



Univerzitet u Sarajevu  
Elektrotehnički fakultet u Sarajevu  
Odsjek za računarstvo i informatiku



# Database handler & meta data visualizer

## Baze podataka

*Mentor:*  
Doc. dr. Emir Buza

*Studenti:*  
Bećirović Šeila  
Džirlo Timur  
Pejčinović Dario

Sarajevo, 08.01.2018.

# Sadržaj

<b>1</b>	<b>Opis teme</b>	<b>2</b>
1.1	Funkcionalnosti sistema DHMV . . . . .	2
<b>2</b>	<b>Detaljni opis funkcionalnosti</b>	<b>4</b>
2.1	Pregled svih baza podataka . . . . .	4
2.2	Pregled svih tabela baze podataka . . . . .	4
2.3	Upravljanje objektima vezanih za bazu i/ili tabelu . . . . .	5
2.4	Pregled svih korisnika baze podataka . . . . .	5
2.5	Vizualizacija meta podataka . . . . .	6
2.6	Prikaz, kreiranje, uređivanje i brisanje podataka unutar baze . . . . .	6
<b>3</b>	<b>Tehnologija</b>	<b>7</b>
3.1	Java Spring . . . . .	7
3.1.1	Spring MVC . . . . .	8
3.1.2	Spring Data JPA . . . . .	8
3.2	AngularJS . . . . .	8
3.3	PostgreSQL . . . . .	9
3.4	OracleDB . . . . .	10
3.5	Android . . . . .	10
<b>4</b>	<b>Web aplikacija</b>	<b>11</b>
<b>5</b>	<b>Android aplikacija</b>	<b>21</b>
<b>6</b>	<b>Optimizacija upita i transakcije</b>	<b>28</b>
6.1	Login transakcija . . . . .	28
6.2	Funkcija getRowCount() . . . . .	29
6.3	Funkcija getNames() . . . . .	29
6.4	Funkcija getColumnNames() . . . . .	29
6.5	Funkcija objectsMetaData() . . . . .	29
6.6	Funkcija createView() . . . . .	29
6.7	Funkcija createTrigger() . . . . .	30
6.8	Funkcija primaryKeys() . . . . .	31
6.9	Funkcija createIndex() . . . . .	31
6.10	Funkcija showERD() . . . . .	32
6.11	Funkcija createAutoIncrement_Trigger() . . . . .	33
6.12	Funkcija createTable() . . . . .	33

# Poglavlje 1.

## Opis teme

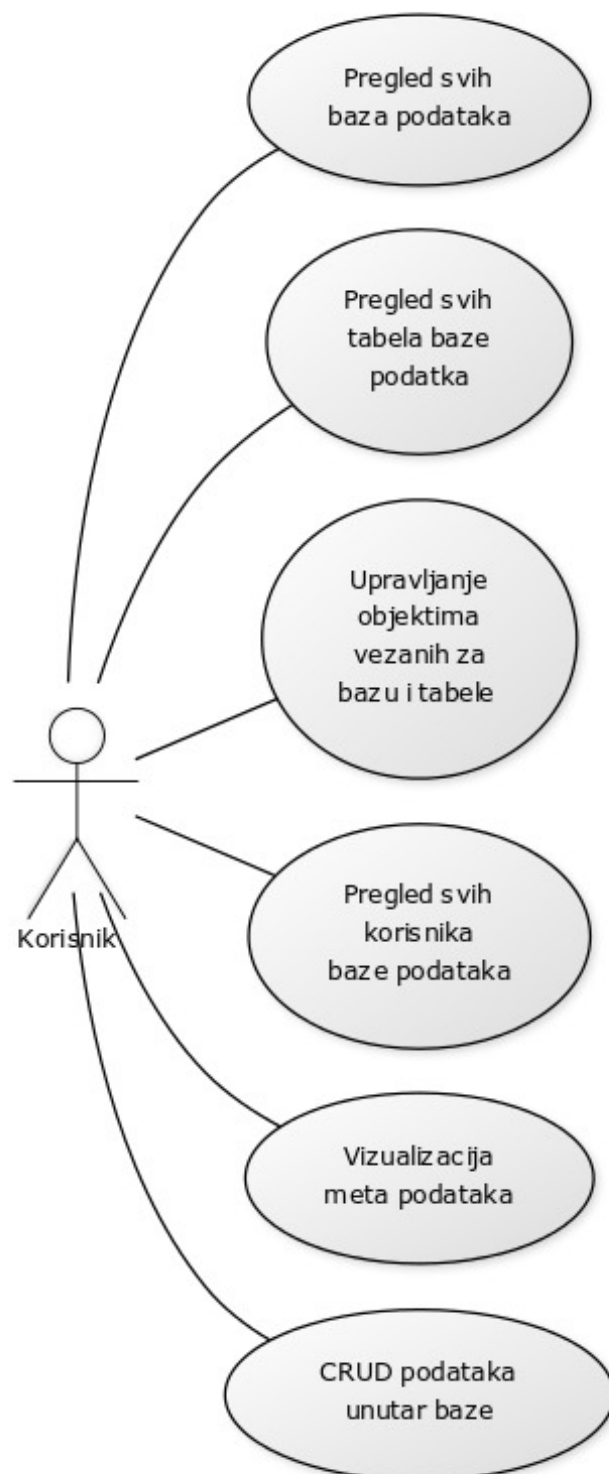
Namjena database handler-a i meta data visualizer-a jeste sama analiza Oracle baze podataka koja postoji na fakultetu, te pretvaranje gomile meta podataka u razumljivu formu koja može dati dosta korisnih informacija. Pod tim se također podrazumijeva i vizualizacija strukture same baze podataka kao i objekata koji su pohranjeni u istoj. Omogućene su izmjene procedura, trigera, odnosno svih objekata baze. Pored izmjena, omogućeno je i dodavanje i brisanje istih, te pregled podataka baze i dodavanje, brisanje i izmjena podataka baze.

### 1.1. Funkcionalnosti sistema DHMV

Funkcionalnosti sistema su podijeljene u dvije kategorije: upravljanje bazom i vizualizacija meta data. Vizualizacija meta data omogućava korisniku da vidi odgovarajuće podatke o bazi, o svojoj bazi, odnosno vrši prevođenje istih u oblik razumljiv korisniku, a upravljanje omogućava uređivanje objekata, tabela, itd.

- Pregled svih baza podataka;
- Pregled svih tabela baze podataka;
- Upravljanje objektima vezanih za bazu i tabele;
- Pregled svih korisnika baze podataka;
- Vizualizacija meta podataka;
- Prikaz, kreiranje, uređivanje i brisanje podataka unutar baze.

U nastavku je prikazan dijagram slučajeva upotrebe.



## Poglavlje 2.

# Detaljni opis funkcionalnosti

U ovom poglavlju su detaljno opisane funkcionalnosti aplikacije. Poslovni procesi koji se ostvaruju kroz ovu aplikaciju jesu vizualizacija metapodataka i ostalih relevantnih informacija o organizaciji baze, shemi baze, objektima i slično. Pod pojmom baze podataka u ovom radu smatra se jedna baza podataka nekog sistema kojima je moguće pristupiti kroz jednu konekciju, a ne skupa baza podataka. Svim funkcionalnostima ove aplikacije se pristupa pomoću grafičkog okruženja. Prije svega, korisnik aplikaciji pristupa unosom korisničkog računa i lozinke (ovom procesu podliježe validacija unesenih vrijednosti).

### 2.1. Pregled svih baza podataka

Nakon uspješne prijave na aplikaciju, korisniku se prikazuje lista svih baza koje se nalaze na jednom sistemu. O svim bazama su date informacije njihovih vlasnika i vremena kreiranja. Dalje se korisniku omogućava odabir pojedinačne baze podataka u svrhu pribavljanja detaljnih informacija koje su relevantne korisniku. Pod tim informacijama se smatraju informacije o objektima kao što su tabele, objekti, itd.

