

*Prolećni semestar, 2016/17*

*PREDMET: CS324 Skripting jezici*

*Projektni zadatak*

WebTech web scrapping

Ime i prezime studenta: **Nemanja Kuzmanovic**

Broj indeksa: **2851**

Ime i prezime profesora: **doc dr Igor Franc**

Ime i prezime asistenta: **Milan Hakaj**

Beograd 2017.

Sadržaj

[Abstrakt: 3](#_Toc490694790)

[Uvod: 4](#_Toc490694791)

[Projekat: 4](#_Toc490694792)

[Zaključak: 10](#_Toc490694793)

[Literatura: 10](#_Toc490694794)

# Abstrakt:

Radio bih web scrapping, to jest izvlacenje informacija sa odredjenih web sajtova. Konkretno, aplikacija bi najverovatnije bila u python-u (Django) (verovatno bih koristio beatuifulsoup), veb aplikacija je u pitanju, i sluzila bi sledecoj svrsi,  
naime, skidao bih svakog dana sve komponente sa sajtova koji posluju kod nas i bave se prodajom racunara i komponenata kao sto su Gigatron,WinWin itd. te skinute objekte bi cuvao u bazi podataka i korisnik bi mogao  
da pretrazuje preko Search-a samu bazu preko kljucnih reci, npr ukoliko bi ukucao fx 6300 izasli bi svi delovi sa tom kljucnom recju, a kada korisnik odabere odredjeni racunar koji imaju ovu kljucnu rec (konfiguraciju) ili samu komponentu  
izlistavao bih specifikacije iz baze o samom objektu, kao i listu svih prodavnica u kojima se ta komponenta nalazi kao i cene u tim prodavnicama (azurirao bih verovatno svaki dan bazu u 00:00 kada bih povlacio sve elemente sa tih sajtova),  
mozda bih koristio odredjene delove drugih skripting jezika kao sto je php ako bih zaglavio u nekom delu, ali o tome bih se vec konsultovali, dodatno verovatno bih implementirao i deo sa kategorijama gde bi se korisniku prikazale  
sve kategorije proizvoda iz baze pa bi mogao manuelno da bira, ali bi glavni fokus bio na search-u, pretrazivanju po kljucnoj reci iz baze, i web scrapping-u tj, skidanju tih komponenata,slika, komentara, cena, recenzija... sa ovih sajtova..  
Nisam toliko upoznat sa tematikom abstrakta koji predlazem te se nadam da nije bas prekomplikovano, i ovo sam vec neko vreme hteo da pokusam samo sto bih radio sa drugim elementima ali mi i ove komponente hardvera  
deluju zanimljivo pa bih se oprobao u ovoj sferi. Kakvo je Vase cenjeno misljenje, da li je ovo nesto izvodljivo i ima li nekih prekomplikovanih delova u kojima bih se zaglavio mozda da znam da obratim paznju?

