Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jošt Lajovec

Program za beleženje rezultatov v curlingu

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Erik Štrumbelj

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Glavni cilj te naloge je razviti program, ki bo omogočal beleženje rezultatov v curlingu. Minimalna potrebna funkcionalnost obsega beleženje podatkov o aktivnih ekipah, tekmovanjih in posameznih dvobojih, pri čemer je potrebno upoštevati specifične lastnosti športa curling. Študent mora izbrati ustrezne tehnologije, zasnovati in izdelati podatkovno bazo ter izdelati uporabniški vmesnik.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Jošt Lajovec, z vpisno številko **63110237**, sem avtor diplomskega dela z naslovom:

Program za beleženje rezultatov v curlingu

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Erika Štrumblja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 10. septembra 2014 Podpis avtorja:

Hvala mentorju, doc. dr. Eriku Štrumblju, za strokovno pomoč ter potrpežljivost tekom izdelave diplomskega dela. Hvala mojim najbližjim za vse ostalo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application programming interface	programski vmesnik
CRUD	create, read, update, delete	ustvarjanje, poizvedba, posodobitev, izbriši
CZS	Slovenian Curling Association	Curling zveza Slovenije
JDBC	Java Database Connectivity	/
JPA	Java Persistence API	/
MVC	Model-View-Controller	model-pogled-kontroler
PHP	Hypertext Preprocessor	/
SQL	Structured Query Language	strukturirani povpraševalni jezik
UML	Unified Modeling Language	/
WCF	World Curling Federation	Svetovna curling zveza

Povzetek

Curling je ekipni zimski olimpijski šport, ki je pri nas prisoten od leta 2010. Beleženje rezultatov tekmovanj pod okriljem Curling zveze Slovenije poteka ročno na liste papirja. V sklopu diplomskega dela je bila razvita podatkovna baza, ki omogoča hranjenje rezultatov v elektronski obliki. Poleg podatkovne baze je bila razvita tudi aplikacija, ki članom Tekmovalne komisije Curling zveze Slovenije omogoča enostavno shranjevanje rezultatov v podatkovno bazo in upravljanje z obstoječimi rezultati. Aplikacija ostalim uporabnikom omogoča brskanje po rezultatih preteklih tekmovanj ter iskanje rezultatov ekip, tekmovalcev in klubov na podlagi želenih kriterijev.

Ključne besede: curling, športni rezultati, beleženje.

Abstract

Curling is a Winter Olympic team sport, which has been present in Slovenia since 2010. Recording the results of competitions under the auspices of the Slovenian curling association is done by hand on sheets of paper. As a part of the thesis, a database for storage of results in electronic form was developed. In addition to the database, an application that allows memebers of the Competition Commission of Slovenian curling associaton simple storing of results to the database and managing existing results was also developed. The application offers users browsing for results of past competitions and searching for results of teams, players and clubs on the basis of their own criteria.

Keywords: curling, sport results, recording.

Poglavje 1

Uvod

Curling je ekipni zimski olimpijski šport, katerega začetki segajo v 16. stoletje. Pri nas se je curling pojavil leta 2010, ko je bila ustanovljena Curling zveza Slovenije. Od takrat je bilo pod njenim okriljem organiziranih že več kot 25 tekmovanj v različnih kategorijah, ki se jih je udeležilo približno 200 različnih tekmovalcev. Beleženje rezultatov tekmovanj in tekem poteka ročno na liste papirja, ki se jih je tekom let nabralo za kar nekaj fasciklov, katerih hranjenje zavzame veliko prostora. Tekmovalci pogosto sprašujejo po rezultatih tekmovanj iz preteklosti, na kar jim v večini primerov iz glave nihče ne zna ponuditi odgovora. V sklopu diplomskega dela bom razvil aplikacijo, ki bo omogočala enostavno beleženje in shranjevanje rezultatov, na drugi strani pa bo vsem, ki se bodo želeli pozanimati o rezultatih in statistikah omogočila pregledovanje le teh.

V samem začetku dela je bilo potrebno razviti podatkovni model, ki zadošča shranjevanju rezultatov na podlagi tekmovalnega sistema in pravil CZS. Sledila je implementacija podatkovne baze tipa MySQL. V zadnji fazi sem razvil spletno aplikacijo v tehnologiji Java Server Faces, ki mi omogoča vnašanje rezultatov, poljubnemu uporabniku pa brskanje po le teh.

Pričakujem, da bo razvita aplikacija močno olajšala opravljanje mojih dolžnosti v sklopu Tekmovalne komisije CZS, hkrati pa večini uporabnikov ponudila odgovore na vprašanja glede rezultatov in statistike curling tekmo-

vanj v Sloveniji.

V Poglavju 2 na kratko predstavimo večini ljudi slabo poznan šport curling ter se dotaknemo trenutne realizacije beleženja in shranjevanja rezultatov v Curling zvezi Slovenije. Poglavje 3 opisuje postopek načrtovanja in implementacije podatkovne baze, ki je predstavljal najpomembnejši del diplomske naloge. V Poglavju 4 se seznanimo z uporabljenimi tehnologijami ter opišemo aplikacijo, ki smo jo razvili v sklopu diplomskega dela. Sledita še diskusija o možnih izboljšavah ter zaključek.

Poglavje 2

Curling

Curling je moštveni šport, pri katerem igralci podajajo kamne po ledeni površini proti ciljni površini, ki ji pravimo hiša. Hiša je razdeljena na štiri koncentrične kroge. Kamni so narejeni iz poliranega granita in tehtajo malo manj kot dvajset kilogramov. Vsaka izmed dveh ekipa ima osem kamnov. Cilj ekipe je osvojiti večje število točk od nasprotne ekipe. Tekma je sestavljena iz iger, vsaka igra se konča, ko obe ekipi vržeta vseh svojih 8 kamnov. Točke se štejejo po koncu vsake igre. Ekipa, ki ima ob koncu posamezne igre kamen najbližje centru hiše, osvoji točko za vsak svoj kamen v hiši, ki leži bližje centru hiše kot najbližji nasprotnikov kamen. Igralec ob izpustu kamen nekoliko zavrti, posledica tega pa je, da kamni nikoli ne drsijo naravnost, temveč zmeraj zavijajo v smeri rotacije. S pometanjem lahko podaljšamo dolžino samega meta ter nekoliko kontroliramo jakost zavijanja kamna. Pri curlingu na najvišjem nivoju pomembno vlogo igra strategija postavljanja kamnov, zato mu mnogi pravijo tudi šah na ledu.

2.1 Pravila

Na curling tekmi se pomerita dve ekipi. Ekipo navadno sestavljajo štirje igralci. Pred vsako tekmo se z metom kovanca ali na kakšen drug način določi, katera ekipa bo imela v prvi igri(angl. end) prednost zadnjega kamna (angl.

hammer). V sklopu ene igre ekipi izmenično podajata kamne. Vsaka ekipa poda osem kamnov, vsak igralec dva. Igralec vedno podaja dva zaporedna kamna. Vrstni red, v katerem ekipa podaja kamne, se navede pred začetkom tekme in se med samo tekmo ne sme spreminjati. Vsaka ekipa pred tekmo določi vodjo (angl. skip), ki je zadolžen za določanje taktike in navadno meče zadnjega dva kamna. Ostali igralci izmenično mečejo in pometajo. Ko ekipi podata vseh 16 kamnov, se igra zaključi in zabeleži se rezultat te igre. Ekipa, ki v tej igri ni osvojila točk, ima v naslednji igri prednost zadnjega kamna. V primeru, da nobena ekipa ne osvoji točk, prednost zadnjega kamna ohrani ekipa, ki ga je imela v zadnji odigrani igri. Končni rezultat posamezne ekipe je vsota njenih točk, ki jih je dosegla v vseh igrah. V primeru neodločenega rezultata se odigra dodatna igra. Na najvišjem nivoju so tekme sestavljene iz desetih iger. Pri nas so tekme zaradi omejenosti s časom krajše, in so navadno dolge šest do osem iger, pri mladinskih kategorijah tudi manj. Poleg opisane različice curlinga, ki je olimpijska disciplina, obstajajo tudi druge kategorije. Omeniti velja kategorijo mešanih dvojic (angl. mixed doubles), pri kateri ekipo sestavljata en moški in ena ženska tekmovalka. Pri mešanih dvojicah ekipa v okviru ene igre poda le 5 kamnov.

