VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO EVIDENCI VÝSLEDKŮ STUDIA

INFORMATION SYSTEM FOR TRACKING STUDY RESULTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ JÍNEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19356/2016/xjinek00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Jínek Tomáš**
Obor: Informační technologie
Téma: **Informační systém pro evidenci výsledků studia**
Information System for Tracking Study Results
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s moderními technologiemi tvorby informačních systémů s webovým uživatelským rozhraním.
2. Prostudujte požadavky na systém pro sledování výsledků doktorandů na Ústavu informačních systémů.
3. Po konzultacích s vedoucím navrhněte architekturu daného systému.
4. Implementujte navržený systém pomocí vhodných technologií.
5. Proveďte testování systému.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015
- Dokumentace k projektu Nette: http://doc.nette.org/cs/2.2/

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

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

Vedoucí: **Burget Radek, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Hlavním cílem této bakalářské práce je navrhnout a implementovat informační systém pro správu a evidenci studijních výsledků doktorandů. Tento systém je uzpůsoben pro snadnou správu studentů a učitelů, z nichž je poté vytvořen rozvrh prezentací každého studenta a jejich výkon následně i oznámkován. Při jeho vývoji byl použit PHP framework Nette a CSS framework Bootstrap. Systém nahrazuje současný způsob tvorby rozvrhu a evidenci známek, který byl řešen Excel dokumentem.

Abstract

The main goal of this bachelor thesis is design and implementation of information system for Tracking study results of doctoral candidates. This system is made for easy student and teacher management, from which the schedule for student presentations is made and their performance graded afterwards. During the development, the PHP Nette framework and the CSS Bootstrap framework were used. System is replacing current way of schedule creating and grade records, which solution was Excel document.

Klíčová slova

Informační systém, HTML, CSS, PHP, JQuery, Nette, Bootstrap, Sass, AJAX

Keywords

Information system, HTML, CSS, PHP, JQuery, Nette, Bootstrap, Sass, AJAX

Citace

JÍNEK, Tomáš. *Informační systém pro evidenci výsledků studia*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Burget Radek.

Informační systém pro evidenci výsledků studia

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Radka Burgeta, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Jínek
17. května 2017

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Radkovi Burgetovi, Ph.D za poskytnutí odborné pomoci.

Obsah

Kapitola 1

Úvod

Cílem této bakalářské práce je vytvořit informační systém, ve kterém bude možné pohodlně a jednoduše spravovat studijní průběh doktorandů na Ústavu informačních technologií. Do této chvíle bylo využíváno pouze tabulek programu Microsoft Excel, které byly nepřehledné, obtížně se s nimi pracovalo a jejich sdílení mezi souvisejícími lidmi bylo nepříjemné. Z tohoto důvodu vznikla potřeba pro informační systém, který by zvládal veškerou správu s minimálním vynaloženým úsilím od člověka.

Tento informační systém pomáhá s vytvořením rozvrhu prezentací pro doktorandy a jejich školitele. Bylo nutné všechny informace přehledně evidovat, aby byly připraveny pro jejich další použití v následujících letech. Rozvrh musí být jednoduše spravovatelný a přenositelný do dalších let, být lehce dostupný pro všechny, kterých se rozvrh týká a umožnit rychlé a přehledné známkování studentů při jejich prezentování. A těmito problémy se bude tahle práce zabývat.

V následujících kapitolách jsou popsány využité technologie a nástroje, analýza požadavků pro systém, ze které následně vzejde návrh systému a jeho implementace, poté jsou vysvětleny způsoby testování a závěrem proběhne zhodnocení dosažených výsledků a možné budoucí rozšíření.

Kapitola 2

Použité technologie

Tato kapitola se bude zabývat technologiemi a nástroji použitými při vývoji. Dále pojednává o tom, které další frameworky by se daly použít při vývoji informačního systému a jejich stručný popis.

2.1 HTML

HTML (*HyperText Markup Language*) je standardní značkovací jazyk používaný pro tvorbu webových stránek. Tento jazyk se skládá z množiny značek (anglicky tag), které představují stavební kameny každé webové stránky. Prohlížeč značky neukazuje, ale používá je pro strukturování a vykreslení webové stránky. Tyto značky se uzavírají do ostrých závorek a každá z nich může navíc obsahovat parametry, které dále upravují její chování a vlastnosti. [19]

2.2 CSS

Pro specifikaci vzhledu HTML dokumentů se používá jazyk CSS (*Cascading Style Sheets*). S jeho pomocí lze nastavit dokumentu mnohem komplikovanější vzhled, než by dokázalo samotné HTML. Značky jazyka HTML specifikujeme tím, že jim nastavíme třídu nebo ID a pomocí toho se poté můžeme v CSS odkazovat na konkrétní značky a dodávat jim vzhled, např. měnit barvu písma, nastavit barvu pozadí, velikost atd. [18]

2.2.1 Sass

Sass (*Syntactially awesome stylesheets*) je rozšíření pro jazyk CSS, které jeho autoři nazývají „CSS se super silami“. K funkcím klasického CSS dodává možnost tvorby proměnných, podmínky, mixiny, funkce aj. Při vývoji je použita syntaxe SCSS (*Sassy CSS*). Nejdůležitějším prvkem této syntaxe, je možnost zanořování jednotlivých elementů a díky tomu se stává kód přehlednější a značně urychluje čas potřebný k jeho napsání. [8]

2.2.2 FontAwesome

Další rozšíření pro jazyk CSS se nazývá FontAwesome a vyvinul jej Dave Gangdy pod MIT licencí. Přináší do jazyka CSS možnost přidávání plně upravitelných ikonek. V tuto chvíli už má k dispozici 675 ikonek (ze kterých můžeme vidět pár ukázek na obrázku 2.1), kterým lze libovolně upravit velikost, barvu, stíny a vše, čeho lze dosáhnout s jazykem CSS.

Obrázek 2.1: Malá ukázka ikonek, kterých lze s pomocí FontAwesome dosáhnout

2.3 MySQL

Databáze obecně je oddělená aplikace, která uschovává kolekci dat. Každá databáze má jedno nebo více unikátních API pro tvorbu, přístup, správu, vyhledávání a klonování dat, která uschovávají.

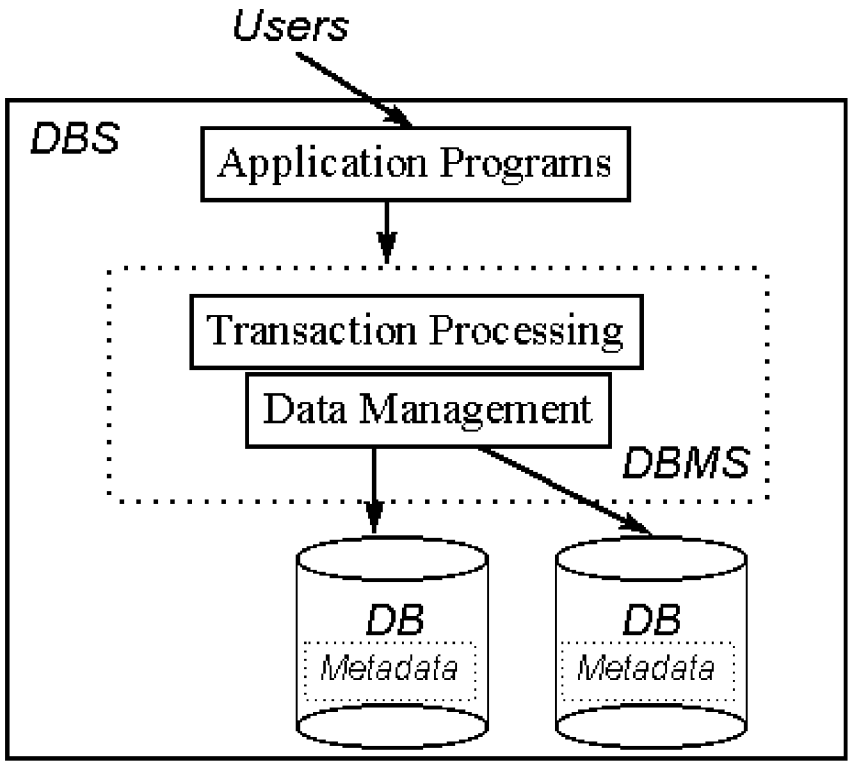
Softwarový systém, který dovoluje uživateli spravovat databázi a poskytuje kontrolovaný přístup k ní, se nazývá DMBS (Database Management Systems). Spolu s databází a aplikacemi, se kterými uživatel manipuluje, tvoří tato trojice DBS (Database System). Vizuální zpracování tohoto systému lze vidět na obrázku 2.2.

Oproti jiným způsobům uschovávání dat (soubory, hashovací tabulky aj.) je čtení a zápis dat v databázových systémech rychlejší a snažší. V dnešní době se používají relační DBMS (Relational Database Management Systems) k uschovávání velkého množství dat. Relační se nazývají, protože veškerá uschovaná data se třídí do různých tabulek, mezi kterými vznikají relace.

MySQL [17] je rychlá, snadno použitelná relační DBMS, vyvíjena pod open-source[1] licencí, takže jej může každý používat bez placení. Vyvinula jej a podporuje švédská společnost MySQL AB. MySQL využívá spousta malých i velkých společností, mezi které patří i YouTube, PayPal, Google, Facebook a spousta dalších. Takto populární se stal díky mnoha dobrým důvodům:

- Jedná se o velmi mocný nástroj, zvládá velké množství funkcí nejdražších a nejsilnějších databázových balíčků.
- Využívá standardní formu známého jazyka SQL.

[1] Otevřený zdrojový kód, každý jej může libovolně modifikovat.



Obrázek 2.2: Vizuální zpracování databázového systému (převzato z http://holowczak.com)

- Pracuje na mnoha operačních systémech a s velkým množstvím programovacích jazyků včetně PHP, Java, C, atd.
- Podporuje velké databáze, které dosahují až 50 miliónů záznamů.
- Snadná správa databáze díky různým nástrojům jako např. phpMyAdmin.

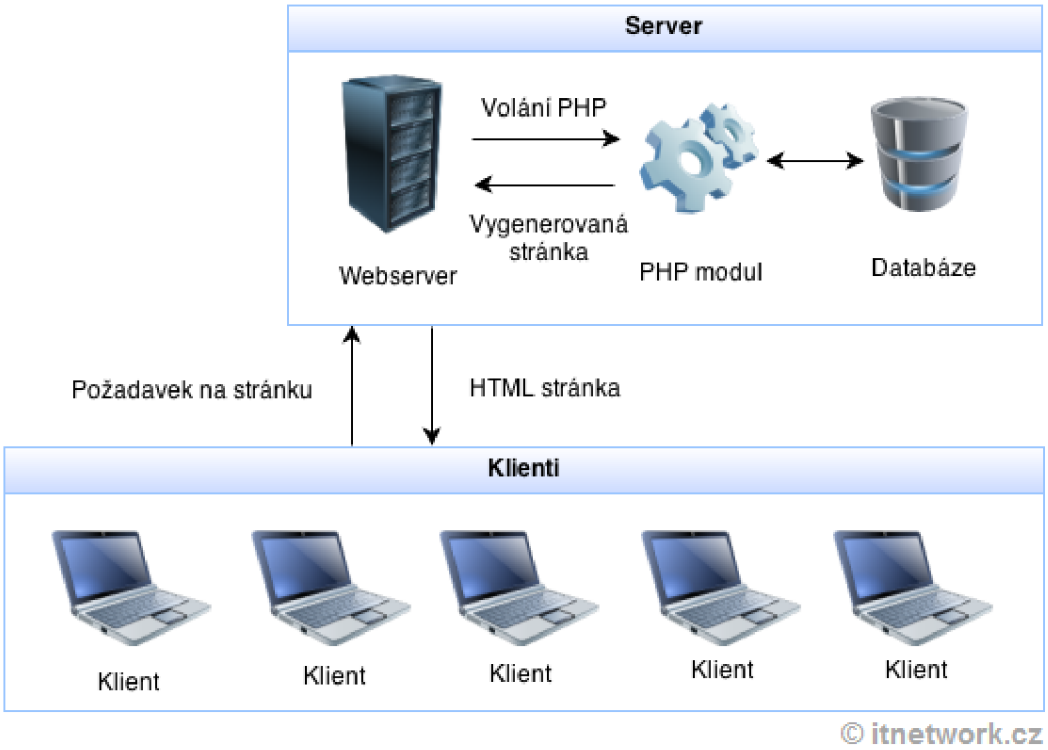
2.4 PHP

Dnešní PHP (*Hypertext Preprocessor*) je jeden z nejoblíbenějších jazyků pro vývoj webových aplikací. Tento jazyk se již natolik vyvinul, aby dovolil programátorovi rychle vyvíjet dobře strukturované a bezchybné programy používáním jak procedurální[2], tak objektově orientované[3] techniky programování. Zprostředkovává schopnost používat mnoho již existujících knihoven hotového kódu, které jsou dodány v základní instalaci nebo mohou být doinstalovány. Tohle dodává možnost splnit určitý úkol několika různými způsoby. Díky tomu se nám dostává více flexibility, než u jiných programovacích jazyků. Jednoduchost přidávání nových knihoven hotového kódu do tohoto prostředí je jedním z mnoha faktorů, díky kterému se dnes těší takové popularitě. [15]

[2]Procedurální technika – obsahuje funkce, které se volají z hlavního proudu programu a poté se vrátí zpět k hlavnímu proudu.

[3]Objektově orientovaná technika – používá třídy a objekty. Třída popisuje co může objekt obsahovat, včetně proměnných a metod.

Na obrázku 2.3 je možné vidět, jak probíhá komunikace klient-server v prostředí PHP. Skripty se provádí na straně serveru, na základě požadavku od klienta, tam se zpracují, načtou se potřebná data z databáze a vygeneruje se požadovaná stránka, která se poté vykreslí na obrazovku klienta.



