MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Aplikace pro správu motocyklových soutěží

BAKALÁŘSKÁ PRÁCE

Pavel Grochal

Brno, jaro 2011

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: RNDr. Jaroslav Ráček, Ph.D.

Poděkování

Chtěl bych poděkovat mému vedoucímu bakalářské práce RNDr. Jaroslavu Ráčkovi, Ph.D. nejen za podnětné rady, které mi při psaní této práce poskytl, ale také za jeho nesmírné nadšení pro vedení této práce. Dále bych chtěl poděkovat svým rodičům a své přítelkyni za morální podporu a prostředí, které mi pro tvorbu práce vytvořili.

Shrnutí

Smyslem této bakalářské práce byla analýza a návrh datových struktur systému, sloužícího ke zpracování a správě dat, získaných při organizování různých typů motoristických akcí. Vytvořený a zdokumentovaný prototyp slouží jako vodítko k vypracování implementace zamýšleného systému.

Abstract

Purpose of this thesis is to analyze data structures used in managing and processing data obtained during motor vehicle races. It's also focused on designing data structures and applications meant to assist organisers of such events. The design and documentation provided serve as guidelines for actual implementation of such system.

Klíčová slova

motocyklové sporty, motoristické soutěže, motokros, závody, PHP, MySQL, HTML, CSS, CodeIgniter, Webová aplikace, entitně-relační diagram

Obsah

1 **Úvod** . . . 3
1.1 *Systémy bodování* . . . 4
1.1.1 Kvalifikace . . . 5
1.1.2 Závod . . . 6
1.1.3 Bodování . . . 7
2 **Analýza a návrh** . . . 8
2.1 *Případy užití* . . . 8
2.1.1 Správce systému . . . 9
2.1.2 Pořadatel . . . 9
2.1.3 Závodník . . . 9
2.2 *Datový model* . . . 9
2.2.1 Administrativní část . . . 10
2.2.2 Organizační část . . . 12
2.2.3 Vyhodnocovací část . . . 13
2.2.4 Uživatelská část . . . 16
3 **Implementace** . . . 18
3.1 *Implementace administrativní části* . . . 19
3.2 *Implementace organizační části* . . . 20
3.3 *Implementace vyhodnocovací části* . . . 22
3.4 *Použité algoritmy* . . . 25
3.4.1 Zobrazení závodních víkendů v sezóně seriálu . . . 25
3.4.2 Vyhodnocení výsledků závodu . . . 27
3.4.3 Vyhodnocení výsledků třídy za závodní víkend . . . 29
3.4.4 Import a kontrola výsledků závodu . . . 32
4 **Funkcionalita** . . . 36
4.1 *Registrace a přihlášení* . . . 36
4.2 *Přidávání, úprava a mazání* . . . 37
4.3 *Propojování* . . . 37
4.4 *Zobrazování* . . . 38
5 **Závěr** . . . 39

Přílohy:

A **Entitně-relační Diagram** 42
B **CD** 44

Kapitola 1

Úvod

Motivací k vytvoření této bakalářské práce byl můj zájem o motocyklové sporty, kterých je celá řada, přičemž metody a postupy hodnocení výsledků soutěžících bývají diametrálně odlišné. Ne vždy jsou soutěžící hodnoceni podle pořadí v cíli v rámci konkrétní rozjížďky. Jako příklady zde dobře poslouží třeba bodové ohodnocení jezdce v časovém závodě na jedno kolo stanovené závodní tratě. Bodována však může být také vzdálenost, kterou dokázal jezdec zdolat a to bez ohledu na časový limit. Typickým příkladem jsou klasické závody motocyklů do vrchu. Odlišná kritéria hodnocení soutěžích mají svůj původ v široké škále motocyklových soutěží i druhu sportovních odvětví, v nichž závodníci soupeří. Motocyklové sporty lze takto rozčlenit na motocyklové závody, pořádané na silnicích, na závodních silničních okruzích, dále pak na závody, pořádané na přírodních okruzích (motokros, superkros, plochá dráha, ledová plochá dráha), závody, pořádané v přírodě mimo závodní okruhy (enduro soutěže, vytrvalostní soutěže, soutěže extrémních dovedností, závody do vrchu, skoky, motocyklová akrobacie, exhibice a freestyle) a nebo na závody halové, které simulují přirozené prostředí motocyklových závodů, jak je uvedeno shora. Dnes však není znám žádný program, který by byl schopen spravovat, zpracovávat a vyhodnocovat výsledky pro tak široké spektrum sportů, které jsou výše uvedeny, ani program, který by bylo možno použít pro více, nežli jeden jediný druh motocyklového sportu. Sám jsem měl možnost být vedoucím členem týmu časomíry na motokrosové trati a pro její činnost jsem vytvořil jednoduchý program, který evidoval závodníky, přiřazoval jim jejich bodové ohodnocení a poskytoval setříděné výsledkové listiny. Tento program byl však zcela nedostačující pro jiné závody, které se začaly na této motokrosové trati pořádat. Tak vznikla myšlenka na vytvoření programu, který by byl univerzálně použitelný. Začal jsem se tedy hlouběji zaobírat touto problematikou a vypracoval jsem analýzu a návrh systému, který je schopen spravovat, zpracovávat a vyhodnocovat výsledky různých závodů.

Cílem této bakalářské práce bylo analyzovat a vytvořit univerzální návrh, věnující se problematice správy, zpracování a vyhodnocování výsledků, který by byl všeobecně použitelný. Jelikož se výsledky a hodnocení jezdců u jednotlivých závodů liší, prozkoumání aspektů více závodů bylo nezbytně nutné k vytvoření obecnějšího a univerzálnějšího ná-

vrhu tohoto systému.

V první části bakalářské práce je vysvětleno, jakým způsobem se člení, organizují, ohodnocují a prakticky realizují různé typy motocyklových soutěží. Do této části byly zahrnuty také specifika, organizace, bonifikace a provádění kvalifikací, coby velmi podstatné součásti mnoha závodů. Ve všech případech jde o motocyklové nebo motoristické sporty, na nichž demonstruji fungování vnitřních mechanizmů a pravidel.

Druhá část práce je zaměřena na samotnou analýzu a návrh systému. Nejprve jsou podrobněji popsány uživatelské role, které by tuto aplikaci měly používat, a také rozsahy a způsob jejich interakcí se systémem. Dále je rozkreslen ERD[11] model systému. Tento je rozdělen na čtyři dílčí sekce, a to administrativní, organizační, vyhodnocovací a část uživatelskou. Administrativní část ERD slouží jako podklad pro vytváření, organizaci a správu soutěží a jejich kategorizaci v rámci ročníků. Organizační část se zaobírá způsobem uspořádání závodů, jejich návazností a charakteristikou způsobu hodnocení a bodování jezdců závodu, přihlašováním jezdců do předem připravených závodních tříd. Část vyhodnocovací má za úkol rozdělit jednotlivé jezdce v kategoriích, stanovit a pamatovat si jejich pořadí a dále také přiřadit patřičné bodové ohodnocení, které za závod získali. Uživatelská část se zabývá správou uživatelů, používajících tento systém. Jedná se tedy o závodníky a týmy, organizátory, kteří jednotlivé soutěže a závodní víkendy pořádají a v neposlední řadě také správců systému udržujících systém a koordinujících jednotlivé skupiny uživatelů.

Ve třetí části je popsána implementace prototypu systému. Jsou zde uvedeny obecné charakteristiky, které jsou jazykově a implementačně nezávislé nebo jen nepatrně závislé. Příkladem jsou rozhraní jednotlivých tříd, která obstarávají chod modelové vrstvy. Třetí část obsahuje také užité zdrojové kódy, jež jsou doplněny komentářem a vysvětlivkami o funkcích a důvodech jejich použití.

Čtvrtá část se primárně zabývá funkcionalitou systému jako celku. Jsou zde vyobrazeny návrhy jednotlivých náhledů grafického rozhraní v souvislosti s praktickým využíváním programu. Opomenuty nebyly ani teoretické možnosti vylepšení, které by mohly do jisté míry více zobecnit systém, to však za cenu složitějšího vývoje a nákladnější údržby systému.

1.1 Systémy bodování

Závody jsou zřídka organizovány izolovaně, ad hoc, bez toho, aniž by tvořily součást nějakého vyššího celku. Obvykle se tyto celky nazývají seriály nebo CUPy[1] a pořádají se v rámci takzvaných ročníků. Každý ze seriálů má zpravidla své unikátní regule a není

1. Z anglického slova „cup“ neboli pohár.

jednoduché vytvořit obecný systém, zohledňující jejich individuální požadavky. Je potřeba najít takové charakteristické rysy jednotlivých závodů, které mohou být brány jako vzorové.

Smyslem jednotlivých závodů v seriálu je ohodnotit jezdce podle jejich sportovních výkonů. Obecně by se dalo říci, že čím více závodů tvoří daný seriál, tím přesnější výsledky o schopnostech a dovednostech soutěžících hodnotící systém poskytuje. K ohodnocení jezdců slouží bodovací systémy, zohledňující jejich výkon a umístění v závodech. Sportovní soutěže mají rozdílný průběh závodů. Vyšší, zpravidla nadnárodní závody se dělí na kvalifikaci a samotný závod. Kvalifikace obvykle slouží jako selektivní prvek v závodech, v nichž je přihlášeno více soutěžících, než kolik jich může startovat v hlavním závodě. Kvalifikace však také umožňuje jezdci například start v závodě z lepší pozice a tím nepřímo ovlivní i závodníkovo celkové umístění v závodu. Hlavní závod je důležitý pro získání bodů do hodnocení.

Je potřeba zdůraznit, že zmíněné členění slouží v této práci pouze jako model, na kterém lze demonstrovat shodné a odlišné vlastnosti soutěží. Existuje i řada jiných způsobů jak soutěže kategorizovat. V závěru práce odkazuji na jednotlivé soutěže a pravidla pro jejich pořádání.

1.1.1 Kvalifikace

Kvalifikace slouží k rozestavení jezdců v startovním poli závodu. Jezdec je povinen splnit určité požadavky pro to, aby se kvalifikoval do hlavního závodu. Nejběžnějším způsobem kvalifikace je závod, kdy se jezdci snaží získat nejlepší čas na vymezeném úseku či kole. Jezdci jsou seřazeni na startovním roštu podle získaných časů[2]. Dalším způsobem je kvalifikace podobná závodu, kdy první závodník, který ujede stanovený počet kol jako první, vyhrává kvalifikaci.

V některých soutěžích si jezdci mohou podle pořadí v kvalifikačních rozjížď'kách také vybrat startovní pozici, tedy místo, o němž předpokládají, že jim napomohou v hlavním závodě k co nejlepšímu umístění bezprostředně po odstartování závodu. Jelikož prakticky každá motokrosová trať je jedinečná, nelze předem určovat výhodnost startovního místa, a proto si svou startovní pozici jezdec vybírá až na místě samotném.

Existují i další způsoby kvalifikací pro motoristické či motocyklové sporty. Tyto způsoby se buď do velké míry shodují s již popisovanými způsoby a s drobnou obměnou využívají nebo se naopak rozcházejí do té míry, že jsou používány pouze v jedné či velmi málo soutěžích, které nejsou tak známé, jako výše uváděné příklady.

U některých soutěží se kvalifikace nepořádá. Jednoduchým způsobem jak seřadit zá-

2. Malá změna platí u závodu WTCC[1], kdy prvních 10 závodníků je na startu umístěno v opačném pořadí.

vodníky na startovní pole, je podle celkového umístění v sérii předešlých závodů nebo podle výsledků v minulém závodu. Toto řazení se používá zpravidla u všech pohárových soutěžích na všech úrovních soutěží. Jedná se tak o jistý druh bonifikace jezdců již v samotném průběhu celého poháru, přičemž právě tím kritériem, které vyhodnocuje a bonifikuje jednotlivé soutěžící, je jejich aktuální umístění v celém sportovním podniku.

1.1.2 Závod

Závody soutěží se dělí podle způsobu hodnocení, jež vymezuje jejich délku. A to z pohledu časového a vzdálenostního anebo jejich kombinací. V motocyklových soutěžích vznikly čtyři skupiny, podle kterých se dají kategorizovat závody.

