# Oblikovanje web sustava za upravljanje podatcima

**Podbojec, Vedran**

Sveučilište
Sjever

*Završni rad br. 377/EL/2016*

# Oblikovanje *web*-sustava za upravljanje podatcima

**Vedran Podbojec, 0231049973**

Varaždin, rujan 2016. godine

Sveučilište
Sjever

**Odjel za elektrotehniku**

**Završni rad br. 377/EL/2016**

# Oblikovanje *web*-sustava za upravljanje podatcima


**Student**

Vedran Podbojec, 0231049973

**Mentor**

dr.sc. Ladislav Havaš, dipl. ing.



Varaždin, rujan 2016. godine

**Sveučilište Sjever**
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

# Prijava završnog rada

## Definiranje teme završnog rada i povjerenstva

| | |
|---|---|
| ODJEL | Odjel za elektrotehniku |
| PRISTUPNIK | Vedran Podbojec |
| MATIČNI BROJ | 0269/601 |
| DATUM | 14.07.2016. |
| KOLEGIJ | Baze podataka i SQL |
| NASLOV RADA | Oblikovanje web sustava za upravljanje podatcima |
| NASLOV RADA NA ENGL. JEZIKU | Designing the web system for data management |
| MENTOR | dr. sc. Ladislav Havaš, dipl. ing. |
| ZVANJE | viši predavač |

ČLANOVI POVJERENSTVA

1. mr. sc. Ivan Šumiga
2. mr. sc. Matija Mikac
3. dr. sc. Ladislav Havaš
4. 
5. 

## Zadatak završnog rada

BROJ 377/EL/2016

OPIS

Cilj ovog završnog rada je oblikovati web sustav za upravljanje podatcima (CMS) u TV-kućama. Potrebno je kreirati sučelja za unos, izmjenu i pretragu video materijala (arhiva) te digitalizacija procesa izrade knjige najava i digitalizacije procesa izrade knjige snimanja. Potrebno je omogućiti korisnicima sustava kreiranje različitih SQL upita u cilju pretraživanja video materijala (arhive), knjige najava te knjige snimanja.

U radu je još potrebno:

- Opisati korištenu programsku platformu za brzi razvoj web aplikacija "Weavemaker".
- Ukratko opisati programske platforme MySQL, Java, Apache Tomcat.
- Kreirati odgovarajući konceptualni ER model informacijskog sustava, te transformirati ER model u relacijski model podataka.
- Kreirati korisnička sučelja.
- Korištenjem parametarskih SQL upita unutar Wavemaker-a realizirati specifične prikaze podataka.
- Kreirati potrebne metode (Java) za slanje obavijesti svim sudionicima snimanja putem e-mail poruka.
- Omogućiti prikaz željenih statističkih podataka (broj snimanja, radni sati zaposlenika).
- Osigurati redovito arhiviranje baze podataka.
- Proanalizirati razvijeni prototip sustava, navesti njegove prednosti i mane, navesti mogućnosti primjene u drugim sličnim okruženjima, te mogućnosti nadogradnje ili povezivanja s drugim sustavima.
- Sukladno navedenom, u tekstualnom dijelu završnog rada treba kreirati odgovarajući sadržaj te napisati cjeloviti tekst završnog rada.

ZADATAK URUČEN 22.8.2016.

POTPIS MENTORA

# Predgovor

Povijest sustava za upravljanje podatcima (engl. *Content management system - CMS*) seže u vrijeme prvih otkrića, odnosno stjecanja prvih znanja ljudskog roda. Kako bi se što efikasnije upravljalo podatcima, razvijale su se posebne tehnike organizacije i pohranjivanja podataka. Početke toga nalazimo u prvim svitcima, knjigama i zbirkama, a nešto kasnije u razvoju knjižnica i arhiva [1].

Kao što je poznato količina podataka u današnje vrijeme, odnosno informacija kojima tvrtke svakodnevno raspolažu, imaju eksponencijalni rast. Nužnost pravovremene dostupnosti podataka koji su relevantni za rad tvrtke danas ima sve veći utjecaj na njihovu produktivnost. U većini slučajeva ti podatci pohranjeni su u registrima, na čvrstim medijima (*CD, DVD, USB, magnetske trake, itd.*), negdje u oblaku (*GoogleDrive, OneDrive, AmazonDrive itd.*), tj. pohranjeni su na više vrsta medija koji mogu biti pohranjeni na više lokacija [2].

Kako bi se korisnicima omogućio brzi pristup podatcima, odnosno kako bi se kreiralo korisničko sučelje u kojem je jednostavno unositi, mijenjati, te pretraživati podatke, a da ujedno od korisnika ne zahtijevamo napredno poznavanje informacijske i komunikacijske tehnologije, razvijeni su alati koji omogućavaju sve gore spomenuto. Izrada jednog takvog sustava tema je ovog završnog rada.

Povijesni razvoj CMS-a može se podijeliti u tri dijela: upravljanje – M (*Management*), sustavi – S (*System*) i sadržaj – C (*Content*). Možemo reći da se razvoj kretao upravo navedenim redoslijedom, od jednostavnijih sustava, pa sve do kompleksnijih sustava s mnogobrojnim svojstvima i mogućnostima. U početnim su fazama sustavi bili orijentirani prema dinamičkoj promjeni podataka *web*-sadržaja koji su do tada bili statični. Nakon toga koncept se usavršavao i krenulo se prema upravljanju, a tek kada su se razvila ta dva područja, krenulo se prema razvoju upravljanja sadržajem. S odmakom vremena industrija se podijelila u dvije grupe koje i danas koegzistiraju, a to su sustavi za upravljanje dokumentima (engl. *Enterprise Document Management System - EDMS*) i sustavi za upravljanje *web*-sadržajem (engl. *Web Content Management System - WCMS*). Možemo krenuti u daljnju podjelu sustava kao što su sustavi otvorenog ili zatvorenog koda, komercijalni, poslovni i drugi, no svi oni zadržavaju osnovnu funkcionalnost, odnosno skupljanje podataka, pohranu informacija, organizaciju informacija, izračun, komunikaciju, prezentaciju podataka, kontrolu unosa, mijenjanja te prikaza podataka [3].

U izradi aplikacije koja je tema ovog završnog rada pomogli su Dunja Apostolovski, Saša Vukšić, te Matija Tomašković kojima ovim putem zahvaljujem na uloženom trudu. Nadalje htio bih zahvaliti svim profesorima kroz visokoškolsko obrazovanje, a posebno svojem mentoru dr.sc. Ladislavu Havašu koji me vodio kroz sve faze nastajanja ovog rada pomažući svojim savjetima, primjedbama i uklanjanjem nedostataka u cilju što kvalitetnije obrade teme.

# Sažetak

U ovom je radu opisana izrada sustava za upravljanje podatcima u televizijskoj kući kroz radne procese organizacije najave snimanja, izrade dnevne knjige snimanja, arhive i prikaza obavijesti.

Izrada aplikacije izvršena je u razvojnoj okolini WaveMaker [4]. Za pretvorbu konceptualnog modela baze podataka (BP) u relacijski model BP korišten je alat MySQL Workbench [5], dok je baza podataka MySQL [6] (više o razvojnim alatima vidjeti u 4. poglavlju ovog rada).

Fokus završnog rada stavljen je na segment razvijanja konceptualnog modela baze podataka. S njime u vezi objašnjeni su osnovni pojmovi koji se pojavljuju pri izradi konceptualnog modela BP i pretvaranja navedenog modela u relacijski model BP. Kada govorimo o pojmovima, objašnjen je pojam samog konceptualnog modela (engl. *Entity-relationship model – ER model*). Pojedinačno su objašnjeni pojmovi entiteta, atributa, i ključeva (primarni, vanjski). Opisane su moguće veze između relacija (tablica) i prikazan je jedan od načina grafičke notacije samih veza (Peter Chen). Prikazano je na koji se način može vizualizirati konceptualni model, uz neizostavna pravila kojih se nužno pridržavati kako bismo konceptualni model BP u konačnici pretvorili u relacijski model BP.

Koz rad je razvijen konceptualni model BP u segmentu arhive. Nabrojeni su svi identificirani atributi, te pripadajući entiteti. Prikazano je i objašnjeno na koji način dolazimo do svih obrađenih veza između relacija. Pridržavajući se pravila pretvorbe konceptualnog modela u relacijski izrađena je relacijska s pomoću alata MySQL Workbench.

Sadržaj je upotpunjen osnovnim informacijama o alatima koji su korišteni pri izradi BP i Java *web*-aplikacije gdje je posebni naglasak stavljen na razvojnu okolinu WaveMaker Studio Desktop. Neizostavan dio je i upoznavanje čitatelja s Java poslužiteljem Apache Tomcat [7] na kojem će se u konačnici aplikacija pogoniti.

Završno su definirane i izrađene *web*-stranice za unos najava, pregled najava, unos snimanja, pregled i pretragu snimanja, obavijesti te stranica za prikaz statističkih podataka. Posebna pozornost posvećena je segmentima koji su razvijeni (Java programski kod), odnosno modulima aplikacije koji nisu sadržani u razvojnoj okolini. Prikazani su SQL upiti, te na koji se način „spajaju“ (engl. *binding*) parametarski SQL upiti s elementima *web*-stranice u razvojnoj okolini.

Na kraju je napravljena sama analiza izrađene *web*-aplikacije pri kojoj se utvrdilo nekoliko nedostataka. Prilikom korištenja razvijene *web*-aplikacije primijećeni su problemi u prikazu određenih segmenata *web*-aplikacije. Također, prijevod stranice, odnosno lokalizacija nije radila. Korištenjem Google Chrome *web*-preglednika riješen je problem prikaza segmenata stranice dok problem lokalizacije, odnosno prijevoda nije u potpunosti riješen. Zaključno se može reći da je aplikacija potpuno funkcionalna i primjenjiva u radnim okolinama kao što je rastuća kreativna industrija, odnosno u radnim okolinama kojima razvijeni konceptualni model baze podataka odgovara modelu opisanom u ovom radu.

# Popis korištenih kratica

| | |
|---|---|
| **CMS** | Sustavi za upravljanje sadržajem (*Content management system*) |
| **EDMS** | Sustavi za upravljanje dokumentima (*Enterprise Document Management System*) |
| **WCMS** | Sustavi za upravljanje *web*-sadržajem (*Web Content Management System*) |
| **BP** | Baza podataka |
| **NAS** | Mrežno mjesto za pohranu podataka (*Network-attached storage*) |
| **LAN** | Lokalna mreža (*Local area network*) |
| **ER model** | Model entiteta i veza (*Entity-relationship model*) |
| **CASE** | Računalno potpomognuti softverski inženjerski alati (*Computer-aided software engineering tools*) |
| **1NF** | Prva normalna forma |
| **2NF** | Druga normalna forma |
| **3NF** | Treća normalna forma |
| **4NF** | Četvrta normalna forma |
| **INT** | Integer (cijeli broj) |
| **RAD** | Razvojno okruženje za brzi razvoj aplikacija (Rapid application *development*) |
| **IDE** | Integrirano razvojno okruženje (*Integrated development environment*) |
| **WYSIWYG** | What You See is What You Get |
| **SOA** | Uslugama orijentirana arhitektura (*Service Oriented Architecture*) |
| **ORM** | Objektno/Relacijsko mapiranje (*Object/Relational Mapping*) |
| **XML** | Programski jezik za označavanje podataka (*Extensible Markup Language*) |
| **SUBP** | Sustav za upravljanje bazom podataka |
| **DDL** | Jezik za definiciju ili deklaraciju objekata u bazi podataka (*Data description language*) |
| **DML** | Jezik za manipulaciju objektima baze podataka (*Data manipulation language*) |
| **DCL** | Jezik za postavljanje dozvola u bazi podataka (*Data control language*) |
| **TCL** | Jezik za upravljanje transakcija (*Transaction control language*) |
| **SQL** | Računalni jezik za izradu, traženje, ažuriranje i brisanje podataka iz relacijskih baza podataka (*Structured Query Language*) |
| **JVM** | Java virtualni stroj (JAVA *virtual machine*) |
| **ADF** | Okruženje za razvoj aplikacija (*Application Development Framework*) |

# Sadržaj

# 1. Uvod

Prije početka izrade same aplikacije bilo je potrebno sagledati organizacijsku shemu tvrtke, te shvatiti na koji se način odvijaju, odnosno koje su posebnosti radnih procesa (engl. *know how*), a sve u cilju kako bi se kreirala odgovarajuća struktura baze podataka. Kroz razgovor s korisnicima uočeni su problemi koji su se pojavljivali u svakodnevnom radu tvrtke.

Dosadašnja praksa u segmentu organizacije najave snimanja te same knjige snimanja odvijala se na način da je postojala bilježnica („Knjiga najava“) u koju su se upisivali događaji te tablica u elektronskom obliku (Excel) u koju su se zapisivala snimanja događaja na određeni dan s pripadajućim dodatnim podatcima (snimatelj, mjesto događaja, vrijeme događaja itd.).

Kao što se može naslutiti, takav je sustav imao mnoge nedostatke. Sama preglednost najavljenih događaja bila je ograničena. Potrebni podatci držali su se na jednom mjestu i tako onemogućavali istovremenu dostupnost podataka svim zaposlenicima. Kod planiranja ljudskih resursa često su se događale situacije nedovoljnog ili prekobrojnog angažiranja zaposlenika. Ekstremno su se događale situacije izostavljanja nekih događaja iz dnevnih snimanja iz razloga previda u knjizi najava.

U segmentu arhiviranja videomaterijala nije postojala nikakva evidencija, što je za posljedicu imalo nemogućnost pretraživanja videomaterijala po bilo kojem kriteriju. Materijali su se kopirali na sustave za pohranu podataka u direktorije strukturirane prema teritorijalnom ustrojstvu Republike Hrvatske (županija, grad ili općina), te segmentirani prema tematskim kategorijama (školstvo, zdravstvo, gospodarstvo, poljoprivreda, politika itd.)

Također nije bilo vođenja statistike o ukupnim snimanjima na razini zadanog perioda (dan, tjedan, mjesec, godina), niti evidencija efektivnog rada zaposlenika, također po zadanom periodu.

S ciljem poboljšanja radnih procesa pronađena su zadovoljavajuča rješenja opisana u nastavku.

U procesu kreiranja „Knjige najava“ nužno je omogućiti korisnicima upis događaja sa svojeg radnog mjesta, koristeći se *web*-preglednikom. Uz naziv samog događaja potrebno je omogućiti

unos mjesta, datuma, vremena, grada i županije u kojem se događaj odvija, napomene, te dodati mogućnost spajanja privitka (tekstualna datoteka, slika itd.) uz određeni događaj, odnosno najavu snimanja (više u poglavlju 5.3). Korisnicima je nužno osigurati mogućnost mijenjanja te brisanja upisanih podataka ovisno o korisničkom pravu. Sve je potrebno vizualizirati u obliku *web*-obrasca pri čemu se odabir županije, grada, datuma i vremena vrši s pomoću padajućih izbornika. U segmentu „Knjige najava“ kreirala se stranica u kojoj se vrši pretraga svih najavljenih događaja po datumu odvijanja događaja.

Kako se „Knjiga snimanja“ kreira iz „Knjige najava“, nametnula se ideja da se iz prozora u kojem se pregledavaju najavljeni događaji, isti mogu preseliti jednim klikom miša u „Knjigu snimanja“, bez potrebnih dodatnih upisivanja podataka koji već postoje za navedeni događaj u „Knjizi najava“. „Knjiga snimanja“ za razliku od „Knjige najava“ sadrži podatke o osobama koje su uključene u samo snimanje (snimatelj, tonski snimatelj, novinar), predviđenom vremenu trajanja snimanja iz kojeg se kreiraju vremena polaska na snimanje, odnosno dolaska sa snimanja, te podatak za koju se emisiju odrađuje snimanje. S obzirom na to, kreirao se *web*-obrazac koji sadržava potrebna polja (više u poglavlju 5.4). Podatci sadržani u „Knjizi najava“ automatski su ispunili potrebna polja u *web*-obrascu „Knjige snimanja“, a ostali su se podatci odabrali iz izbornika. Sve manipulacije podatcima odvijale su se kroz *web*-preglednik. Napomenimo još da se mogućnost brisanja te promjena podataka u knjizi snimanja određuje korisničkim pravima. Dodatna funkcionalnost dobila  se kroz modul koji elektronskom poštom obavještava sudionike o snimanjima na koja su raspoređeni. Kako bi se omogućilo pretraživanje „Knjige snimanja“, dodan je prozor za pretragu, uz mogućnost pretraživanja po zadanom periodu ili osobama koje su bile raspoređene na snimanje.

Zadani statistički parametri vizualizirani su s pomoću grafikona na posebnoj stranici gdje je prikazan ukupan broj snimanja za određeni vremenski period. Vremenski period odabire se iz padajućih izbornika koji se nalaze u prozoru. Također je prikazana godišnja statistika broja snimanja s pomoću trend-grafikona (više u poglavlju 5.6) .

