VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PROGRAM PRO PLÁNOVÁNÍ ROZVRHŮ

TIMETABLE PLANNING SOFTWARE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. DANIEL VOSÁHLO**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAROSLAV DYTRYCH, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Vosáhlo Daniel, Bc.**
Program: Informační technologie Obor: Informační systémy
Název: **Program pro plánování rozvrhů**
Timetable Planning Software
Kategorie: Web

Zadání:

1. Seznamte se s problematikou plánování rozvrhů zkoušek a výuky na FIT VUT v Brně a s programovacími jazyky využitými v nástrojích pro jeho podporu.
2. Prostudujte dostupné programy pro podporu této činnosti a vstupy a výstupy plánování z předchozích semestrů.
3. Navrhněte nový program pro plánování rozvrhů výuky a zkoušek, který poskytne vhodnou podporu rozhodování při manuálním umisťování rozvrhových oken a průběžné automatické kontroly splnění požadavků. Při plánování zkoušek bude umožněno automatické rozmístění do místností.
4. Implementujte navržené řešení.
5. Zhodnoťte dosažené výsledky a vytvořte stručný plakát prezentující výsledky práce.

Literatura:

- Dle doporučení vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2 a 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Dytrych Jaroslav, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 6. listopadu 2018

Abstrakt

Tato diplomová práce se zabývá návrhem a tvorbou víceuživatelské webové aplikace pro plánování rozvrhů výuky a zkoušek na Fakultě informačních technologií Vysokého učení technického v Brně. Aplikace bude uživateli poskytovat chytrou podporu při manuálním umísťování oken rozvrhu. Tato podpora bude díky algoritmu pro předpočítání nejdelších kombinací předmětů se společnými studenty umožňovat v reálném čase kontrolovat a upozorňovat na předměty, které se časově překrývají, mají společné studenty a zobrazovat uživateli informace o počtu hodin přednášek nebo zkoušek pro studenty ve dni. Aplikace bude umožňovat také kontrolu splnění požadavků vyučujících daného předmětu, automatické rozdělení zkoušek do místností a registrovaným uživatelům umožní vyzkoušet si vytvořit vlastní verzi rozvrhu a pokud si budou myslet, že je jejich verze lepší než oficiální, budou jí moci navrhnout autorům rozvrhu přímo v aplikaci. Díky mechanizmu zaznamenávání historie změn oken bude uživatel schopen rychle a efektivně vrátit zpět nechtěné změny. V této práci bude dále detailně popsán postup a problémy při tvorbě rozvrhu a všechna kritéria, která musí rozvrh splňovat.

Abstract

This master's thesis describes the design and implementation of a multi-user web application for scheduling timetables and exams at the Faculty of Information Technology of the Brno University of Technology. This application will provide the user with supporting tools for the manual placement of the timetable windows. Thanks to the algorithm for pre calculating the longest collisions, this application can detect collisions in real time and display information about the number of lectures or exams for students in one day. It will also automatically check the teacher's requirements for the course timetable and automatically allocate exams to rooms. Registered users can create their own timetables and if they think that their version is better than the official one, they can propose it to the administrator directly in the application. Thanks to the history recording mechanism for timetable windows, users will be able to quickly and efficiently return unwanted changes back. In this thesis I will also describe the procedures and problems which occurs in the planning process and all the criteria that the schedule must meet.

Klíčová slova

Webová aplikace, plánování rozvrhů, informační systém, podpora při plánování rozvrhu, nalezení nejdelších kolizí, dynamické kontroly požadavků

Keywords

Web application, schedule planning, information system, schedule planning support, longest collision detection, real time requirement checking

Citace

VOSÁHLO, Daniel. *Program pro plánování rozvrhů*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Dytrych, Ph.D.

Program pro plánování rozvrhů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doktora Jaroslava Dytrycha. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Daniel Vosáhlo
21. května 2019

Poděkování

Na tomto místě bych rád poděkoval Ing. Jaroslavu Dytrychovi, Ph.D., který mě v práci vedl a poskytl mi cenné odborné a praktické rady pro řešení mé práce.

Obsah

Kapitola 1

Úvod

Tato práce se bude zabývat tvorbou webové aplikace pro vytváření rozvrhů výuky a zkoušek pro Fakultu informačních technologií Vysokého učení technického v Brně.

Tvorba rozvrhů pro vysoké školy představuje netriviální problém. Autoři musí řešit, že studenti studují určité obory, které mají různé povinné a povinně volitelné předměty. Všechny tyto studenty a předměty je také nutné rozdělit do volných učeben. Dalším problémem je fakt, že jednotlivé předměty mají diametrálně odlišné kapacity. Existují předměty pro desítky studentů, a zároveň existují předměty pro stovky studentů. Také je nutné naplánovat rozvrh pro zkoušky v časově omezeném zkouškovém období pro všechny studenty. Nesmí se zapomínat i na pedagogy, kteří nemohou být v jeden čas na více místech.

V dnešní době již existují programy, které jsou schopny s řešením tohoto problému pomoci. Dokáží automaticky rozmisťovat předměty do učeben, rozdělovat studenty do skupin a zpřístupňovat aktuální verze rozvrhů na Internetu. Problém je v tom, že většina těchto programů je vytvořena obecně, aby pokryla požadavky co nejvíce škol. Programy tedy nemusí vždy poskytovat plnou podporu pro specifické požadavky, a kvůli své obecnosti mají vysokou náročnost přípravy vstupů. Toto může mít za následek, že mohou být větší přítěží než pomocníkem.

Zjistil jsem, že neexistuje nástroj, který by vyhovoval naší fakultě, a že rozvrh je s pomocí několika málo pomocných nástrojů tvořen ručně. Tyto nástroje jsou navíc již zastaralé a zcela nevyhovující. Následkem toho je, že tvorba první verze rozvrhu zabere v průměru 10 až 12 hodin a následná kontrola jeho kvality či úpravy jsou velmi zdlouhavé a pracné. Rozhodl jsem se tedy, jako cíl své práce vytvořit webovou aplikaci pro tvorbu rozvrhů, která bude autorovi v reálném čase zobrazovat upozornění při špatném umístění okna, kolizi vyučujících nebo při porušení požadavku. Autor bude mít také k dispozici informace o počtu předmětů nebo zkoušek pro jednotlivé studenty ve dni. Aplikace bude víceuživatelská, takže i studenti si budou moci zkusit vytvořit vlastní rozvrh, a pokud by se jim zdál lepší než oficiální, tak jej budou moci navrhnout správci přímo v aplikaci. Díky mechanizmu uchovávání historie umístění oken rozvrhu budou moci uživatelé nebo správci při zpětném krokování snadněji pochopit logiku a postup při tvorbě daného rozvrhu. Věřím, že díky mému řešení budou mít autoři rozvrhu usnadněnou práci, budou mít více času pro jeho optimalizaci, a že tento fakt ocení i budoucí studenti.

V kapitole 2 se budu nejprve věnovat detailnímu vysvětlení mechanizmu a podmínkám plánování rozvrhu na fakultě Informačních technologií a rozeberu různé možnosti řešení. V kapitole 3 budu popisovat vstupní data, se kterými bude aplikace pracovat a popíši jejich strukturu a možné nekonzistence, které se mohou objevit. Dále v kapitole 4 popíši technologie, které budu používat, a v kapitole 5 popíši podrobný návrh celé aplikace včetně

zhodnocení současného stavu a popisu použitých algoritmů. V kapitole 6 se budu věnovat popisu finální implementace. Na závěr se budu věnovat popisu testování 7 a zhodnotím dosažené výsledky 8.

Kapitola 2

Plánování rozvrhů

Rozvrhy má většina z nás spojené se školní docházkou. Ty se však používají v mnoha odvětvích a oblastech. Ve velkých i středních firmách se plánují rozvrhy výroby, rozvrhy školení zaměstnanců, rozvrhy umístění zaměstnanců na pobočkách a tak dále. Pro plánování rozvrhů se dnes používají i softwarové nástroje. Už není nutné plánovat rozvrhy ručně na papír, a poté je vypisovat na tabule. Dnes máme k dispozici programy, které jsou schopny do značné míry tento proces urychlit a usnadnit. Také můžeme výsledné rozvrhy vystavit na Internetu a uživatelé se na ně mohou kdykoli podívat a sledovat případné změny.

Plánování rozvrhů je poměrně složitá a komplexní činnost. V zásadě se jedná o řešení problému, jak rozmístit časové aktivity velké skupiny lidí do určitého omezeného časového intervalu s omezenou kapacitou míst tak, aby se jednotlivé společné aktivity nepřekrývaly. Také je velice žádoucí, aby aktivity byly seskupeny do nějakých celků a nebyly rozesety náhodně v tomto intervalu. Tomuto problému se říká „Timetable Problem“ a jedná se o NP-úplný problém [2]. Neexistuje tedy deterministický algoritmus, který by uměl najít optimální řešení v polynomiálním čase.

2.1 Struktura výuky na FIT VUT

Abychom mohli porozumět problematice tvorby rozvrhů, musíme nejprve znát strukturu výuky na Fakultě informačních technologií Vysokého učení technického v Brně (dále jen FIT). Na FIT se výuka dělí na 3 stupně studia – bakalářské, magisterské a doktorské studium. Studijní plán se dále dělí na ročníky. Bakalářský se skládá ze 3 ročníků, magisterský ze 2 a doktorský studijní plán také ze 2. Magisterské studium se také dělí na různé studijní obory. Výuka probíhá v akademickém roce, který se skládá ze zimního a letního semestru. Každý semestr trvá 13 týdnů. Po těchto 13 týdnech následuje ještě zkouškové období, které trvá 5 týdnů.

Předměty se ve studijních plánech dělí na povinné, povinně volitelné a volitelné. Dále jen P, PV a V předměty. P předměty se vyučují pouze v 1 semestru akademického roku a jsou vázány na ročník. Mohou si je zpravidla zapsat pouze studenti ročníku, pro který jsou určeny nebo studenti vyššího ročníku. To samé platí pro některé PV a V předměty. Pokud student úspěšně neabsolvuje nějaký P předmět, musí si tento předmět znovu zapsat a absolvovat jej, jinak je jeho studium ukončeno. Student může překročit standardní dobu studia, na kterou je studijní plán navrhnut. Studenti také mohou mít ze zdravotních důvodů nebo z důvodu zahraničního studia vytvořený individuální studijní plán.

Na FIT mohou mít předměty kromě přednášek také různé druhy cvičení. Existují demonstrační cvičení, která slouží například pro rozšíření výuky o složitější příklady. Tato demonstrační cvičení mohou být i povinná. Dalšími typy jsou numerická cvičení, počítačová cvičení a laboratorní cvičení.

Jednotlivé předměty si student registruje sám. Toto neplatí pro zimní semestr 1. ročníku, kde jsou studentovi P předměty zaregistrovány automaticky. Předměty se registrují obvykle na přelomu března a dubna, a to na celý příští akademický rok. Na začátku srpna se otevírá pro všechny studenty, kromě studentů 1. ročníků, doregistrace předmětů. Student si tedy dodatečně může změnit své zapsané předměty a v 1. týdnu každého semestru se otevírají přeregistrace pro všechny studenty.

2.2 Plánování rozvrhů na FIT VUT

Na FIT se plánuje rozvrh pro všechny stupně studia, ročníky a obory najednou centrálně. Pokud mají vyučující nějaké speciální požadavky, vyplní je do tabulky požadavků pro výuku a zkoušky. Toto bude popsáno detailně v kapitole 2.3.

V průběhu roku je potřeba vytvořit 4 rozvrhy. Jednotlivé rozvrhy se plánují v jiných částech roku, ale proces plánování má stejnou strukturu. Nejprve se začíná sběrem požadavků od vyučujících, následuje zpracování těchto požadavků a import dat z informačního systému. Poté se může začít s tvorbou samotného rozvrhu. Po dokončení rozvrhu je jeho předběžný návrh zveřejněn a nastává sběr a zpracování připomínek od studentů a pedagogů. Teprve po zpracování či vypořádání všech připomínek může být zveřejněna oficiální verze rozvrhu.

Jako první se v roce plánují zkoušky pro současný letní semestr. Sběr požadavků začíná začátkem února a finální rozvrh bývá zveřejněn během března. Druhý se plánuje rozvrh pro zimní semestr následujícího akademického roku. Proces začíná začátkem dubna a je hotov začátkem května. Třetím rozvrhem je rozvrh pro letní semestr, kde sběr požadavků začíná začátkem května a rozvrh je hotov koncem května. Nakonec se v říjnu plánují zkoušky pro zimní semestr.

Proces tvorby rozvrhů je také komplikován faktem, že některé předměty jsou vyučovány na jiných fakultách. Na těchto fakultách mohou probíhat přednášky, numerická cvičení, počítačová cvičení i zkoušky. Do rozvrhu se v tomto případě umístí místnosti z jiné fakulty. Pokud se počítačová cvičení konají na FIT, tak se konají v Centru výpočetní techniky, dále jen CVT. CVT má vlastní rozvrh, který je také nutné naplánovat.

Z výše uvedeného časového harmonogramu a složitosti problému vyplývá, že tvorba rozvrhu je složitá. Toto je umocněno faktem, že na samotnou tvorbu rozvrhu jsou obvykle dva až tři dny. Na dodatečně změny a jejich zpracování je necelý týden. V kapitole 2.4 detailně popíšu postup při tvorbě rozvrhu výuky a v kapitole 2.5 postup pro sestavení rozvrhu zkoušek.

V dosavadním textu byl mnohokrát zmiňován problém kapacity. Na FIT je v současné době 9 poslucháren, ve kterých mohou probíhat přednášky a také se v nich mohou konat zkoušky. V případě zkoušek může vyučující požadovat, aby studenti mezi sebou měli 1 nebo 2 volná místa. Posluchárny jsou vybaveny audiovizuální technikou, která umožňuje některé z nich propojit. To v případě výuky umožňuje přednášejícímu přednášet v 1 posluchárně a tuto přednášku promítat v dalších posluchárnách. Takto je možné umístit předmět s velkým počtem studentů do více menších poslucháren. Seznam těchto poslucháren a jejich možná propojení je možné vidět v tabulce 2.1 a v tabulce 2.2.

Místnost	Kapacita	Kapacita ob 1	kapacita ob 2
D105	300	150	112
D0206	154	77	59
D0207	90	45	36
E112	156	78	54
E104	72	37	28
E105	72	37	28
G202	80	40	32
A112	64	32	24
A113	64	32	24

Tabulka 2.1: Seznam všech přednáškových místností umístěných na FIT VUT, kde probíhá výuka a zkoušky. V tabulce je u jednotlivých místností celková kapacita a kapacita, pokud studenti mezi sebou mají 1 nebo 2 volná místa [7].

Místnost	Propojit
D105	D0206 + D0207
D105	D0207
D0206	D0207
E112	E104 + E105
E112	E104
E104	E105
G202	-
A112	A113

Tabulka 2.2: Seznam všech přednáškových místností, které lze mezi sebou propojit. Místnosti E104 a E105 zatím propojit nelze, toto by se ale v blízké době mělo změnit.

Jak vůbec tedy takový rozvrh ve výsledku vypadá a jakou má strukturu? Ukázku hotového rozvrhu můžeme vidět na obrázku 2.1. Tento obrázek ukazuje rozvrh pro pondělí a část úterý vytvořený ve staré aplikaci pro plánování rozvrhů. Na obrázku můžeme vidět, že výuka začíná nejdříve v 7:00 a končí nejpozději ve 21:00 a výukové okno trvá 50 minut. Na obrázku si také můžeme všimnout místnosti T10/12, což je zástupná informace odkazující na místnost na fakultě Elektrotechniky a komunikačních technologií.

Den		07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00
		07:50	08:50	09:50	10:50	11:50	12:50	13:50	14:50	15:50	16:50	17:50	18:50	19:50	20:50
Po	T			INM (a		IDA (b									
Po	T10/12					INM (b									
Po	D105			IZP (a)		IDA (a		INP (a			MAT		SIN		2SIN
Po	D0206								PDB						
Po	D0207			IZP (a)		IDA (a			PDB		MAT				
Po	E112			SFC					IUS (a			IZP (a)			
Po	E104								IUS (a			IZP (a)			
Po	E105								IUS (a			IZP (a)			
Po	G202					GZN		I2C		PGRe					
Po	A112		IELe			VYPa			PDI						
Po	A113														
Den		07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00
		07:50	08:50	09:50	10:50	11:50	12:50	13:50	14:50	15:50	16:50	17:50	18:50	19:50	20:50
Út	T10/12					IDA (b									
Út	D105		ISS (b			IDA (a			TIN			IMS			
Út	D0206		ZZN			PGR			TIN			IMS			
Út	D0207		ISS (b			IDA (a		IPZ			MATe		TAMa		

Obrázek 2.1: Ukázka rozvrhu ve staré aplikaci pro pondělí a část úterý zimního semestru roku 2018.