První skupinou jsou závody, ve kterých je potřeba ujet předem stanovenou vzdálenost nebo adekvátní počet kol, pokud jde o závod na okruhu. Jezdci startují vždy najednou, rozmístěni na startovním roštu obvykle dle předchozí kvalifikace. První jezdec, který dorazí do cíle, vyhrává. Tento princip je nejznámější a používá se v mnoha soutěžích. Jako příklad slouží Moto GP[3], kde délka, kterou musí jezdci ujet, se pohybuje od 95 do 130 kilometrů. Vzhledem k tomu, že se jedná o kolový závod, je od tohoto údaje odvozen počet kol, který je unikátní pro každou trať. Dále je možné najít tento způsob hodnocení u závodů WSBK[4].

Další skupinou jsou závody, které mají pevně stanovený čas. Jezdci obvykle nestartují všichni najednou, ale v určitých intervalech. Oproti předchozí skupině není konstantní vzdálenost, kterou musí jezdci ujet. V těchto závodech se jezdci hodnotí podle toho, kolik kol či jak dlouhý vzdálenostní úsek ujeli za stanovený čas. Obvykle se jedná o závody vytrvalostního charakteru. Oproti první skupině se tyto závody jedou vždy po stejně dlouhý čas.

Třetí skupina závodů má hodnocení stanovené podle nejlepšího času za vymezený úsek trati. Jezdci obvykle nestartují souběžně, ale v časových rozestupech. Snaží se zajet co nejlepší čas pro stanovené úseky. Navazující úseky lze rozdělit do závodů, a v případě nutnosti některý z nich opakovat. Tento způsob hodnocení se používá běžně ve vytrvalostních soutěžích, enduro soutěžích, vícedenních soutěžích, soutěžích extrémních, ale také například v soutěži Rallye Paříž-Dakar[5].

V poslední skupině závodů jde o kombinaci závodu ve stanoveném čase a počtu kol. Nejprve závodníci jedou po předem stanovený čas a poté, co vedoucí závodník projede po uplynutí času cílem, začíná druhá část, v níž jezdci musí ještě ujet předem stanovený počet kol. Jezdec s nejvyšším počtem kol a nejrychlejším časem vítězí.

Tento specifický způsob hodnocení je běžně používán u motokrosových soutěží. Doba závodu a počet kol je specificky určen pro každou závodní třídu. U nejnižších tříd, tak-

zvaných tříd dětských, kde se třídy dělí podle věku a kubatury motocyklů na MX50, MX65 a MX85 se závod jede na 10-15 minut plus 1 až 2 kola. U tříd ostatních, které se dělí podle věku závodníků a podle kubatur motocyklů, se závodí zpravidla 15-30 minut plus 1-3 kola.

1.1.3 Bodování

Kvalifikace a závody rozřazují jezdce podle jejich výkonu, který musí být patřičným způsobem ohodnocen. Ve většině profesionálních soutěžích je znám počet jezdců, kteří startují, a proto je možné je bodovat pevně stanoveným počtem bodů za odpovídající místo. Například v soutěži MotoGP[3] se za první místo uděluje 25 bodů, za druhé je bodů 20, dále pak 16 za místo třetí, 13 za místo čtvrté, 11 za místo páté a od šestého do patnáctého místa jsou jezdci hodnoceni 10 až 1 bodem. Tento systém bodování se zachovává na všech tratích po celou sezónu.

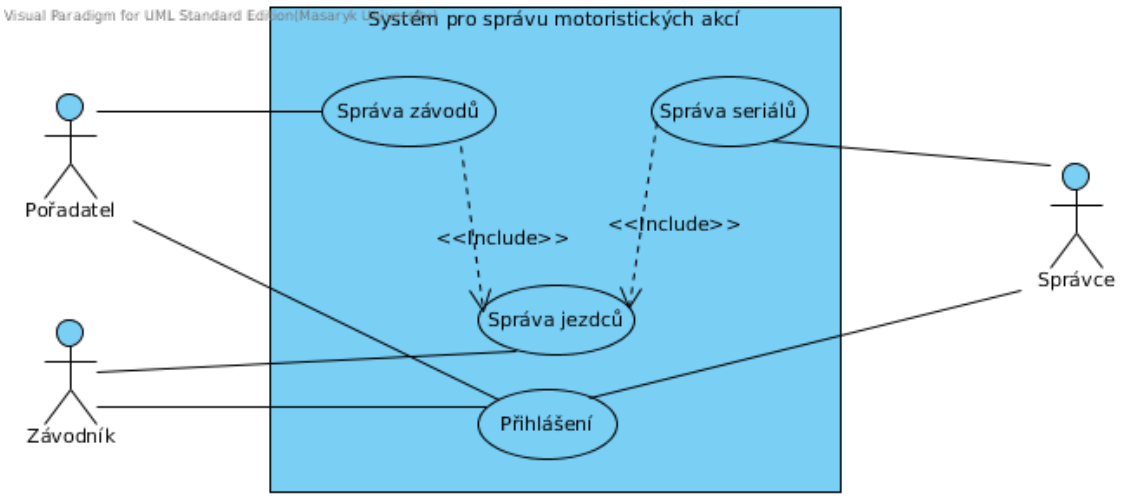
U některých soutěží se může stát, že není možné předem určit počet startujících jezdců, a proto se musí bodové ohodnocení upravit dynamicky podle jejich počtu. Obvykle se to děje způsobem, kdy první jezdec dostává počet bodů podle celkového počtu startujících jezdců. Druhý jezdec pak získává o jeden bod méně a tak dále, přičemž poslední jezdec je ohodnocen jedním bodem. Logika tohoto systému bodového ohodnocení vychází z principu počtu poražených ostatních závodníků v rámci daného závodu. Tento druh bodového ohodnocení soutěžících má přesnou vypovídací hodnotu o tom, jak obtížné bylo v dané soutěži získat to které konkrétní finální umístění. Závěrečný bodový výsledek totiž přesně uvádí, kolik ostatních závodníků musel ten který jezdec porazit, aby vybojoval právě ono konkrétní místo celkového pořadí.

Kapitola 2

Analýza a návrh

2.1 Případy užití

Use case diagram[12] neboli diagram užití představuje jeden z nejzákladnějších pohledů a úplně první pohled na vytvářený systém. Při jeho tvorbě vznikají definice základních pojmů, uživatelských rolí, rozsahu jejich práv a charakteristiky vzájemných interakcí mezi uživateli a systémem. V tomto konkrétním případě zobrazuje diagram 2.1 tři uživatelské role pro vytvářený systém. První z nich je role správce systému, který má za úkol udržoval základní chod systému. Druhou rolí je role pořadatele. Pořadatel organizuje jednotlivé soutěžní podniky a závody, zajišťuje průběh a chod závodů a vkládá základní data hodnocení do systému. Poslední z rolí je role jezdce. Jezdci je umožněno se do závodu přihlásit nebo se z něj odhlásit, zjistit své průběžné pořadí v jednotlivých závodech i ucelených seriálech. Jezdec je současně oprávněn vkládat, měnit a odstraňovat v systému svá osobní data.



Obrázek 2.1: Use case diagram

2.1.1 Správce systému

Tato role zahrnuje nejvyšší autoritu, která do systému může zasahovat. Jeho úkol spočívá ve správě, nastavení a údržbě soutěžních podniků a seriálů. Správce je oprávněn přidávat, odebírat, měnit či mazat tyto soutěžní podniky a seriály a provádět v nich základní nebo doplňující nastavení systému. Správa seriálů zahrnuje rovněž správu jednotlivých tříd, které budou v daném roce soutěžit. A v neposlední řadě přiděluje jednotlivé podniky seriálů konkrétním pořadatelům.

2.1.2 Pořadatel

Roli pořadatele v jednotlivých závodech seriálu určuje správce systému. Toto oprávnění umožňuje pořadateli organizaci a pořádání konkrétních soutěžních podniků v rámci seriálu. Pořadatel také shromažďuje výsledky jednotlivých soutěží a je jeho povinností tyto výsledky vkládat do systému v částech k tomu určených. Role pořadatele umožňuje organizátorům soutěží zobrazovat umístění jednotlivých jezdců, výsledky průběžného pořadí a další údaje, které organizátor pro potřeby organizace soutěžního podniku vyžaduje.

2.1.3 Závodník

Závodníkovi je umožněno registrovat se a přihlásit se do systému. Správce systému povoluje závodníkovi registraci do závodů pořádaných v rámci stanovených seriálů. Závodník je oprávněn prohlížet si dosavadní výsledky v rámci konkrétního závodu či celého seriálu, kterého se zúčastnil. V souvislosti s ustanoveními zákona na ochranu osobních údajů[1] je závodník oprávněn také měnit, vkládat nová, anebo mazat veškerá data, která mohou mít nebo mají povahu osobních informací.

2.2 Datový model

V současnosti je většina relevantních dat ukládána do relačních databází. Důvodem tohoto stavu je fakt, že tyto databáze jsou vybudovány na základě relačního kalkulu a jsou ověřeny dlouhodobým používáním nejen v bázích komerčních ale také nekomerčních. K reprezentaci datové struktury jsou zvoleny právě entitně-relační diagramy. S pomocí těchto diagramů je potom velmi snadné vytvořit schéma nad relační databází.

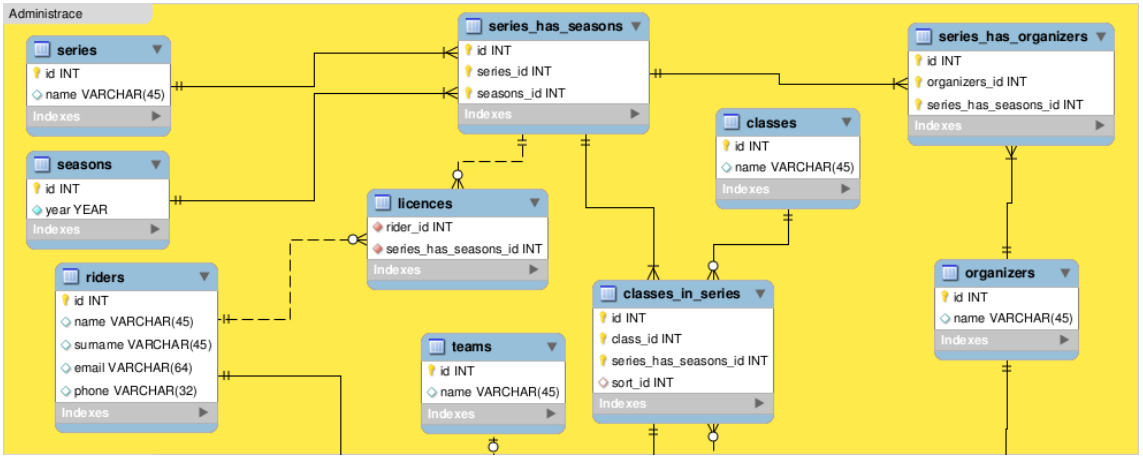
Celý model je rozdělen do čtyř ucelených částí. Jedná se o část administrativní, organizační, vyhodnocovací a část uživatelskou. Každá z nich může být implementována nezávisle na ostatních částech, podmínkou však je, že tato část bude implementována

1. Zákon č. 101/2000 Sb., o ochraně osobních údajů.

nad společným databázovým modelem. Jednotlivé části jsou popsány dále. Celý entitně-relační diagram je zobrazen v příloze A. Programátorskou zvyklostí je nazývat jednotlivé entity anglickými názvy v množném čísle. V dalším textu proto vedle českých názvů jednotlivých entit je uvedent také odpovídající název v jazyce anglickém.

2.2.1 Administrativní část

Administrativní část systému představuje veškeré činnosti administrátora, související především se správou jednotlivých soutěží a ročníků, výběrem a správou pořadatelů, kteří dané soutěže a soutěžní podniky organizují, kategorizací jezdců, kterým je podle druhů licencí umožněno se účastnit jednotlivých soutěží a v neposlední řadě stanovit jednotlivé třídy, které budou v daném seriálu soutěžit. Obrázek 2.2 charakterizuje jednotlivé entity, které se v této části systému vyskytují.



Obrázek 2.2: ERD - Administrace

Seriál (series)

Seriál představuje základní prvek v kategorizaci závodů a je nejvýše postavenou formou sportovního podniku. Seriál je tvořen nižšími celky, které se nazývají závody nebo soutěžní podniky. Doba trvání seriálu odpovídá délce jedné sezóny, což prakticky představuje jeden kalendářní rok, krácený o měsíce, v nichž není technicky možné soutěžní podniky realizovat. Každý seriál má svůj unikátní název.