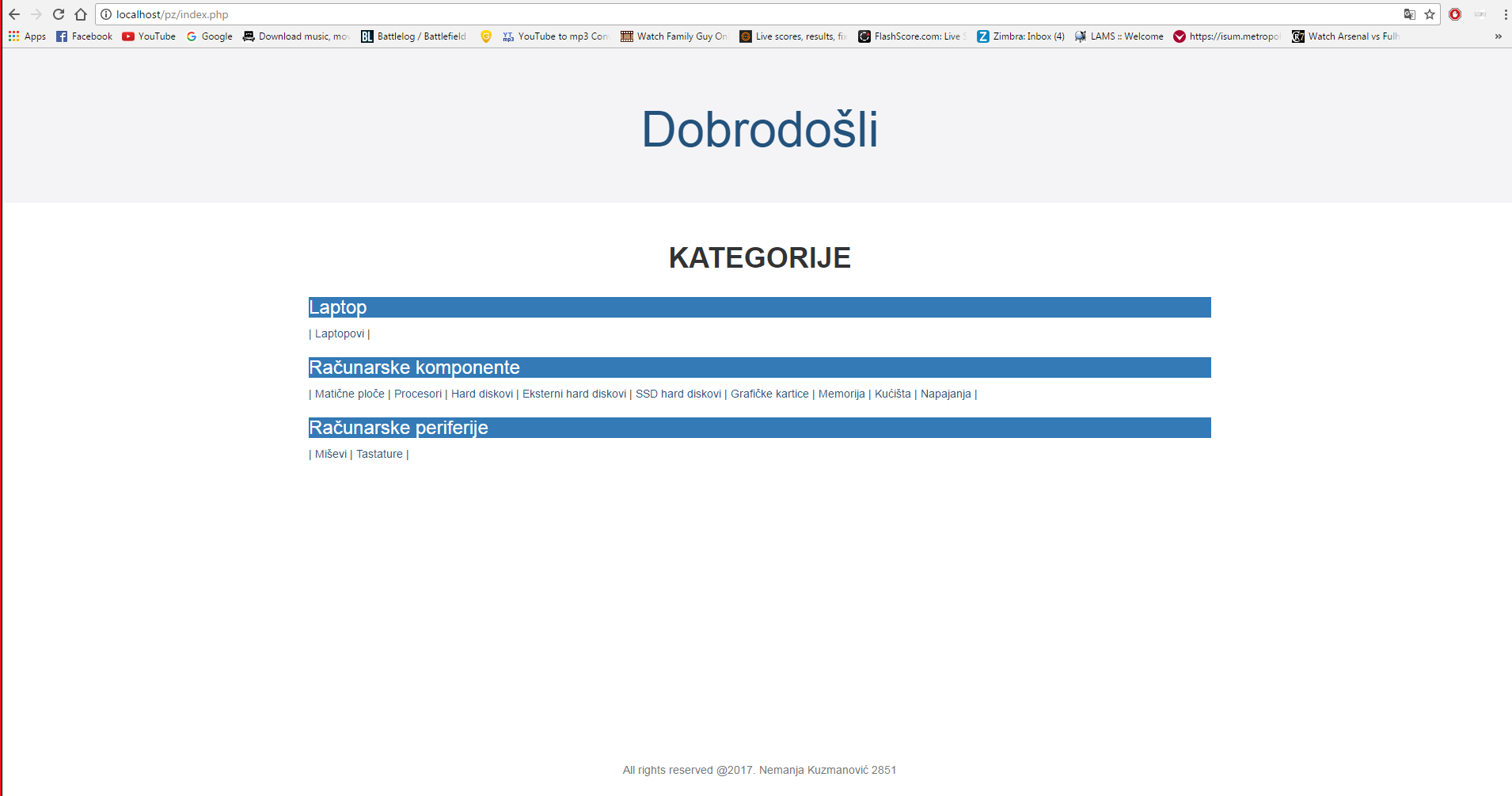
# Uvod:

U abstraktu projekta može se videti želja kako je projekat zamišljen bio prvobitno, međutim, iz tehničkih razloga koji će kasnije biti detaljno obrazloženi, neki delovi su izmenjeni, ali je suština projekta ostala ista, dakle, imamo sajt sa kog skrepujemo podatke, punimo našu bazu, te korisnicima nakon toga ‘’iscrtavamo’’ na sajtu komponente konkretno vezane za računare gde se lako projekat dalje može proširiti i ka ostalim stvarima poput bele tehnike i slično, jer glavni template imamo, i moguće je, dakle, nadograditi ostalim potrebnim stavkama.

