Studentska služba

Realizovati servis za studentsku službu. Zadatak servisa je da prosleđuje ocene od profesora ka studentima. Studenti se prijavljuju studentskoj službi i čekaju notifikaciju o rezultatu ispita. Profesori nakon završenog ispita studentskoj službi šalju ime predmeta i listu koja sadrži indeks studenta i ocenu.

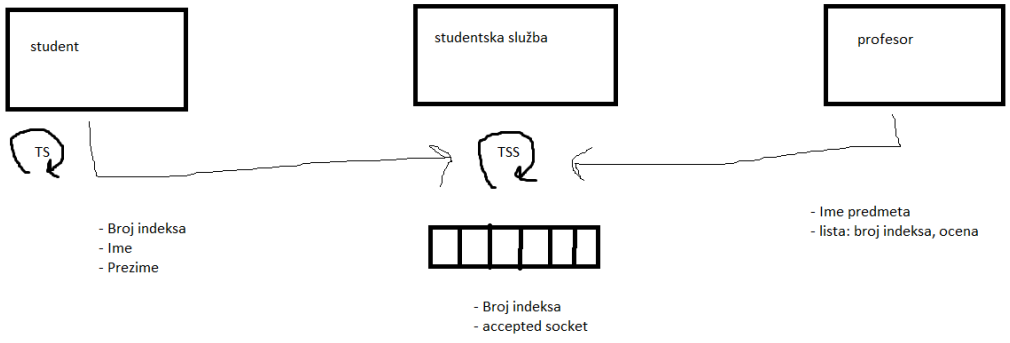
Gore je naveden tekst zadatka, u sledećim sekcijama je opisan tok izrade dizajna

Prikupljanje informacija koje su potrebne za dizajn

1. Definisati celine i broj komponenti sistema što će da predstavlja broj konzolnih aplikacija -> studentska služba, student, profesor
2. Definisati prijavljivanje studenta
    a. Koliko proces 'student' treba da živi -> sve dok se ne odjavi
    b. Koji podaci treba da se šalju prilikom prijave -> broj indeksa, ime, prezime
    c. Kako će biti implementirano konektovanje studenta na studentski servis -> student pošalje strukturu sa svojim podacima, na servisu se ta struktura stavi u listu i u strukturu se doda accepted socket
3. Definisati slanje rezultata profesora
    a. Koliko proces 'profesor' treba da živi -> pošalje rezultat i ugasi se
    b. Koji podaci treba da se šalju studentskom servisu-> ime predmeta, lista koja sadrži indeks studenta i ocenu
    c. Koji podaci treba da se šalju studentu -> predmet, ocena
    d. Kako će biti implementirano slanje ocene studentima na studentskom servisu -> prolazi se kroz listu studenata koju je profesor poslao, nađe se student u listi studenata na studentskom servisu, preko send (accepted soket) studentu se pošalju rezultati

Prva verzija dizajna

Prva verzija dizajna je dovoljna za ocenu 8. Potrebno je priložiti sliku u Visio alatu i opis slike.



Kratak opis: Studentska služba ima thread (TSS) koji je zadužen za prihvatanje i obradu zahteva od klijenata Student i Profesor. Prilikom prijave, student šalje studentskoj službi odgovarajuću strukturu, i otvara thread TS koji će da prima ocene od studentske službe. Studentska služba nakon prijema zahteva, dodaje zahtevu accepted socket i upisuje studenta u listu. Profesor takođe šalje studentskoj službi odgovarajuću strukturu koja se obrađuje u threadu TSS (obrada opisana u 3. d).

