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

INFORMAČNÍ SYSTÉM PRO PLÁNOVÁNÍ ROZVRHŮ

TIMETABLE PLANNING INFORMATION SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MAREK KUCHYNKA**
AUTHOR

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

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Kuchynka Marek, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Informační systémy a databáze
Název: **Informační systém pro plánování rozvrhů**
Timetable Planning Information System
Kategorie: Informační systémy
Zadání:

1. Seznamte se s problematikou plánování rozvrhů zkoušek a výuky na FIT VUT v Brně a s programovacími jazyky využitými v nástrojích pro jeho podporu.
2. Prostudujte dostupné programy pro podporu této činnosti a vstupy a výstupy plánování z předchozích semestrů.
3. Navrhněte nový program pro plánování rozvrhů výuky a zkoušek, který bude provádět automatické kontroly vybraných kritérií pro tvorbu rozvrhů při manuálním umisťování rozvrhových oken. Program umožní jak vytváření celého rozvrhu zaměstnancům fakulty, tak i následné zapojení studentů do vylepšování výsledného rozvrhu.
4. Implementujte navržené řešení.
5. Zhodnoťte dosažené výsledky a vytvořte stručný plakát prezentující výsledky práce.

Literatura:

- Dle doporučení vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2 a 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Dytrych Jaroslav, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 30. října 2020

Abstrakt

Práce se zabývá plánováním rozvrhů výuky a zkoušek na Fakultě informačních technologií Vysokého učení technického v Brně. Jejím cílem je navrhnout a vytvořit nový webový informační systém, který bude pomáhat s vytvářením nových rozvrhů a umožní do procesu plánování zapojit také studenty. Aplikace bude vytvořena na míru požadavkům fakulty, jejichž detailní analýza tvoří první část této práce. Druhá část se podrobně zabývá návrhem, implementací a testováním výsledné aplikace, která byla již využita při plánování skutečných rozvrhů.

Abstract

This work deals with the planning of teaching and exams schedules at the Faculty of Information Technology of the Brno University of Technology. The goal is to design and create a new web-based information system that will help to create new schedules and allow students to be involved in the planning process. The application will be tailored to the requirements of the faculty, whose detailed analysis forms the first part of this work. The second part deals in detail with the design, implementation and testing of the created application, which was already used in planning the actual schedules.

Klíčová slova

webová aplikace, informační systém, plánování rozvrhů, HTML, CSS, JavaScript, jQuery, PHP

Keywords

web application, information system, timetable planning, HTML, CSS, JavaScript, jQuery, PHP

Citace

KUCHYNKA, Marek. *Informační systém pro plánování rozvrhů*. Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Dytrych, Ph.D.

Informační systém pro plánování rozvrhů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Dytrycha, Ph.D., který mi zároveň poskytoval odborné rady v oblasti plánování rozvrhů. Další informace mi poskytla Bc. Alena Tesařová – autorka aplikace, na kterou tato práce navazuje. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Marek Kuchynka
8. května 2021

Poděkování

Rád bych na tomto místě poděkoval Ing. Jaroslavu Dytrychovi, Ph.D. za odborné vedení této práce, za pomoc a cenné rady, které mi v průběhu řešení poskytl. Také bych chtěl poděkovat Bc. Aleně Tesařové za její pomoc při integraci nové aplikace se stávajícím řešením.

Obsah

Kapitola 1

Úvod

Tato diplomová práce se věnuje tvorbě webové aplikace, která na Fakultě informačních technologií Vysokého učení technického v Brně přispěje k efektivnímu vytváření rozvrhů pro výuková i zkoušková období.

S plánováním rozvrhů se lze setkat na všech stupních škol, ovšem na vysoké škole se ukazuje jako náročný proces. Studenti totiž nemají jednotný rozvrh, každý si může dle svého studijního plánu zvolit různé volitelné předměty, dokonce některý předmět mohou mít zapsán studenti různých ročníků. Proto plánující osoba (rozvrhář) musí mít při tvoření rozvrhu na paměti velké množství různých kritérií, kterými jsou například dostupnost a kapacity učeben, povinnosti jednotlivých předmětů, počty studentů zapsaných do předmětů, dostupnost vyučujících, bezkoliznost různých druhů výuky či termínů zkoušek pro studenty a také omezení ze strany jiných fakult, které jsou zapojeny do výuky studentů na FIT.

Překvapivě se i v dnešní době lze stále setkat s univerzitami, které plánují rozvrhy ručně na papír, nástěnku, případně jim s tvorbou „pomáhá“ jednoduchý tabulkový editor. Zjednodušení do tohoto procesu mohou přinést počítače, a to hned ve dvou úrovních – plná automatizace či pouze jako „pomocník“ při plánování. Na fakultě informačních technologií *(FIT)* se využívá již několik let způsob poslední.

I přes existenci různých podpůrných programů je plánování na FIT stále zdlouhavý a náročný proces, který je nutné opakovat minimálně čtyřikrát ročně. Jeho náročnost vyplývá již z podstaty samotného problému plánování rozvrhů v kombinaci s konkrétními požadavky FIT, o nichž bude pojednávat kapitola 2. V této kapitole jsou také uvedena jednotlivé kritéria, pomocí kterých lze zjistit, zda je rozvrh realizovatelný, případně jej zhodnotit, do jaké míry je pro studenty a učitele vyhovující.

Existující aplikace, které byly vytvořeny převážně v rámci jiných bakalářských či diplomových prací, budou popsány v podkapitole 2.4. Většina z nich se v současnosti nevyužívá, jen práce A. Tesařové proces plánování na FIT značně zjednodušuje především z hlediska importu požadavků a předmětů i exportu hotových rozvrhů do informačního systému FIT.

Požadavky, které jsou kladeny na nově vytvářenou aplikaci, budou nastíněny v podkapitole 2.5 a na jejich základě budou popsány zvolené technologie pro implementaci v kapitole 3. Implementovaný nástroj má navazovat na již existující zavedené řešení a má rozvrháři poskytovat rozhraní pro samotné plánování rozvrhů – umisťování rozvrhových oken a využívání podpůrných nástrojů. Zároveň bude aplikace automaticky hlídat některá omezující kritéria.

Návrh výsledné aplikace v kapitole 4 navazuje na zvolené technologie a dříve popsané požadavky na nové řešení. Je také ovlivněn již existujícími aplikacemi, a to jak z hlediska

funkčního (spolupráce se stávající aplikací v podkapitole 4.1 a 4.2), tak z hlediska vzhledu (návrh uživatelského rozhraní v podkapitole 4.4).

V kapitole 5 jsou popsány zajímavé pasáže z průběhu implementace aplikace, která kromě zjednodušení plánovacího procesu pro rozvrháře nabízí také možnost zapojení studentů. V rámci sociálních sítí totiž na FIT probíhá připomínkové řízení k vytvářeným plánům výuky a zkouškového období. Aplikace umožňuje každému studentovi upravit rozvrh a pokud bude takto upravený rozvrh lepší než oficiálně představený, rozvrhář zváží jeho použití. Hlavně z toho důvodu byla vytvořena bezpečná a hlavně uživatelsky přívětivá a intuitivní aplikace.

Aplikace byla v průběhu vývoje řádně testována a na její vývoj dohlížel rozvrhář Ing. Jaroslav Dytrych, Ph.D. Podrobnosti z jejího testování jsou popsány v kapitole 6. O praktickém využití aplikace svědčí i fakt, že byla v současné době již třikrát využita při plánování rozvrhů na FIT. Jednalo se o rozvrh zkoušek na letní semestr 2020/2021, který byl zkomplikovaný situaci okolo pandemie COVID-19, a rozvrhy pro výuku v zimním i letním semestru 2021/2022. Podrobnější informace poskytne čtenáři kapitola 6.3.

Na závěr kapitola 7 shrnuje dosažené výsledky práce a pojednává o možnostech dalšího rozšíření funkcionality implementované aplikace.

Kapitola 2

Plánování rozvrhů

S rozvrhy se setkáváme v různých odvětvích lidské činnosti. Primárně pod tímto pojmem rozumíme rozvrhy školní výuky – základní, střední, vysoké a někdy i mateřské školy. Stejně tak se ovšem může jednat i o jiné časové plány v různých firmách – směnný provoz, plány školení, umístění zaměstnanců na pracoviště atd.

Obecně se jedná o problém přiřazení úloh na zdroje v čase tak, aby byl maximalizován celkový užitek, tedy hledáme optimální řešení z pohledu nákladů, času, lidských zdrojů apod [7]. Můžeme jej klasifikovat jako úlohu s omezujícími podmínkami *(angl. constraint satisfaction problem – CSP)*. Je dokázáno (např. [4]), že nalezení optimálního rozvrhu je NP-úplný problém, tedy neznáme deterministický algoritmus, který by nalezl optimální řešení v polynomiálním čase. Existují ovšem přístupy, kterými se v polynomiálním čase můžeme optimálnímu řešení alespoň přiblížit a považovat je za dostatečně dobré. Jedná se například o použití genetických algoritmů [7, 6].

V případě plánování rozvrhů na vysoké škole, kterým se bude zabývat právě tato práce, problém konkretizujeme:

- **úlohami** rozumíme zkoušky, přednášky, cvičení a jiné události, které je třeba naplánovat,
- **zdroje** chápeme jako místnosti, vyučující a studenty.

Požadovaným řešením je nalezení rozvrhu, který by splňoval všechna předem definovaná omezení. Rozlišujeme dva typy omezení:

- **Tvrdá (silná)** omezení musí být splněna vždy, aby byl rozvrh přípustný. Jsou většinou definována reálným světem (vyučující nemohou být na dvou místech zároveň, místnost má určitou kapacitu atd.), do kterého zasahují také konkrétní předpisy a požadavky univerzity a fakulty.
- **Měkká (slabá)** omezení vychází převážně z požadavků vyučujících na fakultě, případně studentů. Jejich splnění zajišťuje komfort pro obě skupiny a podle míry jejich splnění je možné ohodnotit kvalitu vytvořeného rozvrhu.

Aby bylo možné věnovat se konkrétnímu problému, kterým je plánování rozvrhu na Fakultě informačních technologií Vysokého učení technického v Brně (zkráceně FIT nebo FIT VUT), je třeba pochopit základní fakta ohledně organizace studia a analyzovat možnosti vylepšení plánování rozvrhů.

2.1 Organizace studia na FIT

Akademický rok se skládá ze dvou semestrů (letní a zimní, každý z nich o délce 13 týdnů) a dvou zkouškových období, která na tyto semestry navazují (každé o délce 5 týdnů, letní navazuje přímo na semestr, zimní začíná až po Novém roce). Pro každý akademický rok je vydán časový plán, který přesně stanovuje počáteční a koncová data těchto období [16].

Výuka během semestru může probíhat každý všední den od 7:00 do 20:50. Jednotlivé vyučovací hodiny začínají vždy přesně v celou hodinu a končí deset minut před začátkem další hodiny. První hodina je tedy od 7:00 do 7:50, druhá od 8:00 do 8:50 atd. Po každé hodině následuje 10 minut přestávka.

Na fakultě existují tři stupně studia – bakalářské (3 roky), magisterské (2 roky) a doktorské (4 roky). Jelikož plánování rozvrhů probíhá až na výjimky pouze pro první dva zmiňované stupně, doktorským studiem se práce nadále zabývat nebude.

Studenti jsou v prvních 2 letech bakalářského studia rozděleni na dvě přednáškové skupiny, což umožní rozdělit zátěž na místnosti do dvou různých oken a zároveň existuje možnost překrývání přednášek skupiny *a* s přednáškami nebo cvičeními skupiny *b* a opačně.

Předměty

Předměty jsou v jednotlivých studijních programech a jejich specializacích (dříve oborech) rozděleny do tří skupin.

Povinné předměty *(P)* je nutné absolvovat v konkrétním ročníku studia a jejich zvládnutí je podmínkou úspěšného dokončení studia.

Povinně volitelné předměty *(PV)* si může student typicky zapsat v libovolném ročníku a stačí úspěšně absolvovat alespoň jeden z každé skupiny povinně volitelných předmětů.

Volitelné předměty *(V)* si lze zapsat téměř v libovolném ročníku (v návaznosti na prerekvizity) a umožňují detailnější specializaci studia.

Dokončení studia není neúspěchem ve volitelném předmětu ohroženo, pokud se studentovi podaří celkově za celé studium získat potřebný počet kreditů *ECTS*[1]. Minimální počet získaných kreditů definuje Studijní a zkušební řád VUT [16] – je nutné získat minimálně 180 kreditů v bakalářském a 120 kreditů v magisterském programu.

Výuka se realizuje prostřednictvím klasických přednášek, demonstračních cvičení, na kterých vyučující demonstruje probíranou látku, numerických cvičení, jež se uplatňují hlavně v matematických předmětech, laboratorních cvičení, v nichž jsou studenti zapojeni do praktické výuky, a počítačových cvičení, v rámci kterých studenti pracují na školních počítačích v laboratořích Centra výpočetní techniky *(CVT)*.

Zkoušky jsou organizovány centrálně. Každý předmět má buď tři pevné termíny (řádný, 1. opravný a 2. opravný), nebo 5 variantních termínů. U každého předmětu si může vyučující také zvolit, zda chce organizovat předtermín, či nikoli. Většina zkoušek na FIT probíhá písemnou formou – studenti jsou rozesazeni do místností a všichni zároveň vyzkoušeni. Každý student má právo zúčastnit se jednoho řádného termínu zkoušky a maximálně dvou opravných termínů. Účast na předtermínu se počítá jako využití řádného termínu a v případě pěti variantních termínů se první dva typicky počítají jako řádné, další dva jako 1. opravné a poslední je považován za 2. opravný.

[1] European Credit Transfer and Accumulation System

Místnosti

Fakulta disponuje několika různě velkými přednáškovými místnostmi, které jsou primárně určeny pro přednášky, případně demonstrační nebo numerická cvičení. V těchto velkých místnostech jsou také během zkouškového období plánovány písemné zkoušky, u kterých je v závislosti na požadavcích vyučujících nutné zařídit rozesazení s jednou *(ob 1)* nebo dvěma *(ob 2)* volnými židlemi mezi studenty.

Tabulka 2.1 zobrazuje maximální kapacitu při jednotlivých rozesazeních studentů pro všechny přednáškové místnosti na FIT. Kromě těchto místností se mohou přednášky konat v učebnách jiných fakult, ovšem těmi se pro potřeby plánování rozvrhů na FIT není třeba detailněji zabývat.

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

Tabulka 2.1: Kapacita přednáškových místností FIT (zdroj: [17])

Přednáškové místnosti lze pro zvýšení kapacity propojit tak, že studenti v „připojených“ místnostech vidí a slyší přednášejícího z hlavní místnosti. Možné propojení a výsledné kapacity jsou zobrazeny v tabulce 2.2. Poslední kombinace (místnosti E104 + E105) je sice technicky realizovatelná, ale zatím se nevyužívá.

Kombinace	**Kapacita**
D105 + D0206 + D0207	544
D105 + D0206	454
D105 + D0207	390
D0206 + D0207	244
E112 + E104 + E105	300
E112 +E104	228
E104 + E105	*144*

Tabulka 2.2: Propojení přednáškových místností FIT (zdroj: [17])

Kromě uvedených přednáškových místností jsou na fakultě také specializované a počítačové laboratoře, do kterých je umisťována specifická výuka. Studenti jsou zde rozděleni do mnoha skupin a vyučující si sami na základě kvalifikovaného odhadu určí, kolik místností bude využito. Laboratoře mají ve většině případů kapacitu 20 studentů, ovšem kontrola kapacity v těchto místnostech není v plánovací aplikaci vyžadována. Typicky se totiž plánují na konec a varovná hlášení o nedostatečné kapacitě by rozvrh znepřehledňovala.

2.2 Průběh plánování na FIT

Během roku se k plánování rozvrhů přistupuje celkem 4x. Rozvrh výuky pro zimní semestr se vytváří většinou v průběhu měsíce dubna a pro letní semestr v průběhu měsíce května. Nevýhodou ovšem je, že v tomto období ještě neznáme všechna potřebná data (hlavně počty studentů nastupujících do prvního ročníku) a musíme tedy pracovat s odhady, které vznikly z dat z minulých let.

Rozvrh zkoušek za zimní semestr se vytváří v průběhu měsíce října a za letní semestr v průběhu měsíce března. Při tomto plánování již známe počet zapsaných studentů v předmětech, ovšem nikdy nevíme, kolik studentů získá zápočet, aby mohli být připuštěni ke zkoušce. Zároveň také není známo, jaká bude úspěšnost jednotlivých termínů zkoušky. Tato data je tedy opět nutné odhadnout z let minulých, což komplikují hlavně nové či upravené předměty.

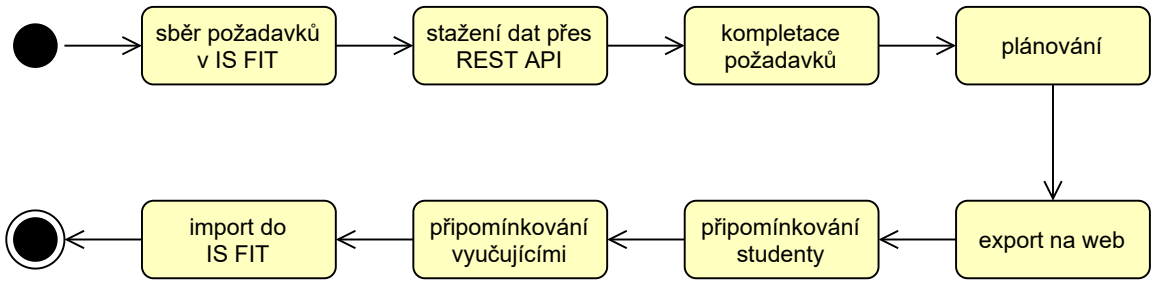
Na počátku plánování každého rozvrhu probíhá sběr dat o předmětech a preferencí vyučujících. Tyto preference pro výuku či zkoušky si musí vyučující a garanti předmětů vyplnit v informačním systému fakulty *(IS FIT – wis)*, odkud jsou poté společně s informacemi o předmětech staženy přes REST API[2] do stávající plánovací aplikace. Tento sběr dat trvá asi 2 týdny.

Některé požadavky se musí ručně doupravit. Jedná se převážně o kopírování požadavků z předešlého roku – učitelé s těmito požadavky automaticky počítají a znovu je do informačního systému nevyplňují. Tento proces kompletace dříve probíhal v tabulkovém editoru Excel, nyní jej lze jednodušeji vykonávat pomocí aplikace vytvořené pro tyto účely A. Tesařovou v rámci její bakalářské práce [17].

Po vytvoření první verze rozvrhu (v řádu několika dní) je tato verze zveřejněna studentům na sociálních sítích, kde mohou podávat návrhy na zlepšení. Pokud si chtějí vyzkoušet, jak by jejich řešení ovlivnilo výsledný rozvrh, mohou tak učinit v aplikaci, kterou pro plánování vytvořil J. Dytrych v roce 2015.

Pokud se během pár dní neobjeví zásadní problém mezi studenty, přejde rozvrh do připomínkového řízení vyučujícím, kteří mají na kontrolu cca 2 týdny. Pokud je všechno v pořádku, rozvrh se importuje do informačního systému fakulty, čímž se stává platným. Tímto importem mimo jiné proběhnou i rezervace místností a jsou upraveny stránky předmětů.

Celý průběh plánování přehledně zobrazuje graf na obrázku 2.1.



Obrázek 2.1: Průběh plánování rozvrhů na FIT

[2]Representational State Transfer Application Programming Interface

2.2.1 Plánování výuky

Vstupem pro plánování výuky je seznam předmětů, které je nutné v daném semestru naplánovat. U předmětů je třeba znát počet hodin pro jednotlivé typy výuky (jak dlouhé mají být přednášky, cvičení apod.). Pro optimální rozmístění výuky do místností je ideální znát počet zapsaných studentů, což bohužel většinou není možné (minimálně u prvních ročníků). K předmětům se samozřejmě váží studijní plány, podle kterých poznáme, zda se jedná o předmět povinný či volitelný, které specializace jej mají v jakém ročníku atd.

Velké množství informací, které je třeba do plánování zahrnout, přichází od samotných vyučujících jednotlivých předmětů. Může se jednat pouze o preference, ale také o tvrdá omezení. K tomu se přidávají požadavky z jiných fakult pro předměty, které jsou vyučované jinde – například na FEKT[3] či FSI[4].

Na základě výše popsaných vstupů definoval V. Čillo ve své diplomové práci [5] kritéria, podle kterých lze rozvrh validovat a hodnotit jeho kvalitu. Kritéria byla přeložena ze slovenštiny a aktualizována.

Tvrdá omezení:

- Bezkoliznost přednášek z povinných a povinně volitelných předmětů pro studenta, bezkoliznost přednášek z jakéhokoli předmětu pro vyučujícího
- Přednáška či cvičení naplánovené do místnosti s dostatečnou kapacitou nebo do více místností spojených za použití streamingu (u přednášky postačuje 70 % počtu zapsaných studentů[5])
- V jedné místnosti v konkrétním čase maximálně jedna přednáška nebo cvičení
- Čas je pro akci dostupný (omezení z jiných fakult, časové možnosti přednášejících)
- Dodržení pořadí pro přednášky a cvičení, kde je vyžadováno
- Dostatek času pro přesun mezi místnostmi, budovami a fakultami
- Předměty s více přednáškami v jednom týdnu je musí mít v různé dny (pokud vyučující nepožaduje opak)

Měkká omezení:

- Úplná bezkoliznost přednášek pro studenta
- Minimalizace počtu dlouhých bloků (několik přednášek za sebou, maximum 6 hodin)
- Umožnění pauzy na oběd
- Koncentrace výuky do určitých dní v týdnu
- Preference vyučujících
- Omezení ranní a večerní výuky (ideálně od 8:00 do 19:00)

[3] Fakulta elektrotechniky a komunikačních technologií VUT
[4] Fakulta strojního inženýrství VUT
[5] https://cuni.cz/UK-3204-version1-technickepodklady.doc

- Omezení páteční výuky (hlavně večerní)

Nelze si nevšimnout, že některá kritéria jsou téměř protichůdná (omezení délky souvislé výuky vs. koncentrace výuky do určitých dní). Je nutné si uvědomit, že se nikdy nepodaří vytvořit ideální rozvrh a vždy se tedy bude jednat o kompromis.

Při vytváření rozvrhu je velmi užitečným nástrojem tabulka kolizí. V ní se zobrazují počty společných studentů dvojic předmětů. Tyto kolize jsou předpočítány již ve fakultním informačním systému a dají se rozšířit na kolize pro všechny kombinace předmětů.

2.2.2 Plánování zkoušek

Informace vstupující do plánování zkoušek jsou velmi podobné jako u rozvrhů výuky popsaných dříve. Základem je seznam předmětů a k nim příslušný počet vypisovaných termínů zkoušek. Povinnosti předmětů a tedy i zkoušek získáváme ze studijních plánů.

Důležitým vstupem je seznam studentů, který je v tomto případě již vždy kompletní, může ovšem obsahovat i studenty, kteří ke zkoušce nebudou připuštěni (chybí jim zápočet, případně zanechali studia nebo nechtějí využít konkrétní termín zkoušky). Pro odhad počtu studentů, kteří se dostaví na jednotlivé termíny, se využívá tabulka termínů, jež je generována na základě předchozích let.

Zásadní informace pro plánování přichází od samotných vyučujících. Oproti požadavkům na výuku jsou zde definovány i požadované délky pro jednotlivé termíny zkoušky, rozesazení studentů v místnostech (vedle sebe, ob 1 nebo ob 2) a počet kol pro každý termín. Navíc zde učitelé definují také požadavky na minimální rozestup termínů od sebe z důvodu opravování.

Další omezení nastává s předměty, které jsou vyučovány jinými fakultami (FEKT, FSI, ...). Tato omezení jsou ve většině případů silná a nelze je při plánování opomenout.

