Projekat :

**University Data Analysis Tool**

Baze Podataka

|  |  |  |
| --- | --- | --- |
|  | Studenti: | Begić Edin |
| Sarajevo, |  | Mulić Vedad |
| 17.12.2017. |  | Mušić Mirza |

## 1. Uvod u University Data Analysis Tool

Namjena ovog sistema jeste kreiranje skladišta podataka (*data warehouse*) koja će prikupljati odabrane informacije iz baze podataka Elektrotehničkog fakulteta. Prilikom selekcije podataka za skladište, birani su samo oni podaci koji mogu biti od značaja nastavnom osoblju i drugim uposlenicima fakulteta. Ti podaci su orijentirani statističkoj analizi, te su iz tog razloga odabrani podaci koji su detaljno vezani za upis studenata na fakultet, analitički podaci za pojedine ispite, kao i podaci koji služe za monitoring studenata na predavanjima i kursevima. Takve informacije su usko povezane s pojmom poslovne inteligencije (*business intelligence (u nastavku BI)*), odnosno postojanje ovakvog skladišta pomaže fakultetskom osoblju da bolje razumiju trenutno stanje organizacije u kojoj rade.

Drugi dio sistema fokusiran je na kreiranje BI alata. Korištenje BI podrazumijeva posebno odabrane strategije i tehnologije koje su odabrane da bi poboljšali analizu podataka i poslovnih informacija ključne za fakultetsku organizaciju. Ključno za svaki takav alat, pa tako i za *Univerisity Data Analysis Tool* (u nastavku UDAT) jeste vremenski orijentiran prikaz poslovnih informacija. Za ovaj sistem to uključuje:

* Historijski pregled vezan za poslovne informacije fakulteta
* Pregled trenutnog stanja
* Predviđanja vezana za poslovne informacije

Da bi ovaj BI alat bio isplativ s strane fakultetske organizacije, on bi trebao biti dovoljno fleksibilan da se koristi u edukacijske svrhe i dovoljno robustan da pruži kvalitetan *data mashup* (spajanje dva ili više skupa podataka u jedan grafički interfejs), te omogući *drill down* podataka.

Da bi se osigurala privatnost studenata, njihovi lični podaci nisu spašeni u skladištu podataka. Ovakav sistem karakteriše i visok stupanj autorizacije, odnosno pravo prikupljanja podataka iz baze fakulteta ima samo sistemski administrator, dok pregled analiziranih i vizualiziranih podataka može vršiti nastavno osoblje.

## 2. Zahtjevi sistema

Razvoj UDAT sistema može se podijeliti na dva dijela:

1. Administratorski dio
2. Dio namijenjen za fakultetsko osoblje

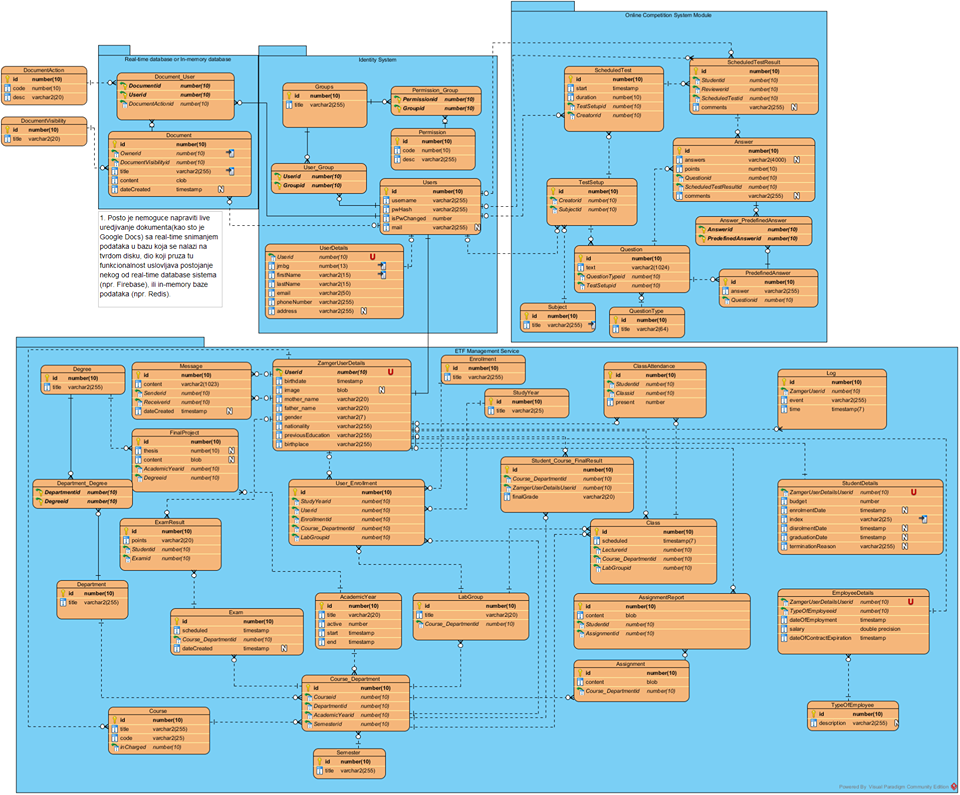
UDAT sistem bi trebao omogućiti administratoru:

* Da u svakom trenutku vrši proces prikupljanja novih podataka iz baze podataka
* Prikupljanje korisničkih podataka koje odgovaraju fakultetskom osoblju, da bi im se omogućila prijava na sistem

Fakultetsko osoblju mora biti omogućeno:

* Mogućnost prijave koristeći web prezentaciju sistema i korisničke podatke prikupljene iz fakultetske baze
* Historijski pregled vezan za broj upisanih studenata i pratećih informacija
* Historijski trend polaganja ispita, uključujući prosjećnu izlaznost i broj bodova
* Historijski trend posjećenosti predavanja određenih predmeta
* Pregled trenutnog stanja za nevedene pojmove
* Predviđanja vezana za broj upisanih studenata u narednoj akademskoj godini na osnovu određenih informacija koje su poznate
* Predviđanja opterećenja nastavnog osoblja na pojedinim kursevima, odnosno obavještenje o potrebi proširenja osoblja

Prilikom izrade modela skladišta podataka bitnu ulogu ima proces migracije podataka iz fakultetske baze, koja je data na sljedećoj slici.



**Slika 2.1** - *Dijagram paketa fakultetske baze*

Iz tog razloga, najčešće se primjenjuje skup procesa **ETL**(izdvajanje, transformacija i učitavanje podataka). Za potrebe skladišta podataka sistema ključan e paket “*ETF Managment Service”* u kome se nalaze poslovne informacije.

Ključni podaci koji se zahtjevaju za uspješnu realizaciju sistema nalaze se u tabelama ZAMGERUSERDETAILS, STUDENTDETAILS,USER\_ENROLLMENT, COURSE\_DEPARTMENT, ACADEMICYEAR, SEMESTER, DEPARTMENT, tj. Iz tih tabela je potrebno “izvuči” podatke vezane za godišnji upis studenata na fakultet. Pored analitičkog prikaza, ovi podaci su važni i radi predviđanja količine upisanih studenata u narednim akademskim godinama, što predstavlja ključnu poslovnu informaciju za fakultet.

Za potrebe analitičkog i vizuelnog prikaza podataka vezane za polaganje ispita, značajne su tabele EXAMRESULT, EXAM, COURSE\_DEPARTMENT iz fakultetske baze podataka. Korištenjem tih tabela moguće je premjestiti podatke kao što su: izlaznost studenata na određeni ispit, prosječan broj bodova, te historijski trend mijenjanja tih atributa (rast/opadanje tokom akademskih godina), itd.

