FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

| | |
|---|---|
| **Název:** | Portál pro správu semestrálních prací |
| **Student:** | Václav Dvořák |
| **Vedoucí:** | Ing. Jiří Chludil |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Webové a softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | Do konce letního semestru 2018/19 |

## Pokyny pro vypracování

1. Analyzujte nástroje a systémy pro odevzdávání semestrálních prací na ČVUT FIT.
2. Analyzujte potřeby vyučujících a studentů vybraných předmětů BI-MGA, BI-PGA a BI-ZNF z hlediska odevzdávání semestrálních prací.
3. Pomocí technik softwarového inženýrství navrhněte strukturu učitelské a studentské části aplikace. Zaměřte se především na
 - kontrolu struktury a formátu odevzdávaných souborů (multimediální data, zdrojové kódy)
 - validace zdrojových kódů (v případě Nette i pomocí testů)
 - publikace odevzdaných prací (multimediální materiály, projekty v Nette)
4. Navrhněte přívětivé uživatelské rozhraní obou částí.
5. Použijte vhodné části Nette frameworku. Jako databázový systém použijte PostgreSQL.
6. Na základě návrhu naimplementujte aplikaci s využitím continuous integration.
7. Hotové řešení podrobte vhodným testům.

## Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 22. listopadu 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

# Portál pro správu semestrálních prací

***Václav Dvořák***

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Chludil

13. května 2018

# Poděkování

Tímto bych chtěl poděkovat panu Ing. Jiřímu Chludilovi za vedení této práce, jeho ochotu a čas strávený při konzultacích. Taktéž děkuji své rodině za podporu během celého studia.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2018 ……

### Odkaz na tuto práci

Dvořák, Václav. *Portál pro správu semestrálních prací.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

# Abstrakt

Bakalářská práce se zabývá návrhem a implementací webové aplikace pro správu semestrálních prací předmětů BI-ZNF, BI-MGA a BI-PGA. Cílem práce je na základě požadavků zadavatele navrhnout a implementovat webovou aplikaci umožňující všechny úkony spjaté s životním cyklem semestrálních prací ze strany vyučujícího i studenta. Aplikace se zaměřuje především na výběr tématu semestrální práce, její odevzdání, ohodnocení a vytvoření galerií z proběhlých kurzů. Výsledný systém je implementován na Nette frameworku s využitím PostgreSQL databáze a Continuous Integration pomocí verzovacího systému Git. Aplikace umožňuje automatické načítání dat z informačního systému KOS pomocí REST API.

**Klíčová slova** Webová aplikace, Nette framework, Conitnuous Integration

# Abstract

The bachelor thesis is about design and implementation of web application built for management of term work with targeted support for BI-ZNF, BI-MGA and BI-PGA. The main ambition is to design and implement web application that allows all possible tasks that come with life cycle of term work from students and teachers perspective. Application aims mainly for selection of theme for the work, its submission, evaluation and creation of galleries based on past courses. Final build is composed of Nette framework and PostgreSQL as a database service. Git version control system is used to handle Continuous Integration. The application enables automatic retrieval of data from the KOS information system using the REST API.

**Keywords** Web application, Nette framework, Continuous Integration

# Obsah

# Seznam obrázků

# Seznam tabulek

# Úvod

Semestrální práce jsou v mnoha předmětech stěžejní či jedinou částí hodnocení studenta. Tématem této práce je tvorba webové aplikace, portálu pro správu semestrálních prací předmětů BI-ZNF, BI-MGA a BI-PGA. Tento portál by měl usnadnit práci jak vyučujícím, tak studentům. V současné době jsou semestrální práce těchto předmětů řešeny přes aplikaci Edux, která slouží primárně jako úložiště studijních materiálů a hodnocení studentů. Edux však bude v několika následujících měsících stažen z provozu. Aplikace, která bude výstupem této bakalářské práce by měla Edux u těchto předmětů nahradit.

V první části této práce se mimo jiné budu zabývat analýzou současných systémů pro správu semestrálních prací na ČVUT FIT. Z této analýzy vyberu zajímavé a užitečné funkce jednotlivých aplikací a poučím se z nalezených chyb a nedostatků. Vybrané funkce implementuji i do mnou vyvíjené aplikace.

Aplikace bude implementována na Nette frameworku. Předmět základy Nette fremeworku patří právě mezi ty, pro které je tato aplikace vyvíjena. Já jsem se poprvé s tímto frameworkem setkal právě díky tomuto předmětu. Nyní se mu věnuji i v zaměstnání, a proto pro mě je aplikace implementována pomocí tohoto frameworku jasnou volbou i pro bakalářskou práci.

Systém bude rozdělen na jednotlivé moduly pro administrátory, vyučující a studenty. Jednotlivé předměty a jejich kurzy budou importovány z informačního systému KOS pomocí dostupného KOSapi. Spolu s kurzy budou importováni i všichni uživatelé, které bude možné nadále manuálně upravovat přes administrátorské rozhraní aplikace. Při vývoji bude kladen důraz především na kontrolu odevzdávaných prací a jejich následnou publikaci v galeriích.

Na závěr bude aplikace podrobená testování jak automatizovanému, tak i uživatelskému. Ostrý provoz aplikace se plánuje na zimní semestr 2018/2019, kdy bude aplikace poprvé spuštěna pro předměty BI-PGA a BI-MGA. Pro předmět BI-ZNF bude spuštěna v letním semestru téhož ročníku. Její vývoj tedy nebude ukončen odevzdáním této práce, ale bude i nadále udržována.

Kapitola 1

# Cíl práce

Cílem této práce je vytvoření webové aplikace pro správu semestrálních prací třech předmětů vyučovaných na ČVUT FIT. Jedná se o předměty programování grafických aplikací, multimediální a grafické aplikace a základy Nette frameworku.

Prvním cílem této práce je analýza nástrojů a systémů, které se na této fakultě využívají pro správu semestrálních prací u jiných předmětů. Dále je potřeba analyzovat potřeby vyučujících vybraných předmětů z hlediska odevzdávání semestrálních prací tak, aby systém umožňoval všechny potřebné funkce. Na základě této analýzy bude navržena struktura učitelské a studentské části aplikace.

V části návrhu bude kladen důraz na kontrolu struktury a formátu odevzdávaných souborů semestrálních prací. Je potřeba počítat s tím, že v rámci jedné práce může být odevzdáváno více souborů. To lze realizovat možností nahrání více souborů zvlášť, či jedním archivem, se kterým bude aplikace umět dále pracovat. Dále bude navrženo automatické testování odevzdávaných zdrojových kódů a jejich validity. Odevzdané práce budou publikovány v galeriích zpřístupněných všem přihlášeným uživatelům.

Na základě návrhu bude aplikace implementována za využití vhodných částí Nette frameworku. Jako databázový systém bude použit PostgreSQL. Celý proces implementace bude doprovázen systémem průběžné integrace (Continuous Integration), který bude automaticky integrovat nové části zdrojového kódu na server. Výsledná aplikace bude podrobena vhodným testům.

Výsledkem celé práce by měla být aplikace, která ulehčí správu semestrálních prací jak vyučujícím tak i studentům. Aplikace bude umožňovat úkony spjaté s celým životním cyklem semestrální práce od zvolení tématu až po jeho odevzdání a ohodnocení. Výsledná aplikace by měla nahradit doposud používaný systém.

Kapitola 2

# Analýza

## 2.1 Analýza systémů pro odevzdávání semestrálních prací na ČVUT FIT

V současné době se na ČVUT FIT používají čtyři webové aplikace, které se alespoň z části zabývají správou semestrálních prací a domácích úkolů. Některé aplikace nabízejí navíc například online zápočtové testy, nebo slouží spíše jako úložiště studijních materiálů.

Pouze jedna z následujících aplikací je zaměřena na konkrétní předmět. Ostatní aplikace jsou určeny pro větší okruh předmětů, či nemají žádné zaměření a lze je použít pro jakýkoliv předmět. V následujících kapitolách tyto aplikace porovnám dle mnou stanovené hodnotící metodiky.

### 2.1.1 Návrh hodnotící metodiky

Každou z aplikací budu hodnotit podle níže specifikovaných kriterií, přičemž součet jednotlivých bodů určí celkové pořadí aplikací. Maximální celkový počet bodů, který může aplikace získat, je 15.

**Možnost výběru a odevzdání semestrální práce**

- neumožňuje výběr ani odevzdání — 0 bodů
- umožňuje pouze výběr nebo odevzdání — 1 bod
- umožňuje výběr i odevzdání semestrální práce — 2 body

**Možnosti hodnocení prací**

- neumožňuje hodnocení — 0 bodů
- k dispozici je pouze jedno hodnotící pole — 1 bod

- k dispozici jsou dvě pole, jedno pro body, druhé pro komentář — 2 body
- lze definovat více kritérií a přidělovat jim body — 3 body
- automatické testování — +1 bod

**Galerie odevzdaných prací**

- neumožňuje vytvářet galerie — 0 bodů
- lze vytvářet galerie — 1 bod
- lze upravovat galerie po vytvoření — +1 bod
- lze definovat vlastní strukturu galerie — +1 bod

**Uživatelské rozhraní aplikace** Uživatelské rozhraní budu hodnotit podle třech níže zvolených bodů Nielsenovy heruistiky. Aplikaci udělím 0 bodů při nesplnění, 1 bod při částečném splnění, či 2 body při úplném splnění daného kritéria. [1]

- Recognition rather than recall – *„Uživatel by měl být při používání systému co nejméně kognitivně zatížen. To zajistíme tím, že bude systém nabízet pouze volby, které lze vybrat a ostatní skryje, nebo vizuálně zneplatní. Dále je vhodné použití drobečkové navigace, stránkování, nebo zvýraznění pozice ve stromové struktuře, aby uživatel ihned viděl jeho aktuální pozici v systému.“*
- Aesthetic and minimalist design – *„Systém by měl zobrazovat co nejméně informací a voleb, tak aby práce v něm byla co nejrychlejší, přehledná, a uživatel musel co nejméně přemýšlet. Položky a volby, které nejsou často využívány, není vhodné umísťovat přímo na běžně používané obrazovky.“*
- Help and documentation – *„Systém musí poskytovat nápovědu přesně tam, kde jí uživatel očekává a v situacích, kdy je nejvíce potřeba. Například vyplňování formulářů, zakládání nových projektů apod. Nápověda by měla být buď kontextová (přímo u daného prvku), nebo globální s možností vyhledávání.“*

### 2.1.2 ProgTest

Systém ProgTest [2] je nejznámější aplikací této fakulty sloužící především pro odevzdávání domácích úkolů a semestrálních prací mnoha předmětů zaměřených na programování.


Obrázek 2.1: Výsledek hodnocení domácího úkolu v systému ProgTest

#### 2.1.2.1 Možnost výběru a odevzdání semestrální práce

Student si může zvolit z různých témat semestrálních prací či úkolů a jsou mu započítány body pouze za téma, či témata, která odevzdá.

