# Upute za izradu P3 zadatka

## 1. Ontologija

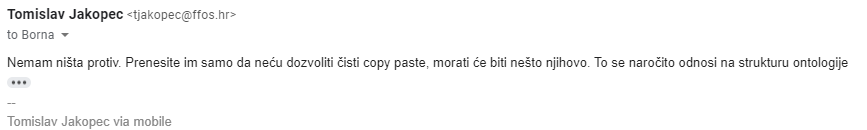
## 2. API

## 3. Android app

Prvo ću napomenuti da uz ovo koristite i Aninu/Katarininu skriptu s kolokvija, pogotovo za Android jer je gotovo isto.

# Ontologija

Ontologija se radi u Protegeu ili WebProtegeu (ne mora se instalirati na komp ali drukcije izgleda od Protegea kojeg smo koristili kod Bose). Kad sam mu rekao da cu ostavljat komentare po aplikaciji za kolege rekao je



Ako ga netko želi pitati jel ok koristiti skupne ontologije jer bi onda po dvoje/troje imali iste ontologije, pitajte ga. Imamo mislim i pojedinačne ontologije u nekom starom kolegiju ako nas Boso nije izbacio iz njih. Ako morate pravit nove, moja sućut, ne gine neki indijski tutorijal. Ja jesam pravio novu ontologiju, ali samo zato što tada nisam još znao kako riješiti taj dio zadatka, tako da ne znam jesu bosine ontologije koristljive ili ne, trebale biti.

Ontologiju potom, neovisno o formatu, kopirate cijeli tekst i pasteate ovdje ( u Input data): <https://www.easyrdf.org/converter> . Input format je Guess, Output format RDF/XML. Rezultat kopirajte cijelu u novu praznu datoteku i nazovite ga kako hoćete (s završetkom .rdf). Stavite je na vaš OZIZ preko filezille i onda je probajte pronaći preko web preglednika. Ako vam učitavanjem linka downloada isti taj rdf, to je dobro. Taj link onda stavite u index tamo gdje je meni $foaf=\EasyRdf\Graph::newAndLoad(**'http://bornaungar.me/ungar.rdf'**);

Tu je gotova sama ontologija.

# API

## Lokalno

Nije potrebno, ali je naporno isprobavati na serveru jer svaki put morate uploadati datoteke, najbolje prvo lokalno raditi dok API ne bude gotov i onda samo uploadati na server. Upaliti XAMPP. Ako koristite moj projekt kao template ne morate ništa drugo dodavati, moj vendor folder već ima sve instalirano (ne morate composer install). Napravite bazu u beaveru il štoveć koristite i nazovite je klasično prezimeime\_19.

U bootstrapu sam ostavio zakomentirano user i pass za lokalno, otkomentirajte to (i preimenujte u svoje) i zakomentirajte serverski user i pass (odmah mozete i to preimenovati). Kako će se tablica zvati i koje će stupce imati ovisi o vama i vasoj ontologiji. Npr. ako je ontologija o knjigama, stavite knjigu, autor, godina i tak. Ako imate datume u ontologiji moja sućut, ako je netko uspješno ubacio datume u android neka podijeli kod u grupi.

Kad smislite tablicu, klasu Ontologija.php prilagodite stupcima iz baze (npr. $knjiga, $autor...). Folder Ungar promijenit u svoje ime. Svaka varijabla ima one gluposti iznad sebe. Ako neki stupac u bazi ne koristi tekst već npr. brojeve, onda iznad varijabli neće pisati „string“ već integer.

Kad je gotova klasa i bootstrap (composer je moj riješen), ostaje index.

Zadatak traži da ruta /search (Flight::route('GET /search', function()) ispisuje json i tu zapravo ne mijenjate ništa osim klasu getRepository('Ungar\Ontologija'); u svoje ime. Taj search znači da kad upišem u preglednik https://oziz.ffos.hr/nastava20192020/bungar\_19/ontologija**/search**, taj dio koda će se izvršit. Isto vrijedi za ostale dijelove indexa, svako će odraditi svoj dio kad se učita ta ruta u preglednik.

