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. VLADIMÍR ČILLO**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAROSLAV DYTRYCH**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Čillo Vladimír, Bc.**
Obor: Bezpečnost informačních technologií
Téma: **Program pro plánování rozvrhů**
Timetable Planning Software
Kategorie: Uživatelská rozhraní

Pokyny:

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, který poskytne vhodnou podporu rozhodování při manuálním umisťování rozvrhových oken a průběžné automatické kontroly splnění požadavků. Předzpracování požadavků musí být provedeno tak, aby byl jeho výstup využitelný i pro algoritmus A. Horkého.
4. Implementujte navržené řešení.
5. Zhodnoťte dosažené výsledky a srovnejte s alternativními přístupy.

Literatura:

- Horký, Aleš: Systém pro pokročilé plánování, diplomová práce, Brno, FIT VUT v Brně, 2015
- Kubalcová, Monika: Porovnání programů pro plánování rozvrhů a zkoušek, bakalářská práce, Brno, FIT VUT v Brně, 2012

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2 a 3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Dytrych Jaroslav, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Táto práca sa zaoberá problematikou plánovania rozvrhov na *Fakultě informačních technologií Vysokého učení technického v Brně*. Cieľom práce je navrhnúť a implementovať novú aplikáciu pre podporu manuálnej tvorby rozvrhov, ktorá poskytne zlepšenie v provnaní so súčasným stavom. Vytvorená aplikácia má architektúru klient-server, pričom klient a server spolu komunikujú prostredníctvom rozhrania REST. Aplikácia poskytuje podporu pri práci s vstupnými dátami, ako aj pri analýze vytvorených rozvrhov a disponuje funkciami pre export dát vo formáte HTML.

Abstract

This work deals with timetabling problems at *Faculty of Information Technology of Brno University of Technology*. The aim of this thesis is to design and implement new application to support manual timetable planning, that will offer some innovations in comparison with current state. Implemented application is based on client-server architecture, at which client and server communicate by means of REST interface. Application offers functions for preprocessing of input data, as well as functions for analysis of created timetables. Data can be exported in HTML format.

Kľúčové slová

Plánovanie rozvrhov, Klient, Server, REST, rozvrh, výuka, skúšky

Keywords

Course timetabling, Client, Server, REST, timetable, lectures, exams

Citácia

ČILLO, Vladimír. *Program pro plánování rozvrhů*. Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Dytrych Jaroslav.

Program pro plánování rozvrhů

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Jaroslava Dytrycha. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Vladimír Čillo
24. mája 2017

Poďakovanie

Ďakujem vedúcemu tejto práce, Ing. Jaroslavovi Dytrychovi, za všetky poskytnuté informácie a odbornú pomoc pri riešení práce.

Obsah

Kapitola 1

Úvod

Tvorba rozvrhu je problém, s ktorým sa každoročne stretáva každá škola, či už základná, stredná alebo vysoká. Nie vždy musí byť tento problém zložitý. V prípade prvého stupňa základnej školy, kde má celá trieda rovnaký rozvrh a výuku zabezpečuje jeden vyčujúci v jednej učebni, sa naopak jedná o problém triviálny. Ako sa však posúvame k vyšším stupňom vzdelania, systém výuky sa stáva zložitejším a spolu sním aj proces plánovania rozvrhov.

Existuje niekoľko spôsobov, ktoré sa v praxi používajú pre tvorbu rozvrhov. Jednoduché varianty problému plánovania je možné vyriešiť bez pomoci počítača. Tento prístup je často preferovaný pred použitím počítača, pretože nevyžaduje špeciálny software, ani prípadné zaškolenie užívateľa na jeho použitie. Pre zložitejšie problémy tvorby rozvrhu však *manuálny* prístup nemusí byť optimálny z hľadiska času a vynaloženého úsilia.

Na stredných školách môže byť situácia komplikovaná používaním odborných učební, v ktorých prebieha výuka len určitých predmetov. Tieto predmety potom nemôžu byť v rozvrhu naplánované na prekrývajúce sa časy. Z pohľadu tvorby rozvrhu týmto spôsobom vznikajú rôzne obmedzenia. Niektoré z týchto obmedzení sú natoľko všeobecné, že sú priamo zahrnuté v dostupných programoch pre tvorbu rozvrhu. Použiteľnosť takéhoto programu v danej inštitúcií potom záleží najmä od toho, či umožňuje definovať, a pri tvorbe rozvrhu zohľadniť, všetky potrebné obmedzenia. Ak áno, rozvrh je potom možné pomocou daného programu zostaviť *automaticky*.

V prostredí vysokých škôl sú požiadavky jednotlivých fakúlt častokrát natoľko špecifické, že žiadny z existujúcich programov neponúka všetky vlastnosti a funkcie, ktoré sú potrebné pre tvorbu rozvrhu. Niekoľko štúdií ukázalo, že podobná situácia je aj na *Fakultě informačních technologií Vysokého učení technického v Brně*. V súčasnosti sa nachádzame v stave, kedy sa snažíme vytvoriť program, ktorý bude kompromisom medzi manuálnou a automatickou tvorbu rozvrhu. Pomocou tohto programu bude užívateľ vytvárať rozvrhy *manuálne, s asistenciou počítača*.

Cieľom tejto práce je návrh a implementácia software, s pomocou ktorého bude možné vytvárať rozvrhy v súlade s potrebami *Fakulty informačních technologií Vysokého učení technického v Brně* (FIT). Kapitola 2 pojednáva o špecifických potrebách FIT z hľadiska tvorby rozvrhov, záver kaptioly je venovaný popisu súčasného stavu. Na základe týchto informácií je v kapitole 3 predstavený návrh nového programu pre tvorbu rozvrhov na FIT. Kaptiola 4 obsahuje popis technológií použitých pri vývoji novej aplikácie, ktorej popisu sa následne venuje kapitola 5. Popis procesu testovania aplikácie je uvedený v kaptiole 6. Kapitola 7 obsahuje zhrnutie dosiahnutých výsledkov.

Kapitola 2

Plánovanie rozvrhov

Niekoľkokrát počas akademického roka rieši každá fakulta problém vytvárania rozvrhov. Plánujú sa dva druhy udalostí, pri ktorých sa študenti stretávajú s vyučujúcimi – výuka a skúšky. Plánovaním rozumieme vloženie daného počtu udalostí (prednášok, skúšok) do určitého počtu časových okien tak, aby boli splnené zadané podmienky. Zložitosť tejto úlohy samozrejme rastie s počtom študentov a predmetov, ktoré si študenti môžu zapísať. Aj pre relatívne nízke počty študentov a predmetov sa však jedná o náročnú úlohu, snahy v tejto oblasti preto smerujú k pokiaľ možno úplnej automatizácii plánovania.

V nasledujúcej časti tejto kapitoly je uvedená definícia všeobecného problému plánovania rozvrhov (angl. *timetabling problem*) spolu s pojmami používanými na jeho popis, ktorá predstavuje širší kontext problematiky, ktorej je venovaná táto práca. S pomocou uvedených pojmov potom v časti 2.2 popisujeme plánovanie výuky a v časti 2.3 plánovanie skúšok na *Fakulte informačních technologií Vysokého učení technického v Brně*.

2.1 Problém plánovania rozvrhov

Výskumu problému plánovania rozvrhov bolo v posledných desaťročiach venované nespočetné množstvo akademických publikácii, pričom vzniklo viacero jeho mierne odlišných formulácií. Definícia, ktorú uvádzame je považovaná za jednu z najobecnejších, ostatné definície je z nej možné získať jednoducho vynechaním niektorých vlastností a obmedzení [8].

Definícia problému

Sú dané nasledujúce množiny [15]:

- množina $\mathcal{E} = \{1, ..., E\}$ udalostí
- množina $\mathcal{T} = \{1, ..., T\}$ časových okien (časových slotov)
- množina $\mathcal{R} = \{1, ..., R\}$ miestností
- množina $\mathcal{S} = \{1, ..., S\}$ študentov
- množina $\mathcal{F} = \{1, ..., F\}$ vlastností miestností[1]
- množina $\mathcal{D} = \{1, ..., D\}$ dní

Platí, že každé časové okno patrí do práve jedného dňa z množiny $\mathcal{D}$, pričom každý deň pozostáva z T/D časových okien. Ďalej sú dané nasledujúce relácie:

- $\mathcal{M} \subseteq \mathcal{E} \times \mathcal{S}$, $(e, s) \in \mathcal{M} \Leftrightarrow$ študent s sa má zúčastniť udalosti e
- relácia dostupnosti $\mathcal{A} \subseteq \mathcal{E} \times \mathcal{T}$, $(e, t) \in \mathcal{A} \Leftrightarrow$ udalosť e môže byť naplánovaná do časového okna t
- relácia precedencie $\Pi \subseteq \mathcal{E} \times \mathcal{E}, (e_1, e_2) \in \Pi \Leftrightarrow$ udalosti e_1 a e_2 musia byť naplánované do časových okien t_1 a t_2 takých, že $t_1 < t_2$
- $\Phi_R \subseteq \mathcal{R} \times \mathcal{F}$, $(r, f) \in \Phi_R \Leftrightarrow$ miestnosť r má vlastnosť f
- $\Phi_E \subseteq \mathcal{E} \times \mathcal{F}$, $(e, f) \in \Phi_E \Leftrightarrow$ udalosť e vyžaduje vlastnosť f

Riešením problému je hľadný rozvrh. Formálne sa jedná o zobrazenie $g(e) : \mathcal{E} \to \mathcal{T} \times \mathcal{R}$, ktoré spĺňa nasledujúce obmedzenia [8]:

- H1: Každý študent môže mať v danom časovom okne najviac jednu udalosť
- H2: Udalosť musí byť naplánovaná do miestnosti, ktorá ma dostatočnú kapacitu
- H3: V danom časovom okne v danej miestnosti môže byť naplánovaná maximálne jedna udalosť, teda zobrazenie $g(e)$ musí byť injektívne.
- H4: Časový slot priradený udalosti musí byť dostupný pre túto udalosť podľa relácie $\mathcal{A}$: $g(e) \to (t, r) \Rightarrow (e, t) \in \mathcal{A}$
- H5: Zobrazenie musí spĺňať vlastnosti relácie Π
- H6: Miestnosť, do ktorej je naplánovaná udalosť, musí spĺňať požiadavky danej udalosti na vybavenie miestnosti: $g(e) \to (t, r) \Rightarrow \big((e, f) \in \Phi_E \Rightarrow (r, f) \in \Phi_R\big)$

Uvedené vlastnosti zobrazenia $g(e)$ označujeme pojmom **silné obmedzenia** (angl. *hard constraints*). Pre e udalostí, t časových okien a r miestností je veľkosť priestoru potenciálnych riešení (počet všetkých zobrazení $f(e) : \mathcal{E} \to \mathcal{T} \times \mathcal{R}$) daný vzťahom $(r \cdot t)^e$ [11]. Jedná sa o tak veľký počet kombinácií, že ani pri súčasnom výkone počítačov neprichádza do úvahy hľadať riešenie systematickým prehľadávaním stavového priestoru.

Silné obmedzenia rozdeľujú tento priestor na dve disjunktné množiny – množinu prípustných a množinu neprípustných riešení. Do množiny neprípustných riešení spadajú tie zobrazenia, ktoré nespĺňajú aspoň jedno z uvedených silných obmedzení a teda nie sú prakticky realizovateľné.

Našim cieľom je vybrať z množiny prípustných riešení to, ktoré má najnižšiu *cenu*. Cena riešenia je definovaná na základe **slabých obmedzení** (angl. *soft constraints*). Riešenia, ktoré porušujú veľké množstvo slabých obmedzení, považujeme za riešenia s vyskou cenou. Nájdenie optimálneho riešenia tejto úlohy je známy NP problém [9], rozsiahly prehľad optimalizačných metód je možné nájsť v literatúre [1].

Konkrétne definície slabých obmedzení neuvádzame, pretože rôzne aplikácie (plánovanie výuky, plánovanie skúšok) majú typicky rôzne požiadavky na výsledné riešenie a teda rôzne slabé obmedzenia. Zatiaľ čo uvedené silné obmedzenia sú nezávislé na konkretnej inštancii

[1] Táto množina modeluje vybavenie miestností, ako napríklad dostupnosť videozáznamu, projektoru, a podobne

problému, slabé obmedzenia môžu byť závislé na požiadavkách konkrétnej inštitúcie. Príklady používaných slabých obmedzení sú dostupné v literatúre [7, 1, 9], obmedzenia, ktoré kladieme na rozvrhy výuky na FIT sú diskutované v časti 2.2, obmedzenia rozvrhov skúšok v časti 2.3.

2.2 Plánovanie výuky na FIT

Fakulta informačních technologií Vysokého učení technického v Brně (ďalej len FIT) ponúka svojím študentom tri študijné programy – trojročný bakalársky, dvojročný magisterský a doktorský. Výuka v bakalárskom a magisterskom študijnom programe je zložená z troch skupín predmetov. Prvou skupinou, ktorá tvorí základ bakalárskeho a magisterského študijného programu, sú predmety *povinné*, ktoré je nutné absolvovať všetky. Druhou skupinou sú *povinne voliteľné* predmety. Tieto sú organizované do podskupín, pričom študent je povinný si z každej podskupiny vybrať a zapísať jeden prípadne viac predmetov. Poslednú skupinu tvoria *voliteľné* predmety, ktoré dávajú študentom možnosť zamerať sa na oblasti, ktoré ich zaujímajú. Skladba vyššie uvedených skupín predmetov je daná oborom študijného programu. Aktuálny prehľad akreditovaných študijných programov a ich oborov je možné nájsť na webových stránkach fakulty[2]. Doktorského študijného programu sa vyššie uvedené členenie netýka, ten pozostáva len z jednej skupiny predmetov, z ktorej si študent spoločne so svojím školiteľom predmety vyberá a stanovuje sa individuálny študijný plán.

Výuka v akademickom roku je rozdelená na zimný a letný semester, každý v trvaní 13 kalendárnych týždňov. Rozvrhom výuky je týždenný plán prednášok a cvičení, ktorý sa v danom semestri každý týždeň opakuje. Výnimkou z tohto pravidla sú cvičenia, ktoré často prebiehajú len v určitých týždňoch. Vyučovanie prebieha v čase od 7:00 do 20:50.

Keďže počet študentov, príjmaných do bakalárskeho študijného programu je, vzhľadom ku kapacitám učební, relatívne vysoký, študenti sú v prvom a druhom ročníku bakalárskeho štúdia rozdelení na dve skupiny. Každá prednáška v danom ročníku potom prebieha dvakrát, osobitne pre každú skupinu.

Informácie používané pri vytváraní rozvrhov

Vytváranie rozvrhov výuky je proces s veľkým množstvom vstupov, ktoré pochádzajú z rôznych zdrojov. Nasledujúci zoznam vznikol na základe konzultácií s Ing. Jaroslavom Dytrychom, ktorý je v súčasnosti na FIT zodpovedný za vytváranie rozvrhov:

- Základným vstupom je **zoznam udalostí**, ktoré je potrebné naplánovať. V podstate sa jedná o zoznam predmetov, ktoré sa budú v danom semestri vyučovať, pričom pre každý predmet musíme poznať dĺžku prednášky – typicky dve alebo tri hodiny – a prednášajúceho. Okrem prednášok priebiehajú v rámci výuky aj cvičenia, ktoré delíme na demoštračné, numerické, laboratórne a počítačové. Demonštračné cvičenia prebiehajú v prednáškových miestnostiach a z pohľadu plánovania je možné ich považovať za špeciálny typ prednášky. Ostatné typy cvičení v súčasnosti nespadajú do procesu centrálneho plánovania výuky, ale termíny cvičení si určujú jednotliví vyučujúci na základe obsadenosti potrebných miestností.

- Pri plánovaní máme k dispozícii obmedzený počet miestností. **Zoznam a kapacity miestností**, v ktorých sa môžu uskutočnovať prednášky, je uvedený v tabuľke 2.1.

[2] www.fit.vutbr.cz/study/programs

Miestnosť	Kapacita
A112, A113	64
D105	300
D206	154
D207	90
E104, E105	72
E112	156
G202	80

Tabuľka 2.1: Zoznam prednáškových miestností

Špecifickým rysom FIT z hľadiska plánovania rozvrhov je možnosť *streamovať* prednášku do viacerých miestností, čo znamená, že jeden vyučujúci efektívne prednáša vo viacerých miestnostiach naraz. Z technických dôvodov nie je možné takýmto spôsobom prepojiť ľubovoľnú kombináciu miestností – v súčasnosti podporované kombinácie zobrazuje tabuľka 2.2.

Kombinácia miestností	Výsledná kapacita
D105 + D206 + D207	554
D105 + D206	454
D105 + D207	390
D206 + D207	244
E112 + E104	228
E104 + E105	144
E112 + E104 + E105	300
A112 + A113 [3]	128

Tabuľka 2.2: Kombinácie miestností, ktoré je možné využiť pre streamovanie.