# Projekat:

Prvenstveno, najbitnija i najveća promena u odnosu na abstrakt koja je na projektu napravljena je da je python jezik zamenjen php-om. Ovo je moralo biti izvedeno iz tehničkih razloga, naime, beautifulsoup koji postoji i namenjen je web scrappingu, u python-u, nije baš jednostavan za korišćenje, i nema mnogo dokumentacije, a pored toga, većina studenata, je više i bolje upoznata sa php-om nego sa pythonom, te su problemi koji su nastali, bili dosta lakši za rešiti. Ne zbog kompleksnosti problema, nego zbog poznavanja jezika. Uz php, koristi se simple\_html\_dom pomoću kojeg je dosta intuitivnije rešavanje, tačnije scrappovanje, nego pomoću beatufullsoap-a. Pored ove promene ima još jedna veća, a to je, da iz želje da se pokaze rukovanje scrappingom, jedan deo projekta se bavi time da se skida sadržaj sa sajta, te se ubacuju podatci u bazu podatata, dok se drugi deo projekta bavi time, da se direktno sa sajta sirovi podatci skidaju i odmah prikazuju na sajtu, kao što je već naglašeno, kako bi se pokazalo rukovanje scrappingom. Zbog ovakve odluke, search je bilo jako tesko i komplikovano implementirati, iz razloga što se podatci skidaju tek onda kada korisnik otvori neku kategoriju, a ne čim otvori sajt. Naravno, u realnoj situaciji bi to bilo mnogo drugačije, jer bi se sajt ažurirao na određeno vreme te bi korisnici imali besprekorno korišćenje sajta, međutim, za potrebe projekta, implementirano je da čim korisnik otvori kategoriju, u realnom vremenu se popunjava baza podataka te se podatci prikazuju korisniku, dok se sa druge strane, takođe u realnom vremenu, trenutno, podatci skidaju ali ne upisuju se u bazu, nego se direktno korisniku prikazuju putem našeg sajta. Još jedan od problema koji je nastao je da sajtovi uglavnom nisu najsloženiji sa tim da im neko skida sadržaj zarad svojih potreba, te ukoliko se puno podataka skida, sam server će ubotrebiti bezbednosne mehanizme i blacklistovati IP adresu, što se u izradi projekta i desilo, više puta, te je tražena dozvola od administratora sajta, koja je kasnije i dobijena međutim nisu želeli da zaštite ni jednu IP adresu te je projekat u stanju u kakvom je morao biti prekinut. Rešenje ovog problema nije teško naći, a to je, da se jednom popuni baza podataka i da se više sajt ne dira, međutim, namena ovog projekta je bila malo drugačija, gde je dosta stvari i naučeno, kako neke stvari funkcionišu i koliko se lako i efikasno server brani od amatera, a s druge strane blokira svaki pokušaj botova da skidaju veći sadržaj sa sajta. Mora se i napomenuti, da se nisu uzimali sajtovi koji su navedeni u abstraktu poput gigatrona, winwina i slično, nego jedan sajt koji objedinjuje sve domaće sajtove koji se bave tehnikom (većinu), a to je eponuda.com. Razloga većeg za ovo nema, samo to što im je sajt user-friendly te je manje teško skrepovati sadržaj nego na ostalim sajtovima, dok je suština ista za svaki sajt i može se lako i jednostavno skrepovati skoro svaki sličan veb sajt. Dakle, još jednom, pre nego se krene detaljnije u projekat, suština je ostala ista, i izvedeno je ono planirano.

