Autorizacija i verifikacija

Leko, Ilija

VELEUČILIŠTE U BJELOVARU
PREDDIPLOMSKI STRUČNI STUDIJ RAČUNARSTVO

AUTORIZACIJA I VERIFIKACIJA

Završni rad br. 08/RAČ/2021

Ilija Leko

Bjelovar, srpanj 2021.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Kandidat: **Leko Ilija** Datum: 27.08.2021. Matični broj: 001977

JMBAG: 0314019324

Kolegij: **BAZE PODATAKA**

Naslov rada (tema): **Autorizacija i verifikacija**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Programsko inženjerstvo**

Mentor: **Tomislav Adamović, mag.ing.el.** zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **Ante Javor, struč.spec.ing.comp., predsjednik**
2. **Tomislav Adamović, mag.ing.el., mentor**
3. **Krunoslav Husak, dipl.ing.rač., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 08/RAČ/2021

U radu je potrebno izraditi sustav za autorizaciju i verifikaciju procesa unutar aplikacije koristeći Oracle bazu podataka. U sklopu autorizacije potrebna je implementacija autentifikacije pomoću PHP tehnologije. Sve autorizacijske uloge i autorizaciju prikazati u testnim ispitivanjima primjenjujući Postman alat.

Zadatak uručen: 27.08.2021. Mentor: **Tomislav Adamović, mag.ing.el.**

Sadržaj

1. UVOD

Zbog napretka tehnologije i potrebe za digitalizacijom danas se zahtjeva online pristup raznim sustavima. Kako bi svakodnevno korištenje online servisa bilo sigurno, koristi se autorizacija i verifikacija. Autorizacija i verifikacija su sigurnosni alati koji omogućuju i osiguravaju siguran i ovlašten pristup te upravljanje pojedinim procesima u online sustavima. Navedeni postupci koriste se u gotovo svim računalnim sustavima kako bi se korisnici zaštitili od neovlaštenog pristupa podacima i krađe identiteta. Da bi se proces autorizacije mogao odvijati potrebna je autentifikacija kojom korisnik dokazuje svoj identitet. Nakon autentifikacije sustav prilikom svakog pokušaja pristupa aplikacijama ili podacima provjerava ima li prijavljeni korisnik ovlasti za pristup tim resursima. Postoji više načina autentifikacije, a najčešći je autentifikacija zaporkom. Ovlaštenja, odnosno prava pristupa razlikuju se ovisno o kojoj grupi korisnika se radi, npr. zaposlenici tvrtke će imati veće ovlasti od anonimnog korisnika sustava, dok će administratori imati neograničen pristup resursima.

Cilj ovog rada je upoznavanje s procesima autorizacije i verifikacije te njihova implementacija na pozadinskom sloju korištenjem Oracle baze podataka. Za pristup bazi podataka koristi se PHP serverska tehnologija, a izgled i funkcionalnosti korisničkog sučelja implementirani su koristeći HTML, CSS, JavaScript, Bootstrap i jQuery. Za pojedine testove sustava korišten je Postman.

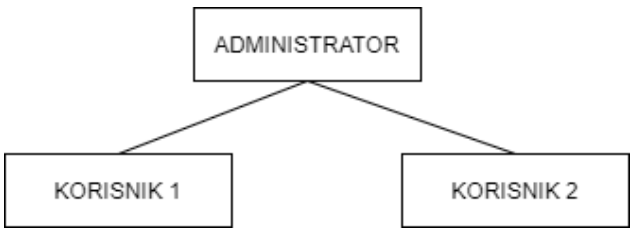
Prvo poglavlje govori o autorizaciji, mogućnostima i primjerima implementacije. U drugom poglavlju opisani su pojmovi autentifikacije i sesije te njihove izvedbe u praksi. Treće poglavlje opisuje proces verifikacije. U četvrtom poglavlju govori se o Oracle bazi podataka, alatima za upravljanje i PL/SQL jeziku koji se koristi u radu sa spomenutom bazom. Šesto poglavlje je prikaz implementacije autentifikacije, autorizacije i verifikacije na zamišljenom poslovnom sustavu. U sedmom poglavlju prikazane su inicijalne postavke sustava koje je potrebno napraviti da bi se sustav mogao koristiti. Osmo poglavlje prikazuje Postman testove nad razvijenim sustavom.

2. AUTORIZACIJA

Autorizacija predstavlja dodjelu i provjeru prava pristupa korisnika koji želi pristupiti određenim resursima. Prava pristupa se najčešće sastoje od više razina. Da bi se dodijelile određene ovlasti korisniku potrebna je njegova autentifikacija kako bi sustav imao informaciju o identitetu korisnika koji pokušava pristupiti određenim resursima.

2.1 Uloge korisnika

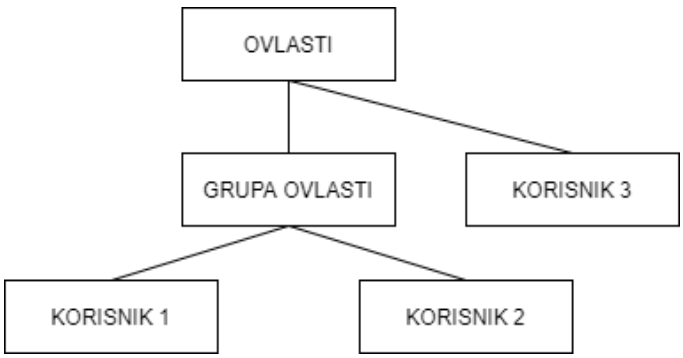
Pojedine uloge korisnika mogu se razlikovati ovisno o dijelovima sustava za koje korisnik ima pristup. Unutar sustava korisniku se dodjeljuje pristup aplikaciji i razina ovlasti, odnosno uloga. Administratori najčešće imaju prava pristupa svim informacijama za čitanje, pisanje i brisanje. Slika 2.1 prikazuje nasljeđivanje unaprijed definiranih prava administratora.



Slika 2.1: Dijagram nasljeđivanja administratorskih ovlasti

2.2 Grupe ovlasti

Zbog pojednostavljenja dodjele i uklanjanja ovlasti koriste se grupe ovlasti. Grupa ovlasti označava skup ovlasti koje se mogu dodijeliti nekom korisniku. Dodavanjem korisnika u grupu ovlasti dodjeljuju se korisniku sva prava koja ima grupa. Dijagramom prikazanim slikom 2.2 može se vidjeti kako korisnici mogu naslijediti prava definirana unutar neke grupe ili samo neku pojedinačnu ovlast.



Slika 2.2: Dijagram nasljeđivanja pojedinačnih i grupnih ovlasti

3. AUTENTIFIKACIJA I SESIJA

3.1 Autentifikacija

Autentifikacija je postupak u kojem se utvrđuje identitet korisnika koji pristupa određenom servisu. U realnom svijetu se za provjeru identiteta osobe najčešće koristi osobna iskaznica, putovnica ili vozačka dozvola, dok se u virtualnom svijetu identifikacija obavlja na drugačije načine. Najčešći oblik autentifikacije danas je autentifikacija pomoću zaporke. U sustavima koji zahtijevaju veliku razinu sigurnosti koriste se dvofaktorske autentifikacije, biometrija, razni fizički tokeni i slično. Dvofaktorska autentifikacija predstavlja dvostruku zaštitu koristeći kombinaciju dva različita koraka provjere identiteta. Primjer jedne takve kombinacije je korištenje zaporke i sigurnosnog tokena koji prilikom potvrde identiteta zaporkom stiže na mobitel ili mail adresu. Uz zaporku danas se sve više koriste biometrijske provjere kao što su otisak prsta ili skeniranje lica.

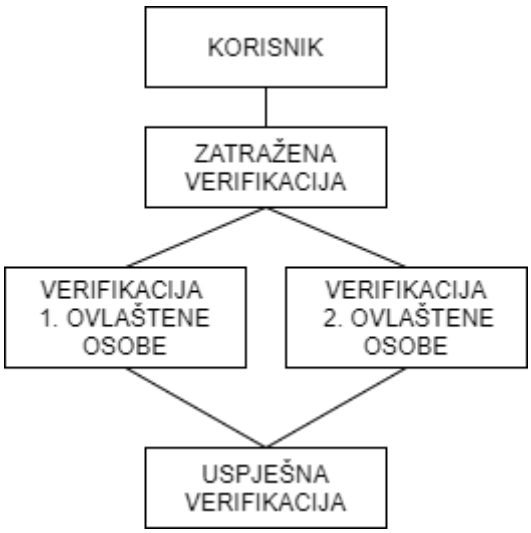
3.2 Sesija

Korisnik aktivira svoju sesiju prilikom posjete web stranice ili uspješnom autentifikacijom. Sesije se mogu koristiti za više namjena, a najčešće za prikupljanje podataka o prijavljenom korisniku i informacija o njegovoj aktivnosti na stranici. Prikupljeni podaci čuvaju se u kolačićima (engl. *cookies*) ili na poslužitelju. Posjedovanjem autentifikacijskog kolačića sustav zna da je korisnik prijavljen te na osnovu tih podataka provjerava prava pristupa. Svakim novim pokušajem pristupa šalje se pohranjeni kolačić s klijentovog računala prema poslužitelju.

