UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matjaž Grosek

Informacijska podpora za obvladovanje komunikacije v poslovnih okoljih

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE

RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Dejan Lavbič

Ljubljana, 2013

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00061/2012
Datum: 05.11.2012

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

Kandidat: **MATJAŽ GROSEK**

Naslov: **INFORMACIJSKA PODPORA ZA OBVLADOVANJE KOMUNIKACIJE V POSLOVNIH OKOLJIH**

INFORMATION SYSTEM SUPPORT FOR BUSINESS COMMUNICATION MANAGEMENT

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Pri uspešnem delovanju organizacije je ključnega pomena zanesljiva in učinkovita komunikacija. V manjših poslovnih okoljih, kjer sodeluje manjše število akterjev, se večinoma uporablja ustno sporazumevanje in komunikacija s pomočjo elektronske pošte. Pri večjem številu uporabnikov in projektnemu delu, kjer so sodelujoči tudi iz različnih geografsko ločenih enot, pa takšen način komunikacije ni primeren. V okviru diplomske naloge naj študent kritično pregleda področje informacijske podpore za obvladovanje komunikacije v poslovnih okoljih in naj implementira svojo rešitev. Le-ta naj temelji na delovnem toku obvladovanja zahtevkov, ki skozi proces spreminjanja statusa in dodeljevanja različnim vlogam potujejo skozi poslovni proces. Rešitev naj bo implementirana s pomočjo spletnih tehnologij.

Mentor:

doc. dr. Dejan Lavbič

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Matjaž Grosek, z vpisno številko **63080076**, sem avtor diplomskega dela z naslovom:

Informacijska podpora za obvladovanje komunikacije v poslovnih okoljih

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Dejana Lavbiča,
- 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 21. februarja 2013 Podpis avtorja:

Zahvala

Za strokovno pomoč in korekten odnos med nastajanjem diplomske naloge se zahvaljujem mentorju doc. dr. Dejanu Lavbiču.

Zahvala gre tudi staršema, ki sta mi omogočila študij.

Hvala!

Kazalo

Uporabljene kratice in pojmi

- PHP (PHP: Hypertext Preprocessor) – razširjen odprtokodni programski jezik, ki se uporablja za razvoj dinamičnih spletnih vsebin
- MySQL – sistem za upravljanje s podatkovnimi bazami
- jQuery – knjižnica za JavaScript
- CSS (Cascading Style Sheets) – kaskadne stilske podloge, ki so namenjene oblikovanju vsebine na spletni strani
- HTML (Hyper Text Markup Language) – označevalni jezik za izdelavo spletnih strani
- AJAX (Asynchronous JavaScript and XML) – skupina medsebojno povezanih spletnih razvojnih tehnik (JavaScript in XML)
- Yii (Yes it is) – napredno ogrodje za razvoj spletnih aplikacij z PHP
- XML (Extensible Markup Language) – označevalni jezik, ki se uporablja za strukturiranje, shranjevanje in prenos podatkov
- MVC (Model–view–controller) – arhitektura, ki ločuje interakcijo z informacijami od prikaza le-teh
- JavaScript – objektni skriptni programski jezik, ki se uporablja pri ustvarjanju interaktivnih spletnih aplikacij
- E-R diagram – diagramska tehnika za opis podatkovne baze
- CRM (Customer relationship management) – upravljanje odnosov s strankami
- API (application programming interface) – protokol, namenjen za vmesnike preko katerih komunicirajo programske komponente
- REST (Representational state transfer) – stil programske arhitekture
- JSON (JavaScript Object Notation) – standard za izmenjavo podatkov v obliki, berljivi za človeka

Povzetek

V diplomski nalogi smo raziskovali področje komunikacije v organizacijah. Najprej smo v prvem delu opisali primer komunikacije in težave, do katerih ponavadi prihaja v organizacijah, ki ne uporabljajo namenskih programov za komunikacijo. V nadaljevanju smo testirali in opisali nekaj že obstoječih rešitev na področju sodelovanja in komunikacije. Nato smo pripravili idejo aplikacije, ki smo jo v nadaljevanju realizirali. Osrednji del diplomske naloge je tako namenjen načrtovanju in implementaciji aplikacije. Opisali smo potek priprave podatkovne baze in realizacijo vodene komunikacije. V nadaljevanju smo predstavili implementacijo vodene komunikacije in obveščanja uporabnikov ob spremembah. V zaključnem delu smo izpostavili nekaj možnih izboljšav in razširitev aplikacije.

Ključne besede:

vodena komunikacija, sodelovanje v organizaciji, spletna aplikacija

Abstract

In the thesis we researched the field of communication in organizations. In the first part we described a case of communication and some problems that usually occur in the organizations, not using programs meant for communication. Later on we tested and described some of the already known solutions in the field of collaboration and communication. Then we presented the idea of the application that we realized in the following part of the thesis. The main part of the thesis is therefore meant for the construction and the implementation of the application. We described the course of preparation of the database and the realization of the lead communication. Further on we presented the implementation of the lead communication and the implementation of informing the users about the changes. In our concluding part we pointed out some of the possible improvements and expansions of the application.

Keywords:

led communication, collaboration in organizations, web application

Poglavje 1

Uvod

Eden izmed pogojev za uspešno delovanje organizacije je tudi zanesljiva in učinkovita komunikacija. Z ustreznim načinom komunikacije je potrebno zagotoviti, da imajo člani organizacije v vsakem trenutku na razpolago vse informacije, ki jih potrebujejo pri svojem delu, saj so lahko le tako kar se da učinkoviti. Informacije morajo biti podane nedvoumno, da ne prihaja do nesporazumov pri komunikaciji in nezadovoljstva med člani ter posledično slabšega delovanja organizacije.

Komunikacija lahko poteka na več načinov, ki so različno primerni za različne organizacije. V manjših organizacijah, kjer ni veliko članov in komunikacije med njimi, lahko slednja poteka ustno ali preko spletne pošte. Za organizacije, ki imajo nekaj deset članov, pa tak način sporazumevanja ni najbolj učinkovit, ker med drugim ne omogoča ustreznega pregleda in arhiviranja komunikacije. Poleg funkcionalnih pomanjkljivosti na področju sodelovanja, strukturiranja in formalizacije komunikacije ima spletna pošta tudi varnostne pomanjkljivosti [1], zato se pojavi potreba po namenskih programih za deljenje informacij, ki bi omogočali sodelovanje med člani organizacije ter vodenje in organizacijo dela.

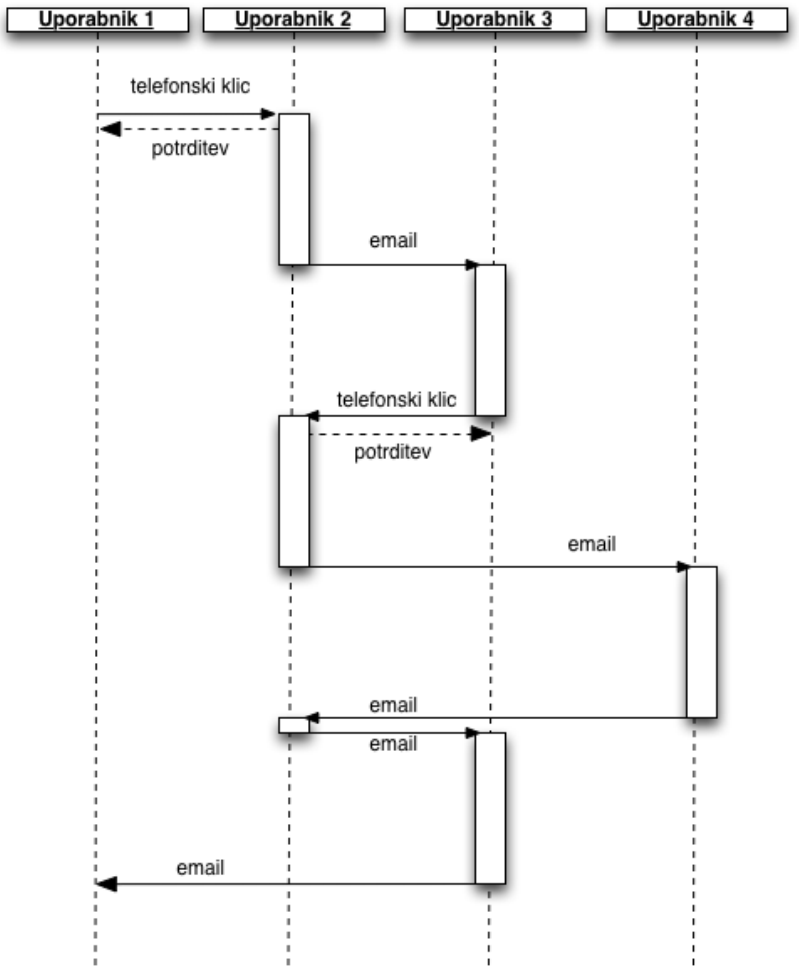
Glavna tema te diplomske naloge je komunikacija in sodelovanje med uporabniki v organizaciji. Razvili bomo prototip spletne aplikacije, ki bo omogočala učinkovitejšo komunikacijo med uporabniki. Z aplikacijo želimo doseči, da ne bo prihajalo do nesporazumov in neskladnosti pri komunikaciji. Poudarek aplikacije bo na enostavnosti za uporabo ter čim lažji uvedbi aplikacije v organizacijo.

Najprej bomo v 2. poglavju predstavili potek komunikacije brez podpore aplikacije, pomanjkljivosti ter možne izboljšave pri tej komunikaciji; v poglavju 3 bomo predstavili nekaj že obstoječih rešitev na področju komunikacije in sodelovanja med uporabniki v organizaciji. V poglavju 4 bomo predstavili funkcionalnosti aplikacije in primer uporabe le-te v organizaciji. V 5. poglavju bomo na kratko opisali načrtovanje in implementacijo aplikacije, v 6. poglavju bomo govorili o možnih izboljšavah in razširitvah aplikacije. V zadnjem poglavju bomo opisali naše ugotovitve.