2.2 Beleženje rezultatov in statistike

Kakor vse delo v CZS, tudi beleženje rezultatov tekmovanj bazira na prostovoljstvu tekmovalcev, tako da smo včasih pri nekaterih nalogah nekoliko omejeni. Na tekmovanjih najvišjega ranga pod okriljem WCF je za vsako tekmo nekdo zadolžen za beleženje učinkovitosti posameznih igralcev. Ker se na tekmovanjih običajno igra po pet tekem naenkrat, bi za takšno beleženje potrebovali pet oseb, česar pa si pri nas žal še ne moremo privoščiti. Vseeno pa je beleženje opravljeno za vsako tekmo, kolikor se le da natančno. Tekmovalna komisija za vsako tekmo na list papirja zabeleži datum ter lokacijo tekme, fazo tekmovanja, vrstni red igralcev znotraj ekipe, rezultat posameznih iger in skupni rezultat.

2.2.1 Konkuri

CZS za objavo rezultatov na svetovnem spletu trenutno uporablja spletno aplikacijo Konkuri. Gre za aplikacijo, razvito s strani italijanske spletne agencije Koinema. Aplikacija je primarno namenjena organiziranju tekmovalnega sistema za tekmovanje in beleženju rezultatov ter rangiranju ekip tekom samega tekmovanja. Posledično je uporabniška izkušnja pri iskanju rezultatov končanega tekmovanja precej nezadovoljiva. Vse, kar o tekmovanju lahko izvemo, je končni vrstni red ekip, končni rezultati posameznih tekem ter podatek, v kateri fazi tekmovanja je bila tekma odigrana. Prikrajšani smo za podatke o članih ekip, ki so tekmovale na tekmovanju, za podrobnejše rezultate tekem, za podatek, kateri klub je zastopala ekipa, in še marsikaj drugega.

2.2.2 WCF Rezultati

Svetovna curling zveza je na tem področju nekoliko bolj napredna. Beležijo rezultate večjih tekovanj (od olimpijskih iger pa do evropskih prvenstev skupine C), ki so nato objavljeni na svetovnem spletu. Njihova aplikacija omogoča iskanje posameznih oseb in nam nato prikaže uvrstitve ter odigrane tekme izbrane osebe. Izpostaviti velja slabost, da za odigrane tekme lahko izvemo le končni rezultat, informacija o razvoju tekme skozi posamezne igre pa nam ni dostopna. Aplikacija omogoča tudi opravljanje zanimivejših poizvedb, kot je npr. iskanje najtrofejnešega igralca na svetu ali v posamezni državi. Nekatere izmed funkcionalnosti aplikacije WCF nam bodo služile kot zgled tekom razvoja naše aplikacije.

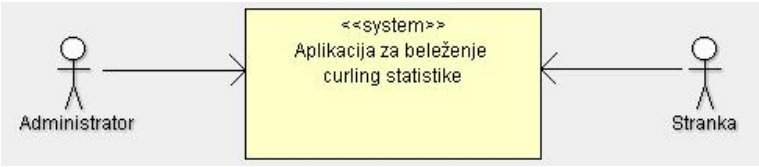
Poglavje 3

Razvoj podatkovne baze

3.1 Primeri uporabe

Na začetku razvoja vsakega sistema oziroma aplikacije je ključnega pomena, da popolnoma razumemo problem, ki ga bo reševala naša aplikacija. Čeprav se to zdi več kot očitno, je presenetljivo, kako pogosto programerji implementirajo podatkovno bazo, preden se seznanijo s podrobnostimi problema, ki ga bo aplikacija na podlagi podatkovne baze reševala [3]. Eden izmed načinov za utrjevanje našega razumevanja problema je uporaba t.i. primerov uporabe. Primeri uporabe so opis načinov, na katere so uporabniki v interakciji s sistemom, ki ga želimo razviti. Primere uporabe razvijalec sistema oziroma aplikacije napiše ob pomoči naročnika aplikacije. Pri pisanju primerov uporabe sem se zanašal na lastno poznavanje tematike in izkušnje pri beleženju rezultatov. Posvetoval sem se tudi s člani Tekmovalne komisije CZS.

V sklopu pisanja primerov uporabe je najprej potrebno določiti uporabnike aplikacije, ki jih formalno poimenujemo akterji. Prvi akter bo administrator - član Tekmovalne komisije CZS, ki bo skrbel za vnašanje rezultatov tekmovanj. Drugi uporabnik aplikacije bodo obiskovalci spletne aplikacije, ki se bodo želeli pozanimati o rezultatih curling tekmovanj pod okriljem CZS. Ker bodo koristili usluge, ki jih bo ponujala naša aplikacija, jih poimenujmo stranke. Tako imamo določena dva akterja, ki bosta uporabljala našo aplik-



Slika 3.1: Akterja primerov uporabe.

cijo.

"Kaj počnejo akterji?"je vprašanje, na katerega je potrebno odgovoriti, če želimo napisati dobre primere uporabe. Če le na hitro pogledamo arhiv CZS, v katerem se hranijo rezultati tekmovanj iz preteklosti, hitro ugotovimo, da je hranjenje rezultatov na papirju potekalo vedno na enak način. Tekmovalna komisija je po vsakem tekmovanju v nov fascikel shranila podatke o tekmovanju, ekipah, končni vrstni red uvrstitev ekip ter podatke in rezultate o posameznih tekmah. Torej bo naloga administratorja shranjevanje in vzdrževanje podatkov o tekovanjih, tekmah, ekipah in tekmovalcih. Da bo lahko opravljal vse navedene storitve, se bo seveda moral najprej prijaviti v sistem. Na drugi strani bodo obiskovalci naše spletne aplikacije oziroma t.i. stranke želeli brskati po rezultatih posameznih tekmovanj, ekip in tekmovalcev. Najbolj pregledno je, da primere uporabe predstavimo kot seznam in jih na kratko opišemo oziroma da jih predstavimo s t.i. diagramom primerov uporab.

Prijava

Prijava v aplikacijo z uporabniškim imenom in geslom.

Vzdrževanje podatkov o tekmovanjih

Vnašanje, urejanje, pregledovanje ter brisanje podatkov o tekmovanjih.

Vzdrževanje podatkov o ekipah

Vnašanje, urejanje, pregledovanje ter brisanje podatkov o ekipah.

Vzdrževanje podatkov o igralcih

Vnašanje, urejanje, pregledovanje ter brisanje podatkov o igralcih.

Slika 3.2: Diagram primerov uporabe.

Vzdrževanje podatkov o tekmah

Vnašanje, urejanje, pregledovanje ter brisanje podatkov o tekmah.

Pregledovanje rezultatov in statistike

Pregledovanje rezultatov tekmovanj, tekem, ekip, tekmovalcev. Iskanje rezultatov ter statistike glede na izbrane kriterije, kot so datum tekmovanja in tekme, kategorija tekmovanja, spol tekmovalca...

3.2 Diagram poteka

Vsak primer uporabe lahko predstavimo tudi s t.i. diagramom poteka. Diagram poteka je predstavljen v jeziku UML in prikazuje interakcije med objekti v določenem vrstnem redu [2]. Na sliki 3.3 je prikazan diagram poteka za dodajanje tekme. Ob dodajanju nove tekme mora aplikacija uporabniku ponuditi obstoječa tekmovanja. Ob izbiri tekmovanja se ponudijo ekipe, ki so se udeležile tega tekmovanja. Uporabnik določi fazo tekmovanja, datum ter čas tekme in tip tekme. Ko izbere prvo ekipo, mu aplikacija ponudi postavo, ki jo je ekipa prijavila na tekmovanje. Uporabnik lahko postavo po želji spremeni. Isti postopek se ponovi ob izbiri druge ekipe. Uporabnik določi še ekipo, ki je imela prednost zadnjega kamna, vpiše rezultat posameznih iger ter skupni rezultat in doda komentar. V primeru, da je uporabnik pravilno izpolnil vsa polja, se tekma shrani v podatkovno bazo, uporabnik pa je obveščen o uspešnosti transakcije. Če vsa polja niso pravilno izpolnjena, je uporabnik o tem obveščen. Ko popravi vse napake se tekma shrani v podatkovno bazo in o tem obvesti uporabnika.

