DELTA – Střední škola informatiky a ekonomie s.r.o.
Ke Kamenci 151, PARDUBICE

IS – LPU Pardubice

Informační systém – OK LOKOMOTIVA PARDUBICE

Zeman Martin
4.A
Obor č. 18: Informatika
2021/22

Zadání maturitního projektu z informatických předmětů

Jméno a příjmení:	*Martin Zeman*
Školní rok:	*2021/2022*
Třída:	*3.A*
Obor:	*Informační technologie 18-20-M/01*
Téma práce:	*Modernizace rozhraní IS LPU*
Vedoucí práce:	*RNDr. Jan Koupil, Ph.D.*

Způsob zpracování, cíle práce, pokyny k obsahu a rozsahu práce:

Cílem práce je vytvořit nové uživatelské rozhraní stávajícího informačního systému oddílu orientačního běhu LPU Pardubice.

Webové uživatelské rozhraní bude navrženo jako responzivní a bude se připojovat ke stávající databázi MySQL.

V případě potřeby autor provede i modernizaci databáze a navrhne úpravy původního administračního rozhraní IS tak aby zůstalo funkční.

Stručný časový harmonogram (s daty a konkretizovanými úkoly):

- září: Analýza stávajícího řešení, hledání nekompatibility s aktuálními verzemi databáze
- říjen: migrace dat na aktuální verzi databáze
- listopad-prosinec: Implementace vlastního rozhraní
- leden-únor: Testování, opravy chyb, připojení původní administrace

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze zdroje a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Pardubicích dne 30.3.2022

_____

(vlastnoruční podpis)

Poděkování

Největší díky patří vedoucímu mého maturitního projektu RNDr. Janu Koupilovi, Ph.D. za odborné rady a nápady, které mi pomohly s tímto projektem.

A druhé díky patří mým spolužákům, kteří mě podrželi a povzbuzovali, i když projekt nešel zdaleka podle představ.

Resumé

Tato práce dokumentuje tvorbu nového uživatelského rozhraní pro členy pardubického klubu OK LOKOMOTIVA, ve kterém mají uživatelé přehled o novinkách v oddíle, plánovaných závodech na tuto sezónu a dalších akcích, mohou se na jednotlivé akce registrovat nebo se z nich naopak odhlásit. Součástí systému není administrace databáze, která zůstává od uživatelského rozhraní oddělena.

Systém je vytvořen s použitím frameworku Laravel a vybudován nad databází MySQL, jejíž restrukturalizace z původní verze je těžištěm této práce.

Klíčová slova

Webová aplikace, IS – LPU, Laravel, MySQL, PHP, Blade, Orientační běh, Maturitní práce

Resume

This thesis documents development of a new user interface for members of the club OK LOKOMOTIVA in Pardubice, in which users have an overview of news in the club, planned races for this season and other events. They can register for individual events or unsubscribe from them. The system does not include database administration, which remains separate from the user interface.

The system is created using the Laravel framework and built on top of the MySQL database. The restructuring of database from the original version is the focus of this work.

Keywords

Web application, IS – LPU, Laravel, MySQL, PHP, Blade, Orienteering, Graduation thesis

Obsah

1 Úvod

Projekt by měl nahradit stávající uživatelské rozhraní pardubického oddílu orientačního běhu OK LOKOMOTVA PARDUBICE. Staré rozhraní bylo neresponzivní, mělo zastaralý design a díky tomu nebylo uživatelsky přívětivé.

Tento projekt má za cíl tyto neduhy vymýtit, a vytvořit hezké a uživatelsky přívětivé rozhraní, ať už se na něj koukáte na počítači nebo telefonu.

Dále je cílem zmodernizovat a zefektivnit stávající databázi. Ta v současném stavu běží na zastaralé verzi MySQL a chybí v ní jakékoliv hlídání integrity databáze například v podobě cizích klíčů.

2 Použité technologie

2.1 Framework

Framework je software, který slouží jako základna, na které se staví následná konkrétní aplikace.

Každý framework je vytvořený k jinému účelu a má podle toho i předpřipravené funkce, které pomohou programátorovi zpřehlednit a zjednodušit jeho práci a kód samotný. Každý programátor však musí zvolit správný framework pro to, co po aplikaci požaduje, aby dělala.

2.1.1 Laravel

Laravel je open source framework pro webové aplikace pod licencí MIT a postavený na jazyce PHP. Je založený na architektuře MVC (model-view-controler). Což znamená, že kód je rozdělený do třech základních kategorií: Do modelu, který obsahuje data a funkce, dále do view, to slouží k prezentaci dat například v HTML, nebo v případě Laravelu Blade, a do controleru, který spravuje komunikace mezi modelem, view a uživatelem.

Laravel je designovaný hlavně pro komunikaci s MySQL databází, ale dá se použít i s dalšími druhy SQL databází, a dokonce i s NoSQL. Celkově je připravený na reálné použití a má připravenou většinu operací, které jsou společné pro většinu projektů, například registraci a přihlašování uživatelů. Má pro to připravené jak funkce, tak i tabulky, které vám vytvoří jedním příkazem hned po tom, co databázi připojíte k aplikaci. [3]

2.2 Šablonovací engine

Šablonovací engine je software, který se stará o prezenční vrstvu databáze, o takzvaný frontend. Kód v něm zapsaný je pak pro vyhledávače překládán do čistého HTML. Šablonovací engine nám dává možnost používat například podmínky nebo cykly, což výrazně zpřehledňuje výsledný kód.

2.2.1 Blade

Blade je jednoduchý, avšak mocný šablonovací engine, který je součástí Laravelu. Tento šablonovací engine má svojí hlavní výhodu v tom, že nám opravdu pouze zpřehledňuje kód, zatímco vše, co je v něm napsané, je uložené ještě v druhém PHP souboru, kde už je to opravdu pouze čisté PHP, které se přepisuje pouze v případě změny zdrojového kódu, tedy souboru Blade. Tento způsob zajišťuje co nejmenší zátěž výpočetního výkonu serveru, na kterém nám aplikace běží.

