Elektronski fakultet u Nišu

Katedra za računarstvo

SEMINARSKI RAD

Projektovanje i implementacija sigurnog softvera

Kontrola pristupa bazirana na ulogama

Student:  
Vukadin Drašković, br. indeksa 1613

Sadržaj

[1. Uvod 3](#_Toc171936163)

[2. Sigurnost softvera 4](#_Toc171936164)

[2.1. Mehanizmi kontrole pristupa 4](#_Toc171936165)

[3. Kontrola pristupa bazirana na ulogama u MongoDB bazi podataka 7](#_Toc171936166)

[3.1. Sigurnost kod MongoDB baze podataka 7](#_Toc171936167)

[3.2. Korisnici u MongoDB-u 8](#_Toc171936168)

[3.2.1. Localhost Exception 8](#_Toc171936169)

[3.2.2. Kreiranje korisnika 11](#_Toc171936170)

[3.3. Uloge u MongoDB-u 13](#_Toc171936171)

[3.3.1. Ugrađene uloge 13](#_Toc171936172)

[3.3.2. Korisnički definisane uloge 14](#_Toc171936173)

[4. Aplikacija za upravljanje korisnicima i ulogama u MongoDB-u 17](#_Toc171936174)

[4.1. Arhitektura aplikacije 17](#_Toc171936175)

[4.2. Demonstracija rada aplikacije 18](#_Toc171936176)

[5. Zaključak 26](#_Toc171936177)

[6. Literatura 27](#_Toc171936178)

# Uvod

Sve veći broj kompanija u današnjem vremenu se oslanja na digitalni oblik poslovanja, koji je podržan softverom koji ta kompanija koristi. Softverski sistemi se sastoje od više komponenti - počevši od interfejsa softvera, do servisa koji pružaju usluge klijentskoj aplikaciji, jedne ili više baza podataka koje omogućavaju perzistenciju podataka i keširanje istih. Pristup tim komponentama mora biti ograničen, kako bi ceo sistem funkcionisao na ispravan način. Kontrola pristupa je koncept koji definiše ko i na koji način može prisupati resursima sistema.

Na početku će u radu biti opisani osnovni koncepti projektovanja sigurnog softvera i koje osobine siguran softver mora da poseduje. Zatim će biti obrađeni mehanizmi kontrole pristupa, a zatim će kontrola pristupa bazirana na ulogama biti detaljno obrađena na primeru MongoDB baze podataka. Za kraj će biti prikazani detalji arhitekture, implementacije i funkcionisanja aplikacije za administraciju sistema koji koristi kontrolu pristupa baziranu na ulogama na primeru MongoDB baze podataka.

# Sigurnost softvera

Sigurnost softvera predstavlja skup osnovnih koncepata i praksi koje omogućavaju njegovo bezbedno izvršenje i sigurno upravljanje. Trijumvirat **CIA** predstavlja vodeći princip kojim se pribegava prilikom projektovanja bezbednosti računarskih sistema [1]. Prema ovom modelu, softverski sistem mora obezbediti sledeće karakteristike kako bi se smatrao sigurnim:

1. **Pouzdanost** (engl. *Confidentiality*) - nosi se sa zabranom pristupa informacijama neautorizovanim korisnicima, odnosno definiše ko ima dozvolu čitanja određenih podataka.
2. **Integritet** (engl. *Integrity*) - pod integritetom se podrazumeva sprečavanje neovlašćene modifikacije podataka.
3. **Dostupnost** (engl. *Availability*) - podaci koji se koriste moraju biti uvek dostupni korisnicima. DoS (engl. *Denial-of-Service*) napad predstavlja sajber napad zlonamernog korisnika koji ima za cilj da obori mašinu ili mrežu na kojoj se nalazi server kojem korisnici pristupaju, kako bi onemogućio pristup podacima.

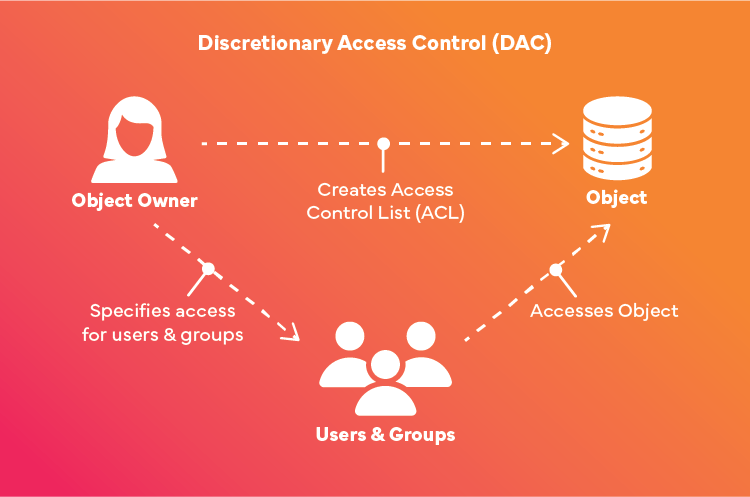
Dva koncepta koja takođe predstavljaju osnovu sigurnosti svakog sistema su **autentifikacija** (engl. *Authentication*) i **autorizacija** (engl. *Authorization*). Pod autentifikacijom se podrazumeva da će korisnik biti identifikovan prilikom pristupa sistemu, odnosno da će se znati ko sistemu pristupa, dok se pod autorizacijom podrazumeva da se prilikom izvršenja određenih akcija ispituje da li korisnik (koji je prethodno autentifikovan) ima pravo na njihovo izvršenje. Autentifikacija i autorizacija se jednom rečju zovu **kontrola pristupa** (engl. *Control* *Access*).

Dobra praksa je da svaki sistem ima jednog ili više administratora koji će imati maksimalne privilegije u sistemu, odnosno koji će imati dozvolu da izvrši bilo koju naredbu. Administrator sistema mora biti pažljiv prilikom upravljanja, jer može izvršiti neke komande koje mogu biti izazvati fatalne posledice po rad sistema.

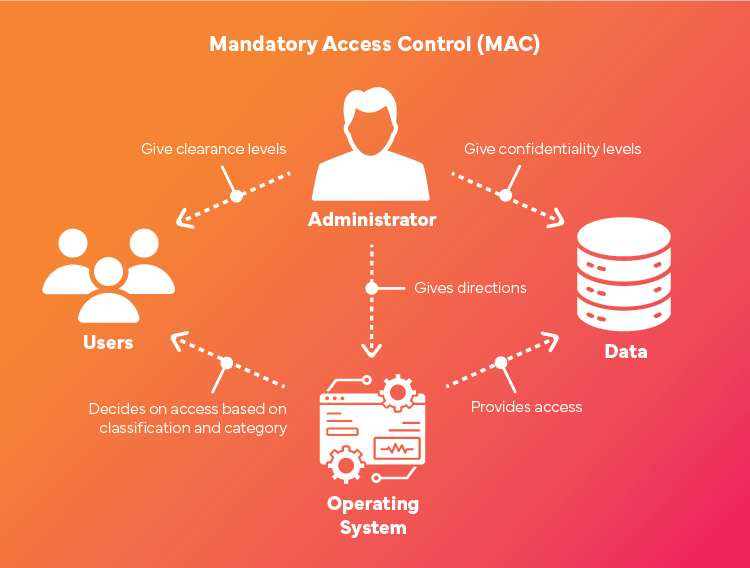
## Mehanizmi kontrole pristupa

Najčešće korišćeni mehanizmi kontrole pristupa su diskreciona kontrola pristupa (engl. *Discretionary access control*, DAC), obavezna kontrola pristupa (engl. *Mandatory access control*, MAC) i kontrola pristupa bazirana na ulogama (engl. *Role-Based access control*).