### 2.2. Pregled svih tabela baze podataka

Nakon što korisnik odabere željenu bazu podataka, prikazuju se sve tabele koje se nalaze u odabranoj bazi podataka. Podaci koji se prikazuju pored naziva tabele su nazivi kolona, ograničenja nad kolonama, primarnom i sekundarnom ključu i informacija o tipu podatka i veličine kolone. Pored navedenog, korisnik će moći odabrati tabelu u svrhu otkrivanja većeg skupa informacija nad samom tabelom. Relevantne informacije koje će korisnik moći prikupiti ovom funkcionalnosti su opisane u nastavku teksta. Pored mogućnosti za odabirom željenih objekata, korisnik će moći pristupiti podacima koje se nalaze u tabeli. Nad tako prikuljenim podacima neće biti omogućeno njihova izmjena

radi sigurnosti i konzistentnosti podataka. Podaci će biti prikazani u formi lakše razumljivoj čovjeku, odnosno pomoću tabele.

## **2.3. Upravljanje objektima vezanih za bazu i/ili tabelu**

Nakon odabira tabele, korisniku se prikazuju svi kreirani objekti koji su vezani za nju. Pod objektima se smatraju indeksi, funkcije, trigeri, pogledi, greške i sekvence. Prikazani podaci će biti kategorizirani te će biti omogućen pristup svakom objektu u cilju pregleda, izmjene ili brisanja istog. Rad nad objektima će se vršiti putem grafičkog interfejsa. Unutar ove funkcionalnosti se podrazumijeva:

- Kreiranje objekata baze;
- Kreiranje tabela;
- Kreiranje pogleda;
- Kreiranje procedura;
- Kreiranje trigera;
- Kreiranje PL/SQL, source objekata;
- Prikaz ERD baze.

## **2.4. Pregled svih korisnika baze podataka**

Ovaj poslovni proces omogućava pribavljanje informacije o svim korisnicima baze podataka. O nima će se dati pregled broja kreiranih objekata, baza podataka nad kojim imaju privilegije te ostalim relevantnim informacijama koje će dati pobliži uvid u informacije o pojedinom korisniku.

## 2.5. Vizualizacija meta podataka

Vizualizacija meta podataka omogućava korisniku pregled svih informacija o samoj bazi podataka, koji su prevedeni na jezik razumljiv korisniku. Obuhvata:

- Pregled svih objekata baze;
- Pregled svih tabela;
- Pregled svih pogleda;
- Pregled svih argumenata funkcija i procedura;
- Pregled svih procedura;
- Pregled svih triggera;
- Pregled svih PL/SQL, source objekata;
- Prikaz ERD baze.

## 2.6. Prikaz, kreiranje, uređivanje i brisanje podataka unutar baze

Data funkcionalnost omogućava korisniku da vidi podatke u tabelama baze, te da dodaje nove redove, uređuje i briše postojeće redove.

## Poglavlje 3.

# Tehnologija

Stack <sup>1</sup> tehnologija koji je odabran za implementaciju ove web aplikacije se sastoji iz:

- Java Spring - back-end;
- AngularJS - front-end;
- PostgreSQL - interna baza podataka;
- OracleDB - baza podataka koja se analizira;
- Android SDK.

U nastavku će biti pojedinačno opisane tehnologije, kao i motivacija za njihovo korištenje.

### 3.1. Java Spring

Java Spring predstavlja aplikacijski framework za razvoj poslovnih aplikacija (engl. enterprise application) korištenjem Java programskog jezika koji podržava višenamjensko programiranje i striktno definisan konfiguracijski model, nezavisan od produkcijske platforme. Spring obezbjeđuje posebnu podršku za razvoj web aplikacija i web zasnovanih arhitekturnih rješenja. Posjeduje veliki broj biblioteka koji se na zvaničnoj stranici razvojnog tima mogu naći pod sekcijom Spring Projects. U nastavku će biti predstavljeni neki od njih.

---

<sup>1</sup>Stack - izraz koji se u IT svijetu koristi da opiše skupinu softverskih komponenti koje kao takve tvore platformu na kojoj će funkcionisati određena aplikacija ili skupina aplikacija. Neki od poznatijih tehnoloških Stack-ova su WAMP ( Windows Apache MySQL PHP) i LAMP ( Linux Apache MySQL PHP)



### 3.1.1. Spring MVC

Spring MVC predstavlja framework za razvoj web aplikacija koji su zasnovani na Model View Controller <sup>2</sup> arhitekturnom paternu. Nudi jasno definisane klase i interfejs, odnosno gotove komponente za razvoj fleksibilnih i prilično neovisnih web aplikacija. Zahvaljujući MVC paternu, razdvojeni su različiti aspekti web aplikacije na sljedeći način:

- Model enkapsulira podatke aplikacije
- View je odgovoran za rendanje podataka modela i generalno generiše HTML izlaz kojeg klijentski pretraživač može interpretirati
- Controller je odgovoran za procesiranje korisničkih zahtjeva i izgradnju odgovarajućeg modela, te prosljeđivanje modela do view-a kako bi se izrendao odgovarajući prikaz rezultata.

Na ovaj način se postiže poprilična međuneovisnost navedenih komponenti.

### 3.1.2. Spring Data JPA

Spring Data JPA predstavlja object-relational mapping framework (ORM), odnosno apstrakciju za rad sa relacionim bazama podataka. Izgrađen je nad Hibernate-om, poznatim ORM framework-om koji predstavlja de facto standard za objektno-relaciono mapiranje u svijetu Java programiranja, na način da je konfigurisanje i briga o upravljanju sesijama i izuzecima prepušteno Spring Data JPA, što znatno ubrzava razvoj aplikacija. Pored toga, implementacija pojedinih pristupnih metoda često koristi princip “konvencija prije nego li konfiguracija” (engl. convention over configuration), što znači da ako se pišu metode u skladu sa Spring konvencijom, dodatno se smanjuje količina “suvišnog koda” koji bi u drugim slučajevima bilo potrebno pisati za konfigurisanje.

## 3.2. AngularJS

AngularJS je u potpunosti JavaScript-bazirani front-end web aplikacijski framework. Predstavlja open source softver koji je razvijan i održavan od strane Google-a. Osnovna ideja s kojom je razvijan je bila da ponudi jednostavno i elegantno rješenje prilikom razvoja tzv. engl. Single Page Applications<sup>3</sup>.

---

<sup>2</sup>MVC - arhitekturni patern kod kojeg se sistem sastoji od tri dijela na način da je poslovna logika razdvojena od prikaza

<sup>3</sup>Single Page Application - bukvalni prevod za ovaj termin je “Jednostranična aplikacija”. Riječ je o aplikacijama koje daju privid fluidnosti po uzoru na desktop aplikacije. SPA ne koriste navigiranje između većeg broja stranica, već samo mijenjaju neke elemente (primjer Gmail)

Zasniva se na MVC ( Model - View - Controller ) i MVVM<sup>4</sup> ( Model - View - Viewmodel ) arhitekturnim paternima, čime se teži ubrzati cjelokupan proces razvoja softvera.

AngularJS je građen na filozofiji da deklarativno programiranje<sup>5</sup> treba biti korišteno za dizajn korisničkih interfejsa i povezivanje softverskih komponenti, dok je imperativno programiranje<sup>6</sup> rezervisano za poslovnu logiku. Deklarativno programiranje je predstavljeno kroz tzv. Angular direktive. One su elementi koji su po izgledu slični HTML atributima. Postavljaju se u različite HTML elemente i tako, deklarativnim putem, omogućavaju niz različitih funkcionalnosti. Dodatna pozitivna stvar kod direktiva je što olakšavaju shvatanje dizajna, jer se svrha nekog elementa označenog direktivom odmah jasna. U skladu sa principom proširivosti, pored dostavljenih direktiva, korisnik može da razvija vlastite direktive.

Još jedna odznačajnijih mogućnosti koje ovaj framework-a nudi je dvosmjerno vezivanje podataka ( engl. two-way data binding ). Omogućava vezanje podataka iz modela za elemente prikaza na jednostavan način. Servis pod nazivom \$scope prati promjene na modelu kroz niz ciklusa koji se Digest ciklusi. Nakon što otkrije da postoje izmjene u modelu ili u prikazi pravi odgovarajuće izmjene na drugoj strani. Na ovaj način se postiže transparentno dvosmjerno vezivanje podataka. Korištenje ovoga znatno ubrzava proces razvoja jer je pristup traženim podacima jednostavan i vrlo je jednostavno dodati validacije.

### 3.3. PostgreSQL

PostgreSQL, ili skraćeno Postgres, predstavlja objektno-relacioni sistem za upravljanje bazom podataka ( ORMDBMS ). Kao database server ima ulogu da spašava i čuva podatke, te da vraća podatke koje zatraže u okviru zahtjeva druge softverske aplikacije. Kao platforma može se koristiti i na Windows, i na Linux i na macOS operativnim sistemima. PostgreSQL je razvijen od strane PostgreSQL Global Development Group-a, odnosno skupine manjih kompanija i individualnih saradnika. Besplatna je i open-source, licencirana sa PostgreSQL License softverskom licencom.

