Implementacija baze podataka za potrebe autoškole u sustavu MySQL

Jović, Filip

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Jović

IMPLEMENTACIJA BAZE PODATAKA ZA POTREBE AUTOŠKOLE U SUSTAVU MYSQL

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Jović

Matični broj: 45149/16–R

Studij: Poslovni sustavi

IMPLEMENTACIJA BAZE PODATAKA ZA POTREBE AUTOŠKOLE U SUSTAVU MYSQL

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Rabuzin Kornelije

Varaždin, rujan 2020.

Filip Jović

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Rad se bavi opisom karakteristika sustava MySQL, te implementacijom baze podataka koja se koristi za potrebe autoškole. Prikazana je izrada baze podataka i jednostavne CRUD aplikacije za upravljanje bazom.

Ključne riječi: baza podataka; MySQL; autoškola; sustav za upravljanje bazom podataka;

Sadržaj

1. Uvod

Baze podataka sastavni su dio svake organizacije, poduzeća i institucije. Baze podataka olakšavaju svakodnevne poslove i od velike su pomoći pri vođenju i obavljanju bilo kojeg posla. Danas postoji veliki broj sustava za upravljanje bazom podataka. Jedan od poznatijih svakako je i MySQL, a o njemu će biti govora i u ovom radu. Još neki poznatiji sustavi za upravljanje bazom podataka su: Oracle, PostgreSQL, MS SQL Server itd. Cilj ovog rada je prikazati izradu baze podataka za potrebe autoškole u sustavu MySQL. Također, prikazat ćemo izradu aplikacije za razvijenu bazu. Aplikacija je izrađena pomoću HTML-a i PHP-a.

U radu će biti govora o MySQL-u koji će biti opisan, također dotaknuti ćemo se i povijesti MySQL sustava, ali i kako se instalira, te njegovog vizualnog sučelja MySQL Workbench. Također dotaknuti ćemo se i InnoDB engine-a za pohranu koji je zadani engine za sustav MySQL. Za kraj spomenuti ćemo naredbe i usporediti MySQL veziju 8.0 s prijašnjom verzijom.

U idućem poglavlju ćemo pojasniti bazu za potrebe autoškole. Prikazat će se njeno kreiranje, ERA model. Pojasnit ćemo sve tablice, veze i okidače koje baza podataka sadrži. Izvršiti ćemo jednostavne, ali i složene upite nad bazom i prikazati poglede.

U zadnjem poglavlju prikazati ćemo malu aplikaciju koja je kreirana kako bismo mogli upravljati bazom podataka za autoškolu. Aplikacija nije primarni dio ovog rada, pa nije objašnjena u potpunosti.

2. Metode i tehnike rada

U ovom poglavlju biti će opisane sve tehnologije koje su korištene u izradi ovog rada tj. baze podataka i aplikacije. Na izradu samog rada potaknule su me knjige „Uvod u SQL“ i „SQL – napredne teme“ čiji je autor Prof. dr. sc. Rabuzin Kornelije. Također veliki dio rada je baziran na „MySQL 8.0 Reference Manual“ priručniku koji je dostupan na stranicama MySQL-a.

MySQL Workbench

Mysql Workbench je vizualni alat za modeliranje baze podataka, razvoj sql-a i administraciju baze. Tokom izrade ovog rada korišten je MySQL Workbench 8.0.18. Workbench će biti detaljnije opisan u nastavku rada.

HTML5

HTML5 je jezik koji se koristi za prikaz sadržaja na World Wide Webu. HTML5 je najnovija verzija i objavljen je 2014. godine.

HTML je korišten za kreiranje male web aplikacije s kojom možemo upravljati bazom podataka.

PHP

PHP je programski jezik namijenjen izradi dinamičnih web stranica. PHP podržava razne baze podataka i Internet protokole.

Pomoću PHP-a web aplikacija se spaja na bazu.

XAMPP

XAMPP je open-source platforma za razvoj PHP-a, a omogućuje lokalno podizanje testnog poslužitelja. Za lokalno izvršavanje korišten je alat XAMPP pomoću kojeg je razvijena mala web aplikacija za bazu koju smo kreirali u radu.

3. MySQL

U ovom poglavlju obraditi ćemo MySQL. Ovo poglavlje je jedno od ključnih poglavlja ovog rada jer je u MySQL-u razvijena baza za autoškolu koju ovaj rad obrađuje. Proći ćemo kroz povijest MySQL-a, karakteristike MySQL-a itd.

Za početak ćemo odgovoriti na pitanje što je to MySQL. MySQL je open-source sustav za upravljanje relacijskim bazama podataka. Relacijska baza podataka organizira podatke u jednoj ili više tablica. SQL se koristi za kreiranje, obradu i pregled podataka u relacijskoj bazi podataka. MySQL je razvijen u C i C++ jeziku i zbog toga može raditi na više različitih platformi. Jedan od razloga velike popularnosti MySQL-a je taj što postoje klijenti za sve popularne jezike kao što su: C, C++, Java, Perl, PHP, Python i td. MySQL se nudi u dvije izvedbe: MySQL Community Server i MySQL Enterprise Server. Glavna razlika između te dvije verzije je ta da je Community Server besplatan, a uz Enterprise Server se dobije 24/7 podrška od strane ORACLE-a, pa zbog toga se eventualni problemi koji nastanu rješavaju jako brzo. Kada dođe do problema s Community Server verzijom rješenje se može pronaći na jednom od mnogobrojnih foruma, ali, kao što znamo, vrijeme je novac i s Enterprise verzijom dobivamo dozu sigurnosti i manje gubitke u poslovanju. [1], [2]

Slika 1: MySQL logo

MySQL je dostupan za besplatno preuzimanje na službenim stranicama MySQL-a. Za operacijski sustav Windows dostupan je MySQL installer koji znatno olakšava konfiguraciju i instalaciju. MySQL installer u paketu sadrži MySQL Server, MySQL Workbench, MySQL Shell, MySQL Router, MySQL for Visual Studio, MySQL for Excel i MySQL Notifier. Jedini zahtjev je da računalo ima instaliran Microsoft .NET Framework 4.5.2. ili novije, u slučaju da nema on je dostupan besplatno na Microsoft-ovim stranicama. Naravno MySQL nije dostupan samo na Windows operacijskom sustavu, nego i na raznim verzijama Linux-a i macOS. [3]

3.1. Povijest MySQL-a

MySQL je kreiran od strane Švedske kompanije MYSQL AB čiji su osnivači David Axmark, Allan Larsson i Michael Widenius. Prva verzija MySQL-a pojavila se 1995. godine.

MySQL je dobio ime po jednom od osnivača tj. po Michaelu „Monty“ Wideniusu, što nam govori da „My“ dio naziva dolazi iz nadimka plus SQL. 2000. godine MySQL je prešao pod open-source licencu, a do kraja 2001. godine MySQL je dosegnuo 2 milijuna aktivnih instalacija. Iako je MySQL pod open-source licencom,ORACLE je vlasnik još od 2010. godine kada su kupili Sun Microsystems koji je do tada bio vlasnik MySQL-a. Iste te godine Michale Widenius napušta Sun MicroSystems i počinje razvijati MariaDB. MariaDB API je potpuno kompatibilan s MySQL-om. MySQL je kroz godine postojanja prošao kroz mnogo verzija od kojih je zadnja 8.0 i koja se koristi u izradi ovog rada. [2]

3.2. MySQL naredbe

Naredbe za upravljanje podatcima u SQL jeziku možemo svrstati u tri kategorije tj. pod DDL, DML i ostali.

DDL tj. Data Definition Language se bavi shemama i opisima baze podataka tj. o tome kako bi podatci trebali biti zapisani u bazi podataka. Pod DDL spadaju naredbe „CREATE“, „ALTER“, „DROP“, „TRUNCATE“, „COMMENT“ i „RENAME“. Dakle sve ove naredbe se bave samom strukturom baze podataka.

DML tj. Data Manipulation Language se bavi obradom tj. unosom, pohranom, ažuriranjem i brisanjem podataka u bazi podataka. Naredbe koje spadaju pod DML su „SELECT“, „INSERT“, „UPDATE“, „DELETE“ i td.

Pod ostale možemo svrstati DCL tj. Data Control Language i TCL tj. Transaction Control Language.

DCL se bavi dozvolama, privilegijama i davanjem tj. oduzimanjem kontrole nad bazom podataka. Naredbe koje spadaju pod DCL su „GRANT“ i „REVOKE“.

TCL se bavi transakcijama unutar baze podataka. Naredbe koje spadaju pod TCL su „COMMIT“, „ROLLBACK“, „SAVEPOINT“ i „SET TRANSACTION“. [4]

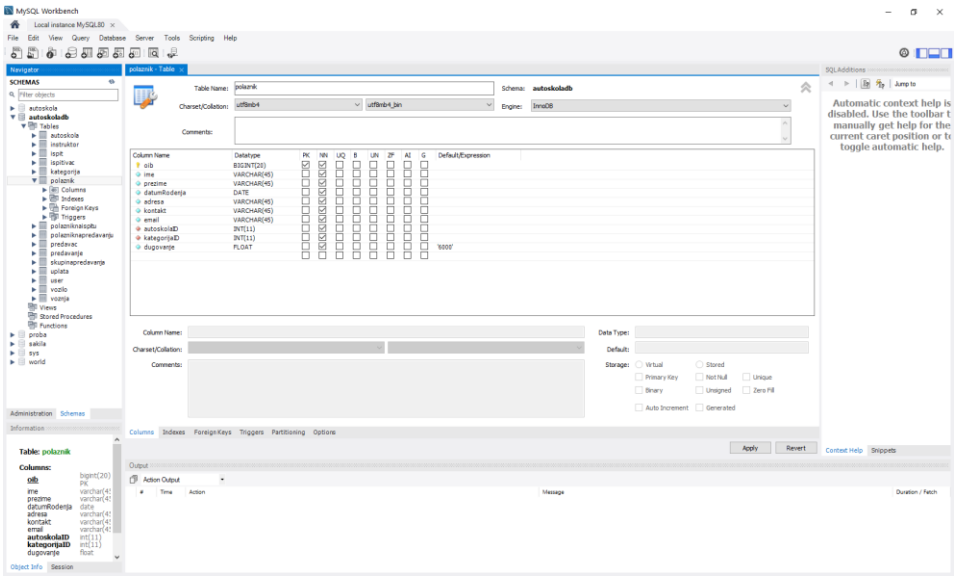
3.3. MySQL Workbench

MySQL Workbench je grafički alat za rad s MySQL poslužiteljima i bazama podataka. Workbench je nasljednik DBDesigner-a 4 i zamjena je za MySQL GUI Tools Bundle. Prva verzija Workbench-a izašla je 2005. godine, a 2007. godine je odlučeno da će Workbench biti

MySQL-ovo „flagship“ grafičko sučelje. Funkcionalnosti MySQL Workbench-a možemo podijeliti:

- Pomoću ugrađenog SQL editora omogućuje izvršavanje raznih SQL upita, omogućuje kreiranje baze i veza itd.
- Modeliranje podataka: Omogućuje grafičko kreiranje sheme baze podataka, reverse i forward engineer između sheme i baze podataka. Table editor pruža jednostavno upravljanje i kreiranje tablica, stupaca, indexa, triggera itd.
- Administracija poslužitelja: Omogućuje administriranje instanci MySQL servera, izvođenje sigurnosnih kopija, pregled nad stanja baze podataka i praćenje performansi MySQL servera.
- Migracija podataka: Migracija podataka omogućuje migraciju s Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, PostgreSQL i drugih RDBMS tablica, objekata, i podataka u MySQL. Migracija također podržava migraciju s prijašnjih verzija MySQL-a na najnoviju.

MySQL Workbench dolazi u tri verzije od kojih je jedna besplatna. MySQL Workbench Community Edition je besplatna verzija, dok su MySQL Workbench Standard Edition i Enterprise Edition komercijalne izvedbe. Razlika između besplatne verzije i komercijalnih verzija je ta što besplatna verzija nema validaciju modela i sheme, automatsku dokumentaciju, grafičko sučelje za MySQL Enterprise Backup, grafičko sučelje za MySQL Enterprise Audit, grafičko sučelje za MySQL Enterprise Firewall. [5]



Slika 2: MySQL Workbench

3.4. InnoDB

InnoDB je zadani engine za pohranu podataka u sustavu MySQL 8.0, koji balansira visoku pouzdanost s visokim performansama. InnoDB je zamijenio MyISAM engine koji je bio zadani engine za MySQL do verzije 5.5. Prilikom kreiranja tablice, dodavanjem klauzule „ENGINE=“ moguće je promijeniti engine tj. Umjesto innoDB odabrati MyISAM. DML operacije u innoDB engine-u su ACID kompatibilne. ACID model je set principa u dizajnu baze podataka koji naglašavaju pouzdanost, koja je vrlo bitna za podatke koje spremaju firme. **A**tomicity, **C**onsistency, **I**solation i **D**urability.

Eng. Atomicity tj. atomnost je pravilo zbog kojeg se sve izmjene u bazi podataka moraju izvršavati na principu „sve ili ništa“, to znači da svaka transakcija mora biti izvršena u potpunosti, u suprotnom se vraća na stanje prije početka izvođenja transakcije.

Eng. Consistency tj. dosljednost je pravilo koje određuje da se u bazu podataka spremaju samo valjani podatci. Ako se prekrši pravilo dosljednosti, transakcija se prekida i vraća se stanje prije početka izvođenja same transakcije.

Eng. Isolation tj. izolacija je pravilo koje određuje da ni jedna transakcija koja se izvodi istovremeno ne utječe na izvođenje drugih transakcija. To znači da ako dvije osobe izvode transakcije na istoj bazi podataka u isto vrijeme, obje transakcije trebaju raditi na bazi podataka na izoliran način tj. jedna transakcija mora biti obavljena prije druge ili obrnuto.

