Priprema za prvi kolokvijum - Osnove distribuiranog programiranja (2022/20223)

Implementirati distribuirani sistem studentske službe (template zadatka se nalazi u prilogu).

Servis treba da izlaže metode za dodavanje, izmenu, čitanje i brisanje studenata. **Student** treba da ima *ime* (string), *prezime* (string) i *broj indeksa* (string). Unutar klase **Student** treba da postoje i svojstva za dobavljanje i podešavanje polja, takođe mogu da postoje i neke dodatne metode ukoliko je to neophodno za realizaciju zadatka. Za skladištenje studenata koristiti kolekciju Dictionary.

```
public interface IStudentskaSluzba
{
        void DodajStudenta(Student student, string token);
        void IzmeniStudenta(Student student, string token);
        bool PronadjiStudenta(string index, out Student student, string token);
        bool IzbrisiStudenta(string index, string token);
}
```

DodajStudenta - Metoda treba da baci izuzetak ukoliko se pokuša dodavanje studenta koji već postoji.
IzmeniStudenta - Metoda treba da baci izuzetak ukoliko se pokuša izmeniti student koji ne postoji.
PronadjiStudenta - Metoda treba da vrati bool vrednost koja ukazuje na to da li je student pronađen, ukoliko student jeste pronađen vrednost se vraća u izlaznom parametru.
IzbrisiStudneta - metoda treba da vrati bool vrednost kao indikator da li je student uspešno obrisan ili nije.

Za bacanje prethodno nabrojanih izuzetaka, neophodno je napraviti i klasu **StudentskaSluzbaIzuzetak** koja ima jedno polje *razlog* (string).

Takođe, treba implementirati mogućnost prijave na sistem, odnosno autentifikaciju, kao i mogućnost provere prava pristupa sistemu (autorizacija). Dodati još jedan interfejs **IBezbednosniMehanizmi**, kao i klasu Korisnik i sve ono što je neophodno da bi se obezbedile autentifikacija i autorizacija. Ove mehanizme treba implementirati nad metodama servisa studentske službe, tako da metode DodajStudenta i IzmeniStudenta mogu da izvršavaju samo korisnici koji imaju pravo pristupa **Modifikacija**, metodu PronadjiStudenta imaju pravo da izvršavaju korisnici koji imaju pravo pristupa **Citanje**, dok metodu IzbrisiStudenta imaju pravo da izvršavaju korisnici koji imaju pravo pristupa **Brisanje**.

```
public interface IBezbednosniMehanizmi
{
        string Autentifikacija(string korisnickoIme, string lozinka);
}
```

Neophodno je napraviti klasu **BezbednosniMehanizmiIzuzetak** koja ima jedno polje *razlog* (string) i obezbediti bacanje izuzetka ukoliko autentifikacija nije moguća.

Metode za replikaciju treba da se implementiraju na serveru kao poseban servis, uz pomoću interfejsa **IReplikator**.

```
public interface IReplikator
{
        void Posalji(List<Student> studenti);
        List<Student> Preuzmi(DateTime vremeReplikacije);
}
```

Klijent Replikator treba da se poveže sa oba ova servisa i da replicira podatake sa primarnog na sekundarni. Na predefinisani vremenski period (4s) Replikator preuzima podatke sa primarnog i prosleđuje ih na sekundarni server. Ne treba replicirati sve podatke, već samo „deltu“, tj. one koji su dodati ili su izmenjeni nakon prve replikacije.

Po ispod navedenom scenariju implementirati klijenta, koji poziva primarni servis. Dati scenario staviti u beskonačnu petlju.

Scenario**:**

- Dodati studenta broj 1
- Dodati studenta broj 2
- Izmeniti studenta broj 1
- Procitaj students broj 1
- Dodati studenta broj 3
- Izmeniti studenta broj 2
- Obrisati studenta broj 1
- Procitaj studenta broj 1
- Izmeniti studenta broj 3
- Izmeniti studenta broj 1

NAPOMENE :

Sve akcije u sistemu ispisati na konzolu.

Prilikom pozivanja metoda koje bacaju izuzetak neophodno je obrađivati izuzetak.

Podešavanja tačaka priastupa raditi preko App.config fajla.