Izvlačenje strukovnog nazivlja iz strojarskih tekstova

Benić, Juraj

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Juraj Benić

Zagreb, 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:
Prof.dr.sc. Mario Essert

Student:
Juraj Benić

Zagreb, 2015.

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studija i navedenu literaturu.

Zahvaljujem se svom mentoru prof. dr. sc. Mariu Essertu što mi je omogućio da napišem ovaj rad, te se zahvaljujem i svojim roditeljima što su mi omogućili studiranje.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **Juraj Benić** Mat. br.: 0035188447

Naslov rada na hrvatskom jeziku: **Izvlačenje strukovnog nazivlja iz strojarskih tekstova**

Naslov rada na engleskom jeziku: Extracting the professional terminology from engineering texts

Opis zadatka:

S obzirom na brz i kontinuiran razvitak tehnologije i znanosti izvan hrvatskoga govornoga područja, postoje velike poteškoće s hrvatskim strukovnim nazivljem unutar brojnih tehničkih disciplina. Preuzeti pojmovi su obično surogati engleskog ili njemačkog nazivlja i teško se usklađuju sa standardiziranim pojmovima hvale vrijednog projekta pod imenom „Struna“ (http://struna.ihjj.hr/) *Instituta za hrvatski jezik i jezikoslovlje*. Stoga je cilj ovog rada načiniti program koji bi djelomično umanjio tu neusklađenost.

U ovom radu potrebno je istražiti i provesti:

1. Osnovne algoritme za izvlačenje informacije po različitim gramatičkim i semantičkim kategorijama iz teksta;
2. Opisati *web2py* mrežnu tehnologiju (http://www.web2py.com/) izgrađenu na programskom jeziku Python i programiranje računalnih baza u Pythonu;
3. Načiniti mrežnu aplikaciju s pomoću *web2py* okvira (eng. *framework*) koja će omogućiti korisnicima poluautomatizirano dohvaćanje željenog strukovnog nazivlja i njihovog spremanja u posebnu *MySql* bazu s kojom će se graditi zajednička baza, nakon stručne provjere.
4. Uspoređivati tako spremljeno nazivlje s normativnim nazivljem na *Struni* po mrežnom obrascu (formi) izgradjenom za upis i provjeru izvučenog nazivlja.

Zadatak zadan:
25. studenog 2014.

Rok predaje rada:
1. rok: 26. veljače 2015.
2. rok: 17. rujna 2015.

Predviđeni datumi obrane:
1. rok: 2., 3., i 4 . ožujka 2015.
2. rok: 21., 22., i 23. rujna 2015.

Zadatak zadao:
M. Essert
Prof.dr.sc. Mario Essert

Predsjednik Povjerenstva:
Prof. dr. sc. Zoran Kunica

Sadržaj

1 UVOD **1**

2 O PYTHON-U **2**
2.1 sqlite3 modul 2
2.2 Poredani rječnik klasa iz "collections" paketa 5

3 O WEB2PY-U **6**
3.1 Tok podataka u web2py-u 6
3.2 Model 8
3.3 Kontroler (controller) 9
3.4 Pregled (view) 11
3.5 Primjer web2py aplikacije 11

4 OD ZAMISLI DO REALIZACIJE **18**
4.1 Unos teksta za obradu 18
4.2 Rastavljanje na rečenice i riječi 19
4.3 Baze podataka 20
4.4 Obrasci (forma) za unos pojmova 21
4.4.1 Podjela područja, polja i grane 23
4.5 Prikaz rezultata 26

5 IZVLAČENJE INFORMACIJE IZ TEKSTA **27**
5.1 Pridruživanje gramatičkih oblika riječima 27
5.2 Sastavljanje uzoraka i izraza 28
5.2.1 Uzorci 29
5.2.2 Izrazi 30
5.3 Pretraživanje teksta i dohvaćanje informacije 30

6 ZAKLJUČAK **35**

LITERATURA **36**

PRILOZI **37**

Popis slika

1 Izgled baze . . . 3
2 Izgled index-a baze . . . 4
3 Administracijsko sučelje u web2py-ju [3] . . . 6
4 Tok zahtjeva u web2py-u [3] . . . 7
5 Veza između imena controllera i view-a . . . 9
6 Veza između funkcija i odgovarajuće datoteke u view-u . . . 10
7 Prijedlog pretraživanja . . . 14
8 Primjer web2py aplikacije . . . 17
9 Skica ideje . . . 18
10 Forma za unos teksta . . . 19
11 Baza riječi . . . 20
12 IHJJ struna . . . 20
13 Tehnički rječnik . . . 21
14 Forma za unos pojmova u bazu podataka . . . 22
15 Ispis nakon obrade teksta . . . 26
16 Ispis nakon korisnikovog uključivanja tooltip-a . . . 26
17 Prikaz tagova za danu rečenicu . . . 28
18 Prikaz zadavanja uzoraka . . . 30
19 Prikaz zadavanja izraza . . . 30
20 Uzorci za pretraživanje teksta . . . 31
21 Izraz za traženje predikata u tekstu . . . 32

Popis tablica

1 Različito značenje pojma anoda u raznim područjima 1
2 Opis dodatnih svojstava uzoraka 29

SAŽETAK

U ovom radu proučiti će se mogućnosti Python-a i web2py-a za obradu i izvlačenje informacija iz teksta. Na početku će biti dan kratki uvod u rad sa bazama podataka u Python-u. Zatim će se pokazati web2py. U tom dijelu pokazuje se veza između modela, kotntrolera i pregleda, te kako su oni međusobno povezani. To će se ilustrirati malim primjerom u web2py-u. U trećem dijelu će se razraditi ideja kako iz teksta izvući tehničke pojmove koji se nalaze u bazama podataka te kako prikazati korisniku na jednostavan način te mu uz to omogućiti interakciju s rezultatima. Za kraj dati će se algoritam koji će biti u mogućnosti izvlačiti informacija iz teksta (subjekt, objekta....) uz pomoć regularnih izraza i gramatičkih oblika riječi.

Ključne riječi: Python; web2py; tehnički pojmovi; izvlačenje informacije;

SUMMARY

In this paper we examine the possibilities of Python and web2py for processing and extracting information from text. At the beginning of the paper we will give a brief introduction for work with databases in Python. After, we present web2py. This part show relation between the model, controller and view.That will be illustrated with a simple web2py application.The third part will elaborate the idea of extracting technical terms from text which are found in databases and how to display it to the user in a simplest possible way and also allowing him to interact with results. At the end will give the algorithm that will be able to extract information from text (subject, object ....) with the help of regular expressions and grammatical forms of a word.

Keywords: Python; web2py; technical terms; extraction of information;

1 UVOD

Godine 1989. Tim Berners-Lee predstavio je osnovnu ideju za današnji internet. Do ideje je došao kada je želio povezati računala u zajedničku mrežu kako bi olakšao dijeljenje podataka i informacija među svojim kolegama u CERN-u. Tada je korištenjem HTTP-a (HyperText Transfer Protocol) pokrenut World Wide Web, a 1991. godine javnosti je predstavljena prva internetska stranica.

Sa sve većim razvojem interneta i novih tehnologija dolazi do širenja stručnog vokabulara s novim stručnim pojmovima. Neke riječi čak poprimaju različita značenja u različitim područjima npr. uzmimo za primjer riječ anoda:

Tablica 1: Različito značenje pojma anoda u raznim područjima

Pojam	Značenje	Područje
anoda	negativno nabijena elektroda	kemija
anoda	pozitivna elektroda u elektrolitskome članku	elektrotehnika
anoda	elektroda na kojoj prevladava anodna reakcija	kemijsko inženjerstvo
anoda	elektroda koja ima veći električni potencijal	fizika
inertna anoda	anoda koja nije podložna anodnomu otapanju i anodnoj koroziji	kemijsko inženjerstvo
netopljiva anoda	anoda koja se primjenjuje u sustavu katodne zaštite s narinutom strujom i koja se tijekom rada sustava ne troši znatno	kemijsko inženjerstvo
pomoćna anoda	dodatna anoda koja se upotrebljava tijekom elektronanošenja radi postizanja željene raspodjele debljine prevlake	strojarstvo

Zbog toga se javlja potreba da se osmisli i napravi program koji bi nekom tko čita neki stručni tekst, a to područje mu je strano, pomogao razumjeti stručne pojmove tako što bi im davao njihov opis, te bi im omogućavao unos novih pojmova u bazu ako taj pojam ne postoji u bazi podataka.

2 O PYTHON-U

Python je besplatni programski jezik. Vrlo je jednostavan zbog toga što je intuitivan i sintaksa mu je jako slična pseudokodu. Stvorio ga je Guido van Rossum ranih 90-tih godina u Nizozemskoj.

Za Python postoji na tisuće različitih modula od kojih su neki uključeni u standardnu biblioteku modula koja dolazi s Python-om dok su ostali moduli od zajednice Python-ovih korisnika koji nude brojne mogućnosti. U ovom radu ćemo obratiti pozornost na nekoliko standardnih modula potrebnih za izradu ovog rada.

2.1 sqlite3 modul

Sqlite3 modul jedan je od standardnih i dolazi s Python-om. Služi za stvaranje baze podataka i njezin rad. Modul ne zahtijeva nikakve dodatne servere za rad s bazom podataka.

Da bi započeli rad s bazama podataka prvo moramo pozvati modul *sqlite3* i spojiti se bazom podataka. Spajanje na bazu podataka radi se s funkcijom *connect* koja vraća objekt koji predstavlja bazu podataka, a ako baza toga imena ne postoji, onda je stvori.

```
1 #!/usr/bin/env python
2 # -*- coding: utf_8
3 import sqlite3
4 db = sqlite3.connect('struna.sqlite')
```

Nakon uspješnog spajanja s bazom moramo stvoriti *cursor* objekt koji služi za izvršavanje sql naredbi. On izvršava sql naredbe te pomoću njega kreiramo tablice, pretražujemo, brišemo i ažuriramo podatke u tablici. Za kreiranje tablice trebamo izvršiti sql naredbu *CREATE TABLE IF NOT EXISTS* te joj dodati ime tablice koje želimo kreirati i navesti polja koja treba u njoj kreirati. Svako polje može biti drugačijeg tipa (text, varchar, integer, boolean, time, date ...) i ograničenja (unique, primary key, not null ...).

```
1 c=db.cursor()
2 c.execute('''CREATE TABLE IF NOT EXISTS struna
3              (id INTEGER PRIMARY KEY, natuknica TEXT, opis
4               TEXT, link TEXT,glagoli TEXT, imenice TEXT,
5               nepoznato TEXT)''')
6 db.commit()
```

Popunjavanje tablice radi se pomoću sql naredbe *INSERT INTO* kod koje moramo navesti ime tablice te u zagradi imena stupaca u koje želimo unijeti podatke te vrijednosti koje želimo unijeti. Ako za vrijednosti stavimo znak *?* onda u funkciju *execute* možemo dodati još tuple argument čija dužina odgovara broju stupaca u

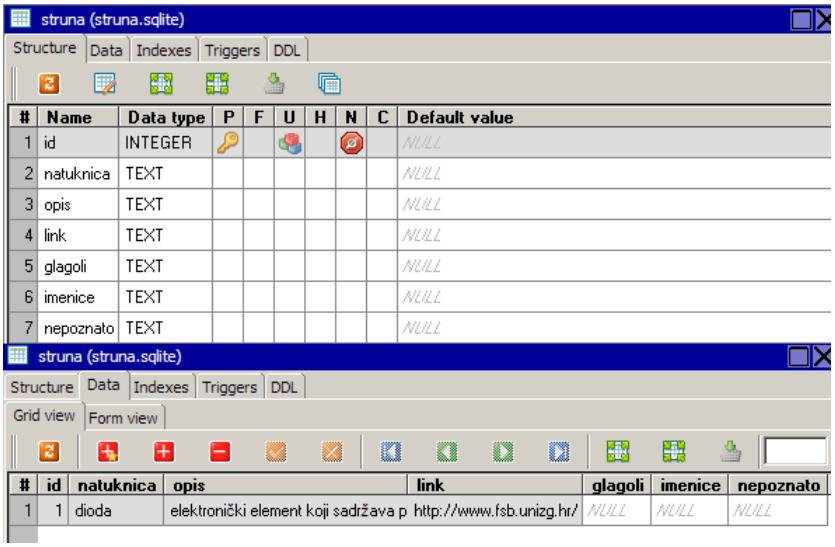
koje želimo unijeti podatke, a u tuple spremimo vrijednosti ili varijable s podacima. Umjesto znaka *?* možemo jednostavno staviti podatke koje želimo unijeti, ali bi onda morali generirati sql naredbu svaki puta za podatak dok sa znakom *?* jednostavnije prođemo kroz *for* petlju i unesemo podatke.

```
1 n=u'dioda'
2 o=u'elektronicki element koji sadrzava poluvodicki p-n spoj...'
3 c.execute('''INSERT INTO struna (natuknica,opis) values(?,?)'''
      ,(n,o))
4 db.commit()
```

Ažuriranje podataka vrši se naredbom *UPDATE* gdje još moramo navesti ime tablicu koju želimo ažurirati, stupac u koji želimo unijeti vrijednost te samu vrijednost i kriterij po kojem želimo da se odabere red u tablici. Odabiranje reda radi se pomoću naredbe *WHERE* koja traži da joj se navede ime stupca i uvjet po kojem će odabrati redove u tablici.

```
1 c.execute('''UPDATE struna SET link = 'http://www.fsb.unizg.hr/'
      WHERE id=1''')
2 db.commit()
```

Kada se svi algoritmi prethodno navedeni u ovom odlomku spoje u jedan dobije se baza s jednim podatkom, a njen izgled se vidi na slici 1.



