UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Primož Kariž

Aplikacija Poštna knjiga

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

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: 00120/2011
Datum: 05.04.2011

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

Kandidat: **PRIMOŽ KARIŽ**

Naslov: **APLIKACIJA POŠTNA KNJIGA**
APPLICATION FOR TRACKING OF RECIEVED POST

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Razvijte aplikacijo za beleženje in arhiviranje prejete pošte. Uporabite platformo .NET in podatkovno bazo Microsoft SQL. Poseben poudarek dajte implementaciji enostavnega in hitrega iskanja po arhivu.

Mentor:

doc. dr. Rok Rupnik

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Rezultati diplomskega dela so intelektualna lastnina Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil Microsoft Word.

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Primož Križ,

z vpisno številko 63070299,

sem avtor diplomskega dela z naslovom:

Aplikacija Poštna knjiga

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 _____ Podpis avtorja: _____

ZAHVALA

Zahvaljujem se svojemu mentorju, doc. dr. Roku Rupniku, za strokovno pomoč in nasvete pri izdelavi diplomske naloge.

Posebno zahvalo namenjam staršema, punci in babici za podporo in spodbudo skozi celoten študij. Zahvalil bi se tudi bratu za pomoč pri študiju.

Nazadnje bi se zahvalil tudi sodelavcem za pomoč in sodelovanje pri izdelavi aplikacije.

To diplomsko delo je posvečeno starim staršem.

Kazalo

Seznam uporabljenih kratic

HTML (angl. HyperText Markup Language) označevalni jezik za razvoj spletnih strani

SQL (angl. Structured Query Language) strukturiran povpraševalni jezik za pisanje poizvedb

FCL (angl. Framework Class Library) celotna .NET Framework knjižnica

BCL (angl. Base Class Library) del FCL

XML (angl. Extensible Markup Language) jezik za grajenje strukturiranih dokumentov

CSS (angl. Cascading Style Sheets) stilski jezik namenjen urejanju izgleda html dokumenta

CLR (angl. Common Language Runtime) okolje za izvajanje programov

SVN (angl. Subversion) sistem za hranjenje vseh verzij različnih datotek

SGML (angl. Standard Generalized Markup Language) tehnologija za opredelitev označevalnih jezikov za dokumente

W3C (angl. World Wide Web Consortium) mednarodna organizacija za spletne standarde

UML (angl. Unified Modeling Language) jezik za modeliranje računalniških sistemov

Povzetek

Namen diplomskega dela je izdelava Windows aplikacije za arhiviranje prispele in poslane pošte v podjetju. Aplikacijo odlikuje enostavna uporaba in možnost hitrega iskanja po arhivu.

V uvodnem delu so opisane uporabljene tehnologije in orodja za razvoj aplikacije. Aplikacija je bila razvita v programskem jeziku C# na podlagi .NET platforme. Uporablja podatkovno bazo Microsoft SQL, ki hrani vse podatke.

V glavnem delu je najprej predstavljena aplikacija, ki jo je naročnik uporabljal v preteklosti. Sledi načrtovanje podatkovne baze in funkcionalnosti nove aplikacije. Grafično orodje PowerDesigner je služilo za izdelavo konceptualnega in fizičnega modela ter diagrama primerov uporabe aplikacije. Jedro glavnega dela je natančen opis funkcionalnosti aplikacije med katere spada tudi napredno iskanje, ki uporabniku omogoča iskanje prejetih oziroma poslanih pošt in prilog ter strank.

Razvito aplikacijo naročnik testno uporablja, zato se utegnejo določene funkcionalnosti v prihodnosti spremeniti oziroma dodati.

Ključne besede:

Načrtovanje, razvoj, Windows aplikacija, uporabljena orodja

Abstract

Purpose of this thesis is to develop a Windows application to archive incoming and outgoing mail within the company. Application is easy to use and allows quick search through the archives.

The introductory section describes technologies and tools which were used for application development. Application was developed in programming language C# on .NET platform. It uses Microsoft SQL database to store all the data.

The main part starts with the description of application which was used by the company in the past, followed by the design of database and application functionality. Graphical tool PowerDesigner was used to produce conceptual and physical model and use case diagram of application. The core of the main part is the detailed description of application functionality which include an advanced search that allows user to search for incoming or outgoing mails, mails attachments and customers.

The developed application is being tested by the company, hence certain functionalities may change or add in the future.

Key words:

Design, development, Windows application, used tools

1 Uvod

Opredelitev problema

Tehnologija se v času sodobnega napredka nenehno razvija. Vse več nalog se opravlja preko računalniških aplikacij, ki se vedno bolj izboljšujejo. Določene aplikacije razvijalci s časom prenehajo nadgrajevati in zato se uporabniki te aplikacije odločijo za zamenjavo z novejšo, zmogljivejšo in predvsem preprostejšo. Naš naročnik je aplikacijo uporabljal že več let. Uporaba te aplikacije je bila kompleksna in ni omogočala funkcionalnosti, ki bi jo naročnik želel uporabljati v prihodnosti. Tako smo se skupaj dogovorili za izdelavo nove, ki bo vsebovala vse funkcionalnosti zahtevane s strani naročnika. Poleg tega bo aplikacija enostavna za uporabo ter licenca bo cenejša od predhodne aplikacije.

Namen diplomske naloge

Orodja za izdelavo aplikacij se vse bolj razvijajo in nam tako omogočajo hitrejše in enostavnejše razvijanje funkcionalnosti aplikacij. Namen diplomske naloge je uporabniku poenostaviti delo in omogočiti dodatno funkcionalnost. Pomembna prednost nove aplikacije je, da omogoča tudi iskanje vnešenih podatkov. Za dosego vseh zastavljenih ciljev smo bili mnenja, da je najprimernejši jezik C#. Zaradi te odločitve aplikacija deluje samo na operacijskem sistemu Windows, kar pa v našem primeru ni bistvenega pomena, saj podjetja večinoma uporabljajo navedeni operacijski sistem.

Struktura diplomske naloge

V diplomski nalogi so najprej povzeta orodja in tehnologije, s pomočjo katerih je bila aplikacija izdelana. Sledi kratek opis načrtovanja in nato tudi podroben opis same aplikacije. Na koncu so povzete ideje za izboljšavo aplikacije, ki jih imamo v prihodnje tudi namen razviti.

2 Uporabljena tehnologija in orodja

Izbira tehnologij in orodij za razvoj aplikacije je odvisna od zahtev naročnika. Odločitev je pomembna, saj nekatera orodja in tehnologije omogočajo hitrejšo izdelavo določenih funkcionalnosti. Pri izbiri je potrebna pazljivost, kajti izbrana orodja in tehnologije morajo biti primerna tudi za izdelavo funkcionalnosti, ki jo ima razvijalec namen razviti v prihodnosti.

2.1 Microsoft Visual Studio 2010

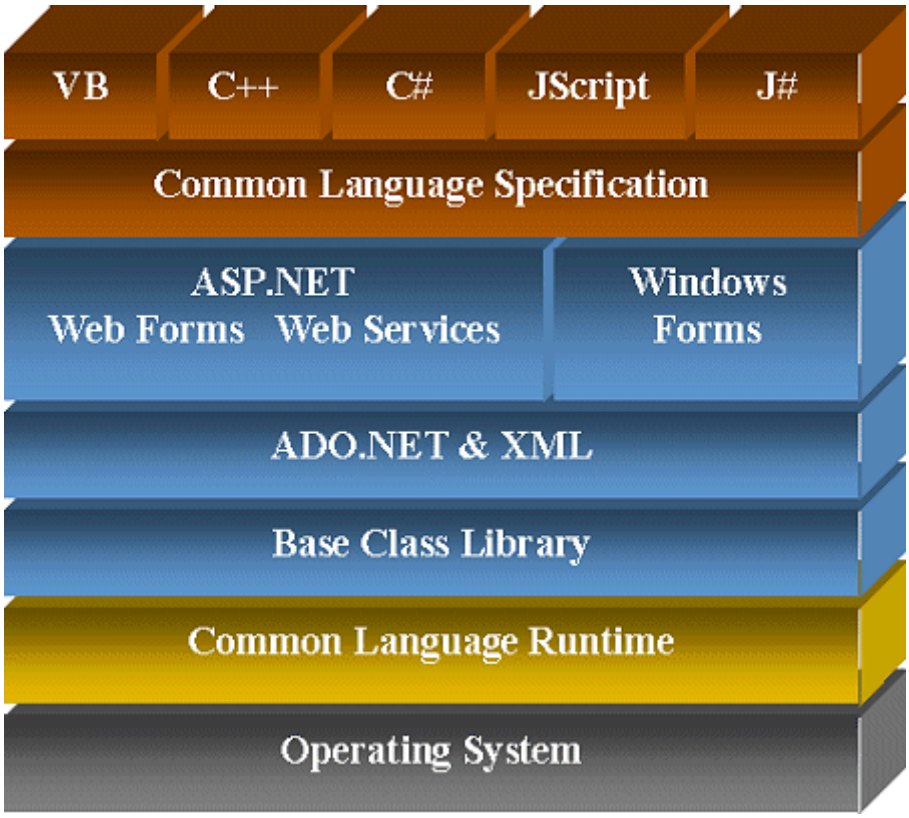
Microsoft Visual studio [1] je integrirano razvojno okolje, ki nam omogoča razvijanje uporabniških vmesnikov, spletnih strani, spletnih storitev ter visokokakovostnih »Windows Forms« aplikacij. Podpira različne programske jezike, kot so C, C++, C#, VisualBasic.NET, Python, Ruby, HTML, XML, Javascript ter CSS. Nekateri izmed njih so že vgrajeni v razvojno okolje, podporo za ostale pa je mogoče namestiti ločeno. Visual studio vsebuje okno za urejanje programske kode, ki vsebuje orodje IntelliSense. To nam omogoča pregled možnih ukazov in samodejno dokončevanje kode. Vgrajen ima tudi razhroščevalnik (ang. debuger), kateri je uporabniku v veliko pomoč pri iskanju napak, ter orodje Windows Forms Designer, ki uporabniku omogoča preprosto izdelavo grafičnih uporabniških vmesnikov ter upravljanje z dogodki, ki se nanašajo na posamezne elemente tega vmesnika. Uporabniku tako ni potrebno urejati osnovnih lastnosti grafičnih elementov v sami programski kodi, saj to lahko stori preko okna namenjenega spreminjanju lastnosti izbranega grafičnega elementa. Visual Studio 2010 ima že vgrajeno zbirko knjižnic .NET Framework 4.

2.2 Programski jezik C#

Programski jezik C sharp [2] je izdelalo podjetje Microsoft leta 2000. Skozi leta je podjetje jezik izboljševalo, zadnja različica jezika pa je prišla leta 2010. Jezik je tako kot večina novejših jezikov objektno orientiran, kar omogoča boljšo strukturiranost kode ter lažje programiranje. Poleg tega je jezik precej podoben programskemu jeziku Java, ki ga je izdelalo podjetje Sun. Omogoča avtomatsko pretvorbo določenih objektov v tipe in obratno. Za sproščanje pomnilnika skrbi zbiralec odvečnih spremenljivk (ang. Garbage collector), ki samodejno sprosti spremenljivke za katere ugotovi, da jih ne potrebujemo več. Uporaba programskega jezika C# se skozi čas povečuje, kar je posledica enostavne uporabe ter možnosti uporabe knjižnic iz .NET Frameworka. Velika prednost opisanega programskega jezika je, da je podprt v vseh verzijah orodja Microsoft Visual Studio.

2.3 .NET Framework

.NET Framework [3] je programski okvir, ki ga lahko namestimo na Microsoftovih operacijskih sistemih. Osnovna arhitektura, ki je prikazana na sliki 2.1, se kljub novim različicam ne spreminja. .NET Framework je sestavljen iz dveh glavnih komponent in sicer knjižnice in CLR. Knjižnica podpira več programskih jezikov, kar razvijalcu omogoča interoperabilnost programskih jezikov, saj lahko vsak izmed podprtih programskih jezikov uporabi kodo drugega programskega jezika. Knjižnica BCL ponuja tudi uporabniški vmesnik, povezljivost podatkovnih baz ter pridobivanje njihovih podatkov, razvoj spletnih aplikacij, kriptografijo, numerične algoritme in povezovanje z omrežjem. CLR je virtualni stroj, ki skrbi za izvajanje kode napisane v jeziku, ki ga knjižnica podpira. Microsoft je začel z razvojem .NET Framework-a leta 1990, ki pa takrat znan kot Next Generation Windows Services. Leta 2000 je prišla prva beta različica .NET 1.0. Novejši operacijski sistemi MS Windows imajo že vključeno eno izmed različic tega programskega okvirja. Najnovejša različica .NET 4.0 je izšla aprila 2010. .NET ima tudi svojega zbiralca odvečnih spremenljivk, ki se požene, ko je porabljena določena količina pomnilnika.



