# Razvoj programskog rješenja za planiranje aktivnosti

**Komaić, Sandra**

**Undergraduate thesis / Završni rad**

**2020**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Dubrovnik / Sveučilište u Dubrovniku**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:155:644991

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-04-18**

*Repository / Repozitorij:*

Repository of the University of Dubrovnik

SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

SANDRA KOMAIĆ

# RAZVOJ PROGRAMSKOG RJEŠENJA ZA PLANIRANJE AKTIVNOSTI

ZAVRŠNI RAD

Dubrovnik, rujan, 2020.

SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

# RAZVOJ PROGRAMSKOG RJEŠENJA ZA PLANIRANJE AKTIVNOSTI

ZAVRŠNI RAD

Studij: Primijenjeno/ poslovno računarstvo
Kolegij: Baze podataka
Mentor: izv.dr.sc Mario Miličević
Student: Sandra Komaić

Dubrovnik, rujan, 2020.

# Sažetak

Cilj ovog rada je da se kroz teorijski i praktični dio prikaže primjena baze podataka. U teorijskom dijelu se opisuju osnovi pojmovi o bazi podataka: povijest nastanka baze podataka, sustav za upravljanje bazom podataka, funkcija sustava za upravljanje bazom podataka, jezici koji se koriste unutar baze, arhitektura baze i vrste baze podataka.

U praktičnom se opisuje desktop aplikacija izrađena u programskom okruženju NetBeans-u koja se povezuje na WAMP serverom. U ovom dijelu cilj je dočarati interakcija između same baze podataka i podataka s kojima manipulira.

# Abstract

The aim of paper is to present the application of the database through the theoretical and practical part. The theoretical part describes the basic concepts of the database: the origin of the database, its function, the languages used within the database, the architecture of the database and the types of database. The practical describes a desktop application created in the NetBeans programming environment that connects to a WAMP server. In this part, the goal is to portray the interaction between the database itself and the data it manipulates.

# SADRŽAJ

Sažetak.....I

Abstract.....II

1. Uvod..... 1

2. Osnovno o bazi podataka.....2

2.1. Povijest baze podataka.....2

2.2. DBMS.....3

2.2.1. Funkcija DBMS.....4

2.2.2. Arhitektura DBMS.....4

2.2.3. Vrste DBMS jezika.....4

2.2.3.1. Data Definition Language (DDL).....5

2.2.3.2. Data Manipulation Language (DML).....5

2.2.3.3. Data Control language (DCL)..... 6

2.2.3.4. Transaction Control Language(TCL)..... 6

2.3. Arhitektura baze podataka..... 7

2.3.1. Jednoslojna arhitektura..... 8

2.3.2. Dvoslojna arhitektura.....9

2.3.3. Troslojna arhitektura.....10

2.4. Vrste baze podataka.....11

2.4.1. Hijerarhijski model baze podataka..... 12

2.4.2. Mrežni model baze podataka..... 12

2.4.3. Relacijski model baze podataka.....13

2.4.4. Objektno orijentirani model baze podataka.....14

3. Izrada desktop aplikacije..... 16

3.1. Zahtjevi kod izrade sustava..... 16

3.2. Arhitektura sustava..... 16

3.3. Funkcija sustava.....19

3.4. Izrada tablica.....19

3.4.1. Tablica Registracija..... 20

3.4.2. Tablica Dogadaj.....20

3.4.3. Tablica Zapis..... 21

3.4.4. Tablica Zadaci..... 21

3.5. Dohvat podataka iz baze podataka..... 22
3.3. WAMP..... 24
3.4. Aplikacija Planer..... 25
3.4.1. Prijava..... 25
3.4.2. Registracija..... 26
3.4.3. Ekran Zadatak..... 27
3.4.3.1. Obavijest o događaju..... 28
3.4.3.1. Izbornik..... 29
3.4.4. Ekran Događaj..... 29
3.4.5. Ekran Bilješka..... 31
3.4.6. Ekran Moji događaji..... 32
3.4.7. Ekran Moje bilješke..... 33
3.4.8. Ekran Moji zadaci..... 33
4. Zaključak..... 35
5. Literatura..... 36
6. Prilozi..... 37
6.1. Prilozi slike..... 37
6.2. Prilozi tablica..... 37

# 1. Uvod

U današnje suvremeno doba baza podataka ima vrlo važnu ulogu u svakodnevnom životu. Baza podataka je organizirana zbirka strukturiranih podataka ili podataka koji se obično elektronički pohranjuju u računalni sustav. Pojedinac svakodnevno izvršava različite aktivnosti kao rezerviranje avionske karte, odlazak u banku, kupnja preko interneta... Sve te aktivnosti popraćene su traženjem pojedinih klijentovih podataka. Na primjer dolaskom u zračnu luku od klijenta se potražuje različiti podaci: njegovo ime, prezime, spol i količina prtljage. Svi ti podaci završavaju u bazi podataka te služe kao evidencija svakog putnika koji ima rezervaciju. Na ovakav način podaci se većinom spremaju u tradicionalnom brojčanom ili tekstualnom obliku.

Funkcija ovog završnog rada je napraviti aplikaciju koja omogućuje korisniku planiranje privatnih i poslovnih aktivnosti s fokusom na bazu podataka. Bitna je organizacija podataka u kategorije radi jednostavnosti i preglednosti. Želi li korisnik vidjeti samo događaje koje je unio, prikazat će mu se samo događaji raspoređeni po datumu.

# 2. Osnovno o bazi podataka

## 2.1. Povijest baze podataka

Baze podataka igrale su vrlo važnu ulogu u evoluciji računala. Prvi računalni programi razvijeni su početkom pedesetih godina prošlog stoljeća te su se u potpunosti usredotočili na kodiranje jezika i algoritama. U to vrijeme su računala bila divovski kalkulatori, a podaci su se smatrali ostacima obrade informacija. Tek kada su računala postala komercijalno dostupna i kad su ih poslovni ljudi počeli koristiti u stvarne svrhe, podaci su iznenada postali važni. 1960. Charles W. Bachman dizajnirao je integrirani sustav baza podataka DBMS, a ne dugo nakon IBM je stvorio vlastiti sustav baze podataka, poznat kao IMS. Oba su sustava baze podataka postala prethodnice navigacijskih baza podataka. Sredinom šezdesetih, kako su računala razvijala popularnost, brzinu i fleksibilnost, postali su dostupni mnogi tipovi baza podataka opće uporabe. Rezultat toga, kupci su zahtijevali da se razvije standard. Potaknut zahtjevima Bachman formira radnu grupu baza podataka. Ova grupa preuzela je odgovornost za dizajn i standardizaciju jezika koji se naziva zajednički poslovni orijentirani jezik (COBOL). Radna skupina za bazu podataka predstavila je ovaj standard 1971. godine, koji je također bio poznat i kao "CODASYL pristup."

CODASYL pristup bio je vrlo kompliciran sustav i zahtijevao je značajnu obuku. Ovisio je o "ručnoj" tehnici navigacije pomoću povezanog skupa podataka, koji je tvorio veliku mrežu. Pretraživanje zapisa moglo bi se obaviti jednom od tri tehnike: korištenjem primarnog ključa, premještanje odnosa iz jednog zapisa u drugi i skeniranje svih zapisa. Na kraju je CODASYL pristup izgubio popularnost jer su se na tržištu pojavili jednostavniji sustavi rada, ali ovaj sustav je početak baze podataka.

Danas su baze podataka svugdje i koriste se za poboljšanje svakodnevnog života. Mnoge usluge koje danas koristimo moguće su zahvaljujući bazama podataka: od osobne pohrane u oblaku do predviđanja vremena. Neke od trenutnih relacijskih baza podataka uključuju Oracle, MySQL i DB2. Također nastaju novi trendove koji se usredotočuju na to da tehnologiju učine dostupnom svima.

## 2.2. DBMS

Sustav za upravljanje bazama podataka, DBMS, softver je koji komunicira s krajnjim korisnicima, aplikacijama i samom bazom podataka za analizu podataka. "Sustav baza podataka" zajedno se odnosi na model baze podataka, sustav upravljanja bazama podataka i bazu podataka.

DBMS se sastoji od integriranog skupa računalnog softvera koji omogućuje interakciju s jednom ili više baza podataka i omogućuje pristup svim podacima koji se nalaze u bazi podataka. Pruža različite funkcije koje omogućuju upravljanje bazom podataka i njezinim podacima (unos, pohranu i dohvaćanje velike količine informacija) koji se mogu svrstati u četiri glavne funkcionalne skupine:

1. **Definicija podataka** - stvaranje, izmjena i uklanjanje definicija koje definiraju organizaciju podataka.
2. **Ažuriranje** - umetanje, izmjena i brisanje stvarnih podataka.
3. **Dohvaćanje** - pružanje informacija u obliku koji je upotrebljiv ili za daljnju obradu u drugim aplikacijama.
4. **Administracija** - registriranje i nadgledanje korisnika, provođenje sigurnosti podataka, nadgledanje performansi, održavanje integriteta podataka, bavljenje kontrolom i vraćanje podataka koji su oštećeni nekim događajem, poput neočekivanog kvara sustava.

Dobiveni podaci mogu biti dostupni u obliku jednakom obliku pohranjenom u bazi podataka ili u novom obliku dobivenom izmjenom ili kombiniranjem postojećih podataka iz baze podataka.

### 2.2.1. Funkcija DBMS

Sustav upravljanja bazom podataka (DBMS) izvlači podatke iz baze podataka kao odgovor na upite. Zapisi i datoteke baze podataka moraju biti organizirani kako bi se omogućilo preuzimanje informacija. Upiti su glavni način na koji korisnici dohvaćaju podatke iz baze podataka. Važnost DBMS-a proizlazi iz njegove sposobnosti da definira nove odnose od osnovnih koji su dati u tablicama i da ih koristi za dobivanje odgovora na upite. Obično korisnik daje niz znakova, a računalo traži u bazi podataka odgovarajući niz i pruža izvorne informacije u kojima se ti znakovi pojavljuju.

### 2.2.2. Arhitektura DBMS

Fizički su poslužitelji baze podataka namjenska računala koja sadrže stvarne baze podataka i pokreću samo DBMS i srodni softver. Poslužitelji baza podataka obično su multiprocesorska računala s obilnom memorijom i RAID diskovnim nizovima koji se koriste za stabilnu pohranu. Hardverski akceleratori baze podataka, povezani na jedan ili više poslužitelja putem brzog kanala, također se koriste u okruženjima za obradu transakcija velikih količina. DBMS-ovi se nalaze u srcu većine aplikacija baza podataka. DBMS-ovi se mogu graditi oko prilagođenog više zadataka jezgre s ugrađenom mrežnom podrškom, ali moderni DBMS-ovi se oslanjaju na standardni operativni sustav za pružanje ovih funkcija.

### 2.2.3. Vrste DBMS jezika

DBMS mora osigurati odgovarajuće jezike i sučelja za svaku kategoriju korisnika za izražavanje upita i ažuriranja baze podataka. Jezici baze podataka koriste se za stvaranje i održavanje baze podataka na računalu. Postoji veliki broj baza podataka kao što su Oracle, MySQL, MS Access. SQL izrazi koji se obično koriste u Oracle mogu se definirati kao jezik za definiranje podataka (DDL), jezik kontrole podataka (DCL) i jezik manipulacije podacima. (DML).

#### 2.2.3.1. Data Definition Language (DDL)

DDL se koristi za određivanje sheme baze podataka te se koristi za pohranu metapodataka poput broja tablica, shema, njihovih imena, indeksa, stupaca u svakoj tablici, ograničenja u bazi podataka.

