

Veb aplikacija za preuzimanje ličnih dokumenata

Petar Ćoćić
Fakultet tehničkih nauka
Univerzitet u Novom Sadu
Trg Dositeja Obradovića 6, 21000 Novi Sad
cocic.sr16.2020@uns.ac.rs

Sažetak:

U ovom radu je opisana veb aplikacija koja je namenjena kao podrška sistema eUprave [1], radi olakšanog pristupa ličnim dokumentima u elektronskom obliku. Veb aplikacija konkretno podržava sistem eMatičar, čija je dužnost da vodi evidenciju o građanima Republike Srbije. Primenom predloženog rešenja se ostvaruje značajna ušteda vremena za prikupljanje određenih dokumenata koji su neophodni za ispunjavanje određenih pravnih dužnosti građana.

Ključne reči: veb aplikacija, eUprava, digitalni oblik dokumenata.

1. Uvod

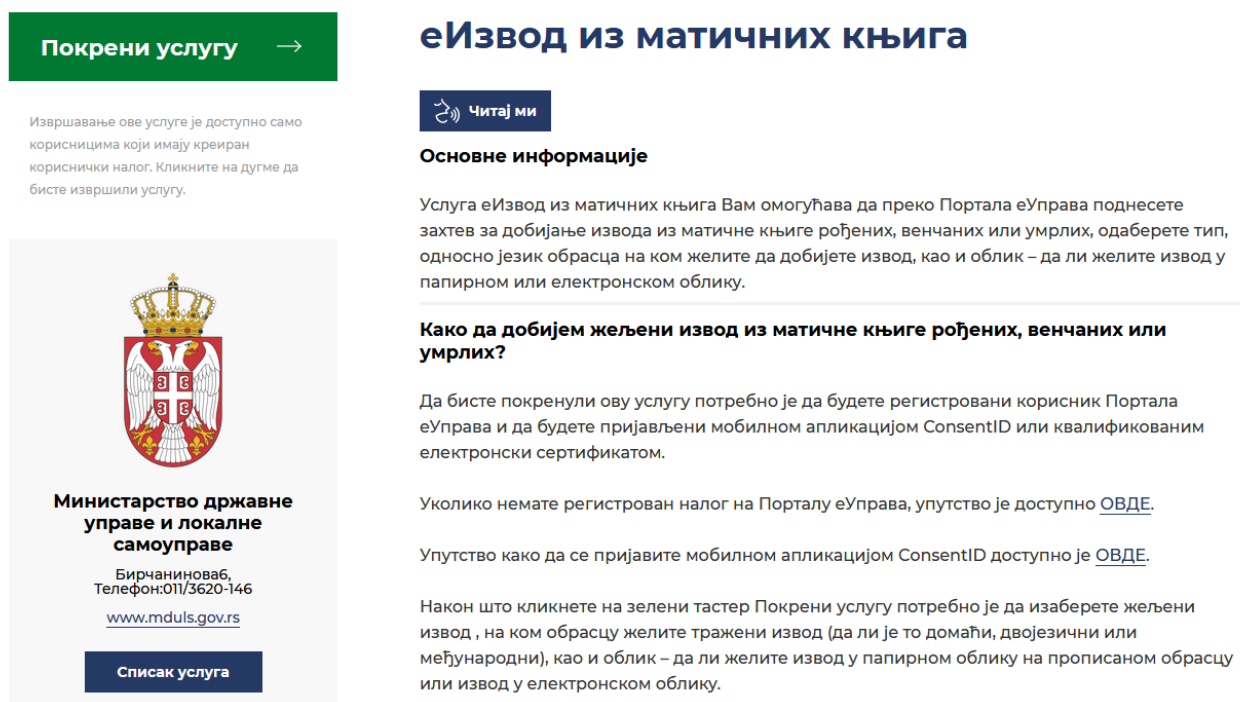
eUprava Republike Srbije predstavlja centralizovano mesto elektronskih usluga za sve građane. Korišćenje ove veb aplikacije građanima olakšava komunikacija sa državnim organima i celokupnom javnom administracijom u smislu lakšeg pronalaženja informacija i odgovarajućih formulara koji su neophodni za završavanje pravnih i drugih obaveza građana, kao i jednostavnije podnošenje zahteva i podizanje ostalih dokumenata.

Cilj ovog sistema jeste da se pojednostavi ciklus prikupljanja potrebnih dokumenata, takođe pravilna upotreba sistema dovodi do smanjenja redova za čekanje u državnim institucijama. U ovom radu je konkretno razvijana veb aplikacija koja podržava sistem eMatičar čija je osnovna svrha izdavanje izvoda iz matičnih knjiga. Aplikacija takođe podržava i podnošenje zahteva za državljanstvo.

2. Srodna rešenja

eUprava je softver pomoću koga je moguće preuzeti dokumente koji sadrže lične podatke u elektronskom obliku. Ideja za kreiranjem ovog sistema potiče od Vlade Republike Srbije. Ovaj softver podržava i sistem eMatičar koji je opisan u ovom radu. Preuzimanje dokumenata se vrši pomoću podnošenja zahteva za izvod iz matične knjige, pri čemu nadležni matičar je u obavezi da u roku od 8 dana dostavi željeni dokument na email adresu ili neki drugi izvor.

Na slici 1 je prikazan izgled eUprave Vlade Republike Srbije.