Slika 2.1: Osnovna arhitektura .NET Frameworka.

2.4 Syncro SVN

Program, ki smo ga v podjetju uporabljali za boljšo organizacijo, varnost samega projekta ter sledenje razvoju projekta se imenuje Syncro SVN. Subversion je sistem namenjen sledenju sprememb različnih datotek. Uporabniku omogoča sočasno delo skupine ljudi na istem projektu ter dostop do poljubne vmesne verzije projekta. Ker bi bilo pri velikih projektih za navadno shranjevanje verzij potrebno veliko prostora, SVN to dela na pameten način in sicer tako, da si shranjuje samo spremembe med verzijami. Grafični vmesnik uporabniku poenostavi pregled in posodobitev datotek.

2.5 Označevalni jezik HTML

HTML je označevalni jezik za izdelavo spletnih strani [4]. Nadbesedilo je način označevanja besedila ali grafičnih elementov (slika ali del slike), ki omogočajo povezavo oziroma skok na drugi del besedila ali večpredstavni element. HTML je standardiziran in za ta standard skrbi W3C ter sodi v družino SGML (angl. Standard Generalized Markup Language). Značilnost jezika HTML je enostavnost, saj je bil zasnovan tekstovno z minimalnim številom oznak, da se ga lahko vsakdo hitro nauči in s poljubnim urejevalnikom napiše lasten dokument. HTML dokument je tekstovna datoteka, ki vsebuje oznake s pomočjo katerih različnim brskalnikom povemo, kako naj prikažejo vsebino naše spletne strani. Sam sem ga uporabil pri generiranju strani za tisk.

HTML oznake

Vsak HTML dokument vsebuje oznake. Vse oznake so definirane med znakoma za manjše (<) in pa večje (>) npr. <td>. Nekatere ozake so enodelne, druge dvodelne. Dvodelne oznake začnemo pisati s prvim delom npr. (<td>) in zaključnim delom (</td>). Zaključni del vedno vsebuje poševno črto in se tako tudi razlikuje od začetnega dela. Oznake lahko vsebujejo atribute, ki nudijo dodatne informacije brskalnikom.

2.6 Microsoft SQL Server 2008

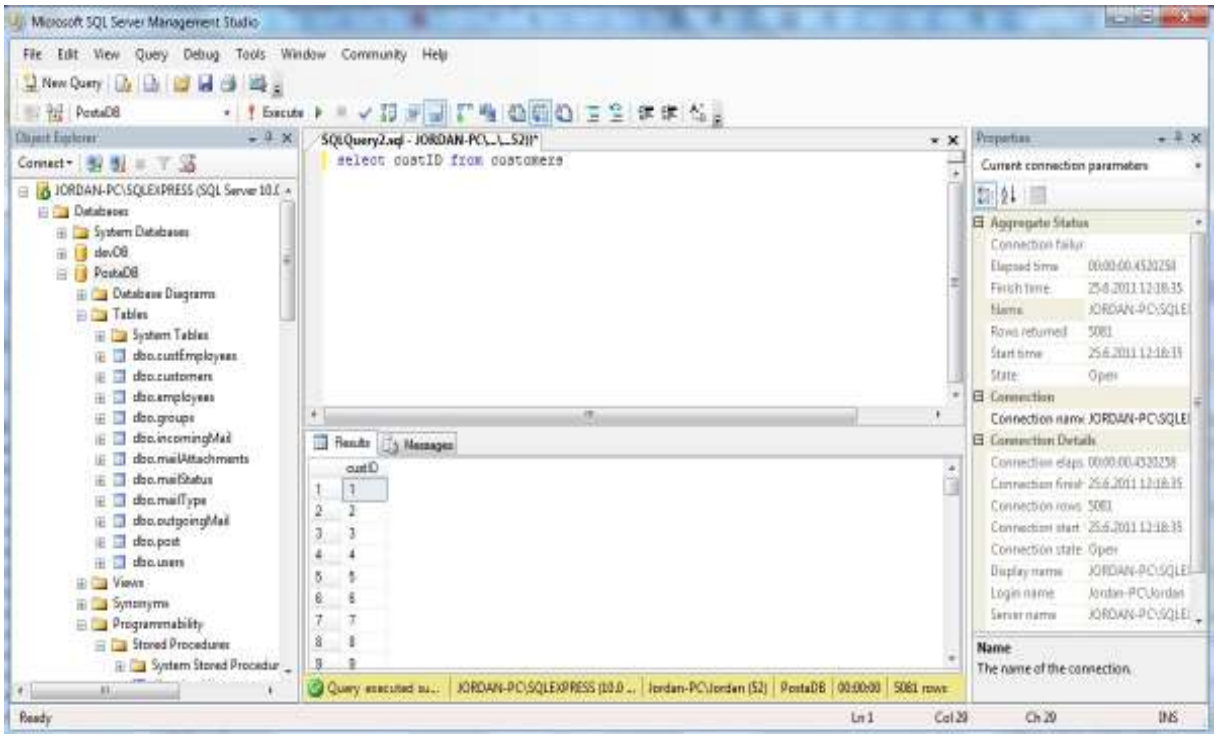
Microsoft SQL Server 2008 je brezplačna tehnologija za upravljanje z relacijskimi podatkovnimi bazami (ang. Relational Database Management System) [5]. Primarni jezik za poizvedovanje je Transact SQL, ki razvijalcu omogoča pisanje bolj zapletenih poizvedb. Strežnik je namenjen hranjenju in obdelavi podatkov. Glavna enota za shranjevanje podatkov je podatkovna baza, ki vsebuje tabele s tipiziranimi stolpci v katerih se nahajajo podatki. Poleg tabel lahko podatkovna baza vsebuje tudi druge objekte, kot so pogledi ter shranjene procedure (ang. stored procedures), ki so namenjene ločevanju programske in SQL kode. Zaradi boljše preglednosti in hitrejših popravkov smo se odločili za ločitev SQL kode od kode programskega jezika C#. Posledično smo imeli veliko shranjenih procedur.

2.7 SQL Management Studio

SQL Management Studio je brezplačno grafično orodje (slika 2.2), ki uporabniku omogoča upravljanje podatkovnih baz in njenih komponent na enostaven način. Grafični vmesnik vsebuje štiri glavne komponente:

- okno, ki prikazuje arhitekturo podatkovnih baz,
- okno namenjeno pisanju poizvedb nad izbrano podatkovno bazo,
- okno namenjeno obveščanju uporabnika o napakah v poizvedbi in
- okno, ki uporabniku prikaže rezultate poizvedb.

Orodje omogoča uporabniku enostavno shranjevanje varnostnih kopij celotne podatkovne baze (vključno s shranjenimi procedurami), kar sem uporabil pri prenosu naročnikove podatkovne baze v bazo, ki jo uporablja aplikacija. Poleg tega omogoča tudi neposredno testiranje shranjenih procedur, tako da uporabniku ni potrebno testirati njihovega delovanja preko programske kode.

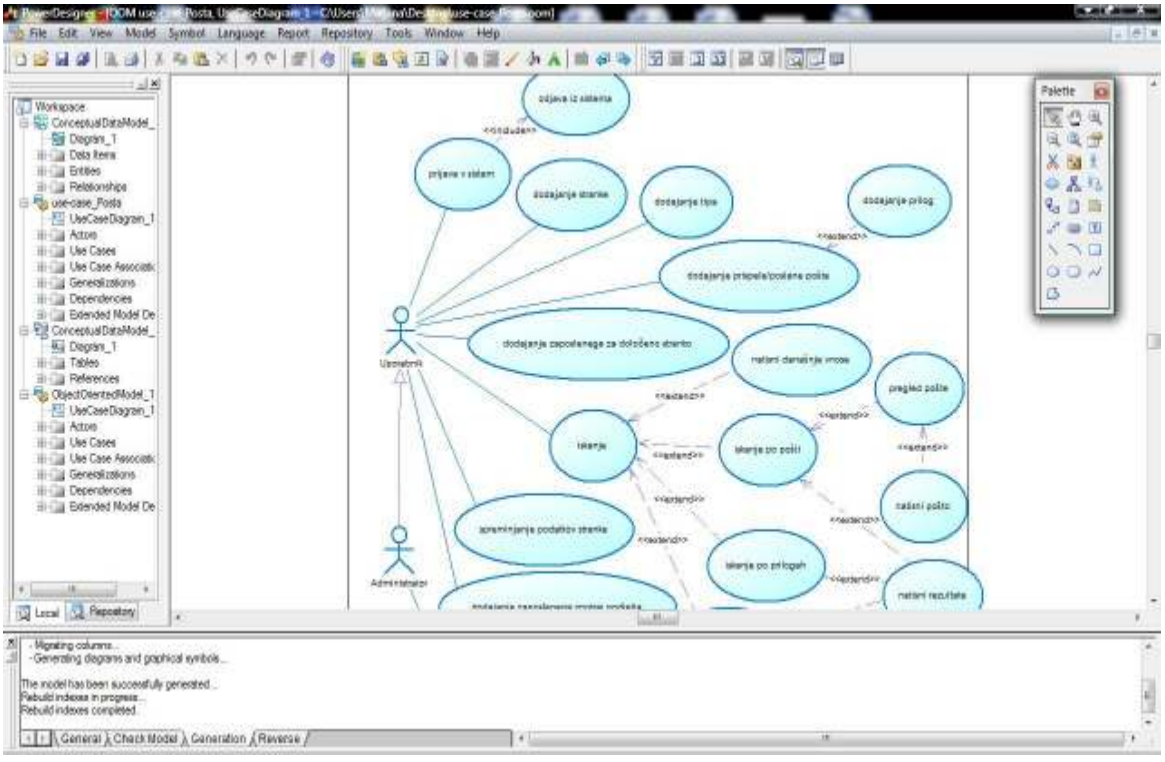


Slika 2.2: Orodje SQL Management Studio

2.8 PowerDesigner

PowerDesigner (slika 2.3) je grafično orodje namenjeno modeliranju podatkovnih baz [6]. Omogoča nam enostavno kreiranje različnih modelov. Med najbolj znanimi modeli so:

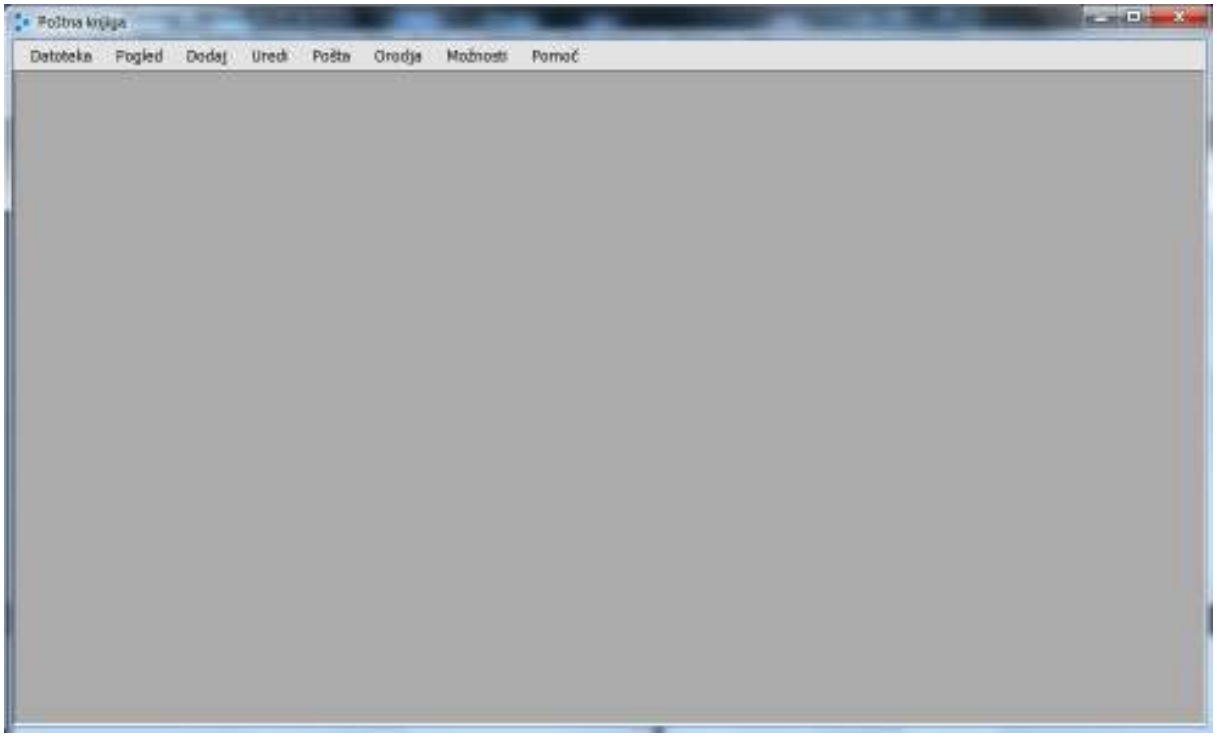
- konceptualni podatkovni model (angl. Conceptual Data Model) za modeliranje logične strukture podatkov,
- fizični podatkovni model (angl. Physical Data Model) za modeliranje fizične strukture podatkov,
- objektno usmerjeni model (angl. Object Oriented Model), ki predstavlja grafično analizo sistema z uporabo simbolov jezika UML in
- poslovni procesni model (angl. Business Process Model), ki omogoča grafični prikaz delovanja podjetja in kroženje informacij med dejavnostmi.
- Poleg modelov omogoča orodje izdelavo tudi različnih diagramov in sicer diagram podatkovnih tokov, primerov uporabe, zaporedij, razredni, objektni, komponentni, aktivnosti, stanj, postavitev in komunikacije.