- Z hľadiska dodržania potrebnej kapacity učebne je užitočné poznať **počet študentov** každého predmetu. Zaujímavosťou je, že táto informácia nie je vždy dostupná. Napríklad v prípade predmetov vyučovaných v prvom ročníku bakalárskeho štúdia v dobe vytvárania rozvrhu – t.j. pred zápisom do štúdia, nemusí byť známy skutočný počet študentov.

- Jedným z hlavných problémov, s ktorými sa pri vytváraní rozvrhu stretávame, sú *kolízie*. Kolíziou v tomto prípade rozumieme situáciu, kedy dva predmety majú aspoň jedného spoločného študenta. **Tabuľka kolízií** pre každú dvojicu predmetov udáva počet spoločných študentov. Detailnejší popis vplyvu kolízií na výsledný rozvrh je uvedený v časti 2.2.

- Značné množstvo informácií, ktoré je potrebné zohľadniť, pochádza priamo od vyučujúcich, ktorý svoje požiadavky zadávajú prostredníctvom informačného systému fakulty. **Požiadavky od vyučujúcich** zahŕňajú napríklad preferovaný čas konania

[3] Táto varianta vyžaduje úpravu konfigurácie inštalovaného hardware, v prípade potreby je ale realizovateľná.

prednášky, prípadne naopak – čas, kedy danému vyučujúcemu nevyhovuje prednášať. Pokiaľ daný človek prednáša dva predmety, často sa stretávame s požiadavkom na umiestnenie daných predmetov do jedného dňa, prípadne dokonca do súvislého bloku, a podobne.

- Pri plánovaní ďalej berieme do úvahy **študijné plány** jednotlivých oborov, ktoré pre každý predmet definújú, či sa jedná o predmet povinný, povinne voliteľný alebo voliteľný. Táto informácia je dôležitá pri práci so silnými a slabými obmedzeniami, kedy napríklad kolíziu medzi voliteľnými predmetmi nepovažujeme za porušenie silného kritéria.

- Výuku niektorých predmetov[4] zabezpečujú zamestatnci FEKT[5], prípadne FSI[6]. Plánovanie výuky týchto predmetov spadá do kompetencií príslušných fakúlt a teda z pohľadu FIT sa jedná o **prednášky, ktoré majú presne daný čas**, ktorý je potrebné vo výslednom rozvrhu dodržať.

Zostavenie rozvrhu výuky z uvedených vstupných dát si vyžaduje zavedenie nejakého mechanizmu, ktorý daným plánovacím informáciam priradí priority a ktorý nám umožní systematicky hodnotiť pužiteľnosť a kvalitu vytváraného riešenia. Týmto mechanizmom sú silné a slabé obmedzenia (kritéria), ktorým sa budeme venovať v nasledujúcom texte.

Kritéria pre hodnotenie rozvrhu výuky

Silné kritérium predstavuje vlastnosť rozvrhu, ktorá musí byť splnená, aby bol tento rozvrh použiteľný v praxi. Rozvrhy, ktoré nespĺňajú niektoré zo silných kritérií, predstavujú akýsi medziprodukt procesu plánovania, z ktorého sa postupnými úpravami snažíme získať použiteľný rozvrh.

Nasledujúce definície silných kritérií vychádzajú z kritérií H1-H6 uvedených v časti 2.1, pričom v niektorých prípadoch došlo k ich spresneniu v súlade s potrebami FIT. Ak pre nietoré kritérium nie je na prvý pohľad zrejmý jeho význam, príslušné kritérium je uvedené spolu s popisom jeho praktického významu.

- **H1**: Bezkolíznosť prednášok z *povinných* a *povinne voliteľných* predmetov z pohľadu študenta a bezkolíznosť z pohľadu vyučujúcich. Keďže jeden vyčujúci môže prednášať viac predmetov, je potrebné dbať nato, aby tieto predmety neboli naplánované na prekrývajúce sa časy.

- **H2**: Prednáška musí byť naplánovaná do miestnosti, ktorá ma dostatočnú kapacitu. Dostatočnou kapacitou v tomto prípade rozumieme počet miest väčší alebo rovný 70% z celkového počtu študentov, ktorý majú zapísaný daný predmet.

- **H3**: V danom čase v danej miestnosti môže byť naplánovaná maximálne jedna prednáška.

- **H4**: Čas na ktorý je naplánovaná prednáška musí byť pre túto prednášku *dostupný*. Toto obmedzenie sa uplatní v prípade pevne stanovených časov prednášok, ktoré sú

[4] Zoznam týchto predmetov je možné nájsť na http://www.fit.vutbr.cz/study/course-l.php.cs zadaním skratky FEKT, resp. FSI, do poľa *Ústav*.

[5] Fakulta elektrotechniky a komunikačních technologií VUT v Brně

[6] Fakulta strojního inženýrství VUT v Brně

výsledkom plánovania výuky na fakúltách, ktoré výuku daného predmetu zabezpečujú. Časový slot pridelený danej prednáške potom považujeme za dostupný, všetky ostatné považujeme za nedostupné a teda nepoužiteľné pre danú prednášku. Analogicky postupujeme v prípade, kedy príslušný prednášajúci v určitú hodinu nemôže prednášať.

- **H5**: Dvojice prednášok, alebo prednášky a cvičenia, pre ktoré je vyžadované určité poradie (typicky chceme prednášku v danom týždni pred príslušným cvičením), musia byť naplánované tak, aby toto poradie bolo dodržané.

V porovnaní s kritériami platnými pre všeobecnú formuláciu problému plánovania bolo vynechané kritérium H6, ktoré hovorí, že prednáška musí byť naplánovaná do *vhodnej* miestnosti. Keďže vybavenie prednáškových miestností na FIT je v zásade rovnaké, toto kritérium je možné považovať za vždy splnené a teda z pohľadu FIT nemá praktický význam.

Všeobecná formulácia problému plánovania ale naopak nepočíta s niektorými obmedzeniami, ktoré vyplývaju zo systému výuky na FIT. Pri vytváraní rozvrhu musíme počítať aj s nasledujúcimi silnými kritériami:

- **H6**: Dostatok času na presun študentov medzi fakultami. Výuka niektorých predmetov prebieha na FEKT, takýto predmet samozrejme nemôžeme naplánovať do nasledujúceho časového slotu za predmetom vyučovaným na FIT, pretože by sa študenti nestihli presunúť z areálu FIT do areálu FEKT, prípadne naopak.
- **H7**: V prípade použitia streamingu využívať len dovolené kombinácie učební (tabuľka 2.2).
- **H8**: Predmety, ktoré majú viac prednášok v jednom týždni ich musia mať v rôzne dni.

Pokiaľ sa nám podarí nájsť riešenie, ktoré spĺňa uvedené kritéria H1 - H8, získali sme rozvrh, ktorý určite bude implementovateľný v praxi. Silné kritéria však nehovoria nič o tom, do akej miery bude študentom a akademickým pracovníkom pohodlné sa takýmto rozvrhom riadiť. Inak povedané, máme k dispozícii mechanizmus, ktorým sme schopný určiť, že sa jedná o pužiteľný rozvrh, ale nie sme schopný povedať nič o kvalite tohto rozvrhu. Pri plánovaní výuky na FIT preto ďalej zohľadňujeme nasledujúce **slabé kritéria**:

- **S1**: Bezkoliznosť jednotlivých prednášok. Toto obmedzenie je podobné ako H1, v tomto prípade ale zohľadňujeme všetky typy predmetov – povinné (P), povinne voliteľné (PV) a voliteľné (V). Kolízie P-P a P-PV rieši kritérium H1, akékoľvek ďalšie kolízie sú teoreticky prípustné, ale samozrejme nežiadúce a snažíme sa ich minimalizovať. V prípade prednášok, ktoré sa opakujú – dve skupiny v jednom týždni – stačí bezkoliznosť s jednou z nich.
- **S2**: Minimalizovať počet dlhých blokov výuky. Blokom v tomto prípade rozumieme viacero prednášok, ktoré nasledujú bezprostredne za sebou. Príliš dlhé bloky prednášok vedú k tomu, že študenti sú unavení a nesústredení. Za hornú hranicu prijateľnej dĺžky bloku je možné považovať 6 hodín (2×3h). Akýkoľvek dlhší blok prednášok budeme považovať za porušenie tohto kritéria.
- **S3**: Obedová pauza. Pri plánovaní výuky je potrebné myslieť nato, že ako študenti, tak vyučujúci, budú v čase obeda chcieť ísť na obed a teda rozvrh by im to mal umožnovať.

- **S3**: Koncentrácia výuky len do určitých dní. Z pohľadu študenta, ktorý má napríklad 5 prednášok týždenne, je určite lepší rozvrh, ktorý mu týchto 5 prednášok rozdelí do dvoch dní (napr. pondelok a utorok), ako rozvrh, v ktorom bude mať na každý deň naplánovanú jednu prednášku. Pri plánovaní sa v rámci možností snažíme vziať túto skutočnosť do úvahy.
- **S4**: Preferencie vyučujúcich. Ako bolo uvedené v časti 2.2, vyučujúci majú možnosť zvoliť si preferovaný čas, prípadne deň ich prednášky. Z pohľadu vyučujúcich bude preto ako *dobrý* hodnotený taký rozvrh, ktorý tieto preferencie spĺňa.
- **S5**: Nezačínať výuku o siedmej ráno. Plánovanie prednášok na siedmu hodinu je v zásade možné, nutíme tým ale študentov, a samozrejme aj vyučujúcich, k skorému vstávaniu, ktoré z pochopiteľných dôvodov nie je obľúbené.
- **S6**: Pri plánovaní nepoužívať piatok. Veľká časť študentov FIT cez víkend odchádza z Brna, táto skupina preto určité ocení, keď v piatok nebude mať žiadnu výuku. Maximálne negatívne je z pohľadu študentov hodnotená výuka naplánovaná na piatok večer.
- **S7**: Nekončiť príliš neskoro. Výuku je možné plánovať až do 20:50, v praxi ale ani študenti, ani vyučujúci pravdepodobne nebudú spokojný s rozvrhom, na základe ktorého by mali byť takto neskoro večer v škole. Akákoľvek výuka po 19:00 je považovaná za porušenie tohto kritéria.

S pomocou uvedených slabých kritérií sme schopný objektívne ohodnotiť kvalitu rozvrhu a jednotlivé rozvrhy medzi sebou porovnávať. To nám pomáha vytvoriť rozvrh, s ktorým bude spokojná veľká časť študentov a vyučujúcich, ktorý sa ním budú riadiť.

Kritérií je však relatívne veľa a sú často protichodné. Na jednej strane sa napríklad snažíme vyhnúť príliš dlhým blokom súvislej výuky, na druhej strane sa ale snažíme, aby výuka nebola rozdelená do zbytočne veľkého počtu dní. Podbným príkladom je snaha o vyhýbanie sa výuke skoro ráno, prípadne neskoro večer. Niektorí vyučujúci si však sami o takéto časy požiadajú. Vyváranie rozvrhu výuky je vo svojej podstate teda hľadaním akéhosi kompromisu s ohľadom na všetky zúčastnené strany. Nie je preto prekvapením, že takmer vždy sa nájde niekto, kto výsledný rozvrh výuky hodnotí negatívne.

2.3 Plánovanie skúšok na FIT

Študijné výsledky sa overujú priebežnou kontrolou štúdia, ktorá najčastejšie prebieha formou polsemestrálnej skúšky, prípadne projektov, pri ktorých študenti demonštrujú schopnosť prakticky aplikovať získané znalosti. Súčasťou niektorých predmetov sú aj cvičenia, pri ktorých sa bodovo hodnotí aktívna účasť študenta. Po skončení 13 týždňového obdobia výuky nasleduje skúškové obdobie, ktoré trvá spravidla 5 týždňov.

Podľa spôsobu zakončenia je možné rozdeliť predmety na zakončené zápočtom, skúškou alebo zápočtom a skúškou. Podmienky skladania skúšok definuje *Studijní a zkušební řád Vysokého učení technického v Brně* [21], ktorý dopĺňa *Směrnice děkana FIT doplňující Studijní a zkušební řád VUT v Brně* [20]. Z pohľadu tejto práce sú dôležité nasledujúce pravidlá:

- Výsledok predmetu je po zložení skúšky klasifikovaný stupňom A-F v súlade s klasifikačnou stupnicou ECTS. Pri klasifikácii stupňom F má študent právo na opravný termín, pričom opravné termíny sú maximálne **dva**.

- Pokiaľ študent ani po troch pokusoch neabsolvuje daný *povinný* predmet, musí si ho v nasledujúcom akademickom roku zapísať znova. Ak sa jedná o povinný predmet, ktorý má študent zapísaný po druhý krát, je mu štúdium ukončené podľa § 56 odst. 1 písm. b) Zákona o vysokých školách č. 111/1998.

- Predchádzajúce pravidlo sa vzťahuje aj na predmety *povinne voliteľné*, pričom v prípade neabsolvovania povinne voliteľného predmetu je možné si vybrať ľubovoľný predmet z príslušnej skupiny predmetov.

Obdobie skúškového v dĺžke 5 týždňov je zvyčajne logicky rozdelené na prelínajúce sa 3 fázy. Počas prvej fázy, ktorá spravidla končí v treťom týždni, prebiehajú *riadne termíny* všetkých skúšok. V druhej fáze, ktorá typicky začína v treťom a končí v štvrtom týždni, prebiehajú *prvé opravné* termíny. V poslednej fáze, ktorá začína v štvrtom a končí v piatom týždni skúškového obdobia, prebiahjú *druhé opravné termíny*. Toto rozdelenie vychádza z toho, že počet študentov, ktorý sa zúčastňujú riadnych termínov, je oveľa väčší ako počet tých, ktorý sa zúčastňujú prvých opravných termínov. Rovnaký vzťah platí pre prvé a druhé opravné termíny. Najviac času je preto vyhradeného pre riadne termíny, kde sa snažíme vytvoriť medzi skúškami čo najväčšie rozostupy. Najmenej času vychádza na druhé opravné termíny, ktorých sa ale zároveň zúčastňuje najmenej študentov. V poslednom týždni teda dochádza k najväčšiemu počtu kolízií, tieto by sa ale mali týkať iba malej časti študentov (tých, ktorý majú dva a viac druhých opravných termínov).

Nasledujúci text pojednáva o informáciach, z ktorých na FIT vychádzame pri tvorbe rozvrhu skúšok, a o kritériach, ktorými hodnotíme použiteľnosť a kvalitu výsledného rozvrhu.

Informácie používané pri vytváraní rozvrhu skúšok

Informácie, ktoré vstupujú do procesu plánovania skúšok, sú analógiou informácií používaných pri plánovaní rozvrhov, uvedených v sekcii 2.2, pričom naviac pracujeme s *tabuľkov termínov*:

- **Zoznam udalostí** je v tomto prípade zoznam všetkých predmetov, pre ktoré potrebujeme naplánovať skúšku.

- **Zoznam miestností** v ktorých sa jednotlivé skúšky môžu konať je totožný so zoznamom miestností dostupných pre výuku (tabuľka 2.1).

- **Počet študentov** každého predmetu pri plánovaní skúšok používame pre odhad počtu študentov, ktorý sa reálne zúčastnia skúšky. Tento počet je typicky nižší (niektorí študenti nesplnia podmienky získania zápočtu, a teda nie sú pripustení ku skúške, prípadne sa na prvý termín nestihnú naučiť apod.)

- **Tabuľka kolízií** pre každú dvojicu predmetov udáva počet spoločných študentov.

- **Požiadavky od vyučujúcich** zásadným spôsobom ovplyvňujú plánovanie skúšok a sú podrobnejšie rozobraté v nasledujúcom texte.

- **Študijné plány** jednotlivých oborov rozdeľujú predmety na povinné, povinne voliteľné a voliteľné.

- **Pevne dané termíny** niektorých skúšok sú výsledkom plánovania skúšok na fakultách, ktoré zabezpečujú výuku daných predmetov.

- **Tabuľka termínov** pre určitý rok obsahuje informácie o počte študentov, ktorý sa zúčastnili jednotlivých termínov skúšok. Túto informáciu pri plánovaní používame pre odhad potrebnej kapacity pre určitý termín skúšky.

