

**TEHNIČKO VELEUČILIŠTE U ZAGREBU**

**STRUČNI STUDIJ INFORMATIKE**

Emanuel Ljubičić

**INFORMACIJSKI SUSTAV ZA PRAĆENJE  
NASTAVNOG PROCESA U VISOKOM  
OBRAZOVANJU**

Završni rad br. 3290

Zagreb, rujan 2020.

**TEHNIČKO VELEUČILIŠTE U ZAGREBU**

**STRUČNI STUDIJ INFORMATIKE**

Emanuel Ljubičić

JMBAG: 0246075033

**INFORMACIJSKI SUSTAV ZA PRAĆENJE  
NASTAVNOG PROCESA U VISOKOM  
OBRAZOVANJU**

Završni rad br. 3290

Zagreb, rujan 2020.

# ZADATAK

**Emanuel Ljubičić**

**Jakob Gračanin**

redovni student Tehničkog Veleučilišta u Zagrebu

mentor

Smjer: Informatika,

Organizacija i informatizacija ureda

JMBAG: 0246075033

## NASLOV RADA

Informacijski sustav za praćenje nastavnog procesa u visokom obrazovanju

## OPIS ZADATKA

- Teorijski aspekti modeliranja podataka
- Teorijski aspekti Java/Java Swinga
- Teorijski aspekti MySql baze podataka
- Izraditi konceptualni, logički i fizički model podataka (MySql) za navedeni proces
- Izraditi desktop aplikaciju (Java Swing) s pripadajućim interakcijskim sučeljem
- Integracija Java & MySql kroz izradu i opis potrebnih rutina

## **ZAHVALA**

Zahvaljujem se mentoru Jakobu Gračaninu, dipl. ing. - na ukazanom povjerenju i savjetima te svim profesorima Tehničkog Veleučilišta u Zagrebu koji su me poučavali kroz ove tri godine studija.

## **SAŽETAK**

U ovome radu je teorijski definiran pojam informacijskog sustava, modeliranja podataka za informacijski sustav, osnovne značajke MySQL-a i Java programskog jezika. Opisana je izrada primjera informacijskog sustava za praćenje nastavnog procesa u visokom obrazovanju kroz modeliranje podatkovnog modela i razvoj aplikacije s grafičkim sučeljem. Za izradu modela baze podataka je korišten MySQL Workbench aplikacija MySQL RDBMS-s, a za izradu aplikacije razvojno okruženje Apache Netbeans i Java Swing.

# SADRŽAJ

1.	UVOD.....	1
1.1	Informatizacija.....	1
1.2	Informacijski sustav.....	1
1.3	IS za praćenje nastave - Academus.....	2
2.	MODELIRANJE PODATAKA.....	3
2.1	Model podataka .....	3
2.2	Konceptualno modeliranje.....	4
2.3	ER model.....	4
2.3.1	Entitet .....	4
2.3.2	Atributi .....	5
2.3.3	Veze .....	5
2.4	Logičko modeliranje .....	6
2.5	Relacijski model.....	7
2.6	Fizičko modeliranje .....	11
3.	MYSQL BAZA PODATAKA.....	13
3.1	MySQL.....	13
3.2	MySQL Workbench.....	14
3.3	SQL jezik .....	15
4.	KREIRANJE BAZE PODATAKA ACADEMUS .....	16
4.1	EER model .....	16
4.2	EER model Academus baze podataka.....	17
4.3	Pregled kreiranog modela baze podataka.....	19
5.	JAVA PROGRAMSKI JEZIK .....	21
5.1	Java programski jezik .....	21
5.2	Posebnost Java kao objektno orijentirani jezik.....	22
5.3	Java klase .....	22
5.4	Java varijable .....	24

5.5 Java metode.....	25
5.5.1 Metode pristupa i promjene .....	26
5.5.2 Konstruktori .....	27
5.6 Java Swing.....	28
5.6.1 Java Swing spremnici .....	29
5.5 GUI Builder.....	30
6. IZRADA APLIKACIJE ACADEMUS.....	32
6.1 Funkcionalnosti aplikacije.....	32
6.2 Kreiranje Java projekta.....	32
6.3 Kreiranje klase za integraciju s bazom podataka .....	34
6.4 Forma za prijavu .....	35
6.4.1 Komponente forme za prijavu.....	35
6.4.2 Prijava korisnika .....	36
6.5 Glavna forma .....	38
6.6 JTable tablica.....	39
6.7 Forma za odabir kolegija.....	42
6.8 Forma za pregled studenata na kolegiju .....	43
6.9 Forma za pregled i unos ocjena.....	43
6.9.1 Korištenje forme za pregled i unos ocjena .....	43
6.9.2 Metoda za unos ocjena .....	44
6.10 Forma za pregled, unos i brisanje nastave .....	45
6.10.1 Unos nove nastave.....	45
6.10.2 Unos prisutnosti studenta .....	46
6.10.3 Brisanje unosa nastave .....	48
6.11 Forma za pregled dolazaka studenta.....	49
6.12 Pogled na organizaciju projekta.....	49
7. ZAKLJUČAK .....	51

## POPIS KRATICA

KRATICA	OBJAŠNJENJE
RDBMS	engl. <i>Relational Database Management System</i>
JDK	engl. <i>Java Development Kit</i>
JDBC	engl. <i>Java Database Connectivity</i>
ER	engl. <i>Entity - Relationship</i>
UML	engl. <i>Unified Modeling Language</i>
SQL	engl. <i>Structured Query Language</i>
DQL	engl. <i>Data Query Language</i>
DCL	engl. <i>Data Control Language</i>
DML	engl. <i>Data Manipulation Language</i>
DDL	engl. <i>Data Definition Language</i>
EER	engl. <i>Enhanced Entity - Relationship</i>
JVM	engl. <i>Java Virtual Machine</i>
JRE	engl. <i>Java Runtime Environment</i>
JDK	engl. <i>Java Development Kit</i>
AGC	engl. <i>Automatic Garbage Collector</i>
API	engl. <i>Application Programming Interface</i>
JFC	engl. <i>Java Foundation Classes</i>
AWT	engl. <i>Abstract Window Toolkit</i>
WYSIWYG	engl. <i>What You See Is What You Get</i>
GUI	engl. <i>Graphical User Interface</i>
XML	engl. <i>Extensible Markup Language</i>
HSB	engl. <i>Hue, Saturation, Brightness</i>



## POPIS SLIKA

Slika 1 Pregled alata korištenih za kreiranje Acadamus IS .....	2
Slika 2 Faze modeliranja baze podataka.....	3
Slika 3 Atributi entiteta student.....	5
Slika 4 Primjer 1 – 1 veze entiteta .....	6
Slika 5 Primjer 1 – N veze entiteta .....	6
Slika 6 Primjer N – N veze entiteta.....	6
Slika 7 Edgar F. Codd (1923. - 2003.).....	7
Slika 8 Terminologija pojmova relacijskog modela .....	8
Slika 9 Tablica „student“ .....	8
Slika 10 Tablica mjesto .....	9
Slika 11 Tablica student koja je povezana s tablicom mjesto stranim ključem .....	9
Slika 12 Veza 1 - 1 između tablica .....	10
Slika 13 Veza N – 1 između tablica .....	10
Slika 14 Veza N – N između tablica .....	11
Slika 15 Pogled na MySQL Workbench uređivač SQL-a.....	14
Slika 16 Rezultat SQL upita .....	15
Slika 17 Umetanje tablice u EER dijagram.....	16
Slika 18 Uređivanje atributa tablice u EER dijagramu .....	17
Slika 19 EER dijagram Acadamus baze podataka .....	18
Slika 20 Generiranje fizičke sheme iz EER dijagrama u MySQL Workbenchu.....	19
Slika 21 MySQL Workbench „Navigator“ .....	20
Slika 22 Ispis tablica naredbom.....	20
Slika 23 Javna klasa Klasa u paketu Kod.....	23
Slika 24 Pod-klasa Klasa nasljeđuje nad-klasu Main .....	23
Slika 25 Razni tipovi varijabli u klasi.....	24
Slika 26 Preopterećivanje metode.....	25
Slika 27 Poništavanje metode zvukZivotinje u pod-klasama Pas i Macka.....	26
Slika 28 Stvaranje instance klase Main .....	27
Slika 29 Zadani konstruktor.....	27
Slika 30 Preopterećivanje konstruktora .....	28
Slika 31 Hijerarhija klasa Swing komponenti .....	29
Slika 32 Primjer JFrame i JPanel komponenti.....	30

Slika 33 GUI builder .....	31
Slika 34 Paleta komponenti (lijevo), opcije atributa komponente (desno) .....	31
Slika 35 Pogledi na projekt.....	33
Slika 36 Metoda konekcijaBaze .....	35
Slika 37 Metoda zatvaranjeKonekcije.....	35
Slika 38 Izgled forme za prijavu .....	36
Slika 39 Postavljanje ograničenja broja znakova na JTextField .....	36
Slika 40 Metoda dohvatiKorisnika .....	37
Slika 41 Unos vrijednosti u tablicu Prijava kroz metodu unosPrijava .....	38
Slika 42 Glavna forma .....	39
Slika 43 Imenik djelatnika.....	39
Slika 44 Uređivanje zaglavlja tablice .....	40
Slika 45 Metoda za dohvati seta rezultata iz baze podataka.....	41
Slika 46 Pohrana vrijednosti u JTable tablicu .....	41
Slika 47 Metoda događaja tablice .....	42
Slika 48 Prozor za odabir kolegija .....	42
Slika 49 Pregled studenata na kolegiju .....	43
Slika 50 Metoda za unos ocjena .....	44
Slika 51 Forma za pregled, unos i brisanje nastave .....	45
Slika 52 Metoda za unos nastave .....	46
Slika 53 SQL procedura postaviBrPrisutnih .....	47
Slika 54 SQL funkcija vratiBrPrisutnih .....	47
Slika 55 Dolasci na nastavu s opcijom za odabir prisutnosti studenta .....	47
Slika 56 Metoda za brisanje dolazaka .....	48
Slika 57 Metoda za brisanje nastave.....	48
Slika 58 Pregled prisutnosti po nastavi za odabranog studenta .....	49
Slika 59 Projekt Academus .....	50

# 1. UVOD

## 1.1 Informatizacija

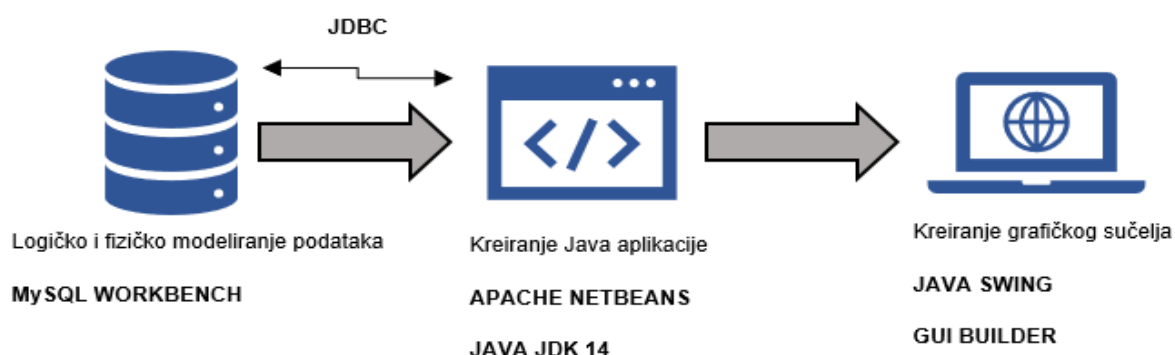
Pojavom računala i razvojem informacijsko komunikacijskih tehnologija došlo je do pojave procesa informatizacije. Informatizacija se odnosi na proces kojim se olakšava protok, spremanje i pristup informacijama, i to implementiranjem računalnih uređaja, a utjecaj informatizacije je vidljiv u svim djelatnostima i ustanovama koje imaju podatak kao resurs. [1] Podatak je u njima neki iskaz, dok je informacija podatak koji primatelju ima neku relevantnu vrijednost stoga vrijednost informacije ovisi o primateljevoj interpretaciji. [4] Shodno informatizaciji poslovanja i generalno društva, mnogi poslovni procesi koji su iziskivali značajne financijske, vremenske i druge resurse su postali unaprijeđeni, pojednostavljeni te pristupačniji kako korisniku tako i rukovoditeljima. Konkretno je olakšana manipulacija i interpretacija podataka koji su bitni za neku organizaciju, ustanovu ili društvo. Manipulacije nad podacima su prikupljanje, obrađivanje, spremanje, pretraživanje i prikazivanje podataka, a sve to se vrši kroz definirane postupke koji su organizirani u informacijske sustave. [2]

## 1.2 Informacijski sustav

Informacijski sustav kao cjelina se sastoji od fizičkog dijela (računala i oprema) – *hardware*, programskog rješenja – *software*, osoba koje koriste sustav – *lifeware*, načina skladištenja podataka tj. baze podataka – *dataware*, komunikacijskih i mrežnih rješenja – *netware* i organizacijskih metoda i postupaka koji objedinjuju sve navedeno – *orgware*. [3] Informacijski sustavi nastaju kao podrška poslovnom sustavu. U ovome radu fokus je usmjeren na informacijski sustav za praćenje nastavnog procesa, i to specifično u ustanovi visokog obrazovanja kao što je fakultet ili visoka škola. U svrhu demonstracije kreiranja informacijskog sustava razvijen je jednostavni informacijski sustav za praćenje nastave u visokom obrazovanju koji nosi naziv *Academus*. Ova vrsta informacijskog sustava služi obrazovnim djelatnicima za praćenje nastavnog procesa odnosno bilježenje raznih podataka o studentima za vrijeme trajanja akademske godine (ocjene, prisustvo nastave). Podaci se kasnije interpretiraju u potrebne informacije te se na temelju njih pojedini student vrednuje.

### 1.3 IS za praćenje nastave - Academus

Informacijski sustav Academus se sastoji od različitih segmenata. *Hardware* dio se odnosi na računalo koje će podržavati *software* (aplikacijski) dio sustava. To se zapravo odnosi na uređaj u vlasništvu ustanove ili privatnog korisnika koji će podržavati funkcioniranje sustava ili će putem njega sustav biti korišten od strane korisnika. *Software* je najkompleksniji dio sustava i sučelje između ostalih komponenti, on će detaljno biti obrađen u nastavku kao i *dataware* koji će biti usko povezan sa *software* segmentom. *Netware* dio se odnosi na komunikaciju putem mreže, a može služiti i za povezivanje baze podataka s aplikacijom preko poslužitelja. Naposljetku *lifeware* kao segment korisnika ovog sustava je moguće podijeliti na djelatnike ustanove koji imaju administratorska prava za upravljanje sustavom i one koji nemaju, odnosno one koji samo koriste sustav za dohvat ili unos limitiranog raspona podataka. Korisnici koji imaju administratorska prava ih koriste na način koji je uređen *orgware* segmentom odnosno procedurama i pravilima upravljanja sustavom. U sklopu ovoga rada, razvoj i implementacija *dataware* i *software* segmenta razvijenog informacijskog sustava je analizirana u daljnjim poglavljima. Baza podataka koja je kreirana za potrebe IS-a se zasniva na MySQL RDBMS-u, a aplikacija na Java Swing tehnologiji za razvoj Windows desktop aplikacija s grafičkim sučeljem, uz razvoj putem GUI builder alata kojeg nudi NetBeans integrirana razvojna okolina. Korištena verzija Java razvojnog alata (JDK) je 14. Modeliranje baze podataka je izvršeno kroz MySQL Workbench aplikaciju, a konekcija baze podataka i aplikacije pomoću JDBC upravljačkog programa. Pregled uloge alata u razvojnem procesu je vidljiv na sljedećoj slici (slika 1).

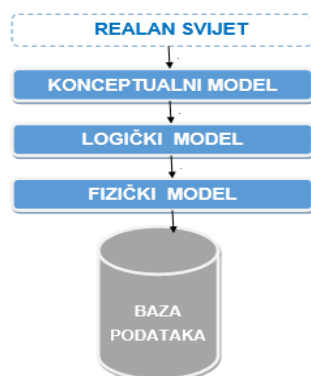


Slika 1 Pregled alata korištenih za kreiranje Academus IS

## 2. MODELIRANJE PODATAKA

### 2.1 Model podataka

Model podataka je apstraktni model čija je primarna svrha deskriptivno prikazati strukturu podataka kojom se koristi poslovni proces, i to na što učinkovitiji način. Pojam modela podataka se može odnositi na teoretski korištenja i strukturiranja podataka ili na rezultat primjene tog teoretskog opisa u svrhu stvaranja modela za neki informacijski sustav. Model podataka definira elemente podataka, njihovo značenje i veze kojima su podaci povezani. Elementi od kojih se sastoji model podataka su sama struktura podataka, operacije nad podacima i ograničenja podataka. Operacije nad podacima su postupci nad strukturom modela koji su preslika postupaka iz realnog sustava. Ograničenja podataka definiraju zadani opis stanja realnog sustava. To je skup pravila koja odvajaju nedopušteno stanje realnog sustava od dopuštenog. [12] Modeli podataka se primjenjuju ponajviše za razvoj informacijskih sustava, tj. baza podataka koje služe kao podrška tom sustavu. Postoje tri načina modeliranja podataka koji rezultiraju s tri različita modela podataka, a to su: konceptualni, fizički i logički. Konceptualni model polazi od specifikacija informacijskih zahtjeva, odnosno zahtjeva strukture podataka i načina korištenja tih podataka. On je potreban za shvaćanje podataka sustava, cjelovit je, konzistentan i ne sadrži redundanciju. Logički model nastaje daljnjom razradom konceptualnog modela, a fizički model je stvarna fizička organizacija podataka koja nastaje na temelju logičkog modela podataka. Realizacijom fizičkom modela nastaje shema baze podataka. Na slici 2 je prikazan proces nastanka baze podataka kroz korake modeliranja.



