VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO PŘIHLAŠOVÁNÍ NA PREZENTACE

REGISTRATION SYSTEM FOR PRESENTATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ HRNČIAR**
AUTHOR

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

BRNO 2017

Zadání bakalářské práce/19422/2016/xhrnci11

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Hrnčiar Tomáš**
Obor: Informační technologie
Téma: **Systém pro přihlašování na prezentace**
Registration System for Presentations
Kategorie: Informační systémy

Pokyny:

1. Prostudujte současné technologie pro tvorbu webových aplikací s architekturou klient-server.
2. Seznamte se s požadavky na systém pro přihlašování doktorandů na závěrečné prezentace.
3. Na základě analýzy požadavků navrhněte architekturu přihlašovacího systému.
4. Po dohodě s vedoucím implementujte navržený systém pomocí vhodných technologií.
5. Proveďte testování systému v reálném provozu.
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

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Práca sa zaoberá návrhom a tvorbou informačného systému pre administráciu prihlasovania na prezentácie doktorandov. Súčasné riešenie problému je nevyhovujúce a potreba zjednodušiť a zefektívniť organizáciu týchto prezentácií viedla k vzniku tejto práce. Jej súčasťou je teoretický popis použitých technológií, rozbor výhod a nevýhod aktuálneho stavu, analýza požiadavkov, popis spôsobu ukladania dát a aplikačnej logiky informačného systému. Systém sa podarilo implementovať tak, aby spĺňal zadané požiadavky a bol úspešne otestovaný na reálnych dátach.

Abstract

This work is assigned to design and create information system for the matter of administration of registration to PhD students' presentations. Current solution was insufficient and the need of simplier and more effective organisation of presentations led to this work. It consists of theoretical description of used technologies, analysis of pros and cons of current solution, analysis of reqiurements, description of way of storing data and application logic of information system. The information system was fully implemented, it meets all requirements and was successfully tested on real data.

Kľúčové slová

informačný systém, PHP, Nette framework, JavaScript, Bootstrap, web

Keywords

information system, PHP, framework, Javascript, Bootstrap, web

Citácia

HRNČIAR, Tomáš. *Systém pro přihlašování na prezentace*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Systém pro přihlašování na prezentace

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Radka Burgeta, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Tomáš Hrnčiar
16. mája 2017

Poďakovanie

Týmto by som chcel poďakovať Ing. Radkovi Burgetovi, Ph.D. za odborné vedenie mojej práce a cenné rady počas celého semestra. Ďalej by som sa chcel poďakovať Martinovi Kolárikovi za rýchle vysvetlenie nejasností, ktoré nastali počas riešenia práce.

Obsah

Kapitola 1

Úvod

Študenti doktorandského štúdia na Fakulte informačných technológií VUT každý rok absolvujú prezentáciu doktorandských prác, kde prednášajú o pokroku, ktorý dosiahli počas školského roka. Prezentácii predchádza príprava, ktorá pozostáva z nevyhnutnej administratívy. Jej súčasťou je aj komunikácia koordinátora prezentácií so študentmi doktorandského štúdia ohľadne potvrdenia termínu prezentácií ako aj vyplnenia dotazníka o činnosti doktoranda za uplynulý akademický rok.

V súčasnosti je táto administratívna práca neúnosne zdĺhavá a časovo veľmi neefektívna. Komunikácia s doktorandmi prebieha pomocou emailu. Všetky odpovede doktorandov si musí administrátor manuálne zaznamenávať. Z hľadiska efektívnej práce ako administrátora, tak aj doktoranda je tento stav neudržateľný. Toto viedlo vedúceho bakalárskej práce k vypísaniu témy na vytvorenie webovej aplikácie, ktorá pokryje všetky potreby a poskytne väčšiu pomoc pri administrácii. Je to napríklad opakované odosielanie vyplneného dotazníka, čo umožní doktorandom aktualizovanie už odoslaných údajov v databáze. Zozbierané údaje sa potom zobrazujú administrátorovi v podobe prehľadnej tabuľky. Aplikácia umožňuje jednoduché zdieľanie výstupov s inými zamestnancami fakulty. Cieľom je dosiahnuť maximálne zjednodušenie práce a čo najväčšiu časovú úsporu oproti aktuálnemu stavu.

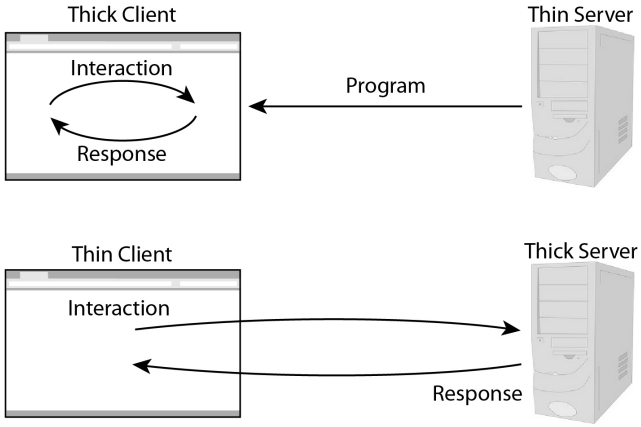
Informačný systém je napísaný v jazyku PHP s využitím frameworku Nette. Osobnou motiváciou autora je záujem o vývoj webových aplikácií a rozvoj znalostí, ktoré s nimi súvisia, a taktiež vyskúšanie si práce na reálnom projekte v danom frameworku.

Práca je logicky členená na jednotlivé ucelené časti, ktoré popisujú fázy vývoja aplikácie. Kapitola 2 sa venuje vysvetleniu rôznych princípov tvorby webových aplikácií. Popisuje hlavné rozdiely medzi použitím tenkého klienta, hrubého servera a opačne, s konkrétnym príkladom používaných technológií.
V nasledujúcej kapitole 3 sú rozoberané dôvody zvolenia frameworku Nette, jednotlivé technológie a metódy, ktoré boli kľúčové pre vytvorenie aplikácie.
Požiadavky zadávateľa práce pre informačný systém z pohľadu administrátora aj doktoranda sú rozpísané v kapitole 4. Nachádza sa tam aj popis architektúry a vysvetlenie použitej dedičnosti pri tvorbe jednotlivých stránok. Je tam ukázané, akým spôsobom sa výsledná stránka skladá z viacerých hierarchicky usporiadaných šablón.
Popis užívateľského rozhrania, formát vstupného súboru, tvorba a odosielanie emailov, ale aj iné implementačné detaily sú popísané v kapitole 5.

Kapitola 2

Aktuálne princípy tvorby webových aplikácií

V dnešnej dobe je dostupné obrovské množstvo technológií, ktoré sa dajú použiť pri vývoji webovej aplikácie. Ich použitie sa rozlišuje v zásade na dve rôzne skupiny. Podľa toho, či väčšinu požiadaviek obsluhuje klient alebo server. To znamená, že buď väčšinu úkonov obslúži klient, v prípade hrubého klienta a tenkého servera alebo server vykoná väčšinu činností, v prípade hrubého servera a tenkého klienta. Porovnanie týchto dvoch prístupov je možné vidieť na obrázku 2.1.



Obr. 2.1: Porovnanie tenkého a hrubého klienta (Zdroj: Carl Sack, 2012)

2.1 Tenký klient, hrubý server

Túto architektúru používajú aplikácie, ktorých veľká časť obsluhy sa deje na serveri. Klientská časť vyžaduje len minimálne kapacitné požiadavky na spracovanie, v porovnaní so serverovou časťou, na ktorej sa nachádza všetka programová logika aplikácie. Klientská časť slúži v podstate len na zbieranie vstupov od užívateľa, ktoré potom odosiela na spracovanie

serverovej časti. Druhou úlohou tenkého klienta je zobrazovať odpovede, ktoré serverová časť po spracovaní úlohy posiela naspäť.

V minulosti bolo použitie tohto prístupu veľmi populárne, čo súviselo aj s obľúbenosťou jazyka PHP v kombinácií s nejakým frameworkom vhodným pre danú aplikáciu. Podľa prieskumu stránky SitePoint[12] z roku 2015, ktorý prebiehal na vzorke približne 7800 čitateľov, sú tri najobľúbenejšie PHP frameworky Laravel, Symfony a Nette - v tomto poradí.

PHP frameworky zjednodušujú vývoj webových aplikácií napísaných v jazyku PHP. Poskytujú základnú štruktúru kódu, na ktorej by mala byť postavená každá webová aplikácia. Inými slovami urýchľujú vývoj stabilných aplikácií a predchádzajú opakovanému programovaniu tých istých častí informačného systému. Je neúnosné, aby sa plytvali ľudské zdroje na programovanie tých istých identických funkcií pri každom novom projekte. Jednoduchšie je použiť framework, v ktorom je daná funkcionalita naprogramovaná a len ho prispôsobiť požiadavkám aplikácie. Tento prístup pomáha predchádzať aj rôznym bezpečnostným chybám, pretože bezpečnosť celej aplikácie nestojí len na konkrétnom programátorovi, ktorý danú aplikáciu vyvíja, ale zodpovedá za ňu celá komunita vývojárov daného frameworku. Medzi spoločné vlastnosti frameworkov patrí použitie šablónovacieho systému, MVC/MVP architektúry a predpripravené prvky, ktoré uľahčujú tvorbu webových aplikácií. Viac informácií o technológiách, ktoré sú použité v rámci tejto práce sa nachádzajú v kapitole 3.2.

2.2 Hrubý klient, tenký server

Druhý typ architektúry, ktorý sa používa na tvorbu webových aplikácií je hrubý klient, tenký server. na rozdiel od tenkého klienta je objem komunikácie medzi klientom a serverom omnoho menší, pretože sa prenášajú len minimálne a nevyhnutné dáta. Princíp spočíva v tom, že si klient najprv stiahne zo servera celú aplikáciu, ktorá sa potom uňho vykonáva. Pokiaľ nastane situácia, že užívateľ vykoná akciu, pri ktorej je potrebná práca s dátami pomocou rozhrania REST, vyžiada si ich a stiahne zo servera. JavaScriptové frameworky React a Angular slúžia na zobrazovanie získaných dát.