2.3 Požadavky vyučujících

Jak již bylo zmíněno dříve, výuka a zkoušky se plánují centrálně. Z informačního systému se načtou údaje o počtu hodin přednášek, počtu hodin cvičení, povinnosti předmětu pro obory, počtu registrovaných studentů atd. Pokud má nějaký vyučující specifické požadavky, jako například preferovaný den nebo čas, musí tyto požadavky vyplnit do speciální tabulky požadavků. Vyučující mají často i požadavky na dny nebo časy, kdy nemohou učit. Existují dvě tabulky – tabulka požadavků pro zkoušky a pro výuku.
Struktura tabulky požadavků na výuku, která je vidět na obrázku 2.2, je následující:

- Položky vyexportované z informačního systému
  - Název předmětu
  - Pro které obory je předmět P
  - Pro které obory je předmět PV
  - Počet registrovaných studentů
  - Maximální počet studentů v předmětu
  - Garant
  - Seznam přednášejících
- Požadavky jednotlivých vyučujících v předmětu
  - Jméno vyučujícího
  - Typ výuky, pro kterou daný požadavek je
  - Počet přednáškových skupin, které má daný vyučující vyučovat

- Požadovaná kapacita místností na skupinu
- Počet hodin výuky na týden
- Jestli bude výuka probíhat ve všech týdnech a nebo jen v sudém nebo lichém týdnu
- Preferované přednáškové místnosti
- Osobní poznámky pro autora rozvrhu
- Preferované dny, ve které by vyučující chtěl mít předmět umístěn
  * Den v týdnu
  * Od kolika hodin
  * Do kolika hodin
  * Priorita – nízká, střední, vysoká
  * Poznámka
- Termíny, kdy vyučující nemohou učit požadovanou výuku
  * Den v týdnu
  * Od kolika hodin
  * Do kolika hodin
  * Poznámka

Struktura tabulky požadavků na zkoušky, kterou můžeme vidět na obrázku 2.3, je následující:

- Položky vyexportované z informačního systému
  - Název předmětu
  - Garant
  - Počet registrovaných studentů
- Požadavky jednotlivých zkoušejících na zkoušku
  - Požadované rozsazení při zkoušce. Žádné, 1 nebo 2, volná místa mezi studenty
  - Poznámka zkoušejícího
  - Požadavek na termín zkoušky. Na všechny, 1., 2., nebo 3. termín zkoušky
    - Preferovaný termín
      - Od jakého data by vyučujícímu vyhovovalo umístit zkoušku
      - Do jakého data by vyučujícímu vyhovovalo umístit zkoušku
      - Jaký den v týdnu
      - Čas konání od
      - Čas konání do
      - Počet hodin pro 1 kolo
      - Počet kol (turnusů) zkoušky
      - Preferované místnosti
      - Poznámka
    - Nemohu zkoušet
      - Od jakého data
      - Do jakého data
      - Den v týdnu
      - Čas od
      - Čas do
      - Poznámka

V těchto tabulkách jsou nejdůležitějšími údaji dny, kdy vyučující nemohou učit nebo zkoušet. Jedná se o takzvaná tvrdá kritéria pro tvorbu rozvrhu. Tato kritéria a jejich význam bude podrobně popsán v kapitole 2.7. V kapitole 2.4 bude popsáno, jak se při vytváření rozvrhu výuky postupuje a jak se s těmito kritérii pracuje. Postup pro tvorbu rozvrhu zkoušek bude popsán v kapitole 2.5.

P pro	PV pro	Počet	Max.	Garant	Přednášející	Vyučující	Typ	Skupin	Kapacita	Hodin	Týden	Místnosti	Poznámka	Vyhovoval by mi termín					Nemohu učit			
														Den	Od	Do	Prio	Poznámka	Den	Od	Do	Poznámka
MIN	MBI, MIS, MMM	50	90	Zbořil František, doc. Ing., Ph.D.	Zbořil František, doc. Ing., Ph.D.	Zbořil František, doc. Ing., Ph.D.	přednáška	1	90	2	každý			-	08:00	15:50	med					
MPV, MSK	MBI, MGM, MMI	55	90	Dvořák Václav, prof. Ing., DrSc.	Dvořák Václav, prof. Ing., DrSc. Jaroš Jiří, Ing., Ph.D.	Dvořák Václav, prof. Ing., DrSc.	přednáška	1	60	3	každý			pondělí	09:00	16:50	high					
														úterý	09:00	13:50	med					
MBI		27	60	Martinek Tomáš, Ing., Ph.D.	Burgetová Ivana, Ing., Ph.D. Martinek Tomáš, Ing., Ph.D.	Martinek Tomáš, Ing., Ph.D. Martinek Tomáš, Ing., Ph.D.	přednáška přednáška	1 1	60 60	2 2	každý každý			-	08:00	16:50	med		pátek pátek	07:00 07:00	20:50 20:50	
MBI	MIN, MMM, MPV	44	90	Sekanina Lukáš, prof. Ing., Ph.D.	Sekanina Lukáš, prof. Ing., Ph.D.	Sekanina Lukáš, prof. Ing., Ph.D.	přednáška	1	70	2	každý	E104, E105							úterý čtvrtek pátek	13:00 09:00 12:00	15:50 11:50 14:50	Senát kolegium děkana seminář UPSY, schůzky

Obrázek 2.2: Ukázka požadavků na výuku z informačního systému FIT pro zimní semestr roku 2017.

Garant	Stud.	Zkoušející	Rozsaz.	Poznámka	Vyhovoval by mi termín									Nemohu zkoušet					
					Termín	Datum od	Datum do		Čas od	Čas do	Hod.	Počet kol	Pref. místnosti	Poznámka	Datum od	Datum do	Den	Čas od	Čas do	Poznámka
Zendulka Jaroslav, doc. Ing., CSc.	67	Rychlý Marek, RNDr., Ph.D.	ob 2		všechny	2019-01-09	2019-02-01	-	10:00	16:50	2	1		před 1. term. musíme stihnout obhajoby proj.			-	7:00	9:50	školka
																	středa	13:00	14:50	schůzka projektu TARZAN
Drahanský Martin, prof. Ing., Dipl.-Ing., Ph.D.	82	Kanich Ondřej, Ing.	ob 1	Počítáme s předtermínem který proběhne před zahájením zkouškového období + klasicky naplánované 3 termíny zkoušek.	všechny	2019-01-02	2019-02-01	-	10:00	20:50	2	1			2018-12-17	2019-02-01	-	7:00	8:50	
					předtermín	2018-12-14	2018-12-21	-	7:00	20:50	2	1								
		Drahanský Martin, prof. Ing., Dipl.-Ing., Ph.D.															úterý	16:00	20:50	
																	čtvrtek	9:00	12:50	Kolegium děkana
																	čtvrtek	16:00	20:50	

Obrázek 2.3: Ukázka požadavků vyučujících na zkoušky z rozvrhu zkoušek zimního semestru roku 2018 [7].

2.4 Plánování rozvrhu výuky

Tato část kapitoly bude popisovat, jak se v praxi plánuje rozvrh výuky na FIT. Rozvrh se prozatím plánuje dohromady pro sudý a lichý týden. V budoucnu se ale bude plánovat zvlášť pro sudý a zvlášť pro lichý týden. Tedy se nejprve vytvoří rozvrh pro sudé a pak pro liché týdny. Při umisťování předmětů do poslucháren musí autor kontrolovat kapacity a propojitelnost poslucháren. Pro bakalářský studijní program bude dále v textu použita zkratka BIT, před kterou bude umístěno číslo ročníku, a pro magisterský studijní program zkratka MIT. Počty studentů v předmětu se berou z počtu registrovaných studentů v informačním systému. Před začátkem plánování se nejprve rozmístí pevné překážky.

Postup plánování rozvrhu od pana doktora Dytrycha, který ho má na starosti:

1. Pošle se e-mail s žádostí o požadavky na jiné fakulty.
2. Pošle se e-mail s žádostí o požadavky od vyučujících z FIT.
3. Překontroluje se tabulka požadavků, jestli neobsahuje chyby, a přidají se požadavky z loňského roku, u kterých se očekává, že stále platí.
4. Z velkých P předmětů 1BIT se vezmou 2, kde je nejvíce studentů, a umístí se (často lze dát 1. předmět přednáškové sk. A a souběžně 2. předmět sk. B).
5. Dále se umisťují P předměty 1BIT, vždy obě přednáškové skupiny současně.

6. Dále se umístí PV a V předměty pro 1BIT.
7. Pokračuje se předměty 2BIT, přičemž je dobré nahlédnout na předměty se striktními požadavky z MIT, u kterých je >80 studentů, a případně je také umístit.
8. 2BIT nemá být v kolizi s 1BIT – tedy pokud možno ke skupině A z 1. ročníku dávat skupinu B z 2. atd.
9. Postupně dochází na situace, kdy je v rámci požadavků potřeba s předměty hýbat. V patových situacích nezbývá než některé požadavky nesplnit.
10. Umístíme V předměty 3BIT a dále se pokračuje velkými předměty z MIT (oba ročníky současně).
11. Umístění zbytku předmětů a cvičení.
12. Zkontroluje se, jestli mají jednotlivé ročníky a přednáškové skupiny rozumné rozvrhy s minimem kolizí (kolize by měly být řešitelné změnou přednáškové skupiny). Kolize mezi P předměty jsou nepřípustné. Kolize mezi P a PV lze výjimečně tolerovat za předpokladu, že počet společných studentů bude nulový. Termín kolize bude vysvětlen a popsán v kapitole 2.6.
13. Kontrola kolizí mezi přednáškovými skupinami. Je-li kolize mezi 1 přednáškovou skupinou a V předmětem, neměla by být kolize mezi 2. přednáškovou skupinou a jiným V předmětem (výjimky se posuzují individuálně).
14. Kontroluje se, jestli některý vyučující nemá být na 2 místech současně.
15. Zveřejnění návrhu rozvrhu a zpracování zpětné vazby.
16. Případné úpravy.
17. Vložení rozvrhu do informačního systému.

Plánování rozvrhu Centra výpočetní techniky

Nově se bude centrálně plánovat i rozvrh Centra výpočetní techniky. V CVT jsou umístěny pouze počítačové učebny. Na každém patře jsou umístěny 3 učebny, které se dají spojit dohromady do 1 velké učebny. Jako v případě rozvrhu výuky se nejprve umístí pevné překážky a je naplánována klasická výuka. Nakonec se naplánuje výuka pro Univerzitu třetího věku.

2.5 Plánování rozvrhu zkoušek

Tato kapitola bude popisovat organizaci zkoušek a jak se tyto zkoušky v praxi plánují. Každý student má nárok na řádný termín a v případě neúspěchu na maximálně 2 opravné termíny. Termíny zkoušky mohou být pro studenty určeny, takové termíny se nazývají pevné termíny. Nebo může být vypsáno více termínů a studenti si termíny volí sami, potom se nazývají variantní. Pokud jsou dány pevně, do rozvrhu se naplánuje řádný termín s 1. a 2. opravným. Pokud jsou zkoušky organizovány pomocí variantních termínů, vypíše se obvykle 5 až 6 termínů. Student se v tomto případě registruje na termíny sám, a může si vybrat, na

jaký půjde. Garant předmětu může v informačním systému zaškrtnout, že zkoušky centrálně plánovat nechce. V tomto případě si je plánuje a rozvrhuje sám.

Rozvrh se plánuje na všechny dny zkouškového období. Nejprve se zkoušky rozmístí do dnů, a až jsou všechny rozmístěné, tak se teprve jednotlivé zkoušky rozmisťují do učeben. Před začátkem plánování se nejprve rozmístí pevné překážky ve dnech do rozvrhu.

Postup plánování je následující:

1. Pošle se e-mail s žádostí o požadavky na jiné fakulty.
2. Pošle se e-mail s žádostí o požadavky od zkoušejících z FIT.
3. Překontroluje se tabulka požadavků, jestli neobsahuje chyby.
4. Nejprve se plánují všechny termíny velkých předmětů pro 1BIT. Za velký předmět je zde považován předmět s velkým množstvím studentů.
5. Naplánují se všechny termíny zbylých předmětů 1BIT.
6. Naplánují se termíny P předmětů pro 2BIT a 3BIT.
7. Naplánuje se zbytek zkoušek pro 2BIT a 3BIT.
8. Naplánují se zkoušky PV a V předmětů pro BIT.
9. Pro 1MIT se nejprve naplánují všechny termíny pro P předměty MAT a TIN[1].
10. Dále se naplánují všechny termíny velkých předmětů pro MIT.
11. Naplánuje se zbytek zkoušek z P předmětů pro MIT.
12. Naplánují se všechny zkoušky z PV a V předmětů pro MIT.
13. Zkontrolují se kolize. Termín kolize bude vysvětlen a popsán v kapitole 2.6.
14. Zkontrolují se kapacity.
15. Kontroluje se, jestli některý zkoušející nemá být na 2 místech současně.
16. Zveřejnění návrhu rozvrhu zkoušek a zpracování zpětné vazby.
17. Případné úpravy.
18. Vložení rozvrhu do informačního systému.

2.6 Kolize v rozvrhu

Pojem kolize v rozvrhu znamená, že jsou minimálně 2 předměty (zkoušky), které se časově v 1 dnu překrývají a mají společné studenty. Studenti by si tedy museli vybrat, na který předmět (zkoušku) jít. U P předmětů s 1 přednáškovou skupinou kolize nastat nikdy nesmí. To samé platí u zkoušek, kde tento fakt je pevně stanoven v předpisech FIT. Dvě zkoušky z P nebo PV předmětů, které mají nějaké společné studenty, se nemohou konat ve stejný čas. Těmto kolizím se říká hodinové kolize v 1 dni a jsou nejjednodušší na odhalení.

[1] MAT - Matematické struktury v informatice, TIN - Teoretická informatika

Dalšímu typu kolize se říká tranzitivní kolize. Tento typ kolize už není tak jednoduchý na odhalení. Pokud je předmět rozdělen na přednáškové skupiny, může v čase přednášky 1 skupiny být umístěn jiný předmět. Pokud nastala pro nějaké studenty kolize, musejí mít tito studenti možnost navštěvovat 2. přednáškovou skupinu, v jejímž čase nemůže být umístěn předmět, který by tito studenti měli také zaregistrovaný a nemohli na něj chodit v jiném termínu.

U zkoušek nás také zajímá, jestli měl student zkoušky předchozí dny nebo jestli má zkoušky nějaké následující dny. Vezmeme všechny zkoušky v 1 dni a podíváme se na všechny zkoušky v předešlých a následujících 2 dnech a vypíšeme všechny dvojce zkoušek, které mají nějaké společné studenty. Tento seznam pomáhá při tvorbě rozvrhu zkoušek a jeho optimalizaci.

2.7 Kritéria pro rozvrhy

Každý rozvrh musí splňovat určitá kritéria. Kritéria mohou být dána předpisy, požadavky vyučujících nebo požadavky studentů. Tato kritéria se dělí na tvrdá a měkká. Tvrdá kritéria nelze při plánování rozvrhu porušit. Oproti tomu měkká kritéria porušit lze. Tato kritéria se totiž používají spíše k optimalizaci rozvrhu pro vyučující a studenty.

Kritéria pro tvorbu rozvrhu výuky

V následujících seznamech budou vypsána a popsána tvrdá a měkká kritéria pro tvorbu rozvrhu.
Tvrdá kritéria pro rozvrh výuky:

- Vzít v potaz pevné překážky. Například, že posluchárna je každé úterý rezervována pro nějakou akci.
- Pevné termíny přednášek, které jsou vyučovány vyučujícími z jiných fakult.
- Pevná omezení, kdy vyučující nemohou učit. Toto vyplývá z tabulky požadavků z kapitoly 2.3.
- Každý předmět má více přednášejících a žádný z nich nesmí mít paralelně jinou přednášku (hodinovou kolizi).
- Dle povinnosti předmětu nesmí vznikat hodinové ani tranzitivní kolize.
- V a PV předměty s vysokým počtem studentů se nesmí krýt s ničím, co by tito studenti mohli mít zapsané a nemohli na to chodit jindy.
- Brát v potaz, že pokud je přednáška mimo FIT, studenti potřebují čas na přesun.
- Předmět může mít demonstrační a numerická cvičení. Platí pro ně stejná pravidla, jako pro přednášky.
- Termíny numerických cvičení se nesmí krýt s přednáškou pro daný předmět. Numerická cvičení mají standardně více termínů. Student si tedy musí vybrat takový termín, který se mu nekryje s nějakou přednáškou.
- Musí se kontrolovat kapacita učeben vůči počtu registrovaných studentů v předmětu.

Měkká kritéria pro rozvrh výuky:

- Vzít na vědomí požadavek na posluchárnu z tabulky požadavků.
- Vzít v potaz preference vyučujících z tabulky požadavků.
- Dávat pozor, aby nevznikaly velké bloky přednášek v kuse pro studenty a vyučující. Například 8 hodin v kuse bez pauzy na oběd.
- Dávat pozor, aby studenti neměli složité předměty (u kterých je známo, že mají vysoký počet opakujících studentů) po sobě.
- Snažit se plánovat rozvrh tak, aby nebyl roztahaný. Například, aby nebyla většina předmětů v pondělí a úterý, a pak byly 2 předměty v pátek.