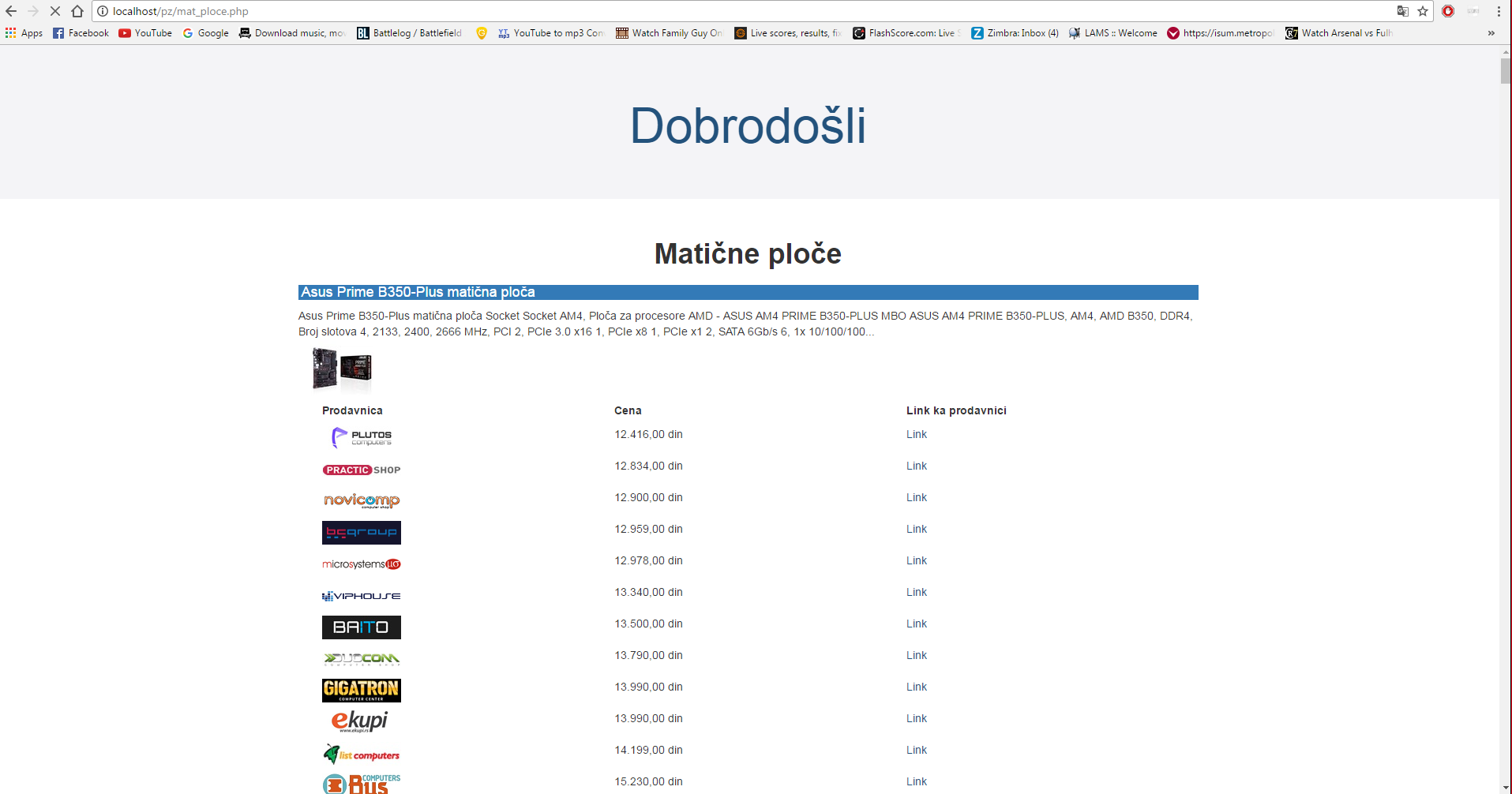
Napomena: što manje puta refrešovati stranice, i ne otvarati baš svaku kategoriju, ukoliko ne želite da Vas server eponuda.com vebsajta blacklistuje, posle pokretanja projekta.



Slika 1: Početna stranica

Početna stranica je bez mnogo dizajna, jednostavna, i prikazuje potrebne informacije kako bi korisnik lako prešao na deo koji ga zanima.

Može izabrati jednu od nekoliko kategorija, te neku od komponente tih kategorija.