*Slika 2 Faze modeliranja baze podataka*

Izrada modela podataka je kompleksan postupak u kojemu je vrlo bitno dobro poznavati procese same ustanove za koju se podaci modeliraju, iz tog razloga je bitna dobra komunikacija i suradnja projektanta modela podataka i rukovoditelja izrade informacijskog sustava.

## **2.2 Konceptualno modeliranje**

U izradi baze podataka, prva faza je konceptualno modeliranje. Rezultat konceptualnog modeliranja je konceptualna shema koja se sastoji od entiteta, atributa i veza. Prikaz konceptualne sheme je lako razumljiv, neformalan i jednostavan, čak i za osobe van informatičke struke. [6] Zbog svoje lake razumljivosti, konceptualna shema služi kao sredstvo komunikacije između rukovoditelja projekta informacijskog sustava, projektanta modela podataka i korisnika. Komunikacija rukovoditelja i projektanta je posebno bitna projektantima dok izrađuju konceptualni model podataka zbog potrebe za dobrim razumijevanjem svih informacija i procesa. Konceptualni model podataka je opis podataka koji je cjelovit, konzistentan i bez redundancije. Da bi konceptualni model bio kvalitetan, potrebno je da projektant dobije informacije koje su mu potrebne te da ih razumije. Ukoliko je provedena nedovoljna analiza zahtjeva, sama baza podataka kao konačni cilj modeliranja neće biti odgovarajuće kvalitete, pa je potrebno u ovom početnom koraku napraviti što više upita prema korisnicima i rukovoditelju. Riječ je o zahtjevnom ali izrazito bitnom poslu u kojemu nije preporučljivo štedjeti na vremenu. Nakon prikupljanja podataka i određivanja veza između njih, podaci se obično konceptualno modeliraju putem metoda kao što je ER ili UML modeliranje.

## **2.3 ER model**

U ER modelu (model entiteti – veze) objekti iz realnog svijeta su prikazani s pojmovima entiteta, atributa i veza. Entiteti se odnose na realne ili apstraktne objekte i događaje, atributi na njihova zajednička svojstva, a veze na odnose između samih entiteta.

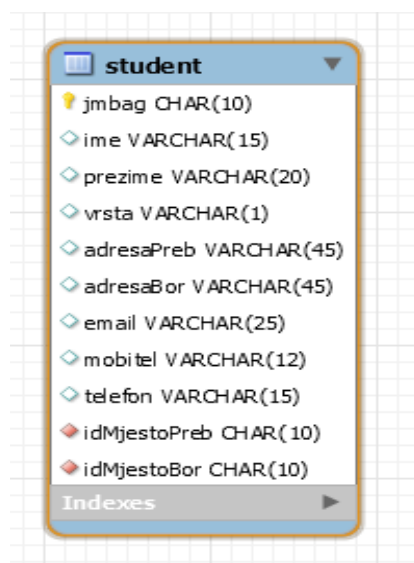
### **2.3.1 Entitet**

Entitet je neki objekt ili skup objekata iz realnog svijeta o kojemu se spremaju podaci u bazu podataka. Entitet može sadržavati objekte kao što su živa bića (npr. djelatnik), mjesta (npr. grad) ili neki drugi objekti koji su potrebni bazi podataka (npr. knjiga). Razlikujemo entitete koji su jaki i slabi. Za razliku od jakih, slabi entiteti su ovisni o

jakom entitetu kako bi se mogli jednoznačno identificirati. Entiteti se grafički prikazuju pravokutnikom.

### 2.3.2 Atributi

Atributi su podaci koji opisuju entitet. Svaki entitet ima skup atributa koji ga opisuju, a oni mogu biti identifikacijski i deskriptivni (opisni). Identifikacijski atributi jedinstveno određuju instancu entiteta tako da ne mogu postojati dvije instance istog entiteta s istom vrijednošću identifikacijskog atributa (entitetski integritet). Identifikacijski atributi se nazivaju još i primarni ključevi. [6] Moguće je da postoji više identifikacijskih atributa u entitetu, tada se to naziva kompozitnim primarnim ključem, a takav entitet je slab te ovisi o jakim entitetima. Opisni atributi služe za širi opis instance entiteta. Primjer entiteta student s njegovim atributima je prikazan na slici 3. Atribut jmbag je identifikacijski jer jednoznačno određuje o kojem studentu se radi, dok su svi ostali atributi opisni jer opisuju studenta ali ga nužno i ne identificiraju.

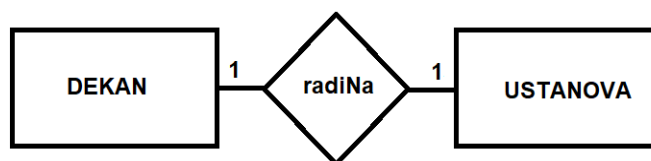


Slika 3 Atributi entiteta student

### 2.3.3 Veze

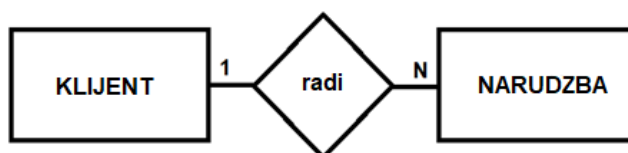
Entiteti se međusobno povezuju vezama i to upravo preko identifikacijskih atributa. Veza u zavisnosti od broja entiteta koje povezuje može biti unarna, binarna, ternarna itd. Kardinalnost veze među entitetima se odnosi na ograničenje preslikavanja pojedinačnih entiteta koje veza povezuje. Može biti „jedan“ (engl. one) ili „više“ (engl. many). Pri tome se uobičajeno kao oznaka za „jedan“ koristi „1“, a kao oznaku za „više“ koristi se „N“. Tako su kombinacije povezanosti entiteta:

- jedan na jedan ( $1 - 1$ ), primjer: dekan vodi samo jednu ustanovu i ustanova ima samo jednog dekana



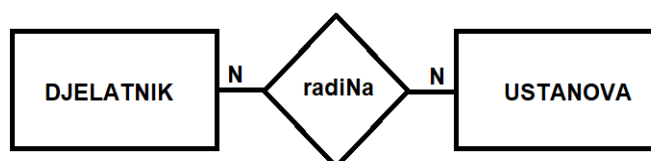
*Slika 4 Primjer 1 – 1 veze entiteta*

- jedan na više ( $1 - N$ ) / više na jedan ( $N - 1$ ), primjer: klijent može raditi više narudžbi i više narudžbi mogu imati jednog klijenta



*Slika 5 Primjer 1 – N veze entiteta*

- više na više ( $N - N$ ), primjer: djelatnik može raditi na više ustanova i ustanova ima više djelatnika [6]



*Slika 6 Primjer N – N veze entiteta*

## 2.4 Logičko modeliranje

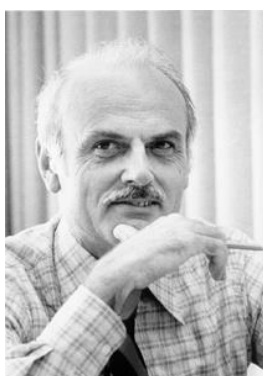
Logički model može biti relacijski, objektni, hijerarhijski, mrežni ili datotečni, no danas je u praksi najčešće korišten relacijski model. Objektno orijentirani model se još uvijek teorijski razvija te se njegova ozbiljnija primjena tek očekuje, dok se ostali modeli ne koriste zbog nadmoćnosti relacijskog modela u brojnim pogledima. U korištenju ovog modela, faza logičkog modeliranja se sastoji od konverzije kreiranog ER modela u relacijski model. Entiteti se prikazuju relacijama, a atributi entiteta postaju i atributi relacije. Primarni ključ entiteta postaje i primarnim ključem relacije. Rezultat logičkog



modeliranja je logička shema koja je sastavljena od relacija koje se nazivaju i tablicama. Na kraju logičkog oblikovanja vrši se i normalizacija, gdje se primjenom posebnih pravila nastoji optimizirati logička struktura samih relacija. [6]

## 2.5 Relacijski model

Relacijski model je teorijski zasnovao engleski znanstvenik Edgar F. Codd (prikazan na slici 7) 60-ih godina prošloga stoljeća. Ovaj model je zasnovan na ideji da se cjelokupni skup podataka koji je bitan za bazu podataka strukturira u pravokutne tablice tj. relacije. Baza podataka koja je organizirana u skladu s uvjetima relacijskog modela naziva se relacijska baza podataka.



*Slika 7 Edgar F. Codd (1923. - 2003.)*

Sama svrha relacijskog modela je da omogući metodu za dohvat podataka iz baze koja je deklarativna, a da se upravljanje strukturama podataka, pohrana i dohvat procedura i izvođenje upita povjeri posebnoj aplikaciji za upravljanje relacijske baze podataka (RDBMS). Relacijski model zahtijeva da je baza podataka strukturirana od skupa tablica. Svaka tablica ima svoje ime po kojem se razlikuje od ostalih u bazi podataka. Tablice su dvodimenzionalne i sastoje se od redova i stupaca. Svaki stupac tablice je određen svojim imenom i sadrži vrijednost jednog atributa. Vrijednosti unutar jednog stupca su podaci istog tipa, a ponekad se tolerira i da vrijednost atributa nije definirana ili upisana. Broj atributa koji tablica sadrži određuje stupanj relacije, a skup vrijednosti jednog atributa se naziva domenom atributa. Redak tablice se naziva još i n-torka i on predstavlja instancu entiteta ili bilježi vezu između dvije instance entiteta. U jednoj tablici ne mogu biti dvije jednake n-torke zbog postojanja identifikacijskog atributa, a broj n-torki određuje kardinalnost relacije. Na slici 8 je vidljivo korištenje

različite terminologije za iste objekte relacijskog modela u različitim tehnološkim domenama.

DBMS	RELACIJSKA TEORIJA	PROGRAMSKI JEZICI
tablica	relacija	datoteka
redak	N - torka	zapis
stupac	atribut	polje

*Slika 8 Terminologija pojmova relacijskog modela*

Kao i kod entiteta i kod relacija se identifikacijski atribut naziva primarni ključ. Primarni ključ služi za povezivanje s drugim tablicama preko vanjskih ključeva. Primjerice na slici 9 je moguće vidjeti primjer jedne relacije u kojem se vrijednosti atributa mjesto i atributa pbr ponavljaju.

STUDENT					
<u>JMBAG</u>	IME	PREZIME	OCJENA	MJESTO	PBR
0246075033	Janko	Janković	5	Zagreb-Centar	10000
0246075034	Ivan	Ivanović	4	Zagreb-Centar	10000
0246075035	Petar	Petrović	3	Split	21000
0246075036	Marin	Marinković	2	Zagreb-Dubrava	10040
0246075038	Filip	Filipović	4	Rijeka	51000

*Slika 9 Tablica „student“*

U ovome slučaju je moguće kreirati novu relaciju mjesto koja sadržava identifikacijski atribut kao primarni ključ te atribut naziva mjesta i poštanskog broja mjesta kao opisne attribute, kao što je prikazano na slici 10.

MJESTO		
ID	NAZIV	PBR
1	Zagreb - Centar	10000
2	Zagreb - Dubrava	10040
3	Split	21000
4	Rijeka	51000
5	Osijek	31000
6	Karlovac	47000

*Slika 10 Tablica mjesto*

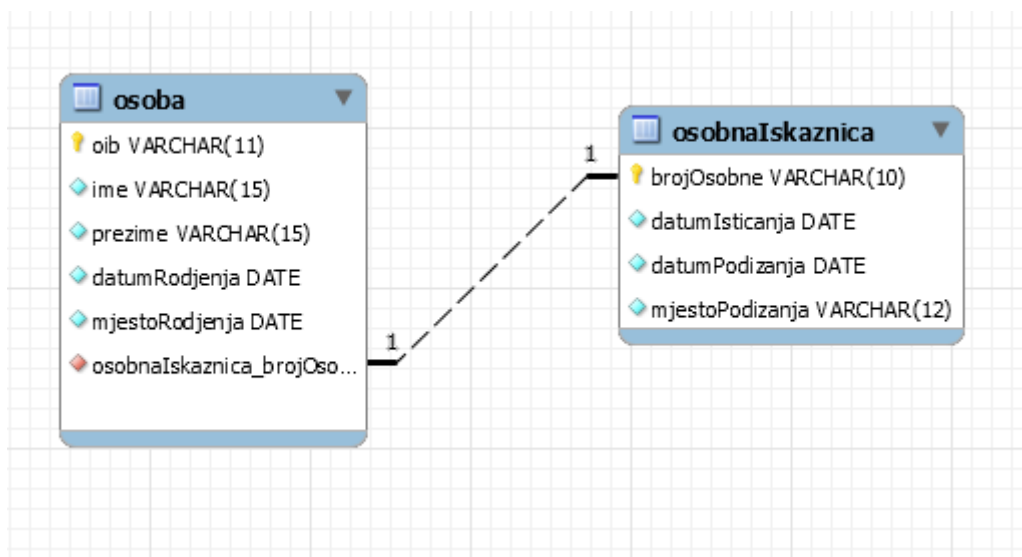
U relaciju student se zatim umjesto opisnih atributa naziva mjesta i poštanskog broja, ubacuje primarni ključ relacije mjesto kao strani ključ relacije student, što je prikazano i na slici 11.

STUDENT			
ID	IME	PREZIME	ID_MJESTO
0246075033	Janko	Janković	1
0246075034	Ivan	Ivanović	1
0246075035	Petar	Petrović	3
0246075036	Marin	Marinković	2
0246075038	Filip	Filipović	4

*Slika 11 Tablica student koja je povezana s tablicom mjesto stranim ključem*

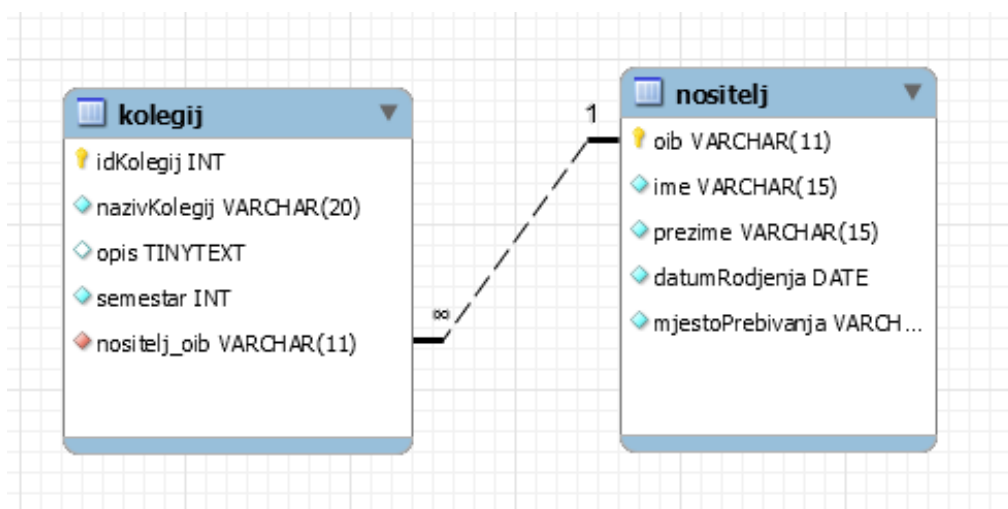
Iz navedenog primjera je vidljivo kako pri pretvorbi u relacijski model može doći do dodavanja broja relacija u odnosu na broj definiranih entiteta. Veze između relacija se ostvaruju preko ključeva sa svrhom sprječavanja redundancije i to na isti način kao kod entiteta i mogu biti:

- jedan na jedan (1 – 1), primjer: jedna osoba može posjedovati samo jednu osobnu iskaznicu



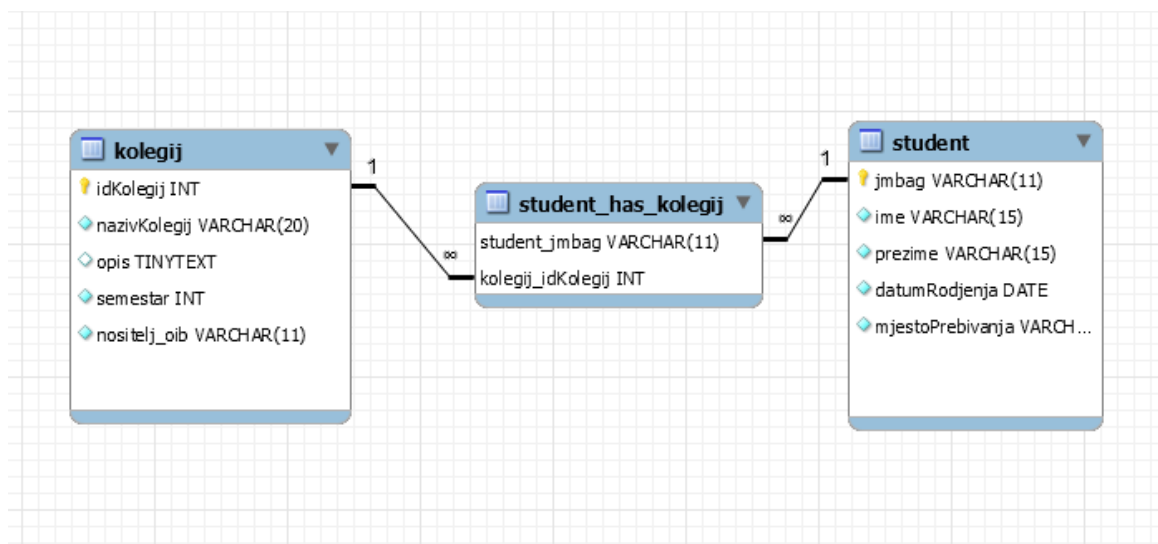
Slika 12 Veza 1 - 1 između tablica

- jedan na više (1 – N) / više na jedan (N – 1), primjer: predavač može biti nositelj na više kolegija, ali kolegij može imati samo jednog nositelja



Slika 13 Veza N – 1 između tablica

- više na više (N – N), primjer: student može imati više upisanih kolegija i kolegij može imati više studenata koji ga pohađaju



*Slika 14 Veza N – N između tablica*

Sa slike 14 je vidljivo da se prilikom veze N – N uvodi još jedna tablica student\_has\_kolegij (engl. has = ima) koja sadrži identifikacijske attribute tablice kolegij i student. Ova dva atributa tvore kompozitni primarni ključ tablice student\_has\_kolegij što ovu tablicu čini ovisnom o tome da postoje vrijednosti primarnog ključa u tablicama kolegij i student. Tablice kolegij i student su prema tome jake tablice, dok je tablica student\_has\_kolegij slaba tablica koja ovisi o jakim tablicama.