Sezóna (seasons)

Sezóna obvykle představuje rok, ve kterém se seriály uskutečňují. V případě pořádání seriálu, jehož sezóna přesahuje období jednoho kalendářního roku, se považuje za rok

ten, ve kterém byl seriál zahájen.

Pořadatel (organizers)

Pořadatel je právnická nebo fyzická osoba, zajišťující faktickou realizaci, správu, organizaci, průběh, vyhodnocování a vyhlašování výsledků seriálu nebo jeho jednotlivých soutěžních podniků. Pořadatel je oprávněn v rámci konkrétního soutěžního podniku vypsat a zorganizovat soutěž pouze těch tříd, ktere jsou definovány v seriálu. Pořadatel má své jedinečné označení, kterým bývá zpravidla jméno a příjmení, nebo název organizace, v kombinaci s názvem obce, v níž soutěžní podnik proběhne.

Třída (classes)

Třída označuje skupinu jezdců, kteří vůči sobě navzájem vystupují jako rovnocení soupeři, přičemž je zohledňován především věk soutěžících nebo výkonnost závodních strojů. Každá stanovená třída je proto specifikována technickými parametry závodních strojů. Jako příklad lze uvést pořádání seriálu MotoGP, v němž jezdci startují ve třech třídách, 125ccm, Moto2 a MotoGP.

Jezdec (riders)

Jezdcem se rozumí fyzická osoba - závodník, který splňuje podmínky, stanovené pořadatelem, pro účast v závodě v dané třídě. Tato entita uchovává informace o osobě jezdce a to jméno, příjmení, emailovou adresu a kontaktní telefon. Jezdec představuje propojující klíč při vypočítávání výsledků celého seriálu.

Tým (teams)

Za tým jsou považováni ti jezdci, kteří se účastní soutěže nebo seriálu pod společným názvem. Název týmu je zpravidla tvořen obchodním jménem hlavního týmového sponzora. Tato entita má pouze organizační charakter, který žádným způsobem neovlivňuje hodnocení jezdců v závodech.

Sezóna seriálu (series_has_seasons)

Sezóna a seriál jsou propojeny pomocí této entity, která vyjadřuje období, zpravidla ročník v němž byl daný seriál pořádán.

Třídy v sezóně seriálu (classes_in_series)

Třídy v sezóně seriálu jsou kategorie soutěžních skupin vypsaných pro danou sezónu seriálu.

Pořadatel seriálu (series_has_organizers)

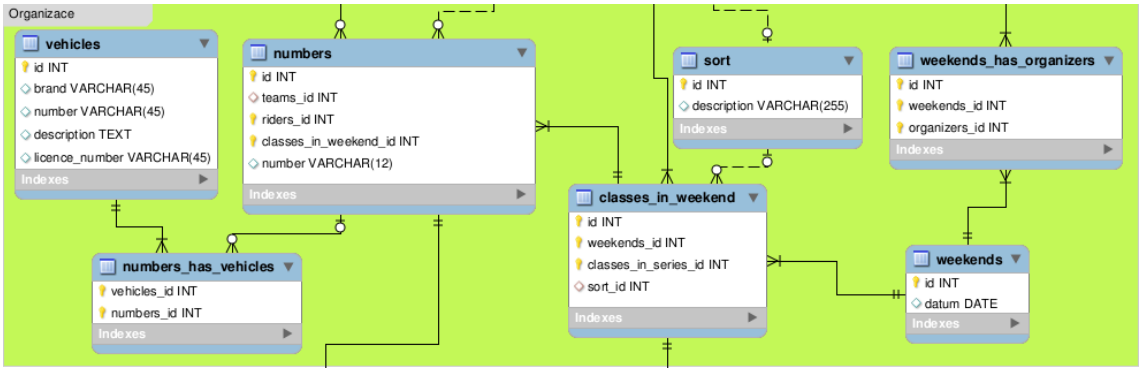
Organizovat soutěžní podniky jsou oprávněni pouze ti pořadatelé, kteří jsou jako pořadatelé vypsáni pro konkrétní seriál. Různé sezóny seriálů mohou mít různé pořadatele.

Licence (licences)

Licence predstavuje opravnění jezdce, účastnit se konkrétních seriálů při současném splnění ostatních specifikovaných podmínek.

2.2.2 Organizační část

Na rozdíl od administrativní části, která je využívána pouze na začátku soutěžní sezóny, organizační část využívají jednotliví organizátoři vždy na začátků konkrétního soutěžního podniku. Organizátoři vypisují soutěžní třídy jednotlivého soutěžního podniku, do nichž se registrují jezdci. Tito buď mají již přidělené startovní číslo nebo si jej mohou vybrat z těch startovních čísel, která nejsou v dáné soutěžní třídě již obsazena. Obrázek 2.3 popisuje přehledně všechny entity, které se v organizační části vyskytují.



Obrázek 2.3: ERD - Organizace

Stroj (vehicles)

Je závodní stroj (motocykl), na němž se jezdec účastní závodu. Tento údaj má význam pro zařazení jezdce do soutěžní kategorie. Informace o soutěžním stroji, tj. výrobce, popis (kubatura), homologační číslo a data technického průkazu jsou zapisovány do systému

a archivovány po celou dobu trvání soutěžního seriálu. Je přípustné, aby jezdec v rámci seriálu soutěžil na více strojích.

Startovní číslo (numbers)

Startovní číslo je unikátní údaj v soutěžní třídě, který indentifikuje osobu jezdce. Není proto možné, aby do jednoho závodu nastoupili jezdci se stejným startovním číslem. V případě, že jezdec závodí v různých třídách, nemusí mít stejné startovní číslo.

Startovní číslo stroje (numbers_has_vehicle)

Startovní číslo stroje určuje, který závodní stroj smí jezdec použít během závodů dané třídy.

Víkend (weekends)

Soutěžní podniky jsou souhrně označovány jako závodní víkendy, soutěžní víkendy nebo jen víkendy. Tento termín je běžně užíván v motocyklové terminologii a představuje organizaci soutěžního podniku jedné nebo více tříd, jednoho nebo více závodů v průběhu víkendu. Každý víkend je nutno označit datem zahájení.

Pořadatel víkendu (weekends_has_organizers)

Závodní víkend může být pořádán jedním nebo více pořadateli, kteří jsou evidováni v rámci seriálu.

Třídy o víkendu (classes_in_weekend)

Pro závodní víkend lze vypsat třídy, vypsané v rámci seriálu. Jezdci se pro účast v soutěžním podniku registrují do jednotlivých tříd o víkendu.

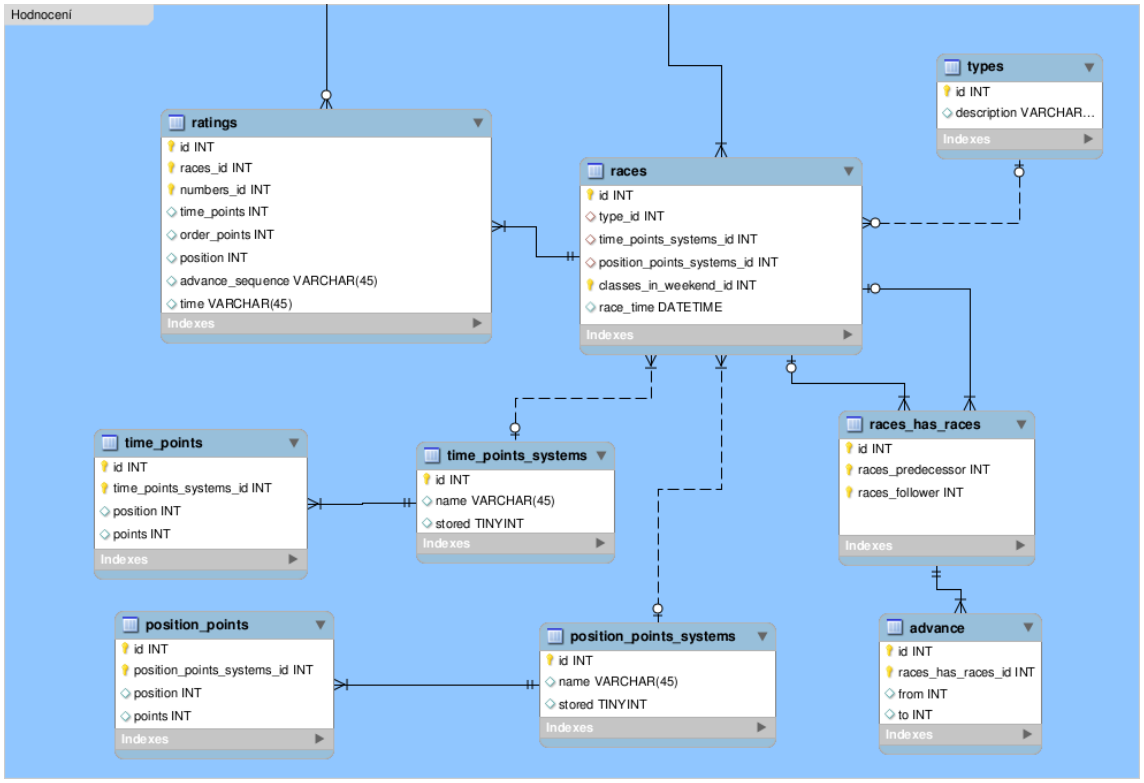
Třídění (sort)

Třídění jednotlivých výsledků se provádí podle třídícího algoritmu, který může být pro různé závody jiný. Tato entita určuje výběr použitého třídícího algoritmu. Má své prvky předdefinované a tyto není žádný uživatel oprávněn měnit.

2.2.3 Vyhodnocovací část

Vyhodnocovací část zpracovává výsledky specifického závodu pořádáného v rámci konkrétní třídy. Součástí vyhodnocení jsou vedle samotných závodů také kvalifikační roz-

jíždky. Jednotlivé závody mohou být vzájemně provázány a jejich organizační uspořádání znázorňuje to, které závody předcházejí ostatním. V každém závodu i kvalifikaci, lze přidělit jezdcům body podle dosažených výsledků. Jednotlivé entity vyhodnocovací části a jejich relace jsou popsány na obrázku 2.4.



Obrázek 2.4: ERD - Vyhodnocování

Závod (races)

Závod obsahuje informace o čase, ve kterém byl odstartován a třídu, v níž se konal. Nepovinnými informacemi v této entitě jsou údaje o bodových systémech, podle kterých jsou jezdci v závodě hodnoceni.

Typ (types)

Výběr nejvhodnějšího bodovacího mechanizmu provádí systém podle typu organizovaného závodu. Tato hodnota má zejména informační charakter s tím, že některé typy závodů nejsou vyhodnocovány a jejich výsledky nejsou archivovány.

Návaznost závodů (races_has_races)

Toto spojení vyjadřuje způsob, jakým na sebe závody navazují. Je možné je organizovat do navazující řady, pyramidy, která se používá jako vyřazovací způsob soutěže, ale též do pyramidy opačné, kde z jednoho závodu mohou jezdci postoupit do celé řady závodů dalších.

Postup (advance)

Postup jednoznačně stanoví klíč, podle kterého účastníci soutěžního podniku postupují mezi dvěma závody nebo kvalifikací a závodem.

Hodnocení (ratings)

Hodnocení uchovává informace o pořadí jezdců, v němž dokončili závod. Umožňuje ukládání jednotlivých dosažených časů každého jezdce. Součástí hodnocení je také počet přidělených bodů za jednotlivý závod podle umístění v závodě nebo podle dosaženého času.

Bodové systémy pořadí (position_points_system)

Bodové systémy představují převodník, použitý při udělování bodového ohodnocení jezdce podle dosaženého výsledku v závodě. Tvorba bodových systému není povinná, avšak její existence výrazně napomáhá práci s výsledky.

Bodové systémy čas (time_points_system)

Bododové systémy čas jsou obdobou bodových systémů pořadí, přičemž základním kritériem pro přidělování bodů je dosažený čas.