Kako sustav „Arhive“, odnosno pisani trag o arhiviranom videomaterijalu nije postojao, no postojala je logička struktura direktorija na mrežnim mjestima za pohranu (engl. *Network attached storage, NAS*), upravo je ona odabrana kao polazište za razvoj konceptualnog modela baze podataka. Zadržalo se segmentiranje videomaterijala po geografskom ustroju Republike Hrvatske (država-županija-grad), s tim da su se dodali podatci koji još detaljnije obilježavaju videosnimku. Uz podatak o samom nazivu videomaterijala, dodan je podatak o datumu nastanka

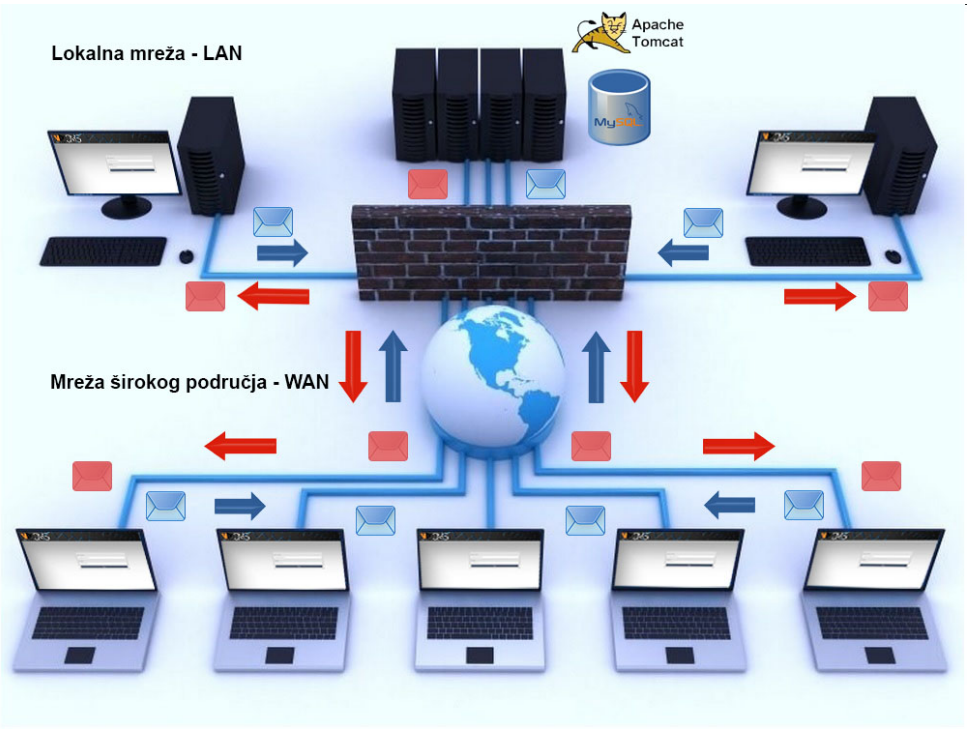
snimke, opisu, ključnoj riječi, putanji do videomaterijala, formatu slike, tipu datoteke, rezoluciji snimke te kategoriji (više u poglavlju 3.).

Razvojem grafičkog sučelja same aplikacije odredili smo dio početnog prozora za postavljanje informacija bitnih za korisnike aplikacije. Dodavanje obavijesti ograničeno je na korisnike s odgovarajućim pravima.

Korisnička prava podijeljena su u nekoliko skupina:

- *Admin* pravo korisnicima daje potpunu kontrolu nad informacijama spremljenim u bazi podataka kroz aplikaciju.
- *SuperSuperUser* pravo korisnicima daje mogućnost unosa, mijenjanja i brisanja događaja iz „Knjige najava“, i „Knjige snimanja“, pregledavanje „Arhive“, te unos, mijenjanje i brisanje obavijesti.
- *SuperUser* pravo korisnicima omogućava unos, mijenjanje i brisanje u „Knjizi snimanja“ dok u ostalom dijelu aplikacije mogu samo pregledavati podatke.
- Korisnicima s *User* pravom omogućen je isključivo pregled sadržaja.

Slika 1.1 prikazuje strukturu sustava, odnosno prikazuje put informacija od korisnika do poslužitelja i obrnuto. Vidljivo je da je sustav dostupan van lokalne mreže (LAN).



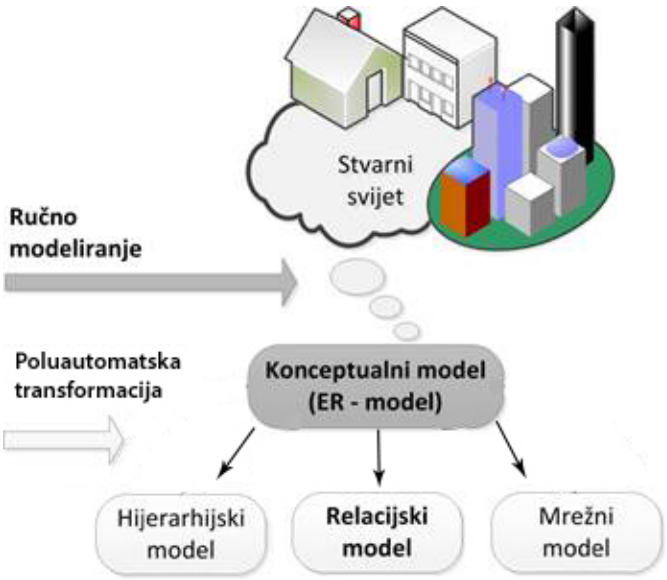

*Slika 1.1: Prikaz strukture sustava te strujanja podataka*

Nakon utvrđivanja polazišnih kriterija za izradu baze podataka, kao što su identifikacija entiteta te pripadajućih atributa, utvrdile su se veze između odgovarajućih entiteta, a sve s ciljem kreiranja optimalne BP, tj. da smanjimo potrebu za modifikacijom BP kada je ona već fizički kreirana. Slijedom navedenog krenuo je razvoj konceptualnog modela BP.

# 2. Model entiteta i veza

Kako bismo uspješno vizualizirali BP, nužno je izraditi detaljan logički prikaz, odnosno logički model BP. Taj logički prikaz BP zovemo model entiteta i veza. Idejni začetnik modela entiteta i veza poznati je kineski stručnjak na području elektrotehnike i računarstva dr. sc. Pin-Shan (Peter) Chen [8]. 1968. godine diplomirao je elektrotehniku u njegovom rodnom gradu Taiwanu, dok je nekoliko godina kasnije (1973.) doktorirao računarstvo i matematiku na Harvardu. Godine 1976. u specijalnom izdanju uglednog časopisa „ACM Transactions on Database Systems“ - TODS, objavljen je njegov članak „The entity-relationship model - toward a unified view of data“, pa tu godinu uzimamo kao nastanak ER modela BP. ER modeli su danas temelj računalno potpomognutih softverskih inženjerskih (engl. *Computer-aided software engineering* - *CASE*) alata i metodologija za sistemsku analizu i dizajn.

ER model je vizualni prikaz različitih podataka s pomoću pravila koja opisuju kako su ti podatci međusobno povezani. Osnovno svojstvo ER modela je dijagramska tehnika kojom se objekti podataka predstavljaju vizualno, a krajnji je cilj stvoriti vjerni prikaz realnosti u bazi podataka. ER model je dovoljno jednostavan da ga ljudi različitih struka mogu razumjeti te stoga služi za komunikaciju projektanta baze podataka i budućih korisnika i to u najranijoj fazi razvoja baze [9].



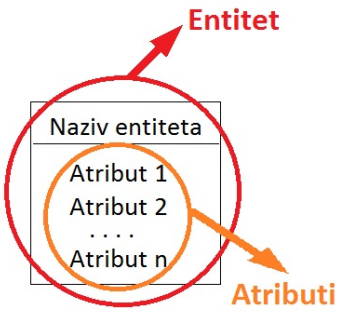

*Slika 2.1: Postupak modeliranja baze podataka*

Prilikom modeliranja entiteta i veza objekti iz realnog svijeta gledaju se kroz tri različita pojma:

- *entiteti*
- *atributi*
- *veze.*

## 2.1. **Entiteti**

Pod entitetom možemo smatrati objekte, događaje ili pojave, odnosno sve o čemu želimo prikupljati određene podatke. Svaki entitet ima atribute, odnosno podatke koji su usko vezani za taj entitet. Najčešće se za ime entiteta uzima imenica u jednini. Radimo li modifikaciju na postojećoj BP i to na način da dodajemo relacije (entitete tj. tablice), poželjno je ispred imena entiteta staviti neki alfa-indeks kako ne bismo došli u koliziju s postojećim imenima tablica koje se već nalaze u BP [9]. Prikaz općenite strukture je na slici 2.3.




*Slika 2.2: Entitet i atributi*

## 2.2. **Atributi**

Atributi su podatci koje želimo imati vezano uz neki entitet. Neki atributi jednoznačno određuju entitet u promatranom skupu, dakle ne postoje dva entiteta s posve istim vrijednostima tih atributa. Takve atribute nazivamo identifikatorima ili ključevima entiteta. Ključ (identifikator) je jednostavan ako se sastoji od samo jednog atributa. Složeni ključ (identifikator) sadrži dva ili više atributa. Imamo li atribut koji je svojim obilježjima dovoljno složen, možemo taj atribut „promovirati“ u entitet. Atribute možemo podijeliti na sljedeći način:

• *osnovni*

(Ime, prezime, adresa itd.) Ove se vrijednosti ne mogu izvesti iz drugih atributa.

• *definirani*

To su izmišljeni atributi i postoje samo u bazi podataka (jedinstveni identifikator odjela, jedinstveni ID osobe itd. Jednom dodijeljena vrijednost se ne mijenja.

• *izvedeni*

Ovo je vrijednost koja se može izračunati iz vrijednosti drugih atributa u bazi podataka, na primjer starost zaposlenika kada je datum rođenja u bazi. Ovi se atributi u općem slučaju ne bi trebali pamtiti u bazi već izračunavati po potrebi [9].

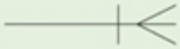
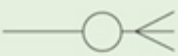
## 2.3. **Kardinalnost veze**

Veze se uspostavljaju između dvaju ili više tipova entiteta. Zapravo je riječ o imenovanoj binarnoj ili *k*-narnoj relaciji između primjeraka entiteta zadanih tipova. Funkcionalnost veze može biti:

*Jedan-naprema-jedan* (1:1). Jedan primjerak prvog tipa entiteta može biti u vezi s najviše jednim primjerkom drugog tipa entiteta te također jedan primjerak drugog tipa može biti u vezi s najviše jednim primjerkom prvog tipa.

*Jedan-naprema-mnogo* (1:N). Jedan primjerak prvog tipa entiteta može biti u vezi sa 0, 1 ili više primjeraka drugog tipa entiteta, no jedan primjerak drugog tipa može biti u vezi s najviše jednim primjerkom prvog tipa.

*Mnogo-naprema-mnogo* (M : N). Jedan primjerak prvog tipa entiteta može biti u vezi sa 0, 1 ili više primjeraka drugog tipa entiteta te također jedan primjerak drugog tipa može biti u vezi s 0, 1 ili više primjeraka prvog tipa.

| Simbol | Značenje |
|---|---|
|  | Jedan - obavezno |
|  | Više - obavezno |
|  | Jedan - opcionalno |
|  | Više - opcionalno |

*Slika 2.3: Oznake veza „Vranino stopalo“*

*Involuirana veza* povezuje jedan tip entiteta s tim istim tipom. Dakle, riječ je o binarnoj relaciji između raznih primjeraka entiteta istog tipa. Funkcionalnost takve veze opet može biti (1 : 1), (1 : N), odnosno (M : N).

*Ternarne veze* uspostavljaju se između triju tipova entiteta. Riječ je o ternarnoj relaciji između primjeraka triju tipova entiteta. Postoje brojne mogućnosti za funkcionalnost ternarne veze, na primjer (N : M : P), (1 : N : M), (1 : 1 : N) ili čak (1:1:1) [10].

Neke od mnogih oznaka koje se koriste za grafički prikaz veza prikazane su na slici 2.3. Tu se konkretno radi o oznakama koje je osmislio James Martin [11], a zovu se „Martinove oznake“ ili oznake „Vranino stopalo“ (engl. *Crow`s foot).*

## 2.4. **Ključevi**

Ključ K relacije R je podskup atributa od R koji ima sljedeća “vremenski nezavisna” svojstva:

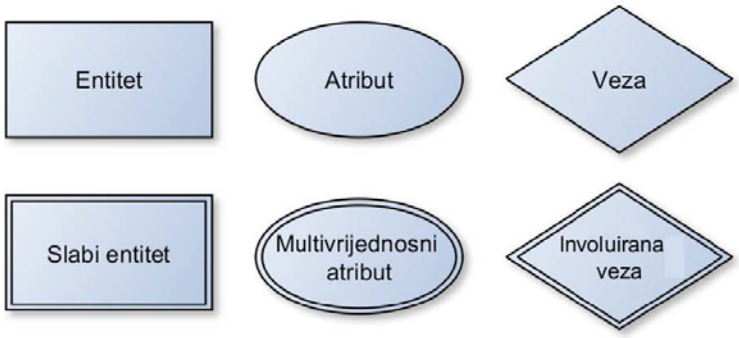
1. Vrijednosti atributa iz K jednoznačno određuju *n*-torku u R. Dakle, ne mogu u R postojati dvije *n*-torke s istim vrijednostima atributa iz K.

2. Ako izbacimo iz K bilo koji atribut, tada se narušava svojstvo 1.

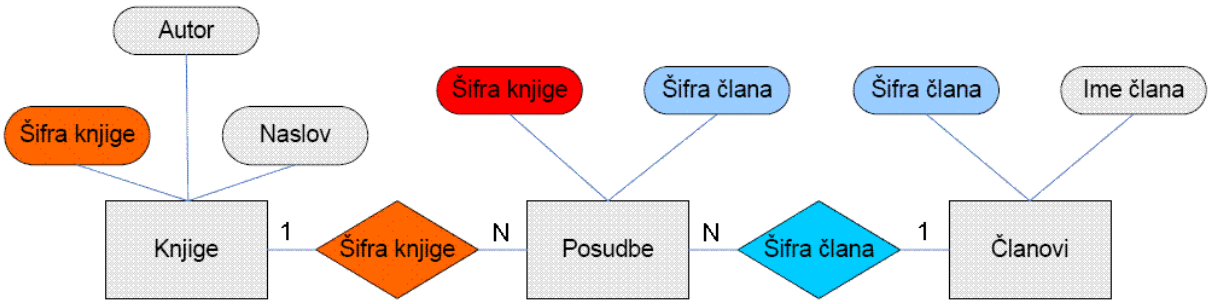
Budući da su sve *n*-torke u R međusobno različite, K uvijek postoji. Naime, skup svih atributa zadovoljava svojstvo 1. Izbacivanjem suvišnih atributa doći ćemo do podskupa koji zadovoljava i svojstvo 2. Događa se da relacija ima više kandidata za ključ. Tada jedan on njih proglašavamo primarnim ključem. Atributi koji sastavljaju primarni ključ zovu se primarni atributi. Vrijednost primarnog atributa ne smije ni u jednoj *n*-torki ostati neupisana [12].

## 2.5. **Prikaz konceptualnog modela s pomoću dijagrama**

Običaj je da se ER model nacrta kao dijagram u kojem pravokutnici predstavljaju tipove entiteta, elipse predstavljaju atribute, a rombovi veze što je osmislio tvorac ER dijagrama Peter Chen (slika 2.4). Veze su povezane crtama s odgovarajućim tipovima entiteta. Imena tipova entiteta i veza, te funkcionalnost veza, uneseni su u dijagram. Posebno se prilaže lista atributa za svaki entitet, odnosno vezu. U toj listi možemo specificirati obaveznost članstva u vezama [9]. Kao primjer navodimo ER model knjižnice prikazan na slici 2.6 s oznakama koje je uveo Peter Chen.




*Slika 2.4: Grafičke oznake (tvorac Peter Chen)*




*Slika 2.5: Primjer razvijenog konceptualnog modela knjižnice nastao prema notacijama Petera Chana*

## 2.6. **Normalizacija relacijskog modela**

Relacijski model dobiven iz ER modela može sadržavati nedorečenosti koje treba otkloniti prije implementacije. Proces daljnjeg dotjerivanja modela zove se normalizacija. Teorija normalizacije zasnovana je na pojmu normalnih formi. Relacije (tablice) dobivene u ER modelu morale bi u najmanju ruku biti u prvoj normalnoj formi (1NF). U praksi je lako naići na relacije koje odstupaju od 2NF, 3NF. Teorija normalizacije nije ništa drugo nego formalizacija nekih intuitivno prihvatljivih principa o "zdravom" oblikovanju modela. Ako već na početku dobro uočimo sve potrebne entitete, atribute i veze, nikakva nam daljnja normalizacija neće biti potrebna.

Relacija je u *prvoj* normalnoj formi (1NF) ako je vrijednost svakog atributa jednostruka i nedjeljiva.

Relacija je u *drugoj* normalnoj formi (2NF) ako je u 1NF i ako je svaki ne-primarni atribut potpuno funkcionalno zavisan o primarnom ključu.