---

<sup>4</sup>MVVM - arhitekturni patern koji je konceptualno sličan MVC patternu

<sup>5</sup>Deklarativno programiranje - ovaj stil programiranja je zasnovan na korištenju oznaka koje imaju eksplicitno značenje, putem kojih se željeni rezultat opisuje. Ne koriste se komande ili instrukcije koje služe za izvršavanje akcija u definisanom redoslijedu. HTML je primjer deklarativnog jezika

<sup>6</sup>Imperativno programiranje - stil programiranja suprotan deklarativnom programiranju kod kojeg se funkcionalnosti ostvaraju tako što se pišu komande koje program određenim redoslijedom izvršava

## 3.4. OracleDB

Oracle je relacionalna baza podataka (RDBMS) koju proizvodi Oracle Corporation. Sistem se zasniva na framework-u relacionih baza podataka, u kojem korisnici pristupaju podacima direktno ili preko aplikacijskog front end-a kroz strukturisani jezik upita SQL. Skalabilna je relacionalna baza podataka i često je koriste velike kompanije, koje rade s podacima diljem svijeta. Sadrži vlastitu mrežnu komponentu koja omogućava komunikaciju.

## 3.5. Android

Aplikacije za Android su napisane u programskom jeziku Java. Za razvoj Android aplikacija koristi se skup alata nazvan Android SDK (software development kit).

Android aplikacije se izvršavaju na Android operativnom sistemu koji je instaliran na uređaju (ili virtuelnoj mašini). Android operativni sistem je varijanta Linux operativnog sistema gdje je svaka aplikacija drugačiji korisnik sa jedinstvenim ID brojem. Za svaku aplikaciju se postavljaju permisije, tako da sve fajlove te aplikacije može koristiti korisnik (tj. aplikacija) sa navedenim ID brojem.

Osnovne komponente Android aplikacije su:

- Aktivnosti – Aktivnost predstavlja jedan ekran sa svojim korisničkim interfejsom. Na primjeru aplikacije za pregled slika, jedna aktivnost može biti ekran na kojem je prikazana lista svih slika, drugi npr. lista opcija za selektovanu sliku.
- Servisi – Servis je komponenta koja se izvršava u pozadini i obavlja operacije koje se dugo izvršavaju ili radi za neke vanjske procese. Servis ne obezbjeđuje korisnički interfejs. Primjer jednog servisa je preuzimanje podataka sa web servera za aplikaciju vremenske prognoze.
- Content provider-i – Content provider upravlja sa dijeljenim skupom podataka aplikacije. Takvi podaci mogu biti sačuvani na file sistemu, SQLite bazi, web-u ili nekoj drugoj lokaciji.
- Broadcast resiveri – Broadcast resiver je komponenta koja odgovara na broadcast objave. Mnogi broadcasti potiču od samog sistema – npr. obavještenje da je baterija skoro prazna. Broadcasti nemaju svoj korisnički interfejs, ali mogu kreirati notifikaciju u statusnoj traci.

## Poglavlje 4.

# Web aplikacija

Izgled naslovne stranice prikazan je na sljedećoj slici:

BP007 Prijava Registracija O nama

Dobrodošli !  
Molimo unesite vaše korisničke podatke

Korisnički mail

Password

PRIJAVI SE

Slika 4.1. Naslovna stranica

Omogućeno je kreiranje vlastitog računa, klikom na dugme za registraciju.

BP007 Prijava Registracija O nama

Novi korisnik ?  
Molimo unesite vaše korisničke podatke

Ime korisnika Prezime korisnika

Korisnički mail

Password

Ponovite password

REGISTRUJ SE

Slika 4.2. Registracija

Nakon prijave, otvara se stranica pomoću koje se uspostavlja konekcija na bazu nakon unosa odgovarajućih podataka.

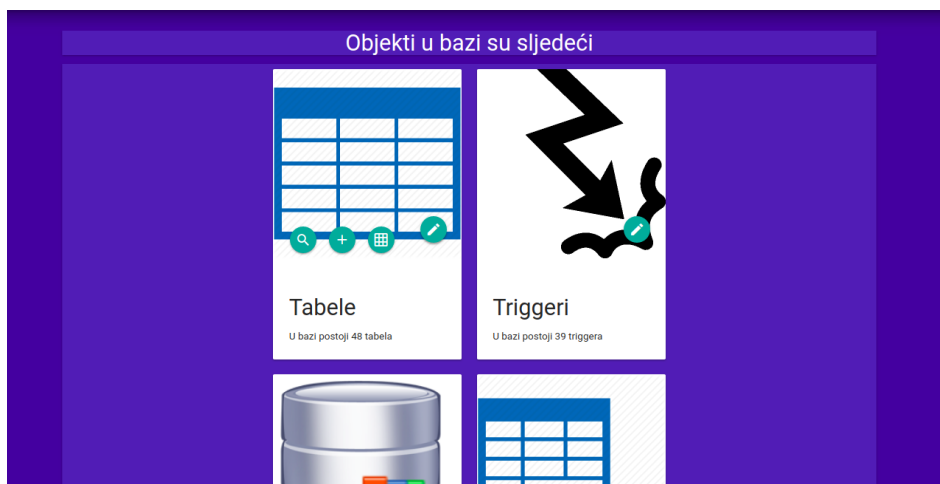
The screenshot shows a web interface with a purple header bar. On the left, it says 'BP007' and on the right, 'Odjavi se'. Below the header, there's a blue box with the text 'Dobrodošli, Šeila !'. Underneath this, there's a large light blue area containing several input fields: 'Ime hosta', 'Port aplikacije', 'SID', 'Korisničko ime za bazu', and 'Korisnički password za bazu'. At the bottom of this area is a green button labeled 'POVEZI SE SA BAZOM'.

Slika 4.3. Stranica za unos konekcijskih podataka

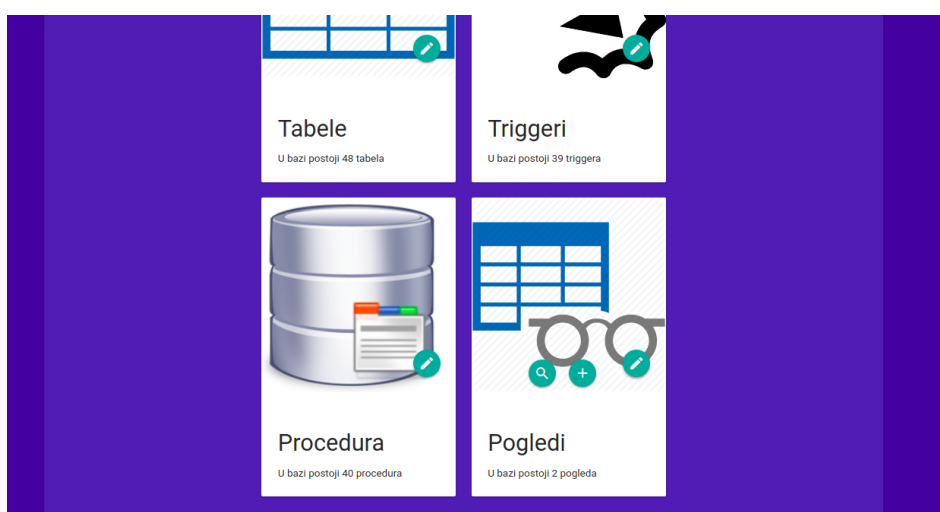
This screenshot shows the same web interface as Slika 4.3, but with a white modal dialog box in the center. The dialog has the title 'Dobrodošao BP07' and the message 'Uspješno ste se konektovali na bazu BP07 !'. There is an 'OK' button in the bottom right corner of the dialog. Behind the dialog, the input fields are visible and filled with values: 'Ime' is 'ETFLAB', 'Port aplikacije' is 'BP07', and 'Korisnički password za bazu' is masked with dots. The green 'POVEZI SE SA BAZOM' button is still at the bottom.

Slika 4.4. Uspostavljanje konekcije na bazu

Nakon ostvarivanja konekcije na bazu korisniku su prikazani objekti baze.



