UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Gašper Hafner

Podobnostne mreže receptov in spletna aplikacija za priporočanje sestavin jedi

DIPLOMSKO DELO
NA VISOKOŠOLSKEM ŠTUDIJU

Mentor: prof. dr. Blaž Zupan

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00015/2010
Datum: 01.10.2010

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

Kandidat: **GAŠPER HAFNER**

Naslov: **PODOBNOSTNE MREŽE RECEPTOV IN SPLETNA APLIKACIJA ZA PRIPOROČANJE SESTAVIN JEDI**

RECIPE NETWORKS AND WEB APPLICATION FOR RECOMMENDATION OF INGREDIENTS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V diplomskem delu uporabite spletne vire in iz njih pridobite podatke o receptih in sestavinah jedi. Iz podatkov razvijte mrežo, ki povezuje med seboj sorodne recepte. Na ta način pridobljene podatke uporabite v spletni aplikaciji, ki jo razvijte z namenom predlaganja dodatnih sestavin in receptov. Aplikacija naj svoje predloge oblikuje na podlagi krajšega seznama sestavin, ki jih uporabnik želi uporabiti in naj ta seznam dopolni skladno s sestavinami v primernih receptih, ki jih poišče v bazi.

Mentor:

prof. dr. Blaž Zupan

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a Gašper Hafner,

z vpisno številko 63060085,

sem avtor/-ica diplomskega dela z naslovom:

Podobnostne mreže receptov in spletna aplikacija za priporočanje sestavin jedi

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom
  prof. dr. Blaža Zupana
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključnene besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 17.03.2011 Podpis avtorja/-ice:

Zahvala

Zahvaljujem se mentorju prof. dr. Blažu Zupanu za tako dobro temo diplomske naloge in za vse nasvete, ki mi jih je nudil tekom pisanja diplomske naloge. Zahvaljujem se tudi asistentu Juretu Žbontarju za nasvete pri samem razvoju diplomske naloge.

Iskreno se zahvaljujem svoji mami, ker me je skozi celoten študij brezpogojno podpirala. Zahvaljujem se tudi punci Tatjani, ki je včasih potrpela mojo odsotnost zaradi pisanja diplomske naloge.

Kazalo

Povzetek

Cilj diplomskega dela je bila gradnja podatkovnih mrež različnih jedi in vizualizacija le teh. S pomočjo orodji za grafično vizualizacijo smo razvili algoritme, ki pametno povežejo različne jedi, ki so si med seboj najbolj podobne. Podatke o jedeh, njihovih sestavinah in receptih smo dobili s spletne strani (http://www.mojirecepti.com) s spletnim razčlenjevalnikom Beautiful Soup. V okviru diplomske naloge smo razvili spletno aplikacijo dostopno na spletni domeni (http://www.mizicapogrnise.si). Razvita spletna aplikacija, je kuharski asistent, ki uporabniku ob izbiri njegovih kuharskih sestavin predlaga kuharski recept, ki je najbolj primeren, ter tabelo manjkajočih sestavin. Spletna aplikacija je bila že v času razvoja diplomske naloge dobro obiskana.

Ključne besede:

podatkovne mreže jedi, vizualizacija, spletna aplikacija, kuharski recepti

Abstract

The aim of this thesis was to develop a network of food recipes. With the help of the tools for graphical visualisation we have developed algorithms that systematically link the dishes that are most alike together. The receipts were gathered from the web page (http://www.mojirecepti.com), parsed with Beutiful Soup parser. The collected recipe data was stored in the local data base. The main objective of the dissertation was to develop an internet application (http://www.mizicapogrnise.si) which suggests the most appropriate recipe for the ingredients that the user has selected and adds the table of the missing ingredients. At the time of this writing, the internet application was well visited.

Key words:

data networks of various dishes, visualization, internet application, cooking recipes

Poglavje 1

Uvod

Številne spletne strani (http://www.mojirecepti.com, http://www.kulinarika-.net, http://www.myrecipes.com) podajajo recepte za mnoge različne jedi. Na teh spletnih straneh lahko uporabnik vpiše naziv poznane jedi, aplikacija pa mu ponudi recept. Za uporabnika, ki točno ve, kaj si želi pripraviti za kosilo, ne ve pa kako, je kuharski recept dovolj dobra informacija, da bo željeno jed uspešno pripravil, seveda ob predpostavki, da uporabnik zna kuhati.

Ko se doma sami lotimo priprave jedi, je vedno prvi problem oziroma prvo vprašanje, kaj sploh želimo pripraviti. Zaradi pomanjkanja domišljije, časa, sestavin največkrat ne znamo odgovoriti na to vprašanje. Če pa smo kuharji s kuharsko kilometrino, največkrat pripravljamo tiste jedi, ki jih poznamo.

Naša želja je bila razviti spletno aplikacijo, ki bi učinkovito odgovorila na to vsakodnevno vprašanje. Naša spletna aplikacija dostopna na domeni (http://www.mizicapogrnise.si) uporabniku predlaga, kaj si lahko pripravi iz izbranih sestavin, ki jih ima. Uporabnika seznani, katerih sestavin še nima in mu ponudi kuharske recepte, ki so najbolj primerni.

V diplomski nalogi smo programsko zbrali podatke o receptih, ki so na spletni strani (http://www.mojirecepti.com). Nato smo zgradili mrežo podobnosti jedi. Podobnost med dvema jedema smo izračunali s pomočjo Jaccardovega podobnostnega koeficienta, ki je količnik med presekom skupnih sestavin in unijo vseh sestavin.

S pomočjo programskih knjižnic NetworkX (http://networkx.lanl.gov) in Matplotlib (http://matplotlib.sourceforge.net) in s programom za vizualizacijo Graphviz (http://www.graphviz.org) smo postopoma vizualizirali podatkovne mreže jedi, iz katerih lahko razberemo, katere jedi so si po sestavinah najbolj podobne.

Spletno aplikacijo smo razvili s pomočjo skriptnega programskega jezika

Python, ki ima vgrajene podatkovne strukture, ki smo jih tekom razvoja s pridom izkoristili. Za pisanje kode smo izbrali urejevalnik kode Pyscripter IDE, predvsem zato, ker je odprtokoden, pregleden in je namenjen razvoju manjših projektov. Spletno aplikacijo smo opremili s skritpnim programskim jezikom Javascript, ki skrbi za dinamično izvajanje spletnih aplikacij. Pri razvoju smo uporablili jQuery, ki je knjižnica programskega jezika Javascript. Knjižnica jQuery poskrbi, da se spletna aplikacija izvaja neodvisno od uporabnikovega brskalnika. Podatki o sestavinah in receptih, ki jih potrebuje aplikacija, hrani podatkovna baza SQLite. Spletna aplikacija teče na Apache strežniku in je dostopna na spletni domeni (http://www.mizicapogrnise.si).

V poglavjih, ki sledijo so podrobnjeje opisana programska orodja in tehnike, ki smo jih uporabili tekom razvoja diplomske naloge.

V drugem poglavju smo predstavili programski jezik Python in njegove uporabne knjižnice za gradnjo in vizualizacijo mrež NetworkX in Matplotlib. V tem poglavju smo predstavili tudi programsko orodje za vizualizacijo podatkovnih mrež Graphviz. Podatke, ki smo jih uporabili pri gradnji in vizualizaciji mrež z zgoraj naštetimi orodji smo dobili s spletnim razčlenjevalnikom Beautiful Soup, katerega delovanje in uporabnost podrobneje opisujemo v tem poglavju.