Relacija je u *trećoj* normalnoj formi (3NF) ako je u 2NF i ako ne sadrži tranzitivne zavisnosti. Preciznije, relacija R je u 3NF ako za svaku funkcionalnu zavisnost $X \rightarrow A$ u R, takvu da A nije u X, vrijedi: X sadrži ključ za R ili je A primarni atribut.

Relacija R je u četvrtoj normalnoj formi (4NF) ako vrijedi: kad god postoji višeznačna zavisnost u R, na primjer A ->-> B, tada su svi atributi od R funkcionalno zavisni o A. Ekvivalentno, R je u 4NF ako je u BCNF i sve višeznačne zavisnosti u R su zapravo funkcionalne zavisnosti.

Za većinu praktičnih primjera dovoljno je relacije normalizirati do 3NF. Ponekad je potrebno neku relaciju i dalje normalizirati do 4NF. Peta normalna forma, koja se također navodi u literaturi, nije od praktičnog značaja. Postoje razlozi zbog kojih izuzetno možemo odustati od pune normalizacije.

*Složeni atribut.* Događa se da nekoliko atributa u relaciji čine cjelinu koja se u aplikacijama nikad ne rastavlja na sastavne dijelove.

*Efikasno čitanje podataka.* Normalizacijom se velike relacije razbijaju na mnogo manjih. Kod čitanja je često potrebno podatke iz malih relacija ponovno sastaviti u veće *n*-torke. Uspostavljanje veza među podatcima u manjim relacijama traje znatno duže nego čitanje podataka koji su već povezani i upisani u jednu veliku relaciju [12].

# 3. Konceptualni model - „Arhiva“

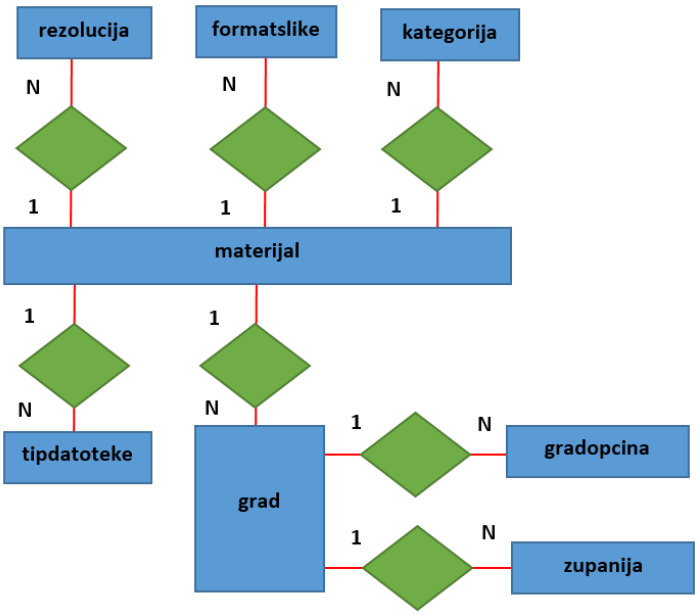
Uvidom u radne procese koje želimo „digitalizirati“ olakšali smo posao kreiranja ER modela. Kod izrade ER modela nužno je slijediti niz pravila od kojih je prvo identifikacija entiteta. U ovom slučaju fokus je stavljen na segment „Arhive“.

Identificirani entiteti su: *materijal, grad, gradopcina, zupanija, tipdatoteke, kategorija formatslike, rezolucija.*

U drugom koraku određujemo atribute identificiranih entiteta. Podvučeni atributi su primarni ključevi relacije.

- **materijal** (*id_materijal, naziv, datum, k_rijec, opis, putanja*)
- **grad** (*id_grad, naziv*)
- **gradopcina** (*id_grad_opcina, naziv*)
- **tipdatoteke** (*id_tip_datoteke, naziv*)
- **rezolucija** (*id_rezolucija, naziv*)
- **formatslike** (*id_format_slike, naziv*)
- **kategorija** (*id_kategorija, naziv*)
- **gradopcina** (*id_grad_opcina, naziv*)
- **zupanija** (*id_zupanija, naziv*)

Nakon identifikacije entiteta i raspisivanja pripadajućih atributa definiraju se veze i njihove kardinalnosti.




*Slika 3.1: Razvijeni konceptualni model s definiranim vezama, bez atributa*

## 3.1. Pretvorba konceptualnog modela u relacijski - „Arhiva“

Kod pretvorbe ER modela u relacijski nužno je pridržavati se određenih pravila, posebno za binarne, posebno za ternarne veze.

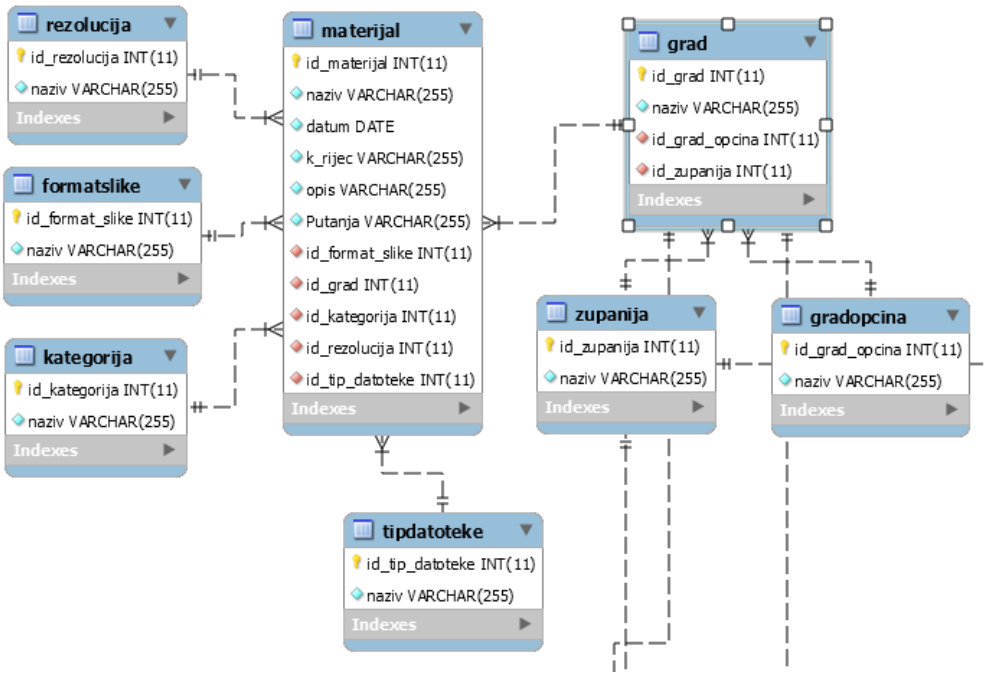
*Pravilo 1.* Ako tip entiteta E2 ima obavezno članstvo u N:1–vezi s entitetom E1, tada relacija za E2 treba uključiti primarne atribute (primarni ključ) od E1.

*Pravilo 2.* Ako tip entiteta E2 ima neobavezno članstvo u N:1– vezi s entitetom E1, tada vezu možemo prikazati

- na prethodni način uvođenjem ključa
- uvođenjem nove relacije čiji su atributi primarni atributi (primarni ključevi) od entiteta E1 i E2.

*Pravilo 3.* Ako je tipa N:M, uvijek se prikazuju posebnom relacijom koja uključuje primarne atribute oba entiteta te još možda dodatne koje sama veza ima.

Pridržavajući se nabrojenih pravila došlo se do modela (slika 3.2) koji smo razvili s pomoću alata MySQL Workbench.




*Slika 3.2: Relacijski model baze podataka u segmentu „Arhive“*

Kao primarne ključeve za sve relacije, odnosno tablice, kreirali smo atribut koji jednoznačno označava *n*-torku, a to je *id* tipa *INT*.

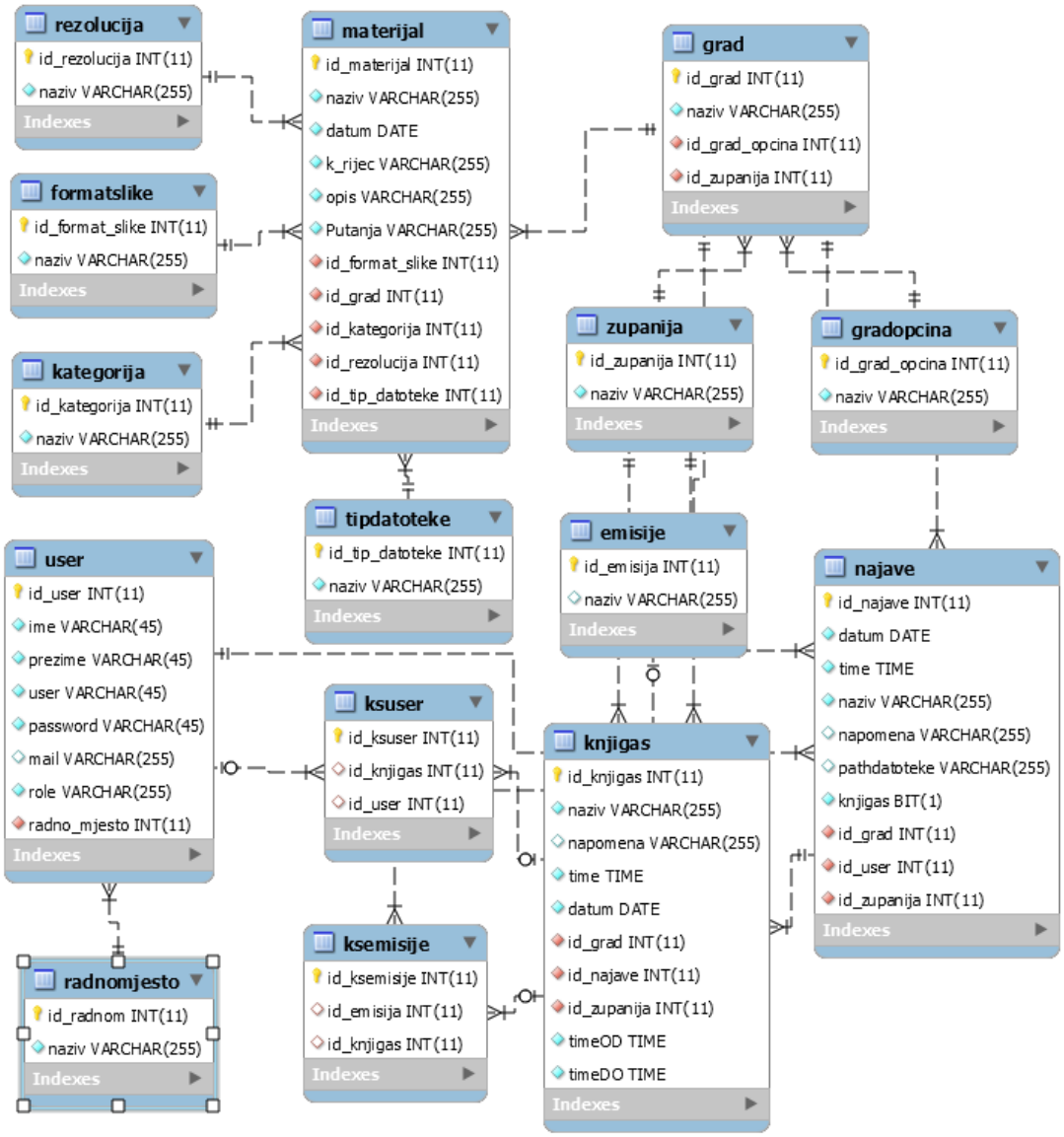
U modelu je vidljivo da su u relaciju „materijal“ uvršteni primarni ključevi relacija koje su u N:1 vezi s relacijom „materijal“, što proizlazi iz pravila da ako tip entiteta E2 ima obavezno članstvo u N:1–vezi s entitetom E1, tada relacija za E2 treba uključiti primarne atribute (primarni ključ) od E1. Također u tablicu „grad“ uvrstili smo primarne ključeve tablice „gradopcina“ i „zupanija“.

Kako bismo shvatili dobiveni relacijski model, nužno je poznavati relacije od kojih smo došli do prikazanog modela. Ako gledamo sa strane materijala, materijal može biti sniman samo na jednom mjestu, dok se u tom mjestu može snimiti više materijala. Također materijal može imati samo jedan tip datoteke, rezoluciju, format slike, kategoriju, dok rezolucija, tip datoteke, format slike i kategorija može biti sadržana u više materijala.

Nadalje, imamo relaciju „grad“. Grad može biti samo u jednoj županiji, dok županija može imati više gradova. Također u ovoj relaciji bilo nam je bitno da možemo identificirati radi li se o gradu ili općini te smo uvrstili primarni ključ iz relacije „gradopcina“ koja je strukturom jednaka „šifrarniku“.

## 3.2. **Relacijski modeli - „Knjiga najava“ , „Knjiga snimanja“**

Pridržavajući se pravila spomenutih u prethodnom tekstu, identificirali smo entitete, pripadajuće atribute te veze između njih koje su potrebne kako bismo kreirali bazu podataka. Radi se o jednoj bazi podataka koja sadrži informacije o arhivi videomaterijala, knjizi najava, knjizi snimanja te obavijestima. Također, zbog prava pristupa u aplikaciju, nužno je bilo kreirati relaciju „*user*“ koja sadržava podatke o korisnicima.



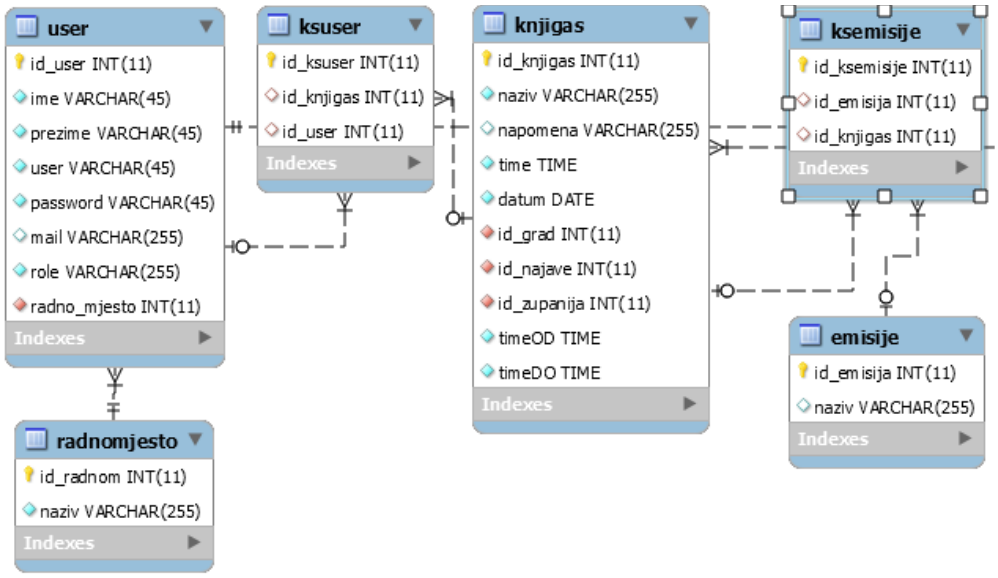

*Slika 3.3: Relacijski model fizički kreirane baze podataka*

Vrijedi istaknuti da se kod kreiranja relacijskog modela cjelokupne baze sa svim relacijama pojavila potreba za primjenom pravila broj 3 koje nam govori da ako je veza između entiteta tipa N:M (više na više), uvijek se prikazuju posebnom relacijom koja uključuje primarne atribute oba entiteta te još možda dodatne koje sama veza ima. To je vidljivo na slici 3.4 između relacija

„*knjigas*“ i „*emisije*“, odnosno dodali smo relaciju „*ksemisije*“ koja sadržava primarne ključeve iz obje povezane relacije. Kako bismo objasnili vezu između tih dviju relacija, potrebno je opisati na koji se način došlo do predmetne veze. Kako jedno snimanje može biti za više emisija, tako može biti i više snimanja za jednu emisiju.

Također, možemo primijetiti da se ista veza pojavljuje između relacija „*knjigas*“ i „*user*“ te smo kreirali dodatnu relaciju „*ksuser*“ iz čega proizlazi da može biti više zaposlenika (snimatelja, novinara, tonskih snimatelja) zaduženih za jedno snimanje, kao što može biti više snimanja s istim zaposlenicima.

Kod kreiranja relacija odmah su određeni tipovi podataka atributa vidljivi na slici 3.3. Nakon uspješno kreirane baze podataka, krenulo se na izradu same aplikacije.