REST

Representational state transfer (REST)[5] alebo webová služba RESTful je jeden zo spôsobov, ako môže vyzerať aplikačné rozhranie poskytujúce komunikáciu medzi počítačovými systémami a internetom. REST je orientovaný dátovo a nie procedurálne. Určuje, akým spôsobom sa pristupuje k dátam. Zdrojom môžu byť dáta alebo stavy aplikácie (pokiaľ je možné ich popísať konkrétnymi dátami). Každý zdroj musí byť jednoznačne identifikovaný pomocou identifikátora URI. REST definuje štyri základné metódy pre prístup k nim. Sú známe pod skratkou CRUD - Create (vytvorenie), Retrieve (získanie), Update (aktualizácia), Delete (zmazanie) a sú implementované pomocou zodpovedajúcich metód HTTP protokolu.

HTTP metódy:

- GET - získanie zdroja, HTTP požiadavka musí obsahovať URI adresu pre identifikovanie požadovaného objektu.
- POST - metóda slúži na odoslanie dát. Keďže v tomto prípade neexistuje identifikátor zdroja, je potrebné komunikovať s dohodnutým identifikátorom - endpoint. Taktiež musí obsahovať meno a heslo užívateľa pre autorizáciu. po odoslaní by mal server vrátiť návratový kód; v prípade úspešného vytvorenia zdroja môže byť súčasťou návratovej správy aj identifikátor zdroja. Pokiaľ nastane chyba, mal by vrátiť chybový kód.
- PUT - podobná metóda ako `POST`, ale namiesto vytvorenia nového zdroja modifikuje už existujúci zdroj.
- DELETE - na základe identifikátora zdroja odstráni zdroj zo serveru.

React

Tvorí najmä zobrazovaciu vrstvu aplikácie a používa sa hlavne na tvorbu interaktívnych užívateľských rozhraní. Jednotlivé prvky sa potom automaticky generujú a obnovujú na základe zmeny dát a stavov. Samotný React nestačí na vytvorenie dynamickej aplikácie, a preto je potrebné ho kombinovať napríklad s AJAXovými požiadavkami.

Výstupom Reactu je vygenerovaný výsledný HTML kód, ktorý je tvorený kombináciou HTML a JavaScriptu. Táto kombinácia sprehľadňuje zdrojový kód, pretože vývojár nemusí pozerať samostatne HTML súbor a javascriptový súbor, ale vidí to všetko pokope. Taktiež je ihneď zrejmé, v ktorej časti kódu sa daný element upravil. Predchádza sa situáciám, kedy sa štýly jedného elementu upravili vo viacerých súboroch. Inak by bol problém určiť, ktorá zmena kde nastala. Teda jednoznačné určenie je príliš zložité. React pracuje so stavmi, ktoré určujú akým spôsobom bude daný komponent vygenerovaný. Umožňuje aj vykresľovanie stránky priamo na servery, čím sa zvyšuje rýchlosť stránky v porovnaní s vykresľovaním až na klientskom zariadení.

AngularJS

AngularJS je javascriptový webový framework, ktorý sa zameriava na tvorbu single-page aplikácií. Kladie si za úlohu zjednodušenie vývoja a testovania týchto aplikácií, využíva pri tom architektúru Model-View-Controller (MVC) alebo Model-View-Viewmodel (MVVM). AngularJS pracuje s HTML stránkou, ktorá obsahuje špeciálne atribúty. Framework ich interpretuje ako direktívy na naviazanie vstupov a výstupov k modelu, ktorý je reprezentovaný štandardnými javascriptovými premennými. Hodnoty týchto premenných môžu byť manuálne nastavené v rámci zdrojového kódu alebo získavané staticky, prípadne dynamicky z JSON zdrojov.

Tento framework je postavený na myšlienke, že deklaratívne programovanie by malo byť použité na tvorbu užívateľského rozhrania a spájať softvérové komponenty, kým imperatívne programovanie sa lepšie hodí na tvorbu programovej logiky. Rozširuje klasické HTML a vďaka dvojsmernému viazaniu dát umožňuje automatickú synchronizáciu medzi modelom

a pohľadom. Výsledkom je, že AngularJS znižuje potrebu práce s DOM[1] a tým sa stáva lepšie testovateľným.

Hlavné úlohy AngularJS:

- Odstrániť manipuláciu s DOM od aplikačnej logiky. Náročnosť závisí od štruktúry zdrojového kódu.
- Oddeliť klientskú časť aplikácie od serverovej. Toto umožní počas vývoja vyvíjať obe časti nezávisle od seba a neskôr ich znovu využiť v iných projektoch.
- Poskytovať štruktúru počas tvorby aplikácie, od návrhu užívateľského rozhrania, cez programovanie aplikačnej logiky až po testovanie.

[1] Document Object Model - objektovo orientovaná reprezentácia HTML dokumentu

Kapitola 3

Použité technológie

V tejto kapitole budú popísané jednotlivé technológie potrebné pre vytvorenie výslednej aplikácie tejto bakalárskej práce. Framework Nette bol zvolený z dôvodu, že v našom geografickom prostredí patrí medzi najpopulárnejšie PHP frameworky. Skúsenosti získané vývojom aplikácie môžu byť využité pri prechode do praxe. Tento český framework má okolo seba komunitu ľudí, ktorí sa mu venujú, či už vo forme písania návodov a dokumentácií, odpovedania na fóre venovanom problémom pri vývoji alebo organizovaním pravidelných stretnutí Poslední sobota. Ako užitočné rozšírenie frameworku sa ukázalo rozšírenie Ublaboo DataGrid[4], ktoré uľahčuje štandardné činnosti pri práci s tabuľkami, akými sú napríklad stránkovanie, radenie, či filtrovanie.

3.1 PHP

Najväčšia časť aplikácie je naprogramovaná pomocou skriptovacieho programovacieho jazyka PHP[2], ktorý sa používa na dynamickú prácu s webovou stránkou. Vykonáva sa na strane servera a k užívateľovi sa prenáša až výsledná stránka. Pokiaľ užívateľ na stránke niečo zmení (napr. vyplní formulár), na pozadí sa odošle zmenená stránka na server, ktorý ju opätovne spracuje a oznámi užívateľovi výsledok spracovania. po každom odoslaní sa musí stránka znovu načítať. Syntax jazyka je inšpirovaná niekoľkými programovacími jazykmi (C, Perl, Java).

3.2 Nette

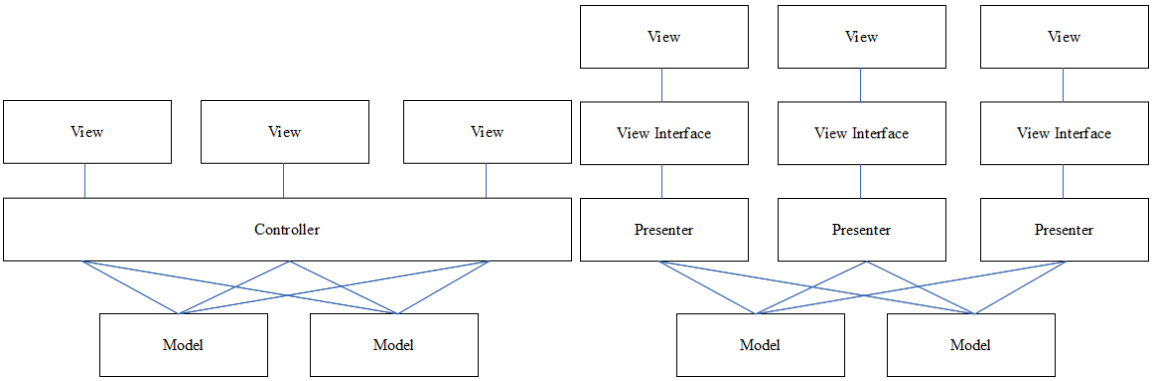
Framework Nette je používaný ako nadstavba nad čistým PHP. Samotné Nette je navrhnuté tak, aby čo najviac uľahčovalo prácu programátorovi. Snaží sa dbať na rýchlosť a bezpečnosť aplikácie. Používa návrhový vzor MVC (Model - View - Controller), komponenty a šablónovací jazyk. Komponenty umožňujú modularitu návrhu a šablónovací jazyk slúži na zobrazovanie samotnej stránky.

Nette obsahuje veľké množstvo rôznych funkcií. Opísať všetky by bolo nad rámec bakalárskej práce, preto v tejto kapitole budú popísané len technológie, ktoré boli použité pri programovaní aplikácie.

MVC

Architektúra Model-View-Controller (Model-Pohľad-Radič)[8] slúži na uľahčenie vývoja, sprehľadnenie aplikácie a zjednodušenie testovania jednotlivých častí. Aplikácia je rozdelená na tri samostatné celky. Model uchováva dáta, ktoré sú získavané na základe požiadaviek radiča a zobrazované v pohľade. Taktiež spracováva požiadavky radiča na dáta. Pohľad slúži na vykreslenie a zobrazenie výsledku. Vie, ako vykresliť jednotlivé komponenty alebo výsledky získané z modelu. V Nette sa používa šablónovací systém *Latte*. Radič spracováva požiadavky od užívateľa a na základe nich volá vhodnú aplikačnú logiku (model) a pohľad pre vykreslenie dát.

Od návrhového vzoru MVC je odvodený návrhový vzor MVP, ktorý sa používa aj vo frameworku Nette, *P* v tomto prípade označuje prezentér. Porovnanie týchto dvoch návrhových vzorov je možné vidieť na obrázku 3.1. pri MVC je jeden radič (controller) spoločný pre viacero pohľadov. Naopak pri architektúre MVP úlohu radiča preberá prezentér, ktorý je samostatný pre každý pohľad. Medzi prezentérom a pohľadom sa ešte nachádza rozhranie pohľadu, ktoré ich prepája.



Obr. 3.1: Rozdiel medzi návrhovými vzormi MVC a MVP

Životný cyklus prezentéra

Úlohu prezentéra je možné zobraziť znázornením životného cyklu[8], viď obrázok 3.2.

