Elektrotehniško-računalniška strokovna šola in gimnazija Ljubljana
Ljubljana, Vegova 4

Seminarska naloga pri splošni maturi

Informacijski sistem

E-redovalnica

April 2011

Luka Zakrajšek, R 4. B

Mentorica: Tea Lončarič, prof.

Povzetek

V seminarski nalogi bom opisal izdelavo informacijskega sistema za spletno redovalnico. Naloga je najprej analizirana. Nato je opisano načrtovanje podatkovnega modela. Iz tega sledi načrtovanje diagrama toka podatkov. Zadnje načrtujemo še uporabniški vmesnik, kjer aplikacijo razdelimo na posamezne strani za različne tipe uporabnikov. Opisana je uporaba orodij za načrtovanje podatkovnega modela, procesov, programski jezik, spletno ogrodje, IDE in sama izvedba aplikacije. Na koncu sledi še testiranje aplikacije.

Ključne besede: informacijski sistem, e-redovalnica, podatkovna baza, uporabniški vmesnik, varnost

Abstract

In this seminar assignment I will describe making of information system for e-markbook. The assignment is first analyzed. Then I describe planning of data model. Follows planning of data flow diagram. At last we design user interface where we split the application on separate pages for each user type. I also describe tools I have used for creating data model, processes, programming language, web framework, IDE and actual application building. On the end I also describe testing my application.

Keywords: information system, e-markbook, database, user interface, security

Kazalo

Kazalo slik

1. Uvod

Naloga je namenjena uporabnikom, ki se odločajo za prehod iz analogne oblike shranjevanja ocen (papir) v digitalizirano obliko, v informacijske sisteme. Namenjena je tudi tistim, ki se s tem področjem poklicno ukvarjajo ali pa jih področje samo zanima.

1.1 Opredelitev problema

V zadnjem času se vse več šol odloča za digitalizacijo podatkov. Da bi omogočili čim lažji prehod, je potrebno zagotoviti zanesljiv, učinkovit in varen informacijski sistem. Na trgu je kar nekaj izbire, a se uporabniki še vedno težko odločijo za sistem, zaradi katerega bodo morali spremeniti način dela. Informacijski sistem omogoči centralizacijo, s tem pa poenoten dostop do podatkov, kar omogoča lažji pregled nad podatki, poenostavljeno izdelavo statistik po obdobjih in dostop v vsakem trenutku.

1.2 Cilji

Cilj te naloge je vzpostavitev varnega in učinkovitega informacijskega sistema, ki uporabnikov ne bo omejeval pri delu, ampak jim bo v pomoč. Tukaj moramo upoštevati, da se na šolah zahteve za hranjenje ocen redno spreminjajo, zato mora biti sistem fleksibilen. Npr. število ocenjevalnih obdobij mora biti spremenljivo, saj je to odvisno od letnega načrta.

Zaradi zahtevnosti celovitih informacijskih rešitev sem se odločil za izdelavo manjšega sistema za spletno redovalnico, ki bo v osnovi vseboval podatke o dijakih in njihovih starših, nato pa bodo profesorji vanj vnašali dogodke in ocene.

Naloga je razdeljena na izdelavo podatkovnega modela in vzpostavitve podatkovne baze in izdelavo spletne aplikacije, preko katere bodo imeli uporabniki centraliziran dostop do njihovih podatkov. Aplikacija mora imeti različne uporabniške vmesnike za različne tipe uporabnikov (dijak, starš, profesor). Administratorju sistema mora omogočati pregled in urejanje vseh podatkov v sistemu.

Spletna aplikacija mora biti preprosta, saj mora uporabnikom omogočiti čim lažji prehod iz analogne oblike hranjenja podatkov. Uporabnik mora takoj po prijavi videti najbolj bistvene informacije. Za dijake je to npr. seznam zadnjih ocen.

1.3 Metode dela

Delo na seminarski nalogi bo samostojno, za posvet pa bom imel na voljo mentorja. Poudarek je na individualnem delu. To bo potekalo s pomočjo sodobne informacijske tehnologije, kar vključuje sodobna orodja za načrtovanje podatkovnih modelov, relacijsko podatkovno bazo, najnovejša orodja za izdelovanje spletnih strani oz. aplikacij, varen spletni strežnik itd.

Čas, ki bo temu namenjen, mora biti ustrezno razdeljen na analizo, načrt, izvedbo in testiranje, saj so vsa ta obdobja pomembna.

2. Analiza

Preden začnemo z izvedbo rešitve, moramo narediti analizo. Pregledati moramo iz česa je sistem sestavljen, kakšne so njegove zahteve, kako bomo to implementiralo in kaj za to potrebujemo.

Sistem mora profesorjem omogočati vnašanje dogodkov in ocen za razrede, katere poučujejo, razrednikom pa vpogled v ocene za njihove razrede. Do ocen lahko dostopajo tudi dijaki in njihovi starši. Administrator mora imeti dostop do vseh podatkov (profesorji, predmeti, razredi, dijaki, ocenjevalna obdobja, ocene …). Možen naj bo ogled statistik in izpis ocen. Podatki so shranjeni v relacijski podatkovni bazi z uporabo sistema za upravljanje s podatkovno bazo (SUPB) MySQL. Do podatkov se lahko dostopa z uporabo spletne aplikacije. Zagotoviti je potrebno tudi varnost podatkov. Dostop do aplikacije mora biti zavarovan z varno povezavo.

Najprej moramo zgraditi podatkovni model. Na podlagi tega modela lahko naredimo kontekstni diagram toka podatkov. Sledi načrt uporabniškega vmesnika, kjer procese pretvorimo v vizualni načrt aplikacije. To pomeni, da naredimo okvirno oblikovanje. Sledi izvedba, kjer na podlagi podatkovnega modela ustvarimo podatkovno bazo.

Za gradnjo aplikacije bomo uporabili MVC[1] arhitekturo. Aplikacija bo sestavljena iz treh slojev, in sicer iz podatkovnega, predstavitvenega in sloja poslovne logike. Za preslikavo podatkovnega modela v aplikacijo bomo uporabili metodo ORM[2]. Vsak entitetni tip se preslika v svoj razred v Python kodi. Nato bomo napisali kodo za poglede (tuj. views). Na koncu bomo napisali HTML predloge za prikaz podatkov in vse skupaj povezali v celoto. Za postavitev sistema bomo na potrebovali strežnik, na katerega bomo namestili MySQL SUPB za podatkovno bazo in Apache strežnik za spletno aplikacijo. Spletni strežnik Apache bomo ustrezno zaščitili s SSL certifikatom.

[1] MVC – model-view-controller arhitektura
[2] ORM – objektno-relacijska preslikava

3. Načrt rešitve

3.1 Podatkovni model

Za analizo problema sledi načrtovanje. Ker sistem temelji na podatkovni bazi, moramo pripraviti podatkovni model, ki mora ustrezati vsem zahtevam uporabnikov.

Uporabnik

Entitetni tip uporabnik vsebuje uporabniško ime, ime, priimek, e-naslov in geslo. Ker za ta entitetni tip poskrbi spletno ogrodje, vsebuje tudi nekaj atributov, ki za nas niso pomembni. Entitetni tip uporabnik je skupen za vse tipe uporabnikov.

Dijak

Entitetni tip dijak vsebuje podatke, ki jih potrebujemo za dijaka. To so EMŠO, datum rojstva, podatek ali živi v dijaškem domu, mobitel, referenci na naslov za stalno in začasno prebivališče, referenci na entitetni tip starš za podatke o očetu in materi ter referenco na uporabnika.

Starš

Entitetni tip starš ima referenco na uporabnika in naslov ter osnovne podatke o osebi, kot so domači telefon, službeni telefon in mobitel. Slednji niso obvezni za vnos, so pa zaželeni zaradi kontaktnih podatkov za dijaka.