Eng. Durability tj. trajnost je pravilo pomoću kojeg se osigurava da počinjene transakcije u bazi ne budu izgubljene tj. pomoću sigurnosnih kopija osigurava obnavljanje baze unatoč eventualnim greškama koje se mogu desiti na samom hardveru ili softveru. [6]

3.5. Usporedba MySQL 8.0 i MySQL 5.7

MySQL 8.0. donosi dosta promjena i poboljšanja u odnosu na MySQL 5.7. Jedno od glavnih poboljšanja su bolje performanse. A poboljšanja koja se vežu uz performanse:

- MySQL 8.0. uključuje rječnik podataka koji sadrži informacije o transakcijama.
- Atomni DDL izvještaj kombinira ažuriranja rječnika podataka, operacije engine-a za pohranu i binary log koji je povezan s DDL operacijama u jednu transakciju.
- MySQL server automatski pri pokretanju obavlja sve potrebne nadogradnje nad tablicama i MySQL shemama, ali i objektima u drugim shemama kao što su

„sys schema“ i „user schema“. Administrator baze podataka više ne mora sam izazvati „mysql_upgrade“.

- Zadani character set više nije latin1 nego utf8mb4.

Također poboljšana je i sigurnost. Neka od poboljšanja koja su vezana uz sigurnost su:

- Grant tablice u MySQL bazi podataka se spremaju pomoći InnoDB tablica

- Autentikacija se u MySQL 8.0. obavlja pomoću novog „caching_sha2_password“ plugin autentifikatora. Donosi sigurniju i efikasniju enkripciju lozinaka od „mysql_native_password“ plugina.

- MySQL od verzije 8.0 podržava uloge. Uloge mogu biti dodijeljeni korisnicima, ali se mogu i oduzeti.

- MySQL od vezije 8.0. zadržava informacije o prijašnjim lozinkama, te omogućava korištenje prijašnjih lozinaka. Također omogućava administratorima kontroliranje broja pogrešnog unosa lozinke prije nego što se zaključa profil.

Poboljšanja i promjene nisu zaobišli ni InnoDB.

- „auto-increment counter“ vrijednost se od 8.0. verzije sprema u „redo log“ svaki put kada se vrijednost promijeni. Zbog toga maksimalna „auto-increment“ vrijednost postaje ista nakon svakog ponovnog pokretanja.

- „innodb_deadlock_detect“ je nova varijabla pomoću koje se može razriješiti deadlock situacija.

Naravno kako se sustav poboljšava neke opcije su zastarjele i ne bi se više trebale koristiti.

- Umjesto „utf8mb3“ character set-a trebalo bi se koristiti „utf8mb4“.

- „ENGINE“ klauzula za „DROP TABLESPACE“ i „ALTER TABLESPACE“ je zastarjela

- „AUTO_INCREMENT“ podrška za „FLOAT“ i „DOUBLE“ je zastarjela i te tipove podataka bi trebalo zamijeniti sa integer tipom.

- Sintaksa za „FLOAT(M,D)“ i „DOUBLE(M,D)“ koja specificira broj znamenaka za „FLOAT“ i „DOUBLE“ vrijednosti.

Također postoji dosta funkcija koje su uklonjene dolaskom verzije 8.0.. Nabrojati ćemo i neke od njih.

- „innodb_locks_unsafe_for_binlog“ varijabla je uklonjena jer „READ COMMITTED“ pruža sličnu funkcionalnost.

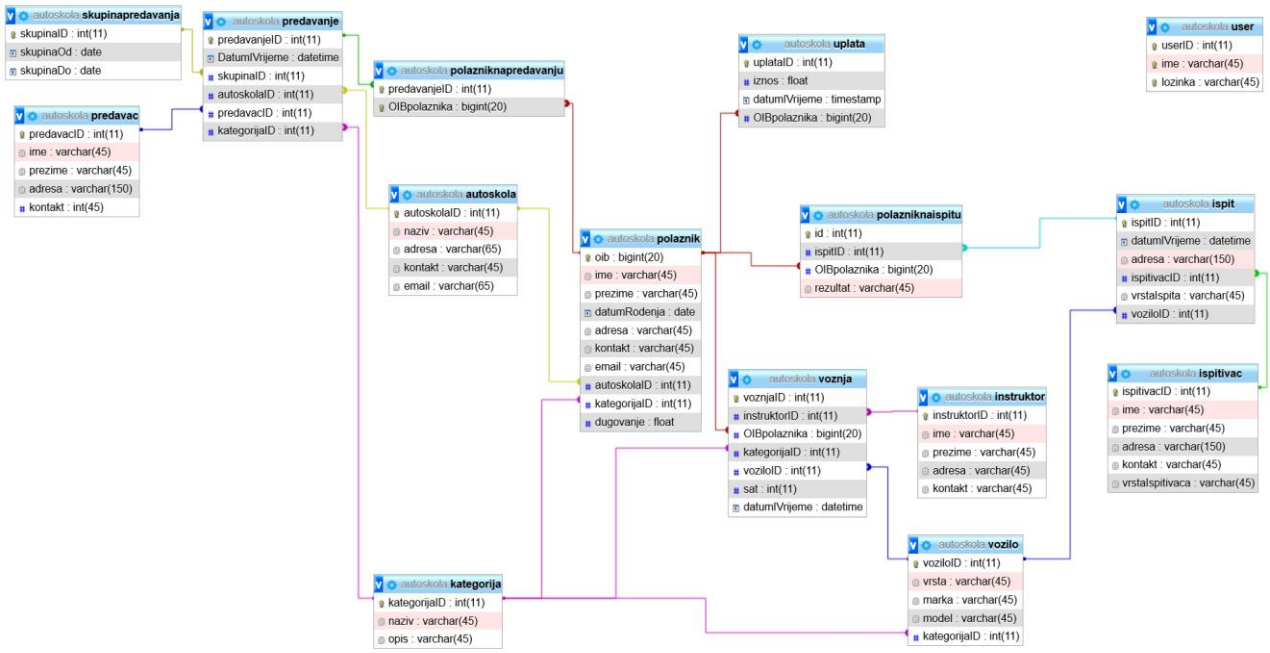
- Korištenje „GRANT“ funkcije za kreiranje korisnika umjesto „CREATE USER“

- Uklonjeno je „ASC“ i „DESC“ sortiranje za klauzulu „GROUP BY“. Upiti koji to sadrže mogu vratiti drugačiji rezultat nego u verziji 5.7. Za sortiranje uz „GROUP BY“ najbolje je koristiti „ORDER BY“. [7]

4. Model baze podataka

U svrhu izrade aplikacije za autoškolu izrađena je baza podataka. U ovom poglavlju će biti opisana baza podataka, tablice i veze te ćemo prikazati ERA model. ERA model prikazuje našu bazu i sve veze između tablica.

4.1. ERA model



Slika 3: ERA model

Ovdje imamo prikazan ERA model baze podataka za autoškolu. Prikazano je 15 tablica koje su međusobno povezane, a čije veze će kasnije biti objašnjene. Savaka tablica ima svoj naziv i prikaz kojoj shemi tj. bazi pripada. U ovom primjeru sve tablice pripadaju istoj shemi. Svaka tablica ima primarni ključ, a jedna tablica „polazniknapredavanju“ ima dvokomponentni primarni ključ. Primarni ključevi su označeni malim zlatnim ključićem s lijeve strane atributa koji je primarni ključ. Vanjski ključevi na modelu nisu označeni, ali ćemo ih nabrojati kada budemo opisivali tablice i veze. Vanjski ključ iz ERA modela možemo prepoznati preko prikazanih veza. S desne strane svakog atributa možemo vidjeti koji je tip podatka tog atributa, ali ćemo to u nastavku dodatno nabrojati tj. opisati.

4.2. Poslovna pravila

U ERA modelu se nalazi 15 tablica koje su međusobno povezane sa 16 veza od kojih su dvije više prema više. Zbog lakšeg praćenja veza po ERA modelu veze su opisane s lijeva na desno.

Veze:

1. Jedan ispitivač ispituje na više ispita, dok jedan ispit ima samo jednog ispitivača
2. Jedan ispit može imati više polaznika, dok jedan polaznik može biti na više ispita
3. Jedan ispit koristi samo jedno vozilo, dok se jedno vozilo može koristiti na više ispita
4. Jedan predavač može predavati na više predavanja, dok jedno predavanje ima samo jednog predavača
5. Jedno predavanje spada u jednu skupinu predavanja, dok jedna skupina predavanja sadrži više predavanja
6. Na jednom predavanju može biti više polaznika, dok jedan polaznik može biti na više predavanja
7. Jedno predavanje se održava u jednoj autoškoli, dok jedna autoškola može održati više predavanja
8. Jedno predavanje se održava za jednu kategoriju, dok jedna kategorija ima više predavanja
9. Jedna autoškola ima više polaznika, dok jedan polaznik pripada samo jednoj autoškoli
10. Jedan polaznik polaže za jednu kategoriju, dok za jednu kategoriju polaže više polaznika
11. Jedan polaznik može izvršiti više uplata, dok je jedna uplata za samo jednog polaznika
12. Jedan polaznik može ići na više vožnji, dok jedna vožnja može biti samo za jednog polaznika
13. Jedno vozilo može biti za samo jednu kategoriju, dok jedna kategorija ima više vozila
14. Jedna vožnja može biti za samo jednu kategoriju, dok jedna kategorija može imati više vožnji
15. Jedna vožnja može koristiti samo jedno vozilo, dok jedno vozilo može biti korišteno na više vožnji
16. Jedan instruktor može biti na više vožnji, dok jedna vožnja ima samo jednog instruktora

4.3. Tablice

Budući da smo u prošlom dijelu opisali sve veze sada dolazi red na tablice. Ovdje ćemo opisati sve tablice tj. njihove atribute i ograničenja. Tablice će biti opisane kao i veze, s lijeva na desno, zbog lakšeg praćenja.

Tablica 1: Opis svih tablica

skupinapredavanja	• skupinaID (int, PK, AI) • skupinaOd (date, NN) • skupinaDo (date, NN)
predavac	• predavacID (int, PK, AI) • ime (varchar(45), NN) • prezime (varchar (45), NN) • adresa (varchar (150), NN) • kontakt (int, NN)
ispit	• ispitID (int, PK, AI) • datumIVrijeme (datetime, NN) • adresa (varchar(150), NN) • ispitivacID (int, NN, references ispitivac (ispitivacID)) • vrstaIspita (varchar(45), NN) • voziloID (int, references vozilo (voziloID))
ispitivac	• ispitivacID (int, PK, AI) • ime (varchar(45), NN) • prezime (varchar(45), NN) • adresa (varchar(150), NN) • kontakt (varchar(45), NN) • vrstaIspitivaca (varchar(45), NN)
	• predavanjeID (int, PK, AI) • DatumIVrijeme (datetime, NN) • skupinaID (int, references skupinapredavanja (skupinaID))

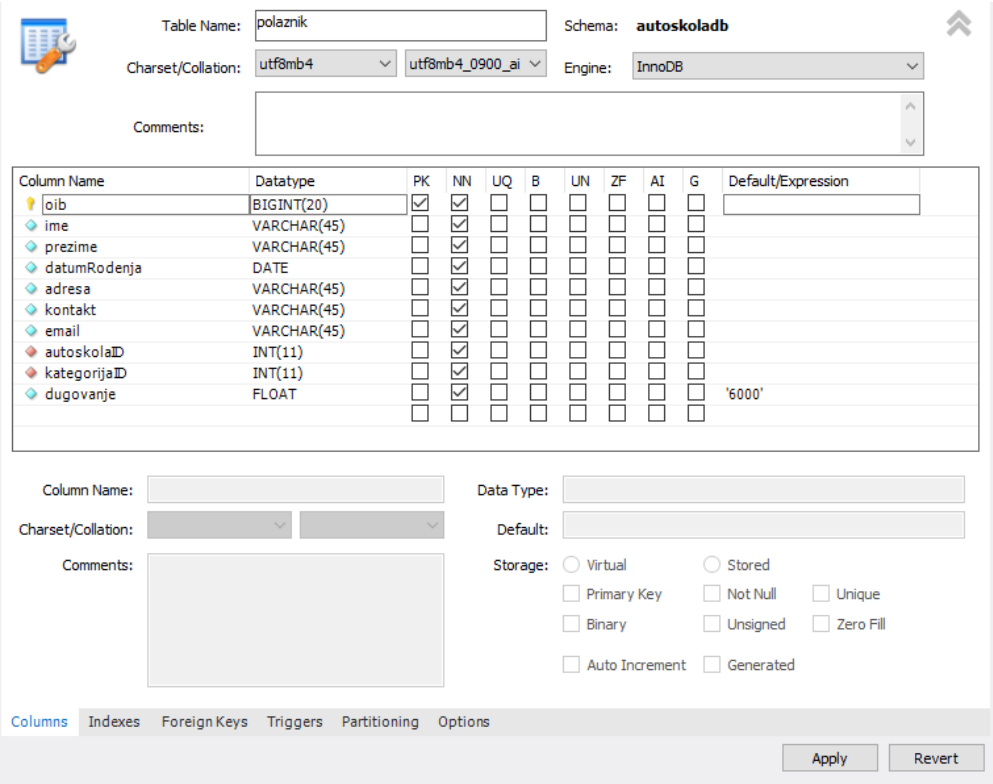
predavanje	• autoskolaID (int, references autoskola (autoskolaID)) • predavacID (int, references predavac (predavacID)) • kategorijaID (int, references kategorija (kategorijaID))
polazniknaispitu	• id (int, PK, AI) • ispitID (int, references ispit (ispitID)) • OIBpolaznika (bigint, references polaznik (oib)) • rezultat (varchar(45))
polazniknapredavanju	• predavanjeID (int, PK, references predavanje (predavanjeID)) • OIBpolaznika (bigint PK, references polaznik (oib))
autoskola	• autoskolaID (int, PK, AI) • naziv (varchar(45), NN) • adresa (varchar(65), NN) • kontakt (varchar(45), NN) • email (varchar(65), NN)
polaznik	• oib (bigint, PK) • ime (varchar(45), NN) • prezime (varchar(45), NN) • datumRodenja (date, NN) • adresa (varchar(45), NN) • kontakt (varchar(45), NN) • email (varchar(45), NN) • autoskolaID (int, references autoskola (autoskolaID)) • kategorijaID (int, references kategorija (kategorijaID)) • dugovanje (float, NN, default="6000")
	• kategorijaID (int, PK, AI) • naziv (varchar(45), NN)

kategorija	• opis (varchar(45), NN)
vozilo	• voziloID (int, PK, AI) • vrsta (varchar(45), NN) • marka (varchar(45), NN) • model (varchar(45), NN) • kategorijaID (int, references kategorija (kategorijaID))
uplata	• uplataID (int, PK, AI) • iznos (float, NN) • datumIVrijeme (timestamp, NN, default „current_timestamp“) • OIBpolaznika (bigint, references polaznik(oib))
voznja	• voznjaID (int, PK, AI) • instruktorID (int, references instruktor (instruktorID)) • OIBpolaznika (bigint, references polaznik (oib)) • kategorijaID (int, references kategorija (kategorijaID)) • voziloID (int, references vozilo (voziloID)) • sat (int, NN) • datumIVrijeme (datetime, NN)
instruktor	• instruktorID (int, PK, AI) • ime (varchar(45), NN) • prezime (varchar(45), NN) • adresa (varchar(45), NN) • kontakt (varchar(45), NN)
user	• userID (int, PK, AI) • ime (varchar(45), NN, UQ) • lozinka (varchar(45), NN, UQ)

4.4. Kreiranje baze

Baza podataka za potrebu autoškole kreirana je pomoću alata MySQL Workbench 8.0.18..