1. `startup()`

   Metóda startup() sa volá ako prvá. Slúži na inicializovanie premenných alebo overenie užívateľských oprávnení.

2. `action<Action>()`

   V prípade potreby prezentér vykoná nejakú akciu (napríklad prihlásenie užívateľa, zapísanie dát do databázy) a potom presmeruje stránku niekde inde.

   Vykonáva sa skôr ako metóda render<View>().

3. `handle<Signal>()`

Spracováva signály. Využíva sa hlavne pri komponentoch a spracovávaní AJAXových požiadaviek. Vykonáva aj spracovanie formulárov.

4. `beforeRender()`

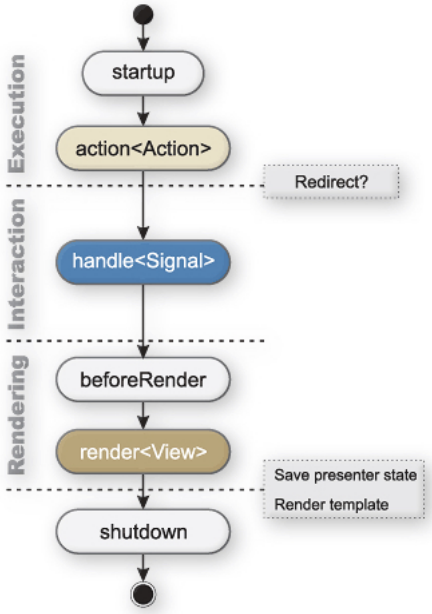
   Vykonáva sa pred metódou render<View>. Môže obsahovať napríklad nastavenie šablóny alebo predanie premenných, ktoré sú spoločné pre viac pohľadov.

5. `render<View>()`

   Vloží do šablóny potrebné dáta.

6. `shutdown()`

   Vykonáva sa na konci životného cyklu prezentéra.



Obr. 3.2: Životný cyklus Nette prezentéra (Zdroj: Nette Foundation, 2017)

Šablónovací systém Latte

Výhoda použitia šablónovacieho systému spočíva v jednoduchšom zápise stránky, ktorá sa má zobraziť. V Latte[7] sa používajú dva typy špeciálnych značiek, sú to makrá a n:makrá. Každé párové makro sa vzťahuje na jeden HTML element a v prípade potreby sa dá prepísať do podoby n:makra použitím špeciálneho atribútu.

N:makrá môžu používať predpony, ktoré mierne modifikujú ich chovanie. Napríklad `n:inner-foreach`, predpona „inner-“ spôsobí, že sa vzťahuje len na vnútornú časť elementu. Okrem párových n:makier existujú aj nie párové napríklad `n:href`.

Dôležitou vlastnosťou šablónovacieho systému Latte z pohľadu bezpečnosti, je automatické escapovanie obsahu premenných, vďaka čomu sa predchádza vzniku bezpečnostných dier

(Cross Site Scripting). vo vnútri makier sa dá používať kód napísaný v jazyku PHP, vrátane komentárov.

pre modifikovanie obsahu premenných je možné použiť vstavané filtre. Zapisujú sa za vertikálu[1] a slúžia na modifikáciu reťazcov (veľkosť, orezanie), na zmenu veľkosti písmen (malé, veľké písmená, kapitalizovať), formátovanie hodnôt (dátum, číslo, veľkosť v bajtoch) a ďalšie (zistenie dĺžky reťazca, kontrola URL adresy od nebezpečných vstupov). Okrem makier, ktoré ponúka samotný framework je možné aj vytváranie vlastných makier.

Zaujímavosťou šablón Latte je aj možnosť ich uloženia nielen v súbore, ale môžu byť uložené aj v premennej (napríklad pri načítaní z databázy).

Formuláre

Formuláre[6] vo frameworku Nette sú navrhnuté tak, aby sa vo všetkom snažili uľahčiť život vývojárovi. Cieľom je, aby užívateľ nemusel riešiť nič iné okrem tvorenia samotného formuláru. Preto je automaticky postarané o validáciu odoslaných dát na strane servera aj JavaScriptu, zabezpečenie voči útokom a únikom dát, viacero režimov vykresľovania, viacjazyčnosť. Formulár je chránený pred útokmi typu Cross Site Scripting (XSS), alebo Cross-Site Request Forgery (CSRF). Automaticky sa postará o ošetrenie vstupných kontrolných znakov, overenie validity UTF-8 kódovania, prípadne skontroluje, či položky vybrané v select boxoch nie sú podvrh. Výhodou je aj to, že programátor nemusí opakovať pri každom novom projekte tie isté činnosti dookola, ale je o ne automaticky postarané.

pre vykreslenie formulára je potrebné jeho odovzdanie do šablóny. vo frameworku Nette sa na to využíva menná konvencia. Funkcia, ktorá vytvára formulár, musí mať pomenovanie vo formáte
`createComponent<názov formuláru>()`, v šablóne sa k nej potom pristupuje pomocou `{control <názov formuláru>}`.

Routovanie URL adries

Routovanie[11] je obojsmerné prekladanie medzi URL a akciami prezentéra. Z URL je možné získať akciu prezentéra a naopak, k akcii prezentéra vygenerovať URL adresu. To je užitočné najmä pri šablónach, kde sa nemusia písať konkrétne URL adresy.

Najjednoduchší router je `SimpleRouter`, ktorý je vhodný, pokiaľ nemáme žiadne požiadavky na výsledný tvar adresy a chceme len čo najjednoduchšie riešenie. Trieda `Route` sa používa, keď potrebujeme vytvoriť jednoducho čitateľnú a zapamätateľnú adresu pre užívateľa, napríklad `http://example.com/product/detail/123`. Adresa v podobnom tvare prispieva aj k lepšej optimalizácií pre vyhľadávacie stránky, čo je v dnešnej dobe užitočné. Tvorí sa pomocou objektu `Route`. Ako prvý parameter má masku cesty, druhý je akcia prezentéra (zapísaná ako reťazec alebo pole), tretí parameter je voliteľný - určuje dodatočné príznaky. Príklad masky: `<presenter>/<action>[/<id>]`, kde sú povinné položky presenter a action a voliteľné id.

Užívateľ zvyčajne potrebuje viac ako jeden formát ciest, na tento účel sa používa objekt `RouteList`, do ktorého zabalíme viacero ciest.

[1] znak |

Framework Nette umožňuje aj vytváranie si vlastného routera, čiže vývojár nie je limitovaný a môže prispôsobiť aplikáciu požiadavkám zadávateľa.

Autentizácia a autorizácia užívateľov

Každá zložitejšia aplikácia musí skôr alebo neskôr riešiť dostupnosť informácií na stránke podľa toho, kto na ňu pristupuje. Administrátorovi sa budú zobrazovať všetky dáta, pričom bežnému užívateľovi len jemu určené a neprihlásenému len tie čo sú verejné pre všetkých.

Nette ponúka vlastné riešenie autentizácie[10] užívateľov. Využíva objekt `user` a jeho metódy, ktoré slúžia na prihlásenie, odhlásenie, prípadne nastavenie automatického odhlásenia v prípade nečinnosti. pre správnu činnosť je potrebné mať funkčný autentikátor. Základným autentikátorom frameworku Nette je `SimpleAuthenticator`, ktorý dostane v konštruktore zoznam užívateľov a hesiel vo forme asociatívneho poľa. Autentikátor sa zvyčajne nastavuje v konfiguračnom súbore.

V prípade, že vývojár potrebuje overovať prihlasovacie údaje, ktoré sú uložené napríklad v databáze, môže si takýto autentikátor sám naprogramovať a potom zaregistrovať v konfiguračnom súbore. Autentikátor vracia súbor informácií o užívateľovi, ktoré sú zvané identita. Tá umožňuje zistiť ID a rolu užívateľa.

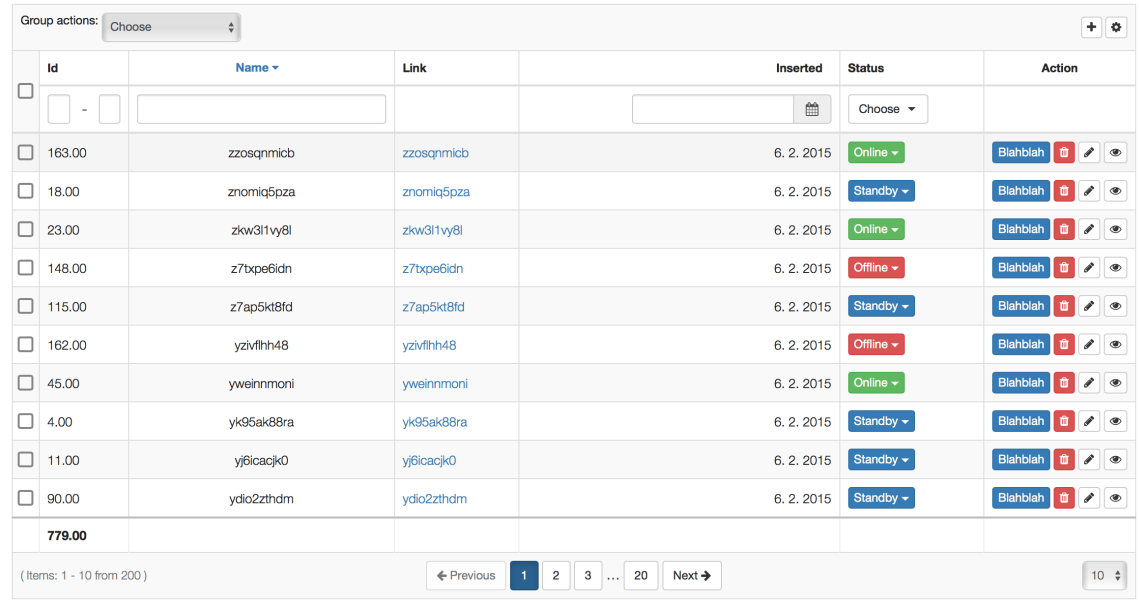
Autorizácia zisťuje, či má užívateľ dostatočné práva na vykonanie určitej akcie. Predpokladom je, že užívateľ je prihlásený. V prípade, že aplikácia nepotrebuje rozlišovať medzi viacerými rolami, stačí nastaviť podmienku, ktorá kontroluje, či je užívateľ prihlásený. Ak áno, získa prístup ku všetkým akciám na stránke. V prípade aplikácií, kde je potreba viacero rozličných rolí, sa ako autorizačné kritérium potom používa metóda `isInRole()`, ktorá zisťuje, či prihlásený užívateľ má požadovanú rolu. Aby programátor nemusel všade dávať podmienky zisťujúce, či je užívateľ prihlásený a zároveň, či má danú rolu, Nette to rieši tak, že neprihlásený užívateľ má automaticky rolu `guest`.

