Najčešće postavljana pitanja i odgovori

1. **Pitanje:** Šta je tema projekta?
   Odgovor: Vizualizacija struktura tipa grafa.
2. **Pitanje:** Koliko je potrebno plugin-a?
   Odgovor: Minimalno 4 plugin-a, ukoliko tim broji 3 ili 4 člana. Ukoliko je tim sastavljen od 5 studenata, tada je potrebno razviti 5 plugin-a. Svaki od članova tima mora razviti minimalno jedan plugin. Broj vizualiacionih plugin-a je 2, dok je broj izvorišnih (engl. *source*) *plugin*-a je 2, odnosno 3.
3. **Pitanje:** Koji su tipovi plugin-a?
   Odgovor: Postoje dva tipa plugin-a.
   Prvi tip predstavlja izvorišni plugin, čiji je zadatak da određeni skup podataka parsira i pri tome izgradi graf (model podataka).
   Drugi tip predstavlja plugin za vizualiaciju modela podataka uz oslonac na *Django template*-a i *force-layout D3.js* biblioteku.
4. **Pitanje:** Koja razlika između plugin-a za vizualizaciju?
   Odgovor: Razlika je u nivou detalja prikazivanja pojedinačnog čvora u grafu.
   Prvi plugin (jednostavan prikaz) prikazuje pojedinačan čvor grafa u obliku proizvoljnog jednostavnog grafičkog elemenata (krug, trouga, pravougaonik, ...) kome je pridružen naziv. Primer ovakvog načina prikazivanja grafa odgovarao bi ER dijagramu.
   Drugi plugin (složen prikaz) prikazuje pojedinačan čvor grafa u obliku pravougaonika na kome je pored naziva potrebno ispisati atribute čvora. Primer ovakvog načina prikazivanja grafa odgovarao bi dijagramu klasa.
5. **Pitanje:** Šta mogu da budu izvori podataka?
   Odgovor: Bilo koji izvor podataka koji interno oslikavaja graf strukturu podataka (JSON, XML, HTML, CSV, File System, šema relacije baze podataka, ...).
6. **Pitanje:** Da li Django aplikaciju treba razviti kao komponentu i instalirati u virtualno razvojno okruženje (virtualenv)
   Odgovor: Da!
7. **Pitanje:** Da li je obavezno koristiti git ili može neki drugi VSC?
   Odgovor: Da!
8. **Pitanje:** Da li je obavezno koristiti GitLab ili može neka druga platforma (BitBucket, GitHub, ...)?

Odgovor: Da!

9. **Pitanje:** Gde implementirati model podataka?
   Odgovor: U *Django* aplikaciji koja će biti instalirana kao Python komponenta?
10. **Pitanje:** Na koji način treba vizualizovati graf?
    Odgovor: Potrebno je graf vizualizovati na tri načina:
    - *Tree View* (prikaz grafa u vidu stabla) - omogućiti akcije za dinamičko otvaranje i zatvaranja čvorova po uzoru na *package explorer* poznatih integrisanih razvojnih okruženja (IDE)
    - *Bird View* (prikaz grafa iz ptičje perspektive) - umanjeni prikaz grafa koji u potpunosti mora stati u površ za iscrtavanje.
    - *Main View* (centralni prikaz grafa) - Django template koji treba prikazati implementirati u oba plugin-a za vizualizaciju. Koristiti *d3.force layout* koji će biti obogaćen operacijama: ***pan*** i ***zoom***.
11. **Pitanje:** Da li *Tree View* i *Bird View* treba implementirati kao posebne plugin-e?
    Odgovor: Može, ali nije neophodno. Dovoljno je da budu deo *Django* aplikacije.
12. **Pitanje:** Koje operacija na modelom treba implementirati?
    Odgovor: U plugin-ima za vizualizaciji implementirati sledeće operacije:
    - pan,
    - point zoom in/out,
    - dobijanje detalja o tekućem objektu

    U Django aplikaciji implementirati sledeće operacije:
    - filter,
    - search,
    - dobijanje detalja o tekućem objektu

    Napomena: Prilikom selekcije čvora grafa na nekom od tri pomenuta prikaza, potrebno je da odgovarajući čvor dobije fokus na sva tri prikaza.
    Opciono: konzola za upravljanje platformom: izmena sintaksi, izmena modela itd
13. **Pitanje:** Na koji način je potrebno implementirati *search* operaciju?
    Odgovor: Potrebno je omogućiti unos proizvoljnog teksta (upit). Na osnovu unetog upita formirati podgraf trenutnog grafa čiji čvorovi sadrže atribute koji zadovoljavaju pomenuti upit.
14. **Pitanje:** Na koji način je potrebno implementirati *filter* operaciju?

Odgovor: Upit za filter unosi se u posebnu tekstualno polje. Format upita je sledećeg oblika <naziv_atributa> <poredbeni_operator> <vrednost_atributa>. <poredbeni_operator> može biti jedan od sledećih operatora:

- ==
- >
- >=
- <
- <=
- **!=**

Nakon unosa filtera, potrebno je formirati podgraf trenutnog grafa čiji čvorovi sadrže atribut koji zadovoljavaju uneti filter.

15. **Pitanje:** Da li je potrebno programski omogućiti izbor izvorišta podataka koji ima očekivani format zapisa (onakav kako to *plugin* očekuje)?
    Odgovor: Može, ali nije neophodno.
16. **Pitanje:** Da li je potrebno voditi računa o sesijama?
    Odgovor: Ne. Možete pretpostaviti da će sa aplikacijom u datom trenutku komunicirati najviše jedan korisnik.
17. **Pitanje:** Gde čuvati model podataka nakon parsiranja?
    Odgovor: Moguće je model grafa perzistirati u bazi (po uzoru na primer sa Vežbi9). Takođe, dozvoljeno je čuvanje na nivou instance klase naslednice *AppConfig* (pogledati način čuvanja plugin-a u primeru sa istih vežbi). Takođe, dozvoljen i neki treći način uz prethodnu konsultaciju sa predmetnim asistenom.
18. **Pitanje:** Da li je potrebno napraviti korisničke naloge (autentifikacija i autorizacija korisnika)?
    Odgovor: Ne!
19. **Pitanje:** Da li je potrebno implementirati prijavu na sistem (log-in), odnosno odjavu sa sistema (log-out)?
    Odgovor: Pročitate prethodno pitanje, kao i odgovor na isto.
20. **Pitanje:** Da li je potrebna *README.md* datoteka?
    Odgovor: Da!
21. Pitanje: Šta je potrebno da *README.md* sadrži?
    Odgovor: Koji tim, članovi tima, uputstvo za instalaciju komponenti, parametrizaciju Django projekta.
22. **Pitanje:** Na koji je način potrebno voditi git repozitorijum?

Odgovor: ~~Git repozitorijum voditi po preporučenom GitFlow modelu. Potrebno je da imate *master* granu, na ovoj grani uvek treba da bude kod koji se **može** pokrenuti. Potrebno je da imate *develop* granu na kojoj se nalazi kod koji testiraju ostali članovi tima. Kada svi istestiraju funkcionalnost (engl. *feature*), onda može da se uradi merge na master. Potrebno je da imate *feature* grane. *Feature* grane služe za razvoj pojedinačnih funkcionalnosti vaše aplikacije i odgovaraju tačno jednom *GitLab issue*-u. Jednu funkcionalnost radi jedan student. Kada završi sa radom, može da uradi *merge* na *develop* granu. Feature grane imenovati u formatu *feature/naziv_grane*.~~

Git repozitorijum voditi po preporučenom GitFlow modelu. Potrebno je da da imate *develop* granu na kojoj se nalazi kod koji se intenzivno razvija i koji često testiraju svi članovi tima. Kod na ovoj grani **mora da radi**, a čine ga do datog trenutka implementirane funkcionalnosti koje su dovoljno dobro istestirane.

Razvoj nove funkcionalnosti radi se na posebnoj *feature* grani. Svaka *feature* grana odgovara tačno jednom *GitHub issue*-u. Ove grane predstavljaju alternativne tokove razvoja. Jednu funkcionalnost radi jedan student. Kada se funkcionalnost završi, potrebno je ovu granu spojiti sa develop granom putem ***merge request*-a** (*pull request*-a). *Feature* grane imenovati u formatu *feature-<naziv_grane>*[1].

U slučaju da primetite neki *bug*, tada pronalazite *issue*[2] koji odgovara funkcionalnosti, ponovo ga otvarate i kreirate odgovarajući *bugfix* granu (format imenovanja je *bugfix-<naziv_grane>*[3]). Na njoj popravljate *bug* i radite inicijalno testiranje. Kada ste utvrdili da je problem otklonjen, kod spajate sa *develop* granom putem ***merge request*-a** (*pull request*-a).

Periodično, kod sa *develop* grane prebacujete na master granu. Master grana bi trebalo da sadrži samo *commit*-e koji se odnose na *release*-ove, tj. zvanične verzije Vaše aplikacije. Potrebno je imati minimalno onoliko verzija koliko i *milestone*-ova.

Napomena: Sve commit poruke moraju početi sa ***#jedinstveni-identifikator-issue*** kako bi se referencirao issue na koga se commit odnosi. Pogledati odgovor na pitanje 27.

23. **Pitanje:** Koliko *GitLab milestone*-ova je neophodno napraviti i do kada su rokovi za izradu svakog?

1 Ignorisati < i > karaktere.
2 Način upotrebe issue-a objašnjen kasnije.
3 Ignorisati < i > karaktere.

Odgovor: *Milestone* oslikava jedan bitan momenat životnog ciklusa vaše aplikacije. Što se tiče ovog projekta možete da imate **2-3** milestone-a. Rokovi su: I kontrolna tačka, (eventualno II kontrolna tačka) i Predaja projekta (biće naknadno specificirani u dogovoru sa profesorom). *Milestone* sadrži *task*-ove (*GitLab issues*), koji su implementirani tokom tog *milestone*-a.