Poglavje 2

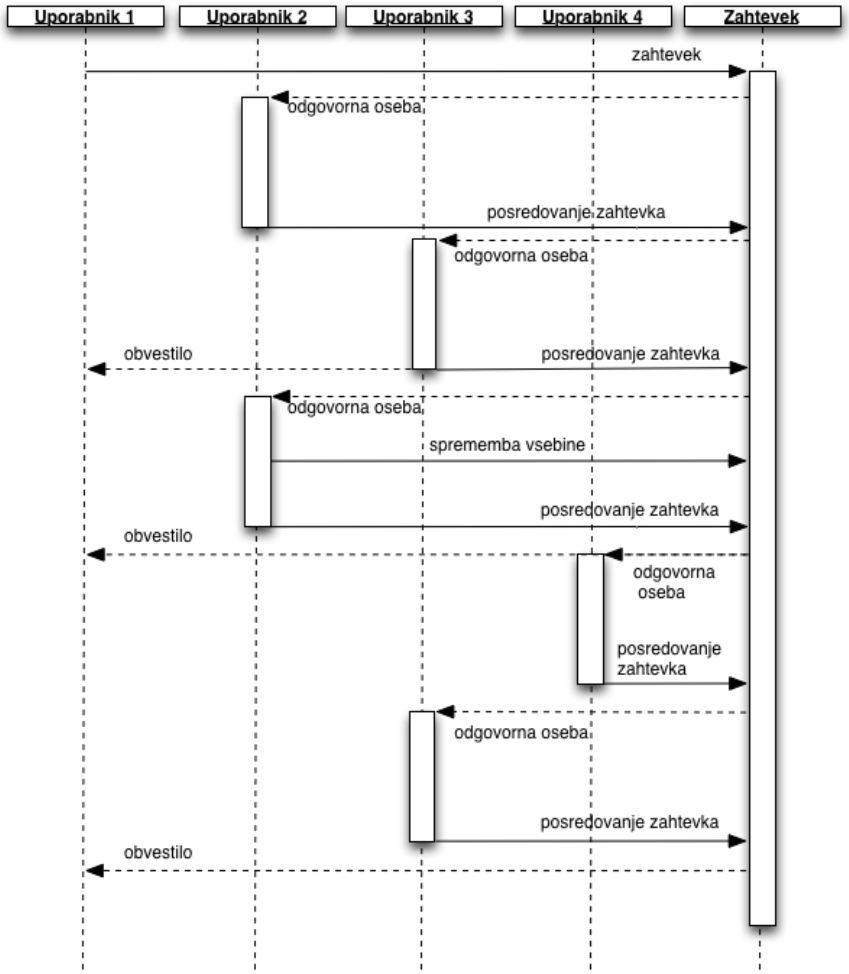
Komunikacija v organizaciji

Komunikacija, ki poteka ustno oz. pisno preko spletne pošte, je nezanesljiva iz več razlogov. Vsebina komunikacije, ki poteka ustno, se lahko v obdobju nekaj dni pozabi ali pa si vsak zapomni svojo različico pogovora, zato kot taka oz. v tej obliki ni primerna za prenos pomembnih informacij. Prednost ustne komunikacije pa je v enostavnosti za uporabo in hitri odzivnosti udeležencev pogovora. Ko naraste število oseb ali količina komunikacije, je potrebno le-to začeti beležiti, in sicer s pisanjem in prenašanjem papirnih dokumentov, s pošiljanjem elektronskih sporočil ali z uporabo namenske aplikacije. [2]



Slika 2.1: Primer komunikacije brez uporabe posebnega programa

Pri komunikaciji z elektronskimi sporočili lahko z ostalimi komuniciramo ne glede na njihovo dosegljivost, to pa vpliva na podaljšanje odzivnega časa. Prav tako od prejemnika ne dobimo takojšnjega odziva na vsebino. Ker je komunikacija zabeležena, je možno kasnejše pregledovanje in iskanje, ampak le po lastnih sporočilih, zato se lahko zgodi, da nimajo vsi konsistentnih informacij. Komunikacija med uporabniki poteka brez pravil. Lahko se zgodi, da uporabnik pošlje sporočilo napačni osebi ali pozabi nadrejenega obvestiti o zaključku zadeve. Ker se lahko sporočila pošiljajo med več uporabniki, se pojavijo težave glede spremljanja napredka in prenosa odgovornosti za določeno zadevo.



Slika 2.2: Komunikacija z namensko aplikacijo

Da bi izboljšali komunikacijo, je treba zagotoviti, da imajo vsi uporabniki, ki so vanjo udeleženi, pregled nad že opravljenim. Le tako se informacije ne podvajajo, hkrati pa so vse, ki so potrebne, na voljo uporabnikom. To lahko realiziramo tako, da se komunikacija izvaja in shranjuje na strežniku. Uporabniki dostopajo do strežnika preko brskalnika in upravljajo s komunikacijo, ki je shranjena v podatkovni bazi. Spremembe so zato vidne vsem ostalim udeležencem, o njih pa so uporabniki lahko obveščeni z elektronskimi ali SMS sporočili.

Ker se v veliko primerih komunikacija odvija na enak način, je smiselno, da se določijo pravila, ki bi usmerjala njen tok. Z njimi bi zagotovili, da se zaradi slabe komunikacije ne izpusti kakšne stopnje oz. da komunikacija ne zaide v napačno smer. Uporabnik ob

spremembi stopnje vnese komentar, izbere naslednjo stopnjo ter odgovorno osebo, ki bo poskrbela za njeno izvedbo. S tem, ko je komunikacija na določeni stopnji, lahko vemo, približno koliko je še do zaključka. Da pa bi bila informacija o predvidenem zaključku natančnejša, in ker mora biti določena komunikacija zaključena do nekega roka, bi uporabniki ob spremembi stopnje izbrali rok zaključka. Kadar bi potrebovali hitrejši odziv od uporabnika kot ponavadi, bi ga o spremembi v komunikaciji obvestili z SMS sporočilom. Prav tako so lahko uporabniki obveščeni, ko komunikacija doseže določeno stopnjo ali jo zapusti (slika 2.2). Uporabnik lahko dobi povratne informacije tudi, kadar naslednja oseba potrdi sprejem. [3]

Poglavje 3

Obstoječe rešitve

Na trgu je mnogo različnih rešitev na področju komunikacije in sodelovanja med uporabniki. Večina teh rešitev je plačljivih, vseeno pa je na voljo nekaj odprtokodnih programov, ki so brezplačni za uporabo. Programi se razlikujejo predvsem v naboru funkcionalnosti, ki jih nudijo, po načinu realizacije in uporabe le-teh. Tako so nekateri programi zaradi množice funkcionalnosti in z več nastavitvami primerni za širšo množico organizacij z različnih področij, drugi programi pa so predvideni za namensko uporabo in so zato primerni samo za določene tipe organizacij.

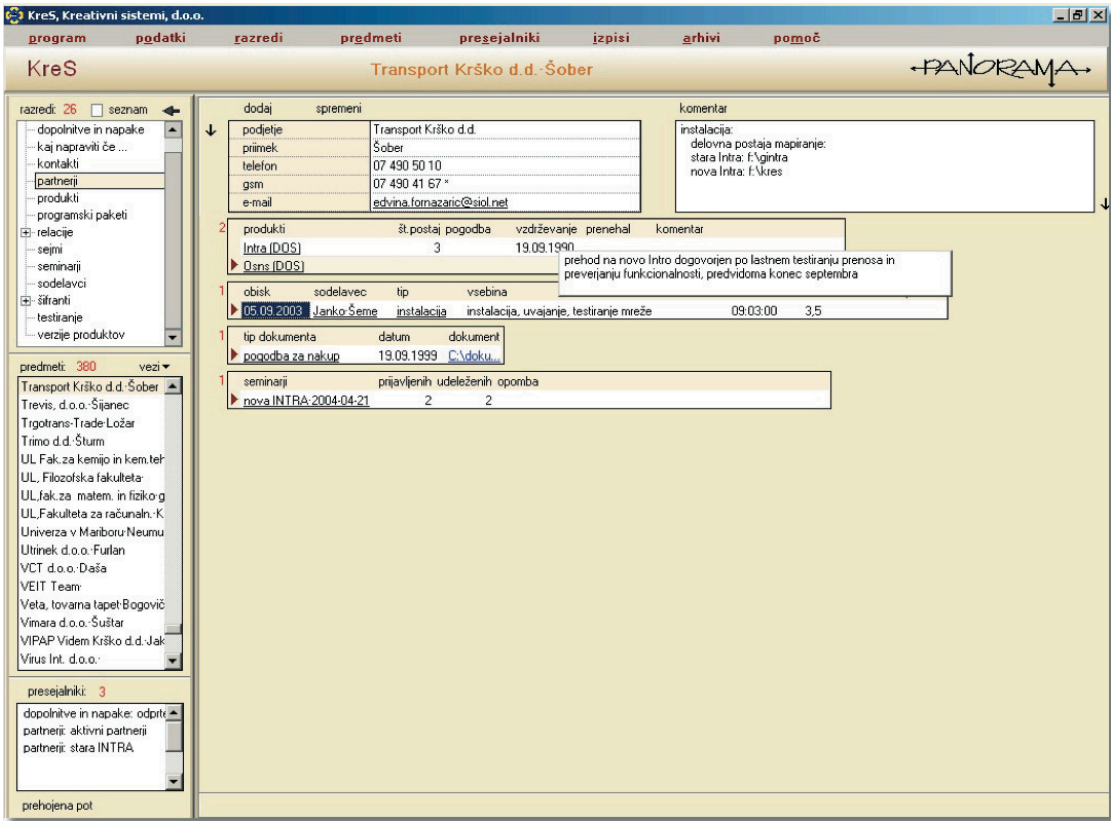
Pregledali in analizirali smo več slovenskih in tujih programov, natančneje pa smo preučili pet programov. Od slovenskih rešitev smo opisali programa Panorama [4] [5] [6] in Intrix [7]. Programi JIRA [8], Lotus Notes [9] in openCRX [10], ki je primer odprtokodne rešitve, pa so iz tujine. Zanimale so nas predvsem možnosti za uporabo v organizacijah, ki jih nudijo programi in načini realizacije določenih funkcionalnosti.

3.1 PANORAMA

3.1.1 Predstavitev

Panorama je podatkovno orodje za nadzorovanje in vodenje razpršenih podatkov. Omogoča sočasen prikaz podatkov določenega predmeta in podatkov vseh predmetov, ki so v povezavi z izbranim. Za delo s programom ni potrebnih posebnih računalniških znanj, saj je uporaba intuitivna.

Panoramo lahko uporabimo kot managerski informacijski sistem, sistem za odnose s strankami, dokumentarno-arhivski informacijski sistem ali informacijski sistem za društva itd. Področju uporabe primerno se izdela podatkovni model ali pa se uvozi že obstoječe podatkovne baze. Podatke je možno izvoziti v različnih formatih, tako da so primerni za nadaljnje analize. Prav tako lahko uporabnik sam oblikuje poročila, ki jih želi generirati.



Slika 3.1: Videz zaslona pri Panorami

3.1.2 Prednosti in slabosti

Ker program Panorama temelji na bazah podatkov, ga je preprosto uporabiti na veliko področjih. Uporabniku omogoča spreminjanje strukture podatkov, zato so kasnejše nadgradnje in spremembe enostavne.

Program teče samo v okolju Windows in potrebuje namestitev, zato ni možno dostopati do podatkov s kateregakoli računalnika. Kot prikazuje Slika 3.1, je uporabniški vmesnik precej zastarel.

3.1.3 Primerjava

Panorama je specializirana za beleženje in analizo podatkov, zato ni najbolj primerna za sodelovanje in komunikacijo med zaposlenimi. Program je mogoče uporabiti v mnogo scenarijih, ampak zaradi te splošnosti v večini primerov uporabniška izkušnja ni najboljša.