**Diskreciona kontrola pristupa** omogućava vlasniku resursa u sistemu da upravlja kontrolom pristupa tom resursu, što znači da vlasnik resursa odlučuje ko i na koji način može pristupati tim resursima [2]. Ovim se upravljanje kontrole pristupa prenosi sa administratora na korisnika. Na slici 2.1 je prikazano funkcionisanje ovog mehanizma. Vlasnik resursa kreira listu kontrole pristupa (engl. *Access Control List*), poznatiju kao ACL listu, kojom definiše ko će imati pristup resursima čiji je on vlasnik i na koji način. Kada korisnici žele da pristupe tim resursima, prvo će se obaviti pristup ACL listi kako bi se utvrdilo da li oni uopšte imaju pravo pristupa tom resursu, i ako imaju, koje su im privilegije dodeljenje nad tim resursom. Privilegije se u gruboj podeli mogu podeliti na privlegije čitanja podataka, modifikacije (upisa, ažuriranja i brisanja) podataka i administracije, odnosno upravljanja samim resursom.

  
Slika 2.1. Prikaz funkcionisanja DAC mehanizma

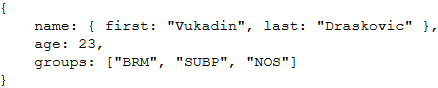
**Obavezna kontrola pristupa** je mehanizam koji funkcioniše pod striktnim sistemski definisanim pravilima [2]. Kod ovog modela, sistem je taj koji kontroliše kojim resursima korisnici mogu pristupiti i na koji način. Svaki resurs u sistemu ima svoj nivo sigurnosti, dok se za korisnika definiše nivo dozvole. Nivoi dozvole koji se najčešće sreću su *Top Secret* (TS), *Secret* (S), *Confidential* (C) i *Unclassified* (U), pri čemu je TS nivo najveće sigurоsti, zatim ide S, C i na kraju U kao nivo najmanje sigurnosti. Na slici 2.2 je prikazan način funkcionisanja ovog mehanizma. Administrator dodeljuje nivoe sigurnosti resursima u sistemu, i daje određene dozvole korisnicima. Kada korisnici pristupaju resursima, proverava se njhov nivo dozvole sa nivoom sigurnosti korisnika. Ukoliko se zahteva čitanje, vrši se provera da li je nivo sigurnosti korisnika veći ili jednak nivou sigurnosti resursa. Prilikom upisa, proverava se da li je nivo sigurnosti resursa veći ili jednak nivou sigurnosti korisnika. Ovo pravilo prilikom upisa može na prvi pogled delovati nelogično, ali je ideja da korisnici sa većim privilegijama u sistemu ne mogu da modifikuju resurse sa nižim privilegijama, jer bi time korisnici sa nižim privilegijama mogli da vide podatke korisnika sa višim privilegijama.

  
Slika 2.2. Način funkcionisanja MAC mehanizma na primeru operativnog sistema

**Kontrola pristupa bazirana na ulogama** [3] je još jedan mehanizam kontrole pristupa. U sistemu se definišu uloge koje mogu imati određene privilegije nad resursima u bazi podataka, a zatim administrator te uloge dodeljuje korisnicima kako bi im ograničio kontrolu pristupa. Uloge se najčešće kreiraju tako da predstavljaju prave uloge pojedinaca u nekoj kompaniji - na primer, ukoliko u kompaniji postoje sektori za menadžment, finansije, razvoj, testiranje i logistiku, dobra je praksa da se kreiraju posebne uloge u sistemu za svaki od ovih sektora. Korisniku se može dodeliti više uloga. Administrator sistema bi se ovim mehanizmom kreirao kao i svaki drugi korisnik, s tim što bi mu bila dodeljena uloga sa maksimalnim privilegijama.

# Kontrola pristupa bazirana na ulogama na primeru MongoDB baze podataka

MongoDB [4] je tipičan predstavnik dokument NoSQL (*Not only SQL*) baza podataka, gde se svaki rekord baze podataka (analogno torci, odnosno slogu u relacionim bazama podataka) skladišti u posebnom dokumentu. Podaci koje skladišti MongoDB su slični JSON (*JavaScript Object Notatio*n) objektima, gde su podaci organizovani u obliku *ključ: vrednost* (Slika 3.1). Vrednost određenog ključa može biti niz, neki drugi dokument, ili čak niz dokumenata. Podaci su organizovani u kolekcije koje su analogne tabelama u relacionim bazama podataka, dok je baza podataka zadužena za upravljanje većim brojem kolekcija.

  
Slika 3.1. Struktura dokumenta u MongoDB-u

## Sigurnost kod MongoDB baze podataka

MongoDB nudi podršku za autentifikaciju, kao i za autorizaciju. Mehanizmi za autentifikaciju [5] koje podržava MongoDB su:

1. SCRAM (*Salted Challenge Response Authentication Mechanism*) – podrazumevani mehanizam autentifikacije u najnovijoj verziji MongoDB-a 7.0
2. Autentifikacija na osnovu x.509 sertifikata
3. Kerberos
4. LDAP Proxy
5. OpenID Connect

Kontrola pristupa se u MongoDB-u može omogućiti na dva načina [6] – navođenjem *--auth* opcije prilikom pokretanja instance MongoDB-a, ili promenom konfiguracionog fajla *mongod.conf.orig* u folderu */etc* dodavanjem dela prikazanog na slici 3.2. Inicijalno je kontrola pristupa onemogućena.

  
Slika 3.2. Izmena konfiguracionog fajla MongoDB-a radi aktivacije kontrole pristupa

Autentifikacija korisnika se u MongoDB-u odvija na osnovu korisničkog imena korisnika i njegove lozinke, dok se autorizacija uspostavlja na osnovu uloga (engl. *role*), koje definišu koji tip privilegija (privilegije za čitanje, modifikaciju, brisanje podataka, kreiranje novih kolekcija, itd.) korisnik može imati nad određenim resursom, gde resurs može predstavljati čitavu bazu podataka ili određeni skup kolekcija unutar baze.