Body za pořadí (position_points)

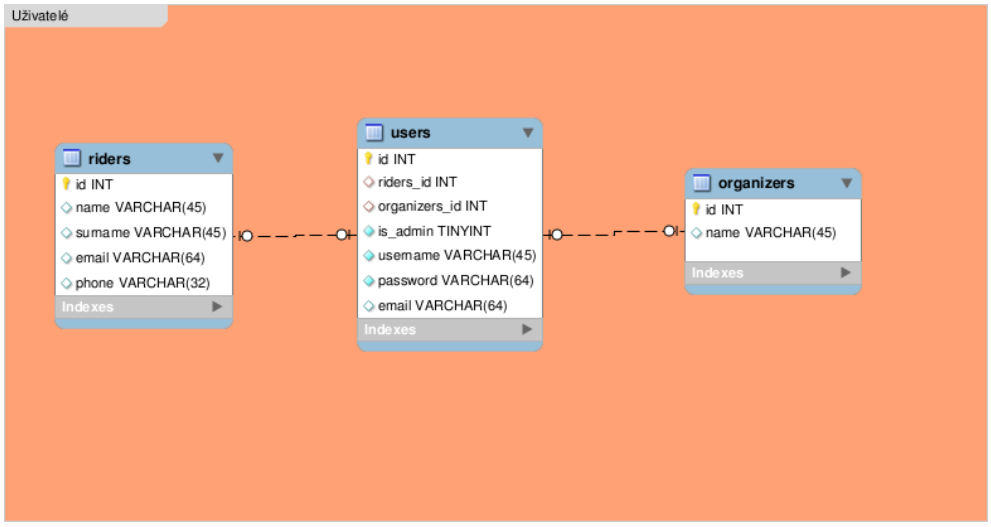
Tato entita definuje počet bodů, přidělovaný jezdců za dosažení konkrétního umístění v závodu. Jsou vázány na konkrétní bodový systém.

Body za časy (time_points)

Jsou analogií bodů za pořadí s tím rozdílem, že určujícím kritériem bodového ohodnocení jsou dosažené nejlepší časy závodu.

2.2.4 Uživatelská část

Uživatelská část představuje základní vstupní bránu do systému pro každého jednotlivého uživatele, kterému je následně přidělena role, jíž tento uživatel má v budoucnu zastávat. Součásti uživatelské části jsou: souhrn definic, charakteristik a oprávnění, specifikující jednotlivé uživatelské role.



Obrázek 2.5: ERD - Uživatelé

Uživatel (users)

Entita představuje souhrn dat uchovávajících informace o uživatelích systému. Uživatel přihlášením do systému získá ta oprávnění a role, které mu byly přiděleny správcem systému. Je přípustné, aby přihlášením do systému do systému uživatel získal více rolí. Typickým příkladem bude správce systému, který současně působí jako organizátor seriálu i jezdec. V případě kumulace rolí získává uživatel ta oprávnění k zásahům do systému, která jsou vyšší. U této entity se vždy uvádí, zda daný uživatel je nebo není správcem systému.

Pořadatel (organizers)

Informace, identifikující pořadatele, jsou ukládány do entity pořadatel. Charakteristika této entity je popsána v kapitole 2.2.1 na straně 11.

Jezdec (riders)

Tato entita přiřazuje uživateli roli jezdce. Blíže je tato entita popsána v kapitole 2.2.1 na straně 11.

Kapitola 3

Implementace

Implementované části systému lze zařadit do několika kategorií. Míra závislosti implementovaných částí systému na použitých technologiích je různá. Méně závislé části implementovaného systému jsou velmi snadno přenositelné na technologie jiné. Do této kategorie patří například návrh rozhraní pro jednotlivé třídy, starající se o chod systému. Veškeré návrhy rozhraní, vytvořené k aplikaci jsou v příloze B. V této kapitole jsou uvedeny a popsány implementace významných částí, které práci tvoří.

Systém jako celek má charakter webové aplikace. Hlavním důvodem této volby je snadná údržba jednotlivých verzí aplikace a především pak její velice snadná dostupnost díky míře rozšíření internetu. Jako implementační jazyk je použit skriptovací jazyk PHP[6] ve spojení se značkovacím jazykem HTML pro jeho jednoduchost, rychlost vykonávání příkazů a rozšíření na webových serverech.

Rozhraní modelové vrstvy aplikace je popsáno také pomocí jazyka PHP, což však pro rozhraní není klíčové, protože je velmi snadno převeditelné do jakéhokoli jiného imperativního jazyka, který rozhraní podporuje. Veškeré komentáře v rozhraních jsou podle programátorských zásad psány v angličtině. Všechna rozhraní, která se vztahují k entitám mají čtyři základní operace souhrnně označovány CRUD[1]. V práci jsou uvedeny ke každé operaci také prerekvizity, které musí být splněny pro úspěšné vykonání dané metody.

Prototyp webové aplikace je zpracován pomocí PHP aplikačního rámce CodeIgniter[7], což je aplikační rámec, využívající technologii MVC[2], která umožňuje rozdělit zdrojový kód do tří oddělených celků. V modelové vrstvě tvoří navržená rozhraní a jejich implementace. Tato vrstva komunikuje s databází. Není žádoucí, aby jakákoli aplikační logika byla vykonávána ve vrstně pohledu. Tato vrstva představuje prosté zobrazování dat definovaným způsobem. Přemostění těchto dvou vrstev zprostředkovává řadič, který určuje, která data mají být v požadovaném pohledu zobrazena. Existuje také celá řada jiných vlastností, díky kterým tento aplikační rámec vyniká oproti jiným. Tou nejdů-

1. Zkratka je odvozena od anglických názvů operací, které představují. Create (vytvořit), Read (číst), Update (aktualizovat), Delete (smazat)
2. Model-View-Controller. Česky tedy: „Model Pohled Řadič“

ležitější vlastností, která měla význam při volbě aplikačního rámce, je interoperabilita tohoto rámce s více databázovými systémy, zprostředkovávaná knihovnami a návrhovým vzorem Active Record[13]. Tento aplikační rámec v současné době podporuje většinu nejpoužívanějších databázových aplikací jako MySQL, MSSQL, SQLite, PostgreSQL nebo ODBC. Při použití těchto knihoven umožňuje implementaci systému nad jakoukoli z výše uvedených databází. Aplikační rámec CodeIgniter podléhá aktivnímu vývoji, když poslední verze 2.0.2 byla vydána v dubnu roku 2011[8]. Právě tyto skutečnosti skýtají možnost udržovat celý rámec aktualizovaný a zabezpečený podle aktuálních požadavků na ochranu dat.

3.1 Implementace administrativní části

Administrativní část tvoří návrhy rozhraní Int_Series, Int_Season, Int_Riders, Int_Classes, Int_Teams a Int_Organizers. Zdrojové kódy těchto rozhraní lze najít v příloze B této práce a nachází uplatnění především při zakládání seriálů a nových ročníků, což se děje zpravidla jednou za sezónu.

Rozhraní Int_Series

Je rozhraním pro soutěžní seriály, které obstarává správu seriálů a propojení seriálů se sezónami. Není možné odstranit jakýkoli seriál, pokud je již vypsán jeho ročník. Další možností je přiřazení nového ročníku k seriálu nebo jeho případné odstranění, za předpokladu, že neobsahuje žádné vypsané třídy.

Rozhraní Int_Seasons

Obsahuje metody práce s ročníky, přičemž vytvářet lze ty ročníky, které nebyly doposud vypsány. Odstranit lze pouze ty ročníky, které nebyly přiřazeny k některému seriálu.

Rozhraní Int_Classes

Int_Classes je rozhraním, umožňujícím správu tříd, vypisovaných v jednotlivých seriálech a závodních víkendech. Vedle standardních CRUD operací rozhraní Int_Clases obsahuje také metody pro přidávání tříd do seriálů. To zaručuje korektní vyhodnocení výsledků třídy pro celý seriál. Z tříd řazených k seriálu lze vypsat třídy pro závodní víkend. Není podmínkou, aby o závodním víkendu startovaly všechny definované třídy. U třídy v seriálu nebo v závodním víkendu je umožněn výběr porovnávacího algoritmu použitého při vyhodnocování výsledků. Více o porovnávacích algoritmech je uvedeno u rozhraní výsledků (Int_Results 3.3 na straně 23). Třídu nelze smazat, pokud je již vypsána v

jakémkoli seriálu či závodním víkendu.

Rozhraní Int_Riders

Umožňuje mimo standardní CRUD operace také přiřazení licence jednotlivým jezdcům, díky nimž se mohou sami registrovat do vypsaných tříd v závodních víkendech daného seriálu a účastnit se jich. Jezdce nelze smazat, pokud se již zúčastnil jakéhokoliv závodu.

Rozhraní Int_Teams

Rozhraní týmů slouží pouze pro formální uspořádání závodníků do týmů. Tým nelze odstranit pokud obsahuje nějakého závodníka.

Rozhraní Int_Organizers

Slouží pro evidenci a třídění organizátorů seriálů i jednotlivých soutěžních podniků. Organizátora nelze smazat, pokud již organizoval alespoň jeden soutěžní podnik.

3.2 Implementace organizační části

V organizační části jsou navržena rozhraní Int_Numbers, Int_Vehicles, Int_Weekends. Jejich implementace slouží k vytvoření závodních víkendů organizátory a vypsání závodních tříd pro daný víkend. Dále slouží k rozdělení a evidenci strojů, na kterých závodníci soutěží, startovních čísel závodníků, která mají přiřazena pro daný závodní víkend a jejich členěním v soutěžních třídách. Metody, které jsou uvedeny v této části využívají organizátoři při pořádání závodních víkendů.

Rozhraní Int_Numbers

Rozhraní Int_Numbers představuje souhrn metod použitých k přidělování startovních čísel jezdcům pro daný soutěžní podnik. Není přípustné, aby dva různí jezdci soutěžili v téže třídě se stejnými čísly. Číslo přiřazené jezdci není možné smazat, pokud již bylo použito v závodě. Pokud dojde k situaci, že se vyskytnou dvě stejná čísla v rámci jedné třídy, je použita metoda vyhledání nejvhodnějšího náhradního startovního čísla, založená na principu podobnosti čísel. Například k číslu 123 by se měla nabídnout čísla 1, 2, 3, 12 a 23, k číslu 45 čísla 4, 5, 145 a 451 za předpokladu, že nejsou již obsazená. Z praktického hlediska je nejsnazší změnit číslo přidáním jedničky. Zdrojový kód 3.1 ukazuje implementaci této metody.

Vstupními argumenty jsou startovní číslo a unikátní identifikační číslo třídy, ve které se zjišťují volná čísla [řádek 1]. Poté je vyloučeno použití jiných než numerických znaků

v argumentech [řádky 2-7]. Náhradní čísla jsou přidělována podle délky čísla nahrazovaného. Pro jednociferná čísla [řádek 9] se další získají přidáváním jedniček. Tedy podle vzorů 1X, X1, 11X, 1X1, X11, kde X je zkoumané jednociferné číslo [řádek 10]. Pro dvouciferná čísla [řádek 12] se získají další čísla nejen přidáváním jedniček, ale i odebráním jedné z číslic a jejich kombinací. Vzory jsou X, Y, 1X, X1, 1Y, Y1, 1XY, XY1, kde X je první a Y druhá cifra čísla [řádek 13]. Pro víceciferná čísla [řádek 15] je nepraktické vytvářet nová přidáváním dalších znaků, a proto se u nich zkoumají kombinace cifer při zachování jejich pořadí. Například pro trojciferné číslo XYZ, kde X představuje první, Y druhou a Z třetí cifru, budou zkoumána čísla X, Y, Z, XY a YZ [řádky 16-20]. U některých čísel se může stát, že vzory vygenerují stejná čísla a je zbytečné je předkládat dvakrát. Všechna duplicitní čísla [řádek 21] jsou proto vyloučena. Další kontrola je zaměřena na neobsazenost vygenerovaných čísel v rámci soutěžní třídy [řádek 23] pomocí metody isNumberFree, která je součástí rozhraní Int_Numbers a jeho implementace. Pokud by číslo obsazeno bylo, vyloučí se z nabízených [řádek 24]. Je nesmyslné předkládat číslo, které je již obsazené a nelze ho použít. Poté se čísla utřídí podle přirozeného pořadí [řádek 27] a jsou vrácena jako výsledek metody [řádek 28].