Slika 1: Izgled baze

Pretraživanje baze se vrši sa *SELECT* naredbom gdje definiramo imena stupaca iz kojih želimo iščitati podatke, tablicu te uvjet pomoću kojeg filtriramo podatke. Ako želimo pročitati podatke iz svih stupaca u bazi stavljamo znak * koji označuje sve stupce u bazi, umjesto da sami pišemo njihova imena.

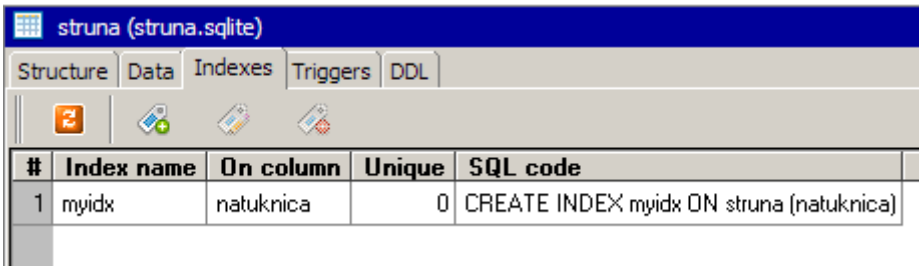
```
red=c.execute('''SELECT id, natuknica FROM struna WHERE
    natuknica LIKE ?''',('dioda',))
```

Pretraživanje podataka nam generira objekt koji možemo pročitati s dvije funkcije ili s pomoću *for* petlje. Za čitanje podataka koristimo dvije funkcije generiranog objekta *fetchone()* i *fetchall()*. Funkcija *fetchone()* nam vraća *none* ako ne postoji podatak ili tuple sa željenim podacima ako je nešto pronađeno. Funkcija *fetchall()* nam vraća praznu listu ako ne postoji podatak, a ako postoji ili ih ima više onda vraća listu podataka s n-tercima.

```
red.fetchone()
red.fetchall()
```

Bazu možemo pretraživati i preko indeksa baze. Da bi to mogli moramo ga prvo stvoriti. Indeks možemo kreirati na cijeloj tablici ili na jednom i/ili više stupaca baze. Index nema smisla koristi na malim bazama podataka gdje se pretraživanje odvija brzo, nego na velikim bazama. Indeks se stvara naredbom *CREATE INDEX IF NOT EXISTS* i tu naredbu nadopunimo s imenom našeg indeksa i kažemo za koju ga tablicu želimo kreirati.

```
c.execute('''CREATE INDEX IF NOT EXISTS myidx ON struna (
    natuknica);''')
db.commit()
```



Slika 2: Izgled index-a baze

Pretraživanje baze pomoću indeksa vrši se slično običnom pretraživanju. Razlika je u tome što sintaksi dodajemo naredbu *INDEXED BY* i ime index-a, a sve ostalo ostaje isto kao kod običnog pretraživanja.

```
c.execute('''SELECT * FROM struna INDEXED BY myidx WHERE
    natuknica == ?;''',('dioda',))
```

Indeks se isto tako daje jednostavno brisati iz baze podataka ako nam više ne koristi ili mu želimo promijeniti ime.

```
c.execute('''DROP INDEX myidx;''')
db.commit()
```

Brisanje podataka vrši se naredbom *DELETE FROM* u kojoj navodimo tablicu iz koje želimo brisati podatak, te uvjet po kojem želimo da se podaci izbrišu.

```
1 c.execute('''DELETE FROM struna WHERE id=?;''',(1,))
2 db.commit()
```

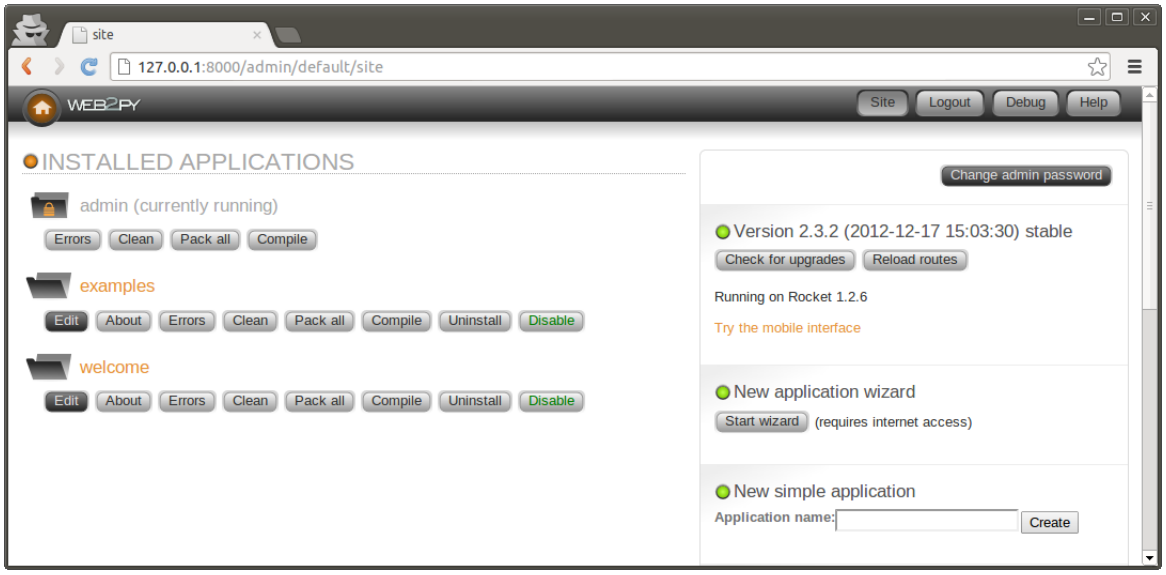
2.2 Poredani rječnik klasa iz "collections" paketa

Poredani rječnici isti su kao i obični, ali oni pamte redoslijed kojim su podatci bili uneseni u njega. Kada iteriramo poredani rječnik vrijednosti se vraćaju onim redoslijedom čiji je ključ bio prvi unesen.
Ako novi podatak prebriše postojeći podatak, njegova pozicija u rječniku ostaje ona koju je imao podatak koji je prebrisan. Brisanje podatka i njegovo ponovo umetanje u rječnik pomiče taj podatak na kraj rječnika.

3 O WEB2PY-U

Web2py je besplatni program za izradu web aplikacija. Napisan je u python-u i programira se s python-om. Massimo Di Pierro najzasluženiji je za njegovo stvaranje. Web2py je 2012. godine proglašen tehnologijom godine. Jednostavan je za korištenje i ne zahtijeva nikakvu instalaciju i dodatnu konfiguraciju.

Njegovo administracijsko sučelje nam omogućava stvaranje nove aplikacije, brisanje stare aplikacija i instaliranje već gotovih aplikacija. Omogućuje nam jednostavan uvid u njihove datoteke i rad s njima, te pregled baze i funkcija koje se koriste u aplikacijama.



Slika 3: Administracijsko sučelje u web2py-ju [3]

Svaka web2py aplikacija sastoji se od modela (datoteke koje sadrži opis prikaz baze), view-a (datoteke koje sadrže opis prikaza baze), controllers-a (datoteke koje sadrže opis poslovne logike i tijeka rada), Cron Jobs-a (zadatci koji moraju biti redovito izvršena u pozadini), modules-a (kolekcija klasa i funkcija) i Static datoteka (slike, skripte, ...).

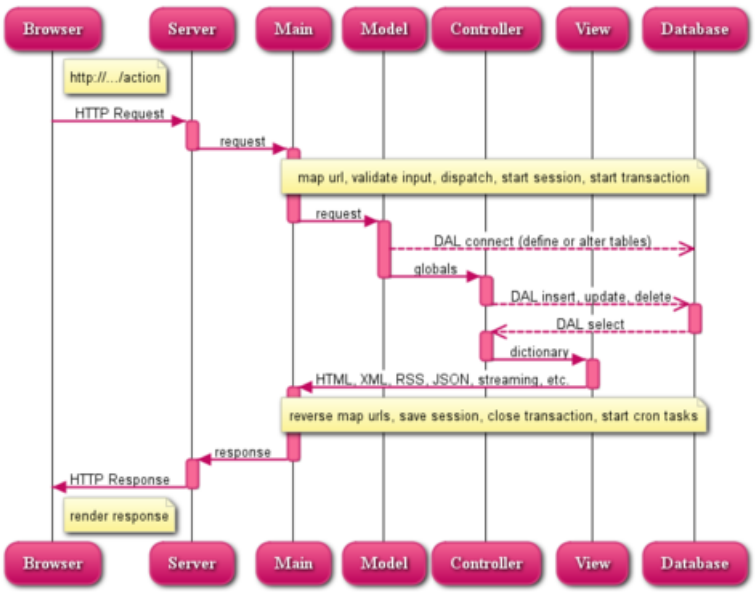
3.1 Tok podataka u web2py-u

Tok podataka u web2py-u je slijedeći:

- HTTP zahtjev dolazi na web server (ugrađeni Rocket server ili drugačiji server spojen s web2py putem WSGI[1] ili nekog drugog adaptera). Web server obrađuje svaki zahtjev u svom thread-u (programskoj niti) i paralelno s ostalim zahtjevima.

[1]WSGI (Web Server Gateway Interface) je skup specifikacija koje opisuju kako web server komunicira s web aplikacijama, te kako više aplikacija obradi jedan zahtjev.

- Zaglavlje HTTP zahtjeva raščlanjuje se i predaje dispečeru.
- Dispečer zatim odlučuje koja od instaliranih aplikacija će obraditi zahtjev i mapira *PATH_INFO* u URL koji poziva funkciju. Svaki URL odgovara jednoj funkciji.
- Zahtjevi koji uključuju datoteke iz static mape obrađuju se direktno, a velike datoteke su automatski poslane korisniku.
- Prije pozivanja akcije, događa se par stvari: ako zaglavlje zahtjeva sadrži session cookie za aplikaciju, vraća se session objekt. Ako ne sadrži, onda se kreira novi session te je time kreirana okolina za izvršavanje zahtjeva i u njoj se izvršavaju model.
- Konačno, izvršavaju se naredbe u controller-u u prije definiranoj okolini.
- Ako funkcija vraća string, on se vraća korisniku.
- Ako funkcija vraća iterabilan objekt, on se pomoću petlje vraća korisniku.
- Ako funkcija vraća rječnik web2py pokušava pronaći view za prikaz rječnika. View mora imati isto ime kao i funkcija i istu ekstenziju kao početna stranica
- Ako se uspješno izvrše sve korisničke naredbe transakcija se prihvaća.
- Ako se korisničke naredbe ne izvrše greška se sprema u ticket i korisniku se vraća ticket ID. Samo sistemski administrator može otvarati i čitati ticket. Ako mu to ne uspije, web2py pokušava otvoriti generički view. Cijeli korisnički kod se izvršava u jednoj transakciji ako nije drugačije specificirano.