Soubory je třeba pojmenovávat s koncovkou .blade.php a v Laravelu jsou ukládány do adresáře /resources/views. [5]

2.3 ORM

Orm neboli Object Relational Mapping je vrstva mezi databází a samotným programem. Funkcí této vrstvy je získat data z databáze a namapovat je na objekty. To umožňuje jednoduší a přehlednější práci s databází. [2]

2.3.1 Eloquent

Eloquent je ORM rozhraní které používá Laravel. Ten nám zde zajišťuje veškerou komunikaci s databází, jak co se týče dotazů na data, tak i jejich importování případně upravování. A pomocí takzvaných migrací zde můžeme i vytvářet a upravovat tabulky. [4]

2.4 Databáze

Pokud začneme historií databází, tak nejstarší předchůdce databází jsou papírové kartotéky. Ty fungovaly na stejném principu jako fungují dnešní databáze. Data zde byla členěna podle jednotlivých kritérií a mohly se na sebe navzájem odkazovat, například pacient má tuto nemoc, informace o nemoci jsou uloženy v této části kartotéky v tomto svazku.

Časem se data přesunula z papíru na výpočetní techniku. Nejdříve to byly děrné štítky a následně, jak vývoj techniky pokračoval, se data přesouvala na výkonnější a efektivnější úložná média. Postupně vznikly hierarchické databáze, poté relační databáze, následně

objektové, avšak ty nedostály svým cílům, a místo aby vytlačily relační, tak se s nimi spojily, a tak vznikly dnešní objektově-relační databáze.

Mezi databázové objekty patří například pohledy, indexy, procedury a funkce nebo E-R schéma

2.4.1 MySQL

MySQL je databázový systém založený na relačním konceptu databáze s podporou objektů. Systém je multiplatformní. Komunikace s databází probíhá v jazyce SQL, který je však drobně modifikovaný. Není tedy možné vzít příkaz z této databáze a vložit ho například do PostgeSQL (jiný databázový systém založený také na jazyce SQL).

MySQL je v dnešní době hojně využívaná pro drobné projekty, a to hlavně díky jednoduchosti implementace. Druhým zásadním důvodem je dostupnost. MySQL je kompletně zdarma. Velice často je využívána v kombinaci s backendem, který je napsaný v jazyce PHP.

MySQL je jednou z jednodušších SQL databází, ale to má vliv na výkon. Databáze je designovaná hlavně na rychlost. [6]

2.4.2 Normalizace databáze

Normalizace databáze je postup, při kterém přeorganizujeme, roztřídíme a rozdělíme data tak, abychom co nejefektivněji využívali potenciál relačních databází.

Při normalizaci databáze řešíme konzistenci (úplnost) a redundanci (opakování) dat.

Při normalizaci databáze se postupuje podle normálových forem. Normálové formy jsou pravidla, podle kterých upravujeme strukturu databáze. Při normalizaci musíme splňovat normálové formy postupně jelikož pro splnění každé normálové formy je vyžadováno splnění všech předchozích forem.

Ve většině projektů se požaduje splnění 3NF a o další se nesnažíme. Je to způsobeno tím, že výsledek ve výkonu a udržitelnosti databáze neodpovídá vynaloženému úsilí, které je třeba, aby databáze tyto normálové formy splňovala.

2.4.2.1 Nenormalizovaná forma (UNF)

Nenormalizovaná forma byla zavedena pro databáze nesplňující normálové formy.

Pro její splnění stačí, že databáze existuje.

2.4.2.2 Nultá normální forma (0NF)

V nulté normálové formě se tabulka vyskytne tehdy, jsou-li nějaká data uložená v jednom sloupečku tabulky dále dělitelná. Což je však přímo v rozporu s relačním modelem, který nám říká, že data v jednom poli jsou dále nedělitelná.

Tabulka splňuje buďto nultou, nebo první normálovou formu, jelikož tyto dvě si navzájem odporují.

2.4.2.3 První normální forma (1NF)

První normálová forma nám naopak říká, že každé pole tabulky obsahuje atomické, dále nedělitelné, hodnoty.

2.4.2.4 Druhá normální forma (2NF)

Druhá normálová forma nám říká, že každý neklíčový atribut je plně závislý na každém kandidátním klíči.

Jinak řečeno každý atribut, který není kandidátním klíčem tabulky, musí souviset se všemi kandidátními klíči tabulky. Pokud nesouvisí tak by měl být umístěn do jiné tabulky, kde toto pravidlo splňovat bude.

2.4.2.5 Třetí normální forma (3NF)

Třetí normálová forma nám říká, že všechny neklíčové atributy musí být vzájemně nezávislé.

Tedy pokud odstraníme kterýkoliv sloupeček neklíčových hodnot tabulky, neporuší se nám tím integrita dat uložených v této tabulce.

2.4.2.6 Boyce-Coddova normální forma (BCNF)

Boyce-Coddova normální forma nám říká že, atributy, které jsou součásti primárního klíče, musí být vzájemně nezávislé.

Tedy žádný sloupeček, který je primárním klíčem, nebo jeho součástí, nesmí být závislí na jakémkoli jiném sloupečku tabulky.

2.4.2.7 Čtvrtá normální forma (4NF)

Čtvrtá normálová forma nám říká, že relace popisuje pouze příčinnou souvislost mezi klíčem a atributy.

To znamená, že v jedné tabulce máme uloženy pouze atributy, které spolu souvisí. Každý atribut tedy musí souviset se všemi ostatními atributy tabulky. Nestačí, že všechny atributy souvisí s primárním klíčem.