Obrázek 2.3: Architektura webových aplikací v PHP [14]

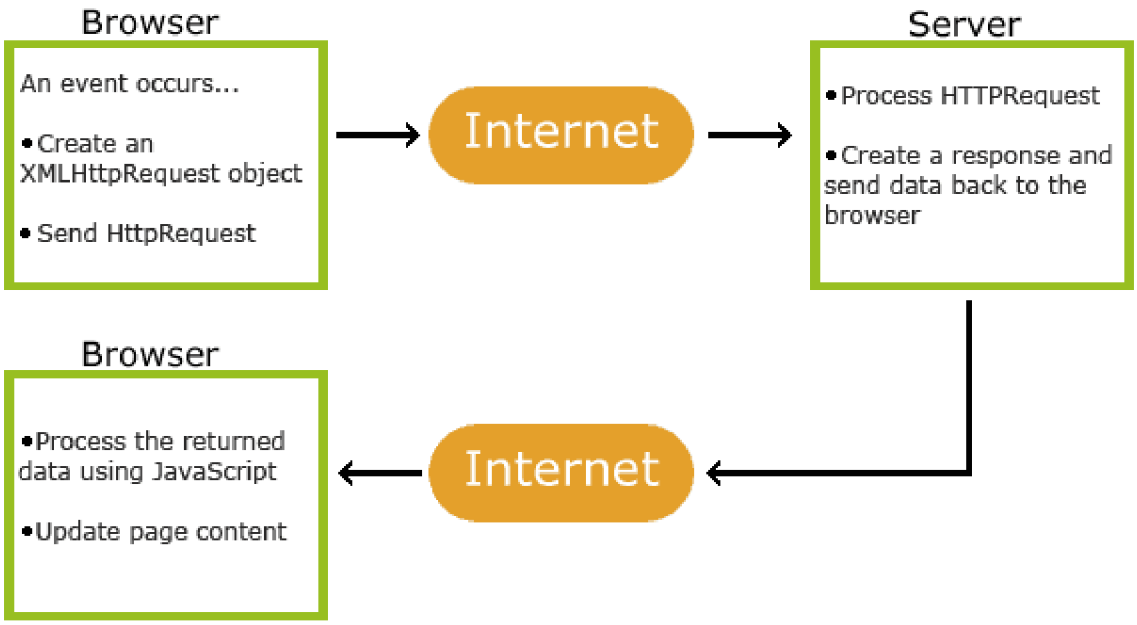
2.5 jQuery

JQuery je malá a rychlá JavaScriptová knihovna s jednoduchou syntaxí. Umožňuje snadné cestování po DOM (Document Object Model) struktuře dokumentu, zpracování událostí (kliknutí na objekt, změnu v políčku pro psaní apod.), animace, změny CSS vlastností u elementů, zasílání Ajaxových žádostí o data apod. Také se sama stará o veškeré problémy s kompatibilitou mezi prohlížeči. Různé elementy, se kterými chceme pracovat, můžeme snadno vybrat pomocí tzv. CSS selectorů, což jsou stejné selectory, jaké se používají v CSS.

Tuto knihovnu používá také Bootstrap Framework, který je součástí této práce a je o něm pojednáno v kapitole 2.11.

2.6 AJAX

V minulosti byly webové aplikace omezeny, protože pokaždé, když byla potřebná nová data, bylo nutné stránku znovu načítat. Poté začaly být čím dál víc přístupné technologie, které umožňovaly asynchronně získávat XML data přes JavaScript. Tyto technologie jsou známé pod zkratkou AJAX (Asynchronous Javascript and XML applications).



Obrázek 2.4: Jak AJAX funguje (převzato z http://w3schools.com)

Na obrázku 2.4 vidíme, jak probíhá proces získávání a zobrazení dat přes AJAX. Jakmile uživatel na stránce vyvolá určitou událost (např. odeslání formuláře, kliknutí na tlačítko atd.), tak JavaScript vytvoří objekt XMLHttpRequest, naplní ho požadovanými informacemi a odešle žádost serveru. Server žádost přijme, sežene požadovaná data a zašle žádost zpět webové stránce, kde si ji přečte JavaScript a provede žádanou akci (např. hlášku o úspěšném zpracování formuláře, výpis dat atd.). [11]

2.7 Composer

Composer je nástroj, použitý pro snadnou a automatickou instalaci a aktualizaci balíčků a závislostí v PHP. Nejedná se o správce balíčků v klasickém slova smyslu. Jedná s balíčky a knihovnami, ale v rámci jednoho projektu a neinstaluje nic globálně, tudíž je to tzv. Dependency Manager.

Práce s ním probíhá přes příkazovou řádku, do které stačí pouze uvést příkaz ke stáhnutí požadovaného balíčku nebo rozšíření, který je vždy uveden na stránce balíčku či rozšíření a stačí jej tedy pouze zkopírovat a nechat nainstalovat. V tomto projektu byl Composer použit ke stáhnutí všech pluginů i Nette samotného [4].

2.8 Bower

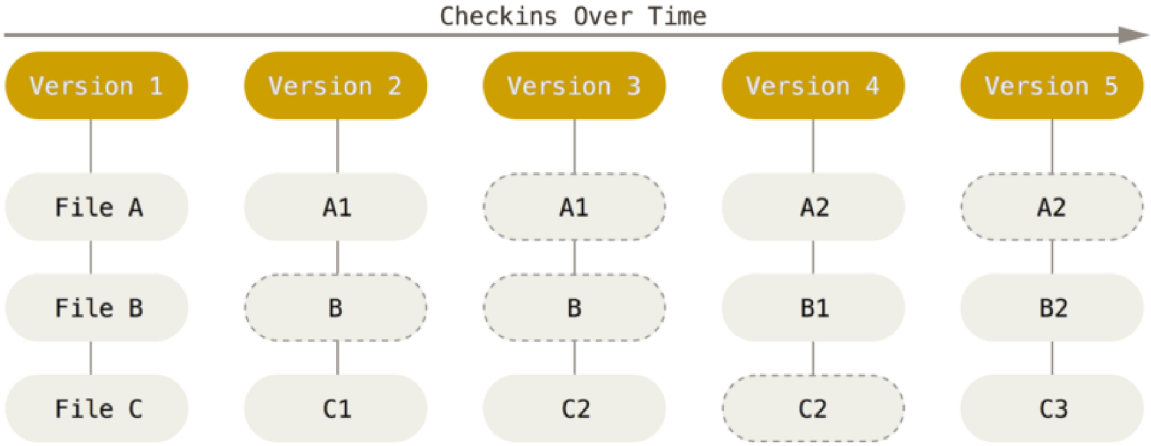
Bower je správce front-end balíčků. Používá se na jednoduché a automatické stahování balíčků, jako jsou jquery nebo bootstrap [2].

2.9 Git

Git je distribuovaný systém pro správu verzí. Byl navržen při vývoji Linuxového jádra, z důvodu rozpadu firmy, která provozovala jejich aktuálně používaný verzovací systém BitKeeper. To inspirovalo stvořitele Linuxu, Linuse Torvaldse, k tvorbě vlastního verzovacího systému založeného na zkušenostech, které získali při používání BitKeeperu.

Pokud pracujete na větším projektu s týmem lidí nebo potřebujete pouze pravidelně zálohovat svoji práci, vyhledání nějakého verzovacího systému je inteligentní věc co udělat. Umožňuje zrcadlit Vaši pracovní složku a zálohovat ji na internetu, vracet změny v souborech do předešlého stavu, sledovat změny v souborech, včetně toho, kdo změny provedl a další. V případě nějakých potíží proto není problém zpětně dohledat, kdy a kde se stala chyba, vrátit celý projekt do stavu kdy byl funkční nebo v případě ztráty lokálních dat stáhnout všechny soubory projektu zpátky do stavu, v jakém byly při posledním nahrání.

Narozdíl od jiných verzovacích systémů, které ukládají data jako seznam změn v souborech, si Git při každém nahrání vytvoří nový miniaturní souborový systém. Aby byl takový způsob ukládání efektivní, tak soubory, které nebyly změněny se znovu nenahrávají, ale pouze se uloží odkaz na předešlý identický soubor (obrázek 2.5) [9].



Obrázek 2.5: Ukládání dat v systému Git jako řadu fotek souborového systému (převzato z https://git-scm.com)

2.10 Nette

Jedná se o PHP framework, který byl vyvinut Davidem Grudlem. Nyní je vyvíjen společností Nette Foundation. Je šířen jako svobodný software pod licencí New BSD nebo GNU General Public License (GPL), což znamená, že neklade téměř žádná omezení na to, co se může s frameworkem dělat a kdokoliv si ho může měnit a používat, aniž by musel někoho žádat o povolení nebo platit. S jeho pomocí byla vyvinuta řada známých českých webových stránek jako např. ČSFD.cz, Bandzone.cz, Uložto, EDNA.cz, Slevomat.cz aj.

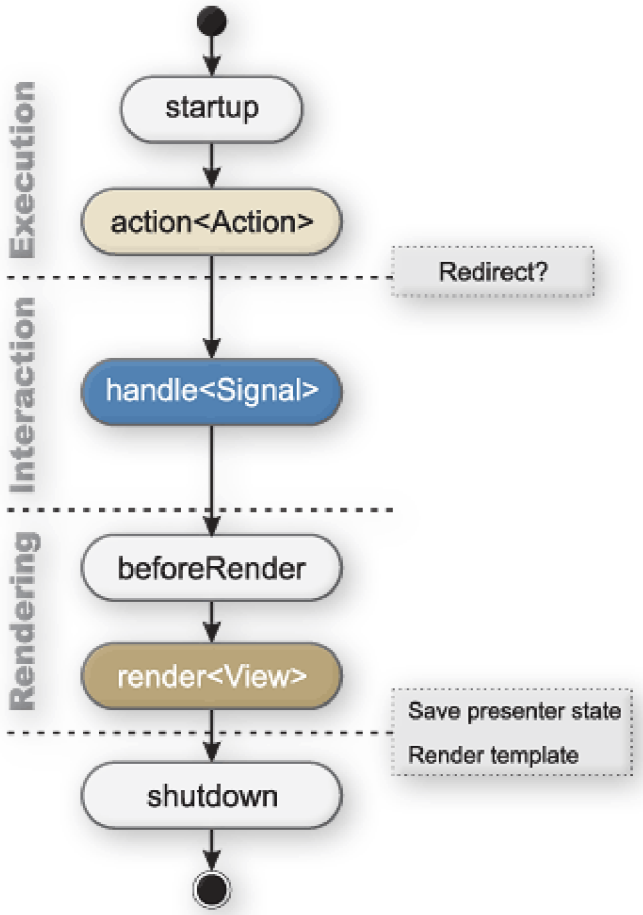
Nette Framework je český projekt a díky tomu se může chlubit nejaktivnější komunitě v České Republice, která si vzájemně radí, pomáhá a vytváří vlastní doplňky a rozšíření. Dokonce se umístil na 3. místě v anketě „Best PHP Framework for 2015“ magazínu Sitepoint [16]. Je postaven na architektuře MVP (Model-View-Presenter), založené na klasickém programovacím modelu MVC (Model-View-Controller) o které si povíme více v kapitole 4.2.

Další vlastnosti, které jsou pro Nette typické jsou [6]:

- Dokonalá bezpečnost díky revoluční technologii, která eliminuje výskyt bezpečnostních děr.
- Ladící nástroj Tracy.
- Excelentní výkon – jedná se o jeden z nejrychlejších frameworků [10].
- Neustále se rozšiřující nabídka pluginů a doplňků.
- Promyšlený a čistý objektový návrh, který využívá PHP verze 7.1.
- Vlastní šablonovací systém Latte.

2.10.1 Životní cyklus presenteru

Každý presenter obsahuje sadu metod, které se volají v určitém pořadí při každém načtení stránky. Můžeme proto provádět akce ještě předtím, než se stránka vykreslí. Příkladem může být kontrola přihlášení uživatele a jeho práv, dříve než se vykreslí stránka, ke které je zapotřebí určitá úroveň práv uživatele. Posloupnost těchto metod je vidět na obrázku 2.6.



Obrázek 2.6: Životní cyklus presenteru

startup

První metoda, volaná okamžitě po vytvoření presenteru. Zde se provádí zmíněné ověřování oprávnění nebo se inicializují proměnné.

action<Action>

Jedná se o obdobu metody `render<View>`. Provádí se zde akce, které nepotřebují vykreslení stránky, např. přihlášení uživatele nebo vkládání/čtení dat do/z databáze a následné přesměrování. Tato metoda je volána dříve, než proběhne vykreslení stránky, je zde proto možné změnit, která metoda se použije pro vykreslení stránky.

handle<Signal>

Tato metoda zpracovává signály. Určena zejména pro komponenty a AJAXové požadavky.

beforeRender

Volá se jako poslední před vykreslením šablony. Obvykle obsahuje například nastavení šablony, předání proměnných, které jsou společné pro více View apod.

render<View>

Zde se vkládají potřebná data do šablony. Ve výchozím stavu se automaticky použije šablona s názvem stejným jako View v názvu této metody. Pokud není uveden konkrétní pohled, volá se metoda `renderDefault()` a použije se šablona `default.latte`.

shutdown

Tato metoda se volá při ukončení životního cyklu presenteru.

2.10.2 Latte

Latte je šablonovací systém pro PHP, který šetří práci a zabezpečí výstup. Vkládání PHP úseků mezi HTML elementy je nepěkné a nepraktické, proto poskytuje Latte množství maker a filtrů, které výrazně zpříjemní práci v šabloně a udělá ji přehlednější.

