SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Filip Novački

NASLOV ZAVRŠNOG/DIPLOMSKOG RADA – LATEX PREDLOŽAK

PROJEKT

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Novački

Matični broj: 35918/07–R

Studij: Informacijski sustavi

NASLOV ZAVRŠNOG/DIPLOMSKOG RADA – LATEX PREDLOŽAK

PROJEKT

Mentor:

dr. sc. Bogdan Okreša Đurić

Varaždin, rujan 2020.

Filip Novački

Izjava o izvornosti

Izjavljujem da je moj projekt izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Opsega od 100 do 300 riječi. Sažetak upućuje na temu rada, ukratko se iznosi čime se rad bavi, teorijsko-metodološka polazišta, glavne teze i smjer rada te zaključci.

Ključne riječi: riječ; riječ; ...riječ; Obuhvaća 7+/-2 ključna pojma koji su glavni predmet rasprave u radu.

Sadržaj

1. Uvod

Ovo je projektna dokumentacija projekta za kolegij Teorija baza podataka na diplomskom studiju na Fakultetu organizacije i informatike Sveučilišta u Zagrebu. Cilj projekta je demonstrirati koncepte baza podataka koji nisu uobičajeni kao što su relacijske baze podataka ili se radi o naprednijim funkcionalnostima relacijskih baza.

Glavna tema ovog projekta je `ZODB` baza podataka te aplikacija koja se njom koristi kao glavnim izvorom podataka.

2. Opis projekta

2.1. Opis aplikacijske domene

Domena ovog projekta stvaranje je rječnika i analiza riječi. Analizu riječi omogućuju biblioteke za neurolingvističko programiranje koje imaju poprilično širok fond raznih funkcionalnosti kojima se riječi mogu analizirati.

Glavna funkcionalnost za korisnika je vrlo jednostavna, a vrlo je intenzivna sa strane poslužitelja koji radi niz koraka kako bi ju ostvario:

- unos teksta
- čišćenje teksta i odvajanje riječi
- analiza riječi
- unos u bazu podataka.

Unos teksta zapravo nije predviđen kao proces gdje korisnik *unosi* tekst, već kao proces gdje korisnik lijepi veću količinu teksta (članci, objave itd.) te se daje aplikaciji na analizu. Taj tekst ne mora biti ni na koji način prilagođen – aplikacija je sposobna u potpunosti izbaciti viškove kao što su točke, zarezi i drugi znakovi, riječi bez značenja, brojeve (znamenke), kratke riječi koje ne nose značenje nego su isključivo pomoćne naravi (en. *stopwords*) itd.

Analiza riječi je proces u kojem se unesenim riječima pripasuje značenje, izgovor, vrsta riječi i ostale informacije te se pripremaju za spremanje u bazu. Unatoč tome što je ovo vrlo jednostavno opisati u prirodnom jeziku, upravo ovaj korak poslužitelju traje najduže.

Unos u bazu podataka korak je gdje se pripremljene informacije zapisuju u bazu. Kod unosa obraća se pozornost na rječnik u kojeg se nešto dodaje i na već postojeće riječi kako ne bi došlo do zapisivanja duplikata.

Osim osnovne funkcionalnosti, aplikacija generira engleski rječnik iz rječnika koji se nalazi u bazi podataka.

2.2. Teorijska podloga baze podataka

`ZODB` je objektno orijentiran sustav za upravljanje bazama podataka pisan u Pythonu. To znači da se u bazu pohranjuju objekti kojima se kasnije pristupa. Kad govorimo o spremanju objekata u Pythonu moramo se prisjetiti biblioteke `pickle` koja služi za serijalizaciju objekata iz Pythona u datoteke. Kombinacija ta dva pristupa omogućuju spremanje podataka u hijerarhiju koja je nalik Pythonovim objektima, a spremljena je na disku računala.

Takav pristup omogućuje pristupanje svim objektima, odnosno cijeloj bazi bez uporabe prilagođenog jezika kao što je `SQL` te se pristup podatcima može jako dobro integrirati u ostatak

koda.