24. **Pitanje:** Na koji način pravimo *GitLab issue*-e (*task*-ove)?
    Odgovor: Jedan *task* treba da se odnosi na jednu funkcionalnost aplikacije (engl. *feature*), koja će biti razvijena na posebnoj *feature* grani. *Task*-ove je potrebno napraviti unapred za *milestone* koji je u toku. Zatim, svaki student uzima task po task (dodeljuje mu se *issue*), te započinje implementaciju odgovarajuće funkcionalnosti. Kada se implementacija funkcionalnosti završi i uspešno odradi inicijalno testiranje, *task* se zatvara, a kod prebacuje na develop granu. Nakon toga, ostale kolege treba da urade testiranje svih funkcionalnosti. Ukoliko se ustanovi problem u nekoj od njih, potrebno je ponovo otvoriti odgovarajući task (**ne praviti nov**).
25. **Pitanje:** Šta treba da bude povratna vrednost vizualizacionog plugin-a?
    Odgovor: String reprezentacija HTML dokumenta, koju je potrebno umetnuti u Django template django aplikacije. String reprezentaciju generisati ručno ili upotrebom neke biblioteke za generisanje *HTML template*-a (npr. jinja2).
26. **Pitanje:** Da li korstiti labele prilikom rada sa *GitLab issue*-ima?
    Odgovor: Da. Možete iskoristiti predefinisani skup labela.
27. **Pitanje:** Da li je obavezno da prilikom commit-a referenciramo issue na koji se commit odnosi?
    Odgovor: Da! Navesti *#issue_jedinstveni_identifikator* unutar commit poruke.
28. **Pitanje:** Na koji način zatvoriti GitLab issue?
    Odgovor: Direktno putem *GitLab*-a ili indirektno kroz poruku *commit*-a.
29. **Pitanje:** Šta ako implementiramo sve obavezne funkcije, ali posao nije ravnopravno podeljen?
    Odgovor: Na adresi (slide 11), se nalazi spisak obaveznih operacija koje treba implementirati. Ako implementirate sve obavezne operacije, ispod njih imate opcione/dodatne operacije koje možete implementirati.
30. **Pitanje:** Ako implementiramo neku od opcionih/dodatnih operacija, da li to može da zameni neku od obaveznih operacija (džoker)?

Odgovor: NE!!! Operacije koje su specificirane su obavezne operacije. Operacije koje su specificirane pod opcione služe isključivo **ako bude studenata zainteresovanih za dodatno zalaganje na predmetu, a koiji** su pri tom implementirali sve obavezne operacije, i žele/mogu/hoće da urade više.

31. **Pitanje:** Da li naš model grafa treba da podrži sve strukture xmla, htmla, jsona ili mi deviništemo na neki način koju strukturu mora imati xml, html, json koji se mapira na model grafa?
    Odgovor: Odgovoriću na pitanje u kontekstu XML formata dokumenta, koji je ujedno i nadskup HTML formata. Slično treba primeniti i za JSON.
    Potrebno je podržati **proizvoljan** XML dokument. Elemente XML dokumenta treba mapirati na čvorove grafa, dok atribute elemenata XML dokumenta treba mapirati na atribute čvorova grafa.
32. **Pitanje:** Takođe na koji način definišemo ciklične grafove u npr. xml fajlu? Jedini način na koji mi sada vidimo je da svaki čvor ima neki id pa da ih preko toga spajamo jer u suprotnom dobijamo samo stabla?
    Odgovor: Što se cikličnih grafova tiče, pogledajte na koji način biblioteke za serijalizaciju perzistiraju graf proizvoljnih objekata u XML dokument. Primer jedne od datoteka je XStream.
33. **Pitanje:** Takođe me zanima da li mi u json smemo da napravimo "pretpostavku" da neki json objekat ima ključ "value" koji bi vizualizacija matematičkog grafa koristila ili ne bi trebalo nikakve pretpostavke da pravimo?
    Odgovor: Treba podržati proizvoljnu strukturu JSON dokumenata, tako da nema pretpostavki.
34. **Pitanje:** Da li u istoriji izmena smeju da se nađu izmene nekoga van članova tima?
    Odgovor: Ne, u istoriji izmena ne treba da se nalaze izmene nikog sem članova vašeg tima. Ako morate da radite na tuđoj mašini iz bilo kog razloga onda podesite **git config.** Primer možete videti ovde. Ako se nađu izmene van članova tima, nema nikakve garancije da taj neko nije radio za vas.
35. **Pitanje:** Rečeno je da svi članovi tima treba rade na core komponenti. Na gitlabu nije moguće dodeliti issue svim članovima tima, već samo jednoj osobi. Šta u tom slučaju da radimo? Da li da issue ostavimo unassigned?

Odgovor: Odraditi finu granulacija *issue*-a, tj. Raščlaniti ga na manje (*atomične*) *issue*-e, koje ćete dodeliti pojedinačnim članovima tima.

36. **Pitanje:** Mala korekcija odgovora na pitanje 22.
    Odgovor: Na **master** grani bi trebalo da imate commit-e koji se odnose na ***release***-ove Vaše aplikacije, tj. "proizvode" koje dostavljate za **kontrolne tačke, tj. odbranu projekta**. Ovi *commit*-i bi se direktno odnosili na kompletirane ***milestone***-ove. Na ***develop*** grani čuvajte verziju aplikacije koja ima **funkcionalnosti koje rade** zajedno. Ukoliko primetite ***bug*** ili želite dodati **novu funkcionalnosti**, tada pravite ***bugfix/feature*** **granu** koju vezujete za Vama **dodeljeni issue**. Ovom prilikom Vam dostavljamo još jedan koristan članak o upotrebi git-a.
37. **Pitanje:** Java biblioteka preporučena u FAQ-u (XStream) serijalizuje graf proizvoljnih objekata dodavanjem atributa "reference=relativna putanja" elementu koji sadrži postojeći element (petlja). Ovaj način označavanja nije prepoznat od strane drugih biblioteka i prilikom serijalizacije grafova, ukoliko sadrže petlju, dolazi do greške. Da li je prilikom učitavanja XML fajla potrebno da se pridržavamo "konvencije" koju XStream koristi (tražiti "reference" atribut i ručno raditi referenciranje) ili samo pustiti biblioteku za čitanje XML-a da odradi posao (u tom slučaju će graf uvek biti stablo)?
    Odgovor: *XStream* je primer biblioteke koja je sposobna da graf objekata perzistira u vidu XML dokumenta.
    Naravno da to nije jedini način, ali svakako predstavlja dobar primer.
    Za manipulaciju čvorovima XML dokumenata mogu se koristiti i jezici poput *XPath*, *XQuery*, itd.
    Neophodno je da ste u mogućnosti da parsirate proizvoljan XML dokument i na osnovu njega izgradite graf. Da li će graf biti cikličan ili acikličan, zavisi od XML dokumenta. Stabla dobijate po prirodi od samog parsera, a Vaš zadatak jeste da razvijete plugin koji će biti u mogućnosti da na osnovu posebnih atributa (*reference* i sl.) odradite naknadno spajanje čvorova u cilju prepoznavanja ciklusa.
38. **Pitanje**: Da li komponenta za vizualizaciju treba da dobije cvorove i grane, pa da renderuje template, i vrati rezultat renderovanja?
    Odgovor: Prihvata čitav graf, a kao rezultat vraća HTML reprezentaciju grafa (DOM stablo), koja se ubacuje u template *core* komponente.
39. **Pitanje:** Da li je u redu da izvorišta podataka budu dva dokumenta istog tipa (npr. JSON), koji sadrže različite podatke (npr. jedan sadrži

informacije o avio-linijama, dok drugi sadrži informacije o društvenim mrežama).

Odgovor: ~~Da~~. Ne!

40. **Pitanje:** Da li brisati feature/bugfix git grane?

    Odgovor: Ne! Sve grane koju su korišćene tokom izrade projektnog zadatka treba da ostanu prisutne na repozitorijumu. Pomenute grane predstavljaju jednu od stavki koje spadaju u “git aktivnost”.

41. **Pitanje:** Koje dijagrame treba postaviti na repozitorijum?

    Odgovor: Model komponenti i **dijagram klasa** Vaše aplikacije.

42. **Pitanje:** U kojoj formi treba predati (postaviti na repozitorijum) dijagrame?

    Odgovor: U formi fotografiji ili PDF dokumenta, kako bi se što lakše mogao videti sadržaj dijagrama.

43. **Pitanje:** Šta sve treba prikazati na dijagramu klasa?

    Odgovor: Minimalno treba prikazati klase koje čine model grafa, kao i “servisne” klase koje koristite u pluginima za učitavanje i vizualizaciju.

44. **Pitanje:** Koji alat koristiti za kreiranja dijagrama?

    Odgovor: Koristiti alat po želji. Profesor je preporučio PlantUML. Takođe, možete koristiti i *online* alat draw.io.

45. **Pitanje:** Da li je moguće commit naknadno povezati sa Issueom? Napravila sam issue i commitovala medjutim nisam id issue-a ubacila u komentar nego van njega pa se nije povezalo.

    Odgovor: Trebalo bi da možete izmeniti poruku pomenutog *commit-a*.

46. **Pitanje:** Kako izmeniti poruku starog *commit-a*?

    Odgovor: Uputstvo možete pronaći na sledećem linku. Pronađite situaciju koja odgovara Vašoj. Obratiti pažnju da to da li su izmene *push*-ovane, kao i da li se menja poslednji *commit* ili neki stariji.

47. **Pitanje:** Kolege i ja diskutujemo dugo o tome, a problem je sto nismo jasno razumeli sta se tacno ocekuje od nas. Spominjali ste ciklicne grafove, moje pitanje je da li nas model treba da ih podrzi, i da stanemo na tome, ili se od nas ocekuje da pravimo neke predpostavke u XML-u recimo, i da od njega formiramo ciklicni graf. XML, pa i njegov podskup HTML su hijerarhijske strukture, odnosno stabla, i one nemaju po njihovoj prirodi kruzne veze.