Odosielanie emailov

Posielanie mailov[9] je jedna z hlavných úloh mojej aplikácie. Spôsob jej použitia je rozpísaný v podkapitole 5.6. Samotný email sa tvorí pomocou triedy `Message`, kde sa pomocou jednotlivých metód nastaví odosielateľ, príjemca, predmet, telo správy. Taktiež je možné vložiť do správy HTML obsah, obrázky a iné prílohy. Umožňuje aj používať šablóny, takže sa dajú vytvoriť vzory správ, ktorých obsah sa dynamicky mení podľa potreby. Odosielanie prebieha pomocou natívnej PHP funkcie `mail`, alebo je možné odosielať pomocou SMTP servra, na čo sa využíva `SMTPmailer`.

3.3 Ublaboo DataGrid

Ublaboo DataGrid je doplnok do tohto frameworku. Je vhodný na zobrazovanie tabuliek na stránke. Medzi jeho hlavné funkcie patrí stránkovanie, triedenie podľa stĺpca, počet položiek na stránku, akceptuje dáta na vstupe v rôznych formátoch (napr. Doctrine, Dibi), filtrovanie podľa kritérií, vykonávanie akcií, export do formátu CSV, skrývateľné stĺpce a iné. na obrázku 3.3 je možné vidieť ako vyzerá tabuľka vytvorená pomocou tohto doplnku.



Obr. 3.3: Ukážka možností doplnku Ublaboo Datagrid (Zdroj: Ublaboo, 2017)

3.4 JavaScript

JavaScript[1] je dynamický, netypovaný, interpretovaný, objektovo orientovaný skriptovací jazyk. Okrem objektovo orientovaného programovania je vhodný aj pre funkcionálne programovanie. Používa sa pre dynamickú činnosť s prvkami na webovej stránke. V súčasnosti patrí medzi jednu z troch kľúčových technológií na tvorbu webových stránok (HTML pre špecifikovanie obsahu na stránke, CSS pre špecifikovanie vzhľadu stránky a JavaScript pre špecifikovanie správania stránky). Väčšina webových stránok ho používa a takmer všetky prehliadače ho automaticky podporujú. Vykonávanie skriptu prebieha až po načítaní stránky na strane klienta, na rozdiel od PHP, ktoré sa spúšťajú na strane servera ešte pred stiahnutím z internetu. Z toho vyplývajú určité bezpečnostné obmedzenia, JavaScript napríklad nemôže pracovať so súbormi, aby tým neohrozil súkromie používateľa.

Použitie JavaScriptu:

- Načítanie nového obsahu stránky alebo odoslanie dát na server pomocou Ajaxu bez obnovenia stránky (napr. užívateľ môže vyplniť dotazník a uložiť ho bez opustenia stránky).
- Zobrazenie všetkých dostupných informácií na stránke (napríklad dátum, priezvisko, školiteľ, ročník a iné...).
- Animovanie elementov na stránke, zmena ich veľkosti, hýbanie s nimi atď...
- Interaktívny obsah, hry, prehrávanie hudby a videa.
- Validácia vstupných hodnôt webového formuláru na uistenie sa, či sú zadané hodnoty vhodné pred odoslaním na server.

CodeMirror

CodeMirror[3] je knižnica, ktorá slúži na editovanie textu na webovej stránke. Jadro knižnice tvorí len samotný editor, ku ktorému sa dajú nastaviť rôzne funkcie ako sú napríklad automatické dopĺňanie slov, zvýrazňovanie syntaxe, formátovanie textu alebo konfigurovateľné klávesové skratky.

```
<!-- Create a simple CodeMirror instance -->
<link rel="stylesheet" href="lib/codemirror.css">
<script src="lib/codemirror.js"></script>
<script>
  var editor = CodeMirror.fromTextArea(myTextarea, {
    lineNumbers: true
  });
</script>
```

Obr. 3.4: Ukážka textového editoru CodeMirror (Zdroj: CodeMirror, 2017)

3.5 MySQL

MySQL je je jedným z najrozšírenejších, voľne dostupných relačných databázových serverov na internete. Je podporovaný na všetkých najväčších platformách (Linux, Microsoft Windows, macOS, Solaris). Implementovaný je vo viacerých programovacích jazykoch ako sú napríklad PHP, C++, Perl. Získal si obľubu užívateľov kvôli rýchlosti a jednoduchosti použitia. Jednoduchosť však spôsobila absencia základných databázových prvkov, akými sú napríklad pohľady, triggery a uložené procedúry. V posledných rokoch začala programátorom (najmä webových aplikácií) táto funkcionalita chýbať, čo viedlo k postupnému dopĺňaniu týchto vlastností. na písanie dotazov sa používa jazyk SQL.

3.6 Bootstrap

Bootstrap je v súčastnosti najpopulárnejší HTML, CSS a JS framework pre vývoj responzívnych web stránok.
Hlavné výhody:

- Zabraňuje opakovaniu činností počas vývoja projektov.
- Využíva responzívny dizajn, pre správne zobrazenie stránky, na čo najväčšom počte zariadení.
- Dizajn aj kód sú konzistentné medzi projektami a vývojármi.
- Rýchle a jednoduché prototypovanie nového dizajnu.
- Pomáha zachovávať kompatibilitu medzi rôznymi internetovými prehliadačmi.

V dnešnej dobe plnej rôznych zariadení a webových prehliadačov sa kladie dôraz na správne zobrazenie obsahu. Bootstrap umožňuje vývojárom jednoduchú spoluprácu, pretože všetci používajú rovnaké princípy, jeden framework.

Významnou súčasťou frameworku je mriežkový systém, ktorý sa používa na členenie stránky do stĺpcov so šírkou podľa potreby. Štandardne sa používa rozloženie do 12 stĺpcov, ktoré sa potom navzájom kombinujú tak, aby boli široké podľa potreby užívateľa. Môžu nastať situácie, kedy užívateľ presiahne 12 stĺpcov, v tom prípade sa stĺpce automaticky usporiadajú. V angličtine sa táto operácia nazýva *column wrapping*. Každému stĺpcu je možné priradiť triedy, ktoré bližšie určujú, ako má vyzerať stĺpec na rôznych zariadeniach s inou veľkosťou obrazovky.

Kapitola 4

Návrh informačného systému

Pred začiatkom implementácie je potrebné poznať aktuálne riešenie a všetky požiadavky pre výslednú aplikáciu. Dôležité je, aby boli opravené všetky nedostatky a nevznikali nové. Požiadavky na informačný systém som iteratívne konzultoval s vedúcim práce v priebehu akademického roka.

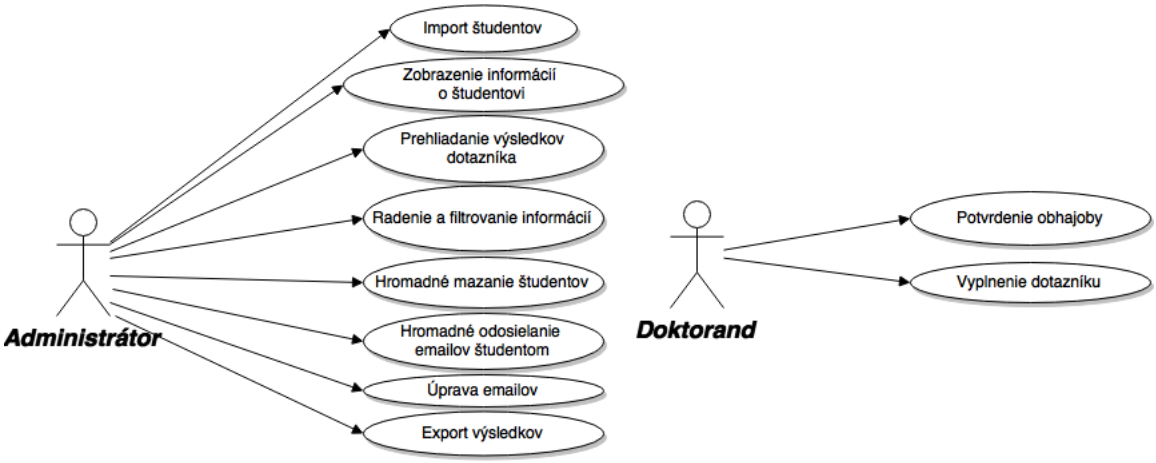
4.1 Aktuálny stav

na administráciu registrácií na prezentácie doktorandských prác sa v súčasnosti používa tabuľkový editor MS Excel. V ňom sa nachádza zoznam všetkých doktorandov, ktorí sa majú danej prezentácie zúčastniť. Administrátor manuálne rozposiela hromadné emaily všetkým doktorandom a zaznamenáva, či od nich dostal potvrdzujúci email s akceptovaním termínu prezentácie. Táto činnosť sa opakuje niekoľkokrát a pokiaľ niekto termín stále nepotvrdil, oznámenie o termíne prezentácií sa odosiela doporučene poštou. Pár týždňov pred začiatkom prezentácií sa tento proces odosielania emailov opakuje a od doktorandov sa očakáva vyplnenie dotazníka o ich činnosti za uplynulý rok. Momentálne riešenie je časovo náročné a potreba zmeniť ho vyústila v túto bakalársku prácu.