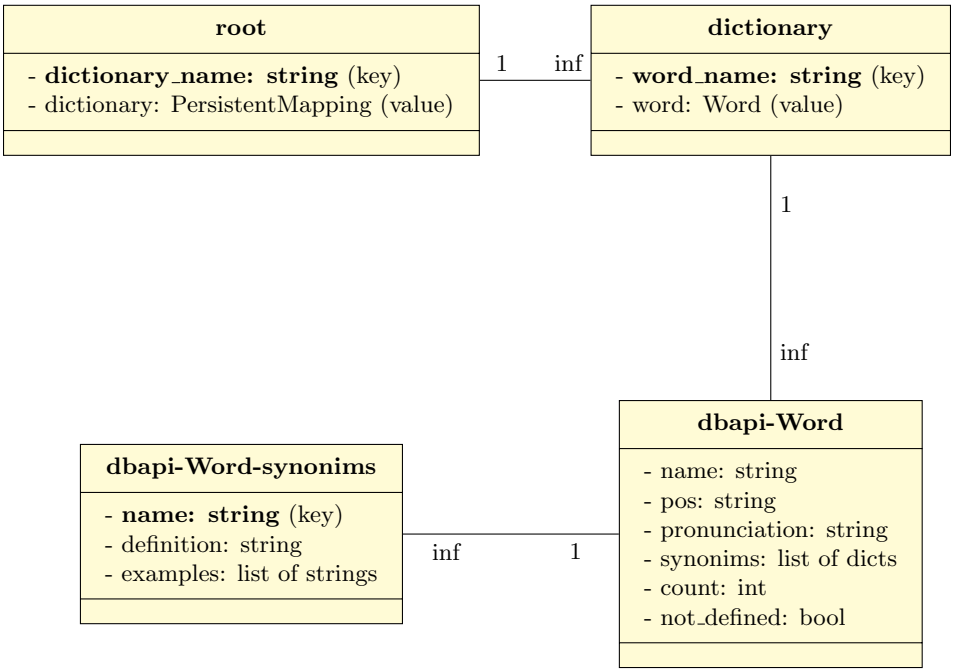
Objekti koji se spremaju u objektnu bazu moraju naslijediti klasu `Persistent` iz biblioteke `persistent` ili se moraju koristiti *non-mutable* objekti kao što su n-torke (*tuples*), stringovi i drugi jednostavni tipovi podataka. Od tipova podataka ugrađenih u Python promjenjivi (*mutable*) su liste (`list`), rječnici (`dict`) i skupovi (`set`).

Za tipove podataka koji jesu promjenjivi napravljeni su posebni tipovi koji se ponašaju slično Pythonovim pretpostavljenim tipovima podataka, a prilagođeni su za korištenje u `ZEO` serverima, a tako su i primjenjivi u `ZODB` bazama podataka.

Kod stvaranja vlastitih klasa koje nasljeđuju klasu `Persistent` potrebno je i promijeniti varijable koje označavaju jesu li se dogodile promjene u toj klasi, odnosno varijablu `_p_changed` je potrebno postaviti na `True`. Tako `ZODB` *zna* da se promjena dogodila i da je potrebno ponovno pohraniti taj objekt u bazu.

2.3. Model baze podataka

Na slici 1 prikazan je model baze podataka. Kod korištenja objektne baze podataka jednostavnije je koristiti UML dijagram umjesto ERA modela unatoč tome što izgledaju slično za danu bazu. Ovakav model ostavlja mjesto i za popisivanje metoda koje se nalaze u pojedinoj klasi, no i to se izbjegava u ovom slučaju s obzirom na to da poslovnu logiku nije potrebno spremati u bazu podataka.