Profesor

Entitetni tip profesor vsebuje samo referenco na entitetni tip uporabnik, saj drugih podatkov o profesorju za ta sistem ne potrebujemo.

Naslov

Entitetni tip naslov vsebuje podatke o prebivališču. Vsebuje podatke o ulici, hišni številki, poštni številki in kraju.

Šolsko leto

Entitetni tip šolsko leto vsebuje začetno leto, končno leto in podatek ali je to trenutno šolsko leto.

Ocenjevalno obdobje

Entitetni tip ocenjevalno obdobje vsebuje ime ocenjevalnega obdobja (npr. 1. polletje), datuma začetka in konca ter referenco na šolsko leto.

Smer

Entitetni tip smer vsebuje ime izobraževalne smeri.

Razred

Entitetni tip razred vsebuje ime razreda, referenco na šolsko leto, smer in profesorja kot razrednika. Preko vmesne tabele je povezana na entitetni tip dijak, s čimer določimo kateri dijaki so v katerem razredu.

Predmet

Entitetni tip predmet vsebuje kratico predmeta (npr. SLO) in njegovo celotno ime.

Poučuje

Entitetni tip poučuje se uporablja za povezovanje profesorja, razreda in predmeta. S pomočjo tega entitetnega tipa lahko ugotovimo, katere predmete ima nek razred, kateri profesorji učijo razred itd.

Dogodek

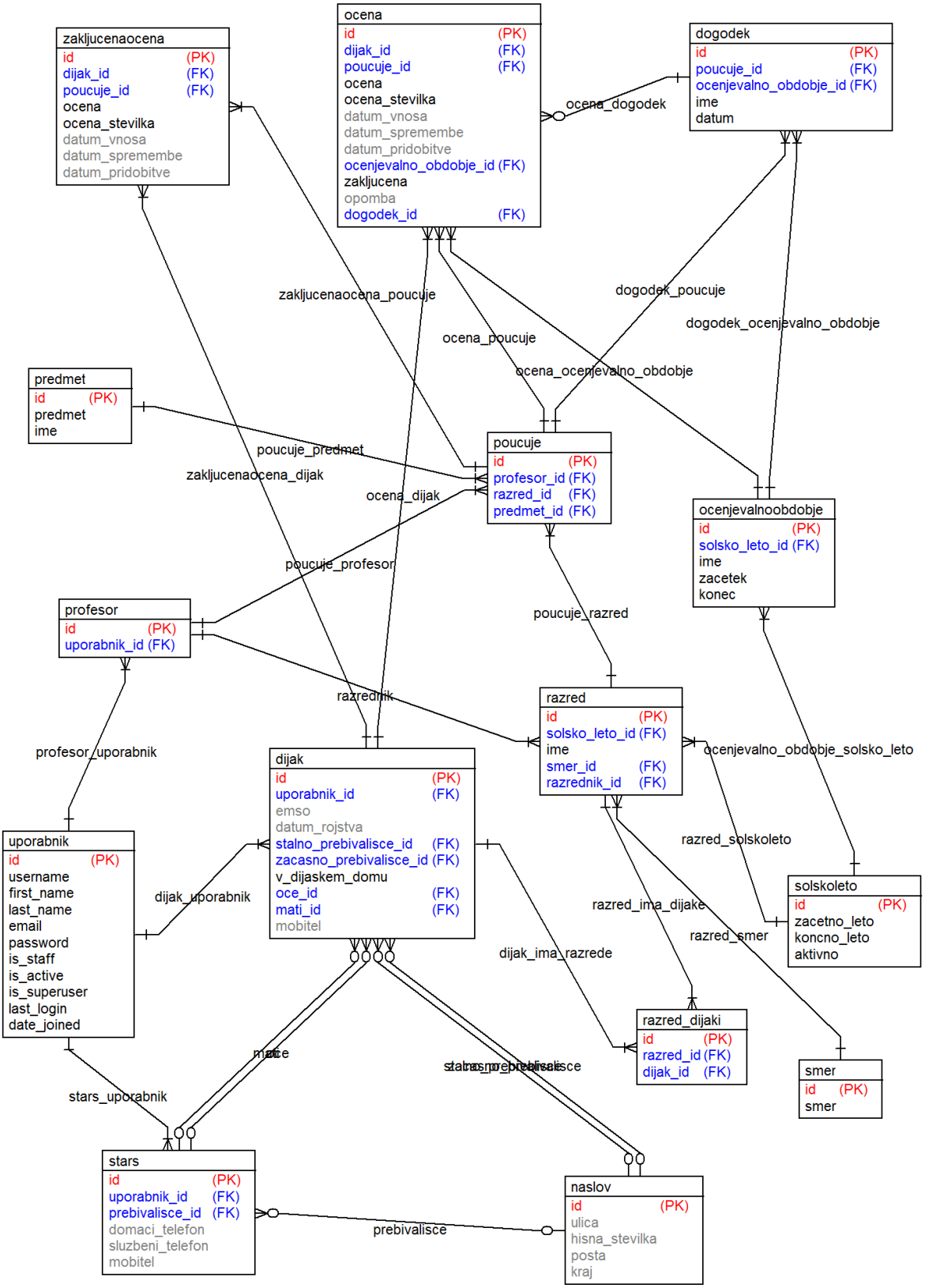
Entitetni tip dogodek vsebuje ime in datum dogodka, ter referenci na entitetni tip poučuje ter ocenjevalno obdobje. Ta tip se uporablja za informiranje uporabnikov o preizkusih znanja. Ker lahko na dogodek vežemo več ocen, nam to poenostavi vnos ocen za preizkuse znanj.

Ocena

Entitetni tip ocena vsebuje znak ocene (npr. 2 ali N), celoštevilčno oceno (2 ali 0), datum vnosa, spremembe in pridobitve ocene ter opombo. Vsebuje tudi reference na dijaka, entitetni tip poučuje, ocenjevalno obdobje in dogodek. Referenca na dogodek ni obvezna, saj lahko ocene vnašamo tudi samostojno.

Zaključena ocena

Entitetni tip zaključena ocena vsebuje znak ocene, celoštevilčno oceno, datum vnosa, spremembe in pridobitve ocene. Vsebuje tudi referenco na dijaka in entitetni tip poučuje. Ta tip se uporablja za podatke o zaključenih ocenah z predmete na koncu leta.



Slika 1: Podatkovni model

3.2 Načrt aplikacije

Pri načrtovanju aplikacije moramo določiti vso funkcionalnost aplikacije. Določiti moramo tok podatkov in module aplikacije.

3.2.1 Moduli aplikacije

Aplikacija je sestavljena iz več modulov. S tem aplikacijo razdelimo na manjše dele, kar nam omogoči čistejšo kodo in lažje programiranje.