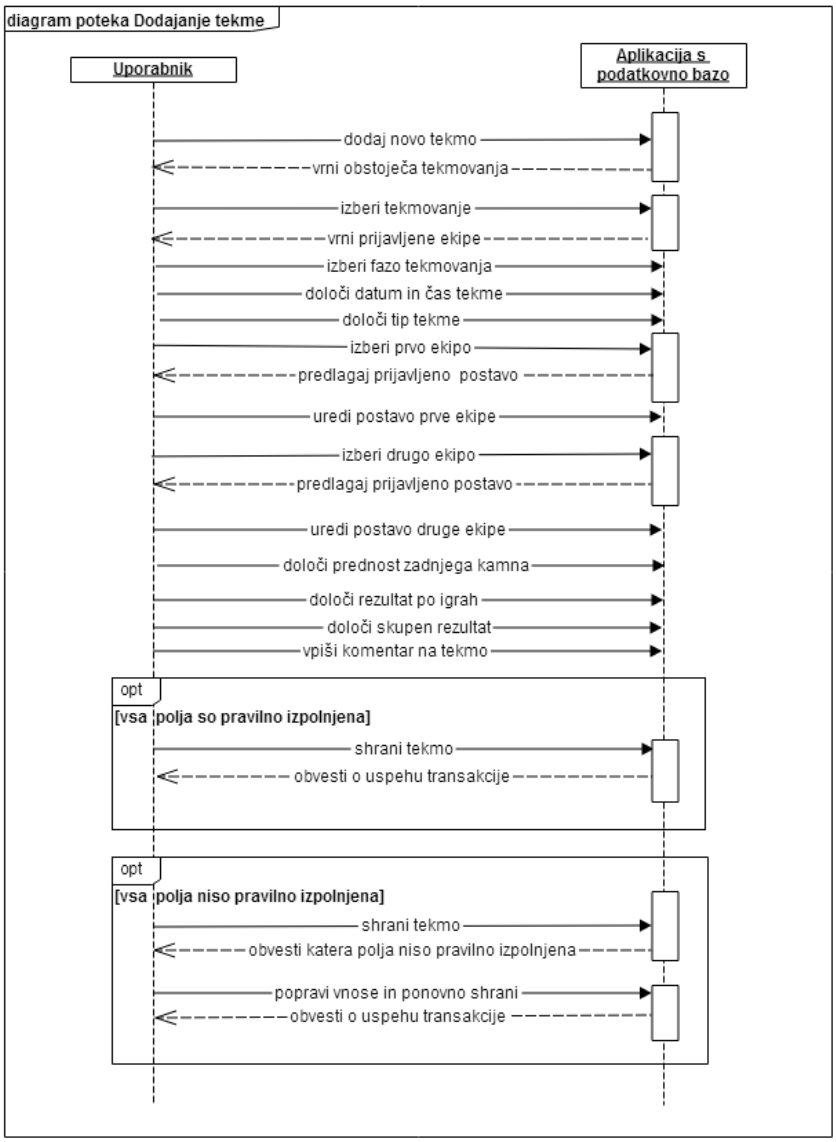
3.3 Konceptualni, logični in fizični podatkovni model

Načrtovanje podatkovne baze je navadno sestavljeno iz treh nivojev. Rezultat vsakega izmed nivojev je svoj podatkovni model [4].

Nivo konceptualnega načrtovanja , katerega rezultat je semantični oz. konceptualni podatkovni model.

Nivo logičnega načrtovanja , katerega rezultat je logični podatkovni model.

Nivo kreiranja fizične podatkovne baze , katerega rezultat je fizični podatkovni model oz. fizična podatkovna baza.



Slika 3.3: Diagram poteka za dodajanje tekme.

3.3.1 Konceptualni podatkovni model

Konceptualni podatkovni model definira glavne entitete in povezave med njimi. Za določitev entitet in povezav med njimi, si podrobneje poglejmo primer, ki ga je potrebno rešiti.

Tekmovanje, ki ga razpiše Tekmovalna komisija CZS, poteka na natanko eni lokaciji in ima točno določen datum začetka ter datum konca. Tekmovanje spada v natanko eno kategorijo. Na tekmovanju je odigrano poljubno število tekem.

Ekipa na tekmovanju zastopa natanko en klub in doseže natanko eno končno uvrstitev. Ob prijavi na tekmovanje ekipa navede dva do pet igralcev (odvisno od kategorije, možnost prijave menjave), ter določi njihov vrstni red igranja. Opcijsko lahko ekipa ob prijavi na tekmovanje navede tudi trenerja. Na različnih tekmovanjih tekom sezone lahko ekipa prijavi različne igralce ter trenerje. Prav tako lahko ekipa na različnih tekmovanjih zastopa različne klube.

Tekma je odigrana v okviru enega tekmovanja v natanko eni fazi tekmovanja. Za vsako tekmo vsaka ekipa ponovno navede vrstni red igralcev, ki bodo igrali na tej tekmi. Vrstni red igralcev je lahko drugačen od tistega, ki so ga podali ob prijavi na tekmovanje. Pri tem je pomembno izpostaviti podatek, da na nekaterih tekmovanjih za ekipo na tekmi lahko nastopi tudi igralec, ki ni prijavljen kot član ekpe na tekmovanju, v okviru katerega poteka tekma. Tekma je sestavljena iz posameznih iger, v okviru posamezne igre točke lahko doseže ena ali nobena ekipa. Končni rezultat ene ekipe na tekmi je vsota njenih točk v vseh igrah. Zmaga ekipa, ki doseže več točk. V primeru, da imata ekipi enako število točk, se odigra dodatna igra, ki določi zmagovalca. V preteklosti je CZS shranjevala samo končne rezultate tekem, zato je vnos rezultatov posameznih iger opcijska možnost.

Igralec je lahko tekom ene sezone član poljubnega števila klubov. Nastopa lahko za poljubo število ekip na poljubnem številu tekem.

Prve entitete bodo tekmovanje, lokacija in kategorija. Tekmovanje ima natanko eno lokacijo in eno kategorijo, medtem ko na eni lokaciji poteka

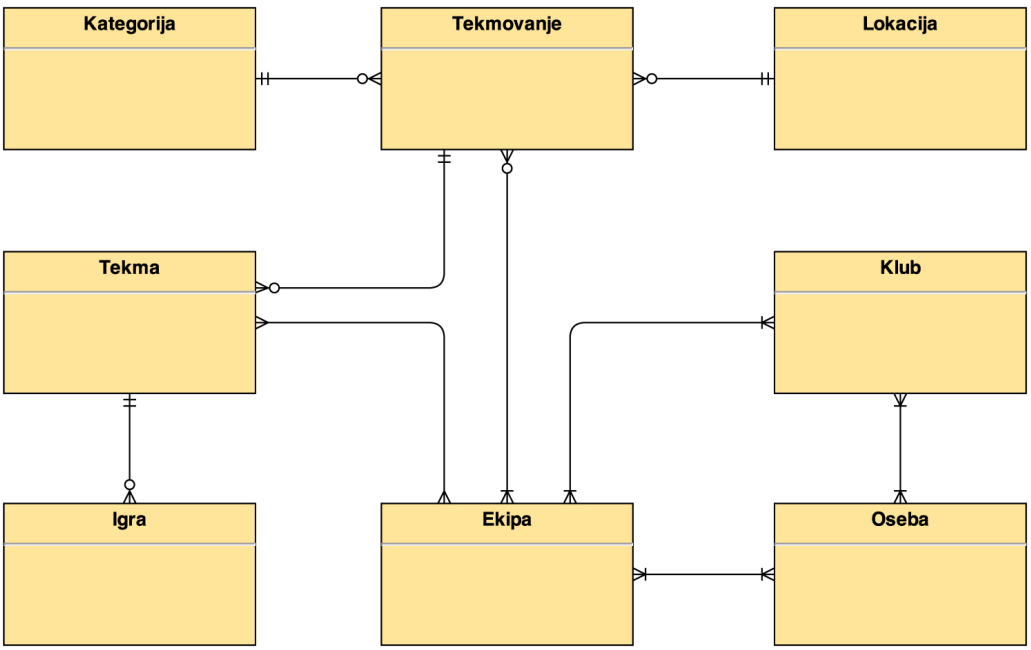
poljubno število tekmovanj in obstaja poljubno število tekmovanj ene kategorije. Povezavi med tekmovanjem in lokacijo ter med tekmovanjem in kategorijo bosta torej tipa “many-to-one”. Naslednja entita, ki pride v poštev, je tekma. Tekmovanje ima več tekem, medtem ko tekma pripada natanko enemu tekmovanje, torej bo razmerje tekmovanje-tekma tipa “one-to-many”. Na enem tekmovanju nastopa več ekip, ekipa pa nastopa na večih tekmovanjih, razmerje med njima je tipa ”many-to-many”. Tipa “many-to-many” bo tudi razmerje med ekipo in klubom. Ker bomo z našo aplikacijo beležili podatke o tekmovalcih, kakor tudi o trenerjih ekip, je smiselno te dve entiteti združiti v eno, ki jo bomo poimenovali oseba. Z eno ekipo je povezanih več oseb, prav tako je oseba lahko povezana na več ekip, torej gre spet za razmerje tipa “many-to-many”. Naslednja entita je igra, ki predstavlja del tekme. Razmerje tekma-igra je tipa ”one-to-many”. Zadnja glavna entiteta je klub. Klub je povezan z osebami in ekipami, obe povezavi pa sta tipa “many-to-many”. Konceptualni podatkovni model je prikazan z diagramom na sliki 3.4.

