

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**  
**FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5384-56111

**ROZVRHOVÝ SYSTÉM PRE VYSOKÉ ŠKOLY**

**Diplomová práca**

Študijný program:	Aplikovaná informatika
Číslo študijného odboru:	2511
Názov študijného odboru:	9.2.9 Aplikovaná informatika
Školiace pracovisko:	Ústav informatiky a matematiky
Vedúci záverečnej práce:	Ing. Ondrej Gallo

**Bratislava 2014**

**Bc. Emília Knapereková**





## ZADANIE DIPLOMOVEJ PRÁCE

Študentka: **Bc. Emília Knapereková**  
ID študenta: 56111  
Študijný program: Aplikovaná informatika  
Študijný odbor: 9.2.9 aplikovaná informatika  
Vedúci práce: Ing. Ondrej Gallo  
Miesto vypracovania: Ústav matematiky a informatiky

Názov práce: **Rozvrhový systém pre vysoké školy**

Špecifikácia zadania:

V súčasnosti existuje viacero systémov na tvorbu rozvrhu. Ich hlavným nedostatkom je, že neriešia všetky problémy spojené s jeho tvorbou. Je to spôsobené aj tým, že tvorba rozvrhu je špecifický problém pre danú školu/inštitúciu. Preto rozvrhár radšej použije namiesto komplexného profesionálneho softvéru manuálny prístup (napr. Excelovskú tabuľku). Cieľom práce je vytvoriť rozvrhový systém vhodný hlavne pre FEI STU.

Úlohy:

1. Naštudujte si problematiku tvorby vysokoškolského rozvrhu.
2. Vytvorte návrh, ktorý bude tvoriť základ pre vlastný rozvrhový systém umožňujúci tvorbu skúškového rozvrhu ako aj rozvrhu hodín na semester.
3. Prispôbte návrh systému tak, aby sa dal neskôr rozšíriť o rôzne typy algoritmov na generovanie rozvrhu.
4. Zvoľte vhodné technológie a implementujte jadro rozvrhového systému.

Zoznam odbornej literatúry:

1. Summerfield, M. *Python 3 Výukový kurz*. Brno: Computer Press, 2010. 584 s. ISBN 978-80-251-2737-7.
2. Kanisová, H. – Müller, M. *UML srozumiteľne*. Brno: Computer Press, 2007. 176 s. ISBN 80-251-1083-4.





Riešenie zadania práce od: 23. 09. 2013

Dátum odovzdania práce: 23. 05. 2014



**Bc. Emília Knapereková**  
študentka

  
**prof. RNDr. Otokar Grošek, PhD.**  
vedúci pracoviska

  
**prof. RNDr. Otokar Grošek, PhD.**  
garant študijného programu



*Podpísaná Bc. Emília Knapereková čestne vyhlasujem, že som diplomovú prácu „Rozvrhový systém pre vysoké školy“ vypracovala na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci.*

*Uvedenú prácu som vypracovala pod vedením Ing. Ondreja Galla.*

*V Bratislave dňa 21.05.2014*

.....

*Podpis autora*





*Z celého srdca ďakujem*

*rodičom za to, že ma podporovali počas celého štúdia a viedli k zodpovednosti a cieľavedomosti,*

*všetkým súrodencom a priateľom, ktorí mi boli veľkou oporou najmä v závere písania tejto práce, menovite Lenke, Macovi, Zuzkám (N., H. a Č.), Becky a zvlášť bratovi Jojovi za odborné diskusie a cenné rady,*

*integrátorke AIS STU, Ing. Andrei Bujdákovej, za promptnosť a ústretovosť počas vývoja modulu pre import dát FEI STU,*

*osobitne vedúcemu tejto práce, Ing. Ondrejovi Gallovi, za podnetné konzultácie a ochotu pri riešení rôznych problémov spojených s jej vypracovaním.*



# SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Aplikovaná informatika
Diplomová práca:	Rozvrhový systém pre vysoké školy
Autor:	Bc. Emília Knapereková
Vedúci záverečnej práce:	Ing. Ondrej Gallo
Miesto a rok predloženia práce:	Bratislava 2014

Cieľom tejto diplomovej práce je preskúmať oblasť tvorby rozvrhov, navrhnúť a vytvoriť aplikáciu na tvorbu vysokoškolských rozvrhov, špecializovanú predovšetkým na potreby FEI STU. Práca opisuje problémy vznikajúce počas procesu tvorby rozvrhov. Analyzuje pozitíva a negatíva vybraných existujúcich softvérových riešení. Podstatnú časť práce tvorí zber požiadaviek a špecifikácia komplexného softvérového systému, ktorý pokrýva všetky potreby jednotlivých fáz procesu tvorby rozvrhu na semester aj skúškové obdobie. Modulárny návrh rozvrhového systému vytvára možnosti aj pre ďalšie rozšírenia. Implementačná časť práce sa zameriava najmä na základnú funkcionálnu nevyhnutnú pre pohodlnú tvorbu rozvrhu na semester aj skúškové obdobie.

Kľúčové slová: Vysokoškolský rozvrh. Požiadavky na rozvrh. Tvorba rozvrhu. Aplikácia.



# ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA  
FACULTY OF ELECTRONICAL ENGINEERING AND INFORMATION  
TECHNOLOGY

Study Program:	Applied Informatics
Master Thesis:	Timetable Creator for Universities
Author:	Bc. Emília Knapereková
Supervisor:	Ing. Ondrej Gallo
Place and year of submission:	Bratislava 2014

The purpose of this thesis is to explore the area of timetable creation, to design and build an application for creation of university timetables, specialized primarily to the needs of FEI STU. This paper describes the problems arising during the process of timetable creation. It analyzes pros and cons of selected existing software solutions. A substantial part of the work consists of gathering requirements and specification of complex software system covering all the needs in various stages of timetable creation workflow, suitable for both the semester and examination period timetable. Modular design of the system creates opportunities for further extensions. Implementation part primarily focuses on the core functionality necessary for convenient creation of both the semester and examination period timetable.

Keywords: University timetable. Timetable requirements. Timetable creation. Application.



# OBSAH

Úvod.....	1
1 Analýza problémovej oblasti.....	3
1.1 Definícia pojmov.....	3
1.2 Nástroje na tvorbu vysokoškolského rozvrhu.....	5
1.2.1 Roger.....	5
1.2.2 FET.....	10
1.2.3 Wise Timetable.....	13
1.2.4 Zhrnutie.....	15
1.3 Tvorba rozvrhu na FEI.....	17
2 Cieľ práce.....	19
3 Špecifikácia požiadaviek.....	21
3.1 Postup tvorby rozvrhu.....	21
3.2 Prípady použitia.....	24
3.2.1 Rozvrh na semester aj skúškové obdobie.....	24
3.2.2 Import a úprava dát z UIS.....	24
3.2.3 Konfigurácia systému.....	24
3.2.4 Definovanie požiadaviek na rozvrh.....	25
3.2.5 Generovanie rozvrhu.....	26
3.2.6 Manuálna úprava rozvrhu.....	26
3.2.7 Zobrazenie kolíznych matíc.....	26
3.2.8 Kontrola kolízií.....	27
3.2.9 Prezeranie výsledného rozvrhu.....	27
3.2.10 Export výsledného rozvrhu do UIS.....	27
3.2.11 Rezervácia miestnosti.....	27
3.2.12 Navrhovanie zmien v rozvrhu.....	28
3.2.13 Zálohovanie rozvrhu.....	28

3.2.14 Priradenie oprávnení používateľom.....	28
3.3 Požiadavky na rozvrh.....	29
4 Návrh.....	35
4.1 Architektúra systému.....	35
4.2 Balíček „Data“ .....	37
4.3 Balíček „Timetable“ .....	40
4.4 Balíček „Calendar“ .....	41
4.5 Balíček „Requirements“ .....	43
4.6 Balíček „FEI“ .....	45
4.7 Balíček „Settings“ .....	46
4.8 Balíček „Authentication“ .....	46
5 Implementácia.....	47
5.1 Výber programovacích jazykov a knižníc.....	47
5.2 Štruktúra projektu.....	49
5.3 Model údajov.....	52
5.4 Riešenie vybraných implementačných problémov.....	53
5.4.1 Dedičnosť modelov.....	54
5.4.2 Smerovanie.....	56
5.4.3 Pohľady.....	57
5.4.4 Dynamické smerovanie a dynamické pohľady.....	59
5.4.5 Inštalácia modulov typov požiadaviek.....	61
5.4.6 Uchovávanie dní a týždňov.....	61
5.5 Nasadenie aplikácie.....	62
6 Overenie riešenia.....	63
6.1 Testovanie „čierna skrinka“ .....	63
6.2 Testovanie „biela skrinka“ .....	65
6.2.1 Overenie funkcie pre kontrolu kolízií.....	65
7 Zhodnotenie.....	69
8 Záver.....	71
9 Literatúra.....	73



Príloha A: Technická dokumentácia.....	I
A.1 Požiadavky na hardvér a softvér.....	I
A.2 Inštalácia rozvrhového systému.....	I
Krok 1: koreňový adresár.....	II
Krok 2: inštalácia a aktivácia virtuálneho prostredia.....	II
Krok 3: základná konfigurácia Django projektu.....	II
Krok 4: statické súbory.....	II
Krok 5: médiá.....	II
Krok 6: konfigurácia webového servera.....	III
Krok 7: databáza.....	III
Krok 8: import dát.....	IV
Krok 9: konfigurácia rozvrhového systému.....	IV
Krok 10: vytvorenie používateľského konta.....	IV
A.3 Nastavenia rozvrhového systému.....	IV
A.3.1 Databázové konštanty.....	IV
A.3.2 Semester a skúškové.....	V
A.3.3 Všeobecné nastavenia.....	V
A.3.4 Nastavenia aplikácie „data“.....	V
A.3.5 Nastavenia aplikácie „timetable“.....	VI
A.3.6 Nastavenia aplikácie „requirements“.....	VI
A.3.7 Nastavenia aplikácie „calendar“.....	VI
A.3.8 Nastavenia aplikácie „fei“.....	VI
A.4 Formát zdrojových dát pre modul „FEI“.....	VII
Príloha B: Používateľská príručka.....	XI
B.1 Časti grafického používateľského rozhrania.....	XI
B.2 Nastavenie časovej mriežky.....	XII
B.3 Možnosti zobrazenia rozvrhu.....	XIII
B.4 Postup tvorby rozvrhu.....	XV
Príloha C: Obsah elektronického nosiča.....	XVII



# ZOZNAM OBRÁZKOV A TABULIEK

Obr. 1: UML diagram aktivít znázorňujúci postup tvorby rozvrhu – 1. časť.....	22
Obr. 2: UML diagram aktivít znázorňujúci postup tvorby rozvrhu – 2. časť.....	23
Obr. 3: Diagram balíčkov (zoskupenia tried) – architektúra systému.....	35
Obr. 4: Triedy balíka „Data“.....	39
Obr. 5: Triedy balíka „Timetable“.....	41
Obr. 6: Triedy balíka „Calendar“.....	42
Obr. 7: Triedy balíka „Requirements“.....	43
Obr. 8: Hierarchia dedičnosti pohľadov tools.views.....	59
Obr. 9: Hierarchia dedičnosti pohľadov mtimetables.views.....	60
Obr. 10: Hierarchia dedičnosti pohľadov mtimetables.misc.....	61
Obr. 11: Výstup programu LinkChecker 9.2.....	66
Obr. 12: Distribúcia udalostí v čase pre testy 1 – 4.....	68
Obr. 13: Distribúcia udalostí v čase pre testy 5 – 8.....	68
Obr. 14: Distribúcia udalostí v čase pre testy 9 – 12.....	68
Obr. 15: Testovanie funkcie pre kontrolu kolízií – výstup z programu Cricket 0.2.3.....	70
Obr. 16: Hlavné časti grafického používateľského rozhrania.....	XI
Obr. 17: Úprava časovej mriežky na skúškové obdobie.....	XIII
Obr. 18: Filter rozvrhových akcií.....	XIV
Obr. 19: Zobrazenie rozvrhu – deň (angl. Day).....	XIV
Obr. 20: Zobrazenie rozvrhu – týždeň (angl. Week).....	XIV
Obr. 21: Zobrazenie rozvrhu – mesiac (angl. Month).....	XV
Tab. 1: Porovnanie vlastností rozvrhových systémov Roger, FET a Wise Timetable.....	16
Tab. 2: Množiny kolíznych udalostí pre testy 1 – 8.....	69
Tab. 3: Množiny kolíznych udalostí pre testy 9 – 12.....	69



# ZOZNAM SKRATIEK A ZNAČIEK

**angl.** – anglicky – skratka, po ktorej nasleduje výraz v angličtine.

**UIS** – Univerzitný informačný systém – všeobecné označenie pre informačný systém využívaný ľubovoľnou fakultou, vysokou školou alebo univerzitou.

**AIS** – Akademický informačný systém – informačný systém používaný na FEI STU.

**URL** – angl. *Uniform Resource Locator* – všeobecné označenie pre internetovú adresu.

**UML** – angl. *Unified Modeling Language* – grafický jazyk používaný v softvérovom inžinierstve na špecifikáciu, návrh a dokumentáciu programov.

**CSV** – angl. *Comma-separated values* – jednoduchý formát pre ukladanie tabuľkových dát (každý záznam na novom riadku, polia záznamu oddelené čiarkou alebo bodkočiarkou).

**XML** – angl. *eXtensible Markup Language* – značkovací jazyk pre ukladanie rôznych typov štruktúrovaných údajov.

**AJAX** – angl. *Asynchronous JavaScript + XML* – označenie technológie vývoja interaktívnych webových aplikácií.

**ORM** – angl. *Object-relational mapping* – je programovacia technika poskytujúca rozhranie pre automatickú konverziu dát medzi relačnou databázou a objektovo orientovaným programovacím jazykom.

**API** – angl. *Application programming interface* – množina funkcií, tried, knižníc, a programov, ktoré poskytujú rozhranie pre použitie s inými aplikáciami.

**CBV** – angl. *Class Based Views* – systém tvorby pohľadov vo *framework-u* Django.

**GUI** – angl. *Graphical User Interface* – grafické používateľské rozhranie.



# ÚVOD

Každá vysoká, stredná aj základná škola každoročne musí riešiť otázky vznikajúce pri tvorbe rozvrhu hodín. Na vysokých školách a univerzitách je tento problém ešte o niečo zložitejší – rozvrh treba robiť častejšie (väčšinou štyrikrát za rok – osobitne zimný a letný semester, osobitne rozvrh na semester a rozvrh na skúškové obdobie), počty učiteľov, študentov, predmetov aj miestností (ktoré sa môžu nachádzať aj vo viacerých budovách) sú niekoľkonásobne vyššie, študenti toho istého ročníka aj študijného odboru nemusia mať zapísané rovnaké predmety (kvôli opakovaniu predmetov, modifikáciám študijného plánu, atď.). Neexistuje jednoznačný postup na to, ako načasovať výuku jednotlivých predmetov tak, aby vyhovovala všetkým učiteľom aj študentom a zároveň nedochádzalo ku kolíziám v rozvrhu.

K problémom tvorby vysokoškolských rozvrhov som sa prvýkrát dostala vo svojej bakalárskej práci [7], ktorá prezentuje alternatívny spôsob zápisu študentov na cvičenia z jednotlivých predmetov (v už vytvorenom rozvrhu hodín na semester). Natrafila som na vážne nedostatky samotného procesu vytvárania termínov cvičení, prednášok aj skúšok a začala som rozmýšľať, ako celý tento proces zjednotiť, prepojiť, zjednodušiť a urýchliť. V rámci predmetov Tímový projekt 1 a 2 sme spolu s ďalšími kolegami začali pracovať na návrhu nového rozvrhového systému [8]. Vzhľadom na časovú náročnosť samotného zberu požiadaviek a návrhu jadra systému, neostal priestor na implementáciu. Preto som sa rozhodla pokračovať vo vývoji komplexného rozvrhového systému aj vo svojej diplomovej práci.

Cieľom tejto práce je naštudovať možnosti existujúcich profesionálnych programov na tvorbu vysokoškolských rozvrhov, dôkladne zanalyzovať a doplniť existujúci návrh rozvrhového systému z tímového projektu [8] a implementovať základnú funkcionality aplikácie pre tvorbu rozvrhu na semester aj skúškové obdobie.





# 1 ANALÝZA PROBLÉMOVEJ OBLASTI

Na riešenie zložitého problému tvorby školských rozvrhov v súčasnosti existuje viacero softvérových nástrojov. Väčšinou obsahujú príjemné a prehľadné používateľské rozhranie, ktoré uľahčuje manuálnu tvorbu rozvrhu a pomáha zabrániť nechceným kolíziám. Niektoré aplikácie ponúkajú aj možnosť viac či menej inteligentného generovania rozvrhu na základe vstupných dát (množiny miestností, predmetov, vyučujúcich, študentov, študijných skupín, atď.) a požiadaviek na umiestnenie rozvrhových akcií. Generátory rozvrhu využívajú rôzne typy algoritmov umelej inteligencie – grafové algoritmy, algoritmy s rekurzívnym vymieňaním (tzv. *Recursive Swapping*), evolučné algoritmy, simulované žíhanie (tzv. *Simulated Annealing*), rôzne heuristické metódy a iné.

Úlohou rozvrhových systémov a osôb poverených tvorbou rozvrhu je nájsť optimálne rozmiestnenie prednášok a cvičení z jednotlivých predmetov vzhľadom na zadané požiadavky. Nutným predpokladom dobrého softvéru je teda široká škála typov požiadaviek, ktoré dokážu v čo najväčšej miere pokryť reálne požiadavky všetkých zainteresovaných subjektov. Nemenej dôležité je vhodné nastavenie priorít pre typy požiadaviek aj požiadavky jednotlivcov.

V prvej kapitole uvádzame definície pojmov, ktoré sú dôležité pre správne pochopenie nasledujúcich častí práce. Podstatná časť tejto kapitoly je venovaná podrobnej analýze troch vybraných profesionálnych nástrojov na tvorbu vysokoškolského rozvrhu – Roger, FET a Wise Timetable. Záverečná časť stručne popisuje súčasnú situáciu tvorby rozvrhu na FEI STU.

## 1.1 Definícia pojmov

**Rozvrhár** je osoba poverená tvorbou školských rozvrhov.

**Rozvrhová akcia** je udalosť v čase a priestore – konkrétny typ výuky konkrétneho predmetu alebo iná aktivita bez viazanosti na predmet, ktorá sa uskutočňuje v konkrétnom

čase na konkrétnom mieste (množina miestností). V závislosti od typu môže (ale nemusí) vyžadovať účasť konkrétnej množiny osôb (študentov a/alebo učiteľov).

**Kolízia rozvrhových akcií** vzniká, ak sa dve (alebo viac) rozvrhové akcie konajú v navzájom prekrývajúcich sa časových intervaloch a zároveň je splnená minimálne jedna z nasledovných podmienok:

- množiny miestností daných rozvrhových akcií obsahujú aspoň jednu rovnakú miestnosť,
- množiny osôb (učiteľov a študentov) daných rozvrhových akcií obsahujú aspoň jednu rovnakú osobu.

**Kolízna matica** je trojuholníková matica, ktorej prvky vyjadrujú počty spoločných objektov pre jednotlivé dvojice predmetov. Na x-ovej aj y-ovej osi sa nachádzajú zoznamy predmetov. Význam číselných hodnôt závisí od typu kolíznej matice:

- kolízna matica študentov – počet spoločných študentov pre dvojicu predmetov,
- kolízna matica vyučujúcich – počet spoločných vyučujúcich pre dvojicu predmetov,
- kolízna matica študijných skupín – počet spoločných študijných skupín pre dvojicu predmetov.

**Časový rámec** predstavuje tabuľku časových preferencií subjektu (napr. používateľa alebo miestnosti). Umožňuje definovať časové intervaly, v ktorých sa subjekt môže alebo nemôže zúčastňovať na výuke.

**Typ aktivity** určuje kategóriu rozvrhovej akcie. Zahŕňa typy výuky predmetov (prednáška, cvičenie, laboratórne cvičenie, seminár a pod.), typy termínov skúšok (riadny termín, opravný termín a pod.) a ďalšie typy aktivít bez viazanosti na akýkoľvek predmet (konferencia, porada, školenie a pod.).

**Definícia aktivity** vzniká spojením typu aktivity a konkrétneho predmetu. Napr. pre predmet Fyzika by sme mohli vytvoriť tri definície aktivity: prednáška, výpočtové cvičenie, laboratórne cvičenie. Každá definícia aktivity môže mať svoje vlastné požiadavky na počty a skupiny študentov, ktoré sa môžu zúčastňovať výuky, preferované miestnosti, časové požiadavky na umiestnenie rozvrhových akcií a pod.

**Požiadavky na rozvrhový systém** definujú správanie a možnosti nového softvéru pre tvorbu rozvrhu.

**Požiadavky na rozvrh** sú vyjadrením konkrétnych očakávaní objektov vstupujúcich do tvorby rozvrhu (najmä vyučujúcich a študentov, ale aj miestností a predmetov) na výsledný rozvrh – rozloženie rozvrhových akcií.

Typ požiadavky na rozvrh (ďalej len **typ požiadavky**) je „šablónou“ pre vytvorenie novej požiadavky na rozvrh. Každý typ požiadavky môže obsahovať iný typ a počet parametrov. Zjednodušene povedané, „vyplnením“ jednotlivých parametrov typu požiadavky vznikne konkrétna požiadavka na rozvrh.

**Modul typu požiadavky** je zásuvný modul (súčasť) rozvrhového systému, ktorý implementuje uchovávanie a vyhodnocovanie typu požiadavky. Pre každý typ požiadavky existuje samostatný modul typu požiadavky, ktorý komunikuje so systémom cez dohodnuté rozhranie.

## 1.2 Nástroje na tvorbu vysokoškolského rozvrhu

### 1.2.1 Roger

*Poznámka: Informácie pre vypracovanie tejto kapitoly sme čerpali z oficiálnej stránky programu Roger [19] a jeho verzie 1.54.13004.3, ktorú sme mali možnosť odskúšať.*

Roger – „ROzumný GEnerátor Rozvrhu“ – je produktom českej softvérovej spoločnosti IS4U, s.r.o, ktorej produktom je aj univerzitný informačný systém (ďalej len UIS) na riadenie študijného a vedecko-výskumného procesu pre viacero slovenských a českých vysokých škôl a univerzít – medzi nimi aj STU v Bratislave. Roger nie je novým modulom pre tento UIS, ale samostatnou desktopovou aplikáciou pre platformu Windows. Má priamo zabudovanú podporu importu/exportu dát z/do kompatibilného UIS, ale je možné použiť ho aj samostatne. Licenciou pre používanie softvéru Roger disponuje od roku 2013 aj STU v Bratislave.

Práca so systémom Roger pozostáva z nasledujúcich krokov:

1. nastavenie základných parametrov aplikácie Roger,

2. nastavenie parametrov UIS,
3. stiahnutie vstupných dát z UIS do aplikácie Roger,
4. korekcia dát v aplikácii Roger (ak je korekcia potrebná),
5. vygenerovanie rozvrhu a prípadná korekcia (ak je potreba),
6. vloženie hotového rozvrhu do UIS,
7. korekcia rozvrhu na strane UIS (ak je potreba).

### 1.2.1.1 Vstupné dáta

Ak sa Roger používa samostatne (bez UIS), je možné vstupné dáta importovať (a exportovať) v XML – vo formáte presne špecifikovanom v dokumentácii aplikácie.

Vstupnými dátami pre generátor rozvrhu sú:

- časový rámec (globálne pre celý rozvrh),
- fakulty,
- miesta štúdia (mestá),
- vybavenie miestností (zoznam vyučovacích pomôcok pre každú miestnosť),
- areály a miestnosti (zoznam areálov všetkých fakúlt univerzity a ich miestností),
- využitie miestností fakultami (umožňuje v určenom čase počas týždňa povoliť výučbu v miestnosti inej fakulte),
- typy lekcií (napr. prednáška, cvičenie, špeciálne laboratórne cvičenie, seminár; každému typu lekcie je možné priradiť farbu, koeficient kapacity<sup>1</sup> a prioritu<sup>2</sup>),
- učitelia (pre každého učiteľa je možné upraviť jeho osobný časový rámec),
- kolízie predmetov (pomocou tzv. kolíznej matice),
- druhy študijných skupín (rozdelenie skupín na jednotlivé druhy umožňuje, aby jeden študent mohol byť súčasne členom viacerých skupín a pre tieto rozdielne skupiny mohli byť zvážené ich vzájomné kolízie),
- študijné programy (hierarchický zoznam študijných programov; ku každému študijnému programu je možné definovať študijné skupiny),

---

1 Koeficient kapacity miestnosti určuje, koľko percent študentov musí miestnosť obsiahnuť, aby bola použiteľná pre výučbu tohto typu lekcie.

