Izrada baze podataka za Gradsku knjižnicu u Ivanić-Gradu

Svetlečić, Luka

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:770415

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Luka Svetlečić

IZRADA BAZE PODATAKA ZA GRADSKU KNJIŽNICU IVANIĆ-GRAD

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Luka Svetlečić

Matični broj: 35918/07–R

Studij: Informacijski sustavi

IZRADA BAZE PODATAKA ZA GRADSKU KNJIŽNICU IVANIĆ-GRAD

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Kornelije Rabuzin

Varaždin, rujan 2022.

Luka Svetlečić

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

Ovaj završni rad bazira se na izradi baze podatka za potrebe Gradske knjižnice u Ivanić-Gradu. U radu se polazi od osnovnih koncepata baza podataka, zatim se dolazi do glavnih koncepata modeliranja i izrade modele prema potrebama knjižnice. U teorijskom dijelu, objašnjeni su koncepti relacija, baza podataka i koncepti modeliranja, a to su konceptualne, logičke i fizičke metode te sama logika primjene navedenih modela. Kroz praktični dio rada, implementirana je baza podataka prema finalnom modelu unutar MySQL sustava, a za potrebe čijeg modeliranja je korišten MySQL Workbench. U radu se nalaze popisi svih kreiranih tablica, sama relacijska shema te da bih se zaokružilo djelovanje knjižnice napravljeni su okidači nad bazom podataka. Primjer korištenja baze podataka realiziran je kroz „MVC framework“ i forme koje pružaju radno okruženje korisniku.

Ključne riječi: knjižnica, baze podataka, SUBP, modeli, okidači, relacija, tablica, „MVC framework“, SQL, PHP.

Sadržaj

1. Uvod

Knjižnica u Ivanić-Gradu osnovana je 1877. godine kao čitaonica te je predstavljala središnju kulturnu ustanovu u Ivanić-Gradu kao svrhu uzdizanja izobrazbe građanstva. Prolaskom vremena, djelatnosti knjižnice proširuju se u raznolika predavanja i okupljanja te se od 1925. utemeljuju i kazališne predstave. Danas ju poznajemo kao modernu knjižnicu, knjižnicu koja nudi potporu i učenicima, ali i studentima kroz razne mogućnosti i djelatnosti vezanih uz pretraživanje informacije prilikom izrade referata, seminara, završnih i diplomskih radova te kao korištenje računala. [1]

Samim time što sam član knjižnice, motivacija upravo proizlazi iz navedenog. Naime, glavni fokus rada je na bazi podataka koja je prijeko potrebna za pohranu i praćenje svih promjena koje se događaju. Važnost same baze podataka iznimno je velika u današnjem vremenu jer postoji velika potreba za pohranom podataka, kako za svaku drugu aplikacijsku domenu, tako i za knjižnicu. Knjižnica u Ivanić-Gradu sadrži preko 35 000 knjiga, kataloga itd., samim time dolazimo do potrebe digitalnog zapisa podataka o svakoj zato što je nemoguće skladištiti takav broj podatka na papiru. Baza podataka nam također omogućuje analiziranje same, što ujedno za našu domenu može biti jako korisno za željeni ispis i pregled podataka. S druge strane, važno je napomenuti da su baze podataka zahtjevne za održavanje i da o njima ovisi rad aplikacija koje koriste tu bazu podataka.

Cilj ovog rada je prikazati složenost jednog sustava iz okoline kako koristi stvarne informacije/podatke koji su nužni za Gradsku knjižnicu Ivanić-Grada koristeći relacijske baze podataka, a ujedno nadograditi s novim podatcima i prikazati ih putem sučelja. Za potrebe ove arhive korišten je MySQL sustav za upravljanje bazom podataka te ujedno MySQL Workbench kao alat za modeliranje.

2. Općenito o bazama podataka

Baza podataka je organizirana zbirka logički povezanih, pretražljivih i međusobno ovisnih podataka (informacija), pohranjena u nekom od računalno čitljivih medija. [2] Napredak tehnologija razvoja baza podataka i pohrane podataka jako je značajan u zadnjih 20-tak godina, naime danas osim tekstualnih i brojčanih vrijednosti možemo pohranjivati slikovne, zvukovne i videozapise kao sadržaje. Unatoč tome, dolazimo do sustava za rukovanje bazom podataka kojega nazivamo sustav upravljanja bazom podataka (SUBP). Sustav u pravilu služi za kreiranje i upravljanje velikim količinama podataka i dopušta siguran opstanak tijekom dugog vremenskoga razdoblja. Postoji nekoliko stvari za koje su suvremeni SUBP namijenjeni: [9, str. 1]

- Dopuštaju korisnicima stvaranje novih baza podataka i specifikacije njihove sheme (logičke strukture) koristeći specijalizirani jezik za definiciju podataka.
- Daje korisnicima mogućnost postavljanja upita i izmjene podataka koristeći prikladan jezik koji se često naziva „query language“ ili „data manipulation language“.
- Podržava pohranu velikih količina podataka tijekom dugo vremena, omogućujući učinkoviti pristup podatcima kroz upite i izmjene baze podataka.
- Omogućuje trajnost, oporavak baze podataka u slučaju grešaka ili namjerne zlouporabe.
- Kontrolirani pristup podatcima odjednom, bez dopuštanja neočekivanih interakcija među korisnicima i ne dopušta djelomični rad na podatcima, ali ne u potpunosti (eng. „atomicity“)

Sustave prvi puta susrećemo u 20. stoljeću. Sustavi tog vremena nisu mogli izvršavati sve navedene aktivnosti koje mogu u današnje vrijeme, također bila je veća potreba za održavanjem sustava u smislu vizualizacije podataka kao i pohrane. Upravo takve karakteristike odgovarale su hijerarhijskim i mrežnim bazama podataka za koje možemo reći da su generacija ispod onih koje danas poznajemo, a to su relacijske baze. Unatoč tome što takvi modeli nemaju mogućnosti kakve nam danas pružaju suvremeni SUBP, mogli smo ih pronaći u bankovnim sustavima i raznim rezervacijama kao što za avionske kompanije itd. Najveći problem dotadašnjih baza podataka bio je u tome što nisu podržavali „high-level query language“, odnosno koristili su se jezici nižih razina kao što je CODASYL („Committee on Data Systems and Languages“) koji je radio pomoću naredbi kojima bih korisnik „skakao“ s elementa na element podataka kroz graf pokazivača među tim elementima.

2.1. Relacijske baze podataka

Baze podataka kakve danas poznajem uglavnom su relacijske baze podataka zahvaljujući Ted Codd-u koji govori o relacijskom modelu za skladištenje podataka potrebnih za banku. T. Codd prvi govori da programer baza podataka mora imati pregled nad podatcima u tablicama, zvanim relacijama. Također, postavlja teorijske temelje za konzistentno semantičko postupanje s podatcima i rješava problem redundancije podataka koji su dotada bili česti.[9, str. 70] Programer u takvom pristupu koristi „high-level“ jezik, odnosno „Structured Query Language“ poznat nam još kao SQL jezik.

Relacija je definirana tablica sa stupcima(atributima) i redovima (n-torkama). Često se za riječ relacija koristi naziv „tablica“. Unatoč tome, moramo biti oprezni jer nije svaka tablica relacija, što podrazumijeva da relacija definira što će sadržavati svaki red tablice. Samim time, podatke pohranjujemo kao instance (zapise) u relaciji.

Stupci prema teoriji relacijskih baza podataka moraju imati sljedeća svojstva:[10, str.90]

- Jedinstveno ime unutar relacije,
- Domenu,
- Poredak nije bitan.

Ime atributa specificirano je unikatnošću na razini jedne relacije, ali nije nužno držati se tog pravila na razini više relacije jer je moguće da više tablica sadrži ime istog atributa. Problem dolazi kod pristupa istoimenim atributima u različitim relacijama, ali to se obično rješava unutar alata koji ima odgovarajuću sintaksu. U većini alata, definira se relacija pa pomoću karakterističnog znaka (točke) navede traženi atribut. Iduće svojstvo je domena. Ona je u pravilu jako dobro definirana kroz SUBP, odnosno koje tipove podataka. Podržava ili sustav daje mogućnost postavljanja domene nad tipom podataka. Teoretski gledano, vrijednosti u stupcu nalaze se u definiranoj jednoj domeni. Također, neizbježno svojstvo je svojstvo da stupci ne moraju biti poredani zadanim redoslijedom, što je jako zanimljivo svojstvo jer nam dopušta drugačije rasporede stupaca s time da ne gubimo prethodno navedena svojstvo relacija.

Relacijski pristup određuje i svojstva za redove: [10, str. 91]

- Jedna vrijednost,
- Unikatnost,
- Primarni ključ,

- Ne poštuju poredak unutar relacije.

Jedna vrijednost odnosi se na to da relaciju možemo promatrati i kao matricu, stoga na sjecištima redova i stupaca na može postojati više vrijednosti od jedne. Unikatnost jednog zapisa nam je važna da ne bi dolazilo do nepotrebnog ponavljanja vrijednosti. Nema smisla zapisivati n-torku istih vrijednosti atributa. Primarni ključ je stupac ili kombinacija kolona s vrijednošću koje jedinstveno opisuju svaki red. Primjenom svojstva primarnog ključa ujedno rješavamo problem unikatnosti, odnosno definiramo unikatnu vrijednost nad svakim zapisom. Kao što nam raspored stupaca nije definiran, isto tako vrijedi i za redove, ali važno je naglasiti da nije moguće mijenjati raspored stupaca i redova u isto vrijeme, samim time gubimo integritet relacije.

2.2. Primjena relacijskoga pristupa

Na slici 1. prikazano je kreiranje relacije. Važno je napomenuti nekoliko važnih pojmova kod kreiranja relacija. Prije svega, domena atributa dio je realnog svijeta i označava ograničenja za svaki atribut zasebno. Na primjer, možemo reći da je ograničenje nad domenom atributa „JMBAG“ to da vrijednost mora biti točno 10 znamenki jer je tako definirana u realnom svijetu. Ono što je važno opaziti je to da atribut poprima vrijednost iz domene pa ako istoimeni atribut koristimo u izradi druge relacije, on mora imati isto značenje. Na primjer, uzevši u obzir atribut „E-mail“ koji ujedno može pripadati nastavniku i studentu, po pravilu ima istu domenu promatranja, a ograničenje bi bilo da mora sadržavati znak „@“ u oba slučaja.

Nadalje, važan pojam je entitet, koji se još naziva objektom promatranja unutar zadane sheme. Entiteti imaju širok raspon, što znači da mogu biti stvari, bića, pojave, odnosno sve ono za što možemo prikupiti podataka, štoviše iz čega imamo informaciju. Odabir entiteta u pravilu je definiran razinom promatranja nad zadanom domenom promatranja, odnosno sam odabir ovisit će kako gledamo na neki problem.Na slici 1. prikazan je entitet STUDENT, odnosno jedinica promatranja.

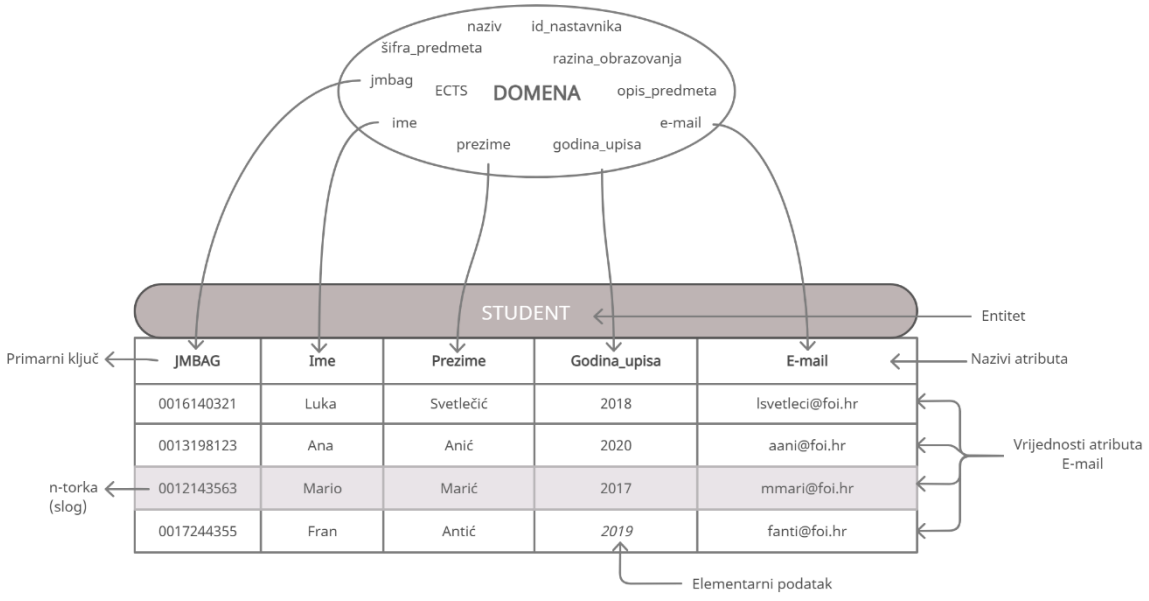
Nakon određivanja entiteta, možemo početi razmišljati koji atributi iz domene su korisni za takvu relaciju te prema pravilu dobivamo sve potrebne atribute za navedeni entitet u našem slučaju: „JMBAG, Ime, Prezime, Godina_upisa, E-mail“. Atributi su najjednostavnije rečeno svojstva entiteta. Svojstva se dodjeljuju prema važnosti za pojedini entitet, odnosno važno je jesu li ona relevantna za njega. Elementarni podatak najmanji je element relacijskog modela, odnosno to je podatak koji je predstavljen u jednoj ćeliji. Drugim riječima, ne

možemo ga rastaviti na manje dijelove a da ne izgubimo njegovo značenje. Na primjer, elementarni podatak „2019“, iz atributa „Godina_upisa“ ima svoje konačno značenje i ne može ga se rastaviti na manje dijelove, a da ne izgubimo njegovo početno značenje.

Unatoč tome da odabir entiteta i atributa ovisi o razini promatranja problema i dalje u specifičnim situacijama može doći do zabune. U takvoj situaciji vodimo se pravilom je li moguće da promatrani entitet ima više atributa od jednoga. Ako da, to je sigurno entitet, odnosno ako promatrani entitet prema našoj domeni promatranja sadrži jedan ili manje atributa, njegova vrijednost se pohranjuje kao atribut drugog entiteta. [4, str.28]

Glavna svojstva stupca i redova sada će se prikazati u primjeru. Naveli smo da prema relacijskog terminologiji njih nazivamo vrijednost jednog atributa i slog (n-torka). Slog je upravo jedan zapis instance za navedenu relaciju, tako na slici 1. imamo sve relevantne podatke o zapisu studenta „Maria Mariča“. S druge strane, stupac nam predstavlja sve vrijednosti jednog atributa pa nad slikom 1. atribut „E-mail“ sadrži popise mailova svih studenata.

Idući pojam su same veze između entiteta koje predstavljaju odnos navedenih, točnije kako jedan entitet utječe na drugi i obratno, što prikazujemo vezom. Proces pronalaženja veza odvija se kroz čitanje poslovne domene ili neke specifikacije aplikacije koju korisnik iznosi. Postupak pronalaženja odvija se tako da imenice postaju entiteti dok glagoli veze ne moraju. Postoje veze koje zahtijevaju posebnu relaciju, a takve primjere objasnit ćemo u poglavlju logičkoga razvoja baze podataka.



Slika 1: Primjer kreiranja relacije STUDENT. [samostalna izrada]

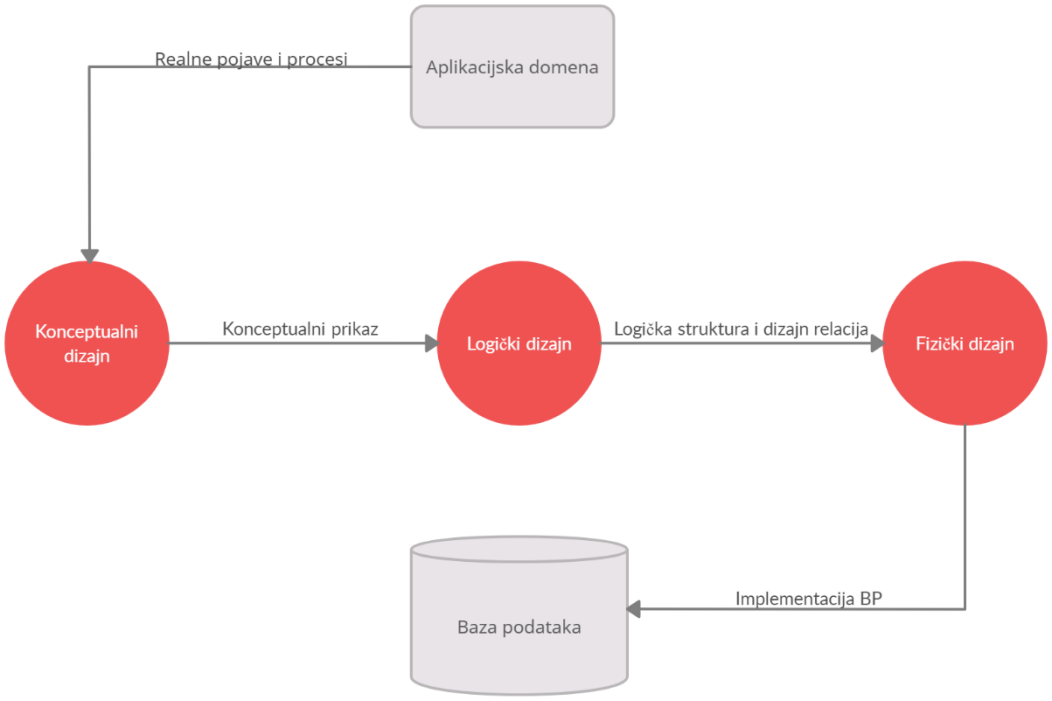
3. Metodologija razvoja relacijske baze podataka

Metodologija, vezana uz razvoj dizajna baze podataka je neophodna. Ona nam omogućuje planiranje, upravljanje, kontrolu i razvoj same baze podataka, što znači da ćemo istu koristiti za potrebe razvoja baze podataka u našem radu. [6]

Metodologija se sastoji od 3 glavne faze, a to su:

- Konceptualni dizajn baze podataka - izrada konceptualnog prikaza kroz identificiranje entiteta, atributa i veza.
- Logički dizajn baze podataka – prijevod konceptualnog prikaza u logičku strukturu baze podataka i relacijski dizajn.
- Fizički dizajn baze podataka – korištene tehnike kako će se logička struktura implementirati u zadanom okruženju (SUBP).

Redoslijed faza je konačan i standardiziran te ga po tom rasporedu treba i slijediti. Odnosno, metodologija dizajna baze podataka vodi dizajnera kroz tehnike prikladne u svakoj fazi da bi kao finalni output procesa dobio suvremenu bazu podataka. [7]



Slika 2: Metodologija razvoja baze podataka [samostalna izrada].

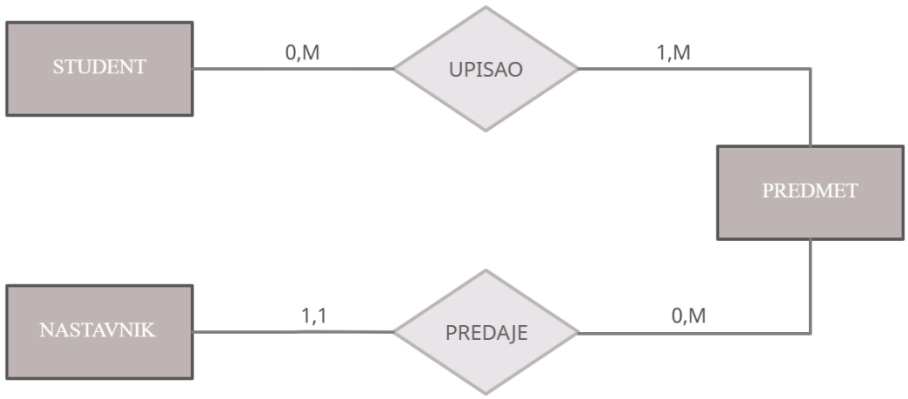
3.1. Konceptualni razvoj baze podataka

Etapa konceptualnog razvoja baze podataka primjenjuje se na promatranoj poslovnoj domeni te je samim time potpuno neovisna o programskim jezicima, sustavom upravljanja baze podataka i drugim fizičkim tehnologijama. Tijekom izgradnje konceptualnog modela izgrađuje se jedna ili više replika konceptualnih podataka poduzeća. Osobina kojom odvajamo konceptualni model (shemu) od ostalih je dinamičnost, odnosno uvijek pruža mogućnost promjene i daje prostor za kreativnost i improvizaciju.