Infosys

Modul bo vseboval vse ORM modele, ki jih bomo uporabili za povezovanje s podatkovno bazo. V teh modelih definiramo vse atribute entitetnega tipa, privzeto razvrščanje podatkov, privzeti prikaz modela in celotna imena za prikaz v administraciji.

```
class SolskoLeto(models.Model):
    zacetno_leto = models.PositiveIntegerField()
    koncno_leto = models.PositiveIntegerField()
    aktivno = models.BooleanField(default=True)

    objects = SolskoLetoManager()

    class Meta:
        verbose_name = u'Šolsko leto'
        verbose_name_plural = u'Šolska leta'
        ordering = ('zacetno_leto',)

    def __unicode__(self):
        return u'%d - %d' % (self.zacetno_leto, self.koncno_leto)
```

Slika 2: Model za entitetni tip šolsko leto

Prijava

Prijava je vstopna stran aplikacije. Ko se uporabnik prijavi, ga sistem glede na tip uporabnika preusmeri na njegovo stran.

```
def profile_url(user):
    profile = user.get_profile()

    if profile is None:
        return

    if profile.is_dijak():
        return reverse('dijak')

    elif profile.is_stars():
        return reverse('stars')

    elif profile.is_profesor():
        return reverse('profesor')

    elif profile.is_admin():
        return reverse('admin:index')
```

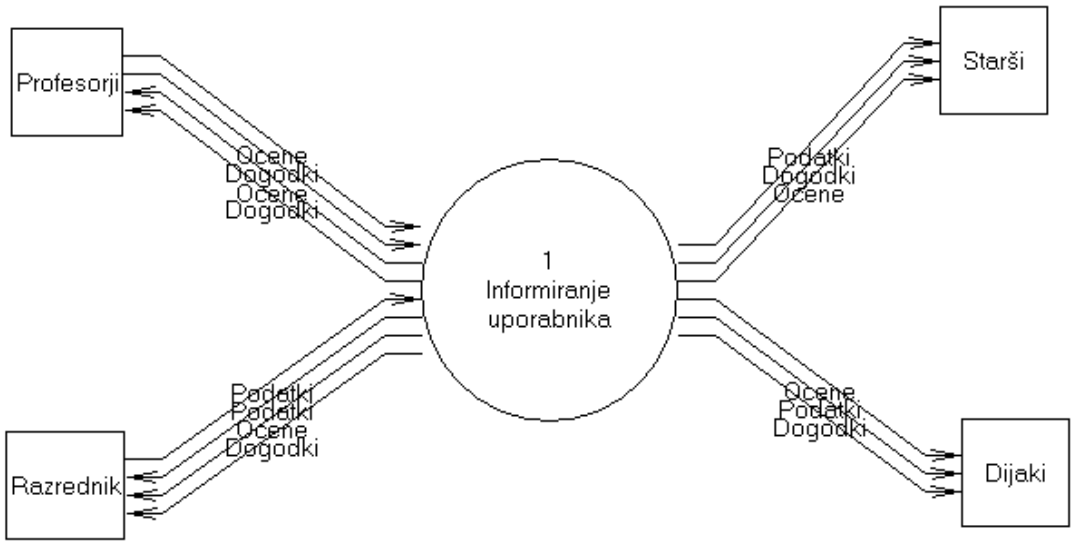
Slika 3: Koda za ugotavljanje tipa uporabnika

Dijak, starš in profesor

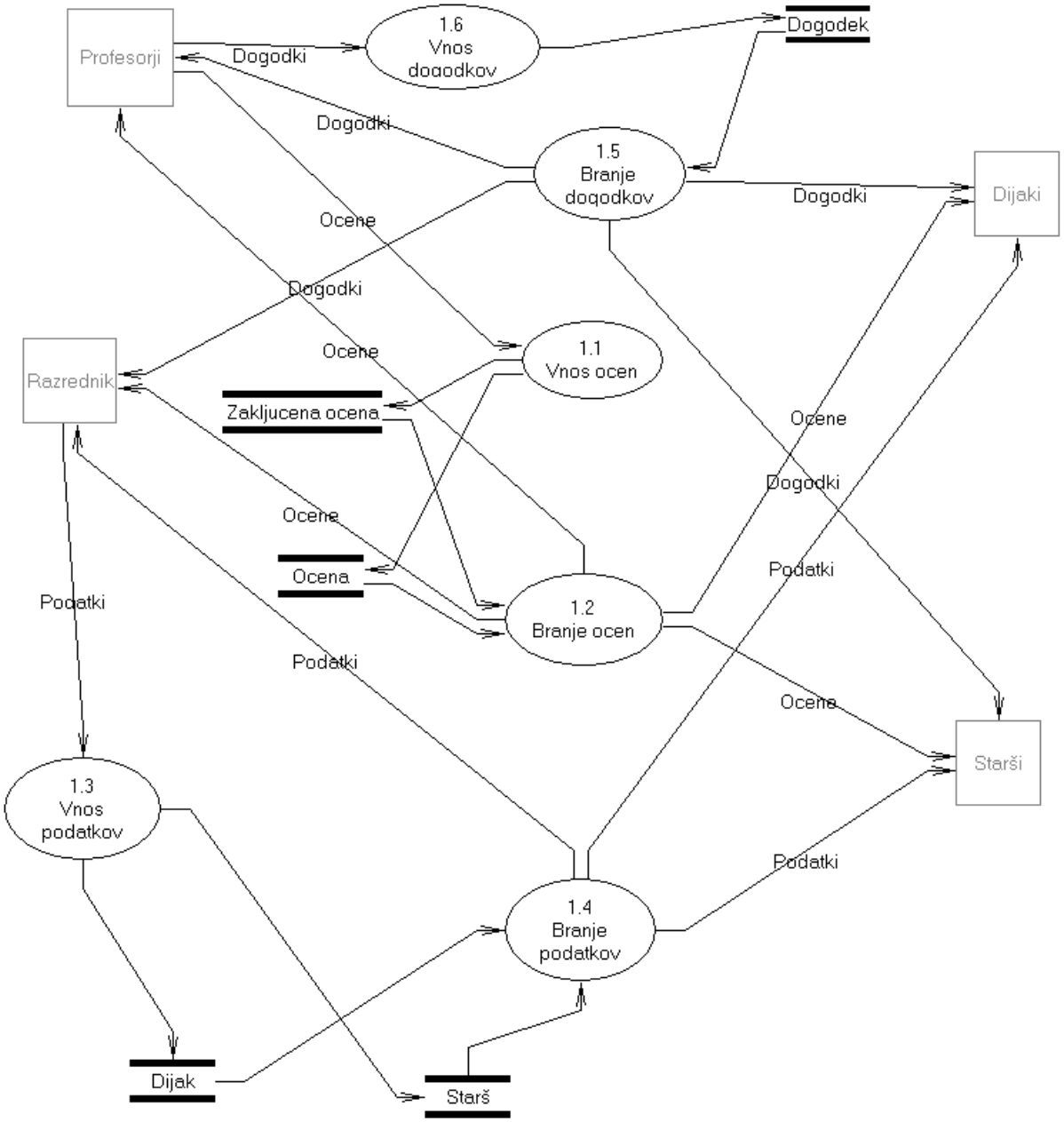
Moduli vsebuje poslovno logiko za odvisno od tipa uporabnika in HTML predloge, ki skrbijo za prikaz podatkov.

3.2.2 Tok podatkov

Ko načrtujemo aplikacijo, moramo določiti tudi tok podatkov. Tukaj definiramo vse procese aplikacije, terminatorje in zbirke podatkov, katere smo določili v podatkovnem modelu aplikacije. V našem primeru je terminator tip uporabnika. Upoštevati moramo vsa pravila za gradnjo diagramov toka podatkov.



Slika 4: Kontekstni diagram toka podatkov

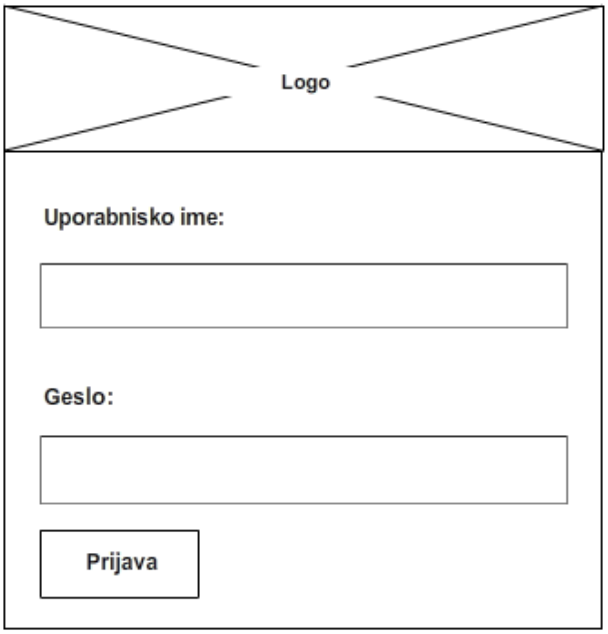


Slika 5: Prvi nivo diagrama toka podatkov

3.3 Načrt uporabniškega vmesnika

3.3.1 Prijava

Prijavni zaslon vsebuje logotip aplikacije, polje za vnos uporabniškega imena, polje za vnos gesla in gumb za prijavo.