Také pro plánování rozvrhu zkoušek stanovil V. Čillo ve své práci [5] kritéria pro validaci a hodnocení kvality výsledného rozvrhu a opět zde byla tato kritéria přeložena ze slovenského jazyka a aktualizována.

Tvrdá omezení:

- Bezkoliznost zkoušek (překrývající se časy – *hodinová kolize*)
- Zkouška naplánována do místností s dostatečnou kapacitou vzhledem na požadované rozesazení
- V jedné místnosti v konkrétním čase maximálně jedna zkouška (pokud vyučující neurčí jinak – např. u české a anglické varianty předmětu)
- Dodržení požadavků na počet kol, délku zkoušky a časové možnosti zkoušejících
- Dodržení termínů pevně daných (omezení z jiných fakult)
- Dostatek času na opravení písemek (mezi jednotlivými termíny)
- Dostatek času pro přesun mezi místnostmi, budovami a fakultami

Měkká omezení:

- Student má maximálně jednu zkoušku během jednoho/dvou/tří dnů (*denní koloze, sousední kolize a kolize ob jeden den*)

- Minimalizace počtu místností
- Omezení zkoušek v pondělí před 9:00 a v pátek po 16:00 (dojíždějící studenti)
- Mezi zkouškami z povinných předmětů má student alespoň dva volné dny
- Preference zkoušejících

Opět platí, že výsledný rozvrh pravděpodobně nebude nikdy splňovat veškerá zde uvedená kritéria a bude tedy vždy kompromisem. I zde je zásadním pomocníkem při přípravě tabulka kolizí, která zobrazí, kolik studentů má zapsané předměty, jejichž zkoušky jsou naplánovány na jeden den, případně v rozmezí dvou dnů.

2.3 Plánovací metody

Veškerá kritéria a požadavky na rozvrh jsou již známé, zbývá objasnit samotný proces plánování. Ten může probíhat třemi různými způsoby – manuální plánování, plánování s asistencí počítače a automatizované plánování.

Manuální plánování

Manuální plánování za pomoci papíru a tužky, případně jiných nástrojů jako je nástěnka a lepící papírky či tabulkový editor typu Excel, je velmi náročné pro rozvrháře, který plánování realizuje. Je totiž nezbytně nutné, aby po každém vložení nového okna do rozvrhu zkontroloval, zda tím nevytvořil novou kolizi. Do plánování žádným způsobem nezasahuje počítač či jiný nástroj na automatizované hlídání kritérií.

Chyby se v tomto systému velmi těžko hledají, neexistuje jednoduchý způsob, jak ohodnotit kvalitu vytvářeného rozvrhu a dokonce výsledný rozvrh nemusí být vždy správně sestaven. I přes jeho náročnost se tento způsob stále na některých vysokých školách využívá (např. Mendelova univerzita v Brně [17]).

Plánování s asistencí počítače

Při plánování s asistencí počítače stále existuje osoba rozvrháře (případně více rozvrhářů), který tentokrát již nepoužívá papír a tužku, ale vytváří rozvrh v počítači pomocí aplikace k tomu určené. Počítač dokáže na základě předem zadaných kritérií vytvářený rozvrh kontrolovat a dávat rozvrháři zpětnou vazbu na jeho vstupy.

I když může být počítač do plánování zapojen v různých úrovních, stále tato metoda předpokládá, že je uživatel v plánování rozvrhů zběhlý a ví, co dělá. Na FIT VUT se tento způsob osvědčil nejvíce a na jeho základě vzniklo několik různých aplikací, které budou popsány dále v textu.

Automatizované plánování

Oproti předchozímu způsobu již není při plánování vyžadován zkušený uživatel. Rozvrhy se generují částečně nebo plně automaticky, je nutné pouze na počátku zadat seznam předmětů k naplánování a kritéria, která se musí dodržet. Jak již bylo uvedeno na začátku kapitoly, plánování rozvrhů je NP-úplný problém a tedy nelze očekávat, že se podaří vždy vytvořit optimální rozvrh. Po vygenerování rozvrhu je nutné jej zkontrolovat, což je náročné

již z toho důvodu, že kontrolující osoba nemusí přesně pochopit veškeré důvody pro jeho konkrétní podobu. Často je potřeba na závěr provést manuální zásahy.

Hlavní nevýhodou tohoto přístupu je potřeba mít přesně specifikovaná a detailní data, která ovšem aktuálně na fakultě nejsou k dispozici. Jejich absence je buď dlouhodobá (např. nikde nejsou uvedeny potřebné časy pro přesun mezi místnostmi) nebo krátkodobá (např. v době plánování rozvrhů neznáme počet zapsaných studentů v předmětech). Přesnost specifikace je také zásadní, jelikož počítač musí přesně určit, co každé z předložených kritérií znamená.

Autor měl možnost asistovat při skutečném plánování rozvrhů na FIT. Při něm se bere ohled například i na kolizi dvou vyučujících manželů – někteří chtějí vyučovat zároveň, aby mohli jet společně domů, jiní se musí doma vystřídat při hlídání potomka. Tyto požadavky jsou získávány od vyučujících v textové formě a nástroj, který by je dokázal sám správně zařadit a splnit, by byl nepředstavitelně náročný na vytvoření.

Další nevýhodou automatického plánování je nemožnost rozhodnutí o ignorování požadavku. Pokud vyučující zadá preference špatně (například dva požadavky se vzájemně vylučují), automat jejich opravu nezvládne bez interakce s daným vyučujícím. Často se tyto situace řeší telefonicky přímo při plánování rozvrhu a vyučující své požadavky vysvětlí a případně z některého sleví, pokud působí příliš velké komplikace.

2.4 Existující software

Pro plánování rozvrhů existuje nespočet různých programů. Komerční produkty jsou podrobně popsány v bakalářské práci M. Kubalcové [8] a A. Tesařové [17]. Ani jedna z autorek nenašla program, který by dokázal uspokojit potřeby fakulty. Dále se tedy budeme věnovat aplikacím, které vznikly přímo pro FIT.

Převážně se jedná o aplikace, které byly vytvořeny jako součást bakalářských či diplomových prací studentů FIT. Každá z nich je podstatnou součástí vývoje plánování na fakultě a analýza jejich kladů a záporů pomůže v návrhu aplikace nové.

Aleš Horký (2015)

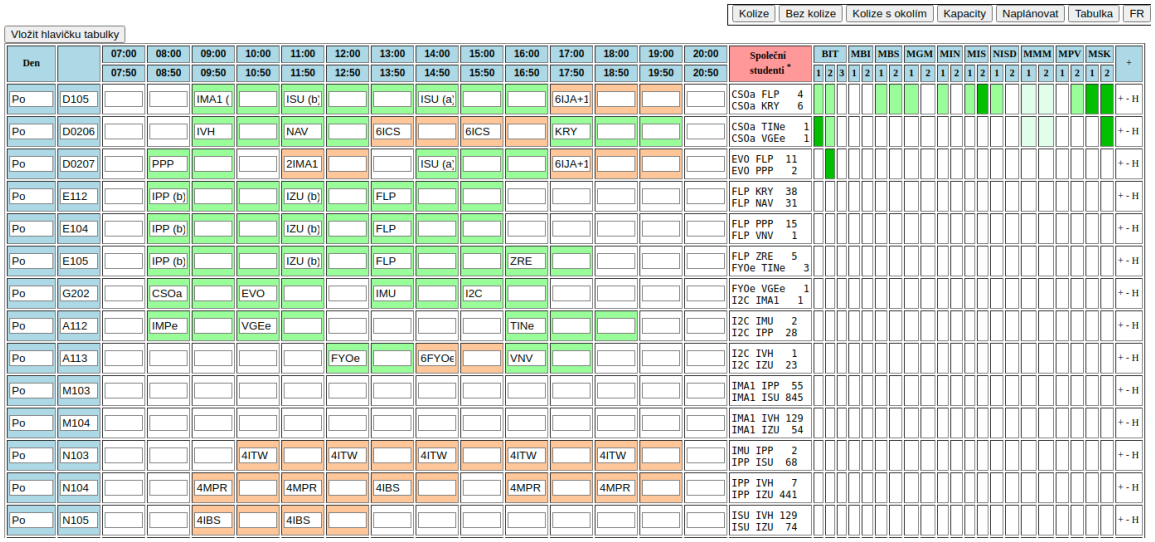
V rámci diplomové práce A. Horkého [6] vznikla jedna z prvních aplikací pro plánování rozvrhů přímo pro potřeby fakulty. Autor vytvořil pomocí kombinace genetického a heuristického algoritmu program, který na základě kritérií a požadavků ve vstupním souboru ve formátu XML[6] automaticky vygeneruje několik výsledných rozvrhů.

Nevýhodou tohoto přístupu je nemožnost editovat výsledný rozvrh, jelikož rozvrhář neví, které podmínky úprava poruší. Rovněž vstupní soubor s podmínkami je velmi náročné vytvořit, jelikož obsahuje přes 3000 řádků. Aplikace se z těchto důvodů v praxi nevyužívá.

Jaroslav Dytrych (2015)

Pro usnadnění manuální práce si J. Dytrych po nastoupení do funkce studijního poradce vytvořil jednoduchou webovou aplikaci v jazyce JavaScript, která je dodnes udržována a využívána pro plánování. Na obrázku 2.2 lze vidět náhled této aplikace se zobrazením rozvrhu výuky pro letní semestr 2020/2021.

[6] Extensible Markup Language



Obrázek 2.2: Původní aplikace vytvořená J. Dytrychem[7]

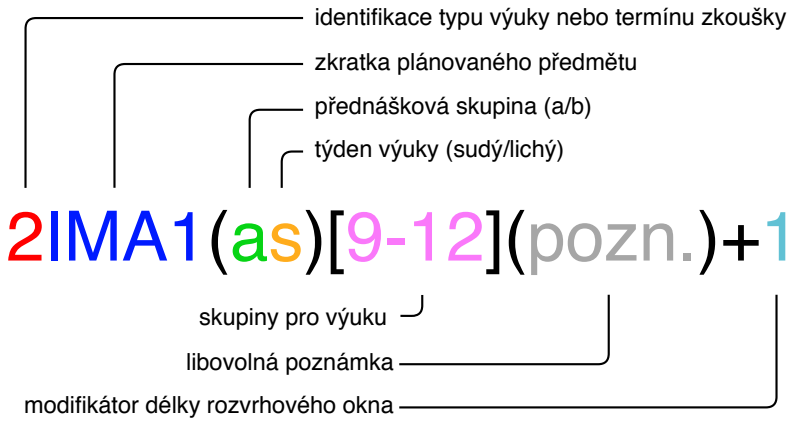
V pravém horním rohu se nachází jednoduché nástroje, které pomáhají rozvrháři při plánování:

- **Kolize** – pro dva zadané předměty vypíše počet společných studentů
- **Bez kolize** – vypíše seznam předmětů, které nemají společné studenty se zadaným předmětem
- **Kolize s okolím** – vypíše kolize pro dvojice předmětů ze zvoleného a předcházejícího i následujícího dne
- **Kapacity** – pro zadaný předmět vypíše počet studentů a uživatel je může zkoušet rozesadit do místností (nástroj vypočítává celkovou kapacitu, kapacitu *ob 1* i *ob 2*)
- **Naplánovat** – kontrola, zda jsou všechny předměty naplánované, případně zda není naplánováno více hodin nebo termínů
- **Tabulka** – vkládání nových řádků tabulky a kontroly
- **FR** – filtr rozvrhu pro konkrétní ročník

Hlavní částí aplikace je tabulka rozvrhu, ve které řádky reprezentují dny a místnosti, sloupce jednotlivé časové úseky. Tato část je upravována uživatelem – jsou do ní vkládány texty obsahující hlavně zkratku předmětu, ale také jiné informace o okně (viz další odstavec). V pravé části se automaticky generuje seznam dvojic předmětů se společnými studenty a graf povinností předmětů pro jednotlivé ročníky, který zobrazuje, kolik má který ročník specializace naplánovaných povinných či povinně volitelných předmětů.

J. Dytrych vytvořil syntaktická a sémantická pravidla pro text rozvrhových oken, která jsou nadále využívána všemi dalšími aplikacemi. Na obrázku 2.3 je struktura textu znázorněna. Kromě zkratky předmětu jsou veškeré údaje volitelné a mezi jednotlivými skupinami se může vyskytovat libovolný počet mezer.

[7] https://www.fit.vut.cz/study/advisor/20202021/priprava_LS/planovani/index.htm



Obrázek 2.3: Struktura textu rozvrhových oken zavedená J. Dytrychem

Použitelné typy výuky a termíny zkoušek ukazuje tabulka 2.3. Přednášková skupina *a* nebo *b* se uplatní v prvních dvou ročnících bakalářského studia (jak bylo zmíněno v kapitole 2.1) a pokud není uvedena, předpokládá se přednáška pro celý ročník. Podobné pravidlo platí i pro týdny výuky – pokud není specifikováno, že se výuka koná pouze sudý (*s*) nebo lichý (*l*) týden, koná se vždy.

V hranatých závorkách je možné definovat rozsah skupin, kterých se výukové okno týká (lze použít například pro rozdělení do místností). Na konci textu se může nacházet délkový modifikátor, který vždy začíná znakem + a následuje číslo určující délku, o kterou se má okno prodloužit vůči své normální délce. Pokud je ovšem nutné okno zkrátit, je třeba zadat „+-1“.

Znak	Význam - výuka	Význam - zkoušky
0	–	předtermín
1/nic	přednáška	1. termín
2	numerické cvičení	2. termín
3	laboratorní cvičení	3. termín
4	počítačové cvičení	4. termín
5	jiné cvičení	5. termín
6	demonstrační cvičení	–
!	omezení (další text není podstatný)	

Tabulka 2.3: Typy výuky a zkoušek

Aplikace neprovádí automatizované kontroly, pouze vypisuje rozvrháři počty společných studentů a poskytuje mu nástroje výše popsané. Konfigurace aplikace je složitá, jelikož se každoročně musí upravit předměty a počty studentů v nich přímo ve zdrojových souborech. Pro použití s novými specializacemi bylo nutné provést opravu povinností předmětů, které jsou v aplikaci také pevně zadefinovány. Program umožňuje jednoduchý export vytvořeného rozvrhu do souboru ve formátu CSV[8], ze kterého je možné rozvrh zpětně do aplikace načíst, což je jediná možnost sdílení. Pro sdílení termínů a rozvrhů se studenty slouží (kromě samotné rozvrhové tabulky) generované soubory ve formátu HTML pro každý ročník zvlášť

[8] Comma-Separated Values

dle vzoru studijního poradce M. Eysselta, který byl dříve za plánování rozvrhů na FIT zodpovědný (příklad takové tabulky lze vidět na obrázku 2.4). Vygenerované soubory je nutné umístit na veřejně přístupný web (v současnosti je jím webová stránka studijního poradenství[9]).

Časový plán zkoušek především **z povinných předmětů**

pro **1. ročník v IT-BC-3** na **FIT**

2018/**2019 - LETNÍ SEMESTR, 1BIA + 1BIB**

Předmět	*Půlsemestrální zkouška*	*Řádná Zkouška*	*První opravná zkouška*	*Druhá opravná zkouška*
IMA Matematická analýza	*stanoví učitel*	2019-**05-09** *12.00 – 14.50*	2019-**05-29** *12.00 – 13.50*	2019-**06-07** *12.00 – 13.50*
ISU Programování na strojové úrovni	*stanoví učitel*	2019-**05-06** *09.00 – 11.50*	2019-**05-21** *09.00 – 11.50*	2019-**06-03** *09.00 – 11.50*
IJC Jazyk C	*stanoví učitel*	2019-**05-07** *15.00 – 16.50*	2019-**05-27** *12.00 – 13.50*	2019-**06-04** *15.00 – 16.50*
ISJ Skriptovací jazyky	*stanoví učitel*	2019-**05-10** *13.00 – 15.50*	2019-**05-28** *14.00 – 15.50*	2019-**06-04** *08.00 – 09.50*
IOS Operační systémy	*stanoví učitel*	2019-**05-13** *12.00 – 16.50*	2019-**05-23** *13.00 – 15.50*	2019-**06-06** *13.00 – 15.50*
INC Návrh číslicových systémů	*stanoví učitel*	2019-**05-16** *08.00 – 09.50*	2019-**05-30** *09.00 – 10.50*	2019-**06-05** *13.00 – 14.50*
Některé **volitelné**	**předměty**	**hledejte**	**u 2BIT**	

Obrázek 2.4: Plán zkoušek v letním semestru 2018/2019 – 1. ročník bakalářského studia[10]

Vladimír Čillo (2017)

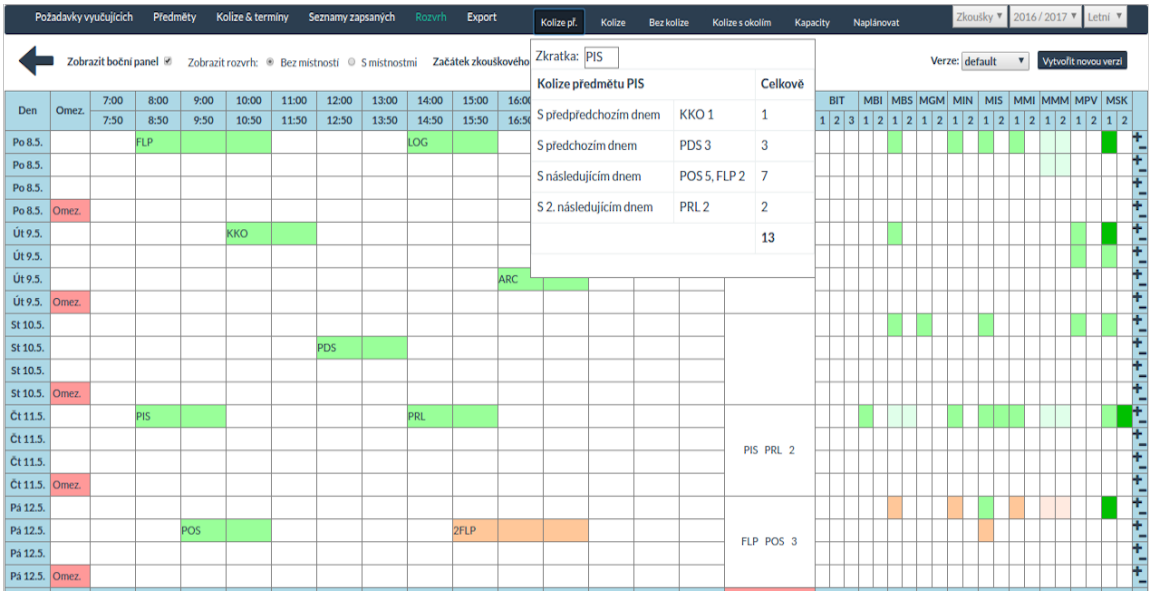
Pod vedením J. Dytrycha vytvořil později aplikaci pro plánování rozvrhů V. Čillo v rámci své diplomové práce [5]. Napsal ji převážně v jazyce JavaScript v kombinaci s různými knihovnami a frameworky, které mu umožnily využít JavaScript na straně klienta i serveru. Pro některé skripty použil jazyk Python a pro databázi zvolil MongoDB.

Při návrhu uživatelského rozhraní se držel původní a prověřené aplikace pana Dytrycha, jak lze vidět na obrázku 2.5. Do aplikace navíc zahrnul práci s požadavky učitelů, které původní aplikace vůbec nepodporovala. Jejich zpracování je prováděno externím skriptem ze souboru ve formátu HTML. Zjednodušil také práci se studijními plány a definicemi předmětů (ty stahuje z webu fakulty), kolizemi a počty studentů, které je možné nahrát pomocí externího skriptu nebo zadat ručně v aplikaci.

Během plánování dovoluje vytvořit několik verzí rozvrhu, vylepšil nástroje představené v původní aplikaci a přidal podporu pro zobrazení rozvrhu bez místností. Na závěr umožňuje aplikace také export rozvrhů ve formátu HTML pro prezentaci studentům. Sdílení rozpracovaných rozvrhů zde není nutné, jelikož aplikace byla plánována pro běh na serveru s jednou databází a více uživatelskými účty.

[9] https://www.fit.vut.cz/study/advisor/

[10] https://www.fit.vut.cz/study/advisor/20182019/zkouskyLS/1bitlet.htm



Obrázek 2.5: Aplikace vytvořená V. Čillem (zdroj: [5])

Aplikaci nelze jednoduše nainstalovat, jelikož vyžaduje již zastaralou verzi Node.js[11] (lze vyřešit např. použitím Docker[12]) a správa uživatelů v ní je nevyhovující. Při importování dat ze souborů aplikace v případě chyby spadne (zamrzne) bez jakéhokoli chybového hlášení, a proto je dohledání chyb ve zdrojových souborech velmi náročné. Při plánování rozvrhů na FIT se v současné době program pana Čilla nevyužívá a není tedy nadále udržován.

Alena Tesařová (2019)

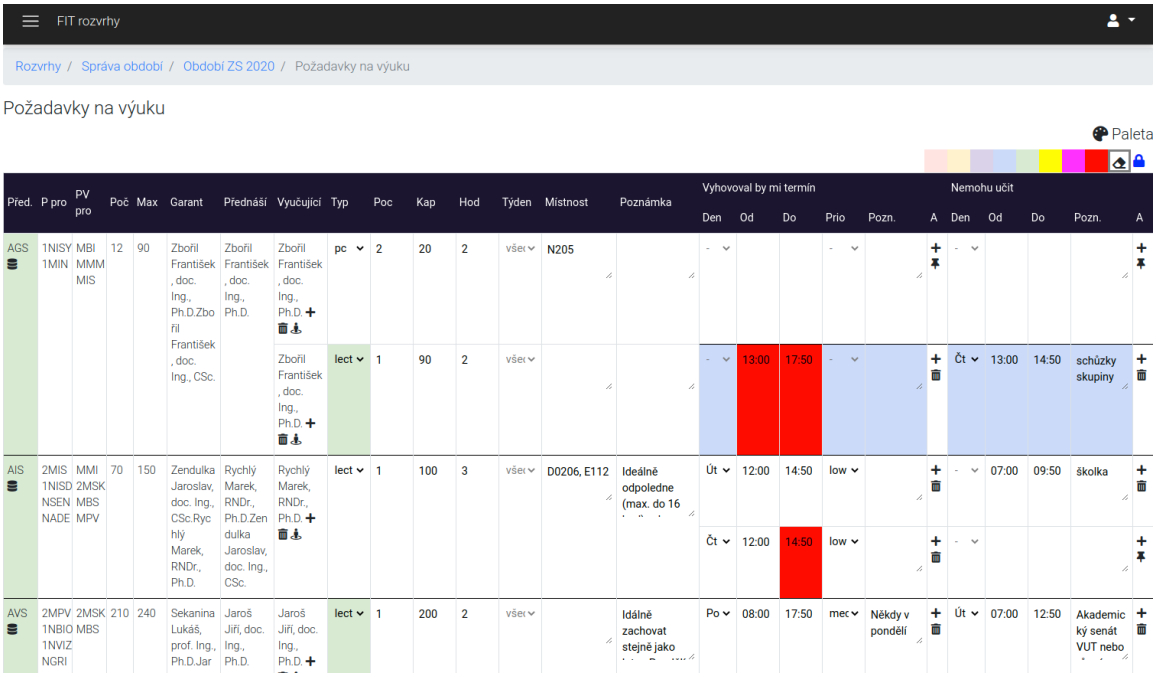
Na práci V. Čilla navázala ve své bakalářské práci A. Tesařová [17], přičemž si původně kladla za cíl vytvořit pevné zázemí pro plánování rozvrhů, ovšem neimplementovat plánování samotné (což měl za úkol vytvořit D. Vosáhlo, viz dále). Její aplikace je napsána v jazyce PHP za použití frameworku Nette a na klientské straně využívá JavaScript. Pro ukládání dat zvolila databázi MySQL.