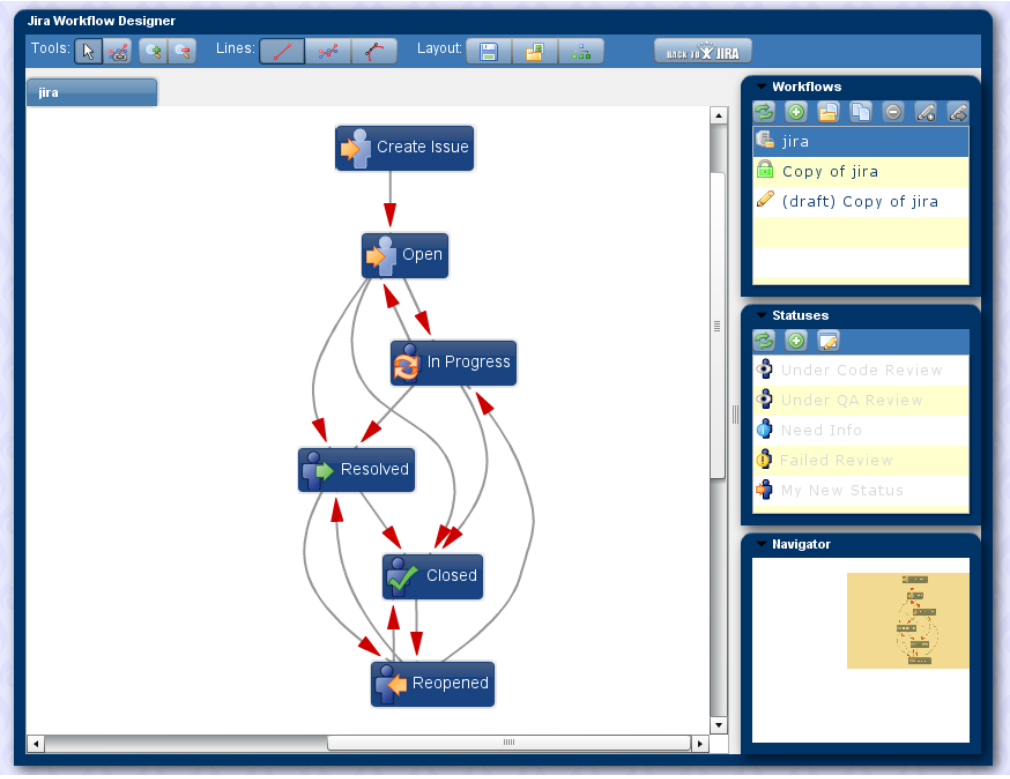
3.2 JIRA

3.2.1 Predstavitev

Programska oprema JIRA je specializirana za vodenje projektov razvoja programske opreme, lahko pa se uporablja tudi za vodenje drugih vrst projektov, kot sistem za podporo uporabnikom ali za zahtevke.

Vsak projekt je sestavljen iz ”problemov”. Problemi so lahko združeni v ”komponente”. Odvisno od uporabe je lahko ”problem” programski hrošč, opravilo na projektu ali zahtevek za podporo uporabniku.

Problemi skozi svoj cikel prehajajo med določenimi statusi, ki se jih lahko nastavi glede na potrebe uporabe programa. Slika 3.2 prikazuje orodje za urejanje statusov in povezav med njimi. V osnovi je vgrajen splošen tok dela, ki je sestavljen iz statusov: odprto, v obdelavi, rešeno, ponovno odprto in zaključeno – z ustreznimi prehodi med njimi.

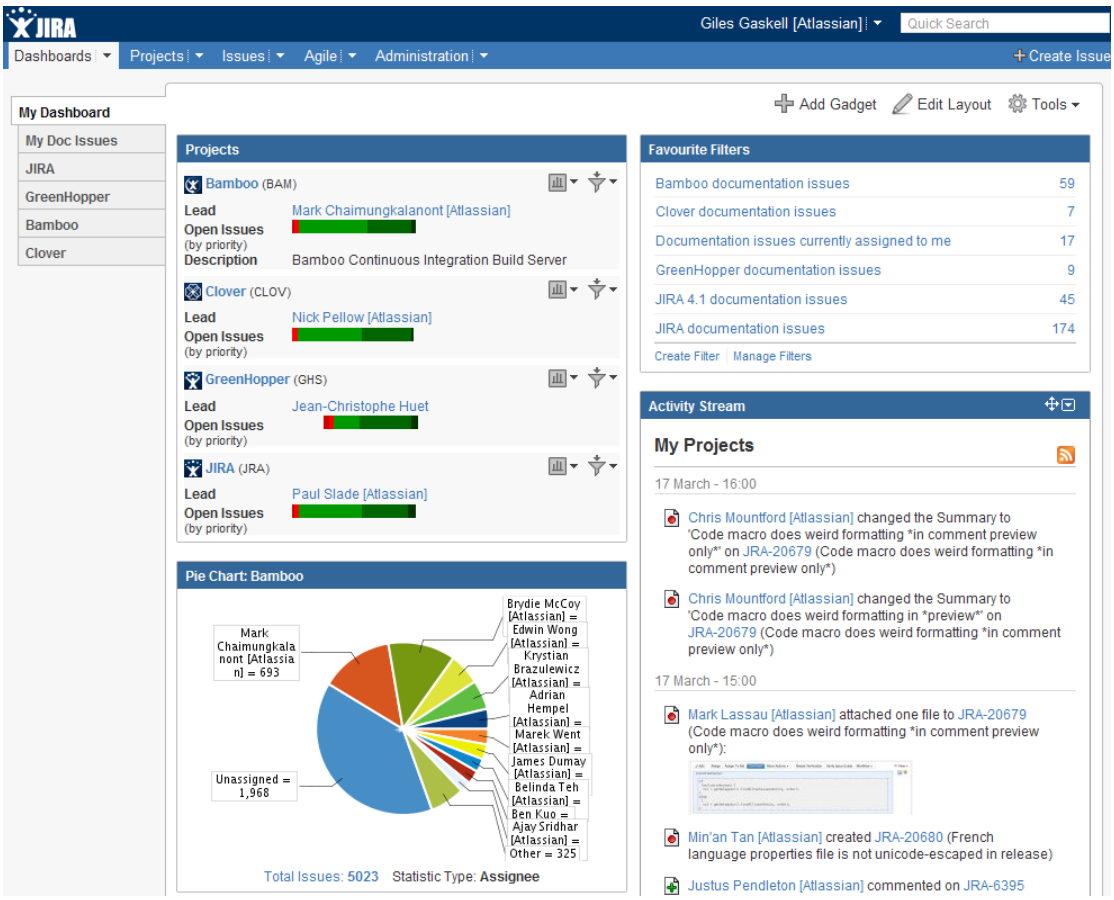


Slika 3.2: Urejanje delovnega toka

3.2.2 Prednosti in slabosti

Program lahko uporabljamo preko spleta ali ga namestimo na svoj strežnik – tako imamo sami nadzor nad svojimi podatki. Uporabnik ima na voljo več kot 30 pripomočkov, s katerimi si lahko opremi domačo stran (Slika 3.3) in jo prilagodi svojim potrebam. Zelo dobro je razvit sistem za generiranje raznih poročil, ki ni namenjen samo vodstvenemu kadru. Tako lahko uporabnik na vizualen način spremlja svoje delo ali projekt, na katerem sodeluje.

Zaradi množice funkcionalnosti, ki so vgrajene, je lahko uporaba za manj veščega uporabnika na začetku precej težavna.



Slika 3.3: Domača stran z raznimi dodatki

3.2.3 Primerjava

JIRA je v prvi vrsti namenjen vodenju obsežnejših projektov v organizaciji, zato nima takega poudarka na delu s strankami/poslovnimi partnerji (CRM). Program za razliko od Kolegija omogoča tudi planiranje projektov. Osnovno delovanje Kolegija na podlagi zahtevkov je zelo podobno kot pri programu JIRA, ki pa ima realiziranih mnogo več funkcionalnosti.

3.3 Lotus Notes

3.3.1 Predstavitev

Lotus Notes je program, namenjen komunikaciji, sodelovanju in izmenjavanju vsebine med uporabniki znotraj lokacijsko razpršenih skupin.

Sestavljen je iz elektronske pošte, koledarjev, stikov, dejavnosti in drugih modulov, namenjenih sodelovanju med uporabniki. Zelo dobro je razvito filtriranje, razvrščanje in označevanje vsebine. Z dodatkom za kreiranje delovnega toka je možno Lotus Notes uporabljati tudi kot sistem za upravljanje dokumentov oz. kot sistem za upravljanje zahtevkov.

3.3.2 Prednosti in slabosti

Poudarek pri Lotus Notesu je na sodelovanju med uporabniki (elektronska pošta, skupinski pogovori, video konference). Dostop do programa je možen z namizno aplikacijo, preko spletnega brskalnika ali z mobilno napravo. Razvitih je tudi veliko modulov, s katerimi si lahko prilagodimo program po svojih željah in potrebah. Velika prednost je tudi možnost uporabe programa takrat, kadar ni povezave z omrežjem.

Program je s svojimi mnogimi funkcionalnostmi bolj primeren za večje organizacije in zahtevnejše uporabnike.

3.3.3 Primerjava

Lotus Notes je obsežnejši in kompleksnejši od Kolegija. Na prvem mestu je namenjen splošnemu sodelovanju in organizaciji med uporabniki, kar pri Kolegiju ni tako poudarjeno (pogovori, zasebna sporočila). Z raznimi moduli in drugimi naprednimi nastavitvami Lotus Notes omogoča uporabo na mnogih področjih, Kolegij pa je bolj specializiran za prenos odgovornosti pri poslovnih procesih in za vodenje odnosov s strankami.