Slika 6: Načrt strani za prijavo

3.3.2 Načrt za stran uporabnika

Vsak tip uporabnika ima drugačno strukturo strani. Strukture ne moremo poenotiti, saj imajo tipi uporabnikov različne pravice za dostop do podatkov.

Dijak

Struktura strani za dijaka:

- **Prva stran**
  - **Zadnje ocene**
  - **Prihajajoči dogodki**
- **Nastavitve**

  Tukaj lahko uporabnik spremeni svoje geslo in e-naslov.
- **Ocene**

  Seznam ocen, razdeljen na posamezna ocenjevalna obdobja.
  - **Natisni**
  - **Ocena**
- **Dogodki**

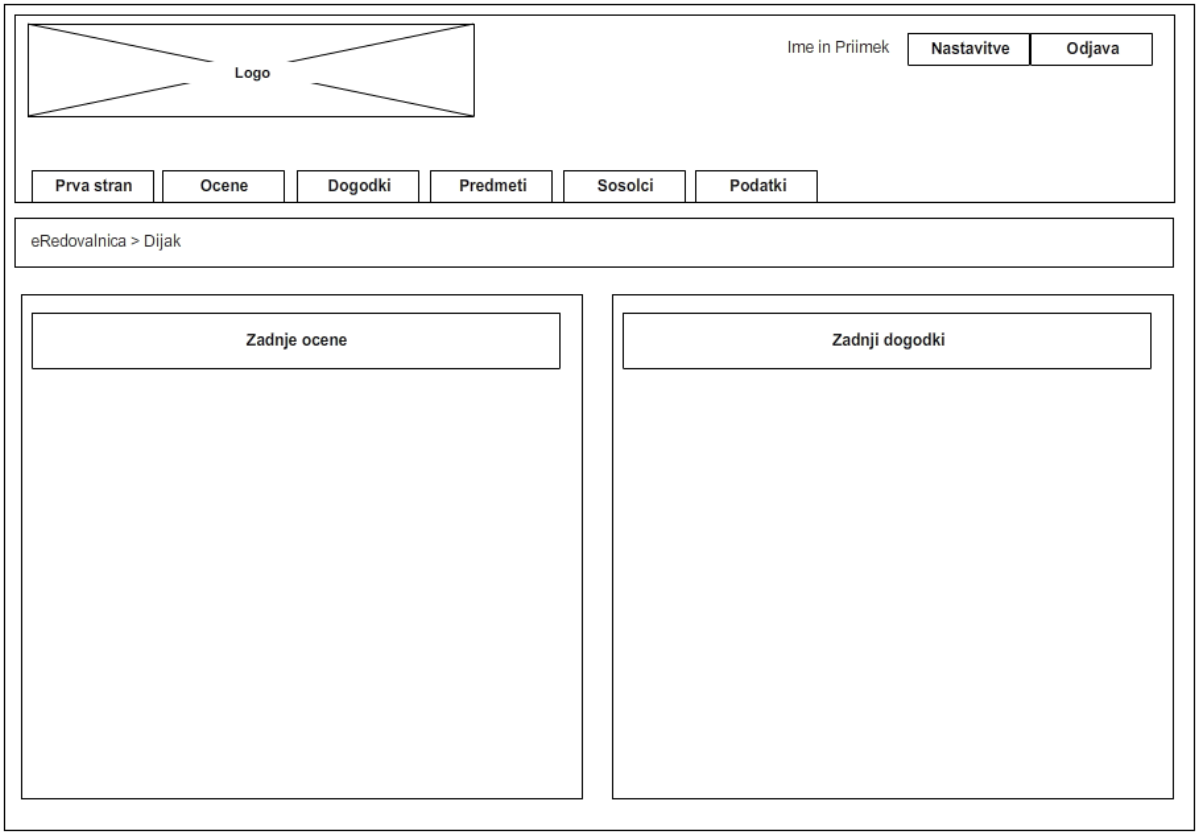
  Seznam prihajajočih in preteklih dogodkov.
  - **Dogodek**

- **Podatki**

  Podatki o dijaku in starših.
- **Sošolci**

  Seznam sošolcev in sošolk.
- **Predmeti**

  Seznam predmetov in profesorjev, ki dijaka poučujejo.



Slika 7: Načrt prve strani za dijake

Starš

Starši na prvi strani izberejo dijaka, nato pa vidijo podobno stran kot dijak.

Struktura strani za starša:

- **Dijak**

  Na strani dijaka je seznam zadnjih ocen in prihajajočih dogodkov.
  - **Ocene**

    Seznam ocen, razdeljen na posamezna ocenjevalna obdobja.
    - **Natisni**
    - **Ocena**

- **Dogodki**

  Seznam prihajajočih in preteklih dogodkov.

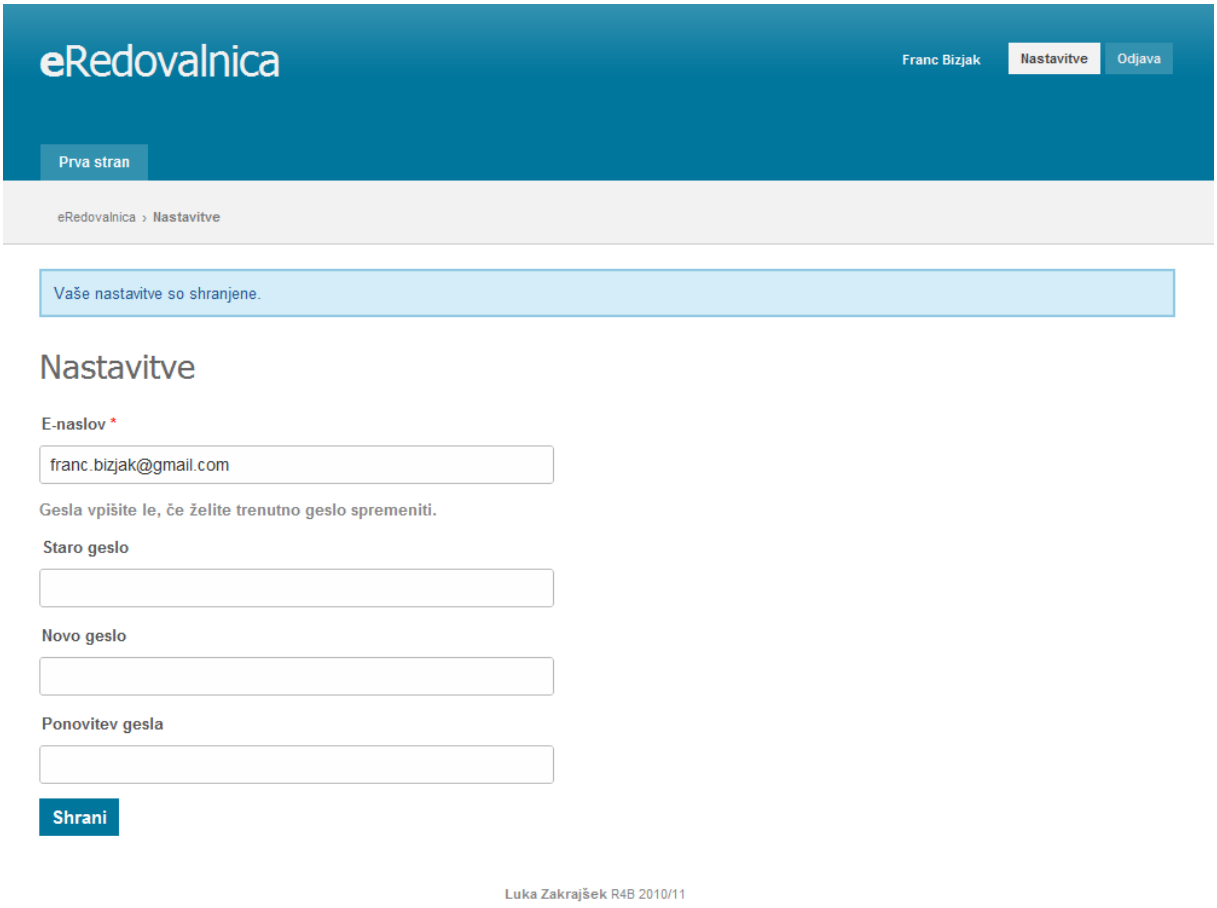
  - **Dogodek**

- **Predmeti**

  Seznam predmetov in profesorjev, ki dijaka poučujejo.