Trajanje sesije na PHP servisima ovisi o postavkama unutar *php.ini* datoteke. Ako neko određeno vrijeme korisnik ne posjeti stranicu, sesija će biti obrisana te će morati ponovno obaviti autentifikaciju. Prema [1] prikazane su postavke u PHP sesijama koje je moguće izmijeniti prema potrebama sustava.

4. VERIFIKACIJA

Verifikacija je proces u kojem se zahtjeva odobrenje jedne ili više ovlaštenih osoba kako bi došlo do promjene unutar sustava. Proces započinje kada korisnik napravi akciju za koju je potrebna verifikacija. Ovisno o važnosti i postavkama sustava, zatražuje se verifikacija od strane ovlaštenih osoba kako bi se izvršila akcija koju je korisnik zatražio kao što je prikazano slikom dijagrama 4.1.



Slika 4.1: Dijagram tijeka procesa verifikacije

Za realizaciju verifikacijskog sustava potrebno je imati autorizacijski sustav kako bi se u svakom trenutku znalo koji korisnik je prijavljen u sustav i koje ovlasti ima. Zatim, na temelju identiteta korisnika provjerava se ima li ovlasti za verificiranje unutar neke aplikacije.

4.1. Primjena verifikacije

Verifikacija se primjenjuje u raznim sustavima. Jedan od primjera može biti bankarski sustav u kojem se potvrda za odobrenje nekog zahtjeva može dobiti verifikacijom jedne ili više osoba. U tom slučaju verifikacija predstavlja potpis osobe koja je odobrila zahtjev. Tako se procesi unutar banke pojednostavljuju i ubrzavaju. Primjena verifikacije može se razlikovati ovisno o potrebama pojedinih sustava. U dostavnim tvrtkama može se implementirati kao niz potvrda o paketu kao što su: dostupnost, najava dostave i status dostave. Kada se ispune sve točke verifikacije, smatra se da je proces naručivanja paketa uspješno prošao fazu od narudžbe do dostave i time se zaključuje narudžba.

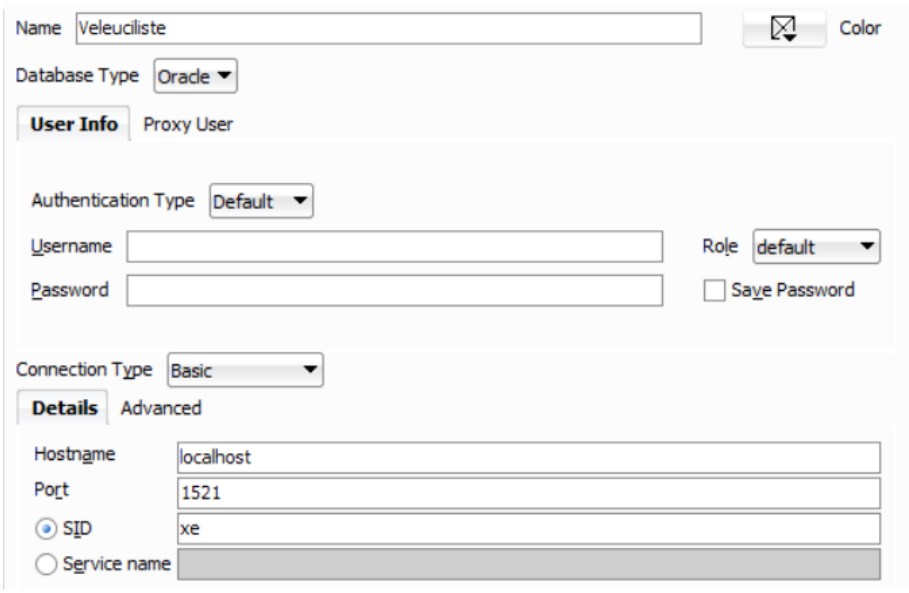
5. ORACLE BAZA PODATAKA I ALATI

5.1 Povijest Oraclea

Tvrtka Oracle osnovana je 1977. godine od strane Larryja Ellisona, Boba Minera i Ed Oatesa sa sjedištem u Kaliforniji pod nazivom Software Development Laboratories. Ime je mijenjala ime u Relational Software pa u sada standardno Oracle Systems Corporation. Ključno za Oracle je što su prvi razvili globalno dostupan sustav za upravljanje bazama podataka temeljenih na relacijskom modelu. Prema [2] od svoga osnivanja do danas jedna su od vodećih tvrtki koja se bavi razvojem sustava za upravljanje podacima.

5.2 Oracle SQL Developer

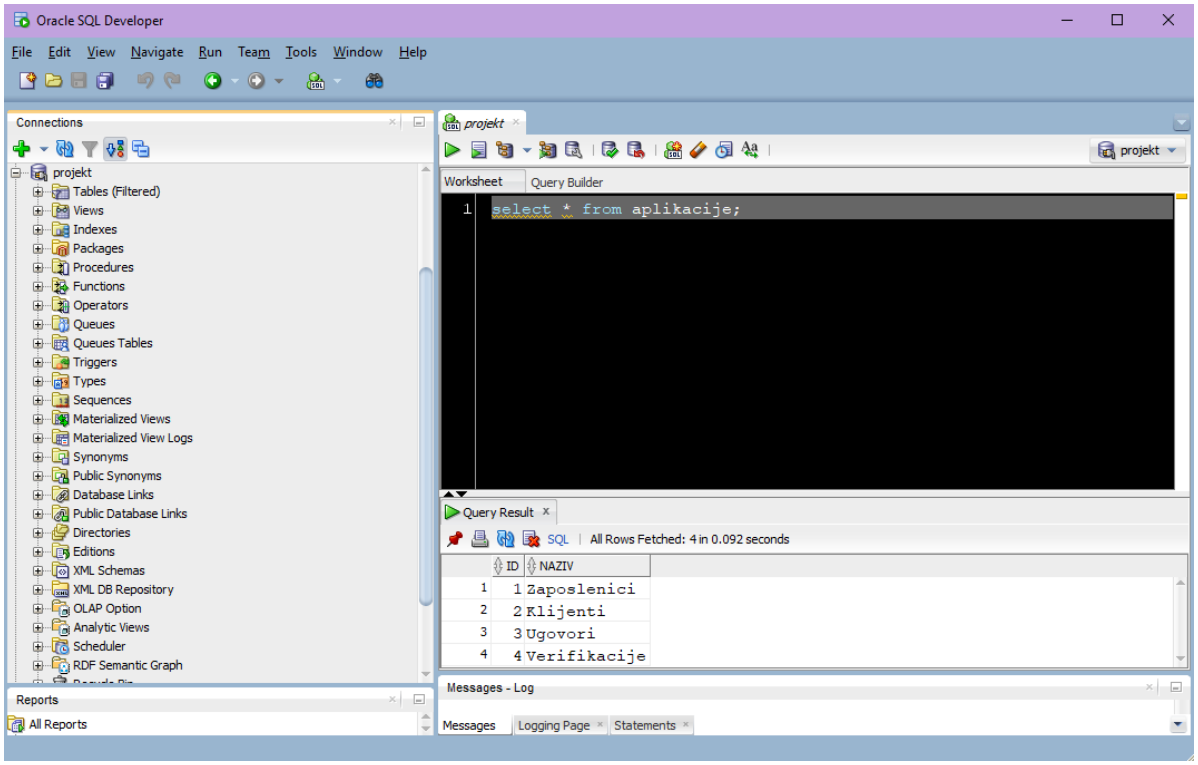
Jedan od alata za upravljanje bazom podataka naziva se SQL Developer. Izvršavanjem SQL naredbi omogućen je unos, pregledavanje, brisanje i ažuriranje podataka. Povezivanje na bazu podataka odvija se putem konekcije (engl. *connection*). Potrebni parametri za povezivanje su: ime konekcije, korisničko ime, zaporka, tip konekcije, lokacija baze, port i SID koji označava o kojoj se izvedbi baze radi kao što je prikazano slikom 5.1. [3]



Slika 5.1: Izgled prozora za povezivanje s bazom podataka

Ime konekcije je proizvoljan parametar koji olakšava povezivanje na željenu bazu. Korisničko ime i zaporka su parametri koji su postavljeni od strane sys korisnika. Tip konekcije je Basic pomoću naziva domaćina (engl. *hostname*), porta i SID-a baze.

Pomoću SQL developera moguće je otvoriti SQL Worksheet u kojem se izvršava PL/SQL program ili SQL upiti. Pregledi veza nalaze se na lijevoj strani grafičkog sučelja. Klikom na pohranjenu vezu ostvaruje se povezivanje s bazom podataka i otvara se novi prozor, spomenuti SQL Worksheet. Ovisno o pravima koja su dodijeljena korisničkom računu kojim se korisnik povezuje na bazu moguće je izvršavati određene SQL upite. Prilikom povezivanja s lijeve strane sučelja pojavljuje se prikaz svih tablica, pogleda (engl. *views*), indeksa, paketa, procedura, funkcija, popis okidača (engl. *trigger*), sekvenci i ostalih komponenti potrebnih za upravljanje bazom podataka kao što je prikazano slikom 5.2.