Koristi se uglavnom za stvaranje datoteka, baza podataka, rječnika podataka i tablica u bazama podataka te za određivanje strukture svake tablice, skupa pridruženih vrijednosti sa svakim atributom, ograničenja integriteta, podataka o sigurnosti i autorizaciji za svaku tablicu i fizičke strukture pohrane svake tablice na disku.

Operacije koje se mogu izvoditi na bazi podataka pomoću DDL-a su:

- **CREATE** - kreiranje objekta u bazi podataka.
- **RENAME** - mijenjanje naziva objekta u bazi podataka.
- **DROP** - uklanjanje objekta iz baze podataka.
- **TRUNCATE** - brisanje svih podatka iz baze podataka.
- **COMMENT** - komentiranje.
- **ALTER** - promjena strukture baze podataka.

Ove se naredbe koriste za ažuriranje tablica baze podataka.

#### 2.2.3.2. Data Manipulation Language (DML)

Jezik koji pruža skup operacija za podršku osnovnim operacijama manipulacije podacima u bazama podataka. Omogućuje korisnicima umetanje, ažuriranje, brisanje i preuzimanje podataka iz baze podataka. Dio DML-a koji uključuje pretraživanje podataka naziva se jezikom upita.

Operacije koje se mogu izvoditi na bazi podataka pomoću DML-a su:

- **SELECT -** čitanje podatka iz baze podataka.
- **INSERT -** umetanje podataka u bazu podataka.
- **DELETE -** brisanje podataka iz baze podataka.

- **UPDATE -** ažuriranje podataka iz baze podataka.

#### 2.2.3.3. Data Control language (DCL)

DCL kontrolira pristup podacima i bazi podataka koristeći naredbe poput GRANT i REVOKE. Administrator baze podataka naredbom GRANT dopušta korisniku privilegije i pristup objektima baze podataka. Dodijeljene privilegije mogu biti SELECT, ALTER, DELETE, EXECUTE, INSERT, INDEX... Osim dopuštanja, privilegije može opozvati (ponovo ih oduzeti) pomoću naredbe REVOKE.

- **GRANT**- odobravanje pristupa korisniku bazi podataka.
- **REVOKE** - opoziv pristupa korisniku bazi podataka.

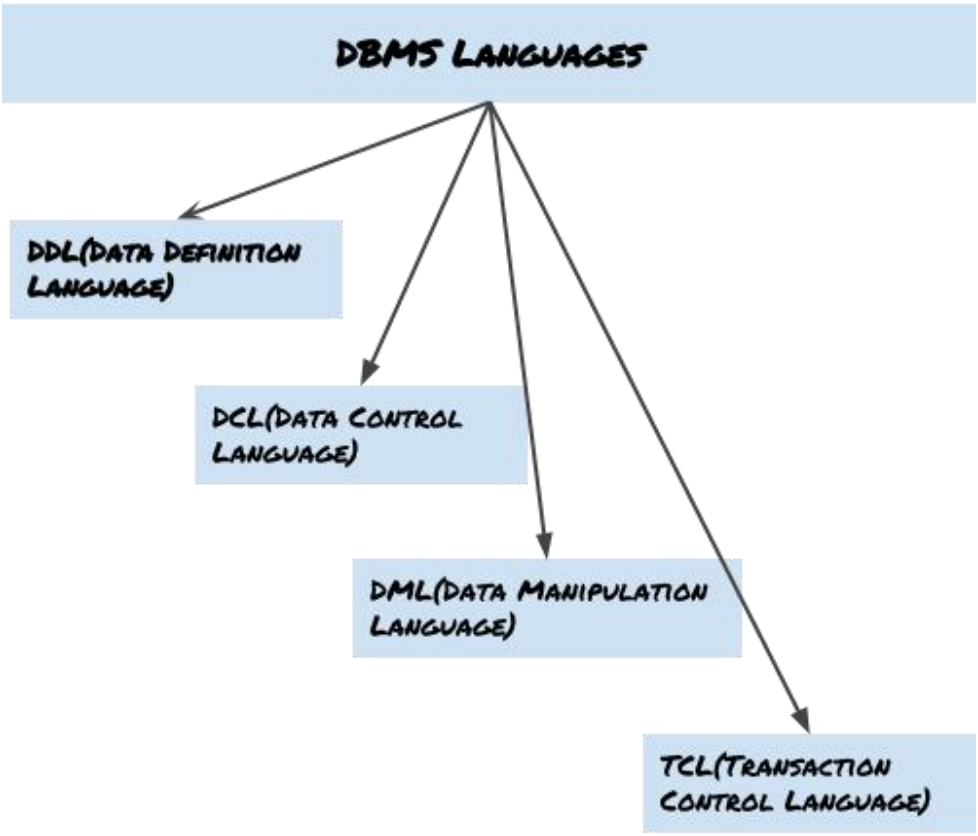
Operacije koje trebaju dopuštenje REVOKE su :
CONNECT, INSERT, EXECUTE, DELETE, UPDATE i SELECT.

#### 2.2.3.4. Transaction Control Language(TCL)

TCL se koristi za pokretanje promjena izvršenih u DML izrazu. TCL se može grupirati u logičku transakciju.

Operacije koje se mogu izvoditi na bazi podataka pomoću TCL-a su:

- **COMMIT**- nakon upisivanje DML naredba COMMIT se koristi za trajno spremanje podataka u bazu podataka .
- **ROLLBACK**- vraća bazu podataka na stanje prije promjena.




**Slika 1. Vrste DBMS jezika [2]**

## 2.3. Arhitektura baze podataka

Na arhitekturu sustava baza podataka jako utječe primarni računalni sustav na kojem radi sustav baze podataka. Sustavi baze podataka mogu biti centralizirani ili klijent-poslužitelj, gdje jedan poslužitelj izvršava rad u ime više klijentskih računala. Sustavi baza podataka mogu se također dizajnirati za korištenje paralelnih računalnih arhitektura. Arhitektura DBMS-a ovisi o računalnom sustavu na kojem se izvodi. Na primjer, u DBMS arhitekturi klijent-poslužitelj sustavi baza podataka na poslužiteljskom stroju mogu pokrenuti nekoliko zahtjeva koje je uputio klijentsko računalo. Arhitektura baze podataka koristi programske jezike za dizajniranje određene vrste softvera za tvrtke ili organizacije. Usmjerena je na dizajn, razvoj, implementaciju i održavanje računalnih programa koji pohranjuju i organiziraju informacije za tvrtke, agencije i institucije. Dizajn DBMS-a ovisi o njegovoj

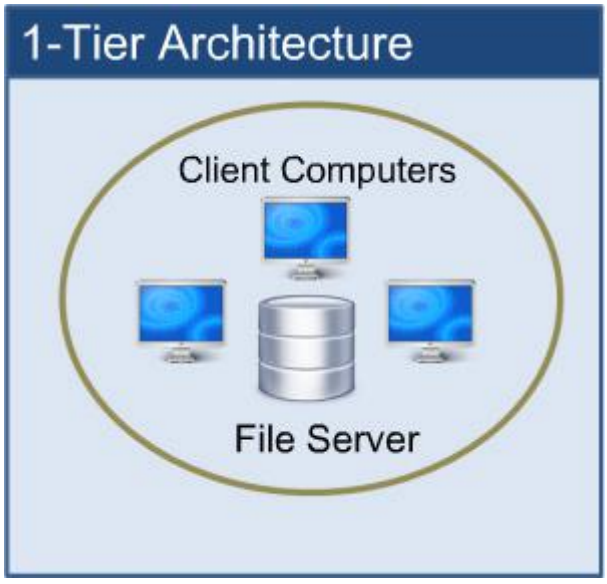
arhitekturi,a može biti centralizirana, decentralizirana ili hijerarhijska, dok arhitektura DBMS-a se grupira kao jednoslojna ili višeslojna:

1. **jednoslojna arhitektura**
2. **dvoslojna arhitektura**
3. **troslojna arhitektura**

### 2.3.1. Jednoslojna arhitektura

Jednoslojna arhitektura uključuje dodavanje svih potrebnih komponenti za softversku aplikaciju ili tehnologiju na jedan poslužitelj ili platformu. U osnovi, jednoslojna arhitektura drži sve elemente aplikacije: klijenta, poslužitelja i bazu podataka na jednom računalu. U ovoj vrsti arhitekture baza podataka je lako dostupna na klijentskom računalu, a bilo koji zahtjev klijenta ne zahtijeva mrežnu vezu za izvođenje akcije na bazi podataka. Ova vrsta arhitekture smatra se najjednostavnija, ali ovakva arhitektura se rijetko koristi u proizvodnji.

Jednoslojna arhitektura se može predočiti sljedećom slikom:

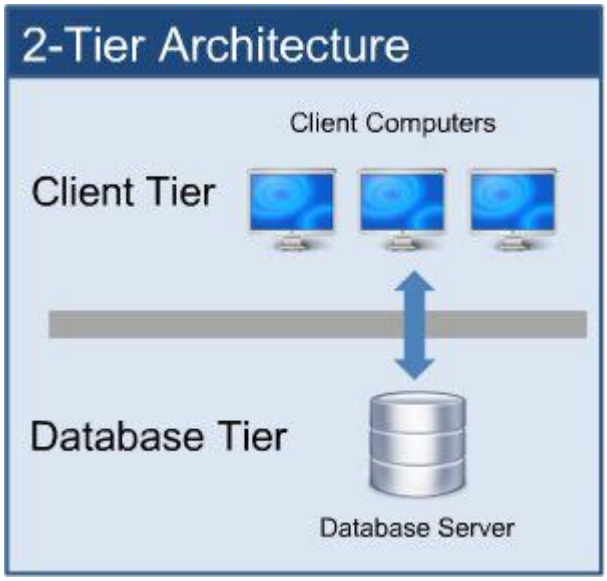



**Slika 2. Jednoslojna arhitektura [3]**

### 2.3.2. Dvoslojna arhitektura

U dvoslojnoj arhitekturi sustav baze podataka prisutan je na poslužitelju, a DBMS aplikacija je prisutna na klijentskom računalu te su međusobno povezana putem pouzdane mreže. Ne postoji posrednik između klijenta i poslužitelja. Kad god klijentsko računalo zatraži pristup bazi podataka prisutnoj na poslužitelju koristeći jezik upita poput SQL-a, poslužitelj izvršava zahtjev u bazi podataka i vraća rezultat natrag klijentu. Sučelje aplikacijske veze poput JDBC, ODBC koristi se za interakciju između poslužitelja i klijenta. ODBC (Open Database Connectivity) pruža API koji omogućava programu sa strane klijenta da poziva DBMS. Ovakva arhitektura pruža DBMS-u dodatnu sigurnost jer nije izravno izložena krajnjem korisniku. Također, sigurnost se može poboljšati dodavanjem provjera sigurnosti i provjere autentičnosti u aplikacijskom sloju.

Dvoslojna arhitektura se može predočiti sljedećom slikom:




**Slika 3. Dvoslojna arhitektura [3]**

### 2.3.3. Troslojna arhitektura

Troslojna arhitektura sadrži aplikacijski sloj između korisnika i DBMS, koji je odgovoran za priopćavanje korisnikovog zahtjeva DBMS sustavu i slanje odgovora iz DBMS-a korisniku. Troslojna arhitektura razdvaja slojeve jedan od drugog na temelju složenosti korisnika i načina na koji koriste podatke prisutne u bazi podataka. Različita je upotreba s različitim aplikacijama, a može se koristiti u web aplikacijama i distribuiranim aplikacijama. Cilj troslojne arhitekture je: neovisnost programa i podataka, podržavanje višestrukih prikaza podataka, preporučeno za podršku karakteristika DBMS-a i odvojiti korisničke aplikacije i fizičku bazu podataka.