Postoje nekoliko koraka izgradnje konceptualnog modela koji služe kao dobra polazna točka izgradnje temeljnog modela, a to su: [7]

- Prepoznati entitete i vrste veza
- Identificirati i povezati atribute s entitetom i li vrstom veze
- Odrediti domene atributa
- Odrediti primarne i vanjske ključeve
- Provjeriti da li model ima redundance
- Uskladiti model s korisničkim zahtjevima.

Da bi se prikazao konceptualni model korisniku, najčešće se koristi „Chenov“ dijagram. Dijagram prikazuje entitete i njihove veze (imenice, glagole), ali i samu oznaku kardinalnosti veze. Prema pravilu, kardinalnost se označuje u smjeru od jednog do drugoga tipa entiteta i piše se bliže drugom tipu. Notacijom kojom se koristimo je sljedeća: entiteti su prikazani kao pravokutnici, a veze kao rombovi. Možemo zapaziti da nemamo popis atributa na dijagramu, naime to se često radi da bi se dobio što jednostavniji prikaz dijagrama, ali se ujedno u potpunoj dokumentaciji nalazi popratni tekst s popisom atributa kojeg još nazivamo konceptualna shema.



Slika 3: Primjer konceptualnog dijagrama [samostalna izrada].

3.2. Logički razvoj baze podataka

Druga faza projektiranja baze podataka je projektiranje na logičkoj razini. Na kraju ove faze trebamo stvoriti relacijsku shemu sa svim njenim ključnim elementima, dakle shemu koja opisuje logičku strukturu baze podataka u skladu s pravilima relacijskog modela podataka. [4, str. 38] Današnje relacijske sheme su više-manje podržane u svim SUBP i značajno olakšavaju implementaciju baze kojom ćemo se baviti na logičkoj razini. Tijekom izrade logičkog modela, odnosno relacijske sheme i relacijskog modela i popratne dokumentacije, na umu treba imati koliko je model pouzdan, štoviše koliko će model dobro podržati buduće razvoj baze podataka u izradi fizičkog modela.

U poglavlju relacijskog pristupa navedena je važnost primarnog ključa relacije i ostalih atributa relacije, a sad će se to i primijeniti. Da bi lakše odredili primarni ključ i atribute relacije, atribute dijelimo na ključne i neključne. Drugim riječima, ključni atributi predstavljaju unikatnost i za njih smo potpuno sigurni da nećemo narušiti pravilo jedinstvenosti (npr. JMBAG). S druge strane, neključni atributi imaju veliku šansu ponavljanja (npr. Ime). Sukladno tome, odnosu relacija pridodaju se i vanjski ključevi (FK). Vanjskim ključem smatramo atribut koji sadrži kopiju vrijednosti iz svoje primarne tablice gdje se on ujedno nalazi u ulozi primarnoga ključa.

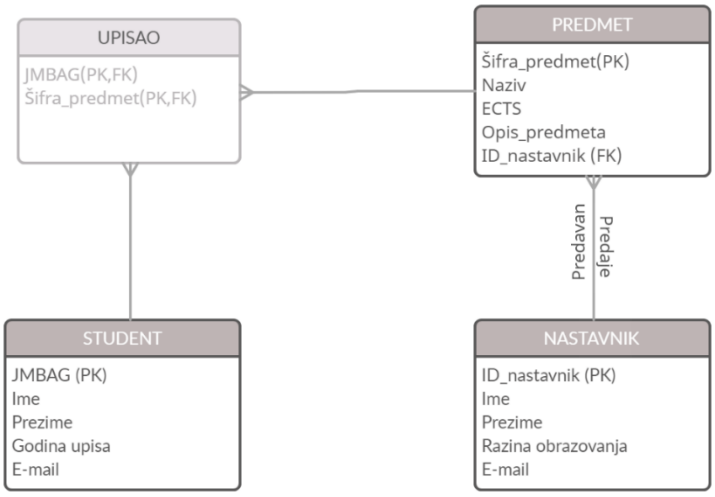
Nadalje važna karakteristika su veze između entiteta. Naime, veze ne smiju više biti apstraktno opisane kao što su bile na konceptualnoj razini, nego ih moramo opisati kao što su one određene prema relacijskoj shemi. Samim time postoje nekoliko vrsta veze: [4, str.25]

- 1 naprema 1 (1 : 1),
- 1 naprema više (1 : M),
- Više naprema više (M : N).

Svaka od veza ima određeni postupak realizacije. Prva od veza je „1 naprema 1“. Veza opisuje da jedna instanca entiteta (E1) mora biti povezana s najviše, ali i najmanje (točno jednim) jednom instancom drugog entiteta (E2). To vrijedi i obratno, odnosno instanca iz drugoga entiteta (E2) ima isto svojstvo prema instanci iz prvoga entiteta (E1). Da bi realizirali vezu potrebno je koristiti ključ, točnije vanjski ključ koji se dodaje proizvoljno jednom entitetu. Ideja gdje je potrebno staviti vanjski ključ nije definirana, ali iskustvo projektanta može puno doprinijeti, odnosno ključ se prema neformalnom pravilu dodaje u

„slabiji“ entitet ili u onaj u kojemu taj atribut predstavlja veću vrijednost. Druga veza je „1 naprema više“ koja opisuje da jedna instanca entiteta (E1) može biti prisutna u više instanci iz drugoga entiteta (E2). Sukladno tome, kada gledamo s druge strane, jedna instanca iz drugog entiteta (E2) može biti povezana samo s jednom instancom iz prvog entiteta (E1). Problem dodavanja ključa u ovoj situaciji poprilično je jasno definiran. Vanjski ključ dodaje se od strane „jedan“ na stranu „više“ jer vanjski ključ iz prvog entiteta (E1) se smije pojaviti više puta u zapisu drugog entiteta (E2). Preostali oblik veze je veza „Više naprema više“ koja označuje da više vrijednosti instanci prvog entiteta (E1) može biti povezano s više instanci drugog entiteta (E2). Gledajući s druge strane, više zapisa drugog entiteta (E2) može poprimiti više vrijednosti zapisa iz prvog entiteta (E1). Određivanje vanjskog ključa veze „Vipe naprema više“ još se naziva postupkom normalizacije. Poanta korištenja normalizacije je osigurati da pozicija relacija ima minimalan, ali opet dovoljan broj atributa potrebnih za podršku zahtjevima podatcima poduzeća. [8] Drugim riječima, normalizacija je način eliminiranja redundancije u relacijskoj bazi podataka pomoću definiranja normalnih formi u kojoj se one nalaze i naposljetku način prevođenja iz niže forme u višu, što ujedno ukazuje i na bolju. Važna stavka razvoje normalne forme je ta što je ona većeg stupnja to je redundanca manja. Da bi postigli što manju redundancu, koristimo se zavisnostima.

Unatoč velikom rasponu normalnih formi koje su nam poznate, mi ćemo se koristiti „trećom normalnom formom“, odnosno finalni model je realiziran prema njoj. Relacija se nalazi u trećoj normalnoj formi (3NF), ako se nalazi u drugoj normalnoj formi (2NF) i ako nijedan neključni atribut nije tranzitivno zavisan o bilo kojem ključu relacije. [6, str.108] Naime, definiciju je lakše shvatiti tako da svaki neključni atribut mora zavisiti samo o ključu, odnosno o cijelom ključu i ni o čemu više.



ž

Slika 4: Primjer logičkog modela [samostalna izrada].

3.3. Fizički razvoj baze podataka

Unatoč tome što je metodologija konačnog rasporeda, nakon napravljenih ER modela (logički model), relacijske sheme vezana uz logički koncept razvoja, sama slijedi fizički razvoj koji pretvara logički u stvarni, točnije modela podataka.

Cilj posljednje faze razvoja baze podataka je stvoriti fizičku shemu baze podataka. Upravo je to glavna, temeljna faza fizičkoga razvoja baze podataka. Fizičku shemu još nazivamo fizičkim modelom, a predstavlja opis stvarne fizičke organizacija podataka, točnije baze podataka realizirane na medijima za pohranu memorijskog prostora. [6, str.129] Fizički model sastavljen je od naredbi zapisanih u SQL programskom jeziku koje SUBP prepoznaje i pokreće akcije. Korišteni alat u radu je MySQL workbench o kojemu ću detaljnije govoriti kasnije u radu. Uzimajući u obzir da je ovo zadnja faza razvoja baze podataka, ona je ujedno najmanje kompatibilna ispravljanju pogrešaka, jer kao što smo naveli prelaskom iz faze u fazu pogreške bi se trebale smanjiti odnosom klijenta i dizajnera koja je u ovoj fazi značajno manja nego prijašnje.

Naveli smo da se fizički razvoj temelji na SUBP te njegovog djelovanju na memoriji, a ona je zasnovana na zaštiti podataka koja u pravilu mora biti jako dobro definirana relacijama, organizacijom datoteke i indeksima koji se koriste za pristupanje podatcima, također ograničenjima integriteta i ostalim sigurnosnim mjerama. [10]

Kao glavnu razliku između logičkog i fizičkog modela uzimamo pojam implementacije, odnosno stvaranja stvarnog modela. Dobar dojam o razlici između navedenih može nam pokazati da se logički razvoj bazira na pitanju „što“, dok fizički odgovara na „kako“. Dobar dizajner mora biti svjestan mogućnosti SUBP-a te ih tako najbolje moguće primijeniti za razvoj što bolje baze podataka.

3.4. MySQL

Vrijeme je da se upoznamo i sa sustavom koji ćemo koristiti u radu, a to je MySQL. MySQL je sustav za upravljanje relacijskim bazama podataka (RDBMS) koji je razvio Oracle i koji se temelji na strukturiranom jeziku upita (SQL) [11]. Već smo spomenuli da se u radu bavimo isključivo relacijskim bazama podataka koje MySQL sustav podržava. Pored velikog broj funkcionalnosti, navest ću neke. MySQL sustavu je kompatibilan sa svim glavnim platformama kao što je Linux operacjiski sustav pa je samim time jednostavan za korištenje i migraciju koda. Osim rada s platformama, MySQL je „open-source“ sustav, što znači da daje mogućnost korisniku pravo korištenja koda. Kao što smo naveli, sustav je namijenjen za rad

s relacijskim tipom baze podataka i samim time omogućuje proceduralno programiranje, „control-flow“ mehanizme, odličnu podršku itd [12]. Sustav MySQL koristi SQL jezik koji radi na principu naredbi koje su navedene u tablici 1.

Definiranje podataka	Manipulacija podataka	Pripremljene izjave	Kontrola transakcije	Kontrolapodataka
CREATE DATABASE/EVENT/ FUNCTIONS/INDEX/ PROCEDURE/TABLE/ TRIGGER/VIEW	SEELCT	PREPARE	BEGIN/END	USER
ALTER DATABASE/ EVENT/FUNCTION/ PROCEDURE/TABLE	UPDATE	EXECUTE	COMMIT	SET DEFAULT
DROP VIEW/DATABASE/ EVENT/FUNCTION/ INDEX/PROCEDURE/ TABLE/TRIGGER/VIEW	INSERT VALUES	DEALLOCATE PREPARE	ROOLBACK	ROLE
RENAME TABLE	DELETE		SAVEPOINT	GRANT
TRUNCATE TABLE	UPDATE			REWOKE

Tablica1: SQLnaredbe. [samostalna izrada]

Tablica prikazuje osnovne, ali ne i sve naredbe za korištenje SQL jezika unutar MySQL sustava. Mi ćemo se koristiti ponajviše s naredbama iz prva dva stupca, naredbama za definiranje i manipulacija podatcima.

4. Opis aplikacijske domene

Do sada je u radu objašnjen teorijski dio izrade baze podatka da bi se shvatio sam koncept i potreba realizacije baza podataka. Vratimo se na domenu aplikacije koja je knjižara, odnosno moramo omogućiti pohranu i skladištenje podataka vezanih uz poslovanje Gradske knjižnice u Ivanić-Gradu. Knjižnica broji veliki broj knjiga, odnosno podataka vezanih uz istih pa iz tog razloga koristimo alat „MySQL Workbench“ koji će pomoću drugih tehnologija pružiti korisničko sučelju administratoru knjižnice. Sučelje je namijenjeno knjižničaru koji će predstavljati poveznicu između članova knjižnice i same knjižnice. Prema tome knjižničar će imati funkcionalnosti pregleda, dodavanja, brisanja, pretraživanja i editiranja podataka unutar sustava.

4.1. Osnovna svojstva knjižnice

Svaki autor u bazi ima svoj popis knjiga s kojima ga obvezuje autorstvo koje je definirano prema razinama autorstva zbog njegove sigurnosti. Autora krase njegov popis knjiga, odnosno njegovo djelovanje na području pisanja koje je karakteristično za njega. Knjiga sadrži sve primarne podatke koji opisuju nju samu. Ako je knjiga prevedena, moramo znati prevoditelja koji je knjigu preveo na pisani jezik zbog mogućnosti dodavanja stranih autora u bazu podataka. Uzimajući u obzir da svaki naslov knjige krasi njeni „ISBN“, moramo znati i koliko je primjeraka kopija svake knjige da bi se znao izračunati stvarni broj postojećih knjiga unutar knjižnice koje će imati inventarni broj. Svaka knjiga ima svoju izdavačku kuću, odnosno kuću koju je knjiga izdala, dok izdavačka kuća dolazi iz grada odnosno države. Informacija o tome u kojem gradu se nalazi knjižnica može biti korisno kod zaduživanja novih knjiga. Uzimajući u obzir da izdavačka kuća ima svoji grad, isto možemo reći i za knjižnicu i same članove knjižnice. Nadalje, moramo uzeti u obzir zaposlene u knjižari, moramo znati u kojoj knjižari oni rade, samim time na kojoj poziciji su zaposlenici te njihovu razinu obrazovanja.

4.2. Specifična svojstva knjižnice

Tiejkom proučavanja domene djelovanja knjižnice, primijetio sam nekoliko funkcionalnosti koje odvajaju GK Ivanić-Grad od drugih pa samim time ta svojstva nužno vrijede za GK Ivanić-Grad, ali za neku drugu ne moraju, a to su sljedeća:

- Jedna od obilježja knjižnice je ta da u biti postoje dvije knjižnice, jedna smještena u Ivanić-Gradu, a druga u općini Kloštar Ivanić. Samim time moramo unijeti osnovne podatke za obadvije.
- Da bi znali u kojoj knjižnici se koja knjiga nalazi, morat ćemo povezati svaku knjigu s jednom od dvije knjižnice, ovisno gdje se nalazi.
- Potrebno je osmisliti evidenciju stanja knjiga, odnosno da li je knjiga u stanju posudbe ili da je knjiga posuđena ili se nalazi u knjižnici. Ovo svojstvo je jako važno da bi zaposlenici znali trenutno stanje u knjižari, a samim time članovi mogli provjeriti dostupnost.
- Broj zaposlenih unutarknjižnice je nešto manji od 10, samim time moramo uzeti u obzir da su podijeljeni na dvije knjižare. Ono što je važno spomenuti je to da postoji hijerarhija između zaposlenika, odnosno imamo voditelja knjižnice i pomoćnog knjižničara.
- Knjižnica broji tri vrste članarine. Prva vrsta članarine je obična odrasla koja dozvoljava članu da posudi do 3 knjige istovremeno i iznosi 50 kn godišnje: druga je dječja članarina koja je definirana za mlađu publiku, ne stariju od 16 godina, a iznosi 25 kn godišnje i također obavezuje da član može posuditi do 3 knjige istovremeno. Zadnja vrsta članarine je neograničena, naplaćuje se 100 kn godišnje i ne obavezuje korisnika usluga knjižnice broj posuđenih knjiga.
- Sustav kažnjavanja, ako član knjižnice vrati knjigu poslije predviđenog vremena vraćanja, uračunava se pola kune po danu.
- Ne uzimajući u obzir vrstu knjige i vrstu članarine, svaka knjiga se posuđuje na dva mjeseca.
- Radno vrijeme knjižnica je svakim radnim danom od 8 sati do 19 sati, dok vikendom radi samo subotom i to od 8 do 13 sati.
- Knjižnica osim domaćih izdavačkih kuća, broji i strane, kao što su iz Srbije, Bosne i Hercegovine, Njemačke i Engleske. Samim time knjiga ne mora nužno biti na hrvatskom jeziku.

5. Konceptualno modeliranje Gradske knjižnice

Kao što smo ranije spomenuli, konceptualni model baze podataka temelji se na konceptima i vezama među koncepta. Ono što mu daje najveću karakteristiku je jednostavnost i trivijalnost, a stvaramo ga iz kvalitetne opservacije problema.

5.1. Definiranje entiteta i veza

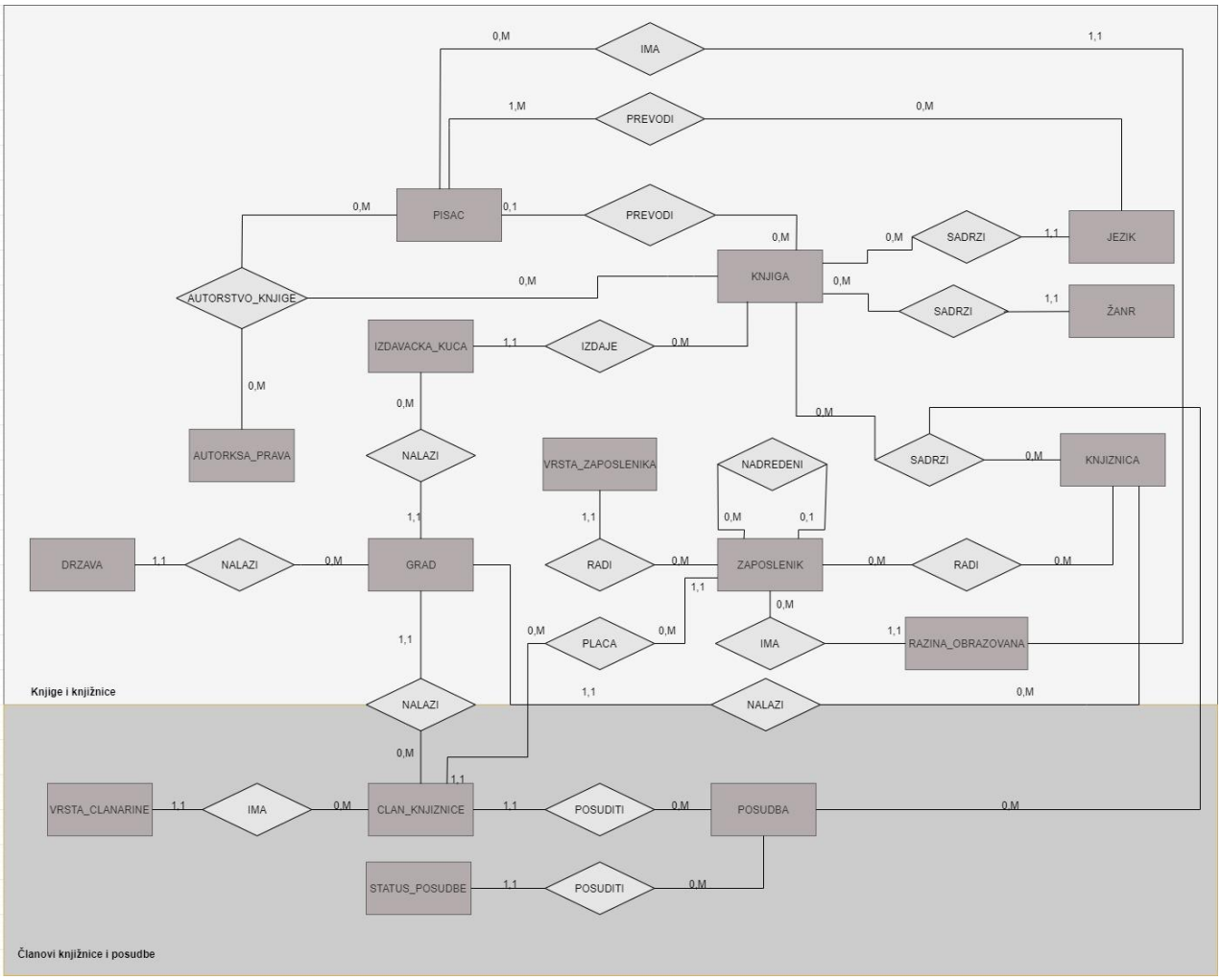
Tijekom dosadašnjeg iskustva, naglasio bih da možda i najteži dio modeliranja baze podataka stoji u definiranju entiteta iz domene. Problemi s kojima sam se susreo bili su sljedeći: prvi problem bio je odrediti razinu promatranja domene, drugim riječima odrediti razinu s kojom ćemo pokriti sve zahtjeve iz domene problema. Naime, baza podataka mora pokriti domenu djelovanja, ali također model mora biti prilagodljiv budućim promjenama i zahtjevima. Uzimajući u obzir razinu promatranja i definiranu domenu, dobio sam prividnu sliku promatranja te pronašao i imenovao entitete. Drugi problem s kojim sam se susreo je bio da li je entitet relevantan, odnosno nužan za trenutno stanje i da li je entitet na ovoj razini možda svojstvo nekog drugog entiteta. Na primjer, relevantan entitet promatranja svakako će biti AUTOR, no pitanje je da li je PREVODITELJ entitet ili jednostavno može biti svojstvo entiteta KNJIGA. Po meni oba slučaja spadaju u korektno rješenje, u prvom slučaju radilo bi se o otvaranju novoga entiteta prevoditelj, dok u drugom slučaju prevoditelj bi bio svojstvu svakoga naslova knjige ako je prevedena. Osobno sam se odlučio na prvu opciju radi toga jer imamo veći potencijal za buduće prilagodbe i postoje više atributa koji su značajni za prevoditelja, ali nisu relevantni za autora. Uzevši u obzir navedeno, možemo očitati sljedeće neophodne entitete: KNJIGA, IZDAVACKA_KUCA, PISAC, ZAPOSLENIK, POSUDBA, GRAD, KNJIŽNICA, CLAN_KNJIZNICE. Iščitavanjem glavnih entiteta moramo dodati popratne entitete da bih zaokružili domenu djelovanja, to su sljedeći entiteti: AUTORSKA_PRAVA, ŽANR, JEZIK, DRŽAVA, VRSTA_ZAPOSLENIKA, RAZINA_OBRAZOVANJA, STATUS_POSUDBE, VRSA_CLANARINE.