    Dakle, da sumiram. Da li trebamo recimo da vodimo racuna o odredjenim atributima u XML-u, pod pretpostavkom da oni predstavljaju referencu ka drugom entitetu (Primer <Student fakultetId="123"></Student>) ili je

dovoljno da mi samo ucitavamo XML takav kakav jeste i podrzimo bilo koji XML bez uvodjenja dodatnih ogranicenja. U slucaju ovog drugog, podrsku za ciklicne grafove bi mogli da demonstriramo kroz Source komponentu nekog drugog formata koji po prirodi u sebi ima kruzne veze, a ne da od XML-a izvlacimo to na silu.
Jasno mi je da postoje biblioteke koje podrzavaju tako nesto, i zapravo nije ni toliki problem podrzati to, ali se malo kosi sa cinjenicom da podrzavamo bilo koji XML fajl, ako se slazete to znaci isto sto i podrzavanje XML formata bez uvodjenja dodatnih ogranicenja.

Odgovor: Model grafa mora uzeti u obzir cikluse.
Ukoliko izvorište podataka za koje ste se opredelili predstavlja XML dokument koji reprezentuje graf (a ne specijalan slučaj stabla), onda morate naći način da cikluse u tom dokumentu obradite (verovatno će se korisitit neki atributi koji predstavljaju reference XML elemente). Plugin bi se u tom slučaju specijalizovao za tako nešto.
Sam postupak parsiranja XML dokumenta je trivijalan, te ne vidim zašto je problem proveriti da li postoje neki atributi tipa reference ili drugi, a na koje ste nailazili u specifičnim formatima XML dokumenata. U tom slučaju, Vi biste de facto učitavali bilo koji XML kao stablo (biblioteka za parsiranja Vam to dozvoli out-of-the-box), a u slučaju da se specifični atributi prepoznaju, korisniku biste ponudili graf.

48. **Pitanje:** Da li je potrebno da imamo tagove u našem repo-u?
    Odgovor: Nema potrebe i nije obavezno. Ako neko ipak želi, može tagovati verziju sa komoj ćete braniti projekat.
49. **Pitanje:** Prilikom izmene views.py fajla u CORE komponenti i njegove ponovne instalacije, ove promene se nekad nasumicno ne azuriraju u venv-lib-sitepackages-core, tako da dobijam da se pokrece stariji kod, cak i nakon ponovne instalacija izmenjene core komponente. Da li znate kako je moguce 'refreshovati' ove fajlove tako da se izmene nastale u komponenti pojave i tu? Takodje, koja je neka preporucena metoda za razvoj komponentnih aplikacija, sa obzirom na cinjenicu da se pri svakoj promeni mora izvršiti ponovna instalacija? U nasem projektu je i CORE jedan plagin, u kojem je django aplikacija (po uzoru na vezbe). Pri svakoj izmeni u ovim komponentama moram opet raditi instalaciju pa pokretati server, sto znatno otezava rad i dugo traje, ali drugacije "ne registruje" promene. Da li sam propustila nesto ili je to stvarno jedini nacin?

Odgovor: Preporuka je napisati skriptu (shell, python, perl, ...) koja bi odradila sav dodatan posao za Vas. Njen zadatak bi otprilike bio da odradi sledeće:

a. Iz razvojnog okruženja izbrisati sve instalirane komponente, njihove build direktorijume, itd.
b. Ponovno instalirati komponente u razvojno okruženje.
c. Pokrenuti server.

50. **Pitanje:** Da li je u redu da za projekat iz SOKA filtraciju i pretragu radimo na frontendu jer smatramo da je to mnogo brže od toga da filtraciju i pretragu radimo na bekendu i ponovo sve vizualizujemo.

    Odgovor: Ne! Izgradnju podgrafa upotrebom *filter*, odnosno *search* operacija potrebno je obaviti unutar klasa Vašeg modela grafa (unutar *Django core* aplikacije).

51. **Pitanje:** Poštovani, ako želimo da napravimo neku sitnu izmenu na komponenti nakon merge-a feature grane sa develop granom, a nije bug u pitanju, kako da nazovemo novu granu na kojoj ćemo praviti izmenu? Ili ponovo da iskoristimo feature granu koju smo koristili za razvoj komponente ali ne znam koliko je to dobra praksa?

    Odgovor: Upotrebom stare feature grana dolazite u situaciju da radite merge develop grane u pomenutu feature granu, što svakako nije dobra praksa. Bolje napraviti novu granu koja će se zvati isto kao feature grana, s tom razlikom da će imati prefiks/sufiks *enhancement* ili sl.

52. **Pitanje:** Da li postoji igde PDF ili koja dokumentacija za to sta specifično mora biti odradjeno u odnosu na broj članova tima?

    Odgovor: Stavka GitLab *aktivnost* podrazumeva i menadžerski deo posla koji se tiče podjednake/fer raspodele funkcionalnosti unutar članova tima. S obzirom na to da se ocenjuje način na koji se zadaci raspoređuju unutar tima, ne postoji PDF koji bi to uradio umesto Vas.

53. **Pitanje:** Takodje, ne razumem najbolje sta tree view treba da sadrzi. Da li npr, ako bi radio vizualizaciju FB i Twitter naloga? Da li bi u tree viewu imao 2 Cvora / klase Koji se zovu "Facebook" i "Twitter"? I ako jeste tako, šta su njihova deca / instance / podešeni izvori konkretno? Npr za Facebook, da li bi njegovo dete bilo "Nalog Pere Perića" i onda ako kliknemo na njega prikaže kroz graf sve prijatelje Pere Perića

    Odgovor: Ukoliko ne zamerate, potrudio bih se da na Vaše pitanje dam uopšten odgovor. Odgovor dajem u kontekstu neusmerenih grafova, ali i na usmerene grafove bi trebalo da je moguće primeniti sličan princip.

Naime, neophodno je da prepoznate da li je graf povezan. Ukoliko jeste, proveravate da li je acikličan. U slučaju povezanog acikličnog grafa (stabla), potrebno je pronaći korenski čvor, nakon čega sledi vizualizacija stabla. U slučaju da je graf cikličan, proglasićete neki čvor za čvor od koga započinje postupak vizualizacije i tada treba voditi računa o potencijalnoj rekurziji koja se javlja kada korisnik zapadne u ciklus.

Ukoliko se ispostavi da je graf nepovezan, tada sve njegove komponente treba prikazati kao podgrafove. Zamislite situaciju u kojoj imate radni prostor *IDE*-a u kome se nalazi nekoliko zasebnih projekata.

Jedan od zadataka jeste da zađete malo dublje u teoriju grafova, kao i načine za njihovu vizualizaciju.

Takođe, Vaš zadatak je da odaberete izvorište podataka koje odgovara strukturi grafa. Pažljivim tumačenjem podataka trebalo bi prepoznati entitete (čvorove), kao i njihovu međusobnu povezanost (grane).

54. **Pitanje:** Imam pitanje u vezi filter opcije. Vecina atributa u nasim izvorima su stringovi ili string reprezentacija brojeva. Kada radimo filter, da li da pokusamo prvo da ih parsiramo u int vrednost i tako poredimo poredbenim operatorima (<, >, ==, ...) ili mozemo da ih poredimo sve leksikografski?

    Odgovor: Zadatak plugina za parsiranje je da podatke parsira u one tipove koje podaci predstavljaju. Svakako da treba posebno izdvojiti brojeve, datume, itd. u odgovarajuće tipove podataka. Filter primenjuje relaciju poretka u skladu sa tipom atributa.

55. Pitanje: Kako mozemo menjati broj polja u django modelu posto imamo cvor koja sadrzi listu elemenata koja nije poznata na pocetku.

Odgovor: Django model je ORMaper za relacionu baze, i kao takav izmena šeme baze ili tabela nije nužno jednostavan proces (odgovor zašto možete potražiti na predmetu Baze Podataka). Morate unapred dobro izmodelovati vaše tabele, tako da mozete da pokrijete sve razlicite situacije, znači da napravite generički model i generičko rešenje. Ako ne želite da potrošite vreme na modelovanje, druga opcija je da nakon obrade *source* plugin-a radite migraciju. Poslednja opcija je da ne koristite django model, nego da koristite mogućnost koja je opisana u pitanju/odgovoru 17.

56. **Pitanje:** Šta tačno podrazumeva "parametrizacija django projekta? koja nam treba u .md fajlu" Da li to obuhvata opis načina pokretanja servera ili nešto drugo?

Odgovor: Uputstvo za preuzimanje, instalaciju, pokretanje i upotrebu Vaše aplikacije.

57. **Pitanje:** Da li je bird view samo statična slika grafa koja ne bi trebalo uopste da se pomera? Dakle nema zoom i pan vec je poenta da ceo graf stane bez obzira na zumiranje na main view?
    Odgovor: Bird view odražava stanje grafa prikazanog na glavnom prikazu u ptičjoj perspektivi. Korisnik nema mogućnosti upotrebe pan i zoom-a, ali se force layout koristi, kako bi se graf pomerao u skladu sa pomerajima na glavnom prikazu.
58. **Pitanje:** Ako se zumira na glavnom prikazu, bird view se ne zumira?
    Odgovor: Da.
59. **Pitanje:** Kao sto se moze videti, cvorovi su dobro povezani, medjutim nisu dobro pozicionirani, zbog toga sto bird view koristi Dekartov koordinatni sistem, a jednostavni prikaz koristi koordinatni sistem u odnosu na ekran racunara. Ovo predstavlja problem jer kada probamo da dodamo tacke na bird view, on ne moze da ih mapira adekvatno. Da li bismo mogli da ostavimo bird view ovakav kakav jeste, ili da ga izmenimo da bude slicniji standardnom prikazu?
    Odgovor: Trebalo bi tačke jednog koordinatnog sistema mapirati na tačke drugog upotrebom operacija kao što su translacija, rotacija, i sl.
60. **Pitanje:** Imamo implenentirano dobijanje detalja o tekućem objektu preko plugina za vizuelizaciju, znači kad kliknemo na node dobićemo detalje o tom objektu ispisane na stranici. Kako onda treba da funkcioniše dobijanje detalja preko Django aplikacije, tj. kad treba da koristimo to dobijanje ako smo već implementirali preko vizuelizacije?
    Odgovor: S obzirom na to da sva tri prikaza sadrže informacije o istom grafu, verujem da iz njih možete izvući informacije o selektovanom čvoru. U suprotnom bi trebalo praviti Ajax poziv koji bi dovukao informacije sa servera o jednom čvoru putem odgovarajućeg identifikatora.
61. **Pitanje:** Da li možemo pretpostaviti da će se projekat pokretati u PyCharm-u te da iz tog razloga samo objasnimo parametrizovanje i podešavanje projekta kroz pokretanje u PyCharm-u ili je potrebno da napišemo i podešavanje kroz komandnu liniju?
    Odgovor: S obzirom na to da su referentni primeri sa vežbi pokretani u terminalu, od Vas se može očekivati isto. Tako da je moj savet napraviti uputstvo po uzoru na uputstva sa vežbi.

