SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-8328

ROZVRHOVÝ SYSTÉM PRE FEI - FRONTEND

DIPLOMOVÁ PRÁCA

2019 **Bc. Adam Kurťák**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-8328

ROZVRHOVÝ SYSTÉM PRE FEI - FRONTEND

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:	Mgr. Ing. Matúš Jókay, PhD.

Bratislava 2019 **Bc. Adam Kurťák**

Slovenská technická univerzita v Bratislave
Ústav informatiky a matematiky

Fakulta elektrotechniky a informatiky
Akademický rok: 2018/2019
Evidenčné číslo: FEI-5384-8328

STU FEI

ZADANIE DIPLOMOVEJ PRÁCE

Študent:	**Bc. Adam Kurťák**
ID študenta:	8328
Študijný program:	aplikovaná informatika
Študijný odbor:	9.2.9. aplikovaná informatika
Vedúci práce:	Mgr. Ing. Matúš Jókay, PhD.
Miesto vypracovania:	Ústav informatiky a matematiky

Názov práce: **Rozvrhový systém pre FEI – frontend**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Cieľom práce je pokračovať v implementácii rozvrhového systému pre FEI STU BA. V tejto fáze je potrebné vytvoriť klientskú aplikáciu. ktorá umožní interakciu rozvrhára so systémom.

Úlohy:
1. Naštudujte aktuálny stav návrhu a implementácie rozvrhového systému.
2. Vytýčte ciele ďalšieho rozvoja.
3. Implementujte vami navrhnuté ciele.
4. Zhodnoťte svoj prínos pre aplikáciu rozvrhového systému.

Zoznam odbornej literatúry:

1. Račák, M. – Jókay, M. *Rozvrhový systém pre FEI.* Diploma thesis. 2017. 39 s.
2. Bednár, D. – Jókay, M. *Tvorba užívateľského rozhrania k rozvrhovému systému pre FEI.* Diploma thesis. 2018. 47 s.

Riešenie zadania práce od: 17. 09. 2018

Dátum odovzdania práce: 13. 05. 2019

Bc. Adam Kurťák
študent

prof. RNDr. Otokar Grošek, PhD.
vedúci pracoviska

prof. Dr. Ing. Miloš Oravec
garant študijného programu

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Aplikovaná informatika
Autor:	Bc. Adam Kurťák
Diplomová práca:	Rozvrhový systém pre FEI - frontend
Vedúci záverečnej práce:	Mgr. Ing. Matúš Jókay, PhD.
Miesto a rok predloženia práce:	Bratislava 2019

Táto diplomová práca sa venuje problematike tvorby rozvrhového systému a jeho používateľského rozhrania. Práca analyzuje existujúce systémy na riešenie tejto problematiky, hodnotí ich výhody a nevýhody. Zaoberá sa analýzou predchádzajúcich verzií vyvíjanej aplikácie, ktoré poskytujú serverovú časť s implementovanou webovou službou a klientsku časť. Na základe týchto analýz špecifikuje požiadavky na novú klientsku časť aplikácie. Práca opisuje proces tvorby rozvrhov na FEI STU, z ktorého navrhuje nové používateľské rozhranie, v ktorom sú implementované chýbajúce funkcionality z predošlej práce. Pri implementácii kladie dôraz na funkcionalitu a používateľský zážitok z aplikácie. Výsledkom našej práce je intuitívne, prehľadné rozhranie, pomocou ktorého je možné vytvárať rozvrhy.

Kľúčové slová: angular, angular material, rozvrhový systém, JWT

ABSTRACT

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

Study Programme:	Applied Informatics
Author:	Bc. Adam Kurťák
Master's thesis:	Timetable for FEI
Supervisor:	Mgr. Ing. Matúš Jókay, PhD.
Place and year of submission:	Bratislava 2019

This thesis deals with issue of timetabling system and its user interface. Thesis analyzes existing systems for solving this problem, evaluates their advantages and disadvantages. It deals with analysis of previous versions of developed application, which provides server part with implemented web service and client part. Based on these analyzes, it specifies the requirements for new client part of application. Work describes process of creating timetables at FEI STU which proposes a new user interface in which missing functionalities from previous work were implemented. Implementation puts emphasis on the functionality and user experience of the application. Result of our work is an intuitive, easy-to-use interface for creating timetables.

Keywords: angular, angular material, timetable, JWT

Poďakovanie

Ďakujem vedúcemu práce Mgr. Ing. Matúšovi Jókayovi, PhD. za jeho odborne rady a pomoc, ktoré mi poskytol pri vypracovávaní práce. Taktiež by som sa chcel poďakovať Mgr. Dávidovi Panczomvi, PhD. a Ing. Júliusom Annusom za informácie, ktoré mi poskytli pri vypracovávaní tejto práce.

Obsah

Zoznam obrázkov a tabuliek

Zoznam skratiek

ACID	Atomicity, Consistency, Isolation, Durability
AIS	Akademický informačný systém
AJAX	Asynchronous JavaScript And XML
API	Application programming interface
CPSolver	Constraint Solver Library
CSS	Cascading Style Sheets
DI	Dependency Injection
DOM	Document Object Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IETF	Internet Engineering Task Force
IP	Internet Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
LDAP	Lightweight Directory Access Protocol
LESS	Leaner Style Sheet
PHP	PHP: Hypertext Preprocessor
REST	Representational State Transfer
SPA	Single-page application
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UX	User Experience
XML	Extensible Markup Language

Zoznam výpisov

Úvod

S nástupom moderných technológií sa mení prístup k tvorbe rozvrhov. Zatiaľ čo v minulosti sa vytvárali rozvrhy ručne bez použitia počítačov, v dnešnej dobe si to už nevieme predstaviť inak ako s použitím moderných technológií. Aj napriek technologickému pokroku ostáva vytváranie rozvrhových akcií problémom a to z dôvodu zložitosti celého procesu. Rozvrhár sa snaží zabezpečiť, aby vyhovel požiadavkám, ktoré mu boli zadané. Pri nedržaní požiadaviek dochádza v systéme ku kolíziám medzi rozvrhovými akciami. V takom prípade vyvíjané aplikácie vedia pomôcť pri vytváraní rozvrhu, či už samotným generovaním rozvrhu alebo poskytnutím funkcionalít, ktoré zjednodušujú tvorbu rozvrhu. Takéto aplikácie teda musia byť prehľadné a ľahko použitelné.

V prvej kapitole sa venujeme analýze problému pri tvorbe rozvrhov. Popisujeme ciele práce ako aj samotný proces tvorby rozvrhov. Analyzujeme existujúce systémy, ktoré problematiku riešia. V závere kapitoly sa venujeme predchádzajúcej práci a zo zistených informácií vyhodnocujeme požiadavky na nami vyvíjaný systém. V druhej kapitole sa venujeme návrhu riešenia. Definujeme používané pojmy a vysvetlíme postup pri tvorbe rozvrhov. Následne zhodnotíme ako vyzerá predchádzajúca verzia používateľského rozhrania. Opisujeme už existujúcu webovú službu a na záver opíšeme vyvíjaný systém, možnosti jeho implementácie a rozhodnutia prečo sme sa rozhodli použiť navrhované komponenty. V tretej kapitole sa venujeme použitým technológiám v navrhovanej aplikácii. Serverová časť je v krátkosti opísaná a hlbšie sa venujeme klientskej časti. Štvrtá kapitola sa venuje už samotnej implementácii. Obsahuje popis Angular architektúry, v ktorej je aplikácia implementovaná, popis štruktúry klientskej aplikácie. Podrobnejšie opísané sú hlavne poskytované moduly, ich implementácia a sú tu znázornené ukážky aplikácie. V závere kapitoly popíšeme implementované používateľské rozhranie a spôsob tvorby dokumentácie.

Práca obsahuje cudzojazyčné výrazy, ktoré kvôli strate významu nie su prekladané.

1 Analýza problematiky

V dnešnej dobe sa rozvrhy pre školy a univerzity vytvárajú výhradne pomocou na to určených programov, ktoré okrem manuálneho vytvárania rozvrhov umožňujú aj ich automatické generovanie. S postupom času už viaceré programy umožňujú vytváranie rozvrhov online za pomoci Internetu, alebo aspoň ich zdieľanie.

Problémom však aj naďalej zostáva práca pri manuálnom vytváraní rozvrhov. Používatelia aplikácie pracujú s väčším objemom dát. Pri vytváraní rozvrhu pritom musia mať prehľad o tom, kde umiestňujú rozvrhové akcie a pod. Preto je potrebné navrhnúť používateľské rozhranie tak, aby bolo jednoduché, intuitívne a prehľadné. Priveľa zobrazených informácií používateľovi môže mať rovnaký vplyv ako málo.

Samostatným problémom je aj vytváranie individuálnych požiadaviek od vyučujúcich. Takéto požiadavky sa doteraz zbierali ručne. Ich zbieranie teda sa musí dať zautomatizovať a ich prezentovanie v rozvrhu musí byť odlíšené od iných akcií.

Táto kapitola sa zaoberá procesom tvorby rozvrhových systémov a analýze existujúcich rozvrhových programov a systémov. Na základe získaných poznatkov sa vyhodnotia požiadavky na vyvíjaný systém.

1.1 Cieľ práce

Cieľom práce je oboznámiť sa s prácami [1], ktorá sa venovala primárne vytvorením backendu aplikácie, a [2], ktorého práca je popísaná v kapitole 1.4. Serverová časť aplikácie je ďalej vyvíjaná v práci kolegu Martina Makšina, s ktorou bolo potrebné sa priebežne oboznamovať. Následne podľa zistených informácií sme navrhli a implementovali webovú aplikáciu, ktorá už komunikuje s najnovšou verziou serverovej časti. Serverová aplikácia je založená na prototype popísaného v práci [3].

Práca je realizovaná aj naďalej pomocou Angular ako to bolo realizované v práci [2], avšak verzia bola aktualizovaná z verzie 5 na 7.

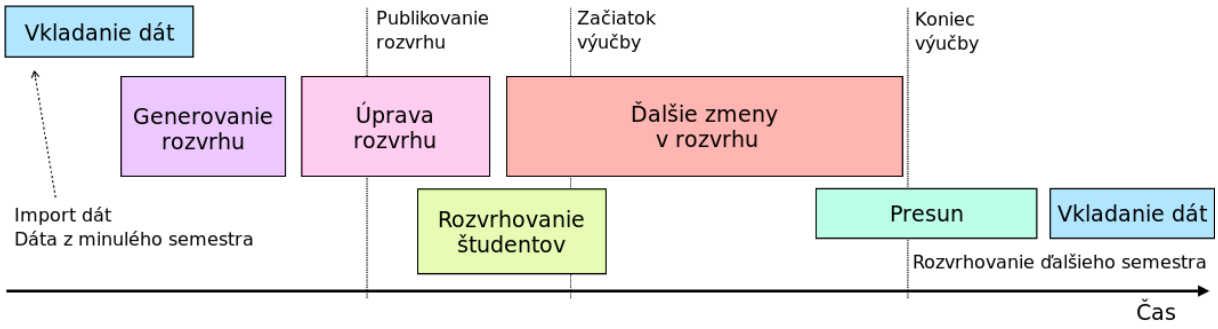
1.2 Proces tvorby rozvrhov

Aj s príchodom moderných technológií sa proces tvorby rozvrhov nezmenil. Ide o ten istý proces a jeho výsledkom je nájdenie optimálneho rozdelenia aktivít. Ide o optimalizačný problém, ktorý v sebe zahŕňa vytvorenie takého rozvrhu, ktorý podľa zvolených obmedzení vytvorí nekonfliktný rozvrh. Tieto obmedzenia následne je možné prerozdeliť podľa ich závažnosti ako tvrdé a mäkké obmedzenia [4]. Tvrdé požiadavky určujú, kedy medzi akciami nesmie dochádzať ku konfliktom, napríklad vyučujúci sa nesmie nachádzať vo viacerých rozvrhových akciách zároveň alebo dve rôzne rozvrhové akcie nesmú prebie-

hať v tej istej miestnosti a v tom istom čase. Mäkké obmedzenia opisujú také obmedzenia, ku ktorým môže dochádzať, napríklad ak vyučujúci má preferované časy ale vie vyučovať aj mimo nich.

Za proces tvorby rozvrhov na FEI STU je zodpovedná jedna osoba. Na procese tvorby sa ale môže podieľať viacero účastníkov. Ak to systém umožňuje, vyučujúci si môžu sami zadefinovať požiadavky na svoje predmety. Taktiež sa na procese tvorby podieľajú aj iné poverené osoby, ktoré môžu vytvárať rozvrhy jednotlivých ústavov.

Každý rozvrh má svoj životný cyklus, ktorý začína jeho vytvorením, importovním aktuálnych dát a končí ukončením semestra. Pri vytváraní nových rozvrhov treba zohľadniť aj to, že rozvrhové akcie z predchádzajúcich období je možné preklápať do nových období v prípade, ak sa zhodujú dáta, podľa ktorých boli vytvárané. Na obrázku č. 1.



Obr. 1: Životný cyklus rozvrhu [1]

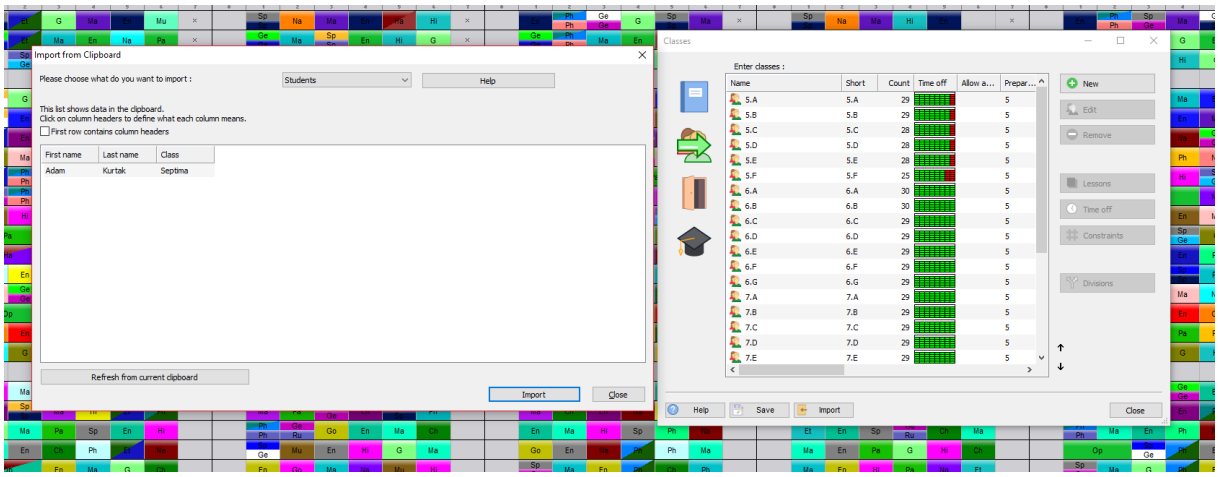
1.3 Existujúce systémy

Na trhu existujú viaceré systémy na vytváranie rozvrhov. Tie sa od seba odlišujú podľa ponúkaných funkcionalít alebo platformy, pre ktorú vyvíjané.

1.3.1 aSc TimeTables

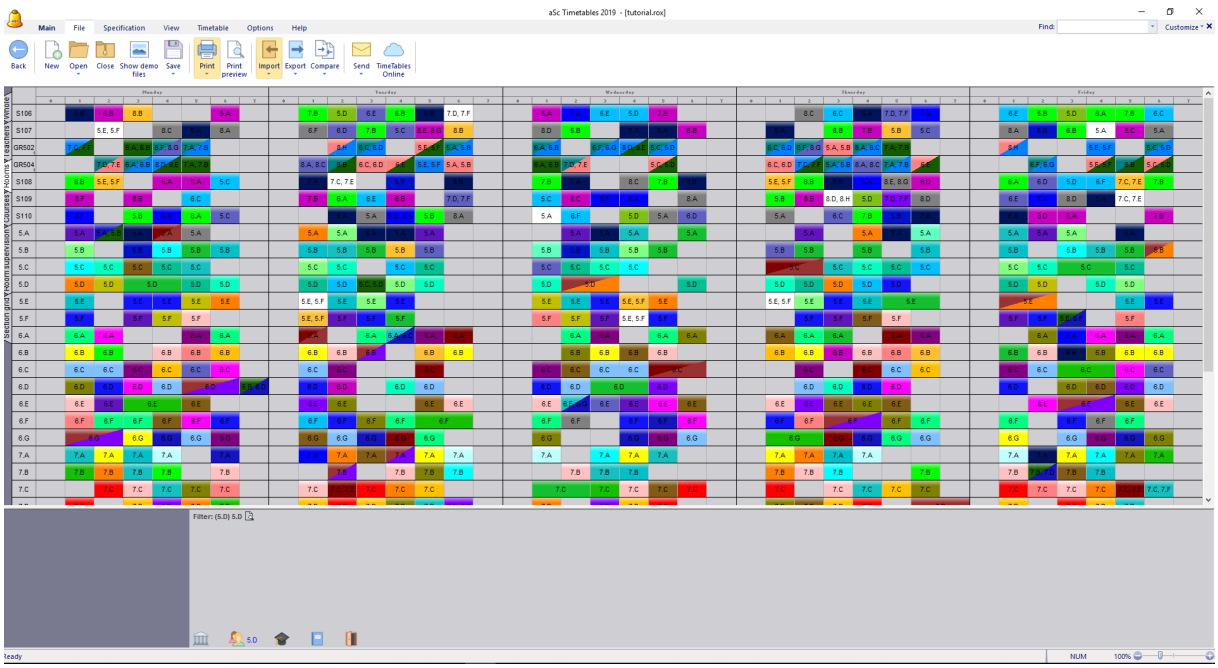
Táto kapitola sa zaoberá rozvrhovým systémom aSc TimeTables [5], pri jej tvorbe sme čerpali z vlastných skúseností s jeho používaním a oficiálnej dokumentácie. [6]

aSc TimeTables je softvér určený na vytváranie rozvrhov na základných a stredných školách. Dáta na prácu sa zadávajú manuálne, synchronizovaním priamo s databázou školy, čo je však obmedzené iba na partnerské školy, importovaním pomocou XML súboru, alebo je možné ich importovať priamo kopírovaním z MS Excel ako je zobrazené na obrázku č. 2. Program umožňuje automatické generovanie rozvrhu a následne manuálnu úpravu vygenerovaného rozvrhu. Program pracuje plynulo a generovanie rozvrhov je pomerne rýchle.



Obr. 2: aSc TimeTables import študentov [6]

Vytváranie rozvrhových akcií je riešené pomocou Drag&Drop funkcionality, avšak je nutné, aby boli potiahnuté na určité miesto, nie je možné napr. zobrať akciu a priradiť ju na nejaký čas v riadku inej skupiny, pritom pre ktorú skupinu akcia platí je už dopredu dané. Na obrázku 3 je zobrazené rozhranie pri rozmiestňovaní rozvrhových akcií. Akcie musia byť dopredu nastavené, ak ideme pracovať s rozvrhom, inak je potrebné sa zložito preklikávať nastaveniami akcie. Pri konfliktoch, ak bol predmet nahradený za iný, dôjde k poprehadzovaniu rozvrhov. Používateľské rozhranie neumožňuje filtrovanie akcií na základe učiteľov a zároveň skupín, vždy je potrebné, aby bol vybratý iba jeden typ filtrovania.



Obr. 3: aSc TimeTables používateľské rozhranie [6]

Program podporuje iba platformu Windows a je realizovaný ako desktopová aplikácia, čo je problémové pri vytváraní rozvrhu viacerými osobami. Po vytvorení rozvrhu, je možné ho exportovať ako HTML, priamo do MS Excel, ale aj napríklad do mobilnej aplikácie. Export nie je možný bez zakúpenia programu.

K aSc Timetables je vytvorený rozsiahly manuál [6], ktorý podrobne popisuje všetky postupy na prácu s programom.

1.3.2 UniTime

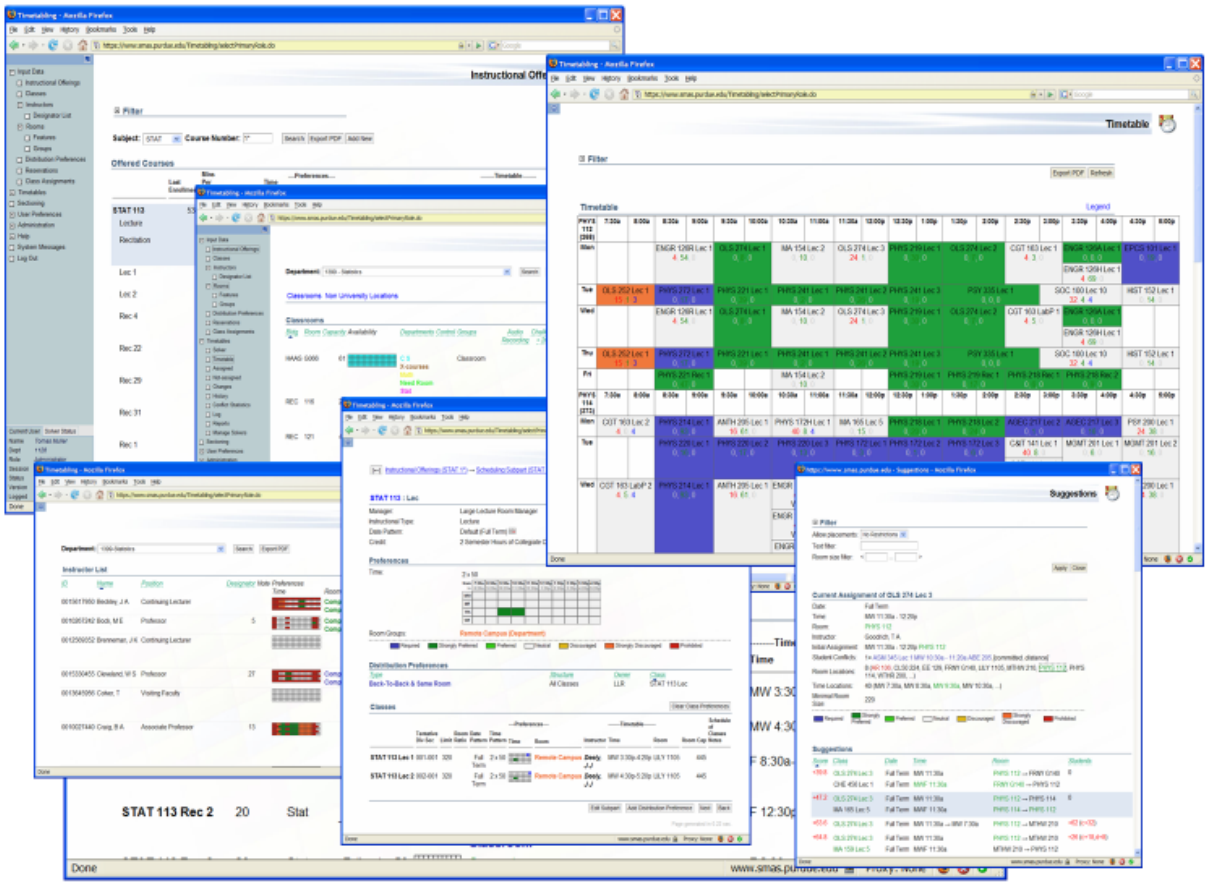
V tejto kapitole sa venujeme rozvrhovému systému UniTime [7] a pri jej tvorbe sme čerpali z oficiálnej dokumentácie. [8]

UniTime je komplexné riešenie pre akademické plánovanie, obsahujúce štyri komponenty:

- Plánovanie študentov: Cieľom je rozdeliť študentov do skupín za účelom, aby sa študenti dostali na kurzy, ktoré potrebujú,
- Plánovanie kurzov: Priradenie časov a miestností kurzom, v ktorých prebiehajú,
- Plánovanie skúšok: Priradenie časov a miestností skúškam, v ktorých prebiehajú,
- Manažment eventov: Manažment nepoúživaných miestnosti.

UniTime je implementovaný pomocou Java J2EE, ako databázu používa MySQL. Umožňuje prihlasovanie sa pomocou LDAP.

Dáta je možné pridávať manuálne alebo pomocou importu XML súboru. Export dát je možný len do XML súboru. Program umožňuje automatické generovanie rozvrhov pomocou knižnice CPSolver. Knižnica obsahuje framework založený na lokálnom vyhľadávaní, ktorý umožňuje modelovanie problému použitím primitív programovania s obmedzujúcimi podmienkami. Vyhľadávanie je založené na iteratívny dopredný vyhladávací algoritmus, ktorý operuje nad realizovateľnými, hoci nie nevyhnutne úplnými riešeniami s dôrazom na to, že všetky tvrdé obmedzenia musia byť splnené. Keďže ide o iteratívny algoritmus celý proces môže byť zastavený. Používateľské rozhranie je zobrazené na obrázku č. 4.



Obr. 4: UniTime používateľské rozhranie [7]

Výhodou tohto systému je aj poskytnutá demo verzia online[1].

[1]https://www.unitime.org/uct_demo.php

1.3.3 eRozvrh FCHPT

Táto kapitola sa zaoberá rozvrhovým systémom eRozvrh FCHPT, z absolvovaného interview s jej tvorcom, Ing. Júliusom Annusom.

Táto aplikácia bola vytvorená priamo pre Fakultu Chemickej a Potravinárskej Technológie STU. Aplikácia je implementovaná pomocou PHP. Pre každú stránku je vytvorený vlastný PHP súbor a zdieľaný súbor slúžiaci na prácu s databázou. Import dát prebieha priamo z databázy AIS, dáta sú exportované pre AIS, no nie priamo do databázy. Prihlásenie do systému je riešené cez vygenerovanú URL adresu, ktorá je odosielaná používateľom na email. URL adresa v sebe obsahuje AIS id študenta a vygenerovaný kód. Na strane servera sa následne kontroluje šte IP adresa.

Systém je rozsiahli a pre fakultu zabezpečuje všetky funkcionality týkajúce sa rozvrhov:

- vytváranie rozvrhov,
- vytváranie rozvrhu skúšok,
- prihlasovanie sa na rozvrhové akcie študentami,
- kontrolu dochádzky študentov.

Systém umožňuje import starého rozvrhu s rozvrhovými akciami s nastavením ekvivalencií pri zmene názvu a označenia predmetu, následne je rozvrh generovaný manuálne. Rozvrhári ústavov dokážu vytvárať rozvrhy a vedia si taktiež medzi sebou vymieňať predmety, ktoré majú na starosti. Pri tvorbe rozvrhov je nastaviť aj špeciálne prípady výučby ako napríklad náhrada dňa za iný deň. Umožňuje zadefinovať ďalšie obmedzenia podľa kapacít miestností a počte študentov zapísaných na predmet. Zber požiadaviek je realizovaný pomocou formulára, no pri vytváraní rozvrhu sa manuálne prezerajú. Pri zbere požiadaviek sa vyučujúcim umožňuje zvoliť si preferovanú miestnosť alebo potrebnú miestnosť. Vygenerované rozvrhy je možné pripraviť priamo na tlač.

Prihlasovanie študentmi na cvičenia ošetruje kolízie opakujúcich študentov. Pri výbere predmetov študenti nemajú možnosť vidieť kto predmet vyučuje. Proces prihlasovania sa na cvičenia prebieha v kolách a schválenie prihlásenia je na základe študijného priemeru. Kolízie medzi predmetmi sa riešia manuálne u rozvrhárov.

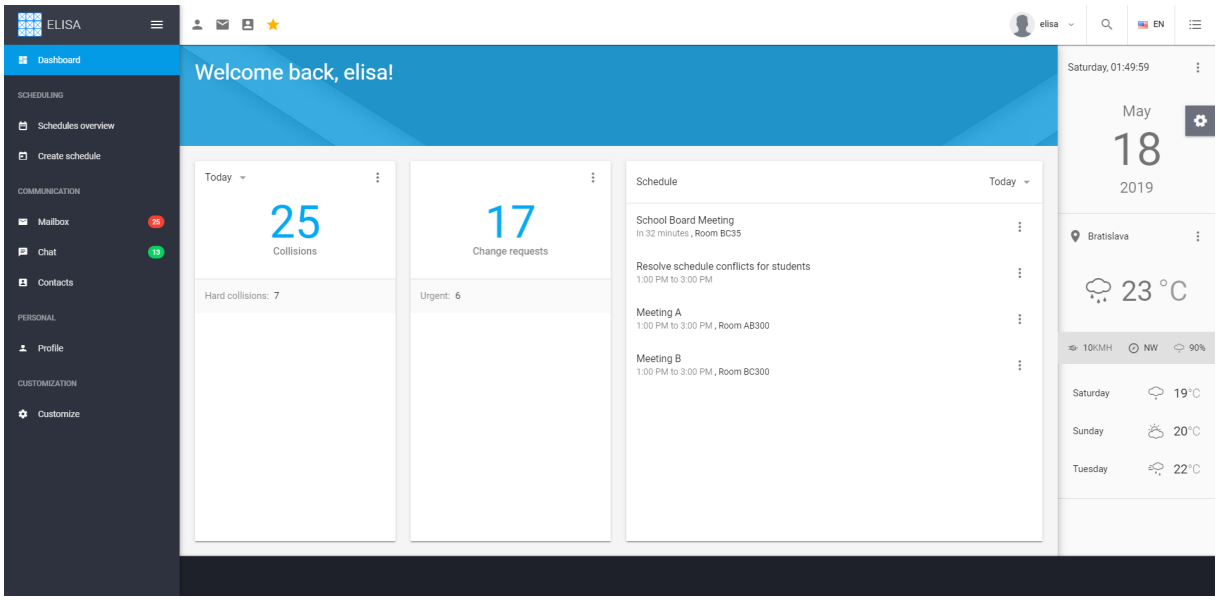
Systém úplne nahrádza AIS, ktorý slúži už len na zobrazenie importovaných rozvrhov.

1.4 Tvorba používateľského rozhrania k rozvrhovému systému pre FEI

V tejto kapitole sa budem venovať aktuálnemu stavu používateľského rozhrania vytvoreného v práci Tvorba používateľského rozhrania k rozvrhovému systému pre FEI (Dávid Bednár, 2018). [2]

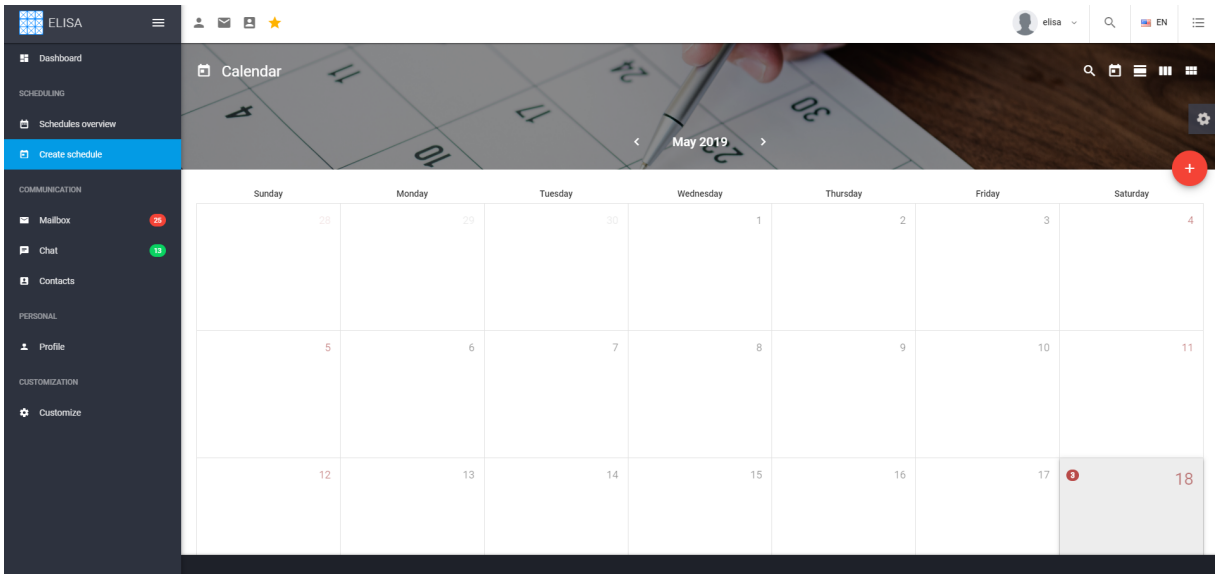
Používateľské rozhranie systému je implementované ako SPA aplikácia, pomocou jazyka TypeScript, s využitím frameworku Angular 5. SPA aplikácia je špecifická tým, že sa všetky zdrojové kódy načítavajú pri prvotnom načítaní aplikácie. Aplikácia je obsahuje viacero modulov:

- Modul autentifikácie: Modul slúži na autentifikačné činnosti, ako sú napríklad stránky na prihlásenie, stratu hesla a pod.,
- Modul prehľad: Modul slúži ako hlavná stránka aplikácie. Zobrazujú sa na nej informácie a metriky, ktoré sú pre používateľa podstatné podľa používateľskej role, modul je zobrazený na obrázku č. 5,
- Modul rozvrhovanie: Modul slúži na vytváranie rozvrhov a udalostí. Modul je najpodstatnejším modulom pre rozvrhára,
- Modul komunikácia: Modul zabezpečuje komunikáciu medzi používateľmi systému. V rámci modulu je implementovaná emailová komunikácia, zoznam kontaktov a chat,
- Modul profil: Modul slúži na zobrazovanie osobných informácií o prihlásenom používateľovi.



Obr. 5: Modul prehľad [2]

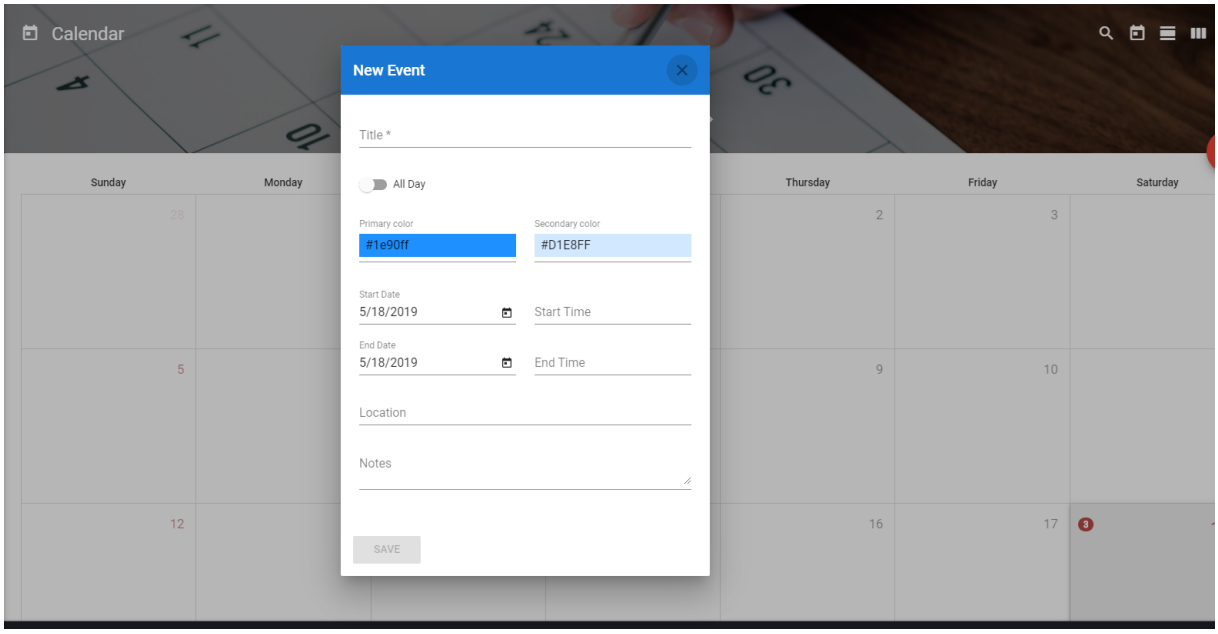
Klientska časť aplikácie nie je prepojená na serverovú časť, z toho dôvodu vytváranie dát bola vytvorená simulácia databázy. Aj napriek tomu, že bol implementovaný modul autentifikácie, v používateľskom rozhraní nie je využitý a teda nie je vykonávaná autentifikácia a autorizácia pomocou JWT [9] ako to bolo navrhnuté v práci.



Obr. 6: Modul rozvrhovanie [2]

Používateľské rozhranie neumožňuje prácu s dátami ako sú kurzy, vyučujúci a skupiny. Zobrazenie rozvrhu na pridávanie rozvrhových akcií je prehľadné, obrázok č 6, avšak pri prídávaní akcií nie je možné si zvoliť, pre ktorý kurz je akcia určená, alebo kto ju vyučuje

ako je zobrazené na obrázku č. 7. Jediné čo sa vypĺňa pri pridávaní akcií je čas kedy sa akcia odohráva, názov, ktorý sa pri akcii zobrazí. Kalendár podporuje funkcionalitu Drag&Drop.



Obr. 7: Pridanie rozvrhovej akcie [2]

Klientska časť aplikácie obsahuje prvky, ktoré nie sú pri tvorbe rozvrhu podstatné, napr. Modul prehľad zobrazuje počasie, alebo chat v Module komunikácie. Po dizajnovej stránke je rozhranie moderné a prehľadné, podporuje responzívny dizajn.

1.5 Požiadavky na systém

Požiadavky na nové rozhranie a prácu s novou verziou aplikácie boli definované na základe konzultácií s hlavným rozvrhárom fakulty FEI STU Mgr. Dávidom Panczom, PhD. nasledovne:

- Prihlásenie sa do systému pomocou AIS login,
- import a export dát,
- import zisťovanie ekvivalencií,
- pridávanie právomocí používateľom na prácu v systéme,
- pridávanie nových osôb, predmetov, miestností,
- možnosť pridávania požiadaviek od vyučujúcich,

- pridávanie poznámok k požiadavkám,
- možnosť vytvárania rozvrhov lokálnymi rozvrhármi pre jednotlivé ústavy a následne ich import do hlavného rozvrhu hlavným rozvrhárom,
- zobrazovanie konfliktov v rozvrhu,
- verzionovanie rozvrhov,
- krok späť,

Z analýzy v kapitole 1.3 sme identifikovali ďalšie požiadavky, ktoré sa týkajú hlavne vytvárania rozvrhov:

- Viacero pohľadov pri tvorbe rozvrhu,
- dizajn,
- jednoduché používanie.

2 Návrh riešenia

Táto kapitola opisuje návrh práce so systémom a návrh používateľského rozhrania.

Klientská časť aplikácie nebude nadväzovať na už implementovanú prácu a to z dôvodov absencií funkcionalít potrebných na navrhovanie rozvrhového systému, ako je napríklad správa dát, práca s rozvrhom nepodporuje mapovanie na reálne objekty, pridaním modulov, ktoré nie sú potrebné na prácu s rozvrhami popísaných v kapitole 1.4.

2.1 Definície pojmov

Schéma predstavuje rozvrh na zvolené obdobia(napr. zimný semester, letný semester).

Verzia rozvrhu je verzia pre zvolenú schému. Pri vytváraní rozvrhu rozvrhár priebežne ukladá svoju prácu a v prípade, že chce vytvoriť bod, do ktorého sa chce neskôr vrátiť, vytvorí novú verziu rozvrhu.

Subrozvrh je rozvrh vytváraný subrozvrhármi. Tí môžu vytvárať subrozvrhy iba z predmetov, ktoré patria do ich ústavu .

Rozvrhár je hlavná osoba poverená vytváraním rozvrhov na fakulte.

Subrozvrhár je osoba poverená vytváraním subrozvrhov na ústave.

Rozvrhová akcia je udalosť, ktorá má udané základné údaje a to: kedy nastáva, kto ju vyučuje, miestnosť alebo miesto kde nastáva a pre ktoré študentské skupiny je určená.

Kolízia je udalosť, ktorá nastáva v prípade, ak dôjde ku konfliktu rozvrhových akcií. Tento konflikt nastáva v prípade, ak v určitom čase má viacero rozvrhových akcií ten istý parameter, mimo študentských skupín.

Požiadavka je mäkké alebo tvrdé obmedzenie od vyučujúcich pri zostavovaní rozvrhu. Môže sa týkať preferovaného času a pod.

2.2 Tvorba rozvrhu

V kapitole 1.5 sme zadefinovali požiadavky na aplikáciu. Z požiadaviek môžme vidieť, že v systéme sa vykonávajú 2 separátne procesy. Táto kapitola sa venuje opisom týchto procesov.

Na prácu s aplikáciou je potrebné, aby sa používateľ autentifikoval do systému pomocou svojich prihlasovacích údajov do systému AIS. Ide o triviálny proces, v rámci ktorého overovanie voči AIS prebieha na strane servera.

Hlavný proces, ktorý prebieha v aplikácii je ale proces tvorby rozvrhu. Aplikácia poskytuje rozvrhárovi prostriedky na jeho vykonávanie. Ide o manuálne vytváranie rozvrhov

na fakulte.

Samotný proces tvorby rozvrhu na dané obdobie začína vytvorením novej schémy rozvrhárom, kedy taktiež zadáva, z ktorej schémy sa budú duplikovať dáta. Server pri tom vytvára aj defaultnú verziu pre schému. V prípade, že došlo ku zmenám dát v ramci AIS, je potrebné spustiť aj importovanie týchto dát. Po týchto nastaveniach schémy sa vyučujúcim a subrozvrhárom umožní vytváranie požiadaviek, resp. vytváranie nových subrozvrhov. Každý z aktérov tohto procesu môže vykonávať svoju činnosť, kým nie je rozvh publikovaný.

Pri tvorbe rozvrhu má rozvrhár možnosť upravovať alebo pridávať nové akcie. Pri pridaní takejto akcie prebieha kontrola na kolízie na strane klienta aplikácie. Ak táto kontrola vyhodnotí, že medzi akciami dochádza ku kolíziám, zobrazí túto kolíziu rozvrhárovi, ktorý vyhodnotí či, je kolízia chcená alebo nie.

Subrozvrhári vytvárajú rozvrhy pre ústavy rovnakým spôsobom. Po vytvorení subrozvrhu ho publikujú pre rozvrhára. Ten má následne možnosť importovať publikovaný subrozvrh. Pri importe subrozvrhu následne opäť prebieha kontrola na kolízie nových rozvrhových akcií. V prípade, že rozvrh neobsahuje žiadne kolízie, rozvrhár môže vytvoriť novú verziu rozvrhu alebo môže rozvrh publikovať. Úpravy rozvrhu sú ale aj naďalej možné, a to napríklad z dôvodu neotvorenia predmetu a pod.

2.3 Architektúra systému

Celá aplikácia je realizovaná pomocou klient-server modelu. Serverová časť aplikácie komunikuje s databázou PostgreSQL, v ktorej sú uložené všetky potrebné dáta na prácu s rozvrhom. Slúži na spracovanie dát z klientskej časti a taktiež zabezpečuje vytváranie verzií schém rozvrhu.

2.4 Webová služba

Na komunikáciu medzi klientskou a serverovou časťou aplikácie sa využíva Representational State TransferRepresentational State Transfer (REST) [10] webová služba. Systémy, ktoré využívajú REST model sú bestavové, teda systém nepotrebuje poznať stav, v ktorom sa nachádza klient. Server aj klient vykonávajú svoju činnosť nezávisle a komunikujú medzi sebou iba pomocou správ. Na komunikáciu sa pritom využíva IETF standard HTTP, v prípade zabezpečeného spojenia ide o HTTPS.

Pôvodne boli webové zdroje definovné ako dokumenty, ku ktorým sa pristupovalo pomocou URL adries, neskôr však dokázali identifikovať, pomenovať a spracovať akékoľvek entitu nachádzajúcu sa na webe. Operácie využívané REST aplikáciami sú definované

HTTP slovesá a to:

- GET operácia zabezpečuje získavanie zdrojov a nemala by mať žiadnu inú funkcionalitu,
- POST je najpoužívanejšia operácia, ktorá podľa odoslaných zdrojov a URI vyvoláva určitú funkcionalitu na server, ako napríklad pridanie dát do databázy,
- PUT operácia slúži na ukladanie údajov. V prípade, že odosielaný objekt už existuje dochádza k jeho modifikácii inak a vytvára nový objekt,
- PATCH sa využíva na čiastočnú úpravu existujúcich zdrojov,
- DELETE vymazáva odosielaný zdroj.

Odpovede REST aplikácie klientskej časti sú najčastejšie formátované ako HTML, XML, JSON alebo sa využívajú iné formáty. V prípade existujúceho server API odpovede prichádzajú vo formáte JSON. [11] Odpoveď webovej služby pozostáva z viacerých častí, no medzi najdôležitejšie patria: kód stavu o vykonanej operácii a telo správy, kde sa môžu prenášať požadované informácie ako zdroje alebo kolekcia zdrojov.

2.5 Webový klient

2.5.1 SPA

Starý prístup tvorenia webových aplikácii bol založený na komunikácii medzi klientom a serverom, kde pri zmene stránky zo servera načítaval celý obsah novej stránky. Logika aplikácií bola taktiež vykonávaná na strane servera.

Single-page application (SPA) je webová aplikácia, ktorá generuje obsah stránok dynamicky. Pomocou JavaScriptu pristupuje k DOM elementom už na existujúcej stránke. To ma za následok zrýchlenie interakcie medzi používateľom a obsahom stránky pretože sa nečaká na odpoveď zo servera. Single-page application (SPA) adresa pri takýchto aplikáciách neslúži na znovu načítanie stránky ale definuje, ktoré komponenty aplikácie sa majú zobraziť. Komunikácia so serverom prebieha pomocou Asynchronous JavaScript And XML (AJAX) požiadaviek alebo pomocou websocketov.

2.5.2 Autentifikácia a Autorizácia

Autentifikácia a autorizácia sú neodmysliteľnými súčasťami pri tvorbe zložitých webových aplikácií. V predchádzajúcich prácach [1] [2] boli podrobne opísané možnosti autentifikácie a postup pri vytváraní JWT. Autentifikácia je proces, pri ktorom sa používateľ overuje voči systému, inak povedané prihlasuje sa do aplikácie. Pri autentifikácií v našej

aplikácií dochádza ku generovaniu JWT. Tento proces je vykonávaný na strane servera, ale má vplyv aj na používateľské rozhranie. Uvádzame niekoľko možností riešenia autentifikácie:

Autentifikácia na základe vytvorených používateľov. Je to najpoužívanejší spôsob autentifikácie, pri ktorej nie je potrebné vytvárať vzdialený prístup na overovanie totožnosti. Pri tomto spôsobe je ale nutné zabezpečiť vytváranie používateľov, ktorý sa budú mocť prihlasovať do systému. Na strane servera preto je potrebné pre nich vytvoriť samostatnú tabuľku do databázy a na strane klienta zase ich registráciu.

Autentifikácia pomocou LDAP protokolu. Medzi požiadavkami bola aj integrácia prihlásenia sa do systému pomocou univerzitného systému AIS. Tento prístup autentifikácie je realizovaný pomocou LDAP protokolu na strane servera.

Na základe požiadaviek a zjednodušenia práce so systémom používame autentifikáciu pomocou LDAP protokolu. Pre uchovanie JWT sme sa rozhodli použiť localstorage, keďže týmto spôsobom bude token uchovaný po odchode zo stránky a token má časovo obmedzenú platnosť, ktorá sa kontroluje na strane servera.

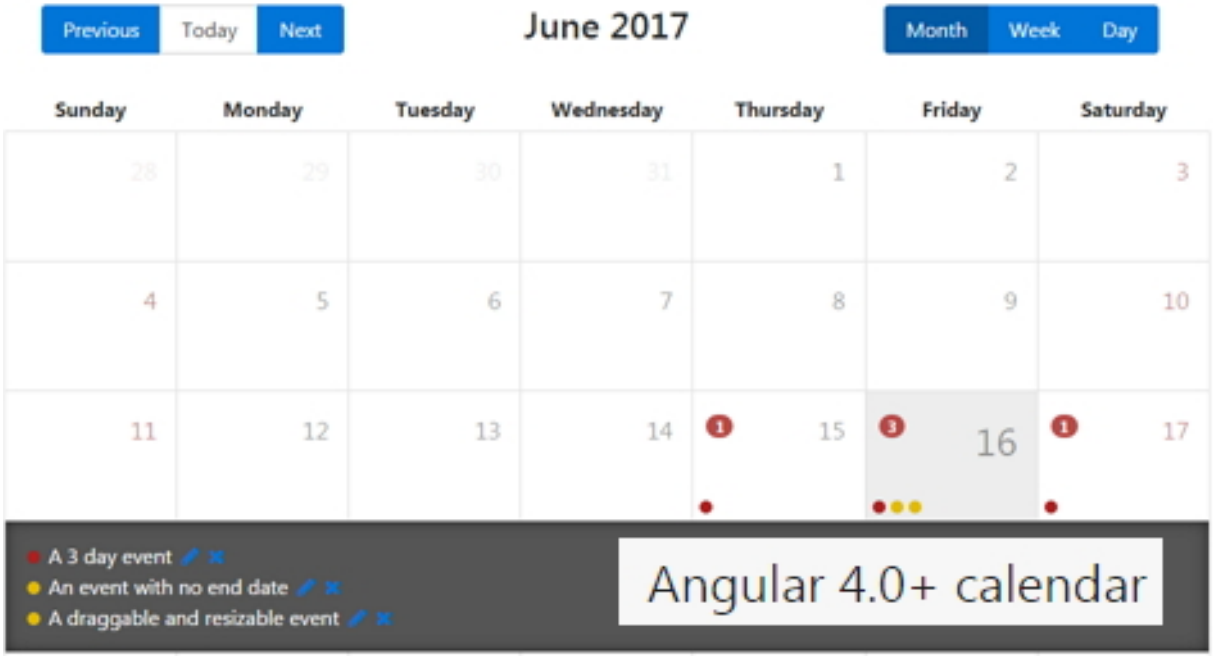
2.5.3 Zobrazovanie rozvrhu

Práca a zobrazovanie kalendárov vo webových aplikáciách je častým problémom. V prípade, ak kalendár má obsahovať mnoho funkcionalít, je vytváranie vlastného kalendára časovo náročné. Práve preto je na túto funkcionalitu vhodné použiť už existujúce knižnice. Na porovnanie sme vybrali iba kalendáre, ktoré sú podporované a priamo implementované ako komponent pre Angular.

Angular 6.0+ calendar. Táto časť sa zaoberá rozvrhovým systémom Angular 6.0+ calendar, pri jej tvorbe sme čerpali oficiálnej z stránky. [12].

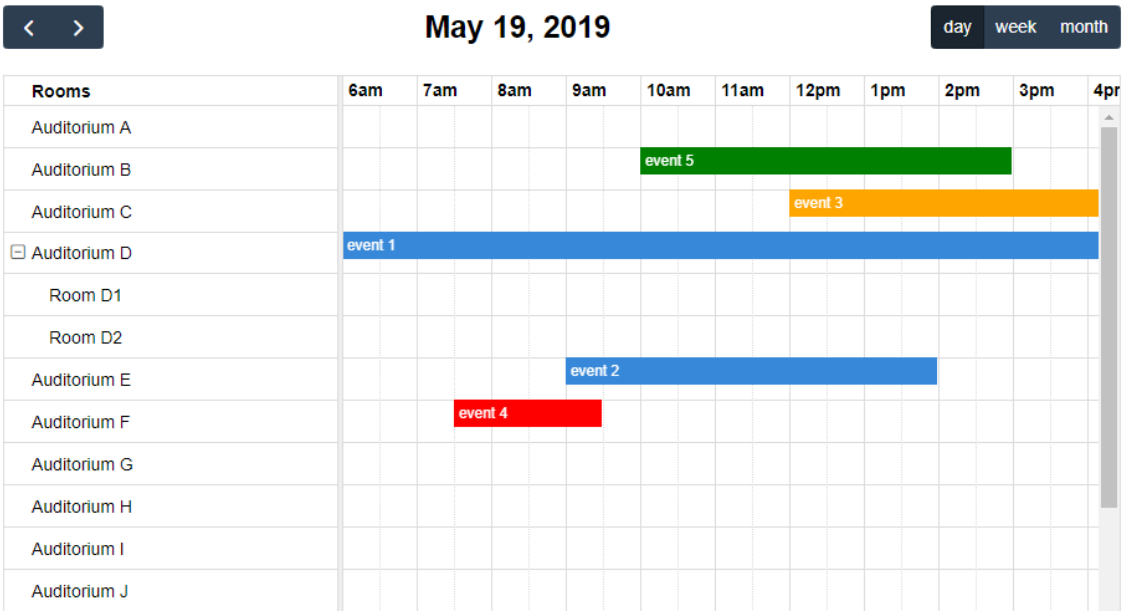
Angular Calendar je distribuovaný s open-source licenciu. Je nástupcom Angular Bootstrap Calendar [**angularBootstrapCalendar**], ktorý bol vyvíjaný pre ako plugin AngularJS. Aj napriek tomu, že je vyvíjaný len jednou osobou, tak svojimi funkcionalitami a User Experience (UX) je na vysokej úrovni. Aktuálna verzia kalendára je 0.27.7. Kalendár podporuje všetky funkcionality, ktoré by sme potrebovali v našej klienskej aplikácii. Kalendár umožňuje prepínať pohľady medzi mesiacom, týždňom a dňom.

Pohľad na týždeň má na horizontálnej osy dni v týždni a na vertikálnej hodiny a nepoporuje obrátenie osí. Na obrázku č. 8 je zobrazený náhľad komponent.



Obr. 8: Angular 6.0+ calendar [12]

FullCalendar Scheduler. Táto časť sa zaoberá rozvrhovým systémom fullCalendar, pri jej tvorbe sme čerpali z oficiálnej stránky [13] a skúseností s používaním. FullCalendar je distribuovaný ako dva rôzne produkty: Standard a Scheduler. Scheduler je distribuovaný pod licenciou GPLv3. Vo vývoji je od roku 2009 a je stále popdporovaný. Ide o JavaScriptový kalendár, neskôr bol prepojený na Angular. Z toho dôvodu je implementácia pri použití tejto kniznice náročnejšia. Podporuje Drag&Drop funkcionalitu, zmenu dĺžky trvania akcie. Zobrazenie týždňa po dňoch je ale podobne ako pri Angular Calendari. FullCalendar Sheduler avšak poskytuje vkladanie zrojov do kalendár ako vertikálnu os. Na obrázku č. 9 je zobrazený FullCalendar Scheduler.



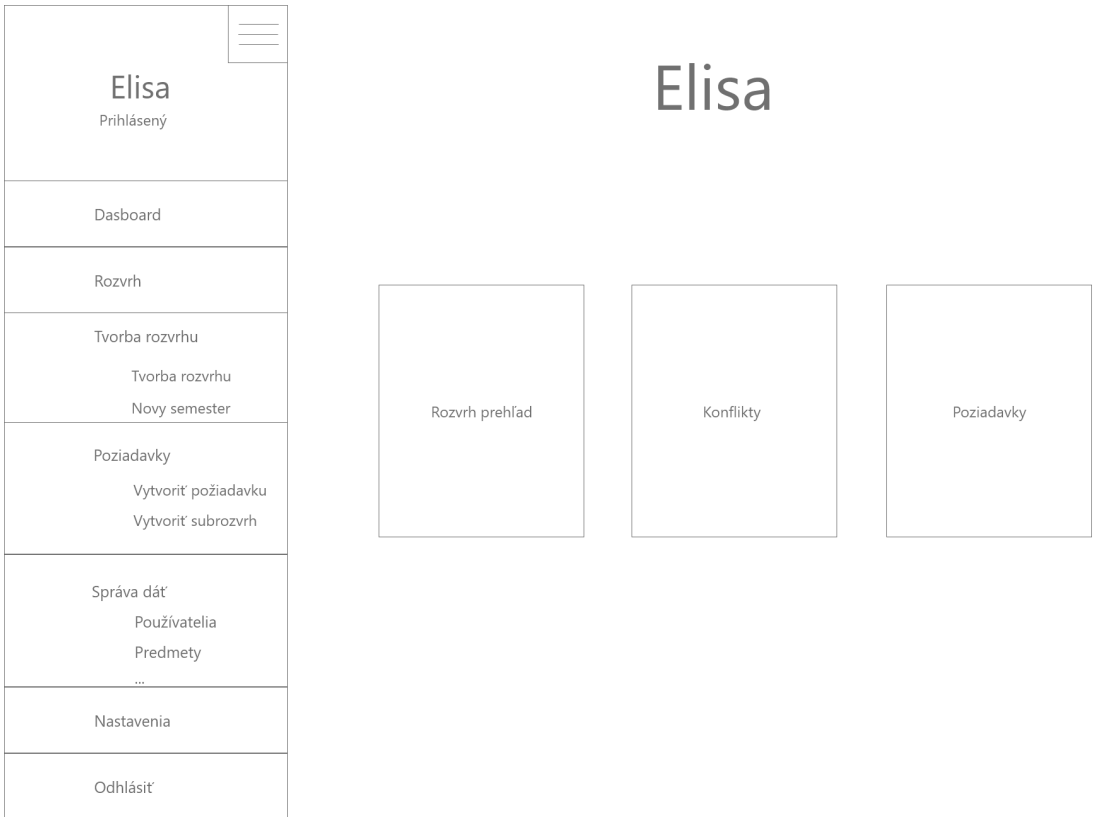
Obr. 9: FullCalendar Scheduler [13]

V našej aplikácii sme sa rozhodli používať FullCalendar Scheduler.

2.5.4 Návrh rozhrania

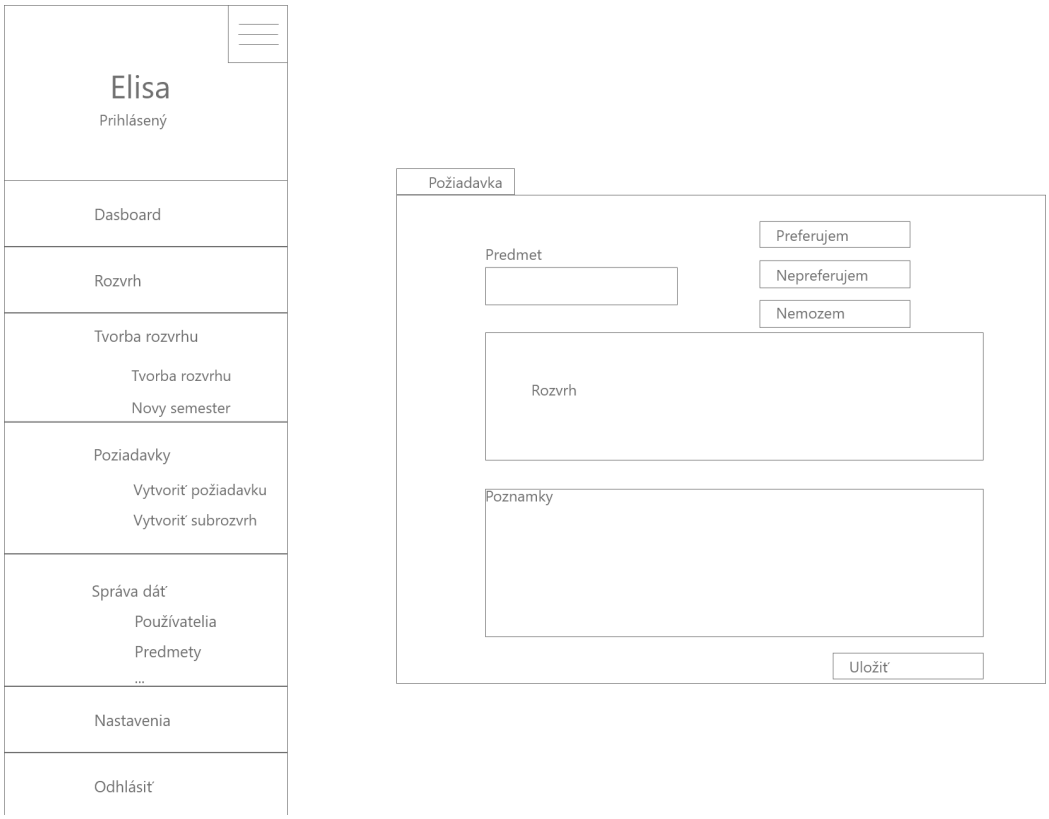
Pri vytváraní systému na vytváranie rozvrhov treba klásť dôraz aj na používateľské rozhranie. Jedna z požiadaviek bola, aby používateľské rozhranie bolo prehľadné a jednoduché na používanie. Takéto rozhranie dokáže zefektívniť prácu. Rozhranie webového klienta môžme rozdeliť na stránky zobrazované pred prihlásením sa do systému a po prihlásení sa. Po príchode na webovú stránku je zobrazený posledný zverejnený rozvrh. Prezeranie rozvrhov je umožnené aj pre neprihlásených používateľov a to z toho dôvodu, aby si študenti mohli prezerať svoje rozvrhy. Na obrázku 10 je zobrazená domovská stránka po prihlásení do systému. Na nej sa nachádzajú potrebné informácie pre rozvrhára po prihlásení a to:

- Prehľad o aktuálnej schéme rozvrhu a jej verzii,
- výpis konfliktov pre schému,
- zoznam požiadaviek od vyučujúcich.



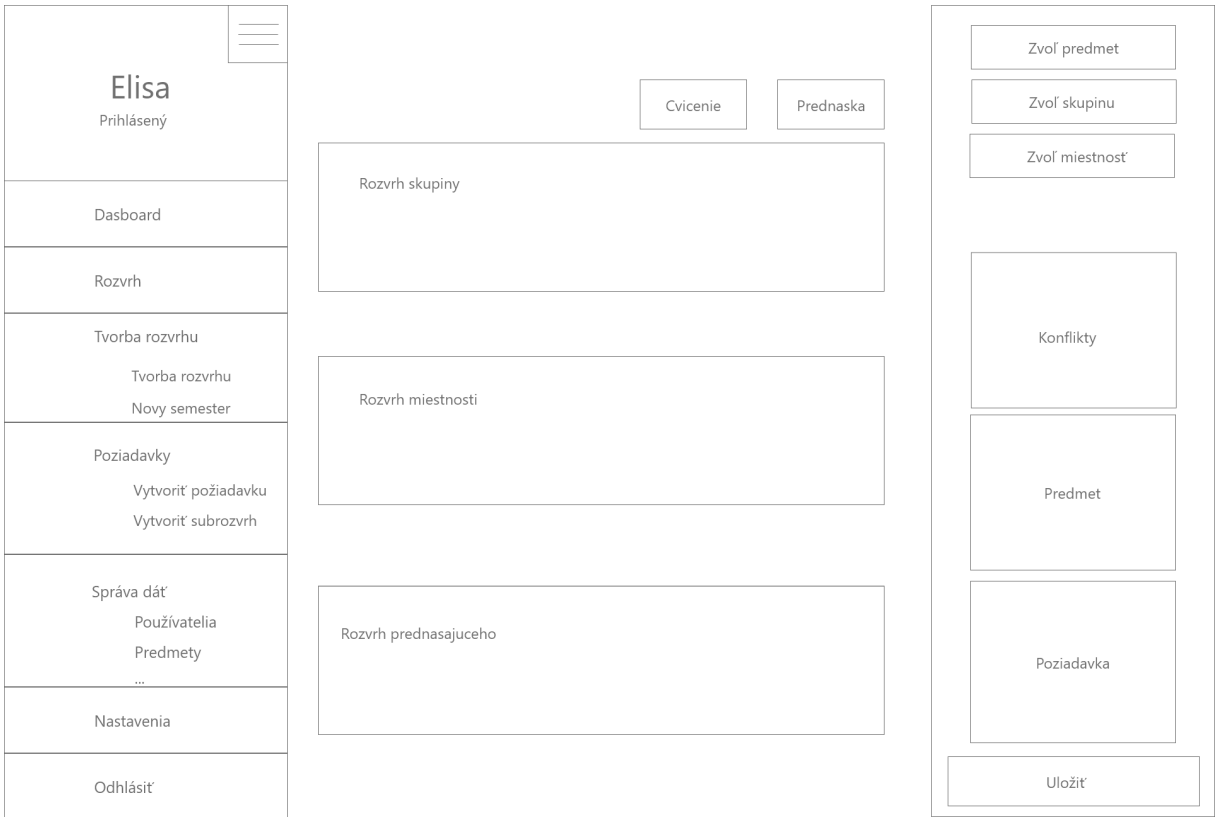
Obr. 10: Návrh Dashboard

Jedna z funkcionalít aplikácie je aj vytváranie požiadaviek na predmety od vyučujúcich. Pri vypĺňaní požiadaviek vyučujúci má možnosť zadefinovať, pre ktoré predmety požiadavku vypĺňa. Následne na zobrazenom rozvrhu má možnosť vyklikať, ktoré časy mú vyhovujú a ktoré nie. Celý návrh rozhrania je zobrazený na obrázku č. 11



Obr. 11: Návrh tvorby požiadaviek

Vytváranie rozvrhov je najdôležitejším komponentom pre rozvrhára a subrozvrhárov. Klientske rozhranie poskytuje 3 rôzne rozvrhy a to podľa zvoleného predmetu (resp. vyučujúceho), miestnosti a zvolených skupín. Pridávanie rozvrhových akcií je riešené pomocou Drag&Drop. Voľba rozvrhovej akcie prebieha výberom predmetu, miestnosti, a skupín, ktorým bude pridaná daná akcia. Rozvrhár musí mať ale pritom prehľad aj o požiadavke, ktorá k danému predmetu prislúcha. Nevyriešené kolízie na aktuálnej verzii rozvrhu sa zobrazujú v pravej časti okna. Náhľad na vytváranie rozvrhov je zobrazený na obrázku č. 12.



Obr. 12: Návrh vytvárania rozvrhov

3 Použité technológie

Táto kapitola sa zaoberá technológiami, ktoré sú použité pri vývoji práce.

3.1 Serverová časť

Serverová časť aplikácie poskytuje REST API. Vývoj je implementovaný v jazyku Python s frameworkom Django. Hlavnou funkcionalitou servera je ukladanie dát do databázy PostgreSQL.

3.1.1 Python

Python je intepretovaný, interaktívny objektovo-orientovaný programovací jazyk. Obsahuje moduly, výnimky, dynamické typovanie, dynamické typy údajov s vysokou úrovňou a triedy. Python kombinuje pozoruhodnú silu s veľmi jasnou syntaxou.[2] Pôvodne bol vytvorný Guidom van Rossumom. Python 2 bol publikovaný v roku 2000 a Python 3 bol publikovaný v roku 2008, ale veľa jeho hlavných funkcionalít je spätne portovaná aj do verzie 2.6 a 2.7. Aktuálna verzia používaná na serveri je Python 3.6.

3.1.2 Django

Django je vysoko úrovňoví, Python Web framework, ktorý podporuje rýchly vývoj a čistý, pragmatický dizajn.[3] Bol navrhnutý tak, aby spoločné úlohy vývoja webu boli rýchle a jednoduché. Je založený na princípe DRY (Don't Repeat Yourself) a snaží sa dosiahnuť prepojenie medzi komponentmi aplikácie, čo ich robí opakovane použiteľnými a dáva vývojárovi možnosť vykonávať zmeny a vylepšenia kódu bez ovplyvnenia celého projektu. Je distrubuovaný pod open-source licenciou. Na serveri sa používa verziu Django 2.2.

3.1.3 PostgreSQL

PostgreSQL je open-source objektovo-orientovaný databázový systém, ktorý používa a rozširuje jazyk SQL.[4] Beží na všetkých hlavných operačných systémoch a od roku 2001 je ACID kompatibilný.

3.2 Klientska časť

Klientska časť aplikácie je vytvorená ako SPA, ktorá komunikuje so serverom pomocou REST API. Implementovaná je pomocou jazyka TypeScript s použitím frameworku Angular a UI frameworku Angular Material. Technická dokumentácia je vytvorená pomocou nástroja Compodoc.

[2] https://docs.python.org/3/faq/general.html
[3] https://www.djangoproject.com/
[4] https://www.postgresql.org/about/

3.2.1 TypeScript

TypeScript[5] je nadstavba jayzka JavaScript so zámerom zjednodušenia vývoja veľkých JavaScriptových aplikácií. [14] Ide o open-source objektovo-orientovaný programovací jazyk. TypeScript ponúka systém modulov, triedy, rozhrania a typový systém. Podpora pre triedy je zosúladená s návrhmi, ktoré sú v súčasnosti štandardizované pre program EcmaScript 6. Každý JavaScriptový program je validným TypeScript programom. Môže byť použitý na vývoj programov na klientskej strane (z angl. Client-side), ale aj na serverovú stranu. Prvý krát bol publikovaný v roku 2012 a vyvíjaný bol spoločnosťou Microsoft.

3.2.2 Angular

Angular[6] je framework a taktiež platforma pre tvorbu klientských aplikácií v HTML a TypeScripte. Základným stavebným blokom Angularu je komponent. Ten definuje, aké služby sa budú používať, alebo ktoré šablény sa zobrazia. Komponenty a služby sú následne zapuzdrené do modulov, ktoré vytvárajú kolekcie navzájom prepojeného kódu. Aplikácie sú teda iba vyskladaním týchto komponentov. Angular je open-source vyvinutý spoločnosťou Google. Ide o kompletné prepísanie AngularJS, ktorý bol napísaný v jazyku JavaScript. Každá nová verzia v sebe prináša bohaté zmeny. Pri prechode na Angular 7 to boli tieto[7]:

- CLI dolňanie: Zrýchlenie práce pri vytváraní nových komponentov v prípade ak sa pracuje s Angular CLI,
- Výkon aplikácie: zlepšenie výkonu ekosystému,
- The Angular Material CDK: Implementovanie Drag&Drop a virtuálneho skrolovania,
- Zlepšenie dostupnosti výberov: Zlepšenie dostupnosti aplíkacie pri využívaní funkcie `selectelement` v `mat-form-field`,
- Angular Elements: Možnosť zadefinovania vlastných elementov,
- Partnerské spustenie: Spustenie komunitných projektov,
- Aktualizovanie závislostí: Aktulizovanie závislostí pre TypeScript 3.1, RxJS 6.3, Node 10.

[5] https://www.typescriptlang.org/

[6] https://angular.io/

[7] https://www.quora.com/Whats-new-in-Angular-7-What-are-the-differences-between-Angular-6-and-7

Nami navrhnutá aplikácia používa Angular 7.2.12, ktorý je podporovný do roku 2020.

3.2.3 Angular Material

Angular Material[8] je front-end UI framework, optimalizovaný pre prácu s Angularom. Poskytuje sadu znovupoužiteľných stabilných UI komponentov, ktoré sú založené na Material Design. Vďaka nemu je možné vytvoriť responzívny dizajn aplikácií. Poskytuje komponenty pre layout, formuláre, ale aj pre zobrazovanie dát do tabuliek s implementovanými prvkami ako je sortovanie, alebo stránkovanie. Nami navrhnutá aplikácia používa verziu 7.3.4.

3.2.4 LESS

LESS[9] dynamický preprocesorový kaskádny štýl kompilovaný do CSS. Prvá verzia bola napísaná pomocou Ruby, no neskôr bol úplne prerobený do JavaScriptu. Rozširuje CSS o dynamické správanie ako je napríklad zadefinovanie premenných, operácií a funkcií. Dokáže byť implementovaný na klientskej strane ale aj na serverovej strane.

3.3 Vývojové prostredie

Aplikácia bola vyvíjaná v prostredí Windows, pomocou Windows Subsystem for Linux, na ktorom je bežiaci operačný systém Ubuntu 16.04.4 LTS. Databázu sme používali PostgreSQL 9.5.14. Ako integrované vývojové prostredie sme používali WebStorm a na inštaláciu Angular modulov sme vyžívali Angular Cli[10].

[8] https://material.angular.io/
[9] http://lesscss.org/
[10] https://cli.angular.io/

4 Implementácia

Kapitola sa venuje realizácií webového klienta. Popísali sme spôsob vývoja aplikácií pomocou frameworku Angular, na základe ktorého prebehla implementácia. Hlavným zdrojom informácii pri implementácii bola oficiálna dokumentácia pre framework Angular [15].

4.1 Angular architektúra

Angular je používaný na vytváranie SPA aplikácií. Angular aplikácie sú modulárne a používajú vlastný modulárny systém NgModules. NgModuls sa líši a zároveň dopĺňa JavaScript (ES2015) [16] moduly.

Aplikácie obsahujú 7 základných zložiek:

1. Komponenty: definujú triedu, ktorá sa obsahuje aplikačné dáta a logiku, je k nemu priradená HTML šablóna, ktorá definuje zobrazenie (z angl. View),

2. Šablóny: slúžia na kombinovanie HTML s Angulár značkovaním(z angl. Markup), ktoré modifikuje HTML pred jeho zobrazením,

3. Metadáta: každý komponent alebo služba v Angular aplikácií je jednoduchá trieda(z angl. Class). Ako aplikácia bude spracovávať triedu je špecifikované pomocou metadát,

4. Data binding: slúži na prepojenie aplikačných dát a DOM. Data binding vieme rozdeliť na dva typy a to: Event binding a Property binding. Event binding zabezpečuje aby aplikácia reagovala na zmeny od používateľa a následne upravovala aplikačné dáta. Property binding zabezpečuje opačný prípad, a to prevod výsledkov aplikačnej logiky do HTML

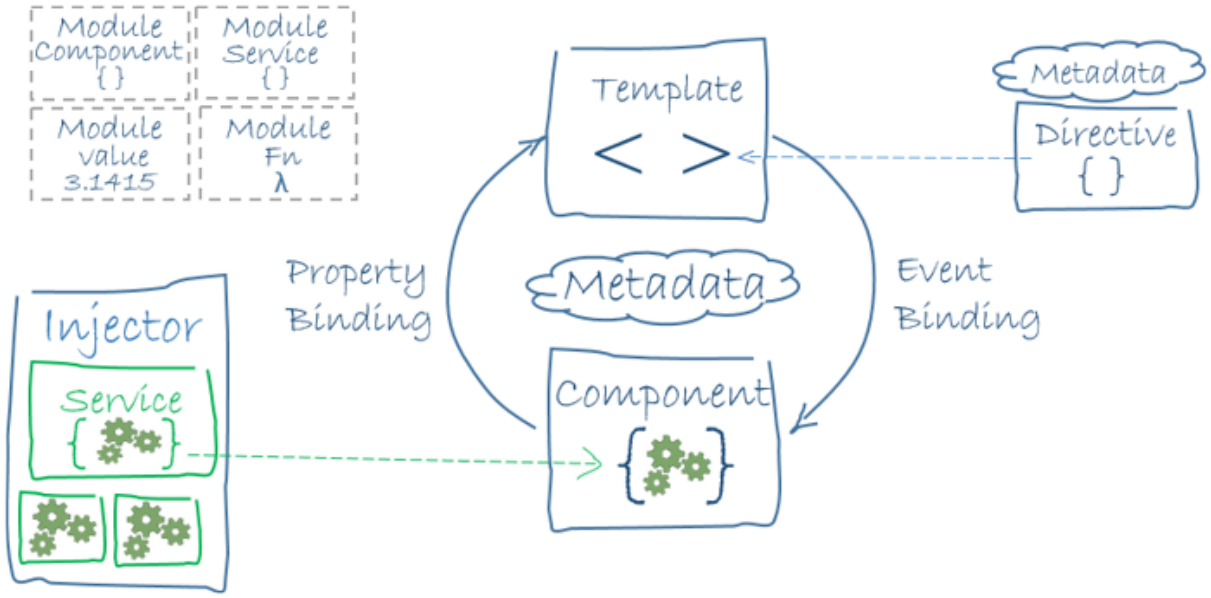
5. Direktívy: ak Angular prekladá (z angl. render) stránku, tak transformuje DOM podľa inštrukcií direktív. Poznáme dva typy direktív, Štrukturálne direktíva, ktoré pridávajú, odstraňujú alebo znovu inicializujú DOM elementy a Atribútové direktíva, ktoré menia správanie sa existujúcich elementov,

6. Služby: poskytujú zdieľanie funkcionalít, ktoré sú využívané vo viacerých komponentoch,

7. Dependency Injection (DI): umožňuje udržiavať triedy komponentov štíhle a efektívne a to delegovaním funkcionalít iným triedam.

NgModule obsahuje komponenty, služby a iné súbory ktoré spolu súvisia svojou funkcionalitou. Môže importovať funkcie, ktoré sú exportované z iných modulov. Každá Angular aplikácia obsahuje aspoň jeden takýto modul a to rootovací modul pomenovaný AppModule.

Na obrázku č. 13 je zobrazený vzťah medzi vymenovanými zložkami.



Obr. 13: Angular architektúra [17]

4.2 Štruktúra klienta

Pri implementovaní klientskej časti aplikácie sme dodržiavali Angular architektúru. Bola implementovaná ako Single-page application (SPA), teda všetok jej zdrojový kód je načítavaný pri prvom otvorení aplikácie. Prechody medzi hlavnými komponentami sú zabezpečené zmenou URL adresy. Pri jej zmene nedochádza k opätovnému načítaniu aplikácie, ale na základe nej aplikácia rozpoznáva, ktorý komponent má zobraziť. Túto funkcionalitu sme implementovali pomocou Angular Router, ktorý interpretuje URL zadané v prehliadači. Router umiestňuje komponenty do elementu router-outlet. V aplikácií sa nachádzajú dva tieto elementy, prvý element zobrazuje stránky podľa toho, či je používateľ autentifikovaný, druhý element slúži na navigáciu po autentifikácii.

Snažili sme sa o to aby komunikácia medzi klientskou časťou a serverovou časťou prebiehala čo minimálne, dáta zo serverovej časti nie sú načítavané hneď pri spustení aplikácie, ale až pri prechode na určité podstránky.

Aplikácia má definované modeli, s ktorými pracuje, tie ale nie sú zhodne s modelmi v serverovej časti, preto pri načítavaní dát prebieha ich konvertovanie. Na prácu s rozvr-

hom sme definovali model pre rozvrhové akcie, ktorý je zobrazený vo výpise č. 3 a model pred kolízie zobrazený vo výpise č. 2

```
export class Event {
  id: string;
  idType: number;
  idCourse: number;
  idTeacher: number;
  idRoom: number;
  day: number;
  startTime: number;
  endTime: number;


  constructor(id: string, idType: number, idCourse: number, idTeacher: number, idRoom: number, day: number,
      startTime: number, endTime: number) {
    this.id = id;
    this.idType = idType;
    this.idCourse = idCourse;
    this.idTeacher = idTeacher;
    this.idRoom = idRoom;
    this.day = day;
    this.startTime = startTime;
    this.endTime = endTime;
  }
}
```

Listing 1: Ukážka modelu akcie

```
export class Collision {
  public id: number;
  public groups: number[];
  public events: number[];
  public type: string;
  public start: number;
  public end: number;
}
```

Listing 2: Ukážka modelu kolízie

Stránka podporuje viacero jazykov, medzi ktorými je možné prepínať. Aktuálny jazyk je uložený v localstorage. Po prepnutí jazyka sa táto hodnota prepíše a dochádza k opätovnému načítaniu aplikácie. Preklady sú definované v XML súboroch.

Informácie potrebné na autorizáciu sa ukladajú do localstorage. Takýmto sposôbom sú viditelné pre používateľa a môže ich upraviť, no overenie autorizácie prebieha na strane servera. Údaje ako naprv. rola používaľa sú uložené iba kvôli správnemu zobrazeniu komponentov na používateľskom rozhraní (napr. vygenerovanie správneho menu).

4.3 Moduly aplikácie

Naša vytvorená aplikácia je modulárna. Podľa funkcionalít aplikácie sme vytvorili dva moduly, ktoré vykonávajú svoju špecifickú činnosť.

4.3.1 ModuleAuthentification

Modul zabezpečuje autentifikáciu a autorizáciu voči serveru. Používateľ sa autorizuje voči serveru pomocou podstránky na prihlásenie pomocou vyplnenia formulára, kde zadáva svoje prihlasovacie údaje do systému AIS. Pri úspešnej autentifikácií, získava JWT token, refresh token, používateľské role a údaje. Tieto údaje aplikácia ukladá do localstorage.

Model v sebe obsahuje aj autorizáciu voči serveru. Pri každom požiadavke sa do hlavičky požiadavky pridáva JWT token, v prípade, ak je používateľ autentifikovaný, ako to je zobrazené vo výpise č. 3.

Token na serverovej časti má časovo obmedzenú platnosť. V prípade, ak táto platnosť vyprší server odpovedá chybovou kódom. Následne požiadame server o vygenerovanie nového tokenu a opakujeme poslednú požiadavku na server.

```
1  @Injectable()
2  export class TokenInterceptor implements HttpInterceptor {
3    intercept(req: HttpRequest<any>,
4              next: HttpHandler): Observable<HttpEvent<any>> {
5
6      const jwt = localStorage.getItem('token');
7
8      if (jwt) {
9        const cloned = req.clone({
10         headers: req.headers.set('Authorization',
11           "Bearer " + jwt)
12       });
13       return next.handle(cloned);
14     }
15     else {
16       return next.handle(req);
17     }
18   }
19 }
```

Listing 3: Ukážka vkladania JWT tokenu do požiadaviek

4.3.2 ModuleTimetable

Hlavný modul aplikácie, ktorý zabezpečuje tvorbu celého rozvrhu na zvolené obdobie. Sú v ňom obsiahnuté tieto podstránky:

- Prehľad základných informácií,

- vytvorenie schémy rozvrhu,
- zobrazovanie rozvrhu,
- formulár na požiadavky od učiteľov,
- navrhovanie rozvrhu,
- správa dát.

Na začiatku procesu pri tvorbe rozvrhu je potrebné vytvoriť novú schému a importovať dáta. Po tomto procese je vyučujúcim umožnené vytvárať svoje požiadavky k jednotlivým predmetom. Pri tvorbe rozvrhu rozvrhár a subrozvrháry využívajú tú istú podstránku a na základe role, ktorú majú v systéme sa im sprístupňujú iné informácie.

Ako sme už spomínali v kapitole 4.2 pri implementácii sme sa snažili, komunikácia medz klientom a serverom bola minimálna. Z toho dôvodu sa dáta, ako sú predmety, vyučujúci, rozvrhové akcie a miestnosti načítavajú všetky na začiatku pri spustení podstránky na tvorbu rozvrhu. Dáta sú objemné a na prácu s nimi je potrebné ich premapovať ako asociatívne pole (z ang. Associative array), preto prvotné načítanie tejto podstránky je časovo náročné, no následne sa s ňou pracuje plynulo bez ďalších prerušení.

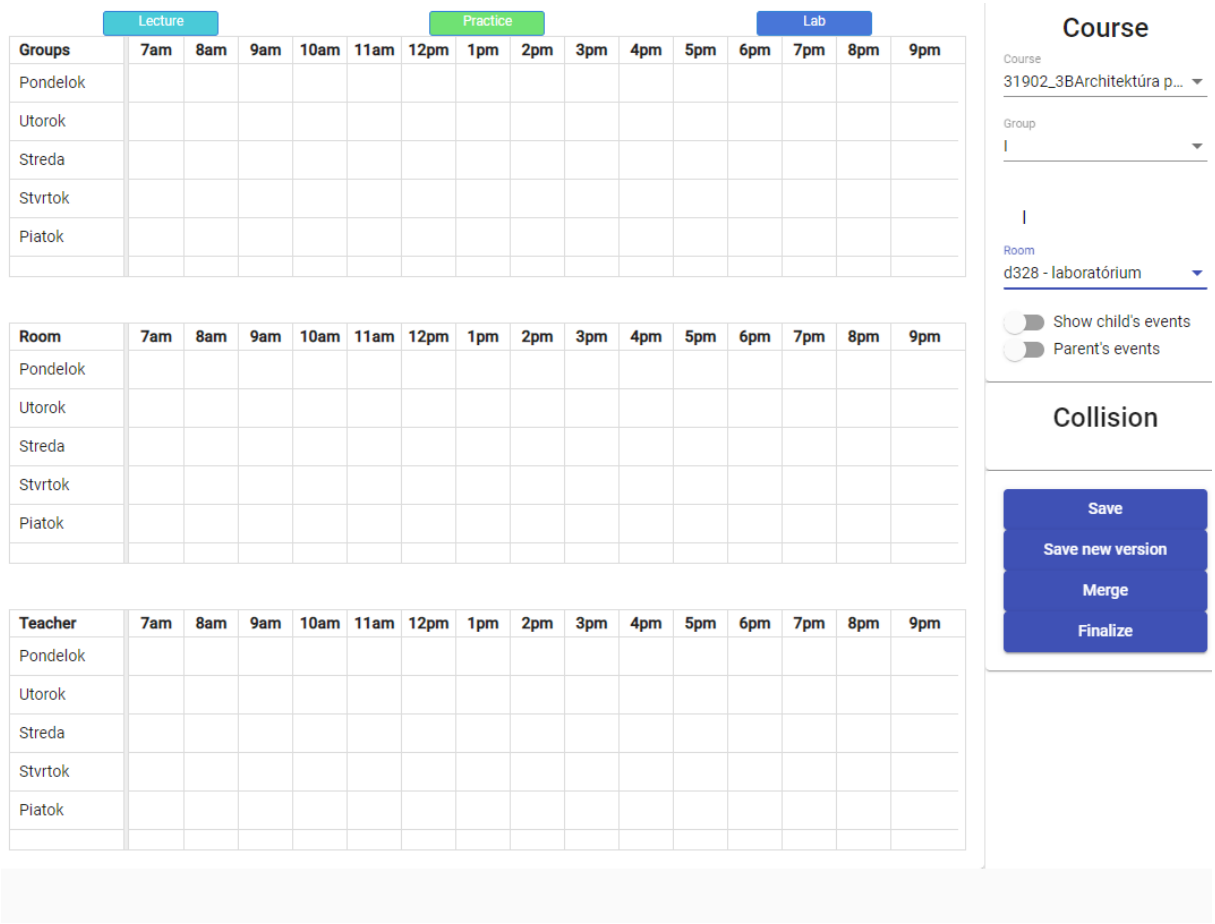
Pri pridávaní rozvrhových akcií prebieha kontrola na možné kolízie, ktoré sa kontrolujú s akciami pre zvolený predmet, miestnosť a skupiny. Po zistení kolízie je následne potrebné zistiť, či akcia bude pridaná do existujúcej kolízie, alebo sa vytvorí nová kolízia z konfliktných akcií.

4.4 Používateľské rozhranie

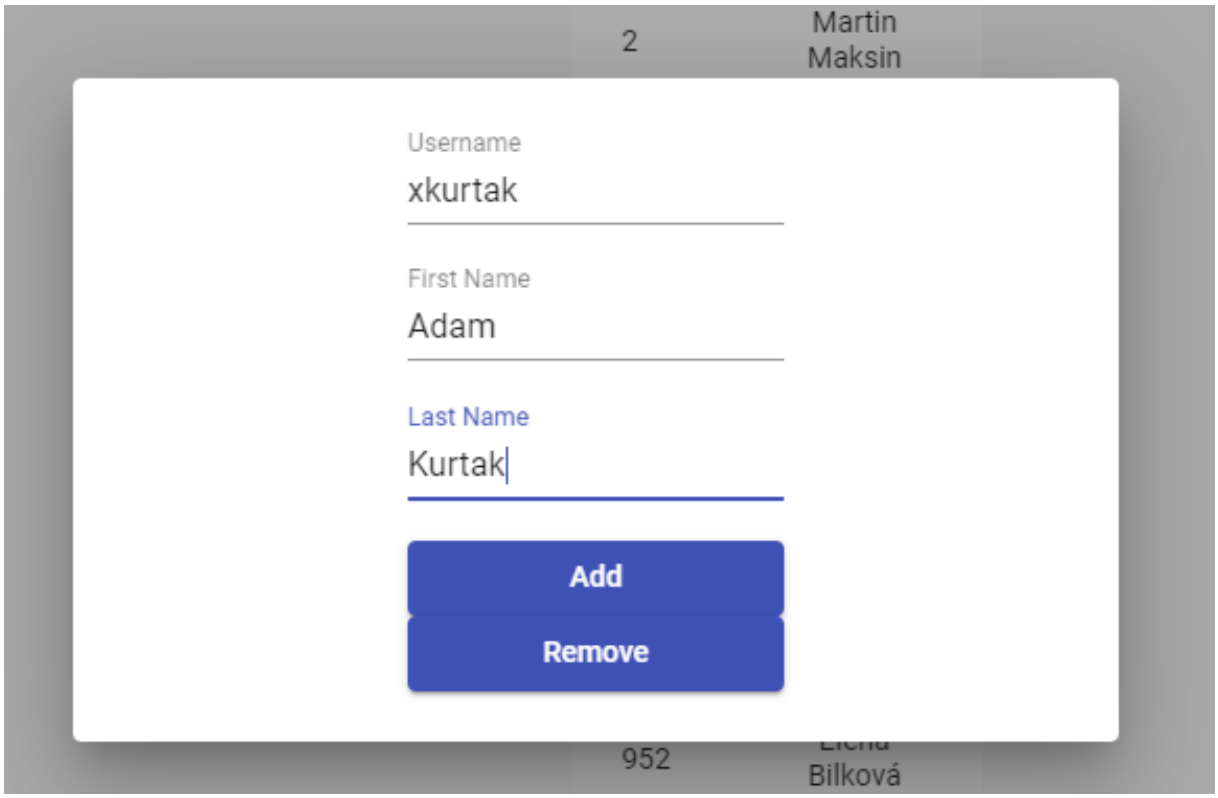
Používateľské rozhranie sme vytvárali s dôrazom na UX. Pri takomto rozhraní je potrebné aby bolo prehľadné, všetky funkcionality boli jednoducho dostupné a hlavne aby bolo intuitívne. Funkcionality poskytuje podľa toho akú rolu zohráva v aplikácii. Ak používateľ nemá dostatočne oprávnenia na vykonávanie danej funkcionality, nemala by mu byť ponúknutá možnosť dostať sa na podstránku.

Neprihlásený používatelia majú možnosť si prezerať posledný zverejnený rozvrh, filtrovaný podľa študijných skupín. Po prihlásení sa študenti dostávajú do administratívy systému. Podľa role používateľa je generovaná navigácia, pri ktorej sme brali dôraz na to, aby sa ku všetkým funkcionalitám dalo dostať pomocou 3 klikov [18]. Ide o nepísané pravidlo, ktoré hovorí o tom, že zobrazenie akejkoľvek podstránky z menu by mal používateľ použiť maximálne 3 kliky.

Podľa navrhovaného rozhrania v kapitole 2.5.4 sme vytvorili podstránky pre nástenku (z angl. Dashboard) a podstránku vytvárania rozvrhov. Rozvrhy v podstránke vytvárania rozvrhov, v sebe majú implementované funkcionality ako ťahanie okrajov a Drag&Drop v rámci rozvrhu. Pridávanie rozvrhových akcií je realizované funkcionalitou Drag&Drop z externých zdrojov, ktoré v našom prípade predstavujú tlačidlá nad rozvrhmi. Týmto prístupom je zabezpečené rýchle pridávanie nových akcií do rozvrhov. Na obrázku č. 14 je zobrazená ukážka tejto podstránky. Rozvrhové akcie v kalendári sú klikateľné a zobrazujú informácie o akcii, pomocou dialógového okna. Pomocou týchto okien je realizované aj pridávanie nových údajov ako je zobrazené na obrazku č. 15.

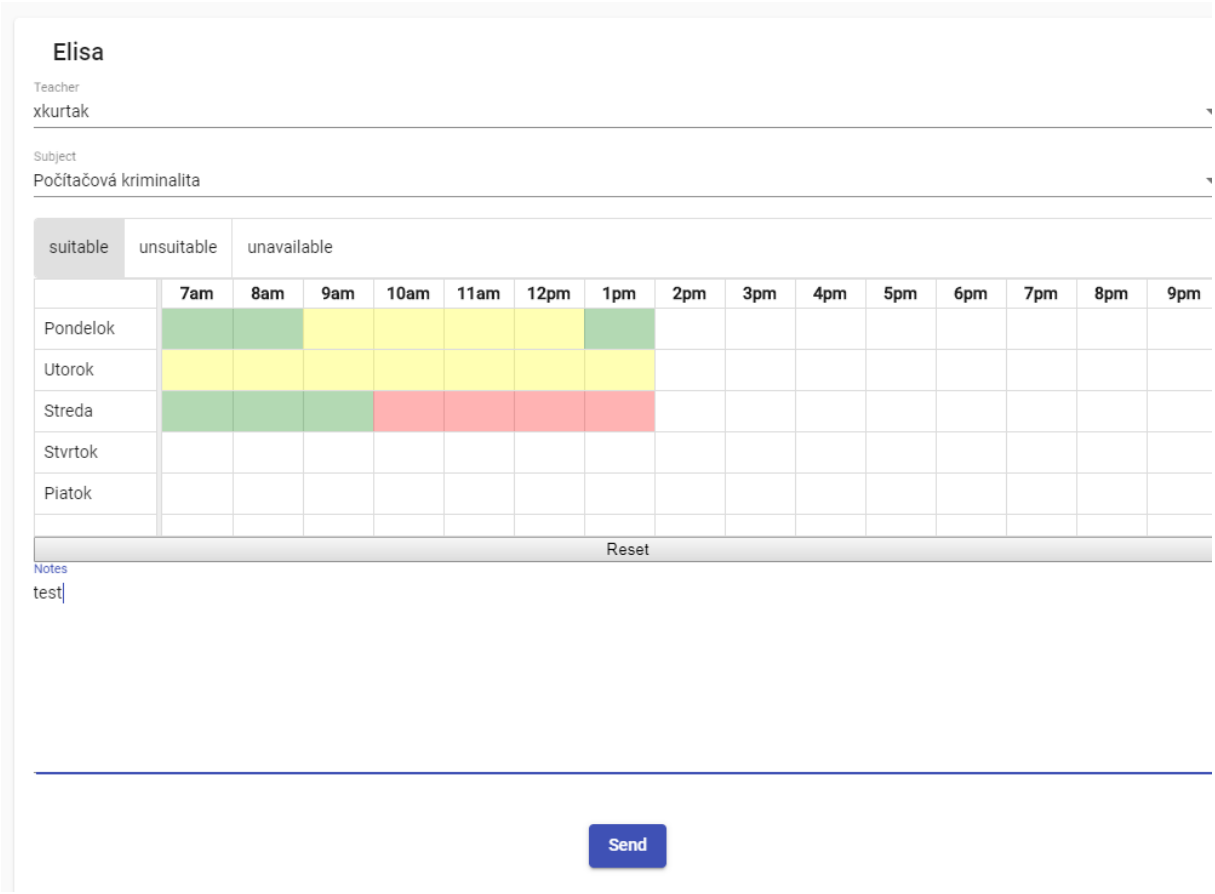


Obr. 14: Vytáranie rozvrhu



Obr. 15: Modálne okno používateľa

Vyučujúci svoje požiadavky vypĺňajú na samostatnej podstránke, pomocou formulára. Na vyplnenie svojich preferovaných, respektíve nevyhovujúcich časov majú k dispozícii vyklikávací rozvrh. Ukážka tohto formulára je zobrazená na obrázku č. 16



Obr. 16: Vypĺňanie požiadavky

Aplikácia je určená pre stolné počítače, alebo notebooky, a to z dôvodu používaných kalendárov, ktoré sú na mobilných zariadeniach ťažko čitateľné. Na rozmiestnenie elementov sme používali Angular Flex-Layout[11] a komponent Angular Material. [19]

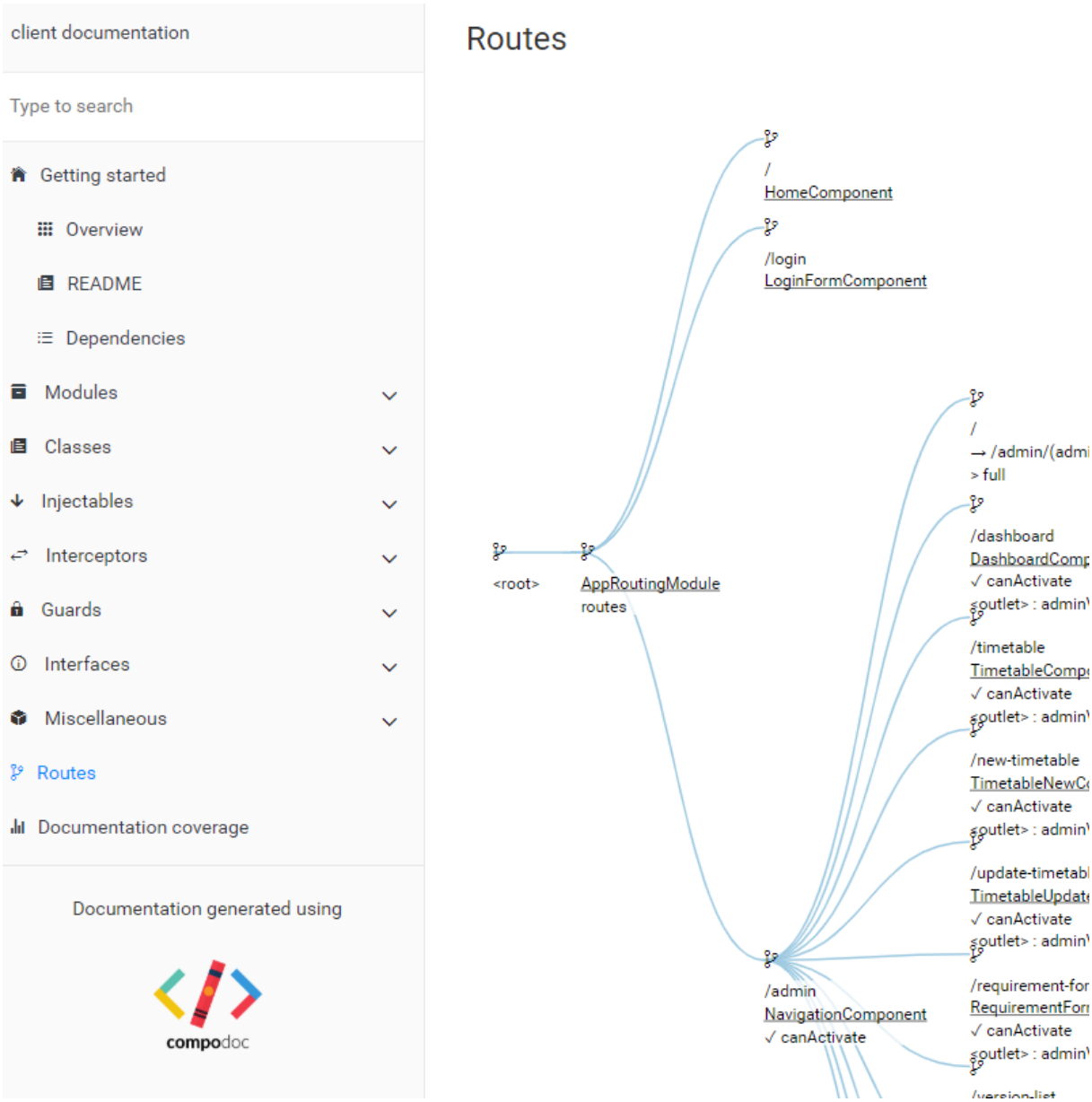
4.5 Dokumentácia

Technická dokumentácia bola vytvorená pomocou dokumentového nástroja Compodoc. [20] Dokumentácia je vygenerovaná ako webová aplikácia. Generovanie prebieha iba pomocou analyzovania zdrojového kódu, ten musí byť vhodne okomentovaný značkovacím jazykom JSDoc tags[12]. Vygenerovaná dokumentácia poskytuje základný prehľad o použitých modeloch a ich komponentoch, grafické zobrazenie týchto komponentov, navigáciu v aplikácii a veľa ďalších pohľadov na aplikáciu. Dokumentácia sa generuje pomocou príkazu:

```
1 $ npm run compodoc
```

[11] https://github.com/angular/flex-layout
[12] https://devdocs.io/jsdoc/

Na obrázku č. 17 je zobrazená ukážka vygenerovanej dokumentácie.



Obr. 17: Compodoc smerovanie

Záver

Úlohou naše práce bolo pokračovať vo vývoji klientskej aplikácie Rozvrhový systém pre FEI. Analyzovali sme proces tvorby rozvrhov a niekoľko existujúcich systémov, ktoré riešia tento proces. Následne sme naštudovali existujúce riešenie Rozvrhového systému pre FEI [2], ktoré bolo zamerane na používateľské rozhranie. Z týchto analýz a s pomocou hlavného rozvrhára pre FEI STU sme definovali požiadavky na systém.

Používateľské rozhranie existujúceho systému nebolo vhodne navrhnuté a implementované, preto sme sa rozhodli o implementácii vlastnej klientskej aplikácie. Navrhli sme používateľské rozhranie doplnené o chýbajúce prvky pôvodnej aplikácie, ako sú stránky na manipuláciu s dátami. Opísali sme používané technológie. Aplikáciu sme vytvárali v súlade s Angular architektúrou. Priblížili sme si vytvorené moduly a implementáciu niektorých špecifických funkcionalít. Následne sme opísali používateľské rozhranie. Na záver sme priblížili ako sa v systéme generuje technická dokumentácia.

Súčasná aplikácia ešte nie je vhodná na nasadenie, aj napriek tomu že sa nám podarilo implementovať niektoré požiadavky. Rozvrhový systém je realizovaný ako spa aplikácia, s ktorou sme sa streli prvý krát. Nie všetky požiadavky boli splnené, medzi patria:

- Import, zisťovanie ekvivalencií,
- dizajn,
- krok späť.

Ďalšie možnosti vývoja sa však otvárajú aj implementovaním ďalších funkcionalít rozvrhového systému, ako je napríklad Rozvrh skúšok.

Zoznam použitej literatúry

1. RAČÁK, Martin. *Rozvrhový systém pre FEI*. 2017. Diplomová práca. Slovenská technická univerzita v Bratislave. EČ: FEI-5384-53920.
2. BEDNÁR, Dávid. *Tvorba uživateľského rozhrania k rozvrhovému systému pre FEI*. 2018. Diplomová práca. Slovenská technická univerzita v Bratislave. EČ: FEI-5384-46049.
3. KNAPEREKOVÁ, Emília. *Rozvrhový systém pre vysoké školy*. 2014. Diplomová práca. Slovenská technická univerzita v Bratislave. EČ: FEI-5384-56111.
4. LUKÁČ, Matej. *Course timetabling at Masaryk University in the UniTime system*. 2013. Dostupné tiež z: `https://is.muni.cz/th/255726/fi_m/Master_Thesis.pdf`. Diplomová práca. Masaryk University.
5. *aSc TimeTables* [online] [cit. 2019-05-11]. Dostupné z: `https://www.asctimetables.com/`.
6. *aSc TimeTables* [online] [cit. 2019-05-11]. Dostupné z: `http://help.asctimetables.com/pdf/asc_timetables_en_P1.pdf`.
7. MÜLLER, Tomáš. *UniTime - University Timetabling* [online] [cit. 2019-05-11]. Dostupné z: `http://www.unitime.org/`.
8. *UniTime* [online] [cit. 2019-05-11]. Dostupné z: `http://www.unitime.org/present/unitime-highlights.pdf`.
9. JONES, M., BRADLEY, J. a SAKIMURA, N. *JSON Web Token (JWT)* [online]. RFC Editor, 2015 [cit. 2019-04-15]. ISSN 2070-1721. Dostupné z: `https://tools.ietf.org/html/rfc7519`. RFC. RFC Editor.
10. FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. University of California, Irvine, 2000. ISBN 0-599-87118-0. Dizertačná práca. AAI9980887.
11. ECMA INTERNATIONAL. *The JSON Data Interchange Syntax* [online]. 2017 [cit. 2019-04-24]. Č. 404. Dostupné z: `https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf`.
12. *Angular 6.0+ calendar*. Dostupné tiež z: `https://mattlewis92.github.io/angular-calendar/#/kitchen-sink`.
13. *FullCalendar*. Dostupné tiež z: `https://fullcalendar.io/`.

14. BIERMAN, Gavin, ABADI, Martín a TORGERSEN, Mads. Understanding TypeScript. In: JONES, Richard (ed.). *ECOOP 2014 – Object-Oriented Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, s. 257–281. ISBN 978-3-662-44202-9.

15. *Angular Documentary* [online] [cit. 2019-05-12]. Dostupné z: `https://angular.io/docs`.

16. *ECMAScript® 2015 Language Specification*. Dostupné tiež z: `http://www.ecma-international.org/ecma-262/6.0/`.

17. *Architecture overview* [online] [cit. 2019-05-16]. Dostupné z: `https://angular.io/guide/architecture`.

18. ZELDMAN, Jeffrey. *Taking Your Talent to the Web: A Guide for the Transitioning Designer*. Thousand Oaks, CA, USA: New Riders Publishing, 2001. ISBN 0735710732.

19. *Angular Material* [online] [cit. 2019-05-15]. Dostupné z: `https://material.angular.io`.

20. *Compodooc* [online] [cit. 2019-05-15]. Dostupné z: `https://compodoc.app/`.

Prílohy

A Technická dokumentácia

Zdrojové súbory k diplomovej práci sú poskytnuté v repoziráty Github[13]. Pre úspešný beh klientskej časti je potrebné ju spúšťať spoločne so serverovou časťou. Z toho dôvodu je potrebné aby systém podporoval Python 3.6 a PostgreSQL 10 alebo vyššia verzia. V prípade generovania dokumentácie je potrebné mať inštalovaný aj Node.js Postup spustenia aplikácie je popísaný v súbore README.md Celý projekt bol vytváraný pomocou Angular CLI, ktorý zabezpečuje aj jeho build.

[13]https://github.com/matusjokay/Elisa

B Štruktúra elektronického nosiča

Štruktúra a prehľad základných súborov a priečinkov.

```
/
|-- thesis                          # diplomová práca
```