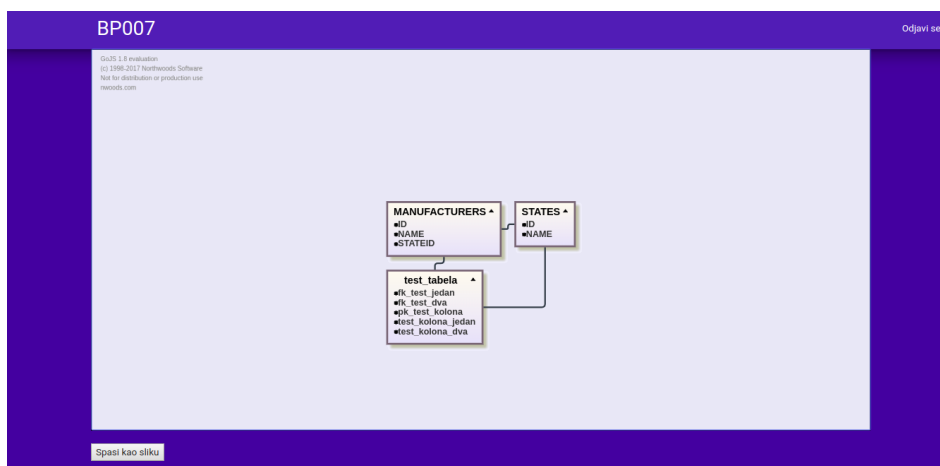
Slika 4.5. Objekti baze a



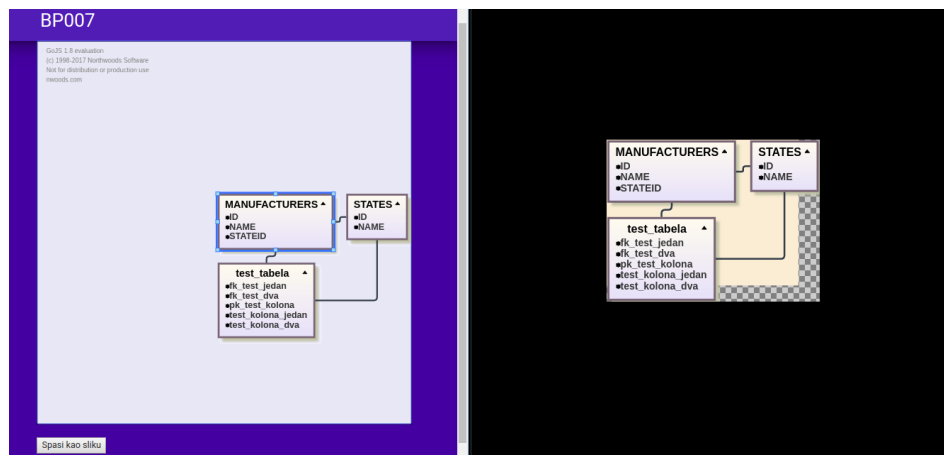
Slika 4.6. Objekti baze b

Za tabele su ponuđene opcije: pregleda svih tabela, kreiranja nove tabele i kreiranje ERD-a, kojeg je moguće preuzeti kao sliku. U nastavku slijedi prikaz slika datih opcija.

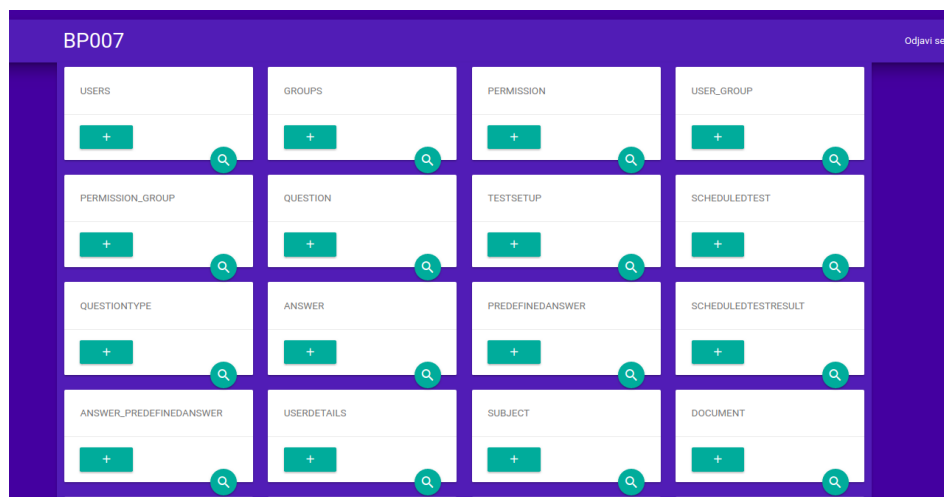
Slika 4.7. Kreiranje tabele



Slika 4.8. Crtanje ERD-a



Slika 4.9. Preuzeta slika ERD-a



Slika 4.10. Tabele baze



U sklopu prikaza tabela baze, moguće je kreiranje indeksa na svakoj tabeli i prikaz detalja iste.

Slika 4.11. Kreiranje indeksa

TABLE_NAME:	GROUPS
TABLESPACE_NAME:	ETFLAB
CLUSTER_NAME:	null
IOT_NAME:	null
STATUS:	VALID
PCT_FREE:	10
PCT_USED:	null
INI_TRANS:	1

Slika 4.12. Detalji tabele

Za triggera su ponuđene opcije: pregled svih triggera, kreiranje novog triggera.

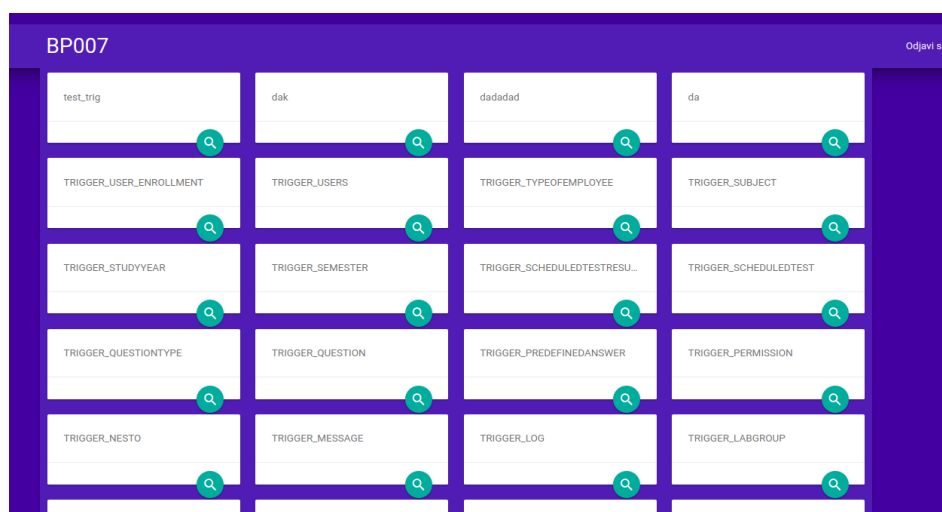
The screenshot shows a web application interface with a dark blue header. On the left, the text 'BP007' is displayed. On the right, there is a link 'Odjavi se'. The main content area has a title 'Kreiranje triggera' in a light blue bar. Below this, there is a form with several input fields: 'Naziv triggera', 'BEFORE', 'UPDATE, INSERT, DELETE', 'GROUPS', and 'Deklaracija varijabli triggera'. A checkbox labeled 'FOR EACH ROW' is checked, indicated by a green checkmark icon.

Slika 4.13. Kreiranje triggera a

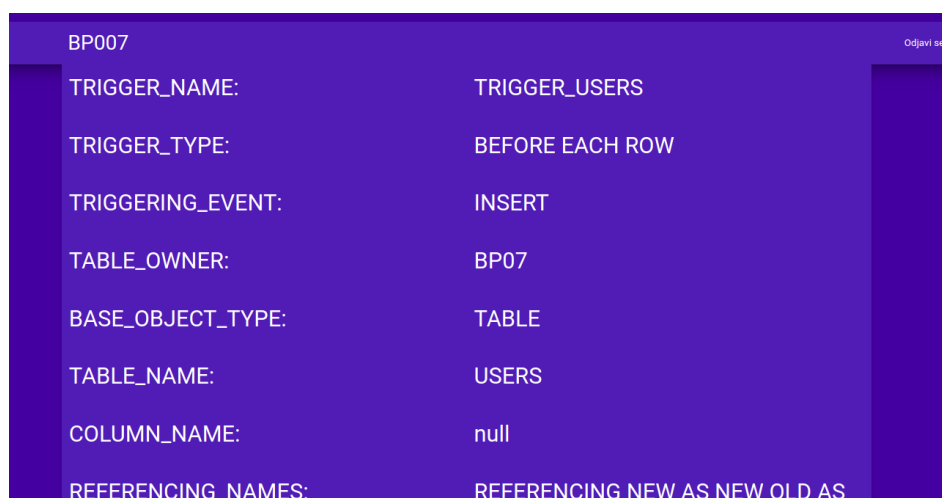
This screenshot is identical to the one above, showing the 'Kreiranje triggera' form. It includes the same header with 'BP007' and 'Odjavi se', the form title, and the input fields for trigger creation, with the 'FOR EACH ROW' checkbox checked.