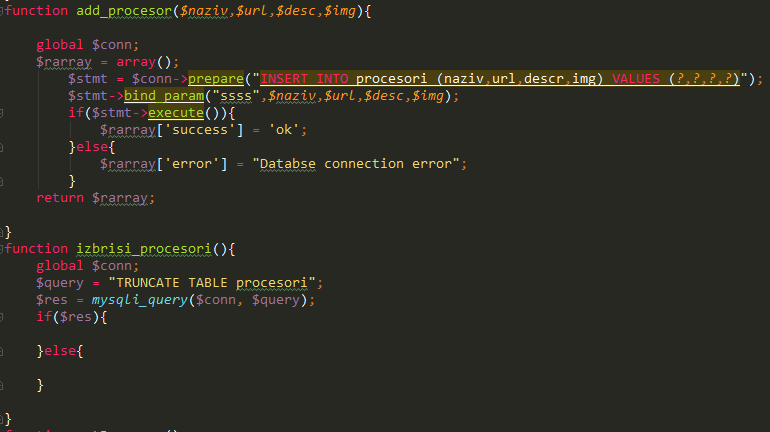


Slika 2: Komponente I kategorije

Recimo, da je korisnik odabrao neku komponentu određene kategorije, konkretno, kategoriju računarske komponente, te matične ploče, dobiće listu svih matičnih ploča sa kratkim opisom iste, slikom komponente, logo radnje u kojoj može kupiti tu komponentu, cene iste u svakoj od prodavnica, te link ka prodavnici, gde može dobiti detaljne podatke o svakoj komponenti, gde je dalje može i naručiti ukoliko prodavnica ima tu mogućnost. Dakle sajt je veoma jednostavan, i ima ulogu da korisnik na jednom mestu nađe željenu stvar, te vidi gde je najpovoljnija ista.

Jedina zamerka koja se od samog početka nameće je search opcija, koja nije trenutno implementirana zbog kompleksnosti, međutim, ista bi u budućnosti svakako dodala novu dimenziju i mnogo lakšu pretragu na sajtu.

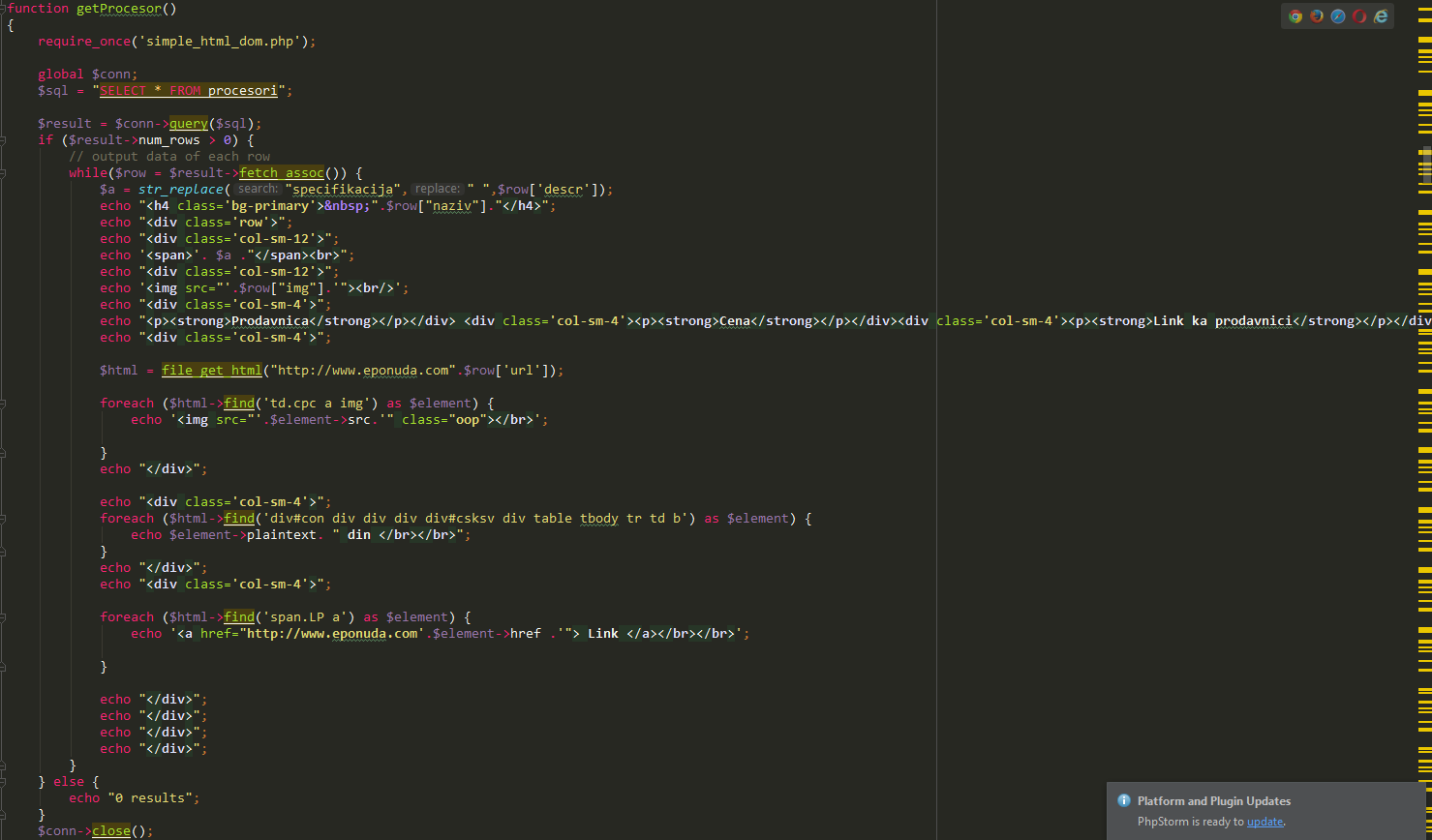
Još jedna začkoljica je da je se moralo ograničiti broj prikazanih komponenti, npr. Veb sajt sa kog se skidaju ima preko 1000 artikala, ali da ne bi došlo do bleklistovanja, ograničeno je skidanje komponenti na jednu njihovu stranicu, to jest, 25 komponenti. Da nije postojala ova barijera, implementrane bi bile stranice (1,2,3,4,5...) gde bi svaka prikazivala 25 komponenti.