Podobne, ako pri plánovaní výuky, aj pri vytváraní rozvrhu skúšok majú vyučujúci možnosť zadať svoje požiadavky. V rámci týchto požiadaviek môžu zadať preferovaný dátum a čas skúšky, prípadne rozsah dátumov alebo časov, kedy by im najviac daná skúška vyhovovala. Ďalej je tu možnosť zadať časy, kedy nemôžu skúšať. Často sa objavuje požiadavka na určitý minimálny rozostup riadneho a opravného termínu, aby vyučujúci mali dostatok času na opravenie písomiek. Okrem toho vyučujúci definujú svoje požiadavky na priebeh skúšky, ktoré zahŕňajú:

- Rozsadenie študentov – pri skúške môžu študenti sedieť vedľa seba, vyčujúci môže ale vyžadovať, aby medzi študentami bolo jedno, prípadne dve voľné miesta. Táto požiadavka zvyšuje nároky na potrebnú kapacitu učebne, v ktorej sa skúška koná.
- Počet kôl – štandardne skúška prebieha v jednom kole, to znamená, že všetci zúčastnený študenti skúšku píšu naraz. Vyučujúci môže požadovať dve kolá, kedy sa študenti rozdelia na dve skupiny, pričom druha skupina začne písať až po tom, ako prvá skupina skončí a opustí miestnosť. Tým sa zdvojnásobí čas potrebný na vykonanie skúšky, ale je možné skúšku naplánovať do menšej miestnosti.
- Doba trvania skúšky – čas na vypracovanie skúšky je pre rôzne predmety rôzny. K čistému času, ktorý majú študenti k dispozícii na vypracovanie skúšky, je ďalej potrebné pripočítať čas na usadenie študentov, rozdanie zadaní a podobne. Celkový potrebný čas zadá vyučujúci ako jeden zo svojich požiadavkov.

Pri zostavovaní rozvrhu skúšok z uvedených informácií sa, rovnako ako pri plánovaní výuky, nezaobídeme bez silných a slabých kritérií. Použiteľnosť rozvrhu v praxi je podmienená splnením silných kritérii. Slabé kritéria predstavujú prostriedok na hodnotenie kvality rozvrhu. Ďalej je uvedený popis jednotlivých kritérií platných pre rozvrhy skúšok na FIT.

Kritéria pre hodnotenie rozvrhu skúšok

Pri plánovaní skúšok sa stretávame s rôznymi typmi kolízií. Pred zavedením konkrétnych kitérií je potrebné zaviesť pojmy k popisu jednotlivých typov kolízií, ktoré nastávaju v rozvrhu skúšok. Zatiaľ čo pri plánovaní výuky kolízia nastáva iba v prípade, keď sú dve prednášky naplánované na prekrývajúce sa časy, pri plánovaní skúšok považujeme za kolíziu aj situáciu, kedy sú dve skúšky naplánované s *nedostatočným* rozostupom. Na základe času, ktorý uplynie od začiatku jednej do začiatku druhej skúšky, môžu medzi danými dvomi skúškami vzniknúť tieto typy kolízií:

- *hodinová kolízia*, ak sú skúšky naplánované na prekrývajúce sa časy
- *denná kolízia*, ak sú skúšky naplánované na rovnaký deň
- *susedná kolízia*, ak sú skúšky naplánované na dva po sebe idúce dni
- *kolízia ob 1 deň*, ak sú skúšky naplánované na dva rôzne dni, medzi ktorými je práve jeden voľný deň

Tieto typy kolízií vznikajú z pohľadu študentov z dôvodu času potrebného na prípravu na jednotlivé skúšky, pričom nie každý typ kolízie je považovaný za silné kritérium. Pre rozvrh skúšok používame nasledujúce **silné** kritéria:

- **H1**: Rozvrh nesmie obsahovať *hodinové kolízie*.
- **H2**: Každá skúška musí byť naplánovaná do miestnosti s dostatočnou kapacitou (vzhľadom na požadované rozsadenie)
- **H3**: Maximálne jedna skúška v jednej miestnosti naraz (výnimkou môže byť prípad, kedy si vyučujúci vyžiada spojiť dve skúšky).
- **H4**: Dodržať požiadavky vyučujúcich na:
  - počet kôl
  - požadovanú dobu skúšky
  - vyučujúci nesmie mať naplánovanú skúšku v čase, kedy nemôže skúšať
- **H5**: Dodržať termín pevne daných skúšok plánovaných na iných fakultách.
- **H6**: Dostatok času pre vyučujúcich na opravenie písomiek medzi jednotlivými termínmi.
- **H7**: Bezkolíznosť skúšok z pohľadu vyučujúcich (maximálne jedna skúška v danom čase).

Pri rozvrhoch, ktoré spĺňajú všetky z uvedených silných kritérií má zmysel ďalej skúmať, do akej miery spĺňajú nasledujúce **slabé kritéria**:

- **S1**: Rozvrh nesmie obsahovať denné a susedné kolízie, ani kolízie ob 1 deň.
- **S2**: Využívanie čo najmenšieho počtu miestností (preferovať radšej jednu veľkú ako dve menšie).
- **S3**: Skúška nesmie byť naplánovaná pred 9:00 v pondelok a po 16:00 v piatok (kôli dochádzajúcim študentom).
- **S4**: Medzi ľubovoľnou dvojicou skúšok z povinných predmetov sú aspoň dva voľné dni.
- **S5**: Rozvrh vyhovuje preferenciam vyučujúcich (spĺňa aj tie, ktoré neboli uvedené v zozname tvrdých kritérií).

Pri vytváraní rozvrhu skúšok sa snažíme počet porušení uvedených slabých kritérií minimalizovať, vzhľadom k veľkému počtu vstupov a obmedzení sa ale väčšinou nepodarí nájsť rozvrh, ktorý by neporušil žiadne slabé kritérium. Tento stav je spôsobený aj tým, že študenti nie sú pri výbere voliteľných predmetov z hľadiska kombinácií ničím obmedzovaní, a teda je pravdepodobné že sa nájde študent alebo skupina študentov, ktorá má skúšky naplánované s relatívne malými rozostupmi.

2.4 Zhrnutie súčasného stavu

Predchádzajúce časti tejto kapitoly boli venované popisu plánovacích informácií a kritériám, ktoré nám z nich pomáhajú zostaviť použiteľný a kvalitný rozvrh. Samotný proces plánovania potom môže prebiehať buď manuálne, automaticky, alebo kombináciou obidvoch prístupov.

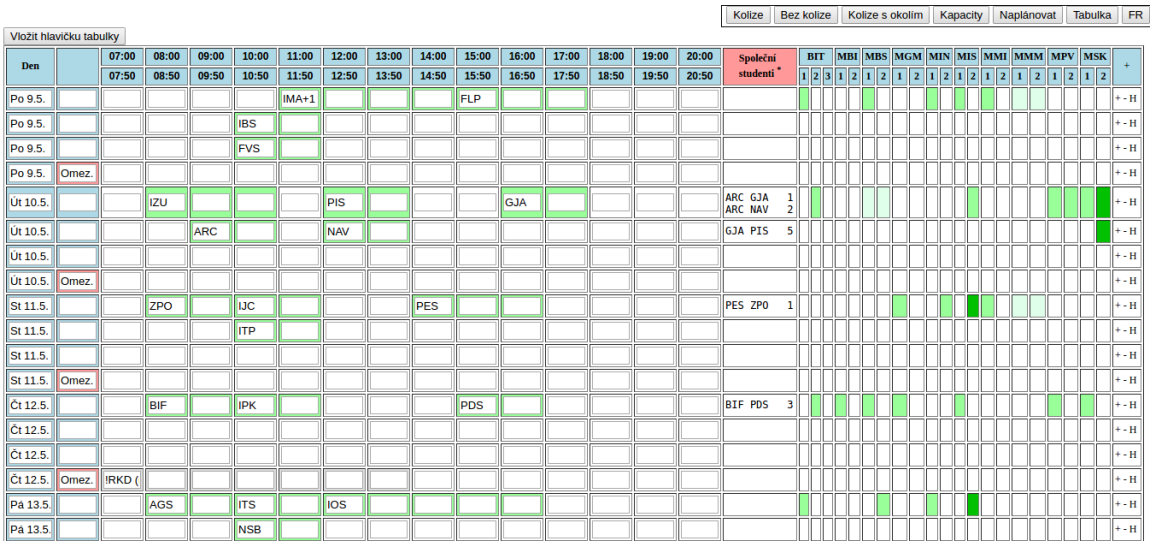
Manuálne plánovanie

Pri tejto metóde za pomoci papieru, ceruzky a gumy postupne skladáme a kombinujeme predmety do výsledného rozvrhu a snažíme sa nájsť najlepšie možné riešenie. Po každom vložení novej prednášky, prípadne skúšky do rozvrhu je nutné ručne skontrolovať na základe tabuľky kolízií potenciálne novovzniknuté kolízie u študentov, vyučujúcich a miestností. Osoba ktorá využíva pri plánovaní túto metódu, musí veľmi dobre poznať všetky silné a slabé kritéria, ktoré sú kladené na výsledný rozvrh. Nevýhodou tejto metódy je, že je časovo náročná a po vytvorení rozvrhu nemáme k dispozíci mnoho spôsobov, ako objektívne ohodnotiť kvalitu vytvoreného rozvrhu [11].

Je zrejmé, že použiteľnosť tejto metódy klesá s veľkosťou daného plánovacieho problému. Napriek tomu bola táto metóda k tvorbe rozvrhov počas niekoľkých rokov na FIT reálne používaná. V súčasnosti ju ale nahradila metóda, ktorá kombinuje manuálny prístup s asistenciou počítača.

Plánovanie s asistenciou počítača

Pri tejto metóde je rozvrh stále vytváraný ručne, postupným pridávaním jednotlivých predmetov do výsledného rozvrhu. Tento proces ale prebieha v rámci nejakého programu, ktorý určité činnosti automatizuje. Daný program môže napríklad automaticky kontrolovať jednotlivé kritéria a upozorňovať užívateľa na ich porušenie. Toto je v súčasnosti využívaný prístup k tvorbe rozvrhov skúšok a výuky na FIT.



Obr. 2.1: Aplikácia, ktorá je v súčasnosti používaná na tvorbu rozvrhov pre FIT.

Aplikácia, ktorá je na FIT používaná na tvorbu rozvrhov, je napísana v jazyku *JavaScript* a užívateľ s ňou pracuje vo webovom prehliadači. Program poskytuje virtuálny rozvrh,

do ktorého sa postupne vkladajú jednotlivé predmety (obr. 2.1). Aplikácia disponuje sadou nástrojov invokovaných pomocou tlačidiel v jej pravej hornej časti. Popis týchto nástrojov je uvedený tabuľke 2.3.

Tlačidlo	Funkcia
`Kolize`	Umožní zadať skratky dvoch predmetov, pre ktoré následne zobrazí počet spoločných študentov.
`Bez kolize`	Po zadaní skratky predmetu vypíše zoznam predmetov, ktoré s daným predmetom nemajú žiadnych spoločných študentov.
`Kolize s okolím`	Zobrazí zoznam spoločných študentov pre každu dvojicu predmetov, ktorá je naplánovaná v rozsahu dvoch dní pred a dvoch dní po aktuálnom dni (aktuálny deň je určený pozíciou kurzora v tabuľke).
`Kapacity`	Po zadaní skratky predmetu aplikácia vypíše počet študentov pre príslušný termín vybraný v zozname hned vedľa skratky predmetu. Zaškrtávaním jednotlivých miestností sa hľadá kombinácia miestností, ktorá poskytne dostatočnú kapacitu (aj s prípadním rozsadením pre potreby skúšok) pre daný predmet. Príklad práce s kapacitami je znázornený na obr. 2.2.
`Naplánovat`	Zobrazí resp. skryje zoznam predmetov, ktoré ešte nie sú naplánované.
`Tabulka`	Zobrazí nástroje pre vkladanie nových riadkov do tabuľky.
`FR`	Umožňuje vyfiltrovať v rozvrhu len predmety pre určitý ročník.

Tabuľka 2.3: Popis funkcií dostupných prostredníctvom menu v hornej časti aplikácie

Po vložení jednotlivých predmetov do rozvrhu a kontrole kolízií je možné výberom položky *Tabulka* → *Doplniť řádky a místnosti* pristúpiť k prideleniu miestností daným prednáškam, resp. skúškam. Aplikácia pre každú miestnosť zobrazí jeden riadok, do ktorého je možné vložiť daný predmet, čím pridelíme danému predmetu túto miestnosť. Pri tomto spôsobe plánovania je implicitne zaručené splnenie silného kritéria H3, ktoré dovoľuje maximálne jednu udalosť v danej miestnosti naraz. Splnenie ostatných kritérií musí kontrolovať sám užívateľ.

V porovnaní s manuálnym prístupom k plánovaniu poskytuje táto aplikácia užívateľovi veľkú podporu, vytváranie rozvrhov si ale stále vyžaduje značné množstvo práce a znalostí zo strany užívateľa.

Automatizované plánovanie

Za plne automatizované plánovanie považujeme stav, kedy rozvrh na základe silných a slabých kritérií, po zadaní potrebných vstupných dát, autonómne zostaví počítačový program. Analýzou existujúcich programov pre plánovanie sa zaoberala bakalárska práca z roku 2012 [14]. Záverom tejto štúdie bolo, že žiadny z dostupných programov nie je pre potreby FIT vhodný.

Obr. 2.2: Ukážka práce s kapacitami učební. V tomto prípade máme zvolené učebne `D105` a `D206`. Posledné tri textové polia zobrazujú celkovú kapacitu vybraných učební pre prípad, kedy študenti sedia vedľa seba (454), ob 1 (227), alebo ob 2 miesta (171). Červená farba textového poľa indikuje, že zvolená kombinácia učební nie je dostatočná pre vybraný predmet (TIN) a dané rozsadenie.

O zavedenie automatického plánovania na FIT sa v rámci svojej diplomovej práce pokúsil Aleš Horký [11]. Na generovanie rozvrhov vo svojej práci navrhol použitie kombinácie genetického a heuristického algoritmu. Pri praktickom použití programu, ktorý vznikol v rámci tejto práce, nastáva niekoľko problémov.

Prvým problémom je jeho komplikovaná konfigurácia. Program pri spustení vyžaduje konfiguračný súbor vo formáte *XML*, ktorý obsahuje kompletný popis vstupného problému plánovania, vrátane požiadavkov vyčujúcich. Veľkosť tohto súboru sa pre praktické problémy pohybuje okolo 3000 riadkov. Práca, ktorá je potrebná na jeho vytvorenie je porovnateľná s plánovaním rozvrhu pomocou aplikácie popísanej v predchádzajúcej časti.

Po zadaní vstupov program niekoľko hodín generuje hľadaný rozvrh. Pokiaľ vo výslednom rozvrhu potrebujeme urobiť zmenu (napríklad z dôvodu, že rozvrh nespĺňa niektoré zo slabých kritérií, ktoré je dôležité), takáto zmena je náročná, pretože užívateľ nepozná vzťahy a súvislosti medzi jednotlivými predmetmi v rozvrhu, keďže rozvrh bol generovaný automaticky.

V úvode tejto kapitoly bola uvedená definícia všeobecného problému plánovania rozvrhov *(angl. timetabling problem)*, spolu so základnou terminológiou používanou v tejto oblasti. Silné kritéria (*hard constraints*) všeobecného problému plánovania boli pre potreby FIT doplnené o niekoľko ďalších obmedzení. Zároveň sme uvideli popis jednotlivých slabých kritérií (*soft constraints*) rozvrhu výuky (2.2) a rozvrhu skúšok (2.3), ktoré sú platné v podmienkach FIT. V závere kapitoly bola stručne popísaná aktuálna situácia v oblasti plánovania na FIT. Súčasný stav plánovania výuky a skúšok nie je možné považovať za uspokojivý, pretože používaný software obsahuje niekoľko praktických nedostatkov, ktoré poskytujú priestor pre ďalší vývoj a vylepšenia.