Slika 4: Tok zahtjeva u web2py-u [3]

3.2 Model

U model se tipično spremaju baze podataka i funkcije koje služe za rad s bazama podataka. Svaka baza i funkcija u modelu postaju globalna varijabla bez da ju se prethodno mora definirati globalnom.

Web2py dolazi s DAL ("Database Abstraction Layer") klasom koja služi za kreiranje tablica i njeno korištenje. DAL dinamički generira sql sintaksu u realnom vremenu pomoću određenih pravila tako da programer ne mora učiti različite sql sintakse za različite tipove baza podataka. Podržani tipovi baza podataka: SQLite, MySQL, Oracle, ...

DAL objekt funkcionira slično kao python-ov modul sqlite3, samo što ne moramo koristiti sql sintaksu već koristiti gotove metode objekta za kreiranje tablica i rad s njima. Prvo se stvara veze s bazom podataka ako ona postoji, a ako ne postoji onda se stvara baza i veza. Nakon što smo se spojili s bazom podataka koristimo metodu *define_table* za stvaranje tablice koja kao argument prima ime tablice i polja koja želimo stvoriti. Metoda sama automatski dodaje polje id. Svakom polju možemo definirati njegov tip i ograničenja.

```
1 baza=DAL('sqlite://moja_baza.db')
2 baza.define_table('moja_tablica',
3     Field('polje_1', 'string'),
4     Field('polje_2', 'integer', required=True))
```

Spremanje podataka vrši se pomoću metode *insert*. Prvo moramo pristupiti tablici u bazi pa onda koristimo metodu. Spremati podatke možemo na 3 načina:

1. kroz *for* petlju punimo bazu gdje metodi navodimo podatak koji želimo spremiti u to polje

```
1 polje_1=['kruske','jabuke']
2 polje_2=[4,3]
3 for i in xrange(len(polje_1)):
4     baza.moja_tablica.insert(polje_1=polje_1[i],polje_2=
          polje_2[i])
```

2. koristeći rječnik čiji ključevi odgovaraju poljima u tablici

3. koristeći sql sintaksu koja je objašnjenja u poglavlju 2.1

Vađenje podataka iz tablice radi se metodom *select*. Za vađenje podataka prvo moramo navesti objekt baze kojem u argumentu navodimo bazu, tablicu i stupac za koji se treba ispuniti uvjet te na to pozivamo metodu *select* koja može i ne mora primati argumente. Ako se u pozivu metode ne navedu argumenti to znači da ona iz tablice vadi sve podatke za ispunjeni uvjet, a ako joj se navedu argumenti onda iz baze za taj uvjet vadi samo navedena polja.

```
rows = baza(baza.moja_tablica.id>0).select(baza.moja_tablica.
    polje_1)
```

DAL nam nudi metodu *executesql* pomoću koje možemo koristiti sql sintaksu. Ona nam isto omogućava stvaranje index-a nad tablicom jer DAL objekt ne posjeduje metodu za stvaranje index-a na tablici.

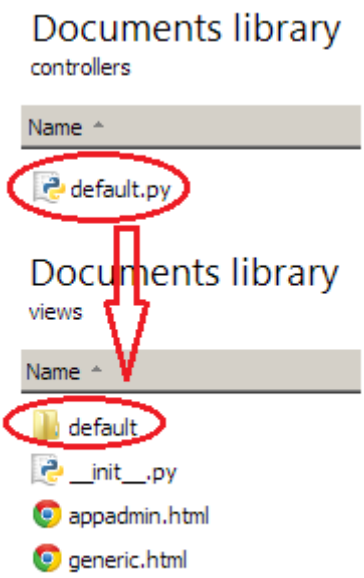
```
db.executesql('SELECT * FROM moja_tablica;')
```

Pomoću *drop* metode možemo ispustiti sve tablice i svi podatci bit će izbrisani.

```
db.moja_tablica.drop()
```

3.3 Kontroler (controller)

Svaki controller ima svoje ime i njegovo ime definira vezu između mape u view-u. To znači da ako imamo controller imenom *default* znači da u view-u moramo imati mapu imenom *default*, to je ilustrirano na slici 5.



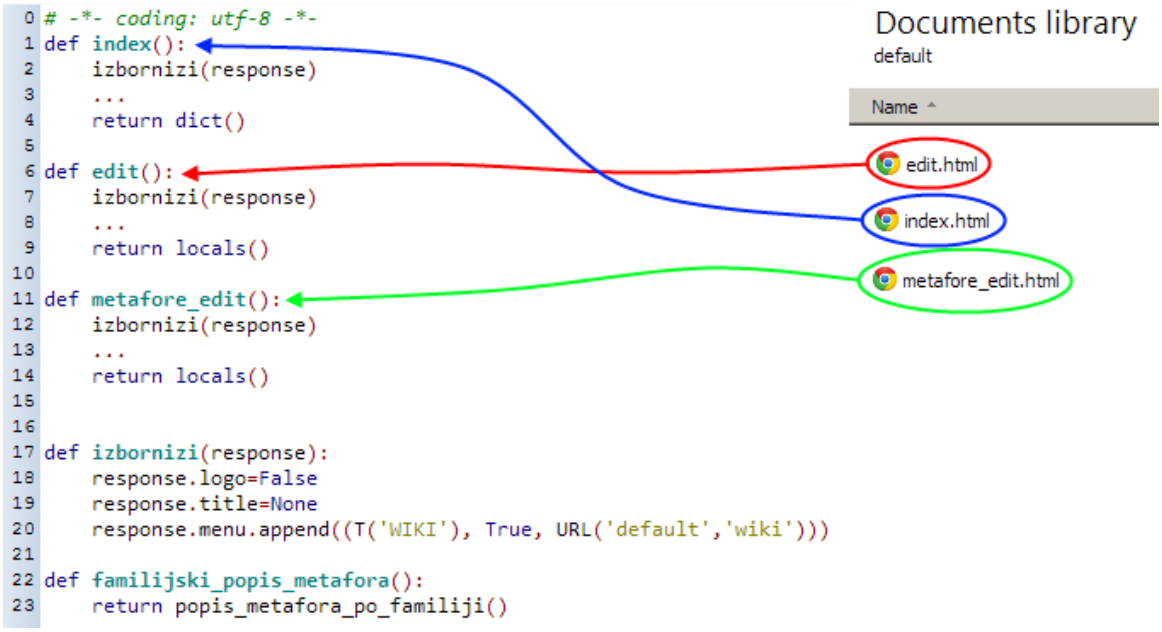
Slika 5: Veza između imena controllera i view-a

Svaki controller se sastoji od funkcija. Razlikujemo tri glavne vrste funkcija:

1. funkcije čije se ime veže na pojedinu stranicu aplikacije.
2. funkcije koje pozivamo unutar funkcija i služe nam skratiti kod.

3. funkcije koje se pozivaju pomoću *ajax-a*. One se ne vežu na ni jednu stranicu već se izvršavaju unutar te stranice, a vežu se na pojedini id nekog elementa gdje se nešto klikom želi promijeniti.

Na slici 6 možemo vidjeti sve tri vrste funkcija i veze između funkcija koje se vežu na pojedinu stranicu.



Slika 6: Veza između funkcija i odgovarajuće datoteke u view-u

Osim rječnika i string-a funkcija može vratiti lokalne varijable:

```
1 def index(): return locals()
```

preusmjeriti korisnika na drugu stranicu:

```
1 def index(): redirect(URL('druga_stranica'))
```

vratiti HTTP stranice koje govore o nekoj grešci:

```
1 def index(): raise HTTP(404)
```

vratiti "helper":

```
1 def index(): return FORM(INPUT(_name='test'))
```

ili rječnik koji sadrži "helper"

```
1 def index(): return dict(form=SQLFORM.factory(Field('name')).
    process())
```

Svaki "helper" prima iste argumente kao i element kojeg zamjenjuje u HTML kodu uz iznimku što kod "helper" moramo staviti _ (donju crticu) ispred argumenta jer bi se u protivnom njegovo ime miješalo s Python-ovim ključnim riječima kao što su id, class... Oni koji se ne podudaraju s Python-ovim ključnim riječima pišu se isto _ (donju crticu) zbog jednostavnosti.

```
polje_za_unos=INPUT(_name='ime', _type='text', _id='prvi',
    _style="color:red;")
```

3.4 Pregled (view)

U view-u se nalazi sav html kôd koji definira izgled stranice. Isto tako u view-u možemo pisati python-ov kod ali moramo naglasiti da se radi o Python-u, pa ga pišemo u vitičastim zagradama.

```
{{ python kod }}
```

Kod pisanja python-ovih naredba i view-u moramo obratiti pozornost na petlje i uvjete. Svaka petlja i uvjet mora imati svoj početak i kraj. To je u html kodu ne možemo riješiti pomoću tabova kako smo naučili u Python editoru-u već moramo staviti naredbu *pass* na kraju, koja interpreter-u označava kraj petlje ili uvjeta.

```
{{for i in xrange(1,11):}}
  <span>{{=i}}</span>
{{pass}}
```

Početak svake stranice u view-u je isti i započinje s naredbom *extend 'layout.html'* pomoću koje odabiremo osnovni izgled stranice, tj. poziciju meni-a, podjelu stranice na stupce i retke. Nakon toga može se, ali i ne mora, nalaziti naredba *block head* koja sve skripte i css kodove uključuje u head html stranice tako da se učitaju prije nego što se ostatak stranice učita. Ta naredba nije potrebna ako nemamo nikakve skripte i css te ako smo ih već uključili u layout-u. Na kraju dolazi *div* element s id-om *container* koji nije ništa drugo nego centrirani div element koji sadrži cijelu našu stranicu.

```
{{extend 'layout.html'}}

{{block head}}
  <!-- Ukljucivanje skripti i css -->
{{end}}

<div id="container">
  <!-- Preostali html kod stranice -->
</div>
```

3.5 Primjer web2py aplikacije

Slijedi primjer jednostavne web2py aplikacije koja može zbrojiti dva broja i stvoriti telefonski imenik te ga pretraživati. Uz to će korisnik aplikacija moći biti između dva jezika, a to su hrvatski i engleski jezik.

