Kreirati konzolnu aplikaciju koja će omogućiti komunikaciju sa Hyperledger Fabric mrežom, kroz pozive chaincode funkcija.

Aplikacija će podržati funkcionalnosti:

- enroll / login korisnika
- upit (query) chaincode-a
- izazivanje (invoke) chaincode-a

Konzolna aplikacija mora da koristi SDK (NodeSDK, JavaSDK, GoSDK…) za rad sa chaincode-om.

Prilikom izrade projekta je neophodno koristiti 2.2.6 ili noviju verziju Hyperleder fabric mreže.

Projekat je potrebno raditi u grupama od dvoje. Moguće je raditi i samostalno ali obim posla ostaje isti.

Potrebno je da Fabric blockchain obezbedi praćenje i poslovnu logiku kojom se obezbeđuju funkcionalnosti opisane u daljem tekstu.

Implementirati sistem za trgovanje proizvoljnim dobrima na mreži. Pored trgovca u sistemu se čuvaju podaci o tipovima trgovaca, proizvodima, korisnicima i računima. Nabrajanja koja su data u specifikaciji projekta su minimalni zahtevi za polja struktura. Dozvoljeno i poželjno je proširivanje specifikacije.

Stanja koja se čuvaju na mreži modelovati na sledeći način.

Trgovac:

- idTrgovca (jedinstveni identifikator)
- tip trgovca
- PIB
- proizvodi dostupni za prodaju
- računi (prodati proizvodi)
- stanje na računu

Treba osmisliti nekoliko **tipova trgovaca** i čuvati ih kao stanje na blokčejnu. Primer tipa trgovca bi bio: supermarket, auto delovi itd.

Proizvod:

- šifra (id)
- ime
- rok trajanja (ako ga ima)
- cena
- količina

Korisnik:

- idKorisnika (jedinstveni identifikator)
- ime
- prezime
- adresu elektronske pošte
- proizvodi koje je korisnik kupio (računi)
- stanje na računu

Račun:

- id
- trgovac
- korisnik
- proizvod
- datum

Prethodno opisane strukture se upisuju u WorldState, prilikom INIT funkcije chaincode-a, ili neke druge invoke funkcije koja će postaviti početno stanje tako da sadrži minimum 2 trgovca i svaka ima po bar 2 proizvoda. Napraviti i nekoliko korisnika.

Funkcionalnosti koje će chaincode obezbediti jesu:

- Unos novog trgovca
- Dodavanje jednog ili više proizvoda trgovcu
- Unos jednog ili više korisnika
- Kupovinu proizvoda od strane korisnika
    - Kupovina je moguća samo ukoliko korisnik koji kupuje ima dovoljno sredstava na računu da plati proizvod. Nakon kupovine se količina priozvoda umanjuje (ukoliko dostigne nula, proizvod se uklanja). Račun proizvoda se dodaje korisniku i trgovcu.
- Uplata novca na račun trgovca ili korisnika
- Query funkcije koje omogućuju pretragu proizvoda po imenu, šifri, tipu trgovca, ceni. Pretragu je moguće izvršiti po svakoj posebno kategoriji ili bili kojoj kombinaciji.
- Potrebno je testirati sve implementirane funkcionalnosti kroz pozive funkcija u konzolnoj aplikaciji. Neophodno je koristi **shell** skripte
- Potrebno je obraditi sve moguće greške (ukoliko ne postoji korisnik ili proizvod sa zadatim ključem u WorldState-u, nedovoljna količina sredstava itd.)

Zahtevi za projektni zadatak:

- ❏ Aplikacija za interakciju sa chaincode-om mora imati opciju rada sa bar dva sertifikata. Tj. kroz aplikaciju je moguće logovanje kao dva različita korisnika Hyperledger mreže koji pripadaju različitim organizacijama.
- ❏ Mapiranje organizacija na specifikaciju projekta je slobodno za interpretaciju. Nije obavezno praviti novog korisnika u Hyperledger fabriku za svakog novog korisnika ili trgovca u konzolnoj aplikaciji. Dozvoljeno je raditi sa više trgovaca i korisnika koristeći jedan Hyperledger Fabric sertifikat.

- ❏ Kreiranje Fabric mreže sa dva kanala
  - specifikacija uključuje tri organizacije sa po tri peer-a
    - ➔ generisanje potrebnih kripto materijala
    - ➔ specifikacija docker kontejnera za elemente Fabric mreže
  - kanalu mora biti pridružen bar 1 orderer
  - svi peer-ovi se nalaze na oba kanala
  - kreiranje svih preostalih neophodnih učesnika u mreži kao i njihove kriptomaterijale (CA, Orderer)
    - ➔ obavezno je koristi CouchDB
    - ➔ Neophodno je napisati par dodatnih bogatih upita koji pokazuju prednosti CouchDB-a i objasniti na odbrani zašto su baš ti upiti zanimljivi i kako bi se ista stvar uradila u levelDB.
    - ➔ svi peer-ovi imaju ulogu endorser-a i commiter-a

- ❏ U world state-u se čuvaju objekti u JSON obliku, konzolna aplikacija takođe vraća JSON podatke.
  - aplikacija koja koristi SDK treba da omogući:
    - ➔ enroll – login
    - ➔ query
    - ➔ invoke

Napomene:

- Preporučuje se pisanje chaincode-a u Golang-u.
- Preporučuje se upotreba dokera.
- OrderingService koristi RAFT konsenzus algoritam. Potrebno je da commit bloka radi tako da se bar 2 transakcije nađu u bloku ili na svaki sekund, šta god se desi prvo.
- **Obavezno** je koristiti CouchDB.
- Imenovanje elemenata mreže (MSP, Org, Peer, Channel, Orderer itd) je ostavljeno studentu, preporučuje se imenovanje koje je pokazano na vežbama.
- Specificiranje pravila prihvatanja transakcije, odnosno *endorsement policy* za chaincode je ostavljen studentu da specificira kako želi.
- **Obavezno** je koristi CA (Certificate Authority) i obavezno je testirati sve funkcionalnosti.
- **Obavezno** je napisati **.sh** skripte za sve potrebne operacije testiranja i pokretanja mreže.
- **Obavezno** je obezbediti GitHub pristup asistentu (GitHub username NebojsaHorvat). Takođe je obavezna ravnomerna raspodela posla koja se vidi kroz GitHub istoriju.
- **SVE ŠTO NIJE EKSPLICITNO NAPISANO U SPECIFIKACIJI PROJEKTA STUDENT MOŽE IMPLEMENTIRATI PO SOPSTVENOJ INTERPRETACIJI**