Základním účelem aplikace je zpracování a úprava požadavků vyučujících (viz obrázek 2.6), předmětů, studijních plánů, místností a dalších informací získaných pomocí REST API z informačního systému FIT *(wis)*. Tyto informace jsou logicky uloženy do databáze a připraveny pro použití jinou plánovací aplikací, která bude pracovat na stejné databázi a ukládat do ní i vytvořené rozvrhy.

Výsledné rozvrhy poté dokáže její aplikace vyexportovat pro prezentaci na webu ve formátu HTML a nově také přímo umožňuje export do IS FIT *(wis)*, čímž odpadá manuální přepisování rozvrhových a zkouškových oken pro rezervaci místností.

A. Tesařová nad rámec zadání vytvořila v aplikaci také možnost přímého plánování rozvrhů zkoušek i výuky. V něm umožňuje provádět základní kontroly rozvrhu a dává k dispozici stejné nástroje, které byly popsány dříve. Bohužel, jak již bylo zmíněno, jedná se o základní kontroly a nejsou vždy úplně spolehlivé. Dlouhá odezva aplikace komplikuje její použití při rychlém ukládání rozvrhových oken a díky ní někdy vznikají v databázi neva-

[11] https://nodejs.org/
[12] https://www.docker.com/



Obrázek 2.6: Aplikace vytvořená A. Tesařovou – požadavky[13]

lidní rozvrhy. I přes tato omezení se aplikace okamžitě začala k plánování rozvrhů používat, osvědčila se a stala se společně s aplikací M. Morese aktuálně využívanou plánovací aplikací.

Daniel Vosáhlo (2019)

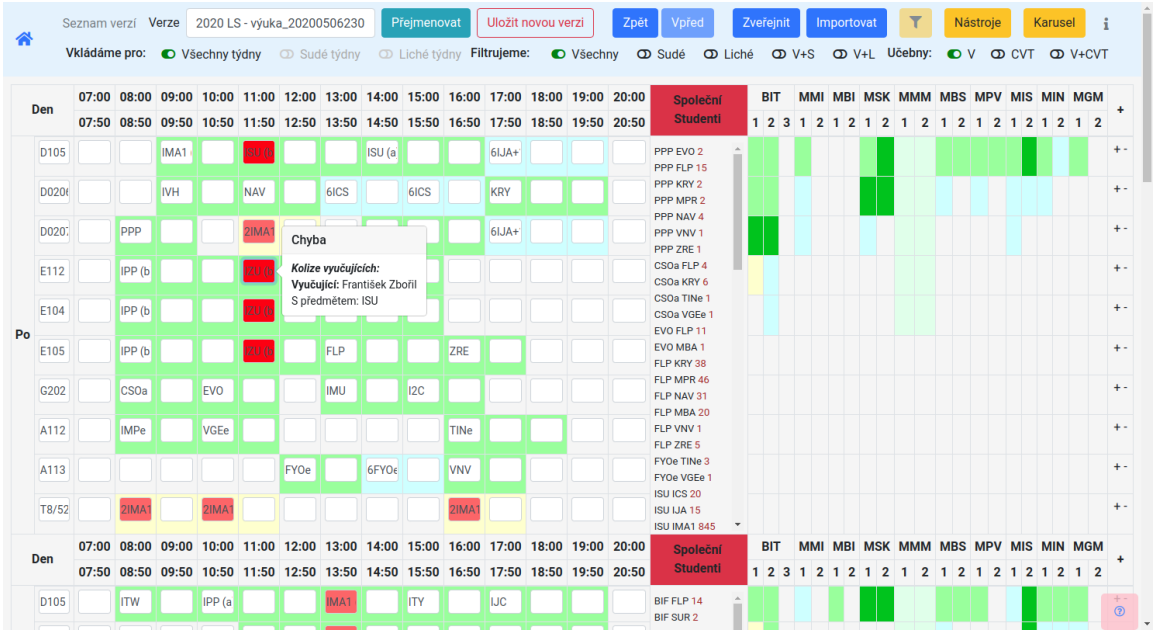
V rámci své diplomové práce [18] D. Vosáhlo spolupracoval na vytvoření plánovací aplikace s A. Tesařovou. Jeho úkolem bylo implementovat pouze samotné plánovací rozhraní, které bude využívat databázi vytvořenou právě zmíněnou kolegyní. Pan Vosáhlo využil k tvorbě aplikace kombinaci jazyků PHP a JavaScript rozšířený o knihovnu jQuery.

Výsledná aplikace jako první zavádí automatizované kontroly požadavků od vyučujících a zkoušejících. Rozvrhář při jejím používání tedy již nemusí mít všechny požadavky v hlavě či vše kontrolovat na 2. monitoru, jelikož ho aplikace na porušení ihned upozorní. Informace o porušeném kritériu jsou vypsány po najetí myší nad rozvrhové okno, což je uživatelsky velmi přívětivé řešení. Detail aplikace i s varováními lze vidět na obrázku 2.7.

Dalším prvenstvím aplikace jsou vícenásobné kolize. Do této doby byly známé pouze počty společných studentů ve dvou předmětech. Nyní jsou tyto počty dostupné pro všechny kombinace předmětů, a to přes nástroj **kolize**. Tato data jsou předpočítána externím skriptem před plánováním a uložena do samostatných tabulek v databázi. Doba běhu skriptu může být díky nevhodně zvolenému algoritmu v řádu desítek hodin.

Poslední větší novinkou v aplikaci je nástroj pojmenovaný *karusel*. Jedná se o sadu pomocných pohledů, které se mohou zobrazovat v pravé části. Z původní aplikace tu zůstává graf povinností, nově je zde možné zobrazit kolize s okolím (počet společných studentů pro předměty dva dny dopředu a dozadu), kolize ve dni a zátěž ve dni, případně u zkoušek zátěž

[13] http://minerva3.fit.vutbr.cz/xtesar36/ostra/BPgit/rozvrhy3/www/obdobi/96/pozvyuka



Obrázek 2.7: Aplikace vytvořená D. Vosáhlem[14]

v týdnu. Některé tyto nástroje ovšem nejsou dopracované nebo nejsou vůbec implementované.

Martin Mores (2020)

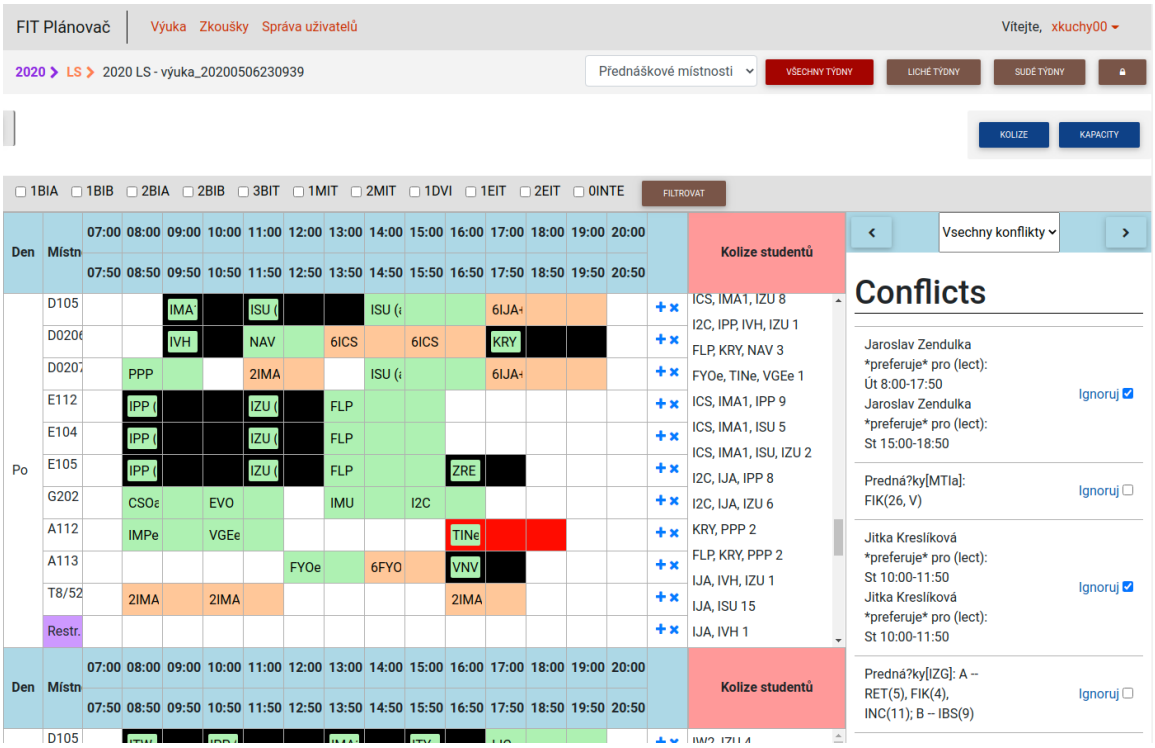
Poslední prací a zároveň aplikací, která na téma plánování rozvrhů na FIT vznikla, je diplomová práce M. Morese [9]. Pro implementaci zvolil jazyk Java, framework Spring a technologii Java Server Pages.

Aplikace vykonává automatické kontroly rozvrhu vůči požadavkům vyučujících podobně jako aplikace předchozí a zároveň řeší některé její nedostatky (např. možnost ignorování chyb – kolizí). Naopak jí chybí funkcionalita *zpět* a *vpřed*, která je v předchozích dvou aplikacích implementována a umožňuje uživateli vracet se při úpravách rozvrhu o několik kroků zpátky. V pravé části rozvrhu umožňuje kromě grafu povinností zobrazit také tabulku rozestupů mezi zkouškami, naplánovaných termínů, n-tice předmětů se společnými studenty a veškeré konflikty rozvrhu. Tyto nástroje jsou dle názoru současného rozvrháře J. Dytrycha užitečné, avšak jejich umístění v pravé části není ideální vzhledem k omezenému prostoru.

Při podrobnějším pohledu na obrázek 2.8, přestavující aplikaci M. Morese při plánování rozvrhů pro výuku v letním semestru, lze problém s místem pochopit. Rovněž si nelze nevšimnout chybějících překladů nebo problémů s českými znaky – pravděpodobně způsobených zvolenou technologií.

Zásadním přínosem práce M. Morese byla úprava, resp. vytvoření nového algoritmu pro výpočet společných studentů pro kombinace více předmětů. Jak bylo popsáno výše, touto problematikou se zabýval i D. Vosáhlo, jehož řešení bylo časově i výpočetně náročné (i desítky hodin). Oproti tomu nové řešení mělo za cíl dosáhnout času 15 až 20 minut [9, str. 37], ovšem skutečné časy se pohybují v řádek stovek milisekund [9, str. 50]. Tento fakt

[14] http://minerva3.fit.vutbr.cz/xvosah00/Ostra/rozvrh/110



Obrázek 2.8: Aplikace vytvořená M. Moresem[15]

omezil nutnost ukládání předpočítaných kolizí do databáze, jelikož mohou být vypočítány kdykoli za běhu.

Vytvořená aplikace se potýká s chybami, které je ovšem velmi obtížné opravit vzhledem ke specifickým technologiím, které na FIT nejsou běžné a tedy je nutné se s opravami spoléhat na jejího autora. Chybějící funkcionalita verzování (pohyb *zpět* a *vpřed*) způsobuje problémy, jelikož je vždy zobrazena poslední uložená verze rozvrhu, nikoli aktuálně zvolená pomocí jiné aplikace. Přesto se aplikace při plánování aktuálně využívá (v kombinaci s aplikací A. Tesařové), jelikož se jedná o dosud nejlepší plánovací aplikaci vytvořenou na FIT.

2.5 Požadavky na nové řešení

Při analýze požadavků na nové řešení jsou brány do úvahy již existující aplikace a připomínky k nim, které byly zjištěny v průběhu jejich vývoje a užívání. Zpětnou vazbu k nim poskytl J. Dytrych z pozice rozvrháře a vedoucího většiny prací, v rámci kterých tyto aplikace vznikly.

Udržovatelnost a spolehlivost

Aplikace pro vytváření rozvrhů se na FIT využívá 4x za rok, jak již bylo výše popsáno. Z důvodu takto malého vytížení nelze všechny nedostatky zjistit a opravit v rámci této práce (nehledě na možnou změnu požadavků), proto je velký důraz kladen na možnost zapojení do vývoje a oprav aplikace i jiných autorů.

[15] https://minerva3.fit.vutbr.cz:8443/schedule/lecture/70/110/allweeks/lecturerooms/view

Veškeré zdrojové kódy by tedy měly být přehledně strukturované a důkladně okomentované. Vhodné by bylo využít pro implementaci jazyk, který je obecně známý a podporovaný s co nejmenším počtem nestandardních rozšíření či knihoven. Zvolené technologie by rovněž měly zajistit možnost dlouhodobého provozování s možností bezproblémového přechodu na novější verze či přenosu na jiná běhová prostředí.

Výsledný produkt bude nasazen přímo na fakultní server *minerva3* a pracovat se s ním bude převážně v areálu fakulty. Jelikož se bude jednat o důležitou aplikaci pro chod fakulty, je třeba omezit závislosti na externích knihovnách, které se mohou kdykoli stát nedostupné. I za cenu větší velikosti zdrojových kódů je tedy vhodné replikovat veškeré využité knihovny lokálně.

Pokud v aplikaci nastane chyba, je žádoucí ji uživateli okamžitě oznámit s veškerými detaily. Jelikož bude aplikace většinou využívána osobami zasvěcenými do problematiky programování, bude pro ně lepší přesně vědět, co se stalo (místo hlášení typu „Hups, něco se nepovedlo“).

Spolupráce se stávajícím řešením

Cílem této práce není vytvoření kompletního systému pro proces plánování rozvrhů výuky a zkoušek. Aplikace A. Tesařové totiž dostatečně kvalitně pokrývá import dat z informačního systému fakulty a kompletaci požadavků vyučujících. Stejně tak dokonale zvládá exportování výsledných rozvrhů do stránek v jazyce HTML i do samotného IS FIT.

Úkolem je navrhnout a implementovat plánovací aplikaci, tedy část, ve které bude uživatel vytvářet a upravovat rozvrh. Data budou sdílena přes společnou databázi s ostatními plánovacími nástroji. Bude tedy nezbytně nutné zajistit vzájemnou kompatibilitu rozvrhů vytvořených v nové aplikaci s ostatními programy (minimálně těmi využívanými) a opačně – pokud bude rozvrh vytvořen v jiné aplikaci, musí existovat možnost jej zobrazit v nově vytvářené.

Zapojení studentů do plánování

Stejně jako předchozí, i nově vytvářená aplikace by měla umožnit zapojení studentů do plánování rozvrhů, a to konkrétně v připomínkovací fázi, kdy je zveřejněn návrh rozvrhu výuky nebo zkoušek na daný semestr. Každý student by zde měl mít možnost zkusit si upravit aktuálně zveřejněnou verzi a zjistit, zda je tato úprava validní a ve výsledku lepší než původní. Studenti samozřejmě nemají znalosti rozvrháře a neznají kritéria, která musí rozvrh splňovat, proto je místo nich bude transparentně hlídat samotná aplikace (viz dále).

V této souvislosti se musí také vyřešit uživatelská práva v systému a umožnění přihlašování studentů do aplikace. Není totiž žádoucí zveřejnit interní informace z fakulty (například důvody nedostupnosti vyučujících) komukoli s přístupem k aplikaci.

Úroveň automatizace a hlídaná kritéria

Jak již bylo uvedeno v podkapitole 2.3, úplná automatizace plánování rozvrhů pro FIT není vhodná. Proto bude aplikace vznikat jako pomůcka pro rozvrháře, ovšem zároveň bude snaha o co největší zjednodušení veškerých úkonů a snížení nároků na něj. Může totiž nastat situace, kdy rozvrhář nebude v čase plánování k dispozici (například pro nemoc) a jeho agendu bude mít na starost méně zkušený kolega.

Aplikace tedy musí být přehledná a jednoduchá na ovládání, umět kontrolovat základní požadavky a upozorňovat uživatele na jejich porušení:

- Plánování jedné akce do místnosti na konkrétní čas
- Hlídání požadované kapacity místností
- Dodržení požadavků na délku akcí
- Možnost studenta navštívit všechny přednášky z povinných a povinně volitelných předmětů (v případě dvou přednáškových skupin alespoň z jedné skupiny)
- Plánování dvou řádných termínů zkoušky z povinných předmětů v jeden den
- Bezkoliznost pro vyučující a zkoušející
- Dostupnost vyučujících a zkoušejících
- Preference vyučujících a zkoušejících dle priority

Odezva aplikace

Obecně u všech aplikací platí, že by měly být co možná nejrychlejší. Zvláště pak aplikace, se kterými uživatel pracuje v reálném čase. Aplikace pro plánování rozvrhů není výjimkou, přesto dosavadní aplikace dosahují (i díky zvoleným technologiím) vysokých prodlev. Nejlépe lze jejich odezvu změřit při načítání již kompletního vytvořeného rozvrhu, kde se pohybují v řádu desítek sekund pro každé načtení stránky. Naměřené časy pro vybrané rozvrhy ukazuje tabulka 2.4.

Rozvrh	**Úkon**	**Aplikace A. Tesařové**	**Aplikace M. Morese**
Výuka ZS 2019/2020	zobrazení	18 s	28 s
	editace	3 s	2 s
	„zpět“	2,5 s	—
Výuka ZS 2020/2021	zobrazení	20 s	27 s
	editace	3 s	1,5 s
	„zpět“	3 s	—
Zkoušky ZS 2019/2020	zobrazení bez místností	9 s	13 s
	editace bez místností	1,5 s	—
	„zpět“ bez místností	1,5 s	—
	zobrazení s místnostmi	21 s	38 s
	editace s místnostmi	2 s	—
	„zpět“ s místnostmi	2 s	—
Zkoušky ZS 2020/2021	zobrazení bez místností	56 s	10 s
	editace bez místností	4 s	—
	„zpět“ bez místností	2,5 s	—
	zobrazení s místnostmi	80 s	42 s
	editace s místnostmi	8 s	—
	„zpět“ s místnostmi	9 s	—

Tabulka 2.4: Měření rychlosti jednotlivých úkonů v používaných aplikacích

Kromě času pro načtení stránky je také klíčový čas mezi úpravou rozvrhového okna, jeho skutečným uložením do databáze, odpovědí od serveru a překreslením pohledu. Pokud

je tento čas příliš dlouhý, může se stát, že během tohoto procesu uživatel vyvolá další změny a zobrazený rozvrh se může ocitnout v nekonzistentním stavu oproti databázi nebo je do databáze uložen rozvrh nevalidní (např. překryv dvou oken), což se stává u aplikace A. Tesařové. V případě nekonzistence pohledu vůči databázi lze situaci napravit jednoduchým obnovením stránky, což ovšem může znamenat dlouhé načítání (viz tabulka 2.4). Pokud nastane druhý případ a aplikace umožní uložení nevalidního rozvrhu, je později problém s jeho znovu-zobrazením (hlavně napříč aplikacemi).

Posledním kritériem, které bylo měřeno, byl čas při procházení podverzí rozvrhu (tlačítka zpět/vpřed). Při plánování je někdy rozvrhář zaveden do slepé uličky a potřebuje se o několik změn vrátit zpět. Bohužel, tuto funkcionalitu podporuje pouze aplikace A. Tesařové a to ne zcela. Problém nastává při zobrazení rozvrhu zkoušek s místnostmi, kde aplikace neumí dynamicky přidat a odebrat více oken zároveň (např. odstranit celý předmět z 5 místností).

Při měření časů v tabulce 2.4 bylo zjištěno, že aplikace M. Morese hlásí chybu při úpravě rozvrhu zkoušek, tedy bylo možné změřit pouze časy editace rozvrhů výuky. Rovněž kvůli chybějící funkcionalitě *zpět* a *před* nebylo možné změřit délku trvání této operace.

Nová aplikace by tedy měla mít co možná nejmenší odezvu na dotazy technologií AJAX pro ukládání oken, rozvrhy by měla načítat v rámci jednotek sekund a také se musí umět vyrovnat se špatně uloženými rozvrhy z jiných aplikací – ty by měla umět opravit do konzistentního stavu. Aplikace pana Morese se s takovými stavy bohužel vyrovnat neumí, oproti tomu program A. Tesařové nemá se zobrazením nevalidního rozvrhu problém, ovšem uživatele o tomto stavu neinformuje.

Kapitola 3

Využité technologie

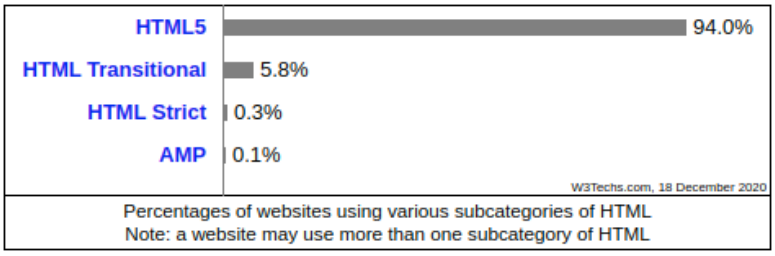
V této kapitole bude uveden výčet technologií, které budou použity při implementaci aplikace pro plánování rozvrhů. Při jejich výběru byl brán ohled na jejich stálost, dlouhodobou udržitelnost a všeobecnou znalost z důvodu snazší budoucí údržby systému.

Obecně při tvorbě jakékoli aplikace je možno zvolit z několika variant – terminálová aplikace, desktopová (případně mobilní) aplikace nebo webová aplikace. Právě poslední ze jmenovaných zažívá v poslední době největší rozvoj, jelikož uživatel není nucen k instalaci speciálního programu lokálně na svůj počítač, stačí mu běžný webový prohlížeč. Přesto jsou tyto aplikace schopny plně nahradit komplexní „běžné“ programy.

Vzhledem k charakteru vyvíjené aplikace a již existujícím implementacím, které se na FIT aktuálně využívají, byla zvolena **webová aplikace** jako nejvhodnější. Jelikož bude nutné při plánování co nejrychleji reagovat na uživatelské vstupy, bude zapotřebí využít tzv. **tlustého klienta** – implementuje na straně uživatele (prohlížeče) kromě prostého zobrazení dat také větší část logiky aplikace.

HTML

Hlavním stavebním prvkem všech webových aplikací je značkovací jazyk HTML *(Hypertext Markup Language)* – dle [13] jej využívá 91,2 % webových stránek. Jazyk vznikl v roce 1990 odvozením od SGML *(Standard Generalized Markup Language)* a jeho tvůrcem je Tim Berners-Lee [12, str. 17]. Aktuálně je udržován organizací W3C a poslední verze 5.2 vyšla 14. prosince 2017 [19]. Přehled využívaných verzí (variant) ukazuje graf na obrázku 3.1 (využitá verze byla rozpoznána na základě *doctype* na začátku souboru – nezaručuje 100% kompatibilitu s uvedenou verzí).



Obrázek 3.1: Aktuálně využívané varianty HTML na webu (zdroj: [13])

Nejdůležitější součástí HTML pro vývoj webových aplikací jsou formulářové prvky. Jako jediné totiž umožňují přímou interakci s uživatelem jako je výběr z možností, zadávání textu či klikání na tlačítka.

CSS

Pro definici vzhledu webové stránky (tedy i webové aplikace) slouží jazyk CSS *(Cascading Style Sheets)*. Autorem je Håkon Wium Lie a jeho počátky sahají do roku 1994. Od roku 1999 je ve vývoji poslední aktuálně používaná verze CSS3, nyní v poslední specifikaci ze dne 22. prosince 2020.

CSS slouží pro popis způsobu zobrazení jednotlivých prvků dokumentů napsaných v jazycích HTML, XHTML či XML, a to pomocí jednotlivých pravidel [12, str. 145-146]. Stylopis (soubor pravidel v jazyce CSS) lze k dokumentu připojit externě, zapsat do hlavičky nebo přímo ke konkrétním prvkům.