V Latte existují 2 druhy maker. První jsou makra, která se vkládají do složených závorek a lze díky nim urychlit výpis proměnných, cykly apod. Kupříkladu, v normální PHP šabloně musí programátor zapisovat cyklus jako:

```
<ul>
<?php foreach ($items as $item): ?>
  <li><?php
    echo $item ?>
  </li>
<?php endforeach ?>
</ul>
```

V Latte je takový kód zjednodušen:

```
<ul>
{foreach $items as $item}
  <li>{$item}</li>
{/foreach}
</ul>
```

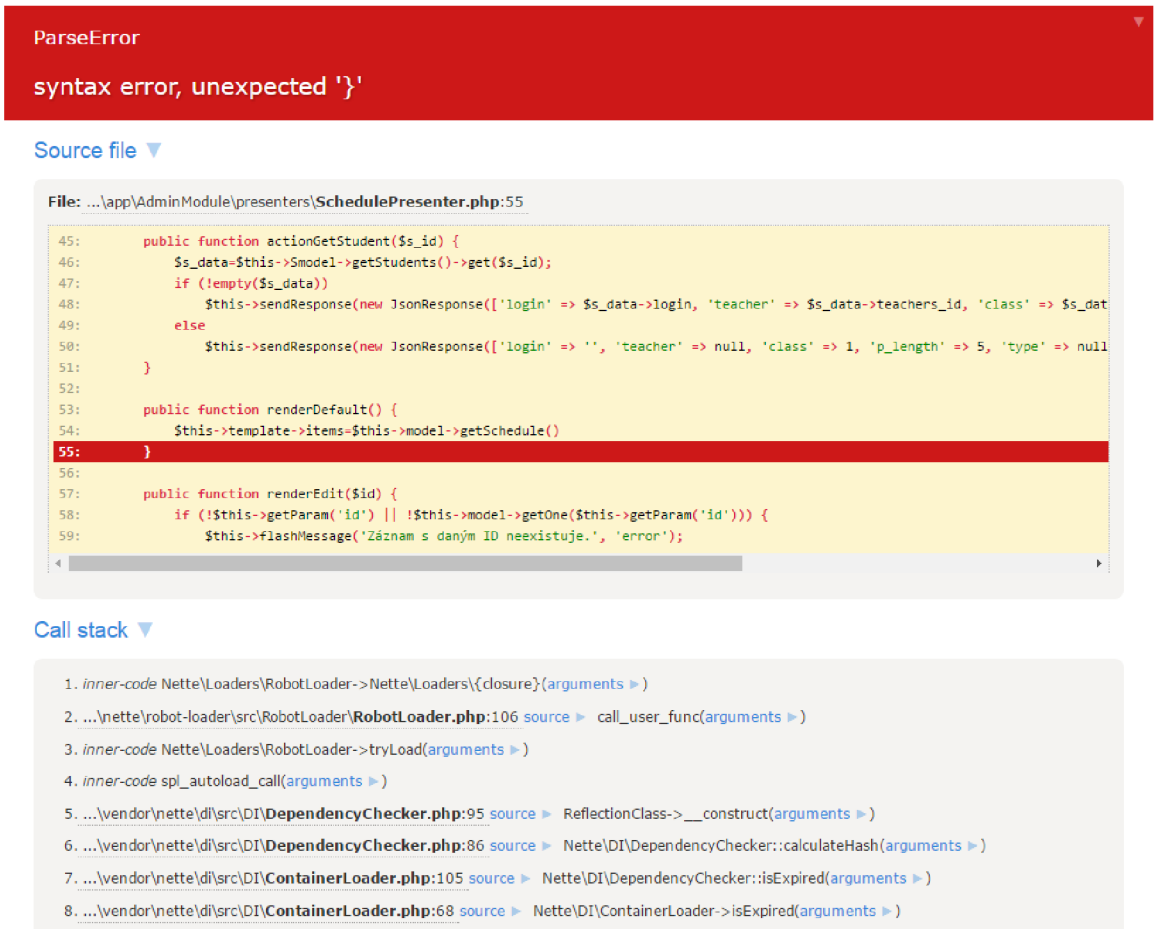
Druhý druh maker jsou tzv. n:makra. Jsou to speciální makra, která se zapisují dovnitř HTML elementu. Můžeme tedy výše uvedené `foreach` makro dále zjednodušit pouze na:

```
<li n:foreach="$items as $item">{$item}</li>
```

Dále lze k vypsání proměnných připojit filtr. Díky filtrům je možné snadno upravit podobu, v jaké bude proměnná vypsána. Kdyby bylo potřeba vypsat proměnnou pouze velkými písmeny, nemusí být volána žádná PHP funkce, stačí za proměnnou připojit filtr `capitalize` pomocí svislé čáry, aby výsledek vypadal následovně: `{$item|capitalize}`.

2.10.3 Tracy

Tracy je debugovací knihovna Nette, která pomáhá rychle odhalit a opravit chyby, zaznamenávat je, přehledně vypisovat proměnné a měřit čas operací. Vypsání chyby je značně přehlednější a konkrétnější než chyba, kterou by vypsal interpret jazyka PHP (obrázek 2.7).



Obrázek 2.7: Diagnostický nástroj Tracy

Dalším užitečným nástrojem, který Tracy poskytuje je Debugger Bar, což je plovoucí panel ve spodním rohu stránky. Poskytuje nám informace o skriptu, jako je čas a náročnost provedení skriptu, systémové informace, provedené dotazy do databáze a jejich časová náročnost, aktuální routovací masku a seznam všech definovaných v pořadí podle priority, a informace o právě přihlášeném uživateli. Debugger bar lze dále rozšiřovat o další funkce. Příklad toho, jak může vypadat, lze vidět na obrázku 2.8.

Obrázek 2.8: Debugger Bar

2.11 Bootstrap

Bootstrap je HTML, CSS a JavaScriptový framework pro snadnou tvorbu vzhledu pro web, který se dynamicky mění dle aktuální velikosti obrazovky a používaného zařízení. Takovému vzhledu se říká responzivní. Stejně jako u Nette Frameworku se jedná o svobodný software, který může kdokoliv libovolně využívat a modifikovat. Obsahuje rozsáhlou sbírku HTML a CSS komponent a řadu jQuery pluginů, připravených pro okamžité použití. Na celém internetu jej využívají milióny úžasných webových stránek mezi které patří např. Spotify | THE DROP, NASA, Vogue aj. [1].

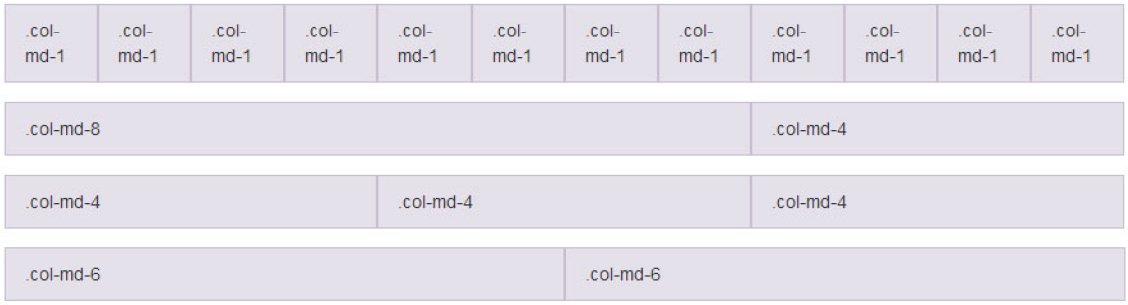
Nejvýznamnější komponentou Bootstrapu pro tvorbu responzivního vzhledu je jeho Grid (mřížkovací) systém. Jeho hlavní myšlenkou je rozložení obsahu stránky na sérii sloupců a řádků pomocí předdefinovaných tříd (obrázek 2.9). Každý řádek se rozděluje na 12 sloupců, uvnitř kterého rozdělíme každému elementu, s pomocí zmiňovaných tříd, počet sloupců, které bude element zabírat a s měnící se velikostí obrazovky zařízení je velikost každého sloupce změněna o stejnou hodnotu, čímž zůstává poměr mezi elementy na řádku totožný. Pro docílení úplné responzivity, nabývá třída sloupce dalšího atributu, který určuje, kolik bude na jakém zařízení zabírat sloupců. Každá třída sloupce má potom tvar `.col-xx-yy`, kde `yy` představuje počet sloupců s rozsahem 1-12 a `xx` představuje množinu hodnot:

- `xs` – pro extra malé zařízení, jako jsou mobilní telefony, s šířkou menší než 768px,
- `sm` – pro malé zařízení, jako jsou tablety, s šířkou větší nebo rovnou 768px,
- `md` – pro průměrně velké zařízení, jako jsou větší tablety nebo menší notebooky, s šířkou větší nebo rovnou 992px,
- `lg` – pro velké zařízení, jako jsou počítače, s šířkou větší nebo rovnou 1200px.

2.12 Další frameworky

2.12.1 Laravel

Stejně jako u Nette, jde o open-source MVC framework. Vytvořil ho Taylor Otwell. Nabízí robustní výběr nástrojů a architekturu, která se inspiruje z těch nejlepších vlastností frameworků jako jsou CodeIgniter, Ruby on Rails, Sinatra aj. V průzkumu o nejoblíbenější framework na webu SitePoint, který byl zmíněn už v kapitole 2.10, vyhrál 1. místo.



Obrázek 2.9: Vizuální zobrazení Bootstrapového grid systému [1]

Na oficiálních stránkách lze nalézt skvělou dokumentaci a stovky videonávodů, kterým zde říkají Laracasty, díky čemuž je snadné s tímto frameworkem začít pracovat. Laravel obsahuje svůj vlastní šablonový engine, nazývající se „Blade“ a vlastní lokální vývojové prostředí s názvem „Homestead“ [5].

2.12.2 CakePHP

CakePHP vyvinula firma Cake Software Foundations už před 12 lety a stále se jedná o jeden z nejoblíbenějších frameworků současnosti. Jedním z důvodů je, že vždy dokázal držet s dobou a s aktuálními technologiemi. Stejně jako Laravel, je CakePHP open-source projekt a používá MVC technologii. Pohání weby velkých společností jako jsou BMW, Hyundai nebo Express. Obsahuje mnoho zabudovaných funkcí pro zvýšení bezpečnosti webu, hodí se proto pro weby, které potřebují zvýšenou úroveň zabezpečení. S frameworkem přichází i sada MVC konvencí, která pomůže nasměrovat při vývoji aplikace [3].

2.12.3 Zend Framework 3

Zend Framework je kolekce profesionálních PHP balíčků s více jak 127 milióny instalací. S jeho pomocí bylo vyvinuto i několik slavnějších produktů, jmenovitě stránky firem McAfee, IBM nebo Magento.

Zend Framework má k dispozici vlastní IDE (Integrated Development Environment) Zend Studio, které obsahuje nástroje pro integraci s ním a k usnadnění práce, například generování kódu. Nehodí se však pro menší projekty z důvodu velkého množství konfiguračních voleb, je však vynikající pro velké, složitější projekty. Nynější verze Zend Framework 3 je 3x až 4x rychlejší než předchozí verze a obsahuje nové komponenty, jako např. konvertor XML do JSON nebo plnou kompatibilitu s PHP 7 [7].

2.12.4 AngularJS

AngularJS je MVC JavaScriptový framework, který byl původně vyvinut v Googlu testujícím inženýrem Miskem Heverym. Nyní má Google vyhrazený tým na rozvíjení a správu AngularJS. Je open-source a kompletně zdarma. Používá se pro jednostránkové aplikace.

Srdcem AngularJS je koncept, který se nazývá *two way data binding*, který váže HTML a CSS do stavu JavaScriptové proměnné a kdykoliv se proměnná změní, AngularJS aktualizuje všechny elementy, které jsou svázané s touto proměnnou. Kupříkladu tento kód:

```
<div ng-show="shouldShow">Hello</div>
```

Jestliže se proměnná `shouldShow` změní na `false`, tak se všechny elementy, které mají k sobě svázanou tuto proměnnou, automaticky skryjí.

Díky tomu eliminuje velkou část kódu který musíte napsat. Tvůrci na jejich oficiálních stránkách popisují AngularJS vtipnou frází, že Vám povoluje „psát méně kódu, jít dříve na pivo“ [12].

Kapitola 3

Analýza požadavků

Důležitým krokem ve vývoji jakéhokoliv projektu je analýza požadavků. Probíhá tu sběr požadavků zákazníka a identifikují se požadavky, které jsou nejasné nebo nekompletní a následně se tyto nejasnosti řeší. Součástí analýzy je také ukládání těchto požadavků do nějaké formy, zde byl použit tzv. diagram případů užití (use-case diagram).

3.1 Specifikace požadavků

Seznam požadavků byl vytvořen na základě konzultací s vedoucím, podle praxe na Ústavu informačních systémů.

- Systém bude umět vytvářet, upravovat a evidovat studenty a učitele,
- bude umět z těchto údajů vytvářet záznamy do rozvrhu,
- rozvrh bude dostupný na veřejné části webu s jednoduchým dohledáním požadovaných záznamů,
- rozvrh bude možno importovat do dalších let pro znovupoužití,
- vytvořeným studentům bude možno udělovat a evidovat známky v průběhu let,
- udělování známek bude pro usnadnění možné udělat přímo u rozvrhu.