- umožňuje výběr i odevzdání semestrální práce — 2 body

#### 2.1.2.2 Možnosti hodnocení prací

Hodnocení prací u většiny předmětů probíhá zcela automaticky. Student získává do několika vteřin bodové ohodnocení jeho práce s přehledem úspěšnosti v jednotlivých testech (2.1).

- lze definovat více kritérií a přidělovat jim body — 3 body
- automatické testování — +1 bod

#### 2.1.2.3 Galerie odevzdaných prací

Systém nepodporuje vytváření galerií již zpracovaných řešení. Tato funkce by však byla vzhledem k povaze předmětů a přísnému zákazu kopírování odevzdaných řešení spíše nežádoucí.

- neumožňuje vytvářet galerie — 0 bodů

#### 2.1.2.4 Uživatelské rozhraní aplikace

- Recognition rather than recall — 2 body

  V aplikaci je použita drobečkova navigace a všechny nedostupné volby jsou skryty.

- Aesthetic and minimalist design — 2 body

  Aplikace zobrazuje pouze nezbytné informace a volby pro stránku, na které se uživatel právě nachází.

- Help and documentation — 1 bod

  Aplikace poskytuje globální nápovědu bez vyhledávání.

#### 2.1.2.5 Shrnutí hodnocení aplikace Progtest

Na aplikaci ProgTest se mi nejvíce líbí automatické testování, které je doplněno o kontrolu duplicity kódu skrze všechny proběhlé kurzy daného předmětu. Aplikace sice díky jejímu designu působí trochu zastarale, nicméně z velké části splňuje vybraná kritéria Nielsonovy heuristiky.

Tabulka 2.1: Hodnocení aplikace ProgTest

| Kritérium | body |
|---|---|
| Možnosti výběru a odevzdání semestrální práce | 2 |
| Možnosti hodnocení prací | 4 |
| Galerie odevzdaných prací | 0 |
| Uživatelské rozhraní aplikace | 5 |
| | 11 |

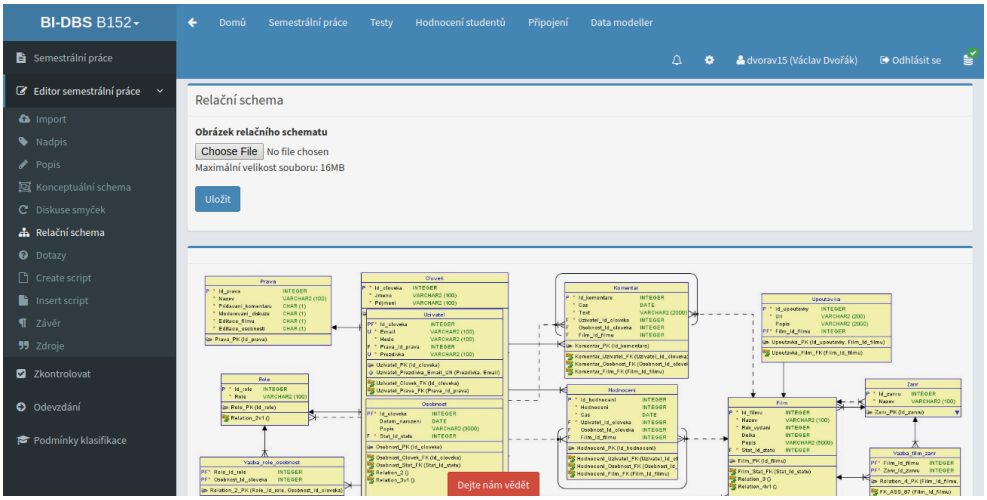
### 2.1.3 Portál BI-DBS

Portál BI-DBS [3] je určen pouze pro tento daný předmět vyučující databázové systémy. Díky tomu, že se zaměřuje pouze na problematiku tohoto předmětu, nabízí plno specifických nástrojů.

#### 2.1.3.1 Možnost výběru a odevzdání semestrální práce

Vzhledem k tomu, že studenti tohoto předmětu mají totožné zadání a téma semestrální práce si vybírají sami, neprobíhá zde žádný výběr. Aplikace disponuje editorem semestrální práce (2.2), pomocí kterého lze celou práci sestavit přímo v systému, zkontrolovat a následně odevzdat.

- umožňuje pouze odevzdání semestrální práce — 1 bod




Obrázek 2.2: Editor semestrální práce na portálu BI-DBS

#### 2.1.3.2 Možnosti hodnocení prací

Hodnocení semestrálních prací probíhá automaticky. V průběhu semestru je zpravidla několik iterací, které se kontrolují v určitém termínu. Případné neshody v hodnocení jsou osobně konzultovány se cvičícím.

- lze definovat více kritérií a přidělovat jim body — 3 body
- automatické testování — +1 bod

#### 2.1.3.3 Galerie odevzdaných prací

Portál BI-DBS, stejně jako aplikace ProgTest, neumožňuje vytvářet galerie odevzdaných prací. Vzhledem k tomu, že si každý student volí své vlastní téma práce (nikoliv zadání), tak by galerie nenapomáhali k opisování, pouze by studenty inspirovali.

- neumožňuje vytvářet galerie — 0 bodů

#### 2.1.3.4 Uživatelské rozhraní aplikace

- Recognition rather than recall — 2 body

  Aplikace zobrazuje pouze dostupné funkce. Aktuální pozice v aplikaci je znázorněna zvýrazněním položky v menu.

- Aesthetic and minimalist design — 2 body

  Aplikace zobrazuje na stránkách pouze potřebné nástroje a informace. Nezatěžuje uživatele zbytečnými prvky.

- Help and documentation — 1 bod

  Aplikace poskytuje pouze částečnou kontextovou nápovědu.

#### 2.1.3.5 Shrtnutí hodnocení portálu BI-DBS

Na portálu BI-DBS mě nejvíce zaujal online editor semestrálních prací, díky kterému lze celou práci sestavit přímo v aplikaci a následně odevzdat. Dále se mi líbí podpora iterací, ve kterých se kontroluje aktuální stav práce, při čemž se provádějí různé automatické testy. V aplikaci mi chybí galerie odevzdaných prací, v tomto případě by to mohla být pro studenty zajímavá inspirace.

Tabulka 2.2: Hodnocení portálu BI-DBS

| Kritérium | body |
|---|---|
| Možnosti výběru a odevzdání semestrální práce | 1 |
| Možnosti hodnocení prací | 4 |
| Galerie odevzdaných prací | 0 |
| Uživatelské rozhraní aplikace | 5 |
| | 10 |

### 2.1.4 Moodle

Aplikace Moodle [4] je na ČVUT FIT známá především jako úložiště studijních materiálů a prostředí pro psaní online testů.

#### 2.1.4.1 Možnost výběru a odevzdání semestrální práce

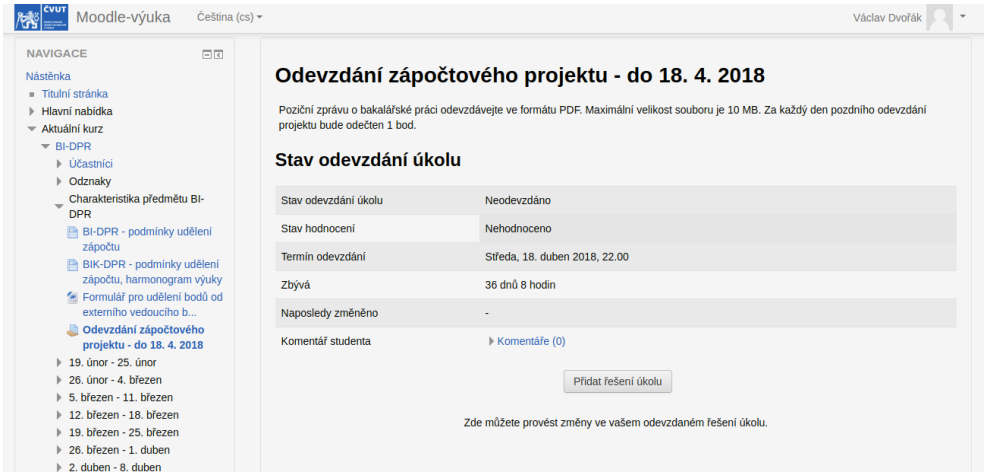
Systém neposkytuje žádnou možnost výběru tématu semestrální práce. Student u předmětu vidí pouze obecný název úkolu a odkaz na něj. Na stránce s úkolem lze odevzdat jeden soubor.

- umožňuje pouze odevzdání semestrální práce — 1 bod

#### 2.1.4.2 Možnosti hodnocení prací

Moodle neposkytuje automatické testování semestrálních prací, ani kontrolu struktury odevzdaného řešení. Hodnocení semestrálních prací v tomto systému probíhá pouze za přítomnosti vyučujícího. Bodové ohodnocení je většinou vkládáno do komentáře úkolu, což je jediné dostupné hodnotící pole v rámci úkolu.

- k dispozici je pouze jedno hodnotící pole — 1 bod




Obrázek 2.3: Stránka s odevzdáním projektu v aplikaci Moodle

#### 2.1.4.3 Galerie odevzdaných prací

Moodle stejně jako všechny předešlé aplikace nepodporuje vytváření galerií.

- neumožňuje vytvářet galerie — 0 bodů

#### 2.1.4.4 Uživatelské rozhraní aplikace

- Recognition rather than recall — 2 body

  Pozice v aplikaci je zobrazena pomocí stromové struktury v levém bočním panelu. Uživatelovi jsou nabídnuty pouze ty funkce, které pro něho mají význam.

- Aesthetic and minimalist design — 0 bodů

  Aplikace je nepřehledná. I když jsou uživatelovi nabídnuty pouze prvky, které může využít, je jimi zbytečně zahlcen. Uživateli může trvat delší dobu, než se na stránce ve velkém množství prvků zorientuje, a najde cestu k tomu, co hledá.

- Help and documentation — 0 bodů

  Aplikace neposkytuje globální nápovědu. Pouze zřídka poskytuje nápovědu kontextovou.

#### 2.1.4.5 Shrnutí hodnocení aplikace Moodle

Aplikace Moodle se dle mého názoru na správu semestrálních prací nehodí, což utvrzuje fakt, že aplikace získala pouze 4 z 15 bodů (2.3). Moodle poskytuje velice omezené možnosti a spoustu uživatelů odradí svým nepřehledným rozhraním.

Tabulka 2.3: Hodnocení aplikace Moodle

| Kritérium | body |
|---|---|
| Možnosti výběru a odevzdání semestrální práce | 1 |
| Možnosti hodnocení prací | 1 |
| Galerie odevzdaných prací | 0 |
| Uživatelské rozhraní aplikace | 2 |
| | 4 |

### 2.1.5 Edux

Systém Edux [5] je podobně jako Moodle známý spíše pro publikování studijních materiálů a informací o jednotlivých předmětech. V některých předmětech se využívá i pro odevzdávání semestrálních prací. Předměty BI-ZNF, BI-PGA a BI-MGA využívají právě Edux.

