

**UNIVERZITA KOMENSKÉHO V BRATISLAVE  
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

**NOVÝ DLHODOBÝ VIACÚČELOVÝ SKLAD ÚLOH NA CIČENIA**

Bakalárska práca

2013

Andrej Jursa

**UNIVERZITA KOMENSKÉHO V BRATISLAVE**  
**FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

**NOVÝ DLHODOBÝ VIACÚČELOVÝ SKLAD ÚLOH NA CIČENIA**

Bakalárska práca

Študijný program: Aplikovaná informatika  
Študijný odbor: 9.2.9 Aplikovaná informatika  
Školiace pracovisko: Katedra aplikovanej informatiky  
Školiteľ: Mgr. Pavel Petrovič PhD.

**Bratislava, 2013**

**Andrej Jursa**





Univerzita Komenského v Bratislave  
Fakulta matematiky, fyziky a informatiky

## ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Andrej Jursa  
**Študijný program:** aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)  
**Študijný odbor:** 9.2.9. aplikovaná informatika  
**Typ záverečnej práce:** bakalárska  
**Jazyk záverečnej práce:** slovenský

**Názov:** Nový dlhodobý viacúčelový sklad zadání

**Cieľ:** V predchádzajúcej bakalárskej práci a v ďalšom vývoji bola vytvorená webová aplikácia na archivovanie zadání úloh, vytváranie zostáv úloh, ich opravovanie, hodnotenie študentov, výber projektov, atď. V súčasnosti je tento systém nasadený štvrtý rok a obsahuje viac ako 800 zadání úloh a stovky zostáv. Používatelia systému (cvičiaci a prednášajúci) však potrebujú doplniť do systému ďalšiu funkcionálnu - napríklad automatické hodnotenie riešení, prípadne sprehľadniť a klasifikovať úlohy, ktoré sú v systéme zadane. Úlohou študenta je analyzovať súčasný stav, navrhnúť, implementovať a otestovať zmeny v prevádzkovanom systéme.

**Literatúra:** 1. Peter Jurčo: Lamsfet - long term and multipurpose Storage for Exercise Tasks, bakalárska práca, FMFI UK, 2009.  
2. Monte Ohrt: Smarty - the compiling PHP template engine, Smarty manual, on-line: <http://www.smarty.net/docs/en/>


**Vedúci:** Mgr. Pavel Petrovič, PhD.  
**Katedra:** FMFI.KAI - Katedra aplikovanej informatiky  
**Vedúci katedry:** doc. PhDr. Ján Rybár, PhD.


**Spôsob sprístupnenia elektronickej verzie práce:**  
bez obmedzenia

**Dátum zadania:** 08.10.2012

**Dátum schválenia:** 24.10.2012

doc. RNDr. Mária Markošová, PhD.  
garant študijného programu

  
.....  
študent

  
.....  
vedúci práce

## Abstrakt

Motiváciou tejto práce je zhodnotenie systému LaMSfET (Longterm and Multipurpose Storage for Exercise Tasks) a vyhodnotenie jeho nedostatkov, či už v jeho zdrojovom kóde alebo v jeho funkcionalite. Následkom tejto analýzy je zhodnotenie možností, akými opraviť pôvodný systém, pričom jednou z možností je vytvoriť systém nový. Takto vzniká systém Long-term Internet Storage of Tasks, skrátené LIST.

Cieľom tejto práce je okrem analýzy pôvodného systému vytvorenie návrhu nového systému LIST a implementácia väčšiny jeho funkcionality. Nový systém má byť postavený na dostupných a dobre zdokumentovaných open-source riešeniach, čo má umožniť jednoduchý vývoj systému v budúcnosti po naplnení cieľov tejto bakalárskej práce.

**Kľúčové slová:** LaMSfET, LIST, sklad úloh.

## **Abstract**

The motivation of this bachelor work is to analyze the original system LaMSfET (Longterm and Multipurpose Storage for Exercise Tasks) for all deficiencies, in the source code and in its functionality. Following this analysis there were evaluation of possibilities, how to improve and fix this original system. One of this possibility is creation of new system. Here is the beginning of Long-term Internet Storage of Tasks, abbreviated as LIST.

The main objective of this bachelor work is, in addition to analysis of original system, create design of new system LIST and to implement majority of its functionality. New system should be build on well documented open-source solutions, which have to allow easy development of this system in the future, after fulfilling the objective of this bachelor work.

**Key words:** LaMSfET, LIST, storage of tasks.

Čestne prehlasujem, že som túto prácu vypracoval samostatne,  
s použitím literatúry a zdrojov uvedených v závere práce.

---

Andrej Jursa

## **Pod'akovanie**

Rád by som sa poďakoval svojmu školiteľovi bakalárskej práce Mgr. Pavlovi Petrovičovi, PhD. za cenné rady, pripomienky a usmernenia počas celého procesu tvorby tejto práce.



# Obsah

<b>1 Úvod .....</b>	<b>1</b>
1.1 Ciele práce .....	1
1.2 Prehľad o obsahu dokumentu .....	2
<b>2 Východiská .....</b>	<b>3</b>
2.1 Prehľad teórie .....	3
2.1.1 E-learning a learning management system .....	3
2.1.2 SCORM .....	3
2.1.3 Unit testing .....	4
2.2 Prehľad existujúcich systémov .....	5
2.2.1 Moodle .....	5
2.2.2 Chamilo .....	5
2.2.3 LaMSfET .....	6
2.3 Prehľad technológie .....	7
2.3.1 JUnit .....	7
2.3.2 Smarty 3 .....	8
2.3.3 TinyMCE .....	10
2.3.4 JQuery / JQueryUI .....	10
2.3.5 CodeIgniter .....	11
2.3.6 DataMapper ORM .....	12
2.3.7 GeSHi .....	13
2.3.8 Plupload .....	13
2.3.9 FancyBox 2 .....	13
<b>3 Zhodnotenie východiskovej práce .....</b>	<b>15</b>
<b>4 Požiadavky na softvér .....</b>	<b>20</b>
<b>5 Návrh riešenia .....</b>	<b>21</b>
5.1 Používatelia systému .....	21
5.2 Jazykový preklad systému .....	21
5.3 Úlohy a zostavy úloh .....	22
5.4 Unit testy .....	23
5.5 Zostavovanie unit testov .....	24
5.6 Spustenie unit testu .....	24
5.7 Zálohovanie a obnova systému .....	25
5.8 Dátová štruktúra databázy .....	26
5.9 Dátová štruktúra súborového systému .....	27
5.10 Používateľské rozhranie pre učiteľov (administrácia) .....	28
5.11 Používateľské rozhranie pre študentov .....	32
<b>6 Realizácia a implementácia .....</b>	<b>36</b>
6.1 Implementácia databázy a modelov .....	36
6.1.1 Implementácia kategórií úloh .....	36
6.1.2 Implementácia filtrovania úloh podľa kategórií .....	37
6.1.3 Zostavy úloh a ich implementácia modelom .....	38
6.2 Implementácia jazykových lokalizácií .....	38
6.2.1 Priama lokalizácia pomocou používateľských konštánt .....	38
6.2.2 Lokalizácia textov pomocou prekrytí .....	39
6.2.3 Vyhľadávanie a zoradovanie výsledkov dopytov podľa lokalizovaných stĺpcov .....	40
6.3 Bezpečnosť systému .....	40
<b>7 Záver .....</b>	<b>42</b>
<b>8 Použitá literatúra a prílohy .....</b>	<b>43</b>
8.1 Literatúra .....	43
8.2 Prílohy .....	43

# 1 Úvod

V dnešnej dobe uponáhľaného životného štýlu každého z nás a špecificky veľkého pracovného vytiaženia učiteľov je potrebné mať nástroje, ktoré dovoľujú efektívne a rýchlo spravovať výučbu. V mojej práci sa venujem hlavne takým vyučovacím kurzom, ktoré sa venujú výučbe programovania.

Počas rokov, v ktorých sa už programovanie vyučuje na nespočte školách bolo vytvorených množstvo rôznych úloh pre žiakov či študentov, ktoré im majú pomôcť lepšie poznať a pochopiť ten či onen programovací jazyk. Je výhodné mať systém pre podporu výučby na webovom princípe, ktorý je navrhnutý tak, aby slúžil aj ako dlhodobá databáza rôznych takýchto úloh, z ktorých sa dajú skladať zadania pre študentov na rôzne príležitosti, či už na cvičenia, skúšky, priebežné testy alebo domáce úlohy.

Predstavme si systém, ktorý dovoľí mať v ňom uložené stovky či tisíce takýchto úloh a príloh k nim (bežne vo forme súborov uložených na disku), vyhľadať a vytvoriť z týchto úloh zadania pre študentov a distribuovať im ich pomocou webovej prezentácie, kde bude študent navyše vidieť aj vlastné hodnotenie, prípadne si môže prezrieť informácie o kurzoch, hlásiť sa na ne a pri jednotlivých zadaniach konzultovať so spolužiakmi o ich riešení či problémov s nimi.

## 1.1 Ciele práce

Cieľom práce je vykonať analýzu pôvodného systému LaMSfET, vytvoreného vrámci bakalárskej práce v roku 2009, nasadeného, používaného na niekoľkých kurzoch a vyvíjaného po dobu štyroch rokov. Cieľom analýzy je rozhodnúť, ako pôvodný systém upraviť do lepšej prehľadnosti, zdrojového kódu ako aj obsahu, alebo rozhodnúť o tom, či vytvoriť nový podobný systém.

Po zanalyzovaní pôvodného systému je treba na základe analýzy navrhnúť zmeny v starom systéme alebo urobiť návrh nového systému a implementovať čo možno najviac z týchto zmien alebo nového systému.

## 1.2 Prehľad o obsahu dokumentu

V nasledujúcich kapitolách tohto dokumentu budú objasnené niektoré východiská pri vytváraní vlastnej implementácie softvéru, povieme si niečo o návrhu novej softvérovej implementácie a požiadavkách na ňu. Tiež sa dozvieme niečo o realizácii implementácie nového softvéru.

## 2 Východiská

V tejto kapitole predstavím východiskové informácie ako je teória, existujúce systémy či východisková práca.

### 2.1 Prehľad teórie

#### 2.1.1 E-learning a learning management system

E-learning je vzdelávací proces, využívajúci informačné a komunikačné technológie k tvorbe kurzov, k distribúcii študijných materiálov, komunikáciou medzi študentami a pedagógmi a k riadeniu štúdia. V e-learningu sa často používajú tzv. learning management systémy.

Learning management system alebo LMS je software na administráciu, dokumentáciu, sledovanie, spravovanie a predávanie vzdelávacích kurzov či tréningových programov.

Takýto LMS by mal byť schopný:

- centralizovať a automatizovať administráciu,
- používať samo-obslužné a samo-riadiace služby,
- zhromažďovať a rýchlo poskytovať vzdelávací obsah,
- podporovať prenositeľnosť a normy,
- upravovať obsah a povoliť znovupoužitie vedomostí,
- poskytovať online tréning a semináre,
- evidencia a správa študentov,
- evidencia a správa kurzov,
- správa študijných plánov,
- evidencia hodnotenia študentov,
- testovanie a preskúšanie študentov,
- správa prístupových práv,
- komunikačné nástroje (diskusné fórum, chat),
- úložisko výukových materiálov.

Väčšina LMS systémov je webovo-založená, čo uľahčuje prístup k vzdelávaciemu obsahu a správe tohto obsahu. LMS sú často používané vzdelávacími inštitúciami na zlepšenie a podporu vzdelávania v triedach a ponúkajú kurzy väčšiemu počtu študentov celosvetovo.

#### 2.1.2 SCORM

Shareable Content Object Reference Model je referenčný model pre e-learning. Ide o súbor špecifikácií a štandardov, ktorých hlavnou úlohou je umožnenie používať obsah vytvorený v

súlade so štandardom SCORM v ľubovольnom LMS. Ako z názvu vyplýva, ide o model zdieľateľných obsahových objektov (SCO), umožňujúce znovupoužitie vzdelávacieho obsahu vo všetkých produktoch alebo platformách, ktoré sú SCORMu prispôsobené. SCORM používa na popis objektov manifest, čo je popisný súbor v XML (eXtensible Markup Language). Aplikačný profil metadát popisujúcich SCORM objekty má 64 prvkov, no len malá časť z vyžadovaná povinne pre dosiahnutie zhody s referenčným modelom. Tieto sa delia do deviatich kategórií:

- všeobecná kategória,
- životný cyklus,
- meta-metadáta,
- technická kategória,
- vzdelávacia kategória,
- právna kategória,
- vzťahy,
- anotácie,
- klasifikácia.

Tento model je vytváraný americkou iniciatívou ADL (Advanced Distributed Learning Initiative) s odvolávaním sa na normy vytvárané konzorciami IEEE a IMS Learning Technology Standards.