Troslojna arhitektura ima sljedeće slojeve:

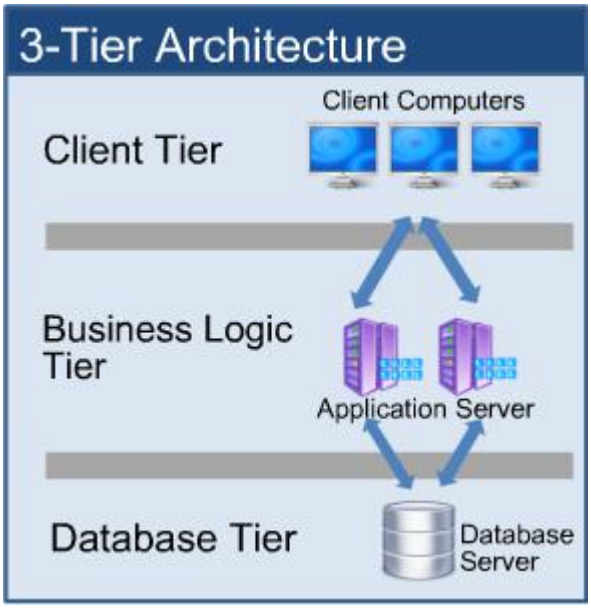
1. Korisnički sloj (računalo, tablet, mobilni telefon itd.)
2.Aplikacijski sloj (poslužitelj)
3. Sloj podataka

**Korisnički (prezentacijski) sloj** − ovaj sloj se sastoji od korisničkog sučelja. Grafički je dostupno putem web preglednika ili web-aplikacije koja prikazuje sadržaj i informacije korisne krajnjem korisniku. Ova se razina često gradi na web tehnologijama poput HTML5, JavaScript, CSS ili putem drugih popularnih okvira za web razvoj i komunicira s drugim slojevima putem API poziva.

**Aplikacijski (srednji) sloj** − na ovom nivou nalaze se aplikacijski poslužitelj i programi koji pristupaju bazi podataka. Aplikacijski sloj sjedi djeluje kao posrednik između krajnjeg korisnika i baze podataka.

**Sloj podataka** − ovaj sloj se sastoji od baze podataka, sustava za pohranu podataka i sloj pristupa podacima.

Troslojna arhitektura se može predočiti sljedećom slikom:




**Slika 4. Dvoslojna arhitektura [3]**

## 2.4. Vrste baze podataka

Model baze podataka definira logički dizajn i strukturu baze podataka i definira kako će podaci biti pohranjeni i ažurirani u sustavu za upravljanje bazama podataka. Iako je relativni model najčešće korišteni model baze podataka, postoje i drugi modeli:

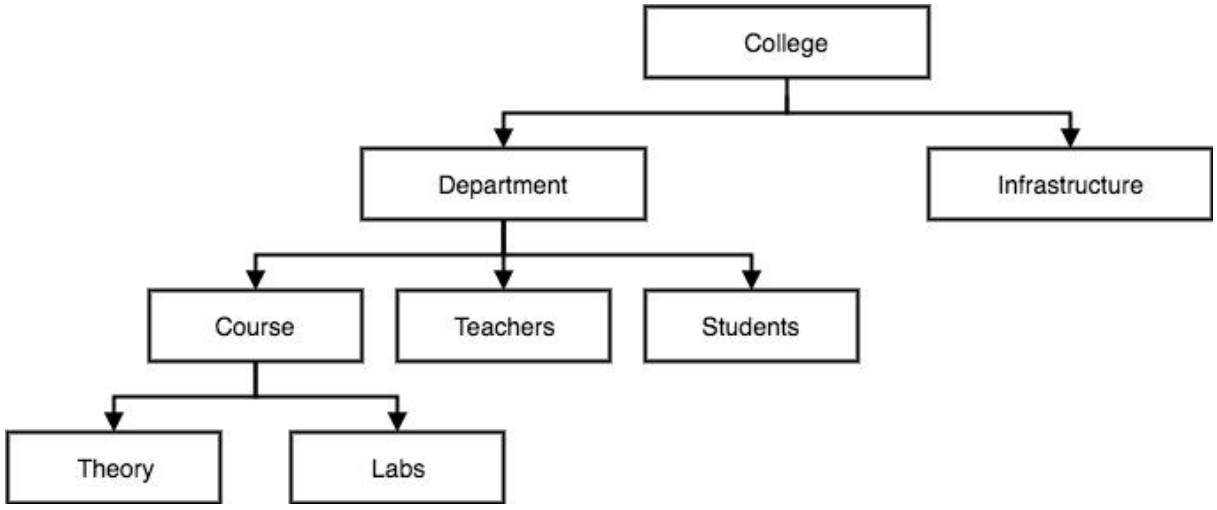
1. **Hijerarhijski model**
2. **Mrežni model**
3. **Relacijski model**
4. **Objektno orijentirani model**

**Slika 5. Modeli baze podataka [1]**

### 2.4.1. Hijerarhijski model baze podataka

Ovaj model baze podataka organizira entitete u stablo-strukturu, s jednim korijenom, na koje su povezani svi ostali entiteti. Hijerarhijski model baze podataka nalaže da svaki podređeni entitet ima samo jednog roditelja, dok svaki nadređeni entitet može imati jedan ili više podređenih entiteta. Da bi se dohvatili podaci iz hijerarhijske baze podataka, potrebno je početi potragu od korijenskog čvora do posljednjeg čvora.

Hijerarhijski model baze podataka se može predočiti sljedećom slikom:

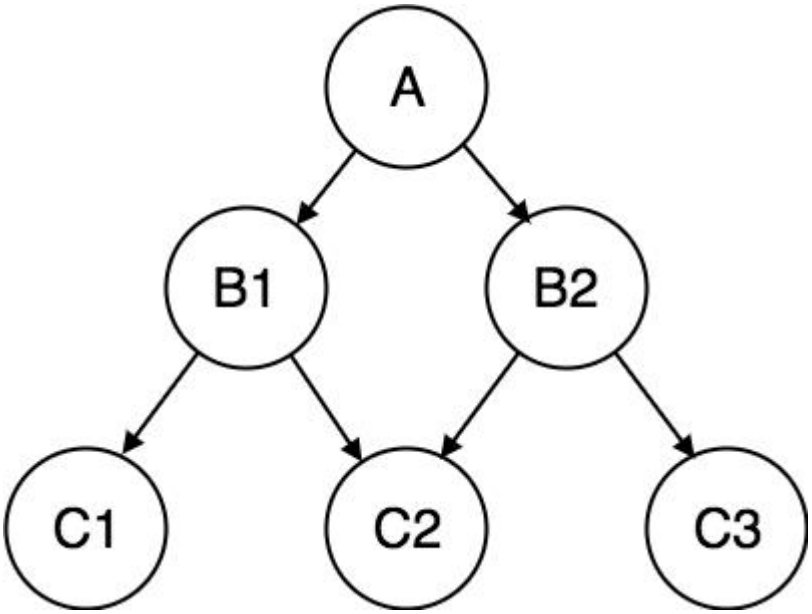



**Slika 6. Hijerarhijski model baze podataka [5]**

### 2.4.2. Mrežni model baze podataka

Mrežni model baze podataka je proširenje hijerarhijskog modela baze podataka. Podaci su u ovom modelu organizirani više poput grafikona, i mogu imati više od jednog roditeljskog čvora, a podaci su više povezani jer je u ovom modelu baze podataka uspostavljeno više veza. Također, kako su podaci povezani, time je i pristup podacima jednostavniji i brži.

Mrežni model baze podataka se može predočiti sljedećom slikom:

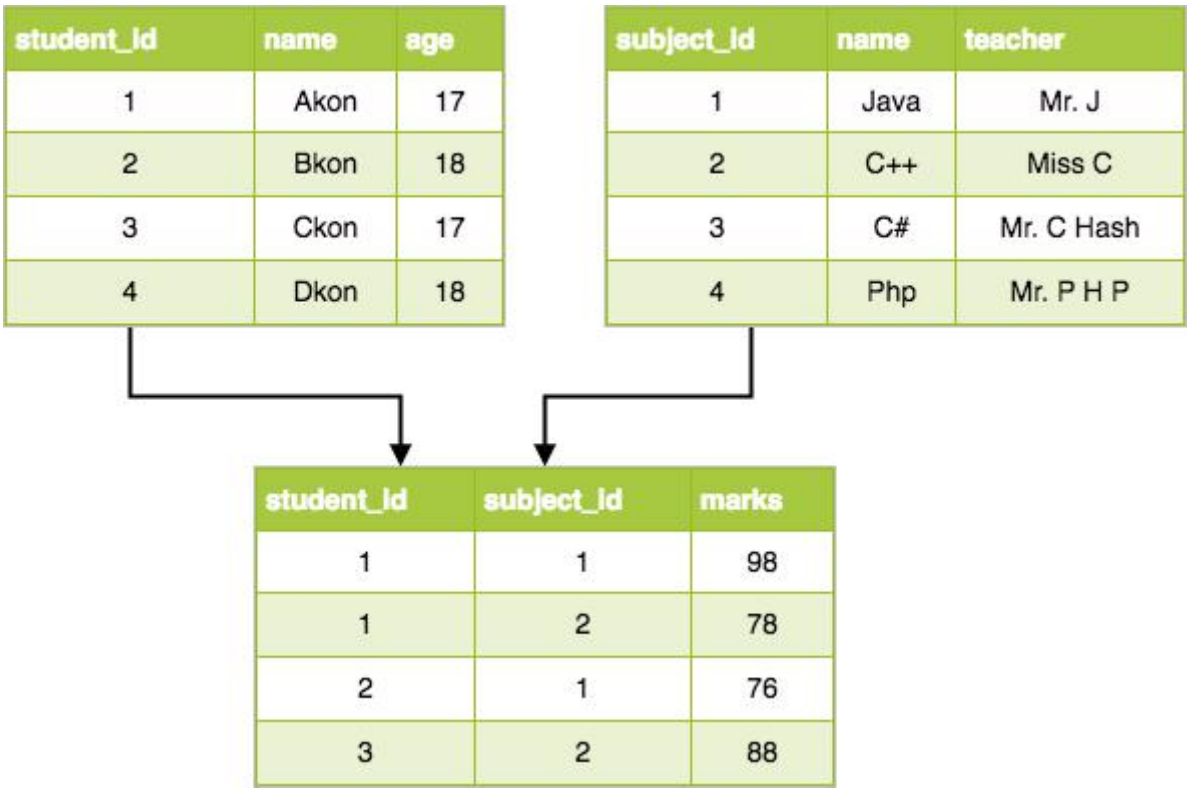



**Slika 7. Mrežni model baze podataka[5]**

## 2.4.3. Relacijski model baze podataka

Relacijski model baze podataka je najkorištenija baza podataka. Koriste se tamo gdje povezanosti između datoteka ili entiteta ne može biti izražene vezama. Ova vrsta DBMS definira odnose baze podataka u obliku dvodimenzionalnih tablica, poznatih i kao relacije. Za razliku od mrežne baze podataka, relacijska ne podržava mnoge veze s mnogim vezama. Relacijska baza podataka obično imaju unaprijed definirane tipove podataka koje može podržavati.