V tretjem poglavju smo se posvetili gradnji in vizualizaciji mrež jedi s programskim knjižnicama NetworkX in Matplotlib ter s programom Graphviz. Opisali smo postopke, kako smo dobili operativne podatke o receptih jedi s pomočjo spletnega razčlenejevalnika Beautiful Soup. Predstavili smo Jaccardov podobnostni koeficient in njegovo uporabnost pri vizualizaciji mrež jedi.

V predzadnjem poglavju je predstavljena spletna aplikacija. Opisani so koraki razvoja aplikacije, uporabljene programske tehnike in orodja, ki smo jih uporabili pri razvoju aplikacije. Opisali smo primere uporabe spletne aplikacije.

V zadnjem petem poglavju smo ovrednotili vse zastavljene zahteve in možnosti za nadaljni razvoj.

Poglavje 2

Uporabljene tehnologije in programska orodja

Pri razvoju programske opreme, ki je nastala v okvirih diplomske naloge, smo uporabili programski jezik Python (http://www.python.org) ter nekatere njegove knjižnice za ustvarjanje, manipulacijo in vizualizacijo mrež NetworkX (http://networkx.lanl.gov) in Matplotlib (http://matplotlib.sourceforge.net). Pri samem razvoju smo uporabili programsko orodje za vizualizacijo mrež Graphviz (http://www.graphviz.org). Podatke, ki smo jih uporabili za gradnjo in vizualizacijo mrež smo dobili s spletne strani s pomočjo spletnega razčlenjevalnika Beautiful Soup (http://www.crummy.com/software/BeautifulSoup).

2.1 Programski jezik Python

Python je prenosljiv, interpretiran, objektno usmerjen programski jezik. Njegov razvoj se je začel leta 1990 v CWI v Amsterdamu na Nizozemskem in se nadaljuje pod lastništvom fundacije Python Software Foundation.

Najnovejše informacije o jeziku dobimo na njegovi uradni domači strani (http://www.python.org). Tam tudi dobimo najnovejšo verzijo tolmača z vsemi možnimi dodatki, ki ga preprosto namestimo na svoj računalnik.

Prednosti programskega jezika Python so [1]:

- Ima zelo jasno in razumljivo sintakso,
- ima močno introspekcijsko zmogljivost,

- je objektno usmerjen programski jezik,
- vsebuje visokonivojske podatkovne strukture.

Za Python obstaja več zanimivih in uporabnih knjižnic, ki mu povečajo zmogljivost in ga povežejo z drugimi uporabnimi produkti.

2.2 Uporabljene Python knjižnice

2.2.1 NetworkX

NetworkX (http://networkx.lanl.gov) je programska knjižnica v programskem jeziku Python. Uporablja se za ustvarjanje, manipulacijo in učenje kompleksnih omrežji, grafov. NetworkX je orodje za preučevanje strukture in dinamike socialnih, bioloških in infrastrukturnih omrežji. Omogočanje udobje pri uporabi in hitrem razvoju v razvojnih multidisciplinarnih skupinah. Je odprtokodna programska oprema, ki zagotavlja funkcionalnosti za raznolike skupnosti aktivnih, sodelujočih uporabnikov in razvijalcev. NetworkX je enostaven vmesnik z obstoječo kodo, napisano v programskih jezikih C, C++ in FORTRAN.

Po definiciji je graf zbirka vozlišč (točk) skupaj z določenimi pari vozlišč (robovi, povezave). V grafu, ki je objekt razreda NetworkX so lahko vozlišča vsi objekti, ki tekom svojega življena nikoli ne spremenijo svoje vrednosti in ki jih lahko primerjamo z ostalimi objekti istega razreda. To so tako imenovani hashable objekti. Primeri takih objektov so znakovni nizi, števila, slike, objekti XML, grafi - objekt razreda NetworkX, vozišča, ...

Prazen graf brez vozlišč in povezav ustvarimo s spodnjim ukazom:

```
>>> import networkx as nx
>>> G = nx.Graph()
```

Graf G je objekt razreda NetworkX in ob inicializaciji ne vsebuje nobene povezave ali vozlišča. V graf G lahko na več načinov dodamo vozlišča in povezave. Knjižnica NetworkX vsebuje številne funkcije za generiranje grafov in možnosti branja in pisanja grafov v različnih formatih. V graf G lahko s spodnjim ukazom naenkrat dodamo eno vozlišče:

```
>>> G.add_node('Grahova_juha')
```

V graf G lahko dodamo vozlišča iz seznama:

```
>>> G.add_nodes_from(['Gobova_juha', 'Bucna_juha'])
```

V zgornjem primeru smo v graf G dodali vsa vozlišča s seznama. Grafu G lahko kot vozlišče dodamo celoten graf H s spodnjim ukazom:

```
>>> H = nx.Graph()
>>> H.add_nodes_from(['Zelenjavna_juha','Bucna_juha'])
>>> G = nx.add_node(H)
```

Zgornja funkcionalost je zelo uporabna, saj omogoča ustvarjanje grafa grafov, grafa datotek, grafa funkcij, ... Ta funkcionalost je v analogiji z razvojem večje aplikacije, kjer je aplikacija graf, entitete pa so vozlišča v tem grafu - aplikaciji.

V graf lahko dodajamo povezave med vozlišči s spodnjim ukazom:

```
>>> G = nx.add_edge('Bucna_juha','Zelenjavna_juha')
```

V graf G lahko dodamo seznam parov povezav med vozlišči:

```
>>> G = nx.add_edge_from([('Bucna_juha','Zelenjavna_juha'),
('Grahova_juha','Bucna_juha')])
```

V graf G lahko dodamo tudi kup povezav imenovano ebunch. Ebunch je vsaka števna množica parov povezav povezava - terka. Par povezava - terka je lahko terka velikosti 2 v kateri sta 2 med seboj povezani vozlišči, lahko pa je terka velikosti 3, v kateri sta 2 med seboj povezani vozlišči in atribut povezave.

Graf G lahko na podoben način tudi porušimo z uporabo funkcij $Graph.remove\_node()$, $Graph.remove\_nodes\_from()$, $Graph.remove\_edge()$ in $Graph.remove\_edges\_from()$.

Vsa vozlišča in povezave z grafa G lahko pobrišemo z ukazom *clear*().

```
>>> G.clear()
```

V vsakem trenutku lahko preverimo koliko vozlišč je v grafu G,

```
>>> G.number_of_nodes()
```

in koliko je v grafu G povezav.

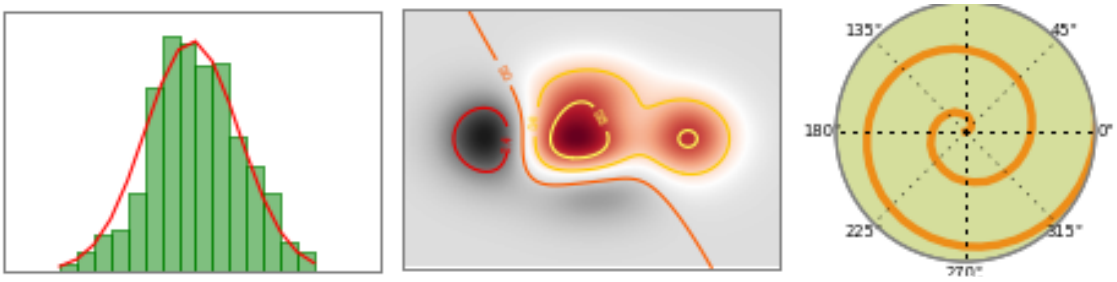
```
>>> G.number_of_edges()
```

Graf G lahko tudi preučujemo. Če nas zanima katera vozišča so v povezavi s točno določenim vozliščem slednje preverimo s spodnjim ukazom.

```
>>> G.neighbors('Bucna_juha')
['Zelenjavna_juha', 'Grahova_juha']
```

2.2.2 Matplotlib

Matplotlib (http://matplotlib.sourceforge.net) je knjižnica za risanje grafov (Slika 2.1) v okviru Python skriptnega jezika in je trenutno najbolj popoln prikazovalnik podatkov. Skupaj s samim Python jezikom tvori zelo močno orodje za obdelavo podatkov. Slike grafov lahko izdelamo z interpretativnim pisanjem programa v Python okolju ali poganjanjem skripte v terminalu.