#### 2.1.5.1 Možnost výběru a odevzdání semestrální práce

Výběr semestrálních prací v systému Edux funguje pouze na základě vypsaných témat na jedné podstránce předmětu (2.4). Student se musí osobně domluvit s vyučujícím, jaké téma si zvolil. Z tohoto důvodu mu tuto funkci neuznávám. Odevzdávání semestrální práce probíhá formou dokumentace — wiki stránky — kterou si student v Eduxu může vytvořit.

- umožňuje pouze odevzdání semestrální práce — 1 bod

#### 2.1.5.2 Možnosti hodnocení prací

Hodnocení semestrálních prací, stejně tak jako v Moodlu, probíhá manuálně. Bodové ohodnocení může být rozděleno do více kritérií, kterým může vyučující přidělit rozdílné body.

- lze definovat více kritérií a přidělovat jim body — 3 body

#### 2.1.5.3 Galerie odevzdaných prací

Edux jako jediná ze zmíněných aplikací umožňuje automatické generování galerií odevzdaných prací. Tyto galerie se generují přímo z odevzdaných souborů,


Obrázek 2.4: Seznam témat semestrálních prací v aplikaci Edux

které musí mít požadovanou strukturu, aby došlo k úspěšnému vytvoření galerie. Struktura galerií je pevně daná a galerie nelze po vygenerování upravovat.

- umožňuje vytvářet galerie — 1 bod

#### 2.1.5.4 Uživatelské rozhraní aplikace

- Recognition rather than recall — 2 body

  Pozice v aplikaci je zobrazena pomocí drobečkovy navigace. Uživatelovi jsou rovněž zobrazeny jen jemu povolené a opodstatněné funkce.

- Aesthetic and minimalist design — 2 body

  Aplikace je přehledná. V aplikaci se vyskytují pouze nezbytné prvky, které s danou stránkou souvisí. Uživatel není zatěžován nadměrným počtem prvků na stránce.

- Help and documentation — 0 bodů

  Aplikace neposkytuje kontextovou ani globální nápovědu.

#### 2.1.5.5 Shrtnutí hodnocení aplikace Edux

Aplikace Edux není primárně určena pro správu semestrálních prací. Nicméně i přesto jako jediná nabízí galerii odevzdaných řešení a vcelku komplexní prostor pro hodnocení studentů.

Tabulka 2.4: Hodnocení aplikace Edux

| Kritérium | body |
|---|---|
| Možnosti výběru a odevzdání semestrální práce | 1 |
| Možnosti hodnocení prací | 3 |
| Galerie odevzdaných prací | 1 |
| Uživatelské rozhraní aplikace | 4 |
| | 9 |

### 2.1.6 Souhrn hodnocení aplikací

V tabulce 2.5 uvádím celkové pořadí aplikací podle mnou stanovených kriterií. Jako nejlepší aplikace pro správu semestrálních prací se umístil ProgTest, který poskytuje velice komplexní automatické hodnocení odevzdaných řešení. Nejméně vhodnou aplikací je podle těchto kritérií Moodle, který je ve správě semestrálních prací nejvíce omezen a ve své podstatě poskytuje pouze nahrávání souborů.

Jednotlivé aplikace nabízejí mnohem více funkcí než jen správu semestrálních prací a tak by se při volbě jiných kritérií mohlo pořadí zdaleka změnit. Za zmínku stojí například portál BI-DBS, který nabízí různé nástroje pro práci s databázemi. Všechny aplikace, až na Edux, umožňují vytváření online testů, které se využívají například u zkoušek.

U všech aplikací mi chybí funkce pro jasnou volbu tématu, kdy by si student zvolil jedno z dostupných témat přímo v aplikaci, které by musel následně vypracovat.

Edux jako jediná z aplikací nabízí publikování odevzdaných řešení v galeriích. Struktura těchto galerií se však nedá změnit. V mé práci bych toto chtěl umožnit.

Tabulka 2.5: Výsledky hodnocení aplikací

| | Možnosti výběru a odevzdání práce | Možnosti hodnocení prací | Galerie odevzdaných prací | Uživatelské rozhraní aplikace | **Celkem** |
|---|---|---|---|---|---|
| **1. ProgTest** | 2 | 4 | 0 | 5 | **11** |
| **2. BI-DBS** | 1 | 4 | 0 | 5 | **10** |
| **3. Edux** | 1 | 3 | 1 | 4 | **9** |
| **4. Moodle** | 1 | 1 | 0 | 2 | **4** |

## 2.2 Funkční požadavky

Funkční požadavky specifikují jednotlivé funkcionality, které musí nová aplikace podporovat. Pro tuto aplikaci jsou stanoveny následující funkční požadavky:

**FP1 - Správa uživatelů** Administrátor bude moci spravovat veškeré uživatele aplikace. Bude je moci přidávat, upravovat a měnit jejich role. Všichni uživatelé budou importováni z informačního systému KOS. Správa uživatelů bude tedy spíše sloužit pouze pro dodatečné úpravy.

**FP2 - Správa předmětů** Systém bude umožňovat kompletní správu předmětů a jejich kurzů. Předměty, jejich kurzy a paralelky budou rovněž importovány z informačního systému KOS. Administrátorům bude umožněno přidávat, či odebírat studenty a vyučující z jednotlivých kurzů.

**FP3 - Správa témat semestrálních prací** K jednotlivým předmětům bude možné přidávat nová témata semestrálních prací, ze kterých si student bude moci vybrat. Jednotlivá témata mohou být tříděny do kategorií. Téma se bude skládat z názvu a popisu.

**FP4 - Vytváření úloh jednotlivým kurzům** U jednotlivých kurzů budou vytvářeny úlohy se specifickými požadavky pro jejich splnění. Mezi tyto požadavky spadá například počet odevzdaných témat, požadovaný počet bodů k úspěšnému splnění a datum odevzdání. V rámci úlohy bude možné definovat z jaké kategorie si může student zvolit téma, či určit jedno téma pro všechny studenty. Dále bude možné povolit studentovi vytvořit si vlastní téma a nastavit automatické nebo manuální schvalování těchto individuálních témat.

**FP5 - Výběr tématu semestrální práce** Studentům bude umožněn výběr tématu na základě specifikací úlohy. U jednotlivých témat bude specifikováno, zda si je může vybrat více nebo pouze jeden student. V případě možnosti individuálních témat bude studentovi nabídnuto vytvoření vlastního tématu.

**FP6 - Odevzdání semestrální práce** Po výběru tématu se studentovi vytvoří nový projekt s prostorem pro odevzdání semestrální práce. V projektu mu bude umožněno vytvořit dokumentaci a nahrát zdrojové kódy aplikace. Následně bude moci semestrální práci odevzdat k hodnocení. V závislosti na úloze bude možné odevzdat projekt i po termínu. Vyučující bude v tomto případě moci udělit bodovou penalizaci.

**FP7 - Testování odevzdávaných prací** Při odevzdávání semestrálních prací bude prováděna kontrola struktury a formátu odevzdávaných souborů. V závislosti na úloze bude prováděna validace zdrojových kódů a automatické testování.

**FP8 - Hodnocení odevzdaných prací** Aplikace bude podporovat hodnocení pomocí více kriterií, které budou specifikovány u úlohy nebo jednotlivých témat. K projektu bude umožněno přidat i další kritéria, ve kterých bude vyučujícímu umožněno přidělit další body například za unikátní řešení úlohy nebo odebrat body za pozdní odevzdání.

**FP9 - Publikace semestrálních prací** Z odevzdaných prací bude možné vytvářet galerie, které budou přístupné všem přihlášeným uživatelům. Studenti si budou moci prohlédnout odevzdaná řešení a inspirovat se nimi.

## 2.3 Nefunkční požadavky

Nefunkční požadavky jsou omezení na systém, které musí splňovat. Pro vyvíjenou aplikaci jsou stanoveny následující nefunkční požadavky:

- NP1 - Systém bude řešený formou webové aplikace
- NP2 - Aplikace bude implementována v jazyce PHP verze 7.0 nebo vyšší
- NP3 - Pro implementaci aplikace budou použity vhodné části Nette frameworku
- NP4 - Jako databázový systém bude použit PostgreSQL
- NP5 - Aplikace bude využívat načítání dat z informačního systému KOS pomocí KOSapi

## 2.4 Definice persón

Aplikace bude kromě nepřihlášeného uživatele a administrátora rozlišovat uživatelské role, které lze získat z informačního systému KOS. Tyto role budou vázány k jednotlivým kurzům a paralelkám předmětů. Díky tomu bude moci uživatel vyučovat předměty a zároveň být u jiných kurzů v roli studenta.

**Nepřihlášený uživatel** Nepřihlášený uživatel uvidí pouze úvodní stránku aplikace s přihlašovacím formulářem.

**Student** Student bude mít přístup do studentského modulu aplikace. Uvidí kurzy, ve kterých je zapsaný, jednotlivé úlohy (semestrální práce) a prostor pro jejich odevzdání.

**Vyučující** Uživatelé s těmito rolemi budou mít přístup do učitelské části aplikace. V této části aplikace budou moci vytvářet nové úlohy, témata semestrálních prací, zařazovat je do kategorií a hodnotit odevzdané práce.

**Administrátor** Administrátor bude mít přístup do učitelské a administrátorské části aplikace. Krom toho, že bude mít všechna práva vyučujících, tak bude moci spravovat všechny předměty a uživatele aplikace z administrátorské části.

## 2.5 Životní cyklus semestrální práce

Životní cyklus semestrální práce v aplikaci lze rozdělit na tyto hlavní body:

**Volba tématu** Nejprve si student vytvoří nebo zvolí jedno z dostupných témat semestrálních prací pro danou úlohu. Témata semestrálních prací mohou být slučována do kategorií, které se vážou na předmět.

**Vytvoření dokumentace** Po výběru tématu se studentovi vytvoří nový projekt s prostorem pro odevzdání semestrální práce. Hlavní částí semestrální práce je dokumentace, kterou student vytvoří přímo v aplikaci pomocí WYSIVYG („What you see is what you get“) editoru.

**Nahrání zdrojových kódů** Další částí semestrální práce jsou zdrojové kódy. V závislosti na úloze, bude student vyzván k nahrání souboru požadovaného formátu. V závislosti na úloze se bude kontrolovat formát a struktura odevzdaného souboru případně dojde k automatickému otestování aplikace.

**Odevzdání semestrální práce** Po vytvoření dokumentace a nahrání zdrojových souborů semestrální práce následuje její odevzdání. Student musí své řešení odevzdat kliknutím na tlačítko do stanoveného termínu.