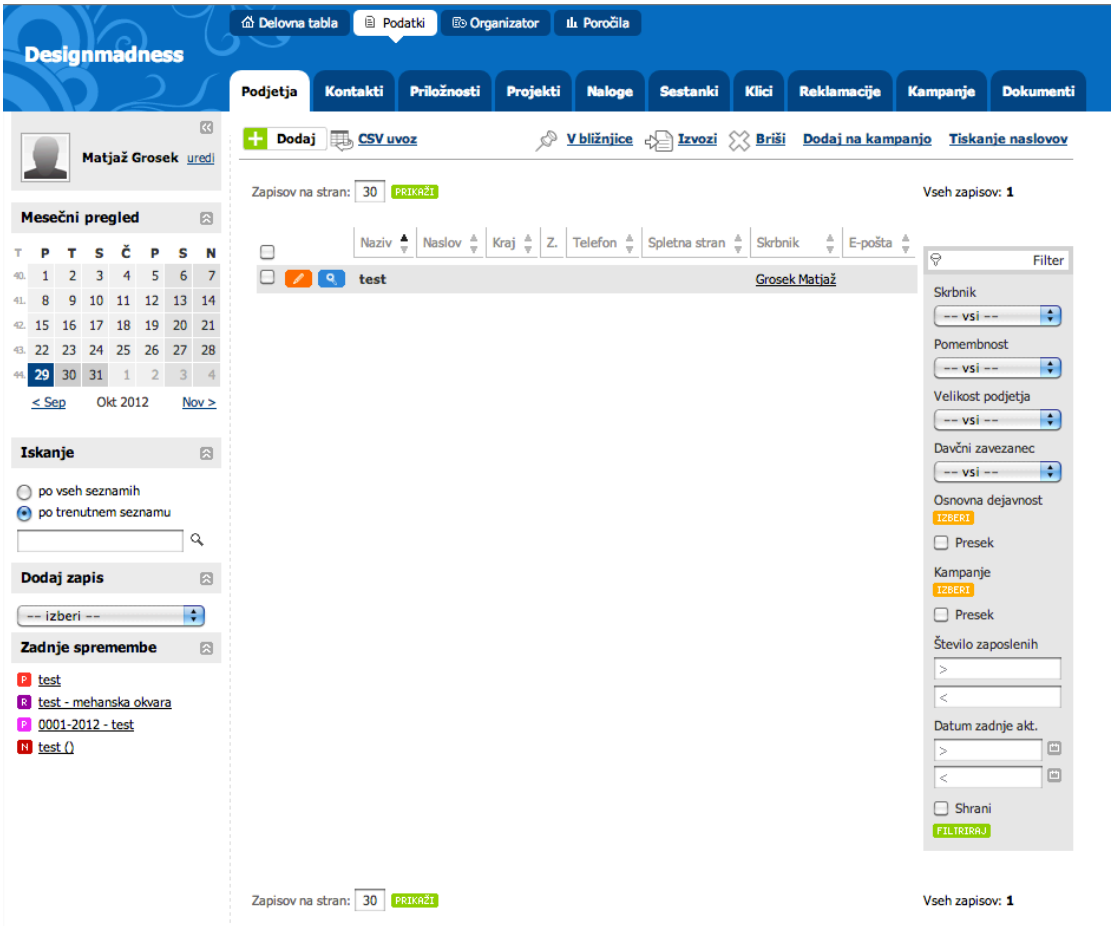
3.4 Intrix

3.4.1 Predstavitev

Intrix je program za spremljanje strank in vodenje projektov. Namenjen je učinkoviti in zanesljivi komunikaciji med uporabniki. Ker omogoča pregled nad prodajnimi aktivnostmi v organizaciji in generiranje raznih poročil, je uporaben tudi za vodstveni kader. Je izjemno prilagodljiv in enostaven za uporabo.

3.4.2 Prednosti in slabosti

Zelo pomembna prednost programa je povezljivost z bolj razširjenimi računovodskimi in ERP programi. Z urniki omogoča pregled nad razpoložljivostjo sodelavcev in optimizacijo porabe časa. Omejitev je, da je možno nastavljati samo statuse priložnosti, projektov in reklamacij, ne pa tudi povezav med temi statusi oz. delovnega toka; zato ni primeren za usmerjanje kompleksnejših poslovnih procesov. Prav tako ni najbolje rešen pregled zgodovine dogodkov na določeni postavki.



Slika 3.4: Pregled podjetij v programu Intrix

3.4.3 Primerjava

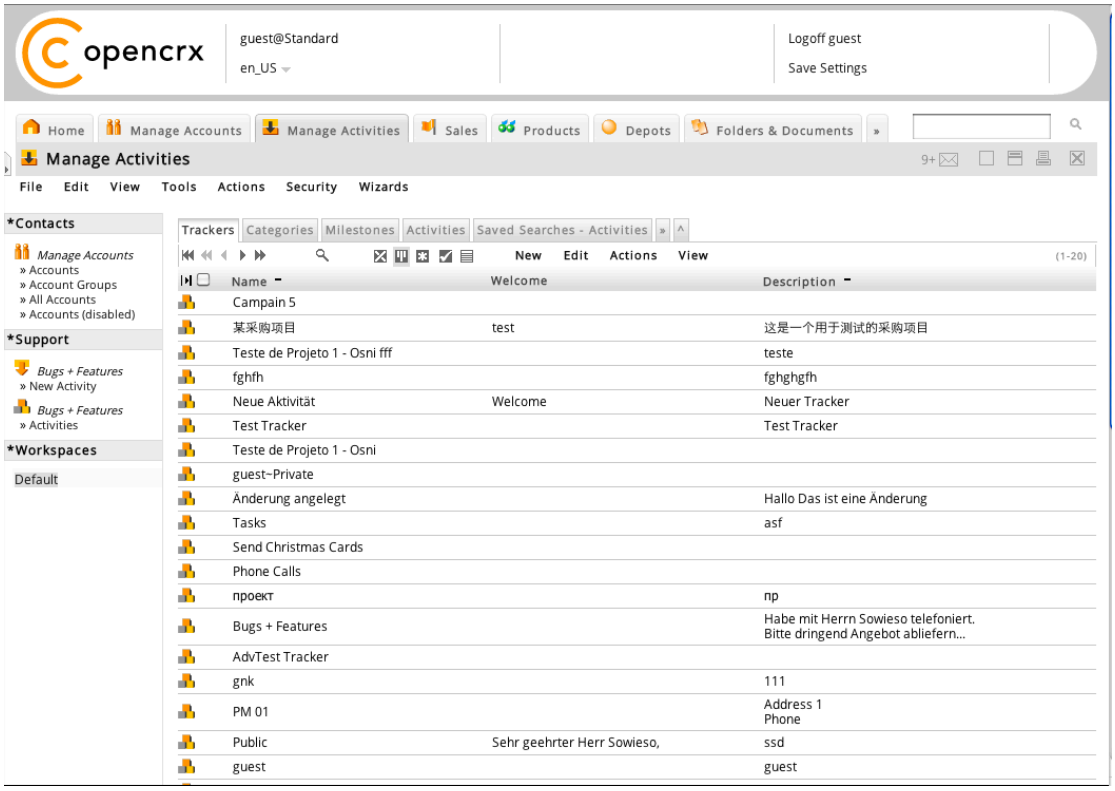
Intrix pomaga pri odnosu s strankami in pri prodajnih aktivnostih. Z ustreznimi nastavitvami lahko Kolegij uporabljamo na istem področju. Ker je Intrix specializiran za odnose s strankami, ima razvitih več funkcionalnosti na tem področju. Tako podpira povezavo aplikacije z IP telefonijo in beleženje pogovorov, tiskanje kuvert, beleženje povpraševanj, prijav itd. z obrazcev na spletnih straneh.

3.5 OpenCRX

3.5.1 Predstavitev

Program je odprtokodna spletna CRM rešitev, ki nudi orodja za koordinacijo prodaje in marketinga ter orodja za spremljanje komunikacije in aktivnosti s strankami, dobavitelji in partnerji.

Razvit je v Javi, zato lahko teče na vseh platformah in uporablja katero koli izmed bolj razširjenih podatkovnih baz. Za dostop se uporablja spletni brskalnik, tako da ni potrebna posebna namestitev programa na računalnike uporabnikov.



Slika 3.5: Pregled aktivnosti v openCRX

3.5.2 Prednosti in slabosti

Ker je openCRX odprtokodni projekt, je brezplačen za uporabo. Prav tako sta na razpolago vsa programska koda in dokumentacija, zato ga lahko prilagodimo svojim potrebam. Program je tudi možno povezati z drugimi že obstoječimi programi, saj ima pripravljene API in REST storitve. Uporabnikom omogoča upravljati z elektronsko pošto, s kontakti in koledarji, ki so lahko zasebni ali skupni.

Kot pri večini odprtokodnih projektov tudi pri openCRX-u deluje skupnost, ki nudi brezplačno podporo in pomoč uporabnikom. Poleg tega je na voljo tudi plačljiva podpora, ki jo nudi več partnerjev projekta. Za področje Slovenije je to podjetje HERMES SoftLab, ki nudi implementacijo, prilagoditve in prehod na openCRX.

3.5.3 Primerjava

Osnovna funkcionalnost openCRX-a je upravljanje odnosov s strankami in vodenje prodaje. Zaradi možnosti prilagajanja programa je openCRX zelo dinamičen in tako primeren za večino organizacij, ki se ukvarjajo s prodajo. Z vodenjem aktivnosti lahko tudi vodimo procese za posamezne projekte v organizaciji.

Kolegij ne omogoča tako celovitega vodenja prodaje kot openCRX, saj ni možnosti vodenja produktov in zaloge. Ima pa openCRX manj razvit del, ki se tiče dokumentov in dokumentacijskega sistema, ki pa se ga seveda lahko nadgradi z dodatnim modulom.

Poglavje 4

Informacijska podpora vodenju organizacije

4.1 Namen

V vseh organizacijah se pojavljajo določeni procesi, ki se mogoče ne izvajajo najbolj optimalno, sploh če pri teh procesih sodeluje več ljudi. Večino procesov se da razčleniti na več stopenj, preko katerih gredo proti zaključku. Za vsak tak ponavljajoč se proces se v Kolegiju kreira zahtevek, ki potem prehaja med vnaprej določenimi statusi, ki se ujemajo s stopnjami v procesu.

Pri procesu lahko nastajajo dokumenti (predračuni, pogodbe, pošta, e-pošta, ponudbe), ki se shranijo na zahtevek. Prav tako lahko pri vsakem zahtevku sodeluje več ljudi, ki si predajajo odgovornost za zahtevek in poskrbijo, da se uspešno zaključi.

Ker je za vsak zahtevek v nekem trenutku odgovorna določena oseba, se točno ve, kdo je kriv, če se proces ustavi. Dobra lastnost tega je, da lahko vodilni spremljajo in se ukvarjajo samo s tistimi zahtevki, ki so problematični.

Pri predajanju odgovornosti je pomembno, da ima tisti, ki prejme zahtevek v obdelavo, pregled nad preteklimi aktivnostmi na zahtevku in lahko tako brez težav nadaljuje proces. Za to je poskrbljeno tako, da se pri vsaki spremembi statusa zahtevka napiše komentar.

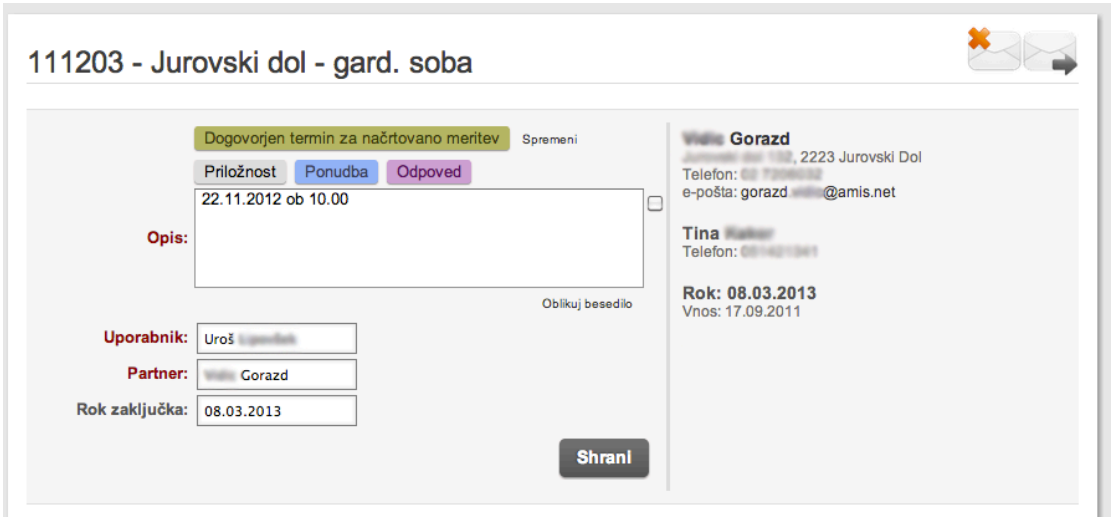
Z ustreznimi nastavitvami statusov in s pravili prehajanj med njimi je možno Kolegij uporabljati kot CRM, dokumentacijski sistem, za vodenje poslovnih procesov, “ticketing” program ali kombinacijo prej naštetega.

