Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Peter Šfiligoj

Analiza primernosti orodja za hiter razvoj aplikacij

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Rok Rupnik

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

[1] V dogovorju z mentorjem lahko kandidat diplomsko delo s pripadajočo izvorno kodo izda tudi pod katero izmed alternativnih licenc, ki ponuja določen del pravic vsem: npr. Creative Commons, GNU GPL.

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: 00374/2013
Datum: 02.04.2013

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

Kandidat: **PETER ŠFILIGOJ**

Naslov: **ANALIZA PRIMERNOSTI ORODJA ZA HITER RAZVOJ APLIKACIJ**
THE ANALYSIS OF TOLL FOR RAPID APPLICATION DEVELOPMENT

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Za potrebe uskladitve potrebnih funkcionalnosti aplikacije z naročnikom praksa pokaže, da je smiselno razviti funkcionalni prototip aplikacije. Nujno je, da je tak prototip možno razviti hitro in učinkovito. Na ta način omogočimo odpravo visokega tveganja nesporazuma glede ključnih funkcionalnosti med razvijalci in naročnikom.
Proučite orodje za hiter razvoj aplikacij Iron Speed Designer. S pomočjo orodja razvijte prototip enostavne aplikacije ter na podlagi pridobljenih izkušenj opredelite prednosti in slabosti tega orodja.

Mentor:

doc. dr. Rok Rupnik

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Peter Šfiligoj, z vpisno številko **63060390**, sem avtor diplomskega dela z naslovom:

Analiza primernosti orodja za hiter razvoj aplikacij

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 11. septembra 2013 Podpis avtorja:

Kazalo

Povzetek

Namen diplomske naloge je predstaviti orodje za hiter razvoj aplikacij ter izdelavo prototipa spletne aplikacije s takim orodjem. Prototip spletne aplikacije v diplomski nalogi je preprosta aplikacija za zahteve po pomoči, kjer uporabniki oddajajo vprašanja v zvezi z računalnikom in računalniškimi programi. V diplomski nalogi najprej predstavimo prototipiranje programske opreme in metodologijo hitrega razvoja aplikacij in razložimo za kaj je pomembna uporaba prototipov programske opreme in kako nam hiter razvoj aplikacij pomaga pri izdelavi le teh. Nato pa predstavimo orodje za hiter razvoj aplikacij po imenu Iron Speed Designer in opišemo njegove osnovne funkcionalnosti ter postopoma prikažemo izdelavo spletne aplikacije za pomoč uporabnikom računalnika, ki je nastala v okviru diplomske naloge. V zadnjem delu pa naštejemo še prednosti in slabosti tega orodja.

Abstract

The purpose of the diploma thesis is to introduce a rapid application development tool and production of a web application prototype with this tool. The prototype in this thesis is a simple help desk application, where the users submit requests about computers and computer programs. In this diploma thesis we first introduce software prototyping and rapid application development methodology and we explain why is the use of software prototypes important and how rapid application development is helping for building them. Next, we introduce the rapid application development tool called Iron Speed Designer and we describe its basic functionalities and gradually show the development of a help desk web application for the computer users, which evolved in this thesis. Then in the last part we list the tool advantages and disadvantages.

Poglavje 1

Uvod

1.1 Motivacija in cilji

Izdelovanje spletnih aplikacij se vsakodnevno povečuje. To pomeni, da se pojavlja vedno več povpraševanja po spletnih aplikacijah. Preden je spletna aplikacija izdelana je vedno potrebno upoštevati zahteve uporabnikov, za katere se aplikacija izdeluje. Na podlagi zahtev naših uporabnikov se nato začne izdelovati aplikacijo. Pogosto pa se dogaja, da izdelovalci spletnih aplikacij narobe razumejo uporabnikove želje ali pa mogoče tudi uporabnik sam ne ve, kaj točno bi želel imeti. Zato je dobro izdelati prototip spletne aplikacije in ga predstaviti uporabniku. Ko uporabnik vidi prototip aplikacije in ga preizkusi, točno vidi ali je prototip aplikacije tisto kar si je želel ali ne. Tu pride v poštev prototipiranje (izdelovanje prototipov) programske opreme, ki je dejavnost v razvoju programske opreme s pomočjo katere izdelovalci najprej določijo osnovne zahteve uporabnikov in na podlagi zahtev izdelajo prototip aplikacije. Uporabniki nato izdelan prototip pregledajo in zagotovijo povratne informacije o aplikaciji. Na podlagi povratnih informacij lahko nato izdelovalec ugotovi ali je na pravi poti in začne izdelovati spletno aplikacijo na podlagi prototipa ali pa prototip spremeni in izboljša, če bi uporabnik to zahteval.

Prototipiranje je metoda, ki jo vključuje tudi metodologija Hiter razvoj

aplikacij. Ključni vidik hitrega razvoja aplikacij je izdelava prototipa aplikacije v zelo kratkem času ter zbrati in ugotoviti vse zahteve uporabnikov. Hiter razvoj aplikacij omogoča izdelovalcem spletnih aplikacij hitrejšo izdelavo prototipa, ker orodja za hiter razvoj aplikacij izdelajo ogrodje spletne aplikacije in grafične vmesnike, za katere je običajno potrebno veliko dela. Izdelovalci spletnih aplikacij si zato lahko pri izdelovanju prototipov pomagajo z orodji za hiter razvoj aplikacij.

Cilji te diplomske naloge so predstaviti bralcu prototipiranje programske opreme in metodologijo hitrega razvoja aplikacij ter pokazati pri katerih projektih je primerna uporaba prototipiranja in hitrega razvoja programske opreme, za konec pa opisati eno orodje za hiter razvoj aplikacij, predstaviti njegove funkcionalnosti in izdelati prototip spletne aplikacije. Izdelava prototipa spletne aplikacije bo prikazana z uporabo orodja Iron Speed Designer. S tem orodjem bomo izdelali preprost prototip spletne aplikacije za pomoč uporabnikom.

1.2 Pregled vsebine

V Poglavju 2 bomo spoznali kaj je to prototipiranje programske opreme. Predstavili bomo kakšen je namen izdelovanja prototipov in kakšne so koristi prototipiranja. Pri prototipiranju programske opreme obstajata dve dimenziji in več načinov prototipiranja, ki jih bomo tudi predstavili v tem poglavju. Predstavili bomo prednosti in slabosti prototipiranja ter povedali pri katerih projektih je uporaba prototipiranja primerna. Poglavje 3 bo opisovalo metodologijo Hiter razvoj aplikacij. Najprej bomo predstavili zgodovino in kako je prišlo do tega, da se je razvila metodologija hitrega razvoja aplikacij. Nato bomo spoznali ključne elemente hitrega razvoja aplikacij in kakšen je njegov proces, ki je sestavljen iz več faz življenjskega cikla aplikacije. Za konec poglavja pa bomo napisali katere so glavne prednosti in slabosti te metodologije ter za katere projekte naj bi se hiter razvoj aplikacij uporabljal. V Poglavju 4 bomo opisali orodje za hiter razvoj aplikacij po imenu Iron Speed Desi-

gner. Najprej bomo spoznali njegove osnovne funkcionalnosti. Nato bomo prikazali potek izdelave prototipa spletne aplikacije za pomoč uporabnikom računalnika in računalniških programov ter s tem prikazali še dodatne zmogljivosti tega orodja. Na koncu poglavja pa bomo spoznali katere prednosti in slabosti nam to orodje prinaša.

Poglavje 2

Prototipiranje programske opreme

Prototipiranje programske opreme (angl. software prototyping) je dejavnost v razvoju programske opreme, to je izdelovanje prototipov programskih aplikacij (angl. prototypes of software applications). Prototipi programskih aplikacij so nepopolne različice programov, ki jih razvijamo. Lahko jih primerjamo s prototipi iz ostalih področji kot na primer s prototipi v proizvodnji. Običajno prototip simulira le nekaj vidikov programske opreme in je v nekaterih primerih lahko popolnoma drugačen od končnega izdelka [1, 2].

Prototipiranje je nasprotje razvoja monolitnega cikla (angl. monolithic development cycle), ki je bil v uporabi v šestdesetih in sedemdesetih letih. Pri tem načinu razvoja se je že na začetku izdelalo celoten program in šele potem popravljalo neskladnosti med načrtovanjem in implementacijo. Uporaba monolitnega cikla razvoja je privedla do velikih stroškov programske opreme ter slabe ocene časa in stroškov za izdelavo le te [1]. Monolitni pristop je bil poimenovan kot tehnika "pobijanje zmaja (angl. slaying the dragon)", ker predpostavlja, da je načrtovalec programske opreme junak, kateri mora sam pokončati celega zmaja. Prototipiranje lahko prepreči velike stroške in težavnosti pri spreminjanju končnega produkta programske opreme.

Če povzamemo proces prototipiranja, ga lahko razdelimo na naslednje korake [1, 2, 3]:

- Opredelitev osnovnih zahtev:
  Določiti moramo osnovne zahteve. Kompleksnosti, kot je na primer varnost, običajno ne upoštevamo.

- Razvoj začetnega prototipa:
  Začetni prototip izdelamo tako, da bo vključeval samo uporabniške vmesnike.

- Pregled prototipa:
  Uporabniki preučujejo prototip in zagotovijo povratne informacije o dopolnitvah in spremembah.

- Revidiranje in okrepitev prototipa:
  Ko pridobimo povratne informacije uporabnikov, lahko izboljšamo specifikacije in prototip. Mogoče bodo potrebna pogajanja o tem, kaj bo v okviru pogodbe/produkta.

Če uvedemo spremembe, bo morda potreba po ponovitvi tretjega in četrtega koraka.

2.1 Namen in koristi pri izdelovanju prototipov programske opreme

Glavni namen prototipiranja programske opreme je ta, da omogočimo uporabnikom naše programske opreme, da preizkusijo program in tako ocenijo predloge razvijalcev za oblikovanje končnega produkta, namesto da bi dobili interpretacijo na podlagi opisov programa [1, 2]. Uporaba prototipov je koristna, ker lahko uporabniki simulirajo vedenje in funkcionalnosti programov ter nato opišejo zahteve, katerih razvijalci še niso upoštevali. Oblikovalec in izdajalec programske opreme lahko pridobita povratne informacije uporabnikov že v samem začetku projekta. Naročnik in pogodbenik lahko primerjata,

če programska oprema ustreza specifikacijam, po katerih naj bi bil program zgrajen. Poleg tega omogoča inženirjem programske opreme vpogled v pravilnost prvotnih ocen projekta ter če so bili roki in mejniki pravilno zastavljeni. Prototipiranje je zato lahko ključnega pomena v komercialnem odnosu med razvijalci in njihovimi strankami [4]. Tehnike, ki se uporabljajo za prototipiranje so bile v razvoju in razpravah že v začetku sedemdesetih, ko so bile te tehnike prvotno predlagane [3].

2.2 Dimenzije prototipiranja programske opreme

Prototipiranje vsebuje dve dimenziji, vertikalno in horizontalno [5, 6]:

- Horizontalni pristop:
  Horizontalni pristop je pogost izraz za prototip uporabniškega vmesnika. Zagotavlja širok pogled nad celotnim sistemom ali podsistemom. Osredotoča se bolj na uporabnikovo interakcijo s programom kot pa na nizko-stopenjske sistemske funkcionalnosti, kot je recimo dostopanje do baze podatkov. Horizontalni pristopi so uporabni za:
  - potrditev zahtev uporabniškega vmesnika ter obseg sistema,
  - prikaz različice sistema za pridobivanje zavzemanja iz poslovanja,
  - razvijanje predhodne ocene razvojnega časa, stroškov in truda.
- Vertikalni pristop:
  Vertikalni pristop prototipiranja je celotna izvedba enega podsistema ali ene funkcije. Ta pristop je uporaben za pridobivanje podrobnih zahtev za določeno funkcijo. Uporaben je za:
  - izboljševanje načrta podatkovne baze,
  - pridobivanje informacij o potrebi sistemskih vmesnikov za velikost omrežja in zmogljivosti strojne opreme,
  - pojasnjevanje kompleksnih zahtev z vrtanjem do dejanske sistemske funkcionalnosti.

2.3 Načini prototipiranja programske opreme

Pri prototipiranju programske opreme imamo več variant. Vse metode pa na nek način delimo na dva večja tipa prototipiranja: prototipiranje za enkratno uporabo (angl. throwaway prototyping) in razvojno prototipiranje (angl. evolutionary prototyping) [1, 2].

2.3.1 Prototipiranje za enkratno uporabo

Prototipiranje za enkratno uporabo, imenovano tudi hitro prototipiranje ali prototipiranje zaprtega tipa, je prototipiranje, pri kateremu izdelujemo model, ki ga sčasoma zavržemo, namesto da bi prototip postal del končne programske opreme. Najprej zberemo začetne zahteve uporabnikov in nato izdelamo enostaven delujoč model sistema, s katerim prikažemo uporabnikom, kako bodo njihove zahteve izgledale, ko bodo implementirane v končni sistem.

Hitro prototipiranje vključuje izdelavo delujočega modela iz različnih delov sistema že v samem začetku, po relativno kratki raziskavi. Metoda, ki jo uporabimo pri izdelavi je običajno precej neformalna, najpomembnejši dejavnik je hitrost s katerim zagotovimo model. Nato postane model izhodiščna točka iz katere lahko uporabniki ponovno preučijo njihova pričakovanja in pojasnijo njihove zahteve. Ko enkrat to dosežemo, model prototipa žavržemo"in začnemo formalno izdelovati sistem na osnovi ugotovljenih zahtev [7].

Najbolj očiten razlog za uporabljanje prototipiranja za enkratno uporabo je ta, da lahko prototip naredimo zelo hitro. Če dobijo razvijalci hitre povratne informacije uporabnikovih zahtev, lahko potem te zahteve izboljšajo že v zgodnji fazi razvoja programske opreme. Če spreminjamo projekt po tem, ko je bilo na njem opravljeno veliko dela, lahko majhne spremembe zahtevajo veliko napora za implementacijo, ker ima takrat sistem programske opreme že veliko odvisnosti. Zato je hitrost pri implementaciji prototipa za enkratno uporabo ključnega pomena, saj lahko naredimo prototip, ki bo potem zavržen z le malo porabljenega časa in denarja.

Še ena prednost pri prototipu za enkratno uporabo je njegova sposobnost

gradnje vmesnikov, katere lahko uporabniki zelo hitro testirajo. Uporabniški vmesnik je tisto, kar uporabniki vidijo kot sistem in ko ta sistem vidijo, lažje razumejo, kako bo ta sistem deloval.

Hitro prototipiranje je efektivnejši način za reševanje uporabnikovih vprašanj, ki so vezana na zahteve, in zaradi tega je razširitev produktivnosti programske opreme večja. Zahteve uporabnikov so lahko identificirane, simulirane in testirane hitreje in ceneje, kadar so vprašanja o nastajanju, vzdrževanju in strukturi programske opreme ignorirana. To vodi v natančnejše specifikacije zahtev ter posledično do izdelave veljavnega in uporabnega sistema z uporabnikovega vidika preko običajnih modelov razvoja programske opreme [8].

Prototipe lahko razvrstimo glede na zvestobo, po kateri so podobni dejanskemu produktu v smislu izgleda, interakcije in časovnega odziva. Ena izmed metod izdelave nizko-zvestega prototipa za enkratno uporabo je papirnato prototipiranje (angl. paper prototyping). Prototip izdelamo z uporabo papirja in svinčnika, tako da posnemamo delovanje dejanskega produkta, vendar je produkt po izgledu popolnoma drugačen. Druga metoda, za izdelavo visoko-zvestega prototipa za enkratno uporabo je ta, da uporabimo gradnike GUI (angl. GUI builder) in z njimi izdelamo samo nadomestek za klikanje (angl. click dummy). S tem izdelamo prototip, ki bo izgledal tak kot končni sistem, vendar pa ne bo imel nobenih funkcionalnosti.

Če povzamemo prototipiranje za enkratno uporabo vidimo, da pri tem pristopu izdelamo prototip z zamislijo, da ga bomo na koncu zavrgli in bomo začeli graditi končni sistem od začetka. Pri tem pristopu uporabimo naslednje korake:

- napišemo predhodne zahtevke,
- načrtujemo prototip,
- uporabniki začnejo uporabljati prototip in s tem določajo nove zahteve,
- po potrebi te korake ponovimo,

- napišemo zaključne zahteve,
- razvijemo končni produkt.

2.3.2 Razvojno prototipiranje

Razvojno prototipiranje se precej razlikuje od prototipiranja za enkratno uporabo. Glavni cilj tega prototipiranja je ta, da zgradimo zelo robusten prototip z strukturiranim načinom ter ga konstantno izboljšujemo. Ko zgradimo razvojni prototip, tvorimo jedro novega sistema in tako zgradimo nadaljnje zahteve in izboljšave. Ko razvijamo sistem z uporabo razvojnega prototipiranja se sistem nenehno izpopolnjuje in obnavlja.

Pri razvojnem prototipiranju ne razumemo in ne dobro poznamo vseh zahtev, pri tem pristopu izdelamo samo tiste zahteve, ki so res dobro razumljive [9]. Ta tehnika omogoča razvojni ekipi dodajanje funkcij in sprememb, ki niso mogle biti ustvarjene med zahtevami in fazo načrtovanja.

Sistem, ki bo uporaben, se mora razvijati med njegovo uporabo v njemu namenjenem delovnem okolju. Produkt ni nikoli "narejen", medtem ko se uporaba okolja spreminja, produkt dozoreva. Predpostavljamo lahko, kako se bo izvajala poslovna pot ter tehnologija na kateri bomo poslovanje implementirali. Sprejmemo načrt za razvoj zmogljivosti in prej ali slej bo nastalo nekaj podobnega predvidenemu sistemu.

Prednost razvojnega prototipiranja pred prototipiranjem za enkratno uporabo je v tem, da so tej prototipi funkcionalni sistemi. Kljub temu da nimajo vseh funkcionalnosti, katere so uporabniki planirali, se jih lahko uporablja začasno, dokler ni dostavljen končni sistem. Pri uporabi prototipnega okolja se lahko uporabniki uvedejo v začetni prototip medtem, ko čakajo na bolj razvito različico sistema. Za uporabnika je lahko boljše, da ima pomanjkljiv sistem kot pa da nebi imel nobenega sistema [7].

V razvojnem prototipiranju se razvijalci osredotočajo na razvoj tistih delov sistema, ki jih razumejo, namesto da bi gradili celoten sistem. Razvijalec ne implementira slabo razumljivih funkcionalnosti in s tem zmanjša tvega-

nje. Delni sistem pošlje stranki in ko uporabniki začnejo delati s sistemom, odkrijejo nove priložnosti za nove funkcionalnosti in tako oddajo razvijalcem zahteve za te funkcionalnosti. Nato razvijalci sprejmejo dodatne zahteve in jih uporabijo za spremembo specifikacij zahtev programske opreme, posodabljanje načrta ter prekodirajo in pretestirajo program [10].

2.3.3 Inkrementalno in ekstremno prototipiranje

Dva manjša tipa prototipiranja sta inkrementalno prototipiranje (angl. incremental prototyping) ter ekstremno prototipiranje (angl. extreme prototyping).

Pri inkrementalnem prototipiranju je končni produkt narejen kot več ločenih prototipov. Na koncu se te ločene prototipe združi v celoto.

Ekstremno prototipiranje uporabljamo predvsem za izdelovanje spletnih aplikacij. Spletni razvoj razbijemo na tri faze in vsaka faza je izdelana na podlagi prejšnje.

- Prva faza je statični prototip, ki je sestavljen predvsem iz HTML strani.
- V drugi fazi so na straneh sprogramirane funkcionalnosti.
- V tretji fazi so implementirane storitve.

Ta proces se imenuje ekstremno programiranje, ker dobi pozornost v drugi fazi procesa, v kateri razvijemo popolnoma funkcionalen uporabniški vmesnik in pri tem zelo malo upoštevamo tiste storitve, ki niso v njihovi pogodbi.

2.4 Prednosti in slabosti prototipiranja

2.4.1 Prednosti prototipiranja

Pri uporabljanju prototipov v razvoju programske opreme obstaja veliko prednosti. Nekaj je oprijemljivih, nekaj pa abstraktnih [10]. Prednosti prototipiranja programske opreme so:

- Zmanjšan čas in stroški:
  Uporaba prototipiranja lahko izboljša kvaliteto zahtev in specifikacij, ki so določene razvijalcem. Če odkrijemo spremembe kasneje v razvoju, so te dražje za implementacijo, zato lahko začetna ugotovitev, kaj si uporabniki zares želijo povzroči hitrejšo in cenejšo programsko opremo [8].

- Izboljšana in povečana udeležba uporabnika:
  Prototipiranje zahteva vključevanje uporabnikov in jim s tem omogoča, da vidijo prototip in delajo z njim. To omogoča uporabnikom boljše in popolnejše povratne informacije ter boljše specifikacije [7]. Ko uporabniki preučujejo prototip, lahko preprečimo številne nesporazume, ker lahko vsaka stran, tako na strani uporabnikov kot na strani razvijalcev verjame, da drugi razumejo to, kar so jim oni povedali. Ker poznajo uporabniki problem bolje kot razvojna ekipa, lahko povečana interakcija uporabnikov s prototipom povzroči boljšo kvaliteto v končnem produktu. Končni izdelek bo tako bolj zadovoljil uporabnikove želje po izgledu, občutku in delovanju produkta.

2.4.2 Slabosti prototipiranja

Uporaba oz. napačna uporaba prototipiranja ima tudi slabosti:

- Nezadostna analiza:
  Razvijalci se lahko preveč osredotočijo na prototip in tako odvrnejo pozornost od pravilne analize celotnega projekta. To lahko nato vodi do spregleda boljših rešitev, priprave nepopolnih specifikacij ali celo preobrazbo prototipov v slabo izdelane končne projekte, ki jih je nato težko vzdrževati. Ker ima prototip samo omejene funkcionalnosti, mogoče ne bo dovolj obsežen kot podlaga za izdelavo končnega produkta. To pa lahko razvijalci ne bodo opazili, če bodo preveč osredotočeni na prototip kot model.

- Zmedenost uporabnika pri prototipu in končnemu sistemu:
  Uporabniki lahko začnejo misliti, da je prototip dejansko končni sistem, ki mora biti samo dokončan ali izboljšan, kljub temu da bi se ga zavrglo. Pogosto se ne zavedajo napora, ki je potreben pri implementaciji lovilcev napak in funkcionalnosti za varnost, ki jih običajno prototip nima. Tako lahko uporabniki pričakujejo, da ima prototip eksaktne zmogljivosti končnega sistema, kar pa ni bilo mišljeno iz strani razvijalcev. Uporabniki se lahko tudi navadijo na določene funkcionalnosti, ki so jih razvijalci vključili v prototip samo za obravnavo in so bile kasneje odstranjene iz specifikacije za končni sistem. Če se zgodi, da uporabniki zahtevajo, da se vse predlagane funkcionalnosti vključijo v končni sistem, lahko to privede do konflikta.

- Napačno razumevanje razvijalcev o uporabnikovih ciljih:
  Razvijalci pogosto domnevajo, da si delijo skupne cilje z uporabniki, brez razumevanja o širših komercialnih problemih kot recimo zagotavljanje glavne funkcionalnosti pravočasno, v določenem roku in v okviru proračuna. Na primer, uporabnikom se pokaže določene funkcionalnosti, brez da se jim pove, da je za te funkcionalnosti potrebno dodatno kodiranje in da se pogosto zahteva več strojne opreme za upravljanje z dodatnimi dostopi do baze podatkov.

- Navezanost razvijalcev na prototip:
  Razvijalci se lahko zelo navežejo na prototipe, v katere so vložili veliko truda pri izdelovanju. To lahko privede do težav, če poizkusijo spreobrniti omejen prototip v končni sistem, ta prototip pa nima niti ustrezne osnovne arhitekture. V takem primeru lahko predlagamo uporabo prototipiranja za enkratno uporabo raje kot razvojnega prototipiranja.

- Prekomerni čas razvoja pri izdelavi prototipa:
  Ključna lastnost pri prototipiranju je dejstvo, da je prototip izdelan hitro. Če razvijalci spregledajo to dejstvo, se lahko zgodi, da začnejo izdelovati prototip, ki je preveč zapleten. Ko je prototip zavržen, lahko

zahteve, ki so zagotovljene, ne dajejo zadostnega povečanja v produktivnosti, da bi nadoknadile čas, ki je bil porabljen pri izdelavi prototipa. Uporabniki lahko obtičijo pri razpravah o podrobnostih prototipa ter s tem zadržujejo razvojno ekipo in odlašajo končni produkt.

- Stroški pri implementaciji prototipa:
  Začetni stroški pri gradnji razvojne ekipe, ki je osredotočena na prototipiranje so lahko veliki. Veliko podjetji že ima svoje metodologije razvoja, in spreminjanje le teh lahko pomeni ponovno prekvalificiranje ljudi, menjavo orodji, ali oboje. Podjetja se pogosto nagibajo k temu, da samo skočijo v prototipiranje, brez da bi jih skrbelo prekvalificiranje njihovih delavcev.

Pogosta težava pri sprejemanju tehnologij za prototipiranje je tudi preveliko pričakovanje za produktivnost z malo napora in novega učenja. Poleg usposabljanja ljudi z uporabo tehnik za prototipiranje, je pogosto spregledana tudi potreba po razvoju specifične osnovne strukture projekta za podporo tehnologije. Če to osnovno strukturo prezremo, je pogosto produktivnost nižja [12].

2.5 Projekti, pri katerih je dobro uporabljati prototipiranje

Razpravljalo se je, da bi se prototipiranje uporabljalo ves čas, v takšni ali drugačni obliki. Vendar pa je prototipiranje najbolj koristno v tistih sistemih, ki bodo imeli veliko interakcij z uporabniki [1].

Ugotovljeno je bilo, da je uporaba prototipov zelo učinkovita pri analizi in načrtovanju spletnih sistemov, zlasti tistih za obdelavo transakcij, kjer je uporaba pogovornih zaslonov (angl. screen dialogs) veliko bolj vidna. Večja kot je interakcija med uporabniki in računalnikom, večje koristi lahko pridobimo iz izdelave hitrega sistema s katerim se uporabniki ukvarjajo [7].

Sistemi pri katerih je interakcija z uporabniki majhna, imajo zelo majhne koristi od prototipiranja (npr. procesi za serijsko obdelavo ali procesi, ki po večini delajo z kalkulacijami). Včasih je lahko kodiranje, ki je potrebno za izvedbo sistemskih funkcij morda preveč intenzivno in so potem potencialne koristi, katere zagotavlja prototipiranje premajhne [7].

Prototipiranje je še posebej koristno za oblikovanje dobrih interakcij med človekom in računalnikom (angl. human-computer interfaces). Eden izmed najproduktivnejših uporab hitrega prototipiranja je bilo orodje za tehnike iterativnih uporabnikovih zahtev in izdelava vmesnika med računalnikom in človekom [8].

Poglavje 3

Hiter razvoj aplikacij

Hiter razvoj aplikacij (angl. rapid application development, krat. RAD) je metodologija za razvoj programske opreme (angl. software development methodology), ki uporablja minimalno načrtovanje v korist hitrega prototipiranja. Načrtovanje razvoja programske opreme z uporabo hitrega razvoja aplikacij je povezano s pisanjem programske opreme. Programsko opremo lahko napišemo veliko hitreje in enostavneje spreminjamo zahteve, če imamo zmanjšan obseg predhodnega načrtovanja.

Hiter razvoj aplikacij omogoča programerjem hitrejšo izdelavo delujočih programov. Sistemi RAD zagotavljajo orodja za pomoč pri izgradnji grafičnih uporabniških vmesnikov, za katere bi bilo običajno potrebno veliko napora pri razvoju [14]. Osredotoča se na razvoj aplikacij v zelo kratkem času, s hitrim izvajanjem, funkcionalnostmi in uporabnostjo. Beseda RAD je v zadnjem času postala modna beseda na tržišču, ki opisuje aplikacije, katere lahko izdelamo in razvijemo v času od 60 – 90 dni, vendar pa je bila prvotno mišljena za opis procesa razvoja, ki vključuje prototipiranje aplikacij (angl. software prototyping) in iterativni razvoj (angl. iterative development) [13].

Metodologija RAD vključuje metode, kot so iterativni razvoj in prototipiranje programske opreme. Whitten (2007) pravi, da je to združitev različnih strukturiranih tehnik (angl. structured techniques), predvsem podatkovno usmerjenih informacijskih tehnik (angl. data-driven information engienee-

ring) s tehnikami za prototipiranje (angl. prototyping techniques) za pospešitev razvoja programskih sistemov [12].

Strukturirane tehnike in prototipiranje so v hitrem razvoju aplikacij uporabljene zlasti za definicijo uporabniških zahtev in za oblikovanje končnega sistema. Z uporabo strukturiranih tehnik se začne proces razvoja z razvojem predhodnih podatkovnih modelov in modelov poslovnih procesov. V nadaljevanju se preveri zahteve z uporabo prototipiranja in sčasoma dodela podatkovne in procesne modele. Te faze se ponavljajo iterativno. Kot rezultat nadaljnjega razvoja dobimo kombinacijo poslovnih zahtev in tehničnega načrtovanja za uporabo pri izdelavi novih sistemov [12].

Pristop hitrega razvoja aplikacij lahko pomeni kompromise pri funkcionalnosti in zmogljivosti v zameno za hitrejši razvoj in lažjega vzdrževanja aplikacije.

3.1 Zgodovina hitrega razvoja aplikacij

Hiter razvoj aplikacij obstaja približno dvajset let, ampak še danes velja za to, kar je bil sprva mišljen [13]. Pogledali si bomo problem razvoja aplikacij in rešitev, ki jo prinaša hiter razvoj aplikacij.

3.1.1 Problem razvoja aplikacij

Procesi, ki so se razvijali v sedemdesetih in osemdesetih letih kot strukturirani sistemi za analiziranje (angl. Structured Systems Analysis), načrtovalne metode (angl. Design Method) ter ostali modeli vodnega slapa (angl. Waterfall model), pogosto niso izpopolnjevali potreb strank pri razvoju aplikacij [13, 14]. Izdelava aplikacij z omenjenimi metodologijami je potekala toliko časa, da so se zahteve strank spremenile še preden je bil sistem dokončan. Večji projekti, kjer so se uporabljale te metodologije, so bili pogosto dokončani, vendar neustrezni ali pa celo neuporabni sistemi.

Vzrok težave je bil v tem, da so strogo upoštevali zaključek ene faze življenjskega cikla aplikacije, preden so prešli na naslednjo fazo življenjskega

cikla aplikacije. Gradnja aplikacije na podlagi zahtev, katere so bile na neki točki zamrznjene, je povzročila to, da daljši kot bo razvoj, večja bo možnost, da se bodo zgodile spremembe v poslovanju in tako se bodo izničile zahteve na katerih je temeljil razvoj sistema [13]. Druga težava je bila predpostavka, da bi faza analiziranja metodičnih zahtev sama identificirala vse kritične zahteve [14].

3.1.2 Rešitev, ki jo prinaša hiter razvoj aplikacij

Na težavo je bilo veliko odgovorov od osemdesetih do danes. Leta 1986 je Barry Boehm napisal razvoj programske opreme z spiralnim modelom (angl. Spiral Model of Software Development and Enhancement), ki je bil sprva definiran kot koncept izdelave prototipov in iterativnega razvoja ter ciljal na zmanjševanje tveganja. V poznih osemdesetih sta Scott Schultz in James Martin izpopolnila idejo o izdelavi prototipov in iterativnem razvoju z metodologijo po imenu hitro iterativno prototipiranje za produkcijo (angl Rapid Iterative Production Prototyping, krat. RIPP), ki je ciljala na razvoj sistemov v kratkem časovnem razvoju z majhnimi ekipami, ki so bile zelo usposobljene, motivirane in imele izkušeno osebje. James Martin je leta 1991 razširil in formaliziral metodologijo, ko je objavil knjigo Rapid Application Development [13].

3.2 Ključni elementi hitrega razvoja aplikacij

Hiter razvoj aplikacij ima veliko ključnih elementov, kateri ga naredijo kot edinstveno metodologijo. Vključuje izdelavo prototipov, iterativni razvoj, pakiranje časa, člane skupine, pristope vodenja in RAD orodja [13, 16, 17].

3.2.1 Prototipiranje (angl. prototyping)

Ključni vidik hitrega razvoja aplikacij je izdelava prototipa za začetno obliko ter zbrati in ugotoviti kakšne so vse uporabnikove zahteve. Cilj je zgraditi

lahko različico končne verzije za produkcijo v zelo kratkem času, po možnosti v dnevih. Začetni prototip naj služi kot dokaz koncepta stranke, vendar je pomembneje to, da služi kot orodje za prečiščevanje zahtev stranke.

3.2.2 Iterativni razvoj (angl. iterative development)

Iterativni razvoj pomeni postopno izdelovanje funkcionalnih različic sistema v kratkih ciklih razvoja. Vsako različico si pogledamo skupaj s stranko ter tako pridobimo zahteve, ki bodo potrebne za naslednjo različico. Postopek se ponavlja, dokler niso pokrite in narejene vse funkcionalnosti aplikacije. Idealna dolžina iteracij je med enim dnevom in tremi tedni. Vsak cikel razvoja omogoča uporabniku, da pridobi povratne informacije, izboljša zahteve ter vidi napredek v razvoju. Iterativni razvoj je tisti, ki rešuje težave, ki so povezane z nefleksibilnimi metodologijami, ki so bile ustvarjene v sedemdesetih letih.

3.2.3 Pakiranje časa (angl. time boxing)

Pakiranje časa je določeno časovno obdobje med katerim moramo opraviti določeno nalogo. Pri hitrem razvoju aplikacij je to proces v katerem odstranimo funkcije kasnejših različic aplikacije, zato da dokončamo trenutno različico aplikacije v čim krajšem možnem času. Strogo pakiranje časa je pomemben vidik hitrega razvoja aplikacij, ker se drugače lahko razvojne iteracije zelo podaljšajo in morebitno vrnejo nazaj do pristopa metodologije modela vodnega slapa.