### 2.1.3 Unit testing

V programovaní je unit testing (voľne preložené ako jednotkové testovanie) metóda, ktorou sa individuálne časti zdrojového kódu, množiny jedného či viacerých modulov programu spojených kontrolnými dátami, procedúry a funkcie testujú aby sa overilo, že pracujú tak ako bolo zamýšľané. Intuitívne, unit je možno vidieť ako najmenšiu testovateľnú časť aplikácie.

V procedurálnom programovaní môže byť takýmto unitom aj celý modul, ale bežnejšie sú to samotné procedúry a funkcie. V objektovo-orientovanom programovaní unitom často býva celý interface, ako sú triedy, ale môžu to byť aj individuálne metódy triedy.

Unit testy sú vytvárané programátormi, prípadne občas testermi, počas procesu vývoja programu.

Pri testovaní sa používajú takzvané test case (testovacie prípady), čo sú množiny podmienok a premenných, pomocou ktorých sa určuje, či aplikácia alebo softvérový systém pracuje správne. Ideálne je, ak sú tieto testovacie prípady na sebe nezávislé, pričom pri testovaní môžu na zlepšenie izolácie jedného od druhého používať rôzne nekompletné metódy, makety objektov, falošné alebo skúšobné dáta či objekty.

Unit testy sa používajú v takzvanom testami-riadenom vývoji (test-driven development, TDD),

metodológiách Extrémneho programovania alebo Scrum, kde sa skôr ako sa píše kód programu, napíšu sa najprv testy, ktoré musí program spĺňať. Ak potom napísaný program spĺňa všetky definované testy, je považovaný za kompletný. Ak unit test zlyhá, je možné zvažovať len dve možnosti, buď je chyba v kóde, ktorý testu nevyhovel, alebo je chyba v teste. Unit testy do veľkej miery napomáhajú k objaveniu práve chýb v kóde, keďže pomáhajú presnejšie vymedziť, pri akých vstupoch sa aká časť kódu správa inak ako je od nej očakávané.

## 2.2 Prehľad existujúcich systémov

### 2.2.1 Moodle

Moodle je open source systém na tvorbu elektronických výukových kurzov na internete. Ide o modulárny learning management system, ktorý poskytuje funkcionality pre:

- pridávanie a správu študijných materiálov vo forme HTML stránok, súborov na stiahnutie, FLASH animácií, štruktúrovaných prednášok a podobne,
- podporuje diskusné fóra s možnosťou odoberania príspevkov e-mailom,
- vytváranie a správa úloh pre účastníkov kurzov,
- podporuje automaticky vyhodnocované testy zložené z rôznych typov testovacích úloh,
- slovníky a databázy, na ktorých napĺňaní sa môžu podieľať aj účastníci kurzov,
- podporuje ankety a hlasovania,
- vzdelávací obsah je v špecifikáciách SCORM alebo IMS Content Package.

Moodle tiež umožňuje evidenciu študijných výsledkov účastníkov kurzov / študentov, taktiež zaznamenáva činnosť používateľov v podrobných protokoloch a súhrnných štatistikách. Moodle je napojiteľný na iné systémy, napríklad autentifikačné systémy ako LDAP, komunikačné ako IMAP či systémy pre správu obsahu ako Postnuke.

Na rozdiel od systému LaMSfET, ktorý je hlavným východiskom tejto práce, systém Moodle nepodporuje dlhodobé skladovanie úloh a ich znovupoužitie či klonovanie do nových zostáv úloh. Tak isto je Moodle na rozdiel od LaMSfETu orientovaný všeobecne, nepodporuje automatické spracovanie odovzdaných riešení, napr. formou testovania riešenia formou unit testov.

### 2.2.2 Chamilo

Chamilo je open-source e-learning a content management system, zameraný na globálne zlepšenie prístupu k vzdelaniu a vedomostiam. Vytvára ho asociácia Chamilo Association, ktorá sa rovnako stará aj o jeho propagáciu, údržbu komunikačných kanálov týkajúcich sa systému a vytvárajú sieť poskytovateľov služieb a prispievateľov do tohto softvéru.

Momentálne existujú dve verzie Chamilo, verzia 1.\*, postavená na Dokeos LMS, a verzia 2.0, ktorá je kompletne novým softvérom.

Chamilo podporuje:

- správu kurzov, používateľov a tréningových cyklov vrátane SOAP webových služieb pre vzdialenú správu,
- je kompatibilný so SCORM 1.2,
- viac-inštitučný mód s centrálnym manažmentom,
- testy študentov, ktoré môžu byť časovo obmedzované / kontrolované,
- natívne používa UTF-8 medzinárodné znakové sady a časové zóny,
- zaznamenáva pokroky používateľov / študentov,
- vytvára sociálnu sieť študentov.

Tak ako Moodle, aj Chamilo je veľmi všeobecne orientovaný softvér, ktorý nepodporuje nijak dlhodobé uchovávanie úloh. Jeho výhodou je množstvo komunitných a komunikačných funkcií, a vcelku jednoduché manažovanie obsahu, dokonca podporuje video konferencie. Nevýhodou je, že nie je takmer vôbec vhodný na vyučovanie informatických predmetov.

### 2.2.3 LaMSfET

Systém LaMSfET (Longterm and Multipurpose Storage for Exercise Tasks – Dlhodobý a viacúčelový sklad úloh na cvičenia) je webovo-orientovaný software podobný LMS, ktorý vznikol ako bakalárska práca študenta aplikovanej informatiky Petra Jurča v roku 2009 a ďalej bol nasadený, vyvíjaný a udržiavaný samotným autorom a Mgr. Pavlom Petrovičom, PhD. na univerzitnej doméne [capek.ii.fmph.uniba.sk](http://capek.ii.fmph.uniba.sk), kde slúži ako kvázi-LMS pre podporu predmetov Programovanie 4, 5, Programovacie paradigmy a pod.

Tento systém sa na rozdiel od iných LMS vyznačuje snahou dlhodobo uchovávať úlohy a ich riešenia, zaraďovať úlohy do zostáv úloh a tieto znovu-používať pri cvičeniach.

LaMSfET umožňuje úlohy označovať značkami z hierarchickej štruktúry značiek, čo umožňuje správcom / učiteľom jednoducho nájsť staršie úlohy na požadovanú tému a použiť ich v novej zostave úloh, prípadne urobiť klon takejto úlohy a pozmeniť v nej hodnoty pred vložením do novej zostavy úloh.

Systém má integrované jednoduché testovanie riešení úloh týkajúcich sa programovacieho jazyka Java. Toto testovanie funguje na princípe očakávaného výstupu z programu po načítaní dát zo štandardného vstupu.

Systém má integrovanú aj podporu pre unit testy Javových programov, táto podpora je v ňom však integrovaná vcelku nesystémovo, čo znamená, že na vytváranie unit testov sa nedá použiť administrátorské rozhranie. Tento stav sa budem snažiť vylepšiť v tejto práci, tj. aby bolo možné z administrácie ku konkrétnym úlohám vygenerovať JUnit testy.

V prípade použitia jednoduchých testov je systém schopný po otestovaní odovzdanej úlohy navrhovať počet bodov.

LaMSfET nie je plnohodnotným LMS, má len veľmi úzku paletu možností zacielenú hlavne na vytváranie, uchovávanie, znovu-používanie úloh, možnosť spravovať študentov, kurzy a skupiny, odovzdávať úlohy a hodnotiť ich, prípadne vyberať študentské úlohy ako vzorové riešenia. Systému chýba takmer akákoľvek interakcia vyučujúcich so študentami, keďže systém neobsahuje žiadne diskusné fóra, chat, ani len komentáre k zostavám či úlohám od študentov. Akákoľvek takáto komunikácia s musí diať mimo systém. LaMSfET takisto nedovoľuje učiteľom vytvoriť informačnú nástenku kurzu, čo býva zvykom v LMS systémoch.

## 2.3 Prehľad technológií

### 2.3.1 JUnit

JUnit je unit testing framework pre programovací jazyk Java. Tento framework je dôležitou súčasťou test-driven development metodológie v tomto jazyku a je súčasťou unit testing frameworkov z rodiny xUnit.

JUnit sa linkuje k programu ako JAR v čase kompilácie, používa sa ako balíček (package), pre verziu JUnit 3.8 a skoršie verzie je to balíček junit.framework, pre verzie JUnit 4 a novšie je to už balíček org.junit.

JUnit vo verzii 4, používajúci balík org.junit, sa programuje ako jednoduchá trieda, ktorá nemusí byť odvodená od žiadnej inej triedy (ako tomu bolo vo verzii 3.8). Táto trieda potom obsahuje verejné metódy bez návratovej hodnoty, ktoré majú pred svojím zápisom pridanú anotáciu @Test. Takáto trieda je JUnit frameworkom chápaná ako test, a každá takáto trieda je vykonaná pri testovaní. Celá trieda obsahujúca tieto metódy je chápaná ako test fixture.

Anotácia @Test podporuje dva voliteľné parametre:

- @Test(expected=<Exception>.class), kde <Exception> je názov triedy výnimky, ktorú očakávame, že test vráti, v prípade, že test vráti inú výnimku alebo žiadnu výnimku, je považovaný za zlyhávajúci test,
- @Test(timeout=<milisekundy>), kde nastavíme časové obmedzenie behu testu, ak potom test pobeží dlhšie ako je nastavené, bude vyhlásený za zlyhaný test.

Okrem metód, ktoré sú označené anotáciou @Test a definujú jednotlivé testy, je možné vytvoriť ešte ďalšie metódy s anotáciami:

- @BeforeClass, takáto metóda bude vykonaná ako prvá, pred prvým testom,
- @AfterClass, takáto metóda bude vykonaná ako posledná, po poslednom teste,
- @Before, takáto metóda bude vykonaná pred vykonaním každého jedného testu,

- @After, takáto metóda bude vykonaná po vykonaní každého jedného testu.

Testová metóda sa skladá s programového kódu, ktorý môže iniciovať nejakú situáciu, objekt, premennú a pod., a potom s funkcií `assert*`, ktoré vyhodnocujú testované programové jednotky. Takéto `assert*` funkcie sú napríklad:

- `assertTrue(boolean condition)`, ktorá vyhodnocuje, či `condition` je `True`,
- `assertFalse(boolean condition)`, podobne ako predchádzajúca funkcia, táto očakáva, že `condition` bude `False`,
- `assertEquals(Object expected, Object actual)`, ktorá vyhodnocuje rovnosť očakávanej (`expected`) a aktuálnej (`actual`) hodnoty, kde aktuálnu hodnotu vráti testovaná programová jednotka,
- `assertArrayEquals(Object[] expecteds, Object[] actuals)`, ktorá vyhodnocuje rovnosť dvoch polí,
- atď...

Tieto `assert` funkcie môžu voliteľne na prvom mieste zadať identifikujúci správu ako textový reťazec, ktorá bude zobrazená ak test zlyhá.

Rôzne integrované vývojové prostredia pre Javu, ako je napríklad Eclipse, majú v sebe zabudovaný mechanizmus spúšťania a prehľadného zobrazovania výsledku testov. Ak nejaký test v tomto rozhraní nie je splnený, po dvojkliku na test sa vyznačí `assert`, ktorý nebol splnený.

Okrem `assert*` funkcií môžeme použiť aj `assume*` funkcie, ktoré pri zistení chyby zastavia test ale neoznačia ho ako zlyhaný, ale ako ignorovaný.

### 2.3.2 Smarty 3

Smarty 3 je php template system (php šablónovací systém), ktorý primárne slúži na oddelenie aplikačnej logiky od prezentačného kódu. Smarty generuje výslednú štruktúru dokumentu, prevažne html, z predlohy, ktorá obsahuje špeciálne značky pre smarty. Takéto predlohové súbory, šablóny, majú štruktúru výsledného dokumentu, napr. html, no zvyčajne majú tieto súbory príponu `.tpl`. Smarty spracováva tieto špeciálne značky v kóde a nahrádza ich iným kódom, typicky php kódom.

Smarty používa vlastné značky, typicky zabalené v znakoch `{ a }`. Tieto značky môžu byť buď výrazy, ktoré sa vyhodnotia a ich hodnota sa zobrazí, priradenia, volanie funkcií a pod. Spolu takto riadia tok programu pri prezentácii dát.

Smarty podporuje medzi-pamäť výsledkov, cache, pričom umožňuje v šablónach určiť, ktoré časti budú z cache vyňaté a vždy spracované s aktuálnymi dátami.

Smarty pri procese zobrazovania šablón kontroluje, či existuje platný cache súbor pre želanú



šablónu. Ak existuje, bude zobrazený jeho obsah, v opačnom prípade smarty nájde požadovanú šablónu a vygeneruje z nej cache súbor, ktorý následne zobrazí. Toto platí v prípade, že je zapnuté používanie cache, ale ak nie je, tak sa vždy generuje výstup so šablóny a dát jej poskytnutej. V každom prípade, ak Smarty zistí, že šablóna ešte nie je skompilovaná, prevedie jej obsah na ekvivalentný obsah s tým, že všetky Smarty značky budú vymenené za php kód, ktorý vykoná ich funkcionality.