Tablice u MySQL Workbench-u kreiramo tako da u navigatoru s lijeve strane odaberemo našu novu schemu i unutar nje odaberemo gumb „create new table“. Tada nam se otvara prozor u koji možemo unijeti naziv tablice, atribute, ograničenja atributa, ali i vanjske ključeve, indekse i ograničenja koji će biti objašnjeni kasnije.



Slika 4: Kreiranje tablice

Na slici se vidi kreiranje tablice „polaznik“. U stupac „Column name“ dodajemo atribute. Stupac „Datatype“ je tip podatka. Nadalje imamo ograničenja i „Default“ stupac u kojem definiramo vrijednost koja se unosi ako korisnik ne unese ništa za taj atribut. U idućoj tablici ćemo pokazati tipove podataka koje podržava MySQL 8.0.

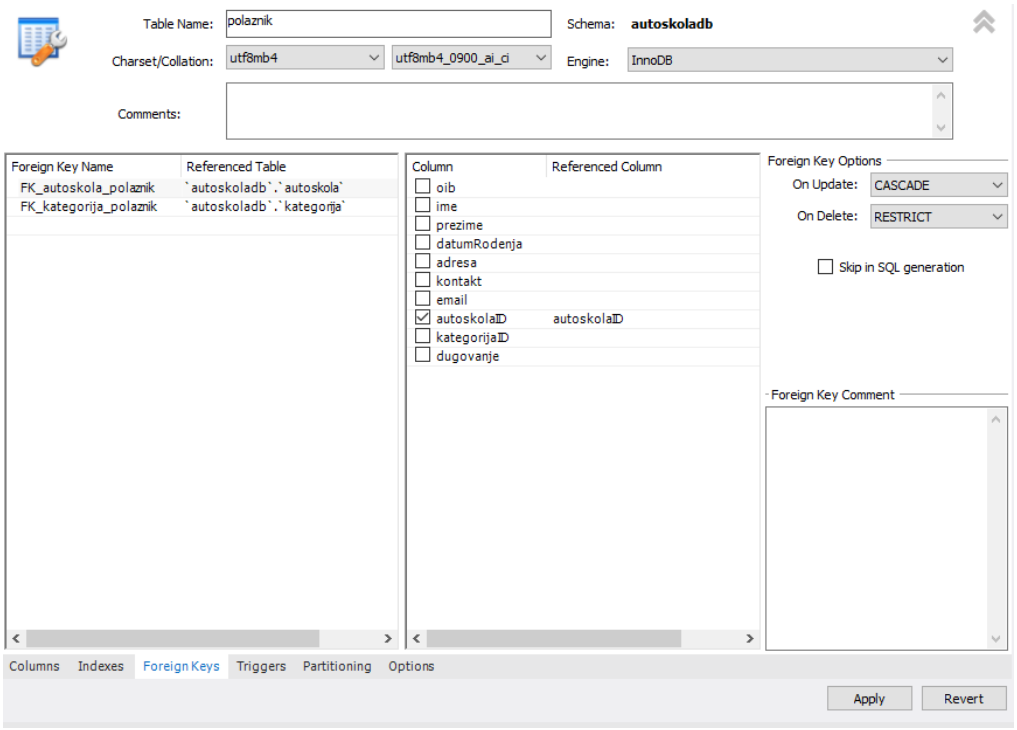
Tablica 2: Tipovi podataka u MySQL-u

Tip podatka	Opis
CHAR(N)	Niz znakova fiksne dužine
VARCHAR(N)	Niz znakova varijabilne dužine, N je limit
BINARY(N)	Jednako kao CHAR, ali pohranjuje binarni byte string.
VARBINARY(N)	Jednako kao VARCHAR, ali pohranjuje binarni byte string.
TINYBLOB	Koristi se za BLOB (Binary Large OBjects) maksimalne dužine 255 byta
TINYTEXT	Sadrži string maksimalne dužine 255 znakova
TEXT(N)	Sadrži string maksimalne dužine 65,535 byta
BLOB(N)	Koristi se za BLOB (Binary Large Objects) maksimalne dužine 65,535 byta
MEDIUMTEXT	Sadrži string maksimalne dužine 16.777.215 znakova
MEDIUMBLOB	Koristi se za BLOB (Binary Large Objects) maksimalne dužine 16,777,215 byta
LONGTEXT	Sadrži string maksimalne dužine 4,294,967,295 znakova
LONGBLOB	Koristi se za BLOB (Binary Large Objects) maksimalne dužine 4,294,967,295 byta
ENUM(vrijednost1, vrijednost2, …)	String tip podatka koji sadrži samo jednu vrijednost, koja se može odabrati sa liste vrijednosti. ENUM lista može imati do 65535 vrijednosti. Ako se unese vrijednost koja nije na listi, biti će unesena blank vrijednost
SET(vrijednost1, vrijednost2, …)	String tip podatka koji sadrži 0 ili više vrijednosti, koje su navedene u SET listi. SET lista sadrži do 64 vrijednosti.
BIT(N)	Tip podatka koji sadrži Bit vrijednost. Broj bitova po vrijednosti specificira (N), od 1 do 64.
TINYINT(N)	Vrlo mali integer tip podatka. Raspona -128 do 127 ili od 0 do 255.

BOOL	Sadrži 0 ili 1. Nulu smatramo kao „Laž“, Jedan smatramo kao „Istina“.
BOOLEAN	Jednak kao i BOOL.
SMALLINT(N)	Mali integer tip podatka. Raspona -32768 do 32767 ili od 0 do 65535.
MEDIUMINT(N)	Srednji integer tip podatka. Raspona – 8388608 do 8388607 ili od 0 do 16777215.
INT(N)	integer tip podatka. Raspona – 2147483648 do 2147483647 ili od 0 do 4294967295.
INTEGER(N)	Jednak kao i INT.
BIGINT(N)	Veliki integer tip podatka. Raspona -9223372036854775808 do 9223372036854775807 ili od 0 do 18446744073709551615.
FLOAT(N)	Float tip podatka, tj. decimalni broj. Veličina se definira pomoću (N).
DOUBLE(N,D)	Kao i float sprema decimalni broj, pomoću (D) određujemo broj decimala nakon decimalne točke.
DECIMAL(N,D)	Sprema decimalni broj s točno definiranim brojem decimala prije decimalne točke (N) i točno definiranim brojem decimala nakon decimalne točke (D). Maksimalna veličina (N) je 65 i 35 za (D).
DEC(N,D)	Jednako kao i DECIMAL
DATE	Pohranjuje datum. Format: YYY-MM-DD. Raspona od 1000-01-01 do 9999-12-31.
TIME	Pohranjuje vrijeme. Format: hh:mm:ss. Raspona od -838:59:59 do 838:59:59.
DATETIME	Pohranjuje DATE i TIME kombinaciju. Format: YYYY-MM-DD hh:mm:ss. Raspona od 1000-01-01 00:00:00 do 9999-12-31 23:59:59.
TIMESTAMP	TIMESTAMP vrijednosti se pohranjuju kao broj sekundi koji su prošle od UNIX epohe. Format: YYYY-MM-DD hh:mm:ss. Raspona od 1970-01-01 00:00:01 do 2038-01-09 03:14:07.
YEAR	YEAR je četveroznamenkasti format. Raspon je od 1901 do 2155 i 0000.

(izvor: w3schools [8])

U MySQL Workbench-u vanjske ključeve možemo kreirat preko istog prozora u kojem smo kreirali tablicu. Na dnu, klikom na tab „Foreign Keys“ otvara nam se novi prozor i kojem možemo dodati novi vanjski ključ.



Slika 5: Kreiranje vanjskog ključa

Slika 5. prikazuje prozor u kojem kreiramo vanjski ključ. U stupcu odaberemo „Foreign Key Name“ zadamo ime vanjskog ključa. U stupcu „Referenced Table“ odaberemo tablicu na koju se veže vanjski ključ. Stupac „Column“ prikazuje atribute iz tablice u kojoj kreiramo vanjski ključ i tu biramo atribut koji postaje vanjski ključ. U stupcu „Reference Column“ biramo na koji atribut se referenciramo u tablice na koju se spajamo. Na kraju imamo „Foreign Key Options“ gdje biramo što će se dogoditi kada vršimo „UPDATE“ ili „DELETE“. Za oboje su nam ponuđene iste opcije: „CASCADE“, „RESTRICT“, „NO ACTION“ ili „SET NULL“. Prije nego što objasnimo ova 4 ograničenja, važno je napomenuti da tablicu koja sadrži vanjski ključ još nazivamo tablica dijete, a tablica na koju se referenciramo se naziva tablica roditelj. „CASCADE“ mijenja red u tablici dijete kada vršimo promjenu na tablici. „RESTRICT“ brani izmjene i brisanje na tablici roditelja ako ima vanjskih ključeva. „SET NULL“ postavlja NULL u tablici dijete kada roditelj nestane. „NO ACTION“ je u MySQL-u isti kao i „RESTRICT“. Vanjski ključ možemo dodati u tablicu i pomoću naredbe:

```
ALTER TABLE naziv_tablice_dijete
ADD FOREIGN KEY (naziv_atributa_koji_postaje_vanjski_ključ)
REFERENCES                                    naziv_tablice_roditelj
(naziv_atributa_na_koji_se_referenciramo);
```

4.5. Okidači

Nakon što smo prikazali tablice i veze među tablicama, na red dolaze okidači. „Iako su korisnici obično upoznati s pojmom okidača (eng. trigger), malo njih zna da u pozadini okidača postoji razvijena teorija poznata kao teorija aktivnih baza podataka. Aktivne baze podataka su u osnovi klasične baze podataka proširene komponentom koja omogućuje automatsko izvršavanje određenih akcija kao reakcija na događaje koji se mogu dogoditi u ili pak van same baze podataka (izvršavanje akcija je automatsko, bez intervencije korisnika ili neke aplikacije)“ [9, str. 57] Okidač, kao što sama riječ kaže, „okida“ tj. reagira na neku promjenu ili radnju koja se izvršava nad bazom tj. nad tablicama i pogledima. Okidač u MySQL sustavu za upravljanje bazom podataka reagira prije ili poslije unosa, ažuriranja ili brisanja. Okidač u MySQL-u kreiramo naredbom „`CREATE TRIGGER`“ nakon čega definiramo naziv okidača i tablicu nad kojom se okidač izvodi. Nakon toga pišemo „kod“ tj. funkciju koja se izvodi kada okidač reagira.

Dakle u sustavu MySQL okidače kreiramo tako da prvo specificiramo naziv okidača koji mora biti jedinstven unutar baze podataka. Kada smo naveli naziv, biramo kada će se okidač aktivirati tj. hoće li se aktivirati prije ili poslije („BEFORE“ ili „AFTER“) izvođenja operacije. Na dalje biramo koja operacija aktivira okidač. Operacije koje aktiviraju okidač su „INSERT“, „UPDATE“ i „DELETE“. Okidač se može aktivirati na samo jednu od tih naredbi ili na dvije, odnosno sve tri. Zadnje što treba definirati prije pisanja samog okidača je tablica nad kojom se okidač izvodi. Kada je sve definirano prelazi se na pisanje funkcije unutar okidača. Funkcija određuje što će se desiti kada se okidač aktivira, a sama funkciju započinjemo s „BEGIN“ i završavamo s „END“.

