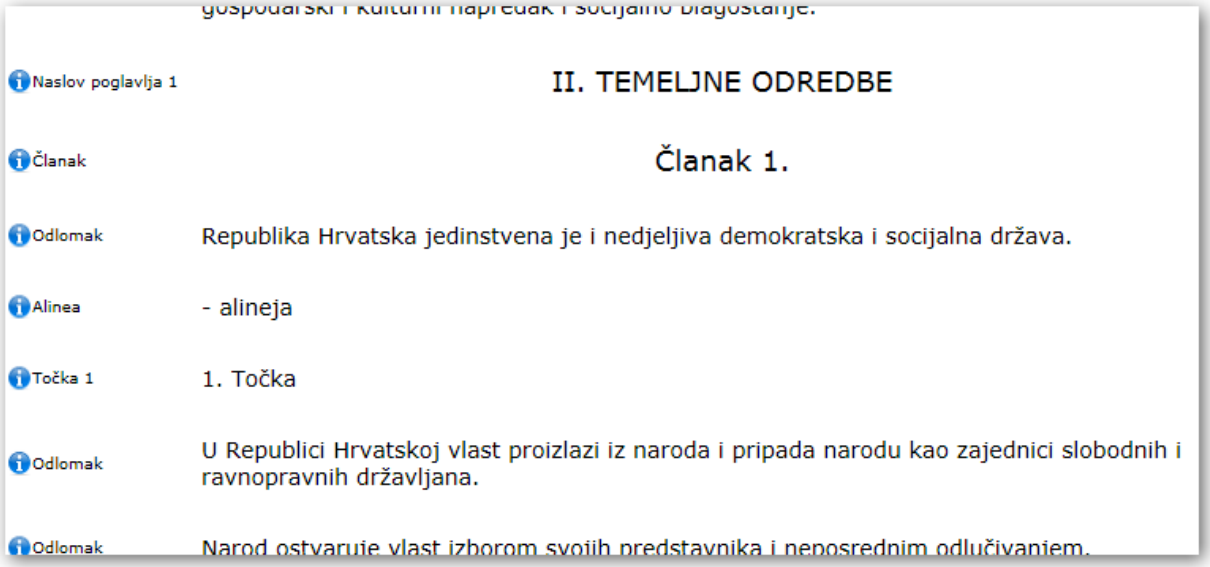
Tabela *DocHeader* vsebuje glavne podatke o verziji dokumenta, npr. tip dokumenta (*DocTypeID*), zunanji identifikator dokumenta (*SOPI*), začetni in končni datum veljavnosti (*ValidStartDate*, *ValidEndDate*), naslov (*DocTitle*), status dokumenta (*DocStatus*, npr. 'V

bazi', 'V urejanju', 'Urejen', itd.), ali je verzija aktivna (*DocActive*), ali je verzija trenutno v urejanju (*DocCheckedOut*). *DocID* je identifikator dokumenta v bazi, s tem poljem se povezuje dokumente z njihovimi segmenti.

DocSegment vsebuje dele/tekste enega dokumenta oz. ene verzije dokumenta. En segment lahko velja samo za določeno verzijo ali pa se razteguje čez več verzij (lahko tudi čez celoten dokument). To je določeno z datumom začetka oz. konca veljavnosti segmenta (*ValidStartDate*, *ValidEndDate*). Vsak segment vsebuje še zaporedno številko (*SegmentOrder*), njegov identifikator znotraj enega dokumenta (*PrivateSegmentID*) – pomemben pri ugotavljanju sprememb, vsebino (*SegmentContent*), kdo in kdaj je spreminjal segment (*ProcessedBy*, *ProcessedDateTime*), ali je segment sploh aktiven (*SegmentActive*), ali je segment trenutno v urejanju (*SegmentCheckedOut*) in nenazadnje tudi *SegmentNum*, s katerim se povezuje v tabelo *DocSegmentDesc*.

DocSegmentDesc predstavlja tipe segmentov. V njem so zapisani tipi segmenta, na katerega se navezuje (*DocTypeID*), identifikator (*SegmentNum*), ime tipa (*XMLTag*), prikaz imena tipa (*OutputLabel*), oblika segmenta (*FieldFormat*), za kakšno vrsto segmenta gre (*DocSegmentTypeID*, npr. 'Tekst', 'Tabela', 'Datotečna priloga'), razred segmenta (*DocSegmentClassID*, npr. 'Meta', 'Text', 'Hide'). S tem razredom ločujemo med segmenti, ki so opisni podatki o dokumentu, tekstovni podatki ali pa podatki, ki navzven niso vidni in jih potrebujejo samo nekatere obdelovalske aplikacije (npr. izvorna datoteka, iz katere izvira dokument). Ali je segment označen samo za branje, nam pove polje *IsReadOnly*. Polje *IsVisible* pove, če je segment viden – v primeru, da je vseeno razreda 'Meta' ali 'Text'. *IsPublishSegment*, se uporablja za (ne)prenašanje segmentov v publikacijsko bazo. Ta baza je sicer enaka IUS-TIME, vendar je prečiščena, tako da vsebuje samo segmente, ki so vidni navzven. Taka baza ni primerna za urejanje, ampak samo za prikaz, npr. na portalu IUS-INFO. Polja, ki vsebujejo besedo Search (*SearchFieldName*, *IsSearchFullText*, *IsSearchField*, *IsSearchPI*), so namenjena ugotavljanju uporabe v iskalniku Verity K2. Polja *DocClassID*, *DocClassJoinField* in *SourceValue* se uporabljajo za identifikacijo šifrantov, tako da lahko tip segmenta predstavlja šifrant. V takem primeru je v vsebino segmenta zapisan identifikator šifranta. Z *DocClassID* se pove tip šifranta (npr. 'Občine Republike Slovenije'), če je polje prazno oz. ima vrednost 0, pomeni, da ta tip segmenta ni šifrant. S poljem *DocClassJoinField* se pove, preko katerega polja v šifrantu se povezuje vsebina segmenta in šifrant, npr. če imamo v tabeli *DocClassCollection* zapisano občino, katere *DocClasKeyID* = '123', *DocClassShortDesc* = '1000' in *DocClassLongDesc* = 'Ljubljana', izbiramo, preko katerega od teh treh polj bomo uporabili za povezavo šifranta z vsebino segmenta – ali bo to poljuben identifikator v bazi ali splošen identifikator občine ali pa kar ime občine. Privzeto se uporablja kar naš identifikator (*DocClasKeyID*), saj edini zagotavlja unikatnost šifranta. Pri občinah bi se seveda lahko uporabil tudi splošni identifikator občine, v takem primeru bi v polje *DocClassJoinField* vnesli 'DocClassShortDesc'. Ostane nam še

SourceValue, ki pa se uporablja samo za prikaz šifranta, torej, iz katerega polja se vzame tekst. V zgornjem primeru bi v to polje vpisali 'DocClassLongDesc', tako da bi se prikazovalo ime občine.



Slika 3: Primer razdelitve dokumenta

3.3.2 Povezave

Povezave so pomembna dodana vrednost portala IUS-INFO in posledično tudi sestavni del procesov v družbi.

Tabela *DocLink* je splošni podatek o tipu povezave. Vsebuje primarni ključ (*DocLinkID*) za povezavo z *DocLinkDoc*, kratek opis (*ShortDesc*) in daljši opis (*LongDesc*). Kratki opisi so tipično enobesedni oz. brez presledkov.

V *DocLinkDoc* se vpisuje povezave med dokumenti. Tu ločujemo dva tipa povezav:

- trda povezava – s primarnimi ključi dokumentov oz. verzij dokumentov (*DocID*, *DocHeaderID*),
- mehka povezava – z zunanjimi identifikatorji dokumentov (*DocName*, *SOPI*) oz. njihovih verzij, če so izpolnjena polja z datumsko veljavnostjo izvornih ali ponornih dokumentov.

Mehka povezava je v našem primeru urejanja boljša, saj omogoča spremembe dokumentov brez skrbi, da bi izgubili podatke o povezavi. Trda povezava pa se večinoma uporablja v publikacijski bazi, kjer urejanje ni več omogočeno, s tem tudi pospešimo bralne funkcije. Izvorna polja (*SrcDocID*, *SrcDocHeaderID*, *SrcDocName*, *SrcSOPI*, *SrcValidStartDate*, *SrcValidEndDate*) skupaj s ponornimi (*DestDocID*, *DestDocHeaderID*, *DestDocName*, *DestSOPI*, *DestValidStartDate*, *DestValidEndDate*) tvorijo eno povezavo med dokumentoma. Taka povezava lahko vsebuje še dodatne podatke, npr. klasifikacija (*LinkClassification*),

pomembnost (*LinkImportance*), avtor (*LinkAuthor*), status (*LinkStatus*), točnost (*Precision*), ali je link aktiven (*Active*) itd.

DocLinkSegments je dodatek k povezavi med dokumenti in pomeni povezave med segmenti. Za kreiranje povezave med segmentoma je obvezna že narejena povezava med dokumentoma, povežeta se s primarnim ključem *DocLinkDocID*. Tako kot pri vrhnjih povezavah tudi tu ločujemo med mehko in trdo povezavo, a s to razliko, da je primarni ključ za trdo vezavo *DocSegmentID*, za mehko pa interni ključ segmenta *PrivateSegmentID*. Veljavnost take povezave se izraža s poljema *ValidStartDate* in *ValidEndDate*.

3.4 Aplikacijski strežnik

Aplikacijski strežnik služi kot vez med aplikacijami in podatki. Dostop do strežnika je mogoč preko t. i. remotinga. V pomoč aplikacijam smo razvili nekaj knjižnic, ki ta dostop zelo poenostavijo. V eni knjižnici (*BusinessEntities.dll*) so zbrane vse entitete, v drugi (*CommonInterfaces.dll*) vsi vmesniki, v ostalih dveh (*DataAccessComponents2.dll* in *ServiceProviders2.dll*) pa so zbrane funkcije za remoting. S konfiguracijsko datoteko (*IUS_SOFTWARE.Client.Services.config*) se določijo poti do strežnikov in profil, ki ga bo aplikacija uporabljala. S profilom se ločujejo baze podatkov (delovne, publikacijske oz. slovenske, hrvaške, evropske itd.), določa iskalni strežnik, pot do prilog itd. Primer konfiguracijske datoteke je na sliki 4.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <applicationServer name="streznik">
        <add key="Profile" value="default"/>
        <add key="Credentials" value="default"/><!-- credentials: default/default -->
        <add key="RemotingType" value="remoting"/><!-- remotingtype: remoting/local/none -->
    </applicationServer>

  <system.runtime.remoting>
        <application>
            <channels>
                <channel ref="http" useDefaultCredentials="true" port="0">
                    <clientProviders>
                        <formatter ref="binary"/>
                    </clientProviders>
                </channel>
            </channels>
            <client>
                <wellknown url="http://streznik/DocumentProvider.rem" type="IUS_SOFTWARE.RemotingServices2.RemoteDocumentProvider, DocumentProvider"/>
                <wellknown url="http://streznik/LinkProvider.rem" type="IUS_SOFTWARE.RemotingServices2.RemoteLinkProvider, LinkProvider"/>
                <wellknown url="http://streznik/ConfigReader.rem" type="IUS_SOFTWARE.RemotingServices2.RemoteConfigReader, ConfigReader"/>
            </client>
        </application>
    </system.runtime.remoting>
</configuration>
```

Slika 4: Primer konfiguracijske datoteke za dostop do aplikacijskega strežnika

Aplikacijski strežnik vsebuje vrsto različnih ponudnikov (angl. Provider) za vrsto različnih aplikacij. Za lažjo predstavo jih bomo nekaj našteli in na kratko opisali:

- DocumentProvider, skrbi za delo z dokumenti (branje, pisanje)
- EditorDocumentProvider, posodobljen DocumentProvider (branje, pisanje, brisanje)
- PortalProvider, portalske funkcije (branje, iskanje)

- LinkProvider, delo s povezavami (branje, pisanje, brisanje)
- EditorLinkProvider, posodobljen LinkProvider (branje, pisanje, brisanje)
- ConfigReader, branje iz konfiguracijske datoteke na aplikacijskem strežniku (branje)

3.4.1 EditorDocumentProvider

Vsebuje sklop funkcij za urejanje dokumentov.

```
public EditorDocumentGetInfo Get(EditorDocumentGetInfo dsIn, int
_maxErrors, string sProcessedBy)
```

Prva in najbolj pomembna je metoda za branje dokumentov. Kot parametre sprejme:

- `EditorDocumentGetInfo dsIn` (*EditorDocumentGetInfo* je entiteta, ki vsebuje vse potrebne podatke za delo z dokumentom, vsebuje tabele *InOutDocument*, *DocType*, *DocHeader*, *DocSegment*, *DocSegmentDesc* ter šifrante, ki so prisotni v *DocSegmentDesc*. V tabeli *InOutDocument* so zapisani osnovni podatki za delo z dokumentom in je edina tabela, ki mora biti izpolnjena. Osnovni podatki so: *DocName*, *SOPI*, *ValidDate*.),
- `int _maxErrors` (število dovoljenih napak, preden se prekine branje dokumenta – privzeto je 0),
- `string sProcessedBy` (domensko ime uporabnika, ki kliče to metodo).

Ti parametri se z enakim pomenom pojavljajo tudi v drugih funkcijah.

V primeru napake funkcija vrne prazno entiteto, razen izpolnjene tabele InOutDocument, kjer je zapisana tudi koda in opis napake. Če ni prišlo do nobene napake, vrne izpolnjeno entiteto.