Zdrojový kód 3.1: Související čísla

```
1  public function getRelatedNumbers($number,$classId){
2      if(!isset($number) OR !is_numeric($number)) {
3          throw new ParamException('$number', '$number parameter
               is missing', 'Numbers_model_number_missing');
4      }
5      if(!isset($classId) OR !is_numeric($classId)) {
6          throw new ParamException('$classId', '$classId parameter
                is missing', 'Numbers_model_classId_missing');
7      }
8      $all_array = array();
9      if(strlen($number) == 1) {
10         array_push($all_array,"1".$number,$number."1","11".
               $number,"1".$number."1",$number."11");
11     }
12     elseif(strlen($number) == 2) {
13         $single = str_split($number);
14         array_push($all_array,$single[0],$single[1],"1".$single
               [0],$single[0]."1","1".$single[1],$single[1]."1","1".
               $number,$number."1");
```

```
15      } else {
16          for($i=1;$i<strlen($number);$i++){
17              for($j = 0; $j<=(strlen($number)-$i);$j++){
18                  array_push($all_array,substr($number,$j,$i));
19              }
20          }
21      }
22      $unique = array_unique($all_array);
23      foreach($unique as $key => $value){
24          if (!$this->isNumberFree($value,$classId)) {
25              unset($unique[$key]);
26          }
27      }
28      natsort($unique);
29      return $unique;
30  }
```

Rozhraní Int_Vehicles

Ke každému jezdci je možno pro konkrétní závodní víkend registrovat stroj, na kterém závodí. Na stejném stroji může závodit více jezdců v různých kategoriích. Tato situace není obvyklá, ale existují případy, kdy by mohla nastat.

Rozhraní Int_Weekends

U závodních víkendů se CRUD metody velmi liší. Při vytváření závodního víkendu organizátorem jsou vybrány požadované třídy a datum. Pokud závodní víkend v určené datum neexistuje, nejprve se vytvoří. K němu se dodatečně připojí vybrané třídy. Víkend nelze smazat, pokud obsahuje nějakou závodní třídu. Je však smazán automaticky, pokud se odebere poslední třída, která obsahuje tento víkend. Toto rozhraní úzce spolupracuje s rozhraním tříd (Int_Classes).

3.3 Implementace vyhodnocovací části

Vedle standardních metod a funkcí, které jsou obsaženy v části administrativní a části organizační tohoto systému, obsahuje část vyhodnocovací celou řadu metod doposud nepoužitých, které však pro svou důležitost a významovost představují základní pilíř využití tohoto navrhovaného systému v praxi.

Rozhraní Int_Races

K vytvoření nového záznamu Závod je nutno zadat informace o datumu a čase konání závodu a soutěžní třídě, v níž se závod pojede. Smazat záznam Závod lze za předpokladu, že tento neobsahuje ani jeden údaj o umístění závodníků.

Vedle základních operací CRUD jsou součástí tohoto rozhraní metody usnadňující orientaci v přehledech závodů. Použitelnost tohoto systému je významně rozšířena vzájemnou návazností jednotlivých závodů. Významným prvkem, který zajišťuje plynulost průběhu každého závodu, je schopnost systému vytvořit listinu o startovním rozestavění jezdců. Je ale žádoucí, aby obsah této listiny byl stručný, srozumitelný a jednoduše interpretovatelný.

Rozhraní Int_PointSystems

Bodové systémy neboli způsob, jakým jsou jednotliví jezdci ohodnoceni podle svého umístění, se závod od závodu liší. Odlišnosti různých typů závodů jsou v obecné rovině popsány v úvodní kapitole. Ačkoli platí zásada, že systém bodování zůstává v průběhu seriálu neměnný, je možno provést jeho změnu ve všech případěch, kdy tato změna bude žádoucí (klasickým příkladem jsou vyřazovací rozjížďky, rychlostní rozjížďky a vícenásobné kvalifikace). Systém využívá dva druhy bodového hodnocení. Prvním z nich je hodnocení za dosažené časy, druhým je pak hodnocení za dosažené pořadí. Rozdělením obou výše uvedených bodovacích systémů do dvou samostatných tabulek, odpadá potřeba provádění kontroly duplicity v aplikační vrstvě. Mazat bodové systémy lze pouze za předpokladu, že tyto nebyly ani v jednom případě použity.

Rozhraní Int_Ratings

Rozhraní Int_Ratings nabízí různé metody přiřazování bodů jezdcům a to v návaznosti na výše uvedené bodové systémy. Rozšiřujícím prvkem, který Int_Ratings nabízí, je doplňování pořadí jezdců podle vkládaných časových údajů, nebo importu sekvence čísel, označujících pořadí průjezdu závodníků cílovým prostorem v průběhu každého kola. Návrh implementace této metody je popsán ve zdrojovém kódu 3.6 na straně 33.

Rozhraní Int_Results

Hraje primární úlohu v samotném vyhodnocování výsledků. Tyto výsledky jsou zobrazovány ve třech úrovních. Nejvyšší úrovní je zobrazování výsledků konkrétní třídy v rámci celého seriálu. O úroveň níže jsou pak všechny výsledky závodů konkrétní třídy v rámci jednotlivého soutěžního víkendu. Nejnižší úroveň představuje vyhodnocení vý-

sledků daného závodu v rámci jedné konkrétní třídy. Zobrazení výsledku samotného závodu je přímočaré a vždy se jedná o zobrazení dat uložených do databáze podle dosaženého pořadí jezdců. Při vyhodnocování tříd v rámci soutěžního víkendu může nastat konflikt ve shodě dosaženého bodového ohodnocení.

Nejjednodušší postup řešení tohoto konfliktu spočívá v přidělení stejného umístění jezdcům, kteří dosáhli této shody bodů v rámci soutěžního víkendu. Tato varianta připadá v úvahu zpravidla v těch případech, kdy se nejedná o tzv. pohárová místa. Tabulka 3.1 znázorňuje právě takovouto situaci.

Pořadí	Body
11.	15
12.	13
13.	11
13.	11
15.	7

Tabulka 3.1: Řešení konfliku bodového hodnocení

Existují však případy, kdy shora uvedené řešení konfliktu není žádoucí. Ukázkovým příkladem může být umístění dvou jezdců na tzv. pohárovém místě, tedy na prvním až třetím místě hodnocení výsledků závodu třídy v rámci soutěžního víkendu. Je tomu tak proto, neboť by organizátor soutěžního víkendu musel vždy predikovat takovouto situaci a v návaznosti na ni obstarat pro první až třetí místo dvoj nebo vícenásobné množství ocenění soutěžících (dva a nebo více vavřínových věnců, pohárů a věcných cen pro první druhé a třetí místo, přičemž většinu těchto cen není možno udělit v závodě jiném). Z výše uvedených důvodů proto organizátoři závodu mohou využít alternativní způsob řešení konfliktu, který spočívá v přidělení unikátního místa výsledku závodů, každému jednotlivému jezdci. Toho lze dosáhnout nastavením třídícího algoritmu pro požadovanou třídu. Tyto třídící algoritmy jsou předem definovány a bez zásahu do zdrojového kódu není možno měnit způsob jejich vyhodnocování. Pokud by nastala situace, kdy by bylo potřeba doplnit systém o nový algoritmus, je nutno dodatečně upravit zdrojový kód a obsah databáze.

Na zdrojovém kódu 3.2 na následující straně je znázorněn porovnávací algoritmus pro motokrosové závody. Použitá metoda požaduje jako své argumenty dvě konkrétní entity, v tomto případě třídy typu „Races_results“ popsané v implementaci rozhraní v příloze B [řádek 1]. Návratová hodnota metody pak určí, která z těchto dvou entit je větší, popřípadě zda jsou si rovny. Při vrácení záporného čísla je první entita ($a) menší než druhá ($b). V případě vrácení kladného čísla je první entita ($a) větší než druhá ($b).

Vrácení nuly znázorňuje jejich rovnost. V tomto případě porovnáme, která z těchto entit má větší počet celkových bodů a výsledek uložíme do proměnné [řádek 2]. Není možné vrátit tuto hodnotu přímo, neboť by jezdci byli seřazeni vzestupně, od nejnižšího bodování po nejvyšší. Z této hodnoty současně zjistíme, zda nedošlo k rovnosti bodů u porovnávaných subjektů [řádek 3]. V případě rovnosti bodového hodnocení v rámci závodního víkendu, jsou porovnány výsledky dosažené v posledním závodu daného soutěžního víkendu a je vrácen příslušný výsledek [řádek 4]. Obrácená hodnota porovnávaných výsledků je vrácena v těch případech, kdy k rovnosti bodů nedošlo. [řádek 5 a 6].

Příklad vyhodnocení shody bodů vyjadřuje obrázek 3.1 na následující straně. V tomto případě se nejedná o simulovanou situaci a použitá data vychází z výsledků závodu seriálu Silesian Cup 2010[9], třída MX50, závodní víkend pořádaný 29. 5. 2010 v Budišově nad Budišovkou. Natálie Burdová získala v druhém závodu více bodů, než Aleš Pražák. Ve vyhodnocení tříd podle motokrosových pravidel se tedy umístila na lepší pozici, přestože mají oba stejný počet bodů.

Zdrojový kód 3.2: Porovnávací algoritmus pro motokrosové závody

```
1 public function compare_motocross($a, $b) {
2     $compare = (strnatcmp($a->getTotalPoints(), $b->
          getTotalPoints()));
3     if ($compare == 0) {
4         return (strnatcmp($a->getLastPoints(), $b->getLastPoints
              ()) > 0 ? 1 : -1);
5     } else
6         return ($compare > 0 ? -1 : 1);
7 }
```

3.4 Použité algoritmy

Implementace vyhodnocovací části je definována v kapitole 3.3. Unikátnost a nezastupitelnost této části je založena na specifických algoritmech, které jsou v navrhovaném systému stěžejní. Právě jim je věnována tato kapitola.

3.4.1 Zobrazení závodních víkendů v sezóně seriálu

Bez znalosti unikátního identifikačního čísla sezóny seriálu není možné získat zobrazení jednotlivých závodních víkendů, které tuto sezónu tvoří. Zdrojový kód 3.3 představuje

Results

Position	Name	Surname	2010-05-29 11:00:00	2010-05-29 12:00:00	Total	Actions
1	Adam	Straka	6	6	**12**	
2	Jakub	Uher	5	5	**10**	
3	Natálie	Burdová	3	4	**7**	
4	Aleš	Pražák	4	3	**7**	
5	Šimon	Tichý	2	2	**4**	
6	Sebastian	Klemens	1	1	**2**	

Obrázek 3.1: Ukázka vyhodnocení výsledků s použitím porovnávacího algoritmu při shodě bodů.

implementaci právě této metody.

Vstupním parametrem je zmíněné unikátní identifikační číslo sezóny seriálu. Za předpokladu, že vstupní parametr není vyplněn případně pokud nenabývá numerických hodnot, dojde k vyhození výjimky s chybovou hláškou, pokud by tento parametr nebyl vyplněn nebo nenabýval číselných hodnot [řádky 2-4]. Následně jsou z databáze vybrány takové třídy v seriálu, které jsou vypsány v sezóně seriálu [řádky 5-7] a provedena kontrola jejich existence [řádky 8-10]. V dalším kroku dochází k výběru třídy stejného typu vypsané v závodním víkendu [řádky 11-13], přičemž následující úkon představuje kontrolu toho, zda závodní víkend alespoň jednu tuto třídu obsahuje [řádky 14-16]. Po výběru datumů, v nichž soutěžní víkendy proběhly a jejich ověření [řádky 17-22], vrací metoda jako výsledek seznam těchto víkendů [řádek 23]. Tato situace je vidět na obrázku 3.3. Jako podkladová data pro tento obrázek byly použity údaje prvních čtyř závodních víkendů seriálu Silesian CUP 2010 [9].

Zdrojový kód 3.3: Zobrazení závodních víkendů v sezóně seriálu

```
1  public function showWeekendsBySeries($id) {
2      if(!isset($id) OR !is_numeric($id)) {
3          throw new ParamException('$id', '$id parameter is
               missing', 'Weekends_model_id_missing');
4      }
5      $query = $this->db->select('id')
6          ->where(array('series_has_seasons_id'=>$id))
7          ->get('classes_in_series');
8      if($query->num_rows == 0) {
9          throw new MethodException('There are no classes for this
               season', 'Weekends_model_no_classes');
10     }
```

```
11      $query2 = $this->db->select('weekends_id')
12          ->where_in('classes_in_series_id', array_flatten($query
                ->result_array()))
13          ->get('classes_in_weekend');
14      if($query2->num_rows == 0) {
15          throw new MethodException('There are no classes in
                weekends for this season', 'Weekends_model_no_classes
                ');
16      }
17      $query3 = $this->db->select('id, datum')
18          ->where_in('id', array_flatten($query2->result_array()))
19          ->get('weekends');
20      if($query3->num_rows == 0) {
21          throw new MethodException('There are no weekends for
                this season', 'Weekends_model_no_weekends');
22      }
23      return $query3->result();
24  }
```

Weekends in series

ID	Date	Actions
1	2010-05-29	
2	2010-06-13	
3	2010-06-27	
4	2010-07-10	

Obrázek 3.2: Zobrazení prvních čtyř závodních víkendů v sezóně 2010 seriálu Silesian Cup[9]

3.4.2 Vyhodnocení výsledků závodu

Zdrojový kód 3.4 zobrazuje možnou implementaci metody, která slouží k získání výsledků závodu.