*Slika 3.4: Prikaz kreiranja dodatnih relacija zbog veze M:N*

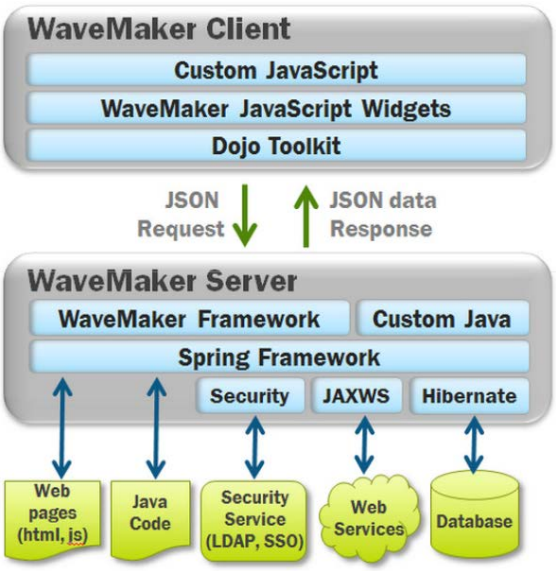
# 4. Alati za izradu *web*-aplikacije

Svi alati navedeni u nastavku su otvorenog koda. Kako financijski aspekt razvoja aplikacije nije bio zanemariv, upravo je to jedan od razloga zbog čega će se koristiti navedeni alati. Također, široka baza korisnika te dobra dokumentiranost bili su razlozi prihvaćanja tih alata.

## 4.1. WaveMaker - razvojno okruženje

Kako je već spomenuto u uvodu, za razvoj Javine *web*-aplikacije, odnosno sustava za upravljanje podatcima, korištena je razvojna okolina WaveMaker Studio Desktop 7.8. WaveMaker je vizualno razvojno okruženje koje služi za razvijanje, održavanje i moderniziranje standardnih Javinih *web*-aplikacija. Temelji se na ubrzanom razvoju softvera (engl. *Rapid Application Development - RAD*) što programerima omogućava povećanje produktivnosti, a smanjuje potrebu za pisanjem Javina programskog koda čak i do 98 %. Temelji se na tehnologiji otvorenog koda (engl. *Open Source*) kao što su Spring, Hibernate, Dojo Toolkot i Tomcat. Slijedom navedenog, početnici mogu razvijati osnovne Javine *web*-aplikacije bez potrebe za pisanjem programskog koda, odnosno korištenjem *drag-and-drop* alata. Naprednijim korisnicima omogućeno je izvoženje projekta (podržan Maven), a potom kroz druga integrirana razvojna okruženja (engl. *Integrated development environment - IDE*) nastaviti njegov razvoj.

WaveMaker je vizualni WYSIWYG (engl. *What You See is What You Get*) razvojni alat koji se izvodi u standardnom *web*-pregledniku. Korištenjem WaveMaker Studija se programerima koji imaju iskustva sa sličnim programskim jezicima i tehnologijama za čak 92 % skraćuje vrijeme potrebno za učenje programskog jezika Java. Njime se najčešće koristimo kod izrade *Form-driven database* aplikacija ili *front-end* dijela za aplikacije s uslugama orijentiranom arhitekturom (engl. *Service Oriented Architecture - SOA*), dok nije pogodan za razvoj sustava sa složenim transakcijskim modelom ili sustava sa složenim radnim procesom [13]. Danas na tržištu postoji komercijalna verzija alata koja se pokreće u „oblaku“ (engl. *Cloud application*) te starije verzije alata koje su besplatne za korištenje, no više nema tehničke podrške [14].




*Slika 4.1: WaveMaker arhitektura*

Podržan je razvoj *web*-aplikacija koje se koriste postojećom BP ili se može kreirati nova BP upotrebom alata za izradu relacijskog modela koji je sastavni dio razvojne okoline. Također podržava HSQLDB, Oracle, PostgreSQL, MySQL, MariaDB, MSSQL, IBM DB2 baze podataka. Koristi se objektno-relacijskim mapiranjem (engl. *Object/Relational Mapping - ORM*), pri čemu samo mapiranje generira automatski (.xml). Izvoženje projekta moguće je u .war datoteku koju podržavaju svi Javini poslužitelji (Tomcat, WebSphere, Jboss, Glassfish, Weblogic itd.).


*Slika 4.2: WaveMaker mogućnosti uvoza i izvoženja*

## 4.2. MySQL - baza podataka

MySQL je sustav otvorenog koda za upravljanje bazom podataka - SUBP. Uz PostgreSQL, MySQL je čest izbor baze za projekte otvorenog koda, te se distribuira kao sastavni dio poslužiteljskih Linux distribucija, no također postoje inačice i za ostale operacijske sustave poput Mac OS-a, Windowse itd. MySQL [6] je izradila švedska tvrtka MySQL AB 1995. godine, koja je osnovana iste godine. Osnivači su joj Michael Widenius, David Axmark i Allan Larsson. Danas je MySQL jedan od najpopularnijih SUBP-a s preko 100 milijuna aktivnih instalacija.

SUBP predstavlja programski sustav kojim se koristimo za pristup, pohranu, te manipulaciju podatcima u bazi podataka. Njime se koristimo za interakciju korisnika i baze podataka, a korisnik može sa SUBP-om imati direktnu interakciju, ili preko aplikacije programirane u nekim od programskih jezika poput Jave, C++, Visual Basica.

Neophodni dijelovi svakog SUBP-a su jezici za upravljanje bazama podataka:

- DDL (engl. *data description language*) - jezik za definiciju ili deklaraciju objekata u bazi podataka
- DML (engl. *data manipulation language*) - jezik za manipulaciju objektima baze podataka
- DCL (engl. *data control language*) - jezik za postavljanje dozvola u bazi podataka
- TCL (engl. *transaction control language*) - jezik za upravljanje transakcijama [14]

## 4.3. MySQL Workbench

MySQL Workbench [5] je grafički alat za dizajniranje baza podataka koji integrira SQL razvoj, administraciju, dizajn i održavanje u jedno zajedničko sučelje za MySQL baze podataka. Prva ogledna inačica razvijena je 2005. godine i nasljednik je DBDesigner4 alata. Dio je MySQL paketa od 2007. godine, a njegovo numeriranje je počelo inačicom 5.0, jer je naslijedio DBDesigner4. Alat postoji u besplatnom i komercijalnom izdanju, te je drugi alat po preuzimanjima sa MySQL *web*-stranice.

## 4.4. JAVA - programski jezik

1995. Sun Microsystems izdaje prvu verziju programskog jezika Java [15]. Osoba s vizijom stvaranja naprednijeg C++-a tj. programskog jezika Java je James Gosling.

Radi se o objektno orijentiranom programskom jeziku. Pojam objektno orijentirano programiranje znači metodu razvijanja programske opreme kod koje su programi koncipirani poput grupe objekata koji međusobno djeluju. Programski jezik Java jednostavniji je za korištenje od programskog jezika C ili C++. Jedna od bitnijih značajki je „neovisnost o platformi“. Ovo omogućuje da se kôd napisan u Javi izvodi bez unošenja promjena u različitim okruženjima na računalima (Windows, UNIX…). Javin programski kôd prevodi se u tzv. „bytecode“ koji pokreće operacijski sustav, drugi program ili uređaji s pomoću Javina interpretera. *Bytecode* nije izvršni kôd, već visoko optimizirani skup instrukcija dizajniran za izvršavanje unutar Javina izvršnog sustava koji se naziva „Javin virtualni stroj“ (engl. *JAVA virtual machine – JVM*) koji predstavlja interpreter za *bytecode*. Java je dizajnirana za izvršavanje u distribuiranom okruženju, prvenstveno na internetu.

## 4.5. Apache Tomcat - poslužitelj

Apache Tomcat je aplikacijski poslužitelj od tvrtke Apache Software Foundation koji pokreče Javin program što se izvršava na *web*-poslužitelju (Servlet – Java klasa koja služi za obradu *web*-zahtjeva, donosno proširuje mogućnosti aplikacijskog poslužitelja.) na zahtjev klijenta. Apache Tomcatom se možemo koristiti kao malim samostalnim *web*-poslužiteljem ili kao poslužiteljem za servlete integriranim u Apache HTTP poslužitelj.

Nakon instalacije samog poslužitelja pristupa mu se preko *web*- preglednika na definiranom portu 8080. Ulaženjem u panel za administraciju moguće je daljnje podešavanje poslužitelja, kao i uvoženje .war datoteka, odnosno Javinih *web*-aplikacija.

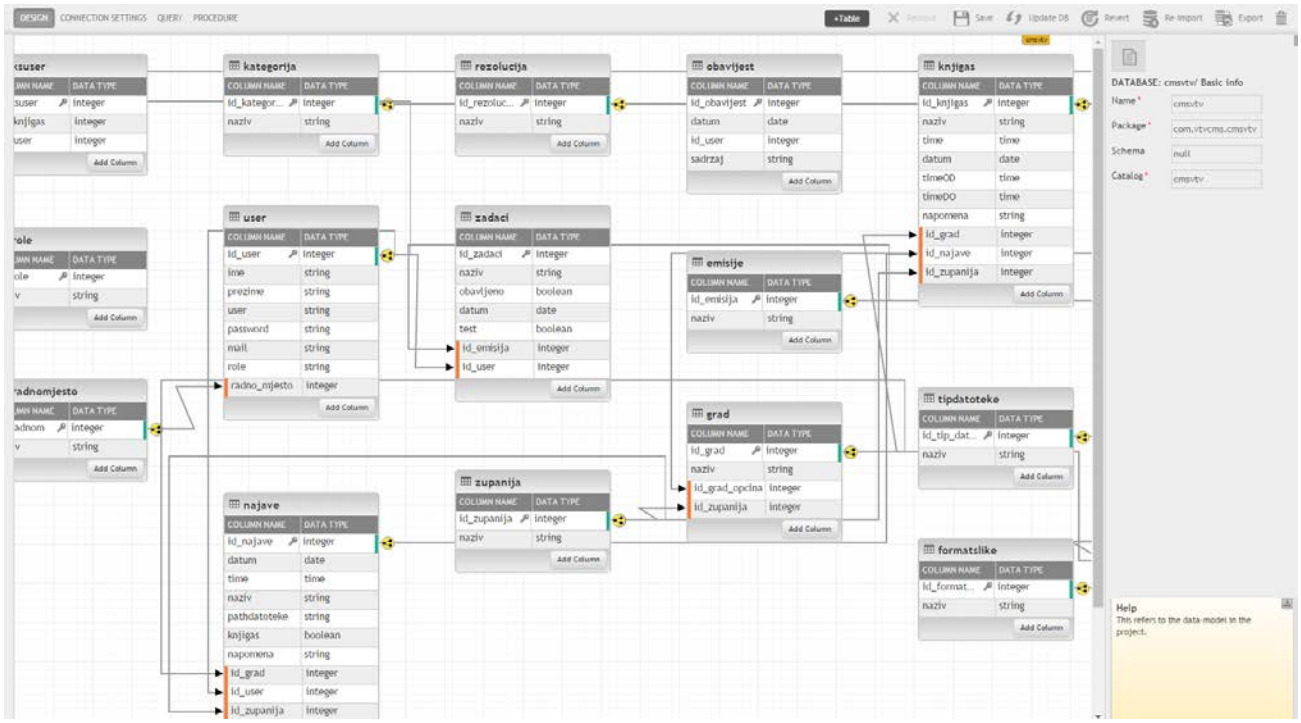
# 5. Izrada *web*-aplikacije

Izrada *web*-aplikacije krenula je uvoženjem BP-a u radnu okolinu. Nakon svakog razvijenog segmenta aplikacija je testirana i prezentirana korisnicima kako bi se dobile dodatne informacije o mogućim problemima, odnosno mogućnostima za unapređenje. U nastavku su opisani koraci izrade *web*-aplikacije.

## 5.1. Uvoz baze podataka

Prvi korak pri izradi aplikacije je podešavanje veze prema već kreiranoj bazi podataka te uvoženje same baze podataka u razvojnu okolinu. Kako je napomenuto korištena je baza podataka MySQL.

Prilikom uvoženja baze podataka automatski se generiraju tablice i pripadajući oblici za unos podataka u predjelu alata razvojne okoline, a kojima se kasnije možemo koristiti na *drag-and-drop* način. To nam uvelike olakšava prikaz, odnosno manipulaciju podatcima iz baze podataka. Dio razvojne okoline sadrži prostor za kreiranje vlastitih SQL upita, što je izrazito pogodno ako se radi o kompleksnim SQL upitima.

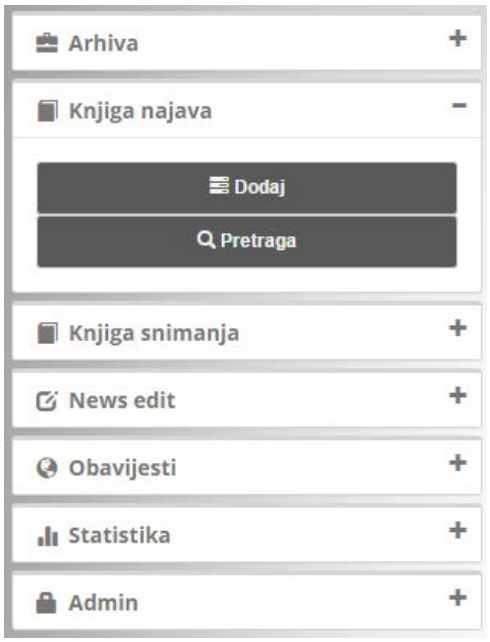



*Slika 5.1: Baza podataka prikazana u razvojnoj okolini*

## 5.2. **Oblikovanje početne stranice i izbornika**

Kako je unaprijed definirana struktura aplikacije, krenulo se u izradu padajućeg izbornika koji je smješten na lijevoj strani i proteže se kroz sve stranice *web*-aplikacije.

Izbornici imaju mogućnost prikazati se, odnosno sakriti, ovisno o korisničkim pravima. Tako je na slici 5.2 prikazan kreirani izbornik koji je vidljiv korisnicima s *Admin* pravom. Također možemo grafički unaprijediti sam izgled izbornika dodavanjem ikona koje su sadržane u „knjižnici“ razvojne okoline. Klikom na pojedini izbornik otvaraju se veze prema drugim stranicama. Izbornik „Arhiva“ otvara pristup stranici „Pretraga“. Kod „Knjige najava“ otvaranjem izbornika dolazimo do stranica za dodavanje najava i pretragu istih. Izbornik „Knjiga snimanja“ otvara mogućnost pristupanja pregledu i pretrazi, dok „Obavijesti“ nude pristup stranici za unos obavijesti. Klikom na „Statistika“ otvara se poveznica prema stranici s prikazom statističkih podataka, dok „Admin“ skriva poveznice do svih tablica kreiranih prilikom uvoza baze podataka u razvojno okruženje.

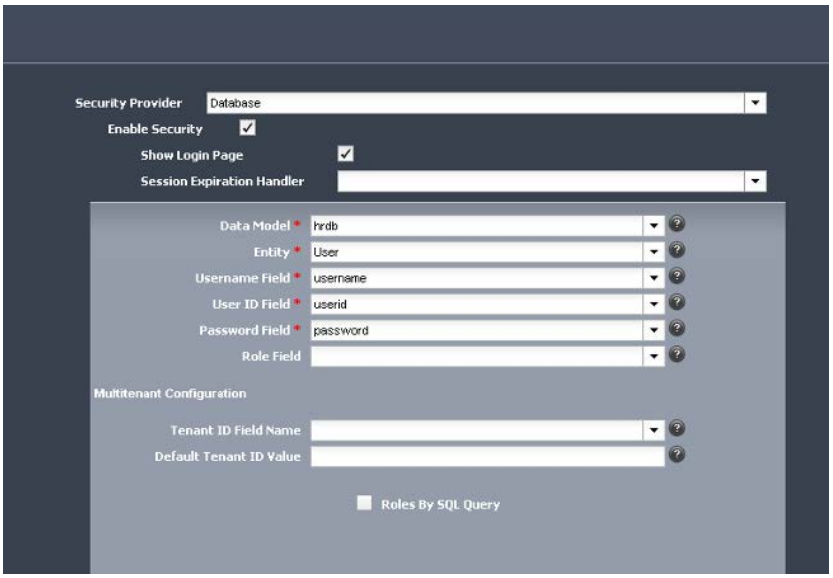



*Slika 5.2: Kreirani izbornik*

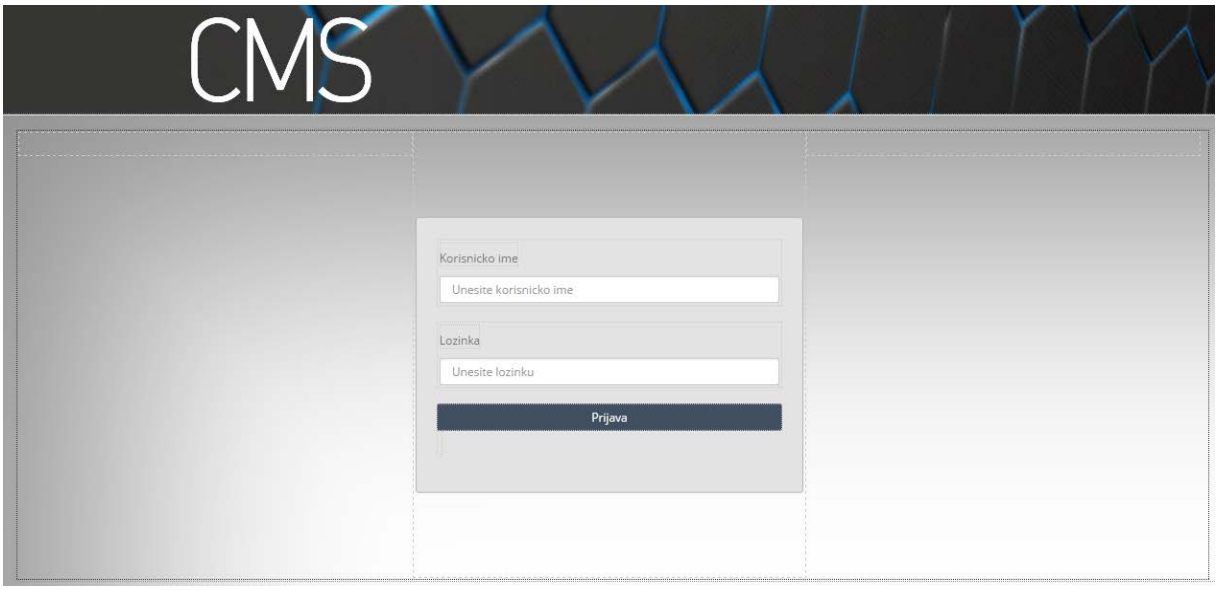
Kod kreiranja stranice uzeo se predložak iz razvojne okoline koji se prilagodio potrebama projekta. Uz dodavanje zaglavlja i podnožja stranice koji se protežu kroz sve stranice aplikacije, dobio se grubi izgled prve stranice *web*-aplikacije koji je prikazan na slici 5.3. Kod kreiranja stranice razvojna okolina generira *.xml* datoteku u kojoj je moguće daljnje fino podešavanje elemenata sadržanih na stranici. Kako se radi o početnoj stranici, iz alatne se trake dodao element koji će prikazivati obavijesti kao što je bilo zadano projektom. Također, dodan je gumb za povratak na početnu stranicu i odjavu iz aplikacije. Tom smo se stranicom koristili kao predloškom za razvoj ostalih stranica aplikacije.