Sintaksa okidača:

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE| DELETE }
ON table_name FOR EACH ROW
trigger_body;
```

Kada pišemo funkciju kako bi razlikovali vrijednosti koje su u tablici i vrijednost koja se unosi koristimo „NEW“ i „OLD“ modifikatore. Dakle pomoću njih možemo pristupiti novoj informaciji koja se unosi, te vršiti razne provjere tj. upite. „OLD“ je dostupan pri ažuriranju i brisanju, dok je „NEW“ dostupan pri unosu i ažuriranju. [10].

Naš primjer baze za autoškolu sadržava 10 okidača koje ćemo pojasniti u nastavku. Zbog lakšeg praćenja grupirat ćemo okidače prema tablicama nad kojima se oni izvršavaju.

Tablica 3: Okidači

OPIS OKIDAČA	KOD OKIDAČA

Okidač „provjera_dobi“ izvršava se pri unosu ili ažuriranju u tablicu „polaznik“, te provjera je li polaznik kojeg unosimo stariji od sedamnaest i pol godina koje su zakonom propisani minimum za upis u autoškolu.	`CREATE TRIGGER `provjera_dobi` BEFORE INSERT OR UPDATE ON `polaznik``
	`FOR EACH ROW BEGIN`
	`IF (DATEDIFF(CURRENT_DATE(),NEW.datumRodenja) < 6387) THEN`
	`SIGNAL SQLSTATE '45000'`
	`SET MESSAGE_TEXT = 'Polaznik se ne može upisati jer nije stariji od 17 i pol godina';`
	`END IF;`
	`END`

Ovaj okidač provjerava posjeduje li autoškola vozilo za kategoriju koju određeni polaznik želi upisati. Ako autoškola ne posjeduje vozilo za tu kategoriju javlja se greška i brani se upis. Također ovaj okidač vrši istu provjeru kada se vrši ažuriranje.	`CREATE TRIGGER `provjera_vozila_insert_polaznika` BEFORE INSERT OR UPDATE ON `polaznik``
	`FOR EACH ROW BEGIN`
	`DECLARE rowcount INT;`
	`SELECT COUNT(*)`
	`INTO rowcount`
	`FROM vozilo v`
	`WHERE v.kategorijaID=new.kategorijaID;`
	`IF rowcount = 0 THEN`
	`SIGNAL SQLSTATE '45000'`
	`SET MESSAGE_TEXT = 'Polaznik ne može biti upisan jer autoškola nema vozilo za odabranu kategoriju!';`
	`END IF;`
	`END`

Okidač „smanjenje_duga“ pri uplati provjerava je li iznos uplate veći od iznosa duga, te ako nije, vrši	`CREATE TRIGGER `smanjenje_duga` AFTER INSERT ON `uplata``
	`FOR EACH ROW BEGIN`
	`declare dug float;`

„UPDATE“ nad tablicom polaznik tj. vrši se smanjenje duga za količinu uplate.	select p.dugovanje into dug from polaznik p where p.oib=new.OIBpolaznika;
	IF new.iznos>dug THEN
	SIGNAL SQLSTATE '45000'
	SET MESSAGE_TEXT = Iznos je veći od polaznikovog duga!';
	ELSE
	update polaznik p set p.dugovanje = p.dugovanje-new.iznos where p.oib = new.OIBpolaznika;
	END IF;
	
	END

Ovaj okidač reagira prije unosa u tablicu „polazniknaispitu“.
Provjerava postoji li zapis da je polaznik položio ispit iz teorije na nekom od prijašnjih ispita. Također, provjerava je li rezultat ispita koji se želi unijeti iz vožnje. Ako polaznik nema položene propise tj. teoriju onda se brani polaganje vožnje.
	CREATE TRIGGER `provjera_polaznik_ispit` BEFORE INSERT OR UPDATE ON `polazniknaispitu`
	FOR EACH ROW BEGIN
	DECLARE teorija int;
	DECLARE jeri_voznja int;
	
	select COUNT(*)
	into teorija
	from polazniknaispitu p
	where p.OIBpolaznika=new.OIBpolaznika and p.rezultat="teorija_polozio" ;
	
	select COUNT(*)
	INTO jeri_voznja
	from polazniknaispitu
	WHERE new.rezultat LIKE 'voznja%';
	
	IF teorija = 0 and jeri_voznja>0 THEN
	SIGNAL SQLSTATE '45000'
	SET MESSAGE_TEXT = 'Polaznik ne može biti prijavljen na ispit iz vožnje jer nije položio ispit iz teorije!';
	END IF;
	END

Ovaj okidač se izvršava prije unosa u tablicu „polazniknaispitu“. Ako se unosi rezultat za vožnju, okidač osigurava da se taj rezultat ne unese pod ispit iz	CREATE TRIGGER `provjera_polaznik_vrsta_ispita` BEFORE INSERT OR UPDATE ON `polazniknaispitu` FOR EACH ROW BEGIN DECLARE vrsta varchar(40); DECLARE reza varchar(40); select i.vrstaIspita

teorije. U slučaju greške javlja se poruka, da se polaznik ne može unijeti jer se vrsta ispita ne slaže.	into vrsta
	from ispit i
	where new.ispitID=i.ispitID;
	select new.rezultat
	into reza;
	
	IF vrsta = "voznja" and reza LIKE 'teorija%' THEN
	SIGNAL SQLSTATE '45000'
	SET MESSAGE_TEXT = "Polaznik ne može biti prijavljen na ispit vrsta ispita se ne slaže";
	END IF;
	END

Ovaj okidač je vrlo sličan prošlom okidaču. Izvršavaju se na istoj tablici i provjeravaju sličan događaj. Ovaj okidač provjerava slažu li se vrste ispita i osigurava da se rezultat iz ispita iz teorije ne unese pod ispit iz vožnje.	CREATE TRIGGER `provjera_polaznik_vrsta_ispita2` BEFORE INSERT OR UPDATE ON `polazniknaispitu`
	FOR EACH ROW BEGIN
	DECLARE vrsta varchar(40);
	DECLARE reza varchar(40);
	
	select i.vrstaIspita
	into vrsta
	from ispit i
	where new.ispitID=i.ispitID;
	select new.rezultat
	into reza;
	
	IF vrsta = "teorija" and reza LIKE 'voznja%' THEN
	SIGNAL SQLSTATE '45000'
	SET MESSAGE_TEXT = "Polaznik ne može biti prijavljen na ispit vrsta ispita se ne slaže";
	END IF;
	END

Ovaj okidač se vrši prije unosa u tablicu „ispit“. Provjerava je li vrijednost atributa „vrstaIspita“ jednak „voznja“ ili „teorija“. Ako unesemo nešto osim toga, sustav javlja grešku da vrsta ispitivača mora biti „voznja“	CREATE TRIGGER `ogranicenje_vrste_ispita` BEFORE INSERT OR UPDATE ON `ispit` FOR EACH ROW BEGIN IF new.vrstaIspita != 'voznja' AND new.vrstaIspita != 'teorija' THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Vrsta ispita mora biti voznja ili teorija'; END IF;

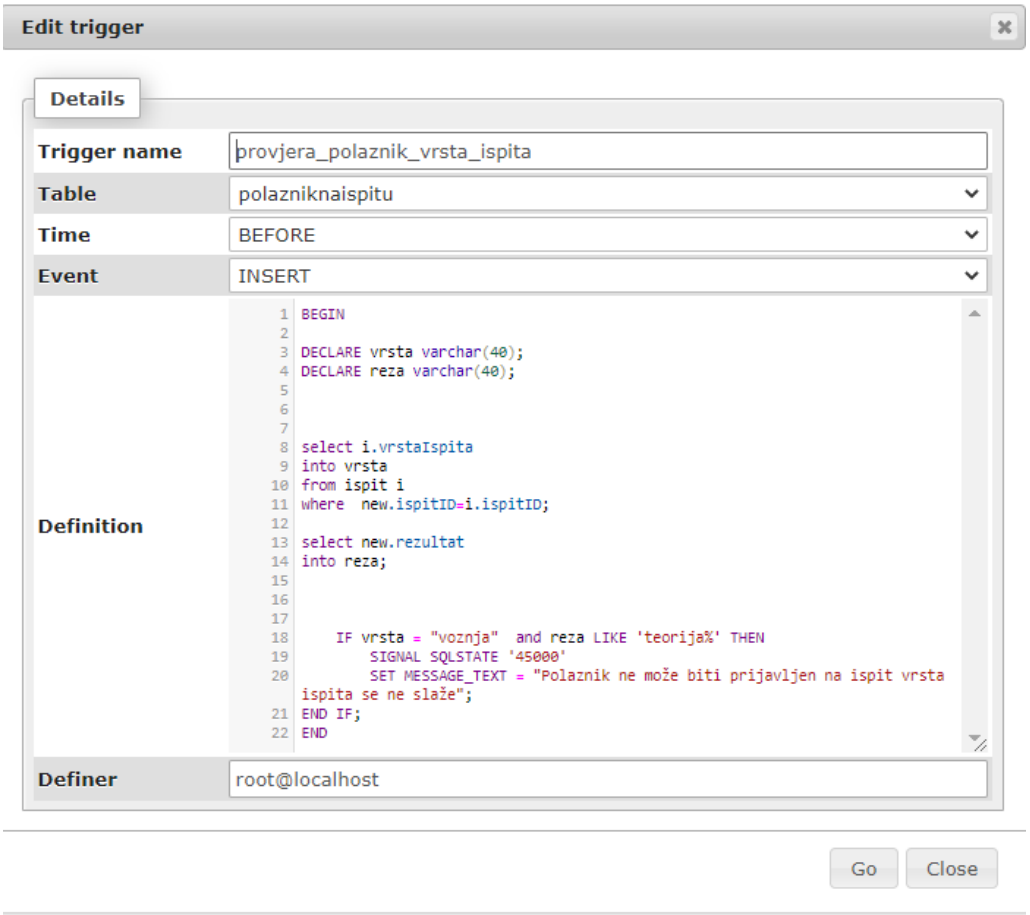
ili „teorija“. Ovo ograničenje nam služi kako bi imali jednake vrijednosti tj. zbog dodjele Ispitivača novom ispitu, a to provjeravamo u drugom okidaču.	END

Ovaj okidač nam služi za provjeru je li vrsta ispita kojeg želimo kreirati jednaka vrsti ispitivača kojeg želimo dodijeliti tom ispitu. Ako se vrsta ispitivača i vrsta ispita ne poklapaju javlja se greška uz poruku da se ispit ne može kreirati jer se vrsta ispita ne poklapa s vrstom ispitivača.	CREATE TRIGGER `provjera_vrste_ispita_i_ispitivaca` BEFORE INSERT OR UPDATE ON `ispit`
	FOR EACH ROW BEGIN
	declare vrsta varchar(45);
	select i.vrstaIspitivaca into vrsta from ispitivac i WHERE i.ispitivacID=new.ispitivacID;