2 Priorita typu lekcie určuje poradie umiestňovania typov lekcií v rozvrhu - čím nižšia hodnota, tým vyššia priorita; pri zostavovaní rozvrhu sa generátor snaží umiestniť typ lekcie s vyššou prioritou pred lekciami s nižšou prioritou.

- študijné skupiny (rozdelenie vrámci študijných programov – niekedy označované aj ako krúžky),
- kolízie študijných skupín (pomocou tzv. kolíznej matice),
- predmety.

### 1.2.1.2 Požiadavky predmetov

Evidencia predmetov a ich požiadaviek je previazaná s dátami miestností, učiteľov, miest štúdia a pod., preto je potrebné vyplniť/importovať najprv ich údaje. Ku každému predmetu je potom možné definovať okrem základných vlastností (kód, názov, garant, fakulta, atď.) aj požiadavky na miesta štúdia (počty študentov pre jednotlivé miesta štúdia – jeden predmet môže byť vyučovaný aj vo viacerých mestách), typy a definície lekcií.

Ku každému predmetu je možné vytvoriť ľubovoľný počet typov lekcií (napr. prednáška, výpočtové cvičenie, laboratórne cvičenie a nepovinný seminár) a pre každý vytvorený typ zdefinovať osobitné požiadavky:

- maximálny počet výskytu typu lekcie za deň,
- či sa môžu v rozvrhu vyskytovať paralelne lekcie daného typu,
- požadované vybavenie,
- učitelia alebo skupiny učiteľov, ktoré môžu daný typ lekcie vyučovať (s možnosťou tzv. „anonymný učiteľ“, ktorý nie je známy v čase tvorby rozvrhu, no je možné preňho zdefinovať požiadavky),
- miestnosti alebo skupiny miestností, ktoré sú odporúčané alebo vyžadované (generátor sa snaží vynútiť ich použitie) pre výučbu, prípadne spojiť viacero miestností do jednej (súbežná výučba).

Na to, aby generátor mohol s daným predmetom pracovať, vytvorenie typov lekcií nestačí. Lekcie treba ešte zdefinovať – vytvoriť nové tzv. „definície lekcií“. Pre každú definíciu lekcie je možné zvoliť:

- typ lekcie (niektorý z predtým vytvorených typov),
- periodicitu (každý/párny/nepárny týždeň),
- počet,
- kapacitu,

- dĺžku,
- obmedzenia (študijný program, ročník, študijná skupina),
- spojenie výuky lekcie s iným predmetom.

### 1.2.1.3 Generátor

Po vyplnení vstupných dát Roger dokáže automaticky vytvoriť rozvrh (s využitím genetických algoritmov). Pokročilému používateľovi poskytuje možnosť zmeniť prednastavené hodnoty generátora:

- beh algoritmu (počas behu ukladať čiastočné verzie rozvrhu, použiť silný generátor náhodných čísel, nastaviť počet vlákien),
- parametre účelovej funkcie (pre nasledovné optimalizačné kritériá je možné nastaviť veľkosť pokuty za porušenie:
  - učiteľ vyučuje viac lekcií naraz,
  - viac lekcií v jednej miestnosti,
  - lekcia koliduje s časovými požiadavkami učiteľa,
  - učiteľ učí viac než zadané maximum cvičení,
  - rozvrh z pohľadu učiteľov je rozložený na priveľký počet dní a medzi prvou a poslednou udalosťou v dni je priveľký časový priestor,
  - maximálny odporúčaný počet hodín za deň pre jedného učiteľa,
  - učiteľ učí viac hodín za deň ako maximálny odporúčaný počet,
  - lekcia nie je v preferovanej miestnosti,
  - lekcia nezodpovedá časovému rámcu,
  - viac než maximum cvičení za deň,
  - lekcie sú vyučované paralelne, ale nemali by byť,
  - lekcia je v miestnosti s nedostatočnou kapacitou,
  - lekcia je v miestnosti inej fakulty alebo na nesprávnom mieste štúdia,
  - rozvrhové udalosti sú zaradené do miestností, ktoré neobsahujú požadované vybavenie,
  - počet použitých miestností je veľký,
  - študent má pravdepodobne viac lekcií súčasne,
  - nie je dostatok času na presun medzi areálmi,

- počiatočná miera nastavenia priorít generátora,
- použitie medzipamätí genotypov),
- nastavenie evolučného algoritmu (
  - populácia – veľkosť, počet elitných jedincov v každej generácii, pravdepodobnosť mutácie a rekombinácie, počet generácií, veľkosť generácie vytvorenej pomocou evolučných operátorov,
  - pravdepodobnosť individuálnych mutácií – vznik novej lekcie, zmena času/miestnosti/učiteľa lekcie, posunutie času lekcie o časovú jednotku vpred alebo vzad, výmena časov/miestností/učiteľov v skupinke lekcii,
  - výber – turnajový, náhodný s pravdepodobnosťou priamo úmernou kvalite jedinca).

#### 1.2.1.4 „Cloud“

Roger umožňuje umiestniť vytvorený projekt s rozvrhom na centrálné úložisko – *Cloud*, odkiaľ si ho môže stiahnuť a ďalej na ňom pracovať aj iný oprávnený používateľ – a to z ľubovoľného počítača, v ktorom je nainštalovaný Roger. Avšak v tom istom čase môže na projekte pracovať len jedna osoba – projekt sa po stiahnutí uzamkne, až kým nie je na *Cloud* opäť nahraná jeho aktualizovaná verzia.

#### 1.2.1.5 Pozitíva

- Možnosť ukladať dáta a vygenerovaný rozvrh do projektov – čo umožňuje pomocou jednej aplikácie pracovať na vývoji viacerých nezávislých rozvrhov súčasne.
- Kompatibilita a priame prepojenie s UIS.
- Možnosť zdieľať zvolené učebné aj medzi fakultami navzájom.
- Ak v čase tvorby rozvrhu nie je známy vyučujúci niektorého predmetu, je možné priradiť k lekcii tzv. „anonymného učiteľa“, s ktorým sa ďalej pracuje ako s regulárnym učiteľom (napr. dajú sa preňho nastaviť požiadavky formou časového rámca).
- Pokročilý používateľ vie nastaviť veľkú časť parametrov genetického algoritmu.

- Funkcia pozastavenia generátora, možnosť vykonania manuálnych zmien a fixácie vybraných lekcií, termínov a miestností.

#### **1.2.1.6 Negatíva**

- Obmedzenie používania len na platformu Windows.
- Desktopová aplikácia – spomaľuje výpočet, import aj export dát (nutnosť sťahovať všetky dáta na lokálny PC a po vytvorení rozvrhu zas všetko nahrávať na server) a obmedzuje používanie len na PC, kde je Roger nainštalovaný.
- Široká škála požiadaviek je dostupná len pre predmety. Vyučujúcemu je možné nastaviť len časový rámec. Pre študenta ani študijnú skupinu nie je možné vytvoriť žiadne požiadavky.
- Kolízu maticu je nutné vyplňať ručne. Roger importuje z UIS iba množinu vyučovaných predmetov, no chýba prepojenie na študentov, ktorí majú jednotlivé predmety zapísané.
- Poradie výučby v týždni je možné nastaviť len všeobecne pre typ lekcie, ale nie pre typ lekcie konkrétneho predmetu. Keď napr. budeme požadovať umiestnenie prednášky pred cvičením, bude toto obmedzenie aplikované na všetky predmety s rovnakou váhou – aj tie, kde to nie je nevyhnutné. Pri generovaní rozvrhu bude s veľkou pravdepodobnosťou musieť byť toto obmedzenie u niektorých (náhodných) predmetov porušené.
- Priority jednotlivých typov požiadaviek sa dajú nastaviť len globálne pre celý rozvrh, nie pre každú inštanciu požiadavky zvlášť.
- Neintuitívne používateľské rozhranie a pomenovania. Pred reálnym používaním je nutné poriadne si prečítať manuálom k programu.

#### **1.2.2 FET**

*Poznámka: Informácie pre vypracovanie tejto kapitoly sme čerpali z oficiálnej stránky programu FET [9] a jeho verzie 5.19.0, ktorú sme mali možnosť odskúšať.*

FET – „Free Timetabling Software“ – je multiplatformový slobodný softvér na tvorbu rozvrhu pre základné, stredné aj vysoké školy. V roku 2002 ho začal vyvíjať Liviu Lalescu



vrámci svojej diplomovej práce a neskôr ho v spolupráci s ďalšími kolegami naďalej vylepšoval.

### **1.2.2.1 Vstupné dáta**

Vstupom pre algoritmus sú množiny vyučujúcich, miestností, predmetov a ich lekcí, študijných skupín a ich požiadaviek. Je možné importovať ich do programu vo formáte XML alebo postupne vytvoriť pomocou grafického používateľského rozhrania aplikácie (ďalej len GUI).

Pre učiteľa (jednotlivo) alebo (hromadne) pre všetkých učiteľov je možné vytvoriť nasledujúce typy požiadaviek:

- množina nevyhovujúcich (nedostupných) časových intervalov,
- max./min. počet dní za týždeň,
- max. počet medzier za deň/týždeň,
- max. počet hodín za deň/za sebou,
- min. počet hodín za deň,
- časový interval, v ktorom je učiteľ ochotný učiť max. zadaný počet dní v týždni,
- preferované učebne,
- max. počet presunov medzi budovami za deň/týždeň,
- min. počet voľných hodín na presun medzi budovami.

Pre konkrétnu skupinu študentov alebo (hromadne) pre všetky skupiny študentov je možné vytvoriť nasledujúce typy požiadaviek:

- množina nevyhovujúcich (nedostupných) časových intervalov,
- začínať prvou hodinou (max. počet dní, kedy môže byť toto pravidlo porušené),
- max. počet medzier za deň/týždeň,
- max. počet hodín za deň/za sebou (voliteľne s označením konkrétnej aktivity),
- min. počet hodín za deň,
- časový interval, v ktorom sa môžu študenti učiť max. zadaný počet dní v týždni,
- preferované učebne,
- max. počet presunov medzi budovami za deň/týždeň,

- min. počet voľných hodín na presun medzi budovami.

Pre lekcie z jednotlivých predmetov je možné vytvoriť nasledujúce typy požiadaviek:

- množina preferovaných miestností,
- množina preferovaných časových intervalov,
- min./max. počet dní medzi lekciami vybraného typu,
- lekcia na konci študentského dňa,
- rovnaký začiatočný čas (deň+hodina)/hodina/deň,
- za sebou, usporiadané, zoskupené - pre 2 alebo 3 lekcie,
- množina lekcí bez prekryvania sa,
- max počet paralelných lekcí vo vybraných časových úsekoch,
- min. počet voľných hodín medzi množinami lekcí,
- obsadiť max. počet rôznych učební (pre množinu lekcí),
- učiteľ/množina učiteľov,
- množina študentov/viac množín študentov.

Pre miestnosť je možné zdefinovať množinu nevyhovujúcich (nedostupných) časových intervalov.

### 1.2.2.2 Generátor

V roku 2007 bol do aplikácie implementovaný úplne nový heuristický algoritmus tzv. „*recursive swapping*” – údajne kvôli nízkej výkonnosti pôvodného genetického algoritmu, ktorý dokázal riešiť iba jednoduché rozvrhy s relatívne malým počtom obmedzení. Vygenerovaný rozvrh je možné exportovať do HTML, XML alebo CSV formátu.

### 1.2.2.3 Pozitíva

- Multiplatformový slobodný softvér.
- Široká škála požiadaviek pre vyučujúcich, študijné skupiny, predmety aj miestnosti.
- Nastavenie priority pre konkrétnu inštanciu typu požiadavky.

#### 1.2.2.4 Negatíva

- Desktopová aplikácia.
- Chýba podpora manuálnej úpravy rozvrhu.

#### 1.2.3 Wise Timetable

*Poznámka: Informácie pre vypracovanie tejto kapitoly sme čerpali z oficiálnej stránky programu Wise Timetable [21].*

Wise Technologies Ltd. z Bosny a Hercegoviny vyvinuli profesionálny softvér na tvorbu rozvrhov. Je dostupný v dvoch variantoch – Wise Timetable (pre univerzity) a Wise Timetable Basic (pre základné a stredné školy). Ide o desktopovú aplikáciu pre platformu Windows, ku ktorej je možné využiť špeciálny balík pre zobrazenie vytvorených rozvrhov na webe.

Funkcionalitu aplikácie Wise Timetable (Wise Timetable Basic) je možné rozšíriť použitím modulov

- *Room reservation* – online modul pre rezerváciu miestností,
- *Exams reservation* – online modul pre rezerváciu skúškových termínov – miestností, predmetov a skupín,
- *Preselecting courses* – webový modul, ktorý umožní študentom vytvoriť si vlastný rozvrh z vybraných predmetov,
- *Mobile browser* – aplikácia pre zobrazovanie online rozvrhov do mobilu,
- *Wise timetable translator* – umožňuje preklad do iného jazyka alebo terminológie používanej na konkrétnej škole.

##### 1.2.3.1 Vstupné dáta

Vstupné dáta je možné vkladať ručne pomocou grafického rozhrania aplikácie, importovať zo súboru XML, CSV, prípadne priamo prepojiť aplikáciu s databázou. Okrem zoznamov učiteľov, miestností a ich pomôcok, predmetov, ich typov lekcií (prednáška, cvičenie) a študijných skupín (hierarchicky), umiestnení (mestá a budovy a vzdialenosti medzi nimi) a nastavenia semestra (od – do, termíny sviatkov a voľných dní) je možné definovať nasledujúce typy požiadaviek:

- využívať miestnosť výhradne pre výučbu predmetu, ku ktorému sa vyžaduje niektorá zo zoznamu pomôcok miestnosti,
- týždne výučby pre predmet (ľubovoľne, nie len párne a nepárne),
- skupina učiteľov pre predmet,
- požadované pomôcky pre predmet,
- manuálne priradenie miestnosti, vyučujúcich, skupín a ich uzamknutie (generátor ich nebude premiestňovať),
- povoliť výučbu jedného učiteľa v 2 miestnostiach naraz (napr. pre účely video-konferencie),
- v zvolenom čase (týždne od – do, dni od – do, hodiny od – do) rezervovať miestnosť, učiteľa, študijnú skupinu alebo ich kombinácie, pričom si možno vybrať z viacerých druhov rezervácií (a ku každému napísať textový komentár): blokovanie, mimoškolská aktivita, voliteľný predmet, skúška, prednostný termín.

### 1.2.3.2 Generátor

Aplikácia Wise Timetable (aj verzia Basic) obsahuje aj okrem možnosti manuálnej tvorby rozvrhu aj generátor. Informácie o použitom type algoritmu sme žiaľ v dostupných materiáloch nenašli. V grafickom rozhraní aplikácie je možné nastaviť nasledovné parametre generovania rozvrhu:

- počet iterácií,
- počet iterácií pre učiteľov,
- počet iterácií pre skupiny,
- generovanie rozvrhu pre soboty a nedele,
- umožniť voľné dni pre vyučujúcich,
- povinná prvá hodina,
- maximálny počet hodín za deň pre učiteľa/študentov,
- najmenej 2 hodiny denne pre vyučujúceho,
- najmenej 5 hodín denne pre študentov,
- povoliť zoskupenie rovnakých kurzov v jednom dni,
- maximálny počet presunov za deň pre učiteľa/študentov.

### 1.2.3.3 Pozitíva

- Veľmi prehľadné a intuitívne grafické rozhranie s rôznymi typmi zobrazení – nezaradené predmety, konflikty, obsadene miestností, rozvrh konkrétnej miestnosti, predmetu, vyučujúceho, študijnej skupiny.
- Priame nastavenie sviatkov a voľných dní a ich zohľadnenie pri generovaní a zobrazení rozvrhu.
- Definovanie vzdialenosti medzi mestami aj budovami.
- Možnosť ad-hoc zmeny rozvrhu aj pre jeden týždeň osobitne.
- Dostupnosť modulov pre rozšírenie funkcionality aplikácie – napr. rezervácia miestností, tvorba rozvrhu skúšok.

### 1.2.3.4 Negatíva

- Obmedzenie používania len na platformu Windows.
- Desktopová aplikácia.
- Požiadavkám nie je možné nastaviť prioritu.

### 1.2.4 Zhrnutie

Po vykonaní podrobnej analýzy všetkých troch dostupných nástrojov na tvorbu vysokoškolského rozvrhu sme dospeli k záveru, že nie je možné jednoznačne určiť víťaza. Hoci niektoré typy požiadaviek ponúkajú rovnaké, v iných, nemenej dôležitých, sa navzájom dopĺňajú. No ani kombinácia všetkých pozitívnych funkcií uvedených aplikácií by nepostačovala na vytvorenie rozvrhového systému, ktorý by dostatočne uspokojil potreby rozvrhárov.

Nasledujúca tabuľka obsahuje prehľad kľúčových vlastností analyzovaných rozvrhových nástrojov.

	Roger	FET	Wise Timetable
platforma	Windows	Windows, Linux, MAC	Windows
licencia	komerčný softvér, IS4U	slobodný softvér, GNU GPL v2	komerčný softvér, Wise Technologies Ltd.
cieľová skupina	vysoké školy	základné, stredné a vysoké školy	základné, stredné a vysoké školy
spôsob tvorby rozvrhu	manuálny, automatický, kombinovaný	automatický	manuálny, automatický, kombinovaný
typy rozvrhu	rozvrh na semester	rozvrh na semester	rozvrh na semester, rozvrh skúšok
generátor	genetický algoritmus	„recursive swapping“	?
požiadavky učiteľov	áno	áno	áno
požiadavky študentov	nie	iba pre celú študijnú skupinu	iba pre celú študijnú skupinu
požiadavky miestností	áno	áno	áno
priority požiadaviek	iba niektoré typy požiadaviek (pre jednotlivé inštalácie)	áno	nie
import	AIS, XML	XML	XML, CSV, databáza
export	AIS, XML	XML, HTML, CSV	CSV, HTML, iCalendar, PDF, Email

Tab. 1: Porovnanie vlastností rozvrhových systémov Roger, FET a Wise Timetable

Významným defektom mnohých rozvrhových systémov je sústredenie náročného výpočtu na osobný počítač (väčšinou výlučne pod OS Windows). Možno aj to je jeden z dôvodov, prečo aplikácie neposkytujú dostatočnú variabilitu požiadaviek – výpočet takého rozvrhu by už v prijateľnom čase nebol zvládnuteľný na osobnom počítači.

Ďalším výrazným hendikepom väčšiny aplikácií tohto typu (aj troch vyššie uvedených) je chýbajúce rozhranie, ktoré by umožnilo formulovať požiadavky na rozvrh priamo jednotlivým učiteľom (prípadne aj študentom). Požiadavky na rozvrh môže priamo do systému vložiť iba rozvrhár. Takýto postup práce rozvrhára zbytočne zaťažuje a zároveň výrazne znižuje počet všetkých požiadaviek, ktoré sa do systému dostanú – čo v konečnom dôsledku negatívne vplýva na kvalitu (predovšetkým vygenerovaného) rozvrhu.

### **1.3 Tvorba rozvrhu na FEI**

Na FEI STU existujú dva oddelené procesy tvorby rozvrhu – tvorba rozvrhu na semester a tvorba skúškového rozvrhu.

Rozvrhom na semester sa zaoberá Mgr. Dávid Pancza, PhD. Využíva starý program WinRozvrhy, pretože súčasná verzia programu Roger (viď kapitola 1.2.1) je pre potreby rozvrhu FEI nedostatočná. Výsledok automatického generovania rozvrhu je nepoužiteľný a podpora manuálnej úpravy je slabá. Po vytvorení termínov rozvrhových akcií nasleduje druhá fáza tvorby rozvrhu na semester – zapisovanie sa študentov na cvičenia (podrobnejšie viď [7]).

Rozvrhom skúšok sa zaoberá Ing. Ondrej Gallo (vedúci tejto práce). Na jeho tvorbu využíva MS Excel s pripravenými skriptami na import dát a čiastočnú kontrolu kolízií.

Obaja rozvrhári pred začiatkom tvorby rozvrhu dostanú z AIS aktuálne zoznamy miestností, predmetov a ich učiteľov a študentov. Postupne (pred začatím tvorby rozvrhu, ale aj počas nej) prichádzajú požiadavky na rozvrh od vyučujúcich – formou e-mailu alebo na papieri. Niektoré typy požiadaviek je možné (ručne) zadať do rozvrhového systému (napr. hodinové dotácie prednášok a cvičení), ale na mnohé z nich musí rozvrhár len „myslieť“ počas tvorby rozvrhu (napr. časové požiadavky, preferované miestnosti, atď.).

Vzhľadom na uvedené skutočnosti, je potreba nového kvalitného rozvrhového systému pre FEI je určite aktuálna.





## 2 CIEĽ PRÁCE

V kapitole 1.2 boli podrobne roanalyzované tri profesionálne programové nástroje na tvorbu školských rozvrhov. Cieľom tejto práce je vytvoriť dôkladný návrh a implementovať jadro systému, ktorý bude môcť neskôr konkurovať existujúcim programom podobného zamerania.

Nový rozvrhový systém sme sa rozhodli implementovať formou webovej aplikácie – čo je hlavným pozitívnym rozdielom oproti spomínaným existujúcim nástrojom. Takýto typ aplikácie umožňuje transfer výkonovo náročných operácií na špecializovaný hardvér (napr. školský server), prístup rozvrhára a ďalších osôb z ľubovoľného počítača s internetovým pripojením a kolaboráciu viacerých osôb pri tvorbe rozvrhu. Významným urýchlením a odľahčením rutinej práce rozvrhára bude možnosť zadávať do systému požiadavky na rozvrh priamo vyučujúcimi (prípadne aj študentami).

Modularita návrhu architektúry nového systému bude umožňovať postupnú implementáciu jednotlivých súčastí systému, rozširovanie ich funkcionality a prispôsobiteľnosť modulov konkrétnym požiadavkám rôznych škôl.

V implementačnej časti práce sa zameriame predovšetkým na tie časti, ktoré sú nevyhnutné pre manuálnu tvorbu rozvrhu jedným rozvrhárom na FEI STU (jadro navrhnutého systému):

- import a úprava dát z AIS,
- tvorba rozvrhu na semester aj skúškové obdobie,
- vyhodnocovanie kolízií rozvrhových akcií,
- rôzne typy prehľadného zobrazenia (napr. deň, týždeň, mesiac) aktuálnej verzie rozvrhu s možnosťou filtrovania rozvrhových akcií (napr. podľa miestnosti, učiteľa, atď.).



## 3 ŠPECIFIKÁCIA POŽIADAVIEK

Pri tvorbe takého komplexného informačného systému akým je rozvrhový systém, je dôležité okrem analýzy existujúcich riešení aj podrobne zmapovať očakávania všetkých typov zainteresovaných osôb – predovšetkým rozvrhárov, ale aj vyučujúcich a študentov. Preto sme tejto časti práce venovali náležitú pozornosť.

Vychádzali sme z poznatkov nadobudnutých počas tímového projektu [8], nových podnetov získaných počas analýzy existujúcich rozvrhových systémov (viď kapitola 1.2) a stretnutí s rozvrhármi FEI STU Mgr. Dávidom Panczom, PhD. (rozvrh na semester), Ing. Ladislavom Körösim, PhD. (bývalý rozvrhár pre rozvrh na skúškové obdobie) a Ing. Ondrejom Gallom (súčasný rozvrhár pre rozvrh na skúškové obdobie).