Glavna prednost Kolegija pred drugimi podobnimi programi je v tem, da se statusi zahtevkov nastavijo za specifično organizacijo in zato zaradi Kolegija ni potrebno spreminjati poslovnih procesov. S statusi zahtevkov lahko dosežemo tudi, da se določeni procesi izvajajo drugače. Zato je potrebno pred uvedbo Kolegija vsako organizacijo temeljito analizirati in skupaj z zaposlenimi zagotoviti idealne nastavitve.

4.2 Opis funkcionalnosti

4.2.1 Obvladovanje poslovnih procesov

S statusi in povezavami med njimi določimo stopnje, preko katerih mora iti zahtevek proti zaključku. Za vsak status se nastavi, kateri so možni naslednji statusi, v katere lahko napreduje zahtevek. Tako je zagotovljeno, da se bo proces razvijal v pravi smeri. Hkrati pa imamo vedno na voljo informacije, v kakšnem stanju je nek proces in kdo je odgovoren zanj. Z obveščanjem odgovornih oseb glede njihovih zadolžitev zagotovimo, da proces ne obstoji, prav tako pa so lahko o spremembah na zahtevku avtomatsko obveščeni tudi določeni uporabniki – npr. vodje oddelkov.



Slika 4.1: Spreminjanje statusa zahtevka

Pri veliki količini podatkov je potrebno imeti tudi učinkovito filtriranje in iskanje po tej vsebini. Iskanje po enem parametru je omogočeno v polju za hitro iskanje. Če želimo poiskati zahtevke, ki so v določenem statusu, imajo rok v določenem obdobju, jih obdeluje določen uporabnik, pa si pomagamo z naprednim iskanjem (Slika 4.2).

Slika 4.2: Iskanje po zahtevkih

Med spreminjanjem parametrov se v zgornjem desnem kotu osvežuje število zahtevkov, ki se ujemajo z izbranimi parametri. V naslednjem koraku imamo možnost, da shranimo parametre iskanja in kasneje dostopamo do rezultatov. Tako si lahko vsak uporabnik pripravi filtre, ki jih potrebuje pri uporabi aplikacije.

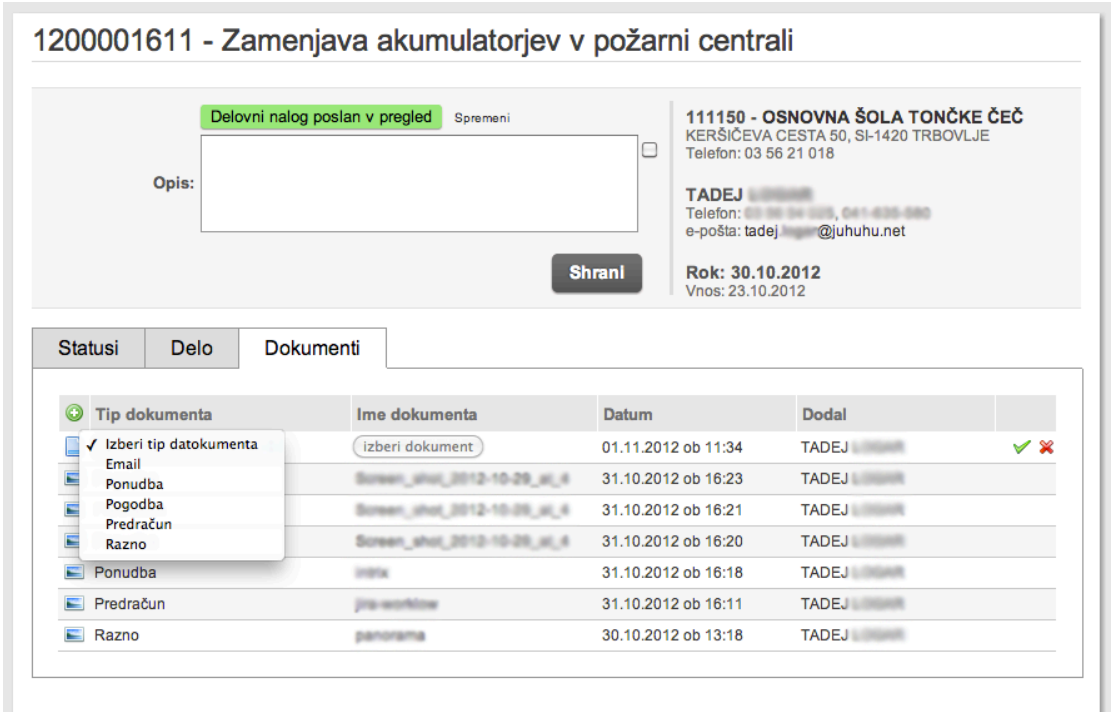
4.2.2 Dokumentni sistem

Dokumentni sistem se lahko uporablja na dveh področjih. Prvo področje je spremljanje dokumentov, ki nastanejo v poslovnem procesu – ponudbe, elektronska pošta, meritve, fotografije itd. Ti dokumenti so vedno vezani na zahtevek. Slika 4.3 prikazuje dodajanje dokumenta na zahtevek.

Drugo področje pa je namenjeno arhiviranju, posredovanju in pregledu prispele pisemske pošte. Tako lahko uporabniki pregledujejo pošto, ki je naslovljena na njih v elektronski obliki, hkrati pa je vsa pošta varno arhivirana. V primeru, da se organizacija želi znebiti fizičnega arhiva računov in ostalih dokumentov, ki jih je obvezno treba arhivirati določeno obdobje, pa je možna povezava programa z enim izmed podjetij, ki se ukvarja z arhiviranjem dokumentov.

Kadar je količina prejete pošte večja, lahko uporabnik vstavi vso pošto v skener in z uporabo ustreznih programov zgenerira dokumente v PDF formatu. Ti dokumenti se shranijo v mapo, do katere ima dostop tudi strežnik, na katerem teče aplikacija. Nato lahko uporabnik hitreje

opremi vsak dokument s podatki o prejemniku, pošiljatelju in vrsti dokumenta. Da uporabnik vnese na dokument točne podatke, se mu ob izbiri dokumenta s seznama prikaže predogled dokumenta.



Slika 4.3: Dodajanje dokumenta na zahtevek

4.2.3 Poslovni partnerji

Kadar so v poslovne procese organizacije vključene stranke ali poslovni partnerji, je potrebno imeti shranjene njihove kontaktne podatke in ostale informacije, ki se tičejo odnosa s partnerjem. Prav tako potrebujemo ustrezne podatke, da lahko izstavimo fakturo.

Predvsem v primeru, da je aplikacija povezana z drugimi računovodskimi ali ERP programi, morajo biti podatki o partnerjih konsistentni. Da se preprečijo podvajanja, se pred vnosom izvede iskanje po lokalni bazi partnerjev za možna ujemanja. Pri dodajanju novega partnerja se izvede iskanje po bazi AJPES in se tako zagotovi točne in popolne podatke. Za primere, ko partner pokliče v organizacijo in je potrebno hitro ugotoviti stanje, se uporabi funkcija hitro iskanje. Tako pridemo do vseh zahtevkov, dokumentov in ostalih podatkov, ki se tičejo partnerja.

4.3 Primer uporabe

Na sestanku z naročnikom vodja projekta pridobi želje in zahteve glede projekta; nato vnese pridobljene podatke na nov zahtevek, za katerega izbere status *analiza*. Ob vnosu zahtevka po potrebi vnese novega poslovnega partnerja. Za odgovorno osebo zahtevka izbere analitika, ki bo pripravil podrobno analizo trenutnega stanja pri naročniku.

Analitik nato pripravi poročilo, v katerem uskladi želje naročnika z njihovim trenutnim stanjem. Ko je poročilo pripravljeno, ga doda na zahtevek in spremeni status v *priprava ponudbe*, ob tem pa izbere naslednjo odgovorno osebo – finančnika.

Glede na pripravljeno poročilo finančnik pripravi ponudbo, ki jo bodo poslali naročniku. Preden ponudbo pošljejo, finančnik spremeni status zahtevka v *potrditev ponudbe* in ga posreduje direktorju. Če direktor oceni, da ponudba ni primerna, spremeni status zahtevka nazaj v *priprava ponudbe*, kar pomeni, da zahtevek avtomatsko dobi v obdelavo prejšnja odgovorna oseba. Ob spremembi statusa lahko direktor napiše komentar, da finančnik ve, kaj mora urediti. Zahtevek s popravljeno ponudbo nato finančnik ponovno posreduje direktorju. Ko ta ponudbo potrdi in spremeni status zahtevka v *ponudba odobrena*, izbere tajnico za naslednjo odgovorno osebo.

Tajnica nato pošlje ponudbo na elektronski naslov naročnika in ko le-ta odgovori na ponudbo, tajnica doda prejeto elektronsko sporočilo na zahtevek. Če je odgovor na ponudbo negativen, tajnica zaključi zahtevek tako, da izbere status *ponudba zavrnjena*. V primeru, da je ponudba sprejeta, pa tajnica spremeni status v *začetek projekta* in zahtevek posreduje oblikovalcu, ki bo pripravil uporabniški vmesnik.

Ko dobi oblikovalec zahtevek v obdelavo, ima pregled nad vso dokumentacijo in lahko takoj prične z izdelavo. Med procesom izdelave uporabniškega vmesnika lahko oblikovalec na zahtevek k dokumentom doda primere in nato zahtevek posreduje vodji projekta, da pregleda in oceni delo. Vodja vnese komentar in spremeni status zahtevka v *priprava uporabniškega vmesnika* ter tako posreduje zahtevek nazaj oblikovalcu. Do zaključka izdelave uporabniškega vmesnika lahko vodja projekta še nekajkrat dobi zahtevek v pregled. Ko vodja projekta dokončno potrdi uporabniški vmesnik, spremeni status zahtevka v *potrditev naročnika* in zahtevek dobi v obdelavo tajnica, ki nato naročniku po spletni pošti pošlje pripravljene dokumente na zahtevku.

Če naročnik potrdi pripravljen uporabniški vmesnik, tajnica spremeni status zahtevka v *programiranje* in ga posreduje glavnemu programerju. V primeru, da naročnik ni popolnoma zadovoljen z izdelanim uporabniškim vmesnikom, tajnica doda prejeto elektronsko sporočilo

na zahtevek in ga posreduje nazaj oblikovalcu. Ko so izdelani popravki, tajnica ponovno pošlje naročniku slike uporabniškega vmesnika, da ga potrdi.

Ko zahtevek prevzame glavni programer in pregleda vse pripravljene dokumente, razdeli delo tako, da odpre podzahtevke. Odpre podzahtevek za podatkovno bazo in ga posreduje ustreznemu programerju. Prav tako odpre podzahtevka za pripravo programske logike in implementacijo uporabniškega vmesnika. V sklopu istega zahtevka se zato odvijajo trije podzahtevki vzporedno. Ko programerji zaključijo podzahtevke, dobi glavni programer obvestilo. Nato spremeni status zahtevka v *testiranje* in ga posreduje programerju, ki bo opravil teste.