Nadalje, nakon identificiranih entiteta moramo uzeti u obzir veze koje će ih karakterizirati i davati ograničenja nad njima. Veze, unatoč tome što daju također veliki prostor djelovanja, ograničene su entitetima, to jest moramo povezati entitete kvalitetno i racionalno. Ono što sam odmah uzeo u obzir je svojstvo veza koje zovemo opcionalnost na svakoj strani entiteta jer sam primijetio da nisu sve veze nužne. Na primjer, entitet CLAN_KNJIZNICE sigurno će sadržavati atribut iz entiteta VRSTA_CLANARINA te nam u toj situaciji nije potrebna direktna

poveznica između VRSTA_CLANARINA i plaćanja članarine nego ju jednostavno možemo pročitati iz entiteta CLAN_KNJIZNICE.

5.2. Model konceptualnog modeliranja

Naime, spomenuli smo da dobar konceptualni model nastaje iz domene promatranja pa tako za našu domenu knjižare možemo primijeniti konceptualno modeliranja i navedene pojmove pa dobivamo sljedeći konceptualni model koji je izrađen u Diagrams.net alatu.



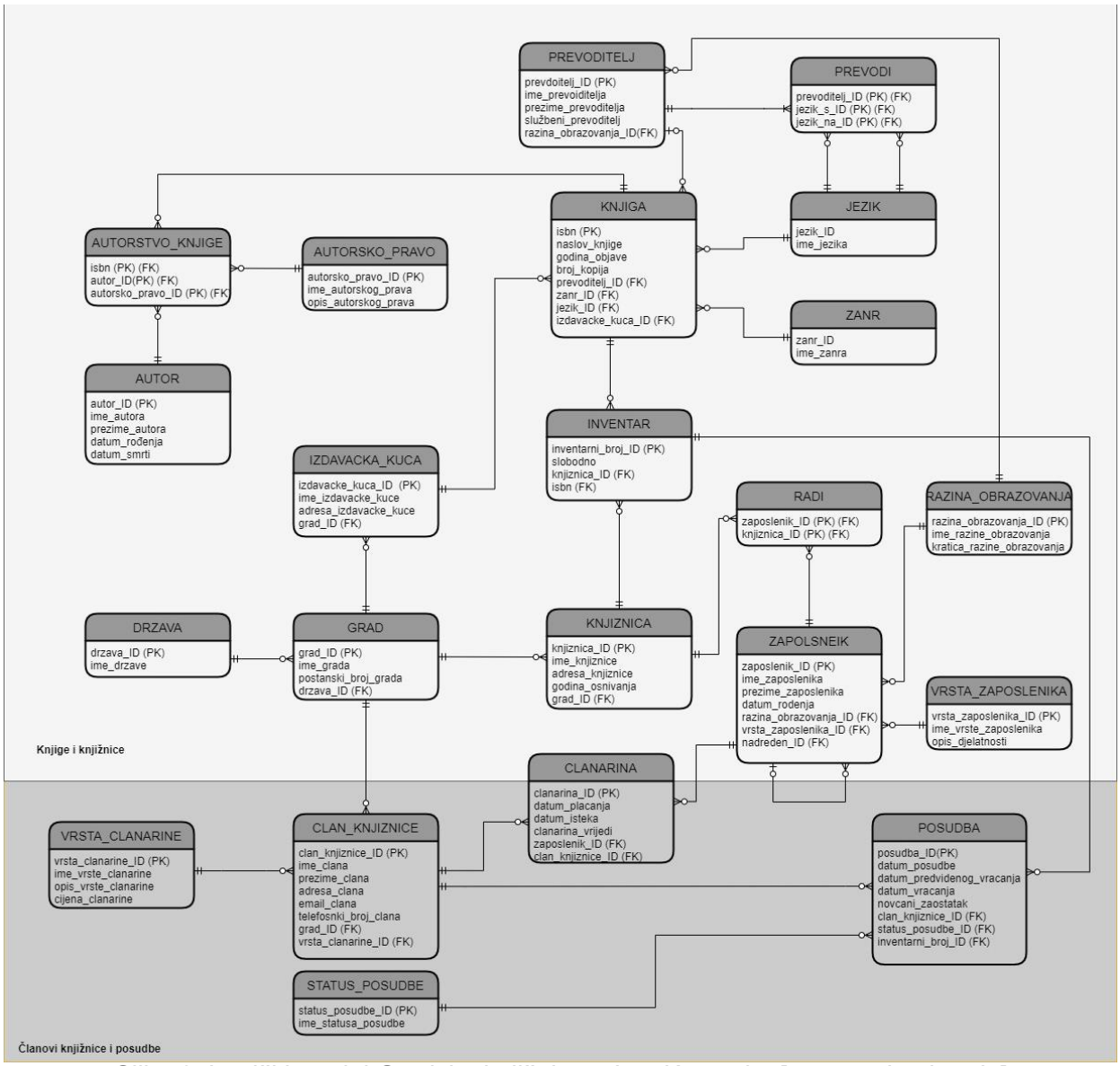
Slika 5: Konceptualni model Gradske knjižnice u Ivanić-Gradu. [samostalna izrada]

Kao što smo već nekoliko puta napomenuli, konceptualni model mora biti jednostavan za čitanje da bi pružio lako razumijevanje i lagano korigiranje, a samim time ga uspješno može pročitati zaposlenik u knjižnici pa i član knjižnice. U modelu, pravokutnicima su označeni entiteti, a veze rombovima. Kroz ovaj model napravili smo već veliki korak u modeliranju baze podataka, trenutno znamo potrebne entitete, potrebne veze između njih, a i definirali smo kardinalnost te možemo prijeći na izradu logičkog modela.

6. Logičko modeliranje Gradske knjižnice

Onime čime ćemo se baviti u ovom poglavlju radu je izrada logičkog modela za Gradsku-Knjižnicu u Ivanić Gradu. Točnije, moramo odrediti atribute entiteta i provesti normalizaciju nad modelom. Prije svega, logički model mora biti razumljiv developerima i mora pružit „stepenicu iznad“ naprema konceptualnog modela po pitanju informacija o budućoj bazi, a samim time moramo i definirati nove entiteta koje ćemo prvo napraviti. Najprije možemo pogledati finalni logički model pa ćemo proći postupak kreiranja i specifične situacije.

6.1. Model logičkog modeliranja



Slika 6: Logički model Gradske knjižnice u Ivanić-Gradu. [samostalna izrada]

Vjerojatno najuočljivija razlika između logičkog i fizičkog modela je povećani broj entiteta. Razlog tome se krije u potrebi otvaranja novih entiteta na vezama „više naprema više“. Budući da smo označavali veze s rombovima, puno nam je jednostavnije očitati postoji li potreba za otvaranjem nove tablice. Za primjer možemo uzeti potreba uvođenja entiteta AUTORSTVO_KNJIGE umjesto istoimene veze. U ovom slučaju čak imamo tri veze tipa „više naprema više“, a činit će je entiteti AUTOR, AUTROSKA_PRAVA i KNJIGA te je po tome ona nužan entitet koju moramo otvoriti.

Također, možemo isto primijetiti za otvaranje entiteta RADI između entiteta KNJIŽNICE i ZAPOSLENIKA jer nastaje isto iz veze tipa „više naprema više“. Drugim riječima, više zaposlenih u knjižnici radit će u više knjižnica. Taj novi entitet nazvat ćemo RADI. Ono što možemo primijetiti nad entitetom ZAPOSLNIK je to da moramo riješiti problem hijerarhije koji je određen na dvjema razinama, na razini nadređenog i podređenog. Postoje više načina rješavanja ovoga problema, ali najjednostavnije je dodati atribut svakom zapisu te na osnovu toga reći da li je podređen i kojem zaposlenim te samim time nemamo potrebe otvarati novi entitet.

Za drugu vrstu potrebe otvaranja entiteta možemo pogledati kako je nastao entitet INVENTAR. Što se tiče entiteta INVENTAR, možemo primijetiti da će sadržavati dva vanjska ključa entiteta KNJIGA, KNJIZNICA, ali ono što je jedinstveno je to da moramo riješiti problem da naslov knjige nije finalan broj knjiga, odnosno kako ćemo bilježiti svaku knjigu. Samim time moramo povezati da svaka knjiga pripada nekoj od knjižnica, tako da joj je nužno dodati novi primarni ključ koji će identificirati svaku knjigu posebno. Veza SADRZI prelazi u entitet, odnosno entitet promatranja koji ćemo zvati INVENTAR.

Slična situacija koju moramo izbliže pogledati je kako smo dobili entitet CLANARINA. Sukladno tome da veza PLACA ima funkcionalnost voditi potrebne informacije o tome koji član je platio članarinu te koji zaposleni je to izdao. Situacija je dosta slična s već prijašnjim otvaranjem novih entiteta, a riječ je o vezi „više naprema više“ te ćemo ju nazivati CLANARINA. Dodajemo novi ključ tablice koji ćemo zvati *clanarina_ID* da bi se omogućio unos zapisa istih kombinacija atrubuta *zaspoelnik_ID* i *clan_knjiznice_ID.*

Ostala nam je još jedna specifična situacija koja čini razliku između modela, a riječ je o entitetima PREVODITELJ i AUTOR. Kao što smo već spomenuli, moramo razlikovati autora knjige i prevoditelja. Problem koji sam primijetio tijekom izrade logičkog modela je taj da konceptualni model nije sasvim točno rješavao problem razlikovanja. Naime, to je i svrha konceptualnog modela, navesti na drugačija razmišljanja i moguća bolja rješenja. Stoga, da bi se AUTOR i PREVODITELJ razlikovali, potrebni ih je promatrati kao zasebne entitete.

Razlog tome je što će pisac imati zasebne atribute dok prevoditelj svoje, te je najjednostavnije rješenje promatrati ih kao zasebne entitete. Ovakvim modelom možemo predvidjeti iduću situaciju koja je vezana ako je neki pisac AUTOR, a ujedno i PREVODITELJ ili obratno. Takav problem neće biti velik jer ćemo pohraniti, recimo pisca, kao AUTORA i kao PREVODITELJA jer su entiteti različiti, odnosno nemaju iste atribute.

Imamo još jednu vezu „više naprema više“ koja će prijeći u zasebni entitet. Riječ je o vezi PREVODI. Entitet PREVODI ima ulogu da prikaže koji jezik PREVODITELJ prevodi na drugi.

Uzevši u obzir veze koje smo morali prebaciti u entitete, broj entiteta povećava se i u ovom modelu. Popis je sljedeći: AUTORSTVO_KNJIGE, AUTORSKO_PRAVO, AUTOR, PREVODITELJ, KNJIGA, ZANR, JEZIK, PREVODI, INVENTAR, IZDAVACKA_KUCA, DRZAVA, GRAD, KNJIZNICA, RADI, ZAPOSLENIK, RAZINA_OBRAZOVANJA, VRSTA_ZAPOSLENIKA, CLANARINA, POSUDBA, CLAN_KNJIZNICE, STATUS_POSUDBE i VRSTA_CLANRINE.

S obzirom da nam je kardinalnost poznata od prije, potrebno ju je samo prebaciti u određeni tip veze prema pravilnoj notaciji koju smo detaljnije definirali u prošlim poglavljima. Možemo izbliže pogledati jednu vezu kao primjer. Riječ je o vezi 1:M koja će spajati entitet PREVODITELJ i KNJIGA. Jasno je da jedan prevoditelj može prevesti više knjiga, dok s druge strane, jednu knjigu preveo je jedan prevoditelj. Ono što je karakteristično je to da moramo na strani prema entitetu KNJIGA staviti mogućnost da vrijednost bude NULL jer knjiga ne mora biti prevedena, odnosno knjižnica može posjedovati knjigu na izvornom jeziku na kojemu ju je AUTOR napisao.

Ono što nam je još preostalo je da raspišemo atribute svakoga entiteta. Atributi svakoga entiteta probao sam svesti samo na one primarne, odnosno one koji su nam najophodniji za prikaz podataka i funkcionalnosti baze podataka. Točniji prikaz atributa pokazat ćemo nešto kasnije, ali ono što je važno navesti je da svaka tablica ima atribute koji su primarni u bilo kojoj situaciji i da te atribute ne možemo zapostaviti, a da ne izgubimo praktičnost baze podataka. Riječ je o primarnim ključevima koji će nam pružiti unikatnost zapisa za tablice. Potrebno je spomenuti i vanjske ključeve koji su također neizostavni dio uz pomoć kojih točno znamo o kojoj vrsti veze se radi. Samim time, primarni ključevi označeni su kao PK, dok vanjski ključevi VK. Samim time, ako imamo neizostavne vrijednosti tablice, postoje ostala svojstva koja nam nisu značajna kada pričamo o funkcionalnosti, ali su nužna ako želimo ostvariti prvenstvenu bit tablica, a to je pohrana podataka. Primjena normalnih formi

Postupak modeliranja nije završio dok ne provedemo zadnji postupak, a to je normalizacija. Razlog provedbe normalizacije je da bi se provjerila validnost baze podataka i

da bi se sankcionirale moguće pogreške na modelu. Potrebno je tablicu provesti do treće normalne forme da bi imali validnu tablicu, tako da počinjemo s prvom normalnom formom (1NF).

1NF govori o atomnosti podataka, odnosno o tome da zapis u sjecištu reda i stupca mora biti nedjeljiv i da sadrži jednu vrijednost. O jednostavnosti ove forme govori i Manger. Ne postoji svaka relacija koja nije u 1NF te ova norma postoji da bi se ipak normalizirali modeli koji nisu jednostavno zapisani [12, str 55]. Ono što nam je činiti je da jednostavno obratimo pozornost na navedene atribute i provjerimo da li su oni atomni i da li se bez potrebno ponavljaju. U našem trenutnom modelu nemamo probleme s 1NF i možemo prijeći na drugu.

Uvjet da bi se provela druga normalna forma (2NF) je zadovoljiti prvu, što i jesmo. Zatim možemo definirati što je 2NF, Relacija je u 2NF ako je svaki njezin ne-primarni atribut potpuno funkcionalno ovisan o primarnom ključu. Drugim riječima, relacija je u 2NF ako u njoj nema parcijalnih ovisnosti atributa o primarnom ključu. [12, str. 58] Ako malo bolje promotrimo definiciju 2NF, možemo vidjeti da ovo ograničenje vrijedi samo za složene primarne ključeve entiteta jer norma zahtjeva da svaki ne-primarni atribut ovisi u potpunosti o primarnom ključu, pa zbog toga tablice s jednim primarnim ključem nema potrebe promatrati jer svaki ne-primarni atribut u potpunosti ovisi o svojem ključu. U našem modelu tablice koje sadrže složene primarne ključeve su sljedeće: AUTORSTVO_KNJIGE i entitet RADI. S obzirom da u tim entitetima ne postoji niti jedno svojstvo koje nije primarno, nemamo potrebe raditi provjeru jer sve vrijednosti unutar tih tablica samo su primarni ključevi.

Da bi se dobio validni model baze podataka, potrebno je još provesti treću normalnu formu (3NF) koja ima uvjet da su sve prethodne norme zadovoljene. 3NF nam govori o tranzitivnim zavisnostima. [12, str. 60] Drugim riječima, moramo provjeriti da li se u entitetu nalazi atribut koji je zavisan o nekom drugom atributu, a da taj drugi atribut nije primarni ključ. Na modelu knjižnice možemo primijetiti da ne postoji atribut koji je ovisan o nekom ne-primarnom atributu unutar tablice. Uzmemo li za primjer tablicu CLAN_KNJIZNICE, možemo vidjeti da je atribut IME_CLANA isključivo ovisan o primarnom ključu i o niti jednom drugom atributu, a isto vrijedi i za ostale atribute. Odnosno, ne postoji atribut kojeg definira neki drugi atribut isključujući primarni ključ.

7. Izrada finalnog modela za GK Ivanić-Grad

Kao što smo već spomenuli, finalna faza razvoja modela je fizički razvoj. U ovom dijelu rada moramo primijeniti ono što smo već definirali u poglavlju izrade fizičkog modela, odnosno finalni razvoj modela baze podatak. Točnije, dobit ćemo fizičku bazu podataka koja je sposobna za pohranu podataka. Da bi se realizirala, koristit ćemo se SQL jezikom unutar MySQL Workbench sustava kojega ćemo sljedeće proći. U ovoj cjelini također ćemo definirati relacijsku shemu, objasniti tablice, koja je njena svrha u shemi i na kraju napraviti kod koji će kreirati bazu podataka.

S obzirom da smo na zadnjoj razini razvoja modela, fizički model dobiva novu karakteristiku, a riječ je o popisu atributa i njihovim vrstama koje ćemo moći iščitati iz finalnog modela. Sukladno tome, svaka greška zahtijevat će veću vremensku jedinicu za popravak i rekonstruiranje.

7.1. MySQL Workbench radno okruženje

Da bi se mogao implementirati finalni model, moramo opisati alat u kojemu ćemo napraviti bazu podataka. Prije svega, MySQL Workbenchalat sadrži veliki broj funkcionalnosti i verzija te se neke od njih plaćaju i primjerene su za profesionalno korištenje. Sukladno tome, koristit ćemo MySQL Workbench sučelje koje se koristi u komercijalne, ali i osobne svrhe. Idući razlog je taj što je sučelje besplatno za korištenje te podržava manipulaciju velikih količina podataka.

Prema dokumentaciji programa, MySQL Workbench pokriva djelovanja u idućih pet područja[13]:

- SQL razvoj – stvaranje i upravljanje vezama s poslužiteljem baze podataka
- Modeliranje podataka - grafičko stvaranje modela baze podataka, uređivanje tablica, stupaca, indeksa, okidača, procedura, pogleda…
- Administracija poslužitelja performanse i sigurnost baze podataka, administracija instanci MySQL poslužitelja.
- Migracija podataka- migracija podataka, tablica s kompatibilnim sustavima poput Microsoft SQL Serverom, Sybase ASE, SQLite, SQL Anywhere, PostgreSQL
- Podrška za MySQL Enterpris

MySQL Workbench je objedinjeni vizualni alat za arhitekte baza podataka, programere i administraciju baze podatka. Također, MySQL Workbench pruža modeliranje

podataka, razvoj SQL-a i sveobuhvatne administrativne alate za konfiguraciju poslužitelja, korisničku administraciju, sigurnosno kopiranje itd. [13] Kao što možemo primijetiti, sučelje će biti savršeno za razvoj našeg fizičkog modela jer će nam omogućiti sve potrebne operacije manipuliranja podataka, administraciju baze podataka, izradu modela, korištenje servera i bit će dobar temelj tijekom izrade aplikacije. Jedna funkcionalnost koja će nam biti od velike koristi i ujedno funkcionalnost koja uvelike odvaja alat MySql Workbench od ostalih je laka izrada fizičkog modela koji ćemo generirati putem čarobnjaka za izradu modela i samim time olakšati izradu dokumentacije.