Slika 4.14. Kreiranje triggera b

U sklopu prikaza triggera baze, moguć je prikaz detalja odabranog triggera.



Slika 4.15. Prikaz svih triggera



Slika 4.16. Detalji odabranog triggera

Za pogled su ponuđene opcije: pregled svih pogleda, kreiranje novog pogleda.

Slika 4.17. Kreiranje pogleda

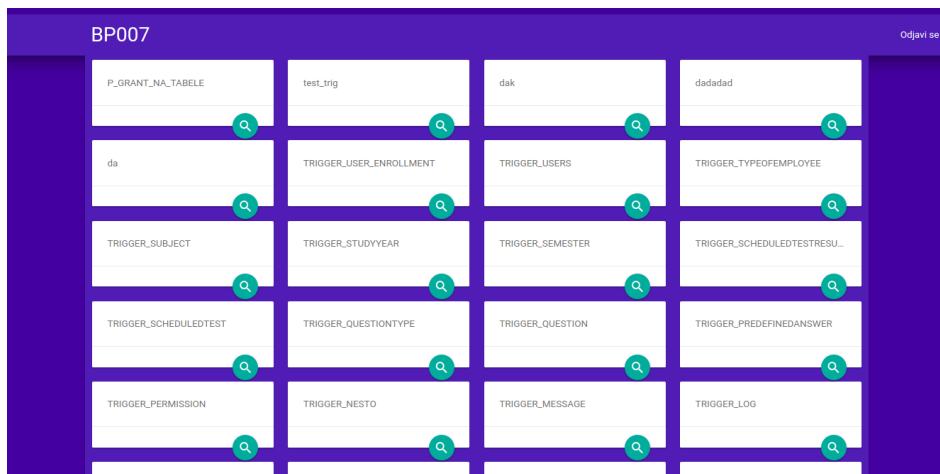
U sklopu prikaza pogleda baze, moguć je prikaz detalja odabranog pogleda.

Slika 4.18. Prikaz svih pogleda

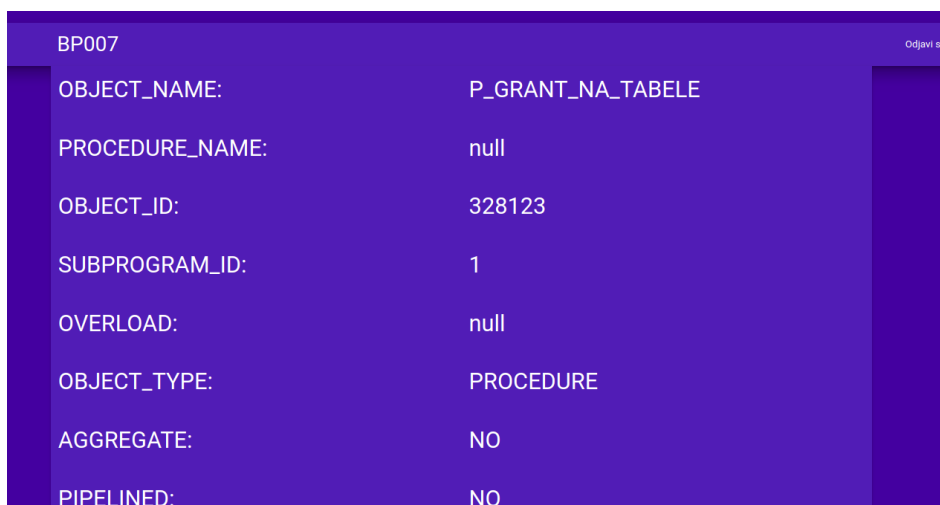
VIEW_NAME:	POGLEDA
TEXT_LENGTH:	44
TEXT:	SELECT USERS.ID FROM USERS WHERE USERS.ID>10
TYPE_TEXT_LENGTH:	null
TYPE_TEXT:	null
OID_TEXT_LENGTH:	null
OID_TEXT:	null

Slika 4.19. Detalji pogleda

Za proceduru je ponuđena opcija pregled svih procedura. U sklopu prikaza procedura baze, moguć je prikaz detalja odabrane procedure.



Slika 4.20. Prikaz svih procedura

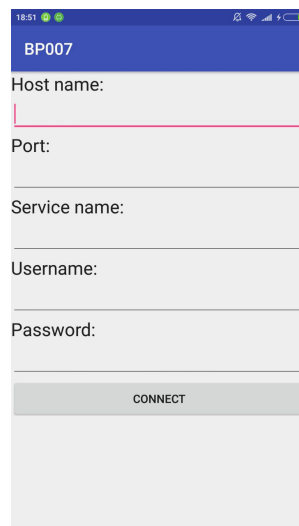


Slika 4.21. Detalji procedure

# Poglavlje 5.

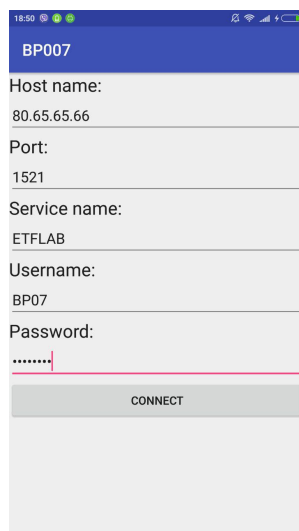
## Android aplikacija

Početni ekran Android aplikacije:



Slika 5.1. Početna aktivnost

Nakon unosa odgovarajućih podataka, pritisnemo dugme za prijavu:



BP007

Host name:  
80.65.65.66

Port:  
1521

Service name:  
ETFLAB

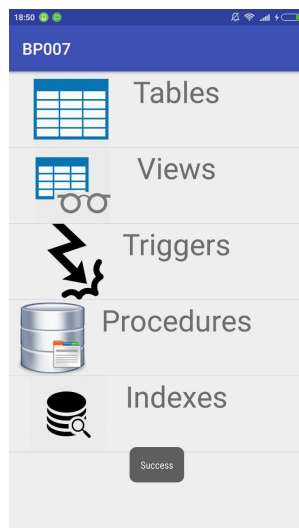
Username:  
BP07

Password:  
.....

CONNECT

Slika 5.2. Početna aktivnost nakon unosa podataka

Nakon uspješne prijave, prelazi se u novu aktivnost na kojoj je prikazana lista objekata baze. U slučaju neuspjeha prikaže se poruka o grešci.



BP007

Tables

Views

Triggers

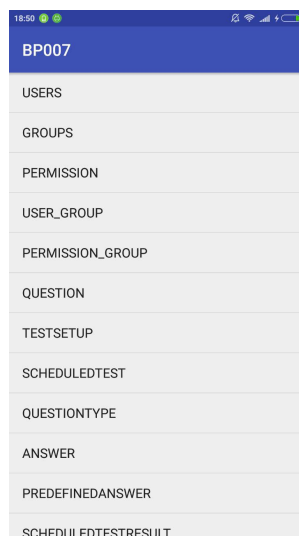
Procedures

Indexes

Success

Slika 5.3. Lista objekata baze

Nakon odabira opcije Tables, prikazuje se lista istih.