Postavljanjem sigurnosnih postavki, odnosno omogućavanjem servisa za autentifikaciju, tj. ograničavanje pristupa aplikaciji korisnicima koji nisu registrirani, kreiran je novi prozor na kojem je prozor u koji je potrebno upisati korisničke podatke kako bi se pristupilo aplikaciji. Aplikacija podatke relevantne za autorizaciju čita iz baze podataka u tablici „*user*“. Zbog toga tablica „*user*“ sadržava atribute „*user*“ i „*password*“ s pomoću kojih se vrši provjera korisnika.

Omogućavanje prikaza stranice za provjeru korisnika vrši se u razvojnoj okolini. Nakon omogućavanja provjere korisnika aplikacija sama generira početni prozor za prijavu. Daljnje mogućnosti podešavanja prikazane su na slici 5.3. Generirani prozor za prijavu prikazan je na slici 5.4.



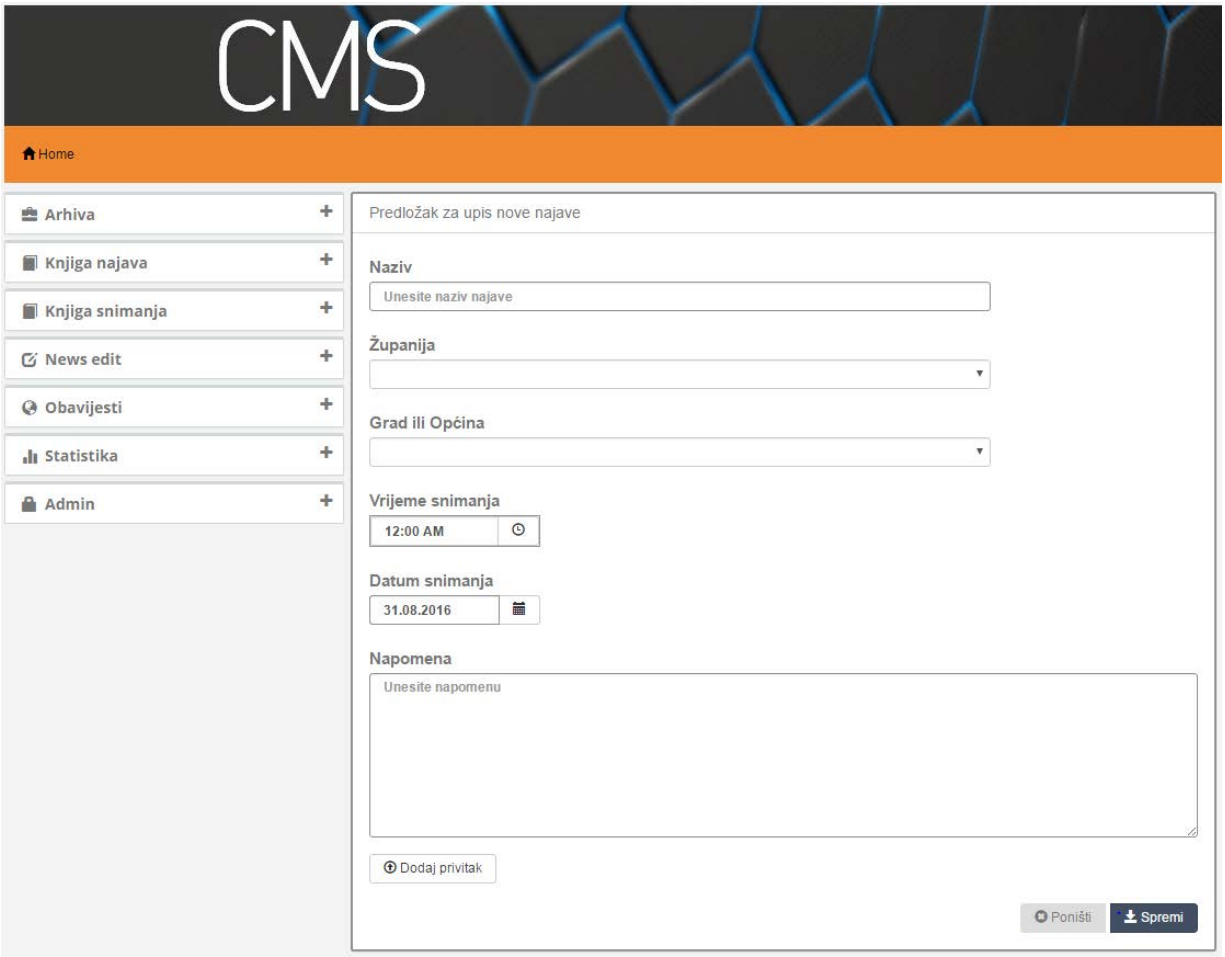

*Slika 5.3: Postavke razvojne aplikacije za provjeru korisnika*



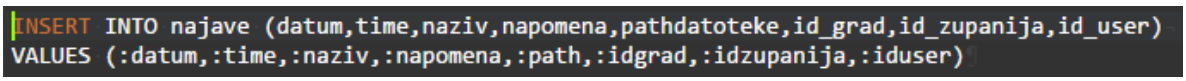

*Slika 5.4: Izgled prozora za prijavu*

## 5.3. **Izrada modula „Knjiga najava“**

Kao što je navedeno, stranica za unos najava ima izgled obrasca za popunjavanje, s time da sadrži elemente koji imaju padajući izbornik (županija, grad ili općina). Kroz razvojnu okolinu moguće je dodatno podešavanje elemenata, odnosno moguće je odrediti koji će se podatci iz baze podataka prikazivati u pojedinom elementu. Kao što je vidljivo na slici 5.6 imamo elemente koji sadržavaju ime županije odnosno grada ili općine. Klikom na te elemente otvara se padajući izbornik gdje se prikazuju podatci upisani u tablici „*zupanija - naziv*“, odnosno „grad - naziv“. Odabirom županije filtrira se prikaz gradova na gradove koji se nalaze samo u odabranoj županiji. Samo upisivanje najave u bazu podataka vrši se automatski iz razvojne okoline. To je riješeno na način da se pišu parametarski SQL upiti. Kreiranjem ručno izrađenih upita u razvojnoj okolini nastaje servisna varijabla u kojoj se parametri iz upita spoje s elementima na stranici. Kod umetanja najave u bazu podataka koristimo se SQL upitom prikazanim na slici 5.7.




*Slika 5.6: Izgled stranice za upisivanje događaja*

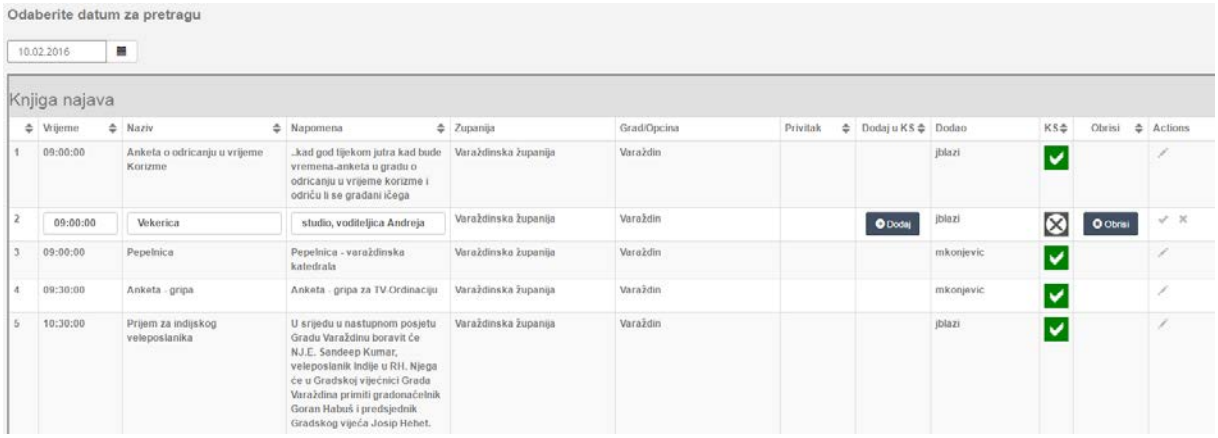



*Slika 5.7: SQL upit za umetanje podataka u bazu*

Kod pregleda upisanih najava korištena je već generirana tablica (engl. *Widget*) u razvojnoj okolini koja sadrži sve podatke iz relacije „*najave*“. Također je bitno napomenuti da prikaz sadrži i podatke iz relacija koje su u vezi s relacijom „*najave*“, odnosno spajanje relacija je napravljeno automatski. Ukoliko se ne želi prikazati sve atribute, kroz podešavanje parametara određeni se atributi mogu sakriti. Na slici 5.8 prikazan je pregled „Knjige najava“, u kojem su isključeni neki atributi kao što su „*user_id*“, „*id_najava*“ itd., a koji nisu bitni korisniku za rad. U tablicu je dodan stupac „KS“ u kojoj je grafički vidljivo nalazi li se određena najava u „Knjizi snimanja“. Ako se najava ne nalazi u „Knjizi snimanja“, pojavljuje se gumb „Dodaj“, odnosno “Obriši“. Nije moguće obrisati najavu koja se nalazi u „Knjizi snimanja“. Kod pritiska na gumb „Dodaj“ korisnik se preusmjerava na stranicu u kojoj upisuje dodatne podatke potrebne kako bi odabranu najavu upisao u „Knjigu snimanja“. Programski kod (AngularJS [17]) koji nam omogućuje navedeno ponašanje te izgled gumba za dodavanje najave prikazan je u nastavku.

```
<ng-switch on="row.getProperty('knjigas')">
   <span ng-switch-when="false">
         <wm-button caption="Dodaj" show="true" class="btn-
         sm btn-primary" iconname="glyphicon glyphicon-
         circle-arrow-right" on-click="goToPage-KnjigaStemp"
         data-ng-class="">
         </wm-button>
         </span>
   <span ng-switch-default="">
   </span>
</ng-switch>
```

*AngularJS - Program 1*




*Slika 5.8: Stranica pretrage Knjige najava*

Na slici 5.9 nalazi se stupac s podatkom o korisniku koji je dodao određenu najavu. Time se postiže brža komunikacija između korisnika. Primjerice, ako su potrebne detaljnije informacije o najavljenom događaju (koje se ne nalaze unutar upisane najave) korisnik ima podatak o osobi koja je upisala najavu. Također, kroz stupac „Actions“ moguće je promijeniti sadržaj atributa „Vrijeme“, „Naziv“ i „Napomena“, bez potrebe za brisanjem, te ponovnim upisivanjem najave. Na slici 5.5 može se uočiti gumb „Dodaj privitak“. Klikom na taj gumb otvara se pretraživač datoteka operativnog sistema te nakon odabira datoteke ista se prenese na poslužitelj. U bazu podataka sprema se samo putanja do navedene datoteke. Dakle, parametar „*pathdatoteke*“ iz SQL upita povezuje se s elementom na stranici zaduženim da odradi prijenos datoteke na vanjsku lokaciju. Programski kod za prijenos datoteke već je razvijen u WaveMakeru i dolazi kao sastavni dio razvojne aplikacije. Klikom na gumb „Preuzmi“ (slika 5.9), u pretrazi „Knjige najava“, u prozoru *web*-preglednika otvara se prikaz datoteke. Na koji će se način prikazati datoteka, ovisno je o web-pregledniku i vrsti tipa datoteke.


*Slika 5.9: Gumb „Preuzmi“*

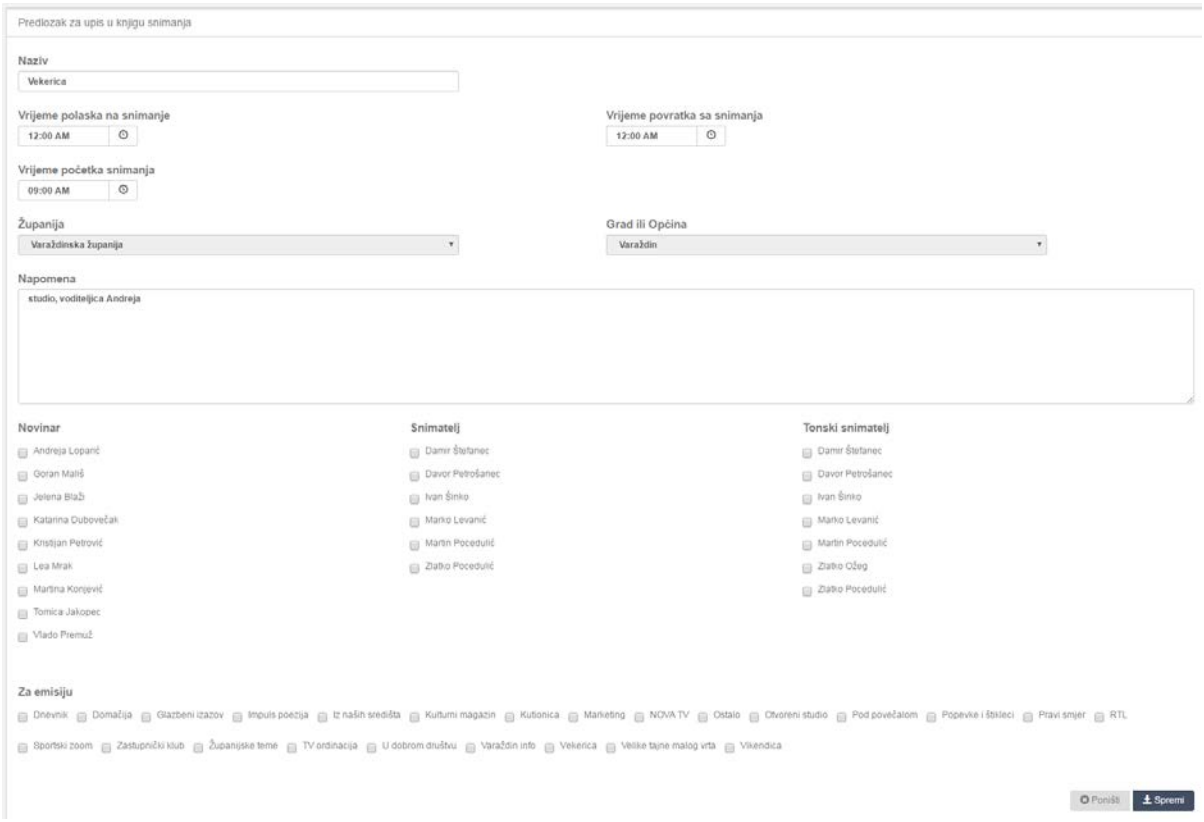
Gumb „Preuzmi“ pokazuje samo za one najave koje sadrže informaciju u entitetu „pathdatoteke“. Programski kod kojim dobivamo takvo ponašanje prikazan je unastavku.

```
<a ng-if="row.getProperty('pathdatoteke') != ''"
nghref=http://ADD_IP:ADD_PORT/****/resources/uploads/{{row.getProp
erty('pathdatoteke')}}
>Preuzmi</a>
```

*AngularJS - Program 2*

## 5.4. **Izrada modula „Knjiga snimanja“**

Nakon pritiska na gumb „Dodaj“ u pretraživanju „Knjige najava“ dolazi se do stranice u koju se upisuju dodatni podatci relevantni za snimanje koje želimo upisati u „Knjigu snimanja“, što je prikazano na slici 5.10. Podatci iz označenog retka u „Knjizi najava“ kopiraju se u „statičnu“ varijablu te se pri prelasku na stranicu za upisivanje dodatnih podataka o snimanju kopiraju u sadržaj (engl. *Data*) elemenata postavljenih na stranici (vrijeme, datum, naziv, napomena itd.). Ostali se podatci ručno odabiru iz izbornika s praznim poljem za označavanje. Kod stranice za upis u „Knjigu snimanja“ opcionalno se može odrediti vrijeme polaska na snimanje i predviđeno vrijeme dolaska sa snimanja. Ti su podatci nužni ako se želi izračunati podatak o efektivnom vremenu provedenom na snimanjima, a koji se nalaze u području statistike opisane *web*-aplikacije. Kako bismo sve podatke upisali istovremeno u sve relacije koje su u korelaciji s relacijom „*knjigas*“, kreirana je nova Javina servisna varijabla.