Da bi se uspješno otvorila relacijska shema, u našem slučaju shemu ćemo zvati *gradska_knjižnica*, potrebno je kreirati server pomoću „MySQL Server Installera“ gdje se definira port, vrsta protokola, profil te dodaje lozinka servera. Nakon što smo riješili administrante potrebe MySQL Workbench alata, možemo prijeći na izradu baze podataka, odnosno finalnog modela.

7.2. Relacijska shema baze podatka

Sljedeću stvar koju moramo riješiti je popis stupaca koji će krasiti svaku tablicu. Legenda koju ćemo koristiti je sljedeća: primarni ključevi bit će podcrtani, a vanjski ključevi bit će označeni kurzivom. Iako je popis tablica i njenih stupaca ispisan u relacijskoj shemi, dodatno ćemo svaku tablicu objasniti kroz opis tablica da bi svako mogao razumjeti na što se svaka tablica odnosi i koja je njena uloga u shemi.

AUTORSTVO_KNJIGE (*isbn*, *autor_ID*, *autorsko_pravo_ID*)

AUTORSKO_PRAVO(autorsko_pravo_ID, ime_autorsko_prava, opis_autorsko_prava)

AUTOR (autor_ID, ime_autora, prezime_autora, datum_rodenja, datum_smrt)

PREVODITELJ (prevoditelj_ID, ime_prevoditelja, prezime_prevoditelja, službeni_prevoditelj, razina_obrazovanja_ID)

PREVODI (*prevoditelj_ID*, *jezik_s_ID, jezik_na_ID*)

KNJIGA (isbn, naslov_knjige, godina_objave, broj_kopija, *prevoditelj_ID,zanr_ID*, *djelo_ID*, *jazik_ID, izdavacka_kuca_ID*)

ZANR (zanr_ID, ime_zanra)

JEZIK (jezik_ID, ime_jezika)

INVENTAR (inventarni_broj_ID, slobodno, *knjiznica_ID*, *isbn*)

IZDAVACKA_KUCA (izdavacka_kuca_ID, ime_izdavacke_kuce, adresa_izdavacke_kuce, *grad_ID*)

DRZAVA (drzava_ID, ime_drzave)

GRAD (grad_ID, ime_grada, postanski_broj_grada, *drzava_ID*)

KNJIZNICA (knjiznica_ID, ime_knjiznice, adresa_knjiznice, godina_osnivanja, *grad_ID*)

RADI (*zaposlenik_ID*, *knjiznica_ID*)

ZAPOSLENIK (zaposlenik_ID, ime_zaposlenika, prezime_zaposlenika, datum_rodenja, *razina_obrazovanja_ID*, *vrsta_zaposlenika_ID*, *nadreden_ID*)

CLANARINA (clanarina_ID, datum_placanja, datum_isteka, clanarina_vrijedi, *zaposlenik_ID*, *clan_knjiznice_ID*)

VRSTA_ZAPOSLENIKA (vrsta_zaposlenika_ID, ime_vrste_zaposlenika, opis_djelatnosti)

RAZINA_OBRAZOVANJA (razina_obrazovanja_ID, ime_razina_obrazovanja, kratica_razine_obrazovanja)

CLAN_KNJIZNICE (clan_knjiznice_ID, ime_clana, prezime_clana, adresa_clana, email_clana, telefonski_broj_clana*, grad_ID, vrsta_clanarine_ID*)

STATUS_POSUDBE (status_posudbe_ID, ime_statusa_posudbe)

POSUDBA (posudba_ID, datum_posudbe, datum_predvidenog_vracanja, datum_vracanja, novcani_zaostatak, *clan_knjiznice_ID*, *status_posudbe_ID*, *inventarni_broj_ID*)

VRSTA_CLANARINE (vrsta_clanarine_ID, ime_vrsta_clanarine, opis_vrste_clanarine, cijena_clanarine)

7.3. Dokumentacija tablica

Svaka tablica ima svoju ulogu u modelu tako da ćemo definirati svaku tablicu i njene atribute te samim time napraviti dokumentaciju i njenu ulogu u cjelokupnom modelu.

7.3.1. ZANR

Pomoću tablice ŽANR definiramo sve žanrove koje tablica KNJIGA može poprimiti, drugim riječima tablica sadrži popis ŽANROVA koje jedna knjiga može imati.

PK/FK	Atribut	Vrsta atributa	Opis
PK	zanr_ID	INT	Šifra zanra

	ime_zanra	VARCHAR(45)	Ime zanra

Tablica **2**: TablicaZANR. [samostalna izrada]

7.3.2. JEZIK

Slično tablici ZANR, tablica JEZIK će imati za ulogu ispisati sve jezike na kojima knjiga može biti napisana, ali i jezike na kojima PREVODITELJ prevodi knjige.

PK/FK	Atribut	Vrsta atributa	Opis
PK	jezik_ID	INT	Šifra jezika
	ime_jezika	VARCHAR(45)	Ime jezika

Tablica **3**: Tablica JEZIK. [samostalna izrada]

7.3.3. AUTORSKO_PRAVO

S obzirom da imamo više prava kojima autor polaže nad knjigom, dobar potez bio bi ih pohraniti u zasebnu tablicu. Tablica će sadržavati primarni ključ nad vrijednošću pod stupcem autorsko_pravo_ID. Postojat će autorsko pravo i njen opis u ostala dva stupca. Ova tablica primarno se koristi da bi mogli lakše definirati prava za tablicu AUTORSTVO_KNJIGE.

PK/FK	Atribut	Vrsta atributa	Opis
PK	autorsko_pravo_ID	INT	Šifra prava
	ime_autorskog_prava	VARCHAR(45)	Ime prava
	opis_autorskog_prava	VARCHAR(100)	Kratki opis prava

Tablica **4**: Tablica AUTORSKO_PRAVO. [samostalna izrada]

7.3.4. DRZAVA

Da bi pohranili podatke o gradovima, odnosno specifično u kojoj državi se nalazi grad, dobivamo tablicu u kojoj možemo sve to pohraniti a zvat ćemo ju DRZAVA. Tablica je jednostavna, a sastoji se od primarnog ključa drzava_ID, i imena države.

PK/FK	Atribut	Vrsta atributa	Opis
PK	drzava_ID	INT	Šifra države
	ime_drzave	VARCHAR(45)	Ime drzave

Tablica **5**: Tablica DRZAVA. [samostalna izrada]

7.3.5. RAZINA_OBRAZOVANJA

RAZINA OBRAZOVANJA jednostavna je tablica koja nema vanjskih ključeva, nego joj je cilj definirati titulu u tablici PREVODITELJ i ZAPOSLENIK. Sadrži primarni ključ, imena razina obrazovanja i radi lakšeg čitanja vrijednosti, kratice.

PK/FK	Atribut	Vrsta atributa	Opis
PK	razina_obrazoavnja_ID	INT	Šifra razine obrazovanja
	ime_razine_obrazovanja	VARCHAR(45)	Ime razine obrazovanja
	kratica_obrazovanja	VARCHAR (3)	Kratica obrazovanja

Tablica **6**: Tablica RAZINA_OBRAZOVANJA. [samostalna izrada]

7.3.6. VRSTA_ZAPOSLENIKA

Tablicom VRSTA_ZAPOSLENIKA definirat ćemo radna mjesta u knjižnici, tako da se za svakog zaposlenog može definirati njegova pozicija u knjižnici. Tablica sadrži slične stupce kao i tablica RAZINA_PBRAZOVANJA i također ju smatramo jednostavnom. Jedina razlika je što imamo opis za svaku radnu poziciju unutar knjižnice.

PK/FK	Atribut	Vrsta atributa	Opis
PK	vrsta_zaposlenika_ID	INT	Šifra vrste zaposlenika
	ime_vrsta_zaposlenika	VARCHAR(45)	Ime vrste zaposlenika
	opis_vrste_zaposlenik	VARCHAR (200)	Opis radnog mjesta

Tablica **7**: Tablica VRSTA_ZAPOSLENIKA. [samostalna izrada]

7.3.7. VRSTA_CLANARINE

Sukladno tome da postoje više vrsta članarina, dobra praksa je pohraniti takve vrijednosti u zasebnu tablicu. Upravo ova tablica sadržavat će popis mogućih članarina i bit će sklona budućim promjenama. Tablica se sastoji od primarnog ključa, imena članarine i njenog opisa te na kraju cijene članarine.

PK/FK	Atribut	Vrsta atributa	Opis
PK	vrsta_clanarine_ID	INT	Šifra članarine
	ime_vrsta_clanarine	VARCHAR(45)	Ime vrste članarine
	opis_vrste_clanarine	VARCHAR (100)	Opis vrste članarine
	cijena_clanarine	VARCHAR(10)	Cijena članarine

Tablica **8**: Tablica VRSTA_CLANARINE. [samostalna izrada]

7.3.8. STATUS_POSUDBE

Da ne bi morali voditi evidenciju svake posudbe, možemo definirati statuse posudbe koje ćemo pohraniti upravo u ovu tablicu. Tablica je jednostavno osmišljena te osim njenog primarnog ključa sadrži o kojom statusu je riječ.

PK/FK	Atribut	Vrsta atributa	Opis
PK	status_posdube_ID	INT	Šifra statusa posudbe
	ime_statusa_posudbe	VARCHAR(45)	Ime statusa posudbe

Tablica **9**: Tablica STATUS_POSUDBE. [samostalna izrada]

7.3.9. GRAD

Da ne bi morali voditi evidenciju svake posudbe, možemo definirati statuse posudbe koje ćemo pohraniti upravo u ovu tablicu. Tablica je jednostavno osmišljena te osim njenog primarnog ključa, sadrži o kojom statusu je riječ.

PK/FK	Atribut	Vrsta atributa	Opis
PK	grad_ID	INT	Šifra grada
	ime_grada	VARCHAR(45)	Ime grada

	postanski_broj_grada	VARCHAR (6)	Poštanski broj
FK	drzava_ID	INT	Šifra države

Tablica **10**: Tablica GRAD. [samostalna izrada]

7.3.10. AUTOR

Uz pomoć tablice AUTOR, pohranit ćemo sve relevantne informacije o autoru. Autor prije svega ima svoju identifikaciju oznaku koju ćemo označiti autor_ID. Nadalje, važne su nam neke osnovne informacije kao što su ime_autora, prezime_autora, datum_rođenja i datum_smrti. Naravno, stupac datum_smrti može poprimiti vrijednost NULL, ali ovaj podatak će nam biti važan za čitanje podataka, ako želimo znati godine stvaralaštva.

PK/FK	Atribut	Vrsta atributa	Opis
PK	autor_ID	INT	Šifra autora
	ime_autora	VARCHAR(45)	Ime prevoditelja
	prezime_autora	VARCHAR(45)	Prezime prevoditelja
	datum_rođenja	DATE	Datum rođenja autora
	datum_smrti	DATE	Datum smrti autora

Tablica **11**: Tablica AUTOR. [samostalna izrada]

7.3.11. AUTORSTVO_KNJIGE

Tablica AUTORSTVO_KNJIGE sastoji se od skupa primarnih ključeva koji su ujedno i vanjski ključevi prema tablicama AUTOR, AUTORSKO_PRAVO i KNJIGA. Tablica je složena prema definiciji primarnih ključeva i njena svrha je da označi koju knjigu je koji autor napisao te koja su prava na korištenje knjige.

PK/FK	Atribut	Vrsta atributa	Opis
PK/FK	Isbn	INT	Šifra knjige
PK/FK	autor_ID	INT	Šifra autora
PK/FK	autorsko_pravo_ID	INT	Šifra prava

Tablica**12**: Tablica AUTORSTVO_KNJIGE. [samostalna izrada]

7.3.12. PREVODITELJ

Tablica PREVODITELJ ima ulogu prikazati entitet prevoditelja s njegovim svim potrebnim atributima. Prvi stupac u tablici bit će primarni ključ tablica s kojim ćemo označiti svaki zapis, znači prevoditelj_ID. Stupci ime_prevoditelja i prezime_prevoditelja definirat će njegovo ime i prezime. Nadalje, stupac službeni_prevoditelj će biti boolean vrijednost koja će nam govoriti o tome da li je prevoditelj ovlašten, odnosno da li je ovlašten ili je riječ o neslužbenom prijevodu. Zadnji stupac u tablicu govorit će o razini obrazovanja prevoditelja.

PK/FK	Atribut	Vrsta atributa	Opis
PK	prevoditelj_ID	INT	Šifra prevoditelja
	ime_prevoditelja	VARCHAR(45)	Ime prevoditelja
	prezime_prevdoitelja	VARCHAR(45)	Prezime prevoditelja
	službeni_prevoditelj	TINYINT	Boolean da li je prevoditelj službeni
FK	razina_obrazovanja_ID	INT	Šifra razine obrazovanja

Tablica **13**: Tablica PREVODITELJ. [samostalna izrada]

7.3.13. PREVODI

Tablica PREVODI čini skup primarnih i vanjskih ključeva između tablica PREVODITELJ i JEZIK. Razlog otvaranja ove tablice je taj što će PREVODITELJ prevoditi barem na jednom jeziku, ako ne i na više. S druge strane, jezik ne mora nužno biti pohranjen u ovoj tablici što može naslutiti da nemamo adekvatnog prevoditelja za neki specifični jezik. Rješenje koje moramo napraviti je da prevoditelj prevodi s jednog jezika na drugi tako da ćemo koristiti isti ključa iz tablice JEZIK.

PKFVK	Atribut	Vrsta atributa	Opis
PK/FK	prevoditelj_ID	INT	Šifra prevoditelja
PK/FK	jezik_s_ID	INT	Šifra jezika
PK/FK	jezik_na_ID	INT	Šifra jezika

Tablica **14**: Tablica PREVODI. [samostalna izrada]

7.3.14. KNJIGA

Što se tiče tablice KNJIGA, tablica sadrži primari ključ kojim ćemo označavati svaku knjigu koju knjiga sadrži i nazivat ćemo ga *isbn*. Posredno tome, svaka knjiga ima svoje osnovne atribute kao što su *naslov_knjige* i *godina_objave*. Važan stupac u ovoj tablici je *broj_kopija* jer on definira koliko je specifične knjiga izdano, odnosno koliko postoji kopija da bi se kasnije mogao izračunati točan broj svake knjige. Nadalje, važni su stupci vanjskih ključeva. Prvi od njih je *prevoditelj_ID* kojim označujemo da li je knjigu preveo prevoditelj ili je pohranjena na izvornom jeziku autora. U situaciji da je knjiga prevedena, također ćemo pohraniti i na kojem jeziku, pomoću *jezik_ID* stupca. S obzirom da svaku knjigu možemo definirati putem žanra, važna nam je poveznica na tablicu ZANR putem stupca *zanr_ID.* Na kraju ćemo pohraniti koja od izdavačkih kuća je izdala knjigu.

PK/FK	Atribut	Vrsta atributa	Opis
PK	isbn	INT	Šifra knjige
	naslov_knjige	VARCHAR(45)	Naslov knjige
	godina_objave	DATE	Godina objave knjige
	broj_kopija	INT	Broj kopija knjige
FK	prevoditelj_ID	INT	Šifra prevoditelja
FK	zanr_ID	INT	Šifra žanra
FK	jezik_ID	INT	Šifra jezika
FK	izdavacka_kuca_ID	INT	Šifra izdavačke kuće

Tablica**15**: Tablica KNJIGA. [samostalna izrada]

7.3.15. IZDAVACKA_KUCA

Da bi znali koju knjigu je izdala izdavačka kuća, postoji potreba pohrane podataka vezanih uz izdavačku kuću. Prvi stupac kao i do sada označavat će primarni ključ koji glasi *izdavacka_kuca_ID.* Osim primarnog ključa, tablica će sadržavati informacije o imenu izdavačke kuće, njenoj adresi te u kojem gradu se nalazi pojedina izdavačka kuća što ćemo riješiti pomoću vanjskoga ključa, *grad_ID.*

PK/FK	Atribut	Vrsta atributa	Opis

PK	izdavacka_kuca_ID	INT	Šifra kuće
	ime_izdavacke_kuce	VARCHAR(45)	Ime kuće
	adresa_izdavacke_kuce	VARCHAR (45)	Adresa kuće
FK	grad_ID	INT	Šifra grada

Tablica**16**: Tablica IZDAVACKA_KUCA. [samostalna izrada]

7.3.16. KNJIZNICA

S obzirom da imamo više knjižnica, točnije dvije, potrebno je imati tablicu u kojoj ćemo pohraniti podatke vezane za te dvije. Naravno, imat ćemo primarni ključ koji će identificirati obadvije knjižnice. Sljedeći stupci bit će vezani za osobne informacije knjižnice, kao što je *ime_knjiznice, adresa_knjiznice i godina_osnivanja*. Također, tablica će sadržavati vanjski ključ *grad_ID* koji će referencirati gdje se knjižnica nalazi.

PK/FK	Atribut	Vrsta atributa	Opis
PK	knjiznica_ID	INT	Šifra knjižnice
	ime_knjižnice	VARCHAR(45)	Ime knjižnice
	adresa_knjižnice	VARCHAR (45)	Adresa knjižnice
FK	grad_ID	INT	Šifra grada

Tablica**17**: Tablica KNJIZNICA. [samostalna izrada]

7.3.17. INVENTAR

Tablicom INVENTAR označit ćemo svaku knjigu koja postoji u KNJIŽNICI. Ova tablica je primarno osmišljena da bi se vodila evidencija o svakoj knjizi. Dok smo u tablici KNJIGA pohranili sve potrebne informacije o knjizi, u ovoj tablici ćemo raspisati svaku kopiju, odnosno dodijelit ćemo ju jednoj od dviju postojećih knjižnica. Možemo reći da ćemo ovom tablicom voditi evidenciju o knjigama te nad ovom tablicom pohranjivati sve knjige koje su spremne za posudbu. Osim prvog stupca koji će označavati primarni ključ tablice, važan nam je stupac *slobodno* koji će sadržavati vrijednost TRUE ili FALSE. Ovaj atribut osmišljen je da bi mogli znati koja knjiga je posuđena ili za neke specifične situacije, recimo da je knjiga

oštećena i samim time nije spremna na posudbu. Od vanjskih ključeva tablice, važni će nam biti ključevi iz tablice knjige i knjižnice (*isbn i knjiznica_ID*),

PK/FK	Atribut	Vrsta atributa	Opis
PK	inventarni_broj_ID	INT	Šifra inventara
	slobodno	TINYINT	Boolean vrijednost da li je knjiga slobodna
FK	knjiznica_ID	INT	Šifra knjižnice
FK	isbn	INT	Šifra knjige

Tablica **18**: Tablica INVENTAR. [samostalna izrada]

7.3.18. ZAPOSLENIK

S obzirom da knjižnica ima nešto manje od desetak zaposlenih, pohranit ćemo njihove podatke u ovu tablicu. Zaposlenik će biti direktno povezan s plaćanjem članarine, odnosno tablicom CLANARINA i u kojoj knjižnici radi. Prvi stupac rezerviran je za primarni ključ, dok će ostali atribut predstavljati osobne informacije o svakom. Zanimljivi atributi su vanjski ključevi tablice ZAPOSLENIK, a to su *razina_obrazovanja_ID*, *vrsta_zaposlenika_ID* i *nadredni*. Pomoću *razine_obrazovanja_ID* ćemo definirati titulu zaposlenoga, zatim stupcem *vrsta_zaposlenika_ID* ćemo reći na kojoj radnoj poziciji zaposlenik radi. Zadnji vanjski ključ osmišljen je da bi mogao prikazati hijerarhiju između zaposlenih, odnosno koji zaposlenik je podređen nekom drugom, imat će vrijednost njegovog ključa.

PK/FK	Atribut	Vrsta atributa	Opis
PK	zaposlenik_ID	INT	Šifra zaposlenika
	ime_zaposlenika	VARCHAR(45)	Ime kuće
	prezime_zaposlenika	VARCHAR (45)	Prezime kuće
	datum_rodenja	DATE	Datum rođenja
FK	razina_obrazovanja_ID	INT	Šifra razina obrazovanja
FK	vrsta_zaposlenika_ID	INT	Šifra vrste zaposlenika
FK	nadreden_ID	INT	Šifra nadređenog zaposlenika

Tablica**19**: Tablica ZAPOSLENIK. [samostalna izrada]

7.3.19. RADI

Tablica radi nastaje kao skup ključeva, točnije primarnih i vanjskih ključeva koji referenciraju na tablice KNJIZNICA i ZAPOSLENIK. Svrha joj je pokazati u kojoj knjižnici određeni zaposlenik radi.