JavaScript

Skriptovací jazyk JavaScript byl vyvíjen Brendanem Eichem od roku 1995 pro společnost Netscape. Jazyk je objektově orientovaný a dynamicky typovaný [20, str. 12]. Udržován je společností *Ecma International*, která jej vydává pod jménem ECMAScript. Současná verze je ECMAScript 2020, jinak známá také jako verze 11 [2].

Kód v jazyce JavaScript je vykonáván přímo v klientském prohlížeči, kam je umístěn po stažení ze serveru. Obsahují jej totiž buď samotné webové stránky přímo v HTML, nebo odkazují na zdrojové soubory, které si klient musí stáhnout zvlášť.

V posledních letech se díky projektu Node.js[1] dostává JavaScript také na serverovou stranu a dokonce je ho možné využít i pro vývoj mobilních či desktopových aplikací (například projekt React Native[2]).

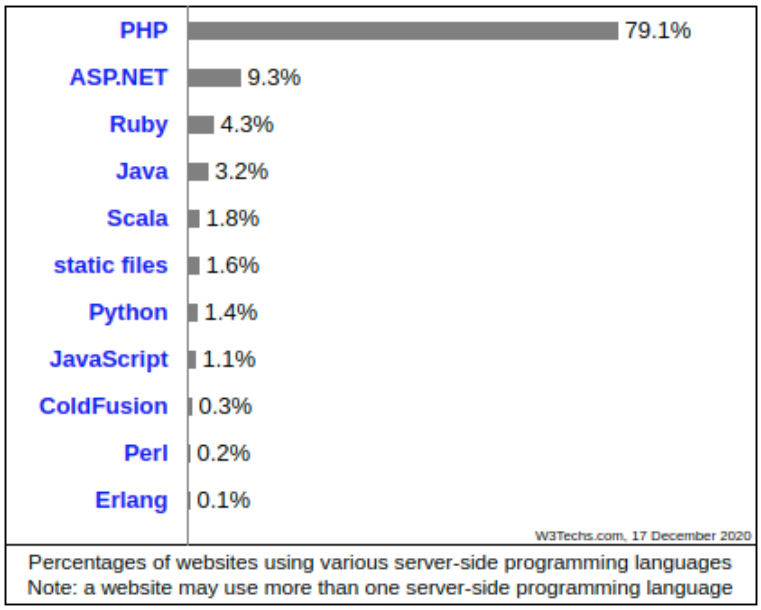
V práci bude kromě samotného JavaScriptu použita také knihovna **jQuery** v aktuální verzi 3.5.1 zjednodušující manipulaci s dokumentem v jazyce HTML, práci s událostmi od uživatele, animace a volání AJAX [10]. AJAX *(Asynchronous JavaScript And XML)* umožňuje asynchronní zpracování požadavků na server, např. dynamické načtení dat ze serveru bez nutnosti znovunačtení celé stránky. Knihovnu začal vytvářet v roce 2005 John Resig a v současné době je udržována nadací *OpenJS Foundation* [10].

PHP

Dominantním jazykem využívaným na webových serverech je jazyk PHP (původně *Personal Home Page*, nyní *PHP: Hypertext Preprocessor)*. Toto prvenství dokazuje graf na obrázku 3.2. Autorem PHP je Rasmus Lerdorf, který jej vytvořil v roce 1995. Jeho podoba byla ovlivněna převážně jazyky C (do kterého se také překládá), Perl a Java [14, str. 15-19]. Udržován je skupinou *The PHP Group* a aktuální verzí je 8.0, která oficiálně vyšla 26. listopadu 2020 [3].

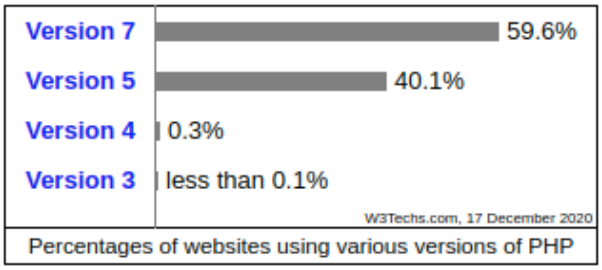
Nespornou výhodou PHP je jeho přenositelnost – je to multiplatformní skriptovací jazyk. I když je určený převážně pro tvorbu dynamických webových stránek, lze ho bez omezení využít i při tvorbě konzolových či desktopových aplikací. Mezi jeho přednosti patří jednoduchost a univerzálnost. Podporuje funkcionální i objektové paradigma, umožňuje práci se soubory nebo připojení k různým typům databází [11, str. 20-22].

[1] https://nodejs.org/
[2] https://reactnative.dev/



Obrázek 3.2: Používané technologie na straně serveru (zdroj: [13])

V rámci této práce budou zdrojové kódy psány pro verzi 7 (konkrétně poslední minoritní verzi – 7.4). Jak lze vyčíst z grafu na obrázku 3.3, jedná se prozatím o dominantní verzi PHP na webu. Verze 8 je v době vzniku této práce úplnou novinkou a zatím není jisté, jak bude později rozšířena a zda vůbec. Samozřejmě v práci nebudou využívány zastaralé konstrukce, o kterých se ví, že budou brzy vyřazeny, ale zároveň bude snaha o zachování zpětné kompatibility alespoň s verzí 7.2, jelikož tato verze je aktuálně v provozu na budoucím produkčním serveru *minerva3*.

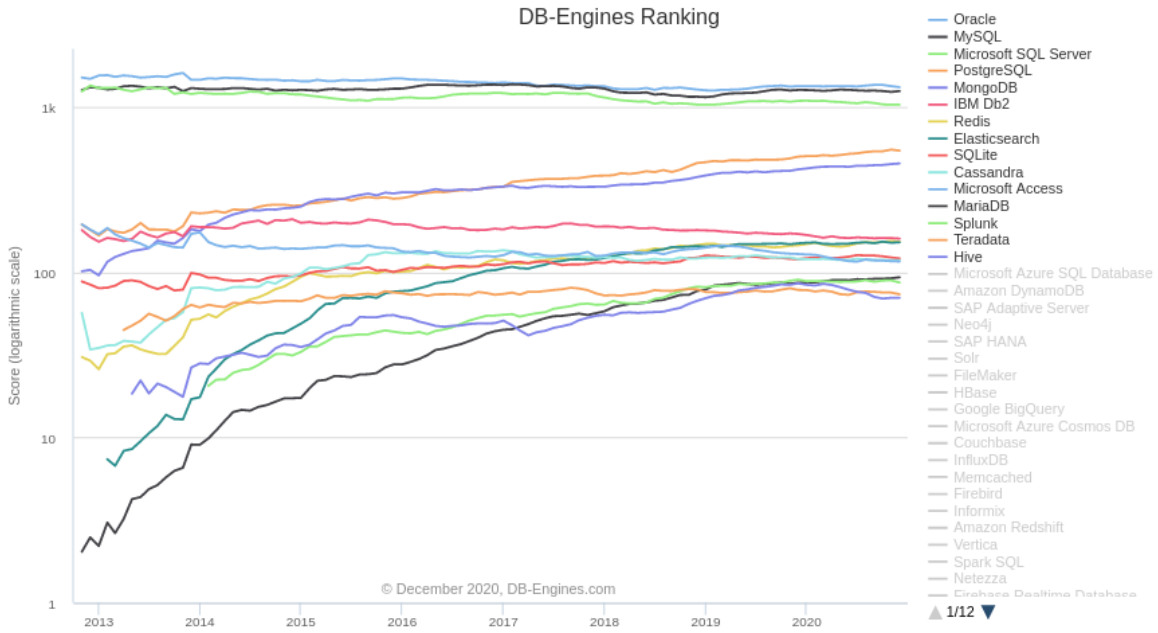


Obrázek 3.3: Aktuálně využívané verze PHP na webu (zdroj: [13])

MySQL

Mezi tři nejpopulárnější systémy pro řízení báze dat (SŘBD) patří MySQL. Dokazuje to graf na obrázku 3.4. MySQL byl vytvořen v roce 1995 firmou *MySQL AB*, nyní jej udržuje společnost *Oracle*. Jedná se o multiplatformní otevřený relační databázový systém a jeho aktuální verze je 8.0.22.

Z důvodu návaznosti na aplikaci A. Tesařové, jež využívá pro ukládání dat právě MySQL, byl výběr SŘBD jednoznačný. Stávající produkční databáze, která se nachází na serveru *minerva3*, je ve verzi 5.7 a pro většinu tabulek je využito úložiště **InnoDB**, které



Obrázek 3.4: Popularita jednotlivých SŘBD (zdroj: [15])

podporuje (oproti MyISAM) transakce, uzamykání při souběžném přístupu na úrovni řádků a obsahuje podporu pro cizí klíče.

Databáze MySQL je možné spravovat přes příkazový řádek nebo pomocí webového nástroje **phpMyAdmin**. Jedná se o nástroj napsaný v jazyce PHP, jehož původním autorem je Tobias Ratschiller. Nyní je vyvíjený jako otevřený software a aktuální verze je 5.0.4 [1].

Kapitola 4

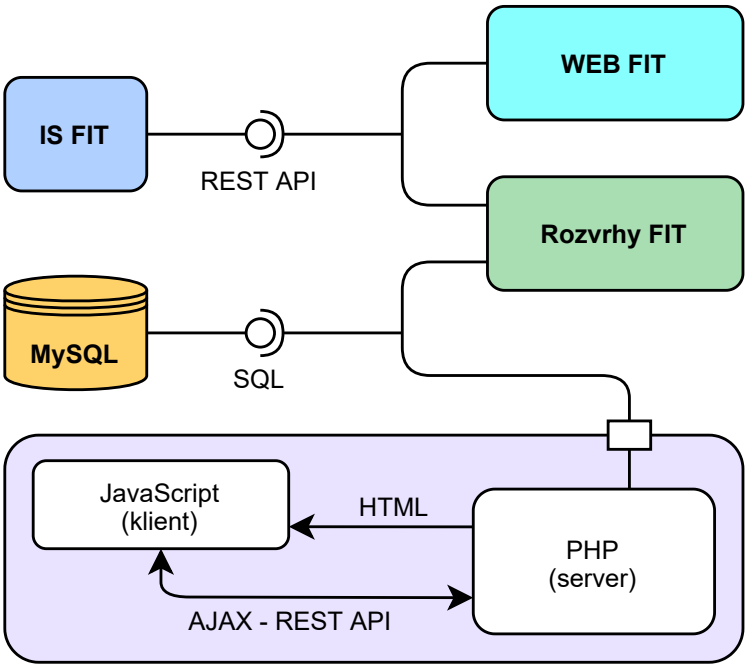
Návrh aplikace

Po analýze procesu plánování v podkapitole 2.2, analýze požadavků uvedených v podkapitole 2.5 a výběru vhodných technologií v kapitole 3 je nyní možné přistoupit k samotnému návrhu aplikace.

4.1 Spolupráce aplikací

Jak již bylo popsáno dříve, vytvářená aplikace má při plánování rozvrhů spolupracovat s již existující aplikací pro kompletaci požadavků a export do IS FIT. Tato spolupráce bude probíhat ve formě využívání totožné databáze, která je již implementována a zprovozněna na serveru *minerva3*.

Diagram komponent na obrázku 4.1 zobrazuje, jak bude vytvářená aplikace (fialová) spolupracovat s již existující aplikací A. Tesařové (zelená). Zároveň naznačuje vnitřní strukturu aplikace, tedy její rozdělení na klientskou a serverovou část.



Obrázek 4.1: Diagram komponent výsledného systému plánování

K databázi bude aplikace přistupovat ze své serverové části psané v jazyce PHP. Klientská část aplikace bude využívat data načtená z databáze, která budou vložena přímo do souborů v jazyce HTML a pokud bude nutné získat další data nebo provádět změny v databázi za běhu, využije se k tomu princip asynchronního volání (AJAX). Tyto požadavky příjme serverová část aplikace a zpracuje je s využitím databáze. Jako odpověď vrací buď data, pokud je cílem jejich získání, nebo pouze informaci o úspěchu či chybě, pokud se jedná o modifikaci dat.

4.2 Databáze

Schéma aktuálně používané databáze lze vidět na obrázku 4.2. Schéma je převzato z dokumentace ke stávající aplikaci, kterou vytvořila A. Tesařová, a bylo poupraveno na základě aktuálního stavu. Zároveň z něj byly vyjmuty nepoužívané tabulky, které se do databáze dostaly v rámci rozšíření z jiných aplikací (např. od D. Vosáhla nebo M. Morese).

Ve schématu jsou šedě označené tabulky, které slouží pro import a export dat mezi stávající aplikací a IS FIT, případně slouží pro export do formátu HTML pro prezentaci studentům. Jak již bylo uvedeno dříve, tato funkcionalita není v nově vytvářené aplikaci vyžadována a jelikož pro plánování neobsahují užitečná data, nebudou tyto tabulky vůbec použity.

Žlutě označené tabulky uchovávají užitečná data, která jsou nutná i pro plánování rozvrhů – jsou to informace o obdobích (semestrech), studijních programech a oborech, předmětech včetně informací o učitelích v nich působících, požadavcích na výuku i zkoušky a informace o místnostech. Z těchto tabulek bude nová aplikace data pouze číst.

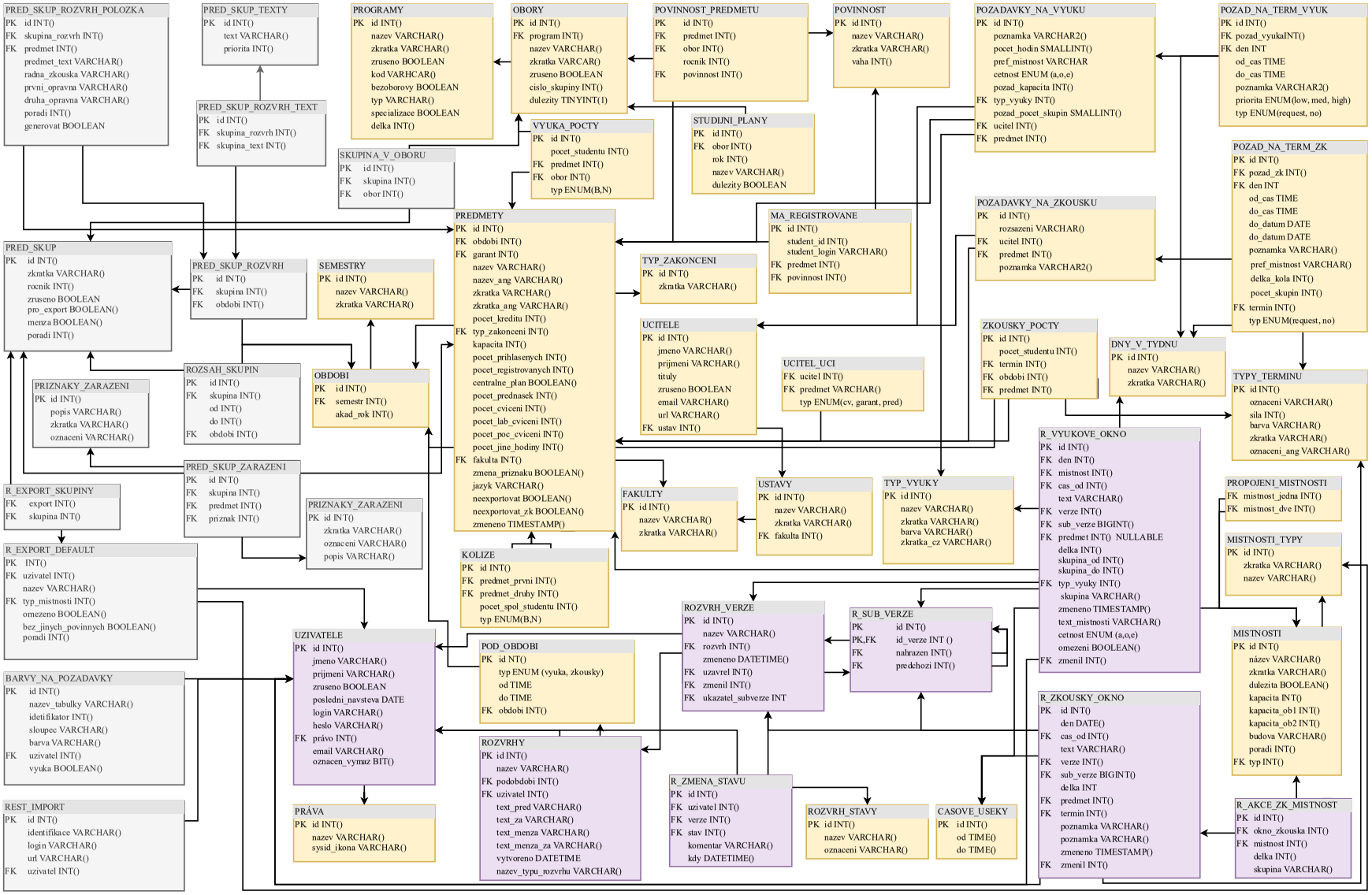
Fialové tabulky slouží pro ukládání rozvrhů, jejich verzí a podverzí, výukových a zkouškových oken a jedna tabulka je zde pro uchování uživatelů. Tyto tabulky bude nově vytvářená aplikace využívat nejvíce, a to i pro zápis.

Ukládání rozvrhů a rozvrhových oken

Princip ukládání rozvrhů v databázi je poměrně komplikovaný, proto zde bude podrobně vysvětlen. Nejdříve se zaměříme na ukládání rozvrhů samotných, poté na ukládání oken výuky a nakonec bude tento postup rozšířen o ukládání oken zkouškových.

Informace o **rozvrzích** jsou ukládány do dvou tabulek. Databázová tabulka `ROZVRHY` obsahuje informace o názvu rozvrhu, jeho typu a období, do kterého patří (zkoušky nebo výuka, semestr, rok – dostupné z tabulek `POD_OBDOBI` a `OBDOBI`). Standardně existuje pouze jeden rozvrh v daném (pod)období, ovšem nic nebrání uživateli vytvořit jich více. Každý takový rozvrh může mít několik verzí, které jsou uchovávány v tabulce `ROZVRH_VERZE`. Tato verze rozvrhu k sobě váže přes podverze rozvrhová okna, což bude popsáno dále. Díky těmto verzím rozvrhů lze do plánování zapojit studenty, kteří si udělají kopii zveřejněné verze a upravují ji jako svoji vlastní verzi (o tomto postupu bude pojednáno dále v případech užití). V dalším textu bude pod označením „rozvrh“ myšleno spojení tabulek rozvrhů a verzí rozvrhů, aby nedocházelo k matení čtenáře (tedy např. „uživatelé vidí zveřejněné rozvrhy“ místo „verze rozvrhů“).

Podverze *(subverze)* jsou identifikovány číslem verze rozvrhu a číslem podverze (pro každý rozvrh od 1), uloženy jsou v tabulce `R_SUB_VERZE`. Pokud některé okno nahrazuje v rozvrhu jiné (uživatel změní text, délku či typ), v tabulce podverzí bude tento fakt uložen v rámci dvou polí:



Obrázek 4.2: Schéma aktuálně používané databáze (zdroj: dokumentace k [17], upraveno)

- `nahrazen` bude nastaveno u původní podverze a bude odkazovat na nově vytvářenou podverzi,
- `predchozi` bude nastaveno u nové podverze a bude odkazovat na původní podverzi.

Každá změna ve **výukovém rozvrhovém okně** vyvolá zápis tohoto nového okna do tabulky rozvrhových oken (`R_VYUKOVE_OKNO`) a zároveň vytvoří novou podverzi rozvrhu. To vše pro zachování historie a možnost navracení změn ve vytvářeném rozvrhu. Pokud uživatel některé okno pouze vymaže (nenahradí jej okamžitě jiným), bude do databáze uloženo znovu celé dle stejného postupu (včetně nové podverze a „nahrazení"), ovšem pole `den` a `cas` budou nastavena na `NULL`.

Zjednodušeně lze konstatovat, že pokud chceme zobrazit rozvrh s jeho posledními úpravami (poslední podverzí), je nutné z databáze vybrat pouze taková okna, jež náleží do podverzí, které nemají nastaveno `nahrazen` (resp. nachází se zde `NULL`) a zároveň se nejedná o okna označující vymazání (nesmí mít nastaven `den` a `cas` na `NULL`). Pokud však uživatele zajímá nějaká konkrétní podverze rozvrhu, musíme vybrat podverze se stejným nebo nižším číslem a zároveň podmínku u pole `nahrazen` změnit – může se zde kromě `NULL` nacházet také číslo nahrazující podverze, které musí být zároveň vyšší než aktuálně zvolené.

Každé výukové okno náleží právě do jedné místnosti, proto je místnost odkazována přímo z tabulky těchto oken. Pokud se jedná o místnost, která v databázi není (typicky místnosti na jiné fakultě, případně zatím skutečnou místnost neznáme a použijeme zástupný název), je pole `mistnost` nastaveno na `NULL` a je třeba se řídit polem `text_mistnosti`.

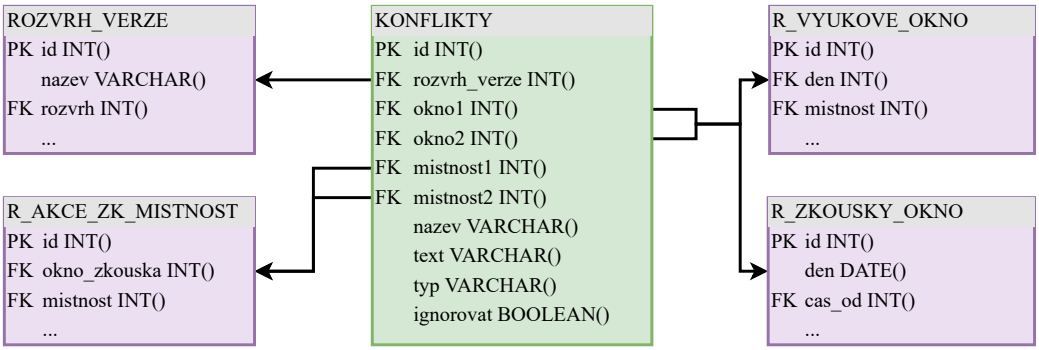
U **rozvrhu zkoušek** je situace komplikovanější právě kvůli plánování místností – běžně se jedna zkouška z jednoho předmětu v jeden čas odehrává ve více různých místnostech. Extrémním příkladem může být rozvrh pro zimní zkouškové období roku 2020/2021, kdy kvůli pandemii COVID-19 byly využity pro jednu zkoušku veškeré dostupné místnosti na fakultě. Bylo by tedy nevýhodné ukládat zkouškové okno zvlášť pro každou místnost, pro ukládání místností proto existuje tabulka `R_AKCE_ZK_MISTNOST` odkazující do tabulky zkouškových oken (`R_ZKOUSKY_OKNO`). Výhodou tohoto řešení je nativní podpora pro zobrazení zkoušek bez místností.

Další pozoruhodnou vlastností při plánování zkoušek je možnost změnit délku zkoušky v konkrétní místnosti bez ovlivnění délky zkouškového okna a ostatních místností. Tato vlastnost vznikla pro studenty se specifickými potřebami ve studiu, mezi které může patřit právě prodloužení času na vypracování zkoušky. Proto se v tabulce `R_AKCE_ZK_MISTNOST` nachází položka `delka`, která je na počátku stejná jako délka zkouškového okna, ovšem lze ji nezávisle měnit.

Rozšíření

Databáze bude rozšířena o jednu tabulku, která bude uchovávat informace o vzniklých kolizích či varováních. Uchovávat se tato data budou v databázi proto, aby nebylo nutné je při každém zobrazení rozvrhu (případně přepnutí na starší verzi) přepočítávat znovu. Druhým důvodem je zamýšlená funkcionalita možnosti ignorování jednotlivých konfliktů a tuto volbu je nutné si zapamatovat. Pro podobný účel vytvořil již M. Mores tabulku jménem `KONFLIKT_IGNOROVAT`, avšak její formát není pro novou aplikaci vyhovující. Schéma nové tabulky lze vidět na obrázku 4.3 včetně jejích závislostí na již existujících tabulkách.