Vstupním parametrem této metody je unikátní číselný identifikátor závodu, jehož výsledky mají být vyhodnoceny [řádek 1]. Následuje kontrola, zda uvedený identifikátor je číslo. Pokud tomu tak není, je vyhozena patřičná výjimka [řádky 2-4]. Dále je proveden výběr bodových ohodnocení jezdců, kteří se závodu zúčastnili a seřazení jezdců podle

dosaženého umístění [řádky 5-7]. To, zda výsledky pro daný závod byly nebo doposud nebyly zadány, provede kontola existence hodnot [řádky 8-10]. Pro každý z nalezených výsledků [řádky 12-24] jsou dohledána konkrétní jména, příjmení a startovní čísla jezdců, kteří se tohoto závodu účastnili [řádky 14-23]. Získaná data jsou vrácena jako výsledek metody [řádek 25]. Zobrazení výsledku takovéhoto algoritmu je demostrováno na obrázku 3.3 na následující straně. Uvedená data vychází z výsledku závodu soutěžní třídy MX 50, seriálu Silesian Cup 2010, který byl pořádán dne 10. 7. 2010 ve Velkých Heraltických.

Zdrojový kód 3.4: Vyhodnocení výsledků závodu

```
1  public function getRaceResults($races_id) {
2      if (!isset($races_id) OR !is_numeric($races_id)) {
3          throw new ParamException('$id', '$id parameter is
               missing', 'Results_model_id_missing');
4      }
5      $query = $this->db->where(array('races_id' => $races_id))
6          ->order_by("position", "asc")
7          ->get('ratings');
8      if ($query->num_rows() == 0) {
9          throw new MethodException('There are no results for this
               race', 'Results_model_no_ratings_in_race');
10     }
11     $return = array();
12     foreach ($query->result_array() as $row) {
13         $id = $row['numbers_id'];
14         $query2 = $this->db->select('riders.name, riders.surname
               , numbers.number')
15             ->from('numbers')
16             ->join('riders', 'riders.id = numbers.riders_id')
17             ->where(array("numbers.id" => $id))
18             ->get();
19         $row2 = $query2->row();
20         $row['name'] = $row2->name;
21         $row['surname'] = $row2->surname;
22         $row['number'] = $row2->number;
23         array_push($return, (object) $row);
24     }
```

```
25      return $return;
26  }
```

Results

Position	Name	Surname	Starting Number	Points (T)	Points (P)	Total	Actions
1	Adam	Straka	7		7	**7**	
2	Jakub	Uher	24		6	**6**	
3	Aleš	Pražák	71		5	**5**	
4	Natálie	Burdová	22		4	**4**	
5	Josef	Bartusek	76		3	**3**	
6	Sebastian	Klemens	19		2	**2**	
7	Adam	Gibes	46		1	**1**	

Obrázek 3.3: Vyhodnocení výsledků závodu

3.4.3 Vyhodnocení výsledků třídy za závodní víkend

Vyhodnocení výsledků tříd v rámci závodního víkendu komplikuje nepravidelnost počtu uskutečněných závodů v rámci konkrétní třídy, ale také rozdíly v počtu jezdců, kteří se závodu účastní. Zdrojový kód 3.5 na následující straně představuje možnou implementaci. Jeho obsah tvoří méně složité dotazy na databázi, které jsou interpretovány ve všech databázích stejně, čímž je dosaženo kompatibility kódu.

Metoda pro výpočet výsledků třídy v závodním víkendu má jako parametr unikátní číselný identifikátor této třídy [řádek 1]. Pokud unikátní identifikátor není číselného typu, je vyhozena patřičná výjimka [řádek 2-4]. Dalším krokem je výběr jezdců, kteří se zúčastnili alespoň jednoho závodu v dané třídě. Toho lze dosáhnout následujícím postupem. Je proveden výběr všech závodů, které se v dané třídě během závodního víkendu odjely [řádky 5-7]. Výjimka je vyhozena v tom případě, pokud by žádný takový závod nemohl být vybrán [řádky 8-10]. Následuje výběr všech startovních čísel jezdců, kteří se závodu účastnili [řádky 11-13]. Výjimka je opět vyhozena v případě, jestliže neexistuje ve vybraném závodě žádné hodnocení [řádky 14-16]. K vybraným startovním číslům jsou přiřazeni jezdci [řádky 17-19]. Přestože provedení kontroly jezdců není nutné, je z opatrnosti tato kontrola provedena [řádky 20-22]. Chyba může v tomto případě vzniknout například jako důsledek výpadku elektrického proudu. Poté jsou vybrána unikátní identifikační čísla, jména a příjmení jezdců podle nalezených záznamů z předchozího dotazu [řádky 23-25]. Není nezbytně nutné kontrolovat, zda i tento dotaz vrátil nějaký výsledek, ale i zde je kontrola z opatrnosti provedena a případně vyhozena výjimka [řádky 26-28].

Následuje vytvoření instance třídy Class_results, jejíž implementace je popsána v pří-

loze B [řádek 29]. Následuje výběr všech závodů, pořádaných v této soutěžní třídě [řádky 30-32]. I zde je provedena kontrola existence hodnot [řádky 33-35]. Pro každý takto nalezený závod [řádky 36-44] jsou vybráni jezdci a jejich bodové hodnocení [řádky 37-39]. Neexistence hodnocení jezdce v žádném vybraném závodu způsobí vyhození výjimky [řádky 40-42]. Metoda add_races přiřadí jednotlivým jezdcům výsledky nalezených závodů [řádek 43]. Implementace této metody je v příloze B. Podle definovaných kriterií jsou výsledky setříděny [řádek 45]. Výsledkem této metody je seznam získaných hodnocení [řádek 46]. Možné zobrazení výsledků je znázorněno na obrázku 3.1 na straně 26.

Zdrojový kód 3.5: Vyhodnocení výsledků třídy za závodní víkend

```
1  public function getClassResults($classes_in_weekend_id) {
2      if (!isset($classes_in_weekend_id) OR !is_numeric(
           $classes_in_weekend_id)) {
3          throw new ParamException('$id', '$id parameter is
               missing', 'Results_model_id_missing');
4      }
5      $query = $this->db->select('id')
6          ->where(array('classes_in_weekend_id' =>
               $classes_in_weekend_id))
7          ->get('races');
8      if ($query->num_rows() == 0) {
9          throw new MethodException('There are no results for this
               race', 'Results_model_no_classes_in_weekend');
10     }
11     $query2 = $this->db->select('numbers_id')
12         ->where_in('races_id', array_flatten($query->
               result_array()))
13         ->get('ratings');
14     if ($query2->num_rows() == 0) {
15         throw new MethodException('There are no results for this
               race', 'Results_model_no_ratings');
16     }
17     $query3 = $this->db->select('riders_id')
18         ->where_in('id', array_flatten($query2->result_array()))
19         ->get('numbers');
20     if ($query3->num_rows() == 0) {
21         throw new MethodException('There are no results for this
```

```
            race', 'Results_model_no_numbers');
22      }
23      $query4 = $this->db->select('id, name, surname')
24          ->where_in('id', array_flatten($query3->result_array()))
25          ->get('riders');
26      if ($query4->num_rows() == 0) {
27          throw new MethodException('There are no results for this
                race', 'Results_model_no_riders');
28      }
29      $results = new Class_results($query4->result());
30      $query6 = $this->db->select('id, race_time')
31          ->where(array('classes_in_weekend_id' =>
               $classes_in_weekend_id))
32          ->get('races');
33      if ($query6->num_rows() == 0) {
34          throw new MethodException('There are no results for this
                race', 'Results_model_no_races_in_class');
35      }
36      foreach ($query6->result() as $row) {
37          $query5 = $this->db->select("riders.id, riders.name,
               riders.surname")
38              ->select("(IFNULL(ratings.time_points, 0 ) + IFNULL(
                   ratings.order_points, 0 )) AS points",FALSE)
39              ->from('riders')
40              ->join('numbers', 'numbers.riders_id = riders.id', '
                   left outer')
41              ->join('ratings', 'ratings.numbers_id = numbers.id',
                    'left outer')
42              ->where(array('ratings.races_id' => $row->id))
43              ->order_by('points', 'desc')
44              ->get();
45          if ($query5->num_rows() == 0) {
46              throw new MethodException('There are no results for
                   this race', 'Results_model_no_races_in_class');
47          }
48          $results->add_races($row->id, $row->race_time, $query5->
               result());
```

```
49      }
50      $results->sortByLast();
51      return $results;
52  }
```

3.4.4 Import a kontrola výsledků závodu

Aby bylo možno vyhodnotit umístění jezdců, je zapotřebí shromáždit a zpracovat celou řadu dat a údajů, získaných během konání sportovního podniku. Automatický sběr a zpracování dat jsou nejefektivnější formou vyhodnocování výsledků, která minimalizuje chyby způsobené zpravidla lidským faktorem. Dalším významným aspektem je rychlost sběru a především pak vyhodnocení získaných dat. Data jsou zpracována a uchována v digitální podobě. Díky tomu je možno data zasílat, exportovat, importovat a konvertovat do nejrůznějších formátů. Plně automatizované zpracování dat je ideálním řešením, k němuž by měl vývoj obecně směřovat, a návrh tohoto systému je právě jedním z takovýchto kroků.

Veškerá data, která jsou systematicky sbírána, je nutno také zpracovat. Sběr dat zpravidla probíhá na jiném místě než jejich zpracování a je nutné vytvořit mechanizmus nebo postupy, které import shromážděných dat obstarají. Za jeden z těchto postupů lze považovat následující algoritmus.

Tento nejprve zjistí podle zadaného závodu startovní čísla, která se jej účastnila. Z vložených dat stanoví pořadí jezdců, načte bodovací systém daného závodu a podle něj přiřadí závodníkům odpovídající bodové ohodnocení. Nakonec všechny tyto výsledky uloží do databáze. Implementace tohoto algoritmu je zobrazena ve zdrojovém kódu 3.6 a popsána dále.

Jako první parametr metody pro vyhodnocení výsledků závodu je použit unikátní číselný identifikátor závodu, pro nejž jsou výsledky vyhodnocovány. Druhým parametrem jsou metodě předána uspořádaná data, která popisují jednotlivé průjezdy závodníků cílovým prostorem v rámci závodu. Pořadí těchto dat nelze v žádném případě měnit. Tato data mohou mít podobu uspořádaného pole nebo řetězce čísel oddělených jiným znakem. Pro tuto metodu byla zvolena čárka, jako oddělovací znak. Předáním třetího parametru je možné oddělovací znak nahradit jiným [řádek 1]. V těch případech, kdy první parametr nebude mít číselnou podobu, je vyhozena výjimka [řádky 2-4]. Dalším krokem je získání informací o tom, která všechna startovní čísla se závodu zúčastnila [řádky 5-9]. Za předpokladu, že v závodu žádné startovní číslo není uvedeno, je znovu vyhozena výjimka [řádky 10-12]. Metoda poté vytvoří pole obsahující čísla, která se závodu účastnila a pole obsahující unikátní identifikátor těchto čísel pro jejich pozdější ukládání do

databáze [řádky 13-18]. Následně jsou tato čísla seřazena [řádek 19].

Je nutno zjistit, v jakém formátu jsou importovaná data uložena. Data, která nemají strukturu pole, jsou na tuto strukturu převedena pomocí definovaného oddělovače [řádky 20-22]. Importovaná data jsou poté zpracována do podoby, ze které je možné přímo určit pořadí jezdců. Toho je dosaženo pomocí dvourozměrného pole, kde prvním rozměrem pole budou řádky vyjadřující počet kol, které daný jezdec ujel v rámci jednoho závodu a druhým rozměrem budou sloupce vyjadřující pořadí jednotlivých jezdců, v němž projeli dané kolo [řádky 25-34]. Z takto strukturovaného záznamu je možné jednoznačně určit výsledné pozice všech jezdců daného závodu [řádky 36-46].