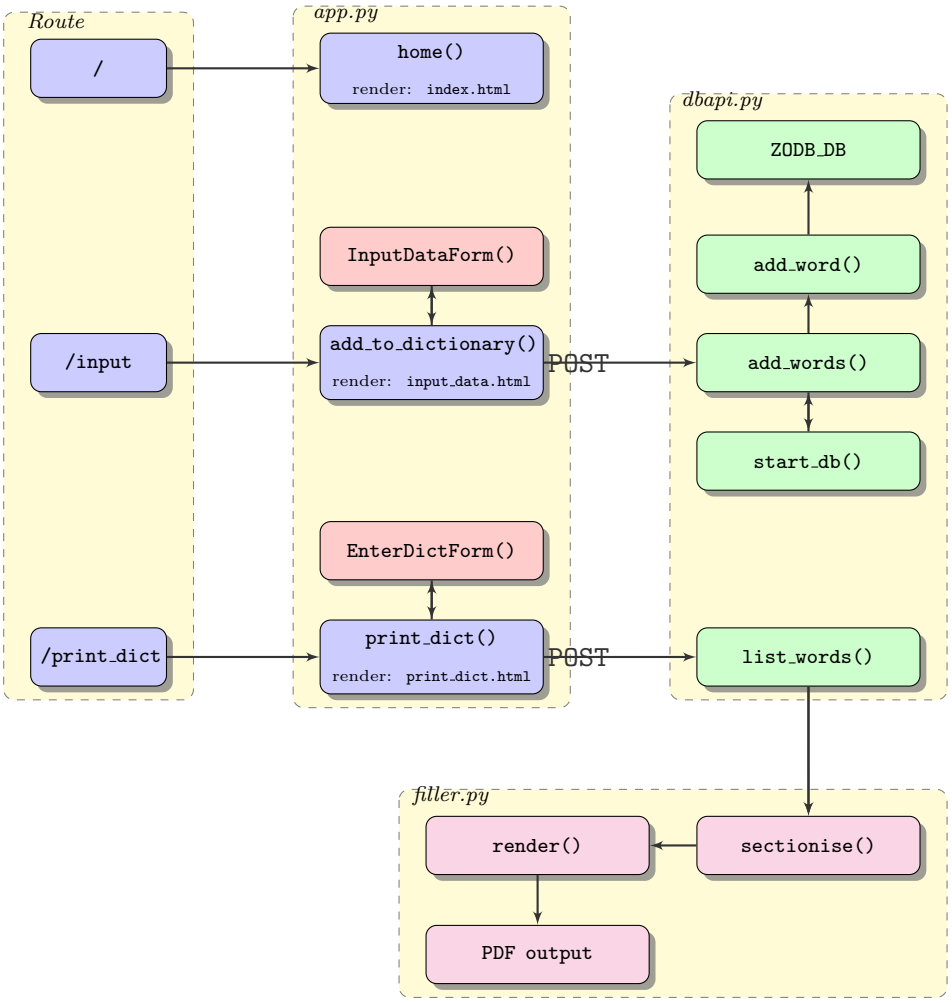
Slika 1: Model baze podataka prikazan pomoću UML diagrama. Kvadrati su klase, masno otisnuti su ključevi, a tipovi podataka su iza dvotočja. Veze između klasa slovima su označene na vezama.

Za ovu domenu objektna baza je izrazito učinkovita jer su klase nekoliko puta ugniježđene jedna u drugu, a u relacijskoj bazi podataka bi to moglo biti neučinkovito jer se za

traženje pojedine vrijednosti moraju proći svi ostali objekti, odnosno zapisi, kojih može biti izrazito mnogo. Zbog pristupa koji je zasnovan na rječnicima, taj problem je spriječen jer se rječnici u Pythonu indeksiraju i pristup im je brz, a svaki objekt koji pripada rječniku ima direktnu vezu spremljenu do njega.

2.4. Implementacija baze podataka

Kao kratki uvod u implementaciju na slici 2 prikazana je karta projekta.



Slika 2: Karta projekta s prikazanim značajnim točkama u kodu

Projekt je napravljen od ovih glavnih elemenata:

- `Flask` server
- ZODB baza podataka

`Flask` server poslužuje cijelu aplikaciju i prikazuje ju sučelju koje je prilagođeno za web preglednike. *Frontend* je izrazito jednostavan - ne koristi se `JavaScript`, a `CSS` koji se koristi je `Bulma`, *open source* projekt koji pristojno i jednostavno izgleda, a značajno poboljšava korisničko iskustvo nad pretpostavljenim izgledom `HTML` datoteka.

Sama aplikacija vrlo je jednostavna i nalazi se u datoteci `app.py`. Na slici 2 razložena je podjela aplikacije tako da se vide putanje s koje se aktiviraju određene funkcije. Tako na putanji `/` aktivira se funkcija `home()` te se renderira `index.html`. Za renderiranje `HTML`-a koristi se `Jinja`, biblioteka koja je integrirana u `Flask`.

Osim za renderiranje `HTML`-a `Jinja` se koristi i za renderiranje LaTeX dokumenta - predložak za rječnik puni se podatcima iz baze te se renderira.