3.2.4 Člani razvojne skupine (angl. team members)

Metodologija hitrega razvoja aplikacij priporoča uporabo majhnih razvojnih ekip, ki so sestavljene iz izkušenih, vsestranskih in motiviranih članov, ki so sposobni opravljati več funkcij. Ker ima stranka ključno vlogo v procesu razvoja, morajo biti sredstva, ki so namenjena strankam na voljo že na začetnih sestankih in tudi na sestankih, ki se izvajajo na koncu razvojnih

ciklov. Za razvojne ekipe bi bilo zato najbolje, da imajo veliko izkušenj s hitrim razvojem aplikacij.

3.2.5 Pristopi vodenja (angl. management approach)

Aktivno in vpleteno vodenje je ključnega pomena za zmanjševanje tveganja, da bi se podaljšali cikli razvoja, da bi prišlo do nesporazumov s stranko in do zgrešenih rokov dobave aplikacije. Vodenje mora biti predvsem krepko in konsistentno pri uporabi metodologije hitrega razvoja aplikacij. Poleg uveljavitve strogega časovnega poteka, mora vodenje ciljati na izbiro članov razvojne ekipe, motivacijo ekipe ter urediti birokratske in politične ovire.

3.2.6 RAD orodja (angl. RAD tools)

Eden glavnih ciljev metodologije hitrega razvoja aplikacij, ki jo je razvil James Martin v poznih osemdesetih, je bila možnost izkoriščanja najnovejših tehnologij, ki so bile na razpolago za pospešitev razvoja. Danes je jasno, da je tehnologija iz osemdesetih zastarela, vendar je fokus hitrega razvoja aplikacij na najnovejša orodja prav tako pomemben kot takrat, ko je bila metodologija prvotno ustvarjena. V nadaljevanju se bomo posvečali procesu hitrega razvoja aplikacij.

3.3 Proces hitrega razvoja aplikacij

Proces hitrega razvoja aplikacij je sestavljen iz štirih faz življenjskega cikla. Te faze so načrtovanje zahtev, uporabniško načrtovanje, izdelava aplikacije in implementacija aplikacije [13, 18, 19]. Poleg naštetih poznamo tudi pred-projektne in po-projektne aktivnosti.

3.3.1 Pred-projektne aktivnosti (angl. pre-project activities)

Pri vsakem projektu je ključnega pomena, da identificiramo vse podrobnosti projekta v naprej in jih damo v neko obliko dokumenta. Vse skupine se morajo dogovoriti v naprej o podrobnostih kot so potencialna tveganja, strategije, urnik razvoja vključno z viri, mejniki in dobavljanjem (kot na primer dobava celotnega podatkovnega modela ali vrste dokumentacije), pristop, ki vsebuje standarde, orodja in tehnologije, ki se bodo uporabljale, želen končni rezultat, želene pogoje in omejitve ter kakšni bodo stroški za izdelavo aplikacije ter stroški orodji za uporabo aplikacije.

3.3.2 Načrtovanje zahtev (angl. requirements planning)

Faza načrtovanja zahtev je sestavljena iz sestankov med ekipo načrtovanja zahtev ter ključnimi uporabniki strank. Sestanki ciljajo na pridobitev pomembnega seznama začetnih zahtev ter določitev obsega projekta. Ekipa načrtovanja zahtev razpravlja o poslovnih potrebah, obsegu projekta, omejitvah in sistemskih zahtevah ter identificira primarne poslovne funkcije in jih razbije na manjše poslovne entitete. V fazi načrtovanja zahtev moramo imeti seznam entitet in diagramov, ki definirajo interakcijo med procesi in podatkovnimi elementi. Faza naj bo izvedena med enim in štirimi tedni. Bolje je, da hranimo idealne zahteve v strukturiranih orodjih kot pa v nestrukturiranem dokumentu. Faza se zaključi, ko ekipa potrdi ključne probleme in pridobi dovoljenje za nadaljevanje dela. Na koncu faze načrtovanja je potrebno narediti še oceno dela.

3.3.3 Uporabniško načrtovanje (angl. user design)

V fazi uporabniškega načrtovanja se ekipa analitikov zbere s končnimi uporabniki. Skupaj komunicirajo ter razvijajo modele in prototipe, ki pred-

stavljajo vse sistemske procese, vhode in izhode. Ekipa analitikov povzame zahteve veliko podrobneje ter razvije entitete, ki so bile razvite v načrtovanju zahtev v podatkovni model, formalizira poslovna pravila, razvije testne načrte ter ustvari pregled nad tokovi in postavitvami pomembnih delov sistema. Skupina hitrega razvoja aplikacij običajno uporablja kombinacijo tehnik razvoja skupnih aplikacij (angl. Joint Application Development, krat. JAD) in orodji CASE (angl. CASE tools) za pretvarjanje uporabniških zahtev v delujoče modele.

V drugi polovici faze uporabniškega načrtovanja, razvijalna ekipa pomaga ekipi analitikov pri izdelavi delujočega podatkovnega modela, ki se ga lahko pretvori v funkcionalno bazo podatkov ter pri identifikaciji komponent za ponovno uporabo. Dobro je, če vse zahteve zajamemo v nekem orodju, ki je za to namenjeno.

Pred prehodom na fazo izdelave aplikacije mora ekipa analitikov ciljati na naslednje korake tako, da se osredotoči na ocenjevanje projekta. Osredotočanje na naslednje korake je pomemben element v fazi uporabniškega načrtovanja, ker naj bi se začetna iteracija pri fazi izdelovanja aplikacije osredotočala na funkcije lahkega prototipa (angl. light prototype). Da bi ohranili razvojne iteracije čim krajše in da bi dobili čim večjo korist hitrega razvoja aplikacij, je potrebno identificirati ključne zahteve za začetni prototip. Ostale zahteve pa naj bodo identificirane v prihodnjih razvojnih iteracijah.

Faza uporabniškega načrtovanja naj traja v obdobju od treh do petih tednov.

3.3.4 Izdelovanje (angl. construction)

V fazi izdelovanja aplikacije ekipa načrtovalcev razvija aplikacijo v iterativnih ciklih razvoja, testira aplikacijo, dodela zahteve ter nato razvija še enkrat, dokler ni aplikacija končana. Razvojne iteracije naj bi trajale od enega dneva do treh tednov. Ekipa razvijalcev mora spremeniti podatkovni model, ki je bil izdelan med fazo uporabniškega načrtovanja v funkcionalno bazo podatkov. Uporabniki v tej fazi še naprej sodelujejo in še vedno predlagajo spremembe

ali izboljšave programa. V tej fazi so torej glavne naloge programiranje in izdelava aplikacije, kodiranje, integriranje elementov in testiranje sistema.

Ključnega pomena je da obdržimo vsako razvojno iteracijo v pravi smeri. Morda je potrebno odstraniti določene funkcionalnosti, da obdržimo razvoj v določenem časovnem obdobju. Vodenje ima v tej fazi ključno vlogo, saj mora zagotavljati, da napreduje vse po načrtu. Obdržati mora stranko na tekočem ne glede na spremembe funkcionalnosti. Poleg tega pa mora obdržati tudi ekipo motivirano.

Ko je prototip razvit (v svojem določenem časovnem obdobju), ekipa izdelovanja testira začetni prototip z uporabo skript za testiranje, ki so bile izdelane med fazo uporabniškega načrtovanja. Aplikacijo pregledajo ekipa načrtovanja in stranka. Na koncu se ekipa izdelovanja, ekipa načrtovanja ter stranka dobijo na sestanku in tam skupaj določijo zahteve za naslednjo iteracijo. Sestanki so sestavljeni iz lažjih sej, ki naj bi trajale približno dve uri. Posrednik mora poznati vsa področja, ki bodo potrebovala razpravo in mora zagotoviti, da vsak problem pridobi dovolj pozornosti. Primerno je tudi, da ima seznam problemov, ki bodo potrebovali dodatne pozornosti in bodo na razpravah v ločenem sestanku.

Po sestanku (ali sestankih, če jih je potrebno več) je dobro, da ekipa razvijalcev in ekipa načrtovalcev posodobi zahteve, podatkovni model, skripte za testiranje in plan projekta tako kot v fazi uporabniškega načrtovanja. Ponovno morajo ekipe tudi identificirati primarne ter sekundarne zahteve, narediti načrt za naslednjo razvojno iteracijo, obdržati uporabnika na tekočem ne glede na to kaj se bo naredilo in nato začeti naslednjo razvojno iteracijo znova. Ko se začne sistem približevati zadostnemu stanju, mora razvojna ekipa ciljati na sistem bolj kot na končno aplikacijo kot pa na prototip aplikacije.

Med končnimi razvojnimi iteracijami mora ekipa načrtovalcev posodobiti uporabniško dokumentacijo, opraviti test uporabnikove odobritve (angl. User Acceptance Testing) in opredeliti korake, ki so potrebni za namestitev/implementacijo.

3.3.5 Implementacija (angl. implementation)

V fazi implementacije integriramo nove sisteme v poslovanje. V primerjavi s tradicionalnimi metodami je tukaj celoten proces skrčen. Nov sistem je izdelan, dostavljen in nameščen v delovanje veliko hitreje. Naloge v tej fazi so predelava podatkov, celotno testiranje, prenova sistema ter izobraževanje uporabnikov. Ekipa razvijalcev pripravi podatke in implementira vmesnike ostalim sistemom. Ekipa načrtovanja usposablja uporabnike sistema, medtem ko uporabniki izvajajo test odobritve. Ekipa načrtovalcev pomaga uporabnikom prehod iz njihovih starih procedur na novejše, ki vsebujejo nov sistem. Nato pregleda težave po namestitvi in identificira ter spremlja potencialne izboljšave. Čas porabljen za dokončanje faze implementacije je odvisen od projekta.

3.3.6 Po-projektne aktivnosti (angl. post-project activities)

Kot pri vseh projektih, ki so dokončno dobavljeni stranki, morajo biti opravljene takšne aktivnosti, ki bodo v korist prihodnjim projektom. Bolj specifično povedano, potrebno je, da vodja projekta pregleda ter dokumentira projektne meritve, organizira in shrani projektna sredstva (kot so komponente za ponovno uporabo, načrt projekta, načrt vodenja projekta in načrt testiranja). Prav tako je dobra praksa, da pripravi kratek dokument, ki opisuje izkušnje, pridobljene pri izdelovanju projekta.

3.4 Prednosti in slabosti hitrega razvoja aplikacij

Glavne prednosti hitrega razvoja aplikacij sta večja hitrost in kakovost izdelka, medtem ko sta glavni dve pomanjkljivosti manjša nadgradljivost in manj funkcionalnosti aplikacije [13, 16].

3.4.1 Prednosti hitrega razvoja aplikacij

3.4.1.1 Večja hitrost

Kot že samo ime pove, je glavna prednost hitrega razvoja aplikacij večja hitrost razvoja aplikacije in hitrejši čas dobave. Cilj hitre dobave aplikacije je pretvorba zahtev v kodo kakor hitro je to mogoče. Tukaj se vidi tudi pomembnost pakiranja časa, v katerem izrinemo funkcionalnosti za prihodnje izdaje aplikacije, da lahko potem dokončamo funkcionalnosti lahke verzije hitreje.

3.4.1.2 Večja kakovost

Povečana kakovost aplikacije je glavni poudarek metodologije sistema za hiter razvoj aplikacij, vendar pa ima izraz drugačen pomen kot je običajno povezano z sistemom za razvoj aplikacij po meri (angl. Custom Application Development). Pred hitrim razvojem aplikacij, kar je bilo mogoče tudi bolj intuitivno, je bila kakovost v razvoju kot stopnja v kolikšni meri je aplikacija ustrezala specifikacijam ter kako malo napak je imela takrat, ko je bila dobavljena uporabnikom. Za hiter razvoj aplikacij pa je kvaliteta mišljena kot stopnja v kolikšni meri aplikacija, ki je dobavljena uporabnikom ustreza zahtevam uporabnikom in v kolikšni meri ima aplikacija nizke stroške vzdrževanja. Hiter razvoj aplikacij poskuša imeti kakovosti, ki bi ustrezala uporabnikom zlasti v fazi načrtovanja.

3.4.1.3 Ostale pomembne prednosti hitrega razvoja aplikacij [20, 21]

- Hiter razvoj aplikacij je fleksibilen in prilagodljiv spremembam.
- Čas, ki je potreben za razvoj programske opreme, je drastično zmanjšan, zaradi zmanjšanega analiziranja zahtev, kot so zahteve poslovne dokumentacije in zahteve programske opreme ter faze načrtovanja.
- Prototipiranje aplikacij pomaga oceniti, če se kritične sistemske zahteve ujemajo s sistemom ali ne. Izhodiščno poročilo se lahko primerja z obstoječimi poročili.
- Vse prototipe programske opreme, ki so bili narejeni lahko hranimo v repozitoriju za nadaljnjo uporabo. Ponovna uporaba komponent krepi hitrost procesa pri razvoju programske opreme.
- Vodja projekta je veliko lažje natančen pri ocenjevanju stroškov projekta, kar pomeni, da je nadzor stroškov prometa izvajati in upravljati enostavnejše.
- Hiter razvoj aplikacij je lahko velik prihranek z vidika proračuna in časa projekta ter ponovne uporabe prototipov.
- Ima kratek razvojni cikel, tako da lahko uporabniki vidijo RAD izdelek zelo hitro.
- Če izberemo komponento iz repozitorija, vemo, da je že testirana in nam jo zato ne bo potrebno testirati še enkrat. To nam prihrani veliko časa, ki je potrebno za testiranje.
- Zahteve za upravljanje projekta so zbrane na dinamičen način. Vsakič, ko je prototip narejen, se zahteve preštudira, če se ujemajo. Če se pojavijo kakršnekoli dodatne zahteve, so te vključene v naslednjem razvoju prototipa.

- Med projektom je nenehna udeležba uporabnikov, ki daje povratne informacije v celotnem procesu. Zato bo stopnja zadovoljstva pri končnih uporabnikih višja, ko bo izdelan končni rezultat.
- Zavzema se za boljšo dokumentacijo skozi pisne testne primere.

3.4.2 Slabosti hitrega razvoja aplikacij

3.4.2.1 Manjša nadgradljivost

Ker se hiter razvoj aplikacij osredotoča na razvoj prototipa, kateri se razvija iterativno v celoten sistem, morda končna rešitev ne bo imela možnosti nadgradnje produkta, ki je bil načrtovan kot celotna aplikacija na začetku.

3.4.2.2 Manj funkcionalnosti

Pri pakiranju časa, kjer so funkcionalnosti aplikacije prestavljene na kasnejše različice aplikacije, da bi bila aplikacija dostavljena v zelo kratkem času, obstaja možnost da se s hitrim razvojem aplikacij izdela takšne aplikacije, ki so manj funkcionalne kot običajno razvite aplikacije. Temu problemu se je potrebno izogniti kakor hitro je mogoče, tako da z komuniciranjem s stranko izvemo kaj točno ji bomo dostavili in kdaj.

3.4.2.3 Ostale pomembne slabosti hitrega razvoja aplikacij [20, 21]

- Ta metoda ni uporabna za velike, edinstvene ali zelo zapletene projekte.
- Metoda ne more biti uspešna, če ekipa ni dovolj motivirana in je nezmožna delati skupaj.
- Uspeh je odvisen od izjemno visoke tehnične spretnosti razvijalcev.
- Lahko se zgodi da mnogi pomembni uporabniki ne bodo imeli dovolj časa in ne bodo morali nameniti dovolj pozornosti razvoju aplikacije za uspešen proces hitrega razvoja aplikacij.

- Včasih ekipa ignorira parametre kakovosti kot so skladnost, zanesljivost in standardizacija. Zato je lahko težko implementirati kvaliteto vodenja projekta med življenjskim ciklom vodenja projekta.

Poglavje 4

Orodje za hiter razvoj aplikacij Iron Speed Designer in primer spletne aplikacije

Sedaj ko smo spoznali, kaj je prototipiranje in kaj je hiter razvoj aplikacij, bomo opisali eno orodje za hiter razvoj aplikacij ter z njim izdelali en osnoven primer in s tem prikazali zmogljivosti tega orodja. Orodje, ki ga bomo izbrali za prikaz se imenuje Iron Speed Designer.

Podjetje Iron Speed objavlja orodje za hiter razvoj aplikacij na osnovi podatkovnih baz po imenu Iron Speed Designer. Ta produkt gradi aplikacije na osnovi podatkovnih baz in poročanja (anlg. Database and reporting applications) za spletna okolja (angl. web environments), okolja v oblaku (angl. cloud environments), mobilna okolja (angl. mobile environments) in Microsoftova SharePoint okolja. Web 2.0 aplikacije izdelane z orodjem Iron Speed Designer je mogoče prilagoditi in postaviti brez programiranja [23].

Podjetje je bilo ustanovljeno leta 1999 kot rešitev na problem, s katerim so se soustanovitelji srečevali pri prejšnjem podjetju po imenu Onsale. Razvili so nov produkt, za avtomatizacijo razvoja aplikacije, z namenom, da bi zmanjšali čas razvoja ter stroške [23]. Program teče na operacijskem sistemu Microsoft Windows in gradi .NET aplikacije na podlagi podatkovnih baz Mi-

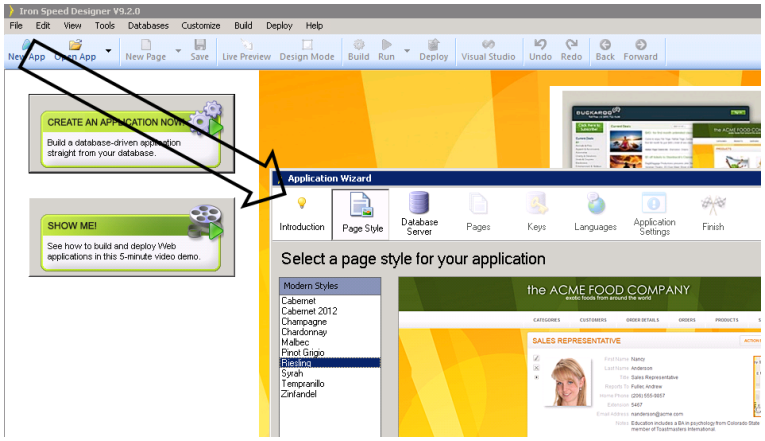
crosoft SQL Server, Oracle, Microsoft Access in MySQL. Izdelane aplikacije so podatkovno usmerjene (angl. database-driven) in tečejo na oblaku, spletu, mobilnih telefonih in SharePoint okoljih [24].

Najprej bomo predstavili osnovne funkcionalnosti tega orodja skozi posnetke zaslona (angl. screen shots), nato bomo s tem orodjem izdelali aplikacijo za oddajanje zahtev po pomoči (angl. help desk application). Aplikacija bo izgledala tako, da se bodo stranke prijavile v aplikacijo, oddale zahtevo po pomoči in bodo nato uporabniki podpore stranki odgovorile z rešitvijo te zahteve. Na koncu pa bomo našteli nekaj prednosti in slabosti tega orodja ter povedali za kakšne aplikacije naj bi se uporabljalo ter za katere vrste aplikacij je to orodje primerno.

4.1 Osnovne funkcionalnosti orodja Iron Speed

4.1.1 Začetni koraki s pomočjo aplikacijskega čarovnika

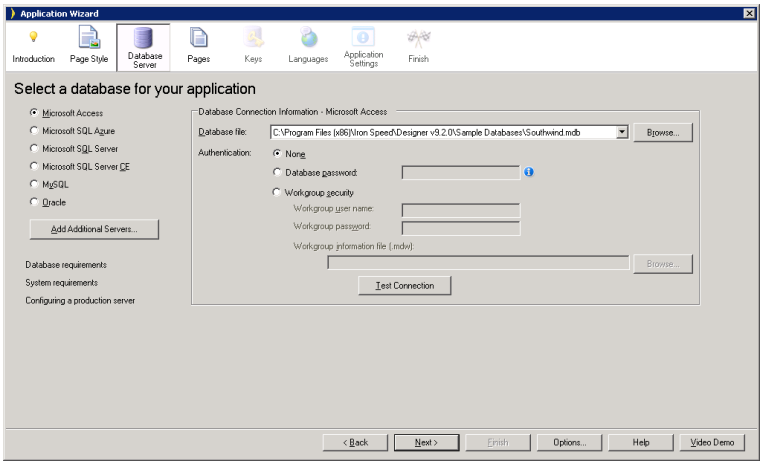
Ko odpremo orodje Iron Speed in pridemo na njegovo začetno stran, začnemo izdelovati spletno aplikacijo ali spletno stran tako, da kliknemo na gumb New App v levem kotu orodja (slika 4.1) in tako odpremo aplikacijski čarovnik (angl. Application Wizard) s pomočjo katerega po korakih izdelamo začetni izgled spletne aplikacije.



Slika 4.1: Odpiranje aplikacijskega čarovnika

Najprej izberemo stil strani (angl. Page Style), ki ga bomo potrebovali za našo aplikacijo. Iron Speed ima možnost izbire modernih ali klasičnih stilov strani za aplikacije. Razlika je v tem, da so moderni lepše in modernejše prikazani kot klasični. Klasični pa imajo imena stolpcev na začetku strani in nato vrstice tabel zapisane spodaj in je njihov izgled kot nekakšna tabela. Za nekatere so mogoče bolj pregledni. Izbira stila strani je odvisna od okusa izdelovalca ali želje tistega, kateremu bo aplikacija namenjena. Za našo predstavitev osnovnih funkcionalnosti izberemo kar privzeti stil, ki ga orodje ponuja in se postavimo na naslednji korak.

Z naslednjim korakom se premaknemo na strežnik podatkovne baze (angl. Database Server), kjer določimo strežnik (slika 4.2), na katerem se nahaja naša baza podatkov, ki jo bomo uporabili za izdelavo naše spletne aplikacije.

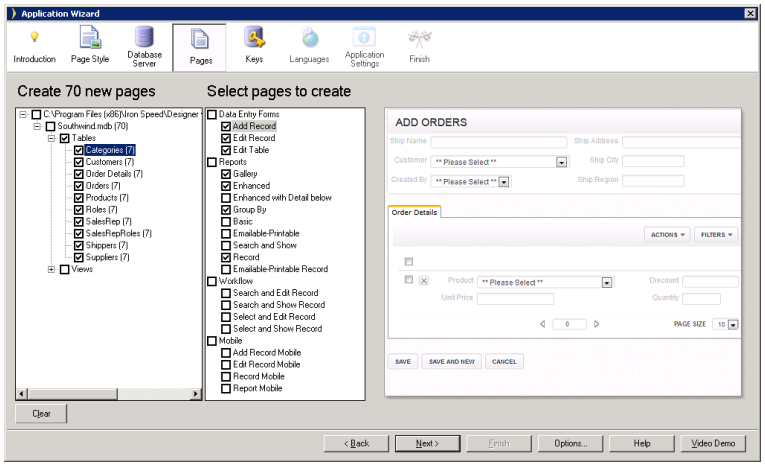


Slika 4.2: Določanje podatkovne baze

Iron Speed že vsebuje vzorec Microsoft Accessove podatkovne baze po imenu Southwind.mdb, ki je namenjena uporabi testiranja tega orodja. Za predstavitev funkcionalnosti bomo izbrali kar ta vzorec in na tej podatkovni bazi prikazali osnovne funkcionalnosti orodja.

Ko izberemo podatkovno bazo se prestavimo na izdelovanje strani naše spletne aplikacije (slika 4.3), kjer nam orodje ponudi podatkovno bazo, ki smo jo izbrali na prejšnjem koraku in ko izberemo podatkovno bazo nam ponudi njene tabele (angl. tables) in poglede (angl. views), ki jih lahko

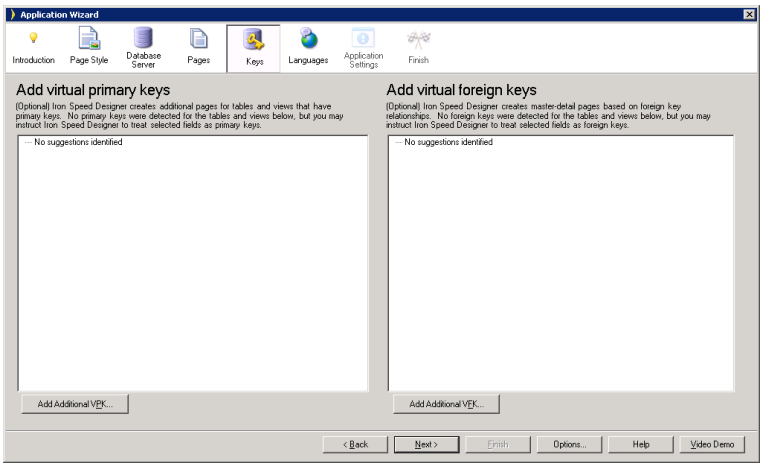
nato izbiramo za izdelavo strani naše spletne aplikacije. Če bi na prejšnjem koraku izbrali strežnik podatkovne baze, na primer Microsoft SQL Server in tam določili strežnik, bi nam orodje ponudilo vse podatkovne baze, ki obstajajo na strežniku.



Slika 4.3: Določanje strani, ki se bodo izdelale

V našem primeru bomo izbrali vse tabele, ki so na voljo v vzorčni podatkovni bazi, ki smo jo izbrali in se nam bodo tako prikazale strani vseh teh tabel. Za vsako tabelo posebej lahko določimo kaj se bo prikazovalo, katere strani nam bo Iron Speed izdelal (na primer dodajanje in urejanje zapisov v tabelo, galerijo slik tabele, iskanje in urejanje zapisov tabele, itd.). V našem primeru pustimo kar privzete vrednosti, ki nam jih orodje ponuja in se postavimo na naslednji korak.

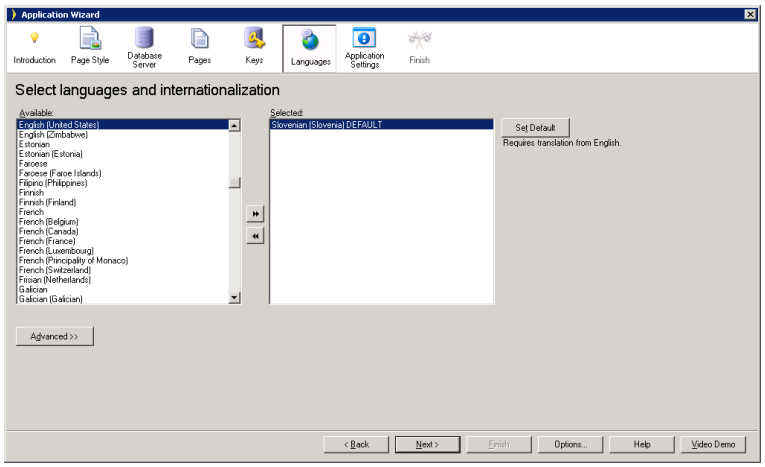
Na naslednjem koraku (slika 4.4) lahko dodajamo virtualne primarne (angl. virtual primary key) in tuje ključe (angl. virtual foreign key), na podlagi katerih Iron Speed izdela še dodatne strani za tiste tabele, ki imajo virtualen ključ.



Slika 4.4: Dodajanje virtualnih ključev

Iron Speed lahko zazna, da nekatere tabele nimajo primarnega ključa ali pa niso povezane s tujimi ključi, vendar bi lahko bile. V takem primeru nam orodje predlaga virtualne ključe. Lahko pa tudi sami dodamo dodatne virtualne primarne ali tuje ključe našim tabelam. Za naš primer prikaza funkcionalnosti pa ne potrebujemo virtualnih ključev, zato se premaknemo na naslednji korak.

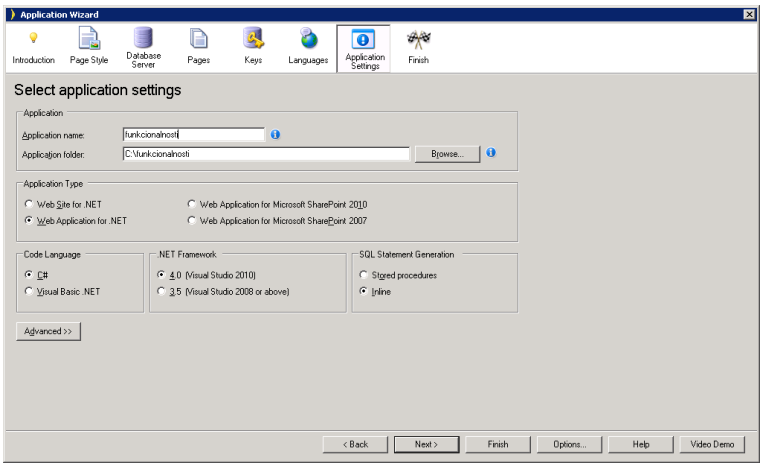
Ko se premaknemo naprej lahko izbiramo jezike naše spletne aplikacije (slika 4.5). Iron Speed ima možnost lokalizacije programa v tuje jezike. Jezike lahko nato izbiramo v naši spletni aplikaciji in z izborom jezika se nam aplikacija prikaže v izbranem jeziku.



Slika 4.5: Izbiranje jezika aplikacije

Orodje nam je predlagalo jezik slovenščina kot privzeti jezik. Pod gumbom Privzeti jezik (angl. Set Default) nam piše da je potreben prevod iz angleščine. Slovenščino je zato potrebno prevesti in aplikacijo lokalizirati v slovenski jezik, ker Iron Speed slovenščine ne vsebuje. Za nekatere jezike pa ima Iron Speed prevode že vgrajene in tistih ni potrebno prevajati, razen če ne želimo imeti drugega izraza. V našem primeru pustimo samo jezik slovenščina in se premaknemo na naslednji korak.

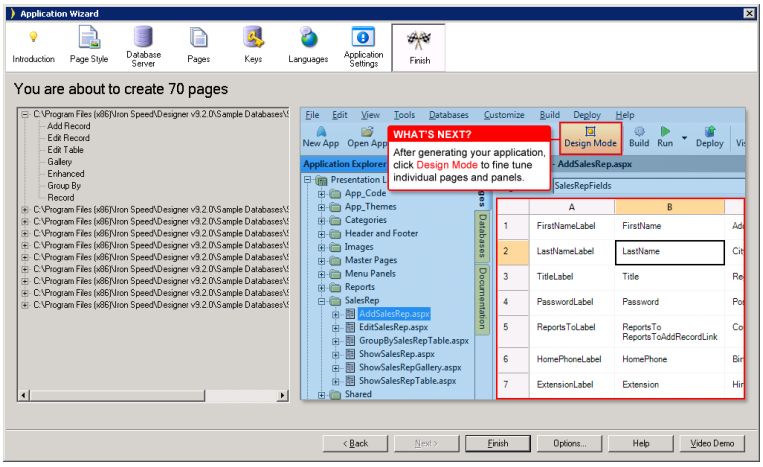
Pri nastavitvah aplikacije (angl. Application Settings) bomo izbrali lokacijo (angl. Application folder) in ime aplikacije (angl. Application name), tip aplikacije (angl. Application Type), programski jezik (angl. Code Language), ogrodje .NET (angl. .Net Framework) in konfiguracijo SQL-a (angl. SQL Statement Generation) na podlagi katerih se bo aplikacija zgradila (slika 4.6).



Slika 4.6: Določanje nastavitev aplikacije

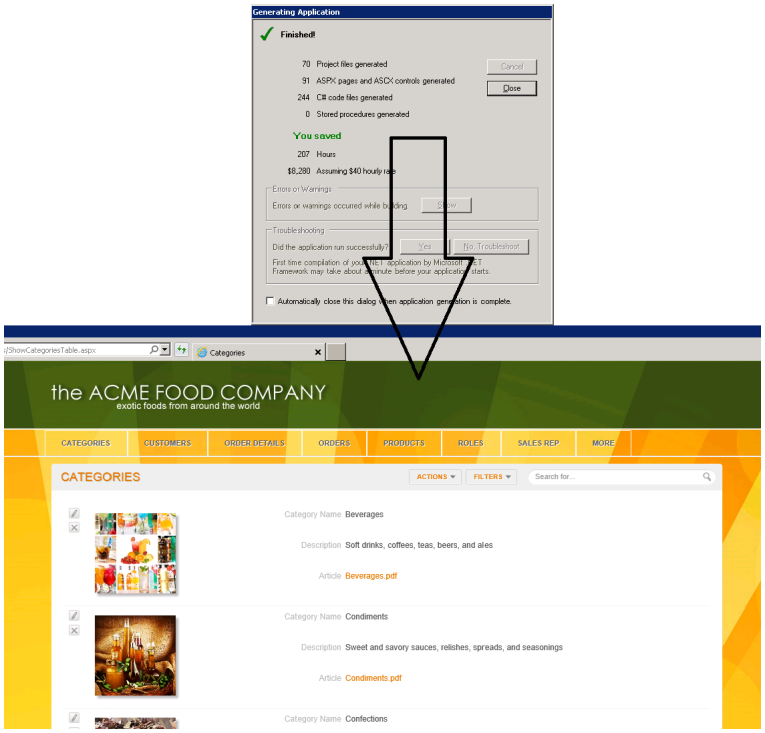
V polje za ime aplikacije bomo vpisali 'funkcionalnosti', Iron Speed nam bo nato poimenoval mapo naše lokacije enako. Tip aplikacije bomo izbrali 'Web application for .NET', ker bomo imeli spletno aplikacijo. Za programski jezik izberemo 'C#', ogrodje '.NET bo 4.0' in pri SQL-u Inline. Nato se postavimo na zadnji korak.

Na končnem koraku (slika 4.7) nam bo samo napisalo koliko strani nam bo izdelalo in kaj naj naredimo za tem, ko nam bo te strani izdelalo za nadaljevanje izdelave spletne aplikacije.