PK/FK	Atribut	Vrsta atributa	Opis
PK/FK	zaposlenik_ID	INT	Šifra zaposlenika
PK/FK	knjiznica_ID	INT	Šifra knjižnice

Tablica **20**: Tablica RADI. [samostalna izrada]

7.3.20. CLAN_KNJIZNICE

Unutar ove tablice vodit ćemo sve potrebne podatke o svakom članu knjižnice. Tablica će sadržavati primarni ključ koji će biti oznaka identifikacije za svakoga člana. Osim primarnih ključeva, tablica će sadržavati i dva vanjska, *grad_ID*, *vrsta_clanarine_ID*. Pomoću tih vanjskih ključeva definirat ćemo u kojem gradu se član nalazi te kakvu vrstu članarine ima. Ostaju nam ne ključni atributi kojima će se definirati osobne informacije i kontakt člana.

PK/FK	Atribut	Vrsta atributa	Opis
PK	clan_knjiznice_ID	INT	Šifra člana
	ime_clana	VARCHAR(45)	Ime člana
	prezime_clana	VARCHAR (45)	Prezime člana
	adresa_clana	VARCHAR(45)	Adresa člana
	email_clana	VARCHAR(45)	Email člana
	telefonski_broj_clana	VARCHAR(45)	Telefonski broj člana
FK	grad_ID	INT	Šifra grada
FK	vrsta_clanarine_ID	INT	Šifra vrste članarine

Tablica **21**: Tablica CLAN_KNJIZNICE. [samostalna izrada]

7.3.21. POSUDBA

Tablica POSUDBA bilježit će sve posudbe koje su se dogodile i koje će se dogoditi. Samim time, svaka posudba imat će svoju identifikaciju, *posudena_knjiga_ID*. Stupac *datum_posudbe* davat će podatake o tome kada je knjigu zadužio član knjižnice, a sukladno tome bilježit ćemo dogovoreni datum vračanja u stupcu *datum_predvidenog_vracanja*. Moramo uzeti u obzir kada je član knjižnice zapravo vratio knjigu, taj podatak ćemo pohraniti u stupac *datum_vracanja*. Na osnovu toga da je *datum_vracanja* prešao datum koji je definiran u *datum_predvidenog_vracanja,* potrebno je pohraniti novčani zaostatak prema pravilu da je dan zakasnine 0,5 kuna. Da bi tablica bila funkcionalna, moramo dodati primarni ključ iz tablice CLAN_KNJIZNICE koji će govoriti koji član je zadužio knjigu. Sukladno tome, moramo znati o kojoj knjizi se radi pa nam je nužan ključ iz tablice INVENTAR. Kada član knjižnice posudi knjigu, potrebno je označiti da je knjiga posuđena, odnosno kada član knjižnice vrati knjigu, također je potrebno pohraniti promjene nad stupcem *status_posudbe_id* koji će referencirati na tablicu STATUS_POSUDBE.

PK/FK	Atribut	Vrsta atributa	Opis
PK	posudena_knjiga_ID	INT	Šifra posudbe
	datum_posudbe	DATE	Datum kad je knjiga posuđena
	datum_predvidenog_vracanja	DATE	Datum do kojega član mora vratiti knjigu
	datum_vračanja	DATE	Datum kad je član vratio knjigu
	novcani_zaostatak	DOUBLE	Novčani zaostatak ako postoji
FK	status_posudbe_ID	INT	Šifra statusa posudbe
FK	inventarni_broj_ID	INT	Šifra inventarnog broja knjige
FK	clan_knjiznice_ID	INT	Šifra člana knjižnice

Tablica **22**: Tablica POSUDBA. [samostalna izrada]

7.3.22. CLANARINA

Da bih implementirali funkcionalnost plaćanja u knjižnici, koristit ćemo se ovom tablicom, CLANARINA. Tablica će imati jedan ključ da bi se moglo identificirati plaćanje članarine. Tablica sadrži stupac *datum_placanja* da bi se unio datum na koji je član knjižnice platio članarinu. Sukladno tome, članarina ima definirani *datum_isteka* koji naglašava do

kada vrijedi članarina. Na pitanje da li je članarina aktivna, odgovorit će nam boolean vrijednost u stupcu *clanarina_vrijedi*. Tablica sadrži dva vanjska ključa, *zaposlenik_ID* odnosno osoba koja je naplatila članarinu i vrijednost i kojem članu je riječ, što ćemo pohraniti u stupac *clan_knjiznice_ID.*

PK/FK	Atribut	Vrsta atributa	Opis
PK	clanarina_ID	INT	Šifra članarine
	datum_plačanja	DATE	Datum kad je članarina plaćena
	datum_isteka	DATE	Datum do kojega članarina vrijedi
	clanarina_vrijedi	TINYINT	Boolean vrijednost da li je članarina aktivna
FK	zaposlenik_ID	INT	Šifra zaposlenika koji je naplatio
FK	clan_knjiznice_ID	INT	Šifra člana knjižnice

Tablica **23**: Tablica CLANARINA. [samostalna izrada]

7.4. Kod kreiranja baze podataka

Prethodno su navedene tablice i kroz njene atribute kreiran je i fizički model. Za to je potrebno koristiti SQL naredbe, a kod ćemo generirati kroz alaz MySQL Workbencha.

```
CREATE DATABASE  IF NOT EXISTS `gradska_knjiznica` /*!40100 DEFAULT
CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT
ENCRYPTION='N' */;
USE `gradska_knjiznica`;
MySQL dump 10.13  Distrib 8.0.28, for Win64 (x86_64)
Host: 127.0.0.1    Database: gradska_knjiznica
Server version    8.0.28
CREATE TABLE `autor` (
`autor_ID` int NOT NULL,
`ime_autora` varchar(45) NOT NULL,
`prezime_autora` varchar(45) NOT NULL,
`datum_rodenja` date NOT NULL,
`datum_smrti` date DEFAULT NULL,
      PRIMARY KEY (`autor_ID`),
      UNIQUE KEY `autor_ID_UNIQUE` (`autor_ID`))
CREATE TABLE `autorsko_pravo` (
```

```
	`autorosko_pravo_ID` int NOT NULL AUTO_INCREMENT,
	`ime_autorskog_prava` varchar(45) NOT NULL,
	`opis_autoroskog_prava` varchar(100) NOT NULL,
	PRIMARY KEY (`autorosko_pravo_ID`),
	UNIQUE KEY `ime_autorskog_prava_UNIQUE` (`ime_autorskog_prava`),
	UNIQUE KEY `autorosko_pravo_ID_UNIQUE` (`autorosko_pravo_ID`))
CREATE TABLE `autorstvo_knjige` (
`isbn` int NOT NULL,
`autor_ID` int NOT NULL,
	`autorsko_pravo_ID` int NOT NULL,
	PRIMARY KEY (`isbn`,`autor_ID`,`autorsko_pravo_ID`),
	KEY `autorstvo_knjige_autor_ID_FK2_idx` (`autor_ID`),
	KEY `autorstvo_knjige_autorsko_pravo_FK3_idx`
(`autorsko_pravo_ID`),
CONSTRAINT `autorstvo_knjige_autor_ID_FK2` FOREIGN KEY (`autor_ID`)
REFERENCES `autor` (`autor_ID`),
CONSTRAINT`autorstvo_knjige_autorsko_pravo_FK3` FOREIGN KEY
(`autorsko_pravo_ID`)REFERENCES `autorsko_pravo`
(`autorosko_pravo_ID`),
CONSTRAINT `autorstvo_knjige_isbn_FK1` FOREIGN KEY (`isbn`) REFERENCES
`knjiga` (`isbn`))
CREATE TABLE `clan_knjiznice` (
	clan_knjiznice_ID` int NOT NULL,
	`ime_clana` varchar(45) NOT NULL,
	`prezime_clana` varchar(45) NOT NULL,
	`adresa_clana` varchar(45) NOT NULL,
	`emaiLclana` varchar(45) NOT NULL,
	`telefosnki_broj_clana` varchar(45) NOT NULL,
	`grad_ID` int NOT NULL,
	`vrsta_clanarine_ID` int NOT NULL,
	PRIMARY KEY (`clan_knjiznice_ID`),
	UNIQUE KEY `clan_knjiznice_ID_UNIQUE` (`clan_knjiznice_ID`),
	KEY `grad_ID_idx` (`grad_ID`),
	KEY `vrsta_clanarine_ID_FK2_idx` (`vrsta_clanarine_ID`),
```

```
CONSTRAINT `clan_knjiznice_grad_ID_FK1` FOREIGN KEY (`grad_ID`)
REFERENCES `grad` (`grad_ID`) ON UPDATE CASCADE,

CONSTRAINT `clan_knjiznice_vrsta_clanarine_ID_FK2` FOREIGN KEY
(`vrsta_clanarine_ID`) REFERENCES `vrsta_clanarine`
(`vrsta_clanarine_ID`) ON UPDATE CASCADE)

CREATE TABLE `clanarina` (

      `clanarina_ID` int NOT NULL,

      `datum_placanja` date NOT NULL,

      `datum_isteka` date DEFAULT NULL,

      `clanarina_vrijedi` tinyint DEFAULT NULL,

      `zaposlenik_ID` int NOT NULL,

      `clan_knjiznice_ID` int NOT NULL,

      PRIMARY KEY (`clanarina_ID`),

      UNIQUE KEY `racun_ID_UNIQUE` (`clanarina_ID`),

      KEY `racun.zaposlenik_ID_FK1_idx` (`zaposlenik_ID`),

      KEY `racun.clan_knjiznice_ID_FK2_idx` (`clan_knjiznice_ID`),

CONSTRAINT `racun.clan_knjiznice_ID_FK2` FOREIGN KEY
(`clan_knjiznice_ID`) REFERENCES `clan_knjiznice`
(`clan_knjiznice_ID`),

CONSTRAINT `racun.zaposlenik_ID_FK1` FOREIGN KEY (`zaposlenik_ID`)
REFERENCES `zaposlenik` (`zaposlenik_ID`))

CREATE TABLE `drzava` (

      `drzava_ID` int NOT NULL,

      `ime_drzave` varchar(45) NOT NULL,

      PRIMARY KEY (`drzava_ID`),

      UNIQUE KEY `drzava_ID_UNIQUE` (`drzava_ID`),

      UNIQUE KEY `ime_drzave_UNIQUE` (`ime_drzave`))

CREATE TABLE `grad` (

      `grad_ID` int NOT NULL,

      `ime_grada` varchar(45) NOT NULL,

      `postanski_broj_grada` varchar(6) NOT NULL,

      `drzava_ID` int NOT NULL,

      PRIMARY KEY (`grad_ID`),

      UNIQUE KEY `grad_ID_UNIQUE` (`grad_ID`),
```

```
	UNIQUE KEY `ime_grada_UNIQUE` (`ime_grada`),
	UNIQUE KEY `postanski_broj_grada_UNIQUE` (`postanski_broj_grada`),
	KEY `drzava_ID_idx` (`drzava_ID`),
CONSTRAINT `grad_drzava_ID_FK1` FOREIGN KEY (`drzava_ID`) REFERENCES
`drzava` (`drzava_ID`) ON UPDATE CASCADE)
CREATE TABLE `inventar` (
	`inventarni_broj_ID` int NOT NULL AUTO_INCREMENT,
	`slobodno` tinyint NOT NULL DEFAULT '1',
	`knjiznica_ID` int NOT NULL DEFAULT '1',
	`isbn` int NOT NULL,
	PRIMARY KEY (`inventarni_broj_ID`),
	UNIQUE KEY `mjesto_knjige_ID_UNIQUE` (`inventarni_broj_ID`),
	KEY `mjesto_knjige_knjiznica_ID_FK1_idx` (`knjiznica_ID`),
	KEY `mjesto_knjige_knjiga_ID_FK2_idx` (`isbn`),
	CONSTRAINT `inventar_isbn_FK2` FOREIGN KEY (`isbn`) REFERENCES
`knjiga` (`isbn`),
CONSTRAINT `inventar_knjiznica_ID_FK1` FOREIGN KEY (`knjiznica_ID`)
REFERENCES `knjiznica` (`knjiznica_ID`))
CREATE TABLE `izdavacka_kuca` (
	`izdavacka_kuca_ID` int NOT NULL,
	`ime_izdavacke_kuce` varchar(45) NOT NULL,
	`adresa_izdavacke_kuce` varchar(45) NOT NULL,
	`grad_ID` int NOT NULL,
	PRIMARY KEY (`izdavacka_kuca_ID`),
	UNIQUE KEY `izdavacka_kuca_ID_UNIQUE` (`izdavacka_kuca_ID`),
	UNIQUE KEY `ime_izdavacke_kuce_UNIQUE` (`ime_izdavacke_kuce`),
	KEY `izdavacka_kuca_grad_ID_FK1_idx` (`grad_ID`),
CONSTRAINT `izdavacka_kuca_grad_ID_FK1` FOREIGN KEY (`grad_ID`)
REFERENCES `grad` (`grad_ID`) ON UPDATE CASCADE)
CREATE TABLE `jezik` (
	`jezik_ID` int NOT NULL AUTO_INCREMENT,
	`ime_jezika` varchar(45) NOT NULL,
	PRIMARY KEY (`jezik_ID`),
```

```
      UNIQUE KEY `ime_jezika_UNIQUE` (`ime_jezika`),
      UNIQUE KEY `jezik_ID_UNIQUE` (`jezik_ID`))
CREATE TABLE `knjiga` (
      `isbn` int NOT NULL,
      `knjiga_ID` int NOT NULL,
      `naslov_knjige` varchar(60) NOT NULL,
      `godina_objave` varchar(10) NOT NULL,
      `broj_kopija` int NOT NULL,
      `prevoditelj_ID` int DEFAULT NULL,
      `zanr_ID` int NOT NULL,
      `jezik_ID` int NOT NULL,
      `izdavacke_kuca_ID` int NOT NULL,
      PRIMARY KEY (`isbn`,`knjiga_ID`),
      UNIQUE KEY `isbn_UNIQUE` (`isbn`),
      UNIQUE KEY `naslov_knjige_UNIQUE` (`naslov_knjige`),
      UNIQUE KEY `knjiga_ID_UNIQUE` (`knjiga_ID`),
      KEY `knjiga_zanr_ID_FK1_idx` (`zanr_ID`),
      KEY `knjiga_jezik_ID_FK3_idx` (`jezik_ID`),
      KEY `knjiga_OIB_izdavacke_kuce_FK4_idx` (`izdavacke_kuca_ID`),
      KEY `knjiga_prevoditelj_ID_FK4_idx` (`prevoditelj_ID`),
      CONSTRAINT `knjiga_jezik_ID_FK3` FOREIGN KEY (`jezik_ID`)
REFERENCES `jezik` (   `jezik_ID`) ON UPDATE CASCADE,
CONSTRAINT `knjiga_OIB_izdavacke_kuce_FK4` FOREIGN KEY
(`izdavacke_kuca_ID`) REFERENCES `izdavacka_kuca` (`izdavacka_kuca_ID`)
ON UPDATE CASCADE,
CONSTRAINT `knjiga_prevoditelj_ID_FK4` FOREIGN KEY (`prevoditelj_ID`)
REFERENCES `prevoditelj` (`prevoditelj_ID`),
CONSTRAINT `knjiga_zanr_ID_FK1` FOREIGN KEY (`zanr_ID`) REFERENCES
`zanr` (`zanr_ID`) ON UPDATE CASCADE)
CREATE TABLE `knjiznica` (
      `knjiznica_ID` int NOT NULL,
      `ime_knjiznice` varchar(45) NOT NULL,
      `adresa_knjiznice` varchar(45) NOT NULL,
      `godina_osnivanja` varchar(10) NOT NULL,
```

```
	`grad_ID` int NOT NULL,
	PRIMARY KEY (`knjiznica_ID`),
	UNIQUE KEY `knjiznica_ID_UNIQUE` (`knjiznica_ID`),
	UNIQUE KEY `ime_knjiznice_UNIQUE` (`ime_knjiznice`),
	KEY `grad_ID_FK1_idx` (`grad_ID`),
CONSTRAINT `knjiznica_grad_ID_FK1` FOREIGN KEY (`grad_ID`) REFERENCES
`grad` (`grad_ID`) ON UPDATE CASCADE)
CREATE TABLE `posudba` (
	`posudena_knjiga_ID` int NOT NULL,
	`datum_posudbe` date NOT NULL,
	`datum_predvidenog_vracanja` date DEFAULT NULL,
	`datum_vracanja` varchar(45) DEFAULT NULL,
	`novcani_zaostatak` double DEFAULT '0',
	`clan_knjiznice_ID` int NOT NULL,
	`status_posudbe_ID` int NOT NULL DEFAULT '1',
	`inventarni_broj_ID` int NOT NULL,
	PRIMARY KEY (`posudena_knjiga_ID`),
	UNIQUE KEY `posudena_knjiga_ID_UNIQUE` (`posudena_knjiga_ID`),
	KEY `posudena_knjiga_clan_knjiznice_ID_FK1_idx`
(`clan_knjiznice_ID`),
	KEY `posudena_knjiga_status_posudbe_ID_FK2_idx`
(`status_posudbe_ID`),
	KEY `posudena_knjiga_mjesto_knjige_ID_FK3_idx`
(`inventarni_broj_ID`),
CONSTRAINT `posudena_knjiga_clan_knjiznice_ID_FK1` FOREIGN KEY
(`clan_knjiznice_ID`)   REFERENCES `clan_knjiznice`
(`clan_knjiznice_ID`) ON UPDATE CASCADE,
CONSTRAINT `posudena_knjiga_mjesto_knjige_ID_FK3` FOREIGN KEY
(`inventarni_broj_ID`) REFERENCES `inventar` (`inventarni_broj_ID`),
CONSTRAINT `posudena_knjiga_status_posudbe_ID_FK2` FOREIGN KEY
(`status_posudbe_ID`) REFERENCES `status_posudbe` (`status_posudbe_ID`)
ON UPDATE CASCADE)
CREATE TABLE `prevodi` (
	`prevoditelj_ID` int NOT NULL,
	`jezik_s_ID` int NOT NULL,
```

