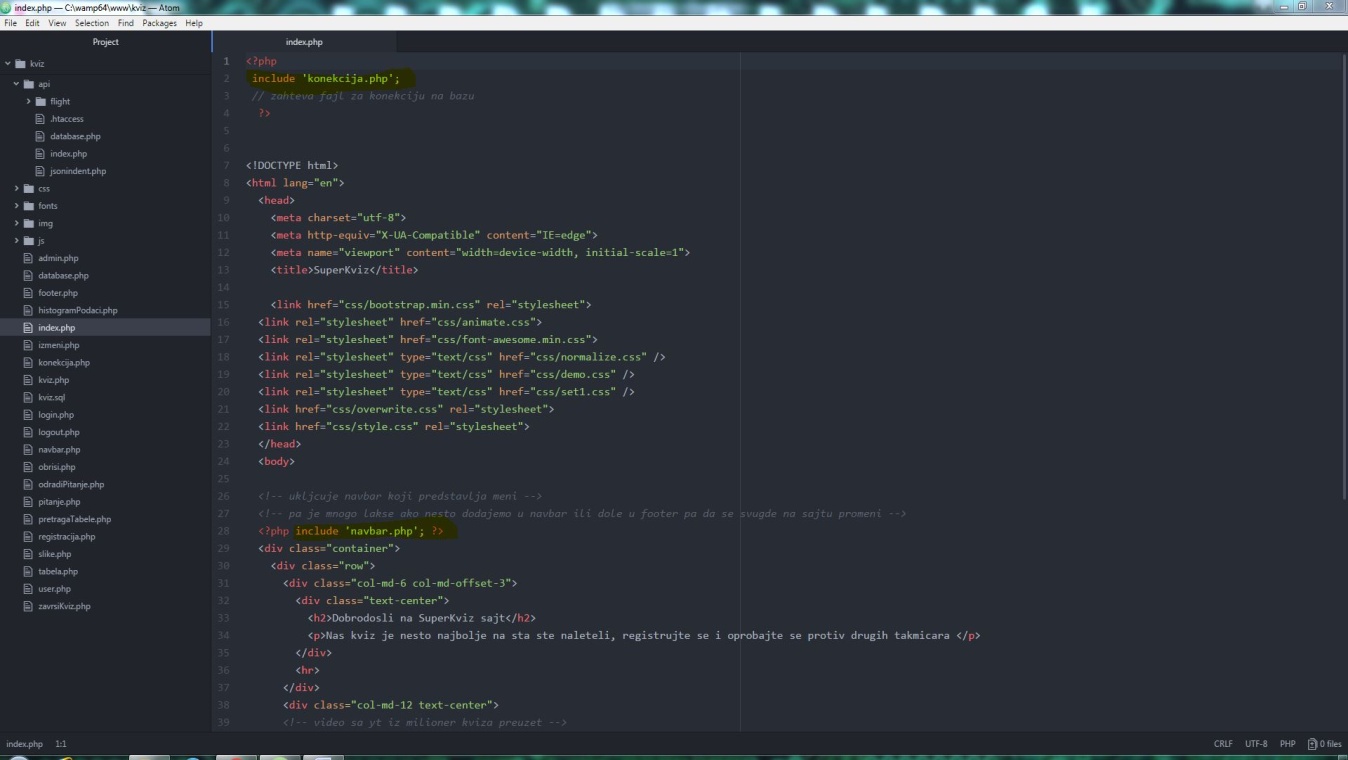
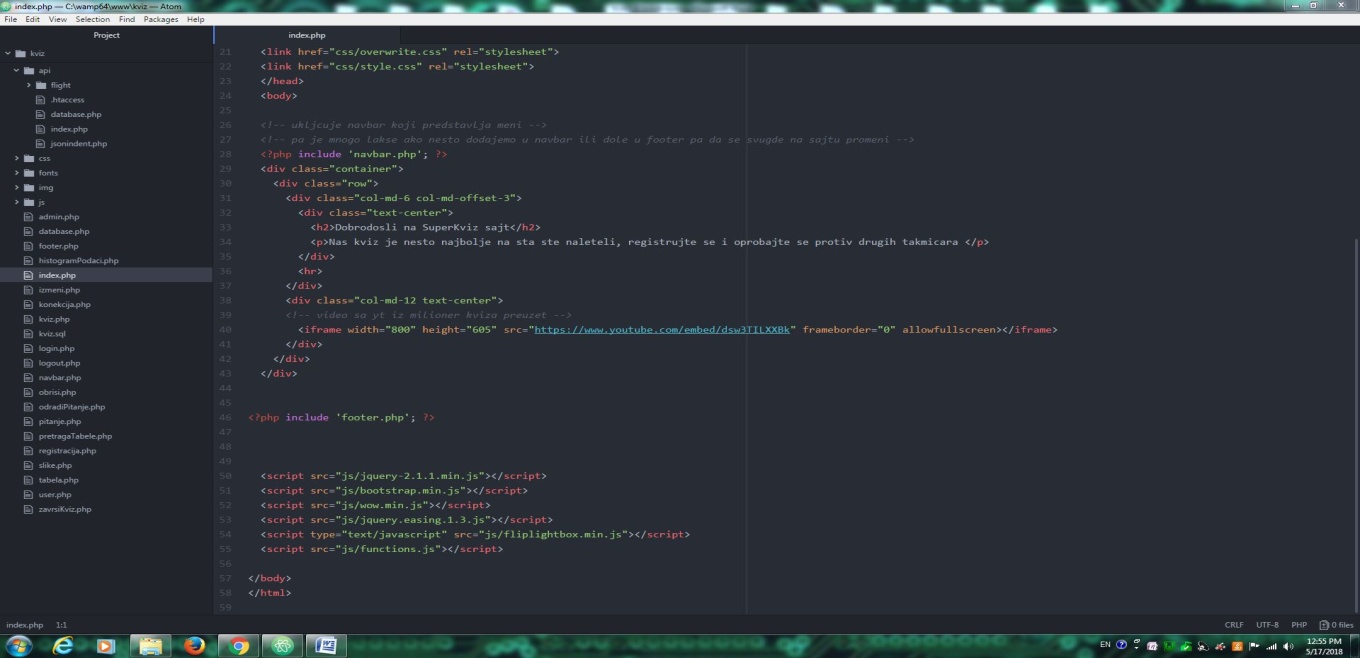
Obradjena veb aplikacija predstavlja igricu u I vidu kviza gde korisnici mogu da se registruju, zatim uloguju preko registrovanog naloga I igraju kviz. Zatim na kraju mogu da pregledaju svoje rezultate u tabeli rezultata. Korisnik koji je admin ima mogucnost i dodavanja novog pitanja kao pretrage korisnika i pitanja.

1. Stranica – index.php koja u sebi ukljucuje konekcija.php, navbar.php I footer.php . To su fajlovi koje skoro na svakoj stranici ukljucujemo pa ako se nesto menja dovoljno je jedno m u tom fajlu da se izmeni I svugde ce biti izmenjeno. Markirali smo ukljucenu konekciju I navbar. U zaglavlju imamo bootstrap biblioteku za css kao I ostale css fajlove.



Na slici ispod je prikazana druga strana index.php fajla gde se koristi <iframe> tag za prikaz video sadrzaja sa youtube servisa.

Takodje se vidi kako je uvezen **footer.php** fajl kao i javascript biblioteke I fajlovi. Koristio sam div kontejnere sa gotvovim nazivima klasa iz bootstrap.