## 2.6 Fizičko modeliranje

Fizičko modeliranje je treća od faza u modeliranju baze podataka. Ova faza se izvršava nakon definiranja relacijskog modela koji je nastao iz logičkog modela. Glavni rezultat fizičkog modeliranja je fizička shema cijele baze. Fizička shema je niz SQL naredbi kojima se relacije iz logičke sheme realiziraju kao fizičke tablice. Tako je i fizički redak u tablici realizacija fizičkog retka. Glavni cilj koji fizičko modeliranje treba postići je kreacija takve fizičke organizacije podataka da je pristup recima tablice što brži. Pritom se dodaju pomoćne strukture i mehanizmi za postizanje traženih performansi te očuvanje integriteta i sigurnosti podataka. O nekim aktivnostima koje se obavljaju tijekom fizičkog modeliranja brine sustav za upravljanje fizičkim prostorom, a o nekima administrator same baze podataka. Navedene aktivnosti su:

- modeliranje strukture retka – za modeliranje strukture se koristi neka od tehnika formatiranja podataka ( pozicija polja fiksne duljine, pozicija polje varijabilne duljine: duljinom prethodnih polja, indeksima, oznakama), za smanjivanje

prostora za smještaj koriste se tehnike kompresije (korištenje kratica, izbacivanje nula, zamjena čestih sekvenci)

- segmentiranje i grupiranje redaka – segmentiranje je smještanje različitih dijelova dugih redaka na različitim fizičkim lokacijama, a grupiranje je smještanje redaka koji se zajedno obrađuju u grupe
- izbor metode pristupa recima – recima se pristupa operacijama unosa, čitanja, brisanja i promjene, a pristup može biti serijski (sekvencijalni) ili direktni
- optimizacija pristupnih puteva – ostvarivanje što bolje distribucije podataka [13]

## 3. MYSQL BAZA PODATAKA

### 3.1 MySQL

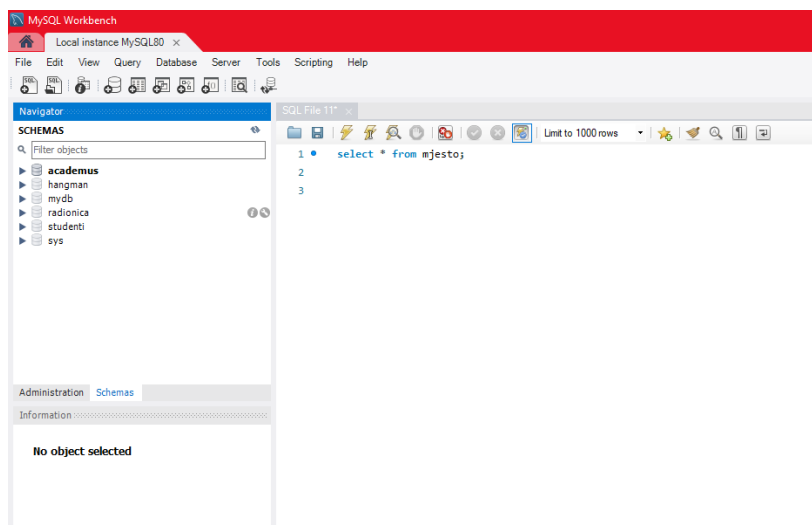
MySQL je upravljački sustav za relacijsku bazu podataka koji je otvorenog (engl. *open source*) koda. Upravljački sustav za relacijsku bazu podataka, poznat i po kratici RDBMS je *software* koji omogućuje korisniku da kreira, održava i kontrolira relacijsku bazu podataka. RDBMS kao što je MySQL surađuje s operativnim sustavom računala kako bi implementirao relacijski model podataka u pohranu računala, rukovao korisničkim računima, omogućio mrežnu povezivost baze i kreiranje sigurnosnih pohrana. Danas je MySQL korišten ponajviše u razvoju web aplikacija i koriste ga mnoge uspješne IT kompanije. Napisan je u C / C++ programskom jeziku i odlikuje se svojom brzinom, pouzdanošću, multikorisničkim načinom rada i višedretvenošću. Zbog svoje višedretvenosti je vrlo skalabilan i ima mogućnost obrade i do 50 milijuna redova podataka. Osim toga, pouzdan je u rukovanju memorijom te se vrlo rijetko dešavaju „curenja“ memorije (engl. *memory leak*). Sigurnost u MySQL-u je velika jer se koristi čvrsti sigurnosni sloj oko povjerljivih podataka, a lozinke su enkriptirane. Funkcionira u klijent / poslužitelj arhitekturi u kojoj veliki broj klijenata može imati interakciju s bazom podataka putem poslužitelja. Podržan je na Windows i iOS operacijskim sustavima ali i na mnogim varijacijama Unix-a. Osim tehničkih prednosti, MySQL ima bogatu i lako dostupnu dokumentaciju. Od funkcionalnosti podržava povratak transakcija (engl. *rollback*), potvrđivanje transakcija (engl. *commit*) i oporavak od rušenja (engl. *crash recovery*) te pogleda, okidače i spremljene rutine. Kvaliteta MySQL-a je priznata od strane portala DB-Engines DBMS, kada je proglašen kao RDBMS godine za 2019. godinu. Trenutna verzija RDBMS-a je 8.0, a razvojni tim redovito izbacuje zakrpe, nadogradnje i poboljšava funkcionalnosti. MySQL je moguće koristiti kroz velik broj aplikacija koje služe kao grafička sučelja koja omogućuju da korisnik ima interakciju s bazom podataka putem grafičkih elemenata umjesto komandne linije. Službeno MySQL grafičko sučelje se zove MySQL Workbench, no zbog svog otvorenog koda postoji i pregršt aplikacija koje su neslužbene odnosno razvijene od treće strane. Neki od neslužbenih grafičkih sučelja za MySQL koji se popularno koriste su phpMyAdmin, SQLyog i HeidiSQL.

## 3.2 MySQL Workbench

MySQL Workbench je aplikacija grafičkog sučelja za MySQL poslužitelje i baze podataka. Glavne funkcionalnosti ove aplikacije su:

- SQL razvoj – omogućuje kreiranje i održavanje konekcija s poslužiteljem baze podataka s opcijama konfiguracije parametara konekcije. Omogućuje izvršavanje SQL naredbi prema povezanom poslužitelju baze podataka kroz uređivač SQL-a.
- Modeliranje podataka – omogućuje grafičko kreiranje sheme baze podataka i dvosmjerno konvertiranje između sheme i baze podataka. Pruža mogućnost uređivanja tablica preko uređivača tablica (engl. *table editor*) koji na jednostavan način omogućuje uređivanje tablica, stupaca, indeksa, okidača, rutina, pogleda i dr.
- Administracija poslužitelja – omogućuje administriranje instanci MySQL poslužitelja administriranjem korisnika, izvođenjem sigurnosnih kopija i oporavaka, uvidom u revizijske podatke, pregledom baze podataka i nadziranjem performansi MySQL poslužitelja.
- Migracije podataka – omogućuje migriranje tablica i ostalih RDBMS objekata iz raznih baza podataka (Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostgreSQL itd.) u MySQL.

Izgled prozora MySQL Workbench s otvorenim uređivačem SQL-a je vidljiv na sljedećoj slici (slika 15).



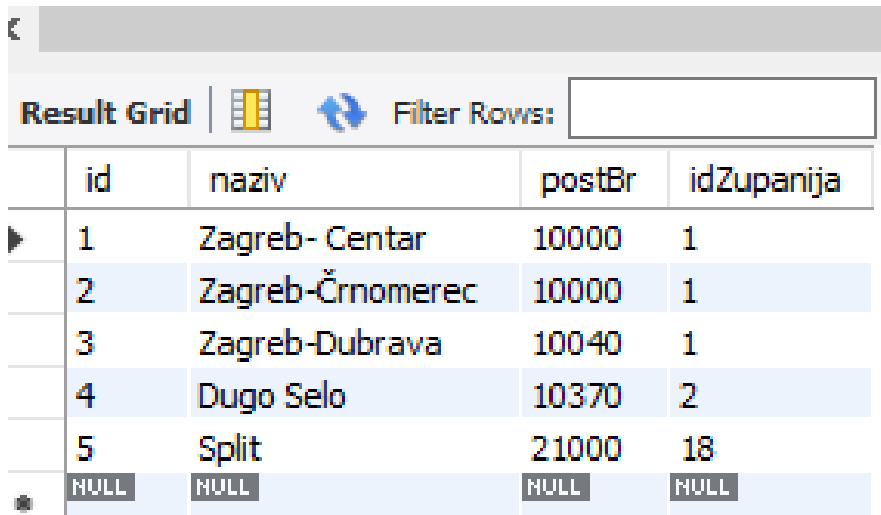
Slika 15 Pogled na MySQL Workbench uređivač SQL-a



### 3.3 SQL jezik

MySQL za radnje nad relacijskom bazom podataka koristi SQL jezik. SQL (engl. *Structured query language*) se sastoji od mnoštvo naredbi koje dijelimo u nekoliko skupina. Prva skupina je DQL (engl. *Data query language*) kojemu je svrha dobivanje rezultata upita prema bazi (primjer je naredba SELECT). Druga skupina je DCL (engl. *Dana control language*) koja služi za autorizaciju i davanje prava nad bazom podataka (primjer su naredbe GRANT i REVOKE), zatim DML (engl. *Data manipulation language*) koji služi za manipulaciju nad podacima (primjer su naredbe INSERT, UPDATE i DELETE) i naposljetku DDL (engl. *Data definition language*) koji služi za kreiranje i modificiranje objekata i strukture baze podataka (primjer su naredbe CREATE, ALTER i DROP). SQL je jezik koji ignorira razliku velikih i malih slova, a na kraju naredbe obavezno dolazi simbol „točka-zarez“. [11] Na slici 16 je prikazana tablica rezultata SQL upita koji je od baze podataka dohvatio sve vrijednosti stupaca u tablici mjesto.

```
5 • select * from mjesto;
```



The screenshot shows a database interface with a query editor at the top containing the SQL command `select * from mjesto;`. Below the editor is a toolbar with a 'Result Grid' button, a grid icon, a refresh icon, and a 'Filter Rows:' input field. The main area displays a table with the following data:

	id	naziv	postBr	idZupanija
▶	1	Zagreb- Centar	10000	1
	2	Zagreb-Črnomerec	10000	1
	3	Zagreb-Dubrava	10040	1
	4	Dugo Selo	10370	2
	5	Split	21000	18
●	NULL	NULL	NULL	NULL

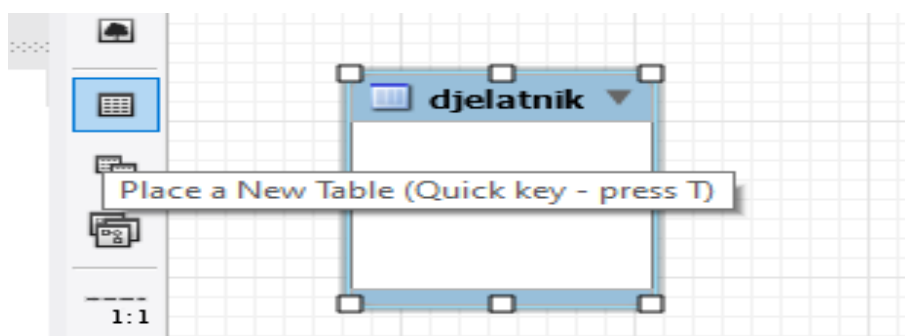
Slika 16 Rezultat SQL upita

## 4. KREIRANJE BAZE PODATAKA ACADEMUS

Za kreiranje baze podataka poželjno je ali ne i nužno imati konceptualni model. Ukoliko je razvijen logički model, moguće je isti automatski pretvoriti u relacijski. Nakon toga potrebno je postaviti dodatna ograničenja, postaviti automatsku inkrementaciju primarnih ključeva gdje je moguće, postaviti attribute da ne mogu primiti NULL vrijednost ako to zahtijevaju itd. Za izradu Academus fizičkog modela baze podataka korištena je značajka EER dijagrama unutar MySQL Workbench alata.

### 4.1 EER model

EER (engl. *Enhanced ER*) dijagram omogućuje brzo i jednostavno stvaranje naprednog ER modela. EER model je osnova za kasnije generiranje fizičkog modela podataka u MySQL Workbenchu. Sami grafički prikaz modela je moguće izvesti u obliku slike ili PDF datoteke i koristan je korisniku i projektantu za pregled strukture baze podataka. Izrada EER dijagrama se pokreće s glavnog izbornika alata MySQL Workbench odabirom stavke *file*, pa *new model*. Prilikom kreiranja ovog modela, prvi korak je definiranje koje tablice baze podataka su potrebne na temelju konceptualnih entiteta. Ovaj postupak se vrši pritiskom na ikonu tablice pa pritiskom na površinu za stavljanje tablice (slika 17).



Slika 17 Umetanje tablice u EER dijagram

Tablici EER dijagrama je moguće zadati naziv i attribute (stupce) koje će sadržavati fizička tablica. Atributima je moguće odrediti razne postavke kao što su tip podatka, primanje NULL vrijednosti, započinjanje s vodećom nulom, unikatnost itd. Isto tako omogućeno je lako povezivanje entiteta vezama i kreiranje ključeva (slika 18).

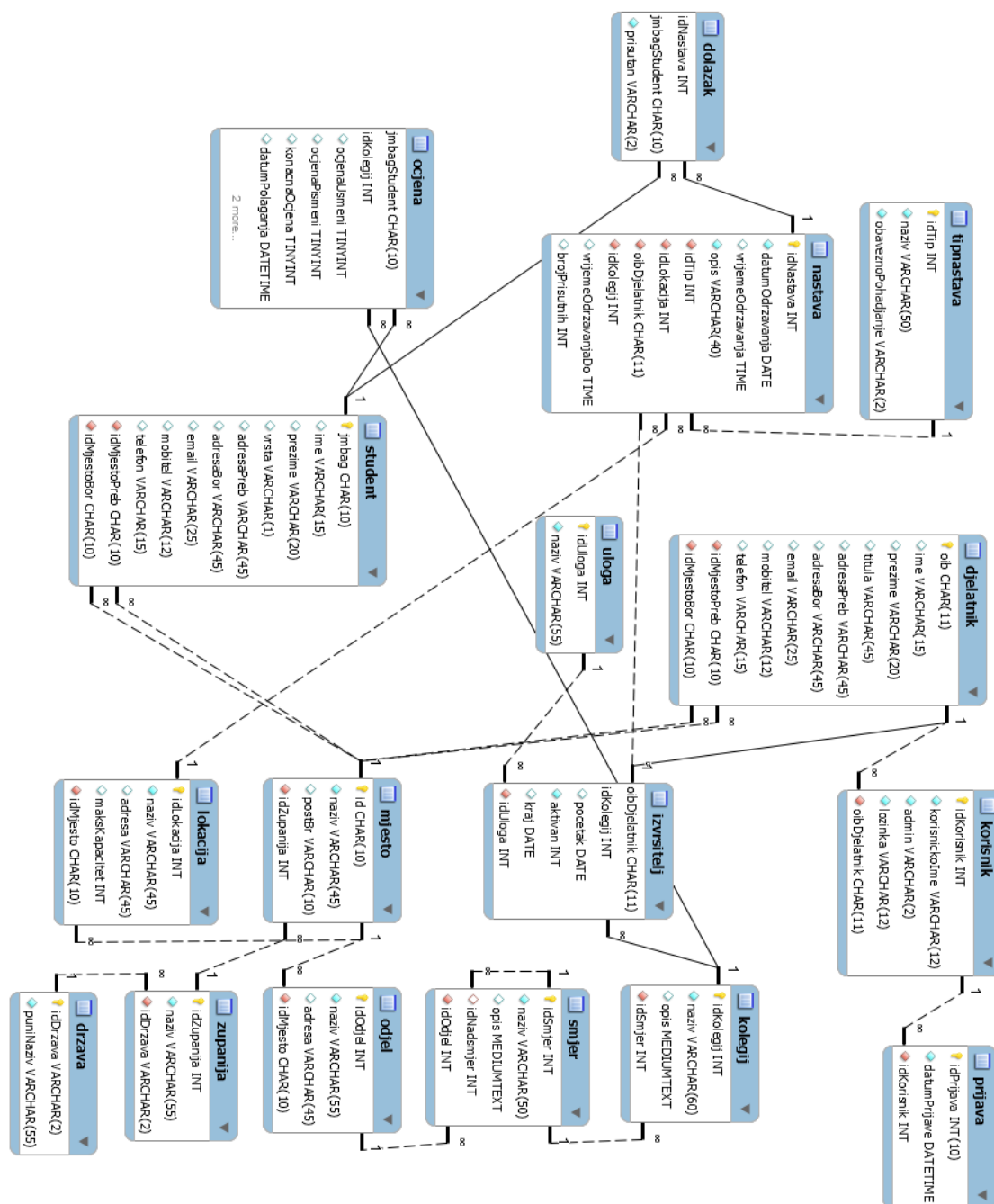
nastava - Table										
Table Name: nastava										
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idNastava	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
datumOdrzavanja	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
vrijemeOdrzavanja	TIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
opis	VARCHAR(40)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Slika 18 Uređivanje atributa tablice u EER dijagramu

Nakon definiranja tablica, tablice se povezuju pritiskom na vrstu željene veze te pritiskom na odredišnu i izvorišnu tablicu veze. Alat zatim grafički povezuje tablice te uspostavlja vezu automatski generirajući potrebne ključeve. Same veze koje je moguće odabrati osim što su različite po kardinalitetu, mogu biti i identificirajuće ili neidentificirajuće. Neidentificirajuće veze su u EER dijagramu grafički prikazane crtkanim linijama, a označavaju veze između tablica u kojoj slaba tablica sadrži primarni ključ jake tablice ali u obliku stranog ključa. Ova veza može biti obavezna ili opcionalna. Ako je obavezna tada vrijednost stranog ključa ne smije biti NULL vrijednost. S druge strane, identificirajuće veze su grafički prikazane punim linijama, a označavaju veze između tablica u kojoj slaba tablica sadrži primarni ključ jake tablice kao dio svog primarnog ključa (kompozitni primarni ključ). U navedenom slučaju vrijednost tog atributa slabe tablice mora postojati kao vrijednost atributa jake tablice i slaba tablica ne može postojati bez jake tablice.