4.2 Požiadavky

Výsledkom tejto bakalárskej práce je webová aplikácia, ktorá bude umiestnená na serveri a prístupná cez internet všetkým zúčastneným stranám. Užívatelia sa rozdeľujú na dve role (doktorand, administrátor) a podľa toho, akú rolu zastávajú sa im zobrazuje rozdielny obsah na stránke.

na základe požiadaviek a konzultácií s vedúcim práce bol vypracovaný diagram prípadov použitia viď. obrázok 4.1. Zobrazuje všetkých užívateľov systému a ich akcie v ňom.



Obr. 4.1: Diagram prípadov použitia

Administrátor

V celom informačnom systéme má najväčšiu úlohu, pretože rozposiela emaily a vyhodnocuje ich. Učiteľ, ktorý bude mať prístup do systému, bude len jeden; respektíve môže ich byť aj viac, ale budú zdieľať jedny prístupové údaje. Z tohto dôvodu sa nenachádza v databáze tabuľka s prihlasovacími údajmi viacerých učiteľov. Namiesto toho sa len nakonfiguruje jedno prístupové konto v konfiguračnom súbore.

Rola administrátor:

- Nahratie zoznamu doktorandov do aplikácie pomocou CSV.
- Zobrazenie všetkých doktorandov a informácií o nich (dátum prezentácie, priezvisko, meno, školiteľ, ročník, typ štúdia, dĺžka prezentácie, email, schválené).
- Zoradenie doktorandov podľa mena a podľa stavu (schválené/neschválené).
- Kliknutím na priezvisko doktoranda zobraziť výsledky dotazníku daného študenta.
- Hromadné mazanie doktorandov zo systému.
- Hromadné odosielanie oznámení doktorandom pomocou emailu.
- Úprava emailových šablón na stránke.
- Zobrazenie výsledkov dotazníku všetkých doktorandov v danom roku.
- Export výsledkov do CSV.

Doktorand

Úlohou doktoranda v informačnom systéme sú dve hlavné akcie. Prvá (po obdržaní emailu od administrátora) je samotné otvorenie URL adresy a potvrdenie oboznámenia s termínom prezentácie kliknutím na potvrdzovacie tlačidlo. Druhá akcia je vyplnenie dotazníka na stránke. pri doktorandoch sú v databáze uchovávané:

- ID - primárny kľúč
- dátum prezentácie
- unikátny odkaz - prístup na stránku doktoranda, cudzí kľúč do tabuľky s dotazníkom
- priezvisko
- meno
- školiteľ
- ročník
- štúdium
- email
- informácia o tom, či schválili dátum prezentácie, alebo nie

Registráciu doktorandov na stránke vykonáva administrátor tým, že vloží do systému ich zoznam. Prihlasovanie potom prebieha kliknutím doktoranda na URL adresu, ktorú dostane cez email.

Rola doktoranda:

- Potvrdenie termínu prezentácie kliknutím na odkaz v emaile.
- Vyplnenie dotazníka.

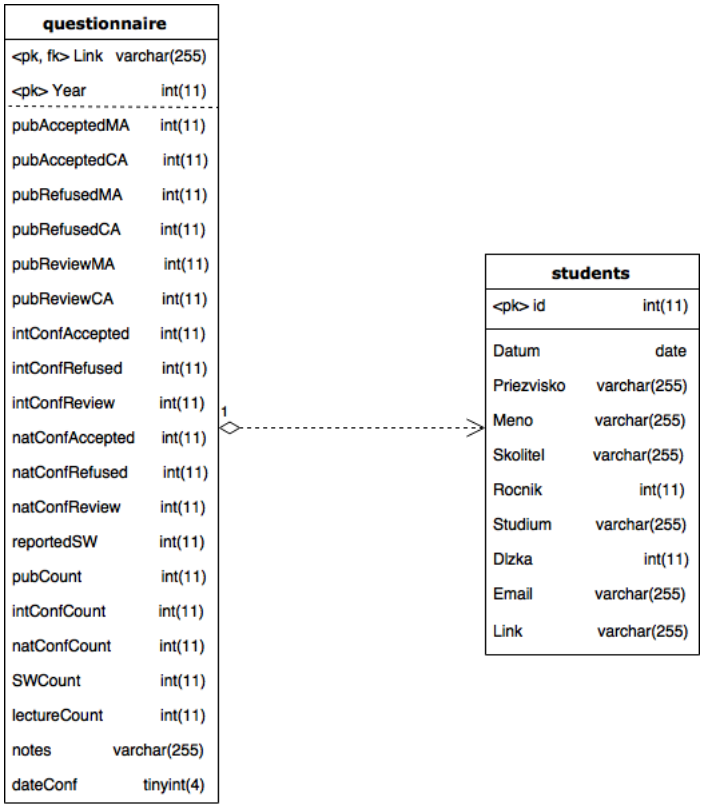
Dotazník

Samotný dotazník nemá v informačnom systéme žiadnu rolu a v podstate len uchováva dáta, ktoré doňho doktorand vložil. Tie sa potom zobrazujú administrátorovi v prehľade doktorandov.

- unikátny odkaz
- akademický rok
- počet schválených príspevkov, kde bol študent autor
- počet schválených príspevkov, kde bol študent spoluautor
- počet odmietnutých príspevkov, kde bol študent autor
- počet odmietnutých príspevkov, kde bol študent spoluautor
- počet posudzovaných príspevkov, kde bol študent autor
- počet posudzovaných príspevkov, kde bol študent spoluautor
- na medzinárodnej konferencii prijaté

- na medzinárodnej konferencii odmietnuté
- na medzinárodnej konferencii posudzované
- na národnej konferencii prijaté
- na národnej konferencii odmietnuté
- na národnej konferencii posudzované
- vykázané softvérové diela
- celkom prijatých v časopise
- celkom na medzinárodnej konferencii
- celkom na národnej konferencii
- celkom softvérových diel
- počet vystúpení na interných seminároch

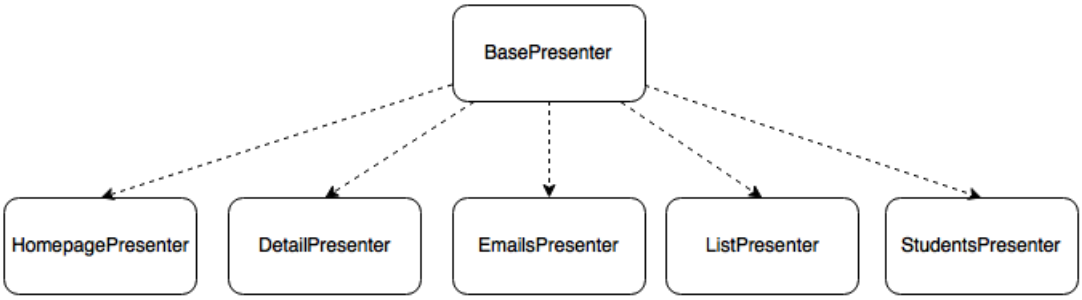
Ako primárny kľúč slúžia unikátny obsah a akademický rok. Jednotlivé časti informačného systému sú zobrazené v ER diagrame, ktorý je možné vidieť na obrázku 4.2. Z toho vyplýva, že rozšírenie polí dotazníka je možné len zásahom do kódu a nie na strane bežného užívateľa prípadne administrátora.



Obr. 4.2: ER diagram

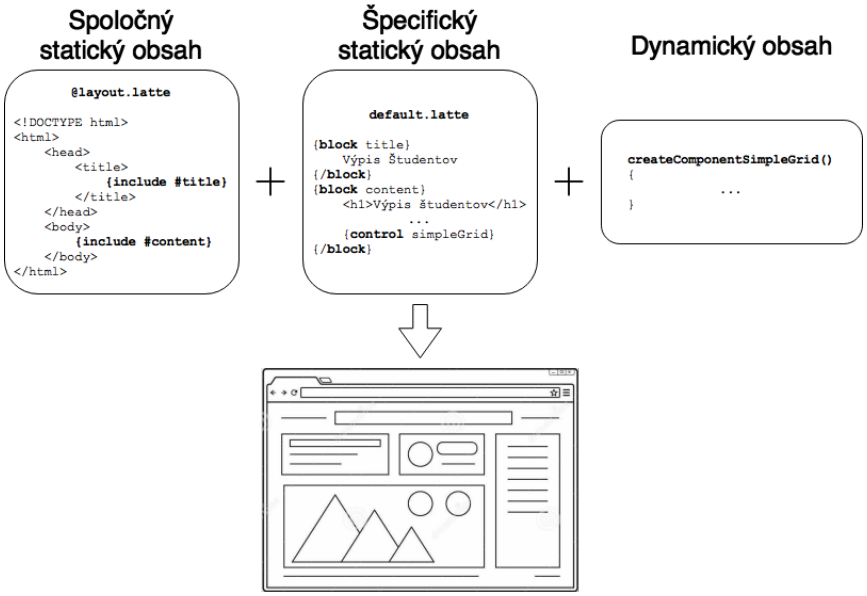
Dedičnosť prezentéra a šablón

Framework Nette je objektovo orientovaný a preto využíva všetky možnosti, ktoré mu táto metodika poskytuje. Jednou z nich je aj dedičnosť. Aby programátor nemusel kopírovať spoločné časti zdrojového kódu do viacerých súborov, používa sa nasledovné. Všetky zdieľané prvky viacerých stránok sú v spoločnom súbore, rozdielne časti má každá stránka v samostatnom súbore. Príklad dedičnosti v mojej práci je možné vidieť na obrázku 4.3. Konkrétne sa jedná o prezentér, pričom základný `BasePresenter` zobrazuje menu na vrchu stránky, rozlišuje aké meno zobraziť podľa prihláseného užívateľa a ktorá stránka je momentálne aktívna. Metódy v prezentéroch sa potom vkladajú do Latte šablóny, ktorá ich vykresľuje.



Obr. 4.3: Dedičnosť

Aj Latte šablóny využívajú dedičnosť pri komponovaní výslednej stránky. Celý proces je znázornený na obrázku 4.4. Je tam vidieť, ako sa do základnej šablóny `@layout.latte`(spoločný statický obsah) vkladá špecifický statický obsah z ostatných šablón (`default.latte`) každého prezentéra. na záver sa k nim ešte vloží dynamický obsah z metód prezentérov (napríklad rôzne formuláre, alebo tabuľky).