**

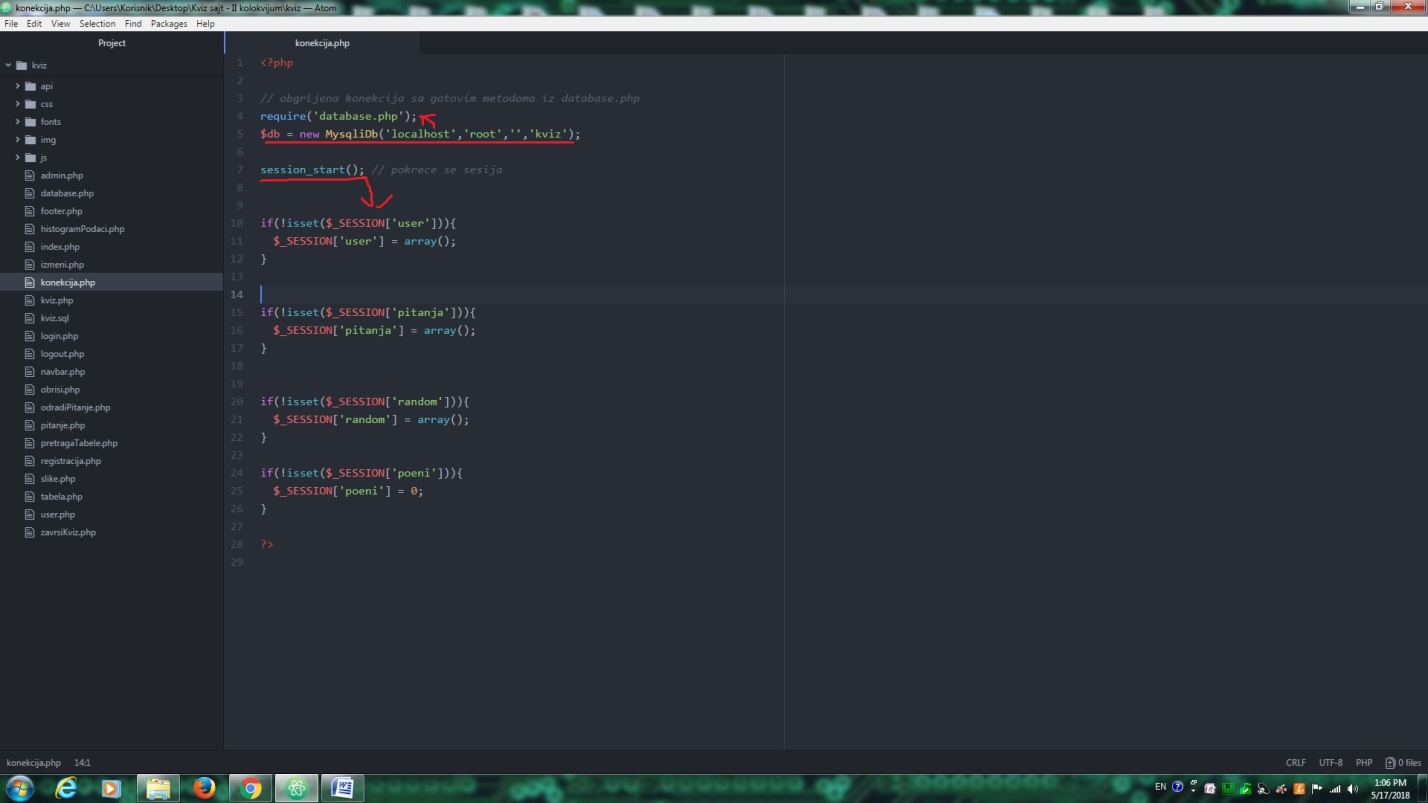
1. Konekcija.php je fajl koji je svugde ukljucen I tu se kreira konekcija na bazu preko localhost veze, odredjenim imenom, lozinkom I na odredjenu bazu. Zahtevali smo database.php fajl koji sam nasao na internetu. To je klasa za instanciranje objekata za konekciju sa bazom u konstruktoru I odredjenim gotovim funkcijama za SQL klauzule I naredbe. Nakon toga naredbom **session\_start()**  pokrecemo sesiju I ako nisu setovane vrednosti za odredjena polja koja ce se posle objasniti setuj ih za kasnije provere u logovanju I kvizu.

**$\_SESSION[‘pitanja’]** ce biti niz koji se puni sa pitanjima koja dolaze u kvizu.

**$\_SESSION[‘user’]** ce biti niz koji se puni sa podacima korisnika kada se uloguje uspesno za kviz.

**$\_SESSION[‘random’]** ce biti brojevi za ID pitanja gde ce se proveravati da se ne izvuce pitanje koje je vec bilo.

**$\_SESSION[‘poeni’]** ce biti broj poena koji je korisnik osvojio.



1. **Tabela.php** je fajl u kome su prikazani rezultati korisnika koji su odigrali kviz. Mogu se sortirati po poenima, nazivima korisnika ili redosledom izvucenih pitanja. Kodom ispod smo prvo inicijalizovali zahtev za tabelom rezultata od api.

Zatim smo setovali opciju za zahtev da vrati neke podatke kao transfer.

Izvrsavanjem zahteva dobijamo odgovor u json formatu.

Dekodovanjem toj json formata sa listom objekata smestamo ih u niz koji zovemo tabela I zatvaramo zahtev za servisom.

Preko foreach petlje za svaki element niza (element niza je jedna odigrana igra) ispisujemo za svako polje tabele Ime korisnika, broj poena I niz odradjenih pitanja.

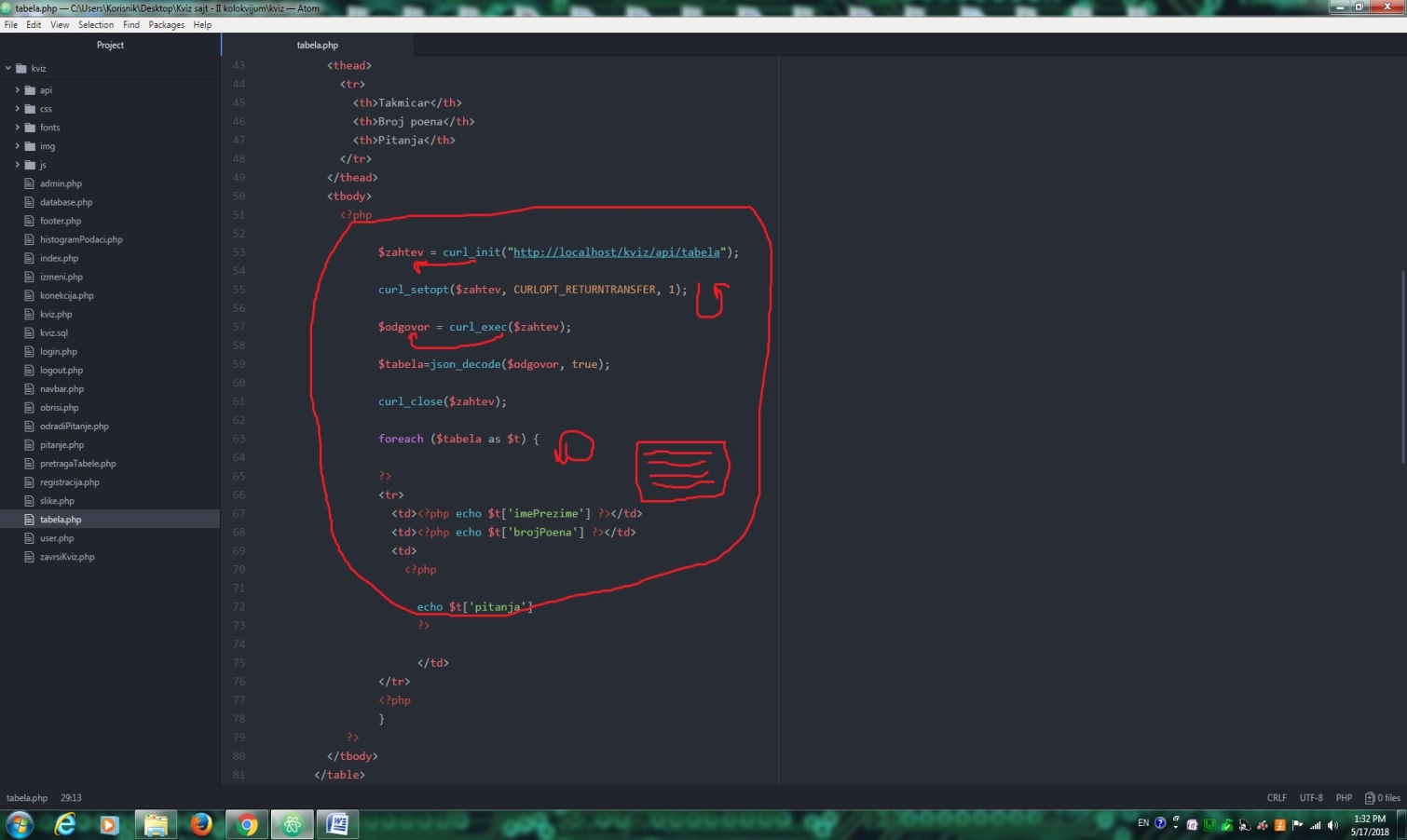
**Na drugoj slici od tabela.php** je kreiranje histograma koji prikazuje pojavljivanje odredjenih rezultata u kvizu. Uvezli smo javascript google biblioteku za rad sa podacima u tabeli I dodatnim funkcionalnostima za sortiranja po definisanim kolonama.