Fakultetu su od velike važnosti i podaci koji daju informacije o posjećenosti predavanja kurseva, odnosno koliko studenata od ukupnog broja upisanih studenata na kursu zaista dolazi na predavanja. To su također poslovne informacije koje omogućavaju vršenje predviđanja o optimizaciji ljudskih resursa fakulteta, odnosno na kursevima gdje je velika posjećenost treba javiti upozorenje da je potrebno zaposliti dodatno nastavno osoblje. Tabele iz fakultetske baze koje sadrže te poslovne informacije su: CLASSATTENDANCE, CLASS, COURSE\_DEPARTMENT.

3. Realizacija sistema

UDAT sistem kao što je već ranije spomenuto može se podijeliti u dvije cjeline, odnosno potrebno je prvo kreirati skladište podataka, te poslije na osnovu podataka iz skladišta kreirati web i Android prezentaciju BI alata.

Ovaj sistem će posjedovati lokalnu bazu tipa *MYSQL* (verzija 5.7) koja će predstavljati skladište podataka, dok *remote* fakultetska baza je rađena koristeći Oracle DBMS. Web aplikacija koja će služiti za prikaz BI alata implementiran je koristeći *Spring* *framework (*baziran na *Java* programskom jeziku) na *backendu* (pri čemu je korišten Maven kao *build* alat), a za kreiranje *frontenda* korišten je *AngluarJS framework*.

3.1 Zašto Spring?

Ovaj sistem će se implementirati koristeći *light* verziju Spring *frameworka*, tzv. Spring Boot. Prednost korištenja ovog frameworka jeste što omogućava brz razvoj aplikacija, što uključuje:

* Povećanje produktivnosti razvojnog tima
* Smanjuje se vrijeme potrebno za razvoj, *unit testiranje* i *integration testiranje* aplikacije, zbog postojanja šablona koji se mogu primjeniti prilikom konfiguracije i razvoja koda
* Olakšava proces razvoja, *unit testiranja i integration testiranja* aplikacije, jer programeri ne moraju “ulaziti duboko” u proces konfiguracije
* Reducira se potreba za pisanjem šablonskog koda, anotacija i XML konfiguracija
* Veoma je jednostavno integrisati Spring boot aplikacije s Spring ekosistemom kao što je *Spring JDBC*, *Spring ORM*, itd.
* Spring Boot aplikacije uključuju unaprijed podešeni HTTP server kao što je *Tomcat*, što ubržava razvijanje i testiranje web aplikacije
* Ovaj framework uključuje CLI (*Command Line Interface*) alat pomoću kojeg se brzo i jednostavno testira i pokreće aplikacija
* Sadrži mnogo *plugina* za razvoj i testiranje aplikacije koristeći *build alate* kao što su *Maven* i *Gradle*
* Sadrži razne vrste *plugin*a*a* za rad s bazama podataka



**Slika 3.1.1** - *Spring framework arhitektura*

3.1.1 Hibernate

U sklopu primjene Spring *frameworka*, koristit će se *framework* baziran na objektno relacionom mapiranju (ORM) - Hibernate. Kako su obje baze korištene u ovom sistemu relacione baze podataka, omogućeno je korištenje ORM *frameworka*. Korištenje Jave kao programskog jezika, podrazumijeva primjenu objektno orijentisanog pristupa. Uvođenjem takvog pristupa u razvoju aplikacija pokazalo je slabosti relacionih baza podataka (klase, složene strukture podataka, polimorfizam itd.). Takvi objekti se ne mogu direktno predstaviti u relacionim bazama podataka u okviru redova i kolona. ORM ima ulogu povezujućeg elementa između dva potpuno različita načina manipulacije podacima, objektno orijentisanog i relacionog. Korisniku se korištenjem objektno relacionog mapiranja uz zanemariv uticaj na performanse daje privid rada sa objektno orjentisanom bazom, pri čemu sam način preslikavanja nema nikakav uticaj na rad korisinka.

