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, 05.11.2017.

Sadržaj

Poglavlje 1.

Opis teme

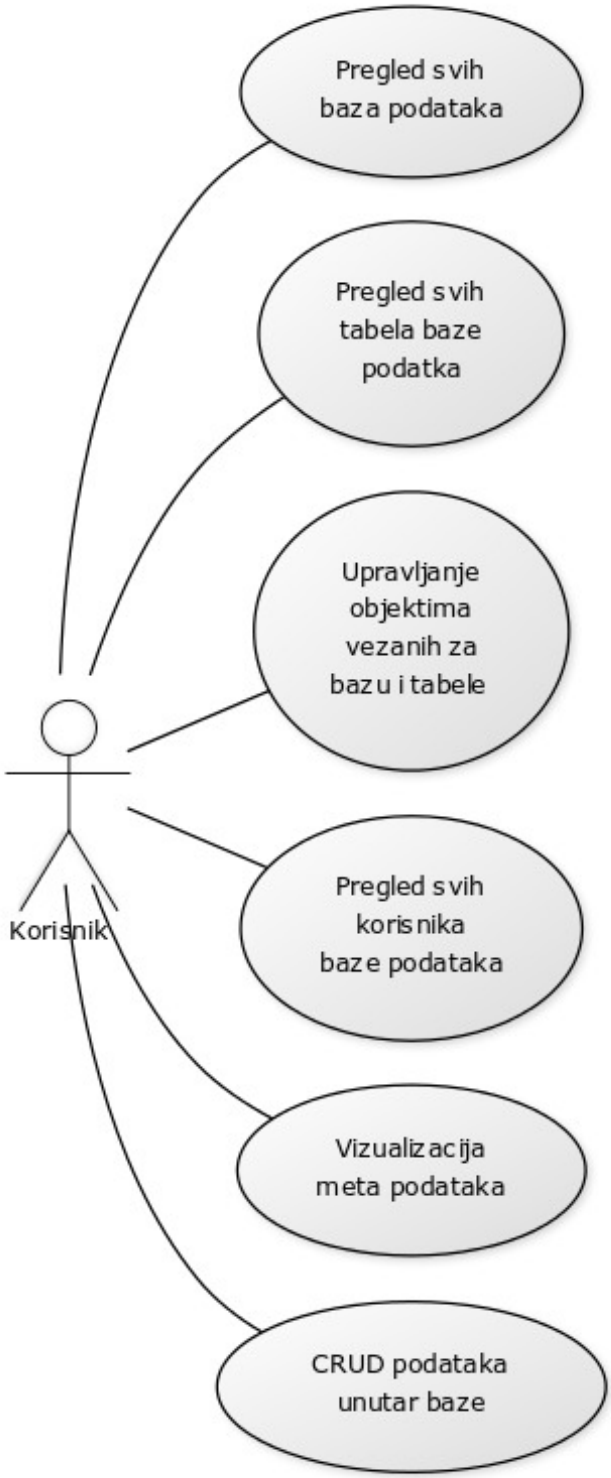
Namjena database handler-a i meta data visualizer-a jeste sama analiza Oracle baze podataka koja postoji na fakultetu, te pretvaranje gomile meta podataka u razumljivu formu koja može dati dosta korisnih informacija. Pod tim se također podrazumijeva i vizualizacija strukture same baze podataka kao i 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 sitema 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 podatka;
- 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.


Pregled svih baza podataka
Pregled svih tabela baze podatka
Upravljanje objektima vezanih za bazu i tabele
Pregled svih korisnika baze podataka
Vizualizacija meta podataka
CRUD podataka unutar baze
Korisnik

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 detaljinih 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, uređivanje i brisanje objekata baze;
- Kreiranje, uređivanje i brisanje tabela;
- Kreiranje, uređivanje i brisanje pogleda;
- Kreiranje, uređivanje i brisanje kolona;
- Kreiranje, uređivanje i brisanje argumenata funkcija i procedura;
- Kreiranje, uređivanje i brisanje error-a;
- Kreiranje, uređivanje i brisanje procedura;
- Kreiranje, uređivanje i brisanje trigera;
- Kreiranje, uređivanje i brisanje PL/SQL, source objekata.

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 kolona;
- Pregled svih argumenata funkcija i procedura;
- Pregled svih error-a;
- Pregled svih procedura;
- Pregled svih trigera;
- Pregled veličina svih objekata;
- Pregled svih PL/SQL, source objekata;

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 [1] 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 arhitekturalnih 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.

[1] 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 [2] arhitekturalnom paternu. Nudi jasno definisane klase i interfejse, 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[3].

[2] MVC - arhitekturalni patern kod kojeg se sistem sastoji od tri dijela na način da je poslovna logika razdvojena od prikaza

[3] Single Page Application - bukvalni prevod za ovaj termin je "Jednostranična aplikacija". Riječ je o apikacijama 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[4] (Model - View - Viewmodel) arhitekturalnim paternima, čime se teži ubrzati cjelokupan proces razvoja softvera.

AngularJS je građen na filozofiji da deklarativno programiranje[5] treba biti korišteno za dizajn korisničkih interfejsa i povezivanje softverskih komponenti, dok je imperativno programiranje[6] 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 korz niz ciklusa koji se Digest ciklusi. Nakon što otkrije da postoje izmijenje 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.

[4]MVVM - arhitekturalni patern koji je konceptualno sličan MVC patternu

[5]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

[6]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 relaciona 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 izvšavaju 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.