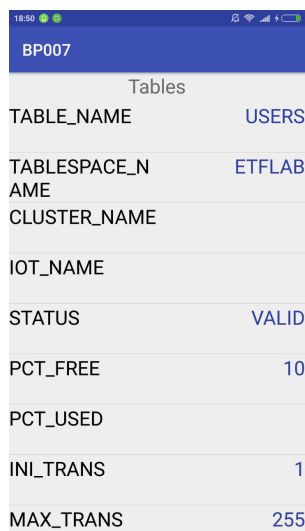


The screenshot shows a mobile application interface with a blue header bar labeled 'BP007'. Below the header is a list of database tables. The tables are listed in a single column, each on a separate line. The tables are: USERS, GROUPS, PERMISSION, USER\_GROUP, PERMISSION\_GROUP, QUESTION, TESTSETUP, SCHEDULEDTEST, QUESTIONTYPE, ANSWER, PREDEFINEDANSWER, and SCHEDULEDTESTRESULT.

BP007
USERS
GROUPS
PERMISSION
USER_GROUP
PERMISSION_GROUP
QUESTION
TESTSETUP
SCHEDULEDTEST
QUESTIONTYPE
ANSWER
PREDEFINEDANSWER
SCHEDULEDTESTRESULT

Slika 5.4. Lista tabela

Nakon odabira tabele otvara se nova aktivnost na kojoj su prikazani detalji o tabele.



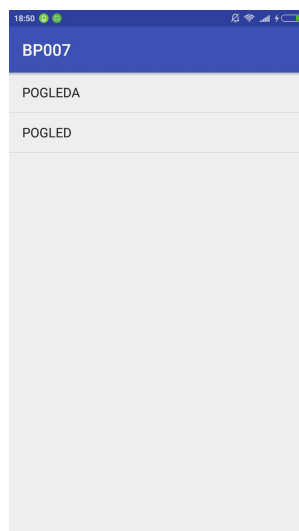
The screenshot shows a mobile application interface with a blue header bar labeled 'BP007'. Below the header is a section titled 'Tables'. Under this section, the details of a selected table are displayed. The details are shown in a two-column format, with the table name 'USERS' in blue text. The details are: TABLE\_NAME (USERS), TABLESPACE\_NAME (ETFLAB), CLUSTER\_NAME, IOT\_NAME, STATUS (VALID), PCT\_FREE (10), PCT\_USED, INI\_TRANS (1), and MAX\_TRANS (255).

Tables	
TABLE_NAME	USERS
TABLESPACE_NAME	ETFLAB
CLUSTER_NAME	
IOT_NAME	
STATUS	VALID
PCT_FREE	10
PCT_USED	
INI_TRANS	1
MAX_TRANS	255

Slika 5.5. Prikaz detalja odabrane tabele

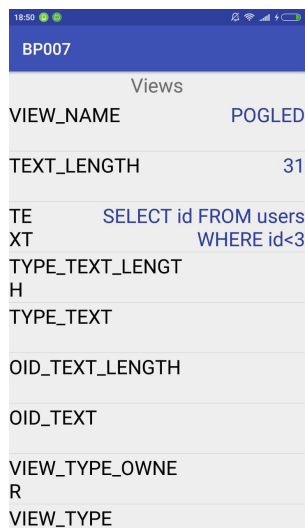


Nakon odabira opcije Views, prikazuje se lista istih.



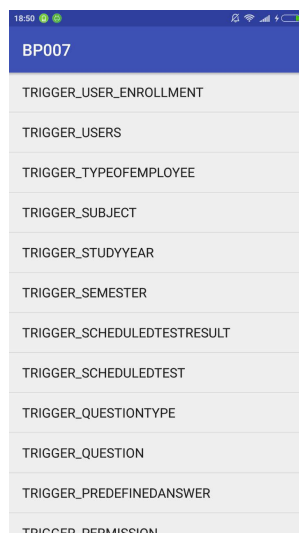
Slika 5.6. Lista pogleda

Nakon odabira pogleda otvara se nova aktivnost na kojoj su prikazani detalji o pogledu.



Slika 5.7. Prikaz detalja odabranog pogleda

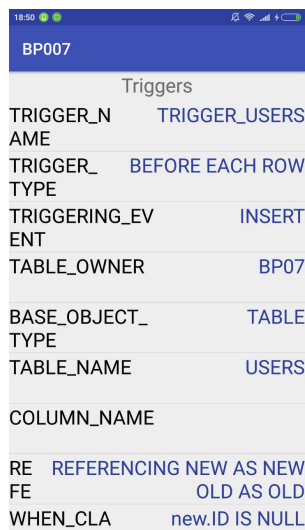
Nakon odabira opcije Triggers, prikazuje se lista istih.



BP007
TRIGGER_USER_ENROLLMENT
TRIGGER_USERS
TRIGGER_TYPEOFEMPLOYEE
TRIGGER_SUBJECT
TRIGGER_STUDYYEAR
TRIGGER_SEMESTER
TRIGGER_SCHEDULEDTESTRESULT
TRIGGER_SCHEDULEDTEST
TRIGGER_QUESTIONTYPE
TRIGGER_QUESTION
TRIGGER_PREDEFINEDANSWER
TRIGGER_PERMISSION

Slika 5.8. Lista triggera

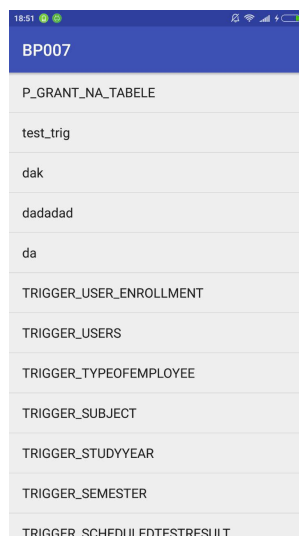
Nakon odabira triggera otvara se nova aktivnost na kojoj su prikazani detalji o triggeru.



Triggers	
TRIGGER_NAME	TRIGGER_USERS
TRIGGER_TYPE	BEFORE EACH ROW
TRIGGERING_EVENT	INSERT
TABLE_OWNER	BP07
BASE_OBJECT_TYPE	TABLE
TABLE_NAME	USERS
COLUMN_NAME	
REFERENCING	REFERENCING NEW AS NEW OLD AS OLD
WHEN_CLAUSE	new.ID IS NULL

Slika 5.9. Prikaz detalja odabranog triggera

Nakon odabira opcije Procedures, prikazuje se lista istih.






BP007
P_GRANT_NA_TABELE
test_trig
dak
dadadad
da
TRIGGER_USER_ENROLLMENT
TRIGGER_USERS
TRIGGER_TYPEOFEMPLOYEE
TRIGGER_SUBJECT
TRIGGER_STUDYYEAR
TRIGGER_SEMESTER
TRIGGER_SCHEDULEDTESTRESULT

Slika 5.10. Lista procedura

Nakon odabira procedure otvara se nova aktivnost na kojoj su prikazani detalji o proceduri.

18:51



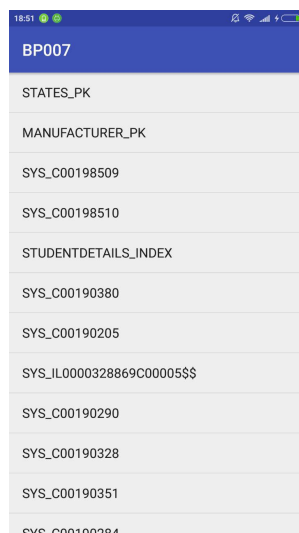
BP007

Procedures

OBJECT_NAME	P_GRANT_NA_TABELE
PROCEDURE_NAME	
OBJECT_ID	328123
SUBPROGRAM_ID	1
OVERLOAD	
OBJECT_TYPE	PROCEDURE
AGGREGATE	NO
PIPELINED	NO
IMPLTYPEOWNER	

Slika 5.11. Prikaz detalja odabranog procedure


Nakon odabira opcije Indexes, prikazuje se lista istih.



BP007
STATES_PK
MANUFACTURER_PK
SYS_C00198509
SYS_C00198510
STUDENTDETAILS_INDEX
SYS_C00190380
SYS_C00190205
SYS_IL0000328869C00005\$\$
SYS_C00190290
SYS_C00190328
SYS_C00190351
SYS_C00190284

Slika 5.12. Lista indeksa

Nakon odabira indeksa otvara se nova aktivnost na kojoj su prikazani detalji o indeksu.