Relacijski model baze podataka se može predočiti sljedećom slikom:




**Slika 8. Relacijski model baze podataka [5]**

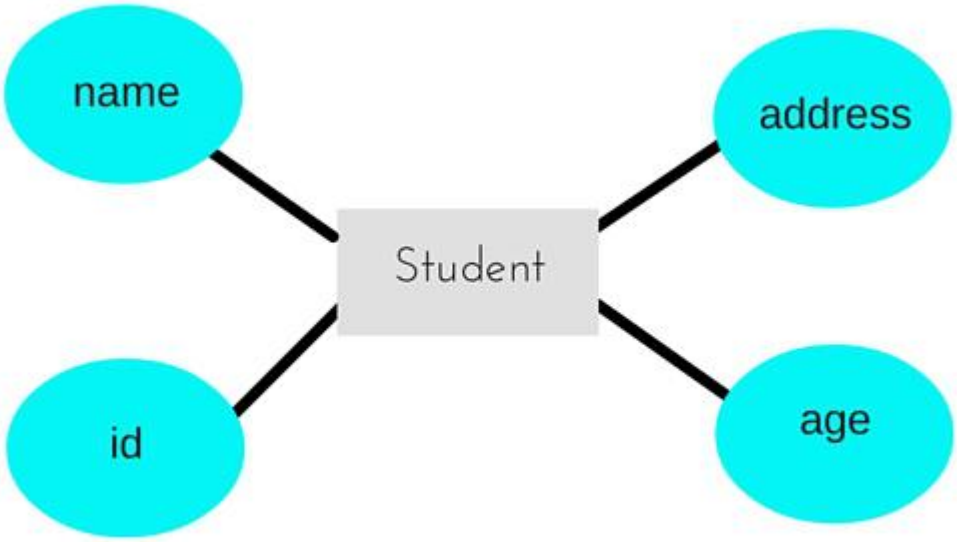
## 2.4.4. Objektno orijentirani model baze podataka

Objektno orijentirani model baze podataka pohranjuje i manipulira složenijim podatkovnim strukturama, zvanim objekti, koji su organizirani u hijerarhijske klase koje mogu naslijediti svojstva iz nadređenih klasa. Objekti koji se pohranjuju u bazi podataka imaju atribute i metode koje definiraju što treba učiniti s podacima.

Ovaj je model dobar za izradu baze podataka koja se nakon toga može pretvoriti u tablice u relacijskom modelu.

Na primjer. Ako moramo osmisliti bazu podataka u kojoj će Student biti entitet s atributima ime, prezime, starost, adresa... Kako je adresa općenito složena, to može biti drugi entitet Mjesto s atributima naziv ulice, poštanski broj, grad... Između entiteta Student i Mjesto postojat će veza.

Objektno orijentirani model baze podataka se može predočiti sljedećom slikom:




**Slika 9. Objektno orijentirani model baze podataka [5]**

# 3. Izrada desktop aplikacije

Cilj izrade ove aplikacije je prikazi koje su mogućnosti baze podataka i kako se baza podataka može iskoristiti u stvarnim događajima.

## 3.1. Zahtjevi kod izrade sustava

Prije izrade svake aplikacije potrebno je prikupiti dovoljno informacija o zahtjevu na kojim će se temeljiti pojedina aplikacija. Svrha zahtjeva je odgovoriti na dva najvažnija pitanja "Tko će koristiti pojedini sustav" i "Koja će biti funkcija sustava".
Zahtjevi se dijele u dvije vrste: korisnički zahtjevi i zahtjevi za razvojni tim.
Korisnički zahtjevi opisuju što bi rješenje trebalo raditi sa stajališta korisnika.
U slučaju aplikacije Planer korisnik izrazi da bi želio sustav koji će biti kao osobni dnevnik koji bi sadržavao sve aktivnosti koje su bitne za korisnika. Mogao bi unositi događaje na koje planira ići, zadatke koje želi obavljati u određenom periodu te bilješke koje bih htio zapisati odmah u željenom trenutku. Korisnikova želja je da mu kod događaja dolazi obavijest koja će ga obavijestiti da se bliži datum važnog događaja na lokaciji i vremenu već unaprijed upisanom od stane korisnika.
Nakon što korisnik iznese svoje zahtjeve, analizira se da li je sve razumljivo, koje će biti funkcionalnosti sustava i koja je specifikacija zahtjeva. Zatim se rangira zahtjev na dijelove koji su nužni: upis događaja, zadataka, bilješki, dijelovi koje bi sustav trebao imati: obavijest i izbornik, dijelovi koji nisu tako važni i mogu biti izostavljeni: obavijest kod isteka vremena za zadatke.

## 3.2. Arhitektura sustava

Arhitektura sustava konceptualan je model koji definira ponašanje, strukturu i bolji pogled na sam sustav. Tijekom izgradnje arhitekture sustava cilj je prepoznati i oblikovati podsustave. Kreće se od postojećeg stanja, ono što je poznato o sustavu pa sve do biranja alata i tehnologija. U globalu arhitektura sustava je faza oblikovanja koja kreira vrlo grub opis rješenja, a uloga arhitekture sustava je shvaćanje kakav

sustava je potrebno izraditi, olakšava dijeljenje cjelina na podcjeline. Arhitekturu sustava aplikacije bit će ilustrirana dijagramom slučajeva korištenja. (Slika 10.)

Dijagram slučajeva korištenja primarni je oblik softverskih zahtjeva za novi softverski nerazvijen program. Dijagram slučajeva prikazuje ponašanje sustava ili određenog razreda iz perspektive korisnika. Jednom navedeni slučajevi upotrebe mogu se označiti i kao tekstualni i vizualni prikaz. Ključni koncept modeliranja slučajeva je da pomaže u dizajniranju sustava iz perspektive krajnjeg korisnika. Dijagram slučajeva upotrebe obično je jednostavan. Ne prikazuje detalje slučajeva korištenja: već samo neke veze između slučajeva upotrebe, aktera i sustava.
Prije same predodžbe dijagramom slučajeva korištenja, potrebno je opisati zahtjev korisnika glavnim scenarijom i alternativnim scenarijem.

**Preduvjeti**

- Korisnik ima korisničko ime.
- Korisnik ima lozinku.
- Korisnik želi koristiti aplikaciju.
- Korisnik prethodno nije prijavljen u aplikaciju.
- Korisnik želi dodati događaje/ bilješke/ zadatke.
- Korisnik želi vidjeti događaje/ bilješke/ zadatke.

**Očekivani rezultat**

Klijent je prijavljen u aplikaciju i dobiva informacije o događaje/ bilješke/ zadatke ili ih upisiva.

**Glavni scenarij**

1. Korisnik unosi korisničko ime.
2. Korisnik unosi lozinku.
3. Korisnik naređuje sustavu da odradi prijavu.

4. Sustav provjerava lozinku u bazi podataka.

5. Sustav prijavljuje klijenta u aplikaciju.

6. Korisnik upisiva nove događaje.

7. Korisnik naređuje sustavu da prikaže njegove događaje.

**Alternativni scenarij**

5.1. Greška kod prijave.

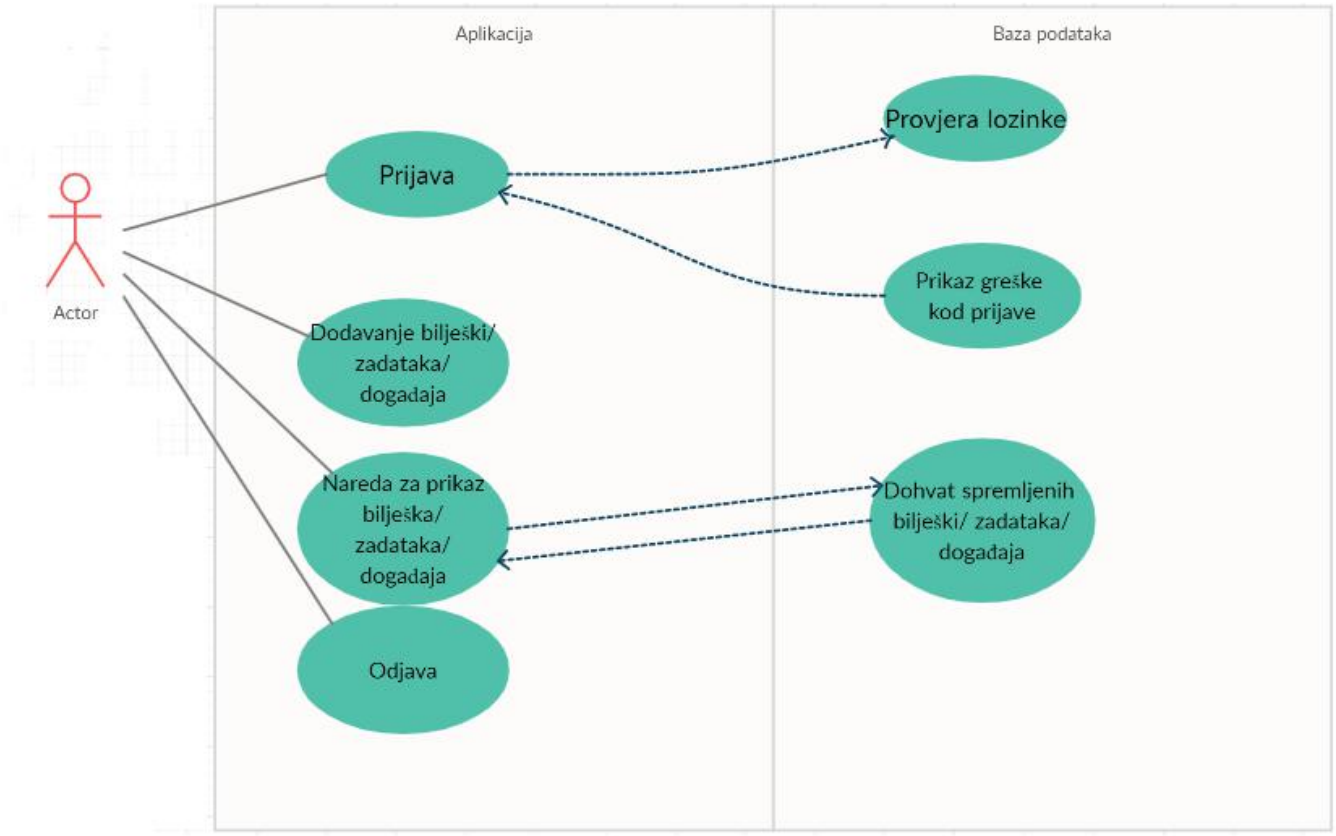
5.1.1. Sustav javlja klijentu da se dogodila greška.