- **Nastavitve**

  Tukaj lahko uporabnik spremeni svoje geslo in e-naslov.



Slika 8: Spreminjanje nastavitev za starše

Profesor

Profesorji imajo na prvi strani seznam predmetov, katere poučujejo in razred, v katerem je ta profesor razrednik, kar pa ni nujno. Profesor ima tudi stran spreminjanje nastavitev.

Struktura strani za profesorja:

- **Predmet**

  Na strani predmeta je seznam razredov, v katerih profesor uči.

  - **Razred**

    Na strani razreda je seznam dijakov in seznam zadnjih dogodkov.

    - **Seznam dijakov**
      - **Natisni seznam dijakov**
      - **Dijak**
        - **Ocene**

          Ocene so razporejene na ocenjevalna obdobja. Profesor lahko tukaj zaključi predmet, katerega poučuje.

          - **Natisni**
          - **Dodaj oceno**
          - **Ocena**
            - **Uredi oceno**
            - **Izbriši oceno**
          - **Zaključi predmet**
          - **Zaključena ocena**
            - **Izbriši zaključeno oceno**
        - **Podatki**

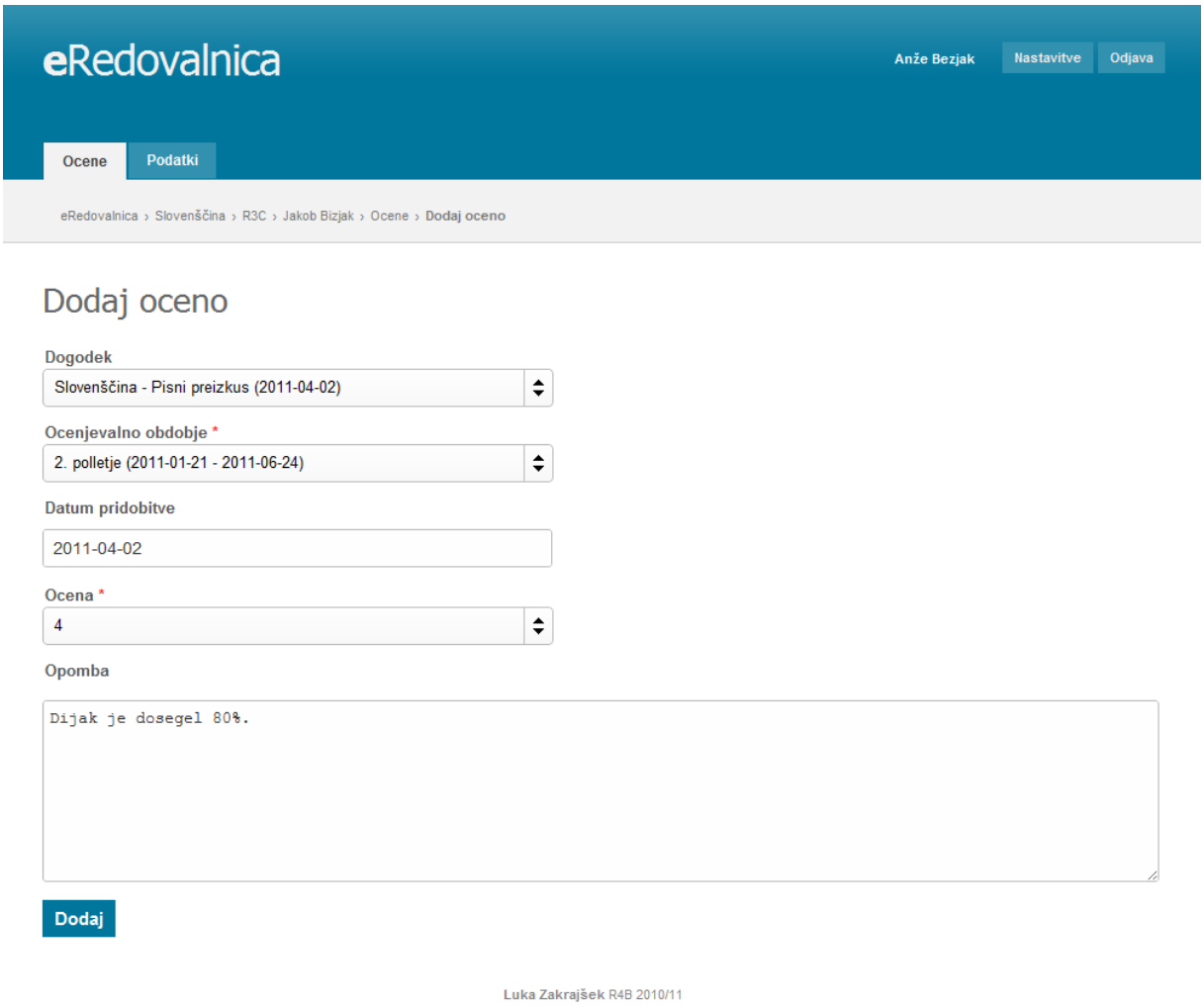
          Profesor, lahko vidi podatke o dijaku, katerega uči, ne more pa urejati njegovih podatkov.
    - **Dogodki**

      Profesor lahko za predmet, katerega uči, dodaja in ureja dogodke, kot npr. pisni preizkus.

      - **Dodaj dogodek**
      - **Dogodek**

        Na strani dogodka je tudi seznam dijakov, katerim lahko profesor doda oceno za dogodek.

        - **Uredi dogodek**
        - **Izbriši dogodek**



Slika 9: Dodajanje ocene

Struktura strani za razrednika:

- **Razred**

  Seznam dijakov in dogodkov. Profesor lahko doda novega dijaka v razred in natisne seznam.

  - **Dijaki**
    - **Natisni**
    - **Dodaj dijaka**
    - **Dijak**
      - **Ocene**
        - **Natisni**
        - **Ocena**
          - **Uredi oceno**

- **Podatki**

  Razrednik lahko dijaka, očeta in mater izbriše, uredi podatke in dodeli novo geslo.