Slika 5.2: Izgled korisničkog sučelja SQL Developera

U donjem dijelu SQL developera nalazi se rezultat izvršenog koda. Osim toga, prikazani su logovi izvršavanja i poruke je li izvršavanje uspješno ili ne. Prikazane podatke moguće je izvesti (engl. *export*) u više formata.

SQL developer napisan je u programskom jeziku Java te je kompatibilan s Windows, Linux i Mac OS X operacijskim sustavima. Može se besplatno preuzeti sa službenih stranica tvrtke Oracle. [4]

5.3 PL/SQL

PL/SQL predstavlja naziv za proceduralni programski jezik za strukturirani jezik upita (engl. *Procedural Language extensions to the Structured Query Language*). Radi se o proširenju za SQL koje pruža više mogućnosti od samoga SQL jezika. Izvršava se unutar PL/SQL blokova na bazi podataka. Programski kod pohranjuje se na bazi u paketima (engl. *package*) u obliku funkcija ili procedura. Sve funkcije i procedure moguće je pohraniti unutar jednog ili više paketa.

Blokovi PL/SQL koda najčešće se sastoje od tri manja logička bloka: *declare*, *begin* i *exception*. *Declare* predstavlja ključnu riječ nakon koje se deklariraju potrebne varijable se koriste u begin izvršnom dijelu. *Begin* je dio koji služi za pisanje izvršnog koda dok potencijalnim iznimkama rukovodi u exception bloku. Kraj bloka označen je ključnom riječi *end* nakon koje kompajler programskog koda završava svoj rad. Primjer PL/SQL bloka prikazan je programskim kodom 5.1.

Programski kod 5.1: Primjer PL/SQL bloka

```
SET SERVEROUTPUT ON
DECLARE
    l_id  number(10);
    l_ime varchar2(40);
BEGIN
    select ime into l_ime from klijenti where id = l_id;
EXCEPTION
  WHEN no_data_found THEN
      dbms_output.put_line('Nema traženih podataka.');
  WHEN others THEN
      dbms_output.put_line('Nepoznata greška.');
END;
```

Kod upravljanja iznimkama mogu se koristiti ugrađene iznimke kao što su *no_data_found* za slučaj kada nema traženih podataka ili *zero_divide* kada se u izvršnom kodu pokuša dijeliti s nulom. Za slučajeve koji nisu obuhvaćeni ugrađenim iznimkama moguće je definirati vlastite.

Prilikom provjere ispravnosti bloka potrebno je iznad declare logičkog bloka pokrenuti *"set serveroutput on"*. Svrha navedene naredbe je omogućavanje ispisa funkcije *dbms_output.put_line()* u konzoli. Koristeći ispis u konzoli ispituje se ispravnost izlaznih podataka prije implementacije bloka unutar neke funkcije ili procedure.

PL/SQL ima ugrađene uvjete i petlje unutar kojih se izvršava SQL kod na temelju određenih uvjeta kao što je prikazano programskim kodom 5.2. Navedene funkcionalnosti slične su kao u drugim programskim jezicima.

Programski kod 5.2: Primjer izvršavanja koda na temelju uvjeta u PL/SQL-u

```
DECLARE
    l_id  number(10);
    l_broj number(10);
BEGIN

    select broj_zaposlenika
    into l_broj
    from tvrtke
    where id = l_id;

    IF (broj_zaposlenika > 10) THEN
        dbms_output.put_line('Više od 10 zaposlenika.');
    ELSE
        dbms_output.put_line('Manje od 10 zaposlenika.');
    END IF;
END;
```

Navedene funkcionalnosti PL/SQL-a samo su neke od mnogih koje ima. Svojom brzinom i mogućnostima koje pruža predstavlja izuzetno kvalitetan alat za rad s relacijskim bazama podataka, ali i za nerelacijske baze nakon što su u novim izdanjima dodane razne funkcionalnosti za manipulaciju podacima zapisanim u JSON ili XML formatima. [5]

5.4 Dinamički PL/SQL

Dinamički PL/SQL naziv je za programski kod koji se kreira prilikom izvršavanja programa. Mogućnosti za rad su raznovrsne. Dinamički SQL predstavlja naredbe kreirane prilikom izvođenja programa ovisno o ulaznim parametrima. Prema [7] prikazano je kako se u većini slučajeva za izvršavanje dinamičkog SQL ili PL/SQL programskog koda koristi *"EXECUTE IMMEDIATE"* izraz s mogućim dodatnim naredbama za pohranu, uključivanje dodatnih parametara u izraz i slično. Programski kod 5.3 prikazuje dohvaćanje podataka iz tablice čiji naziv je određen ulaznim parametrom.

Programski kod 5.3: Primjer dinamičkog SQL koda

```
l_naredba := 'select result from ' || l_tablica ||
             '_v WHERE JSON_VALUE(result,' ||
             '''$.id''' || ') = ' || l_id;

EXECUTE IMMEDIATE l_naredba into l_output;
```

Ovaj način upravljanja upitima omogućuje izradu jedne procedure koja će dohvaćati podatke iz više različitih tablica s različitim uvjetima, ovisno o ulaznom parametru koji je poslan s nekog servisa. Zbog podrške za objektno orijentirano programiranje navedeni primjer je samo jedan od mnogih načina za korištenje dinamičkog izvršavanja SQL i PL/SQL programskog koda.

Za razliku od standardnog statičnog SQL-a, programski kod dinamičkog SQL-a provjerava se tek prilikom izvođenja što može uzrokovati sintaktičke pogreške prilikom izvršavanja stoga je važno odrediti iznimke i rukovoditi njima u slučaju pogreške.

6. RAZVOJ APLIKACIJE

6.1 Korištene tehnologije

Prilikom izrade aplikacije kao temeljni dio koristi se Oracle baza podataka na kojoj se odvija cijeli proces autorizacije i pohrane podataka. PHP server korišten je za implementaciju autentifikacije koristeći ugrađene funkcije za manipuliranje sesijama i kao srednji sloj u komunikaciji između korisnika i pozadinskoj sloja. Za komunikaciju između sučelja i PHP servera koristi se AJAX tehnologija iz jQuery JavaScript biblioteke. Prezentacijski sloj izrađen je koristeći HTML, CSS, JavaScript i Bootstrap CSS biblioteke. Ispitivanje autorizacije i autentifikacije je realizirano koristeći Postman, alat za API (engl. *Application programming interface*) testiranje. API predstavlja aplikacijsko programsko sučelje odnosno skup pravila koja program mora zadovoljiti da bi se njegov rad smatrao pravilnim. Slika 6.1 prikazuje tijek komunikacije između baze podataka, PHP servisa i korisnika.

Slika 6.1: Prikaz slojeva komunikacije

6.2 Logika na bazi podataka

Temelj cjelokupne logike sadržan je u paketu koji sadrži glavnu *p_main* proceduru kojoj se prosljeđuje JSON objekt prilikom svakog poziva s PHP servera. JSON objekt ulazi u proceduru u obliku stringa te se onda pretvara u pravi JSON objekt kojim PL/SQL može manipulirati. Programski kod izvršava se unutar blokova koji omogućuju upravljanje iznimkama (engl. *exception handling*).

Programski kod 6.1: Deklaracija p_main procedure

```
procedure p_main(in_json in varchar2, out_json out varchar2);
```

Procedura p_main zadužena je za logičko preusmjeravanje na druge procedure ovisno o uvjetima koji se zadovolje ulaznim podacima. Ako se radi o pozivu *p_login* procedure, u tom

slučaju se provjerava postoji li u bazi korisnik s navedenim korisničkim imenom i zaporkom. Navedena procedura je zadužena za autentifikaciju, dok su ostale procedure namijenjene autorizaciji, verifikaciji i manipulaciji podacima. Svaki poziv procedure koji nije povezan s *p_login* procedurom podrazumijeva da je poziv napravio korisnik koji je autentificiran te se tom prilikom odvija autorizacija na temelju identifikatora korisnika zapisanog u PHP sesiji. Deklaracija *p_main* procedure i njezina implementacija prikazani su programskim kodovima 6.1 i 6.2.