Slika 4.7: Zaključni korak izdelave aplikacije

S klikom na gumb Zaključi (angl. Finish) nam bo program začel izdelovati aplikacijo in bo ob zaključku pognal spletno aplikacijo s privzetim internetnim brskalnikom (slika 4.8).



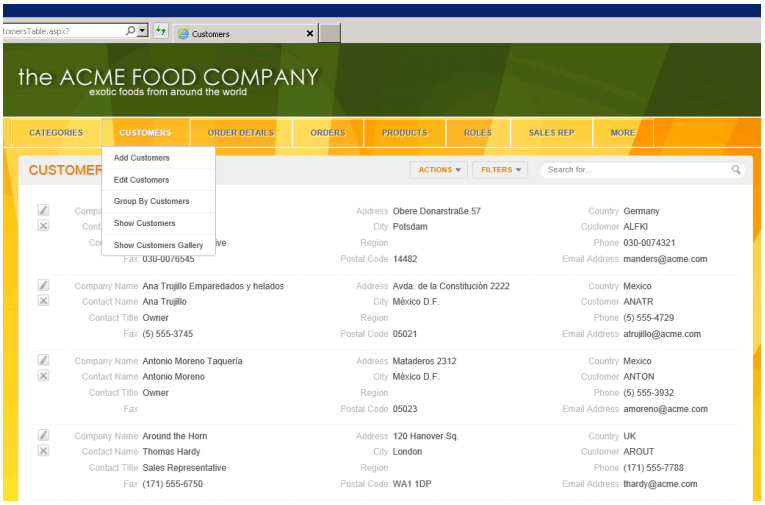
Slika 4.8: Zaključek generiranja aplikacije in pognana aplikacija

Z zadnjim korakom smo zaključili aplikacijski čarovnik in sedaj lahko pogledamo kaj vse nam je orodje izdelalo.

4.1.2 Pregled aplikacije in funkcij, ki jih Iron Speed izdela

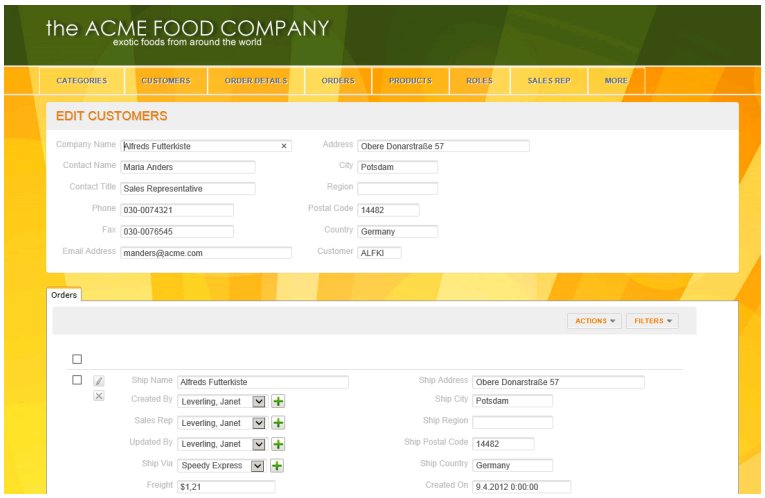
Sedaj ko imamo izdelano aplikacijo, si bomo podrobneje pogledali, kaj nam je Iron Speed zgradil iz podatkovne baze Southwind. Kot lahko vidimo že na zgornji sliki (Slika 4.8), nam je za vsako tabelo, ki smo jo izbrali v aplikacijskem čarovniku izdelal strani za prikaz podatkov, ki so shranjeni v tej tabeli. Imena tabel so prikazana v zgornjem meniju. S kliki na imena tabel nas pre-

usmeri na pogled tiste tabele na katero kliknemo, če se pa z miško postavimo na ime te tabele nam ponudi tudi masovno dodajanje in urejanje podatkov te tabele. Na primer s klikom na gumb CUSTOMERS, nas aplikacija preusmeri na pogled tabele Customers, če se pa z miško postavimo na CUSTOMERS, se nam odpre spustni seznam, kjer lahko izberemo določene akcije (slika 4.9).



Slika 4.9: Spustni seznam tabele Customers

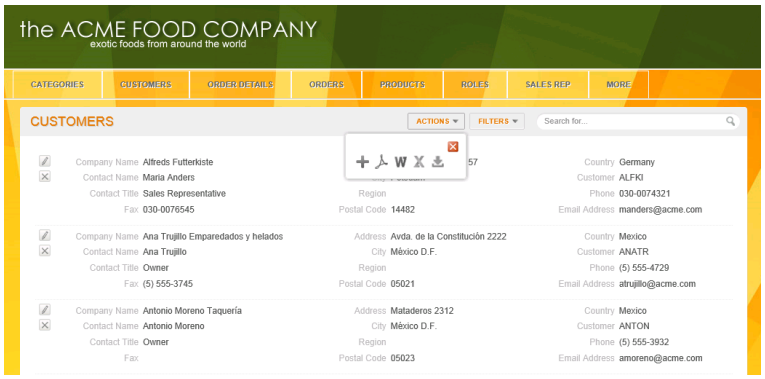
Na zgornji sliki vidimo, kako nam je Iron Speed razporedil podatke vsake tabele in kako nam je razporedil vsako vrstico tabele v vsako vrstico na pogledu. Poleg tega nam je na levi strani, pri vsaki vrstici izdelal dva osnovna gumba. Zgornji gumb je za urejanje vrstice, spodnji gumb je za brisanje vrstice. Če kliknemo na gumb za urejanje, nas preusmeri na urejanje te vrstice tabele (slika 4.10).



Slika 4.10: Urejanje izbranega zapisa

Kot vidimo na zgornji sliki nam ponuja urejanje vseh podatkov v tabeli. Če ima tabela kakšne povezljive tabele (angl. child table) se bodo te pojavile spodaj, pod urejanjem tabele. Na sliki vidimo, da je tabela Customers povezana s tabelo Orders.

Če se vrnemo na pogled tabele, vidimo gumb Dejanja (angl. Actions), kjer se nam odprejo dodatne akcije, če kliknemo na njega (slika 4.11).

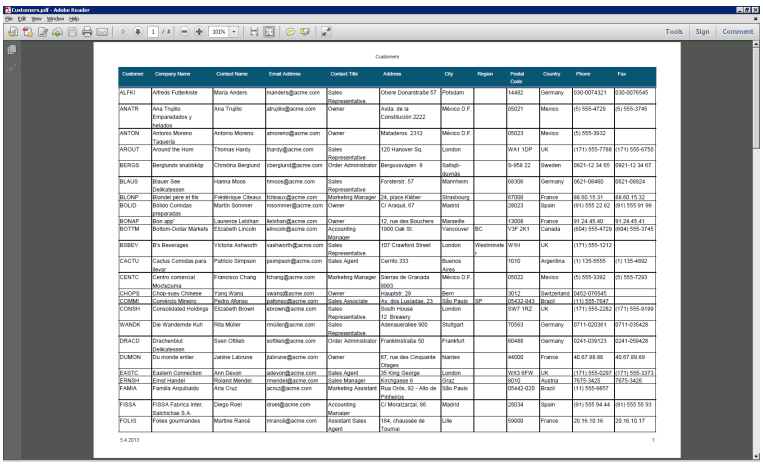


Slika 4.11: Prikaz izbiranja dejanj v spletni aplikaciji

Z leve proti desni si sledijo gumbi takole:

- najprej imamo gumb za dodajanje novega zapisa v tabelo,
- drugi gumb je za izvoz prikaza v PDF dokument (slika 4.12),

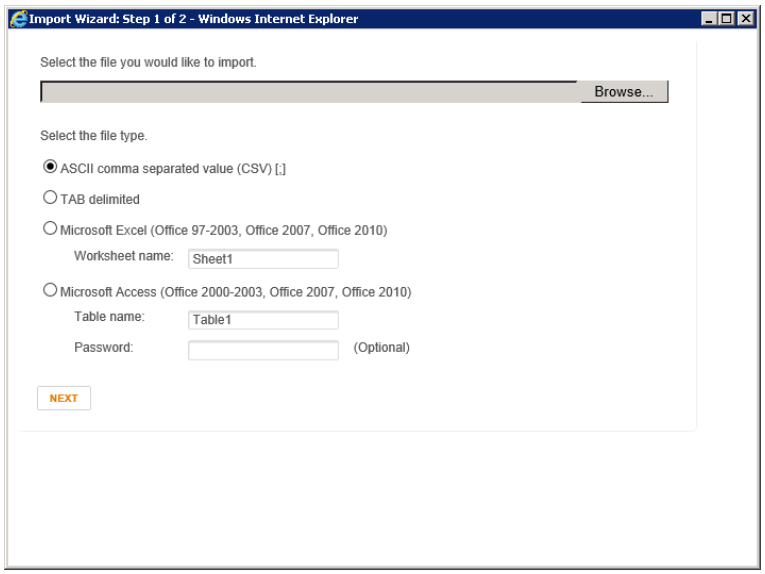
- tretji gumb je za izvoz prikaza v Word dokument,
- četrti gumb je za izvoz prikaza v Excel dokument,
- zadnji gumb se pa uporablja za uvažanje podatkov v aplikacijo.



Slika 4.12: Izvoz preglednice v PDF dokument

Na zgornji sliki so prikazani podatki, ki smo jih izvozili v PDF dokument. V PDF dokument se izvozijo tisti podatki, ki so na pregledu. Če bi na primer filtrirali podatke in na pregledu imeli samo tri vrstice, bi se v PDF dokument izvozile samo te tri vrstice. Enako velja za Word in Excel.

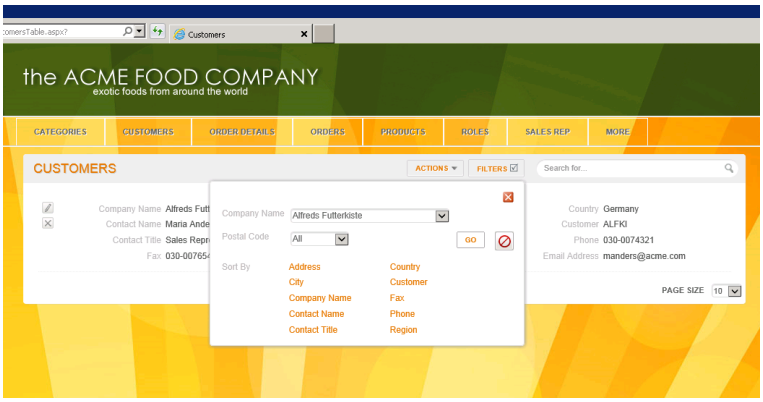
Pri uvozu podatkov pa uvažamo podatke s pomočjo Čarovnika za uvažanje (angl. Import Wizard), ki ga Iron Speed zgradi sam v naši aplikaciji, v katerem izberemo datoteko in jo uvozimo (slika 4.13).



Slika 4.13: Čarovnik za uvažanje zapisov

Tako lahko na primer podatke, ki jih želimo uvoziti zapišemo v Excel in jih nato uvozimo v našo aplikacijo. Iron Speed prebere te podatke, ki jih imamo v Excelu in jih prikaže v drugem koraku čarovnika za uvažanje. Tam lahko izberemo, kaj bomo uvozili in v katere stolpce v tabeli nam bo izbrane podatke uvozilo. Z gumbom Uvozi (angl. Import) nato uvozimo podatke v našo aplikacijo in če ni nobenih napak nam te podatke uvozi v našo aplikacijo in jih nato lahko vidimo na pregledu.

Poleg gumba z dejanji imamo gumb Filtri (angl. Filters), s pomočjo katerega si filtriramo pregled naših podatkov (slika 4.14).

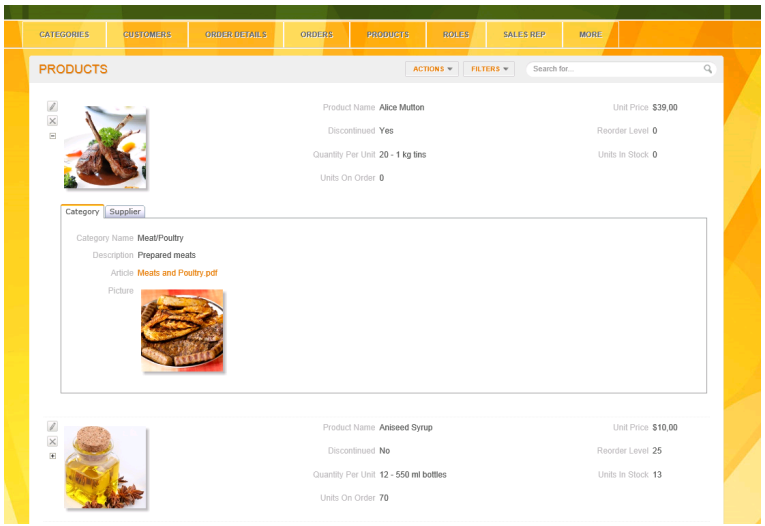


Slika 4.14: Prikaz filtriranja podatkov v aplikaciji

V našem primeru iz spustnega seznama Company Name izberemo ime podjetja 'Alfreds Futterkiste' in kliknemo na gumb Pojdi (angl. Go). S temi akcijami pridemo do našega izbranega prikaza, kot je prikazano na zgornji sliki. Če želimo pobrisati filtre, imamo poleg gumba Pojdi gumb prečrtan krogec, na katerem piše Clear filters, če se z miško pomaknemo nanj in ko kliknemo na ta gumb se nam filtri počistijo in pridemo ponovno na prvotni pregled.

Poleg filtrov imamo tudi polje za iskanje, v katerega vpišemo besedo, ki jo vsebuje nek zapis, ki ga želimo poiskati in tako z vpisom te besede in klikom na gumb za iskanje pridemo do iskanega rezultata. Če pobrišemo iskano besedo in kliknemo nazaj na gumb za iskanje, nam bo vrnilo vse rezultate, ki so v tabeli.

Če imamo v naši podatkovni bazi shranjene slike, nam jih bo Iron Speed prikazal v aplikaciji (slika 4.15). Kot je prikazano na spodnji sliki nam Iron Speed prikaže sliko, na katero lahko tudi kliknemo za povečan pogled slike.



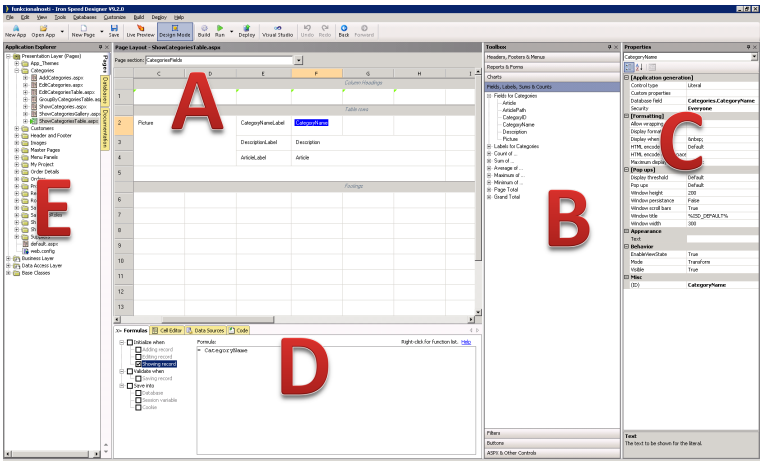
Slika 4.15: Prikaz slik v aplikaciji

Kot vidimo ima prva vrstica naše tabele še povezavo na drugo tabelo, zato nam pod to vrstico prikaže še kategorijo produkta, ki je prikazan.

To kar smo prikazali so funkcije, ki nam jih orodje Iron Speed že sam na podlagi podatkovne baze generira. Če si želimo karkoli preurediti in si nastaviti po svoje, lahko to Naredimo v načinu za oblikovanje (angl. Design Mode).

4.1.3 Prilagoditev aplikacije v Načinu za oblikovanje (angl. Design Mode)

Ko nam orodje na podlagi podatkovne baze zgradi aplikacijo si jo lahko nato preuredimo v Načinu za oblikovanje(slika 4.16). V Načinu za oblikovanje si lahko preuredimo ali dodajamo nove elemente in določimo razne druge ali preuredimo obstoječe funkcionalnosti.

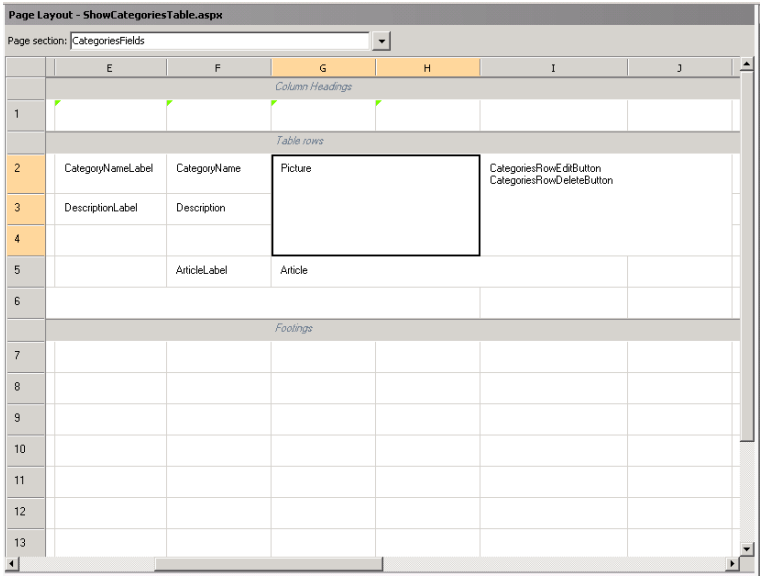


Slika 4.16: Način za oblikovanje v orodju Iron Speed

Če si pogledamo zgornjo sliko, vidimo da je razdeljena na več območji. Tako izgleda orodje Iron Speed v Načinu za oblikovanje. Opisali pa bomo še vsako območje posebej in povedali čemu služi.

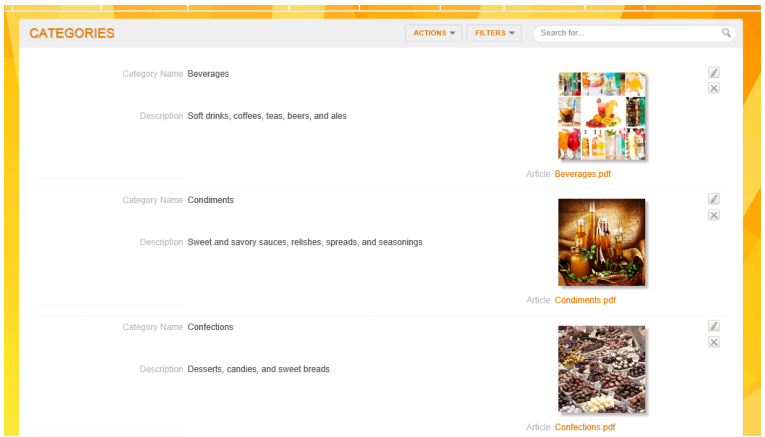
Če si pogledamo najprej območje A (slika 4.16) – Postavitev strani (angl. Page Layout), vidimo elemente postavljene v poljih tabele podobno kot v programu Excel. Tako kot so elementi tam postavljeni, se potem prikažejo na strani aplikacije. Trenutno je na sliki prikaz podatkovne tabele Categories, kar tudi vidimo v spustnem seznamu Dela strani (angl. Page section), nam piše CategoriesFields in na levem območju E – Raziskovalec aplikacije (angl. Application Explorer) imamo izbrano ShowCategoriesTable.aspx, kar pomeni, da v Načinu za oblikovanje urejamo pregled strani za tabelo Categories.

Prikazali bomo kako si lahko v Načinu za oblikovanje preuredimo prikaz elementov tabele Categories na pregledu kategorij v naši spletni aplikaciji (slika 4.17). Recimo, da si želimo imeti na levi strani ime in opis (CategoryName in Description), nato desno od imena in opisa sliko (Picture), pod sliko članek (Article) in desno od vsega kar preuredimo pa gumbe za urejanje in brisanje (CategoriesRowEditButton in CategoriesRowDeleteButton).



Slika 4.17: Preurejeni elementi v Postavitvi strani

Na zgornji sliki vidimo, kako smo premaknili elemente in jih postavili drugače, kot nam jih je Iron Speed sam postavil. Z gumbom Zaženi (angl. Run) v zgornjem meniju Iron Speed-a poženemo aplikacijo in pogledamo, kaj je nastalo, ko smo si preuredili prikaz v Načinu za oblikovanje (slika 4.18).

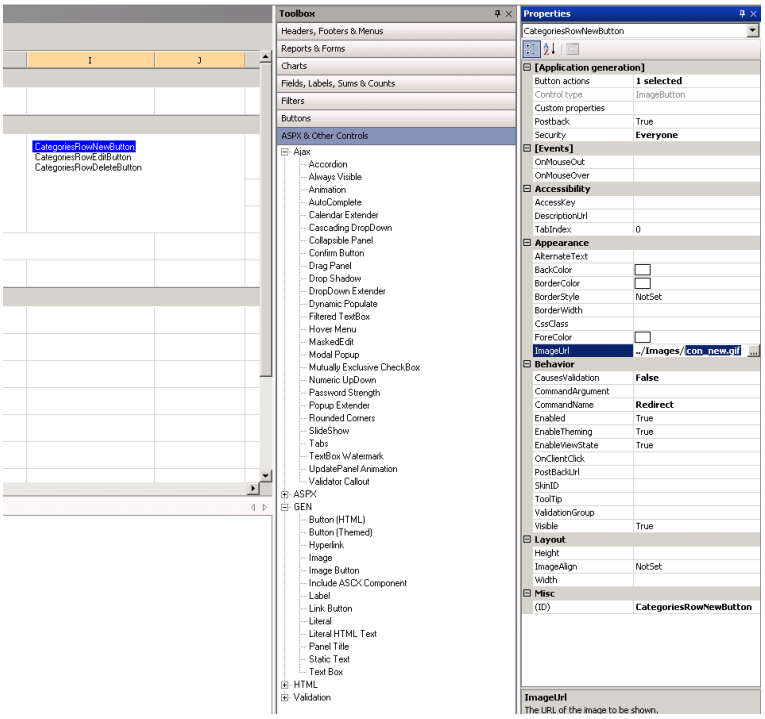


Slika 4.18: Preurejeni elementi v aplikacijii

Ko se nam aplikacija zgradi in požene v spletnem brskalniku vidimo, da imamo sedaj ime kategorije in opis na levi, sliko na desni, članek pod sliko ter gumbe za urejanje in brisanje desno od slike.

Nato si bomo pogledali območje B (slika 4.16) – Seznam orodji (angl. Toolbox). Tukaj imamo razna orodja, katera lahko prenesemo v Postavitev strani. Lahko prenašamo oznake polj (angl. labels), lahko prenašamo vrednosti polj, lahko prenašamo gumbe, filtre, ASPX kontrole, grafe, itd. Desno od seznama orodji je območje C (slika 4.16) – Lastnosti (angl. Properties), kjer lahko urejamo lastnosti elementov v Postavitvi strani. Za vsak element, ki ga imamo označenega se nam prikažejo njegove lastnosti, ki jih lahko pustimo takšne kot so ali pa spremenimo, če to potrebujemo.

Na primer da bi poleg gumba za urejanje in brisanje vrstice želeli imeti še gumb za dodajanje nove vrstice (slika 4.19). S tem bomo prikazali funkcionalnost Orodji in Lastnosti.

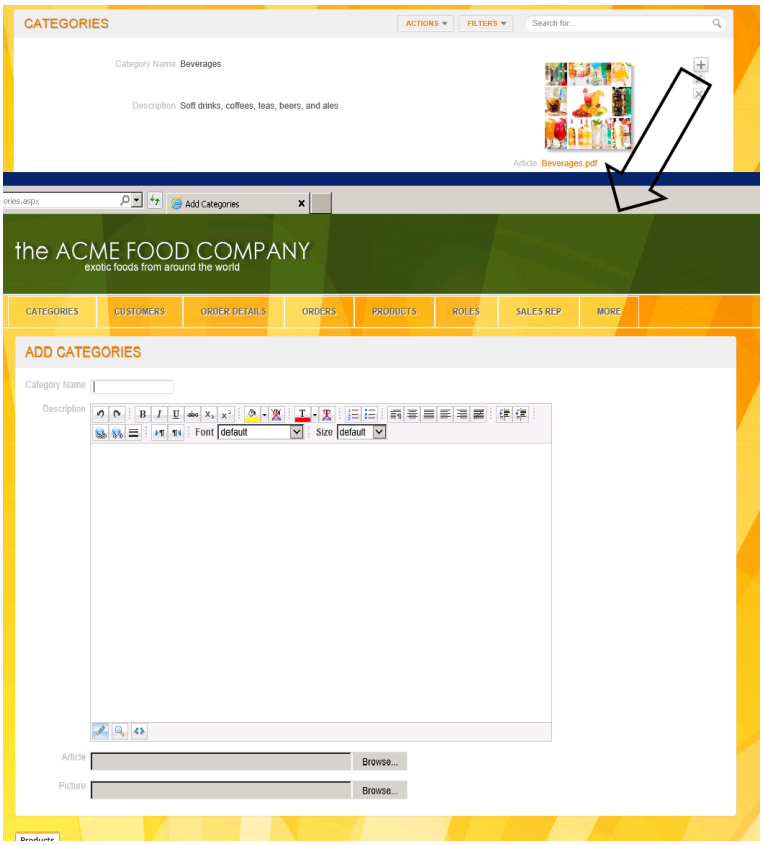


Slika 4.19: Dodajanje gumba za dodajanje nove vrstice

Najprej iz seznama orodji in zavihka ASPX in ostale kontrole (angl. ASPX & Other Controls) razširimo kontrolo GEN in iz tam povlečemo Slikovni gumb (angl. Image Button) ter ga postavimo nad CategoriesRowEditButton, ker

želimo, da je gumb za dodajanje nove vrstice nad gumbom za urejanje vrstice. Nato ga preimenujemo v CategoriesRowNewButton, da bomo imeli gumbe enako poimenovane. Za tem kliknemo na Akcije gumba (angl. Button actions) v območju Lastnosti in izberemo Preusmeri (angl. Redirect) in nato pri akcijah kliknemo na gumb Uredi... (angl. Edit...), da uredimo preusmeritev ob kliku na ustvarjen gumb. Nato pridemo na zavihek Akcije (angl. Actions), kjer izberemo 'Pojdi na specifičen URL' (angl. Go to a specific URL) in se pomaknemo na naslednji korak na zavihek Preusmeritev URL-ja (angl. Redirect URL). Na tem zavihku se preusmerimo na URL naslov za dodajanje nove kategorije tako, da izberemo specifičen URL, ki je '.../Categories/AddCategories.aspx' in kliknemo na gumb Zaključi (angl. Finish). Ko zaključimo kliknemo na gumb OK. Nato nastavimo še lastnost Postback na True, sicer nas ne bo preusmerilo na stran za dodajanje kategorij. Za lažje razumevanje čemu je ta gumb namenjen v aplikaciji, mu izberemo še primerno sličico. Iron Speed ima na izbiro že nekaj svojih sličic, za najpogostejše gumbe. Pri lastnosti UrlSlike (angl. ImageUrl) zato izberemo 'icon_new.gif' kot sličico za prikaz gumba, ker je ta sličica namenjena gumbu za dodajanje novega zapisa.

Ko poženemo aplikacijo in nam jo Iron Speed zgradi, vidimo nad gumbom za urejanje gumb za dodajanje nove kategorije in ko kliknemo nanj se nam odpre nova stran, katera je namenjena za dodajanje nove kategorije v tabelo (slika 4.20).



Slika 4.20: Prikaz funkcionalnosti gumba za dodajanje nove vrstice

Nato se vrnemo ponovno na Način za oblikovanje. Pod območjem A imamo tudi območje D (slika 4.16), ki je del območja A in se prikaže samo če imamo Postavitev strani. Če v Raziskovalcu aplikacije kliknemo na kaj drugega, na primer na kakšno sliko v mapi Images ali pa če kliknemo na datoteko s končnico '.aspx.cs' (katera nam prikaže kodo te strani), se nam to območje ne prikaže. Območje je sestavljeno iz štirih zavihkov:

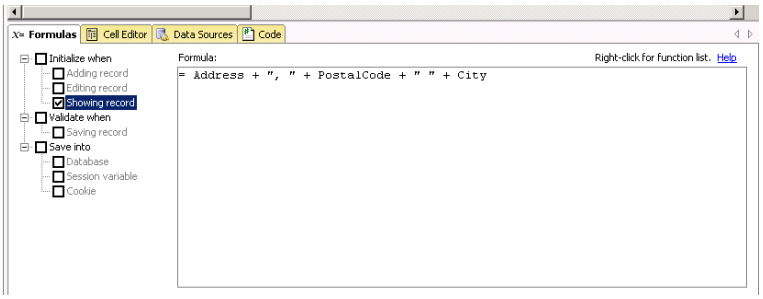
- Prvi zavihek se imenuje Formule (angl. Formulas) in je zavihek v katerem zapišemo formule za določene elemente v tabeli. Na primer kaj se bo prikazalo za prvotno vrednost, ko prvič pridemo na določeno stran. Recimo ko pridemo na dodajanje novega zapisa lahko inicializiramo vrednost spustnega seznama. Lahko tudi zapišemo razne validacije in opozorila ob shranjevanju nekega zapisa.

- Drugi zavihek je Urejevalnik polja (angl. Cell Editor), v katerem lahko spreminjamo izgled našega polja, ki se bo prikazalo na strani. Spreminjamo jih z jezikom HTML, imamo pa na voljo tudi nekaj funkcij, za spreminjanje barve, lege in ostalega urejanja besedila podobno kot v programu Word ali v ostalih urejevalnikih besedila.

- V tretjem zavihku, Izvor podatkov (angl. Data Sources), lahko pišemo SQL stavke in napišemo kaj se bo prikazalo v določenem polju ali spustnem seznamu.

- Zadnji zavihek je zavihek Kodiranje (angl. Code), v katerem lahko spreminjamo funkcije elementov s programskim jezikom, v našem primeru je to C#, ker smo ga ob generiranju aplikacije izbrali in ko se aplikacija generira, se piše v tem jeziku. Tako da če želimo še kaj preurediti lahko to naredimo s programskim jezikom C# v zavihku Kodiranje.

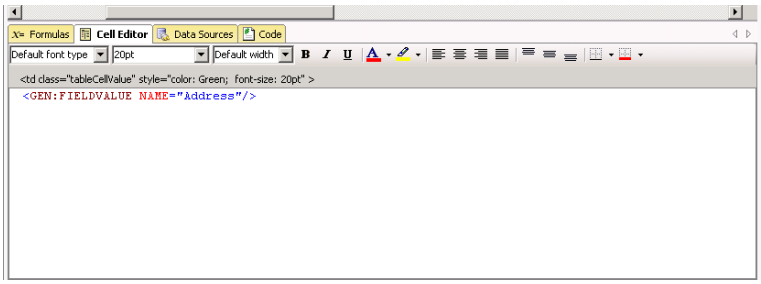
Na primer, da želimo imeti v pogledu tabele Customers v naslovu še poštno številko in mesto, se postavimo v Načinu za oblikovanje na polje Address in v območju D pod zavihkom Formule kliknemo na Inicializiraj ko – prikazuješ zapis (angl. Initialize when – Showing record) in v urejevalniku za formule kjer imamo = Address dodamo še PostalCode in City (slika 4.21). To lahko naredimo tako, da kar zapišemo imena polj ali pa z desnim klikom kliknemo na urejevalnik, nato izberemo DB Record in nato izberemo PostalCode in City.



Slika 4.21: Dodajanje poštne številke in mesta k naslovu v Formulah

V Načinu za oblikovanje nato zbrišemo polja CityLabel, City, PostalCodeLabel in PostalCode, ker bomo te vrednosti prikazali kar v polju Address.

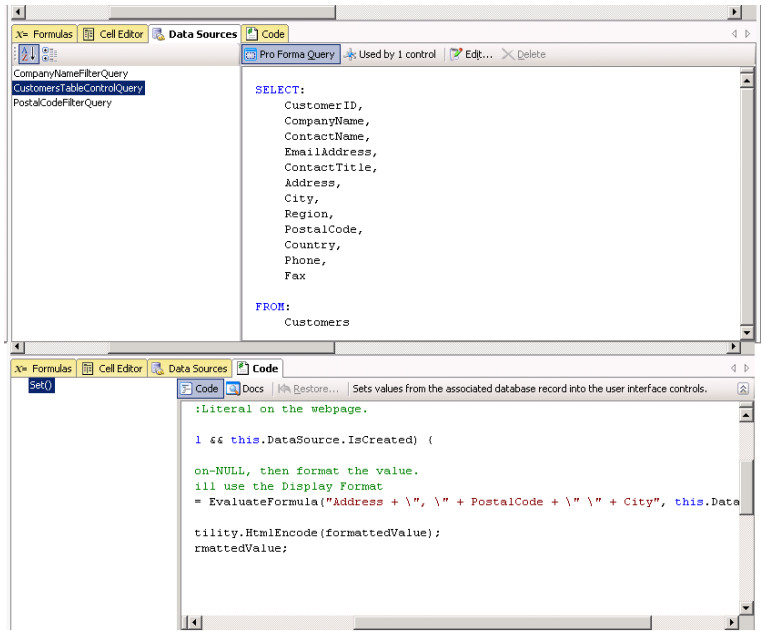
Recimo da bi želeli povečati še velikost pisave in barvo naslova. Pomaknemo se v zavihek Urejevalnik polja in tam izberemo velikost pisave 20 in barvo pisave zeleno (slika 4.22). Velikost pisave spremenimo v spustnem seznamu Default font size in barvo spremenimo tako, da kliknemo na spremembo barve – velika črka A z rdečim vzorcem barve spodaj in izberemo zeleno barvo.



Slika 4.22: Spreminjanje oblike polja v Urejevalniku polja

Na zgornji sliki lahko vidimo, da so se v našem stolpcu td dodali tudi atributi tako, da kadar spremenimo velikost pisave, se doda atribut 'font-size: 20pt' in ko spremenimo barvo se doda atribut 'color: Green'.