Slika 2.3: Orodje PowerDesigner

3 Razvoj aplikacije

Vedno več podjetij se dandanes odloča za uporabo novih aplikacij, saj jim te omogočajo različne funkcionalnosti, ki uporabniku poenostavijo nadzor nad podatki, omogočajo hiter in enostaven pregled podatkov ter naredijo uporabo aplikacije bolj zabavno. Pri izdelavi aplikacije (slika 3.1) sem bil zadolžen za izdelavo funkcionalnosti iskalnika, različna dodajanja podatkov v podatkovno bazo, implementiranje prijave v sistem, prenosa obstoječe podatkovne baze naročnika ter še nekaterih manjših funkcionalnosti. V nadaljevanju sledi opis načina izdelave ter sama funkcionalnost obsežnejših delov aplikacije.



Slika 3.1: Glavno okno aplikacije Poštna knjiga

3.1 Opis obstoječe aplikacije

Predhodna aplikacija, namenjena arhiviranju prispele in odhodne pošte je stara več kot 10 let. V podjetju smo imeli priložnost za kratek čas aplikacijo tudi uporabljati in ugotovili smo, da je zelo nepregledna, njena uporaba pa je posledično bistveno otežena. Vsebovala je neuporabne funkcionalnosti, določene stvari niti niso delovale. Nekaterih uporabnih funkcionalnosti, kot sta na primer hranjenje celotne vsebine pošte in dodajanje neomejenega števila prilog posamezni pošti, aplikacija ni imela, zato je bilo v podjetju potrebno shranjevanje vseh papirjev.

3.2 Analiza zahtev

Naročniki imajo vedno svoj pogled na aplikacijo in tako tudi svoje zahteve, ki je potrebno pri izdelavi aplikacije upoštevati. Naročnik je tisti, ki razvijalcem razloži svoje želje po začetnih funkcionalnostih in funkcionalnostih, ki jih pričakujejo v prihodnosti. Po analizi vseh zahtev naročnika je potrebno funkcionalnostim določiti prioriteto. To razvijalec stori zaradi mogočih odvisnosti med različnimi funkcionalnostmi in nesmiselnostjo razvijanja nekaterih funkcionalnosti pred ostalimi.

Uspešnost izvedbe projekta je odvisna predvsem od kvalitete načrtovanja, saj nam uspešen načrt lahko prihrani ogromno napak in predvsem časa.

Zahteve s strani naročnikov so bile:

- aplikacija mora delovati na Microsoftovih operacijskih sistemih,
- prenos obstoječe podatkovne baze v bazo nove aplikacije,
- biti mora enostavna za uporabo,
- imeti mora možnost kvalitetnega iskanja po podatkih,
- omogočati mora printanje rezultatov iskanja,
- omogočati mora dodajanje neomejenega števila prilog posamezni pošti in
- biti mora skalabilna za dodajanje novih funkcionalnosti v prihodnosti.

3.3 Načrtovanje sistema

Po analizi naročnikovih zahtev sledi najpomembnejši del projekta, načrtovanje. Med načrtovanje uvrščamo izdelavo modelov za podatkovno bazo, prikaz možnih funkcionalnosti posameznih akterjev v sistemu ter še mnoge majhne podrobnosti. Vse to je prvi korak do pravočasne, uspešne in kvalitetne realizacije projekta.

Vsaka napaka pri načrtovanju je lahko vzrok za ponovno generiranje celotne podatkovne baze, zato si je dobro vzeti čas in pregledati vse podrobnosti, ki bi lahko to povzročile.

Pri načrtovanju smo se odločili za izdelavo diagrama primera uporabe ter konceptualnega in fizičnega podatkovnega modela.

3.3.1 Primer uporabe

Glavni namen diagrama primera uporabe (ang. use-case diagram) je predstaviti komunikacijo med uporabniki in sistemom. Iz diagrama je razvidno katere funkcionalnosti so posameznemu akterju omogočene oziroma onemogočene. Diagram primera uporabe vsebuje pet osnovnih gradnikov [7]:

- akter,
- primer uporabe,
- povezava med akterjem in primerom uporabe,
- povezava med dvema primeroma uporabe,
- povezava med dvema akterjema in
- sistem oziroma podsistem.

Diagram primera uporabe temelji na osnovi razgovora z naročniki. Po analizi omenjenega diagrama je bralec, zaradi enostavne grafične podobe diagrama, seznanjem s celotnim delovanjem aplikacije. Vsak primer uporabe mora biti dostopen iz vsaj enega akterja, ni pa nujno, da je tudi neposredno povezan z vsaj enim akterjem.

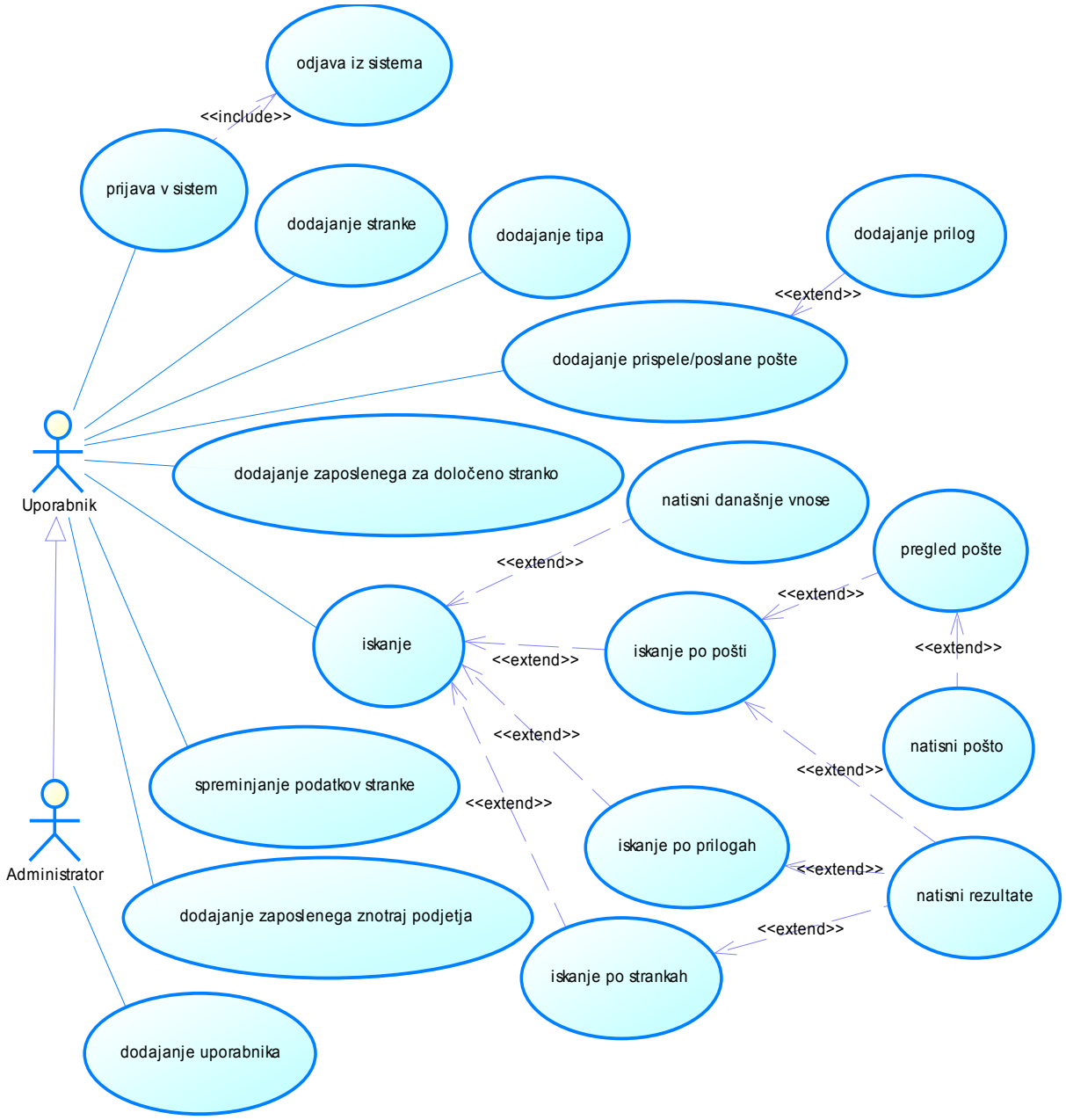
Povezava med dvema primeroma uporabe je lahko stereotipa vključuje (ang. include) oziroma razširja (ang. extend). V primeru, da primer uporabe A razširja primer uporabe B, to pomeni, da primer uporabe A lahko sledi primeru uporabe B, ni pa nujno. Stereotip vključuje, ko primer uporabe A vključuje primer uporabe B, pomeni, da primeru uporabe A vedno sledi tudi primer uporabe B.

Za lažje razumevanje in predstavo funkcionalnosti aplikacije sem diagram primerov uporabe tudi izdelal (slika 3.2).

Uporabnik ima naslednje funkcionalnosti:

- prijava v sistem (kasneje se mora tudi odjaviti oziroma zapreti program),
- dodajanje stranke,
- spreminjanje podatkov stranke,
- dodajanje tipa,
- dodajanje zaposlenega za določeno stranko,
- dodajanje zaposlenega znotraj podjetja,
- dodajanje prispele oziroma poslane pošte, kateri se lahko dodajo tudi priloge in
- iskanje, kjer lahko uporabnik natisne današnje vnose in uporabi iskalnik za iskanje po pošti, prilogah ter strankah. Pri vseh treh iskanjih ima uporabnik možnost natisniti rezultate iskanja.

Administrator generalizira uporabnika, poleg njegovih funkcionalnosti pa ima možnost še dodajanja novega uporabnika aplikacije.



Slika 3.2: Diagram primerov uporabe

3.3.1.1 Konceptualni podatkovni model

Konceptualni podatkovni model je dandanes skoraj nujno potreben pri načrtovanju. Model grafično prikazuje strukturo celotne podatkovne baze. Glavni gradniki konceptualnega podatkovnega modela so:

- entitete – predstavljajo posamezne instance tipov objektov. Vsebujejo lahko enolični primarni ključ ter več tujih ključev in poljubno število ostalih atributov, ki v podatkovni bazi predstavljajo lastnost posameznega vnosa. Vsak izmed atributov je tipiziran, saj to zagotavlja večjo varnost
- relacije med dvema entitetama – predstavljajo razmerje med entitetama. Razmerje med entitetama je predstavljeno s povezavo med entitetama, ki ima na vsakem koncu tudi svoj simbol. Najbolj znani simboli so:
    - ena,
    - nič ali ena,
    - nič ali več,
    - ena ali več.

Prednost izdelave konceptualnega modela je, da nam nekatera orodja, med katera spada tudi PowerDesigner, omogočajo avtomatsko pretvorbo iz konceptualnega podatkovnega modela v fizični podatkovni model. Pri tem nas v primeru napak na te tudi opozori, da jih lahko še pravočasno odpravimo.