## 4.2 EER model Academus baze podataka

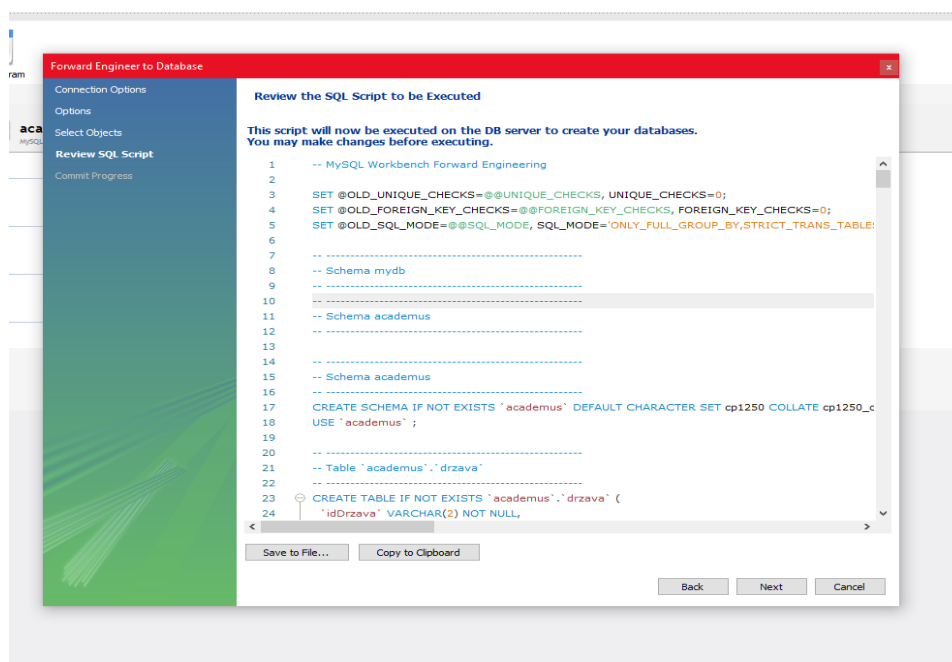
U razvoju EER dijagrama za Academus bazu podataka prvo su postavljene tablice, određeni njihovi nazivi, atributi, te vrste podataka koje će ti atributi nositi. Nakon definiranja atributa, postavljene su opcije atributa (primarni ključevi, jedinstveni atributi, atributi koji ne smiju biti NULL). Na kraju su tablice EER dijagrama povezane i odgovarajućim vezama. Pregled EER dijagrama za Academus bazu podataka je vidljiv na sljedećoj slici (slika 19).



Slika 19 EER dijagram Academus baze podataka

MySQL Workbench alat kreira odgovarajuću fizičku shemu baze podataka iz EER dijagrama putem *forward engineer* opcije. Putem ove opcije se za zadani dijagram automatski generiraju SQL - DDL naredbe za kreiranje fizičkog modela podataka. Osim naredbi za stvaranje fizičke strukture, dodaju se i razne naredbe za optimizaciju mehanizama baze podataka. Ovaj način modeliranja je pogodan zbog svoje

jednostavnosti i detaljnih opcija koje alat nudi. Osim toga, manualno pisanje svih SQL naredbi za kreiranje fizičke sheme bi zahtijevalo znatno više vremena. Kreiranje fizičkog modela iz dijagrama se pokreće iz menija Workbench alata pritiskom na *database*, pa otvaranjem padajućeg menija na stavku *forward engineer*. Otvaranjem prozora potrebno je prvo odrediti konekciju s bazom podataka (adresa, korisničko ime, lozinka), potom opcije same kreacije, koji objekti će se generirati u skripti, koji sustav prikaza znakova će se koristiti u bazi, a zatim je moguće vidjeti i samu skriptu koja će se izvršiti. Prozor za pregled generirane skripte je vidljiv na slici 20.

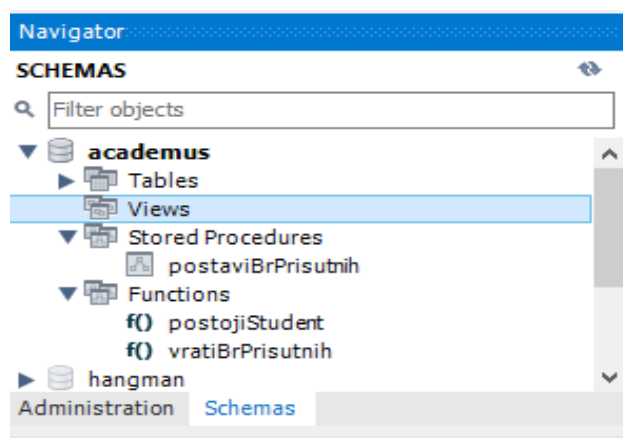


Slika 20 Generiranje fizičke sheme iz EER dijagrama u MySQL Workbenchu

### 4.3 Pregled kreiranog modela baze podataka

Po završetku kreiranja fizičke sheme, baza podataka je spremna za korištenje i integraciju s odgovarajućom aplikacijom. Kroz MySQL Workbench u *navigator* okviru (slika 21) je moguće odabrati fizičku shemu baze podataka te u izborniku ispod nje vidjeti prikaz tablica koje baza podataka sadrži, kao i pohranjenih rutina (engl. *stored procedures*) i pogleda (engl. *views*). Pohranjene rutine su funkcije ili procedure, a odnose se na definirani niz naredbi koje je moguće pozvati nad nekim parametrima ili i bez ulaznih parametara. Funkcije u bazi podataka vraćaju uvijek jedan rezultat, dok procedure mogu vraćati i više rezultata ili uopće ne vraćati rezultat. Pogledi su privremene tablice koje su definirane rezultatom SQL upita nad jednom ili više tablica.

Podaci u pogledu nisu duplicirani već se dinamički izvršavaju kod izvođenja upita. Najčešće se koriste za analiziranje podataka bez narušavanja strukture i normalizacije baze podataka.



Slika 21 MySQL Workbench „Navigator“

Osim kroz *navigator* okvir u MySQL Workbenchu, popis kreiranih baza podataka je moguće saznati i kroz izvođenje SQL naredbe. Da bi unosili SQL naredbe potrebno je na alatnoj traci MySQL Workbencha pritisnuti ikonu za novi prostor za pisanje upita ili odabrati iz izbornika *file* pa *new query tab*. Podatke o kreiranim bazama saznajemo s SQL naredbom `SHOW DATABASES`. Naredbom `SHOW TABLES` dobivamo podatke o tablicama koje se nalaze u bazi podataka, a naredbom `SHOW FULL TABLES` dobivamo u dodatnom stupcu i podatak je li tablica uobičajena ili je riječ o pogledu. Rezultat naredbe `SHOW FULL TABLES` je prikazan na slici 22.

Result Grid			Filter Rows:
	Tables_in_academus	Table_type	
▶	djelatnik	BASE TABLE	
	dolazak	BASE TABLE	
	drzava	BASE TABLE	
	izvrsitelj	BASE TABLE	
	kolegij	BASE TABLE	
	korisnik	BASE TABLE	
	lokacija	BASE TABLE	
	mjesto	BASE TABLE	

Slika 22 Ispis tablica naredbom

## 5. JAVA PROGRAMSKI JEZIK

Aplikacija je sučelje kroz koje korisnik informacijskog sustava upravlja, obrađuje i prikazuje rezultate obrade procesa informacijskog sustava. Osim što je bitno da aplikacija informacijskog sustava bude funkcionalna i točna u postupcima nad podacima ustanove, bitno je i da korisniku bude intuitivna i pristupačna. Naime, informacijski sustav za praćenje nastave je sustav koji se koristi gotovo svakodnevno za vrijeme trajanja akademske godine i zato je bitno omogućiti da se njegove funkcije izvršavaju u što manje koraka, stoga je bitan i sami raspored grafičkog sučelja aplikacije. Iako je aplikaciju moguće razviti u bilo kojem programskom jeziku, potrebno je imati na umu koliko je taj jezik zastupljen ili aktualan kako se ne bi dogodilo da kroz izgledno vrijeme aplikacija koja podržava informacijski sustav ostane bez odgovarajuće stručne i tehničke podrške. Dakle, cilj je odabrati rješenje koje će biti dugoročno. Za primjer informacijskog sustava u ovome radu kao programski jezik je odabran upravo Java, koji je jedan od najzastupljenijih programskih jezika, ponajviše kod ovakvih desktop aplikacija.

### 5.1 Java programski jezik

Java je objektno-orientirani, klasno bazirani, „visoki“ programski jezik opće svrhe. Java jezik je u potpunosti baziran na klasama koje opisuju podatke i ponašanje koje je povezano s instancama te klase. Instanca klase je objekt koji je stvoren prema klasi i jednak je svim ostalim objektima koji su instance iste klase. Podaci vezani uz objekt ili klasu su spremljeni u varijablama, a ponašanje vezano uz objekte ili klase se opisuje metodama (funkcijama). Sav kod je sadržan u klasama, i to u datotekama koje nose isto ime kao i klasa. Klase su dalje grupirane u pakete (engl. *package*) koji se mogu poistovjetiti s direktorijima u datotečnom sustavu. Java programski jezik je razvijen s namjerom da jednom napisani Java programski kod može biti pokrenut na više platformi koje podržavaju Javu bez potrebe za rekompilacijom. Java aplikacije su kompilirane u set instrukcija koji se izvršavaju na Java virtualnom stroju – JVM (engl. *Java virtual machine*) koji omogućava interoperabilnost među platformama bez obzira na sami *hardware*. Osim JVM-a, razvoj aplikacija u Java programskom jeziku omogućuju još dvije tehnologije: JRE (engl. *Java runtime environment*) i JDK (engl. *Java development kit*). JRE je Java komponenta koja na disku pokreće JVM, a JDK je kolekcija alata za razvoj Java programa koji se pokreće na JVM-u. Bitna komponenta

JDK-a je *javac* - kompajler Java koda. Kompajler ili programski prevoditelj, pretvara Java kod u strojni kod. Java programski jezik počiva na pet osnovnih principa, a to su:

1. Mora biti jednostavan i objektno - orijentiran
2. Mora biti robustan i siguran
3. Mora biti arhitekturno neovisan i portabilan
4. Mora se izvršavati s visokim performansama
5. Mora biti višedretven i s dinamičkim izvršavanjem [7]

## 5.2 Posebnost Jave kao objektno orijentirani jezik

Sintaksa Java koda je inspirirana starijim jezicima kao što je C++, ali je Java za razliku od njega napisana kao isključivo objektno – orijentirani jezik. Sav kod se nalazi u klasama i svaki podatak je objekt klase, uz izuzetak primitivnih tipova podataka (*integer, float, boolean, char*) koji nisu objekti klase zbog optimizacije performansi koda. Za razliku od C++, Java ne dozvoljava preopterećenje (engl. *overloading*) operatora i višestruko nasljeđivanje klase. Isto tako Java ne koristi aritmetiku pokazivača te ima automatsko upravljanje memorijom tako da sami programer ne mora ručno upravljati memorijom što često dovodi do grešaka. Automatsko upravljanje memorijom se postiže preko automatskog kolektora otpada AGC (engl. *Automatic garbage collector*) koji se brine o memoriji za vrijeme životnog vijeka objekta – odnosno od kreiranja do uništavanja pojedinog objekta. [14] Java kao i ostali objektno orijentirani jezici počiva na osnovnim načelima objektnog programiranja: enkapsulaciji, abstrakciji, nasljeđivanju i polimorfizmu. [15]

## 5.3 Java klase

Klase se u Javi nalaze u istoimenoj Java datoteci, no moguće je i kreirati klasu unutar klase - unutarnju ili tzv. ugniježdenu klasu (engl. *inner class, nested class*). Klasa se deklarira modifikatorom pristupa, ključnom riječju *class* i nazivom klase (po pravilu počinje velikim slovom) iza koje slijedi tijelo klase omeđeno vitičastim zagradama. [16] Modifikatori pristupa (engl. *access modifiers*) su ključne riječi koje određuju vidljivost klase, metoda i varijabli. Ovi modifikatori omogućuju prvo od osnovnih načela objektno – orijentiranog programiranja, a to je enkapsulacija. Enkapsulacija je ograničavanje pristupa nad podacima u kodu. Ukoliko je kod klase modifikator pristupa



zadan kao *public* tada će klasa biti vidljiva na razini svih paketa koda. S druge strane, ako nije zadan ni *public* modifikator već je klasa deklarirana samo ključnom riječi *class* tada se primjenjuje zadani (engl. *default*) modifikator. Ovakva vrsta klase je dostupna paketu u kojem se nalazi ali ne i ostalim paketima. [17] Na slici 23 je prikazana deklaracija klase Klasa kojoj je zadan *public* modifikator i nalazi se u paketu Kod.

```
package Kod;

/**
 *
 * @author Emanuel
 */
public class Klasa {

}
```

Slika 23 Javna klasa Klasa u paketu Kod

Osim modifikatora pristupa još neki od važnijih modifikatora su: *final* i *abstract*, koji se mogu dodati nakon modifikatora pristupa, a prije ključne riječi *class*. *Final* modifikator označava da klasa ne može biti nasljeđivana. Nasljeđivanje je drugo od osnovnih načela objektnog programiranja, a ono se odnosi na stvaranje pod-klasa koje nasljeđuju neku klasu. U tom slučaju pod-klase sadrže sve atribute i metode klase koju nasljeđuju, a uz to mogu definirati i svoje atribute i metode. Da bi klasa nasljeđivala drugu klasu potrebno je da joj u deklaraciji stoji ključna riječ *extends* i naziv nad-klase. [18] Primjer klase koja nasljeđuje drugu je vidljiv sa slike 24.

```
public class Klasa extends Main {
```

Slika 24 Pod-klasa Klasa nasljeđuje nad-klasu Main

*Abstract* modifikator označava da se ne mogu kreirati objekti klase već samo objekti nasljeđene klase. Abstraktne klase su sačinjene od isključivo abstraktnih metoda koje ne sadrže implementaciju, već samo definiraju što je svrha metode. Ovo se naziva abstrakcija i još je jedno od osnovnih načela objektno – orijentiranog programiranja.