Slika 2.1: Primer grafov, narisanih s pomočjo matplotlib knjižnice.

2.3 Spletni razčlenjevalnik Beautiful Soup

Beautiful Soup (http://www.crummy.com/software/BeautifulSoup) je razčlenjevalnik HTML/XML dokumentov v programskem jeziku Python.

Beautiful Soup razčlenjevalnik se ne zaduši s podatki, če smo slabo podali zahteve za razčlenitev. Vrne nam drevesno strukturo razčlenjenih podatkov, ki spominjajo na prvotni dokument. Ponuja nekaj preprostih načinov in Pythonovih slogov za krmiljenje, iskanje in prilagajanje razčlenjenega drevesa. Služi kot orodje za razčlenjevanje dokumentov in pridobivanje tistega, kar potrebujemo. Samodejno pretvarja vhodne dokumente v Unicode in izhodne dokumente v formatu UTF-8.

Beautiful Soap razčleni dokument HTML/XML in vrne drevesno strukturo. Kot argumente lahko navedemo "Poišči vse spletne povezave", "Poišči vse spletne povezave razreda externalLink", "Poišči vse spletne povezave, katerih naslovi se ujemajo s foo.com"ali "Poišči naslove tabel, ki vsebuje krepko besedilo, nato pa vrni to besedilo".

Z uporabo Beautiful Soup razčlenjevalnika programer hitro pride do vsebin iz drevesne strukture HTML/XML, ki ga zanimajo.

```
<div class="intro">
  <img src="/slike/kislo-zelje-s-krompirjem.jpg">
  <h1>Kislo zelje s krompirjem</h1>
  <p>Recept za kislo zelje s krompirjem, korenjem.</p>
  <p>Skupina: <strong>Glavne jedi</strong></p>
  <p>Količine za 4 osebe</p>
</div>
```

Iz zgornje kode, napisane v jeziku HTML, smo s pomočjo Beautiful Soup razčlenjevalnika dobili ime jedi s spodnjim ukazom:

```
>>> from BeautifulSoup import BeautifulSoup
>>> imeJedi = BeautifulSoup(dokument_HTML)
>>> print str(imeJedi.find('h1')).replace('<h1>','').replace('</h1
    >','')
'Kislo zelje s krompirjem'
```

2.4 Program za vizualizacijo Graphviz

Graphviz (http://www.graphviz.org) je program za grafično predstavitev informacij s pomočjo grafov in podatkovnih mrež. Na trgu programskega inženiringa, razvoja podatkovnih baz in spletnega dizajna obstaja veliko drugih programov, ki vsebujejo avtomatičen izris grafov. Graphviz je odprtokodno grafično orodje. Vsebuje veliko podprogramov za izris različnih grafov.

Sestoji iz petih programov[3]:

- Dot - program za izris hierarhično usmerjenih grafov.
- Neato - program za izris splošnih neusmerjenih grafih.
- Twopi - program za izris grafov z radialno inženiringa vozlišč.
- Circo - program za izris grafov z krožno inženiringa vozlišč.
- Fdp - program za izris splošnih neusmerjenih grafih.

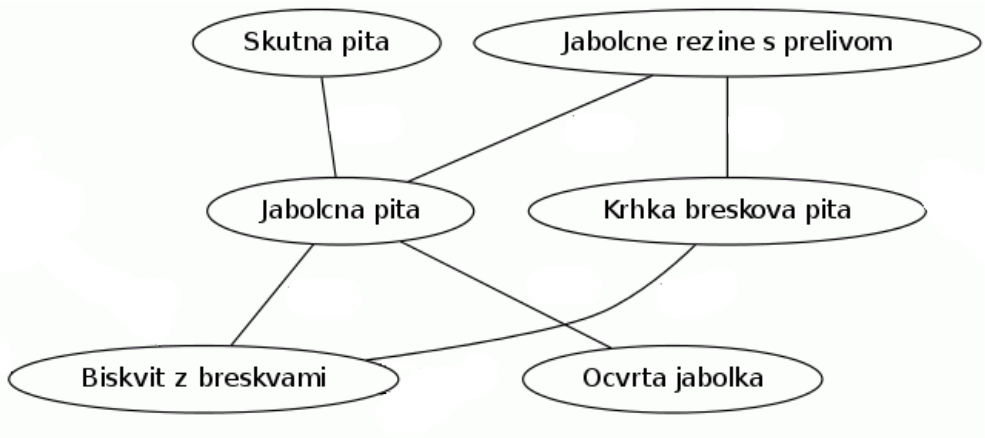
Program Graphviz iz tekstovne datoteke dobi podroben opis grafa, sam pa poskrbi za pravilen izris vozlišč in povezav glede na podan tip grafa. Končni rezultati obdelave so lahko v različnih formatih (GIF, PS, SVG, GXL, XML). Primer opisa grafa in vizualizacija tega grafa je na sliki 2.2.

```
graph Podobne_Jedi{
  Skutna pita -- Jabolcna pita;
  Biskvit z breskvami -- Jabolcna pita;
  Biskvit z breskvami -- Krhka breskova pita;
  Krhka breskova pita -- Jabolcne rezine s prelivom;
  Jabolcne rezine s prelivom -- Jabolcna pita;
  Jabolcna pita -- Ocvrta jabolka;
}
```

Z zgornjim opisom grafa, ki smo ga podali programu Graphviz kot argument smo dobili spodnji izris grafa jedi (Slika 2.2)

Program Graphviz ima številne uporabne funkcije za konkretne načrte, kot so možnosti barve, pisave, tabele postavitve vozlišč, slogi črt, hiperpovezave in oblike po meri.

V praksi grafi običajno nastanejo iz zunanjih virov podatkov, lahko pa se ustvarjajo in urejajo tudi ročno, kot surovine besedilne datoteke ali v grafičnem urejevalniku.

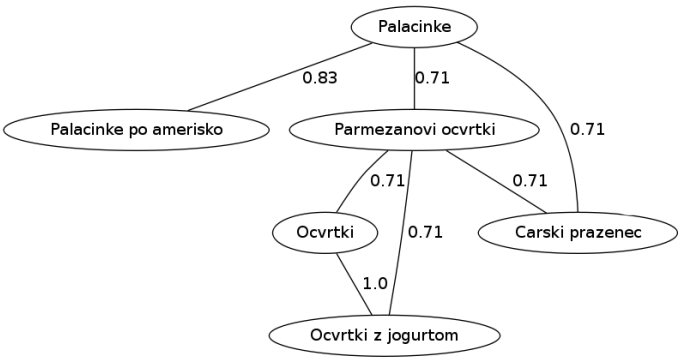


Slika 2.2: Primer grafa narejenega v programu Graphviz iz predhodnega opisa.

