VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ANALÝZA A OPTIMALIZACE GRAFICKÉHO UŽIVATELSKÉHO ROZHRANÍ INFORMAČNÍHO SYSTÉMU VUT

ANALYSIS AND OPTIMIZATION OF THE GRAPHICAL USER INTERFACE OF THE BUT INFORMATION SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MATĚJ ŽALMÁNEK**
AUTHOR

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

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

148619

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Žalmánek Matěj**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Analýza a optimalizace grafického uživatelského rozhraní Informačního systému VUT**
Kategorie:	Uživatelská rozhraní
Akademický rok:	2022/23

Zadání:

1. Seznamte se s jazyky JavaScript, PHP a SQL a s technologiemi využívanými při rozvoji webové části centrálního informačního systému VUT.
2. Prostudujte aktuální komponenty uživatelského rozhraní IS VUT a proveďte průzkum jejich uživatelské přívětivosti s ohledem na nejčastější případy použití studenty a vyučujícími.
3. Navrhněte nové komponenty uživatelského rozhraní IS VUT a úpravy stávajících komponent za účelem zvýšení uživatelské přívětivosti systému. Zaměřte se při tom na různé pohledy na rozvrhy.
4. Implementujte navržené řešení.
5. Zhodnoťte dosažené výsledky a vytvořte stručný plakát prezentující výsledky Vaší práce.

Literatura:

- Dle doporučení vedoucího.

Při obhajobě semestrální části projektu je požadováno:
Body 1, 2 a 3.

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

Vedoucí práce:	**Dytrych Jaroslav, Ing., Ph.D.**
Konzultant:	Strakoš Marek, Ing.
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	31.10.2022

Abstrakt

Tato práce se zaměřuje na uživatelskou přívětivost rozvrhů a souvisejících komponent informačního systému VUT s cílem identifikovat málo uživatelsky přívětivé komponenty a ty zoptimalizovat a vylepšit jejich uživatelskou přívětivost. Nejprve jsem vyhledal problematické komponenty a ty se dále staly podnětem dotazníkového šetření. Na základě tohoto dotazníkového šetření jsem provedl návrh nového uživatelského rozhraní, které jsem poté implementoval a testoval s uživateli. V práci se mi podařilo vytvořit řadu nových funkcí uživatelského rozhraní, které uživatelům ulehčují práci s rozvrhy. Přínosem této práce je lepší subjektivní hodnocení rozvrhů uživateli, včetně zkrácení času pro vykonání určitých úkonů s rozvrhy.

Abstract

This thesis focuses on the user-friendliness of the schedules and related components of the BUT information system with the aim of identifying poorly user-friendly components and optimizing and improving their user-friendliness. First, I located problematic components and these further became the impetus for the questionnaire survey. Based on this questionnaire survey, I made a design for a new user interface, which I then implemented and tested with users. In this work, I was able to create a number of new user interface features that make it easier for users to work with the schedules. The benefits of this work include better subjective evaluation of schedules by users, including a reduction in the time to perform certain tasks with schedules.

Klíčová slova

Uživatelské rozhraní, Informační systém VUT, Analýza uživatelského rozhraní, React, Rozvrhy, Uživatelské rozhraní rozvrhů, Javascript, Responzivita, UX, Uživatelská přívětivost

Keywords

User Interface, BUT Information System, User Interface Analysis, React, Schedules, Schedules user interface, Javascript, Responsiveness, UX, User experience

Citace

ŽALMÁNEK, Matěj. *Analýza a optimalizace grafického uživatelského rozhraní Informačního systému VUT*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Dytrych, Ph.D.

Analýza a optimalizace grafického uživatelského rozhraní Informačního systému VUT

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana proděkana Jaroslava Dytrycha. Další informace mi poskytl pan inženýr Marek Strakoš. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Matěj Žalmánek
9. května 2023

Poděkování

Mé poděkování patří především panu proděkanovi Jaroslavu Dytrychovi za jeho rady a pomoc při řešení této práce. Dále také mé poděkování patří panu inženýrovi Marku Strakošovi za jeho pomoc při analýze stávajícího uživatelského rozhraní a komponent informačního systému VUT.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Úspěch většiny aplikací spočívá v jejich uživatelské přívětivosti. Aplikace může mít nespočet vynikajících funkcionalit, ovšem bez snadné použitelnosti jsou všechny tyto funkcionality k ničemu. Navíc v dnešní době asi polovina uživatelů přistupuje k aplikacím z mobilního zařízení, takže je třeba uživatelskou přívětivost řešit s ohledem na mobilní zařízení, stejně jako na počítač.

Tato práce se zaměřuje na uživatelské rozhraní informačního systému VUT, konkrétně uživatelské rozhraní rozvrhů a souvisejících komponent. Podnětem pro tuto práci byly časté negativní názory uživatelů na rozvrhy v tomto informačním systému. Současné řešení obsahuje řadu zmatečných komponent. Často jsou zmatečné i samotné rozvrhové jednotky ve vykresleném rozvrhu. Navíc orientace v rozvrhu není vždy dobrá a použitelnost z mobilního zařízení je velmi omezená.

Cílem práce je tedy vytvořit uživatelské rozhraní rozvrhů, které bude snadno použitelné ze všech druhů zařízení. Dále bude snadné se v tomto rozhraní orientovat a používat ho s co nejmenším potřebným úsilím.

V kapitole 2 je popsán současný stav uživatelského rozhraní rozvrhů a souvisejících komponent. Poté je zde popsaný způsob výběru prvků uživatelského rozhraní, které se staly předmětem následného dotazníkového šetření. Dále se tato kapitola zmiňuje o způsobu výběru otázek pro dotazníky a nakonec se zabývá zpracováním a vyhodnocením dat z těchto dotazníků. Další kapitola 3 je zaměřena na návrh nového uživatelského rozhraní, který je založen na výstupech z dotazníkového šetření. Kromě návrhu rozhraní se zde popisuje také návrh formátu přenosu dat mezi klientskou a serverovou částí aplikace a návrh samotné architektury aplikace. Následující kapitola 4 popisuje nástroje a technologie použité v této práci a dává důležitý kontext pro čtení dalších kapitol. Další kapitola 5 se zaměřuje na vlastní implementaci navrženého rozhraní, včetně řešení vzniklých komplikací při implementaci. V této kapitole jsou popsány také implementace databázových dotazů, včetně optimalizace dotazů z původního řešení. Předposlední kapitola 6 zobrazuje postupy použité při testování aplikace. Součástí této kapitoly je i výsledek testování uživatelské přívětivosti nového rozhraní. Poslední kapitola 7 obsahuje závěrečné shrnutí výsledků této práce.

Kapitola 2

Analýza stávajícího řešení uživatelského rozhraní

Při analyzování uživatelského rozhraní informačního systému VUT jsem se v rámci této práce rozhodl vylepšit uživatelskou přívětivost rozvrhů a komponenty, které s rozvrhy souvisejí. Tomuto výběru také pomohla častá nespokojenost uživatelů s rozvrhy a jejich stížnosti.

2.1 Rozvrhy v informačním systému VUT

Při důkladném studiu původního řešení bylo třeba postupovat systematicky, modul od modulu, jelikož rozvrhy byly ve více modulech a každá fakulta VUT může používat jednotlivé moduly odlišně.

Konkrétně se rozvrhy využívají v následujících modulech informačního systému VUT: Individuální rozvrh studenta, Individuální rozvrh vyučujícího, Rozvrh místností, Hromadný rozvrh místností, Rozvrh skupin, Rozvrh přednáškových skupin, Registrace vyučování, Rozvrh předmětů, Rezervace místností, Výuka zapsaných studentů v předmětu.

Je tedy patrné, že agenda s rozvrhy je velice rozsáhlá a je třeba k její analýze přistupovat velice zodpovědně.

Tato práce se zaměřovala na optimalizaci rozvrhů ve všech výše uvedených modulech s výjimkou registrace vyučování a rezervace místností, které jsou už velmi specifické.

2.2 Odhad hlavních problémů původního řešení

Před samotnou analýzou stávajícího řešení zpracování rozvrhů v informačním systému VUT bylo třeba nejprve vytipovat pravděpodobně nejméně uživatelsky přívětivé komponenty, které by bylo vhodné dále analyzovat, studovat a po konzultaci s uživateli dále upravovat.

Individuální rozvrh

Při analýze výchozí varianty rozvrhu – **standardního rozvrhu** jsem na **počítači** odhalil následující části uživatelského rozhraní, které jsou dle mého názoru problematické.

- **Přeškrtnutí rozvrhového okna** – Toto přeškrtnutí značí zrušenou výuku v jednom dni v semestru (nebo i více, maximálně pak n - 1 zrušených výuk, pokud n značí celkový počet zobrazovaných výuk daného typu). V člověku však může toto přeškrtnutí

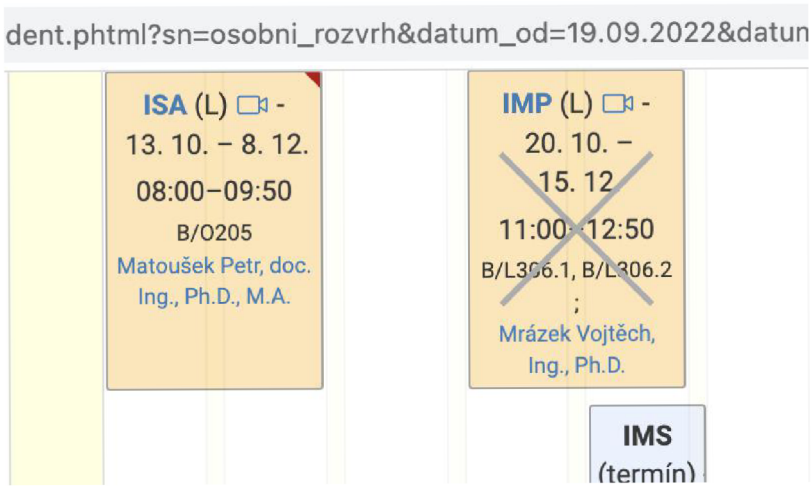
Obrázek 2.1: Nejasné přeškrtnutí rozvrhové jednotky

implikovat dojem, že byla výuka zrušena úplně (tedy n výuk daného typu). Toto je zobrazeno na obrázku 2.1.

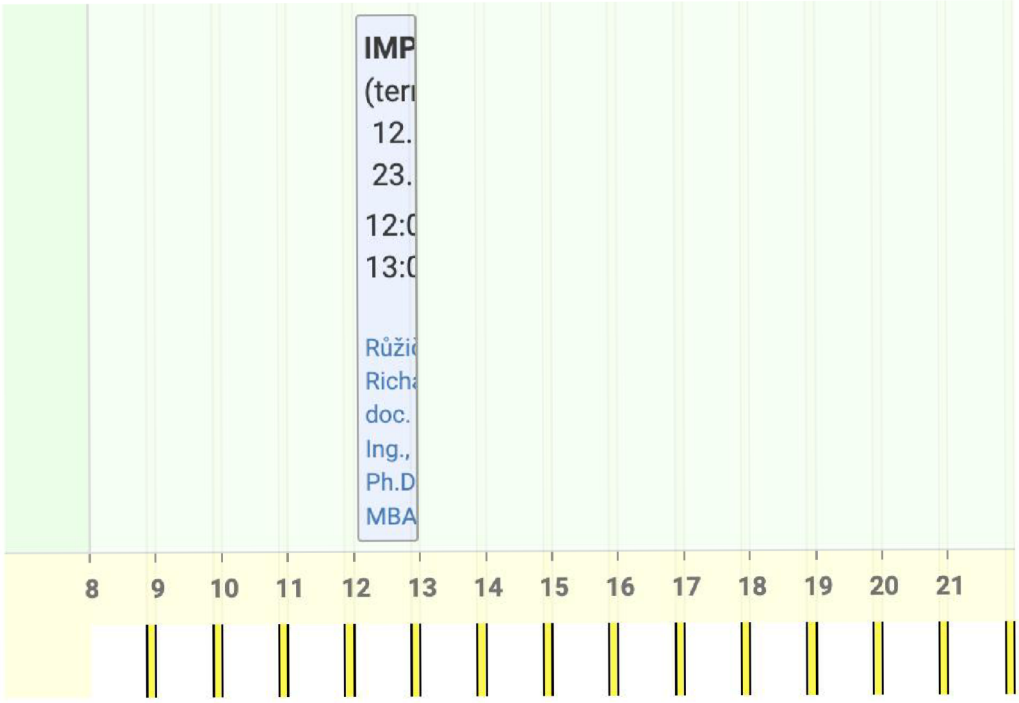
- **Výška rozvrhových jednotek** – Myslím si, že výška rozvrhových jednotek je příliš vysoká, a to zejména u rozvrhových jednotek, které trvají pouze 1 hodinu a jsou tedy užší. Při úvaze, jak tento problém řešit, se mi zdálo, že se v rozvrhové jednotce zobrazují nepodstatné informace, které tuto jednotku mohou zvyšovat, například: Zkratka typu výuky, titul vyučujícího, čas konání od – do rozvrhové jednotky, datum konání rozvrhové jednotky.
- **Nefunkční rozhraní po změně typu rozvrhu** – Při testování tohoto typu rozvrhu si bylo možné všimnout, že po prvním načtení rozvrhu se zobrazovala vyskakovací okna s detailem rozvrhového okna. Ovšem při změně typu rozvrhu na týdenní se tento detail již nezobrazoval, a to ani po vrácení zpět na standardní rozvrh.
- **Neukotvení záhlaví rozvrhů** – Při používání rozvrhu jsem narazil na problém se záhlavím rozvrhu s vypsanými hodinami, kdy se koná výuka. Konkrétně při posunu po stránce směrem dolů záhlaví nezůstalo ukotveno v horním rohu, ale zůstalo na původní pozici, čímž se skrylo. Toto demonstruje obrázek 2.2.
- **Chybějící znázornění aktuálního času** – Pro rychlejší orientaci v rozvrhu by dle mého názoru bylo vhodné použít nějakou formu vizualizace aktuálního času na ose x rozvrhu.

Při analýze tohoto typu rozvrhu na **mobilním zařízení** jsem narazil na následující části rozhraní, které si myslím, že jsou málo uživatelsky přívětivé.

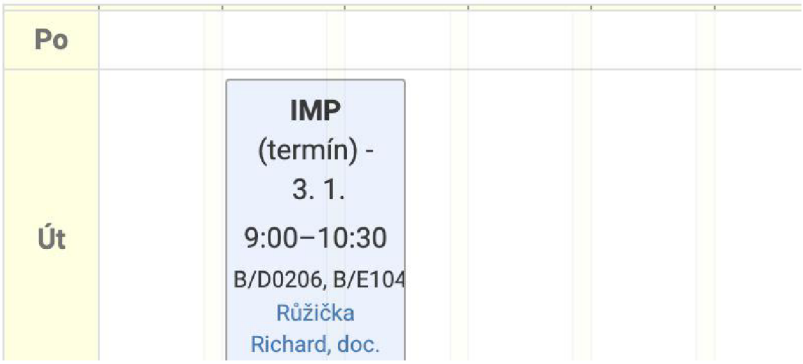
- **Šířka rozvrhových jednotek** – Při analýze jsem nabyl dojmu, že rozvrhové jednotky jsou příliš úzké, což může způsobovat problém s rozkliknutím jednotky.
- **Chyby vykreslování rozvrhu** – Špatně vykreslená spodní osa s hodinami, kdy se konají rozvrhová okna, jak je ilustrováno na obrázku 2.3.
- **Smazané záhlaví s hodinami** – Při změně velikosti okna se záhlaví s hodinami, kdy se koná výuka, úplně smazalo, jak lze vidět na obrázku 2.4.



Obrázek 2.2: Neukotvené záhlaví s vypsanými hodinami, kdy se koná výuka



Obrázek 2.3: Chyba vykreslování osy s hodinami na mobilním zařízení



Obrázek 2.4: Smazané záhlaví s hodinami

Obrázek 2.5: Komponenta pro filtrování obsahující podivné chování

Nakonec si myslím, že jako výchozí rozvrh je použita nesprávná varianta (standardní rozvrh) a že by výchozím rozvrhem měl být týdenní rozvrh. V této variantě rozvrhu jsem odhalil následující problém.

- **Chyba filtru** – Po kratší době si šlo všimnout, že filtr na výběr týdne s posunem o jedna vzad se přesune o jeden rok a dále se chová velmi zvláštně. Komponenta, která se takto chová je na obrázku 2.5.

Rozvrh místností

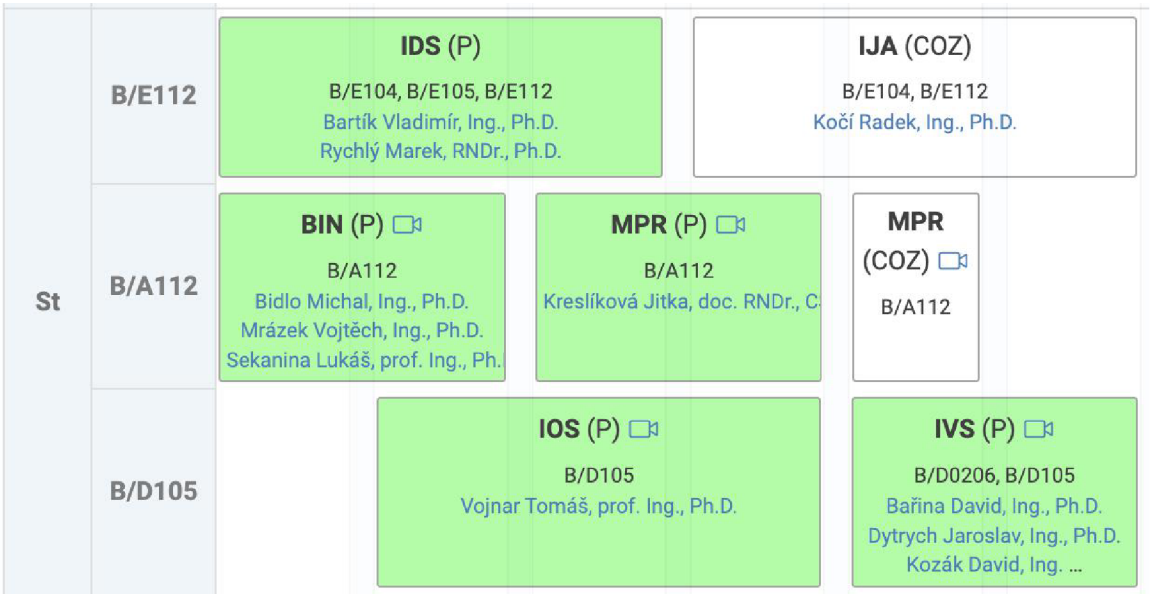
U rozvrhu místností byly dle mého názoru problematické podobné položky jako u individuálního rozvrhu. Tedy zobrazovaná data v jednotce, křížky přes rozvrhovou jednotku, nevhodný výchozí typ rozvrhu, neukotvené záhlaví s časem konání rozvrhových jednotek.

Navíc si bylo možné všimnout matoucí šedé čáry, která vypadala jako chyba při vykreslování a v uživateli dle mého názoru může vyvolat dojem nečistoty nebo šmouhy na displeji a tím zpomalí prohlížení rozvrhu. Po zkoumání kódu tohoto rozvrhového okna jsem zjistil, že se podle třídy elementu zobrazující tuto čáru jedná o nedokonale vykreslený křížek zrušené rozvrhové jednotky. Toto je vidět na obrázku 2.6.

Hromadný rozvrh místností

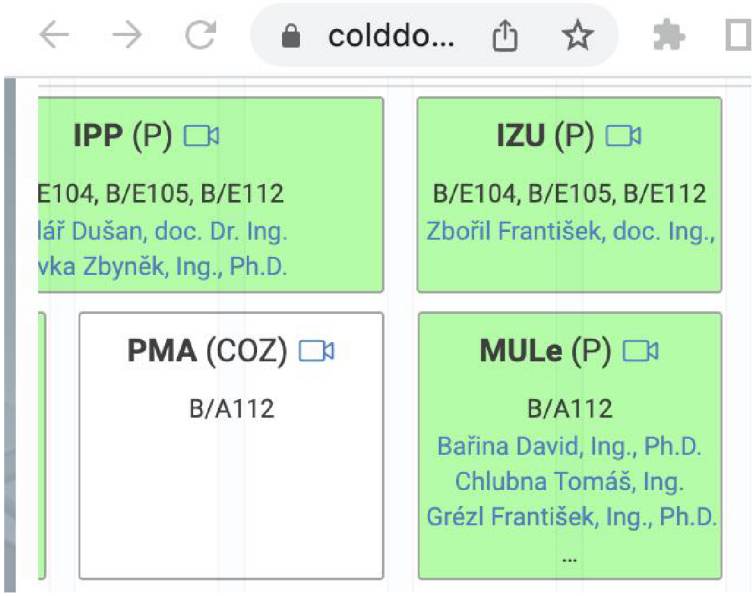
U hromadného rozvrhu místností jsem kromě již zmiňovaných problémů identifikoval jako problematické následující položky.

Obrázek 2.6: Šedá čára v rozvrhové jednotce



Obrázek 2.7: Neoddělení rozvrhů jednotlivých místností

- **Neoddělení rozvrhů jednotlivých místností** – Při procházení rozvrhů místností chybělo dle mého názoru oddělení jednotlivých místností čárami, aby se zaručila lepší orientace v rozvrhu. Neoddělení jednotlivých místností je zobrazeno na obrázku 2.7.
- **Neukotvení záhlaví sloupců** – Při zúžení okna prohlížeče se rozvrh nevešel na šířku displeje a musel se zobrazit vodorovný posuvník. Ovšem při vodorovném posunu v rozvrhu se neukotvilo záhlaví s popisem dnů a místností a tak se rozvrh stal hůře čitelným. Na obrázku 2.8 lze vidět, že při použití obou posuvníků (vodorovného i svislého) se neukotvilo ani jedno záhlaví, čímž se rozvrh stal prakticky nečitelným.



Obrázek 2.8: Neukotvení ani jednoho záhlaví a s tím související špatná čitelnost rozvrhu

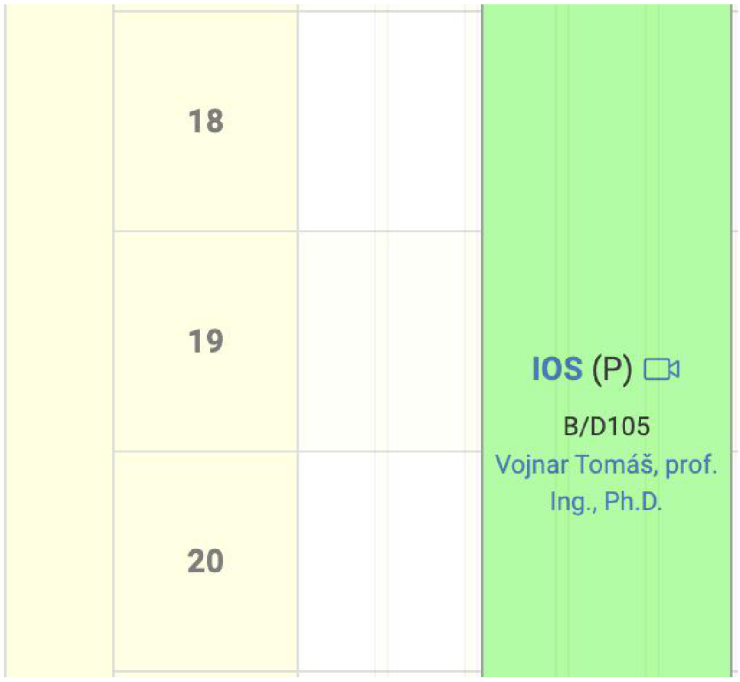
Rozvrh skupin, rozvrh předmětů

U těchto dvou druhů rozvrhů jsem neobjevil kromě již zmíněných problémů u ostatních rozvrhů, jako například zobrazovaná data v jednotce, křížky přes rozvrhovou jednotku, neukotvené záhlaví a chyby při vykreslování, žádné další problémy.

Rozvrh přednáškových skupin

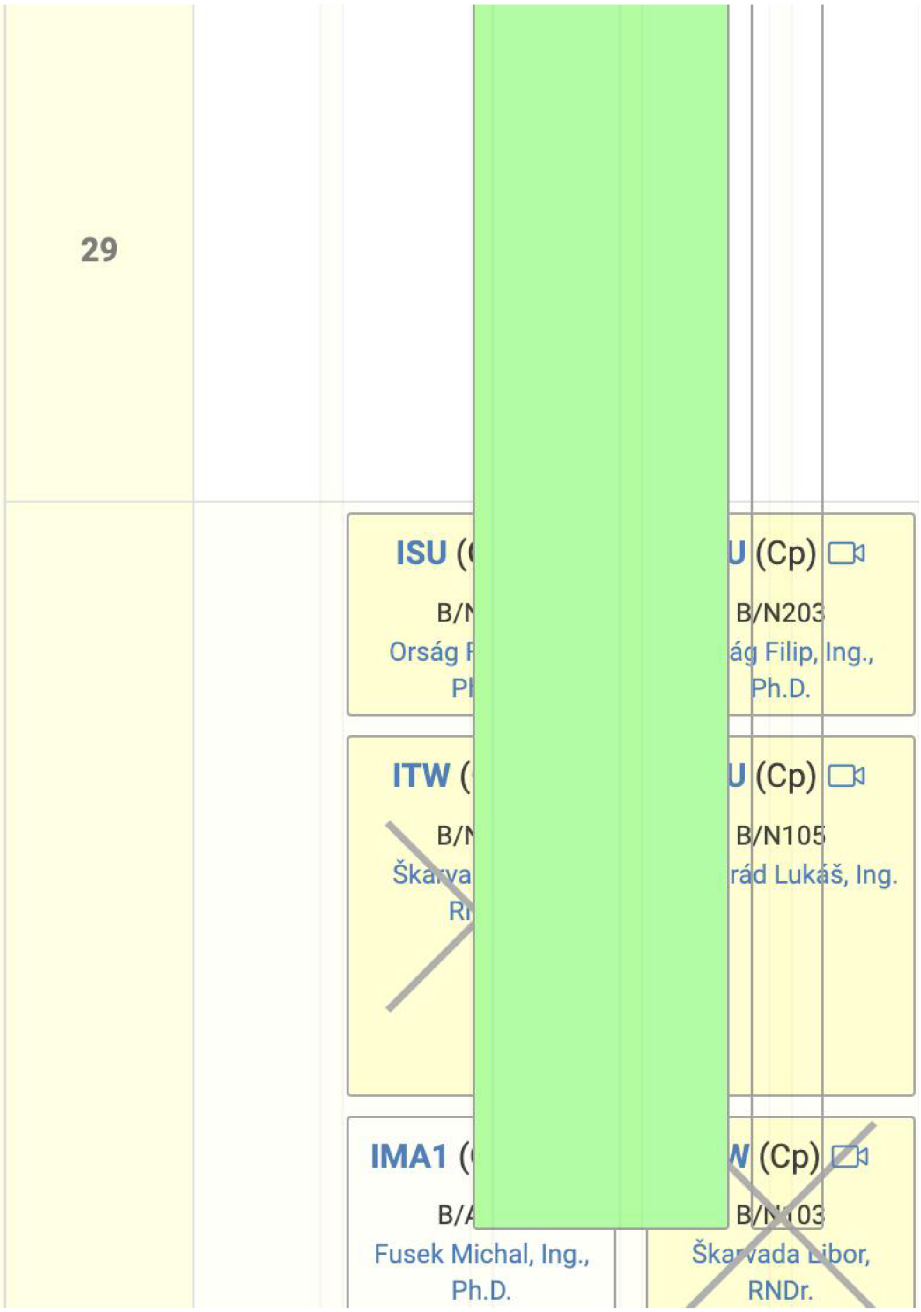
U rozvrhu přednáškových skupin se zdálo být problematických hned několik věcí.

- **Nesjednocení stejných rozvrhů více přednáškových skupin** – Kvůli stejnému rozvrhu několika přednáškových skupin, se rozvrh roztáhl na výšku, protože každá přednášková skupina se vykreslila zvlášť, místo toho, aby byla snaha skupiny agregovat. Toto lze vidět na obrázku 2.9.
- **Chybné chování při změnách velikosti okna** – Po změně velikosti okna se celá rozvrhová okna podivně posouvala po ose x a podivným způsobem se měnila jejich výška. Toto je zobrazeno na obrázcích 2.9, 2.10, 2.11, kde se postupně měnila velikost okna prohlížeče.
- **Neukotvení textu v záhlaví sloupců** – Také se zde ve větší míře projevilo neukotvení textu s dnem konání rozvrhových jednotek, což při posuvu posuvníku směrem dolů způsobilo, že nebylo vidět, o jaký den se jedná, viz obrázek 2.9.