Abstrakcija omogućava da je vidljivo samo ono što je bitno, a da je implementacija „skrivena“. [19]

## 5.4 Java varijable

Varijable u Javi se još nazivaju i atributi klase i polja (engl. *fields*). Deklaracija varijable se vrši ili u tijelu klase ili u tijelu neke metode i to deklaracijom modifikatora pristupa, tipom podatka i nazivom koji će varijabla nositi (pravilo je da počinje malim slovom). Varijablu je poželjno ukoliko situacija dopušta, inicijalizirati prilikom deklaracije, odnosno dodijeliti joj neku vrijednost. Modifikatori pristupa za varijable su: *public*, *private*, *protected* i zadani modifikator (engl. *default*). *Public* odnosno javni modifikator određuje da je varijabla dostupna svim klasama, *private* odnosno privatni modifikator određuje da je varijabla dostupna samo u klasi u kojoj je deklarirana. *Protected* odnosno zaštićeni modifikator određuje da je varijabla dostupna samo u istom paketu i u klasama koje nasljeđuje druge. Ukoliko ispred varijable ne stoji modifikator tada se primjenjuje zadani, i on određuje da je varijabla dostupna samo u paketu u kojem je deklarirana. Osim modifikatora pristupa, bitni opcionalni modifikatori koji dolaze nakon njih su *final* i *static*. *Final* označava da se vrijednost varijable ne može mijenjati nakon što je jednom inicijalizirana, dakle služi stvaranju konstante. *Static* modifikator označava varijablu koja pripada klasi, a ne instanci klase pa ju je moguće koristiti bez kreiranja objekta klase. Primjer deklaracija i inicijalizacija nekih varijabli je prikazan na sljedećoj slici (slika 25). Varijable *a* i *b* sa slike kao tip podataka imaju primitivne tipove dok ostale varijable kao tip podataka imaju klase. [20]

```
public class Klasa {  
  
    final int a = 1;  
    public static boolean b = true;  
  
    private String c = "Tekst";  
    String d = "Drugitekst";  
    Double e = 2.42;  
    static Integer f = 43;  
    String g;  
  
}
```

Slika 25 Razni tipovi varijabli u klasi

## 5.5 Java metode

Java metode su blokovi Java koda koji se izvršavaju na poziv. Metode se formiraju navođenjem modifikatora pristupa, tipa podatka povratne varijable (ukoliko metoda vraća varijablu – ako ne vraća tip je *void*), naziva varijable (pravilo je malim početnim slovom) i ulaznih parametara varijable ukoliko ih ima, a navedeni su unutar zagrada. Zatim slijedi tijelo metode koje se nalazi unutar vitičastih zagrada. Ukoliko je definiran povratni tip metode tada tijelo metode mora sadržavati ključnu riječ *return* i naziv varijable koja se vraća. Modifikatori za varijable se primjenjuju i za metode. Metode je moguće i preopteretiti (engl. *overloading*) odnosno imati više metoda istog imena ali različitog tipa i broja ulaznih parametara. Deklaracija i preopterećivanje metode je prikazano na sljedećoj slici (slika 26). [21]

```
public String metoda (String x)
{
    String var = x + " - plus X";
    return var;
}

public String metoda (String x,String y)
{
    return x + y;
}

public void metoda ()
{
    System.out.println("METODA");
}
```

Slika 26 Preopterećivanje metode

Osim preopterećivanja metode postoji i poništavanje (engl. *overriding*) metode. Pojam se odnosi na metode pod-klase koje imaju drugačiju implementaciju ali istu deklaraciju kao metode iz nad-klase. Poništavanje metode je prikazano na slici 27. Ova mogućnost različitog ponašanja metoda u pod-klasama omogućuje polimorfizam (višeobličnost) i posljednja je od navedenih načela objektno – orijentiranog programiranja. [22]

```

class Zivotinja {
    public void zvukZivotinje() {
        System.out.println("Zvuk.");
    }
}

class Pas extends Zivotinja {
    @Override
    public void zvukZivotinje() {
        System.out.println("VAAAUU");
    }
}

class Macka extends Zivotinja {
    @Override
    public void zvukZivotinje() {
        System.out.println("MJAUUU");
    }
}

```

Slika 27 Poništavanje metode zvukZivotinje u pod-klasama Pas i Macka

### 5.5.1 Metode pristupa i promjene

Metode pristupa (engl. *accessor*) i metode promjene (engl. *mutator*) su javne metode koje služe pristupanju i promjeni privatnih varijabli iz druge klasi. Imaju ključnu ulogu u osiguravanju enkapsulacije koda. Metoda promjene je namijenjena promjeni vrijednosti privatne varijable, i još se naziva i *set* metoda (engl. *set* = postaviti). Metoda pristupa je namijenjena dohvat u vrijednosti varijable i još se naziva i *get* metoda (engl. *get* = dobiti). Ovaj način osigurava da se dohvat i postavljanje varijabli izvršava na način kako je i zamišljeno u klasi u kojoj se varijable i nalaze. Pravilo imenovanja ovih metoda je da se koristi prefiks *get* ili *set* i naziv varijable kojoj se preko metode pristupa. Primjer ovih metoda je vidljiv na sljedećoj slici (slika 28). [23]

```

private String vrijednost = "tajna";

public String getVrijednost() {
    return vrijednost;
}

public void setVrijednost(String vrijednost) {
    this.vrijednost = vrijednost;
}

```

*Slika 28 Metode za pristup i promjenu*

### 5.5.2 Konstruktori

Konstruktori u Javi su blokovi koda koji služe za instanciranje, odnosno stvaranju nove instance klase. Iako podsjećaju na metode i ponekad se nazivaju specijalnim metodama, konstruktori nemaju povratni tip. Konstruktor kao i metoda izvršava kod i može primiti ulazne parametre, a poziv konstruktora kreira novi objekt. Naziv konstruktora je uvijek jednak nazivu klase koju instancira. Poziva se navođenjem naziva klase i ulaznih parametara, a nazivu obavezno prethodi ključna riječ *new* (slika 28).

```

Main objekt = new Main();

```

*Slika 28 Stvaranje instance klase Main*

Konstruktor ne mora biti manualno definiran u klasi, jer svaka klasa već sadrži zadani (engl. default) konstruktor za kreiranje instance. Ovaj konstruktor se nalazi u klasi i da ga klasa ne sadrži u kodu. Taj konstruktor ne prima nikakve ulazne parametre i ima prazno tijelo (slika 29).

```

Klasa () {

}

```

*Slika 29 Zadani konstruktor*

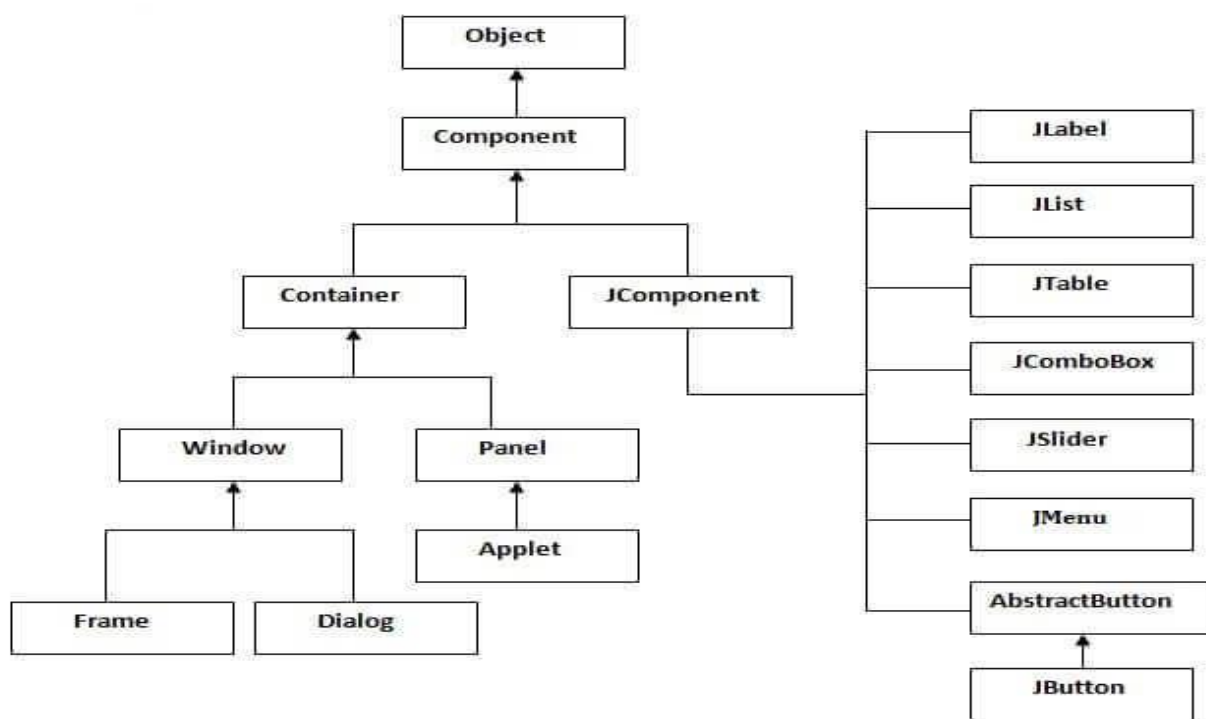
Kao i metode, konstruktore se također može „preopteretiti“, kao što je prikazano i na sljedećoj slici (slika 30).

```
Klasa () {  
  
}  
  
Klasa (String x)  
{  
    System.out.println("overload");  
    //Neki kod  
}
```

*Slika 30 Preopterećivanje konstruktora*

## 5.6 Java Swing

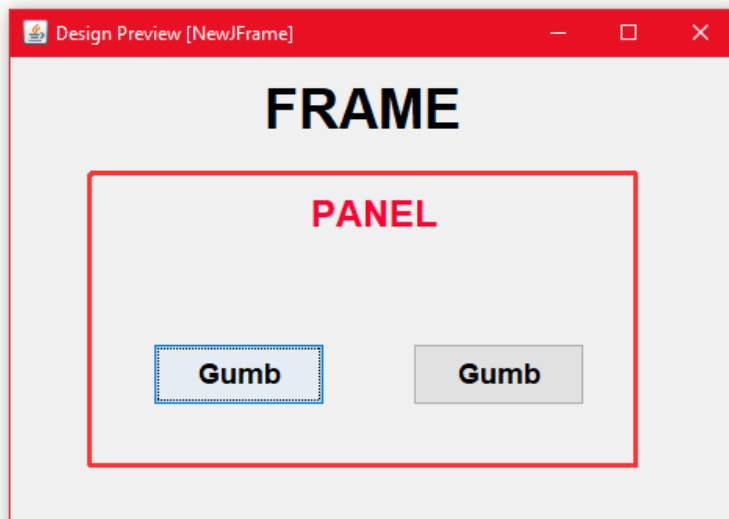
Java u današnje vrijeme (2020. godina) ima vrlo široku primjenu, i slovi kao jedan od najpopularnijih programskih jezika. I dok je Java nekoć primarnu ulogu imala u Windows *desktop* aplikacijama danas je najviše korištena za web aplikacije, web servise i Android aplikacije. Osim zadane verzije Jave, postoji mnoštvo dodatnih alata i sučelja za programiranje aplikacija (API). Jedan od njih je i Swing koji je skup alata za razvoj grafičkog sučelja u Javi. Swing je dio sučelja za programiranje aplikacija koje se naziva JFC (engl. *Java foundation classes*).[8] Ovaj alat je razvijen kako bi zamijenio prethodni alat za izradu grafičkog sučelja u Javi – AWT (engl. *Abstract window toolkit*) i kao takav je ekstenzivniji i prilagodljiviji od prethodnika. Omogućuje raznovrsnu promjenu izgleda teme komponenti bez velikih promjena u kodu. Swing elementi su za razliku od prethodnika neovisni o platformi zbog svoje baziranosti na Java kodu pa ih se naziva i „laganim“ elementima (engl. *lightweight element*). Grafički elementi koji su dostupni kroz Swing su spremnici (engl. *containers*) kao što su prozori i ploče, te komponente kao što su gumbi, tekstna polja, tablice, izbornici itd. Sve Swing grafičke komponente su zapravo klase koje nasljeđuju Java klasu Object. Swing klase su smještene u paketu koji se zove javax.swing. Komponente je lako prepoznati po prefiksu „J“ tako primjerice Swing gumb (engl. *button*) odgovara klasi JButton.



Slika 31 Hijerarhija klasa Swing komponenti [8]

### 5.6.1 Java Swing spremnici

Grafičko sučelje Windows *desktop* aplikacije se uvijek nalazi u nekoj vrsti spremnika koji služi kao objekt na kojemu se nalaze sve ostale grafičke komponente. U Swing-u ta vrsta spremnika može biti klasa `JPanel` (hrv. ploča) ili `JWindow` (hrv. prozor). Ploča je jednostavni spremnik, dok je prozor spremnik najviše razine koji može sadržavati ikonu, naslov te kontrolne gumbе. Postoje dvije vrste prozora odnosno dvije klase Swing komponenata koje nasljeđuju klasu `JWindow`, a to su: `JFrame` (hrv. okvir) i „`JDialog`“ (hrv. dijalog). `JFrame` spremnik za razliku od `JDialog` spremnika uvijek ima ikonu, naslov, dekoracije i gumbе za kontrolu prozora. Objekt klase `JDialog` je uglavnom korišten kao modalni spremnik, odnosno spremnik koji blokira korištenje drugih komponenti dok nije zatvoren). [9] Swing aplikacije počivaju u potpunosti na ovim vrstama spremnika. Te klase u svojem konstruktoru sadržavaju metodu `initComponents()` koja kreiranjem objekta klase spremnika inicijalizira i sve ostale komponente koje se nalaze u spremniku. Razlika `JFrame` i `JPanel` komponenti je prikazana na slici 32.

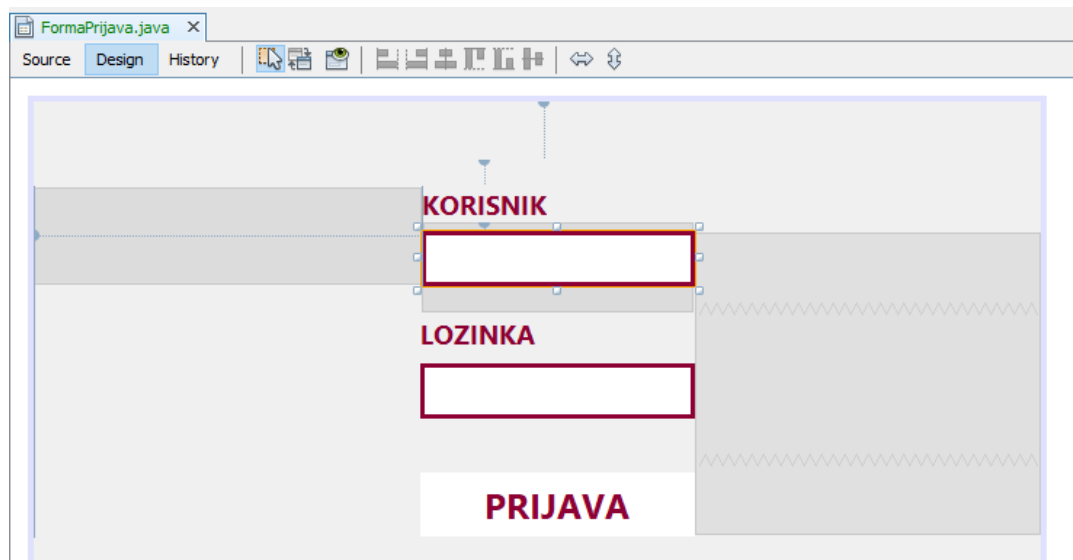


Slika 32 Primjer JFrame i JPanel komponenti

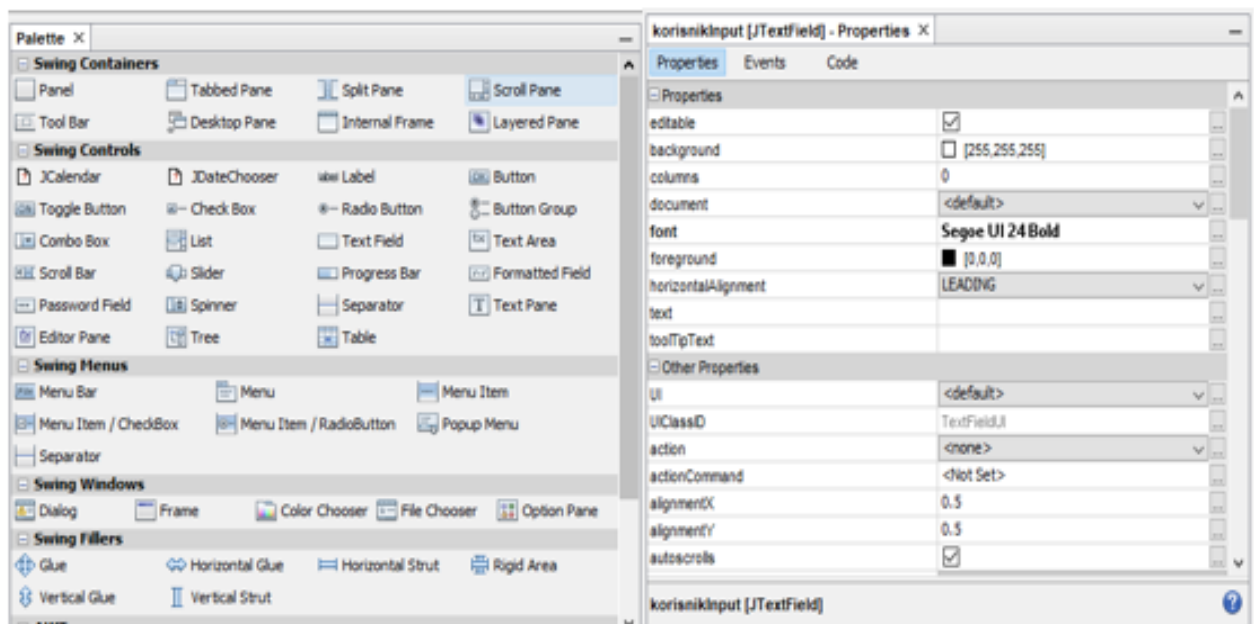
## 5.5 GUI Builder

U većini razvojnih okolina Swing je potrebno dodatno instalirati uz postojeće okruženje za razvoj Java. NetBeans već dolazi sa Swing klasama komponenti ali i s GUI Builder alatom koji omogućuje brzo kreiranje Swing grafičkih sučelja uz minimalno pisanje koda za komponente. Tako primjerice nije potrebno kreirati objekt klase JFrame i ručno upisivati dimenzije samog okvira u Java Main datoteku, već samo povući komponentu JFrame iz palete komponenti na radnu površinu GUI Builder-a. Ovaj princip se naziva i „WSIWYG“ princip (engl. *What you see is what you get* = hrv. Što vidiš to i dobiješ). Dimenzije i ostale opcije komponente je moguće uređivati kroz *properties* prozor. Osim opcijama (atributima) komponente, kroz *events* karticu je moguće pristupiti i metodama ponašanja komponente. GUI builder se otvara kreiranjem nove JFrame forme (desni klik na naziv projekta – *new* – *JFrame form*) pa odabirom *design* prikaza. Sučelje GUI buildera je prikazano slikom 33, a izbor i atributi komponenti na slici 34.