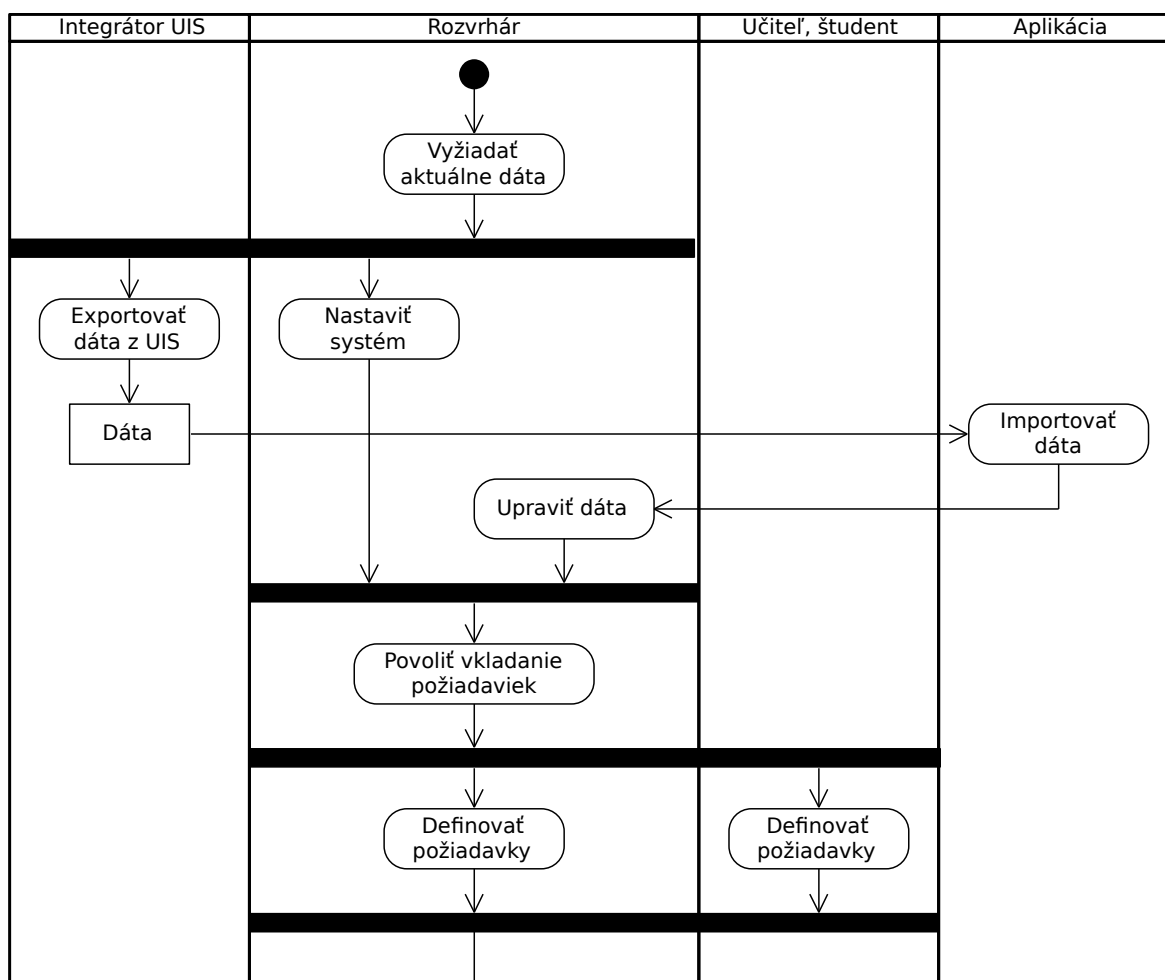
Prvá časť tejto kapitoly sa zaoberá analýzou procesov vyskytujúcich sa počas tvorby vysokoškolského rozvrhu. Druhá časť obsahuje stručný prehľad prípadov použitia nového rozvrhového systému. Posledná kapitola je venovaná dôkladnému rozboru požiadaviek na rozvrh pre všetky typy zainteresovaných subjektov (učitelia, študenti, predmety, atď.).

### 3.1 Postup tvorby rozvrhu

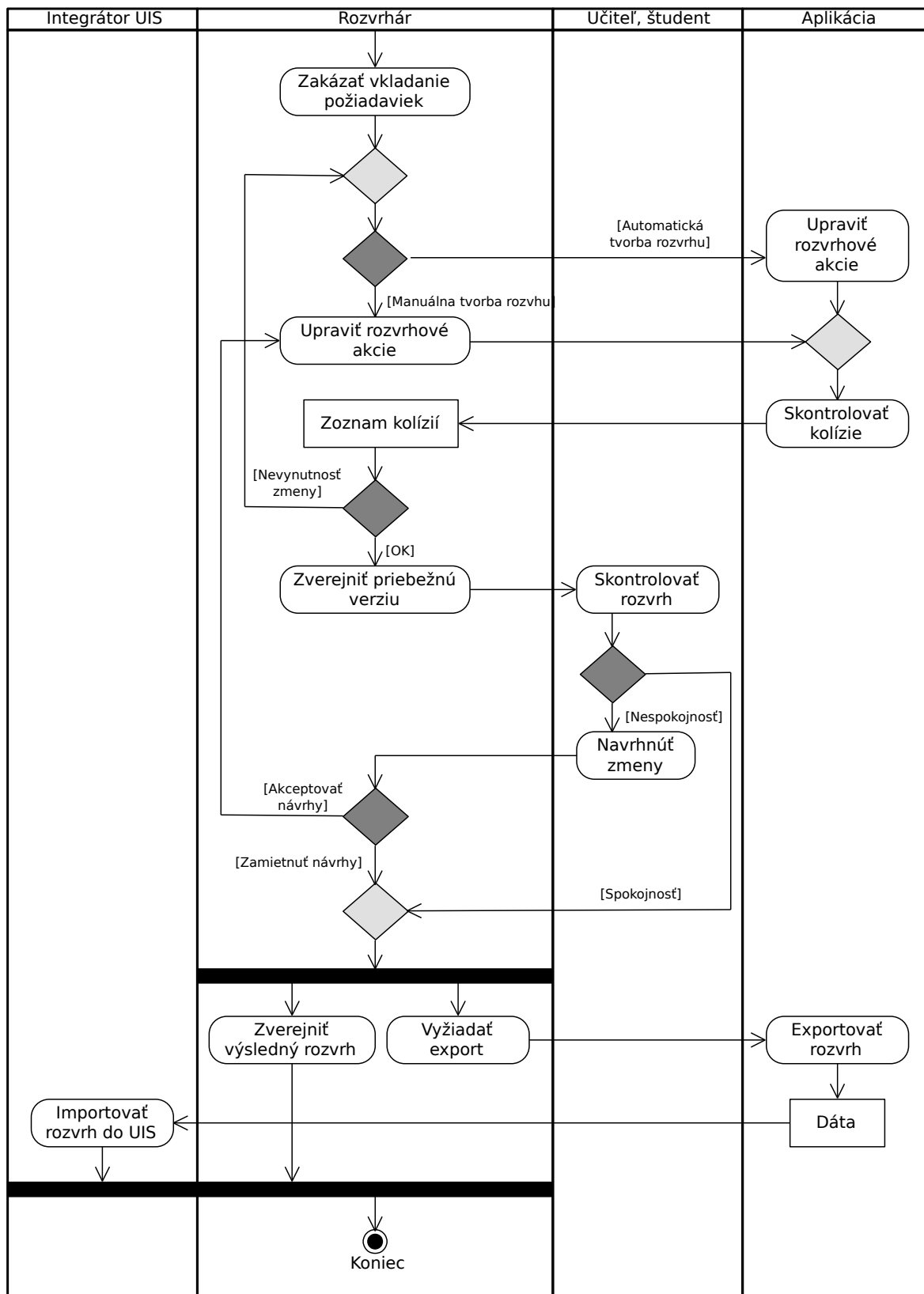
Kvalitný rozvrhový systém by mal čo najviac automatizovať rutinnú prácu rozvrhára a pokrývať všetky kroky tvorby rozvrhu – od začiatku po koniec. Táto kapitola špecifikuje poradie aktivít tvorby rozvrhu (na semester aj skúškové obdobie), ktoré by malo byť obsiahnuté v novom rozvrhovom systéme. Postup tvorby rozvrhu schematicky znázorňuje UML diagram aktivít rozdelený do obrázkov 1 a 2 (obrázok č. 2 je priamym pokračovaním obrázka č. 1).

Na začiatku procesu tvorby rozvrhu pre dané obdobie rozvrhár požiada integrátora UIS o export aktuálnych zoznamov predmetov, ich učiteľov a študentov, miestností a ich vybavenia. Kým integrátor pripraví potrebné dáta, rozvrhár nastaví parametre rozvrhového systému pre dané obdobie (napr. začiatok a koniec semestra a skúškového obdobia, dni

voľna a náhradu výučby, atď.). Po importe dát do rozvrhového systému môže rozvrhár doplniť údaje, ktoré neboli dostupné v UIS, ale pre tvorbu rozvrhu sú kľúčové (napr. skutoční prednášajúci a cvičiaci z jednotlivých predmetov, zoskupenie výuky viacerých predmetov, atď.). Ďalšou fázou je definícia požiadaviek na rozvrh. Okrem rozvrhára môžu v stanovenom čase vybrané typy požiadaviek vkladať do systému aj učitelia, prípadne študenti. Nasleduje cyklická fáza tvorby rozvrhových akcií, ktoré môže rozvrhár vytvárať manuálne alebo ponechať ich automatické generovanie špeciálnemu algoritmu rozvrhového systému. Po zverejnení priebežnej verzie rozvrhu, môžu učitelia aj študenti vyjadriť svoju nespokojnosť s rozvrhom a navrhnúť konkrétne zmeny, ktoré rozvrhár následne môže (ale nemusí) akceptovať. Úprava rozvrhu prebieha dovtedy, kým sa nenájde vyhovujúce umiestnenie rozvrhových akcií. Export výsledného rozvrhu je odoslaný integrátorovi UIS, ktorý zabezpečí jeho korektný import do UIS.



Obr. 1: UML diagram aktivít znázorňujúci postup tvorby rozvrhu – 1. časť



Obr. 2: UML diagram aktivít znázorňujúci postup tvorby rozvrhu – 2. časť

## 3.2 Prípady použitia

V tejto kapitole sa nachádza prehľad prípadov použitia, ktoré by mal pokrývať nový rozvrhový systém. Väčšina z nich vychádza z poznatkov získaných počas tímového projektu, preto uvádzame iba ich stručnú charakteristiku [8].

### 3.2.1 Rozvrh na semester aj skúškové obdobie

*Podpora tvorby rozvrhu hodín na semester aj rozvrhu skúšok.*

### 3.2.2 Import a úprava dát z UIS

*Podpora importu zoznamov predmetov, vyučujúcich, študentov a miestností z UIS a ich korekcia pre potreby vytvorenia rozvrhu.*

Na začiatku procesu tvorby rozvrhu administrátor požiada integrátora UIS o export aktuálnych zoznamov predmetov, vyučujúcich, študentov, miestností a ich vzájomných prepojení z UIS a následne ich importuje do rozvrhového systému. Keďže v UIS sa často nachádzajú nedostatočné informácie, musí systém dovoliť korekciu importovaných dát:

- zlúčenie predmetov, ktoré sú v UIS evidované pod viacerými identifikátormi, ale ich výuka prebieha spolu (akoby to bol jeden predmet),
- zmazanie predmetov, ktoré z nejakého dôvodu nechceme zaradiť do rozvrhu (napr. telesná výchova, anglický jazyk, bakalársky, diplomový a tímový projekt, atď.),
- pridanie/úprava/zmazanie vyučujúcich, študentov, predmetov, miestností a ich vybavenia,
- pridanie/zmazanie študentov zapísaných na predmet a nastavenie stupňa príslušnosti študenta k predmetu (napr. môžeme nastaviť nižší stupeň príslušnosti študenta k opakovanému predmetu, ak sme ochotní akceptovať kolíziu s iným jeho predmetom),
- pridanie/zmazanie vyučujúcich a ich úloh (garant, prednášajúci, skúšajúci, cvičiaci, administrátor, tútor) pre jednotlivé predmety.

### 3.2.3 Konfigurácia systému

Poskytnutie rozhrania pre nastavenie nasledovných parametrov:

- začiatok a koniec semestra,
- začiatok a koniec skúškového obdobia,
- počet týždňov semestra,
- voľné dni a náhrada výučby,
- vytvorenie časových mriežok pre zobrazenie rozvrhu hodín (napr. iná mriežka pre rozvrh na semester – 1. hodina, 2. hodina, atď – a iná pre rozvrh skúškového obdobia – 1. beh, 2. beh, atď.),
- parametre generátora rozvrhových akcií (bude vysvetlené v kapitole 3.2.5),
- zapnúť/vypnúť automatickú kontrolu kolízií pri vytvorení/premiestnení rozvrhovej akcie (bude vysvetlené v kapitole 3.2.8),
- farby (pre zobrazenie rozvrhu):
  - všeobecne pre prednášky, cvičenia a skúšky,
  - pre skupinu predmetov zastrešovaných konkrétnym pracoviskom,
  - pre konkrétny predmet (t.j. spoločne pre jeho prednášky, cvičenia aj skúšky),
  - pre konkrétnu udalosť – napr. pre prednášku z konkrétneho predmetu alebo mimoriadnu udalosť, ktorá sa neviaže na žiaden predmet (konferencia, mimoriadna schôdza, mimoriadna prednáška alebo prezentácia).

### 3.2.4 Definovanie požiadaviek na rozvrh

*Systém bude umožňovať definovať požiadavky na rozvrh všetkým zainteresovaným objektom – osobám (vyučujúcim a študentom), celým skupinám osôb (napr. študijnému odboru, ročníku, atď.), predmetom a miestnostiam.*

Každý vyučujúci aj študent si bude môcť nastaviť svoje osobné požiadavky na rozvrh, prípadne požiadavky predmetu (garant alebo iná osoba s potrebnými oprávneniami). Požiadavky miestností a skupín osôb bude nastavovať rozvrhár. Jednotlivé typy požiadaviek bude možné implementovať aj neskôr a v systéme nastaviť, pre ktoré typy objektov môžu byť použité. Typmi požiadaviek sa budeme podrobnejšie zaoberať v kapitole 3.3.

### 3.2.5 Generovanie rozvrhu

*Podpora inteligentného umiestnenia prednášok a cvičení (prípadne skúšok) do času a priestoru.*

Použitie tejto funkcie je voliteľné a má význam, iba ak sú v rozvrhovom systéme vyplnené všetky skutočné požiadavky predmetov, vyučujúcich aj študentov. Na základe týchto požiadaviek sa po spustení generátora systém snaží nájsť optimálne rozloženie rozvrhových akcií. Typ generátora ako aj jeho parametre budú nastaviteľné v konfigurácii systému. Systém bude poskytovať rozhranie pre dodatočnú implementáciu ľubovoľného počtu algoritmov inteligentného generovania rozvrhu.

### 3.2.6 Manuálna úprava rozvrhu

*Podpora pohodlného drag&drop vytvárania a premiestňovania rozvrhových akcií.*

Túto funkciu je možné použiť ako hlavný nástroj pre tvorbu rozvrhu, alebo len na doladenie vygenerovaných rozvrhových akcií.

Rozvrhár si zvolí, ktorý typ rozvrhu chce upravovať (rozvrh na semester alebo na skúškové). Zobrazí sa mu tabuľka s rozmiestnením už vytvorených rozvrhových akcií (ak nejaké sú). Intuitívnou metódou *drag&drop* bude môcť vytvárať nové rozvrhové akcie a premiestňovať (odstraňovať) už vytvorené.

Systém by mal dovoliť zafixovanie rozvrhovej akcie, ktorá by mala ostať na svojom mieste aj po spustení generátora. Ak sa zafixovanú akciu pokúsi premiestniť rozvrhár manuálne, systém ho upozorní, že akcia je zafixovaná, ale povolí jej premiestnenie.

Lepšiu orientáciu v rozvrhu bude poskytovať možnosť vytvárania rozvrhu po častiach z rôznych uhlov pohľadu – podľa ročníkov, študijných odborov a zameraní, ústavov a katedier, typov výuky (prednáška, cvičenie) a termínov skúšky (riadny, opravný).

### 3.2.7 Zobrazenie kolíznych matíc

*Podpora prehľadného zobrazenia všetkých troch typov kolíznych matíc – študentov, vyučujúcich aj študijných skupín.*



### **3.2.8 Kontrola kolízií**

*Nájdenie a zobrazenie kolízií rozvrhových akcií.*

Rozvrhár si bude môcť zobrazit' kolízie pre konkrétnu rozvrhovú akciu, vyučujúceho, študenta, miestnosť a tiež zoznam všetkých kolízií v aktuálnej verzii rozvrhu.

Ak je zapnutá automatická kontrola kolízií, bude systém kontrolovať kolízie pri každej zmene v rozvrhu. Ak sa nájde nová kolízia, objaví sa upozornenie, ktoré bude možné schovať.

Dôležité je, aby logika systému dovolila ponechanie kolíznych udalostí v rozvrhu (môže sa stať, že rozvrhár spôsobí kolíziu úmyselne a chce ju zachovať aj vo výslednom rozvrhu).

### **3.2.9 Prezeranie výsledného rozvrhu**

*Podpora prezerania a tlače výsledného rozvrhu s ohľadom na dni voľna a náhradu výučby.*

Systém bude podporovať tri rôzne typy zobrazenia tabuľky s rozvrhom – mesiac, týždeň, deň – a filtrovanie rozvrhových akcií podľa

- miestnosti,
- predmetu,
- študijného programu a zamerania študentov,
- konkrétneho vyučujúceho alebo študenta.

Zobrazenie udalostí pre konkrétne dni bude posunuté podľa nastavených dní voľna a náhrady výučby.

### **3.2.10 Export výsledného rozvrhu do UIS**

*Hotový rozvrh bude možné exportovať do UIS.*

### **3.2.11 Rezervácia miestnosti**

*Podpora rezervácie ľubovoľnej miestnosti na výnimočnú udalosť (napr. konferencia).*

Miestnosť bude možné rezervovať už počas tvorby rozvrhu – ak je daná udalosť v tom čase známa. Ak sa potreba rezervácie miestností objaví až počas semestra/skúškového, bude možné vyhľadať voľnú miestnosť v zvolenom čase a následne rezervovať. Táto funkcia môže byť užitočná napríklad aj pre vyhradenie miestnosti na písomku alebo doučovanie.

### **3.2.12 Navrhovanie zmien v rozvrhu**

*Umožnenie zásahu do rozvrhu aj vyučujúcim a študentom.*

Ak si vyučujúci alebo študent pri prezeraní rozvrhu všimne chybu, inkonzistenciu alebo nájde lepšie umiestnenie rozvrhovej akcie, bude môcť priamo cez systém odoslať rozvrhárovi návrh na zmenu aj s komentárom. Rozvrhár bude môcť návrh

- akceptovať (zmeny v umiestnení rozvrhovej akcie sa vykonajú bez nutnosti ďalšieho zásahu rozvrhára),
- pozmeniť (zmeny v umiestnení rozvrhovej akcie urobí rozvrhár manuálne),
- ignorovať (návrh ostane v zozname návrhov a bude ho možné akceptovať neskôr),
- zamietnuť (návrh bude vymazaný zo zoznamu návrhov).

### **3.2.13 Zálohovanie rozvrhu**

*Podpora ukladania jednotlivých verzií rozvrhu, ich prezeranie a potenciálna obnova.*

Počas procesu tvorby rozvrhu bude možné kedykoľvek uložiť aktuálnu verziu rozvrhu. Systém bude poskytovať rozhranie na správu uložených verzií rozvrhov – prezeranie, mazanie a obnovu. Rozvrhár sa tak bude môcť vrátiť k predchádzajúcej verzii bez toho, aby musel všetky zmeny vracať späť manuálne. Okrem toho môže mať náhľad do starších verzií rozvrhu význam aj pri tvorbe nového rozvrhu – pre inšpiráciu, ako bol daný problém riešený v minulosti.

### **3.2.14 Priradenie oprávnení používateľom**

*Podpora vytvárania používateľských rolí a nastavovanie rôznych stupňov oprávnení vyučujúcim aj študentom.*

V systéme sa budú dať hromadne nastaviť oprávnenia pre jednotlivé typy používateľov (študent, vyučujúci) a skupiny používateľov (garanti predmetov, členovia senátu, doktorandi). Okrem toho vybraným jednotlivcom bude možné priradiť ďalšie práva (napr. pre každý ústav určiť administrátora, ktorý bude môcť upravovať požiadavky predmetov zastrešovaných daným ústavom, vyplňať nekompletné požiadavky iných zamestnancov daného ústavu, atď.).

Každý používateľ systému bude môcť na zvolenú časť svojich právomocí splnomocniť iného používateľa. Zánikom práv pôvodného používateľa, zaniknú aj práva splnomocnenca. Takto bude môcť garant predmetu prenechať prednášajúcemu svoje práva na definovanie požiadaviek predmetu.

### 3.3 Požiadavky na rozvrh

Kvalita vygenerovaného rozvrhu nezávisí len od výberu a parametrizácie algoritmu, ale aj od možností vstupov pre tento algoritmus. Napríklad ak systém neumožňuje vložiť používateľovi požiadavku na obednú prestávku, nemôžeme od vygenerovaného rozvrhu očakávať, že ju zohľadní. Nutnou podmienkou úspešného generátora rozvrhu je komplexná množina typov požiadaviek, z ktorých sa budú dať „poskladať“ všetky reálne požiadavky na výsledný rozvrh hodín.

Vkládanie požiadaviek zainteresovaných subjektov priamo do rozvrhového systému je rovnako užitočné aj pri manuálnej tvorbe rozvrhu – systém môže rozvrhára upozorňovať na porušenie požiadavky a tiež ohodnotiť kvalitu aktuálnej verzie rozvrhu vzhľadom na uspokojenie jednotlivých požiadaviek.

V tejto kapitole postupne vymenujeme typy požiadaviek všetkých zainteresovaných subjektov: používateľov (učiteľov aj študentov), skupín používateľov, miestností a ich typov, predmetov, typov aktivít a definícií aktivít. Čerpali sme z poznatkov získaných z tímového projektu [8], ktoré sme lepšie kategorizovali a doplnili o ďalšie typy požiadaviek.

#### **Používateľ, skupina:**

- časový rámec,

- max. počet presunov medzi budovami za deň/týždeň,
- min. počet voľných hodín na presun medzi dvojicami budov,
- max./min. počet dní za týždeň,
- max./min. počet medzier za deň/týždeň,
- max./min. počet hodín za sebou/za deň,
- prestávka v intervale od X do Y dĺžky Z,
- každý deň rovnaký začiatok/koniec v rovnakom čase,
- časové intervaly, v ktorých je osoba ochotná byť v škole max. zadaný počet dní v týždni,
- max. počet dní, kedy používateľ nemusí začínať prvou hodinou,
- preferované učebne,
- ostať čo najdlhšie v tej istej učebni,
- minimálny rozstup medzi dvomi skúškami,
- stráženie kolízií s nejakou osobou v čase aj priestore, prípadne len v čase,
- stráženie kolízií s inou aktivitou – napr. zasadanie senátu, výuka nezapísaného predmetu.

#### **Miestnosť, typ miestnosti:**

- časový rámec,
- využiť miestnosť, len ak je o ňu explicitne požiadané,
- využiť miestnosť, iba ak je explicitne požiadané o niektoré jej vybavenie,
- povoliť miestnosť iba pre predmety zastrešované konkrétnym pracoviskom,
- viacero miestností spojiť do jedného logického celku, prípadne jednu miestnosť rozdeliť na niekoľko častí,
- povoliť v miestnosti iba výuku konkrétnych predmetov,
- povoliť zdieľanie miestnosti s inými fakultami.

#### **Predmet:**

- časový rámec,
- množina preferovaných miestností,
- min. počet dní/hodín medzi dvojicami zadaných aktivít,

- spojiť vybrané aktivity do jedného bloku (aby nasledovali za sebou bez prestávky),
- množina aktivít bez prekryvania sa,
- spojiť výuku s iným predmetom.

#### **Typ aktivity:**

- časový rámec,
- týždne výučby,
- množina preferovaných miestností.

#### **Definícia aktivity:**

- časový rámec,
- týždne výučby,
- počet povinných účastí pre študenta za týždeň,
- počet vyučovacích hodín,
- počet študentov (možnosť *ALL* pre všetkých zapísaných) na jednu rozvrhovú akciu definície aktivity (min., max., prípadne interval/intervaly počtov),
- skupina študentov, pre ktorú budú rozvrhové akcie danej definície aktivity určené (podmnožina študentov, ktorí majú daný predmet zapísaný),
- počet (interval) paralelných rozvrhových akcií (mala by tam byť možnosť *ALL*),
- max. počet (interval) rozvrhových akcií za sebou (mala by tam byť možnosť *ALL*),
- max. počet (interval) rozvrhových akcií za deň (mala by tam byť možnosť *ALL*),
- umiestnenie rozvrhovej akcie do miestnosti – nasledovné typy požiadaviek:
  - skupiny miestností – každá skupina pozostáva z 1 alebo viacerých miestností,
    - zlúčiť kapacitu miestností (dá sa nastaviť pre každú skupinu miestností) – skupina miestností bude chápaná ako jedna veľká miestnosť (kapacita miestností v skupine bude spočítaná), inak prebieha výučba vo všetkých miestnostiach skupiny súčasne (kapacita je určená podľa najmenej z nich),
  - povoliť učebňu, ktorá je len logickým celkom (CPUa, CPUb),
  - zvoliť len typ učebne (napr. poslucháreň, laboratórium),
  - min./max. kapacita miest,

- koeficient kapacity – koľko percent študentov musí miestnosť obsiahnuť, aby bola použiteľná pre výučbu (napr. iba 80% lebo všetci tam určite nebudú chodiť),
- množina pomôcok vybavenia miestnosti,
- priradenie vyučujúcich:
  - skupiny vyučujúcich – každá skupina pozostáva z 1 alebo viacerých vyučujúcich a parametru počet (interval), ktorý určuje, koľko rozvrhových akcií definície aktivity môže daná skupina vyučujúcich vyučovať,
  - anonymný učiteľ – dajú sa mu nastaviť aj požiadavky, neskôr ho možno nahradiť naozajstným vyučujúcim,
- poradie aktivity v rozvrhu – rozvrhár (alebo generátor) sa bude snažiť umiestniť aktivitu s vyššou prioritou pred aktivitu s nižšou prioritou – vrámci toho istého predmetu (všeobecne – aj iný deň; ak je nastavené spojenie aktivít do jedného bloku, budú za sebou v ten istý deň),
- spojiť výuku s iným predmetom,
- aktivita na konci študentského dňa.