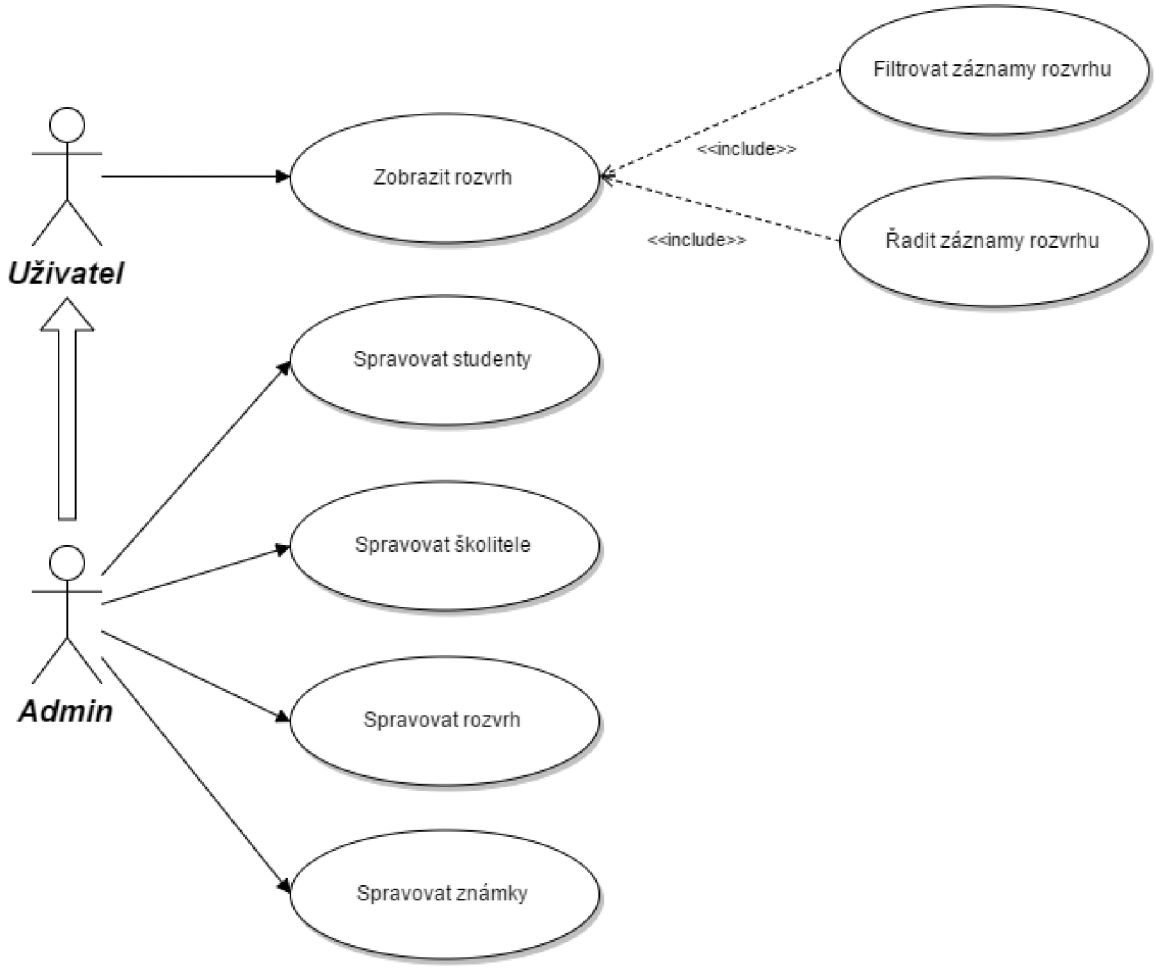
3.2 Diagram případů užití

Diagram případů užití je jeden z diagramů jazyka UML (Unified Modeling Language), který se stal standardem v oblasti návrhu a analýzy. Jde o nejjednodušší zobrazení vztahů uživatele se všemi funkcemi, které bude možný využívat. Nezobrazuje jak budou fungovat, pouze to, co má systém umět a jaké typy uživatelů ke kterým funkcím budou mít přístup. Je to proto první diagram, který vytváříme při návrhu systému.

Diagram případů užití se skládá z:

- Use Case (případ užití) – Akce, kterou může uživatel v systému vykonat. Obvykle zakreslován elipsou s názvem akce uvnitř.
- Actor (aktér) – Role, která má určenou množinu případů užití. Může jím být uživatel nebo i externí systém. Obvykle zakreslován jako panáček s názvem role pod ním.

- Relace – Vztahy mezi případy užití a aktéry. Znázorňuje, kteří aktéři mají přístupné jaké případy užití. Nebo v případě vztahu mezi aktéry navzájem jde o zobrazení dědičnosti. Vztahy se zobrazují čarou nebo šipkou.



Obrázek 3.1: Diagram případů použití

3.2.1 Uživatel

Uživatel zde představuje studenta nebo školitele, kteří nemají přístup do administrační části webu. Uživatel přejde do veřejné části, ve kterém je zobrazen rozvrh, vytvořený administrátorem. Zpočátku je vidět celý rozvrh, seřazen podle data od nejbližšího po nejvzdálenějšího. Každý sloupec rozvrhu lze řadit vzestupně i sestupně a u těch sloupců, u kterých je to logické (jméno studenta, jméno školitele apod.), je přítomný vstup pro filtrování hodnot z toho sloupce.

3.2.2 Admin

Systém bude obsahovat jediného administrátora, který má přístup k celé administraci. Bude moct upravovat, přidávat a mazat veškeré záznamy o školitelích, studentech, jejich

známkách a rozvrhu. Admin dědí případy užití uživatele, tudíž je schopen zobrazení rozvrhu a operací s ním.

Kapitola 4

Návrh aplikace

Tato kapitola pojednává o návrhu informačního systému. Je zde zobrazen a detailně popsán Entity-Relationship (ER) diagram (kapitola 4.1) a v kapitole 4.2 je vysvětlena architektura MVC, kterou tento systém používá.

4.1 ER diagram

Pro vizualizaci struktury databáze využijeme ER diagramu (obrázek 4.1). Pro jeho správné pochopení je důležité znát následující pojmy.

Entita

Představuje objekt z reálného světa, který chceme popisovat a z čeho se skládá (atributy). Ve fyzickém modelu entita představuje tabulku.

Instance

Skutečný objekt, který je popsaný entitou. Ve fyzickém modelu představuje řádek tabulky.

Atribut

Popisuje jednu vlastnost entity. Entita má obvykle více atributů. Ve fyzickém modelu představuje sloupec tabulky.

Identifikátor

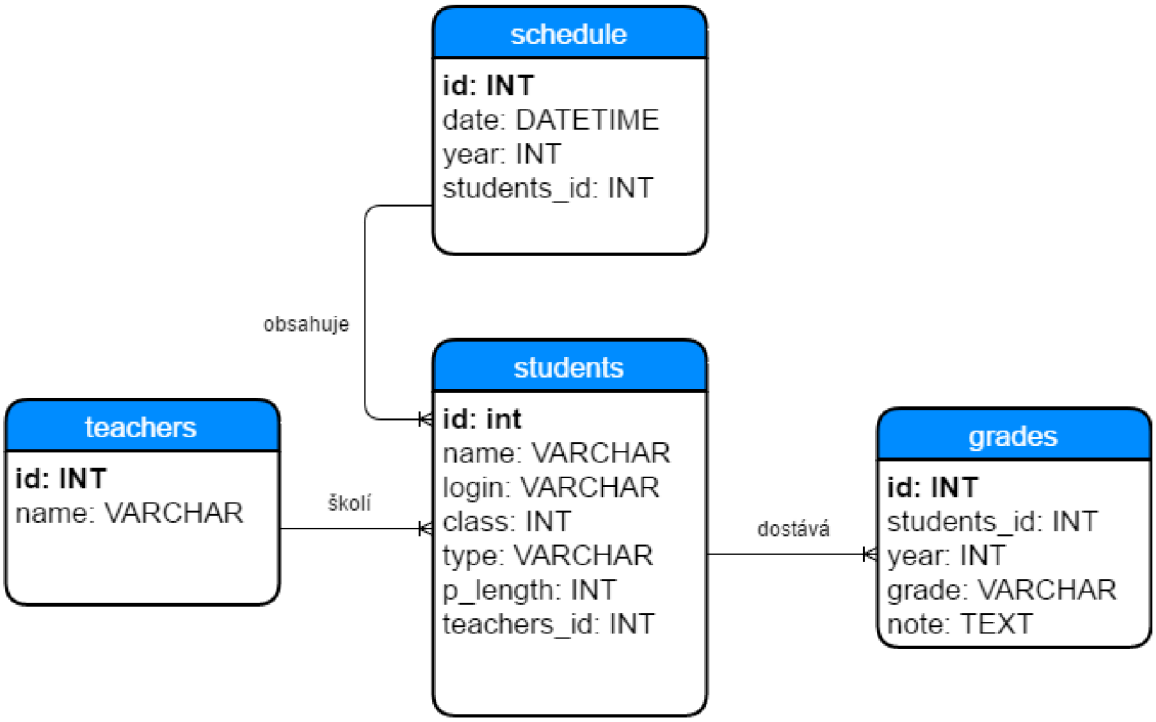
Identifikátor je druh atributu, který je vždy unikátní pro danou tabulku a jednoznačně identifikují každou instanci entity. V databázi se bude jednat o primární klíč.

Kardinalita

Kardinalita (nebo také vztah) určuje vazby mezi tabulkami, kolik výskytů jedné entity může vstoupit do vztahu s kolika výskyty druhé entity. Existují 3 typy kardinality:

- **1:1** – Jeden výskyt entity může být vázán pouze na jeden výskyt druhé entity, např. 1 manžel má 1 manželku a manželka má 1 manžela. Tento typ se příliš nepoužívá, protože vzniká otázka, zda není vhodnější tyto 2 entity spojit do jedné.

- **1:N** – Jeden výskyt entity může být vázán na více výskutů z druhé entity, ale ne naopak. Např. Skladatel napsal více písniček ale jednu písničku napsal vždy pouze 1 skladatel.
- **N:M** – Více výskytů z jedné entity může být ve vztahu s více výskyty z druhé entity. Např. Autor napsal mnoho knih a kniha má více autorů, kteří se na ní podíleli. U takového typu kardinality už nestačí pouze 2 entity. Využíváme proto spojovací tabulky, která neobsahuje žádný primární klíč, pouze určuje vazby mezi výskyty z první a druhé entity.



Obrázek 4.1: Entity-Relationship diagram

4.1.1 Entity

V této sekci budou popsány jednotlivé entity ER diagramu.

students

Tato entita představuje studenty. Ukládá jejich jména, unikátní loginy, ročník, délku prezentace a typ studia, který může být prezenční nebo kombinovaný. Každá instance studenta je vázaná na jednoho školitele, který má studenta na starosti.

grades

Entita známek ukládá informace o známkách studentů. Každá instance známky uchovává informace o studentovi, ke kterému se známka váže, roku, ve kterém byla známka udělena, známce samotné a případné poznámce k dané známce.

teachers

V této entitě se uchovávají školitelé, u kterých nepotřebujeme jiný údaj než jejich jméno.

schedule

Nakonec entita, ve které se uschovává celý aktuální rozvrh. Obsahuje informace o přesném času prezentace, o roku, ke kterému se rozvrh vztahuje a o studentovi, který v daný čas bude prezentovat.

4.2 Architektura systému

Architektura systémů vychází z návrhového vzoru MVC (Model-View-Controller). Nette nazývá svůj framework MVP (Model-View-Presenter), nicméně princip je stejný a Presenter má stejnou funkci, jakou má v MVC Controller.

MVC architektura dělí aplikaci na 3 logické části a každá z nich má specifickou úlohu:

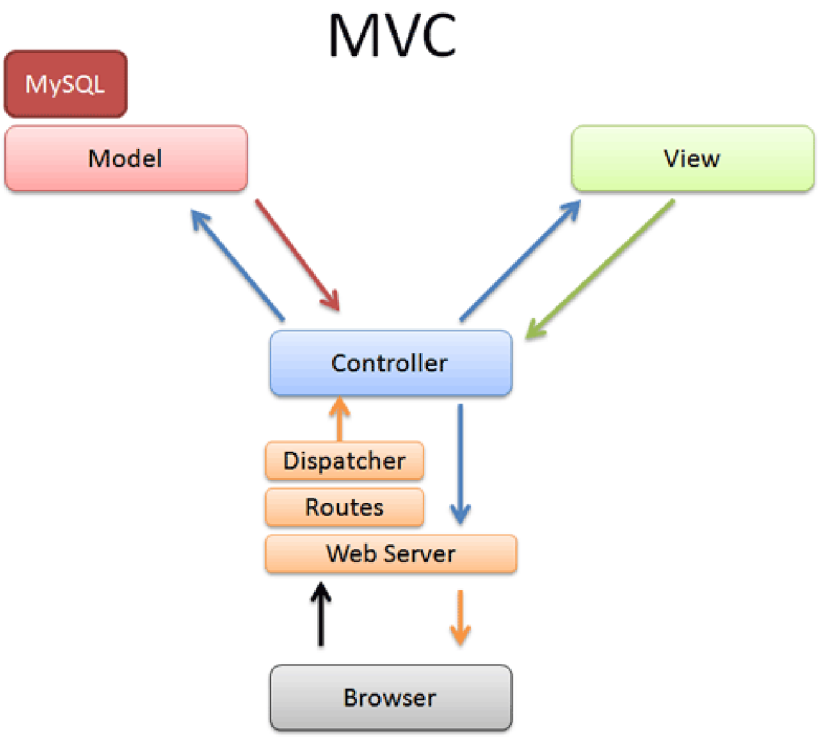
- **Model** – Stará se o logiku aplikace. Přistupuje k databázi, zprostředkovává data do řadiče a provádí výpočty. Většinou má každá datová entita vlastní model.
- **View** (pohled) – Převádí data zprostředkovaná modelem do uživatelsky přívětivé podoby. Obsahuje HTML kód. Pohled by měl omezit veškerou logiku pouze na iterace (for, while,...) a if, else.
- **Controller** (řadič) – Komponenta, se kterou uživatel komunikuje, řídí tok událostí aplikace. Je to takový prostředník mezi modelem a pohledem. Podle parametrů získá požadovaná data z modelu a předá je pohledu.

Model by neměl vědět, že existuje nějaký řadič nebo pohled. Pohled by také neměl vědět o modelu, pokud není zvolena koncepce, ve které si pohled přímo od modelu žádá data. Řadič je ten, který by měl seznámit model s pohledem [13].