**Hodnocení** Semestrální práce budou v závislosti na jejich charakteru hodnoceni buď automaticky nebo je vyučující ohodnotí manuálně. V obou případech bude moci vyučující povést korekci hodnocení.

**Publikace v galerii** Jednotlivé semestrální práce budou publikovány v galeriích proběhlých kurzů, ve kterých bude uživatelům aplikace umožněno prohlédnout si odevzdaná řešení svých kolegů, kteří kurz již absolvovali.

## 2.6 Odevzdávání semestrálních prací ve vybraných předmětech

### 2.6.1 BI-ZNF

V předmětu Základy Nette frameworku se odevzdává jedna semestrální práce a pět domácích úkolů. Téma semestrální práce si student může vybrat z vypsaných témat z libovolné kategorie nebo si může vymyslet své vlastí. V takovém případě musí být téma schváleno vyučujícím. U domácích úkolů je zadání shodné pro všechny studenty a odevzdává se pouze archiv s aplikací. Hodnocení uděluje cvičící. Dále se domácí úkoly vystavují na server. U semestrálních prací se kromě aplikace odevzdává dokumentace a nepovinná prezentace přičemž se hodnocení rozděluje na několik kritérií. U každého kritéria je stanoven maximální počet bodů. U některých je stanoven i počet minimální.

### 2.6.2 BI-MGA

V předmětu Multimediální a grafické aplikace student musí vypracovat tři semestrální práce, které na sebe navazují a dohromady tvoří jeden větší projekt. Každý student si volí libovolné téma, pouze je zadán určitý okruh, čeho se musí zvolené téma týkat (například „Čím byste chtěli cestovat do kampusu FIT?“). Zvolené téma musí být schváleno vyučujícím. Semestrální práce se hodnotí podle různých kritérií. Za pozdní odevzdání je student penalizován -5 body. Odevzdává se dokumentace a archiv se zdrojovými kódy, který musí splňovat požadovanou strukturu.

### 2.6.3 BI-PGA

V předmětu Programování grafických aplikací student odevzdává minimálně po jednom tématu ze tří nezávislých kategorií. Jednotlivá témata jsou ohodnocena různým počtem bodů podle náročnosti od 10 do 60. U některých témat jsou doplňující úlohy, za které může student získat další bonusové body. Student musí získat maximálně 70 bodů, aby získal zápočet. Ke každé semestrální práci musí být odevzdána dokumentace. Další body lze získat za prezentaci.

## 2.7 Integrace dat z informačního systému KOS

KOS [6] je studijní informační systém, ve kterém se spravují data školy. Díky dostupnému API rozhraní s názvem KOSapi [7] je možné pohodlně získávat data pomocí HTTP GET požadavků ve formátu atom a XML. Přes toto rozhraní budou importovány následující data:

- Uživatelé
- Semestry

- Předměty
- Kurzy přemetů
- Pralelky kurzů
- Vztahy uživatelů ke kurzům a paralelkám

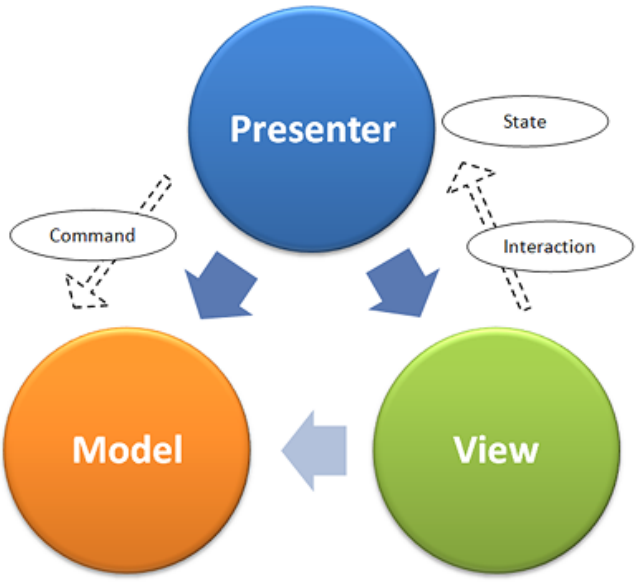
## 2.8 MVP architektura a modulární přístup

Jedním z nefunkčních požadavků je využití Nette frameworku [8], který se řadí mezi frameworky s MVP architekturou. Tato architektura se vyznačuje tím, že rozděluje aplikaci na následující tři vrstvy:

**Model** Vrstva, která se stará o práci s daty a poskytuje jasně stanovené rozhraní. Může například ukládat, upravovat či mazat data z databáze.

**View** View je vrstva aplikace, která zobrazuje výsledek požadavku. V Nette frameworku tato vrstva používá šablonovací systém Latte.

**Presenter** Tato vrstva přijímá požadavky od klienta, stará se o získávání dat z modelů a následně žádá view vrstvu o vykreslení těchto dat.




Obrázek 2.5: MVP architektura Nette frameworku [9]

**Moduly** U složitější aplikace můžeme její presentery a šablony členit do více modulů. Toto členění napomáhá především k přehlednější struktuře aplikace. Pokud by aplikace obsahovala například moduly `Public` a `Admin`, její struktura by mohla vypadat takto:

```
app/                  adresář s aplikací
    PublicModule/     modul Public
        presenters/   jeho presentery
        templates/    jeho šablony
    AdminModule/      modul Admin
        presenters/   jeho presentery
        templates/    jeho šablony
```

Při přechodu na modulární přístup nesmíme zapomenout na změnu routeru — služby, která se stará o obousměrné překládání mezi URL a akcí presenteru.

Kapitola 3

# Návrh

## 3.1 Moduly aplikace

Vzhledem k rozsáhlosti webové aplikace bude využit modulární přístup vysvětlený v kapitole 2.8. Aplikace bude rozdělena na čtyři moduly, které jsou blíže specifikované v následujících kapitolách. Funkce jednotlivých modulů jsou zvoleny podle uživatelských rolí, podle kterých jsou i pojmenovány.

### 3.1.1 Public modul

Public modul bude sloužit pro veřejnost. Tedy pro uživatele, kteří nejsou autentizováni. Tento modul bude obsahovat úvodní stránku s přihlašovacím formulářem a základními informacemi.

**Prohlížení galerií semestrálních prací** Součástí public modulu budou i galerie odevzdaných řešení. Tyto galerie budou však dostupné pouze přihlášeným uživatelům bez ohledu na jejich role. Z tohoto důvodu jsem se rozhodl tuto funkci umístit do tohoto modulu. Jednoduchou úpravou kódu bude možné toto omezení odstranit a zpřístupnit tak galerie i pro veřejnost.

Uživatelé si budou moci prohlédnout galerie proběhlých kurzů všech předmětů v systému. Galerie budou členěny do stromové struktury podle předmětu a semestru.

### 3.1.2 Admin modul

Admin modul bude určený pouze pro uživatele s rolí administrátora. Tento modul bude poskytovat následující funkce:

**Správa uživatelů** První funkcí tohoto modulu bude kompletní správa uživatelů aplikace. Administrátor bude umožněno přidávat, mazat, upravovat a měnit role všem uživatelům. Při tvorbě nového uživatele bude možné jeho

údaje manuálně vyplnit nebo automaticky naimportovat z databáze KOS přes KOSapi pomocí shody uživatelského jména.

**Správa semestrů** Správa semestrů bude sloužit spíše pro informativní účely a smazání celého semestru včetně všech dat, které se k němu budou vázat. Jednotlivé semestry budou automaticky vytvářeny při importování předmětů z KOSu včetně jejich časového intervalu.

**Správa předmětů** Administrátorovi bude umožněno spravovat jednotlivé předměty. Při vytváření předmětů bude dotázán pouze na jeho kód. Ostatní data se naimportují z databáze KOS.

**Správa kurzů** Poslední funkcí toho modulu bude správa kurzů jednotlivých předmětů. Opět zde bude využito importu dat z databáze KOS. Administrátor bude moci hromadně importovat kurzy aktuálně probíhajícího semestru nebo si bude moci zvolit specifický semestr. S kurzem budou do aplikace naimportovány i všechny paralelky, studenti, vyučující a vazby mezi nimi.

### 3.1.3 Teacher module

Teacher modul bude přístupný pro administrátory a všechny vyučující. Vyučující budou mít přístup pouze k předmětům, ve kterých vyučují. Tento modul bude umožňovat následující funkce:

**Vytváření kategorií** Uživatel bude moci spravovat kategorie semestrálních témat. Tyto kategorie budou vázány na jednotlivé předměty, aby mohlo docházet k recyklaci napříč kurzy předmětů. Kategorie budou zavedeny především kvůli předmětu BI-PGA, ve kterých je u každé ze tří úloh požadován výběr z určitého okruhu témat.

**Vytváření témat semestrálních prací** Téma semestrální práce se bude skládat z názvu a zadání, které bude editovatelné pomocí WYSIWYG editoru Summernote [10]. Uživatel bude moci text zadání strukturovat a vkládat do něho obrázky. Dále bude u zadání definován maximální počet studentů, kteří si mohou zadané téma vybrat.

**Vytváření úloh** U jednotlivých kurzů bude možné vytvářet úlohy, které bude možné omezit na určitou kategorii témat, či jedno konkrétní téma. V zadání úlohy bude možné studentovi povolit vytvoření jeho vlastního tématu a odevzdání projektu po termínu. Dalšími parametry úlohy budou minimální a maximální počet odevzdaných témat.

**Hodnocení semestrálních prací** Semestrální práce se budou hodnotit podle kriterií definovaných buď u úlohy nebo konkrétních zadání. Krom těchto kritérií bude moci hodnotící vytvořit další kritérium, například pro udělení penalizace za pozdní odevzdání nebo bonusových bodů za unikátní řešení.

### 3.1.4 Student modul

Student modul bude sloužit výhradně pro studenty. Student po přihlášení uvidí a bude mít přístup pouze ke svým kurzům. U těchto kurzů mu budou umožněny následující funkce:

**Výběr tématu semestrální práce** Na stránce s úlohou bude studentovi v závislosti na povaze úlohy umožněn výběr tématu semestrální práce. Pokud budou u úlohy povolena i individuální témata, bude studentovi nabídnut formulář pro tvorbu jeho vlastního tématu, který bude mít téměř shodnou strukturu jako v učitelském modulu.

**Odevzdání semestrální práce** Po výběru tématu bude studentovi vytvořen projekt. V případě, že si zvolil jedno z dostupných témat, bude studentovi ihned umožněno ho upravovat. V případě, že si vytvořil své vlastní téma a u úlohy nebude aktivní automatické schvalování individuálních témat, tak bude muset student vyčkat na schválení.