**Skúška** je špeciálnym typom aktivity z predmetu. Vzhľadom na jej jedinečnosť, uvádzame osobitne relevantné typy požiadaviek pre každý termín skúšky:

- časový rámec,
- miestnosti:
  - presne zvolená miestnosť,
  - ľubovoľná miestnosť zvolenej kapacity (napr. len „300-ky“),
  - ľubovoľná počítačová učebňa,
  - koeficient kapacity miestnosti <sup>3</sup>,
- počet paralelných rozvrhových akcií,
- počet rozvrhových akcií za sebou s rôznymi alebo rovnakými množinami študentov,
- umiestniť skúšky z 2 rôznych predmetov do rovnakej miestnosti v rovnakom čase,
- umožniť robiť bloky predmetov – napr. všetky matematiky v jeden deň,

---

3 Predvolená hodnota koeficientu kapacity miestnosti pre skúšku môže byť napr. 600% (medzi študentami sa väčšinou nachádzajú voľné miesta).

- min. počet dní na prípravu na predmet (napr. min 1 deň, štandardne 3, ideálne týždeň),
- nie vtedy, keď sú skúšky z predmetov X, Y, Z,
- zistiť obtiažnosť predmetov pre skupinu študentov a určiť ich poradie, napr. striedať ľahký a ťažký predmet (pozn. niekedy môže byť skupina predmetov rovnocenná, napr. je jedno, či najprv M alebo F),
- nastaviť skúšajúcich a sledovať im kolízie, aby neskúšali naraz viac predmetov, prípadne povoliť, ak ide o predmety v rovnakej miestnosti a vyučujúci si to vyžiada,
- rešpektovať termíny štátnic pre končiacich bakalárov a inžinierov (snažiť sa dať predtým aj opravné termíny, hlavne z ťažkých predmetov),
- skúška dohodou (rozvrhár nemusí vytvárať rozvrhovú akciu pre danú skúšku).



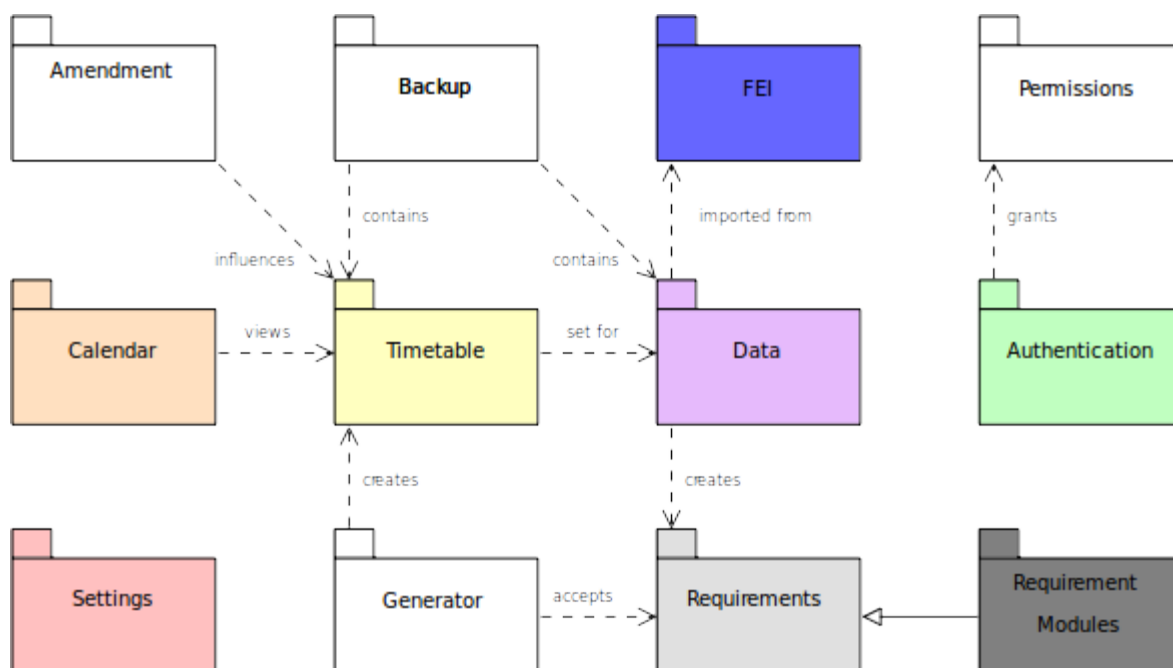


## 4 NÁVRH

Táto časť práce je venovaná návrhu komplexného rozvrhového systému, ktorého špecifikácia bola podrobne predstavená v predchádzajúcej kapitole. V prvej kapitole popíšeme rámcový návrh architektúry celého systému. V ďalších kapitolách sa sústreďíme na detailný návrh hlavných častí systému, ktoré sú predmetom implementačnej časti tejto práce.

### 4.1 Architektúra systému

Na obrázku 3 sa nachádza zjednodušená schéma celého systému znázornená pomocou UML diagramu balíčkov. Farebne sú zvýraznené balíčky tvoriace jadro systému, ktorých funkcionality sme aspoň čiastočne implementovali. Podrobnejší popis ich objektových tried a vzťahov sa nachádza v kapitolách 4.2 – 4.8. Návrh a implementácia ostatných častí aplikácie (bielou) sú mimo rozsahu tejto práce.



Obr. 3: Diagram balíčkov (zoskupenia tried) – architektúra systému

**Backup** – podpora zálohovania verzií rozvrhu. Každá záloha bude obsahovať kompletný obraz inštancií modelov z balíkov *Data* a *Timetable* (v čase vytvorenia zálohy). K takto vytvorenej zálohe sa bude môcť rozvrhár neskôr vrátiť (obnoviť), prezerať alebo zmazať.

**Calendar** – zobrazenie rozvrhových akcií v tabuľke. Umožní vytvoriť rôzne typy mriežok (intervaly hodín), medzi ktorými sa bude dať prepínať pri zobrazení rozvrhu na týždeň alebo deň. Okrem toho bude ponúkať aj mesačný prehľad rozvrhových akcií (bez mriežky).

**FEI** – import/export dát z/do UIS (v našom prípade AIS, ktorý používa FEI STU). Pre každú fakultu (univerzitu) môže existovať podobný balík implementovaný iným spôsobom – v závislosti od formátu a štruktúry dát, ktorú poskytne daná fakulta (univerzita).

**Authentication** – overenie totožnosti používateľa po zadaní prihlasovacích údajov. Na základe zistenej totožnosti systém pridelí aktívnemu používateľovi práva a umožní mu prístupovať k vybraným funkciám systému.

**Permissions** – nastavenie oprávnení používateľom.

**AmendmentProposals** – navrhovanie zmien v umiestnení rozvrhových akcií.

**Timetable** – vytváranie rozvrhových akcií pre rozvrh na semester aj skúškový rozvrh.

**Data** – úprava dát nevyhnutných pre tvorbu rozvrhu (zoznamy predmetov, študentov, miestností, atď.). Rýchly import týchto údajov (voliteľne) zabezpečí osobitný balík (v našom prípade balík *FEI*).

**Settings** – správa nastavení celého rozvrhového systému. Tento balík je využívaný, všetkými ostatnými balíkmi, avšak kvôli lepšej prehľadnosti tieto závislosti nie sú na obrázku vyznačené.

**Generator** – inteligentný algoritmus na generovanie rozvrhových akcií na základe vstupných požiadaviek (balík *Requirements*).

**Requirements** – správa požiadaviek vyučujúcich, študentov, miestností. Bude obsahovať rozhranie pre pripojenie modulov z balíka *RequirementModules*.

**RequirementModules** – balík pre implementáciu jednotlivých typov požiadaviek na rozvrh. Každý typ požiadavky bude musieť dodržať rozhranie na pripojenie k modulu *Requirements*.

## 4.2 Balíček „Data“

Väčšina tried tohto balíčka obsahuje nepovinný atribút `uis_id`, ktorý slúži na uchovanie jedinečného identifikátora, pod ktorým je objekt evidovaný v UIS. Nastaví sa pri importe dát z UIS a ďalej k nemu nebude možné z používateľského rozhrania pristupovať. Neskôr sa využije pri exporte rozvrhových akcií do UIS.

**RequireObject** – trieda, z ktorej ďalej dedia všetky triedy predstavujúce objekty, ktorým je možné priradiť nejakú požiadavku na rozvrh (`User`, `Group`, `Subject`, `ActivityDefinition`, `ActivityType`, `Room`, `RoomType`). Atribút `priority` slúži na odlíšenie dôležitosti splnenia požiadaviek jednotlivých objektov.

**User** – trieda pre uchovanie základných informácií o jednotlivých používateľoch (učiteľoch aj študentoch).

**Group** – trieda s reflexívnou asociáciou pre uchovanie a prácu s hierarchickým modelom používateľských skupín. Na najvyššej úrovni sa budú nachádzať dve skupiny – *učitelia* a *študenti* predstavujúce dva samostatné stromy. Skupina *študenti* bude koreňom stromu reprezentujúceho hierarchiu študijných programov, zameraní a ročníkov. Skupinu *učitelia* bude podľa potreby tiež možné ďalej členiť. Medzi triedami `User` a `Group` sa nachádza vzťah s násobnosťou M:N, pretože študent môže študovať súčasne viacero študijných programov, vo výnimočných prípadoch môže byť zároveň študentom aj učiteľom, atď.

**Subject** – trieda s reflexívnou asociáciou pre uchovanie a prácu s hierarchickým modelom predmetov. Hierarchia má význam pri potrebe zoskupenia predmetov, ktorých výuka bude prebiehať vždy súčasne (hoci v UIS sú evidované ako osobitné predmety) – vytvorí sa nový predmet (s prázdnu hodnotou `uis_id`), ktorý bude koreňom uzlov s predmetmi,

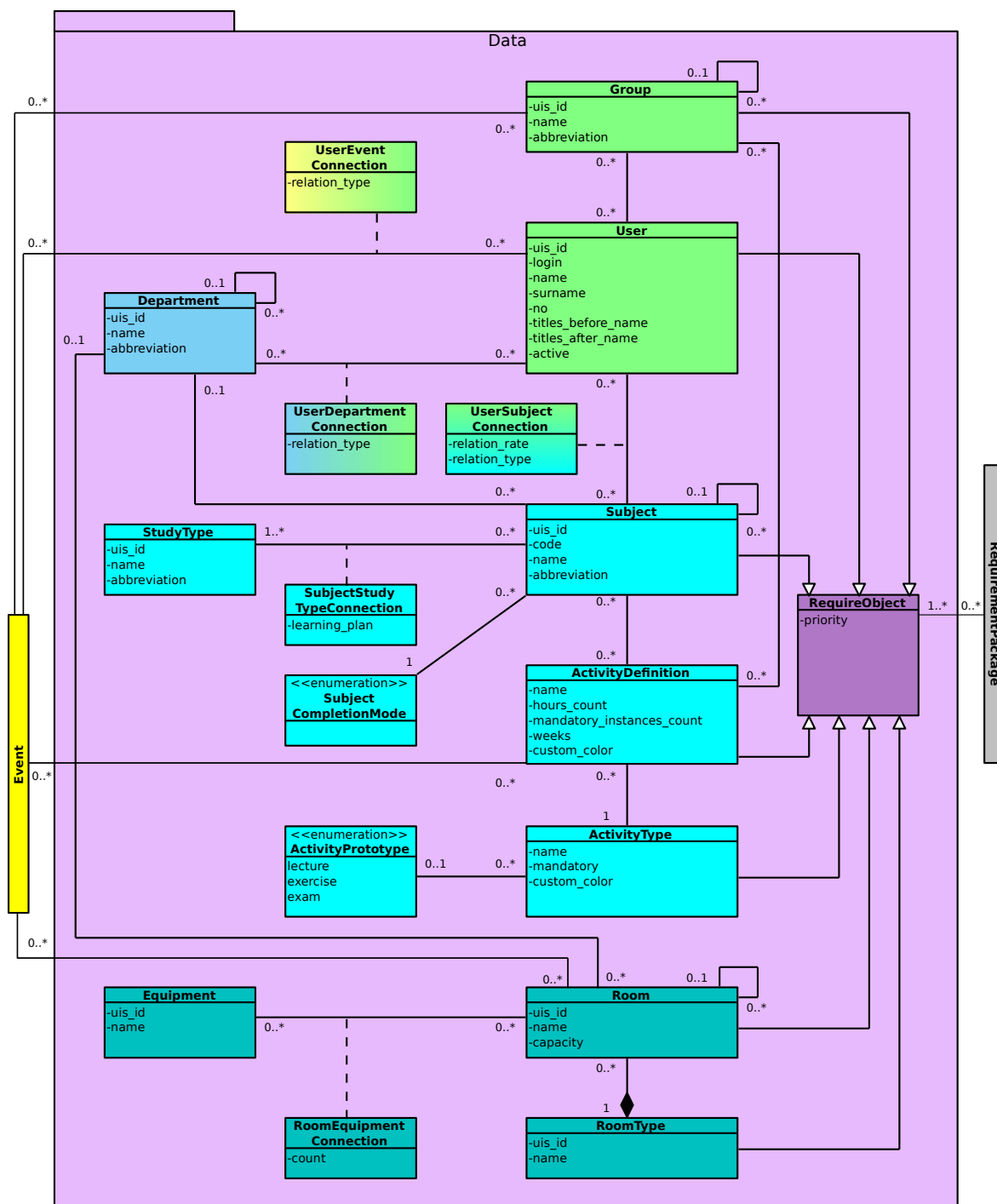
ktoré je potrebné zoskupiť. V ďalších častiach rozvrhového systému budeme považovať za predmety iba tie inštancie triedy `Subject`, ktoré sú koreňmi stromov. Typy ukončenia predmetu (enumerácia `SubjectCompletionMode`) je možné nastaviť v module *Settings*, napr. zápočet, skúška, klasifikovaný zápočet, štátna skúška. Výuka predmetu môže byť realizovaná viacerými formami – prezenčne alebo dištančne (trieda `StudyType`) a pre každú formu môže mať iný rozsah výučby (hodinová dotácia prednášok a cvičení) – atribút `learning_plan` v asociačnej triede `SubjectStudyTypeConnection`.

Triedy `User` a `Subject` sú prepojené väzbou typu M:N s asociačnou triedou `UserSubjectConnection` s atribútmi `relation_type` – celé číslo vyjadrujúce typ prepojenia (číselník typov prepojení je možné nastaviť v module *Settings*, napr. študent, garant, prednášajúci, cvičiaci, skúšajúci, administrátor, tútor) a `relation_rate` – reálne číslo vyjadrujúce stupeň príslušnosti používateľa k predmetu (viď kapitola 3.2.2).

**ActivityType** – trieda pre vytvorenie a prácu s generickými typmi aktivít, ktoré sa budú ďalej využívať v rozvrhovom systéme. Objektmi tejto triedy budú typy výuky predmetov (prednáška, cvičenie, laboratórne cvičenie, seminár a pod.), typy termínov skúšok (riadny termín, opravný termín a pod.) a ďalšie typy aktivít bez viazanosti na akýkoľvek predmet (konferencia, porada, školenie a pod.). Atribút `mandatory` hovorí o tom, či je typ aktivity povinný alebo nie. Typ aktivity môže byť (voliteľne) založený na jednom z troch prototypov – `lecture` (prednáška), `exercise` (cvičenie), `exam` (skúška).

**ActivityDefinition** – trieda, ktorej objekt je spojením konkrétneho typu aktivity a (voliteľne) predmetov. Obsahuje atribúty pre nastavenie týždňov výuky (`weeks`), farby (`custom_color`), počtu hodín trvania rozvrhovej akcie (`hours_count`) a počtu povinných rozvrhových akcií (`mandatory_instances_count`), ktorých sa musí zúčastniť každý študent, pre ktorého je aktivita určená. Skupina študentov, pre ktorú je aktivita určená môže byť nastavená dvomi spôsobmi – automaticky podľa zapísaných študentov na predmety priradené aktivite alebo explicitne previazaním aktivity s objektami triedy `Group`.

**RoomType** – trieda, ktorej inštancie tvoria číselník typov miestností – napr. poslucháreň, laboratórium, telocvičňa, učebňa, knižnica, atď. Každá inštancia tejto triedy môže mať



Obr. 4: Triedy balíka „Data“

priradených viaceru inšancií triedy Room. Medzi týmito dvomi triedami je vzťah typu kompozícia.

**Room** – trieda s reflexívnou asociáciou pre uchovanie a prácu s hierarchickým modelom miestností, ktorý umožňuje zoskupiť viaceru miestností do jedného celku alebo naopak

jednu miestnosť rozdeliť na niekoľko logických častí. Obsahuje väzbu typu M:N na triedu `Equipment` – pomôcka. Atribút `count` asociačnej triedy `RoomEquipmentConnection` udáva počet pomôcok daného typu, ktoré sa nachádzajú v danej miestnosti.

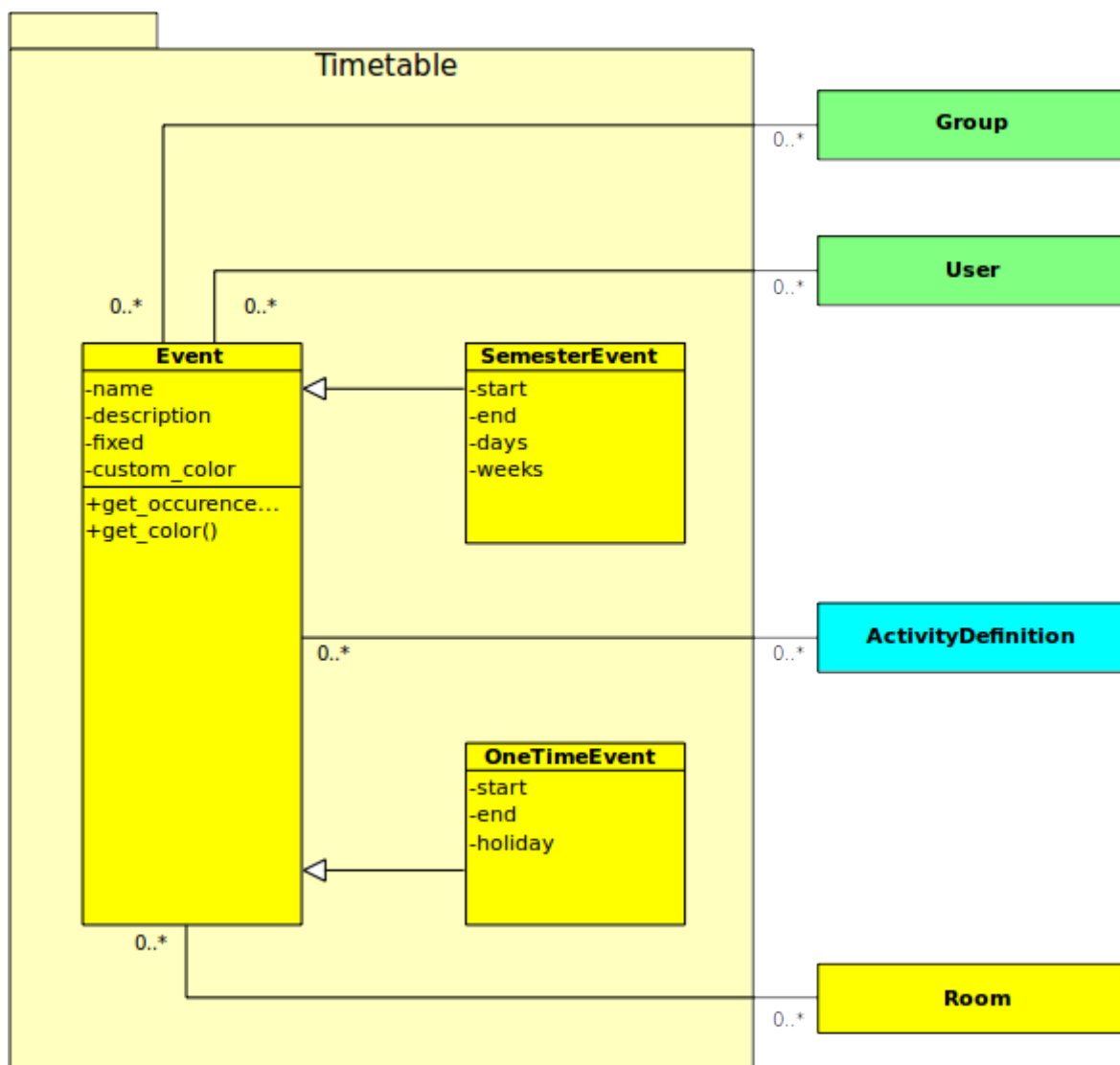
**Department** – trieda pre prácu s hierarchickým modelom pracovísk. Usporiadanie objektov tejto triedy závisí od stromovej štruktúry pracovísk konkrétnej univerzity/fakulty. Uzлами môžu byť napr. fakulty, ústavy, katedry, atď. Asociáciami typu M:N je trieda `Department` prepojená s triedami `User`, `Subject` a `Room`. Atribút `relation_type` asociačnej triedy `UserDepartmentConnection` – celé číslo vyjadrujúce typ prepojenia (číselník typov prepojení je možné nastaviť v module *Settings*, napr. interný zamestnanec, externý zamestnanec, doktorand).

## 4.3 Balíček „Timetable“

**Event** – trieda, ktorej objekty – udalosti – reprezentujú rozvrhové akcie. Obsahuje väzby typu asociácia násobnosti M:N s triedami balíka *Data* – `User`, `ActivityDefinition` a `Room`. Atribút `fixed` s hodnotou `True` označuje fixovanie rozvrhovej akcie (viď kapitola 3.2.6). Voliteľný atribút `custom_color` umožňuje nastaviť farbu pre udalosť. Návrátová hodnota metódy `get_color` (kód farby pre zobrazenie udalosti) je neprázdny atribút `custom_color` danej udalosti, prvej aktivity (inštancia triedy `ActivityDefinition`), jej typu (inštancia triedy `ActivityType`), predvolená hodnota prototypu aktivity alebo prázdny reťazec. Metóda `get_occurences` vracia zoznam termínov výskytu udalosti (má význam najmä pri periodických udalostiach).

**SemesterEvent** – trieda odvodená od triedy `Event` určená na reprezentáciu periodických udalostí – rozvrhové akcie predmetov počas semestra. Atribúty `start` (začiatok) a `end` (koniec) sú typu „čas“ a spolu s atribútmi `days` (dni v týždni) a `weeks` (čísla týždňov semestra) vymedzujú rozsahy výskytov udalosti.

**OneTimeEvent** – trieda odvodená od triedy `Event` určená na reprezentáciu jednorázových udalostí – rozvrhové akcie počas skúškového obdobia, mimoriadne udalosti počas semestra, dni voľna (nastavenie atribútu `holiday` na hodnotu `True`). Atribúty `start` (začiatok) a `end` (koniec) sú typu „dátum + čas“ a vymedzujú rozsah trvania udalosti.