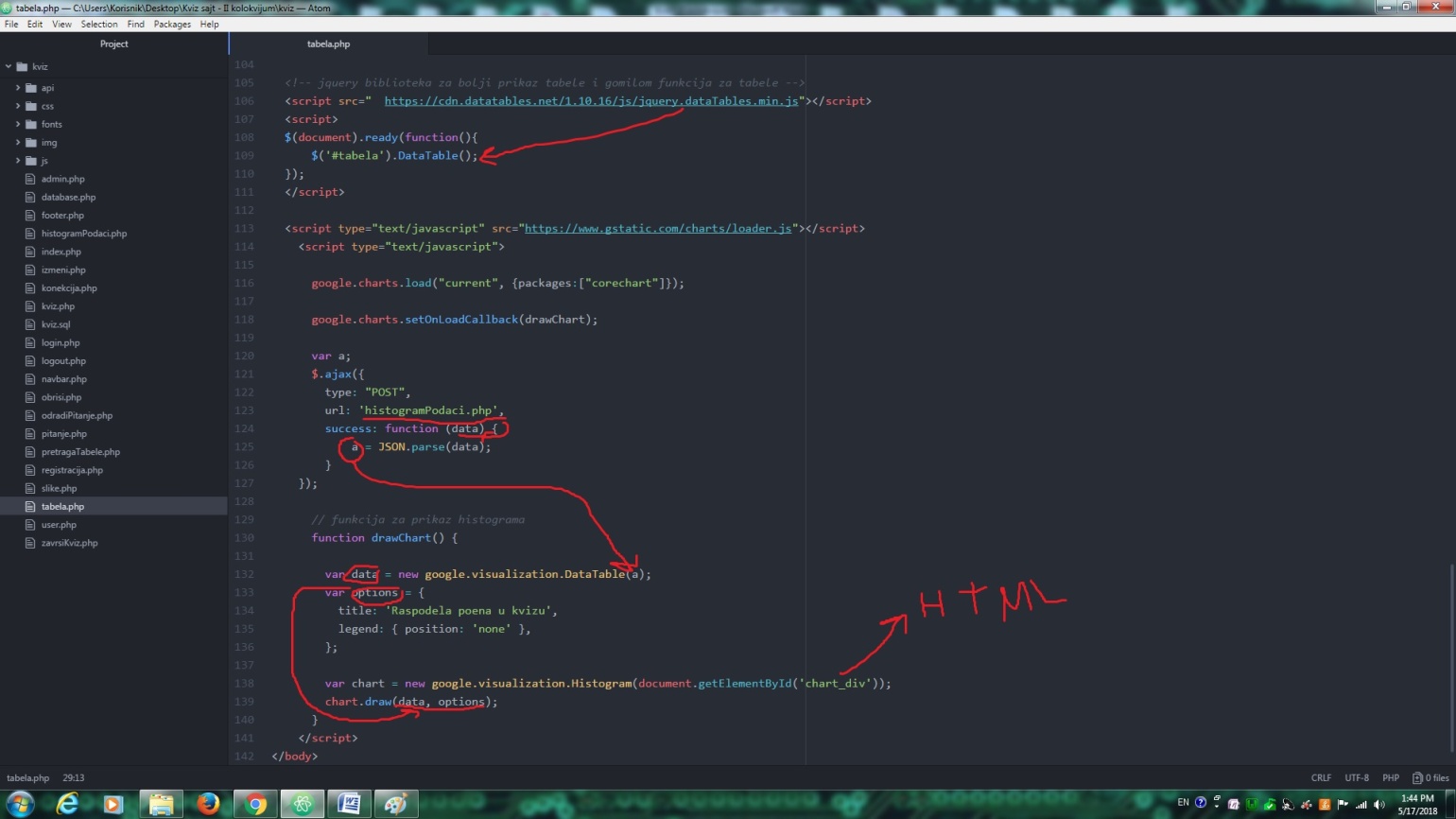
Zatim sam pozvao **google funkcije** da se pri ucitavanju stranice ucita **corechart** **histogram** I da se on iscrta pri ucitavanju stranice.

Onda se preko ajax vade podaci sa histogramPodaci.php I parsiraju iz json I objekte koji se posle u funkciji ispod u **drawChart()** pripremaju za podakte za vizuelni prikaz I bira se naslov I pozicija za legend. Zatim se uzima div blok unutar koga se se icrtati histogram.

**#chart\_div** je blok koji se napuni tim grafikonom histograma kada se ucita stranica u funkciji **setOnLoadCallBack(drawChart)**

I na kraju se funkcijom **draw()** nad chart promneljivom (sto je html div zapravo) iscrtava histogram sa prosledjenim podacima I opcijama.