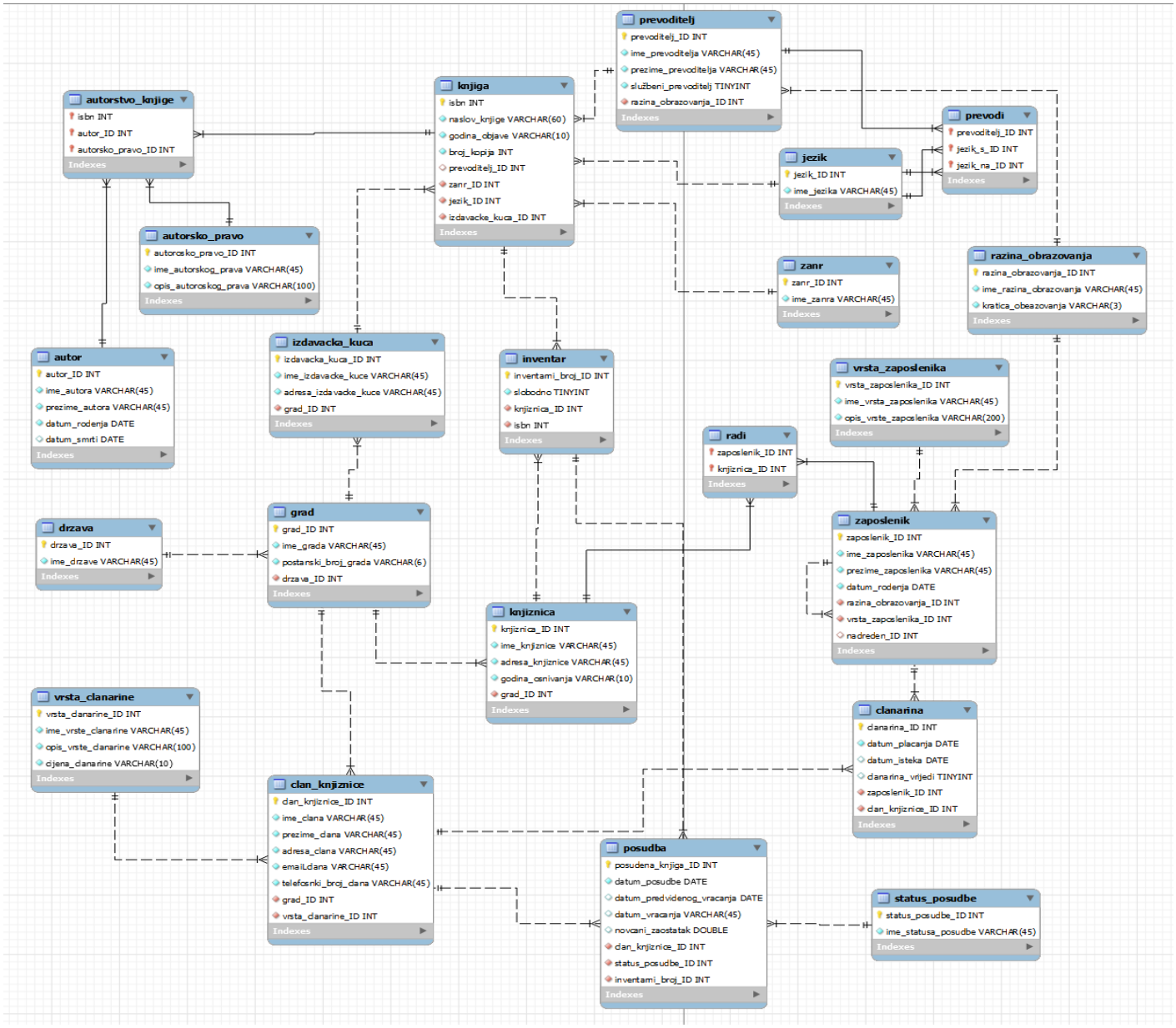
```
	`jezik_na_ID` int NOT NULL,

	PRIMARY KEY (`prevoditelj_ID`,`jezik_s_ID`,`jezik_na_ID`),

	KEY `prevodi_jezik_ID_FK2_idx` (`jezik_s_ID`),

	KEY `prevodit_jezik_na_ID_FK3_idx` (`jezik_na_ID`),

CONSTRAINT `prevodi_jezik_s_ID_FK2` FOREIGN KEY (`jezik_s_ID`)
REFERENCES `jezik` (`jezik_ID`),

CONSTRAINT `prevodi_prevoditelj_ID_FK1` FOREIGN KEY (`prevoditelj_ID`)
REFERENCES `prevoditelj` (`prevoditelj_ID`),

CONSTRAINT `prevodit_jezik_na_ID_FK3` FOREIGN KEY (`jezik_na_ID`)
REFERENCES `jezik` (`jezik_ID`))

CREATE TABLE `prevoditelj` (

	`prevoditelj_ID` int NOT NULL,

	`ime_prevoditelja` varchar(45) NOT NULL,

	`prezime_prevoditelja` varchar(45) NOT NULL,

	`službeni_prevoditelj` tinyint NOT NULL,

	`razina_obrazovanja_ID` int NOT NULL,

	PRIMARY KEY (`prevoditelj_ID`),

	KEY `prevoditelj_razina_obrazovanja_ID_FK1_idx`
(`razina_obrazovanja_ID`),

CONSTRAINT `prevoditelj_razina_obrazovanja_ID_FK1` FOREIGN KEY
(`razina_obrazovanja_ID`) REFERENCES `razina_obrazovanja`
(`razina_obrazovanja_ID`))

CREATE TABLE `radi` (

	`zaposlenik_ID` int NOT NULL,

	`knjiznica_ID` int NOT NULL,

	PRIMARY KEY (`zaposlenik_ID`,`knjiznica_ID`),

	UNIQUE KEY `zaposlenik_ID_UNIQUE` (`zaposlenik_ID`),

	KEY `radi_knjiznica_ID_FK2_idx` (`knjiznica_ID`),

CONSTRAINT `radi_knjiznica_ID_FK2` FOREIGN KEY (`knjiznica_ID`)
REFERENCES `knjiznica` (`knjiznica_ID`) ON UPDATE CASCADE,

CONSTRAINT `radi_zaposlenik_ID_FK1` FOREIGN KEY (`zaposlenik_ID`)
REFERENCES `zaposlenik` (`zaposlenik_ID`) ON UPDATE CASCADE)

CREATE TABLE `razina_obrazovanja` (

	`razina_obrazovanja_ID` int NOT NULL,

	`ime_razina_obrazovanja` varchar(45) NOT NULL,
```

```
	`kratica_obeazovanja` varchar(3) NOT NULL,
	PRIMARY KEY (`razina_obrazovanja_ID`),
	UNIQUE KEY `razina_obrazovanja_ID_UNIQUE`
(`razina_obrazovanja_ID`),
	UNIQUE KEY `ime_razina_obrazovanja_UNIQUE`
(`ime_razina_obrazovanja`))
CREATE TABLE `status_posudbe` (
	`status_posudbe_ID` int NOT NULL,
	`ime_statusa_posudbe` varchar(45) NOT NULL,
	PRIMARY KEY (`status_posudbe_ID`),
	UNIQUE KEY `status_posudbe_ID_UNIQUE` (`status_posudbe_ID`),
	UNIQUE KEY `ime_statusa_posudbe_UNIQUE` (`ime_statusa_posudbe`))
CREATE TABLE `vrsta_clanarine` (
	`vrsta_clanarine_ID` int NOT NULL,
	`ime_vrste_clanarine` varchar(45) NOT NULL,
	`opis_vrste_clanarine` varchar(100) NOT NULL,
	`cijena_clanarine` varchar(10) NOT NULL,
	PRIMARY KEY (`vrsta_clanarine_ID`),
	UNIQUE KEY `vrsta_clanarine_ID_UNIQUE` (`vrsta_clanarine_ID`),
	UNIQUE KEY `ime_vrste_clanarine_UNIQUE` (`ime_vrste_clanarine`))
CREATE TABLE `vrsta_zaposlenika` (
	`vrsta_zaposlenika_ID` int NOT NULL,
	`ime_vrsta_zaposlenika` varchar(45) NOT NULL,
	`opis_vrste_zaposlenika` varchar(200) NOT NULL,
	PRIMARY KEY (`vrsta_zaposlenika_ID`),
	UNIQUE KEY `vrsta_zaposlenika_ID_UNIQUE` (`vrsta_zaposlenika_ID`),
	UNIQUE KEY `ime_vrsta_zaposlenika_UNIQUE`
(`ime_vrsta_zaposlenika`))
CREATE TABLE `zanr` (
	`zanr_ID` int NOT NULL AUTO_INCREMENT,
	`ime_zanra` varchar(45) NOT NULL,
	PRIMARY KEY (`zanr_ID`),
	UNIQUE KEY `zanr_ID_UNIQUE` (`zanr_ID`),
```

```
      UNIQUE KEY `ime_zanra_UNIQUE` (`ime_zanra`))
CREATE TABLE `zaposlenik` (
      `zaposlenik_ID` int NOT NULL,
      `ime_zaposlenika` varchar(45) NOT NULL,
      `prezime_zaposlenika` varchar(45) NOT NULL,
      `datum_rodenja` date NOT NULL,
      `razina_obrazovanja_ID` int NOT NULL,
      `vrsta_zaposlenika_ID` int NOT NULL,
      `nadreden_ID` int DEFAULT NULL,
      PRIMARY KEY (`zaposlenik_ID`),
      UNIQUE KEY `zaposlenik_ID_UNIQUE` (`zaposlenik_ID`),
      KEY `zaposlenik_vrsta_zaposlenika_ID_FK1_idx`
(`vrsta_zaposlenika_ID`),
      KEY `zaposlenik_nadreden_ID_FK2_idx` (`nadreden_ID`),
      KEY `zaposlenik_razina_opbrazovanja_ID_FK3_idx`
(`razina_obrazovanja_ID`),
CONSTRAINT `zaposlenik_nadreden_ID_FK2` FOREIGN KEY (`nadreden_ID`)
REFERENCES `zaposlenik` (`zaposlenik_ID`) ON DELETE SET NULL ON UPDATE
CASCADE,
CONSTRAINT `zaposlenik_razina_opbrazovanja_ID_FK3` FOREIGN KEY
(`razina_obrazovanja_ID`) REFERENCES `razina_obrazovanja`
(`razina_obrazovanja_ID`),
CONSTRAINT `zaposlenik_vrsta_zaposlenika_ID_FK1` FOREIGN KEY
(`vrsta_zaposlenika_ID`) REFERENCES `vrsta_zaposlenika`
(`vrsta_zaposlenika_ID`) ON UPDATE CASCADE
```

7.5. Finalni model Gradske knjižnice

Uspješnim izvršenjem navedenih SQL naredbi prema relacijskoj shemi dobivamo finalni model.



Slika 7: Fizički model Gradske knjižnice u Ivanić-Gradu. [samostalna izrada]

8. Unos podataka

Prije svega da bi se unosili podatci, koristi se posebna naredba za unos podataka koja glasi „INSERT INTO“. Ova naredba definira nad kojom tablicom će se dogoditi operacija umetanja podataka te klauzulom „VALUES“ definiraju se vrijednosti. Za primjer unosa, pogledat ćemo unos nad tablicama KNJIGA i POSUDBA, naravno uz uvjet da su poštivane smjernice za unos podataka koje ćemo definirati.

8.1. Unos podataka za tablicu KNJIGA

Za primjer unosa, pogledat ćemo unos nad tablicom KNJIGA. Tablica KNJIGA poprima čak četiri vanjska ključa tablica čije vrijednosti prethodno moraju biti unesene.

```
INSERTINTO `gradska_knjiznica`.`knjiga`
(`isbn`, `naslov_knjige`, `godina_objave`, `broj_kopija`, `prevoditelj_ID`,
`zanr_ID`, `jezik_ID`, `izdavacke_kuca_ID`)
VALUES ('65346', 'Crni mačak', '1843', '3', '23', '8', '1', '4');
```

Svakom naslovu knjige dodijeljen je *isbn* da bi se identificirao svaki naslov knjige. Zatim pohranjujemo osnovne informacije o knjizi te na kraju unosimo vrijednosti vanjskih ključeva. Od vanjskih ključeva koje tablica KNJIGA prima obavezni za unos su stupci *zanr_ID,jezik_ID*, i *izdavacka_kuca_ID.* S druge strane, vrijednosti ključa koji se referencira na tablicu PREVODITELJ nije nužan, drugim riječima knjiga ne mora imati prevoditelja.

8.2. Unos podataka za tablicu POSUDBA

Drugi primjer unosa podataka ujedno predstavlja i zadnju vrijednost koju unosimo. Tablica POSUDBA sadrži identifikator, odnosno *posudba_ID*, odnosno da bi se mogle unositi iste posudbe, odnosno posudbe u kojima je ista knjiga dodijeljena istom članu. Iz tog razloga specificiramo novi ključ. Pored ključa, u tablicu se pohranjuju datumi, odnosno kad je knjiga posuđena i do kada se očekuje da korisnik vrati knjigu i datum kada je član zapravo vratio knjigu. Vanjski ključevi označuju kojem članu knjižnice je dodijeljena knjiga te se vodi status posudba.

```
INSERT INTO `gradska_knjiznica`.`posudba` (`datum_posudbe`,
`clan_knjiznice_ID`, `inventarni_broj_ID`) VALUES ('2022-06-15', '8',
'31');
```

Kao što možemo primijetiti, od korisnika se ne očekuje unos svih vrijednosti tablice, nego je unos omogućen okidačima koje ćemo posvetiti više pažnje u idućem poglavalja.

9. Okidači

Važna karakteristika svake baze podataka su njeni okidači. Pomoću okidača ćemo definirati izravno okidanje tijekom promjene stanja nad tablicom. U našem primjeru, koristit ćemo se okidačima da bi se stvorila logiku baze podataka tijekom unosa podataka u specifičnu tablicu. Provjeravat ćemo pravilnost unesenih podataka, odnosno da li oni zadovoljavaju specifične uvjete. Nakon unosa, promijenit ćemo stanje zahvaćenih tablica i na kraju omogućiti jednostavniji unos podataka korisniku.

Da bi se uspješno kreirao okidač nad tablicom, moramo poznavati sintaksu kreiranja okidača. Najprije se definira ključna riječ „CREATE TRIGGER“ te slijedi vremensko ograničenje (BEFORE, AFTER), odnosno kada će se okidač okinuti. Osnovna razlika vremenske specifikacije je ta da BEFORE omogućuje da okidač ne vidi promjene koje se trebaju izvršiti, dok AFTER vidi promjene. U slučaju da postoji više okidača za isti događaj, tada se oni izvršavaju abecednim redom [18]. Slijedi definiranje aktivnosti koje može biti UPDATE, INSERT i DELETE. Nakon definiranja „glave“ okidača, važno je naglasiti početak okidača s BAGIN i završetak s END klauzulom.

S obzirom da u našem slučaju izrade baze podataka, radimo administrativnu podršku sustavu. Koristit ćemo INSERT i UPDATE stanje, odnosno da bih pokrili situacije u kojima korisnik baze podataka mijenja stanje već nekog unesenog zapisa, okidač UPDATE bit će gotovo isti okidaču INSERT pa ćemo se iz toga razloga fokusirati samo na jednom.

9.1. Okidač računanja zaostatka i statusa posudbe

Prvi primjer okidača bit će nad tablicom POSUDBA. Okidač ima ulogu olakšati unos podataka nad tablicom i izračunati eventualni zaostatak za posudbu. Samim time, unos podatka za stupac *predvidenog_vracanja* nije nužno ispuniti jer će okidač samostalno računa datum vraćanja.

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`racunanje_podataka_BEFORE_INSERT` BEFORE INSERT ON `posudba` FOR EACH ROW
BEGIN

DECLARE zaostatak DOUBLE DEFAULT 0;

SET new.datum_predvidenog_vracanja = DATE_ADD(new.datum_posudbe, INTERVAL 2
MONTH);

SET zaostatak =DATEDIFF(new.datum_vracanja,
new.datum_predvidenog_vracanja);

      IF zaostatak > 0 THEN
```

```
        SET new.novcani_zaostatak = zaostatak * 0.5;
    ELSE
        SET new.novcani_zaostatak =0; END IF;
    IF new.datum_vracanja IS NULL THEN
        SET new.status_posudbe_ID=1;
    ELSE
        SET new.status_posudbe_ID=2; END IF;
    END
```

9.2. Okidač ažuriranja tablice INVENTAR

Idući primjer je također vezan uz tablicu POSUDBA, a imat će za ulogu postaviti vrijednost stupca nad drugom tablicom. Riječ je o stupcu *slobodno* iz tablice INVENTAR. Logika je ta da ako ne postoji neka vrijednost u zapisu nad stupcem *datum_vracanja*, odnosno ako knjiga nije zapravo vraćena, vrijednost stupca *slobodno* će biti FALSE. U suprotnom, ako je knjiga vraćena, postavlja se vrijednost TRUE nad stupcem *slobodno*.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `promjena_slobodno_BEFORE_UPDATE`
BEFORE UPDATE ON `posudba` FOR EACH ROW BEGIN
IF new.datum_vracanja IS NOT NULL THEN
    UPDATE inventar SET slobodno =1 WHERE inventarni_broj_ID =
new.inventarni_broj_ID;
ELSE
    UPDATE inventar SET slobodno =0 WHERE inventarni_broj_ID =
new.inventarni_broj_ID;END IF;
END
```

9.3. Okidač provjere članarine i stanja knjige

Možemo pogledati sljedeći okidač. U ovom primjeru, okidač će provjeriti stanje knjige u tablici INVENTAR i provjeriti članarinu člana, odnosno da li je članu knjižnice istekla članarina.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `provjera_knjige_AFTER_INSERT`
BEFORE INSERT ON `posudba` FOR EACH ROW BEGIN
DECLARE knjigaSlobodna TINYINT;
DECLARE posudeno INT;
DECLARE clanarinaVrijedi TINYINT DEFAULT 0;
```

```
SET knjigaSlobodna =(SELECT slobodno FROM inventar WHERE
new.inventarni_broj_ID = inventarni_broj_ID);

SET posudeno =(SELECT status_posudbe_ID FROM posudba WHERE
inventarni_broj_ID = new.inventarni_broj_ID AND datum_vracanja IS NULL);

IF posudeno = 1 AND knjigaSlobodna =0 THEN

      SIGNAL SQLSTATE '45000'

      SET MESSAGE_TEXT = 'Knjiga je već posuđena.'; END IF;

SET clanarinaVrijedi =

(SELECT c.clanarina_vrijedi FROM clanarina c, clan_knjiznice ck

WHERE new.clan_knjiznice_ID=c.clan_knjiznice_ID AND c.clanarina_vrijedi
=1);

IF clanarinaVrijedi IS NULL THEN

      SIGNAL SQLSTATE '45000'

      SET MESSAGE_TEXT = 'Članu knjižnice je istekla članarina.'; END IF;

END
```

9.4. Okidač provjere broja zaduženih knjiga

Zadnji okidač vezan za tablicu POSUDBA je za provjeru već posuđenih knjiga. Okidač će imati za ulogu provjeriti broj zaduženih knjiga koje je član knjižnice trenutno zadužio. Naravno, ovo pravilo ne vrijedi za članove knjižnice koji imaju članarinu za neograničeni broj posuđenih knjiga.

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`provjera_clanarine_BEFORE_INSERT` BEFORE INSERT ON `posudba` FOR EACH ROW
BEGIN

DECLARE vrstaClanarine INT;

DECLARE brojPosudba INT;

SET vrstaClanarine =(SELECT vrsta_clanarine_ID FROM clan_knjiznice WHERE
new.clan_knjiznice_ID = clan_knjiznice_ID);

IF vrstaClanarine != 3 THEN

SET brojPosudba =(SELECT count(new.clan_knjiznice_ID) FROM posudba WHERE
datum_vracanja IS NULL AND new.clan_knjiznice_ID =clan_knjiznice_ID );

IF brojPosudba >= 3 THEN

      SIGNAL SQLSTATE '45000'

      SET MESSAGE_TEXT = 'Član knjižnice je već zadužio 3 knjige.';

END IF; END IF; END
```

9.5. Okidač postavljanja stanja članarine

Iduća tablica nad kojom ćemo promatrati stanje unosa i modificiranja podataka je tablica CLANARAINA. Nad tablicom imamo okidač koji će automatski postavljati vrijednost stupca *clanarina_vrijedi*, odnosno da li je članarina aktivna ili nije.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `racun_BEFORE_INSERT`
BEFORE INSERT ON `clanarina` FOR EACH ROW BEGIN
DECLARE danasnjiDatum DATE;
SET new.datum_isteka = DATE_ADD(new.datum_placanja, INTERVAL 1 YEAR);
SET danasnjiDatum = CURDATE();
IF new.datum_isteka >=danasnjiDatum  THEN
      SET new.clanarina_vrijedi = 1;
ELSE
      SET new.clanarina_vrijedi = 0; END IF;
END
```

9.6. Okidač provjere članarine

Drugi primjer okidača je da li član knjižnice ima već važeću članarinu na osnovu stupca *clanarina_vrijedi*.

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`provjera_valjajuće_članarine_BEFORE_INSERT` BEFORE INSERT ON `clanarina`
FOR EACH ROW BEGIN
DECLARE clanarinaVrijedi TINYINT DEFAULT 0;
SET clanarinaVrijedi =(SELECT clanarina_vrijedi FROM clanarina WHERE
new.clan_knjiznice_ID =clan_knjiznice_ID AND clanarina_vrijedi =1);
IF clanarinaVrijedi=1 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Član knjižnice već ima valjano članstvo.'; END
IF;
END
```

10. Korisničko sučelje za rad

U zadnjem dijelu rada razvit ćemo potrebno sučelje za rad s bazom podataka. Do potrebne realizacije sučelja za rad dolazi iz potrebe da omogućimo jednostavniji i praktičniji rad knjižničarima unutar nekog drugog sustava koji će komunicirati s bazom podataka. S obzirom da govorimo o realnom primjeru GK Ivanić Grad, očekivano je da korisnik ima nekakvo sučelje za rad koje će biti intuitivnije i olakšan rad s bazom podataka.

Prije svega, moramo definirati okvire buduće aplikacije, odnosno same funkcionalnosti aplikacije. Aplikacija je zamišljena kao administrativna podrška sustavu GK Ivanić-Grad. Samim time, naš primarni korisnik je knjižničar/pomoćni knjižničar kojeg ćemo autentificirati, odnosno dodijelit ćemo mu atribute lozinke i email na već postojeće atribute da bi se moglo realizirati navedeno. Zamišljeno je da oni imaju pristup funkcionalnostima aplikacija, odnosno ne želimo da ostali rade promjene unutar baze podataka.

Forme za rad su napravljene kao web aplikacija. Razlog odabira web aplikacije je u tome što je značajan broj aplikacija u današnje vrijeme upravo web i mislim da je pogodnije nastaviti implementaciju, dodati nove uloge unutar sustava, a s druge strane MySQL Workbench pogodan je za rad s bazom podataka na web stranicama. Promatrajući knjižničara unutar baze podataka, možemo prepoznati njegove dvije glavne funkcionalnosti. Prva funkcionalnost je kreiranje, čitanje, editiranje i brisanje nad tablicom POSUDBA, dok je druga rad s tablicom CLAN_KNJIZNICE. Ove dvije funkcionalnosti su od presudne važnosti za rad knjižnice. Knjižničar će evidentirati i spremiti u bazu podataka dolazak novog člana knjižnice, a isto tako će se unijeti i posudba knjiga tog novog člana. Da bih zaokružili sliku, pored ovih dviju glavnih funkcionalnosti dodat ćemo funkcionalnost čitanja za rad s tablicama ČLANARINA, KNJIGA, INVENTAR, AUTOR i PREVODITELJ. Od navedenih tablica svakako ćemo omogućiti prikaz podataka na stranici te također omogućiti dodavanje novih zapisa, editiranje postojećih i brisanje.