```
public EditorDocumentGetInfo CheckOut(EditorDocumentGetInfo dsIn, int
_maxErrors, string sProcessedBy)
```

CheckOut funkcija pripravi dokument za urejanje. V bazo (tabelo *DocHeader*) se zapiše, kdo ga ureja, kdaj ga je zaklenil. S tem se prepreči, da bi dokument urejala dva hkrati. Hkrati se ustvari tudi nova (tehnična) verzija dokumenta (polje *(Minor)VersionNum* se poveča za 1). Razlikujemo med verzijo in tehnično verzijo dokumenta. Verzija dokumenta pomeni nov časovni razpon enega dela dokumenta. Tehnična verzija pa pomeni samo spremembo obstoječega časovnega razpona dokumenta. V primeru, da je dokument zaklenjen (v urejanju), funkcija vrne dokument z napako, v kateri piše, kdo ga trenutno ureja. Napako povzroči tudi dvakratni zaklep istega dokumenta, v tem primeru pa v napaki piše tudi, katera verzija dokumenta je v urejanju.

```
public EditorDocumentGetInfo Update(EditorDocumentGetInfo _dsIn, int
_maxErrors, string sProcessedBy)
```

Ko je bil dokument enkrat zaklenjen za druge, ga je moč posodabljati. Vsaka sprememba segmenta ustvari nov zapis v bazi in razveljavi prejšnjega. Na ta način se beležijo spremembe dokumenta.

```
public EditorDocumentGetInfo CheckIn(EditorDocumentGetInfo _dsIn, int
_maxErrors, string sProcessedBy)
```
Po končani seansi spreminjanja dokumenta se izvede t. i. *CheckIn* – potrditev vseh sprememb v dokumentu. Dokument se zopet odklene drugim uporabnikom.

```
public EditorDocumentGetInfo UndoCheckOut(EditorDocumentGetInfo _dsIn, int
_maxErrors, string sProcessedBy)
```
Včasih se zgodi, da je bil *CheckOut* nepotreben ali celo klican po pomoti, tu priskoči na pomoč razveljavitvena funkcija. Ta funkcija razveljavi vse spremembe v zadnji seansi in dokument nazaj odklene drugim, kot da se ne bi nič zgodilo.

```
public EditorDocumentGetInfo DestroyLastTechVersion(EditorDocumentGetInfo
dsIn, int _maxErrors)
```
Funkcija je namenjena razveljavitvi zadnje seanse sprememb trenutno odprte verzije dokumenta. Deluje podobno kot *UndoCheckOut*, le da dokument ni v urejanju.

```
public bool SetStatus(string DocName, string SOPI, DateTime ValidDate,
string NewStatusShortDesc)
```
Podporna funkcija za hitro spreminjanje statusa ene verzije. Kot parametre dobi vse tri zunanje identifikatorje ene verzije dokumenta in nov status. Vrne true/false, če je bila funkcija pravilno/nepravilno izvedena.

```
public DocumentInsertInfo AddNewVersion(string _sProfile,
DocumentInsertInfo _doc, int _maxErrors)
```
V bazo vpiše nov dokument, če še ne obstaja, če pa že obstaja, pa temu dokumentu doda novo verzijo. Funkcija je zmožna tudi vrivati nove verzije, v takem primeru je potrebno v vhodni parameter _doc zapisati številko verzije, ki se vriva.

```
public DataSet GetList(string _sProfile, string DocNames, string Fields,
string Where, string SortOrder, int StartRow, int Length, bool
LastVersions)
```
Napredna funkcija, pri kateri mora razvijalec dobro poznati strukturo baze. Uporabljamo jo lahko npr. za seznam dokumentov ali pa seznam verzij enega dokumenta.

3.4.2 EditorLinkProvider

Vsebuje funkcije za urejanje povezav. Pri povezavah je situacija precej bolj enostavna, saj ne beležimo sprememb, ampak kar spreminjamo/brišemo že obstoječe.

```
public EditorLinkInfo ListLinks(EditorLinkInfo dsIn, int _maxErrors)
```
Funkcija vrne seznam povezav glede na vhodne podatke. EditorLinkInfo je entiteta, ki vsebuje tabele *DocLink*, *DocLinkDoc*, *DocLinkSegments* in *InOutLink*. *InOutLink* je

pomembna povezava, v kateri so zapisani osnovni podatki za pridobivanje seznama povezav, pri vračanju pa je v primeru napake notri zapisana tudi njena koda in opis.

```
public EditorLinkInfo Update(EditorLinkInfo dsIn, int _maxErrors, string
sProcessedBy)
```

Posodabljanje povezav v bazi. Če povezava še ne obstaja, se jo kreira, če je zbrisana, pa zbriše.