2.4.2.8 Pátá normální forma (5NF)

Pátá normálová forma nám říká, že relace již není možno bezeztrátově rozložit. [1]

3 Praktická část

3.1 Databáze

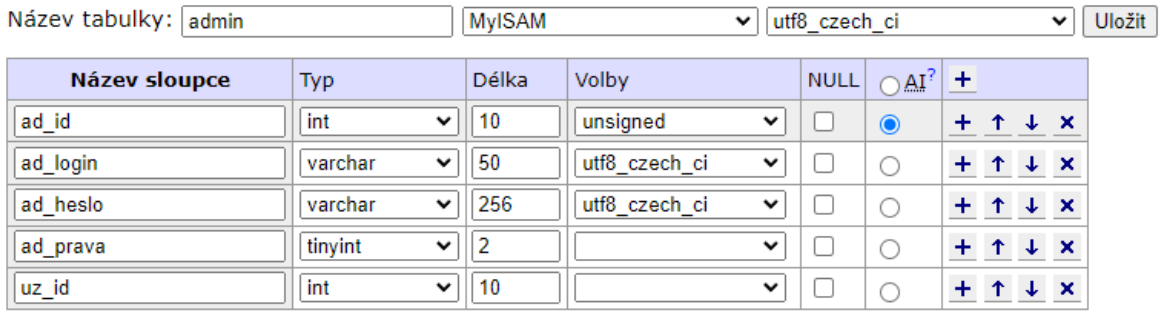
Jak jsem již psal v části technologií, pro tento projekt jsem použil SQL databázi, konkrétně se jedná o databázi MySQL. Výběr nebyl složitý, jelikož v MySQL byla vytvořena i původní databáze, kterou jsem dostal jako jeden z podkladů k tomuto projektu. Naštěstí MySQL je databáze, která naprosto vyhovuje veškerým potřebám tohoto projektu. Je jednoduchá a rychlá, což je vše, co na tento projekt potřebujeme. Navíc není problém tuto databázi spustit zdarma na jakémkoliv stroji.

jako velký problém se ukázal fakt, že tato práce navazuje na existující zastaralé řešení, jehož součástí byla již používaná databáze, kterou si zadavatelé přáli zachovat. Většina tabulek byla nevyhovující, a to ať se bavíme o použité kolekci znaků, nebo například o vazbách mezi jednotlivými tabulkami.

Databáze také běžela na několik let staré verzi, a měla implementované funkce, které již nové verze MySQL nepodporují.

Díky tomuto všemu musel být první krok projektu úprava stávající databáze.

Obrázek 1 Detail tabulky admin



Na obrázku výše vidíte původní tabulku admin. Na obrázku je vidět hned několik chyb, které se objevují ve skoro každé tabulce databáze, některá dokonce v každé.

V prvním kroku musela být změněna znaková kolekce z utf8_czech_ci na utf8mb4_czech_ci a to jak pro celou tabulku, tak i pro všechny sloupce typu varchar. Tato změna byla nutná, aby v databázi nedělaly problém čtyřbitové znaky jako mohou být například emotikony.

Zároveň s tím se také měnil formát ukládání dat v tabulkách z formátu MyISAM na formát InnoDB. To je pro chod databáze zásadní změna. Formát InnoDB je sice náročnější na výkon databáze, ale podporuje cizí klíče, což je základ pro udržení konzistence dat v SQL databázích. V původní databázi nebyly cizí klíče vůbec implementovány.

Po tomto kroku tedy bylo zapotřebí projít celou databázi a zjistit všechny vazby mezi tabulkami. Po této analýze musely být všechny tyto vazby definovány pomocí cizích klíčů.

V první příloze vidíte všechny nadefinované relace mezi tabulkami. Pravidla pro tyto cizí klíče byla po konzultaci s klientem nastaveny následovně: Při změně dat, která by zapříčinila nekonzistenci dat, databáze tuto změnu zablokuje a neprovede se a při mazání dat budou kaskádovitě smazána i všechna závislá data. Díky tomu se nám nestane, že bychom někde měli cizí klíč, který by odkazoval na neexistující řádek tabulky.

Dalším krokem bylo upravit jednotlivé tabulky tak, aby splňovaly alespoň třetí normální formu. U tohoto kroku jsem však narazil na neshody s klientem, kde podle normálních forem a obecně známých pravidel měla tabulka vypadat jedním stylem, ale klient si nepřál tak rozsáhlé změny v databázi a tím pádem jsme museli najít nějaký kompromis.

3.1.1 Tabulka admin

Obrázek 2 Tabulka admin

Sloupec	Typ	Komentář
ad_id	bigint(20) unsigned *Auto Increment*	
ad_prava	tinyint(4)	
uz_id	bigint(20) unsigned	

Z této tabulky byly odebrány dva sloupce, těmi jsou ad_login a ad_heslo, ve kterých byla uložena stejná data jako v tabulce uživatelé, na kterou je každý záznam z této tabulky pomocí cizího klíče navázán.

Také byl přidán samotný cizí klíč na sloupeček uz_id, který odkazuje, jak už bylo zmíněno na sloupeček uz_id z tabulky uzivatelé.

3.1.2 Tabulka akce

Obrázek 3 Tabulka akce

Sloupec	Typ	Komentář
ak_id	bigint(20) unsigned *Auto Increment*	
ak_status	tinyint(4) [**0**]	
ak_typ	smallint(6) [**0**]	
ak_nazev	varchar(100)	
ak_misto	varchar(200)	
ak_popis	text	
ak_detail	text	
ak_zacatek_datum	date	
ak_zacatek_cas	time [**00:00:00**]	
ak_konec_datum	date	
ak_konec_cas	time [**00:00:00**]	
ak_externi_odkazy	varchar(200)	
ak_komentare	tinyint(4) [**0**]	
ak_typ_prihlasek	smallint(6) [**0**]	
ak_porada	varchar(100)	
ak_spravce	bigint(20) unsigned	

V této tabulce nebyly potřeba až tak velké změny. Na sloupeček ak_spravce byl nadefinován cizí klíč odkazující na sloupeček uz_id z tabulky uživatelé. A ze sloupců s typem date byla odebrána vlastnost nullable, protože akce musí mít datum začátku i konce.