Слика 1 – eUprava Vlade Republike Srbije

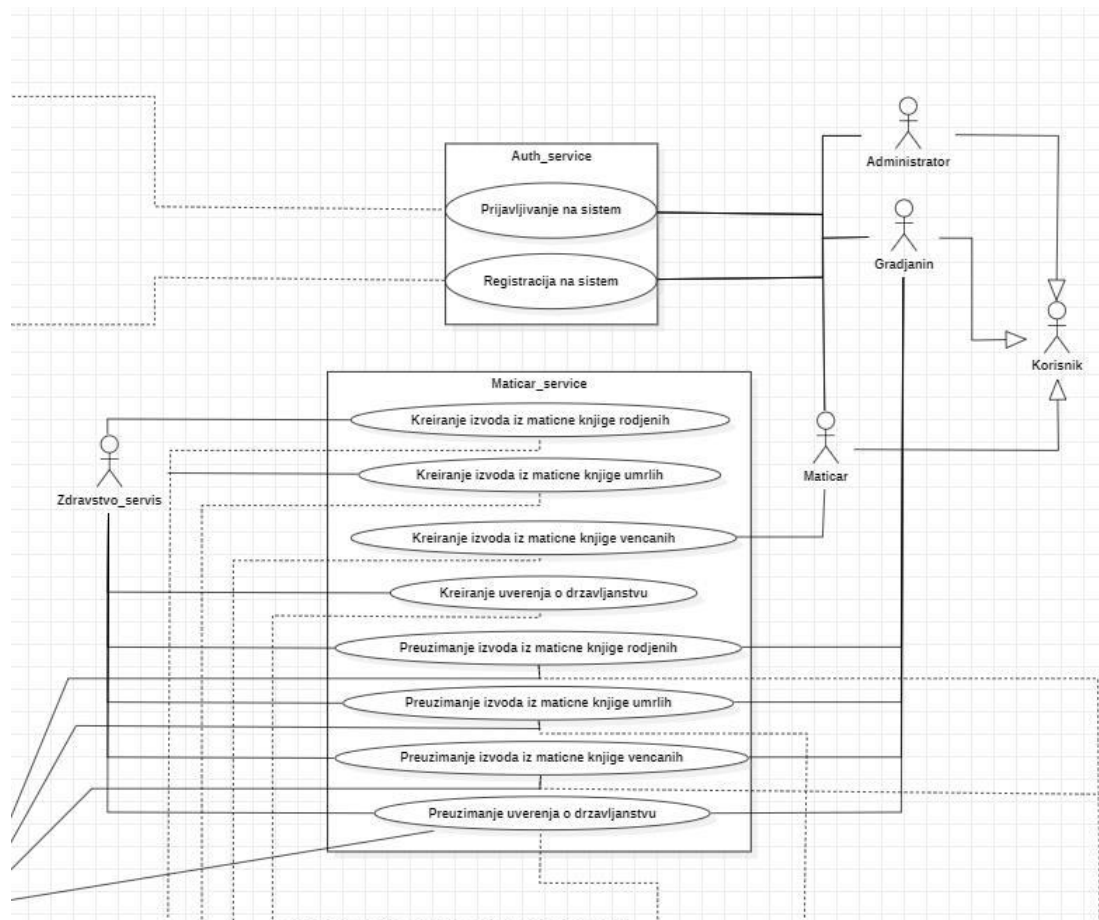
3. Korisćene tehnologije

Ovaj odeljak pruža pregled i objašnjenja tehnologija pomoću kojih je moguće implementirati adekvatno rešenje. Za izradu veb aplikacije je korišćen klasičan šablon koji se sastoji od Frontend dela aplikacije i Backend dela aplikacije. Frontend deo aplikacije implementiran je uz pomoć Angular [\[2\]](#) radnog okvira. Backend deo aplikacije implementiran je uz pomoć GoLand IDE koji omogućava kreiranje Backend dela aplikacije u GoLang [\[3\]](#) programskom jeziku. Perzistencija podataka se vrši unutar MongoDB [\[4\]](#) baze podataka.

4. Specifikacija zahteva

U ovom odeljku su objašnjeni funkcionalni zahtevi softverskog rešenja predstavljenog u ovom radu.

Na slici 2 je prikazan dijagram slučajeva korišćenja.



Slika 2 – Dijagram slučajeva korišćenja

Specifikacija funkcionalnih zahteva:

1. Authservice

U tabeli 1 prikazan je opis slučaja korišćenja “Registracija”.

Naziv	Registracija
Učesnici	Korisnik
Preduslovi	1. Korisnik mora biti evidentiran od strane Matičara 2. Korisnik mora imati jedinstveni JMBG
Koraci	1. Korisnik bira opciju za registraciju

	2. Korisnik unosi svoj JMBG 3. Korisnik unosi svoju lozinku 4. Korisnik ponovo unosi svoju lozinku radi potvrde 5. Korisnik potvrđuje unos
Rezultat	Korisnik je registrovan na sistem
Izuzeci	JMBG već postoji u sistemu

Tabela 1 – Opis slučaja korišćenja “Registracija”

U tabeli 2 prikazan je opis slučaja korišćenja “Prijavljivanje”.

Naziv	Prijavljivanje
Učesnici	Korisnik
Preduslovi	1. Korisnik mora biti registrovan na sistem
Koraci	1. Korisnik bira opciju za prijavu 2. Korisnik unosi JMBG i lozinku 3. Korisnik potvrđuje unos
Rezultat	Korisnik je prijavljen na sistem
Izuzeci	Pogrešan JMBG ili lozinka

Tabela 2 – Opis slučaja korišćenja “Prijavljivanje”

2. Registrar service (Matičar service)

U tabeli 3 prikazan je opis slučaja korišćenja “Dodavanje novorođene osobe”.

Naziv	Dodavanje novorođene osobe
Učesnici	Doktor
Preduslovi	1. Korisnik koji je logovan mora biti Doktor
Koraci	1. Doktor bira opciju za dodavanje novorođene osobe 2. Doktor unosi ime, prezime, JMBG, datum rođenja, mesto rođenja, ime oca, JMBG oca, ime majke, JMBG majke, pol 3. Doktor potvrđuje unos

Rezultati	Korisniku su kreirana osnovna dokumenta
Izuzeci	-

Tabela 3 – Opis slučaja korišćenja “Dodavanje novorođene osobe”

U tabeli 4 prikazan je opis slučaja korišćenja “Preuzimanje izvoda”.