V zavihku Izvor podatkov imamo poizvedbe za filtre ter poizvedbo ki prebere vse vrednosti naše tabele Customers (slika 4.23 – zavihek Izvor podatkov). Tukaj ne bomo nič spreminjali. Če se premaknemo na naslednji zavihek Kodiranje, vidimo da imamo metodo 'SetAddress()', katera nam nastavi vrednost polja Address v programskem jeziku C#. Vidimo lahko kako nam je na podlagi formule zgradilo C# kodo (slika 4.23 – zavihek Kodiranje).



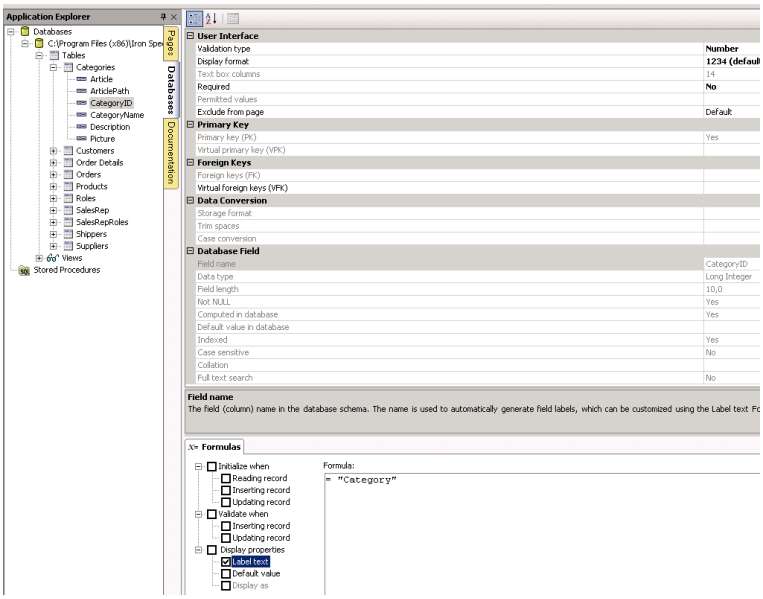
Slika 4.23: Prikaz zavihka Izvor podatkov in zavihka Kodiranje

Nato še poženemo aplikacijo, da vidimo rezultat (slika 4.24) in kot je prikazano na spodnji sliki imamo v polju Address poleg naslova še poštno številko in mesto, polje ima povečano velikost pisave ter barvo pisave spremenjeno v zeleno.

Slika 4.24: Prikaz spremenjenega polja Address v aplikaciji

Za konec nam ostane še območje E (slika 4.16) – Raziskovalec aplikacije, ki je sestavljen iz treh zavihkov, to so Strani (angl. Pages), Podatkovne baze

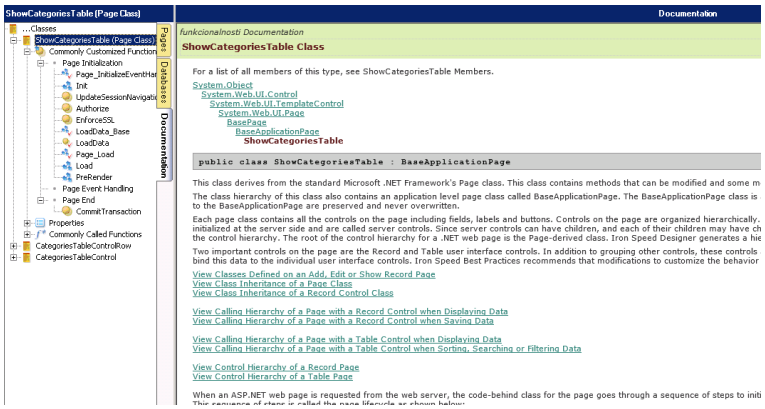
(angl. Databases) in Dokumentacija (angl. Documentation). V prvem zavihku imamo na voljo drevesno strukturo map iz katerih je sestavljena naša spletna aplikacija. Če na kateri element, ki je hranjen v teh mapah kliknemo, se nam bo na območju A odprl urejevalnik tega elementa. Drugi zavihek obsega podatkovno bazo, ki jo uporabljamo za izdelavo naše spletne aplikacije (slika 4.25).



Slika 4.25: Lastnosti stolpca CategoryID v zavihku Podatkovne baze

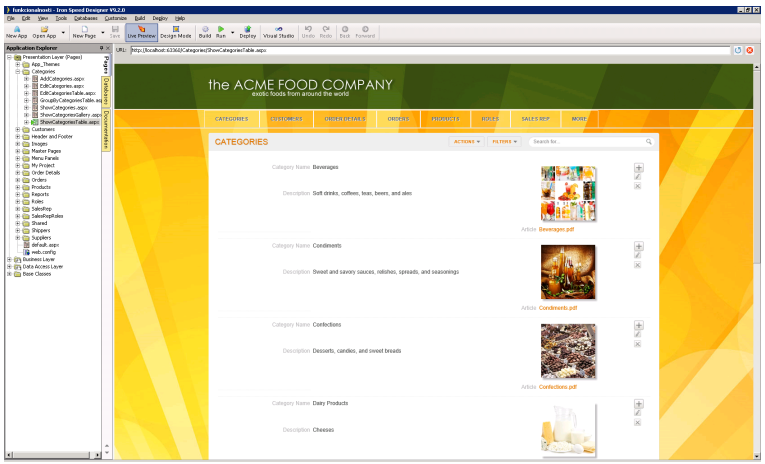
Če kliknemo na kateri stolpec tabele, se nam odprejo lastnosti tega stolpca, kjer jih lahko določamo ter ustvarjamo tudi virtualne primarne in tuje ključe. Spodaj imamo na voljo urejevalnik Formul, v katerem lahko zapišemo formule za inicializacijo, validacijo in prikaz stolpca, ki ga imamo označenega.

V zadnjem zavihku je zapisana dokumentacija tega orodja (slika 4.26).



Slika 4.26: Prikaz dokumentacije v orodju Iron Speed

Orodje Iron Speed omogoča tudi pregled aplikacije brez da jo poženemo v brskalniku ampak kar znotraj orodja, tako da kliknemo na gumb Predogled v živo (angl. Live Preview). To funkcionalnost lahko uporabimo za hitrejši pregled aplikacije, da nam ni ves čas potrebno poganjati aplikacije v spletnem brskalniku (slika 4.27).

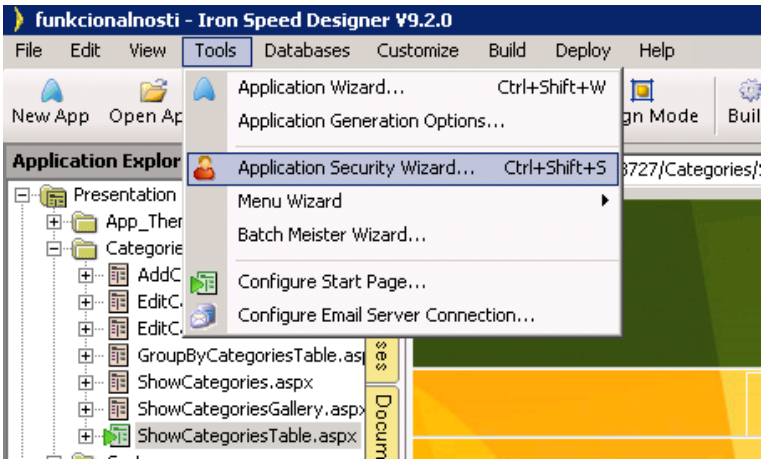


Slika 4.27: Predogled v živo v orodju Iron Speed

4.1.4 Izdelava varnosti z orodjem Iron Speed

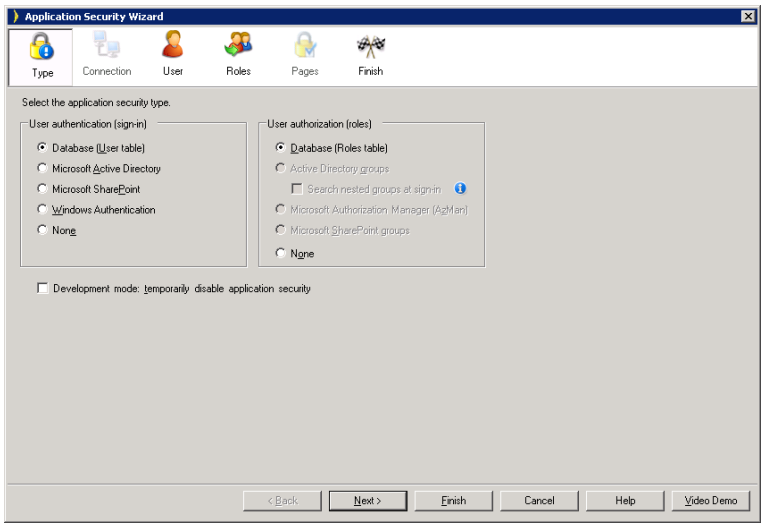
V orodju Iron Speed imamo možnost izdelave prijave uporabnikov in njihovih pravic o tem kaj bodo lahko videli, kaj bodo lahko urejali, kaj bodo lahko brisali, itd. Orodje Iron Speed vsebuje čarovnik, s katerim si pomagamo iz-

delati varnost naše spletne aplikacije. Ta čarovnik se imenuje Čarovnik za varnost aplikacije (angl. Application Security Wizzard), ki ga najdemo v zgornjem meniju aplikacije (slika 4.28) v seznamu Orodja (angl. Tools).



Slika 4.28: Pot do čarovnika za varnost aplikacije

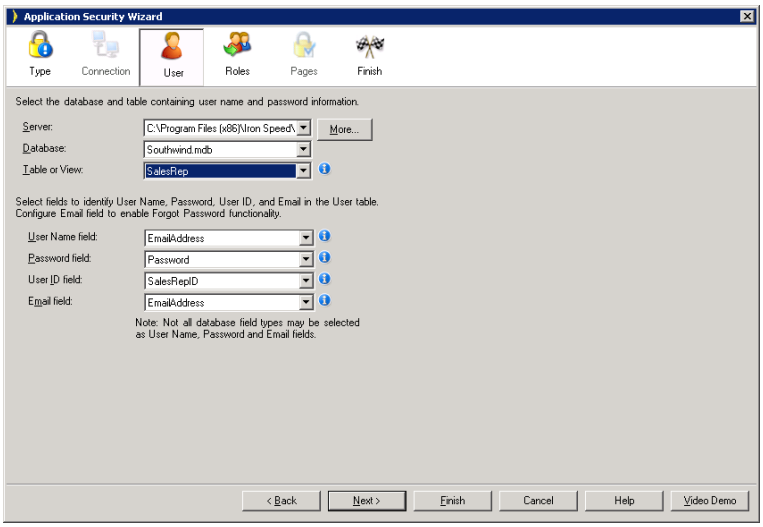
Ko kliknemo nanj, se nam odpre čarovnik za varnost aplikacije. Z njegovo pomočjo lahko nato po korakih izdelamo prijavo uporabnikov ter določanje pravic uporabnikov v aplikaciji. Prvi korak je določitev tipa varnosti (angl. Type) naše spletne aplikacije (slika 4.29). Na prvem koraku imamo več možnosti avtentikacije in avtorizacije, ena izmed možnosti je avtentikacija z bazo podatkov, tako da uporabimo tabele v naši podatkovni bazi, tabelo uporabnika in tabelo vlog. Če uporabljamo Aktivni direktorij (angl. Active Directory), se lahko prijavimo z njim v aplikacijo in v tem primeru izberemo avtentikacijo z Microsoft Active Directoryjem. Lahko uporabimo tudi skupine v SharePointu, za uporabo pravic v aplikaciji, tako da izberemo avtentikacijo z Microsoft SharePointom. Če bi se pa recimo želeli prijaviti v aplikacijo kar z Windows računom, izberemo Windows Autenhication avtentikacijo in tako se prijavimo v našo spletno aplikacijo kar z Windowsovim računom.



Slika 4.29: Izbiranje tipa varnosti aplikacije

V našem primeru izberemo avtentikacijo z bazo podatkov in avtorizacijo z bazo podatkov ter se pomaknemo na naslednji korak.

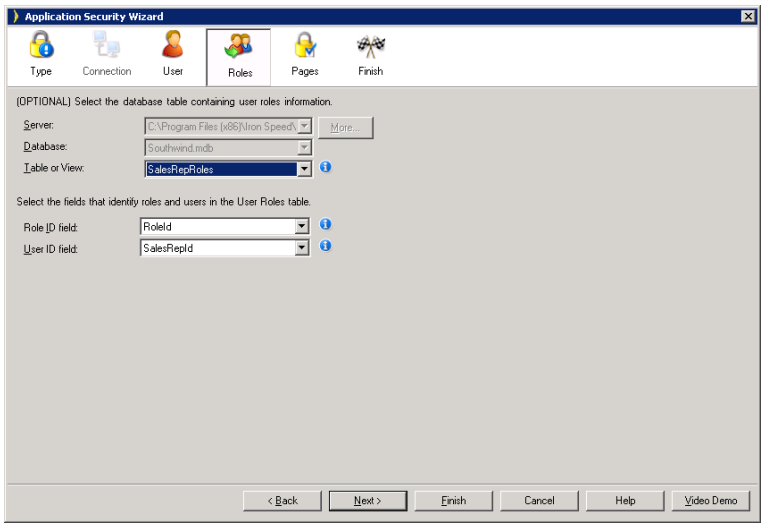
Ko se pomaknemo na zavihek Uporabnik (angl. User) lahko izberemo tabelo v kateri imamo shranjene uporabnike, ki se bodo vpisovali v našo aplikacijo (slika 4.30). Nato izberemo polja za določitev uporabniškega imena, gesla, ID-ja uporabnika ter e-poštnega (angl. User Name, Password, User ID, Email) naslova.



Slika 4.30: Izbiranje tabele, ki hrani uporabnike ter njenih polj

Za uporabniško ime izberemo 'EmailAddress', ker se bomo z e-poštnim naslovom prijavili v aplikacijo, za geslo izberemo polje 'Password', za ID uporabnika izberemo 'SalesRepID' ter za e-poštni naslov ponovno izberemo 'EmailAddress'.

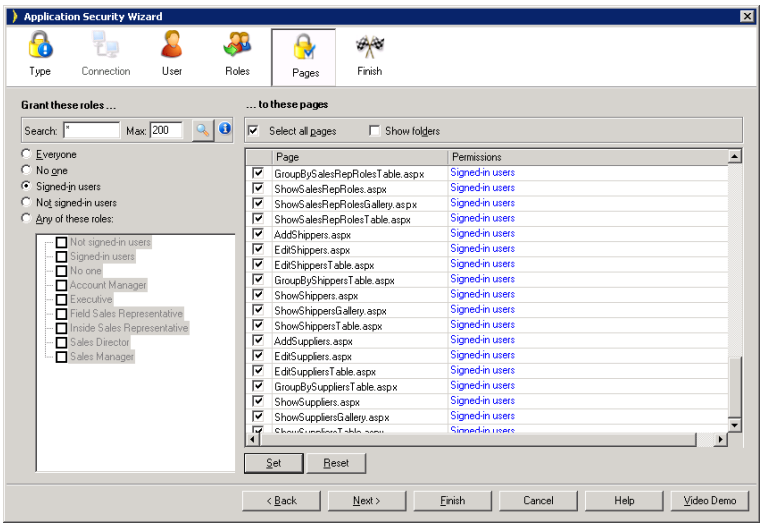
V naslednjem koraku Vloge (angl. Roles) izberemo tabelo, ki hrani ID uporabnika in ID vloge (slika 4.31). Izbrana tabela se imenuje SalesRepRoles. Ta tabela povezuje tabeli SalesRep in Roles.



Slika 4.31: Izbiranje tabele, ki povezuje uporabnike in njihove vloge

Nato izberemo polja, ki identificirajo vloge in uporabnike, tako da v spustnem seznamu Role ID field izberemo 'RoleId' in v spustnem seznamu User ID field izberemo 'SalesRepID' ter se pomaknemo na naslednji korak.

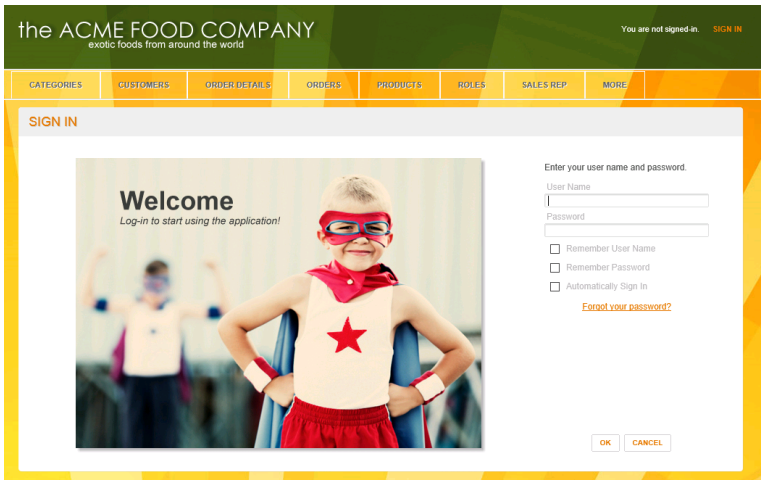
Ko pridemo na zavihek Strani (angl. Pages) začnemo določati pravice uporabnikov, ki uporabljajo našo spletno aplikacijo (slika 4.32). Na levi strani izberemo, kdo bo imel pravico, na desni strani pa s klikom na določeno stran in s klikom na gumb Nastavi (angl. Set) določimo kdo bo te pravice videl. Izbiramo lahko med Vsakdo (angl. Everyone), s tem bodo lahko aplikacijo videli vsi uporabniki. Če izberemo Nihče (angl. No one), noben uporabnik ne bo mogel videti naše aplikacije. Z izbiro Samo prijavljeni uporabniki (angl. Signed-in users) določimo aplikacijo tako, da bodo vsebino videli samo uporabniki, ki so vpisani v aplikacijo. Če pa izberemo Neprijavljeni uporabniki (angl. Not signed-in users) bodo pa vsebino videli uporabniki, ki niso vpisani v aplikacijo. Če pa želimo, da določene strani vidijo tisti uporabniki, kateri imajo določeno vlogo pa kliknemo na Vsaka izmed teh vlog (angl. Any of these roles) in izberemo vlogo ter jo nastavimo.



Slika 4.32: Določanje dostopa do strani aplikacije

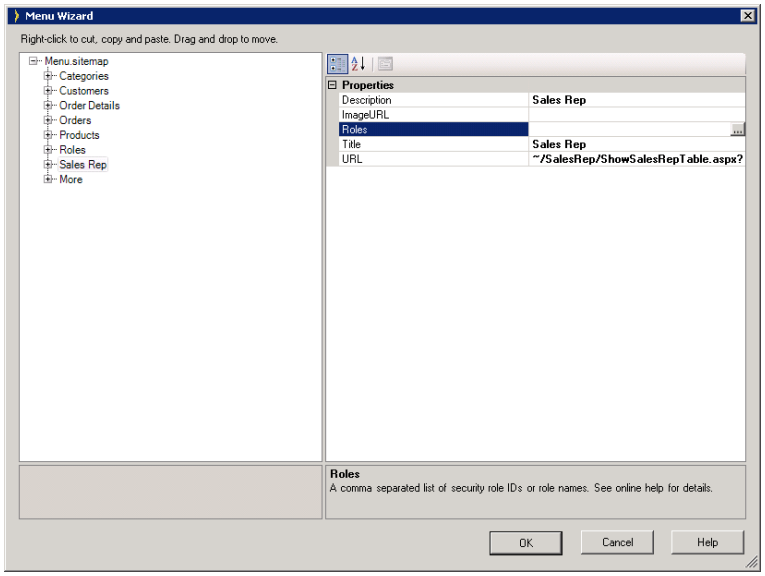
Mi bomo našo aplikacijo zastavili tako, da bodo njeno vsebino videli le vpisani uporabniki naše aplikacije. Izberemo Samo prijavljeni uporabniki in nato na desni strani označimo Izberi vse strani (angl. Select all pages), da nam bo izbralo vse strani in z klikom na gumb Nastavi določimo našo izbiro vsem stranem naše aplikacije. Na koncu kliknemo na tipko Zaključi (angl. Finish) in zaključimo s čarovnikom za varnost.

Ko zaženemo našo aplikacijo, nam na začetni strani ponudi vpis v spletno aplikacijo (slika 4.33). Tako se morajo sedaj uporabniki spletne aplikacije vpisati, če želijo uporabljati spletno aplikacijo in gledati njeno vsebino.



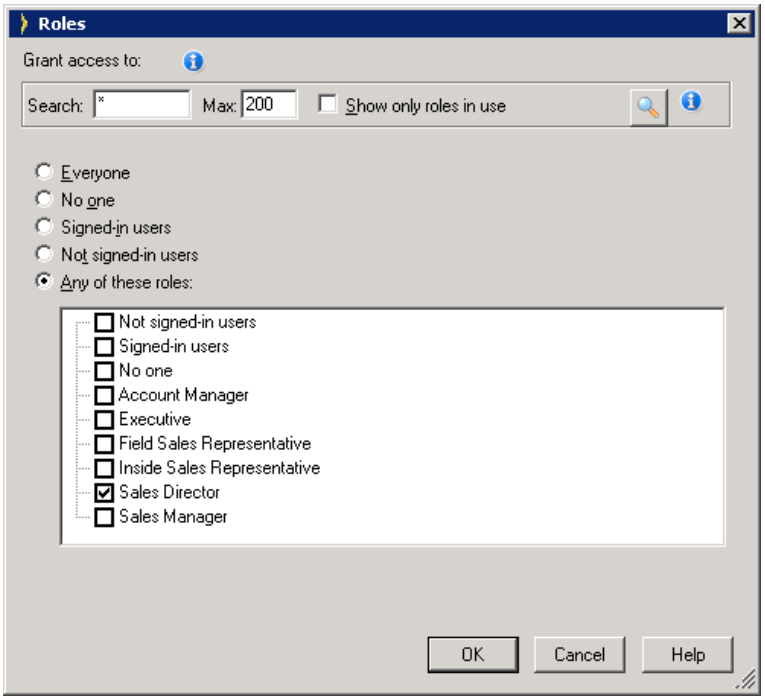
Slika 4.33: Začetna stran za prijavo v spletno aplikacijo

Če bi recimo nato želeli, da bi samo uporabniki, ki imajo vlogo 'Sales Director' imeli dostop do tabel Roles, SalesRep in SalesRepRoles, se pomaknemo na Raziskovalcu aplikacije na mapo Menu Panels in tam z desnim klikom kliknemo na Menu.ascx in izberemo Konfiguriraj (angl. Configure). S temi akcijami pridemo do Čarovnika za menije (angl. Menu Wizard) v katerem lahko določamo pravice našim tabelam oziroma stranem naše aplikacije (slika 4.34).



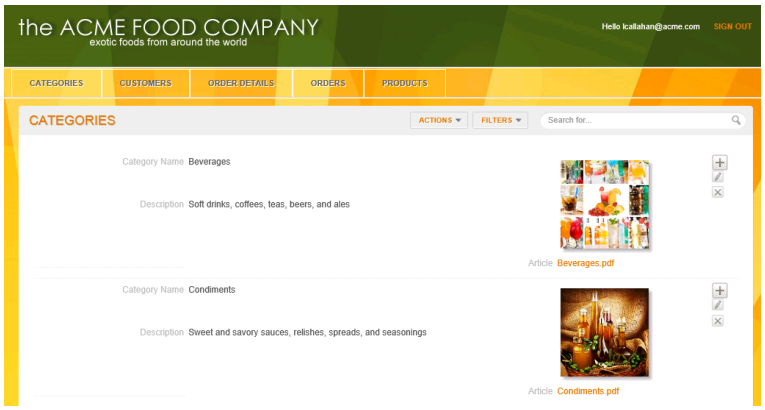
Slika 4.34: Določanje pravic v Čarovniku za menije

Pravice določimo tako, da kliknemo na stran, v našem primeru kliknemo na Roles, Sales Rep in More ter v Lastnostih kliknemo na Vloge (angl. Roles) in izberemo Katerekoli izmed teh vlog (angl. Any of these roles) in nato označimo 'Sales Director' (slika 4.35). Zavihek More smo izbrali, ker se pod tem zavihkom nahaja zavihek SalesRepRoles, ki ga je Iron Speed dodal v spustni seznam zavihka More.



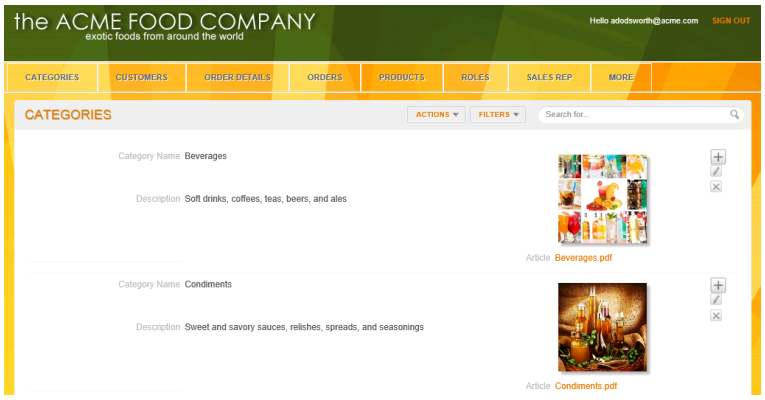
Slika 4.35: Izbiranje vloge 'Sales Director'

Nato kliknemo na gumb OK in zaženemo našo aplikacijo, da vidimo kaj se zgodi, če se prijavimo z uporabnikom, ki ima vlogo 'Sales Director' in z uporabnikom, ki nima vloge 'Sales Director'. Najprej se prijavimo z uporabnikom Callahan Laura, ki ima vlogo 'Field Sales Representative' (sllika 4.36). Ko se prijavimo, vidimo da nam manjkajo strani Roles, Sales Rep in More. Te strani nam manjkajo, ker imajo dostop do teh strani samo uporabniki z vlogo 'Sales Director'.



Slika 4.36: Prijava v aplikacijo z uporabnikom Callahan Laura

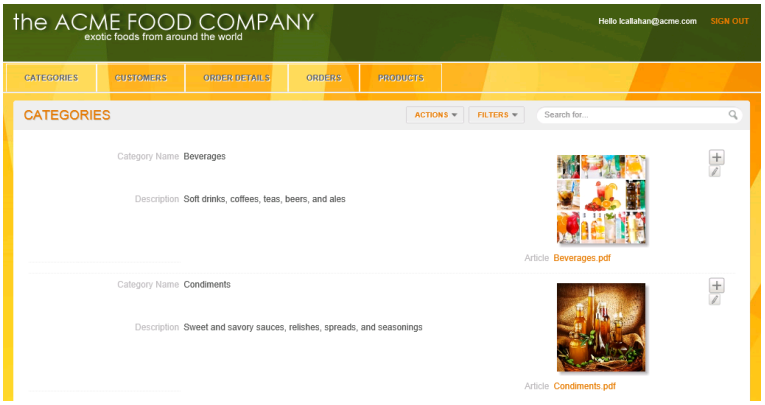
Nato se odjavimo in ponovno prijavimo v aplikacijo z uporabnikom Dodsworth Anne, ki ima vlogo 'Sales Director' (slika 4.37). Ko se prijavimo, lahko vidimo, da imamo sedaj dostop tudi do strani Roles, Sales Rep in More, ker smo se prijavili z uporabnikom, ki ima vlogo 'Sales Director'.



Slika 4.37: Prijava z uporabnikom Dodsworth Anne

Ko opravimo s tem želimo še, da imajo samo uporabniki, z vlogo 'Sales Director' pravico brisanja kategorij na preglednici. To naredimo tako, da se premaknemo na ShowCategoriesTable.aspx v Raziskovalcu aplikacije, nato v Postavitvi strani izberemo CategoriesRowDeleteButton in v Lastnostih pod Varnost (angl. Security) tako kot smo naredili v prejšnjem primeru izberemo Sales Director. Za tem zaženemo aplikacijo in se prijavimo z uporabnikom Callahan Laura in pogledamo ali lahko brišemo vrstice v preglednici kategorij

(slika 4.38).



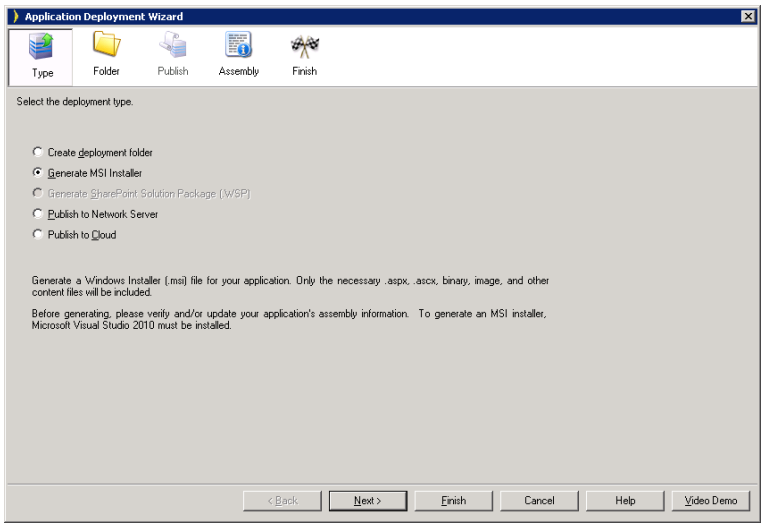
Slika 4.38: Ponovna prijava z uporabnikom Callahan Laura

Kot lahko opazimo na zgornji sliki, je dostop do gumba za brisanje onemogočen, kar pomeni da uporabnik, ki nima vloge 'Sales Director' ne more brisati zapisov kategorij na preglednici.

4.1.5 Postavitev spletne aplikacije

Ko imamo našo aplikacijo izdelano, jo običajno želimo postaviti na splet. Iron Speed ponuja več možnosti za postavitev naše spletne aplikacije.

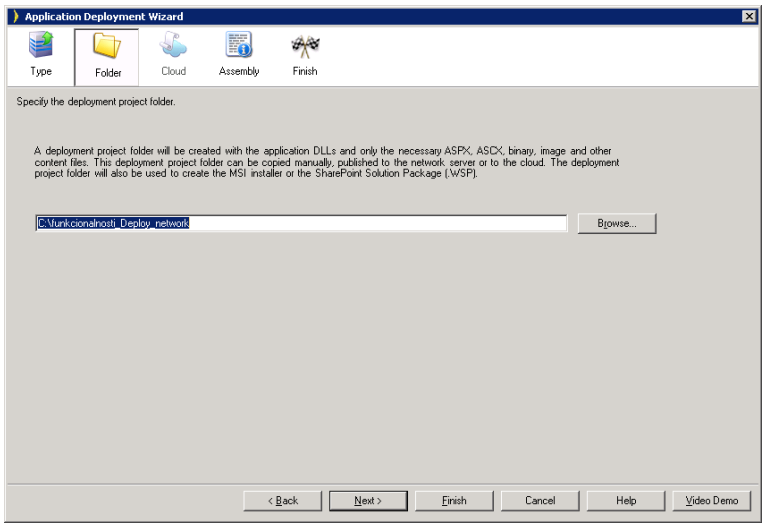
V glavnem meniju aplikacije kliknemo na spustni seznam Postavitev (angl. Deploy) in nato na Čarovnik za postavitev... (angl. Deployment Wizard...) ter s tem odpremo Čarovnik za postavitev aplikacije (angl. Application Deployment Wizard). To je čarovnik za pomoč pri postavljanju naših spletnih strani na produkcijsko okolje (slika 4.39).



Slika 4.39: Čarovnik za postavitev aplikacije

V prvem koraku čarovnika imamo na izbiro več tipov postavitve (angl. deployment types). Izbiramo lahko med:

- Izdelaj postavitveno mapo (angl. Create deployment folder), kateri nam zgradi mapo za postavitev projekta (angl. deployment project folder) naše aplikacije (slika 4.40). Zgradi nam eno DLL aplikacijo in samo potrebne datoteke kot so .aspx, .ascx, binary, image, itd. bodo vključene. Mapo za postavitev projekta lahko nato skopiramo na postavitveni produkcijski strežnik (angl. production server for deployment). Pri izdelavi postavitvene mape moramo imeti inštalirano orodje Microsoft Visual Studio.



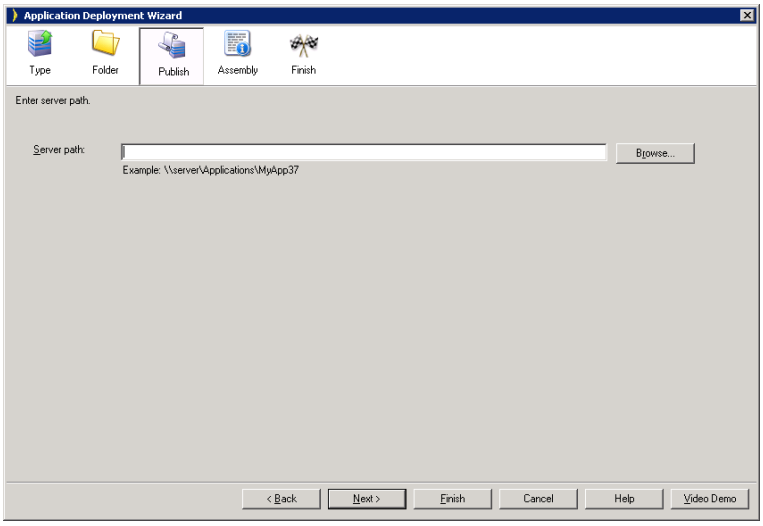
Slika 4.40: Navajanje mape za postavitev projekta

- Izdelaj MSI inštalacijo (angl. Generate MSI installer) nam izdela .msi datoteko za našo aplikacijo. Samo potrebne datoteke kot so .aspx, .ascx, binary, image, itd. bodo vključene. Tudi ta opcija izdela postavitveno mapo, kot smo omenili pri prejšnji opciji. To je najlažji način postavitve naše aplikacije. Načrtovalci orodja Iron Speed zelo priporočajo uporabo te opcije pri postavitvi naše aplikacije na produkcijo. MSI inštalacije so izvršljivi samostojni inštalacijski programi. Ne potrebujejo nobenega kopiranja datotek ali postavitve baznih procedur (angl. stored procedures). Ko poženemo MSI inštalacijo nam napravi vse inštalacije. Enostavno skopiramo setup.exe in .msi datoteke iz postavitvene mape v mapo na našem produkcijskem strežniku in zaženemo setup.exe za postavitev naše aplikacije na produkcijo. Inštalacijska datoteka MSI, ki jo zgradi Iron Speed poveže vse v eno MSI datoteko, ki je lahko nato inštalirana kjerkoli. MSI inštalacija vsebuje vse potrebne datoteke aplikacije, razen podatkovne baze. Kot pri prej omenjeni opciji moramo imeti tudi za to opcijo inštalirano orodje Microsoft Visual Studio.

