Spirala 4

Napravite sljedeće modele koristeći sequelize. Napravite MySQL bazu sa imenom wt2018, username i password za pristup bazi neka budu root, baza neka se nalazi na standarnom portu.

Model student (datoteka student.js):
- id
- imePrezime : Sequelize.STRING
- index : Sequelize.STRING, unique

Model godina (datoteka godina.js):
- id
- naziv : Sequelize.STRING, unique
- nazivRepSpi : Sequelize.STRING
- nazivRepVje : Sequelize.STRING

Model zadatak (datoteka zadatak.js):
- id
- naziv : Sequelize.STRING
- postavka : Sequelize.STRING

Model vjezba (datoteka vjezba.js):
- id
- naziv : Sequelize.STRING, unique
- spirala : Sequelize.BOOLEAN

Veze između modela (datoteka db.js):
student -- *više na jedan* -- **godina**, fk studentGod, as studenti
godina -- *više na više* -- **vjezba**, mt **godina_vjezba**, fk idgodina i idvjezba, as godine i vjezbe
vjezba -- *više na više* -- **zadatak**, mt **vjezba_zadatak**, fk idvjezba i idzadatak, as vjezbe i zadaci
fk - foreign key, as - alias, mt - međutabela

Modeli se trebaju kreirati pri pokretanju node aplikacije, samo ako već nisu kreirani. Ako modeli nisu ispravno kreirani neće se bodovati sljedeći zadaci (osim zadatka 3.b).

Zadatak 1. [2b] Prepravite zadatke 2,3,4 i 5 sa prošle spirale tako da koriste ovu bazu podataka.

Zadatak 2.a [0.5b] Implementirajte funkcionalnost dodavanja vježbe/spirale. Ako se pošalje POST zahtjev iz forme fPostojeca sa podacima sGodine i sVjezbe na url http://localhost:8080/addVjezba kreira se veza između godine sa id-em sGodine i vježbom sa id-em sVjezbe. Nakon dodavanja vratite na stranicu addVjezba.html

Zadatak 2.b [0.5b] Implementirajte funkcionalnost dodavanja nove vježbe/spirale. Ako se pošalje POST zahtjev iz forme fNova sa podacima sGodine, naziv i spirala kreira se nova vježba sa podacima naziv i postavlja joj se polje spirala na true ili false u zavisnosti od vrijednosti spirala iz forme. Nakon kreiranja nove vježbe ona se povezuje sa godinom sa id-em sGodine. (Dodavanje zadataka na vježbu ne radite u ovom zadatku) Nakon dodavanja vratite na stranicu addVjezba.html

Zadatak 2.c [1b] Dio iz forme fNova koji se odnosi na dodavanja zadataka na vježbu izdvojite u novu formu (na istoj stranici). U novoj formi fPoveziZadatak treba da se nalazi

- select - name: sVjezbe
- select - name: sZadatak
- input submit - name: dZadatak

Kada se klikne na dugme dZadatak zadatak se dodaje na vježbu. Šalje se POST zahtjev na url http://localhost:8080/vjezba/:idVjezbe/zadatak. U select tagu sZadatak prikažite sve zadatke koji već nisu dodjeljeni odabranoj vježbi iz sVjezbe. Ne treba biti moguće u formi odabrati zadatak koji je već dodan vježbi. Nakon dodavanja zadatka vratite na stranicu addVjezba.html

Zadatak 3.a [2b] Stranicu addStudent ispravite tako da ima jednu formu sa poljima

- select - name: sGodina
- input text - name: key
- input text - name: secret
- button - value: Učitaj
- input button- value: Dodaj, disabled

- Kada se klikne na dugme učitaj kreira se instanca modula BitBucket (ako već nije kreirana) i poziva se metoda modula BitBucket.ucitaj(nazivRepSpi,nazivRepVje,callback)
- Nakon učitavanja liste studenata omogućen je klik na dugme dodaj
- Klikom na dugme dodaj pravi se POST zahtjev putem AJAX-a na http://localhost:8080/student i šalju se podaci u JSON formatu {godina:X,studenti:[{imePrezime: Y, index: Z},...]}. Na serveru ukoliko ne postoji student sa imenom i indexom dodaje se i onda se upisuje na godinu koja ima id X. Ako student već postoji tada se samo upisuje (student postoji ukoliko već postoji neki student sa istim indexom u bazi). Nakon obrađenog zahtjeva vratite JSON kao odgovor {message:"Dodano je N novih studenata i upisano M na godinu NAZIVGODINE"}. Tekst poruke ispišite u web browseru kao alert. Broj N predstavlja broj studenata koji nisu postojali u sistemu i koji su dodani, M predstavlja ukupan broj studenata koji su upisani na godinu.

Zadatak 3.b [1.5b] Napravite modul BitBucket (datoteka BitBucket.js) sa **konstruktorom** koji kao parametre prima key i secret. Kada se pozove konstruktor pravi se AJAX zahtjev na Bitbucket api kako bi se dobio token. Rezultat (token) sačuvajte kao privatni atribut modula u vidu promise-a. Kao rezultat poziva konstruktora dobijate objekat sa metodom ucitaj.

Metoda **ucitaj** ima parametre nazivRepSpi,nazivRepVje i callback funkciju koju ćete pozvati kada dobijete rezultat. Ona koristi token kojeg ste dobili kao rezultat poziva konstruktora. Kada se pozove metoda ucitaj pravi se **samo jedan** AJAX zahtjev na Bitbucketapi koji će dohvatiti sve repozitorije koji su podijeljeni sa vama a koji imaju naziv koji počinje sa nazivRepSpi ili sa nazivRepVje. Za filtriranje repozitorija obavezno koristite BitBucket api filtering. Kada dobijete listu svih repozitorija iz njih izdvojite studente bez ponavljanja. Podatke o studentu dobijate u odgovoru Bitbucket api-a. Ime vlasnika repozitorija predstavlja ime studenta, a index je uvijek 5 posljednjih znakova naziva repozitorija. Kada učitate sve studente pozovite callback i proslijedite niz studenata kao parametar. Metoda ucitaj baca izuzetak ukoliko token nije ispravan ili ako nije moguće dohvatiti listu repozitorija.

Primjer korištenja modula:

```
var bbucket = new BitBucket("neki key", "neki secret");
function ispisi(x){console.log("Lista studenata:\n"+JSON.stringify(x));}
//sljedeći poziv se treba nalaziti u try-catch bloku:
bbucket.ucitaj("wtv18","wtProjekat18",ispisi);
```

Napomena 1. Zadatak 3.a možete uraditi i bez zadatka 3.b u tom slučaju stavite da metoda Bitbucket.ucitaj vraća hardkodiranu listu studenata.

Rok za rad na spirali 4 je do 14.01.2018 do 23:59. Nakon roka urađena spirala se treba nalaziti na branchu spirala4 na vašem repozitoriju. Review spirale 4 će biti bodovan sa 1b.