Kritéria pro tvorbu rozvrhu zkoušek

V následujících seznamech budou vypsána a popsána tvrdá a měkká kritéria pro tvorbu rozvrhu zkoušek.
Tvrdá kritéria pro rozvrh zkoušek:

- Brát v potaz pevné překážky ve dnech. Například 5. ledna probíhá den otevřených dveří a nebude k dispozici posluchárna D105.
- Brát v potaz pevné termíny zkoušek u předmětů z jiných fakult.
- Brát v potaz dny, kdy zkoušející podle tabulky požadavků nemohou zkoušet.
- Zkoušející mohou mít více zkoušek za den. Nesmí pro ně tedy vznikat hodinová kolize.
- Brát v potaz počet studentů a kapacitu poslucháren. Zkoušející mohou mít požadavek, že studenti mezi sebou budou mít 1 nebo 2 volná místa.
- Nesmí vznikat hodinová kolize zkoušek, které mají společné studenty.
- V 1 dni nesmí být 2 P zkoušky pro obor a ročník. Toto platí jen pro 1. termíny zkoušek.

Měkká kritéria pro rozvrh zkoušek:

- Brát v potaz požadavky zkoušejících na termín a posluchárnu.
- Dávat pozor, aby mezi 1. termíny P zkoušek v ročníku byla alespoň 2 dny pauza. Pro 2. termíny stačí 1 den.
- Brát v potaz, že zkoušky z některých vybraných předmětů by neměly být v 1 týden.
- Snažit se mezi zkouškami v 1 dni mít volnou hodinu pro odpočinek a přesun studentů.
- Při plánování 2. a 3. termínů počítat s tím, že v předmětech s hodně studenty trvá opravování déle.

2.8 Automatická tvorba rozvrhu

Pro autora rozvrhu by ideálním řešením bylo, kdyby mohl vyplnit nějakou tabulku, zmáčknout tlačítko a rozvrh by se vytvořil sám. Nebo aby alespoň mohl používat nějaký chytrý systém, který by mu s plánováním pomohl. Bohužel žádný z komerčních programů pro tvorbu rozvrhu FIT úplně nevyhovuje. Na téma hledání komerčního programu pro potřeby FIT byla vypracována bakalářská práce Porovnání programů pro plánování rozvrhů a zkoušek [10]. Výsledkem této práce bylo, že žádný z tehdy existujících systému nebyl pro FIT vhodný. Toto potvrdil i doktor Dytrych, který je za tvorbu rozvrhu na FIT zodpovědný.

Pro automatizované řešení „Timetable“ problému se často využívá řešení pomocí grafů, algoritmy pro vrcholové barvení grafu a algoritmy pro nalezení maximálního toku v síti. Algoritmus pro barvení grafu pracuje s předměty jako vrcholy grafu a hrany značí sdílení studentů a vyučujících mezi propojenými vrcholy. Díky tomuto je možné problém obarvení převést na problém obarvení vrcholů grafu x barvami, kde x značí x časových úseků rozvrhu. Vrcholy obarvené stejnou barvou pak lze vyučovat současně ve stejný čas [20, 14]. Algoritmus pro maximální tok zase bere každého studenta jako zdroj a počet jeho zapsaných předmětů jako velikost daného zdroje. Každému studentovi také odpovídá 1 výpust a úkolem je maximalizovat tok mezi těmito 2 body při daných parametrech a zajistit, aby tok každého studenta skončil pouze v jeho výpusti [4].

V roce 2015 byly vytvořeny 2 diplomové práce, které se zabývaly automatickým generováním rozvrhu přímo pro potřeby FIT. První práce Optimalizátor rozvrhu zkoušek na FIT, používala Constraint logic programming a byla implementována v prostředí SWI Prolog [14]. Druhou prací byl Systém pro pokročilé plánování. Práce řešila tento problém pomocí kombinace genetického a heuristického algoritmu a byla implementována v jazyce Python3 [9]. Obě tyto práce potvrdily, že pro specifické potřeby FIT se nehodí žádný komerční program, a že se jedná o NP úplný problém. Obě tyto práce dokázaly jako svůj výsledek vytvořit pouze suboptimální řešení. Tato řešení však byla nedostatečná a plně nevyhovovala potřebám praxe, viz kapitola 2.9. Proto se dnes žádný z těchto programů nepoužívá.

2.9 Problémy v praxi

Automatická tvorba rozvrhu se v praxi kvůli své efektivitě na FIT neosvědčila. Důvodů je hned několik. Jak bylo uvedeno v kapitole 2.7, rozvrh musí splňovat velké množství kritérií. Toto má za následek, že doba výpočtu výsledného řešení bude velká. To by samo o sobě nevadilo, program by se spustil a nechal by se běžet. Ale co když vytvořené řešení bude špatně? Budeme spouštět program znovu a znovu a čekat, jestli nějaký výsledek bude použitelný? V praxi je na naplánovaní rozvrhu typicky pár dní. A toto by zabralo příliš mnoho času a nemuselo by to vést k výsledku. A co by se stalo, pokud po vytvoření přijatelné verze rozvrhu se nějaký požadavek na rozvrh neočekávaně změní? Nebo co dělat v připomínkovém řízení, kdy se standardně rozvrh dodatečně mění? Můžeme rozvrh upravit dodatečně ručně, ale neznáme jeho systém a můžeme ho narušit. I pro drobné úpravy stejně musíme ručně zkontrolovat, jestli se nenarušily nějaké podmínky. Pokud se narušily, tak můžeme spustit program znovu a doufat a čekat, že program vygeneruje nějaké přijatelné řešení. I kdyby se podmínky nenarušily, tak strávíme poměrně dlouhou dobu kontrolou rozvrhu.

Dalším problémem je, jak zadat vztahy mezi jednotlivými předměty, měkká kritéria, a jak zadat osobní zkušenosti. Například že 2 náročné předměty by neměly být po sobě, nebo že by určitý předmět neměl být večer, případně že ze zkušeností z minulých let víme,

že pokud dáme nějaký předmět před nějaké cvičení z jiného předmětu, tak návštěvnost přednášky bude malá? Pokud bychom nalezli cestu, jak ohodnotit a započítat tyto vztahy, tak bychom museli pro všechny předměty a podmínky vytvořit matici vztahů. Tato matice by byla poměrně velká. A její vyplnění by zabralo velké množství času a před každým semestrem by se musela ručně aktualizovat. Přidáním dalších předmětů nebo podmínek by také její velikost geometricky narůstala, což je opět velice nežádoucí a neřeší to náš problém s časem. Musíme také počítat s tím, že ne všechna vstupní data jsou konzistentní a validní, jak bude ukázáno v kapitole 3.

Kapitola 3

Vstupní data

Jak bylo zmíněno v předchozích kapitolách, vstupní data jsou opravdu rozmanitá. Některé informace se importují z informačního systému, jiné se doplňují z požadavků z minulých let, některé informace se vyplňují ručně, další informace se získávají z externích souborů a zdrojů. Toto není vůbec jednoduchý proces, a proto paralelně s touto prací vzniká Bakalářská práce slečny Aleny Tesařové [19], která se zabývá importem a zpracováním informací z informačního systému, importem a pokročilým zpracováním požadavků na výuku a zkoušky, manuálním vložení dat, jednoduchou úpravou okna rozvrhu a následným pokročilým exportem. Dále jen aplikace na import a export. V této práci také vznikla databáze, kterou bude naše aplikace využívat, jak bude popsáno v další kapitole. Tato aplikace bude tvůrcům rozvrhu sloužit pro naplnění databáze daty, se kterými bude moje aplikace pracovat. Obě tyto aplikaci jsou navrženy tak, aby spolu byly kompatibilní a spolupracovaly spolu, jak bude popsáno v kapitole 5.

3.1 Struktura databáze

Návrh a realizaci databáze vytvořila slečna Tesařová, se kterou jsem poté spolupracoval na úpravách a přidání věcí pro potřeby mé aplikace. Díky této spolupráci mohou obě aplikace využívat stejnou databázi a data.

Schéma výsledné databáze můžeme vidět na obrázku 3.1. Oranžově jsou označeny tabulky, se kterými můj program nepracuje a slouží pro importy a exporty. Tabulky označené zelenou barvou slouží pro uložení informací o předmětech, programech, obdobích, požadavcích, místnostech, typech výuky a typech termínů zkoušek. Moje aplikace používá tyto tabulky pouze pro čtení.

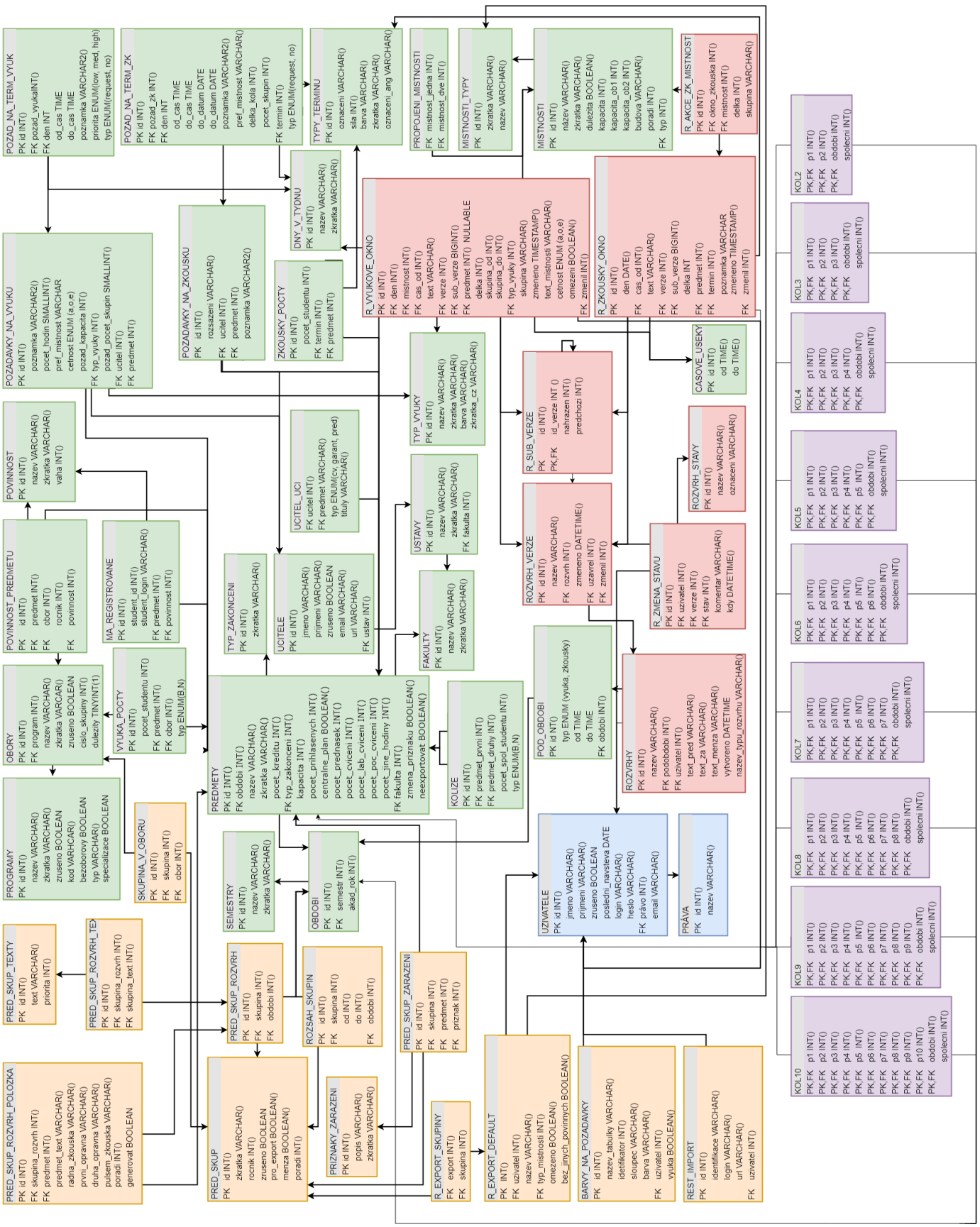
Tabulky označené dalšími barvami již moje aplikace používá a zapisuje do nich. Modře jsou označeny tabulky pro správu uživatelů jejich práv a akcí. Tabulky označené červeně slouží pro uložení informací o rozvrzích. V tabulce *ROZVRHY* je uloženo, jestli je to rozvrh výuky nebo zkoušek. Dále je uložen název, pro jaký akademický rok a semestr je, kdo ho vytvořil a sloupec pro speciální název rozvrhu. Tento sloupec se vyplňuje, pokud se jedná o speciální rozvrh, například pro týden, kdy na FIT probíhá den otevřených dveří. V tabulkách *R_ZMENA_STAVU* a *ROZVRH_STAVY* je uložena historie stavů rozvrhu. V tabulce *ROZVRH_VERZE* jsou pak uloženy jednotlivé verze rozvrhu. Uživatel si může vytvářet více verzí jednoho rozvrhu a pojmenovávat si je. Například „rozvrh pokus1“ atd. Tabulka *R_SUB_VERZE* slouží pro uložení sub verzí dané verze rozvrhu, a toto slouží pro ukládání historie umístění oken v dané verzi. Tuto tabulku a příslušné spouštěče a procedury

jsem vytvořil pro potřeby této aplikace, ale jsou plně kompatibilní i s aplikací pro import a export. Tabulka obsahuje složený primární klíč, který se skládá z atributu **id_verze** specifikujícího verzi rozvrhu a atributu **id**, který specifikuje pořadí umístění v dané verzi rozvrhu. Posloupnost pořadí umístění oken začíná pro každou verzi rozvrhu od 1. Toto bylo implementováno pomocí spouštěče, který po vložení nového záznamu zavolá uloženou proceduru, která nalezne největší číslo pořadí pro danou verzi, inkrementuje ho a nastaví ho novému záznamu. Výhodou použití tohoto postupu oproti postupu za použití **id** s klasickým inkrementováním pro každý záznam každé verze je úspora rozsahu atributu. Pro každou verzi rozvrhu by se vkládaly stovky záznamů. Každý uživatel může mít několik rozvrhů a pro každý třeba i několik desítek verzí. Posledními dvěma atributy jsou atributy **nahrazen** a **predchozi**. Tyto atributy odkazují pomocí sebereflexe na jiné záznamy tabulky pro stejnou verzi a slouží pro zřetězení historie umístění. Toto bude detailně popsáno v kapitole 6. V tabulce *R_VYUKOVE_OKNO* jsou uloženy informace o okně rozvrhu výuky a v tabulce *R_ZKOUSKY_OKNO* jsou uloženy informace o okně zkoušky. Tato tabulka je propojena s tabulkou *R_AKCE_ZK_MISTNOSTI*, ve které je jsou uloženy informace o umístění zkoušek do místností. V databázi jsou také implementovány spouštěče, které po změně okna upraví atribut **zmeneno** v tabulce *ROZVRH_VERZE*.

Fialově jsou označeny tabulky, které byly implementovány přímo pro tuto aplikaci a žádná jiná aplikace je nepoužívá. Tyto tabulky jsou pojmenovány *KOL2* až *KOL10* a slouží pro uložení nejdelších kolizí předmětů. Každá tabulka má složený primární klíč z atributů **p1** až **p10**, které odkazují na atribut **id** tabulky *PREDMETY* a atributu **obdobi**, který odkazuje na dané období. Práce s těmito tabulkami bude popsána v kapitole 5.5.

3.2 Kompletnost a validita dat

Počítáme s tím, že data již budou v databázi nahraná. To ale neznamená, že se můžeme spolehnout na to, že budou plně validní. V procesu importu dat se může stát, že dojde k chybě a některá data budou nevalidní. Do aplikace pro import a export může správce některá data vkládat ručně a při tom například do poznámky zkopírovat uvozovky, nebo může dojít k chybě ukládání některých dat. V obou aplikacích bude moci uživatel ukládat do databáze nějaké své vstupy. Ať už názvy rozvrhů, názvy oken rozvrhu nebo poznámky k oknům atd. Budeme tedy muset zajistit dobré zpracování vstupních dat a zabránit tak nekonzistentnostem nebo dokonce cíleným útokům. Toto bude detailněji popsáno v sekci 6.



Obrázek 3.1: Ukázka databáze se kterou tato aplikace pracuje.

Kapitola 4

Technologie

Tato kapitola bude popisovat technologie, které byly použity v návrhu a implementaci výsledné aplikace.

4.1 MySQL

MySQL je systém řízení báze dat uplatňující relační databázový model. Je multivláknový, multiuživatelský, multiplatformní a dokáže pracovat s velkou zátěží. MySQL byl od počátku optimalizován především na rychlost za cenu menších ústupků ve funkcionalitě. Tyto ústupky jsou však postupně doplňovány, jako například přidání lepší podpory pro uložené procedury, triggery atd. Je také jedním z nejpopulárnějších databázových systémů používajících jazyk SQL a v kombinaci s Apache[1] a PHP tvoří základní software webových serverů [18]. Je vyvíjen firmou Oracle a je dostupný s bezplatnou a komerční licenci.

Současný systém pro tvorbu rozvrhů a všechny jeho podpůrné programy používají pro uložení dat systém MySQL. Tento systém se v praxi osvědčil, a já se ho rozhodl použít i ve své aplikaci, a to ve verzi 5.7.24.

4.2 JavaScript

JavaScript je multiplatformní, interpretovaný, objektově orientovaný skriptovací jazyk, který je ve většině případů používán ve webových stránkách. JavaScript je také dynamicky typovaným jazykem. Je používán na straně klienta pro dynamickou změnu webové stránky. Díky němu můžeme na různé prvky stránky navázat události, které mění obsah a vzhled stránky a tak udělat stránku interaktivní [13]. JavaScript je dnes přítomen ve všech nejpoužívanějších prohlížečích. Je ale nutné dát si pozor na fakt, že v těchto prohlížečích jde jeho podporu vypnout.