3.1.3 Tabulka akce_komentar

Obrázek 4 Tabulka akce_komentar

Sloupec	Typ	Komentář
ako_id	bigint(20) unsigned *Auto Increment*	
uz_id	bigint(20) unsigned	
ak_id	bigint(20) unsigned	
ako_datumcas	datetime [**CURRENT_TIMESTAMP**]	
ako_text	text	

V této tabulce byla přidána defaultní hodnota na sloupeček ako_datumcas. A dále byly přidány cizí klíče na sloupečky uz_id a ak_id. Uz_id odkazuje na tabulku uživatelé a ak_id odkazuje na tabulku akce.

3.1.4 Tabulka akce_prihlaska

Obrázek 5 Tabulka akce_prihlaska

Sloupec	Typ	Komentář
pr_id	bigint(20) unsigned *Auto Increment*	
uz_id	bigint(20) unsigned	
ak_id	bigint(20) unsigned	
pr_kategorie	varchar(255)	
pr_poznamka	varchar(255)	
pr_datum_prihlaseni	datetime [**CURRENT_TIMESTAMP**]	
pr_ucast	tinyint(4) [**1**]	

Do této tabulky musel být přidán sloupeček pr_id, jelikož tabulka původně neměla žádný primární klíč a nebyl zde nadefinovaný ani složený klíč. Dále se zde přidala defaultní hodnota na sloupeček pr_datum_prihlaseni a cizí klíče na sloupečky uz_id a ak_id, které jsou stejné jako v předchozí tabulce.

3.1.5 Tabulka akce_prihlasky_def

Obrázek 6 Tabulka akce_prihlasky_def

Sloupec	Typ	Komentář
apd_id	bigint(20) unsigned *Auto Increment*	
ak_id	bigint(20) unsigned	
apd_konec_prihlasek	date	
apd_kategorie	varchar(255)	

V této tabulce byl přejmenován sloupeček apd_id z původního zavádějícího názvu a byl na něj přidán index primárního klíče. Dále byl přidán sloupeček ak_id který se nastavil jako cizí klíč odkazující na tabulku akce.

3.1.6 Tabulka aktuality

Obrázek 7 Tabulka aktuality

Sloupec	Typ	Komentář
ak_id	bigint(20) unsigned *Auto Increment*	
ak_nadpis	varchar(255) [**0**]	
ak_znak	tinyint(4) [**0**]	
ak_text	text	

Na této tabulce nebyly prováděny žádné další úpravy, krom již výše zmíněných.

3.1.7 Tabulka bazar

Obrázek 8 Tabulka bazar

Sloupec	Typ	Komentář
bz_id	bigint(20) unsigned *Auto Increment*	
uz_id	bigint(20) unsigned	
bz_bzdopr	tinyint(4) [**0**]	
bz_doprdatum	varchar(32) *NULL*	
bz_doprpocet	varchar(32) *NULL*	
bz_typ	tinyint(4) [**0**]	
bz_rodic	int(11) [**0**]	
bz_datum	datetime [**CURRENT_TIMESTAMP**]	
bz_nadpis	varchar(255) *NULL*	
bz_popis	text *NULL*	
bz_odkaz	varchar(255) *NULL*	
bz_status	tinyint(4) [**0**]	

V této tabulce byla na sloupeček bz_datum přidána defaultní hodnota a na sloupeček uz_id byl přidán cizí klíč odkazující na tabulku uživatelé.

3.1.8 Tabulka bzsoubor

Obrázek 9 Tabulka bzsoubor

Sloupec	Typ	Komentář
bzs_id	bigint(20) unsigned *Auto Increment*	
bz_id	bigint(20) unsigned	
bzs_poradi	tinyint(4) [**0**]	
bzs_cesta	varchar(255)	
bzs_popis	varchar(128)	
bzs_status	tinyint(4) [**0**]	

V této tabulce bylo potřeba pouze přidat cizí klíč na sloupeček bz_id odkazující na tabulku bazar.

3.1.9 Tabulka chattr

Obrázek 10 Tabulka chattr

Sloupec	Typ	Komentář
tr_id	bigint(20) unsigned *Auto Increment*	
tr_datum	datetime [**CURRENT_TIMESTAMP**]	
tr_odkaz	varchar(200)	
tr_podpis	varchar(35)	
tr_text	text	
tr_autor	bigint(20) unsigned	

V této tabulce se odebrala vlastnost nullable ze sloupečků tr_datum a tr_odkaz a na sloupeček tr_datum byla zároveň přidána defaultní hodnota. Dále na sloupeček tr_autor byl přidán cizí klíč odkazující na tabulku uživatelé.

3.1.10 Tabulka diskuze

Obrázek 11 Tabulka diskuze

Sloupec	Typ	Komentář
di_id	bigint(20) unsigned *Auto Increment*	
di_datum	datetime [**CURRENT_TIMESTAMP**]	
di_nazev	varchar(60)	
di_podpis	varchar(30)	
di_text	text	
di_autor	bigint(20) unsigned	

V táto tabulce byla opět vlastnost nullable na sloupečku di_datum nahrazena defaultní hodnotou. A na sloupec di_autor byl přidán cizí klíč.

3.1.11 Tabulka login_check

Obrázek 12 Tabulka login_check

Sloupec	Typ	Komentář
lc_id	bigint(20) unsigned *Auto Increment*	
lc_uzid	bigint(20) unsigned	
lc_string	varchar(128)	
lc_type	tinyint(4)	

V této tabulce byl pouze přidán cizí klíč na sloupeček lc_uzid odkazující na tabulku uživatelé.

3.1.12 Tabulka obleceni

Obrázek 13 Tabulka obleceni

Sloupec	Typ	Komentář
ako_id	bigint(20) unsigned *Auto Increment*	
uz_id	bigint(20) unsigned	
ob_typ	varchar(5)	
ob_velikost	varchar(16)	
ob_pocet	tinyint(4) [**0**]	
ob_poznamka	varchar(256)	

V této tabulce byl pouze přidán cizí klíč na sloupeček uz_id odkazující na tabulku uživatelé.

3.1.13 Tabulka poradani

Obrázek 14 Tabulka poradani

Sloupec	Typ	Komentář
po_id	bigint(20) unsigned *Auto Increment*	
uz_id	bigint(20) unsigned	
po_znak	varchar(20)	
po_poznamka	varchar(255)	

Z této tabulky byla odebrána defaultní hodnota ze sloupečku uz_id, která vkládala 0, a zároveň na něj byl přidán cizí klíč odkazující na tabulku uživatelé.