Vizuální zpracování lze vidět na obrázku 4.2.

Životní cyklus stránky v Nette

1. Uživatel zadá do adresního řádku požadavek nebo klikne na stránce na nějaký odkaz, který může vypadat například následovně: `www.stranka.cz/clanek/nauc-se-mvc`.
2. Jako první se požadavek dostane do routeru, který adresu rozebere. V tomto případě router zjistí, že se má odkazovat na `ClanekPresenter` a předat mu jako parametr `nauc-se-mvc`.
3. Presenter zjistí, že má v parametrech nějakou hodnotu. V první řadě zkontroluje, jestli se první parametr shoduje s názvem nějaké akce v presenteru a když ho nenajde, zavolá výchozí akci. V této akci zavolá požadovaný model, že chce článek s tímto názvem.
4. Model se podívá do databáze a v případě, že článek existuje, zašle všechna potřebná data zpět presenteru.
5. Presenter data obdrží a předá pohledu (šabloně), což je v tomhle případě detail článku.
6. V šabloně si vytvoříme HTML strukturu a vložíme do ní obdržená data.
7. Uživateli se vykreslí HTML stránka.



Obrázek 4.2: Návrhový model MVC

Kapitola 5

Implementace

Tato kapitola se zaobírá konkrétní implementací informačního systému. Je zde, v kapitole 5.1, podrobně popsána adresářová struktura, kterou systém používá, vycházející ze struktury, kterou Nette definuje, známou jako sandbox. O zabezpečení systému a principu přihlašování do administrace je věnována kapitola 5.2. Více o práci s databázi v systému je pojednáno v kapitole 5.3. Nakonec, v kapitole 5.4, je podrobně vysvětlena funkčnost a možnosti celého implementovaného systému.

Jak vyplývá s předchozích textů, celý systém je postaven na frameworku Nette. Při vývoji byla snaha využívat co nejvíc z toho, co nám Nette, popřípadě jeho komunita, nabízí a celkově se o frameworku co nejvíce naučit. K vývoji bylo použito vývojové prostředí Sublime Text 3.

K práci s databází bylo využito relační DMBS MySQL.

5.1 Adresářová struktura

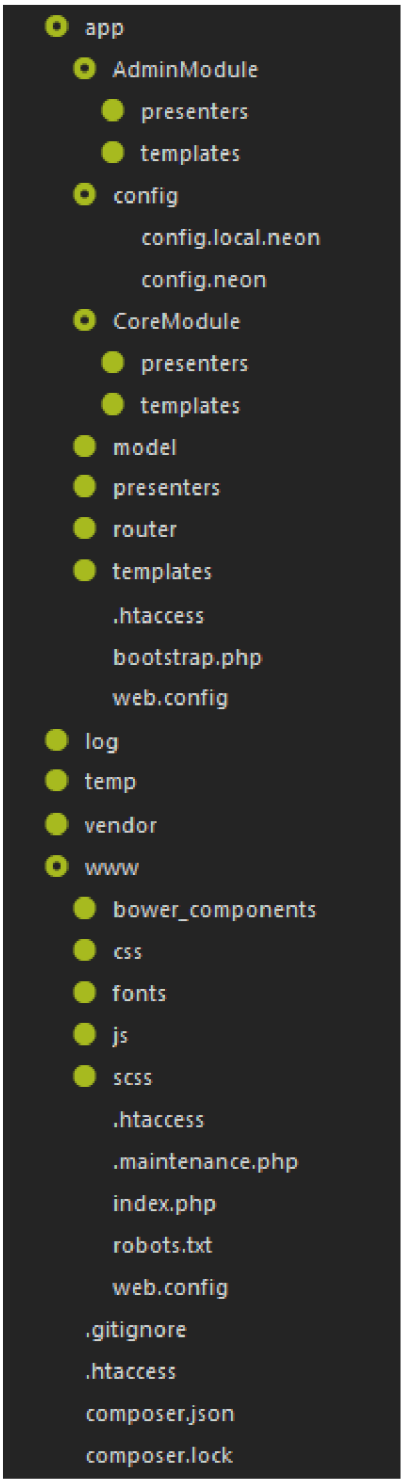
Adresářová struktura vychází z předpřipravené a předkonfigurované adresářové struktury Nette, známé jako sandbox, navíc rozšířenou o moduly. Je navržena, aby byla co nejpoužitelnější a nejpřátelštější, se zaměřením na bezpečnost a výkon. Adresářovou strukturu, použitou v tomto projektu, lze vidět na obrázku 5.1.

V kořenovém adresáři se nachází celkově 5 složek a 4 soubory. V první řadě si rozebereme funkci jednotlivých souborů:

- **.gitignore** – Soubor, určený pro verzovací systém Git (kapitola 2.9). Říká mu, které složky a soubory vyřadit při nahrávání souborů do online repozitáře.
- **.htaccess** – Soubor, který slouží pro úpravu vlastností serveru, aniž by musel žádat správce. Zde říká serveru, aby požadavek na zobrazení webu směřoval do složky `www/`, která obsahuje indexový soubor webu.
- **composer.json**, **composer.lock** – Konfigurační soubory pro nástroj Composer (kapitola 2.7).

app

Tato složka obsahuje zdrojové kódy projektu. Je kriticky důležité, aby se do této složky nedostala žádná třetí strana, např. přes browser. Tomu zabraňuje obsah souboru `.htaccess`, který úplně zabrání jakémukoliv zobrazení této složky z prohlížeče.



Obrázek 5.1: Adresářová struktura Nette

Pro lepší logické členění různých částí webu, jsem se rozhodl využívat modulů. Zde máme 2 moduly: `AdminModule` pro administraci a `CoreModule` pro veřejnou část webu. Každý z modulů obsahuje svoje vlastní presentery a šablony, které využívají. Každý model může být využíván ve více modulech, proto je složka s modely umístěna v kořenové části složky `app`.

Podobně jako s modely, složka `presenters` a `templates` v kořenové části obsahuje presentery a šablony, které využívá více modulů.

Globální konfigurace projektu je uložena ve složce `config`. Obsahuje 2 soubory:

- `config.local.neon` – Obsahuje nastavení, které je specifické pro lokální server, např. přihlašovací údaje do databáze.
- `config.neon` – Obsahuje nastavení projektu, např. časovou zónu, výchozí presenter pro zpracování errorů apod.

Poslední složka `router` obsahuje třídu, která zajišťuje překlad url adres a určuje, který presenter a akce se má provést a jaké parametry jim poslat.

V poslední řadě obsahuje důležitý soubor `bootstrap.php`, který zajišťuje nastartování celého frameworku a načtení konfigurace.

log

Zde Nette ukládá události a chybové stavy. Na produkční verzi webu jsou již veškeré chybové hlášky vypnuty, aby nerušily návštěvníka stránky. Ale přesto je potřeba mít přehled o chybách, které mohly nastat a proto se chyby vypíší do této složky.

temp

Složka pro veškeré dočasné soubory, které si Nette vytvoří.

vendor

Zde jsou umístěny knihovny frameworku Nette. Zde se také ukládají rozšíření nainstalované nástrojem Composer.

www

Tato složka představuje veřejnou část webu. Jsou zde umístěny soubory, které jsou volně přístupné přes prohlížeč.

Nejdůležitějším souborem zde je `index.php`, který je prvním načteným souborem webu a ten následně nastartuje celý systém zavoláním souboru `bootstrap.php` nebo v případě údržby webu soubor `maintenance.php`, který zabrání načtení systému a oznámí návštěvníkům, že web je momentálně nedostupný kvůli údržbě.

Zbytek složky představuje styly stránky, veškeré použité pluginy, soubory JavaScriptu a knihovny stažené nástrojem Bower (kapitola 2.8).

5.2 Zabezpečení

Do administrační části webu musí mít přístup pouze povolané osoby, proto využíváme přihlašovací obrazovky a zabezpečovacích prostředků, které nám Nette nabízí.

5.2.1 Šifrování dat

Pro zajištění bezpečnosti se heslo neukládá v čitelné formě. Ukládá se pouze otisk (tzv. hash), který je nemožné zrekonstruovat zpět do původní podoby, ale je důležité použít kvalitní algoritmus pro vytvoření otisku pro heslo.

Při autentizaci využíváme knihovny `Nette\Security\Passwords`, kterou nám Nette poskytuje. Nabízí nám následující metody:

- `hash($password, array $options = NULL)` – Tato metoda vygeneruje otisk pro dodané heslo `$password` pomocí moderního algoritmu bcrypt. Algoritmus počítá otisk iterativně tolikrát, kolik je exponenciální hodnota parametru `cost`, který můžeme uvést do `$options`. Výchozí hodnota tohoto parametru je 10.
- `verify($password, $hash)` – Zjistí, zda se uvedené heslo shoduje s otiskem, tak, že heslu spočítá otisk stejným způsobem, jakým bylo vygenerováno a porovná ho s otiskem, který je v databázi k danému uživateli.
- `needsRehash($password, array $options = NULL)` – Zjistí, zda je potřeba heslu znovu spočítat otisk. Lze jí opět nastavit parametr `cost`.

5.2.2 Přihlašování

Každý uživatel je představován instancí třídy `Nette\Security\User`, která nám dokáže snadno zjistit, zda je uživatel přihlášený metodou `isLoggedIn()`. Po přihlášení nám také poskytuje informace o uživateli.

O proces přihlašování se stará třída `UserManager`, která obsahuje metodu `authenticate (array $credentials)`, kde `$credentials` obsahuje jméno a heslo. Tato metoda nám zjistí, pomocí již zmíněné metody `verify`, zda se přihlášení zdařilo. Při úspěšném přihlášení také zkontroluje, zda je heslo potřeba znovu spočítat pomocí metody `needsRehash`.

Pro rozdělení částí webu, které potřebují být zabezpečeny, využíváme 2 presenterů, od kterých dědí každý další vytvořený presenter:

- `BasePresenter` – Základ pro všechny vytvořené presentery. Všechny metody a vlastnosti definované v této třídě, jsou dostupné i všem třídám, které z této dědí.
- `BaseSecuredPresenter` – Tato třída také dědí od třídy `BasePresenter` a navíc přidává do předdefinované metody `startup()`, která se spouští automaticky před vykreslením šablony, kontrolu, zda je uživatel přihlášený. V případě, že není, je uživatel přesměrován na přihlašovací obrazovku. Od této třídy dědí každý presenter, do kterého není umožněn přístup nepřihlášeným uživatelům, v našem případě všechny presentery administrace. Pro zvýšenou bezpečnost trvá přihlášení pouze 120 minut, poté je uživatel automaticky odhlášen.

5.3 Práce s databází

Jak jsme si již řekli, jediná část systému, která přistupuje k databázi jsou modely. Pro jednoduchý přístup do databáze ze všech modelů je vytvořen základní model s názvem `BaseManager`, ze kterého dědí všechny ostatní modely a ve kterém je definována třídní proměnná `database`, pomocí které přistupujeme do databáze.

Pro práci s tabulkami využíváme 2 knihoven, které nám Nette nabízí:

- `Nette\Database\Context` – Vytváří kontext databáze, který se používá pro přístup k tabulkám v databázi. Pomocí kontextu můžeme vybrat celou tabulku v databázi metodou `$database->table('tabulka');`.

- `Nette\Database\Table\Selection` – Díky této knihovně můžeme dále vyhledávat, filtrovat, vkládat, aktualizovat a mazat data z tabulky. Například můžeme rychle získat záznam z tabulky podle primárního klíče posloupností metod `$database->table('students')->get($id);` a následně tento záznam i aktualizovat připojením `->update(array $changes)` k výsledku předchozího dotazu. Proměnná `$changes` zde představuje asociativní pole dvojic `název => hodnota`, kde `název` je sloupec v databázi a `hodnota` je nová hodnota záznamu v tomto sloupci. Také zajišťuje automatické spojování tabulek přes cizí klíče, nemusíme proto používat pracné `JOIN` příkazy.

Každý řádek získaný pomocí knihovny `Selection` se stávají instancemi třídy `Nette\Database\Table\ActiveRow`. Díky tomu můžeme dále se záznamy velice jednoduše pracovat. K datům můžeme přistupovat jako k členským proměnným (`$student->name`) nebo přes klíče pole (`$student['name']`).

5.3.1 Routování

Routování představuje způsob, jakým jsou překládány URL adresy a vysvětluje následné hledání požadovaného presenteru a akce v něm, která se má použít pro danou URL. Stejně tak i generování URL adresy z poskytnutého presenteru a akce.

Díky routování v Nette už nemusíme do šablon přímo psát URL. Odkaz vytvoříme jednoduše pomocí n:makra (kapitola 2.10.2) a framework jej vytvoří sám:

```
<a n:href="Admin:student:edit $id">Upravit studenta</a>
```

Tento kód vytvoří odkaz do modulu Admin, presenteru StudentPresenter a akce editAction, do které předá parametr `$id`.

Překlad adres a definici způsobu překladu zajišťuje třída `app\router\RouterFactory`. V této třídě jsou definovány masky adres, seřazeny podle priority a Nette při každém načtení stránky prochází seznam masek a hledá shodu s aktuální adresou. V tomto systému jsou definovány různé způsoby překladu pro veřejnou část a pro část administrační. Pro vstup do administrační části musí být za adresou stránky uvedeno `\admin\`, pomocí čeho Nette rozpozná masku, která uvádí, že má směřovat do Admin modulu. Celá tato maska má tvar `admin\<presenter>\<action>\<id>`, kde jednotlivé části znamenají:

- **<presenter>** – Označuje, že na tomto místě má být název presenteru, na který se Nette odkáže.

- **<action>** – Akce v daném presenteru, která se použije.

- **<id>** – Unikátní identifikátor, který je předán do akce.

Tyto části v masce mají definovány výchozí hodnoty, není proto nutné pokaždé uvádět celou adresu, aby proběhlo rozpoznání masky. Stačí například uvést pouze presenter a Nette automaticky přejde na definovanou výchozí akci.