V této aplikaci bude použit JavaScript společně s knihovnou JQuery pro interaktivní práci s formuláři a pro dynamickou práci s tabulkou pro rozvrh. Bude také použit pro kontrolu, jestli rozvrh splňuje všechna kritéria a v případě porušení bude uživateli interaktivně poskytovat zpětnou vazbu.

JQuery je rychlá JavaScriptová knihovna pro snadnou práci a procházení HTML elementy a pro snadné zpracování požadavků technologie AJAX a událostí jazyka JavaScript [6]. Díky použití této knihovny nebudeme muset řešit rozdíly v podpoře JavaScriptu v jednotlivých prohlížečích.

[1] Apache je softwarový webový server

4.3 AJAX

AJAX, neboli asynchronní JavaScript a XML, je technika, jak asynchronně získávat data ze serveru po načtení stránky. Díky této technice může web získávat data ze serveru bez nutnosti znovu načtení stránky a může tak simulovat plnohodnotnou interaktivní aplikaci. Tohoto efektu se docílí kombinaci technologií HTML, CSS, JavaScript, XML nebo JSON a PHP [8].

Princip je takový, že uživatel klikne na nějaký ovládací prvek na stránce. Pomocí JavaScriptu zachytíme tuto událost a pomocí funkce *XMLHttpRequest* JavaScript zašle požadavek na skript umístěný na serveru. Obvykle se jedná o skript v jazyce PHP, který z databáze vybere potřebná data a ve formátu XML nebo JSON je pošle zpět stránce. Skript v jazyce JavaScript tuto odpověď přijme a pomocí kombinace HTML a CSS ji zobrazí uživateli bez nutnosti znovu načtení stránky.

Při návrhu se bude muset dát pozor na fakt, že jelikož stránka bude prohlížečem načtena pouze jednou, tak se po zmáčknutí tlačítka zpět v prohlížeči ztratí veškerý dynamicky načtený a neuložený obsah. V této aplikaci bude AJAX použit pro automatické ukládání rozvrhu po každé jeho změně, pro získávání dodatečných informací ze serveru a pro načtení nejdelších kolizí předmětů.

4.4 Bootstrap

Bootstrap je volně dostupná knihovna pro vývoj moderních responzivních webových aplikací. Používá kombinaci HTML, CSS, JavaScriptu a je zaměřena pouze na tvorbu frontendu. Obsahuje návrhové šablony sloužící pro úpravu typografie, formulářů, tlačítek, navigace a dalších komponent rozhraní, stejně jako další volitelná rozšíření. Knihovnu vyvinuli Mark Otto a Jacob Thornton ze společnosti Twitter jako knihovnu, která měla zajišťovat konzistenci mezi interními nástroji firmy [17].

V této práci bude použita ke stejnému účelu. Bude zajišťovat konzistentní vzhled celého systému a jeho responzivnost. Hlavní výhodou využití této knihovny je fakt, že je pravidelně aktualizovaná a reaguje na změny ve funkcionalitě webových prohlížečů. Další nespornou výhodou je, že usnadní případná rozšíření systému dalšími lidmi.

Pro zobrazení a filtrování dat v tabulkách jsem použil rozšíření „DataTables“. Toto rozšíření automaticky rozšiřuje tabulku v HTML o možnost řazení obsahu sloupce, vyhledávání dat v tabulce a o filtrování maximálního počtu zobrazených položek [12].

4.5 PHP

PHP je zkratkou pro hypertextový preprocesor. Je to všestranný skriptovací programovací jazyk, který se používá především pro tvorbu webových aplikací. Jde ale také využít pro tvorbu konzolových aplikací. Je vyvíjen komunitou, má otevřený zdrojový kód a jeho používání je bezplatné. V současné době existuje již ve verzi 7.3 [1].

Při použití ve webových aplikacích jsou skripty v jazyce PHP umístěny na serveru. Po příchodu HTTP[2] požadavku od uživatele je tato stránka na serveru dynamicky sestavena a odeslána uživateli. Vytvoření stránky se skládá z načtení dat z databáze, jejich zpracování a vytvoření dokumentu v jazyce HTML pomocí PHP. [11].

[2]HTTP (Hypertext Transfer Protocol) je aplikační protokolem nad transportním protokolem TCP a je založen na principu požadavek/odpověď.

PHP jsem si vybral kvůli jeho rychlosti, jednoduchosti a vhodnosti pro tvorbu složitých webových aplikací. Dalším důvodem bylo, že s tímto jazykem mám mnoho zkušeností, jak už ze své bakalářské práce, nebo své praxe.

Kapitola 5

Návrh aplikace

Tato kapitola se bude zabývat nejprve zhodnocením současného stavu 5.1. Déle bude popsán návrh samotného informačního systému 5.2 následovaný návrhem celkové struktury aplikace 5.3 a popisem algoritmu pro nalezení nejdelších kolizí 5.5. V závěru této kapitoly bude popsán návrh stránky pro torbu rozvrhu výuky 5.6 a návrh stránky pro tvorbu rozvrhů zkoušek 5.7. V těchto kapitolách budou také popsány mechanizmy kontroly splnění požadavků, mechanizmy pro nalezení kolizí a všechny vytvořené nástroje.

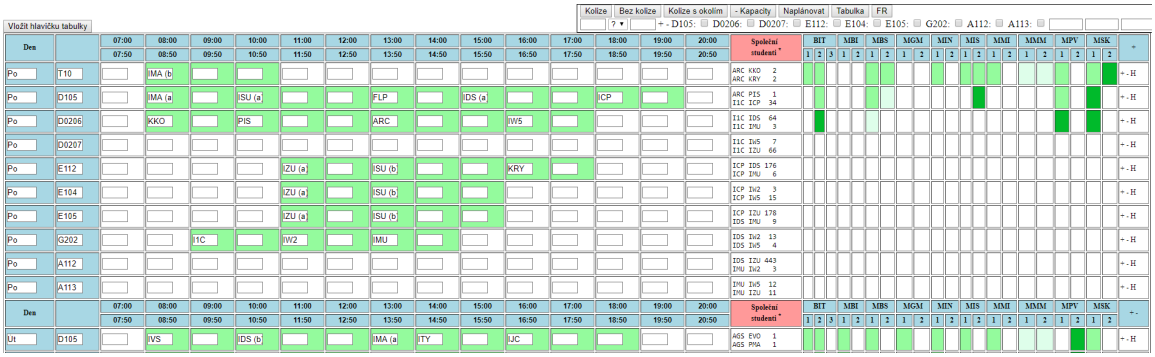
5.1 Současný stav

V současné době se pro tvorbu rozvrhů na FIT používá stará webová aplikace, které byla postupně doplňována nástroji pro alespoň nějakou pomoc při tvorbě rozvrhu. Ukázku okna této aplikace pro tvorbu rozvrhu výuky můžeme vidět na obrázku 5.1. A na obrázku 5.2 můžeme vidět ukázku okna pro tvorbu rozvrhu zkoušek.

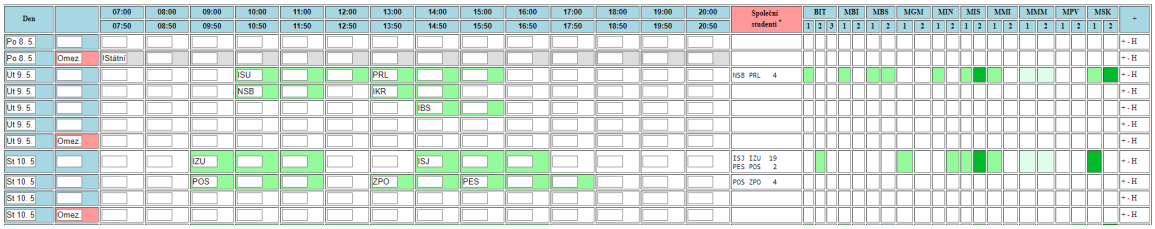
V pravém horním rohu můžeme vidět panel s nástroji. Jsou k dispozici nástroje pro výpis počtu společných studentů ve 2 předmětech, nástroj, který po zadání předmětu vypíše všechny předměty, se kterými není v kolizi, nástroj pro výpis společných studentů s předměty v ostatních dnech, nástroj pro výpočet kapacit poslucháren a nástroj pro zobrazení všech předmětů, které je ještě potřeba naplánovat. Jedním z problémů těchto nástrojů je jejich neintuitivnost. Uživateli, který nezná tuto aplikaci, zabere dlouhou dobu než vůbec pochopí, na co tyto nástroje jsou a jak se používají. Dále nejsou dobře zpracovány výpisy těchto nástrojů. Například při použití nástroje kolize s okolím se vypíše hloupý dlouhý seznam dvojic seskupený podle dnů, ve kterém se nelze posouvat, takže často není vidět celý. Zde chybí možnost skrytí a zobrazení kolizí podle dnů a možnost posouvání se v tomto seznamu.

Hlavním prvkem aplikace je však tabulka rozvrhu. Do levé části se vkládají okna předmětů. Podbarvení buňky indikuje délku rozvrhového okna (přednášky, zkoušky apod.). Ve středu se nachází sekce, do které se vypisují dvojice předmětů z naplánovaného rozvrhu, které mají společné studenty, a v pravé části se nachází graf předmětů. Tento graf uživateli říká, kolik předmětů a jaké povinnosti mají jednotlivé ročníky oborů v daný den. Nevýhodou je, že některá data se musejí vložit přímo do kódu aplikace a nejsou tedy snadno měnitelná. Kromě těchto nástrojů pak aplikace uživateli nijak nepomáhá a uživatel musí provádět kontrolu rozvrhu zcela ručně. Všechny tyto problémy vycházejí z faktu, že tato aplikace byla vytvářena bez nějakého dlouhodobějšího konceptu. Když byl potřeba nějaký nástroj, tak se rychle vytvořil. A na nějaký složitější návrh nebo předělání aplikace nebyl

čas. V posledních letech začali tvůrci rozvrhu úzce spolupracovat se studenty, kteří jim posílají svoje verze rozvrhů s vylepšeními. Pro tento účel poskytuje aplikace možnost exportu rozvrhu ve formátu CSV. Tento exportovaný soubor pak uživatel může poslat druhému uživateli například přes e-mail, a ten si jej může naimportovat a otevřít. Tento způsob je ale nešikovný a nespolehlivý. Uživatel musí využít více aplikací, e-mail se může ztratit a tvůrce rozvrhu si musí ručně vést evidenci návrhů a informace o tom, které již zpracoval.



Obrázek 5.1: Ukázka aktuální aplikace pro tvorbu rozvrhu výuky na FIT.



Obrázek 5.2: Ukázka aktuální aplikace pro tvorbu rozvrhu zkoušek na FIT.

V roce 2017 byla v rámci diplomové práce vytvořena webová aplikace pro plánování a správu rozvrhů [5] pro FIT. Tato aplikace umožňovala načtení dat z fakultních stránek pomocí speciálního skriptu a následný manuální import do aplikace. Umožňovala jednoduchou úpravu těchto dat v aplikaci, manuální vytváření rozvrhů, automatické umisťování zkoušek do učeben a pokročilý export dat do stránek v jazyce HTML. Export do CSV však chyběl. Tato aplikace se v sobě snažila zakomponovat vše od importu, tvorby rozvrhu až po export. Toto mělo bohužel za následek, že ani jeden modul nebyl propracován dostatečně do hloubky a tato aplikace přinesla jen malé zlepšení oproti staré aplikaci. Aplikace navíc dostatečně nepodporovala registraci a správu více uživatelů a uživatelé si nemohli vytvářet svoje rozvrhy. Aplikace byla vytvořena v jazyce Node.js, knihovně Angular a využívala mnoho dalších knihoven. Toto mělo za následek, že aktualizace systému a knihoven způsobila nefunkčnost aplikace. Aplikace by se musela pravidelně aktualizovat, aby dokázala pracovat s novými, nebo alespoň nepříliš starými verzemi knihoven. Díky tomuto aplikace nebyla nikdy uvedena do ostrého procesu plánování rozvrhu a dnes se nepoužívá.

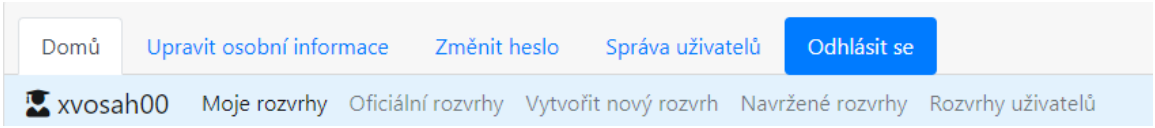
5.2 Informační systém pro správu rozvrhů

Jak bylo řečeno v předešlé kapitole, studenti se chtějí stále více podílet na tvorbě rozvrhu. Proto nová aplikace bude navrhnuta jako víceuživatelský informační systém. Tento sys-

tém bude poskytovat kompletní správu uživatelů, správu rozvrhů a jejich verzí a správu navržených verzí rozvrhů. Systém bude obsahovat 3 typy uživatelů. Administrátory z řad zaměstnanců FIT. Ti budou mít možnost spravovat všechny ostatní uživatele, spravovat jejich rozvrhy, spravovat navržené rozvrhy a zveřejňovat oficiální rozvrhy. Přihlášení uživatelé budou mít možnost importu cizích rozvrhů, vytváření vlastních rozvrhů s verzemi a budou mít možnost navržení verze rozvrhu administrátorům. Nepřihlášení uživatelé pak budou mít možnost zobrazení rozvrhu pomocí přímého odkazu. Podrobný přehled oprávnění jednotlivých uživatelů je zobrazen v diagramu případu užití 5.6. Tento návrh byl z části inspirován diplomovou prací pro tvorbu informačního systému se správou rozvrhů pro taneční studia [16].

5.3 Celkový návrh aplikace

Aplikace bude rozdělena do 4 hlavních oken podle funkcionality. Prvním oknem bude okno pro registraci a přihlášení. Druhým oknem bude okno pro správu rozvrhů a uživatelů v případě administrátorů. Pro lepší přehlednost bude vytvořeno menu, kde každé funkcionalitě bude náležet položka. Po kliknutí na položku bude v okně zobrazeno podokno, které bude mít svoje podmenu, a pod ním budou zobrazené dané informace. Toto můžeme vidět na obrázku 5.3. Díky tomuto návrhu bude mít uživatel všechny informace vždy přehledně zobrazení a dostupné. Zlepší se také udržovatelnost kódu a zjednoduší se případné změny. Posledními okny budou okna pro tvorbu rozvrhu výuky a zkoušek, jejichž návrh bude popsán dále v této kapitole.

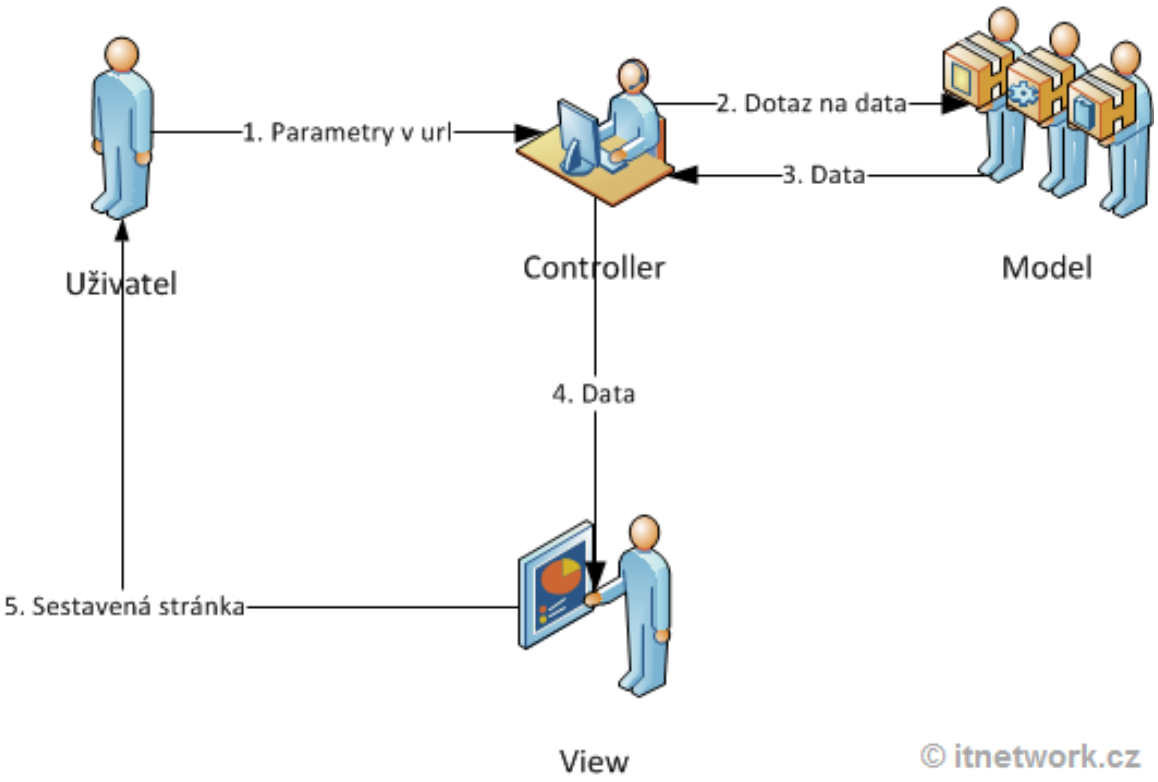


Obrázek 5.3: Hlavního menu aplikace s podoknem pro rozvrhy.