62. **Pitanje:** Da li je u redu da napišemo jednu bat skriptu koja će da vrši installaciju komponenti i u read me samo da napišemo kako se ona poziva? Ukoliko jeste, da li se projekat brani na Windows računaru, tj. da li je u redu da napišemo readme kao da se samo sa Windowsa pokreće?
    Odgovor: Podržavam ideju za pisanje bat skripte. Takođe, zamolio bih da u istom maniru napišete shell/python/perl skriptu koja bi omogućila instalaciju i pokretanje aplikacije na Unix-like sistemima. U NTP, koliko znam, instaliraniu su Linux operativni sistemi.
63. **Pitanje:** Dodeljen nam je još jedan kolega u tim. Ako se ne varam, projekat sa 5 članova je kako teže ukombinovati međusobno tako i teže zbog veće količine rada koji projekat nosi sa 5. članom. Šta raditi ukoliko neki od članova neće da sarađuje?
    Odgovor: Što se izrade projekta tiče, ništa nije teže za petočlane timove. *Core* aplikacije ćete podeliti na 5 delova (manje posla po čoveku), a svaki od članova razvija po jedan plugin. Napomenuo bih da se ocena dobija pojedinačno.
    Ako neko od članova pak odbije saradnju, broj plugina će se smanjiti za jedan. Na kraju svako od studenata treba da razvije tačno jedan *plugin*. Funkcionalnosti *Core* aplikacije se ne skaliraju (već dele na članove koji "žele" da rade).
64. **Pitanje:** Koliko će biti kontrolnih tačaka? Da li se zna kada će biti?
    Odgovor: Biće najmanje jedna, a najviše tri kontrolne tačke. Biće Vam blagovremeno javljeni termini istih.
65. **Pitanje:** Kako izgledaju kontrolne tačke? Da li se ocenjuju?
    Odgovor: Asistent će odabrati nekoliko timova koji treba da ukratko prezentuju (10-15 minuta) šta su do datog trenutka odradili. Prilikom prezentacije, osvrnuti se na sledeće:
    - Šta je odrađeno?
    - Zanimljivi problemi sa kojima ste se susreli?
    - Kakav je plan za dalje?
    - Nedoumice za nastavak izrade?
66. **Pitanje:** Da li se kontrolne tačke ocenjuju?
    Odgovor: Ne, ali mogu doprineti sticanju boljeg utiska o zalaganju tima, što svakako može značiti tokom odbrane projekta.
67. **Pitanje:** Može li nekoliko primera *source plugin*-a?
68. **Odgovor:** XML,YAML, JSON, CSV dokument, wiki stranica, šema relacije baze podataka, binarni fajlovi, itd.

69. **Pitanje:** Da li imamo specifikaciju?
    Odgovor: Profesorova prezentacija i ovaj dokument sasvim dovoljno opisuju šta je potrebno uraditi za projektni zadatak.
70. **Pitanje:** Da li plugini (vizualizacioni i *source*) treba da budu eksterni?
    Odgovor: Da. Razvijate ih i instalirate upotrebom *setuptools*-a, a dinamičko prepoznavanje radite upotrebom i prepoznajete sa *pkg_resources.iter_entry_points*.
71. **Pitanje:** Da li *Core Django* aplikacije treba da bude poseban plugin?
    Odgovor: Da! Pogledati primer sa vežbi 9.
72. **Pitanje:** Koliko čvorova treba da ima izvorište podataka?
    Odgovor: Što više to bolje. Neka bude minimalno nekoliko stotina. U tom slučaju ima smisla primenjivati search i filter. Nekoliko čvorova čovek manuelno može pretražiti. Prilikom razvijanja aplikacije, možete koristiti neka manja izvorišta, kako bi Vam učitavanje radio brže. Na odbrani ćemo videti koliko je Vaše rešenje primenljivo za neki stvarni slučaj, te prodiskutovati eventualno o mogućim optimizacijama.
73. **Pitanje:** Gde pravimo filter i search?
    Odgovor: U *Core* komponenti, jer rade nad modelom grafa. Nezavisni su od plugins koji je učitao/prikazao podatke
74. **Pitanje:** Gde čuvati source fajlove za source plugine?
    Odgovor: Bilo gde unutar tog plugina.
75. **Pitanje:** Da li je bitan procenat završenosti milestone?
76. **Odgovor:** Da! Težite da do krajnjeg roka milestone procenat završenosti tog milestone bude 100%. U tom slučaju se milestone smatra uspešnim. U suprotnom, došlo je do probijanja roka.
77. **Pitanje:** Da li imate neki primer xml ili json fajla za projekat iz SOK-a? Pošto ne znamo odakle da izvučemo recimo podatke o korisnicima društvenih mreža.
    Odgovor: Nemam kod sebe konkretan dokument, ali bih mogao da Vam preporučim da iskoristiti neki od javnih API-a za tako nešto. Npr. Twitter, FaceBook, Instagram, GitHub. Takođe verujem da postoji API o aerodromima, državama, i slično.
78. **Pitanje:** Kako izgleda komunikacija između plugin-a i Core Django aplikacije?
    Odgovor: *Source* plagin parsira izvorište podataka i na osnovu toga instancira klase modela kako bi napravio graf. Taj graf prosleđuje Core aplikaciji koja ga negde interno sačuva (npr. AppConfig, baza, itd).

Nakon toga, Core aplikacija šalje podatke plaginu za vizualizaciju koji na osnovu njih gradi HTML(+CSS+JS) DOM stablo. Taj HTML se vrati *Core* aplikaciji koja ga umeće na odgovarajuće mesto trenutno prikazanog Django template.
Kada korisnik poželi da pretraži ili filtrira čvorove grafa, to radi *Core* aplikacija nad grafom koji je negde prethodno perzistirala (videti početak odgovora). Rezultat ove operacije je podgraf koji se šalje vizualizacionom plaginu, koji ponovo gradi HTML DOM stablo. Ono biva vraćeno *Core*-u i umetnutno u template.
Napomena: Search i Filter ne implementirati na klijentskom delu aplikacije (JavaScript).

79. **Pitanje:** Da li je u redu da se Search i Filter implementiraju u JavaScriptu i rade nad vizualizovanom reprezentacijom grafa?
    Odgovor: Ne! Ove operacije se implementiraju na serverskom delu aplikacije i rade nad modelom (instancama Python klasa modela)
80. **Pitanje:** Pošto se u projektu traži postojanje virtuelnog environment-a, kako da se on deli kroz git, pošto se u fajlu venv/pyvenv.cfg nalaze apsolutne putanje do python-vezanih stvari za instalaciju, one osobe koja je kreirala taj virtual environment, a nije praktično da svaki put menjamo putanju do naše instalacije pythona (i ostalih python-vezanih stvari).
    Odgovor: Svako od Vas treba da ima posebno okruženje koje se ne stavlja na git. Na git stavite requirements.txt fajl u kome ćete čuvati verzije neophodnih paketa. Više informacija možete pronaći ovde.
81. **Pitanje:** Nismo sigurni da li treba deo dužnosti koje se izvršavaju u core aplikaciji na primeru sa vežbi - konkretno tamplates, html, css i views - prebaciti u druge apilikacije (vizualizatore), ili ih treba ostaviti tu i staviti ostale dužnosti u druge aplikacije
    Odgovor: Sav statički sadržaj treba da se nalazi u core aplikaciji (Django templates, CSS, JS), kao i Django views.
    Ono što vizualizatori treba da odrade jeste da generišu HTML koji se inplace-uje u odgovarajući template glavne stranice Core aplikacije. Naime, jedan deo stranice biće predviđen za glavni prikaz i tu umećete HTML koji vizualizacioni plugin. Više informacije možete videti u pitanju 25 ovog dokumenta.
    Suštinski, vizualizacioni plugin primi referencu na instancu grafa koji pripada modelu Core aplikacije. Nakon toga, generiše HTML tagove koji će odgovarati informacijama u grafu (g tagovi, krugovi, pravougaonici,

tekst, itd.). Taj HTML se vrati Core aplikaciji, koja ga umetne na odgovarajuće mesto u DOM stablo stranice.

82. **Pitanje:** Da li je uredu da prolazimo kroz krljučeve npr. u JSON fajlu i da pri pravljenju grana izmedju nasih objekata smemo da pitamo da li taj neki kljuc ima naziv ‘references’ i ako ima da onda povezujemo trenutni objekat sa tim objektima iz references? Ukoliko je to uredu, da li bi mogli da imamo ‘pretpostavku’ o tome da ce ta imena u references predstavljati kljuceve(jedinstveni atribut) tog drugog objekta sa kojim nas objekat zelimo da povezemo? Odnosno, sta vrednost te reference treba da predstavlja?

    Odgovor: Predloženo rešenje bi se moglo primeniti samo na posebnu vrstu JSON dokumenata, ali ne i na generički JSON dokument. Pokušajte sa generičkim parsiranjem, koga možete kasnije obogatiti parsiranjem specijalnih atributa. Npr. Tokom parsiranje, uvideli ste da je u pitanju neki specijalan JSON i onda ste njega parsirali drugačije od ostalih JSON dokumenata.

    U nastavku sledi jedan primer JSON dokumenta (paste-ujte u pretty printer za bolji prikaz). Jedan pristup u generičkom parsiranju bi mogao rezultovati grafom u kome važe sledeće veze između čvorova::

    čvor "id1" (sa atributom Name=John), ima **2 children grane** na čvorove **child1, child2** i jednu **wife granu** na čvor sa imenom Lucy (ukoliko recimo fali id, možete ga sami generisati).