3.3.2 Logični podatkovni podatkovni model

Logični podatkovni model opisuje podatke kolikor se le da podrobno, ne da bi to vplivalo na samo implementacijo podatkovne baze. Logični podatkovni model vsebuje vse entitete in povezave med njimi. Za vsako entito so navedeni vsi atributi, primarni ključ (angl. primary key) in tuji ključi (angl. foreign key). Na nivoju logičnega modeliranja se opravi tudi normalizacija podatkovne baze in z uvedbo vmesnih tabel odpravijo “many-to-many” relacije.

Če je v tej fazi razvoja podatkovnega modela določanje atributov posameznih entitet bilo bolj kot ne trivialna naloga, pa je toliko večji izziv predstavljala uvedba vmesnih tabel, saj so tekmovalna pravila CZS precej specifična in unikatna.

Vsako tekmovanje ima svoj naziv, datum začetka ter datum konca. Primarni ključ predstavlja *identifikacijska številka* tekmovanja. Tekmovanje ima

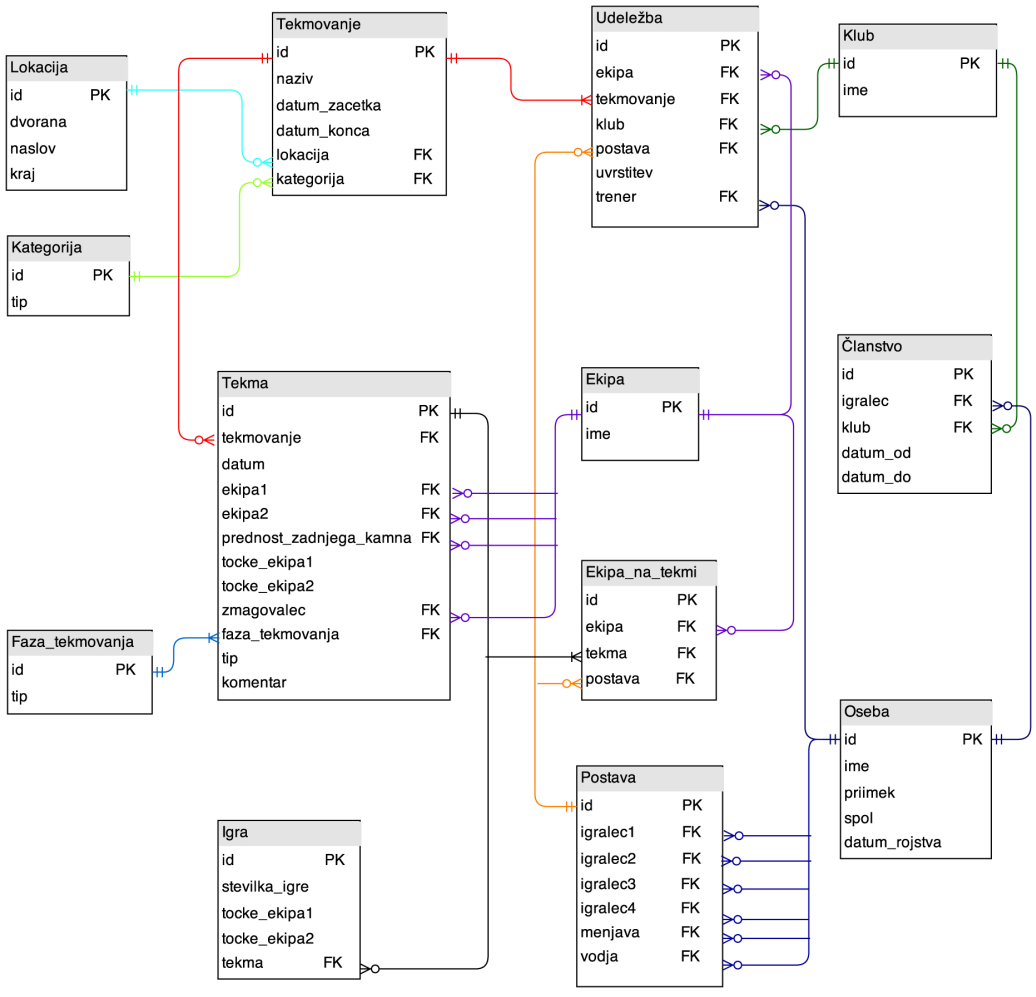


Slika 3.4: Konceptualni model podatkovne baze.

še dva tuja ključa, ki sta *lokacija* in *kategorija*. Primarni ključ entitete lokacija je njena *identifikacijska številka*, lokacija pa ima še atribute *dvorana*, *naslov* ter *kraj*. Kategorijo enolično določa njena *identifikacijska številka*, vsebuje pa še atribut *tip*, ki predstavlja ime kategorije.

Ekipa je gledano s pravilnika CZS precej abstrakten pojem. Ekipa lahko tekom sezone zastopa več različnih klubov, zanjo lahko na različnih tekmovanjih tekom ene sezone nastopa poljubno število različnih igralcev, na nekaterih tekmovanjih pa lahko za eno ekipo igrajo tudi igralci, ki so na tekmovanje prijavljeni kot igralci neke druge ekipe. Če ekipa zamenja ime, se to smatra kot ustanovitev nove ekipe. Zato sem se odločil, da bo entiteta *ekipa* hranila le *identifikacijsko številko* ki je primarni ključ, ter *ime* ekipe. Podatki kot so igralci in klub, ki ga ekipa zastopa, pa se bodo hranili v drugih tabelah.

Prva vmesna tabela bo hranila podatke o udeležbi ekipe na tekmovanju, zato smo jo poimenovali kar *udeležba*. Njen primarni ključ je *identifikacijska številka*. Vsak zapis v tej tabeli bo hranil podatek o uvrstitvi ekipe na tek-



Slika 3.5: Logični model podatkovne baze.

movanju, o klubu, ki ga ekipa zastopa, o postavi, katero je ekipa prijavila na tekmovanje ter o trenerju, če ga je ekipa prijavila. Tuji ključi v tabeli so torej *ekipa*, *tekmovanje*, *klub* in *trener*. Tuji ključ *trener* predstavlja povezavo do entitete tipa *oseba*, ostali tuji ključi pa povezave na entitete enake imenu tujega ključa. Prisoten je še tuji ključ *postava*, ki predstavlja povezavo do vmesne tabele, ki vsebuje vse postave.

Postava je naslednja vmesna tabela, ki predstavlja vrstni red igralcev, v katerem je ekipa igrala na tekmi ali s katerim se je prijavila na tekmovanje. Vsebuje *identifikacijsko številko* kot primarni ključ ter tuje ključe *igralec1*, *igralec2*, *igralec3*, *igralec4*, *menjava* ter *vodja*, ki kažejo na entiteto tipa oseba. Ker na vsaki tekmi nastopata najmanj dva igralca, sta *igralec1* in *igralec2* obvezna, ostali pa opcijski.

Entiteta *oseba* je enolično določena z *identifikacijsko številko*, ima pa atribute *ime*, *priimek* in *spol*. Neobvezen atribut je še *datum rojstva*. Entiteta *klub* ima le *identifikacijsko številko* ter *naziv kluba*, saj s statističnega vidika ostali podatki niso pomembni. Članstvo osebe v klubu se hrani v vmesni tabeli *članstvo*, ki hrani tuja ključa s povezavo na igralca in na klub, ter datuma začetka in prenehanja članstva.

Tudi entiteta *tekma* ima za primarni ključ *identifikacijsko številko*. Tuji ključ *tekmovanje* pove, v sklopu katerega tekmovanja je bila tekma odigrana. Atribut *datum* hrani datum in čas tekme. Imamo štiri tuje ključe s povezavo na entiteto ekipa: *ekipa1*, *ekipa2*, *prednost_zadnjega_kamna* ter *zmagovalec*. Prva dva predstavljata ekipi, ki tekmujeta na tekmi, *prednost_zadnjega_kamna pove*, katera ekipa je na tekmi v prvi igri imela prednost. Čeprav se da zmagovalca določiti s primerjavo atributov *točke_ekipa1* in *točke_ekipa2* ga vseeno hranimo posebaj, kar nam bo kasneje omogočalo nekoliko lažje poizvedovanje. Tekma ima še tuji ključ *faza_tekmovanja* s povezavo na istoimensko entiteto. Atribut *tip* se bo uporabljal za rekonstrukcijo tekmovalnega sistema. To je sicer nekoliko bolj nerodna rešitev samega problema, ki bi se ga dalo rešiti bolj elegantno, o čemer bomo več povedali v nadaljevanju. Neobvezni atribut *komentar* bo shranjeval komentarje na

tekmo, v primeru da pride do nepredvidenih situacij kot je npr. nepojavitev ene izmed ekip na sami tekmi. Entiteta *igra*, ki predstavlja del tekme, ima atribute *številka_igre*, *točke_ekipa1*, *točke_ekipa2* ter tuji ključ tipa *tekma*. Na podlagi takšne strukture je enostavo rekonstruirati potek same tekme.