Na sličan način kao što je opisano radi i ostatak aplikacije stoga on neće biti u detalje opisan ovdje.

2.4.1. Korištene tehnologije

Kao što je već napomenuto, glavne okosnice projekta su `Flask` i `ZODB` koji pokreću cijeli sustav. Osim tih tehnologija korišteni su i:

`Jinja` – renderiranje `HTML`-a i LaTeX-a

`persistent` – tipovi podataka za rad s bazom podataka

`nltk` – većina obrade riječi i dobavljanje njihovog značenja

`textblob` – također za obradu riječi, ali na višoj razini od `nltk`-a. `textblob` je implementiran funkcijama `nltk`-a

`os` – Pythonova biblioteka za rad s naredbama operacijskog sustava, korišteno za pokretanje naredbi u ljusci te za provjeravanje postojećih datoteka u sustavu

`wtforms` – biblioteka za kreiranje formi, integrirano s `Flaskom` i `Jinjom`

LaTeX – služi za renderiranje PDF datoteke iz teksta

ostalo – nekoliko Pythonovih biblioteka za rad s operacijskim sustavom, uglavnom korišteno za uredan rad s LaTeX-om

2.4.2. Problemi u implementaciji

S obzirom na to da se ovaj projekt bavi bazama podataka, manje se pozornosti stavljalo na korisničko iskustvo. S tim u vidu jedan problem ostaje neriješen – rad stranice za vrijeme intenzivnog rada s bazom.

U slučaju kad se javi neki malo intenzivniji poduhvat za bazu dogodi se da se stranica ne učita dok se sva obrada ne dovrši, odnosno procesuiranje se odvija sinkrono. U projektu se taj problem događa u najmanje dva slučaja:

- unos riječi
- dohvaćanje riječi iz baze

Problem kod unosa riječi je taj što analiza riječi dugo traje – pregledava se baza engleskih riječi koja ima preko 130 kilozapisa, a iz te baze se izvlače izgovori, sinonimi, vrste riječi i objašnjenja sinonima. Trenutno rješenje radi tako da se slijedno analiziraju i zapisuju podatci, a rješenje problema bi bilo da se poslužitelju predaje lista riječi, a da on u pozadini obrađuje dane riječi, a da aplikacija dalje normalno radi. Osim asinkronog pristupa, moguće je i pokrenuti drugi poslužitelj koji će raditi isključivo obradu podataka te na taj način rasteretiti poslužitelj na kojem je aplikacija. To je također asinkroni pristup, ali se pristup razlikuje u osnovnom principu koji ga pokreće.

Također je moguće i ostaviti učitavanje kako jest te pomoću `JavaScripta` i `AJAX` tehnologije napraviti učitavanje stranice, no to je ipak tema za neki drugi projekt.

Kod dohvaćanja liste riječi postoji problem ukoliko se ponovno pozove funkcija za prikaz rječnika, a baza podataka se u međuvremenu nije zaključala. Ukoliko se to dogodi, aplikacija pokušava otključati već otključanu bazu i aplikacija se ruši. Iz tog se razloga korisnika moli za oprez.

2.4.3. Problematika implementacije baze podataka

Baza podataka implementirana je u Pythonu koristeći funkcije baze. Unatoč tome što su `ZODB` baze podataka predviđene za neupadljivu integraciju u ostatak Python koda, korisno je bilo odvojiti funkcije baze podataka od serverske logike te od poslovne logike.

Osim toga, `ZODB` baza jako je osjetljiva na istovremeni pristup podatcima te na pristupanje objektima kad je baza zaključana. Stoga je idealno bazu držati što kraće otključanom, pristupiti podatcima i neposredno nakon toga zaključati ju.

Problem na koji se može naići u pisanju aplikacije za bazu podataka je u slučaju da korisnik želi neku klasu koja je pospremljena u bazi koristiti za obradu. Jedna opcija je napraviti obradu za vrijeme dok je baza otključana, a druga je da se napravi novi objekt u koji će se ti podatci zapisati. To su jedine dvije opcije jer nakon što se baza zaključa nije moguće pristupiti objektu koji je u bazi unatoč tome što u programu postoji referenca na taj objekt.

Popis slika

Popis tablica

1. Prilog 1

2. Prilog 2