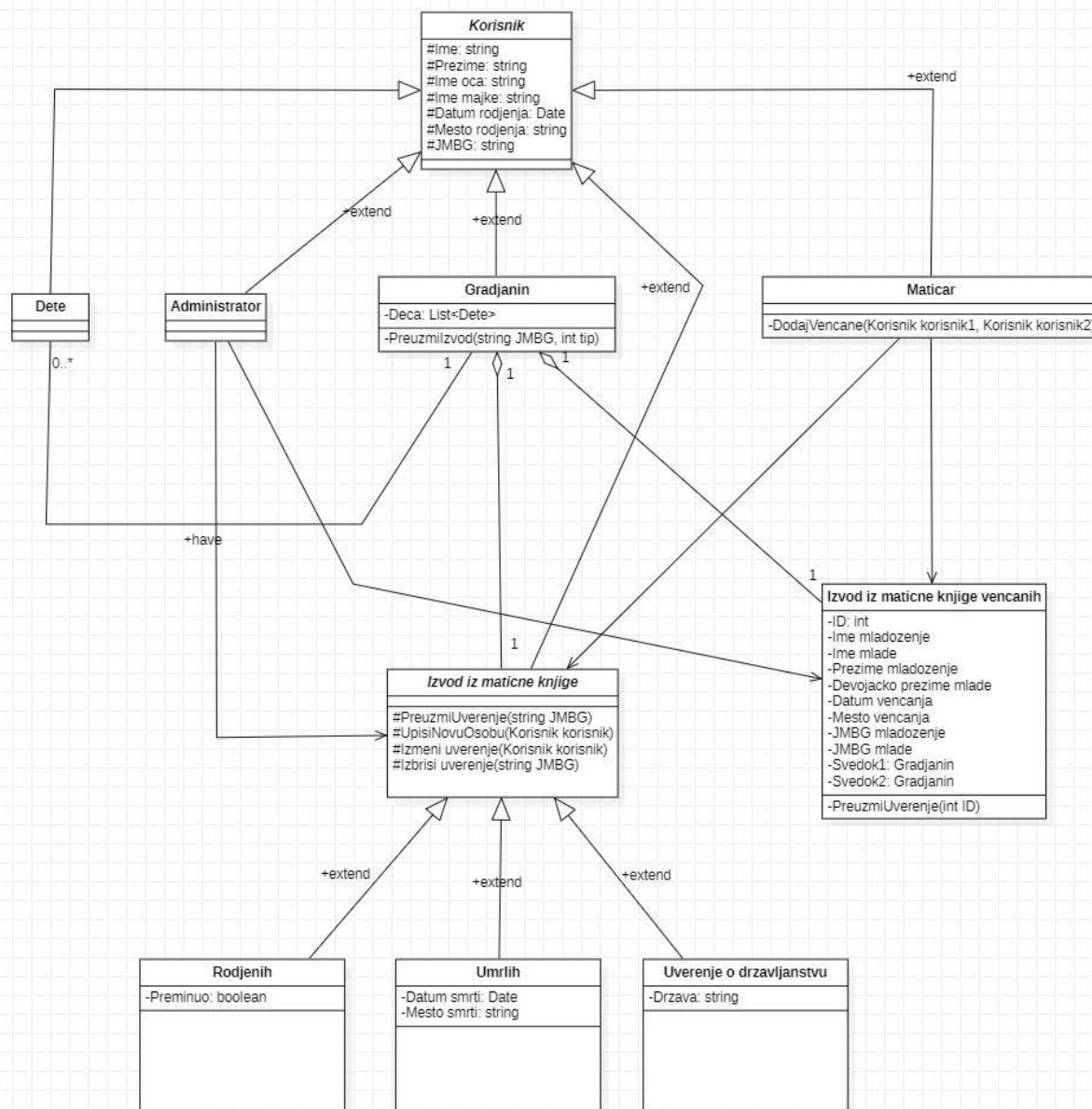
Naziv	Preuzimanje izvoda
Učesnici	Korisnik
Preduslovi	1. Korisnik mora biti prijavljen na sistem
Koraci	1. Korisnik bira opciju za preuzimanje izvoda 2. Korisnik bira jednu od ponuđenih opcija za izvode 3. Korisnik potvrđuje odabir
Rezultati	Korisnik je uspešno preuzeo željeni izvod
Izuzeci	-

Tabela 4 – Opis slučaja korišćenja “Preuzimanje izvoda”

4. Specifikacija dizajna

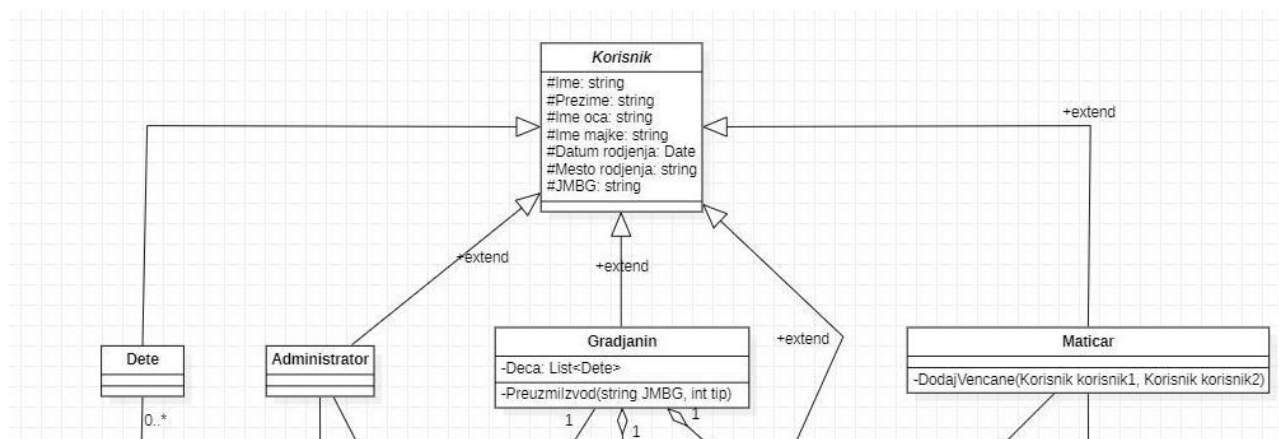
Ovo poglavlje objašnjava dizajn softverskog rešenja sistema eMatičar. U nastavku je predstavljeno pomoću UML Class dijagrama, nad kojim sve entitetima eMatičar vrši manipulaciju podataka.

Na slici 3 je prikazan UML Class dijagram.