Izdelan konceptualni model, ki je prikazan na sliki 3.3, vsebuje enajst entitet in sicer:

customers – predstavlja stranke, ki pošiljajo oziroma prejemajo pošto. Entiteta vsebuje enolični identifikator custID, tuj ključ postID ter ostale atribute, ki vsebujejo različne informacije o stranki.

custEmployees – predstavlja zaposlene pri določeni stranki. Poleg primarnega ključa custEmpID tako vsebuje tudi tuj ključ custID, ki pove v katerem podjetju je zaposlena oseba z določenim primarnim ključem. Entiteta vsebuje tudi atribute custEmpFirstName (ime zaposlenega), custEmpLastName (priimek zaposlenega) ter custEmpEmail (email zaposlenega).

post – je zbrirka vseh poštnih številk, ki jih je naročnik imel v njihovi podatkovni bazi. Vsebuje primarni ključ postID in atribute postNr (poštna številka, ki ni integer zaradi določenih poštnih številk, ki vsebujejo znake, ki niso številke), postCity (kraj pošte) in postCountry (država pošte).

incomingMail – vsebuje podatke o prispeli pošti. Entiteta vsebuje primarni ključ inMailID ter atribute:

- typeID (tuj ključ iz entitete mailType, ki predstavlja tip prispele pošte),
- custID (tuj ključ iz entitete customers, ki predstavlja stranko, ki pošilja),
- usrID (tuj ključ iz entitete users, ki nam pove kdo je vnesel to pošto),
- mStatID (tuj ključ iz entitete, ki je namenjen za naslednje različice programa),

- empID (tuj ključ iz entitete employees, ki nam pove kdo je prejemnik, atribut je lahko tudi null),
- inMailSubject (predstavlja zadevo prispele pošte),
- inMailContent (predstavlja celotno vsebino prispele pošte),
- inMailReceiveDate (datum prispele pošte),
- inMailEntryDate (datum vnešene pošte),
- inMailSender (pošiljatelj znotraj stranke, ki ni obvezen) in
- zapSt (enolični identifikatorji starih prispelih pošt).

outgoingMail – vsebuje podatke o poslani pošti. Struktura entitete je zelo podobna entiteti incomingMail. Razlikuje se le v tem, da ima svoj primarni ključ outMailID, empID predstavlja pošiljatelja znotraj podjetja, ki to aplikacijo uporablja (atribut je lahko null), outMailSentDate ima shranjen datum poslane pošte in outMailReceiver ima shranjeno osebo, ki je prejemnik te pošte (atribut je lahko prazen).

mailType – vsebuje zbirko vseh možnih tipov pošte. Primarni ključ entitete je typeID, atribut pa typeDesc, ki vsebuje opis tipa.

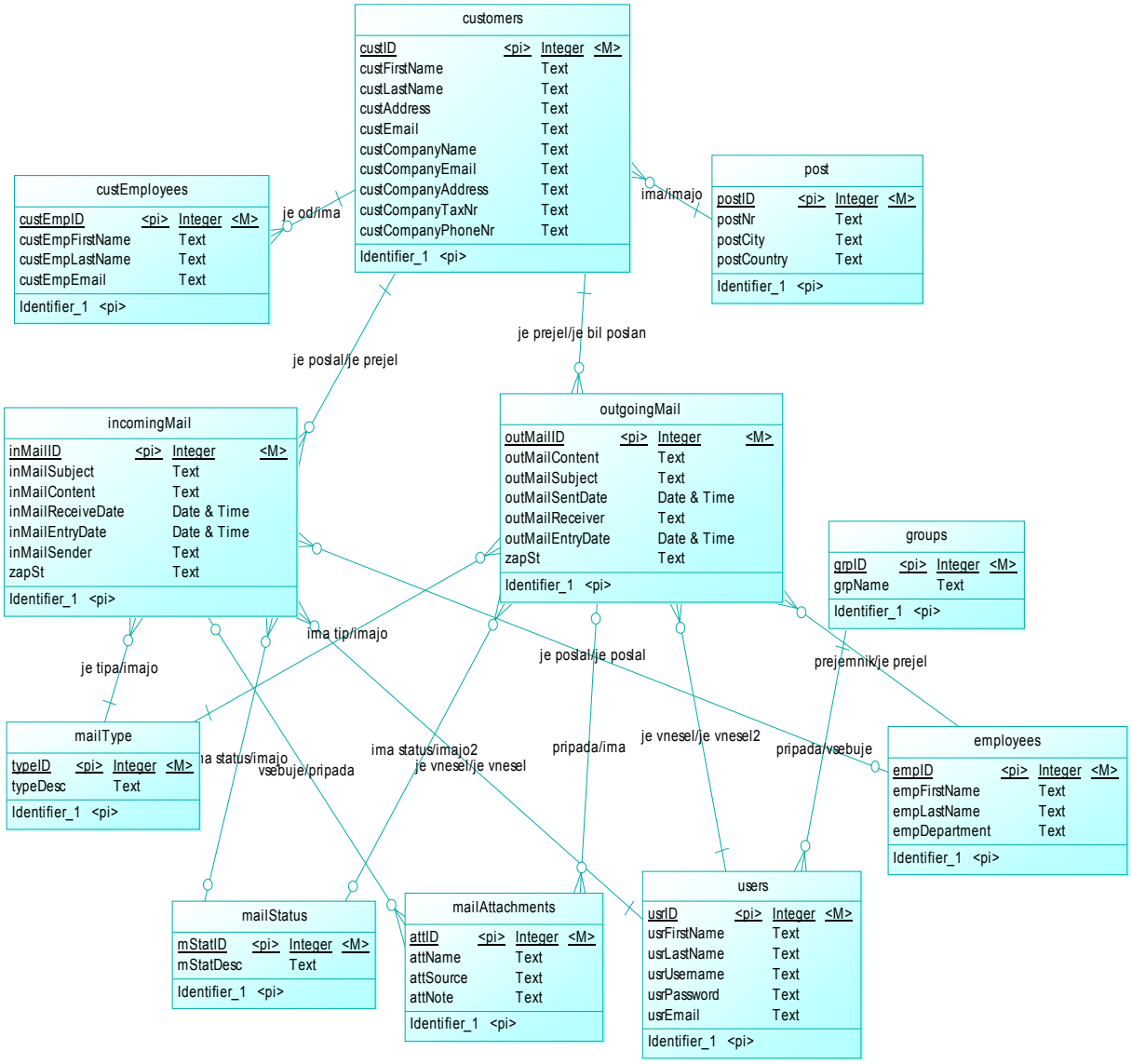
mailStatus – entiteta vsebuje primarni ključ mStatID ter atribut mStatDesc.

mailAttachments – Entiteta je namenjena shranjevanju podatkov o prilogah. Sestavljajo jo enolični identifikator attID, tuja ključa outMailID in inMailID (vedno je natanko eden null) ter atributi attName (ime datoteke), attSource (pot do datoteke) in attNote (opomba priloge).

users – entiteta v kateri so zabeleženi uporabniki aplikacije. Poleg enoličnega identifikatorja usrID vsebuje še tuj ključ grpID in atribute usrFirstName (ime uporabnika), usrLastName (priimek uporabnika), usrUsername (uporabniško ime uporabnika), usrPassword (kriptirano geslo uporabnika) in usrEmail (email naslov uporabnika).

employees – entiteta hrani podatke o zaposlenih znotraj podjetja, ki to aplikacijo uporablja. empID enolično določa zaposlenega, poleg tega vsebuje tudi atribute empFirstName (ime zaposlenega), empLastName (priimek zaposlenega) in empDepartment (področje zaposlenega).

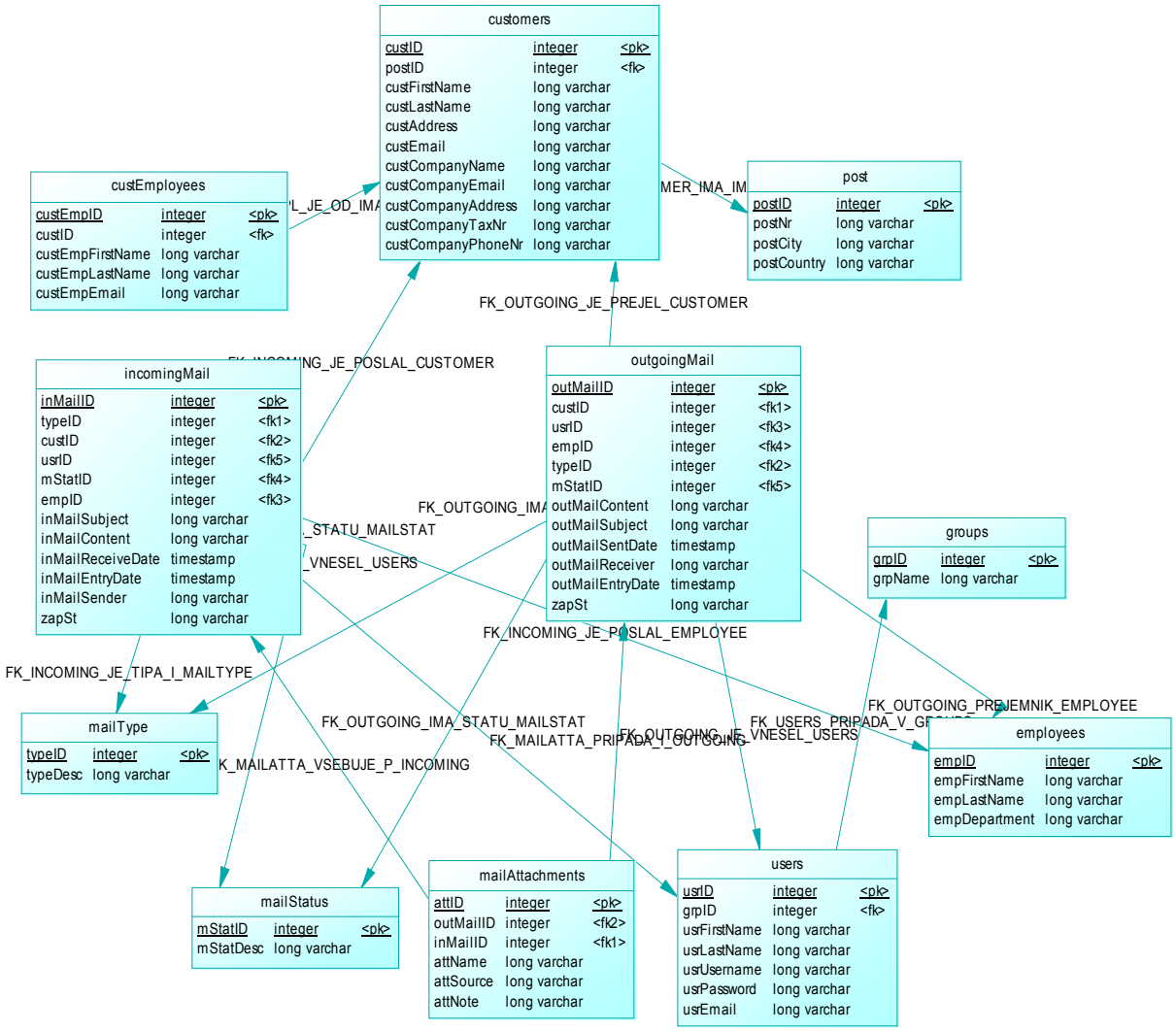
groups – hrani skupine v katere lahko spada posamezen uporabnik. Določena je s primarnim ključem grpID in atributom grpName, ki predstavlja ime skupine.



Slika 3.3: Konceptualni podatkovni model aplikacije

3.3.1.2 Fizični podatkovni model

Fizični podatkovni model (slika 3.4) prikazuje podatke tako, kot so prikazani v sami podatkovni bazi. Za razliko od konceptualnega modela so v njem prikazani tudi tuji ključi. Opisani model sem s pomočjo orodja PowerDesigner generiral iz konceptualnega. Orodje PowerDesigner nam omogoča tudi kreiranje datoteke, s katero lahko generiramo podatkovno bazo podanega fizičnega modela, za različne vrste podatkovnih baz. Pri razvijanju aplikacije sem to funkcionalnost tudi uporabil.



Slika 3.4: Fizični podatkovni model aplikacije

3.4 Izdelava funkcionalnosti

V podjetju smo se morali odločiti kako bi poimenovali projekt. Pri izbiri imena smo bili enotnega mnenja, da mora biti le-to preprosto in tako smo se odločili, da nov projekt, ki smo ga kreirali v orodju Visual Studio, poimenujemo Poštna knjiga. Projekt je kreiral sodelavec ter ga postavil na SVN strežnik podjetja. Iz strežnika sem moral projekt prenesti na svoj računalnik preko programa Syncro SVN, ki sem ga uporabljal tekom projekta. Podatkovno bazo smo, kot sem že omenil v načrtovanju, kreirali v grafičnem orodju PowerDesigner.

Pred začetkom razvoja je bilo potrebno določiti zaporedje funkcionalnosti, ki smo jih imeli namen razviti. Brez podatkov v podatkovni bazi je težko testirati posamezne funkcionalnosti, zato smo se odločili, da je najbolje najprej prenesti obstoječo podatkovno bazo naročnika v podatkovno bazo naše aplikacije. Sledila je implementacija prijave v sistem, izdelava form za dodajanje različnih podatkov v podatkovno bazo ter izdelava iskalnika. Celoten postopek izdelave bom predstavil v nadaljevanju.