- Izdelaj paket SharePoint rešitve (angl. Generate SharePoint Solution Package(WSP)), ki nam izdela mapo za postavitev našega pro-

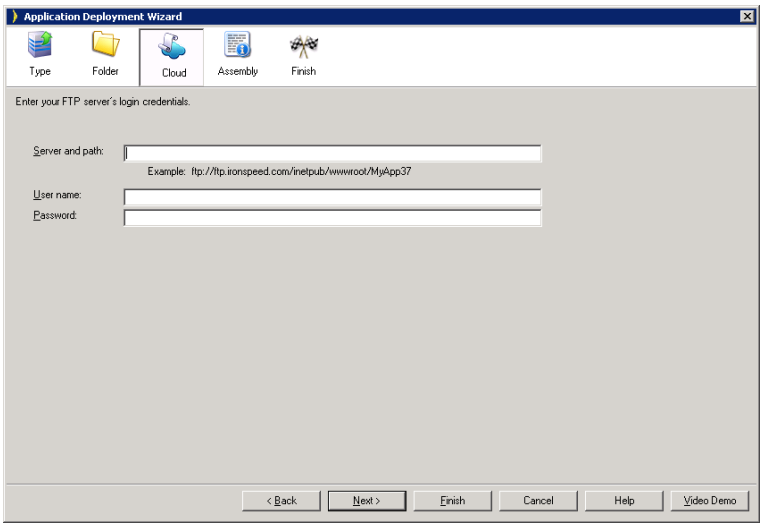
jekta naše aplikacije. Izdela nam aplikacijski paket rešitve za Windows (angl. Windows Solution Package, krat. WSP), ki ga nato lahko postavimo na našem strežniku SharePoint. Paket rešitve je CAB datoteka z .wsp končnico, ki vsebuje vse datoteke, ki morajo biti postavljeni na čelni del spletnega strežnika (angl. front-end Web server) ter množico inštalacijskih navodil baziranih na XML-ju (angl. XML-based installation instructions).

- Objavi na omrežni strežnik (angl. Publish to network server), ki nam izdela mapo za postavitev projekta naše aplikacije. Naša aplikacija je nato skopirana na specifično lokacijo na našem omrežju (slika 4.41).



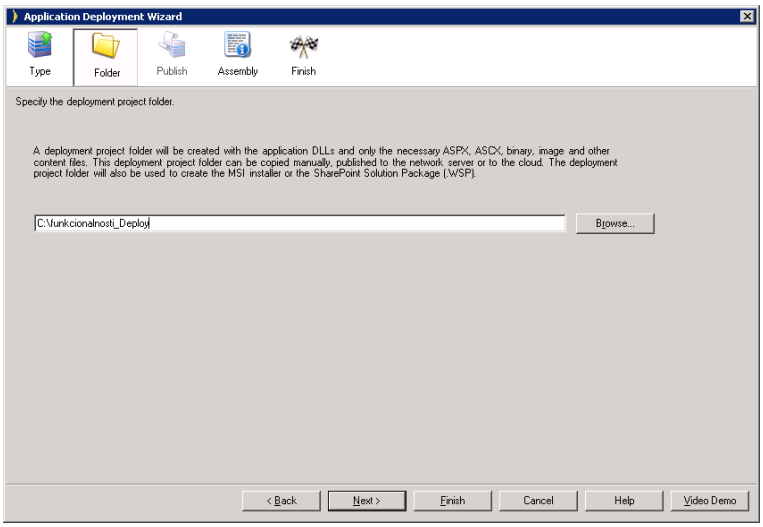
Slika 4.41: Določanje poti do strežnika

- Objavi v oblaku (angl. Publish to cloud), ki nam izdela mapo za postavitev projekta naše aplikacije. Aplikacija je nato skopirana preko protokola za prenos datotek (angl. File transfer protocol, krat FTP) na specifično lokacijo strežnika (slika 4.42).



Slika 4.42: Določanje FTP strežnika

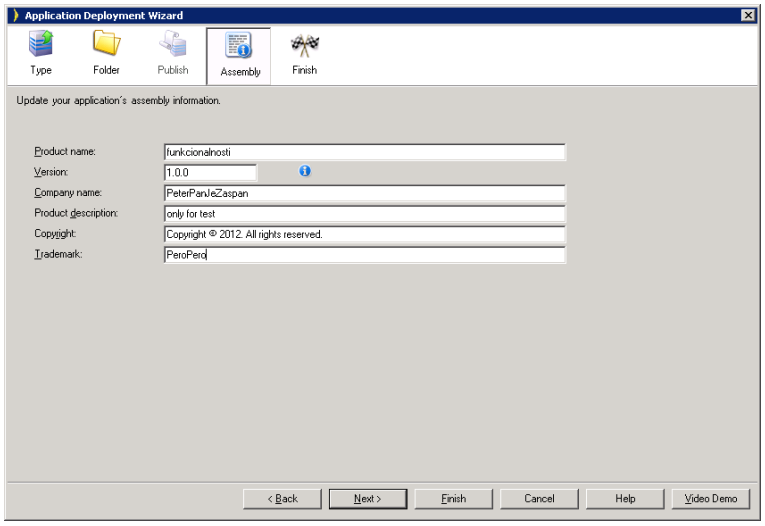
V našem primeru bomo prikazali izbiro opcije Izdelaj MSI inštalacijo, zato izberemo to opcijo in se pomaknemo naprej. Na naslednjem koraku izberemo mapo, v kateri bomo imeli shranjene podatke za postavitev aplikacije (slika 4.43).



Slika 4.43: Izbiranje mape, v kateri bodo podatki za postavitev aplikacije

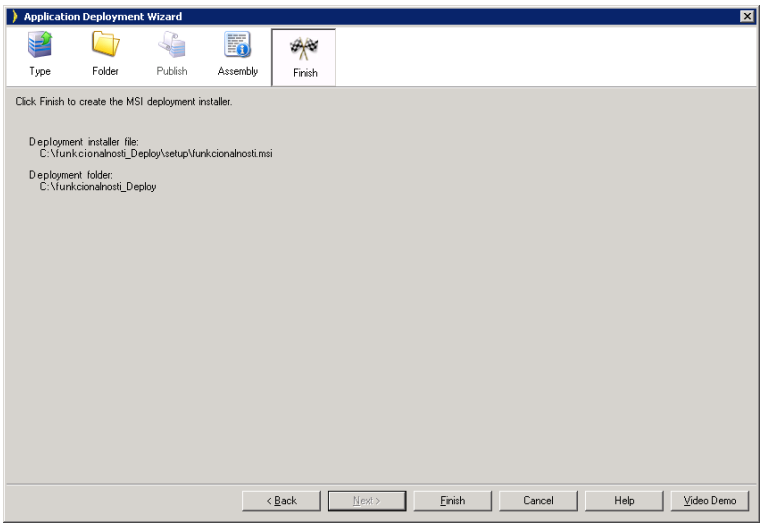
Določimo na C: disku mapo funkcionalnosti_Deploy in se pomaknemo na naslednji korak. Na naslednjem koraku vpišemo podatke naše aplikacije

(slika 4.44). V polje Ime produkta (anlg. Product name) napišemo 'funkcionalnosti', v polju Verzija (angl. Version) pustimo privzeto 1.0.0, v polje Ime podjetja (angl. Company name) napišemo 'PeterPanJeZaspan', v Opis produkta (angl. Product description) napišemo 'only for test', v polju Avtorska pravica (angl. Copyright) pustimo privzeto 'Copyright © 2012. All rights reserved.' in pod Blagovno znamko (angl. Trademark) napišemo 'PeroPero'. Podatki so izmišljeni in so samo za prikaz te funkcionalnosti.



Slika 4.44: Testni podatki naše aplikacijee

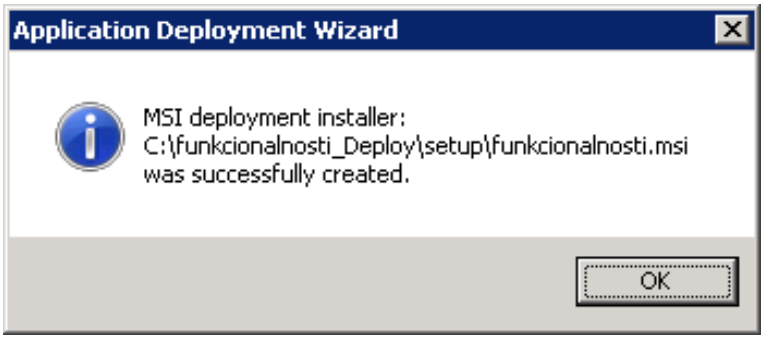
Na zgornji sliki imamo prikazano kako smo zapisali naše podatke za naš testni primer ter se nato postavimo na končni korak (slika 4.45).



Slika 4.45: Zaključni korak Čarovnika za postavitev

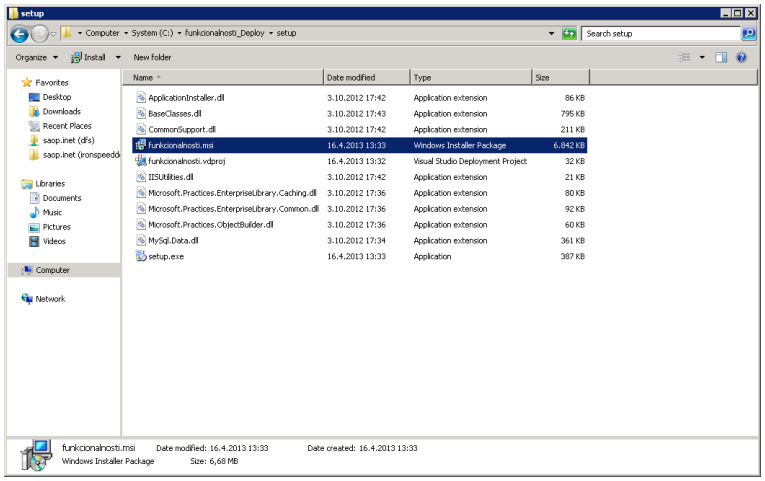
Kot lahko razberemo iz slike, vidimo da nam bo ob zaključitvi čarovnika izdelalo inštalacijsko datoteko .msi v mapi 'C:/funkcionalnosti_Deploy/setup' ter izdelalo mapo za postavitev 'C:/funkcionalnosti_Deploy'. Za zaključitev kliknemo na gumb Zaključi (angl. Finish) in s tem nam bo orodje izdelalo postavitveno mapo ter .msi datoteko.

Na koncu nam izpiše, da je bila inštalacijska datoteka MSI uspešno izdelana (slika 4.46).



Slika 4.46: Opozorilo, da je bila MSI inštalacija uspešno izdelana

Sedaj imamo na 'C:/funkcionalnosti_Deploy' vse podatke, ki jih potrebujemo za postavitev naše aplikacije. Če gremo v mapo setup lahko vidimo, da imamo noter inštalacijsko datoteko MSI in setup.exe (slika 4.47).



Slika 4.47: Mapa za postavitev naše aplikacije

To datoteko lahko nato inštaliramo na našem produkcijskem okolju.

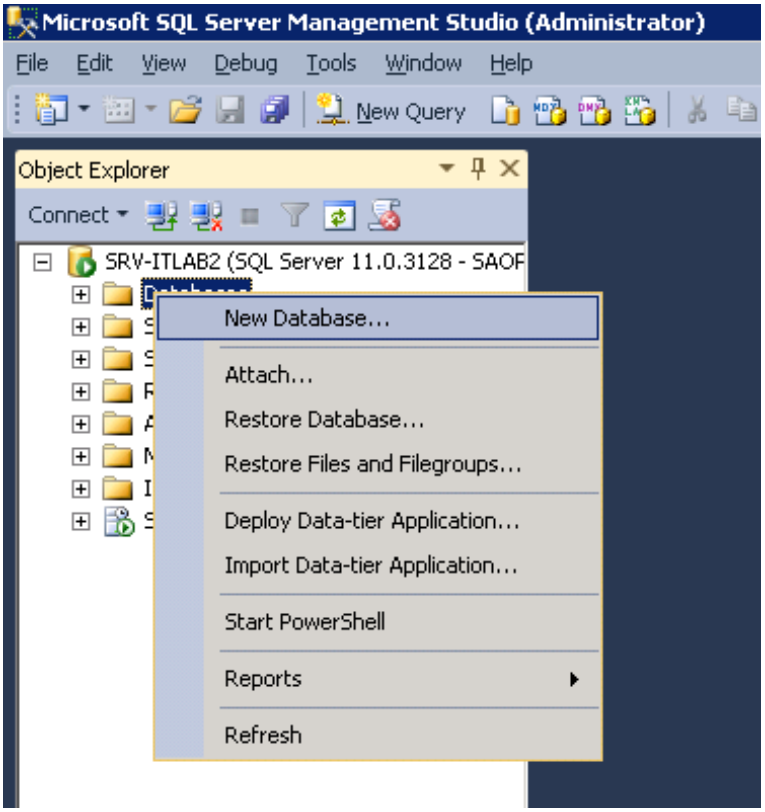
S tem smo prikazali osnovne funkcionalnosti orodja Iron Speed. V naslednjem podpoglavju pa bomo prikazali izdelavo spletne aplikacije in s tem še podrobneje prikazali zmogljivosti orodja Iron Speed ter si še podrobneje pogledali njegove lastnosti.

4.2 Izdelava spletne aplikacije za pomoč uporabnikom

Za podrobnejši prikaz funkcionalnosti orodja Iron speed bomo naredili preprost primer spletne aplikacije, za pomoč uporabnikom računalnika in računalniških programov. To bo aplikacija, v katero se bodo uporabniki prijavili in napisali zahtevo po pomoči. Ko bodo zahtevo oddali jo bo podpora uporabnikov pregledala in odgovorila na zahtevo z rešitvijo. Zahteve po pomoči, ki jih bodo oddajali se bodo navezovale na uporabo računalnika ter računalniških programov. Tako da bo to spletna aplikacija za pomoč računalniškim uporabnikom.

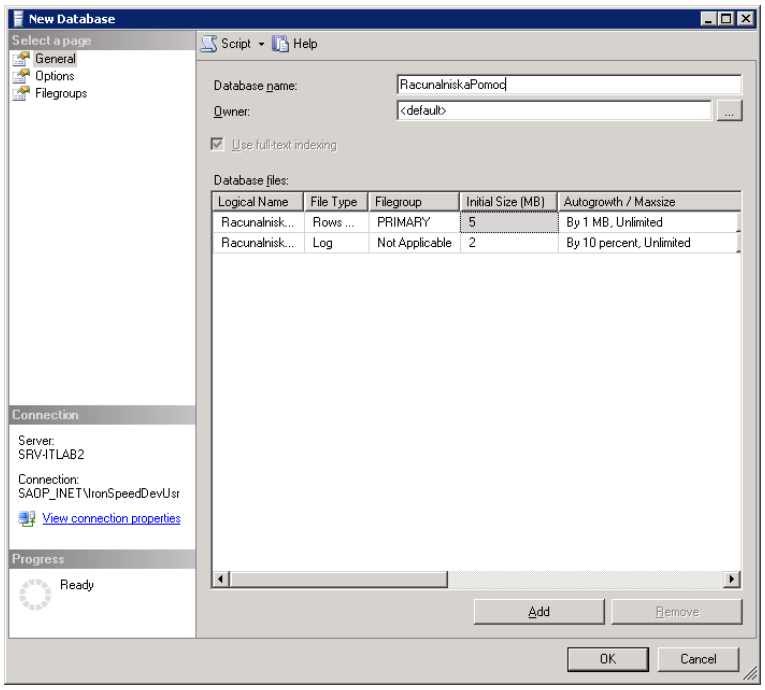
4.2.1 Izdelava podatkovne baze

Orodje Iron Speed nam na podlagi podatkovne baze izdela spletno aplikacijo, zato moramo najprej izdelati podatkovno bazo, ki jo bomo uporabili za izdelavo naše aplikacije. Za izdelavo podatkovne baze uporabimo orodje Microsoft SQL Server Managment Studio v katerem se povežemo lokalno. Ko se povežemo, kliknemo v Raziskovalcu objektov (angl. Object Explorer) na levi strani z desnim klikom na mapo Databases (slika 4.48) in nato kliknemo na gumb Nova podatkovna baza... (angl. New Database...).



Slika 4.48: Klik na gumb za izdelavo nove baze podatkov

Ko kliknemo na Nova podatkovna baza se nam odpre vnos nove podatkovne baze (slika 4.49), kjer vpišemo ime naše podatkovne baze v vnosno polje Ime podatkovne baze (angl. Database name). Ostale stvari pustimo tako kot so in kliknemo na gumb OK.

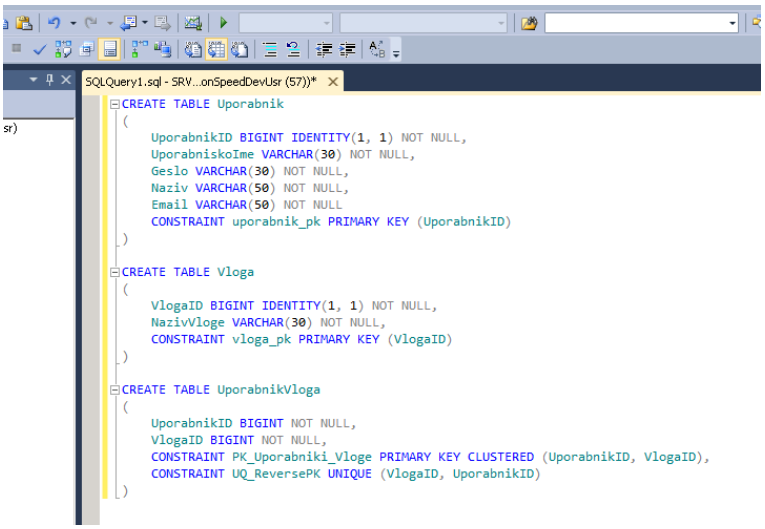


Slika 4.49: Določanje imena naše podatkovne baze

Naša podatkovna baza je sedaj ustvarjena in jo lahko napolnimo s tabelami, ki jih bomo potrebovali za izdelavo naše spletne aplikacije.

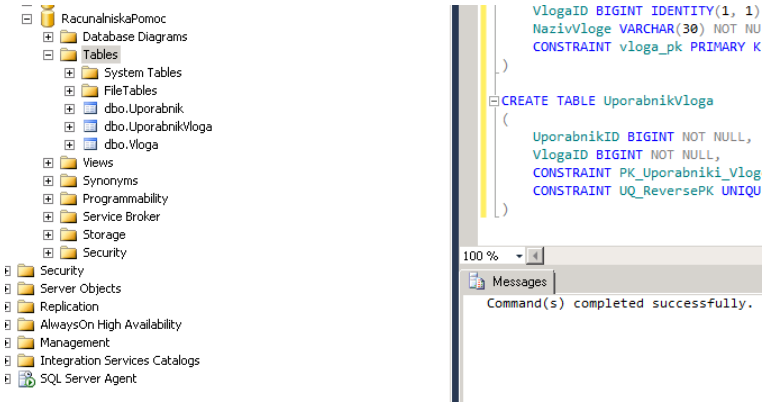
Ker se bodo uporabniki naše spletne aplikacije prijavljali v našo aplikacijo in bomo v naši aplikaciji uporabili varnost z vlogami uporabnikov (angl. role based security), bomo najprej naredili tabele za prijavo v našo spletno aplikacijo.

Z desnim klikom kliknemo na našo podatkovno bazo in nato kliknemo Nova poizvedba (angl. New Query) ter s tem odpremo stran za pisanje poizvedb (slika 4.50). Na tej strani napišemo poizvedbe za izdelavo tabele, ki bo hranila uporabnike, tabelo vlog, ki bo hranila vloge uporabnikov in tabelo uporabnik-vloga, katera bo povezovala ti dve tabeli.



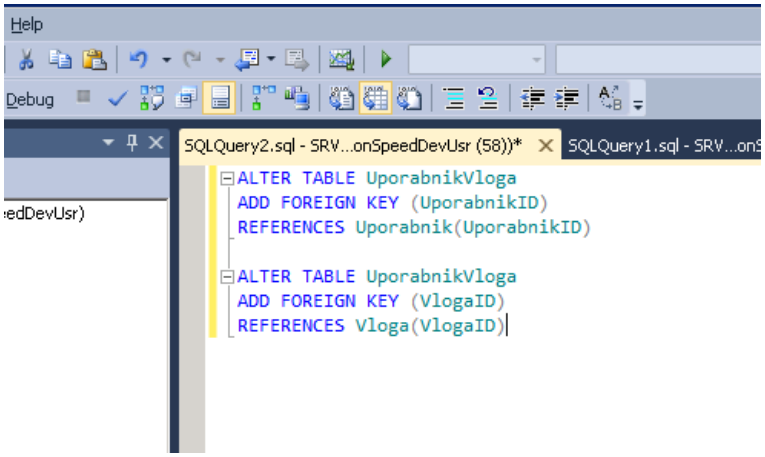
Slika 4.50: Poizvedba za izdelavo tabel Uporabnik, Vloga in UporabnikVloga

Tabela Uporabnik potrebuje uporabniško ime in geslo, s katerim se bodo naši uporabniki prijavljali. Naziv smo dodali, da bomo za prikaz imena uporabnika izbrali njegov naziv namesto uporabniškega imena. Tabelo Eposta smo pa dodali za hranitev uporabnikovih naslovov elektronske pošte. Pri vlogah potrebujemo samo naziv vloge. Seveda potrebujemo pri tabeli Uporabnik in pri tabeli Vloga tudi primarne ključe (angl. primary key). V UporabnikVloga tabeli pa potrebujemo ključ uporabnika in vloge. Ko napišemo poizvedbe za izdelavo tabel pritisnemo na gumb Izvedi (angl. Execute) v zgornjem meniju, ki ima poleg napisa Execute tudi rdeč klicaj ali pa pritisnemo na tipko F5 in tako poženemo poizvedbe. Ko poženemo te poizvedbe lahko na levi strani vidimo, da nam je izdelalo te tri tabele (slika 4.51), ki jih bomo potrebovali za izdelavo varnosti naše spletne aplikacije.



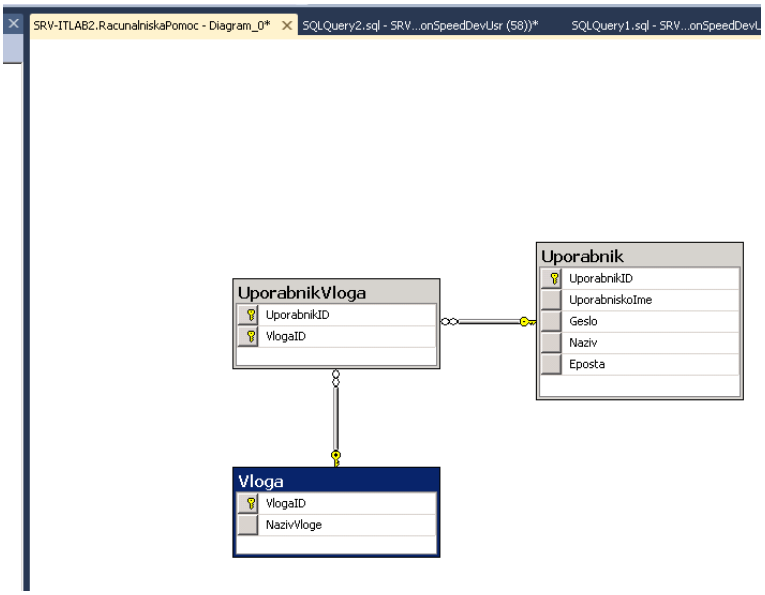
Slika 4.51: Prikaz izdelanih tabel, ko smo pognali poizvedbo

Tabeli UporabnikVloga moramo dodati še tuje ključe (angl. foreign key) za povezavo do tabele Uporabnik in tabele Vloga (slika 4.52).



Slika 4.52: Dodajanje tujih ključev tabeli UporabnikVloga

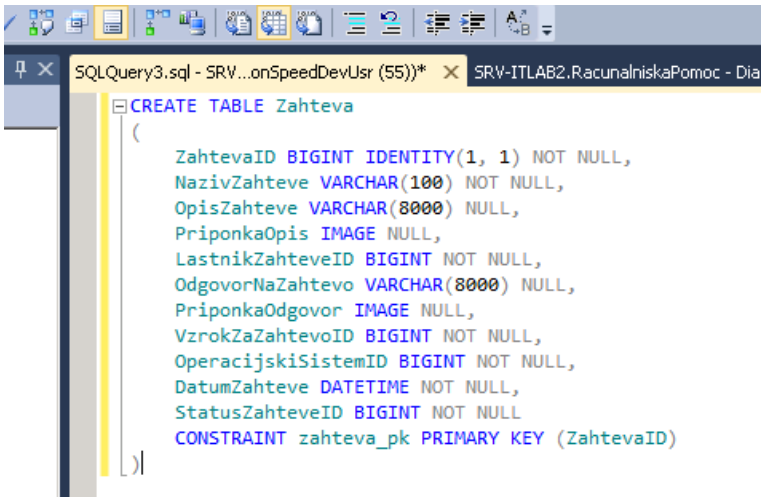
Nato pritisnemo na tipko F5 in poženemo še to poizvedbo. Če ustvarimo nov diagram podatkovne baze (angl. Database Diagrams), vidimo da so tabele lepo povezane (slika 4.53).



Slika 4.53: Podatkovni diagram tabel Uporabnik, Vloga in UporabnikVloga

Za prijavo naših uporabnikov v aplikacijo so te tri tabele dovolj. Tako da moramo sedaj načrtovati in izdelati ostale tabele.

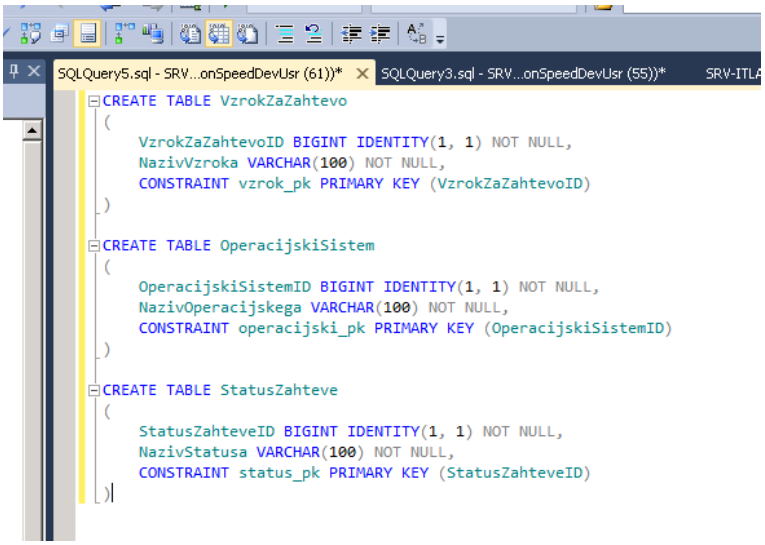
Ker bodo v naši aplikaciji uporabniki oddajali zahteve po pomoči, moramo izdelati tabelo, ki bo hranila zahteve po pomoči. Ponovno kliknemo na Nova poizvedba in izdelamo tabelo za zahteve po pomoči (slika 4.54) in kliknemo F5, da se poizvedba požene in izdela se nam nova tabela, ki bo hranila zahteve.



Slika 4.54: Poizvedba za izdelavo tabele Zahteva

V tabeli Zahteva imamo naziv in opis zahteve, ki ga bo uporabnik vpisal ter izbiral priponke in dobil odgovor od podpore uporabnikov na zahtevo. Uporabnik bo lahko izbral kakšen je vzrok za zahtevo in kakšen operacijski sistem ima, zato smo naredili polja z VzrokZaZahtevoID in OperacijskiSistemID. Polje LastnikZahteveID bo hranilo primarni ključ tabele uporabnika, ki bo zahtevo oddal. Ko se bo zahteva po pomoči shranila se bo avtomatično shranil trenutni datum s trenutno uro in določil status zahteve zato potrebujemo tudi polja za uro in status.

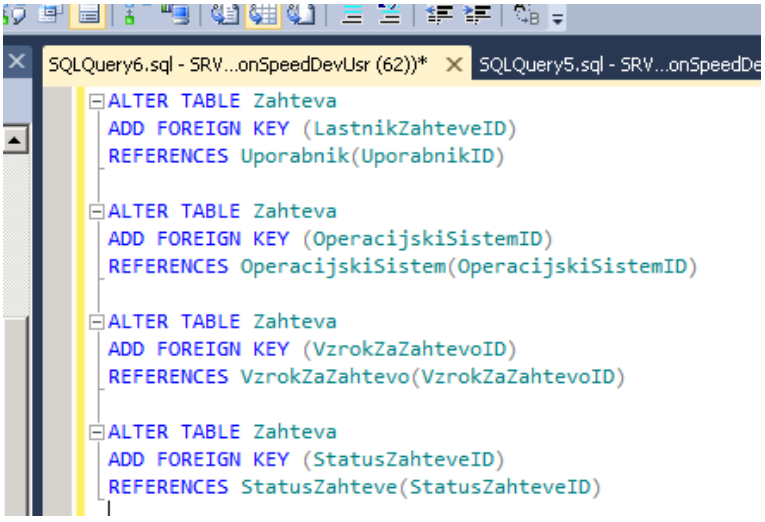
Nato izdelamo tabeli ki bosta hranili vzroke za zahtevo, operacijske sisteme in statuse zahteve (slika 4.55) ter nato pritisnemo F5 in poženemo poizvedbe.



Slika 4.55: Izdelava VzrokZaZahtevo, OperacijskiSistem in StatusZahteve

Pri vsaki od tabel potrebujemo samo nazive, se pravi naziv vzroka za zahtevo, naziv operacijskega sistema in naziv statusa ter primarne ključe.

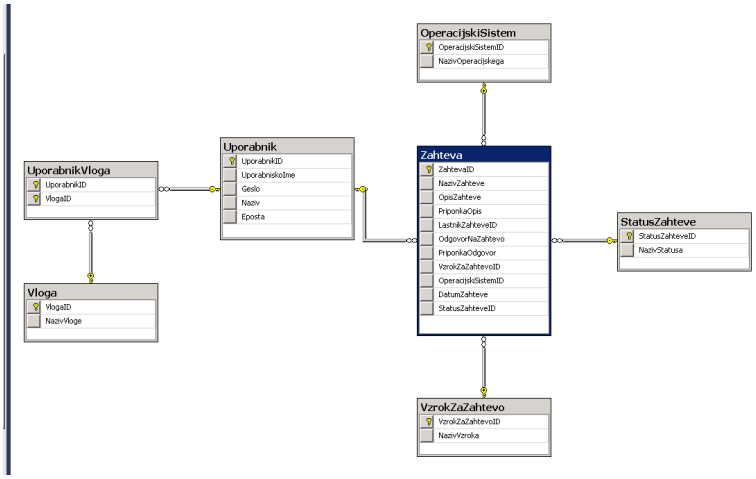
Tabeli zahteva moramo sedaj dodati tuje ključe za povezavo s tabelami Uporabnik, VzrokZahteve, OperacijskiSistem in StatusZahteve (slika 4.56) ter s tipko F5 poženemo poizvedbo.



Slika 4.56: Dodajanje tujih ključev tabeli Zahteva

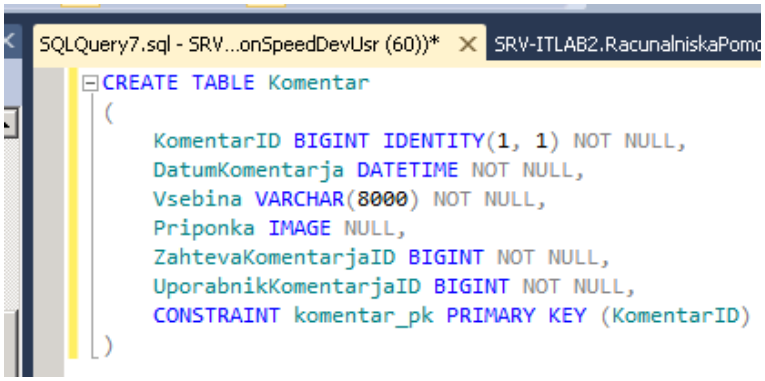
Če odpremo sedaj tabele v diagramu podatkovne baze lahko vidimo, da so

vse tabele med sabo lepo povezane (slika 4.57).



Slika 4.57: Diagram podatkovne baze še z ostalimi tabelami

Uporabnikom bomo dovolili, da bodo izdelanim zahtevam lahko dodajali komentarje, če bodo imeli še dodatna vprašanja ali pa če jim recimo odgovor na problem ne bo zadostoval. Zato izdelamo še tabelo (slika 4.58), ki bo hranila komentarje, ki jih bodo uporabniki pisali na izdelano zahtevo. S tipko F5 poženemo poizvedbo in izdelamo tabelo Komentar.