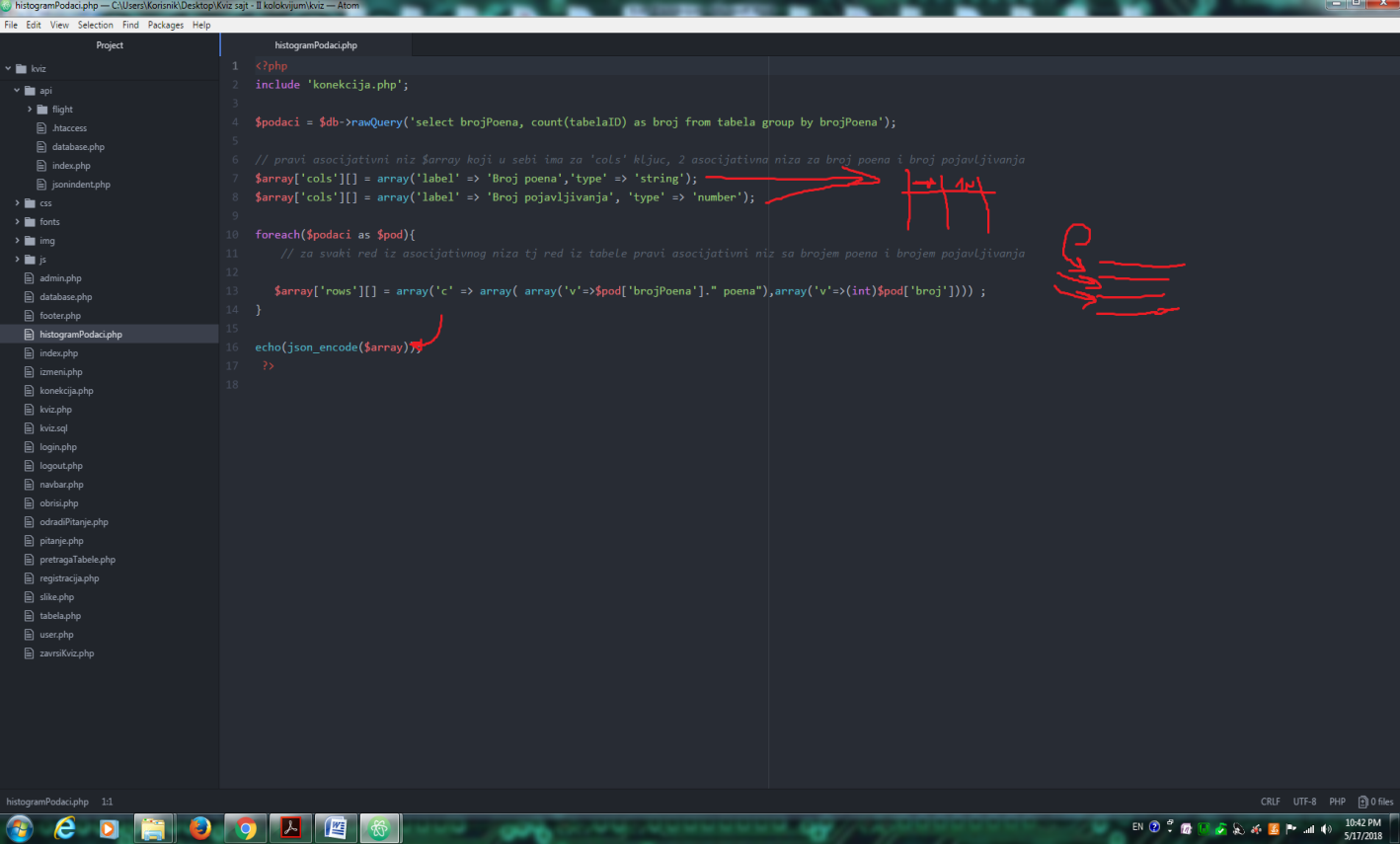


****

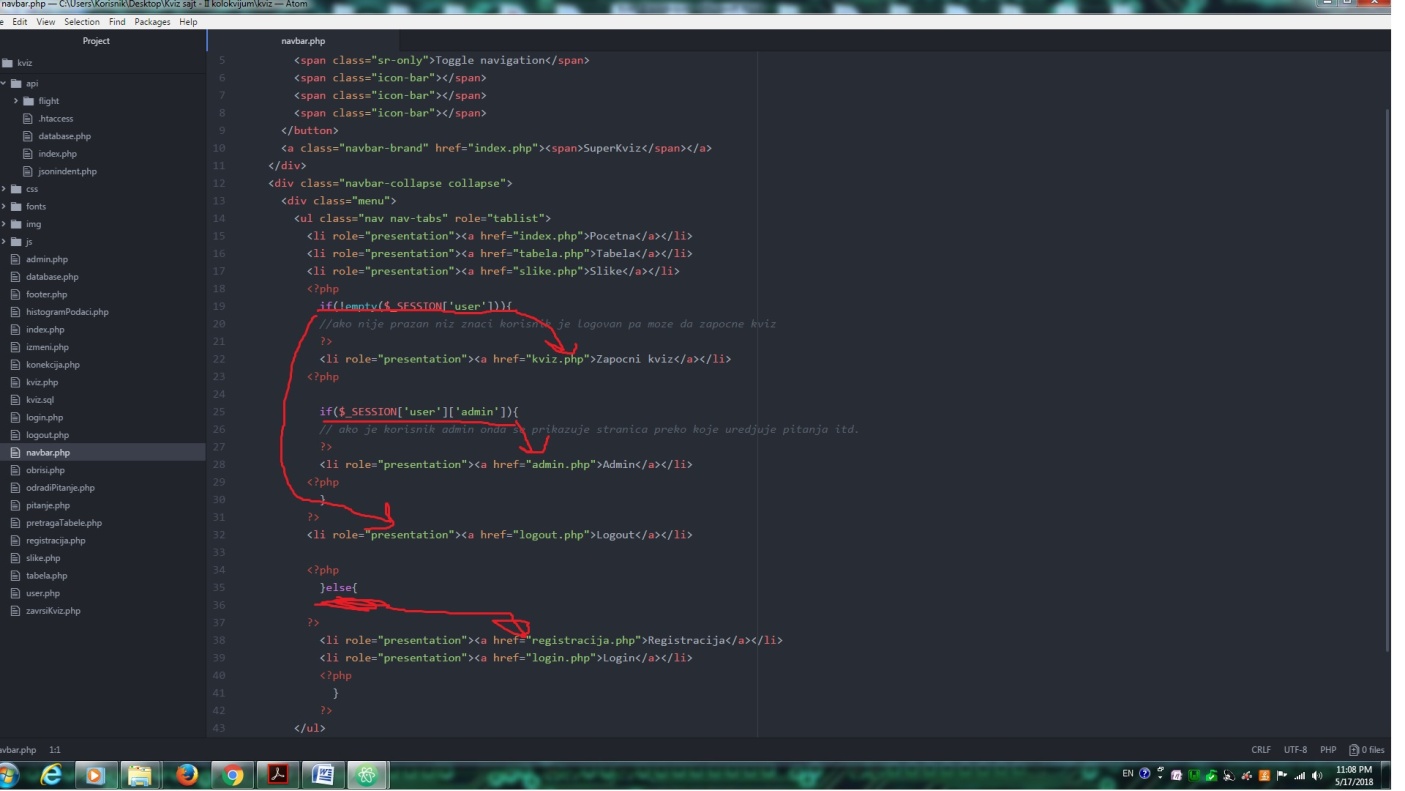
1. **histogramPodaci.php** je skripta koja iz baze izlvaci podatke za histogram I salje ih u json format preko ajax.

Prvo se skrpita poveze sa bazom pa se preko upita funkcije rawQuery() vade podaci o brojem rezultata sa istim brojem poena.

U $array asocijativni niz pravimo 2 zaglavlja za kolone Broj poena I Broj pojavljivanja, a ispod toga foreach petljom za svaki zapis iz upita unosimo u svaki red niza kao trodimenzionalni niz I na kraju pakujemo u JSON format koji ce se proslediti preko ajax nazad u **tabela.php** fajl nakon cega ce google funkcija vizualizacije da kreira histogram grafikon na osnovu ovih podataka.



1. Treba napomeniti samo da u **navbar.php** imamo php deo koda tj u meniju na sajtu koji prikazuje stranice za registrovanje i logovanje ako nismo ulgovani. Zatim ako smo ulogovani kao obican korisnik prikazuje stranicu za kviz, a ako smo ulogovani kao admin prikazuje dodatnu opciju za editovanje pitanja na admin stranici.



1. **Slike.php** je stranica koja kupi slike sa javnog Flickr apija ciji se url po odredjenim ocpijama prosledjuje preko AJAX I pakuju u JSON format.

Zatim se ti podaci preko each petlje za svaki item u nizu data.items pravi tag za sliku sa adresom koja se nalazi u svakom itemu pod media pojlem pa u njemu pod m poljem i nadodajemo u blok sa slikama. To se moze I proveriti kada se kao adresa unese [http://api.flickr.com/services/feeds/photos\_public.gne?**tags=qiuz&tagmode=any&format=json**](http://api.flickr.com/services/feeds/photos_public.gne?tags=qiuz&tagmode=any&format=json) pa se nadje items pa unutar njega media pa m.