Pro návrh byl zvolen návrhový vzor MVC. O zpracování údajů z adresového řádku a vybrání příslušného kontroleru se bude starat směrovač. Kontrolery (C) budou reagovat na události od uživatele, získají informace od modelu a předají je pohledu. Pohledy (V) se budou starat o sestavení a zobrazení stránky. Modely (M) se pak budou starat o načtení a zpracování dat z databáze [3]. Budeme mít abstraktní třídu *KontrolerKon*, která bude obsahovat metody pro zobrazení pohledu s obsahem a přesměrování. Tuto třídu budou dědit všechny ostatní kontrolery. Aplikace bude fungovat tak, že po zadání požadavku na stránku se vytvoří směrovač, který tento požadavek zpracuje a vytvoří příslušný kontroler. Kontroler získá potřebná data pomocí modelů a ta předá pohledu, který vytvoří finální stránku. Ukázku této architektury můžeme vidět na obrázku 5.4.

5.4 Rozvrhy, verze rozvrhů a stavy rozvrhů

Každý uživatel si bude moci v okně pro správu vytvořit vlastní rozvrhy. Rozvrh bude vždy jednoho typu, výuky nebo zkoušek a bude vytvářen pro daný semestr v daném akademickém roce. Uživatel může například vytvořit rozvrh pro výuku v letním semestru akademického roku 2019. Po vytvoření se tomuto rozvrhu automaticky vytvoří jeho první verze. Uživatel si tuto verzi může přejmenovat nebo vytvořit úplně novou verzi rozvrhu. Tento proces



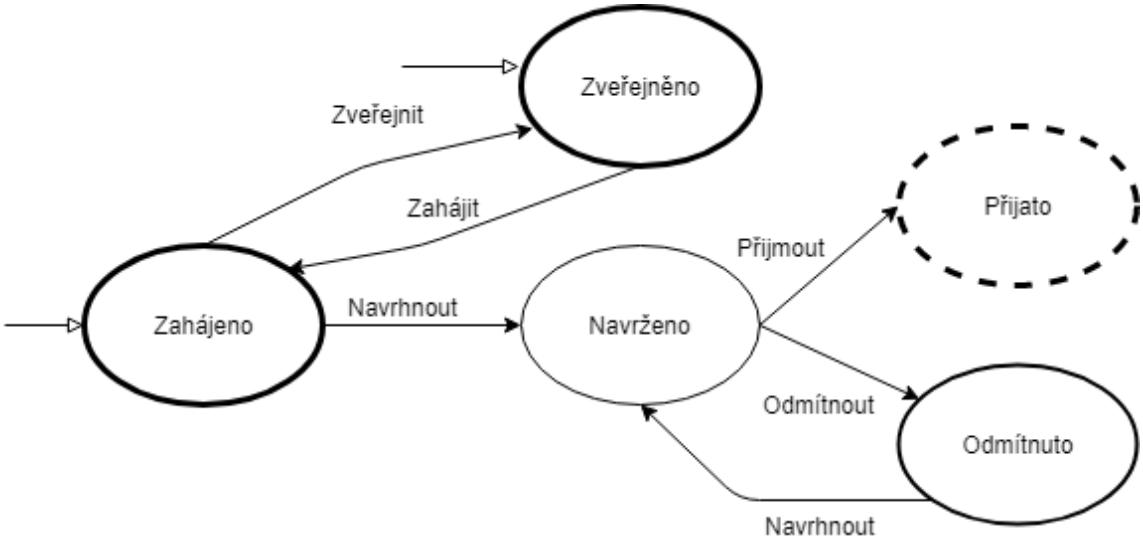
Obrázek 5.4: Ukázka architektury MVC. Obrázek byl převzat ze článku [21].

bude mít dvě varianty. První variantou bude možnost vytvoření nové prázdné verze a druhou variantou bude vytvoření nové verze ze stávající verze. Tím vznikne kopie staré verze s novým názvem. Toto bude poskytovat uživateli možnost zkoušet různé způsoby tvorby rozvrhu a případný rychlý návrat zpět k původní verzi. Ukázku změny stavů můžeme vidět na obrázku 5.5.

Každá verze rozvrhu bude mít svůj stav i historii. Po vytvoření dostane verze stav **zahájeno**. Pokud se uživatel rozhodne navrhnout verzi administrátorovi, tak se tento stav změní na **navrženo**. Administrátor takové rozvrhy může přijmout, v tom případě se vytvoří kopie pro administrátora, nastaví se jí stav **zveřejněno** a verzi uživatele se nastaví stav **přijato**. Stav **přijato** je koncovým stavem. Pokud by uživatel chtěl tuto verzi znovu navrhnout, tak musí z této verze vytvořit novou verzi, které se opět nastaví stav **zahájeno**. Nebo ji může odmítnout a verzi se nastaví stav **odmítnuto**. Administrátor bude mít také možnost ke změně stavu napsat svojí poznámku. Díky tomu se uživatel dozví, co se administrátorovi na verzi nelíbilo.

5.5 Algoritmus pro nalezení nejdelších kolizí

Jednou z funkcí pro chytrou podporu tvůrce rozvrhu je zobrazování počtu společných studentů v různých předmětech. V databázi máme uložené počty společných studentů pro dva předměty. Tato data byla importována z informačního systému a dají se použít pro výpis všech dvojic předmětů ve dni, které mají společné studenty. Pro nástroj pro výpis počtu přednášek nebo zkoušek ve dni pro studenty se však použít nedají. Není v nich totiž uložena



Obrázek 5.5: Stavový automat pro změny stavů verze rozvrhu.

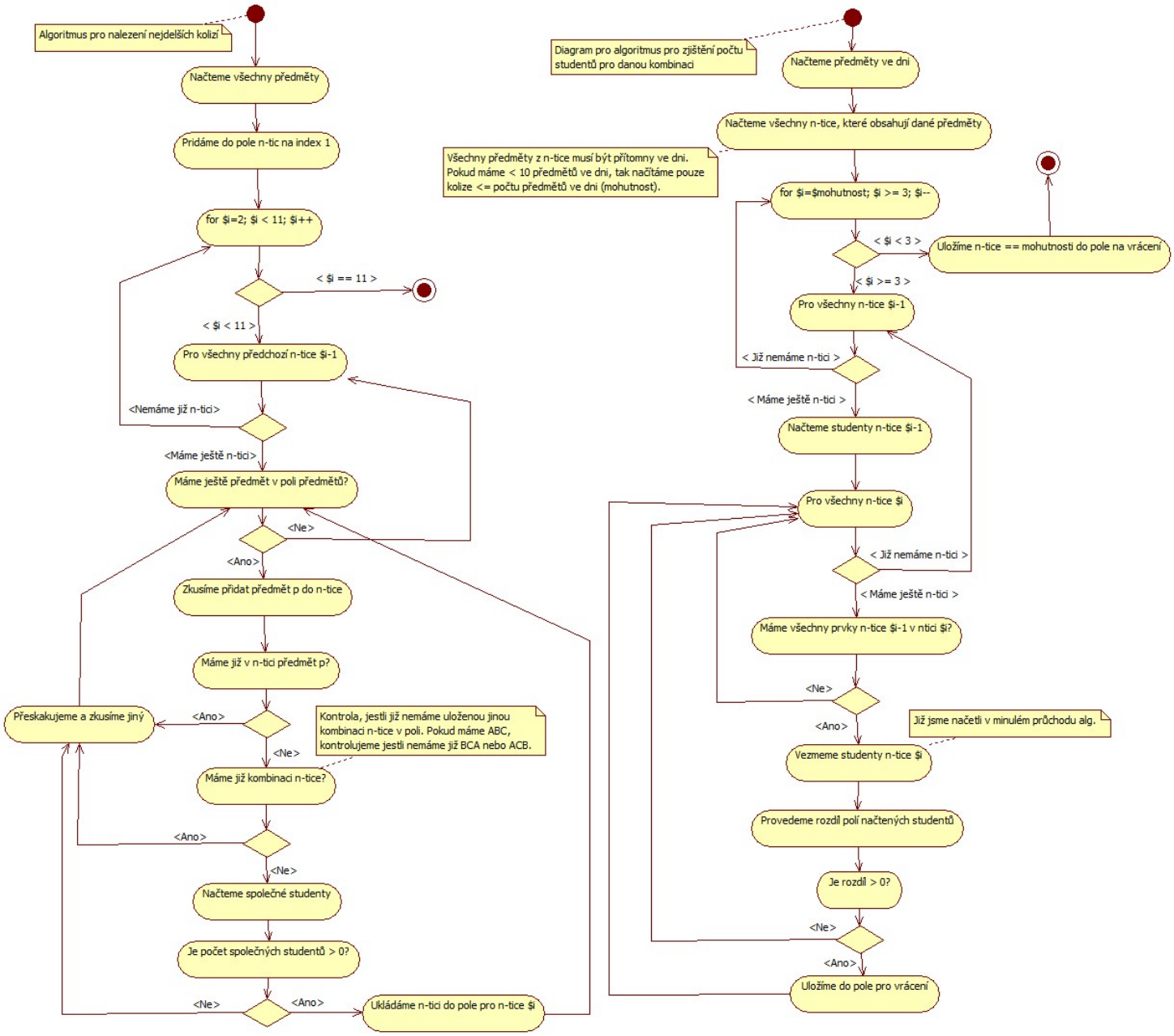
informace o tom, jací studenti mají tyto předměty společné. Takže můžeme mít společné studenty pro AB a BC, ale nevíme jestli máme společné studenty pro ABC. Museli bychom ke každé dvojici zkusit přidat předmět, načíst společné studenty z databáze a takto pokračovat dále až do desetice. Více jak 10 předmětů totiž žádný student mít zapsáno nemůže. A to pro každou možnou kombinaci předmětů ve dni, což je neřešitelné v reálném čase. Proto byl pro tento problém vyvinut postup, který využívá 2 algoritmů a předpočítává všechny možné kolize předmětů dopředu.

První algoritmus slouží k vytvoření všech nejdelších možných n-tic předmětů v zadaném období, které mají společné studenty. Tento algoritmus můžeme vidět znázorněný pomocí diagramu aktivity na obrázku 5.7 vlevo. Algoritmus nejprve načte všechny předměty daného období a uloží je jako n-tice délky 1. Každému předmětu se snaží přidat další předmět a vytvořit n-tice o délce 2. Takto pokračuje až do délky 10. Při tom musí hlídat, aby neměl v n-tici stejně předměty, například n-tici ABA. Také musí hlídat duplicity n-tic. Tímto postupem by totiž mohl dostat n-tici ABC, ale také n-tici BCA. Po vypočtení n-tic dané velikosti jsou tyto n-tice uloženy do daných tabulek do databáze, jak bylo popsáno v kapitole 3.1. Tímto dostaneme všechny kombinace předmětů se společnými studenty v daném období i s jejich počtem. Tento algoritmus je výpočetně velice náročný a je nutné ho spustit s dostatečným časovým předstihem před začátkem plánování rozvrhu.

Mohlo by se zdát, že jsme již dostali data, která potřebujeme, ale není tomu tak. Problémem je, že student, který má zapsané předměty ABC, může mít zapsané také ABCD. Pokud bychom tedy použili tato data a měli v rozvrhu ABCD, započítali bychom tohoto studenta 2×. Pro řešení tohoto problému a pro získání přesných čísel slouží druhý algoritmus, který můžeme vidět na obrázku 5.7 vpravo a je taktéž znázorněn diagramem aktivit. Algoritmus tentokrát postupuje od nejdelších n-tic k nejmenším. Pro všechny n-tice řádu n-1, hledá všechny n-tice řádu n, které obsahují n-tici řádu n-1. Pro každou tuto dvojici pak načte studenty a udělá rozdíl polí těchto studentů. Pokud je tento rozdíl větší jak 0, je n-tice řádu n-1 uložena do výsledného pole pro vypsání se skutečným počtem studentů pro tuto n-tici. Tyto hodnoty se bohužel nedají předpočítat, protože v rozvrhu můžeme mít n-tici ABCD nebo jenom ABC. V případě ABC bychom tedy ztratili informaci o společných studentech této n-tice. Proto se musí tento algoritmus spustit po každé změně okna rozvrhu

Obrázek 5.6: Diagram případů užití nové aplikace.

a bude vyžadovat optimalizaci. Toto bude popsáno v kapitole 6 a výsledek bude zhodnocen v kapitole 7.



Obrázek 5.7: Ukázka diagramů aktivit pro nalezení nejdelších kolizí (vlevo) a pro nalezení počtu předmětů pro studenty (vpravo).

5.6 Okno pro plánování rozvrhu výuky

Okna pro plánování rozvrhu výuky a zkoušek jsou v podstatě 2 interaktivní webové aplikace. Všechen obsah v nich bude načítán, měněn a ukládán dynamicky. Tato okna budou sdílet několik společných nástrojů a rysů, které popíši v této kapitole.

Jedním z požadavků pro návrh bylo vytvořit strukturu stránky a tabulky, která by byla podobná staré aplikaci, a zachovat a vylepšit nástroje, které ve staré aplikaci již jsou. Naopak novým požadavkem byla možnost vkládat předměty do sudých nebo lichých týdnů a možnost tyto předměty samostatně zobrazit. Další novinkou je možnost plánování výuky CVT přímo v rozvrhu výuky a možnost tuto výuku filtrovat v zobrazení.

Pro zobrazení všech podpůrných informací o rozvrhu jsem navrhl systém karuselu. Tento karusel bude umístěn v pravé části tabulky rozvrhu a bude obsahovat několik oken, jejichž zobrazení bude možné měnit. Bude zde umístěno okno s grafem povinností předmětů, okno obsahující kolize s okolím, okno zobrazující n-tice všech předmětů v kolizi pro den a okno zobrazující orientační zátěž předmětů pro studeny ve dni. Uživatel si pak pomocí menu bude moci vybrat pohled, který bude chtít aktuálně zobrazit.

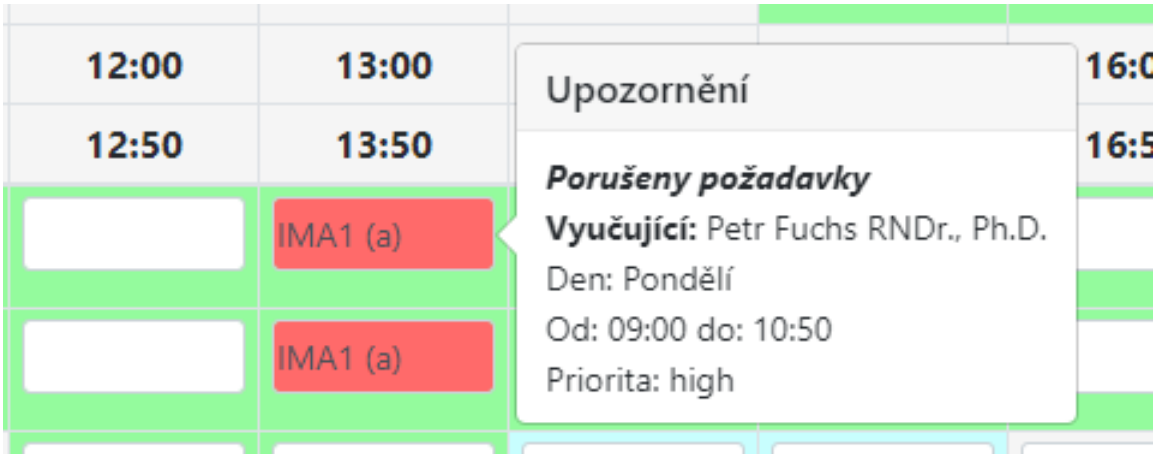
V levé části tabulky se bude nacházet rozvrh rozdělený do dnů, kde každému řádku ve dni odpovídá 1 učebna. Pro každý den budou vypsány nejvíce používané učebny, což urychlí práci uživateli. Uživatel bude samozřejmě mít možnost vkládat další řádky. První buňkou řádku je vždy buňka pro název učebny, za níž následují buňky pro okna výuky. Každá buňka v sobě obsahuje vstup pro zadání názvu předmětu nebo místnosti. Po zadání předmětu bude buňka podbarvena příslušnou barvou pro typ výuky nebo termín zkoušky a bude tak indikovat obsazenost buňky. Pokud bude buňka porušovat nějakou kontrolu, bude příslušnou barvou vážnosti porušení podbarven samotný vstup a buňce bude přidáno okno s typem a popisem chyby, které se zobrazí po najetí na buňku. Toto můžeme vidět na obrázku 5.8. Do buněk pro okna můžeme vkládat také poznámky. Poznámka musí začínat „!“ a nebude brána v úvahu při kontrolách. Uživatel může také do rozvrhu vložit řádky pro poznámky, které budou mít místo zkratky místnosti zadáno „*Omez.*“, a toto bude podbarveno červeně. Tyto řádky nebudou opět brány v potaz při kontrolách.