Slika 33 GUI builder



Slika 34 Paleta komponenti (lijevo), opcije atributa komponente (desno)

## 6. IZRADA APLIKACIJE ACADEMUS

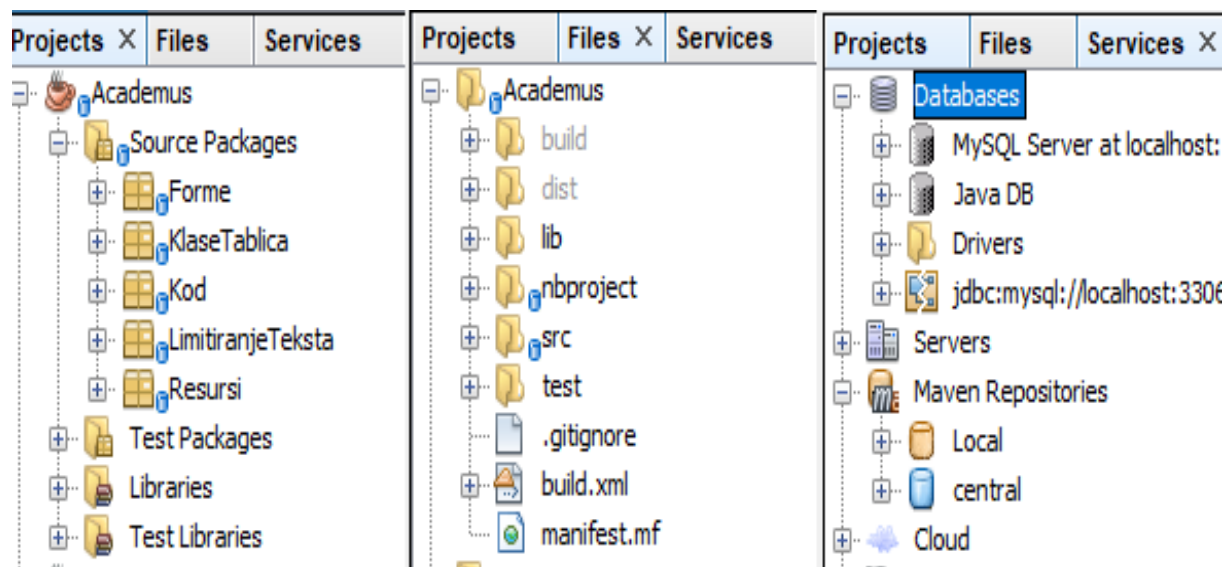
### 6.1 Funkcionalnosti aplikacije

Glavna svrha aplikacije Academus je praćenje nastavnog procesa neke obrazovne ustanove za visoko obrazovanje. Korisnik aplikacije je djelatnik obrazovne ustanove koji može biti izvršitelj (profesor, asistent) na jednom ili više kolegija koji su mu zadani unutar baze podataka. Uz djelatnika je vezano korisničko ime i lozinka kojima se služi za prijavu u aplikaciju, a može pristupiti aplikaciji i kroz više korisničkih imena / lozinki. Osim toga djelatnik može imati upisane kolegije s različitih odjela i smjerova. Nakon prijave u aplikaciju, korisnik može koristiti funkcionalnosti koje se odnose općenito na ustanovu (kao što su imenici i adresari djelatnika i studenata te pregled zadnjih prijava u sustav) ili koristiti funkcionalnosti koje se tiču specifičnog kolegija. Kako bi pristup funkcionalnostima kolegija bio moguć, korisnik mora najprije odabrati željeni kolegij iz liste pridodanih. Nakon odabira kolegija, korisnik na raspolaganju ima pregled studenata koji su upisani na navedeni kolegij, pregled i unos ocjena za pojedinog studenta (ocjena usmenog i pismenog ispita te konačna ocjena), pregled, unos i brisanje održavanja nastave, unos prisustva pojedinog studenta za uneseno održavanje nastave te pregled svih dolazaka pojedinog studenta na nastave kolegija. U svakome trenutku izvršavanja aplikacije, korisnik ima mogućnost odjaviti se i prijaviti se s drugim korisničkim profilom u aplikaciju.

### 6.2 Kreiranje Java projekta

Prvi korak u kreiranju Java aplikacije je kreiranje projekta u razvojnoj okolini. Za razvoj Academus aplikacije je korištena NetBeans razvojna okolina koju je razvila Apache kompanija. Za kreiranje projekta u NetBeansu potrebno je s izbornika odabrati *File*, pa stavku *New project*. Nakon odabira novog projekta prikazuje se prozor za odabir vrste projekta koji će se kreirati. Na raspolaganju su tri Java kategorije izrade projekta – Maven, Gradle i Ant. Razliku između njih je u načinu izrade aplikacije (engl. *build*), organizaciji repozitorija i zavisnosti (engl. *dependencies*). Maven i Ant imaju XML datoteku kao format izradbene skripte (engl. *build script*), dok Gradle koristi Groovy (jedan od JVM jezika). [10] Za kreiranje Academus projekta kao opcija kreiranja je odabrana *java application* opcija Java Ant kategorije. Nakon kreiranja projekta, razvojna okolina generira u izvorišnom paketu (engl. *source package*) Java klasu *Main* unutar koje se nalazi *main* metoda. Unutar metode *main* se u Java programskom

jeziku po pravilu uvijek izvršava čitavi kod Java aplikacije. NetBeans omogućava da programer na jednostavan način za vrijeme rada na projektu pregledava i organizira datoteke projekta. Projekt je moguće promatrati kroz tri opcije, kroz pogled na pakete, kroz pogled na datoteke i kroz pogled na servise aplikacije. Ako se projekt promatra kroz pakete moguće je vidjeti podjelu na osnovne pakete od kojih se sastoji projekt. U paketu *Source* se nalazi kod za izvršavanje aplikacije, u *Test* paketu se nalazi kod koji je namijenjen testiranju aplikacije, a u *Libraries* paket se smještaju kolekcije klasa nekog vanjskog koda koji možemo koristiti u svojoj aplikaciji. Ako je odabran pogled na projekt kroz datoteke, vidljive su sve datoteke koje se nalaze u projektu, uključujući i XML izradbenu datoteku. Pogled kroz servise omogućuje upravljanje servisima koji su integrirani s aplikacijom, a to mogu biti baza podataka, vanjski repozitoriji, poslužitelji i drugo. Različiti pogledi na projekt su vidljivi na slici 35.



Slika 35 Pogledi na projekt

Nakon kreiranja Academus projekta, paket u kojem se nalazi Main klasa je preimenovana iz generičkog naziva koji je dodijelio NetBeans u Kod zbog lakšeg snalaženja u organizaciji datoteka projekta. Preimenovanje datoteka i općenito objekata u NetBeansu se vrši pritiskom desnog klika na objekt preimenovanja te odabirom opcije *Refactor*, pa *Rename*. Ova metoda omogućava da se željenom objektu promijeni naziv na razini čitavog projekta.

## 6.3 Kreiranje klase za integraciju s bazom podataka

U paketu Kod je kreirana klasa naziva BazaPodataka. Zadaća ove klase je da bude poveznica Java i MySQL baze podataka, odnosno da sadrži varijable koje su vezane uz konekciju s bazom i da sadrži metode koje omogućuju rad s bazom. Da bi Java i MySQL mogli biti integrirani u aplikaciji potrebno je implementirati JDBC (engl. Java database connectivity API) upravljački program (engl. *driver*) u projekt. JDBC upravljački program je *software* koji je zadužen za to da omogući Java kodu da komunicira s bazom podataka. Instalacija ovog upravljačkog programa započinje preuzimanjem odgovarajuće JAR (Java arhiva) datoteke sa službene MySQL stranice. Za implementaciju u NetBeans, potrebno je u pogledu projekta kroz servise pritisnuti desni klik na *Databases*, pa odabrati *New connection*. Kao vrstu upravljačkog programa je potrebno odabrati MySQL *driver*. Pritiskom na *Add* preuzeta JAR datoteka upravljačkog programa se dodaje u projekt iz direktorija u kojem je spremljena. Upravljački program je po dodavanju vidljiv u paketu *Libraries*. Unutar klase BazaPodataka je napisana metoda *konekcijaBaze* (slika 36) koja koristi implementirani JDBC upravljački program za uspostavljanje konekcije s bazom podataka. Uspostavljanje konekcije se vrši kroz metodu *getConnection* koja kao ulazne parametre prima varijable: lokaciju upravljačkog programa, adresu baze podataka te korisničko ime i lozinku za prijavu u bazu podataka. Ova metoda kao rezultat izvršavanja vraća objekt klase *Connection*. Varijable ulaznih parametara koje su potrebne za konekciju s bazom podataka su spremljene kao privatne varijable unutar klase BazaPodataka. Sama metoda *konekcijaBaze* vraća integer vrijednost i to 1 ako je konekcija s bazom podataka uspješno ostvarena, -1 ako upravljački program nije i -2 ako je konekcija upravljačkog programa s bazom podataka neuspješna. Osim ove metode, u klasi je napisana i metoda *zatvaranjeKonekcije* (slika 37), koja nad *Connection* objektom poziva metodu *close* i tako zatvara konekciju na kraju korištenja. Objekt klase BazaPodataka s nazivom *db* je kreiran u klasi *Main* i svi pozivi metoda koji se tiču komunikacije s bazom podataka se obavljaju upravo preko tog objekta.

```

public int konekcijaBaze() {
    int status = 0;
    try {
        Class.forName(DRIVER);
        dbCon = DriverManager.getConnection(URL, USER, PASSWORD);
        dbCon.setAutoCommit(false);
        status = 1;
    }
    catch (ClassNotFoundException ex) {
        ex.printStackTrace(System.out);
        System.out.println("Driver klasa nije pronađena!");
        status = -1;
    }
    catch (SQLException ex) {
        ex.printStackTrace(System.out);
        System.out.println("Problem s uspostavljanjem konekcije baze podataka!");
        status = -2;
    }
    return status;
}

```

*Slika 36 Metoda konekcijaBaze*

```

public void zatvaranjeKonekcije()
{
    try{
        dbCon.close();
        System.out.println("Konekcija s bazom podataka uspješno zatvorena.");
    }
    catch(SQLException ex)
    {
        System.out.println("Problem sa zatvaranjem konekcije baze podataka.")
    }
}

```

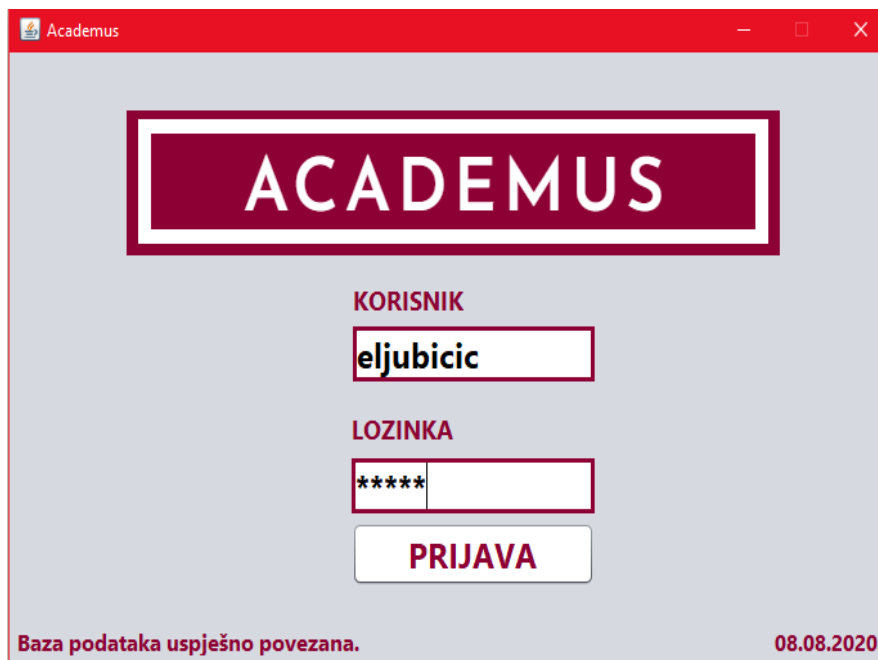
*Slika 37 Metoda zatvaranjeKonekcije*

## 6.4 Forma za prijavu

Prva JFrame komponenta koja je kreirana je forma FormaPrijava. Riječ je o prvoj formi koju korisnik vidi prilikom ulaska u aplikaciju i formi koja služi za prijavu (engl. *log in*).

### 6.4.1 Komponente forme za prijavu

Forma se sastoji od polja za unos korisničkog imena (JTextField), polja za unos lozinke (JPasswordField), gumba (JButton) i dva natpisa teksta (JLabel). Forma je prikazana na sljedećoj slici (slika 38).



Slika 38 Izgled forme za prijavu

Polja za unos korisničkog imena i za unos lozinke imaju ograničen broj znakova koje korisnik može unijeti što je postignuto postavljanjem *Document* atributa na objekt klase *JTextFieldLimit* s proslijeđivanjem željenog ograničenja broja znakova u konstruktor objekta. Klase za limitacije teksta su postavljene u *LimitiranjeTeksta* paket projekta. Natpisi teksta na formi se učitavaju prikazivanjem prozora, jedan se puni informacijom je li baza podataka uspješno povezana, a rezultat je djelovanja metode *konekcijaBaze*, a drugi natpis je trenutni datum. Postavljanje limitacije teksta je prikazano na slici 39.

```
korisnikInput.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N
korisnikInput.setDocument(new LimitiranjeTeksta.JTextFieldLimit(12));
korisnikInput.setBorder(javax.swing.BorderFactory.createLineBorder(new j
```

Slika 39 Postavljanje ograničenja broja znakova na *JTextField*

#### 6.4.2 Prijava korisnika

Pritiskom gumba na formi, okida se događaj (engl. *event*) koji poziva metodu *prijavaGumbActionPerformed*. Ova metoda čita podatke koje je korisnik unio kroz formu (korisničko ime i lozinku), ako korisnik nije valjane podatke prikazuje mu se dijaloški okvir koji ga upozorava da mora unijeti podatke. Ako je unio valjane podatke oni se proslijeđuju metodi *dohvatiKorisnika* koja se nalazi u klasi *BazaPodataka*. Metoda *dohvatiKorisnika* (slika 40) preko JDBC upravljačkog programa šalje upit

prema bazi podataka postoji li korisnik u tablici korisnik s korisničkim imenom koje je uneseno. Ako postoji, metoda provjerava odgovara li unesena lozinka onoj koja je spremljena u bazi podataka za tog korisnika. Ukoliko ne postoji navedeni korisnik u bazi ili je lozinka netočna, korisnik aplikacije dobiva odgovarajuću dijalošku poruku na ekran. Ako korisnik u bazi postoji i lozinka je ispravna tada su rezultati SQL upita za ovog korisnika: korisničko ime, lozinka i administratorska ovlast. Rezultati SQL upita iz baze podataka dolaze unutar *Result set* objekta iz kojeg se ekstrahiraju odgovarajućom metodom, zavisno o tipu podatka. Tako se primjerice varchar vrijednost iz baze podataka, u Javi dohvaća putem `getString` metode koja varchar niz znakova konvertira u Java string. Ovi rezultati upita se zatim koriste za kreiranje objekta Java klase *Korisnik* koji se nalazi u klasi *Main*. Isto tako rezultat SQL upita *oibDjelatnika* se koristi za kreiranje objekta klase *Djelatnik* koji se nalazi kao varijabla unutar klase *Korisnik*. Java klase koje predstavljaju entitete kao što su ovdje korisnik i djelatnik se nalaze u paketu *KlaseTablica*. Nakon kreiranja objekata ovih klasa, vrši se skrivanje prozora pozivom metode `dispose` nad trenutnom instancom prozora, a istovremeno i prikazivanje sljedeće forme – glavne forme aplikacije. U pozadini se poziva metoda klase *BazaPodataka* koja se zove `unosPrijave` (slika 41). Ova metoda u bazu podataka radi unos u tablicu *Prijava* i to s podatkom o kojem se korisniku radi i u koje vrijeme je prijava izvršena.

```
public int dohvatiKorisnika(String u,String p)
{
    int status = 0;
    String oib = "oib";
    boolean admin=false;

    try {
        sql = "SELECT DISTINCT(korisnickoIme),lozinka,admin,oibDjelatnik "
            + "FROM korisnik WHERE korisnickoIme = '"+u+"' ";

        try {
            dbStmnt = dbCon.prepareStatement(sql);

        } catch (SQLException ex) {
            ex.printStackTrace();
            System.out.println("Problem s kreiranjem Statement objekta");
            this.konekcijaBaze();
        }

        rs = dbStmnt.executeQuery(sql);
        if (rs.next()) {

            status = 1;

            String pass = rs.getString("lozinka");
            admin = rs.getString("admin").equals("DA");
            oib = rs.getString("oibDjelatnik");
        }
    }
}
```

*Slika 40 Metoda dohvatiKorisnika*

```

        public void unosPrijava(String kime)
    {

        try {
            sql = " INSERT INTO prijava(idKorisnik,datumPrijava) "
                + " SELECT idKorisnik,NOW() from korisnik"
                + " where korisnickoIme = '" + kime + "' ";

            dbStmnt.execute();
            dbCon.commit();
        }
    }

```

Slika 41 Unos vrijednosti u tablicu Prijava kroz metodu unosPrijava

## 6.5 Glavna forma

Centralni upravljački JFrame prozor aplikacije je FormaGlavna. Putem ove forme korisnik pristupa funkcionalnostima informacijskog sustava i otvara druge forme koje su u odnosu na ovu modalni dijalozi. Forma se sastoji od JMenuBar komponente na vrhu koja je zapravo traka izbornika i od četiri JPanel komponenti koji služe kao spremnici druge razine. U JMenuBar traci izbornika nalaze se dva izbornika (JMenu komponente). Oba izbornika su padajućeg tipa. Prvi izbornik nosi naziv Imenik, a drugi izbornik nosi naziv Adresar. Oba izbornika sadrže stavke (JMenuItem komponente) za odabir podataka iz tablice Djelatnik ili Student. Tako svakim klikom na pojedinu stavku izbornika je moguće otvoriti različitu formu koja je po prirodi modalni dijalog. Forme su jednostavne i sastoje se od jedne JTable komponente. Imenici nude informacije o brojevima telefona i e-mail adresama osoba, dok adresari nude informacije o adresama prebivališta i boravišta. Tablice s djelatnicima kao osnovne podatke sadrže ime i prezime djelatnika te njegovu titulu dok tablice sa studentima kao osnovne podatke sadrže ime i prezime i njihov JMBAG. U prvom spremniku druge razine se nalazi gumb za odabir kolegija, te informacije o odabranom kolegiju: naziv kolegija, smjera i odjela (ako je kolegij uopće odabran). U drugom spremniku se nalaze podaci o korisniku koji je prijavljen u aplikaciji (korisničko ime, ime, prezime i titula djelatnika koji je vezan uz korisnika) te gumb za odjavu korisnika. U trećem spremniku se nalaze gumbi za glavne funkcionalnosti aplikacije: pregled studenata, unos ocjena, unos nastave i pregled dolazaka. Svi ovi gumbi su onemogućeni (engl. *disabled*) dok nije odabran željeni kolegij od strane korisnika. I u zadnjem, četvrtom spremniku se



nalazi tablica s popisom posljednjih prijava u aplikaciju i korisnika koji su izvršili te prijave. Glavna forma i prozor imenika djelatnika su prikazani na slici 42 i 43.