V procese vývoja aplikácie je Smarty väčšinou nastavené tak, aby v každom prípade kontroloval rozdiel šablón a ich skompilovaných variant a prekompilovával ich podľa potreby. Táto funkcionality Smarty je však spomalujúca, preto sa neodporúča aby bola zapnutá v produkčnom prostredí.

Smarty dovoľuje v šablónach robiť viacero pokročilých vecí, ako je:

- volanie iného súboru so šablónou, pomocou značky `{include}`, táto dokonca vytvára nový scope pre premenné,
- vytvárať si pomocné premenné pomocou značky `{assign}` alebo pomocou výrazu `{ $\$$ premenna = hodnota}`, druhý spôsob dovoľuje priradovať hodnoty aj do polí alebo vlastností objektov,
- rozširovať šablóny, pomocou značky `{extends}`, táto značka sa musí zapísať ako prvá v súbore a určuje iný súbor so šablónou, ktorý obsahuje pomenované bloky značkami `{block} {/block}`, súbor, ktorý túto šablónu rozširuje potom obsahuje rovnako pomenované `{block} {/block}` značky, týmto sa dajú vytvoriť layouty pre šablóny,
- definovať v šablóne funkcie, pomocou značky `{function}`, ktoré sa dajú neskôr volať, tieto funkcie môžu byť rekurzívne, platí, že musia byť vždy definované pred použitím, sú nenahraditeľné pri generovaní navigácií a pod. (keďže navigácia je väčšinou stromová štruktúra),
- dovoľuje aplikovať modifikátory na zobrazovaný obsah premenných, napríklad `{ $\$$ studenti.email|default:"Študent nezadal svoj e-mail."}`, tieto modifikátory môžu byť jak plugin Smarty, tak aj php funkcie, ktoré na prvom mieste dostávajú vstup, na ktorom majú vykonať nejakú modifikáciu podľa parametrov na druhom, treťom a ďalšom mieste, ak je to potrebné, dá sa modifikátor aplikovať aj na výsledok funkcie či textový reťazec, modifikátory sa dajú skladať,
- zobrazíť prehľad dát, ktoré šablóna aktuálne má k sebe pripojené a ďalších informácií, na ktoromkoľvek mieste, pomocou značky `{debug}`.

Samotný objekt Smarty nám dovoľuje robiť napríklad tieto veci:

- priraďovať, dopĺňať či odstraňovať dáta pripojené k šablóne,
- kontrolovať cache pre šablónové súbory, mazať cache,
- dynamicky registrovať modifikátory, funkcie, bloky alebo značky kompilátora (ktoré do skompilovanej šablóny vkladajú obslužný php kód),
- dynamicky odregistrovať spomenuté pluginy,
- zobrazíť spracovanú šablónu na výstupe alebo vrátiť celý výsledok spracovanej šablóny do premennej.

### 2.3.3 TinyMCE

TinyMCE alebo Tiny Moxiecode Content Editor je open source multiplatformový webovo-orientovaný WYSIWYG editor, vytvorený v jazyku JavaScript/HTML vyvíjany spoločnosťou Moxiecode Systems AB. Tento softvérový balík je schopný konvertovať HTML elementy textarea na plnohodnotný textový editor. Samotné TinyMCE je vytvorené tak, aby bolo ľahko nasaditeľné do CMS systémov, ako je Drupal, Joomla! či WordPress, ale aj do akýchkoľvek iných webových aplikácií, ktorých autor sa rozhodne toto riešenie použiť.

Editor podporuje štandardné formátovacie nástroje vo webovom prostredí, ako je hrúbka písma, sklon písma, podčiarkovanie, zoradované a nezoradované zoznamy, rozloženie textu, vkladanie obrázkov alebo videí, odkazov, nezalomiteľných medzier, kopírovanie, vystrihovanie a vkladanie textu a pod.

Editor podporuje používanie rôznych skinov, na prispôsobenie vzhľadu editora vlastnej aplikácii, prípadne dovoľuje upravovať samotné UI editora, pridávať a odoberať funkčné tlačidlá alebo vytvárať vlastné tlačidlá a ich funkcionality, či vytvoriť totálnu konverziu, kedy je možné celý interface editora prepracovať podľa vlastných potrieb.

Editor navyše ponúka API na vytváranie vlastných pluginov a vývojári TinyMCE sami ponúkajú niekoľko prepracovaných pluginov na prácu so súbormi alebo obrázkami, tieto sú však komerčné.

Editor je do stránky možné integrovať pomocou obvyčajného JavaScriptu, ale aj pomocou knižnice JQuery, pre ktorý má vytvorené pluginy na integráciu.

### 2.3.4 JQuery / JQueryUI

JQuery je knižnica pre JavaScript navrhnutá tak, aby zjednodušovala skriptovanie HTML stránok na klientskej strane. Jej motto, píš menej, urob viac, vystihuje silu tohto nástroja. JQuery je open source projektom, pôvodným autorom je John Resig.

Syntax JQuery je navrhnutá tak, aby zjednodušovala navigáciu v dokumente, výber DOM elementov, vytváranie animácií, obsluhu udalostí a vytváranie Ajax aplikácií. Rovnako vývojárom umožňuje písať vlastné pluginy a teda im dovoľuje vytvárať abstrakciu nízko-úrovňovej

interakcie a animácií.

Niektoré korporácie, ako je Microsoft, zabudovávajú JQuery do svojich produktov, konkrétne Microsoft pridal JQuery do Visual Studia pre ASP.NET AJAX framework a ASP.NET MVC framework.

JQuery teda prináša tieto výhody:

- výber DOM elementov pomocou multi-browser systému pre selektory Sizzle,
- traverzovanie a modifikovanie DOM s podporou pre CSS 1 až 3,
- obsluha udalostí,
- efekty a animácie,
- AJAX,
- rozšíriteľnosť pluginmi,
- multi-browser podpora.

JQueryUI je JavaScript knižnica, ktorá poskytuje abstrakciu pre nízko-úrovňovú interakciu a animácie, pokročilé efekty a vysoko-úrovňové widgety, ktorá je postavená na knižnici JQuery. Používa sa na vytvorenie interaktívnych webových aplikácií a ponúka celý rad widgetov a efektov:

- akordeón – harmonikové prepínanie obsahu so zachovaním výšky,
- automatické dokončovanie vkladania textu,
- tlačidlá, klasické aj prepínateľné, možno doplniť o ikony,
- kalendár na výber dátumu,
- dialógové okná,
- viac-úrovňové menu,
- progressbar,
- posuvníky s jazdcom,
- záložky,
- nástrojové tipy,
- a rôzne ďalšie widgety, efekty a pod.

### 2.3.5 CodeIgniter

CodeIgniter je open source framework pre PHP webové aplikácie založený na architektúre Model-View-Controller MVC. Poskytuje množstvo knižníc, ktoré pomáhajú rýchlejšie vytvárať dynamické webové aplikácie. Jadro frameworku je založené na PHP verzii 4.1, čo však nebráni písať v ňom aplikácie využívajúce výhody PHP 5 či framework rozširovať o podporu tejto verzie

PHP.

Framework má oddelené jadro systému s všetkými dodávanými knižnicami, helpermi, databázovými adaptérmí a jazykovými súbormi od samotnej aplikácie. Typicky v koreňovom adresári aplikácie postavenej na CodeIgniter sa nachádza adresár system (so súbormi jadra, knižnicami, atď) a adresár application (obsahujúci radiče, pohľady, modely, konfiguračné súbory, vlastné knižnice a pod.). Súbor index.php v koreňovom adresári je potom jediný vstupný bod pre aplikáciu, ktorý spúšťa jadro, a to na základe konfigurácie routeru obsluhuje užívateľské dopyty volaním príslušných radičov.

Vďaka architektúre MVC sa aplikácia napísaná v tomto frameworku skladá z troch hlavných na sebe nezávislých komponentov:

- radič (controller) je komponent, ktorý prijíma dopyty, spracúva ich a vráti odpoveď,
- model je komponent, ktorý obsluhuje dáta, bežne v databáze ale aj v súborovom systéme,
- pohľad (view) je komponent, ktorý sa stará o prezentáciu dát používateľovi.

CodeIgniter vývojárom prináša podporu pre viaceré databázy, ako sú MySQL(i), PostgreSQL, MsSQL, SQLite, Oracle, Cubrid a PDO. Použitím ActiveRecord je možné aplikáciu prevádzkovať na ktorejkoľvek podporovanej databáze. Ďalej prináša knižnice na prácu so súbormi, formulármi, obrázkami, sessions, xml-rpc, zip, jazyky či jednoduchý template parser.

### 2.3.6 DataMapper ORM

DataMapper ORM je knižnica pre objektovo relačné mapovanie vyvinutá pre CodeIgniter framework. Umožňuje jednoduché vytvorenie databázových modelov pre konkrétne tabuľky a definovanie relačných vzťahov medzi týmito modelmi (tabuľkami v databáze).

Hlavnými znakmi tejto knižnice je jednoduchosť jej nastavenia a použitia, lenivé načítanie relačných objektov (objekty v relácii sa načítavajú pri prístupe k nim), automatické manažovanie integrity relácií, relácie typu jeden-jeden, jeden-viac a viac-viac, výber dát v štýle Active Record ako ho používa CodeIgniter.

DataMapper ORM je rozšírenou verziou DataMapper-u, ktorá navyše pridáva cudzie kľúče priamo v tabuľkách pre relácie typu jeden-jeden a jeden-viac, možnosť zobrazovať, upravovať a dopytovať sa pomocou extra stĺpcov v relačných join tabuľkách, možnosť pripojiť dáta zo singulárnych relačných objektov, možnosť dopytovať sa a pripájať dáta z hlbších relácií (relácia cez viac tabuliek), možnosť mať viacero relácií medzi tými istými objektami, možnosť rozširovať DataMapper ORM pomocou rozšírení, možnosť používať subdopyty a SQL funkcie, nastavovanie východzieho zoradenia, ukladanie objektov s existujúcim ID, zoskupovanie dopytov atď...

### 2.3.7 GeSHi

GeSHi alebo Generic Syntax Highlighter je free softvér umožňujúci zvýrazňovanie syntaxe zdrojového kódu veľkého množstva programovacích či značkových jazykov. GeSHi je napísaný v PHP, dostupný je samostatne či ako add-on pre rôzne webové aplikácie, ako MediaWiki, phpBB, Mambo atď.

GeSHi podporuje zvýrazňovanie syntaxe pre cca. 200 programovacích a značkových jazykov, pričom môže byť rozšírený o ďalšie jazyky dodaním vlastných definícií. Zdrojový kód, ktorý je spracovaný GeSHi je prevedený na html a zvýraznený pomocou kaskádových štýlov, výstup je kompatibilný s XHTML 1.1 a CSS úrovne 2. Zaujímavou funkcionalitou je aj generovanie liniek v zvýraznenom kóde na dokumentáciu pri niektorých programovacích jazykoch.

### 2.3.8 Plupload

Plupload je softvér integrovateľný do webovej stránky na klientskej a serverovej strane, ktorý umožňuje asynchrónne uploadovanie viacerých súborov v rade. Z väčšej časti je napísaný v JavaScripte a do webovej stránky je integrovateľný aj pomocou jQuery(UI). Podporuje viacero runtime prostredí, pomocou ktorých môže uploadovať súbory.

Plupload dokáže uploadovať dáta pomocou prostredí Flash, Gears, HTML 5, Silverlight, BrowserPlus a HTML 4. Okrem HTML 5 ostatné prostredia podporujú takmer všetky operácie a možnosti, ktoré Plupload poskytuje, a to sú:

- delenie súboru do chunkov,
- drag/drop pridávanie súborov do zoznamu súborov na upload,
- automatická zmena rozmerov obrázkov vo formáte PNG a JPEG,
- filtrovanie typov,
- streamovaný upload,
- multipart upload,
- obmedzovanie veľkosti prenášaného súboru,
- zobrazovanie postupu uploadu,
- odosielanie vlastných hlavičiek.

HTML 4 runtime prostredie podporuje z tohto len multipart uploadovanie.

### 2.3.9 FancyBox 2

Fancybox je plugin pre jQuery, ktorý dokáže otvárať obrázky, HTML elementy, SWF súbory, iframe alebo ajax požiadavky v špecificky vytvorenom viewporte vytvorenom ponad obsah webovej stránky. Vzhľad tohto viewportu a jeho správanie je upraviteľné cez nastavenia a CSS.

### 3 Zhodnotenie východiskovej práce

Počas písania tejto podkapitoly som analyzoval zdrojové kódy pôvodného systému LaMSfET, ako aj funkcionality systému.