6.1. Korisnik unosi zadatke.

6.2. Korisnik unosi bilješke.

7.1. Korisnik naređuje sustavu da prikaže njegove zadatke.

7.2. Korisnik naređuje sustavu da prikaže njegove bilješke.




**Slika 10. Slučajevi korištenja za aplikaciju Planer**

## 3.3. Funkcija sustava

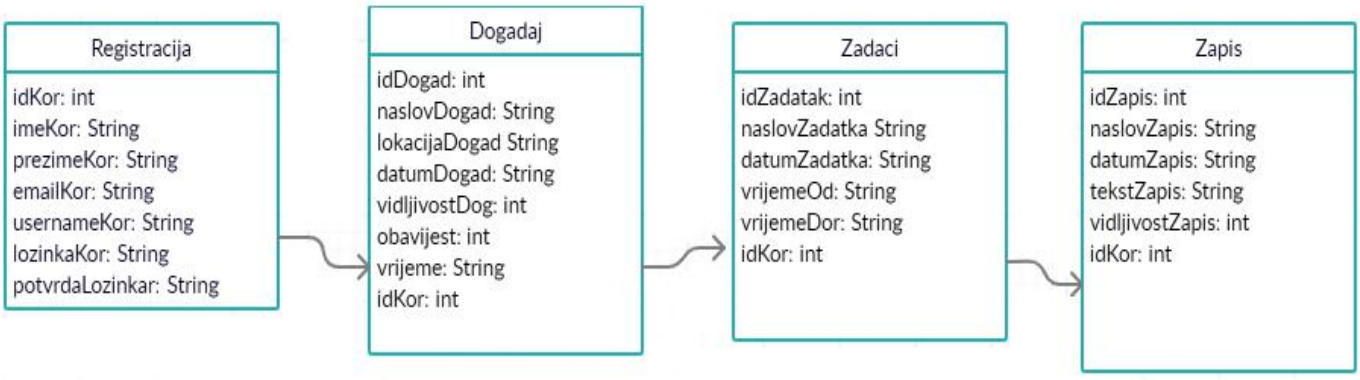
Glavne funkcije sustava je organizacija privatnih i poslovnih aktivnosti, jednostavnosti i vizualni način organizacije rada, olakšava stvaranje novih događaja, zadataka i bilješki te nudi upis aktivnosti: događaja, bilješka i zadataka i sami pregled upisanih događaja, zadataka i bilješki. Nastala je sa svrhom da olakša organizaciju dana korisnicima.

## 3.4. Izrada tablica

Za izradu aplikacije Planer korišten je NetBeans. NetBeans je integrirano razvojno okruženje (IDE) za Java. Omogućava razvoj aplikacija iz skupa modularnih softverskih komponenti koje se nazivaju moduli. NetBeans radi na sustavima Windows, MacOS, Linux i Solaris. Pored Java razvoja, a ima proširenja za druge jezike poput PHP, C, C ++, HTML5 i JavaScript.

Baza podataka aplikacije Planer se sastoji od 4 tablice koja svaka zasebno predstavlja jednu cjelinu, odnosno entitet u procesu razvoja proizvoda.

Naziv tablica koje su korištene su: Registracija, Dogadaj, Zapis i Zadaci (Slika 10.)




**Slika 11. Naziv tablica iz aplikacije Planer**

### 3.4.1. Tablica Registracija

Ova tablica služi za dva događaja: događaj prilikom prijave postojećeg korisnika ili prilikom stvaranja novog korisnika.

Primarni ključ ove tablice je „idKor“ i jednoznačno određuje svaki entitet u tablici. Atribute entiteta „Registracija“ možemo vidjeti u tablici:

**Tablica 1. Tablica Registracija**

| Ključ | Atribut | Tip podatka | Opis |
|---|---|---|---|
| *PRIMARY KEY* | idKor | int | ID korisnika |
| | imeKor | Varchar | Ime korisnika |
| | prezimeKor | Varchar | Prezime korisnika |
| | emailKor | Varchar | E-mail korisnika |
| | usernameKor | Varchar | Username korisnika |
| | lozinkaKor | Varchar | Lozinka korisnika |
| | potvrdaLozinka | Varchar | Potvrda Lozinke |

### 3.4.2. Tablica Dogadaj

Ova tablica služi za spremanje svih događaja korisnika na kojim korisnik želi prisustvovati. Primarni ključ ove tablice je „idDogad“ i jednoznačno određuje svaki entitet u tablici.

Vanjski ključ je „idKor“ koji je povezan s primarnim ključem tablice „Registracija“.

Atribute entiteta „Dogadaj“ možemo vidjeti u tablici:

**Tablica 2. Tablica Dogadaj**

| Ključ | Atribut | Tip podatka | Opis |
|---|---|---|---|
| *PRIMARY KEY* | idDogad | int | ID događaja |
| | naslovDogad | Varchar | Naslov događaja |
| | lokacijaDogad | Varchar | Lokacija događaja |
| | datumDogad | Varchar | Datum događaja |
| | vidljivostDog | int | Vidljivost |
| | obavijest | int | Obavijest |
| | vrijeme | Varchar | Vrijeme događaja |

| | | | |
|---|---|---|---|
| *FOREIGN KEY* | idKor | int | ID korisnika |

### 3.4.3. Tablica Zapis

Ova tablica služi za spremanje bilješka korisnika. Zamišljena je da služi kao dnevnik korisnika. Primarni ključ ove tablice je „idZapis“ i jednoznačno određuje svaki entitet u tablici. Vanjski ključ je „idKor“ koji je povezan s primarnim ključem tablice „Registracija“.

Atribute entiteta „Zapis“ možemo vidjeti u tablici:

**Tablica 3. Tablica Zapis**

| Ključ | Atribut | Tip podatka | Opis |
|---|---|---|---|
| *PRIMARY KEY* | idZapis | int | ID zapis |
| | naslovZapis | Varchar | Naslov zapisa |
| | datumZapis | Varchar | Datum zapis |
| | tekstZapis | Varchar | Tekst zapis |
| | vidljivostZapis | int | Vidljivost |
| *FOREIGN KEY* | idKor | int | ID korisnika |

### 3.4.4. Tablica Zadaci

Ova tablica služi za spremanje zadataka korisnika. Zamišljena je da služi kao “mjesto” na kojem će korisnik sebi zadati pojedine zadatke. Primarni ključ ove tablice je „idZapis“ i jednoznačno određuje svaki entitet u tablici. Vanjski ključ je „idKor“ koji je povezan s primarnim ključem tablice „Registracija“

Atribute entiteta „Zadaci“ možemo vidjeti u tablici.

**Tablica 4. Tablica Zadaci**

| Ključ | Atribut | Tip podatka | Opis |
|---|---|---|---|
| *PRIMARY KEY* | idZadatak | int | ID zadatka |
| | naslovZadatka | Varchar | Naslov zadatka |
| | datumZadatka | Varchar | Datum zadatka |

| | | | |
|---|---|---|---|
| | vrijemeOd | Varchar | Vrijeme zadatka |
| | vrijemeDo | Varchar | Istek zadatka |
| *FOREIGN KEY* | idKor | int | ID korisnika |

## 3.5. Dohvat podataka iz baze podataka

Da bi bilo moguće dohvatiti podatke i spremati ih potrebno je spojiti se na bazu podataka.

```
Connection conn=null;

conn=DriverManager.getConnection("jdbc:mySQL://localhost:3306/Planer?useTimez
  one=true&serverTimezone=UTC","root","");
```

public interface **Connection**

Connection uspostavlja vezu s određenom bazom podataka. Izvodi se SQL i rezultati se vraćaju u kontekstu veze. Connection pruža informacije koje opisuju njegove tablice, podržanu SQL gramatiku, pohranjene procedure, mogućnosti veze itd.

Nakon uspostave veze između razvojnog okruženja i baze podataka, potrebno je podatke dodati, ažurirati ih, čitati ih ili zbrisati.
Čitanje podataka iz bazu se izvršava tako da se pripremi SQL izraz i naredbom SELECT se čita se iz baze traženi podatak.

```
PreparedStatement pst=null;
String SQL ="SELECT * FROM `registracija` WHERE usernameKor=? and
lozinkaKor=?";
pst=conn.prepareStatement(SQL);
```

public interface **PreparedStatement**

Objekt koji predstavlja SQL izraz.

U ovom slučaju podaci koji se čitaju su korisničko ime i lozinka prilikom prijave. SELECT naredbom iščitavaju se iz tablice “Registracija” i uspoređivaju s podacima dobivenih od korisnika .

Spremanje podataka u bazu se izvršava tako da se naredbom INSERT unose podaci u bazu.

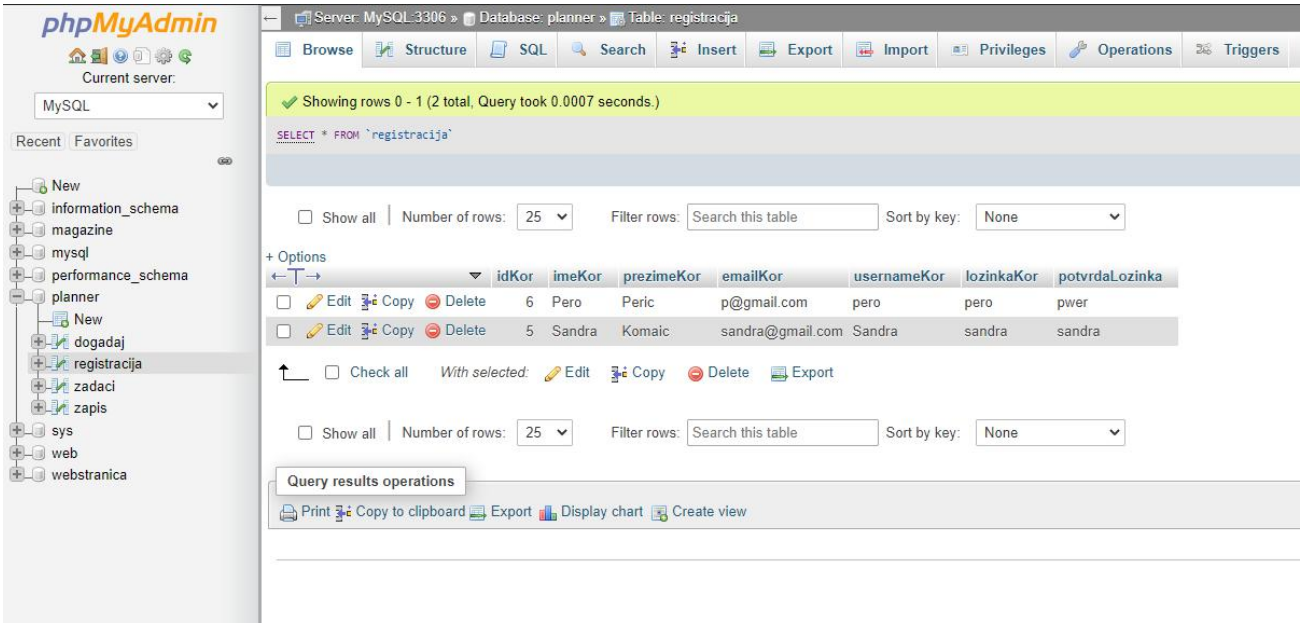
```
String query ="INSERT INTO `registracija`( `imeKor`, `prezimeKor`, `emailKor`, `usernameKor`, `lozinkaKor`, `potvrdaLozinka`) VALUES (?,?,?,?,?,?)";
```

Prilikom registracije korisnik unosi svoje ime, prezime, email, korisničko ime, lozinku i potvrdu lozinke. SQL naredbom INSERT INTO podaci se spremaju u bazu podataka u tablicu “Registracija”. Svaki podatak će biti spremljen u svom stupcu (ime u imeKor, prezime u prezimeKor...).

## 3.3. WAMP

WampServer se odnosi na skup rješenja za operativni sustav Microsoft Windows, a sastoji se od web poslužitelja Apache, OpenSSL za SSL podršku, MySQL baze podataka i PHP programskog jezika. Njegov je cilj pružiti otvoreni standard za meku razmjenu poruka u stvarnom vremenu između komponenata aplikacije i olakšati stvaranje labavo povezanih arhitektura temeljenih na mikroservisima. Zbog toga je prikladna poslovna sabirnica usluga, pogodna za razvoj web aplikacija ili za koordinaciju više povezanih uređaja u IoT-u.