	IF NEW.vrstaIspita != vrsta THEN
	SIGNAL SQLSTATE '45000'
	SET MESSAGE_TEXT = 'Ispit se ne može kreirati vrsta ispita se ne poklapa s vrstom ispitivaća!';
	END IF;
	END

Ovaj okidač kao i kod tablice „ispit“ provjerava vrstu. Provjerava tj. ograničava nam da vrijednost atributa „vrstaIspitivaca“ mora biti „voznja“ ili „teorija“. Služi nam za usklađivanjem s ostatkom baze tj. tablicom „ispit“.	CREATE TRIGGER `ogranicenje_vrste_ispitivaca` BEFORE INSERT OR UPDATE ON `ispitivac` FOR EACH ROW BEGIN IF new.vrstaIspitivaca != 'voznja' AND new.vrstaIspitivaca != 'teorija' THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Vrsta ispitivaca mora biti voznja ili teorija'; END IF; END
Ovaj okidač se izvršava prije unosa u tablicu „voznja“. Provjerava se je li polaznik položio ispit iz teorije tj. propisa. Ako je ispit položen,	CREATE TRIGGER `provjera_polozenog_ispita` BEFORE INSERT ON `voznja` FOR EACH ROW BEGIN DECLARE polozen int; SELECT COUNT(*) into polozen

onda polaznik može pristupiti praktičnoj obuci tj. vožnji. Ako polaznik nije položio ispit iz propisa, polaznik ne smije ići na vožnju te se javlja greška da polaznik ne smije na vožnju jer nije položio ispit iz propisa.	FROM polazniknaispitu p
	WHERE new.OIBpolaznika=p.OIBpolaznika and p.rezultat="teorija_polozio";
	
	IF polozen=0 THEN
	SIGNAL SQLSTATE '45000'
	SET MESSAGE_TEXT = 'Polaznik ne može na vožnju jer nije položio ispit iz teorije';
	END IF;
	END

U tablici su objašnjeni i prikazani svi okidači koji se koriste u bazi podataka. Svi okidači su kreirani u alatu phpMyAdmin.



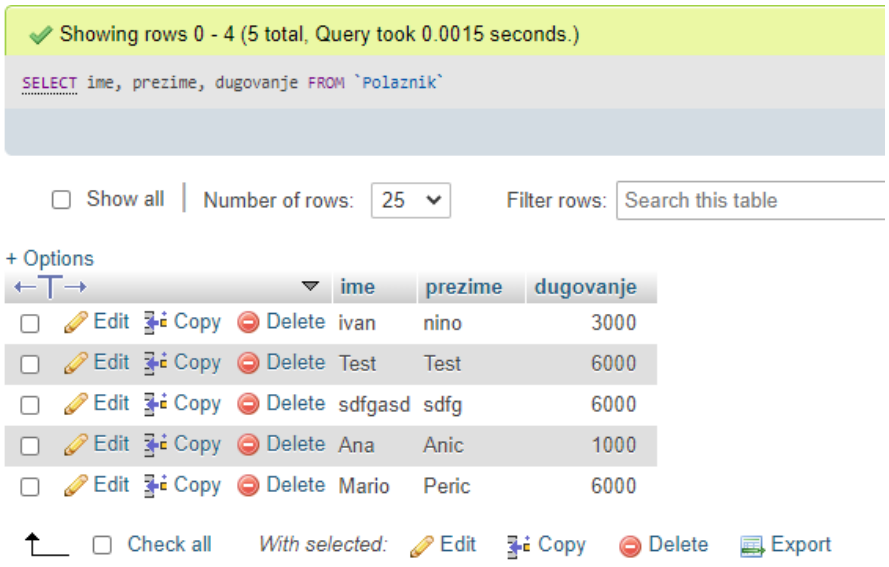
Slika 6: Okidač u alatu phpMyAdmin

Na slici 6. je prikazan okidač koji je već objašnjen. Na slici se vidi prozor u alatu phpMyAdmin u kojem se okidač definira. Pomoću prikazanog sučelja moguće je kreirati i ažurirati okidače.

4.6. Upiti nad bazom

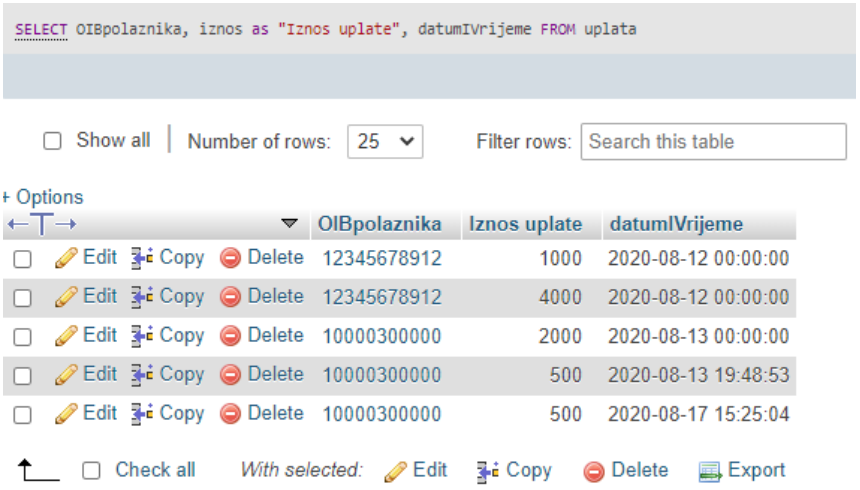
Budući da smo kreirali bazu podataka sa svim ograničenjima i unaprijed smo popunili tablice podatcima, možemo nad bazom vršiti različite upite. Upite možemo podijeliti na jednostavne i složene. Jednostavni upiti se izvršavaju nad jednom tablicom, dok se složeni izvršavaju kombinacijom više tablica. Prikazat ćemo par jednostavnih i par složenih upita. U MySQL-u upite pišemo pomoću naredbe „SELECT“. Primjer jednog najjednostavnijeg upita nad našom bazom je „`SELECT * FROM polaznik;`“, ovaj upit ispisuje sve iz tablice polaznik.

4.6.1. Jednostavni upiti



Slika 7: Jednostavni upit 1

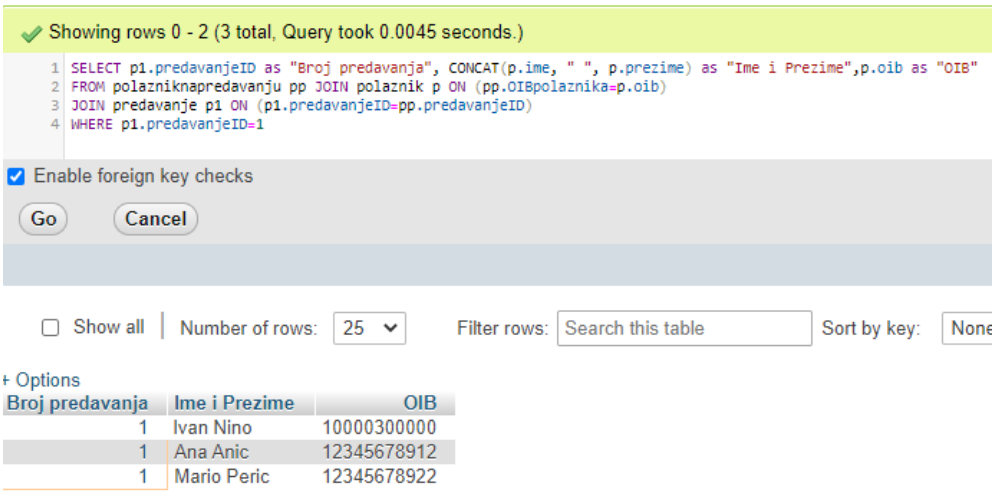
Na slici 7. vidimo jednostavni upit koji dohvaća ime, prezime i dugovanje iz tablice polaznik i ispisuje ih. Također, važno je napomenuti da smo sve upite pisali u alatu phpMyAdmin.



Slika 8: Jednostavni upit 2

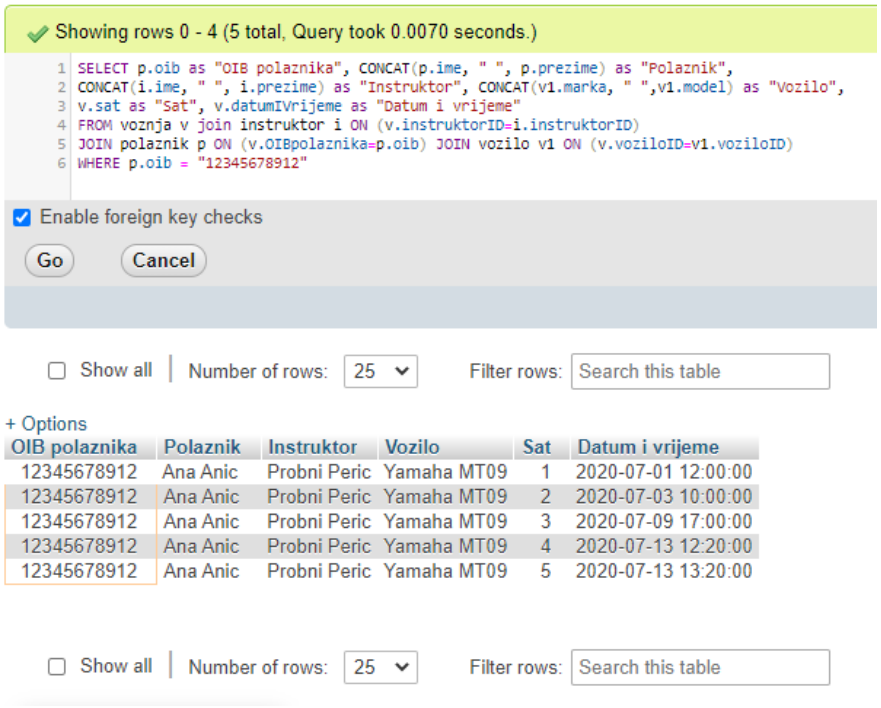
Na slici 8. vidimo jednostavni upit koji dohvaća OIBpolaznika, iznos koji smo preimenovali u „iznos uplate“ i datumIVrijeme iz tablice uplata.

4.6.2. Složeni upiti



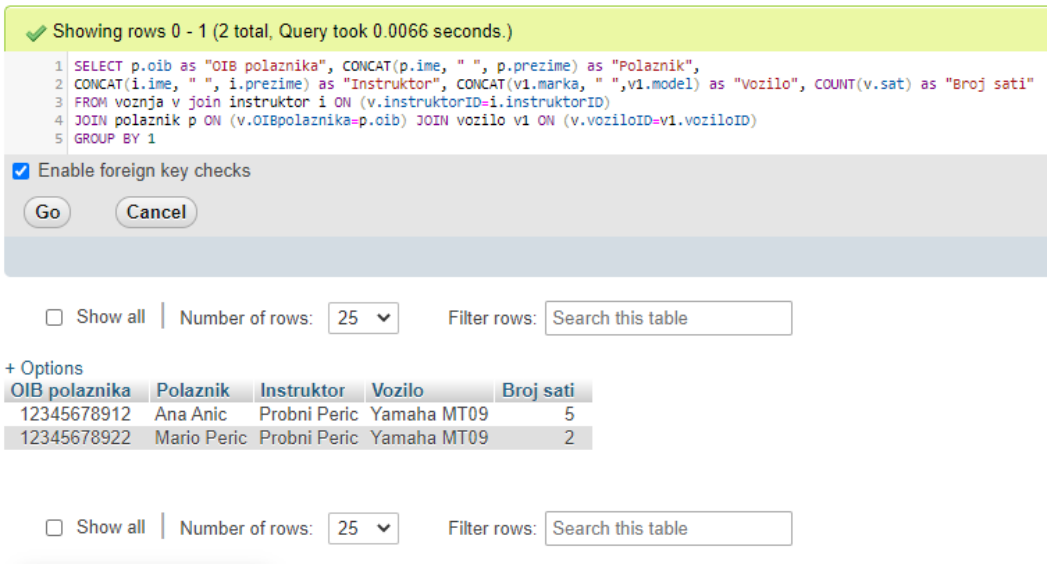
Slika 9: Složeni upit 1

Sada imamo primjer jednog složenog upita u koji smo spojili rezultate iz 3 tablice. Upit dohvaća ID predavanja iz tablice „predavanje“, dohvaća ime i prezime iz tablice „polaznik“, ali ih spaja u jedan stupac pomoću naredbe „CONCAT“ i na kraju dohvaća OIB iz tablice „polaznik“. Spajanje tablica se vrši pomoću naredbe „JOIN“. Također imamo i jedno ograničenje pomoću kojeg smo prikazali prisutnost samo na prvom predavanju.



Slika 10: Složeni upit 2

Slika 10. prikazuje složeni upit koji spaja 4 tablice: voznja, polaznik, instruktor, vozilo. Iz tih tablica ispisuje OIB polaznika, ime i prezime polaznika, ime i prezime instruktora, model i marku vozila, sat vožnje i datum i vrijeme vožnje. Pomoću ograničenja „`p.oib="12345678912"`“ prikazali smo samo vožnje polaznice „Ana Anic“.

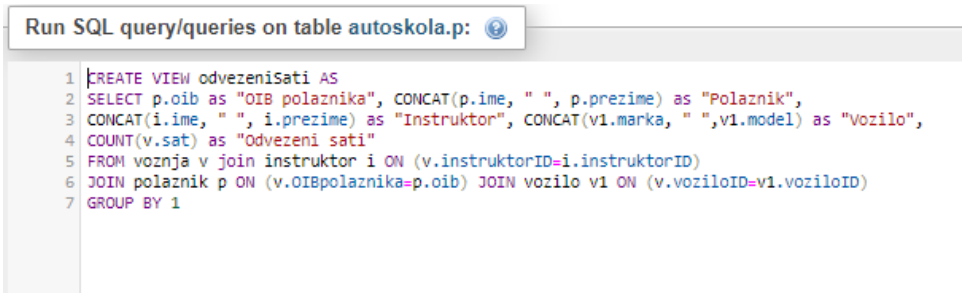


Slika 11: Složeni upit 3

Slika 11. prikazuje složeni upit koji je vrlo sličan prošlom upitu. U ovom upitu je izbačen „WHERE“, te smo pomoću naredbe „GROUP BY“ grupirali sve polaznike koji voze i pomoću naredbe „COUNT“ prebrojen je broj sati koji su odvezli.

4.6.3. Pogledi

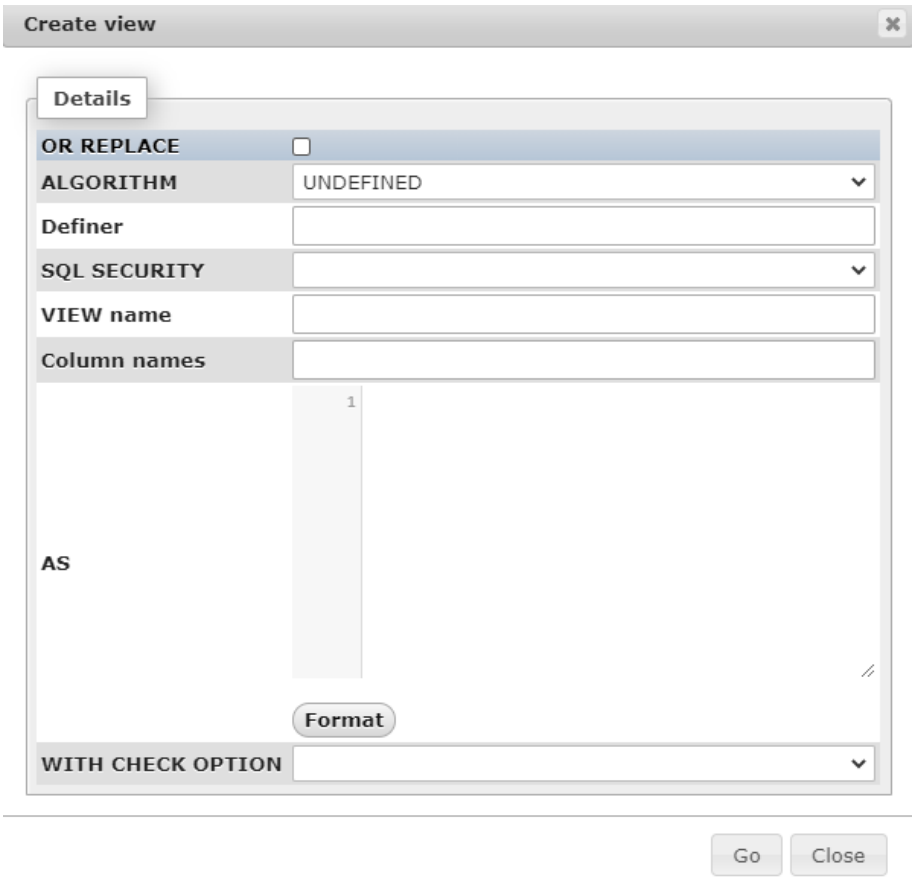
Pogledi (eng. view) su vrlo korisni kada trebamo neki složeni upit izvršiti više puta. Prošli upit nam je vrlo koristan jer nam prikazuje koliko je koji polaznik odvozio sati. Kako ne bismo svaki put pisali taj upit spremiti ćemo ga kao pogled. Pogled kreiramo pomoću naredbe „CREATE VIEW „naziv pogleda“ AS „dalje pišemo upit kao inače““.



Slika 12: Pogled 1

Slika 12. prikazuje prošli upit koji smo spremili kao pogled pomoću „`CREATE VIEW odvezeniSati AS …`“

U sustavu phpMyAdmin možemo kreirati poglede pomoću obrasca. Pritiskom na gumb „new view“ otvara nam se prozor u kojem možemo kreirati ili zamijeniti pogled.

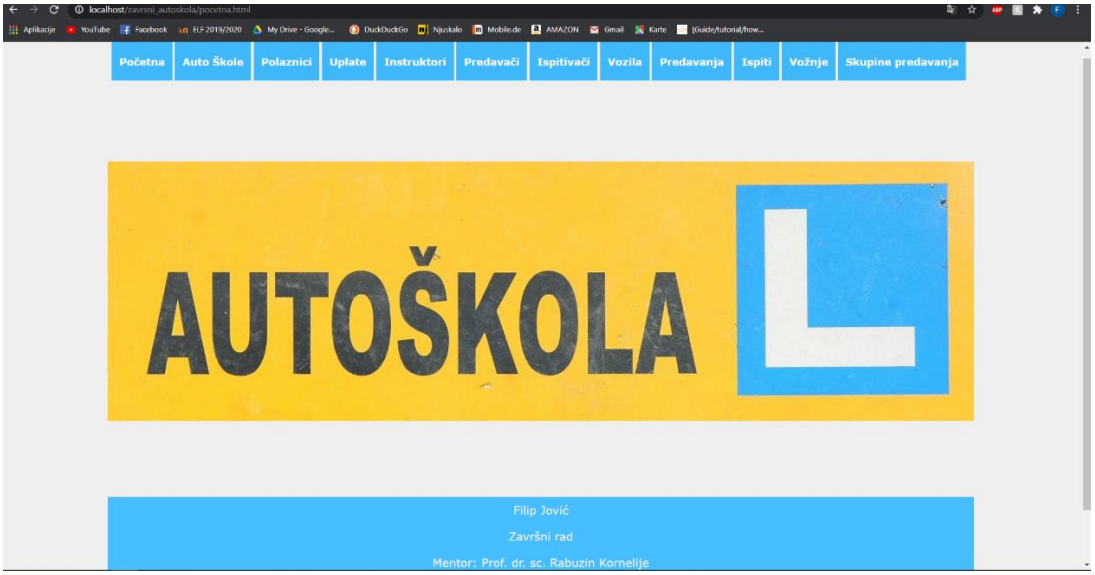


Slika 13: Prozor Create View

5. Aplikacija

U ovom poglavlju ćemo proći kroz izgled i funkcionalnosti same aplikacije. Aplikacija za potrebe autoškole je kreirana pomoću HTML-a i PHP-a. Aplikacija je vrlo jednostavna i pomoću nje možemo vršiti CRUD operacije nad samom bazom. CRUD je skraćenica od Create, Read, Update i Delete.

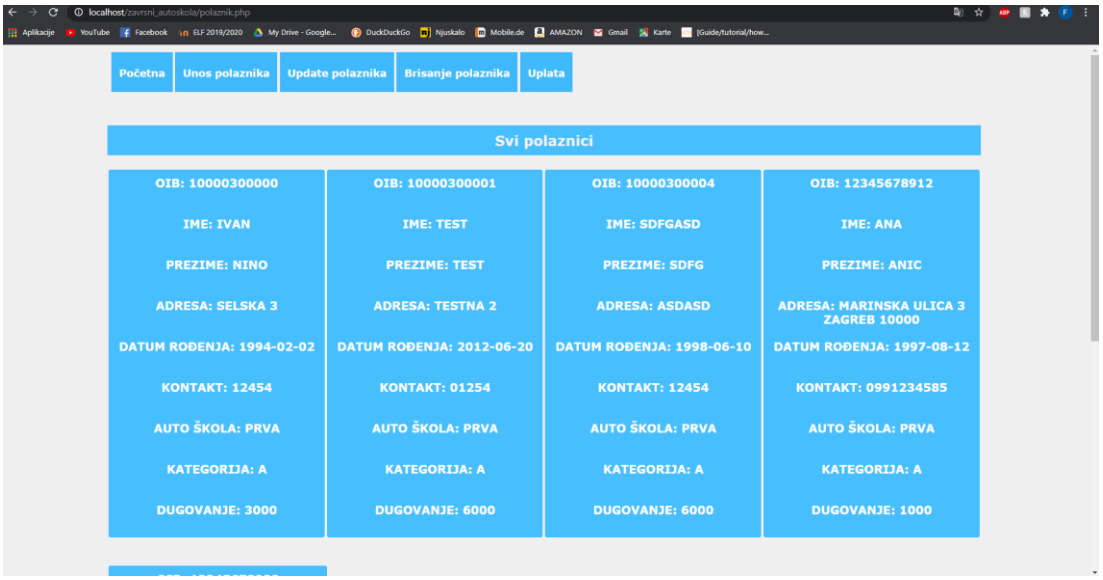
5.1. Početna stranica



Slika 14: Početna stranica

Slika 14. prikazuje početnu stranicu aplikacije za autoškolu. Na vrhu stranice se nalazi izbornik koji sadrži 11 linkova. Svaki link vodi na stranicu za upravljanje tim dijelom baze i svaki link ima još linkova tj. opcija. Većina stranica je slična, pa ćemo opisati Polaznike, Uplate i Ispite. Te tri stranice imaju sve opcije kao i ostatak aplikacije, pa ih možemo uzeti kao primjer.

5.2. Polaznici



Slika 15: Stranica polaznici

Slika 15. prikazuje stranicu polaznici. Na vrhu stranice imamo nekoliko opcija za upravljanje polaznicima i opciju za vršenje uplate. Na sredini imamo prikaz svih polaznika i njihove podatke.