V projektu bude mít prostor pro vytvoření dokumentace. K tomu bude sloužit opět WYSIWYG editor. Pod tímto editorem bude tlačítko pro zobrazení náhledu dokumentace, aby si mohl student zkontrolovat její výslednou formu před odevzdáním. Pod dokumentací bude studentovi umožněno nahrání souboru. Při nahrávání se bude kontrolovat požadovaná struktura nebo formát souboru. Po tom co student projekt dokončí, tak ho bude moci odevzdat pomocí tlačítka na konci stránky. V případě, že již bude po termínu a nebude povoleno pozdní odevzdání, bude o této skutečnosti student informován varovnou hláškou.

## 3.2 Kontrola odevzdávaných řešení

### 3.2.1 Struktura a formát souborů

U jednotlivých úloh bude možné nastavit kontrolu odevzdávaného souboru. Na výběr bude formát ZIP a možnost definovat si vlastní formát. V případě výběru ZIP archivu bude možné definovat požadovanou strukturu. Tato struktura bude kontrolována pomocí ZIP extenze pro PHP [11].

Požadovaná struktura bude uložena v entitě úlohy ve formátu JSON. Tento formát se dá v jazyce PHP lehce převést do normálního pole a zpět pomocí

funkcí `json_encode` a `json_decode`. Každý záznam v poli bude jedna absolutní cesta souboru v archivu. Při zadávání požadovaného formátu bude kontrolována validita JSON, aby nedošlo k chybnému zadání.

Pokud bychom například požadovali, aby odevzdávaný archiv obsahoval soubor `readme.txt` a `image.png` ve složce `src/`, bude hodnota vypadat následovně:

```
["src\/image.jpg","readme.txt"]
```

V případě, že nahrávaný soubor nebude splňovat požadovaná kritéria, tak nebude přijat.

### 3.2.2 Testování

Automatické testování Nette projektů a validace zdrojových kódů bude probíhat pomocí CI podobně, jako při vývoji tohoto portálu navrženého v sekci 4.2. Při nahrání projektu do GitLab repozitáře, který bude studentovi přidělen, bude v rámci CI spuštěn skript s automatizovanými testy. Tento skript následně předá informaci portálu o tom, zda došlo k úspěšnému odevzdání aplikace. Ten na základě této informace přidělí body určitým kritériím projektu.

## 3.3 Publikace odevzdaných prací

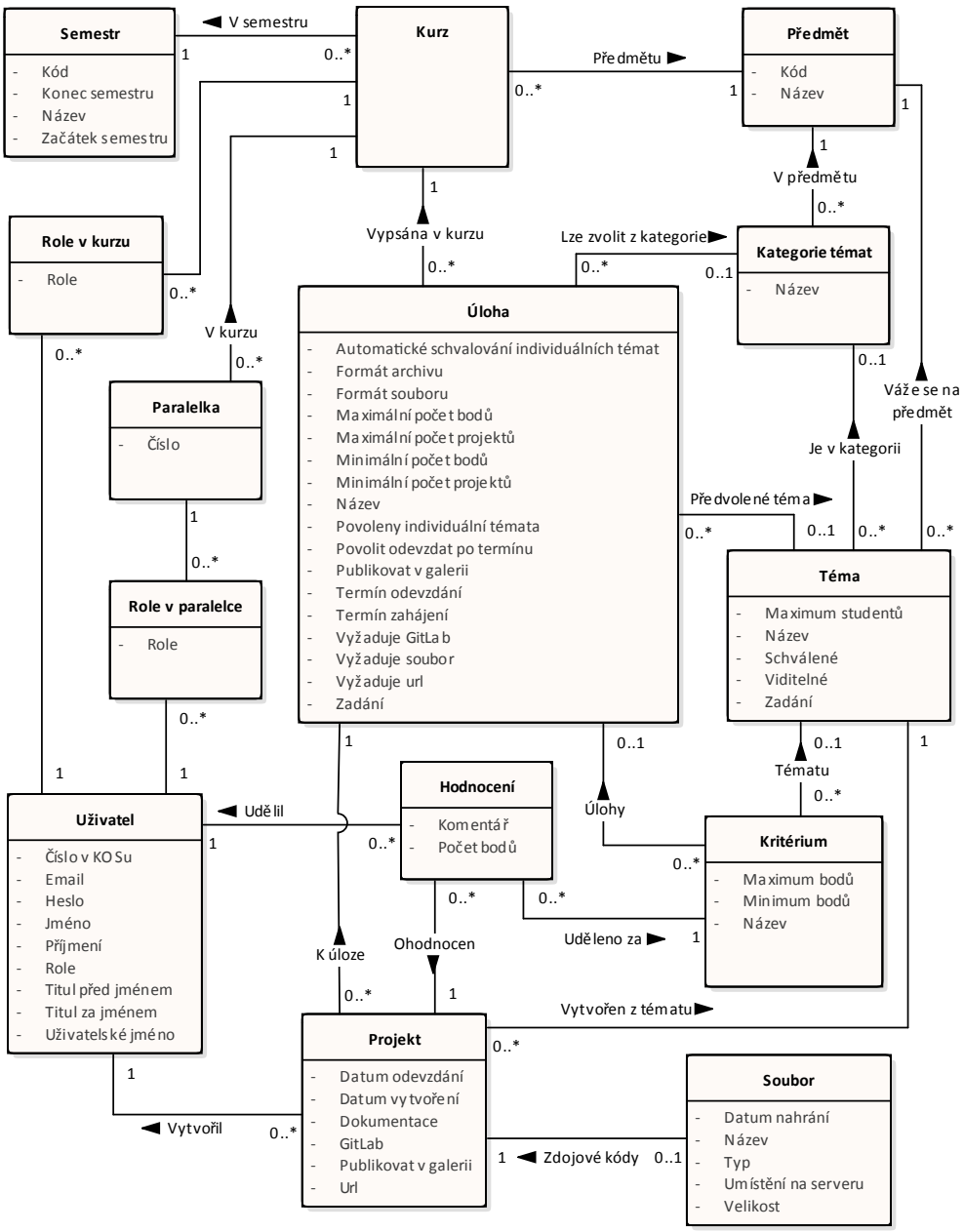
Odevzdané práce se budou publikovat v automaticky vytvářených galeriích. U jednotlivých projektů bude možné nastavit, aby se v galerii nezobrazovali, či zakázat zobrazení celé úlohy. V galeriích budou zobrazeny pouze úplná řešení.

Galerie budou mít výchozí šablonu, podle které se budou vykreslovat. Další šablony bude možné do aplikace jednoduše doprogramovat. V šablonách bude možné používat všechny proměnné projektu a entit, se kterými je v relaci. Dále bude možné použít obrázky z odevzdaného archivu. Tyto obrázky budou načítány přímo z archivu pomocí třídy `ZipArchive` [12], která je součástí ZIP extenze PHP.

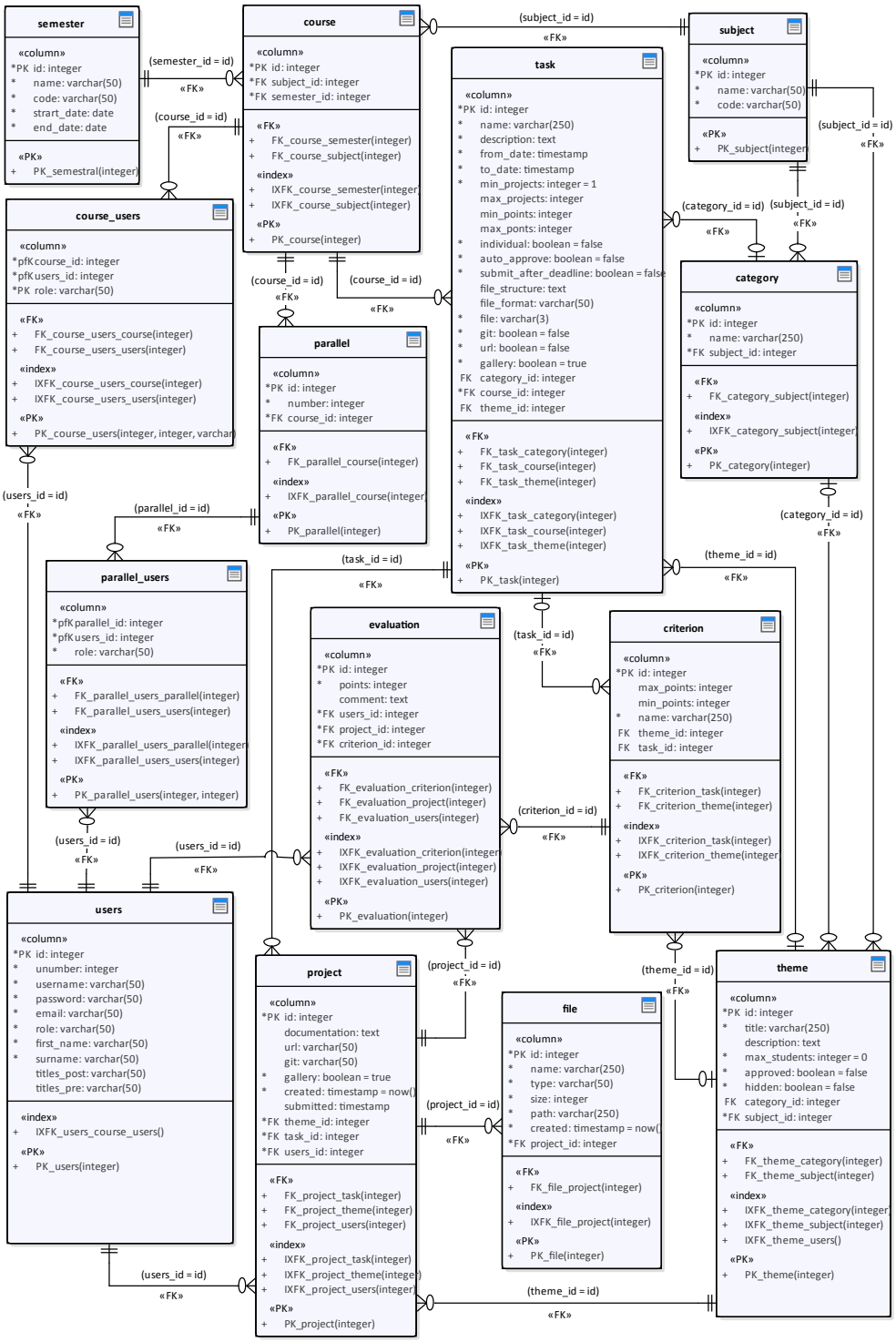
## 3.4 Doménový model

Doménový model na obrázku 3.1 znázorňuje základní entity, jejich parametry a vztahy mezi nimi. Model jsem navrhl na základě analýzy funkčních požadavků na aplikaci. Do budoucna by mohl být model rozšířen o entity pro archivaci tak, aby se s narůstajícím počtem kurzů zbytečně nezahlcovala databáze.