## 3.2. Korisnici u MongoDB-u

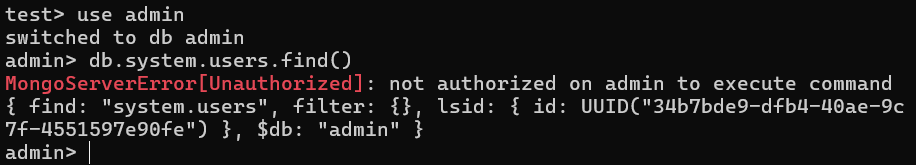
Kontrola pristupa bazirana na ulogama funkcioniše tako što se uloga dodeljuje korisniku, te je s toga u bazi podataka neophodno kreirati korisnika kome će biti dodeljena određena uloga.

Ukoliko nismo omogućili kontrolu pristupa u MongoDB-u, korisnik nije dužan da se identifikuje pre izvršenja naredbi u bazi podataka. U suprotnom, ako je kontrola pristupa aktivirana, svaki korisnik je dužan da se autentifikuje na osnovu svog korisničkog imena, lozinke i baze. Korisnici se u MongoDB-u kreiraju *createUser* naredbom, a autentifikuju se ili prilikom povezivanja na bazu, ili nakon povezivanja izvršenjem *auth* naredbe. Za korisnika je potrebno da se specificira njegova uloga u bazi, gde uloga označava skup privilegija za koje su omogućene određene akcije nad resursima MongoDB-a.

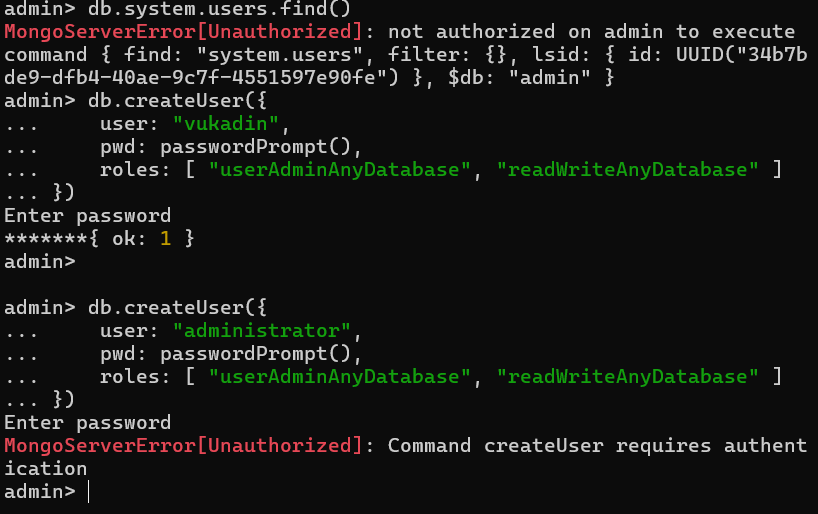
Dobra je praksa da svaki korisnik baze podataka ima svoj nalog, gde se u korisnike pored ljudi svrstavaju i aplikacije koje koriste MongoDB. Kao što je već rečeno u uvodnom poglavlju, očekivano je da postoji administrator baze podataka koji će imati najveće privilegije u bazi podataka koji će biti zadužen za upravljanjem celog sistema, odnosno samim podacima u bazi, korisnicima, rolama, itd...

### 3.2.1. Localhost Exception

Aktivacijom kontrole pristupa se zahteva da korisnici koji se povežu na bazu moraju biti autentifikovani. Kada se prvi put pokrene baza podataka, u sistemu ne postoji nijedan korisnik koji bi mogao da se autentifikuje kako bi mogao da izdaje naredbe. Tu nastaje problem jer nijedan korisnik nije u stanju da upravlja bazom podataka ili dodaje podatke u istu zbog nemogućnosti autentifikovanja, što je prikazano na slici 3.3. Ali i za to postoji rešenje.

  
Slika 3.3. Nemogućnost izvršenja naredbi u bazi podataka sa omogućenom kontrolom pristupa pri nepostojanju korisnika

Prilikom prvog povezivanja na MongoDB, kada je kontrola pristupa omogućena, dešava se *localhost exception* [7] koji omogućava da se kreira inicijalni korisnik u *admin* bazi podataka. Najčešća praksa je da se na ovaj način kreira nalog administratora.

  
Slika 3.4. Localhost exception

Na slici 3.4 je prikazan *localhost exception*. Uočavamo da nije dozovljeno izvršavanje drugih naredbi u *admin* bazi podataka, osim naredbe za kreiranje korisnika. Nakon kreiranja inicijalnog korisnika nije moguće kreirati novog, zato što MongoDB zna da sada u sistemu već postoji korisnik koji se može autentifikovati, i ne dozvoljava da se bez autentifikacije izvršava bilo koja naredba u sistemu, jer je kontrola pristupa omogućena. Na slici 3.5 je prikazano da je nakon autentifikovanja kao novokreirani korisnik moguće izvršavati naredbe u sistemu. Konkretno, naredba *db.system.users.find()* u admin bazi podataka ima za cilj da prikaže sve korisnike koje postoje u MongoDB-u. U trenutnoj listi korisnika baze podataka postoji samo admin korisnik koji je kreiran u prethodnom primeru.

  
Slika 3.5. Logovanje u sistem kao novokreirani korisnik i mogućnost izvršenja naredbi

Naredba *auth* omogućava logovanje na MongoDB [8]. Može se pozvati na više različitih načina:

1. Navođenjem samo korisničkog imena korisnika nakon čega se od korisnika zahteva unos lozinke.

admin> db.auth(“vukadin”)

1. Navođenjem korisničkog imena korisnika uz *passwordPrompt()* argument koji od korisnika eksplicitno zahteva da unese lozinku. Ekvivalenat je prvom načinu izvršenja *auth* naredbe.

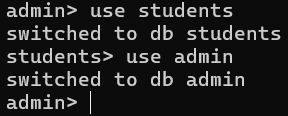
admin> db.auth(“vukadin”, passwordPrompt())

1. Navođenjem korisničkog imena i lozinke. Ovaj metod se izbegava jer je korisnikova lozinka vidljiva u terminalu prilikom autentifikovanja što je prikazano na slici 3.8, čime se narušava privatnost korisnika, tako da se je preporučeno da se koristi neki od prethodna dva načina.

admin> db.auth(“vukadin”, “vukadin”)

### Kreiranje korisnika

Za potrebe kreiranja korisnika u MongoDB-u postoji *createUser* naredba [9]. Korisnik će se kreirati u onoj bazi podataka u kojoj se terminal nalazi pri izvršenju *createUser* komande. Naredba koja nam omogućava da pređemo iz jedne u drugu bazu podataka je *use <database>*, koja je prikazana na slici 3.6. Pri promeni baze podataka, menja se *prompt* (početni deo linije terminala) koji signalizira u kojoj bazi podataka se trenutno korisnik nalazi.