```
<main class="fleks">
    <?php
        $dbc = mysqli_connect('localhost', 'root', '', 'autoskola')
               or die('Neuspješno spajanje na bazu.');
        $query = "SELECT DISTINCT p.oib,p.ime,p.prezime,p.datumRodenja,p.adresa,p.kontakt,a.naziv as skola_naziv,k.naziv as
        kategorija, p.dugovanje
                  FROM polaznik p join autoskola a on(p.autoskolaID=a.autoskolaID)
                  join kategorija k on(p.kategorijaID=k.kategorijaID) ";
        $result = mysqli_query($dbc, $query);
        while($row = mysqli_fetch_array($result)) {
                echo '<article class="polaznik">';
                    echo '<section class="opis-kandidata">';
                        echo '<h2 class="naziv boldw">OIB: ' . $row['oib'] . '</h2>';
                        echo '<h2 class="naziv boldw">Ime: ' . $row['ime'] . '</h2>';
                        echo '<h2 class="naziv boldw">Prezime: ' . $row['prezime'] . '</h2>';
                        echo '<h2 class="naziv boldw">Adresa: ' . $row['adresa'] . '</h2>';
                        echo '<h2 class="naziv boldw">Datum Rođenja: ' . $row['datumRodenja'] . '</h2>';
                        echo '<h2 class="naziv boldw">Kontakt: ' . $row['kontakt'] . '</h2>';
                        echo '<h2 class="naziv boldw">Auto Škola: ' . $row['skola_naziv'] . '</h2>';
                        echo '<h2 class="naziv boldw">Kategorija: ' . $row['kategorija'] . '</h2>';
                        echo '<h2 class="naziv boldw">Dugovanje: ' . $row['dugovanje'] . '</h2>';
                        echo '</section>';
                echo '</a>';
            echo '</article>';
        }
        mysqli_close($dbc);
    ?>
</main>
```

Slika 16: Polaznici kod

Na slici 16. vidimo PHP kod pomoću kojeg se spajamo na bazu i dohvaćamo podatke koji su prikazani na slici 15. tj. na stranici Polaznici. Podatke dohvaćamo pomoću SQL upita koji glasi:

```
        $query= "SELECT DISTINCT
p.oib,p.ime,p.prezime,p.datumRodenja,p.adresa,p.kontakt,a.naziv as
skola_naziv,k.naziv as kategorija, p.dugovanje
```

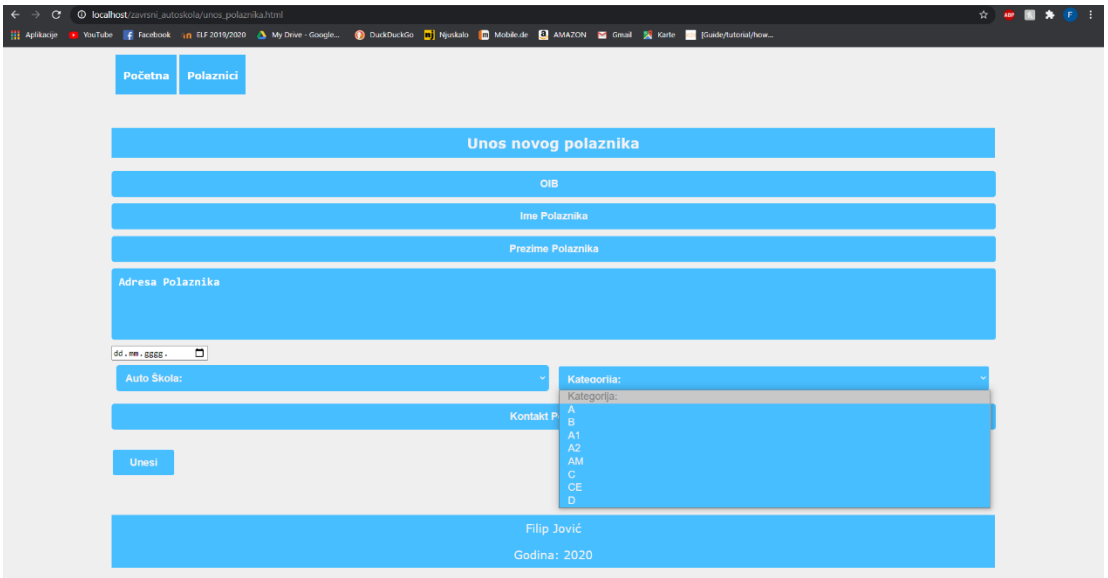
```
FROM polaznik p join autoskola a on(p.autoskolaID=a.autoskolaID)
join kategorija k on(p.kategorijaID=k.kategorijaID) ";
```

Dohvaćene podatke ispisujemo pomoću „echo“ naredbi:

```
echo '<section class="opis-polaznika">';
echo '<h2 class="naziv boldw">OIB: ' . $row['oib'] . '</h2>';
echo '<h2 class="naziv boldw">Ime: ' . $row['ime'] . '</h2>';
echo '<h2 class="naziv boldw">Prezime: ' . $row['prezime'] . '</h2>';
echo '<h2 class="naziv boldw">Adresa: ' . $row['adresa'] . '</h2>';
echo '<h2 class="naziv boldw">Datum Rođenja: ' . $row['datumRodenja']
. '</h2>';
echo '<h2 class="naziv boldw">Kontakt: ' . $row['kontakt'] . '</h2>';
echo '<h2 class="naziv boldw">Auto Škola: ' . $row['skola_naziv'] .
'</h2>';
echo '<h2 class="naziv boldw">Kategorija: ' . $row['kategorija'] .
'</h2>';
echo '<h2 class="naziv boldw">Dugovanje: ' . $row['dugovanje'] .
'</h2>';
echo '</section>';
```

Pomoću sličnih upita i naredbi dohvaćamo podatke za druge stranice kao što su: instruktori, predavači, predavanja, vožnje i td.. Zbog toga što je aplikacija jednostavna i većina stranica koristi sličan kod nije ih potrebno sve opisati.

5.2.1. Unos polaznika



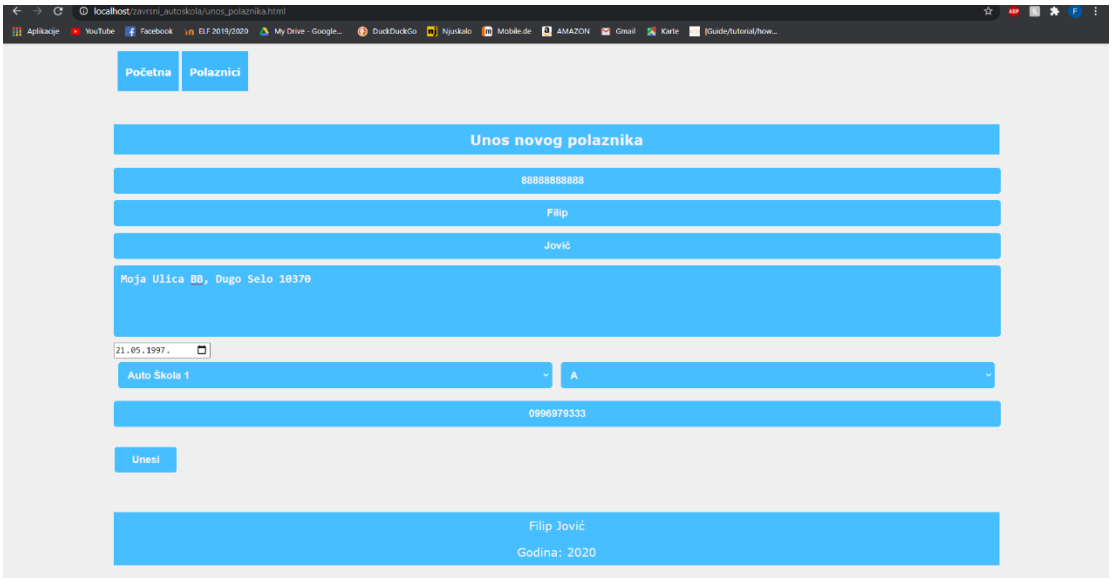
Slika 17: Unos polaznika

Unos u tablicu „polaznik“ vrši se preko forme prikazane na slici 17. Kako bi se izvršio unos polaznika moramo popuniti sva polja i godište novog polaznika mora biti starije od 17 i pol godina. To smo ograničili okidačima koje smo već naveli.