3.1.14 Tabulka prihlasky

Obrázek 15 Tabulka prihlasky

Sloupec	Typ	Komentář
pr_id	bigint(20) unsigned *Auto Increment*	
uz_id	bigint(20) unsigned	
za_id	bigint(20) unsigned	
pr_kategorie	varchar(10)	
pr_naklady	double(6,2) [**0.00**]	
pr_doprava	tinyint(4) [**0**]	
pr_ubytovani	tinyint(4) [**0**]	
pr_prijmeni	varchar(33)	
pr_poznamka	text *NULL*	
pr_ucast	tinyint(4) [**0**]	
pr_datum_prihlaseni	datetime [**CURRENT_TIMESTAMP**]	
pr_datum_admin	datetime *NULL*	
pr_datum	date	

V této tabulce bylo potřeba přidat sloupeček pr_datum_admin, jelikož admin původně, když upravil tabulku, přepsal sloupeček pr_datum_prihlaseni, čímž jsme přišli o data kdy se na závod přihlásil uživatel.

Dále zde byly přidány cizí klíče na sloupečky uz_id a za_id odkazující na tabulky uživatelé a závody.Přidána byla také zde defaultní hodnota na sloupeček pr_datum_prihlaseni a byly odebrány defaultní hodnoty ze sloupců uz_id a za_id.A byl doplněn sloupec pr_id, který slouží jako primární klíč.

Sloupeček pr_datum by mohl být odstraněn, protože je v něm stejná informace jako v tabulce závody, ale klient ho si jej přál ponechat nechat kvůli řazení.

Do sloupečku pr_kategorie by se nově mělo správně ukládat id kategorie a nastavit cizí klíč, ale klientovi se toto řešení zdálo nepřehledné, proto sbylo zanecháno ukládání varcharové hodnoty kategorie.

3.1.15 Tabulka prihlaskyvic

Obrázek 16 Tabulka prihlaskyvic

Sloupec	Typ	Komentář
pr_id	bigint(20) unsigned *Auto Increment*	
uz_id	bigint(20) unsigned	
za_id	bigint(20) unsigned	
pr_kategorie	varchar(10)	
pr_ubytovani	tinyint(4) [**-1**]	
pr_doprava	tinyint(4) [**-1**]	
pr_program1	tinyint(4) [**-1**]	
pr_program2	tinyint(4) [**-1**]	
pr_poznamka	text *NULL*	
pr_datum	date	
pr_prijmeni	varchar(33)	
pr_datumprihl	datetime [**CURRENT_TIMESTAMP**]	
pr_terminprihl	tinyint(4) [**0**]	
pr_potvrzeni	tinyint(4) [**0**]	
pr_vzkaz	text *NULL*	
pr_vklad	double(8,2)	
pr_datum_admin	datetime *NULL*	

Do této tabulky byl přidán sloupeček pr_id a nastaven jako primární identifikátor.

Na sloupeček pr_datumprihl byla přidána defaultní hodnota a odstraněna původní vlastnost nullable.

Sloupeček pr_kategorie na tom je problematicky stejně jak to je již vysvětleno výše u jednodenních přihlášek.

3.1.16 Tabulka soutez_odpovedi

Obrázek 17 Tabulka soutez_odpovedi

Sloupec	Typ	Komentář
sod_id	bigint(20) unsigned *Auto Increment*	
sot_id	bigint(20) unsigned	
uz_id	bigint(20) unsigned	
sod_odp	varchar(3)	
sod_zisk	tinyint(4) [**0**]	
sod_datum	datetime [**CURRENT_TIMESTAMP**]	
sod_ipadr	varchar(50)	
sod_host	varchar(50)	
sod_tema	tinyint(4) [**0**]	

Do této tabulky byl přidán sloupeček sod_id a byl nastaven jako primární klíč.

Dále sloupečku sod_datum byla odebrána vlastnost nullable a byla přidána defaultní hodnota.

A na sloupečky sot_id a uz_id byly přidány cizí klíče odkazující na tabulky uživatelé a soutez_otazky.

3.1.17 Tabulka soutez_otazky

Obrázek 18 Tabulka soutez_otazky

Sloupec	Typ	Komentář
sot_id	bigint(20) unsigned *Auto Increment*	
sot_start	date	
sot_end	date	
sot_textot	text	
sot_odpa	text	
sot_odpb	text	
sot_odpc	text	
sot_odpd	text	
sot_odpe	text	
sot_odpf	text	
sot_odpg	text	
sot_odph	text	
sot_sprodp	varchar(3)	
sot_autor	bigint(20) unsigned	
sot_status	tinyint(4) [**0**]	
sot_tema	tinyint(4) [**0**]	
sot_typ	tinyint(4) [**0**]	

V této tabulce byla ze sloupců sot_start a sot_end odstraněna vlastnost nullable. A na sloupeček uz_id byl přidán cizí klíč odkazující na tabulku uživatelé.

3.1.18 Tabulka termin

Obrázek 19 Tabulka termin

Sloupec	Typ	Komentář
te_id	bigint(20) unsigned *Auto Increment*	
te_datum	date	

Tato tabulka slouží pouze k uložení pouze jedné hodnoty. Celá tabulka by měla být předělána na tabulku konstant, protože toto je jediná konstanta v databázi a ostatní konstanty jsou uloženy v textovém souboru v aplikaci, ale klient si takové změny nepřeje, jelikož by musel měnit administrátorské rozhraní.

3.1.19 Tabulka tmakce

Obrázek 20 Tabulka tmakce

Sloupec	Typ	Komentář
tm_id	bigint(20) unsigned *Auto Increment*	
uz_id	bigint(20) unsigned	
tm_datum	datetime *NULL*	
tm_nazev	varchar(255)	
tm_popis	text	
tm_status	tinyint(4) [**0**]	

Do této tabulky byl pouze přidán cizí klíč na sloupeček uz_id odkazující na tabulku uživatelé a byla z něho odebrána defaultní hodnota 0.