Za hranjenje podatkov o vrstnem redu igralcev ekipe na posamezni tekmi smo uvedli še eno vmesno tabelo, ki smo jo poimenovali *ekipa_na_tekmi*. Ta vsebuje tuje ključe tipa *ekipa*, *tekma* in *postava* ter *identifikacijsko številko* kot primarni ključ. Entitete s pripadajočimi atributi, ključi ter medsebojnimi povezavami so prikazane na sliki 3.5.

3.3.3 Implementacija baze in fizični podatkovni model

MySQL in phpMyAdmin

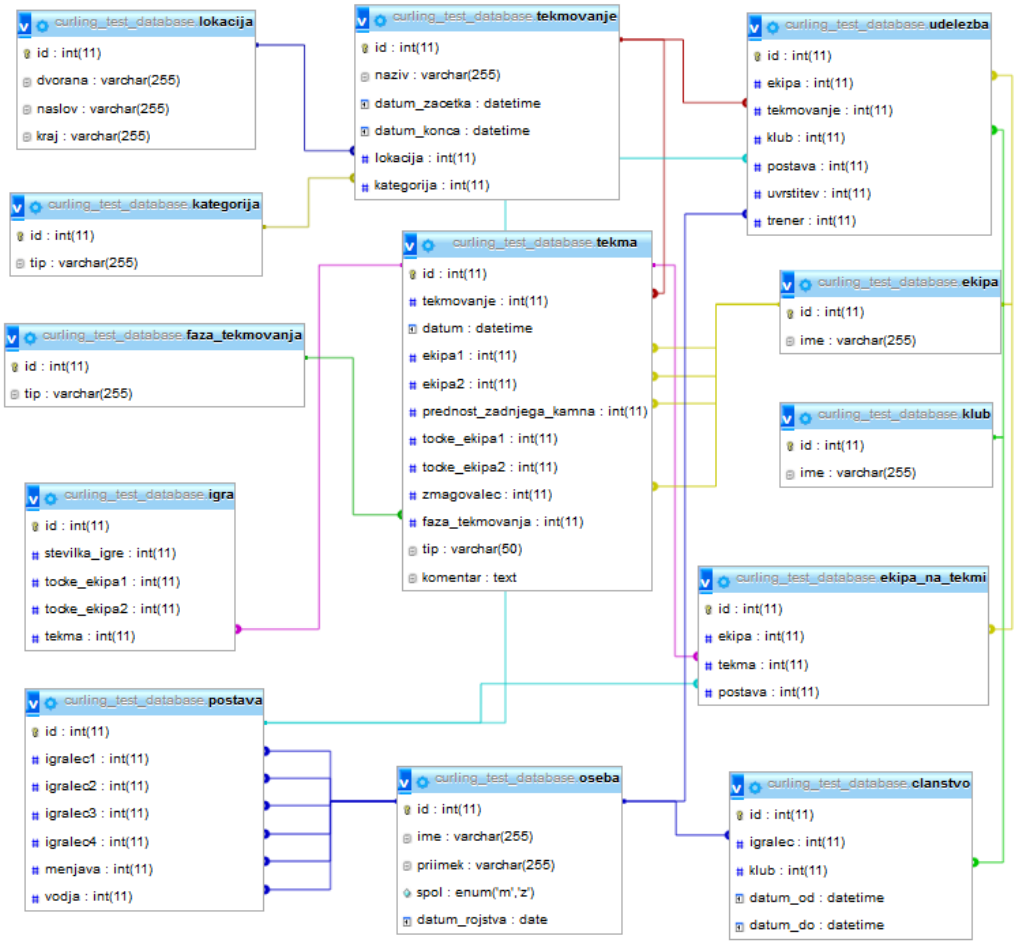
MySQL je odprtokodni sistem za upravljanje s podatkovnimi bazami. Temelji na strukturiranem poizvedovalnem jeziku SQL, ki se uporablja za dodajanje, odstranjevanje in urejanje informacij v podatkovni bazi. V sistemu MySQL se lahko uporabljajo standardni ukazi SQL, kot so ukazi “ADD”, “DROP”, “INSERT” in “UPDATE”. MySQL je lahko uporabljen za različne aplikacije, vendar se z njim najbolj pogosto srečamo na spletnih strežnikih. Podatkovne baze MySQL so relacijske podatkovne baze, kar pomeni, da hranijo podatke v več različnih tabelah, ki so med seboj lahko povezane. Alternative sistemu oziroma sami podatkovni bazi MySQL so npr. Microsoft SQL Server, IBM DB2 ali Oracle Database. Vse izmed naštetih alternativ so plačljive in bolj kot malim aplikacijam namenjene uporabi v večjih podjetjih. Odprtokodna alternativa je SQLite, ki pa je bolj namenjen aplikacijam, kjer le en klient dostopa do podatkovne baze. K odločitvi za uporabo podatkovne baze MySQL sta poleg vsega že naštetega pripomogli tudi njena razširjenost in priljubljenost, ki zagotavljata dobro podporo skupnosti. Ob morebitnih težavah se lahko računa na hitro pomoč s strani skupnosti.

Za implementacijo same baze sem uporabil odprtokodno orodje phpMyAdmin, ki je napisano v skriptnem jeziku PHP in je namenjeno uporavljanju

s sistemom MySQL v brskalniku. Omogoča kreiranje, urejanje in brisanje podatkovnih baz, tabel, polj ali vrstic, izvajanje stavkov jezika SQL ter upravljanje z uporabniki in pravicami. Samo kreiranje tabel je dokaj enostavno, tudi ustvarjanje povezav med njimi ne predstavlja večjega problema. Potrebna je bila nastavitev kodiranja na način UTF-8, ki omogoča shranjevanje šumnikov.

Fizični podatkovni model

Fizični podatkovni model je podatkovni model z vsemi atributi, ki je odvisen od tehnologije, ki jo uporabimo za implementacijo same podatkovne baze. Poleg atributov so v fizičnem podatkovnem modelu označeni tudi njihovi podatkovni tipi. Fizični model naše podatkovne baze implementirane v MySQL prikazuje slika 3.6.



Slika 3.6: Podatkovna baza implementirana z orodjem phpMyAdmin.

Poglavje 4

Razvoj spletne aplikacije

4.1 Uporabljene tehnologije

4.1.1 Aplikacijski strežnik GlassFish

GlassFish je odprtokodni aplikacijski strežnik, ki nudi gostovanje javanskih aplikacij. Kot specifikacija standarda Java EE, ki podpira uporabo tehnologij za implementacijo poslovne logike (JPA, JDBC, servleti, JSF), razvijalcem omogoča ustvaritev prenosnih in skalabilnih poslovnih aplikacij, ki omogočajo integracijo z javanskimi tehnologijami. Večino tega omogočajo tudi drugi aplikacijski strežniki, kot sta Apache Tomcat in JBoss, GlassFish pa je bil izbran zaradi dobre uporabniške izkušnje ob razvoju drugih aplikacij v preteklosti.

4.1.2 JavaServer Faces

JavaServer Faces je standardno javansko ogrodje za igradnjo spletnih aplikacij [5]. Pristop, usmerjen k razvoju uporabniških vmesnikov z modularnimi gradniki, olajša razvoj samih aplikacij. Uporablja se sintaksa jezika XHTML, JSF pa ponuja nove knjižnice značk. JSF omogoča razvoj aplikacij z arhitekturo MVC. Razvoji posameznih nivojev so med seboj neodvisni, kar posledično omogoča tudi lažje vzdrževanje. Vrednosti v komponentah se

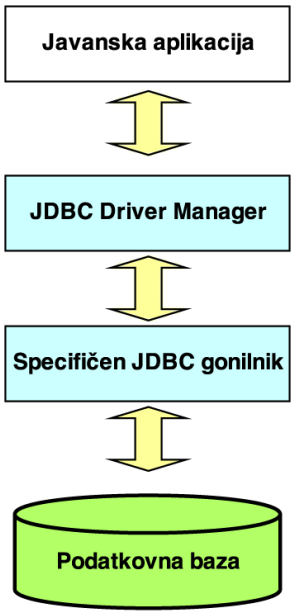
hranijo v javanskih zrnih. JSF omogoča tudi enostavno uporabo tehnologije Ajax. Namesto ogrodja JSF bi se lahko odločili za uporabo npr. spletno aplikacijskih ogrodij ASP.NET, Ruby on Rails ali Django. Vsa našteta ogrodja omogočajo ahrhitekturo MVC. Odločitev za uporabo ogrodja JSF je bila osebne narave, saj sem se želel nekoliko bolj spoznati s tehnologijami, ki jih ponuja Java EE.