Flight::route('GET /fill\_table', function() stvara bazu (promijenite fill\_table). Ako je lokalno, napunit će bazu u beaveru, ako je na serveru, napunit će bazu koju gledate preko puttya. Ono što će te se morati sami snaći je kako izvući pojedine stvari iz ontologije jer će svaka ontologija imati drukčije podatke. Ono zajedničko je vjerojatno način na koji prolaziti kroz svaki predmet iz ontologije, a to je s petljom foreach ($foaf->resources() as $resource) {}. Unutar vitičastih pronalazite sve ostalo. Unutar te petlje, petlja će početi s prvim resursom i proći kroz ostatak koda, pa drugi resurs itd. Ako je nešto što ima svaki resurs, npr ime, izvučete podatak s $name = $foaf->get($resource, 'foaf:name'); (**ALI** ne mora nužno biti **foaf:name** kao moja, to pripazite i prepišite kako piše u vašoj ontologiji jer vam inače neće pronaći). Ako je neki podatak koji ima više stvari, na ovaj način će samo izvući prvu stvar, da bi sve izvukli treba varijablu pozvat kao niz ( npr ako ima više imena, bit će prvo $name[] = []; pa petlja u petlji, vidite kako sam ja morao za types i biti će jasno ). Dalje od toga ne mogu pomoći, ostavio sam par korisnih kodova na dnu indexa, tipa ako su vam svi podaci kao ružni URLovi, da izvučete ime na kraju urla. Na kraju te rute mijenjate sve moje varijable i dodajete vaše.

Ako je ikoga briga, $em->persist($ontologija); $em->flush(); su dio koju unose u bazu tako da sve što radite s varijablama mora biti iznad njih. Ispod njih može ići samo nevezane stvari, tipa echo. Predlažem da zakomentirate persist i flush i stavite echo tako da kad učitate fill\_table možete vidjeti šta će vam se stavljati u bazu i tako prilagoditi kod.

Flight::route('GET /search/@name', function($name) je dio koji služi za android i način je kako android pretražuje bazu prema nazivu i prikazuje rezultat. To sam morao naknadno dodati jer isprva mi je aplikacija izbacivala cijelu bazu pa bi imao pretraživanje, ali je rekao da aplikacija mora početi prazna pa pretraživanjem ispisivati samo rezultate. O tom kasnije, ali isto prilagodite sebi. Vjerojatno će biti slično mome samo ćete morati prilagoditi SQL.

Mislim da je to to, unesite jednu test red u bazu i probajte search ( kao što smo na kolokvijima, ako vam nešto ne radi doslovno možete proći kroz skripte za kolokvij da to ispravite ). Kada search radi, probajte fill\_table (drukčije nazovite). Kada ga jednom učitate, probate search i sad bi trebao imati više od testnog podatka.

Ne zaboravite prilagoditi podatke u bootstrap ovisno jel radite lokalno ili na serveru. Kad pređete na server morate prvo sve datoteke prebaciti na oziz pa preko puttya napraviti bazu. To je sve kao u skripti.

## 3. Android app

Za Android vam malo mogu pomoći jer se traži da bude drukčije od moga znači mijenjajte izgled Layouta i tak, ali sve što možete napraviti se može naći u skripti s kolokvija. **Adapter liste i klasu Ontologija sve po skripti.** Sve što sam zakomentiro u Adapter listi obrišite jer nije potrebno za zadatak (zanemarite sve što spominje filter i obrišite, te svugdje varijablu podaciTemp koja je ionako siva i ne koristi se).

Kao što sam spomenuo najviše možete promijeniti u Layoutu, zapravo i morate promijeniti i prilagoditi sve TextViewe svojim ontologijama (zapravo ontologija više nema veze ovdje, sad samo radite s bazom koja se napunila s podaci iz ontologije). Tu je najlakše napraviti promjene tako da se razbacajte izgledom. Nije ni teško jer možete u Designu sve, ne morate ništa pisati sami osim id koji onda pozivate u adapteru.

Što se tiče MainActivitya, preporučio bi da drukčije napravite searchview i ostavit ću link na tutorijal kako ga napraviti. Izgledat će odma drukčije od moga, a vi neće morati brinuti što treba raditi u mainactivityu: <https://youtu.be/sJ-Z9G0SDhc?t=536> .

U mom MainActivityu obrišite sve od linije 40 do 65. Video sam stavio da počne od 8:56 jer sve što je pisao u Adapteru nama ne treba (ne pišite ništa iz videa u adapter!). Napravite sve do 14:14. On tu sad stavlja neki kod u onQueryTextChange, ali nama treba ovaj gornji, onQueryTextSubmit (da funkcija nešto radi kad stisnemo Pretraži, ne tijekom tipkanja). Unutar njega stavite ovo

asyncTask = new RESTTask();

asyncTask.execute("https://oziz.ffos.hr/nastava20192020/bungar\_19/ontologija/search/"+query);

Umjesto mog linka, ide link na vašu /search rutu, i svakako ide ovaj „+query“ na kraju. Ovo se sad nadovezuje na onaj Flight::route('GET /search/@name', function($name)... Što god se stavili da se pretražuje u tom dijelu index.phpa, to će se izvršiti kada pritisnete ikonu za pretraživanje u androidu.

To bi trebalo biti to, sretno i nadam se da ćete uspjeti.