Prvo u modelu moramo definirati novu bazu podataka u koju ćemo spremati nove kontakte. Bazu ćemo nazvati "moja_baza" i imati će sqlite ekstenziju. U njoj ćemo definirati jednu tablicu "Imenik" kojoj ćemo dodijeliti polja i ograničenja, a kako to

izgleda u programu vidi se u primjeru 1.

```
1  baza = DAL('sqlite://moja_baza.sqlite')
2
3  baza.define_table('Imenik',
4      Field('Ime', 'string',label=T('Ime'),requires=IS_NOT_EMPTY()
           ),
5      Field('Prezime', 'string',label=T('Prezime'),requires=
           IS_NOT_EMPTY()),
6          Field('Telefon', 'integer',label=T('Telefon'),requires=
               IS_INT_IN_RANGE(0, 1e10)),
7          Field('Adresa', 'string',label=T('Adresa')),
8      )
```

Primjer 1: Model

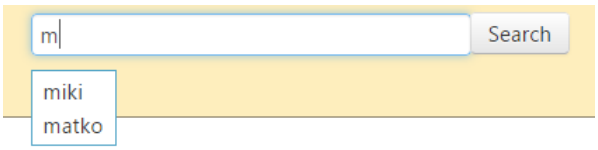
U controller-u imamo funkciju *izbornici* koja služi za micanje loga i naslova sa svake nove stranice te dodaje izbornike i podizbornike na vrh stranice. Ona isto tako pomoću naredbe *T.force* mijenja jezik na kojem je stranica ako korisnik odabere drugi jezik. Nju pozivamo u svakoj funkciji za svaku stranicu (index i kontakt). Funkcija *kontakt* služi samo za prikaz rada prazne stranice izbornika.
Funkcija *index* je glavna funkcija i ona se veže za view index. U njoj isto pozivamo funkciju izbornici i radimo formu nad bazom pomoću web2py naredbe *SQLFORM* te odmah provjeravamo da li je nešto uneseno u formu ili ne, te tu formu vraćamo pomoću rječnika kojeg poslije čitamo u view-u.
Funkcije *zbroj*, *trazi* i *ispis* poziva ajax i za njih ne moramo definirati posebni view, već moramo definirati jedan HTML element za svaku od njih i u ajax funkciju navesti id elementa da funkcija zna kamo mora vratiti rezultat. Funkcija *ispis* pomoću naredbe *request.vars.ime_elementa* dohvaća dvije varijable za koje prvo provjerava jesu brojevi ili slova, te ako su uneseni brojevi onda vraća njihov zbroj. Funkcije *trazi* služi za prijedlog imena pomoću kojeg pretražujemo bazu podatka, a slika 7 prikazuje kako to izgleda. Funkcija vraća DIV elementa na kojem se nalaze pronađena imena u bazi podataka. Svako ime se nalazi u svom DIV elementu kojem je dodijeljen događaj *onclick* s pomoću kojeg, kada se klikne na ime, ono se pojavljuje u formi za pretraživanje baze. Pomoću funkcije *ispis* ispisujemo pronađeni podatak iz baze u tablicu podataka.

```
1  # -*- coding: utf-8 -*-
2
3  def index():
4      izbornici(response)
5
6      forma = SQLFORM(baza.Imenik)
7      if forma.accepts(request.vars):
8          response.flash = T('Uspješno ste dodali kontakt!')
9      elif forma.errors:
10         response.flash = T('Greška u ispunjavanju!')
11
```

```
12      return dict(forma=forma)
13
14  def kontakt():
15      izbornici(response)
16      return dict()
17
18  #---------------------------------
19
20  #   ajax funkcije
21
22  #---------------------------------
23  def zbroj():
24      prvi=request.vars.prvi
25      drugi=request.vars.drugi
26      try:
27          int(prvi)
28          int(drugi)
29          return int(prvi)+int(drugi)
30      except:
31          str(prvi)
32          str(drugi)
33          response.flash=T('Morate unjeti broj')
34          return ''
35
36  def trazi():
37      if not request.vars.trazi:
38          return SCRIPT("jQuery('#suggestions').hide();")
39
40      uzorak = request.vars.trazi.decode('utf8').lower()+'%'
41      nadeno = [row.Ime for row in baza(baza.Imenik.Ime.like(
            uzorak)).select(baza.Imenik.Ime)]
42      if nadeno:
43          return DIV( * [DIV(k,
44              _onclick="jQuery('\#trazi').val('%s');jQuery('\#
                    suggestions').hide();" % k,
45                _onmouseover="this.style.backgroundColor='yellow'",
46                _onmouseout="this.style.backgroundColor='white'"
47                ) for k in nadeno])
48
49  def ispis():
50      T.force(request.uri_language)
51      ime=request.vars.trazi.decode('utf8')
52      nadeno=baza(baza.Imenik.Ime==ime).select(baza.Imenik.Ime,
            baza.Imenik.Prezime,baza.Imenik.Telefon, baza.Imenik.
            Adresa).first()
53
54      tablica=TABLE(TR(T('Ime'),nadeno.Ime),
55                    TR(T('Prezime'),nadeno.Prezime),
56                    TR(T('Telefon'),nadeno.Telefon),
```

```
57                      TR(T('Adresa'),nadeno.Adresa),
58                       _class="table table-striped table-bordered")
59      return tablica
60
61  #--------------------------------------
62
63  #    pomocne funkcije
64
65  #--------------------------------------
66  def izbornici(response):
67      response.logo=False
68      response.title=None
69      T.force(request.uri_language)
70      podMenu=[]
71      podMenu.append((IMG(_src=URL(r=request, c='static', f='
            images/HR.png')),
72                                  True, URL(language='hr')))
73      podMenu.append((IMG(_src=URL(r=request, c='static', f='
            images/UK.png')),
74                                  True, URL(language='en')))
75      response.menu=[]
76      response.menu.append((IMG(_src=URL(r=request, c='static', f=
            'images/home.png')),
77                              True, URL(c='default',f='index')))
78
79      response.menu.append((T('FSB'), True, 'http://www.fsb.unizg.
            hr/'))
80      response.menu.append((T('Kontakt'), True, URL('kontakt')))
81      response.menu.append((T('Jezik'), True, URL(''), podMenu))
```

Primjer 2: default_controller



Slika 7: Prijedlog pretraživanja

U view-u imamo samo index.html view koji nam predstavlja našu stranicu. U njemu smo definirali predložak web stranice koji želimo koristiti. Pomoću naredbe *block head* dodali smo u "head" dokumenta css koji želimo koristiti i jQuery skriptu koja nam pri pokretanju stranice inicijalizira tabove koje koristimo na stranici. Sve ostalo u index-u su standardni HTML elementi sa svojim svojstvima osim ajax-a koji nam poziva funkcije iz index controller-a. Ajax kao argumente prima adresu funkcije gdje se nalazi tj. url do nje, imena argumenata koje želimo poslati i id elementa na kojem želimo prikazati rezultate. URL do funkcije je najbolje pisati pomoću web2py naredbe *URL* jer ako mijenjamo jezik, domenu ili početnu aplika-

ciju u web2py mijenja nam se URL stranice i ajax više nezna gdje se funkcija nalazi.

```
{{extend 'layout.html'}}

{{block head}}
<link rel="stylesheet" type="text/css" href="{{=URL('static', 'css/moj.css')}}">
<script type="text/javascript">
$(document).ready(function(){
  $("#tabs").tabs();});
</script>
{{end}}

<div id="container">
  <div class="tabbable" id="tabs">
    <ul class="nav nav-tabs">
      <li class="active"><a href="#tab1" data-toggle="tab">{{=T('Imenik')}}</a></li>
      <li><a href="#tab2" data-toggle="tab">{{=T('Dodavanje kontakata')}}</a></li>
      <li><a href="#tab3" data-toggle="tab">{{=T('Zbroji dva broja')}}</a></li>
    </ul>

    <div class="tab-content">
      <!-- Imenik -->
      <div class="tab-pane active" id="tab1">
        <table class="sredina">
          <tbody>
            <tr>
              <td>
                <input name="trazi" type="text" placeholder="{{=T('Pretraži imenik')}}"
                  id="trazi">
                <input type="button" class="btn spoji" value="{{=T('Pretraži')}}"
                  onclick="ajax('{{=URL('default', 'ispis')}}',['trazi'],'prikaz')" />
                <div style="position: absolute; display: none;" id="suggestions" class="suggestions"></div>
              </td>
            </tr>

            </tr>
              <td>
                <div id="prikaz" class="prikaz">

                </div>
              </td>
```

```
40              </tr>
41            </tbody>
42          </table>
43
44        </div>
45
46        <!-- Dodavanje kontakata -->
47        <div class="tab-pane" id="tab2">
48          {{=forma}}
49        </div>
50
51        <!-- Zbroji dva broja -->
52        <div class="tab-pane" id="tab3">
53          <table>
54            <tbody>
55              <tr>
56                <th>{{=T('Unesi prvi broj')}}:</th>
57                <td> <input name="prvi" type="text"> </td>
58              </tr>
59
60              <tr>
61                <th>{{=T('Unesi drugi broj')}}:</th>
62                <td> <input name="drugi" type="text"> </td>
63              </tr>
64
65              <tr>
66                <th></th>
67                <td>
68                  <input type="button" value="{{=T('Zbroji')}}"
                        class="btn"
69                  onclick="ajax('{{=URL('default', 'zbroj')}}',['
                        prvi', 'drugi'],'suma_dva')" />
70                </td>
71              </tr>
72
73              <tr>
74                <th>{{=T('Rezultat')}}:</th>
75                <td>
76                  <span id="suma_dva"></span>
77                </td>
78              </tr>
79            </tbody>
80          </table>
81        </div>
82
83      </div>
84    </div>
85  </div>
86  <script>
```

```
87    $("#trazi").keyup(function(){
88      $("#suggestions").css("display", "");
89      ajax("{{=URL('default', 'trazi')}}", ['trazi'], 'suggestions
            ');
90    });
91  </script>
```

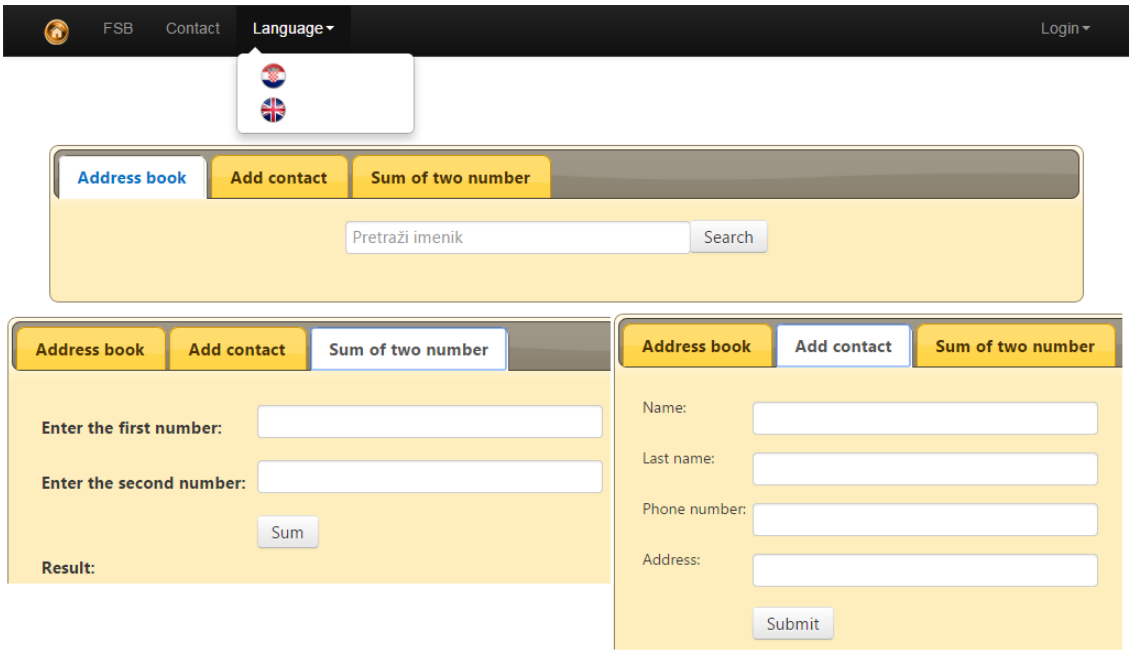
Primjer 3: index.html

Uz pomoć routes.py možemo postavljati početnu aplikaciju, mijenjati jezik na kojem će se prikazivati naša stranica, te postavljati različite domene. Ovdje je pokazano u primjeru 4 kako uz pomoć naredbe *default_application* mijenjamo početnu aplikaciju, te je pokazano da se jezik mijenja tako da prvo navedemo aplikaciju u kojoj želimo mijenjati jezik, a zatim u rječniku navedemo koje jezike nudimo, te koji jezik je početni kada se aplikacija pokrene.

```
1  routers = dict(
2      BASE  = dict(default_application='primjer'),
3      primjer = dict(languages=['en', 'hr'], default_language='hr'
           ),
4  )
```

Primjer 4: routes.py

Na slici 8 možemo vidjeti primjer web2py aplikacije koja je načinjena od prethodno navedenih primjera controller, view-a i modela.