Část doménového modelu je inspirována ER modelem KOSapi pro jednodušší import dat z informačního systému KOS.




Obrázek 3.1: Doménový model



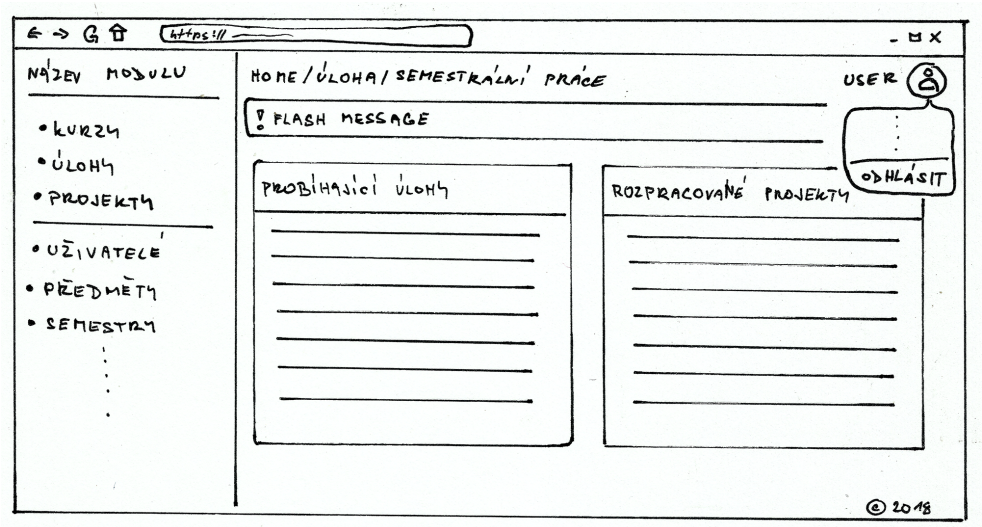

Obrázek 3.2: Relační model

## 3.5 Relační model

Relační model na obrázku 3.2 byl navržen na základě doménového modelu. Znázorňuje všechny tabulky včetně vazebních tak, jak budou uložené v databázi. Mimo hlavních atributů entit jsou v tomto modelu znázorněné i primární a cizí klíče.

## 3.6 Uživatelské rozhraní

Pro uživatelské rozhraní aplikace bude použita šablona Material Dashboard [13], která je v základní verzi volně dostupná pod MIT licencí. Tato šablona je postavena na Bootstrapu [14], což je open source sada nástojů pro tvorbu webových šablon založených na HTML a CSS. Mezi hlavní komponenty tohoto frameworku patří například formulářové prvky, tlačítka, navigace nebo základní rozložení stránky včetně podpory responzivity.




Obrázek 3.3: Wireframe rozložení základních prvků stránky

Na obrázku 3.3 je znázorněno základní rozdělení prvků na stránce. Po levé stráně bude panel, v jehož záhlaví bude název modulu, ve kterém se uživatel aktuálně nachází. Mezi jednotlivými moduly se bude moci přepínat pomocí ikonky siluety uživatele v pravém horním rohu. Po kliknutí na tuto ikonu se rozbalí menu s možností změny modulu a odhlášení. Změna modulu bude umožněna, protože jedna osoba může být studentem v jednom předmětu a v jiném může být v roli vyučujícího. V hlavní části stránky bude na začátku umístěna drobečkova navigace znázorňující polohu uživatele v rámci aplikace.

Pod touto navigací se budou zobrazovat případné informativní zprávy o stavu systému. Dále budou následovat jednotlivé panely s obsahem.

Jednotlivé wireframy všech částí aplikace jsou k dispozici na přiloženém CD ve složce `src/wireframes/`.

## 3.7 Zvolené technologie

Technologie využité pro tuto aplikaci byly z velké části určeny zadáním práce, tedy nefunkčními požadavky.

### 3.7.1 Nette framework

Nette framework je open source framework pro tvorbu webových aplikací v programovacím jazyku PHP. Tento framework vytvořil David Grudl v České republice, kde je v současné době jedním z nejpoužívanějších hned vedle frameworků jako Symfony či Laravel.

### 3.7.2 Šablonovací systém Latte

K propojení frontendu a backendu aplikace bude použit šablonovací systém Latte [15], který je součástí Nette frameworku. Tento systém překládá šablony na optimalizovaný PHP kód. Latte umožňuje namísto používání klasického PHP přímo v HTML šabloně využívat přehledná makra, která se zapisují do složených závorek. Pokud bychom i přesto potřebovali vložit do stránky klasické PHP, tak je to možné pomocí makra s názvem php.

Dalším kladem tohoto systému je možnost použít filtry, které se píší za svislítko hned po proměnné. Mezi standardní filtry patří například formátování data a ořezávání řetězce. Krom těchto filtrů je možné nadefinovat si vlastní filtry v presenteru aplikace.

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

V přiloženém kódu si můžeme všimnout nahrazení klasického php Latte makry. V elementu ul je využit tzv. n:makro, které v tomto případě vypíše blok pouze, pokud má nějaký obsah. Uvnitř tohoto bloku je klasický foreach, který iteruje přes pole $items. Při výpisu prvku pole je využit filtr, který nám změní první písmeno řetězce na velké.

### 3.7.3 PostgreSQL

Jako databázový systém bude zvolen PostgreSQL [16], který se řadí mezi objektově-relační databázové systémy. Stejně tak jako jeho nejznámější konkurent MySQL podporuje použití standardizovaného strukturovaného dotazovacího jazyka SQL.

### 3.7.4 REST

Neboli representational state transfer je architektura rozhraní (API), které umožňuje vytvářet, číst, mazat či editovat data na vzdáleném serveru pomocí jednoduchých HTTP požadavků. Tato technologie bude využita při importování dat z KOSapi, který zprostředkovává část dat z databáze studijního informačního systému KOS.

### 3.7.5 Continuous Integration

Při vývoji bude využito CI (systému průběžné integrace) pomocí GitLab webového repozitáře [17]. Pro konfiguraci CI se používá soubor .gitlab-ci.yml, který je potřeba nahrát do kořenového adresáře projektu. Skripty definované v tomto souboru se budou spouštět při každé změně repozitáře. GitLab se bude k serveru připojovat pomocí SSH klíče k účtu git-user, který bude na serveru vytvořen právě pro CI. Privátní část klíče bude uložena v proměnné na GitLabu, veřejná část bude uložena na serveru. [18]

Jednou z dostupných proměnných je i aktuální větev, do které je změna nahrána. Díky tomu můžeme nahrávat aplikaci na více serverů v závislosti na větvi, do které byla změna nahrána. Při vývoji budu využívat pouze jeden produkční server a svůj počítač pro vývoj, takže se bude na server nahrávat každá změna `master` větve. V budoucnu by mohli být zavedeny další servery pro testování a vývoj. V tomto případě by se mohli na tyto servery nahrávat například větve `test` a `dev`.

Proces bude rozdělen na dvě fáze – `build` a `deploy`. Fáze `build` nainstaluje všechny potřebné knihovny pomocí Composeru zmíněného v kapitole 4.1.2. Ve fázi `deploy` dojde k načtení privátního klíče do systému (veřejná část klíče je na serveru, kam se aplikace bude instalovat), vytvoření balíčku aplikace, zvolení serveru v závislosti na větvi, nahrání balíčku na tento server a jeho instalaci.

V případě automatického testování semestrálních prací bude proces doplněn o třetí fázi `test`, ve které bude prováděno automatické testování.

Kapitola 4

# Implementace

## 4.1 Nástroje pro vývoj

Při implementaci aplikace byly využity nástroje popsané v následujících kapitolách.

### 4.1.1 PHPStorm

PHPStorm [19] je jedno z nejpoužívanějších IDE (integrované vývojových prostředí) pro psaní aplikací v jazyce PHP. Toto IDE vyvíjí společnost JetBrains s.r.o., která mimo tento editor pro PHP vyvíjí IDE i pro jiné programovací jazyky jako Java, Ruby nebo C++. Kromě PHP tento nástroj nabízí plnohodnotný editor pro HTML a JavaScript, správu databází, kontrolu syntaxe, automatické formátování kódu, našeptávání a mnoho dalšího.

### 4.1.2 Composer

Composer [20] je nástroj pro správu PHP knihoven. Umožňuje programátorovi jednoduše nainstalovat všechny požadované knihovny včetně jejich závislostí na jiných knihovnách přičemž kontroluje celkovou kompatibilitu. Composer se stará i o aktualizace již přidaných knihoven.

### 4.1.3 Git

Git [21] je distribuovaný verzovací systém. To znamená, že každý uživatel má staženou svou vlastní kopii repozitáře. V případě, že dojde ke změně na centrálním serveru, tak se tato změna u uživatele neprojeví, dokud si ji nestáhne.

Jednou z hlavních výhod Gitu oproti jiným verzovacím systémům je tzv. větvení. Git umožňuje vytvořit současně více větví, které mohou být navzájem zcela nezávislé. Díky tomuto systému může být vyvíjeno více funkcí aplikace nezávisle na sobě, které lze spojit až po jejich dokončení. Tato vlastnost je

zásadní při práci v týmu, kdy může více programátorů současně pracovat na různých úlohách.

V rámci implementace této práce bude použit školní GitLab [22], což je webový Git repozitář.

### 4.1.4 Adminer

Adminer [23] je nástroj pro správu databází. Tento nástroj je napsaný v jazyce PHP v jednom jediném souboru. Mimo PostgreSQL podporuje správu MySQL, SQLite, Oracle, MS SQL, Firebird, SimpleDB a MongoDB databází. Nově přichází i se správou Elasticsearch.

V aplikaci je Aminer přibalen v adresáři `www/adminer/`. K jeho spuštění stačí přistoupit na adresu `http://<base_uri>/adminer/` pomocí webového prohlížeče, kde hodnota `<base_uri>` je domovská url adresa, na které aplikace běží.

## 4.2 Continues Integration

Při implementaci byl využit systém CI navržený v kapitole 4.2. Vytváření balíčku jsem nejdříve zkoušel pomocí knihovny `dpkg-deb`, u které jsem měl problémy s přepisováním celého projektu při aktualizaci aplikace. Po domluvě s vedoucím práce mi byly poskytnuty jeho skripty pro vytváření balíčků, které jsem následně pozměnil pro tuto práci.