Kontroly rozvrhu budou probíhat na straně klienta pomocí JavaScriptu. Budou vytvořeny datové struktury, které budou obsahovat načtené předměty s informacemi o předmětu, informacemi o požadavcích pro daný předmět a informace o vyučujících a cvičících. Dále budou načteny všechny místnosti a další potřebné informace. Informace o obsazených buňkách výuky nebo zkoušek budou také umístěny přímo v HTML pomocí atributu *data*. Díky tomuto přístupu nebudeme muset při každé změně načítat data znovu, nebudeme muset znovu zpracovávat vstupní data obsazených buněk a budeme moci pracovat pouze s datovými strukturami.

Prvními kontrolami, které budou probíhat, budou kontroly validity vstupů. Aplikace dovoluje vložit pouze existující místnosti a předměty. Pro vstupy oken rozvrhu je předepsaný formát. Příklad takového vstupu je **2IMA(a)[5-10]+-1**. Kde „2“ znázorňuje numerické označení typu podle tabulky 5.1, „IMA“ zkratku předmětu, „(a)“ nám říká, že se jedná o okno pro přednáškovou skupinu a, „[5-10]“ nám říká, že se jedná o skupiny 5–10 a „+ -1“ znamená, že od délky, kterou máme vypočítánu pro daný typ výuky, máme odečíst 1 hodinu. Pro přidání 1 hodiny bychom použili „+1“. Aplikace také nebude umožňovat mít 2 stejné místnosti ve dni nebo umisťovat buňky rozvrhu do řádku bez místnosti nebo bez označení pro omezení. Toto by mohlo vést k neúmyslnému vytvoření kolize, kdy by v 1 místnosti v 1 čase byly umístěny 2 předměty. Déle budou implementovány kontroly přesahu rozvrhu a překryvu oken rozvrhu.

Musíme také kontrolovat kolize vyučujících, aby žádný z vyučujících neměl v 1 čas více přednášek. Nejprve načteme časový interval, v kterém leží náš předmět, a poté načteme všechny předměty, které do tohoto časového intervalu zasahují. Zde si budeme muset dát pozor, abychom nevybrali náš předmět, který je umístěn pouze v další učebně, nebo naopak abychom nevybrali vícekrát jiný předmět, který je umístěn ve více učebnách. Za stejný předmět budeme považovat předmět, který má stejnou zkratku a začíná ve stejný čas. Kontrola bude také umět automaticky podbarvit všechny předměty, které jsou v kolizi, a naopak po odebrání buňky tyto předměty odbarvit a smazat danou poznámku.

Poslední kontrolou bude kontrola požadavků vyučujících na umístění předmětu. Jak bylo řečeno v kapitole 2.3, existují 2 typy požadavků. Požadavek, kdy vyučující nemůže učit, a požadavek, kdy by chtěl učit. Načteme všechny požadavky pro daný předmět a typ výuky okna a nejprve budeme kontrolovat požadavky nemohu učit. Pokud naše okno bude zasahovat do zadaného intervalu, podbarvíme vstup červeně, vypíšeme poznámku a dále již nekontrolujeme. Není to potřeba, protože stačí, že 1 z vyučujících v tomto intervalu nemůže. Zde si musíme dát pozor na požadavky, které mají nastavený časový interval na celý den a nespecifikují den v týdnu. V tomto případě se jedná o poznámku, vstup podbarvíme světle modrou barvou a pokračujeme v kontrole dalších požadavků. Oproti požadavkům nemohu jsou požadavky chci odstupňovaný podle priority. Máme požadavky s prioritou *low*, *med* a *high*. Těmto prioritám bude odpovídat červené podbarvení s odstínem odstupňovaným podle priority. Pro *low* nejsvětlejší a pro *high* nejtmavší odstín. Okno může porušovat hned více požadavků najednou, v tom případě bude obarveno barvou dle porušeného požadavku s nejvyšší prioritou.



Obrázek 5.8: Ukázka podbarvení buněk a ukázka okna s popisem chyby.

Numerické označení	Význam
	Přednáška implicitně
2	Numerické cvičení
3	Laboratorní cvičení
4	Počítačové cvičení
5	Jiné cvičení
6	Demonstrační cvičení

Tabulka 5.1: Syntaxe numerického označení vstupů pro výuku

5.7 Okno pro plánování rozvrhu zkoušek

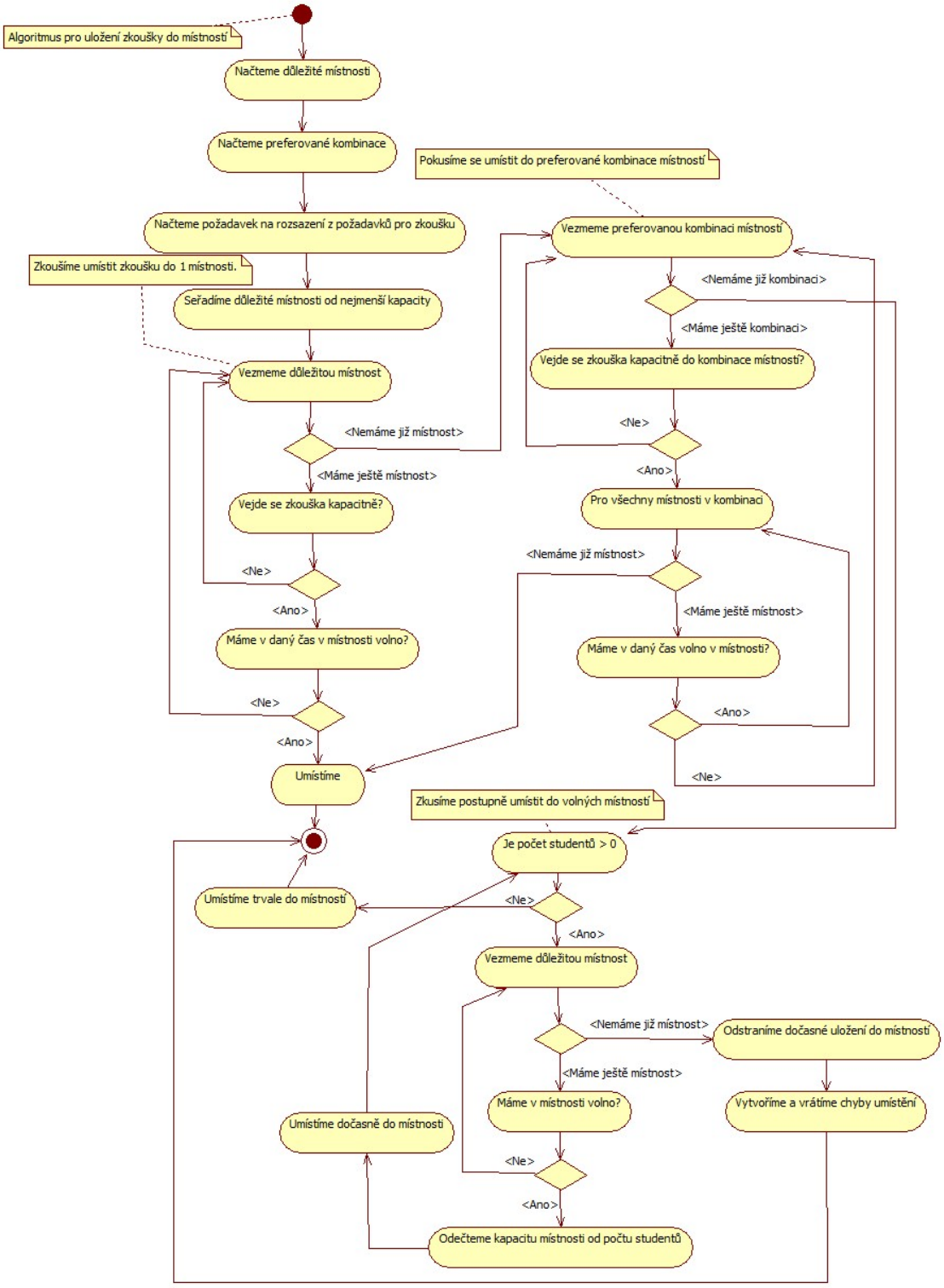
Struktura okna pro plánování rozvrhů zkoušek byla opět navrhnuta podle vzhledu staré aplikace, jak bylo požadováno. Budeme tedy mít pro zkoušky 2 pohledy. Pohled bez místností, ve kterém bude uživatel rozmisťovat zkoušky pouze do dnů, a pohled s místnostmi, kde bude moci tyto zkoušky rozmístit do místností ve dni. Oproti oknu pro výuku zde neplánujeme pouze pro dny v týdnu, ale pro celé zkouškové období. Struktura tabulky pro okna bude jinak pro uživatele stejná. V karuselu pro zkoušky přibude obrazovka pro zátěž v týdnu. Na této obrazovce bude zobrazeno, kolik zkoušek mají daní studenti v daném týdnu.

V pohledu s místnostmi bude implementována stejná kontrola duplicity místností jako u okna výuky a v okně bez místností není dovoleno zkratku místnosti vyplňovat vůbec. Uživatel může pouze vyplnit omezení do vstupu pro místnost. Syntaxe vstupu pro zkoušky je až na numerické označení a absenci skupin podobná. Pro zkoušky máme např. **2IMA(a)(poznámka)+1**. Přibude nám zde možnost umístit poznámku do závorek za skupinu. Tato poznámka bude ovšem muset mít více než 2 znaky. Syntax pro termíny zkoušek je uvedena v tabulce 5.2. Stejně jako u výuky budeme moci do buněk vkládat poznámku pomocí „!“.

Bude zde i drobná změna v kontrole požadavků. Pro zkoušky mohou zkoušející zadávat kromě dnu a hodin ve dni i kalendářní intervaly, kdy omezení platí. Pokud je buňka umístěna mimo tento kalendářní interval, ignorujeme všechny požadavky pro tento interval. Okno pro zkoušky s místnostmi uživateli nabídne možnost rozmístit všechny zkoušky ve dnech automaticky do místností pomocí algoritmu popsaného pomocí diagramu aktivity na obrázku 5.9. Algoritmus načte důležité místnosti seřazené podle velikosti od nejmenší a preferované kombinace místností, které bude moci uživatel zadat do nastavení aplikace. Algoritmus nejprve zkusí umístit zkoušku do 1 důležité místnosti. Pokud se mu to nepovede zkusí zkoušku umístit do nějaké z preferovaných kombinací. Pokud v nějaké místnosti kombinace není místo, zkusí další kombinaci. Pokud se mu nepodaří zkoušku ani teď umístit, tak půjde po důležitých místnostech a zkusí jí umístit tam, kde je místo. V případě neumístění bude uživateli hlásit chybu umístění zkoušky ve dni v daném čase.

Numerické označení	Význam
0	Předtermín
	1. termín implicitně
2	2. termín
3	3. termín
4	4. termín
5	5. termín
6	6. termín

Tabulka 5.2: Syntaxe numerického označení termínů zkoušek



Obrázek 5.9: Diagram aktivit pro automatické umístění zkoušky do místností.

Kapitola 6

Implementace

Pro implementaci serverové části byl zvolen jazyk PHP a podle návrhu byl vytvořen vlastní informační systém s architekturou MVC. V aplikaci je vytvořena metoda pro automatické načítání tříd. Stačí tedy vytvořit nový soubor, který má stejný název jako třída v něm umístěná, a následně stačí vytvořit objekt dané třídy a soubor se třídou bude automaticky načten. V souboru *nastaveni.php* pak uživatel může editovat konfiguraci celé aplikace a v souboru *TableSortRozvrhy.js* může měnit nastavení filtrování a řazení tabulek v aplikaci. Byla vytvořena souborová struktura, která je členěna do složek. Máme složku pro kontrolery, modely, pohledy, skripty pro AJAX, soubory CSS a samostatnou složku pro skripty v jazyce JavaScript. Klientská část je pak implementována pomocí HTML v kombinaci s CSS a knihovnou Bootstrap. Klientská část byla navržena s ohledem na mobilní zařízení a je možné jí na nich používat. O dynamické změny v klientské části se pak stará JavaScript. Skripty v jazyce JavaScript jsou umístěny v externích souborech a tyto soubory jsou pak vloženy do stránky při kompletaci na straně serveru. Tyto skripty berou data z globálních proměnných, které jsou vytvořeny a vloženy do záhlaví stránky.

Chyby jsou v aplikaci zachycovány a zpracovávány pomocí kaskády výjimek. Jako první se zachytávají výjimky *PDOException* způsobené chybou získávání dat z databáze. Tyto výjimky vznikají v modelech. Po zachycení je tato výjimka zpracována, vytvoří se text chyby a metoda vyvolá novou výjimku *Exception*. Tato výjimka je zachycena v kontroleru a chyba vypsána uživateli. Díky tomuto systému je aplikace schopná i přes chybu uživatele na tuto skutečnost upozornit a dále fungovat. Metody v aplikaci také používají kontrolu typů vstupních parametrů a kontrolu návratového typu. Pokud dojde k předání špatného parametru, je zachycena výjimka *TypeError* a aplikace je ukončena. Toto zabraňuje kritickým chybám, které by mohly vést k nekonzistenci dat nebo nepředvidatelnému chování.

6.1 Zabezpečení aplikace

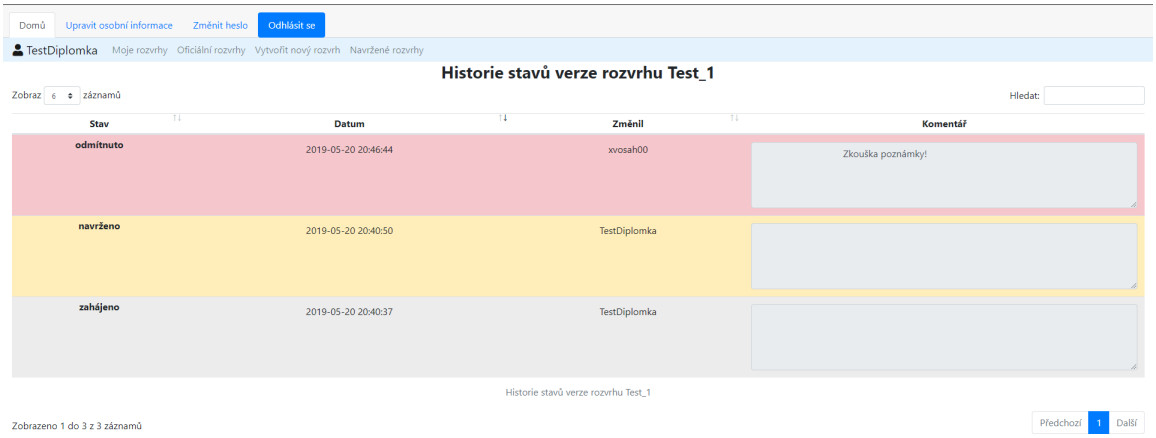
Jelikož se jedná o aplikaci, kterou bude moci využívat kdokoli, je nutné věnovat pozornost i jejímu základnímu zabezpečení. Prvním stupněm zabezpečení je použiti technologie reCAPTCHA[1] u formulářů pro přihlášení a registraci. Toto by mělo zabránit robotům, kteří by chtěli zatěžovat stránku pomocí registrace mnoha uživatelů nebo robotům, kteří by se snažili zaútočit silou na heslo nějakého uživatele. Dalším citlivých bodem jsou skripty v ja-

[1] reCAPTCHA je bezplatný modul od společnosti Google, který je schopen rozpoznat, zda uživatel je člověk nebo stroj.

zyce PHP volané pomocí technologie AJAX. Tyto skripty jsou náchylné k útokům CSRF[2] a je nutné je zabezpečit. V této aplikaci jsou tyto skripty chráněny pomocí CSRF tokenu, který je každému uživateli pravidelně generován a při volání nějakého skriptu je tento token ověřen za pomocí **SESSION**. Posledním citlivým bodem aplikace jsou uživatelské vstupy, které by se daly využít například pro SQL injection. Proto tato aplikace používá objektový databázový ovladač PDO, který poskytuje takzvané *prepare statements*. Tato funkcionalita každý dotaz před jeho provedením nejprve předpřipraví, v zadaných hodnotách upraví speciální znaky a posléze provede. U všech dat, která budou vložena do pohledů a zobrazena uživateli, se provádí nahrazení speciálních znaků. Tímto se zabrání případům, kdy administrátor například omylem zkopíruje do textu uvozovky, nebo případům, kdy by někdo chtěl provést útok na klientský JavaScript jiného uživatele.

6.2 Uživatelské rozhraní systému

Po přihlášení do systému se uživateli zobrazí hlavní okno aplikace, které můžeme vidět na obrázku 6.2. V hlavním menu jsou karty pro práci s rozvrhy, změnu osobních údajů, změnu hesla a v případě administrátora pro správu uživatelů. Na kartě pro práci s rozvrhy pak uživatel může nalézt stránku s vlastními rozvrhy, oficiálními rozvrhy, navrženými rozvrhy, stránku pro vytvoření vlastního rozvrhu a administrátor si pak může zobrazit všechny rozvrhy všech uživatelů. Po kliknutí na rozvrh se uživateli zobrazí všechny verze daného rozvrhu, jak je možné vidět na obrázku 6.3. Zde si může také zobrazit historii změn stavů rozvrhu nebo svůj rozvrh navrhnout administrátorovi nebo v případě administrátora ho zveřejnit. V okně pro navržené rozvrhy pak může uživatel vidět všechny své navržené rozvrhy včetně zamítnutých, zatímco administrátor uvidí všechny rozvrhy navržené uživateli a může si otevřít formulář pro rozhodnutí o změně stavu rozvrhu. V tomto formuláři může změnit stav rozvrhu a přidat k této změně poznámku, která se pak zobrazí autorovi rozvrhu. Historii změny stavů můžeme vidět na obrázku 6.1.