Poglavje 3

Mreža jedi in njena vizualizacija

Uporabnika smo želeli seznaniti z informacijo, katere jedi so si po sestavinah najbolj podobne. Odločili smo se, da bomo dobljene podatke predstavili s pomočjo podatkovne mreže. Mreže so zelo primeren model za predstavitve znanja, saj so večini ljudi razumljive, prav tako pa se lahko uporabljajo v računalniških sistemih in pri razvijanju umetne inteligence. Znanje je predstavljeno z medsebojno povezanimi vozlišči, ki označujejo koncepte. Povezave med vozlišči pa označujejo relacije med koncepti. V našem primeru so vozlišča v mrežnem grafu jedi, povezave med njimi pa vsebujejo informacijo o podobnosti med njimi (Slika 3.1).

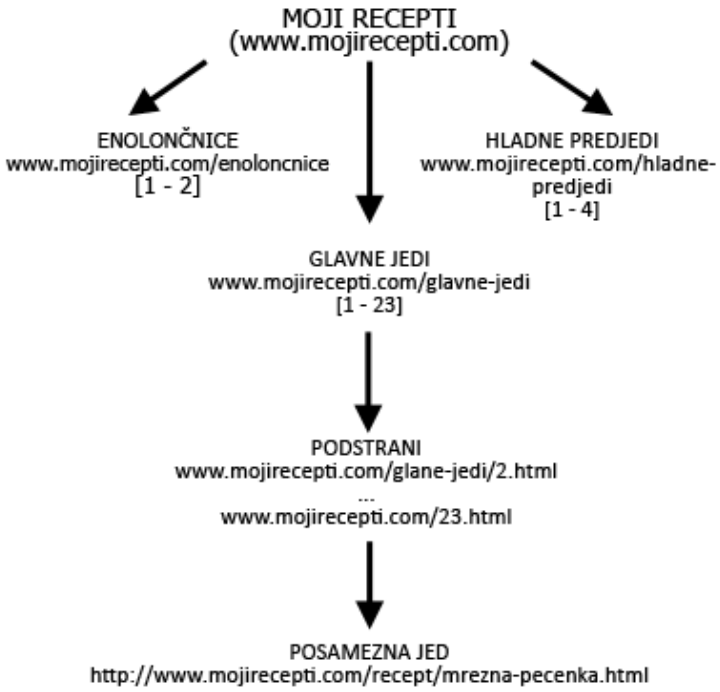


Slika 3.1: Primer podatkovne mreže jedi. Vozlišča v mreži so jedi, povezave med njimi pa označujejo podobnost med jedmi.

3.1 Podatki o receptih in sestavinah

Za gradnjo podatkovnih mrež so potrebna vozlišča in relacije med njimi. V našem primeru so bila vozlišča jedi, relacije med njimi pa odstotki podobnosti med sestavinami le-teh. Podatke o jedeh in njihovih sestavinah smo dobili na spletni strani (http://www.mojirecepti.com), ki vsebuje več kot 1800 različnih receptov.

Na spletni strani so jedi ločene v različne kategorije (enolončnice, glavne jedi, hladne predjedi, juhe, ...). V vsaki kategoriji je več različnih jedi, vsaka jed pa ima opis priprave jedi, seznam sestavin in seznam pripomočkov (Slika 3.2).



Slika 3.2: Struktura spletnih naslovov na spletni strani (http://www.mojirecepti.com).

Programski jezik Python vsebuje modul *urllib*. Modul *urllib* je programski vmesnik za pobiranje podatkov s svetovnega spleta. Z uporabo njegove funkcije *urlopen*, ki kot argument sprejme naslov spletne strani, smo dobil vsebino spletne strani (strukturo HTML) za posamezno jed. Iz dobljene drevesne strukture dokumenta HTML (Slika 3.3) smo z algoritmom pod sliko 3.3

dobili naslov jedi.

```
<div class="intro">
        <img src="/slike/mesna-lazanja.jpg" alt="Mesna lazanja" class="img" width="270" height="200">
        <h1>Mesna lazanja</h1>
        <p>Recept za lazanjo iz mletega mesa z bešamel omako.</p>
        <p>Skupina: <strong>Glavne jedi</strong></p>
        <p>Količine za 4 osebe</p>
</div>
```

Slika 3.3: Del kode HTML, kjer se nahaja naslov jedi, ki ga želimo izluščiti.

```
>>> from BeautifulSoup import BeautifulSoup
>>> imeJedi = BeautifulSoup(dokument_HTML)
>>> print str(imeJedi.find('h1')).replace('<h1>','').replace('</h1
    >','')

    'Mesna lazanja'
```

Vse sestavine jedi so se v HTML dokumentu nahajale v tabeli (Slika 3.4). Z algotimom pod sliko 3.4, smo dobili vse sestavine in jih shranili v tekstovno datoteko skupaj z imenom jedi.

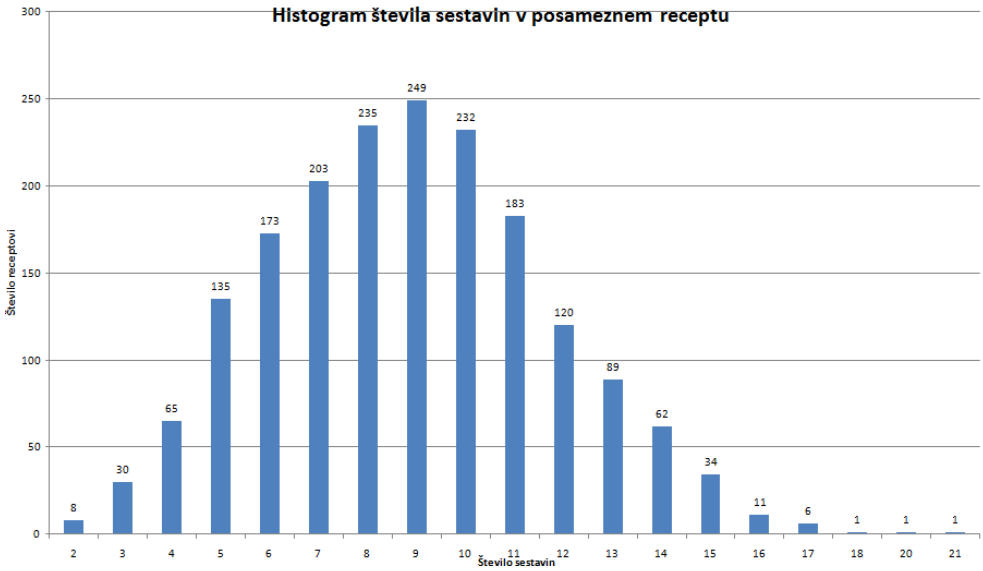
```
<div class="title">Sestavine</div>
<div class="hilite text">
        <table>
                <tbody>
                        <tr class="first">
                                <th>Cebula</th>
                                <td>1 kos</td>
                        </tr>
                        <tr>
                                <th>Olje - soncnicno</th>
                                <td> </td>
                        </tr>
```

Slika 3.4: Del kode HTML, kjer se nahaja tabela sestavin, ki jih želimo izluščiti.

```
>>> from BeautifulSoup import BeautifulSoup
>>> objektBeautifulSoup = BeautifulSoup(HTML_Dokument)
>>> tabela_sestavin = htmlContent.findAll('div', {'class': 'hilite
    text'}
>>> objektBeautifulSoup = BeautifulSoup(tabela_sestavin)
>>> sestavine = "".join([str(x) for x in objektBeautifulSoup.
    findAll('th')]).replace('</th><th>','|').replace('</th>','').
    replace('<th>','')
```

```
'Cebula|Olje - soncnicno|Mleto meso|Korenje|Sol|Poper|Muskatni
    orescek|Lovorov list|Bazilika|Paradiznik - svez|Paradiznik -
    mezga|Maslo|Psenicna bela moka|Kravje mleko - manj mastno|Listi
     za lazanjo|Sir - edamec'
```