Zbog njegove jednostavnosti i preglednosti svi podaci iz aplikacije Planer spremljeni su i dohvaćaju se s WampServer-a . (Slika 12.)




**Slika 12. Primjer tablice Dogadaj s Wampservera**

## 3.4. Aplikacija Planer

### 3.4.1. Prijava

Postojeći korisnik unosi svoje korisničko ime i lozinku, ako su korisničko ime i lozinka ispravni korisnik je već registriran i aplikacija ulazi na sljedeći ekran te je spremna za izvršavanje ostalih funkcija (unos događaja, zapisa ili bilješki). Podaci username i lozinka već su spremljeni u bazu podataka, u tablicu “Registracija”, te se na ekranu “Prijava”, čitaju, jednostavnim SELECT upitom iz tablice.

**Slika 13. Ekran Prijava**

### 3.4.2. Registracija

Ako korisnik nema račun pokreće novi ekran za registraciju novog korisnika. Korisnik unosi svoje podatke koji se zahtijevaju od njega: ime, prezime, email, korisničko ime, lozinku te na kraju potvrdu lozinke.

Ako su svi uneseni podaci ispravni, korisnik je tek registriran te se pokreće novi ekran, a podaci se spremaju za sljedeću prijavu korisnika.

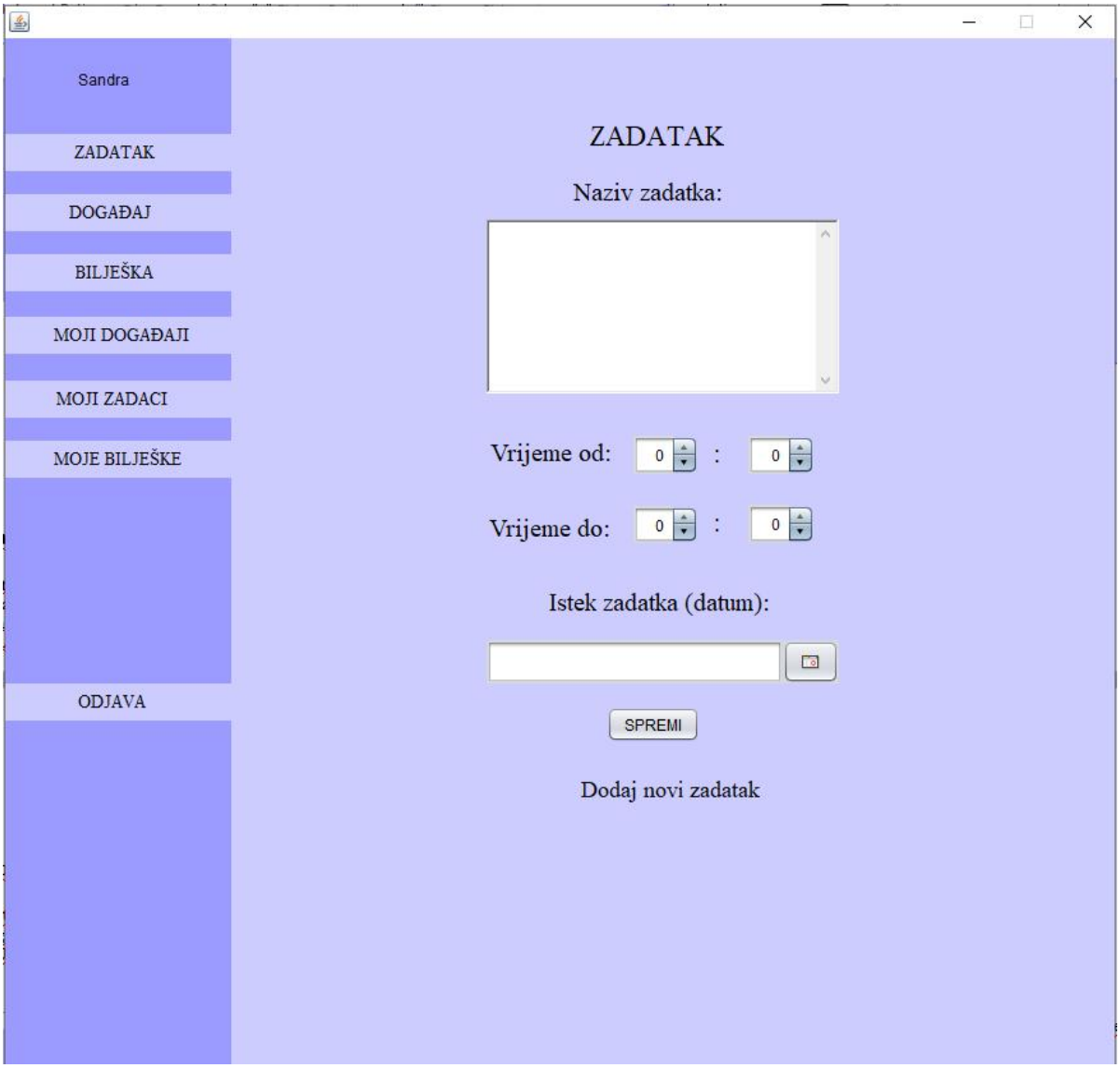
Nakon što se korisnik prijavi na aplikaciju. Pojavit će se početni ekran.

Kod registracije unosom podataka, podaci se automatski spremaju u tablicu "Registracija", INSERT naredbom te su ti podaci spremni kada se korisnik sljedeći put bude prijavio na aplikaciju. Sama funkcija baze podataka se može uočiti na ovom primjeru, naime korisnik samo jednom unosi svoje podatke te se ti podaci spremaju na dulje vrijeme i time se olakšava korištenje aplikacije samom korisniku.

**Slika 14. Ekran Registracija**

### 3.4.3. Ekran Zadatak

Ekran zadatak je početni ekran aplikacije. Sastoji se od tri komponente: podataka koje korisnik unosi za pojedini zadatak: naziv zadatka, vrijeme od početka izvršenja zadatka, pa sve do isteka zadatka. (na primjer u slučaju da je korisnik student koji želi učiti do određenog vremena, vrijeme mu služi kao timer. Ako je zadatak rok predaje moguće je odrediti i datum isteka), obavijesti o događaju u budućnosti i izbornik.
Kao i kod registracije, na ekranu Zadatak podaci koje unosi korisnik se spremaju u tablicu Zadaci.




**Slika 15. Ekran Zadatak**

### 3.4.3.1. Obavijest o događaju

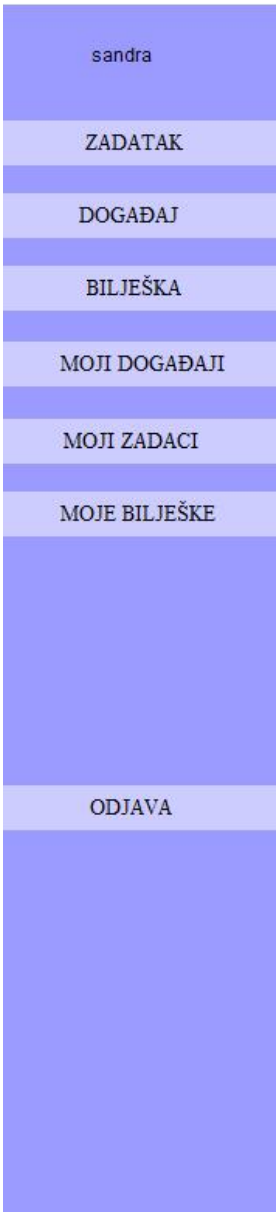
Korisniku se nakon prijave ili registracije otvara ekran Zadatak. U trenutku kad mu se otvori ekran pojavit će se obavijest koju je postavio pri unosu događaja u prošlost, prilikom unosa podatka o događajima na kojim želi sudjelovati u budućnosti.

**Slika 16. Obavijest o događajima**

### 3.4.3.1. Izbornik

Izbornik se sastoji od ekrana Zadatak, Događaj, Bilješka, Moji događaji, Moji zadaci, Moje bilješke i na kraju Odjava.

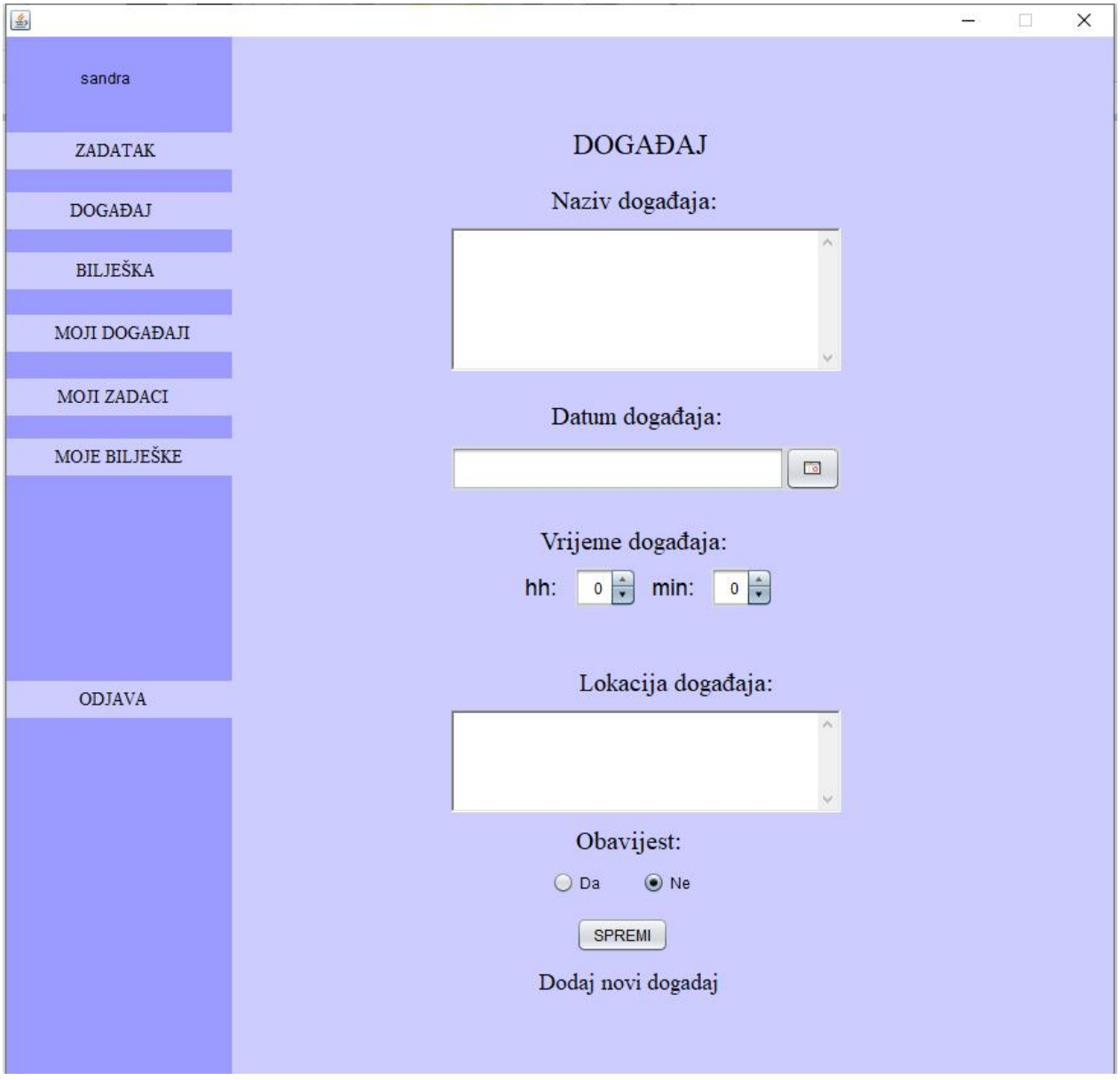
Klikom na bilo koju opciju iz izbornika korisnik pokreće novi ekran.