    ```
    {
    "id1": {
    "Name": "John",
    "Children": [
    {
    "child1": {
    "Name": "Mike"
    },
    "child2": {
    "Name": "Lucy"
    }
    }
    ],
    "Wife": {
    "Name": "Angie"
    ```

}
}
}

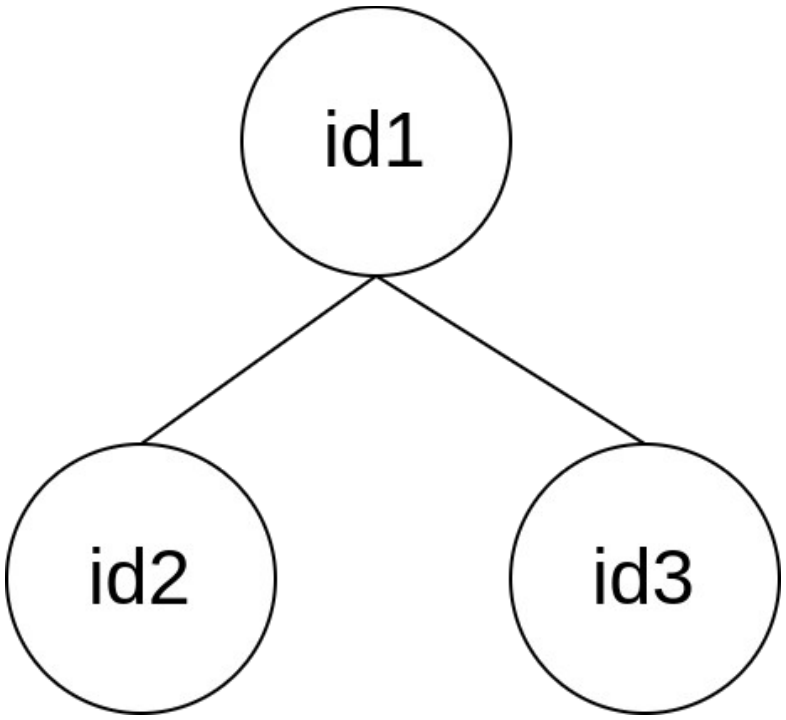
83. **Pitanje:** Da li možete dati neki primer jednostavnog izvorišta (npr. JSON dokumenta) i objasniti kako treba da izgradimo TreeView, BirdView, odnosno prost i složen prikaz MainView-a?
Odgovor: Primer JSON dokumenta bi mogao biti sledeći.

```
{
  "id": "id1",
  "first": "John",
  "last": "Doe",
  "years": 53,
  "children": [
    {
      "id": "id2",
      "first": "Mike",
      "last": "Doe",
      "years": 25,
      "children": []
    },
    {
      "id": "id3",
      "first": "Lucy",
      "last": "Doe",
      "years": 15,
      "children": []
    }
  ]
}
```

Jednostavne attribute prikazujemo kao parove ključ:vrednost, dok *children* listu koristimo za izgradnju veza. Ukoliko je neki drugi atribut složen JSON objekat, onda i taj attribute možete koristiti kao granu (pogledati primer iz prethodnog pitanja za atribut *Wife*). Sledi primer TreeView-a pošto je on najpogodniji za početak. TreeView je prikazan u formi teksta. Znak plus pored objekta znači da se on može otvoriti/raširiti, dok minus znači da se objekat može zatvoriti/sakriti. *TreeView* ne morate razviti kao posebnu komponentu, nego ga možete implementirati unutar Core komponente.

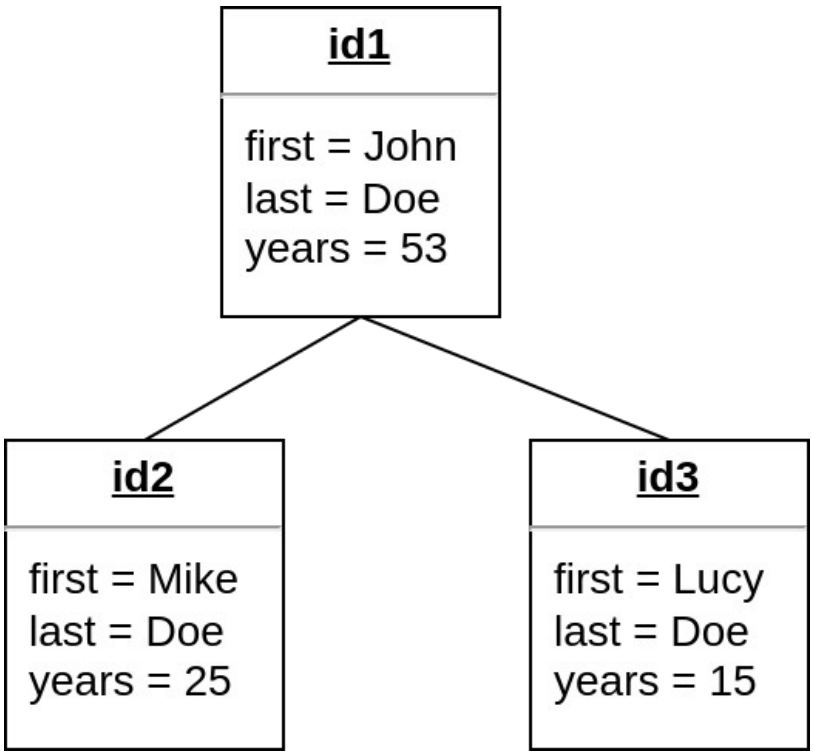
```
    id: id1       # jednostavni atributi
first: John,
last: Doe,
years: 53,
        -    id: id2,         # referenca na drugi objekat sa id2
      first: Mike,
      last: Doe,
      years: 25,
         +    Id: id3  # Sakrili smo informacije o cvoru id3
```

MainView je belo platno koje popunjavate sadržajem generisanim u pluginima. Ukoliko je korisnik odabrao jednostavan prikaz, tada će odgovarajući plugin generisati DOM HTML stablo/graf koje/i oslikava graf nastao parsiranjem priloženog JSON dokumenta. Ovaj HTML biva umetnut na platno MainView-a. Jednostavan prikaz služi da korisnika upozna sa strukturom grafa. Od informacije prikažite nešto što jedinstveno određuje čvor grafa (id, naziv, itd.). Možete prikazati i nešto više informacije ukoliko smatrate da su posebno korisne za jednostavan prikaz, ali težite da zaista bude jednostavan. Sledi primer jednostavnog prikaza.



Ukoliko korisnik promeni odluku i odabere složen prikaz, tada ispraznite platno MainView-a. Nakon toga, upotrebom odgovarajućeg plugina generišete novo/i DOM HTML stablo/graf koje/i oslikava graf nastao

parsiranjem priloženog JSON dokumenta. Složen prikaz treba da prikaže sve informacije o objektu i mogao bi da izgleda ovako.



BirdView treba da oslikava glavni prikaz sa tom razlikom da **ceo graf mora stati na platno predviđeno za iscrvanje**. Vaš zadatak je da pronađete odgovarajući faktor skaliranje. Drugačije rečeno, to je prikaz grafa koji se već nalazi na MainView-u, samo iz **pričje perspektive**. Onemogućiti operacije pan, zoom, drag, itd. na BirdView-u. BirdView ne morate razvijati kao posebnu komponentu, nego to možete uraditi u okviru Core komponente.

Napomena: Selekcija čvora na bilo kom prikazu, dovodi do selekcije istog čvora na ostalim prikazima. Operacije Search i Filter implementirati u Core komponenti. One rade nad grafom modelovanim Python objektima koji čine model Vaše aplikacije. Ove operacije dovode do formiranja podgrafa u zavisnosti od upita, odnosno kombinacije filtera. Nakon toga, ažurirate sve prikaze tako da oslikavaju rezultujući podgraf.

84. **Pitanje:** Da li je obavezna sukcesivna primena filter, odnosno search operacije? Na koji način da vratiti incijalni graf, koji je rezultat parsiranja skladišta?
Odgovor: Da! Primer: Prikazan je početni graf G1 nad kojim primenjujete filter opeaciju. Ona rezultuje podgrafom G2. Nad G2 možete primeniti

search koji rezultuje grafom G3. Nad G3 možete primeniti filter, koji rezultuju G4, itd. Način povratka na inicijalni graf prepušten je Vama. Potrudite se da za to odaberete akciju koja je intuitivna korisniku.

85. **Pitanje:** Da li treba priložiti još nešto osim koda?
    Odgovor: Pogledati pitanje 41.
86. **Pitanje:** Da li treba podržati *undo* operaciju (vraćanje na prethodni podgraf) prilikom Search i Filter operacija?
    Odgovor: Možete, ali nije obavezno
87. **Pitanje:** Da li su svi plugini međusobno nezavisni?
    Odgovor: Da!
88. **Pitanje:** Da li možemo koristiti gotove parsere za JSON, XML, CSV, itd.?
    Odgovor: Da!
89. **Pitanje:** Koji plugin je zadužen je zadužen za pretvaranje podataka u strukturu grafa, tj. instanciranje objekata modela?
    Odgovor: Izvorišni plugin parsira odgovarajuće skladište s podacima na osnovu koji pravi instance klasa Vašeg modela (čvorove, grane, graf).
90. **Pitanje:** Da li dijagram klasa treba da sadrži sve napravljene klase?
    Odgovor: Da!
91. **Pitanje:** Da li selekciju čvora i prikaz detalja o selektovanom čvoru implementirati u *Core* komponenti ili pluginima?
    Odgovor: Svejedno je. Možda je lakše da to odraditi u Core komponenti, ali izbor prepuštam Vama.
92. **Pitanje:** Da li BirdView/TreeView može da se razvije u pluginu za vizualizaciju?
    Odgovor: Ne! Plugini za vizualizaciju odnose se na centralni prikaz (*MainView*) i tiču se različitog nivoa detaljnosti prikazanih podataka (prost ili složen). Ono što možete uraditi jeste da razvijite poseban plugin za BirdView, odnosno TreeView, ali nije obavezno.
93. **Pitanje:** Da li je obavezna upotreba obrazaca?
    Odgovor: Ne! Postoji šansa da ćete tokom odbrane dobiti pitanje da li je neki od obraza mogao biti primenjen za rešavanje nekog od problema koji se javio tokom izrade projekta, i na koji biste to način uradili (Dajete ukratko objašnjenje, ali ne kodirate na licu mesta).
94. **Pitanje:** Da li možemo sa neta naći neke izvorišne podatke ili ih moramo generisati?
    Odgovor: Kako god Vam je lakše.