```
public string EnhanceTextWithLinks(string sProfile, string sText, string
sUserName, string sDocLinkShortDesc, string sSourceSOPI, string
sSourceDocName, DateTime dtSourceVSD, int iLinkClassification)
```

Funkcija, ki vhodni tekst (parameter sText) opremi s povezavami glede na druge parametre.

4 Aplikacija za urejanje dokumentov in povezav med njimi

Velika količina različnih vrst dokumentov zahteva močno orodje. Do sedaj smo imeli za vsako vrsto dokumenta svojo aplikacijo ali celo več aplikacij. V naših bazah je trenutno 143 vrst dokumentov, vsak s svojo strukturo in svojimi posebnostmi. Izdelati tako aplikacijo, ki bi lahko pokrila vse vrste, bi pomenilo tudi spremembe pri procesih. Zato smo se odločili, da aplikacijo najprej razvijemo za hrvaške sodelavce, saj so ravno začeli z delom in še nimajo velikega števila različnih vrst dokumentov, kot tudi ne procesov, ki bi jih bilo potrebno spreminjati. Kasneje pa že preizkušeno aplikacijo postopoma uvedemo tudi pri nas. In odločitev za projekt *Stručko* je bila sprejeta.

4.1 Funkcionalne zahteve

Aplikacija mora podpirati osnovne funkcije za delo z dokumenti in povezavami med njimi, saj je to tudi njen namen. Omogočiti mora:

- večjo produktivnost (hitrost, pregled, enostavnost),
- večjo svobodo pri formatiranju teksta, vključno z urejanjem tabel,
- podporo datotečnim in dokumentnim prilogam,
- boljši pregled stanja dokumentov,
- predogled dokumentov na portalih,
- primerjanje teksta,
- razširljivost,
- večjezičnost,
- enostavno namestitev in dostop do namestitve.

5 Implementacija aplikacije za urejanje dokumentov in povezav

5.1 Uporabniški vmesnik

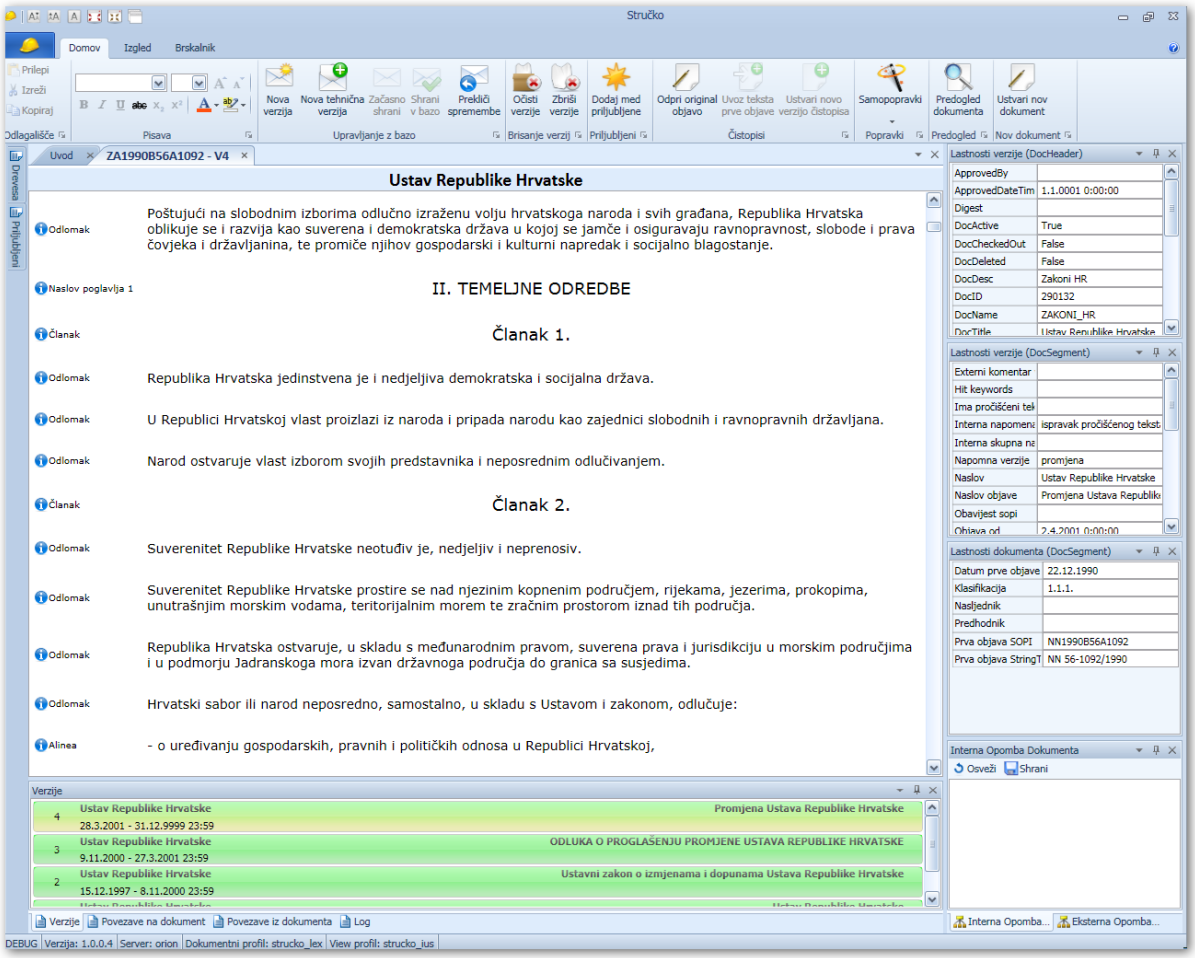
Vse se je začelo pri uporabniškem vmesniku. Da bi pokrili zahteve, bi morala aplikacija imeti uporabniško funkcionalnost razvojnega okolja Microsoft Visual Studio in posodobljen izgled pisarniškega orodja Microsoft Word. Odločili smo se za nakup že razvitih komponent podjetja DevComponents (http://www.devcomponents.com), in sicer 'DotNetBar for WPF', ki omogočajo prav to. Poleg novejšega stila pisarniškega paketa Office ter poljubnega razmetavanja oken nam je paket prinesel še nekaj drugih prednosti:

- koledar s podporo urnikom,
- navigacijski panel,
- komponente za izbiro datumov, realnih ter celih števil,
- panele z navigacijo z jezički,
- drevesno navigacijo, podobno kot je v okolju Windows Vista in Windows 7 ter
- komponente za izbiro barv.

Razvoj podobnega vmesnika bi nam vzel preveč časa in zahteval preveč virov.

Kupljene komponente smo uporabili kot osnovo. Na to osnovno smo dodali še lastna stranska okna (npr. dnevnik, povezave, priljubljene, lastnosti), funkcionalno ločena osrednja okna (uvod, beležka, primerjalnik, dokumentno okno, brskalnik), pomožna okna, večjezičnost, sistem vtičnikov ter vse skupaj zapakirali v namestitveni paket. Vsak od teh si zasluži svojo obrazložitev. Na sliki 5 je primer odprtega dokumenta.

K enostavni uporabi bistveno pripomore sistem za o(ne)mogočanje funkcionalnosti glede na trenutno odprto funkcionalno okno. Če imamo npr. odprto okno za delo z dokumenti, bodo omogočene samo funkcije za delo s tem oknom, ne pa tudi za delo z oknom z beležnico, primerjalnikom ali brskalnikom. Šli smo še malce dlje; in če je npr. en dokument trenutno zaklenjen, bodo omogočene samo funkcije, ki povzročijo njegovo odklepanje. Vsak gumb, ko po določenem času nad njim bedi miškin kazalec, pokaže podroben opis. Ta funkcionalnost velja za čisto vse gumbe v tem sporedu. Čisto vsi gumbi so tudi dostopni preko bližnjic na tipkovnici. S pritiskom na ALT se poleg vsakega gumba prikaže črka (ali več črk), ki predstavlja ta gumb. Tudi ta funkcionalnost je izvzeta iz omenjenega pisarniškega paketa Microsoft Office.

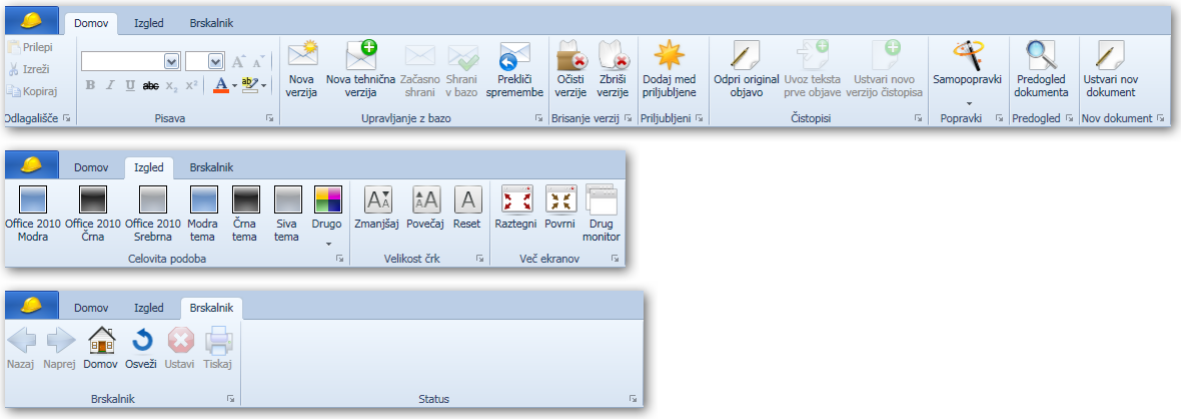


Slika 5: Stručko z odprtim dokumentom

5.1.1 Vrhnji spored

Vrhnji spored (angl. Main Menu) predstavlja največkrat uporabljene funkcije uporabniškega vmesnika. Kot je prikazano na sliki 6, je razdeljen na tri dele:

1) Domov – glavne funkcije,
2) Izgled – funkcije za spreminjanje izgleda aplikacije,
3) Brskalnik – funkcije za upravljanje z brskalnikom.



Slika 6: Vsi trije deli vrhnjega sporeda

5.1.1.1 Domov

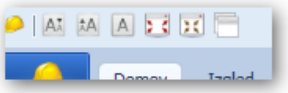
Prvi in privzeti pogled aplikacije predstavlja najbolj uporabljane funkcije. Microsoftova študija (http://blogs.msdn.com/b/jensenh/archive/2006/04/05/568947.aspx) je pokazala, da uporabniki s pogledom največkrat posegajo v levi del sredine. Tja smo namestili funkcije za upravljanje z verzijami, saj so največkrat uporabljane v tem sporedu. Te funkcije so samo drugače imenovane funkcije iz aplikacijskega strežnika (*CheckIn*, *CheckOut*, *Update*), ki so bolj podrobno opisane v poglavju *3.4.1*. Levo od njih se nahajajo kontrolniki za delo z beležnico (*Odlagališče* in *Pisava*), omenjena v poglavju *5.1.2.1*. Desno od razdelka *Upravljanje z bazo* pa se nahajajo še:

- *Brisanje verzij* (omogočajo brisanje samo teksta trenutne verzije – čiščenje ali pa brisanje kar celotne verzije),
- *Priljubljeni* (gumb za dodajanje trenutnega dokumenta v seznam priljubljenih),
- *Čistopisi* (funkcije za delo s čistopisi),
- *Popravki* (seznam vtičnikov za samopopravke, poglavje *5.3.1*),
- *Predogled* (trenutni dokument odpre v osrednjem oknu z brskalnikom, tako kot bi izgledal na portalu) ter
- *Nov dokument* (gumb, ki sproži postopek za ustvarjanje novega dokumenta).

5.1.1.2 Izgled

Za uporabnike je zelo pomemben izgled aplikacije. Pritisk na gumbe v razdelku *Celovita podoba* spremeni celotno barvno podobo aplikacije, torej ne samo glavnega okna, marveč vsa. Privzeta podoba je prva iz leve, *Office 2010 Modra*, ki je zelo podobna privzetemu pogledu na najnovejši različici pisarniškega paketa Microsoft Office 2010. Poleg same podobe je tu možno izbirati še velikost črk v dokumentih (dostopne tudi z bližnjico CTRL + miškino kolo). Desno od črk se nahaja podpora za več zaslonov. Naši uredniki imajo večinoma po vsaj dva zaslona, ker imajo lahko na enem npr. seznam sprememb dokumenta, na drugem pa ga

urejajo. Tako prihranijo veliko časa, ki bi ga sicer porabili za neprestano menjavanje oken. Aplikacija omogoča raztezanje čez zaslone in povrnitev v prejšnje stanje, kot tudi samo prestavitev na drug zaslon. Te funkcije se nahajajo tudi nad vrhnjim sporedom, v skrajno zgornjem levem kotu aplikacije, v hitro dostopnih funkcijah. Te hitro dostopne funkcije je moč urejati in prilagajati po svojih potrebah in željah, te, kot so na sliki 7, so privzete.



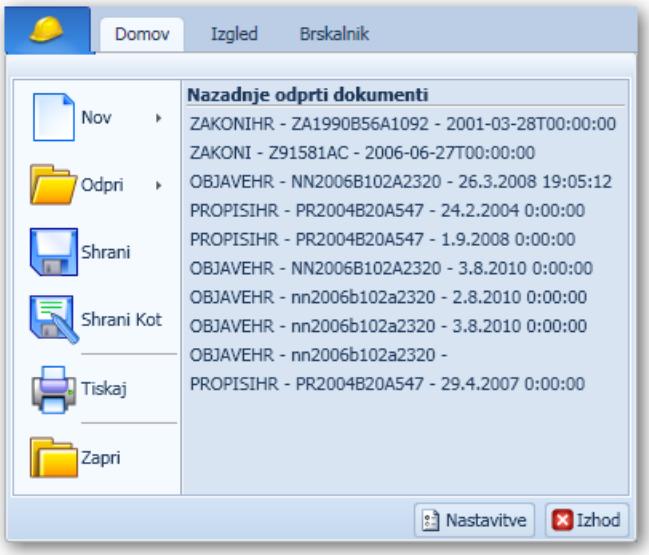
Slika 7: Hitro dostopne funkcije

5.1.1.3 Brskalnik

Zadnji del sporeda je namenjen spletnemu brskalniku. Omogočajo vse osnovne funkcije za brskanje (*Nazaj*, *Naprej*, *Domov*, *Osveži*, *Ustavi* in *Tiskaj*). Gumb *Domov* uporabnika popelje na prvo stran portala. Desno od gumbov se nahaja še Status, kamor se vpisuje trenutni status brskalnika, npr. 'Connecting...'.