Peter Jurčo, autor pôvodného systému, mal peknú myšlienku urobiť systém modulárny, čo by malo prispieť k jeho flexibilita a mal by sa dať takto jednoduchšie upravovať. Keď som však dostal od môjho školiteľa pôvodné zdrojové kódy a databázovú štruktúru, zistil som, že je tu celý rad problémov.

Po niekoľkých hodinách trápenia sa z rozbehnutím systému, čoho príčina bola množstvo chýb vrátených php procesorom a rovnako tak snaženie sa opraviť tieto chyby v častiach kódu, ktoré síce boli spúšťané, no nemali žiaden dopad na funkčnosť, som zistil, že celé zloženie systému je nutné nastaviť v príslušných databázových tabuľkách, každú stránku, každý modul. Keďže som pôvodne mal k dispozícii iba štruktúru databázy, aj potom, ako sa mi podarilo systém rozbehnúť, som s nim prakticky nič nemohol robiť.

Systém je teda rozdelený do v databáze vytvorenej mapy stránok a k nim náležitých boxov (v podstate pod-modulov), tieto stránky majú jak ID číslo, tak aj krátky unikátny názov. Zaujímavé však je, že systém má v sebe hard-kódované názvy stránok, ktoré vynucuje na zobrazenie, ako je napr. login, task\_list a pod. Nikde nie je konfiguračná položka, ktorá by určovala východziu stránku ani stránku pre prihlásenie. Systém ani nehľadá na to, že by databáza mohla byť prázdna a tak používateľ dostane len rad nezrozumiteľných chybových hlásení, ak sú príslušné databázové tabuľky prázdne.

Situácia s databázou sa však vyriešila a po vytvorení účtu učiteľa / administrátora, som sa dostal do systému. Tak ako počas snahy systém rozbehnúť na mojom vývojovom webserveri (Apache 2.2, PHP 5.4.4), zobrazovalo sa množstvo chybových hlásení úrovne notice prípadne úrovne strict standard, ktoré efektívne znemožňujú prácu s veľkým množstvom funkcií systému a tie funkcie, ktoré akoby zázrakom ostávajú funkčné, sú týmito chybami rovnako negatívne postihnuté. Z tohto usudzujem, že autor systému bol v tom čase nováčikom v PHP a celý systém vytváral v produkčnom nastavení systému (žiadne chybové hlásenia). Jeho server a rovnako tak aj server capek.ii.fmph.uniba.sk, kde je systém momentálne nasadený, sú teda nutne nastavené tak, aby tento softvér dokázali spúšťať a pritom ignorovali jeho chyby v zdrojovom kóde.

Okrem toho, že sa systém spolieha na benevolenciu PHP procesora a dúfa v najlepšie (že PHP automaticky vytvorí neexistujúce premenné, indexy v poliach, nevšimne si rozdieli v interface metód potomkov tried a pod.), systém je vytvorený takmer bez akejkoľvek technológie tretej

strany, až na použitie Smarty template engine. Fakt, že neexistuje ani dokumentácia vygenerovaná z kódu efektívne brzdí akúkoľvek snahu do systému preniknúť. Naviac systém je plný mŕtveho kódu a komentárov všade tam, kde nemajú byť, pričom jediné komentáre, ktoré chýbajú, sú dokumentačné. Vygenerovať z kódu zrozumiteľnú dokumentáciu pomocou PHPDocumentor-u je takmer nemožné.

Potom, čo som vypol na lokálnom serveri všetky hlásenia chýb som dúfal, že budem schopný systém LaMSfET otestovať aj po stránke jeho funkcionality. Bohužiaľ, ani potom nebolo možné systém používať plnohodnotne. Administrátorská stránka, ktorá má slúžiť na vytváranie nových zostáv úloh, nebola schopná fungovať ani teraz. Zostavy úloh sú pre systém kľúčovým elementom, nakoľko práve do zostáv sa v systéme zaraďujú jednotlivé úlohy a bez zostáv prakticky nie je možné pracovať s väčšinou funkcionality systému.

Nakoniec ostala len možnosť spoločne so školiteľom preskúmať aktívne používanú nasadenú verziu systému a zosumarizovať spoločne zoznam funkčných chýb:

- Stránka so zoznamom úloh;
  - stránkovanie úloh je vytvorené pomocou záložiek (tabov) nad tabuľkou so zoznamom úloh, bohužiaľ, pri návrhu systému na dlhodobé uskladnenie stoviek úloh autor pozabudol, čo robí a výsledkom je presah záložiek cez layout stránky pri väčšom počte ako 500 uložených úloh v databáze, záložky sa fakticky rozťahujú až za vertikálnu hranicu pravého okraja obrazovky a treba skrolovať doprava kvôli nájdeniu konkrétnej stránky,
  - ak sa použije rýchle vyhľadávanie úlohy podľa názvu, nie je možné opätovne vyhľadávať cez vstupné pole na stránke s výsledkom, pretože používateľ dostane php chybu úrovne fatal error,
  - vo filtri úloh podľa tagov je rozbaľovací zoznam týchto tagov, no vnútri tohto zoznamu nie je vidno hierarchiu, tj. podradené tagy nie sú nijak odsadené aby bolo vidno, že sú podradené svojmu nadradenému tagu,
  - existuje tu filter úloh podľa ich použitia resp. nepoužitia v špecifikovanom období, no bohužiaľ nefunguje a naviac samotná jeho funkcionality je používateľom systému nejasná,

- Stránka na zadávanie úloh (vytvorenie novej úlohy);
  - editor obsahuje automatický náhľad textu úlohy, bohužiaľ tento text je zobrazovaný v širšom boxe ako reálna verzia textu, ktorú uvidí študent,
  - neexistuje akákoľvek podpora pre písanie textu formou WYSIWYG editora, všetko formátovanie je nutné písať ručne,
  - SESSION údaje v systéme expirujú príliš rýchlo, ak sa toto stane pri písaní úlohy, všetok rozpísaný text sa pri ukladaní stratí,
- Stránka na úpravu existujúcej úlohy;
  - pri označovaní úloh tagmi, ak učiteľ označí úlohu nejakým tagom a tento tag má v hierarchickej štruktúre svoje podradené tagy, nie je už možné k úlohe priradiť niektorý z týchto podradených tagov,
  - úlohy podporujú vstupno-výstupné testovanie študentských riešení, avšak chýba možnosť vstupu do programu so štandardného vstupu ako aj porovnanie výstupu programu do súboru,
- Stránka na importovanie úloh;
  - systém by mal podporovať importovanie textu úlohy z PDF súboru, no implementácia pri pokuse o zobrazenie textu PDF súboru skončí s chybou PHP úrovne fatal error,
  - import úloh z HTML funguje, bohužiaľ pri importe sa nezachová formátovanie, do textu úlohy sa skopíruje plain text,
- Stránka so zoznamom zostáv;
  - zoznam vôbec neukazuje k akej skupine v danom kurze zostava patrí,
  - ak je vybratý kurz cez filter, zbytočne sa zobrazuje v zozname,
  - zobrazujú sa iba dva z troch časových údajov v zostave úloh, pričom ten, ktorý zobrazený nie je, je najdôležitejší (ide o časový limit odovzdávania riešení),
  - v zozname sa zobrazuje tlačidlo s názvom „Strhnúť body“, ktoré má strhnúť body všetkým študentom, ktorý neodovzdali vypracované riešenie danej zostavy, bohužiaľ tlačidlo nemá žiadnu formu potvrdenia akcie a navyše vôbec nezisťuje, či už vypršal čas na odovzdanie riešení,
  - v zozname zostalo z pôvodnej implementácie tlačidlo na zverejnenie alebo nezverejnenie zostavy, ktoré vyplýva z pôvodného návrhu zamietnutého ešte v priebehu vytvárania pôvodného systému ako bakalárskej práce a má spornú



funkcionalitu,

- tlačidlo na stiahnutie všetkých odovzdaných riešení so zostavy nepodlieha zabezpečeniu prístupu, teda ktokoľvek so znalosťou url linky na akciu tohto tlačidla si vie stiahnuť všetky riešenia úloh bez toho, aby bol v systéme riadne autentifikovaný,
- Stránka na hodnotenie riešení;
  - ak študent odovzdá nové riešenie zadania, potom čo mu bolo udelené hodnotenie, učiteľ o tom nie je nijako informovaný,
  - ak učiteľ použije hromadné hodnotenie riešení, je tu určitá pravdepodobnosť, že systém vytvorí pre niektorého so študentov duplicitný používateľský účet, tento problém pretrváva po celý čas odkedy bol systém nasadený,
  - pri hodnotení jedného riešenia sa súbory riešenia zo zip archívu vždy rozbaľujú do toho istého jediného adresára, čo znemožňuje paralelné hodnotenie riešení viacerým učiteľom v jednu chvíľu.

Okrem týchto problémov boli identifikované ešte ďalšie problémy, ktoré nemajú len lokálny charakter:

- neexistuje žiadna možnosť ako zálohovať súbory a databázu systému pomocou CRON skriptu,
- príliš krátka expirácia SESSION spôsobuje časté odhlasovanie používateľov so systému,
- je veľkým problémom implementovať do systému novú funkcionalitu vplyvom zlého návrhu architektúry systému,
- všetky filtrovacie formuláre k zoznamom (či už úloh, zostáv a pod.) si nezapamätajú svoje nastavenie a sú resetované do pôvodného stavu pri obnovení stránky alebo vrátení sa na stránku s filtrom,
- architektúra a návrh systému vo všeobecnosti nepočíta s behom viacerých kurzov naraz v jednom semestri, pričom tieto kurzy by mali využívať na svoju podporu výuky tento systém.

Ako vidno s predchádzajúceho textu, systém má značné problémy s návrhom svojej architektúry a tiež jej implementáciou, taktiež s implementáciou drobnej funkcionality. Navyše počas štyroch rokov nasadenia systému bolo doň pridaných mnoho funkčných prvkov spôsobom, ktorý obchádza architektúru systému aby bolo vôbec možné tieto prvky implementovať.

Z dôvodom hore uvedených som nutne došiel k záveru, že akokoľvek pokračovať v práci na pôvodnom systéme je kontraproduktívne, pretože aj po odstránení radu menších či stredných

problémov bude architektúra systému stále neprehľadná a systém by sa len veľmi ťažko adaptoval na podporu výuky viacerých kurzov naraz. Preto vrámci tejto bakalárskej práce vytvorím návrh nového systému, so základom v architektúre Model-View-Controller, ktorý bude implementovať svoju funkcionality pomocou voľných technológií tretích strán, nakoľko k týmto technológiám existuje online dokumentácia, čo by malo prispieť k zjednodušeniu budúceho vývoja nového systému po jeho nasadení.

## 4 Požiadavky na softvér

V tejto kapitole sformujeme požiadavky na softvérový systém L.I.S.T., ktoré sú nasledovné:

- systém musí podporovať paralelne bežiacie výukové kurzy ako aj umožniť paralelné hodnotenie odovzdaných riešení,
- systém bude používaný dvoma úrovňami používateľov, učiteľom a študentom. Tieto dve úrovne musia byť dve nezávislé entity a každá táto entita bude mať svoje vlastné rozhranie, v ktorom bude pracovať,
- systém musí umožniť vytváranie a správu úloh a príloh k úlohám (prílohy vo forme súborov uložených na serveri), je tiež nutné, aby boli úlohy kategorizované a aby sa dali filtrovať podľa týchto kategórií,
- kategórie úloh musia mať hierarchickú štruktúru,
- systém musí umožňovať vytvárať zostavy úloh, alebo inak povedané zadania pre študentov, ktoré budú pozostávať z viacerých úloh, tak isto jedna úloha môže patriť k viacerým zadaniam,
- systém by mal umožniť klonovať úlohy aj zadania úloh, (napríklad kvôli zmenám, ktoré chceme aby sa prejavili lokálne),
- systém by mal podporovať viacjazyčné používateľské rozhranie a mal by umožňovať prekladať obsah (úlohy, zostavy atď ...) do viacerých jazykov,
- systém by mal umožňovať vytvárať výučbové kurzy, ku kurzom vytvárať skupiny a ku skupinám priradovať miestnosti, v ktorých prebieha výučba pre danú skupinu,
- systém musí umožniť študentovi prihlásiť sa na kurz a vybrať si študijnú skupinu, učiteľovi musí umožniť spravovať priradenie študentov do skupín a riadiť ich účasť v kurzoch,
- systém by mal študentom dovoliť odovzdať svoje riešenie zverejnených zadaní (zostáv úloh) online a učiteľovi by mal umožniť tieto úlohy kontrolovať a hodnotiť,
- systém by mal umožňovať vytvárať k jednotlivým úlohám unit testy, vygenerovať z nich spustiteľný kód a otestovať ním odovzdané riešenie so zobrazením výsledkov.