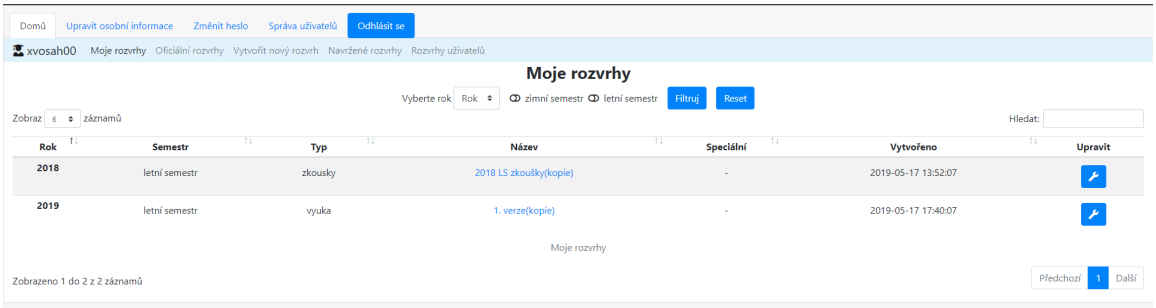


Obrázek 6.1: Ukázka historie verze rozvrhu..

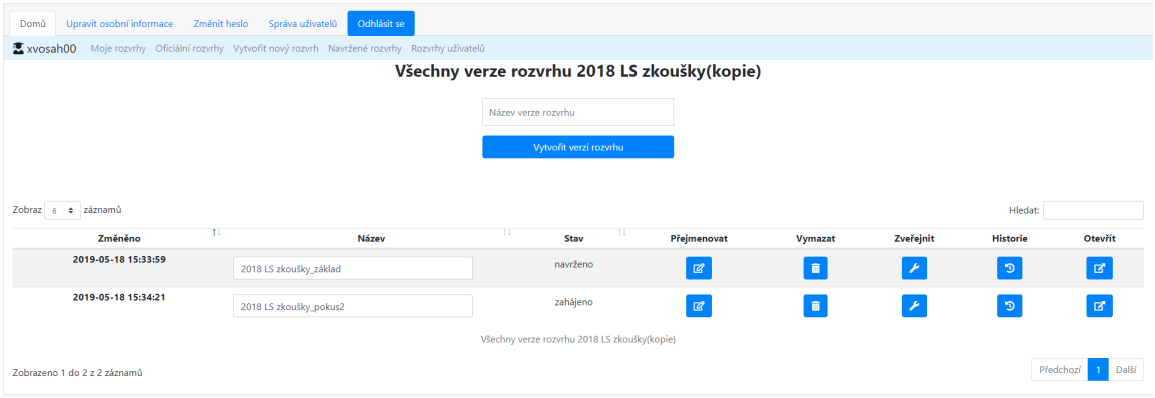
[2]Cross-site request forgery je útok, kdy útočník úmyslně volá nějaké skripty aplikace, snaží se vydávat za autentizovaného uživatele a snaží se z aplikace získat citlivé informace [15]

6.3 Předpočítání nejdelších kolizí

Pro tento účel byl vytvořen skript pro PHP CLI[3], který není spustitelný z aplikace, ale musí se spustit ručně na serveru. Skript je umístěn ve složce *CLISkripty* ve složce s aplikací a pro svůj běh používá soubory *ajaxKostra.php* pro připojení k databázi a *Rozvrhy.php* s modelem a metodami pro práci s rozvrhy. Skript se spouští příkazem **php kolize.php -s ZS -r 2019**, kde **-s** je parametr pro zkratku, **ZS** pro zimní semestr a **LS** pro letní semestr. Skript sám vyhledá období a pokud uživatel zadal neexistující období skript vrátí chybové hlášení. Skript z důvodu optimalizace výkonu vkládá záznamy do databáze průběžně po dávkách po 1000 záznamech. Jelikož skript pracuje pouze s identifikátory studentů a předmětů, které do databáze vkládá administrátor, nemusíme tyto dotazy předpřipravovat. Po dokončení generování n-tic dané délky skript na výstup vypíše počet vložených položek do databáze a délku běhu pro danou délku n-tic.



Obrázek 6.2: Ukázka hlavního okna aplikace.



Obrázek 6.3: Ukázka okna pro výpis verzí rozvrhu.

6.4 Okno pro tvorbu rozvrhu výuky

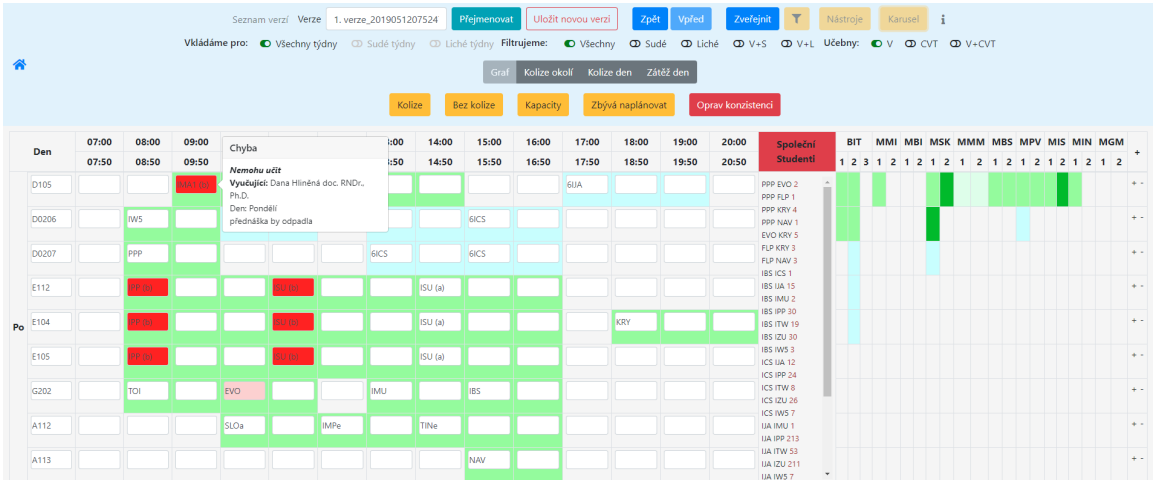
Na obrázku 6.4 můžeme vidět okno pro plánování výuky. V horní části se nachází menu s nástroji, které má pevnou pozici, takže je dostupné na celé stránce. Obsahuje nástroje pro vytvoření nové verze ze současné verze a tlačítko pro změnu jména verze. Následují tlačítka

[3]PHP Command Line Interface. Skripty v jazyce PHP nejsou spouštěny webovým serverem, ale jsou spouštěny pomocí příkazové řádky.

pro ovládání procházení historie, tlačítko pro rychlé zveřejnění nebo navrhnutí a ovládací prvky pro skrývání menu, filtrování zobrazení, skrývání menu karuselu a skrývání menu nástrojů. Pod tímto menu můžeme vidět část tabulky pro rozvrh, která obsahuje reálný rozvrh.

Co vidět není je vnitřní struktura. Pro kontroly rozvrhu je potřebné mít informace načtené v datových strukturách. Pro tento účel jsem využil objektu *Map*, ve kterém jsem podle klíče uložil další objekty. Například objekty jednotlivých předmětů, které obsahují informace o počtu hodin, vyučujících, požadavcích atd., jsou uloženy podle svých zkratek. U buněk pro předměty zase pomocí atributu *data* uchovávám informace o typu výuky, id předmětu, délce okna a skupině. Po změně okna je vstupní text zpracován, pomocí technologie AJAX uložen do databáze a následně uložen do atributů buňky. Do databáze se nejprve vloží nová subverze rozvrhu, dále se vloží záznam do tabulky pro okno výuky a jako poslední se hledá, jestli neexistuje nějaká stará subverze pro toto okno. Tuto starou subverzi hledáme podle verze rozvrhu, dnu, času a místnosti. Pokud jsme nějakou starou subverzi našli, nastavíme jí atribut *nahrazen* id naší nové subverze a naší nové subverzi nastavíme atribut *predchozi* na id staré subverze. Tomuto se říká zřetězení subverzí a je to nezbytné pro nástroj pro procházení historie

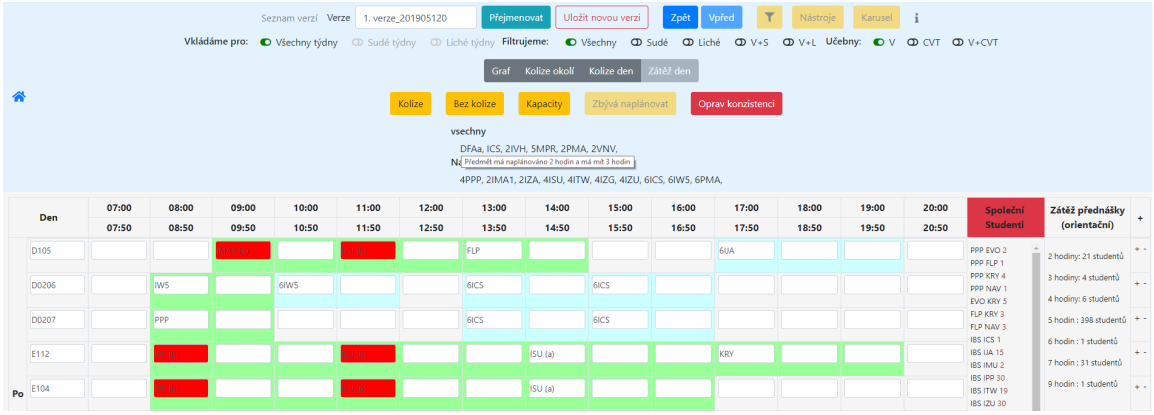
Nástroj pro procházení historie je ovládán tlačítky v hlavním menu a funguje stejně jako v jiných aplikacích. Po stisknutí tlačítka se nalezne současná subverze s odkazovaným oknem výuky. Pokud jdeme zpět, podíváme se, jestli má subverze nastaven atribut *predchozi*. Pokud ne, je jasné, že buňka zde byla poprvé obsazena a budeme ji mazat. Pokud ano, podíváme se na odkazované okno výuky a určíme, jestli budeme mazat, nebo pouze měnit obsah buňky. Vykonáme akci a upravíme id současné subverze v rozvrhu. Pro posun vpřed nejprve načteme další subverzi podle id. Načteme odkazované okno výuky. Pokud nemá subverze nastaven atribut *predchozi*, budeme vkládat. Pokud ano, načteme staré okno výuky a určíme, jestli budeme mazat nebo měnit a upravíme id současné subverze v rozvrhu. Pokud uživatel při procházení historie změní nějakou buňku, bude vytvořena nová subverze a procházená historie smazána.



Obrázek 6.4: Ukázka okna pro plánování výuky.

Z původní aplikace byly převzaty a vylepšeny principy nástrojů, které lze zobrazit v menu pro nástroje. Prvním nástrojem je nástroj pro zobrazení společných studentů. Ve staré aplikaci tento nástroj pracoval pouze s dvojicemi předmětů. Nyní díky předpočíta-

ným kolizím můžeme zadat až 10 předmětů. Dalším nástrojem je nástroj zobrazující všechny předměty, s kterými nemá zadaný předmět společné studenty. Nástroj kapacita zobrazuje důležité učebny a po vybrání vrátí jejich společnou kapacitu a kapacitu ob 1 nebo 2 místa. Po zadání předmětu vrátí kapacitu daného předmětu. Nástroj zbývá naplánovat je o něco zajímavější. Tento nástroj zobrazuje předměty, které ještě nejsou naplánované v rozvrhu nebo nemají naplánován dostatečný počet hodin. Toto je možné vidět na obrázku 6.5. Nástroj také zobrazuje předměty, které mají naplánováno více hodin než mají mít, a předměty, které jsou naplánované a nemají žádné požadavky. Nástroj tyto údaje zobrazuje odděleně pro všechny četnosti. Takže uživatel bude vědět, že mu například zbývá naplánovat předmět IMA v lichém týdnu. Posledním nástrojem je nástroj pro opravu konzistence rozvrhu. Při importu starých rozvrhů do databáze může nastat situace, kdy text buňky neodpovídá délce buňky. Toto nastává, pokud stará aplikace pracovala s jinými základními délkami rozvrhových oken pro dané předměty. Moje aplikace na tuto skutečnost uživatele upozorní při načtení rozvrhu. Jemu poté stačí stisknout tlačítko pro opravu a texty buněk budou opraveny v rozvrhu i v databázi. Uživateli pak bude zobrazen seznam změněných buněk.

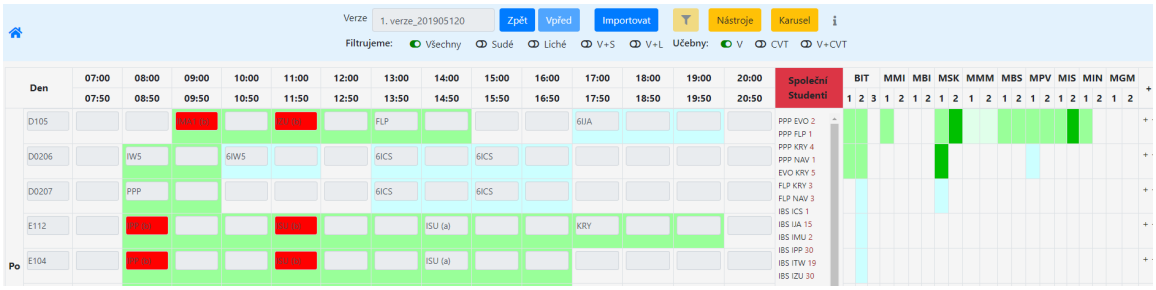


Obrázek 6.5: Ukázka okna pro plánování výuky s nástrojem zbývá naplánovat a oknem karuselu pro počty přednášek ve dni pro studenty.

Karusel pro výuku obsahuje okna pro graf předmětů, kolize s okolím, kolize ve dni a okno pro zátěž studentů ve dni. Výpisy kolizí byly oproti staré aplikaci vylepšeny. Jde v nich posouvat a je možné je po celcích skrývat. Například můžeme skrýt celek pro kolize se včerejším dnem. Na obrázku 6.5 můžeme vidět okno s počtem přednášek ve dni pro studenty. Tento údaj se vypočítává tak, že načteme nejdelší kolize podle předmětů ve dni a pak vytvoříme pole předmětů s jejich délkami. Při vytváření tohoto pole musíme dávat pozor, abychom ignorovali stejné předměty, které jsou ve stejný čas ale v jiné učebně. Poté vezmeme všechny n-tice a pole s předměty a počítáme. Pokud máme v poli předmět z n-tice přičteme jeho délku a posuneme ukazatel v poli podle jeho délky a takto pokračujeme déle. Zde nastává problém, pokud se 2 předměty z n-tice překrývají. Jeden je například od 10 a druhý od 11. V tomto případě bude studentovi přičtena délka prvního předmětu a počítáme, že na konec druhého předmětu již nepůjde. Dalším problémem jsou přednáškové skupiny. V 1 dni můžeme mít umístěn předmět pro obě přednáškové skupiny a nemáme šanci poznat, do které skupiny student patří, nebo ještě lépe, na kterou kvůli tranzitivním kolizím chodí. V tomto případě bereme skupinu, která je ve dni umístěna první. Proto je u tohoto nástroje umístěna poznámka, že se jedná pouze o orientační odhad a autorovi slouží pro odhadnutí přibližné zátěže pro studenty.

Načítání a zpracování nejdelších kolizí probíhá podle algoritmu uvedeného v kapitole 5.5. Jelikož se tento algoritmus provádí po každé změně buňky bylo jej nutné optimalizovat. První optimalizace se odehrává již v databázi zavedením klíčů a indexů nad požadovanými záznamy a nepoužíváním předzpracování dotazů. Pokud je počet načtených předmětu ve dni menší než 10, tak nám stačí načíst kolize délky, která je rovna počtu předmětů ve dni. Poslední optimalizací je zavedení postupného načítání studentů a vkládání výsledných počtů do externího pole. Pro každou n-tici potřebujeme načíst registrované studenty pouze 1. Do výsledného pole pak ukládáme jen ty n-tice, které mají počet studentů větší než 0. V PHP jsou pole implementována jako hashovací tabulky. Pokud bychom tedy v cyklu měnili údaje pole, které procházíme, ztráceli bychom výkon, protože by se neustále musely přepočítávat hashovací klíče. Výsledky měření výkonnosti budou uvedeny v kapitole 7.

V pravém dolním rohu aplikace je umístěna nápověda, kterou si uživatel může zobrazit. Najde v ní informace o syntaxi vstupů, barvách výuky a další funkcionalitě. Po najetí na informaci se mu navíc zobrazí stručný popis. Pokud chce uživatel sdílet svoji verzi s jiným uživatelem, stačí mu poslat adresu z adresového řádku. Druhému uživateli se zobrazí tento rozvrh v zobrazení pro návštěvníka s tlačítkem pro import. V tomto zobrazení není uživateli povoleno rozvrh upravovat. Může však využívat nástroje a krokovat historii. Toto zobrazení je možné vidět na obrázku 6.6. Nepřihlášený uživatel pak může rozvrh pouze zobrazit. Jedině administrátor může editovat rozvrh a to pouze v případě, že se jedná o rozvrh jiného administrátora.