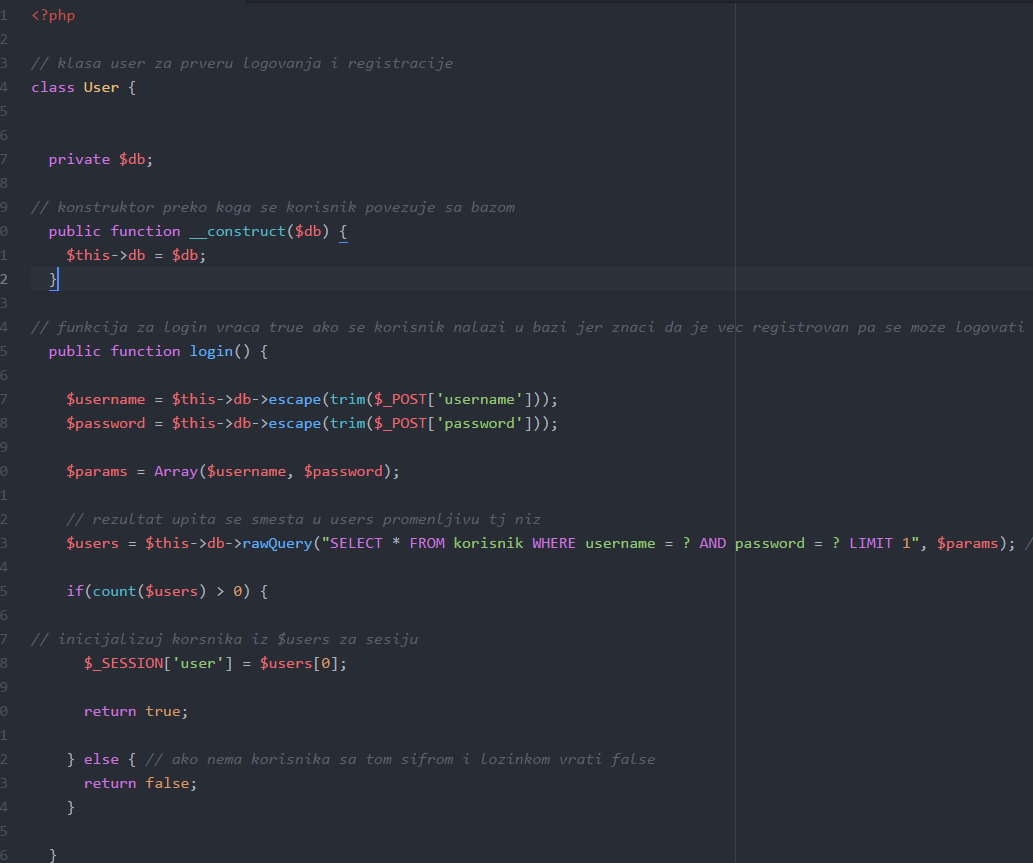


1. **Login.php** sadrzi html formu (*desna slika*) po Bootstrap strukturi I prihvata podatke za username I password (autentifikacija) koji se prosledjuju php kodu (*leva slika*). Uvezena je skripta sa konekcijom na bazu kako bi smo proverili da li je korisnik u bazi. Za to nam takodje treba **user.php** skripta koja u sebi sadrzi definiciju klase za korisnika I funkcije za validaciju registracije I logovanja. Funkcija **login()** iz klase user vraca Boolean vrednost nakon koje se kreira adekvatna poruka u zavisnosti od vrednosti promenljive **$sacuvano**. I ta poruka se ispisuje na pocetku html forme. **Takodje ako je vracena true vrednost kreira se fajl ‘sesija.txt’ u modu za upis i kreira novi** ako je prethodno vec postojao sa istim imenom. Upisuje se poruka da je korisnik ulogovan i ispod iz POST polja username I password korisnika I na kraju se zatvara fajl. **Naredbe \r\n** znace prelazak u novi red radi bolje preglednosti sadrzaja fajla.

****

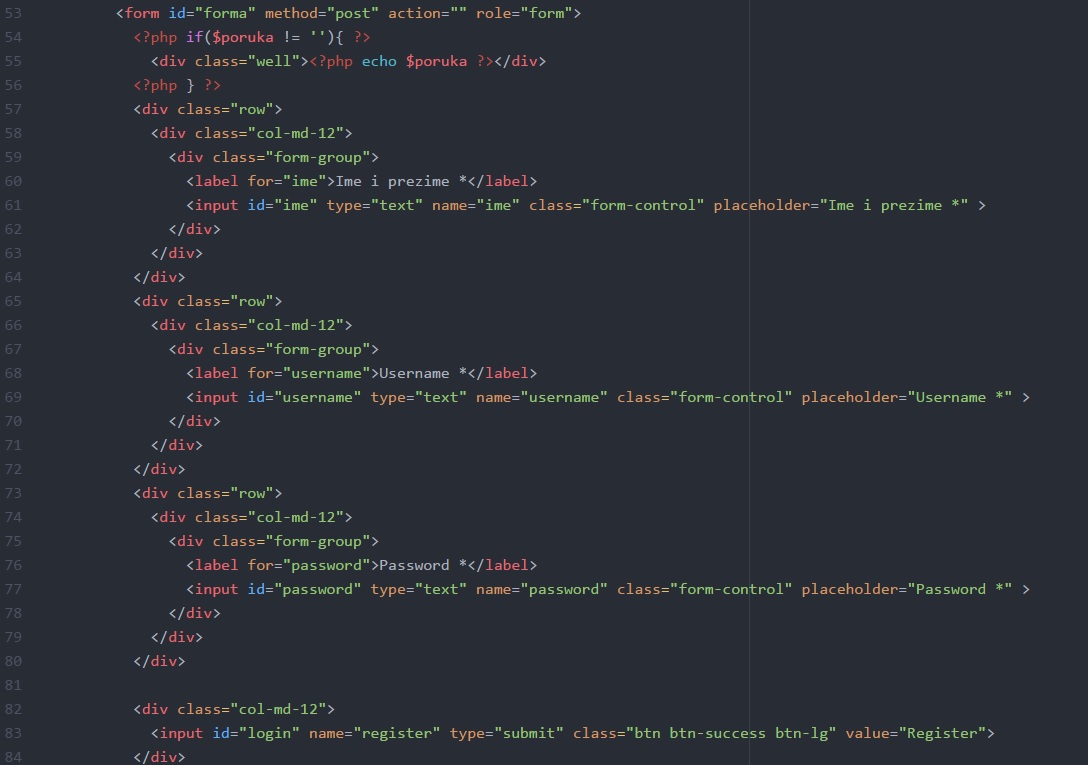
1. **User.php** je klasa koja se koristi na stranicama za registraciju I logovanje. Konstruktor uzima polje kojim se instancira konekcija na bazu. Prva funkcija je za logovanje I u njoj se prvo username I password ciste na krajevima stringa I funkcijom **escape(definisana u database.php omotac klasi)** uklanjaju potencijalni stetni karakteri **povodom SQL Injection** napada. Zatim se ti sredjeni podaci prosledjuju promenljivama koje se cuvaju u niz **$params**. Pa se preko rawQuery**(definisana u database.php omotac klasi)** u kome su bind-ovani podaci vadi taj korisnik. Pa se proverom duzine niza da je >0 inicijalizuje ‘user’ vrednost u sesiji I vraca true. U suprotnom se vraca false. U drugoj funkciji se radi isto sa podacima u pocetku samo sto se oni nakon ciscenja pakuju u asocijativni niz I koduju u JSON format.

Zatim se preko zahtev koji se nalazi u api koji vraca **Flight::route** za noviKorisnik (**definisano u api/index.php**) metod inicijalizuje zahtev za dodavanje korsinika. Setuju se opcije da se iz POST nizu salju podaci I da se vrati transfer kad se izvrsi zahtev. Nakon toga se odgovor dekoduje iz json u odgovor - **$json\_objekat** (**ima vrednsot OK ili GRESKA -** **definisano u api/index.php**)na osnovu kojeg se vraca boolean kao rezultat funkcije.



1. **Registracija.php** je slicno kao login samo je dodato da se unese ime I prezime I password je tekstualnog tipa karaktera kako bi korisnik video sta je uneo pre nego sto se registruje. Podaci kad se unesu I kad se submituje forma preko POST metoda se podaci salju gore da se obrade u php kod.

Zatim u php kod opet ukljucujemo konekciju pa proveravamo da li je setovano polje za **register** tj submit odradjen. Ako jeste onda se uvozi klasa User I kreira objekat kome se prosledjuje instance konekcije nad bazom I pozvima metod **registracija()** koji smo gore objasnili i koji vraca true ili false pa na osnovu toga kreiramo poruku koja ce se prikazati na pocetku forme u html.

****

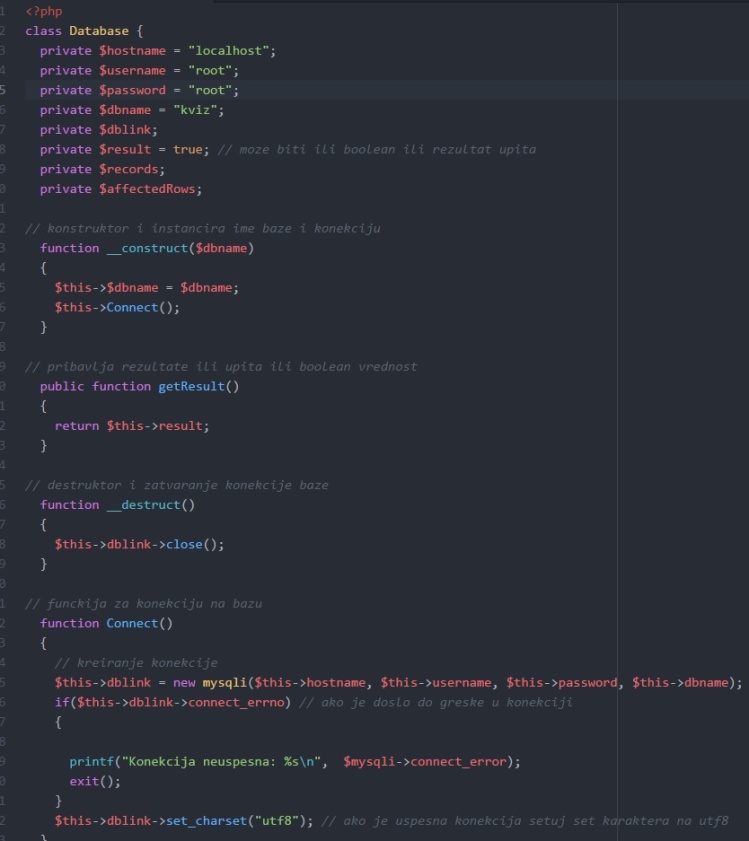
1. Prelazimo u api folder gde imamo index.php u kome se nalaze sve rute – Flight::route za api zahteve koji ce se obradjivati zajedno sa fajlom **database.php u tom api folderu** **(to nije database.php u korenu projekta**). Zahtevamo klasu **Flight.php** **I jsonIndent.php** koji je skroz preuzet sa internet I sluzi za lepsi prikaz u JSON format. Registrujemo bazu I pravimo rute za GET I POST metode transfera( **kod ovih funkcija je samo GET metoda**). Definisemo zaglavlje da je json I utf8 I uzimamo registrovanu bazu koja je registrovana na pocetku fajla. Pozivamo u sve 3 funkcije odredjene metode iz **database.php** klase koji vracaju sve podatke iz neke tabele. Zatim se u svakoj od tih funkcija while petljom red po red koji se kupi zi rezultata upita ubacuje u element niza I na kraju koduje u json da se salje preko **curl** funkcija.