95. **Pitanje:** Da li graf treba da ima neku strukturu, tipa stabla ili tako nesto ili je u redu da budu samo razbacani cvorovi?
    Odgovor: Nisam siguran o kom prikazu je reč. Izgled grafa zavisiće od korišćenog (D3) layouta. Kad razvijate aplikaciju, razmišljajte i kao korisnik. Šta biste Vi očekivali da vidite i šta je Vama korisnije.
96. **Pitanje:** Zanima me da li možemo samo doraditi onaj html dokument koji je već u D3Core komponenti iz vežbi 9, i time završiti plugin za vizuelizaciju. Jer koliko sam razumio taj plugin za vizuelizaciju treba biti u posebnom folderu, a na vežbama ste rekli da je dovoljno samo da doradimo taj html fajl.
    Odgovor: Sve materijale koji su dostupni na vežbama možete iskoristiti. Vodite samo računa o tome da ih morate prilagoditi kako to specifikacija i FAQ nalažu. Morate referencirati sav eksterni kod (link do *stackoverflowa*, materijala sa vežbi, itd.), a takođe i razumeti kako kod funkcioniše.
97. **Pitanje:** Naš tim ima pitanje oko XML formata. Da li je u redu da fajl ima strukturu kao fajl na ovom linku https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms762271(v=vs.85)?fbclid=IwAR3LNwE-hGTv-4XuysnChuD3hV9APfgmqBa6mYnBp_XSR6e0cnpHNDLm3oc ? Problem nam predstavlje što se svaki tag učita kao čvor stabla, ali po strukturi smo češće pronalazili ovakve fajlove nego fajlove sa atributima.
    Odgovor: Priloženi fajl ima odgovarajuću strukturu, ali bi mogao biti većeg obima (više čvorova). S obzirom na to da je XML stablo sastavljeno od čvorova koji odgovaraju tagovima, XML parseri rade na način kako ste Vi rekli, tako to da je to u redu. Pokušajte da nađeti još neki XML (ili čak i HTML) koji bi imao više atributa unutar tagova.
98. **Pitanje:** Ako imamo xml fajl kao sto se nalazi u prilogu, da li je dobra logika da imamo cvorove za drzave i njihove susede, a da npr. rank bude atribut same drzave, a to bi se radilo logikom da tag koji nema neke dodatne atribute samo bude dodat u listu atributa njegovog roditeljskog taga(cvora)?
    Odgovor: Ovakav način je racionalan i mogao bi se primeniti i na ostale tipove XML dokumenata.
99. **Pitanje:** Kako da generišemo JSON reprezentaciju ciklučnog grafa kao što to radi XStream biblioteka?

Odgovor: Trebalo bi da se može iskoristiti ista biblioteka. Pogledati primer sa dokumentacije.

100. **Pitanje:** Kako da *inplace*-ujemo string koji predstavlja HTML u Django template
Odgovor: Trebalo bi da ovako nešto može poslužiti.

101. **Pitanje:** Samo da potvrdim da li sam dobro skontala iscitavanje xml fajla.Znaci mi za svaki tag formiramo cvor i dodajemo u listu djece, a za atribute posmatramo vrijednosti u okviru taga npr. id u slucaju <user id="1">Marko</user>. A vrijednost Marko iscitam kao atribut name klase cvor. Da li je to u redu?
Odgovor: Nisam siguran da razumem sve u potpunosti, ali ako mislite na sledeće:

Node
name: Marko,
- id: 1.

onda je to u redu. Ne mora nužno biti name atribut, može i content, value, šta god ima smisla.

102. **Pitanje:** Samo me zbunilo jer sam mislila da ako imamo <user><id>1</id></user> da id moramo ucitati kao atribut,ali vidjeh na pitanjima da treba biti dijete.Pa sam samo htjela da potvdim.
Odgovor: I jedna i druga opcija je u redu. Tagove koji nemaju decu, možete gledati samo kao atribute, dok tagove koji imaju podstablo, možete posmatrati kao čvorove grafa.

103. **Pitanje:** Da li bird i tree view treba prikazai u isto vreme sa main grafom ili korisnik bira koju će vrstu prikaza?
Odgovor: Sva trei prikaza (main, bird, tree) moraju biti istovremeno prikazana korisniku. Bira se nivo složenosti glavnog prikaza (plugin za prost, odnosno složen prikaz).

104. **Pitanje:** U jednom od odgovora na pitanje stoji da je string reprezentacija HTML-a umetnuta u django template povratna vrednost plugin-a za vizuelizaciju. Da li to znači da unutar neke funkcije tog vizuelnog plugin-a mi napišemo baš HTML kod unutar navodnika i onda to prosledimo django template engines funkciji sa još nekim parametrima i to što se dobije, taj template je povratna vrednost plugin-a?
Odgovor: Kao što ste rekli, plugin treba da generiše HTML i vrati ga core aplikaciji. Predloženi način jeste da generišete string koji će imati

HTML tagove. U zavisnosti od ulaza u plugin, HTML string varira (nije *hardcode*-ovan).

105. ~~**Pitanje:** da li ce biti jos jedan rok za projekat iz Sok-a~~
~~**Odgovor:** Biće krajem avgusta, početkom septembra. Osvojeni bodovi će se množiti faktorom 0.8 (možda čak i manjim).~~

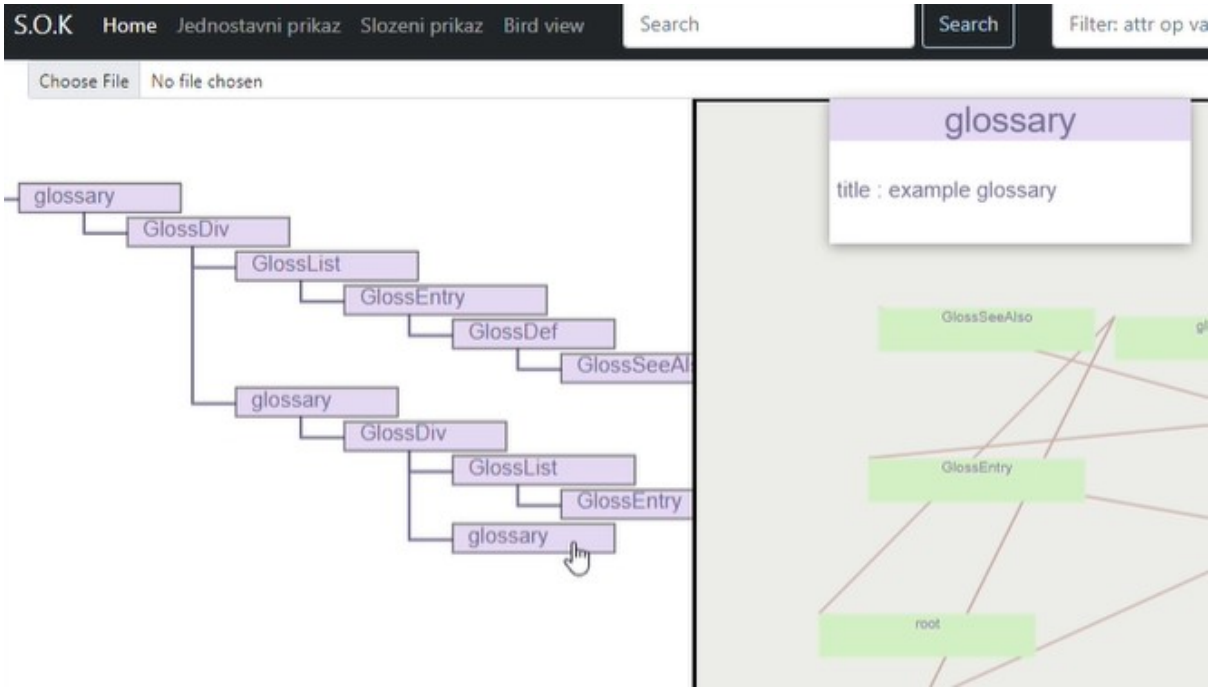
106. **Pitanje:** Ako postoji ciklus između čvora 1 i čvora 2, kako to prikazati u TreeView? Kako sprečiti rekurziju.
Odgovor: Kad korisnik zatraži otvaranje čvora, dovući njegov sadržaj (ne morate slati poseban zahtev na server, dovoljno je iščitati iz nekog JS objekta) i dinamički napraviti HTML za njega. Sledi i prikaz TreeView. Pritiskom na plus bi se mogao dovući sadržaj.

1
2
1
2
+ 1 (Kada se pritisne plus, prikaže se čitav sadržaj čvora 1.)

107. **Pitanje:** Kod main prikaza grafa, ukoliko imamo situaciju da postoji grana A->B i grana B->A, te grane će se na grafu preklopiti i videće se jedna linija. Da li je to problem ili možemo da ostavimo tako?
Odgovor: Ukoliko stignete (nakon što sve ostalo odraite), pokušajte da prilagodite ForceLayout da se povratne grane ne preklapaju.

108. **Pitanje:** Da li tree view i bird view prikaz odgovaraju nivou detaljnosti prikaza u main view, ili postoji samo jedna vrsta prikaza da tree i bird view?
Odgovor: TreeView prikazuje uvek sve informacije o pojedinačnom čvoru, dok BirdView predstavlja prikaz grafa iz ptičje perspektive i služi da prikaže strukturu grafa, te samim tim sadrži manje detalja o pojedinačnim čvorovima.