  
Slika 3.6. *use* naredba

Naredba createUser ima sledeći format:

db> db.createUser(user, writeConcern)

gde prvi argument *user* predstavlja sve podatke o korisniku za koga se kreira nalog, dok drugi argument *writeConcern*, koji je za razliku od prvog opcioni, predstavlja nivo potvrde za *write* operacije korisnika u MongoDB-u.

Argument *user* predstavlja MongoDB dokument koji može imati sledeći oblik:

{

user: “<name>”,

pwd: passwordPrompt(), // ili “<plaintext password>”

customData: { <any information> },

roles: [

{ role: “<role>”, db: “<database>” } | “<role>”,

…

],

authenticationRestrictions: [

{

clientSource: [“<IP>” | “<CIDR range>”, …],

serverAddress: [“<IP>” | “<CIDR range>”, …]

},

…

],

mechanism: [“<SCRAM-SHA-1|SCRAM-SHA-256>”, …],

passwordDigestor: “<server|client>”

}

Polja koja predstsavljaju *user* argument su:

1. **user** - podatak koji je tipa *string* i predstavlja korisničko ime.
2. **pwd** - *string* podatak koji predstavlja lozinku korisnika. Postoje dva načina za postavljanje lozinke - prvi način je da korisnik postavi šifru u čitljivom obliku, što nije preporučljivo jer će se lozinka videti na ekranu čime se narušava bezbednost. Drugi način je da se za *pwd* polje ostavi poziv *passwordPrompt()* metode koja zahteva od korisnika da unese šifru naknadno. Kada se naknadno unosi šifra u terminal, karakteri nisu vidljivi na ekranu, čime se ne narušava bezbednost.
3. **customData** - ovo polje je tipa *document* i opciono je. Očekuje se da se u *customData* navedu informacije koje će bolje opisati korisnika za koga se kreira nalog.
4. **roles** - predstavlja niz rola koje korisnik može da ima. Uloge mogu biti ugrađene ili definisane od strane korisnika. Ovo polje nije opciono, i moramo navesti prazan niz, odnosno *[]* ukoliko korisnik nema role. Role se mogu navoditi tako što će se navesti dokument koji sadrži dva polja, prvo polje koje definiše rolu, a drugo polje koje predstavlja za koju bazu podataka važi rola. Drugi način je da se samo navede naziv role, gde se podrazumeva da korisnik dobija navedenu ulogu u bazi podataka u kojoj je kreiran.
5. **authenticationRestrictions** - predstavlja niz ograničenja gde *clientSource* definiše sa kojih IP adresa, ili iz kog opsega IP adresa se korisnik može prijaviti na bazu podataka, dok *serverSource* predstavlja opseg adresa, ili adresu sa koje server može da prihvati korisnika (ima primenu u distribuiranom korišćenju MongoDB-a na klasteru).
6. **mechanism** - opciono polje koje definiše koji se mehanizam koristi za autentifikaciju korisnika. Podrazumevano se koriste “*SCRAM-SHA-1“* i “*SCRAM-SHA-256“*, tako da korisnik može da se loguje na sistem koristeći oba mehanizma. Ostali mehanizami koje se mogu navesti su “*MONGODB-X509*“, “*GSSAPI*“, “*PLAIN*“ i “*MONGODB-OIDC*“.
7. **passwordDigestor** - opciono string polje koje može uzimati isključivo dve vrednosti - prva vrednost je “*server*“, koja je i podrazumevana, dok je druga vrednost “*client*“. Označava ko će „začiniti“ lozinku prilikom kreiranja korisnika.

## Uloge u MongoDB-u

Uloge se, kako u ostalim softverskim sistemima, tako i u MongoDB-u, definišu kao skup privilegija, gde svaka privilegija označava koji skup akcija se može izvršiti nad definisanim resursima. MongoDB ima **ugrađene role** (engl. *Built-In Roles*) koje se mogu dodeliti korisnicima, a postoji mogućnost kreiranja **korisnički definisanih uloga** (engl. *User-Defined Roles*) ukoliko za tim ima potrebe.

### 3.3.1. Ugrađene uloge

Ugrađene uloge [10] se u MongoDB-u mogu podeliti na korisničke i administratorske uloge koje važe za sve baze podataka, i ostale uloge koje se mogu dodeliti korisniku isključivo nad *admin* bazom podataka.

Korisničke uloge (engl. *Database User Roles*) koje se mogu dodeliti korisnicima u svim bazama podataka su:

1. **read** - omogućava čitanje podataka iz kolekcija u bazi podataka, prikaz indeksa i kolekcija koje postoje u bazi, kao i nekih statističkih parametara.
2. **readWrite** - na skup privilegija koje podrazumeva *read* uloga dodaje i privilegije za modifikaciju podataka, kao i privilegije za kreiranje i modifikaciju indeksa i kolekcija.

Korisničke uloge ne omogućavaju administraciju baze podataka, već samo čitanje i upravljanje podacima u njoj. Uloge za potrebe administracije baze podataka (engl. *Database Administration Roles*) su:

1. **dbAdmin** - omogućava modifikaciju šeme i indeksa baze podataka, kao i prikupljanje određene statistike. Ne uključuje privilegije za upravljanje korisnicima i ulogama. Nad nesistemskim bazama ne omogućava *find* (srp. *pronaći*) privilegiju za čitanje podataka. Jedina sistemska baza nad kojom se može dodeliti ova uloga je *system.profile*, koja služi da čuva istoriju komandi koje su se izvršile nad instancom MongoDB-a.
2. **dbOwner** - predstavlja skup privilegija koje omogućavaju *dbAdmin*, *readWrite* i *userAdmin* uloge.
3. **userAdmin** - isključivo omogućava privilegije za kreiranje korisnika i uloga u bazi podataka, kao i modifikaciju istih.

MongoDB sadrži ugrađene uloge za administraciju klastera (engl. *Cluster Administration Roles*), s obzirom na to da nudi podršku za rad u distribuiranom okruženju na klasteru uređaja. Ove uloge se isključivo mogu primeniti nad *admin* bazom podataka, a to su **clusterManager**, **clusterMonitor**, **hostManager** i **clusterAdmin** koji uključuje sve privilegije prethodnih uloga za administraciju klastera.

Za potrebe kreriranja rezervnih kopija (engl. *backup*) i oporavka baze podataka (engl. *restore*) postoje uloge **backup** i **restore**. Ove uloge se moraju primeniti nad *admin* bazom podataka, a predstavljaju skup privilegija koje su neophodne za izvršenje akcija vezanih za kreiranje rezervnih kopija i oporavka svih baza podataka.

Uloge koje su dostupne u *admin* bazi podataka, a čije se privilegije primenjuju na sve baze podataka (engl. *All-Database Roles*) u sistemu, izuzev na *local* i *config* bazu su:

1. **readAnyDatabase** - omogućava privilegije *read* uloge nad svim bazama podataka u sistemu.
2. **readWriteAnyDatabase** - na skup privilegija *readAnyDatabase* uloge dodaje privilegije *readWrite* uloge koja se može primeniti nad svim bazama podataka.
3. **userAdminAnyDatabase** - predstavlja skup privilegija *userAdmin* uloge koje se mogu primeniti nad svim bazama podataka.
4. **dbAdminAnyDatabase** - skup privilegija *dbAdmin* uloge koje se mogu primeniti nad svim bazama podataka u sistemu.