3.4.1 Prenos podatkovne baze

Zaradi zahteve naročnika po ohranitvi obstoječih podatkov smo se odločili za prenos naročnikove podatkovne bazo v bazo aplikacije. Pri prenosu sem moral biti pazljiv, saj nekatere tabele vsebujejo tuje ključe, kar pomeni, da te tabele potrebujejo podatke iz drugih tabel. Tako sem si sestavil zaporedje prenosov in mu tudi sledil.

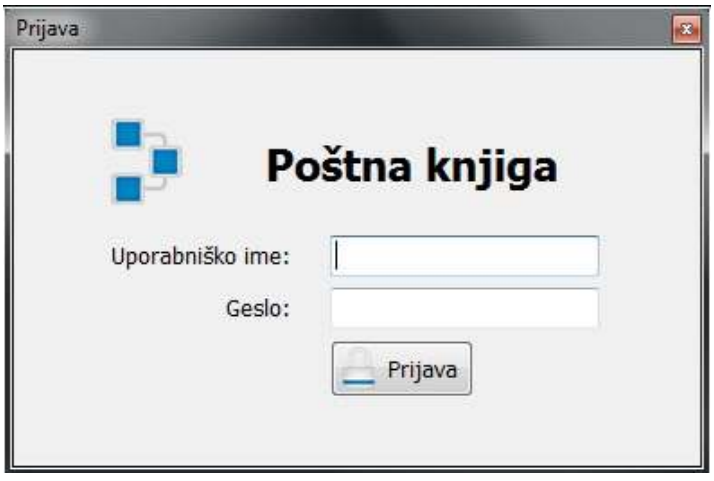
Nakar sem začel s kodiranjem samega prenosa. Izdelal sem začasno okno v aplikaciji preko katerega sem uvozil posamezne tabele. Pri prenosu podatkov o uporabnikih, zaposlenih znotraj podjetja, ki uporabljajo aplikacijo, poštnih številkah in tipih pošte nisem imel nobenih problemov. Prvi problem se je pojavil pri prenosu strank naročnika, saj prejšnja aplikacija ni ločevala podjetij in fizičnih oseb, kar je sicer programersko zelo težko, predvsem zaradi tega, ker ni enotnega pravila (npr. d.o.o.), ki bi zapovedovalo kam naj stransko uvrstimo. Tako smo vedeli, da bo potrebno ročno pregledati vse osebe in jih po potrebi uvrstiti v drugo skupino. Za nekoliko lažje delo pri pregledovanju strank sem najprej imena strank spremenil v male črke ter nato razporedil stranke, ki imajo znotraj imena d.o.o, d.n.o, z.o.o, k.d, s.p, d.d in s.r.o med podjetja, ostale sem pa uvrstil med fizične osebe. Pri prenosu sem moral biti pozoren na to, da je primarni ključ posameznega podjetja enak primarnemu ključu iz stare podatkovne baze, saj sta prispela in poslana pošta vsebovala tako ta ključ kot tudi tuj. Postavil sem števec, ki je vseboval zaporedno številko primarnega ključa, ki ga je potrebno dodati na ena in se povezal na staro podatkovno bazo. Za vsako vrstico v stari podatkovni bazi sem prebral primarni ključ in dokler prebrani ključ ni bil enak števcu sem dodajal prazne vnose v novo podatkovno bazo ter števec povečeval. S tem sem zagotovil, da so vsi primarni ključi podjetij pravilni. Zaradi mogoče vsebovanosti stranke, ki sem jo dodal kot prazen vnos, v tabelah prispela in poslana pošta sem te prazne vnose zaenkrat pustil v tabeli strank. Sledil je prenos prispele in poslane pošte. V stari podatkovni bazi so prispele pošte imele stolpec »Tip« nastavljen na številko ena, medtem ko je imela poslana pošta isti stolpec nastavljen na številko dva, kar je omogočilo enostavno ločevanje prispele in poslane pošte. Pošta je bila enolično določena s tremi stolpci in sicer s stolpcem »Tip«, stolpcem »ZapSt1«, ki je predstavljal zaporedno številko poslane oziroma prejete pošte v določenem letu, ter stolpcem

»PoslovnoLeto«, ki je vseboval štirimestni zapis leta v katerem je bila pošta vnesena. Odločili smo se, da bosta tabeli prejetih in poslanih pošt v novi podatkovni bazi vsebovali vsaka svoj primarni ključ in sicer inMailID za prispelo ter outMailID za poslano pošto. Naročnik je v starem programu med drugim imel možnost iskanja pošte z izbiro tipa pošte ter vnosom zaporedne številke pošte in poslovnega leta. Odločili smo se, da bosta tabeli incomingMail ter outgoingMail vsebovali dodaten stolpec z imenom zapSt, ki je vseboval vsebino stolpcev »ZapSt« in »PoslovnoLeto« ločeno s pomišljajem. Razlog za to odločitev je bil, da bo uporabnik aplikacije lahko preko novega iskalnika iskal pošte, ki so bile vnešene v stari aplikaciji, na isti način, kot je to počel do sedaj. Sledila je izvedba prenosa med katerim sem ugotovil, da imajo nekatere podatke v podatkovni bazi narobe vnešene, saj so imele nekatere prispele pošte izpolnjen datum poslane pošte namesto prispele. Te primere sem nato tudi programersko odpravil ter bazo ponovno kreiral. Tabeli custEmployees, mailAttachments sem pustil prazni, saj stara aplikacija ni shranjevala teh podatkov oziroma jih naročnik ni nikoli vpisoval. V tabeli groups in mailStatus sem ročno vnesel podatke, ker nimajo nobene povezave s prenosom podatkovne baze.

Nova podatkovna baza z vsemi podatki nam je omogočila lažje in bolj učinkovito testiranje vseh funkcionalnosti, ki jih bom v nadaljevanju tudi predstavil.

3.4.2 Prijava v sistem

Ob zagonu aplikacije se uporabniku odpre okno (slika 3.5), kjer se lahko z vnosom uporabniškega imena in gesla tudi prijavi v sistem. Tekstovno polje za vnos gesla skrije vpisano besedilo z znaki '*', kot je to običajno pri geslih zaradi varnosti. Po vnosu uporabnika se kliče funkcija, ki vnešeno geslo kriptira. Zatem se vneseno uporabniško ime ter novo geslo posreduje funkciji checkCredentials, ki vrne število uporabnikov v podatkovni bazi s posredovanimi podatki. V kolikor uporabnik s temi podatki ne obstaja, aplikacija uporabnika opozori na napačen vnos in mu omogoči ponoven vnos. V primeru pravilnega vnosa se uporabniku odpre glavno okno aplikacije, kjer lahko prične z delom.



Slika 3.5: Okno za prijavo v sistem

3.4.3 Glavno okno aplikacije

Po uspešni prijavi v sistem se uporabniku odpre glavno okno, preko katerega lahko doda nov tip, stranko, zaposlenega pri izbrani stranki, zaposlenega znotraj podjetja, prispelo in poslano pošto. Uporabnik ima tudi možnost iskanja po treh kategorijah in sicer:

- po prispeli oz. poslani pošti,
- po prilogah in
- po strankah.

Pri iskanju prejete in poslane pošte si lahko uporabnik tudi podrobno ogleda posamezno pošto ter njene podatke po želji tudi natisne.

Vse to lahko uporabnik počne preko orodne vrstice.

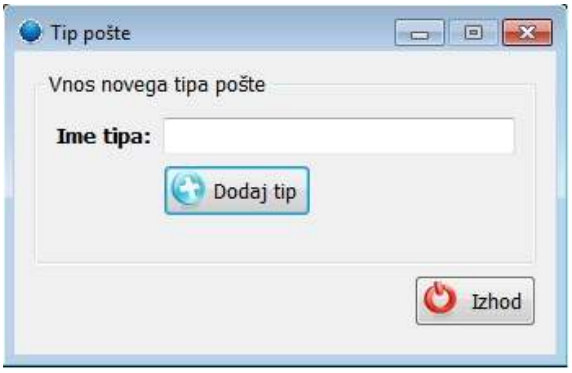
3.4.3.1 Dodajanje tipa, stranke ter uporabnika

Preko elementa orodne vrstice »Dodaj« ima uporabnik možnost dodati:

- stranko ter uslužbenca izbrane stranke,
- uporabnika aplikacije,
- zaposlenega znotraj podjetja, ki to aplikacijo uporablja in
- tip pošte.

Dodajanje tipa pošte

Pri dodajanju prispele in poslane pošte je potrebno izbrati tip pošte. V kolikor željenega tipa ni v podatkovni bazi, ga lahko uporabnik ročno doda preko okna za dodajanje novega tipa pošte (slika 3.6). V okno mora uporabnik vnesti le ime novega tipa ter klikniti na gumb »Dodaj tip«. Dogodek, ki se sproži na klik omenjenega gumba kliče funkcijo, ki se poveže na podatkovno bazo ter preko shranjene procedure insertType doda nov tip v bazo.



Slika 3.6: Okno za dodajanje novega tipa pošte

Dodajanje in urejanje stranke

Podjetja prejemajo in pošiljajo pošto različnim strankam, tj. strankam s katerimi so že bili v kontaktu kot tudi novim. Tako smo uporabniku aplikacije morali omogočiti tudi dodajanje nove stranke. Do okna za dodajanje strank (slika 3.7) lahko uporabnik pride preko elementa »Dodaj«, ki se nahaja v orodni vrstici. Zaradi dogovora z naročnikom o delitvi strank na podjetja in fizične osebe smo uporabniku omogočili izbiro vrste stranke. Glede na izbiro stranke je uporabnik moral izpolniti posamezna polja, ki so bila obvezna. Za obvezna polja smo se odločili, ker bi pripomogla k uporabnosti iskanja po strankah, saj je iskanje po določenih podatkih strank neuporabno v kolikor polovica strank teh podatkov v podatkovni bazi nima. Uporabnik mora, po izpolnitvi vseh polj, za dodajanje stranke v podatkovno bazo klikniti še na gumb »Dodaj«. Ob napačnem vnosu katerega izmed polj uporabnika aplikacija na to tudi opozori.

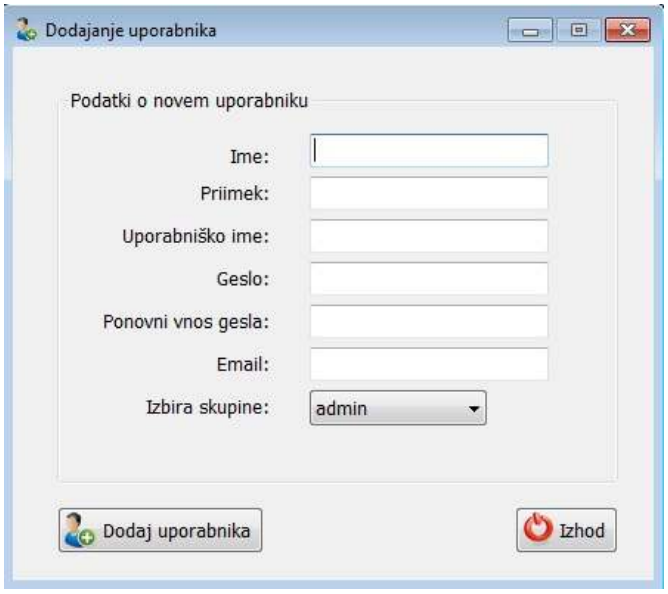
V istem oknu sem uporabniku omogočil urejanje podatkov obstoječe stranke, saj se lahko določeni podatki o stranki spremenijo. Za urejanje stranke mora uporabnik v za to namenjeno tekstovno polje vpisati ime stranke. To tekstovno polje ima omogočeno avtomatsko predlaganje stranke glede na trenutni vpis. Avtomatsko predlaganje sem implementiral preko Visual Studia, saj ima to funkcionalnost že vgrajeno. Po vpisu oziroma izbiri željene stranke mora uporabnik klikniti gumb »Prikaži podrobnosti«. Ob kliku se kliče funkcija, ki najprej prepozna ali gre pri vneseni stranki za podjetje ali fizično osebo ter vnese podatke v ustrezna polja. Uporabnik lahko spremeni vse podatke stranke z izjemo imena. Po končanem spreminjanju mora uporabnik klikniti na gumb »Shrani«, ki te spremembe shrani v bazo. V primeru, da sprememb ne želi shraniti ima možnost klika na gumb »Prekliči«.

Slika 3.7: Okno za dodajanje oz. urejanje stranke

Dodajanje uporabnika aplikacije

Uporabnika aplikacije lahko doda le tisti uporabnik, ki spada v skupino administratorjev. Administrator mora v okno za dodajanje uporabnika, ki je prikazano na sliki 3.8, vnesti ime in priimek novega uporabnika, uporabniško ime novega uporabnika, željeno geslo ter ponovitev gesla, elektronski naslov in izbrati skupino v katero bo nov uporabnik uvrščen.