Pro případ, kdy není nalezena cesta k akci, je vytvořena třída `ErrorPresenter`, která dokáže podle uvedeného kódu chyby vygenerovat patřičnou šablonu s textem ke vzniklé chybě.

5.4 Popis hlavních částí

V této sekci si detailně popíšeme funkčnost hlavních částí systému.

5.4.1 Veřejná sekce

Část webu, která je přístupná pro veřejnost, obsahuje jedinou komponentu: plugin `DataGrid` od `Ublaboo` (obrázek 5.2). Tento plugin vypíše z databáze rozvrh prezentací, to zahrnuje: čas prezentace, jméno studenta, login studenta, jméno školitele, čas prezentace, ročník studenta a typ studia. Každý z těchto sloupečků je libovolně seřaditelný, vzestupně i sestupně. U těch sloupečků, u kterých to dává smysl je zahrnuto i filtrovací políčko.

Datum a čas	Student	Login	Školitel	Ročník	Typ studia	Délka prezentace
					-- Všechno --	
1.1. 16:30	Hypský Roman	xhypsky01	Kreslíková Jitka, doc. RNDr., CSc	2	Prezenční	5
19.1. 16:30	Láznička Stanislav	xlaznic00	Meduna Alexander, prof. RNDr., CSc.	3	Prezenční	12
19.1. 16:40	Kajan Dušan	xkajan01	Meduna Alexander, prof. RNDr., CSc.	1	Prezenční	5
7.4. 12:30	Hypský Roman	xhypsky01	Kreslíková Jitka, doc. RNDr., CSc	2	Prezenční	5
27.4. 21:48	Tomáš Jínek	xjinek00	Burget Radek, Ing., Ph.D.	2	Prezenční	8
10.5. 16:30	Jiří Bašta	xbasta03	Burget Radek, Ing., Ph.D.	1	Prezenční	5

Obrázek 5.2: Ukázka pluginu Ublaboo/DataGrid

Prakticky to bude fungovat tak, že student navštíví stránku a chce se podívat v jaký čas se koná jeho prezentace. Do filtrovacího políčka zadá svůj login nebo své jméno a řádky tabulky se automaticky, pomocí AJAXu, vyfiltrují a studentovi se zobrazí jediný řádek. Nebo školitel, který má pod sebou více studentů, se bude chtít podívat kdy má nejbližší termíny prezentací studentů. Zadá do filtrovacího políčka ke školitelům své jméno a automaticky se zobrazí pouze záznamy, ve kterých je uveden jako školitel.

5.4.2 Správa školitelů

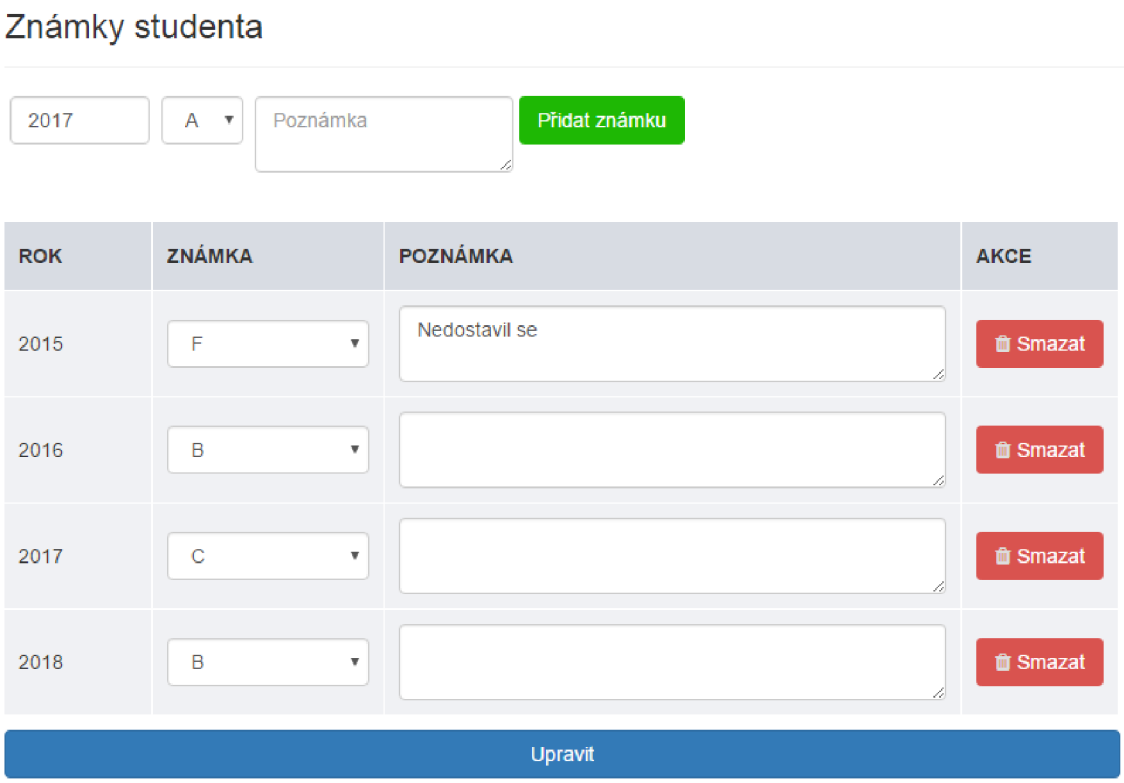
Obsahuje možnost přidání záznamu a tabulku s výpisem existujících záznamů o školitelích, včetně možností na úpravu a smazání každého z nich. U školitelů ukládáme pouze informaci o jméně školitele a v úpravě školitele je vypsán seznam studentů, které momentálně školí, včetně odkazu na úpravu daného studenta.

5.4.3 Správa studentů

V této sekci webu je struktura stejná jako ve správě školitelů, proto se zaměříme spíše na úpravu a přidávání nového studenta. Úprava i přidávání obsahuje formulář s informacemi o jméně, loginu, školiteli, ročníku, typu studia a délce prezentace studenta. Pro usnadnění práce administrátora je přidávání školitele ke studentovi řešeno 2 možnými způsoby. Administrátor má na výběr, zda vyplní jméno nového školitele, který zatím nebyl evidovaný a bude tímto přidán do databáze nebo pomocí výběrového pole označí již existujícího školitele. Tyto 2 vstupy jsou ošetřeny, aby zabránily vyplnění obou polí nebo žádného.

Navíc obsahuje úprava studenta pod formulářem další tabulku se zavedeným formulářem, která slouží pro správu známek studenta (obrázek 5.3). Každá známka obsahuje informace o roku, v jakém byla udělena, o známce samotné a poznámce k dané známce. Při

přidávání známky je možné zvolit rok známky, kde je automaticky předvyplněn aktuální akademický rok, zatímco u změny známky je možné měnit pouze údaje o známce a poznámce. Změna známek se provádí hromadně pro všechny záznamy známek studenta, takže je možné změnit více záznamů zároveň a potvrdit formulář jediným tlačítkem. Přidávání známky je ošetřeno proti duplicitnímu roku známky.



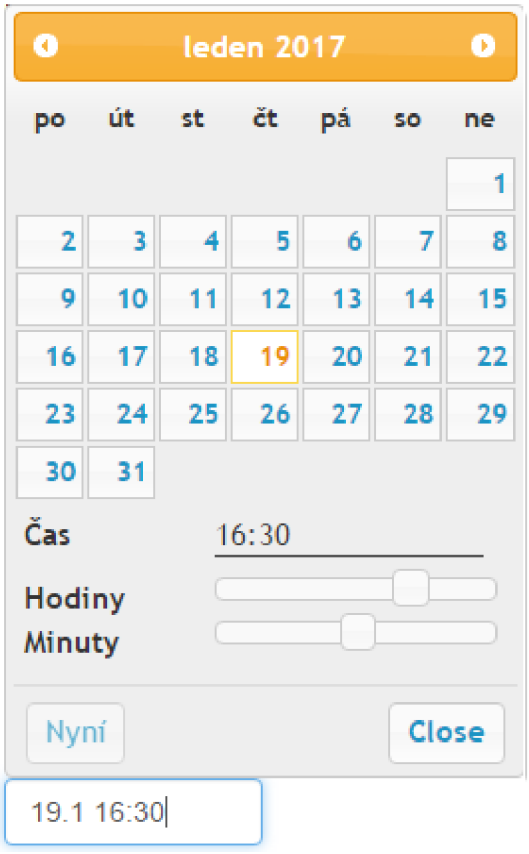
Obrázek 5.3: Formulář pro správu známek studenta

5.4.4 Tvorba rozvrhu

Opět obsahuje obvyklou strukturu správy, s tím rozdílem, že obsahuje nad výpisem rozvrhu jednořádkový formulář pro zvolení roku rozvrhu. Tento rok je zobrazen na veřejné části webu nad výpisem rozvrhu a je takto řešen problém importu rozvrhu do dalších let. Návštěvníci veřejné části proto uvidí, zda je rozvrh zastaralý z minulého akademického roku nebo už je aktuální.

Úprava a přidávání záznamů rozvrhu vychází z formuláře ze správy studenta, rozšířený o pole pro čas (obrázek 5.5). Výběr studenta je zde řešen stejným způsobem jako výběr školitele ve správě studenta. Jelikož u studenta máme více údajů než u školitele a chceme administrátorovi co nejvíc ušetřit čas při vytváření rozvrhu, je zde při výběru existujícího studenta zahrnut AJAX. Při výběru existujícího studenta je zaslán dotaz na server, který se dožaduje informací o studentovi, kterého jsme právě zvolili a jQuery tyto informace zpracuje a vyplní všechna pole automaticky. Jediné, co poté administrátorovi zbývá, je vyplnit čas, což je také vyřešeno co nejpřátelštějším způsobem díky pluginu `DateInput`. Ten zařídí, aby se po kliknutí na políčko s časem zobrazilo okno, ve kterém je zobrazen kalendář, kde si administrátor vybere měsíc a den, a posuvník pro výběr hodin a minut.

Toto pole je, v případě nového záznamu, předvyplněno nejvzdálenějším časem prezentace v databázi nebo aktuálním časem. Vzhled toho pluginu lze vidět na obrázku 5.4.



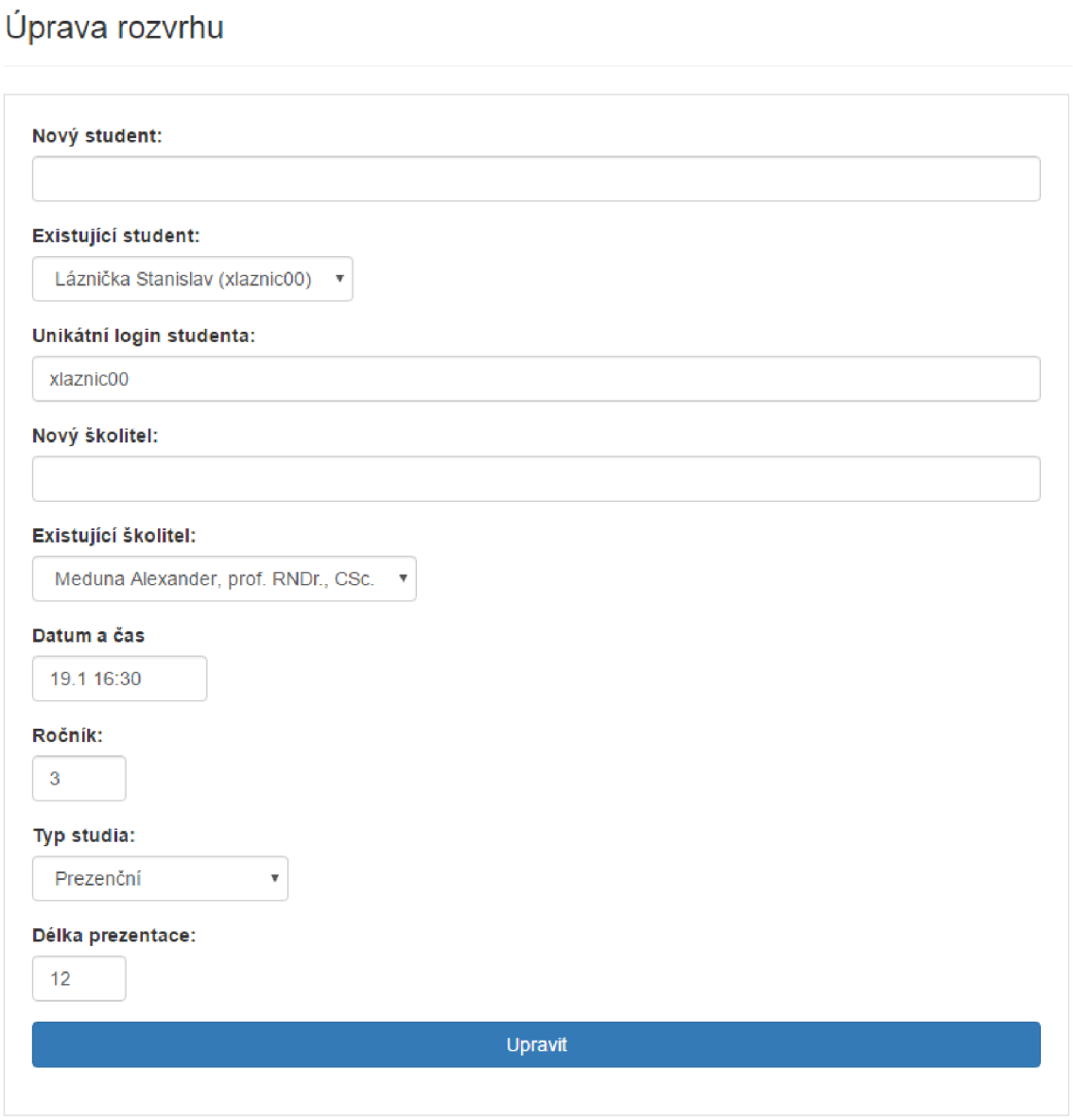
Obrázek 5.4: Ukázka vzhledu pluginu DateInput

5.4.5 Přehled

Výchozí stránka administrace. Obsahuje vypsaný celý rozvrh, seřazený od nejbližšího data. Ke každému záznamu je přiřazen formulář, ve kterém vybíráme známku pro studenta, kterého se záznam týká. Po odeslání formuláře se známka automaticky připíše ke studentovi pro rok, který je uveden v tvorbě rozvrhu jako aktuální. Lze tedy rovnou známkovat studenty při právě probíhajících prezentacích v rámci jedné obrazovky a eliminuje tak potřebu kvůli oznámkování přecházet do karty úpravy studenta. Ukázka je zobrazena na obrázku 5.6.