3.1.20 Tabulka tmsoubor

Obrázek 21 Tabulka tmsoubor

Sloupec	Typ	Komentář
tms_id	bigint(20) unsigned *Auto Increment*	
tm_id	bigint(20) unsigned	
tms_poradi	tinyint(4) [**0**]	
tms_cesta	varchar(255)	
tms_popis	varchar(128)	
tms_status	tinyint(4) [**0**]	

Do této tabulky byl pouze přidán cizí klíč na sloupeček tm_id odkazující na tabulku tmakce a byla z něho odebrána defaultní hodnota 0.

3.1.21 Tabulka uzivatele

Obrázek 22 Tabulka uzivatele

Sloupec	Typ	Komentář
uz_id	bigint(20) unsigned *Auto Increment*	
uz_jmeno	varchar(24)	
uz_prijmeni	varchar(33)	
uz_login	varchar(10)	
uz_heslo	varchar(256)	
uz_reg_cislo	varchar(10)	
uz_licence	char(3) [**C**]	
uz_cip	int(11) [**0**]	
uz_mobil	varchar(33) *NULL*	
uz_email	varchar(50) *NULL*	
uz_icq	varchar(20) *NULL*	
uz_vklady	double(8,2) [**0.00**]	
uz_vydaje	double(8,2) [**0.00**]	
uz_placprisp	tinyint(4) [**0**]	
uz_aktivita	tinyint(4) [**0**]	
uz_prisp1	double(2,1) [**0.5**]	
uz_prisp2	double(2,1) [**0.5**]	
uz_maillist	tinyint(4) [**0**]	
uz_naroz	varchar(10) [**0**]	
uz_registrace	tinyint(4) [**1**]	
uz_rodcislo	varchar(16)	
uz_zeme	varchar(8) [**CZ**]	
uz_zust_loni	double(8,2) [**0.00**]	
uz_todelete	tinyint(4) [**0**]	
uz_prihl_ost	varchar(100)	
uz_mailtrenink	tinyint(4) [**0**]	

Na této tabulce nebyly provedeny žádné změny.

3.1.22 Tabulka vklady

Obrázek 23 Tabulka vklady

Sloupec	Typ	Komentář
vk_id	bigint(20) unsigned *Auto Increment*	
uz_id	bigint(20) unsigned	
vk_status	tinyint(4) [**0**]	
vk_castka	double(8,2) *NULL*	
vk_termin	date	
vk_poznamka	varchar(100) *NULL*	

V této tabulce byla ze sloupečku vk_termin odebrána vlastnost nullable. A na sloupeček uz_id byl přidán cizí klíč odkazující na tabulku uživatelé.

3.1.23 Tabulka zavody

Obrázek 24 Tabulka zavody

Sloupec	Typ	Komentář
za_id	bigint(20) unsigned *Auto Increment*	
za_status	tinyint(4) [**0**]	
za_oris	varchar(64) *NULL*	
za_typ	tinyint(4) [**0**]	
za_nazev	varchar(100)	
za_misto	varchar(200)	
za_termin	date	
za_termin_cas	time [**00:00:00**]	
za_konec_datum	date	
za_konec_cas	time [**00:00:00**]	
za_konec_prihl	date	
za_obdobi	tinyint(4) [**0**]	
za_oddil	varchar(10)	
za_doprava	tinyint(4) [**0**]	
za_cena_dopravy	double(6,2) [**0.00**]	
za_ubytovani	tinyint(4) [**0**]	
za_cena_ubytovani	double(6,2) [**0.00**]	
za_web	varchar(100) *NULL*	
za_poznamka	text *NULL*	
za_ucast	tinyint(4) [**0**]	
za_odjcas	varchar(10) *NULL*	
za_odjmisto	tinyint(4) [**0**]	
za_vysledky	varchar(128) [**0**]	
za_zebr0	varchar(32) [**0**]	
za_zebr1	varchar(32) [**0**]	
za_zebr2	varchar(32) [**0**]	

Z této tabulky bylo odebráno několik sloupečků, které deklarovaly kategorie a vklady pro dané kategorie, a tyto sloupečky byli nahrazeny novou tabulkou zavody_kategorie. Původní řešení znamenalo osm sloupečků pro kategorie a osm pro vklady k těmto kategoriím, ale problém byl, že na každém závodě byl jiný počet kategorií, a tím pádem nebyly většinou všechny tyto sloupečky využity. Proto byly nahrazeny novou tabulkou kde je nyní jedno kolik kategorií který závod má, jelikož jsou na sobě v tomto ohledu závody naprosto nezávislé.

Dále zde byla z datumů odebrána vlastnost nullable, protože každý závod musí mít začátek a konec.

3.1.24 Tabulka zavody_kategorie

Obrázek 25Tabulka zavody_kategorie

Sloupec	Typ	Komentář
za_kat_id	bigint(20) unsigned *Auto Increment*	
za_id	bigint(20) unsigned	
kategorie	text	
vklad	double(6,2) [**0.00**]	

Toto je nová tabulka vzniklá z dříve popsaných důvodů.