Slika 3 – UML Class Dijagram

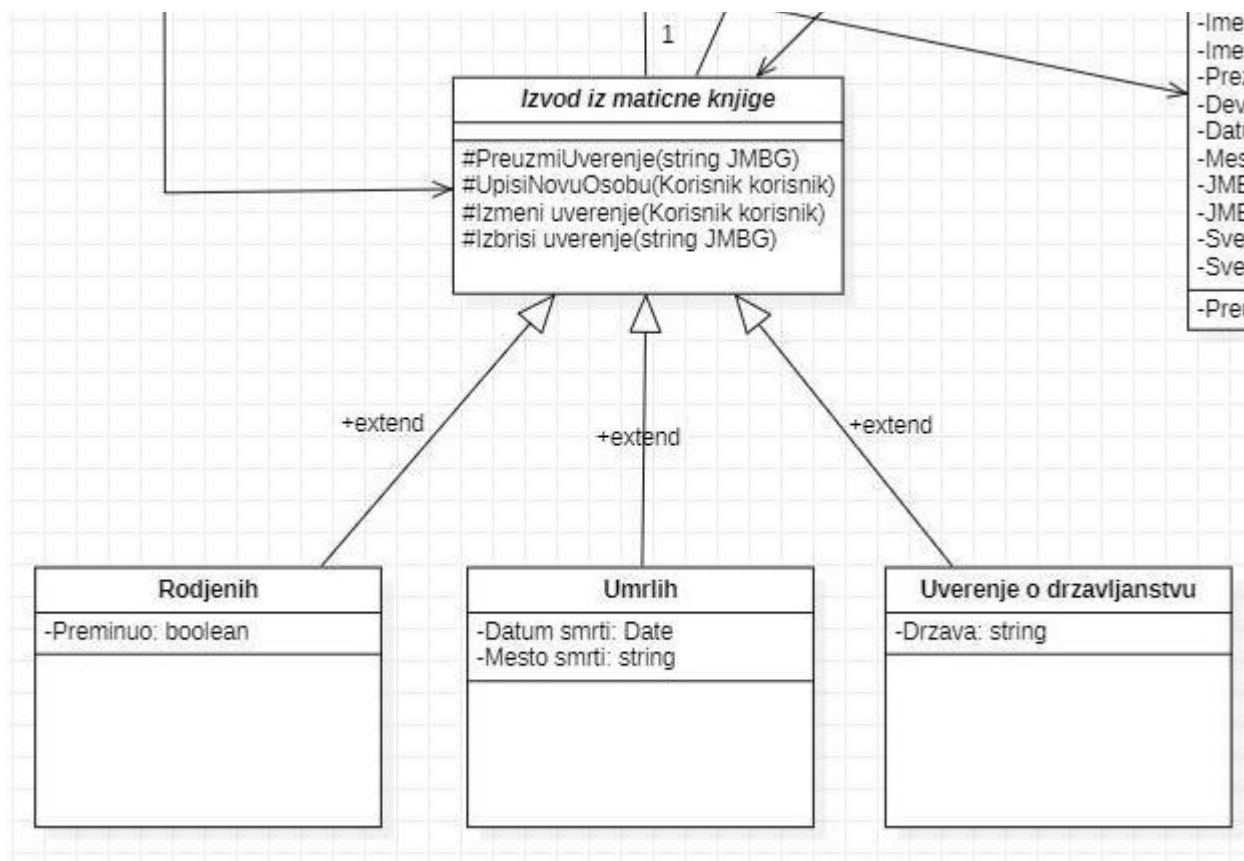
Na slici 4 je prikazana klasa Korisnik i nasledjivanje klase Korisnik.



Slika 4 – Klasa korisnik sa nasledjivanjem

Klasa Korisnik reprezentuje apstraktnu klasu svih korisnika sistema i sadrži njihove lične podatke. Na dijagramu je prikazano da postoje i klase koje su izvedene od klase Korisnik i proširuju njenu baznu strukturu.

Na slici 5 je prikazana klasa Izvod iz matične knjige i nasledjivanje klase Izvod iz matične knjige.



Slika 5 – klasa Izvod iz matične knjige sa nasleđivanjem

Klasa Izvod iz matične knjige reprezentuje apstraktnu klasu izvoda koju svaki korisnik u sistemu poseduje. U slučaju klase Izvod iz matične knjige prikazano je da postoje klase koje su izvedene iz nje i proširuju njenu baznu strukturu. Korisnicima aplikacije su omogućene akcije da preuzmu sva tri navedena izvoda.

5. Implementacija

U ovom odeljku je objašnjena implementacija klijentskog - (Frontend) i serverskog - (Backend) dela veb aplikacije.

1. **Frontend deo aplikacije - Angular**
2. **Backend deo aplikacije – GoLang**

Frontend deo aplikacije je implementiran pomoću Angular radnog okvira koji se u pozadini oslanja na programski jezik TypeScript. **Backend** deo aplikacije je implementiran pomoću GoLang programskog jezika. U ovom poglavlju je predstavljena implementacija funkcionalnosti kao što su registracija, prijavljivanje i preuzimanje određenog izvoda iz matične knjige.

Na slici 6 je prikazan odsečak html koda koji je neophodan za registraciju korisnika na sistem. Prikupljanje unetog teksta unutar input polja se izvršava pomoću “[formGroup]”.

```
1 <section class="vh-100 mt-5">
2   <div class="mask d-flex align-items-center h-80 gradient-custom-3">
3     <div class="container h-100">
4       <div class="row d-flex justify-content-center align-items-center h-100">
5         <div class="col-12 col-md-9 col-lg-7 col-xl-6">
6           <div class="card" style="background-color: #fff; border: 1px solid #ccc; border-radius: 10px; padding: 20px; width: 100%; height: 100%; position: relative; box-shadow: 0 10px 10px #00000033;">
7             <div class="card-body p-5">
8               <h2 class="text-uppercase text-center mb-5">Registrujte vaš nalog</h2>
9
10              <form [formGroup]="formGroup">
11
12                <div class="form-outline mb-4">
13                  <input type="text" id="form3Example1cg" class="form-control form-control-lg" formControlName="jmbg" />
14                  <label class="form-label" for="form3Example1cg">Unesite vaš JMBG</label>
15                </div>
16
17                <div class="form-outline mb-4">
18                  <input type="password" id="form3Example4cg" class="form-control form-control-lg" formControlName="password" />
19                  <label class="form-label" for="form3Example4cg">Unesite vašu šifru</label>
20                </div>
21
22                <div class="form-outline mb-4">
23                  <input type="password" id="form3Example4cdg" class="form-control form-control-lg" formControlName="repeatPassword" />
24                  <label class="form-label" for="form3Example4cdg">Ponovo unesite vašu šifru</label>
25                </div>
26
27                <div class="d-flex justify-content-center">
28                  <button type="button"
29                    class="btn btn-primary btn-block btn-lg gradient-custom-4 text-body" (click)="onSubmit()">Registrujte se</button>
30                </div>
31              </form>
32            </div>
33          </div>
34        </div>
35      </div>
36    </div>
37  </div>
38 </section>
```