4.1.3 Java Database Connectivity

Java Database Connectivity je standardna javanska knjižnica, ki javanskim programom omogoča pošljanje ukazov SQL do podatkovne baze. Knjižnica je usmerjena k delu z relacijskimi podatkovnimi bazami. Vsebuje metode za iskanje in posodabljanje podatkov v podatkovni bazi. Povezave JDBC podpirajo kreiranje in izvajanje ukazov. To so lahko posodobitveni ukazi kot so npr. ukazi SQL “CREATE”, “INSERT”, “UPDATE” in “DELETE” ali pa poizvedovalni ukazi kot je ukaz “SELECT”. Javanska aplikacija, ki želi dostopati do podatkovne baze, komunicira z JDBC Driver Managerjem. Ta komunicira s specifičnim gonilnikom, ki opravi dejansko komunikacijo s podatkovno bazo. Komunikacijska pot od javanske aplikacije do podatkovne baze z uporabo JDBC je prikazana na sliki 4.1. Javanske aplikacije v končni fazi s podatkovno bazo vedno komunicirajo preko vmesnika JDBC, tako da tukaj ni bilo možnosti izbiranja med alternativami.

4.1.4 Java Persistence API

Java Persistence API je javanska specifikacija za dostopanje, shranjevanje in upravljanje podatkov med javanskimi objekti oziroma razredi ter relacijsko podatkovno bazo. JPA se danes smatra kot standardni industrijski pristop za objektno relacijsko preslikavo (angl. Object to Relational Mapping) za javanske aplikacije. JPA je sam po sebi samo specifikacija in ne produkt. Je le skupek vmesnikov, ki potrebujejo implementaicijo. V našem primeru nam JPA dejansko implementira aplikacijski strežnik GlassFish. JPA defi-



Slika 4.1: Komuniciranje aplikacije s podatkovno bazo ob uporabi JDBC.

nira EntityManager API, ki omogoča izvajanje poizvedb in transakcij objektov iz in v podatkovno bazo [1]. JPA definira tudi objektno poizvedovalni jezik JPQL, ki omogoča enostavno poizvedovanje nad objekti iz podatkovne baze. Podobne funkcionalnosti kot Java Persistence API ponuja tudi EJB 2.0 Container-Managed Persistence (CMP), ki pa se kar zadeva prenosnost aplikacije, izkaže za slabšo rešitev. JPA je trenutno najboljša rešitev za uporabo v javanskih aplikacijah.

4.2 Dodajanje rezultatov

Del aplikacije, ki je namenjen dodajanju in urejanju rezultatov, bi lahko označili za CRUD aplikacijo. Čarovniki v razvojnem okolju NetBeans, ki smo ga uporabili za razvoj aplikacije, omogočajo enostavno izgradnjo osnove za našo CRUD aplikcijo na osnovi povezane podatkovne baze. Najprej je bilo potrebno vse tabele iz podatkovne baze preslikati v ustrezne entitetne

razrede. Čarovnik za vsako tabelo iz podatkovne baze ustvari svoj entitetni razred. Naslednji korak je bilo kreiranje spletnega vmesnika za pregledovanje in vzdrževanje podatkov, za kar ponovno poskrbi čarovnik. Čarovnik iz obstoječih entitetnih razredov ustvari t.i. JSF Pages. Koda, ki jo čarovnik generira bazira na anotacijah v entitetnih razredih. Za vsak entitetni razred čarovnik ustvari sejno zrno brez stanja, sejno upraviteljsko zrno in direktorij, ki vsebuje štiri "Facelets" datoteke, ki omogočajo CRUD storitve(Create.xhtml, Edit.xhtml, List.xhtml in View.xhtml). Aplikacija, kreirana s čarovnikom nam že zagotovalja izvajanje CRUD operacij, vendar ne omogoča dobre uporabniške izkušnje. Trenutno je mogoče dodajanje vsake entitete posebej. To pomeni, da bi v primeru, ko želimo dodati tekmo, v sklopu katere je bilo odigranih 10 iger, morali vsako igro dodati posebej, kar bi predstavljajo veliko več dela, kot si ga želimo. Zato je bila potrebna združitev nekaterih pogledov in logike, ki se izvaja v ozadju. Tako se zapis v tabelo postava opravi v sklopu dodajana udeležbe ali tekme, zapisi v tabeli igra in ekipa_na_tekmi pa v sklopu dodajanja tekme.

4.2.1 Seznam obstoječih entitet

Na prvi strani, do katere lahko dostopa administrator, se nahajajo povezave do seznamov že obstoječih podatkov. Izbiramo lahko med tekmovanji, osebami, ekipami, tekmami, udeležbami, klubi, članstvi, fazami ter kategorijami tekmovanj in lokacijami. Ob kliku na želeno povezavo se nam odpre stran, ki prikazuje vse trenutno obstoječe entitete izbranega entitetnega tipa. Za prikaz je uporabljena komponenta JSF dataTable, ki se generira dinamično glede na podatke v podatkovni bazi. Za boljši pregled je uporabljena paginacija, podatki pa so smiselno urejeni (npr. tekmovanja po datumu od najnovejšega do najstarejšega). V vsaki vrstici tabele je prikazana ena entiteta (npr. tekma) oziroma tej entiteti pripadajoči glavni atributi (tekmovanje, datum, ekipi, rezultat). Poleg atributov entitete vsaka vrstica vsebuje še povezave za ogled podrobnosti entitete, urejanje entitete ter izbris entitete iz podatkovne baze. Pod tabelo se nahajata še povezava za ustvarjanje nove en-

Slika 4.2: Seznam obstoječih entitet tipa tekma.

titete izbranega entitetnega tipa in poveza do začetne strani. Primer izgleda seznama je prikazan na sliki 4.2.

Seznami entitet za vse entitetne tipe so realizirani enako, razlika je le pri prikazanih atributih, ki so za vsak entiteti tip seveda drugačni.

4.2.2 Dodajanje, pregledovanje in urejanje obstoječih entitet

Ob kliku na povezavo "Dodaj novo entiteto", ki se nahaja pod seznamom obstoječih entitet se odpre stran, kjer lahko dodamo novo entiteto. Beseda "entiteto"v labeli povezave "Dodaj novo entiteto"je seveda zamenjana z imenom ustreznega entitenega tipa, kar je razvidno iz slike 4.2. Ob kreiranju nove entitete se primarni ključ *identifikacijska številka* generira sama, za ostalo pa je treba poskrbeti ročno. Tuje ključe se določi z izborom ene izmed vrednosti iz spustnega menija. Spustni meniji so realizirani s komponento JSF *selectOneMenu*. Vrednosti, ki jih lahko izberemo v spustnem meniju, se napolnijo z uporabo komponente *selectItems*. Komponenta *selectItems* pokliče

ustrezno metodo iz ustreznega zrna ter tako pridobi vredosti, ki so kandidati za tuji ključ. Izjema je tuji ključ *prednost_zadnjega_kamna*, ki se nahaja v tabeli *tekma*. Tega se določi s klikom na potrditveno polje ob ekipi, ki ima prednost zadnjega kamna. Atribute, ki so v entitetnem razredu deklarirani kot cela števila, znak, niz znakov ali datum, se vpisuje preko vnosnih polj, ki so realizirana z komponentno JSF *inputText*. Čeprav na večini spletnih strani, kjer je potreben vnos datuma, vidimo drugačen pristop, sem se sam odločil za vnosno polje, ker mi izpolnitev tega polja vzame manj časa, kot bi mi ga npr. izbor želenega datuma s klikanjem na koledar. Izgled pogleda za dodajanje entitete tekma je prikazan na sliki 4.3.