Obr. 5: Triedy balíka „Timetable“

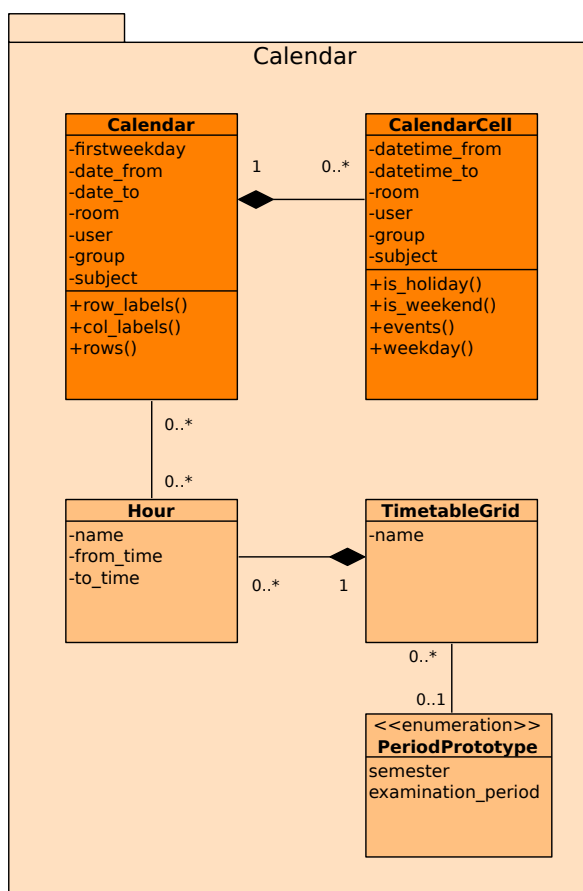
## 4.4 Balíček „Calendar“

**TimetableGrid** – trieda, ktorej inštanciou je časová mriežka pre zobrazenie rozvrhu. Môže byť dvoch typov – „semester“ (mriežka pre zobrazenie rozvrhu hodín na semester) alebo „examination period“ (mriežka pre zobrazenie rozvrhu skúšok). Každá mriežka môže obsahovať niekoľko hodín, čo vyjadruje vzťah typu kompozícia s triedou **Hour**, v ktorej je **TimetableGrid** nadradenou triedou. Inštancia triedy **Hour** vymedzuje čas (od – do) jednej hodiny časovej mriežky.

**Calendar** – inštancia tejto triedy reprezentuje jedno zobrazenie kalendára – tabuľky s rozvrhom hodín. Povinnými atribútmi sú `date_from` (dátum od), `date_to` (dátum do)

a zoznam hodín časovej mriežky (objekty triedy `Hour`). Voliteľne je možné nastaviť aj ďalšie obmedzenia pre zobrazenie rozvrhu – atribúty `room` (miestnosť), `user` (používateľ – študent alebo učiteľ), `group` (skupina používateľov), `subject` (predmet). Podľa rozsahu dátumov kalendára a zoznamu hodín sa vytvorí príslušný počet buniek – objektov triedy `CalendarCell`.

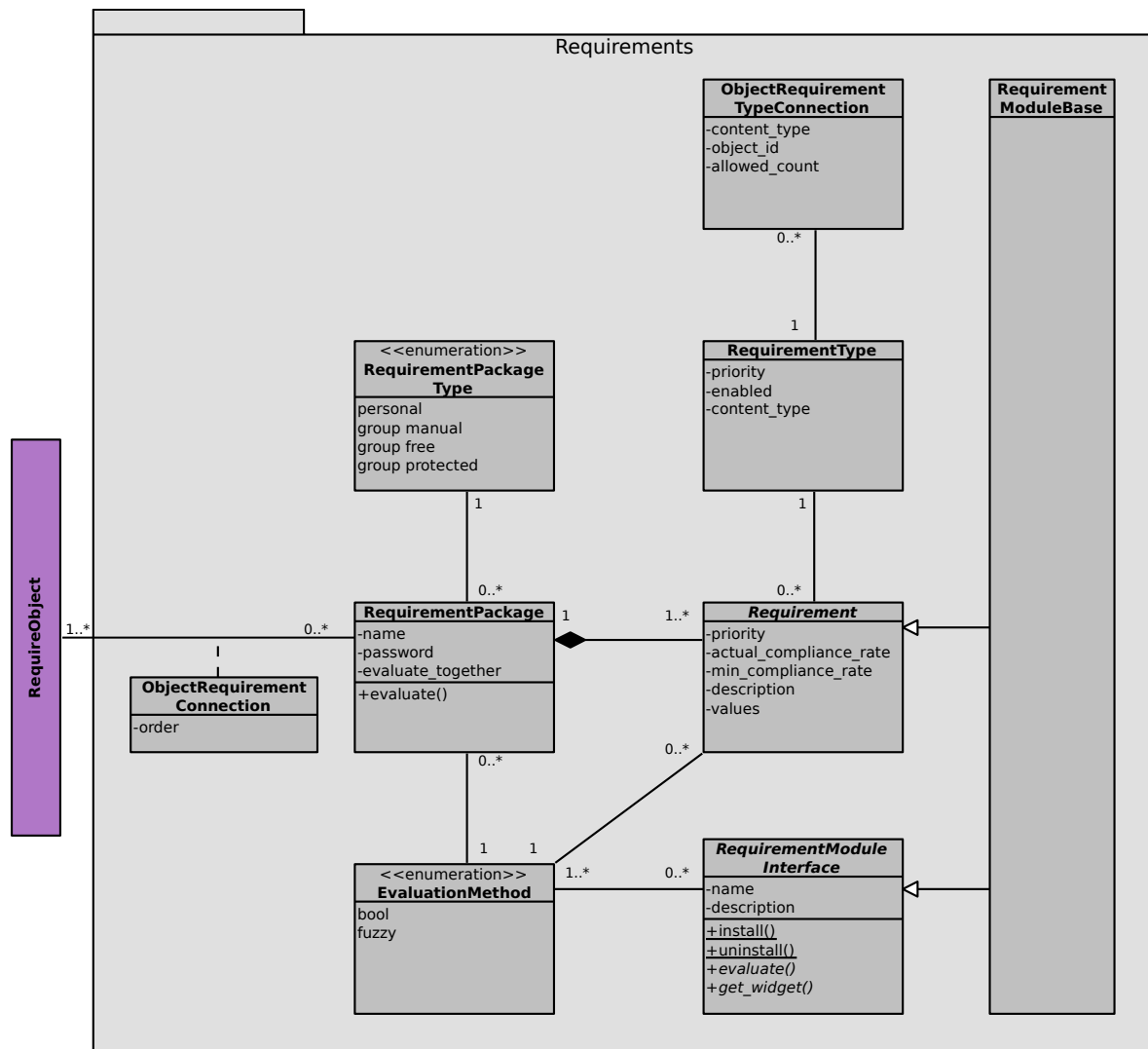
**CalendarCell** – inštancia tejto triedy reprezentuje jednu bunku v kalendári. Atribúty `datetime_from` a `datetime_to` vymedzujú časový rozsah bunky. Ďalšími atribútmi sú smerníky na inštancie miestností, používateľov, skupín, predmetov. Metóda `events()` vracia zoznam udalostí, ktorých časový rozsah pokrýva časový rozsah bunky a zároveň sú asociované s rovnakými miestnosťami, používateľmi, skupinami a predmetmi ako daná bunka. Metóda `is_holiday()` vracia `True`, ak aspoň jedna udalosť zo zoznamu udalostí bunky označuje voľný deň (má nastavený atribút `holiday` na `True`). Metóda `is_weekend()` vracia `True`, ak sa časové rozmedzie bunky pokrýva s víkendom.



Obr. 6: Triedy balíka „Calendar“



## 4.5 Balíček „Requirements“



Obr. 7: Triedy balíka „Requirements“

**RequirementType** – objektami tejto triedy sú nainštalované moduly typov požiadaviek na rozvrh. Atribút `content_type` je smerníkom na triedu modulu. Typy objektov (prípadne konkrétne inštancie), ktorým je možné nastaviť daný typ požiadavky, sú obsiahnuté v objektoch triedy `ObjectRequirementTypeConnection`, kde atribút `content_type` je smerník na triedu typu objektu, `object_id` je (voliteľne) je smerník na inštanciu konkrétneho objektu a `allowed_count` je povolený počet inštancií daného typu požiadavky.

**Requirement** – abstraktná trieda, ktorá je základom každého typu požiadavky na rozvrh. Význam atribútov:

- `priority` – priorita požiadavky,
- `min_compliance_rate` – minimálna miera splnenia požiadavky (hranica, ktorú sa snaží dosiahnuť rozvrhár, prípadne generátor rozvrhu),
- `actual_compliance_rate` – aktuálna miera splnenia požiadavky,
- `evaluation_method` – zvolený typ vyhodnotenia pre konkrétnu požiadavku,
- `values` – serializované hodnoty konkrétneho typu požiadavky.

**RequirementModuleInterface** – rozhranie definujúce abstraktné atribúty (`name` – názov, `description` – popis, `evaluation_methods` – dostupné metódy vyhodnotenia požiadavky) a metódy (`evaluate()` – vráti mieru splnenia požiadavky, `get_widget()` – vráti formulárové prvky pre úpravu hodnôt požiadavky), ktoré musí implementovať každý modul typu požiadavky. Statické metódy `install()` a `uninstall()` slúžia na nainštalovanie a odinštalovanie modulu (t.j. vytvorenie alebo zmazanie objektu triedy `RequirementType`).

**RequirementModuleBase** – trieda odvodená od triedy `Requirement`, implementujúca `RequirementModuleInterface`. Neobsahuje žiadne ďalšie atribúty ani metódy. Je základom, z ktorého ďalej dedia všetky moduly s typom požiadavky (v balíku `RequirementModules`). Inštanciami daných modulov sú konkrétne požiadavky na rozvrh.

**RequirementPackage** – balík požiadaviek – trieda, ktorej objekt obsahuje zoskupenie požiadaviek (na triedu `Requirement` sa viaže vzťahom typu kompozícia násobnosti 1:N). Požiadavky jedného objektu na rozvrh sa môžu nachádzať vo viacerých balíkoch (väzba násobnosti M:N s asociačnou triedou `ObjectRequirementConnection` pre určenie poradia dôležitosti jednotlivých balíkov požiadaviek pre daný objekt). Hodnota atribútu `evaluate_together` určuje spôsob vyhodnotenia splnenia požiadaviek balíka (všetky požiadavky spolu alebo každá požiadavka osobitne). Metódu vyhodnotenia určuje hodnota atribútu `EvaluationMethod` (podľa *Boolean* alebo *Fuzzy* logiky). Balíky môžu byť nasledovných typov (enumerácia `RequirementPackageType`):

- *Personal* – osobné požiadavky objektu (`RequireObject`). Každý objekt môže mať najviac jeden balík tohto typu, ktorý zároveň už nemôže byť priradený žiadnemu inému objektu.
- *Group manual* – špeciálny typ balíku, ktorý môže rozvrhár priradiť manuálne viacerým objektom, ktorým chce priradiť rovnaké požiadavky. Napr. môže vytvoriť balík „Požiadavky senátorov“, ktorý priradí všetkým učiteľom a študentom, ktorí sa pravidelne zúčastňujú zasadaní senátu, a tak im spoločne zabezpečí voľno v stanovenom termíne.
- *Group free* – balík s pripravenými konkrétnymi požiadavkami na rozvrh, ktorý si môže sám pridať medzi svoje balíky ľubovoľný používateľ.
- *Group protected* – podobný ako *Group free*, s tým rozdielom, že balík je chránený (atribút `password`) a môže si ho pridať len ten, kto pozná heslo.

## 4.6 Balíček „FEI“

Tento balíček obsahuje jedinú triedu – `FEI` – ktorej metóda `csv_import()` postupne volá metódy pre vytvorenie objektov zo záznamov v jednotlivých zdrojových súboroch (kompletná špecifikácia formátu zdrojových súborov sa nachádza v prílohe A.4). Nasledovné poradie volania importovacích funkcií je dôležité dodržať (kvôli zachovaniu konzistencie prepojených modelov):

1. `create equipments()` – vytvorenie pomôcok (objektov triedy `Equipment`),
2. `create roomtypes()` – vytvorenie typov miestností (objektov triedy `RoomType`),
3. `create departments()` – vytvorenie pracovísk (objektov triedy `Department`),
4. `create rooms()` – vytvorenie miestností (objektov triedy `Room`),
5. `create rooms equipments()` – vytvorenie prepojení miestností a pomôcok (objektov triedy `RoomEquipmentConnection`),
6. `create groups()` – vytvorenej skupín (objektov triedy `Group`),
7. `create users()` – vytvorenej používateľov (objektov triedy `User`),
8. `create users groups()` – vytvorenie prepojení používateľov a skupín (objektov triedy `UserGroupConnection`),

9. `create_users_departments()` – vytvorenie prepojení používateľov a pracovísk (objektov triedy `UserDepartmentConnection`),
10. `create_studytypes()` – vytvorenie typov štúdia (objektov triedy `StudyType`),
11. `create_subjects()` – vytvorenie predmetov (objektov triedy `Subject`),
12. `create_subjects_studytypes()` – vytvorenie prepojení predmetov a typov štúdia (objektov triedy `SubjectStudyTypeConnection`),
13. `create_subjects_users()` – vytvorenie prepojení predmetov a používateľov (objektov triedy `UserSubjectConnection`),
14. `create_activitydefinitions()` – vytvorenie definícií aktivít pre jednotlivé predmety (objektov triedy `ActivityDefinition`).

Balíček FEI bude v budúcnosti možné rozšíriť aj o funkcie na export vytvoreného rozvrhu do AIS.

## 4.7 Balíček „Settings“

Balíček pre centralizovanú správu nastavení všetkých ostatných súčastí rozvrhového systému. Pre účely diplomovej práce bude postačovať konfigurácia pomocou textového súboru (viď príloha A.3). V budúcnosti bude možné implementovať podporu grafického používateľského rozhrania.

## 4.8 Balíček „Authentication“

Úlohou balíčka *Authentication* je identifikovať používateľa po vyplnení prihlasovacích údajov. Predstavuje oddeliteľnú časť systému, ktorú je možné implementovať osobitne pre potreby konkrétnej školy (napr. s prepojením na UIS). V tejto práci sme sa na jeho funkcionality viac nezameriavali. Rozhodli sme sa implementovať štandardný spôsob autentifikácie odporúčaný pre kombináciu zvolených implementačných techník.

## 5 IMPLEMENTÁCIA

V tejto kapitole sa budeme zaoberať praktickou realizáciou jadra navrhnutého systému. Zdôvodníme výber programovacích jazykov a použitých knižníc, popíšeme základnú štruktúru projektu, objasníme riešenie vybraných implementačných problémov. V závere tejto kapitoly uvedieme odporúčané nástroje a prostredie pre nasadenie aplikácie.

### 5.1 Výber programovacích jazykov a knižníc

Vlastnú aplikáciu sme sa rozhodli implementovať v jazyku **Python** (verzia 2.7) v kombinácii s jeho populárnym webovým *framework-om* **Django** (verzia 1.6) a databázou **PostgreSQL** (verzia 9.1). Na spríjemnenie používateľského rozhrania sme použili **Bootstrap** (verzia 3.1) a **JavaScript** s knižnicou **jQuery** (verzia 1.11).

**Python** – otvorený, multiplatformový a stabilný jazyk vysokej úrovne. Medzi jeho ďalšie výhody patria prehľadnosť, pohodlnosť a rýchlosť písania zdrojového kódu, perfektná dokumentácia, množstvo zabudovaných aj voliteľných knižníc, široká používateľská základňa. Uprednostnili sme verziu 2.7, ktorá je ešte stále štandardnou verziou Python-u vo väčšine operačných systémov, pred novšími nekompatibilnými verziami 3.X, ktoré zatiaľ nemajú takú podporu knižníc tretích strán [23].

**Django** – vyspelý webový *framework* napísaný v jazyku Python, určený „perfekcionistom, ktorí potrebujú stíhať termíny“. Snaží sa čo najviac vecí automatizovať a riadi sa princípom DRY (z anglického „*Don't Repeat Yourself*“ – neopakuj sa). Ponúka prehľadné a pružné nastavenie pekných URL adries. Obsahuje rozšíriteľný a zrozumiteľný šablónovací jazyk, vyspelé objektovo-relačné mapovanie ideálne pre tvorbu databázovej aplikácie s automatickou tvorbou jednoduchých formulárov a v neposlednom rade pohodlné ladiace nástroje. Podobne ako samotný Python, aj Django disponuje rozsiahlou komunitou používateľov, prehľadnou referenčnou príručkou a tiež veľmi kvalitne spracovanými tematickými návodmi [20].

Využili sme aj nasledovné balíčky tretích strán pre Django:

- **django-simple-menu** – podpora tvorby viacúrovňového menu a označenie aktívnej menu položky [2],
- **django-bootstrap3** – funkcie pre šablóny na zobrazenie formulárových prvkov v štýle Bootstrap [15],
- **django-mptt** – hierarchické modely (modely s reflexívnou asociáciou, ktoré obsahujú cudzí kľúč sami na seba) [13],
- **django-bitfield** – podpora bitových atribútov pre modely [16],
- **django-debug-toolbar** a **django-extensions** – nástroje pre vývoj a ladenie [22] [14].

**PostgreSQL** – najznámejšia databáza, ktorá je slobodným softvérom a v Django-u má najlepšiu podporu [18].

**Bootstrap** – v súčasnosti najobľúbenejšia *front-end* knižnica pre vývoj webových dizajnov s variabilnou šírkou, ktorá ponúka jednoduché a pre používateľa príjemné štýly zobrazenia najčastejšie používaných komponentov webových stránok. Obsahuje sadu 200 vektorových ikôn, ktoré je možné použiť na stránke v ľubovoľnej veľkosti. Náročnejší dizajnéri si môžu upraviť farby, veľkosti a štýly pomocou *Less*<sup>4</sup> súborov a premenných [11].

**JavaScript** – skriptovací programovací jazyk, ktorý je štandardnou súčasťou moderných webových prehliadačov. Používa sa najmä na interakciu webovej stránky s používateľom na klientskej strane, asynchrónnu komunikáciu a dynamickú úpravu zobrazeného obsahu (webovej stránky). Keďže syntax tohto jazyka je dosť ťažkopádna a nejednotná v jednotlivých internetových prehliadačoch, na serióznu prácu sa používajú knižnice (napr. jQuery, Prototype, MooTools), ktoré poskytujú množstvo užitočných funkcií garantujúcich kompatibilitu medzi prehliadačmi.

**jQuery** – v súčasnosti najpoužívanejšia JavaScript knižnica, ktorá poskytuje množstvo funkcií na manipuláciu s obsahom webového dokumentu, spracovanie udalostí, animácie, asynchrónnu komunikáciu so serverom. Veľkou výhodou tejto knižnice je veľká

---

<sup>4</sup> Less je rozšírenie jazyka CSS, ktoré umožňuje definovať premenné, funkcie a iné vlastnosti, ktoré uľahčujú vytváranie a rozširovanie tém. Na konverziu Less súborov do CSS existuje množstvo nástrojov.

používateľská základňa a množstvo zásuvných modulov a knižníc založených na jQuery dostupných z tretích strán [12].

V našom projekte sme využili nasledovné jQuery knižnice:

- **jQuery UI** – množina efektov, formulárových prvkov, tém a iných nástrojov na interakciu s používateľským rozhraním (napr. *drag&drop* premiestňovanie objektov) [17],
- **jQuery UI Timepicker** – formulárový prvok na pohodlné vkladanie času a časového rozsahu do formuláru v štýle jQuery UI [3],
- **jQuery UI Colorpicker** – formulárový prvok na výber farby v štýle jQuery UI [10],
- **jQuery Form Plugin** – odosielanie formulárov cez AJAX [1],
- **perfect scrollbar** – zobrazenie posuvníka pre rolovanie elementu na stránke [4].

## 5.2 Štruktúra projektu

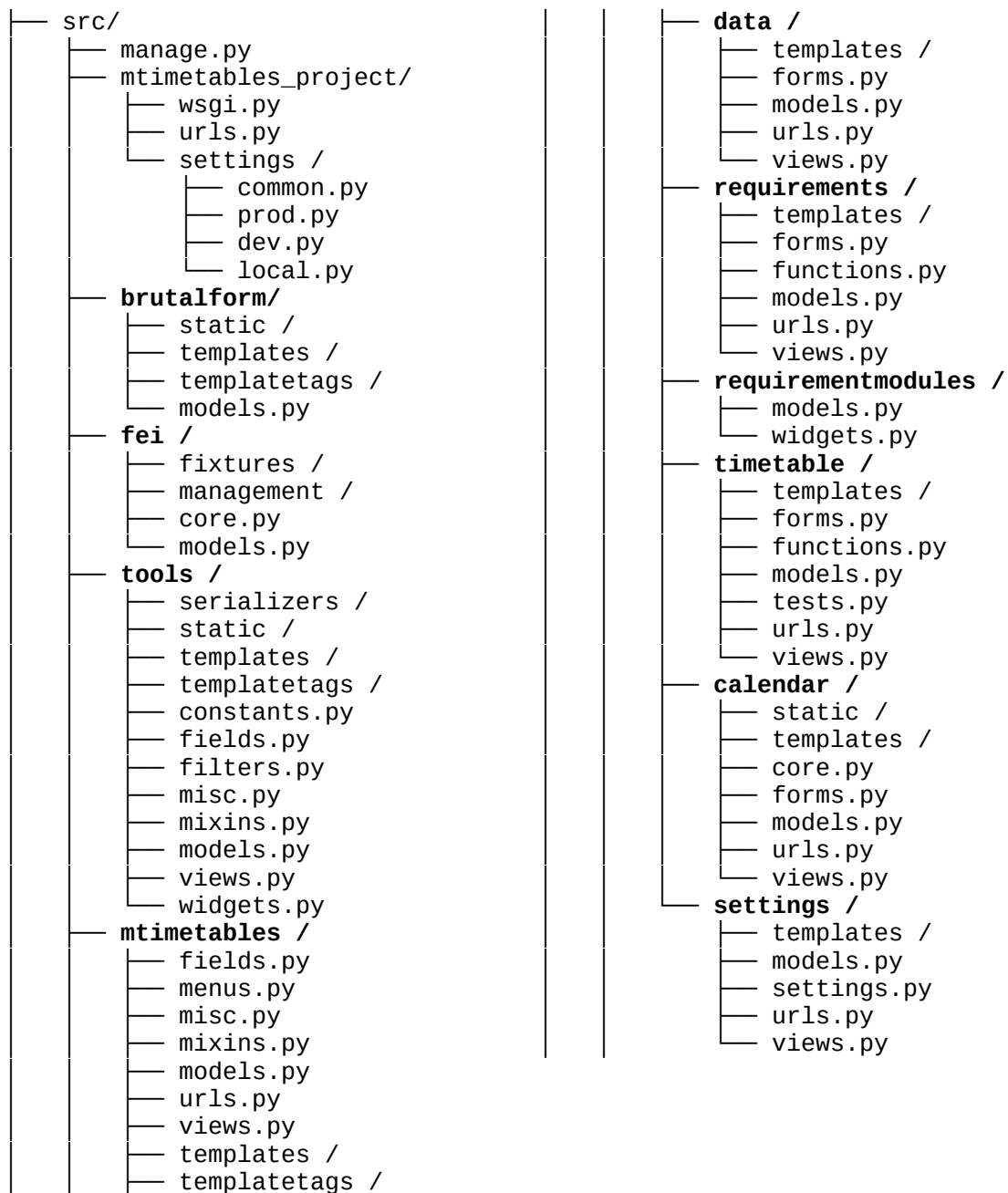
Adresárová štruktúra projektu je rozdelená nasledovným spôsobom:

- *doc* – adresár s dokumentáciu projektu
- *requirements* – adresár obsahujúci textové súbory so zoznamami závislostí aplikácie pre ostré nasadenie (*prod.txt*) a vývoj (*dev.txt*)
- *src* – zdrojové kódy projektu – počiatočný adresár Django projektu

V ďalšej časti tejto kapitoly bližšie popíšeme súborovú štruktúru adresára *src* (neuvádzame súbory `__init__.py`, ktoré obsahuje každý Python balík), tučným písmom sú zvýraznené názvy adresárov, ktoré sú zároveň Django aplikáciami<sup>5</sup>:

---

5 Django aplikácia je Python balík, ktorý je súčasťou Django projektu a obsahuje súbor *models.py* (alebo adresár *models*) s modelmi, podľa ktorých Django (po spustení príkazu *python manage.py syncdb*) vytvára štruktúru databázy. Jeden Django projekt obvykle obsahuje viacero aplikácií. Dobre napísaná aplikácia môže byť opätovne použitá v ďalších projektoch.



*manage.py* – pomocný skript, ktorý je súčasťou inštalácie Django-a, slúžiaci na interakciu s celým Django projektom (napr. synchronizácia modelov a databázy)

*mtimetables\_project* – základný Python balík celého projektu (ďalej ako **adresár projektu**), ktorý obsahuje vstupný bod pre webový server (súbor *wsgi.py*), deklarácie URL adries (súbor *urls.py*) a adresár *settings* s konfiguračnými súbormi webovej aplikácie:

- *common.py* – všeobecné nastavenia,
- *prod.py* – nastavenia produkčnej aplikácie,



- *dev.py* – nastavenia pre vývojárske účely,
- *local.py* – súbor s lokálnymi nastaveniami (prístup k databáze, bezpečnostný kľúč, atď.), ktorý nie je zahrnutý v repozitári.