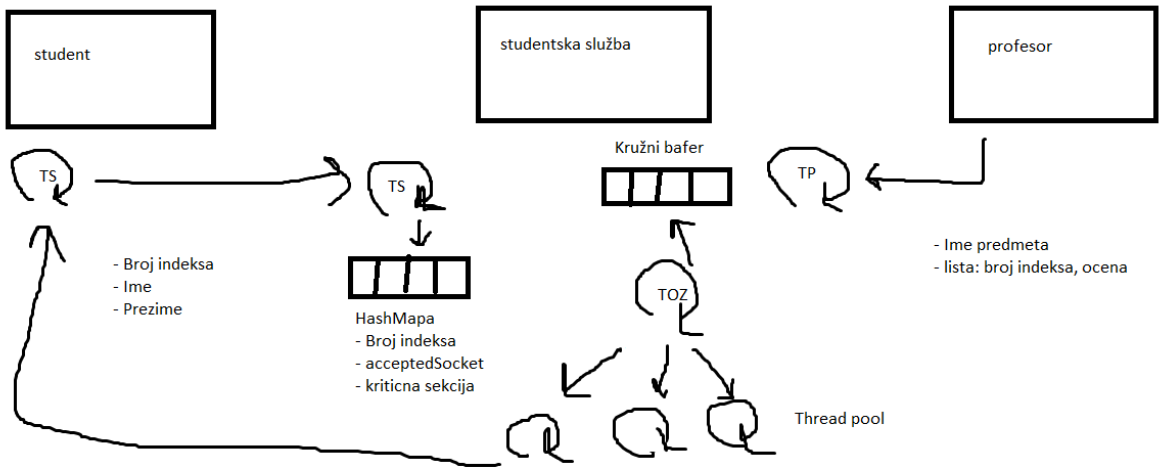
Detektovanje uskih grla

Što se tiče procesa Student i Profesor, mogu biti implementirani bilo kako, ne baziramo se na njih. U nastavku biće navedena potencijalna uska grla procesa studentski servis i saveti kako ta uska grla mogu da se otklone dizajnom.

- Sve obrade se nalaze u threadu TSS, što može dovesti do problema ukoliko profesor pošalje dugačku listu studenata jer će prilikom obrade te liste biti onemogućeno prijavljivanje novih studenata kao i obrada zahteva drugih profesora.
    - Može se razdvojiti na dva threada, jedan da sluša i obrađuje zahteve studenata (TS), drugi zahteve profesora (TP)
    - Zahteve profesora bi bilo dobro upisivati u proširivi kružni bafer, ovde bismo imali producer/consumer šablon. TP thread bi samo upisivao zahteve, a drugi thread (TOZ) bi skidao zahteve sa kružnog bafera i obrađivao ih.
    - Imati u vidu da send funkcija može da potraje, tako da ona treba da se izdvoji u poseban thread. Ovde imamo dve varijante:
        1. Ukoliko na primer ima puno zahteva od profesora, može da se napravi thread pool koji će da sadrži već kreirane threadove i da TOZ ne obrađuje zahtev već ga samo skine sa kružnog bafera i prosledi threadu iz thread poola. Ovako možemo više poruka u isto vreme da obradimo. Paziti samo na sinhronizaciju pristupa socketima, jer može da se desi da se iz dva threada pokuša send na isti socket.
        2. Thread TS koji obrađuje prijavu studenata može i da kreira thread za svakog studenta koji će da bude aktiviran iz threada TOZ. Ovaj pristup nije dobar ukoliko imamo puno studenata, jer ne želimo puno threadova. Ovde za aktivaciju threada koristiti semafor koji će da bude prosleđen threadu kroz parametre.
- Lista kao struktura podataka nije optimalna, jer u slučaju ovog zadatka imamo najviše pretrage po indeksu tako da bi bilo bolje koristiti HashMapu

Druga verzija dizajna

Ovde je dat primer dizajna za ocenu 10, prva varijanta iz prethodnog poglavlja.



Kratak opis: Studentska služba ima dva glavna threada: TS koji je zadužen za prihvatanje i obradu zahteva od klijenata Student i TP koji je zadužen za pijem zahteva od klijenta Profesor. Prilikom prijave, student šalje studentskoj službi odgovarajuću strukturu, i kod sebe otvara thread TS koji će da prima ocene od studentske službe. Studentska služba nakon prijema zahteva od studenta, dodaje zahtevu accepted socket, kritičnu sekciju za kontrolu pristupa socketu i upisuje studenta u strukturu HashMap. Profesor takođe šalje studentskoj službi odgovarajuću strukturu koja se prilikom prijema u threadu TP smešta u proširivi kružni bafer. Thread TOZ je zadužen za čitanje zahteva iz kružnog bafera. Nakon što se zahtev pročita, uzme se slobodan thread iz thread pool-a u kom će zahtev biti obrađen.