```
28          <h2>Unos novog polaznika</h2>
29          </section>
30                  <form name="unosPolaznika" enctype="multipart/form-data" action="skripta_polaznik.php" method="POST">
31                      <section class="odvoj">
32                      <span id="porukaoibPolaznika"></span>
33                      <input type="number" min=10000000000 max=99999999999 name="oib" id="oib" placeholder="OIB"/>
34                      <span id="porukaimePolaznika"></span>
35                      <input type="text" name="imePolaznika" id="imePolaznika" placeholder="Ime Polaznika"/>
36                      <span id="porukaimePolaznika"></span>
37                      <input type="text" name="prezimePolaznika" id="prezimePolaznika" placeholder="Prezime Polaznika"/>
38                      </section>
39                      <section class="odvoj">
40                      <textarea name="adresaPolaznika" id="adresaPolaznika" placeholder="Adresa Polaznika"></textarea>
41                      </section>
42                      <section class="odvoj">
43                      <span id="porukaDatumRodenjaPolaznika"></span>
44                      <input type="date" name="datumRodenjaPolaznika" id="datumRodenjaPolaznika" placeholder="Datum Rođenja Polaznika"/>
45                      </section>
46                      <section class="odvoj">
47                      <span id="porukaNazivautoskole"></span>
48                      <select name="autoSkola" id="autoSkola">
49                          <option value="" disabled selected class="drop">Auto Škola:</option>
50                          <option value="1" class="drop">Auto Škola  1</option>
51                      </select>
52                      <span id="porukaNazivautoskole"></span>
53                      <select name="kategorija" id="kategorija">
54                          <option value="" disabled selected class="drop">Kategorija:</option>
55                          <option value="1" class="drop">A</option>
56                          <option value="2" class="drop">B</option>
57                          <option value="3" class="drop">A1</option>
58                          <option value="4" class="drop">A2</option>
59                          <option value="5" class="drop">AM</option>
60                          <option value="6" class="drop">C</option>
61                          <option value="7" class="drop">CE</option>
62                          <option value="8" class="drop">D</option>
63                      </select>
64                      <span id="porukaimePolaznika"></span>
65                      <input type="text" name="kontaktPolaznika" id="kontaktPolaznika" placeholder="Kontakt Polaznika"/>
66                      </section>
67                      <input type="submit" id="slanje" value="Unesi">
68                  </form>
69              </article>
70          </main>
71
```

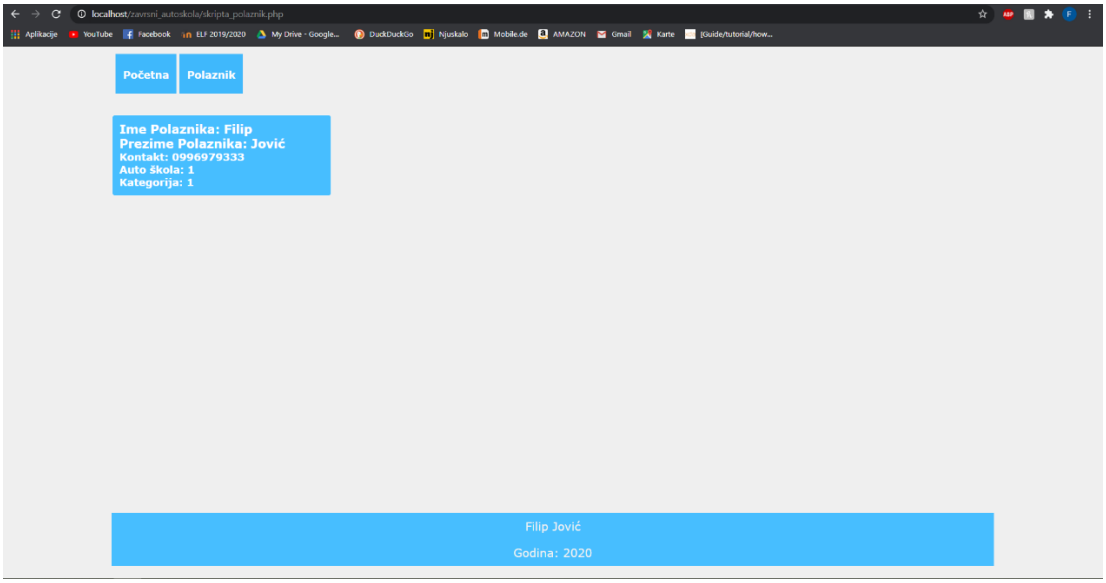
Slika 18: Unos polaznika kod

Na slici 18. je prikazan kod forme za unos. Unos se vrši pomoću POST metode. Klikom na gumb „unesi“ uneseni podatci šalju SE na skriptu za unos tj. u ovom slučaju „skripta_polaznik.php“. Također iz ovog koda možemo vidjeti da je „OIB“ ograničen tj. da mora biti broj između 10000000000 i 99999999999. Također, odabir autoškole i kategorije su napravljeni kao padajući izbornici. U slučaju da se autoškola proširi, ovdje bismo ručno morali dodati novu opciju. Na iduće dvije slike ćemo prikazati unos u bazu.



Slika 19: Primjer unosa polaznika

Klikom na „unos“, izvršava se skripta za unos i daje nam povratnu informaciju da je polaznik unesen.



Slika 20:Primjer izvršenog unosa polaznika

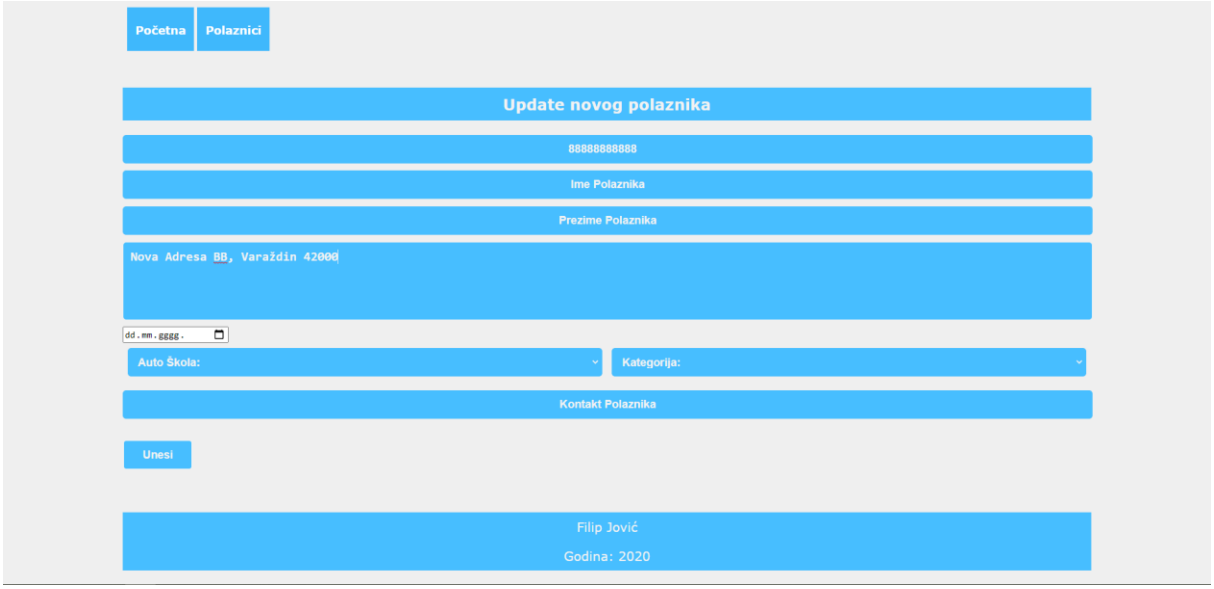
```php
<?php

// Spremanje podataka poslanih putem forme u varijable
$oib = $_POST['oib'];
$imePolaznika = $_POST['imePolaznika'];
$prezimePolaznika = $_POST['prezimePolaznika'];
$adresaPolaznika = $_POST['adresaPolaznika'];
$datumRodenjaPolaznika = $_POST['datumRodenjaPolaznika'];
$kontaktPolaznika = $_POST['kontaktPolaznika'];
$autoSkola = $_POST['autoSkola'];
$kategorija = $_POST['kategorija'];

// Spajanje i spremanje u bazu unosa
$dbc = mysqli_connect('localhost', 'root', '', 'autoskola')
       or die('Neuspješno spajanje na bazu.');
$query = "INSERT INTO polaznik (oib,
                                ime,
                                 prezime,
                                 adresa,
                                 datumRodenja,
                                 kontakt,
                                 autoskolaID,
                                 kategorijaID
                                 )
          VALUES ('$oib',
                  '$imePolaznika',
                  '$prezimePolaznika',
                  '$adresaPolaznika',
                  '$datumRodenjaPolaznika',
                  '$kontaktPolaznika',
                  '$autoSkola',
                  '$kategorija')";
$result = mysqli_query($dbc, $query)
          or die('Nije uspio upit');
mysqli_close($dbc);
?>
```

Slika 21: Skripta za unos polaznika

Na slici 21. vidimo kod koji se izvršava kada kliknemo unesi. Dohvaćaju se uneseni podatci pomoću POST metode. I izvršava se jednostavni „INSERT“ u bazu. Dugovanje ne unosimo jer je ono definirano kao default 6000 kn, i njega smanjujemo pomoću stranice Uplata koju ćemo prikazati kasnije.

5.2.2. Ažuriranje polaznika

Ažuriranje polaznika se vrši pomoću iste forme kao i unos. Razlika je ta da se ne unose sve vrijednosti nego samo one koje želimo mijenjati. Jedina vrijednost koju moramo unijeti je „OIB“ jer preko njega u bazi pretražujemo polaznika kojemu mijenjamo podatke. Ažuriranje ćemo prikazati kao promjenu adrese polaznika kojeg smo unijeli kao primjer u prošlom poglavlju.



Slika 22: Primjer ažuriranja polaznika

Dakle unijeli smo „OIB“ starog polaznika, i samo onaj podatak koji želimo promijeniti, a to je adresa. Klikom na unos izvršava se skripta za ažuriranje i vraća nam potvrdu.

Slika 23: Potvrda o promjeni

Vidimo da je samo adresa nova i svi prijašnji podatci su zadržani.

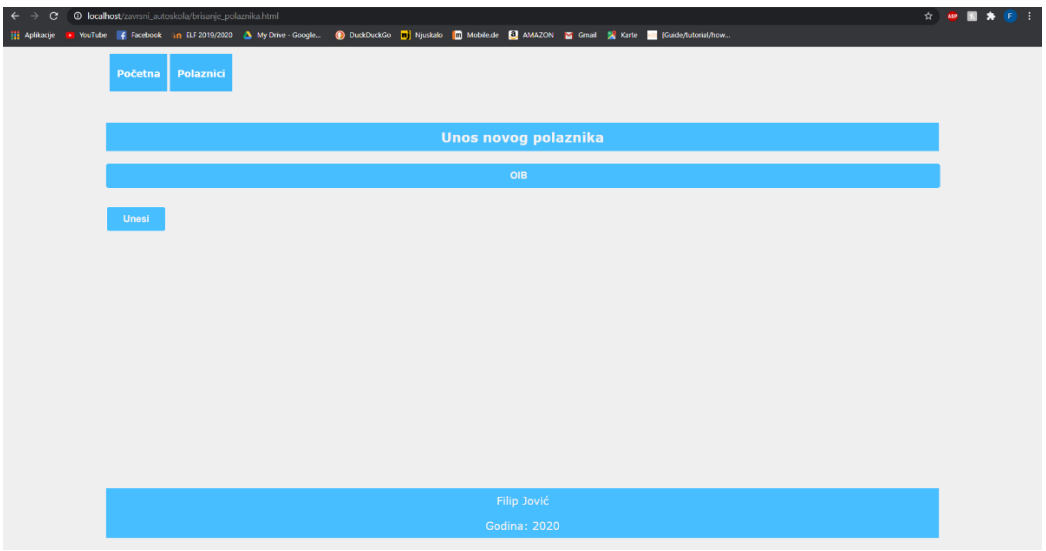
```php
<?php

// Spremanje podataka poslanih putem forme u varijable
$oib = $_POST['oib'];
$imePolaznika = $_POST['imePolaznika'];
$prezimePolaznika = $_POST['prezimePolaznika'];
$adresaPolaznika = $_POST['adresaPolaznika'];
$datumRodenjaPolaznika = $_POST['datumRodenjaPolaznika'];
$kontaktPolaznika = $_POST['kontaktPolaznika'];
$autoSkola = $_POST['autoSkola'];
$kategorija = $_POST['kategorija'];

// Spajanje i spremanje u bazu unosa
$dbc = mysqli_connect('localhost', 'root', '', 'autoskola')
       or die('Neuspješno spajanje na bazu.');
$query = "UPDATE polaznik SET          ime = IF('$imePolaznika'='',ime,'$imePolaznika'),
                                       prezime= IF('$prezimePolaznika'='',prezime,'$prezimePolaznika'),
                                       adresa= IF('$adresaPolaznika'='',adresa,'$adresaPolaznika'),
                                       kontakt= IF('$kontaktPolaznika'='',kontakt,'$kontaktPolaznika'),
                                       datumRodenja= IF('$datumRodenjaPolaznika'='',datumRodenja,'$datumRodenjaPolaznika'),
                                       autoskolaID= IF('$autoSkola'='',autoskolaID,'$autoSkola'),
                                       kategorijaID= IF('$kategorija'='',kategorijaID,'$kategorija')
                                WHERE
                                oib='$oib'";