5.1.2 Dokumentni spored

Največkrat uporabljani spored znotraj vrhnjega sporeda je vedno datotečni spored. Vendar pa tu nimamo opravka z datotekami, ampak z dokumenti. Na sliki 8 je prikazan razpored, leva stran so funkcije za delo s trenutno odprtim osrednjim oknom ter za nastavitve in izhod iz aplikacije. Privzeta funkcija desne strani je prikaz desetih zadnje odprtih dokumentov, ki pa se spremeni, če z miškinim kazalcem dlje časa stojimo nad levim gumbom, ki vsebuje pod-elemente (gumba *Nov* in *Odpri*). Odpreti je možno datoteko (v beležki) ali pa dokument. Ustvarja se nova okna z brskalnikom, beležko, primerjalnikom ter nov dokument s pomočjo vtičnikov za uvoz (poglavje 5.2.2). Ostanejo še gumbi za shranjevanje (beležke), tiskanje (vseh oken) in zapiranje osrednjih oken.



Slika 8: Dokumentni spored

5.1.3 Stranska okna uporabniškega vmesnika

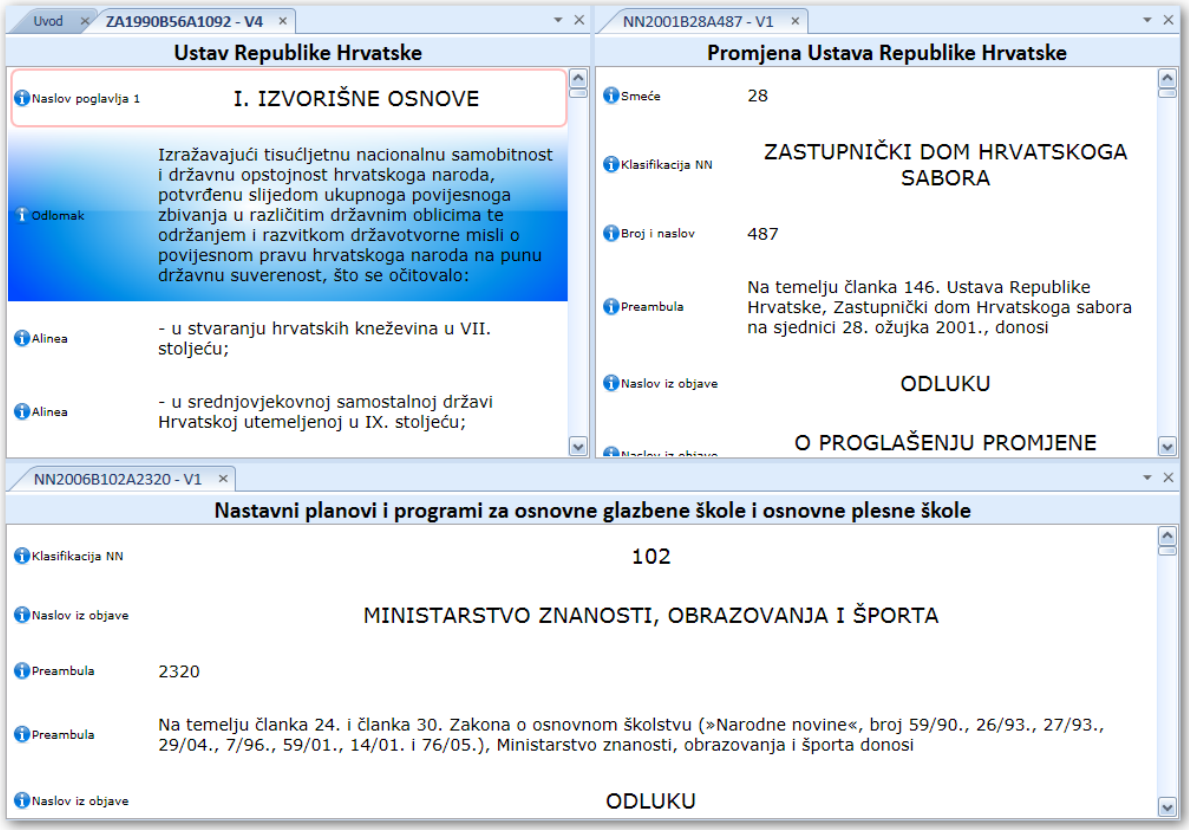
Aplikacija vsebuje vrsto stranskih oken. Vsa okna je moč poljubno premikati po aplikaciji, celo izven nje. Če uporabnik ni prepričan, kje se nahajajo, pritisk na kombinacijo CTRL + TAB prikaže vsa odprta stranska in funkcionalno ločena osrednja okna. Samo klik (ali večkratni pritisk tipke TAB) na enega izmed njih in že je v ospredju. Trenutno omogočena stranska okna so:

- Drevesa (seznami dokumentov, primernih za urejanje, v drevesni strukturi),
- Priljubljeni (seznam priljubljenih dokumentov, tudi v drevesni strukturi, glede na *DocName*),
- Lastnosti verzije (*DocHeader*) (lastnosti verzije, ki so del tabele *DocHeader*),
- Lastnosti verzije (*DocSegment*) (lastnosti verzije, ki so del tabele *DocSegment*),
- Lastnosti dokumenta (*DocSegment*) (lastnosti dokumenta oz. vseh verzij, ki so del tabele *DocSegment*. To so segmenti, ki so preko tabele *DocSegmentClass* kvalificirani kot Meta_doc),
- Interne in eksterne opombe (opombe trenutne verzije za notranje ali zunanje potrebe),
- Verzije (seznam verzij tega dokumenta. Zeleno obarvane so dokončno urejene in primerne za publiciranje, rumena verzija je trenutno odprta, zeleno-rumena pa se pojavi takrat, ko sta oba pogoja izpolnjena),
- Povezave na dokument (seznam povezav na trenutno odprt dokument. Ta seznam je moč urejati – dodajati nove povezave, spreminjati obstoječe ali brisati obstoječe),
- Povezave iz dokumenta (seznam povezav iz trenutno odprtega dokumenta, tudi tega je moč urejati),

- Log (dnevnik bolj pomembnih akcij. Ta dnevnik se v primeru napake pošlje po elektronski pošti do upravitelja aplikacije, za lažje ugotavljanje vzroka).

5.1.4 Funkcionalno ločena osrednja okna

V glavnem, osrednjem delu aplikacije se odpirajo funkcionalno ločena okna. Stručko pozna več tipov oken, vsak tip predstavlja svojo funkcionalnost. Glavna izmed njih je delo z dokumenti, poleg okna z brskalnikom, primerjalnikom in beležko. Vsako okno je lahko odprto večkrat in vsako je možno poljubno premikati po aplikaciji. Osrednji del je možno povsem poljubno razdeliti, npr. na levo in desno stran, tako da je na levi en dokument, na desni pa drug. Še bolj nazoren primer je na sliki 9. Obstaja tudi posebno okno z imenom *Uvod*. To okno je lahko odprto samo enkrat in je privzeto odprto takoj po zagonu aplikacije. V tem oknu so povezave do avtorjeve interne spletne strani, kjer se nahajajo namestitve, elektronski poštni naslov ter zadnje spremembe oz. popravki te aplikacije.

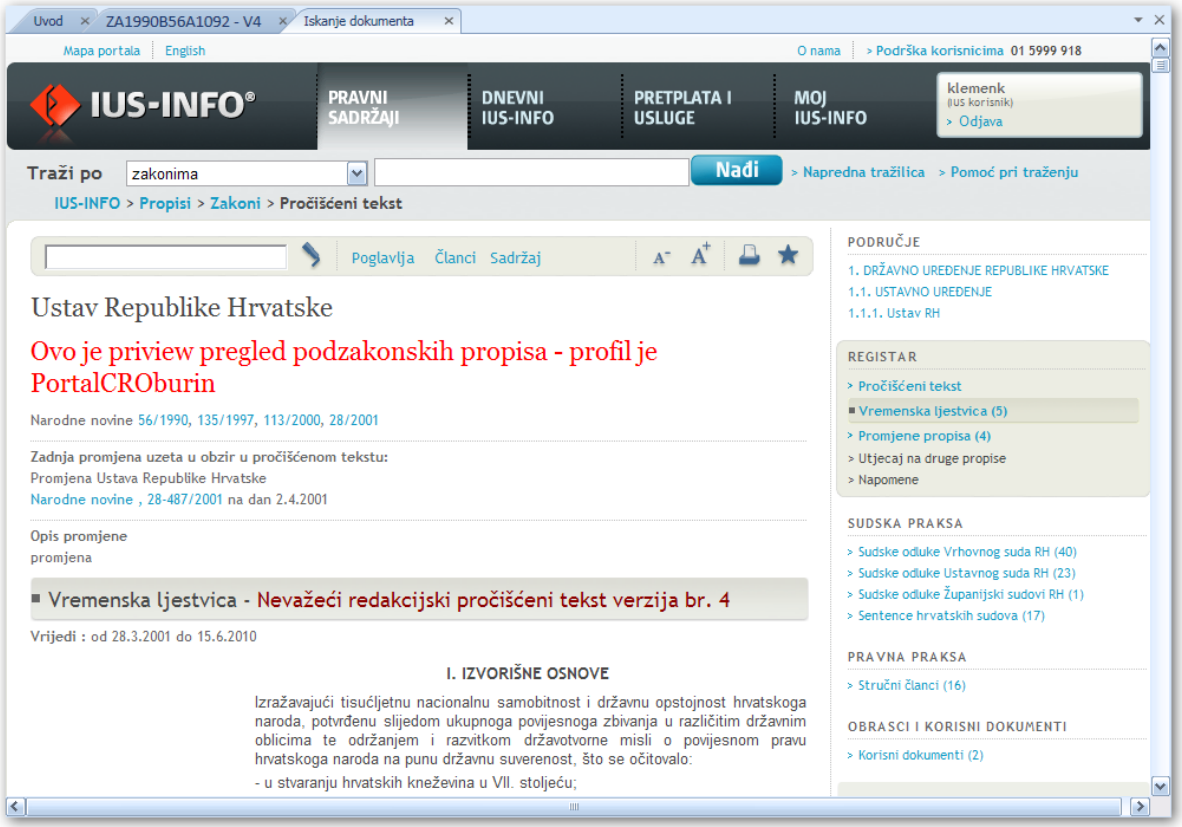


Slika 9: Primer razdelitve osrednjega dela aplikacije

5.1.4.1 IUSBrowserWindow

Okno z brskalnikom ima več namenov. V prvi vrsti je namenjen iskanju dokumentov preko portala. To poteka tako, da uporabnik s kliki po portalu (možno je uporabiti Verity K2

iskalnik) doseže iskani dokument, aplikacija ga zazna in uporabniku ponudi dialog z možnostjo odprtja zaznanega dokumenta. Drugi namen pa je podpora predogledu – klik na gumb Predogled odpre trenutno odprt dokument v tem brskalniku. Osnova brskalnika je Microsoft Internet Explorer.



Slika 10: Okno z brskalnikom z odprto domačo stranjo

5.1.4.2 IUSDocumentWindow

Poanta te aplikacije je delo z dokumenti, zato je to okno najbolj pomembno in mu je bilo posvečenega največ časa in misli. Okno je sestavljeno kot seznam segmentov – t. j. delov dokumenta. Glavni del segmenta je tekst, poleg katerega so pomembni tudi opisni podatki tega segmenta, npr. tip tega segmenta, časovni razpored, stil, povezave itd. En segment je prikazan tako, da ima na levi strani ikono s ključnimi informacijami o tem segmentu (informacije se kažejo, ko uporabnik drži miškin kazalec nad ikono), takoj za njo je tip segmenta, osrednji del zaseda tekst oz. vsebina segmenta, na koncu pa je še prostor za opombe segmenta. V sistemu IUS-TIME poznamo več vrst opomb:

- Klasične opombe (se prikazujejo na portalu),
- Razveljavitvene opombe (razveljavljajo segment z opombo, prikazano na portalu),
- Interne opombe (so namenjene internim zapiskom in se ne prikazujejo izven aplikacije).

Preden je dokument sploh možno urejati, mora biti v stanju urejanja, kar naredi klic funkcije *CheckOut*. Če še ni v stanju urejanja, uporabnik pa želi povzročiti eno izmed akcij, ga aplikacija ustavi in mu poleg opozorila izda možnost *CheckOut*-a.

Urejanje segmentov je možno na več načinov – točke 2, 3 in 4 prikazuje tudi slika 11:

1. dvoklik na en segment,
2. desni klik na tekst enega segmenta,
3. desni klik na tekst več segmentov,
4. desni klik na tip enega ali več segmentov.

Točka 1 povzroči odprtje pomožnega okna za urejanje segmentov, ki je bolj podrobno opisan v poglavju 5.1.5.7. Reakcija točke 2 je prikaz sporeda za poenostavitev dela z označenim segmentom:

- dodajanje novega segmenta pod ali nad označenim segmentom. Po kliku se prikaže okno za urejanje segmenta, tako da ga je moč takoj urediti,
- brisanje označenega segmenta,
- dodajanje ter brisanje opomb (klasičnih, razveljavitvenih in internih),
- prestavljanje segmenta gor/dol,
- združevanje segmenta z zgornjim ali spodnjim segmentom,
- razbijanje segmenta (s pomožnim oknom WindowRazbijanje, poglavje 5.1.5.15),
- rezanje, kopiranje in lepljenje segmenta na/iz odlagališča,
- kopiranje teksta označenega segmenta ter
- kopiranje teksta segmenta na levo ali desno stran primerjalnika (klik na *Desna stran primerjalnika* šele odpre okno in primerja teksta med sabo).

Okno omogoča označitev več segmentov hkrati (tipka SHIFT), s točko 3 (desni klik na tekst več segmentov) se prikaže spored za delo z več segmenti naenkrat:

- združevanje označenih segmentov z več možnostmi presledka tekstov (presledek, nova vrstica, dve novi vrstici),
- brisanje segmentov,
- dodajanje ter brisanje opomb segmentov,
- kopiranje tekstov na odlagališče ali primerjalnik z združevanjem,