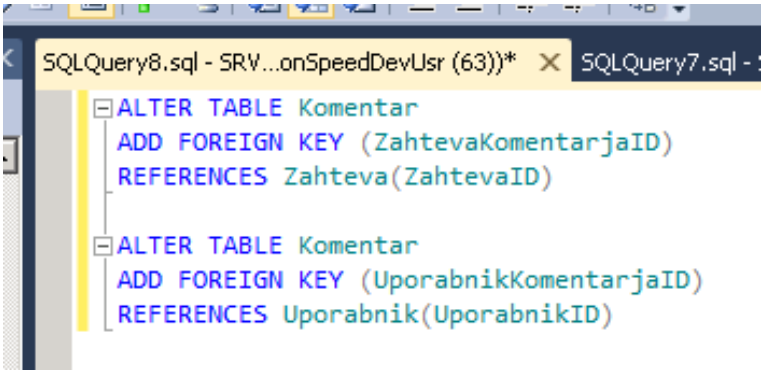


Slika 4.58: Poizvedba za izdelavo tabele Komentar

V tabeli Komentar, potrebujemo polje za datum, kamor se bo shranil trenutni datum, ko bo komentar oddan. Uporabnik bo vnesel vsebino komentarja in če bo želel pripel priponko ob komentarju. Tabela potrebuje tudi ključ

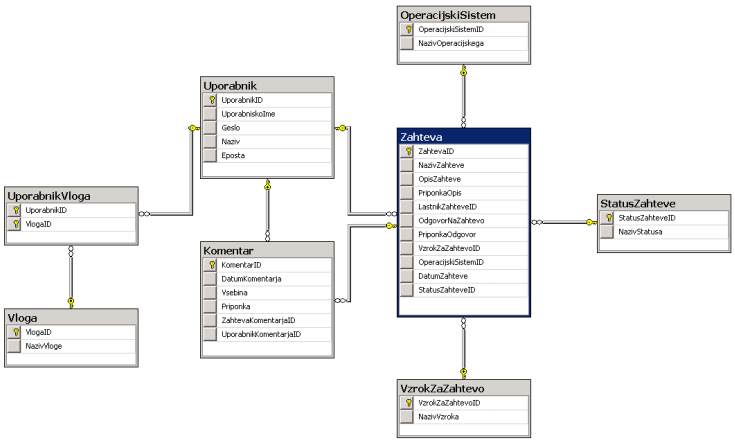
ZahtevaKomentarjaID, ki bo hranil ključ zahteve, na katero se bodo pisali komentarji in UporabnikKomentarjaID, ki bo hranil primarni ključ uporabnika, ki bo vnesel komentar.

Nato bomo morali še povezati tabelo Komentar z tabelo Zahteva in Uporabnik, tako da bomo dodali tej tabeli dva tuja ključa (slika 4.59) in s tipko F5 pognali poizvedbo.



Slika 4.59: Dodajanje tujih ključev tabeli Komentar

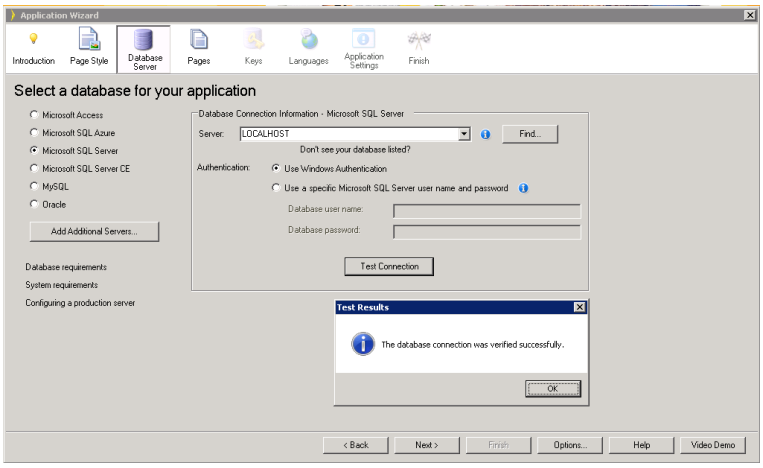
S tem smo naredili še zadnji korak pri izdelavi naše baze podatkov in sedaj imamo izdelano bazo podatkov na podlagi katere bomo izdelali našo spletno aplikacijo za pomoč uporabnikom. Če nato ustvarimo nov diagram podatkovne baze si lahko pogledamo našo shemo podatkovne baze (slika 4.60) in vidimo da so tabele sedaj v redu povezane in lahko nadaljujemo delo v orodju Iron Speed.



Slika 4.60: Diagram podatkovne baze z vsemi tabelami

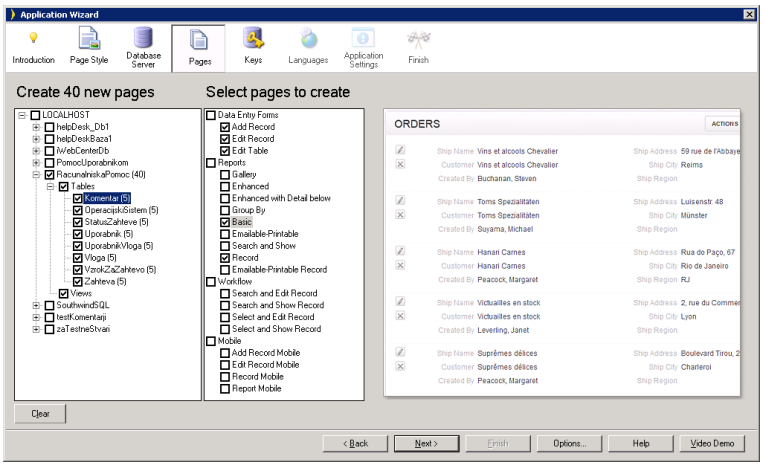
4.2.2 Začetek izdelave spletne aplikacije

Sedaj ko imamo pripravljeno shemo podatkovne baze odpremo orodje Iron Speed in zgradimo našo spletno aplikacijo. Tako kot smo prikazali pri funkcionalnostih orodja, tudi v tem primeru izdelamo novo aplikacijo tako da izberemo Datoteka –>Nova –>Aplikacija... (angl. File –>New –>Application...) in po korakih določimo nastavitve naše spletne aplikacije. Pri stilu strani lahko pustimo kar že izbran stil z imenom Reisling in se z tipko Naprej (angl. Next) pomaknemo na naslednji korak. Na zavihku za izbiranje strežnika podatkovne baze izberemo Microsoft SQL Server in se povežemo lokalno, tako da vpišemo 'localhost' v polje Strežnik (angl. Server) in z miško kliknemo na gumb Testiraj povezavo (angl. Test Connection), da preverimo našo povezavo (slika 4.61). Ko nam orodje vrne uspešno preverjeno povezavo kliknemo na gumb OK in se pomaknemo na naslednji zavihek.



Slika 4.61: Testiranje povezave do strežnika podatkovne baze

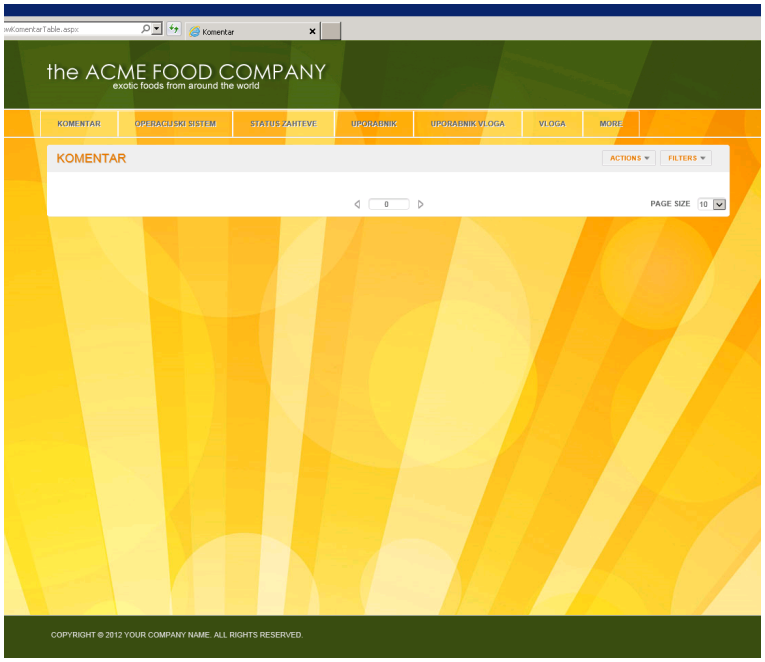
V zavihku za izdelavo spletnih strani izberemo sedaj našo podatkovno bazo, ki smo jo pravkar izdelali in pri straneh, ki jih bo orodje izdelalo (angl. Select pages to create) označimo pod Poročili (angl. Reports) samo 'Basic' in 'Record' (slika 4.62), nato izberemo vse tabele naše baze podatkov in se pomaknemo naprej.



Slika 4.62: Izbiranje strani, ki jih bo Iron Speed izdelal

Pri izbiranju ključev nam ni potrebno dodajati nobenih virtualnih primarnih in tujih ključev, ker smo primarne in tuje ključe že izdelali v naši bazi podatkov. Nato se premaknemo naprej na zavihek za izbiranje jezikov in pustimo

kar slovenščino pod privzeto (čeprav bo potreben prevod iz angleščine) in se premaknemo na naslednji korak. Na koraku za nastavitve naše aplikacije vpišemo pod ime naše spletne aplikacije 'RacunalniskaPomocUp', za tip aplikacije izberemo Web Application for .NET, programski jezik C#, ogrodje .NET 4.0 in izdelavo SQL pustimo Inline. Nato se postavimo na zadnji korak in kliknemo na gumb Zaključi, da nam bo Iron Speed izdelal našo spletno aplikacijo (slika 4.63).



Slika 4.63: Aplikacija, ki jo je Iron Speed izdelal na podlagi podatkovne baze

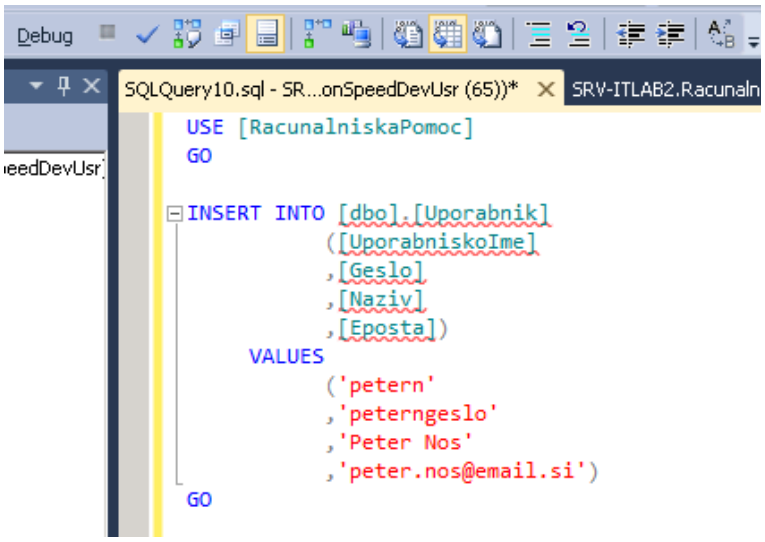
Orodje Iron Speed nam je sedaj izdelalo našo spletno aplikacijo, vendar tej aplikaciji manjka uporabnost. Tako da se moramo sedaj posvetiti uporabnosti našega programa. Da bomo uporabnost naše spletne aplikacije lažje preverili bomo v našo bazo podatkov vnesli nekaj vzorčnih uporabnikov in njihovih vlog. Vnesli bomo tudi vzroke za zahtevo, ki jih bodo uporabniki izbirali, operacijske sisteme, ki jih bodo izbirali in statuse, ki se bodo nastavljali ob izdelavi ali urejanju zahteve.

Za dodajanje novih zapisov v tabelo imamo sedaj dve možnosti, lahko uporabimo INSERT stavke v SQL Server Managment Studiu ali pa dodamo

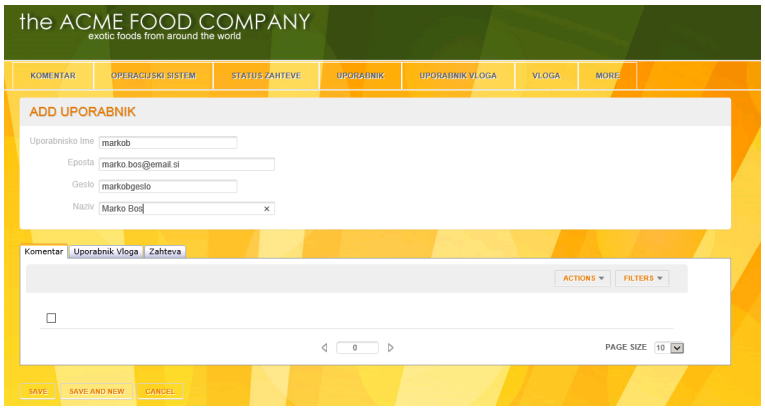
nove zapise kar preko naše na novo izdelane aplikacije, ker imamo izdelane gumbe za dodajanje zapisov in nam naredijo enako kot SQL stavki INSERT.

V tabelo Uporabnik dodamo pet uporabnikov in v tabelo Vloga dodamo dve vlogi. Uporabnikom določimo njihove vloge v tabeli UporabnikVloga. Vlogi bosta "Uporabnik računalnika"in "Podpora uporabnikom", naši uporabniki pa bodo trije uporabniki računalnika in dva uporabnika iz podpore uporabnikom. Dodamo vzorčne uporabnike računalnika z podatki uporabniško ime, geslo, naziv in elektronska pošta: "petern, peterngeslo, Peter Nos, peter.nos@email.si", "markob, markobgeslo, Marko Bos, marko.bos@email.si", "frankok, frankokgeslo, Franko Kos, franko.kos@email.si"in vzorčne uporabnike za podporo uporabnikom "darkop, darkopgeslo, Darko Podpora, darko@podpora.si", žarkop, zarkopgeslo, Žarko Podpora, zarko@podpora.si". Med vzroki za zahtevo bodo uporabniki lahko izbirali med štirimi opcijami: "Ni mogoče nadaljevati z delom", "Premalo informacij", "Neznanje programa"in "Ostalo". Za operacijski sistem bodo izbirali med "Linux", "Windows XP", "Windows 7", "Windows 8", "Mac OS X". Status zahteve bo pa lahko "Odprt"ali "Rešen".

Na spodnjih dveh slikah (slika 4.64 in slika 4.65) prikazujemo kako lahko podatke vnašamo v tabele z SQL stavki ali pa kar preko naše izdelane spletne aplikacije, ki nam jo je Iron Speed pravkar generiral.



Slika 4.64: Prikaz dodajanja zapisov z SQL stavki



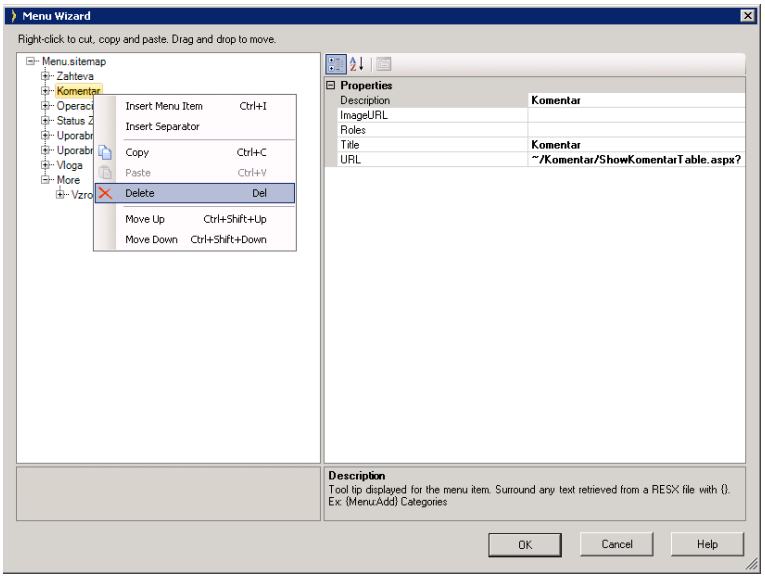
Slika 4.65: Prikaz dodajanja zapisov preko aplikacije

S tema dvema načinoma lahko vnesemo podatke v naše tabele in nato nadaljujemo z izdelavo naše spletne aplikacije tako, da ji dodamo uporabnost.

4.2.3 Izdelava uporabnosti spletne aplikacije

Orodje Iron Speed je že samo izdelalo veliko funkcionalnosti naše spletne aplikacije, vendar jo moramo sedaj z orodjem sami preurediti, da bo spletna aplikacija takšna kot smo si zamislili. Najprej odstranimo odvečne strani, ki nam jih je orodje izdelalo. V našem primeru bodo uporabniki delali z zahtevami po pomoči (jih oddajali, pregledovali, itd.), zato potrebujemo samo

stran Zahteva, ostale strani pa odstranimo iz menija. To naredimo tako, da se v Raziskovalcu aplikacije postavimo v mapo Menu Panels in z desnim klikom kliknemo na Menu.ascx ter nato kliknemo na Konfiguriranje (angl. Configure). S temi akcijami se nam bo odprl Čarovnik za menije (angl. Menu Wizard), ki je čarovnik za urejanje menija naše aplikacije. V Čarovniku menija kliknemo z desnim klikom na vsa polja z desnim klikom, razen na polje Zahteva in nato izberemo Pobriši (angl. Delete) in tako pobrišemo polje iz našega menija (slika 4.66).



Slika 4.66: Brisanje strani, ki jih ne potrebujemo

Nato v Raziskovalcu aplikacije odpremo mapo Zahteva in z desnim klikom kliknemo na ShowZahtevaTable.aspx in izberemo na Nastavi kot začetno stran (angl. Set As Web Start Page) ter s tem nastavimo prikaz zapisov v tabeli Zahteva kot začetno stran naše aplikacije. Če nato ponovno poženemo aplikacijo imamo na voljo v meniju samo tabelo Zahteva in kot začetna stran se nam prikaže pregled zahtev, ampak ker še nismo oddali nobene zahteve je naša stran trenutno prazna.

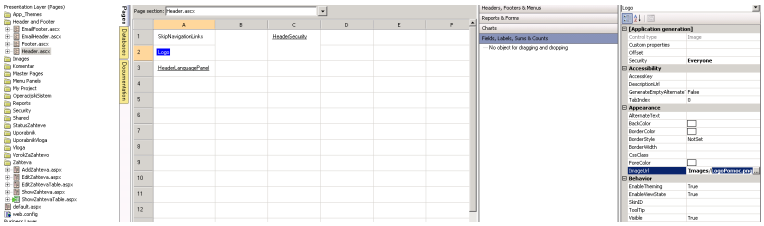
Naša spletna aplikacija je sedaj brez varnosti. Če bi našo spletno aplikacijo postavili na splet bi jo lahko vsakdo uporabljal. Zato bomo v naslednjem

koraku izdelali varnost tako, da se bodo morali naši uporabniki vpisati v spletno aplikacijo, če jo bodo želeli uporabljati.

Tukaj pridejo v poštev naše tabele Uporabnik, Vloga in UporabnikVloga. V našem orodju se postavimo na Orodja in nato izberemo Čarovnik za varnost aplikacije. Nato izberemo na prvem koraku avtentikacijo in avtorizacijo preko podatkovne baze ter se postavimo na naslednji korak, kjer izberemo našo tabelo, ki hrani uporabnike in določimo polja za uporabniško ime, geslo, ID in elektronski naslov. Nato se premaknemo na naslednji zavihek, kjer izberemo tabelo UporabnikVloga, ki povezuje uporabnike in vloge ter določimo ključ vloge in ključ uporabnika, in se pomaknemo na korak za izbiranje varnosti na naših straneh. Na zavihku strani izberemo Samo prijavljeni uporabniki, ker bodo našo aplikacijo lahko uporabljali samo uporabniki, ki se bodo v aplikacijo vpisali. Ko izberemo to, izberemo še vse strani in kliknemo gumb Nastavi ter tako vsem stranem nastavimo, da so vidne samo za prijavljene uporabnike. Postavimo se na naslednji korak in zaključimo z varnostjo.

Če sedaj poženemo našo spletno aplikacijo vidimo, da ne moremo iti dalje, če se ne prijavimo. S tem smo omejili dostopnost naše spletne aplikacije na naše uporabnike.

Naša spletna aplikacija ima trenutno privzet logotip, privzeto stran za prijavo in angleške izraze. Na tem koraku izdelave spletne aplikacije bomo zamenjali logotip in popravili stran za prijavo ter prevedli kar je potrebno v slovenski jezik. Izdelamo naš logotip in ga shranimo v mapo Images naše spletne aplikacije. Nato odpremo v orodju Iron Speed mapo Header and Footer, kjer lahko urejamo glavo in nogo naše spletne aplikacije ter kliknemo na Header.ascx. S tem pridemo do urejanja glave naše spletne aplikacije in kliknemo na polje Logo ter v Lastnostih (angl. Properties) pod Videz (angl. Appearance) v polju ImageUrl vpišemo ime našega logotipa (slika 4.67), ki smo ga pravkar shranili.



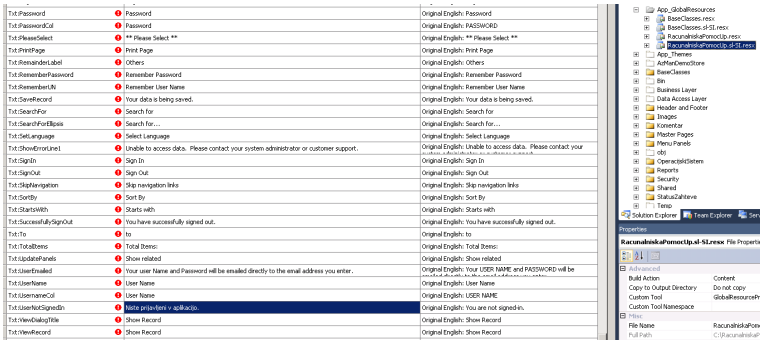
Slika 4.67: Določanje slike za logotip naše aplikacije

Nato premaknemo HeaderSecurity na polje B3, da bo na desni strani in malo nižje od logotipa. Za tem pa prevedemo vse tekste naše aplikacije v slovenski jezik, ker so teksti trenutno v angleškem jeziku.

Za lokalizacijo naše aplikacije v slovenski jezik bomo prešli na orodje Microsoft Visual Studio 2010. Iron Speed ima direktno povezavo do tega orodja v zgornjem meniju, kjer lahko kliknemo na gumb Visual Studio, ki bo ob kliku odprlo Visual Studio in nas prestavilo v to orodje. Visual Studio lahko uporabljamo tudi za lažje pisanje naše lastne kode, ker je kodo lažje pisati v programu Visual Studio, ker ima Intelli-sense, katerega žal Iron Speed ne vsebuje. Intelli-sense nam sprotno pomaga pri programiranju, tako da nam poda razne namige, sprotno popravlja kodo, prikaže funkcije, ki so na voljo, itd. Visual Studio je tudi boljše orodje za razhroščevanje (angl. debug) naše kode. Za enkrat pa še ne bomo pisali nobene kode ampak bomo orodje Visual Studio uporabili samo za prevajanje besedila aplikacije v slovenski jezik.

Z klikom na gumb Visual Studio v orodju Iron Speed odpremo orodje Visual Studio, ki se nam že avtomatično odpre na strani Header.ascx.cs, kjer imamo kodo naše glave. Teksti Iron Speed aplikacije so hranjeni v izvornih datotekah RESX (angl. RESX resource files). Tekste najdemo v mapi App_GlobalResources, ki jo pa moramo še vključiti v projekt. Ker imamo to mapo že v projektu, gremo na meniju na Projekt (angl. Project) in kliknemo Prikaži vse datoteke (angl. Show All Files) in s tem v Raziskovalcu rešitev (angl. Solution Explorer) na desni strani Visual Studia prikažemo vse datoteke, ki obstajajo v našem projektu. Za vključitev datotek v projekt kliknemo na njih z desnim klikom in nato izberemo Vključi v projekt (angl. Include

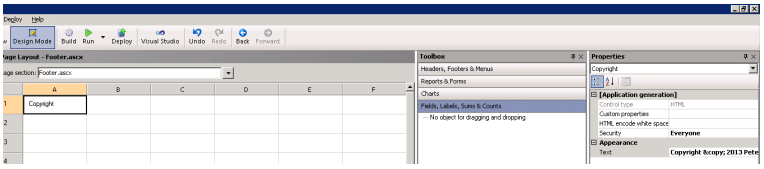
In Project) ter tako začnemo upravljati z našimi datotekami. Ko vključimo App_GlobalResources v projekt, odpremo RacunalniskaPomocUp.sl-SI.resx (slika 4.68) in spremenimo tekste v slovenski jezik.



Slika 4.68: Prikaz prevoda v slovenski jezik

Kot smo naredili na zgornji sliki za Txt:UserNotSignedIN napravimo še za ostale tekste in jih v resx datoteki prevedemo v slovenščino in tako prevedemo tekste naše spletne aplikacije v slovenski jezik. Ko prevedemo vse tekste v resx datoteki se postavimo nazaj na orodje Iron Speed in tam preuredimo še nogo naše aplikacije, ker je sedaj v uporabi privzeta noga.

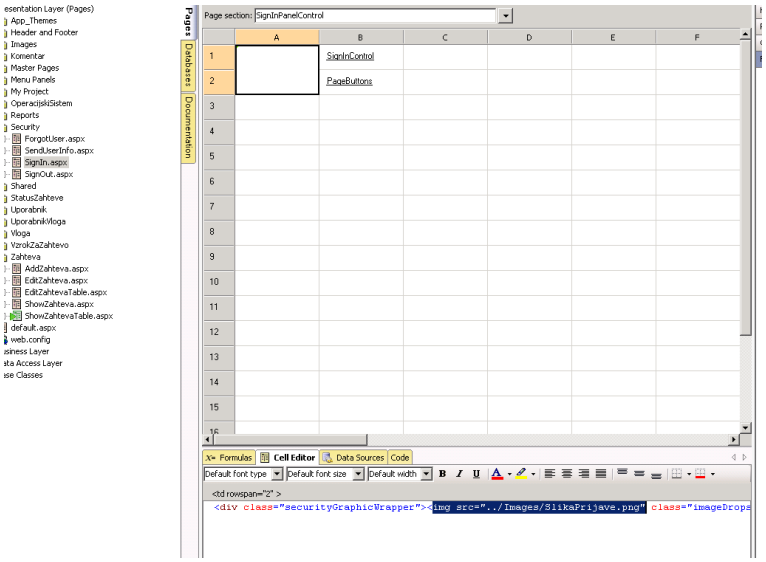
V Raziskovalcu aplikacije se postavimo na Footer.ascx in kliknemo na polje Copyright. V Lastnostih tega polja imamo sedaj vpisan privzeti tekst, ki nam ga Iron Speed ponudi 'Copyright © 2012 Your Company Name. All rights reserved'. Ta tekst spremenimo (slika 4.69) v 'Copyright © 2013 Peter Šfiligoj. Diplomska naloga'.



Slika 4.69: Spreminjanje teksta polja Copyright v nogi

Nato moramo podobno kot smo pri logotipu zamenjali sliko, zamenjati privzeto sliko pri prijavi in odjavi naše spletne aplikacije. Slika pri prijavi in odjavi je shranjena v html-ju strani SignIn.aspx in SignOut.aspx v Delih

strani SignInPanelControl in SignInPanelControl, zato jim lahko sliko zamenjamo tako, da kliknemo na to polje in jim spremenimo html v Cell Editorju. Privzete slike o odjavi in prijavi zamenjamo z našimi slikami, ki jih, ko jih izdelamo, prenesemo v mapo Images in iz tam nato pokličemo namesto privzete slike (slika 4.70).



Slika 4.70: Nastavljanje naše slike ob prijavi v aplikacijo

Sedaj ko smo prevedli našo aplikacijo in ji preuredili začetno stran za prijavo lahko poženemo aplikacijo (slika 4.71) in vidimo, da imamo izraze v slovenskem jeziku in spremenjen logotip v glavi aplikacije in pomaknjen vpis desno pod logotipom, spremenjeno sliko prijave in spremenjeno nogo aplikacije.

Slika 4.71: Vstopna stran spletne aplikacije

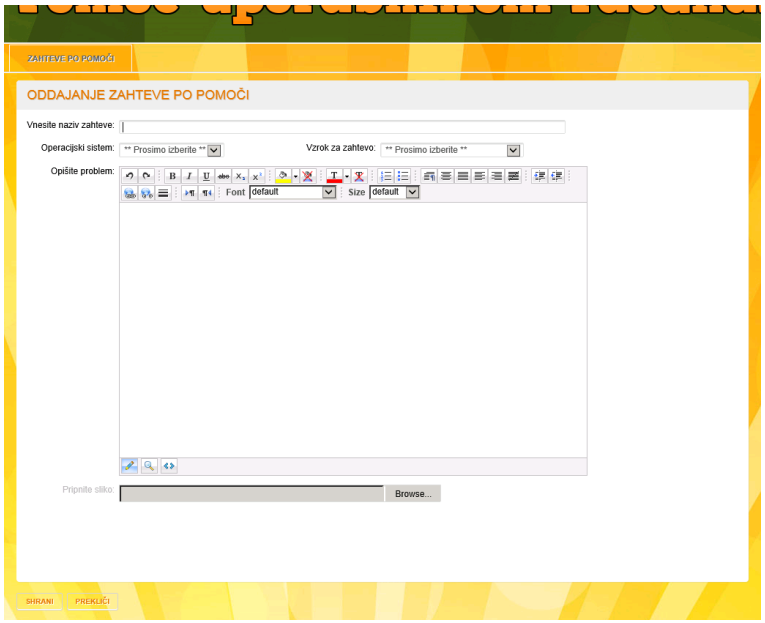
Naši uporabniki se lahko sedaj prijavljajo v aplikacijo in iz nje odjavljajo, poleg tega pa smo še prevedli aplikacijo in jo preoblikovali. V nadaljevanju se bomo posvetili dodajanju in upravljanju z zahtevami po pomoči.

Ko smo prijavljeni v našo aplikacijo lahko dodamo novo zahtevo na dva načina. Lahko se postavimo na meni ZAHTEVA in kliknemo Dodaj Zahteva ali pa kliknemo na Dejanja in nato kliknemo gumb za dodajanje. V naši aplikaciji omejimo dodajanje zahteve tako, da bodo lahko samo uporabniki z vlogo 'Uporabnik računalnika' oddajali nove zahteve po pomoči, zato moramo te dve možnosti dodajanja omejiti na samo te uporabnike. Pobrišemo tudi masovno urejanje zahtev in spremenimo naziv ZAHTEVA na ZAHTEVE PO POMOČI. Iz Dodaj Zahteva spremenimo na Dodaj zahtevo in iz Prikaži Zahteva na Prikaz zahtev po pomoči. To napravimo v Čarovniku za menije. Gumbu Dodaj pri Dejanjih pa spremenimo pravice kar v Prilagoditvi strani ShowZahtevaTable.aspx pri Lastnostih tega gumba.

Dodajanje zahtev po pomoči moramo sedaj malo preurediti, da bo bolj uporabno. Uporabnost preuredimo kar v Prilagoditvi strani AddZahteva.aspx. Odgovor na zahtevo lahko dodaja samo podpora, zato polje za odgovor in za dodajanje priponke k odgovoru pobrišemo. Datum zahteve ne bo viden, ampak se bo ob shranitvi zahteve shranil trenutni datum in trenutna ura. Zato datumu spremenimo Vidljivost (angl. Visible) na 'false' in s tem zakri-

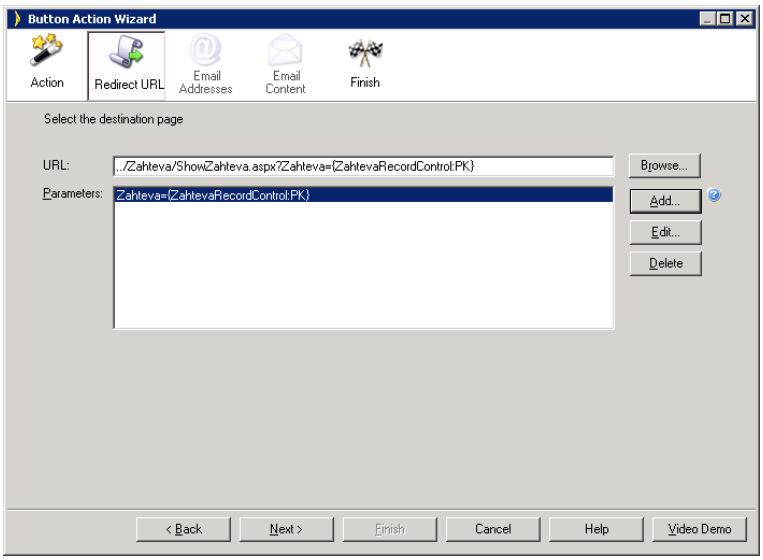
jemo njegovo vidljivost ter ga inicializiramo v Formulah na 'Now()'. Tako bo datum pridobil trenutni datum in uro. Tudi lastnika zahteve uporabnik ne bo izbiral ampak se bo lastnik zahteve inicializiral na trenutno prijavljenega uporabnika, zato ga v Formulah inicializiramo na 'UserId()' in polje zakrijemo, da ne bo vidno. Vsem spustnim seznamom (lastnik zahteve, operacijski sistem, status in vzrok za zahtevo) pobrišemo AddRecordLink, zato da uporabnik ne bo mogel dodajati novih zapisov tem tabelam. Status zahteve se bo nastavil na 'Odprt', zato njegov ID inicializiramo na 1 (ker je naziv pri tabeli Status, ki ima ID 1 'Odprt'). Tudi status zahteve nato skrijemo. Polja za prikaz nato še malo preuredimo, da bo dodajanje zahtev bolj pregledno in našim uporabnikom prijaznejše (popravimo postavitev, spremenimo nazive oznakam (angl. label), spremenimo razvrščanje spustnih seznamov, itd.).