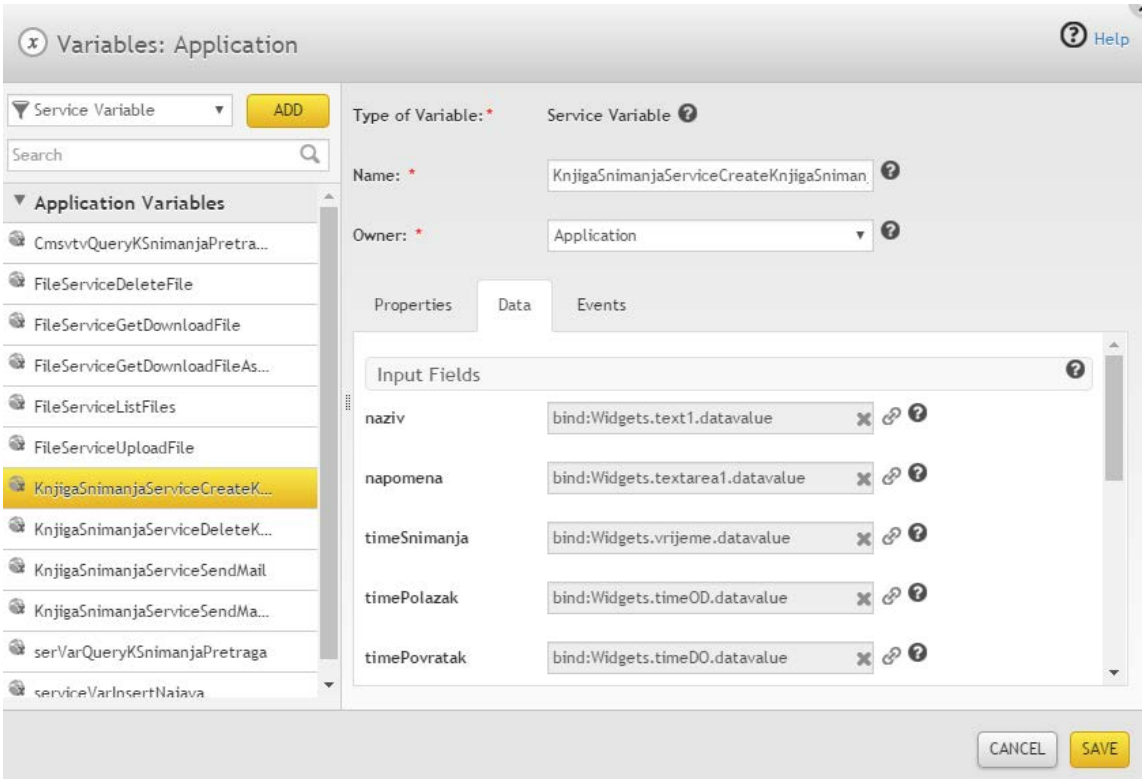

*Slika 5.10: Stranica za upis najava u „Knjigu snimanja“*

Koristeći se „onClick“ metodom na gumbu „Spremi“ aktiviramo dio programskog koda koji se brine da se svi podatci zapišu u relaciju „*knjigas*“ i vezane relacije. Također zadužen je i da se u relaciji „*najave*“ atribut pod nazivom „*knjigas*“ koji je tipa *bool* promijeni iz „false“ u „true“. Time je omogućeno da se u „Knjizi najava“ grafički prikaže kako je najava dodana u „Knjigu

snimanja“ te se onemogući brisanje i ponovno dodavanje iste najave. Također, aktiviranjem Javine servisne varijable dohvaća se adresa elektronske pošte svih sudionika snimanja te se na iste šalje poruka elektronske pošte s podatcima snimanja (mjesto, vrijeme, napomena itd.).

Sam rad Javine servisne varijable možemo opisati na sljedeći način:

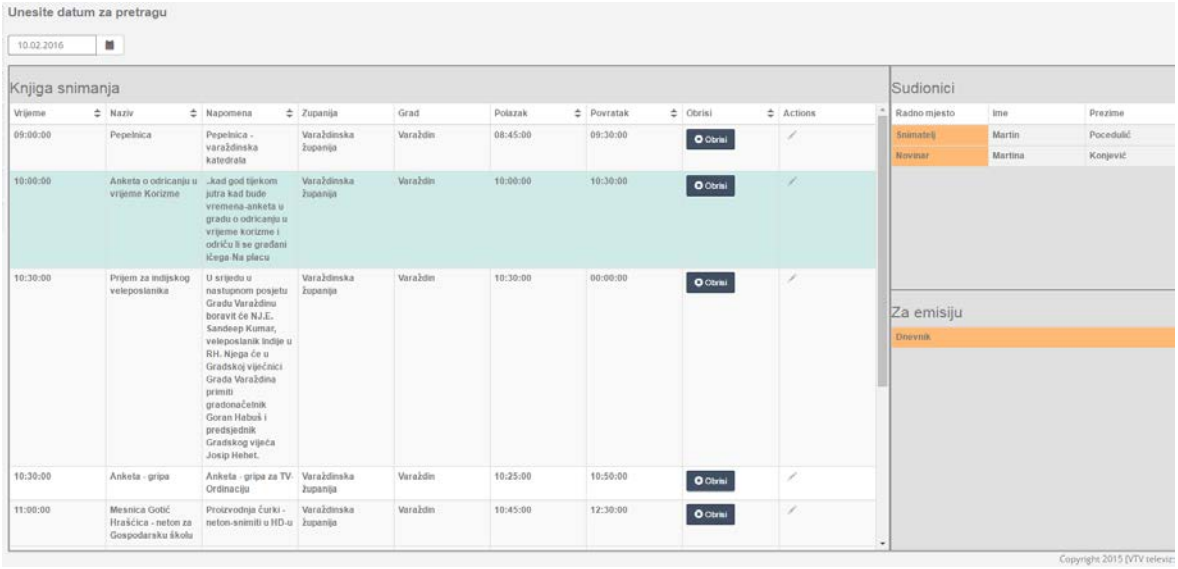
Kada u Javinu programskom kodu kreiramo funkciju s parametrima, automatski se u razvojnom okruženju stvara Javina servisna varijabla koja nudi mogućnost spajanja definiranih parametara u programskom kodu s elementima stranice, a sve kroz grafičko sučelje. Na slici 5.11 prikazano je grafičko spajanje parametara iz Javina koda s elementima postavljenim na stranicu.




*Slika 5.11: Spajanje vanjskih parametara s elementima na stranici*

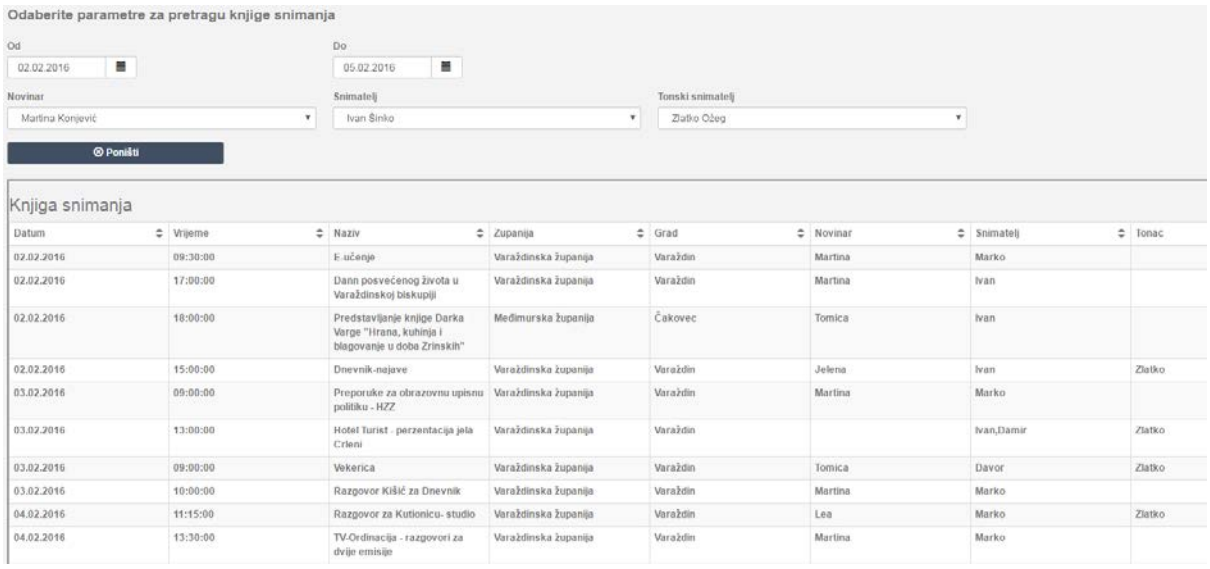
Kako bi se pogledala „Knjiga snimanja“, napravljen je prozor koji nam kroz skup tablica prikazuje podatke o označenom snimanju (slika 5.12). Bitno je da korisnik označi željeno snimanje, nakon čega mu se mu s desne strane prikažu ostali podatci. Važno je napomenuti da je pretraživanje moguće kroz odabir datuma preko elementa (datum) koji se nalazi na stranici. Ovisno o korisničkim pravima prikazuje se gumb „Obriši“. Klikom na gumb „Obriši“ označeni se podatci brišu iz baze podataka, a u „Knjizi najava“ grafički se označava da najava više nije u „Knjizi snimanja“. Vidljivo je da tablica sadrži stupac „Actions“. Klikom na navedeni stupac

moguće je direktno u „Knjizi snimanja“ promijeniti informacije o vremenu snimanja, polasku na snimanje, dolasku sa snimanja i napomene. Neki elementi unutar stranice označeni su različitim bojama s ciljem bržeg uočavanja bitnih podataka.




*Slika 5.12: Izgled stranice za prikaz „Knjige snimanja“*

Pretraživanje knjige snimanja je pogodnost koja omogućuje korisnicima da unutar odabranog vremenskog perioda pretraže snimanja na koja su bili raspoređeni. Obavlja se s pomoću parametarskog SQL upita stoga što tražimo podatke iz više vezanih tablica, a to u ovoj verziji razvojne okoline nije moguće automatski prikazati. Parametri koji su bitni za pretragu su novinar, snimatelj i tonski snimatelj unutar odabranog vremenskog perioda. Prema tome je izrađena i stranica za pretragu po zadanim kriterijima prikazana na slici 5.13.




*Slika 5.13: Izgled stranice za pretraživanje „Knjige snimanja“*

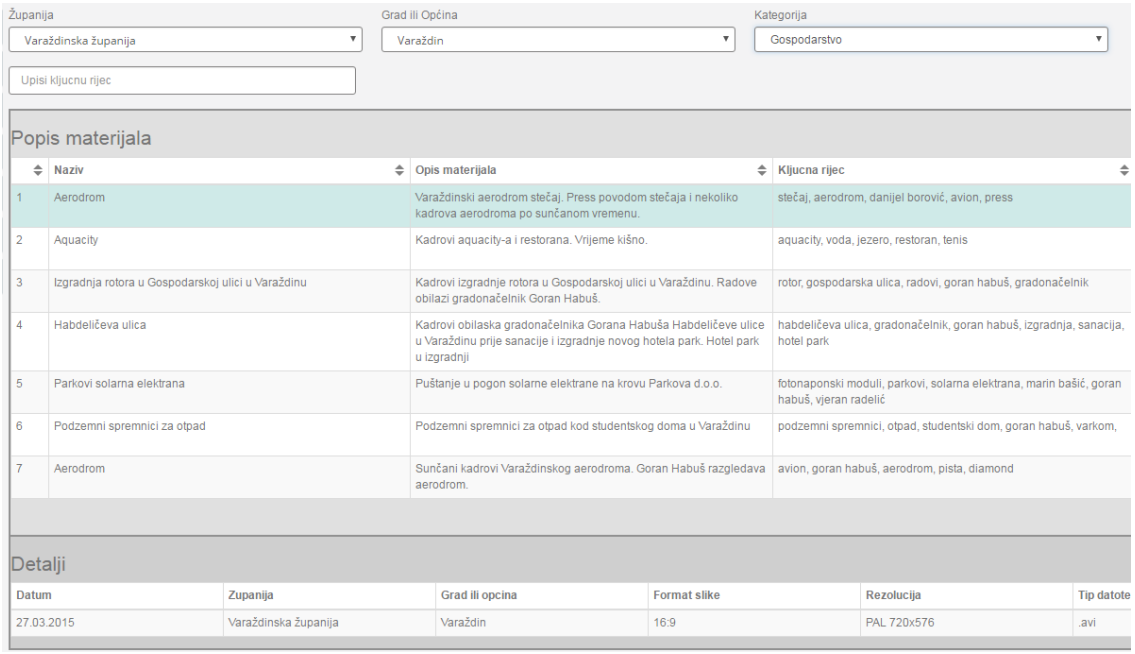
Kao što je već spomenuto, kreiranjem SQL upita stvara se servisna varijabla. Nužno je naglasiti da svaku varijablu možemo definirati kao varijablu stranice ili aplikacije. Varijabla stranice osvježi se svaki puta kada se dođe na tu stranicu, dok se varijabla aplikacije osvježi pokretanjem aplikacije. Kako je zamišljeno da *web*-aplikaciju koristi više korisnika odjednom, te je predviđeno da oni upisuju, modificiraju ili brišu podatke iz baze, nužno je sve varijable koje se koriste za prikaz podataka iz tablica definirati kao varijable stranice. SQL upit kojim dohvaćamo podatke iz „Knjige snimanja“ prikazan je na slici 5.14. Napomenimo još da se u sklopu svih stranica s pomoću kojih pretražujemo podatke u bazi pojavljuje element „datum“, koji nam omogućuje da odaberemo željeni datum za pretragu. Kako bi prikaz bio što pregledniji, pojedini elementi stranice kreirani su u različitim bojama, a sve s ciljem lakšeg uočavanja bitnih podataka.

```
SELECT t2.* FROM(
SELECT t.knigasId, t.ksNaziv, t.ksDatum, t.ksTime, g.`naziv` AS Grad, z.`naziv` AS Zupanija,
GROUP_CONCAT(Novinar) AS Novinar,
GROUP_CONCAT(idNovinar) AS idNovinar,
GROUP_CONCAT(Snimatelj) AS Snimatelj,
GROUP_CONCAT(idSnimatelj) AS idSnimatelj,
GROUP_CONCAT(Tonac) AS Tonac,
GROUP_CONCAT(idTonac) AS idTonac
  FROM(
SELECT ks.`id_knjigas` AS knigasId, ks.naziv AS ksNaziv, ks.datum AS ksDatum, ks.time AS ksTime,
ks.timeOD as timeOd, ks.timeDO as timeDo, ks.id_grad AS ksGrad, ks.id_zupanija AS ksZupanija,
CASE WHEN u.radno_mjesto = 2 THEN u.ime END AS Novinar,
CASE WHEN u.radno_mjesto = 2 THEN u.id_user END AS idNovinar,
CASE WHEN u.radno_mjesto = 3 THEN u.ime END AS Snimatelj,
CASE WHEN u.radno_mjesto = 3 THEN u.id_user END AS idSnimatelj,
CASE WHEN u.radno_mjesto = 4 THEN u.ime END AS Tonac,
CASE WHEN u.radno_mjesto = 4 THEN u.id_user END AS idTonac
 FROM knjigas ks
INNER JOIN `ksuser` ksu ON ksu.`id_knjigas` = ks.`id_knjigas`
INNER JOIN USER u ON u.`id_user` = ksu.`id_user`) t
INNER JOIN `grad` g ON g.`id_grad` = t.ksGrad
INNER JOIN `zupanija` z ON z.`id_zupanija` = t.ksZupanija
WHERE t.ksDatum >= :from AND t.ksDatum <= :to
GROUP BY t.knigasId) t2
WHERE t2.idNovinar = :idNovinarParam OR t2.idSnimatelj = :idSnimateljParam OR t2.idTonac = :idTonacParam
```

*Slika 5.14: SQL upit za pretragu „Knjige snimanja“*

## 5.5. **Izrada modula „Arhiva“**

Kod izrade dijela za prikaz arhive koristila se prazna tablica koja se spojila sa servisnom varijablom. Dizajn tablice promijenjen je kroz *.xml* datoteku (boja pozadine, font) kako bi se što čitljivije prikazali traženi podatci (slika 5.15). Kako bi korisnici što efikasnije pretraživali arhivu, kreiran je parametarski SQL upit koji omogućuje pretragu po županiji, gradu, kategoriji i ključnoj riječi. Nakon što korisnik odabere, odnosno označi željeni podatak, u donjem dijelu prikazuju se ostali podatci vezani uz videomaterijal (format slike, rezolucija, tip datoteke itd.). Opisani SQL upit prikazan je na slici 5.16.