$result = mysqli_query($dbc, $query)
          or die('Nije uspio upit');
mysqli_close($dbc);
?>
```

Slika 24: Ažuriranje polaznika kod

Vidimo da se „UPDATE“ također izvršava pomoću POST metode. Izvršava se „UPDATE“ naredba, ali je važno napomenuti da kod svakog atributa imamo „IF“ provjeru koja provjerava jesmo li što unijeli ili smo ostavili polje prazno. Ako je polje prazno, zadržava se stara vrijednost, a ako smo unijeli nešto, stara vrijednost se mijenja novo unesenom vrijednosti.

5.2.3. Brisanje polaznika

Brisanje polaznika vršimo pomoću jednostavne forme u kojoj upisujemo samo „OIB“ polaznika kojeg želimo obrisati.



Slika 25: Brisanje polaznika

Pritiskom na gumb „Unos“, izvršava se skripta za brisanje polaznika iz baze.

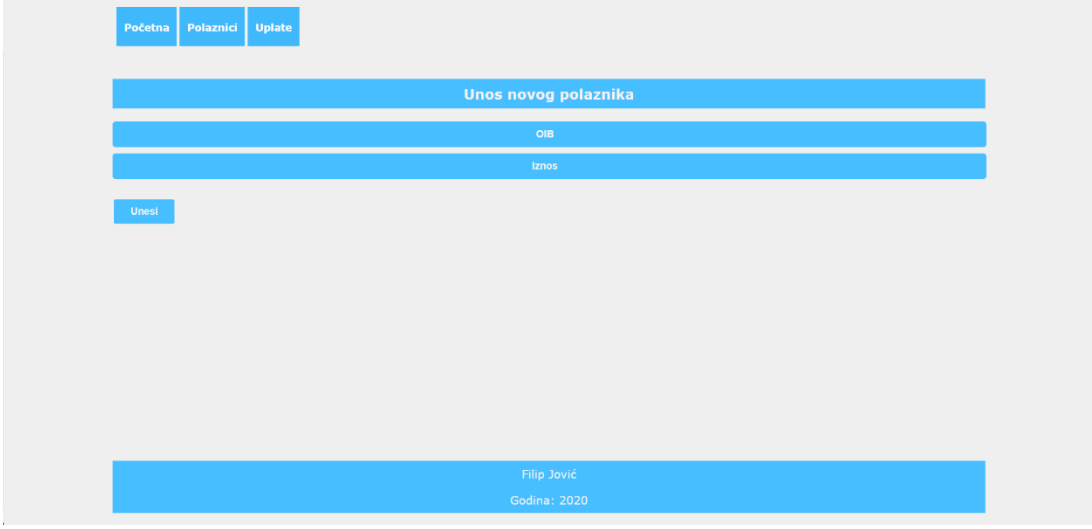
```php
// Spremanje podataka poslanih putem forme u varijable
$oib = $_POST['oib'];


// Spajanje i spremanje u bazu unosa
$dbc = mysqli_connect('localhost', 'root', '', 'autoskola')
       or die('Neuspješno spajanje na bazu.');
$query = "DELETE FROM polaznik
                                WHERE
                                oib='$oib' ";
$result = mysqli_query($dbc, $query)
          or die('Nije uspio upit');
mysqli_close($dbc);
?>
```

Slika 26: Brisanje polaznika

Dakle, kada smo unijeli „OIB“, dohvaćamo ga pomoću POST metode i pomoću „DELETE“ upita polaznik se briše iz baze. Brisanje vršimo na isti način za sve tablice, samo koristimo drugi atribut tj. primarni ključ. Budući da smo to sada pokazali, nema potreba da pokazujemo za ostale tablice.

5.2.4. Uplate

Naveli smo kako svakom polazniku pri unosu automatski tj. pomoću default vrijednosti dodjeljujemo dugovanje koje iznosi 6000 kn. Dugovanje se smanjuje automatski kada polaznik izvrši uplatu. Način na koji je to riješeno je prikazan na slici 11. tj. pomoću okidača koji vrši ažuriranje nad tablicom polaznika kada određeni polaznik izvrši uplatu.



Slika 27: Uplata

Sama uplata se vrši pomoću forme koja je prikazana na slici 39. Unosimo „OIB“ polaznika koji vrši uplatu i iznos uplate. Pritiskom na „Unesi“ izvršava se „INSERT“ u tablicu „uplata“ i pomoću okidača automatski se smanjuje dugovanje u tablici „polaznik“. Također, postoje ograničenja kao što su zabrana unosa većeg iznosa od iznosa dugovanja ili zabrana unosa većeg iznosa od 6000 kn.

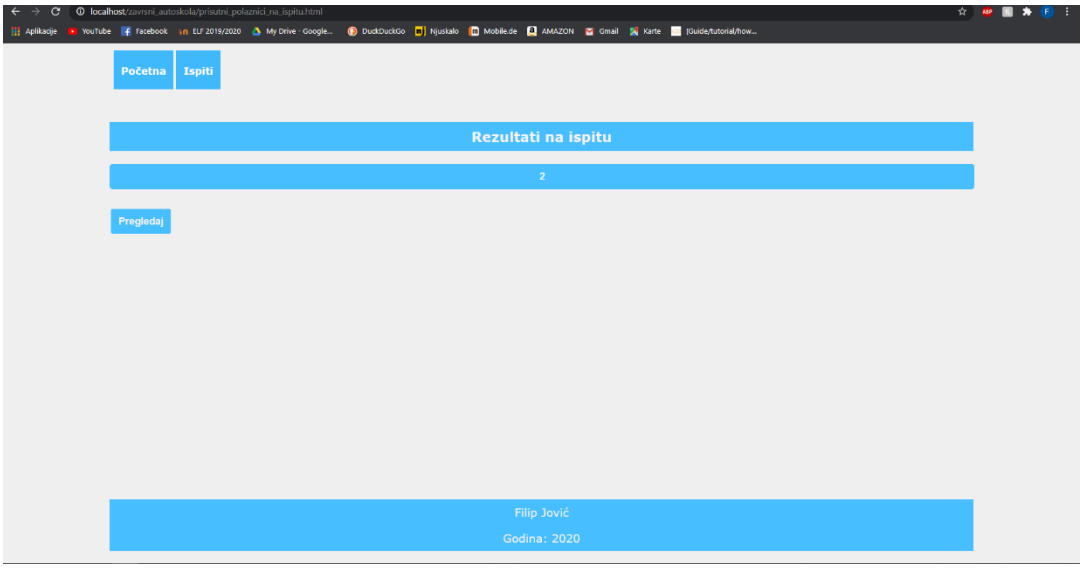
5.3. Ispiti

Slika 28: Ispiti

Slika 28. prikazuje stranicu „ispiti“, na stranici su prikazani svi ispiti, te podatci vezani uz svaki ispit. Sve što je prikazano se dohvaća pomoću upita i POST metoda i vrlo je slično već objašnjenoj stranici „polaznici“. Budući da smo već objasnili unos i brisanje na ovoj stranici ih nećemo objašnjavati. Prikazati ćemo stranice „Rezultati na ispitu“ i „Rezultati po polazniku“.

5.3.1. Rezultati na ispitu, rezultati po polazniku i unos

Stranica rezultati na ispitu nam prikazuje rezultate na odabranom ispitu. Pritiskom na gumb „rezultati na ispitu“, otvara nam se forma za pretragu ispita.



Slika 29: Rezultati na ispitu

U polje za pretragu ispita upisujemo ID traženog ispita, pritiskom na „Pregledaj“ otvara nam se prikaz svih polaznika koji su bili na ispitu skupa s njihovim rezultatom. Podatke koji su nam potrebni za prikaz rezultata dohvaćamo iz baze pomoću upita:

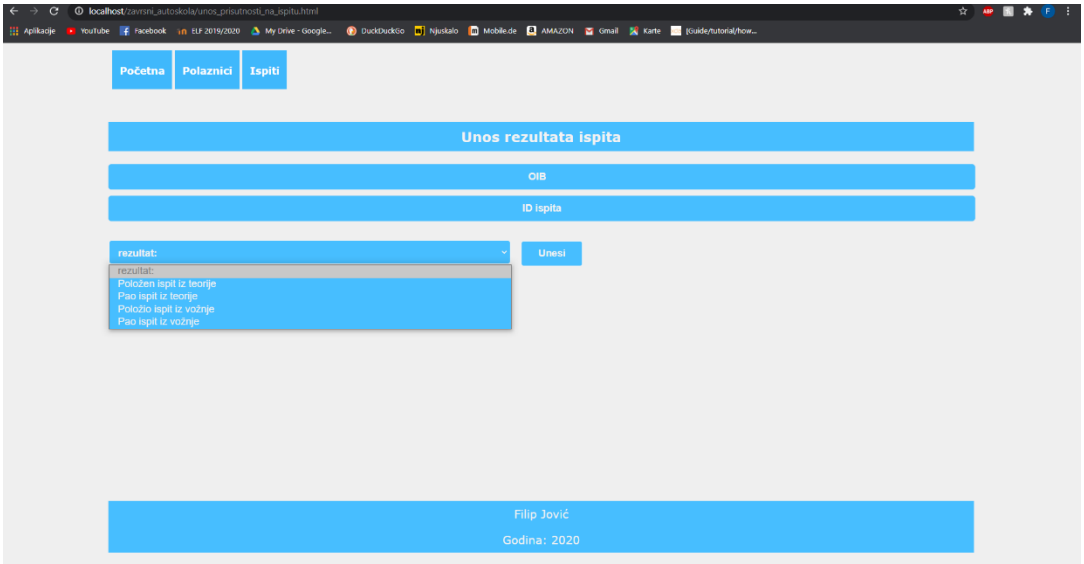
```
$query = "SELECT
pi.ispitID, i.datumIVrijeme, concat(p.ime, ' ', p.prezime) as polaznik,pi.OIBpolaznika, pi.rezultat
FROM polazniknaispitu pi JOIN ispit i ON (pi.ispitID = i.ispitID) JOIN polaznik p ON (p.oib=pi.OIBpolaznika)
WHERE i.ispitID=$idIspita";
```

Slika 30:Prikaz rezultata na ispitu

Dakle podatak o „ispitID“ je dohvaćen pomoću POST metode, a ostale smo dohvatili pomoću upita. Ispis podataka se obavlja pomoću „echo“ naredbe i to smo prikazali kod polaznika, zbog toga nema potrebe to pokazivati ponovno. Stranica „Rezultati po polazniku“ bazira se na istom principu. U pretragu se upisuje „OIB“ polaznika kojeg tražimo i pomoću sličnog upita dohvaćamo sve ispite i rezultate na kojima je polaznik bio. Upit za dohvaćanje „Rezultata po polazniku“:

```
$query = "SELECT
pi.ispitID, i.datumIVrijeme, concat(p.ime, ' ', p.prezime) as polaznik,pi.OIBpolaznika, pi.rezultat
FROM polazniknaispitu pi JOIN ispit i ON (pi.ispitID = i.ispitID) JOIN polaznik p ON (p.oib=pi.OIBpolaznika)
WHERE p.oib=$oib";
```

Unos rezultata se obavlja pomoću INSERT upita i njega nećemo prikazati jer je vrlo sličan unosu polaznika. Prikazat ćemo samo formu za unos rezultata jer ima padajući izbornik pomoću kojeg smo osigurali da unos bude formuliran po našoj želji tj. da se slaže s bazom. Naravno, to je osigurano i unutar same baze za svaki slučaj.



Slika 31: Unos rezultata ispita

6. Zaključak

Tema ovog rada je bila implementacija baze podataka za potrebe autoškole u sustavu MySQL. U radu je opisan način izrade same baze, tablica te ograničenja. Za bazu je također izrađena CRUD aplikacija kojom je moguće upravljati bazom. Aplikacija je izrađena pomoću PHP-a i HTML-a, a za lokalno izvršavanje iskorišten je XAMPP.

Baze podataka su sastavni dio našeg života i ovaj rad je samo zagrebao površinu te teme. U radu je prikazana izrada jedne baze za potrebe autoškole u SUBP MySQL. MySQL je jedan od mnogih sustava za upravljanje bazama podataka. Baza je mogla biti izrađena u bilo kojem sustavu, ali je odabran MySQL kao poznatiji i popularniji za Web. Ovaj rad je samo jedan od mnogobrojnih radova i primjera za taj sustav. Mnogi su pokušali ukinuti ili na neki način zamijeniti relacijske baze podataka, ali to je vrlo teško i relacijske baze podataka će se u budućnosti zasigurno koristiti. MySQL kao jedan od najpopularnijih će također samo rasti i razvijati se još više, zahvaljujući svojim autorima tj. njihovom trudu i želji za poboljšanjem, ali i jako velikoj zajednici koja se je okupila oko MySQL-a na razno raznim forumima.

Pišući ovaj završni rad shvatio sam koliko su baze podataka bitne za našu svakodnevicu. Ovaj rad je samo mali dio svega što je moguće napraviti sa sustavom MySQL. Naravno MySQL nije jedini sustav ali je u većini slučajeva prvi izbor, kako drugima tako i meni. Drago mi je što sam imao ovako zanimljivu temu i veselim se vidjeti gdje će nas razvoj sustava za upravljanje bazama podataka dovesti.

Popis literature

[1] What is MySQL, MySQL 8.0 Reference Manual, preuzeto 17.8.2020. s https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html

[2] MySQL, Wikipedia, preuzeto 17.8.2020. s https://en.wikipedia.org/wiki/MySQL

[3] MySQL Installer, MySQL 8.0 Reference Manual, preuzeto 30.8.2020. s https://dev.mysql.com/doc/refman/8.0/en/windows-installation.html

[4] SQL naredbe, w3schools, preuzeto 30.8.2020. s https://www.w3schools.in/mysql/ddl-dml-dcl/

[5] MySQL Workbench intro, MySQL Workbench Manual, preuzeto 20.8.2020. s https://dev.mysql.com/doc/workbench/en/wb-intro.html

[6] MySQL ACID, MySQL 8.0 Reference Manual, preuzeto 30.8.2020. s https://dev.mysql.com/doc/refman/8.0/en/mysql-acid.html

[7] Insausti Sebastian, „Moving from MySQL 5.7 to MySQL 8.0 - What You Should Know“, Several nines, 2020, preuzeto 4.9.2020. s https://severalnines.com/database-blog/moving-mysql-57-mysql-80-what-you-should-know

[8] MySQL Data Types, w3schools, preuzeto 21.8.2020. s https://www.w3schools.com/sql/sql_datatypes.asp

[9] Rabuzin Kornelije, „SQL – NAPREDNE TEME“, Fakultet organizacije i informatike, Varaždin, 2014.

[10] Create Trigger in MySQL, MySQL Tutorial, preuzeto 5.9.2020. s https://www.mysqltutorial.org/create-the-first-trigger-in-mysql.aspx/

[11] Rabuzin Kornelije, „Uvod u SQL“, Fakultet organizacije i informatike, Varaždin, 2011.

[12] Maleković M., Rabuzin K., „Uvod u baze podataka“, Fakultet organizacije i informatike, Varaždin, 2016

Popis slika

Popis tablica