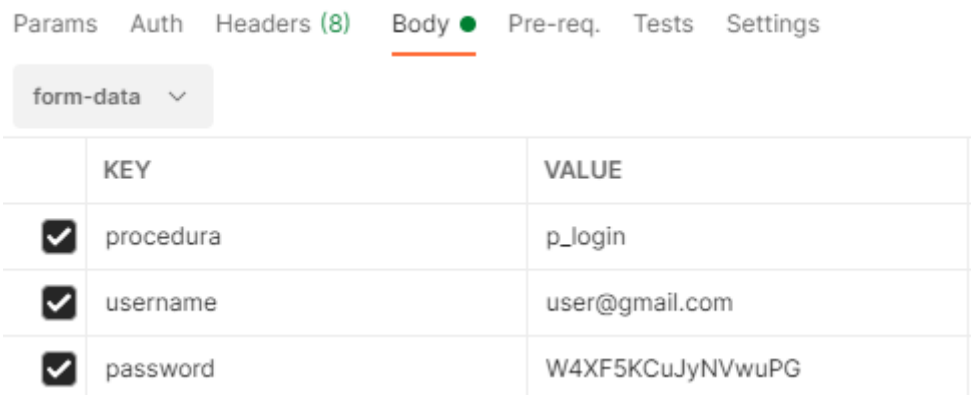
Programski kod 6.2: Implementacija p_main procedure

```
procedure p_main(in_json in varchar2, out_json out varchar2) AS
    l_obj         json_object_t;
    l_out         json_object_t;
    l_string      varchar2(4000);
    l_procedura   varchar2(200);
    l_grupa       varchar2(200);
    l_tablica     varchar2(200);
    l_id          number(10);
    l_UserID      number(10);
BEGIN
      l_obj := json_object_t(in_json);
SELECT
    JSON_VALUE(in_json, '$.procedura' RETURNING varchar2),
    JSON_VALUE(in_json, '$.UserID' RETURNING number),
    JSON_VALUE(in_json, '$.id' RETURNING number),
    JSON_VALUE(in_json, '$.grupa[0]' RETURNING varchar2),
    UPPER(JSON_VALUE(in_json, '$.tablica' RETURNING varchar2))
INTO l_procedura, l_UserID, l_id, l_grupa, l_tablica
FROM DUAL;
CASE l_procedura
    WHEN 'p_login' THEN
      p_login(l_obj, l_out);
    WHEN 'p_save' THEN
      IF(NVL(l_id, 0) = 0) THEN
        IF(f_authorise(l_UserID, l_grupa, l_tablica, 'insert'))
          THEN
            p_save(l_obj, l_obj);
        ELSE
          l_obj.put('h_message', 'Nemate potrebne ovlasti.');
          l_obj.put('h_errcode', 401);
        END IF;
      ELSE
        IF(f_authorise(l_UserID, l_grupa, l_tablica, 'update'))
          THEN
            p_save(json_object_t(in_json), l_obj);
        ELSE
            l_obj.put('h_message', 'Nemate potrebne ovlasti.');
            l_obj.put('h_errcode', 401);
        END IF;
      END IF;
```

Prije poziva procedura koje dohvaćaju, mijenjaju, dodaju ili brišu podatke, poziva se *f_authorise* funkcija koja provjerava ima li korisnik ovlasti za određenu operaciju. Navedena provjera ovlasti sprječava neovlašteni pristup resursima.

6.3 Razvoj autentifikacije

Za potrebe autentifikacije koristi se jednostavna autentifikacija korisničkim imenom i zaporkom. Prosljeđuju se parametri o proceduri, korisničkom imenu i zaporki kao što je prikazano slikom 6.2.

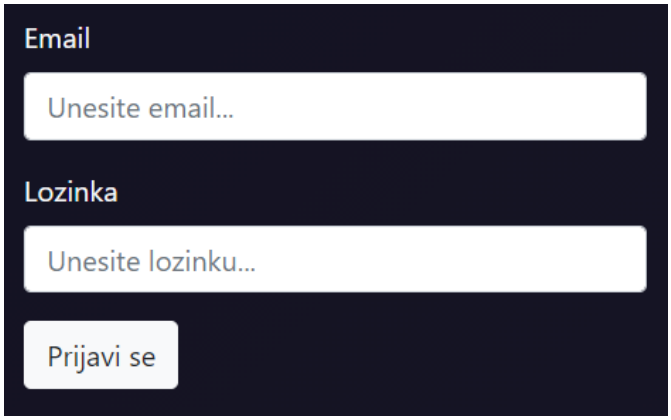


Slika 6.2: Potrebni parametri za prijavu korisnika

Navedeni podaci šalju se prema PHP serveru koji pozivom na bazu provjerava ispravnost pristupnih podataka. Ako su podaci ispravni, isti se zapisuju u obliku kolačića na klijentovom računalu i koriste se u budućim procesima autorizacije. Podaci o korisniku zapisuju se u globalnu varijablu *$_SESSION* koja predstavlja polje s informacijama o korisniku. Zbog potrebe za autorizacijom u polje se dodaje *$_SESSION["username"]* kako bi prilikom svakog novog zahtjeva od strane klijenta znali o kome se radi. Tako se izbjegava konstanta autentifikacija od strane korisnika.

Sesija se otvara pokretanjem funkcije *session_start()*. Navedena funkcija omogućava učitavanje varijabli u *$_SESSION* polje i potrebno ju je pozvati odmah na početku skripte. Zatvaranje sesije može se odviti automatski nakon nekog vremena ako klijent u određenom vremenskom intervalu nije uputio zahtjev prema poslužitelju ili pozivom funkcije *session_destroy()* prilikom zahtjeva korisnika za odjavu iz sustava.

Forma za autentifikaciju ima jednostavno sučelje s poljima za unos e-maila i zaporke dizajniranih koristeći Bootstrap. Oba polja imaju validacijske provjere u obliku regularnog izraza (engl. *regular expression*) implementiranog u JavaScriptu. Ako uneseni podaci ne zadovoljavaju uvjete validacije, prikazuje se skočni prozor s upozorenjem.



Slika 6.3: Izgled forme za prijavu

Klikom na gumb "*Prijavi se*" na sučelju prikazanim slikom 6.3 poziva se *login()* funkcija koja Ajax pozivom prikazanim programskim kodom 6.3 prosljeđuje podatke za prijavu PHP serveru koji poziva bazu i provjerava ispravnosti tih podataka. Ako su podaci ispravni, korisnik se preusmjerava na glavni ekran na kojem će moći vidjeti aplikacije za koje ima dozvoljen pristup.

Programski kod 6.3: Ajax poziv prema PHP serveru

```
$.ajax({
      type: 'POST',
      url: 'router.php',
      data: {
        procedura: 'p_login',
        username: $('#email').val(),
        password: $('#password').val(),
      });
```

Programski kod 6.4: Pohrana podataka u sesiju

```
$_SESSION['UserID'] = $data[0]['ID'];
$_SESSION['ime'] = $data[0]['ime'];
$_SESSION['prezime'] = $data[0]['prezime'];
$_SESSION['sessionID'] = session_id();
$_SESSION['aplikacije'] = $data[1];
$_SESSION['grupa'] = $data[2];
$data[0]['sessionID'] = session_id();
```

Ako procedura iz baze podataka odgovori s podacima o korisniku, tada se ti podaci zapisuju u sesiju kako se prilikom kretanja stranicom korisnik ne bi morao svaki puta iznova prijavljivati u sustav. Programski kod 6.4 predstavlja pohranu podataka u sesiju.

Provjera podataka o korisniku na bazi podataka odvija se u *p_login* proceduri. Prije pokretanja SQL upita potrebno je pohraniti korisničko ime i zaporku iz pristiglog JSON objekta u varijable na bazi podataka. Programski kod 6.5 zaslužan je za spomenutu radnju.

Programski kod 6.5: Čitanje podataka za prijavu na bazi podataka

```
SELECT
    JSON_VALUE(l_input, '$.username' RETURNING varchar2),
    JSON_VALUE(l_input, '$.password' RETURNING varchar2)
INTO
    l_username,
    l_password
FROM
    dual;
```

Uvjet za pokretanje upita je postojanje informacija u varijablama. Provjera unosa postoji na klijentskoj strani i na bazi podataka. Nakon što su navedeni uvjeti zadovoljeni, pokreće se upit prikazan programskim kodom 6.6 kojim se dohvaća ID pomoću kojeg se u idućem upitu dohvaćaju ostali potrebni podaci.

Programski kod 6.6:Provjera podataka i pokretanje upita

```
IF(l_username IS NULL or l_password IS NULL) THEN
  l_obj.put('h_message', 'Molimo unesite korisničko ime i
                          zaporku');
  l_obj.put('h_errcod', 101);
  RAISE e_iznimka;
ELSE
  BEGIN
    SELECT
      id
    INTO
      l_id
    FROM
      zaposlenici
    WHERE
      username = l_username AND
      password = l_password;
```

Na temelju ulaznog identifikatora dohvaćaju se podaci o korisniku koji su potrebni za kreiranje SESSION-a. Upotrebom JSON_OBJECT funkcije prikazane programskim kodom 6.7 SQL upit će kao rezultat vratiti podatke u JSON formatu.

Programski kod 6.7:Dohvaćanje podataka na temelju ID-a

```
SELECT
    JSON_OBJECT(
        'ID' VALUE zap.id,
        'ime' VALUE zap.ime,
        'prezime' VALUE zap.prezime,
        'username' VALUE zap.username)
    INTO l_record
    FROM zaposlenici zap
    WHERE
         id = l_id;
```

Izlazni podaci pohranjuju se u JSON objekt koji u sebi sadrži poruke o statusima upita, odnosno jesu li upiti uspješno izvršeni ili se pojavila greška na bazi podataka te podatke o korisniku i aplikacijama kojima ima dozvoljen pristup. Navedena pohrana obavlja se izvršavanjem programskog koda 6.8.

Programski kod 6.8:Pohrana podataka u JSON objekt

```
l_out.append(json_object_t(l_record));
l_out.append(json_array_t(f_applications(l_id)));
l_obj.put('data', l_out);
out_json := l_obj;
```

Kako bi korisnik nakon prijave mogao odabrati aplikaciju za rad, potreban je popis aplikacija kojima ima pristup. Popis aplikacija dohvaća se pozivom funkcije *f_applications* koja na temelju identifikatora vraća JSON polje s popisom aplikacija. Funkcija je prikazana programskim kodom 6.9.