**Slika 17. Izbornik**

## 3.4.4. Ekran Događaj

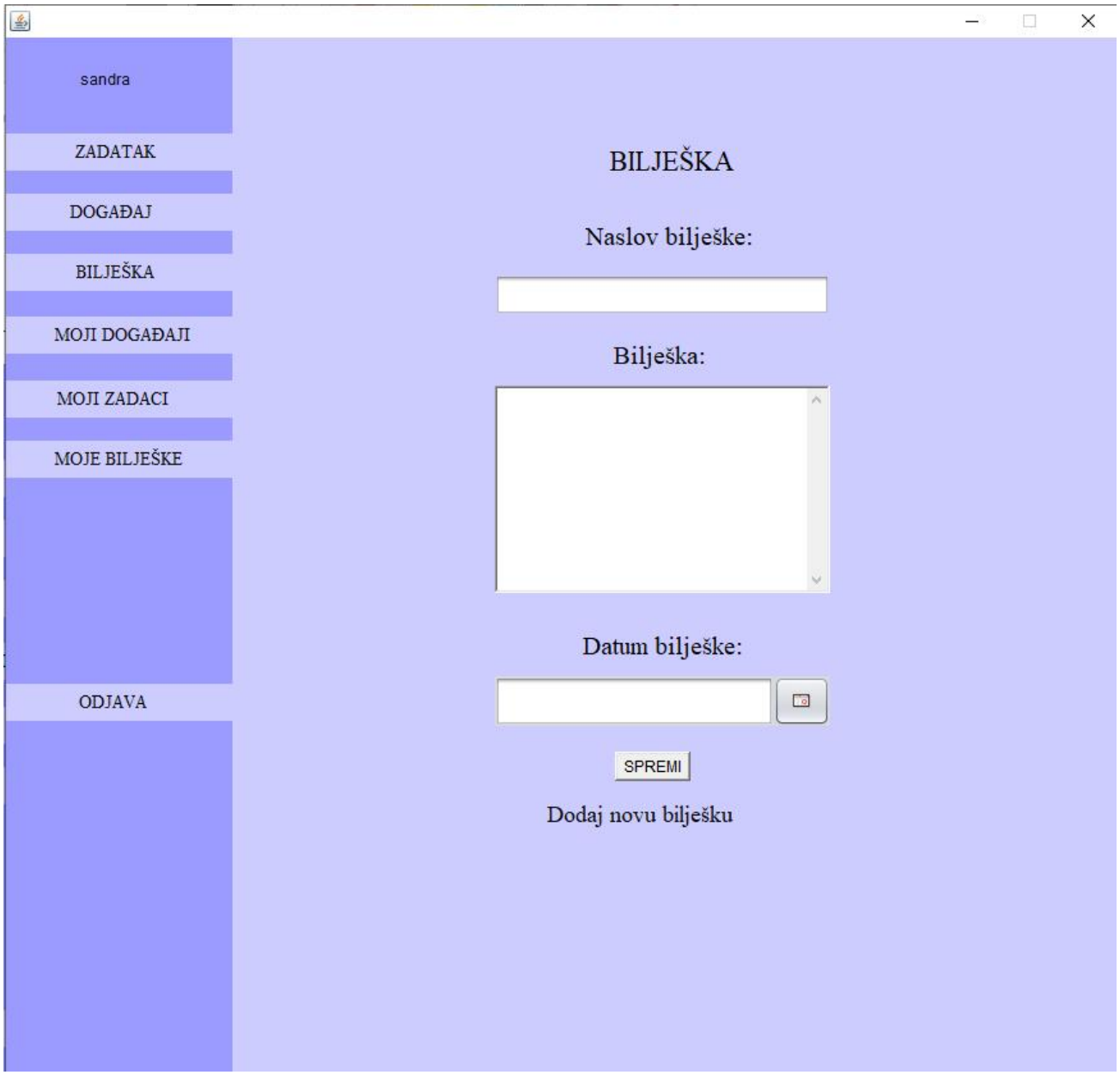
Ekran događaj koristi se za unos događaja na kojima korisnik želi sudjelovati. Uključuje naziv događaja, datum događaja, vrijeme početka događaja te lokacija događaja. Ako korisnik želi da ga aplikacija obavijesti da se bliži datum događaja može postaviti naredbu za obavijest. Podaci se prilikom unosa spremaju u tablicu „Dogadaj“ za buduću uporabu.




**Slika 18. Ekran događaj**

### 3.4.5. Ekran Bilješka

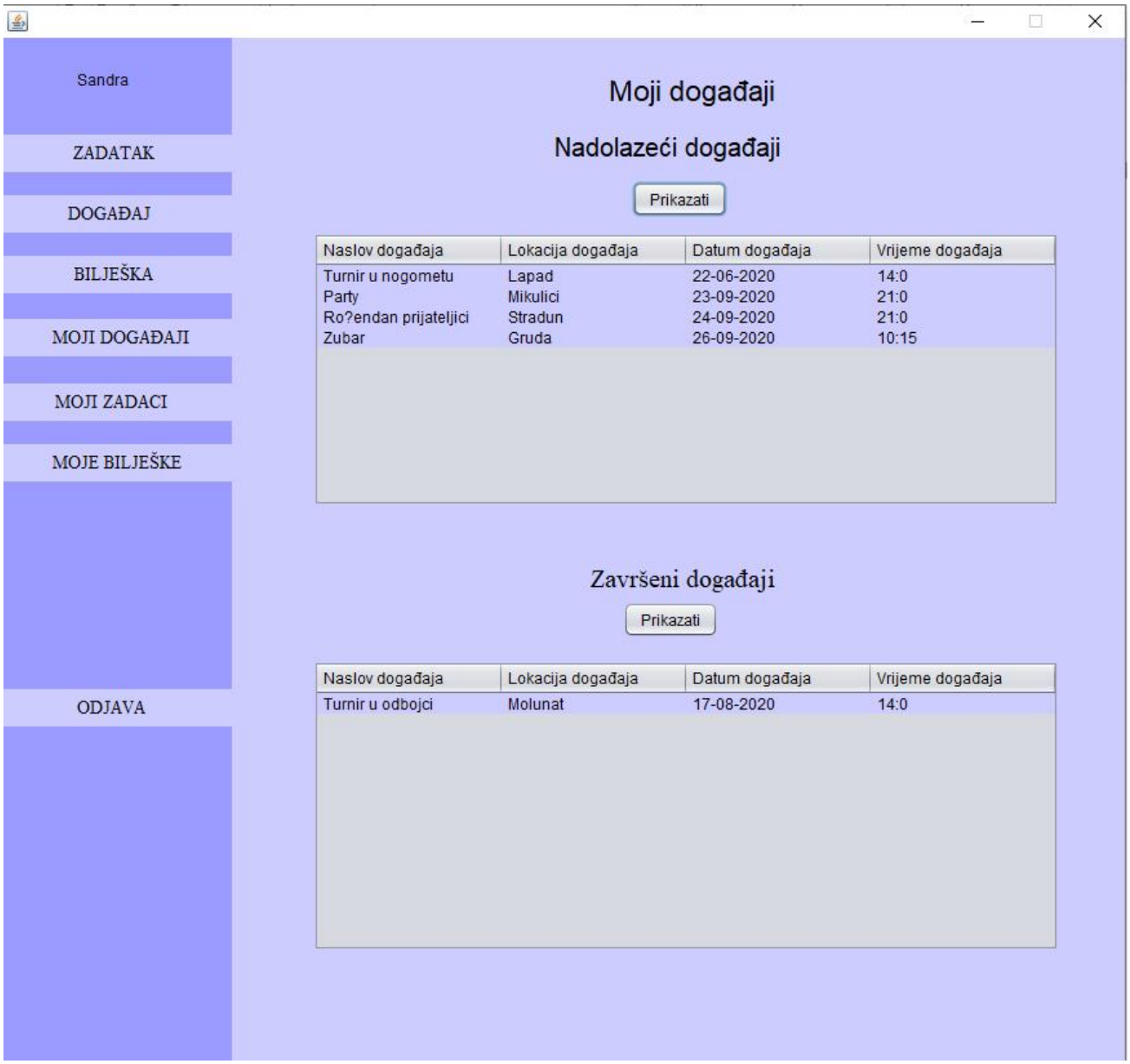
Ekran bilješka je zamišljena kao osobni dnevnik korisnika. U ovom ekranu omogućeno mu je unijeti naslov, zapis te bilješke i datum. Naravno bilješka se može koristiti i za druge namjene, na primjer popis namjernica za kupovinu. Podaci se prilikom unosa spremaju u tablicu „Zapisi“ za buduću uporabu.




**Slika 19. Ekran bilješka**

### 3.4.6. Ekran Moji događaji

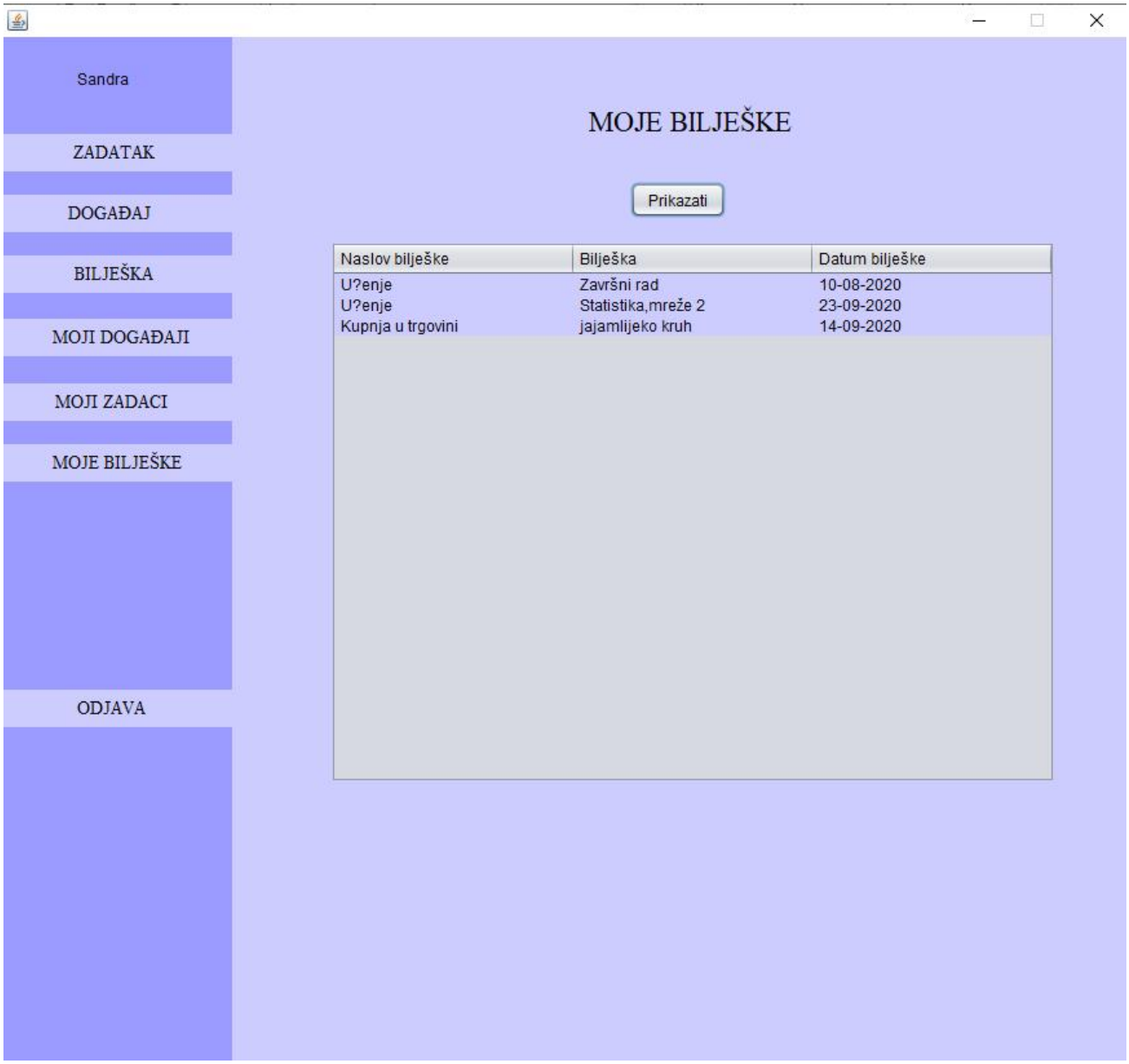
Ekran Moji događaji se sastoji od dviju tablica: tablica nadolazećih događaja i završenih događaja. Prva tablica su budući događaji korisnika, a druga tablica je događaji koji su završili. Oba dvije tablice se sastoje od podataka: naslova događaja, lokacije događaja, datuma događaja i vrijeme događaja. Ovdje ulogu ima tablica „Dogadaji“ iz baze podataka, jer se iz tablice „Dogadaji“ čitaju svi gore navedeni podaci. (3.4.5. Ekran događaj)