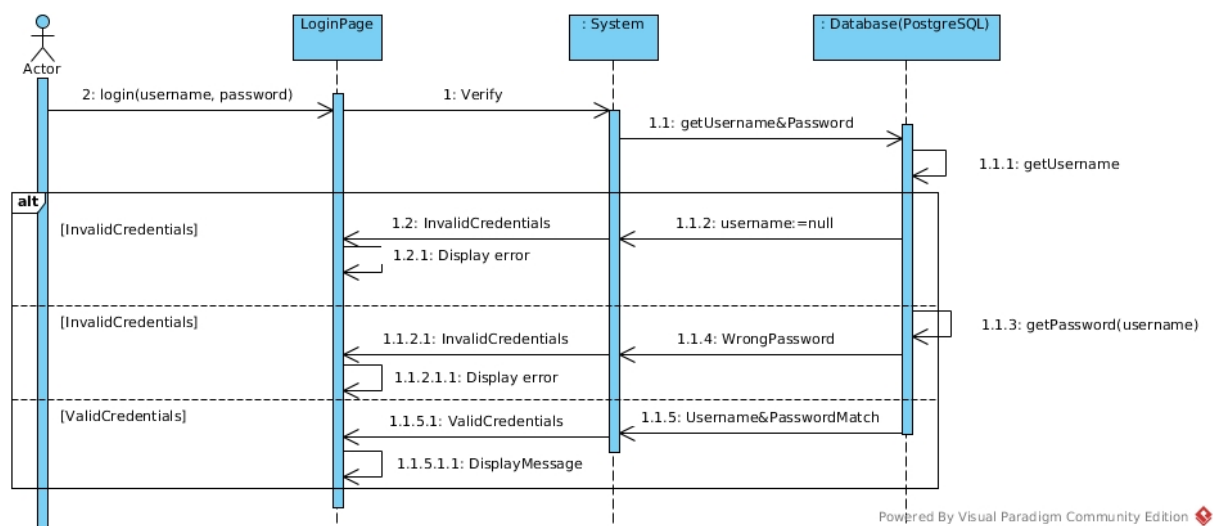
BP007	
Indexes	
INDEX_NAME	STATES_PK
INDEX_TYPE	NORMAL
TABLE_OWNER	BP07
TABLE_NAME	STATES
TABLE_TYPE	TABLE
UNIQUENESS	UNIQUE
COMPRESSION	DISABLED
PREFIX_LENGTH	
TABLESPACE_N	ETFLAB

Slika 5.13. Prikaz detalja odabranog indeksa

## Poglavlje 6.

# Optimizacija upita i transakcije

### 6.1. Login transakcija



Slika 6.1. Login

### 6.2. Funkcija getRowCount()

**SELECT COUNT(\*) as brojRedova FROM + objectName**

Prethodni upit vraća broj redova ciljanog objekta (objectName). U aplikaciji, kroz ovaj upit se pribavljaju informacije o broju korisničkih tabela (tabela unutar sheme), broju procedura, broju pogleda i broju trigeri (napomena: trigeri predstavljaju poseban vid procedura koji se indirektno koriste. To su procedure koje se izvršavaju u pozadini).

### 6.3. Funkcija `getNames()`

**SELECT \* FROM + objectName**

**SELECT + nazivKolone + FROM + objectName**

Prvi upit se koristi za pribavljanje odgovarajućih podataka koje će olakšati dobavljanje željenih informacija iz drugog upita. Drugi upit dobavlja nazive kolona (nazivKolone) iz tabele objectName. Tabela koja je od interesa u ovoj metodi je `user_tables`. Napomena: u metodi `getTableColumns()` korišten je identičan prvi upit za pribavljanje naziva kolona tabele objectName.

### 6.4. Funkcija `getColumnNames()`

**SELECT \* FROM USER\_TAB\_COLUMNS**

**SELECT + nazivTabele + , + nazivKolone + FROM USER\_TAB\_COLUMNS**

Kao u prethodnoj metodi, prvi upit se koristi za pribavljanje podataka. Drugi upit vraća nazive svih korisničkih tabela zajedno sa nazivima kolona pripadajućih tabela.

### 6.5. Funkcija `objectsMetaData()`

**SELECT \* FROM + params.objectName**

**SELECT \* FROM + params.objectName + WHERE + columnName += params.name**

Kao u prethodna dva slučaja, prvi upit dobavlja informacije koje će biti korištene u narednom upitu (pribavljaju se nazivi određenih kolona). Drugi upit ima u cilju vraćanje metapodataka objekta (u ovom slučaju tabela prosljeđenog kao parametar name) koji se nalazi u tabeli objectName (`user_tables`).

### 6.6. Funkcija `createView()`

**String sql = "CREATE VIEW " + params.title + " AS SELECT**  
**”;**

**if (params.kolone.size() != 1)**

**for (int i = 0; i<params.kolone.size() - 1; i++)**

**Sql += params.kolone.get(i) + ", ”;**

**sql += params.kolone.get(params.kolone.size()-1);**

**sql += " FROM ”;**

**if (params.tabele.size() != 1)**

**for (int i=0; i<params.tabele.size()-1;i++)**

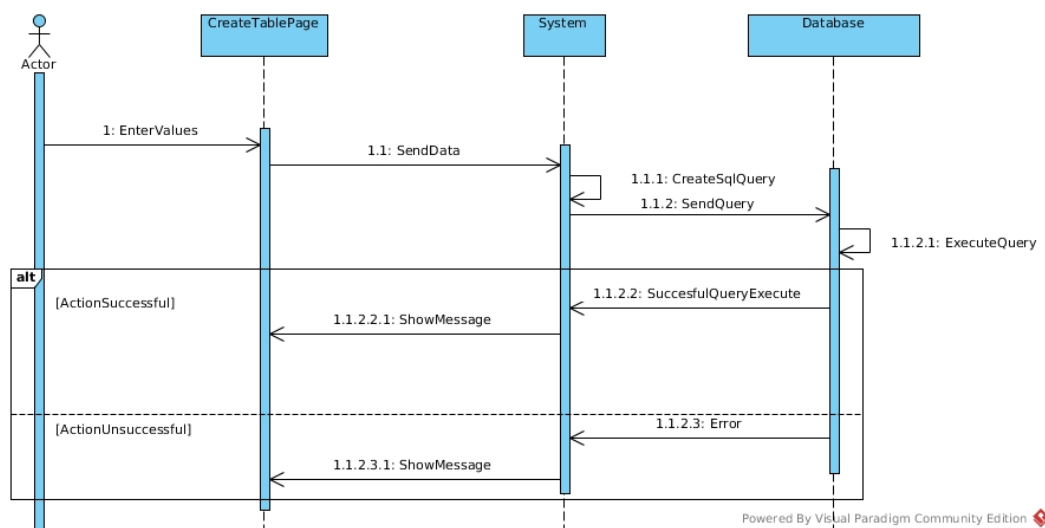
**sql += params.tabele.get(i) +”, ”;**

**sql += params.tabele.get(params.tabele.size()-1);**

**sql += " WHERE ” + params.uslov;**

Prethodni kod ima za zadatak da kreira odovarajuću sekvencu s navedenim parametrima. Za pravilno kreiranje sekvence kroz datu funkciju, potrebno je proslijediti parametre: ime pogleda, kolone za select klauzulu, tabele za from klauzulu i uslov za where klauzulu.

Transakcija:

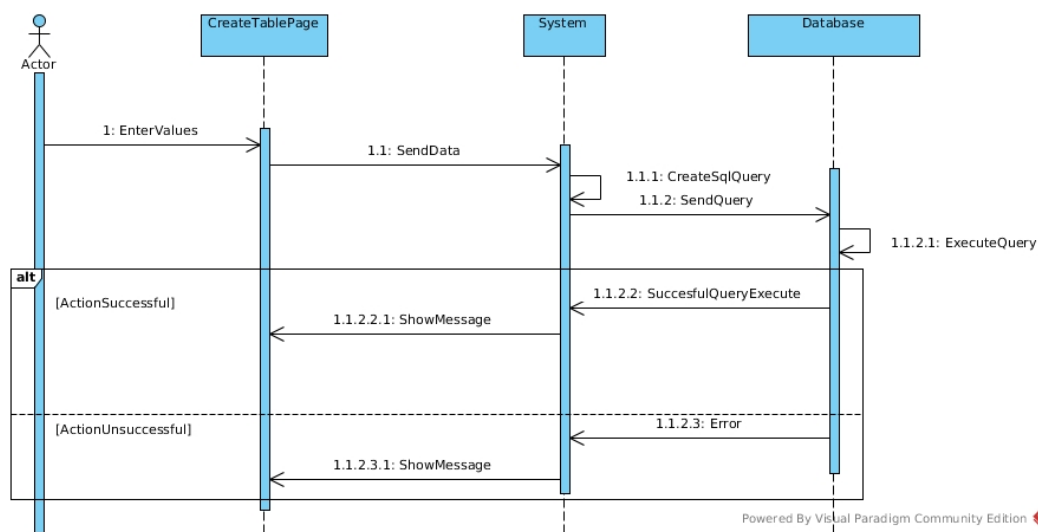


Slika 6.2. Kreiranje pogleda

## 6.7. Funkcija createTrigger()

Kako je tijelo funkcije createTrigger() za kreiranje trigera robustan, u nastavku neće biti priložen, ai će biti objašnjen smisao funkcije. Naime, za kreiranje trigera, potrebni su parametri: naziv trigera, trenutak okidanja trigera, akcija, tabele koje obuhvaća, varijable i kod.