- delo z odlagališčem segmentov (rezanje, kopiranje, lepljenje) ter
- množično spreminjanje datumov uporabe.

Z desnim klikom na tip enega ali več označenih segmentov (točka 4) se prikaže seznam vseh tipov, ki jih lahko ima trenutno označeni segment ali več segmentov. Klik na enega od teh tipov spremeni trenutni tip v izbranega.

To okno vsebuje več bližnjic na tipkovnici:

- CTRL+F (iskanje teksta po dokumentu, poglavje 5.1.5.9),
- CTRL+G (seznam vseh številk členov, poglavje 5.1.5.10),
- CTRL+miškino kolo (povečevanje/zmanjševanje velikosti pisave segmentov),

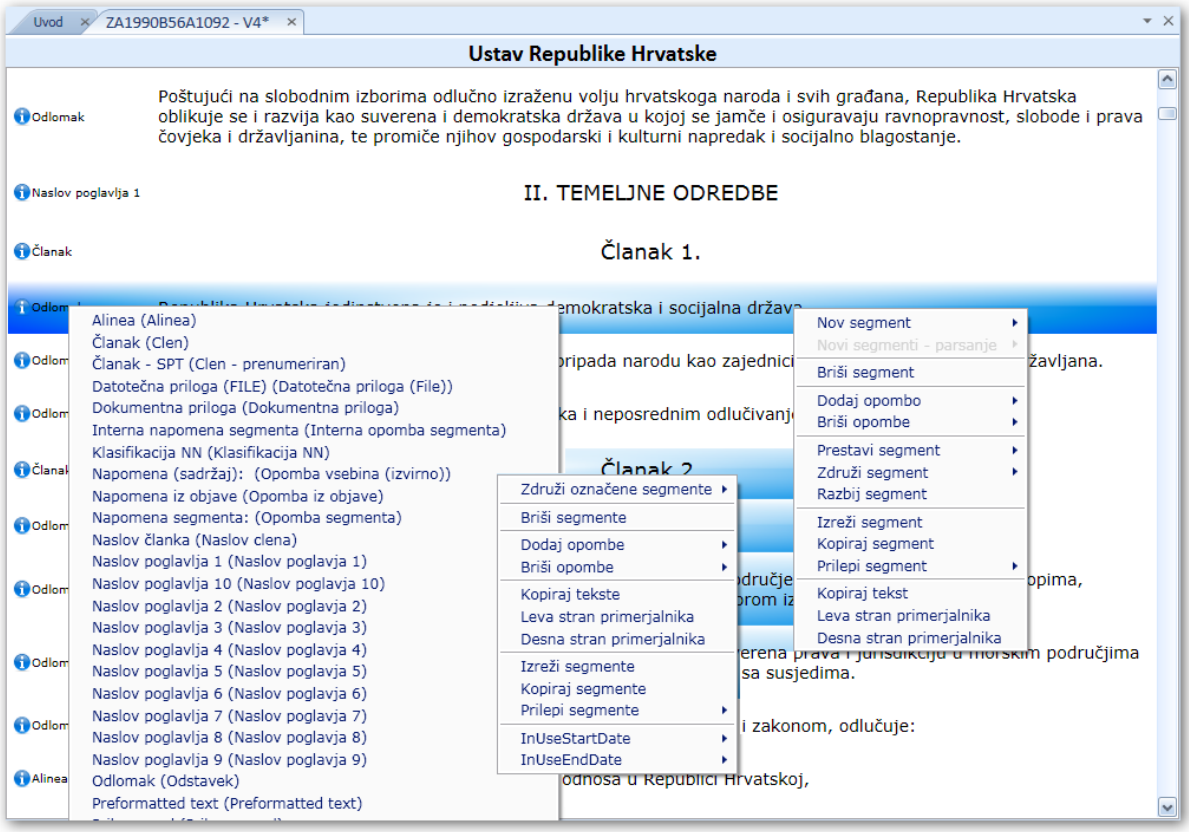
- F2 (povečevalno steklo).

Ima pa tudi nekaj drugih posebnosti. Višina enega segmenta nikoli ni večja od tretjine višine okna. Če je vsebine več, je za večjo preglednost nad vsemi segmenti dana možnost drsenja teksta znotraj enega segmenta. Druga posebnost pa je težava prikazovanja velikih količin segmentov. Obstajajo namreč dokumenti, ki vsebujejo več kot 10.000 segmentov. Prikaz oz. že samo inicializacija vseh bi povzročila sesutje aplikacije (napaka *Out of memory*, http://msdn.microsoft.com/en-us/library/system.outofmemoryexception.aspx), kar se je tudi nam dogajalo. Težavo smo rešili s pomočjo virtualizacije uporabniškega vmesnika (UI Virtualization, [1,2]). Virtualizacija deluje tako, da so dejansko prisotni samo objekti, ki so trenutno vidni uporabniku. V seznamu z nekaj tisoč segmenti je torej zgrajenih samo tistih 20, ki so trenutno vidni. Torej samo teh 20 tudi zaseda računalniški pomnilnik. Kako in kdaj se objekti gradijo oz. uničujejo, pa določata dva možna načina virtualizacije:

- *Container recycling* (Deluje tako, da se elementi seznama ne uničujejo, ampak se uporabijo za naslednje elemente, ki jih nadomeščajo. Če bi se v našem primeru pomaknili en segment nižje, bi se vrhnji segment uporabil za spodnjega, torej bi dobil nov tekst, nov tip, vse nove podatke za gradnjo enega segmenta. Žal pa je bila ta metoda za nas neuspešna, saj se niso prenesli čisto vsi deli segmenta, npr. stili.)
- *Deferred scrolling* (Ko uporabnik uporablja drsnik, se seznam vizualno ne spreminja, vse dokler uporabnik drsnika ne spusti na enem mestu. Po tem se prikažejo vsi segmenti na izbranem mestu. Poleg tega se segmenti uničujejo in gradijo na novo, tako da nimamo težav s stili. S tem načinom je aplikacija bolj odzivna, saj se elementom ni potrebno neprestano spreminjati, ampak samo enkrat. To je način, ki smo ga uporabili tudi mi.)

Kot pomoč pri drsenju, ki se vizualno ne spreminja, smo uporabnikom dali na voljo iskanje po tekstu in skakanje na številko člena, tako da je drsenje učinkoviteje.

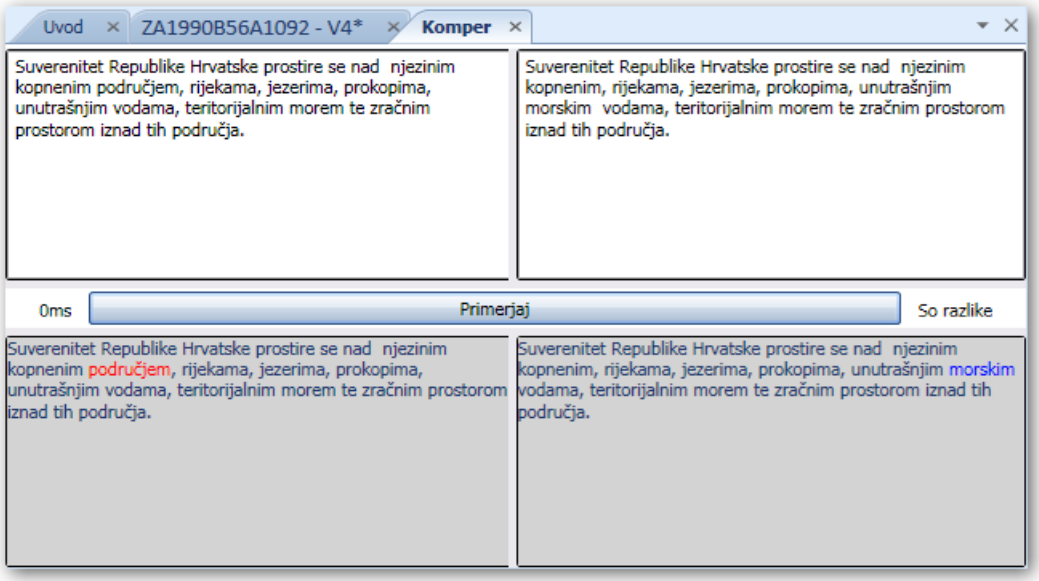
Še ena izmed posebnosti tega okna je zabava uporabnikov. Medtem ko dokument čaka na neko operacijo na aplikacijske strežniku (npr. da se shranjuje), se okno zamegli, v sredini okna se pa pokaže naključno izbran vic. Tako uporabniki laže počakajo tudi nekaj sekund, da se zahtevek izvede.



Slika 11: Okno z odprtim dokumentom

5.1.4.3 IUSKomperWindow

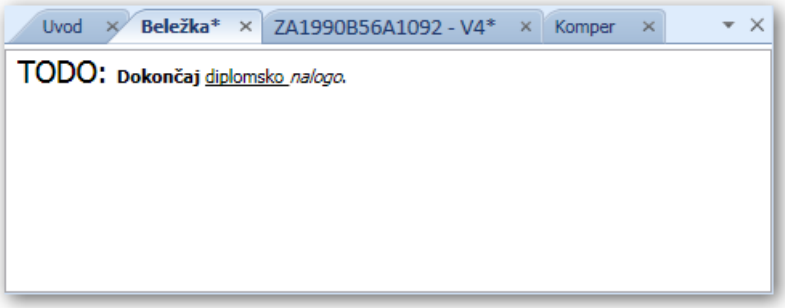
Primerjanje tekstov med sabo je uporabna funkcionalnost pri izdelovanju čistopisov. S takim orodjem se hitro preverja spremembe teksta in točno to omogoča okno *IUSKomperWindow*. Okno je razdeljeno na štiri dele. Leva in desna stran ločujeta teksta, ki se primerjata med sabo, zgornja in spodnja pa prvotni tekst in rezultat primerjanja. Vmes med zgornjo in spodnjo stranjo se nahaja gumb za začetek primerjanja. Leva stran se vedno obravnava kot prva verzija teksta, desna pa kot nova verzija teksta. To pomeni, da so na levi strani lahko samo zbrisane besede, na desni pa samo dodane besede. Bolj podrobno, če je na levi strani beseda, ki je ni na desni, bo na levi strani prikazana kot zbrisana, če pa manjka beseda na levi strani in je na desni, bo na desni označena kot dodana. Primerjalnik deluje po sistemu besed – med tekstoma se primerjajo besede in ne samo posamezni znaki, kot tudi ne samo celotne vrstice. Razlog je povsem logičen. Največkrat se v praksi med sabo primerja samo ena vrstica, torej bi bilo primerjanje po vrsticah neuporabno. Če pa bi bilo narejeno primerjanje po znakih, spremembe ne bi bile dovolj opazne. Besede, ki jih ni na desni strani (zbrisane), so v rezultatu označene z rdečo, tiste, ki pa manjkajo na levi strani (dodane), so v rezultatu označene z zeleno. Sama motorika primerjanja je bila razvita »v hiši«.



Slika 12: Okno s primerjalnikom

5.1.4.4 IUSNotepadWindow

Včasih potrebujemo eno okno za odložitev teksta, ki ga ne bi radi izgubili, ali pa bi si samo radi nekam zapisali, kaj moramo še postoriti. V ta namen je izdelano okno z beležko oz. kar Beležka. Beležka v naši aplikaciji ponuja spreminjanje pisave, spreminjanje stila in velikosti pisave, delo z odlagališčem (angl. Clipboard), shranjevanje v poseben format ('.xfd'), lahko tudi standardno datoteko ('.txt') in odpiranje le-teh.



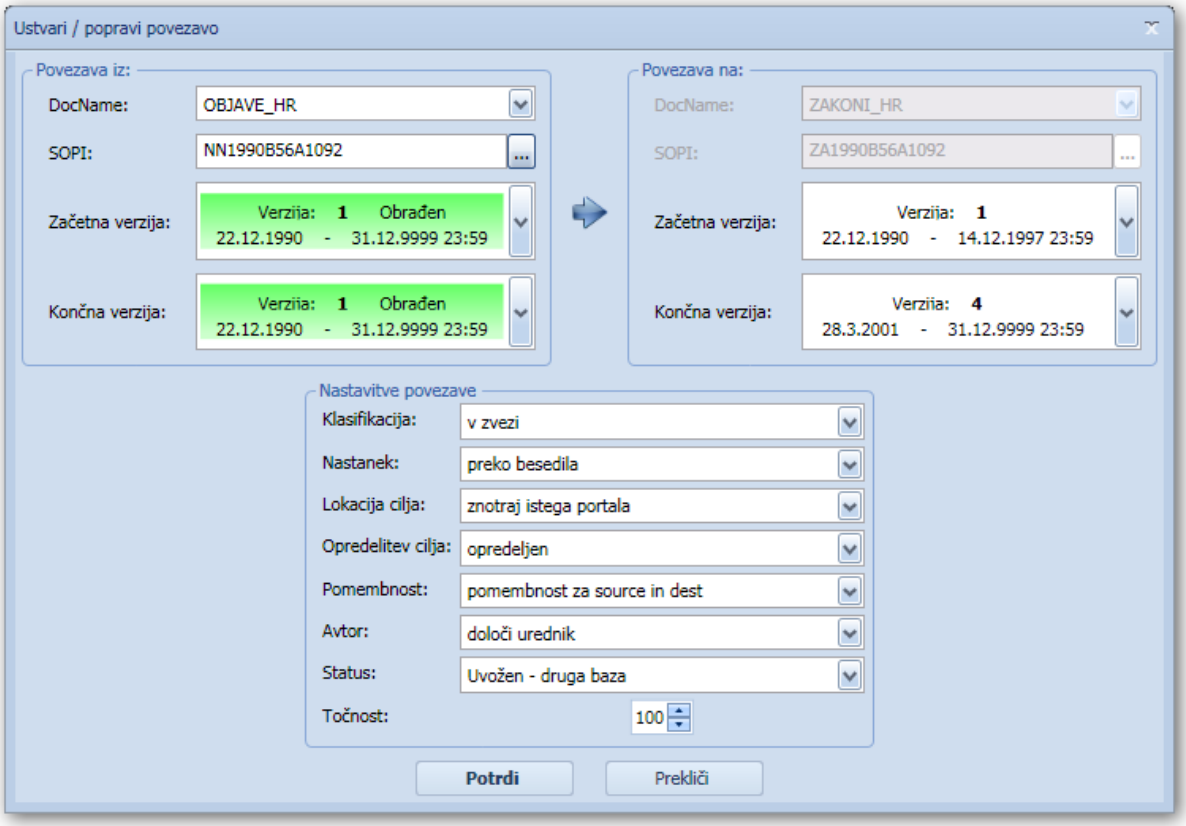
Slika 13: Okno z beležko

5.1.5 Pomožna okna

Pomožna okna so sestavni del vsake večje aplikacije. Razlog je precej enostaven – aplikacije bi bile brez pomožnih oken prenatrpane in prezapletene.

5.1.5.1 WindowCreateEditLink

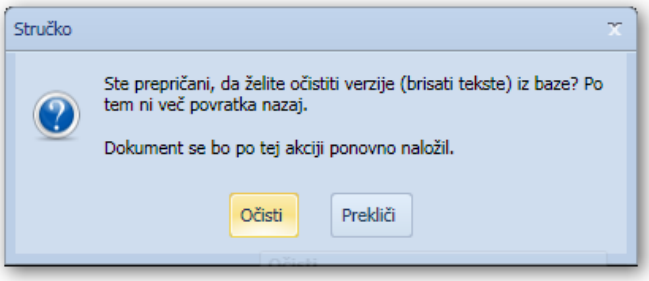
To okno je namenjeno ustvarjanju in urejanju povezav med dokumenti oz. verzijami dokumentov. Zajeta sta oba scenarija, obe smeri povezave, iz trenutnega dokumenta (oz. verzije) v nekega drugega (oz. v neko drugo verzijo) in obratno. Poleg same povezave je tu moč urejati tudi druge podatke o povezavi (natančnost, klasifikacije, avtor itd).



Slika 14: WindowCreateEditLink

5.1.5.2 WindowCustomDialog

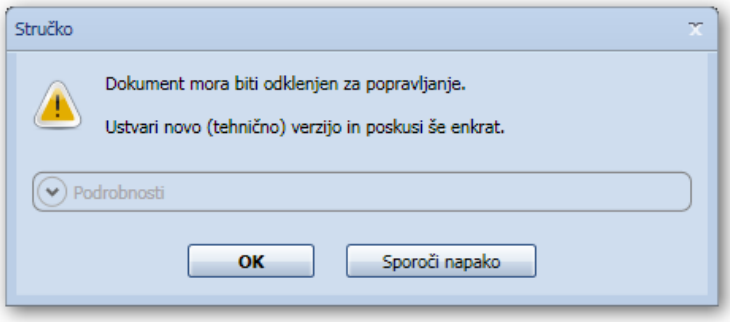
Za to, da bi bila vsa okna konsistentna, je bilo potrebno ustvariti lastno okno za dialog (sporočanje s pričakovanjem povratne informacije). Okno je razdeljeno na tri dele. Zgornji z naslovom, osrednji s prikazom sporočila ter ikonico, ki predstavlja tip tega sporočila (opozorilo, vprašanje, ostalo) in spodnji del z gumbi. Posebnost tega okna je, da se gumbi dodajajo po potrebi. To pomeni, da če aplikacija potrebuje npr. pritrdilni gumb ter odklonilni gumb, razvijalec pri ustavjanju okna za vsak gumb poda tekst in opis gumba. Vsak gumb zapre okno, rezultat dialoga pa je tekst gumba, na katerega je kliknil uporabnik. Na ta način je bila dosežena dobra univerzalnost (lastna selekcija gumbov) ter prijaznost do uporabnika (opis gumba se kaže, ko uporabnik nad gumbom drži miškin kazalec).



Slika 15: WindowCustomDialog

5.1.5.3 WindowCustomMessageBox

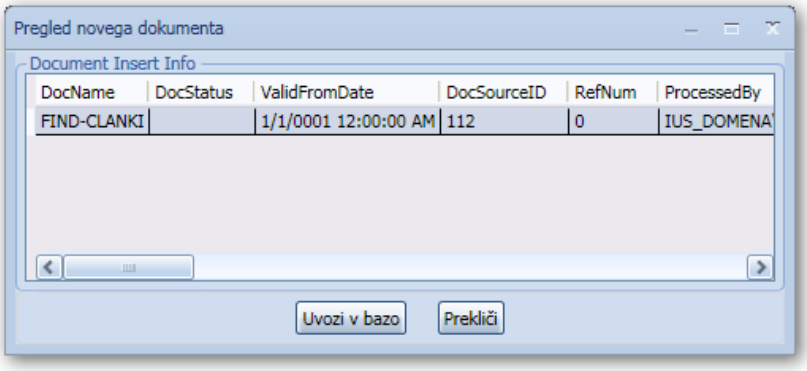