Slika 6 – html kod za registraciju

Na slici 7 je prikazan odsečak TypeScript koda koji prikuplja uneti tekst iz input polja koja se nalaze unutar “[formGroup]”. Konkretno vrednosti iz input polja se prikupljaju na osnovu zadatog naziva unutar “formControlName”. Pozivanjem odgovarajućeg servisa pokreće se slanje HTTP zahteva ka Backend delu.

```

1 usage  nananeko1305
onSubmit() {
  const credentials = new Credentials();
  credentials._id = 0
  credentials.jmbg = this.formGroup.get('jmbg')?.value
  credentials.password = this.formGroup.get('password')?.value
  credentials.userType = "Regular"
  if(this.formGroup.get('password')?.value == this.formGroup.get('repeatPassword')?.value){
    this.authService.Registration(credentials).subscribe(
      observer: {
        next: (response) => {
          console.log(response)
          this.openSnackBar( message: "Uspesno ste se registrovali", action: "OK")
          this.router.navigate( commands: ['Login'] ).then()
        },
        error: (error) => {
          console.log(JSON.stringify(error?.error?.text))
          this.openSnackBar(error?.error?.text, action: "OK")
        }
      }
    )
  } else {
    this.openSnackBar( message: "Sifre se ne poklapaju", action: "OK")
  }
}
}

```

Slika 7 – TypeScript logika za registraciju

Na slici 8 je prikazan Auth servis koji je zadužen za kreiranje HTTP zahteva koji u sebi sadrži “RequestBody” tipa “Credentials”. Slanjem definisanog zahteva se ostvaruje komunikacija sa Backend delom i na taj način korisnik biva registrovan na sistem ukoliko njegov JMBG je prethodno evidentiran kod matičara.

```

1 usage  nananeko1305+1
public Registration(credentials: Credentials): Observable<any> {
  return this.http.post( url: `${environment.baseApiUrl}/${this.url}/registration`, credentials);
}

```

Slika 8 – Metoda Registracije unutar Auth Servisa

Nakon što zahtev pristigne na Backend deo, jedan od kontrolera sa odgovarajućim endpoint-om će prihvatiti pristigli zahtev. Na slici 9 je prikazan način obrade pristiglog zahteva.

```

func (controller *AuthController) SignUp(response http.ResponseWriter, request *http.Request) { 1 usage  nananeko1305 +1*

    var credentials domain.Credentials
    err := json.NewDecoder(request.Body).Decode(&credentials)
    if err != nil {
        response.WriteHeader(http.StatusInternalServerError)
        response.Write([]byte("There is problem in decoding JSON"))
        return
    }
    value, err := controller.service.SignUp(credentials)
    if err != nil {
        response.WriteHeader(http.StatusInternalServerError)
        return
    }

    if value == -1 {
        response.WriteHeader(http.StatusAccepted)
        response.Write([]byte("JMBG je vec registrovan!"))
        return
    } else if value == -2 {
        response.WriteHeader(http.StatusCreated)
        response.Write([]byte("JMBG nije pronadjen u izvodima rođenih lica!"))
        return
    }
}

```

Slika 9 – Metoda unutar kontrolera za registraciju

Prilikom prihvatanja zahteva, kreira se nova struktura tipa Credentials na koju se automatski mapiraju vrednosti koji su pristigli u JSON [5] formatu. Na slici 10 je prikazan način automatskog mapiranja JSON vrednosti na konkretna polja strukture. Ukoliko se uspešno izvrši mapiranje vrednosti na strukturu, novokreirana struktura se prosleđuje servisnom sloju.

```

type Credentials struct { 9 usages  nananeko1305
    ID      primitive.ObjectID `bson:"_id" json:"id"`
    JMBG     string              `bson:"jmbg" json:"jmbg" validate:"onlyCharAndNum,required"`
    Password string              `bson:"password" json:"password" validate:"onlyCharAndNum,required"`
    UserType UserType          `bson:"userType" json:"userType" validate:"onlyChar"`
}

```

Slika 10 – Struktura Credentials

Na slici 11 je prikazan servisni sloj, odnosno Auth servis, unutar koga je implementirana logika za upisivanje u bazu.

```

func (service *AuthService) SignUp(credentials domain.Credentials) (int, error) { 1 usage  nananeko1305 +1*

    dataToSend, err := json.Marshal(credentials)

    response, err := service.natsConnection.Request(os.Getenv("key: \"CHECK_USER_JMBG\""), dataToSend, 5*time.Second)

    var isJMBGExist bool
    err = json.Unmarshal(response.Data, &isJMBGExist)
    if err != nil {
        log.Println("Error in unmarshal json")
        return 0, err
    }

    if isJMBGExist {
        isExists := service.IsJMBGUnique(credentials.JMBG)
        if isExists == true { -1, nil }

        credentials.ID = primitive.NewObjectID() //creating unique UUID for MongoDB
        password, err := bcrypt.GenerateFromPassword([]byte(credentials.Password), bcrypt.DefaultCost) //hashing password
        credentials.Password = string(password)
        if err != nil { 0, err }
        service.store.SignUp(credentials)
        return 0, nil
    } else { -2, nil }
}

```

Slika 11 – Metoda unutar Auth servisa za registraciju

Pri samom ulazu u metodu unutar servisnog sloja, korišćenjem Message Broker-a NATS [6], vrši se provera da li pristigli JMBG postoji unutar baze koja se nalazi u Registrar servisu odnosno servisu eMatičar. Dobijanjem povratne poruke koja je tipa Boolean, tj. **true**, vrši se dodavanje kredencijala unutar baze Auth servisa i korisnik biva registrovan na sistem pri čemu njegova lozinka se hash-ira pomoću Bcrypt [7] biblioteke. U slučaju da pristigli odgovor ima vrednosti **false**, korisniku će biti ispisana poruka da je već registrovan na sistem.