## 5 Návrh riešenia

V tejto kapitole bude rozobratá časť návrhu systému LIST – Long-term Internet Storage of Tasks. Viac k návrhu systému je možné vidieť v samostatnom dokumente, tvoriacom prílohu tejto bakalárskej práce.

### 5.1 Používatelia systému

Softvér umožní pracovať so systémom dvom úrovniam používateľov, študentom a učiteľom. Každá úroveň používateľa bude mať na prácu so systémom vyhradenú vlastnú časť systému.

Študent bude vystupovať v roli bežného používateľa, ktorí má minimálne možnosti, ohraničené na prihlasovanie sa do študentského rozhrania, prihlasovanie sa na kurzy, výber skupiny v kurzoch, prezeranie zadaní úloh, odovzdávanie vlastného riešenia, prezeranie hodnotenia odovzdaných riešení a úpravu vlastného používateľského účtu.

Učiteľ bude vystupovať v roli administrátora systému, a bude sa vedieť so svojím účtom prihlasovať do administrácie, kde bude vedieť meniť takmer všetky parametre obsahu. Všetci učitelia sú si rovný, všetci budú mať prístup ku tým istým možnostiam správy systému.

Používateľské účty budú mať minimálne nastavenia, ako prihlasovacie meno sa bude používať e-mailová adresa. Ak bude učiteľ chcieť, môže si vytvoriť aj študentský účet s rovnakou e-mailovou adresou ako používa na učiteľskom účte, avšak rámci tabuliek študentov a učiteľov budú e-mailové adresy unikátne.

### 5.2 Jazykový preklad systému

Systém má byť vytvorený tak, aby umožňoval preklad rozhrania a obsahu do viacerých jazykov. Za týmto účelom budú v systéme prítomné tri riešenia, ako tohto efektu dosiahnuť.

Prvé riešenie bude predpokladať existenciu súborov s definíciou textu v danom jazyku priradenou k nejakému kľúčovému slovu – konštante. Toto riešenie je orientované hlavne na preklad používateľského rozhrania. Takýto textový súbor sa načíta a páry konštanta – text sa uložia do spoločného zásobníka, odkiaľ sa budú dať získať pre potreby zobrazenia na stránke systému.

Druhé riešenie spočíva v existencii tabuľky prekladov v databáze, ktorá bude obsahovať informácie jednak o zvolenom názve konštanty, tak aj o jazyku, ktorý je ňou pokrytý a konečne text v danom jazyku. Pre každý takýto preklad sa pripojí prefix ku konštante a všetky takto poopravené páry konštanta – text sa vložia do rovnakého zásobníka ako ho rieši prvé riešenie, čo znamená, že prístup k takémuto prekladu bude rovnaký ako v prvom riešení.

Tretie riešenie bude počítat s existenciou inej tabuľky, ktorá bude realizovať prekrytie

pôvodného textu textom v inom jazyku. Všetok bežný textový obsah sa bude ukladať akoby v nedefinovanom východizme jazyku a k vybraným položkám bude možné definovať prekrytie tohto textu v konkrétnom jazyku. Tabuľka týchto prekrytí bude obsahovať informáciu o texte prekrytia, ktorý spadá do nejakej konkrétnej tabuľky, špecifikovanej hodnotou jej ID, informáciou o tom, v ktorom stĺpci tejto tabuľky má byť text prekrytý a o aký jazyk sa jedná. Preklad realizovaný pomocou prekrytia bude následne fungovať tak, že sa z pôvodnej tabuľky načíta záznam a potom sa načítajú príslušné texty prekrytí k príslušným stĺpcom. Tieto prekrytia je nutné realizovať optimálne, aby bolo možné pre viac záznamov jedným volaním databázového dopytu inicializovať všetky potrebné prekrytia pre konkrétny jazyk.

Druhé a tretie riešenie sa môže dopĺňať. Z návrhu vyplýva, že sa môže aj kombinovať, teda v prekrytí by mohla byť definovaná konštanta, ktorej text sa má zobrazíť, avšak toto správanie je z pohľadu celkovej funkčnosti nevyužiteľné. Teda v systéme sa bude používať v konkrétnej tabuľke a jej konkrétnom stĺpci preklad buď pomocou druhého alebo tretieho riešenia.

### 5.3 Úlohy a zostavy úloh

Systém ma umožniť vytvárať, upravovať a uchovávať čiastkové úlohy (ďalej len úlohy), ktoré pozostávajú z názvu úlohy, textu úlohy, jej kategorizácie vrámci editovateľnej hierarchickej štruktúry kategórií a súborov k úlohe.

Každá úloha musí implementovať jazykové prekrytie, kvôli lokalizácii vlastného textu zadania do viacerých jazykov, ktoré systém podporuje.

Úloha môže aj nemusí mať vzorové riešenia, tie však nebudú viditeľné pre študentov, budú slúžiť len učiteľom.

Samotné úlohy nie sú viditeľné pre študentov. Na to aby bola niektorá úloha zobrazená študentovi, musí byť zaradená do zostavy úloh (v tomto texte tiež označované ako zadanie). Zostava úloh je kombináciou rôznych úloh, ktorú bude zostavovať učiteľ a ako taká bude aj určovať ktorým študentom sa zobrazí a kedy.

Na nastavenie zobrazovania bude mať zostava niekoľko volieb. Každá zostava bude musieť byť zaradená do niektorého vyučovacieho kurzu a bude musieť byť označená nejakým typom zostavy (ten bude aj hovoriť o tom, či študent bude alebo nebude odovzdávať súbory s riešením pre túto zostavu).

Voliteľne sa bude dať nastaviť skupina, v rámci zvoleného kurzu, len pre ktorú sa má zostava zobrazovať.

Bude možné zadať časovú informáciu, dátum a čas, kedy sa má začať zostava zobrazovať. Rovnakým spôsobom sa bude dať nastaviť dátum a čas ukončenia odosielania súborov s riešením.

Ak tieto časové informácie nebudú zadané, zostava sa jednoducho zobrazí ihneď a odosielanie súborov nebude nijako časovo limitované.

Ak to bude potrebné, bude môcť učiteľ zvoliť miestnosť v ktorej prebieha vyučovanie predtým zvolenej skupiny, čo spôsobí, že sa zostava zobrazí v momente, keď začne vyučovanie v čase definovanom v miestnosti. Tu sa bude brať do úvahy čas začiatku publikovania, ktorým sa vymedzí čas časový bod, od ktorého sa má prihliadať na splnenie času a dňa v týždni definovanom v miestnosti. To znamená, že ak má skupina miestnosť, napríklad H3, v ktorej začína tejto skupine výučba v utorok o 9:50, túto miestnosť učiteľ zvolí a ako čas začiatku publikovania nastaví 22.9.2014 00:00:00, pondelok ráno, k zverejneniu zostavy príde v utorok 23.9.2014 o 09:50:00. Samozrejme, že ak nebude nastavený žiaden čas zverejnenia zostavy, výber miestnosti nijako neovplyvní, kedy sa zostava zobrazí.

Ak zostava raz splní podmienku svojho zobrazenia, ostane zobrazená nastálo. Bude tu však ešte jedna možnosť a to prepínač, ktorý bude hovoriť, či je alebo nie je možné zostavu publikovať. Ak tento prepínač učiteľ vypne, aj predtým zverejnená zostava sa zverejňovať prestane.

Keďže je občas nutné pozmeniť zadanie úlohy alebo tú istú zostavu zadať napríklad inej skupine, bude možné zostavy aj úlohy klonovať. To spôsobí, že sa celý obsah a súbory pôvodnej úlohy skopírujú a uložia ako nová úloha a v prípade klonovania zostavy sa vytvorí nová zostava s rovnakými úlohami (skopíruje sa zostava a relácie k úlohám), ale v kópii zostavy sa nastaví automaticky jej nezverejňovanie. Učiteľ tak bude môcť bezpečne nastaviť zverejňovanie kópie a študent neuvidí duplikát zostavy v svojom zozname zadání.

## 5.4 Unit testy

K jednotlivým úlohám bude možné vytvoriť unit testy. Na vytváranie unit testov bude existovať abstraktný adaptér a z neho odvodené inštancie pre unit testy pre rôzne programovacie jazyky.

Abstraktný adaptér bude viac-menej interface, ktorý budú od neho odvodené triedy pre jednotlivé programovacie jazyky implementovať. Takéto riešenie by malo zjednotiť správu unit testov, keďže pre všetky programovacie jazyky bude návrh takmer rovnaký.

Spoločným menovateľom adaptérov pre unit testy bude podobný editor unit testov v administrácii, kompletizácia unit testov do skompilovateľného kódu a spúšťanie unit testov pre jednotlivé súbory riešení.

Samotný návrh editora pozostáva z troch tabuliek v databáze.

Prvá tabuľka unit testov bude popisovať ku ktorej úlohe unit testy patria a aký programovací jazyk (adaptér unit testov) používajú.

Druhá tabuľka bude obsahovať informácie o menách jednotlivých test case a bude odkazovať na

prvú tabuľku. Teda unit testy pre jednu úlohu budú môcť mať viacero test case.

Tretia tabuľka bude popisovať testovacie funkcie vrámci test case. Bude obsahovať odkaz na druhú tabuľku, bude mať unikátne meno testovacej funkcie vrámci test case, kód samotnej funkcie a typ tejto funkcie (napríklad anotácia funkcie).

K jednotlivým unit testom ako celku bude možné pridávať súbory, či už dátové súbory potrebné pre testy alebo súbory s zdrojovým kódom, ktoré sú vyžadované pri kompilácii testov.

Keďže unit testy budú viazané na samostatné úlohy, pri testovaní zostavy úloh sa budú vždy spúšťať pre každú jednu členskú úlohu zostavy samostatne s tým istým študentským riešením.

Z rovnakých dôvodov návaznosti na samostatné úlohy sa budú aj unit testy klonovať spolu s úlohou (ak sa učiteľ rozhodne klonovať úlohu, ktorá obsahuje aj unit testy).

## 5.5 Zostavovanie unit testov

Zostavovanie unit testov bude riešené odvodeným adaptérom pre unit testy pre každý programovací jazyk samostatne, ale mal by zohľadňovať tieto princípy:

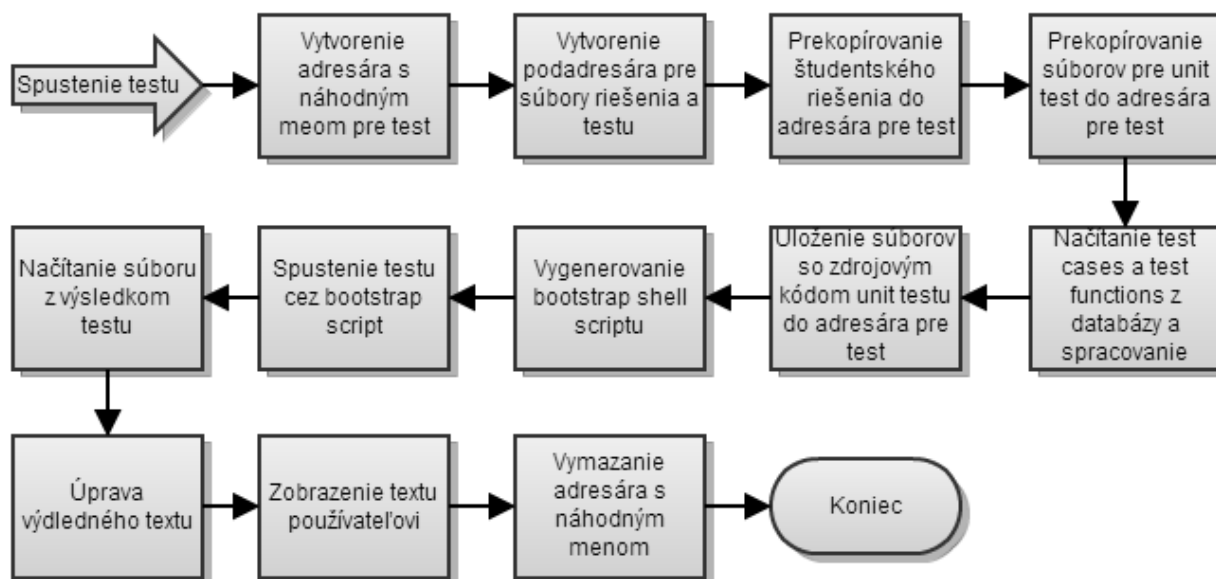
1. pred kompiláciou unit testov sa vytvorí adresár s náhodným menom, v ktorom sa vytvorí podadresár pre zostavované unit testy,
2. do podadresára sa rozbalí študentské riešenie zadania,
3. do podadresára sa skopírujú dodatočné súbory k testom, ak je treba, prepíšu existujúce súbory,
4. načítajú sa `test_cases` a `test_functions`, vygeneruje sa obsah súborov s testami a uloží sa do podadresára,
5. v nadradenom adresári s náhodným menom sa vytvorí bootstrap shell script, ktorý bude spúšťať test.