Po zaključenem testiranju programer v primeru odkritih napak zahtevek posreduje nazaj glavnemu programerju s potrebnimi opisi napak, sicer spremeni status zahtevka v *produkcija* in ga posreduje sistemskemu administratorju, ki bo poskrbel za namestitev aplikacije na produkcijski strežnik. Po opravljeni namestitvi dobi zahtevek v obdelavo tajnica, ki obvesti naročnika o zaključku razvoja aplikacije, hkrati pa spremeni status zahtevka v *zaključeno*.

Ko naročnik preveri in potrdi aplikacijo, tajnica spremeni status v *neplačano* in nato ob prejemu plačila zahtevek dokončno zaključi s spremembo statusa v *plačano*.

Poglavje 5

Razvoj aplikacije za podporo vodenju organizacije

Glede na zaznane potrebe in že obstoječe rešitve smo se odločili, da bomo razvili aplikacijo s spletnimi tehnologijami. Tako bo aplikacija dostopna preko spletnega brskalnika iz katere koli naprave, povezane v internet. Ena izmed prednosti spletnih aplikacij je tudi, da na strani odjemalca ni potrebno nameščati posebne programske opreme in zato ni potrebnega posebnega vzdrževanja aplikacije. [11]

Odločili smo se, da se bo entiteta, ki se bo ustvarila za določen proces, začetek komunikacije ali ob določenem dogodku – ko prispe pismo, dobimo zahtevo po reklamaciji, zahtevo za izdelavo ponudbe – imenovala zahtevek. Korak oz. stopnja, v kateri se bo nahajal zahtevek, pa se bo imenovala status.

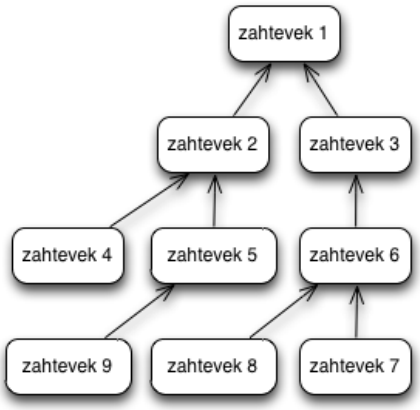
Aplikacijo bomo razvili z ogrodjem Yii, ki je narejeno z odprtokodnim programskim jezikom PHP. Za Yii smo se odločili, ker omogoča MVC arhitekturo. Prav tako Yii omogoča generiranje programske kode iz podatkovnega modela [12].

MVC (model-pogled-krmilnik) je programska arhitektura, ki ločuje prikaz podatkov od programske logike. Model sestavljajo podatki aplikacije in pravila glede podatkov. Ti podatki se ponavadi nahajajo v podatkovni bazi. V pogledu se generira HTML koda za prikaz iz podatkov, dobljenih iz krmilnika. V krmilniku se nahaja programska logika za pridobivanje ustreznih podatkov. Tako sta preko krmilnika povezana model in pogled.

5.1 Načrtovanje

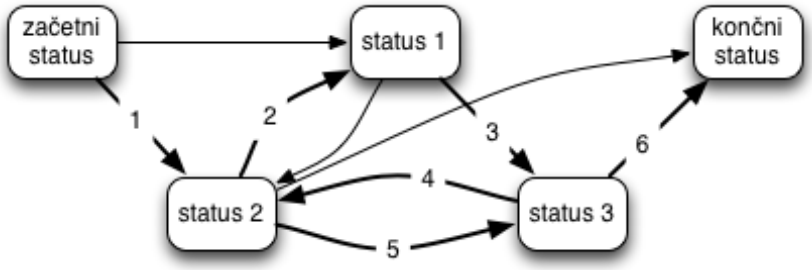
5.1.1 Funkcionalnosti

Za začetek vsake komunikacije bo uporabnik odprl zahtevek. Pri odpiranju zahtevka bo potrebno izbrati uporabnika, kateremu je namenjen zahtevek, poslovnega partnerja, na katerega se nanaša komunikacija in rok, v katerem se pričakuje, da bo uporabnik izvedel potrebne aktivnosti. Poleg naštetih podatkov bo uporabnik moral izbrati status, s katerim naj se začne zahtevek. Statusi zahtevkov bodo razdeljeni v skupine, ki bodo namenjene združevanju zahtevkov po področjih. Zahtevek bo lahko povezan z drugim zahtevkom, ki bo predstavljal njegovega starša, zato bodo lahko zahtevki urejeni v hierarhično drevesno strukturo, kot prikazuje Slika 5.1.



Slika 5.1: Drevesna struktura zahtevkov

Za vodenje in usmerjanje komunikacije bomo določili statuse zahtevkov in povezave med njimi. Za vsak status bomo nastavili vse možne naslednje statuse, zato bo možno tudi vračanje zahtevka v prejšnji status, kot je prikazano na Slika 5.2. Poudarjene in oštevilčene povezave prikazujejo primer življenjskega cikla zahtevka. Posameznemu statusu lahko vnaprej določimo parametre, s katerimi od uporabnika zahtevamo ali onemogočamo vnos določenih podatkov ob spremembi statusa. Poleg omenjene so možne še druge nastavitve: zahtevek se zaključi, podaljša se rok za izvedbo, omogoči se vnos opravljenega dela.



Slika 5.2: Primer povezav med statusi

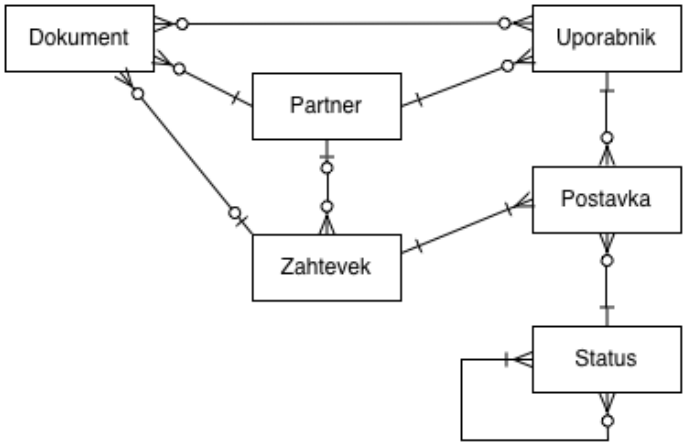
Pri vnosu zahtevka in nato pri vsaki spremembi statusa zahtevka bomo kreirali njegovo postavko, na kateri bodo poleg komentarja zabeleženi podatki o tem, kdo je kdaj in v kateri status prestavil zahtevek ter kdo je prejemnik.

Uporabniki bodo o prejetem zahtevku ali o spremembi na zahtevku lahko obveščeni preko spletne pošte ali z SMS sporočilom. Obveščanje bo realizirano tako, da bomo potrebne podatke za pošiljanje obvestila zapisali v podatkovno bazo ob spremembi statusa zahtevka. Na strežniku se bo vsakih nekaj minut zaganjala skripta, ki bo iz podatkovne baze pridobila podatke in poslala obvestila.

Dokumente, ki bodo nastali v procesu zahtevka, bodo lahko uporabniki dodali na zahtevek. Dodajanje dokumenta bomo realizirali s pojavnim oknom. Dokument bo možno povezati z zahtevkom, partnerjem ali z uporabniki. Prav tako pa bo mogoče dodajati dokumente neodvisno od partnerja in zahtevka. Tako bodo na primer lahko v organizaciji vodili prejeto pošto in jo posredovali prejemnikom v elektronski obliki. Vsak dokument bo lahko imel več prejemnikov, da se tako izognemo podvojenim vnosom dokumentov. Dokumenti bodo razdeljeni glede na njihov izvor, da bo mogoče ločeno pregledovanje interne, poslane ali prejete dokumente.

Za potrebe odnosov s poslovnimi partnerji bomo vodili njihov seznam. Na vsakega partnerja bo možno vnesti podatke kontaktnih oseb. Uporabniki bodo lahko na partnerja dodajali opombe, ki bodo namenjene predvsem beleženju aktualnih aktivnosti v zvezi s partnerjem, ki niso vezane na noben zahtevek.

Na Slika 5.3 so prikazane glavne entitete in relacije med njimi.



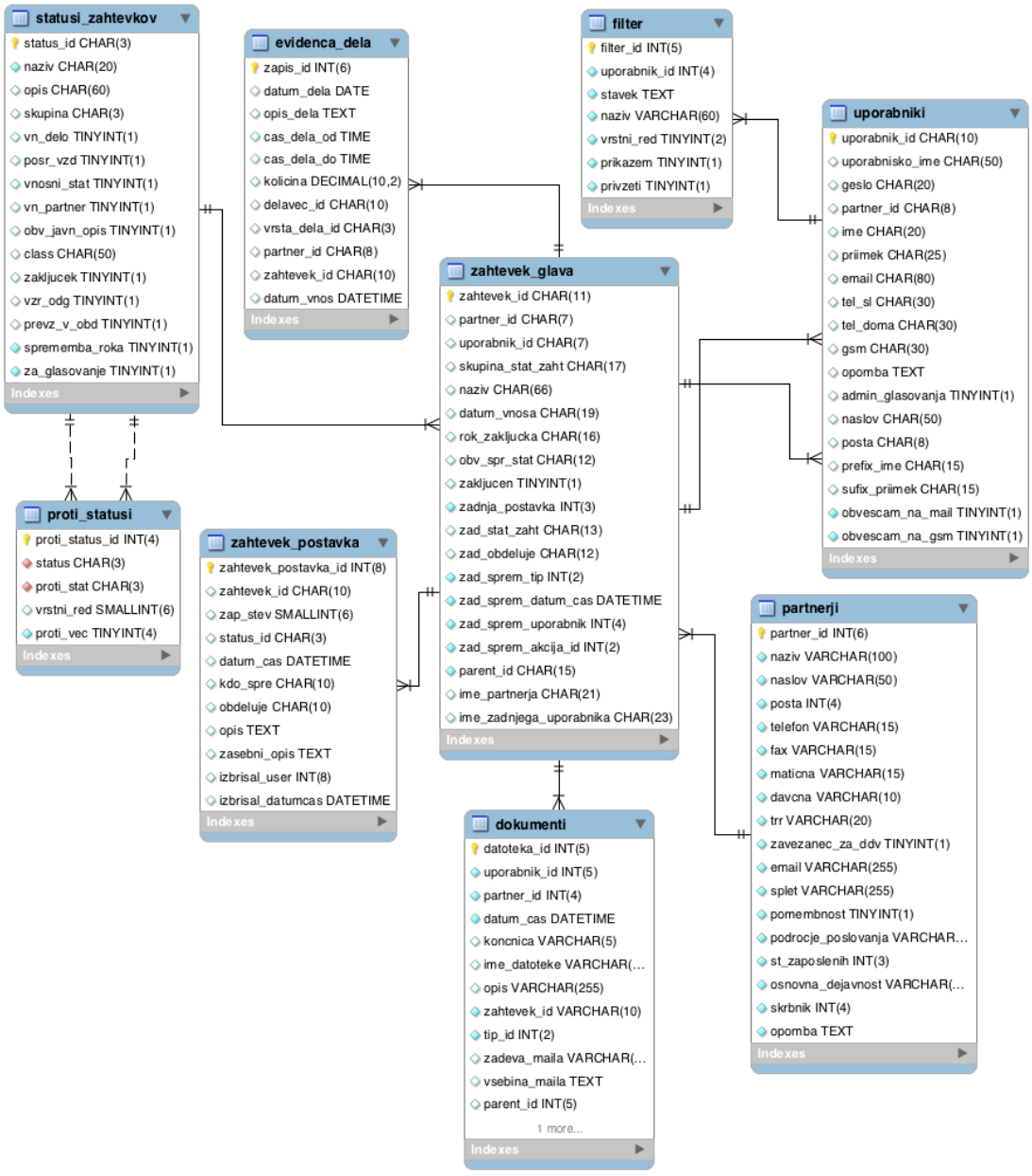
Slika 5.3: Entitetno-relacijski diagram

5.1.2 Podatkovna baza

Ko smo imeli definirane entitete in njihove povezave, smo entitetam določili vse potrebne atribute. Na Slika 5.4 je podatkovni model. Na diagramu so prikazane glavne entitete z atributi in pomembnejše povezave med njimi. Ker so entitete denormalizirane, je povezav med njimi več kot jih je na diagramu. Entitete so denormalizirane, da ni potrebno vedno združevati tabel pri poizvedbah in so zato lahko določene poizvedbe hitrejše.