109. **Pitanje:** U prilogu Vam šaljem prikaz TreeView da li bi ovako prikazana rekurzija bila okej? Prelazom preko poslednjeg prikazanog elementa iz rekurzije ("glossary") se prikazuju njegovi detalji.



Odgovor: Rekurzija je dobro prikazana. Ono što nedostaje TreeView-u jesu informacije o čvorovima koji su otvoreni. Bacite pogled na pitanje 83.

Nastavak: Informacije o čvorovima smo prikazali u prozorčićima koji iskaču kada se mišem pređe preko čvora (prikazan je ljubičasti prozorčić za desne strane na slici koji ima ime čvora i infirmacije o čvoru).

Odgovor: To je meni jasno, samo kažem da informacije u čvoru treba da se prikažu kada se čvor otvori na TreeView-u, a ne samo u prozorčiću sa strane. Znači u GlossList treba da stoji niz jednostavnih atributa, plus složen koji dovodi do otvaranje novog čvora. Kao kada prikažete package explorer. Fajlovisu bi bili prosti atributi, a ugnježedni direktorijumi složeni.

110. **Pitanje:** Dvoumimo se da li je potrebno da oba plugina za vizualizaciju imaju detaljan prikaz. Složen prikaz sam po sebi ima sve attribute, pa ne znam da li je uopšte potreba ta operacija. Selektovanje čvora je potrebno za oba plugina.

Odgovor: Korisnik na intefejsu bira plugin sa odgovarajućim nivoem detaljnosti. Nakon toga, plugin iscrtava ono što je potrebno. S obzirom na to da selekcija čvora na bilo kom prikazu treba da dovede do selekcije na ostalim i prikaza detaljnih informacija o čvoru, ovo bi imalo smisla razviti u Core aplikacije, a ne u pluginu.

111. **Pitanje:** Da li prilikom upotrebe opcije filter korisniku treba prikazati grešku ukoliko je uneta vrednost neodgovarajućeg tipa?

Odgovor: Da! Ako korisnik odabere filtriranje po atributu koji je numeričkog tipa, string vrednost nema baš mnogo smisla.

112. **Pitanje:** Ako razvijamo generičko rešenje, da li je u redu da sve atribute gledamo kao string?

Odgovor: Ne! Prilikom parsiranje, treba pokušati sa prepoznavanjem ostalih tipova (npr. numeričkih, vremenska odrednica-*timestamp*, datum, itd.), a ne samo stringova.

113. **Pitanje:** Da li TreeView treba da se implementira uz pomoć D3.js biblioteke?

Odgovor: Možete koristiti šta god želite. Upotreba gotovih biblioteka za iscrtavanje podataka u vidu stabla, odnosno prikaz *package explorer*-a je dozvoljena. Šta god Vama najviše odgovara i što Vam može olakšati posao.

114. **Pitanje:** Imam par pitanja u vezi sa parsiranjem cikličnih grafova iz JSON datoteke. Da li je validno pretpostaviti da svaki "čvor" ima ID u ključevima kako bih mogla ispratiti da li se čvor već pojavio? I u tom slučaju, pošto ne mogu da generišem svoje identifikatore za čvorove (jer koristim one koji su učitani iz datoteke), šta bi se desilo ako postoji čvor sa ključem koji pokazuje na listu. Tokom učitavanja acikličnih grafova sam implementirala da ta lista postaje novi čvor, i njeni elementi su njena deca, odnosno čvorovi na koje ona pokazuje. Da li je u redu da ako naiđem na par ključ vrednost kao što su sledeći "lista" : [123, "pera", {neki json}] oni ostaju samo atributi čvora u kom se nalaze, bez proširenja?

Odgovor: Kao što nekoliko pitanja ukazuje u FAQ-u, JSON dokument treba generička parsirati (u najgorem slučaju u stablo). Nakon toga pribegavate traganju za ciklusima. Možda ih ima, a možda i nema. Nema pretpostavke unapred da nema ciklusa, već se do tog zaključka dolazi tokom parsiranja (ne mora bukvalno tokom rada JSON parsera, možete i naknadno drugi put proći kroz izgrađene Python objekte). Kako ćete tražiti cikluse, prepuštam Vama. Na neki način morate utvrditi da se o istom JSON podobjektu radi.

Jedan od načina za prepoznavanje ciklusa bi bio da tražite ID-eve i reference atribute, drugi bi bio da pokušate da pronađete dva JSON podobjekta sa istih nekoliko atributa i odgovarajućim vrednostima. Setite se relacionih baza i definicije primarnog ključa. Ako je id atribut zauzet,

tj. već postoji u dokumentu, uvek možete izgenerisati neki novi surogatni id (npr. __sok__id ili kako god).

115. Pitanje: Imamo grešku kod učitavanja plugina i mučimo se već neko vreme.

```
Traceback (most recent call last):
  File "/Applications/Xcode.app/Contents/Developer/Library/Frameworks/Python3.framework/Versions/3.8/lib/python3.8/threading.py", line 932, in _bootstrap_inner
    self.run()
  File "/Applications/Xcode.app/Contents/Developer/Library/Frameworks/Python3.framework/Versions/3.8/lib/python3.8/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "/Users/nikolina/Downloads/pythonProject/inter/lib/python3.8/site-packages/django/utils/autoreload.py", line 64, in wrapper
    fn(*args, **kwargs)
  File "/Users/nikolina/Downloads/pythonProject/inter/lib/python3.8/site-packages/django/core/management/commands/runserver.py", line 115, in inner_run
    autoreload.raise_last_exception()
  File "/Users/nikolina/Downloads/pythonProject/inter/lib/python3.8/site-packages/django/utils/autoreload.py", line 87, in raise_last_exception
    raise _exception[1]
  File "/Users/nikolina/Downloads/pythonProject/inter/lib/python3.8/site-packages/django/core/management/__init__.py", line 381, in execute
    autoreload.check_errors(django.setup)()
  File "/Users/nikolina/Downloads/pythonProject/inter/lib/python3.8/site-packages/django/utils/autoreload.py", line 64, in wrapper
    fn(*args, **kwargs)
  File "/Users/nikolina/Downloads/pythonProject/inter/lib/python3.8/site-packages/django/__init__.py", line 24, in setup
    apps.populate(settings.INSTALLED_APPS)
  File "/Users/nikolina/Downloads/pythonProject/inter/lib/python3.8/site-packages/django/apps/registry.py", line 122, in populate
    app_config.ready()
  File "/Users/nikolina/Downloads/pythonProject/inter/lib/python3.8/site-packages/Core-0.1-py3.8.egg/Core/apps.py", line 13, in ready
    self.plugin_prikaz = load_plugins("view.load")
  File "/Users/nikolina/Downloads/pythonProject/inter/lib/python3.8/site-packages/Core-0.1-py3.8.egg/Core/apps.py", line 19, in load_plugins
    p = ep.load()
  File "/Users/nikolina/Downloads/pythonProject/inter/lib/python3.8/site-packages/pkg_resources/__init__.py", line 2450, in load
    return self.resolve()
  File "/Users/nikolina/Downloads/pythonProject/inter/lib/python3.8/site-packages/pkg_resources/__init__.py", line 2456, in resolve
    module = __import__(self.module_name, fromlist=['__name__'], level=0)
ModuleNotFoundError: No module named 'plugin.view'
```

Odgovor: U slučaju da Vam se paketi zovu identično u različitim pluginima, onda je potreban `namespace_packages` parametar setup funkcije u kombinaciji sa `__import__('pkg_resources').declare_namespace(__name__)` u `__init__.py`. Pogledati primer *StudentskaSluzbaComponent* sa vežbi 6.

Druga opcija je da se paketi u različitim pluginima različito zovu.

116. **Pitanje:** Imam nedoumicu vezanu za funkcionisanje i integraciju plugina za vizualizaciju. Da li kod poziva za za vizualizaciju, treba da se vrati string koji sadrzi html koji nalijepimo na trenutni html prikaz ili se kreira novi html fajl koji prosiruje pocetni i onda se on prikazuje?

Odgovor: Ovo prvo.

117. **Pitanje:** U odgovoru na pitanje 83 iz FAQ dokumenta napisali ste da je BirdView graf koji se već nalazi u MainView samo iz ptičije perspektive, a meni ste u odgovoru napisali da BirdView sadrži manje detalja o pojedinačnim čvorovima. Moje pitanje je da li onda treba praviti

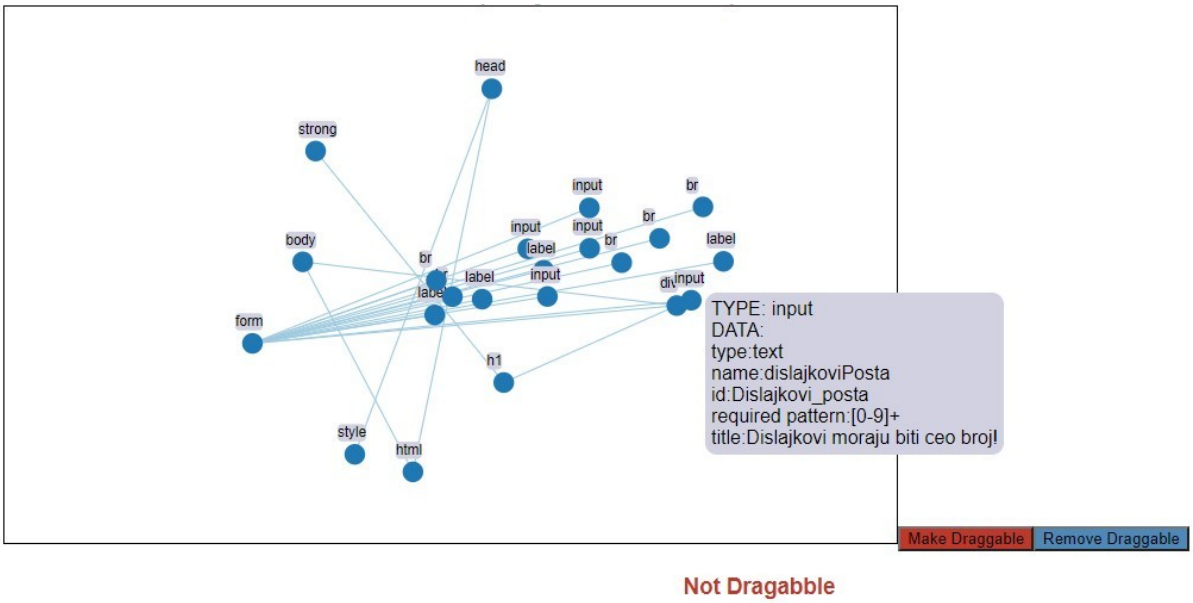
poseban prikaz za BirdView, ili se može u nekom javascript fajlu u BirdView ubaciti sadržaj iz MainView-a i samo skalirati da bude prikazan ceo graf?