Programski kod 6.9: Funkcija za dohvaćanje popisa aplikacija

```
SELECT
    '["' || listagg(aplikacije.naziv,'|| '"', "'' || ') || '"]'
INTO
    l_izlaz
FROM
    zaposlenici_app_ovlasti, grupe, grupa_aplikacija,
    aplikacije
where
    (grupe.id = zaposlenici_app_ovlasti.grp_id and
    zaposlenici_app_ovlasti.zap_id = id_korisnika) and
    grupa_aplikacija.id_grupe = grupe.id and
    grupa_aplikacija.id_aplikacije = aplikacije.id;
```

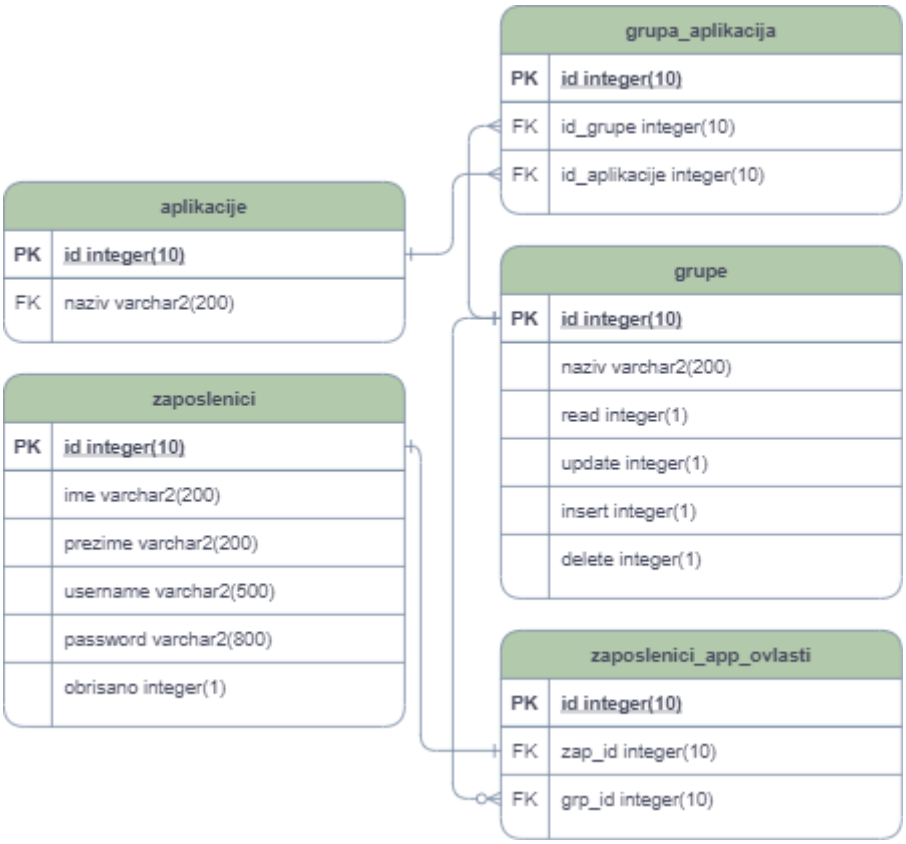
Svaki pokušaj prijave zapisuje se u log datoteku na serveru pozivom funkcije *writeLog* s ulaznim JSON podacima kao što prikazuje programski kod 6.10.

Programski kod 6.10: Funkcija za kreiranje log podataka

```
static function writeLog($injson, $outjson){
  $now = DateTime::createFromFormat('U.u', microtime(true));
  $date = $now->format("m-d-Y H:i:s.u");
  file_put_contents('/app/docroot/project/service/log.txt',
                    $date . ";" . " " . $injson . " " .
                    $outjson . ";", FILE_APPEND | LOCK_EX);
}
```

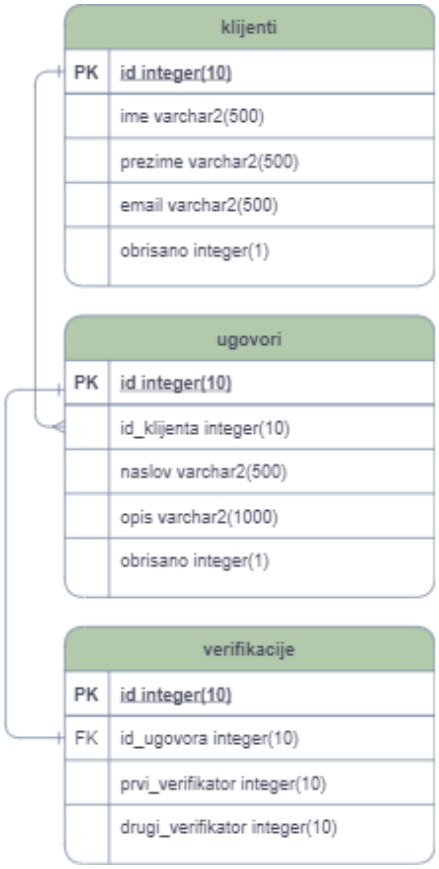
6.4 Ovlasti i model baze podataka

Za dodavanje grupnih ovlasti potrebno je najprije dodati zaposlenika i aplikacije u bazu podataka. Nakon dodavanja zaposlenika i aplikacija moguće je kreiranje grupa ovlasti. Svaka grupa može imati ovlasti nad jednom ili više aplikacija. Jedna grupa ima iste ovlasti na svim aplikacijama koje su dodijeljene toj grupi.



Slika 6.4: Temeljna struktura autorizacijskog sustava

Slika 6.4 prikazuje entitete koji su potrebni za rad autorizacijskog sustava. Tablica *zaposlenici* osim prilikom autorizacije, koristi se za autentifikaciju korisnika. Polje *obrisano* postavljeno je na zadanu vrijednost *0* što označuje podatak kao vidljiv korisnicima. Prilikom brisanja slogovi se ne uklanjaju iz baze podataka nego se postavljaju u status *1* čijom se provjerom prilikom dohvaćanja podataka ne uzimaju u obzir podaci s navedenim statusom. Slikom 6.5 prikaz je primjer poslovnog modela koji je implementiran u aplikaciji kako bi se na realnom primjeru pokazali autorizacijski i verifikacijski procesi.



Slika 6.5: Poslovni i verifikacijski modeli

Pravila koja onemogućuju neispravne unose u bazu definirana su posebnim ograničenjima (engl. *constraints*). Tablica verifikacije nad stupcem *id_ugovora* ima ograničenje jedinstvenosti (engl. *unique constraint*) što osigurava pohranu isključivo jednog ugovora za verifikaciju. Isto ograničenje koristi se u tablici *zaposlenici_app_ovlasti* kod ID oznake grupe kako bi jedan korisnik mogao biti u samo jednoj grupi. U tablicama gdje postoji stupac *obrisano* postoji ograničenje provjere (engl. *check constraint*) koji provjerava je li unos u rasponu između 0 i 1. Oznaka *0* predstavlja vidljivi podatak dok oznaka *1* predstavlja izbrisani podatak.

6.5 Razvoj autorizacije

Autorizacija se odvija na Oracle bazi podataka. Također, na istoj bazi su pohranjeni podaci o zaposlenicima, klijentima i svim aplikacijama. Autorizacijska logika napisana je u paketima u obliku procedura koje se izvršavaju prilikom svakog poziva prema bazi. Ulazni podaci na bazi podataka su u JSON obliku, a informacije koje PHP servis šalje prema bazi ovise o aplikaciji i akciji (*read*, *write*, *update*, *insert*) koju korisnik napravi na sučelju aplikacije.

Programskim kodom 6.11 prikazana je funkcija koja na temelju ulaznih parametara provjerava ovlasti.

Programski kod 6.11: Funkcija za provjeru ovlasti

```
IF(l_grupa != f_group(id_korisnika)) THEN
        RETURN false;
END IF;

CASE l_ovlast
    WHEN 'read' THEN
      select "read" into l_check from grupe where naziv =
l_grupa;
    WHEN 'insert' THEN
        select "insert" into l_check from grupe where naziv =
l_grupa;
    WHEN 'update' THEN
        select "update" into l_check from grupe where naziv =
l_grupa;
    WHEN 'delete' THEN
        select "delete" into l_check from grupe where naziv =
l_grupa;
END CASE;

l_array := json_array_t(f_applications(id_korisnika));

FOR indx IN 0 .. l_array.get_size -1
  LOOP
    IF(UPPER(l_array.get_string(indx)) = l_tablica) THEN
      IF(nvl(l_check, 0) = 1) THEN
        return true;
      END IF;
    END IF;
   END LOOP;

RETURN false;
```

6.6 Razvoj verifikacije

Verifikacija u ovom sustavu implementirana je tako da obaviti verifikaciju može samo zaposlenik koji se nalazi u grupi *"Verifikator"*. Da bi dokument bio verificiran potrebne su dvije verifikacije od dva različita verifikatora. Kod prikaza ugovora verifikatoru su dostupni samo oni ugovori koji u tom trenutku nisu verificirani. U bazi podataka zadana vrijednost polja prve i druge verifikacije je *null* što označava da taj dokument nije verificiran. Prilikom verifikacije to polje se popunjava s ID oznakom koja je verifikatoru dodijeljena prilikom kreiranja korisničkog računa. Procedura za obavljanje verifikacije implementirana je pod nazivom *p_verificiraj* i prima kao ulazne parametre ID oznaku ugovora i ID oznaku korisnika koji želi obaviti verifikaciju. Procedura je prikazana programskim kodom 6.12.