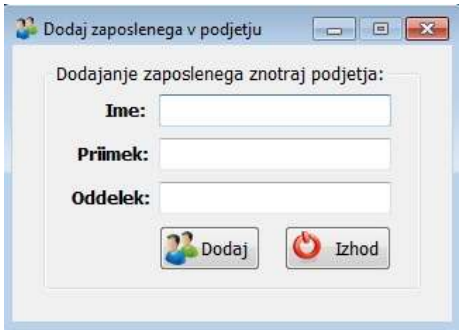
Po vnosu vseh podatkov mora administrator klikniti na gumb »Dodaj uporabnika«. Ob napačnem vnosu ponovitve gesla oziroma kateregakoli drugega vnosnega polja administratorja na to opozori aplikacija.



Slika 3.8: Okno za dodajanje novega uporabnika

Dodajanje zaposlenega znotraj podjetja

Namen dodajanja zaposlenega znotraj podjetja, ki aplikacijo uporablja, je, da lahko to osebo pri dodajanju pošte označimo za pošiljatelja oziroma prejemnika pošte. Zato smo izdelali okno (slika 3.9) preko katerega lahko uporabnik doda sodelavca.



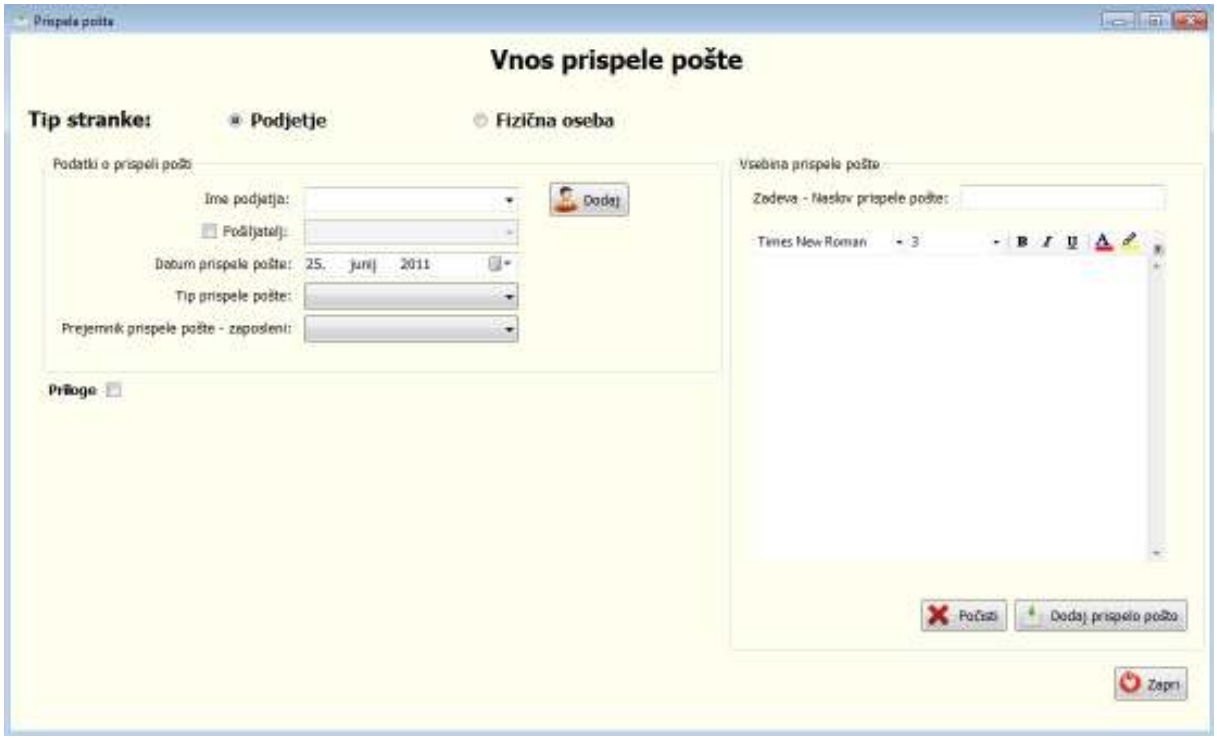
Slika 3.9: Okno za dodajanje zaposlenega znotraj podjetja

3.4.3.2 Dodajanje prispele in poslane pošte

Vse več podjetij se odloča za arhiviranje pošte z uporabo različnih aplikacij, saj jim to omogoča, da si v primeru izgube papirjev le-te ponovno natisnejo. Aplikacija ima tako možnost arhiviranja pošte, ki prispe v podjetje oziroma je poslana iz podjetja.

Dodajanje prispele pošte

Podjetje lahko prejme pošto po pošti ali na elektronski naslov. V obeh primerih mora uporabnik aplikacije podatke prejete pošte vnesti, saj ima podjetje le tako lahko arhivirano vso pošto. Za dodajanje prispele pošte mora uporabniki odpreti okno (slika 3.10), do katerega pride preko elementa »Pošta«, ki se nahaja v orodni vrstici.



Slika 3.10: Okno za dodajanje prispele pošte

Tam mora uporabnik najprej izbrati ali gre za podjetje ali za fizično osebo (privzeto je podjetje). Nato mora izpolniti še naslednje podatke o pošti:

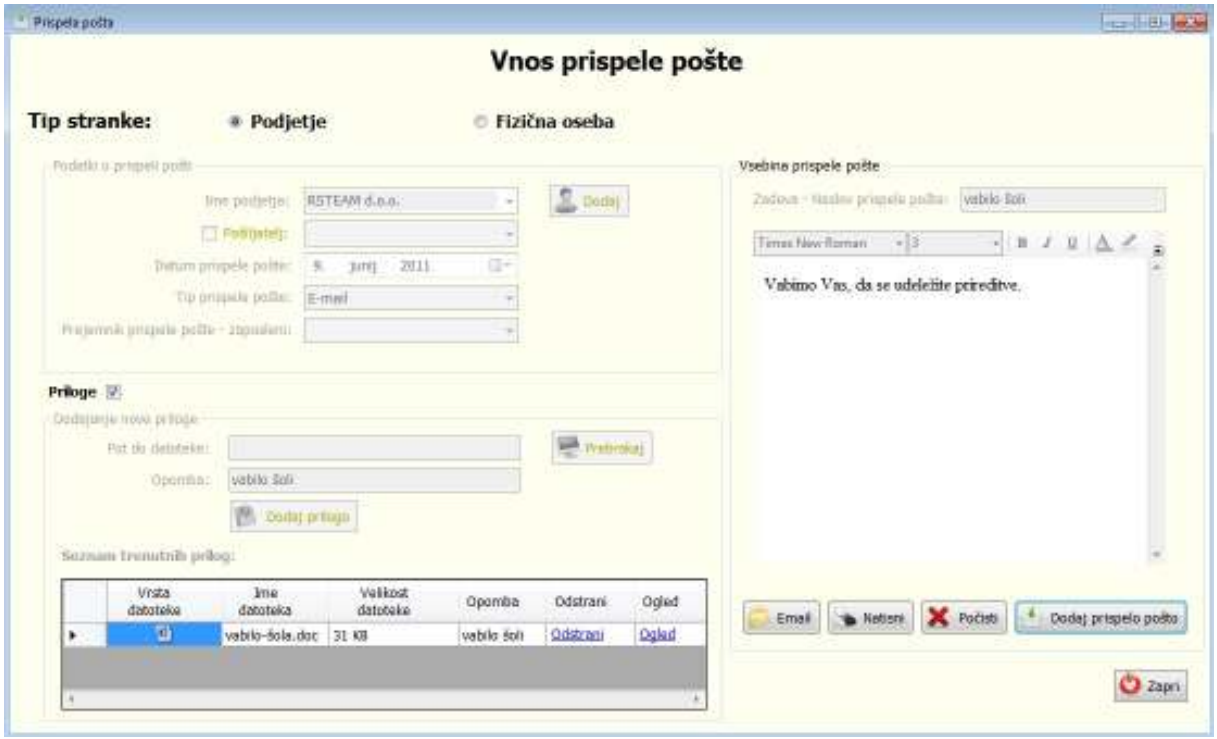
- Ime podjetja oziroma fizične osebe - uporabniku se med vnosom imena avtomatsko predlagajo podjetja oziroma fizične osebe, ki vnešenemu imenu ustrezajo. Število predlaganih rezultatov je omejeno na šestnajst. V kolikor stranka še ne obstaja ima uporabnik zraven gumb za dodajanje stranke. Med dodajanjem nove stranke se okno za dodajanje prispele pošte onemogoči dokler se dodajanje stranke ne zaključi. V

primeru, da je bila stranka uspešno dodana, se avtomatsko vpiše v polje okna prispela pošta, v nasprotnem pa se vsebina polja ne spremeni.

- Pošiljatelj – v primeru, da je pošto poslala točno določena oseba iz izbranega podjetja, je potrebno obkljukati pošiljatelja ter ga izbrati izmed spodaj naštetih. V kolikor pošiljatelja ni med zaposlenimi izbranega podjetja ga je potrebno ročno dodati.
- Datum prispele pošte – izbere uporabnik in predstavlja datum, ko je pošta prispela v podjetje oziroma na elektronski poštni naslov. Datum je privzeto nastavljen na današnji, saj se pošta največkrat vnese kar na dan dospelosti.
- Tip prispele pošte – uporabnik mora izbrati tudi tip prispele pošte. Izbira lahko med tipi pošte, ki so shranjeni v podatkovni bazi. V kolikor željenega tipa ni, ga mora uporabnik ročno dodati.
- Prejemnik prispele pošte – v primeru, da je pošta naslovljena na točno določenega uslužbenca iz podjetja, lahko uporabnik izbere prejemnika. V nasprotnem primeru se smatra, da je prejemnik podjetje.
- Zadeva pošte – kratek povzem bistva prispele pošte. Uporabnik je pri pisanju zadeve omejen na 50 znakov.
- Vsebina pošte – polje vsebuje urejevalnik podoben Microsoft Word-u. Polje je namenjeno vnosu celotne vsebine pošte. Prednost uporabe urejevalnika je, da vsebuje možnost vnosa slik, tabel, hiperpovezav, poudarka določenega dela vsebine ipd. Uporabnik lahko v primeru, da je pošto dobil po elektronski pošti, vsebino elektronske pošte prilepi v urejevalnik. Tako si prihrani nepotrebno izgubljen čas.
- Priloge – uporabnik lahko po želji doda tudi priloge k pošti. To stori tako, da obkljuka priloge. Ko uporabnik to izvede, se mu prikažejo dodatna polja (slika 3.11) za vnos posamezne priloge ter podatki o trenutnih prilogah. Za vsako prilogo mora uporabnik prilogo izbrati preko iskalnika datotek, ki se odpre ob kliku na gumb »Prebrskaj«. Po želji lahko uporabnik vpiše tudi opombo priloge, ki je koristna pri iskanju po prilogah. Po izpolnjenih poljih mora uporabnik klikniti na gumb »Dodaj prilogo«. Ob tem se priloga doda v datagridview, ki za vsako prilogo vsebuje ikono priloge, ime datoteke, velikost datoteke in opombo. Poleg teh podatkov lahko vsako prilogo tudi odpremo za vpogled oziroma odstranimo iz seznama prilog. Po vsaki dodani prilogi nam aplikacija izpiše, da smo jo uspešno dodali.

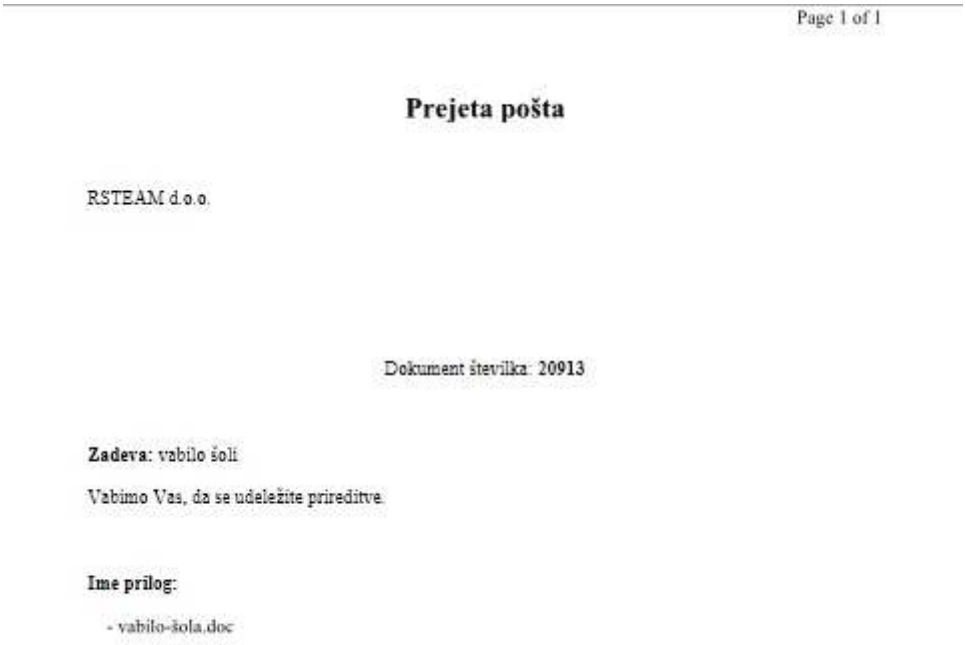
Slika 3.11: Uporabnik lahko pošti doda priloge

Po izpolnjenih vseh obveznih poljih lahko uporabnik doda prispelo pošto v podatkovno bazo. To stori s klikom na gumb »Dodaj prispelo pošto«. V primeru, da je uporabnik napačno izpolnil eno izmed polj ga na to opozori aplikacija. Po uspešno dodani pošti aplikacija uporabniku sporoči enolični identifikator pošte s pomočjo katerega lahko kasneje preko iskalnika pride do te pošte. Uporabniku se ob tem tudi blokirajo vsa polja, kot je prikazano na sliki 3.12, saj spremembe niso več mogoče.