Prednosti korištenja Hibernate-a uključuju:

* Smanjenje količine koda potrebnog za osiguranje perzistentnosti podataka čime se programeru ostavlja više vremena za razvoj biznis logike aplikacije
* Lakše održavanje, manje linija koda osigurava bolju razumljivost, a ovako razvijeni kodovi naglašavaju aplikacijsku logiku
* portabilnost , princip transparentnosti koji komunikaciju aplikacije sa ORM-om čini neovisnom od pozadinskog SQL-a omogućava povezivanje aplikacije sa različitim bazama podataka
* Hibernate je *open source framework*

Upiti korišteni za vršenje operacija nad skladištem podataka pisani su u HQL-u (*Hibernate Query Language*), tj. jeziku koji se koristi da se pišu Hibernate upiti. Ovaj jezik karakteriše što ne zavisi od konkretne baze podataka sa kojom Hibernate radi. Prednost takvog pristupa leži u činjenici da je programeru za rad s bazom podataka dovoljno naučiti HQL i svi upiti tako napisani će biti prevedeni u ispravan kod na bazi podataka. Druga prednost je da Hibernate može izvršiti optimizaciju ovakvih upita.

3.2 Komponente vezane za kreiranje skladišta podataka

U prethodnim poglavljima opisani su zahtjevi koje bi skladište podataka u UDAT sistemu trebalo da ispuni kako bi se uspješno kreirao BI alat. Iz tog razloga u ovom poglavlju je detaljno objašnjen proces prijenosa podataka iz fakultetske baze u skladište podataka.

Efikasna migracija podataka u novo skladište je moguće izvršiti primjenom niza procesa koji su poznati pod imenom ETL.

3.2.1 Extraction

**Extraction** - izdvajanje podataka iz fakultetske baze. Takve podatke je potrebno spasiti u ujedinjenom i konzistentnom formatu koji će omogućiti brzi proces transformacije skladištenih podataka. U poglavlju **2** navedene su tabele fakultetske baze iz koje je potrebno izdvojiti određene podatke. U nastavku će biti prikazani dati upiti i njihova namjena.

Za fakultet od najvećeg prioriteta predstavlja poslovna informacija u broju upisanih studenata tokom akademskih godina, a date informacije je moguće *importovati* pisanjem sljedećeg upita:

|  |
| --- |
| SELECT d.id AS dep\_id, ay.id AS ay\_id, s.id AS semester\_id, sd.budget, Count(zud.userid) AS enrolled\_count FROM zamgeruserdetails zud, USER\_enrollment ue, course\_department cd, academicyear ay, semester s,   department d, studentdetails sd  WHERE ue.userid = zud.userid AND ue.course\_departmentid = cd.id AND cd.academicyearid = ay.id  AND cd.semesterid = s.id AND cd.departmentid = d.id AND sd.zamgeruserdetailsuserid = zud.userid GROUP BY d.id, ay.id, s.id, sd.budget ORDER BY d.id, ay.id, s.id, sd.budget; |

Sve informacije vezane za ispite moguće je spasiti sljedećim upitom:

|  |
| --- |
| SELECT cd.academicyearid AS ayid, e.scheduled as full\_date, cd.semesterid as semester\_id,   cd.departmentid as department\_id, cd.courseid as course\_id,  Round(Avg(er.points), 2) AS avg\_points, count(er.studentid) AS turnout FROM examresult er, exam e, course\_department cd WHERE er.examid = e.id AND e.course\_departmentid = cd.id  GROUP BY cd.academicyearid, e.scheduled, cd.semesterid, cd.departmentid, cd.courseid ORDER BY cd.academicyearid; |

U skladištu podataka potrebno je kreirati i tabelu koja će čuvati relevantne podatke o posjećenosti predavanja na kursevima, što se realizuje na sljedeći način:

|  |
| --- |
| SELECT cd.departmentid as department\_id, cd.courseid as course\_id, c.scheduled as full\_date,   c.lecturerid as lecturer\_id,   Sum(ca.present) AS attendance,   Round(Sum(ca.present) / Decode(Count(ca.id), 0, 1, Count(ca.id)), 2) AS attendance\_percentage FROM classattendance ca, class c, course\_department cd WHERE ca.classid = c.id AND c.course\_departmentid = cd.id GROUP BY cd.departmentid, cd.courseid, c.scheduled, c.lecturerid ORDER BY cd.courseid; |