## 5.6 Spustenie unit testu

Po zostavení testu a vytvorení bootstrap shell scriptu spustí tento shell script. Počas behu skriptu by mal byť vygenerovaný súbor s výstupom testovania, ktorý následne adaptér načíta. Adaptér môže voliteľne implementovať lexikálny analyzátor, ktorý môže nejak pozmeniť text načítaný so súboru (napr. správy o chybách označí červenou farbou, aby sa ľahšie identifikovali). Následne sa text zobrazí používateľovi, ktorý spustil test.

Keďže študent bude spúšťať test celkovo pre celú zostavu, načíta sa zoznam unit testov pre každú úlohu, v náhodnom poradí, a postupne sa každý test vybuduje a spustí.

Po dokončení testu sa vymaže príslušný adresár, v ktorom sa nachádzajú súbory a bootstrap shell script k testu.



1. Obrázok: Postup vygenerovania a spustenia unit testov.

## 5.7 Zálohovanie a obnova systému

Keďže systém má dlhodobo skladovať úlohy, musí byť schopný dáta o úlohách zálohovať a obnovovať.

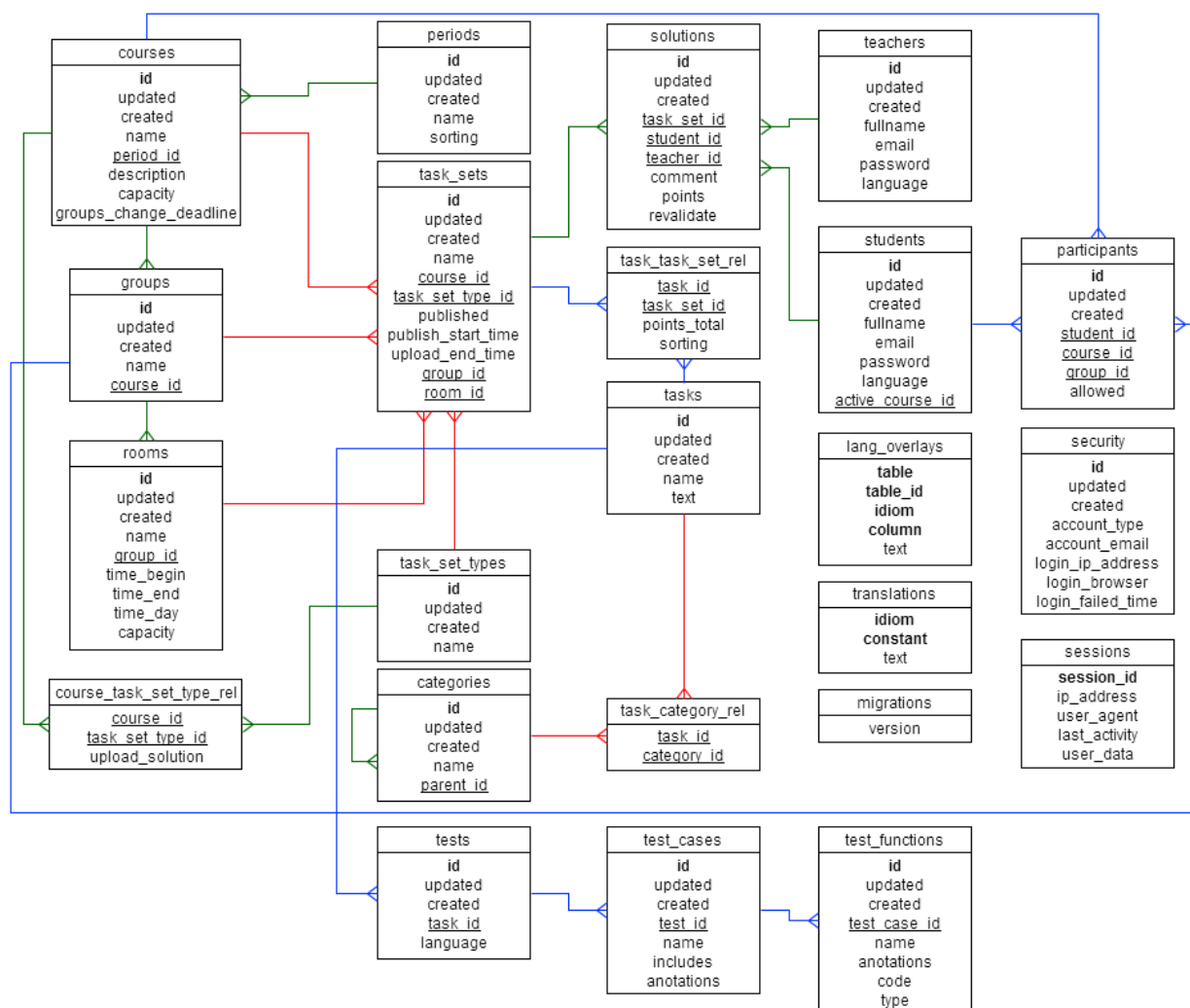
Aby bolo možné toto dosiahnuť, bude systém obsahovať zálohovací podsystém, ktorý bude prístupný z príkazového riadku. Pred spustením zálohy sa zamkne zbytok systému, aby nebolo možné meniť dáta z administrácie alebo študentského rozhrania, čiže záloha bude konzistentná. Zálohované budú všetky súbory a databáza a výsledok bude uložený v ZIP archíve. Po dokončení a zapísaní archívu na disk bude systém odblokovaný.

Obnovovanie systému bude fungovať rovnako z príkazového riadku. Pri obnovovaní systému sa zadá zdrojový ZIP archív, zamkne sa systém a obnovia sa všetky súbory do pôvodného stavu a tiež databáza. Obnova zmaže pôvodné súbory a celú databázovú štruktúru.

Riešenie zálohovania pomocou príkazového riadku dovoľí vykonávať zálohu pomocou CRON skriptu na serveri.



## 5.8 Dátová štruktúra databázy



2. Obrázok: Databázová štruktúra

V zjednodušenej schéme zapojenia databázových tabuliek do relácií platia tieto pravidlá:

- na hrubo označený text je časťou primárneho kľúča tabuľky,
- podčiarknutý text je cudzím kľúčom do inej tabuľky,
- názvy tabuliek končiace sufixom \_rel označujú JOIN tabuľky, nebudú v zdrojovom kóde inštanciované ako objekty modelov.

Päť tabuliek, migrations, translations, lang\_overlays, security a sessions sú pomocné tabuľky a tabuľky obsahujúce jazykové preklady, nie sú v žiadnej priamej relácii s inými tabuľkami.

Táto štruktúra databázy je predmetom zmien vo vývojovom procese a nie je finálnou štruktúrou.

## 5.9 Dátová štruktúra súborového systému


Všetky súbory, ktoré sa týkajú priamo aj nepriamo obsahu stránky generovanej systémom budú začlenené do dvoch hlavných adresárov, public a private.

Do adresára public sa budú umiestňovať všetky súbory verejnej povahy, ako sú javascript, css, súbory obrázkov k používateľskému prostrediu alebo iné priamo zobrazované súbory (napr. flash).

Adresár private bude obsahovať chránené súbory, ako sú súbory k úlohám, študentské odovzdané riešenia zostáv úloh (zadaní) alebo súbory k unit testom. K týmto súborom sa bude z vonku dať pristupovať len cez volania php skriptov, ktoré vygenerujú príslušné hlavičky a vynútia stiahnutie súboru.

Dôvodom tohto rozdelenia je snaha zabrániť neautorizovanému prístupu k súborom, ktoré by mohli poskytnúť študentom výhodu (napríklad by mohli z týchto súborov vyčítať riešenie úloh a pod.).

## 5.10 Používateľské rozhranie pre učiteľov (administrácia)



The image shows a screenshot of a graphical user interface (GUI) window. The window has a title bar with standard minimize, maximize, and close buttons. Inside the window, there is a centered dialog box titled "Prihlásenie učiteľa". The dialog box contains two input fields: "E-mail:" and "Heslo:". Below these fields is a button labeled "Prihlásiť sa".

Obrázok 3: Dialógové okno pre prihlásenie učiteľa

Na obrázku 3 môžeme vidieť návrh rozhrania pre prihlásenie učiteľa do administrácie systému. Ide o jednoduchý formulár, v ktorom učiteľ zadá svoju e-mailovú adresu a heslo. Ak budú zadané údaje správne, učiteľ bude prihlásený do administrácie a bude pokračovať buď na východziu stránku, alebo na tú stránku, ktorú mal otvorenú posledne a pri prístupe k nej systém zistil, že platnosť jeho SESSION dát vypršala.

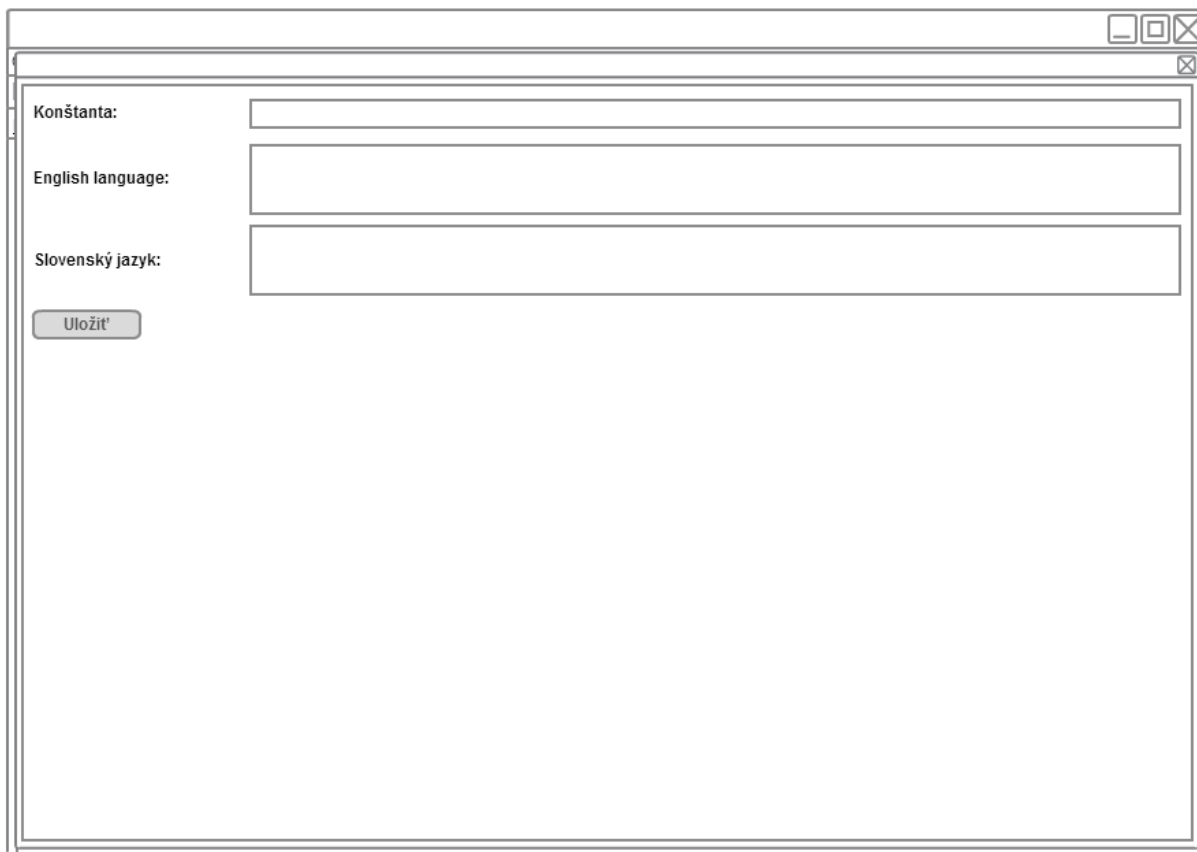
Konštanta	English language	Slovenský jazyk	Ovládanie
konštanta	<input type="text" value="Text v angličtine"/>	<input type="text" value="Text v slovenčine"/>	<input type="button" value="Uložiť"/> <input type="button" value="Vymazať"/>
konštanta	<input type="text" value="Text v angličtine"/>	<input type="text" value="Text v slovenčine"/>	<input type="button" value="Uložiť"/> <input type="button" value="Vymazať"/>
konštanta	<input type="text" value="Text v angličtine"/>	<input type="text" value="Text v slovenčine"/>	<input type="button" value="Uložiť"/> <input type="button" value="Vymazať"/>
konštanta	<input type="text" value="Text v angličtine"/>	<input type="text" value="Text v slovenčine"/>	<input type="button" value="Uložiť"/> <input type="button" value="Vymazať"/>
konštanta	<input type="text" value="Text v angličtine"/>	<input type="text" value="Text v slovenčine"/>	<input type="button" value="Uložiť"/> <input type="button" value="Vymazať"/>
konštanta	<input type="text" value="Text v angličtine"/>	<input type="text" value="Text v slovenčine"/>	<input type="button" value="Uložiť"/> <input type="button" value="Vymazať"/>
konštanta	<input type="text" value="Text v angličtine"/>	<input type="text" value="Text v slovenčine"/>	<input type="button" value="Uložiť"/> <input type="button" value="Vymazať"/>
konštanta	<input type="text" value="Text v angličtine"/>	<input type="text" value="Text v slovenčine"/>	<input type="button" value="Uložiť"/> <input type="button" value="Vymazať"/>
konštanta	<input type="text" value="Text v angličtine"/>	<input type="text" value="Text v slovenčine"/>	<input type="button" value="Uložiť"/> <input type="button" value="Vymazať"/>
konštanta	<input type="text" value="Text v angličtine"/>	<input type="text" value="Text v slovenčine"/>	<input type="button" value="Uložiť"/> <input type="button" value="Vymazať"/>
konštanta	<input type="text" value="Text v angličtine"/>	<input type="text" value="Text v slovenčine"/>	<input type="button" value="Uložiť"/> <input type="button" value="Vymazať"/>
konštanta	<input type="text" value="Text v angličtine"/>	<input type="text" value="Text v slovenčine"/>	<input type="button" value="Uložiť"/> <input type="button" value="Vymazať"/>

Obrázok 4: Obrázovka editora jazykových konštánt

Obrázok 4 zobrazuje rozhranie editora jazykových konštánt. Jazykové konštanty sú zobrazené v riadkoch tabuľky spolu s prekladom do všetkým podporovaných jazykov. Stĺpce z jazykmi pribudnú, ak je do systému pridaný nový jazyk. Každý riadok sa ukladá alebo vymazáva pomocou AJAX volania, takže nie je nutné zakaždým znovu-renderovať a zobrazovať celú stránku.