Slika 3: Funkcije

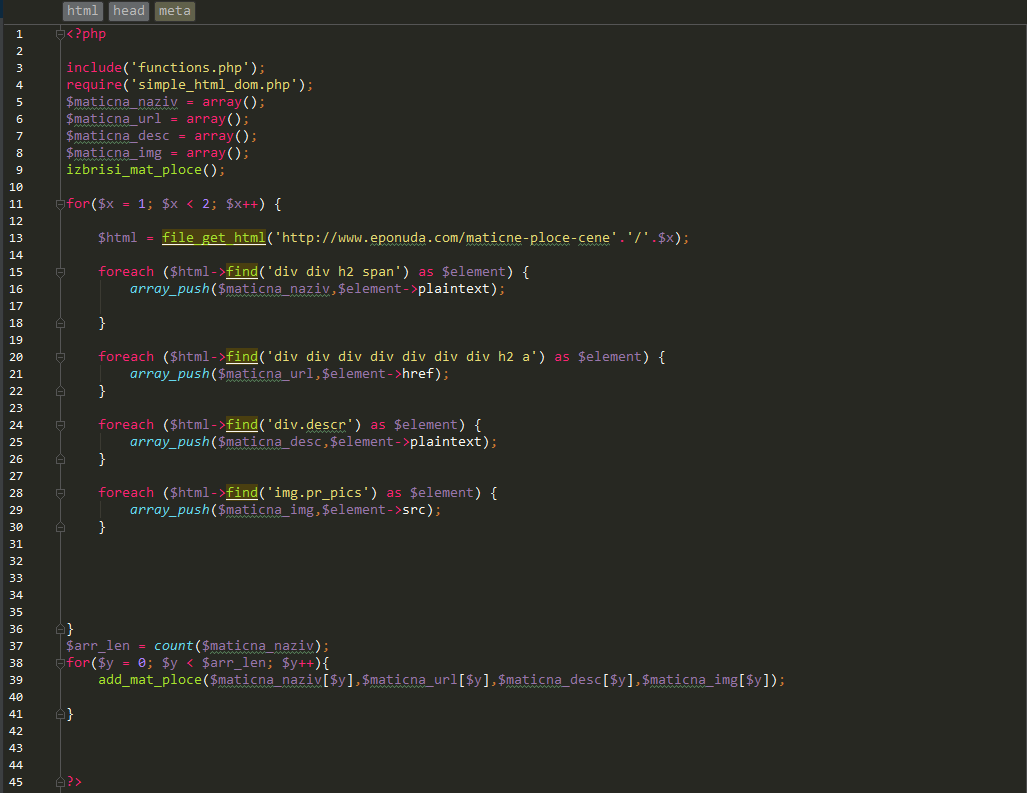
Prvenstveno, pre samog skrepovanja podataka, imamo bazu sa određenim tabelama, i kolonama, u koje će se kasnije upisati određeni podatci.