**Slika 20. Ekran Moji događaji**

### 3.4.7. Ekran Moje bilješke

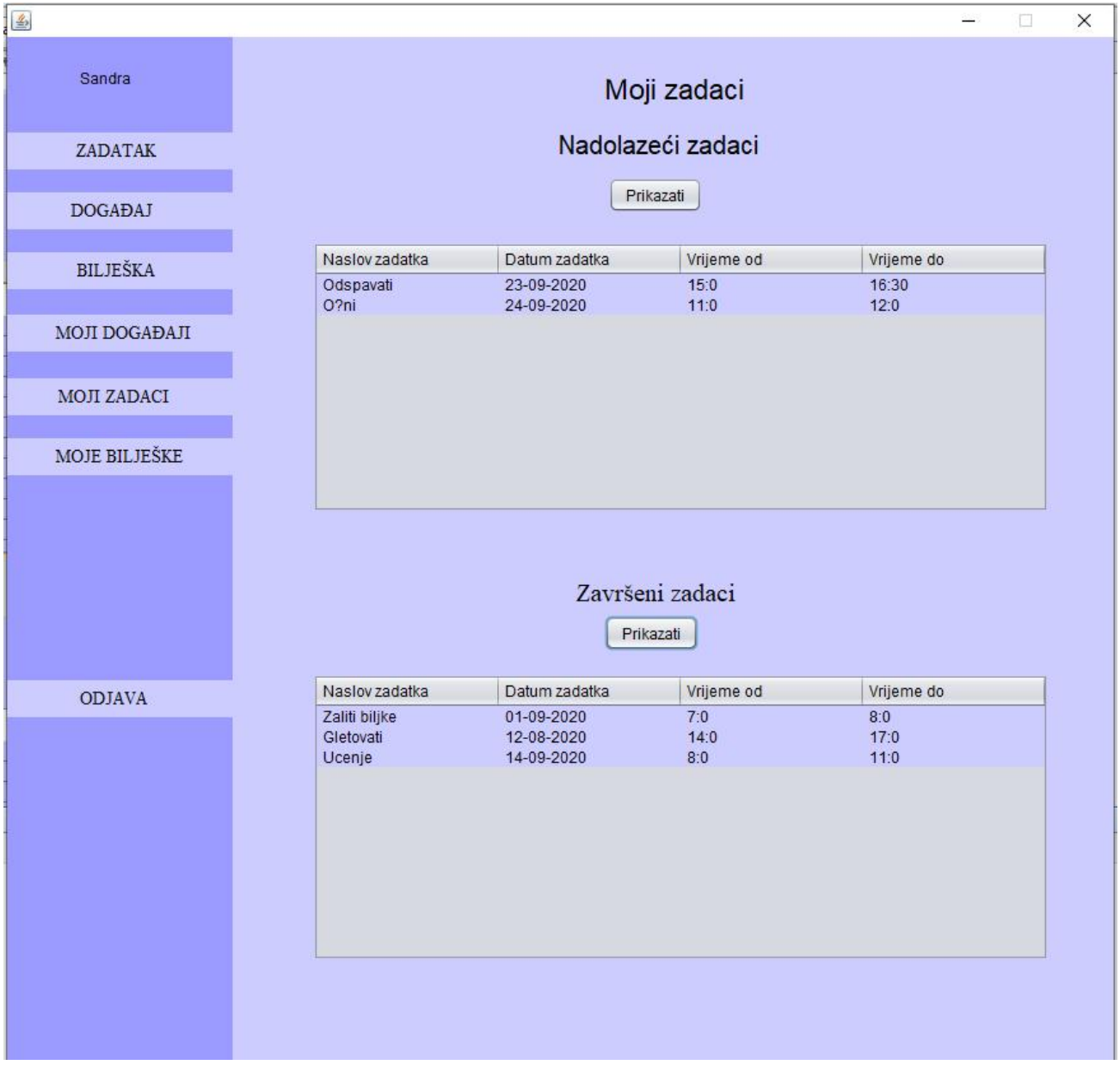
Ekran „Moje bilješke“ se sastoji od tablice. Tablica prikazuje sve bilješke koje je korisnik dodao: naslov bilješke, tekst bilješke i datum unosa bilješke. Svi podaci se iščitavaju iz tablice „Zapisi“ koja se nalazi u bazi podataka .




**Slika 21. Ekran Moje bilješke**

### 3.4.8. Ekran Moji zadaci

Ekran „Moji zadaci“ se sastoji od dviju tablica: tablica nadolazećih zadataka i završenih zadataka. Prva tablica su budući zadaci korisnika, a druga tablica je zadaci koji su završili. Oba dvije tablice se sastoje od podataka: naslov zadatka, lokacija zadatka, datum od zadanog zadatka, pa sve do isteka tog zadatka. Isto kao i kod ekrana „Moji događaji“ i „Moje bilješke“, podaci koji se već unaprijed spremljeni se iščitavaju jednostavnim SELECT upitom te se prikazuju na ekranu Moji zadaci.




**Slika 22. Ekran Moji zadaci**

# 4. Zaključak

Danas postoje mnoge tehnologije za rad sa spremljenim podacima, ali prije rada na projektima vrlo je važno izabrati pravu vrstu tehnologije. U većinom slučajeva baza podataka je vrlo korisna. Baza podataka je zbirka podataka organizirana tako da joj se lako može pristupati, upravljati i ažurirati je. Da bi baza podataka funkcionirala njome mora upravljati određeni sustav. Sustav koji upravlja bazom podataka naziva se sustav za upravljanjem bazom podataka ili DBMS. Funkcija DBMS-a je ta da DBMS komunicira s krajnjim korisnicima, aplikacijama i samom bazom podataka za analizu podataka tako da izvlači podatke iz baze podataka kao odgovor na upite, a upiti su glavni način na koji korisnici dohvaćaju podatke iz baze podataka.

Na koji način baza funkcionira prikazano je pomoću aplikacije Planer. Koja nudi mogućnost organizacije ljudskih aktivnosti tako da su svi događaji, zadaci i bilješke organizirani na jednom mjestu, jednostavni i pregledni. Svrha baze podataka je spremanje tih podataka, ažuriranje i brisanje. Ti podaci se spremaju u relacijsku bazu podataka i organizirani su u retke, stupce i tablice koji se indeksiraju da bi se olakšalo pronalaženje relevantnih podataka putem SQL upita.

# 5. Literatura


[1]"What is Database? What is SQL?", *Guru99.com*, 2020. [Online]. Available: https://www.guru99.com/introduction-to-database-sql.html. [Accessed: 25- Sep- 2020].

[2]"DBMS languages", *beginnersbook.com*, 2020. [Online]. Available: https://beginnersbook.com/2015/04/dbms-languages/#:~:text=Database%20languages%20are%20used%20to,SQL%20(Structured%20Query%20Language). [Accessed: 24- Sep- 2020].

[3]"Concepts of Database Architecture", *Medium*, 2017. [Online]. Available: https://medium.com/oceanize-geeks/concepts-of-database-architecture-dfdc558a93e4#:~:text=Database%20architecture%20focuses%20on%20the,DBMS%20depends%20on%20its%20architecture. [Accessed: 24- Sep- 2020].

[4]"DBMS Database Languages", *W3schools.in*, 2020. [Online]. Available: https://www.w3schools.in/dbms/database-languages/. [Accessed: 24- Sep- 2020].

[5]"Database Models in DBMS | Studytonight", *Studytonight.com*, 2020. [Online]. Available: https://www.studytonight.com/dbms/database-model.php. [Accessed: 24- Sep- 2020].

# 6. Prilozi

## 6.1. Prilozi slike

Slika 1. Vrste DBMS jezika [2]..... 7
Slika 2. Jednoslojna arhitektura [3]..... 8
Slika 3. Dvoslojna arhitektura [3]..... 9
Slika 4. Dvoslojna arhitektura [3]..... 11
Slika 5. Modeli baze podataka [1]..... 11
Slika 6. Hijerarhijski model baze podataka [5]..... 12
Slika 7. Mrežni model baze podataka[5]..... 13
Slika 8. Relacijski model baze podataka [5]..... 14
Slika 9. Objektno orijentirani model baze podataka [5]..... 15
Slika 11. Naziv tablica iz aplikacije Planer..... 19
Slika 12. Primjer tablice Dogadaj s Wampservera..... 24
Slika 13. Ekran Prijava..... 25
Slika 14. Ekran Registracija..... 26
Slika 15. Ekran Zadatak..... 27
Slika 16. Obavijest o događajima..... 28
Slika 17. Izbornik..... 29
Slika 18. Ekran događaj..... 30
Slika 19. Ekran bilješka..... 31
Slika 20. Ekran Moji događaji..... 32
Slika 21. Ekran Moje bilješke..... 33
Slika 22. Ekran Moji zadaci..... 34

## 6.2. Prilozi tablica

Tablica 1. Tablica Registracija..... 20
Tablica 2. Tablica Dogadaj..... 21
Tablica 3. Tablica Zapis..... 21
Tablica 4. Tablica Zadaci..... 22

# IZJAVA

Izjavljujem pod punom moralnom odgovornošću da sam seminarski / završni / diplomski rad izradio/la samostalno, isključivo znanjem stečenim na studijima Sveučilišta u Dubrovniku, služeći se navedenim izvorima podataka i uz stručno vodstvo mentora izv.dr.sc Mario Miličević, kome/kojoj se još jednom srdačno zahvaljujem.

Sandra Komaić