Programski kod 6.12: Verifikacija ugovora

```
IF(l_prvi_verifikator = 1) THEN
  UPDATE verifikacije SET prvi_verifikator = l_UserID
  WHERE id = l_id;
  l_obj.put('h_message', 'Uspješno ste verificirali ugovor!');
  l_obj.put('h_errcode', 0);
ELSIF(l_drugi_verifikator = 1) THEN
  select prvi_verifikator into l_prvi_verifikator from
  verifikacije where id = l_id;
    IF(l_prvi_verifikator = l_UserID) THEN
      l_obj.put('h_message', 'Već ste verificirali ovaj
      ugovor!');
      l_obj.put('h_errcode', 0);
    ELSE
      update verifikacije set drugi_verifikator = l_UserID
      where id = l_id;
      l_obj.put('h_message', 'Uspješno ste verificirali
                ugovor!');
        l_obj.put('h_errcode', 0);
    END IF;
ELSE
  l_obj.put('h_message', 'Ovaj ugovor je verificiran!');
  l_obj.put('h_errcode', 0);
END IF;
```

6.7 Rad s podacima

Rad s podacima predstavlja dohvaćanje, zapisivanje, izmjenu i brisanje podataka. Za potrebe dohvaćanja podataka implementirana je procedura *p_get_data*. Navedena procedura na temelju podataka o tablici i identifikatora dohvaća potrebne podatke iz pogleda. Za svaku novu tablicu u bazi je potrebno kreirati pogled kako bi na izlazu dobili podatke u JSON formatu i izbjegli

naknadne izmjene nad PL/SQL paketima u slučaju dodavanja novih tablica. Programski kod 6.13 predstavlja pogled za tablicu klijenti. Na isti način kreirani su pogledi za sve ostale tablice u bazi podataka.

Programski kod 6.13: Pogled za tablicu klijenti

```
SELECT
    JSON_OBJECT ('id' value id,
                 'ime' value ime,
                 'prezime' value prezime,
                 'email' value email) as result
FROM
    klijenti
WHERE
    obrisano = 0
```

Upit za dohvaćanje podataka generira se dinamički te se pokreće pokazivač (engl. *cursor*) za kreiranu naredbu. Pokazivač prolazi kroz sve slogove te svaki pojedinačni zapis dodaje u JSON polje koje predstavlja izlazne podatke iz baze namijenjene prikazivanju korisniku. Prilikom dohvaćanja podataka važno je provjeriti ulazni podatak o ID oznaci te vratiti slog na temelju oznake ili vratiti sve slogove koji se nalaze u traženoj tablici. Programski kod 6.14 zadužen je za dohvaćanje podataka.

Programski kod 6.14: Upit za dohvaćanje podataka i implementacija pokazivača

```
IF NVL((l_id, 0) > 0) THEN
  l_naredba := 'select result from ' || l_tablica || '_v WHERE
JSON_VALUE(result,' || '''$.id''' || ') = ' || l_id;
ELSE
  l_naredba := 'select result from ' || l_tablica || '_v';
END IF;
OPEN c_zapisi FOR l_naredba;
     LOOP
        FETCH c_zapisi INTO l_izlaz;
        EXIT WHEN c_zapisi%notfound;
        l_array.append(JSON_OBJECT_T(l_izlaz));
     END LOOP;
CLOSE c_zapisi;
```

Procesi zapisivanja i izmjene podataka odvijaju se u procedurama *p_save* i *p_ins_upd*. Korisnik klikom na korisničkom sučelju prilikom unosa novih podataka ili izmjene poziva proceduru *p_save*. Unutar procedure p_save poziva se procedura za kreiranje dinamičkog SQL-a, procedura *p_ins_upd*. Za kreiranje dinamičkog SQL-a potrebni su podaci koji će biti zapisani u

bazi podataka. Ako je korisnik zatražio spremanje izmjene podataka, tada se prilikom slanja podataka na bazu šalje i ID oznaka zapisa kojega je potrebno izmijeniti. U slučaju kada ID oznaka nije poslana podrazumijeva se da je korisnik poslao zahtjev za kreiranje novog sloga u tablici. Navedeni pristup rješavanju problema omogućuje da se za izmjenu i pohranu podataka koristi jedna procedura. Na temelju naziva tablice procedura pomoću sistemske tablice *all_tab_columns* dohvaća sve atribute kako bi se dinamički mogla generirati SQL naredba za unos ili izmjenu podataka. Programski kod 6.15 prikazuje uvjet kojim se određuje hoće li se izvršiti dinamički SQL kod za pohranu ili izmjenu podataka. Ugrađena funkcija *NVL* će u slučaju da ID oznaka nije proslijeđena vratiti vrijednost 0, dok će u drugom slučaju vratiti proslijeđenu vrijednost. Tako će se izvršavati potrebne naredbe.

Programski kod 6.15: Pohranjivanje i izmjena podataka

```
IF NVL((l_ID, 0) > 0) THEN
    EXECUTE IMMEDIATE l_update;
ELSE
    EXECUTE IMMEDIATE l_insert;
END IF;
```

Posljednja mogućnost kod rada s podacima, brisanje podataka, implementirana je u proceduri *p_delete*. Procedura zahtjeva ID oznaku zapisa koji je potrebno ukloniti i naziv tablice u kojoj se nalazi. Zbog sigurnosti podataka u velikom broju sustava se slogovi ne uklanjanju iz baze nego se atribut poput *obrisano* postavlja u vrijednost *1* za koju je prilikom planiranja sustava određeno da označava uklonjeni slog. Slogovi sa spomenutom vrijednošću ne uzimaju se u obzir prilikom rada s podacima te ih je u potrebno pozadinski obraditi ako se dogodila neželjena akcija nad podacima.

Programski kod 6.16: Brisanje podataka

```
l_naredba := 'SELECT count(*) FROM ALL_TAB_COLUMNS WHERE
              TABLE_NAME = upper(''' || l_tablica || ''') AND
              COLUMN_NAME = ''OBRISANO''';
EXECUTE IMMEDIATE l_naredba INTO l_check;

IF (nvl(l_id, 0) > 0 AND l_check > 0) THEN
  l_naredba := 'update ' || l_tablica || ' set obrisano = 1
               where id = ' || l_id;
ELSIF nvl(l_id, 0) > 0 then
  l_naredba := 'delete from ' || l_tablica || ' where id = ' ||
               l_id;
END IF;
```

Koristeći sistemsku tablicu *all_tab_columns* moguće je provjeriti postoji li atribut *obrisano* unutar tablice nad kojom se želi pokrenuti naredba brisanja. Tako će se nad tablicama gdje ne postoji spomenuti atribut pokrenuti naredba za brisanje te se navedeni slog više neće nalaziti u tablici. U slučaju da atribut *obrisano* postoji, njegova vrijednost će se postaviti u *1* i tako više neće biti vidljiv korisnicima, ali će se i dalje nalaziti zapisan u tablici. Brisanje podataka prikazano je programskim kodom 6.16.

7. INICIJALNE POSTAVKE SUSTAVA

Prije kretanja sustava s radom potrebno je podesiti određene segmente kako bi sustav autorizacije imao podatke na temelju kojih može određivati ima li korisnik pristup traženom resursu. Prvo što je potrebno je kreirati grupe ovisno o potrebama poslovanja. Za potrebe testiranja sustava kreirane su četiri grupe prikazane programskim kodom 7.1.

Programski kod 7.1: Kreiranje grupa

```
insert into grupe (naziv, "read", "update", "insert", "delete")
values ('Administrator', 1, 1, 1, 1);

insert into grupe (naziv, "read", "update", "insert", "delete")
values ('Tajnik', 1, 1, 1, 1);

insert into grupe (naziv, "read", "update", "insert", "delete")
values ('Verifikator', 1, 1, 0, 0);

insert into grupe (naziv, "read", "update", "insert", "delete")
values ('Analitičar', 1, 0, 0, 0);
```

Nakon kreiranja grupa potrebno je kreirati aplikacije. Svaka aplikacija predstavlja tablicu u bazi podataka. Zbog veza između tablica potrebno je pojedine podatke prikazati korisniku iako nema pristup jednoj od tablica te se zbog toga koriste pogledi. Pogled u bazi podataka je kreiran SQL upitom i ponaša se kao tablica. Programski kod 7.2 prikazuje kreiranje aplikacija dodavanjem u tablicu aplikacije.

Programski kod 7.2: Kreiranje aplikacija

```
insert into aplikacije (naziv) values ('Zaposlenici');
insert into aplikacije (naziv) values ('Klijenti');
insert into aplikacije (naziv) values ('Ugovori');
insert into aplikacije (naziv) values ('Verifikacije');
```

Aplikacije i grupe potrebno je logički povezati. Svaka od SQL naredbi prikazanih programskim kodom 7.3 predstavlja zapisivanje sloga u tablicu *grupa_aplikacija* kojim se definira veza između grupe i aplikacije. Za svaku aplikaciju potrebno je dodati novi slog u tablicu *grupa_aplikacija*. Svakoj grupi moguće je dodijeliti sve aplikacije koje se nalaze u tablici *aplikacije*.