****

**Jedino je ruta za funkciju dodavanja novog korisnika preko POST metode.** Sve je isto sto se tice pocetka fukcije samo se preko funkcije file\_get\_contents() kupe podaci iz input polja u json format koji se dekoduju u niz **$json\_data**. Prosledjujemo funkciji za dodavanje novog korisnika te json podatke. Ako se upit uspesno izvrsio vrati OK, ako nije vrati Greska! . Na kraju ispisemo u json format odgovor preko **indent()** funkcije za lepsi prikaz (jedni ispod drugih objekti)

****

1. **U database.php fajlu** smo definisali klasu koja kreira svoju konekciju preko prosledjenih parametara **(hostname, username, password, dbname).** Definisali smo polja za konekciju na pocetku klase a ispod smo u metodu za konekciju instancirali konekciju pa proverili ako je doslo do greske u povezivanju sa bazom da izbaci poruku preko **printf** (za formatiranje ispis) I pozove I prikaze **error** zatim izadje. Ako nije doslo do greske setuj utf8 set karaktera. Zatim smo ispod implementirali funkcije koje kreiraju I izvrsavaju **raw** upite za izvlacenje svih pitanja, korisnika I rezultata (**u upitu za rezultate kviza se koristi prirosno spajanje 2 tabele I vrsi sortiranje u opadajucem poretku)**. Jedino je drugacija funkcija za dodavanje novog korisnika jer se prvo ciste podaci za bazu preko escape I onda se preko INSERT naredbe I izvrsavanjem upita se rezultat setuje na true ili false u zavisnosti od toga da li se upit dobro izvrsio, dok kod drugih funkcija se rezultat setuje da bude ceo niz vracen iz rezultata upita koji bi posle koristili. Ispod na dnu se nalazi core funkcija koju sam preuzeo sa neta za izvrsavanje upita

****

1. **Kviz.php** fajl sadrzi blok u kome ce se ispisati izvuceno pitanje, **#loadbar** koji se nakon ucitavanja novog pitanja pravi animaciju prikazivanja. Prvo se izvlaci pitanje u php kodu. Proveravamo da li je slucajno broj pitanja dosao do 5. Ako jeste idi na kraj kviza preko **header funkcije ucitavnja zavrsiKviz.php.** Inace se izvlace pitanja preko get funkcije u database.php koja izvlaci I smesta u niz sva pitanja. Zatim se preko **rand** funkcije izlvaci nasumican broj od 0 do brojPitanja-1.

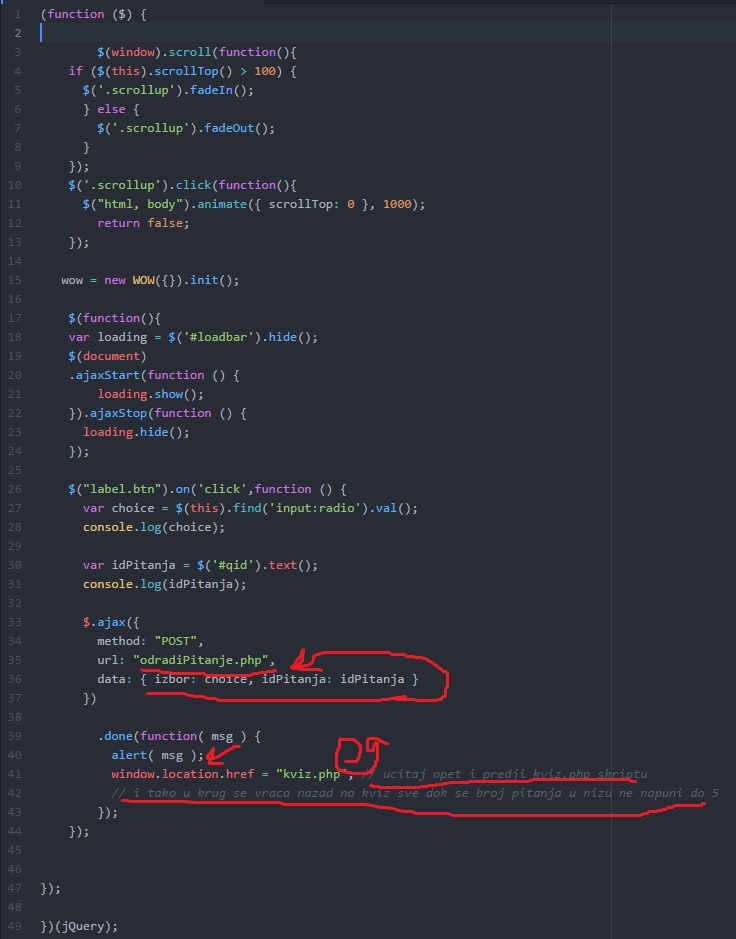
Pa se preko uslova petlje gde se proverava da li je taj izvucen broj u nizu random izvucenih brojeva (ID pitanja koja su vec bila). Ako jeste ponovo se izlvaci broj za ID pitanja. Na kraju se izvuce pitanje iz niza sa indeksom tog broja I taj broj se smesti u niz random izvucenih brojeva.



Prikaze se u h3 tagu izvuceno pitanje I dole su kao labele dati ponudjeni odgovori koji sadrze u sebi input polja u vidu radio button sa **value {A, B, C ,D}** kao ponudjeni odgovori I kada se klikne na njih taj izbor se salje u **functions.js** skriptu koja izmedju ostalog obradjuje izbor pa kad obradi vraca se nazad na **kviz.php** za novo pitanje. Takodje tu se nalazi I jQuery funkcija za obradu **loadbar** kojim se pri ucitavanju pravi animacija za blok pitanja.

****

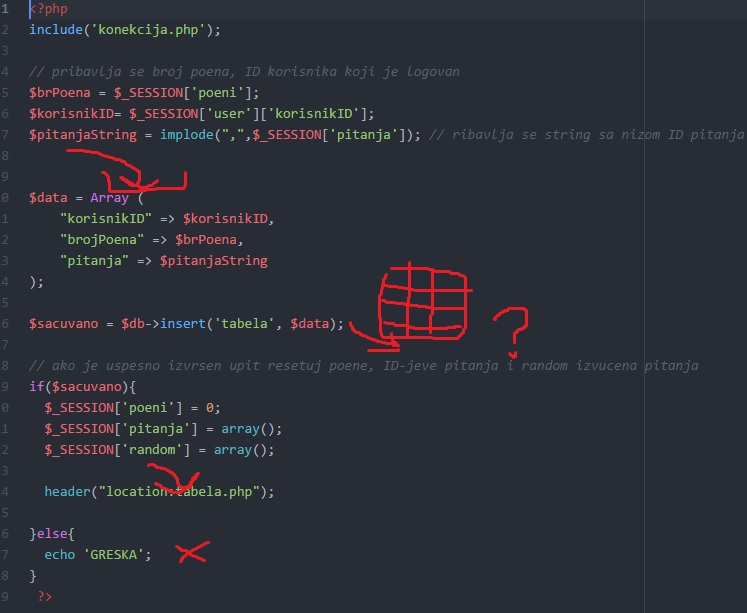
1. **Functions.js** skripta koja obradjuje scrollup u footer.php koji nas vraca a vrh strane kad god odemo na dno preko provere za **scrollTop**, obradjuje animaciju za loader tako sto prikazuje I skriva **loadbar** kod ucitavanja svakog pitanja. Funkcija koja obradjuje izbor registruje klik na tag labela i izvlaci izbor preko input polja tipa radio button I izvlaci ID pitanja preko #qid sto je id html taga za ID pitanja pa se onda preko **ajax POST metodom salju podaci na obradu u obradiPitanje.php.** I na kraju se kad se obradi pitanje prikaze vracena poruka iz php skripte koja obradjuje pitanje I prikazuje se da li je odgovor tacan ili ne I broj poena. Pa se opet ucitava **kviz.php** I tako u krug dok se broj pitanja ne poveca do 5 sto je ogranicenje za kraj kviza.