  - **Dijak**
    - **Uredi podatke**
    - **Izbriši dijaka**
    - **Novo geslo**
  - **Oče**
    - **Uredi podatke**
    - **Izbriši**
    - **Novo geslo**
  - **Mati**
    - **Uredi podatke**
    - **Izbriši**
    - **Novo geslo**

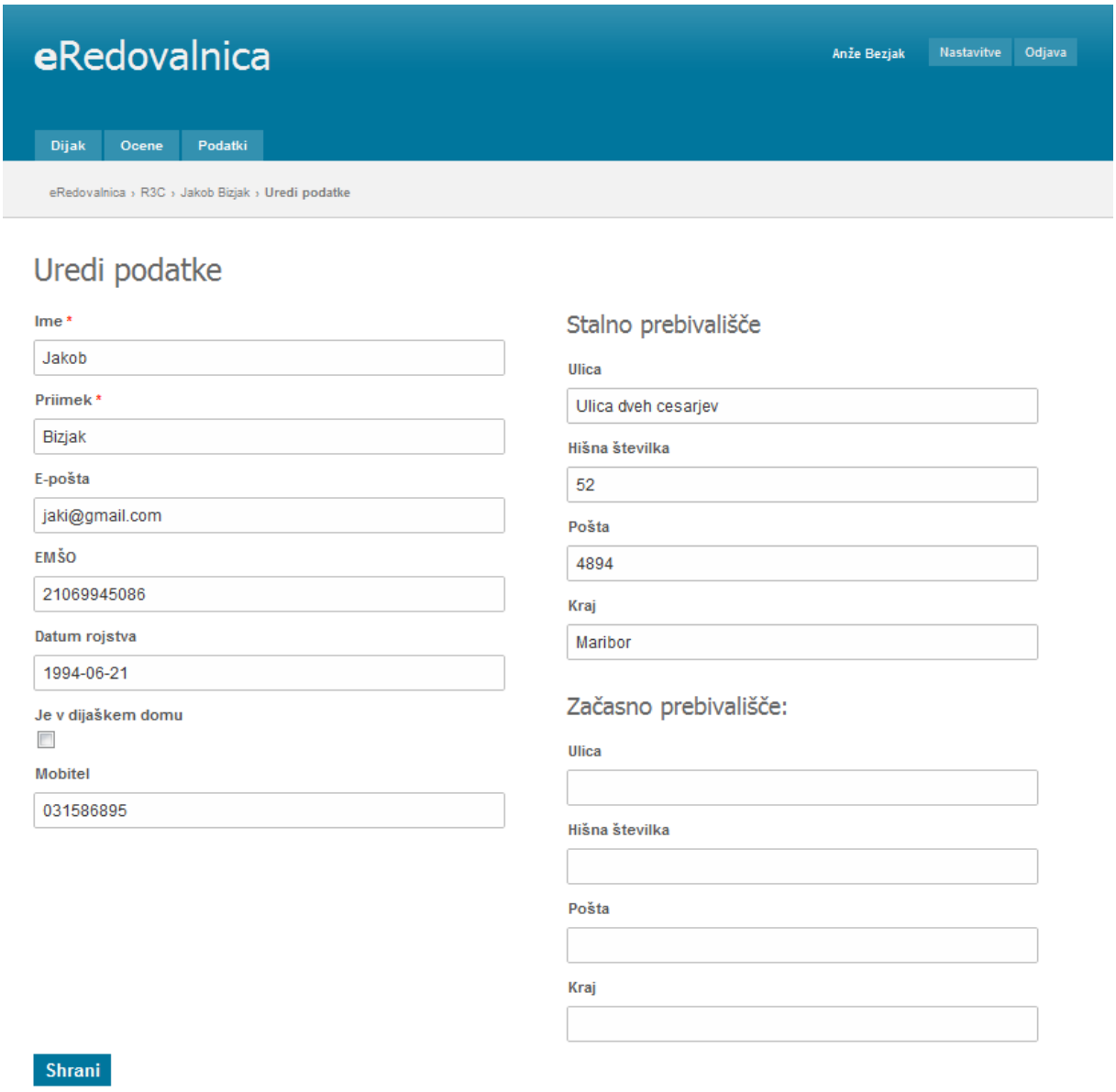
- **Dogodki**

  Seznam prihajajočih in preteklih dogodkov.

  - **Dogodek**
    - **Uredi**
- **Predmeti**

  Seznam predmetov in profesorjev, ki učijo v tem razredu.
- **Statistika**

  Stran za statistiko vključuje seznam dijakov s povprečnimi ocenami in tortni diagram z razmerji ocen.



Slika 10: Urejanje dijakovih podatkov

4. Izvedba

4.1 Orodja

4.1.1 Načrtovanje podatkovnega modela

Za načrtovanje podatkovnega modela sem uporabil programsko orodje Case Studio. To orodje omogoča izdelavo entitetno-relacijskih diagramov, ki jih lahko uporabimo za gradnjo kontekstnih diagramov toka podatkov ali pa za izvoz v SUPB.

4.1.2 Programski jezik

Za izvedbo aplikacije sem uporabil programski jezik Python. Python je tolmačeni programski jezik. Ima popolnoma dinamične podatkovne tipe, samodejno upravlja s pomnilnikom in podpira funkcionalno, imperativno oziroma proceduralno, strukturirano in objektno orientirano računalniško programsko paradigmo. Zaradi svoje dinamične narave je programiranje spletnih aplikacij z uporabo spletnih ogrodij zelo poenostavljeno.

4.1.3 Spletno ogrodje

Uporabil sem spletno ogrodje Django. Uporaba tega ogrodja poenostavi izdelavo spletnih aplikacij hitreje in z manj kode.

Objektno relacijska preslikava

Objektno relacijska preslikava (ali tuje ORM) nam omogoča, da definiramo podatkovni model v celoti v Python kodi. Uporabimo lahko vmesnik za enostavno dostopanje do podatkovne baze, še vedno pa lahko uporabimo SQL[3].

Avtomatski administracijski vmesnik

Za izdelavo administratorskega vmesnika pri izdelavi spletnih aplikacij porabimo zelo veliko časa. Avtomatski vmesnik nam to poenostavi, saj lahko z nekaj vrsticami kode izdelamo celoten vmesnik.

Čisti URL[4]-ji

[3] SQL – strukturiran povpraševalni jezik za delo s podatkovnimi zbirkami, s programskimi stavki, ki posnemajo ukaze v naravnem jeziku.
[4] URL – je edinstven naslov, ki enolično določa neki spletno stran.

Z uporabo čistih URL-jev lahko polepšamo URL in uporabniku poenostavimo dostop do spletne strani. Uporabnik v brskalnik lažje vnese naslov http://www.eredovalnica.si/dijaki/11 kot http://www.eredovalnica.si/index.php?mod=mod_dijak&action=view&id=11.

Sistem za predloge

Z uporabo sistema za predloge ločimo obliko od vsebine in Python kode.