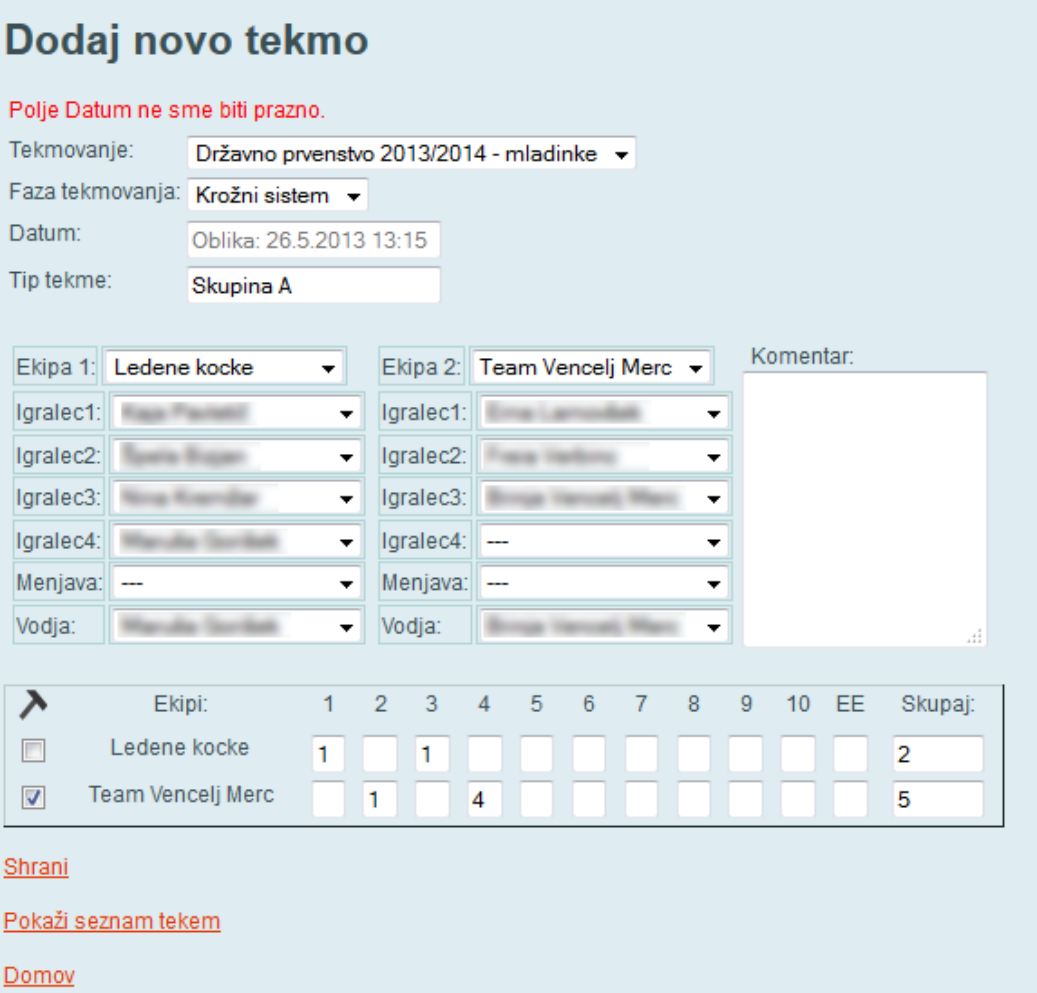
V primeru, da uporabnik ne izpolni katerega izmed obveznih polj, ga aplikacija o tem obvesti. Uporabnik je obveščen tudi, če v kakšno polje vnese napačno vrednost, kot je npr. format datuma. Če so vsa obvezna polja pravilno izpolnjena, se ob kliku na povezavo "Shrani" pošlje objekt do vmesnika JPA, ki z objektno relacijskim preslikovanjem ustvari zapis primeren za podatkovno bazo in nato s pomočjo objekta EntityManager tudi ustvari nov zapis v sami podatkovni bazi. Postopek dodajanje tekme torej izgleda takole:

- Iz spustnega menija izberemo želeno tekmovanje. Ob tem se z uporabo tehnologije Ajax napolnita tudi spustna meniju za izbor ekip. Na voljo so le ekipe, ki so prijavljene na izbrano tekmovanje.

- Iz spustnega menija izberemo fazo tekmovanja ter v vnosno polje vpišemo datum ter uro tekme.

- Če je potrebno vpišemo tip tekme in komentar.

- Izberemo prvo ekipo. S pomočjo tehnologije Ajax nam aplikacija predlaga postavo ekipe. Predlagana postava ekipe je enaka tisti, ki jo je ekipa navedla ob prijavi na tekmovanje. Prav tako se s pomočjo tehnologije Ajax v tabelo rezultatov izpiše ime izbrane ekipe, da ob vnašanju rezultatov slučajno ne pride do zamenjave.



Slika 4.3: Dodajanje nove tekme.

- Če je potrebno spremenimo vrstni red igralcev prve ekipe.
- Postopek ponovimo še za drugo ekipo.
- Označimo, katera ekipa je na tekmi imela prednost zadnjega kamna.
- Vpišemo končni rezultat tekme ter opcijsko še rezultate posameznih iger in kliknemo povezavo "Shrani".
- V primeru, da smo pozabili izpolniti katero izmed obveznih polj ali pa v kakšno polje vnesli neveljavno vrednost, smo na to opozorjeni ter popravimo vnos.
- V podatkovni bazi se v tabeli *tekma* ustvari nova vrstica z izpolnjenimi podatki.
- V primeru, da katera izmed vnešenih postav v podatkovni bazi še ne obstaja, se v tabelo *postava* shrani nova postava.
- V tabeli *ekipa_na_tekmi* se ustvarita nova zapisa, za vsako ekipo svoj.
- V primeru, da smo vnesli rezultate posameznih iger, se za vsako igro v podatkovno bazo v tabelo *igra* vnese ena vrstica.

Postopki dodajanja drugih entitet sledijo enakemu principu, le da pri večini ostalih dodajanje ustvari zapis le v eni tabeli podatkovne baze. Za urejanje entitet se uporablja enak pogled kot za dodajanje, s to razliko, da se nam ob kliku na povezavo "Uredi" ob izbrani entiteti iz seznama ta entiteta oziroma njeni atributi naložijo v sam pogled in jih po želji lahko spreminjamo. Ko želimo entiteto shraniti, se v ustrezni tabeli podatkovne baze ne ustvari nova vrstica, temveč se posodobi obstoječa. Tudi za ogled entitet je pogled enak, le da v tem pogledu atributov ne moramo spreminjati, temveč si lahko samo ogledamo njihove vrednosti.

4.3 Pregledovanje rezultatov

Del aplikacije, ki je namenjen širši množici uporabnikov, ki želijo pregledovati rezultate in statistiko curling v Sloveniji, je razdeljen na štiri spletne podstrani:

- tekmovanja,
- osebe,
- ekipe in
- napredno iskanje

4.3.1 Tekmovanja

Stran tekmovanja hkrati predstavlja tudi domačo stran aplikacije. Na njej je v začetku prikazanih zadnjih deset pod okriljem CZS organiziranih tekmovanj. Tekmovanja so prikazana z uporabo komponente JSF dataTable, v vsaki vrstici tabele pa so izpisani naziv, lokacija ter datum začetka in konca tekmovanja. Omogočeno je tudi iskanje tekmovanj na podlagi sezone in kategorije tekmovanja. Tabela se ob spreminjanju iskalnega kriterija osvežuje s tehnologijo Ajax.

Ob kliku na naziv tekmovanja se tabela vseh tekmovanj skrije, pokaže pa se gradnik, ki vsebuje podrobnosti tekmovanja. V sklopu podrobnosti lahko izbiramo med prikazom vrstnega reda ekip in prikazom rezultatov tekem. Vrstni red ekip prikazuje ekipe, razvrščene od prvega proti zadnjemu mestu. Poleg uvrstitve in imena ekip je prikazan klub, ki ga je ekipa zastopala ter postava, ki jo je ekipa prijavila na tekmovanje. V sklopu rezultatov tekem so prikazani rezultati vseh tekem razvrščeni po datumu in uri padajoče. Za vsako tekmo je navedena faza tekmovanja, ekipi ki sta igrali, rezultat po posameznih igrah, končni rezultat. Ekipa, ki je imela na tekmi prednost zadnjega kamna v prvi igri ima ob imenu izrisano majhno kladivo, ki je standardni simbol za označevanje prednosti zadnjega kamna. S klikom

na povezavo “Nazaj na seznam tekmovanj” se gradnik s podrobnostimi tekmovanja skrije, ponovno pa se izriše seznam tekmovanj. Ponovno izrisani seznam hrani stanje našega prej izbranega iskalnega kriterija. Vsi prehodi med zavihki se opravijo s pomočjo tehnologije Ajax.

4.3.2 Osebe

Na strani *osebe* uporabnik lahko pregleduje povzetke uvrstitev tekmovalcev na tekmovanjih. Uporabnik v vnosno polje vnese ime ali priimek iskane osebe, aplikacija mu v odgovor ponudi seznam ustreznih oseb. Iskanje je realizirano z uporabo podnizov in je neodvisno od velikosti črk. Torej nam bodo poizvedbe “Miha Novak”, “miha”, “Novak” ali ”nova” vse vrnile osebo Miha Novak, če ta oseba seveda obstaja. Ob kliku na osebo iz seznama vrnjenih oseb se nam odpre dejanski povzetek uvrstitev te osebe. Povzetek vsebuje preglednico, kjer so za vsako kategorijo tekmovanja, ki se jo je tekmovalec udeležil, navedeni število odigranih tekmovanj te kategorije, število prvih, drugih ter tretjih mest, najboljša uvrstitev, število tekem, število zmag in število porazov. Pod to preglednico se nahaja še seznam vseh tekmovanj, na katerih je tekmovalec igral. Poleg naziva tekmovanja sta napisana še ekipa, ki jo je tekmovalec zastopal in končna uvrstitev ekipe na tekmovanju.