Pored ovih ključnih podataka koji su u skladištu definisani kao fakti (o tome više u poglavlju **4**), postoje i tzv. tabele dimenzije (o njima također detaljnjije u poglavlju **4**), koje se isto tako moraju popuniti, a to je moguće sljedećim upitima:

|  |
| --- |
| SELECT id, title FROM course ORDER BY id; -- Informacije o predmetu/kursu |

|  |
| --- |
| SELECT id, title FROM semester ORDER BY id; -- Informacije o semestru |

|  |
| --- |
| SELECT id, title FROM department; -- Informacije o odsjeku |

|  |
| --- |
| SELECT zud.userid AS id, ud.firstname AS first\_name, ud.lastname AS last\_name,  Decode(  (SELECT Count(\*) FROM zamgeruserdetails zud1, studentdetails sd  WHERE zud1.userid = sd.zamgeruserdetailsuserid AND zud1.userid = zud.userid),  0, 0, 1)  AS "is\_student",   ed.salary AS "salary", zud.gender AS "gender" FROM zamgeruserdetails zud, users u, userdetails ud, employeedetails ed WHERE zud.userid = u.id AND ud.userid = u.id AND ed.zamgeruserdetailsuserid = zud.userid ORDER BY zud.userid; -- Informacije o predavaču |

|  |
| --- |
| SELECT scheduled AS full\_date, extract(YEAR FROM scheduled) AS year, extract(MONTH FROM scheduled) AS month,  extract(DAY FROM scheduled) AS day, extract(HOUR FROM scheduled) as hour,   to\_char(scheduled, 'MONTH') AS month\_word, to\_char(scheduled, 'DAY') AS day\_word FROM exam UNION ALL SELECT scheduled AS date, extract(YEAR FROM scheduled) AS year, extract(MONTH FROM scheduled) AS month,  extract(DAY FROM scheduled) AS day, extract(HOUR FROM scheduled) as hour,   to\_char(scheduled, 'MONTH') AS month\_word, to\_char(scheduled, 'DAY') AS day\_word FROM class ORDER BY scheduled DESC; -- Vremenska dimenzija |

|  |
| --- |
| SELECT id, title, active, EXTRACT(YEAR FROM "start") AS start\_year FROM academicyear ORDER BY start\_year; -- Informacije o akademskoj godini |

3.2.2 Transformation/Load

Transformacija podataka uključuje pročišćivanje podataka (npr. u prethodnim upitu vezan za vremensku dimenziju izdvojeni su samo podaci vezani za dan/mjesec, koji su bitni za kolonu u novoj tabeli skladišta. U navedenim upitima vrši se jedna vrsta filtriranja podataka, odnsono odabiru se samo određene kolone za učitavanje iz pojedinih tabela fakultetske baze podataka, što je još jedna operacija u procesu transformacije podataka.

Dimenzije i fakti su u relacije 1:n, što znači da je potrebno prvo učitati dimenzije u skladište podataka, pa tek onda se vrši punjenje fakt tabela. Kako je navedeno u poglavlju **3.1**, u ovom sistemu koristit će se Spring *framework* s Hibernate-om. Iz tog razloga je dovoljno u Javi napisati klase (modele) koje će odgovarati tabelama iz skladišta podataka.

Punjenje fakta znači upotrebu već spomenutih modela dimenzija kako bi se mogli definirati *foreign key*-evi za pojedine slogove. Punjenje ovih tabela je realizirano kroz HQL upite spomenute u poglavlju **3.1.1**.

3.3 Komponente vezane za web prezentaciju BI alata

3.4 Komponente vezane za Android prezentaciju BI alata

4. ERD