Ukoliko korisnik dobije **root** ulogu nad *admin* bazom podataka, to znači da ima sve privilegije nad svim bazama u sistemu. S toga ovaj korisnik predstavlja glavnog administratora MongoDB-a (engl. *superuser*).

### 3.3.2. Korisnički definisane uloge

Iako MongoDB nudi veliki broj ugrađenih rola, postoji šansa da nijedna od njih ne zadovoljava potrebe korisnika, odnosno da ne obuhvata skup privilegija koji želi da dodeli korisniku nad nekim resursom. S toga postoji mogućnost kreiranja korisnički definisanih uloga [11]. Za te potrebe postoji *createRole* naredba [12]. Rola se kreira u bazi podataka u kojoj se terminal nalazio u trenutku izvršenja *createRole* komande. Kako bi jedinstveno identifikovao ulogu, MongoDB koristi naziv uloge i bazu podataka u kojoj uloga važi. Ovo omogućava kreiranje uloga sa istim nazivom u različitim bazama podataka.

Naredba *createRole* ima sledeći oblik:

db.createRole(role, writeConcern)

gde je prvi argument *role* namenjen za navođenje ključnih stvari u ulozi koja se kreira, dok je drugi argument isti kao i kod *createUser* naredbe. Argument role predstavlja dokument koji može imati sledeći oblik:

{

role: “<name>”,

privileges: [

{ resource: { <resource> }, actions: [ “<action>”, …] },

…

],

roles: [

{ role: “<role>”, db: “<database>” } | “<role>”,

…

],

authenticationRestrictions: [

{

clientSource: [“<IP>” | “<CIDR Range>”, …],

serverSource: [“<IP>” | “<CIDR Range>”, …]

},

…

]

}

Polja koja predstavljaju *role* argument su:

1. **role** - *string* podatak koji predstavlja naziv uloge koja se kreira.
2. **privileges** - niz koji predstavlja skup privilegija koje će dobiti novokreirana uloga. Ovo nije opcioni argument, tako da je neophodno ostaviti prazan niz u slučaju kada uloga nema privilegije. Elementi ovog niza su dokumenti koji mogu imati dva polja - prvo polje *resource*, koji definiše nad kojim resursom se dodeljuje privilegija, i drugo polje *actions*, koje predstavlja niz podataka tipa string, gde svaki od njih označava koja je akcija omogućena nad resursom. Resurs može biti cela baza podataka, a može biti i kolekcija unutar baze. Neke od najčešće korišćenih akcija su *find*, *insert*, *remove*, *update*, *createRole*, *grantRole*, *dropRole*, *createUser*, *dropUser*, *createCollection*, *dropCollection*...
3. **roles** - predstavlja niz uloga od kojih će novokreirana uloga naslediti privilegije. Uloge čije privilegije novokreirana rola nasleđuje mogu biti i korisnički definisane, i ugrađene. Argument nije opcioni i analogno *privileges* polju zahteva prazan niz ukoliko uloga ne nasleđuje privilegije drugih rola.
4. **authenticationRestrictions** - jedini opcioni argument. Predstavlja niz ograničenja koja definišu opseg IP adresa sa kojih se može pristupati resursima, analogno istom polju u *createUser* naredbi.

Za potrebe prikaza svih uloga koje postoje u bazi podataka u kojoj se trenutno nalazi terminal postoji *getRoles* komanda [13]. Ova komanda za argument prihvata dokument, koji je opcioni, sa sledećim poljima:

1. **rolesInfo** - ukoliko se navodi dokument koji je argument *getRoles* komande, onda se ovo polje mora navesti. Celobrojnog je tipa i ukoliko se za vrednost stavi 1 komanda će vratiti korisnički definisane uloge.
2. **showAutentificationRestrictions** - opcioni *boolean* argument kojim se označava da li će komanda prikazati *authenticationRestrictions* za ulogu. Podrazumevano je postavljen na *false*.
3. **showBuiltInRoles** - opciona *boolean* vrednost koja određuje da li će se prikazati ugrađene role.
4. **showPrivileges** - opcioni argument koji može uzimati *true* ili *false* vrednost. Vrednost ovog argumenta određuje da li će se za ulogu prikazati privilegije ili ne. Podrazumevana vrednost je *false*.

Ukoliko se komanda *getRoles* izvrši bez argumenata, onda će vratiti sve korisnički definisane uloge u bazi.

# Aplikacija za upravljanje korisnicima i ulogama u MongoDB-u

Aplikacija za upravljanje korisnicima i ulogama u MongoDB-u je Windows Forms aplikacija napisana u .NET-u 8.0. Motiv za razvojem aplikacije je pojedostavljeno upravljanje MongoDB-om.

Relacione baze podataka koriste DCL (engl. *Data control language*) jezik [14] za upravljanje bazaom podataka, koji je tip skript jezika nalik SQL jeziku, na kome se mogu pisati skripte koje će se izvršiti nad instancom baze podataka. Ovakav pristup umnogome olakšava administraciju baze, jer se pisanjem jedne skripte koja se poziva prosleđivanjem određenih argumenata značajno automatizuje proces administracije.

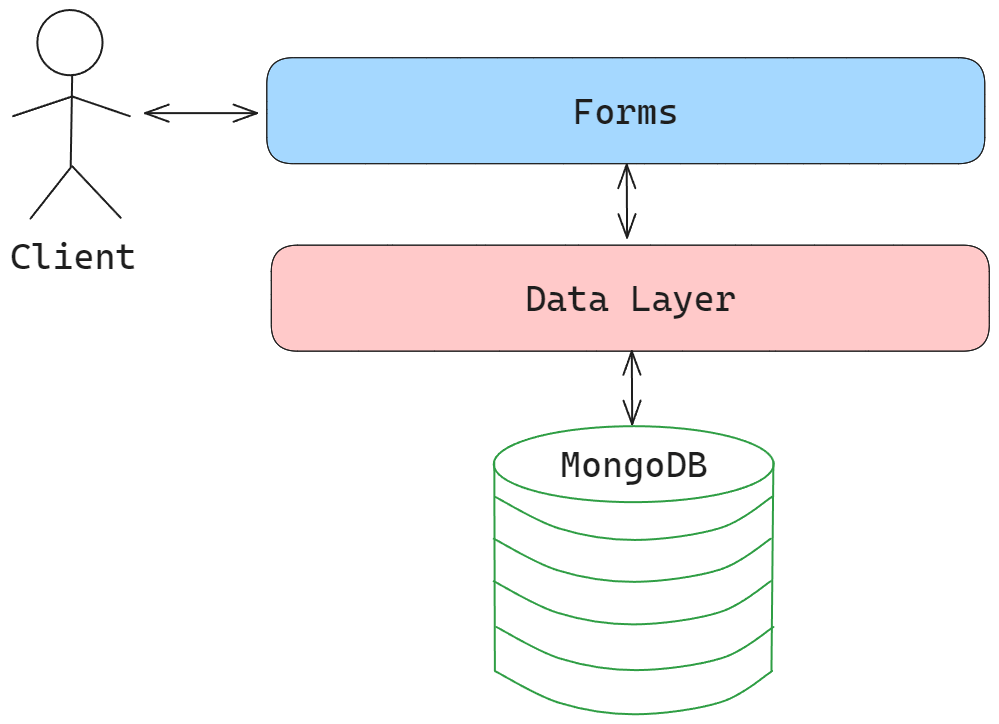
MongoDB nije tip relacione baze podataka i time je sama sintaksa naredbi za upravljanje korisnicima i ulogama dosta drugačija od istih naredbi kod relacionih baza. Postojanje opcionih argumenata, različiti tipovi podataka i njihova nestruktuiranost je glavni razlog nepraktičnosti administracije ove baze. Naredbe MongoDB-a ne nalikuju naredbama iz drugih baza podataka, već predstavljaju poziv metode nad instancom baze podataka. Iz ovog razloga se skripte za MongoDB pišu u JavaScript programskom jeziku.