Obr. 4.4: Ukážka spájania šablón do výsledného celku

Kapitola 5

Implementácia

V kapitole Implementácia sa pojednáva o databázovej vrstve, adresárovej štruktúre, užívateľskom rozhraní, formáte vstupného súboru, tvorbe formuláru pre dotazník, spôsobe úpravy emailov a ich odosielania, vytváraní URL adries.

5.1 Databázová vrstva

Jedna z vlastností frameworku Nette je aj rozhranie pre prácu s databázou. na pripojenie k nej slúži trieda `Nette\Database\Connection`. Existujú dva spôsoby nastavenia pripojenia, buď pomocou metódy `Connection`, alebo použitím konfiguračného súboru. V mojej práci som zvolil použitie konfiguračného súboru. Z pohľadu administrátora je jednoduchšie nastaviť jeden takýto súbor, ako upravovať jednotlivé zdrojové súbory. Konfiguračný súbor sa nachádza v zložke `/app/config/config.neon`. Ten vytvára okrem spojenia aj službu `Nette\Database\Context`, ktorá slúži na prácu s databázou. Framework Nette podporuje všetky najznámejšie databázové servery, v mojej práci bol využitý databázový server MySQL.

5.2 Adresárová štruktúra

Informačný systém bol tvorený podľa architektúry MVC, preto tomu zodpovedá aj usporiadanie súborov v jednotlivých zložkách. V podkapitole Adresárová štruktúra bude popísaný obsah súborov tak, aby čitateľ mohol jednoducho nájsť tie časti zdrojového kódu, ktoré ho zaujímajú.

Všetky zdrojové kódy sa nachádzajú v zložke `/src/`. Adresárová štruktúra odovzdaných zdrojových súborov je nasledovná:

- `/app/` - obsahuje všetky zdrojové súbory aplikácie, konfiguračný súbor, autentikátor na prihlasovanie, všetky prezentéry aj s ich šablónami, ako sa majú vykresliť, ďalej sa tu nachádza router pre správne vytvorenie URL adresy a SQL súbor pre vytvorenie databáz.

- `/bower.json` - konfiguračný súbor pre nainštalovanie všetkých potrebných knižníc pomocou programu Bower.
- `/composer.json` - konfiguračný súbor pre nainštalovanie všetkých potrebných knižníc pomocou programu Composer.
- `/log/` - priečinok obsahujúci všetky logovacie súbory.
- `/exampleData.csv` - súbor s ukážkou dát vhodných pre import
- `/README.txt` - súbor s návodom na inštaláciu.
- `/temp/` - priečinok pre dočasné súbory, cache.
- `/vendor/` - obsahuje všetky externé knižnice po nainštalovaní použitím programu Composer.
- `/www/` - priečinok obsahujúci inicializačný PHP súbor, CSS súbor so štýlmi pre stránku a emailové Latte šablóny.

Zložky `/temp/` a `/log/` sa v odovzdanom archíve nenachádzajú, pre správny chod aplikácie je potrebné ich vytvoriť v adresári `/src/` s možnosťou zápisu do nich. Framework Nette do nich ukladá rôzne pomocné a dočasné súbory. Zložky `/vendor/` a `/bower_components/` vzniknú po nainštalovaní externých knižníc.

5.3 Popis užívateľského rozhrania

Užívateľské rozhranie je prehľadné, aby užívateľ na prvý pohľad pochopil, ako s ním pracovať. Napriek jeho jednoduchosti sa na pozadí dejú určité operácie, ktoré je vhodné opísať pre lepšie pochopenie činnosti informačného systému.

Z pohľadu administrátora

po prihlásení je administrátor presmerovaný na domovskú stránku (viď. príloha A), kde má možnosť vložiť zoznam doktorandov v aktuálnom roku. Systém automaticky priradí novým študentom doktorandského štúdia kalendárny rok podľa aktuálneho dátumu. Zobrazia sa v tabuľke, kde je možné vidieť základné informácie ako napríklad meno a priezvisko. Administrátor má možnosť zoradiť a filtrovať dáta v niektorých stĺpcoch, ktoré boli dopredu dohodnuté s vedúcim bakalárskej práce. Taktiež je možné skryť celé stĺpce, prípadne exportovať a stiahnuť ich do počítača vo formáte CSV. po označení doktorandov je možné vykonať jednu z dvoch akcií. Prvá je hromadné odoslanie informačného emailu, kde sa dá zvoliť niektorá z predpripravených šablón. Druhá akcia pri označenom doktorandovi je jeho odstránenie zo systému. Administrátorovi je umožnené zobrazenie detailu doktoranda kliknutím na jeho meno v tabuľke.

Keďže informačný systém má uchovávať archív, v momente, keď nastane nový kalendárny rok, záznam o študentovi z predchádzajúceho roku sa nastaví ako trvalý. Už sa nedá vymazať ani upravovať a je možné len prezerať údaje. To znamená, že užívateľ môže zmazať študenta, ale zmaže len jeho záznam z aktuálneho roku. Ku všetkým aktuálnym aj minulým záznamom je možné pristúpiť z hlavného menu zvolením možnosti „Výpis doktorandov“

(viď. príloha B). na tejto stránke sa zobrazia všetky dôležité informácie o doktorandovi, informácia o tom, či schválil dátum prezentácie a výsledky z jeho dotazníku. Stránka funguje ako archív, takže každý doktorand má vždy jeden riadok pre každý rok štúdia. V budúcnosti môže byť takáto tabuľka neprehľadná, preto sa očakáva, že administrátor si vyfiltruje len tie dáta, ktoré aktuálne potrebuje zobraziť. Systém umožňuje filtrovať doktoranda v danom roku, alebo len dáta o jednom konkrétnom študentovi, prípadne zobraziť informáciu o schválení. Tieto voľby je možné kombinovať medzi sebou a taktiež uložiť do počítača vo formáte CSV.

Nevyhnutnou súčasťou užívateľského rozhrania je úprava emailových šablón, ktorá je dostupná na stránke „Email“ (viď. príloha C). Viac o spôsobe úpravy a uchovávanie emailových šablón je popísané v kapitole 5.6.

Z pohľadu doktoranda

Doktorand sa neprihlasuje štandardným spôsobom pomocou webstránky, ale len otvorením odkazu stránky (viď. príloha D), ktorý obdrží pomocou emailu a stlačením tlačidla „Potvrdiť termín prezentácie“. Potom sa mu zobrazí informácia o tom, že úspešne potvrdil termín prezentácie. na pozadí sa zapíše zmena do databázy, takže aj učiteľ vidí schválenie termínu. Okrem toho študent vidí formulár, do ktorého môže vpísať odpovede potrebné pre prezentáciu.

Menu na stránke je jednotné pre všetky role, pričom podľa role užívateľa sa určuje, ktorá položka sa mu zobrazí a ktorá nie. Systém je chránený tak, že pokiaľ sa pokúsi užívateľ pristúpiť na stránku (napríklad upravením URL adresy), na ktorú nemá prístupové právo, bude automaticky odhlásený. Systém ho presmeruje na úvodnú obrazovku s prihlásením.

5.4 Formát vstupného súboru

Aby aplikácia fungovala korektne a bez problémov dokázala spracovať vstupný súbor s údajmi o študentoch, je potrebné detailne špecifikovať, ako by mal tento súbor vyzerať. Informačný systém využíva oddeľovanie jednotlivých stĺpcov pomocou čiarky; v prípade, že sa v obsahu stĺpca nachádza čiarka, musí byť celý obsah v úvodzovkách. na prvom priadku sa musia nachádzať nasledujúce názvy stĺpcov so zachovaným názvom a poradím. Je nevyhnutné zachovať názvy stĺpcov v správnom tvare, pretože pod tými istými názvami sú uložené aj v databáze. Údaje z databázy sa odovzdávajú do šablóny, kde sa k nim pristupuje pomocou Latte makier. Dátum musí byť napísaný vo formáte rrrr-mm-dd (rok-mesiac-deň). V súbore sa nesmie nachádzať viac (ani prázdnych) stĺpcov ako tie, čo sú nižšie popísané, v opačnom prípade skončí vkladanie doktorandov do systému chybou.
Formát prvého riadku vstupného súboru:
`Datum,Priezvisko,Meno,Skolitel,Rocnik,Studium,Dlzka,Email`

Počas nahrávania doktorandov môžu nastať situácie, kedy užívateľ nechtiac nahrá toho istého doktoranda viac ako jeden krát. Systém je naprogramovaný tak, aby sa duplicitné informácie automaticky filtrovali a zabránilo sa ich vloženiu do databázy. Primárnym kľúčom v tabuľke študentov je email. Pokiaľ sa užívateľ pokúsi vložiť do tabuľky doktoranda s emailom, ktorý sa v nej už nachádza, zachytí sa výnimka. Tá sa spracuje tak, že daný riadok sa do tabuľky nevloží. na základe tohto prístupu je dôležité, aby o každom študentovi

bol uchovávaný školský email, ktorý by mohol byť použitý ako jednoznačný identifikátor doktoranda.