Kapitola 3

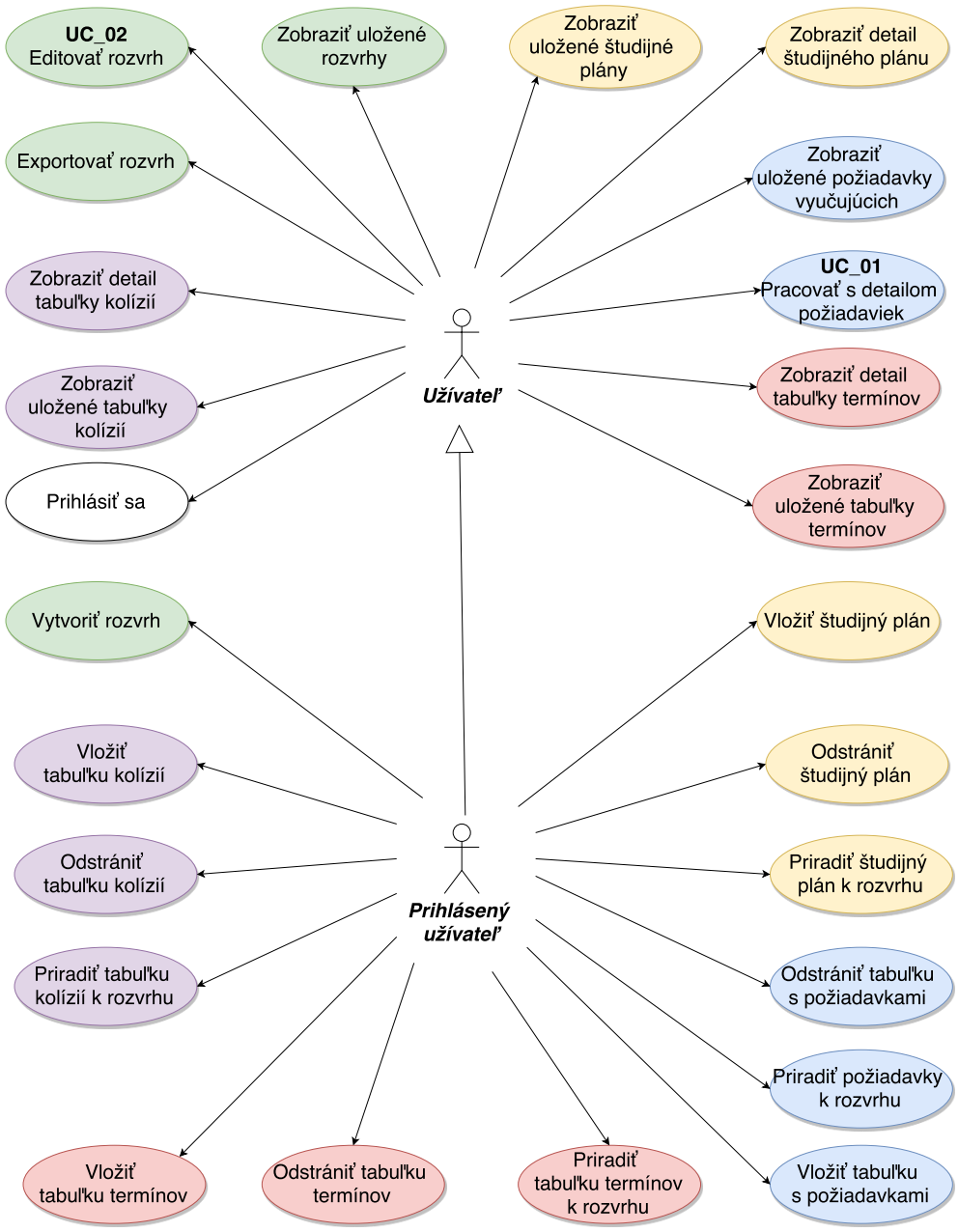
Návrh aplikácie

Na základe informácií uvedených v predchádzajúcej kapitole predstavujeme v tejto kapitole návrh software, ktorý poskytne podporu pri manuálnom vytváraní rozvrhu. Nasledujúci text je rozdelený na dve časti – sekcia 3.1 obsahuje popis navrhovanej funkcionality a sekcia 3.2 popis navrhovanej architektúry aplikácie.

3.1 Navrhovaná funkcionalita

Z analýzy procesu tvorby rozvrhov na FIT vyplynulo, že značnú časť práce je potrebné venovať príprave vstupných dát (vstupnými dátami rozumieme požiadavky vyučujúcich, tabuľky kolízií, tabuľky termínov a študijné plány). Do novej aplikácie preto bude zakomponovaná podpora práce s týmito typmi dát. Pre uvedené typy vstupných dát bude nová aplikácia podporovať ich import, export a priradenie k určitému rozvrhu. Samozrejmosťou u všetkých typov vstupných dát je zobrazenie prehľadu uložených položiek a odstránenie vybranej položky. Dostupné funkcie pre prácu so vstupnými dátami, spolu so základnými operáciami s rozvrhmi, sumarizuje diagram prípadov použitia na obrázku 3.1.

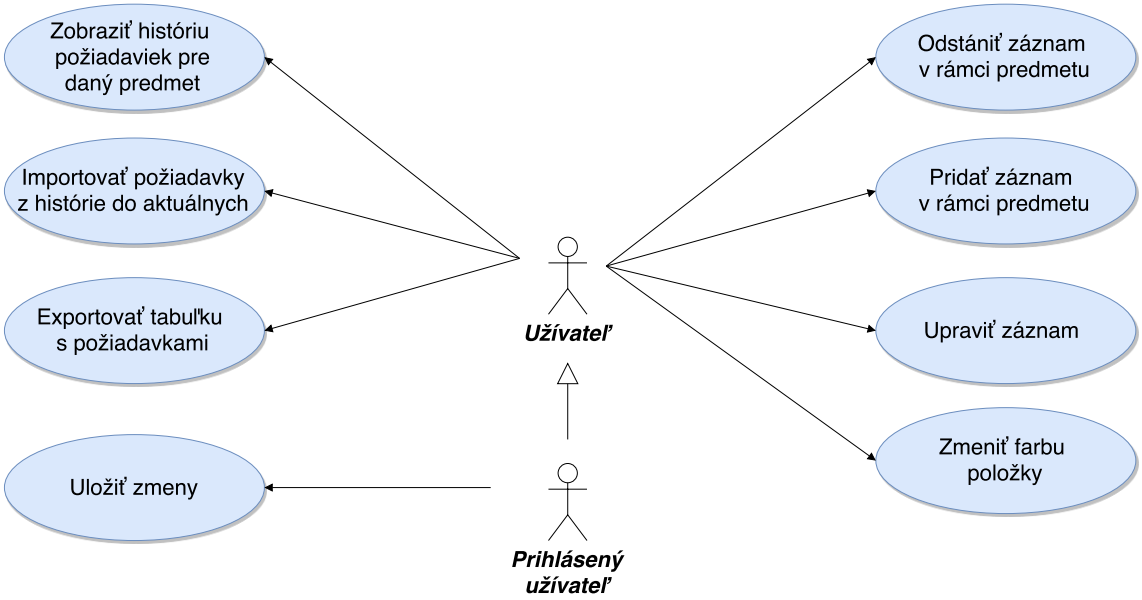
Špeciálna pozornosť je venovaná požiadavkam vyučujúcich. Zobrazenie zoznamu požiadaviek vyučujúcich formou tabuľky bude disponovať možnosťou pridania novej položky alebo odstránenia existujúcej položky. Okrem toho je potrebné mať možnosť zafarbiť v tabuľke s požiadavkami vybranú bunku, alebo riadok, a to z viacerých dôvodov. Pri vytváraní rozvrhu sa napríklad nemusí podariť vyhovieť všetkým požiadavkám a nesplnené požiadavky sa takto môžu označiť, aby bolo neskôr možné dohľadať, ktorých vyučujúcich je potrebné kontaktovať a vzniknutú situáciu s nimi riešiť.



Obr. 3.1: Celkový diagram prípadov použitia navrhovanej aplikácie. Prípady použitia v obrázku označené ako UC_01 a UC_02 sú ďalej podrobnejšie rozobraté v obrázkoch 3.2 a 3.4.

Ďalšiou potrebnou funkciou pri práci s požiadavkami je možnosť importovať staršie požiadavky do aktuálnych. Nová aplikácia preto bude obsahovať obrazovku, ktorá pre daný predmet zobrazí kompletnú historiu požiadaviek vyučujúcich a to tak pre rozvrh skúšok, ako aj rozvrh výuky. Prostredníctvom tejto obrazovky bude možné označiť, ktoré zo starších požiadaviek sa vzťahujú aj na aktuálny rozvrh. Potreba tejto funkcie vyplýva z praktických skúseností s tvorbou rozvrhu, kde sa stretávame napríklad so situáciou, kedy vyučujúci každoročne uvádza, že nechce prednášať skoro ráno (7:00-9:00). Ak vyučujúci pravidelne zadáva takúto požiadavku a tento rok v požiadavkach chýba, je pravdepodobné, že táto požiadavka stále platí a je užitočné mať možnosť ju jednoduchým spôsobom importovať do

aktuálnych požiadaviek. Túto funkciu je zároveň možné kombinovať s už zmienením ofarbovaním jednotlivých políčok a označiť si, ktoré požiadavky boli explicitne zadané a ktoré boli importované z minulých rokov. Možnosti užívateľa pri práci s požiadavkami vyčujúcich sú znázornené diagramom prípadov použitia na obr. 3.2.



Obr. 3.2: UC_01: Práca s požiadavkami vyučujúcich

Príprava vstupných dát je nutnou prerekvizitou práce s rozvrhom. V porovnaní so súčasným stavom, kedy sú napríklad informácie o študijných plánoch uvedené priamo v zdrojových kódoch používanej aplikácie a musia sa pred vytvorením rozvrhu ručne kontrolovať, by navrhovaný prístup mal byť rýchlejší a komfortnejší. Po príprave vstupných dát je možné vytvoriť a editovať rozvrh. Editácia rozvrhu vyžaduje tieto funkcie:

- Vizualizácia jednotlivých časových okien rozdelených podľa dní v týždni umožňujúca vkladanie predmetov
- Vizualizácia obsadenosti miestností
- Zobrazenie zoznamu predmetov, ktoré ešte nie sú naplánované
- Práca s kolíziami, ktorá zahŕňa:
  - Zobrazenie počtu spoločných študentov pre danú dvojicu predmetov
  - Zobrazenie zoznamu predmetov, ktoré s daným predmetom nemajú žiadnych spoločných študentov
  - Zobrazenie počtu spoločných študentov pre každú dvojicu predmetov v danom dni
- Funkcie pre prácu s kapacitami miestností a potrebnou kapacitou pre danú udalosť, a to najmä:
  - Zobrazenie počtu študentov, ktorý sa zúčastnia danej udalosti (prednášky, prípadne určitého termínu skúšky)

– Zobrazenie dostupnej kapacity pre vybranú kombináciu miestností
– Zobrazenie potrebnej kapacity pre rôzne rozsadenia študentov podľa požiadaviek vyučujúcich pre rozvrhy skúšok

Spôsob, ktorým je riešené zobrazenie rozvrhu v aktuálne používanej aplikácií (popísaný v kapitole 2.4) a ktorý rozvrh zobrazuje ako maticu *deň v týždni* `x` *čas* (prípadne ⟨*deň v týždni, miestnosť*⟩ `x` *čas* pri pohľade na obsadenosť miestností), je na tento účel vhodný a intuitívny. Tento koncept preto bude zachovaný aj v novej aplikácii pre tvorbu rozvrhov.

V súčasnosti používaná aplikácia neposkytuje samostatný pohľad na kolízie jedného predmetu. Užívateľ má možnosť túto informáciu získať z výstupu funkcie `Kolize s okolím`, hľadané dáta sú tu ale prezentované spolu s informáciami o ďalších predmetoch. Informácia o počte kolízií pre jeden vybraný predmet umožní priamo porovnať dve potenciálne umiestnenia daného predmetu v rozvrhu a vybrať tak lepšie z nich. Navrhovaná podoba výstupu tejto funkcie je znázornená na obr. 3.3.

Kolize předmětu **SEN**		Celkově
S předpředchozím dnem	MAT 33	33
S předchozím dnem	ACH 23, EIP 1, THE 1	25
S aktuálním dnem	AIS 2, PDB 1, ROS 3	6
S následujícím dnem	BIO 13, ZZN 12, GMU 2	27
S 2. následujícím dnem	HSC 5, SIN 3, FAV 2	10
		101

Obr. 3.3: Navrhovaná podoba zobrazenia kolízii pre naplánovaný predmet

Aby osoba, ktorá rozvrh plánuje, nemusela súčasne sledovať veľké množstvo kritérií, rozvrh sa typicky plánuje v dvoch krokoch. V prvom kroku sa jednotlivým udalostiam pridelia časy, pričom berieme ohľad na kolízie a ostatné silné kritéria týkajúce sa času, na ktorý je udalosť naplánovaná. V druhom kroku, keď už má každá udalosť pridelený čas, pristúpime k plánovaniu do miestností. Je preto vhodné, aby aplikácia pre tvorbu rozvrhov obsahovala dva pohľady na rozvrh – s miestnosťami a bez nich. Spôsob, akým bol tento problém vyriešený v aktuálne používanej aplikácii, viedol k situáciu, že užívateľ síce mal dva pohľady na rozvrh, ale synchronizácia medzi nimi nebola automatická a v prípade zmeny v jednom bolo potrebné manuálne upraviť aj druhý pohľad.

V rámci navrhovanej aplikácie bude implementovaná automatická synchronizácia medzi týmito dvomi pohľadmi na rozvrh, pričom pri prepnutí z pohľadu *bez miestností* do pohľadu *s miestnosťami* program automaticky (podľa algoritmu 1) rozdelí udalosti do dostupných miestností. Aby algoritmus využíval dostupnú kapacitu miestností (resp. kombinácií miestností) efektívne, je potrebné, aby boli zoradené podľa kapacity vzostupne.

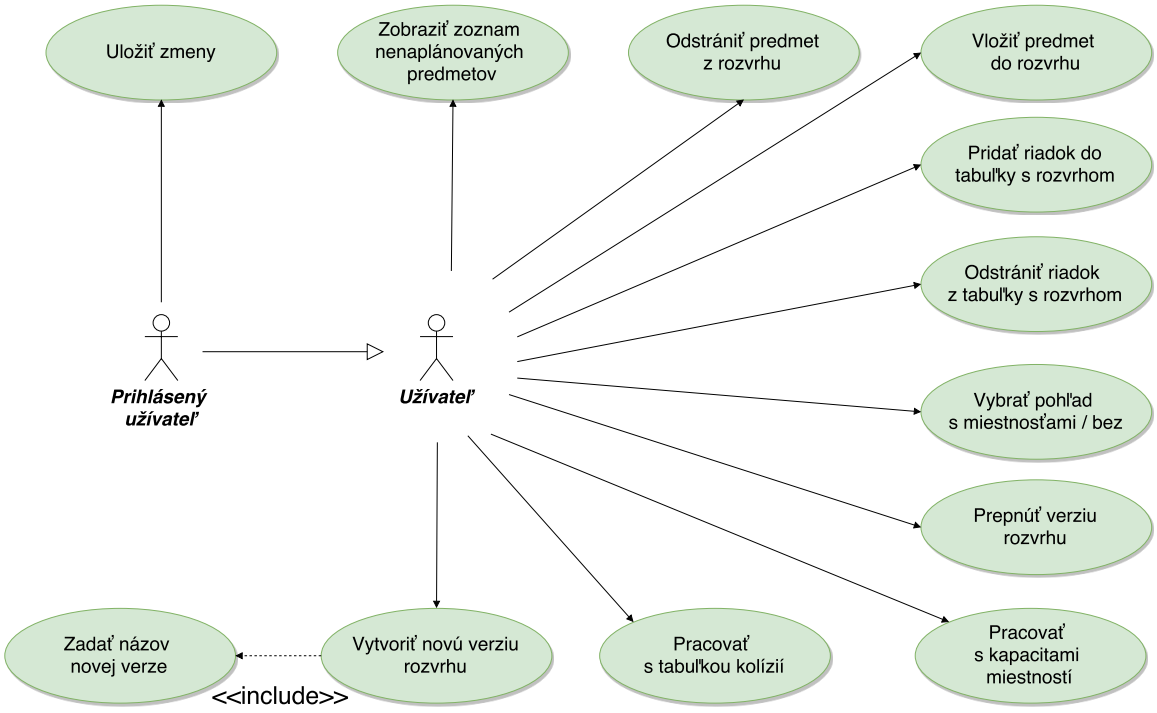
Pri manuálnom umiestňovaní predmetov do rozvrhu môže nastať situácia, kedy máme pre predmet k dispozícii viacero vhodných umiestnení a musíme zvoliť jedno z nich. Neskôr

```
Input: Rozvrh, zoznam miestností, zoznam povolených kombinácií miestností
Output: Každá udalosť má pridelenú miestnosť alebo kombináciu miestností
foreach den in tyzden do
    zoznam := Rozvrh.udalosti[den];
    zorad zoznam od najvacsej udalosti po najmensiu;
    foreach udalost in zoznam do
        if udalost uz ma pridelenu miestnost then
            continue
        end
        if pridelovanie miestnosti je zamknute then
            miestnost := docasnaMiestnost;
            continue
        end
        miestnostPridelena := false;
        foreach miestnost in miestnosti do
            if miestnost.kapacita >= udalost.potrebnaKapacita and
             miestnost.jeVolna(udalost.cas) then
                naplanuj udalost do miestnosti;
                miestnostPridelena := true;
                break
            end
        end
        if not miestnostPridelena then
            foreach kombinacia in povolene_kombinacie_miestnosti do
                if kombinacia.kapacita >= udalost.potrebnaKapacita and
                 kombinacia.jeVolna(udalost.cas) then
                    naplanuj udalost do miestnosti;
                    miestnostPridelena := true;
                    break
                end
            end
            if not miestnostPridelena then
                miestnost := docasnaMiestnost;
            end
        end
    end
end
```

Algoritmus 1: Rozdelenie naplánovaných udalostí do miestností. Algoritmus nezohľadňuje požiadavky vyučujúcich (rozsadenie, explicitne určená miestnosť a ďalšie), preto výsledok nemusí byť vždy použiteľný bez dodatočných úprav a je potrebné ho manuálne skontrolovať. *Zamknutie miestností* je funkcia, ktorá spôsobí, že algoritmus každej udalosti, ktorá ešte nemá pridelenú miestnosť, priradí *dočasnú miestnosť* a *zamknuté* rozloženie mestností sa už nebude meniť. Po pridaní novej udalosti do rozvrhu a prepnutí sa do pohľadu s miestnosťami užívateľ uvidí túto udalosť v dočasnej miestnosti a skutočnú miestnosť jej pridelí manuálne.

sa však môže ukázať, že vybraná cesta nevedie k cieľu – neumožňuje nájsť také rozloženie udalostí do miestností alebo časových okien, ktoré by spĺňalo všetky potrebné kritéria. Vzniká teda potreba vrátiť sa do určitého bodu, v ktorom bolo vykonané dané rozhodnutie, a skúsiť rozvrh zostaviť inak. Pre podporu tejto činnosti bude do novej aplikácie implementovaná možnosť pracovať v rámci jedného rozvrhu s viacerými **verziami**. Ak sa určitá verzia ukáže ako problematická, je možné sa jednoducho prepnúť na predchádzajúcu verziu.