Slika 3.12: Uporabniku se po vnosu onemogočijo vsa polja

Dodano prispelo pošto lahko uporabnik natisne s klikom na gumb »Natisni«. Pri tem se kliče funkcija, ki generira html niz, ki vsebuje vse podatke o pošti. Ta se nato vpiše v nevidno polje urejevalnika in kliče funkcijo za natisniti vsebino urejevalnika. Slika 3.13 prikazuje primer natisnjene vsebine prispele pošte. Poleg tega lahko uporabnik tudi s klikom na gumb »Email« pošlje vsebino pošte na elektronski naslov, ki ga aplikacija prebere iz konfiguracijske datoteke. Elektronski naslov pošiljatelja aplikacija ravno tako prebere iz konfiguracijske datoteke. Enako kot pri tiskanju se tudi tu generira html niz, ki se nato določi kot vsebino pri pošiljanju elektronske pošte. Pri pošiljanju preko elektronske pošte se pošljejo tudi vse priloge, ki jih je uporabnik dodal k pošti. Levo od gumba »Dodaj prispelo pošto« se nahaja tudi gumb »Počisti«, ki je uporaben pri dodajanju več prejetih pošt zaporedoma. S klikom nanj se vsebina vseh polj postavi na začetno stanje in vsa polja se omogočijo.



Page 1 of 1

Prejeta pošta

RSTEAM d.o.o.

Dokument številka: 20913

Zadeva: vabilo šoli

Vabimo Vas, da se udeležite prireditve.

Ime prilog:

- vabilo-šola.doc

Slika 3.13: Primer natisnjene vsebine prejete pošte

Dodajanje poslane pošte

Dodajanje poslane pošte smo naredili zelo podobno dodajanju prejete pošte z namenom, da se uporabnik lažje privadi na dodajanje pošte. Okno (slika 3.14) ima barvo ozadja drugačno s čimer želimo, da se uporabnik nauči, da ena barva pomeni dodajanje prispele pošte, druga pa poslane.

Slika 3.14:Okno za dodajanje poslane pošte

Sicer ima uporabnik napisano na vrhu ali gre za poslano ali za prejeto pošto, vendar smo mnenja, da se tovrstne stvari zlahka spregleda. Spremembe v primerjavi z dodajanjem prispele pošte so zelo majhne, saj smo spremenili le kar se logično spremeni. Zato mora uporabnik vnesti prejemnika iz podjetja kateremu se je pošta poslala (če želi), namesto datuma prejete pošte mora vnesti datum poslane pošte in vnesti pošiljatelja znotraj podjetja (če pošilja točno določena oseba).

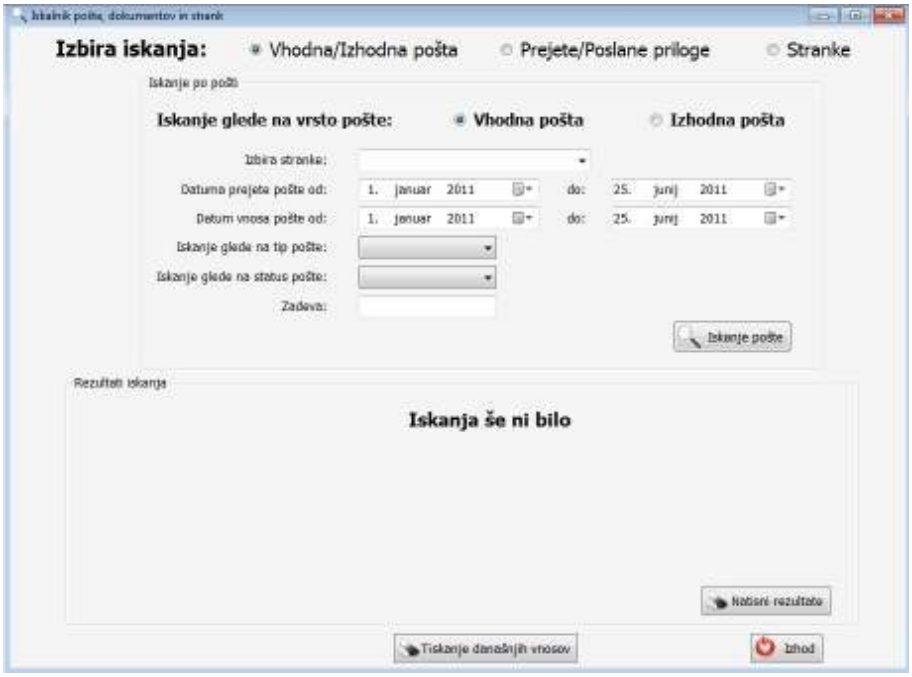
Ob pravilnem vnosu aplikacija izpiše enolični identifikator pravkar vnešene poslane pošte. Pri tem se, tako kot pri prispeli pošti, vsa polja onemogočijo za urejanje. Uporabnik lahko nato vsebino pošte natisne ali pošlje po elektronski pošti. V primeru, da želi uporabnik vnesti novo poslano pošto mora najprej klikniti na gumb »Počisti«, da se vsa polja postavijo v začetno stanje ter nato ponovi postopek. V nasprotnem primeru lahko okno zapre s klikom na gumb »Izhod«.

3.4.3.3 Iskanje in pregled podrobnosti pošte

Predhodna aplikacija je imela zelo zakompliciran iskalnik, zato smo mi stremeli k enostavnosti. Odločili smo se, da bo iskanje razdeljeno na tri dele in sicer:

- iskanje po pošti,
- iskanje po dokumentih (prilogah) in
- iskanje po strankah.

Okno za iskanje (slika 3.15) lahko uporabnik odpre preko elementa »Orodja« v orodni vrstici. Privzeto je prikazano iskanje po pošti. Med vsemi tremi vrstami iskanja uporabnik preklopi s klikom na željeno iskanje.



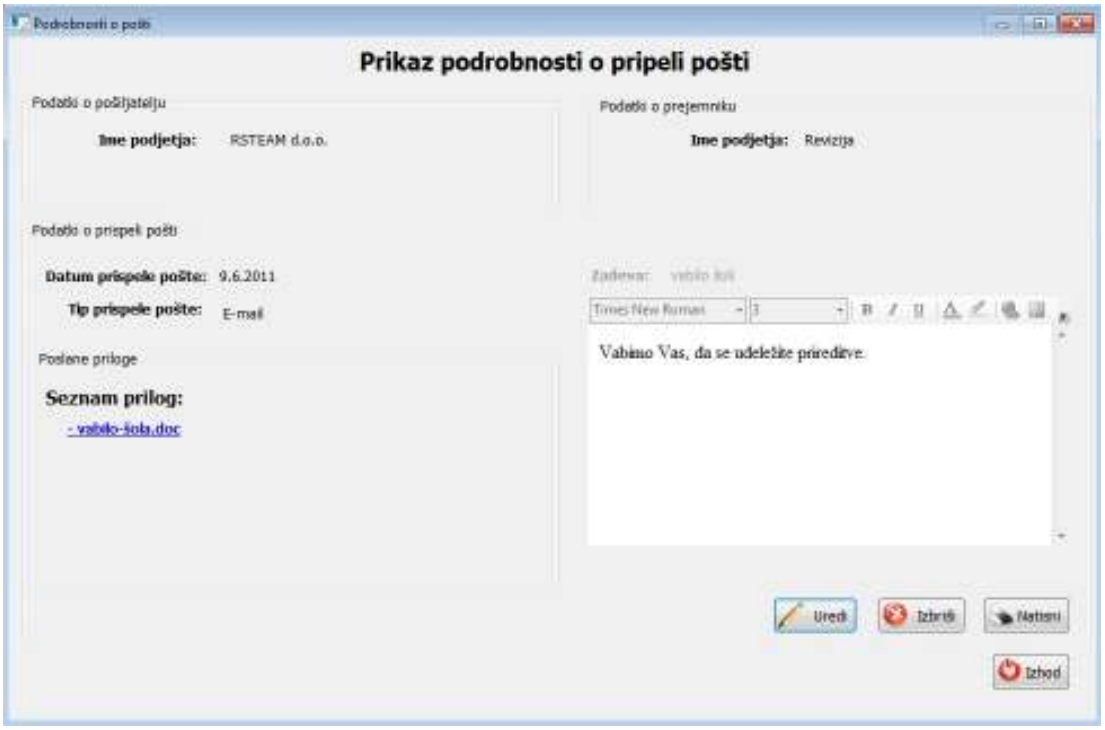
Slika 3.15: Iskanje po pošti je privzeto

Iskanje po pošti

Pri iskanju po pošti mora uporabnik najprej izbrati ali bo iskal po prejeti (vhodni) pošti ali po poslani (izhodni) pošti. Privzeto je izbrano iskanje po prejeti pošti. Vsa preostala polja niso obvezna, ima pa uporabnik možnost vpisati določeno stranko, katere pošto želi videti, interval datuma prejete pošte (privzeto je interval nastavljen na od 1. januarja trenutnega leta do vključno trenutnega dne), datum vnesene prispele pošte (privzet interval je enak kot pri datumu prejete pošte), izbrati tip in status pošte ter vpisati zadevo pošte. Pri zadevi se za zadetek šteje, če vsebuje zadeva pošte podniz vneseno zadevo. Po izbiri vseh željenih omejitev mora uporabnik klikniti na gumb »Iskanje pošte«. To sproži klic funkcije, ki pridobi ustrezne rezultate ter jih vnese v do sedaj nevidno tabelo, ki je prikazana na sliki 3.16.

Slika 3.16: Prikaz rezultatov iskanja pošte

V tabeli ima tako uporabnik prikazane vse rezultate poizvedbe z informacijami o enoličnem identifikatorju pošte, imenu stranke, ki je pošto poslala oziroma prejela ter zadevi in vsebini pošte. V kolikor mu te informacije niso dovolj podrobne, lahko pogleda podrobnosti vsake pošte. To stori tako, da klikne na hiperpovezavo »Podrobnosti« željene pošte. S klikom na hiperpovezavo se odpre novo okno (slika 3.17), kjer so zbrane vse informacije o izbrani pošti. Tako lahko uporabnik pošte pogleda natančne podatke o pošiljatelju ter prejemniku, datum, tip, zadevo, vsebino ter seznam vseh prilog izbrane pošte. S klikom na hiperpovezavo posamezne priloge se uporabniku izbrana priloga tudi odpre. Uporabnik ima na tem oknu tudi gumb »Natisni« preko katerega lahko vsebino prikazane pošte tudi natisne. V kolikor je v podatkih o pošti napaka jo lahko tudi odpravi, saj ima na voljo gumb »Uredi«, ki mu omogoča urejanje vseh podatkov o pošti z izjemo prilog, ki se jih po vnosu pošte ne more več spremeniti. S klikom na gumb »Uredi« se vsa polja uporabniku omogočijo in tako lahko spremeni pošiljateljeve in naročnikove podatke (da je pošiljatelj oziroma prejemnik podjetje v katerem je zaposlen seveda ni mogoče spremeniti), datum in tip ter zadevo in vsebino pošte. Po končanem spreminjanju podatkov mora uporabnik, če želi spremembe shraniti, klikniti gumb »Shrani«, saj ta povzroči, da vsi trenutni podatki pošte prepišejo obstoječe podatke v podatkovni bazi. Ob kliku na gumb »Shrani« postanejo zaradi varnosti vsa polja onemogočena.