Pošto je riječ o web aplikaciji koristiti ćemo se sljedećim jezicima:

- HTML,
- CSS,
- PHP,
- JavaScript.

Od navedenih jezika, najznačajniji nama je PHP. Dok su ostali jezici korišteni u svrhu izrade i dizajna web aplikacije na korisničkoj strani, PHP je jezik koji je kompatibilan s HTML pa samim time se lako koristi unutar njega. Ono što odvaja PHP od ostalih navedenih jezika

je to što radi na poslužiteljskoj strani aplikacije, a ne na klijentovoj kao što je slučaj s JavaScriptom. Postoje nekoliko aspekta zašto je PHP korišten u današnje vrijeme, a to su [14]:

- Programiranje na poslužiteljskoj strani,
- Programiranje u „Command line“,
- Programiranje desktop aplikacija.

Naime, pored glavnih funkcionalnosti mi ćemo koristiti PHP jezik da bi se iskoristila jedna od njegovih najboljih funkcionalnosti, a riječ je o radu s bazama podataka. Odnosno, PHP pruža podršku raznim sustavima za rad s bazom podataka pa tako i MySQL Workbench-u.

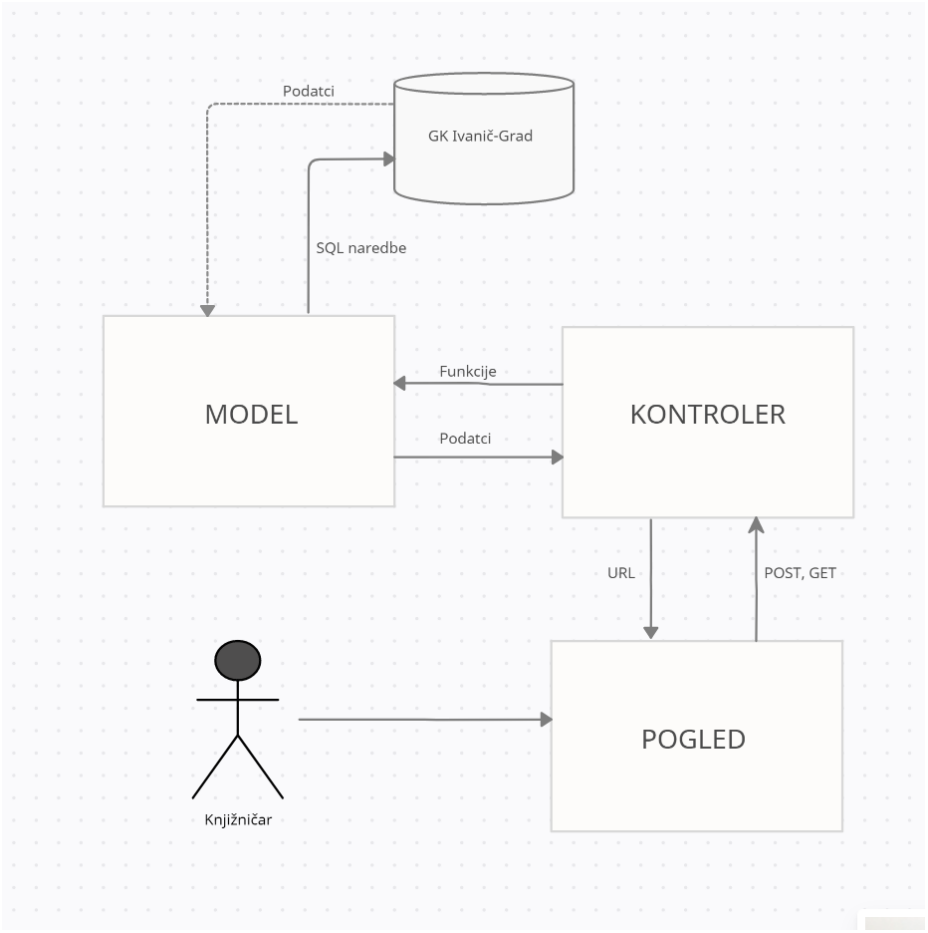
Što se tiče tehnologija korištenih za izradu aplikacije, korištene su sljedeće:

- Bootstrap,
- Font Awesome,
- MVC framework,
- XAMPP server.

Da bi forme bile uredne, koristio sam Bootstrap koji je zadužen za dizajn stranica, teksta, gumbova te jednostavnost prikaza web stranice u različitim rezolucijama. Što se tiče Font Awesome-a, korišten je zbog drugačijih stilova fonta i ikona koje su omogućene u besplatnoj verziji. Ono što nam je važnije od samoga dizajna je server koji je od presudne važnosti ako želimo raditi s bazom podataka. U tu svrhu korišten je XAMPP server.

10.1.MVC framework

Zadnja, ali najvažnija tehnologija koja je korištena za izradu formi je MVC framework. Riječ je o dizajniranoj aplikaciji koja razdvaja aplikacijske podatke i poslovnu logiku od vizualne web stranice [14]. MVC „framework podijeljen je na tri logička dijela, a to su model, kontroler i pogled. Za bolje razumijevanje „MVC Framework“-a, napravljen je dijagram za našu domenu. „Framework“ je objektno orijentiran, tako da moramo raditi klase koje će reprezentirati neku pojavu u aplikaciji. U našoj situaciji, to će biti intuitivno tablicama koje smo definirali u bazi podataka. Uzmemo li za primjer tablicu CLAN_KNJIZNICE, očigledno je da će biti potreba za klasom s istim imenom te ćemo sve metode vezane uz tu tablicu pisati pod tu klasu.

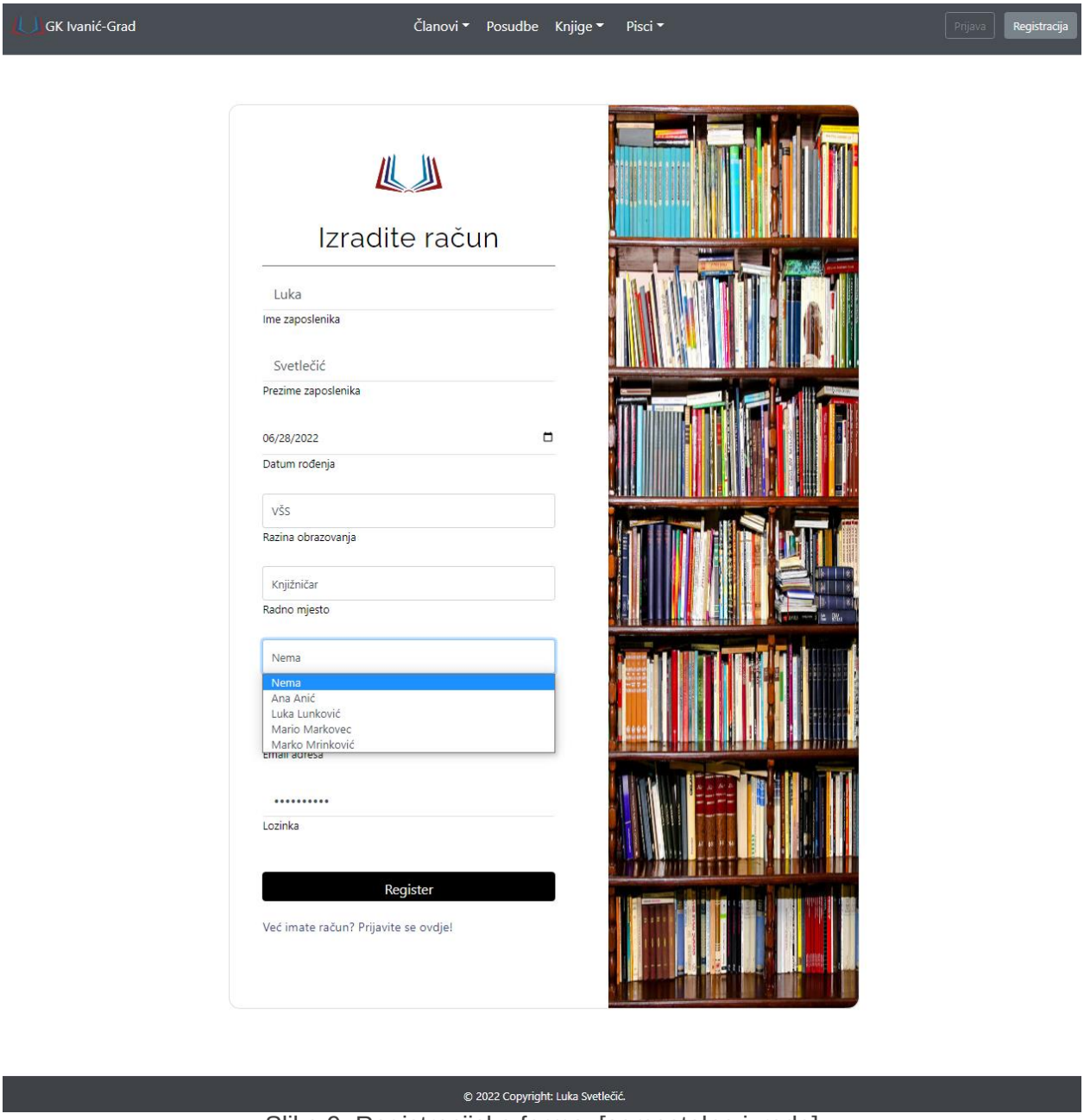


Slika 8: MVC za GK Ivanić-Grad. [samostalna izrada]

Kao što smo naveli, logika će se odvijati kroz 3 dijela. Model dio zadužen je za komunikaciju s bazom podataka uz pomoć SQL jezika. Baza podataka vraća podatke na osnovu, ako je naredba upit, što znači da će se sve akcije vezane uz manipulaciju bazom podataka izvršavati bez povrata. Model sam po sebi nije funkcionalan, ali ga zato pokreće kontroler koji uz pomoć funkcije poziva model da izvrši potrebnu akciju/e, a kao povratnu informaciju vrača podatke u slučaju upita, ili zastavicu ako je u pitanju manipulacija podatcima. Samim time zaključujemo da je kontroler zadužen za logiku aplikacije te čini glavnu poveznicu između baze podataka i pogleda kojeg ćemo sljedećeg definirati. Pogled je vizualni aspekt koji korisnik, u našem slučaju knjižničar, vidi i koristi. Na osnovu korisnikovog zahtjeva šalje se adresa te nove stranice i kontroler zna što mora učiniti. Na primjer, kada korisnik pritisne gumb za dodavanje nove posudbe, kontroler će to iščitati putem URL adrese. Postavlja se pitanje kako će kontroler doći do korisnikovih podataka. Naime, kontroler dohvaća podatke pomoću PHP globalnih varijabla GET i POST. POST koristimo da bi se očitao korisnikov unos, dok se s druge strane GET iščitava kao parametar zadan unutar adrese.

10.2. Autentifikacijske forme

Prva forma korištenja baze podataka vezana je uz to kako ćemo korisnika autentificirati, odnosno ne želimo svakome dati mogućnost rada s bazom podataka pa u tom slučaju moramo znati kada knjižničar ili pomoćni knjižničar koristi aplikaciju. Da bi se to omogućilo, uveli smo registraciju korisnika i samu prijavu. Da bi se mogla realizirati ova forma, dodana su nova dva stupca unutar tablice ZAPOSLENIK, a riječ je o *email_zaposlenika* i *lozinka*.

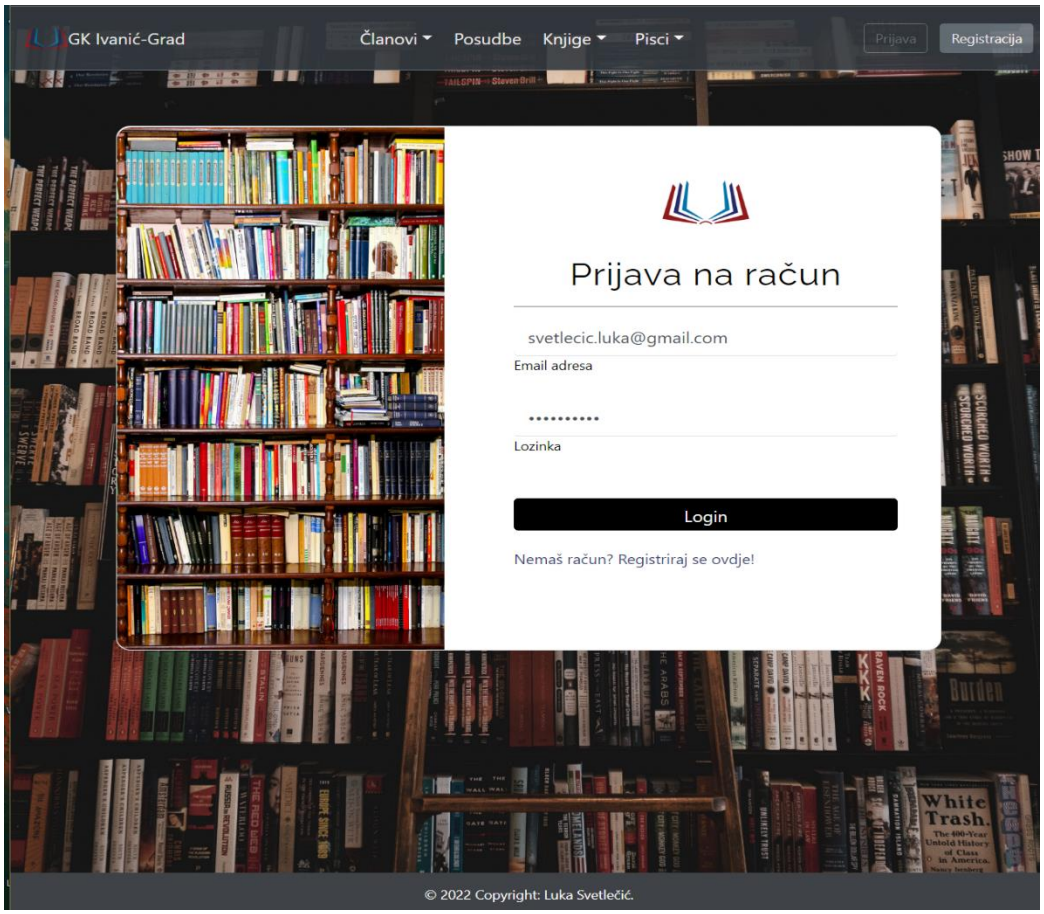


Slika 9: Registracijska forma. [samostalna izrada]

Knjižničar u ovoj situaciji mora unijeti svoje osobne podatke te odabrati radno mjesto i nadređenog, ako ga ima. Ovom formom radimo manipulaciju unutar baze podataka te se pohranjuje nova vrijednost unutar tablice ZAPOSLENIK putem sljedeće naredbe. Da bi se forma uspješno izvršila, prije pozivanje ove forme moramo pronaći vrijednosti koje korisnik unosi kroz padajuće liste, a to će se obaviti putem drugim metoda koje upravo pronalaze poslani podatak i vraćaju ključ iz primarne tablice.

```
INSERT INTO zaposlenik
(ime_zaposlenika,          prezime_zaposlenika,          datum_rodenja,
   razina_obrazovanja_ID,
vrsta_zaposlenika_ID, nadreden_ID, email_zaposlenika, lozinka)
VALUES(:ime_zaposlenika,:prezime_zaposlenika,:datum_rodenja,
:razina_obrazovanja_ID,:vrsta_zaposlenika_ID,:nadreden_ID,
:email_zaposlenika, :lozinka
```

S obzirom da korisnik mora napraviti račun, isto tako potrebno je i prijaviti se da bi se znalo koji knjižničar koristi aplikaciju, odnosno tko radi promjene nad bazom podataka.



Slika 10: Forma za prijavu knjižničara. [samostalna izrada]

Za razliku od registracijske forme, u formi za prijavu moramo provjeriti podatke, odnosno da li oni odgovaraju podatcima unutar tablica pa iz tog razloga selektiramo podatke i vraćamo zapis u kojem se vrijednosti poklapaju. Samim time, koristimo PHP varijablu SESSION koja je globalna i pohranjujemo vrijednosti u nju da bi znali da li je korisnik prijavljen ili se odjavio.

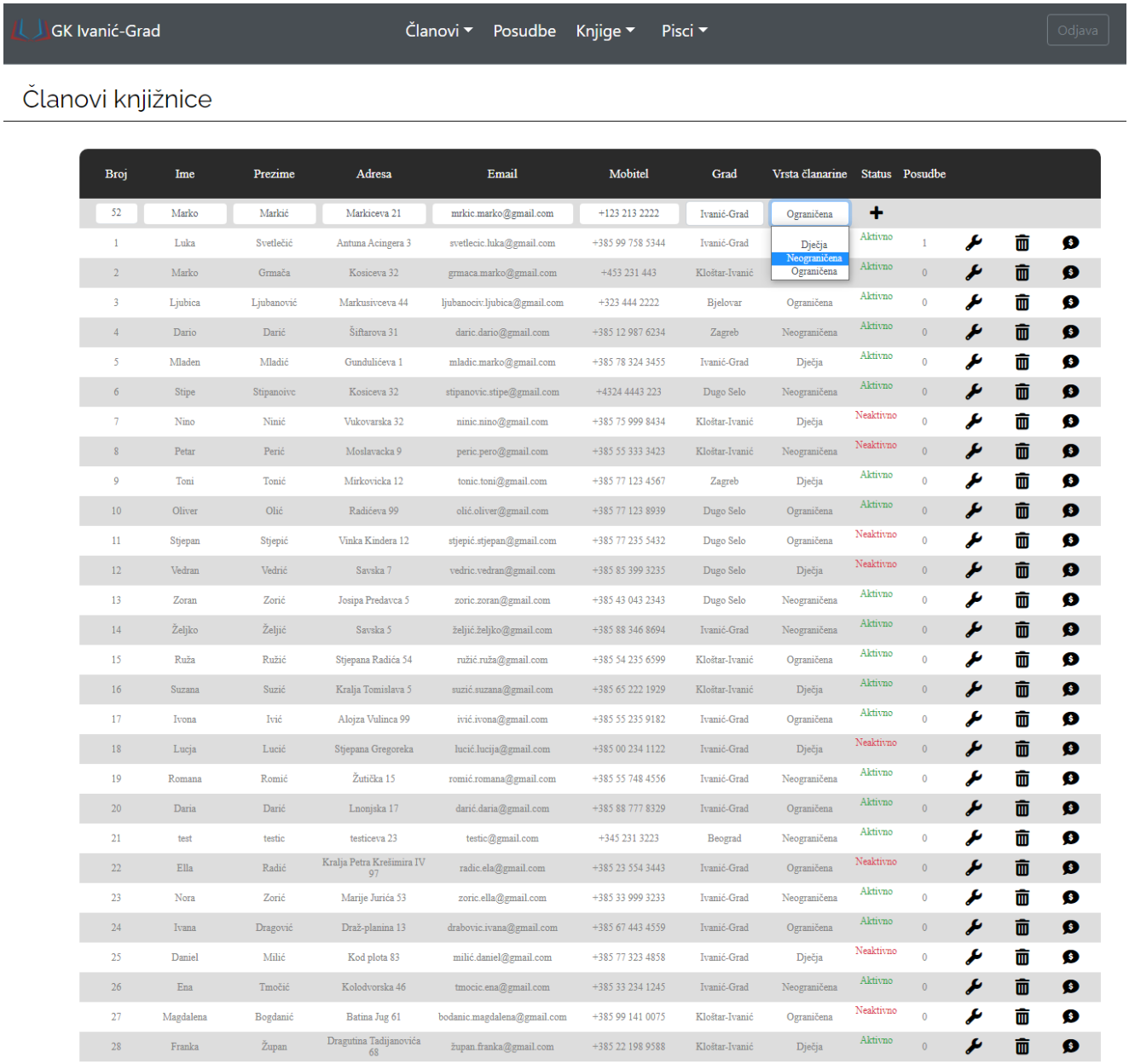
10.3.Forma za prikaz članova knjižnice

Knjižničar je do sada mogao samo čitati zapisa unutar baze podatke, ali u situaciji kada je prijavljen korisniku se dodjeljuju ovlasti korištenja same. Pored čitanja zapisa članova knjižnice, može obavljati operacije dodavanja novih zapisa, ažuriranja postojećih zapisa i brisanja starih. Sve ove operacije pokazane su u formi za prikaz članova knjižnice te je ujedno dodan stupac koji se odnosi na članarine tako da knjižničar može brzo obaviti novo plaćanje ako je potrebno ili provjeriti kada je zadnji put član platio članarinu.