Celkový prehľad akcií, ktoré bude mať užívateľ v navrhovanej aplikácii pri editácii rozvrhu dostupné, je znázornený diagramom prípadov použitia na obrázku 3.4.



Obr. 3.4: UC_02: Funkcie dostupné pri editácii rozvrhu.

Vyučujúci má možnosť si v informačom systéme zobraziť stránku so zoznamom zapísaných študentov v predmete, a to aj pre predmety, ktore sám neučí. Tieto informácie je možné použiť pri plánovaní na zobrazenie zoznamu študentov spoločných pre určitú skupinu predmetov. V súčasnosti používaná aplikácia neumožňuje pre danú trojicu predmetov zobraziť spoločných študentov a kolízie je nutné skúmať len na úrovni dvojíc, a aj v tomto prípade má užívateľ k dispozícii len agregovanú informáciu (počet spoločných študentov pre danú dvojicu predmetov). V rámci návrhu preto počítam s funkciou, ktorá pre zadanú skupinu predmetov zobrazí informácie o spoločných študentoch.

Zoznam študentov zapísaných v danom predmete však nie je verejne dostupnou informáciou, preto bude nutné do navrhovanej aplikácie implementovať autentizáciu užívateľa. Neautentizovaný užívateľ nebude mať možnosť pracovať so zoznamami študentov. Ako vyplýva z prezentovaných diagramov prípadov použitia, obmedzenie práv záťňa aj možnosť vykonávať perzistentné zmeny.

3.2 Architektúra aplikácie

V oblasti plánovania rozvrhov pripadajú do úvahy dva typy aplikácií – desktopová a webová. Pri vytváraní rozvrhu je niekedy potrebné riešiť reklamácie s vyučujúcim, čo by v prípade desktopovej aplikácie znamenalo potrebu fyzicky priniesť notebook s danou aplikáciou za ním, prípadne by sa táto aplikácia musela distribuovať medzi vyučujúcich, čo je zásadne nepraktické. V rámci tejto práce sme preto zvolili variantu webovej aplikácie (architektúra klient-server), ktorá poskytne potrebnú flexibilitu. *Klientská časť* aplikácie bude mať za úlohu poskytnúť užívateľské rozhranie pre prácu s vstupnými dátami a pre samotnú editáciu rozvrhov. *Serverová časť* bude zodpovedná za komunikáciu s databázou a ukladanie zmien vykonaných prostredníctvom klienta.

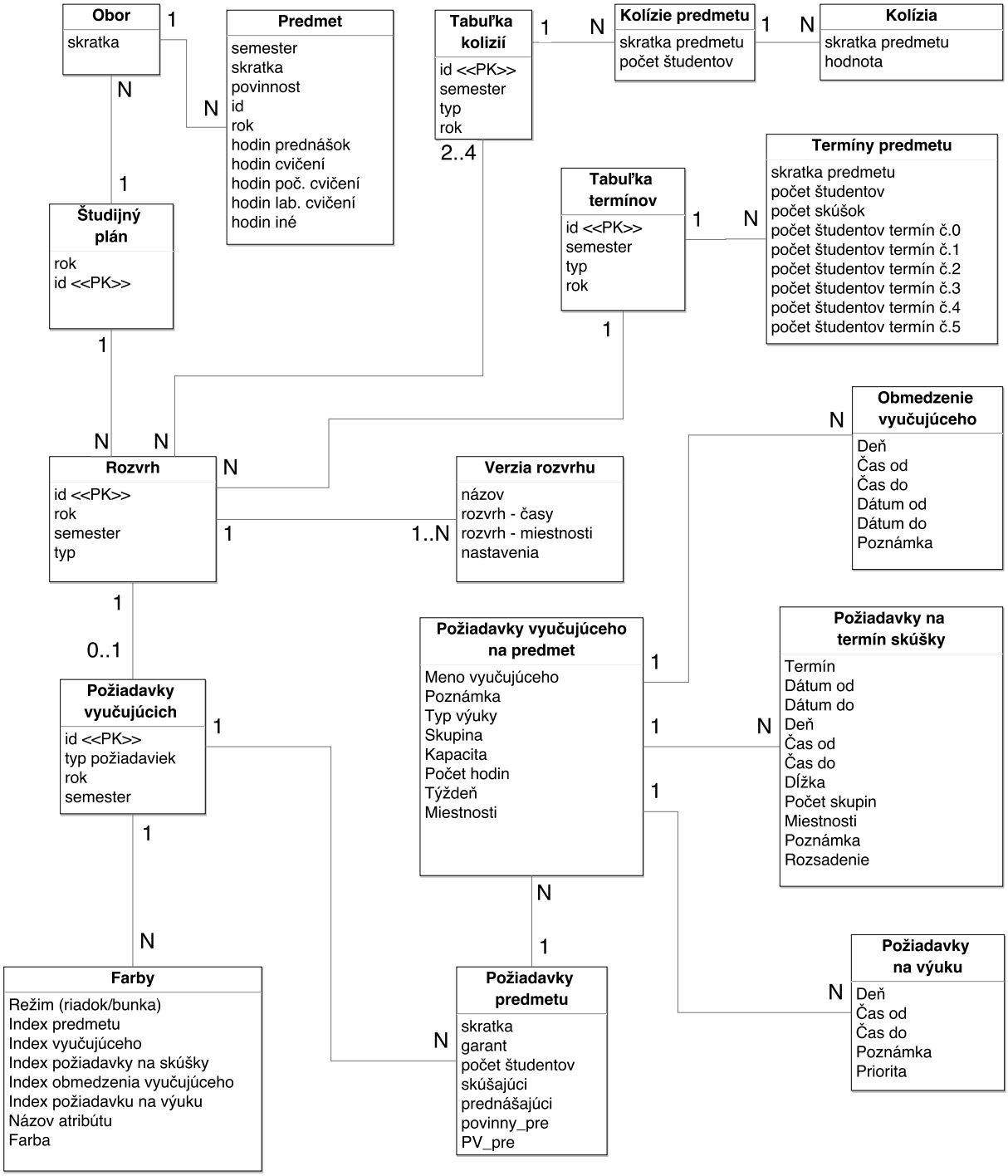
Keďže je predpoklad, že software, ktorý ako výsledok tejto práce vznikne, sa bude na FIT aktívne používať, je vhodné pri návrhu myslieť na udržovateľnosť a škálovateľnosť celého riešenia. Z tohto dôvodu budú spolu klient a server komunikovať výhradne prostredníctvom REST API (tento štýl rozhrania je podrobne popísaný v časti 4.1). Oddelenie vecí týkajúcich sa užívateľského rozhrania od vecí súvisiacich s uložením dát povedie k jednoduchšej štruktúre obidvoch komponent výsledného systému, ako pri použití tradičného MVC[1] prístupu, pri ktorom sa obsah stránok generuje na serveri a odosiela na klienta. Tento prístup zároveň umožní použiť rozdielne technológie na implementáciu užívateľského rozhrania a na implementáciu serverovej časti aplikácie. V budúcnosti bude teda možné reimplementovať serverovú časť s použitím iných technológií, prípadne naopak implementovať nové užívateľské rozhranie bez nutnosti modifikovať serverovú časť. Podrobný návrh rozhrania je uvedený v tabuľke 3.1.

Dáta, ktoré je potrebné v súvislosti s funkcionalitou popísanou v časti 3.1 ukladať v databáze, popisuje obr. 3.5.

[1] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

URL	Metóda	Popis
/	GET	Hlavná stránka
/savedReqs	POST*	Uloženie tabuľky s požiadavkami vyučujúcich
/savedReqs	GET	Prístup k uloženým tabuľkám s požiadavkami
/requirements/id	GET	Prístup k tabuľke s požiadavkami s daným id
/req/rok/sem/typ	PUT*	Aktualizácia tabuľky s požiadavkami pre daný rok, semester a typ rozvrhu (skúšky alebo výuka)
/colors/rok/sem/typ/	POST*	Uloženie informácie o tom, že užívateľ zmenil farbu určitej položky v tabuľke s požiadavkami pre daný rok, sem a typ rozvrhu
/colors/rok/sem/typ/	GET	Získanie informácie o farbách pre tabuľku s požiadavkami pre daný rok, semester a typ rozvrhu
/colors/rok/sem/typ/	DELETE*	Zmazanie informácie o farbách
/timetables	GET	Prístup k uloženým rozvrhom
/timetables/rok/sem/typ/ verzia/atribut	PUT*	Aktuálizácia atribútu určitej verzie rozvrhu
/createVersion/rok/sem/typ/ nazov	POST*	Vytvorenie novej verzie rozvrhu identifikovaného rokom, semestrom a typom
/createTimetable/rok/sem/typ	POST*	Vytvorenie nového rozvrhu
/setRequirements/rok/sem/typ/	PUT*	Priradenie tabuľky s požiadavkami k rozvrhu
/setCollisions/rok/sem/typ/	PUT*	Priradenie tabuľky kolízií k rozvrhu
/setTerms/rok/sem/typ/	PUT*	Priradenie tabuľky termínov k rozvrhu
/setStudyPlan/rok/sem/typ/	PUT*	Priradenie študijného plánu k rozvrhu
/courses/id	GET	Prístup k zoznamu predmetov pre určitý študijný plán
/courses	POST*	Uloženie informácie o predmetoch
/courses	GET	Získanie všetkých zoznamov predmetov
/study_programs/id	GET	Prístup k študijnému plánu s daným id
/study_programs	GET	Získanie všetkých študijných plánov
/study_programs/id	DELETE*	Odstránenie študijného plánu s daným id
/collisions/rok	GET	Získanie tabuľky kolízií pre daný rok
/collisions/rok	POST*	Uloženie tabuľky kolízií pre daný rok
/collisions/id	GET	Prístup k tabuľke kolízií s daným id
/collisions	GET	Získanie všetkých tabuliek kolízií
/collisions/id	DELETE*	Odstránenie tabuľky kolízií s daným id
/terms/rok/sem	POST*	Uloženie tabuľky termínov pre daný rok a semester
/terms/id	GET	Prístup k tabuľke termínov s daným id
/terms	GET	Získanie všetkých tabuliek termínov
/terms/id	DELETE*	Odstránenie tabuľky termínov s daným id
/enrollments	POST*	Uloženie zoznamu zapísaných študentov
/enrollments	GET*	Získanie zoznamu zapísaných študentov
/login	POST	Prihlásenie užívateľa

Tabuľka 3.1: Návrh REST API, prostredníctvom ktorého budú spolu klient a server komunikovať. Metódy označené * sú prístupné len autentizovanému užívateľovi.



Obr. 3.5: ER diagram zachytávajúci požiadavky na dáta, ktoré je potrebné v navrhutej aplikácii ukladať v databáze. Rozvrhu sú priradené 4 typy vstupných dát (študijný plán, tabuľka kolízií, tabuľka termínov, tabuľka s požiadavkami vyučujúcich). Na prvý pohľad netypická kardinalita 2-4 vzťahu rozvrh – tabuľka kolízií reprezentuje skutočnosť, že každý rozvrh musí mať priradené minimálne 2 tabuľky kolízií, jednu pre bakalárske a jednu pre magisterské predmety. Zároveň používame dva typy tabuliek kolízií, a to tabuľku so všetkými predmetmi a tabuľku len s predmetmi, ktoré plánujeme.

Kapitola 4

Nástroje a technológie použité pre vývoj

Táto kapitola obsahuje popis technológií, nástrojov a knižníc použitých pri implementácii aplikácie navrhnutej v kapitole 3. Úvod kapitoly je venovaný popisu architektúry REST (podkapitola 4.1), ktorá tvorí rozhranie medzi klientom a serverom. Ďalej sú diskutované technológie použité pri implementácii klienta (podkapitola 4.2) a servru (podkapitola 4.3). Prehľad použitých technológií uzatvára podkapitola 4.4 venovaná predspracovaniu vstupných dát. Záver kapitoly poskytuje prehľad využitia uvedených technológii v kontexte architektúry aplikácie.

4.1 REST API

REST predstavuje štýl architektúry používany pri návrhu distribuovaných systémov založený na webových štandardoch. Jedná sa o architektúru pre webové API, ktorá pre prístup k zdrojom používá protokol HTTP[1], pričom je založená na nasledujúcich princípoch [10]:

- **Klient-Server**. Server implementuje množinu služieb, ktoré sú dostupné klientom, ktorý o ne požiadajú (komunikácia prebieha štýlom dotaz-odpoveď).
- **Bezstavovosť**. Každá požiadavka klienta v sebe musí obsahovať všetky informácie potrebné na jej spracovanie a nesmie sa spoliehať na žiadny kontext uložený na strane servra. Keďže server nemusí udržovať žiadny stav medzi jednotlivými požiadavkami klienta, môže rýchlejšie uvoľnovať zdroje potrebné na obsluhu jednotlivých požiadaviek.
- **Cache**. Odpoveď servru obsahuje príznak, ktorý definuje, či je možné túto odpoveď na strane klienta uložiť do medzipamäti (cache) a znovu ju použiť pri ďalšej rovnakej požiadavke. Tým architektúra REST dosahuje zvýšenie efektivity využitia siete a zníženie latencie tam, kde je to možné.
- **Jednotné rozhranie**. Požiadavka jednotného rozhrania predstavuje súbor obmedzení kladených na služby dostupné prostredníctvom architektúry REST s cieľom zjednodušenia celej architektúry a zabezpečenia oddelenia jednotlivých služieb. Jedná sa o tieto obmedzenia:

[1] Hypertext Transfer Protokol, https://cs.wikipedia.org/wiki/Hypertext_Transfer_Protocol

  - Každý zdroj je jednoznačne identifikovaný svojou URI
  - Ak má klient určitý zdroj, má dostatok inormácií k tomu, aby daný zdroj mohol modifikovať alebo zmazať
  - Správy môžu obsahovať odkazy na ďalšie oblasti API

  Prístup k zdrojom prebieha s využitím súboru operácií CRUD, ktoré sú popísané nižšie.

- **Systém vrstiev**. Architektúra môže pozostávať z viacerých vrstiev, pričom každá komponenta vidí iba vrstvu, s ktorou komunikuje. To umožňuje prítomnosť zdieľaných prostredníkov (dodatočných servrov), ktoré zapúzdria staršie služby, nové služby tým pádom nemusia byť spätne kompatibilné. Zároveň je možné delegovať zriedka používanú funkcionalitu na prostredníka a tým zjednodušiť inú komponentu.

- **Code-On-Demand**. Táto vlastnosť umožňuje rozširovanie funkcionality klienta aj po nasadení prostredníctvom sťahovania a výkonávania kódu vo forme appletov alebo skriptov.

Podrobnejší popis jednotlivých vlastností REST architektúry je možné nájsť v práci Roya Fieldinga, ktorý tieto princípy navrhol a popísal vo svojej dizertačnej práci [10].

CRUD

REST API je založené na použití metód HTTP pre komunikáciu medzi klientom a serverom. CRUD alebo **C**reate-**R**ead-**U**pdate-**D**elete predstavuje štyri základné funkcie perzistentného ukladania, ktoré sú mapované na metódy HTTP a ich význam je nasledujúci (zdroj je identifikovaný pomocou URI požiadavku):

- **Create** – vytvorenie nového zdroja metódou POST
- **Read** – získanie zdroja metódou GET
- **Update** – aktualizácia zdroja alebo zmena jeho stavu metódou PUT
- **Delete** – odstránenie zdroja pomocou metódy DELETE

Po spracovaní požiadavku server zasiela klientovi odpoveď, ktorou ho informuje o výsledku požadovanej operácie. Používané odpoveďové kódy a ich význam je možné nájsť v literatúre [19].

4.2 Klient

Táto podkapitola obsahuje popis technológií použitých na implementáciu klientskej časti aplikácie, teda užívateľského rozhrania.