Sistem za »predpomnjenje«

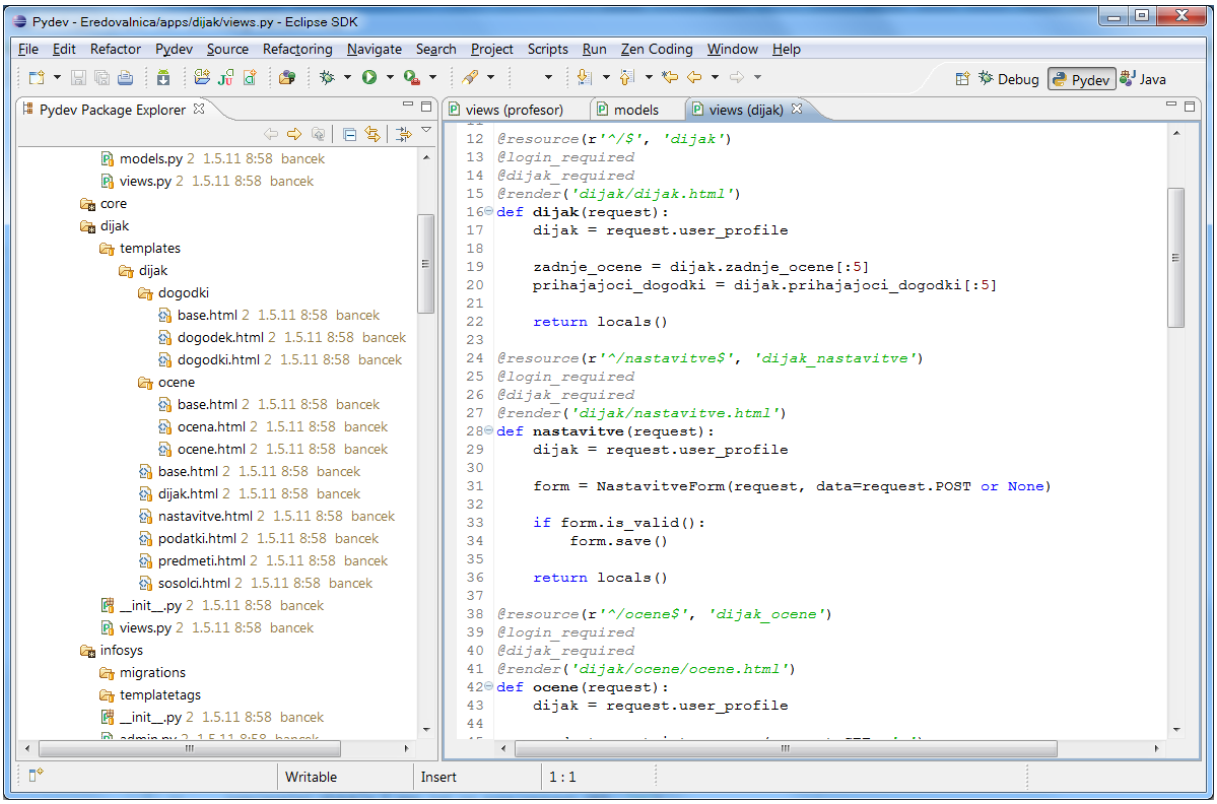
Z uporabo predpomnilnika lahko optimiziramo hitrost strani , izboljšamo uporabniško izkušnjo in zmanjšamo potrebo po boljši strojno opremi. Namesto da bi nekatere podatke vedno nalagali iz podatkovne baze ali jih preračunavamo, jih lahko začasno shranimo v predpomnilnik.

Internacionalizacija

Ogrodje Django ima celotno podporo za izdelovanje več jezičnih aplikacij. Tako bi našo aplikacijo z lahkoto prevedli v druge jezike.

4.1.4 Integrirano razvojno okolje

Za pomoč pri izvedbo aplikacije lahko uporabimo integrirano razvojno okolje (IDE) Eclipse. Eclipse je odprtokodno okolje za razvoj programske opreme. Orodjem Eclipse nam omogoča razvijanje aplikacij v različnih programskih jezikih. Za pomoč pri razvijanju Python aplikacij lahko uporabimo dodatek za Eclipse PyDev. Ta dodatek nam zelo poenostavi tudi razvijanje Django aplikacij.

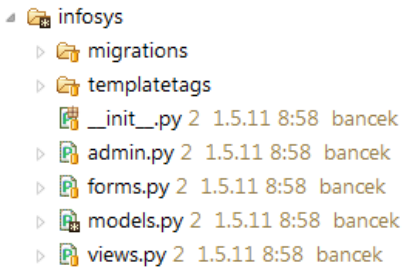


Slika 11: Razvojno okolje Eclipse

4.2 Programski moduli

4.2.1 Modul »infosys«

Modul vsebuje podatkovne modele za komunikacijo s podatkovno bazo. Vsebuje tudi modul za administracijo. Spletno ogrodje Django zelo poenostavi administracijo, saj že pri modelih podatkov določimo, kako so podatki povezani, katera polja so opcijska, kakšna so celotna imena polj itd. S temi podatki se administracijski vmesnik avtomatsko generira.

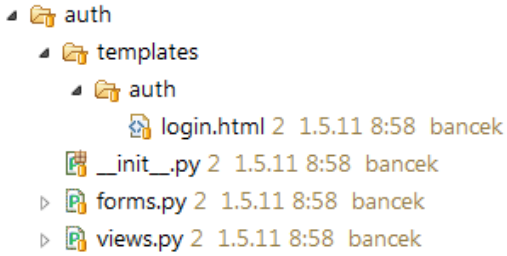


Slika 12: Struktura modula »infosys«

4.2.2 Modul »prijava«

Slika 13: Prijavni zaslon

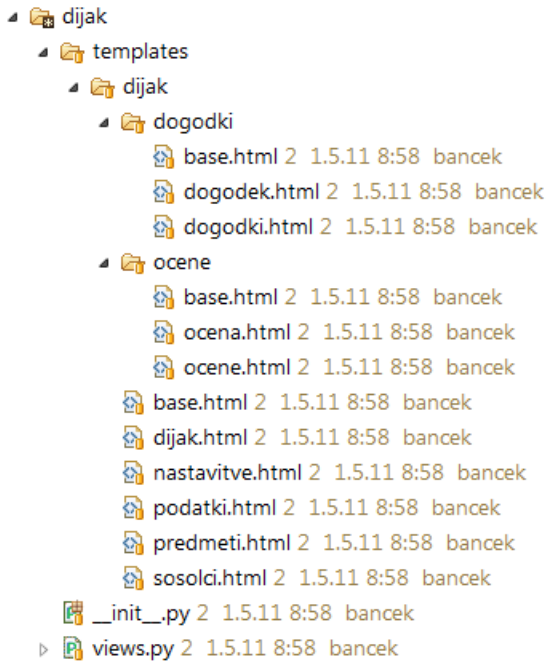
Modul za prijavo oz. preverjanje pristnosti je edini skupni del aplikacije za vse vrste uporabnikov. Sestavljen je obrazca za vnos uporabniškega imena in gesla ter prijavne logike, ki uporabnika, glede na njegov tip, preusmeri na njegovo stran.



Slika 14: Struktura modula »prijava«

4.2.3 Modul »dijak«

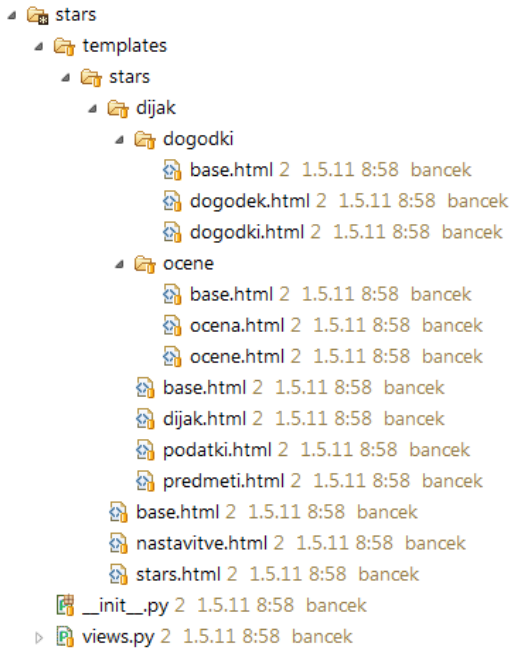
Modul dijakom omogoča pregled dogodkov, njihovih ocen, osebnih podatkov, seznam predmetov in sošolcev ter stran za spreminjanje nastavitev. Na strani za nastavitve si lahko dijak spremeni svoje geslo in e-naslov.