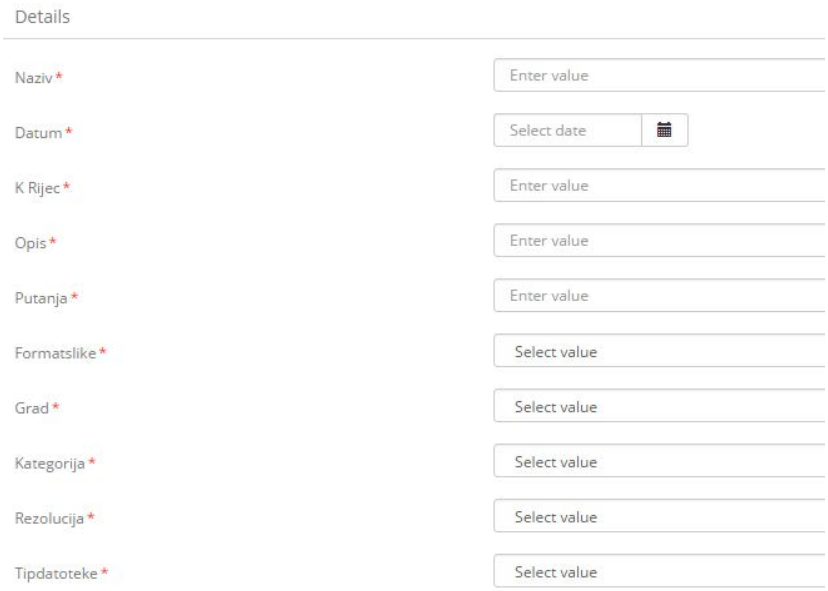



*Slika 5.15: Izgled stranice za pretragu „Arhive“*

```
select gra.naziv as gradNaziv, ma.opis, ma.k_rijec, ma.datum,
rez.naziv as rezNazaiv, td.naziv, ka.naziv as kaNaziv, ma.naziv as maNaziv,
ma.Putanja, fs.naziv as fsNaziv, zu.naziv as zuNaziv from materijal ma
inner join formatslike fs on fs.id_format_slike = ma.id_format_slike
inner join tipdatoteke td on td.id_tip_datoteke = ma.id_tip_datoteke
inner join rezolucija rez on rez.id_rezolucija = ma.id_rezolucija
inner join grad gra on gra.id_grad = ma.id_grad
inner join zupanija zu on zu.id_zupanija = gra.id_zupanija
inner join kategorija ka on ka.id_kategorija = ma.id_kategorija
where ma.k_rijec like :krijec
and zu.id_zupanija = :idZupanija
and gra.id_grad = :idGrad
and ka.id_kategorija = :idKategorija
```

*Slika 5.16: SQL upit za pretragu „Arhive“*

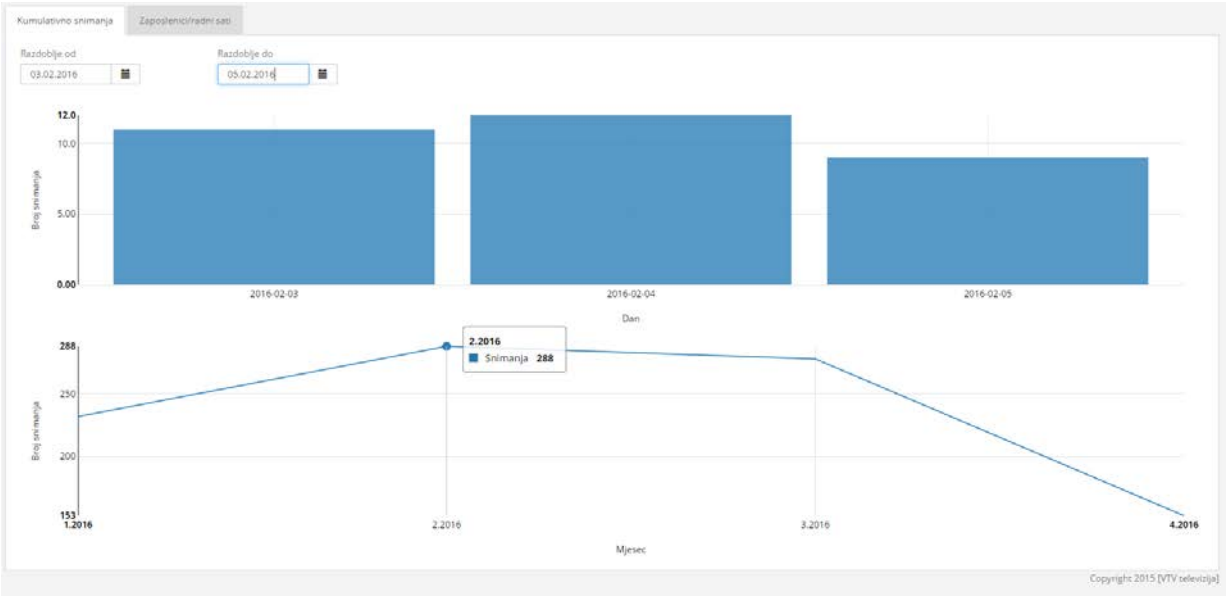
Napomenimo da razvojna aplikacija sadrži pogodnost automatski generiranih tablica (Widget), jednako kao i općenitih obrazaca za unos podataka u pojedine relacije (slika 5.16), što se događa pri samom uvozu baze podataka. Obrasci sadržavaju sva polja, odnosno sve atribute iz predmetne relacije, te vezanih relacija. Također je moguće prikazati i sakriti određene atribute u samom obrascu za unos podataka. Kao mogućnost nudi se promjena imena polja atributa kako bi bila što razumljivija korisnicima. Pojedina polja ovisno o sadržaju mogu poprimiti oblik padajućih izbornika. Vidljiva je crvena zvjezdica pokraj naziva atributa što označava da je taj atribut postavljen u bazi podataka s ograničenjem da ne smije biti prazan, odnosno ne smije imati vrijednost „*null*“ (engl. *constraint - not null*).




*Slika 5.17: Prikaz forme za unos podataka o videomaterijalu u „Arhivu“*

## 5.6. **Izrada modula „Statistika“**

Kako bi se prikazali statistički podatci vezani uz broj snimanja na godišnjoj razini ili u odabranom periodu, kreirala se stranica na kojoj se postavio trend-graf i grafikon s pomoću kojih se vizualiziraju navedeni podatci (slika 5.18). Također je prikazan efektivan broj radnih sati za sve sudionike snimanja na posebnoj stranici. Kako bi se mogli prikazati svi navedeni podatci, napisan je SQL upit koji se preko servisne varijable spaja s navedenim elementima postavljenim na stranicu.




*Slika 5.18: Prikaz statističkih podataka*

Napomenimo da je broj snimanja moguće prikazati grafikonom kroz željeni vremenski period odabiranjem početnog i završnog datuma. Uvjet za prikaz radnih sati zaposlenika su uredno popunjena vremena odlaska i dolaska sa snimanja u „Knjizi snimanja“.

```
select count(id_knjigas) as Snimanja, datum from knjigas
WHERE datum >= :from AND datum <= :to
group by datum
```

*Slika 5.19: SQL upit za broj snimanja u određenom vremenskom periodu*

```
select count(id_knjigas) as Snimanja,
concat(month(datum),'.', year(datum)) as mjgod from knjigas
group by mjgod
```

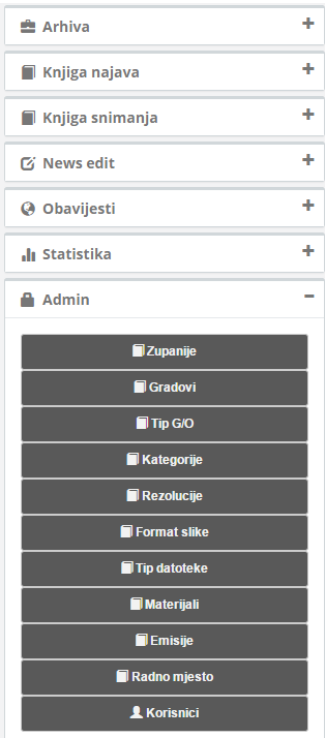
*Slika 5.20: SQL upit za dohvat broja snimanja na godišnjoj razini*

```
SELECT SEC_TO_TIME(SUM(TIME_TO_SEC(TIMEDIFF(timeDO,timeOD))))
AS ukupnoSati,`datum` FROM knjigas ks
INNER JOIN `ksuser` ksu ON ks.`id_knjigas`=ksu.`id_knjigas`
INNER JOIN USER u ON u.`id_user` = ksu.`id_user`
WHERE u.id_user = :id_user
AND datum BETWEEN :datumOD AND :datumDO
GROUP BY datum
```

*Slika 5.21: SQL upit za dohvat broja radnih sati provedenih na snimanju po odabranom zaposleniku*

## 5.7. **Izrada modula „Admin“**

Kako bi korisnik s administratorskim pravima mogao upravljati aplikacijom, u izbornik se dodalo „Admin“ područje koje sadržava sve tablice baze podataka koje nisu nužne ostalim korisnicima za rad. Kroz „Admin“ izbornik moguće je upisivati županije, gradove, tipove datoteka, rezolucije materijala, kategorije, korisničke podatke i drugo u bazu podataka. Napomenimo da je za svaki unos podataka kreirana posebna stranica s pripadajućom tablicom, odnosno ukoliko odaberemo iz izbornika „Zupanije“ aplikacija nas prosljeđuje na stranicu na kojoj se nalazi tablica „zupanija“ i omogućeno nam unos ili promjena podataka.




*Slika 5.22: Izgled administratorskog izbornika*

# 6. Rezultati primjene sustava u praksi

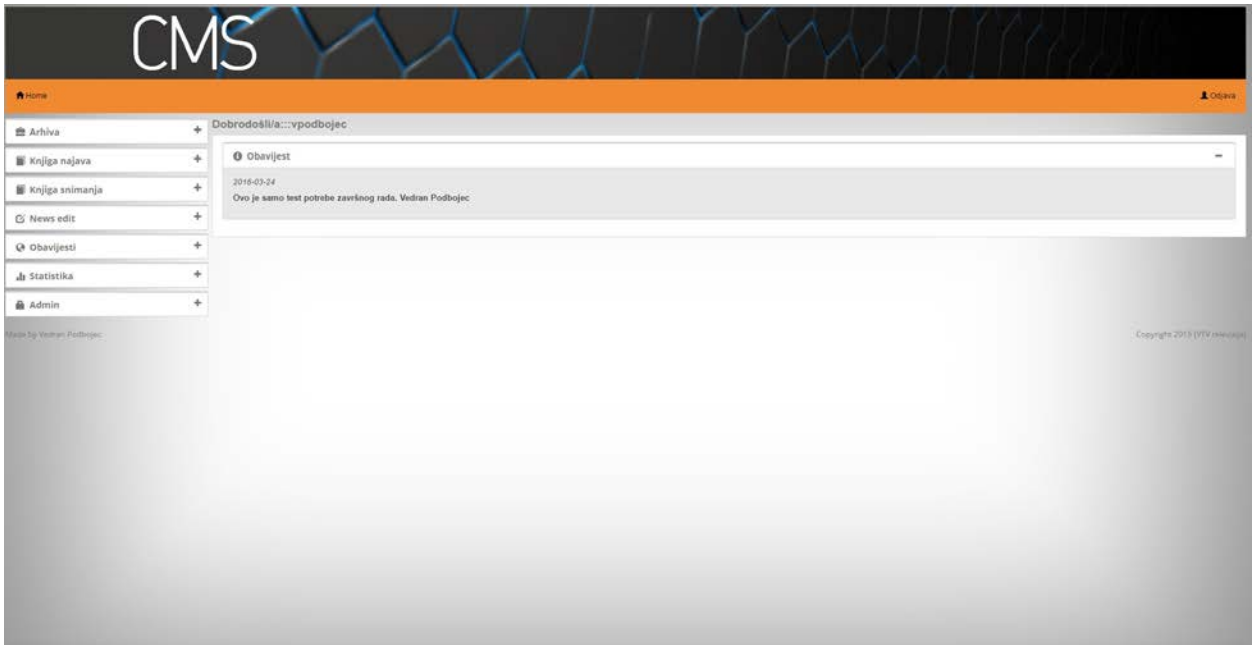
Razvijena aplikacija (početna stranica prikazana je na slici 5.23) tijekom korištenja od strane korisnika pokazala je nekoliko nedostataka.

Pojedini elementi stranice nisu se prikazivali, odnosno njihovo ponašanje nije bilo u skladu s programiranim značajkama (Angular JS). Također sam prijenos podataka iz „Knjige najava“ u „Knjigu snimanja“ nije radio kao što je to predviđeno u projektu. Daljnjom analizom došlo se do zaključka da određeni *web*-preglednici (Microsoft Internet Explorer i Mozilla Firefox) imaju problema pri prikazu navedenih segmenta predmetne *web*-aplikacije. Kao rješenje primijećenih problema za rad u *web*-aplikaciji odabran je *web*-preglednik Google Chrome u kojem ista radi bez poteškoća. Problem koji je ostao otvoren je lokalizacija stranice, odnosno automatski prijevod stranice na jezik korisnika koji pristupa *web*-aplikaciji. Analizom je utvrđeno da razvojna okolina u korištenoj verziji ne podržava taj segment.

Cilj izrade predmetne aplikacije bio je unaprijediti poslovne procese, što je u potpunosti postignuto. Korisnici u svakom trenutku mogu pristupiti *web*-aplikaciji unutar matične tvrtke, ili izvan nje te pregledavati ili upisati željene podatke. Napomenimo da kroz „Knjigu najava“ osobe zadužene za upravljanje ljudskim resursima mogu bolje predvidjeti te planirati raspored zaposlenika. Obavijesti koje se zaposlenicima šalju putem elektronske pošte pružaju uvid u detaljan raspored snimanja na koji su raspoređeni što za posljedicu ima bolju koordinaciju između matične tvrtke i zaposlenika na terenu. Kroz minimalnu prilagodbu aplikacija se može koristiti u organizacijama sa sličnim radnim procesima kao što je npr. radijska postaja, novinska agencija, medijska kuća ili u kreativnoj industriji. Nužno je da konceptualni dijagram baze podataka odgovara bazi podataka kreiranoj ovim radom.

Kao mogućnost nadogradnje *web*-aplikacije, predviđen je modul za editiranje teksta, te kreiranje hodograma emisija što će zaposlenicima pružiti jednostavnije i brže izvršavanje dnevnih obaveza, a tvrtku osloboditi od financijskih ulaganja u komercijalne programe za editiranje tekstualnog sadržaja. Također je predviđen modul u kojem će se pratiti izvršavanje kreiranih zadataka od strane odjela Marketinga prema odjelu Produkcije što je izuzetno bitno kako bi se klijentima tvrtke uvijek mogle pružiti valjane informaciju o trenutačnom stanju projekta.

Daljnji razvoj aplikacije moguć je jedino kroz razvojnu okolinu WaveMaker Studio Desktop 7.8. koja je besplatna za korištenje i može se preuzeti iz izvora na internetu [14]. Napomenimo da je proizvođač navedenog alata prekinuo s razvojem i podrškom besplatne inačice početkom 2016. godine stavljajući fokus na komercijalno rješenje u „oblaku“. Iako komercijalno rješenje ima integrirane module koji su u ovom radu razvijani ručno (i mnogo više), besplatna inačica alata nudi dovoljne mogućnosti za izradu spomenutih nadogradnji. Mnogobrojna zajednica korisnika navedenog alata pruža sigurnost u dostupnost informacija potrebnih za daljnji razvoj *web*-aplikacije.




*Slika 5.23: Izgled početne stranice web-aplikacije*

# 7. Zaključak

Nužnost sigurne i pouzdane pohrane podataka te istovremeno njihove brze i jednostavne dostupnosti od velike su važnosti za funkcioniranje suvremenih dinamičnih tvrtki. Kao rješenje nameću se sustavi koji su razvijeni s ciljem da korisnicima olakšaju pristup navedenim podatcima (CMS), a da od korisnika ne zahtijevaju napredno poznavanje informacijskih i komunikacijskih tehnologija.

Kako bi se podatci spremili, potrebno je razviti odgovarajući model baze podataka uz prethodnu sveobuhvatnu analizu radnih procesa. Nužnost analize vidljiva je pri samom razvoju konceptualnog modela baze podataka u kojem je bitno identificirati entitete, atribute i veze između relacija.