5.5 Formuláře

Velkou částí implementace administrace je tvoření formulářů, které je značně zjednodušeno a zautomatizováno díky třídě `Nette\Forms`, kterou nám Nette poskytuje. Je díky ní možné tvořit formuláře jako komponenty přímo v presenteru. Například místo pracného vypsání formuláře a každého políčka do šablony, poté v presenteru kontrolovat odeslání formuláře, ošetřovat vstupy kvůli bezpečnosti a kontrolovat validitu vstupů, můžeme napsat komponentu:



Obrázek 5.5: Formulář pro úpravu rozvrhu

DATUM	JMÉNO	ŠKOLITEL	ZNÁMKA	AKCE
25.5 12:00	Tomáš Jínek (xjinek00)	Ing. Radek Burget, Ph.D.	A	Změnit známku
25.5 12:20	Jiří Bašta (xbasta03)	Ing. Radek Burget, Ph.D.	-- Vyberte známku --	Změnit známku
25.5 12:40	Hypský Roman (xhypsky01)	Doc. RNDr. Jitka Kreslíková, CSc.	-- Vyberte známku --	Změnit známku
26.5 13:00	Láznička Stanislav (xlaznic00)	Doc. RNDr. Jitka Kreslíková, CSc.	-- Vyberte známku --	Změnit známku
26.5 13:20	Kajan Dušan (xkajan00)	Prof. RNDr. Alexander Meduna, CSc.	-- Vyberte známku --	Změnit známku
26.5 13:40	Josef Novák (xnovak15)	Ing. Radek Burget, Ph.D.	-- Vyberte známku --	Změnit známku

Obrázek 5.6: Vzhled známkování v rozvrhu

```
protected function createComponentFormular() {
    $form = new UI\Form;
    $form->addText('jmeno', 'Jmeno:');
    $form->addSubmit('odeslat', 'Odeslat');
    $form->onSuccess[] = [$this, 'formularUspel'];
    return $form;
}
```

Tuto komponentu následně vložíme do šablony makrem `{control formular}` a Nette nám zařídí vše ostatní a po úspěšném odeslání formuláře se zavolá metoda `formularUspel` se všemi daty odeslaných formulářem.

Každému políčku formuláře se navíc dají přiřazovat podmínky, které musí být splněny, aby se formulář úspěšně odeslal, např. políčko nesmí být prázdné, věk musí být číslo, číslo musí být v určitém rozsahu apod.

Tvorba nejsložitějšího formuláře v systému, pro tvorbu rozvrhu, je ukázán na obrázku 5.7. Tento kód představuje formulář, který je vidět na obrázku 5.5.

```
$form = new Form;

$nameInput=$form->addText('name', 'Nový student:')->setAttribute('class', 'form-control');
$nameSelInput=$form->addSelect('name_select', 'Existující student:', $students)->setAttribute('class',
'form-control name_select')->setRequired(FALSE)->setPrompt('-- Vyberte studenta --')->setValue($action=
='edit'?$student->id:NULL)->setAttribute('onchange', 'redrawStudent($(this).val(), $(this));');
$nameSelInput->addConditionOn($form['name'], Form::BLANK)->addRule(Form::FILLED, 'Musí být vybrán
student nebo vytvořen nový.');
$nameSelInput->addConditionOn($form['name'], Form::FILLED)->addRule(Form::BLANK, 'Může být vybrána
pouze jedna možnost.');
$nameInput->addConditionOn($form['name_select'], Form::FILLED)->addRule(Form::BLANK, 'Může být vybrána
pouze jedna možnost.');
$nameInput->addConditionOn($form['name_select'], Form::BLANK)->addRule(Form::FILLED, 'Musí být vybrán
student nebo vytvořen nový.');

$form->addText('login', 'Unikátní login studenta:')->setValue($action=='edit'?$student->login:'')->
setAttribute('class', 'form-control login')->setRequired()->setAttribute('n:snippet', 'login');

$teacherInput=$form->addText('teacher', 'Nový školitel:')->setAttribute('class', 'form-control');
$teacherSelInput=$form->addSelect('teacher_select', 'Existující školitel:', $teachers)->setAttribute(
'class', 'form-control teacher_select')->setRequired(FALSE)->setPrompt('-- Vyberte školitele --')->
setValue($action=='edit'?$teacher->id:NULL);
$teacherSelInput->addConditionOn($form['teacher'], Form::BLANK)->addRule(Form::FILLED, 'Musí být vybrán
školitel nebo vytvořen nový.');
$teacherSelInput->addConditionOn($form['teacher'], Form::FILLED)->addRule(Form::BLANK, 'Může být
vybrána pouze jedna možnost.');
$teacherInput->addConditionOn($form['teacher_select'], Form::FILLED)->addRule(Form::BLANK, 'Může být
vybrána pouze jedna možnost.');
$teacherInput->addConditionOn($form['teacher_select'], Form::BLANK)->addRule(Form::FILLED, 'Musí být
vybrán školitel nebo vytvořen nový.');

$datetime=new DateTime($action=='edit'?$schedule->date:$date);
$form->addDate('date', 'Datum a čas', DateInput::TYPE_DATETIME)->setRequired()->setAttribute('class',
'form-control date')->setAttribute('placeholder', 'DD.MM HH:mm')->setDefaultValue($datetime);

$form->addInteger('class', 'Ročník:')->setValue($action=='edit'?$student->class:1)->setAttribute(
'class', 'form-control class')->setRequired();
$form->addSelect('type', 'Typ studia:', $this->Smodel->getTypes())->setAttribute('class', 'form-control
type')->setPrompt('-- Vyberte typ studia --')->setValue($action=='edit'?$student->type:NULL)->
setRequired();
$form->addInteger('p_length', 'Délka prezentace:')->setValue($action=='edit'?$student->p_length:5)->
setAttribute('class', 'form-control p_length')->setRequired();

$form->addSubmit('send', $action=='edit'?'Upravit':'Vytvořit')->setAttribute('class', 'btn btn-primary
btn-block');

$form->onSuccess[] = [$this, 'scheduleFormSucceeded'];
return $form;
```

Obrázek 5.7: Tvorba složitějšího formuláře v Nette

Kapitola 6

Testování

Testování a ladění funkčnosti celého systému probíhalo po celou dobu vývoje. Po každé implementaci jakékoliv menší či větší části systému proběhlo manuální testování vstupů a situací, které by mohly nastat při používání uživatelem. Nakonec proběhlo testování systému jako celek, kdy jsem vymazal celou databázi a v roli uživatele si zkusil vytvořit rozvrh prezentací pro následující rok a následně na veřejné stránce vyhledal informace o času prezentací různých školitelů a různých studentů.

Při implementaci byla většina syntaktických chyb zachycena editorem SublimeText a ty zbylé hlásil debugger Nette frameworku, který se nazývá `Tracy` nebo česky laděnka. Při vypisování proměnných nelze použít klasické PHP funkce `var_dump()`, jelikož Nette používá téměř pro vše své vlastní struktury. Namísto toho bylo použito metody laděnky `Tracy\Debugger::dump()`, která dokáže vypsat celé struktury s možností skrývání a zobrazování jednotlivých částí struktury.

Kapitola 7

Závěr

Cílem této práce bylo vytvořit informační systém, usnadňující tvorbu a správu rozvrhu, který bude sloužit doktorandům a školitelům ke snadnému zjišťování informací o časech jejich prezentací. Výsledkem je systém, postavený na frameworku Nette s architekturou MVP, který dokáže evidovat a spravovat studenty, školitele, dokáže lehce vytvářet rozvrh, importovat jej do dalších let a přímo u rozvrhu udělovat známky za prezentace. Rozvrh je veřejně dostupný a je velice jednoduché v něm najít požadované informace.

Začátkem práce jsme byli seznámeni se všemi technologiemi a nástroji, využívanými při vývoji tohoto systému, které značně ulehčovaly práci. Následující kapitola se věnovala analýze požadavků na systém, kde byly specifikovány požadavky, které se od systému vyžadovaly. Tyto požadavky byly následně vizuálně zobrazeny pomocí diagramu případů užití. Následoval návrh systému, ve kterém byla navrhnuta struktura databáze, zobrazena ER diagramem a detailně vysvětlena. Byla také vysvětlena architektura systému a návrhový vzor MVC. Poté, v implementační části práce, byla vysvětlena adresářová struktura Nette, jaké zabezpečení bylo použito, jak se v systému pracuje s databází a detailní popis hlavních sekcí v něm. Jako poslední bylo vysvětleno, jak probíhalo testování při vývoji a jaké testovací prostředky byly použity.

Systém byl vyvíjen s ohledem na budoucí rozšíření. Jedním takovým by mohlo být vytvoření více rolí administrátora s různými oprávněními, např. administrátor s nejnižším oprávněním by měl přístup pouze k udělováním známek v přehledu administrace apod. Dalším takovým rozšířením by mohlo být vytvoření sekce pro studenta, který by si po přihlášení mohl prohlédnout své známky.

Literatura

[1] *Bootstrap – The world's most popular mobile-first and responsive front-end framework.* [Online; navštíveno 10.03.2017].
URL http://getbootstrap.com/

[2] *Bower – a package manager for the web.* [Online; navštíveno 10.03.2017].
URL https://bower.io/

[3] *CakePHP – Build fast, grow solid.* [Online; navštíveno 10.03.2017].
URL https://cakephp.org/

[4] *Composer.* [Online; navštíveno 10.03.2017].
URL https://getcomposer.org/

[5] *Laravel – The PHP Framework For Web Artisans.* [Online; navštíveno 10.03.2017].
URL https://laravel.com/

[6] *Rychlý a pohodový vývoj webových aplikací v PHP.* [Online; navštíveno 28.02.2017].
URL https://nette.org/cs/#toc-features

[7] *Zend Framework.* [Online; navštíveno 10.03.2017].
URL https://framework.zend.com/

[8] Bittner, H.: *Úvod do CSS preprocesoru Sass.* [Online; navštíveno 26.02.2017].
URL https://www.itnetwork.cz/html-css/webove-portfolio/tutorial-moderni-webove-portfolio-sass

[9] Chacon, S.; Straub, B.: *Pro Git: Second Edition.* Apress, 2014, ISBN 978-1484200773.

[10] Daněk, P.: *Velký test PHP frameworků: Zend, Nette, PHP a RoR.* [Online; navštíveno 28.02.2017].
URL https://www.root.cz/clanky/velky-test-php-frameworku-zend-nette-php-a-ror/

[11] Hertel, M.: *Aspects of AJAX.* [Online; navštíveno 12.03.2017].
URL http://www.mathertel.de/Ajax/AspectsOfAJAX0704.pdf

[12] Netto, D.; Karpov, V.: *Professional AngularJS.* Wrox, 2015, ISBN 1118832078.

[13] Čápka, D.: *Úvod do Nette frameworku pro PHP.* [Online; navštíveno 10.04.2017].
URL https://www.itnetwork.cz/php/nette/zaklady/uvod-do-php-frameworku-nette/

[14] Čápka, D.: *Úvod do PHP a webových aplikací*. [Online; navštíveno 26.02.2017].
URL https://www.itnetwork.cz/php/zaklady/php-tutorial-uvod-do-webovych-aplikaci

[15] Prettyman, S.: *Learn PHP 7: object oriented modular programming using HTML5, CSS3, Javascript, XML, JSON, and MYSQL*. Apress, 2015, ISBN 978-1-4842-1730-6.