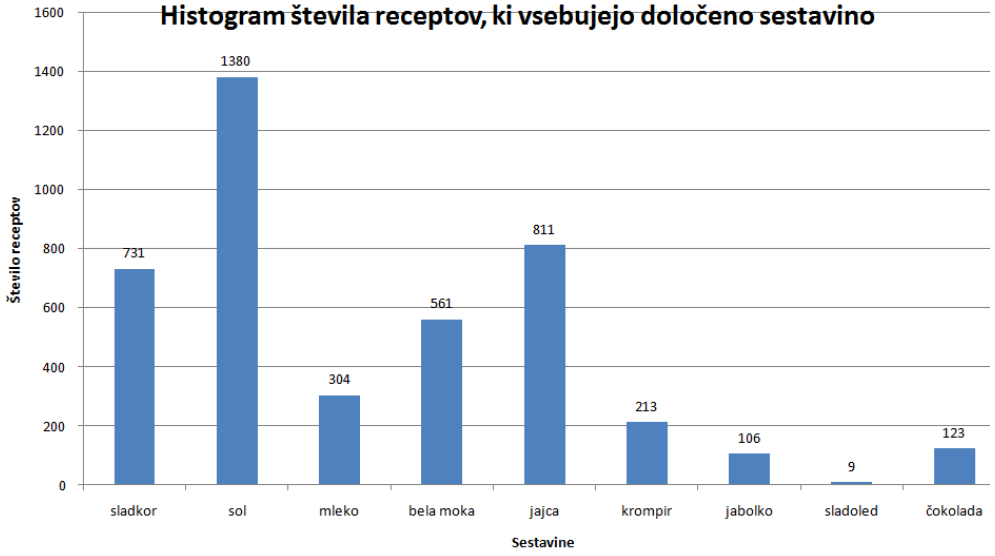
S spletne strani (http://www.mojirecepti.com) smo dobili 1838 različnih receptov. Vse recepte gradi 421 različnih sestavin. Spodnji graf (Slika 3.5) prikazuje histogram števila sestavin v posameznem receptu.



Slika 3.5: Histogram števila sestavin v posameznem receptu.

Z zgornjega histograma (Slika 3.5) lahko razberemo, da je največ jedi sestavljenih iz 9 različnih sestavin. Vseh skupaj je 249. Med vsemi recepti je tudi recept, ki vsebuje kar 21 različnih sestavin.

Naredili smo tudi graf receptov, ki vsebujejo določeno sestavino (Slika 3.6). Iz vseh sestavin smo izbrali tiste, ki se pri pripravi jedi največkrat uporabljaj (sol, sladkor, moka, jajca) in nekaj poljubnih sestavin, ki so nas zanimali.



Slika 3.6: Graf, ki prikazuje število receptov, ki vsebujejo določeno sestavino.

V receptih se največkrat pojavi sol, ki jo vsebuje 1380 različnih receptov.

3.2 Vizualizacija mreže jedi

Preden smo se lotili vizualizacije podatkovne mreže jedi, smo definirali, kaj bodo naša vozlišča in kaj bodo relacije med temi vozlišči. Za vozlišča smo določili imena jedi, relacije pa so bili odstotki podobnosti med temi jedmi. Za izračun podobnosti med jedmi smo uporabili Jaccardov podobnostni koeficient.

3.2.1 Jaccardov podobnostni koeficient

Jaccardov podobnostni koeficient (3.1), je statistični podatek, ki ga uporabljamo za računanje podobnosti med dvema množicama.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

V našem primeru smo podobnost med dvema receptoma izračunali tako, da smo število skupnih sestavin, ki so prisotne v obeh jedeh, delili s številom

vseh različnih sestavin, ki so v obeh jedeh. Pri razčlenjevanju spletne strani smo dobili 1838 različnih jedi. Po izračunu vseh odstotkov podobnosti smo dobili matriko veliko 1838 x 1837. Podatkovna struktura, kamor smo shranili matriko je bila slovar. V slovarju so bili ključi indeksi jedi, vrednosti pa so bil slovarji v katerih so bili indeksi jedi, ki jih primerjamo z jedjo ki je ključ vrhnjega slovarja, vrednosti pa Jaccardov indeks oziroma odstotek podobnosti. Celoten slovar smo shranili v tekstovno datoteko.

```
{1: {2:0.11, 3:0.0, 4:0.08, 5:0.22, ...}}
{2: {1:0.11, 3:0.0, 4:0.2, 5:0.5, ...}}
{3: {1:0.0, 2:0.0, 4:0.0, 5:0.5, ...}}
{4: {1:0.08, 2:0.2, 3:0.0, 5:0.18, ...}}
{5: {1:0.22, 2:0.5, 3:0.13, 4:0.18, ...}}
```

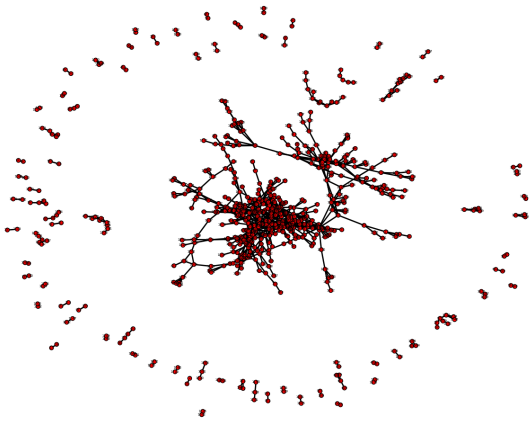
3.2.2 Vizualizacija podatkovne mreže jedi z uporabo knjižnic NetworkX in Matplotlib

Naloga algoritma povprečnih povezav je bila, da iz množice parov med dvema vozliščema izbere n parov, ki imajo podobnost večjo od določene vrednosti, praga. Algoritem se programsko sprehodi čez tekstovno datoteko, kjer imamo shranjeno matriko primerjav med vozlišči. Iz vsake vrstice prebere slovar in primerja podobosti s pragom, ki smo ga podali kot argument funkciji. Prag je v našem primeru podobnost med dvema vozliščema. Na kopico postavlja samo tiste pare vozlišč, ki presežejo prag podobnosti.

Algoritem za sortiranje s kopico je že vgrajen v programski jezik Python in se imenuje heapq. Z metodo heappush na kopico pravilno vstavljamo vrednosti, z metodo nlargest pa dobimo seznam dolžine n parov vozlič, katerih podobnost je največja.

Iz tako dobljenega seznama parov vozlišč, katerih podobnost je največja, smo s pomočjo knjižnic NetworkX in Matplotlib vizualizirali podatkovno mrežo jedi.

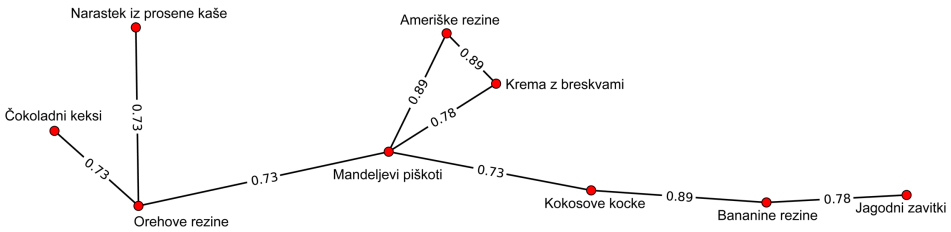
Najprej smo zgradili in vizualizirali podatkovne mreže jedi, pri kateri smo iz množice izbrali 1000 najboljših parov vozlišč, ki imajo odstotek podobnosti večji od 0 (Slika 3.7).