Sa druge strane, u funkcijama, koje su bitnije od same baze i tabela u istoj, za svaku komponentu (npr. Procesor) imamo tri glavne funkcije, a to su, addprocesor koja prima parametre naziv, url, opis i slika, gde su ti parametri zapravo oni podatci koje skidamo sa sajta skrepovanjem. Unutar ove funkcije imamo jednostavan insert, koji zbog sigurnosti urađen preko prepared statementa, i on nam služi da te podatke popunimo u samoj bazi. Pored ove imamo i izbriši procesor funkciju koja nam radi pražnjenje naše tabele u bazi podataka, kako bi korisnik svaki put kada otvori stranu imao sveže ažuriran sadržaj. Naravno, kao što je već rečeno, u reanoj situaciji bi se ovo drugačije rešilo nego što je za potrebe projekta.



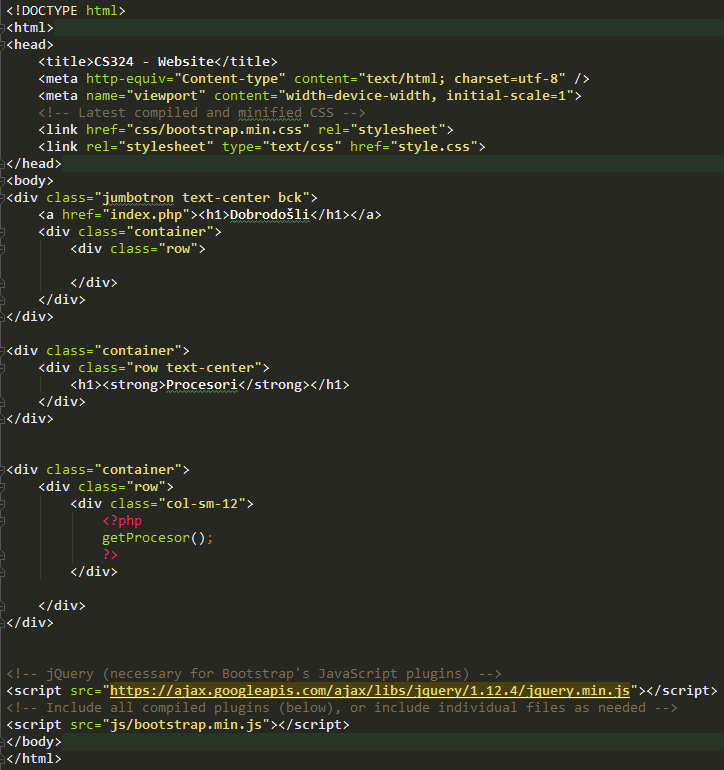
Slika 4: Get funkcija

Kao jedna od najbitnijih stvari u ovom projektu, prikazana će biti get funkcija koju ima svaka komponenta. Dakle, posle ubacivanja podataka u bazu poziva se ova funkcija get, koja uziva sve podatke iz određene tabele putem jednostavnog sql upita, te se pomoću bootstrapa (framework css-a napravljen od strane twittera) iscrtava korisnicima na sajtu. U donjem delu koda, tačnije od polovine, možemo videti $html promenljivu kojoj je nešto dodeljeno. To nešto je funkcija simple\_html\_dom\_parser-a koja kao parametar uzima sajt sa kog će se skrepovati podatci. U nastavku će biti detaljnije objašnjeno kako se izvršava sam veb skreping, ali se sada mora pomenuti jer je bitan deo. Dakle, prosleđuje se vebsajt, i kolona iz baze nazvana url, ovo se radi jer svaka komponenta ima svoju stranu details na sajtu, u kojoj se nalaze detaljno cene u svakoj prodavnici kao i linkovi ka tim prodavnicama. U nastavku, imamo nekoliko foreach petlji, pomoću kojih funkcijom find() ‘pokazujemo’ ka elementima vebsajta čije podatke želimo da skrepujemo. Ukratko, u prvom delu iscrtavamo na sajtu podatke iz baze, dok u drugom delu funkcije u realnom vremenu skrepujemo detalje o svakoj komponenti i direktno sa sajta eponuda prikazujemo na naš sajt.



Slika 5: Simple\_html\_dom\_parser