Broj prijave	Korisnicko ime
11	ppasic
10	ppasic
9	ppasic
8	mrivic
7	ppasic
6	ppasic
5	ppasic
4	ppasic
3	mrivic
2	pasic2
1	mrivic

Slika 42 Glavna forma

Prezime	Ime	Titula	E-pošta	Mobitel	Telefon
Ivkić	Ivan	bacc. ing. comp.	ivkiic12@academ...	0998989912	012757454
Mačić	Marko	dipl. iur.	macko@academ...	0911112313	012986553
Pašić	Jerolim	dipl. ing.	jere@academus.hr	0989297651	012999915
Rašić	Stevan	dipl. ing.	stevo@academus...	0911211133	
Ričić	Predrag	dipl. ing.	rricic@academus....	0991209311	012978312

Slika 43 Imenik djelatnika

## 6.6 JTable tablica

U aplikaciji se često koristi JTable element tablice kako bi se prikazao set vrijednosti koji je vraćen SQL upitom iz baze podataka. Primjer takve tablice je i popis prijava s glavne forme. Tablica se prvo povuče u spremnik na željeno mjesto i na odgovarajuće dimenzije. U Academus aplikaciji su sve tablice uređene kako bi izgledom odgovarale tematici ostatka aplikacije. Uređivanje izgleda tablice je rađeno manualnim pisanjem koda u konstruktoru klase forme, i to zadavanjem HSB (engl. *hue*, *saturation*,

*brightness*) vrijednosti boje zaglavlju (engl. header) tablice i postavljanjem fonta slova u zaglavlju na željenu boju i veličinu (slika 44).

```
JTableHeader header = tablica.getTableHeader();

float [] hsb = Color.RGBtoHSB(141, 0, 51, null);
float hue = hsb[0];
float saturation = hsb[1];
float brightness = hsb[2];

header.setForeground(Color.getHSBColor(hue, saturation,
brightness));
header.setFont(new Font("Segoe UI", Font.BOLD, 20));
```

*Slika 44 Uređivanje zaglavlja tablice*

Sami podaci koji će popunjavati tablicu dolaze pozivom jedne od metoda klase BazaPodataka koja obavlja upit prema bazi podataka. Pritom se metodi predaju varijable koje su potrebne za SQL WHERE uvjet. Pošto metode vraćaju set rezultata, kao povratnu vrijednost nemaju običan tip podatka (npr. String) već kolekciju podataka u obliku ArrayList objekata. ArrayList je spremnik Java varijabli kojima se može pristupiti preko indeksa elementa. Unutar metode svaki red koji je vraćen kao rezultat upita se dodaje u ArrayList kao neki objekt, obično objekt jedne od klase tablica entiteta iz paketa KlaseTablica. Primjerice ako je cilj dohvatiti podatke o studentu i pohraniti ih kao jedan element ArrayList-a tipa student tada će se dohvatiti redak tablice student te od vrijednosti atributa JMBAG, ime, prezime itd. Zatim se instancira objekt klase student s dohvaćenim vrijednostima i pohranjuje u ArrayList. Postupak se ponavlja dok god postoje redovi u rezultatu upita.

```

public ArrayList <Student> dohvatiImenikStudenata()
{
    ArrayList <Student> studenti = new ArrayList();

    try {
        sql = "SELECT ime, prezime, jmbag, email, mobitel, telefon "
            + "FROM student ORDER BY 2,1,3,4,5,6 asc";

        try {
            dbStmnt = dbCon.prepareStatement(sql);

        } catch (SQLException ex) {
            ex.printStackTrace();
            System.out.println("Problem s kreiranjem Statement objekta");
            this.konekcijaBaze();
        }

        rs = dbStmnt.executeQuery(sql);

        boolean records = false;
        while (rs.next()) {

            records = true;
            String jmbag = rs.getString("jmbag");
            String ime = rs.getString("ime");
            String prezime = rs.getString("prezime");
            String email = rs.getString("email");
            String telefon = rs.getString("telefon");
            String mobitel = rs.getString("mobitel");

            Student student = new Student(jmbag, ime, prezime,
                email, mobitel, telefon);

            studenti.add(student);
        }
    }
}

```

*Slika 45 Metoda za dohvat seta rezultata iz baze podataka*

Dohvaćena kolekcija podataka (u primjeru ArrayList tipa Student) je pridružena varijabli ArrayLista na formi. Zatim je stvoren objekt klase DefaultTableModel koji služi za pristupanje modelu tablice kako bi unijeli podatke u redove JTable komponente. Korištena je *for* petlja za pristupanje svakoj varijabli unutar objekta iz ArrayList kolekcije i pohranili ga na odgovarajući indeks unutar polja varijabli (engl. *array*).

```

ArrayList <Student> studenti = Main.db.dohvatiImenikStudenata();
DefaultTableModel model = (DefaultTableModel) this.tablica.getModel();

Object [] row = new Object [6];

for(int i=0;i<studenti.size();i++)
{
    row[0]=studenti.get(i).getJmbag();
    row[1]=studenti.get(i).getPrezime();
    row[2]=studenti.get(i).getIme();
    row[3]=studenti.get(i).getEmail();
    row[4]=studenti.get(i).getMobitel();
    row[5]=studenti.get(i).getTelefon();
    model.addRow(row);
}

```

*Slika 46 Pohrana vrijednosti u JTable tablicu*

JTable tablica u ovoj aplikaciji je često korištena na način da pritiskom na red tablice se izvršava neki kod. To je postignuto događajem pritiska miša na tablicu koji okida

metodu `tablicaMousePressed` (slika 47). Metoda funkcionira tako da prati koji je red tablice trenutno selektiran te s obzirom na te vrijednosti izvršava dalje željeni kod.

```
private void tablicaMousePressed(java.awt.event.MouseEvent evt) {
    int i = tablica.getSelectedRow();
    TableModel model = tablica.getModel();
    jLabel2.setText(model.getValueAt(i, 0).toString());

    odabraniKolegijID = model.getValueAt(i, 1).toString();
    nazivOdabraniKolegij = model.getValueAt(i, 0).toString();
    nazivOdabraniSmjer = model.getValueAt(i, 3).toString();
    nazivOdabraniOdjel = model.getValueAt(i, 4).toString();
    ulogaNaziv = model.getValueAt(i, 2).toString();

    jButton1.setEnabled(true);
}
```

Slika 47 Metoda događaja tablice

## 6.7 Forma za odabir kolegija

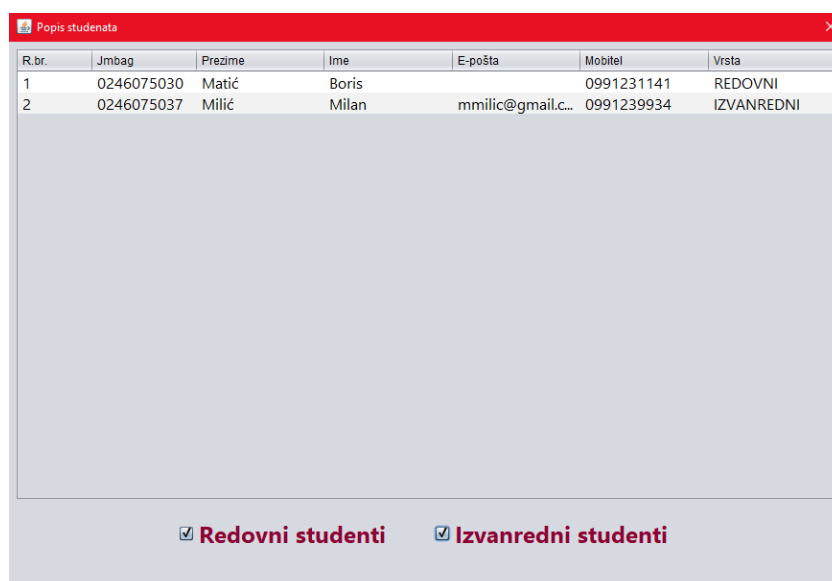
Odabir kolegija je prvi korak u korištenju funkcionalnosti *Academus* informacijskog sustava, a to se ostvaruje upravo kroz prozor za odabir kolegija čija Java datoteka se zove `FormaOdaberiKolegij` (slika 48). Korisnik u tablici kolegija može vidjeti samo one kolegije kojima je dodijeljen kao izvršitelj u bazi podataka. Korisnik kroz tablicu vidi dostupne kolegije, svoju ulogu na njima, smjer i odjel kojem pripadaju. Pritiskom na red tablice je moguće odabrati kolegij, a pritiskom gumba i potvrditi odabir. Nakon potvrde odabira, u klasi forme se učitavaju varijable s odabranim ID kolegija i nazivom kolegija te se omogućuju (engl. *enable*) gumbi za druge funkcionalnosti na glavnoj formi.



Slika 48 Prozor za odabir kolegija

## 6.8 Forma za pregled studenata na kolegiju

Djelatnik obrazovne ustanove kroz formu FormaKolegijStudenti (slika 49) ima mogućnost pregleda podataka svih studenata koji su upisani na odabranome kolegiju. Kroz navedenu formu na raspolaganju ima i odabir želi li prikaz redovnih, izvanrednih ili svih studenata. Odabir ove opcije je izveden kroz JCheckbox komponente koje kada se odaberu okidaju događaj osvježavanja tablice s podacima o studentima te se vrši učitavanje podataka zavisno o odabiru.



The screenshot shows a window titled "Popis studenata" with a table of student data and two checkboxes at the bottom.

R.br.	Jmbag	Prezime	Ime	E-pošta	Mobitel	Vrsta
1	0246075030	Matić	Boris		0991231141	REDOVNI
2	0246075037	Milić	Milan	mmilic@gmail.c...	0991239934	IZVANREDNI

Below the table, there are two checkboxes: ☒ Redovni studenti and ☒ Izvanredni studenti.

Slika 49 Pregled studenata na kolegiju

## 6.9 Forma za pregled i unos ocjena

### 6.9.1 Korištenje forme za pregled i unos ocjena

Pritiskom gumba za unos ocjena s glavne forme, korisniku se prikazuje forma FormaKolegijOcjene na kojoj je tablica s ocjenama svih studenata na kolegiju. Ovdje je moguće filtrirati prikaz da prikazuje samo studente koji nemaju unesenu konačnu ocjenu, samo studente koji imaju unesenu konačnu ocjenu, ili sve studente. Unos ocjene ili ispravljanje unosa ocjene se vrši pritiskom miša na željeni red studenta u tablici te pritiskom na gumb s imenom tog studenta. Nakon ove radnje se otvara dijaloška forma za unos ocjena FormaUnosOcjene. Ova forma sadrži osnovne informacije o studentu i polja za unos ocjene iz pismenog i usmenog ispita te konačne ocjene. Osim toga, na formi se nalazi JCalendar komponenta pomoću koje je moguće odabrati datum polaganja ispita. Ova komponenta pretvara datum automatski u željeni

format. Prilikom završetka upisivanja željenih podataka, korisnik radi potvrdu unosa na gumb Unos. Pritisak na ovaj gumb okida metodu koja poziva metodu unosOcjena klase BazaPodataka za upisivanje u tablicu ocjena baze podataka.

### 6.9.2 Metoda za unos ocjena

Metoda unosOcjena (slika 50) prosljeđene vrijednosti iz polja s forme ne upisuje kao nove redove u tablicu baze podataka (INSERT), već ažurira postojeće vrijednosti atributa u tablici (UPDATE). Ovaj princip se koristi radi mogućnosti ispravka već unesenih vrijednosti kroz istu formu i metodu, osim toga postupak pridruživanja nekog studenta kolegiju se u Academus bazi podataka čini tako da se unese novi red u tablicu Ocjena s kompozitnim ključem koji se sastoji od JMBAG-a studenta i identifikacijskog atributa kolegija kojem pripada. Ukoliko bi se ponovno pokušalo napraviti unos s istim podacima navedenih ključeva, baza bi dojavila pogrešku u vidu povrede integriteta. Bitno je napomenuti da vrijednosti datuma prije unosa u bazu treba prilagoditi u odgovarajući oblik pretvorbom Java oblika u SQL oblik.

```
public void unosOcjena(String jmbag,String idKolegij,String p,
    String u,String k,Date polaganje)
{
    try {
        sql = " UPDATE  ocjena SET ocjenaUsmeni = ? , ocjenaPismeni = ? , "
            + " konacnaOcjena = ? , datumPolaganja = ? , datumUnosa = ? "
            + " WHERE jmbagStudent = ? AND idKolegij = ? ";

        try {
            dbStmtnt = dbCon.prepareStatement(sql);

            java.sql.Date datumPolaganja;
            java.sql.Date datumUnosa= new java.sql.Date(Calendar.getInstance().
                getTime().getTime());
            dbStmtnt.setString (1,u);
            dbStmtnt.setString (2,p);
            dbStmtnt.setString (3,k);
            if(polaganje != null)
            {
                datumPolaganja= new java.sql.Date(polaganje.getTime());
                dbStmtnt.setDate(4, datumPolaganja);
            }
            else
            {
                dbStmtnt.setNull(4,java.sql.Types.NULL);
            }
            dbStmtnt.setDate(5,datumUnosa);
            dbStmtnt.setString (6,jmbag);
            dbStmtnt.setString (7,idKolegij);
```

*Slika 50 Metoda za unos ocjena*

## 6.10 Forma za pregled, unos i brisanje nastave

Pritiskom gumba Unos nastave s glavne forme, korisniku se prikazuje forma FormaKolegijNastava (slika 51) na kojoj se nalazi tablica s upisanim održavanjima nastave na odabranom kolegiju, i to prema vrsti nastave. Promjena vrste nastave koja je fokusirana se obavlja preko JCombobox komponente koja služi kao padajući izbornik. Promjenom odabira u JCombobox komponenti okida se i osvježavanje vrijednosti u tablici. Podaci o nastavi koji se nalaze u pojedinim redovima tablice su ID, opis, datum održavanja, vrijeme održavanja, prostorija, lokacija, djelatnik koji je održao nastavu te broj prisutnih na nastavi. Osim pregleda upisanih nastava kroz ovu formu su omogućene i funkcionalnosti unosa prisustva pojedinog studenta za odabranu nastavu, unos nove nastave i brisanje odabrane nastave iz baze podataka.

The screenshot shows a window titled 'Nastava' with a red header bar. At the top, there is a dropdown menu showing 'Auditorne vježbe - neobavez.' and a button labeled 'NOVI UNOS'. Below this is a table with the following data:

Opis	Datum	Od	Do	Prostorija	Lokacija	Djelatnik	Broj prisut.
Audit1	01.02.2002	07:00:00	07:00:00	Predavaona A	Vrbik 16, Za...	Ričić Predrag	0
Audit23	02.04.2011	07:00:00	07:00:00	Predavaona A	Vrbik 16, Za...	Ričić Predrag	1

Below the table, the text 'Audit23 - 02.04.2011' is displayed. At the bottom, there are two buttons: 'UNOS PRISUSTVA' and 'BRISANJE'.