Tlačidlo Nová konštanta otvorí novú stránku pomocou FancyBox-u, bude tu formulár na vytvorenie konštanty a prekladu pre všetky podporované jazyky.

Na tomto obrázku si tiež môžeme povšimnúť základné rozčlenenie rozhrania pre administráciu. V hornej časti nad nadpisom je lišta s informáciou o otvorenej zostave úloh v ľavo a informácie o prihlásenom učiteľovi v pravo. Pod nadpisom sa nachádza hlavná navigácia, ktorá bude viacúrovňová. Celá táto štandardná hlavička bude navyše pevne umiestnená na vrchu stránky a bude vždy viditeľná (teda obsah stránky sa bude posúvať pod ňu pri vertikálnom posuve obsahu).



The image shows a graphical user interface for editing a language constant. It features a window with a title bar and standard window controls (minimize, maximize, close). Inside the window, there are three input fields with labels: 'Konštanta:', 'English language:', and 'Slovenský jazyk:'. Below these fields is a button labeled 'Uložiť' (Save). The 'Konštanta:' field is the first and is currently empty. The 'English language:' and 'Slovenský jazyk:' fields are also empty. The 'Uložiť' button is a small, rounded rectangle with a light gray background and a thin border.

Obrázok 5: FancyBox s editorom pre novú jazykovú konštantu

Obrázok 5 zobrazuje samotný FancyBox (obrazne) s editorom novej jazykovej konštanty. Po uložení by sa mala zobraziť správa o úspechu a čistý editor na vloženie ďalšej konštanty. Keď učiteľ editor zatvorí, znovu sa načítajú všetky existujúce konštanty do editora zobrazeného na obrázku 4.

The screenshot shows a web application window titled "L.I.S.T. - ADMINISTRÁCIA". At the top, there is a header bar with "Otvorená zostava: Zostava 1" on the left and "Meno Učiteľa | Môj účet | Slovenský jazyk | Odhlásiť sa" on the right. Below the header is a navigation bar with the text "NAVIGÁCIA".

There are two tabs visible:

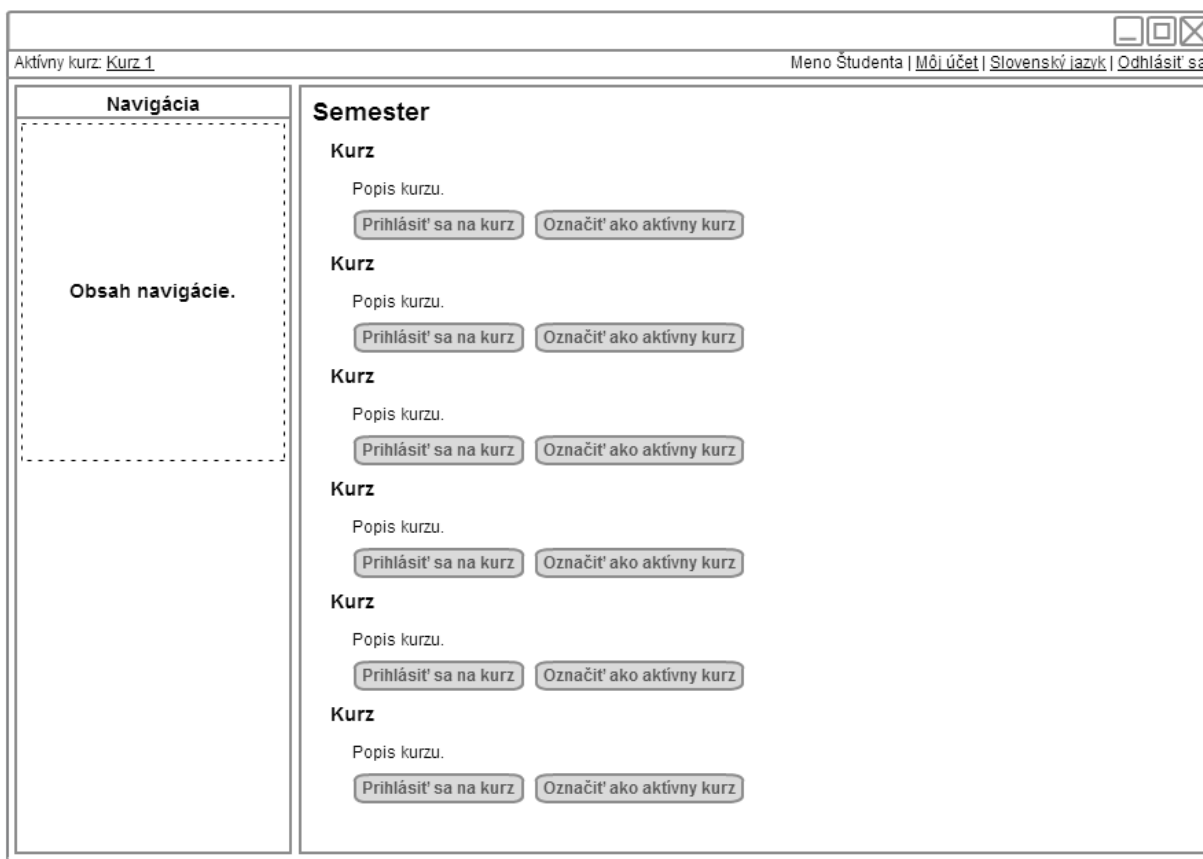
- Nová položka**: This tab contains a large dashed box with the text "Formulár na pridanie nového riadku v tabuľke."
- Všetky položky**: This tab contains a large dashed box with the text "Formulár na filtrovanie zobrazeného obsahu tabuľky." Below this, there is another dashed box with the text "Zobrazenie obsahu tabuľky podľa filtrov."

Obrázok 6: Všeobecné rozdelenie editorov tabuľkových záznamov

Obrázok 6 zobrazuje všeobecné rozdelenie editorov tabuľkových záznamov. Symbolicky tu možno vidieť tri hlavné oblasti:

- formulár na pridanie nového riadku v tabuľke, inak povedané vytvorenie nového záznamu, po vyplnení a odoslaní formulára sa zobrazí informácia o úspechu a znovu sa načíta obsah tabuľky s aplikáciou zobrazovacích filtrov,
- formulár na filtrovanie zobrazeného obsahu tabuľky, bude voliteľný, teda v editoroch, ktoré ho nepotrebujú, nebude prítomný, môže obsahovať rôzne nastavenia a tlačidlo na aplikáciu filtra, filter môže byť niekde aplikovaný aj automaticky po zmene hodnôt v ovládacích prvkoch,
- zobrazenie obsahu tabuľky podľa filtrov, bude HTML tabuľka zobrazujúca informácie o tabuľke v databáze či reláciách k iným tabuľkám z tejto tabuľky, posledný stĺpec bude vždy tvoriť zoznam ovládacích tlačidiel k jednotlivým záznamom, s najväčšou pravdepodobnosťou to budú hlavne tlačidlá na úpravu záznamu a vymazanie záznamu, ďalšie tlačidlá môžu byť v rôznych editoroch rôznych tabuliek prítomné podľa potreby.

## 5.11 Používateľské rozhranie pre študentov



Obrázok 7: Zobrazenie všetkých kurzov študentovi

Obrázok 7 zobrazuje zoznam semestrov a kurzov v nich vyučovaných pre každého študenta (prihláseného aj neprihláseného). Študent sa bude môcť prihlásiť na kurz (ak nie je prihlásený so svojim účtom do systému, bude sa musieť najprv prihlásiť). Po prihlásení dostane študent oprávnenie prezerat' úlohy z kurzu, až keď bude jeho prihlásenie do kurzu schválené učiteľom.

Tlačidlo prihlásiť sa na kurz bude zobrazené len pri tých kurzoch, na ktoré študent nemá v systéme evidovanú prihlášku.

Tlačidlo označiť ako aktívny kurz nastaví študentovi aktívny kurz a následne bude študent pracovať s úlohami v tomto kurze. Tlačidlo nebude zobrazené pri kurze, ktorý je aktívny.

Môžeme si tiež všimnúť podobnosť hornej lišty s tou, ktorá bude v administrácii. V tomto prípade ale nebude statická a bude viditeľná len ak je do systému prihlásený študent so svojim účtom.

Aktívny kurz: Kurz 1
Meno študenta | [Môj účet](#) | [Slovenský jazyk](#) | [Odhlásiť sa](#)

**Navigácia**

Obsah navigácie.

**Typ úloh**

Názov úlohy	Termín odovzdania	Získané body	Komentár k vypracovaniu
Názov úlohy	DD.MM.RRRR HH:MM:SS	x / y	Komentár.
Názov úlohy	DD.MM.RRRR HH:MM:SS	x / y	Komentár.
Názov úlohy	DD.MM.RRRR HH:MM:SS	x / y	Komentár.
Názov úlohy	DD.MM.RRRR HH:MM:SS	x / y	Komentár.
Názov úlohy	DD.MM.RRRR HH:MM:SS	x / y	Komentár.
Názov úlohy	DD.MM.RRRR HH:MM:SS	x / y	Komentár.

Celkové hodnotenie za typ úlohy: n bodov

**Typ úloh**

Názov úlohy	Termín odovzdania	Získané body	Komentár k vypracovaniu
Názov úlohy	DD.MM.RRRR HH:MM:SS	x / y	Komentár.
Názov úlohy	DD.MM.RRRR HH:MM:SS	x / y	Komentár.
Názov úlohy	DD.MM.RRRR HH:MM:SS	x / y	Komentár.
Názov úlohy	DD.MM.RRRR HH:MM:SS	x / y	Komentár.
Názov úlohy	DD.MM.RRRR HH:MM:SS	x / y	Komentár.
Názov úlohy	DD.MM.RRRR HH:MM:SS	x / y	Komentár.

Celkové hodnotenie za typ úlohy: n bodov

Hodnotenie spolu: m bodov

Obrázok 8: Zobrazenie zadaní pre študenta spolu s ich vyhodnotením

Obrázok 8 zobrazuje zoznam úloh pre študenta v aktívnom kurze. Úlohy sú rozdelené podľa ich typov (domáca úloha, cvičenie, prémiová úloha, atď.), každá úloha obsahuje aj hodnotenie (počet bodov) a komentár od hodnotiaceho učiteľa.

Pod každou tabuľkou k jednotlivým typom úloh je zobrazený celkový počet bodov za tento typ úloh, ktoré študent ich vypracovaním dosiahol. Úplne dole pod všetkými úlohami je zobrazený celkový počet bodov za všetky typy úloh spolu.



Aktivný kurz: Kurz 1 Meno študenta | [Môj účet](#) | [Slovenský jazyk](#) | [Odhlásiť sa](#)

**Navigácia**

**Obsah navigácie.**

**Názov zadania**

Zadanie
Vaše riešenia

Lorem ipsum dolor sit amet, urna eget, commodi amet vivamus quibusdam donec. Wisi ultricies cum blanditiis turpis quis, ante et eu. Metus eu sodales pede arcu. Quam augue et duis. Ullamcorper metus vitae nunc pellentesque lacus, amet vitae odio ligula pellentesque urna, ridiculus tortor pretium quam aenean maecenas, faucibus eget neque. In et proin massa tortor, quis ridiculus, id et facilisis vitae viverra mattis. Eu litora purus sapien.