Ker je tabela Komentar povezana s tabelo Zahteva, nam je orodje že samo izdelalo oddajanje komentarjev na strani za oddajanje zahtev. Naši uporabniki ne bodo oddajali komentarjev ob izdelovanju nove zahteve po pomoči, zato se postavimo v Postavitvi strani na Del strani AddZahteva.aspx in tam pobrišemo ZahtevaTabContainer, ki vsebuje KomentarTableControl in tako onemogočimo oddajanje komentarjev ob oddajanju zahteve. Nato poženemo aplikacijo, se prijavimo in gremo na dodajanje zahteve (slika 4.72), kjer vidimo kako smo uredili dodajanje zahtev po pomoči.



Slika 4.72: Stran za oddajanje zahtev po pomoči

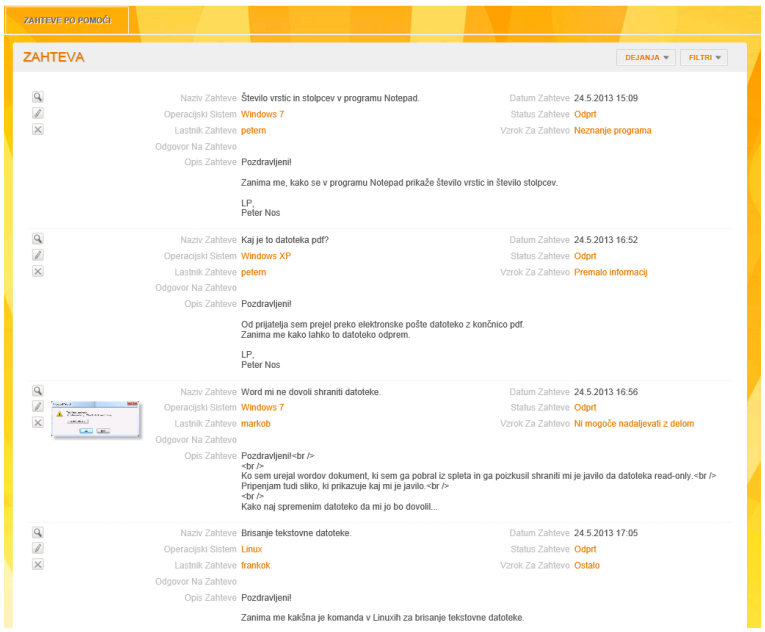
Gumb Shrani nastavimo tako, da ko bo uporabnik shranil zahtevo ga bo preusmerilo na pogled te zahteve. To naredimo tako, da se v Iron Speed-u postavimo na polje gumba SaveButton in v Lastnostih izberemo Akcije gumba (angl. Button actions), kjer vidimo, da se ob kliku na gumb Shrani shranijo podatki na trenutno prikazani stran. Tam nato kliknemo Uredi... (angl. Edit...) in izberemo 'Pojdi na specifičen URL' (s to opcijo nastavimo URL naslov pogleda shranjene zahteve) ter se pomaknemo naprej (slika 4.73) in izberemo URL naslov ShowZahteva.aspx. Kot parameter mu dodamo primarni ključ zahteve, se pomaknemo naprej in kliknemo na Zaključi. S tem smo naredili to, da se bo ob shranitvi zahteve uporabnik premaknil na pogled zahteve, ki jo je pravkar izdelal.



Slika 4.73: Izbira strani za preusmeritev

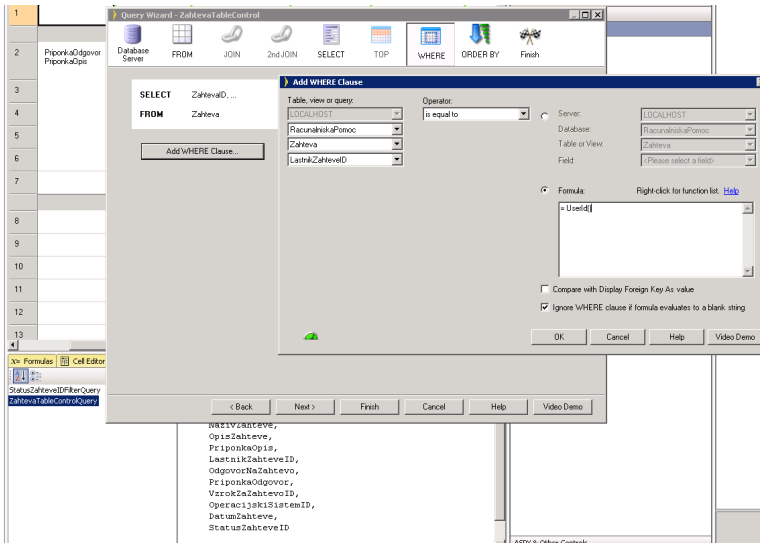
Ko pridemo iz dodajanja nove zahteve na pogled zahteve imamo tam gumb V redu. Ta gumb ima trenutno nastavljeno, da se ob kliku nanj vrne na prejšnjo stran. Če pustimo ta gumb tako kot je se bomo iz pogleda vrnili na novo dodajanje zahteve, česar pa nočemo. Zato ta gumb preuredimo tako, da se ne bo vrnil nazaj na prejšnjo stran ampak se bo vrnil na pregled zahtev po pomoči. Tako kot pri prejšnjem gumbu, napravimo preusmeritev tudi na tem gumbu in sicer preusmerimo uporabnika na pregled zahtev po pomoči, se pravi na stran ShowZahtevaTable.aspx. Ko izdelamo novo zahtevo se nato z gumbom V redu premaknemo na pregled zahtev po pomoči.

Sedaj naredimo nekaj zahtev po pomoči z našimi uporabniki in tako preverimo delovanje našega programa. Ko zaključimo z izdelovanjem zahtev (slika 4.74) vidimo, da imamo prikazane vse zahteve vseh uporabnikov, ne glede na to s katerim uporabnikom se prijavimo v aplikacijo. Zahteve moramo omejiti tako, da bodo prijavljeni uporabniki videli samo svoje zahteve, podpora pa bo videla vse zahteve po pomoči.



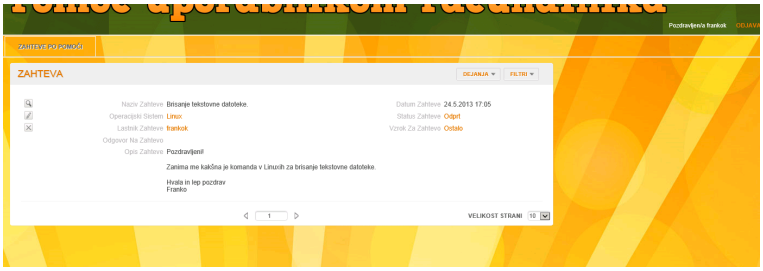
Slika 4.74: Prikaz zahtev po pomoči vseh uporabnikov

Da uporabnikom omejimo zahteve tako, da bodo videli samo svoje zahteve na preglednici, se postavimo na ShowZahtevaTableControl.aspx, tam izberemo zavihek Izvor podatkov in nato kliknemo na ZahtevaTableControlQuery. Tam najdemo SELECT stavek, ki izbere vsa polja v tabeli zahteva in jih uporabi za prikaz. Kliknemo na gumb Uredi... (angl. Edit...) in nato na zavihku WHERE kliknemo na gumb Dodaj WHERE stavek... (angl. Add WHERE Clause...) ter tam omejimo našo preglednico na uporabnike tako, da bomo pri Tabela, pogled ali poizvedba (angl. Table, view or query) izbrali 'LastnikZahteveID', pri Operaterju (angl. Operator) bomo izbrali 'is equal to' in nato pri Formuli (angl. Formula) izbrali, da je enako trenutno prijavljenemu uporabniku (slika 4.75). To naredimo tako, da z desnim klikom na polje za formulo izberemo Funkcije –>Varnost –>UserId (angl. Functions –>Security –>UserId). Kliknemo na gumb OK in se pomaknemo naprej na naslednji korak. Tam izberemo razvrščanje po datumu zahteve padajoče, da bomo imeli novejše zahteve na vrhu preglednice. Nato se postavimo naprej in kliknemo Zaključi.



Slika 4.75: Dodajanje WHERE stavka

Če sedaj poženemo aplikacijo in se prijavimo (slika 4.76), vidimo, da imamo samo zapise uporabnika z nazivom Franko Kos.

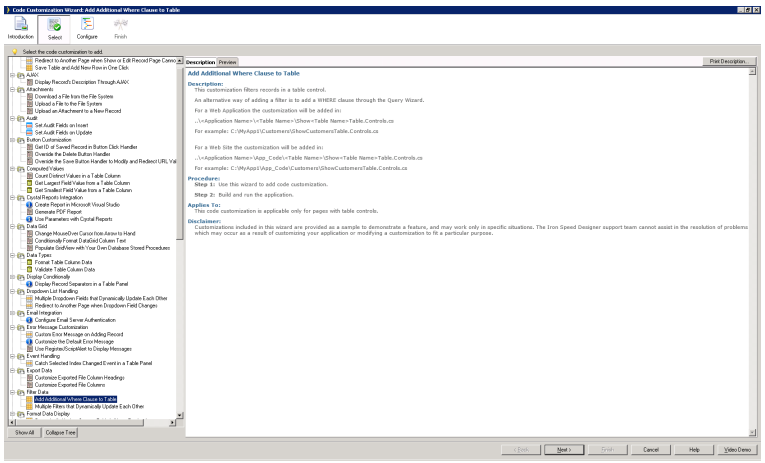


Slika 4.76: Prikaz zahtev po pomoči prijavljenega uporabnika

To je v redu, vendar če se prijavimo z uporabnikom iz podpore, ne bomo imeli nobene zahteve po pomoči na preglednici. Zato moramo dodati nekaj naše kode, ki nam bo omogočila, da bodo uporabniki podpore videli vse zapise zahtev po pomoči. Najprej pa moramo pobrisati WHERE stavek, ki smo ga pravkar izdelali.

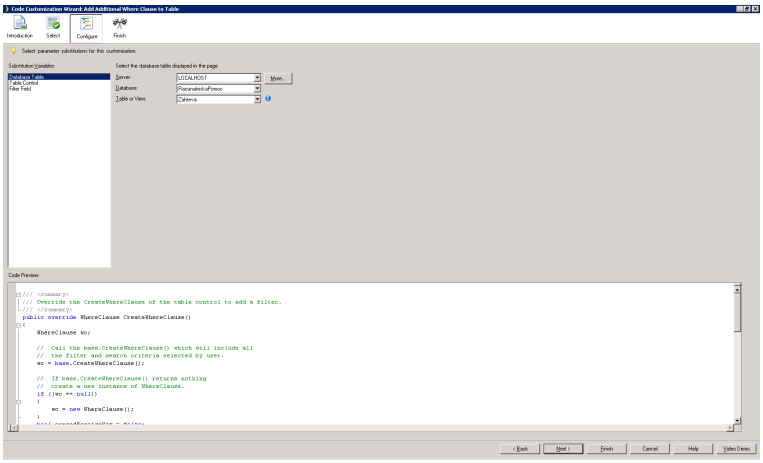
Iron Speed vsebuje Čarovnik za prilagoditev kode (angl. Code Customization Wizard), s katerim si lahko pomagamo izdelati naš košček kode, ki ga potrebujemo. V meniju Iron Speeda kliknemo na Prilagodi (angl. Customize) in nato na Čarovnik za prilagoditev kode. Odpre se nam čarovnik

(slika 4.77), v katerem se premaknemo na naslednji korak in tam pod Filtri podatkov (angl. Filter Data) izberemo Dodaj dodatni where stavek k tabeli (angl. Add Additional Where Clause to Table).



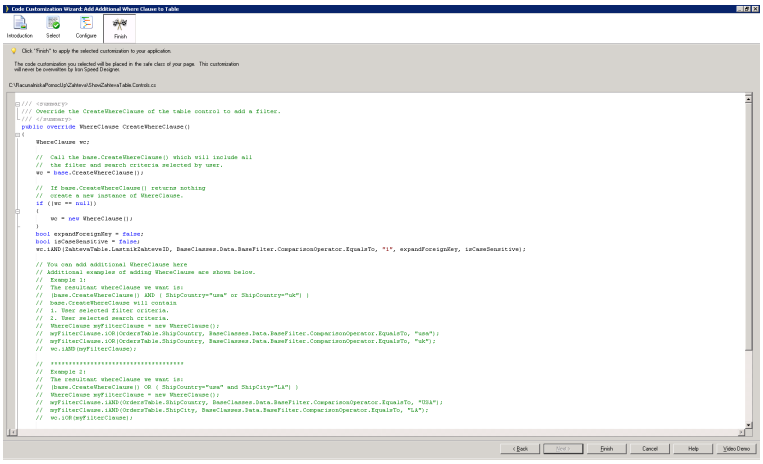
Slika 4.77: Izbiranje prilagoditve kode v Čarovniku za prilagoditev

Ko izberemo to prilagoditev se pomaknemo naprej na naslednji korak (slika 4.78), kjer določimo parametre za prilagoditev kode. Pod Tabela podatkovne baze (angl. Database Table) izberemo 'Zahteva', pod Kontrola tabele (angl. Table Control) izberemo 'ZahtevaTableControl' in pri Filtru polja (angl. Filter Field) pa izberemo 'LastnikZahteveID', ker bomo potrebovali filtriranje glede na prijavljenega uporabnika, se pravi lastnika oddane zahteve.



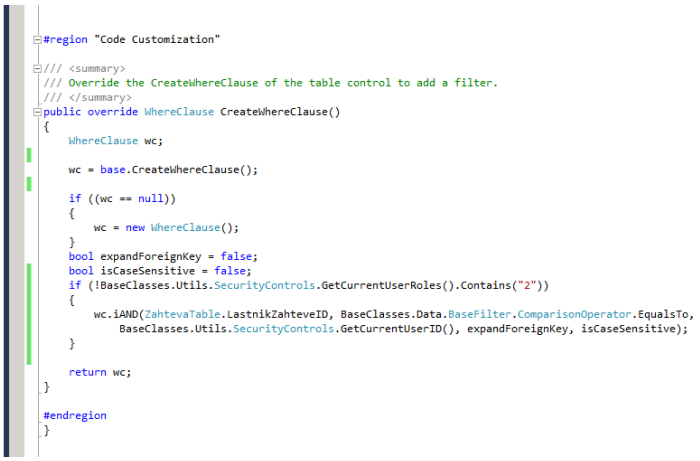
Slika 4.78: Izbiranje parametrov za prilagoditev kode

Nato se pomaknemo na zadnji korak (slika 4.79), kjer vidimo kaj nam bo Iron Speed izdelal in nato kliknemo na gumb Zaključi.



Slika 4.79: Zadnji korak Čarovnika za prilagoditev

Da bomo lažje predelali našo prilagoditev kode se premaknemo na orodje Visual Studio in tam nadaljujemo s predelavo kode. Najprej pa našo aplikacijo shranimo in zgradimo (pritisnemo na gumb Zgradi (angl. Build)) in se nato premaknemo v Visual Studio, kjer sprogramiramo naše spremembe (slika 4.80).



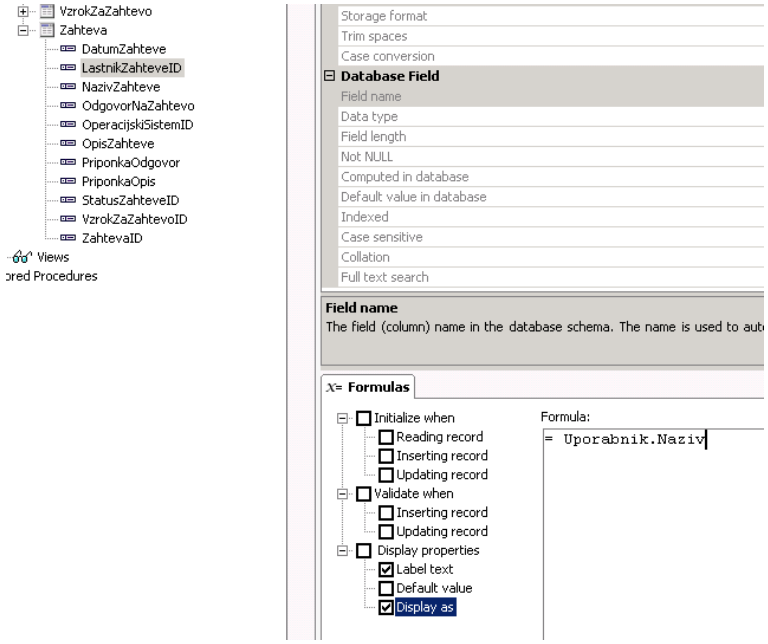
```
#region "Code Customization"

/// <summary>
/// Override the CreateWhereClause of the table control to add a filter.
/// </summary>
public override WhereClause CreateWhereClause()
{
    WhereClause wc;

    wc = base.CreateWhereClause();

    if ((wc == null))
    {
        wc = new WhereClause();
    }
    bool expandForeignKey = false;
    bool isCaseSensitive = false;
    if (!BaseClasses.Utils.SecurityControls.GetCurrentUserRoles().Contains("2"))
    {
        wc.iAND(ZahtevaTable.LastnikZahteveID, BaseClasses.Data.BaseFilter.ComparisonOperator.EqualsTo,
            BaseClasses.Utils.SecurityControls.GetCurrentUserID(), expandForeignKey, isCaseSensitive);
    }

    return wc;
}

#endregion
}
```

Slika 4.80: Prilagoditev kode za filtriranje zahtev po pomoči

Pobrišemo komentarje in ustavimo našo kodo. V naši kodi preverimo, če vloga uporabnika ne vsebuje ID-ja številka 2, ki je ID za vlogo 'Podpora uporabnikom' in če ne vsebuje tega ID-ja dodamo WHERE stavek, kjer preverja če je LastnikZahteveID enak ID-ju trenutno prijavljenega uporabnika. Če se sedaj prijavimo z podporo uporabnikom vidimo vse zahteve po pomoči in če se prijavimo z uporabnikom računalnika vidimo samo tiste zahteve, ki smo jih s tem uporabnikom oddali.

Sedaj je čas, da preuredimo preglednico zahtev po pomoči. Na podoben način kot smo predelali prikaz dodajanja nove zahteve po pomoči sedaj preuredimo preglednico zahtev po pomoči. Na prikazu preuredimo naziv zahteve, ki ga pretvorimo iz tipa Literal v LinkButton in bo nato uporabnika s klikom na gumb preusmerilo na pogled tiste zahteve, na katero je kliknil (na njen naziv). Prikažemo operacijski sistem in vzrok za zahtevo ter datum in status zahteve. Lastnika zahteve bo videla samo podpora uporabnikov, zato nastavimo pravice polja lastnik samo na uporabnike z vlogo 'Podpora uporabnikom'. Za prikaz lastnika je orodje privzeto nastavilo njegovo uporabniško ime. Mi bomo to spremenili in za prikaz lastnika nastavili njegov naziv. To naredimo tako, da se postavimo v orodju Iron Speed na zavihek Database in tam pri tabeli Zahteva kliknemo na LastnikZahteveID in v Formulah pod Lastnosti prikaza (angl. Display properties) kliknemo na Prikaži

kot (angl. Display as) in tam v vnosnem oknu za formule vpišemo Uporabnik.Naziv (slika 4.81).



Slika 4.81: Določanje uporabnikovega naziva

S tem nastavimo prikaz lastnika v tabeli Zahteva z njegovim nazivom in ne z uporabniškim imenom, kot je bilo privzeto. Sedaj moramo še na enem mestu spremeniti iz uporabniškega imena na naziv. Ko se prijavimo v aplikacijo nas pozdravi v glavi naše aplikacije kot uporabniško ime, na primer, namesto 'Pozdravljen/a Franko Kos' nas pozdravi kot 'Pozdravljen/a frankok'. To moramo sedaj spremeniti, vendar bo potrebno za to ponovno spreminjati kodo v našem programu. Tako kot prej, ko smo spreminjali kodo si lahko tudi sedaj pomagamo s Čarovnikom za prilagoditev kode. V tem primeru poiščemo Varnost in Avtentikacija (angl. Security and Authentication) in tam izberemo Shrani dodatne informacije ob času vpisa (angl. Store Extra Information at Sign-In Time) za prilagoditev naše kode. Nato se premaknemo naprej in na koraku Konfiguriraj pod Ime tabele (angl. Table name) izberemo tabelo Uporabnik in pod Poljem UserId (angl. UserId Field) izberemo UporabnikID. Nato se bomo premaknemo na končni korak in kliknemo

Zaključi. Za predelavo in izdelavo naše nove kode si ponovno pomagamo z Visual Studiem in v njem napišemo potrebno kodo (slika 4.82), ki bo shranila naziv uporabnika, ko se bomo prijavili v aplikacijo.

```
        // Write out the methods for DataSource



#region "Code Customization"

/// <summary>
/// Store information in session variable in loginSucceeded event.
/// This event is raised after user has successfully logged-in
/// Instead of creating a where string from database you might choose other
/// information to store (like time of sign in, for example).
/// </summary>
protected void SignIn_LoginSucceeded(object sender, System.EventArgs e)
{
    string userId = BaseClasses.Utils.SecurityControls.GetCurrentUserID();

    string whereStr = UporabnikTable.UporabnikID.UniqueName + " = " + "'" + userId + "'";

    BaseClasses.Data.OrderBy oB = null;

    UporabnikRecord myRecord  = UporabnikTable.GetRecord(whereStr, oB);

    string Naziv = myRecord.Naziv;

    System.Web.HttpContext.Current.Session["Naziv"] = Naziv;
}

#endregion
#endregion

"Section 2: Do not modify this section."


}
```

Slika 4.82: Shranjevanje uporabnikovega naziva v sejo

V zgornji kodi shranimo naziv uporabnika v sejo. Naziv nato pokličemo v Header.ascx. V kodi header-ja se status teksta za prijavo inicializira v metodi 'UserStatusInit()'. Mi to kodo uporabimo in jo pokličemo v regiji za prilagoditev naše kode v metodi 'LoadData()' (slika 4.83) za klicem metode 'LoadData_Base();' in jo še malo preuredimo.

```
// Code-behind class for the Header user control.
ublic partial class Header : BaseApplicationUserControl , IHeader

    #region "Section 1: Place your customizations here."

    public Header()
      {
          this.Initialize();


      }

      public void LoadData()
      {
          // LoadData reads database data and assigns it to UI controls.
          // Customize by adding code before or after the call to LoadData_Base()
          // or replace the call to LoadData_Base().
          LoadData_Base();

          switch (((BaseApplicationPage)(this.Page)).CurrentSecurity.GetUserStatus())
          {
              case null:
              case "":
                  this.UserStatusLbl.Text = GetResourceValue("Txt:UserNotSignedIn", "RacunalniskaPomocUp");
                  break;
              default:
                  this.UserStatusLbl.Text = GetResourceValue("Txt:Hello", "RacunalniskaPomocUp");
                  this.UserStatusLbl.Text = (this.UserStatusLbl.Text + Session["Naziv"]);
                  break;
          }
      }

    private string EvaluateFormula(string formula, BaseClasses.Data.BaseRecord dataSourceForEvaluate, string format, Syst
    {
        return EvaluateFormula_Base(formula, dataSourceForEvaluate, format, variables, includeDS);
    }
```

Slika 4.83: Uporaba naziva pri pozdravu uporabnika

Spremenili smo samo delček kode, tako da smo oznaki UserStatusLbl dodali tekst 'Session["Naziv"]', ki pokliče naziv, ki smo ga prej zapisali v sejo in se bo shranil ob prijavi v aplikacijo.

Če sedaj poženemo aplikacijo in se prijavimo vanjo z uporabnikom podpore (slika 4.84) vidimo, da je pregled zahtev po pomoči lepše razporejen in prikazan ter da imamo možnost klika na naziv zahteve, ki nas bo preusmeril na pogled zahteve, ki jo izberemo in vidimo tudi, da nas pozdravi z nazivom uporabnika in ne z uporabniškim imenom.

Slika 4.84: Urejen pregled zahtev po pomoči

Nato moramo izdelati možnost odgovora podpore, da bodo lahko uporabnikom odgovarjali na njihove zahteve. Vnos odgovora omogočimo na pogledu zahteve po pomoči. Tako bo podpora kliknila na zahtevo, jo prebrala in odgovorila. Zato se sedaj posvetimo preurejanju strani pogleda Zahteve po pomoči. Pogled preuredimo tako, da bodo stvari bolj pregledne in bolj uporabniško prijazne ter dodamo vnosno polje za odgovor ter dodajanje priponke in gumb za odgovor. Tukaj pustimo kar vsa polja za prikaz, tako da bodo uporabnikom vidna vsa polja, razen lastnik bo viden samo podpori uporabnikom. Odstranimo gumb za urejanje, poleg naslova Zahteva in gumb Uredi na dnu strani. Tako onemogočimo urejanje obstoječe zahteve po pomoči. Postavimo se v Del strani ZahtevaTitleRegion in v Postavitvi strani odstranimo ZahtevaDialogEditButton ter nato na Delu strani PageButtons pa preuredimo gumb Uredi v gumb Odgovori. V Lastnostih tega gumba spremenimo v Akcijah gumba ta gumb tako, da bo shranil trenutno stanje na strani in spremenimo ID in ime gumba v 'Odgovori'. Gumbu moramo nastaviti tudi varnost v lastnostih, da bo viden samo pri podpori uporabnikom. Poleg tega pa gumbu določimo tudi to, da ko bo uporabnik podpore kliknil na gumb odgovori in odgovoril na zahtevo po pomoči, se bo status zahteve spremenil na 'Rešen'. Tukaj moramo ponovno spreminjati kodo. Kodo spremenimo v metodi 'Odgovori_Click'. Za ta košček kode si bomo pomagali s Čarovnikom za prilagoditev kode tako, da pri Posodobi in dodaj podatke (angl. Update and Add Data) izberemo Posodobi zapis podatkov (angl. Update Record Data) in se pomaknemo na naslednji korak ter tam pri Tabela ali pogled (angl. Table Or View) izberemo Zahteva in se pomaknemo naprej. Na zadnjem koraku pa ne kliknemo Zaključi, ampak ta del kode v metodi 'UpdateRecord()' skopiramo in ga uporabimo v metodi 'Odgovori_Click'. Ko skopiramo kodo kliknemo Prekliči (angl. Cancel) in se postavimo v orodje Visual Studio. Postavimo se na ShowZahteva.aspx.cs in poiščemo metodo 'Odgovori_Click' v regiji Section 1, ki je regija za prilagajanje naše kode. Ta del kode iz Čarovnika za prilagoditev kode preuredimo (slika 4.85) in s tem omogočimo, da se bo ob kliku gumba Odgovor status

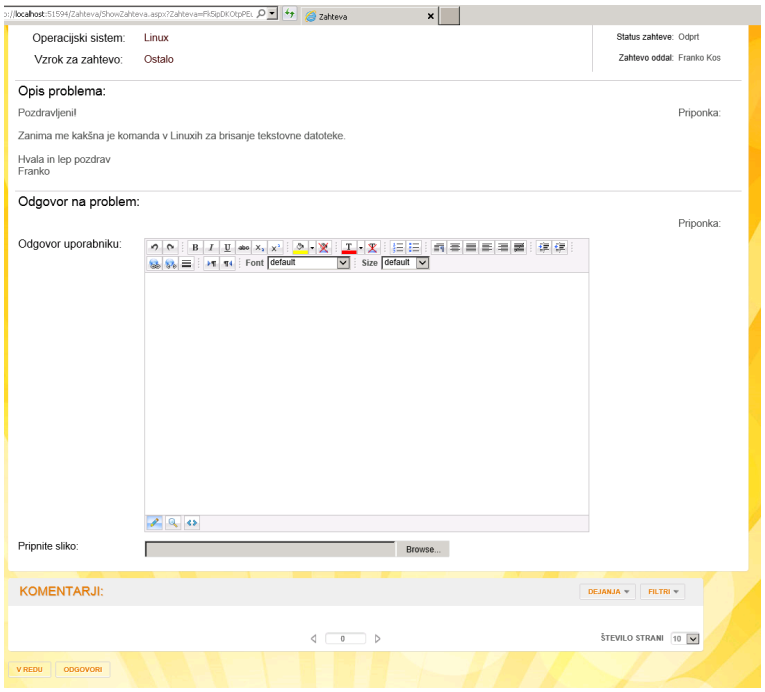
zahteve spremenil na 'Rešen'.

```
public void Odgovori_Click(object sender, EventArgs args)
        {
            string recId = ((BaseApplicationPage)(this.Page)).Decrypt(this.Page.Request.QueryString["Zahteva"]);

            KeyValue pkValue = KeyValue.XmlToKey(recId);

            string ternutnaZahtevaID = pkValue.GetColumnValueString(ZahtevaTable.ZahtevaID);

            try
            {
                DbUtils.StartTransaction();
                ZahtevaRecord rec = ZahtevaTable.GetRecord(ternutnaZahtevaID, true);
                rec.StatusZahteveID = 2;
                rec.Save();
                DbUtils.CommitTransaction();
            }
            catch (Exception ex)
            {
                DbUtils.RollBackTransaction();
                BaseClasses.Utils.MiscUtils.RegisterJScriptAlert(this, "UNIQUE_SCRIPTKEY", ex.Message);
            }
            finally
            {
                DbUtils.EndTransaction();
            }

          // Click handler for Odgovori.
          // Customize by adding code before the call or replace the call to the Base function with your own code.
          Odgovori_Click_Base(sender, args);
          // NOTE: If the Base function redirects to another page, any code here will not be executed.
        }
#endregion
```

Slika 4.85: Sprememba statusa ob kliku na gumb Odgovori

Pri kodi, ki smo jo uporabili smo kar dosti preuredili. ID zahteve pobe-remo iz URL naslova in ga nato uporabimo za pridobitev zahteve po pomoči, ki jo trenutno obdelujemo. Nato spremenimo ID statusa zahteve na 2 in nato shranimo ta ID v zahtevo po pomoči, ki smo jo obdelovali.

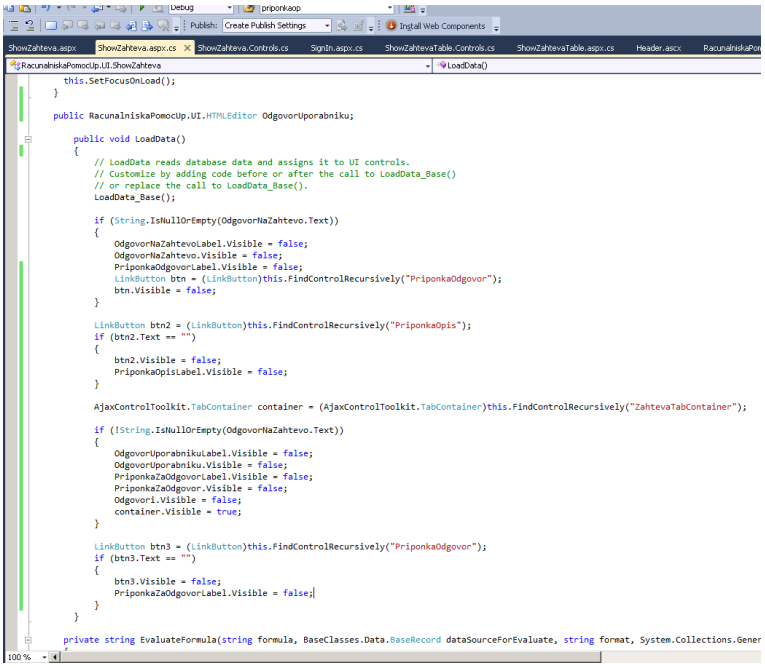
Poljem za odgovor in priponko določimo, da bodo vidna samo podpori uporabnikom tako, da spremenimo varnost na vlogo 'Podpora uporabni-kom'. Na pogledu razporedimo naša polja in jim spremenimo obliko ter dodamo ločilne črte, tako da bo pogled preglednejši. Če poženemo aplikacijo (slika 4.86) vidimo, da je pogled zahteve po pomoči spremenjen tako, da so polja lepše razporejena in bolj pregledna ter da je dodano vnosno tekstovno polje za odgovor in priponko.



Slika 4.86: Pogled zahteve po pomoči

Na zgornji sliki pa še vedno ni vse narejeno tako kot smo si zamislili. Tekst 'Priponka:', pri opisu, bo viden samo če bo priponka obstajala in pri odgovoru enako. Odgovor na problem in priponka odgovora pa bosta vidna šele takrat, ko bo podpora odgovorila uporabnikom. Ko bo podpora odgovorila uporabnikom, se bodo polja za vnos odgovora in pripenjanje priponke skrila. Komentarje pa bodo lahko uporabniki dodajali šele takrat, ko bo oddan odgovor na zahtevo. Zato bodo komentarji skriti, ko odgovor še ne bo oddan.

Vidljivosti nastavimo v kodi, zato se postavimo na orodje Visual Studio in tam v ShowZahteva.aspx.cs v metodi 'LoadData()' napišemo kodo (slika 4.87) in nastavimo vidljivost polj pri pogledu zahteve po pomoči.