Tabulka `KONFLIKTY` bude obsahovat odkaz na rozvrhové okno, kterého se konflikt týká (atribut `okno1`). Pokud bude konflikt mezi dvěma okny (např. stejný vyučující), bude druhé okno adresováno druhým cizím klíčem `okno2`. Podobně budou odkazovány také místnosti

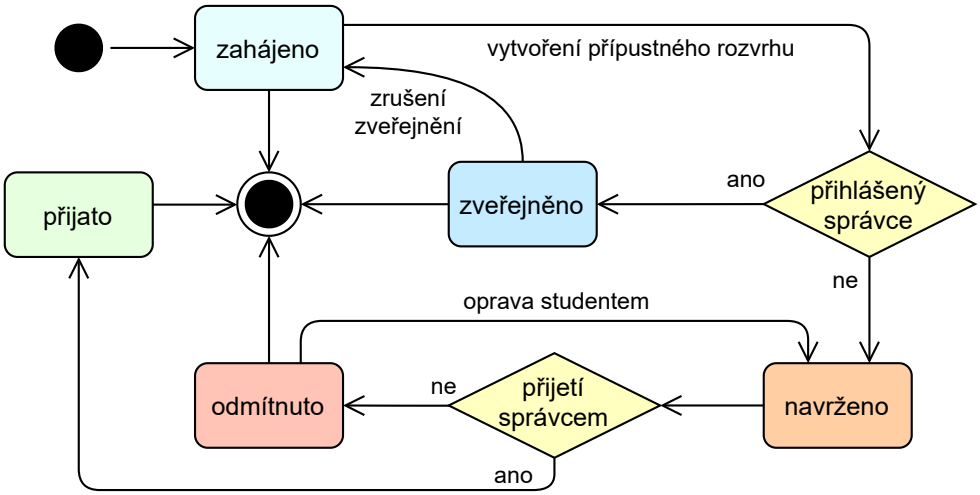


Obrázek 4.3: Návrh na rozšíření databáze – tabulka konfliktů

v případě, že se bude jednat o konflikt při plánování zkoušek. Každý konflikt bude mít název, popis a typ, který bude udávat jeho závažnost.

4.3 Případy užití

Jak již bylo uvedeno v podkapitole 2.5, bude přístup do aplikace umožněn pouze přihlášeným uživatelům. Uživatel se bude moci přihlásit přes svůj účet Google s doménou `vutbr.cz`, čímž se v aplikaci automaticky stává studentem. Uživatel se může také registrovat, ovšem musí jej schválit správce. Správa uživatelů probíhá v aplikaci A. Tesařové, proto zde nebude dále řešena. Studenti mají možnost prohlížet si zveřejněné rozvrhy. Rozvrhy zveřejňují správci (rozvrháři, studijní poradci atd.), kteří mohou zobrazovat a upravovat veškeré rozvrhy. Jednoduchý stavový diagram pro verze rozvrhů lze vidět na obrázku 4.4.

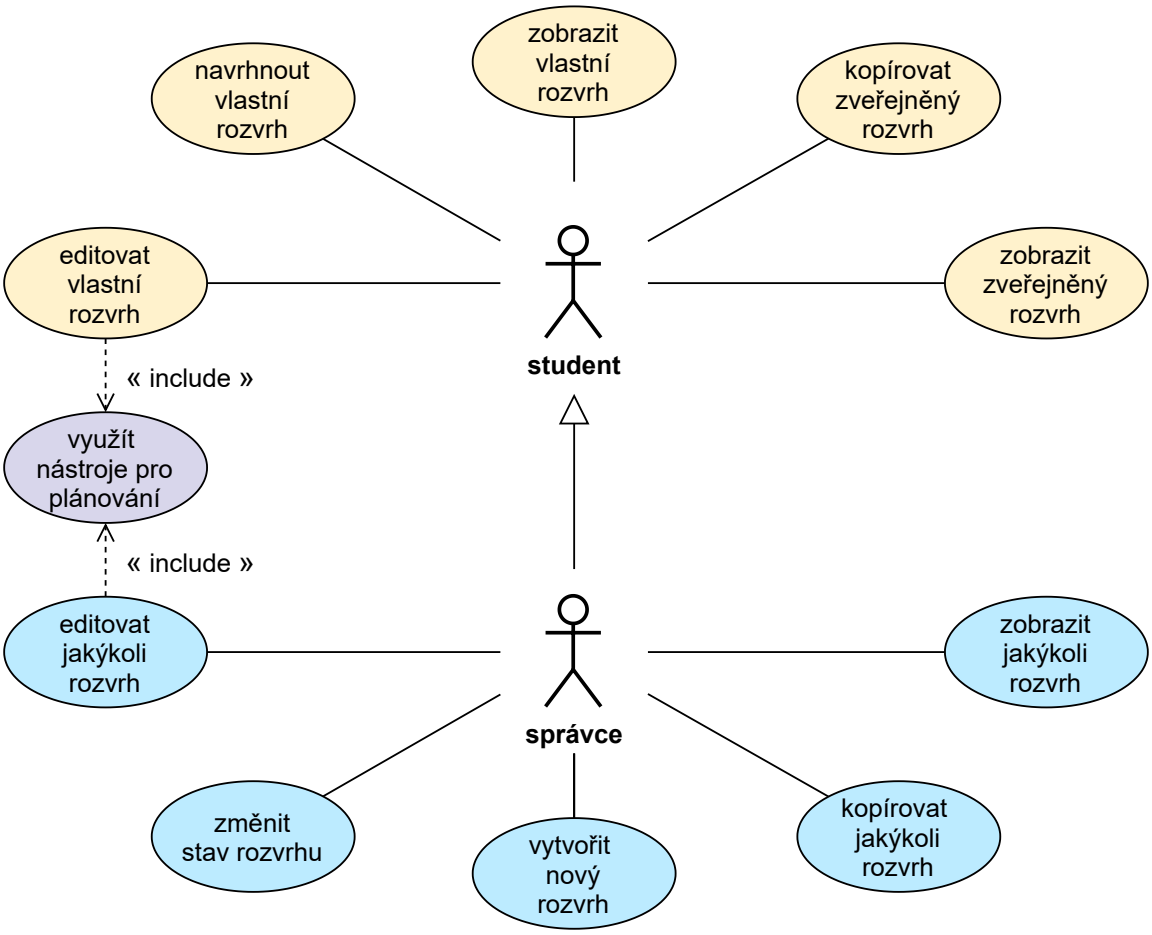


Obrázek 4.4: Stavový diagram verze rozvrhu

Každý nový rozvrh bude nejdříve ve stavu *zahájeno*. V případě, že jej vytváří správce a rozhodne se, že se jedná o finální verzi, může jej zveřejnit. Pokud jej ovšem vytváří student, musí jej pro zveřejnění nejdříve navrhnout správci, který se může rozhodnout pro přijetí či odmítnutí. Pokud je rozvrh odmítnut, student jej může po úpravách opět navrhnout správci ke schválení. V případě přijetí rozvrhu je vytvořena jeho kopie, jejíž vlastníkem je

schvalující správce a tento nový rozvrh je zveřejněn jako oficiální. Zveřejněný rozvrh může správce kdykoli „zneviditelnit“ změnou jeho stavu zpět na *zahájeno*. Zde popsaný způsob práce se stavy rozvrhů byl navrhnut v rámci práce A. Tesařové [17] a je nutné, aby jej i nové aplikace (pracující se stejnou databází) dodržovaly.

Z výše popsaného postupu vytváření rozvrhů a spravování stavů vyplývají také případy užití, jejichž diagram je na obrázku 4.5. Uživatelé aplikace se mohou vyskytovat pouze ve dvou uvedených rolích – student nebo správce. Neregistrovaní uživatelé nemají přímo do aplikace přístup.



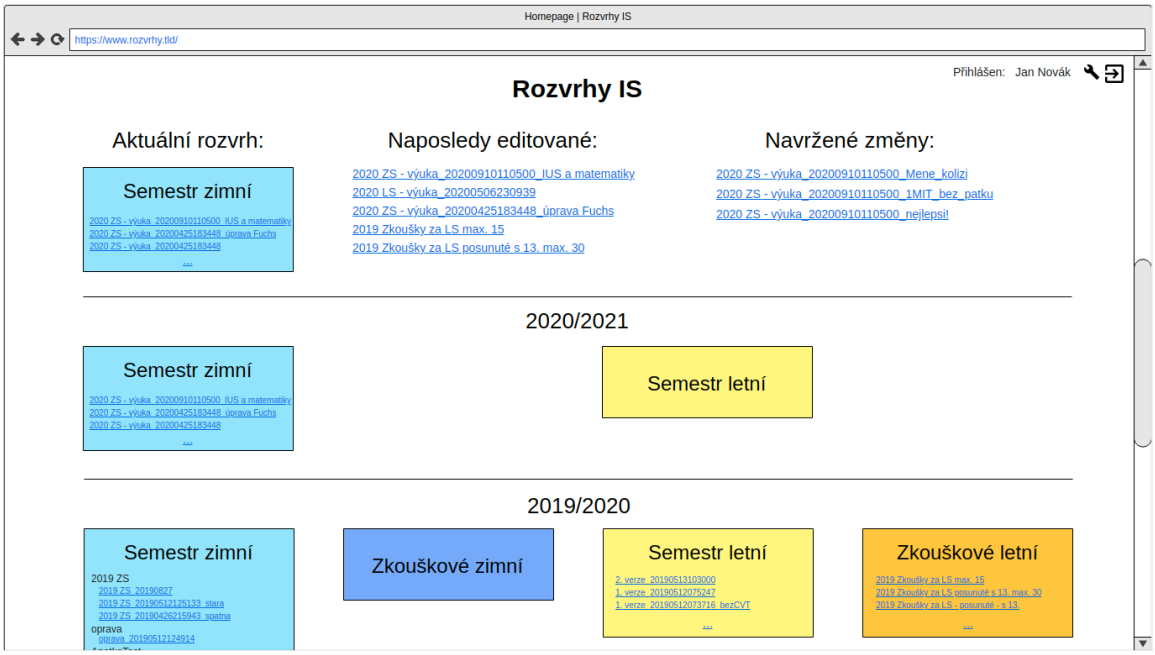
Obrázek 4.5: Diagram případů užití

4.4 Uživatelské rozhraní

Při návrhu uživatelského rozhraní aplikace byl kladen důraz na jednoduchost a přehlednost při zachování již zaběhnutých a osvědčených praktik.

Nejdříve byla navrhnuta **úvodní strana** aplikace, která je dostupná uživateli po přihlášení. Zvoleno bylo jednoduché řádkové rozložení. V prvním rádku se nachází vždy odkaz na aktuální rozvrh, seznam naposledy upravovaných rozvrhů uživatele a poslední navržené rozvrhy od studentů, pokud je přihlášen správce. Každý další řádek představuje jeden školní

rok a řazeny jsou sestupně (od nejaktuálnějšího po nejstarší). Návrh je vizualizován na obrázku 4.6.



Obrázek 4.6: Návrh úvodní strany aplikace

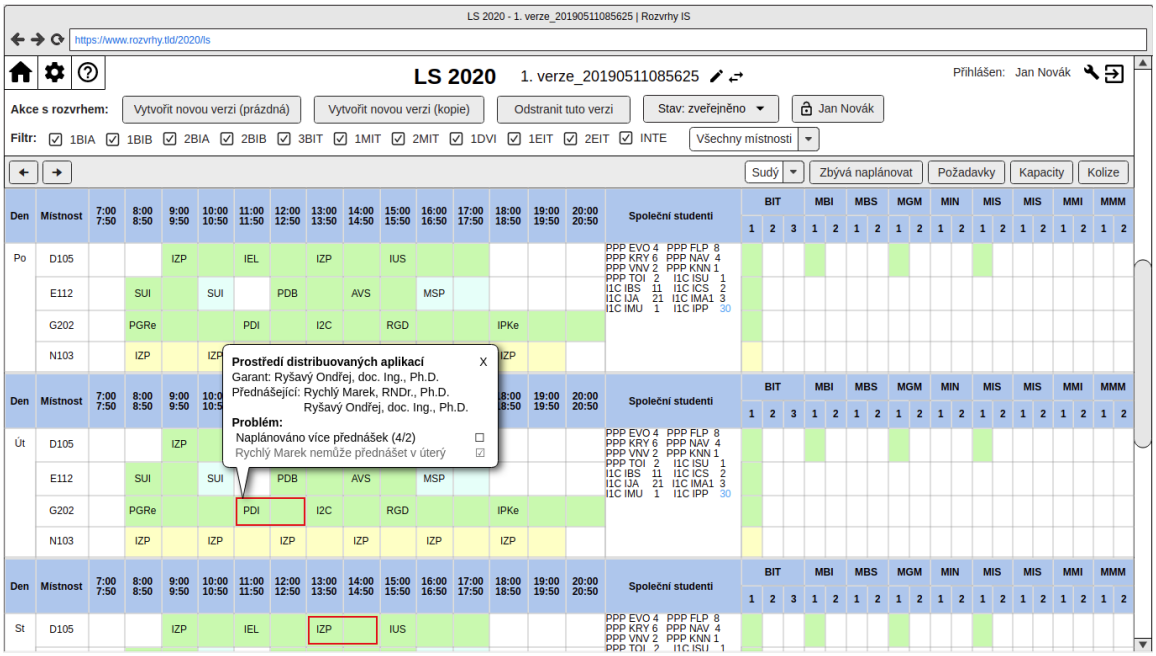
V jednotlivých řádcích (školních rocích) budou až čtyři období – výuka v zimním semestru, zkouškové období zimního semestru, výuka v letním semestru a zkoušky za letní semestr. Každé z těchto období bude obsahovat seznam rozvrhů, které jsou pro ně vytvořené. Pokud přihlášeným uživatelem bude student, uvidí zde pouze zveřejněné rozvrhy. Kliknutím na rozvrh, resp. verzi rozvrhu se uživatel dostane k jeho zobrazení a možnosti plánování. Pro ještě větší zjednodušení ovládání pro uživatele byla navržena optimalizace výpisu verzí – pokud bude zveřejněna/vytvořena pouze jedna verze rozvrhu, výpis zde nebude a uživatel se na rozvrh dostane přímo po kliknutí na období (na obrázku 4.6 je toto zobrazeno v zimním zkouškovém období 2019/2020 nebo ve výuce letního semestru 2020/2021).

Druhý návrh se zabývá plánovacím oknem, resp. oknem pro zobrazení rozvrhu. Navrhovanou podobu uživatelského rozhraní lze vidět na obrázku 4.7

V horní části obrazovky je umístěn název rozvrhu a upravitelné jméno verze. V levém rohu je tlačítko pro návrat na domovskou obrazovku, tlačítko *nápověda*, které by mělo hlavně novým uživatelům pomoci s ovládáním aplikace a vysvětlením principů plánování rozvrhů a tlačítko *nastavení*, které slouží pro úpravu některých kritérií jako například které sloupce (specializace) zobrazovat v grafu povinností.

Pod názvem rozvrhu se nacházejí nástroje pro správu rozvrhu. Uživatel zde může zvolit vytvoření úplně nové prázdné verze, vytvořit kopii této verze, případně verzi odstranit nebo změnit její stav. Poslední tlačítko v této řadě slouží k získání zámku na zobrazenou verzi rozvrhu pro přihlášenou osobu. Pouze vlastník zámku může verzi editovat. Vlastnictví zámku je zapsáno v databázi (sdílené napříč aplikacemi) a je vyžadováno z důvodu, aby dva správci neupravovali rozvrh zároveň a nedocházelo tím k nekonzistencím v databázi.

V další řadě se nacházejí filtry pro jednotlivé ročníky a programy, kterými si může uživatel rozvrh zobrazit například z pohledu studenta daného ročníku. Navíc se zde také



Obrázek 4.7: Návrh plánovacího okna aplikace

nachází filtr místností (všechny, pouze přednáškové nebo pouze CVT), případně v rozvrhu zkoušek přepínání mezi pohledy s místnostmi a bez místností.

Šedá lišta je plovoucí a obsahuje základní nástroje pro plánování a ovládání zobrazení rozvrhu. V levé části jsou umístěny (podobně jako např. v prohlížeči) šipky *vpřed* a *zpět* pro přecházení mezi podverzemi rozvrhu. Zobrazené nástroje v pravé části závisí na typu rozvrhu (plánování výuky nebo zkoušek). Dostupné nástroje budou podrobně popsány v kapitole 4.6. V návrhu je zvolen rozvrh výuky, proto se v liště kromě nástrojů nachází také přepínání mezi pohledem na sudý a lichý týden.

Samotná plánovací tabulka vychází z původní aplikace J. Dytrycha. Konflikty a porušené požadavky, které při plánování vzniknou, budou uživateli vyznačeny barevným rámečkem nebo jednoduchým symbolem či proužkem, aby nebyly příliš rušivé. Kontrolovaná kritéria budou popsána v kapitole 4.5. Po kliknutí myší na konkrétní okno rozvrhu se zobrazí kromě informací o předmětu (počtu hodin pro jednotlivé typy výuky, počty studentů na termíny zkoušek) a jeho vyučujících také seznam problémů, které nastaly. Zde je může uživatel jednoduše ignorovat pomocí zašrtávacího pole.

4.5 Hlídání kritérií

Po umístění rozvrhového okna by měla být zkontrolována kritéria a omezení, která byla popsána v kapitolách 2.2.1 a 2.2.2. Porušení těchto kritérií bude hlášeno barevným upozorněním (na základě priority – navrženy 3 stupně) u konkrétního rozvrhového okna a v informační bublině zobrazitelné pro každé okno bude zobrazen popis problému a možnost jej ignorovat, viz návrh na obrázku 4.7.

Precizní kontroly jsou někdy obtížné, jelikož výuka nemusí být úplně pravidelná a tak nelze vše stoprocentně algoritmizovat. Příkladem může být kolize cvičícího dvou předmětů, kde probíhají cvičení nepravidelně a nikdy se nepotkají (z prvního předmětu ve 2., 5. a 8.

týdnu, z druhého předmětu ve 3., 6. a 9. týdnu). Obecně tedy pro hlídání všech kritérií platí, že je lepší přísnější hlídání s možností nahlášené problémy označit jako ignorované, než nějakou kolizi opomenout a zjistit ji až „za provozu“, tedy při probíhající výuce nebo uprostřed zkouškového období.

Požadavky vyučujících

Před plánováním rozvrhů se od vyučujících zjistí preference a požadavky na výuku či zkoušky pro každý předmět a následně jsou uloženy do databáze pro plánování. Je tedy možné kontrolovat jejich splnění či porušení.

K jednomu předmětu může zadat požadavky více vyučujících a každý požadavek se může vázat na typ výuky a konkrétní den (v případě plánování výuky) nebo termín zkoušky (v případě plánování zkoušek). Vazba na den a termín není povinná – takový požadavek pak platí pro každý den a každý termín.

Požadavky vyučujících se navíc dělí na typy „chtěl bych“ a „nemohu učit“. V případě typu **chtěl bych** vyjadřuje vyučující, že by mu vyhovoval konkrétní termín (čas, datum nebo den v týdnu) nebo místnost. Tento typ se dělí se na 3 priority, přičemž platí:

- **nízká** – preferuji tento den a/nebo čas, ale může být i jiný den a/nebo čas (umožní zadat preference bez obavy z komplikování plánování);
- **střední** – takto bych to potřeboval, ale pokud to nepůjde, požadavek nemusí být splněn (nesplnění se konzultuje nejpozději po zveřejnění 1. předběžného návrhu);
- **vysoká** – tento požadavek musí být splněn a nebude-li to možné, je potřeba se s vyučujícím domluvit ještě před zveřejněním 1. návrhu rozvrhu (konzultuje se individuálně při plánování).

Pokud zadá vyučující požadavek typu **nemohu učit**, oznamuje tím, že není k dispozici v konkrétním čase a tedy mu nelze v tomto termínu výuku nebo zkoušky naplánovat.

Požadavky se váží vždy k jednomu předmětu a tedy je možné je po umístění okna zkontrolovat bez jiných souvislostí a oken. Požadavek typu „chtěl bych“ se považuje za nesplněný a bude hlášen (s nízkou prioritou) pokud okno *neleží* v časovém intervalu tohoto požadavku. Naopak požadavek typu „nemohu učit“ se považuje za nesplněný a bude hlášen (s vysokou prioritou) pokud okno *leží* v časovém intervalu požadavku.

U plánování výuky vznikla navíc potřeba zjistit, zda je splněný alespoň jeden z požadavků typu „chtěl bych“. Pokud totiž učitel zadá více termínů, které by mu vyhovovaly a rozvrhář jeden z nich splní, bylo by vhodné označit jako splněné i všechny ostatní. Pro kontrolu tohoto splnění je již nutné brát v úvahu všechna naplánovaná rozvrhová okna předmětu.

Kolize vyučujících

Ve výsledném rozvrhu musí být zajištěno, aby vyučující neměl naplánována dvě různá výuková okna na jeden čas, přičemž vyučující může být k předmětu přiřazen třemi různými rolemi. U **garanta předmětu** (případně jeho zástupce) není třeba kolize řešit, jelikož je dle typu výuky uveden také jako přednášející nebo cvičící. **Přednášející** jsou hlavními vyučujícími v předmětu a je třeba se u nich vyhnout kolizím hlavně u přednášek a zkoušek. **Cvičící** obstarávají výuku praktických cvičení, tedy typicky výuku menších skupin studentů v laboratořích nebo demonstrační cvičení. Obvykle bývá těchto cvičících více stejně jako

rozvrhových oken pro cvičení, tudíž nemusí být v praxi problém se i případné kolizi vyhnout (dané okno bude vyučovat jiný z cvičících).

Při kontrole těchto kolizí je nutné nejdříve identifikovat možná kolizní okna v rozvrhu (překrývající se časy a jiný předmět nebo typ okna) a poté k nim přiřadit vyučující dle typu výuky. Když se mezi vyučujícími obou kolizních oken vyskytne průnik, je nutné tuto kolizi hlásit pouze v případě, že nastane jedna z následujících variant:

- obě okna jsou přednášky a v kolizi je přednášející,
- obě okna jsou cvičení a v kolizi je vyučující, který je alespoň v jednom z předmětů vedený jako jediný cvičící,
- okno A je přednáška, okno B je cvičení a přednášející předmětu okna A je zároveň jediným cvičícím v předmětu okna B.

U plánování rozvrhu zkoušek se typ výuky nerozlišuje a do kolizí jsou zařazeni automaticky všichni vyučující z předmětu (včetně garanta). Tato vlastnost je vyžadována kvůli organizaci písemných zkoušek, kdy jsou garantem většinou osloveni pro zajištění dozorů přednášející a cvičící z daného předmětu.

Jednoduché kolize studentů

Kolize studentů jsou velmi podobné kolizím vyučujících. V případě, že dvě časově se překrývající okna mají nenulový počet společných studentů a zároveň nemá okno jinou variantu (přednášková skupina, skupiny na cvičení), je třeba tuto kolizi řešit. V detailu daného upozornění je vhodné také uvést, kolika studentů se tato kolize týká.

U rozvrhu výuky se kolize studentů hlídají pouze mezi dvěma přednáškami a rozdělují se do několika skupin podle povinností předmětů pro studenta u této dvojice. Rozlišuje se tedy:

- nejvíce závažná kolize mezi dvěma povinnými předměty, povinně volitelnými předměty nebo jejich kombinací,
- středně závažná kolize povinného nebo povinně volitelného předmětu s volitelným,
- nejméně závažná kolize dvou volitelných předmětů.

Oproti tomu u rozvrhu zkoušek se povinnost kolizních předmětů nerozlišuje, jelikož student musí mít možnost zúčastnit se všech termínů zkoušek u svých zapsaných předmětů.