Jazyk HTML

HyperText Markup Language (skratka HTML) je značkovací jazyk používaný pre tvorbu webových stránok. Pomocou značiek (tzv. tagov) a ich vlastností (atribútov) popisujeme štruktúru dokumentu, ktorý tvorí základ webovej stránky [12]. HTML je v rámci tejto práce používané v aktuálnej verzií 5.

CSS

Cascading Style Sheets (CSS) je jazyk, ktorý slúži na popis spôsobu zobrazenia dokumentu vo formáte HTML [6]. Kód v jazyku CSS je interpretovaný webovým prehliadačom, pričom môže byť vložený buď v samostatnom súbore (typicky s príponou .css), v HTML dokumente medzi značkami `<style>` a `</style>`, alebo priamo v atribútoch jednotlivých značiek HTML. Použitá a zároveň aktuálna verzia CSS je 3.

JavaScript

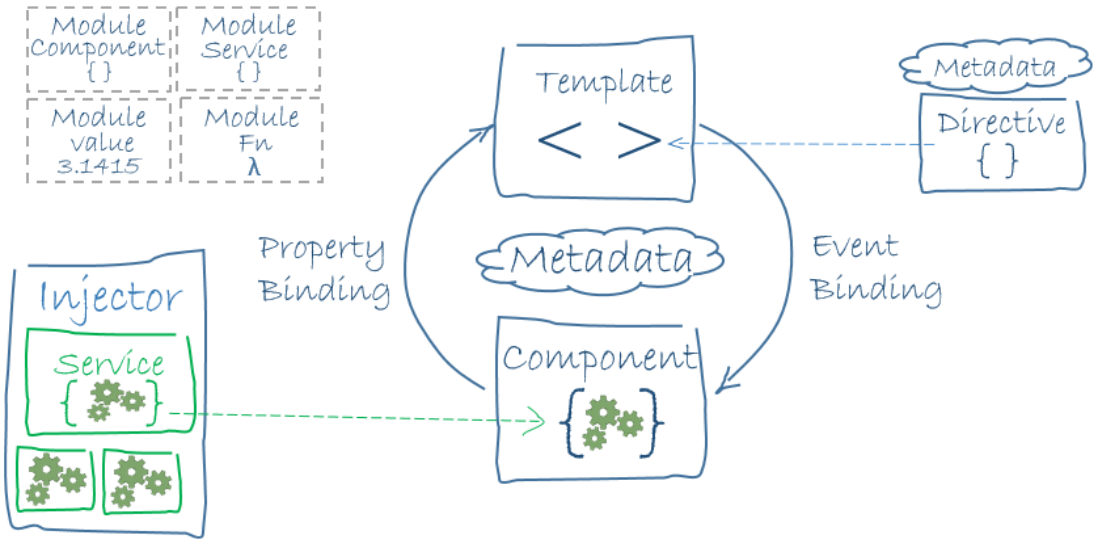
JavaScript je objektovo-orientovaný skriptovácí jazyk, v súčastnosti používaný najmä pri vývoji webových aplikácií. Kód písaný v JavaScripte je interpretovaný prehliadačom, teda na strane klienta. To umožňuje vytvárať vysoko responzívne užívateľské rozhrania, ktoré poskytujú dynamickú funkcionalitu bez nutnosti čakania nato, kým server zobrazí ďalšiu stránku [24] [13].

TypeScript

TypeScript je typovaná nadmnožina jazyka JavaScript, prekladá sa na čistý JavaScript. Ako napovedá samotný názov, TypeScript pridáva do jazyka JavaScript možnosť používať statické typové kontroly, čo napomáha odhaleniu niektorých chýb už v dobe prekladu [22]. Používanie typov je voliteľné, takže v rámci kódu v jazyku TypeScript je možné používať už existujúce knižnice napísané v jazyku JavaScript.

Angular

Angular je framework pre vývoj dynamických webových aplikácií s otvoreným zdrojovým kódom. Aplikácie napísane s použitím Angularu sú zložené z *komponent*. Komponenta odpovedá triede v jazyku TypeScript, ktorá je doplnená šablonou a svojím vlastným štýlom (CSS). Šablony su definované v jazyku HTML doplneným o špecifické prvky frameworku Angular. Inštancie objektov danej triedy vytvára Angular automaticky pri zobrazení príslušnej stránky [23] [5]. Základný princíp činnosti frameworku je znázornený na obr. 4.1.



Obr. 4.1: Architektúra frameworku Angular. Šablona (*Template*) je obojsmerne previazaná so svojou komponentou. *Event binding* umožňuje zavolať ľubovoľnu metódu komponenty pri vyvolaní určitej udalosti (kliknutie, stlačená klávesa apod.). *Property binding* automaticky aktualizuje šablonu pri zmene dát komponenty. *Service* je špeciálny druh komponenty, ktorého úlohou je poskytovanie určitej služby, typicky komunikácie zo serverom. Pri deklarácii komponenty stačí uviesť, že komponenta požaduje určitý *Service*, o jeho vytvorenie sa Angular postará automaticky. Tento mechanizmus nazývame *dependency injection*. Podrobný popis jednotlivých častí je možné nájsť v dokumentácii, z ktorej bol tento obrázok prevzatý [3].

Angular rozširuje jazyk HTML o syntaktické konštrukcie, ktoré umožňujú vytváranie šablón webových stránok. Príkladom takejto konštrukcie je `ngFor`, ktorého použitie približuje príklad 4.1.

```
<select>
   <option *ngFor="let opt of options"> {{ opt }} </option>
</select>
```

Príklad 4.1: Ukážka použitia konšturkcie `ngFor` v šablone pre vytvorenie menu pre výber z viacerých možností. Premenná *options* je pole obsahujúce možnosti, z ktorých chceme dať užívateľovi na výber. Premenná *opt* postupne nadobúda hodnoty jednotlivých prvkov poľa *options*. V každej iterácii cyklu je do výsledného HTML pridaný jeden tag `<option>`.

Pri implementácii bola použitá verzia Angular 2.4.8.

Bootstrap

Bootstrap je framework pre vývoj webových aplikácií poskytujúci predpripravené komponenty v HTML a množstvo tried CSS, ktoré zohľadňujú vlastnosti použitého zariadenia (desktop, tablet, mobil) a prispôsobuju mu rozloženie stránky. Súčasťou tohto frameworku je aj široká sada ikon. Viac informácií je možné nájsť na stránkach projektu Bootstrap[2]. Pri implementácii bola použitá verzia 3.3.7.

[2] http://getbootstrap.com/

ReactiveX

ReactiveX je knižnica zameraná na podporu asynchrónneho programovania. Z tejto knižnice bola pri implementácii využitá najmä trieda `Observable`, ktorá implementuje návrhový vzor Observer. Trieda `Observable` poskytuje API, ktoré umožňuje zaregistrovať funkciu, ktorá bude zavolaná pri výskyte sledovanej udalosti. Klasickým príkladom použitia je asynchrónne volanie servra pre získanie dát, ktoré je s využitím triedy observable možné bez explicitného vytvárania vlákien. Podrobný prehľad možností tejto knižnice je možné nájsť na jej webových stránkach[3].

Táto knižnica je dostupná pre všetky bežné programovacie jazyky. Použitá bola varianta určená pre jazyk JavaScript (označovaná RxJS) verzie 5.1.1.

Underscore

Underscore je knižnica napísaná v jazyku JavaScript poskytujúca sadu funkcií známych napríklad z funkcionálneho jazyka Haskell (`map`, `reduce`, `filter`, `foldr` a iné). V rámci tejto práce bola z knižnice Underscore použitá iba funkcia `extend`, ktorá ma dva parametre. Prvým parametrom je ľubovoľný objekt, druhým parametrom je prototyp iného objektu. Funkcia vezme všetky atribúty a metódy daného prototypu, a pridá ich do objektu, ktorý je prvým parametrom. Táto funkcia bola použitá v kombinácii so serializáciou objektov pomocou formátu JSON[4], ktorý neumožňuje serializáciu metód a teda pri serializácii a následnej deserializácii objektu prídeme o všetky metódy. Funkcia `extend` vznikla za účelom vyriešenia práve tohto problému.

jQuery

jQuery je multiplatformová knižnica pre jazyk JavaScript navrhnutá s cieľom zjednodušiť použitie JavaScriptu pri vývoji webových stránok. Veľké množstvo bežných úloh, ktoré vyžaduju väčší počet riadkov kódu v čistom JavaScripte, je možné pomocou jQuery implementovať jedným riadkom kódu [4]. Pri implementácii bola použitá knižnica jQuery verzie 3.1.1.

4.3 Server

Táto podkapitola popisuje technológie použité pri vývoji serverovej časti aplikácie.

Node.js

Node.js je prostredie pre beh JavaScriptu na strane serveru založené na interprete `V8` od Googlu a navrhnuté pre vývoj škálovateľných sieťových aplikácií [2]. Charakteristickým rysom Node.js je udalosťami riadený programovací model. Programátor definuje, aká funkcia sa má zavolať pri príchode požiadavku na danú URI, a nemusí implementovať mechanizmy pre paralelnú obsluhu viacerých klientov (vlákna, zamykanie) [2].

[3] http://reactivex.io/
[4] JavaScript Object Notation, http://www.json.org/json-cz.html.

Express

Express je framework pre vývoj webových aplikácii v prostredí Node.js. Poskytuje rozhranie pre jednoduché vytváranie koncových bodov (tzv. *endpoints*), na ktorých sú prístupné jednotlivé zdroje, resp. služby poskytované serverom. Popis implementovaných koncových bodov bol uvedený v tabuľke 3.1. Implementácia serverovej časti aplikácie je založená na verzii 4.14.1.

Passport

Passport je knižnica určená pre použitie v prostredí Node.js, ktorá implementuje autentizáciu užívateľov. Prehľad dostupných autentizačných schém a možností tejto knižnice je možné nájsť na jej webových stránkach[5]. Pri implementácii bola použitá verzia 0.3.2.

MongoDB

MongoDB je multiplatformová dokumentová databáza. Ukladanie a spracovanie dát v databáze MongoDB používa iné prostriedky než tabuľkové schémy tradičnej relačnej databázy, MongoDB sa preto radí medzi databáze typu NoSQL [16]. Dokumenty su uložené vo formáte podobnom formátu JSON. Pri implementácii bola použitá verzia 2.6.10.

Mongoose

Mongoose je framework, ktorý sprostredkováva prácu s databázou MongoDB v prostredí Node.js. Poskytuje funkcie pre definíciu štruktúry dokumentov, vyhľadávanie v databáze a modifikáciu a ukladanie dokumentov. Kompletný prehľad dostupných funkcií je možné nájsť v dokumentácii tohto frameworku[6]. Pri implementácii bola použitá verzia 4.5.8.

HTMLMinifier

HTMLMinifier je konfigurovateľný nástroj na zmenšenie veľkosti dokumentov vo formáte HTML napísaný v jazyku JavaScript, ktorý aplikácia využíva pri exporte dát. Funkcionalita tohto nástroja je dostupná prostredníctvom funkcie `minify`, ktorej parametrom je dokument vo formáte HTML a objekt špecifikujúci nastavenia minimalizácie. Nástroj je dostupný priamo zo svojho repozitára[7].

4.4 Predspracovanie vstupných dát

Externou súčasťou aplikácie je nástroj na predspracovanie vstupných dát, ktoré sú dostupné vo formáte HTML buď priamo na stránkach fakulty (odkiaľ je potrebné ich stiahnúť), alebo v podobe súborov exportovaných z informáčného systému fakulty. Na jeho implementáciu bol použitý jazyk Python.

[5] http://passportjs.org/
[6] http://mongoosejs.com/docs/guide.html
[7] https://github.com/kangax/html-minifier

Python

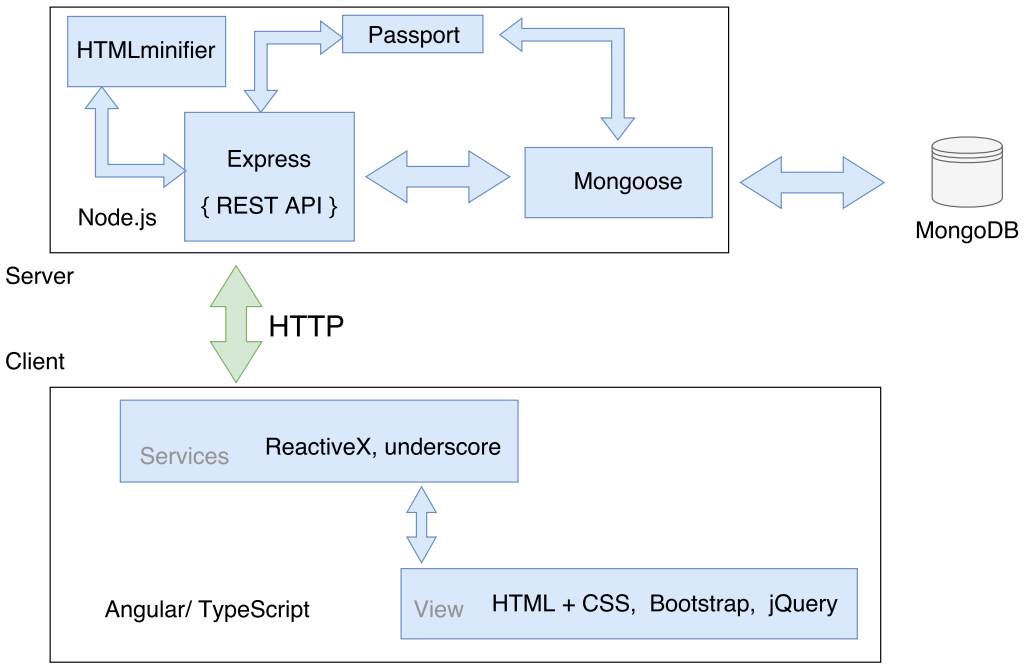
Python je vysokoúrovňový interpretovaný programovací jazyk často používaný pri vývoji automatizovaných nástrojov a skriptov. Podrobnejšie informácie o tomto programovacom jazyku je možné nájsť napríklad na oficiálnych webových stránkach [17]. Pri implementácii bol použitý Python verzie 3.

BeautifulSoup

BeautifulSoup je knižnica pre jazyk Python na prácu so súbormi vo formáte HTML a XML. Poskytuje rozhranie pre jednoduchý prístup k elementom a ich obsahu podľa rôznych kritérií. Komplentý prehľad možností knižnice BeautifulSoup je možné nájsť na jej webových stránkach[8]. Pri implementácii bola použitá verzia 4.4.1.

4.5 Zhrnutie

Keďže počet použitých technológií je relatívne veľký, uvádzam na záver tejto kapitoly prehľadový obrázok architektúry aplikácie s ich znázornením (obr. 4.2).



Obr. 4.2: Celková architektúra aplikácie s vyznačením vybraných použitých technológií.

[8] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Kapitola 5

Implementácia aplikácie pre tvorbu rozvrhov

Ná základe návrhu prezentovaného v kapitole 3 bola implementovaná webová aplikácia pre podporu manuálnej tvorby rozvrhov. Táto kapitola je venovaná popisu implementovanej aplikácie. Vytvorená aplikácia má na najvyššej úrovni nasledujúcu štruktúru:

```
client/
   app/
   assets/
   package.json
   config.ts
server/
   models/
   controllers/
   app.js
   package.json
   rozvrhy.sh
inputs/
   inputs.py
```

Adresár `client` obsahuje implementáciu webového rozhrania aplikácie (zdrojové kódy sú umiestnené v podadresári `app`). Podadresár `assets` obsahuje skripty, súbory štýlov (.css) a ikony, ktoré sú súčasťou použitých knižníc. Súbor `package.json` definuje závislosti (ostatné potrebné knižnice) pre beh klienta. `config.ts` je konfiguračný súbor umožňujúci jednoducho meniť vybrané vlastnosti užívateľského rozhrania. Implementácii klienta je venovaná podkaptiola 5.2.

Adresár `server` obsahuje implementáciu serverovej časti aplikácie. V podadresári `models` sú umiestnené definície databázových modelov (vytvorené na základe ER Diagramu prezentovaného na obr. 3.5). Podadresár `controllers` obsahuje funkcie pre prácu s definovanými modelmi. Súbor `app.js` je vlastnou implementáciou serveru – na základe požadovanej URI predáva riadenie jednej z týchto funkcií. Súbor `package.json` obsahuje závislosti potrebné pre beh serveru. Skript `rozvrhy.sh` automatizuje úkony potrebné pri spustení serveru. Implementácii serverovej časti aplikácie je venovaná podkapitola 5.1.

Skript `inputs.py` slúži k predspracovaniu vstupných dát a jeho možnosti sú podrobne popísané v prílohe A.