Zaradi pregleda nad dejavnostjo uporabnikov in sledenju spremembam se bodo beležile vse akcije nad podatkovno bazo, ki jih bodo izvedli uporabniki. Tako bo možno vedno izvedeti, kdaj je kdo nazadnje urejal določen zahtevek, partnerja ali kaj drugega. Poleg tega, da bo možno spremljati aktivnost uporabnikov na zahtevkih, bo beleženje uporabno tudi, kadar bo prišlo do brisanj ali sprememb, ki so bile napačno izbrane.

Za sistem za upravljanje s podatkovnimi bazami smo izbrali odprtokodni MySQL.

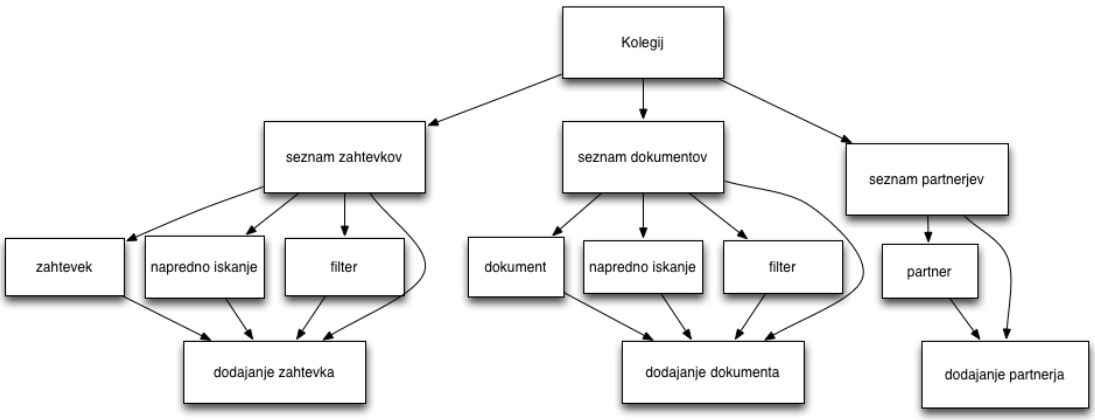


Slika 5.4: Podatkovni model

5.1.3 Uporabniški vmesnik

Glede na definirane funkcionalnosti smo sestavili diagram pogledov aplikacije, ki ga prikazuje Slika 5.5. Aplikacijo smo razdelili na tri module: zahtevki, dokumenti in partnerji. Vsak modul bo lahko deloval neodvisno od drugih dveh. Tako bodo lahko v vsaki organizaciji izbrali, katere module potrebujejo. Za učinkovito uporabo aplikacije morajo biti

funkcije za dodajanje zahtevka, dokumenta ali partnerja hitro dosegljive, zato bodo dostopne iz katerega koli dela posameznega modula.



Slika 5.5: Diagram pogledov

Pri načrtovanju uporabniškega vmesnika je bil poudarek predvsem na preprostosti za uporabo in možnosti za hiter začetek uporabe, brez posebnega uvajanja. Zato so na ekranu vedno prikazani samo podatki in funkcije, ki se navezujejo na kontekst. Tako so na primer polja za spremembo statusa pri odprtem zahtevku skrita in se prikažejo šele, ko izbereš naslednji status ali začneš vpisovati opis. Takrat se prikažejo samo polja, za katera se pričakuje uporabnikov vnos – odvisno od nastavitev na statusu.

Uporabniki se morajo zavedati, da ne morejo ničesar storiti narobe, saj ne morejo nič izbrisati. V najslabšem primeru se ponovi kakšen status ali se pošlje zahtevek k napačnemu uporabniku, ki slednjega nato lahko ustrezno posreduje naprej.

Gumbi za pomembnejše akcije, kot so shranjevanje ali potrjevanje zahteve, so bolj poudarjene in je zato uporaba bolj intuitivna.

Da je dostop do form za vnos zahtevka ali partnerja, ki se pogosteje uporabljajo, čimbolj enostaven, je realiziran s pojavnimi okni. Tako nam dodajanje zahtevka ne zmoti dela in lahko takoj, ko vnesemo zahtevek, nadaljujemo kjer smo ostali, ker se forma za vnos prikaže nad vsebino v pojavnem oknu. Z nastavitvami barv značk statusov lahko ustvarimo barvno kombinacijo, ki ustreza celostni podobi organizacije, pripadnosti določeni skupini statusov ali stopnji, v kateri je zahtevek.

5.2 Implementacija

Ker smo izbrali MVC arhitekturo aplikacije, smo morali pripraviti modele, poglede in krmilnike. Vsaka entiteta je dobila svoj model z vsemi atributi, nato smo za posamezni modul določili krmilnike in akcije, ki se bodo izvajale.

5.2.1 Orodja

Da je uporabniška izkušnja čimboljša, je večina funkcionalnosti in nalaganj vsebine realizirana s tehniko AJAX. Z AJAX-om dosežemo, da osveževanje in nalaganje cele spletne strani nista potrebna, ampak se lahko osveži samo del, ki se je spremenil. Tako ni potrebno osvežiti strani za dodajanje zahtevka, za spremembo statusa na zahtevku, pri urejanju seznama zahtevkov in skoraj pri vseh drugih akcijah, zato je aplikacija bolj odzivna in prijaznejša za uporabo. Za realizacijo AJAX tehnike smo uporabili ogrodje za JavaScript, imenovano jQuery, ki poenostavi izvajanje AJAX klicev in manipuliranje s HTML vsebino spletne strani. Za določene efekte smo uporabili tudi knjižnico za jQuery, imenovano jQuery UI, ki je namenjena generiranju grafičnih elementov in manipulaciji z uporabniškim vmesnikom.

Slika 5.6: Diagram programskih komponent

5.2.2 Varnost

Vsi podatki morajo biti ustrezno zaščiteni, da ne pride do zlorab, zato je potrebno zagotoviti, da lahko do podatkov dostopajo samo uporabniki, ki imajo ustrezne pravice. Ker dostop do aplikacije poteka preko HTTPS protokola, se vsi podatki med brskalnikom in strežnikom pošiljajo zakodirani. Vsi dokumenti se shranjujejo v mapi, ki ni dostopna preko spleta, da se onemogoči nenadzorovan dostop do teh datotek. [13]

Poleg tega, da moraš biti prijavljen v aplikacijo, da lahko izvajaš kakršne koli akcije, se pri pošiljanju podatkov na strežnik vedno zraven pošlje še žeton. Tako preprečimo, da bi lahko podatke na strežnik pošiljal kdo v imenu druge osebe. Vse akcije, ki se izvedejo na podatkovni bazi, se zabeležijo v posebno tabelo. Beležijo se podatki o času, uporabniku, tabeli, akciji in nekateri drugi. Tako imamo v primeru, da pride do težav, možnost vpogleda v zadnje dejavnosti.

Za dodatno varnost lahko poskrbijo tudi uporabniki sami, in sicer tako, da si izberejo dovolj močna gesla in da uporabljajo najnovejše verzije brskalnika Google Chrome ali Firefox.

5.2.3 Zahtevki

Pri spremembi statusa zahtevka uporabnik najprej izbere ustrezen status s seznama statusov, v katere lahko preide zahtevek. Ko je status izbran, se izvede AJAX klic na strežnik, ki je prikazan v Koda 5.1. Pri klicu se na strežnik pošlje podatek o izbranem statusu in zahtevku, na katerega se status navezuje. Zaradi varnosti je potrebno pri vsakem klicu poslati tudi unikaten žeton, da se na strežniku preverja pristnost izvora klica.

```
$.post("/status/sprememba", {
    'YII_CSRF_TOKEN': $("input[name=YII_CSRF_TOKEN]").val(),
    status_id: $("span#status").attr("data-id"),
    zahtevek_id: $("input#zahtevek_id").val()
}, function(data) {
    nastavitve = $.parseJSON(data);
    //kateremu uporabniku se posreduje zahtevek
    switch(nastavitve.posredovanje) {
        case "1"://ne
            $("div#posredujem_zahtevek").fadeOut();
        break;
        case "2"://da
            $("div#posredujem_zahtevek").fadeIn();
            $("input#posredujem").val(nastavitve.uporabnik_id);
        break;
        case "3": //obvezen vnos
           $("div#posredujem_zahtevek").fadeIn().find("td:first")
                                            .addClass("opozorilo");
        break;
        case "4": //ostane pri isti osebi
        case "5": //uporabniku, ki izvaja akcijo
            $("div#posredujem_zahtevek").fadeOut();
            $("input#posredujem").val(nastavitve.uporabnik_id);
        break;
        default:
            $("div#posredujem_zahtevek").fadeOut();
    }
});
```

Koda 5.1: Primer implementacije AJAX klica pri spremembi statusa zahtevka

AJAX klic povzroči na strežniku izvajanje funkcije, ki je implementirana v Koda 5.2. Funkcija iz prejetih parametrov '*status_id*' in '*zahtevek_id*' pridobi iz ustreznih modelov vse ostale podatke o statusu in zahtevku. Nato se glede na vrednost atributa '*posredovanje*' nastavi, kateri uporabnik bo dobil zahtevek v obdelavo. Vrednosti za izpis se shranjujejo v tabelo, ki se pred izpisom pretvori v JSON niz.

```
public function actionSprememba() {

            $status = Status::model()-
            >findByPk(CHttpRequest::getParam('status_id'));
            $zahtevek = Zahtevek::model()->findByPk(
                                    CHttpRequest::getParam('zahtevek_id'));

            switch($status->POSREDOVANJE) {
            case 2:  //da
                    $postavka = Postavka::model()->findByAttributes(array(
                            "UNI_ID"        => $zahtevek->UNI_ID,
                            "ZAP_STEV"      => $zahtevek->ZADNJA_POSTAVKA));
                    $uporabnik_id = $postavka->KOMU_PRED;
            break;
            case 4:  //istemu – obstoječi
                    $postavka = Postavka::model()->findByAttributes( array(
                            "UNI_ID"        => $zahtevek->UNI_ID,
                            "ZAP_STEV"      => $zahtevek->ZADNJA_POSTAVKA));
                    $uporabnik_id = $postavka->KOMU_PRED;
            break;
            case 5: //vpisovalcu zahtevka
                    $uporabnik_id = $zahtevek->ID_USER;
            break;
            case 6:
                    $uporabnik_id = Yii::app()->user->id;
                    break;
            }

            //…

            $array['posredovanje'] = $status->POSREDOVANJE;
            $array['sprememba_roka']       = $status->SPREMEMBA_ROKA;
            $array['rok']                  = date("d.m.Y", $rok);
            $array['partner_id']           = $zahtevek->ID_PARTNER;
            $array['uporabnik_id'] = $uporabnik_id;
            $array['obvezen_komentar']     = $status->OBVEZEN_KOMENTAR;

            echo json_encode( $array );
}
```