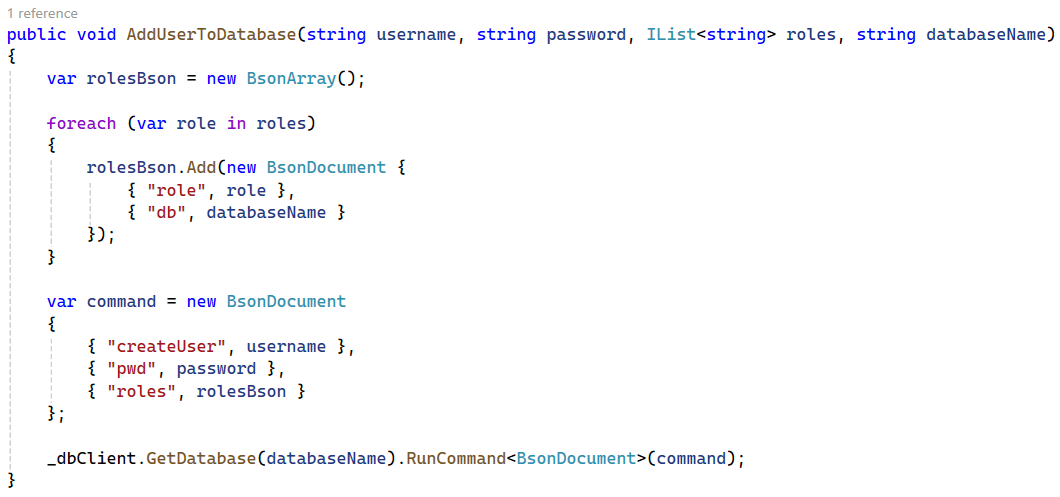
Aplikacija ima za cilj da u par klikova omogući administratoru kreiranje korisnika, uloga, modifikaciju istih, kao i njihovo brisanje. Takođe, ono što je najvažnije, aplikacija ima za cilj dodelu uloge određenim korisnicima, kao i brisanje uloge korisniku.

## Arhitektura aplikacije

Aplikacija ima jednostavnu arhitekturu (Slika 4.1) - postoji nivo formi preko kojih korisnik interaguje sa aplikacijom. Tu je i DataLayer nivo koji komunicira sa MongoDB instancom enkapsuliranim izvršenjem MongoDB naredbi i implementira projektni obrazac Singleton.

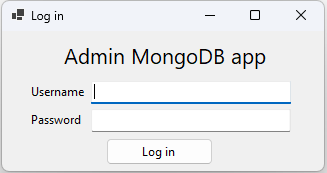
Za komunikaciju sa MongoDB-jem je korišćen *MongoDB.Driver* NuGet paket verzije 2.25.0 [15]. Važno je napomenuti da ovaj NuGet paket ne sadrži mogućnost poziva naredbi za administraciju MongoDB baze podataka. Jedini način izvršenja naredbi je korišćenjem *RunCommand* metode nad određenom bazom, gde se kao argument prosleđuje *string* koji definiše komandu koja se treba izvršiti sa svojim argumentima (na slici 4.2 je prikazan način kreiranja korisnika korišćenjem *RunCommand* metode). Time je dosta otežan posao razvoja aplikacije za administraciju MongoDB-a, jer projektanti ovog NuGet paketa nisu kreirali metode koje se mogu direktno pozivati, već se treba tražiti alternativni način za izvršenje tih naredbi.

  
Slika 4.1. Arhitektura aplikacije

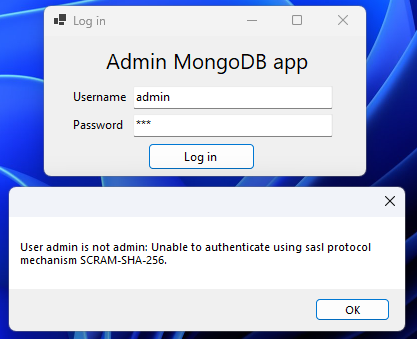
  
Slika 4.2. Kreiranje korisnika u bazi podataka korišćenjem RunCommand metode

## Demonstracija rada aplikacije

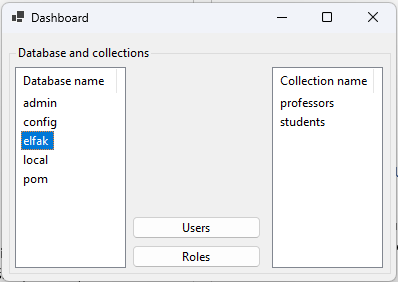
Na slici 4.2 se može videti prozor koji se prikazuje na ekranu po pokretanju aplikacije. Od korisnika se zahteva unos korisničkog imena i lozinke kako bi pristupio administraciji MongoDB-a.

  
Slika 4.2. Početni prozor aplikacije

Ukoliko korisnik pokuša da se loguje na sistem pogrešnim kredencijalima ili kredencijalima korisnika koji nije administrator, aplikacija će onemogućiti konekciju i prijaviti grešku (Slika 4.3).