Slika 51 Forma za pregled, unos i brisanje nastave

### 6.10.1 Unos nove nastave

Unos održane nastave u bazu podataka se vrši kroz formu FormaUnosNastava. Navedena forma se prikazuje kada korisnik na formi FormaKolegijiNastava pritisne gumb Novi unos. Prikazana forma je modalna i sadrži informacije o kolegiju i odabranom tipu nastave za koji se radi novi unos, a sadrži i polja za unos datuma održavanja, vrijeme održavanja, opis nastave te JCombobox komponentu za odabir jedne od upisanih lokacija održavanja. Unos se potvrđuje gumbom Unos koji poziva

metodu unosNastave iz klase BazaPodataka (slika 52). Ova metoda za razliku od metode unosOcjena u bazu podataka ubacuje nove redove u tablicu (INSERT).

```
public void unosNastave(Date datumOdrz, Time vrijemeOd, Time vrijemeDo,
                        String opis, int idTip, int idLokacija,
                        String oibDjelatnik, String idKolegij)
{
    try {
        sql = " INSERT into nastava(datumOdrzavanja,vrijemeOdrzavanja,opis,idTip,"
            + "idLokacija,oibDjelatnik,idKolegij,vrijemeOdrzavanjaDo) "
            + " VALUES(?, ?, ?, ?, ?, ?, ?, ?) ";

        try {
            dbStmnt = dbCon.prepareStatement(sql);

            java.sql.Date datumOdrzavanja;

            datumOdrzavanja= new java.sql.Date(datumOdrz.getTime());

            dbStmnt.setDate(1, datumOdrzavanja);
            dbStmnt.setTime(2, vrijemeOd);
            dbStmnt.setString(3, opis);
            dbStmnt.setInt(4, idTip);
            dbStmnt.setInt(5, idLokacija);
            dbStmnt.setString(6, oibDjelatnik);
            dbStmnt.setString(7, idKolegij);
            dbStmnt.setTime(8, vrijemeOd);
        }
    }
}
```

*Slika 52 Metoda za unos nastave*

### 6.10.2 Unos prisutnosti studenta

Osim unosa nastave koja je održana, korisnik sustava ima mogućnost unijeti i prisutnost pojedinog studenta koji je prisustvovao (ili nije prisustvovao) na određenoj nastavi. Odabirom reda tablice na formi FormaKolegijiNastava i pritiskom gumba Unos prisustva prikazuje se prozor na kojem je moguće označiti prisutnost bilo kojeg od studenata koji su upisani na odabrani kolegij. Prilikom kreiranja unosa nastave u tablici nastava baze podataka, automatski je kreiran i unos dolaska u tablici dolazak za svakog od studenata koji su pridruženi kolegiju unesene nastave. Svaki od tih dolazaka kao zadanu vrijednost prisutnosti ima vrijednost „NE“. Označavanje prisutnosti preko JCheckbox komponente okida poziv metode unosPrisutnosti koja ažurira vrijednost atributa prisutnosti u tablici dolazak na „DA“ (UPDATE). Isto tako ukoliko se komponenta JCheckbox odznači, tada će metoda unosPrisutnosti postaviti atribut prisutnosti u tablici na „NE“. Svaki put nakon ažuriranja vrijednosti atributa prisutan u tablici dolazak, pokreće se i poziv metode izvediProcPostaviBrPrisut koja se nalazi u klasi BazaPodataka. Ova metoda uz ulazne parametre identifikacijskih atributa odabrane nastave i kolegija izvodi proceduru baze podataka koja se zove



postaviBrPrisutnih (slika 53). Ova procedura vrši ažuriranje (UPDATE) vrijednosti atributa broja prisutnih u tablici tako što postavlja vrijednost atributa na rezultat SQL funkcije vratiBrPrisutnih (slika 54). Ova SQL funkcija prima kao ulazni parametar ID nastave, a vraća broj prisutnih koji odgovaraju toj nastavi.

```

1 • CREATE DEFINER='root'@'localhost' PROCEDURE `postaviBrPrisutnih`(IN idN int,IN idK int)
2 BEGIN
3 DECLARE br int;
4 set br = vratiBrPrisutnih(idN);
5 UPDATE nastava
6 set brojPrisutnih = br
7 where idNastava = idN and idKolegij = idK;
8 END

```

*Slika 53 SQL procedura postaviBrPrisutnih*

```

1 • CREATE DEFINER='root'@'localhost' FUNCTION `vratiBrPrisutnih`(
2 idN int
3 ) RETURNS int
4 DETERMINISTIC
5 BEGIN
6
7 DECLARE broj int;
8 SELECT COUNT(prisutan) into broj from dolazak where prisutan = 'DA' and dolazak.idNastava = idN ;
9 return broj;
10 END

```

*Slika 54 SQL funkcija vratiBrPrisutnih*

Dolasci na nastavu				
Jmbag	Prezime	Ime	Vrsta	Prisutan
0246075030	Matić	Boris	REDOVNI	NE
0246075037	Milić	Milan	IZVANREDNI	DA

<b>Milić Milan</b>	<input checked="" type="checkbox"/> <b>Prisutan</b>
--------------------	---

*Slika 55 Dolasci na nastavu s opcijom za odabir prisutnosti studenta*

### 6.10.3 Brisanje unosa nastave

Brisanje nastave iz tablice „Nastava“ u bazi podataka se vrši kroz formu FormaKolegijiNastava odabirom željene nastave i pritiskom na gumb Brisanje. Prije samog brisanja će se pojaviti dijaloški okvir koji korisnika pita je li siguran u svoju odluku o brisanju navedene nastave. Potvrdom brisanja se pozivaju dvije metode klase BazaPodataka. Prva metoda se zove brisanjeDolazaka i ona je zadužena da se obrišu svi dolasci studenata koji su upisani s ID-em nastave koja je odabrana za brisanje. Druga metoda se zove brisanjeNastave i ona briše red same nastave iz tablice nastava. Nakon brisanja nastave, JTable tablica forme se automatski osvježava i prikazuje novo stanje tablice. Implementacija metoda za brisanje dolazaka i nastave je vidljiva na slikama 56 i 57.

```
public void brisanjeDolazaka(String idNastava)
{
    try {
        sql = " DELETE from dolazak where idNastava = ?";

        try {
            dbStmtnt = dbCon.prepareStatement(sql);

            Integer id = Integer.parseInt(idNastava);
            dbStmtnt.setInt(1, id);
```

*Slika 56 Metoda za brisanje dolazaka*

```
public void brisanjeNastave(String idNastava)
{
    try {
        sql = " DELETE from nastava where idNastava = ?";

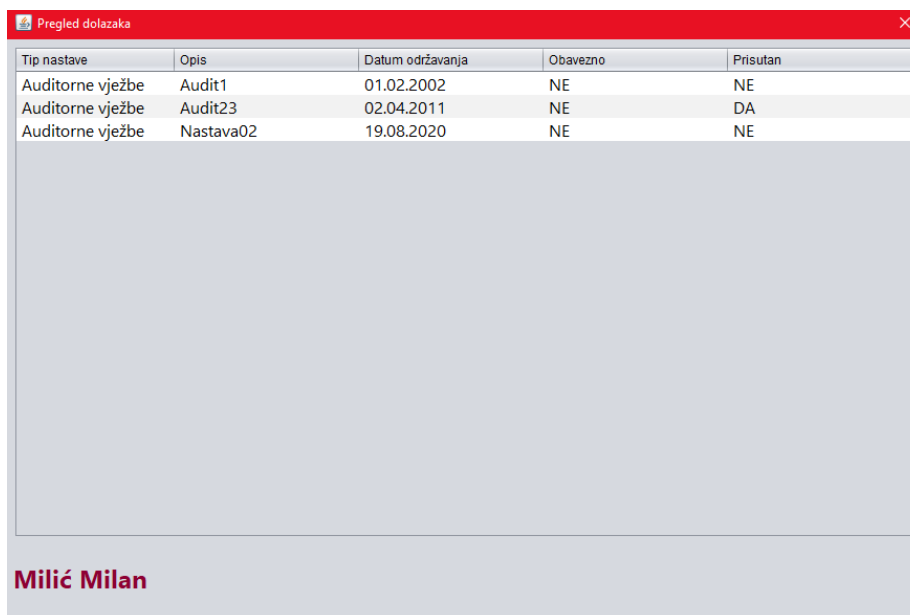
        try {
            dbStmtnt = dbCon.prepareStatement(sql);

            Integer id = Integer.parseInt(idNastava);
            dbStmtnt.setInt(1, id);
```

*Slika 57 Metoda za brisanje nastave*

## 6.11 Forma za pregled dolazaka studenta

Pregled dolazaka pojedinog studenta je posljednja od funkcionalnosti koju je moguće dobiti kroz glavnu formu Academus aplikacije za praćenje nastavnog procesa. Prozor FormaPregledDolazakaStudenti se prikazuje nakon pritiska na gumb Pregled dolazaka koji se nalazi na glavnoj formi. Navedena forma prikazuje u JTable komponenti popis studenata na odabranom kolegiju. Odabir studenata se vrši kao i u ostatku aplikacije, pritiskom na željeni red u tablici. Nakon odabira studenta pritiskom na gumb dolazaka na nastavu se prikazuje dijaloški prozor sa svim nastavama koje su upisane u bazu podataka, a na kojima je student mogao prisustvovati. Tu su prikazani podaci o nastavi, je li nastava obavezna i je li student bio prisutan na odabranoj nastavi. Prozor s tablicom o prisutnosti studenata na nastavi je vidljiv na slici 58.



The screenshot shows a Java Swing window titled "Pregled dolazaka" with a red title bar. Inside the window is a table with five columns: "Tip nastave", "Opis", "Datum održavanja", "Obavezno", and "Prisutan". The table contains three rows of data. Below the table, the name "Milić Milan" is displayed in red text.

Tip nastave	Opis	Datum održavanja	Obavezno	Prisutan
Auditorne vježbe	Audit1	01.02.2002	NE	NE
Auditorne vježbe	Audit23	02.04.2011	NE	DA
Auditorne vježbe	Nastava02	19.08.2020	NE	NE

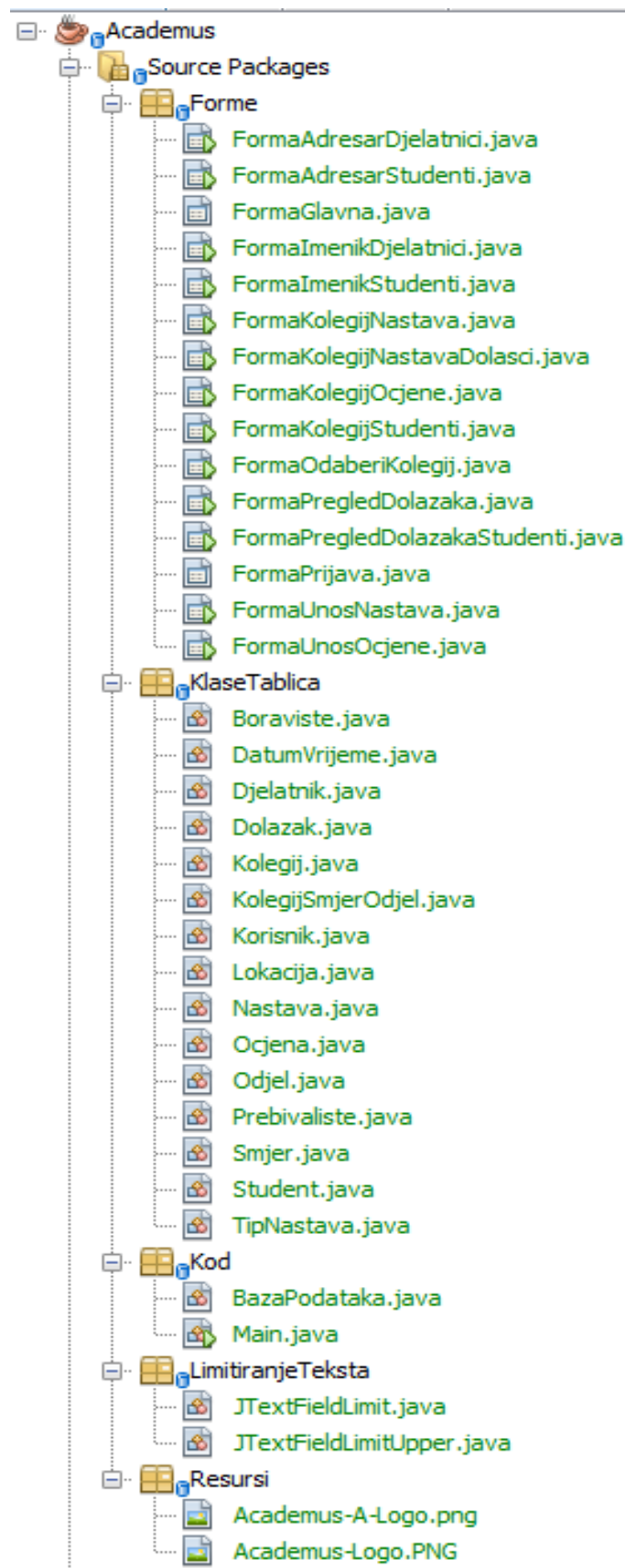
**Milić Milan**

Slika 58 Pregled prisutnosti po nastavi za odabranog studenta

## 6.12 Pogled na organizaciju projekta

Kao što je navedeno, projekt aplikacije Academus je organiziran u pakete koda u kojima se nalaze sve Java klase, a u paketu Resursi i slike koje su potrebne za vizualni dio aplikacije. Na slici 59 su vidljive sve datoteke projekta Academus. Sav kod se nalazi i u Academus repozitoriju GitHub servisa, a poveznica je:

<https://github.com/eljubicic/Academus>.



Slika 59 Projekt Academus

## 7. ZAKLJUČAK

Ovim radom je demonstriran primjer procesa informatizacije jedne ustanove visokog školstva uvođenjem informacijskog sustava. Ovaj informacijski sustav demonstrira prednosti pohrane i manipulacije podacima računalom, te korisniku omogućuje kroz jednostavno i pristupačno sučelje obavljanje svakodnevnih poslovnih procesa na brz i siguran način. Ove karakteristike korištenja informacijskog sustava pokazuju koliko je ovakav sustav bitan za današnje poslovanje bilo poduzeća, bilo ustanove - koja ne samo da želi ostati u koraku s najnovijim trendovima već i koristi priliku da smanji troškove i optimizira poslovanje. Ako neko poduzeće ili ustanova odluči informatizirati poslovanje informacijskim sustavom, uz početni ulog i troškove razvoja će prosperirati dugoročno zadovoljstvom korisnika ali benefitima samog poslovanja. Konkretnim primjerom informacijskog sustava za praćenje nastavnog procesa u visokom školstvu je demonstrirana i upotreba u obrazovanju koje omogućuje da djelatnik ovakve vrste ustanove direktno pristupa pojedinim studentima te prati njihov napredak na jednom centralnom mjestu. Ovaj način omogućuje poboljšanje samog nastavnog procesa i bržeg upoznavanja djelatnika s mogućim izazovima u samom nastavnom procesu kolegija na kojem je izvršitelj. Aplikacija koja je razvijena uspješno ispunjava osnovne postupke u manipulaciji i prikazu podataka studenta i daje uvid u ono što je moguće primijeniti u svakoj obrazovnoj ustanovi na široj razini.

# LITERATURA

- [1] Zagrebački infoacijski centar: „Poduzetnički pojmovnik“, s Interneta, <https://www.zicer.hr/Poduzetnicki-pojmovnik/Informatizacija> , 5.07.2020.
- [2] Leksikografski zavod Miroslav Krleža: „Hrvatska enciklopedija“ (natuknica Informacijski sustav), s Interneta, <https://www.enciklopedija.hr/Natuknica.aspx?ID=27410>, 12.07.2020.
- [3] Gregersen E., „5 Components of information systems“, s Interneta, <https://www.britannica.com/list/5-components-of-information-systems>, 14.07.2020.
- [4] Ackoff R.L., „From data to wisdom“, Journal of Applied systems analysis, 1989., vol. 16, 3 - 9
- [5] Carić T. ; Erdelić T., „Entiteti, veze i ER modeliranje“, s Interneta, <https://www.weboteka.net/fpz/Baze%20podataka/Predavanja/03%20-%20BP%20-%2003.tjedan.pdf>, 27.06.2020.
- [6] Kramberger T. ; Duk S. ; Kovačević R., „Baze podataka“, Tehničko Veleučilište u Zagrebu, 2018., 1 - 28
- [7] Oracle Corporation, „Introduction to Java TM technology“, s Interneta, <https://www.oracle.com/java/technologies/introduction-to-Java.html>, 14.07.2020.
- [8] JavaTPoint, „Java Swing“, s Interneta, <https://www.javatpoint.com/java-swing>, 21.07.2020.
- [9] Tutorialspoint, „Swing“, s Interneta, [https://www.tutorialspoint.com/swing/swing\\_containers.htm#:~:text=Containers%20are%20an%20integral%20part,add%20a%20component%20to%20itself.&text=Sub%20classes%20of%20Container%20are,%2C%20JPanel%2C%20JFrame%20and%20JWindow.](https://www.tutorialspoint.com/swing/swing_containers.htm#:~:text=Containers%20are%20an%20integral%20part,add%20a%20component%20to%20itself.&text=Sub%20classes%20of%20Container%20are,%2C%20JPanel%2C%20JFrame%20and%20JWindow.), 21.07.2020.
- [10] But C., „Ant vs Maven vs Gradle“, s Interneta, [https://medium.com/@Colin\\_But/ant-vs-maven-vs-gradle-801fde21af80](https://medium.com/@Colin_But/ant-vs-maven-vs-gradle-801fde21af80), 21.07.2020.
- [11] Chatham M., „SQL languages“, „Structured Query Language by example – volume I“, 2012., 8
- [12] Pavlić M., „Oblikovanje baza podataka“, Sveučilište u Rijeci, 2011., 25 – 32

- [13] Varga M., „Baze podataka: Konceptualno, logičko i fizičko modeliranje podataka“, „DRIP“, 1994., 40 – 50
- [14] Geeks for geeks, „Garbage collection in Java“, s Interneta, <https://www.geeksforgeeks.org/garbage-collection-java/>, 22.07.2020.
- [15] CodeMonk, „What are four principles of OOP?“, s Interneta, <https://www.javacodemonk.com/what-are-four-principles-of-oop-how-aggregation-is-different-than-composition-5b534baf>, 23.07.2020.
- [16] Tutorials Point, „Java object and classes“, s Interneta, [https://www.tutorialspoint.com/java/java\\_object\\_classes.htm](https://www.tutorialspoint.com/java/java_object_classes.htm), 23.07.2020.
- [17] Tutorials Point, „Java access modifiers“, s Interneta, [https://www.tutorialspoint.com/java/java\\_access\\_modifiers.htm](https://www.tutorialspoint.com/java/java_access_modifiers.htm), 23.07.2020.
- [18] Tutorials Point, „Java inheritance“, s Interneta, [https://www.tutorialspoint.com/java/java\\_inheritance.htm](https://www.tutorialspoint.com/java/java_inheritance.htm), 23.07.2020.
- [19] Tutorials Point, „Java abstraction“, s Interneta, [https://www.tutorialspoint.com/java/java\\_abstraction.htm](https://www.tutorialspoint.com/java/java_abstraction.htm), 23.07.2020.
- [20] Tutorials Point, „Java variables“, s Interneta, [https://www.tutorialspoint.com/java/java\\_variable\\_types.htm](https://www.tutorialspoint.com/java/java_variable_types.htm), 23.07.2020.
- [21] Tutorials Point, „Java methods“, s Interneta, [https://www.tutorialspoint.com/java/java\\_methods.htm](https://www.tutorialspoint.com/java/java_methods.htm), 23.07.2020.
- [22] Tutorials Point, „Java polymorphism“, s Interneta, [https://www.tutorialspoint.com/java/java\\_polymorphism.htm](https://www.tutorialspoint.com/java/java_polymorphism.htm), 23.07.2020.
- [23] Minh N. H., „Java getter and setter tutorial - from basics to best practices“, s Interneta, <https://www.codejava.net/coding/java-getter-and-setter-tutorial-from-basics-to-best-practices>, 23.07.2020.