3.1.25 Tabulka zavvic

Obrázek 26 Tabulka zavvic

Sloupec	Typ	Komentář
za_id	bigint(20) unsigned *Auto Increment*	
za_status	tinyint(4) [**0**]	
za_oris	varchar(64) *NULL*	
za_nazev	varchar(100)	
za_misto	varchar(200)	
za_termintext	varchar(25)	
za_terminzac	date	
za_terminzac_cas	time [**00:00:00**]	
za_konec_datum	date	
za_konec_cas	time [**00:00:00**]	
za_oddil	varchar(10)	
za_ubytpopis	varchar(512)	
za_ubyt	varchar(255)	
za_ubytvklad	varchar(255)	
za_dopravapopis	varchar(512)	
za_doprava	varchar(255)	
za_dopravavklad	varchar(255)	
za_program1popis	varchar(512)	
za_program1	varchar(255)	
za_program1vklad	varchar(255)	
za_program2popis	varchar(512)	
za_program2	varchar(255)	
za_program2vklad	varchar(255)	
za_web	varchar(100)	
za_poznamka	text	
za_platba	text	
za_terminprihl	tinyint(3) unsigned [**0**]	
za_vedouci	int(11) [**0**]	
za_vysledky	varchar(128) [**0**]	
za_zebricek	varchar(32) [**0**]	
za_mail_posledni	datetime	
za_typ_kateg	tinyint(4) [**1**]	
za_zkratka	varchar(8)	

Z této tabulky muselo být také odstraněno několik sloupců, a to ze stejného důvodu jako u tabulky závody. Tyto sloupce byly nahrazeny dvěma tabulkami, a to termin_zavvic a zavvic_termin_prihlasek. V tabulce termin_zavvic jsou uloženy datumy a tagy jednotlivých

datumů, které nám určují kolikáté kolo přihlášek je tento termín. To by šlo sice zjišťovat i automaticky, ale zatěžovalo by to výpočetní výkon a klient si také chce mezi těmito koly sám ručně přepínat. To, v kolikátém kole právě přihlášky jsou, se určuje podle sloupečku za_typ_kateg, který si klient chce sám upravovat. Z tohoto sloupečku poté vychází druhá tabulka, ve které jsou deklarovány kategorie, na které se může uživatel přihlásit, a ceny jednotlivých kategorií podle toho ve kterém kole termínů se přihlásíte. U těchto závodů to funguje tak, že čím dříve se přihlásíte, tím méně vás to bude stát.

3.1.26 Tabulka termin_zavvic

Obrázek 27 Tabulka termin_zevvic

Sloupec	Typ	Komentář
ter_id	bigint(20) unsigned *Auto Increment*	
za_id	bigint(20) unsigned	
ter_tag	tinyint(3) unsigned	
ter_datum	date	

Toto je nová tabulka vzniklá z dříve popsaných důvodů.

3.1.27 Tabulka zavvic_termin_prihlasek

Obrázek 28 Tabulka zavvic_termin_prihlasek

Sloupec	Typ	Komentář
za_kat_id	bigint(20) unsigned *Auto Increment*	
za_id	bigint(20) unsigned	
za_terminprihl	tinyint(3) unsigned	
vklad	text	
za_kateg	text	

Toto je nová tabulka vzniklá z dříve popsaných důvodů.

3.1.28 Tabulka zebprihl

Obrázek 29 Tabulka zebprihl

Sloupec	Typ	Komentář
zp_id	bigint(20) unsigned *Auto Increment*	
uz_id	bigint(20) unsigned	
ze_id	bigint(20) unsigned	
zp_kategorie	varchar(6)	

Do této tabulky byl přidán sloupeček zp_id a byl nastaven jako primární klíč.

Ze sloupečků uz_id a ze_id byla odebrána vlastnost nullable, a naopak byl na tyto sloupečky přidán cizí klíč odkazující na tabulky uživatelé a žebříček.

Sloupeček zp_prijmeni byl odebrán, protože přímení je uloženo právě v tabulce uživatelé.

3.1.29 Tabulka zebricek

Obrázek 30 Tabulka zebricek

Sloupec	Typ	Komentář
ze_id	bigint(20) unsigned *Auto Increment*	
ze_nazev	varchar(20)	
ze_typ	tinyint(4) [**0**]	
ze_datum	date	
ze_status	tinyint(4) [**0**]	

V této tabulce byla pouze odebrána ze sloupečku datum možnost nullable.

3.2 Webová aplikace

Jak je zmíněno výše, webová aplikace je postavená na PHP frameworku Laravel, který se skládá z několika částí. Pro práci s uživateli byl použit middleware od Laravelu, který byl upraven pro potřeby této aplikace. Dále se je zde router, který převezme informace ze zadané adresy a předá je příslušné funkci v kontroleru. Kontroler je následně zpracuje, provede požadované akce, které v sobě má deklarovány, a předá výsledek view - zobrazovací vrstvě. Ta výsledek zobrazí uživateli, dokud ten neprovede další akci a celý proces se znovu nezopakuje.

3.2.1 Migrace

Migrace jsou soubory, které používá Laravel pro práci se strukturou databáze. Jsou uloženy ve složce *database/migrations* a píší se sem veškeré změny struktury databáze. I v tomto projektu máme každou tabulku definovanou ve vlastní migraci a poté je zde jedna velká migrace přidávající cizí klíče. To nám dohromady zajišťuje velice jednoduchou implementaci projektu, kdy nám pro inicializaci stačí v souboru .env přepsat potřebné informace pro připojení databáze a následně v příkazovém řádku projektu zadáme příkaz *php artisan migrate* a tím máme rozchozenou celou databázi.

3.2.2 Modely

Modely v Laravelu slouží taktéž pro práci s databází, ale tentokrát se jedná o práci s daty. Každá tabulka má k sobě vytvořený odpovídající model, ve kterém je řečeno, jaké sloupečky daná tabulka má, co je primární klíč, do jakých sloupců můžeme zapisovat napřímo a do jaký pouze s pomocí metod které pro to připravíme a tak dále. Jsou zde také definovány relace mezi tabulkami. Tyto modely pak slouží Laravelu pro převod databázových záznamů na objekty, s nimiž je dále jednodušší práce.

3.2.3 Middleware

Pro potřeby této aplikace bylo potřeba upravit originální middleware, který je nastaven na Laravelem předdefinovanou tabulku. To pro naše potřeby nebylo vhodné, jelikož jsme dostali tabulku uživatelů od klienta. Proto jsme museli v souboru *config/auth.php* upravit poskytovatele, jak je vidět na obrázku

Obrázek 31 Úprava poskytovatelů

```
'providers' => [
    'users' => [
        'driver' => 'eloquent',
        'model' => App\Models\User::class,
    ],

    'uzivatele' => [
        'driver' => 'eloquent',
        'model' => App\Models\Uzivatel::class,
    ],
```

vpravo, a v tomtéž souboru jsme tohoto poskytovatele museli poskytnout funkci guards namísto starého poskytovatele.