Pokud načtení bodovacího systému [řádky 47-53] není zakončeno vyhozením výjimky [řádky 54-56], provede metoda bodové ohodnocení jezdců podle zjištěných pozic jednotlivých jezdců. Tu část jezdců, která se umístila pod hranicí přidělování bodů, systém ohodnotí nula body [řádky 58-72].

Po předchozím výpočtu jsou získaná bodová ohodnocení jednotlivých jezdců uložena do databáze [řádky 73-81].

Zdrojový kód 3.6: Import a kontrola výsledků závodu

```
1  public function importResults($raceId,$data,$separator = ",") {
2      if(!isset($raceId) OR !is_numeric($raceId)) {
3          throw new ParamException('$id', '$id parameter is
               missing', 'Results_model_id_missing');
4      }
5      $query = $this->db->select('numbers.number,numbers.id')
6          ->from('numbers')
7          ->join('ratings', 'numbers.id = ratings.numbers_id')
8          ->where(array('ratings.races_id' => $races_id))
9          ->get();
10     if ($query->num_rows() == 0) {
11         throw new MethodException('There are no numbers for this
               race', 'Results_model_no_numbers_in_race');
12     }
13     $numbers = array();
14     $lookup = array();
15     foreach ($q->result() as $value) {
16         array_push($numbers,$value->number);
17         $lookup[$value->number] = $value->id;
18     }
```

```
19      natsort($numbers);
20      if(!is_array($data)) {
21          $data = explode($separator,$data);
22      }
23      $results = array();
24      $results[0] = array();
25      foreach ($data as $number) {
26          $i=0;
27          while(in_array($number,$results[$i])) {
28              $i++;
29              if(count($results) >= $i) {
30                  $results[$i] = array();
31              }
32          }
33          array_push($results[$i],$number);
34      }
35      $position = array();
36      for($j=count($results)-1;$j<=0;j--){
37          for($k=0;$k<count($results[$j]);$k++){
38              if(!in_array($position,$results[$j][$k])){
39                  array_push($results[$j][$k]);
40              }
41              if(count($position)==count($numbers))
42                  break;
43          }
44          if(count($position)==count($numbers))
45              break;
46      }
47      $query2 = $this->db->select('position_points.position,
           position_points.points')
48          ->from('position_points')
49          ->join('position_points_systems', '
               position_points_systems.id = position_points.
               position_points_systems_id')
50          ->join('races', 'position_points_systems.id = races.
               position_points_systems_id')
51          ->where(array('races.id'=>$raceId))
```

```
52          ->order_by('position_points.position','asc')
53          ->get();
54      if ($query2->num_rows() == 0) {
55          throw new MethodException('There is no position system
               for this race', 'Results_model_no_position_system');
56      }
57      $points = array();
58      foreach($position as $key => $value){
59          $pos = $key+1;
60          $racer['position'] = $pos;
61          $racer['number'] = $value;
62          $racer['number_id'] = $lookup[$value];
63          foreach($query2->result() as $row) {
64              if($row->position == $pos) {
65                  $racer['points'] = $row->points;
66              }
67          }
68          if(!isset($racer['points]) {
69              $racer['points'] = 0;
70          }
71          array_push($points,(object)$racer);
72      }
73      foreach($points as $ratings){
74          $data = array(
75              'races_id' => $raceID,
76              'numbers_id' => $ratings->number_id,
77              'order_points' => $ratings->points,
78              'position' => $ratings->position
79          );
80          $this->db->insert('ratings', $data);
81      }
82  }
```

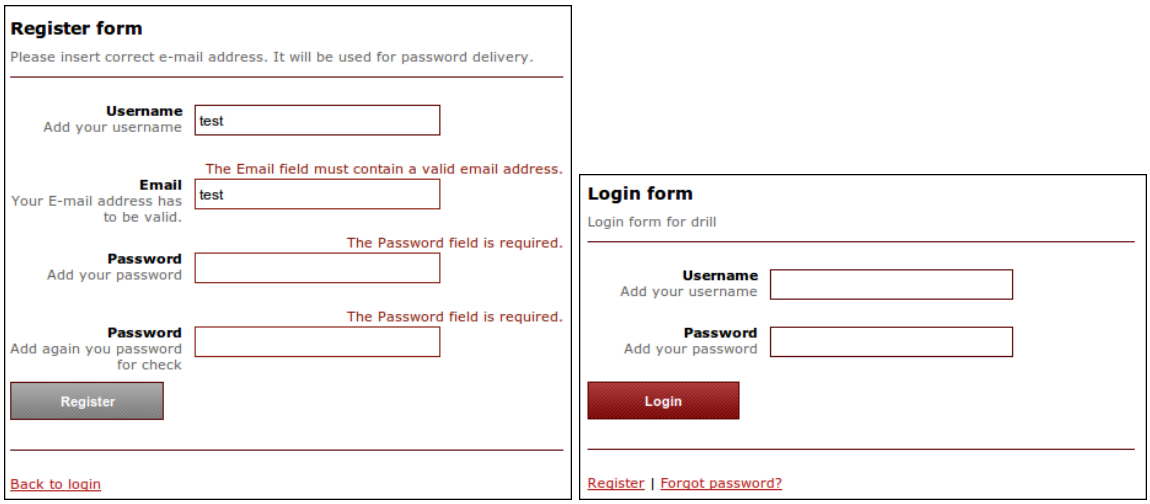
Kapitola 4

Funkcionalita

Obsah kapitoly 4 je zaměřen na podrobnější popis prototypu navrhovaného systému. Některé jeho funkce a jejich ukázky již byly popsány v předchozích kapitolách. Jako celek je prototyp popsán a jeho funkce demonstrovány v kapitole této. Veškeré níže popsané funkce mohou být s malými úpravami použity při budování ostatních částí systému.

4.1 Registrace a přihlášení

Do systému mohou vstupovat pouze uživatelé, kteří mají registrovaný účet. Ten se vytváří vyplněním a odesláním registračního formuláře, do kterého registrující se uživatel zadává uživatelské jméno, kontaktní e-mail a heslo, které musí být alespoň 8 znaků dlouhé. Na obrázku 4.1a je zobrazen návrh registračního formuláře společně s vyobrazením polí, do kterých se jednotlivé údaje zapisují a jejichž hodnoty jsou kontrolovány a musí být tedy validní.



(a) Registrace (b) Přihlášení

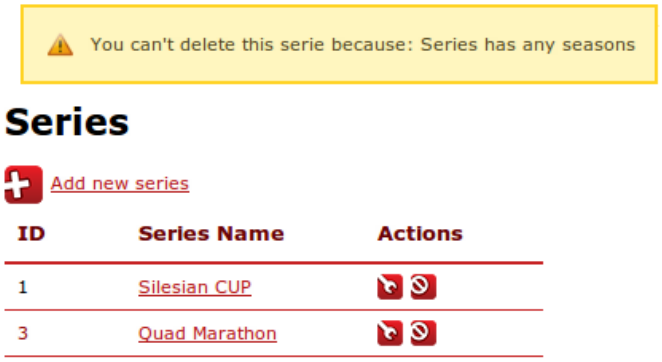
Po provedené registraci je uživateli vytvořen jeho účet pro přihlášení do systému. Vstup do systému umožňuje registrovanému uživateli prohlížet výsledky jednotlivých seriálů. Aby uživatel mohl přidávat, upravovat nebo mazat záznamy v systému, musí

mít oprávnění správce, jezdce nebo pořadatele. Tyto role přiřazuje správce systému. Příklad přihlašovací obrazovky je na obrázku 4.1b na předchozí straně. Po zadání platných údajů do přihlašovacího formuláře je uživatel přihlášen do systému.

4.2 Přidávání, úprava a mazání

Pokud má přihlášený uživatel současně oprávnění správce, je mu umožněno vytvořit například nový seriál nebo jej editovat. K vytváření rozhraní systému slouží předem definovaný formulář. Validita údajů, zadávaných prostřednictvím formuláře, je systémem kontrolována.

Stejně, jako je umožněno do systému data ukládat, je také možno data ze systému odstraňovat. Každé rozhraní pracující s jinou entitou má definovány individuální podmínky, jejichž splnění podmiňuje odstraňování údajů ze systému. Jako příklad shora uvedeného lze demostrovat postup při odstraňování Seriálu. Po kliknutí na ikonku pro mazání označenou „Ø“, se zobrazí informativní hláška, která oznamuje, že seriál nelze odstranit pokud obsahuje jakoukoli sezónu. Tato situace je vyobrazena na obrázku 4.1.



Obrázek 4.1: Mazání seriálu

4.3 Propojování

K propojení dvou entit v relaci M:N se využívají pomocné tabulky. Těmi se znázorňují, která data z entity první jsou navázána na data z entity druhé. Vezmeme-li v úvahu entity sezóna a seriál, pak jejich pomocná propojovací tabulka znázorňuje seriály, ke kterým jsou přiřazeny sezóny. Propojení se provádí pomocí výběru z nabídky, neboť jsou spojována již uložená data. Příklad spojování seriálů a sezón demonstruje vytvoření nového ročníku seriálu (viz. obr. 4.2 na následující straně). Analogicky lze takto vytvářet další metody, sloužící k propojování dvou entit.



Obrázek 4.2: Přidání nové sezóny seriálu

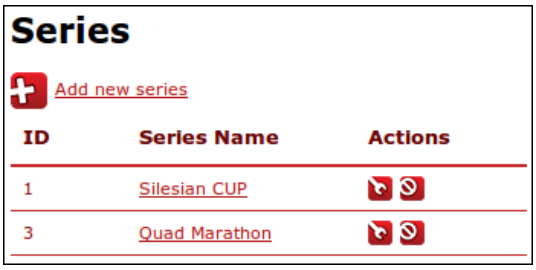
4.4 Zobrazování

Zobrazovaná data je možné vybírat pomocí dvou způsobů. Prvním způsobem lze vybrat data bez jakékoli další specifikace. Takto se například děje při výběru seriálů, kdy se zobrazují pokaždé stejná data. Druhý způsob výběru předpokládá znalost dodatečných informací, jimiž jsou data filtrována. Při výběru ročníků specifického seriálu je touto dodatečnou informací unikátní číselný identifikátor, který definuje seriál, jehož zobrazení je požadováno. V některých případech je nutno zadat dodatečných infomací více. Jako příklad poslouží výběr tříd, vypsaných v průběhu závodního víkendu. V takovémto případě je nutno zadat nejen identifikátor víkendu, ale také identifikátor sezóny seriálu.

Obrázek 4.3a znázorňuje vyobrazení všech vypsaných ročníků. V tomto případě se jedná o ročníky 2010 a 2011. Na obrázku 4.3b jsou zachyceny seriály Silesian CUP[9] a Quad Marathon[10].



(a) Zobrazení ročníků



(b) Zobrazení seriálů

Kapitola 5

Závěr

Vytvořil jsem návrh prototypu systému, který je schopen zpracovávat a archivovat data získaná v průběhu konání motoristických sportovních podniků a který by svými funkcemi napomáhal při organizaci a řízení průběhu závodů. Tato mnou vytvořená aplikace, demonstruje základní funkce zamýšleného systému. Jeho stávající podoba byla při zveřejnění bakalářské práce dostupná na <motokros.darkless.cz>. Částečné uplatnění tohoto prototypu se našlo při kontrole výsledků prvního závodu Silesian Cup 2011. Model tohoto univerzálního systému je možné dále rozšiřovat o celou řadu dalších funkcí, podporující například nahrávání a archivaci naskenovaných dokumentů v listinné podobě.

Mnou navrhovaný systém není v žádném případě postaven pouze do roviny teoretické a vychází z praktických požadavků, kladených na činnost časoměřičů při organizování motokrosových soutěží na zemské i republikové úrovni. Díky této bakalářské práci jsem si ještě dále prohloubil své znalosti, týkající se široké škály motoristických podniků.

Literatura

[1] *WTCC - Fia World Touring Car Championship* [online]. 2011 [cit. 2011-03-03]. Dostupné z WWW: <http://www.fiawtcc.com/Default.asp>.

[2] *Wikipedie: Otevřená encyklopedie: Pole position* [online]. c2011 [citováno 5. 03. 2011]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Pole_position&oldid=6607353>.

[3] *motogp.com* [online]. 2011 [cit. 2011-03-03]. Dostupné z WWW: <http://www.motogp.com>.