Programski kod 7.3: Pridruživanje aplikacija grupama

```
insert into grupa_aplikacija (id_grupe, id_aplikacije) values
(1,1); -- Administrator može pristupiti Zaposlenicima
insert into grupa_aplikacija (id_grupe, id_aplikacije) values
(1,2); -- Administrator može pristupiti Klijentima
insert into grupa_aplikacija (id_grupe, id_aplikacije) values
(1,3); -- Administrator može pristupiti Ugovorima
insert into grupa_aplikacija (id_grupe, id_aplikacije) values
(2,2); -- Tajnik može pristupiti Klijentima
insert into grupa_aplikacija (id_grupe, id_aplikacije) values
(2,3); -- Tajnik može pristupiti Ugovorima
insert into grupa_aplikacija (id_grupe, id_aplikacije) values
(3,4); -- Verifikator može pristupiti Verifikacijama
insert into grupa_aplikacija (id_grupe, id_aplikacije) values
(4,1); -- Analitičar može pristupiti Zaposlenicima
insert into grupa_aplikacija (id_grupe, id_aplikacije) values
(4,2); -- Analitičar može pristupiti Klijentima
insert into grupa_aplikacija (id_grupe, id_aplikacije) values
(4,3);
```

Zadnji korak je pridruživanje zaposlenika grupama. Zapisivanjem sloga u tablicu *zaposlenici_app_ovlasti* korisnik sustava se dodaje u grupu te nakon toga ima sve ovlasti grupe u koju je dodan. Programski kod 7.4 prikazuje dodavanje zaposlenika u grupe koje su prethodno definirane. Zbog postavki autorizacijskog sustava jedan zaposlenik može biti član samo jedne grupe. Ovisno o poslovnom modelu i njegovim potrebama moguće je prilagoditi sustav tako da se dodjela ovlasti odvija na neki drugi način.

Programski kod 7.4: Pridruživanje aplikacija grupama

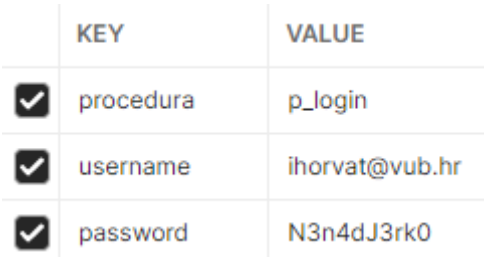
```
insert into zaposlenici_app_ovlasti (zap_id, grp_id) values
(1,1); -- 1. zaposlenik postaje administrator
insert into zaposlenici_app_ovlasti (zap_id, grp_id) values
(2,2); -- 2. zaposlenik postaje tajnik
insert into zaposlenici_app_ovlasti (zap_id, grp_id) values
(3,3); -- 3. zaposlenik postaje verifikator
insert into zaposlenici_app_ovlasti (zap_id, grp_id) values
(4,3); -- 4. zaposlenik postaje verifikator
insert into zaposlenici_app_ovlasti (zap_id, grp_id) values
(5,4); -- 5. zaposlenik postaje analitičar
```

8. TESTIRANJE APLIKACIJE

Testiranje aplikacije napravljeno je u Postman alatu. Prema [7] Postman alat se koristi za automatsko i istraživačko testiranje. Ispitane su funkcionalnosti autentifikacije, autorizacije i verifikacije. Prikazani su potrebni parametri i odgovor s baze podataka nakon slanja parametara prema bazi.

8.1 Testiranje autentifikacije

Prilikom autentifikacije poziva se PHP servis s podacima o proceduri, korisničkim imenom i zaporkom. Korisničko ime i zaporka šalju se prema bazi podataka isključivo ako je naziv procedure *p_login*. Slika 8.1 prikazuje potrebne podatke za prijavu u sustav.



Slika 8.1: Parametri potrebni za prijavu

Ako se podaci ne podudaraju s podacima u bazi podataka, odgovor baze je tekstualna poruka i šifra greške. Primjena šifre je pojednostavljivanje prikaza poruke korisniku. U ovom slučaju će se prilikom prijave prikazati poruka u obliku skočnog prozora isključivo ako je šifra 102 koja označava neuspješnu prijavu. Neuspješna prijava prikazana je programskim kodom 8.1.

Programski kod 8.1: Odgovor baze podataka na nepostojeće podudaranje podataka za prijavu

```
{
    "h_message": "Nepoznato korisničko ime ili zaporka.",
    "h_errcode": 102
}
```

Programski kod 8.2 odgovor je na uspješnu prijavu. Prilikom uspješne prijave s baze podataka vraćaju se podaci o prijavljenom korisniku, grupama u kojima se nalazi i aplikacijama za koje ima dozvoljen pristup. Navedeni podaci pohranjuju se u PHP sesiju i koriste se u daljnjem radu sustava.

Programski kod 8.2: Odgovor baze podataka na ispravne podatke za prijavu

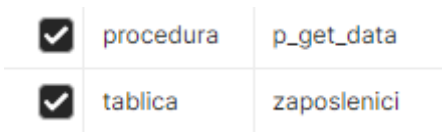
```
{
    "h_message": "",
    "h_errcode": "",
    "data": [
        {
            "ID": 1,
            "ime": "Ivan",
            "prezime": "Horvat",
            "username": "ihorvat2021@vub.hr"
        },
        ["Zaposlenici", "Klijenti", "Ugovori"],
        ["Administrator"]
    ]
}
```


8.2 Testiranje autorizacije

Prilikom testiranja autorizacije postoji više slučajeva koje je potrebno ispitati da bi se uvjerili u pravilan rad sustava. Zbog toga testovi obuhvaćaju dohvaćanje, izmjenu, pohranu i brisanje podataka koristeći se korisničkim računima koji se nalaze u različitim grupama ovlasti.

8.2.1 Dohvaćanje podataka

Za dohvaćanje podataka potrebno je pozvati proceduru *p_get_data* i proslijediti naziv tablice iz koje je potrebno dohvatiti podatke. Slika 8.2 predstavlja parametre koji su potrebni za dohvaćanje svih podataka iz tablice *zaposlenici*.



Slika 8.2: Parametri potrebni za dohvaćanje podataka iz tablice zaposlenici

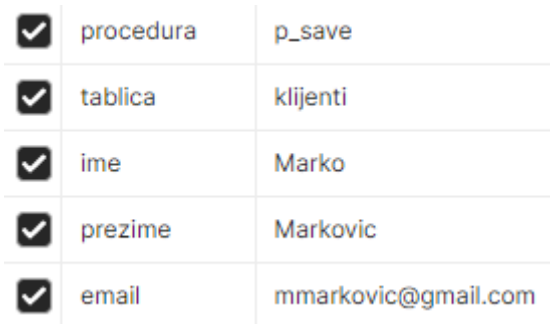
Prilikom slanja zahtjeva, na temelju ID oznake zapisane u sesiji i pristiglog naziva tablice, poziva se funkcija za provjeru ovlasti. Ako korisnik nema pristup tablici, funkcija će kao odgovor vratiti *false* nakon čega će baza podataka vratiti odgovor koji je prikazan programskim kodom 8.3 u kojem se nalaze dva parametra. Prvi je poruka koja će se prikazati na korisnikovom ekranu, a druga šifra greške prilikom koje se prikazuje spomenuta poruka.

Programski kod 8.3: Odgovor baze podataka na pokušaj neovlaštenog pristupa

```
{
    "h_message":
        "Nemate ovlasti za pristup aplikaciji ZAPOSLENICI.",
    "h_errcode": 401
}
```

8.2.2 Izmjena i pohrana podataka

Izmjena i pohrana podataka u autorizacijskom sustavu funkcionira na sličnom principu kao i dohvaćanje podataka. Autorizacijske provjere obavljaju se na identične načine provjerom posjeduje li korisnik ovlasti za pristup i potrebnu ovlast nad tablicom.



Slika 8.3: Parametri potrebni za pohranu podataka

Slika 8.3 prikazuje parametre potrebne za pohranu podataka u tablicu *klijenti*. Prilikom pohrane podataka potrebno je pozvati proceduru *p_save* i proslijediti joj sve parametre na temelju atributa koje tablica posjeduje. U slučaju izostavljanja nekog od parametara koji je potreban, naredba za pohranu se neće izvršiti. Programskim kodom 8.4 prikazan je odgovor baze nakon uspješne pohrane podataka.

Programski kod 8.4: Odgovor baze podataka na uspješnu pohranu podataka

```
{
    "h_message": "Uspješan unos.",
    "h_errcode": 0
}
```

Prilikom izmjene podataka potrebni su parametri kao za pohranu podataka, uz ID oznaku zapisa kojeg je potrebno izmijeniti.

8.2.3 Brisanje podataka

Za brisanje podataka potrebno je proslijediti ID zapisa, naziv tablice i naziv procedure *p_delete* kao što je prikazano slikom 8.4.