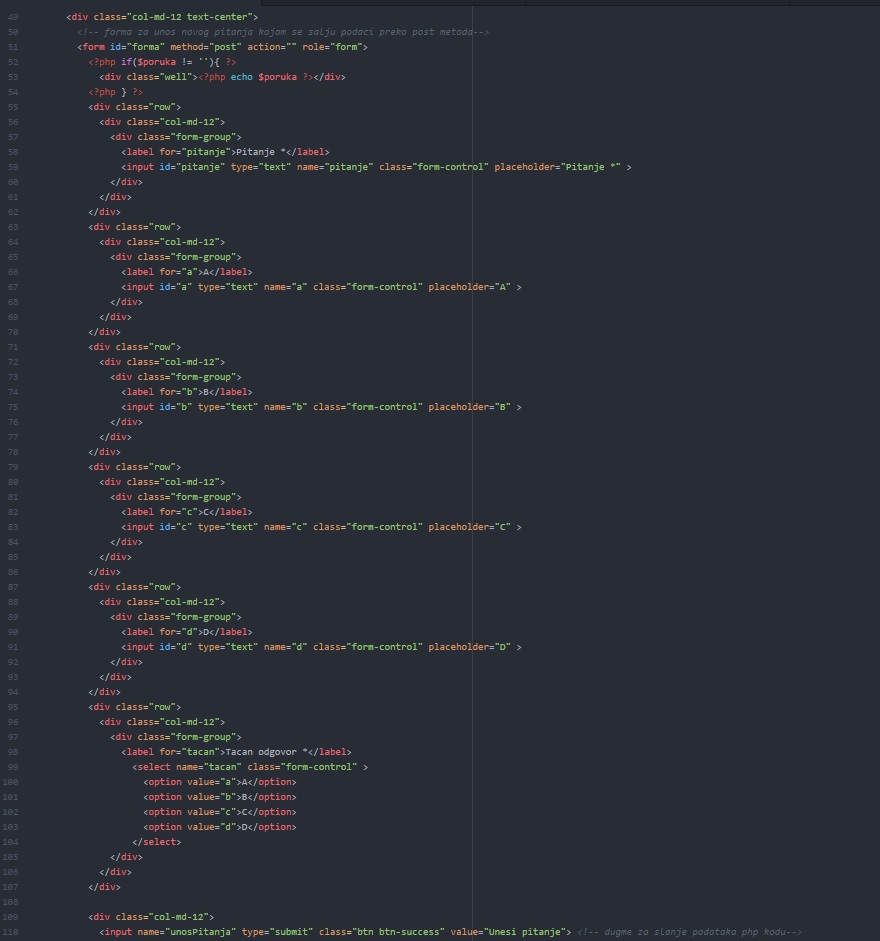
1. **odradiPitanje.php** je skripta koja ukljucuje konekciju na bazu samim tim I omotac klasu sa predefinisanim funkcijama nad bazom. Podaci za izbor i ID pitanja koja smo dobili preko **ajax** cuvamo I zatim izvrsimo upit sa where naredbom da nadje to jedno pitanje sa tim ID I sacuvamo ga kad nam vrati preko **getOne()** fukncije za tu **tabelu ‘pitanje’** . **Ubacimo u niz pitanja u sesiji taj ID pitanja**. Zatim proveravamo da li je nas **izbor jednak onom sto je u polju pod ‘tacan’ a to su vrednosti {a,b,c,d}** pa ako jeste inkrementiraj broj poena I setuj poruku. Ako nije tacno samo setuj negativnu poruku I na kraju ispisi poruku koja ce se vratiti kao nazad u **functions.js** za **alert**.

****

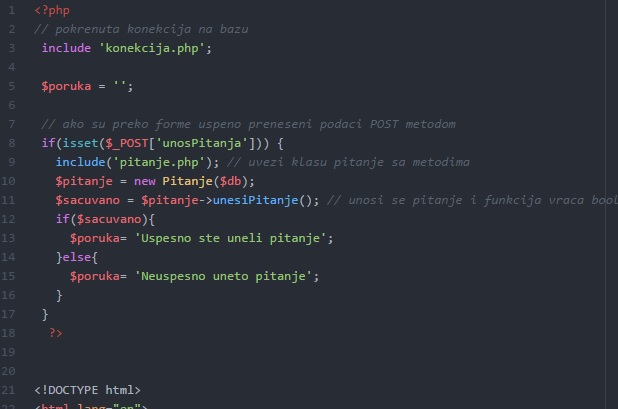
1. **zavrsiKviz.php** je skripta koja ucitava konekciju za rad sa bazom. Cuva u promenljive broj poena, ID korisnika I niz pitanja. Niz pitanja pretvara u string funkcijom **implode()** I umetanjem zareza izmedju svakog ID pitanja. Posle sve pakuje u asocijativni niz I preko predefinisanog upita insertuje podatke nad tabelom ‘tabela’ koja vraca na kraju Boolean u zavisnosti od toga da li su uspesno uneti podaci. Pa ako je uspesno sacuvano **resetuju se vrednosti niza SESIJE** I ucitava se tabela.php stranica sa prikazom rezultata I histograma. Ako nije ispravno uneto izbacuje se poruka o gresci.



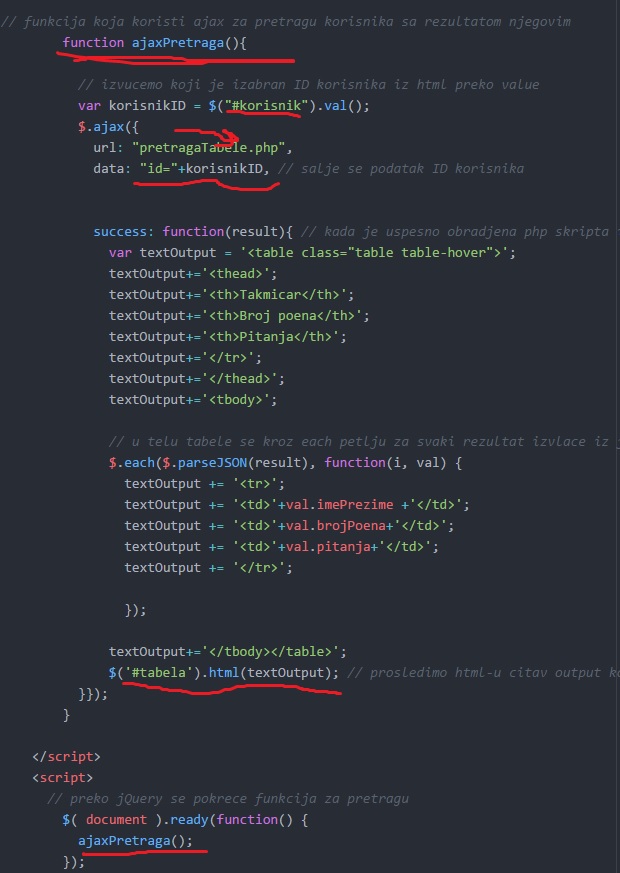
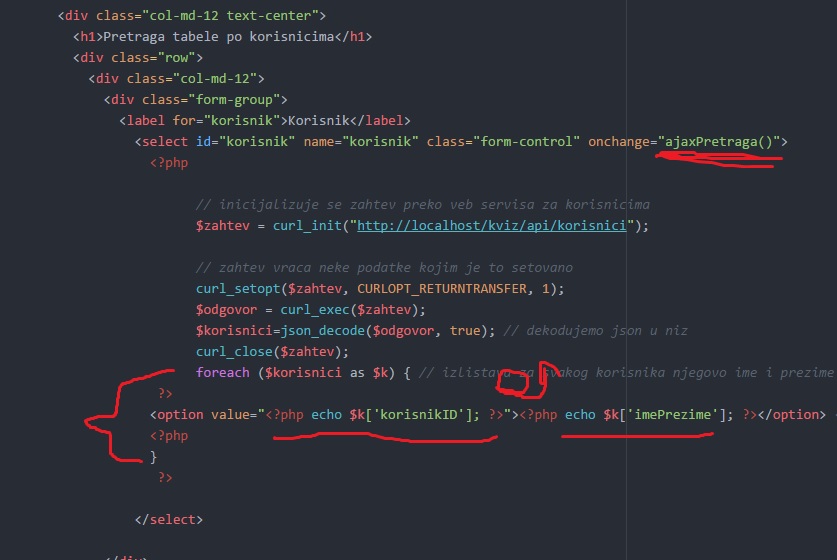
1. **admin.php** je skripta na kojoj admin ima pregled svih pitanja kao I da dodaje, menja ili obrise pitanje. Gore pri vrhu na liniji 52 ce nam se ispisati statusna poruka da li je ispravno uneto pitanje. Zatim imamo input polja za unos pitanja, ponudjene I tacan odgovor kao select meni sa opcijama {a,b,c,d} I na kraju dugme za slanje preko POST metoda skripti na pocetku fajla za obradu.