Slika 3.7: Podatkovna mreža jedi, sestavljena iz 1000 najboljših parov vozlišč in odstotkom podobnosti, večjim od 0.

Iz grafa, ki je s slike 3.7, smo ugotovili, da je zgrajena podatkovna mreža jedi nepregledna, neberljiva in da nam ni podala informacije o tem, katere jedi so si po sestavinah najbolj podobne. Da bi iz podatkovne mreže jedi to lahko razbrali, bi morali vozlišča poimenovati z imeni jedi, na povezave med vozlišči pa bi bilo potrebno vpisati odstotke podobnosti med vozlišči. Ker bi bil z dodatnimi atributi graf še bolj nepregleden smo se odločili, da bomo zmanjšali število najboljših parov vozlišč in dvignili prag podobnosti.

Na sliki (Slika 3.8) je izrez iz podatkovne mreže jedi s stotimi najboljšimi pari vozlišč in odstotkom podobnosti večjim od 0,7.



Slika 3.8: Izrez iz podatkovne mreže jedi sestavljen iz sedemdesetih najboljših parov vozlišč in odstotkom podobnosti večjim od 0,7.

3.2.3 Vizualizacija podatkovne mreže jedi z uporabo programa Graphviz

Graphviz postavi mrežo iz opisa grafa, ki mu ga podamo v tekstovni datoteki. Programska knjižnica NetworkX vsebuje metodo *writedot*, ki generira opis grafa in ta opis zapiše v tekstovno datoteko.

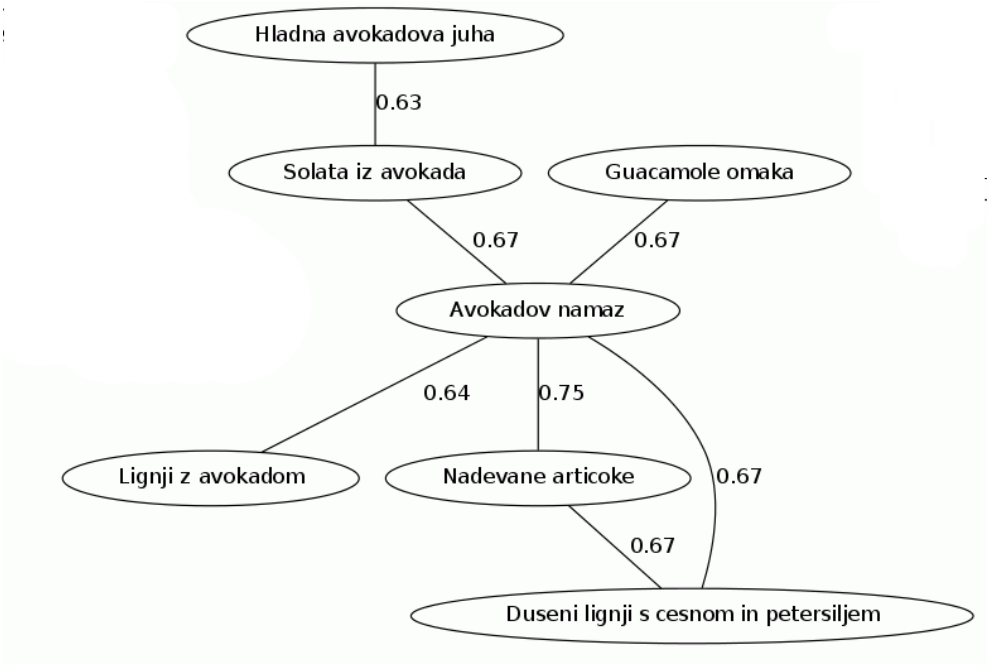
```
graph {
   "Hladna avokadova juha" -- "Solata iz avokada" [label="0.63"];
   "Solata iz avokada" -- "Avokadov namaz" [label="0.67"];
   "Avokadov namaz" -- "Lignji z avokadom" [label="0.83"];
   "Avokadov namaz" -- "Nadevane artičoke" [label="0.83"];
   "Avokadov namaz" -- "Guacamole omaka" [label="0.83"];
   "Avokadov namaz" -- "Duseni lignji s cesnom in petrsiljem"
   [label="0.83"];
   "Nadevane artičoke" -- "Duseni lignji s cesnom in petrsiljem"
   [label="0.83"];
}
```

V vsaki vrstici tekstovne datoteke se nahaja par vozlišč v grafu in njuna podobnost. Program Graphviz poženemo iz ukazne vrstice z ukazom:

```
dot -Tpng opisGrafa.dot -o graf.png
```

Dot je podprogram programa Graphviz, ki se ga v splošnem uporabljamo za vizualizacijo usmerjenih grafov v hierarhično drevo. Prednosti, ki jih ta program ponuja, so njegovi zmogljivi algoritmi, ki optimalno razporedijo vozlišča in povezave, da se le-te ne prekrivajo, križajo med sabo. Ta lastnost je bila pri našem delu zelo pomembna, saj smo na koncu dobili nekaj zelo preglednih grafov. Programu dot smo podali argument $-Tpng$, ki programu pove, naj bo končni rezultat slika v formatu PNG. Naslednji argument $opisGrafa.dot$ je tekstovna datoteka v kateri hranimo opis grafa, ki je programu dot razumljiv. Z zadnjim argumentom $-ograf.png$ programu povemo, da naj bo končni rezultat slika formata PNG, z nazivom graf.png.

Slika (Slika 3.9) prikazuje graf podatkovne mreže jedi, ustvarjena s programom Graphviz. Prednosti uporabe programa Graphviz pred uporabo Python knjižnic NetworkX in Matplotlib so lepo vidne. Graf je pregleden in berljiv. Z dane slike je mogoče dobro razbrati, katere jedi (vozlišča) so si med seboj najbolj podobne.



Slika 3.9: Izrez semantične mreže sestavljena iz sedemdesetih najboljših parov vozlišč in odstotkom podobnosti večjim od 0,7 nastala z uporabo programa Graphviz.

Poglavje 4

Razvoj spletne aplikacije

Spletna aplikacija dostopna na spletni domeni (http://www.mizicapogrnise.si) je namenjena vsem, ki imajo željo po kuhanju. Glavni namen aplikacije je, da uporabniku predlaga seznam jedi, ki si jih lahko pripravi. V tem seznamu so samo tiste jedi, ki vsebujejo sestavine, ki so bile predhodno izbrane v aplikaciji. Ob vsaki izbiri sestavin se na strežnik pošljejo podatki o izbranih sestavinah. Na strežniku se ustvari seznam najprimernejših jedi, seznam manjkajočih sestavin in recepti za vsako jed posebej. Prijazen in pregleden uporabniški vmesnik uporabniku omogoča, da se uporabnik ne izgubi v množici odvečnih podatkov.

V naslednjih poglavjih bomo natančno predstavili razvito aplikacijo. Spoznali bomo njen glavni namen, funkcionalnost, uporabniški vmesnik ter možnosti za nadaljnji razvoj in uporabo v realnem okolju.