Odgovor: I jedna i druga opcija su u redu (ukoliko su izvodljive).

118. **Pitanje:** Da li je u redu da nakon izvršene pretrage ili filtriranja grafa, čvorovi koji odgovaraju unetim parametrima budu samo prikazani drugom bojom?

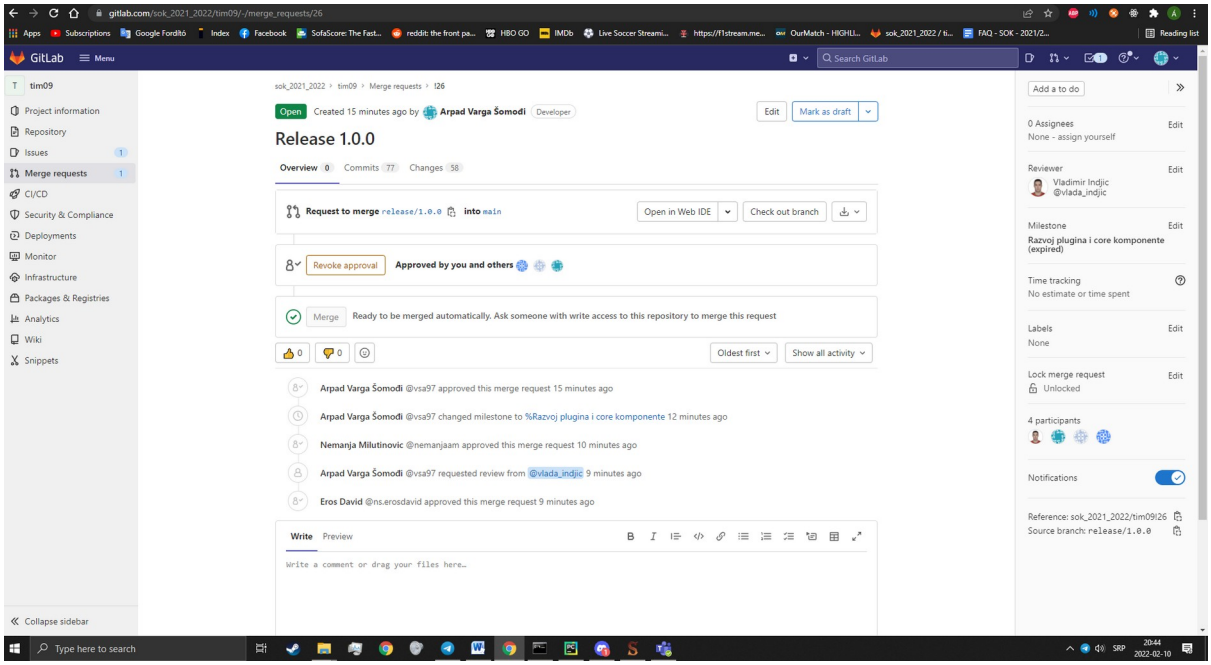
Odgovor: Odgovor je nažalost ne. Potrebno je formirati podgraf shodno upitu unutar Core komponente, koji će odgovarajući plugin za vizuelizaciju prikazati.

119. **Pitanje:** Da li je u redu, da složeni prikaz bude rezultat mouseover funkcije ili čvor mora biti veliki, sa ispisom podataka? I da li mogu da imam opciju da uključim/isključim pomjeranje čvorova?



Odgovor: Odgovor je nažalost odričan. Kada korisnik odabere složen prikaz, tada je potrebno čvor prikazati sa svim informacijama. Pri čemu mouseover događaj i dalje treba da dovede do prozorčića koji ste prikazli na slici. Opcija pomeranja čvora se ne sme isključiti. Glavni prikaz mora imati funkcionalnosti kao što su: pan, zoom, drag and drop.

120. **Pitanje:** Hteli smo da mergujemo release granu u main, ali piše da nemamo write access(??) i ne možemo, samo approve. Da li je to greška ili je tako namenjeno?



Odgovor: Mislim da je Gitlab od skoro uveo da samo maintainer/owner može da odradi merge request na master. Assignujte MR (merge request ili pull requst kako se negde zove) meni i zahtevajte review, pa ću to prihvatiti. U krajnjem slučaju i da ostane na develop grani, nije problem.

121. **Pitanje:** Da li je moguće da u istom timu budu članovi koji imaju različite ambicije prema broju bodova? Na primer jedan član tima želi da radi za ocenu 10, a drugi za ocenu 8. Da li će to uticati na broj bodova, ili se svačiji rad ocenjuje zasebno ?

Odgovor: Na projektu se pored saradnje studenata gleda i pojedinačno zalaganje. Neretko se dešavalo da studenti dobiju različit broj bodova shodno zalaganju.

122. **Pitanje:** Hteo bih da se upišem u tabelu neraspoređenih, pa imam pitanje u vezi GitLab naloga. GitLab trenutno ima Free Trial 30 dana, plaćenu opciju ili Education, pa me interesuje koji tip naloga otvaram? Ukoliko je Education, da li to otvaram sa @uns mejl adresom ili dobijam nalog od fakulteta? Hvala.

Odgovor: Odaberi Education opciju za koju preporučujem upotrebu uns maila. Moguće da možeš i drugu mail adresu da koristiš, ali mislim da će ti tražiti transkripte ocena kao verifikaciju da si student.

123. **Pitanje:** Na koji način treba da se importuje D3.js biblioteka u naš projekat?

Odgovor: Ovo možete uraditi na dva načina. Prvi je da iz `src` atributa `script` taga direktno gađate zvaničnu stranicu D3 biblioteke. Mana ovog načina je što možda neće dobro raditi ukoliko ste iza proxy-a. U tom slučaju bi najbolje bilo da prilikom odbrane koristite sopstveni hotspot. Drugi način jeste da preuzmete D3.js biblioteku i u projektu referencirate lokalnu kopiju (kao što je to urađeno na vežbama). U tom slučaju bi trebalo da u .gitignore stavite putanju do lokalne verzije biblioteke kako se ista ne bi pojavila na Vašem GitLab repozitorijumu i bespotrebno usporavala preuzimanja sadržaja sa istog.

124. **Pitanje:** Da li se kontrolne tačke ocenjuju?

Odgovor: Kontrolne tačke se ne ocenjuje direktno, ali indirektno svakako ulaze u ocenu. Naime, rad tokom kontrolnih tačaka nosi ukupno 6 bodova i tiče se stavke *GitLab aktivnost* sledeće tabele. Da biste osvojili svih 6 bodova, neophodno je da svaka kontrolna tačka bude dobro isplanirana, tako što ćete se kao tim dogovoriti šta je potrebno uraditi tokom trajanja iste. Ovo dokumentujete u formi issue-a koje dodeljujete pojedinačnim članovima tima na izradu. Kontrolna tačka se smatra uspešno završenom, ukoliko su svi issue-a koji su planirani uspešno završeni i zatvoreni pre kraja roka definisanog *Milestone*-om. Ovo dovodi do toga da *GitLab* prikaže da je *Milestone* završen 100%.

125. **Pitanje:** Šta ako isplaniramo da ne uradimo ništa tokom kontrolne tačke?

Odgovor: Kontrolna tačka koja ima 0 issue se smatra neuspešnom, što će dovesti do gubitka određenog broja bodova na samoj odbrani projektnog zadatka.

126. **Pitanje:** Koliko termina za odbranu projekta će biti?

Odgovor: Ukupno 3. Prvi termin biće u janurasko-februarskom ispitnom roku (najverovatnije 04-05.02.2023) i u tom roku se može osvojiti maksimalnih 60 bodova. Naredni termin odbrane biće u junsko-julskom roku i osvojenih bodovi će se skalirati faktorom 0.9. Poslednji termin odbrane u ovoj iteraciji kursa biće u avgustovsko-septembarskom roku i osvojeni bodovi biće množeni faktorom 0.8

127. **Pitanje:** Da li je obavezna upotreba D3 biblioteke?

Odgovor: Nije, ali se preporučuje. Ukoliko ste upoznati sa nekom drugom bibliotekom za vizualizacije, možete je koristiti. Vodite računa da u tom slučaju ne možete očekivati preveliku tehničku pomoć od asistenata.

128. **Pitanje:** Da li RDF (Resource Description Framework) može biti jedno od izvorišta podataka?

Odgovor: RDF je validan izvorišni podatak, pod uslovom da nije u obliku RDF-XML-a a već jedan od izvorišnih plugina učitava podatke iz XML-a. U tom slučaju, morate koristiti neku drugu reprezentaciju RDF-a (npr. Turtle, i sl).

129. **Pitanje:** Kako tačno treba da radi pretrage? Da li se čvorovi pretražuju po nazivima atributa ili po vrednostima atributa?

Odgovor: Nakon unošenja upita u formi teksta na klijentskom delu aplikacije, na serverskom delu aplikacije neophodno je pronaći sve čvorove koji sadrže uneti tekst (operacija *contains*). Uzimajući u obzir veze između ovih čvorova neophodno je izgraditi podgraf koji je rezultat pretrage. Čvor zadovoljava kriterijum pretrage ukoliko bilo koji od njegovih atributa u nazivu sadrži uneti tekst, kao i ako vrednost bilo kog atributa tog čvora sadrži uneti tekst. Na ovaj način mogu se izdvajati čvorovi po nazivima, odnosno vrednostima atributa.

130. Pitanje: