Kreirati web 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

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

Prilikom izrade projekta je neophodno koristiti 2.2.6 verziju svih modula Hyperleder fabric mreže.

Projekat je moguće raditi u paru ukoliko se ispoštuju svi dodatni zahtevi označeni plavom bojom.

Aplikacija za interakciju sa chaincode-om, može da koristi samo jedan sertifikat, može i druge, ali je jedan dovoljan **(ukoliko se zadatak radi u paru, potrebno je omogućiti rad sa bar 3 različita sertifikata).**

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

U sistemu inicijalno postoji nekoliko automobila koji su modelovani na sledeći način:

- idAutomobila(jedinstveni identifikator)
- marka
- model
- godiste
- boja
- idVlasnika
- lista kvarova (gde kvar sadrži opis i cenu popravke). Primer kvara je probušena desna prednja guma

Pored automobila u sistemu se čuvaju i podaci o osobama (vlasnik automobila, majstor itd.) .

Osoba ima:

- idOsobe(jedinstveni identifikator)
- ime
- prezime

- adresu elektronske pošte
- količinu novca koju poseduje

Prethodno opisane svojine se upisuju u WorldState, prilikom INIT funkcije chaincode-a, ili neke druge invoke funkcije koja će postaviti stanje tako da sadrži minimum 6 automobila i minimum 3 osobe.

Funkcionalnosti koje će chaincode obezbediti jestu: promena vlasništva, kreiranje kvara, promena boje automobila i popravka automobila.

Zahtevi za projektni zadatak:

- ❏ Kreiranje Fabric mreže sa jednim kanalom **(tri kanala)**
  - specifikacija uključuje četiri organizacije sa po četiri 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 pomenutom kanalu **(na sva tri kanala)**
  - kreiranje svih preostalih neophodnih učesnika u mreži kao i njihove kriptomaterijale (CA, Orderer)
    - ➔ korišćenje CouchDB-a nije obavezno **(obavezno je koristi CouchDB)**
    - ➔ **Ukoliko se radi u paru neophodno je napisati par dodatnih bogatih upita koji pokazuju prednosti CouchDB-a**
    - ➔ svi peer-ovi imaju ulogu endorser-a i commiter-a

- ❏ Chaincode funkcionalnosti (sve podrazumevaju umanjivanje sredstava platioca usluga i povećanje sredstava naručioca usluga odnosno prodavca)
  - promena vlasništva
    - ➔ promena vlasništva je moguća samo ukoliko automobil nema kvarove, a u slučaju da ima kvarove, cena se umanjuje za ukupnu cenu kvarova, i to samo ako kupac izrazi želju da kupi auto čak i ako ima kvar (to može biti specificirano kao parametar invoke funkcije)
  - promena boje
    - ➔ pri čemu se boja specificira kao parametar funkcije, a boja je predstavljena kao stringovski podatak
  - kreiranje kvara
    - ➔ podrazumeva da se funkcija pozove sa parametrom oznake automobila na koji se kvar odnosi, kao i kompletan podatak o kvaru koji će se dodati u listu kvarova pomenutog automobila
    - ➔ ukoliko ukupna cena aktivnih kvarova na automobilu postane veća od cene automobila, potrebno je taj automobil izbrisati iz WorldState-a
  - popravka automobila
    - ➔ podrazumeva da se kvarovi označe kao popravljeni ili obrišu iz podatka o automobilu
  - query funkcije koje omogućuju pretragu po boji, po vlasniku i boji istovremeno

- potrebno je testirati sve implementirane fukncionalnosti (kroz pozive funkcije ili REST pozive)
- potrebno je obraditi sve moguće greške (ukoliko ne postoji automobil ili osoba sa zadatim ključem u WorldState-u, nedovoljna količina sredstava i drugo)

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

Napomene:

- Preporučuje se pisanje chaincode-a u Golang-u,
- 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,
- moguće je koristiti CouchDB (obavezno je koristiti CouchDB)
  - ➔ U tom slučaju potrebno je dozvoliti vršenje upita po proizvoljnom skupu polja
- 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
- Sve što nije ograničeno student može uraditi kako želi, ili se može obratiti asistentu za dodatna pojašnjenja
- **Obavezno je koristi CA (Certificate Authority) i obavezno je testirati sve funkcionalnosti kroz rad sa peer-ovima i userima koji pripadaju Organizaciji 4.**
- .sh skripte za sve potrebne operacije, koje omogućuju testiranje i pokretanje mreže su obavezne
- Sve potrebne fajlove, zajedno sa uputstvom za testiranje i pokretanje mreze, je potrebno arhivirati i poslati na emeil horva.n@uns.ac.rs ili dozvoliti asistentu pristup git repozitorijumu (NebojsaHorvat - git username)
- **Ukoliko se radi u paru neophodno je obezbediti gitHub pristup asistentu. Takođe je obavezna ravnomerna raspodela posla koja se vidi kroz gitHub istoriju.**