5.5 Formulár pre odosielanie dotazníka

Formulár pre odosielanie študentských dotazníkov bol vytvorený pomocou objektu `Nette\Application\UI\Form`. Ako už bolo spomenuté v kapitole 3.2 je dôležité zachovávať mennú konvenciu pre korektné vytvorenie formuláru. Metóda, ktorá vytvára formulár sa preto volá `createComponentQuestionnaire()` a v šablóne sa k nemu potom pristupuje pomocou makra `{control questionnaire}`. Spôsob vytvárania formuláru je možný vidieť v ukážke zdrojového kódu 5.1. Všetky polia vo formulári (okrem priestoru pre poznámky) sú nastavené ako povinné, aby ich užívateľ vzal na vedomie a nenechal ich nevyplnené. Používa sa na to metóda `setRequired(TRUE)`. Taktiež sa používa aj pravidlo pre kontrolu typu vkladaných dát, ktoré bráni vkladať užívateľovi iné ako celočíselné čísla.

```
1  public function createComponentQuestionnaire() {
2      $defaultValues = $this->database->query('SELECT * FROM questionnaire
       WHERE Link = \''.$this->sessionStudent->username.'\'')->fetch();
3      $form = new UI\Form;
4      $form->addText('pubAcceptedMA', 'Prispevky v casopise prijate - autor:')
5          ->setAttribute('class','form-control')->setRequired(TRUE)
6          ->setAttribute('placeholder', $defaultValues['pubAcceptedMA'])
7          ->addRule(Form::INTEGER, 'Hodnota musi byt cislo');
8                  ...
9      $form->addSubmit('select', 'Potvrdit');
10     $form->onSubmit[] = [$this, 'sendQuestionnaireSucceeded'];
11     return $form;
12 }
```

Listing 5.1: Vytvorenie formuláru

5.6 Úprava emailov a spôsob odosielania

Vďaka šablónovaciemu systému Latte môže administrátor dynamicky meniť informácie v emaile podľa toho, komu bol daný email odoslaný. pre vkladanie jednotlivých údajov z databázy sa používajú predpripravené makrá.

Sú nasledujúce:

- {$Meno} - krstné meno
- {$Priezvisko} - priezvisko
- {$Email} - email študenta
- {$Skolitel} - meno školiteľa
- {$Rocnik} - rok štúdia
- {$Studium} - typ štúdia (kombinované/prezenčné)

- {$Dlzka} - dĺžka prezentácie
- {$Link} - URL adresa študenta
- {$Datum|date} - dátum prezentácie

Užívateľ ich vidí v prehľadnej tabuľke priamo na stránke aj s vysvetlením ich funkcie. na ukážke kódu 5.2 je možné vidieť ukážku, ako vyzerá obsah `Latte` súboru, ktorý slúži ako šablóna emailu.

Každý email je uložený na serveri v podobe `.latte` šablóny, nachádzajúcej sa v zložke `app/www/EmailTemplates`. pre úpravu sa používa online textový editor `CodeMirror`, ktorý slúži na úpravu šablóny v reálnom čase priamo na stránke užívateľom.

```
1 Dobry den p. {$Meno} {$Priezvisko}
2
3 chcel by som Vam oznamit, ze prezentacia Vasej dizertacnej prace
4 v~akademickom roku 2016/2017 sa uskutocni dna~{$Datum|date}. Prosim o
      potvrdenie oboznamenia sa s terminom prezentacie kliknutim na~
      potvrdzovacie tlacitko na~stranke.
5
6 {$Link}
```

Listing 5.2: Ukážka Latte šablóny

Ako už bolo spomenuté v kapitole 3.2 na odosielanie emailov sa používajú triedy pripravené vo frameworku Nette. Konkrétne sa jedná o triedu `Nette\Mail\Message`, ktorá slúži na vytvorenie správy a triedu `Nette\Mail\SendMailer` pre jej odoslanie. Podľa požiadaviek zadávateľa má byť telo správy v textovej forme a nie v HTML, na to sa používa metóda `setBody()`. Spôsob vytvárania emailu je možné vidieť na ukážke 5.3.

```
1 $mail = new Message;
2 $mail->setFrom($this->context->parameters["email_from"], $this->context->
      parameters["email_name"])
3     ->addTo($params['Email'])
4     ->setSubject($this->context->parameters["email_subject"])
5     ->setBody($latte->renderToString($this->context->parameters["dir"].$send,
       $params)); //predanie premennych sablone
6 $this->mailer->send($mail); //odoslanie emailu
```

Listing 5.3: Tvorba emailu

5.7 Tvorba URL adries užívateľov

pre každého študenta je potrebné vygenerovať unikátnu URL adresu, pomocou ktorej sa bude prihlasovať do systému. Hneď počas vkladania študenta do databázy sa vytvorí unikátny číselný reťazec, ktorý sa spolu so študentom zapíše do databázy. pri odosielaní emailu sa potom vytvorí celá URL adresa, ktorá sa vloží na miesto Latte makra `{$Link}`. Daná adresa sa počas otvárania spracováva na základe nastaveného routovania, ktoré je v nasledujúcom tvare: `<presenter>/<action>[/<id>]`.

Kapitola 6

Testovanie

Testovanie aplikácie prebiehalo iteratívne počas celého vývoja aplikácie. Aktuálny stav a jeho možné zlepšenia boli pravidelne konzultované s vedúcim práce. na základe týchto diskusií boli postupne pridávané ďalšie funkcie. Vždy po pridaní novej funkcionality bolo kontrolované správne chovanie aplikácie a vo väčších celkoch to bolo testované spolu s vedúcim práce na konzultáciách.

Našim cieľom na začiatku akademického roku bolo nasadenie aplikácie už počas tohtoročných príprav prezentácií. Tento cieľ sa nakoniec nepodarilo naplniť, pretože pôvodný termín začatia plánovania prezentácií sa neočakávane posunul na skorší termín. Aplikácia v tej dobe už síce spĺňala základnú funkcionalitu, ale nebola dostatočne otestovaná. Reálne nasadenie preto nebolo bezpečné. po dohode s vedúcim práce sme sa nakoniec rozhodli, že odložíme nasadenie až na nasledujúci rok, aby sme predišli komplikáciám v prípade, že by nastala nejaká neočakávaná chyba.

Testovanie prebiehalo na reálnych dátach s tým, že emaily sa neodosielali študentom, ale mne. Všetky emaily sa úspešne rozposlali a po kliknutí na tlačidlo na stránke sa zmeny prejavili v databáze a teda aj učiteľ videl potvrdenie v informačnom systéme.

Kapitola 7

Záver

V rámci mojej bakalárskej práce som sa naučil pracovať s PHP frameworkom Nette, na ktorom je postavená celá aplikácia. Cieľom práce bolo navrhnúť a naprogramovať informačný systém, ktorý bude slúžiť na administráciu prihlasovania sa na študentské prezentácie. Výsledné riešenie je robené formou webovej aplikácie, takže pre prístup k dátam stačia v podstate len prihlasovacie údaje a prístup na internet. Administrátor môže jednoducho vložiť údaje o študentoch do informačného systému, vytvoriť návrh emailu a ten im hromadne rozposlať. V prehľadných tabuľkách môže sledovať aktuálny stav schvaľovania potvrdení o oboznámení s termínom prezentácie a prípadne ich urgovať opakovanými emailami. po vyplnení dotazníka doktorandom sa zmeny premietnu do tabuľky zobrazujúcej odpovede. Administrátor môže tiež odstraňovať aktuálnych doktorandov zo systému. Všetky dáta z tabuliek je možné vyexportovať vo formáte CSV a zdieľať ich s inými pracovníkmi fakulty. Všetky dáta o doktorandoch sa uchovávajú a je teda možné porovnávať výsledky doktorandov s predchádzajúcimi rokmi.

Hlavný prínos tejto práce pozostáva z vytvorenia informačného systému, ktorý šetrí čas administrátorovi a uľahčuje organizáciu prezentácií. Už nie je potrebné manuálne odosielať emaily a vyhodnocovať tie prijaté, ale všetko sa prehľadne zobrazuje na webovej stránke. Aj pre doktorandov je jednoduchšie vyplniť jeden webový formulár, ako si dávať pozor, či odpovedali na email v požadovanom formáte.

Informačný systém je hotový a pripravený na použitie pri organizácií prezentácií v nasledujúcom školskom roku. Momentálne nie je plánované žiadne rozšírenie do budúcnosti, pretože spĺňa všetky požiadavky zadávateľa, avšak po dohode je možné pridať ďalšie funkcie.

Literatúra

[1] Flanagan, D.: *JavaScript: The Definitive Guide (6th ed.)*. O'Reilly & Associates, 2011, ISBN 978-0-596-80552-4.

[2] Gutmans, A.; Bakken, S. S.; Rethans, D.; aj.: *Mistrovství v PHP 5*. CP Books, 2005, ISBN 80-251-0799-X.

[3] Haverbeke, M.: *CodeMirror*. [Online; navštívené 3.4.2017].
URL https://codemirror.net

[4] Janda, P.: *Datagrid | Ublaboo*. [Online; navštívené 14.03.2017].
URL https://ublaboo.org/datagrid/

[5] Malý, M.: *REST: architektura pro webové API*. [Online; navštívené 1.5.2017].
URL https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

[6] Nette, F.: *Formuláře*. [Online; navštívené 10.5.2017].
URL https://doc.nette.org/cs/2.4/forms

[7] Nette, F.: *Latte*. [Online; navštívené 6.4.2017].
URL https://latte.nette.org/cs/

[8] Nette, F.: *MVC aplikace & presentery*. [Online; navštívené 6.4.2017].
URL https://doc.nette.org/cs/2.4/presenters

[9] Nette, F.: *Odesílání e-mailů*. [Online; navštívené 7.4.2017].
URL https://doc.nette.org/cs/2.4/mailing

[10] Nette, F.: *Přihlašování & oprávnění uživatelů*. [Online; navštívené 7.4.2017].
URL https://doc.nette.org/cs/2.4/access-control