To okno je malce okrnjena verzija okna *WindowCustomDialog*. S tem je mišljeno, da je namenjen le obveščanju, ne pa tudi povratnim informacijam. Zatorej potrebuje le pritrdilni gumb. Ker pa se to okno uporablja tudi za sporočanje napak, je bila dodana še funkcionalnost s sporočanju napak. Poleg samega sporočila napake je v posebnem, prav za to namenjenem razdelku, ki s klikom nanj poveča celotno okno, moč videti tudi njene podrobnosti. Poleg pritrdilnega gumba se nahaja še gumb za sporočanje napak. S tem gumbom se odpre okno *WindowErrorReport*, opisano v poglavju 5.1.5.8.



Slika 16: WindowCustomMessageBox

5.1.5.4 WindowDocInsertInfoCheck

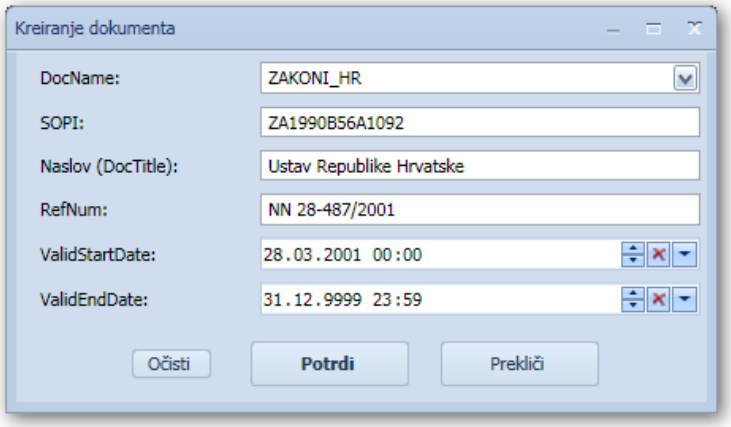
Okno za podporo vtičnikom za uvoz novih dokumentov (poglavje 5.3.2). V osrednjem delu okna se nahaja pregled dokumenta v podrobnem, neoblikovanem pogledu – tabeli *DocHeader* in *DocSegment*. S klikom na gumb *Uvoz v bazo* se dokument uvozi v bazo, s klikom na *Prekliči* pa se okno zapre brez kakršnekoli druge akcije, povezane z bazo.



Slika 17: WindowDocInsertInfoCheck

5.1.5.5 WindowDocument

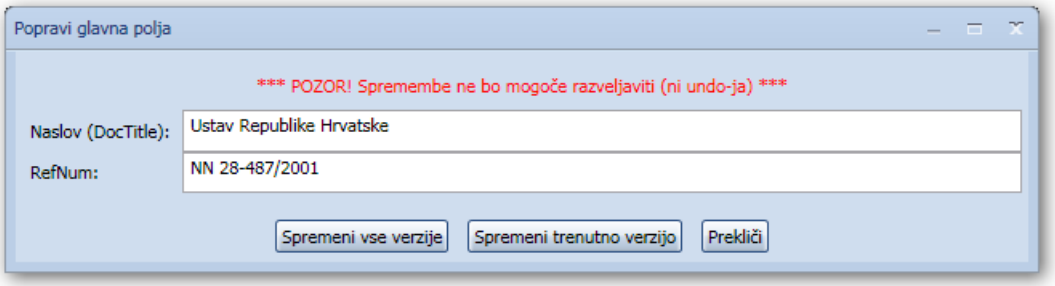
Ustvarjanje novega dokumenta je vrlina tega okna. Za stvaritev je potrebno izpolniti samo osnovne podatke dokumenta in klik na gumb *Potrdi*. Gumb *Očisti* postavi vsa vnosna polja v prvotno stanje (prazno), *Prekliči* pa zapre okno.



Slika 18: WindowDocument

5.1.5.6 WindowEditHeader

Okno za spreminjanje dveh pomembnih podatkov v tabeli *DocHeader*, naslova dokumenta (*DocTitle*) ter referenčnega zapisa dokumenta (*RefNum*). Okno ponuja dve možnosti, lahko spremenimo vse verzije dokumenta ali pa samo trenutno ter seveda prekličemo oz. zapremo okno.



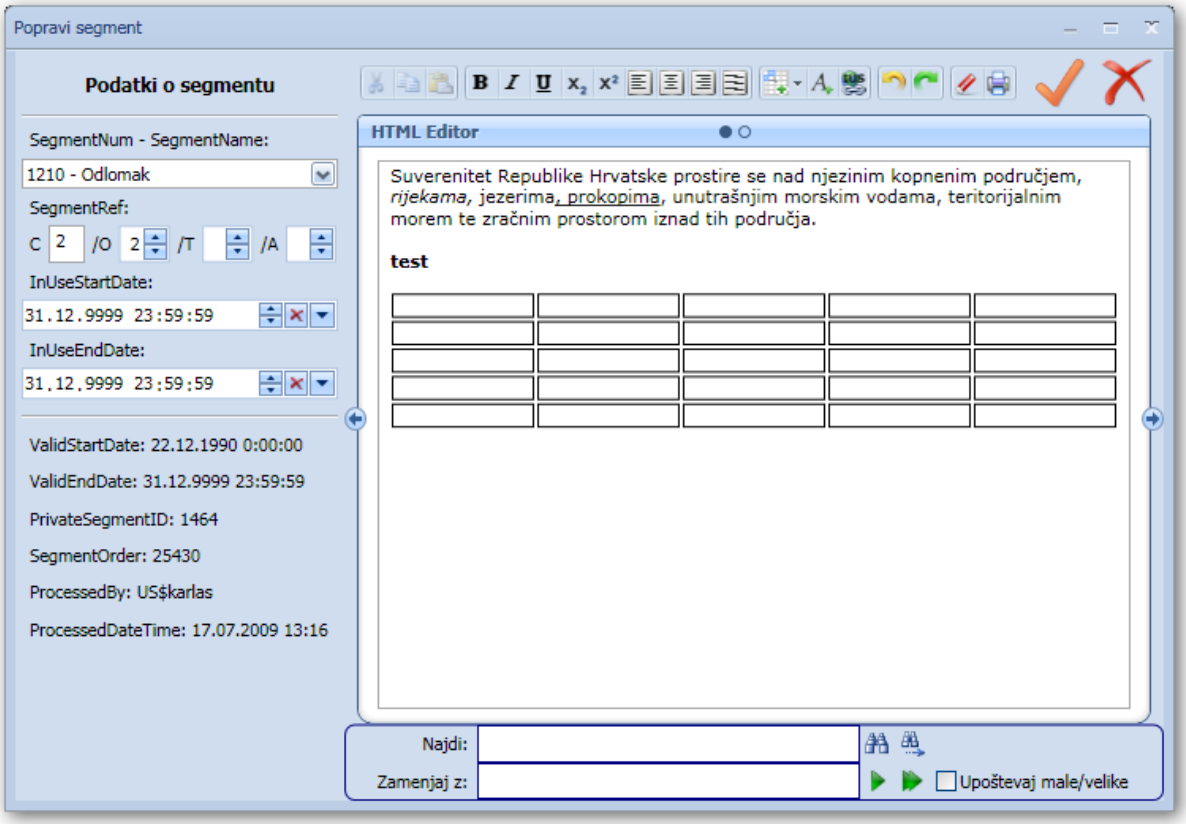
Slika 19: WindowEditHeader

5.1.5.7 WindowEditSegment

Zelo pomembno okno, namenjeno urejanju enega segmenta. Na levi strani ponuja izbiro tipa segmenta (*SegmentNum*), zapisa referenčnega zapisa segmenta (*SegmentRef*), ki je v obliki COTA (številčna oznaka formata Člen/Odstavek/Točka/Alineja), spreminjanje datuma uporabe segmenta (*InUseStartDate*, *InUseEndDate*) ter pregled ostalih podatkov segmenta, ki se spreminjajo avtomatično glede na akcije uporabnika. Večinski del aplikacije je namenjen urejanju vsebine segmenta. Na vrhu si sledijo gumbi za delo z odlagališčem (kopiranje, lepljenje, rezanje), oblikovanjem (krepki, poševni, podčrtani tekst, pozicija teksta), dodajanjem tabel, simbolov ter povezav. Poleg teh so še gumbi za preklic sprememb (angl. *Undo*, *Redo*), brisanje oblikovanja in tiskanje. Na koncu ostaneta še gumba za potrditev oz. preklic sprememb. Oba zapreta okno. Pod gumbi se nahaja kontrolnik za različne tipe segmentov:

- navadni segment z omogočenimi vsemi možnostmi oblikovanja teksta,
- segment, ki ne dovoljuje oblikovanja ter je prikazan s pisavo z nespremenljivo širino znakov,
- segment, ki predstavlja datotečno prilogo in ima njim namenjena polja (uvoz priloge v namenski direktorij, naslov, opis, velikost) ter
- segment, ki predstavlja dokumentno prilogo in vsebuje polja s podatki o dokumentu (*DocName*, *SOPI*, verzija, naslov in opis).

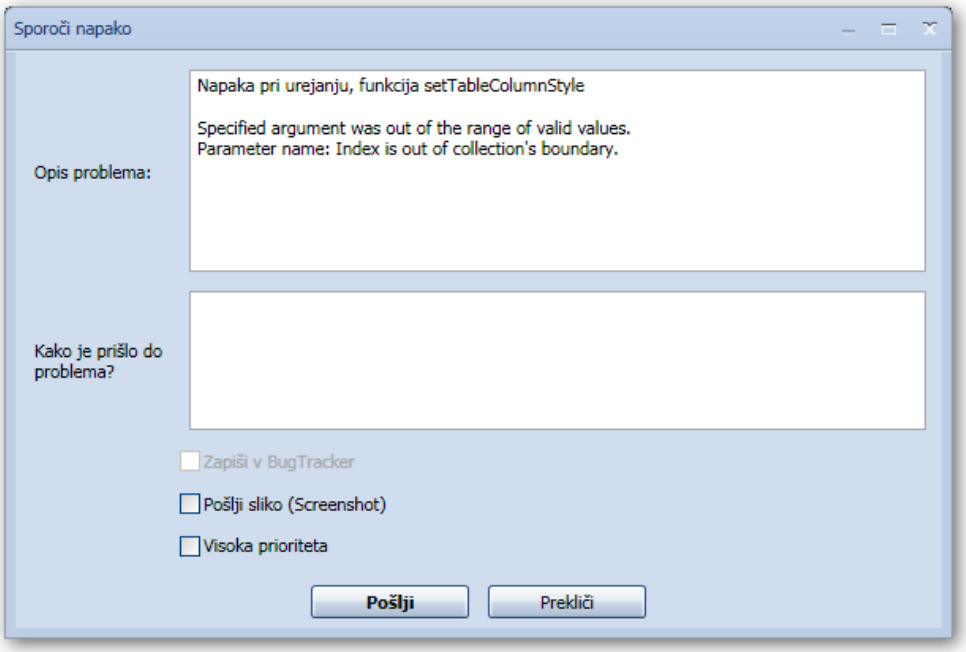
Kontrolnik se avtomatsko spreminja glede na izbrani tip segmenta. V spodnjem delu se nahaja še panel, ki vsebuje polja za iskanje in zamenjavo besed znotraj segmenta. Ne deluje pri segmentih, ki predstavljajo datotečno ali dokumentno prilogo.



Slika 20: WindowEditSegment

5.1.5.8 WindowErrorReport

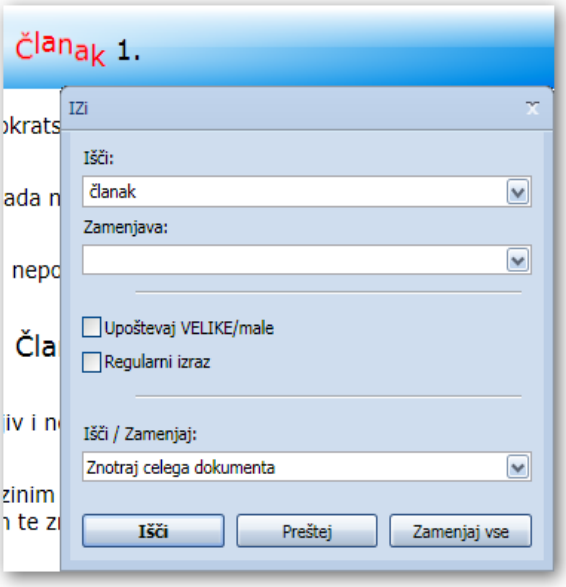
Okno za sporočanje napak skrbniku aplikacije. Vsebuje dve vnosni polji, enega za vnos opisa problema ter drugega za vnos opisa, kako je prišlo do tega problema. Pred pošiljanjem napake je moč vključiti pošiljanje slike aplikacije ter visoke prioritete. Slika aplikacije se ustvari še pred odprtjem tega okna, zato da ne zastira pogleda na problem. Visoka prioriteta pa pomeni, da mora skrbnik aplikacije čim prej preveriti vzrok težave. Aplikacija po kliku na gumb *Pošlji* sestavi sporočilo, ki se pošlje preko elektronske pošte. V tem sporočilu se poleg vnosnih in izbirnih polj nahajajo še podatki z imenom uporabnika ter njegovim računalnikom, verzijo aplikacije ter podroben opis napake (če obstaja).



Slika 21: WindowErrorReport

5.1.5.9 WindowFindReplace

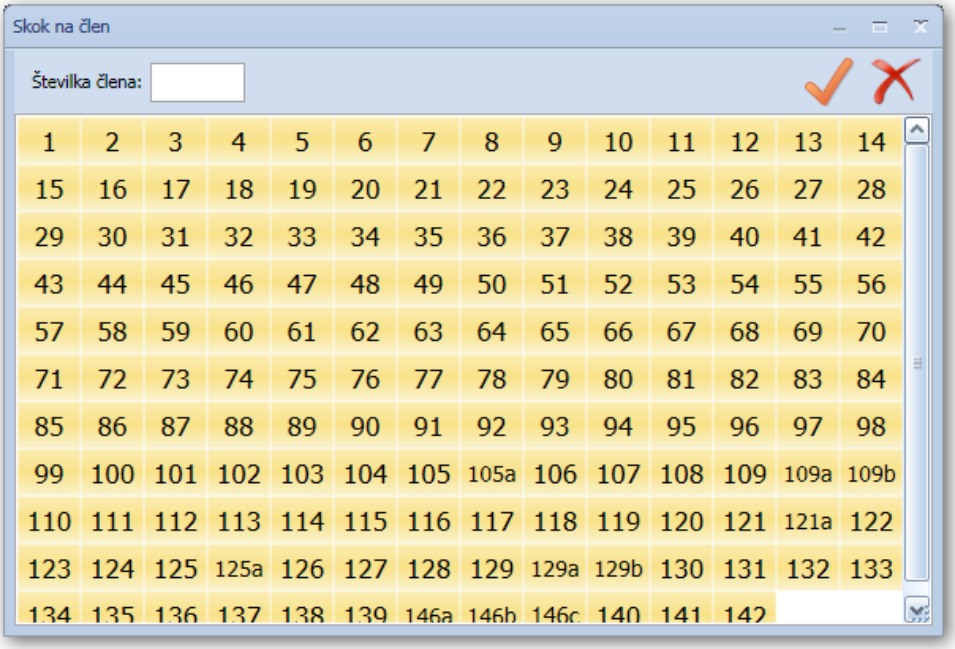
Iskanje ter zamenjava teksta olajša delo z dokumenti. V tem oknu se nahajata vnosni polji za iskalni ter zamenjalni niz, izbirna polja za upoštevanje velikosti črk ter uporabo regularnih izrazov (angl. Regular Expressions), na koncu pa še izbira območja iskanja (znotraj označenih segmentov, znotraj vseh segmentov označenega tipa segmenta ter znotraj celotnega dokumenta). Prvi dve območji sta na voljo le, če je v dokumentu označen vsaj en segment. Z gumbi uporabnik nadzoruje, ali gre za iskanje, štetje ali zamenjavo teksta. Najdeni tekst v dokumentu plapola z rdečo barvo.



Slika 22: WindowFindReplace s plapolanjem teksta v ozadju

5.1.5.10 WindowJumpToClen

Z bližnjico CTRL+G se prikaže to okno, v katerem se iz dokumenta izlušči seznam vseh členov. Ta seznam se prikaže v obliki kvadratov s številkami v njih. Klik na enega izmed njih povzroči zaprtje okna ter skok na izbrani člen. Kot bližnjica je v oknu še eno vnosno polje, kamor se s tipkovnico vpiše številko člena ter s pritiskom tipke ENTER izvrši tak skok kot pri kvadratih. Enak rezultat povzroči tudi gumb s kljukico, drugi gumb s križem okno samo zapre, torej brez skoka (bližnjica ESC).

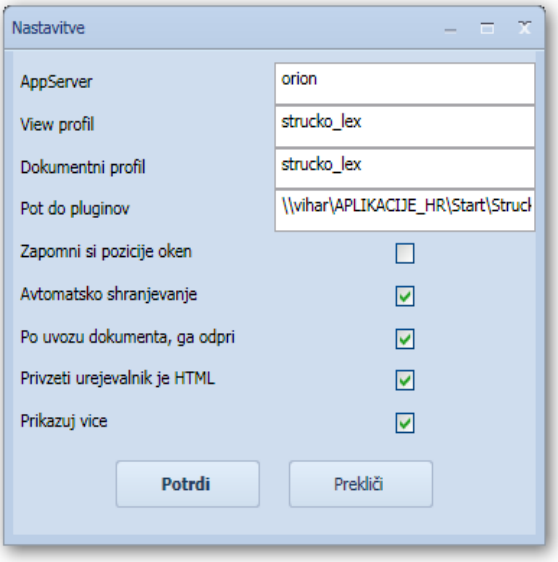


Slika 23: WindowJumpToClen

5.1.5.11 WindowNastavitve

Okno za spreminjanje nastavitev aplikacije. Vnosna polja predstavljajo najbolj osnovne nastavitve, ime aplikacijskega strežnika (*AppServer*), profil za drevesa (*View*) in dokumente ter fizično pot do vtičnikov. Privzeta pot je vedno enaka (kot si jo je zamislil skrbnik aplikacije), vendar se jo za testne namene lahko tudi spreminja. Na koncu so še izbirna polja za:

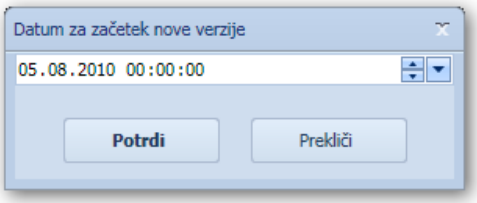
- pomnjenje pozicije stranskih oken (ko se aplikacija zapre in ponovno odpre, so vsa okna na enaki poziciji kot pred zaprtjem),
- avtomatsko shranjevanje dokumentov (po vsaki akciji, ki spremeni segment),
- odpiranje dokumenta po uvozu z vtičniki,
- privzeti urejevalnik v oknu za urejanje segmentov (če polje ni izpolnjeno, je privzeti urejevalnik brez možnosti oblikovanja teksta) ter
- prikaz vicev med čakanjem odgovora od aplikacijskega strežnika (omejen nabor vicev je povzročil nerganje uporabnikov, zato se lahko namesto vicev prikazuje samo »Nalaganje ...«).



Slika 24: WindowNastavitve

5.1.5.12 WindowNovaVerzija

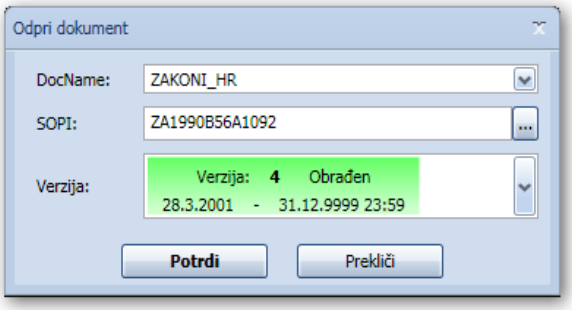
Okno vsebuje samo eno polje za vnos datuma, s katerim se prične veljavnost novo ustvarjene verzije.



Slika 25: WindowNovaVerzija

5.1.5.13 WindowOpenDocument

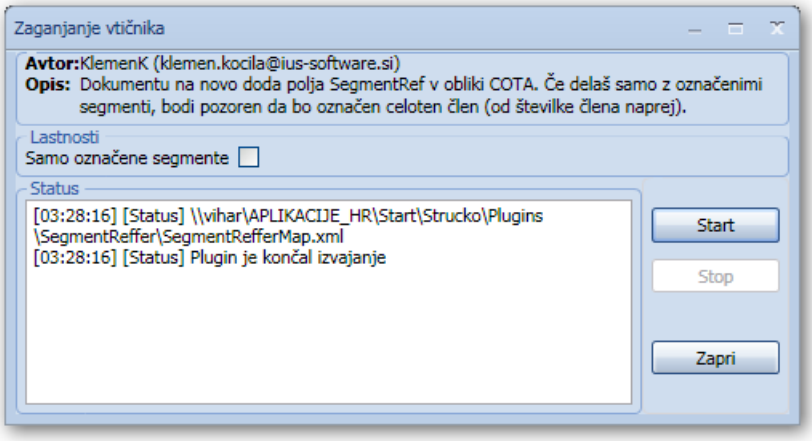