Kod koji obradjuje uneseno pitanje ukljucuje konekciju na bazu I proverava da li je setovana vrednost za submit **u POST nizu**. Ako jeste ukljucuje fajl **pitanje.php** koji u sebi sadrzi **klasu Pitanje** sa funkcijama za unos pitanja I izmenu pitanja koju cemo posle analizirati. Funkcija za unos vraca true ili false u zavisnosti od toga da li je uspesno uneto pitanje setuje vrednost poruke koja se prikazuje pri vrhu gore prikazane forme.



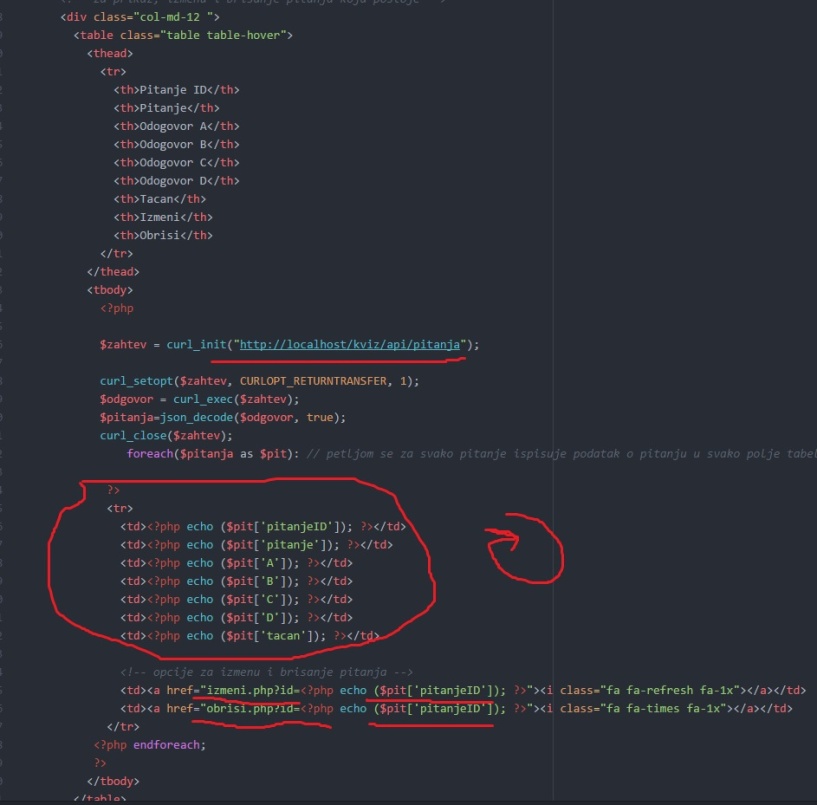
U ovom bloku ima pretragu po korisnicima u select tagu koja poziva js funkciju ajaxPretraga() za ostvarene rezultate tog korisnika I poziva na svaku promenu ispocetka. Prvo se u php kodu **inicijalizuje zahtev preko Flight ruta** I setuje se da vrati transfer podataka kada se izvrsi zahtev I vraca ga u json format kao sve korisnike iz tabele koji se dekoduju u niz. Zatim se **foreach** petljom redom u svakom option tagu prikazuju korisnici iz niza (ID korisnika I ime I Prezime). Zatim se poziva funkcija na slici desno koja izvlaci iz **value od option taga ID korisnika** zatim se pokrece ajax koji salje kao podatak ID korisnika u **pretragaTabele.php** koja vraca rezultate za odredjenog korisnika. Zatim se nakon uspesno izvrsene skripte vraca rezultat I konkatenacijom se kreira sktruktura html tabele sa nazivima kolona u zaglavlju I kroz each petlju se izvlace podaci iz JSON u objekat I prikazuju u poljima vrseci konkatenaciju na output-a I na kraju se taj output smesti u div blok **#tabela** ispod bloka sa korisnicima u html.



1. **pretragaTabele.php** ukljucuje konekciju na bazu I uzima id prosledjen preko ajax zatim ako je id razlicit od nule **kreira where uslov** za mecovanje sa dobijenim ID I na kraju izvrsi upit **prirodnim spajanjem** tabela ‘**pitanje’** I **‘korisnik’** sa zadatim where uslovom koji ce vratit kao rezultat sve podatke o rezultatima korisnika u kvizovima koje je odigrao I te podatke **na kraju kodovati u json**.



Ovde smo napravili tabelu u html u kojoj ce se prikazati sva pitanja sa opcijama sa strane da se **izmene ili obrise**. Inicijalizujemo zahtev preko ruta za sva pitanja iz tabele I setujemo opciju za transfer da vrati te podatke kad se izvrsi zahtev. Odgovor dobijen dekodiramo iz json u niz I onda foreach petljom prikazujemo sva polja za svako pitanje iz tabele u jednom redu I **dodatna polja kao ikonice za izmenu I brisanje koje pokrecu php skripte I salju preko GET metoda ID pitanja**.



1. **obrisi.php** uvozi konekciju I proverava da li **je setovan ID pitanja preko GET metoda**, pa ako nije vraca ga na admin. Inace cuva taj id I prosledjuje ga **where** funkciji objektu baze da nadje to pitanje koje ce smestiti u rezultat I onda samo izvrsi delete upit sa id tog pitanja iz tabele I ucita opet admin stranicu.

****

1. **izmeni.php** skripta ima deo forme koji ce prvo ucitati vrednosti u poljima koja su trenutno u bazi. Zatim admin unosi nove vrednosti za sva polja I na kraju klikne na submit gde se pokrece php kod na levoj slici. Tu je ukljucena konekcija **pa proverava da li je setovan ID pitanja preko GET metoda**, ako nije vrati ga na admin stranicu. **Inace sacuvaj ID pitanja I preko where** klauzule pronadji to pitanje I pribavi ga preko **getOne** funkcije. Ako je setovano polje za izmenu pitanja tj submitovana forma onda ukljuci **klasu Pitanje iz pitanje.php** gde ce se kreirati objekat pitanje I nad kojim ce se pokrenuti funkcija za izmenu pitanja koja vraca true ili false pa u zavisnosti od toga setuj adekvatnu poruku koja ce se prikazati na pocetku forme.

****

1. **pitanje.php** je skripta u kojoj je definisana klasa za pitanje sa funkcijama za unos novog pitanja I izmenu pitanja**. Imamo definisano polje koje hvata konekciju na bazu I koje se setuje preko konstruktora**. Zatim imamo **funkciju unesiPitanje** koje preko POST metoda pribavlja sva polja prosledjena preko inputa I cisti string od praznina I stetnih karaktera po bazu pa cuva u promenljive koje pakuje u asocijativni niz nakon cega **izvrsava insert upit nad tabelom ‘pitanje’ I bind-uje odredjene podatke** u upit koji vraca true ili false nakon cega I funkcija vraca true ili false sto ce sluziti posle za statuse poruke. Zatim ispod nje imamo funkciju za izmenu pitanja koja na isti nacin filtrira podatke koji su uneseni preko inputa I smesta ih u obican niz koji se **bind-uju rawQuery funkciji u update upitu koji vraca Boolean u cijoj zavisnosti I cela funkcija za izmenu pitanja vraca true ili false.**



1. **logout.php** je skrpita koja ukljucuje konekciju gde **je pokrenuta sesija pa je zatim unistava. Zatim nadodaje na kraj fajla sesija.txt da je korisnik izlogovan**, pa zatvara fajl I ucitava index tj pocetnu stranicu sajta.