Slika 8.4: Parametri potrebni za brisanje podataka

Ako korisnik ima prava za brisanje nad tablicom koju je proslijedio, odgovor baze podataka je programski kod 8.5. Kod tablica koje posjeduju atribut *obrisano* vrijednost će se postaviti u *1*, ako nema onda će se zapis u potpunosti izbrisati iz tablice.

Programski kod 8.5: Odgovor baze podataka na uspješno brisanje podataka

```
{
    "h_message": "Uspješno brisanje!",
    "h_errcode": 0
}
```

8.3 Testiranje verifikacije

Za potrebe ispitivanja procesa verifikacije koriste se dva korisnička računa koja se nalaze u grupi *verifikatori*. Navedena grupa omogućava korisnicima pristup verifikacijskom dijelu sustava i verificiranje dokumenata. Programski kod 8.6 prikazuje odgovor na uspješnu verifikaciju od strane verifikatora. Uspješna verifikacija podrazumijeva da je korisnik napravio jednu od dvije verifikacije nad dokumentom.

Programski kod 8.6: Odgovor baze podataka na uspješnu verifikaciju

```
{
    "h_message": "Uspješno ste verificirali ugovor!",
    "h_errcode": 0
}
```

Da bi se dokument smatrao verificiranim potrebne su dvije verifikacije od strane dva različita verifikatora. Zbog toga, u slučaju da jedan verifikator pokuša dva puta verificirati dokument kao odgovor će dobiti zapis koji prikazuje programski kod 8.7.

Programski kod 8.7: Odgovor baze podataka na pokušaj dvostruke verifikacije od strane istog korisnika

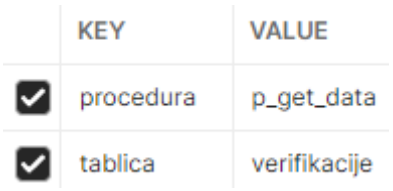
```
{
    "h_message": "Već ste verificirali ovaj ugovor!",
    "h_errcode": 0
}
```

Nakon što je ugovor u potpunosti verificiran, on se ne prikazuje verifikatorima na sučelju, ali postoji mogućnost da je prikazan ako je više verifikatora u istom trenutku posjetilo listu s ugovorima. Ako je neki ugovor s liste u međuvremenu verificiran, a korisnik pokuša verificirati isti, baza će vratiti odgovor prikazan programskim kodom 8.8.

Programski kod 8.8: Odgovor baze podataka na pokušaj verificiranja već verificiranog ugovora

```
{
    "h_message": "Ovaj ugovor je verificiran!",
    "h_errcode": 0
}
```

Lista s informacijama o ugovorima i verifikacijama dohvaća se pozivom procedure *p_get_data* i prosljeđivanjem parametra o nazivu tablice. Slika 8.5 predstavlja potrebne parametre za dohvaćanje podataka.



Slika 8.5: Parametri potrebni za dohvaćanje podataka o ugovorima i verifikacijama

Programski kod 8.9: Dohvaćanje ugovora i podataka o verifikaciji

```
{
    "h_message": "",
    "h_errcode": "",
    "data": [
        {
            "id": 81,
            "id_klijenta": 1,
            "klijent": "johndoe@gmail.com",
            "naslov": "Ugovor o pretplati",
            "opis": "Informacije o ugovoru.",
            "prvi_verifikator": "verifikator@gmail.com",
            "drugi_verifikator": "-"
        }
    ]
}
```

Programskim kodom 8.9 prikazan je odgovor baze na poziv za dohvaćanje dokumenata koje je potrebno verificirati. Sustav je postavljen tako da verifikator ne može mijenjati podatke dokumenta, nego isključivo obaviti verifikaciju dokumenta.

9. ZAKLJUČAK

Autorizacijski i verifikacijski sustavi neizbježni su u današnjem svijetu koji za cilj ima pojednostavljivanje i ubrzavanje raznih procesa. Kako bi bilo moguće izložiti resurse većem broju korisnika potreban je autorizacijski sustav koji će na temelju ovlasti dozvoliti ili odbiti pristup resursima. Izrađeni sustav prilagođen je jednostavnom poslovnom modelu koji bez dodatnih izmjena nije prilagodiv na druge poslovne modele. Autorizacijski i verifikacijski sustavi zahtijevaju prilagodbu poslovnom modelu kako bi mogli pravilno obavljati funkciju. U pojedinim slučajevima će grupne ovlasti biti zadovoljavajuće, dok će u zahtjevnijim sustavima biti potrebne i pojedinačne ovlasti. Grupne i pojedinačne ovlasti se u tom slučaju isprepliću i tako čine složeniji sustav.

Temelj razvijanja je Oracle baza podataka čije okruženje pruža sve mogućnosti za izradu autorizacijskog i verifikacijskog sustava. Komunikacija je ostvarena PHP serverskom tehnologijom, a korisničko sučelje koristeći JavaScript i jQuery. Za razvoj je potrebno osnovno poznavanje navedenih programskih alata i struktura baze podataka. Procedure za dodavanje, dohvaćanje, izmjenu i brisanje podataka izrađene su koristeći dinamički SQL što može olakšati prilagodbe sustava u budućnosti.

Za poboljšanje aplikacije potrebno je raditi na sigurnosnom dijelu i rukovanju iznimkama zbog dinamičkog SQL-a koji zahtjeva veću pažnju od standardnog SQL-a. Također, postoji prostor za napredak kod bilježenja aktivnosti korisnika zbog lakšeg uklanjanja potencijalnih problema u radu.

10. LITERATURA

[1] Securing Session INI Settings [Online]. Dostupno na: https://www.php.net/manual/en/session.security.ini.php. (10.9.2021.)

[2] Oracle Corporation [Online]. Dostupno na: https://www.oracle.com/corporate/. (28.9.2021.)

[3] Connect SQL Developer [Online]. Dostupno na: https://docs.oracle.com/en/cloud/paas/exadata-express-cloud/csdbp/connect-sql-developer.html. (11.9.2021.)

[4] Oracle SQL Developer [Online]. Dostupno na: https://www.oracle.com/database/technologies/appdev/sqldeveloper-landing.html (1.10.2021.)

[5] JSON in Oracle Database [Online]. Dostupno na: https://docs.oracle.com/en/database/oracle/oracle-database/21/adjsn/json-in-oracle-database.htm.(15.9.2021.)

[6] Performing SQL Operations with Native Dynamic SQL [Online]. Dostupno na: https://docs.oracle.com/cd/B19306_01/appdev.102/b14261/dynamic.htm. (17.9.2021.)

[7] Postman for Testers [Online]. Dostupno na: https://www.postman.com/use-cases/api-testing. (22.9.2021.)

11. OZNAKE I KRATICE

AJAX - Asynchronous JavaScript and XML (JSON)

API - Application programming interface

CSS - Cascading Style Sheets

HTML - HyperText Markup Language

ID - Identity column

JSON - JavaScript Object Notation

PL/SQL - Procedural Language for SQL

PHP - Hypertext Preprocessor

SID - Site Identifier

12. SAŽETAK

Autorizacija i verifikacija:

Cilj ovog rada je izrada sustava autorizacije i verifikacije koristeći Oracle bazu podataka. Navedeni sustav omogućava kontrolu pristupa i verificiranje dokumenata. Proces autorizacije i verifikacije prilagođen je jednostavnom poslovnom modelu kako bi bilo moguće prikazati sve funkcionalnosti sustava. Za komunikaciju između baze podataka i korisnika koristi se PHP tehnologija u kombinaciji s JavaScriptom i jQueryjem. Rad sadržava opis procesa autorizacije i verifikacije i tehnologija korištenih za izradu sustava. Struktura baze podataka predstavljena je dijagramima, a realizacija sustava programskim kodom. Za interakciju sa sustavom izrađeno je korisničko sučelje u svrhu prikaza primjene realiziranog sustava.

Ključne riječi: Oracle baza podataka, autorizacija, verifikacija

13. ABSTRACT

Authorisation and verification:

The goal of this paper is to develop an authorization and verification system using the Oracle database. This system implements access control and document verification. The authorization and verification process is adapted to a simple business model so that all system functionalities can be displayed. PHP technology is used in combination with JavaScript and jQuery to communicate between the database and users. The paper contains a description of the authorization and verification processes and technologies used to create the system. The structure of the database is represented by diagrams and the realization of the system by code. To interact with the system, a user interface was created in order to show the application of the implemented system.

Keywords: Oracle database, authorisation, verification

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 4 listopada 2021	Ilija Leko	Ilija Leko

Prema Odluci Veleučilišta u Bjelovaru, a u skladu sa Zakonom o znanstvenoj djelatnosti i visokom obrazovanju, elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru bit će pohranjene i javno dostupne u internetskoj bazi Nacionalne i sveučilišne knjižnice u Zagrebu. Ukoliko ste suglasni da tekst Vašeg završnog rada u cijelosti bude javno objavljen, molimo Vas da to potvrdite potpisom.

Suglasnost za objavljivanje elektroničke inačice završnog rada u javno dostupnom nacionalnom repozitoriju

Ilija Leko

ime i prezime studenta/ice

Dajem suglasnost da se radi promicanja otvorenog i slobodnog pristupa znanju i informacijama cjeloviti tekst mojeg završnog rada pohrani u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu i time učini javno dostupnim.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 4 listopada 2021.

Ilija Leko

potpis studenta/ice