To okno je namenjeno iskanju ter odpiranju dokumentov. V prvi vrstici uporabnik izbira tip dokumenta iz seznama vseh tipov, ki jih je moč dobiti z izbranim dokumentnim profilom. Ko je izbran tip dokumenta, je potrebno izbrati še *SOPI* – identifikator dokumenta. V pomoč pri iskanju obstaja drugo pomožno okno, *WindowSearchDocument*, opisano v poglavju 5.1.5.16. Nazadnje je, opcijsko, moč izbrati še verzijo dokumenta iz seznama vseh verzij tega dokumenta. Nenazadnje sta tu še gumba za potrditev ter preklic.



Slika 26: WindowOpenDocument

5.1.5.14 WindowPluginRun

To okno podpira zaganjanje obeh vrst vtičnikov, samopopravke ter uvoz dokumentov. Na vrhu je prikaz osnovnih podatkov o izbranem vtičniku (avtor in opis vtičnika). Za tem se nahaja seznam lastnosti vtičnika, ki jih uporabniki spreminjajo/vnašajo. Ta seznam se avtomatsko kreira glede na to, katere lastnosti so v posameznih vtičnikih vnaprej določene. Podpira datumska polja, izbirna polja, osnovna vnosna polja za tekst ter specializirana vnosna polja (za vnos celih ali realnih številk in poti do datotek oz. direktorijev). Poleg vsakega polja se prikazuje še njegov opis. S klikom na gumb *Start* se prične izvajanje vtičnika. Status izvajanja se sproti izpisuje v sredinski del okna. Izvajanje je moč tudi prekiniti, in sicer z gumbom *Stop*. Po končanem izvajanju vtičnika za uvoz dokumenta se prikaže okno *WindowDocInsertInfo*, opisano v poglavju 5.1.5.4.

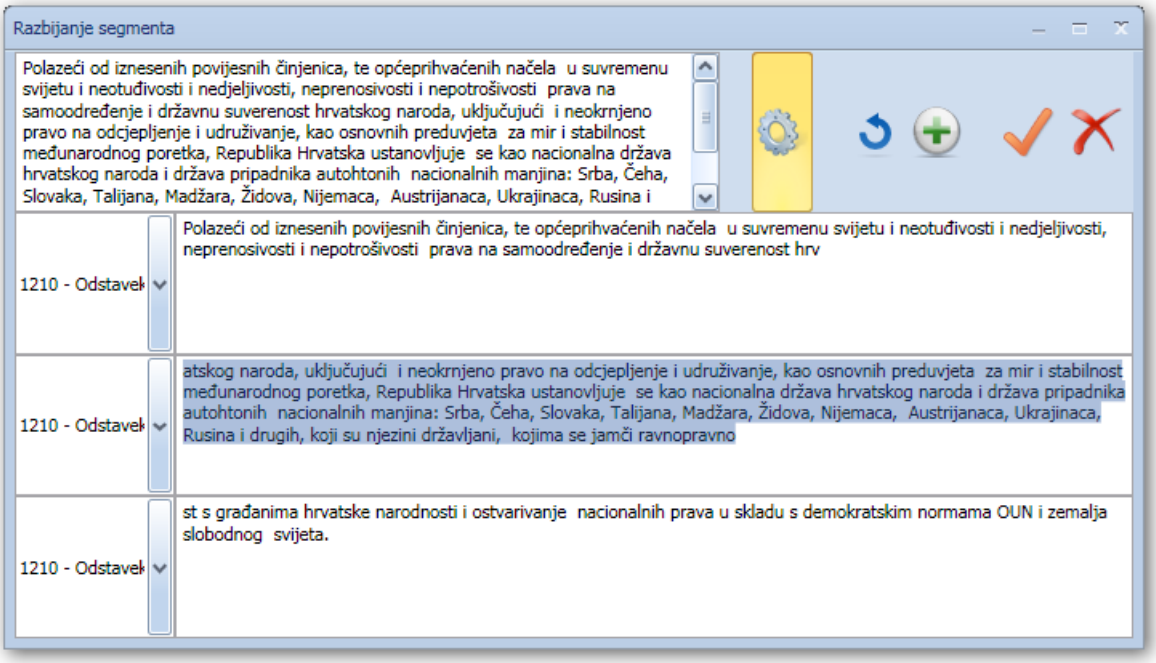


Slika 27: WindowPluginRun

5.1.5.15 WindowRazbijanje

Vrlina tega okna je razbijanje teksta na več kosov. Sistem razbijanja je razvit tako, da uporabniki s klikom na tekst takoj povzročijo razbitje na dva dela – levi in desni. Levi del ostane segmentu, ki se razbija, desni pa se prenese v novi segment. Tako levega kot desnega

je moč večkrat razbiti s takim načinom, dokler ni dosežen željen rezultat. Možen je tudi ročni način, ki se ga vklopi s klikom na ikono nazobčanega kroga. V primeru, da se uporabnik kje zmoti, ima na voljo dva gumba za pomoč, razveljavitev zadnjega razbijanja ali pa kar začetek znova. S klikom na kljukico se potrdi trenutno razbito stanje, klik na križ pa dokumentu ne naredi nobene spremembe.



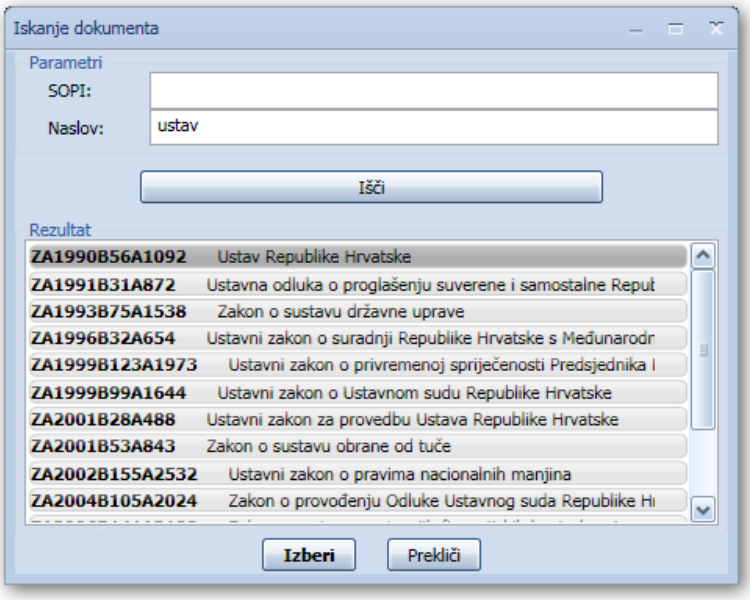
Slika 28: WindowRazbijanje

5.1.5.16 WindowSearchDocument

Iskanje dokumenta je možno preko dveh vnosnih polj:

- vnos celega ali samo dela identifikatorja dokumenta (*SOPI*) ali
- vnos celega ali samo dela naslova iskanega dokumenta.

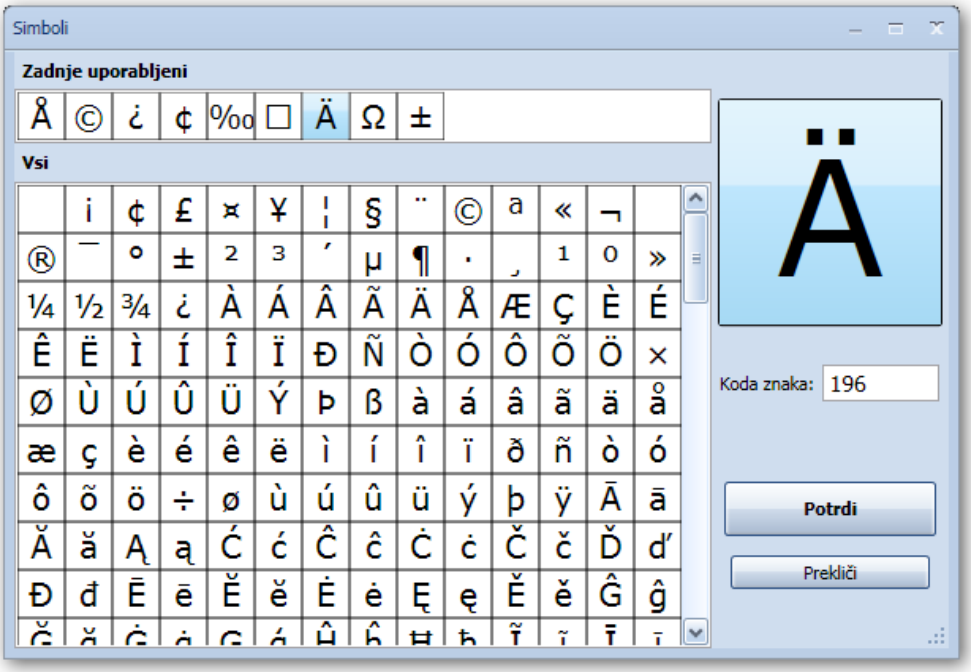
Rezultat je seznam vseh najdenih dokumentov. Dvoklik na enega izmed njih zapre okno in sporoči vrhnjemu oknu izbran *SOPI*.



Slika 29: WindowSearchDocument

5.1.5.17 WindowSimboli

Okno za podporo pri urejanju segmentov. Vsebuje dva seznama simbolov. Prvi je seznam zadnjih 15 uporabljenih simbolov, drugi pa seznam vseh simbolov. Izbran simbol se pokaže v večji obliki na desni strani okna, pod katero pa se pokaže še številska koda tega simbola. Dvoklik na enega izmed simbolov ali na gumb *Potrdi* zapre okno in preda izbrani simbol oknu, ki ga uporablja (*WindowEditSegment*, poglavje 5.1.5.7).



Slika 30: WindowSimboli

5.1.6 Večjezičnost

Že na začetku je bilo predvideno, da bo imela aplikacija večjezične uporabnike, zato je bilo potrebno ustvariti univerzalni izdelek, v katerem bo razvita podpora večjezičnosti. Vsi teksti, ki se prikazujejo na uporabniškem vmesniku, se nahajajo v ločenih datotekah. Vsaka datoteka ima v imenu podatek o kulturi (npr. ime datoteke s hrvaško kulturo: *Language.hr-HR.resx*). V primeru, da iskanega teksta ni v datoteki s trenutno izbrano kulturo, se vzame iz privzete datoteke z imenom *Language.resx*. Kultura se glede na izbrano vrednost v okolju Windows izbere avtomatsko ob zagonu aplikacije.

5.2 Izvajanje v ozadju

Za večjo odzivnost aplikacije je od uporabniškega vmesnika potrebno ločiti procese, ki lahko vzamejo več časa. V ta namen ima *Stručko* »delavce«, ki tečejo v ozadju (angl. Backgroundworkers). To so večinoma klici na aplikacijski strežnik, za katere ne vemo oz. ne moremo predvideti, koliko časa bodo trajali. Aplikacijski strežnik ima več vlog, lahko ga uporablja več uporabnikov, pričakovati moramo celo neodzivnost. Vsak izmed teh delavcev je ločen poslovni objekt, ki ga je mogoče uporabljati povsem neodvisno.

5.2.1 BackgroundWorkerEksternaOpomba, BackgroundWorkerInternaOpomba

Delo z eksterno in interno opombo trenutno odprtega dokumenta. Ta dva delavca se uporabljata pri stranskih oknih *Interne in eksterne opombe* (poglavje 5.1.3) in omogočata

branje ter posodabljanje oz. tvorjenje opombe (interne ali eksterne). Če segment še ne obstaja, se opomba tvori, če segment že obstaja, pa posodobi. Na ta način se ustvarja opombo na nivoju dokumenta (vidna v vseh verzijah).

5.2.2 BackgroundWorkerExtendValidEndDate

Spreminjanje časovne veljavnosti tekstovnih segmentov je pomembna funkcionalnost, ki se uporabi ob tvorjenju nove verzije čistopisa. Pred izvedbo je potrebno preveriti, če ima naslednja verzija slučajno že kakšen veljaven segment – v takem primeru funkcija ne sme nadaljevati, saj bi se lahko segmenti časovno prekrivali. S tem, ko podaljšamo segmentom časovno veljavnost v novo verzijo, lahko uporabniki začnejo z urejanjem nove verzije.

5.2.3 BackgroundWorkerGetDataSet

Orodje za klic na aplikacijski strežnik s poljubno poizvedbo SQL. Delavec nato vrne podatke v obliki *DataSet*.

5.2.4 BackgroundWorkerGetDocNames

Delavec, ki vrne seznam vseh tipov dokumentov iz baze. Ta seznam se potrebuje pri urejanju povezav (poglavje 5.1.5.1) in odpiranju dokumentov (poglavje 5.1.5.13).

5.2.5 BackgroundWorkerGetPrilogePot

Iz konfiguracijske datoteke aplikacijskega strežnika vrne fizično pot do prilog. To pot se v oknu za urejanje segmentov (poglavje 5.1.5.7) uporablja pri urejanju prilog.

5.2.6 BackgroundWorkerInsertDocument

Omogoča kreiranje novega dokumenta v bazi z vsemi podatki, vključno s segmenti. Ta funkcionalnost se največkrat uporablja pri vtičnikih (poglavje 5.3.2) in oblikovanju novega dokumenta (poglavje 5.1.5.5).

5.2.7 BackgroundWorkerLoadDocument

Najbolj dejaven delavec – vsebuje klice na aplikacijski strežnik za delo z dokumentom. Vsebuje klice za *Get*, *CheckOut*, *Update*, *CheckIn*, *UndoCheckOut* ter *DestroyLastTechVersion*. Izstopa klic za *Get*, po katerem se pokliče poizvedba največje vrednosti polja *PrivateSegmentID*, vendar samo v primeru, da je dokument v stanju urejanja. To vrednost se uporabi pri tvorjenju novih segmentov, tako da ima vsak nov segment novo, večjo vrednost. Delavec se uporablja v osrednjem oknu *IUSDocumentWindow* (poglavje 5.1.4.2).

5.2.8 BackgroundWorkerLoadLinks

Po dejavnosti je ta delavec na drugem mestu, saj vsebuje klice na aplikacijski strežnik za delo s povezavami – branje seznama povezav ter urejanje povezav. Uporablja se v stranskem oknu za seznam povezav (poglavje 5.1.3) ter pomožnem oknu za urejanje povezav *WindowCreateEditLink* (poglavje 5.1.5.1).