Osim osobnih informacija svakoga člana, možemo primijetiti da se dohvaćaju i vrijednosti izvan tablice ČLAN_KNJIŽNICE, a to su vrijednosti da li član ima aktivnu članarinu i broj trenutnih posudba. Na formi su prikazani zapisi iz više tablica tako da naredba koja dohvaća te vrijednosti izgleda sljedeće:

```
SELECTck.clan_knjiznice_ID, ck.ime_clana, ck.prezime_clana,ck.adresa_clana,
ck.email_clana,ck.telefosnki_broj_clana,g.ime_grada,vc.ime_vrste_clanarine,
   (SELECTCOUNT(*) FROM posudba p WHERE p.clan_knjiznice_ID = ck.
   clan_knjiznice_ID
   AND p.status_posudbe_ID =1 OR p.status_posudbe_ID =3) AS broj_posudba
FROM clan_knjiznice ck
INNERJOIN grad g ON g.grad_ID = ck.grad_ID
INNERJOIN vrsta_clanarine vc ON
    vc.vrsta_clanarine_ID = ck.vrsta_clanarine_ID
ORDERBY ck.clan_knjiznice_ID ASC
```

Na slici 10. pored popisa članova knjižnice možemo primijetiti i formu za unos koja funkcionira slično kao forma za registraciju, odnosno pretražuje vrijednosti odabranog teksta unutar padajuće lsite pa pohranjujevrijednosti u novom zapisu.

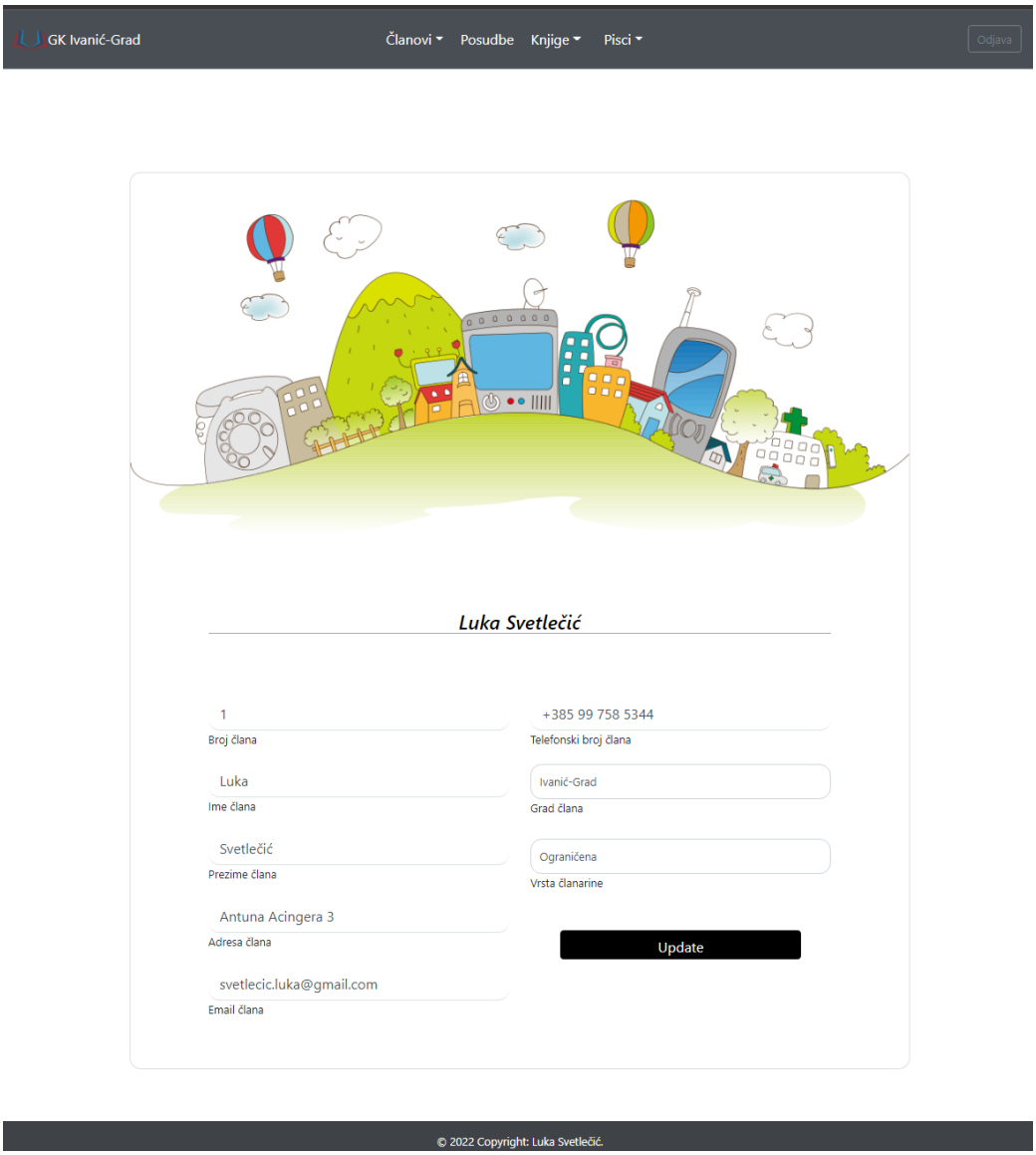


Slika 11: Forma za pregled i dodavanja članova. [samostalna izrada]

10.4.Forma za ažuriranje profila člana

Kao što smo naveli, svakoga člana knjižnice moguće je ažurirati ako dolazi do promjena osobnih informacija ili pogrešnog unosa. Naredba kojom to omogućujemo je sljedeća:

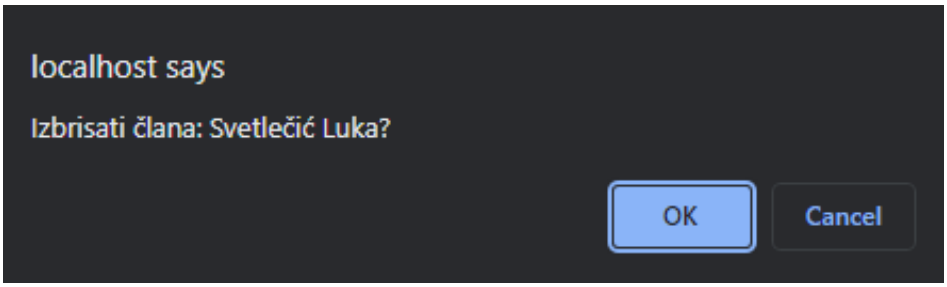
```
UPDATE clan_knjiznice SET clan_knjiznice_ID=:clan_knjiznice_ID,ime_clana =
:ime_clana,prezime_clana=:prezime_clana,adresa_clana=:adresa_clana,
email_clana = :email_clana,telefosnki_broj_clana = :telefosnki_broj_clana,
grad_ID = :grad_ID,vrsta_clanarine_ID = :vrsta_clanarine_ID
WHERE clan_knjiznice_ID = :clan_knjiznice_ID
```



Slika 12: Forma za ažuriranje člana. [samostalna izrada]

10.5.Forma za brisanje profila člana

Da bi se omogućilo brisanje članova, implementirana je potvrdna poruka koja se odaziva na gumb. Funkcionalnost može biti korisna, ako se željenog člana knjižnice želi izbrisati.

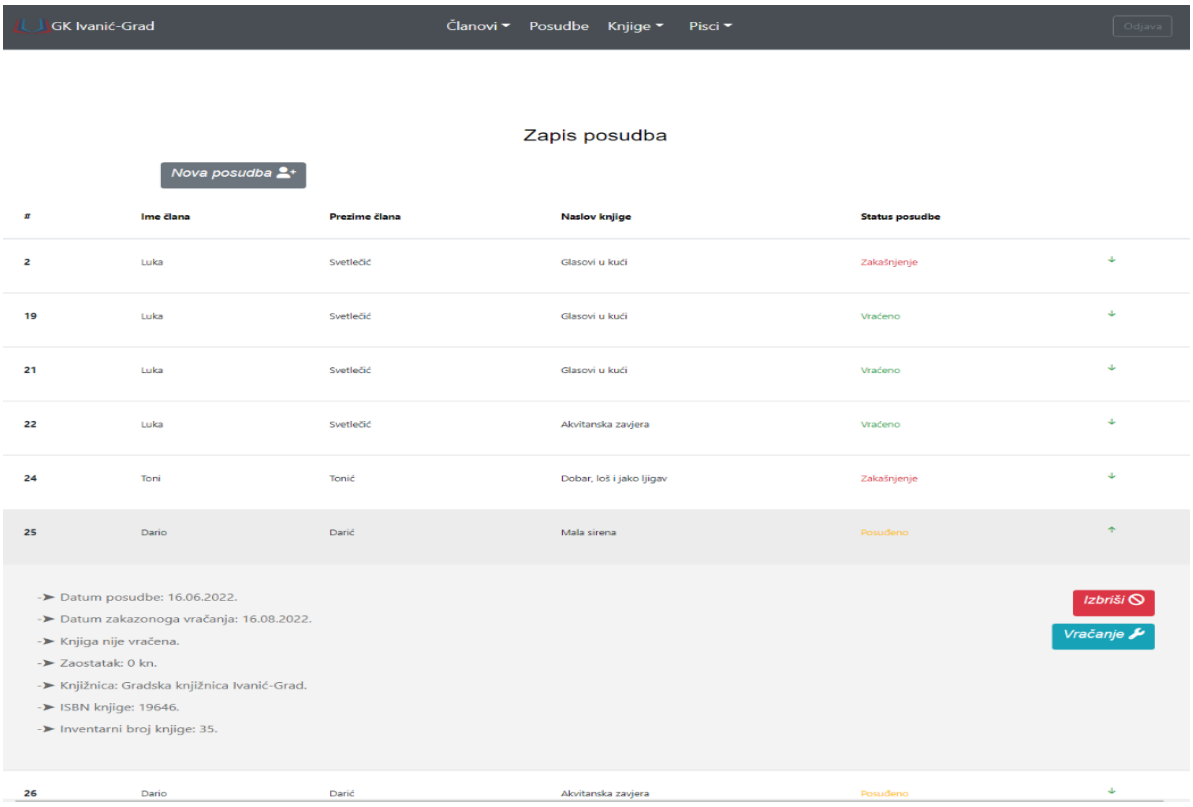


Slika 13: Forma za brisanje člana. [samostalna izrada]

10.6. Forma za pregled posudba

Forma posudbi sačinjena je od popisa svih kreiranih posudbi i njihovih podataka. Podatci su vezani za člana knjižnice koji je obavio posudbu, naslovu knjige, statusu posudbe, datumima posudbe, novčanom zaostatku, u kojoj knjižnici je obavljena posudba, ISBN knjige, i inventarnom broju knjige. Na formi se također nalazi gumb za dodavanje nove posudbe, za brisanje posudbe i za povrat knjige. Naveden podatke dohvatili smo sljedećom naredbom.

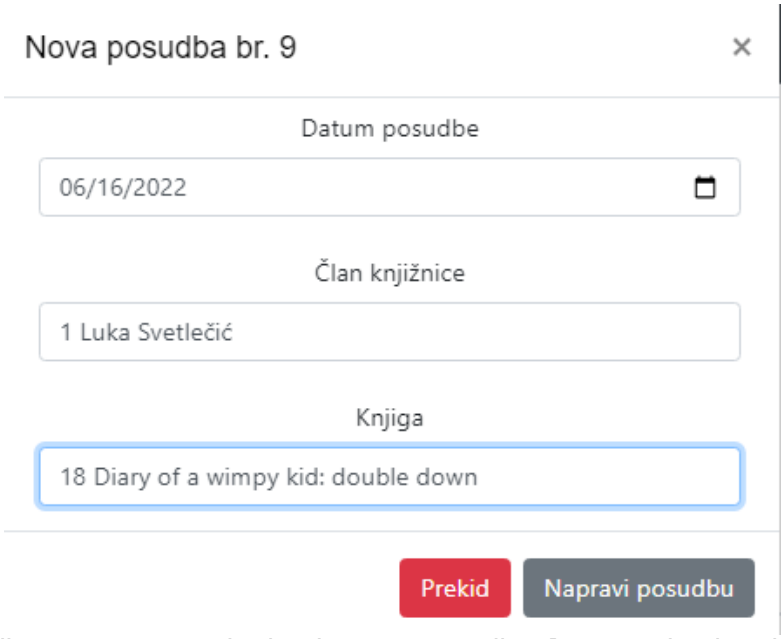
```
SELECT p.posudena_knjiga_ID, p.datum_posudbe, ck.prezime_clana,
p.datum_predvidenog_vracanja, p.datum_vracanja, p.novcani_zaostatak,
ck.ime_clana, ck.prezime_clana, s.ime_statusa_posudbe, k.ime_knjiznice,
i.isbn, i.inventarni_broj_ID,
    (SELECT k.naslov_knjige FROM knjiga k WHERE i.isbn = k.isbn) AS
naslov_knjige
FROM posudba p
INNERJOIN status_posudbe s ON s.status_posudbe_ID = p.status_posudbe_ID
INNERJOIN clan_knjiznice ck ON ck.clan_knjiznice_ID = p.clan_knjiznice_ID
INNERJOIN inventar i ON p.inventarni_broj_ID = i.inventarni_broj_ID
INNERJOIN knjiznica k ON k.knjiznica_ID = i.knjiznica_ID
ORDERBY p.posudena_knjiga_ID ASC
```



Slika 14: Forma za pregled posudba. [samostalna izrada]

10.7.Forma za kreiranje posudba

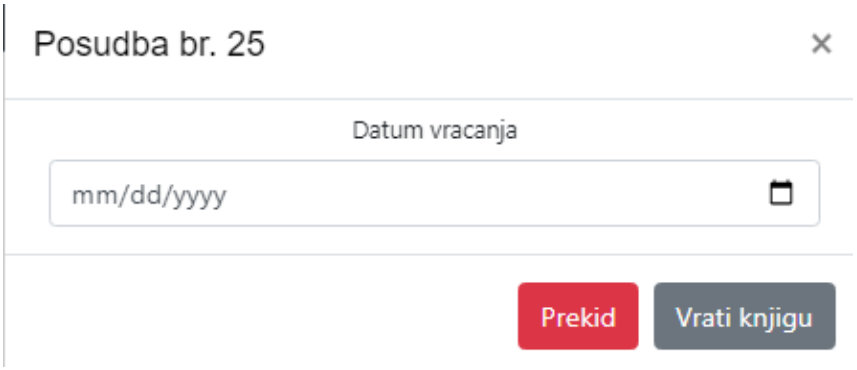
Forma je osmišljena kao pop-up model koje se otvara putem gumba. Od knjižničara se očekuju samo osnovni podatci posudbe, a to su datum posudbe te kojem članu knjižnice je posuđena knjiga. Forma unutar ispisuje broj posudbe.



Slika 15: Forma za kreiranje nove posudba. [samostalna izrada]

10.8.Forma za vračanje posudba

Forma za vraćanje svega zahtijeva jedan podatak, a to je kada je knjiga vraćena. Na osnovu tog podatka, u bazu se spremaju podatci vezani uz novčani zaostatak i o samom statusu posudbe. Forma se otvara na isti način kao forma za kreiranje posudbe, uz pomoć gumba.



Slika 16: Forma za kreiranje nove posudba. [samostalna izrada]

11. Zaključak

Kroz rad je prikazana nevjerojatno velika potreba pohrane za pohranom podatcima, a to se danas radi uz pomoć fizičkih medija, odnosno baze podataka. Drugim riječima „papir i olovku“ danas zamjenjuju baze podataka te gotovo svako poslovno okruženje koristi neku vrstu digitalne pohrane gdje dolazimo do važnosti korištenja baza podataka u današnje vrijeme. U radu sam se fokusirao isključivo na relacijske baze podataka, zato što su upravo one u doba pisanja ovoga rada i dalje najkorišteniji način pohrane.

Cilj ovoga rada bio je napraviti funkcionalnu i smislenu bazu podataka za Gradsku knjižnicu u Ivanić-Gradu, od samoga modeliranja pa sve do dijela gdje se baza podataka može primijeniti u pravom svijetu. Upravo zbog njenog načina poslovanja i specifičnih karakteristika, odabrao sam Gradsku knjižnicu kao dobar primjer iz poslovnog svijeta, samim time jer sam član knjižnice pa imam iskustva o njenom radu, a također smatram da je jako važno poznavati domenu poslovanju i njene ključne funkcionalnosti, jer na kraju krajeva svaka domena poslovanja zahtjeva drugačije principe modeliranja i domenu znanja.

Da bi model bio funkcionalan i samim time da bi se mogla ostvariti poslovna logika cijelog sustava, uz ograničenja putem ključeva, koristili smo se dodatnim ograničenjima, odnosno okidačima. Okidačima smo uzdigli cjelokupni model na razinu više jer se od korisnika više ne očekuje da unosi neke podatke koje današnje tehnologije mogu izračunati na osnovu relevantnih unosa ili da korisnik mora voditi brigu o stanjima drugih tablica itd.

Na kraju rada, prikazan je primjer kako implementiranu bazu podataka možemo koristiti, odnosno kako ona komunicira s drugim sustavima. Baza podataka sama po sebi nije značajna ako je promatramo kao zasebni sustav, obično se njeni potencijali najbolje realiziraju kroz druge sustave kao što su aplikacije, serveri i slično, s obzirom na to da je korištena već postojeća baza u implementaciji forma unutar MVC frameworka. Trenutna verzija aplikacije prikazuje glave poslovne procese vezanih za rad knjižnice.

12. Popis literature

[1] „Gradska knjižnica Ivanić-Grad“, *Gradska knjižnica Ivanić-Grad*. https://www.gkig.hr/ (pristupljeno 02. veljača 2022.).

[2] „baza podataka | Hrvatska enciklopedija“. https://www.enciklopedija.hr/natuknica.aspx?id=6404 (pristupljeno 07. veljača 2022.).

[3] H. Garcia-Molina, J. D. Ullman, i J. Widom, *Database systems: the complete book.*, Second Edition. New Jersey: Upper Saddle River, 2005.

[4] J. L. Harrington, *Relational Database Design and Implementation*. Morgan Kaufmann, 2016.

[5] R. Manger, *Osnove projektiranja baza podataka*. Zagreb: Srce, 2010.

[5] „What is Database Design Methodology? Different Phases of Design Methodology.“, *Computer Notes*, 24. ožujak 2013. https://ecomputernotes.com/database-system/rdbms/phases-of-design-methodology (pristupljeno 02. veljača 2022.).

[7] „DBMS Methodology (Conceptual)“. https://www.w3schools.in/dbms/conceptual-methodology/ (pristupljeno 02. veljača 2022.).

[8] „DBMS Methodology (Logical)“. https://www.w3schools.in/dbms/logical-methodology/ (pristupljeno 04. veljača 2022.).

[9] M. Varga, *Baze podataka: Konceptualno, logičko i fizičko modeliranje podataka*. Mladen Varga, 2020.

[10] „DBMS Methodology (Physical)“. https://www.w3schools.in/dbms/physical-methodology/ (pristupljeno 05. veljača 2022.).

[11] „What is MySQL? Everything You Need to Know“, *Talend - A Leader in Data Integration & Data Integrity*. https://www.talend.com/resources/what-is-mysql/ (pristupljeno 05. srpanj 2022.).

[12] „MySQL :: Why MySQL?“ https://www.mysql.com/why-mysql/ (pristupljeno 16. veljača 2022.).

[13] „MySQL :: MySQL Workbench Manual :: 1 General Information“. https://dev.mysql.com/doc/workbench/en/wb-intro.html (pristupljeno 16. lipanj 2022.).

[14] „PHP: What can PHP do? - Manual“. https://www.php.net/manual/en/intro-whatcando.php (pristupljeno 16. lipanj 2022.).

[15] R. Manger, *Baze podataka*. Zagreb: Element, 2012.

[16] K. Rabuzin, *Uvod u SQL*. Varaždin: Fakultet Organizacije i Informatike, 2011.

[17] M. Maleković i K. Rabuzin, *Uvod u baze podataka*. Varaždin: Fakultet Organizacije i Informatike, 2016.

[18] K. Rabuzin, *SQL napredne teme*. Varaždin: Fakultet Organizacije i Informatike, 2014.

13. Popis slika

14. Popis tablica