Ukoliko se korisnik uspešno registrovao, biće mu omogućen pristup sistemu. Neposredno pre pristupanja sistemu, potrebno je da se uloguje na sistem. Na slici 12 je prikazan html odsečak koji pomoću “formGroup” prikuplja vrednosti iz input polja.

```

<section class="vh-100 mt-5">
  <div class="mask d-flex align-items-center h-80 gradient-custom-3">
    <div class="container h-100">
      <div class="row d-flex justify-content-center align-items-center h-100" >
        <div class="col-12 col-md-9 col-lg-7 col-xl-6">
          <div class="card" style="...">
            <div class="card-body p-5">
              <h2 class="text-uppercase text-center mb-5">Prijavite se</h2>

              <form [formGroup]="formGroup">

                <div class="form-outline mb-4">
                  <input type="text" id="form3Example1cg" class="form-control form-control-lg" formControlName="jmbg" />
                  <label class="form-label" for="form3Example1cg">Unesite vaš JMBG</label>
                </div>

                <div class="form-outline mb-4">
                  <input type="password" id="form3Example4cg" class="form-control form-control-lg" formControlName="password"/>
                  <label class="form-label" for="form3Example4cg">Unesite vašu šifru</label>
                </div>

                <div *ngIf="notFound" class="not-found">
                  <h3>Nepravilan JMBG ili šifra</h3>
                </div>

                <div class="d-flex justify-content-center">
                  <button type="button"
                    class="btn btn-primary btn-block btn-lg gradient-custom-4 text-body" (click)="onSubmit()">Prijavite se</button>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  </section>

```

Slika 12 – html stranica za prijavljivanje

Prilikom pozivanja metode “onSubmit()”, povući će se svi podaci iz input polja na osnovu “formControlName”. Na slici 13 je prikazan način povlačenja vrednosti iz input polja.

```

onSubmit() {
  this.credentials.jmbg = this.formGroup.get('jmbg')?.value
  this.credentials.password = this.formGroup.get('password')?.value
  this.authService.Login(this.credentials).subscribe(
    ({
      next: (response) => {
        if (response != null){
          if (response == "JMBG not exist!"){
            localStorage.clear()
          }else if (response == "Password doesn't match!"){
            localStorage.clear()
          }else{
            localStorage.setItem('authToken', response)
            this.router.navigate( commands: ['/chose-service']).then();
          }
        }
      },
      error: (error) => {
        localStorage.clear()
        if (error.status = 403) {
          this.notFound = true;
        }
      }
    })
  );
}

```

Slika 13 – TypeScript logika za prijavljivanje

Prilikupljene vrednosti se dodeljuju poljima unutar klase Credentials i prosleđuju se servisu koji će kreirati HTTP zahtev sa POST metodom i smestiće unutar RequestBody klasu Credentials. Na slici 14 je prikazana metoda za prijavljivanje unutar Auth servisa.

```

3 usages  nananeko1305
public Login(credentials: Credentials): Observable<any> {
  return this.http.post(<url>: `${environment.baseApiUrl}/${this.url}/login`, credentials, options: {responseType: 'text'});
}

```

Slika 14 – Metoda za prijavljivanje unutar Auth Servisa

Nakon što zahtev pristigne na Backend deo, jedan od kontrolera sa odgovarajućim endpoint-om će prihvatiti pristigli zahtev. Na slici 15 je prikazan način obrade pristiglog zahteva.

```

func (controller *AuthController) Login(response http.ResponseWriter, request *http.Request) {

    var credentials domain.Credentials
    err := json.NewDecoder(request.Body).Decode(&credentials)
    if err != nil {
        response.WriteHeader(http.StatusInternalServerError)
        response.Write([]byte("There is problem in decoding JSON"))
        return
    }

    token, value := controller.service.Login(credentials.JMBG, credentials.Password)
    if value == 1 {
        response.WriteHeader(http.StatusNotFound)
        response.Write([]byte("JMBG not exist!"))
        return
    } else if value == 2 {
        response.WriteHeader(http.StatusAccepted)
        response.Write([]byte("Password doesn't match!"))
        return
    } else if value == 3 {
        response.WriteHeader(http.StatusInternalServerError)
        response.Write([]byte("Problem with generating token"))
        return
    }

    response.WriteHeader(http.StatusOK)
    response.Write([]byte(token))
}

```

Slika 15 – Metoda unutar kontrolera za Prijavljivanje

Na slici 16 je prikazan servisni sloj, odnosno Auth servis, unutar koga je implementirana logika za generisanje JWT [\[8\]](#) tokena.


```

func (service *AuthService) Login(jmbg string, password string) (string, int) { 1 usage  nananeko1305

    credentials, err := service.store.GetCredentials(jmbg)
    if err != nil {
        log.Println(err)
        return "", 1
    }

    err = bcrypt.CompareHashAndPassword([]byte(credentials.Password), []byte(password))
    if err != nil {
        log.Println(err)
        return "", 2
    }

    tokenString, err := GenerateJWT(credentials)
    if err != nil : "", 3

    return tokenString, 0
}

func GenerateJWT(credentials *domain.Credentials) (string, error) { 1 usage  nananeko1305
    key := []byte(os.Getenv(key: "SECRET_KEY"))
    signer, err := jwt.NewSignerHS(jwt.HS256, key)
    if err != nil {
        log.Println(err)
    }

    builder := jwt.NewBuilder(signer)

    claims := &domain.Claims{
        UserID:    credentials.ID,
        JMBG:      credentials.JMBG,
        Role:      credentials.UserType,
        ExpiresAt: time.Now().Add(time.Minute * 60),
    }

    token, err := builder.Build(claims)
    if err != nil {
        log.Println(err)
    }

    return token.String(), nil
}

```

Slika 16 – Metoda unutar Auth servisa za prijavljivanje i metoda za kreiranje JWT tokena

Nakon prosleđivanja kredencijala servisnom sloju, prvi korak jeste provera da li kredencijali postoje unutar baze podataka. Ako su kredencijali validni, vrši se proveravanje hash-irane vrednosti lozinke unutar baze i pristigle lozinke u plain text formatu. Ukoliko se lozinke poklapaju dalje se vrši pozivanje metode “GenerateJWT()” čiji je zadatak da kreira JWT token koji će u sebi sadržati Id korisnika koji želi da se loguje na sistem, JMBG korisnika, ulogu korisnika u sistemu i vreme kada će JWT token isteći.