5.2.9 BackgroundWorkerLoadTree

V stranskem oknu *Drevesa* vrača sezname za grajenje drevesa (poglavje 5.1.3). Prvič ko uporabnik odpre to stransko okno, se naložijo samo vrhnji elementi tega drevesa. Klik na enega od teh elementov vrne seznam vseh njegovih podelementov, prav tako tudi klik na podelemente povzroči dobivanje seznama podpodelementov (itd. rekurzivno do lista drevesa). Vsak list pa potem omogoča dvoklik, ki odpre ta dokument, ki ga predstavlja list.

5.2.10 BackgroundWorkerLoadVersions

Vrača seznam verzij enega dokumenta. Vsaka verzija v tem seznamu vsebuje podatke o njenem časovnem razporedu, naslovu, statusu ter v primeru čistopisov še naslov objave, ki je spreminjala to verzijo.

5.2.11 BackgroundWorkerPluginStart

Delavec za zaganjanje vtičnikov. Kot argument sprejme vmesnik vtičnika, ki se zaganja, pokliče funkcijo Start in po izvršeni funkciji rezultat vtičnika vrne v dokument (samopopravki, poglavje 5.3.1) ali pa zapiše nov dokument v bazo (poglavje 5.3.2).

5.2.12 BackgroundWorkerReadServerConfig

Podpora branju konfiguracijske datoteke iz aplikacijskega strežnika. Funkcija preko argumentov dobi ključ, katerega vrednost potem vrne.

5.2.13 BackgroundWorkerStatus

Branje ter posodabljanje statusa verzije dokumenta je delo, ki ga opravlja ta delavec. Poleg tega zna z določenimi argumenti vrniti tudi seznam vseh možnih statusov, tako da uporabniki lahko izbirajo med njimi. Ta funkcionalnost se uporablja v stranskem oknu *Lastnosti verzije (DocHeader)*, poglavje 5.1.3.

5.2.14 BackgroundWorkerUvozTekstaPrveObjave

Enostaven klic na aplikacijski strežnik za uvoz teksta prve objave v trenutno odprto, prvo verzijo čistopisa. Prva objava je navadno kar prva verzija čistopisa, vendar sta to vseeno dva ločena dokumenta. Funkcija skopira vse tekstovne segmente iz objave v prvo verzijo čistopisa.

5.3 Vtičniki

Vtičniki so bili razviti za omogočanje razširljivosti aplikacije. S tem smo razširili razvoj na več lokalnih razvijalcev (za hrvaške vtičnike skrbi hrvaški razvijalec). Podpiramo dve vrsti vtičnikov, samopopravke in vtičnike za uvoz dokumentov. Obe vrsti imata razvit vmesnik, preko katerega aplikacija ve, da gre za vtičnik ter zna iz njega izluščiti vse nujne lastnosti (*PluginAuthor*, *PluginAuthorEmail*, *PluginName*, *PluginDescription*, *DocName*, *Properties*), metode (*PreparePlugin*, *Start*) ter dogodke (*OnSendMessage*, *OnDBSelect*). Lastnosti so namenjene opisu vtičnika. Metode aplikaciji zagotavljajo implementiranost, prva se izvrši, ko

uporabnik klikne na vtičnik, druga pa, ko ga zažene. Dogodki vtičnikov so namenjeni prenašanju sporočil aplikaciji. Vtičniki se naložijo tako, da se naredi selektivni seznam knjižnic, datotek s končnicama ».dll« in ».exe« iz direktorija, ki je namenjen vtičnikom. Vsako knjižnico se potem preveri, ali vsebuje katerega izmed dveh vmesnikov za vtičnike. Če ga vsebuje, se shrani njegove podatke za kasnejše nalaganje.

5.3.1 Samopopravki

Samopopravki so mišljeni kot vtičniki za popravljanje obstoječih dokumentov (npr. če bi radi tistim segmentom, ki se začnejo s pomišljajem, popravili tip segmenta v tip *Alineja*). Na ta način precej olajšamo delo uporabnikom. Ta vrsta vtičnikov ima poleg osnovnih lastnosti še štiri druge:

- *DocGetInfo* (celoten dokument v obliki *DataSet*),
- *EditorLinkInfoFrom* (seznam povezav iz trenutnega dokumenta),
- *EditorLinkInfoTo* (seznam povezav na trenutni dokument) ter
- *SelectedSegments* (seznam označenih segmentov v trenutnem dokumentu).

V primeru, da ima dokument označene segmente, ima vtičnik možnost uporabiti samo te.

5.3.2 Uvoz dokumentov

Vtičnik je namenjen uvažanju novih dokumentov v bazo in poleg osnovnih lastnosti vsebuje še:

- *DocInsertInfo* (dokument v obliki za vnos v bazo) in
- *EditorLinkInfo* (povezave dokumenta).

Aplikacija vzame dokument in povezave ter ju preko delavca v ozadju zapiše v bazo. Razvijalec ima tako možnost razviti različne module za uvoz dokumentov v bazo, npr. uvoz novega dokumenta iz datoteke formata Microsoft Word.

5.4 Namestitev

Pomembna je tudi namestitev aplikacije. Uporabili smo tehnologijo *ClickOnce*, s katero je namestitev za uporabnike zelo poenostavljena. Potreben je samo en klik na spletni strani za namestitev in še eden za potrditev namestitve. V nekaj sekundah je aplikacija nameščena na lokalni računalnik, v skrito mapo, namenjeno takim tipom namestitev. Poskrbljeno je tudi za bližnjice za zagon, ki se nahajajo na običajnem mestu, v t. i. Start menu-ju.

Namestitve so podpisane s certifikatom, ki ga uporabljamo znotraj družbe, kar je tudi eden od pogojev za uporabo tehnologije *ClickOnce*. Dodaten pogoj, ki je del same namestitve, pa je ogrodje .NET verzija 3.5 SP1. Če tega pogoja uporabnik nima izpolnjenega, se namestitev aplikacije prekine in mu ponudi povezavo do namestitve pogojnega ogrodja.

V družbi imamo trenutno možne štiri različne namestitve aplikacije:

- testna in delovna namestitev za domači del podatkovnih baz,
- testna in delovna namestitev za hrvaški del podatkovnih baz.

Spletne strani teh namestitev se nahajajo na strežniku, ki je dostopen vsem zaposlenim v družbi.

6 Sklepne ugotovitve

Razvoj predstavljene aplikacije poteka dve leti in vsebuje približno 14.000 izvršljivih vrstic programske kode oz. približno 1400 metod. Doživela je veliko popravkov in sprememb, vse za to, da bi bila uporabniška izkušnja in s tem produktivnost čim boljša. Aplikacija se že uporablja za hrvaški, pa tudi za domači portal. Odzivi uporabnikov so zelo pozitivni, vmes je bila tudi kakšna nepravilno delujoča verzija aplikacije, vendar smo napake razmeroma hitro popravili. Največ težav je povzročala velika količina teksta in posledično velika poraba računalniških virov, kar smo rešili z virtualizacijo vmesnika.
Idej za naprej je še ogromno, če jih naštejem samo nekaj:

- fitriranje po tipih segmentov,
- nadzor vsake namestitve aplikacije preko portala (sproten ogled dnevnika uporabe, spreminjanje nastavitev itd.),
- premikanje vrstnega reda segmentov z načinom povleci-spusti (angl. Drag-Drop),
- spreminjanje vrhnjega seznama gumbov glede na izbrano vrsto dokumenta.

Med samim razvojem aplikacije smo opazili, da je uporabnike najbolj razveselila možnost izbire barvne sheme aplikacije. Seveda so veseli vsake nove funkcionalnosti, ki jim olajša delo za nekaj klikov, vendar ne tako kot to, da si barve lahko izberejo sami.
Kakorkoli, dosegli oz. presegli smo zastavljene cilje in jih poleg tega še dnevno izboljšujemo ter nadgrajujemo.

7 Viri

[1] Adam Nathan, *Windows Presentation Foundation Unleashed*, Indiana: Sams publishing, 2007.

[2] Trey Nash, *Accelerated C# 2008*, ZDA: Apress, 2007.

[3] Jeffrey E. F. Friedl, *Mastering regular expressions - Second edition*, ZDA: O'reilly, 2002.