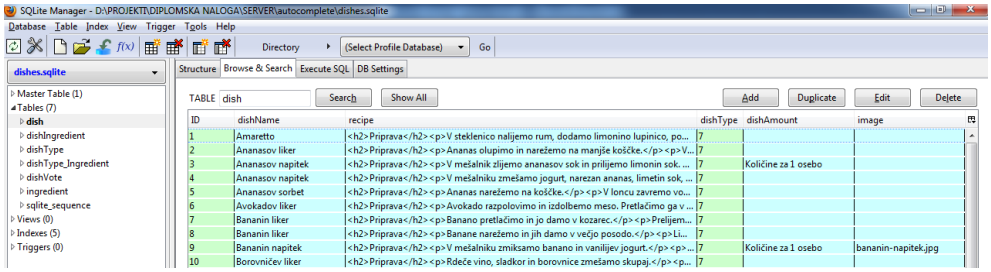
4.1 Orodja in tehnike

Pri razvoju spletne aplikacije smo uporabili več različnih orodji in tehnik. Spletno aplikacijo smo napisali v programskem jeziku Python. Za pisanje kode v Pythonu smo uporabili programsko okolje PyScripter, ki se uporablja pri razvoju manjših aplikacij napisanih v tem programskem jeziku. Ima vse lastnosti dobrega programskega okolja. Vsebuje pregledovalnik razredov, programerju ponuja pomoč pri pisanju kode in ima napreden razhroščevalnik.

Aplikacija teče na spletnem strežniku Apache. Poleg strežnika Apache smo namestili še modul mod_python. Modul mod_python [2] je dodatek za Apache strežnik, ki omogoča izvajanje Python skript znotraj Apache strežnika. Narejen je bil z namenom, da reši problem zunanjega izvajanja skript, napisanih v programskem jeziku Python. Ob vsaki zahtevi, ki prispe na strežnik, se ustvari

nov proces, v katerem se izvede Python skripta. Ta način delovanja je počasen, saj je kreiranje novega procesa počasno. Mod_python deluje znotraj Apache procesa, zaradi česar ni več potrebno kreirati zunanjega procesa.

Za bazo podatkov smo si izbrali SQLite bazo podatkov, kjer hranimo naše podatke o kuharskih receptih. SQLite je odprtokodna podatkovna baza napisana v programskem jeziku C. Za razliko od večine ostalih SQL podatkovnih baz je sistemsko neodvisna in za svoje delovanje ne potrebuje ločenega procesa na strežniku. Baza SQLite bere in zapisuje podatke v običajno datoteko na strežniku. Za izgradnjo podatovne baze smo uporabili programsko orodje SQLite Manager (Slika 4.1), ki je dodatek spletnemu brskalniku Mozilla Firefox. SQLite Manager ponuja pregleden grafični vmesnik ki nam omogoča da lahko hitro in enostavno izvedemo katerokoli operacijo nad podatki v bazi. Omogoča izvoz tabel v bazi v različnih formatih (CSV, XML, SQL, ...) v formatih UTF-8 ali UTF-16.

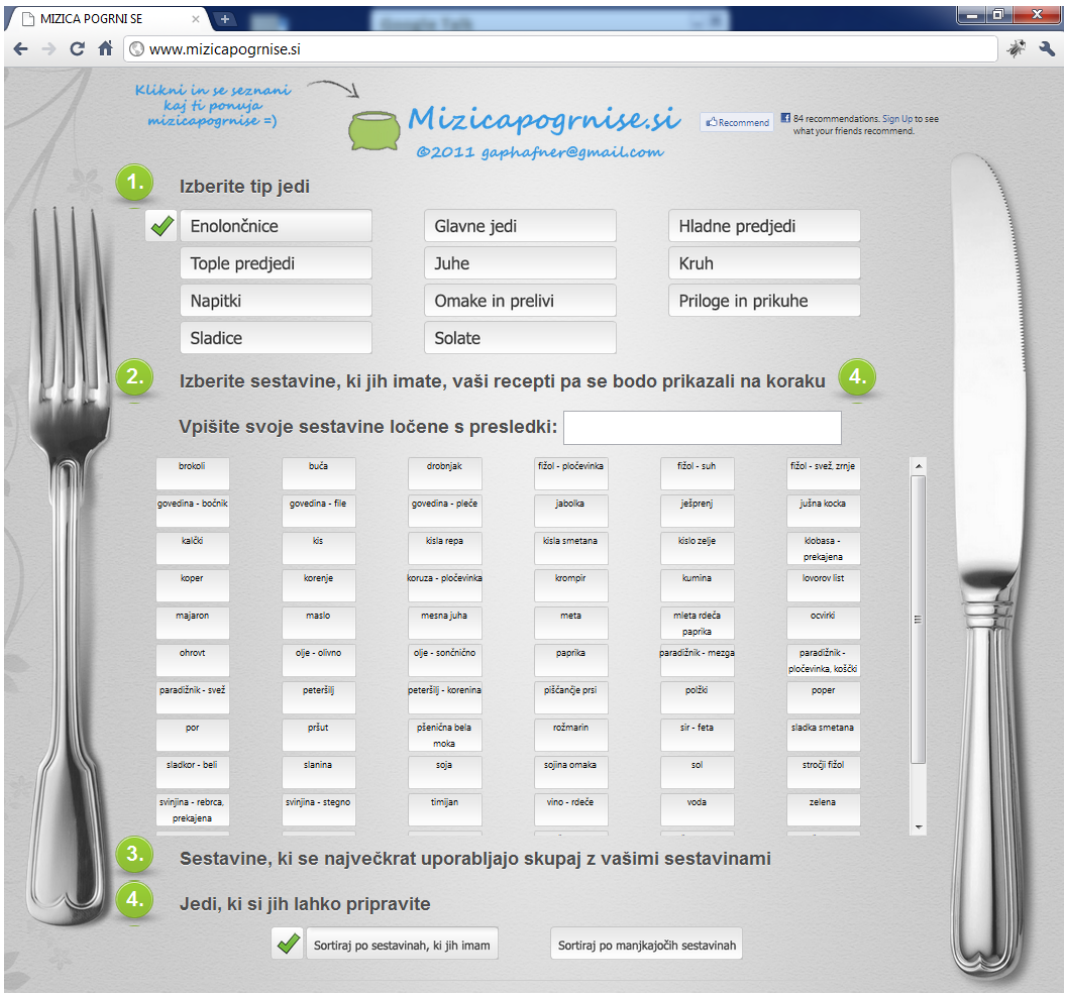


Slika 4.1: Na zgornji sliki je programsko orodje SQLite Manager, ki smo ga uporabili pri rokovanju s podatkovno bazo SQLite.

Aplikacija je podprta s programskim jezikom Javascript. Pri samem razvoju smo uporabli javascript programsko knjižnico jQuery. Knjižnica jQuery se izvaja na klientovi strani neodvisno od brskalnika, ki ga uporabnik uporablja. Omogoča lažje dinamično spreminjanje dokumentov HTML, opremljanje HTML značk z animacijami in dogodki ter lažje delo s tehnologijo AJAX.

AJAX (asinhroni JavaScript in XML) je skupina medsebojno povezanih spletnih razvojnih tehnik, uporabljenih za ustvarjanje interaktivnih spletnih aplikacij. Z Ajaxom si lahko spletne aplikacije izmenjujejo podatke s strežnikom asinhrono v ozadju, brez potrebe po ponovnem nalaganju strani. S tem je mogoče tekoče in hitrejše spremljanje ter spreminjanje vsebine na spletni strani.

Grafični vmesnik spletne aplikacije (Slika 4.2) smo grafično opremili s CSS datotekami. Za vizualno oblikovanje gradnikov smo uporabili program Adobe Photoshop.



Slika 4.2: Grafični vmesnik spletne aplikacije.