Grafički prikaz konceptualnog modela uvelike pridonosi razumijevanju strukture baze podataka što je nužno potrebno. Primjenjujući osnovna pravila prebacivanja konceptualnog modela baze podataka u relacijski model baze podataka osiguravamo se od mogućih dodatnih intervencija nakon kreiranja same baze podataka.

Primjenom alata za razvijanje Javinih *web* ili mobilnih aplikacija kao što su WaveMaker, AllcountJS, OpenXava ili Oracle ADF korisnicima s ograničenim znanjem Javina programskog jezika daje se mogućnost da razviju aplikacije za poslovna okruženja. Koje će se razvojno okruženje odabrati, uvelike ovisi o financijskim sredstvima te poznavanju suvremene tehnologije. Bitno je imati na umu da se odabere razvojno okruženje koje ima široku bazu korisnika te je dobro dokumentirano kako bismo se mogli bezrezervno osloniti na pouzdanu pretragu informacija nužnih za razvoj aplikacije.

Vedran Podbojec

U Varaždinu 7. rujna 2016.

# 8. Literatura


[1] https://hr.wikipedia.org/wiki/CMS, dostupno 27.08.2016.

[2] I. Horvat: Sustavi za upravljanje sadržajem, FER, Zagreb

[3] B. Košak, M. Tomiša, M. Čačić: Statičko i dinamičko upravljanje web sadržajem, *Tehnički glasnik 9*, 1(2015), str. 80

[4] http://www.wavemaker.com/, dostupno 11.10.2015.

[5] https://www.mysql.com/products/workbench/, dostupno 15.10.2015.

[6] https://www.mysql.com/, dostupno 09.09.2015

[7] http://tomcat.apache.org/, dostupno 12.11.2015.

[8] https://en.wikipedia.org/wiki/Peter_Chen, dostupno 01.09.2016.

[9] http://databases.about.com/od/specificproducts/l/blentity-relationship-diagrams.htm, dostupno 02.07.2016.

[10] http://www.znanje.org/abc/tutorials/accessMMX/01/Relacioni_model.html, dostupno 20.08.2016.

[11] https://de.wikipedia.org/wiki/Martin-Notation, dostupno 31.08.2016.

[12] Joseph M. Hellerstein, Michael Stonebraker (MIT): Readings in Database Systems, četvrto izdanje, 2005. godina

[13] http://www.wavemaker.com/rad/rapid-workflow-based-application-development/, dostupno 02.09.2016.

[14] https://plus.google.com/+BalaSakthi/posts/5qrDQyKt8S5 , dostupno 17.09.2016.

[15] D. Buytaert. „The history of MySQL AB“, www.buytaert.net/the-history-of-mysql-ab, dostupno 12.08.2016.

[16] https://www.java.com/en/about/, dostupno 16.08.2016.

[17] https://en.wikipedia.org/wiki/AngularJS, dostupno 17.09.2016.

# 9. Popis slika

# 10. Prilozi

## 10.1. **Javin programski kod korišten u izradi *web*-aplikacije:**

```
/*Generated by WaveMaker Studio*/

package com.vtvcms.knjigasnimanjaservice;

import com.wavemaker.runtime.service.annotations.ExposeToClient;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.vtvcms.cmsvtv.Grad;
import com.vtvcms.cmsvtv.Knjigas;
import com.vtvcms.cmsvtv.Najave;
import com.vtvcms.cmsvtv.Zupanija;
import com.vtvcms.cmsvtv.Ksemisije;
import com.vtvcms.cmsvtv.Emisije;
import com.vtvcms.cmsvtv.service.GradService;
import com.vtvcms.cmsvtv.service.ZupanijaService;
import com.vtvcms.cmsvtv.service.KnjigasService;
import com.vtvcms.cmsvtv.service.NajaveService;
import com.vtvcms.cmsvtv.service.EmisijeService;
import com.vtvcms.cmsvtv.service.KsemisijeService;
import com.vtvcms.cmsvtv.service.KsuserService;
import com.vtvcms.cmsvtv.service.UserService;
import com.vtvcms.cmsvtv.Ksuser;
import com.vtvcms.cmsvtv.User;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import java.util.Iterator;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.ArrayList;
import java.util.List;
import com.wavemaker.runtime.data.spring.WMPageImpl;

import org.hibernate.SessionFactory;
//import org.hibernate.Session;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import com.wavemaker.runtime.data.spring.WMPageImpl;
import org.springframework.transaction.annotation.Transactional;

import java.util.Properties;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
```

```
import javax.mail.internet.MimeMessage;

import java.util.Set;

/**
 * This is a singleton class with all of its public methods exposed
to the client via controller.
 * Their return values and parameters will be passed to the client
or
 * taken from the client respectively.
 */

@ExposeToClient

public class KnjigaSnimanjaService {

    private static final Logger
    logger=LoggerFactory.getLogger(KnjigaSnimanjaService.class);

@Autowired
private KnjigasService kService;
@Autowired
private GradService gradService;
@Autowired
private ZupanijaService zupanijaService;
@Autowired
private NajaveService najavaService;
@Autowired
private KsemisijeService ksemisijeService;
@Autowired
private EmisijeService emisijeService;
@Autowired
private KsuserService ksuserService;
@Autowired
private UserService userService;
@Autowired
private SessionFactory sessionFactory;




public String createKnjigaSnimanja(String naziv, String napomena,
String timeSnimanja, String timePolazak, String timePovratak,
String dateSnimanja, String idGrad, String idZupanija, String
idNajave, String snimatelji, String tonci, String novinari, String
emisije) {
        String result = null;

try {

//Session session = sessionFactory.getCurrentSession();

logger.debug("Starting create knjiga snimanja");
            Knjigas knjigaSnimanja = new Knjigas();
            DateFormat format = new SimpleDateFormat("yyyy-M-d");
            Date datumSnimanja = format.parse(dateSnimanja);
```

```
            DateFormat formatTime = new
SimpleDateFormat("HH:mm:ss");
            Date vrijemeSnimanja = formatTime.parse(timeSnimanja);

            DateFormat formatTime1 = new
SimpleDateFormat("HH:mm:ss");
            Date vrijemePolaska = formatTime1.parse(timePolazak);

            DateFormat formatTime2 = new
SimpleDateFormat("HH:mm:ss");
            Date vrijemePovratak = formatTime2.parse(timePovratak);

            knjigaSnimanja.setDatum(datumSnimanja);
            knjigaSnimanja.setTime(vrijemeSnimanja);
            knjigaSnimanja.setTimeOd(vrijemePolaska);
            knjigaSnimanja.setTimeDo(vrijemePovratak);
            knjigaSnimanja.setNapomena(napomena);
            knjigaSnimanja.setNaziv(naziv);

//Grad
            Grad grad = new Grad();
            grad = gradService.findById(new Integer(idGrad));

//Zupanija
            Zupanija zupanija = new Zupanija();
            zupanija = zupanijaService.findById(new
Integer(idZupanija));

//set grad i zupanija
            knjigaSnimanja.setGrad(grad);
            knjigaSnimanja.setZupanija(zupanija);

//Get najava
            Najave najava = new Najave();
            najava = najavaService.findById(new Integer(idNajave));
            knjigaSnimanja.setNajave(najava);

//Save knjiga snimanja
            knjigaSnimanja = kService.create(knjigaSnimanja);

            String zaEmisije = "";

            String novinariSnimanja = "";
            String tonciSnimanja = "";
            String snimateljiSnimanja = "";

            List<SudionikSnimanja> sudioniciSnimanja = new
ArrayList<SudionikSnimanja>();

            if(emisije != null && !emisije.isEmpty()){
                String[] aemisije = emisije.split(",");
                for(int i= 0; i< aemisije.length; i++){
                Ksemisije ksemisije = new Ksemisije();
                Emisije emisijeE = emisijeService.findById(new
Integer(aemisije[i]));
                ksemisije.setKnjigas(knjigaSnimanja);
                ksemisije.setEmisije(emisijeE);
```

```
                ksemisijeService.create(ksemisije);
                zaEmisije += emisijeE.getNaziv() + ", ";
                }
            }
            zaEmisije = zaEmisije.substring(0, zaEmisije.length() -
2);


//Save snimatelji
     if(snimatelji != null && !snimatelji.isEmpty()){

                String[] ausers = snimatelji.split(",");
                for(int i= 0; i< ausers.length; i++){
                Ksuser ksuser = new Ksuser();
                User user = userService.findById(new
                Integer(ausers[i]));
                ksuser.setKnjigas(knjigaSnimanja);
                ksuser.setUser(user);
                ksuserService.create(ksuser);

//Send mail to snimatelj
                String osobaNaziv = user.getIme() + " " +
                user.getPrezime();

                SudionikSnimanja sudionik = new SudionikSnimanja();
                sudionik.setEmail(user.getMail());
                sudionik.setNaziv(osobaNaziv);
                sudioniciSnimanja.add(sudionik);

                snimateljiSnimanja += osobaNaziv + ", ";
                }
            }

//Save novinari
     if(novinari != null && !novinari.isEmpty()){
                String[] ausers = novinari.split(",");
                for(int i= 0; i< ausers.length; i++){
                Ksuser ksuser = new Ksuser();
                User user = userService.findById(new
                Integer(ausers[i]));
                ksuser.setKnjigas(knjigaSnimanja);
                ksuser.setUser(user);
                ksuserService.create(ksuser);

//Send mail to novinar
                String osobaNaziv = user.getIme() + " " +
                user.getPrezime();

                SudionikSnimanja sudionik = new SudionikSnimanja();
                sudionik.setEmail(user.getMail());
                sudionik.setNaziv(osobaNaziv);
                sudioniciSnimanja.add(sudionik);

                novinariSnimanja += osobaNaziv + ", ";
                }
            }
```

```
//Save tonski snimatelj
     if(tonci != null && !tonci.isEmpty()){
                String[] ausers = tonci.split(",");
                for(int i= 0; i< ausers.length; i++){
                Ksuser ksuser = new Ksuser();
                User user = userService.findById(new
                Integer(ausers[i]));
                ksuser.setKnjigas(knjigaSnimanja);
                ksuser.setUser(user);
                ksuserService.create(ksuser);

//Send mail to tonski snimatelji
                String osobaNaziv = user.getIme() + " " +
                user.getPrezime();

                SudionikSnimanja sudionik = new SudionikSnimanja();
                sudionik.setEmail(user.getMail());
                sudionik.setNaziv(osobaNaziv);
                sudioniciSnimanja.add(sudionik);

                tonciSnimanja += osobaNaziv + ", ";
                }
           }

if(snimateljiSnimanja != null && !snimateljiSnimanja.isEmpty())
                snimateljiSnimanja =
                snimateljiSnimanja.substring(0,
                snimateljiSnimanja.length() - 2);
if(novinariSnimanja != null && !novinariSnimanja.isEmpty())
                novinariSnimanja = novinariSnimanja.substring(0,
                novinariSnimanja.length() - 2);
if(tonciSnimanja != null && !tonciSnimanja.isEmpty())
                tonciSnimanja = tonciSnimanja.substring(0,
                tonciSnimanja.length() - 2);

for(SudionikSnimanja sudionik : sudioniciSnimanja){
                sendMail(sudionik.getEmail(), sudionik.getNaziv(),
                grad.getNaziv(), knjigaSnimanja.getNaziv(),
                zaEmisije, timeSnimanja, datumSnimanja,
                timePolazak, timePovratak, napomena,
                novinariSnimanja, tonciSnimanja,
                snimateljiSnimanja);
           }


// Slanje maila ako je emisija u STUDIO

if(grad.getIdGrad() != null && grad.getIdGrad() == 113){

                sendMail("mail1", "John Doo", grad.getNaziv(),
                knjigaSnimanja.getNaziv(), zaEmisije, timeSnimanja,
                datumSnimanja, timePolazak, timePovratak, napomena,
                novinariSnimanja, tonciSnimanja,
                snimateljiSnimanja);

                sendMail("mail2", "John Doo", grad.getNaziv(),
                knjigaSnimanja.getNaziv(), zaEmisije, timeSnimanja,
```

```
                datumSnimanja, timePolazak, timePovratak, napomena,
                novinariSnimanja, tonciSnimanja,
                snimateljiSnimanja);

                sendMail("mail3", "John Doo", grad.getNaziv(),
                knjigaSnimanja.getNaziv(), zaEmisije, timeSnimanja,
                datumSnimanja, timePolazak, timePovratak, napomena,
                novinariSnimanja, tonciSnimanja,
                snimateljiSnimanja);

                sendMail("mail4", "John Doo", grad.getNaziv(),
                knjigaSnimanja.getNaziv(), zaEmisije, timeSnimanja,
                datumSnimanja, timePolazak, timePovratak, napomena,
                novinariSnimanja, tonciSnimanja,
                snimateljiSnimanja);

            }


            //Update najava connection to knjiga snimanja
            najava.setKnjigas(true);
            najavaService.update(najava);

            logger.debug("Returning {}", result);
            return result;
        }catch(java.text.ParseException pe){
           pe.printStackTrace();
           logger.error("Sample java service operation has failed",
pe);
        } catch (Exception e) {
            logger.error("Sample java service operation has
failed", e);
            throw e;
        }
        return result;
    }

        //Update najava connection from knjiga snimanja on delete

    public String deleteKnjigaSnimanja(String id){
      Knjigas knjigas = kService.findById(new Integer(id));
      Najave najava =
najavaService.findById(knjigas.getNajave().getIdNajave());
      najava.setKnjigas(false);
      najavaService.update(najava);
     kService.delete(new Integer(id));

      return null;
    }

    //Slanje maila kad si upisan u knjigu snimanja

    public String sendMail(String toEmailAddress, String
osobaNaziv, String mjestoNaziv, String nazivSnimanja, String
popisEmisija, String timeSnimanja, Date datumSnimanja, String
vrijemeOd, String vrijemeDo, String napomena, String novinari,
String tonci, String snimatelji){
```

```
        try{

            // Use javamail api, set parameters from
registration.properties file
            // set the session properties
            Properties props = System.getProperties();
            //props.put("mail.smtp.starttls.enable", "true");
            props.put("mail.smtp.host", "smtp.*****.hr");
            props.put("mail.smtp.port", "25");
            //props.put("mail.smtp.auth", "true");
            //props.put("mail.smtp.starttls.required", "true");
            Session session = Session.getDefaultInstance(props,
null);

            String mailBodyMessage = "Postovani/a "+ osobaNaziv
+",\n\n" +
            "upisani ste za snimanje " + nazivSnimanja + " u mjestu "
+ mjestoNaziv + " na dan " + formateDateOnly(datumSnimanja) + " u "
+ timeSnimanja +" sati, za emisije " + popisEmisija + ". " +
            "\n\n" +
            "DETALJI SNIMANJA : " +
            "\n\n" +
            "Naziv snimanja : " + nazivSnimanja +
            "\n\n" +
            "Datum snimanja : " + formateDateOnly(datumSnimanja) +
            "\n\n" +
            "Mjesto snimanja : " + mjestoNaziv +
            "\n\n" +
            "Vrijeme pocetka snimanja : " + timeSnimanja +
            "\n\n" +
            "Predvideno vrijeme polaska na snimanje : " + vrijemeOd +
            "\n\n" +
            "Predvideno vrijeme dolaska sa snimanja : " + vrijemeDo +
            "\n\n" +
            "Novinar : " + novinari +
            "\n\n" +
            "Snimatelj : " + snimatelji +
            "\n\n" +
            "Tonski snimatelj : " + tonci +
            "\n\n" +
            "Napomena : " + napomena +
            "\n\n" +
            "\n\n" +
            "S postovanjem, Vasa knjiga snimanja. ";

            // Create email message
            MimeMessage message = new MimeMessage(session);
            message.setFrom(new
InternetAddress("knjiga.snimanja@****.hr"));
            message.addRecipient(Message.RecipientType.TO, new
InternetAddress(toEmailAddress));
            message.setSubject("Dodani ste za snimanje");
            message.setText(mailBodyMessage);

            // Send message
                    Transport.send(message);
```

```
            return "Mail sent successfully.";

        } catch (MessagingException e) {
            logger.error("Error in method sendEmailNotification: ",
e);
            return "Error in method sendEmailNotification: " + e;
        }
    }


    private String formateDateOnly(Date datum){
            DateFormat dateFormat = new
SimpleDateFormat("dd.MM.yyyy");
            return dateFormat.format(datum);
     }

}
```

Sveučilište
Sjever

**IZJAVA O AUTORSTVU**
**I**
**SUGLASNOST ZA JAVNU OBJAVU**

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, **Vedran Podbojec** pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor završnog rada pod naslovom **Oblikovanje web-sustava za upravljanje podatcima** te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

**Student:**
Vedran Podbojec

(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, **Vedran Podbojec** neopozivo izjavljujem da sam suglasan s javnom objavom završnog rada pod naslovom **Oblikovanje web-sustava za upravljanje podatcima** čiji sam autor.

**Student:**
Vedran Podbojec

(vlastoručni potpis)