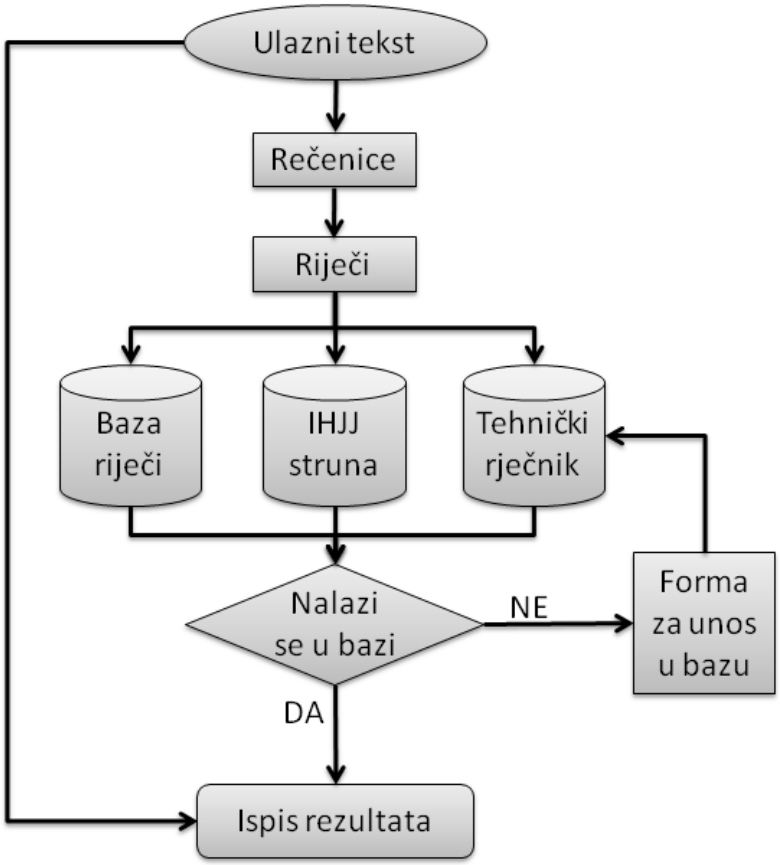


Slika 8: Primjer web2py aplikacije

4 OD ZAMISLI DO REALIZACIJE

Ideja se zasniva na tehničkom riječima koje se nalaze u nekom tekstu kojeg unosi korisnik, a koji se potom obrađuje. Postojanje svake riječ provjerava se u tri baze podataka (baza riječi, IHJJ struna i tehnički rječnik) te će se na temelju toga određivati postoji li određeni pojam u bazi podataka. Ako pojma nema, korisniku će se ponuditi forma za unos pojma u tehnički rječnik pod uvjetom da korisnik zna o kojem se tehničkom pojmu radi. Ako se pojam nalazi u bazi, onda će se korisniku pojaviti tekst koji je unesen i rezultati pronađenih pojmova za moguću promjenu.

Slika 9 ilustrira tok podataka od unosa teksta do prikaza rezultata i unosa u bazu podataka.



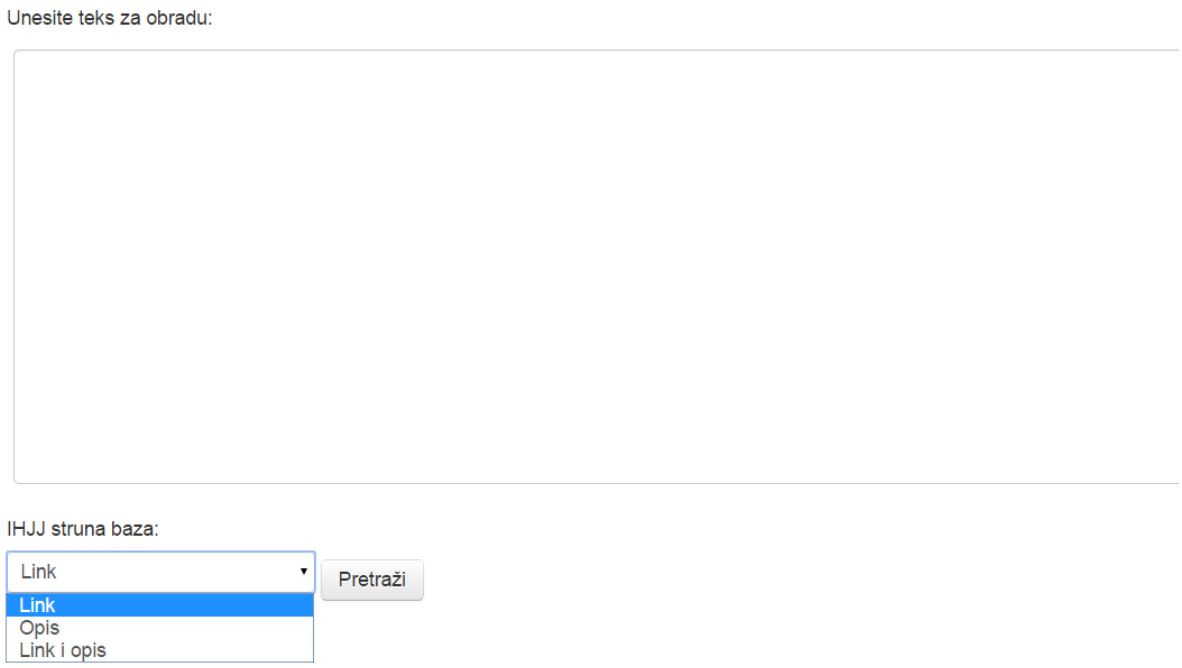
Slika 9: Skica ideje

4.1 Unos teksta za obradu

Unos teksta za obradu odvijati će se uz pomoć forme (obrasca). Forma će se sastojati od polja za unos teksta za obradu, koji će se jednostavno moći kopirati iz neke datoteke ili dovući te staviti u formu, i padajućeg izbornika koji će davati mogućnost da se za nađene pojmove, iz IHJJ struna baze, ispisuje njihov opis, link

ili oboje. Za bazu tehnički rječnik ispisivati će se samo definicija pojma.

Izgled zamišljene forme (obrasca) se može vidjeti na slici 10



Slika 10: Forma za unos teksta

4.2 Rastavljanje na rečenice i riječi

Priprava teksta za dohvaćanje informacije iz rečenice postiže se nizom koraka:

1. Tekst se rastavi u rečenice.
2. Rečenice se rastavljaju u riječi i interpunkcijske znakove.

Rastavljanje teksta u rečenice i rečenica u riječi provodi se s dvjema funkcijama koje su zbog specifičnosti hrvatskoga jezika prilagođene iz poznatoga jezikoslovnog modula NLTK[4] (Natural Language ToolKit) kao funkcije imenima:

word_tokenize(tekst_rečenice) i *sent_tokenize(tekst_dokumenta)*.

Funkcija *sent_tokenize()* kao ulazni argument uzima tekst dokumenta, a kao rezultat vraća listu rečenica. Iz te liste rečenica uzima se pojedina rečenica kao ulazni argument za funkciju *word_tokenize()*, koja vraća listu riječi za tu rečenicu. Problemi koje je trebalo riješiti u prvoj funkciji bili su kako otkriti kraj rečenice. To nije moguće samo na temelju točke, uskličnika i upitnika, jer se isti znakovi (npr. točka) mogu pojavljivati i u drugim okolnostima, npr. kad su dio neke kratice (npr. prof. dr. sc.). Stoga je trebalo prikupiti više od 600 kratica koje su česte u hrvatskome jeziku te ugraditi stanovitu heuristiku za slučaje koji nisu kratice, a ne mogu biti ni rečenice (sastavljene od riječi s malim brojem slova, kojih uz to nema u bazi).

4.3 Baze podataka

Baza riječi sadrži jednu tablicu, tablicu s riječima. U njoj je spremljena svaka riječ za sebe, a ne pojmovi kao u druge dvije baze. Svaka riječ je spremljena u bazu sa svojim gramatičkim oblikom, i jednim dodatnim svojstvom koji je u ovom slučaju lema tj. osnovni oblik riječi iz koje je izvedena. Neke riječi mogu imati više gramatičkih oblika pa su oblici onda razdvojeni znakom "/", te pomoću jednostavnog regularnog izraza možemo pretražiti sve gramatičke oblike.

#	id	rijec	tag	dod1	dod2	dod3
157	157	osmjehnuste	gr0j	osmjehnuti	NULL	NULL
158	158	procvilite	gp2m/gm2m	procviljeti	NULL	NULL
159	159	ustajalijem	pmjd0nk/pmjl0nk/psjd0nk/psjl0nk	ustajao	NULL	NULL
160	160	centrirali	gr0m	centrirati	NULL	NULL
161	161	tornadima	immd0/imml0/immi0	tornado	NULL	NULL
162	162	brbljavoj	pzjd0op/pzjl0op	brbljav	NULL	NULL
163	163	brbljavom	pzji0op	brbljav	NULL	NULL
164	164	čelisticama	izmd0/izml0/izmi0	čelistica	NULL	NULL
165	165	tustima	pmmd0op/pmml0op/pmmi0op/psmd0op/psml0op/psmi0op/pzmd0op/pzml0op/pzmi0op/pmmd0np...	tust	NULL	NULL
166	166	jednogodišnjaci	immn0/immv0	jednogodišnjak	NULL	NULL

Slika 11: Baza riječi

IHJJ struna baza se također sastoji od jedne tablice. U tablicu su spremljeni pojmovi hrvatskog strukovnog nazivlja sa svojim opisom, linkom koji pokazuje na IHJJ[7] stranicu gdje se pojam nalazi, te stupcima s osnovnom gramatikom (glagoli, imenice i nepoznato). U stupce glagol i imenice spremaju se riječi koje čine pojmove, a koji se nalaze u bazi riječi, a u stupac nepoznato se spremaju riječi iz pojma koje su nepoznate, tj. ne nalaze se u bazi riječi.

#	id	natuknica	opis	link	glagoli	imenice	nepoznato
1	1	abaksijalan	koji je nasuprot osi tijela	http://struna.ihjj.hr/naziv/abaksijalan/16717/#naziv	NULL	NULL	[u'abaksijalan']
2	2	abakterijski	koji je bez bakterija	http://struna.ihjj.hr/naziv/abakterijski/16780/#naziv	NULL	NULL	[u'abakterijski']
3	3	abapikalan	koji je suprotno od vrha zubnoga korijena	http://struna.ihjj.hr/naziv/abapikalan/16657/#naziv	NULL	NULL	[u'abapikalan']
4	4	aberacija	odstupanje od normalna rasta i razvoja	http://struna.ihjj.hr/naziv/aberacija/14186/#naziv	NULL	NULL	[u'aberacija']
5	5	aberantan	koji odstupa od normalna oblika ili smjera	http://struna.ihjj.hr/naziv/aberantan/16669/#naziv	NULL	NULL	[u'aberantan']
6	6	aberantna žlijezda slinovnica	žljezdano tkivo koje se razvija na neuobičajenim	http://struna.ihjj.hr/naziv/aberantna-zlijezda-slinovnica/1	NULL	NULL	[u'aberantna', u'slinovnica']
7	7	abfrakcija	patološki gubitak tvrdih zubnih tkiva prouzročen t	http://struna.ihjj.hr/naziv/abfrakcija/17303/#naziv	NULL	NULL	[u'abfrakcija']
8	8	ablacija	nestajanje leda ili snijega zbog otapanja i ispari	http://struna.ihjj.hr/naziv/ablacija/23019/#naziv	NULL	NULL	[u'ablacija']
9	9	aboralan	koji se nalazi dalje od usta	http://struna.ihjj.hr/naziv/aboralan/16664/#naziv	NULL	NULL	[u'aboralan']
10	10	abortivni zub	trajni zub prerano izgubljen zbog karijesa	http://struna.ihjj.hr/naziv/abortivni-zub/16704/#naziv	NULL	NULL	[u'abortivni']

Slika 12: IHJJ struna

Tehnički rječnik se sastoji od četiri tablice. Glavna tablica se zove pojam i u nju se spremaju sljedeći podatci: naziv pojma, definicija i napomena. Ostale tablice se odnose na polja u formi koja se mogu dodavati više puta (vidjeti sliku 14), pa tako imamo tablice kontekst, razredba i istovrijednice.
Tablica razredba se sastoji od područja, polja i grane. U **tablici istovrijednica** se nalaze polja jezik i pojma, a u **tablici kontekst** polje za kontekst. Ostale tablice su povezane s glavnom tablicom preko ID-a pojma u glavnoj tablici, tj. u svaku od ostalih tablica dodano je još jedno polje naziva *pojamID* koje nam govori na koji

pojam se podatak odnosi.