Transakcija:



Slika 6.3. Kreiranje triggera

## 6.8. Funkcija primaryKey()

**SELECT object\_name FROM all\_objects WHERE object\_type = 'TABLE'** Upit vraća sve nazive objekta tipa TABLE iz tabele all\_objects.

## 6.9. Funkcija createIndex()

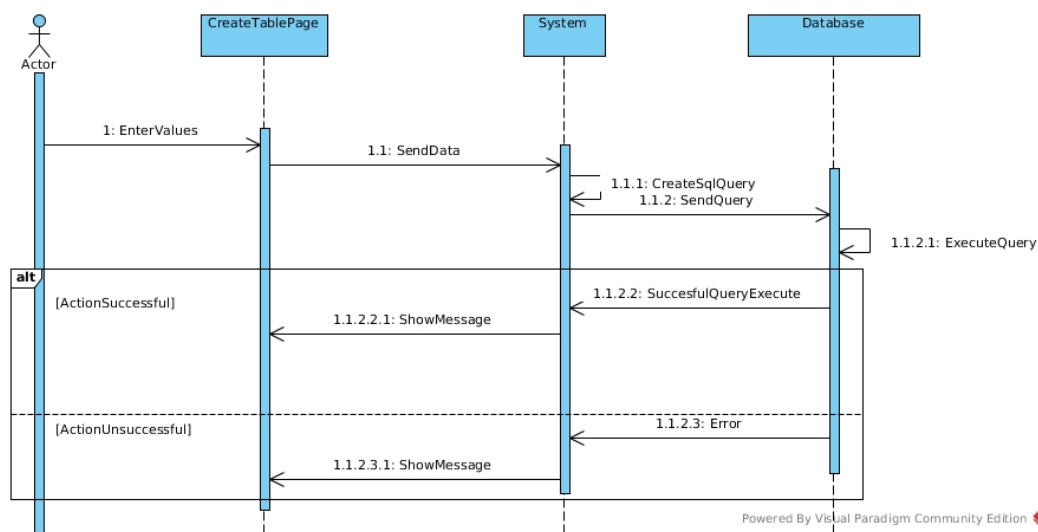
```

CREATE INDEX + params.title + ON + params.table + "(";
for(int i = 0; i < params.columns.size(); i++)
if (params.columns.size() - 1 != i)
sql += params.columns.get(i) + ", ";
else sql += params.columns.get(i) + ")";

```

Prethodno napisan kod kreira indeks naziva title nad tabelom table. Unutar indeksa su obuhvaćene kolone specificirane kroz parametar columns. Transakcija:





Slika 6.4. Kreiranje indeksa

## 6.10. Funkcija showERD()

```

SELECT utc.table_name tableName, utc.column_name columnName,
CASE WHEN (utc.table_name, utc.column_name) IN (
SELECT uc.table_name, ucc.column_name
FROM USER_CONSTRAINTS uc, USER_CONS_COLUMNS ucc
WHERE ucc.table_name = uc.table_name
AND ucc.constraint_name = uc.constraint_name
AND uc.constraint_type IN ('P', 'R')
)
THEN 'true'
ELSE 'false' END AS isKey
FROM USER_TAB_COLUMNS utc, USER_OBJECTS uo
WHERE uo.created >= TO_DATE('20171220', 'YYYYMMDD')
AND utc.table_name = uo.object_name
AND uo.object_type = 'TABLE'
ORDER BY utc.table_name ASC;
  
```

Prethodni upit vraća informacije o nazivu tabela, svih njenih kolona i informacija da li kolona primary ili foreign ključ. Za potrebe projekta u kojem je korišten ovaj upit, u where klauzuli, filtriranje je pojačano sa datumom kreiranja tabela, odnosno da se vraćaju samo one koje su nastale nakon specificiranog datuma.

```

SELECT table_name tableName
FROM USER_CONSTRAINTS
WHERE constraint_type = 'R'
AND r_constraint_name IN(
  
```

```

SELECT constraint_name
FROM ALL_CONSTRAINTS
WHERE table_name = + lastTableName +
AND constraint_type = 'P');

```

Prethodni upit ima za cilj vraćanje svih tabela vezani za specificiranu tabelu (kroz parametar lastTableName) preko primarnog ključa, odnosno sve tabele koje za foreign ključ imaju primary ključ specificirane tabele.

## 6.11. Funkcija create\_AutoIncrement\_Trigger()

```

CREATE SEQUENCE seq_ + nazivTabele + _ + nazivKolone
MINVALUE 1
START WITH 1
INCREMENT BY 1
CACHE 20";
CREATE OR REPLACE TRIGGER sequence_trigger_ + nazivTa-
bele + _ +
nazivKolone +
BEFORE INSERT ON + nazivTabele +
FOR EACH ROW
BEGIN
SELECT seq_ + nazivTabele + _ + nazivKolone + .NEXTVAL
INTO :new. + nazivKolone +
FROM dual;
END;

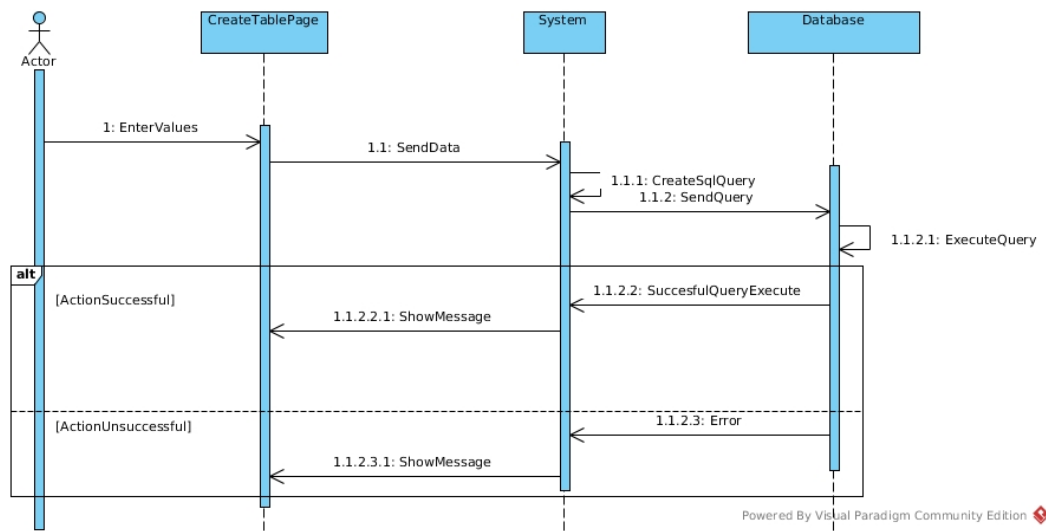
```

Funkcija create\_AutoIncrement\_Trigger() kreira odgovarajuću sekvencu i trigger ukoliko je u funkciji createTable() specificirana opcija autoincrement.

## 6.12. Funkcija createTable()

Radi opširnosti i nepreglednosti koda, isti neće biti priložen u nastavku teksta, već će ukratko biti opisan smisao funkcije. Zadatak funkcije je kreiranje tabele/a na osnovu korisničkih ulaznih parametara. Za kreiranje potrebni su: naziv tabele, i informacije o koloni ili kolonama (primary/foreign ključ, unique tip ili kolona bez ograničenja, tip kolone, da li je nullable, opcija autoincrement). Opcija autoincrement kreira sekvencu i trigger koji ima ulogu nadomještanja vrijednosti ukoliko korisnik nije specificirao istu prilikom unosa.

Transakcija:



Slika 6.5. Kreiranje tabele