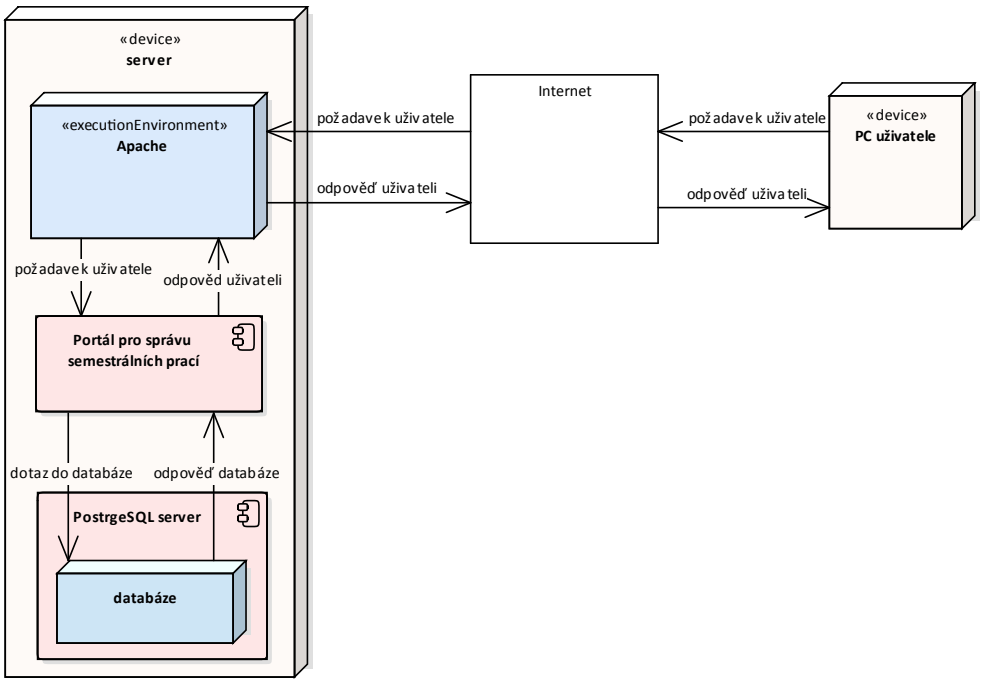
Při vytváření balíčku dochází nejdříve k nakopírování celého projektu vyjma složek Gitu, dočasných souborů, logů a skriptů do nové složky `dist`. Z této složky je následně odmazán soubor `config.local.neon` s parametry pro připojení k databázi. Tento soubor je vymazán z toho důvodu, aby nedošlo k případnému přepsání souboru na serveru, kde budeme balíček instalovat. Místo tohoto souboru je vytvořen vzorový konfigurační soubor, ze kterého lze vycházet. Tento soubor je vytvořen v totožném adresáři jako ostrý soubor. Následně je zavolán příkaz `fpm`, který vytvoří celý balíček:

```
fpm -s dir -t deb -C dist -n $projectname \
    -m "<dvorav15@fit.cvut.cz>" \
    --before-install scripts/before-install.sh \
    --after-install scripts/after-install.sh \
    -v "$version" \
    --iteration "$iteration" \
    -d "apache2" \
    -d "postgresql" \
    -d "postgresql-contrib" \
    -d "php7.0" \
    -d "php7.0-pgsql" \
    var usr etc
```

Při instalaci jsou volány skripty `before-install.sh` a `after-install.sh`, které zajišťují smazání cache projektu, případné vytváření adresářů a nastavení oprávnění. Při instalaci balíčku je ověřováno, zda jsou nainstalované všechny nezbytné knihovny pro chod aplikace, které jsou definovány při jeho vytváření.

## 4.3 Schéma nasazení

Na obrázku číslo 4.1 je znázorněno schéma nasazení aplikace. Aplikace byla během vývoje nasazena na serveru společnosti Big Cloud [24].




Obrázek 4.1: Schéma nasazení

## 4.4 Instalační příručka

Pro spuštění aplikace budeme potřebovat linuxový server s následujícími balíčky:

- postgresql
- postgresql-contrib
- apache2

- php7.0
- php7.0-pgsql
- composer

### 4.4.1 Nastavení serveru

Následující postup byl aplikován na linuxové distribuci Ubuntu 14.04, na jiných distribucích se může postup lišit. Veškeré příkazy budou spouštěny pod root uživatelem. Začneme tím, že se přihlásíme pod root uživatele, aktualizujeme databázi balíčků a vygenerujeme českou lokalizaci:

```
sudo -i
apt-get update
locale-gen cs_CZ.UTF-8
```

**Postresql a postgresql-contrib** Začneme instalací PostrgreSQL serveru, vytvoříme cluster a nastavíme vše potřebné.

```
apt-get install postgresql postgresql-contrib
pg_createcluster 9.3 main --start
service postgresql start
cd /etc/postgresql/9.3/main/
nano pg_hba.conf
```

Na níže uvedeném řádku změníme klíčové slovo `peer` na `md5`.

```
local   all all peer
```

Tato změna nám umožní pozdější přihlášení pomocí hesla k PostgreSQL serveru přes Adminer. Službu PostgreSQL restartujeme, vytvoříme uživatele, databázi a nastavíme potřebná práva:

```
service postgresql restart
sudo -u postgres createuser root
sudo -u postgres createdb bachelor
sudo -u postgres psql
alter user root with encrypted password 'root'; #zvolíme heslo
grant all privileges on database bachelor to root;
```

**Apache2** Další nezbytnou částí instalace je Apache HTTP server, který nainstalujeme následujícími příkazy:

```
apt-get install apache2
service apache2 start
a2enmod rewrite
```

Nyní povolíme použití `.htaccess` souborů pro složku `/var/www/`. V konfikuračním souboru `apache2.conf` v části nastavení zmíněné složky přepíšeme řádek AllowOverride None na AllowOverride all.

```
nano /etc/apache2/apache2.conf

<Directory /var/www/>
        Options Indexes FollowSymLinks
        #AllowOverride None
        AllowOverride all
        Require all granted
</Directory>
```

Na závěr konfigurace je potřeba nastavit cestu do `www` složky webové aplikace. V souboru `/etc/apache2/sites-available/000-default.conf` přidáme do cesty k dokumentu složku `www`. Na závěr zrestartujeme Apache službu.

```
nano /etc/apache2/sites-available/000-default.conf

<VirtualHost *:80>
    #zde můžeme nastavit doménu,
    #kterou musíme povoliti v souboru /etc/hosts
    ServerName localhost

    #zde můžeme změnit cestu k aplikaci
    DocumentRoot /var/www/html/www
</VirtualHost>

service apache2 restart
```

Samozřejmě nemusíme používat výchozí konfigurační soubor, ale můžeme si vytvořit vlastní. To se hodí, pokud nám na serveru poběží více aplikací. Případnou novou stránku aktivujeme příkazem `a2ensite`.

**PHP7.0** PHP 7 nainstalujeme pomocí následujících příkazů:

```
apt-add-repository ppa:ondrej/php
apt-get update
apt-get install php7.0
```

**PHP7.0-pgsql** Pro PHP 7 musíme nainstalovat balíček umožňující připojení k PostgreSQL databázi:

```
apt-get install php7.0-pgsql
service postgresql restart
service apache2 restart
```

**Composer** Na závěr nainstalujeme Composer do složky `/bin`, abychom ho mohli spouštět odkudkoliv na celém serveru:

```
php -r "copy('https://getcomposer.org/installer',
'composer-setup.php');"
php -r "if (hash_file('SHA384', 'composer-setup.php') === '544e0
9ee996cdf60ece3804abc52599c22b1f40f4323403c44d44fdfdd586475ca981
3a858088ffbc1f233e9b180f061') { echo 'Installer verified';} else
{ echo 'Installer corrupt'; unlink('composer-setup.php'); } echo
PHP_EOL;"
php composer-setup.php --install-dir=/bin --filename=composer
```

### 4.4.2 Instalace aplikace

Existuje mnoho způsobů, jak nainstalovat aplikaci na server. V následujících krocích popíši způsob, jak aplikaci nainstalovat pomocí balíčků, které byli využity u Continuous Integration.

1. Vytvoříme si balíček spuštěním příkazu `./makedeb.sh` v kořenovém adresáři aplikace. Tento skript nám vytvoří balíček začínající slovem bachelor a končící koncovkou .deb.

2. Vytvořený balíček zkopírujeme na server pomocí příkazu `scp`.

3. Přihlásíme se na server pomocí příkazu `ssh`.

4. Přesuneme se do složky s balíčkem a spustíme instalaci pomocí příkazu `sudo dpkg -i <soubor>`. Před instalací dojde k automatické kontrole, zda jsou nainstalovány všechny potřebné knihovny popsané v předchozí kapitole 4.4.1. Po instalaci by měla aplikace běžet na IP adrese serveru.

5. Smažeme soubor `/var/www/html/index.html`, který byl automaticky vytvořen při instalaci Apache2. Zároveň zkontrolujeme, zda se nám aplikace do této složky nainstalovala.

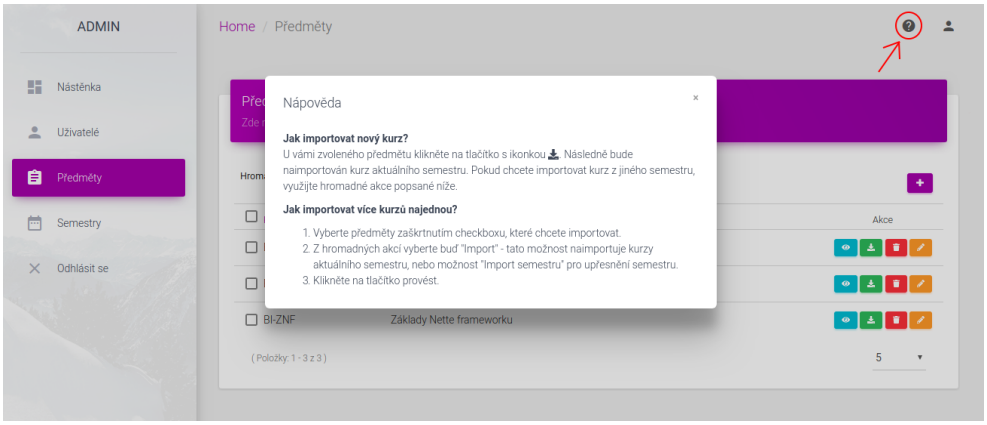
6. Ve složce `/var/www/html/app/config/` vytvoříme konfigurační soubor `config.local.neon` s nastavením připojení k databázi. V této složce je vzorový soubor `example.config.local.neon`, který stačí upravit.

7. Otevřeme si Adminer na adrese `http://<base_uri>/adminer`. Za systém zvolíme PostgreSQL, jako server vyplníme 127.0.0.1:5432, vyplníme přihlašovací údaje zvolené při instalaci databáze a přihlásíme se.

8. Naimportujeme soubor `sql/dump.sql` z adresáře aplikace.

## 4.5 Uživatelská příručka

Uživatelská příručka je řešena přímo v aplikaci. Každá stránka má své modální okno s nápovědou, které lze zobrazit kliknutím na ikonku otazníku v pravém horním rohu. Tato nápověda je zobrazena, pokud je v Latte šabloně stránky dané definovaný blok `helpModal`, který je zobrazen právě v modalu nápovědy. Titulek modálního okna lze změnit definicí bloku `helpTitle`. Příklad modálního okna s nápovědou je na obrázku 4.2.