Prilikom vraćanja odgovora na Frontend deo, JWT će biti smešten unutar LocalStorage pod ključem “authToken” kao što je prikazano na slici 12 i na taj način je korisnik prijavljen na sistem.

Uspešno prijavljivanje na sistem omogućava korisniku da preuzme svoje dokumente tj. Izvode iz matične knjige.

Svaki korisnik poseduje tri vrste izvoda:

1. Izvod iz matične knjige rođenih,
2. Izvod iz matične knjige umrlih(samo ukoliko je osoba preminula) i
3. Uverenje o državljanstvu.

Na slici 17 je prikazan html stranica za odabir izvoda za preuzimanje.

```
<div>
  <h1 class="d-flex justify-content-center mt-3" style="...">Koji izvod zelite da preuzmete?</h1>
</div>
<div class="row w-75 d-flex justify-content-center mt-5" style="...">
  <div class="col d-flex justify-content-center">
    <button class="btn btn-primary" (click)="GetCertificate( type: '1')">Izvod iz maticne knjige rodjenih</button>
  </div>
  <div class="col d-flex justify-content-center">
    <button class="btn btn-primary" (click)="GetCertificate( type: '2')">Izvod iz maticne knjige umrlih</button>
  </div>
  <div class="col d-flex justify-content-center">
    <button class="btn btn-primary" (click)="GetCertificate( type: '3')">Uverenje o drzavljanstvu</button>
  </div>
</div>
```

Slika 17 - html za odabir izvoda

Odabirom određenog izvoda, poziva se metoda “GetCertificate()” koja kao parametar prima string vrednost na osnovu koje se određuje vrsta izvoda koja će biti preuzeta iz baze podataka koja je deo Registrar servisa.

Na slici 18 je prikazano pozivanje metode “GetCertificate()” koja na osnovu parametra dobavlja podatke o korisniku i generiše željeni izvod iz matične knjige.

```

3 usages  nananeko1305 *
GetCertificate(type: String) {
  this.registrarService.GetCertificate(type).subscribe(
    observer: {
      next: (value) => {
        if(value==null){
          this.openSnackBar( message: "Korisnik nije preminuo!", action: "OK")
        }else{
          this.response = value
        }
      }
    }
  )
}

```

Slika 18 – slanje HTTP zahteva za dobijanje određenog izvoda

U zavisnosti od tipa, Backend deo na nivou servisnog sloja će kreirati i vratiti odgovarajući izvod. Na slikama 19, 20 i 21 je prikazano kreiranje nove strukture u zavisnosti od tipa traženog izvoda.

```

if certificateType == 1 {

  var certificate entity.BirthCertificate
  certificate.Ime = user.Ime
  certificate.Prezime = user.Prezime
  certificate.ImeOca = user.ImeOca
  certificate.JMBGOca = user.JMBGOca
  certificate.ImeMajke = user.ImeMajke
  certificate.JMBGMajke = user.JMBGMajke
  certificate.DatumRodjenja = user.DatumRodjenja
  certificate.MestoRodjenja = user.MestoRodjenja
  certificate.JMBG = user.JMBG
  certificate.Pol = user.Pol

  return &certificate, nil, nil
}

```

Slika 19 – izvod iz matične knjige rođenih

```

} else if certificateType == 2 {

    if user.Premينو == true {
        var certificate entity.ExtractFromTheDeathRegister
        certificate.Ime = user.Ime
        certificate.Prezime = user.Prezime
        certificate.ImeOca = user.ImeOca
        certificate.JMBGOca = user.JMBGOca
        certificate.ImeMajke = user.ImeMajke
        certificate.JMBGMajke = user.JMBGMajke
        certificate.DatumRodjenja = user.DatumRodjenja
        certificate.MestoRodjenja = user.MestoRodjenja
        certificate.JMBG = user.JMBG
        certificate.Pol = user.Pol
        certificate.DatimSmrti = user.DatimSmrti
        certificate.MestoSmrti = user.MestoSmrti

        //slanje nazad

        return nil, &certificate, nil
    }
}

```

Slika 20 – izvod iz matične knjige umrlih

```

} else if certificateType == 3 {

    var certificate entity.CertificateOfCitizenship
    certificate.Ime = user.Ime
    certificate.Prezime = user.Prezime
    certificate.ImeOca = user.ImeOca
    certificate.JMBGOca = user.JMBGOca
    certificate.ImeMajke = user.ImeMajke
    certificate.JMBGMajke = user.JMBGMajke
    certificate.DatumRodjenja = user.DatumRodjenja
    certificate.MestoRodjenja = user.MestoRodjenja
    certificate.JMBG = user.JMBG
    certificate.Pol = user.Pol
    certificate.Drzava = user.Drzava

    //slanje nazad

    return nil, nil, &certificate
}

```


Slika 21 – uverenje o državljanstvu

6. Demonstracija

Ovo poglavlje prikazuje korišćenje aplikacije za registraciju, prijavljivanje i dobavljanje izvoda iz matične knjige. Pristup sistemu je moguć putem veb pretraživača na računaru.

Registracija

Na slici 22 je demonstrirana funkcionalnost Registracija.




The screenshot shows a web application interface with a dark blue header bar. On the right side of the header, the words 'PRIJAVA' and 'REGISTRACIJA' are visible. The main content area is white and contains a registration form titled 'REGISTRUJTE VAŠ NALOG'. The form has three input fields: the first contains '111' with the label 'Unesite vaš JMBG' below it; the second contains three dots with the label 'Unesite vašu šifru' below it; the third contains three dots with the label 'Ponovo unesite vašu šifru' below it. A blue button labeled 'Registrujte se' is positioned at the bottom of the form.