Štruktúra jednotlivých Django aplikácií vychádza zo štandardu Django-a a riadi sa nasledovnými pravidlami:

- *templates* – adresár s html šablónami aplikácie,
- *templatetags* – adresár obsahujúci vlastné funkcie využiteľné v šablónach,
- *static* – adresár so statickými súborami (obrázky, skripty, štýly) aplikácie,
- *fixtures* – adresár obsahujúci súbory so serializovanými dátami modelov, ktoré je možné načítať do databázy po spustení príkazu `python manage.py loaddata <súbor>` (dáta v súboroch *fixtures/initial\_data.json* budú načítané do databázy automaticky pri jej synchronizácii – `python manage.py syncdb`),
- *management* – adresár pre definíciu vlastných funkcií príkazového riadka (atribúty príkazu `python manage.py`),
- *models.py* – definície modelov aplikácie,
- *core.py* – definície tried aplikácie, ktoré nie sú Django modelmi<sup>6</sup>,
- *fields.py* – definície vlastných dátových typov pre atribúty modelov,
- *forms.py, widgets.py* – definície formulárov a ich prvkov využívaných aplikáciou,
- *tests.py* – jednotkové testy dôležitých funkcií aplikácie,
- *urls.py* – deklarácie URL adries aplikácie,
- *views.py* – definície pohľadov,
- *mixins.py* – definícia pomocných tried a funkcií pre *views.py*,
- *functions.py, misc.py* – špeciálne funkcie využívané v aplikácii,

**brutalform** – aplikácia podporujúca tvorbu a zobrazenie formulárov pre modely obsahujúce cudzí kľúč na iný model.

**fei** – aplikácia pre import dát z AIS FEI STU pomocou príkazu `python manage.py fei_import <súbor1, súbor2, ..., súborN>`. Špecifikácia formátu vstupných dát je popísaná v kapitole A.4.

---

<sup>6</sup> Django model je trieda, ktorej objekty sa ukladajú do databázy pomocou automatického Django ORM.

**tools** – znovu použiteľná aplikácia poskytujúca všeobecné šablóny, štýly a skripty pre zobrazenie základných typov stránok webových aplikácií – napr. zoznam, detail, formulár, chybové stránky, atď. V adresári *serializers* sa nachádza vlastný serializér do JSON formátu.

**mtimetables** – aplikácia zahŕňajúca funkcionality jadra rozvrhového systému. Priamo vo svojej adresárovej štruktúre obsahuje ďalšie Django aplikácie (*data*, *requirements*, *requirementmodules*, *timetable*, *calendar*, *settings*), ktorých funkcionality sme popísali v návrhu (viď kapitoly 4.2 – 4.7). V súbore *menus.py* sú zaregistrované jednotlivé položky menu.

## 5.3 Model údajov

Vďaka prepracovanému Django ORM nie je potrebné osobitne udržiavať logický a fyzický model. Modely sú definované len raz – obvykle v každej Django aplikácii v súbore s názvom *models.py*. Ide o obyčajné triedy jazyka Python, ktoré v svojej hierarchii dedičnosti obsahujú triedu `django.db.models.Model` poskytujúcu API pre konštruovanie databázových dotazov. Pri spustení príkazu `manage.py syncdb` Django automaticky vytvorí potrebné databázové tabuľky, ich prepojenia a indexy pre jednotlivé modely.

Modely balíčka *Data*:

- `Department`,
- `RequireObject`,
- `Group`,
- `User`,
- `UserEventConnection`,
- `UserDepartmentConnection`,
- `Subject`,
- `StudyType`,
- `SubjectStudyTypeConnection`,
- `UserSubjectConnection`,
- `ActivityType`,

- `ActivityDefinition`,
- `Room`,
- `RoomType`,
- `Equipment`,
- `RoomEquipmentConnection`.

Modely balíčka *Timetable*:

- `Event`,
- `SemesterEvent`,
- `OneTimeEvent`.

Modely balíčka *Requirements*:

- `RequirementType`,
- `Requirement`,
- `RequirementPackage`,
- `ObjectRequirementConnection`,
- `ObjectRequirementTypeConnection`.

Modely balíčka *Calendar*:

- `TimetableGrid`,
- `Hour`.

## 5.4 Riešenie vybraných implementačných problémov

V tejto kapitole popíšeme riešenie zaujímavých implementačných problémov, s ktorými sme sa stretli. Vysvetlíme typy dedičnosti vo *framework-u* Django a ich výber v konkrétnych situáciách. Objasníme spôsob smerovania URL adries, hierarchiu dedičnosti vytvorených pohľadov a elegantné riešenie dynamicky vytváraných smerovacích pravidiel a pohľadov. Nakoniec vysvetlíme spôsob inštalácie modulov typov požiadaviek a spôsob uchovávaní množín dní a týždňov formou bitovej masky.

### 5.4.1 Dedičnosť modelov

Dedičnosť modelov v Django-u funguje podobným spôsobom ako dedičnosť klasických tried v Python-e (vrátane viacnásobnej dedičnosti). Na základe rozhodnutia, či rodičovská trieda bude tiež modelom (so svojou vlastnou databázovou tabuľkou) alebo nie, rozlišujeme tri typy dedičnosti [20]:

1. **„Abstract base classes“** – rodičovská trieda je označená ako abstraktná. Používa sa najmä v prípadoch, keď potrebujeme nejaké základné informácie implementovať vo viacerých modeloch. Takáto trieda nie je inštanciovateľná a nemôže byť nikdy použitá samostatne (napr. nie je možné sa na ňu odkazovať cudzím kľúčom). Nemá svoju databázovú tabuľku, ale jej atribúty sa nachádzajú v tabuľkách odvodených modelov.
2. **„Multi-table inheritance“** – každá trieda v hierarchii dedičnosti je modelom (má svoju vlastnú databázovú tabuľku) a môže byť inštanciovateľná osobitne. V odvodenej triede Django automaticky vytvára odkazy na každý rodičovský model. Pri inštanciácii odvodeného modelu databázový dotaz automaticky vyberá informácie aj zo všetkých tabuliek rodičovských modelov, takže je rozloženie atribútov modelu do viacerých tabuliek pre programátora takmer úplne transparentné.
3. **„Proxy models“** – odvodené triedy, ktoré menia funkcionality modelu bez potreby uchovávať v databáze ďalšie atribúty. Nie sú mapované na vlastnú databázovú tabuľku, pretože všetky ich atribúty sú uložené v tabuľkách rodičovských modelov. Sú inštanciovateľné rovnakým spôsobom ako obyčajné Django modely.

#### 5.4.1.1 Prepojenie žiadateľov a požiadaviek

Pod žiadateľom budeme rozumieť objekt, ktorý môže mať požiadavku na rozvrh. V návrhu sme identifikovali typov žiadateľov ako inštancie tried `Group`, `User`, `Subject`, `ActivityDefinition`, `ActivityType`, `Room`, `RoomType`, ktorých základom je trieda `RequireObject`. Dedičnosť sme implementovali ako tzv. *Multi-table inheritance*.

#### 5.4.1.2 Prepojenie požiadaviek a ich modulov

Moduly typov požiadaviek sú triedy balíčka *Requirement Modules*, odvodené od triedy *RequirementModuleBase* z balíčka *Requirements* (viď obrázok 7). Každý modul požiadavky bude obsahovať okrem zdedených atribútov od triedy *Requirement* aj vlastné atribúty – parametre požiadavky – ktorých počet a typ bude závisieť od konkrétneho typu požiadavky.

Ako je zjavné z návrhu, na triedu *Requirement* sa prostredníctvom cudzích kľúčov odkazujú iné triedy, preto ju nemôžeme označiť ako abstraktnú z hľadiska Django dedičnosti – nemôžeme použiť dedičnosť typu *Abstract base classes*.

Keďže modulov požiadaviek bude rádovo niekoľko desiatok a musia byť jednoducho prídávateľné, bolo by nepraktické, aby každý model bol mapovaný na vlastnú databázovú tabuľku. Nemôžeme teda použiť ani tzv. *Multi-table inheritance*.

Pre implementáciu dedičnosti modulov požiadaviek sme preto zvolili tretí typ – *Proxy models*, ktorý sme čiastočne upravili. Každú triedu s modulom požiadavky odvodenú od *RequirementModuleBase* sme označili ako *proxy model*. Požiadavky všetkých typov (inštancie všetkých modulov) sa budú ukladať do databázovej tabuľky pre model *Requirement* a hodnoty ich špecifických parametrov budú serializované v atribúte *values*. Pri inštanciacii požiadavky budú deserializovanými hodnotami atribútu *values* triedy *Requirement* inicializované jednotlivé parametre konkrétneho modulu požiadavky. Aj keď v terminológii dedičnosti Django modelov, trieda *Requirement* nie je abstraktná, z hľadiska objektovo orientovaného návrhu abstraktná je – nemá význam vytvárať jej objekty samostatne bez toho, aby boli inštanciami konkrétneho modulu požiadavky. Aby sme dosiahli požadované správanie, implementovali sme magickú metódu `__new__()` v triede *Requirement* tak, aby namiesto inštancií vlastnej triedy vytvárala inštancie konkrétnych modulov požiadaviek:

```

class Requirement(models.Model):
    ...
    @staticmethod
    def __new__(cls, *args, **kwargs):
        if cls == Requirement and len(args) > 2:
            requirement_type = get_object_or_404(RequirementType, pk=args[2])
            cls = requirement_type.class_object
        return super(Requirement, cls).__new__(cls)
    ...

```

### 5.4.2 Smerovanie

Pravidlá pre smerovanie (angl. *routing*) URL adries sa v Django-u obvykle nastavujú v súbore *urls.py* v adresári projektu (*mtimetables\_project/urls.py* – vid’ kapitola 5.2) ako mapovanie regulárnych výrazov URL adries na Python funkcie – pohľady (podrobnejšie vysvetlené v nasledujúcej kapitole). Definícia mapovania môže mať stromovú štruktúru – t.j. regulárnemu výrazu nie je priradená priamo funkcia pohľadu, ale ďalší strom mapovania regulárnych výrazov. Každý modul mapovania (väčšinou súbor *urls.py*) musí obsahovať premennú *urlpatterns* so zoznamom inštancií smerovacích pravidiel daného podstromu. Keďže ide o kód jazyka Python, môže byť URL mapovanie konštruované aj dynamicky.

V súbore *mtimetables\_project/urls.py* je smerovanie definované nasledovným spôsobom:

```

urlpatterns = [
    url(r'^login/$', 'django.contrib.auth.views.login'),
    url(r'^logout/$', 'django.contrib.auth.views.logout'),
    url(r'^$', include('mtimetables.urls', namespace='mtimetables')),
]
urlpatterns += staticfiles_urlpatterns()

```

To znamená, že ak pri spracovávaní HTTP požiadavky nastane zhoda s prvým alebo druhým regulárnym výrazom ('^login/\$' alebo '^logout/\$'), vykoná sa zabudovaná Django funkcia pohľadu pre prihlásenie alebo odhlásenie používateľa. Ak zhoda nenastane, bude sa pokračovať ďalej v hľadaní vhodného regulárneho výrazu v podstrome *mtimetables.urls* (t.j. v súbore *mtimetables/urls.py*). Posledný riadok uvedeného kódu do zoznamu smerovacích pravidiel pridá mapovania pre obsluhu statických súborov (obrázky, skripty, štyly, atď.).

V súbore *mtimetables/urls.py* sú smerovacie pravidlá definované nasledovne:

```
# home
url(r'^$', views.TemplateView.as_view(
    template_name='mtimetables/index.html'
)),
# custom data app urls:
url(r'^data/',
    include('mtimetables.data.urls', namespace='data')),
# custom timetable app urls:
url(r'^timetable/',
    include('mtimetables.timetable.urls', namespace='timetable')),
# custom requirements app urls:
url(r'^requirements/',
    include('mtimetables.requirements.urls', namespace='requirements')),
# custom calendar app urls:
url(r'^calendar/',
    include('mtimetables.calendar.urls', namespace='calendar')),
# custom settings app urls:
url(r'^settings/',
    include('mtimetables.settings.urls', namespace='settings')),
```

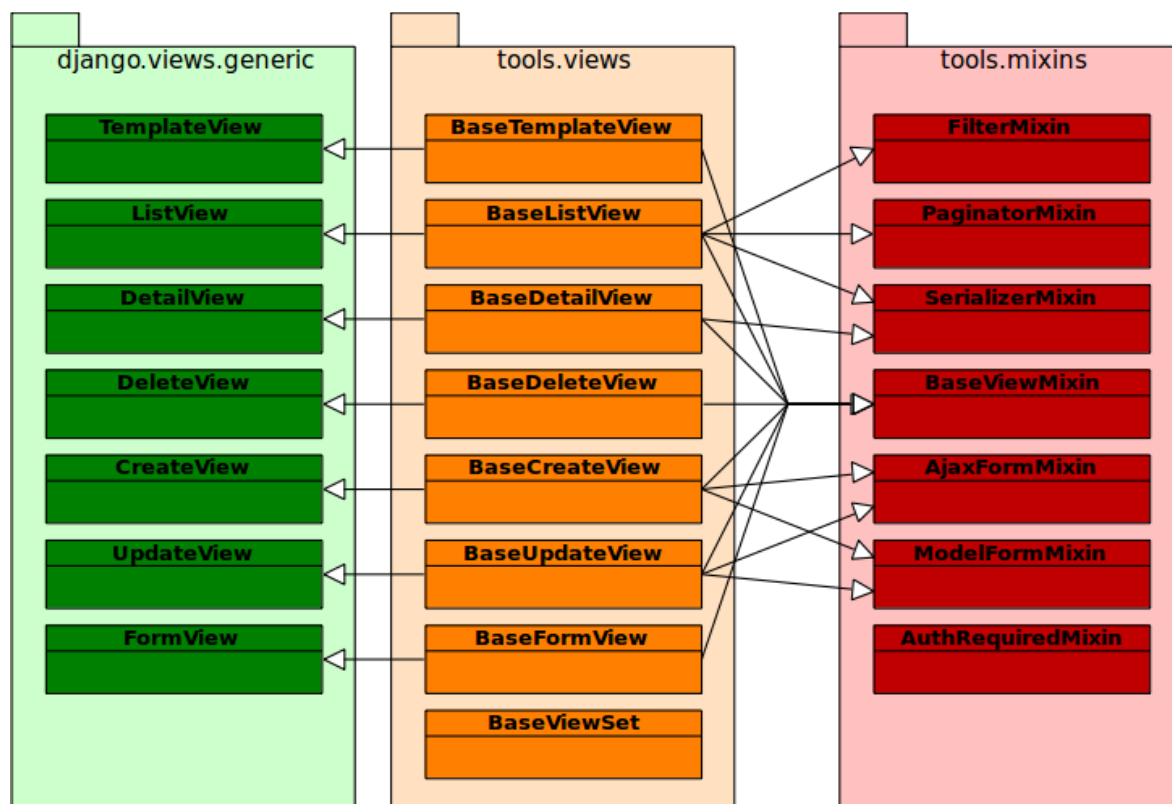
V jednotlivých súboroch *mtimetables/X/urls.py*, kde *X* je *data*, *timetable*, *requirements*, *calendar* alebo *settings* sú definované už iba mapovania regulárnych výrazov na konkrétne pohľady.

### 5.4.3 Pohľady

Django pohľad (z angl. *view*) je volateľný Python objekt (funkcia alebo „špeciálna“ trieda – tzv. *Class-based view* – ďalej len CBV), ktorý pre HTTP požiadavku vráti HTTP odpoveď (HTML stránku, presmerovanie, chybu 404, obrázok, atď.). Podľa Django konvencie sú pohľady definované v súboroch *views.py* v adresári projektu alebo aplikácie.

Kvôli štruktúrovanému, hierarchickému a znovu použiteľnému kódu, ktorý vychádza z filozofie CBV, sme sa ich rozhodli uprednostniť pred obyčajnými funkciami.

Ako vidno aj na obrázku č. 8, v module *views* aplikácie *tools* sme vytvorili základné CBV, od ktorých budú ďalej dediť všetky pohľady nášho systému. Rozširujú funkcionality Django *generic* CBV o automatické generovanie nadpisov, prepojenie s ostatnými povolenými pohľadmi daného objektu, generovanie systémových správ pri úspešnom a neúspešnom uložení objektu, kontrolu GET parametrov URL adresy, filtrovanie



Obr. 8: Hierarchia dedičnosti pohľadov *tools.views*

a stránkovanie zoznamu objektov, predvolené šablóny zobrazenia atď. Adresár *tools/templates* obsahuje okrem iného aj predvolenú šablónu pre každý typ pohľadu.

Každé CBV sa špecializuje na spracovanie iného typu akcie:

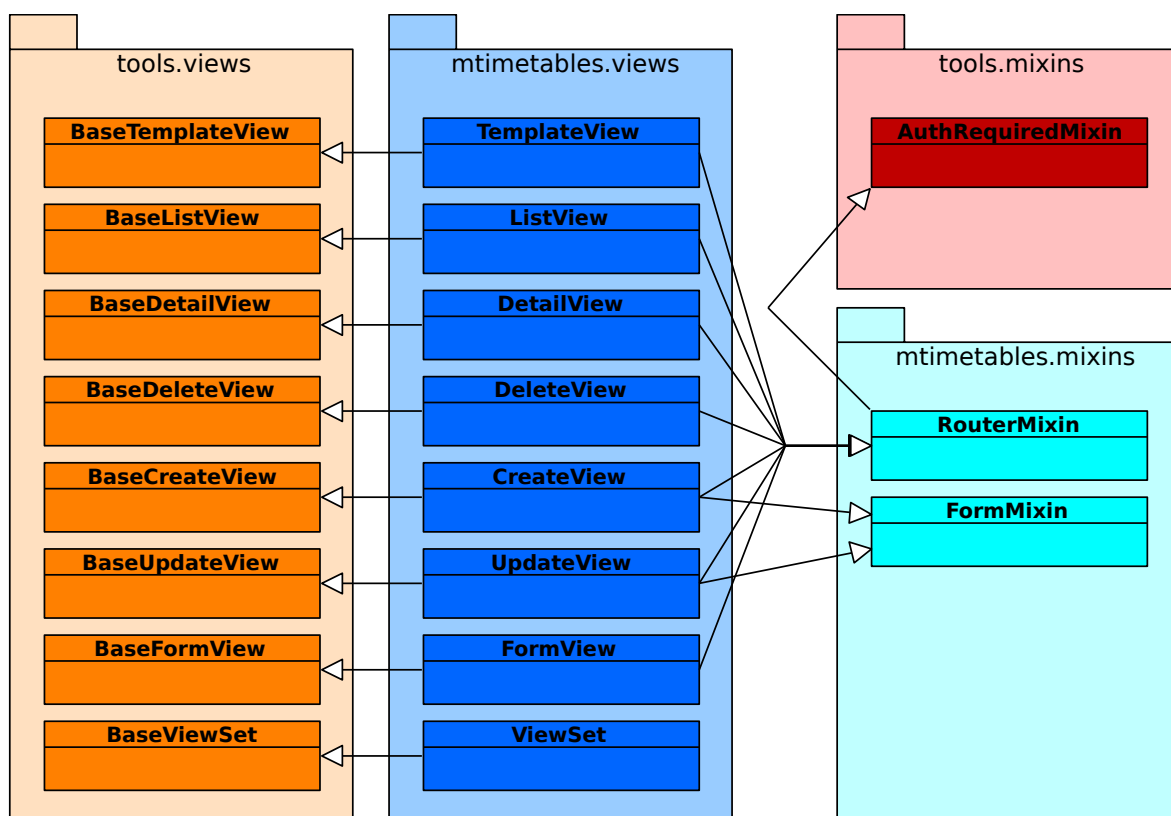
- `BaseTemplateView` – zobrazenie všeobecného HTML obsahu,
- `BaseListView` – zoznam objektov danej triedy s podporou stránkovania, filtrovania a serializácie,
- `BaseDetailView` – detail konkrétneho objektu s podporou serializácie,
- `BaseDeleteView` – zmazanie konkrétneho objektu,
- `BaseCreateView` – vytvorenie nového objektu danej triedy (aj cez AJAX),
- `BaseUpdateView` – úprava existujúceho objektu (aj cez AJAX),
- `BaseFormView` – všeobecné zobrazenie a spracovanie ľubovoľného formuláru (ktorý nie je viazaný na žiadny model).

`BaseViewSet` je *mixin* pre spoločné definovanie atribútov a metód všetkých typov CBV pre daný model. Jeho využitie bude vysvetlené v kapitole 5.4.4.



Základné CBV aplikácie *tools* predstavujú znovu použiteľný kód, ktorý je možné použiť (prípadne rozšíriť) v ďalších aplikáciách rozvrhového systému, ale aj úplne iných projektoch.

Aplikácia *mtimetables* definuje svoje vlastné pohľady odvodené od základných CBV aplikácie *tools* (viď obrázok 9). Najväčším rozšírením je kontrola identity používateľa pri zobrazení každého pohľadu – pomocou triedy *RouterMixin* odvodenej od základného autentifikačného *mixin-u* aplikácie *tools*. V budúcnosti je možné jednoducho doplniť kontrolu autorizácie odvodením triedy *RouterMixin* aj od nejakého autorizačného *mixin-u*. Tieto CBV sú ďalej základom všetkých pohľadov jednotlivých aplikácií jadra systému (*data*, *timetable*, *requirements*, atď.).



Obr. 9: Hierarchia dedičnosti pohľadov *mtimetables.views*

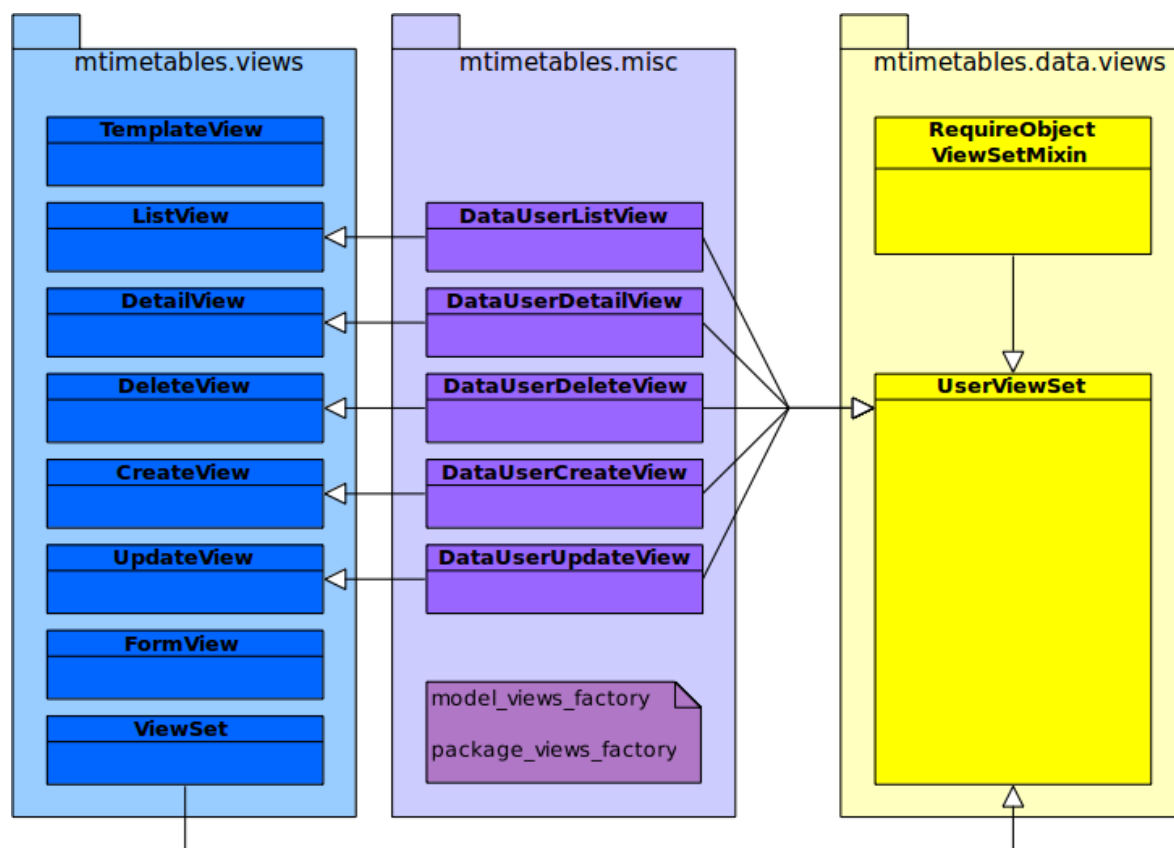
#### 5.4.4 Dynamické smerovanie a dynamické pohľady

Systémová požiadavka na možnosť úpravy zdrojových dát pre tvorbu rozvrhu spôsobila, že pre väčšinu modelov aplikácie *data* (ale aj niektoré modely iných aplikácií) sme potrebovali vytvoriť 5 osobitných pohľadov – *Detail*, *Create*, *Update*, *Delete* a *List* –

a k nim príslušné smerovacie pravidlá. Funkcionalita naprieč jednotlivými typmi pohľadov sa vo väčšine prípadov líši iba modelom prideleným danému CBV.

Funkcia `model_views_factory` (z modulu `misc`), pre daný model vytvorí množinu pohľadov a zoznam smerovacích pravidiel mapovaných na tieto pohľady. Každý pohľad je dynamicky vytvorená trieda CBV odvodená od CBV daného typu modulu `views` a tzv. *ViewSet-u* – predvolenej triedy `ViewSet` alebo inej z nej odvodenej.

Na obrázku 10 sa nachádza ukážka hierarchie dynamicky vytvorených pohľadov pre model `User` aplikácie `data`.



Obr. 10: Hierarchia dedičnosti pohľadov `mtimetables.misc`

Funkcia `package_views_factory` zapúzdruje vytvorenie dynamických pohľadov pre slovník, ktorého kľúčom je model a hodnotou je slovník ďalších parametrov funkcie `model_views_factory`. Príklad jej využitia pri vytvorení dynamických pohľadov a smerovacích pravidiel v module `mtimetables.data.urls`:

```

viewsets = collections.OrderedDict([
    ...
    (models.Department, {'viewset': views.DepartmentViewSet}),
    (models.Group, {'viewset': views.GroupViewSet}),
    (models.User, {'viewset': views.UserViewSet}),
    ...
])
urlpatterns = [
    ...
]
urlpatterns += mtimetables_misc.package_views_factory('data', viewsets)

```

### 5.4.5 Inštalácia modulov typov požiadaviek

Modulom typu požiadavky je každá trieda nachádzajúca sa v súbore *models.py* v aplikácii *requirementmodules*, ktorá je zdedená od triedy *RequirementModuleBase* z aplikácie *requirements*. Na detekciu a prípadnú inštaláciu modulov typov požiadaviek sa špecializuje pohľad *RequirementTypesInstallView* aplikácie *requirements*.

Pri každej HTTP požiadavke typu GET sa preverí dostupná množina modulov typov požiadaviek a aktuálne nainštalované typy – inštancie modelu *RequirementType* aplikácie *requirements*. Korektnou konštrukciou HTTP požiadavky typu POST je možné nainštalovať alebo odinštalovať vybrané moduly typov požiadaviek, čo sa taktiež prejaví vytvorením alebo zmazaním objektov triedy *RequirementType*.

Pre použitie nainštalovaných modulov typov požiadaviek je potrebné nastaviť typy objektov, ktorým môže byť daný typ požiadavky priradený – vytvoriť príslušné relácie *ObjectRequirementTypeConnection* (model aplikácie *timetable*).

### 5.4.6 Uchovávanie dní a týždňov

Atribúty *days* a *weeks* modelu *SemesterEvent* v balíčku *Timetable* (a tiež *weeks* v modeli *ActivityDefinition* v balíčku *Data*) sú bitové masky, kde jednotlivé bity reprezentujú poradové čísla dní (atribút *days*) a týždňov (atribút *weeks*) výskytu udalosti. Využili sme externú aplikáciu *django-bitfield* poskytujúcu databázové API a formulárové prvky na prácu s bitovými maskami.

## 5.5 Nasadenie aplikácie

Vytvorenú webovú aplikáciu sme testovali a uviedli do ostrej prevádzky na OS Debian GNU/Linux. Vzhľadom na platformovú nezávislosť použitých programovacích jazykov a knižníc, je možné aplikáciu spustiť aj na väčšine ostatných moderných operačných systémov.

Podrobný popis inštalácie a konfigurácie systému sa nachádza v priloženej technickej dokumentácii (vid' príloha A). Ukážky obrazoviek grafického používateľského a stručný návod sa nachádza v priloženej používateľskej príručke (vid' príloha B).

## 6 OVERENIE RIEŠENIA

Navrhnutá štruktúra rozvrhového systému pokrýva všetky vopred vytýčené požiadavky a prípady použitia.

Funkčnosť implementovaného jadra rozvrhového systému bola overovaná rôznymi spôsobmi počas celého procesu vývoja. V tejto kapitole sa budeme ďalej zaoberať najdôležitejšími testami, ktoré boli vykonané na finálnej verzii aplikácie – a to z pohľadu používateľa – tzv. testovanie „čierna skrinka“ (viď kapitola 6.1), ako aj z pohľadu testera oboznámeného s vnútornou štruktúrou systému – tzv. testovanie „biela skrinka“ (viď kapitola 6.2).

### 6.1 Testovanie „čierna skrinka“

Z pohľadu používateľa – rozvrhára – sme manuálne otestovali funkčnosť aplikácie prostredníctvom webového rozhrania. Snažili sme sa pokryť čo najviac možných scenárov použitia.

Korektné správanie sa jednotlivých formulárov sme preverili zadávaním rôznych vstupov (platných aj neplatných) a následne sme kontrolovali odozvu systému.

Dostupnosť hypertextových odkazov sme overili použitím slobodného softvéru LinkChecker verzie 9.2 [6], ktorý rekurzívne navštívil všetky URL adresy každej podstránky našej webovej aplikácie. Program bol spustený na vzorke testovacích dát, ktorá obsahovala 1 – 5 objektov každého modelu a ukážkové typy prepojení medzi jednotlivými objektami. Výstup programu LinkChecker sa nachádza na obrázku 11.

```
LinkChecker 9.2      Copyright (C) 2000-2014 Bastian Kleineidam
LinkChecker comes with ABSOLUTELY NO WARRANTY!
This is free software, and you are welcome to redistribute it
under certain conditions. Look at the file `LICENSE' within this
distribution.
Get the newest version at http://wummel.github.io/linkchecker/
Write comments and bugs to
https://github.com/wummel/linkchecker/issues
Support this project at
http://wummel.github.io/linkchecker/donations.html

Start checking at 2014-05-13 22:26:43+002
10 threads active, 14 links queued, 9 links in 2 URLs checked, runtime
1 seconds
...
6 threads active, 0 links queued, 13635 links in 337 URLs checked,
runtime 2 minutes, 1 seconds

Statistics:
Downloaded: 4.18MB.
Content types: 638 image, 8278 text, 0 video, 0 audio, 3123
application, 0 mail and 1872 other.
URL lengths: min=22, max=91, avg=48.

That's it. 13911 links in 337 URLs checked.
0 warnings found. 0 errors found.
Stopped checking at 2014-05-13 22:28:46+002 (2 minutes, 2 seconds)
```

*Obr. 11: Výstup programu LinkChecker 9.2*

## 6.2 Testovanie „biela skrinka“

Testovaním vnútornej štruktúry systému sme sa zaoberali počas celého vývoja – väčšinou manuálnym spôsobom. Vzhľadom na náročnosť manuálneho overovania správnosti kontroly kolízií udalostí, sme vytvorili automatický test, ktorého podrobnosti a výsledky uvádzame v kapitole 6.2.1.

### 6.2.1 Overenie funkcie pre kontrolu kolízií

Kľúčovou funkciou pre kontrolu kolízií je funkcia `get_event_collisions` implementovaná v module `mtimetables.timetable.functions`, ktorá pre danú udalosť vracia zoznam kolíznych udalostí. Jej správanie sa sme overili pomocou dvanástich jednotkových testov (angl. *unit tests*):

1. `test_onetimeevents_room`
2. `test_onetimeevents_user`
3. `test_onetimeevents_group`
4. `test_onetimeevents_activity`
5. `test_semesterevents_room`
6. `test_semesterevents_user`
7. `test_semesterevents_group`
8. `test_semesterevents_activity`
9. `test_bothevents_room`
10. `test_bothevents_user`
11. `test_bothevents_group`
12. `test_bothevents_activity`

V každom teste sme vytvorili udalosti distribuované v čase podľa nasledujúcich obrázkov (viď obrázky 12, 13 a 14). Každý test nastavoval pre svoje udalosti iný spoločný parameter – miestnosť (testy 1, 5, 9), používateľa (testy 2, 6, 10), skupinu (testy 3, 7, 11) alebo definíciu aktivity predmetu (testy 4, 8, 12).

	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00
Pondelok 17.2.2014					A							
	B		C	D	E				F	G		H
				X					Y			

Obr. 12: Distribúcia udalostí v čase pre testy 1 – 4

	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00
Pondelok 17.2. - 5.5. 2014					A							
	B		C	D	E				F	G		H
				X					Y			

Obr. 13: Distribúcia udalostí v čase pre testy 5 – 8

	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00
Pondelok 17.2.2014					A1							
	B1		C1	D1	E1				F1	G1		H1
				X1					Y1			
Pondelok 17.2. - 5.5. 2014					A2							
	B2		C2	D2	E2				F2	G2		H2
				X2					Y2			
Utorok 18.2. - 6.5. 2014					A3							
	B3		C3	D3	E3				F3	G3		H3
				X3					Y3			

Obr. 14: Distribúcia udalostí v čase pre testy 9 – 12

Následne každý test postupne spustil funkciu `get_event_collisions` pre každú udalosť a jej návratovú hodnotu porovnal s očakávanou hodnotou. Test je úspešný práve vtedy, keď pre všetky jeho udalosti platí, že množina kolíznych udalostí získaná funkciou `get_event_collisions` je zhodná s očakávanou množinou kolíznych udalostí. Očakávané množiny kolíznych udalostí pre jednotlivé udalosti testov sú uvedené na v tabuľkách 2 a 3.



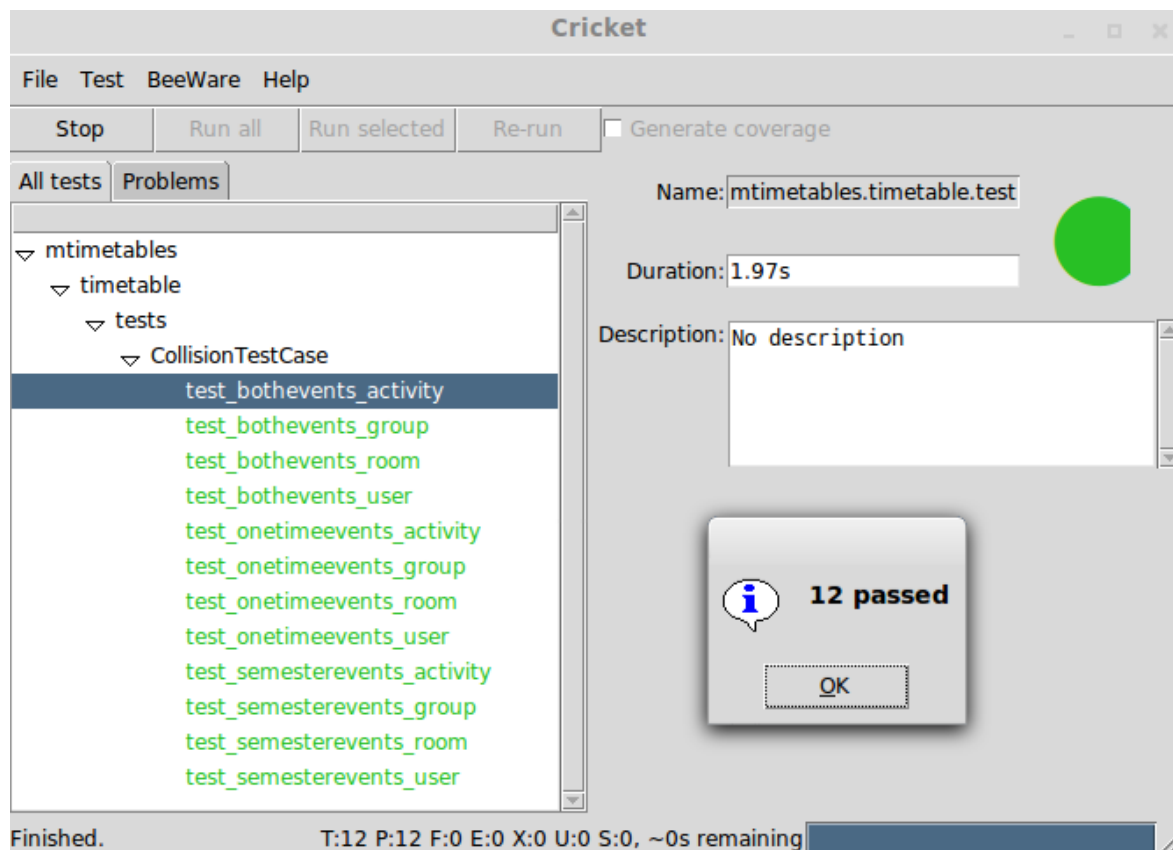
Udalosť	Množina kolíznych udalostí	Udalosť	Množina kolíznych udalostí
A	{D, E, F, X, Y}	F	{A, Y}
B	{}	G	{Y}
C	{X}	H	{}
D	{A, X}	X	{A, C, D}
E	{A}	Y	{A, F, G}

Tab. 2: Množiny kolíznych udalostí pre testy 1 – 8

Udalosť	Množina kolíznych udalostí	Udalosť	Množina kolíznych udalostí	Udalosť	Množina kolíznych udalostí
A1	{A2, D1, D2, E1, E2, F1, F2, X1, X2, Y1, Y2}	A2	{A1, D1, D2, E1, E2, F1, F2, X1, X2, Y1, Y2}	A3	{D3, E3, F3, X3, Y3}
B1	{B2}	B2	{B1}	B3	{}
C1	{C2, X1, X2}	C2	{C1, X1, X2}	C3	{X3}
D1	{A1, A2, D2, X1, X2}	D2	{A1, A2, D1, X1, X2}	D3	{A3, X3}
E1	{A1, A2, E2}	E2	{A1, A2, E1}	E3	{A3}
F1	{A1, A2, F2, Y1, Y2}	F2	{A1, A2, F1, Y1, Y2}	F3	{A3, Y3}
G1	{G2, Y1, Y2}	G2	{G1, Y1, Y2}	G3	{Y3}
H1	{H2}	H2	{H1}	H3	{}
X1	{A1, A2, C1, C2, D1, D2, X2}	X2	{A1, A2, C1, C2, D1, D2, X1}	X3	{A3, C3, D3}
Y1	{A1, A2, F1, F2, G1, G2, Y2}	Y2	{A1, A2, F1, F2, G1, G2, Y1}	Y3	{A3, F3, G3}

Tab. 3: Množiny kolíznych udalostí pre testy 9 – 12

Na dostupnej vzorke testovacích dát všetky testy prebehli bez chýb. Na obrázku 15 sa nachádza grafický výstup programu Cricket verzie 0.2.3 [5], pomocou ktorého sme testy spúšťali.



Obr. 15: Testovanie funkcie pre kontrolu kolízií – výstup z programu Cricket 0.2.3

## 7 ZHODNOTENIE

Vo fáze analýzy problémovej oblasti sme identifikovali výrazné nedostatky súčasných rozvrhových systémov (napr. slabé pokrytie typov požiadaviek na rozvrh, chýbajúce alebo nedostatočné rozhranie na prepojenie s UIS, sústredenie výpočtovej sily na osobný počítač, atď.). Zároveň sme objavili niektoré dôležité vlastnosti, ktoré neobsahovala špecifikácia rozvrhového systému tímového projektu, z ktorého sme vychádzali [8].

Existujúcu špecifikáciu rozvrhového systému sme rozšírili o ďalšie prípady použitia (rozvrh na semester aj skúškové obdobie, zobrazenie kolíznych matíc, rezervácia miestností a konfigurácia systému) a niektoré typy požiadaviek na rozvrh (najmä špeciálne požiadavky týkajúce sa rozvrhu skúškového obdobia). Nová komplexná špecifikácia pokrýva množstvo rôznorodých požiadaviek rozvrhárov, vyučujúcich aj študentov.

Vzhľadom na vážne nedostatky návrhu z tímového projektu a nové skutočnosti v špecifikácii aplikácie, bolo potrebné celý návrh výrazne modifikovať. Architektúru systému sme rozdelili na niekoľko samostatných častí, ktoré medzi sebou komunikujú cez definované rozhrania. Zamerali sme sa na podrobný návrh modulov tvoriacich jadro systému, ktoré sme v ďalšej časti práce implementovali. Pripravili sme rozhranie pre implementáciu rôznych typov požiadaviek na rozvrh a algoritmov pre inteligentné generovanie rozvrhových akcií. Modularita vytvoreného návrhu poskytuje priestor aj pre prípadnú potrebu ďalšieho rozšírenia systému.

Jadro rozvrhového systému sme implementovali formou webovej aplikácie s využitím moderných technológií ako Python, Django, PostgreSQL, Bootstrap, jQuery, atď. Vytvorili sme moduly pre import a úpravu vstupných dát, manuálnu tvorbu rozvrhových akcií s kontrolou kolízií, rôzne typy zobrazenia priebežnej aj výslednej verzie rozvrhu. Chyby zistené počas testovania aplikácie sme následne odstránili.

Štruktúru zdrojových CSV súborov pre import dát FEI STU (zoznamy predmetov, ich učiteľov a študentov, miestností a ich vybavenia), sme odkomunikovali s hlavnou

integrátorkou AIS STU, Ing. Andreou Bujdakovou. Vytvorený importovací modul môže byť využitý aj inou fakultou/univerzitou (pri dodržaní špecifikovaného formátu vstupných dát), prípadne je možné modifikovať jeho implementáciu podľa vlastných potrieb.

Hlavné možnosti ďalšieho rozvíjania systému (zoradené podľa dôležitosti):

- grafické rozhranie pre import zdrojových dát a zmenu nastavení systému,
- export hotového rozvrhu do AIS,
- reálne zobrazenie kalendára rozvrhových akcií vzhľadom na definované dni voľna a náhradu výuky,
- spríjemnenie grafického používateľského rozhrania (napr. vytvorenie ďalších filtrov, implementácia vyhľadávania, atď.),
- zálohovanie rozvrhu,
- návrh a implementácia jednotlivých modulov typov požiadaviek uvedených v špecifikácii (viď kapitola 3.3),
- zastupovanie chýbajúcich,
- systém používateľských rolí a práv (vrátane možnosti delegovania oprávnení používateľa),
- autentifikácia prepojená s UIS,
- navrhovanie zmien rozvrhu,
- rôzne algoritmy generovania rozvrhu,
- podpora nekonfliktnej kolaborácie viacerých rozvrhárov na tom istom rozvrhu,
- lokalizácia do slovenčiny.

## 8 ZÁVER

V tejto práci sme sa postupne zaoberali analýzou troch profesionálnych nástrojov na tvorbu vysokoškolských rozvrhov, zberom a špecifikáciou požiadaviek nového rozvrhového systému, jeho návrhom a implementáciou.

Pri analýze existujúcich riešení boli identifikované dva druhy koncepcie rozvrhových systémov. Prvý typ síce zastrešuje tvorbu rozvrhov pre široké spektrum škôl (základné, stredné, vysoké školy a univerzity), absentuje však individuálny prístup k potrebám jednotlivých škôl. Druhý typ koncepcie sa podrobnejšie sústreďí na konkrétny problém (napr. automatický generátor rozvrhových akcií), pričom zaostáva v množstve a kvalite ostatných dôležitých funkcií. Zistili sme, že existujúce softvérové nástroje dostatočne nepokrývajú potreby tvorby rozvrhov mnohých vysokých škôl (napr. aj FEI STU).

Veľkým prínosom tejto práce je vytvorená webová aplikácia obsahujúca najdôležitejšie funkcie potrebné pre pohodlnú tvorbu vysokoškolských rozvrhov na semester aj skúškové obdobie. Implementovaný modul pre import vstupných dát vyvinutý špeciálne pre potreby FEI STU je možné v budúcnosti jednoducho modifikovať pre potreby inej fakulty či celej univerzity. Súčasťou práce je rozsiahla špecifikácia konkrétnych požiadaviek nového systému, modulárny návrh jeho architektúry a rozhranie umožňujúce v budúcnosti ďalšie rozšírenie funkcionality vytvoreného rozvrhového systému.



## 9 LITERATÚRA

- [1] ALSUP, M. a kol., 2012. jQuery Form Plugin [online]. [cit. 28. marec 2014]. Dostupné z: <http://jquery.malsup.com/form/>
- [2] BORGSTROM, E., 2014. django-simple-menu Documentation (v. 1.0.9) [online]. [cit. 1. máj 2014]. Dostupné z: <https://media.readthedocs.org/pdf/django-simple-menu/latest/django-simple-menu.pdf>
- [3] GÉLINAS, F., 2014. jQuery UI Timepicker [online]. [cit. 28. marec 2014]. Dostupné z: <https://fgelinas.com/code/timepicker/>
- [4] Jun, H. a kol., 2014. Perfect Scrollbar [online]. [cit. 28. marec 2014]. Dostupné z: <https://github.com/noraesae/perfect-scrollbar/>
- [5] KEITH-MAGEE, R., 2014. Cricket [online]. [cit. 18. máj 2014]. Dostupné z: <http://cricket.readthedocs.org/en/v0.2.3/>
- [6] Link Checker [online]. [cit. 30. apríl 2014]. Dostupné z: <http://wummel.github.io/linkchecker/>
- [7] KNAPERREKOVÁ, E., 2011. Aplikácia na personalizáciu rozvrhu: bakalárska práca. Bratislava: STU.
- [8] KNAPERREKOVÁ, E., M. KORDÍK, J. FIRÁK, P. VILÁGI a J. BUČKULIAK, 2012. Rozvrhový systém pre FEI STU: tímový projekt. Bratislava: STU.
- [9] LALESCU, L., 2013. Free Timetabling Software [online]. [cit. 21. január 2013]. Dostupné z: <http://www.lalescu.ro/liviu/fet/>
- [10] LEE, M., 2013. jQuery Colorpicker [online]. [cit. 28. marec 2014]. Dostupné z: <http://vanderlee.github.io/colorpicker/>
- [11] OTTO, M., J. THORNTON a kol., 2014. Bootstrap [online]. [cit. 2. marec 2014]. Dostupné z: <http://getbootstrap.com/>
- [12] RESIG, J. a kol., . jQuery [online]. [cit. 20. marec 2014]. Dostupné z: <http://jquery.com/>
- [13] STIGER, C., 2014. django-mptt Documentation (v. 0.6.0) [online]. [cit. 18. máj 2014].

Dostupné z: <https://media.readthedocs.org/pdf/django-mptt/latest/django-mptt.pdf>

[14] TRIER, M. a B. OOSTVEEN, 2014. django-extensions Documentation (v. 1.3.7) [online]. [cit. 18. máj 2014]. Dostupné z: <https://media.readthedocs.org/pdf/django-extensions/latest/django-extensions.pdf>

[15] VERHEUL, D., 2014. django-bootstrap3 Documentation (v. 3.2.0) [online]. [cit. 1. máj 2014]. Dostupné z: <https://media.readthedocs.org/pdf/django-bootstrap3/latest/django-bootstrap3.pdf>

[16] django-bitfield [online]. [cit. 27. apríl 2014]. Dostupné z: <https://github.com/disqus/django-bitfield>

[17] jQuery UI [online]. [cit. 28. marec 2014]. Dostupné z: <https://jqueryui.com/>

[18] PostgreSQL [online]. [cit. 4. máj 2014]. Dostupné z: <http://www.postgresql.org/>

[19] Roger [online]. [cit. 13. január 2013]. Dostupné z: <http://www.time-tables.com/sk>

[20] The Django framework [online]. [cit. 14. február 2014]. Dostupné z: <https://www.djangoproject.com/>

[21] Wise Timetable [online]. [cit. 26. január 2013]. Dostupné z: <http://www.wisetimetable.com/>

[22] Django Debug Toolbar Documentation (v. 1.2.1) [online]. [cit. 18. máj 2014]. Dostupné z: <https://media.readthedocs.org/pdf/django-debug-toolbar/latest/django-debug-toolbar.pdf>

[23] Python [online]. [cit. 2. február 2014]. Dostupné z: <https://www.python.org/>



# PRÍLOHA A:

## TECHNICKÁ DOKUMENTÁCIA

Tento dokument je prílohou k diplomovej práci „Rozvrhový systém pre vysoké školy“. Obsahuje požiadavky na hardvér a softvér pre spustenie vytvorenej webovej aplikácie, postup inštalácie a konfigurácie rozvrhového systému a špecifikáciu formátu zdrojových dát pre modul *FEI*.

### A.1 Požiadavky na hardvér a softvér

Pre bezproblémovú prevádzku implementovaného rozvrhového systému úplne postačuje 512 MB hlavnej pamäte, jedno-jadrový procesor rodiny x86, 20 GB diskového priestoru, sieťové pripojenie.

Softvérové nástroje a príkazy uvedené v tejto dokumentácii sú pre operačný systém Debian GNU/Linux (verzia Wheezy), na ktorom sme aplikáciu testovali.

Pomocou nasledovných príkazov (spúšťať ako *root*) je možné nainštalovať potrebné softvérové balíky z repozitáru operačného systému:

```
# apt-get install python-pip postgresql python-psycopg2 apache2  
libapache2-mod-wsgi  
# pip install virtualenv
```

Pre spustenie webovej aplikácie sa odporúča použiť najnovšiu verziu niektorého z moderných internetových prehliadačov (napr. Mozilla Firefox, Google Chrome) so zapnutou podporou súborov *cookies* a skriptov jazyka JavaScript.

### A.2 Inštalácia rozvrhového systému

Táto kapitola obsahuje odporúčaný spôsob inštalácie produkčnej verzie aplikácie na operačnom systéme Debian Wheezy.

### **Krok 1: koreňový adresár**

Vytvorte koreňový adresár celej webovej aplikácie (napr. */home/mtimetables/production/* ďalej len *HOME*), prekopírujte doňho súbor *mtimetables.zip* (z priloženého DVD nosiča) a rozbaľte.

### **Krok 2: inštalácia a aktivácia virtuálneho prostredia**

V adresári *HOME* spustíte príkazy:

```
$ virtualenv venv
$ . venv/bin/activate
(venv)$ pip install -r mtimetables/project/requirements/prod.txt
```

### **Krok 3: základná konfigurácia Django projektu**

V adresári *HOME/mtimetables/project/src/mtimetables\_project/settings* premenujte súbor *local\_default.py* na *local.py* a nastavte

1. údaje potrebné na pripojenie k databáze,
2. hodnotu *DEBUG* na *False*,
3. hodnotu *SECRET\_KEY* na reťazec aspoň 30 náhodne vygenerovaných znakov (ktoré si netreba pamätať),
4. hodnotu *STATIC\_ROOT* na reťazec obsahujúci absolútnu cestu k adresáru *HOME/collected\_static*, ktorý bude vytvorený v kroku 4,
5. hodnotu *MEDIA\_ROOT* na reťazec obsahujúci absolútnu cestu k adresáru *HOME/media*, ktorý bude vytvorený v kroku 5.

### **Krok 4: statické súbory**

V adresári *HOME* vytvorte nový adresár s názvom *collected\_static* a spustíte príkazy:

```
(venv)$ cd mtimetables/project/src
(venv)$ ./manage.py collectstatic
```

### **Krok 5: médiá**

V adresári *HOME* vytvorte nový adresár s názvom *media*.

## Krok 6: konfigurácia webového servera\*

Upravte súbor `/etc/apache2/sites-available/mtimetables.conf` podľa nasledovného vzoru:

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName mtimetables.mwebair.info
    ServerAlias www.mtimetables.mwebair.info

    Alias /static HOME/collected_static
    Alias /media HOME/media
    WSGIScriptAlias / HOME/mtimetables/src/mtimetables_project/wsgi.py
    WSGIDaemonProcess mtimetables user=mtimetables group=mtimetables
    threads=2 python-path=HOME/venv/lib/python2.7/site-packages
    WSGIProcessGroup mtimetables

    <Directory HOME/mtimetables/src/mtimetables_project>
        <Files wsgi.py>
            Order allow, deny
            Allow from all
        </Files>
    </Directory>

    <Directory HOME/collected_static>
        Order allow, deny
        Allow from all
    </Directory>
</VirtualHost>
```

Spustením nasledovných príkazov uložte konfiguráciu a reštartujte webový server:

```
# a2ensite mtimetables.conf
# service apache2 restart
```

## Krok 7: databáza

Vytvorte databázu pomocou príkazu:

```
(venv)$ createdb mtimetables
```

---

\* Tento krok možno vynechať, ak chcete aplikáciu spúšťať na lokálnom počítači (napr. na vývojárske účely). Štandardná inštalácia *framework-u* Django obsahuje server pre vývojárov, preto nie je potrebné osobitne konfigurovať webový server Apache. Django server na adrese 127.0.0.1 a porte 8000 spustíte pomocou príkazu z adresára `HOME/mtimetables/project/src`: `./manage.py runserver --insecure 8000`.

Pre synchronizáciu databázy spustíte z adresára *HOME/mtimetables/project/src* príkaz:

```
(venv)$ ./manage.py syncdb
```

### **Krok 8: import dát**

V adresári *HOME* vytvorte nový adresár *data* a prekopírujte doňho zdrojové CSV súbory. Z adresára *HOME/mtimetables/project/src* spustíte príkaz:

```
(venv)$ ./manage.py fei_import HOME/data/*
```

Podrobný popis formátu zdrojových dát sa nachádza v kapitole A.4.

### **Krok 9: konfigurácia rozvrhového systému**

V súbore *HOME/mtimetables/project/src/mtimetables/settings/settings.py* nastavte parametre rozvrhového systému (viď kapitola A.3 Nastavenia rozvrhového systému).

### **Krok 10: vytvorenie používateľského konta**

Z adresára *HOME/mtimetables/project/src* spustíte nasledovný príkaz a riad'te sa pokynmi na obrazovke – vyplňte prihlasovacie meno, email (voliteľne) a heslo:

```
(venv)$ ./manage.py createsuperuser
```

## **A.3 Nastavenia rozvrhového systému**

V súčasnej verzii rozvrhového systému je možné meniť konfiguráciu v súbore *settings.py* v adresári aplikácie *settings* (*HOME/mtimetables/project/src/mtimetables/settings*).

### **A.3.1 Databázové konštanty**

ABBREVIATION\_LENGTH – max. dĺžka skratiek.

NAME\_LENGTH – max. dĺžka názvov.

PASSWORD\_LENGTH – max. dĺžka hesla.

UIS\_ID\_LENGTH – max. dĺžka ID importovaných dát z UIS.

### **A.3.2 Semester a skúškové**

WEEKDAYS – dni víkendu.

SEMESTER\_START\_DATE – začiatok semestra.

SEMESTER\_END\_DATE – koniec semestra.

SEMESTER\_WEEKS\_COUNT – počet týždňov semestra.

SEMESTER\_WEEK\_DAYS\_COUNT – počet dní výuky v týždni.

WEEK\_STARTS\_AT – prvý deň v týždni.

EXAMINATION\_PERIOD\_START\_DATE – začiatok skúškového.

EXAMINATION\_PERIOD\_END\_DATE – koniec skúškového.

CALENDAR\_START\_DATE – prvý deň rozsahu kalendára.

CALENDAR\_END\_DATE – posledný deň rozsahu kalendára.

### **A.3.3 Všeobecné nastavenia**

DEFAULT\_PRIORITY – predvolená hodnota pre prioritu.

### **A.3.4 Nastavenia aplikácie „data“**

DEFAULT\_USER\_SUBJECT\_RELATION\_RATE – predvolená hodnota koeficientu príslušnosti používateľa k predmetu.

BACHELOR\_ROOT\_GROUP\_ID – id skupiny, ktorá je koreňom stromu bakalárskych študijných programov a zameraní.

ENGINEER\_ROOT\_GROUP\_ID – id skupiny, ktorá je koreňom stromu inžinierskych študijných programov a zameraní.

USER\_SUBJECT\_RELATION\_TYPES – typy úloh používateľov na predmetoch (prednášajúci, garant, študent, atď.).

USER\_EVENT\_RELATION\_TYPES – typy úloh používateľov na rozvrhových akciách (prednášajúci, garant, študent, atď.).

USER\_DEPARTMENT\_RELATION\_TYPES – typy úväzkov používateľov na pracoviskách.

ACTIVITY\_PROTOTYPES – zoznam prototypov aktivít (prednáška, cvičenie, skúška).

USER\_GROUP\_STUDY\_METHODS – formy štúdia študijných programov (prezenčná, dištančná, kombinovaná).

### **A.3.5 Nastavenia aplikácie „timetable“**

DEFAULT\_COLORS – predvolené kódy farieb pre zobrazenie prednášok, cvičení a skúšok.

### **A.3.6 Nastavenia aplikácie „requirements“**

DEFAULT\_ALLOWED\_RT\_COUNT – predvolená hodnota pre max. povolený počet požiadaviek objektu.

DEFAULT\_EVALUATION\_METHOD – predvolená metóda vyhodnocovania požiadavky.

REQUIREMENT\_PACKAGE\_TYPES – typy balíkov požiadaviek.

### **A.3.7 Nastavenia aplikácie „calendar“**

DEFAULT\_SEMESTER\_TIMETABLE\_GRID – id predvolenej časovej mriežky pre zobrazenie rozvrhu na semester.

DEFAULT\_EXAMINATION\_PERIOD\_TIMETABLE\_GRID – id predvolenej časovej mriežky pre zobrazenie rozvrhu na skúškové obdobie.

### **A.3.8 Nastavenia aplikácie „fei“**

FEI\_OMIT\_SUBJECTS\_REGEXES – regulárne výrazy názvov predmetov, ktoré majú byť z importu vynechané

SUBJECT\_EXAM\_COMPLETION\_MODES – typy ukončenia predmetov, pre ktoré sa vytvárajú skúškové termíny

DEFAULT\_[LECTURE,EXERCISE,EXAM]\_MANDATORY\_INSTANCES\_COUNT – predvolená hodnota počtu povinných účastí študenta na prednáške (za týždeň), cvičení (za týždeň), skúške (celkový počet)

DEFAULT\_[LECTURE,EXERCISE,EXAM]\_ACTIVITY\_TYPES – id typov aktivít prototypu [prednáška, cvičenie, skúška].

DEFAULT\_ROOM\_CAPACITY\_RATE – predvolená hodnota koeficientu pre výpočet kapacity miestnosti.

## A.4 Formát zdrojových dát pre modul „FEI“

Modul *FEI* akceptuje zdrojové dáta vo formáte CSV – stĺpce oddelené bodkočiarkou a riadky znakom nového riadku. Štruktúra jednotlivých súborov je nasledovná:

**departments.csv** (pracoviská):

- id
- skratka (napr. ÚIM)
- názov (napr. Ústav informatiky a matematiky)
- id nadradeného oddelenia (voliteľne – ak má nadradené pracovisko)

**equipments.csv** (pomôcky):

- id
- názov (napr. projektor, tabuľa, umývadlo, ...)

**roomtypes.csv** (typy miestností):

- id
- názov (napr. učebňa, počítačová učebňa, laboratórium, ...)

**rooms.csv** (miestnosti):

- id
- názov (napr. BC300)
- kapacita (napr. 300)
- id typu miestnosti (id z *roomtypes.csv*)
- id pracoviska (voliteľne – ak miestnosť patrí nejakému pracovisku, id zo súboru *departments.csv*)

**rooms\_equipments.csv** (vybavenie miestností):

- id miestnosti (id z *rooms.csv*)
- id pomôcky (id z *equipments.csv*)
- počet

**groups.csv** (skupiny):

- id
- názov
- skratka (pre vnorené skupiny generovať aj z ich rodičov, napr. I-API-MaSUS-1)
- id nadradenej skupiny (voliteľne – ak má nadradenú skupinu)

Skupiny budú hierarchické, pričom každý používateľ môže patriť do viacerých skupín (môže byť súčasne študent aj vyučujúci – najmä pri doktorandskom type štúdia, prípadne študent môže študovať súčasne viacero študijných programov). Príklad hierarchie skupín (názov, skratka):

- učitelia (vyučujúci, NULL)  
- študenti (študent, NULL)  
- - - typ štúdia (inžiniersky, I)  
- - - - študijný program (Aplikovaná informatika, I-API)  
- - - - - zameranie (Modelovanie a simulácia udalostných systémov, I-API-MaSUS)  
- - - - - - ročník (1, I-API-MaSUS-1)

Poznámka: vytvárať iba také skupiny, ktoré obsahujú nejakých používateľov => zameranie môže byť v niektorých (hlavne v bakalárskych) vetvách vynechané.

**users.csv** (používatelia):

- id
- prihlasovacie meno do AIS
- meno
- priezvisko
- tituly pred menom (vyskladať reťazec z AIS)
- tituly za menom (vyskladať reťazec z AIS)

**users\_groups.csv** (zaradenie používateľov do skupín):

- id používateľa (id zo súboru *users.csv*)
- id skupiny (id zo súboru *groups.csv*)
- číslo krúžku (voliteľne, hlavne pre študentov – prvého)



- metóda štúdia (voliteľne, len pre študentov, 1 – prezenčná, 2 – dištančná, 3 – kombinovaná)

**users\_departments.csv** (priradenie pracovísk používateľom):

- id používateľa (id zo súboru *users.csv*)
- id pracoviska (id zo súboru *departments.csv*)
- typ pracovného pomeru (1 – interný, 2 – externý, 3 – doktorand)

**studytypes.csv** (typy štúdia)\*:

- id
- názov

**subjects.csv** (predmety):

- id
- kód
- názov
- id garantujúceho pracoviska (id zo súboru *departments.csv*)
- typ ukončenia predmetu (z – zápočet, s – skúška, kz – klasifikovaný zápočet, Šsk – štátna skúška)

**subjects\_users.csv** (priradenie predmetov používateľom):

- id predmetu (id zo súboru *subjects.csv*)
- id používateľa (id zo súboru *users.csv*)
- úloha na predmete (1 – študent, 2 – garant, 3 – prednášajúci, 4 – cvičiaci, 5 – skúšajúci, 6 – administrátor, 7 – tútor)

**subjects\_studytypes.csv** (definovanie parametrov výučby predmetov):

- číslo formy výučby (1 – prezenčná, 2 – dištančná)
- rozsah výučby vo formáte X/Y

X vo formáte x1[+x2[+x3[...]]] je rozdelenie hodinovej dotácie prednášok

Y vo formáte y1[+y2[+y3[...]]] je rozdelenie hodinovej dotácie cvičení

---

\* Pre FEI bude obsahovať dva záznamy – prezenčná (1) a dištančná (2).



# PRÍLOHA B:

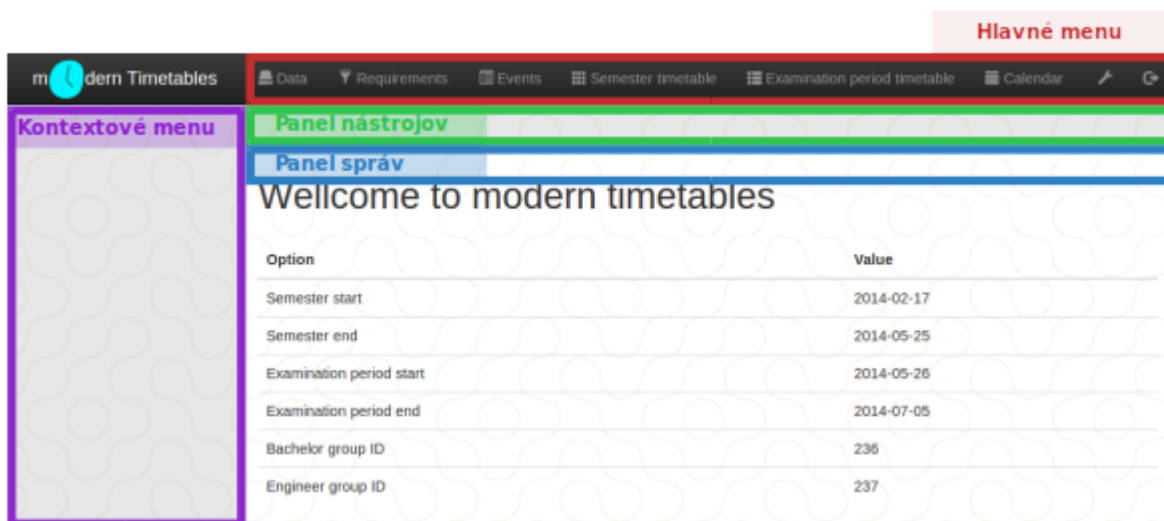
## POUŽÍVATEĽSKÁ PRÍRUČKA

Tento dokument je prílohou k diplomovej práci „Rozvrhový systém pre vysoké školy“. Obsahuje popis základných častí grafického používateľského rozhrania a vybrané scenáre použitia.

**Upozornenie:** pre korektné zobrazenie aplikácie použite najnovšiu verziu niektorého z moderných internetových prehliadačov (napr. Mozilla Firefox, Google Chrome) so zapnutou podporou súborov *cookies* a skriptov jazyka JavaScript.

### B.1 Časti grafického používateľského rozhrania

Po úspešnom prihlásení sa do rozvrhového systému je zobrazená úvodná stránka (viď obrázok 16) so zoznamom hodnôt najdôležitejších konfiguračných parametrov.



Obr. 16: Hlavné časti grafického používateľského rozhrania

V hornej časti stránky sa nachádza **hlavné menu** s nasledovnými položkami:

- *Data* – úprava zoznamov pracovísk (*Departments*), skupín používateľov (*Groups*), učiteľov a študentov (*Users*), študijných typov (*Study types*), definícií aktivít

(*Activity definitions*), miestností (*Rooms*), typov miestností (*Room types*) a pomôcok (*Equipments*);

- *Requirements* – vytváranie požiadaviek na rozvrh pre všetky zainteresované objekty (učiteľov, študentov a ich skupiny, predmety, definície aktivít, miestnosti a ich typy) a ich zoskupovanie do balíkov pre priradenie rovnakých požiadaviek viacerým objektom;
- *Events* – zobrazenie a vytváranie rozvrhových akcií cez formulár (*Semester events* – opakované udalosti počas semestra, *One time events* – jednorázové udalosti – skúšky alebo mimoriadne jednorázové udalosti počas semestra), zobrazenie zoznamu kolíznych rozvrhových akcií;
- *Semester timetable* – úprava rozvrhu na semester;
- *Examination period timetable* – úprava rozvrhu skúšok;
- *Calendar* – zobrazenie aktuálnej verzie rozvrhu (bez možnosti úpravy);
- ikona *Settings* – nastavenia systému – vytvorenie časových mriežok s rôznymi časovými intervalmi, vytvorenie typov aktivít, inštalácia typov požiadaviek a ich nastavenia;
- ikona *Logout* – odhlásenie sa zo systému.

Pod hlavným menu sa nachádza **panel nástrojov** s tlačidlami akcií aktívnej stránky (napr. pridať objekt, nasledujúca/predchádzajúca strana zoznamu objektov, ísť na zoznam objektov, ísť na požiadavky objektu, atď.).

**Panel správ** zobrazuje zoznam systémových správ – ak sú – (napr. informácia o úspešnom uložení alebo vytvorení objektu, chybové hlásenie).

V ľavej časti stránky sa nachádza priestor pre **kontextové menu**, ktorý je využívaný najmä pre zobrazenie zoznamu vnorených stránok aktívnej položky hlavného menu.

## B.2 Nastavenie časovej mriežky

Pre vytvorenie novej časovej mriežky (alebo úpravu existujúcej) zvolíte *Settings -> Hours -> Add new timetable grid* (alebo *Edit timetable grid*). Zobrazí sa formulár pre úpravu atribútov časovej mriežky (viď obrázok 17). Vyplňte názov mriežky a zvolíte jej typ

(semester alebo *examination period* – skúškové obdobie). Upravte názvy a časové intervaly jednotlivých hodín. Zvoľte *Add hour* pre pridanie novej hodiny. Pre uloženie vykonaných zmien zvoľte *Save*.

Obr. 17: Úprava časovej mriežky na skúškové obdobie

## B.3 Možnosti zobrazenia rozvrhu

V sekciách *Semester timetable*, *Examination period timetable* a *Calendar* je možné použiť tri typy zobrazenia rozvrhu – mesiac (angl. *Month*), týždeň (angl. *Week*) alebo deň (angl. *Day*). Rozvrhové akcie vo zvolenom type zobrazenia je možné filtrovať podľa miestnosti (angl. *Room*), skupiny používateľov (angl. *Group*), konkrétneho používateľa (angl. *User*) a predmetu (angl. *Subject*). Rozvrhové akcie sa zobrazia v tabuľke zarovnané podľa časových intervalov zvolenej mriežky (angl. *Hour set*).

Na obrázku 18 sa nachádza ukážka vyplneného formulára pre filtrovanie rozvrhových akcií podľa uvedených parametrov. Povinné je vyplnenie položky *Hour set* a práve jednej z položiek *Month*, *Week* alebo *Day*. Nastavenie položiek *Room*, *Group*, *User* a *Subject* je voliteľné, môžu byť vyplnené aj všetky súčasne. Na obrázkoch 19, 20 a 21 sa nachádzajú ukážky zobrazenia rozvrhu podľa zvolenej položky *Day*, *Week* alebo *Month*. Pre mesačné a týždenné zobrazenie je možné zapnúť/vypnúť zobrazenie dní víkendu.

Toggle calendar filter ?

Hour set ★ skúškové ▼

Room bc300 ▼

Group ----- ▼

User ----- ▼

Subject Workflow manažment systémy ▼

Month ----- ▼

Week 16: Jun 02, 2014 - Jun 08, 2014 ▼

Day YYYY-MM-DD

Apply filter

Obr. 18: Filter rozvrhových akcií

	Wednesday, Jun. 04
1. beh (07:30 - 10:00)	Analýza a zložitosť algoritmov - skúška 1. termín
2. beh (10:30 - 13:00)	Architektúra počítačov - skúška 1. termín
3. beh (13:30 - 16:00)	Matematika 1 - skúška 2. termín
4. beh (16:30 - 19:00)	Matematika 1 - skúška 2. termín

Obr. 19: Zobrazenie rozvrhu – deň (angl. Day)

	Monday, Jun. 02	Tuesday, Jun. 03	Wednesday, Jun. 04	Thursday, Jun. 05	Friday, Jun. 06
1. beh (07:30 - 10:00)			Analýza a zložitosť ...	Klasické šifry - skúš...	Senát
2. beh (10:30 - 13:00)		Konferencia	Architektúra počítač...		Senát Fyzika - skúška 1. t...
3. beh (13:30 - 16:00)		Konferencia	Matematika 1 - skúš...	Komunikačné siete ...	Senát
4. beh (16:30 - 19:00)			Matematika 1 - skúš...	Komunikačné siete ...	

Obr. 20: Zobrazenie rozvrhu – týždeň (angl. Week)

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

Obr. 21: Zobrazenie rozvrhu – mesiac (angl. Month)

## B.4 Postup tvorby rozvrhu

Predpokladom uvedeného postupu tvorby rozvrhu sú importované a upravené zoznamy predmetov a ich učiteľov a študentov, miestností a ich vybavenia, prípadne požiadaviek na rozvrh:

1. V hlavnom menu zvolíte položku *Semester timetable* alebo *Examination period timetable* podľa toho, či chcete upravovať rozvrh na semester alebo skúškové obdobie.
2. Nastavte filter rozvrhových akcií (z voliteľných položiek sa odporúča zvoliť aspoň položku miestnosť).
3. V ľavej časti stránky označenej ako kontextové menu (viď kapitola B.1) vyplňte filter pre zobrazenie zoznamu definícií aktivít, ku ktorým chcete vytvárať rozvrhové akcie.
4. Pre vytvorenie novej rozvrhovej akcie, pomocou metódy „*drag&drop*” presuňte definíciu aktivity z kontextového menu do požadovanej bunky v rozvrhu.

5. Existujúce rozvrhové akcie môžete premiestniť jednoducho pomocou metódy „*drag&drop*“ priamo v tabuľke s rozvrhom alebo zo zoznamu vytvorených rozvrhových akcií v kontextovom menu.

Pre úpravu podrobností existujúcej rozvrhovej akcie, kliknite na jej názov a následne sa na novej karte zobrazí príslušný formulár.

Užitočnou funkciou je označenie všetkých buniek, v ktorých sa nachádza vybraná rozvrhová akcia. Ukážte kurzorom myši na názov rozvrhovej akcie. Následne sa príslušné bunky podfarbia farbou danej rozvrhovej akcie.



# PRÍLOHA C:

## OBSAH ELEKTRONICKÉHO NOSIČA

Priložený DVD nosič obsahuje nasledovné súbory:

- *text.pdf* – text diplomovej práce,
- *technical\_documentation.pdf* – technická dokumentácia vytvoreného rozvrhového systému (požiadavky na hardvér a softvér, postup inštalácie a konfigurácie),
- *user\_manual.pdf* – používateľská príručka vytvorenej aplikácie,
- *mtimetables.zip* – zdrojové kódy implementovaného jadra rozvrhového systému,
- *activity\_diagram.svg* – UML diagram aktivít postupu tvorby rozvrhu (viď kapitola 3.1),
- *class\_diagram.svg* – UML diagram tried implementovaného rozvrhového systému (viď kapitoly 4.2, 4.3, 4.4 a 4.5),
- *package\_diagram.svg* – UML diagram balíčkov architektúry systému (viď kapitola 4.1),
- *views\_diagram.svg* – UML diagram tried dedičnosti pohľadov systému (viď kapitola 5.4.3).