  
Slika 4.3. Greška prilikom logovanja na MongoDB

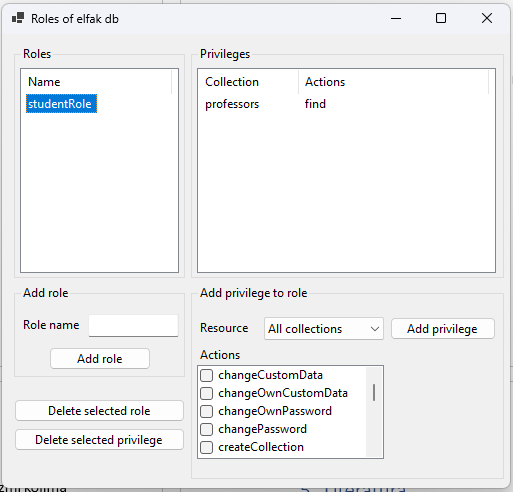
Nakon uspešnog logovanja administratorskim nalogom, korisniku će se otvoriti prozor gde će se naći spisak baza podataka koje postoje u MongoDB-u, kao i kolekcije selektovane baze što je prikazano na slici 4.4. Nakon toga, pritiskom na dugme Users ili Roles, korisnik može da odluči da li želi da upravlja korisnicima ili ulogama u selektovanoj bazi podataka.

  
Slika 4.4. Prikaz bazi podataka u instanci MongoDB-a

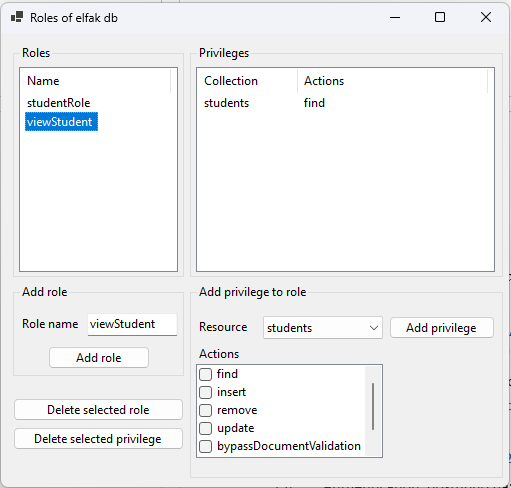
Prvo će biti obrađeno upravljanje ulogama, a zatim upravljanje korisnicima. Klikom na dugme Roles će na ekranu biti prikazan spisak korisnički definisanih uloga i privilegije selektovane uloge. Razlog prikaza isključivo korisnički-definisanih uloga je taj što se jedino tim ulogama može upravljati. Na slici 4.5 se može videti da za sada u *elfak* bazi podataka postoji samo jedna korisnička uloga *studentRole*, sa privilegijom koja je definisana akcijom *find* nad *professors* kolekcijom.

Administratoru je omogućeno kreiranje nove uloge, za koju je neophodno da navede samo njen naziv. Nakon toga može dodeliti privilegije selektovanoj ulozi. Važno je napomenuti da će ponuđene akcije koje se mogu dodeliti privilegiji strogo zavisiti od resursa koji je korisnik odabrao. Ukoliko je korisnik odabrao sve kolekcije, odnosno čitavu bazu, onda će mu se pružiti mogućnost odabira akcija koje se mogu izvršiti nad svim kolekcijama odabrane baze. Ukoliko je odabrao jednu kolekciju unutar baze, skup akcija koje se mogu dodeliti ulozi će se razlikovati.

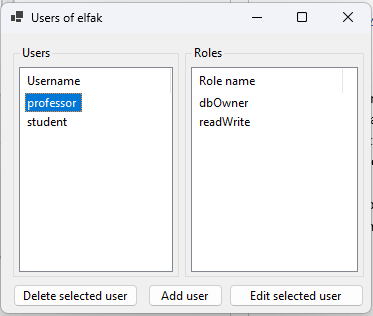
Za potrebe demonstracije rada aplikacije će biti kreirana uloga *viewStudent* koja će omogućiti korisniku koji bude imao tu ulogu isključivo pretraživanje kolekcije *student* (prikazano na slici 4.6).

  
Slika 4.5. Prikaz prozora za upravljanje ulogama

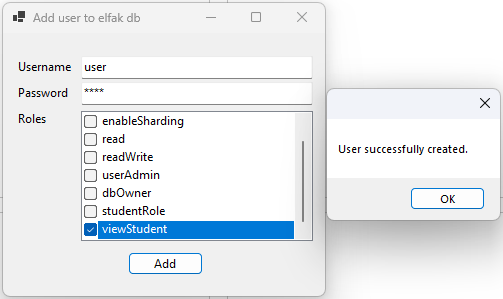
U donjem levom uglu prozora za upravljanje ulogama se nalaze dugmad za brisanje selektovane uloge, odnosno selektovane privilegije uloge.

  
Slika 4.6. Kreiranje *viewStudent* uloge

Sada će biti prikazano kreiranje jednog korisnika u *elfak* bazi podataka. Prozor za upravljanje korisnicima je predstavljen spiskom korisnika sa leve strane, i spiskom uloga selektovanog korisnika sa desne strane, kao što je prikazano na slici 4.7.

  
Slika 4.7. Prikaz korisnika i njihovih uloga u *elfak* bazi podataka

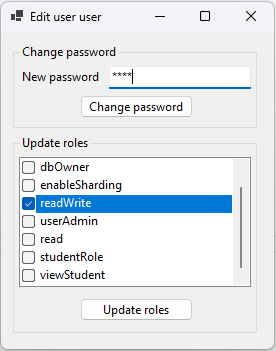
Ukoliko se odabere opcija *Add user*, otvoriće se prozor za kreiranje i dodavanje novog korisnika (Slika 4.8), gde se zahteva unos korisničkog imena i lozinke korisnika, kao i dodela uloga. Novokreirani korisnik će imati korisničko ime *user* i lozinku *user*, sa prethodno kreiranom ulogom *viewStudent* u *elfak* bazi podataka.

  
Slika 4.8. Kreiranje *user* korisnika

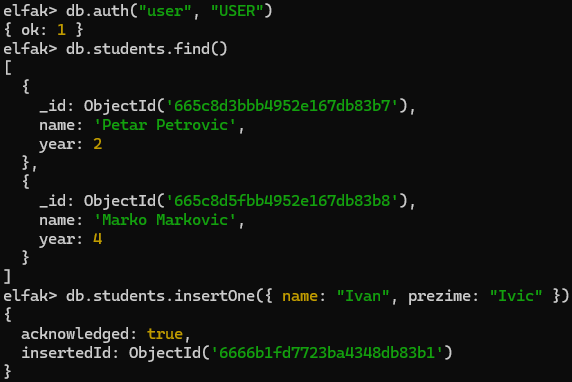
Na slici 4.9 je dokazano funkcionisanje rada aplikacije. Korisnik *user* se logovao na MongoDB instancu svojim kredencijalima, može pretraživati podatke u *students* kolekciji, ali nema privilegije za modifikaciju iste.