5.1 Server

Serverová časť aplikácie (ďalej len server), plní dve funkcie. Plní úlohu webového servera, ktorý na základe HTTP požiadavok sprístupňuje klientom statické súbory. Keby sme v rámci implementovanej aplikácie požadovali iba túto funkcionalitu, bolo by možné použiť niektorý z existujúcich voľne dostupných HTTP serverov ako napríklad Apache, alebo Nginx. Server v tomto prípade ale poskytuje aj funkcie špecifické pre túto aplikáciu, ako napríklad vytvorenie rozvrhu, priradenie tabuľky s požiadavkami vyučujúcich k určitému rozvrhu apod. Podporovaná funkcionalita je definovaná REST rozhraním uvedeným v tabuľke 3.1. Zo strany serveru sa samozrejme jedná len o zabezpečenie perzistentnosti akcií, ktoré iniciuje užívateľ prostredníctvom webového rozhrania (klientskej časti) aplikácie.

Súčasťou jednotlivých operácií na strane serveru sú kontroly vzájomnej previazanosti objektov prostredníctvom atribútu `_id`, ktorý databáza MongoDB automaticky prideľuje každému objektu a ktorým je tento objekt zároveň unikátne identifikovaný v rámci danej databázy. Napríklad pri požiadavke na odstránenie študijného plánu je potrebné skontrolovať, že daný študijný plán nie je priradený k žiadnemu rozvrhu (tzn. žiadny rozvrh nemá nastavený atribút `study_plan`, ktorý označuje priradený študijný plán, na hodnotu `_id` študijného plánu, ktorý sa pokúšame odstrániť. V prípade, že takýto rozvrh existuje, operácia skončí s chybovým kódom *403 (Forbidden)*.

Pri príchode požiadavku, ktorý vyžaduje, aby bol užívateľ autentizovaný, server kontroluje hlavičku `Authorization`, ktorá obsahuje identifikáciu užívateľa. Ak táto hlavička nie je v správe prítomná alebo nemá platnú hodnotu, operácia skončí s chybou *401 (Unauthorzied)*.

Webový server bol implementovaný s využitím frameworku *Express*, ktorý poskytuje podporu aj pre riadenie cache. Odpovede serveru teda obsahuju príznak, na základe ktorého klienti môžu metódou *HEAD* zistiť, či bol daný zdroj modifikovaný a ten sa tak nemusí prenášať znovu, ak sa od posledného stiahnutia klientom nezmenil. Ďalším opatrením pre zvýšenie efektivity využitia siete je kompresia odpovedí na strane serveru. Každá odpoveď je pred odoslaním skomprimovaná nástrojom *gzip*. Výhodou z pohľadu programátora je, že táto funkčnosť nevyžaduje žiadne úpravy klienta, keďže o dekompresiu sa automatický stará prehliadač. Vďaka tomuto opatreniu sa významným spôsobom podarilo znížiť objem prenášaných dát.

Komunikácia medzi klientom a serverom prebieha výhradne vo formáte JSON. Jedinou výnimkou z tohto pravidla je minifikácia exportovaného HTML, kedy klient zasiela v rámci požiadavky pôvodné, nezmenšené HTML a server mú v odpovedi zasiela jeho minifikovanú verziu. Pri procese minifácie sú z daného HTML odstránene komentáre, biele znaky a ďalšie informácie tak, aby sa nezmenila jeho výsledná podoba. V praxi to umožňuje zmenšiť veľkosť exporotvaných súborov na približne 10% pôvodnej veľkosti. Posielanie exportovaného HTML na server a späť sa môže na prvý pohľad javiť ako zbytočné. Tento spôsobo implementácie bol zvolený preto, že všetky dostupné nástroje s použiteľnou licenciou boli určené výhradne pre použitie na strane serveru.

Pre potreby inštalácie a správnej funkčnosti serveru je nutné mať nainštalované tieto nástroje:

- `npm` – Node Package Manager. Správca balíčkov v prostredí node.js, potrebný pre automatickú inštaláciu závislostí
- `node` – prostredie pre beh JavaScriptu, nutné pre spustenie aplikácie
- `tsc` – TypeScript Compiler. Prekladač z jazyka TypeScript do jazyka JavaScript

- `mongod` – databázový démon MongoDB

Ostatné potrebné nástroje a knižnice je možné doinštalovať na základe konfigurácie v súbore `package.json`, ktorý je distribuovaný spolu s aplikáciou, príkazom `npm install`.

Ovládanie serveru je sprostredkované skriptom `rozvrhy.sh`. Skript je možné použiť nasledovnými spôsobmi:

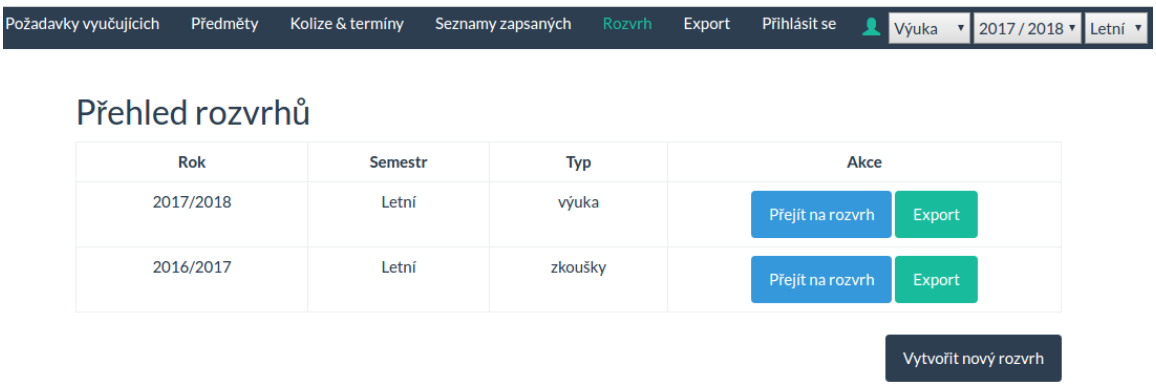
- `rozvrhy.sh clean` – odstráni všetky súbory vygenerované prekladačom jazyka TypeScript.
- `rozvrhy.sh run PORT DBURL` – spustí server. Parameter `PORT` definuje číslo portu, na ktorom bude server očakávať spojenia klientov. Server bude komunikovať s databázou určenou parametrom `DBURL`, ktorý musí mať nasledovnú syntax:

  mongodb://**host**/**dbname**

  kde parameter **host** udáva doménové meno stanice, na ktorej je spustená príslušná databáza. Parameter **dbname** definuje názov databázy, ku ktorej sa chceme pripojiť. Názov je možné nastaviť na ľubovoľný reťazec a ak databáza s daným názvom neexistuje, automaticky sa pri pokuse o pripojenie vytvorí.

5.2 Klient

Úlohou klientskej časti je sprostredkovať dostupné funkcie aplikácie a to podľa možností jednoduchým a intuitívnym spôsobom. Aplikácia je rozdelená do niekoľkých častí, medzi ktorými sa užívateľ prepína pomocou horizontálneho menu v jej hornej časti (obr. 5.1). Štruktúra tejto podkapitoly bude ďalej odpovedať jednotlivým častiam implementovanej aplikácie.



Obr. 5.1: Ukážka obrazovky s prehľadom uložených rozvrhov. Horná časť aplikácie obsahuje menu, pomocou ktorého si užívateľ volí, či chce pracovať so vstupnými dátami (kategórie *Požadavky vyučujících, Předměty, Kolize & Termíny, Seznamy zapsaných studentů*), s rozvrhom, alebo exportovať informácie z uložených rozvrhov. V pravej časti navigačného panelu sú umiestnené ovládacie prvky pre výber aktívneho rozvrhu.

Požiadavky vyučujúcich

V tejto sekcii má užívateľ možnosť zobraziť si prehľad uložených tabuliek s požiadavkami vyučujúcich, priradiť vybranú tabuľku k aktívnemu rozvrhu, prípadne, ak je to možné, odstrániť tabuľku s požiadavkami z databázy. Okrem toho sú v rámci tejto sekcie dostupné funkcie pre prácu s detailom požiadaviek a históriou požiadaviek tak, ako boli popísané diagromom prípadov použitia 3.2.

Funkčnosť ofarbovania jednotlivých políčok je implementovaná pomocou panelu nástrojov umiestneného staticky v ľavej hornej časti stránky. Prostredníctvom tohto panela užívateľ zvolí akciu (či chce vybranú položku zafarbiť, alebo odfarbiť), režim danej akcie (riadkový alebo bunkový) a prípadne výslednú farbu. V prípade použitia riadkoveho režimu je vybraná akcia aplikovaná na celý riadok tabuľky, inak má akcia dopad len na vybranú bunku. Príslušná akcia je potom vyvolaná kliknutím **pravým** tlačidlom myši do vybranej bunky, resp. riadku. Zoznam dostupných farieb je možné nastaviť v konfiguračnom súbore klienta, pričom každá farba je definovaná svojim hexadecimálnym kódom.

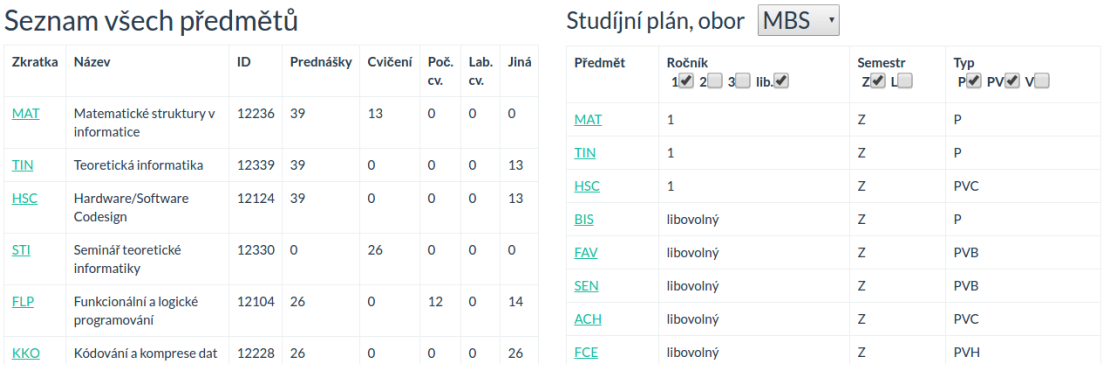
Pre zobrazenie tabuľky s požiadavkami vyučujúcich je v konfiguračnom súbore aj položka, ktorá definuje implicitné hodnoty niektorých políčok. Takéto hodnoty sú potom zobrazované inou, menej výraznou, farbou. Príkladom v tabuľke s požiadavkami na rozvrh výuky je hodnota *prednáška* v poli typ výuky, kde sa táto hodnota vyskytuje pri veľkej väčšine položiek. Zmyslom je umožniť užívateľovi sústrediť sa na dôležité informácie v danom zobrazení.

Zobrazenie skratky predmetu je implementované ako tlačidlo, ktoré prepne zobrazenie na históriu požiadaviek pre daný predmet. Tu je možnosť kliknutím označiť požiadavky pre import do aktuálnych. Po kliknutí na tlačidlo *Import* je užívateľ vrátený späť na zobrazenie tabuľky s aktuálnymi požiadavkami, pričom táto je doplnená o importované požiadavky. Pri návrate aplikácia zachováva presnú vertikálnu pozíciu na stránke, takže užívateľ nemusí kolieskom myši hľadať pozíciu, kde sa nachádzal predtým.

Zobrazenie tabuľky s požiadavkami zahŕňa aj možnosť exportovať aktuálnu podobu tabuľky vo formáte HTML.

Študijné plány

Táto sekcia poskytuje základné funkcie pre manipuláciu študijných plánov a to vkladanie, priradenie k aktívnemu rozvrhu, zobrazenie detailu a odstránenie. Zobrazenie detailu študijného plánu a dostupných funkcií približuje obrázok 5.2.



Obr. 5.2: Zobrazenie študijného plánu pozostáva z dvoch časti. V ľavej časti je zoznam všetkých predmetov vyučovaných v rámci daného študijného plánu, pričom pre každý predmet su zobrazované detailné informácie získané z webových stránok fakulty. Skratka predmetu funguje ako odkaz na stránku predmetu. Práva časť umožňuje zobraziť si informácie o predmetoch pre konkrétny obor a filtrovať ich pomocou zaškrtávacích tlačítok.

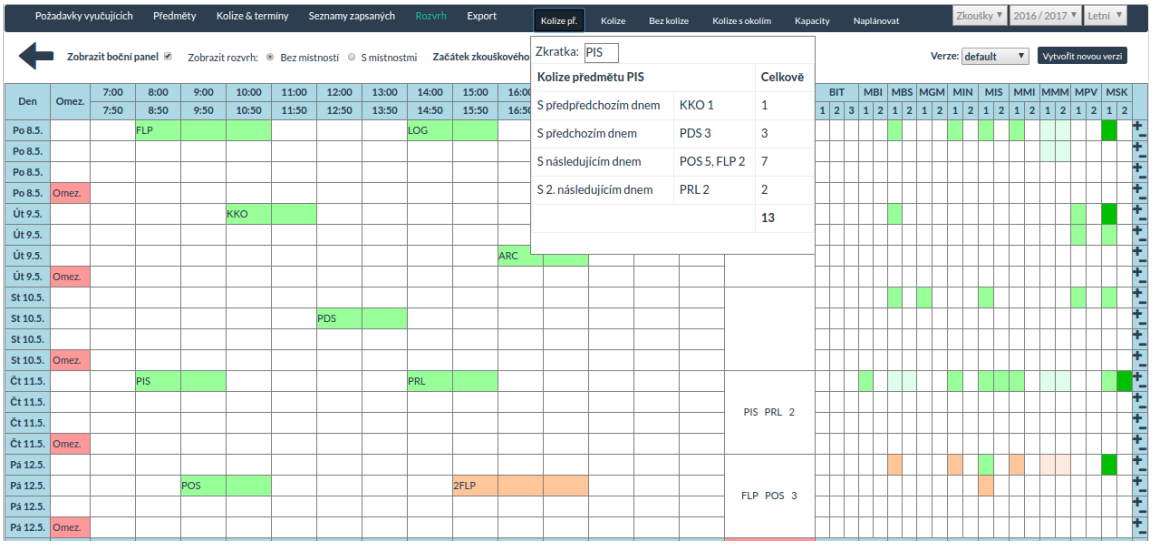
Kolízie & termíny, Zoznamy zapísaných študentov

V časti *Kolíze & termíny* je možné spravovať tabuľky kolízií, ktoré udávaju počty spoločných študentov pre každú dvojicu predmetov, a tabuľky s termínmi skúšok, ktoré obsahujú informácie o počte študentov, ktorý sa zúčastnili jednotlivých termínov skúšok. Podobne, v časti *Seznamy zapsaných studentů*, sú dostupné funkcie pre správu zoznamov zapísaných študentov. Správou rozumieme ich vkladanie, odstraňovanie a priraďovanie k aktívnemu rozvrhu.

Rozvrh

Primárnu časť aplikácie tvorí prostredie pre editáciu rozvrhu. Po kliknutí na položku *Rozvrh* v navigačnom menu aplikácia zobrazí prehľad uložených rozvrhov (obr. 5.1). Pomocou tlačidla *Přejít na rozvrh* sa aplikácia prepne do zobrazenia, v ktorom je možné editovať zvolený rozvrh. Funkcionalita dostupná pri editácií rozvrhu odpovedá pôvodnej aplikácii, o ktorej pojednávala kapitola 2.4. Naviac bola pridaná možnosť prepínať pohľad medzi režimom s miestnotami a bez, možnosť zobraziť kolízie pre jeden predmet a možnosť pracovať s viacerými verziami rozvrhu. Implementované boli všetky funkcie znázornené diagramom prípadov použitia na obr. 3.4. Podobu implementovaného prostredia pre prácu s rozvrhom zobrazuje obr. 5.3.