Koda 5.2: PHP funkcija za pripravo JSON podatkov

Strežnik na AJAX klic odgovori z JSON nizom (Koda 5.3), v katerem so podatki o tem, katero vnosno polje je potrebno prikazati na uporabniškem vmesniku, vnos katerega polja je obvezen za spremembo statusa zahtevka in kako naj bodo določena polja že izpolnjena. Tako je v nekaterih primerih, odvisno od nastavitev statusa, že določeno, kakšen bo rok zahtevka ali kateri uporabnik bo dobil ob spremembi statusa zahtevek v obdelavo. Odgovor strežnika dobi kot parameter funkcija, ki se izvede po uspešnem klicu (Koda 5.1). Funkcija prebere JSON niz in ustvari nov objekt '*nastavitve*'. Glede na vrednosti atributov tega objekta se nato

spremeni uporabniški vmesnik, tako da se določena vnosna polja prikaže ali skrije in nastavi njihove vrednosti.

```
{
    "posredujem": "6",
    "obvezen_komentar": "0",
    "sprememba_roka": "4",
    "rok": "23.01.2013",
    "partner_id": "700062",
    "uporabnik_id": "1345"
}
```

Koda 5.3: Primer JSON odgovora

Ko uporabnik vnese vse potrebne podatke in potrdi vnos spremembe statusa zahtevka, se ponovno izvede AJAX klic, ki pošlje vse vnesene podatke na strežnik. Na strežniku se nato ustvari nova postavka zahtevka, v odgovor pa pošlje novo osveženo stran pregleda zahtevka (Koda 5.4), ki nadomesti trenutno prikazano. Tako so spremembe na zahtevku vidne takoj po vnosu in brez tega, da bi v celoti osvežili spletno stran.

```
<div id="zahtevek">
    <table id="head-zaht">
        <tr>
            <td id="head-zaht-opis">
                <span id="zaht_id"><?php echo $zahtevek->UNI_ID;?>
                </span> - <?php echo $zahtevek->NAZIV; ?>
            </td>
        </tr>
    </table>
   <div id="main-zaht">
   <?php echo
        CHtml::form(CHtml::normalizeUrl('zahtevek/premembaStatusa'),
        'post', array("class" => 'form', "id" =>
        "spremembaStatusa")); ?>
...
```

Koda 5.4: Del HTML odgovora strežnika

5.2.4 Obveščanje

Pri spremembi statusa zahtevka lahko uporabnik izbere osebe, ki jih želi obvestiti o spremembi. Pri vnosu osebe na seznam za obveščanje lahko uporabnik izbere, ali bo prejemnik obvestila prejel SMS sporočilo ali elektronsko pošto. Zapis o obvestilu se zabeleži v tabelo '*obvestila*'. Na strežniku se nato vsako minuto pokliče konzolna skripta *PosiljanjeObvestil*, ki je implementirana v kodi 5.4. Skripta poišče vsa obvestila, ki še niso bila označena kot poslana in jih glede na atribut '*tip*' pošlje kot elektronsko sporočilo ali SMS. Sporočila SMS se pošiljajo preko ponudnika tovrstnih storitev. Za pošiljanje sporočila je potrebno na strežnik ponudnika poslati ustrezne podatke. Mi smo pošiljanje implementirali s pomočjo HTTP GET metode.

```
class PosiljanjeObvestilCommand extends CConsoleCommand {
   public function actionIndex() {
      $obvestila = Obvestila::model()->findAllByAttributes(array(
            "poslano" => 0,
            "zakljucen" => 1));
      foreach ($obvestila as $obvestilo) {
         if($obvestilo->tip == "1") {
            $email = Yii::app()->email;
            $email->subject = "Obvestilo o spremembi na zahtevku:
         ".$obvestilo->zahtevek->naziv;
            $email->message = $obvestilo->email();
            $email->to = $obvestilo->prejemnik->email;
            if($email->send()) {
               $obvestilo->poslano = 1;
               $obvestilo->poslano_cas = date("Y-m-d H:i:s");
               $obvestilo->save();
            }
         } else if($obvestilo->tip == "2") {
            $arr = array(
               'user' => urlencode('username'),
               'pass' => urlencode('password'),
               'gsm' => urlencode($obvestilo->prejemnik->gsm),
               'sms' => urlencode($obvestilo->sms()),
            );
            $url = 'http://www.ponudniksmsstoritev.si/posljiSms'
         .http_build_query($arr);
            $get = file_get_contents($url);
            if($get) {
               $obvestilo->poslano = 1;
               $obvestilo->poslano_cas = date("Y-m-d H:i:s");
               $obvestilo->save();
            }
         }
      }
   }
}
```

Koda 5.5: Implementacija skripte za pošiljanje obvestil

Poglavje 6

Možnosti izboljšav in razširitev

Ker se vedno bolj uporabljajo mobilne naprave, bi bilo potrebno prilagoditi aplikacijo za dostop iz teh naprav. Tako bi uporabniki lahko aplikacijo uporabljali na terenu in po potrebi podatke vnašali sproti. Z omogočenim dostopom za mobilne naprave bi imeli uporabniki dostop do relevantnih podatkov kjerkoli in kadarkoli.

Vedno bolj so v uporabi različne spletne storitve za hranjenje datotek v oblaku (Dropbox, Google Drive), zato bi lahko te storitve integrirali in izboljšali deljenje dokumentov. Tako bi lahko na primer uporabnik imel sinhronizirane vse dokumente iz zahtevkov, ki jih ima trenutno v obdelavi ali pa bi dokument lahko naložil tako, da bi ga skopiral v določeno mapo storitve Dropbox in ga nato v Kolegiju le še opremil z ustreznimi podatki. Uporabnikom, ki uporabljajo Google koledar, bi lahko omogočili dodajanje obvestil in avtomatski vnos pomembnih datumov.

Za organizacije, ki veliko komunicirajo preko telefona s poslovnimi partnerji, bi lahko dodali možnost klica partnerja neposredno iz aplikacije preko internetnega telefona. Prav tako bi lahko aplikacija avtomatsko odprla stran s podatki partnerja kadar bi na internetni telefon prispel klic s telefonske številke partnerja. Tako bi imel uporabnik, ki bi se javil na telefon, pred sabo vse relevantne informacije, ki jih potrebuje, da lahko izpelje pogovor.

Poglavje 7

Zaključek

V diplomski nalogi smo obravnavali področje komunikacije in sodelovanja v organizacijah. Razvili smo prototip spletne aplikacije, ki omogoča vodeno komunikacijo med člani organizacije. Vodeno komunikacijo smo realizirali s statusi, preko katerih gre zahtevek od svojega začetka do zaključka. Odgovorno osebo za določen zahtevek se izbere ob spremembi statusa, ko trenutno odgovorna oseba za zahtevek opravi svoje delo in zahtevek posreduje naslednjemu. Za primere, ko je delovni proces sestavljen iz več podprocesov, se lahko vsakemu zahtevku doda podzahtevek in komunikacija glede procesa in njegovih podprocesov ostane povezana. Ker komunikacijo pogosto spremljajo različni dokumenti, smo razvili tudi področje za vodenje dokumentov. Tako lahko uporabniki dodajo dokumente na zahtevke, poslovnega partnerja ali izbrane uporabnike – prejemnike dokumentov. Ker vsaka organizacija tako ali drugače posluje s poslovnimi partnerji, smo razvili tudi modul za vodenje poslovnih partnerjev.

Aplikacijo smo razvili do te mere, da je uporabna za manjšo organizacijo z nekaj deset člani. Primerna je predvsem za manjše organizacije, ki poslujejo s poslovnimi partnerji in organizacije, v katerih morajo člani sodelovati in si predajati zadolžitve z namenom, da zaključijo nek delovni proces. Menimo, da je aplikacija dovolj enostavna za uporabo, da uporabniki, razen kratke predstavitve koncepta, za prvo uporabo ne potrebujejo posebnega uvajanja.

Razvita aplikacija se od drugih razlikuje predvsem v tem, da je potrebno pred začetkom uvedbe aplikacije v organizacijo ustrezno nastaviti statuse zahtevkov, sicer aplikacija ne zagotavlja svoje glavne funkcije – učinkovite komunikacije. Za pravilno nastavitev statusov je potrebna temeljita analiza delovnih procesov organizacije, da je potem mogoče s statusi opisati vse delovne procese.

Literatura

[1] (2012) Email privacy. Dostopno na: http://en.wikipedia.org/wiki/Email_privacy

[2] Robert E. Kraut, Robert S. Fish , Robert W. Root, and Barbara L. Chalfonte, *Informal Communication in Organizations: Form, Function, and Technology*, Beverly Hills, CA: Sage Publications, 1990.

[3] Gerardine DeSanctis in Peter Monge, "Communication Processes for Virtual Organizations", *Journal of Computer-Mediated Communication*, št. 4, zv. 3, 1998.

[4] (2012) Panorama. Dostopno na: http://www2.kres-ks.si/dokumentacija/panorama/index.html

[5] Dejan Lavbič, Iztok Lajovic in Marjan Krisper, "Facilitating information system development with panoramic view on data", *Computer Science and Information Systems*, št. 4, zv. 3, str. 737-767, 2010.

[6] Jaka Žorž, "Podatkovna Panorama", *Moj mikro*, št. 9, str. 36-37, 2005.

[8] (2012) JIRA. Dostopno na: http://www.atlassian.com/software/jira/features

[7] (2012) Intrix. Dostopno na: http://www.intrix.si/sl/predstavitev

[9] (2012) Lotus Notes. Dostopno na: http://www-01.ibm.com/software/lotus/products/notes/features.html

[10] (2012) openCRX. Dostopno na: http://www.opencrx.org/

[11] (2012) Web application - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Web_application

[12] (2012) Scaffold (programming). Dostopno na: http://en.wikipedia.org/wiki/Scaffold_(programming)

[13] Tomaž Bratuša, *Hekerski vdori in zaščita*, Ljubljana, Slovenija: Pasadena, 2006.

[14] Harold J. Leavitt, "Some effects of certain communication patterns on group performance", *The Journal of Abnormal and Social Psychology*, št. 1, zv. 46, str. 38-50, 1951.