  
Slika 4.9. Funkcionalnosti *user* korisnika

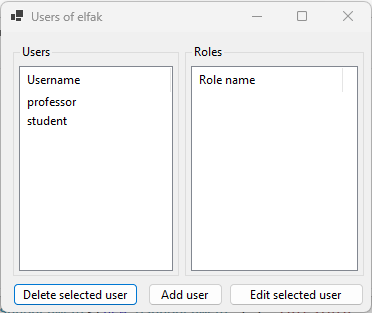
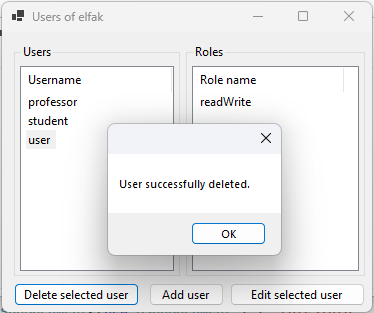
Aplikacija nudi mogućnost modifikovanja postojećeg korisnika u bazi podataka. Korisnik se može modifikovati tako što će mu biti promenjen skup uloga koje ima u određenoj bazi, ili tako što će mu biti promenjena lozinka. Korisnik *user* će sadržati samo *readWrite* ulogu i lozinka će mu biti *USER* (Slika 4.10).

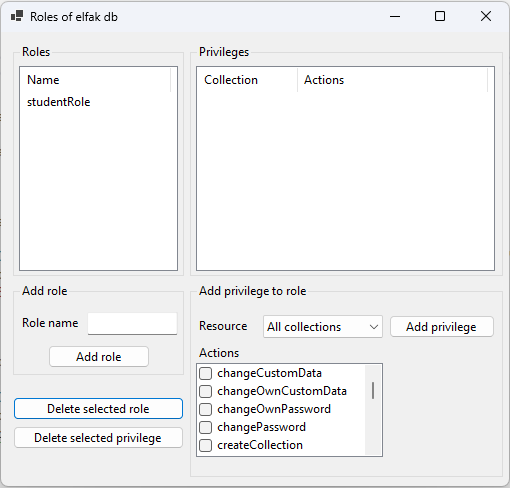
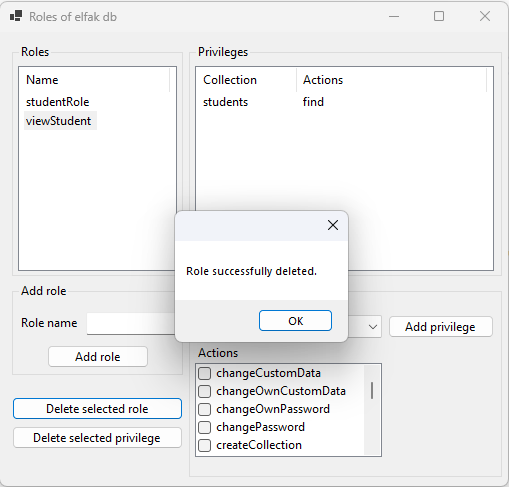
  
Slika 4.10. Modifikacija korisnika *user*

Na slici 4.11 je prikazano logovanje korisnika novim kredencijalima i mogućnost izvršenja *insert* akcije nad kolekcijom u *elfak* bazi podataka.

  
Slika 4.11. Nova uloga i kredencijali *user* korisnika

Za kraj je demonstrirana mogućnost brisanja korisnika (Slika 4.12) i uloga (Slika 4.13).

  
Slika 4.12. Brisanje korisnika

  
Slika 4.13. Brisanje uloge

# Zaključak

U radu su prvo predstavljeni osnovni pojmovi sigurnosti softvera i mehanizmi kojima se sigurnost sprovodi. Kasnije je dat osvrt na mehanizme kontrole pristupa, gde je ostatak rada posvećen kontroli pristupa baziranoj na ulogama. Kao primer sistema koji koristi kontrolu pristupa baziranu na ulogama je uzeta MongoDB baza podataka.

Kontrola pristupa bazirana na ulogama nudi veliku fleksibilnost administratoru sistema po pitanju privilegija i uloga koje se mogu dodeliti korisniku. Mogućnost kreiranja korisnički-definisanih uloga umnogome olakšava posao administratoru, jer se sa pojavom novih korisnika u sistemu i određivanje njihovih privilegija veoma lako sprovodi kreiranjem uloga.

Aplikacija za upravljanje korisnicima i ulogama u MongoDB bazi podataka ima za cilj da apstrahuje proces modifikacije korisnika i uloga i značajno pojednostavi administratoru taj proces. Sa stanovišta projektanta aplikacije se može reći da nepostojanje interfejsa za izvršenje naredbi administracije u MongoDB-u značajno otežava razvoj ovakve aplikacije. Ipak, korišćenjem interfejsa za izvršenje komandi u *raw* obliku je premošćen nedostatak NuGet paketa koji je korišćen za potrebe razvoja aplikacije.

# Literatura

[1] Mark Stamp, *Information Security - Principles and Practice*, WILEY

[2] Discretionary Access Control vs. Mandatory Access Control: What's the Difference?, dostupno na: <https://www.zuar.com/blog/discretionary-access-control-vs-mandatory-access-control/> (pristupljeno 10. juna 2024.)

[3] What Is Role-Based Access Control & Why Your Portal Needs It, dostupno na: <https://www.zuar.com/blog/what-is-role-based-access-control-for-portals/> (pristupljeno 10. juna 2024.)

[4] MongoDB, dostupno na: <https://www.mongodb.com/> (pristupljeno 11. juna 2024.)

[5] Authentication, dostupno na: <https://www.mongodb.com/docs/manual/core/authentication/> (pristupljeno 11. juna 2024.)

[6] Use SCRAM to Authenticate Clients, dostupno na: <https://www.mongodb.com/docs/manual/tutorial/configure-scram-client-authentication/> (pristupljeno 11. juna 2024.)

[7] Localhost Exception, dostupno na: <https://www.mongodb.com/docs/manual/core/localhost-exception/> (pristupljeno 12. juna 2024.)

[8] db.auth(), dostupno na: <https://www.mongodb.com/docs/manual/reference/method/db.auth/> (pristupljeno 12. juna 2024.)

[9] db.createUser(), dostupno na: <https://www.mongodb.com/docs/manual/reference/method/db.createUser/> (pristupljeno 14. juna 2024.)

[10] Built-In Roles, dostupno na: <https://www.mongodb.com/docs/manual/reference/built-in-roles/> (pristupljeno 15. juna 2024.)

[11] User-Defined Roles, dostupno na: <https://www.mongodb.com/docs/manual/core/security-user-defined-roles/> (pristupljeno 15. juna 2024.)

[12] db.createRole(), dostupno na: [https://www.mongodb.com/docs/manual/reference/method/db.createRole/#mongodb-method-db.createRole](https://www.mongodb.com/docs/manual/reference/method/db.createRole/%23mongodb-method-db.createRole) (pristupljeno 15. juna 2024.)

[13] db.getRoles(), dostupno na: <https://www.mongodb.com/docs/manual/reference/method/db.getRoles/> (pristupljeno 15. juna 2024.)

[14] Types of Database Languages and Their Uses (Plus Examples), dostupno na: <https://www.indeed.com/career-advice/career-development/database-languages> (pristupljeno 19. juna 2024.)

[15] MongoDB.Driver, dostupno na: <https://www.nuget.org/packages/MongoDB.Driver/> (pristupljeno 20. juna 2024.)