3.2.4 Router

Obrázek 32 Web router

```
Route::get( uri: '/login', [CustomAuthController::class, 'index'])->name( name: 'login');
Route::post( uri: '/custom-login', [CustomAuthController::class, 'customLogin'])->name( name: 'login.c
//Route::get('/registration', [CustomAuthController::class, 'registration'])->name('register-user')
//Route::post('/custom-registration', [CustomAuthController::class, 'customRegistration'])->name('r
Route::get( uri: '/signout', [CustomAuthController::class, 'signOut'])->name( name: 'signout');
Route::get( uri: '/home', [CustomController::class, 'home'])->name( name: 'home');
Route::get( uri: '/zavody', [CustomController::class, 'zavody'])->name( name: 'zavody');
Route::get( uri: '/zavod/{zav_id}', [CustomController::class, 'zavod'])->name( name: 'zavod')->where(
Route::get( uri: '/vicedenniZavody', [CustomController::class, 'vicedenniZavody'])->name( name: 'viced
Route::get( uri: '/vicedenniZavod/{zav_id}', [CustomController::class, 'vicedenniZavod'])->name( name
Route::get( uri: '/zavodLogin/{zav_id}', [CustomController::class, 'zavodLogin'])->name( name: 'zavodL
Route::post( uri: '/zavodLoginPost', [CustomController::class, 'zavodLoginPost'])->name( name: 'zavodL
Route::post( uri: '/zavodLogoutPost', [CustomController::class, 'zavodLogoutPost'])->name( name: 'zavo
Route::get( uri: '/vicedenniZavodLogin/{zav_id}', [CustomController::class, 'vicedenniZavodLogin'])->
Route::post( uri: '/vicedenniZavodLoginPost', [CustomController::class, 'vicedenniZavodLoginPost'])->
```

V routeru máme definovány veškeré podadresy naší webové aplikace. Router nám tuto adresu vezme a přesměruje nás do kontroleru na odpovídající funkci. Pokud zde máme get metodu, anebo post metodu, tak zde můžeme pomocí regulárních výrazů vyfiltrovat data, které chceme znát, od těch, které by mohly naši aplikaci poškodit. Dále se zde používají takzvané divoké karty, které se píší do složených závorek a router přijme cokoliv, co zde uživatel nebo program zadá, a předá to dál pod názvem dané divoké karty. Taktéž se to dá filtrovat pomocí funkce where a pomocí regulárního výrazu.

3.2.5 Controlery

Pro účely tohoto projektu byly vytvořeny dva kontrolery. Jeden pro autentifikaci uživatelů a druhý pro logiku stránek. V kontroleru pro autentifikaci uživatelů jsou vytvořeny funkce pro přihlášení, odhlášení a vytvoření uživatelů. Vytvoření neboli registrace, byla však pro účely tohoto projektu zablokována, neboť si klient nepřeje, aby se uživatelé mohli zaregistrovat sami, ale chce aby je mohl vytvořit pouze on ze své vlastní administrátorské webové aplikace. Druhý kontroler v sobě má funkce pro jednotlivé stránky a provádí vytahování dat z databáze. Pokud je potřeba, tak naformátuje do požadované podoby a pošle je dál pohledu. Dále jsou zde funkce, které naopak berou informace z postu a ukládají je do databáze. Každá funkce v tomto kontroleru také začíná tím, že ověří, zdali je uživatel přihlášen.

3.2.6 Pohledy

Pohledy nám zde pomáhají s psaním kódu pohledu stránky, který vidí uživatel. Tyto pohledy se píší v jazyce Blade. Ten nám umožňuje základní řízení toku kódu, jakým jsou podmínky a cykly. Pro ušetření výpočetního výkonu serveru se však tento kód pak překládá do čistého PHP. Také zde používáme dědičnost jako v objektově orientovaném programování. Proto máme pro účely tohoto projektu vytvořený jeden pohled, ve kterém máme nadefinovanou hlavičku a patičku stránky a ostatní pohledy od tohoto hlavního dědí. Používáme zde také dynamické doplňování z proměnných, které byli předány kontrolerem. Toto doplňování se píše pomocí dvou do sebe vnořených složených závorek. Toto všechno nám umožní psát velice univerzální kód. V tomto projektu se toto používá například u detailu závodu, kde je vytvořený jeden pohled pro všechny závody. Navbar je rozdělený podle toho, jestli je uživatel přihlášený nebo ne a podle toho má také dané možnosti kam se může dostat.

4 Závěr

Na základě přání klienta byl vyvinut systém pro uživatele na přihlašování na závody v orientačním běhu v rámci klubu. Systém respektuje, ale do značné míry modernizuje architekturu databáze. Z čehož vyplívaly i značné problémy s implementací.

Systém je vytvořen v jazyce PHP a implementuje relační databázi MySQL.

5 Zdroje

1. KULHAN, Jakub. Normalizace relačních databází. *Programujte.com* [online]. 2008 [cit. 2022-03-14]. Dostupné z: http://programujte.com/clanek/2008071900-normalizace-relacnich-databazi/
2. ORM (Object Relational Mapping): automatizuje záznam dat. *hwlibre.com* [online]. [cit. 2022-03-14]. Dostupné z: https://www.hwlibre.com/cs/orm-object-relational-mapping/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+hwlibreweb+%28Hardware+libre%29
3. Laravel docs. *Laravel.com* [online]. [cit. 2022-03-14]. Dostupné z: https://laravel.com/docs/9.x
4. Laravel docs.: Eloquent. *Laravel.com* [online]. [cit. 2022-03-14]. Dostupné z: https://laravel.com/docs/9.x/eloquent
5. Laravel docs.: Blade. *Laravel.com* [online]. [cit. 2022-03-14]. Dostupné z: https://laravel.com/docs/9.x/blade
6. MySQL Documentation: What is MySQL?. *Mysql.com* [online]. [cit. 2022-03-14]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html

6 Seznam obrázků

7 Přílohy

7.1 Schéma databáze