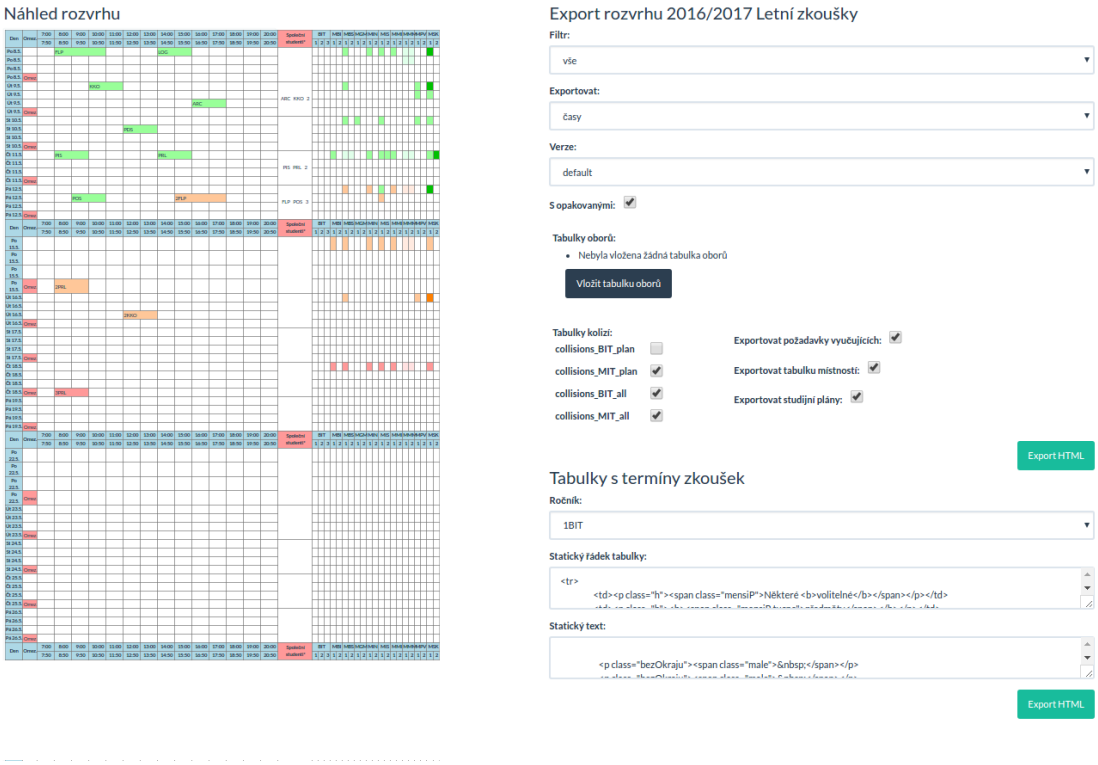
Ak boli k rozvrhu priradené zoznamy zapísaných študentov, v tejto časti aplikácie je zároveň dostupná funkcia, ktorá po zadaní čiarkou oddelených skratiek predmetov vypíše zoznam spoločných študentov medzi nimi. Súčasťou výstupu tejto funkcie je aj informácia o tom, či daný študent má predmet zapísaný ako povinný, povinne voliteľný alebo voliteľný.



Obr. 5.3: Prostredie pre prácu s rozvrhom vo vytvorenej aplikácii. Ľavá časť aplikácie zobrazuje rozloženie predmetov v čase. Pravá časť vizualizuje informácie o počte skúšok v rámci jednotlivých oborov (táto funkcionalita bola dostupná aj v existujúcej aplikácii). Na obrázku je okrem toho možné vidieť zobrazenie kolízií pre predmet PIS (Pokročilé informačné systémy).

Export dát

Po dokončení rozvrhu je potrebné zostavený rozvrh vyexportovať. Funkcie pre export sú zoskupené v sekcii dostupnej v navigačnom menu cez položku *Export*. Stránka venovaná exportu je rozdelená na dve časti. Ľavá časť poskytuje náhľad rozvrhu tak, ako bude exportovaný. Práva časť obsahuje ovládacie prvky pre filtrovanie výstupu a výber exportovaných dát. Filtrovaním rozumieme výber určitého ročníku, skupiny, alebo programu (bakalársky, magisterský). Okrem toho je tu možnosť zvoliť, či sa má exportovať pohľad na rozloženie do časových okien, alebo pohľad na rozdelenie do miestností. Aplikácia podporuje vytváranie rôznych verzií rozvrhu, preto je tu aj možnosť vybrať, ktorá verzia sa má exportovať. Pri výbere určitého ročníku, verzie alebo pri prepnutí na pohľad s miestnosťami resp. bez nich, sa náhľad v ľavej časti stránky automaticky aktualizuje. Celkovú podobu stránky venovanej exportu dát môžeme vidieť na obr. 5.4.



Obr. 5.4: Stránka s nastaveniami pre export dát. Voliteľne je možné do exportovaného rozvrhu zahrnúť aj vstupné dáta, a to konkrétne použité tabuľky kolízií, požiadavky vyučujúcich a študijné plány. Súčasťou exporotvaných dát môže byť aj tabuľka s počtami študentov na oboroch, ktorá sa inde v aplikácii nepoužíva, preto je tu možnosť ju explicitne vložiť. Aplikácia umožňuje exportovať aj tabuľku miestností, ktoré boli použité pri plánovaní.

Okrem exportu stránky s rozvrhom aplikácia umožňuje exportovať stránku s tabuľkou termínov skúšok pre vybraný ročníky. Tabuľka zahŕňa povinné predmety pre daný ročník spolu s vybranými voliteľnými predmetmi. Predmety zahrnuté v tomto type výstupu sú definované v konfiguračnom súbore. Pre export týchto tabuliek je k dispozícii možnosť vybrať ročník, upraviť obsah statického riadku tabuľky a statický text vkladaný za tabuľku. Statický riadok tabuľky typicky obsahuje odkazy na tabuľky pre iné ročníky, statický text obsahuje dodatočné informácie pre študentov.

Kapitola 6

Testovanie a nasadenie aplikácie

Testovanie aplikácie prebiehalo s využitím rozvrhu skúšok pre letný semester akademického roku 2015/2016. Tento rozvrh bol manuálne prepísaný do implementovanej aplikácie, pričom boli porovnané dostupné výstupy existujúcej aplikácie s aktuálnymi. Tým bolo overené, že algoritmus pre vizualizáciu počtu termínov skúšok pre jednotlivé študijné obory funguje rovnako, ako v existuj[cej aplikácii. Pri vkladaní predmetov do rozvrhu došlo zároveň k otestovaniu funkcionality spojenej s vizualizáciou dĺžky jednotlivých termínov skúšok a zobrazovania počtu spoločných študentov. Pri prepisovaní existujúceho rozvrhu do implementovanej aplikácie bolo zároveň vytvorených niekoľko verzií rozvrhu a otestované prepínanie medzi nimi. Prerekvizitou vytvorenia a editácie rozvrhov je vloženie potrebných vstupných dát. Algoritmus na spracovanie tabuľky s požiadavkami vyučujúcich bol testovaný na požiadavkách pre výuku pre letný a zimný semester rokov 2014, 2015, 2016 a letný semester 2017 a na požiadavkách na rozvrh skúšok pre roky 2016 a 2017. Pri testovaní bola odhalená a odstránena chyba, ktorá viedla k zacykleniu algoritmu, čo sa z pohľadu užívateľa prejavilo zamrznutím celej aplikácie.

Testovanie exportu dát prebiehalo s využitím už zmieňovaného rozvrhu skúšok pre letný semester akademického roku 2015/2016. V rámci testovania exportu boli vyskúšané rôzne kombinácie nastavenia zaškrtávacích tlačítok pre voliteľné zahrnutie jednotlivých typov exportovaných dát do výstupného súboru. Počas testovania sa ukázalo, že v prípade vyexportovania viac ako jednej tabuľky s počtami študentov na oboroch sa výstup nezobrazuje správne. Táto chyba bola odstránená.

Ďalšou z funkcií, ktoré bolo potrebné dôkladne otestovať, je import požiadaviek vyučujúcich z histórie požiadaviek jedného predmetu do požiadaviek na aktuálny rozvrh. Ako testovacia sada boli opäť použité tabuľky požiadaviek z rokov 2014, 2015 a 2016. V rámci testovania bola overená funkčnosť importu medzi všetkými typmi požiadaviek (výuka – výuka, výuka – skúšky, skúšky – skúšky). Pri testovaní neboli odhalené žiadne chyby.

Testovanie funkčnosti spojenej so zoznamami zapísaných študentov prebiehalo s využitím umelo vytvorenej sady 16 HTML súborov odpovedajúcich vybraným bakalárskym predmetom. Formát testovacieho súboru je určený stránkou, ktorú si môže vyučujúci zobraziť v informačnom systéme a ktorá obsahuje zoznam študentov zapísaných v určitom predmete.

Každý súbor v rámci testovacej sady obsahoval zoznam zapísaných študentov pre jeden predmet. Mená študentov boli vygenerované generátorom mien[1]. Následne boli títo študenti

[1] http://www.jmenaprijmeni.cz/generator-jmen-online/1610-generator-ceskych-jmen-a-prijmeni-pro-kluka

náhodne rozdelení do jednotlivých predmetov. Na tejto testovacej sade bol otestovaný skript `inputs.py` (popísaný v prílohe A), ktorý všetky súbory spojil do jedného súboru vo formáte JSON. Tento súbor bol následne vložený do aplikácie a uložený do databázy. Vďaka tomu mohla byť overená správna činnosť funkcie, ktorá pre daný zoznam predmetov zobrazí informácie o spoločných študentoch.

Klientska časť aplikácie bola testovaná v prehliadačoch Google Chrome verzie 57 a Mozilla Firefox verzie 53. Obidva prehliadače zobrazujú aplikáciu mierne odlišne, ale nejedná sa o rozdiely, ktoré by mali vplyv na funkčnosť a použiteľnosť aplikácie.

Pri práci s aplikáciou sa používanie skriptu `inputs.py` na konverziu formátu vstupných dát ukázalo ako nepraktické a preto bol nakoniec tento skript integrovaný do serverovej časti aplikácie, vďaka čomu je možné do aplikácie priamo vkladať súbory získané z informačného systému fakulty.

Serverová časť aplikácie bola v rámci testovania nasadená na fakultný server *knot09*, kde bol každý potrebný nástroj nainštalovaný v podstatne staršej verzii, než verzia použitá pri vývoji. V dôsledku toho bolo potrebné prepnúť kód do tzv. *striktného režimu*, ktorý poskytuje lepšiu kompatibilitu, ale zakazuje použitie vybraných konštrukcií jazyka JavaScript [18]. Pre správnu funkčnosť v strikntom režime bolo potrebné niektoré časti aplikácie upraviť. Pri inštalácii bolo odhalených niekoľko chýb v súboroch `.gitignore`, ktoré viedli k tomu, že niektoré potrebné súbory neboli v repozitári. Tieto chyby boli opravené.

Kapitola 7

Záver

V tejto práci bola detailne rozobratá problematika plánovania výuky a skúšok na *Fakultě informačních technologií Vysokého učení technického v Brně*. Vychádzajúc zo súčasného stavu plánovania na *Fakultě informačních technologií Vysokého učení technického v Brně*, bol navrhnutý a implementovaný nový systém pre plánovanie rozvrhov.

V porovnaní s pôvodným, existujúcim riešením obsahuje implementovaná aplikácia pokročilý systém správy vstupných dát. Súčasťou aplikácie je skript na automatické získanie informácií zo študijných plánov. Ostatné informácie potrebné pri plánovaní rozvrhov je možné do aplikácie jednoduchým spôsobom vložiť. Špeciálna pozornosť bola venovaná tabuľkám s požiadavkami vyučujúcich, pre ktoré bola implementovaná stránka s možnosťou ich editovania, farebného vyznačovania a následného exportu. Naviac aplikácia poskytuje podporu kombinovania aktuálnych požiadaviek s požiadavkami zadanými v predošlých akademických rokoch.

Prostredie pre editáciu rozvrhu poskytuje rovnakú sadu funkcií, ako pôvodná aplikácia. Okrem toho bolo implementované poloautomatické prideľovanie miestností, v rámci ktorého aplikácia automaticky rozdelí naplánované udalosti do dostupných miestností, ale užívateľ musí manuálne skontrolovať, že rozdelenie odpovedá všetkým kritériam. Novou funkciou je aj možnosť vytvárať viacero verzií každého rozvrhu. Pri editácii rozvrhu je v porovnaní s pôvodnou aplikáciou naviac dostupný pohľad na kolízie jedného predmetu v rámci časového okna ± 2 dní.

Nová aplikácia obsahuje aj možnosť získavať informácie o konkrétnych študentoch zapísaných v jednotlivých predmetoch. Podpora tejto funkcie si vyžiadala implementáciu autentizácie s následným obmedzením práv neprihláseného užívateľa. Neprihlásený užívateľ nemá prístup k zoznamom zapísaných študentov a zároveň nemá právo ukladať vykonané zmeny v rozvrhu a vo vstupných dátach.

V rámci ďalšieho vývoja budú do aplikácie doimplementované automatické kontroly splnenia vybraných tvrdých kritérií, a to konkrétne kontrola, že vyučujúci nemá dva predmety paralelne, a kontrola, že obor nemá dva povinné, resp. povinne voliteľné predmety paralelne.

Literatúra

[1] Abdullah, S.; Turabieh, H.: On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems. *Information Sciences*, ročník 191, 2012: s. 146–168.

[2] About Node.js: *Introduction*. [online], [cit. 2017-05-13]. Dostupné z: https://nodejs.org/en/about/.

[3] Angular2: *Architecture overview*. [online], [cit. 2017-05-13]. Dostupné z: https://angular.io/docs/ts/latest/guide/architecture.html.

[4] Angular2: *jQuery Introduction*. [online], [cit. 2017-05-13]. Dostupné z: https://www.w3schools.com/jquery/jquery_intro.asp.

[5] Angular2: *Quickstart. Typescript*. [online], [cit. 2017-05-12]. Dostupné z: https://angular.io/docs/ts/latest/quickstart.html.

[6] Bos, B.: *CSS*. [online], [cit. 2017-05-07]. Dostupné z: https://www.w3.org/Style/CSS/.

[7] Burke, E. K.; Newall, J. P.: A multistage evolutionary algorithm for the timetable problem. *IEEE transactions on evolutionary computation*, ročník 3, č. 1, 1999: s. 63–74.

[8] Ceschia, S.; Di Gaspero, L.; Schaerf, A.: Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers & Operations Research*, ročník 39, č. 7, 2012: s. 1615–1624.

[9] Daskalaki, S.; Birbas, T.: Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research*, ročník 160, č. 1, 2005: s. 106–120.

[10] Fielding, R. T.: *Architectural styles and the design of network-based software architectures*. Dizertační práce, University of California, Irvine, 2000.

[11] Horký, A.: *Systém pro pokročilé plánování*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2014.
URL http://www.fit.vutbr.cz/study/DP/DP.php?id=17111

[12] *HTML5*. [online], [cit. 2017-05-07]. Dostupné z: https://www.w3.org/TR/html5/.

[13] *JavaScript*. [online], [cit. 2017-05-07]. Dostupné z: https://cs.wikipedia.org/wiki/JavaScript.

[14] Kubalcová, M.: *Porovnání programů pro plánování rozvrhů a zkoušek*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2011. URL http://www.fit.vutbr.cz/study/DP/BP.php?id=13230

[15] Lewis, R.; Paechter, B.; McCollum, B.; aj.: *Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition*. Cardiff Business School, 2007.

[16] *MongoDB*. [online], [cit. 2017-05-13]. Dostupné z: https://cs.wikipedia.org/wiki/MongoDB.

[17] Python (programovací jazyk). [online], [cit. 2017-05-19]. Dostupné z: https://sk.wikipedia.org/wiki/Python_(programovac%C3%AD_jazyk).

[18] Resig, J.: *ECMAScript 5 Strict Mode, JSON, and More*. [online], [cit. 2017-05-19]. Dostupné z: https://johnresig.com/blog/ecmascript-5-strict-mode-json-and-more/.

[19] REST API Tutorial: *HTTP Status Codes*. [online], [cit. 2017-05-07]. Dostupné z: http://www.restapitutorial.com/httpstatuscodes.html.

[20] Směrnice děkana FIT doplňující Studijní a zkušební řád VUT v Brně. 2014. URL http://www.fit.vutbr.cz/info/rd/2014/rd43-141015.pdf

[21] Studijní a zkušební řád VUT. 2011. URL http://www.fit.vutbr.cz/info/predpisy/szvut-2011-07-21.pdf

[22] *TypeScript*. [online], [cit. 2017-05-07]. Dostupné z: https://www.typescriptlang.org/.

[23] *What Is AngularJS?*. [online], [cit. 2017-05-12]. Dostupné z: https://docs.angularjs.org/guide/introduction.

[24] *What can you do with JavaScript*. [online], [cit. 2017-05-07]. Dostupné z: https://www.w3.org/community/webed/wiki/What_can_you_do_with_JavaScript.

Príloha A

Manuál k skriptu na predspracovanie vstupných dát

Pri spustení skriptu je možné použiť nasledovné parametre:

- `-o output.json` – Výstupom všetkých operácii, ktoré skript podporuje je súbor vo formáte JSON. Parametrom `-o` je možné špecifikovať názov tohto súboru. Predvolená hodnota je `output.json`
- `-t terminy.html` – Spracuje tabuľku s počtami študentov pre termíny skúšok uloženú v súbore `terminy.html`
- `-k kolize.html` – Spracuje tabuľku s počtami spoločných študentov (tabuľka kolízií) uloženú v súbore `kolize.html`
- `-b obory.html` – Spracuje tabuľku s počtami študentov na jednotlivých oboroch
- `-d adresar` – Spracuje adresár obsahujúci súbory vo formáte HTML obsahujúce zoznamy zapísaných študentov
- `-w, --web` – Z webových stránok fakulty získa informácie o študijnych programoch, ich študijných plánoch a vyučovaných predmetoch