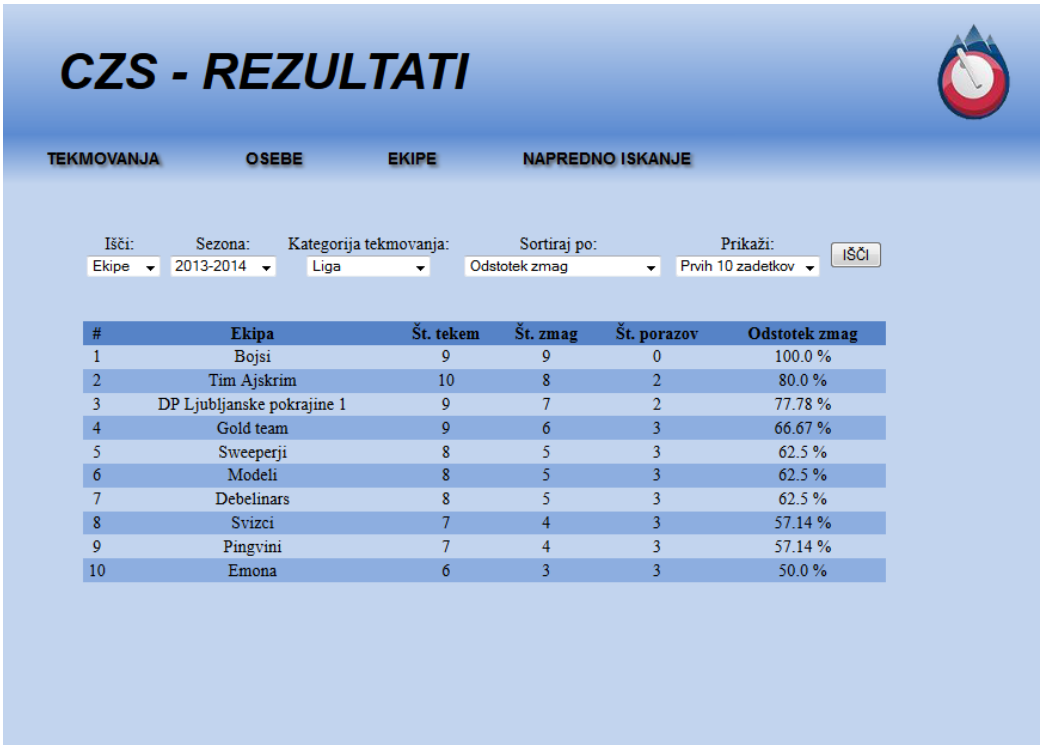
4.3.3 Ekipe

Stran *ekipe* ponuja iste funkcionalnosti kot stran *osebe*, le da tukaj kot iskalni niz vpišemo ime ekipe, povzetek rezultatov ter seznam odigranih tekmovanj pa se nanašata na izbrano ekipo.

4.3.4 Napredno iskanje

Stran napredno iskanje je kot že samo ime pove namenjena nekoliko bolj specifičnemu poizvedovanju. Na tej strani lahko spreminjamo pet iskalnih kriterijev, ki se nahajajo v spustnih menijih:

1. Izbira entitetnega tipa, ki ga želimo prikazati (oseba, ekipa ali klub).



#	Ekipa	Št. tekem	Št. zmag	Št. porazov	Odstotek zmag
1	Bojsi	9	9	0	100.0 %
2	Tim Ajskrim	10	8	2	80.0 %
3	DP Ljubljanske pokrajine 1	9	7	2	77.78 %
4	Gold team	9	6	3	66.67 %
5	Sweeperji	8	5	3	62.5 %
6	Modeli	8	5	3	62.5 %
7	Debelinars	8	5	3	62.5 %
8	Svizci	7	4	3	57.14 %
9	Pingvini	7	4	3	57.14 %
10	Emona	6	3	3	50.0 %

Slika 4.4: Primer prikaza rezultatov naprednega iskanja.

2. Izbira tekmovalne sezone.
3. Izbira kategorije tekmovanja.
4. Izbira kriterija, po katerem želimo razvrstiti rezultate (odstotek zmag, največ osvojenih medalj ali največ osvojenih točk).
5. Izbira željenega števila prikazaih zadetkov.

Pri izboru željene tekmovalne sezone in kategorije tekmovanja lahko poleg obstoječih sezon in kategorij izberemo tudi opciji vse sezone oziroma vse kategorije. Tudi pri izbiri željenega števila prikazanih zadetkov obstaja možnost izbire vsi zadetki. V primeru, da želimo prikazati večje število zadetkov, kot jih dejansko obstaja, aplikacija prikaže vse obstoječe zadetke. Primer prikaza rezultatov naprednega iskanja je prikazan na sliki 4.4.

Poglavje 5

Zaključek

V sklopu diplomskega dela je bila razvita celovita spletna aplikacija za shranjevanje in brskanje po rezultatih curling tekmovanj v organizaciji Curling zveze Slovenije. Aplikacija bo dostopna na spletni strani CZS. Pričakujem, da bo zadovoljevala potrebe slovenskih navdušencev nad curlingom po pregledovanju rezultatov tekmovanj, ki so se jih udeležili sami ali pa so na njih tekmovali njihovi znanci. Aplikacija bi lahko prinesla še nekaj dodane vrednosti hitro razvijajoči se Curling zvezi Slovenije, ki bo tako lahko še v nečem za zgled drugim članicam Svetovne curling zveze. Na nek način bi lahko doprinesla tudi k popularnosti samega curlinga pri nas. Večini ljudi pojavitev njihovega imena na svetovnem spletu prinaša zadovoljstvo in ravno s tega psihološkega vidika bo slovenska curling skupnost morda pridobila kakšnega novega člana.

Razvita aplikacija izpolnjuje vse cilje, ki sem si jih zadal pred začetkom izdelave, vsekakor pa ostaja še nekaj maneverskega prostora za izboljšave. Ena izmed možnih izboljšav bi bila uvedba predlog sistemov tekmovanj, ki bi omogočala enostavnejšo rekonstrukcijo poteka samega tekmovanja, ki je trenutno realizirana nekoliko nerodno. Za vsak sistem tekmovanja, CZS trenutno uporablja 3 različne, bi se ustvarila svoja predloga. Razvita aplikacija trenutno omogoča določanje faze tekmovanja, v kateri je bila tekma odigrana, vendar pa znotraj skupinskega dela ne razlikuje med skupinami na način, ki

ga je enostavno rekonstruirati.

Druga optimizacija, ki bi jo bilo mogoče opraviti, je implementacija pogledov SQL (angl. SQL Views). To so virtualne tabele, ki bazirajo na rezultatu shranjene poizvedbe SQL in ki se avtomatsko posodabljajo. To pomeni, da bi na mestih, kjer se rezultati neke poizvedbe prikazujejo pogosto (kot npr. seznam tekmovanj pri iskanju rezultatov) te rezultate shranili v pogled SQL in se tako izognili nepotrebnemu večkratnemu izvajanju iste poizvedbe.

V primeru, da v bo CZS v prihodnosti uspela zagotoviti tudi beleženje uspešnosti posameznih tekmovalcev na tekmah, pa se nam odpira popolnoma nova zgodba, ki bi ob manjši nadgradnji aplikacije lahko uporabnikom ponudila še več zanimivih funkcionalnosti.

Literatura

[1] Developing with ejb and jpa components. http://docs.oracle.com/middleware/1212/jdev/OJDUG/dev_ejb_jpa.htm. Dostop: 1.9.2014.

[2] Donald Bell. Uml basics: The sequence diagram. http://www.ibm.com/developerworks/rational/library/3101.html. Dostop: 5.9.2014.

[3] Clare Churcher and Stéphane Faroult. *Beginning database design*, volume 2. Springer, 2007.

[4] Dejan Lavbič. Načrtovanje pb. https://ucilnica.fri.uni-lj.si/mod/folder/view.php?id=23695. Dostop: 15.8.2014.

[5] Chris Schalk. Introduction to javaserver faces - what is jsf? http://www.oracle.com/technetwork/topics/index-090910.html. Dostop: 1.9.2014.