Slika 22 – demonstracija registracije

Prilikom odabira stranice “Registracija” unutar navigacione trake, otvara se forma za Registraciju korisnika. Korisnik je dužan da popuni sva polja pri čemu unosi svoj JMBG i dva puta unosi lozinku pomoću koje će moći da pristupi sistemu. Nakon što se sva polja popune, potrebno je potvrditi registraciju klikom na dugme “Registrujte se”.

Nakon potvrde registracije, korisnik biva preusmeren na stranicu za prijavljivanje.

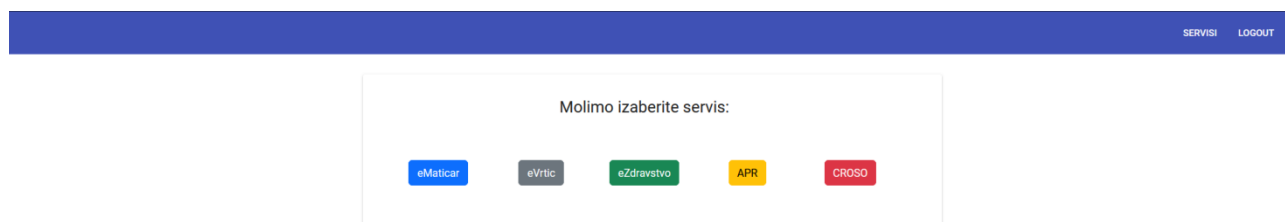
Na slici 23 je prikazana forma za prijavljivanje.



The image shows a web application interface with a dark blue header bar. On the right side of the header, there are two links: "PRIJAVA" and "REGISTRACIJA". Below the header, centered on the page, is a white rectangular box with rounded corners. Inside this box, the title "PRIJAVITE SE" is displayed at the top. Below the title, there are two input fields. The first field contains the text "111" and has the label "Unesite vaš JMBG" below it. The second field contains three dots "..." and has the label "Unesite vašu šifru" below it. At the bottom of the box, there is a blue button with the text "Prijavite se" in white.

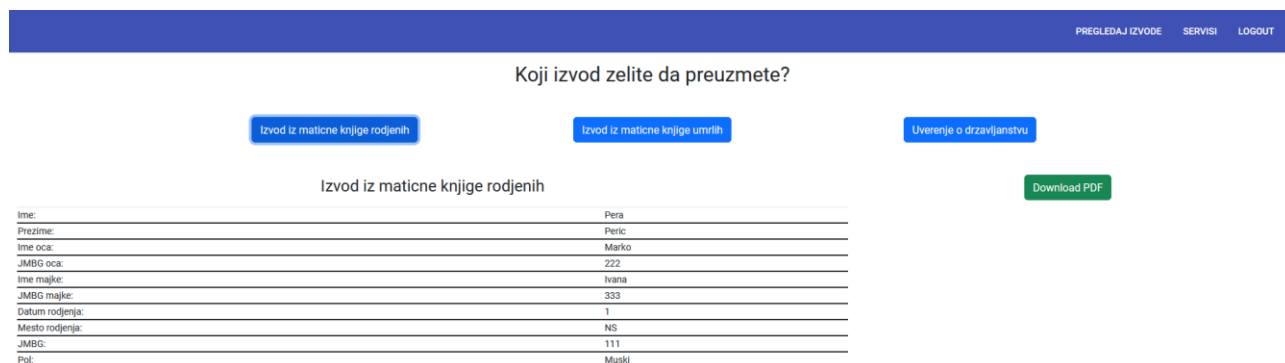
Slika 23 – forma za prijavljivanje korisnika

Forma za prijavljivanje se popunjava tako što korisnik je dužan da unese JMBG i lozinku pomoću koje se registrovao na sistem. Ukoliko su kredencijali validni, korisnik biva prebačen na sledeću stranicu gde treba odabrati servis eMatičar kao što je prikazano na slici 24.



Slika 24 – odabir eMatičar servisa

Nakon uspešno odabranog servisa, korisnik dobija mogućnost odabira izvoda koje može preuzeti. Na slici 25 je prikazan primer odabira izvoda iz matične knjige rođenih.



Slika 26 – preuzimanje izvoda iz matične knjige rođenih

Nakon odabira Izvoda iz matične knjige rođenih, korisniku su u tabelarnom prikazu prikazani njegovi lični podaci. Korisniku je dostupna i opcija preuzimanja

dokumenta u PDF formatu. Akcija preuzimanja PDF fajla se ostvaruje odabirom dugmeta “Download PDF”.

Na slici 27 je prikazan izgled PDF fajla preuzetog izvoda iz matične knjige rođenih.

Izvod iz matične knjige rođenih	
Ime:	Pera
Prezime:	Perić
Ime oca:	Marko
JMBG oca:	222
Ime majke:	Ivana
JMBG majke:	333
Datum rođenja:	1
Mesto rođenja:	NS
JMBG:	111
Pol:	Muski

Slika 27 – izvod iz matične knjige rođenih u PDF formatu

7. Zaključak

U ovom radu je predstavljeno softversko rešenje za registraciju i prijavljivanje na sistem eUprave i predstavljeno je preuzimanje potrebne dokumentacije u elektronskom obliku i demonstriran je rad prototipske aplikacije. Prikazano rešenje olakšava korisnicima da putem veb aplikacije preuzmu potrebnu dokumentaciju bez čekanja u redovima u nekim od državnih institucija. Sistem prikazan u ovom radu, u poređenju sa srodnim rešenjima, pruža jednostavnost pri korišćenju, a istovremeno obezbeđuje funkcionalnosti.

8. Reference

- [1] eUprava [Online]. Dostupno na: <https://euprava.gov.rs/>
- [2] Angular framework [Online]. Dostupno na: <https://angular.io/>
- [3] Go programski jezik [Online]. Dostupno na: <https://go.dev/>
- [4] MongoDB baza podataka [Online]. Dostupno na: <https://www.mongodb.com/>
- [5] JavaScript Object Notation (JSON) [Online]. Dostupno na: <https://www.json.org/json-en.html>
- [6] NATS message broker [Online]. Dostupno na: <https://nats.io/>
- [7] Bcrypt hash-ing funkcija [Online]. Dostupno na: <https://pkg.go.dev/golang.org/x/crypto/bcrypt>
- [8] JSON Web Token (JWT) [Online]. Dostupno na: <https://jwt.io/>