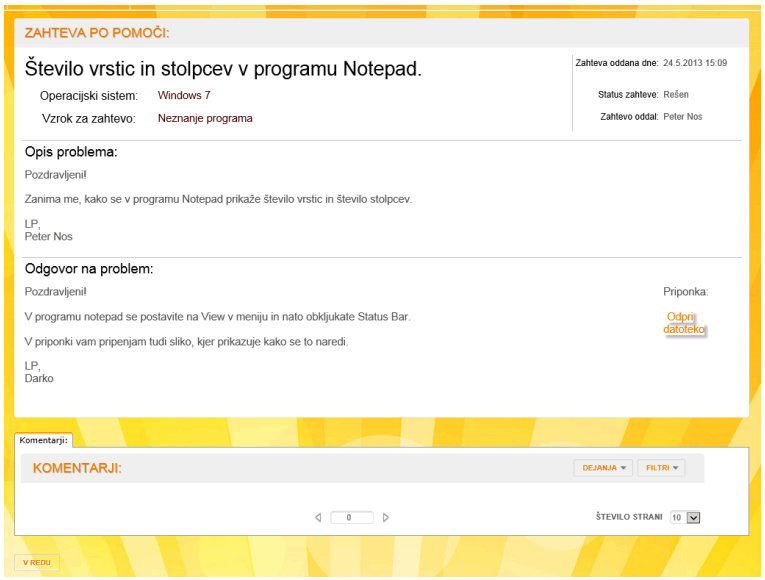
Slika 4.87: Nastavitev vidljivosti polj pogleda zahtev po pomoči

V kodi napišemo to, da če je odgovor na zahtevo prazen, se nam polja za odgovor na zahtevo in priponko ne prikažejo. Preverimo če obstaja gumb za odpiranje priponke in če ne obstaja skrijemo tudi oznako priponke. V Iron Speed-u se postavimo na ShowZahteva.aspx in v Delu strani ShowZahteva.aspx označimo ZahtevaTabContainer v njegovih Lastnostih nastavimo Vidljivost na 'False' in s tem onemogočimo vidljivost komentarjev. V kodi nato preverimo, če je polje za odgovor na zahtevo izpolnjeno in nato zakrijemo tekst za vnosno polje in priponko ter vnosno polje in priponko. Skrijemo gumb Odgovori in postavimo Vidljivost komentarjev na 'true' in s tem prikažemo komentarje, ko bo odgovor oddan. Preverimo tudi, če obstaja priponka za odgovor, in če ne obstaja skrijemo tudi tekst za priponko ob odgovoru.

Sedaj testiramo vidljivost tako, da pogledamo kaj vidimo, ko smo prijavljeni z uporabnikom z vlogo 'Uporabnik računalnika' in kaj vidimo ko smo prijavljeni z uporabnikom z vlogo 'Podpora uporabnikom'. Nato oddamo en odgovor na zahtevo in preverimo ali so vidni komentarji in polja za odgovor

šele ko je odgovor oddan.

Če nato poženemo aplikacijo in oddamo odgovor (slika 4.88) z uporabnikom iz podpore, vidimo da se nam je vnos za odgovor in priponko skril ter da so se nam prikazali komentarji. Lahko opazimo tudi, da se nam je status zahteve spremenil na 'Rešen'.



Slika 4.88: Pogled zahtev po pomoči z oddanim odgovorom

V nadaljevanju se bomo posvetili oddajanju komentarjev na zahtevo po pomoči. Uporabniki bodo lahko oddajali komentarje. Če bo uporabnik z vlogo 'Uporabnik računalnika' oddal komentar, bomo nastavili, da se bo status zahteve ponovno nastavil na 'Odprt' in ko bo uporabnik z vlogo 'Podpora uporabnikom' oddal komentar se bo status nastavil na 'Rešen'.

Najprej dodamo gumb za dodajanje komentarjev ter odstranimo naslov KOMENTARJI in spustna seznama DEJANJA in FILTRI. Tako bo uporabnik imel na voljo samo gumb za dodajanje komentarjev. Nato nastavimo gumbu za dodajanje komentarjev preusmeritev na stran AddKomentar.aspx in mu v URL naslovu podamo še primarni ključ zahteve po pomoči.

Nato bomo morali preurediti obliko in funkcionalnosti oddajanja komentarjev tako, da se postavimo v Iron Speed-u na stran AddKomentar.aspx in

preuredimo obliko oddajanja komentarjev, da bo uporabniško bolj prijazna ter ji dodamo funkcionalnosti. Ime naslova spremenimo v Komentar na zahtevo. Vidna bodo samo polja za oddajanje vsebine komentarja in priponka komentarja, zato ostala polja skrijemo in izbrišemo AddRecordLink gumbe ob spustnih seznamih. Polju ZahtevaKomentarjaID dodamo kar primarni ključ zahteve, ki ga imamo v URL naslovu. To naredimo tako, da povozimo metodo 'SetZahtevaKomentarjaID()' in tej v metodi napišemo našo kodo. Postavimo se v orodje Visual Studio in tam v AddKomentar.Controls.cs kjer napišemo našo kodo (slika 4.89). Še prej, pa v orodju Iron Speed spremenimo polju ZahtevaKomentarjaID Tip kontrole (angl. Control type) iz 'DropDown' na 'TextBox', ker bomo tako lažje povozili kodo v metodi 'SetZahtevaKomentarjaID()'.

```
#region "Section 1: Place your customizations here."


public class KomentarRecordControl : BaseKomentarRecordControl
{

        // The BaseKomentarRecordControl implements the LoadData, DataBind and other
        // methods to load and display the data in a table control.

        // This is the ideal place to add your code customizations. For example, you can override the LoadData,
        // CreateWhereClause, DataBind, SaveData, GetUIData, and Validate methods.


    public override void SetZahtevaKomentarjaID()
    {
        string recId = ((BaseApplicationPage)(this.Page)).Decrypt(this.Page.Request.QueryString["Zahteva"]);

        if (!String.IsNullOrEmpty(recId))
        {
            KeyValue pkValue = KeyValue.XmlToKey(recId);

            string trenutnaZahtevaID = pkValue.GetColumnValueString(ZahtevaTable.ZahtevaID);

            this.ZahtevaKomentarjaID.Text = trenutnaZahtevaID;
        }
        else
        {

            // ZahtevaKomentarjaID is NULL in the database, so use the Default Value.
            // Default Value could also be NULL.

            this.ZahtevaKomentarjaID.Text = KomentarTable.ZahtevaKomentarjaID.DefaultValue;
        }
    }
}
```

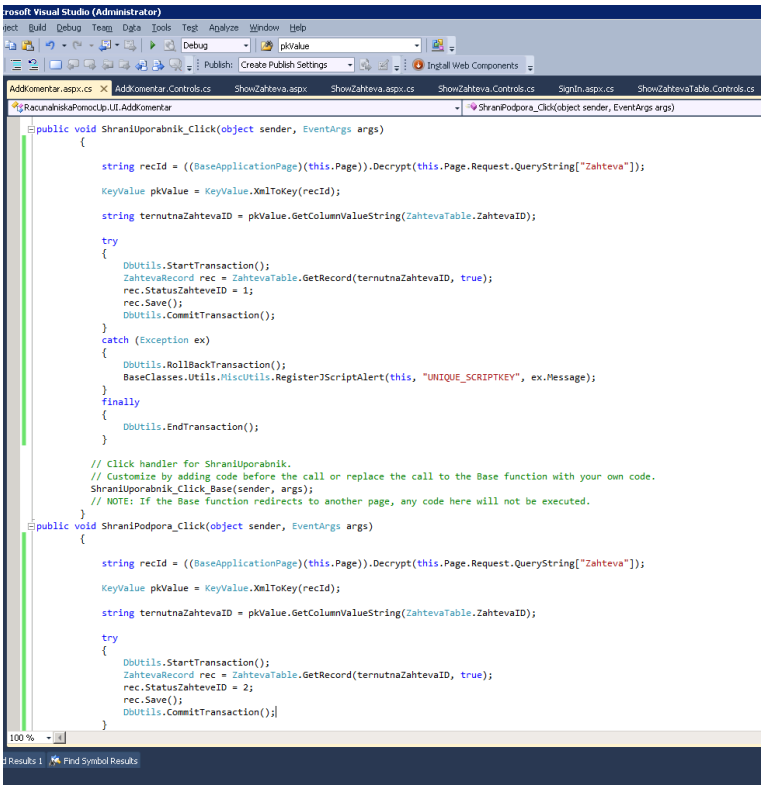
Slika 4.89: Dodajanje primarnega ključa zahteve v komentar zahteve

V kodi preberemo iz URL naslova ID zahteve, na katero bomo oddali komentar in nato ta ID zapišemo v ZahtevaKomentarjaID, da ko oddamo komentar vemo h kateri zahtevi po pomoči komentar spada. To naredimo zato, da ne bodo imele vse zahteve po pomoči vseh komentarjev ampak samo svoje komentarje. Polju UporabnikKomentarjaID dodelimo ID trenutno prijavljenega uporabnika tako, da kar v Formulah pri Inicializiraj ob dodajanju zapisa

(angl. Initialize when Adding record) napišemo 'UserId()'. Pri polju za datum, pa nastavimo shranjevanje trenutnega datuma tako, da v Formulah pri Inicializiraj ob dodajanju zapisa napišemo 'Now()'.

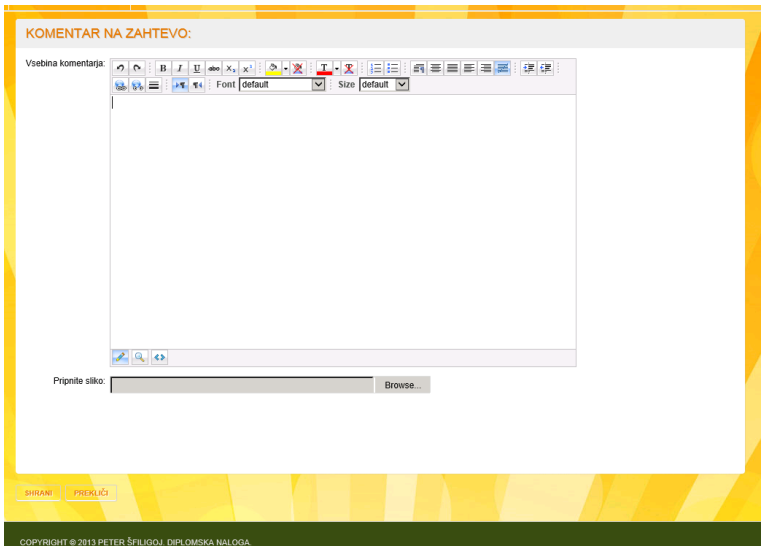
Nato moramo nastaviti spreminjanje statusa zahteve ob shranjevanju. Najprej odstranimo gumb 'Shrani in nov', ker ga v našem primeru ne potrebujemo. Nato dodamo še en gumb Shrani v polje kjer je že gumb za shranjevanje in en gumbov ID poimenujemo 'ShraniUporabnik', drugega pa 'ShraniPodpora'. Gumbu 'ShraniUpoarbnik' nastavimo vidljivost tako, da ga bodo videli samo uporabniki z vlogo 'Uporabnik računalnika' in gumb ShraniPodpora bodo videli samo uporabniki z vlogo 'Podpora uporabnikom'.

Za spreminjanje statusa uporabimo takšno kodo kot pri kliku na gumb Odgovori. Zato se premaknemo v Visual Studio, kjer povozimo in spremenimo kodo metodam 'ShraniUporabnik_Click' in 'ShraniPodpora_Click' (slika 4.90). Spremenimo le ID statusa zahteve na 1, ko bo uporabnik oddajal komentar. Tudi zaradi tega dela kode, smo potrebovali ID zahteve, ki smo ga podali v URL naslovu.



Slika 4.90: Nastavljanje ID-ja statusa zahteve ob shranitvi komentarja

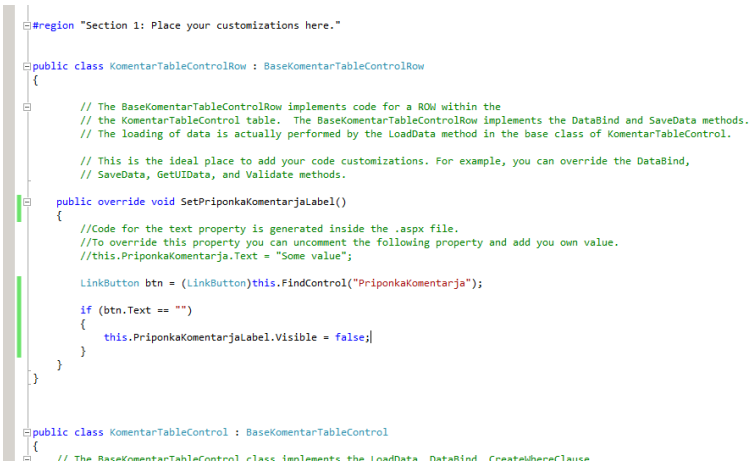
Nato poženemo aplikacijo (slika 4.91) in vidimo kako zgleda oddajanje komentarjev. Imamo vnos teksta vsebine komentarja in polje za pripenjanje slike ob komentarju. Gumb Shrani je odvisen od vloge prijavljenega uporabnika.



Slika 4.91: Oddajanje komentarjev na zahtevo po pomoči

Preverimo spreminjanje statusa zahteve in oddamo nekaj komentarjev iz strani uporabnikov z različnimi vlogami. Vidimo, da se status zahteve spreminja. Sedaj nam preostane še preoblikovanje pogleda komentarjev, da bodo komentarji bolj pregledni ter lepši in bo pregledovanje komentarjev uporabnikom prijaznejše.

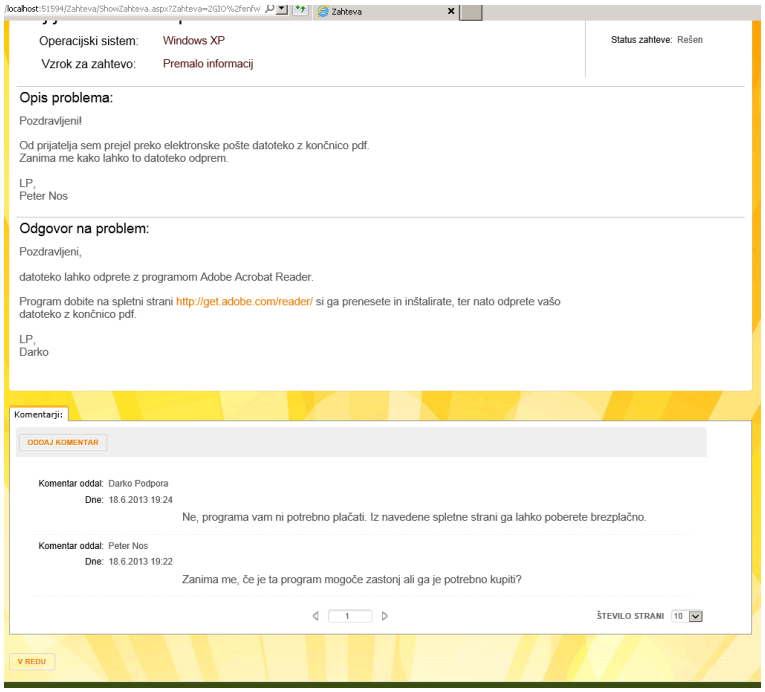
Odstranimo gumbe za urejanje in brisanje komentarjev. Preuredimo postavitev polj in spremenimo obliko kjer je potrebno. Odpiranje priponke naredimo tako kot pri priponkah opisa in odgovora. V zavihku Baza podatkov (angl. Database) označimo tabelo Komentar in nato stolpec UporabnikKomentarjaID ter v Formulah pod Prikaži kot (angl. Display as) zapišemo 'Uporabnik.Naziv', da se bo namesto uporabniškega imena izpisal naziv uporabnika, ki odda zahtevo. V kodi moramo nastaviti Vidljivost teksta 'Priponka:' na 'false', da se ne bo videl, če priponka ne bo obstajala v komentarju. To naredimo podobno kot smo naredili pri opisu in odgovoru. Postavimo se v orodje Visual Studio in tam v ShowZahteva.Controls.cs povozimo metodo 'SetPriponkaKomentarjaLabel()' (slika 4.92) in skrijemo tekst Priponka, če priponka ne obstaja.



```
#region "Section 1: Place your customizations here."


public class KomentarTableControlRow : BaseKomentarTableControlRow
{

        // The BaseKomentarTableControlRow implements code for a ROW within the
        // the KomentarTableControl table.  The BaseKomentarTableControlRow implements the DataBind and SaveData methods.
        // The loading of data is actually performed by the LoadData method in the base class of KomentarTableControl.

        // This is the ideal place to add your code customizations. For example, you can override the DataBind,
        // SaveData, GetUIData, and Validate methods.

    public override void SetPriponkaKomentarjaLabel()
    {
        //Code for the text property is generated inside the .aspx file.
        //To override this property you can uncomment the following property and add you own value.
        //this.PriponkaKomentarja.Text = "Some value";

        LinkButton btn = (LinkButton)this.FindControl("PriponkaKomentarja");

        if (btn.Text == "")
        {
            this.PriponkaKomentarjaLabel.Visible = false;
        }
    }
}


public class KomentarTableControl : BaseKomentarTableControl
{
    // The BaseKomentarTableControl class implements the LoadData, DataBind, CreateWhereClause
```

Slika 4.92: Nastavljanje vidljivosti teksta 'Priponka:'

Komentarje razvrstimo po ID-ju komentarja padajoče. Tako bodo novejši komentarji prikazani na vrhu. V orodju Iron Speed se na strani ShowZahteva.aspx postavimo na zavihek Izvor podatkov in tam pri KomentarTableControlQuery kliknemo na Uredi... in v zavihku ORDER BY izberemo za sortiranje polje 'DatumKomentarja' padajoče (angl. descending). Nato poženemo aplikacijo in vidimo (slika 4.93) kako so oblikovani naši komentarji in da so razvrščeni tako, da je novejši komentar na vrhu.



Slika 4.93: Prikaz oddanih komentarjev v aplikaciji

Nato moramo še malo preurediti pregled zahtev po pomoči. Če se bo prijavil uporabnik z vlogo 'Uporabnik računalnika', bo prišel na pregled zahtev po pomoči, ki bodo imele status 'Rešen', če se bo pa prijavil uporabnik z vlogo 'Podpora uporabnikom', bo prišel na pregled zahtev po pomoči, ki bodo imele status 'Odprt'. Seveda bodo lahko uporabniki videli tudi Odprte, Rešene ali Vse zahteve, tako da si bodo spremenili filter zahtev. Za to funkcionalnost bomo morali zapisati nekaj naše kode. Postavimo se v Visual Studio in v ShowZahtevaTable.Controls.cs dodamo nekaj kode kar v metodo 'CreateWhereClause()', ki smo jo izdelali za pregled zahtev po pomoči (slika 4.94).

```
#region "Code Customization"

/// <summary>
/// Override the CreateWhereClause of the table control to add a filter.
/// </summary>
public override WhereClause CreateWhereClause()
{
    WhereClause wc;

    wc = base.CreateWhereClause();

    if ((wc == null))
    {
        wc = new WhereClause();
    }
    bool expandForeignKey = false;
    bool isCaseSensitive = false;

    if (!BaseClasses.Utils.SecurityControls.GetCurrentUserRoles().Contains("2"))
    {
        wc.iAND(ZahtevaTable.LastnikZahteveID, BaseClasses.Data.BaseFilter.ComparisonOperator.EqualsTo,
            BaseClasses.Utils.SecurityControls.GetCurrentUserID(), expandForeignKey, isCaseSensitive);

        this.PopulateStatusZahteveIDFilter("2", 500);
        this.DataChanged = true;
    }
    else
    {
        this.PopulateStatusZahteveIDFilter("1", 500);
        this.DataChanged = true;
    }

    return wc;
}

#endregion
}
```

Slika 4.94: Nastavljanje filtrov preglednice zahtev po pomoči

V kodi spreminjamo ID filtra glede na prijavljenega uporabnika. Če je prijavljen uporabnik z vlogo 'Uporabnik računalnika' spremenimo filter na '2' in s tem prikažemo rešene zahteve po pomoči. Če pa je prijavljen uporabnik z vlogo 'Podpora uporabnikom' pa spremenimo filter na '1' in s tem prikažemo odprte zahteve po pomoči. Uporabniki si pa lahko pogled zahtev po pomoči drugače prikažejo tako, da spremenijo filter na 'Odprt', 'Rešen' ali 'Vsi'.

Za konec še poženemo aplikacijo in pogledamo kako deluje pregled zahtev po pomoči, z različnimi uporabniki.

S tem smo zaključili z našim prototipom spletne aplikacije za zahteve po pomoči. Še enkrat se sprehodimo skozi aplikacijo in pregledamo, če vse deluje tako kot je potrebno in po potrebi še kakšne malenkosti spremenimo, dodamo ali odstranimo.

Prototip aplikacije za zahteve po pomoči je zaključen in imamo sedaj na voljo delujoč prototip, kjer uporabniki oddajajo zahteve po pomoči o uporabi računalnika in računalniških programov.

V nadaljevanju poglavja pa bomo za konec našteli še prednosti in slabosti tega orodja.

4.3 Prednosti in slabosti orodja Iron Speed

Tako kot vsako orodje ima tudi orodje Iron Speed svoje prednosti in svoje slabosti. V tem podpoglavju bomo predstavili nekatere prednosti in slabosti orodja Iron Speed v tabeli 4.2.

Prednosti	**Slabosti**
Izdelovanje aplikacij s tem orodjem nam lahko prihrani zelo veliko časa, ker imamo z orodjem možnost na zelo hiter način vzpostaviti celotno ogrodje spletne aplikacije. Zelo veliko časa nam prihrani tudi, če moramo več spletnih aplikacij začeti delati znova, od začetka.	Potrebno je dobro znanje o načrtovanju podatkovne baze, ker je orodje podatkovno usmerjeno (angl. database driven tool). Razvijalci aplikacije morajo imeti podatkovno bazo že izdelano ali pa morajo dobro vedeti kako se podatkovno bazo izdela.
Večinski del aplikacije nam izdela samo na podlagi podatkovne baze. Iz podatkovne baze nam orodje zelo lepo izdela objektno orientirane razrede ter strani za vzdrževanje in sprotno zaznava spremembe v podatkovni bazi. Na podlagi sprememb podatkovne baze ustrezno posodobi osnovne razrede.	Orodje včasih generira preveč zaslonov iz podatkovne baze. Iron Speed poskuša biti predvidljiv, vendar včasih malo pretirava s preslikavo relacij.
Iron Speed je izredno stabilen generator kode. Koda, ki jo orodje ustvari izhaja iz dobrih praks uporabe, je dobro strukturirana, zelo berljiva in razširljiva. Mogoče jo je enostavno vzdrževati. Koda, ki je zgrajena ni skrita razvijalcem. Vso kodo aplikacije lahko razvijalci vidijo ter jo je mogoče enostavno prilagoditi.	Za kompleksnejše prilagoditve in razhroščevanje aplikacije je bolj priporočljiva uporaba orodja Visual Studio. Tudi z orodjem Iron Speed je večino mogoče narediti, vendar je to veliko težje kot v orodju Visual Studio.

Orodje se lahko poveže skoraj z vsemi sistemi za upravljanje podatkovnih baz (angl. database managment system, krat. DBMS). Iron SPeed je tesno integriran z zalednim delom sistema baze podatkov (angl. back-end database). Med aplikacijo in podatki je zelo tesna povezanost. Podatkovna plast (angl. data layer) je lahko konfigurirana za uporabo kar direktno iz SQL-a ali pa iz baznih procedur (angl. stored procedures).	Orodje izdela zelo nestabilne strani, če mu podamo slabo izdelano podatkovno bazo. Mogoče to ni ravno slabost tega orodja, je pa dobro, če imamo podatkovno bazo dobro zgrajeno in povezano, sicer orodje ne bo znalo izdelati vsega tako kot je potrebno. Če je podatkovna baza slabo načrtovana, bo razvijalec moral narediti še kar nekaj dodatnega dela, da bo dokončal svojo spletno stran in da bo funkcionirala tako kot si je zamislil.
Robustno integrirano razvojno okolje (angl. integrated development enviorment, krat. IDE), dobra dokumentacija, velika izbira motivov ter prilagajanje obstoječih ali ustvarjanje novih motivov. Orodje vsebuje profesionalno oblikovane predloge, ki naredijo aplikacijo pregledno in videz aplikacije je uporabniško prijazen. Dokumentacija orodja je dobro napisana in je zelo pregledna. Orodje tudi vsebuje že veliko prilagoditev, s katerimi si lahko pomagamo in izpopolnimo našo aplikacijo. Lokalizacija programa in validacije v orodju so zelo dobro urejene.	Ko pridemo čez osnove orodja postane krivulja učenja zelo strma (angl. steep learning curve). Za obvladovanje pravil in operacij orodja je potrebno vložiti kar veliko časa. Nekatere opcije in funkcije so zelo intuitivne, za nekatere je pa potrebno vložiti kar nekaj časa, da jih najdemo in da se na njih privadimo. Orodje vsebuje skoraj vse kar potrebujemo, vendar je potreben čas, da odkrijemo vse njegove funkcije in zmogljivosti. Potrebno je kar nekaj časa kopati v orodje in odkrivati njegove funkcionalnosti.

Obstajajo dobri spletni viri za pomoč ter podporo. Na spletu je veliko filmov z raznimi vajami (angl. video tutorials) in prikazovanjem funkcionalnosti ter zmogljivosti Iron Speed-a. Spletni forumi za odpravljanje težav so zelo učinkoviti, odgovore na vprašanja dobimo zelo hitro in v skupnosti je veliko ljudi, ki nam lahko na katerokoli vprašanje z veseljem odgovorijo.	Model kode, ki ga orodje ustvari je zelo velik in za nekatere mogoče zastrašujoč. To pa pomeni tudi, da obstaja zelo veliko napisanih rutin za upravljanje običajnih nalog. Tako da je mogoče vredno, da si vzamemo veliko časa za čim večje razumevanje modela.
Enostavna avtentikacijsko – avtorizacijska (angl. authentication – authorization) infrastruktura za obrazce ali uporaba varnosti na podlagi Aktivnega Direktorija (angl. Active Directory-based security).	Mogoče je orodje nekoliko zastrašujoče, ker ima zelo veliko opcij in možnosti.
Za manj zahtevne aplikacije lahko orodje uporabljajo tudi osebe brez programerskega znanja. Tudi če nimamo programerskega znanja, imamo možnost izgradnje lepih spletnih strani.	
Orodje izdela aplikacije z bogatim grafičnim vmesnikom (angl. graphical user interface, krat. GUI), paginacijo, menije, poročila, izvoz v Word, Excel, itd. V orodju je tudi mogoče dodati poljubno število ASP.NET kontrol tretjih oseb (angl. third-party ASP.NET controls).	

Orodje je zelo hitro in učinkovito pri CRUD operacijah (CRUD je angleška kratica za Create, Read, Update and Delete ali po slovensko Ustvarjanje, Branje, Posodabljanje in Brisanje. Včasih poimenovana tudi kot SCRUD, črka S naj bi pomenila Search ali po slovensko Iskanje).	
Fleksibilno licenciranje. Na primer, mesečno licenco je mogoče preklicati med tem, ko ne uporabljamo orodja za izdelavo aplikacije ter jo ponovno aktivirati takrat, ko ponovno potrebujemo preurejanje naše aplikacije.	
Enostavno .NET spletno aplikacijo je mogoče postaviti v enem samem delovnem dnevu. Orodje je zelo primerno tudi za izdelavo prototipov.	
Mogoče je orodje dobro tudi za učenje jezika C# ali Visual Basic, strukturiranega programiranja, spletnih aplikacij, html-ja in načrtovanja podatkovne baze ob proizvajanju demonstracijske kode z minimalnim naporom.	

Tabela 4.2: Prednosti in slabosti orodja Iron Speed

Iron Speed lahko marsikomu pomaga pri izdelovanju spletnih aplikacij ter prihrani veliko časa pri izdelavi začetnega ogrodja spletne aplikacije. Orodje pride predvsem v poštev za manjše ekipe ter za manjše projekte. Če razvojna

ekipa izdeluje manjše projekte, jih lahko zelo hitro izdela in prikaže njihovim strankam, lahko tudi že v enem samem delovnem dnevu. Projekte lahko izdelujejo kar po istem postopku in samo zahteve strank spreminjajo sproti in s tem prihranijo veliko časa, ker jim ni potrebno vsake aplikacije izdelovati znova, ker jim večino funkcionalnosti že Iron Speed sam izdela. Z orodjem je mogoče izdelati tudi ogromne projekte, vendar je za ogromne projekte bolj priporočljivo, da se začne izdelovati projekt iz nule, brez orodja za hiter razvoj aplikacij, ker imamo tako večji nadzor nad kodo in celotnim projektom.

Poglavje 5

Sklepne ugotovitve

V okviru diplomske naloge sem predstavil izdelavo prototipov programske opreme ter zakaj je prototipiranje aplikacij pomembno. Za hitrejšo izdelavo prototipov programske opreme je na voljo metodologija hitrega razvoja aplikacij, ki dela na tem, da je izdelovanje prototipov hitrejše. Tako sem predstavil tudi to metodologijo in njene značilnosti. Na voljo so orodja za hiter razvoj aplikacij, ki omogočajo izdelovalcem hitrejšo izdelavo ogrodja in grafičnih vmesnikov spletne aplikacije, zato sem v okviru diplomske naloge predstavil tudi orodje za hiter razvoj aplikacij po imenu Iron Speed Designer in predstavil njegove značilnosti ter funkcionalnosti. Iron Speed na podlagi podatkovne baze izdela ogrodje spletne aplikacije, grafični vmesnik, navigacijo, gumbe in razne funkcionalnosti.

Z omenjenim orodjem sem nato izdelal prototip spletne aplikacije, s pomočjo katere lahko uporabniki oddajajo zahteve po pomoči z vprašanji o računalniku in računalniških programih. Spletna aplikacija za pomoč uporabnikom računalnika bi pomagala manj izkušenim uporabnikom računalnika pri pridobitvi dodatnega znanja in izkušenj na računalniškem področju ter jim pomagala reševati težave ob uporabi raznih programov.

Prototip spletne aplikacije, ki sem ga izdelal je samo prikaz kako naj bi aplikacija delovala. Preden bi aplikacijo postavili na produkcijo bi jo bilo mogoče smiselno še malo dodelati, vendar je mogoče za prvo različico aplika-

cija že dovolj dobra tudi za postavitev na produkcijo. V okviru diplomskega dela je mogoče spletno aplikacijo preizkusiti samo lokalno.

Danes se na trgu pojavlja vedno več povpraševanja o spletnih aplikacijah, zato je za manj kompleksnejše aplikacije dobro uporabljati orodja za hiter razvoj aplikacij, ker nam prihranijo veliko časa pri izdelavi začetnega ogrodja aplikacije in lahko tako izdelamo več spletnih aplikacij v krajšem času.

Slike

Literatura

[1] (2009) SDLC Models. Dostopno na:
http://www.onestopqa.com/resources/SDLC

[2] (2009) Software Prototyping. Dostopno na:
https://www.classle.net/book/software-prototyping#

[3] A. T. Sadabadi, N. M. Tabatabaei. "Rapid prototyping for software projects with user interfaces", *Department of Computer Systems and Informatics Seraj Higher Education Institute, Iran*, št. 9, zv. 2, str. 85–90, 2009.

[4] M. F. Smith. *Software Prototyping*, London: McGraw-Hill, 1991.

[5] What is prototyping approach of system development. Explain the process of prototyping in detail and discuss the advantages of using prototyping approach. Dostopno na:
http://www.taleem-e-pakistan.com/a-i-o-u/mba_assignments_aiou/management_informantion_systems_solved_assignemts/m_i_s_solved_assignment_1_mba_aiou_2b.html

[6] (2013) Prototyping. Dostopno na:
http://www.interaction-design.org/encyclopedia/prototyping.html

[7] (2007) Top 10 Simulation Tools for UI Designers, Information Architects and Usability Specialists. Dostopno na:
http://uidesign-usability.blogspot.com/2007/03/top-10-simulation-tools-for-ui.html

[8] (2006) Visio Replacement? You Be the Judge. Dostopno na: http://boxesandarrows.com/visio-replacement-you-be-the-judge/

[9] (2007) How Simulation Software Can Streamline Application Development. Dostopno na: http://www.cio.com/article/print/28501

[10] (2013) Gui Prototyping Tools. Dostopno na: http://c2.com/cgi/wiki?GuiPrototypingTools http://www.cio.com/article/print/28501

[11] Luqi Ketabchi, "A computer-aided prototyping system", *IEEE Software*, št. 5, zv. 2, str. 66–72, 1988.

[12] Whitten, Barlow, Bentley, *System Analysis and Design Method*, London: McGraw-Hill, 1997.

[13] (2008) Rapid Application Development. Dostopno na: http://www.corepartners.com/pdf/rad.pdf

[14] (2013) Rapid application development. Dostopno na: http://www.webopedia.com/TERM/R/Rapid_Application_Development.html

[15] (2012) Warehouse Management System. Dostopno na: http://www.scribd.com/doc/109600087/Warehouse-Management-System

[16] (2005) Rapid Application Development. Dostopno na: http://www.blueink.biz/RapidApplicationDevelopment.aspx

[17] (2013) Rapid Application Development. Dostopno na: http://prttek.com/index.php?option=com_content&view=article&id=78&Itemid=76

[18] (2005) What is Rapid Application Development?. Dostopno na: http://www.casemaker.com/download/products/totem/rad_wp.pdf

[19] (2006) Rapid application development. Dostopno na:
http://www.hit.ac.il/staff/leonidm/information-systems/ch32.html

[20] (2012) The Advantages and Disadvantages of RAD Software Development. Dostopno na:
http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-rad-software-development.html

[21] (2012) Advantages and Disadvantages of Rapid Application Development. Dostopno na:
http://msdnnepal.net/blogs/pranita/archive/2012/01/24/advantages-and-disadvantages-of-rapid-application-development-rad.aspx

[22] (2013) Process/Project RAD – RAD – Rapid Application Development Process. Dostopno na:
http://www.projectmanagement.com/content/processes/11306.cfm

[23] (2013) Iron Speed. Dostopno na:
http://www.ironspeed.com/company/

[24] (2010) Iron Speed adds SharePoint backing to database app tool. Dostopno na:
http://www.infoworld.com/d/developer-world/iron-speed-adds-sharepoint-backing-database-app-tool-673