Na samom početku imamo for petlju koja prolazi kroz stranice sajta i skida sadržaj, dakle, ukoliko bi postojale tehničke mogućnosti, mogli bi pustiti petlju da ide do 10 npr, što bi značilo da skrepujemo 10 stranica sa najnovijim sadržajem sajta eponuda. Na sličan način kao u prethodnom delu, funkcijom find pokazujemo ka elementu od koga želimo da uzmemo podatke, i petljom foreach upisujemo u arraylistu, iz razloga kako bi te podatke posle ubacili u samu bazu podatka. Dakle ovo je deo koji se tiče skrepovanja i slanja podataka u bazu. Na samom kraju možemo videti petlju koja poziva funkciju add gde prosleđujemo kao parametre naše liste sa $y mestom (prvo mesto svake liste, drugo mesto svake liste...) i upisujemo u bazu.



Slika 6: Sajt

Sajt je urađen putem htmla, i bootstrapa, i ovde se može to i videti, jednostavan sajt bez mnogo opcija i mogućnosti. Imamo php kod gde pozivamo get funkciju koja nam izlistava sve podatke koji se vide na sajtu, što iz baze što skrepovanjem uzivo.

Veb sajt nije dizajnerski uređen jer to nije poenta ovog predmeta, što se naravno moglo doterivati ali nije poenta istog. Sa druge strane, dodatne funkcije poput sortiranja, iz istog razloga kao search nisu implementirane.

U daljoj budućnosti, može se desiti da se isključi opcija skrepovanja u realnom vremenu, te se svi podatci koji se skrepuju ubace direktno u bazu, što automatski donosi veoma jednostavnu implementaciju search opcije kao i sortiranja.

# Zaključak:

Iako su postojali problemi u toku izrade projekta, i nisam uspeo da u potpunosti implementiram ono što sam želeo te sam bio ograničen zbog vebsajta sa kog su skidani podatci, upotrebio sam i implementirao većinu stvari koja je tražena za ovaj projekat a koju sam u abstraktu napisao da će biti urađena. Naučeno je dosta novih stvari, koje će svakako služiti dobro u budućnosti, gde je jedna od njih ukoliko se radi skreping da se dobro istraži koliko i šta server sajta dozvoljava da se skida. Naravno API bi u ovakvim slučajevima dobro došao a veći i obiljniji sajtovi ga i imaju, baš da ne bi dolazilo do ovakvih problema.

# Literatura:

Lams.metropolitan.ac.rs

Stackoverflow.com

<http://simplehtmldom.sourceforge.net>

<https://www.w3schools.com>

php.net

getbootstrap.com

youtube.com