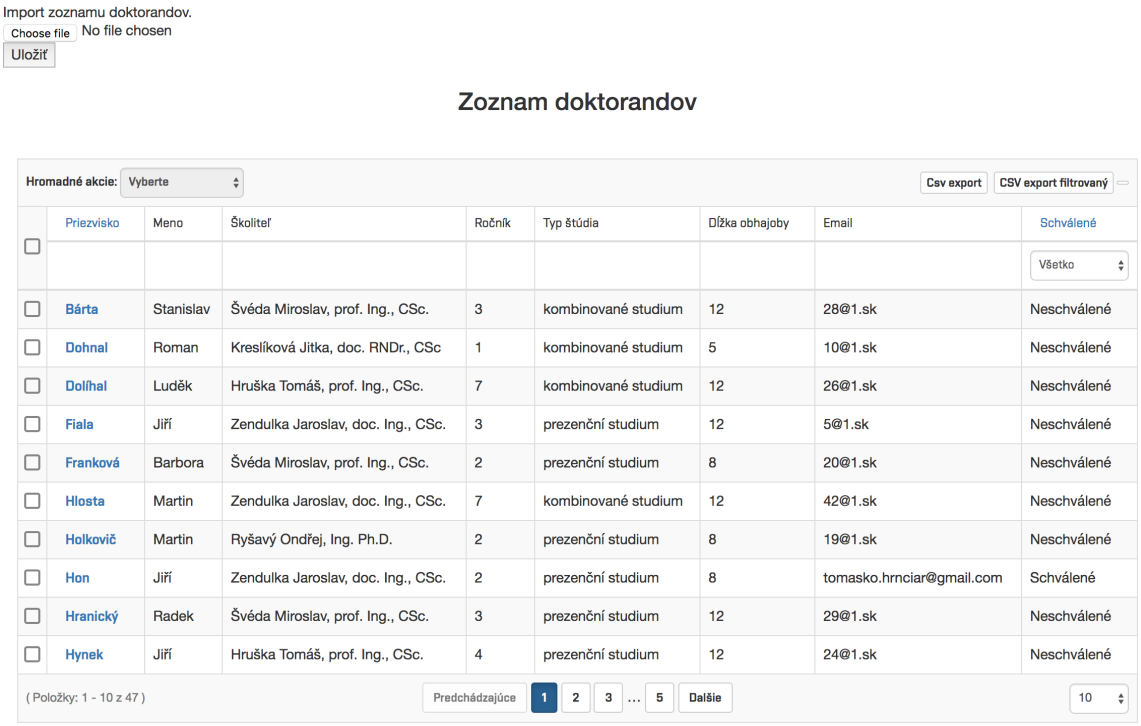
[11] Nette, F.: *Routování URL*. [Online; navštívené 6.4.2017].
URL https://doc.nette.org/cs/2.4/routing

[12] Skvorc, B.: *The Best PHP Framework for 2015: SitePoint Survey Results*. [Online; navštívené 14.03.2017].
URL https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

Prílohy

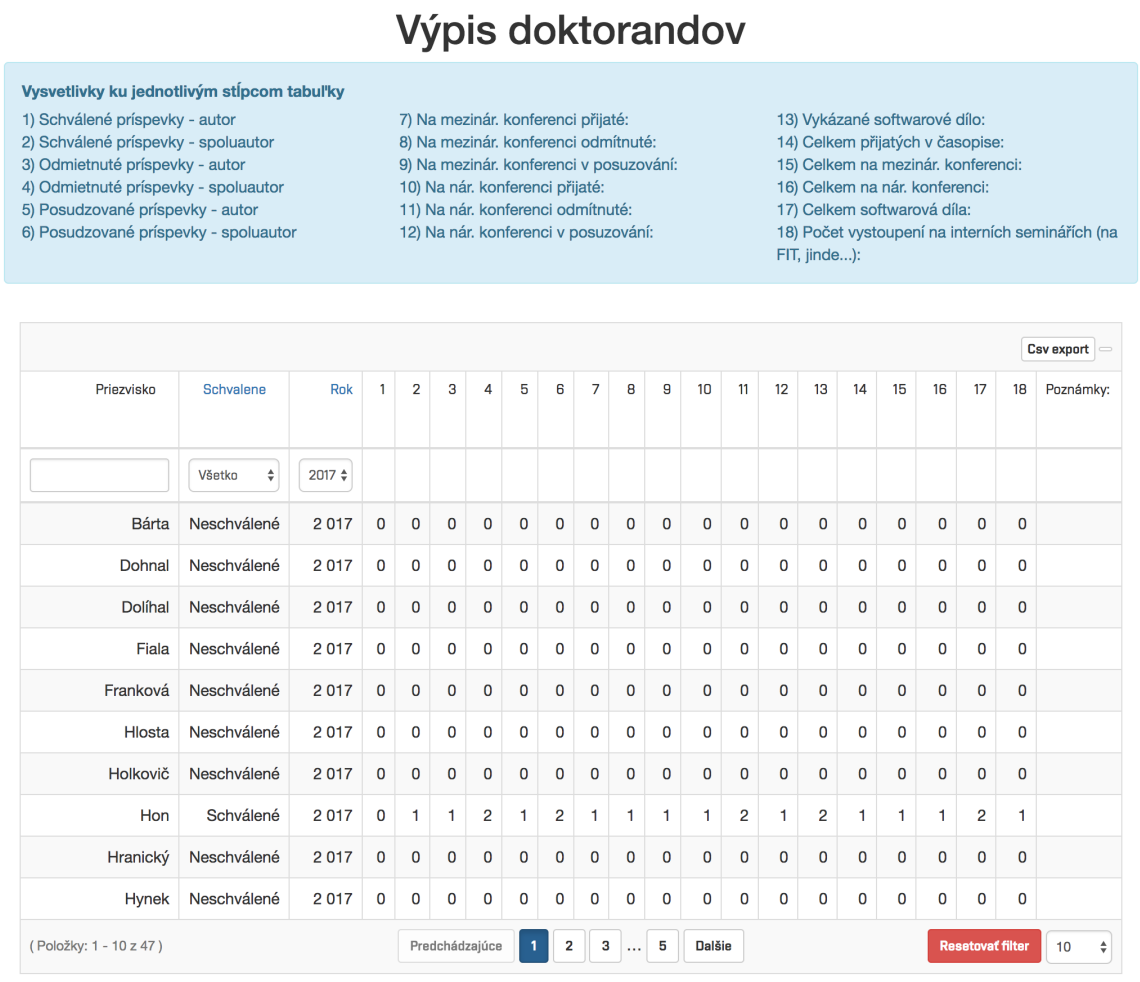
Príloha A

Užívateľské rozhranie - domovská stránka



Príloha B

Užívateľské rozhranie - výpis študentov



Výpis doktorandov

Vysvetlivky ku jednotlivým stĺpcom tabuľky

1) Schválené príspevky - autor
2) Schválené príspevky - spoluautor
3) Odmietnuté príspevky - autor
4) Odmietnuté príspevky - spoluautor
5) Posudzované príspevky - autor
6) Posudzované príspevky - spoluautor
7) Na mezinár. konferenci přijaté:
8) Na mezinár. konferenci odmítnuté:
9) Na mezinár. konferenci v posuzování:
10) Na nár. konferenci přijaté:
11) Na nár. konferenci odmítnuté:
12) Na nár. konferenci v posuzování:
13) Vykázané softwarové dílo:
14) Celkem přijatých v časopise:
15) Celkem na mezinár. konferenci:
16) Celkem na nár. konferenci:
17) Celkem softwarová díla:
18) Počet vystoupení na interních seminářích (na FIT, jinde...):

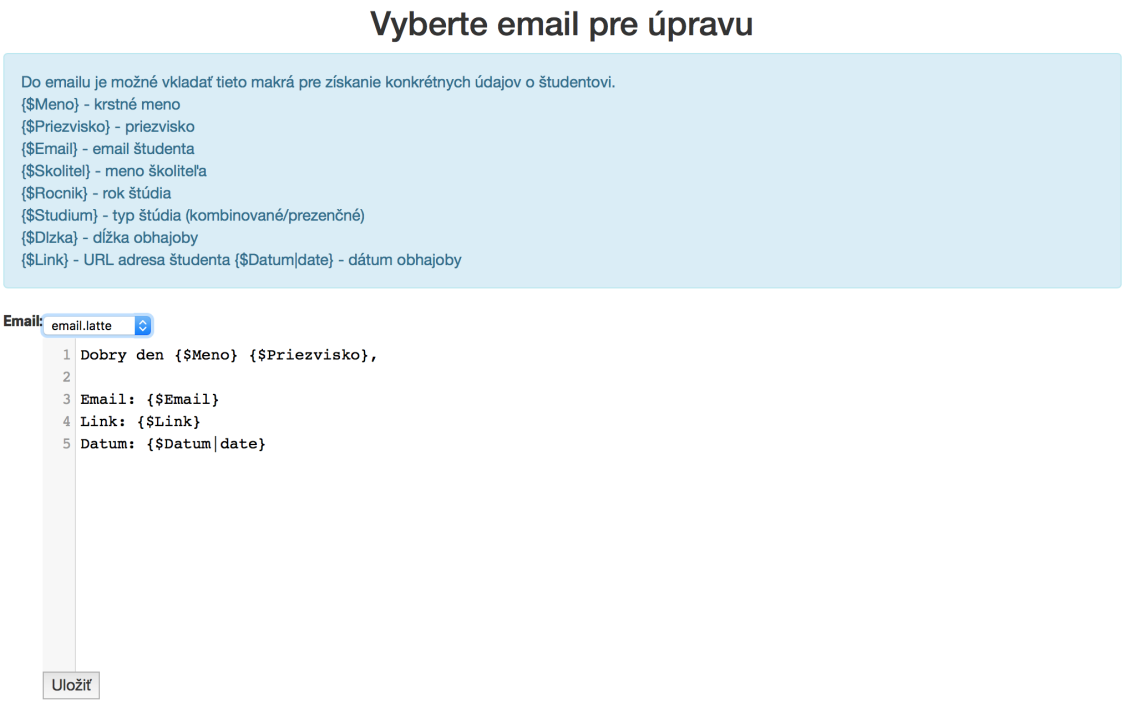
Csv export

Priezvisko	Schvalene	Rok	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Poznámky:
	Všetko	2017																			
Bárta	Neschválené	2 017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Dohnal	Neschválené	2 017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Dolíhal	Neschválené	2 017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Fiala	Neschválené	2 017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Franková	Neschválené	2 017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Hlosta	Neschválené	2 017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Holkovič	Neschválené	2 017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Hon	Schválené	2 017	0	1	1	2	1	2	1	1	1	1	2	1	2	1	1	1	2	1	
Hranický	Neschválené	2 017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Hynek	Neschválené	2 017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

(Položky: 1 - 10 z 47) Predchádzajúce 1 2 3 … 5 Ďalšie Resetovať filter 10

Príloha C

Užívateľské rozhranie - úprava emailu



Príloha D

Užívateľské rozhranie - dotazník doktoranda