Kapacity

Rozvrhové akce se plánují do konkrétních místností, a proto je potřeba hlídat, že má místnost (popř. místnosti) pro tuto akci dostatečnou kapacitu. Maximální kapacita je uložena v databázi u každé místnosti, a proto se zdá, že ji lze zkontrolovat velmi jednoduše. Ovšem je nutné si uvědomit, že rozvrhové okno se může nacházet ve více místnostech (případně v žádné) a může být rozděleno na více skupin (dvě přednáškové skupiny, cvičení nebo více běhů u zkoušek).

U rozvrhů výuky je třeba vždy sečíst kapacitu veškerých místností, ve kterých se nachází daný typ výuky, a to pro všechny skupiny. Zvýšenou pozornost je nutné věnovat rozděleným přednáškám, kdy první část probíhá například dopoledne a druhá odpoledne – v tomto případě se jedná o jednu skupinu. Výslednou kapacitu stačí porovnat s počtem zapsaných

studentů v předmětu a pokud bude nedostatečná, je třeba zobrazit upozornění o vysoké prioritě. Pokud se ovšem bude jednat o nedostatečnou kapacitu přednášek v intervalu 70 - 100 % kapacity, je možné zobrazit pouze upozornění se střední prioritou.

Při plánování rozvrhu zkoušek je situace komplikovanější, a to nejen z důvodu možného rozesazení „ob1 a ob2“. Nejdříve se totiž plánují zkoušky bez přiřazení jednotlivých oken do místností. U těchto oken se bude kontrolovat kapacita vůči areálu, tedy že součet studentů vynásobený modifikátorem rozesazení u překrývajících se oken nepřesáhne 1052. Po umístění zkoušek do místností je kontrola podobná jako u výuky, tedy stačí sečíst kapacitu místností (s modifikátorem rozesazení) a porovnat s počtem studentů pro daný termín.

Jelikož počet studentů, kteří se zúčastní jednotlivých termínů zkoušky, není samozřejmě dopředu znám, vychází se pro každý termín ze dvou možných variant. Pro pesimistickou variantu se počítá na předtermín s 10 % zapsaných studentů, na 1. termín se všemi zapsanými, na 2. termín s 60 % zapsaných a na 3. a další termíny s 30 % zapsaných studentů. Pokud kapacita pro tuto variantu nebude dostačovat, jedná se pouze o středně závažný problém, jelikož jsou procenta záměrně nadsazena a počítá se tedy na zkouškách s menší účastí. Druhou variantou je odhad počtu studentů, který se vypočítá poměrově ze skutečných loňských počtů studentů, kteří se zúčastnili jednotlivých termínů. Pokud by kapacita nedostačovala pro tuto variantu, bude se jednat o problém velmi závažný, jelikož lze očekávat, že tyto odhady se velmi blíží realitě.

Zkoušky z povinných předmětů v jednom dni

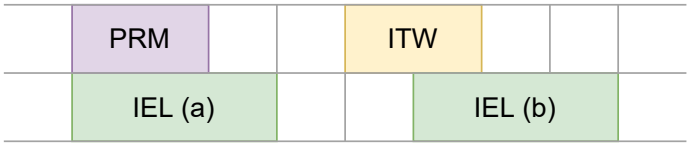
Při plánování zkoušek je třeba zajistit, aby neměl jeden student více zkoušek z povinných předmětů v jednom dni (minimálně u prvních termínů). Stejně jako ve výuce u běžných studentských kolizí, i zde je nutné zjistit povinnost předmětu pro každého studenta zvlášť (stejný předmět mohou mít studenti povinný, povinně volitelný nebo volitelný).

Pokud bude existovat alespoň jeden student s více než jednou povinnou zkouškou v jednom dni, bude třeba zobrazit problém u těchto kolizních oken s vysokou prioritou. Rovněž bude nutné zobrazit v detailu i počet těchto studentů, aby se rozvrhář mohl rozhodnout pro ignorování konfliktu nebo pro jeho řešení.

Možnost navštívit výuku (tranzitivní kolize)

Posledním kritériem pro hlídání je zajištění možnosti pro každého studenta navštívit veškerou výuku v předmětech, které má zapsané. Jedná se o hlídání složitějších, tzv. tranzitivních kolizí.

Nejjednodušším příkladem tranzitivní kolize je situace se třemi předměty znázorněná na obrázku 4.8. I když obě kolizní dvojice (*PRM* x *IEL* a *ITW* x *IEL*) mají nějaký počet společných studentů, ani jeden konflikt není vyvolán, protože *IEL* má i jinou skupinu, kterou si studenti mohou k navštívení přednášky zvolit. Pokud však bude mít student zapsány všechny tři vyobrazené předměty, je jisté, že se mu je nepodaří navštívit. Jednoduché kolize v tomto případě nepomohou, a proto je nutné kontrolovat také kolize tranzitivní.



Obrázek 4.8: Příklad jednoduché tranzitivní kolize

V tomto jednoduchém příkladu je samozřejmě možné kolizi identifikovat na první pohled, ovšem není těžké odhadnout, že kolize budou s přibývajícími okny v rozvrhu stále komplikovanější a náročnější na kontrolu. Vstupují do nich také skupiny pro cvičení a nezjednodušuje je ani fakt, že studenti mohou při studiu volit z široké škály volitelných předmětů, čímž se možnosti pro rozvrh bez kolizí poměrně redukují.

Pro zjištění tranzitivních kolizí je nutné přesně vědět, který student má zapsané jaké předměty a řešit tyto kolize v podstatě individuálně pro každého studenta. Pokud se pro všechny podaří sestavit bezkolizní rozvrh (postupným výběrem přednáškových skupin a cvičení), tranzitivní kolize se v rozvrhu nenachází. V opačném případě je nutné na kolize upozornit u každého kolizního okna a zjistit počet studentů, kterých se kolize (nemožnost navštívit výuku) týká.

4.6 Nástroje

Pod pojmem nástroje jsou chápány pomůcky, které rozvrháři nepřímo pomáhají s plánováním rozvrhů a jejich obsah není vždy ovlivňován vytvářeným rozvrhem. Jak již bylo zmíněno v předchozí kapitole, jejich nabídka v plovoucím menu závisí na typu rozvrhu, viz tabulka 4.1.

Nástroj	**Typ rozvrhu**	
	výuka	**zkoušky**
kapacity	ano	ano
kolize	ano	ano
požadavky	ano	ano
zbývá naplánovat	ano	ano
rozestupy	ne	ano
zkoušky v týdnu	ne	ano

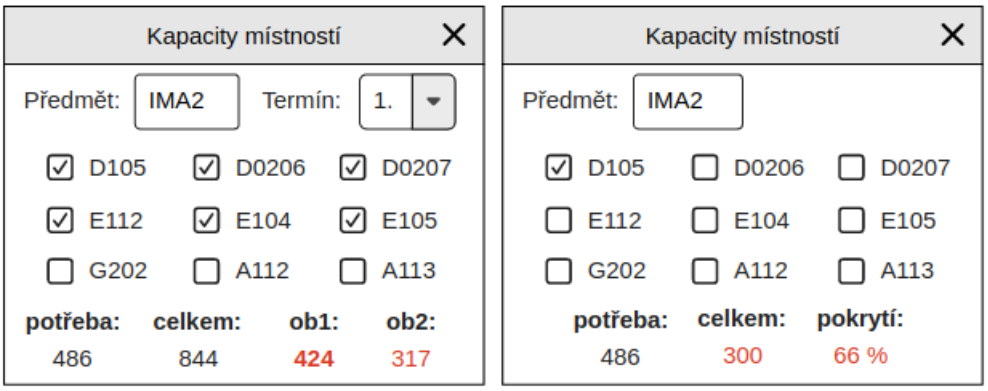
Tabulka 4.1: Nabídka nástrojů v závislosti na typu plánovaného rozvrhu

Jednotlivé nástroje jsou navrženy jako samostatná okna, která uživateli zůstanou zobrazená na stránce přes plánovací tabulku, kde s nimi může pohybovat, pokud bude potřebovat. Této vlastnosti lze s výhodou použít například u nástroje *požadavky*, kdy si rozvrhář v pravé části obrazovky zobrazí v okně tohoto nástroje požadavky ke konkrétnímu předmětu a v levé části okna předmětu umisťuje do rozvrhu.

Kapacity

Nástroj *kapacity* slouží pro naplánování rozvrhové akce do místností a zjištění, zda je pro ni kapacita zvolených místností dostatečná. Návrh nástroje pro oba typy rozvrhů lze vidět na obrázku 4.9.

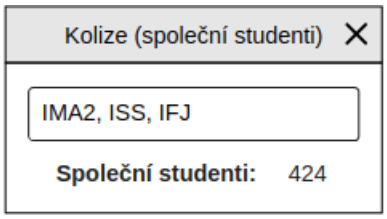
Do vstupního pole stačí zadat zkratku předmětu a pokud je správná, zobrazí se ve spodní části potřebný počet míst. Po zvolení místností nástroj zobrazí jejich celkovou kapacitu a informaci o tom, zda je kapacitně možné do nich předmět naplánovat. U plánování zkoušek lze zvolit konkrétní termín (což ovlivní počet studentů a jejich cílové rozložení v místnosti), při plánování výuky je zobrazené procento dostatečné kapacity (u přednášek postačuje pokrýt 70 %). Nástroj má pouze informativní charakter, tedy neumisťuje okna do rozvrhu.



Obrázek 4.9: Návrh nástroje *kapacity* (vlevo pro zkoušky, vpravo pro výuku)

Kolize

V dosavadních aplikacích sloužil nástroj *kolize* pouze pro zjištění počtu společných studentů ve dvojici předmětů. Po konzultaci s rozvrháři bylo dohodnuto, že se nástroj rozšíří tak, aby bylo možné zjistit počet společných studentů ve dvou a více předmětech. Jeho návrh je na obrázku 4.10.



Obrázek 4.10: Návrh nástroje *kolize*

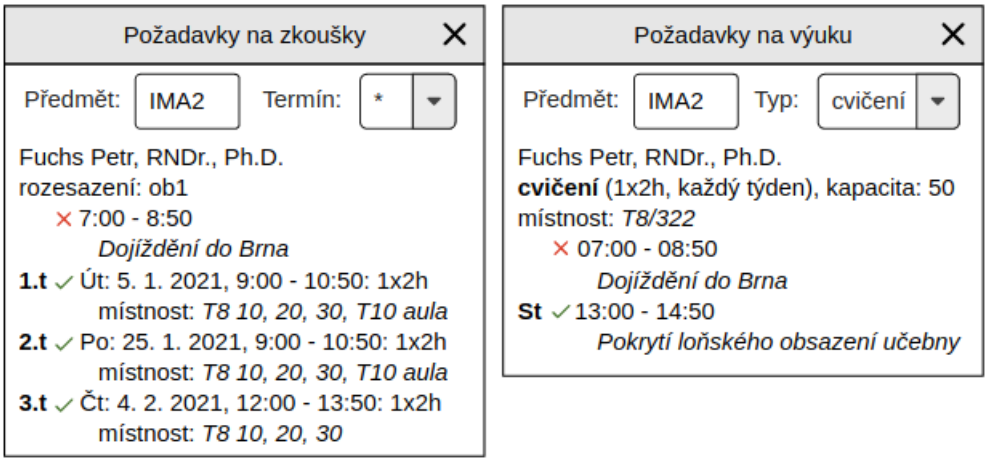
Pro vstup bude použito jedno textové pole do kterého budou vepsány zkratky předmětů oddělené čárkami. Nástroj následně vyhledá a vypíše počet společných studentů zadaných předmětů.

Požadavky

Pro rychlé zobrazení požadavků vyučujících na výuku poslouží rozvrháři nástroj *požadavky*. Ty budou vizuálně dělené na preference (vyučující zadal požadavek „chtěl bych/vyhovoval by mi termín“) a omezení typu „nemohu učit v termínu“. Pokud jsou vázány na termín zkoušky či na konkrétní den týdnu, budou sdruženy i podle tohoto kritéria a bude možné je podle něj i filtrovat, viz návrhy nástroje na obrázku 4.11.

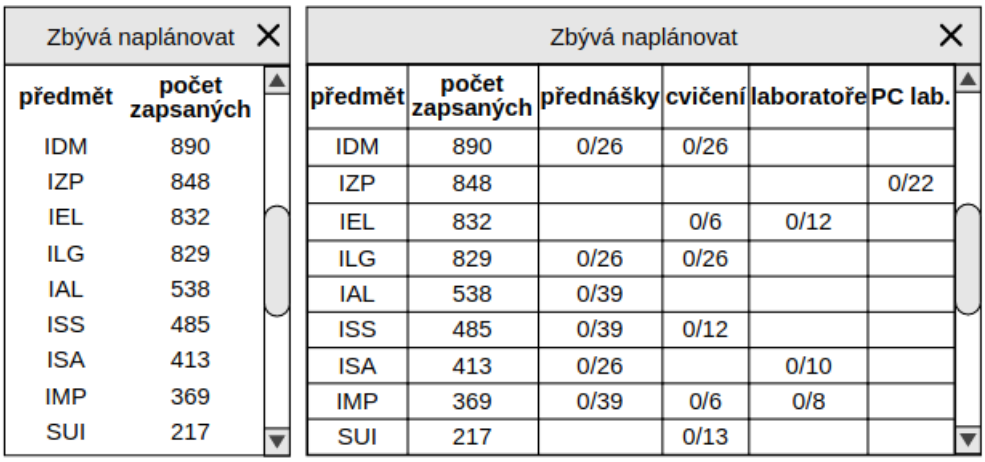
Zbývá naplánovat

Aby měl rozvrhář stále přehled o tom, které předměty a k nim konkrétní typy výuky ještě nemá v rozvrhu naplánované, může využít nástroj *zbývá naplánovat*. V něm se zobrazí seznam předmětů seřazených sestupně dle počtu studentů (předměty se plánují od „největších“), které ještě nemají v rozvrhu naplánované všechny požadované typy výuky v daném rozsahu. Jedná se tedy o nástroj, který bude spolupracovat s vytvářeným rozvrhem a svůj obsah v závislosti na něm dynamicky měnit.



Obrázek 4.11: Návrh nástroje *požadavky* (vlevo pro zkoušky, vpravo pro výuku)

Později při plánování zkoušek bylo rozvrháři zjištěno, že by se stejný nástroj využil i při plánování zkoušek. Zde ovšem stačí vypsat pouze předměty seřazené podle počtu studentů bez nutnosti kontrolovat existenci všech termínů (ty lze zkontrolovat v nástroji *rozestupy*, který bude popsán dále v textu). Návrh nástroje pro oba typy rozvrhů lze vidět na obrázku 4.12.

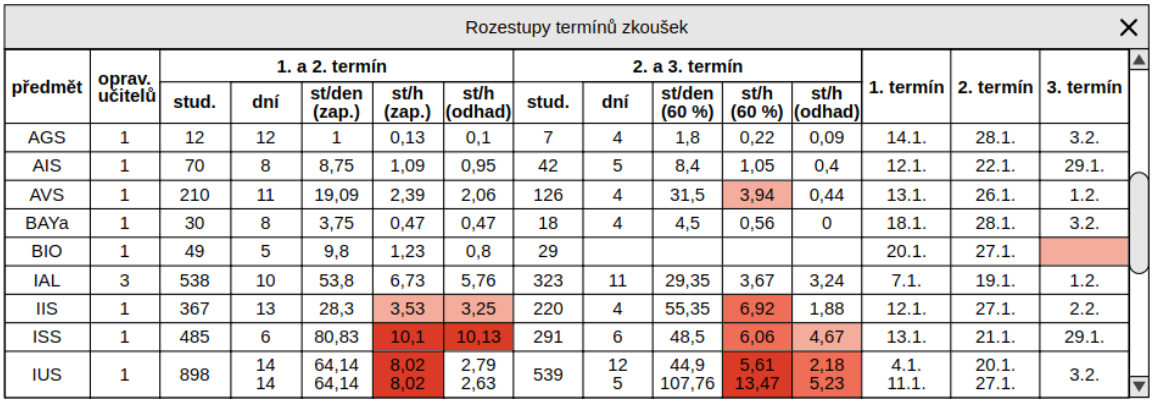


Obrázek 4.12: Návrh nástroje *zbývá naplánovat* (vlevo pro zkoušky, vpravo pro výuku)

Rozestupy

Při plánování zkoušek je nutné dodržet mezi jednotlivými termíny u konkrétního předmětu požadované rozestupy. Ty mohou být definovány požadavky učitelů, ale častěji je nutné, aby čas mezi termíny hlídal rozvrhář. Čas potřebný na opravu jednoho termínu zkoušky je přímo závislý na počtu studentů na termínu zkoušky a nepřímo závislý na počtu opravujících.

Pro jednodušší kontrolu těchto rozestupů byl navržen nástroj *rozestupy*, viz obrázek 4.13. Samozřejmě se jedná o nástroj, který bude dynamicky měnit svůj obsah v závislosti na upravovaném rozvrhu.



Rozestupy termínů zkoušek

předmět	oprav. učitelů	1. a 2. termín					2. a 3. termín					1. termín	2. termín	3. termín
		stud.	dní	st/den (zap.)	st/h (zap.)	st/h (odhad)	stud.	dní	st/den (60 %)	st/h (60 %)	st/h (odhad)			
AGS	1	12	12	1	0,13	0,1	7	4	1,8	0,22	0,09	14.1.	28.1.	3.2.
AIS	1	70	8	8,75	1,09	0,95	42	5	8,4	1,05	0,4	12.1.	22.1.	29.1.
AVS	1	210	11	19,09	2,39	2,06	126	4	31,5	3,94	0,44	13.1.	26.1.	1.2.
BAYa	1	30	8	3,75	0,47	0,47	18	4	4,5	0,56	0	18.1.	28.1.	3.2.
BIO	1	49	5	9,8	1,23	0,8	29					20.1.	27.1.	
IAL	3	538	10	53,8	6,73	5,76	323	11	29,35	3,67	3,24	7.1.	19.1.	1.2.
IIS	1	367	13	28,3	3,53	3,25	220	4	55,35	6,92	1,88	12.1.	27.1.	2.2.
ISS	1	485	6	80,83	10,1	10,13	291	6	48,5	6,06	4,67	13.1.	21.1.	29.1.
IUS	1	898	14 14	64,14 64,14	8,02 8,02	2,79 2,63	539	12 5	44,9 107,76	5,61 13,47	2,18 5,23	4.1. 11.1.	20.1. 27.1.	3.2.

Obrázek 4.13: Návrh nástroje *rozestupy*

Pro výpočet počtu opravených studentů na hodinu se mezi 1. a 2. termínem využije skutečný počet zapsaných studentů i odhad pro 1. termín z minulého roku. Stejně tak se pro výpočet mezi 2. a 3. termínem využije 60 % počtu zapsaných i odhad pro 2. termín z minulého roku. Pro každý předmět tedy vzniknou 4 čísla, která jsou případně zvýrazněna podle jejich vážnosti (čím vyšší, tím závažnější). V tabulce jsou také vypsány jednotlivé termíny pro kontrolu, že jsou všechny 3 termíny zkoušky naplánovány.

Pokud se jedná o předmět s pěti variantními termíny (na obrázku 4.13 je jako příklad uveden IUS), jsou výpočty zdvojené (1. a 2. variantní termíny jsou ve skutečnosti 2 varianty 1. termínu zkoušky, 3. a 4. variantní termíny jsou varianty 2. termínu). Výpočet počtu studentů na opravu probíhá mezi dvěma navazujícími, tedy mezi 1. a 3. termínem a 2. a 4. termínem.

Zkoušky v týdnu

Posledním navrženým nástrojem jsou *zkoušky v týdnu*. Jedná se o výpis počtu studentů, kteří mají v daném týdnu více než dvě zkoušky. Nástroj bude využívat stejná data jako nástroj *kolize*, ovšem místo uživatelského vstupu bude n-tice předmětů generovat na základě upravovaného rozvrhu.

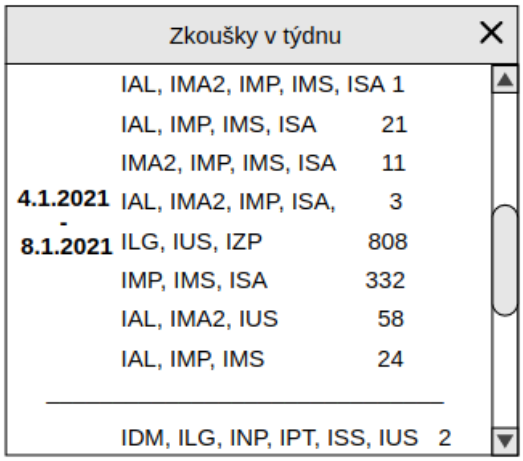
Na návrhu zobrazení nástroje na obrázku 4.14 je vidět, že budou n-tice kolizních předmětů seřazeny od nejdelší k nejkratší, tedy přednostně budou řešeni studenti, kteří mají mnoho zkoušek v jednom týdnu.

4.7 Zobrazení zveřejněných rozvrhů

Zveřejňování rozvrhů probíhá aktuálně tak, že po vytvoření výsledného rozvrhu je proveden jeho export v aplikaci A. Tesařové. Výsledkem exportu je několik souborů ve formátu HTML, které se umístí na webové stránky studijního poradenství[1].

Při tomto postupu je nutné přepínat z plánovací aplikace na aplikaci A. Tesařové, která je většinu času otevřena kvůli obarvování požadavků vyučujících. Větším problémem jsou pozdější změny rozvrhu, např. na základě studentských podnětů nebo připomínek vyučujících. V tomto případě je nutné proces exportu celý zopakovat.

[1] https://www.fit.vut.cz/study/advisor/#rozvrhy



Obrázek 4.14: Návrh nástroje *zkoušky v týdnu*

Po konzultaci s rozvrháři se zjistilo, že by bylo ideální zobrazovat rozvrhy ve stavu *zveřejněno* přímo v plánovací aplikaci, a to nejen přihlášeným uživatelům (studentům, správcům), ale veřejně každému, kdo dostane na rozvrh odkaz. I v současné době se odkazy na návrhy rozvrhů šíří pomocí sociálních sítí (nejvíce se využívají studentské skupiny na síti Facebook a fakultní server platformy Discord) a tedy není problém odkaz do plánovací aplikace přidat nebo nahradit. Případně lze odkaz na veřejné rozvrhy umístit i na web studijního poradenství.

Výhodou přímého zobrazení rozvrhů v aplikaci bude především jejich aktuálnost, jelikož budou vykreslovány přímo z plánovací databáze a každá změna ze strany rozvrháře se v nich okamžitě projeví.

Další nespornou výhodou je ulehčení práce rozvrháře ohledně generování a vkládání souborů ve formátu HTML s výslednými rozvrhy na web poradenství. V aplikaci by mu nyní stačilo pouze změnit stav rozvrhu na *zveřejněno* a studentům sdělit vygenerovaný veřejný odkaz.

Třetí výhodou je intuitivnější přístup pro studenty, kteří se chtějí zapojit do plánování a vyzkoušet si upravit rozvrh ve vytvářené plánovací aplikaci. Z každého zveřejněného rozvrhu se totiž student po přihlášení dostane do pohledu pro editaci rozvrhu, odkud si může vytvořit vlastní kopii nebo úplně prázdnou verzi.

Při vytváření zobrazování veřejných rozvrhů se bude přihlížet ke stávající podobě, která je již několik let zavedená a studentům i vyučujícím vyhovuje.

Rozvrh výuky bude rozdělen dle jednotlivých přednáškových skupin. Každá tato skupina bude mít svoji vlastní webovou stránku, na které bude zobrazen rozvrh, tabulky společných studentů, počty studentů v jednotlivých oborech (specializacích), kapacity místností a seznam povinných předmětů v jednotlivých ročnících studia. Seznam přednáškových skupin pro rozvrh výuky v zimním semestru 2020/2021 lze nalézt na `https://www.fit.vut.cz/study/advisor/20202021/priprava_ZS/index.html` a konkrétní zobrazení pro skupinu *1BIA* na `https://www.fit.vut.cz/study/advisor/20202021/priprava_ZS/1bia.html`.