[16] Skvorc, B.: *The Best PHP Framework for 2015: SitePoint Survey Results*. [Online; navštíveno 28.02.2017].
URL https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

[17] tutorialspoint.com: *MySQL Tutorial*. [Online; navštíveno 25.02.2017].
URL http://www.tutorialspoint.com/mysql/

[18] w3schools.com: *CSS Tutorial*. [Online; navštíveno 25.02.2017].
URL https://www.w3schools.com/css/

[19] w3schools.com: *HTML5 Tutorial*. [Online; navštíveno 25.02.2017].
URL https://www.w3schools.com/html/

Přílohy

Příloha A

Obsah přiloženého paměťového média

- /src/ – složka se zdrojovými soubory
- /doc/ – složka s HTML dokumentací vygenerovanou pomocí Doxygenu
- /zprava/ – složka se zprávou ve formátu PDF a její zdrojové soubory
- /db/ – složka s vytvářecím skriptem databáze
- readme.txt – soubor s návodem na instalaci a spuštění

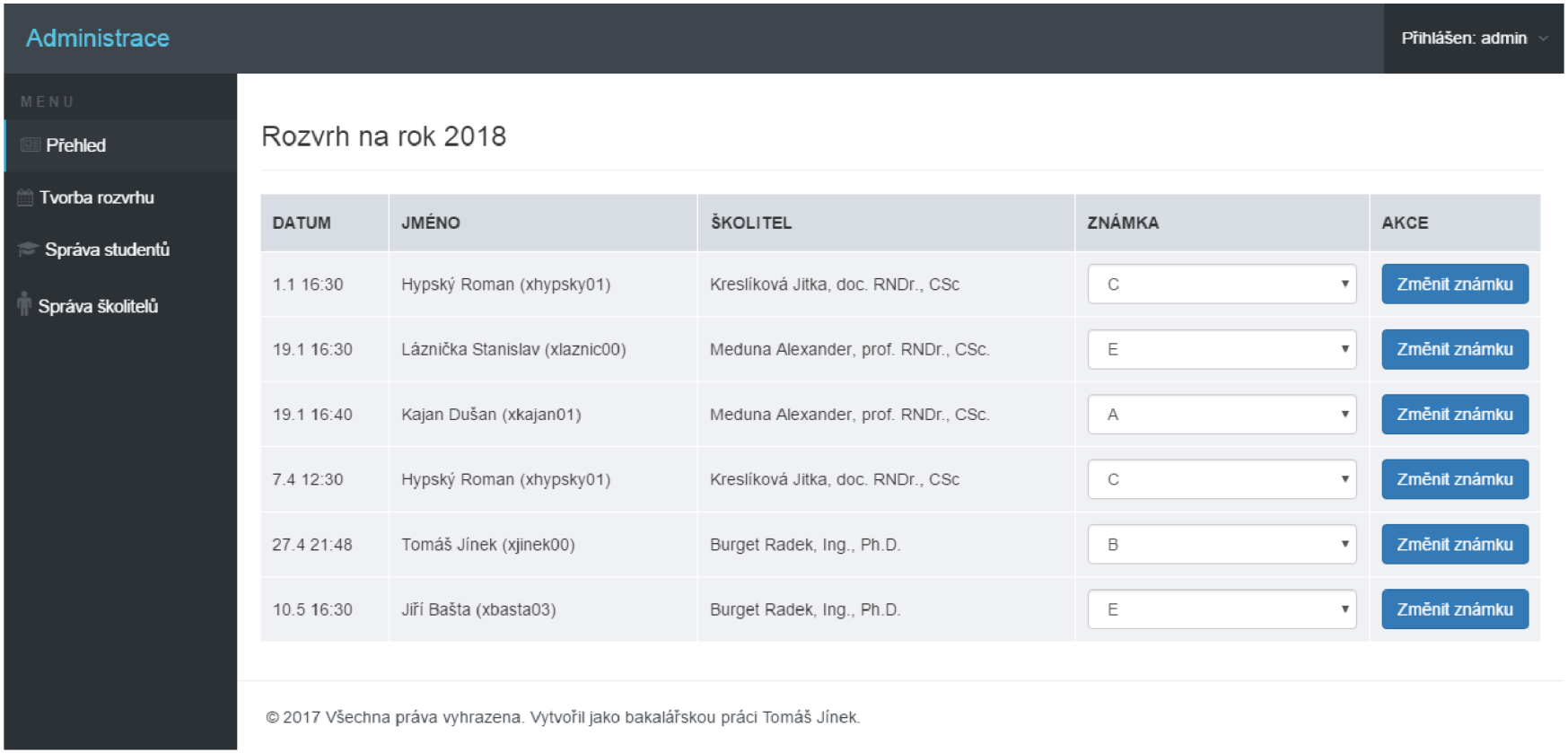
Příloha B

Ukázka vzhledu systému

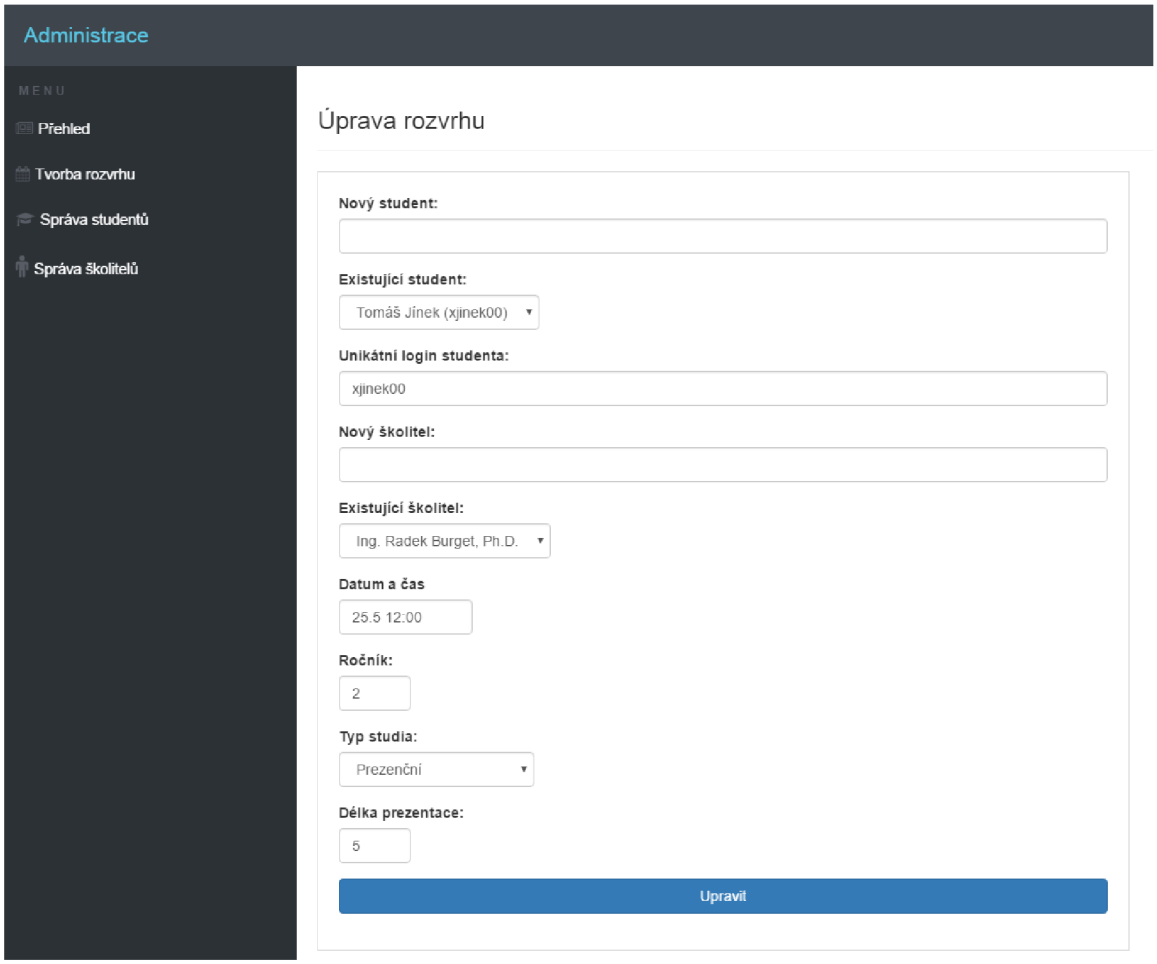
Rozvrh prezentací na rok 2018

Datum a čas	Student	Login	Školitel	Ročník	Typ studia	Délka prezentace
					-- Všechno --	
1.1. 16:30	Hypský Roman	xhypsky01	Kreslíková Jitka, doc. RNDr., CSc	2	Prezenční	5
19.1. 16:30	Láznička Stanislav	xlaznic00	Meduna Alexander, prof. RNDr., CSc.	3	Prezenční	12
19.1. 16:40	Kajan Dušan	xkajan01	Meduna Alexander, prof. RNDr., CSc.	1	Prezenční	5
7.4. 12:30	Hypský Roman	xhypsky01	Kreslíková Jitka, doc. RNDr., CSc	2	Prezenční	5
27.4. 21:48	Tomáš Jínek	xjinek00	Burget Radek, Ing., Ph.D.	2	Prezenční	8
10.5. 16:30	Jiří Bašta	xbasta03	Burget Radek, Ing., Ph.D.	1	Prezenční	5

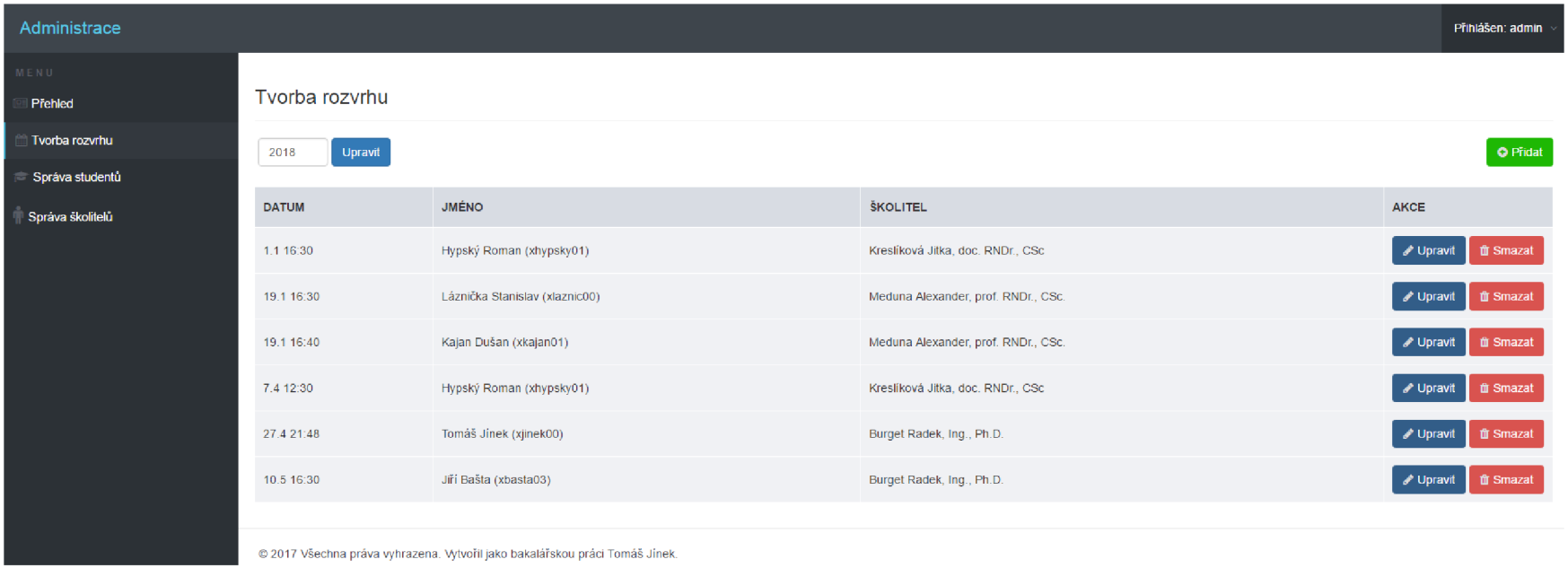
Obrázek B.1: Veřejná část systému



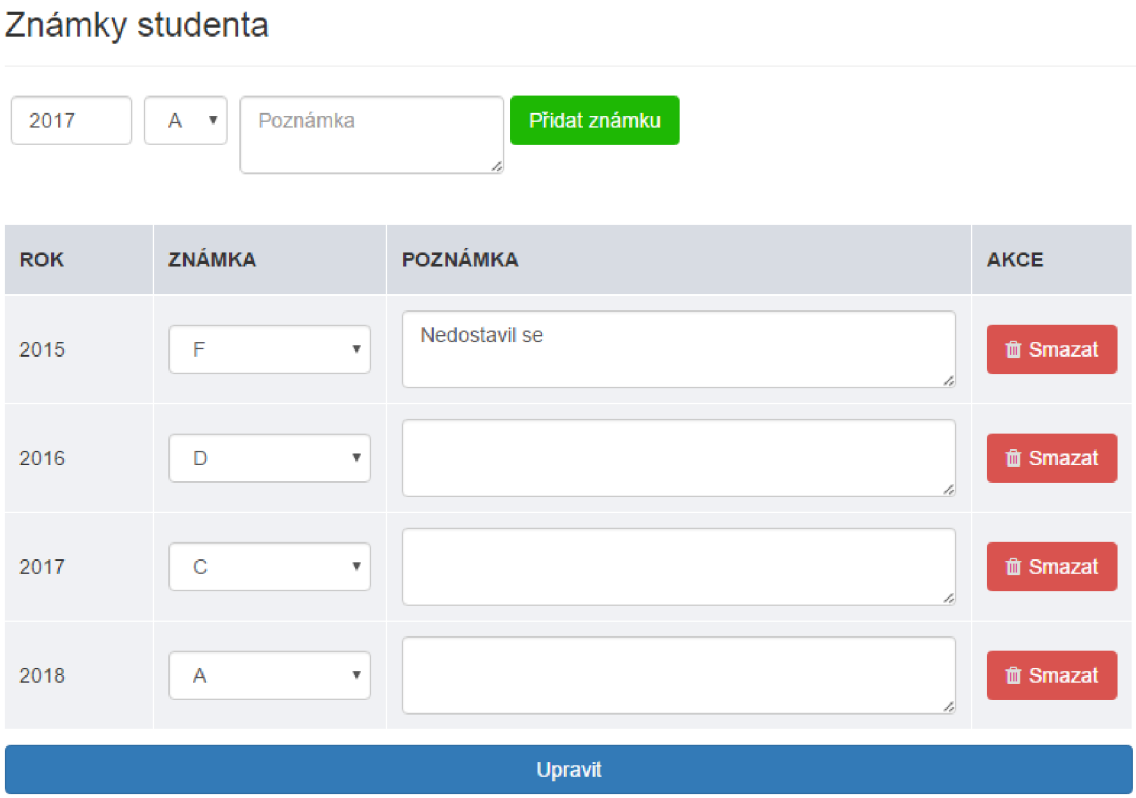
Obrázek B.2: Administrace – Přehled



Obrázek B.3: Administrace – Úprava rozvrhu



Obrázek B.4: Administrace – Správa rozvrhu



Obrázek B.5: Administrace – Správa známek v kartě úpravy studenta