Slika 3.17: Prikaz podrobnosti izbrane pošte

Iskanje po prilogah

Druga možnost iskanja je iskanje po prilogah oziroma dokumentih, ki je prikazano na sliki 3.18. Tudi pri tem iskanju mora uporabnik najprej izbrati ali bo iskal poslane ali prejete priloge, ki so izbrane privzeto. Iskanje lahko omeji z izbiro stranke, intervala med katerim je bila priloga poslana oziroma prejeta ter vnosom imena ter opombe priloge. Med iskanjem preverja pri imenu in opombi priloge podnize, pri datumu pa interval, v katerega sta vnešena datuma vključena.

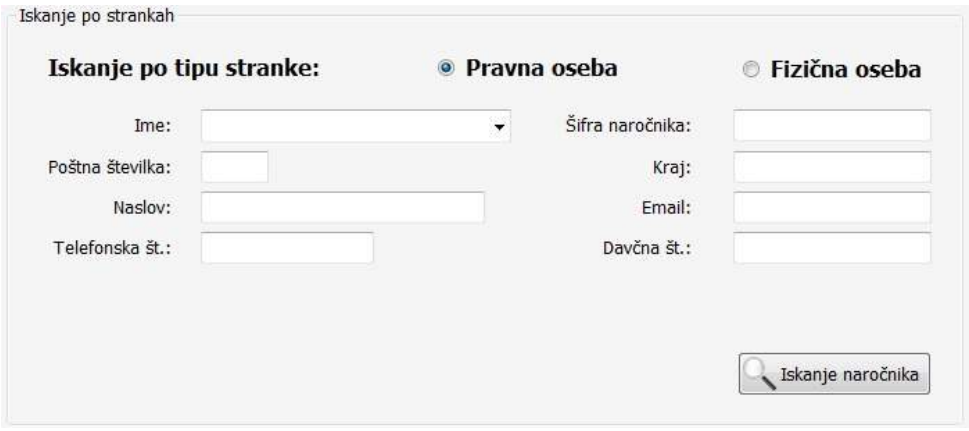
Slika 3.18: Iskanje po prilogah

Za izvedbo iskanja po prilogah mora uporabnik klikniti na gumb »Iskanje priloge«. Ta nato sproži klic funkcije, ki se poveže na podatkovno bazo ter shranjeni proceduri poda vse podatke za iskanje. Shranjena procedura nato vrne rezultate (slika 3.19), ki se nadalje vnesejo v tabelo. Ti rezultati vsebujeje enolični identifikator pošte, ki nakazuje h kateri pošti priloga pripada, ime stranke, ki je prilogo poslalo oziroma ji je bila priloga poslana, ime datoteke s končnico in opombo, ki jo je vneševalec pošte napisal pri prilogi. Tabela vsebuje tudi stolpec »Odpri datoteko«, ki vsebuje hiperpovezavo. S klikom nanjo se uporabniku odpre izbrana priloga.

Slika 3.19: Prikaz rezultatov iskanja prilog

Iskanje po strankah

Zadnja opcija iskanja je iskanje po strankah, ki je razvidno iz slike 3.20. Uporabnik velikokrat potrebuje določeno informacijo o stranki in mu v takih primerih pride prav, če lahko to informacijo enostavno poišče. Pri iskanju mora uporabnik najprej izbrati ali bo iskal pravno (privzeta) ali fizično osebo. V primeru iskanja pravne osebe lahko iskanje omeji z vnosom imena stranke, poštne številke, naslova, kraja, telefonske številke, šifre naročnika, elektronskega naslova in davčne številke. Pri iskanju fizične osebe so polja enaka, le da polja za vnos davčne številke ni. Vsa polja vsebujejo ustrezne validacije z namenom, da uporabnika aplikacija v primeru napačnega vnosa, na to opozori.



Slika 3.20: Iskanje po strankah

Tiskanje današnjih vnosov in rezultatov iskanja

Uporabnik ima vedno tudi možnost tiskanja rezultatov iskanja. To stori tako, da klikne na gumb »Natisni rezultate«, ki se nahaja pod tabelo z rezultati. Klik sproži klic funkcij generateSearchPrint() ter Print(). Prva na podlagi vsebine tabele generira html in nato rezultat vnese v skriti urejevalnik, medtem ko druga ponudi možnost samega printanja.

Podobno deluje tudi funkcionalnost tiskanja današnjih vnosov prejete in poslane pošte. Uporabnik klikne na gumb »Tiskanje današnjih vnosov«, kar sproži klic funkcije getTodaysMails(), ki preko shranjene procedure vrne objekt DataTable z vsemi informacijami o današnjih vnosih. Nato funkcija preveri število vnosov in če je število vnosov večje od nič nadaljuje z izvajanjem. Sledi generiranje html kode, kjer najprej generira tabelo prejete pošte ter zatem še tabelo poslane pošte. Na koncu se vse to shrani v skriti urejevalnik in uporabnik lahko izbere tiskalnik ter natisne generiran dokument. Primer generiranega dokumenta prikazuje slika 3.21.

Page 1 of 1

Statistika pošte za dan 25. 6. 2011

Prejeta pošta

Stranka: RSTEAM d.o.o. — ID pošte: 20913
Zadeva: vabilo šoli — Tip pošte: E-mail
Priloge:
- vabilo-šola.doc

Stranka: RSTEAM d.o.o. — ID pošte: 20914
Zadeva: vabilo šoli — Tip pošte: priporoceno
Priloge:
- vabilo-šola.doc

Stranka: rsteam2 — ID pošte: 20912
Zadeva: ff — Tip pošte: priporoceno

Poslana pošta

Stranka: RSTEAM d.o.o. — ID pošte: 3889
Zadeva: Podatki za vnos — Tip pošte: Pismo
Priloge:
- BONI.txt
- tpj.pdf
- svetovanje_rar.zip

Slika 3.21: Tiskanje današnje pošte

4 Sklepne ugotovitve

Diplomsko delo opisuje celoten razvoj aplikacije Poštna knjiga. Za izdelavo aplikacije smo se odločili po pogovoru z naročniki, ki so izrazili željo po novi aplikaciji, saj so bili mnenja, da je potrebna menjava obstoječe. Ta je bila nepregledna, njena uporaba pa je bila posledično bistveno otežena. Poleg tega so jo razvijalci zelo redko nadgrajevali in ni vsebovala vse funkcionalnosti, ki so si jo naročniki želeli. Naročnik nam je tako zaupal izdelavo nove aplikacije, ki bi nadomestila sedanjo.

Za uspešno izvedbo projekta je potrebna temeljita analiza uporabnikovih zahtev. Na podlagi omenjenih zahtev se nadalje izdela načrt željene aplikacije. Včasih je potrebna tudi naknadna sprememba načrta, upoštevajoč uporabnikove zahteve po dodatnih funkcionalnostih aplikacije. Pri razvijanju aplikacije sem naletel na težave, ki bi se jim, ob boljšem poznavanju delovanja in knjižnic programskega jezika C# in orodja Visual Studio, lahko izognil ter tako prihranil veliko časa. Trenutna verzija aplikacije vsebuje tri glavne funkcionalnosti in sicer prijavo v sistem, dodajanje prispele in poslane pošte ter napredno iskanje, ki uporabniku omogoča, da hitro najde željene podatke.

Aplikacijo bi lahko še izboljšali že, če bi priloge prenašali na strežnik, ki bi bil lociran pri naročniku. Tako bi bile priloge neodvisne od vsebine podatkov uporabnikovega računalnika. Isto je moč trditi tudi za podatkovno bazo. Za lažjo nadgradnjo aplikacije bi bilo potrebno dodati možnost posodobitve preko interneta. S sodelavci smo razmišljali tudi o dodatnem modulu, ki bi optično prebrano datoteko avtomatsko uvrstil med priloge.

Naročnik aplikacije trenutno uporablja testno različico nad katero je navdušen. Pozdravlja tudi zamisli o možni izboljšavi, rad pa doda tudi svoje ideje, ki jih poskušamo v celoti realizirati.

Priloge

Programska koda

funkcija btnPrintTodayMail_Click(object Sender, EventArgs e)

```
private void btnPrintTodayMail_Click(object sender, EventArgs e)
{
    DataTable mailIDs = new DataTable();
    mailIDs = getTodaysMailIDs();
    if(mailIDs.Rows.Count == 0){
        MessageBox.Show("Danes ni bilo pote!"); return;
    }
    string body = "",customer, subject, id, type, attachments;
    string date = DateTime.Now.Day.ToString() + ".
"+DateTime.Now.Month.ToString()+". "+DateTime.Now.Year.ToString();
    body += "<P align=center><STRONG>Statistika pote z dne
"+date+"</STRONG></P><BR></BR>";
    //generiranje tabele prejeta pota
    bool atLeastOne = false, oneAttachment = false;
    DataTable attachmentsData = new DataTable();
    foreach (DataRow row in mailIDs.Rows)
    {
        oneAttachment = false;
        if (row["IsInMail"].ToString() == "1")
        {
            if (!atLeastOne)
            {
                atLeastOne = true;
                body += "<P align=center><STRONG>Prejeta
pota</STRONG></P><br></br>";

            }
            attachments = "";
            customer = row["Name"].ToString();
            subject = row["Subject"].ToString();
            id = row["ID"].ToString();
            attachmentsData = getAttachments(Int32.Parse(id), true);
            foreach (DataRow attRow in attachmentsData.Rows)
            {
                if(!oneAttachment)
                {
                    attachments += "<div style=\"margin-
left:12.5%\">Priloge:<br />";
                    oneAttachment = true;
                }
                attachments += "    - " +
attRow["attName"].ToString() + "<br />";
            }
            if (oneAttachment)
                attachments += "</div>";
            else
                attachments += "<br></br>";
            type = row["typeDesc"].ToString();
```

```
            body += "<table border=0 celling=0 padding=0 width=\"75%\"
align=center><tr style=\"padding:5px\"><td width=\"65%\">Stranka: " +
customer + "</td><td width=\"35%\">ID pote: " + id +
"</td></tr><trstyle=\"padding:5px\"><td width=\"65%\">Zadeva: " + subject +
"</td><td width=\"35%\">Tip pote: " + type + "</td></tr></table>" +
attachments +"<hr align=center width=\"75%\"></hr><br />";
        }
    }
    //generiranje tabele poslana pota
    atLeastOne = false;
    foreach (DataRow row in mailIDs.Rows)
    {
        oneAttachment = false;
        if (row["IsInMail"].ToString() == "0")
        {
            if (!atLeastOne)
            {
                atLeastOne = true;
                body += "<P align=center><STRONG>Poslana
pota</STRONG></P><br></br>";
            }
            attachments = "";
            customer = row["Name"].ToString();
            subject = row["Subject"].ToString();
            id = row["ID"].ToString();
            attachmentsData = getAttachments(Int32.Parse(id), false);
            foreach (DataRow attRow in attachmentsData.Rows)
            {
                if (!oneAttachment)
                {
                    attachments += "<div style=\"margin-
left:12.5%\">Priloge:<br />";
                    oneAttachment = true;
                }
                attachments += "    - " +
attRow["attName"].ToString() + "<br />";
            }
            if (oneAttachment)
                attachments += "</div>";
            else
                attachments += "<br></br>";
            type = row["typeDesc"].ToString();
            body += "<table border=0 celling=0 padding=0 width=\"75%\"
align=center><tr style=\"padding:5px\"><td width=\"65%\">Stranka: " +
customer + "</td><td width=\"35%\">ID pote: " + id +
"</td></tr><trstyle=\"padding:5px\"><td width=\"65%\">Zadeva: " + subject +
"</td><td width=\"35%\">Tip pote: " + type + "</td></tr></table>" +
attachments + "<hr align=center width=\"75%\"></hr><br />";
        }
    }
    editorPrintTodays.BodyHtml = body;
    editorPrintTodays.Print();
}
```

Kazalo slik

Literatura in viri

[1] (2011) Microsoft Visual Studio. Dostopno na:
http://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[2] (2011) C#. Dostopno na:
http://en.wikipedia.org/wiki/C_Sharp_(programming_language)

[3] (2011) .NET Framework. Dostopno na:
http://en.wikipedia.org/wiki/.NET_Framework

[4] (2011) HTML. Dostopno na:
http://en.wikipedia.org/wiki/HTML

[5] (2011) Strežnik Microsoft SQL. Dostopno na:
http://en.wikipedia.org/wiki/Microsoft_SQL_Server

[6] (2011) Orodje PowerDesigner. Dostopno na:
http://www.sybase.com/products/modelingdevelopment/powerdesigner

[7] (2006) Način modeliranje diagrama primerov uporabe. Dostopno na:
http://mitjab.homelinux.net:8080/~mitjab/Sola/VAJE/OIS/Modeliranje%20primerov%20uporabe%20(PU)/modeliranje_USE_CASE.pdf