#	Name	Data type	P	F	U	H	N	C	Default value	
1	id	INTEGER	🔑						NULL	
2	PojamIme	CHAR(512)							NULL	TABLICA POJAM
3	Definicija	TEXT							NULL	
4	Napomena	TEXT							NULL	

#	Name	Data type	P	F	U	H	N	C	Default value	
1	id	INTEGER	🔑						NULL	
2	PojamID	INTEGER							NULL	
3	RazredbaPodrucje	CHAR(512)							NULL	TABLICA RAZREDBA
4	RazredbaPolje	CHAR(512)							NULL	
5	RazredbaGrana	CHAR(512)							NULL	

#	Name	Data type	P	F	U	H	N	C	Default value	
1	id	INTEGER	🔑						NULL	
2	PojamID	INTEGER							NULL	TABLICA KONTEKS
3	KontekstKontekst	TEXT							NULL	

#	Name	Data type	P	F	U	H	N	C	Default value	
1	id	INTEGER	🔑						NULL	
2	PojamID	INTEGER							NULL	TABLICA
3	IstovrijedniceJezik	CHAR(512)							NULL	ISTOVRIJEDNICE
4	IstovrijednicePojam	CHAR(512)							NULL	

Slika 13: Tehnički rječnik

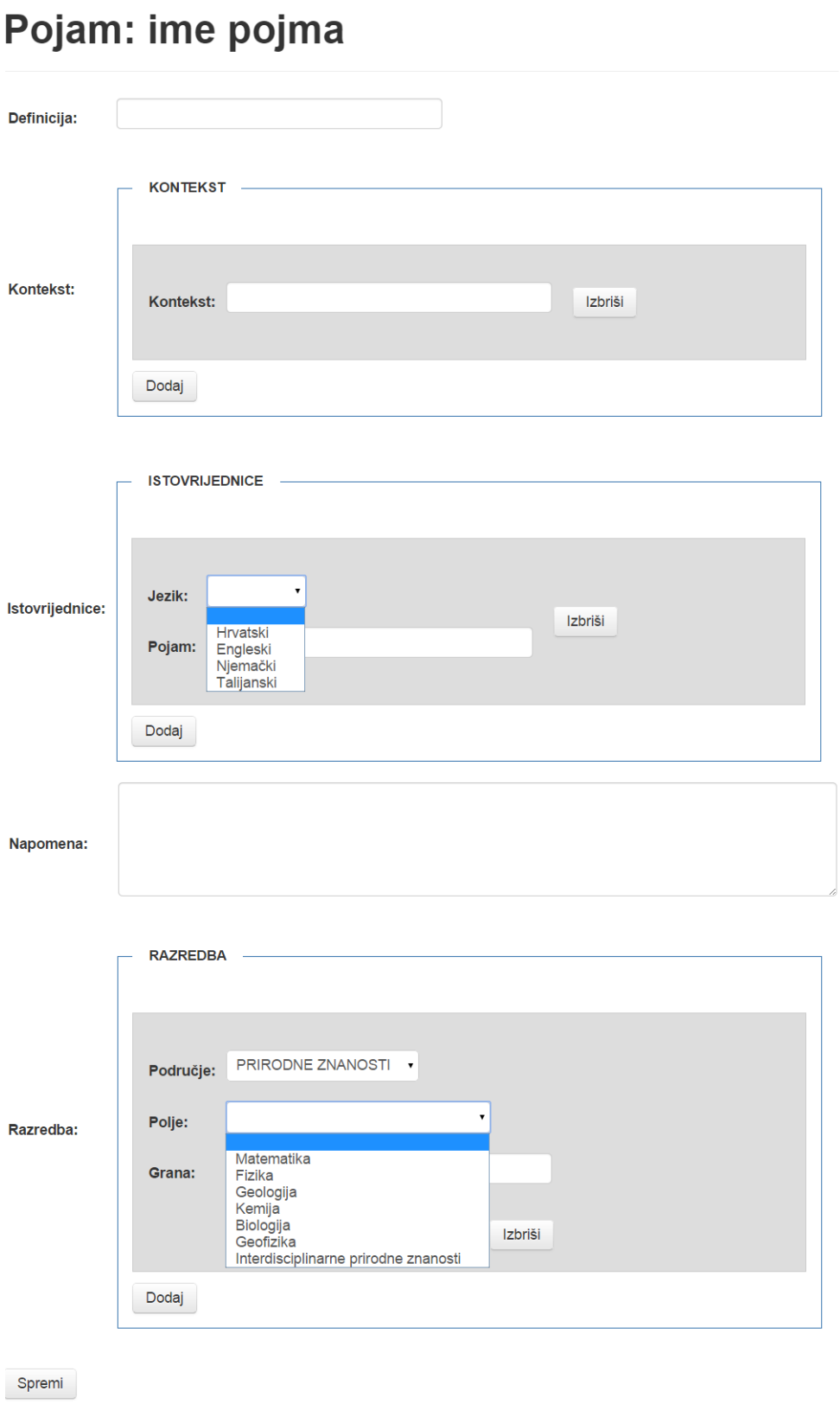
4.4 Obrasci (forma) za unos pojmova

Forma za unos pojma sačinjavati će se od sljedećih polja za unos podataka:

1. definiciju pojma
2. kontekst u kojem se taj pojma može koristiti
3. istovrijednice, tj. naziv toga pojma na hrvatskom i drugim jezicima
4. napomenu
5. u razredbu se unosi glavno područje, polje (fizika, kemija...) kojem taj pojam pripada, te grana (opća fizika, fizika elementarnih čestica...) kojoj pripada

Forma neće dopuštati da se jedan pojam unese više puta. Kada se u formu unese pojam koji već postoji, on će prvo popuniti sva polja u formi koja su bila ispunjena prilikom njegovo zadnjeg spremanja u bazu, te će forma samo ažurirati novo unesene podatke.

Izgled forme može se vidjeti na slici 14.



Slika 14: Forma za unos pojmova u bazu podataka

4.4.1 Podjela područja, polja i grane

Na temelju članka 115. stavka 5. Zakona o znanstvenoj djelatnosti i visokom obrazovanju[8] znanstvena i umjetnička područja su:

1. prirodne znanosti
2. tehničke znanosti
3. biomedicina i zdravstvo
4. biotehničke znanosti
5. društvene znanosti
6. humanističke znanosti
7. umjetničko područje
8. interdisciplinarna područja znanosti
9. interdisciplinarna područja umjetnosti

Znanstvena područja dijele se na znanstvena polja, a polja se dalje dijele na grane.

1. **PODRUČJE PRIRODNIH ZNANOSTI**

1. 1. **Matematika**
Grane:
- 1. 1. 1. algebra
- 1. 1. 2. geometrija i topologija
- 1. 1. 3. diskretna i kombinatorna matematika
- 1. 1. 4. matematička analiza
- 1. 1. 5. matematička logika i računarstvo
- 1. 1. 6. numerička matematika
- 1. 1. 7. primijenjena matematika i matematičko modeliranje
- 1. 1. 8. teorija vjerojatnosti i statistika
- 1. 1. 9. financijska i poslovna matematika
- 1. 1. 10. ostale matematičke discipline

1. 2. **Fizika**
Grane:
- 1. 2. 1. opća i klasična fizika
- 1. 2. 2. fizika elementarnih čestica i polja
- 1. 2. 3. nuklearna fizika
- 1. 2. 4. atomska i molekulska fizika
- 1. 2. 5. fizika kondenzirane tvari
- 1. 2. 6. astronomija i astrofizika
- 1. 2. 7. biofizika i medicinska fizika

1. 3. **Geologija**
Grane:
- 1. 3. 1. geologija i paleontologija
- 1. 3. 2. mineralogija i petrologija

1. 4. **Kemija**
Grane:
- 1. 4. 1. fizikalna kemija
- 1. 4. 2. teorijska kemija
- 1. 4. 3. analitička kemija
- 1. 4. 4. anorganska kemija
- 1. 4. 5. organska kemija

1. 4. 6. biokemija i medicinska kemija
1. 4. 7. primijenjena kemija

1. 5. **Biologija**
Grane:
1. 5. 1. biokemija i molekularna biologija
1. 5. 2. botanika
1. 5. 3. mikrobiologija
1. 5. 4. zoologija
1. 5. 5. ekologija
1. 5. 6. genetika, evolucija i filogenija
1. 5. 7. opća biologija

1. 6. **Geofizika**
Grane:
1. 6. 1. Meteorologija s klimatologijom
1. 6. 2. Fizička oceanografija
1. 6. 3. Seizmologija i fizika unutrašnjosti Zemlje
1. 6. 4. Ostale geofizičke discipline

1. 7. Interdisciplinarne prirodne znanosti
Grane:
1. 7. 1. metodike nastavnih predmeta prirodnih znanosti
1. 7. 2. znanost o moru
1. 7. 3. znanost o okolišu
1. 7. 4. znanost o zračenju

2. PODRUČJE TEHNIČKIH ZNANOSTI

2. 1. **Arhitektura i urbanizam**
Grane:
2. 1. 1. arhitektonsko projektiranje
2. 1. 2. urbanizam i prostorno planiranje
2. 1. 3. arhitektonske konstrukcije, fizika zgrade, materijali i tehnologija građenja
2. 1. 4. povijest i teorija arhitekture i zaštita graditeljskog naslijeđa
2. 1. 5. pejsažna arhitektura

2. 2. **Brodogradnja**
Grane:
2. 2. 1. konstrukcija plovnih i pučinskih objekata
2. 2. 2. hidromehanika plovnih i pučinskih objekata
2. 2. 3. osnivanje plovnih i pučinskih objekata
2. 2. 4. tehnologija gradnje i održavanje plovnih i pučinskih objekata

2. 3. **Elektrotehnika**
Grane:
2. 3. 1. elektroenergetika
2. 3. 2. elektrostrojarstvo
2. 3. 3. elektronika
2. 3. 4. telekomunikacije i informatika
2. 3. 5. radiokomunikacije
2. 3. 6. automatizacija i robotika

2. 4. **Geodezija**
Grane:
2. 4. 1. kartografija
2. 4. 2. fotogrametrija i daljinska istraživanja
2. 4. 3. pomorska, satelitska i fizikalna geodezija
2. 4. 4. primijenjena geodezija
2. 4. 5. geomatika

2. 5. **Građevinarstvo**
Grane:
2. 5. 1. geotehnika
2. 5. 2. nosive konstrukcije
2. 5. 3. hidrotehnika

2. 5. 4. prometnice
2. 5. 5. organizacija i tehnologija građenja

2. 6. **Grafička tehnologija**
Grana:
2. 6. 1. procesi grafičke reprodukcije

2. 7. **Kemijsko inženjerstvo**
Grane:
2. 7. 1. reakcijsko inženjerstvo
2. 7. 2. mehanički, toplinski i separacijski procesi
2. 7. 3. analiza, sinteza i vođenje kemijskih procesa
2. 7. 4. kemijsko inženjerstvo u razvoju materijala
2. 7. 5. zaštita okoliša u kemijskom inženjerstvu

2. 8. **Metalurgija**
Grane:
2. 8. 1. procesna metalurgija
2. 8. 2. mehanička metalurgija
2. 8. 3. fizička metalurgija