Obrázek 4.2: Ukázka modálního okna s nápovědou

## 4.6 GDPR a možnost anonymizace

Kvůli obecnému nařízení o ochraně osobních údajů (angl. General Data Protection Regulation neboli GDPR), které nastupuje v platnost dne 25. 5. 2018, je aplikace přizpůsobena pro anonymizaci uživatelů importovaných z informačního systému KOS.

Tato funkce při importování dat z informačního systému KOS nahrazuje všechny osobní údaje uživatelů nesmyslnými řetězci. Anonymizace se bude využívat během uživatelského testování. Lze ji aktivovat v konfiguračním souboru `config.neon`.

Kapitola 5

# Testování

Pro automatické testování této aplikace byl použit Nette Tester framework [25]. Tento framework standardně spouští testy PHP interpretem bez `php.ini`. V tomto souboru je definováno, které rozšíření a nastavení PHP se mají použít. Pro testování jsem si musel vytvořit vlastní `php.ini` soubor, ve kterém jsou definovány všechny potřebná rozšíření k testování, mezi které patří například připojení k databázi, práce s JSON formátem či řetězci. Výsledný obsah souboru vypadá takto:

```
[PHP]
extension=json.so
extension=tokenizer.so
extension=mbstring.so
extension=iconv.so
extension=pdo.so
extension=pdo_pgsql.so
```

Implementaci jednotlivých testů naleznete na přiloženém CD ve složce `src/impl/tests/`.

## 5.1 Jednotkové testy

Unit testy neboli jednotkové testy mají za úkol ověřit funkčnost dílčích částí neboli jednotek zdrojového kódu. Za jednotku zdrojového kódu lze považovat například nějakou funkci či metodu. Tyto testy byly použity k testování funkcí presenterů, služeb a továrniček, kde se testuje jejich návratová hodnota v závislosti na vstupních parametrech.

## 5.2 Integrační testy

Integrační testy slouží k ověření komunikace mezi jednotlivými komponentami uvnitř aplikace. V mé práci se integrační testy používají k otestování komunikace mezi modely a databází či u funkcí služeb, které pracují s daty z databáze.

## 5.3 Uživatelské testování

Uživatelské testování je jednou z nejdůležitějších částí testování použitelnosti aplikace. Tato metoda zkoumá především reakce uživatelů na uživatelské rozhraní aplikace, zda se v něm dokáží sami zorientovat bez pomoci další osoby, zda je přehledné atd. Veškeré podklady k uživatelskému testování jsou na přiloženém CD ve složce `user-testing/`.

**Vstupní dotazník** Na začátku testování uživatelé vyplní krátký vstupní dotazník, který poskytne informace o jejich zkušenostech s podobnými aplikacemi a jejich roli v rámci aplikace. Dotazník obsahuje tyto otázky:

- Jaká je vaše role na fakultě? a) Vyučující b) student c) oboje d) žádná
- Máte zkušenosti s fakultními systémy pro správu semestrálních prací? Pokud ano, s jakými?
- Absolvoval/a jste či vyučoval/a některý z předmětů BI-ZNF, BI-PGA nebo BI-MGA? Pokud ano, tak který?
- Máte zkušenost s verzovacím systémem Git?
- Víte co to je Continuous Integration a soubor .gitlab-ci.yml?

**Scénáře** Scénáře pro uživatelské testování budou dva. Jeden pro studenty a druhý pro vyučující. Uživatel dostane scénář v závislosti na vstupním dotazníků. Pokud uživatel vyplnil, že na fakultě vyučuje, bude mu zadán scénář pro vyučující. V opačném případě dostane scénář pro studenty.

**Výstupní dotazník** Cílem výstupního dotazníku je zjistit uživatelovi pocity ze scénářů. Tento dotazník obsahuje otázky na kvalitu aplikace. Otázky v tomto dotazníku jsou inspirovány Nielsonovou heuristikou.

# Závěr

V první části věnované analýze byly zhodnoceny aktuálně používané systémy pro správu semestrálních prací na této fakultě. Systémy jsem ohodnotil podle mnou stanovených kritérií, přičemž jsem přišel na jejich silné a slabé stránky, které jsem využil při návrhu. Dále v této části byly analyzovány životní cykly semestrálních prací jednotlivých předmětů, které pomohly upřesnit funkční požadavky na aplikaci.

V části věnované návrhu byly navrženy jednotlivé části aplikace, doménový model, relační model, proces odevzdávání semestrálních prací a uživatelské rozhraní jednotlivých částí. Dále jsou v této části popsány zvolené technologie, které byly z části určené zadáním. Mezi tyto technologie patří i systém Continuous Integration, který po celý čas vývoje integroval nové části aplikace na server.

Aplikace byla implementovaná jako webová aplikace na Nette frameworku, který byl hlavním bodem nefunkčních požadavků. Byly splněny všechny nefunkční požadavky na aplikaci. Z funkčních požadavků nebylo splněno automatické testování odevzdávaných prací. Tento proces je navržen, avšak po domluvě s vedoucím práce, nedošlo kvůli jeho náročnosti k implementaci. Ostatní funkční požadavky byly splněny.

V poslední fázi byla aplikace otestována jednotkovými a integračními testy. Pro uživatelské testování byly vypracovány všechny materiály, avšak teprve proběhne během letních prázdnin. Během tohoto období dojde i k implementaci školní Shibboleth SSO brány, která zajišťuje autentizaci uživatelů fakulty. Ta vyžaduje platný SSL certifikát, který jsem neměl k dispozici.

Aplikace byla primárně vyvíjena pro tři určené předměty, lze ji však použít pro jakýkoliv předmět vyučovaný na ČVUT. První ostré spuštění aplikace pro předměty BI-PGA a BI-MGA proběhne v zimním semestru 2018/2019, pro předmět BI-ZNF v letním semestru téhož ročníku.

# Literatura


[1] Svoboda, V.: Nielsen's Heuristic Evaluation [online]. [cit. 2018-04-01]. Dostupné z: http://blog.vojtasvoboda.cz/nielsens-heuristic-evaluation

[2] Progtest [online]. [cit. 2018-04-04]. Dostupné z: https://progtest.fit.cvut.cz/

[3] BI-DBS: Přihlášení [online]. [cit. 2018-04-04]. Dostupné z: https://dbs.fit.cvut.cz/

[4] ČVUT Moodle-výuka [online]. [cit. 2018-04-04]. Dostupné z: http://moodle-vyuka.cvut.cz/

[5] Edux [online]. [cit. 2018-04-04]. Dostupné z: https://edux.fit.cvut.cz/

[6] STUDIJNÍ INFORMAČNÍ SYSTÉM (KOS) [online]. [cit. 2018-04-10]. Dostupné z: https://kos.is.cvut.cz/

[7] KosAPI [online]. [cit. 2018-04-10]. Dostupné z: https://kosapi.fit.cvut.cz/projects/kosapi/wiki

[8] Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework [online]. [cit. 2018-04-14]. Dostupné z: https://nette.org/

[9] Grudl, D.: Nette Framework: MVC a MVP [online]. [cit. 2018-04-15]. Dostupné z: https://www.zdrojak.cz/clanky/nette-framework-mvc-mvp/

[10] Summernote - Super Simple WYSIWYG editor [online]. [cit. 2018-04-16]. Dostupné z: https://summernote.org/

[11] PHP: Zip - Manual [online]. [cit. 2018-04-20]. Dostupné z: http://php.net/manual/en/book.zip.php

[12] PHP: ZipArchive - Manual [online]. [cit. 2018-04-20]. Dostupné z: http://php.net/manual/en/class.ziparchive.php

[13] Material Dashboard: Free Bootstrap 4 Material Admin [online]. [cit. 2018-04-21]. Dostupné z: https://www.creative-tim.com/product/material-dashboard

[14] Bootstrap · The most popular HTML, CSS, and JS library in the world. [online]. [cit. 2018-04-21]. Dostupné z: https://getbootstrap.com/

[15] Latte | Nette Framework [online]. [cit. 2018-04-22]. Dostupné z: https://latte.nette.org/cs/

[16] PostgreSQL: About [online]. [cit. 2018-04-22]. Dostupné z: https://www.postgresql.org/about/

[17] Hujer, M.: GitLab jako Continuous Integration (nejen) pro PHP [online]. [cit. 2018-04-25]. Dostupné z: https://www.zdrojak.cz/clanky/gitlab-jako-continuous-integration-nejen-pro-php/

[18] Using SSH keys with GitLab CI/CD | GitLab [online]. [cit. 2018-04-25]. Dostupné z: https://docs.gitlab.com/ee/ci/ssh_keys/

[19] PhpStorm: Lightning-Smart IDE for PHP Programming by JetBrains [online]. [cit. 2018-04-28]. Dostupné z: https://www.jetbrains.com/phpstorm/

[20] Composer [online]. [cit. 2018-04-28]. Dostupné z: https://getcomposer.org/

[21] Git [online]. [cit. 2018-04-28]. Dostupné z: https://git-scm.com/

[22] GitLab [online]. [cit. 2018-04-28]. Dostupné z: https://gitlab.fit.cvut.cz/

[23] Adminer - Správa databáze v jednom PHP souboru [online]. [cit. 2018-04-28]. Dostupné z: https://www.adminer.org/cs/

[24] Big Cloud [online]. [cit. 2018-05-01]. Dostupné z: https://bigcloud.cz

[25] Nette Tester - pohodové testování | Nette Framework [online]. [cit. 2018-05-10]. Dostupné z: https://tester.nette.org/cs/

Dodatek A

# Seznam použitých zkratek

**API** Application Programming Interface

**BI-DBS** Databázové systémy

**BI-MGA** Multimediální a grafické aplikace

**BI-PGA** Programování grafických aplikací

**BI-ZNF** Základy programování v Nette

**CI** Continuous Integration

**CSS** Cascading Style Sheets

**ČVUT** České vysoké učení technické v Praze

**FIT** Fakulta informačních technologií

**GDPR** General Data Protection Regulation

**HTML** HyperText Markup Language

**HTTP** Hypertext Transfer Protocol

**JSON** JavaScript Object Notation

**MIT** Massachusetts Institute of Technology

**MVP** Model-view-presenter

**PHP** Hypertext Preprocessor

**REST** Representational State Transfer

**SQL** Structured Query Language

**SSH** Secure Shell

**SSL** Secure Sockets Layer

**URI** Uniform Resource Identifier

**URL** Uniform Resource Locator

**WYSIWYG** What you see is what you get

**XML** Extensible markup language

Dodatek **B**

# Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
    - `images` ..... obrázky použité v práci
  - `wireframes` ..... wireframy jednotlivých částí aplikace
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `user-testing` ..... materiály k uživatelskému testování