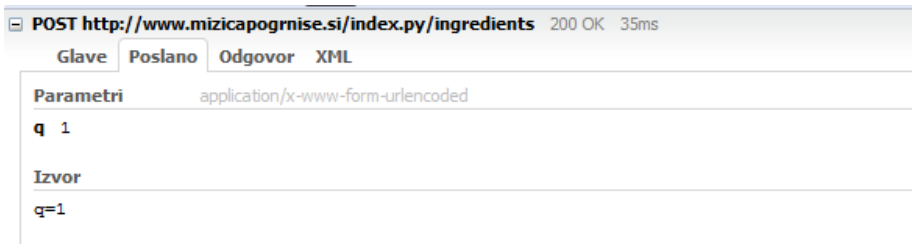
4.2 Primeri uporabe

Aplikacija uporabniku omogoča, da si slednji lahko izbere vrsto jedi, ki si jo želi pripraviti. Ob spremembi tipa jedi, se na strežnik s spletno tehnologijo AJAX asinhrono pošlje zahteva z informacijo o izbrani vrsti jedi. V prejšnjem poglavju smo omenili, da smo uporabili jQuery javascript knjižnico, ki vsebuje metode za delo s spletno tehnologijo AJAX. Knjižnica jQuery vsebuje metodo jQuery.post(), ki poskrbi da se asinhrono na strežnik pošlje POST zahteva. Spodaj napisana koda poskrbi, da se na streznik pošlje POST zahteva:

```
var dishType =  $('#list  a.active').attr('id');

$.post('index.py/ingredients', { q : dishType},
        function(data) {
            $('.sestavine').html(data)
        },
        'text');
```

V zgornji kodi se ob kliku na gumb, ki označuje vrsto jedi, v spremenljivko *dishType* shrani identifikator tipa jedi. Nato se z uporabo metode *.post*() pošlje asinhrona zahteva na strežnik (Slika 4.3).



Slika 4.3: Slika prikazuje asinhrono zahtevo POST na strežnik, ob kliku na vrsto jedi.

Metoda $ingredients(req, q)$ sprejme dva argumenta. S prvim req argumentom metodi povemo, kakšen tip vsebine naj metoda vrača klientu, drugi q pa je identifikator vrste jedi. Metoda $ingredients(req, q)$ vrača seznam vseh tistih sestavin, ki sestavljajo recepte izbrane vrste jedi v navadnem tekstovnem formatu.

Uporabnik lahko s kliki označi poljubne sestavine oziroma sestavine, ki jih ima pri roki in bi jih rad porabil pri pripravi jedi. Ob vsakem kliku, se tako kot zgoraj, na strežnik asinhrono pošlje zahteva z informacijo o izbranem identifikatorjem vrste jedi in seznamom identifikatorjev uporabnikovih izbranih sestavin (Slika 4.4).

Slika 4.4: Slika prikazuje asinhrono zahtevo POST na strežnik, ob izboru sestavine.

Metoda $whatToAdd(req, dishType, ingListIDs)$ sprejme tri argumente. Prva dva sta ista kot kot pri zgoraj omenjeni metodi $ingredients()$, tretji argument pa je seznam identifikatorjev sestavin. Metoda vrne seznam sestavin, ki se največkrat pojavljajo skupaj s sestavinami, ki jih je pred tem izbral uporabnik. Ta seznam uporabniku olajša nadaljno izbiro sestavin, saj so v seznamu tiste sestavine, ki se pri pripravi jedi največkrat pojavljajo skupaj s sestavinami, ki jih je izbral uporabnik.
Ko se metoda $whatToAdd(req, dishType, ingListIDs)$ uspešno izvede se začne izvajati metoda $whatToAdd2(req, dishType, ingListIDs, sort)$. Postopek pošiljanja je isti kot pri prejšnjih, le da se na strežnik posreduje tudi identifikator sortiranja. Ko se metoda $whatToAdd2$ uspešno izvede se uporabniku prikažejo najprimernejši kuharski recepti. Po prevzeti vrednosti so le ti sortirani po številu sestavin, ki jih uporabnik ima. Več uporabnikovih izbranih sestavin vsebuje kuharski recept, višje na strani se pojavi najbolj primeren recept (Slika 4.5). Uporabnik, lahko sortira kuharske recepte tudi po številu manjkajočih sestavin. Manj sestavin kot jih uporabniku primankuje, višje na strani se pojavi najbolj primeren recept.

Slika 4.5: Slika prikazuje seznam receptov jedi, ki si jih lahko uporabnik sestavi iz izbranih sestavin.

Da bi uporabnika seznanili z funkcionalnostjo in uporabo spletne aplikacije, smo na vrhu strani dodali avdio zapis informacij, ki jih lahko nov uporabnik posluša ob kliku na logotip aplikacije.

Slika 4.6: Uporabnik ima v aplikaciji možnost poslušanja avdio zapisa, v katerem je predstavljena uporabnost aplikacije.

Poglavje 5

Sklepne ugotovitve

V diplomskem delu so opisani postopki za gradnjo in vizualizacijo mrež jedi. Za gradnjo in vizualizacijo mrež jedi, smo uporabili programski jezik Python in njegovi knjižnici NetworkX in Matplotlib. Mreže jedi smo gradili iz kuharskih receptov, ki smo jih dobili na spletni strani (http://www.mojirecepti.com) s spletnim razčlenjevalnikom Beautiful Soup. V mrežah jedi, ki smo jih gradili in vizualizirali so bila vozlišča različne jedi, povezave med njimi pa so bili odstotki podobnosti med jedmi. Pri gradnji mrež jedi smo v mrežo vstavljali samo tiste jedi, katerih odstotek podobnosti je bil nad določeno mejo. Podobnost med jedmi smo računali z Jaccardovim podobnostnim koeficientom. Knjižnici NetworkX in Matplotlib nam nista omogočili načina vizualizacije, da bi bila vozlišča in povezave optimalno porazdeljene v koordinatnem sistemu. Z njuno uporabo smo dobili nepregledne, neberljive grafe, ker so se povezave in vozlišča prevečkrat križale. S programskim orodjem Graphviz nam je na podlagi opisa grafa uspelo vizualizirati mrežo jedi tako, da so bila vozlišča in povezave optimalno porazdeljene v koordinatnem sistemu. Grafi, ki smo jih vizualizirali s programskim orodjem Graphviz so bili pregledni in iz njih se je dalo razbrati, katere jedi so si po sestavinah najbolj podobne.

V drugem delu diplomske naloge smo razvili spletno aplikacijo. Aplikacija ima funkcijo kuharskega asistenta. Uporabniku ponudi seznam kuharskih receptov, ki si jih lahko pripravi iz njegovih izbranih sestavin sortirano po številu sestavin, ki jih ima. Aplikacija je dostopna na spletni domeni (http://www.mizicapogrnise.si). Z uporabo spletnega orodja Google Analitics za spremljanje in analizo spletnega prometa strani smo ugotovili, da je stran v enem tednu obiskalo več kot 3000 različnih uporabnikov, njihovo število pa vztrajno narašča.

Za nadaljni razvoj omenimo le nekaj možnosti, ki zadevajo spletno aplikacijo. Podatkovno bazo aplikacije bi lahko razširili z novimi tipi jedi in recepti. Tako bi uporabniku ponudili širšo paleto kuharskih receptvo in lažjo izbiro. Naslednja razširitev bi bila implementacija ocenjevanja in komentiranja kuharskih receptov. Uporabnik bi tako od drugih uporabnikov dobil dodatne informacije o jedeh, ki si jih želi pripraviti, na primer koliko napora je potrebno dodati v samo pripravo jedi, katere jedi so si podobne.

Literatura

[1] D.Janez *Python za programerje*, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Ljubljana, 2008.

[2] M.H. Lie, *Beginning Python: From Novice to Professional*, New York: Micheal Apress, 2005.

[3] W.John, *An open graph visualization system and its applications to software engineering*, New York: AT&T Labs - Research, Shannon Laboratory, 1999.