2. 9. **Računarstvo**
Grane:
2. 9. 1. arhitektura računalnih sustava
2. 9. 2. informacijski sustavi
2. 9. 3. obradba informacija
2. 9. 4. umjetna inteligencija
2. 9. 5. procesno računarstvo
2. 9. 6. programsko inženjerstvo

2. 10. **Rudarstvo, nafta i geološko inženjerstvo**
Grane:
2. 10. 1. rudarstvo
2. 10. 2. naftno rudarstvo
2. 10. 3. geološko inženjerstvo

2. 11. *Strojarstvo*
Grane:
2. 11. 1. opće strojarstvo (konstrukcije)
2. 11. 2. procesno energetsko strojarstvo
2. 11. 3. proizvodno strojarstvo
2. 11. 4. brodsko strojarstvo
2. 11. 5. precizno strojarstvo

2. 12. **Tehnologija prometa i transport**
Grane:
2. 12. 1. cestovni i željeznički promet
2. 12. 2. pomorski i riječni promet
2. 12. 3. poštansko-telekomunikacijski promet
2. 12. 4. zračni promet
2. 12. 5. inteligentni transportni sustavi i logistika

2. 13. **Tekstilna tehnologija**
Grane:
2. 13. 1. tekstilno-mehaničko inženjerstvo
2. 13. 2. tekstilna kemija
2. 13. 3. odjevna tehnologija
2. 13. 4. dizajn tekstila i odjeće

2. 14. **Zrakoplovstvo, raketna i svemirska tehnika**
Grane:
2. 14. 1. konstrukcija i osnivanje letjelica
2. 14. 2. zrakoplovne tehnologije i održavanje
2. 14. 3. vođenje i upravljanje letjelicama

2. 15. **Temeljne tehničke znanosti**
Grane:
2. 15. 1. automatika
2. 15. 2. energetika
2. 15. 3. materijali
2. 15. 4. mehanika fluida
2. 15. 5. organizacija rada i proizvodnje
2. 15. 6. tehnička mehanika (mehanika krutih i deformabilnih tijela)
2. 15. 7. termodinamika

2. 16.	**Interdisciplinarne znanosti** Grane:	**tehničke**	2. 16. 1. inženjerstvo okoliša 2. 16. 2. mikro i nanotehnologije

4.5 Prikaz rezultata

Rezultati se ispisuju na DIV element tako da je svaka riječ u svojem SPAN html elementu. Crvenom bojom će biti označene riječi koje se ne nalaze ni u jednoj bazi. Zeleno su označene riječi koje se nalaze u **IHJJ struna bazi** ili **tehničkom rječniku**. Slika 15 pokazuje kako izgleda prvobitni ispis nakon obrade podatka.

Slika 15: Ispis nakon obrade teksta

Svaka riječ bila ona crvena ili zelena imam pridruženo svojstvo *tooltip-a*. Pomoću njega se omogućuje daljnja interakcija s korisnikom. Za zelene riječi na njemu se prikazuje definicja pojma ili link zavisno o tome što je korisnik izabrao i forma koja omogućuje da se promijeni ime pojam ako mislimo da je krivo. Kod crvenih riječi *tooltip* ima samo formu za unos pojma u bazu podataka. Slika 16 prikazuje kako izgledaju riječi sa *tooltip*-om.

Slika 16: Ispis nakon korisnikovog uključivanja tooltip-a

5 IZVLAČENJE INFORMACIJE IZ TEKSTA

Da bi smo mogli izvlačiti informacije iz teksta moramo proći kroz sljedeće korake:

1. Rastaviti tekst na rečenice i riječi
2. Pridružiti riječi s njihovim gramatičkim oblicima iz **baze riječi**
3. Sastaviti uzorke i izraze
4. Pretražiti tekst uz pomoć uzoraka i izraza

5.1 Pridruživanje gramatičkih oblika riječima

Nakon što je načinjena lista riječi iz nekoga zadanog teksta, moramo za te riječi pronaći njihove tagove u **bazi riječi** i spremiti ih jednoznačnim pridruživanjem, stvarajući na taj način listu tagova. Riječ iz liste riječi i pripadni tag/ovi u listi tagova imaju isti indeks. Algoritam koji to radi prikazan je u Algoritmu 5.

```
1  class Greska(Exception):
2    pass
3
4  class Baza():
5    def __init__(self, path):
6      baza_connect=sqlite3.connect(path)
7      self.baza=baza_connect.cursor()
8      self.lista_stupaca=['rijec','tag','dod1','dod2','dod3']
9
10   def pretrazi(self, rijec, stupac):
11     if stupac in self.lista_stupaca:
12       trazi="SELECT "+stupac+" FROM Rijeci WHERE rijec='"+rijec+
           "';"
13       self.baza.execute(trazi)
14       pod=self.baza.fetchall()
15       if pod:
16         return pod[0][0]
17       else:
18         return None
19     else:
20       raise Greska('NE POSTOJI STUPAC '+stupac)
21
22
23 lista_obradeno=[]
24 rijeci=word_tokenize(recenica) #lista riječi iz rečenice
25
26 for r in rijeci: #za sve riječi u listi
27   podatak=baza.pretrazi(r.lower(),stupac) #pretraživanje baze za
       pojedinu riječ i njenog svojstva
28   if podatak is None: #ako riječ nije u bazi dodaj u listu #
29     lista_obradeno.append('#')
```

```
30    else:
31      lista_obradeno.append(podatak) #inače spremi tag
32
33  print lista_obradeno
```

Primjer 5: Algoritam za pridruživanje tagova iz baze podataka

S istim programom/algoritmom mogu se pridruživati i ostali tagovi /sintaksni ili semantički/ iz baze, ako su spremljeni za tu riječ.Uzmimo npr. rečenicu Uz našu kuću raste zeleni bršljan već treću godinu. za koju želimo naći tagove za pojedine riječi.

```
[[u'Uz', u'na\u0161u', u'ku\u0107u', u'raste', u'zeleni', u'br\u0161ljan', u've\u0107',
 u'tre\u0107u', u'godinu', u'.'], [u'saj', u'imjd0/imjv0/imjl0', u'izja0', u'imjv0/gp3j'
, u'izjg0/izjd0/izjv0/izjl0/izji0/pmjv0op/pmmn0op/pmmv0op/pmjn0np/pmja0np/pmjv0np
/pmmn0np/pmmv0np/gp3j/gm2j', u'imjn0/imja0', u'vus/vn', '#', u'izja0', u'q.']]
```

Slika 17: Prikaz tagova za danu rečenicu

Na slici 17 vidimo da neke riječi mogu imati više tagova odvojenih znakom „/”, što znači da jedna riječ u nekom kontekstu može biti imenica, glagol ili pridjev ili da se isto piše u različitim padežima, licima, brojevima. Kao što možemo vidjeti, riječ raste ima tag „imjv0/gp3j” koji govori da riječ raste može biti prezentski oblik glagola rasti sa značenjem 'postajati većim ili višim...' ili vokativ imenice rast sa značenjem 'povećavanje jedinke...' Znak „# ” govori nam da ta riječ ne postoji u bazi podataka, u ovom slučaju riječ treću nije spremljena u bazu podataka.

5.2 Sastavljanje uzoraka i izraza

Sastavljanje uzoraka i izraza temelji se na regularnim izrazima (eng. regular expression). To je način zadavanja općih oblika preko programskih kratica, kako bi se iz teksta (niza znakova, eng. string) mogla izvući struktura, a ne samo znakovi. Tipičan je primjer dohvaćanje e-adrese iz nekog teksta. Iako adresa može biti gotovo beskonačno, sve one imaju istu strukturu (ne samo u znaku @ koji odjeljuje ime osobe od računalne domene). Moguće je s pomoću regularnih izraza (npr. `\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b`) napisati program koji će iz teksta izvlačiti bilo koju e-adresu. Pisanje regularnih izraza nije jednostavno, ali jednom dobro napisani služe zauvijek, osim poboljšavanja njihova filtriranja, nije ih potrebno mijenjati. Za potrebe našega programa kojim izvlačimo bilo koju informaciju iz dokumenata nužno je dobro poznavanje kako problematike (sintakse hrvatskoga jezika), tako i regularnih izraza za pisanje uzoraka kojim se slovni i gramatički oblici dohvaćaju. Štoviše, povezivanje uzoraka u sintaksno-semantičku cjelinu također se provodi regularnim izrazima (druga razina izvlačenja informacije), pa je njihova uporaba nezaobilazna. Naš program kao ulazne argumente prima niz uzoraka i izraza kojima se onda djeluje na pojedinačne rečenice iz teksta (dokumenta).

5.2.1 Uzorci

Uz pomoć (s regularnim izrazima) napisanih uzoraka pretražujemo tekst na osnovi slova i gramatičkih obilježja riječi. Moguće je birati neko od ponuđenih svojstava riječi ili uz dodatne argumente definirati kontekst (okoliš) riječi. Svaki uzorak sastoji se od triju osnovnih argumenata i deset mogućih dodatnih argumenata koji se stavljaju po volji korisnika. Opći oblik uzorka izgleda ovako:

('Riječ', 'Gramatički oblik', Logika, 'Dodatni argumenti').

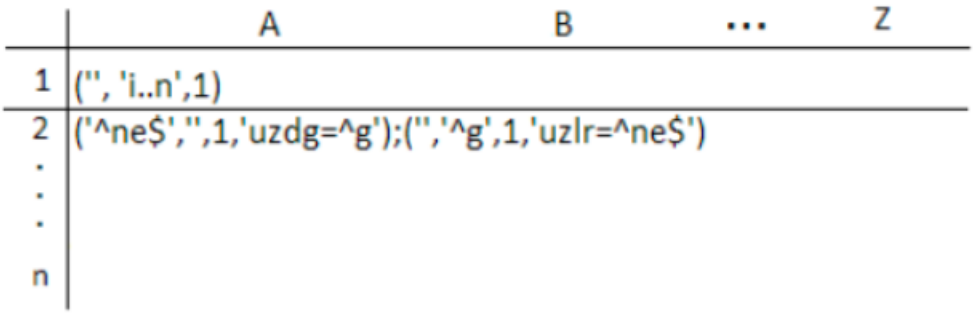
Prvi argument „Riječ" odnosi se na regularni izraz kojim pretražujemo riječi u nekom tekstu na temelju slovnih znakova. Drugi argument „Gramatički oblik" omogućuje nam da pretražujemo riječi po gramatičkoj osnovi, na primjer, da tražimo imenicu muškoga roda ili glagol u prezentu jednine ili zamjenicu itd. Treći argument „Logika" može imati vrijednost 1 ili 0, što znači da se uzima u obzir traženje i po slovnom i po gramatičkom obliku (1) ili bilo kojem od njih (0). Zadnji argument može imati više svojstava koja se vide u tablici 2.:

Tablica 2: Opis dodatnih svojstava uzoraka

Svojstva	Značenje
lg	lijevo od nađene riječi traži se gramatika
lr	lijevo od nađene riječi traži se riječ
dg	desno od nađene riječi traži se gramatika
dr	desno od nađene riječi traži se riječ
uzlg	neposredno lijevo uz nađenu riječ traži se gramatika
uzlr	neposredno lijevo uz nađenu riječ traži se riječ
uzdg	neposredno desno uz nađenu riječ traži se gramatika
uzdr	neposredno desno uz nađenu riječ traži se riječ
imag	traži se da u rečenici bude riječ sa zadanom gramatikom
imar	traži se da u rečenici bude tražena riječ

Slika 18. pokazuje kako se pravilno unose uzorci. U njima se mogu koristiti sve oznake regularnih izraza kako bi se ispravno definiralo što se želi naći u zadanom

tekstu. Uzorci se odvajaju znakom ';' i navode u n-redaka. Svaki redak može imati najviše 26 uzoraka (označenih slovima engleske abecede radi lakše daljnje obrade u programu kada se pozivaju izrazi).