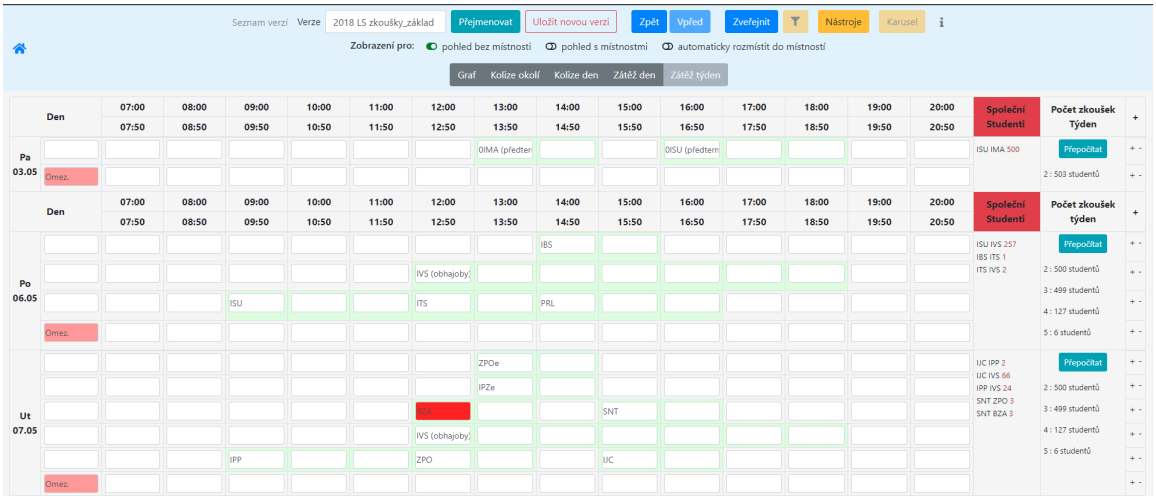
Obrázek 6.6: Ukázka okna pro plánování výuky v pohledu z pohledu jiného uživatele.

6.5 Okno pro tvorbu rozvrhu zkoušek

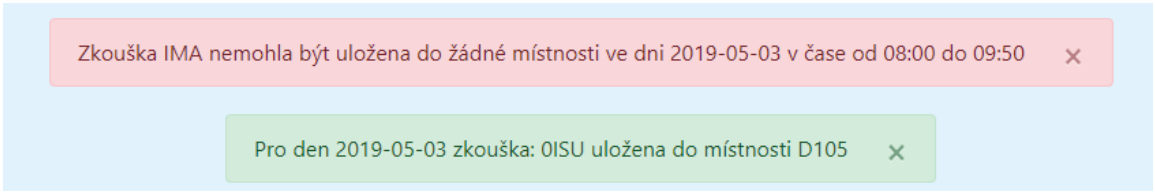
Na obrázku 6.7 můžeme vidět okno pro plánování zkoušek bez místností. Struktura je stejná jako u okna pro výuku. Zde místo filtrování pohledů pro liché, sudé týdny a CVT máme na výběr z pohledu bez místností, pohledu s místnostmi a pohledu s místnostmi s automatickým přidělením zkoušek do místností. Na obrázku 6.8 můžeme vidět příklad hlášení pro automatické vložení do místností. V karuselu přibylo okno pro zobrazení počtu zkoušek pro studenty v celém týdnu. Tento výpočet, stejně jako výpočet pro den, se provádí po změně okna, běží asynchronně, a po dokončení se automaticky zobrazí. Výpočet počtu pro týden může být časově náročnější a může například vypršet časový limit. Pro tento případ bylo do okna karuselu přidáno tlačítko pro přepočet kolizí v daném týdnu.

Pro zkoušky se trochu liší způsob uložení okna do databáze. Jak bylo řešeno, máme 2 pohledy bez a s místnostmi. Pokud jsme v pohledu bez místností, tak do databáze ukládáme údaj o okně zkoušky. Nejprve zkusíme v databázi najít, jestli jsme už v daný den neměli tuto zkoušku. Hledáme podle předmětu, termínu a času. Pokud jsme již měli, tak

zkopírujeme všechny místnosti tohoto starého okna do našeho nového okna. V pohledu s místnostmi totiž ukládáme do databáze pouze informaci o uložení do místností k danému oknu zkoušky. Při vložení do místnosti najdeme podle termínu předmětu a časového intervalu okno zkoušky, a tomu přidáme danou místnost. Pokud například v okně s místnostmi prodloužíme okno o hodinu, tato změna se promítne i do okna bez místností. Pokud v pohledu s místnostmi vložíme zkoušku mimo interval okna zkoušky, tak se vytvoří nové okno zkoušky. Tato synchronizace je nutná, aby nevznikaly nekonzistentnce mezi pohledem s a bez místností. Nesmí se stát, že v jednom pohledu máme naplánovánu zkoušku od 12 a v místnostech jí máme naplánovánu od 13.



Obrázek 6.7: Ukázka okna pro plánování zkoušek s pohledem bez místností a oknem karuselu pro kolize v týdnu.



Obrázek 6.8: Ukázka výpisu pro úspěch a neúspěch při automatickém rozdělování do místností.

Kapitola 7

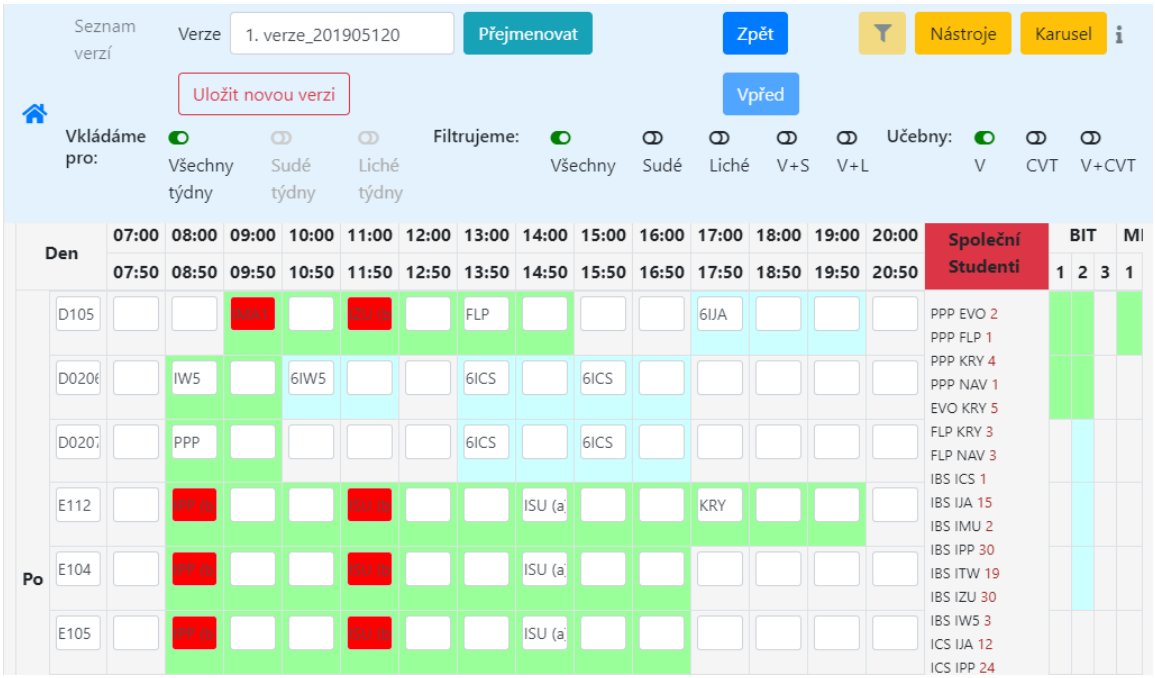
Testování

Aplikace prošla důkladným lokálním testováním. Nejprve byl každý modul otestován na základní funkčnost na testovacích datech. Testovaly se základní úkony, jestli jde vytvořit rozvrh, přejmenování verze, testoval se automat pro stavy verzí, testovalo se vkládání buněk, krokování historie na malém fiktivním rozvrhu atd. Po splnění těchto testů následovaly testy nad reálnými daty, kde se načetly staré rozvrhy a testovala se celková funkčnost. Posledním krokem této fáze byly testy s nevalidními vstupy.

Další fází bylo testování aplikace jako celku a testování kompatibility s aplikací pro import a export. Tato fáze již probíhala na školním serveru **pcknot4** za použití společné databáze pro oba programy. V tomto procesu byla aplikace otestována v reálném použití při plánování rozvrhu pro letní semestr akademického roku 2019. Při této příležitosti bylo odhaleno mnoho skrytých chyb, dořešeny některé problémy s kompatibilitou a navrhnuto mnoho vylepšení, která byla posléze implementována.

Byl také měřen čas potřebný pro výpočet kolizí ve dni a týdnu. Metodikou bylo provedení 10 měření v nejobsazenějším dni/týdnu rozvrhu výuky pro letní semestr roku 2019 a rozvrhu zkoušek zimního semestru roku 2018. Čas byl měřen od zahájení požadavku technologií AJAX do přijetí odpovědi od serveru. Testováno bylo na školním serveru **pcknot4**. Bylo provedeno měření při načtení stránky, kde se vypočítávají kolize pro všechny dny a týdny, a měření po změně buňky ve dni. Výsledné průměrné časy je možné vidět v tabulce 7.1. Byl testován i čas kontrol rozvrhu. Jelikož tyto kontroly běží na straně klienta, výsledný čas se může pro každého uživatele trochu lišit. Bude záležet na HW konfiguraci a na použitém prohlížeči a jeho verzi. Pro prohlížeč Google Chrome verze 74.0.3 jsem provedl testování na stolním počítači s 8 jádry a 16 GB RAM a notebooku se 4 jádry a 8 GB RAM. Časy kontrol se zde pohybovaly v průměru od 0,6 až do 1,2 sekundy.

Aplikace byla také otestována v prohlížeči Mozilla Firefox verze 66.05 a v prohlížeči Microsoft Edge. Testoval jsem také zobrazení a použitelnost aplikace na malých rozlišeních a mobilních zařízeních. I zde je díky použití knihovny Bootstrap aplikace použitelná, jak můžeme vidět na obrázku 7.1.



Obrázek 7.1: Ukázka zobrazení okna výuky na zařízeni s rozlišením 1024 na 445px.

Operace	doba trvání
Načtení dnu výuky	2,16 s
Změna ve dni výuky	0,7 s
Načtení týdnu zkoušek	6,27 s
Změna v týdnu zkoušek	4,38 s
Změna ve dni zkoušek	1,34 s

Tabulka 7.1: Tabulka s výslednými průměrnými časy doby výpočtu kolizí ve dni a týdnu.

Jako poslední byla testována výkonnost skriptu pro nalezení nejdelších kolizí předmětů. Pro letní a zimní semestr akademického roku 2019 se čas běhu pohyboval od 6 do 6,3 hodiny. Pro letní semestr akademického roku 2018 byl čas okolo 20 hodin. Toto bylo způsobeno faktem, že si studenti zapsali daleko více kombinací předmětů.

Kapitola 8

Závěr

V této práci jsem popsal obecné problémy při tvorbě rozvrhů. Vysvětlil jsem strukturu výuky na FIT a podrobně popsal problémy, které přináší. Popsal jsem detailní postup procesu tvorby rozvrhů na FIT a jednotlivé požadavky a kritéria, která musí být splněna. S ohledem na tyto poznatky jsem popsal problémy automatické tvorby rozvrhu a dostupných komerčních programů v podmínkách FIT. Vyhodnotil jsem vstupní data, popsal strukturu databáze a vytvořil návrh nové aplikace.

Výsledkem této práce je nová webová aplikace pro tvorbu rozvrhů, která uživateli dokáže v reálném čase zobrazovat informace o kolizích učitelů i studentů, informovat uživatele o porušení požadavků na výuku nebo zkoušku a díky použití karuselu dokáže uživateli přehledně zobrazit další důležité informace. Díky systému ukládání historie oken pak budou moci uživatelé krokovat své nebo oficiální rozvrhy a lépe tak pochopit jejich systém nebo vracet nechtěné změny. Aplikace je implementována jako informační systém se správou uživatelů a rozvrhů. Každý student se tedy bude moci registrovat, vytvořit nebo naimportovat rozvrh a zkusit si vytvořit vlastní lepší verzi. Pokud se mu to podaří, bude moci tuto verzi nabídnout autorům rozvrhu a čekat na zpětnou vazbu, kterou mu autoři poskytnou přímo v aplikaci. Nebo může jednoduše sdílet tuto verzi s ostatními uživateli. Aplikace také dokáže automaticky, oproti předchozím řešením, efektivně rozdělovat zkoušky do místností.

Podařilo se mi vytvořit intuitivní a interaktivní aplikaci, která dokáže autorům rozvrhu výrazně usnadnit práci. Podařilo se mi vyřešit problém s počítáním počtu přednášek a zkoušek ve dnech pro jednotlivé studenty, kdy po provedených optimalizacích je tento proces prováděn v reálném čase, a podařilo se mi vytvořit platformu pro kontrolu rozvrhu na straně klienta. Tato práce byla velice náročná a nepodařilo se mi z časových důvodů dokončit všechny cíle, které jsem si vytyčil na začátku. V budoucnu bych tedy v této práci chtěl pokračovat, soustředit se na další optimalizace, vylepšení současných kontrol a přidání dalších kontrol kvality rozvrhu.

Literatura

[1] Achour, M.; Betz, F.; Dovgal, A.; aj.: *Co je PHP?* [Online; navštíveno 06.01.2019]. URL http://php.net/manual/en/introduction.php

[2] Almeida, M. W. S.; Medeiros, J. P. S.; Oliveira, P. R.: *Solving the Academic Timetable Problem Thinking on Student Needs*. IEEE, 2015, ISBN 978-1-5090-0287-0.

[3] Bernard, B.: *Prezentační vzory z rodiny MVC*. [Online; navštíveno 18.05.2019]. URL https://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/

[4] Cheng, E.; Kruk, S.; Lipman, M.: Flow Formulations for the Student Scheduling Problem. 08 2002, s. 299–309, doi:10.1007/978-3-540-45157-0_20.

[5] Čillo, V.: *Program pro plánování rozvrhů*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2017.
URL http://www.fit.vutbr.cz/study/DP/DP.php?id=19244

[6] Duckett, J.: *JavaScript and JQuery: Interactive Front-End Web Development*. Wiley, 2014, ISBN 978-1118531648.

[7] Dytrych, J.; Veigend, P.: *Plán zkoušek na zimní semestr 2018/2019*. [Online; navštíveno 02.01.2019].
URL http://www.fit.vutbr.cz/study/advisor/20182019/zkouskyZS/.cs

[8] Holdener, A. T.: *Ajax: The Definitive Guide: Interactive Applications for the Web*. O'Reilly Media, 2008, ISBN 0596528388.

[9] Horký, A.: *Systém pro pokročilé plánování*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2015.
URL http://www.fit.vutbr.cz/study/DP/DP.php?id=17111

[10] Kubalcová, M.: *Porovnání programů pro plánování rozvrhů a zkoušek*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2012.
URL http://www.fit.vutbr.cz/study/DP/BP.php?id=13230

[11] Lockhart, J.: *Modern PHP: New Features and Good Practices*. O'Reilly Media, 2015, ISBN 1491905018.

[12] Ltd, S.: *Datatables manual*. [Online; navštíveno 17.05.2019]. URL https://datatables.net/manual/

[13] Pache, C.; Scholz, F.; Xue, F.; aj.: *Co je to JavaScript?* [Online; navštíveno 05.01.2019].

URL
https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

[14] Paulík, M.: *Optimalizátor rozvrhu zkoušek na FIT*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2015.
URL http://www.fit.vutbr.cz/study/DP/DP.php?id=15222

[15] Pejša, J.: *Co je Cross-Site Request Forgery a jak se mu bránit.* [Online; navštíveno 13.05.2019].
URL https://www.zdrojak.cz/clanky/co-je-cross-site-request-forgery-a-jak-se-branit/

[16] Polesný, O.: *Informační systém pro správu kurzů s automatickou tvorbou rozvrhů.* Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2013.
URL http://www.fit.vutbr.cz/study/DP/DP.php?id=14316

[17] Rahman, S. F.: *Jump Start Bootstrap: Get Up to Speed With Bootstrap in a Weekend.* SitePoint, 2014, ISBN 978-0992279431.

[18] Schwartz, B.: *MySQL profesionálně : optimalizace pro vysoký výkon.* Zoner Press, 2009, ISBN 978-80-7413-035-9.

[19] Tesarova, A.: *Aplikace na plánování rozvrhů, brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií, Ústav počítačové grafiky a multimédií.*

[20] Čangalović, M.; Schreuder, J. A.: Exact colouring algorithm for weighted graphs applied to timetabling problems with lectures of different lengths. [Online; navštíveno 18.05.2019].
URL
https://www.sciencedirect.com/science/article/pii/037722179190254S

[21] Čápka, D.: Popis MVC architektury. [Online; navštíveno 15.04.2019].
URL https://www.itnetwork.cz/php/mvc/objektovy-mvc-redakcni-system-v-php-popis-architektury