[4] *The Official SBK Website* [online]. 2011 [cit. 2011-03-03]. Dostupné z WWW: <http://www.worldsbk.com/en/home.html>.

[5] *Official website of the Dakar rallye raid* [online]. 2011 [cit. 2011-03-03]. Dostupné z WWW: <http://www.dakar.com/index_DAKus.html>.

[6] *PHP: Hypertext Preprocessor* [online]. 2001 [cit. 2011-05-05]. Dostupné z WWW: <http://www.php.net/>

[7] *CodeIgniter - Open source PHP web application framework* [online]. 2006 [cit. 2011-05-05]. Dostupné z WWW: <http://codeigniter.com/>.

[8] *CodeIgniter User Guide* [online]. 2006 [cit. 2011-05-05]. Change Log. Dostupné z WWW: <http://codeigniter.com/user_guide/changelog.html>.

[9] *SILESIAN CUP 2010 základní informace* [online]. 2010 [cit. 2011-05-07]. MSK MOTOR Velké Heraltice. Dostupné z WWW: <http://www.msk-motor.cz/node/4874>.

[10] *Výsledky 3. ročníku QUAD MARATHONU*[online]. 2010 [cit. 2011-05-07]. MSK MOTOR Velké Heraltice. Dostupné z WWW: <http://www.msk-motor.cz/node/4435>.

[11] *Wikipedie: Otevřená encyklopedie: Entity-relationship model* [online]. c2011 [citováno 12. 05. 2011]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Entity-relationship_model&oldid=6792331>.

[12] *Wikipedie: Otevřená encyklopedie: Diagram užití* [online]. c2010 [citováno 12. 05. 2011]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Diagram_u%C5%BEit%C3%AD&oldid=6242817>.

[13] *Wikipedie: Otevřená encyklopedie: Active Record* [online]. c2011 [citováno 13. 05. 2011]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Active_Record&oldid=6550839>.

[14] *IVRacing : pravidla czech endurance cup* [online]. 2009 [cit. 2011-05-14]. Dostupné z WWW: <http://www.ivracing.cz/czech-endurance-cup/pravidla/30>.

[15] Ráček, Jaroslav. *Strukturovaná analýza systémů.* Brno : Masarykova univerzita, 2006. 104 s. FI. ISBN 80-210-4190-0.

[16] Pressman, Roger S. *Software Engineering :A practitioner's approach.* 5th ed. Boston : McGraw-Hill, 2001. xxvii, 860. ISBN 0-07-365578-3.

[17] Sommerville, Ian. *Software engineering.* 5th ed. Wokingham : Addison-Wesley Publishing Company, 1996. xvi, 742 s. ISBN 0-201-42765-6.

[18] Arlow, Jim - Neustadt, Ila. *UML 2.0 and the unified process :practical object-oriented analysis and design.* 2nd ed. Boston : Addison-Wesley, 2005. xxiii, 592. ISBN 0321321278.

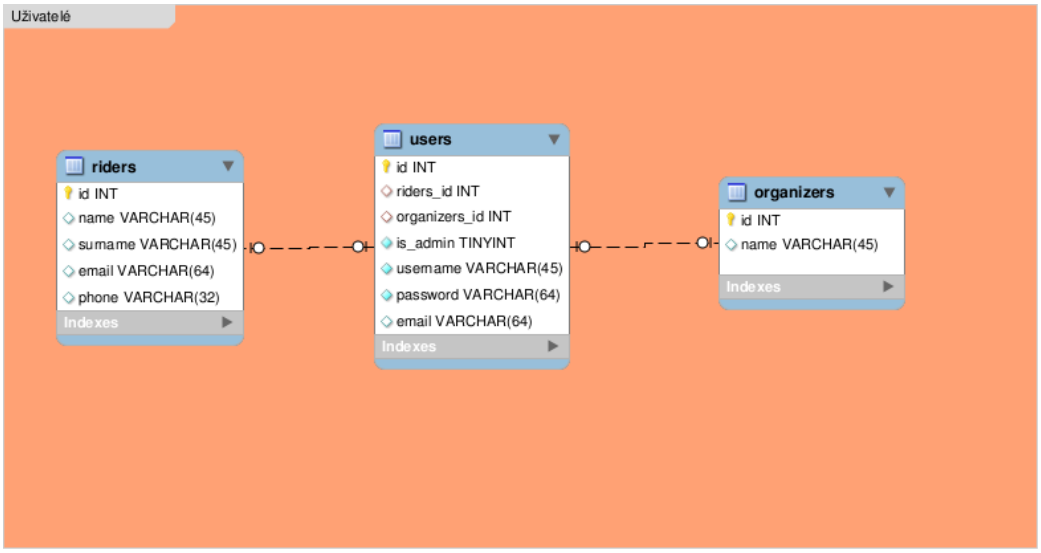
[19] Řády FIA : mezinárodní sportovní řády. In *MEZINÁRODNÍ SPORTOVNÍ ŘÁD : 2011* [online]. [s.l.] : [s.n.], 2011 [cit. 2011-05-14]. Dostupné z WWW: <http://www.autoklub.cz/show.php?page=acr/fasacr/radyfia/index.htm&asoc=2>.

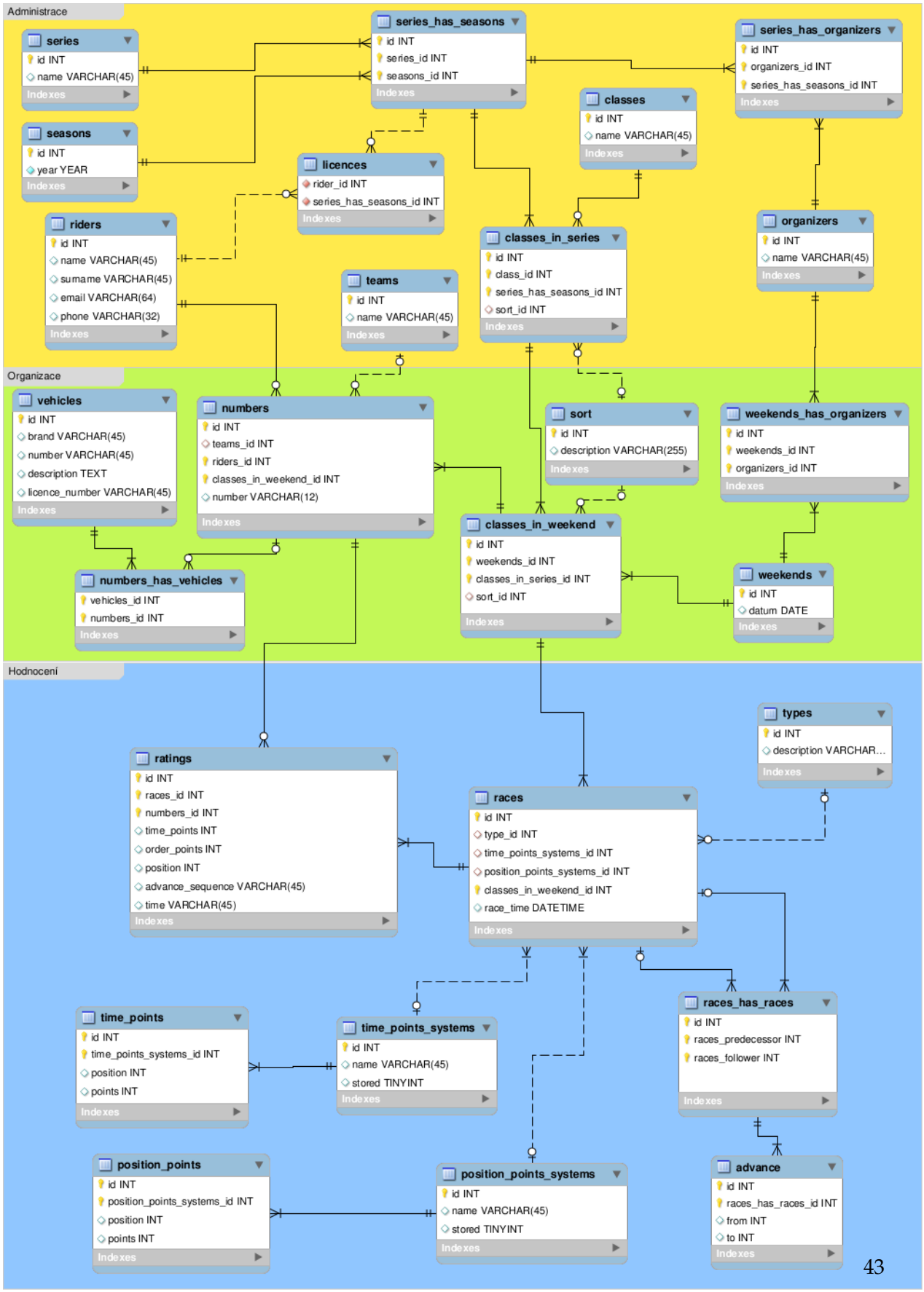
[20] Federace motocyklového sportu AČR. Ročenka 2011. In *Ročenka FMS AČR : 2011* [online]. [s.l.] : [s.n.], 2011 [cit. 2011-05-14]. Dostupné z WWW: <http://www.autoklub.cz/show.php?page=acr/fmsacr/fmsacr/rocenka/2011/index.htm&asoc=3>.

[21] Federace motocyklového sportu AČR. Všeobecný sportovní řád 2011. In *VŠEOBECNÝ SPORTOVNÍ ŘÁD : 2011* [online]. [s.l.] : [s.n.], 2011 [cit. 2011-05-14]. Dostupné z WWW: <http://www.autoklub.cz/show.php?page=acr/fmsacr/fmsacr/2011/rady/vseobecny/index.htm&asoc=3>.

Příloha A

Entitně-relační Diagram




Administrace
series
id INT
name VARCHAR(45)
Indexes
series_has_seasons
id INT
series_id INT
seasons_id INT
Indexes
series_has_organizers
id INT
organizers_id INT
series_has_seasons_id INT
Indexes
classes
id INT
name VARCHAR(45)
Indexes
seasons
id INT
year YEAR
Indexes
licences
rider_id INT
series_has_seasons_id INT
Indexes
riders
id INT
name VARCHAR(45)
surname VARCHAR(45)
email VARCHAR(64)
phone VARCHAR(32)
Indexes
classes_in_series
id INT
class_id INT
series_has_seasons_id INT
sort_id INT
Indexes
organizers
id INT
name VARCHAR(45)
Indexes
teams
id INT
name VARCHAR(45)
Indexes
Organizace
vehicles
id INT
brand VARCHAR(45)
number VARCHAR(45)
description TEXT
licence_number VARCHAR(45)
Indexes
numbers
id INT
teams_id INT
riders_id INT
classes_in_weekend_id INT
number VARCHAR(12)
Indexes
sort
id INT
description VARCHAR(255)
Indexes
weekends_has_organizers
id INT
weekends_id INT
organizers_id INT
Indexes
classes_in_weekend
id INT
weekends_id INT
classes_in_series_id INT
sort_id INT
Indexes
numbers_has_vehicles
vehicles_id INT
numbers_id INT
Indexes
weekends
id INT
datum DATE
Indexes
Hodnoceni
types
id INT
description VARCHAR...
Indexes
ratings
id INT
races_id INT
numbers_id INT
time_points INT
order_points INT
position INT
advance_sequence VARCHAR(45)
time VARCHAR(45)
Indexes
races
id INT
type_id INT
time_points_systems_id INT
position_points_systems_id INT
classes_in_weekend_id INT
race_time DATETIME
Indexes
time_points
id INT
time_points_systems_id INT
position INT
points INT
Indexes
time_points_systems
id INT
name VARCHAR(45)
stored TINYINT
Indexes
races_has_races
id INT
races_predecessor INT
races_follower INT
Indexes
position_points
id INT
position_points_systems_id INT
position INT
points INT
Indexes
position_points_systems
id INT
name VARCHAR(45)
stored TINYINT
Indexes
advance
id INT
races_has_races_id INT
from INT
to INT
Indexes

Příloha B

CD

Na přiloženém CD se nachází náslodující obsah.

- V kořenovém adresáři
  - Tato bakalářská práce ve formátu PDF a TEX
- V adresáři rozhraní
  - Všechna rozhraní výše popsaná v kapitole Implementace
- V adresáři prototyp
  - Prototyp vytvořeného systému
  - MySQL skript k vytvoření databázové struktury potřebné pro fungování prototypu.