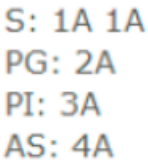


Slika 18: Prikaz zadavanja uzoraka

5.2.2 Izrazi

Sintaksno-semantički izraz sastoji se općenito od niza riječi (dohvaćenih na temelju njihova slovnog ili gramatičkog obilježja). Izraz se dohvaća iz rečenice na temelju regularnoga izraza kojim se povezuju gore opisani uzorci. Uzorak je jednoznačno opisan svojim položajem (u retku i stupcu), pa će 1A značiti prvi (A) uzorak iz prvoga retka, 3B će značiti drugi (B) uzorak iz trećega retka, 4C će značiti treći uzorak iz četvrtoga retka i slično.

Slijedi faza povezivanja uzoraka u izraze. Što će naš izraz tražiti, u ovom je času nevažno. Može se na primjer pokazati traženje službe riječi u rečenici (subjekta, objekta, predikata, atributa sročnog i nesročnog, priložnih oznaka i slično). Svaki od izraza može imati ime ili kraticu, da bi se kod rezultata jasno razabralo što je program uspio pronaći. Izrazi se sastavljaju tako da se prvo napiše naziv ili ime izraza, zatim stavimo znak dvotočke (':') iza čega slijedi kombinacija uzoraka označenih njihovim kraticama (1A, 3B i sl.), kako pokazuje slika 19.



Slika 19: Prikaz zadavanja izraza

5.3 Pretraživanje teksta i dohvaćanje informacije

Konačno, nakon što su definirani uzorci i izrazi, pristupa se pozivu programa koji će pretražiti tekst i izvući željenu informaciju. Na primjer, u slici 20 . navedeni su uzorci za dohvaćanje predikata, i to kao pomoćnoga glagola biti uz njega glagolski

pridjev radni, zatim uz riječ 'ne' koja stoji ispred glagola i glagola u prezentu, infinitivu, aoristu i imperativu.

```
('','^ob',1);('','^gr',1,'lg=^ob');('^ne$','',1,'uzdg=^g');('','^
('','^gp',1);('','^gi',1);('','^gm',1);('','^ga',1)
```

Slika 20: Uzorci za pretraživanje teksta

Algoritam za pretraživanje teksta po uzorcima radi u koracima. Prvo rastavi rečenicu na riječi i zatim iz baze izvadi tagove za pojedinu riječ i spremi ih u listu. Nakon toga uzima riječ po riječ i za svaku od njih pretražuje poklapa li se riječ ili njezin tag sa zadanim. Ako se podudara, onda rezultat spremi u listu. Nakon što je program prošao kroz sve riječi u rečenici, počinje pretraživati sve mogućih kombinacije te rečenice, jer jedna riječ može biti imenica u nekom kontekstu, dok je u drugom glagol ili se ista riječ definira s dva različita uzorka, pa imamo dvije slovne oznake za nju. Nakon što se izračuna broj mogućih kombinacija, program pravu rečenicu zamjenjuje rečenicom sa slovnim oznakama i vraća listu tih rečenica. Kako to izgleda u programu, vidi se u Algoritmu 6.

```
1  def pretraz_recenice(self,recenica, uzorak, sl_oz, stupac):
2    """Pretrazuje svaku rijec u recenici za svaki zadani uzorak
3    Vraća listu svim mogućih kombinacija i listu pozicija u
        recenici"""
4      rijeci=word_tokenize(recenica)      #lista rijeci iz recenice
5      gram_oblik=self.iz_baze([r.lower() for r in rijeci], stupac)
           #gram_oblik za svaku rijec
6
7      pozicija=[] #pozicija rijeci u recenici
8      rez=[]; rez_2=[]     #rez_2 - nadjeno dodatno
9      for r in xrange(len(rijeci)):      #za sve rijeci u recenici,
           r -> pozicija rijeci u recenici
10       rez_1=[]     #rez_1 - nadjeno opcenito;
11         for u in xrange(len(uzorak)):   #za sve uzorke
12           for g in re.split(r'/',gram_oblik[r]): #za sve gram.
                oblike rijeci
13             if self.trazi(rijeci[r].lower(),g,uzorak[u]) is
                   True: #ako je nadjeno nesto
14
15               if len(uzorak[u])>3:   #ako uzorak ima vise od
                     3 argumenta
16                   uzr=uzorak[u][3:len(uzorak[u])]
17                     pom=dodatni_argumenti(rijeci,gram_oblik,
                           r,uzr) #pretrazivanje dodatnih
                           argumenata
18                   if pom:
19                     rez_1.append(sl_oz[u])
```

```
20                      else:
21                        rez_1.append(sl_oz[u])
22          if rez_1:
23              pozicija.append(r)
24              rez.append(list(set(rez_1)))
25
26    return list(it.product( * rez)), pozicija, gram_oblik
```

Primjer 6: Funkcija koja pretražuje tekst po uzorcima

Nakon što program nađe podudaranje uzorka i riječi u tekstu, nju zamjenjuje s oznakom koju izraz ima, kao što je bilo prikazano u 3.5.2., u ovom slučaju to su oznake 1A, 1B, 1C, 1D, 2A, 2B, 2C i 2D, a za ostale riječi koje se ne traže program stavlja znak crtice ('–'), dok za riječi koje nema u bazi podataka stavlja znak povisilice ('#'). Nakon toga dolaze izrazi koji kombiniraju oznake kojima su zamijenjene pronađene riječi i onda na temelju novih podudaranja između riječi i oznaka dobivamo željeni podatak iz teksta. Za slučaj traženja predikata mogli bismo napisati sljedeće izraze, kako se vidi na slici 21.

```
PRED: ((1A)[-|#]*(1B))|(1C1D)|(1A)|(2A)|(2B)|(2C)|(2D)
```

Slika 21: Izraz za traženje predikata u tekstu

Algoritam za pretraživanje izraza radi po sljedećim koracima. Kao ulazni argument program prima listu rečenica u kojima su riječi zamijenjene slovnim oznakama. Funkcija prolazi kroz svaku rečenicu pojedinačno i na njoj isprobava izraze koje smo prethodno bili sastavili, a nakon što je nađeno podudaranje izraza i oznaka u novoj rečenici, program sprema mjesto podudaranja u rečenici zajedno s vrstom izraza. Nakon što je program prošao kroz sve rečenice, vraća listu sa svim tim rezultatima. Kako to izgleda u programskom kodu, može se vidjeti u algoritmu 7.

```
1  def trazenje_po_izr(self,nadjeno_uzr, izr,pozicije):
2      """Trazi koja kombinacija odgovara zadanom izrazu
3      nadjeno_uzr=['X1X0X2',...]  vraca koja kombinacija odgovara,
           pocetak, kraj i reg_izraz"""
4      vise_odv_rj=[];vise_sku_rj=[];jedno_rj=[]
5      for n in nadjeno_uzr:   #za sve kombinacije iz nadjeno_uzr
6          komb=''.join(n)
7          for i in re.split(r'\|',izr): #za svaki izraz iz
               kombinacije
8
9              if '[\d+\D]' in i:  #vise odvojenih rjesenje
10                 for g in re.finditer(i,komb):
11                     pom=[]
12                     for p in xrange(2,len(g.groups())+1):
13                         pom.append(self.poz_u_tekstu(pozicije,
                               komb,g.start(p)))
```

```
14                      if pom:
15                          vise_odv_rj.append((tuple(pom),i))
16
17              elif re.search(r'\(\d+\D\)',i): #jedno rjesenje
18                  for g in re.finditer(i,komb):
19                      jedno_rj.append((self.poz_u_tekstu(pozicije,
                            komb,g.start()),i))
20
21              else:   #vise spojenih rjesenje
22                  for g in re.finditer(i,komb):
23                      c=self.poz_u_tekstu_skupa(pozicije,komb,g.
                            start(), g.end())
24                      if self.niz(c) is True: vise_sku_rj.append((
                            c,i))
25
26      return sorted(list(set(jedno_rj))), sorted(list(set(
            vise_sku_rj))),sorted(list(set(tuple(i) for i in
            vise_odv_rj)))
```

Primjer 7: Funkcija za pretraživanje po izrazima

Nakon što je program pretražio rečenice po uzorcima i izrazima, moraju se još samo ispisati rezultati. Da bi se oni ispisali, prvo moramo dohvatiti riječi koje su se nalazile na mjestima na kojima su se slovne oznake poklopile s izrazima. Njih vadimo s pomoću rezultata koje vraća funkcija iz Tablice 9., tj. uz pomoć početka i kraja mjesta na kojem je program našao podudaranje sa slovnim oznakama. Kako je to realizirano u algoritmu, vidi se u algoritmu 8.

```
1   def rezultat(self,rec, uzr, izr, slovne_oz, objekti, stupac):
2           """rec->recenice, uzr->uzorak izr->izrazi, slovne_oz,
                stupac->po_cemu_se_baza_pretrazuje"""
3           rez=''
4           for r in xrange(len(rec)):
5               rijeci=word_tokenize(rec[r])
6               if self.provjera_uzoraka(uzr) is True:
7                   nadjeno_uzr,pozicije, gram_obl=self.
                        pretraz_recenice(rec[r],uzr,slovne_oz,stupac)
8
9                   for i in xrange(len(izr)):
10                      jedno_rj,vise_skupa,vise_odvojeno=__filtri__
                            (rijeci,*self.trazenje_po_izr(nadjeno_uzr
                            ,izr[i],pozicije))
11
12
13
14                      nema='Nema u bazi: '
15                      for z in xrange(len(gram_obl)):
16                          if gram_obl[z]=='#':
17                              nema+=rijeci[z]+', '
```

```
18
19                      if i==0:
20                          if nema=='Nema u bazi: ': rez+=rec[r]+'\
                                n'
21                          else: rez+=rec[r]+'\n\n'+nema+'\n\n'
22
23                      rez+= objekti[i]+'\n'
24
25                      for j in jedno_rj:
26                          if rijeci[j[0]]=='da': pass      #
                                izbacuje rijec da ako je sama kao
                                rješenje
27                          else:  rez+='\t'+rijeci[j[0]]+'\n'
28
29                      for j in vise_skupa:
30                          tekst=''
31                          for k in j[0]: tekst+=' '+rijeci[k]
32                          rez+='\t'+tekst+'\n'
33
34                      for j in vise_odvojeno:
35                          tekst=''
36                          for k in j[0]: tekst+=' '+rijeci[k]
37                          rez+='\t'+tekst+'\n'
38              rez+='\n'
39              rez+='-' * 30+'\n'
40          return rez
```

Primjer 8: Algoritam za ispis rezultata

Kao što možemo vidjeti, funkcija za ispis rezultata poziva funkciji iz algoritama 6 i 7 te neke dodatne vlastite funkcije za filtriranje podataka. Tek se njihovim povezivanjem stvara i prikazuje konačni rezultat.

6 ZAKLJUČAK

Python i web2py zajedno u kombinaciji nude brojne mogućnosti obrade teksta i izvlačenja informacija iz teksta uz lijep ispis na web-u. Zbog toga rad može biti dostupan svima za korištenje bez da se mora imati aplikacija, već se samo posjeti web stranica.

U ovom radu osmišljen je jedan takav sustav koji nam omogućuje da korisnik unese tekst te mu program za uneseni tekst prikazuje koji su tehnički pojmovi u tekstu sa njihovim opisom. Omogućava interakciju korisnika i rezultata, tj. omogućuje da se pojmovi naknadno uređuju ili da se dodaju u bazu ako ne postoje.

Kao daljnja mogućnost razvoja programa mogla bi se baza proširiti ne samo pojmovima iz tehničke znanosti već na sva područja koja su dana u poglavlju 4.4.1. Također trebala bi se korisniku dati mogućnost da ima svoju bazu za pretraživanje teksta koju može puniti po svojoj volji sa svojim pojmovima.

LITERATURA

[1] http://en.wikipedia.org/wiki/World_Wide_Web

[2] https://www.python.org/

[3] http://www.web2py.com/

[4] http://www.nltk.org/

[5] http://jquery.com/

[6] http://getbootstrap.com/2.3.2/javascript.html

[7] http://struna.ihjj.hr/

[8] http://narodne-novine.nn.hr/clanci/sluzbeni/2009_09_118_2929.html

PRILOZI

I. CD-R disc