Rozvrh zkoušek bude mít dvě verze – s místnostmi a bez místností. Obě verze budou na stránce zobrazovat kromě rozvrhu také požadavky a poznámky k rozvrhu, tabulky společných studentů, počty studentů v jednotlivých oborech (specializacích), kapacity místností, požadavky vyučujících (s obarvením na základě jejich splnění) a seznam povinných předmětů v jednotlivých ročnících studia. Kromě zobrazení rozvrhů by měly být studentům

k dispozici tabulky pro jednotlivé ročníky dle vzoru Ing. Eysselta, které obsahují seznam předmětů pro daný ročník a časy jejich zkoušek. Rozvrh bez místností pro zkoušky v zimním semestru 2020/2021 lze vidět na https://www.fit.vut.cz/study/advisor/20202021/zkouskyZS/index.html, verzi s místnostmi na https://www.fit.vut.cz/study/advisor/20202021/zkouskyZS/mistnosti.html a tabulku zkoušek pro přednáškovou skupinu *1BIT* na https://www.fit.vut.cz/study/advisor/20202021/zkouskyZS/1bitzs.html.

Jelikož není tato funkcionalita předmětem zadání diplomové práce a je možné rozvrhy zveřejňovat původním postupem, bude na ni pohlíženo jako na rozšíření a bude implementována jako nejméně prioritní.

Kapitola 5

Implementace

Kapitola pojednává o různých pasážích z implementace aplikace pro plánování rozvrhů, které by mohly být pro čtenáře zajímavé. Mimo jiné je zmíněna také architektura klientské aplikace. Aby bylo pro budoucího řešitele snadnější pochopit koncept a dopsat modul do programu, jsou vysvětleny principy kontrol a nástrojů a na závěr je opět zmíněna problematika tranzitivních kolizí.

Důraz bude kladen také na témata, která nebyla možná přesně, správně nebo vůbec identifikovat při návrhu. Příkladem mohou být využité knihovny nebo změny v zobrazení tranzitivních kolizí.

5.1 Využité knihovny

V kapitole 3 jsou popsány programovací jazyky a technologie, které byly použity pro implementaci aplikace. Ovšem v jejím průběhu vznikla potřeba využít pro některé problémy i knihovny, převážně pro jazyk JavaScript. Ty řeší funkcionalitu, která nebyla identifikována ve fázi návrhu a její implementace by byla zbytečně náročná. V tabulce 5.1 je uveden jejich kompletní výčet včetně použitých verzí, licencí a způsobu jejich využití v aplikaci.

Název	Verze	Licence	Popis a webové stránky
jQuery	3.5.1	MIT	manipulace s DOM, AJAX https://jquery.com/
jQuery UI	1.12.1	MIT	dialogová okna, zaškrtávací pole https://jqueryui.com/
jQuery.floatThead	2.2.1	MIT	plovoucí hlavička tabulky https://mkoryak.github.io/floatThead/
Font Awesome	5.15.3	CC BY 4.0	ikony https://fontawesome.com/
iziToast	1.4.0	Apache 2.0	vyskakovací upozornění a informace https://izitoast.marcelodolza.com/

Tabulka 5.1: Knihovny použité při implementaci aplikace

5.2 Architektura aplikace

Většina logiky plánování je kvůli rychlosti aplikace přesunuta na klientskou stranu do JavaScriptu. Veškerá potřebná data jsou předána při načtení stránky s rozvrhem vypsáním do proměnných.

Klientská aplikace je rozdělena do několika tříd, které budou krátce popsány v následujícím textu. Základem celého rozvrhu jsou objekty rozvrhových oken reprezentovaných třídou `Window`. Kromě atributů uchovávajících informace o okně obsahuje metody například pro aktualizaci zobrazení varování a seznamy konfliktů daného okna.

Třída `Timetable` uchovává strukturu rozvrhu (v případě výuky jednu pro sudý a druhou pro lichý týden) a implementuje metody pro vkládání, mazání a úpravu oken a místností v nich.

Pro vizualizaci struktury rozvrhu slouží třída `Renderer`. Pomocí knihovny jQuery upravuje vizuální rozvrhovou tabulku při každé změně. Navíc se stará také o filtrování zobrazených oken.

Pohyb „zpět“ a „vpřed“ (verzování) v rozvrhu zajišťuje třída `Versions`. Tato třída udržuje cache 10 verzí vpřed i zpět pro rychlé přepnutí. Každá verze zde uchovává dvojici přidaného a odebraného okna. Pokud je potřeba, cache se doplní dotazem na server přes API. Při vložení nového okna do rozvrhu se tato třída postará o informování serveru o tomto okně a ten je uloží do databáze (čímž se zároveň vygeneruje nová verze).

Pomocná statická třída `Helper` slouží pro sdružení pomocných funkcí, převážně pro práci s textem okna (vstupem od uživatele).

Prováděné kontroly jsou definovány ve zvláštních statických třídách `Capacities` (hlídání kapacit místností a areálu), `Conflicts` (učitelské a studentské konflikty) a `Requests` (splnění nebo porušení učitelských požadavků). Každá z těchto tříd implementuje v nějaké podobě metodu `check(...)`, která je volána po vložení okna do struktury rozvrhu ze třídy `Timetable`) nebo po uložení okna do databáze ze třídy `Vesrsions` (záleží na typu kontroly). Každá z těchto tříd vkládá nalezené problémy do atributů rozvrhového okna a také zajišťuje ignorování těchto problémů včetně jejich ukládání do databáze přes volání API.

Veškeré nástroje jsou sdruženy do souboru **tools.js**, kde je každý nástroj reprezentován jednou funkcí. Po zavolání této funkce se inicializuje obsah nástrojového okna nebo tuto funkci využívá nástroj samotný (např. zjištění kolizních studentů).

5.3 Přihlášení pomocí účtu Google přes OAuth

Do aplikace je možné přihlášení pomocí uživatelského jména a hesla. Databáze uživatelů je sdílena napříč plánovacími aplikacemi, a proto bylo nutné tento způsob zachovat.

Ovšem jak již bylo v návrhu zmíněno, bylo by nevhodné nutit studenty zakládat si další účet do nové aplikace. Z tohoto důvodu byla implementována možnost přihlásit se pomocí školního účtu Google. Pokud se uživatel prokáže důvěryhodné autoritě přes OAuth (v tomto případě společnosti Google) na základě školního e-mailu, není důvod mu do aplikace neumožnit přístup s rolí studenta.

Aby bylo možné využívat služby Google tímto způsobem, je třeba nejdříve aplikaci (její webovou adresu) zaregistrovat na stránkách *Google Cloud Platform* v záložce *APIs & Services*, čímž získá aplikace jedinečný identifikátor. Na samotné přihlašovací stránce stačí připojit jednu JavaScriptovou knihovnu od Google, nastavit v hlavičce element *meta google-signin-client_id* na vygenerovaný identifikátor a vytvořit element s třídou *g-signin2* (viz obrázek 5.1).

Obrázek 5.1: Přihlašovací obrazovka s možností přihlášení přes Google

Po kliknutí na tento element se uživateli zobrazí možnost přihlášení do účtu Google, a pokud se přihlášení podaří, údaje o uživateli (e-mail, jméno a příjmení) jsou asynchronně vráceny zpět na přihlašovací stránku. Po obdržení těchto údajů je zkontrolováno, zda se uživatel opravdu přihlásil pomocí školního e-mailu (s doménou *vutbr.cz*).

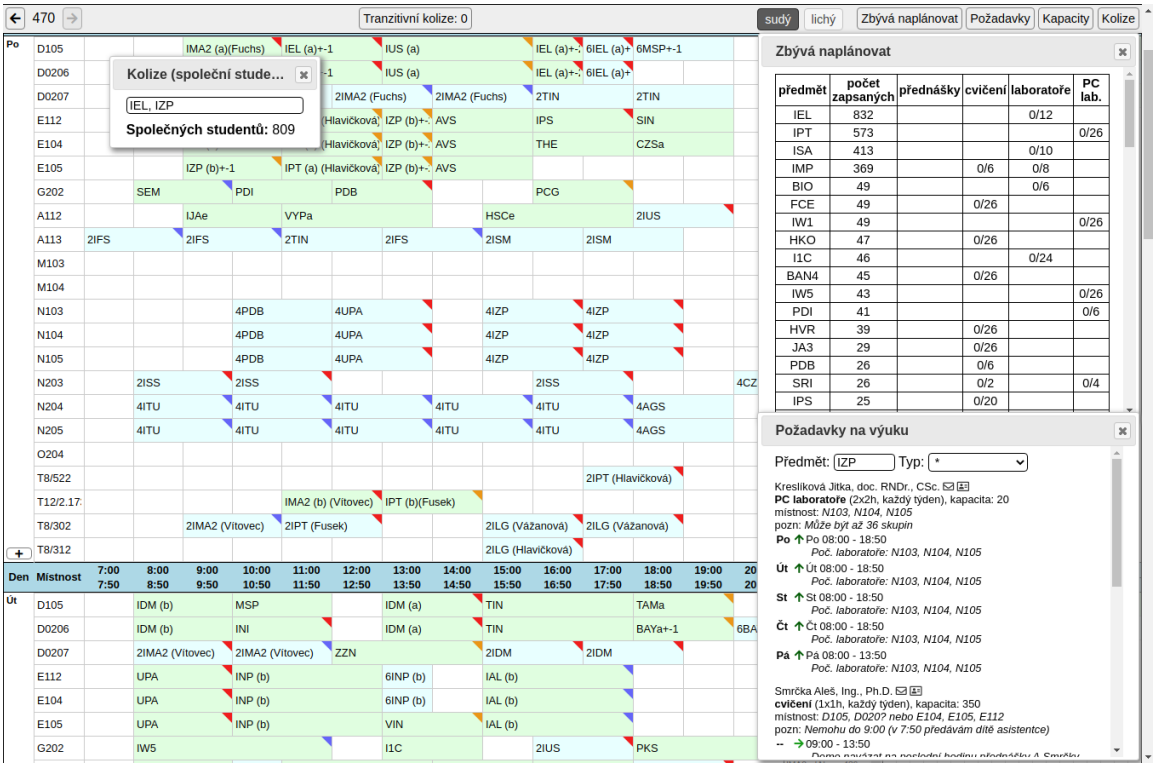
Následně stačí zpracovat tyto údaje jako při obvyklém přihlášení. V aplikaci zpracovává obě metody přihlášení (i registraci, pokud je povolena) třída `Authenticator`, která dále zajišťuje jednotný přístup k uživatelům.

Pokud se uživatel přihlásí přes účet Google, pokusí se `Authenticator` zjistit, zda již uživatel nemá v aplikaci účet založený přes klasickou registraci (pomocí porovnání e-mailové adresy nebo přezdívky). Když existuje, pokračuje uživatel s tímto účtem. Pokud se přihlašuje uživatel úplně poprvé, je mu z důvodu kompatibility vytvořen v databázi záznam s jeho e-mailem, jménem, přezdívkou (shodnou s loginem – část e-mailu před znakem „@“) a rolí studenta. Kdyby v budoucnu bylo potřeba uživatele přihlásit pomocí hesla, může mu je správce přidělit (změnit) nebo si je uživatel nastaví sám po přihlášení přes účet Google.

5.4 Nástroje

Pro podporu plánování slouží jednotlivé nástroje popsané v kapitole 4.6. Implementovány jsou jako samostatná dialogová okna pomocí knihovny jQuery UI, která zajišťuje možnost posouvání oken po obrazovce, změnu jejich velikosti a tedy jejich umístění dle přání uživatele. Je možné mít na obrazovce zobrazených několik různých nástrojů a tím si uspořádat pracovní plochu dle preferencí rozvrháře, jak je vidět na obrázku 5.2.

Po zavření nástrojového okna zůstanou vyplněny vstupy od uživatele a pokud by k zavření došlo nedopatřením, může se k rozpracovanému oknu uživatel vrátit. Tato vlastnost je výhodná zejména u nástroje *kapacity*, kde zůstávají uživateli vybrány místnosti pro umístění předmětu, a lze je tedy využít jako šablonu pro další okna.



Obrázek 5.2: Okna nástrojů

5.5 Kontroly a informace o rozvrhovém okně

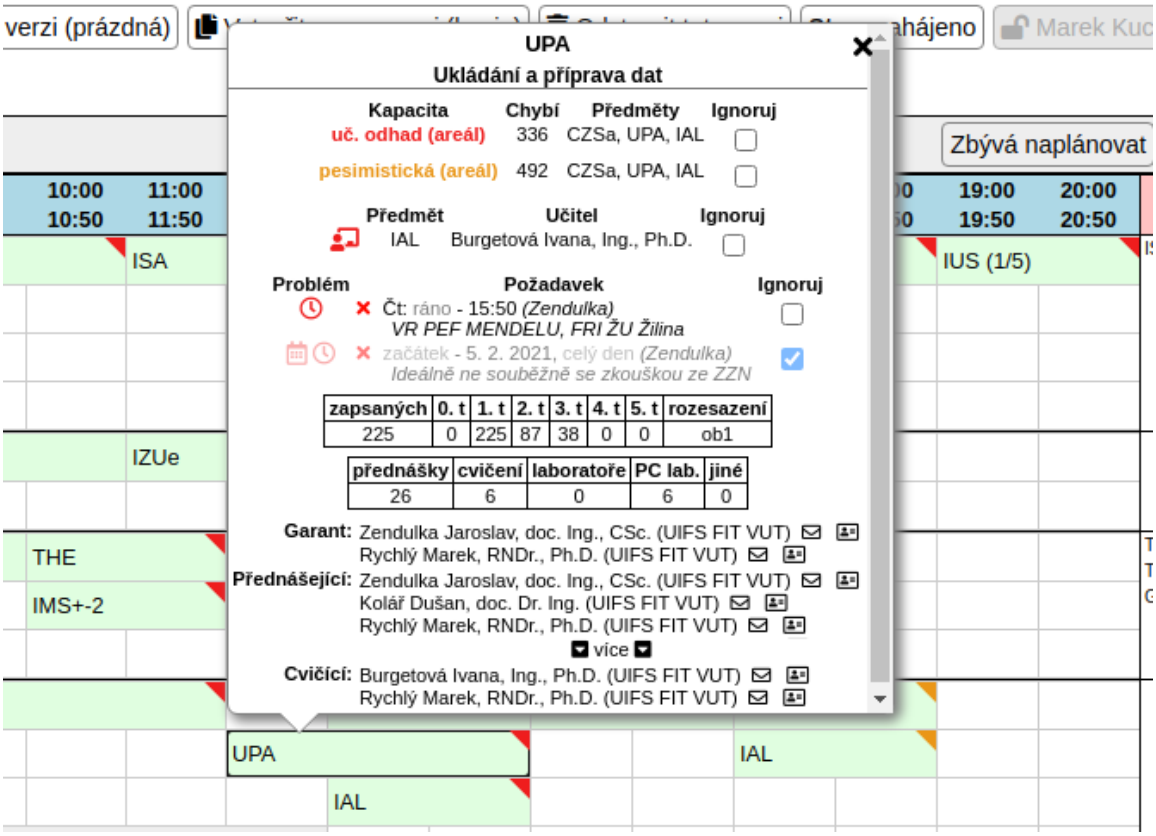
Jak bylo zmíněno již v návrhu, po kliknutí na každé rozvrhové okno se zobrazí nejdůležitější informace o předmětu a také problémy, které u okna jsou. Příklad takové informační bubliny je na obrázku 5.3.

Mezi zobrazené informace patří vyučující předmětu (garant, přednášející a cvičící), počty hodin plánované pro jednotlivé typy výuky (podle nich lze odvodit například délky přednášek nebo pravidelnost cvičení), počet zapsaných studentů, odhady počtů na jednotlivé termíny zkoušky a požadované rozesazení u zkoušky (to je získáno z požadavků vyučujících).

Problémy jsou zobrazovány různě podle jejich typu. Pokud se jedná o jednoduché informace jako například počet kolizních studentů dvou zkoušek, je zobrazeno pouze červené tlačítko, kterým lze tento problém ignorovat. Složitější typy jsou vypsány v řádcích. Na začátku je vždy symbol nebo text značící problém a jeho barva udává závažnost. Dále následuje komentář nebo data k problému a na konci řádku je zaškrtávací pole, přes které je možné konflikt ignorovat.

V rozvrhu se zobrazuje barevný informační rožek u oken, kde existuje nějaký neignorovaný problém. Jeho barva odpovídá jeho závažnosti a pokud je problémů více, má stejnou barvu jako nejzávažnější z nich. Barvy a závažnosti byly stanoveny následovně:

- **málo závažně** – porušený požadavek typu „chtěl bych“ a společní studenti ve volitelných předmětech;
- **středně závažně** – porušený požadavek na délku okna nebo počet skupin, společní studenti v povinném a volitelném předmětu, překročena kapacita místnosti pesimis-



Obrázek 5.3: Informace o rozvrhovém okně v bublině

tickým odhadem počtu studentů nebo je kapacita přednášky dostatečná pro 70–99 % studentů;

- **velmi závažně** – porušený požadavek typu „nemohu učit“, kolize vyučujících, společní studenti povinných předmětů, překročena kapacita reálným odhadem nebo skutečným počtem studentů a studenti s jinou zkouškou v jednom dni, zvláště z povinných předmětů.

Většina kontrol probíhá okamžitě po úpravě rozvrhového okna, a to převážně na straně klienta. Během implementace se velmi často řešil problém rozdílnosti struktury rozvrhu výuky a zkoušek, hlavně co se týče chování k místnostem. Jak již bylo naznačeno v kapitole 4.2, rozvrhová okna u výuky jsou ukládána jako samostatná, i když logicky patří k sobě (např. přednáška ve 3 místnostech propojených audiovizuální technikou nebo cvičení několika skupin v jeden čas). Oproti tomu se zkouškovými okny je nakládáno jako s jedním oknem, které má několik místností, avšak zde se zase liší pohledy s místnostmi a bez nich. Proto bylo tedy nutné a někdy velmi obtížné přizpůsobit veškeré kontroly pro každý z těchto tří uvedených typů.

5.6 Tranzitivní kolize

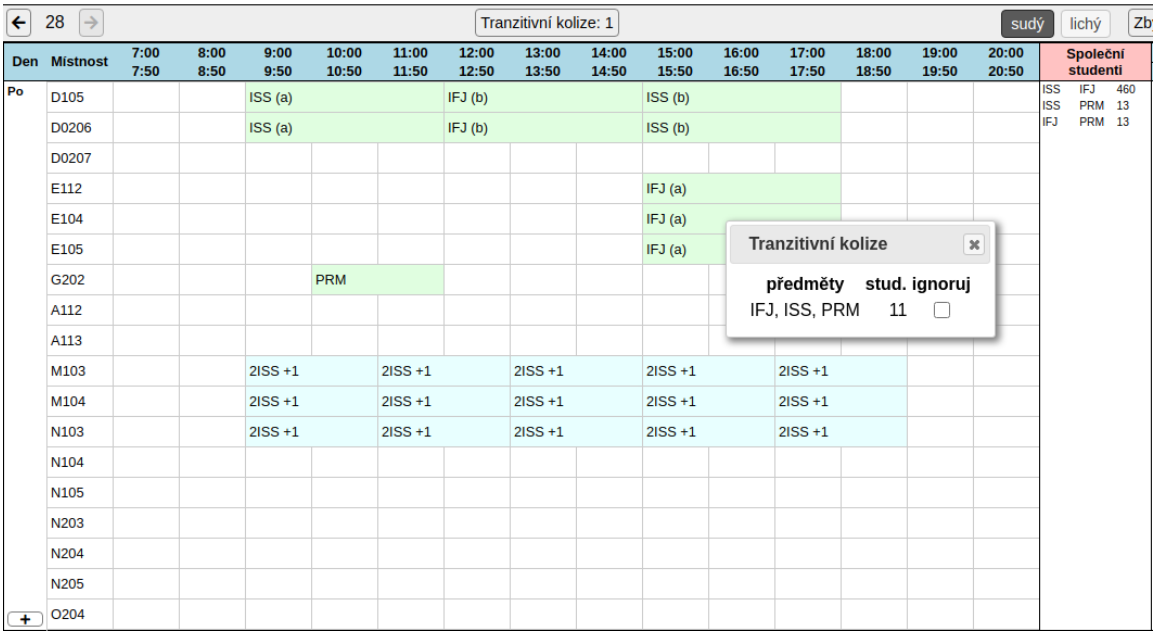
Pro výpočet tranzitivních kolizí rozvrhových oken bylo vyzkoušeno několik různých přístupů a algoritmů. Finální podoba se liší od návrhu v kapitole 4.5.

Výsledný algoritmus pro výpočet tranzitivních kolizí byl od základů vytvořen autorem a prošel několika vývojovými stádii. Základem bylo nejdříve nalezení všech oken, která mohou ovlivnit okno právě upravené (vložené, odstraněné), tedy těch, která se váží k předmětu, jenž má s upraveným alespoň jednoho společného studenta. Po nalezení všech těchto oken se pomocí metody *backtracking* pro každou skupinu předmětů se společnými studenty skládají okna předmětu (vždy jedno od každého typu a skupiny) do prázdné struktury rozvrhu. Pokud vznikne nekolizní rozvrh, je právě vložené okno bez tranzitivních kolizí.

Následně bylo měřením zjištěno, že hledání oken souvisejících s právě vloženým se v komplikovanějším rozvrhu stává časově náročnější než samotný výpočet tranzitivních kolizí. Navíc do těchto souvisejících oken byl nakonec často zahrnut celý rozvrh a algoritmus byl po změně okna zavolán i několikrát. Proto se hledání kolizních oken ve výsledném algoritmu vypustilo a tranzitivní kolize se vždy jednou po změně okna vypočítají pro celý rozvrh.

Zpočátku byla snaha počítat tranzitivní kolize v aplikaci na klientské straně, tedy v JavaScriptu. Veškerá potřebná data jsou zde k dispozici, ovšem problémem byla velká náročnost výpočtu spojená s chvilkovým zamrznutím aplikace, což se jevilo jako nepřípustné, a proto byl výpočet tranzitivních kolizí přesunut na server, se kterým je komunikováno pomocí asynchronních požadavků. Tím se docílilo plynulého běhu aplikace za cenu drobného zpoždění získání výsledku výpočtu, avšak to u tohoto druhu kontroly není problém.