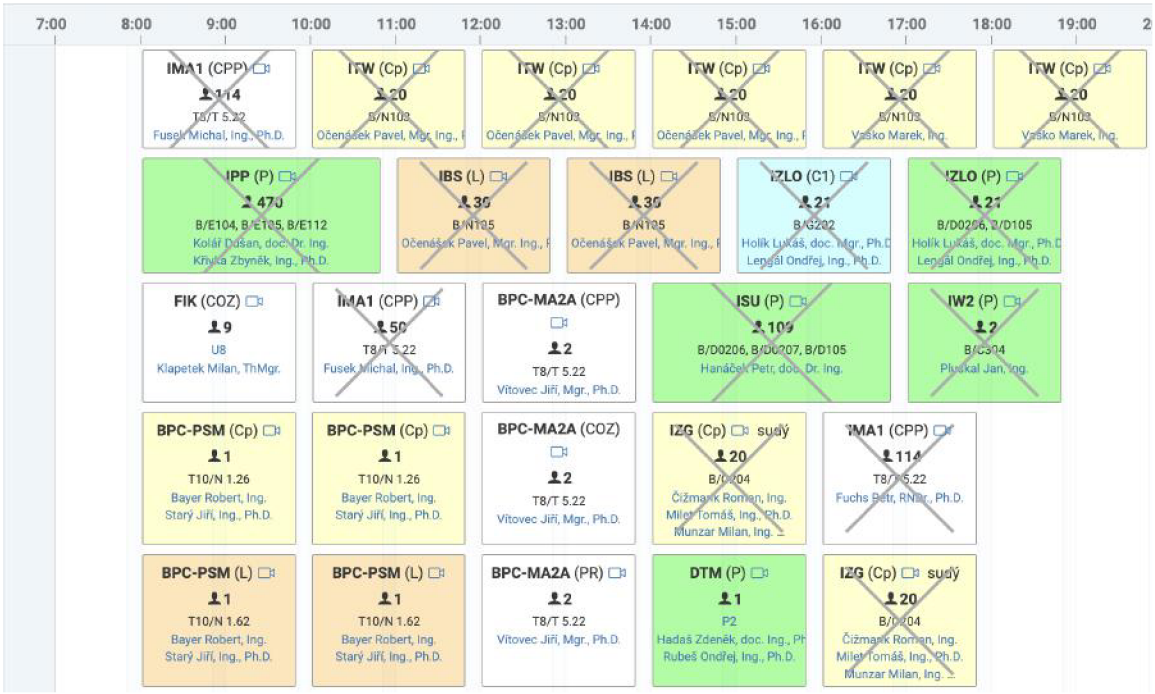


Obrázek 2.9: Rozvrh před změnou velikosti okna prohlížeče, nemá ukotvený text s popisem dne, ani agregované skupiny 18, 19, 20 se stejným rozvrhem

Obrázek 2.10: Rozvrh po změně velikosti okna prohlížeče – jiná pozice v ose x



Obrázek 2.11: Rozvrh po další změně velikosti okna prohlížeče – změna výšky rozvrhových jednotek



Obrázek 2.12: Rozvrh obsahující příliš velké množství dat

Rozvrh Výuka zapsaných studentů v předmětu

U výuky zapsaných studentů se zase velmi projevilo neukotvení textu s dnem konání rozvrhových jednotek.

Taky se na první pohled zdál rozvrh nepřehledný kvůli enormně velkému množství dat, což lze řešit vhodně vybranou filtrací, viz obrázek 2.12.

2.3 Analýza nejčastěji používaných prohlížečů a zařízení uživatelů z cílové skupiny

Díky přístupu k datům z Google Analytics informačního systému VUT bylo možno snadno definovat prohlížeče, které nové rozhraní bude muset podporovat, a minimální podporovanou šířku displeje.

V tabulce 2.1 lze vidět zastoupení nejpoužívanějších webových prohlížečů při přístupu na webové rozhraní informačního systému VUT (data z období od 1. října 2022 do 16. března 2023). Z těchto dat bylo dále rozhodnuto, že funkčnost bude testována a garantována pouze na prohlížečích Chrome, Safari, Firefox, Android Webview, Edge a Opera. Rozhraní na ostatní prohlížeče z důvodu nízké používanosti nebude garantováno.

Tabulka 2.2 obsahuje data používaných rozlišení při přístupu na webové rozhraní IS VUT (data z období od 1. října 2022 do 16. března 2023). Podle těchto dat bylo rozhodnuto, že minimální podporované rozlišení bude 360px (Tabulka neobsahuje rozlišení se zastoupením pod 0,5 %, kromě rozlišení pod 360px).

Prohlížeč	Počet ustavených sezení	Zastoupení [%]
Chrome	1 984 169	53,12
Safari	572 114	15,32
Firefox	266 314	7,13
Safari (in-app)	248 564	6,65
Android Webview	238 054	6,37
Edge	229 867	6,15
Opera	155 196	4,15
Samsung internet	28 084	0,75
Seznam	3 378	0,09
Internet explorer	3 001	0,08

Tabulka 2.1: Přehled nejpoužívanějších prohlížečů při používání informačního systému VUT

Šířka displeje [px]	Počet ustavených sezení	Zastoupení [%]
3440	12519	0,34
2560	113152	3,03
2048	28636	0,77
1920	847973	22,7
1707	27777	0,74
1680	46838	1,25
1600	28415	0,76
1536	643269	17,22
1440	120486	3,23
1366	72762	1,95
1280	165233	4,42
820	19780	0,53
412	307715	8,24
393	242869	6,5
390	171578	4,59
384	57946	1,55
375	310404	8,31
360	246502	6,6
339	547	0,01
320	15212	0,41

Tabulka 2.2: Přehled zastoupení šířky displeje zařízení při přístupu k informačnímu systému VUT

2.4 Dotazníkové šetření

Po vyhledání dle mého názoru problematických komponent uživatelského rozhraní rozvrhů bylo potřeba si svůj názor potvrdit nebo vyvrátit v dotazníkovém šetření.

Nástroj pro tvorbu dotazníku

Při hledání vhodného nástroje pro tvorbu dotazníku byla vhodnost nástroje posuzována podle následujících kritérií:

- Rychlé a jednoduché vytvoření dotazníku
- Možnost omezení respondentů pouze na určitou doménu
- Jednoduché šíření dotazníku
- Snadný sběr dat v reálném čase a přehlednost statistik
- Dostatečné možnosti při tvorbě otázek uvnitř dotazníku
- Cena nástroje
- Zkušenosti s daným nástrojem
- Možnost omezení počtu odpovědí od jednoho respondenta

Po prohledávání internetu byly nalezeny následující nástroje:

- **Microsoft Forms** – přehledné statistiky, možnost omezení přístupu k dotazníku, možnost větvení otázek
- **Google Forms** – větvení otázek, možnost omezit přístup k dotazníku, snadnost, jednoduchost, přehledné statistiky
- **Typeform** – možnost větvení otázek, neobsahuje nebo nelze snadno nalézt možnost omezit přístup k dotazníku, na první pohled méně přehledné statistiky oproti ostatním nástrojům, rozšířené možnosti jsou zpoplatněny
- **Survio** – možnost omezení počtu odpovědí z jednoho zařízení, přehledné statistiky, možnost větvení otázek je zpoplatněna
- **NPM balíček survey-react** – možnost tvorby libovolných otázek s vlastním kódem v jazycích HTML a CSS, ale toto řešení je časově náročné

Z výše zmíněných nástrojů se nakonec rozhodovalo mezi Microsoft Forms a Google Forms, neboť obě tyto možnosti splňovaly všechna kritéria. Ovšem z důvodu osobních preferencí a zkušeností s Google Forms jsem se rozhodl využít právě Google Forms, aby se dotazník dal co nejrychleji publikovat.

Koncepce otázek

Při tvorbě otázek jsem se snažil působit nezaujatě, neutrálně. Tedy při otázkách na stávající řešení jsem se snažil tázat tak, abych nijak respondenta neovlivňoval. Zároveň pořadí otázek bylo směrováno tak, aby se respondent nejprve zamyslel nad zadaným problémem konkrétně a až v závěru dotazníku odpovídal obecně. Tedy například pokud by byla otázka: „Ohodnoťte dosavadní zpracování zobrazení rozvrhů na počítači známkou 1-10“ na začátku dotazníku, mohlo by dojít k situaci, kdy by se uživatel ještě pořádně nezamyslel nad problematikou a odpověděl by jen na základě chybných prvotních dojmů.

Dále bylo důležité určit, zda odpověď na danou otázku bude otevřená, nebo uzavřená. Uzavřené odpovědi sice ulehčí vyhodnocování, ale naopak znemožní respondentovi zadat možnost, kterou dotazník mohl opomenout. Toto bylo řešeno za pomoci uzavřených otázek a na konci dotazníku byly poslední otázky otevřené pro zapsání vlastních pocitů / dojmů. Také u některých uzavřených otázek byla možnost „Jiné“ pro uvedení odpovědi, která nebyla nabízena.

Nakonec byla v rámci dotazníku snaha o rychlé a snadné vyplnění dotazníku respondentem, aby respondent nebyl z důvodu vysoké časové náročnosti nucen k přeskakování otázek nebo ještě hůře k náhodnému vyplnění některých otázek. Toto se řešilo především větvením a přeskakováním otázek podle odpovědi.

Více o problematice, jak psát dotazníky, lze vyčíst z [3], ze kterého jsem čerpal i v rámci tvorby dotazníku v této práci.

Tvorba otázek

Při tvorbě otázek jsem se snažil ověřit nebo vyvrátit předem definované potencionální problémy v uživatelském rozhraní popsané v sekci 2.2. Jelikož byly dvě cílové skupiny respondentů, tak byly také vytvořeny dva dotazníky, kdy se každý šířil k jiné cílové skupině. Oba dotazníky měly podobné otázky, ale lišily se zejména v otázkách ke zjištění informací o respondentovi.

Oba dotazníky měly následující podobu:

- Otázky pro získání informací o respondentovi
- Otázky k filtraci dat v rozvrhu
- Otázky k dostupným modulům s rozvrhy a jejich možnému tisku / exportu do formátu iCal (Formát je dále popsaný v sekci 4.7)
- Otázky zaměřené na vložení vlastního názoru a zadání hodnocení původního řešení

Otázky k jednotlivým modulům s rozvrhy se zabývaly především důležitostí jednotlivých informací v rozvrhové jednotce jako například příjmení, jméno a titul vyučujícího, místnost, ve které se vyučuje, zkratka, název předmětu, typ výuky a další. Díky těmto otázkám, pokud by se zjistilo, že nějaká informace je v rozvrhové jednotce zbytečná, by se mohla snížit šířka jednotlivých rozvrhových oken a tím i jejich výška při přetékání textu.

Dále se dotazník zaměřoval na názor respondentů ohledně optimálního způsobu zobrazení detailu rozvrhového okna v daných modulech. Také pro odhalení, jaký modul se používá z jakého zařízení nejčastěji, se objevovaly otázky na frekvenci používání modulu z počítače / mobilního zařízení. Otázky k jednotlivým modulům se sice opakovaly, ale toto bylo třeba, neboť každý modul má jiný případ užití a tím i jiné odpovědi na výše zmiňované otázky ohledně modulů.

Kanály pro šíření dotazníku

Při volbě kanálů pro šíření dotazníku bylo třeba rozlišovat dvě skupiny cílových respondentů, a to konkrétně:

- **Vyučující a zaměstnanci**
- **Studenti**

Pro šíření mezi studenty fakulty FIT byl zvolen Discord server. Mezi studenty ostatních fakult se dotazník šířil skrze facebookové skupiny kolejí, na kterých jsou ubytovaní studenti VUT. Nakonec se dotazník šířil přímo z modulů s rozvrhy v informačním systému VUT.

Mezi vyučující a zaměstnance se dotazník šířil také z modulů s rozvrhy a navíc skrze fakultní integrátory, kteří e-mailem přeposlali dotazník respondentům, kteří by dle nich byli ochotni dotazník vyplnit.

2.5 Analýza a zpracování dat z dotazníků

Po spuštění propagace dotazníků následoval sběr dat a následné zpracování těchto dat. Z důvodů, které budou popsány níže, jsem se rozhodl pro zpracování dat napsat vlastní skript v jazyce Python, i když jsem již měl data zpracovaná přímo v aplikaci Google Forms.

Google Forms

Sběr dat byl celý v režii aplikace Google Forms. Díky této aplikaci bylo prvotní zpracování dat velice jednoduché, protože aplikace automaticky sama data zpracovala a zobrazila v podobě grafů. Dohromady mezi studenty bylo nasbíráno 739 odpovědí a mezi vyučujícími a zaměstnanci 87 odpovědí.

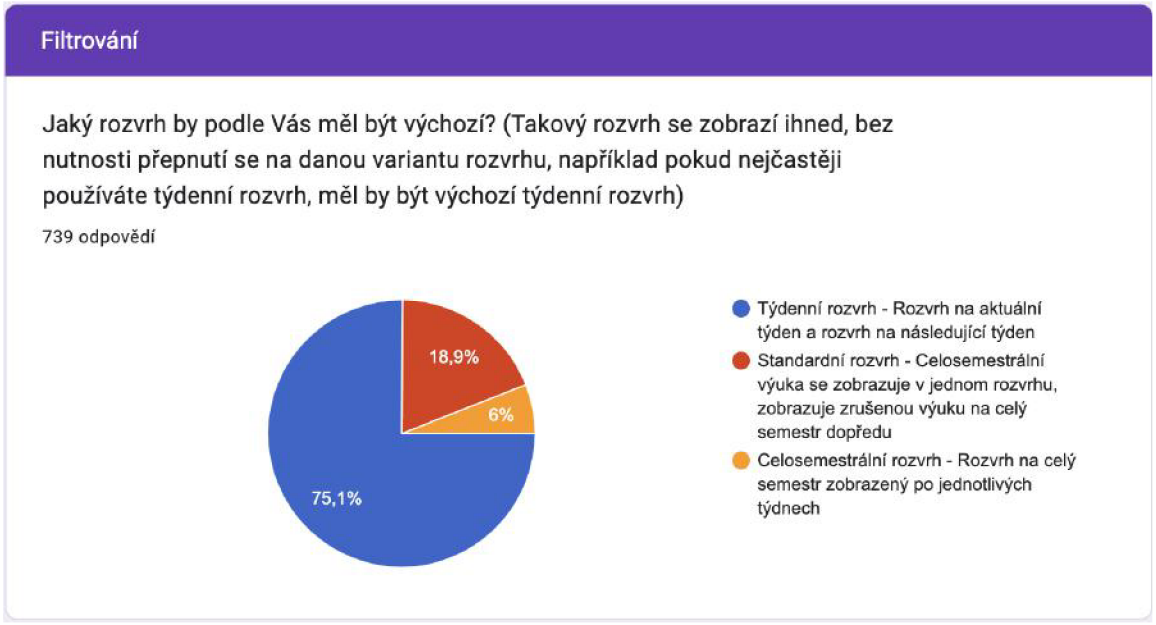
Na grafu 2.13, který vygenerovala aplikace Google Forms, lze vidět, že dle studentů by výchozím rozvrhem měl být týdenní rozvrh.

Skript v jazyce Python

Výstupy v aplikaci Google Forms byly sice přehledné, ovšem z důvodu, že byla snaha odpovědi také roztřídit podle fakult, vzdělání nebo pracovní pozice, jsem se rozhodl, že data vyexportuji ve formátu CSV a dále zpracuji v jazyce Python.

Díky vlastní implementaci vyhodnocení dat bylo možné nad daty provádět nejrůznější operace a rozšířit tak informace získané z grafů zobrazených přímo v aplikaci Google Forms. Například jsem nad daty počítal následující statistické funkce:

- **Aritmetický průměr** – Jelikož aritmetický průměr je velice citlivý na krajní hodnoty, kterými se může velmi vychýlit a stát se nevhodnou statistikou, byl aritmetický průměr při posuzování dat využit především k ucelení informací o souboru dat.
- **Směrodatná odchylka** – Směrodatná odchylka udává jak moc jsou prvky v souboru dat rozptýlené od střední hodnoty. Tato hodnota byla použita především k určení váhy aritmetického průměru při vyhodnocování dotazníku k vyvození nějakého závěru a k získání další informace o souboru dat.
- **Medián** – Medián udává prostřední hodnotu v seřazeném souboru dat relací menší nebo rovno. V případě sudého počtu prvků v souboru dat se jedná o aritmetický



Obrázek 2.13: Graf vygenerovaný aplikací Google Forms zobrazující výsledek otázky zabývající se volbou výchozího rozvrhu

průměr dvou prostředních hodnot. Tato statistika se zdála být nejvhodnější pro co nejpřesnější vyhodnocení většiny otázek v dotazníku, neboť na rozdíl od aritmetického průměru není ovlivněn krajními hodnotami.

- **Modus** – Modus udává prvek s největší relativní četností v souboru dat. Spolu s mediánem na tuto statistiku byl brán největší zřetel při vyhodnocování a vyvozování závěrů znovu z důvodu, že není ovlivněn krajními hodnotami.

2.6 Vyhodnocení dat z dotazníků

Na základě grafů a vypočtených statistik byly vyvozeny následující závěry:

Barva rozvrhových jednotek

Vyučující i studenti dle výsledků z dotazníků považují barvu rozvrhových jednotek za důležitou. Zkratky typu výuky u rozvrhových jednotek považují v určitých případech také za důležité, ale více se orientují podle barev. Tudíž bylo navrženo, že zkratky typu výuky u rozvrhových jednotek se přesunou do detailu o rozvrhové jednotce, a jelikož některé typy výuky neměly přiřazenou barvu, těmto typům výuky se barvy přiřadí.

Tisk

Po vyhodnocení dat z dotazníků bylo zjištěno, že dohromady, skrze všechny fakulty VUT, zhruba 15 % uživatelů používá tisk rozvrhu. Sice se v rámci jednotlivých modulů používanost tisku liší, závěrem však zůstává, že tisk musí být stále podporovaný.

Export do iCal

Export do formátu iCal používá dohromady napříč všemi fakultami VUT asi 18 % uživatelů. Zároveň však bylo zjištěno, že někteří uživatelé ani nevěděli, že v informačním systému taková možnost vůbec je, jinak by ji používali také. Export tedy nadále musí zůstat podporovaný.

Filtrace

Asi 75 % všech respondentů odpovědělo, že by chtělo filtraci podle typu výuky a další rozšířenou filtraci. Z tohoto vyplývá, že v rámci této práce je nutné klást velký důraz na tuto filtraci.

Způsob zobrazení okna s podrobnějšími informacemi

Respondenti v dotazníku vybírali z následujících možností, jak se má zobrazovat okno s podrobnějšími informacemi.

- Po najetí myší na rozvrhovou jednotku
- Po kliknutí na rozvrhovou jednotku
- Chci mít možnost si změnit nastavení způsobu zobrazování tohoto okna na webu, někdy se mi to hodí více po kliknutí, jindy po najetí myší

Jelikož byly mnohdy odpovědi nerozhodné, rozhodl jsem se implementovat možnost přepnutí způsobu zobrazení detailu rozvrhového okna s nastaveným výchozím způsobem zobrazení pro jednotlivé moduly. Statistiky odpovědí na tuto otázku jsou zobrazeny v příloze A, konkrétně v tabulce A.1.

Odkazy v rozvrhové jednotce

Jelikož bylo podezření, že odkazy v rozvrhové jednotce mohou při nechtěném kliknutí na daný odkaz znepříjemňovat používání rozvrhů, dotazník se zaměřoval i na toto téma.

Výsledky odpovědí na tyto otázky jsou zobrazeny v příloze A, konkrétně v tabulkách A.2, A.3.

Důležitost informací o rozvrhové jednotce

Statistiky použité pro vyhodnocení odpovědí na tyto otázky v jednotlivých modulech jsou znázorněny v příloze A, konkrétně tabulky A.4, A.5, A.6, A.7, A.8, A.9. Součástí těchto tabulek je také závěr plynoucí z těchto statistik, kterým se dále řídil návrh a implementace uživatelského rozhraní.

Kapitola 3

Návrh nového řešení agendy s rozvrhy

Pro navržení nového uživatelského rozhraní jsem nejprve navrhl samotné komponenty rozhraní a jejich rozložení na stránce. Poté jsem navrhl strukturu klientské a serverové části aplikace, včetně formátu pro přenos dat mezi nimi.

3.1 Návrh uživatelského rozhraní

Při tvorbě návrhu nového uživatelského rozhraní rozvrhů jsem využíval informace zjištěné z kapitoly 2. Pro zaměření se na podstatné části uživatelského rozhraní jsem se snažil ignorovat vzhled aplikace a zaměřit se pouze na to podstatné – rozložení prvků na stránce a volba vhodných komponent uživatelského rozhraní. Tohoto bylo docíleno skrze tzv. **wireframy**.

Výběr typu rozvrhu

Z důvodu, že některé prvky uživatelského rozhraní se zdály být dostatečně uživatelsky přívětivé, tyto prvky v rozhraní zůstaly v téměř původní podobě. Příkladem je například přepínání mezi typy rozvrhů (Standardní, týdenní, celosemestrální) včetně jejich umístění na stránce. Wireframe této komponenty lze vidět na obrázku 3.1.

Obrázek 3.1: Wireframe rozhraní pro výběr typu rozvrhu

Filtrace

Dále bylo třeba vymyslet, jakým způsobem zobrazit filtry. Zde byla zohledněna jak efektivita kódu, tak uživatelská přívětivost. Návrh z tohoto důvodu obsahuje dva typy filtrů. Jeden typ filtru provede filtraci až po stisknutí tlačítka „Aktualizovat“, oproti tomu druhý typ filtraci provede již po změně hodnoty filtru.

Při posuzování, zda je vhodné použít na jednom místě aplikace dva typy chování filtrů, jsem usoudil, že toto bude možné provést pouze v případě, kdy rozdílné chování dvou typů

filtrů uživatel bez větších problémů pochopí. Pro tento účel jsem navrhl i grafické odlišení těchto filtrů, kdy oba typy filtrů jsou odděleny viditelnou čárou.

Výběr data pro zobrazení rozvrhu

Filtrace datumů, pro které se má zobrazit rozvrh, je navržena jako filtr, který se projeví až po stisknutí tlačítka pro aktualizaci. To z důvodu, aby se nezahlcoval server neustálými požadavky při změně datumů, pokud se uživatel delší dobu rozhoduje nad správným datem, popřípadě se přesouvá na požadované datum po týdnech. Pokud by se filtrace, tedy požadavek na server, odesílal po každé aktualizaci komponenty pro výběr data, mohlo by to vést k delším odezvám z důvodu velkého množství požadavků a negativně by to ovlivnilo jak rozhraní rozvrhů, tak také všechny ostatní moduly.

Při posuzování, zda umožnit filtraci datumu také podle čísla týdne v semestru (např. 4. týden v letním semestru), bylo určeno, že podle pořadí týdne v semestru nelze jednoznačně data filtrovat, neboť každá fakulta může začínat semestr v jiné datum, stejně tak jako každý student může studovat více fakult najednou, nebo vyučující může vyučovat na více fakultách najednou.

Filtrace datumů je umožněna skrze specificky navrženou komponentu uživatelského rozhraní, kterou lze vidět na obrázku 3.2. Tato komponenta se skládá ze dvou podkomponent označených na obrázku jako 1a a 1b, které slouží pro výběr počátečního, respektive koncového data. Obě tyto komponenty se dále skládají z následujících částí: Část 1 z obrázku je pole pro výběr konkrétního data. Díky této části lze libovolně měnit aktuálně vybrané datum jako výběr z kalendáře. Tato část je poté rozšířena o několik dalších částí, které se snaží ulehčit práci s touto komponentou. Jedná se o části 2 a 3 z obrázku, které přesouvají aktuálně vybrané datum o týden vpřed, respektive o týden vzad. Část 4 z obrázku vybere aktuální den. Části 2, 3, 4 z obrázku aktualizují datum zobrazené v části 1.

V části 5, která je společná pro komponenty 1a i 1b, se poté zobrazuje interval datumů, pro který se doopravdy provede filtrace. Tato hodnota se stanovuje tak, že se ze zvoleného data v části 1a přepočítá počáteční datum jako nejbližší pondělí směrem po časové ose zpět od vybraného data. Koncové datum se zvolí jako nejbližší neděle na časové ose vpřed z vybraného data v komponentě 1b.

Nakonec komponenta pro výběr počátečního a koncového data obsahuje potvrzovací tlačítko. Po kliknutí na toto tlačítko se aktualizují zobrazené rozvrhy. Zároveň se kliknutím na toto tlačítko zaktualizuje i hodnota zobrazovaná v části 1 z obrázku, tedy se přepíše na přepočítanou hodnotu data pondělí a neděle, pro které se reálně aplikoval filtr.

Změna zobrazovaných dat a typů rozvrhových aktivit

Pro filtraci typů rozvrhových aktivit a zobrazovaných dat byl použit druhý typ filtru. Zde se zdálo být lepší změnu projevit ihned po změně hodnoty filtru, neboť nutnost každého dalšího stisknutí tlačítka by znamenala pohyb myší / prstem po obrazovce.

Pro zobrazení těchto filtrů bylo navrženo pole zaškrtávacích políček s omezením výšky elementu, který obsahuje pole těchto políček. Tím se umožní provedení filtrace na jeden klik a komponenta díky omezení výšky zaručí, že nebude zabírat spoustu prostoru a tím nebude vynucovat posun posuvníkem směrem dolů na vykreslené rozvrhy. Návrh této komponenty lze vidět na obrázku 3.3.

Alternativou by mohl být rozbalovací seznam, kdy by se po rozkliknutí seznamu zobrazily všechny možnosti dané filtrace. To by však znamenalo klik uživatele navíc a v rámci rozhraní jsem se snažil omezit množství nutných uživatelských interakcí se systémem.

Obrázek 3.2: Wireframe komponenty pro výběr týdne s označením jednotlivých částí

Obrázek 3.3: Wireframe komponenty filtrování podle typu rozvrhových jednotek

Možnosti tisku, rozšířené možnosti a export do iCal

Jelikož bylo z dotazníků zjištěno, že jak tisk, tak export do formátu iCal, jsou uživateli využívané funkce, bylo třeba k tomuto vytvořit uživatelské rozhraní.

iCal

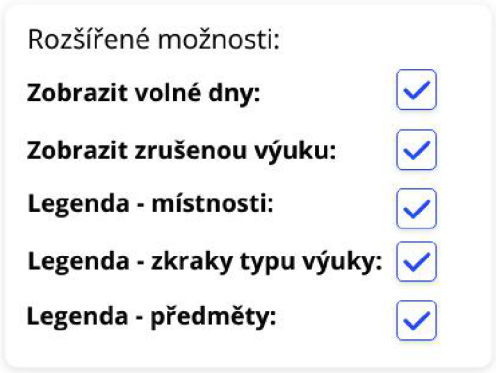
Možnost exportu do iCal se zobrazuje v následujících variantách rozvrhů:

- Osobní rozvrh – student
- Osobní rozvrh – vyučující
- Rozvrh osob

Exportování do formátu iCal se provádí jediným stisknutím tlačítka, které je umístěno na podobném místě jako u původního řešení, a to z důvodu zvyklosti a z důvodu, že na tomto místě se tlačítko zdálo být nejoptimálněji umístěné. Kromě exportování do formátu iCal jsem navrhl také možnost exportování rozvrhu do formátu `PNG`. To z důvodu, že jsem se při analýze od studentů dozvěděl, že by tuto variantu využili pro uložení rozvrhu do galerie v telefonu.

Rozšířené možnosti

Při návrhu uživatelského rozhraní jsem se snažil toto rozhraní udržovat co nejvíce jednoduché s co nejmenším počtem viditelných interaktivních prvků. Abych tohoto docílil, rozhodl jsem se pro zobrazení prvků, které ovládají nějakou specifičtější funkci, navrhnout komponentu zobrazující `rozšířené možnosti`. Touto komponentou je tlačítko, po jejímž kliknutí se zobrazí rozšířené možnosti a automaticky se okno prohlížeče posune na element s rozšířenými možnostmi. Automatický posun jsem zvolil za účelem zlepšení přívětivosti zejména na užších zařízeních, kde se rozšířené možnosti mohou vykreslit pod viditelnou částí stránky. Do rozšířených možností jsem umístil funkci pro zobrazení legend k rozvrhům. Zobrazení legend k rozvrhu je asi nejvíce potřeba při tisku, avšak pokud by si uživatel chtěl zobrazit legendy mimo tisk, nemuselo by ho napadnout, že tuto možnost má hledat právě v `možnostech tisku` (popsané v sekci 3.1). Do `rozšířených možností` jsem dále umístil možnost zrušení zobrazování zrušené výuky. Nakonec jsem do `rozšířených možností` umístil také políčko pro zapnutí / vypnutí zobrazení volných dnů v rozvrhu. Navržený vzhled těchto možností je na obrázku 3.4.



Obrázek 3.4: Návrh zobrazení rozšířených možností

Tyto možnosti se zobrazí vedle filtrů, které se aktualizují ihned po změně hodnoty filtru, neboť i rozšířené možnosti se projeví ihned bez nutnosti potvrzení dalším tlačítkem.

Tisk

Možnost tisku se zobrazuje v následujících variantách rozvrhů:

- Osobní rozvrh – student
- Osobní rozvrh – vyučující
- Rozvrh osob
- Rozvrh předmětů
- Rozvrh přednáškových skupin
- Rozvrh skupin
- Rozvrh místností

Funkce tisku byla navržena tak, aby se výsledný tisk provedl ve dvou krocích. Prvním krokem je zobrazit možnosti tisku a samotné tlačítko pro tisk. Toto se děje tlačítkem umístěným vedle tlačítka pro exportování do iCal. Následně se zobrazí možnosti tisku jako například:

- Černobílý tisk – Při převodu z barevného rozvrhu na rozvrh v odstínech šedi se snaží u různých typů rozvrhových jednotek držet dostatečně odlišný odstín šedi. Navíc často uživatelé pravděpodobně neumí používat možnost černobílého tisku v možnostech tiskárny, neboť požadavek na černobílý tisk zaznával často v dotaznících.
- Změna formátu na standardizované formáty (A5, A4, A3) – Uživatelé v dotaznících často chtěli lepší možnost nastavení velikosti tisknutých rozvrhů pro běžné velikosti papírů, aby nemuseli ručně odhadovat, jakou šířku a výšku rozvrhu mají nastavit.
- Změna velikosti písma
- Otočení papíru na výšku / šířku
- Roztažení rozvrhu přes celý papír

Kromě těchto možností se při kliknutí na tlačítko pro zobrazení možností tisku rovnou zobrazí také rozšířené možnosti, protože obsahují často potřebnou funkci při tisku – zobrazení legend. Ihned pod těmito možnostmi pak budou umístěny samotné možnosti tisku. Obdobně jako u rozšířených možností se po zobrazení možností tisku okno prohlížeče posune na element zobrazující tyto možnosti. Navržené rozhraní pro možnosti tisku lze vidět na obrázku 3.5.

Druhým krokem pro provedení tisku je stisknutí tlačítka „Tisk“, které nastavený rozvrh vytiskne.

Obrázek 3.5: Návrh zobrazení možností tisku

Výhodou nově navrženého rozhraní pro tisk rozvrhů je to, že narozdíl od původního uživatelského rozhraní neprobíhá přesměrování na další URL adresu, ze které se může provést tisk. V původním řešení se tedy uživatel musel pro další procházení rozvrhu vrátit pohybem myší a kliknutím na předchozí stránku.

Nastavení rozvrhů

Jelikož z odpovědí respondentů na otázku v dotaznících ohledně způsobu zobrazení detailu rozvrhového okna nešlo vyvodit jednoznačný závěr, navrhl jsem možnost nastavení způsobu zobrazení detailu rozvrhového okna, s výchozím nastavením, které bylo určeno v sekci 2.6.

Stejně tak nešlo určit jednoznačný závěr z odpovědí na otázku, jakým způsobem by se měly zobrazovat rozvrhy na mobilním zařízení. Tudíž se rozhodlo, že uživatel bude mít možnost transponovat si zobrazení rozvrhů. Tedy se mu prohodí časová a denní osa.

Jelikož tato nastavení nijak nesouvisí s filtrací, bylo třeba vhodně tyto dvě možnosti umístit v uživatelském rozhraní. Tato nastavení rovněž nijak nesouvisí s tiskem nebo exportem, takže ani v tomto prostoru nemohla být umístěna.

Jako vhodné umístění jsem zvolil prostor těsně nad prvním vykresleným rozvrhem, jelikož toto umístění může naznačovat, že tato tlačítka budou dělat nějakou operaci se zobrazením rozvrhů. Tato tlačítka však nemohla být duplikována nad každým vykresleným rozvrhem, neboť stisk těchto tlačítek musí ovlivnit všechny zobrazené rozvrhy a bylo by matoucí, že tlačítko nad jedním rozvrhem ovlivnilo i další rozvrh.

Nastavení způsobu zobrazení detailu rozvrhových jednotek

Kromě umístění bylo třeba zvolit i vhodnou formu vizualizace komponenty pro přepínání mezi způsoby zobrazení detailu rozvrhových jednotek. Zde bylo navrženo řešení, které vykreslí dvě tlačítka, každé pro jednu možnost zobrazení, kdy jedno tlačítko bude podbarveno jako aktuálně zvolená varianta nastavení. Toto je ilustrováno na obrázku 3.6.

Obrázek 3.6: Návrh komponenty pro přepínání mezi způsoby zobrazení detailu rozvrhové jednotky

Vykreslení rozvrhů

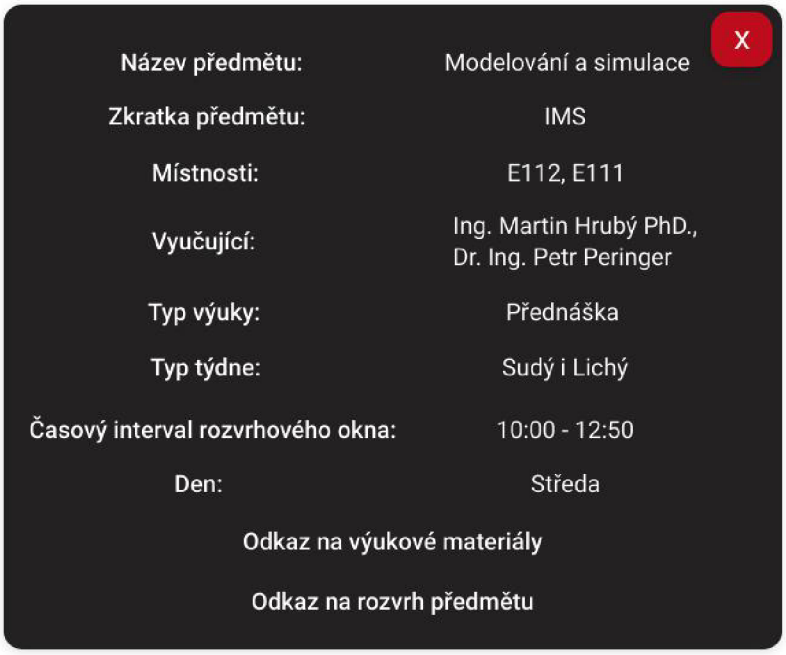
Před každým vykresleným rozvrhem je uvedený interval datumů od pondělí do neděle, pro které platí daný vykreslený rozvrh. To z důvodu, aby pro delší časový interval, pro který si uživatel nechá zobrazit rozvrhy (například celosemestrální), uživatel u jednotlivého vykresleného rozvrhu věděl, pro jaké datum tento rozvrh platí. Pro tyto účely byl určen pouze jednoduchý nadpis.

Také bylo třeba navrhnout, jaké typy rozvrhových jednotek se budou v rozvrhu zobrazovat. Kromě typů rozvrhových jednotek, které do rozvrhů načítal původní informační systém, jsem navrhl zobrazovat i termíny registrací projektů a zkoušek, termíny odevzdávání projektů. Tyto termíny jsem navrhl zobrazovat jako celodenní aktivity.

Detail rozvrhové jednotky

Informace, které se zobrazují v detailech rozvrhových jednotek jednotlivých variant rozvrhů, jsou rozebrány v příloze A. Na počítači bylo navrženo, že se detail bude zobrazovat jako obyčejné vyskakovací okno. Na mobilním zařízení bylo navrženo zobrazení tohoto okna v

horním rohu obrazovky, přes celou šířku obrazovky. To má zaručit, že detail bude vždy celý viditelný bez nutnosti posunu svislým posuvníkem nahoru (např. při rozkliknutí rozvrhové jednotky, která se nachází v horní části obrazovky), a také, že tento detail bude vždy na stejné pozici. Vzhledově však toto okno vypadá stejně jak na počítači, tak na mobilním zařízení, jak lze vidět na obrázku 3.7.

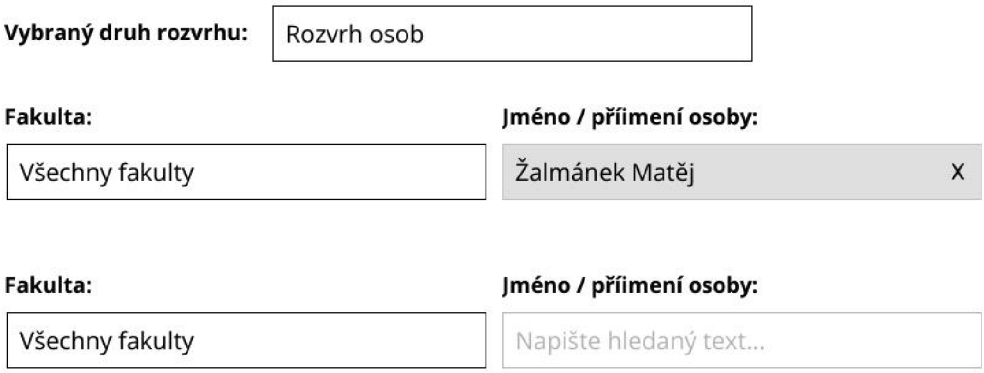


Obrázek 3.7: Návrh zobrazení detailu rozvrhové jednotky

Volba typu rozvrhu

Součástí této práce bylo také vykreslování několika druhů rozvrhů, jako například rozvrh osob, rozvrh přednáškových skupin, rozvrh skupin, rozvrh místností a rozvrh předmětů. Za tímto účelem jsem navrhl výběr druhu rozvrhu skrze komponentu rozevírací seznam (`select`). Následný výběr entity (místnost, osoba, předmět, ...) je umožněn skrze navrženou komponentu pro vyhledání konkrétní entity. Tato komponenta se skládá ze tří částí. Konkrétně to jsou prvek zobrazující zvolenou entitu, textové pole pro vyhledávání entity a rozevírací seznam pro výběr konkrétní entity. Při návrhu této komponenty jsem vycházel z již existující komponenty combobox v informačním systému VUT, které jsem mírně poupravil chování. Tuto komponentu bude možno ovládat jak myší, tak klávesnicí skrze šipky a klávesu enter. Jelikož jsem dopředu přemýšlel nad výkonností SQL dotazů, rozhodl jsem se k jednotlivým prvkům pro výběr entity přidat rozevírací seznam pro výběr fakulty, ze které je daná entita. Kromě tohoto jsem ponechal možnost vyhledat místnost podle jejího typu, jak již bylo v původním řešení. Současně jsem výběr všech entit navrhl tak, aby bylo vždy možné vybrat více entit najednou, a tak například umožnit porovnávání rozvrhu dvou osob nebo libovolných dvou a více jiných entit, což v případě osob bylo žádáno i v dotaznících. Tohoto bylo docíleno skrze dynamické přidávání komponent pro výběr entity po vybrání entity. Ukázka navržené komponenty pro toto rozhraní je uvedena

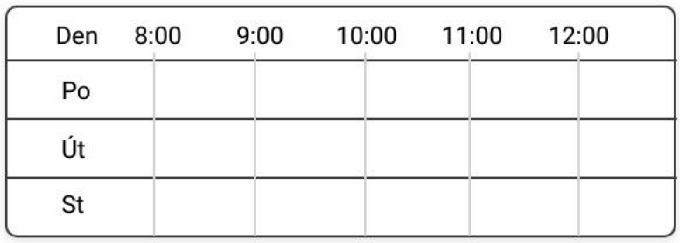
na obrázku 3.8. Pro potvrzení výběru jsem znovu navrhl potvrzovací tlačítko a celou tuto komponentu pro výběr entit jsem umístil nad všechny ostatní komponenty rozvrhů.



Obrázek 3.8: Návrh komponenty pro výběr entity, pro kterou se má zobrazit rozvrh

Vykreslený rozvrh

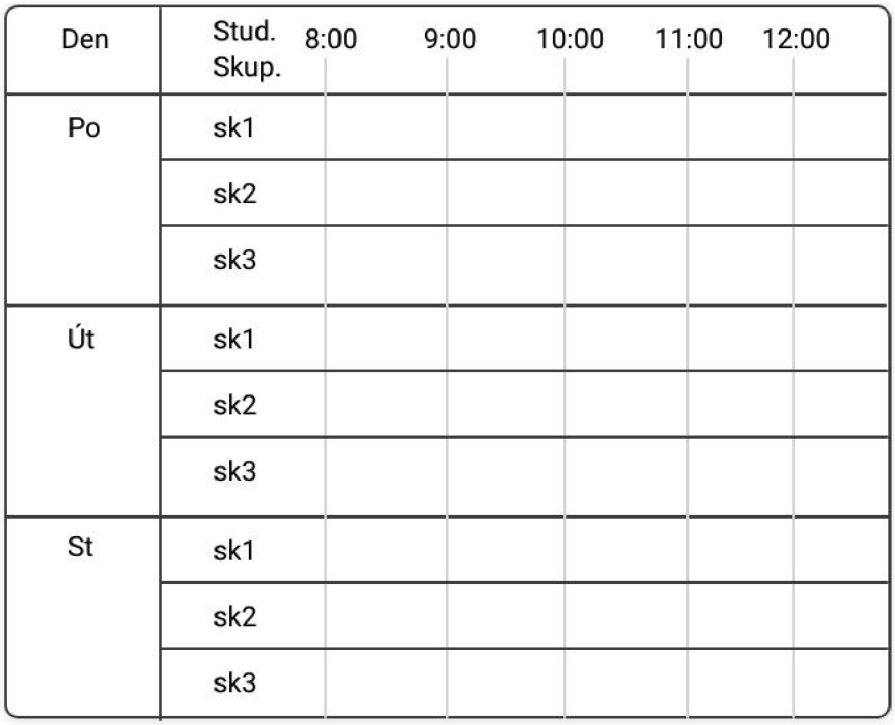
Vykreslený rozvrh byl navržen na sloupce a řádky, kdy vždy první sloupec a první řádek obsahuje nějaký popisek, co se v dalších sloupcích / řádcích zobrazuje (např. den, čas, stud. skupina, ...). Pro jednoznačné odlišení jednotlivých hodin jsem navrhl, že každý sloupec (popřípadě řádek u transponovaného rozvrhu) bude oddělen čárou. Tato čára slouží pro snazší orientaci v rozvrhu. Ze stejného důvodu jsou také jednotlivé dny jsou oddělené čárami. Tedy celý rozvrh, který by obsahoval dny Po, Út, St a časy 8:00, 9:00, 10:00, 11:00 a 12:00, bez zobrazených dat by vypadal podle návrhu asi jako na obrázku 3.9.



Obrázek 3.9: Navržený rozvrh obsahující dny Po, Út, St a časy 8:00, 9:00, 10:00, 11:00, 12:00 bez zobrazených dat

Pokud bychom chtěli do jednoho rozvrhu vykreslit také další typ popisku, například studijní skupinu (u rozvrhu studijních skupin), přibyl by ještě jeden sloupec, respektive řádek u transponovaného rozvrhu s popisem daných studijních skupin, kdy každá studijní skupina je znovu oddělená čárou. Celý rozvrh, který by obsahoval dny Po, Út, St, časy 8:00, 9:00, 10:00, 11:00, 12:00 a studijní skupiny sk1, sk2 a sk3, bez zobrazených dat by vypadal podle tohoto návrhu asi jako na obrázku 3.10:

Tento typ zobrazení dále prošel následujícími optimalizacemi pro uživatelskou přívětivost.

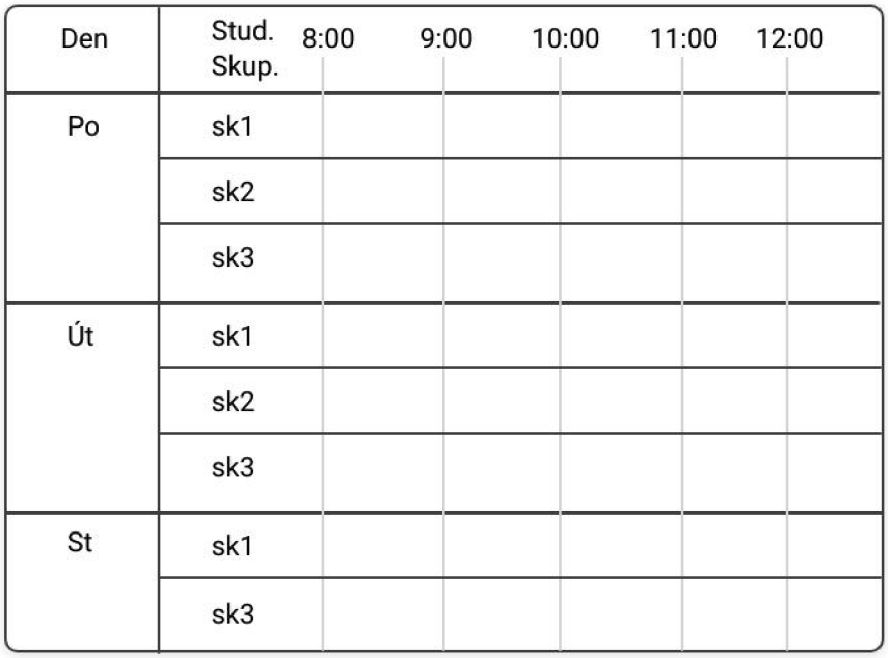


Obrázek 3.10: Navržený rozvrh obsahující dny Po, Út, St, skupiny sk1, sk2, sk3 a časy 8:00, 9:00, 10:00, 11:00, 12:00 bez zobrazených dat

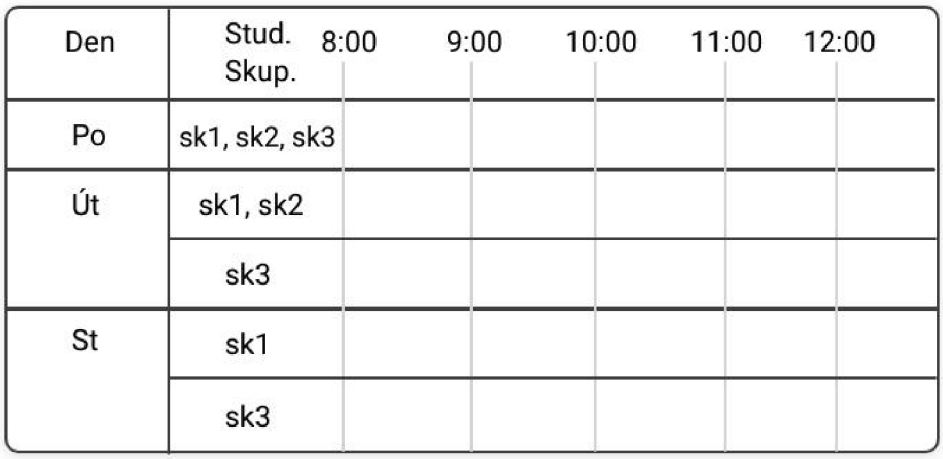
- 1) Pokud v jednotlivé dny daná studijní skupina nemá žádné rozvrhové jednotky, jednoduše se v tento den daná studijní skupina nevykreslí, například pokud studijní skupina sk2 z předchozího příkladu ve středu neobsahuje žádné rozvrhové jednotky, bude vykreslený rozvrh vypadat asi jako na obrázku 3.11.
- 2) Pokud studijní skupiny obsahují v daný den naprosto totožnou výuku, poté se popisky těchto skupin spojí do jednoho řádku, respektive sloupce u transponovaného rozvrhu. Tedy pokud příklad z předchozího již optimalizovaného rozhraní upravíme tak, že skupiny sk1 a sk2 mají v úterý totožnou výuku a skupiny sk1, sk2 i sk3 mají totožnou výuku v pondělí, bude výsledný rozvrh vypadat podle obrázku 3.12.
- 3) Pokud v daný den obsahují úplně všechny studijní skupiny totožnou výuku, popis všech skupin se zapíše jako „vše“.

Dále bylo usouzeno, že bude pro uživatelskou přívětivost komplexnějších rozvrhů zásadní, aby hlavičky rozvrhů (tedy sloupce a řádky s popisky) byly ukotveny i při posunu posuvníkem po stránce ve všech směrech. Tedy že sloupce s popiskami se ukotví při vodorovném posuvu posuvníkem a řádky se ukotví při svislém pohybu posuvníkem. Díky tomuto je vždy možná snadná orientace v rozvrhu.

Jelikož se u komplexního standardního rozvrhu v původním řešení stávalo, že nebylo vždy jasné, na jakém dni se aktuálně uživatel nachází, bylo navrženo, že i popisky se budou v rámci sloupce posouvat. Toto nastávalo v situaci, kdy rozvrh jednoho dne byl vyšší než výška stránky. Na obrázku 3.13 lze vidět navržené řešení s ukotveným popiskem dne.

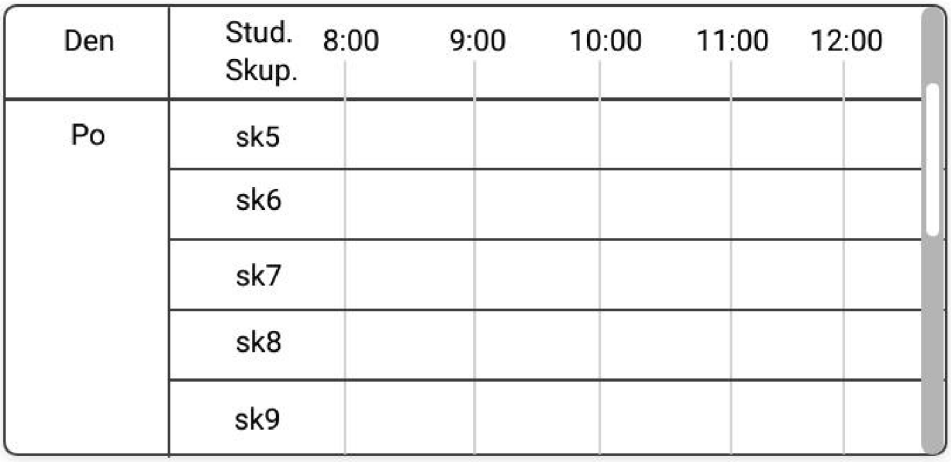


Obrázek 3.11: Navržený rozvrh s první optimalizací obsahující dny Po, Út, St, skupiny sk1, sk2, sk3 a časy 8:00, 9:00, 10:00, 11:00, 12:00 bez zobrazených dat. Skupina sk2 neobsahuje ve středu žádné rozvrhové jednotky.



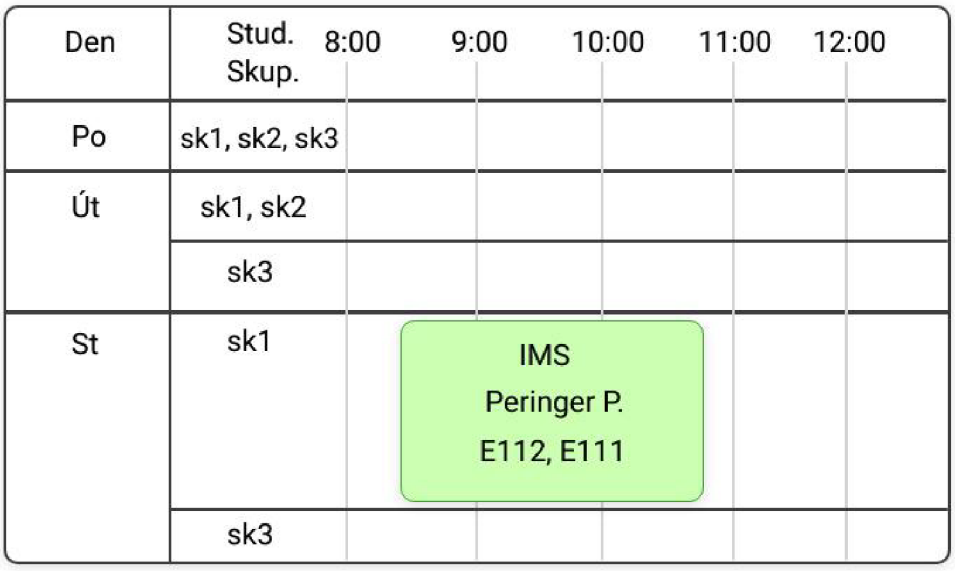
Obrázek 3.12: Navržený rozvrh s druhou optimalizací obsahující dny Po, Út, St, skupiny sk1, sk2, sk3 a časy 8:00, 9:00, 10:00, 11:00, 12:00 bez zobrazených dat. Skupina sk2 neobsahuje ve středu žádné rozvrhové jednotky, skupiny sk1, sk2, sk3 mají totožný rozvrh v pondělí a skupiny sk1, sk2 mají totožný rozvrh i v úterý.

Dále bylo třeba vhodným způsobem navrhnout zobrazení rozvrhových jednotek. Po diskuzi s kolegy v CVIS VUT jsem zjistil, že minimální šířka rozvrhového okna má být 10 minut. Pro tyto účely byl tedy každý sloupec, respektive řádek v transponovaném rozvrhu



Obrázek 3.13: Navržený rozvrh s ukotveným popiskem dne „Po“

rozdělen na dalších 6 menších sloupců, které reprezentují daný desetiminutový úsek. Tyto desetiminutové úseky však uživatel v rozhraní nijak nerozezná, kromě situace, kdy se na jejich rozmezí vykreslí rozvrhová jednotka. Díky tomuto návrhu bylo umožněno zobrazit i rozvrhové jednotky, které trvají například v čase 8:30–10:40. Tato rozvrhová jednotka by se v rozvrhu zobrazila jako jeden obdélník, jako na obrázku 3.14.

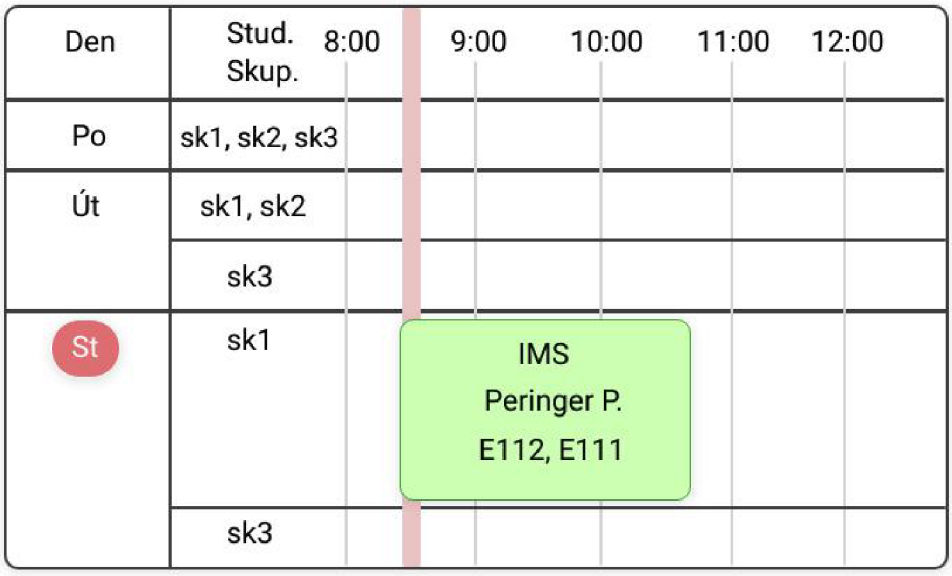


Obrázek 3.14: Navržený rozvrh obsahující rozvrhovou jednotku ve středu pro skupinu sk1 v čase 8:30–10:40

Jelikož však rozvrhové jednotky, které trvají kratší časový úsek než 1 hodinu, mohou být příliš úzké pro zobrazení všech podstatných informací nebo pro zobrazení detailu, bylo navrženo, že při kliknutí do okolí této kratší rozvrhové jednotky se rozvrhová jednotka zvětší, aby bylo možné přečíst patřičné informace, případně rozkliknout její detail. Díky

tomuto se také zajistí, že uživatel nebude muset několikrát trefovat rozvrhovou jednotku pro otevření detailu, ale rozhraní se mu tuto plochu pokusí zvětšit. Toto chování bude důležité zejména na mobilních zařízeních, nebo v případě úzkého okna prohlížeče.

Nakonec bylo navrženo, že pro ulehčení orientace v rozvrhu bude zobrazen aktuální den a čas. Aktuální čas je zobrazen jako čára přes adekvátní deseti minutovou podčást sloupce, respektive řádku v transponovaném rozvrhu. Tedy například pro aktuální čas 9:30–9:39 podle obrázku 3.15. Pro zobrazení aktuálního dne bylo navrženo, že se popisek s aktuálním dnem vloží do barevného kroužku. Tedy například, pokud je aktuální den středa, tak podle stejného obrázku 3.15.



Obrázek 3.15: Konečná podoba navrženého rozvrhu i se zobrazením aktuálního dne (St)

Zobrazení na mobilních zařízeních

Jak již bylo zmíněno dříve, pro účely zobrazení na mobilních zařízeních byla navržena možnost přepínání mezi transponovanou verzí rozvrhu a výchozí verzí rozvrhu. Toto lze využít sice i mimo tento typ zařízení, ale pravděpodobně to bude nejvíce používané právě na mobilních zařízeních. Za účelem zaručení dostatečné minimální šířky rozvrhu pro zobrazení čitelných rozvrhových jednotek jsem určil, že minimální šířka rozvrhu bude 700px. Aby rozvrh mohl mít takovouto šířku, musí se tedy v některých případech zobrazit vodorovný posuvník. Toto však na základě výsledků z dotazníků bylo žádané řešení z pohledu uživatelů na mobilních zařízeních.

3.2 Návrh formátu dat pro komunikaci mezi klientskou a serverovou částí

Server svoji odpověď odesílá v jedné odpovědi protokolem HTTP, která má formát JSON, který velmi zjednodušeně popisuje následující schéma. Atributy **data**, **labels**, **filtrs**, **sche-**

duleUnitsDataFilters, **weekTypes** a **possibleStudyGroups** mají složitější strukturu, která však pro jednoduchost není v uvedeném schématu znázorněna.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "data": {
        },
        "labels": {
        },
        "filters": [],
        "scheduleUnitsDataFilters": {
        },
        "weekTypes": {
        },
        "possibleStudyGroups": []
    },
    "required": [
        "data"
    ]
}
```

Popis jednotlivých vlastností je uveden níže:

- **Vlastnost filters** – obsahuje aktuální nastavení filtrování podle typů rozvrhových jednotek, včetně informace o výchozím zapnutí / vypnutí.
- **Vlastnost data** – v této vlastnosti se odesílají informace o všech rozvrhových jednotkách, které se podle identifikátoru předmětu a identifikátoru typu výuky v daném předmětu (např. přednášky, cvičení) sdružují do vlastnosti `blocks`. Tato vlastnost poté obsahuje všechny informace o vyučujících, místnostech a datech, ve kterých se vyučuje určitý blok rozvrhové jednotky. Toto slouží k možnosti vykreslit standardní rozvrh stejně jako týdenní rozvrh.

  Jelikož se musel navrhnout vhodný způsob, jak na straně klienta rozlišit různé typy rozvrhových jednotek, bylo navrženo, že celodenní aktivity nebudou nikdy obsahovat vlastnost `timeStart`, která je potřebná pro blokovou aktivitu, ale nedává smysl v případě celodenní aktivity. Takto tedy lze jednoduše odlišit, kdy vykreslit celodenní aktivitu a kdy blokovou aktivitu.
- **Vlastnost labels** – v této vlastnosti se odesílají popisky jednotlivých informací v rozvrhové jednotce, například že vlastnost rozvrhové jednotky `subjectName` se má uživateli vypsat jako „Název předmětu“ a další.
- **Vlastnost weekTypes** – obsahuje popisky k jednotlivým identifikátorům typů týdne, jako například lichý, sudý týden atp.
- **Vlastnost scheduleUnitsDataFilters** – obsahuje všechny možné datové filtry, které uživatel může zapínat / vypínat v uživatelském rozhraní. Lze tedy jednoduše ze serveru měnit zobrazované filtry a jejich výchozí zapnutí / vypnutí.

- **Vlastnost possibleStudyGroups** – obsahuje všechny studijní skupiny, které se pro daný rozvrh mají brát v potaz. Toho lze využít například pro agregaci výpisu všech názvů skupin jako výpis „všechny“.

3.3 Návrh architektury

Po návrhu formátu přenášených dat jsem navrhl celou architekturu aplikace, která je navržena skrze diagramy tříd. Tyto diagramy lze najít v příloze B.

Při návrhu jsem se snažil co nejvíce využívat vlastností objektově orientovaných jazyků (Javascript, PHP), jako jsou zejména dědičnost, polymorfismus a zapouzdřenost. Také jsem při návrhu aplikoval vybrané návrhové vzory, jako například `Jedináček` (`Singleton`), `Tovární metoda` (`Factory method`), `Fasáda` (`Facade`) a `Repozitář` (`Repository`).

Důležitou částí návrhu bylo také oddělení logiky vykreslování uživatelského rozhraní od logiky, která upravovala data pro toto rozhraní. Architektura komponent frameworku React, jež se starají o logiku vykreslování, je uvedena na diagramu tříd ve výše zmíněné příloze na obrázku B.4. Oproti tomu architektura pomocných tříd určených pro úpravu dat a komunikaci s těmito komponentami je uvedena ve stejné příloze na obrázcích B.5, B.6, B.8, B.9, B.10 a B.11.

Kapitola 4

Nástroje a technologie použité pro vývoj

Jelikož se jedná o část aplikace informačního systému VUT, volba technologií nebyla součástí této práce. Technologie jako PHP, SQL byly zvoleny kvůli tomu, že jsou v současné době využívány k vývoji informačního systému VUT. Technologie jako React.js pak byla zvolena na základě preferencí zaměstnanců CVIS z důvodu udržovatelnosti.

4.1 Značkovací jazyk HTML

HTML (Hypertext Markup Language) je značkovací jazyk určený k tvorbě nejen webových stránek.

Značkovací elementy, ze kterých je tvořen tento jazyk, jsou vytvářeny tzv. **tagy**. Většina elementů se skládá z počátečního tagu jako například * * a koncového tagu, který se vyznačuje zpětným lomítkem, například * *. Avšak ne všechny tagy musí být nutně párové, existují i nepárové tagy, které neobsahují koncový tag. Více o tomto jazyce se lze dočíst v [7].

V rámci této práce byla použita nejnovější verze tohoto jazyka HTML5.

4.2 Kaskádové styly a knihovna Bootstrap

CSS (Cascading Style Sheets) je jazyk, který byl vyvinut pro oddělení tvorby struktury dokumentu pomocí značkovacích jazyků od definice jeho vzhledu. Tento jazyk se tedy stará o definici vzhledu dokumentů, přičemž vzhled bývá většinou definovaný v separátních souborech, ale nemusí to být vždy pravidlem. Vzhled dokumentu je definován tzv. **pravidly**, přičemž každé pravidlo se skládá ze **selektoru** a následně **N vlastností**, oddělených oddělovačem **„;“**, které mají vždy jednu **hodnotu**. Více o jazyce CSS se lze dočíst v [7].

V rámci této práce bylo nejvíce využito vlastnosti `display grid`, která slouží pro uspořádání prvků uvnitř kontejneru s tímto stylem do mřížky. Dále jsem využíval pro tuto práci vlastnost `visibility hidden`, která skryje prvek na stránce, ale prvek stále bude zabírat svůj prostor na stránce. Dále jsem využíval mnoho dalších základních vlastností CSS, o kterých se lze dočíst z výše uvedeného zdroje.

Bootstrap Jedním problémem kaskádových stylů je to, že na různých zařízeních a v různých prohlížečích se stránka může zobrazovat odlišně. Abychom nemuseli vzhled definovat pro každé zařízení a pro každý prohlížeč zvlášť, používá se CSS knihovna Bootstrap. Bo-

otstrap využívá HTML, CSS i Javascript a je podporován v drtivé většině moderních prohlížečů. Tato knihovna obsahuje sadu předdefinovaných pravidel CSS, která lze využít pro tvorbu nejrůznějších základních, běžně používaných komponent webu [6].

V této práci byla využita verze Bootstrap 3, jelikož tato verze je použita v celém informačním systému VUT a byl interní požadavek toto dodržet i v této práci.

4.3 Javascript pro klientskou část modulu

Javascript je interpretovaný, slabě typovaný, objektově orientovaný, vysokoúrovňový jazyk. Bývá využíván zejména v oblasti vývoje webových aplikací na straně klienta, kde se stará především o interakci s uživatelem, ale díky rozvoji node.js se stal často i programovacím jazykem na straně serveru.

V rámci této práce jsem využíval jednu z nejnovějších verzí specifikace Javascriptu, specifikaci ECMAscript 13. Jelikož však bylo třeba zaručit funkčnost kódu i v prohlížečích, které nepodporují nejnovější specifikace ECMAscript, využil jsem nástroj, který převádí kód novějších specifikací na funkčně ekvivalentní kód starších specifikací ECMAscript, v mém případě ES6, jež se jmenuje *Babel*.

4.4 Souvislosti mezi jazyky HTML, CSS, Javascript

Pro efektivní implementaci uživatelského rozhraní bylo také zapotřebí chápat souvislosti mezi jazyky HTML, CSS, Javascript a to, jakým způsobem se kód v těchto jazycích vykreslí na displej uživatele.

Kritická vykreslovací cesta

Na obrázku 4.1 je uvedena tzv. **kritická vykreslovací cesta (anglicky Critical rendering path)**, která se používá při vykreslování webových stránek. Tento obrázek popisuje životní cyklus vykreslování webové stránky, kdy se nejprve musí vstupní řetězec v HTML zpracovat do **DOM (Document object model)**. Následně se všechna pravidla CSS přepíší do tzv. **CSSOM (CSS Object Model)**. Po vytvoření těchto dvou stromových struktur se pokračuje v tvorbě tzv. **renderovacího stromu**, který obsahuje pouze uzly, které se budou tisknout. Tedy nebude obsahovat elementy s vlastností display: none. Pokud však bude mít prvek na stránce vlastnost visibility hidden, v renderovacím stromě i nadále bude, neboť takovýto prvek zabírá určitý prostor na stránce, pouze je skrytý. Po vytvoření vykreslovacího stromu se pokračuje tvorbou tzv. **layoutu**. Ten obsahuje informace o pozicích a velikostech jednotlivých uzlů v tomto stromě. Po tomto kroku lze vytvořit také informace o vrstvách, tedy na jaké pozici v ose **z** bude element vykreslen – tento krok se označuje jako **vykreslování** nebo také rasterizace. V posledním kroku kritické vykreslovací cesty se pouze odešlou vytvořená data o způsobu vykreslování na grafickou kartu, kde se tato data vykreslí. Tento krok se nazývá **kombinování**. Tedy až po tomto kroku nastanou změny, které vidí uživatel na jeho displeji.

Podstatnou vlastností této vykreslovací cesty je to, že v mezičase, kdy se tvoří jedna část vykreslovací cesty, je její následující část blokována, neboť například nelze tvořit vykreslovací strom bez vytvořeného objektového modelu CSS. Toto je velmi důležité při prvním načtení stránky, ovšem při složitějších prvcích, které se na webu musí vykreslovat, je vhodné tyto koncepty chápat, aby bylo možné efektivně implementovat uživatelská rozhraní. Tato část popisující kritickou vykreslovací cestu byla převzata z [4].

Obrázek 4.1: Kritická vykreslovací cesta, převzato z [4]

4.5 Balíčkovací systém NPM

Node package manager (NPM) je balíčkovací systém, který poskytuje rozhraní pro instalaci a správu knihoven psaných v jazyce Javascript, tzv. **node moduly**. Mimo to taky poskytuje úložiště pro tyto moduly, kam kdokoliv může nahrát svůj modul nebo si stáhnout do svého projektu modul cizí.

Hlavní výhodou tohoto balíčkovacího systému je velké množství dostupných modulů, díky kterým lze řešit mnoho problémů efektivně, a to bez nutnosti vlastní implementace. V rámci tohoto projektu byly využity následující node moduly:

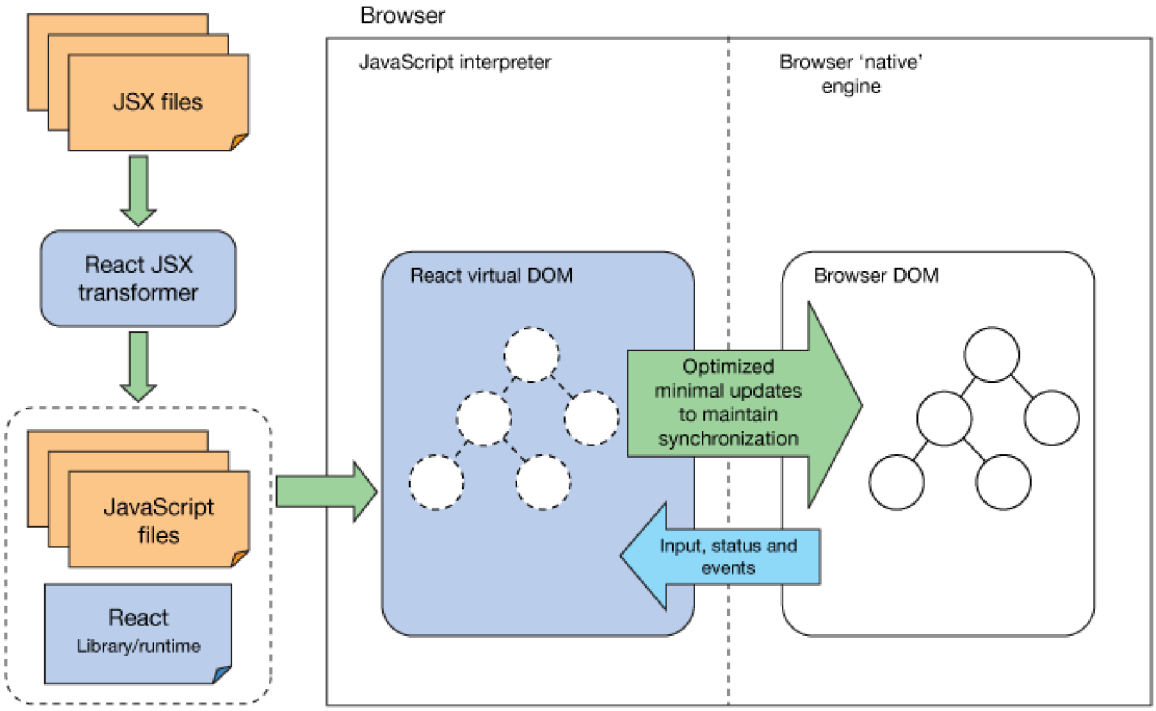
- Babel – nástroj pro konverzi kódu z novější specifikace ECMAscript na starší.
- Jest – knihovna pro testování kódu v jazyce Javascript
- React – modul pro psaní komponent uživatelského rozhraní
- Bootstrap
- Jquery – Tento modul byl nainstalován z důvodu správné funkce některých prvků uživatelského rozhraní z knihovny bootstrap. Mezi takové prvky se řadí například implementovaná vyskakovací okna.

4.6 React pro tvorbu uživatelského rozhraní

React je knihovna napsaná v jazyce Javascript pro tvorbu uživatelských rozhraní, za jejímž vývojem stojí firma Facebook. Aplikace vytvořené s využitím React jsou tvořeny z komponent, které se vyznačují svým vzhledem a svojí logikou. Tyto komponenty se mohou skládat z mnoha jiných komponent, a tak se z jednodušších komponent dají tvořit komponenty složitější. Díky Reactu se navíc zpřehledňuje tvorba elementů HTML uvnitř kódu v jazyce Javascript, a to díky JSX (Javascript XML), což je rozšíření jazyka Javascript, které svojí syntaxí připomíná běžné HTML, ovšem je nepatrně striktnější.

Díky tomuto lze nahradit veškerá volání `let el = document.createElement("div");` a následné operace s proměnnou `el` za běžný kód v jazyce HTML (`<div></div>`). Veškerý JSX kód se před spuštěním musí transpilovat do běžného kódu v jazyce Javascript, o což se stará již zmíněný Babel. Více se lze dočíst z [8].

Obrázek 4.2 demonstruje celý proces od transpilace souborů ve formátu JSX po samotné zobrazení stránky v prohlížeči. Po vytvoření souborů JSX s kódem v jazyce Javascript a definic všech komponent se veškerý kód transpiluje do čistého Javascriptu. React si poté vytvoří svoji vlastní reprezentaci DOM (Document object model), ve které si uchovává informace o komponentách, a následně v případě potřeby přepisuje celý svůj virtuální DOM do nativního DOM v prohlížeči. Díky tomuto virtuálnímu DOM, o který se stará React, je možné překreslovat pouze ty elementy v reálném DOM, které se změnily, čímž se výrazně



Obrázek 4.2: Proces transpilace souborů ve formátu JSX po samotné zobrazení v prohlížeči, převzato z [5]

zvyšuje efektivita oproti neustálému přepisování reálného DOM. Naopak při interakci uživatele s reálným DOM v prohlížeči se veškeré události oznámí do virtuálního DOM Reactu, díky čemuž se události mohou dále zpracovat běžným způsobem.

4.7 Formát iCalendar

ICalendar (Ical) je standardizovaný formát, určený pro přenos kalendářových dat. Skrze komponentu tohoto formátu, tzv. **VEVENT**, se vytváří informace o jednotlivých událostech v kalendáři. Více o tomto formátu se lze dočíst z [1].

4.8 PHP pro serverovou část modulu

PHP (Hypertext Preprocessor) je skriptovací, slabě typovaný, objektově orientovaný programovací jazyk, který se v současnosti používá především pro tvorbu serverové části webových stránek.

V rámci této práce se pracovalo s PHP verze 7.4.30 a to z důvodu, že se jedná o aktuálně nainstalovanou a používanou verzi na serverech informačního systému VUT. Konkrétně byl v PHP napsán koncový bod pro zpracování požadavků od klientské části aplikace.

4.9 SQL a komunikace s databází

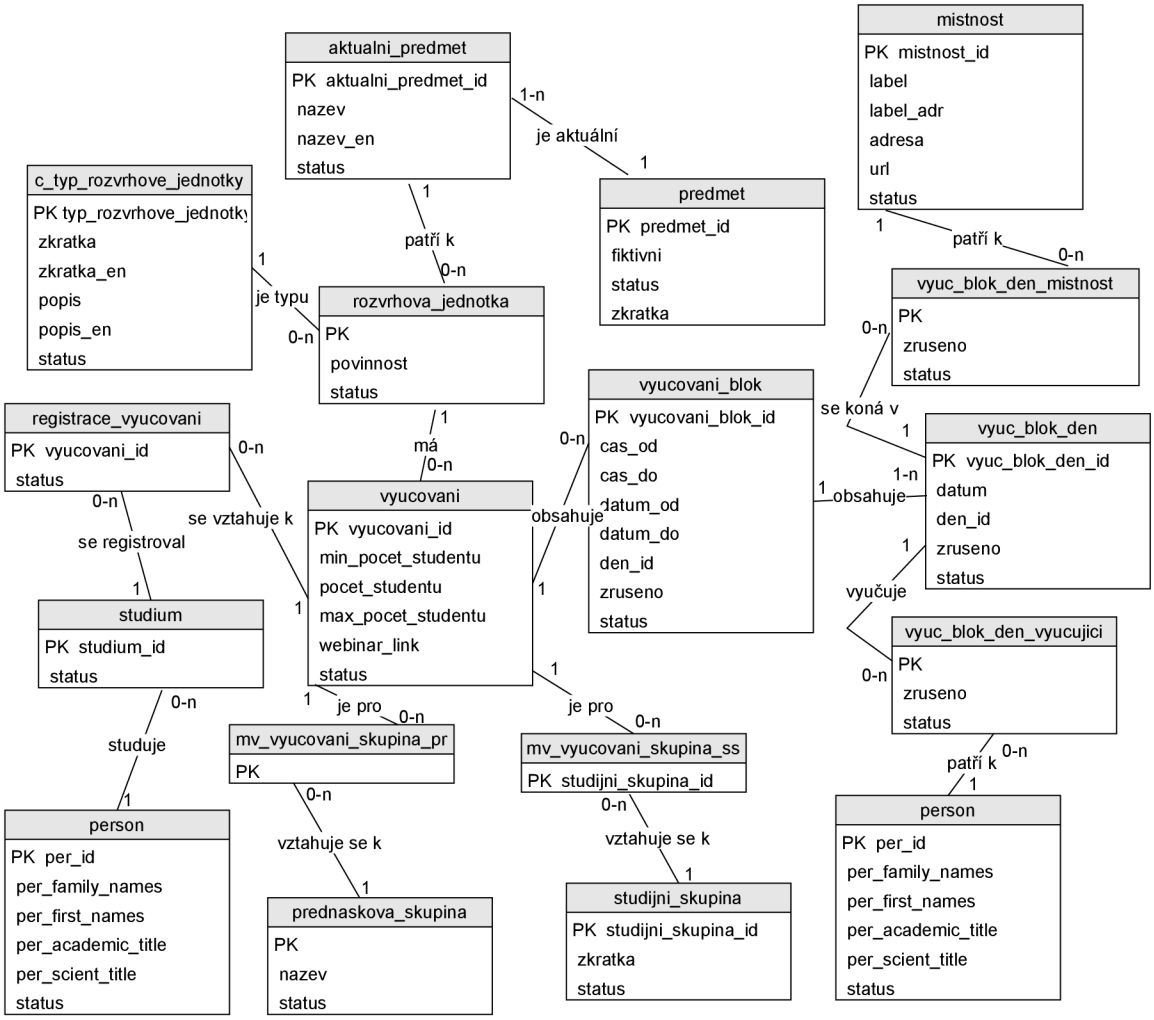
SQL je dotazovací jazyk, který poskytuje možnosti pro získání, editaci, nebo jinou formu manipulace s daty v relačních databázích.

V rámci této práce byla používána konkrétně databáze Oracle, protože tento typ databáze je užit na serverech informačního systému VUT.

Návrh ER diagramů nebyl součástí této práce a všechny diagramy, se kterými jsem v této práci pracoval, byly navrženy zaměstnanci VUT CVIS.

Nejdůležitější a zároveň nejsložitější diagram pro výpis výuky v předmětech je uveden na obrázku 4.3. Tento diagram poskytuje informace o všech typech výuky v daných předmětech a jejich konání, včetně místností a vyučujících, kteří v daný den dané vyučování vyučují. Entitní množina `vyucovani_blok` slouží pro uchování informací o všech datech konání daného vyučování. Díky této entitní množině a entitní množině `vyucovani` lze tvořit standardní rozvrh, protože standardní rozvrh nezobrazuje konkrétní bloky, pouze obecné vyučování.

Ostatní ER diagramy používané při této práci jsou uvedeny v příloze C.



Obrázek 4.3: ER diagram popisující problematiku výuky v předmětech

Kapitola 5

Implementace uživatelského rozhraní

Uživatelské rozhraní bylo implementováno podle výše zmíněného návrhu a analýzy požadavků uživatelů. Občas bylo třeba provádět změny, které reagovaly na uživatelské testování, ale díky vhodně navržené architektuře tyto změny byly vždy pouze změnami v implementaci, nikoliv architektuře.

5.1 Příprava vývojového prostředí NPM

Nejprve jsem si nainstaloval balíček `react-create-app`, který předpřipraví projekt pro vývoj v knihovně React. Tento i všechny další balíčky byly instalovány za pomoci příkazu `npm install nazev_balicku`.

Tento příkaz nainstaluje do projektu balíček s názvem `nazev_balicku` a vloží referenci na tento balíček do souboru `package.json`, který slouží pro inicializaci projektu pomocí příkazu `npm install` v případě, že někdo další bude chtít provádět změny v tomto projektu.

Poté se pomocí `npm create-react-app` inicializoval projekt, včetně základních skriptů pro spuštění či testování aplikace.

Výchozí adresářová struktura, kterou používá většina projektů s využitím React.js, a kterou vygeneroval i balíček `create-react-app` je následující:

```
/node_modules/
/public/
/src/
/package-lock.json
/package.json
```

Ve složce `node_modules` jsou nainstalovány všechny závislé NPM moduly, včetně jejich závislostí. Složka `public` obsahuje soubory, ke kterým je přístup skrze URL a které nejsou součástí kompilace projektu skrze Webpack. Složka `src` obsahuje zdrojový kód celého uživatelského rozhraní a dalších implementačních záležitostí. Nakonec soubor `package-lock.json` popisuje strom závislostí projektu i jeho nainstalovaných modulů a zaručuje instalaci správných závislostí nezávisle na době, kdy se projekt inicializuje na daném zařízení.

5.2 Implementace vykreslování rozvrhů

Pro implementaci mřížky pro vykreslování rozvrhů se již z podstaty problému hodila vlastnost CSS `display grid`. Tato vlastnost například automaticky upravuje výšku a šířku řádků / sloupců mřížky tak, že všechny buňky řádku / sloupce jsou vždy uspořádány za sebou a žádná buňka nevyčnívá.

Mapování rozvrhových jednotek do mřížky

Pro základní mapování rozvrhové jednotky do mřížky se využívají tři vlastnosti rozvrhové jednotky. Konkrétně jimi jsou:

- identifikátor dne, ve který rozvrhová jednotka probíhá,
- počáteční čas konání jednotky a
- koncový čas konání jednotky.

Z identifikátoru dne se určí číslo řádku, z počátečního a koncového času konání se dále určí sloupce v daném řádku. Pro složitější strukturu rozvrhu, jako je například studijní skupina, bylo nadále třeba rozdělit řádky na menší „podřádky“, ve kterých se vypíše výuka dané studijní skupiny v daný den. Tyto podřádky však lze využít i k vykreslení libovolných typů dat, například hromadných osob nebo hromadných místností.

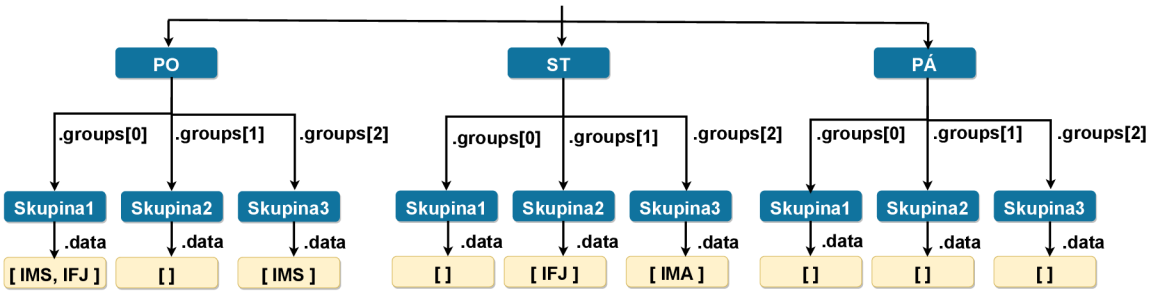
Objekt skupina

Z hlediska implementace jsem k tomuto účelu vytvořil koncept **skupin (groups)**. Tato `skupina` je objekt, který umožňuje snadno přistupovat k těmto řádkům a podřádkům a rozvrhovým jednotkám v nich uloženým. Tyto `skupiny` lze do sebe rekurzivně zanořovat a vytvářet tak složitější strukturu rozvrhu. `Skupina (group)` vypadá následovně:

```
{
    groupName: "string",
    dayID: int,
    height: "string",
    width: "string",
    classNames: [],
    data: [],
    groups: []
}
```

Vlastnost `groupName` označuje název skupiny, a tento název bude použit v hlavičce daného řádku / podřádku. Například, pokud by `groupName` mělo hodnotu „Pá“, vykreslí se buňka obsahující text „Pá“. Vlastnost `classNames` obsahuje pole všech tříd CSS, které se mají přiřadit této buňce při vykreslení jako element HTML. Vlastnost `dayID` reprezentuje identifikátor dne a podle této hodnoty lze později v kódu vypsat název dne v patřičném formátu, nebo provádět jiné operace s identifikátorem dne. Vlastnosti `height` a `width` slouží pro určení výšky a šířky řádku v mřížce. Nakonec vlastnosti `data` a `groups`, z nichž vždy právě jedna musí nabývat hodnoty `null`, reprezentují data rozvrhových jednotek a podskupiny (podřádky).

Tyto skupiny jsou na nejvyšší úrovni uspořádány do pole, díky čemuž z nich lze tvořit výše zmiňované řádky. Pokud bychom například chtěli vykreslit rozvrh, který obsahuje



Obrázek 5.1: Skupiny uspořádané do stromové struktury

rozvrhové jednotky v úterý a ve středu, vytvořili bychom pole skupin, kdy první položkou tohoto pole je skupina s `groupName` „Út“ obsahující pole rozvrhových jednotek z úterý, druhým indexem by byla skupina s `groupName` „St“ obsahující jednotky ze středy.

Pokud by nás naopak zajímal rozvrh přednáškových skupin, každou skupinu v poli bychom ještě rozdělili na další podskupiny, které by obsahovaly rozvrhové jednotky dané skupiny v úterý, obdobně pro středu. Takto prakticky lze na `skupinu` nahlížet jako na uzel n–nárního stromu. Pole skupin na nejvyšší úrovni je pak kořenem tohoto stromu.

Díky tomuto stromovému uspořádání skupin tak lze vypsat teoreticky nekonečné množství skupin, které je však omezené velikostí zásobníku, jelikož se strom zpracovává rekurzivním voláním funkcí. Navíc z uživatelského hlediska nikdy nebude vhodné vypisovat velké množství skupin, takže toto omezení stejně prakticky nikdy nebude dosaženo.

Uspořádání do skupin dále ilustruje obrázek 5.1. V tomto obrázku jsou tři skupiny v hlavním poli skupin a to konkrétně s označením PO, ST, PÁ. Potom každá hlavní skupina obsahuje tři podskupiny s označením SKUPINA1, SKUPINA2, SKUPINA3. Tato tři označení skupin označují například studijní skupiny. Nakonec každá skupina má svoje pole `data`, ve kterém jsou rozvrhové jednotky, které se konají v daný den pro danou studijní skupinu.

Řešení překryvů ve výuce a příprava výsledné matice

Jelikož koncept skupin řeší pouze ideální stav, tedy že rozvrh nikdy neobsahuje překrývající se rozvrhové jednotky, bylo třeba tento koncept dále rozšířit. Pro tento účel jsem vymyslel algoritmus uvedený v příloze D, který z pole skupin na vstupu vygeneruje řídkou matici rozvrhových jednotek na výstupu, která reprezentuje již finální podobu mřížky pro vykreslení rozvrhu.

Pseudokód algoritmu z přílohy využívá dvě pomocné funkce, kde první funkce nazvaná `prepareGroupData` připravuje data do dalších řádků, které reprezentují překryvy v dané skupině.

Druhá funkce `getGridMatrixRows` se stará o získání všech řádků pro výstupní mřížku ze skupiny `actualGroup`. Tato funkce vytváří výstupní řádky tak, že každý řádek, který je vytvořen, obsahuje nejprve informace o všech skupinách, pod které patří data v tomto řádku, a až poté obsahuje svoje data. To se hodí například v případě studijních skupin, kdy máme například v pondělí výuku dvou studijních skupin sk1 a sk2. Proměnná `actualGroup` tedy obsahuje skupinu Po, která obsahuje dvě podskupiny sk1, sk2, z nichž každá obsahuje svoje data dataSk1, respektive dataSk2.

Výstup tohoto algoritmu by vypadal následovně:

```
[
    [Po, sk1, dataSk1],
    [Po, sk2, dataSk2]
]
```

Hlavní část pseudokódu za pomoci těchto dvou funkcí poté generuje výstupní řídkou matici, která reprezentuje finální mřížku, kterou je třeba pouze transformovat do patřičného řetězce. Důležitou vlastností tohoto algoritmu je, že algoritmus do prvního řádku mřížky vždy generuje všechny buňky mřížky, aby bylo možné určit, jak moc široká mřížka se má vykreslit. Další důležitou vlastností tohoto algoritmu je, že první řádek a první sloupec vždy obsahuje kromě informací o buňkách záhlaví mřížky také nastavení výšky nebo šířky řádku, respektive sloupce.

Transformace řídkého 2D pole na řetězec

Pro transformaci řídkého pole bylo třeba vymyslet algoritmus, který toto řídké pole převede na řetězcový zápis, který lze použít jako hodnotu CSS vlastnosti `grid-template`. Jelikož výstupní matice z předchozího algoritmu nijak neřeší transponaci, bylo třeba vymyslet také algoritmus pro transponování této matice přímo do řetězcového zápisu.

Pseudokód pro přepis řídké matice do řetězcového zápisu je uveden v příloze D.1. Tento algoritmus generuje maticový zápis mřížky v řetězci, který není transponovaný a je obohacen o definici výšky řádků, respektive šířky sloupců skrze první prvky vstupní řídké matice m, které obsahují nastavení výšky / šířky.

Pokud bychom chtěli vytvořit transponovanou variantu tohoto řetězce, museli bychom zavést pro jeho tvorbu pole, do kterého bychom průběžně ukládali části všech řádků řetězce, a po zpracování celé řídké matice bychom tyto řádky spojili do jednoho řetězce. Tedy body 7, 11, 14 a 16 původního algoritmu by se změnily z přidání do proměnné `res` na přidání do pole na pozici `(renderedCount + renderedCountTimes)`. V bodě 20 by se poté řetězce v tomto poli spojily do jednoho velkého řetězce a přidala by se k nim definice výšky / šířky buněk.

Vytvoření elementů HTML

Z původní řídké matice lze jednoduše vytvořit také elementy HTML. Tyto elementy jsou vytvářeny vždy tak, aby v případě výskytu pojmenované buňky v této matici vícekrát se takto pojmenovaný element vytvořil pouze jednou. Jelikož data v řídké matici obsahují všechna data o rozvrhové jednotce, není problém vykreslit jak rozvrhovou jednotku, tak i její detail.

Po vytvoření všech elementů a nastavení správné vlastnosti CSS `grid-area` těmto elementům a po vytvoření kontejneru těchto elementů s nastavenými vlastnostmi `display grid` a `grid-template` na řetězec z algoritmu 5.2 již máme vykreslenou mřížku, kterou jsem dále pouze graficky upravoval vhodnými vlastnostmi CSS.

Tvorba uživatelsky přívětivých rozvrhů

Výstupem výše zmíněných algoritmů je již sice v základních případech použitelný rozvrh, ovšem pro složitější rozvrhy je výstup tohoto algoritmu stále nedostačující. Pro vylepšení

uživatelské přívětivosti bylo třeba implementovat chování rozvrhů tak, jak bylo popsáno v sekci 3.1.

Ukotvení záhlaví řádků

Jelikož pro snadnou orientaci ve vykresleném rozvrhu je třeba vždy vidět záhlaví s popisem časů konání rozvrhových oken, a to i v případě skrolování v rámci rozvrhu, bylo třeba vhodným způsobem implementovat ukotvení tohoto záhlaví.

Obecně lze říci, že toto lze řešit dvěma způsoby:

- S pomocí vlastností CSS (`position`, `left`)
- S pomocí Javascriptu

Jelikož druhá možnost, tedy řešení s pomocí Javascriptu nebude nikdy tak efektivní jako nativní řešení přímo skrze CSS, maximální úsilí jsem věnoval vyhledávání vhodného způsobu, jak tohoto skrze CSS docílit.

Zjistil jsem, že kombinací dvou vlastností CSS a jejich hodnot `position: sticky` a `top: N px` lze tohoto docílit, ale bohužel toto řešení není využitelné, jak je popsáno dále.

Vlastnost `position: sticky` slouží pro tvorbu plovoucích prvků, které mohou plovat v rámci nějakého nadřazeného kontejneru. Plovat však mohou pouze v případě, že mají nějaké sesterské elementy, tedy že mohou plovat vůči jiným elementům. Pokud jsou tyto podmínky splněny, může být element nastaven jako plovoucí. Tyto podmínky v rámci elementů pro rozvrhy byly splněny, neboť rozvrh obsahuje svůj kontejner s vlastností `display: grid` a buňky rozvrhu, které mohou být právě nastaveny jako plovoucí.

Nastavil jsem tedy buňkám, které měly být nastaveny jako plovoucí (záhlaví rozvrhů), vlastnosti `position: sticky` a `top` na vhodnou hodnotu. Toto způsobilo, že požadované buňky doopravdy disponovaly plovoucím chováním. Ovšem nastal problém, kdy při zmenšování obrazovky se vykreslený rozvrh nevešel na šířku okna prohlížeče (kvůli vlastnosti rozvrhu `min-width`), a tak se musel zobrazit vodorovný posuvník v rámci celé stránky. Toto bylo třeba opravit, aby se vodorovný posuvník zobrazil jenom v rámci kontejneru s rozvrhy, čehož se docílilo skrze vlastnost `overflow: auto` pro tento kontejner. Kvůli tomuto nastavení se však změnilo chování vlastnosti `position: sticky`, konkrétně se nyní požadované záhlaví ukotvilo pouze pro případ skrolování v rámci kontejneru s rozvrhy, nikoliv při skrolování tělem dokumentu. Toto z pohledu uživatelské přívětivosti také nebylo vhodné, protože přepínání mezi posunem těla dokumentu a samotným rozvrhem způsobuje zvláštní chování při přechodu ze skrolování uvnitř rozvrhu na skrolování uvnitř těla dokumentu. Toto chování lze popsat jako neochotu prohlížeče začít ihned skrolovat vůči tělu dokumentu, pokud se svislý posuvník ocitl vlivem skrolování na konci rozvrhu. Stejné chování lze zpozorovat i v opačném případě.

I přes rozsáhlé snahy a průzkumy jiných alternativních řešení se nepodařilo nalézt vyhovující řešení. Bylo tedy přistoupeno ke způsobu, který kombinuje jak ukotvení rozvrhu skrze kód v jazyce Javascript, tak nativní ukotvení skrze CSS. Konkrétně, v případě, že šířka okna prohlížeče (popřípadě nadřazeného elementu rozvrhu) je větší než šířka kontejneru s rozvrhy, použije se nativní způsob s využitím CSS. V opačném případě se použije ruční implementace `sticky` vlastnosti skrze Javascript.

Ukotvení hlavičky rozvrhu s pomocí funkcí psaných v jazyce Javascript

Pro ukotvení hlaviček rozvrhů byl zvolen práh přepínání mezi nativním CSS a kódem v jazyce Javascript na 700px šířky rozvrhů. Tato hodnota je navržena v sekci 3.1 jako nejmenší

možná hodnota pro stále ještě přehledné zobrazení rozvrhů bez přílišného zúžení rozvrhových oken. Tedy při načtení stránky, popřípadě změně velikosti okna se podle šířky nadřazeného elementu s rozvrhem určí, jaká varianta se použije.

Implementace této funkce funguje tak, že nejprve se na tělo dokumentu pověsí naslouchání na událost typu „scroll“. Toto naslouchání je vytvořeno pouze jednou pro všechny vykreslené rozvrhy, aby se co nejméně plýtvalo pamětí. Poté se při pohybu po stránce detekuje, zda uživatel vrchním okrajem výřezu prohlížeče najel na kontejner s rozvrhy, pokud ano, zobrazí se nový kontejner, obsahující pouze hlavičku rozvrhu, který má nastavenou CSS vlastnost `position fixed`. Díky tomuto se tato nově zobrazená hlavička chová stejně jako v případě nativního řešení skrze CSS. Pokud naopak vrchní výřez prohlížeče opustí spodní okraj rozvrhů, kontejner obsahující tuto hlavičku se znovu skryje.

Jelikož toto řešení je méně efektivní než řešení skrze nativní CSS a cílovou platformou tohoto řešení jsou často méně výkonná mobilní zařízení, snažil jsem se kód psát co možná nejefektivněji. Toho bylo dosaženo skrze následující stavový automat 5.2, díky němuž jsem nemusel při každé události nastavovat vlastnost `position`, ale mohl jsem si pamatovat, v jakém jsem aktuálně stavu vůči vykreslenému rozvrhu, a na základě toho se rozhodnout pro jednu z následujících možností:

- Nastavit vlastnost `position` na hodnotu „fixed“
- Nastavit vlastnost `position` na hodnotu „unset“
- Vlastnost `position` neměnit

Zápis do vlastnosti `position` se bude provádět pouze při provedení přechodu automatu, čímž se zefektivní kód. Stav automatu `INSIDE` značí stav, kdy vrchní výřez okna prohlížeče se nachází přímo uvnitř zobrazeného rozvrhu, což znamená, že je vlastnost `position` na hodnotě „fixed“. Stav `OUTSIDE_START` značí stav mimo zobrazený rozvrh, konkrétně nad prvním zobrazeným rozvrhem. Stav `OUTSIDE_END` značí také stav mimo zobrazený rozvrh, ovšem za posledním zobrazeným rozvrhem. Nakonec stav `OUTSIDE` značí stav, kdy vrchní výřez okna prohlížeče se nachází mezi dvěma vykreslenými rozvrhy. Stavy `INSIDE` a `OUTSIDE` se opakují tolikrát, kolik je vykreslených rozvrhů mínus 1. Přechody mezi určitými stavy poté obsahují událost, kdy se skryje nebo zobrazí ukotvené záhlaví rozvrhu.

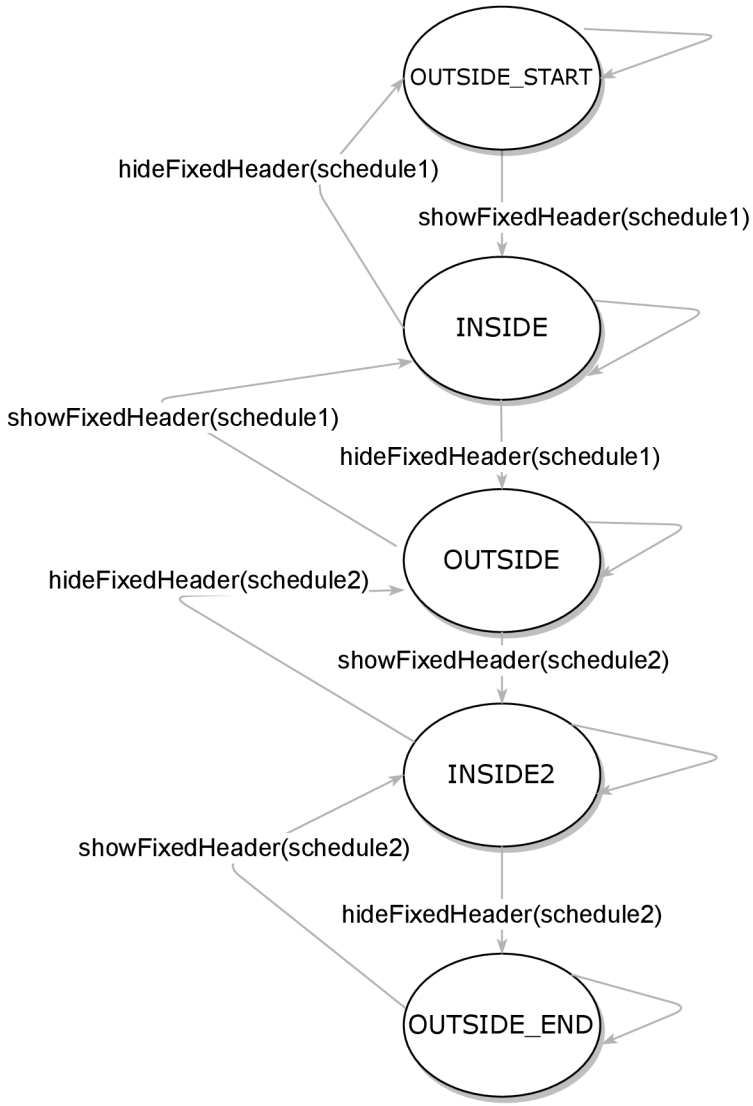
Ukotvení záhlaví sloupců

Jelikož při menší šířce okna prohlížeče mohlo dojít k zobrazení vodorovného posuvníku uvnitř rozvrhu, bylo třeba vyřešit i ukotvení sloupců, aby uživatel při složitějších typech rozvrhů měl vždy přehled, jaký rozvrh v jakém dni, popřípadě studijní skupiny či jiné entity si právě prohlíží. Tohoto mohlo být docíleno nativně skrze CSS. Využil jsem k tomu výše popisovanou vlastnost `position: sticky` v kombinaci s vlastností `left`, která byla nastavena na patřičnou hodnotu.

Ukotvení popisků sloupců v záhlaví

Ukotvení popisků bylo potřeba pro případy, kdy je například rozvrh jednoho dne vyšší, než je výška okna prohlížeče. Poté, pokud se neukotví element s názvem dne, uživatel neuvidí, na jaký den se aktuálně viditelný rozvrh vztahuje.

Tohoto bylo docíleno znovu za pomoci CSS – vlastnostmi `position` a `top`, které byly nastaveny na „sticky“ a vhodnou hodnotu pozice odshora pro prvky HTML uvnitř buněk



Obrázek 5.2: Stavový automat modelující stavy a přechody mezi jednotlivými rozvrhy při skrolování

záhlaví sloupců. Ovšem znovu bylo třeba řešit ukotvení těchto elementů pro menší velikosti oken prohlížeče.

Toto jsem vyřešil tak, že v případě, že se aktuální pozice horního výřezu okna prohlížeče nachází uvnitř rozvrhu, tedy že se automat 5.2 nachází ve stavu `INSIDE`, nastaví se elementu obsahujícímu popisek buňky umístěné právě v horním výřezu okna prohlížeče vlastnost `position fixed`. Pokud se tato buňka dostane mimo horní výřez okna prohlížeče, tato vlastnost se nastaví zpět na hodnotu „unset“. Tímto je možné i při prohlížení sofistikovanějších rozvrhů umožnit uživateli zobrazení stále přehledného rozvrhu.

Vodorovné skrolování rozvrhem

Jelikož při menších rozlišeních dochází k zobrazení externí hlavičky, která není přímo prvkem rozvrhu, musela se implementovat synchronizace pozice skrolování v této externí hlavičce vůči skrolovanému rozvrhu.

Tohoto bylo docíleno skrze nastavení vlastnosti `scrollLeft` externí hlavičky podle načtené hodnoty `scrollLeft` z kontejneru rozvrhu. Aby se zabránilo snaze překreslit změnu `scrollLeft` na hodnotu, která již byla v předchozím kroku nastavena, se ukládá její předchozí hodnota a v případě shody s aktuální hodnotou se zápis do vlastnosti `scrollLeft` externí hlavičky neprovede, aby se nemusely vykonávat složité operace související se zápisem do zmíněné vlastnosti. Přesný popis, jaké operace se musí provést při zápisu do této vlastnosti, lze najít v [2].

Zobrazení více rozvrhů najednou

Při zobrazení více než 3 rozvrhů najednou jsem na starších mobilních zařízeních zpozoroval zasekávání při skrolování. Po následném testování a průzkumu jsem zjistil, že renderování mnoha komplikovaných prvků skrze `display grid`, ale i skrze `display flex`, je pro starší zařízení poměrně náročné a způsobuje zasekávání zejména při onom skrolování. Toto tedy bylo třeba vyřešit.

Profilování kódu

Pro profilování kódu bylo nejprve nutno zprovoznit vývojové nástroje při používání aplikace z mobilního zařízení. Jako testovací zařízení jsem využil Iphone 6s, na kterém se zasekávání projevovalo. Tento telefon disponuje následujícími parametry: 2 GB RAM, dvoujádrový procesor Apple Twister s provozní frekvencí každého jádra 1840 MHz, šestijádrovou grafickou kartu PowerVR GT7600.

Pro profilování na tomto zařízení jsem využil nástroje pro vývojáře v prohlížeči Safari. Obrázek 5.3 zobrazuje časovou osu všech vykonávaných částí kódu při skrolování dokumentem, který obsahoval 28 vykreslených rozvrhů. Tento počet rozvrhů se sice pravděpodobně nikdy nebude při reálném používání zobrazovat, ovšem pro analýzu kódu a lepší projevení výkonnostních problémů jsem profiloval na malinko vyšším množství vykreslených rozvrhů. Z tohoto obrázku si lže všimnout, že kritická vykreslovací cesta je nejvíce blokována červenou částí grafu, což je část `layout` této kritické vykreslovací cesty. Zelené části grafu označují buďto část `vykreslování` nebo `kombinování`. Fialová část grafu poté označuje čas strávený vykonáváním kódů v Javascriptu.

Další obrázek 5.4 zobrazuje profilování při vodorovném skrolování rozvrhem, o které se primárně stará klientský Javascript z důvodů uvedených v sekci 5.2. Na tomto obrázku nelze na první pohled vidět zásadní problém. Při detailnějším prozkoumání grafu si však

Obrázek 5.3: Záznam z profilování uvnitř těla dokumentu bez použití optimalizace

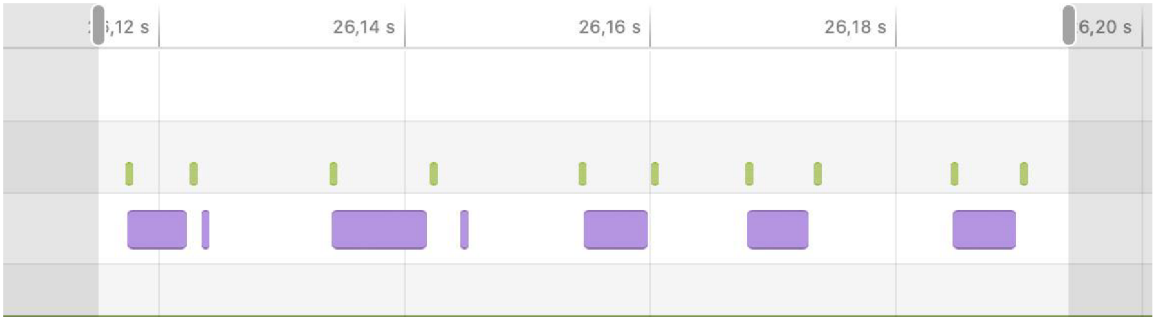
bylo možné všimnou, že obnovovací frekvence (perioda fáze kombinování) se pohybovala v rozmezí asi 15–23 snímků za sekundu, což odpovídá projevům zasekávání skrolování. Toto demonstruje následující přiblížený obrázek. Na tomto obrázku zelené části grafu ilustrují střídavě `vykreslování`, `kombinování`, `vykreslování`, `kombinování` a tak dále v uvedeném pořadí. Jedna perioda střídání těchto dvou částí kritické vykreslovací cesty je 53,45 ms. Tato perioda tedy odpovídá 18,7 snímků za sekundu.

Obrázek 5.4: Záznam z profilování vodorovného skrolování uvnitř rozvrhu bez použití optimalizace

Na základě těchto dat jsem usoudil, že s vysokou pravděpodobností lze optimalizovat takovým způsobem, aby se urychlilo přepočítávání při fázi `layout`. Vyřešil jsem to tím způsobem, že pokud se zobrazují více než 3 rozvrhy, tak rozvrhy, které jsou daleko od aktuálně viditelných rozvrhů ve výřezu okna prohlížeče, jsou skryté pomocí CSS vlastnosti `visibility hidden`. Toto zaručí, že rozvrhy, které aktuálně není třeba vykreslovat, nebudou vykresleny. Naopak rozvrhy, které je třeba vykreslit, neboť jsou ve výřezu okna prohlížeče, se vykreslí a to díky funkci, která přepíše vlastnost `visibility` na „visible“. Takto tedy prohlížeč může například ignorovat rozvržení všech elementů uvnitř elementu, na který byla aplikována tato vlastnost, čímž se pravděpodobně i urychlilo počítání části `layout` kritické vykreslovací cesty. Konkrétně nastalo zrychlení trvání této fáze z asi 100–200 ms na 50–100 ms, tedy zhruba dvojnásobně. Sice kvůli rozšíření kódu v jazyce Javascript vykonávání tohoto kódu trvalo delší dobu oproti neoptimalizované variantě, ovšem tento čas byl stále kratší než vykonávání fáze `layout`, čímž se nesnížila účinnost této optimalizace.

Alternativně by bylo možné použít vlastnost `display: none`, ale ta by teoreticky mohla vyústit ve složité přeuspořádávání elementů na stránce, protože tato hodnota vlastnosti se neukládá do vykreslovacího stromu. Navíc se vlastnost `visibility hidden` hodila pro jednodušší výpočet, zda již má být rozvrh zobrazen či nikoliv.

Po měření druhého profilování bylo naměřeno také zlepšení, které demonstruje obrázek 5.5. Na tomto obrázku lze vidět, že se snížila doba periody mezi jednotlivými vykresleními na displej. Konkrétně jsem naměřil periodu asi 14,21 ms, což v přepočtu dělá asi 70 snímků za sekundu, což je zlepšení oproti původní verzi o 376 %.



Obrázek 5.5: Záznam z profilování vodorovného skrolování uvnitř rozvrhu s použitím optimalizace

Tato měření sice mohou kolísat v závislosti na mnoha faktorech, jako například aktuální zatížení systému, stav nabití baterie zařízení, avšak tato optimalizace bude stále efektivnější oproti neoptimalizované variantě za stejných podmínek provozu. Jedinou změnou mohou být rozdíly v naměřených hodnotách a mírná změna procentuálního vyjádření navýšení efektivity touto optimalizací.

5.3 Tvorba podpůrných komponent pro uživatelské rozhraní rozvrhů

Pro použitelné uživatelské rozhraní rozvrhů jsem musel implementovat kromě samotného vykreslování rozvrhů také několik dalších komponent, které uživateli zjednodušují práci s rozvrhy, jak bylo navrženo v sekci 3.1.

Komponenta pro výběr data

Při implementaci této komponenty bylo zejména potřeba vhodně navrhnout způsob, jak přepočítávat vybrané datum na nejbližší datum pondělí, popřípadě neděle.

Javascriptová funkce `Date.prototype.getDay()` používá následující mapování z čísla dne na jeho odpovídající název:

```
0 -> Neděle
1 -> Pondělí
2 -> Úterý
3 -> Středa
4 -> Čtvrtek
5 -> Pátek
6 -> Sobota
```

Pro výpočet, kolik dní zbývá do pondělí, jsem na základě výše uvedeného mapování využil vzorec 5.1. Ve vzorci `dayID` značí číslo dne, které se typicky určí z nějakého data,

`daysToMonday` značí počet dní, které je třeba od data odečíst, abychom dostali datum nejbližšího pondělí, které se nachází na časové ose směrem dozadu.

$$daysToMonday = (dayID + 6) \ \% \ 7 \tag{5.1}$$

Pro přepočet data naopak na nejbližší neděli se využije výše zmíněný vzorec a k výsledku se připočte číslo 6.

Filtrace podle typu rozvrhové jednotky

Jak bylo uvedeno v návrhu, pro zobrazení možnosti filtrace byla zvolena komponenta pole zaškrtávacích polí. Samotná filtrace pak probíhá tak, že se v datech rozvrhových jednotek vyhledá typ výuky, který uživatel zaškrtne ve vybraném zaškrtávacím poli. Pro zlepšení uživatelského rozhraní často jedno zaškrtávací políčko filtruje více typů rozvrhových aktivit, jako například *Cvičení odborného základu, Jazykové cvičení, Cvičení, Cvičení s počítačovou podporou* jako jednotný filtr pojmenovaný jako „Cvičení“. Poté, pokud se v datech rozvrhových jednotek vyhledá daný typ rozvrhové jednotky, nastaví se k těmto datům příznak `filtered` na hodnotu zaškrtávacího políčka, tedy „false“ / „true“. Poté, když se vytváří skupiny podle algoritmu ze sekce 5.2, se veškeré rozvrhové jednotky, které mají nastavený příznak `filtered`, nedostanou do pole *data* dané skupiny. To tedy znamená, že se nedostanou ani ke zpracování dalšími algoritmy pro zobrazení rozvrhu a tudíž se vůbec nevykreslí.

Filtrace zobrazení dat v rozvrhové jednotce

Při implementaci této filtrace jsem využil následující způsob, který garantuje vždy stejnou pozici vykreslených dat v rozvrhové jednotce, bez ohledu na to, v jakém pořadí se jejich zobrazení bude zapínat / vypínat. Nejprve se vykreslí všechna data, která bude moci uživatel filtrovat. Následně se přepínáním filtrů pouze skrze vyhledávání elementů, které mají podle typu dat nastavený atribut `class` daného elementu HTML, vyhledávají tyto elementy. Nakonec se všem vyhledaným elementům nastaví vlastnost CSS `display` na hodnotu „null“, popřípadě „none“ podle toho, zda mají být viditelné či nikoliv.

5.4 Serverová část pro práci s rozvrhy

Při implementaci serverové části byl využit jeden společný koncový bod, jak již bylo popsáno v návrhu. Pro tento koncový bod jsem pouze implementoval diagramy tříd z přílohy B.

Jelikož databázové dotazy jsou často poměrně náročné na výkon, protože se prochází poměrně velké množství dat, bylo pro urychlení přístupu k datům a snížení zátěže databáze implementováno cachování. Samozřejmě byla snaha optimalizovat i samotné SQL dotazy, ale cachování byla jedna z velmi důležitých optimalizací. O cachování se stará funkce v jazyce PHP implementovaná vývojáři na CVISu, takže pro její úspěšné používání bylo třeba pouze volat její rozhraní.

Tabulka 5.1 demonstruje rozdíly v době čekání na data variant bez použití cache a s jejím použitím.

5.5 Databázové dotazy

Jelikož spousta SQL dotazů již byla v původním řešení, využil jsem toho a snažil jsem se tyto dotazy znovu použít. Jelikož však byl téměř totožný dotaz na cca osmi místech v kódu,

	bez použití cache		s použitím cache	
požadovaný rozvrh	průměr [ms]	medián [ms]	průměr [ms]	medián [ms]
Rozvrh studenta	52,42	42,76	0,87	0,60
Rozvrh vyučujícího	63,11	44,04	1,09	1,13
Rozvrh místnosti	145,37	107,13	3,33	2,23
Rozvrh předmětu	21,8	12,46	0,34	0,217
Rozvrh přednáškových skupin	763,46	664,07	16,38	9,75
Rozvrh studijních skupin	38,25	18,29	0,72	0,53

Tabulka 5.1: Rozdíly v době čekání na data z databáze při použití varianty bez a s cache

nejprve jsem musel tyto SQL dotazy porovnat abych zjistil, kde konkrétně se dotazy liší a z jakého důvodu. Z tohoto porovnávání jsem zjistil, že změny byly často bez vlivu na výsledek dotazů, nebo se pravděpodobně jednalo o opravy chyb, které někdo zapomněl opravit na dalších místech, kde to bylo také potřeba. Poté jsem se snažil každý z dotazů detailně pochopit a pokusil se jej optimalizovat.

Nejsložitější a zároveň hlavní dotaz pro zobrazení rozvrhu byl dotaz pro získání výuky v předmětech. ER diagram použitý pro implementaci tohoto SQL dotazu byl popsán v sekci 4.9. Jak jsem již zmínil výše, po zkoumání SQL dotazů v původním řešení jsem zjistil, že dotazy jsou téměř identické. Liší se pouze v prvním `WITH` dotazu, ve kterém se vyberou všechna potřebná data a poté se se nad těmito daty provádí totožné operace. Dále jsem si všiml, že se jako výsledek dotazu vrací několik výsledků se shodným `ID`, lišících se pouze daty, která se do výsledku přidala skrze `JOIN` a tudíž se původní data znásobila. Abych tomuto předešel, využil jsem dvě funkce, `JSON_OBJECT`, která vytvoří JSON s požadovanými vlastnostmi s požadovanou hodnotou, a funkci `JSON_ARRAGG`, která agreguje data z SQL skupin vytvořených skrze `GROUP BY`, nebo data z okének vytvořených skrze `PARTITION BY`. Díky tomuto tak bylo možné místo duplikace původního výsledku skrze spojení tabulek s vyučujícími a místnostmi do původního výsledku v JSON vložit pole vyučujících a pole místností. Jediný problém, který se zde musel řešit, byl nevhodnost výchozího návratového typu funkce `JSON_ARRAGG`. Výchozí datový typ je `VARCHAR2(4000)`, což ale pro tyto účely byl příliš malý typ. Při překročení velikosti tohoto datového typu databázový server vyhazoval výjimku, kvůli které potom nebyla získána žádná data. Toto se vyřešilo nastavením typu `CLOB` jako návratového datového typu funkce. `CLOB` je omezený velikostí 2 GiB.

Jelikož se odpověď na dotaz odesílá ve formátu JSON, bylo toto vhodné i z tohoto důvodu. Níže je ukázka části SQL dotazu, která implementuje tuto transformaci do JSON. Tento dotaz předpokládá tabulku `vyuc_ss`, která obsahuje všechny již vyfiltrované rozvrhové jednotky, ke kterým dále zjišťuje patřičné vyučující, data konání, či místnosti. Pro zpřehlednění dotazu se však v tomto příkladu neprovádí získávání všech místností, ovšem tato část dotazu vypadá velmi podobně jako část dotazu pro získání všech vyučujících.

```
SELECT
vyuc_ss.*,
(
    SELECT
        DISTINCT JSON_ARRAYAGG(
            JSON_OBJECT(
                KEY 'teachers' VALUE (
```

```
                    SELECT
                        DISTINCT JSON_ARRAYAGG(
                            JSON_OBJECT(
                                KEY 'name'
                                    VALUE vyuc_ss_osoby2.name,
                                KEY 'surname'
                                    VALUE vyuc_ss_osoby2.surname,
                                KEY 'title_before'
                                    VALUE vyuc_ss_osoby2.title_before,
                                KEY 'title_after'
                                    VALUE vyuc_ss_osoby2.title_after,
                                KEY 'id'
                                    VALUE vyuc_ss_osoby2.vyucujici_id
                            ) RETURNING CLOB
                        )
                    FROM
                        vyuc_ss_osoby vyuc_ss_osoby2
                    WHERE
                        vyuc_ss_osoby2.vyucovani_blok_id =
                        vyuc_ss_osoby.vyucovani_blok_id
                        AND vyuc_ss_osoby2.datum = vyuc_ss_osoby.datum
                ),
                KEY 'date' VALUE vyuc_ss_osoby.datum,
                KEY 'deleted' VALUE vyuc_ss_osoby.zruseno
            ) RETURNING CLOB
        )
    FROM
        vyuc_ss_osoby
    WHERE
        vyuc_ss_osoby.vyucovani_blok_id = vyuc_ss.vyucovani_blok_id
) AS blocks
FROM
    vyuc_ss
```

Kromě této úpravy původních SQL dotazů jsem se také snažil SQL dotazy optimalizovat. Zejména bylo vhodné odstranit podmínky z části `JOIN` a přesunout do části `WHERE`. Díky tomuto databázový server může nejprve omezit data na co nejmenší možný počet řádků a až poté se snaží provádět spojování tabulek, což je doporučovaný postup. Jako jeden z příkladů lze uvést převod následující části `JOIN` na ekvivalentní část `WHERE`. Tento příklad předpokládá, že tabulka `table` obsahuje cizí klíč na tabulku `místnost`. Tabulka `mistnost` poté obsahuje sloupec `status`, který slouží jako tzv. **soft delete**. Účelem spojení tabulek je tedy spojit pouze řádky z `table` s takovými místnostmi, které nejsou z databáze odstraněny, nebo jiným způsobem zneplatněny.

```
JOIN st01.mistnost m
    ON (m.mistnost_id = table.mistnost_id
        AND m.status = 9)

JOIN st01.mistnost m
    ON (m.mistnost_id = table.mistnost_id)
```

```
WHERE m.status = 9
```

Podobným způsobem jsem implementoval i další SQL dotazy, například pro výpis rezervovaných místností, termíny zkoušek a jejich přihlašování, projektů, komisí, nebo i konzultací vyučujících.

Kapitola 6

Testování vytvořeného rozhraní

V rámci této práce jsem využíval několik druhů testování. Od jednotkových testů po testování uživatelské přívětivosti s uživateli. Jednotlivé druhy testování budou popsány v následujících sekcích.

6.1 Jednotkové testy a zvolené testovací frameworky

Jak již bylo zmíněno v sekci 4.5, pro testování kódu v jazyce Javascript byla zvolena knihovna Jest. Pro testování webových aplikací je podstatné oddělit vykreslovací logiku od zpracování dat, aby bylo toto rozhraní co nejlépe testovatelné. To jsem se snažil splnit a většinu funkcí, které pracují nad daty, jsem implementoval jako samostatné třídy. Důležité metody v těchto třídách jsem poté dále testoval skrze jednotkové testy.

6.2 Testování uživatelské přívětivosti

Pro testování uživatelské přívětivosti jsem navrhl testovací scénář, který jsem poté nechal uživatele vykonávat a zaznamenával jsem si jejich časy potřebné pro vykonání daného úkolu včetně jejich subjektivních pocitů při používání rozhraní. S několika vybranými uživateli jsem prováděl dva testy, kdy cílem prvního testu bylo určit, jak obtížné je používat rozhraní při prvním kontaktu s rozhraním. Cílem druhého testu bylo zjistit, jak se použitelnost rozhraní zvýší oproti prvnímu použití. Tyto testy prováděli uživatelé z různých fakult VUT, včetně studentů a jednoho vyučujícího.

Tabulka 6.1 zobrazuje jednotlivé úkoly testovacích scénářů a výsledky měření s jednotlivými uživateli při používání aplikace na počítači. Označení `Ux` značí uživatele číslo `x`, který rozhraní používal. Uživatel s označením `U3` je vyučujícím. Z této tabulky lze usuzovat, že mnoho prvků rozhraní je umístěno na správném místě a lze je snadno používat. Některé prvky rozhraní uživatelé hledali delší dobu, ale všichni sami v závěru testování zmínili, že komponenty jsou umístěné správně, pouze je nutno se při prvním používání mírně zorientovat.

Tabulka 6.2 zobrazuje úkoly testovacích scénářů a výsledky měření s vybranými uživateli při používání aplikace z mobilního zařízení. Uživatel U1 používal aplikaci z mobilního zařízení týden po použití aplikace z počítače z předchozí tabulky. Oproti tomu uživatel U2 použil mobilní zařízení nejdříve a týden poté testoval rozhraní z počítače z předchozí tabulky. I z této tabulky lze usuzovat, že rozložení komponent na mobilním zařízení je

	Čas potřebný pro vykonání úkolu respondetem [s]						
Úkol	U1	U2	U3	U4	U5	U6	U7
1) Zobrazte si rozvrh pro aktuální týden a vyjmenujte počáteční a koncový čas vybrané rozvrhové jednotky	7	6	25	13	26	16	21
2) U vybrané rozvrhové jednotky zjistěte následující: vyučující, místnost, typ výuky a typ týdne, ve kterém se dané rozvrhové okno vyučuje	4	5	17	10	9	13	18,8
3) Vytiskněte si rozvrh	18	7	9	15	20	6	12
4) Změňte interval týdnů, pro které chcete zobrazit rozvrh o 2 týdny vzad (počáteční týden) a 1 týden vpřed (koncový týden)	7	3	3	20	5	7	15
5) Určete, pro jaké data platí zobrazený filtr (pro jaké data se aplikoval výše uvedený filtr)	5	7	5,5	4	9	7	5
6) Zkuste si zobrazit standardní rozvrh a zjistěte, v které data je libovolná rozvrhová jednotka vyučována	20	6	14	19,5	10	18	32
7) Zkuste smazat vyučující z rozvrhových oken	10	4	19	8	5	2,6	4
8) Vyfiltrujte pryč z rozvrhu přednášky	6	3	12	8	2	6,8	4
9) Zkuste exportovat rozvrh do formátu iCal, který slouží pro export / import rozvrhu do externích aplikací	9	3	3	3	1,5	3	8
10) Zobrazte si legendy k rozvrhu a tyto legendy najděte na stránce	60	18	32	20	9	25	32
11) Transponujte si rozvrh (prohození os dnů a hodin)	8	4	12	5	7	14	6
12) Zobrazte si detail rozvrhového okna pouze najetím myší	3,5	10	50	11	43	10	45
13) Zobrazte si volné dny	10	5	28	35	3	4	28
14) Zobrazte si rozvrh přednáškové skupiny 1BIA	34	10	63	77	20	45	16
15) Odstraňte zrušenou výuku	7	9	15	5	3	3	3
16) Zobrazte si svůj rozvrh a rozvrh doktora Aleše Smrčky (najednou)	39	15	55	60	24	35	35
17) Zobrazte si rozvrh předmětu ISA (Síťové aplikace a správa sítí)	15	11	46	45	25	20	17
18) Přejděte na editaci vypsané zkoušky skrze odkaz v rozvrhu	4	3	32	4	5	7	5,5

Tabulka 6.1: Tabulka zobrazující úkoly jednotlivých testovacích scénářů a výsledky měření jednotlivých uživatelů při používání aplikace z počítače

vhodné. Také lze vidět, že při opětovném použití aplikace po týdnu se rychlost vykonání jednotlivých úkolů zvýšila bez ohledu na použité zařízení.

Nakonec jsem s vybranými uživateli porovnával čas potřebný pro vykonání akce v původním systému a v aplikaci, která je výstupem této práce. Například vykonání úkolu **1** z testovacího scénáře se urychlilo v průměru o 33 %, úkol **14** se urychlil o 25 %, úkol **17** o 290 % a úkol **16** dokonce o 514 %. Jelikož se však jedná o pouhé testovací scénáře, v praxi při reálném používání aplikace tento rozdíl může být daleko vyšší, neboť při používání aplikace v praxi je nutno, aby se uživatel více soustředil na zobrazené informace v rozvrhu.

Po provedení testování s jednotlivými uživateli jsem od uživatelů zjistil, že rozhraní se jim zdá dostatečně přívětivé a neshledali v něm žádné podstatné nedostatky. Pouze často zmiňovali mírné grafické nedostatky, které by je však neodradily od používání nové aplikace. Ocenili zejména funkce pro ukotvení záhlaví sloupců a řádků, možnost vyfiltrovat si z rozvrhu různé druhy aktivit, zobrazení aktuálního času, možnost zobrazovat rozvrh více entit najednou, zejména pak v případě porovnávání rozvrhů dvou a více osob, a vyučující dále ocenil možnost skrýt zrušenou výuku.

	Čas potřebný pro vykonání úkolu respondetem [s]	
Úkol	U1	U2
1) Zobrazte si rozvrh pro aktuální týden a vyjmenujte počáteční a koncový čas vybrané rozvrhové jednotky	7	60
2) U vybrané rozvrhové jednotky zjistěte následující: vyučující, místnost, typ výuky a typ týdne, ve kterém se dané rozvrhové okno vyučuje	8	7
3) Vytiskněte si rozvrh	12	20
4) Změňte interval týdnů, pro které chcete zobrazit rozvrh, o 2 týdny vzad (počáteční týden) a 1 týden vpřed (koncový týden)	5	4
5) Určete, pro jaká data platí zobrazený filtr (pro jaká data se aplikoval výše uvedený filtr)	5	4
6) Zkuste si zobrazit standardní rozvrh a zjistěte, ve kterém datu je libovolná rozvrhová jednotka vyučována	18	20
7) Zkuste smazat vyučující z rozvrhových oken	6	15
8) Vyfiltrujte pryč z rozvrhu přednášky	3	4
9) Zkuste exportovat rozvrh do formátu iCal, který slouží pro export / import rozvrhu do externích aplikací	10	2
10) Zobrazte si legendy k rozvrhu a tyto legendy najděte na stránce	5	11
11) Transponujte si rozvrh (prohození os dnů a hodin)	3	4
12) Zobrazte si detail rozvrhového okna pouze najetím myší	–	–
13) Zobrazte si volné dny	18	30
14) Zobrazte si rozvrh přednáškové skupiny 1BIA	15	3,7
15) Odstraňte zrušenou výuku	4	5
16) Zobrazte si svůj rozvrh a rozvrh doktora Aleše Smrčky (najednou)	20	45
17) Zobrazte si rozvrh předmětu ISA (Síťové aplikace a správa sítí)	18	30
18) Přejděte na editaci vypsané zkoušky skrze odkaz v rozvrhu	6	4

Tabulka 6.2: Tabulka zobrazující úkoly jednotlivých testovacích scénářů a výsledky měření jednotlivých uživatelů při používání aplikace z mobilního zařízení

Kapitola 7

Závěr

Cílem této práce bylo zlepšit uživatelskou přívětivost rozvrhů v informačním systému VUT. Tento cíl byl splněn, neboť i na základě reakcí uživatelů nové uživatelské rozhraní značně ulehčuje práci s rozvrhy a přináší celou řadu nových užitečných funkcí.

V této práci jsem implementoval dohromady více než 12 nových funkcí uživatelského rozhraní, o které měli uživatelé zájem. Jmenovitě se jedná například o ukotvené záhlaví sloupců a řádků, včetně ukotvení popisků řádků, přepínání způsobu zobrazení detailu o rozvrhové jednotce, transponace rozvrhu, filtrace podle typu rozvrhové jednotky, skrývání a zobrazování informací o rozvrhové jednotce, nastavení formátu tisku na standardizovaný formát, komponenta pro výběr intervalu dat pro zobrazení rozvrhů, spojení studijních skupin se shodnou výukou do jednoho řádku, hromadný rozvrh osob, zobrazení aktuálního času a několik dalších funkcí. Současně jsem však implementoval původní funkce s ohledem na modernější technologie a postupy. Do rozvrhů jsem přidal vykreslování celodenních aktivit a upozornění na blížící se termíny, jako například termíny zkoušek a projektů.

Při testování uživatelské přívětivosti jsem zjistil, že aplikace je z pohledu uživatelů poměrně snadno pochopitelná a použitelná. Uživatelé měli v zásadě pozitivní ohlasy a spoustu nových funkcí rozhraní ocenili. Po porovnání časů potřebných pro vykonání operací s rozvrhy jsem naměřil zlepšení nové aplikace oproti původnímu informačnímu systému o cca 22 % u jednodušších úkolů, o asi 290 % u složitějších úkolů a zlepšení až o 514 % u úkolů, které v původním systému nebylo možné vykonat a musely se různě obcházet. Z tohoto tedy vyplývá, že nové uživatelské rozhraní je navrženo správně a přináší řadu výhod uživatelům.

Při psaní této práce jsem si prohloubil znalosti programovacích jazyků jako jsou PHP, Javascript a SQL. Také jsem se díky této práci více vžil do problematiky psaní a analýzy aplikací s ohledem na uživatelskou přívětivost, včetně lokalizace problematických komponent rozhraní. Také jsem absolvoval proces návrhu uživatelského rozhraní až po jeho samotnou implementaci, díky čemuž jsem získal mnoho cenných zkušeností v oblasti vývoje uživatelsky přívětivých aplikací. Nakonec jsem si rozšířil vědomosti z oblasti posuzování a testování uživatelské přívětivosti navrženého rozhraní.

Do budoucna bych chtěl práci dále rozšířit především o další typy rozvrhů, jako jsou rezervace místností a registrace vyučování. Také bych se chtěl zasadit o lepší uživatelskou přívětivost z pohledu vyučujících při vytváření výuky, termínů, zkoušek nebo dalších položek, které souvisí s problematikou rozvrhů. Nakonec bych chtěl k této práci přidat možnost editace rozvrhových jednotek a informací o ní přímo u vykreslené rozvrhové jednotky, což budu znovu muset nejprve konzultovat s uživateli a zjistit jejich detailnější požadavky a názory na tuto funkci.

Literatura

[1] Desruisseaux, B. *Internet Calendaring and Scheduling Core Object Specification (iCalendar)* [RFC 5545]. RFC Editor, září 2009. DOI: 10.17487/RFC5545. Dostupné z: https://www.rfc-editor.org/info/rfc5545.

[2] Fraser, S., Álvarez, E. C., Pieters, S., Adams, G. a Kesteren, A. van. *CSSOM View Module* [online]. W3C, prosinec 2022. Dostupné z: https://w3c.github.io/csswg-drafts/cssom-view/#dom-element-scrollleft.

[3] Glasow, P. A. *Fundamentals of Survey Research Methodology* [online]. MITRE, duben 2005. Dostupné z: https://www.mitre.org/sites/default/files/pdf/05_0638.pdf.

[4] Hiwarale, U. *How the browser renders a web page? — DOM, CSSOM, and Rendering* [online]. jsPoint, srpen 2019 [cit. 2023-04-16]. Dostupné z: https://medium.com/jspoint/how-the-browser-renders-a-web-page-dom-cssom-and-rendering-df10531c9969.

[5] Li, B. S. *React: Create maintainable, high-performance UI components - IBM Developer* [online]. IBM Developer, 2015 [cit. 2023-04-15]. Dostupné z: https://developer.ibm.com/tutorials/wa-react-intro/.

[6] Patel, M. K. *HTML, CSS, Bootstrap, Javascript and jQuery* [online]. 2017 [cit. 2023-04-16]. Dostupné z: https://buildmedia.readthedocs.org/media/pdf/htmlguide/latest/htmlguide.pdf.

[7] Powell, T. A. *The complete reference HTML & CSS*. 5. vyd. The McGraw-Hill Companies, 2010. ISBN 978-0-07-174170-5.

[8] Source, M. O. *Quick Start – React* [online]. Meta Open Source [cit. 2023-03-20]. Dostupné z: https://react.dev/learn.

Příloha A

Tabulky

Způsob zobrazení okna s podrobnějšími informacemi

Způsob výpočtu statistik v tabulce A.1:
0 = Po kliknutí na rozvrhovou jednotku
1 = Po najetí myší na rozvrhovou jednotku
0,5 = Chci mít možnost si změnit nastavení způsobu zobrazování tohoto okna na webu, někdy se mi to hodí více po kliknutí, jindy po najetí myší

Modul	**Arit. průměr**	**Směrodatná odchylka**	**Modus**	**Medián**	**Výchozí způsob zobrazení detailu**
Individuální rozvrh studentů	0,46	0,42	0	0,5	Po kliknutí
Individuální rozvrh vyučujících	0,40	0,46	0	0	Po kliknutí
Rozvrh předmětů	0,39	0,46	0	0	Po kliknutí
Rozvrh místností a hromadný rozvrh místností	0,41	0,46	0	0	Po kliknutí
Rozvrh přednáškových skupin a rozvrh skupin	0,43	0,47	0	0	Po kliknutí
Rozvrh Výuka zapsaných studentů	0,40	0,45	0	0,25	Po kliknutí

Tabulka A.1: Vyhodnocené odpovědi na otázku zabývající se způsobem zobrazení detailu o rozvrhové jednotce v jednotlivých modulech

Odkazy v rozvrhové jednotce

V tabulkách A.2, A.3 je vidět, jak respondenti odpovídali na tuto otázku v rámci jednotlivých modulů. Statistiky se vypočítávaly následujícím způsobem:
0 = Odkaz v rozvrhové jednotce zavazí
0,5 = Odkaz v rozvrhové jednotce nezavazí

1 = Odkaz je v rozvrhové jednotce důležitý

Důležitost odkazů – odpovědi studentů					
Typ odkazu	**Arit. průměr**	**Směrodatná odchylka**	**Modus**	**Medián**	**Umístění odkazu**
Odkaz místnosti	0,54	0,34	0,5	0,5	V detailu
Odkaz vyučujícího	0,53	0,31	0,5	0,5	V detailu
Odkaz předmětu	0,78	0,28	1	1	V rozvrhové jednotce

Tabulka A.2: Vyhodnocené odpovědi studentů na otázku zabývající se odkazy v rozvrhové jednotce

Důležitost odkazů – odpovědi vyučujících					
Typ odkazu	**Arit. průměr**	**Směrodatná odchylka**	**Modus**	**Medián**	**Umístění odkazu**
Odkaz místnosti	0,72	0,27	0,5	0,5	V rozvrhové jednotce
Odkaz vyučujícího	0,66	0,29	0,5	0,5	V rozvrhové jednotce
Odkaz předmětu	0,72	0,28	0,5	0,5	V rozvrhové jednotce
Odkaz se seznamem studentů	0,73	0,32	1	1	V rozvrhové jednotce

Tabulka A.3: Vyhodnocené odpovědi vyučujících na otázku zabývající se odkazy v rozvrhové jednotce

Důležitost informací o rozvrhové jednotce

Při výpočtu následujících statistik byly odpovědím respondentů přiřazeny následující hodnoty:
0 = Nejméně důležitá informace
2 = Středně důležitá informace
4 = Nejvíce důležitá informace

Důležitost informací – individuální rozvrh studentů					
Informace	**Arit. průměr**	**Směrodatná odchylka**	**Modus**	**Medián**	**Zobrazení informace**
Jméno vyučujícího	1,95	1,17	2	2	V detailu, v rozvrhové jednotce pouze první písmeno ze jména
Příjmení vyučujícího	2,82	1,06	3	3	V rozvrhové jednotce
Titul vyučujícího	1,68	1,23	2	2	V detailu
Místnost, kde se vyučuje	3,92	0,33	4	4	V rozvrhové jednotce
Areál, kde se místnost nachází	3,01	1,18	4	3	V rozvrhové jednotce
Studijní skupina	1,39	1,04	1	1	V detailu
Obsazenost	1,40	1,05	1	1	V detailu
Odkaz na výukové materiály	1,72	1,08	2	2	V detailu
Odkaz na rozvrh daného předmětu	1,60	0,95	2	2	V detailu
Seznam registrovaných studentů	1,39	1,07	1	1	V detailu
Typ týdne (sudý / lichý)	3,32	0,92	4	4	V rozvrhové jednotce
Počáteční a koncový týden, kdy se jednotka vyučuje	1,83	1,10	2	2	V detailu
Seznam týdnů, kdy se jednotka vyučuje	1,86	1,10	2	2	V detailu

Tabulka A.4: Vyhodnocené odpovědi studentů na otázku zabývající se důležitostí informací o rozvrhové jednotce

Důležitost informací – individuální rozvrh vyučujících					
Informace	**Arit. průměr**	**Směrodatná odchylka**	**Modus**	**Medián**	**Zobrazení informace**
Jméno vyučujícího	2,27	1,39	4	2	V detailu, v rozvrhové jednotce pouze první písmeno ze jména
Příjmení vyučujícího	3,28	1,1	4	4	V rozvrhové jednotce
Titul vyučujícího	1,0	1,0	0	1	V detailu
Místnost, kde se vyučuje	3,92	0,28	4	4	V rozvrhové jednotce
Areál, kde se místnost nachází	2,10	1,27	2	2	V detailu
Studijní skupina	2,46	1,33	4	3	V detailu co nejvýše
Obsazenost	1,95	1,16	2	2	V detailu
Odkaz na výukové materiály	1,47	0,98	2	2	V detailu
Odkaz na rozvrh daného předmětu	1,87	1,03	2	2	V detailu
Seznam registrovaných studentů	2,29	1,15	2	2	V detailu
Typ týdne (sudý / lichý)	2,81	1,21	4	3	V rozvrhové jednotce
Počáteční a koncový týden, kdy se jednotka vyučuje	1,54	1,06	1	1,5	V detailu
Seznam týdnů, kdy se jednotka vyučuje	1,54	1,02	2	2	V detailu

Tabulka A.5: Vyhodnocené odpovědi vyučujících na otázku zabývající se důležitostí informací o rozvrhové jednotce v individuálním rozvrhu

Důležitost informací – rozvrh předmětů					
Informace	**Arit. průměr**	**Směrodatná odchylka**	**Modus**	**Medián**	**Zobrazení informace**
Jméno vyučujícího	2,72	1,24	4	3	V detailu, v rozvrhové jednotce pouze první písmeno ze jména
Příjmení vyučujícího	3,83	0,38	4	4	V rozvrhové jednotce
Titul vyučujícího	1,33	1,31	0	1	V detailu
Místnost, kde se vyučuje	3,9	0,30	4	4	V rozvrhové jednotce
Areál, kde se místnost nachází	2,15	1,28	2	2	V detailu
Studijní skupina	2,52	1,48	4	3	V detailu co nejvýše
Obsazenost	2,20	1,29	2	2	V detailu
Odkaz na výukové materiály	1,43	1,23	0	1	Nikde
Odkaz na rozvrh daného předmětu	1,67	1,25	2	2	V detailu
Seznam registrovaných studentů	2,15	1,28	2	2	V detailu
Typ týdne (sudý / lichý)	2,77	1,32	4	3	V rozvrhové jednotce
Počáteční a koncový týden, kdy se jednotka vyučuje	1,75	1,14	1	2	V detailu
Seznam týdnů, kdy se jednotka vyučuje	1,70	1,26	2	2	V detailu

Tabulka A.6: Vyhodnocené odpovědi vyučujících na otázku zabývající se důležitostí informací o rozvrhové jednotce v rozvrhu předmětů

Důležitost informací – rozvrh místností a hromadný rozvrh místností					
Informace	**Arit. průměr**	**Směrodatná odchylka**	**Modus**	**Medián**	**Zobrazení informace**
Jméno vyučujícího	2,41	1,34	4	2	V detailu, v rozvrhové jednotce pouze první písmeno ze jména
Příjmení vyučujícího	3.63	0,74	4	4	V rozvrhové jednotce
Titul vyučujícího	1,25	1,24	0	1	V detailu
Místnost, kde se vyučuje	2,83	1,45	4	4	V rozvrhové jednotce
Areál, kde se místnost nachází	1,97	1,38	2	2	V detailu
Studijní skupina	2,42	1,45	4	2	V detailu
Obsazenost	1,75	1,18	1	2	V detailu
Odkaz na výukové materiály	0,91	0,94	0	1	Nikde
Odkaz na rozvrh daného předmětu	1,75	1,09	2	2	V detailu
Seznam registrovaných studentů	1,39	1,20	1	1	Nikde
Typ týdne (sudý / lichý)	2,73	1,32	4	3	V rozvrhové jednotce
Počáteční a koncový týden, kdy se jednotka vyučuje	2,02	1,32	1	2	V detailu
Seznam týdnů, kdy se jednotka vyučuje	1,85	1,30	1	2	V detailu

Tabulka A.7: Vyhodnocené odpovědi vyučujících na otázku zabývající se důležitostí informací o rozvrhové jednotce v rozvrhu místností

Důležitost informací – rozvrh přednáškových skupin a rozvrh skupin					
Informace	**Arit. průměr**	**Směrodatná odchylka**	**Modus**	**Medián**	**Zobrazení informace**
Jméno vyučujícího	3,04	1,12	4	3,5	V detailu, v rozvrhové jednotce pouze první písmeno ze jména
Příjmení vyučujícího	3,81	0,40	4	4	V rozvrhové jednotce
Titul vyučujícího	2,06	0,66	2	2	V detailu
Místnost, kde se vyučuje	3,73	0,45	4	4	V rozvrhové jednotce
Areál, kde se místnost nachází	2,5	1,19	1	2,5	V detailu
Studijní skupina	3,41	0,67	4	3,5	V detailu co nejvýše
Obsazenost	2,57	1,12	2	2	V detailu
Odkaz na výukové materiály	1,33	0,49	1	1	Nikde
Odkaz na rozvrh daného předmětu	1,8	0,77	1	2	V detailu
Seznam registrovaných studentů	2,25	1,02	3	2	V detailu
Typ týdne (sudý / lichý)	2,88	1,19	4	3	V rozvrhové jednotce
Počáteční a koncový týden, kdy se jednotka vyučuje	2,26	1,19	1	2	V detailu
Seznam týdnů, kdy se jednotka vyučuje	1,94	1,16	1	1,5	V detailu

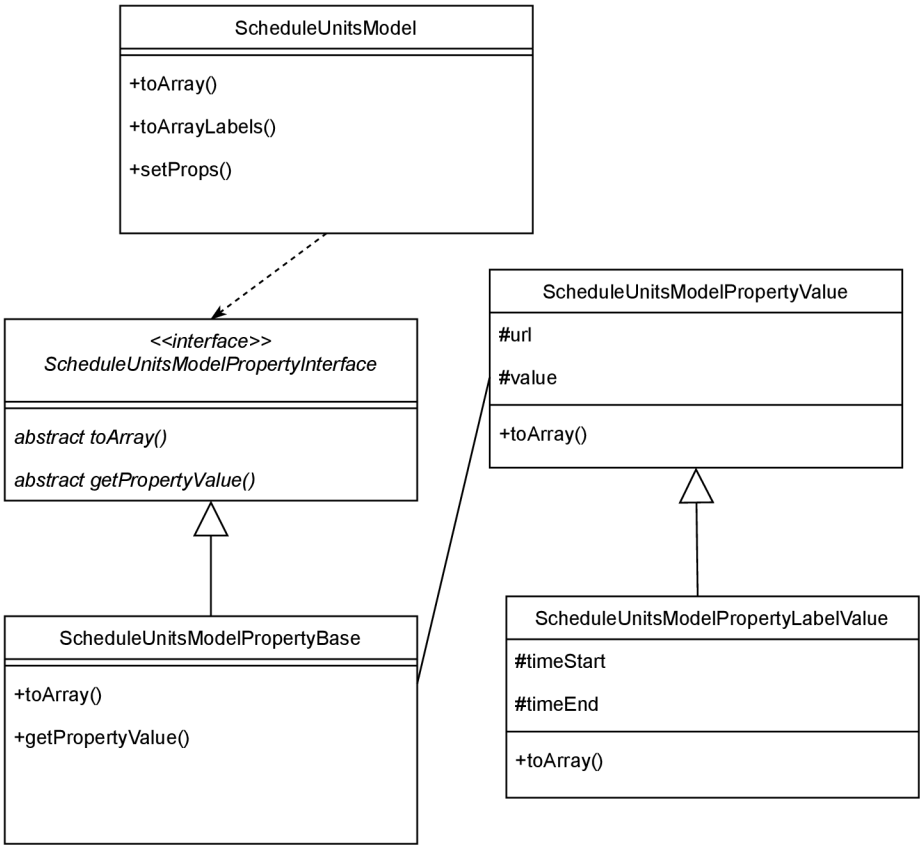
Tabulka A.8: Vyhodnocené odpovědi vyučujících na otázku zabývající se důležitostí informací o rozvrhové jednotce v rozvrhu přednáškových skupin a rozvrhu skupin

Důležitost informací – rozvrh Výuka zapsaných studentů					
Informace	**Arit. průměr**	**Směrodatná odchylka**	**Modus**	**Medián**	**Zobrazení informace**
Jméno vyučujícího	2,3	1,22	2	2	V detailu, v rozvrhové jednotce pouze první písmeno ze jména
Příjmení vyučujícího	3,53	0,51	4	4	V rozvrhové jednotce
Titul vyučujícího	1,2	1,06	0	1	V detailu
Místnost, kde se vyučuje	3,5	0,51	3	3,5	V rozvrhové jednotce
Areál, kde se místnost nachází	1,6	1,47	0	2	V detailu
Studijní skupina	2,53	1,39	2	3	V detailu
Obsazenost	1,79	1,40	2	2	V detailu
Odkaz na výukové materiály	0,95	0,91	0	1	Nikde
Odkaz na rozvrh daného předmětu	1,84	1,30	2	2	V detailu
Seznam registrovaných studentů	1,95	1,43	0	2	V detailu
Typ týdne (sudý / lichý)	2,9	1,21	3	3	V rozvrhové jednotce
Počáteční a koncový týden, kdy se jednotka vyučuje	2,16	1,50	0	2	V detailu
Seznam týdnů, kdy se jednotka vyučuje	2,0	1,41	2	2	V detailu

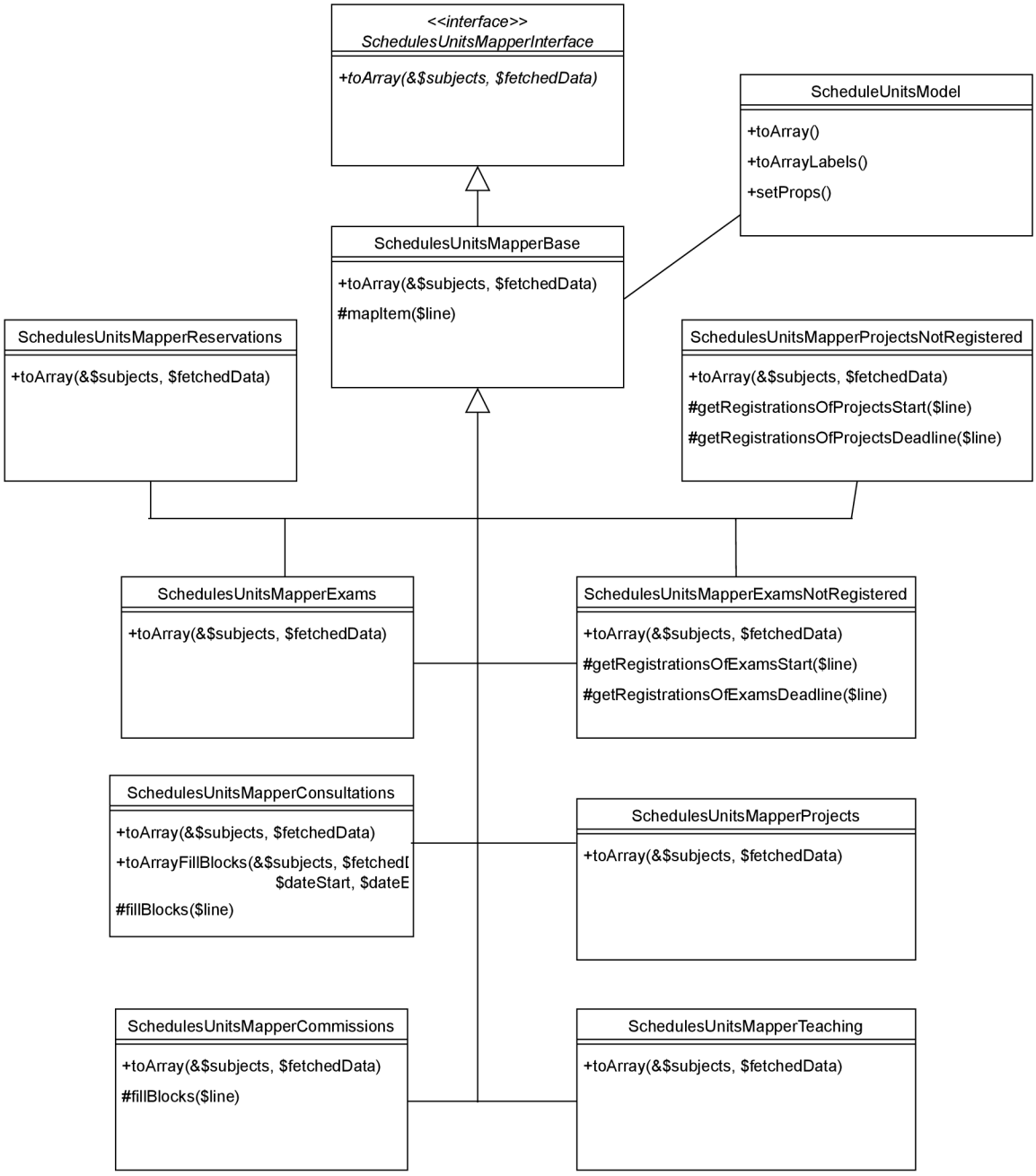
Tabulka A.9: Vyhodnocené odpovědi vyučujících na otázku zabývající se důležitostí informací o rozvrhové jednotce v rozvrhu Výuka zapsaných studentů

Příloha B

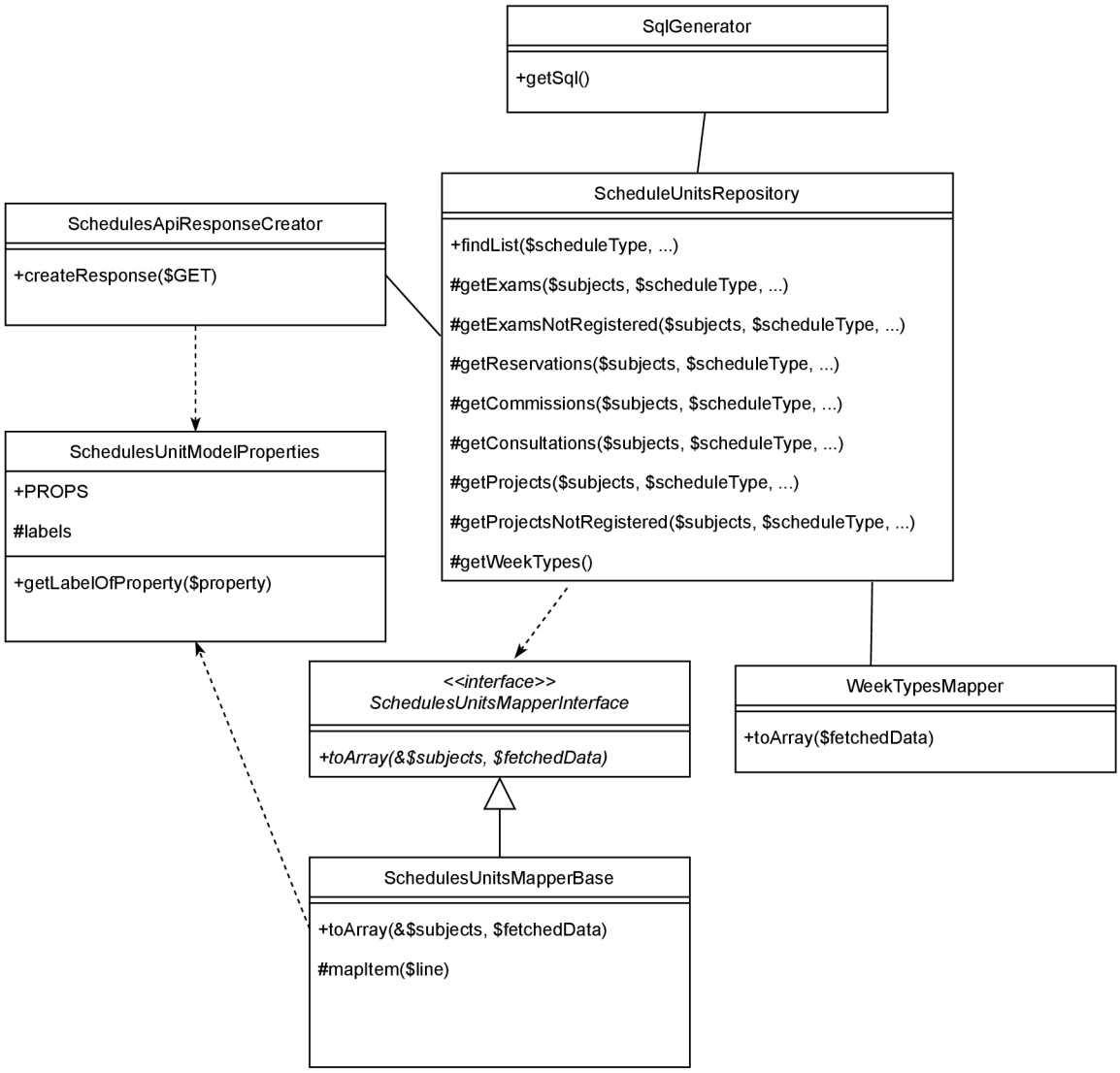
Diagramy tříd



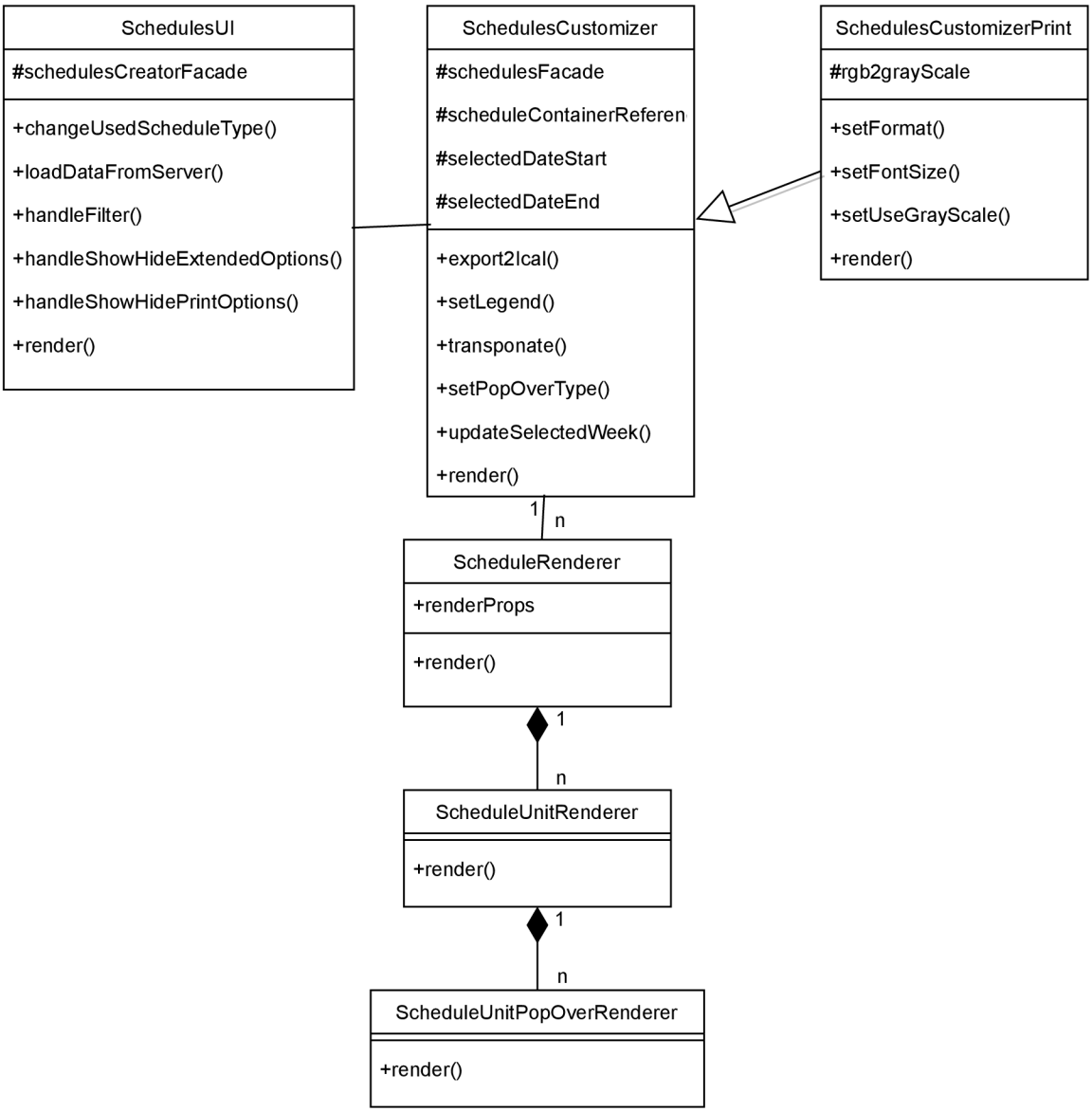
Obrázek B.1: Diagram tříd znázorňující navrženou architekturu modelů a jejich vlastností



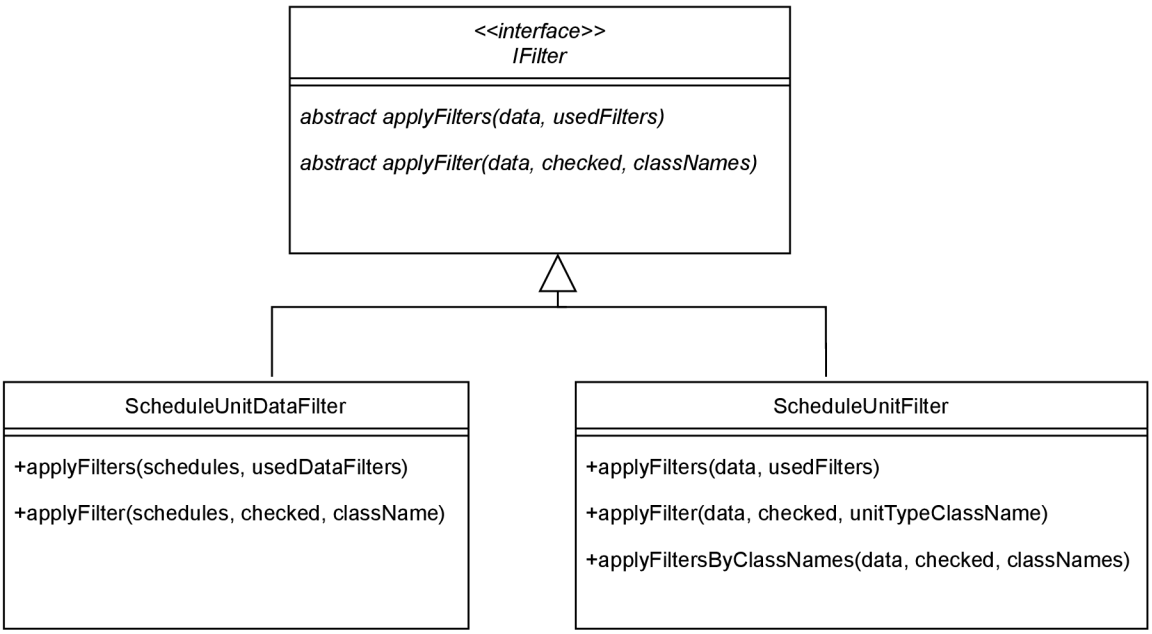
Obrázek B.2: Diagram tříd znázorňující navrženou architekturu mapování dat z databáze na modely



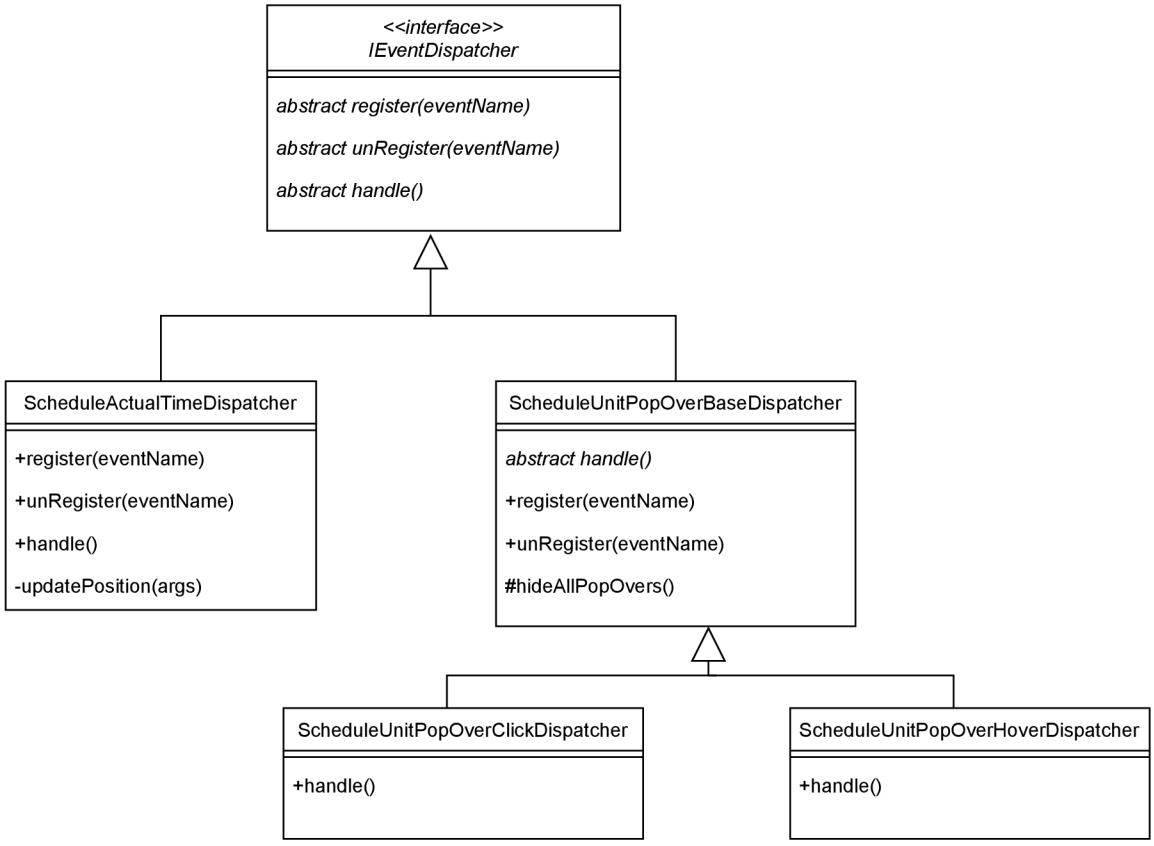
Obrázek B.3: Diagram tříd znázorňující navrženou architekturu využívanou v rámci celého procesu načtení dat z databáze až po vytvoření výstupního formátu JSON



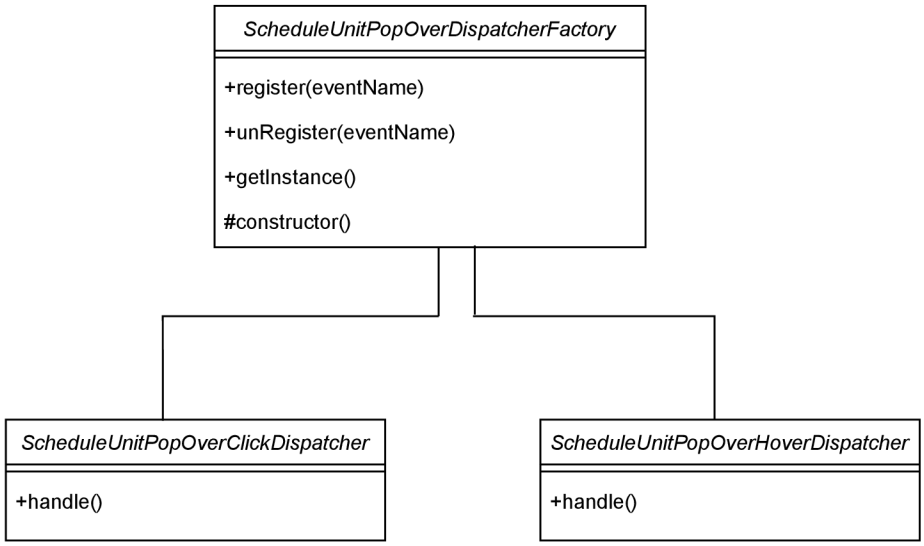
Obrázek B.4: Diagram tříd znázorňující navrženou architekturu React komponent



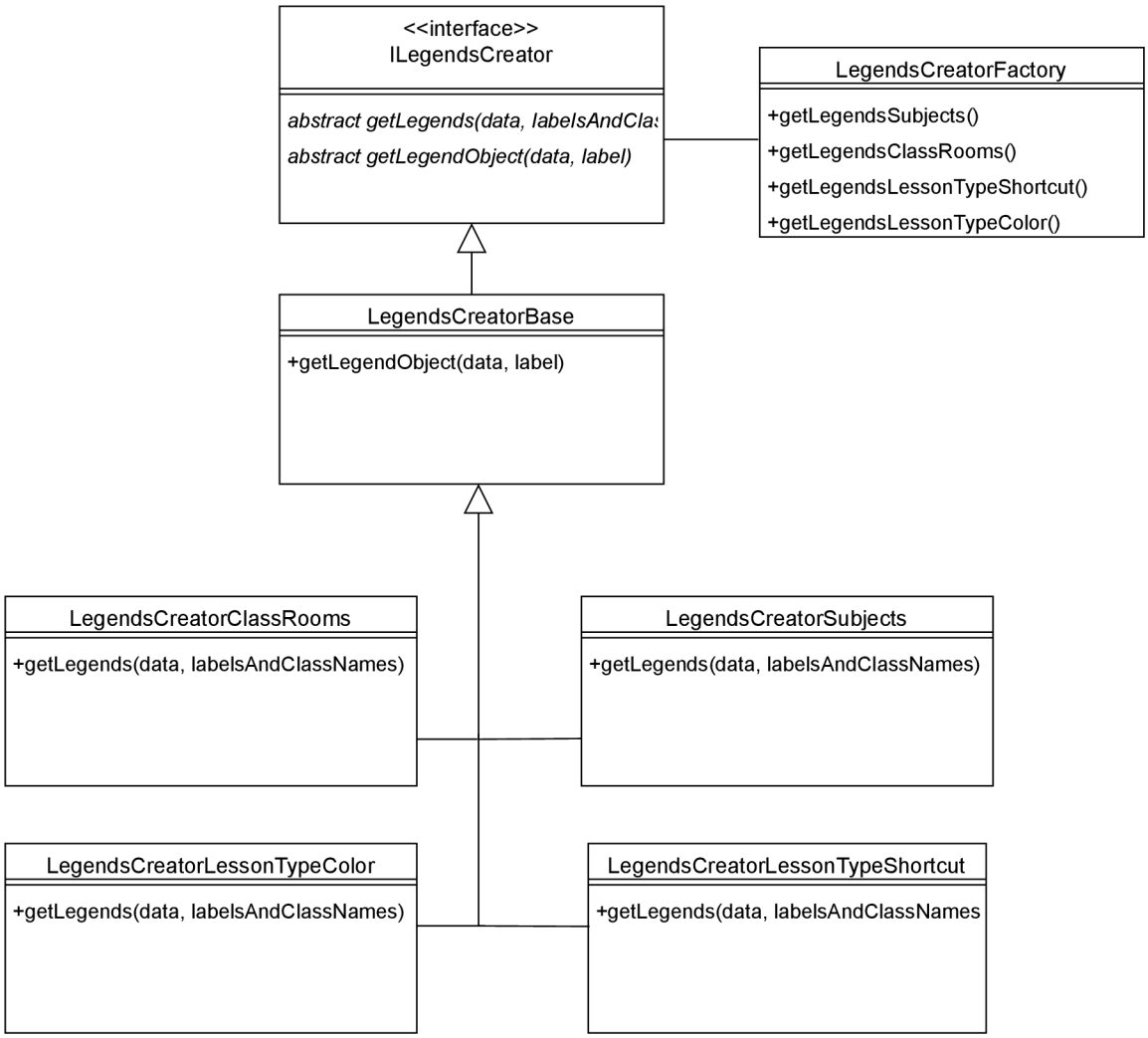
Obrázek B.5: Diagram tříd znázorňující navrženou architekturu filtrování



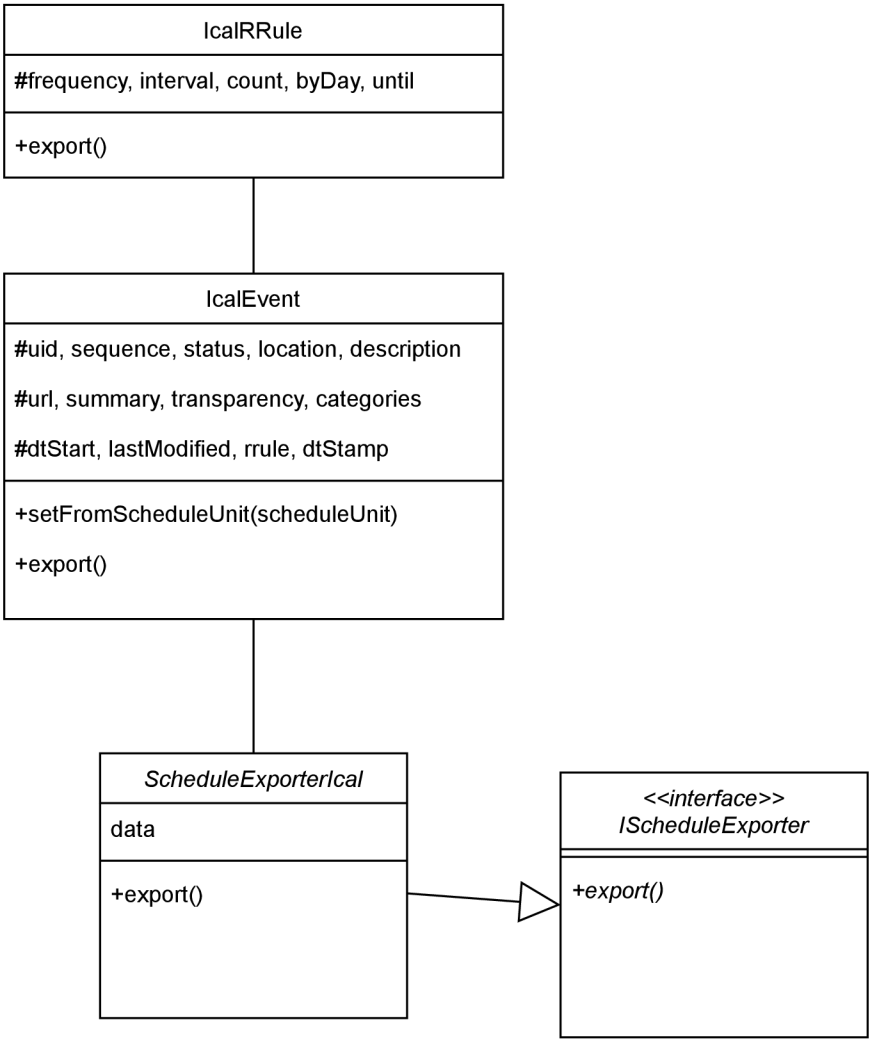
Obrázek B.6: Diagram tříd znázorňující navrženou architekturu zpracování událostí



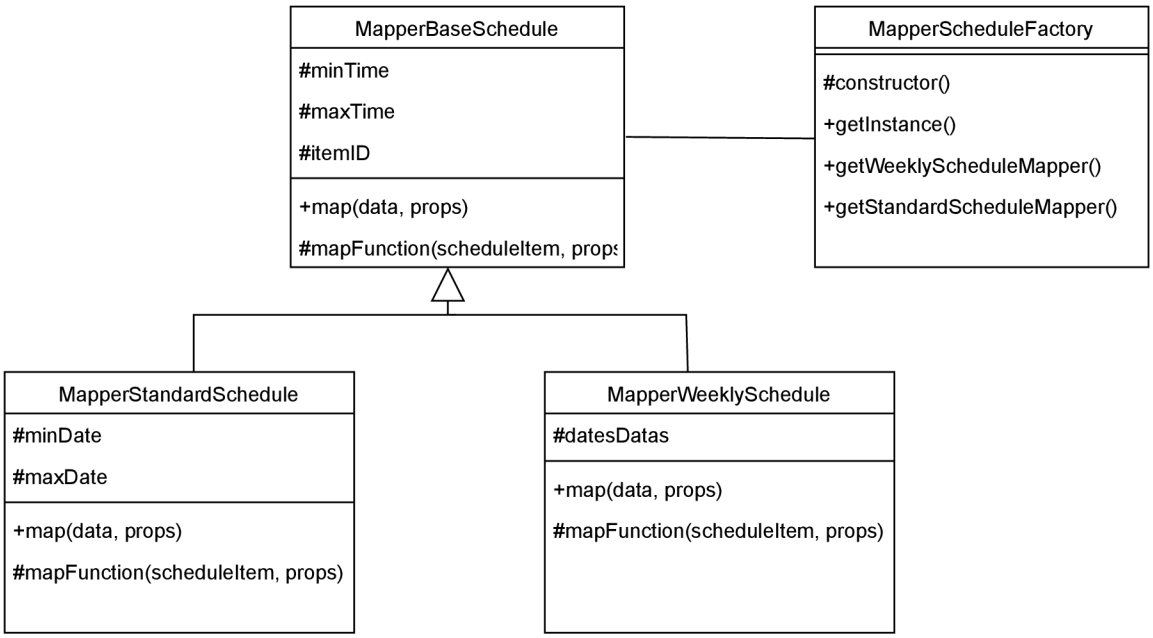
Obrázek B.7: Diagram tříd znázorňující navrženou architekturu pro změnu typu zobrazení podrobnějších informací o rozvrhové jednotce



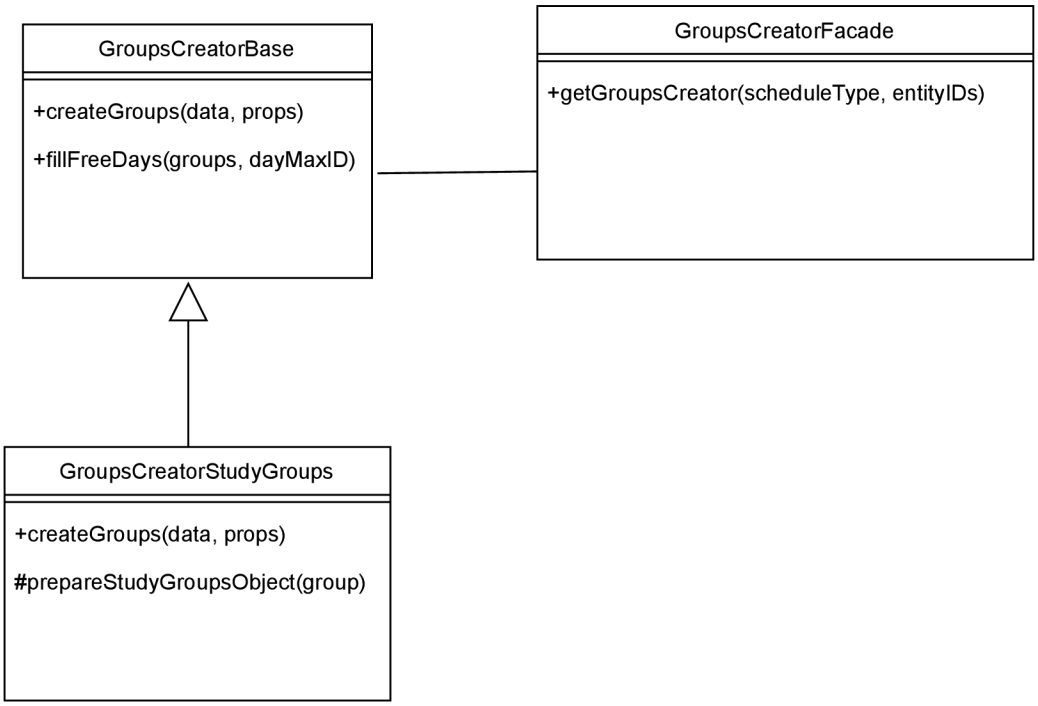
Obrázek B.8: Diagram tříd znázorňující navrženou architekturu zobrazování legend k rozvrhům



Obrázek B.9: Diagram tříd znázorňující navrženou architekturu pro filtraci do formátu iCal



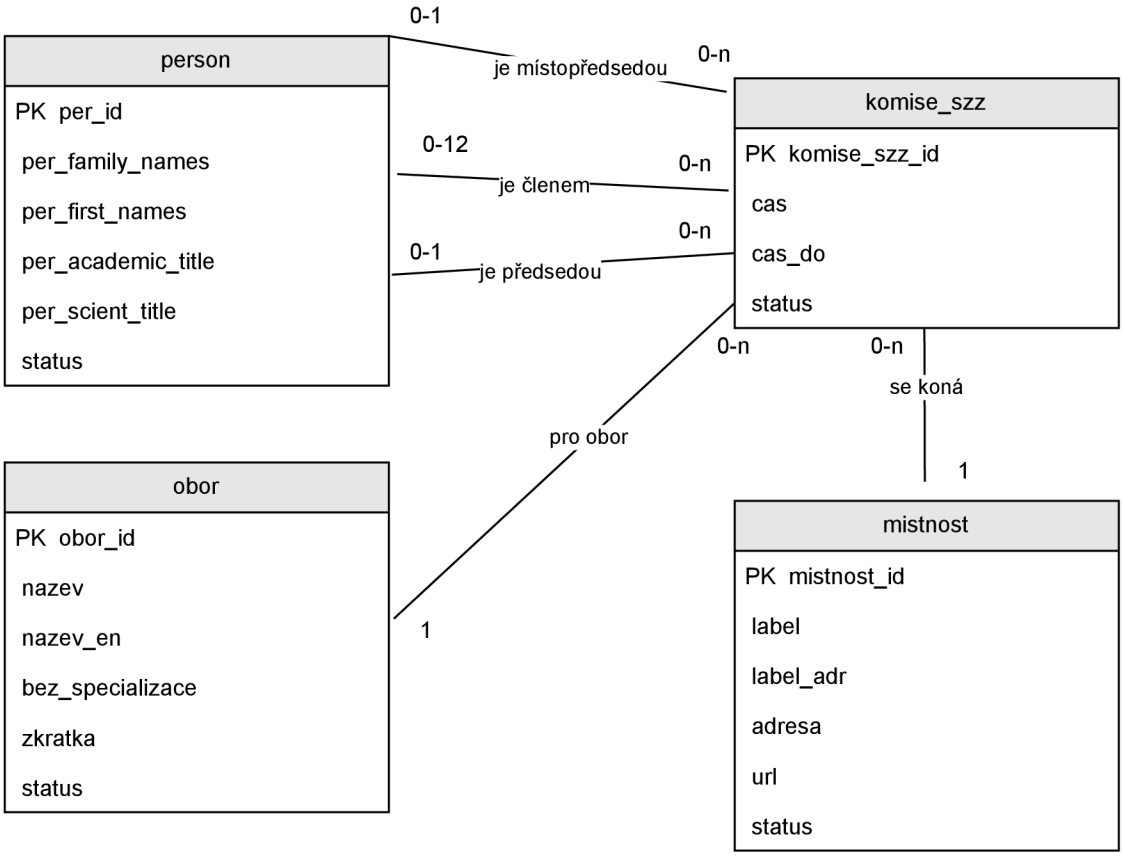
Obrázek B.10: Diagram tříd znázorňující navrženou architekturu mapování dat pro vykreslení rozvrhů



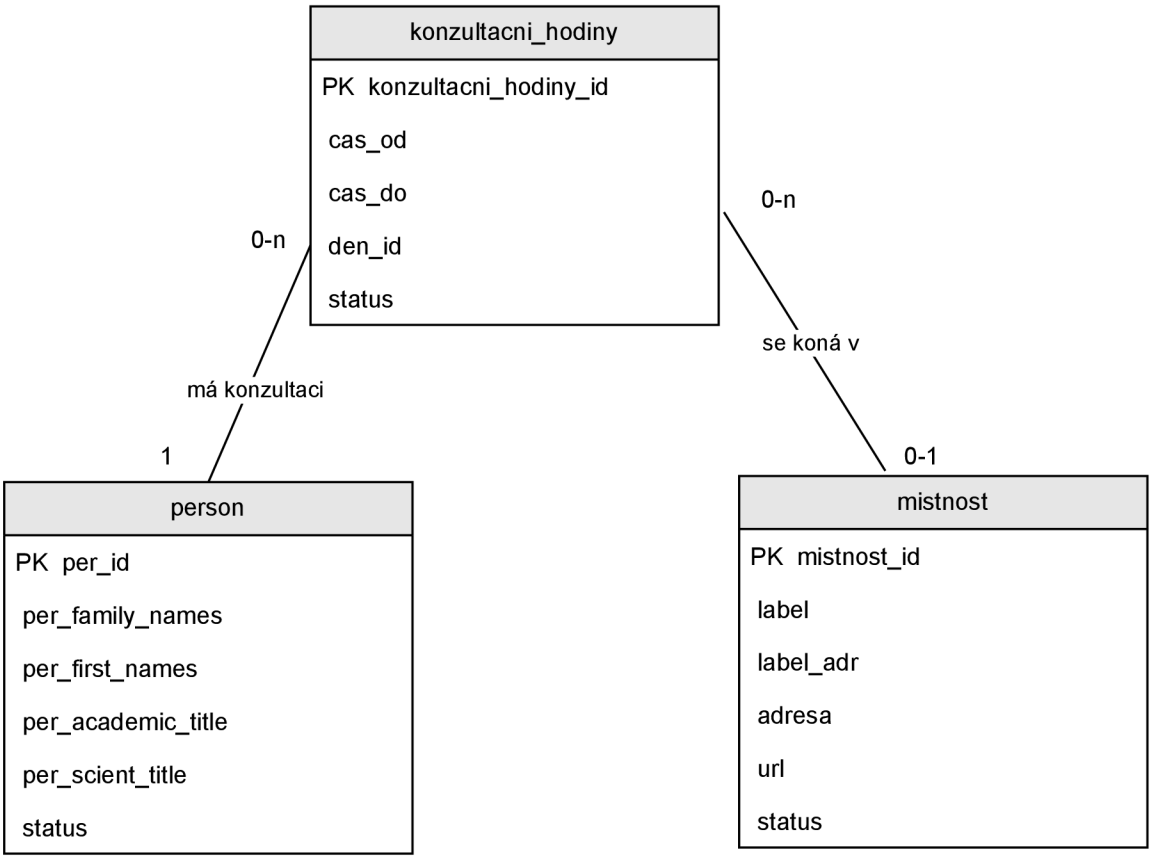
Obrázek B.11: Diagram tříd znázorňující navrženou architekturu pro tvorbu konceptu skupin

Příloha C

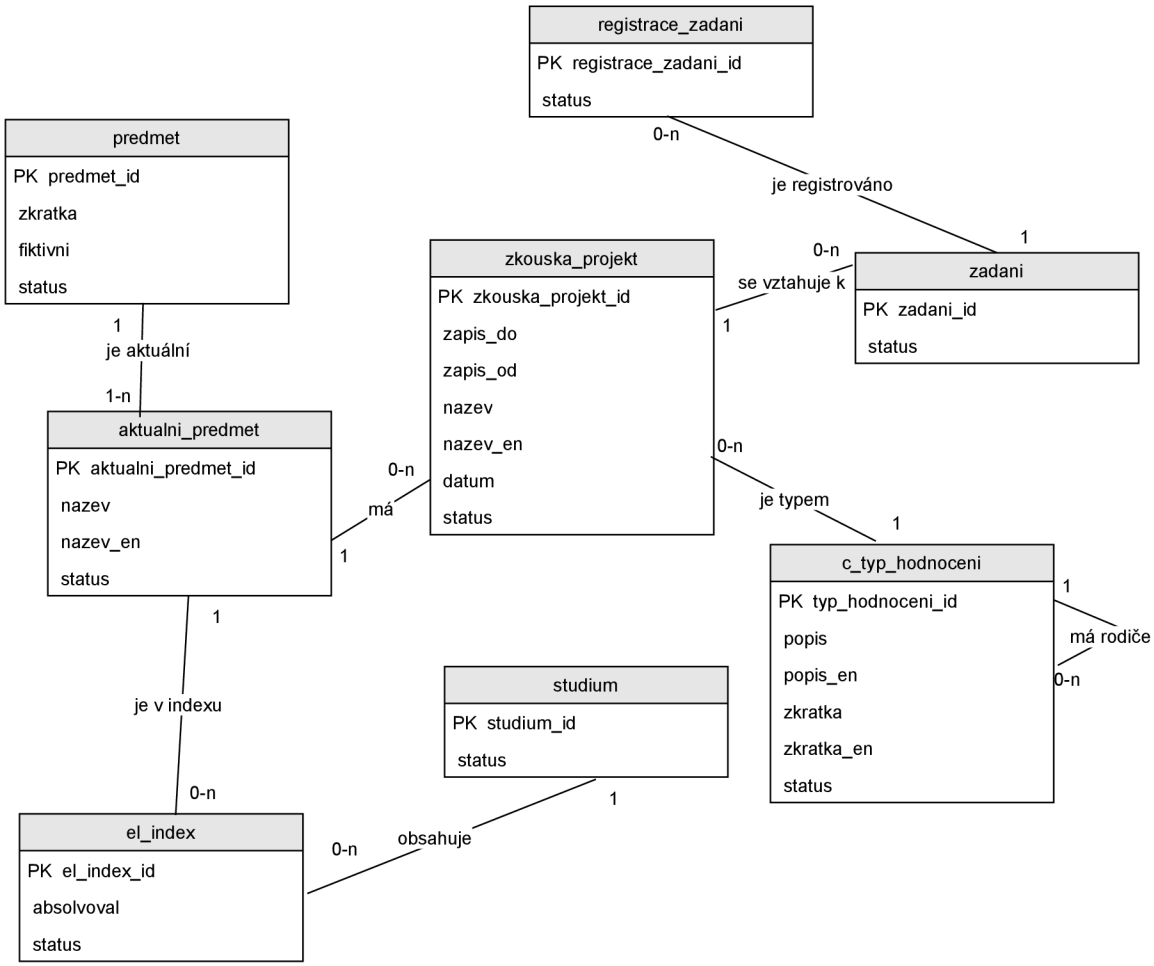
ER diagramy



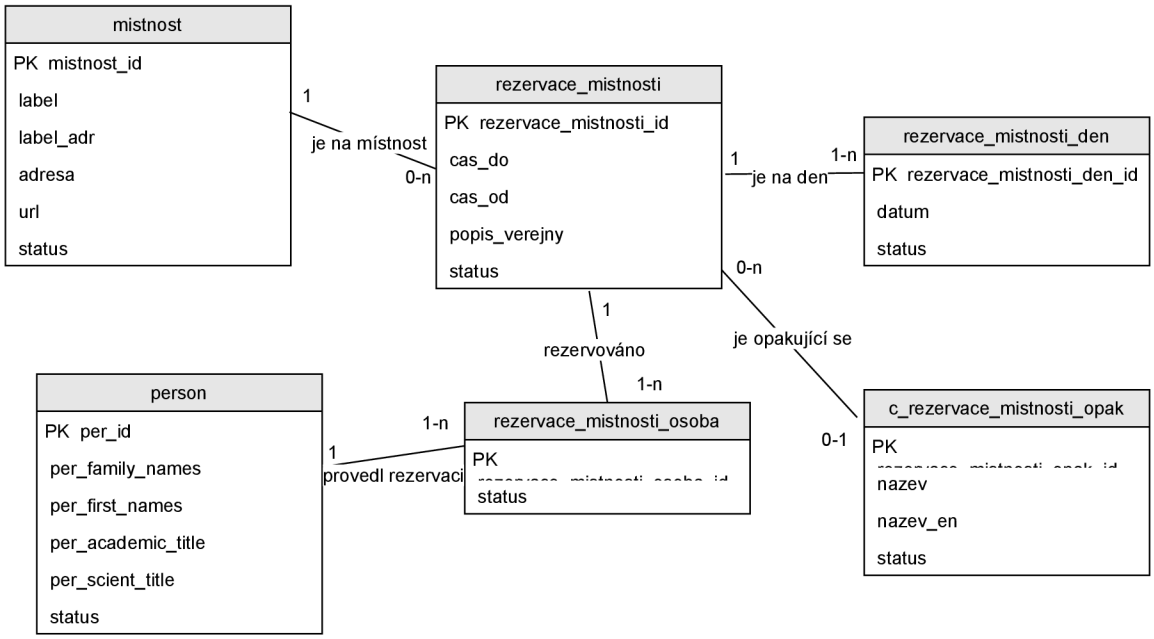
Obrázek C.1: ER diagram popisují agendu konání komisí státních závěrečných zkoušek



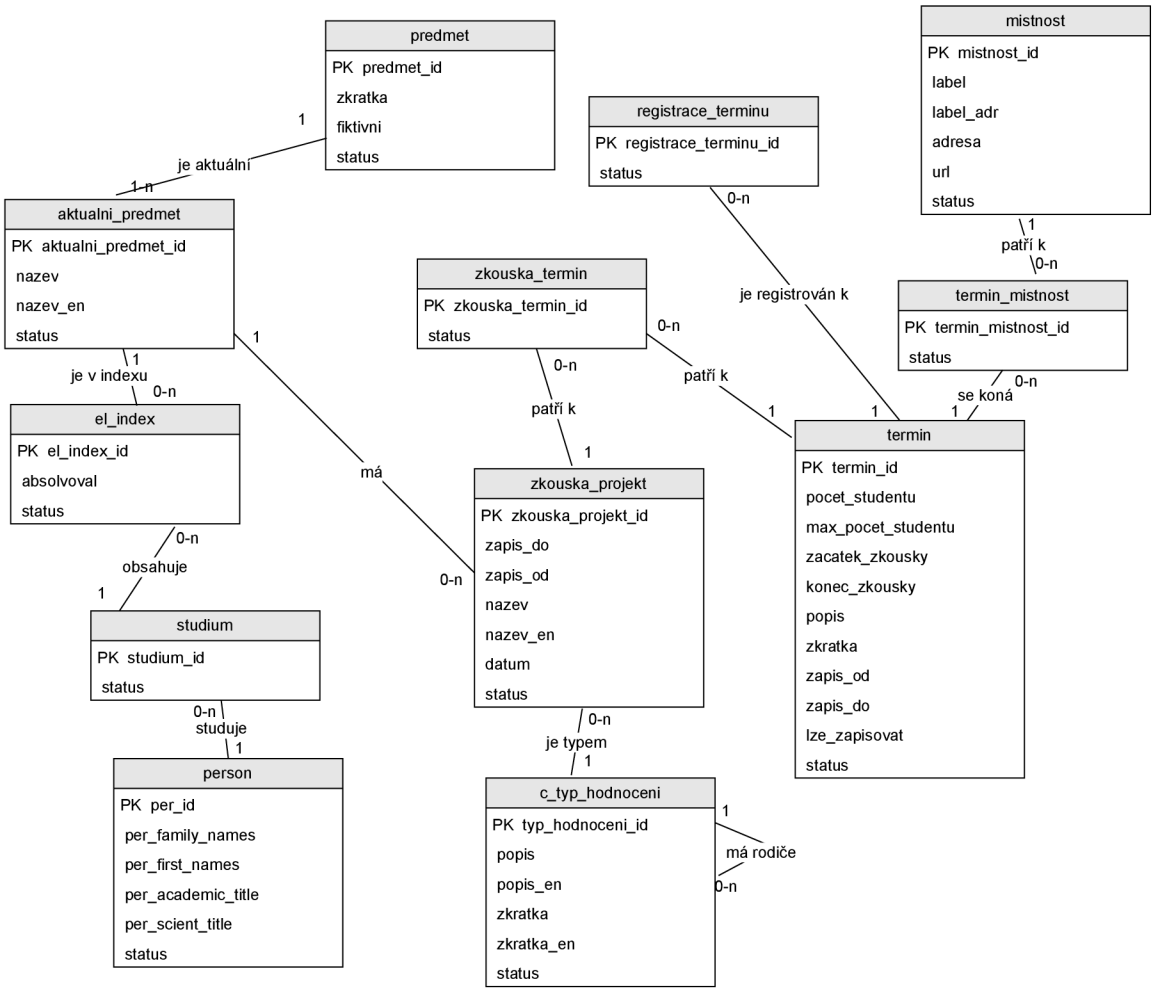
Obrázek C.2: ER diagram popisují konání konzultačních hodin vyučujících



Obrázek C.3: ER diagram popisující agendu zadání a registrace studentů na tato zadání



Obrázek C.4: ER diagram popisující problematiku rezervací místností



Obrázek C.5: ER diagram popisující problematiku termínů zkoušek

Příloha D

Použité algoritmy

Generování řídké matice

```
getGridMatrixRows(row, actualGroup):
    Nastav rows = []
    1) Vlož do row informace o actualGroup
    2) Pokud skupina actualGroup obsahuje podskupiny, pak:
            Pro všechny podskupiny, mající index i zavolej rekurzivně
                getGridMatrixRows(row, actualGroup.groups[i]) a
            výsledek této rekurze rozbal do proměnné rows
        Jinak:
            pro všechny položky v actualGroup.preparedData
            s indexem i prováděj:
                inicializuje pole r = []
                Vlož všechny prvky z row do pole r a poté
                vlož všechny prvky z actualGroup.preparedData[i]
                do proměnné r a vlož proměnnou r do pole rows
            vrať proměnnou rows jako výsledek volání funkce


prepareGroupData(group):
    nastav preparedData = []
    vlož do prepared data []
    1) Pro všechna data v group.data d vykonávej:
        1a) Zjisti interval i, ve který se rozvrhová jednotka
            reprezentovaná skrze d vyučuje
        1b) Zjisti, zda rozvrhová jednotka s intervalem i
            lze vložit do preparedData na indexy i:
                pokud ano, pak:
                    Vlož d do preparedData na pozice intervalu i
                Jinak:
                    nastav row = []
                    Vlož d do row na pozice intervalu i
                    vlož do preparedData row
    2) nastav group.data = preparedData
```

```
0) Inicializuj výstupní pole na []
1) Seřaď data uvnitř všech skupin s podle počátečního času,
   kdy se daná rozvrhová jednotka koná a poté uprav data
   skupiny s voláním prepareGroupData(s)
3) Připrav si první řádek výstupní matice jako:
   Nastav proměnnou row = []
   Nastav jako aktuální pole skupin vstupní pole skupin
   3a) Nastav do proměnné group prvek na indexu 0 z
   aktuálního pole skupin
   3b) Pokud proměnná group není rovna null, tak:
          Vlož do proměnné row proměnnou group a proveď krok 3a) s
          polem skupin uvnitř skupiny group
       Jinak:
          Vlož do proměnné row všechny objekty, reprezentující
          časy výuky v hlavičce rozvrhu (8:00, 9:00, ...)
   3c) Vlož do výstupního pole proměnnou row
4) Pro všechny skupiny na vstupu g prováděj následující:
   Zavolej funkci getGridMatrixRows([], g[0]) a
   návratovou hodnotu této funkce rozbal do výstupního pole
```

D.1 Přepis řídké matice na řetězec

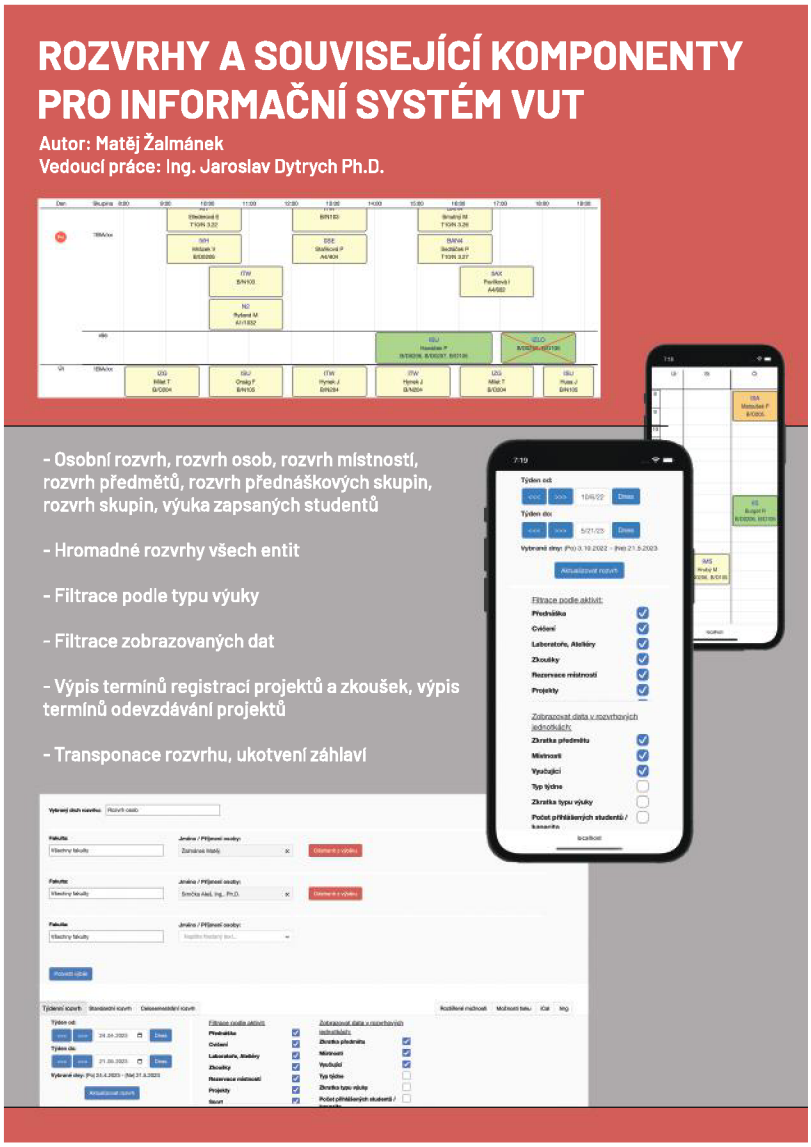
```
0) Označ si vstupní řídkou matici jako proměnnou m
1) Inicializuj výstupní řetězec res = ""
2) Pro všechny řádky r v m prováděj:
   Pokud r.length > 0, prováděj:
      3) Do res přidej řetězec "'"
      4) Nastav proměnné matrixIndex = 0, renderedCount = 0,
         renderedCountTimes = 0
      5) Dokud renderedCount + rendererCountTimes < m[0].length:
         6) Pokud r[matrixIndex] je null, potom proveď:
            7) do res přiděj řetězec ". " a zvyš renderedCount o 1
         6b) Jinak:
            8) pokud r[matrixIndex].data.timeStart != null, pak:
               9) Načti timeStart a timeEnd z r[matrixIndex].data
               10) Dokud renderedCountTimes < timeStart, prováděj:
                  11) do res přidej řetězec ". "
                  12) zvyš renderedCountTimes o 1
               13) Dokud renderedCountTimes < timeEnd, prováděj:
                  14) Do res přidej řetězec označující
                     název třídy buňky z proměnné
                     r[matrixIndex].className + " "
                  15) zvyš renderedCountTimes o 1
            8b) Jinak:
               16) do res přidej řetězec označující název
```

```
                    třídy buňky z proměnné
                    r[matrixIndex].className + " "
                17) Zvyš renderedCount o 1
            18) Zvyš matrixIndex o 1
        19) vlož do res řetězec "’ " + r[0].height

20) Nakonec vlož do res řetězec " / " spojený s
    hodnotou width ze všech sloupců v prvním řádku m
```

Příloha E

Plakát



Obrázek E.1: Plakát reprezentující výsledky práce