Slika 15: Struktura modula »dijak«

4.2.4 Modul »starš«

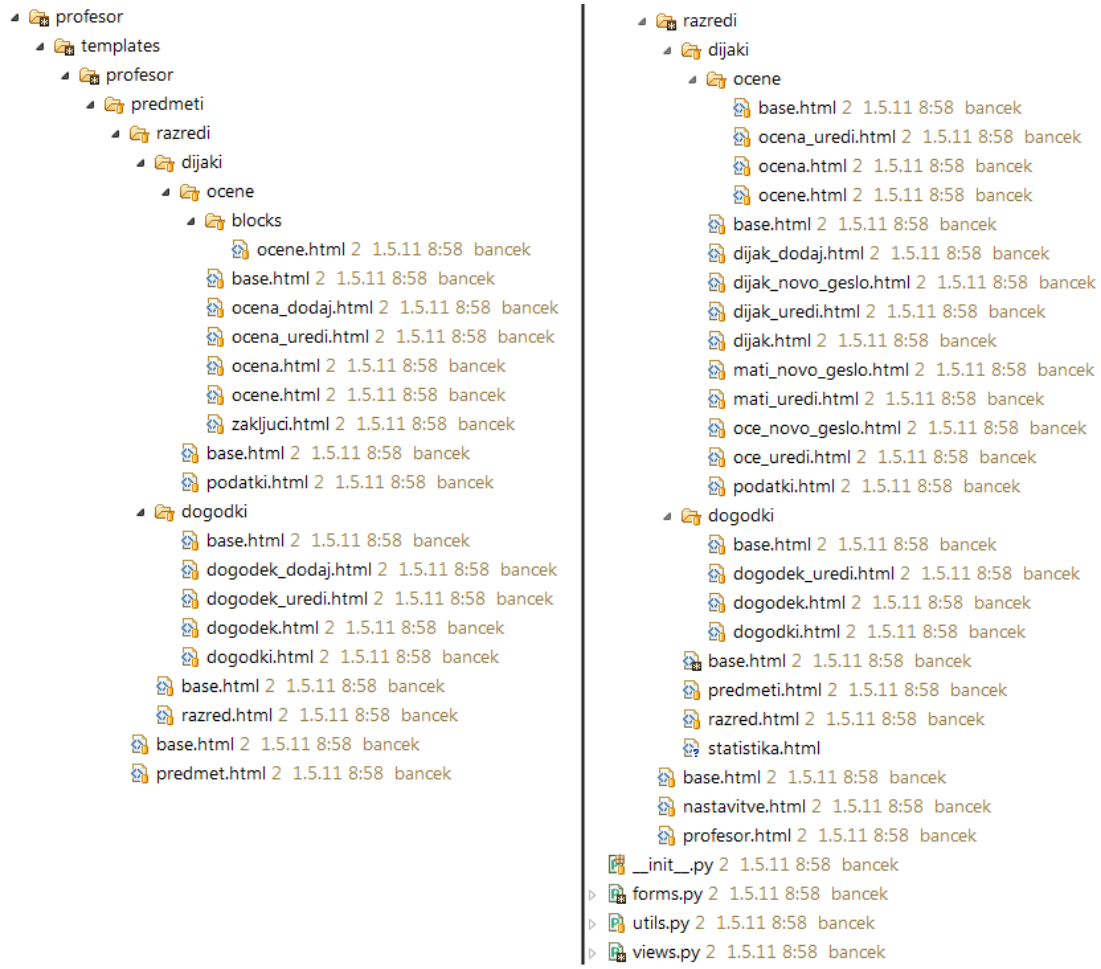
Modul staršem omogoča pregled podatkov o njihovih otrocih. To vsebuje pregled dogodkov, ocen dijaka, osebnih podatkov in seznam predmetov. V nastavitvah si lahko spremenijo geslo in e-naslov.



Slika 16: Struktura modula »starš«

4.2.5 Modul »profesor«

Modul je razdeljen na strani za profesorje, ki poučujejo predmete in strani za razrednike, kjer lahko razredniki spreminjajo podatke o dijakih za svoj razred.



Slika 17: Struktura modula »profesor«

5. Testiranje

Za testiranje aplikacije sem pred izdelavo same aplikacije napisal modul, ki je ustvaril naključne podatke za 60 profesorjev, 44 razredov in 1305 dijakov. Podatki so ustvarjeni tako, da so profesorji smiselno razdeljeni po razredih, v enem razredu pa je od 25 do 32 dijakov, kar se lahko primerja s povprečno srednjo šolo. To mi je zelo poenostavilo razvoj aplikacije, saj sem lahko delal z realnim številom podatkov, tako da sem lahko kodo ustrezno optimiziral in temu primerno naredil oblikovanje. Napak pri testiranju ni bilo, saj sem aplikacijo gradil z uporabo testnih podatkov.

Oblikovanje strani sem testiral v večini sodobnih brskalnikov in poskrbel, da strani izgledajo enako. Nekaj težav sem imel v starejših različicah brskalnika Internet Explorer.

Informacijski sistem sem prenesel na Linux strežnik, na katerega sem postavil podatkovni strežnik MySQL in spletni strežnik Apache 2. Strežnik se nahaja za požarnim zidom, tako da sta za javnost odprti samo dve TCP vrati (80 in 443). Povezava med uporabnikom in spletnim strežnikom je zavarovana s HTTPS povezavo in SSL certifikatom.

6. Zaključek

Kot cilj sem si zastavil izdelavo celotnega informacijskega sistema, podprtega z podatkovno bazo in oplemenitenega s spletno aplikacijo. Zadane cilje sem dosegel, saj mi je zastavljeno uspelo realizirati. Med izdelavo sem se naučil veliko novega o izdelavi entitetno-relacijskih diagramov, kontekstnih diagramov toka podatkov, načrtovanju modulov aplikacije, načrtovanju uporabniškega vmesnika in izdelave celovite rešitve. Sistem mi je uspelo postaviti na privatnem strežniku, do katerega lahko uporabniki dostopajo preko javnega naslova. Strežnik je ustrezno zavarovan.

Sistema mi žal ni uspelo testirati pri velikem številu uporabnikov, saj za to nisem imel ustreznih virov. V prihodnje bi želel, da bi sistem preizkusili tudi dejanski uporabniki, tako dijaki kot starši in profesorji, saj bi za izpopolnjeno uporabniško izkušnjo potreboval odzive uporabnikov.

Sistem sem testiral tudi na prenosnih napravah, na pametnem mobilnem telefonu z operacijskim sistemom Android in tabličnem računalniku iPad in ugotovil, da je aplikacija primerna tudi za uporabo pri pouku v šoli. To bi poenostavilo vpis ocen, saj nebi bilo potrebno vzdrževati različnih redovalnic, kot so profesorjeva »ročna«, razredna in še spletna.

Glede na to, da sistem že vsebuje možnosti za vnos podatkov za dijake, razrede in predmete, bi ga lahko nadgradil z modulom za spletni dnevnik. To bi poenostavilo vnašanje podatkov pri pouku. Npr. pri vnosu manjkajočih dijakov, bi lahko izbirali iz seznama dijakov v razredu, kar bi olajšalo vnos in zmanjšalo možnost napak. Ker aplikacija deluje na tabličnih računalnikih, bi profesorji to lahko uporabljali kar za katedrom.

7. Viri

- http://www.akadia.com/services/ssh_test_certificate.html (7. 4. 2011)
- http://www.cek.ef.uni-lj.si/magister/zaloznik90-B.pdf (8. 4. 2011)
- http://colos1.fri.uni-lj.si/ERI/RACUNALNISTVO/PROG_JEZIKI_ORODJA/prog_ide.html (9. 4. 2011)
- http://sl.wikipedia.org/wiki/Python_(programski_jezik) (9. 4. 2011)
- http://www.djangoproject.com/ (10. 4. 2011)
- http://sl.wikipedia.org/wiki/SQL (10. 4. 2011)
- http://sl.wikipedia.org/wiki/URL (12. 4. 2011)
- http://hpc.fs.uni-lj.si/eclipse (13. 4. 2011)

8. Literatura

- Belak, Janko. 1980. Informacijski sistemi. Maribor: Visoka ekonomsko komercialna šola
- Demšar, Janez. 2009. Python za programerje. Ljubljana: Fakulteta za računalništvo in informatiko
- Holovaty, Adrian, Kaplan-Moss, Jakob. 2009. The definitive guide to Django : Web development done right. Berkeley: Apress, New York: Springer-Verlag.