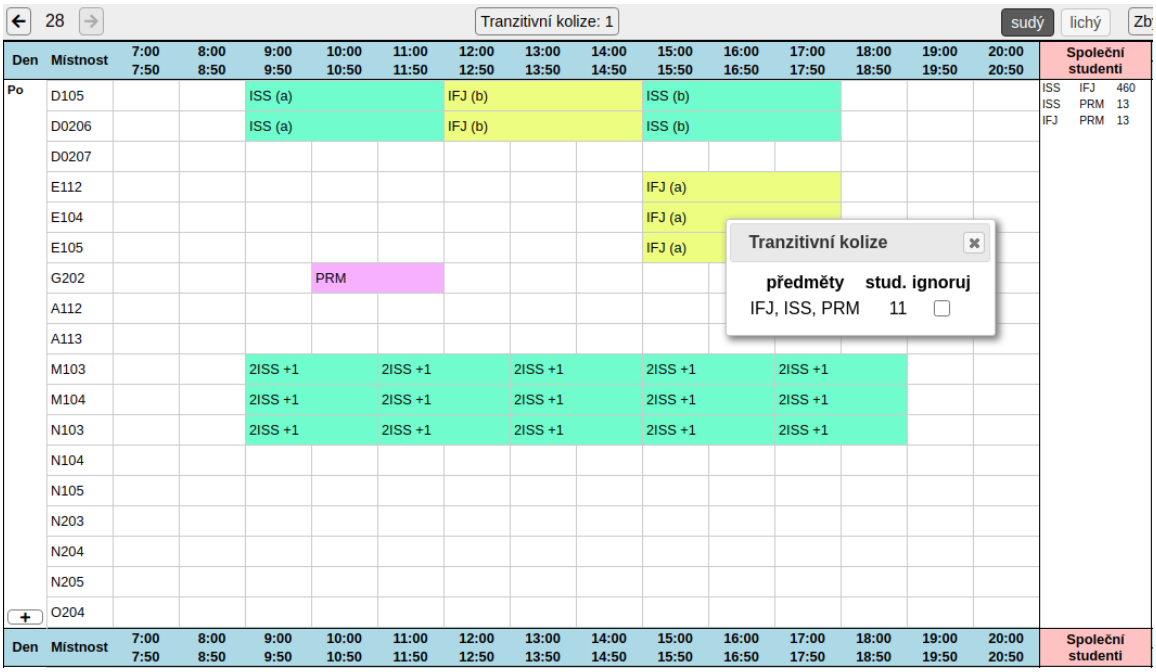
Hlavní změnou oproti návrhu je vizualizace tranzitivních kolizí. Návrh počítal se zobrazením varování u každého kolizního okna, což ovšem způsobilo, že při konfliktu např. 3 předmětů svítilo varování u několika (až desítek) oken (všechny přednášky i cvičení ze všech těchto předmětů). Tento způsob ztratil na významu i poté, co bylo rozhodnuto o přepočítávání vždy celého rozvrhu. Proto bylo hlášení kolizí přesunuto na jedno místo – doprostřed horního plovoucího menu, jak lze vidět na obrázku 5.4. Ve výpisu zobrazeném po kliknutí na tlačítko se vypíše seznam předmětů a počet studentů, které kolize ovlivní.



Obrázek 5.4: Zobrazení tranzitivních kolizí

Pro lepší orientaci v předmětech byla implementována funkce zvýraznění oken předmětů. Po kliknutí na seznam kolizních předmětů v okně tranzitivních kolizí se obarví všechny tyto

kolizní předměty v rozvrhu, okna jednoho předmětu vždy jednou barvou. Tato vlastnost je demonstrována na obrázku 5.5.



Obrázek 5.5: Obarvení oken kolizních předmětů

Kapitola 6

Testování

Testování aplikace probíhalo ve třech fázích, postupně od lokálního ladění autorem přes konzultace a verifikace s rozvrháři až po výsledné testování na produkčním serveru při skutečném plánování rozvrhů.

6.1 Průběžné testování při vývoji

Při implementaci aplikace byla funkcionalita autorem průběžně ověřována na obou typech rozvrhů, tedy na rozvrhu zkoušek i na rozvrhu výuky. Pro účely tohoto testování byla zreplikována plánovací databáze z produkčního serveru *minerva3* na lokální stroj autora. Rovněž aplikace A. Tesařové byla používána lokálně z důvodu omezení zásahů do ostré databáze a možnosti zobrazení změn v její testovací kopii.

Struktura rozvrhu

V počátcích vývoje (tedy při pouhém zobrazení rozvrhu) se jednalo pouze o porovnávání výsledků (vykresleného rozvrhu) s rozvrhem zobrazeným v aplikaci A. Tesařové nebo M. Morese. V rámci tohoto testování byly použity rozvrhy uložené v databázi z dříve provedených plánování. Během tohoto testování byly odladěny struktury pro uložení rozvrhových oken a zjištěny veškeré nejasnosti a komplikace, které nebyly odhaleny při analýze.

Následně se testovala editace rozvrhů, což probíhalo již na nově vytvořených verzích rozvrhu, aby zůstaly oficiální verze zachovány pro pozdější využití například při kontrolách. Zde již vznikly základní testovací scénáře, aby bylo možné veškerou funkcionalitu řádně a strukturovaně otestovat. Jednalo se o:

- odstranění okna z místnosti,
- vložení rozvrhového okna do místnosti,
- vložení rozvrhového okna do další místnosti (v případě zkoušek),
- odstranění dříve vloženého okna z místnosti,
- změna rozvrhového okna na jiné (např. kvůli změně délky),
- pokus o vložení konfliktního okna (jiné okno nebo hranice rozvrhu),
- pohyb zpět o několik verzí a
- pohyb vpřed o několik verzí.

Uživatelé a operace s rozvrhy

Pro účely další implementace a testování bylo nutné mít možnost provádět operace s rozvrhy (vytváření nových verzí, kopie a mazání). Navíc bylo vhodné mít již na počátku implementace jistotu, že uživatelé s nižšími právy (studenti) vidí a mohou provádět pouze to, co mají (např. mohou upravovat pouze vlastní rozvrhy). Otestovány byly také změny stavu rozvrhů, tedy že správce může rozvrh *zveřejnit* nebo *schválit* a student jej může pouze *navrhnout* ke zveřejnění.

Vše bylo důkladně otestováno dalším postupem v implementaci, a to zejména zamykání rozvrhů, změna stavu a vytváření kopií verzí. Některé rozvrhy byly vytvořeny přímo z pohledu studenta, aby bylo ověřeno, že celý proces proběhne i z jeho pohledu bez problémů.

Kontroly

Testování kontrol probíhalo vždy nejdříve na jednoduchých, uměle vytvořených rozvrzích. Zde byly vkládáním a měněním rozvrhových oken simulovány jak různé kolizní a extrémní případy, tak případy, ve kterých by kolize vzniknout neměla. Zároveň se testovala možnost ignorování hlášení těchto chyb společně se změnami při pohybu verzí, přidávání a ubírání oken, případně přidávání místností do jednoho okna v rozvrhu zkoušek.

Po vyladění byla každá z kontrol podrobena testu na skutečných rozvrzích. Po otevření rozvrhu byla okna s varováním manuálně zkontrolována, zda varování v daném případě dává smysl. Vycházelo se ze skutečnosti, že tyto rozvrhy by měly obsahovat velmi malé množství chyb, kolizí a varování, jelikož se skutečně použily k výuce nebo zkoušení. Během testování bylo však v těchto rozvrzích objeveno několik drobných nedostatků (například společní studenti na zkoušku v jednom dni), které předchozí aplikace neodhalily.

Nástroje

Při testování nástrojů se vycházelo z těch, které existovaly v podobné formě i v předchozích aplikacích. Jelikož aplikace využívají stejnou databázi, bylo možné zkontrolovat například že počet kolizních studentů v nástroji *kolize* je stejný s aplikací A. Tesařové.

Nástroje *rozestupy* a *zkoušky v týdnu* byly testovány proti odpovídajícím pohledům v „karuselu“ aplikace M. Morese. Výstup nástroje *požadavky* byl porovnáván s tabulkou požadavků v aplikaci A. Tesařové (viz obrázek 2.6).

6.2 Konzultace s rozvrháři

Průběžně během celého vývoje probíhaly konzultace s lidmi, kteří se návrhu rozvrhů na fakultě věnují. Především se jednalo o Ing. Jaroslava Dytrycha, Ph.D., který pravidelně poskytoval zpětnou vazbu k implementovaným částem aplikace. Schůzky se konaly téměř každých 14 dní a kvůli pandemii COVID-19 realizovaly většinou telekonferenčně.

Kontrolováno bylo převážně splnění zadání, tedy že chování nové či upravené části aplikace odpovídá požadavkům rozvrhářů. V rámci konzultací bylo vyřešeno několik nejasností a komplikací, které se během vývoje vyskytly, a byly objeveny chyby, které vznikly hlavně z důvodu nejasných významů některých dat. Na schůzkách byl také koordinován další postup a naplánováno ostré testování při plánování rozvrhů.

6.3 Ostré plánování rozvrhů

Zkoušky LS 2020/2021

Aplikace se stihla naimplementovat tak, aby se v ní mohl plánovat zkouškový rozvrh na letní semestr 2020/2021, což se realizovalo 6. března 2021. Při tomto ostrém testování byl autor připojen k hovoru rozvrhářů Ing. Jaroslava Dytrycha, Ph.D. a Bc. Kristýny Zaklové (z důvodu pandemie COVID-19 se plánování odehrálo telekonferenčně) a sledoval jejich práci s aplikací.

Během plánování se v aplikaci vyskytly jen dvě méně závažné chyby. První spočívala v chybném počítání odhadů počtu studentů na termíny zkoušek. Autor pochopil jinak data v databázi, protože budila dojem, že udávají k aktuálnímu předmětu již přepočítané odhady. Ovšem ve skutečnosti tato čísla znamenají loňské počty studentů a je nutné je pro správnou interpretaci poměrově upravit.

Druhá chyba se objevila v poslední implementované funkcionalitě – hlídání kapacit. Pokud předmět neměl specifikovány počty studentů na termíny, zobrazovalo se v informační bublině k překročené kapacitě *NaN* a toto překročení nebylo možné ignorovat.

Obě chyby byly během pár minut opraveny autorem. Zároveň rozvrháři vyslovili požadavek, že by rádi nástroj *zbývá naplánovat* využili i pro plánování zkoušek (původně byl určen pouze pro výuku). Byl tedy v průběhu plánování implementován i pro zkoušky a rozvrháři jej mohli většinu času ještě využívat.

Výuka ZS 2021/2022

Druhým ostrým testem prošla aplikace o víkendu 24. a 25. 4. 2021 při plánování rozvrhu výuky na zimní semestr 2021/2022. Začátek plánování doprovázely komplikace, které vznikly díky nekonzistentním datům v informačním systému FIT, ze kterého jsou získávána do aplikace A. Tesařové. Dlouhá doba načítání těchto dat a nepřipravenost aplikace na nově otevřené studijní programy způsobila asi sedmihodinové zdržení.

Během čekání na data bylo zjištěno, že se v nové plánovací aplikaci nezahrnuly odhady počtů studentů v prvních ročnících. V období plánování (duben až květen) totiž ještě nejsou k dispozici skutečné počty zapsaných studentů, proto by kontroly kolizí nefungovaly správně. Autorem byl tento nedostatek vyřešen a jelikož plánování stále nezačalo, nebyl jeho průběh problémem ovlivněn.

Během importování dat přes aplikaci A. Tesařové byly do databáze vloženy studijní programy, které neměly řádně vyplněné datum ukončení. V průběhu plánování rozvrhu zbůsobil tento fakt problém, protože v plánovací aplikaci byl daný program vyhodnocen jako neaktuální. Opravena byla data v databázi, ale zároveň byla aplikace upravena tak, aby tento problém již nevznikl.

Při používání nástroje *zbývá naplánovat* vyslovili uživatelé přání filtrovat nenaplánované předměty dle fakulty. Jelikož se rozvrh vytváří pouze pro předměty FIT, není nutné zobrazovat předměty z jiných fakult, které byly z IS FIT importovány také. Filtr byl autorem během krátké chvíle implementován a tak jej bylo možné nadále při plánování využívat.

Dalšími vylepšeními, která vznikla v průběhu plánování zimního rozvrhu, bylo zobrazení počtu kolizních studentů v loňském roce v nástroji *kolize* a hlídání studentských kolizí i u demonstračních cvičení. Původní požadavek na hlídání společných studentů se vztahoval pouze na přednášky, ovšem demonstrační cvičení jsou plánována podobně jako přednášky pouze jednou pro každý předmět či přednáškovou skupinu a tedy kolize jiné přednášky nebo demonstračního cvičení je nežádoucí.

Jedinou větší chybou, která se při plánování v aplikaci projevila, bylo zmizení okna po jeho vložení do rozvrhu za podmínky, že jej délkový modifikátor zkrátil na délku 0. Tato chyba byla opravena stanovením minimální délky okna na hodnotu 1.

Drobnou chybou byla nemožnost ignorovat porušení požadavků u některých oken ve více místnostech. Tento problém vznikl kombinací komplikovanosti zjišťování souvisejících oken a ověřování splněných požadavků – po zjištění, že požadavek není splněn, bylo ze souvisejících oken vybráno k označení ignorování špatné. Oprava byla provedena po ukončení sobotní části plánování, aby hledání chyby nenarušilo hladký průběh vytváření rozvrhu (po problémech s importem dat).

Po úspěšném vytvoření byl studentům díky implementovanému rozšíření rozvrh zveřejněn přímo v aplikaci. Odpadla tedy nutnost exportu přes aplikaci A. Tesařové. Toto řešení bylo velmi kladně oceněno již v průběhu plánování. Díky novému rozšíření bylo totiž možné rozvrh zobrazit i vyučujícím, se kterými se telefonicky ladily termíny výuky jednotlivých předmětů. Studentům bylo poté také umožněno přihlásit se do aplikace a vyzkoušet si rozvrh pozměnit nebo naplánovat úplně od začátku.

Výuka LS 2021/2022

Posledním ostrým testováním, které proběhlo, bylo vytváření rozvrhu na výuku v letním semestru 2021/2022 dne 8. 5. 2021.

Během umisťování oken se zjistilo, že nestačí délku výukových oken určovat pouze výpočtem z hodinové dotace. V požadavcích totiž učitelé někdy upřesňují, že potřebují na přednášku například o hodinu více. Tato úprava byla během chvíle implementována a délka výukových oken se nyní přednostně čte z požadavků. Na žádost uživatelů byly do odhadu kolizí studentů zahrnuty také předměty povinné nebo povinně volitelné v libovolném ročníku.

Dále byl s rozvrháři konzultován výpočet tranzitivních kolizí. Způsob jejich výpočtu způsobuje časté hlášení kolize kvůli nepravidelnému střídání dvou typů výuky u jednoho předmětu. Pro tento předmět nebylo možné algoritmem vytvořit bezkolizní rozvrh, avšak bylo potvrzeno, že principiálně je výpočet v pořádku. Z tohoto důvodu byl navržen lepší způsob ignorování těchto hlášení, díky kterému bude možné z kontroly tranzitivních kolizí vyjmout problematické předměty.

Kapitola 7

Závěr

V této diplomové práci byl popsán průběh tvorby rozvrhů pro výukové i zkouškové období na Fakultě informačních technologií Vysokého učení technického v Brně a uveden přehled programů, které za tímto účelem na fakultě vznikly a používají se. Na základě tohoto popisu a detailních informací získaných od osob, které rozvrhy na FIT vytváří, byla navržena nová aplikace, která si klade za cíl zjednodušit plánování samotné a zapojit v připomínkovém řízení do plánování také studenty.

Během návrhu nové aplikace byl kladen důraz na zachování již ověřených nástrojů a vzhledu uživatelského rozhraní pro plánování. Naopak byla snaha vyvarovat se chyb (funkčních i estetických), jež se vyskytly v aplikacích dříve vytvořených.

Vytvořený program asistuje rozvrháři při manuálním plánování rozvrhu tím, že kontroluje a varuje při vzniku konfliktů a porušení požadavků vyučujících a poskytuje mu několik užitečných nástrojů. Kvalitu vytvořeného rozvrhu je možné ohodnotit na základě počtu chyb či varování, které se v rozvrhu vyskytují. Do plánování je také možné zapojit studenty, kteří si mohou vytvořit vlastní rozvrh od začátku nebo využít možnosti upravit kopii zveřejněného rozvrhu a změnit jen několik rozvrhových oken.

Aplikace byla otestována při vytváření rozvrhů na letní zkouškové období 2020/2021 a na výuku v zimním i letním semestru 2021/2022 společně s rozvrháři Ing. Jaroslavem Dytrychem, Ph.D. a Bc. Kristýnou Zaklovou. Aplikace se setkala s velmi kladnými ohlasy a rozvrhy byly s její pomocí úspěšně naplánovány. Největším přínosem byla dle uživatelů její rychlost a spolehlivost. Také kontroly a nástroje, kterými aplikace uživateli při plánování rozvrhů pomáhá, jsou na vysoké úrovni. Aplikace bude na FIT nadále využívána pro plánování rozvrhů a bude zváženo její použití i na jiných fakultách VUT.

V rámci implementace vzniklo také rozšíření funkcionality, které bylo otestováno a použito na rozvrzích pro zimní i letní semestr 2021/2022. Podstatou tohoto rozšíření je zobrazení zveřejněných rozvrhů přímo v aplikaci bez nutnosti generovat soubory ve formátu HTML přes aplikaci A. Tesařové.

Dalším rozšířením funkcionality implementované aplikace může být například přidání dalších nástrojů, automatizace některých kroků plánování nebo vytvoření samostatného modulu. Za jeden z takových modulů lze považovat tvorbu osobních rozvrhů studentů, ve kterém by si studenti mohli na základě zveřejněného rozvrhu upravit do přehledné formy svůj vlastní rozvrh a poté jej vytisknout, případně importovat do svého oblíbeného kalendáře. Dále by se mohlo jednat o začlenění aplikací pro organizaci zkoušek, například plánování dozorů nebo rozmístění studentů v místnostech a přípravu zadání.

Literatura

[1] Bennetch, I., Bansod, D., Ungureanu, D. et al. *PhpMyAdmin* [online]. Dostupné z: https://www.phpmyadmin.net/.

[2] Braun, A. R., Palmer, R. a Terlson, B. *ECMAScript 2020 Language Specification* [online]. [cit. 2020-12-17]. Dostupné z: https://www.ecma-international.org/ecma-262/.

[3] Brown, D., Lopes, N., Pena, F. et al. *PHP: Hypertext Preprocessor* [online]. Dostupné z: https://www.php.net/.

[4] Cooper, T. B. a Kingston, J. H. The Complexity of Timetable Construction Problems. 1996. Dostupné z: http://jeffreykingston.id.au/tt_papers/cooper_kingston_1996_complexity.pdf.

[5] Čillo, V. *Program pro plánováná rozvrhů*. Brno, CZ, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/19244/.

[6] Horký, A. *Systém pro pokročilé plánování*. Brno, CZ, 2015. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/17111/.

[7] Hsiao Lan, F. *Genetic Algorithms in Timetabling and Scheduling*. Edinburgh, UK, 1994. Disertační práce. University of Edinburgh, Department of Artificial Intelligence. Dostupné z: https://era.ed.ac.uk/handle/1842/30185.

[8] Kubalcová, M. *Porovnání programů pro plánování rozvrhů a zkoušek*. Brno, CZ, 2012. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/13230/.

[9] Mores, M. *Program pro plánování rozvrhů*. Brno, CZ, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22888/.

[10] OpenJS Foundation. *JQuery* [online]. Dostupné z: https://jquery.com/.

[11] Procházka, D. *PHP 6: začínáme programovat*. 1. vyd. Praha: Grada Publishing, 2012. ISBN 978-80-247-3899-4.

[12] Písek, S. *HTML: začínáme programovat*. 3. vyd. Praha: Grada Publishing, 2010. ISBN 978-80-247-3117-9.

[13] Q Success. *W3Techs – World Wide Web Technology Surveys* [online]. [cit. 2020-12-17]. Dostupné z: https://w3techs.com/.

[14] Sklar, D. *PHP 7: Praktický průvodce nejrozšířenějším skriptovacím jazykem pro web*. 1. vyd. Brno: Zoner Press, 2018. ISBN 978-80-7413-363-3.

[15] solid IT gmbh. *DB-Engines – Knowledge Base of Relational and NoSQL Database Management Systems* [online]. [cit. 2020-12-18]. Dostupné z: https://db-engines.com/.

[16] *Studijní a zkušební řád VUT* [online]. [cit. 2020-12-08]. Dostupné z: https://www.vutbr.cz/uredni-deska/vnitrni-predpisy-a-dokumenty/studijni-a-zkusebni-rad-vut-d149085.

[17] Tesařová, A. *Program pro podporu plánování rozvrhů*. Brno, CZ, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/21894/.

[18] Vosáhlo, D. *Program pro plánování rozvrhů*. Brno, CZ, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22080/.

[19] W3C. *World Wide Web Consortium* [online]. Dostupné z: https://www.w3.org/.

[20] Žára, O. *JavaScript: Programátorské techniky a webové technologie*. 1. vyd. Brno: Computer Press, 2015. ISBN 978-80-251-4573-9.

Příloha A

Obsah přiloženého média

- `/`
  - `aplikace/`
    - `API/` – zdrojové kódy pro API (server)
    - `CSS/` – styly pro definici vhledu
    - `Entity/` – objekty používané na serveru
    - `Exception/` – definice tříd výjimek
    - `IMG/` – obrázky
    - `JS/` – třídy a soubory v jazyce JavaScript (klient)
    - `Libs/` – knihovny
    - `Page/` – šablony pro vykreslení stránek
      - `Fragment/` – moduly stránek
      - `View/` – stránky pro veřejné zobrazení rozvrhů
    - `Repository/` – repozitáře pro práci s databází
    - `Sessions/` – úložiště pro sezení
    - `.htaccess` – konfigurace serveru Apache
    - `Authenticator.php` – třída pro autentizaci
    - `Router.php` – třída pro směrování
    - `config-example.ini` – příklad konfiguračního souboru
    - `index.php` – vstupní bod aplikace
    - `favicon.ico` – ikona aplikace
    - `SQL-data.sql` – vzorová data
    - `SQL-structure.sql` – inicializační SQL skript
  - `zprava/`
    - `bib-styles/` – bibliografické styly
    - `obrazky-figures/` – obrázky a diagramy
    - `template-fig/` – loga VUT
    - `Makefile` – překlad zprávy
    - `fitthesis.cls` – šablona zprávy pro FIT
    - `xkuchy00-IS-planovani-rozvrhu-01-kapitoly.tex` – kapitoly
    - `xkuchy00-IS-planovani-rozvrhu-20-literatura.bib` – bibliograf. databáze
    - `xkuchy00-IS-planovani-rozvrhu-30-přílohy.tex` – přílohy
    - `xkuchy00-IS-planovani-rozvrhu.pdf` – zpráva v PDF
    - `xkuchy00-IS-planovani-rozvrhu-tisk.pdf` – zpráva v PDF – verze pro tisk
    - `xkuchy00-IS-planovani-rozvrhu.tex` – hlavní .tex soubor
    - `zadani.pdf` – zadání práce
  - `plakat.pdf` – plakát
  - `plakat.svg` – zdrojová podoba plakátu

Příloha B

Instalace

V této příloze bude popsán postup instalace aplikace. Jelikož se jedná o webovou aplikaci, bude nutné mít zprovozněný webový server, nejlépe Apache. Jeho součástí musí být **PHP ve verzi 7.2** nebo vyšší. Pro ukládání dat je potřebný databázový server **MySQL** podporující *InnoDB* a *MyISAM*.

Aplikaci je třeba umístit tak, aby měl webový server přístup do její kořenové složky (na dodaném médiu je to složka `aplikace/`) a veškeré požadavky směřovaly na soubor `index.php` (to zajišťuje předpřipravený konfigurační soubor `.htaccess`). Navíc je nutné podniknout následující kroky:

- Složka `Sessions/` musí být zapisovatelná pro webový server (většinou uživatel a skupina `www-data` na Linuxovém serveru).
- Soubor `config-example.ini` je nutné přejmenovat na `config.ini` a upravit jej (viz dále).
- Přejmenovaný soubor `config.ini` by z bezpečnostních důvodů neměl být přístupný z webu.
- Pokud bude povoleno zapisování chyb z aplikace do souboru, musí tyto soubory existovat a být zapisovatelné pro webový server.

Pro běh aplikace je nezbytné založit MySQL databázi a poté v ní vytvořit veškeré potřebné tabulky, případně nahrát vzorová data, což lze provést pomocí přiložených SQL skriptů `SQL-structure.sql` a `SQL-data.sql`, které jsou umístěné na dodaném médiu ve složce `aplikace/`. V databázi musí existovat uživatel, jenž bude mít oprávnění číst a zapisovat do všech tabulek v založené databázi.

Informace o připojení k databázi (adresu, port, přihlašovací jméno a heslo) je nezbytné vyplnit v konfiguračním souboru `config.ini`. Dále je v tomto souboru vhodné nastavit politiku přihlašování a registrování nových uživatelů a období, ve kterém se plánují jednotlivé rozvrhy. Pro produkční verzi je také nutné upravit zobrazování chyb (v dodaném souboru se jedná o nastavení pro testování).