Lobortis nulla urna eget suscipit pellentesque, integer rhoncus nibh ut sed justo fringilla, tortor magnis eu, integer anim nisl. Vestibulum vestibulum sit dolor, tempus in dui, ante integer sit et. Volutpat urna ultricies libero velit, nunc morbi est malesuada metus, quam nam mauris. Ac magna viverra aliquet cras, ligula ac ac lobortis, eleifend porttitor quis lacinia ut. Malesuada vestibulum quam urna eget, aliquam malesuada aliquet. Ut vestibulum, aenean wisi nullam mi magna non, non felis, wisi feugiat praesent nunc. Arcu nullam duis, aliquam in erat luctus lorem, eros dui feugiat velit lacinia fringilla.

Ligula elit magnis dictum mauris quis litora, mus per magna luctus nulla, suscipit leo ipsum. Quibusdam quis est quis mollis, in conubia suscipit fames vehicula. Pellentesque eget viverra ea odio, felis porttitor sodales mi neque et, sit a interdum nec, vitae sapien. Amet modi ornare, cras enim duis faucibus mauris libero pellentesque, vitae erat purus, ante sapien bibendum at ornare, ligula vulputate feugiat rutrum sed. Sit at, platea non habitasse turpis donec vel est, a tempore donec vivamus wisi, quos sagittis. Luctus in vestibulum nec, commodo nulla ut a ipsum, vel pellentesque ligula lorem libero dolor mauris, in pede sed a a vitae duis, ac montes justo et quis. Eros bibendum non lacus, mi montes, eos orci, eu nec, scelerisque sed volutpat nec scelerisque. Volutpat lacus dictum eleifend nulla ut justo, metus sit pede praesent nec quam pellentesque, nec lectus ut. Volutpat facilisis vitae quisque a, vitae neque nam enim pulvinar molestie in, aliquet augue integer at metus et metus. Ultricies ornare vitae nibh, dapibus odio duis tortor eu orci quisque, lacus per suspendisse wisi sed praesent, dolor vitae quis duis dignissim vestibulum.

Súbor
Odoslať riešenie

Obrázok 9: Stránka s textom zadania úlohy pre študenta

Obrázok 9 zobrazuje ako bude vyzerat' zobrazenie textu zadania úlohy pre študenta, študent sa sem dostane po kliknutí na názov úlohy v zozname úloh (obrázok 8).

Táto stránka bude rozdelená do dvoch záložiek, obrázok 9 zobrazuje textovú časť, kde si študent prečíta, čo má urobiť aby splnil zadanie. Na konci stránky pod textom zadania by mal byť zobrazený odosielač formulár na riešenie zadania. Tento formulár sa nebude zobrazovať, ak zadanie nebude vyžadovať odoslanie riešenia (napr. v systéme bude úloha, ktorú študenti vypracujú a odovzdajú na hárku papiera priamo na cvičení).

V prípade, že nebude povolené odosielanie súborov riešenia, nebude viditeľná ani záložka s riešeniami zadania prihláseným študentom. Obsah tejto záložky zobrazuje obrázok 10.

Aktivný kurz: Kurz 1 Meno študenta | Môj účet | Slovenský jazyk | Odhliásiť sa

**Navigácia**

Obsah navigácie.

**Názov zadania**

Zadanie
Vaše riešenia

Názov súboru	Kapacita	Posledná zmena
<input type="checkbox"/> <u>Názov súboru</u>	1MB	01.01.1970 00:00:00
<input type="checkbox"/> <u>Názov súboru</u>	1MB	01.01.1970 00:00:00
<input type="checkbox"/> <u>Názov súboru</u>	1MB	01.01.1970 00:00:00
<input type="checkbox"/> <u>Názov súboru</u>	1MB	01.01.1970 00:00:00
<input type="checkbox"/> <u>Názov súboru</u>	1MB	01.01.1970 00:00:00
<input type="checkbox"/> <u>Názov súboru</u>	1MB	01.01.1970 00:00:00
<input type="checkbox"/> <u>Názov súboru</u>	1MB	01.01.1970 00:00:00
<input type="checkbox"/> <u>Názov súboru</u>	1MB	01.01.1970 00:00:00

**Unit testy**

Názov testu

Spustiť

Názov testu

Spustiť

Názov testu

Spustiť

Názov testu

Spustiť

Obrázok 10: Zoznam odovzdaných riešení študentom k zadanej úlohe

Obrázok 10 zobrazuje zoznam odovzdaných riešení študentom v práve otvorenom zadaní. Súbor bude mať pomenovanie podľa mena študenta spolu s ID číslom študenta a číslovaním súboru. Novo odovzdané riešenie teda neprepisuje skorej odovzdané riešenie.

Pri každom súbore bude označovacie políčko, ktorým bude môcť študent spustiť unit test na zvolené riešenie. Na spustenie unit testu sa bude dať vybrať maximálne jedno riešenie.

Ak k zadaniu úlohy nie je prístupný žiaden unit test, nebudú viditeľné ani označovacie políčka. Jednotlivé unit testy sa budú líšiť len tým, v akom programovacom jazyku sú definované.

Po stlačení tlačidla spustiť sa na vybrané riešenie postupne spustia unit testy všetkých čiastkových úloh, výsledok unit testu bude zobrazený v oddelenom okne.

## 6 Realizácia a implementácia

Systém L.I.S.T. je implementovaný v programovacom jazyku pre webové aplikácie PHP, implementácia je dosiahnutá pomocou frameworku CodeIgniter. Do frameworku bolo pridané rozšírenie na mapovanie databázových tabuliek na PHP objekty, DataMapper ORM, ktoré takto vytvárajú akési alternatívne druhy modelov k tým, ktoré pozná samotný framework. Tak isto sa namiesto pôvodného riešenia pre prezentačnú časť používa PHP template engine Smarty.

Z technológií, ktoré sa používajú na skriptovanie na strane klienta je najzaujímavejšou použitá knižnica jQuery. Vďaka nej sa dá jednoducho manipulovať s DOM, realizovať volania cez AJAX a pridávať nové funkcionality k tejto knižnici vďaka pluginom, ako je napríklad FancyBox.

### 6.1 Implementácia databázy a modelov

Aby sa neopakovala situácia ako pri predchádzajúcej práci, kedy databázová štruktúra nebola nijako generovaná pôvodným systémom LaMSfET, boli v tomto systéme vytvorené migrácie pre databázu. Ide o triedy v PHP, ktoré CodeIgniter načíta a postupne vykonáva v doprednom alebo spätnom posúvaní sa po verzii aktuálnej migrácie k cieľovej verzii.

Týmito migráciami je podchytená celá databázová štruktúra, čo umožní pri nasadení jednoducho vygenerovať celú štruktúru programovo a nie je nutné niekde externe držať exportovaný .sql súbor so štruktúrou (čo navyše môže spôsobiť aj bezpečnostné riziko, ak by sa takýto súbor s reálnou štruktúrou databázy nachádzal prístupný v document root-e aplikácie).

Väčšina databázových tabuliek je v systéme sprístupnená pomocou modelov odvodených od triedy DataMapper. Tieto modely majú definované relácie medzi sebou, prípadne majú definované ďalšie užitočné funkcie.

Kvôli efektívnosti a zníženiu zaťaženia databázového serveru a databázy samotnej nadmernými dopytmi, je od začiatku vývoja zapnutá produkčná vyrovnávacia pamäť pre DataMapper ORM. Táto vlastnosť dovoľuje, aby si DataMapper vytvoril obraz o tom, ako vyzerá tabuľka a jej model, ktorý ju reprezentuje v systéme. Toto pozitívum má ale z hľadiska vývoja jednu negatívnu vlastnosť. Produkčnú vyrovnávaciu pamäť treba aspoň pre ovplyvnené modely vymazať v momente, keď je vykonaná zmena štruktúry tabuľky alebo je vykonaná zmena v modeli (napr. je pridaná nová relácia alebo validátor). Na toto správanie by nemal zabúdať prípadný ďalší študent alebo iný vývojár, ktorý bude neskôr v implementácii softvéru pokračovať. Samotné vypnutie produkčnej vyrovnávacej pamäte nevymaže obsah pôvodnej!

#### 6.1.1 Implementácia kategórií úloh

Model pre kategórie úloh je síce modelom odvodeným od DataMapper ORM, ale kategórie

samotné vytvárajú stromovú štruktúru, ktorú samotný DataMapper nevie reprezentovať svojimi volaniami.

Taktiež kvôli absencii rekurzívnych dopytov v databázach MySQL je implementácia kategórií riešená jedným dopytom, ktorý načíta všetky riadky tabuľky kategórií a následne sa z nich rekurzívne vygeneruje stromová reprezentácia.

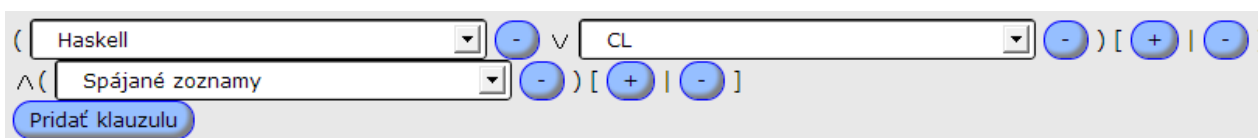
V stromovej štruktúre kategórií neexistuje reálny koreň, teda nejaká jediná nadradená kategória, z ktorej sa rozvíja strom. Koreň je imaginárny a jeho potomci sú všetky tie kategórie, ktoré majú `parent_id` stĺpec nastavený na hodnotu NULL. Toto riešenie má výhodu v tom, že ak je vymazaná nadradená kategória, vďaka automatickému nastaveniu potomkom v relácii ich `parent_id` stĺpca na NULL sa nepokazí reprezentácia stromu a všetci potomci zmazanej kategórie sa stanú potomkami imaginárneho koreňa.

Okrem metód na generovanie stromu model implementuje aj metódy na generovanie zoznamu ID (hodnôt primárneho kľúča), ktoré fungujú tak, že dostanú ID kategórie a zoznam ktorý vráti obsahuje ID samotnej kategórie a všetkých potomkov. Ak však ID kategórie neexistuje alebo príde k inej chybe, je vrátený zoznam s číslom nula. Toto opatrenie je tu preto, aby sa dal výsledok funkcie priamo použiť v podmienkach CodeIgniteru typu `where_in`, ktoré očakávajú zoznam prvkov a ak je prázdny, vygeneruje sa chybný SQL dopyt (hodnota nula sa samozrejme ako hodnota primárneho kľúča nepoužíva).

Metóda na generovanie zoznamu ID je tiež optimalizovaná, aby mohla byť volaná viackrát po sebe s rôznymi vstupnými ID tým, že prvé volanie uloží do privátnej statickej členskej premennej výsledok SELECT dopytu.

### 6.1.2 Implementácia filtrovania úloh podľa kategórií

Úlohy je treba vedieť filtrovať pri ich vyhľadávaní pomocou kategórií, do ktorých sú priradené, avšak toto vyhľadávanie ma byť na základe konjunktívnej normálnej formy.



Obrázok 11: Filtrovanie úloh podľa kategórií na základe konjunktívnej normálnej formy

Na obrázku 11 možno vidieť ukážku filtra. Tento filter hovorí o tom, že úlohy, ktoré hľadáme, majú byť zadané buď pre Haskell alebo CL a súčasne majú byť zamerané na Spájané zoznamy. Obrázok tiež ukazuje, že sa bude filtrovať podľa dvoch klauzúl.

Keďže ID, ktoré budú z filtra poslané na spracovanie predstavujú koreňové hodnoty v podstromoch celého stromu kategórií, je treba brať do úvahy aj potomkov. Napríklad kategória

Spájané zoznamy by mohla mať ďalšie podkategórie, napr. Obojsmerne spájané zoznamy či Cyklicky spájané zoznamy. Ak je vybraná nejaká kategória, do výsledku sa majú zahrnúť aj tie úlohy, ktoré sú označené príslušnosťou do podkategórie a spĺňajú všetky podmienky.

Problém je, že MySQL nedovoľuje rekurziu, takže na rad prichádzajú funkcie na získanie zoznamu kategórií popísané v kapitole 6.1.1 Implementácia kategórií úloh. Tak isto každá jedna klauzula musí byť v dopyte reprezentovaná jedným JOIN-om na join tabuľku, ktorá má informácie o reláciách medzi úlohami a kategóriami. Toto však neide realizovať priamočiaro pomocou DataMapperu alebo Active Records z CodeIgniter a bolo treba na tento účel vytvoriť metódu modelu úloh, ktorá do dopytu vygeneruje požadované JOIN-y a podmienky.

### 6.1.3 Zostavy úloh a ich implementácia modelom

Zostava úloh je akýmsi súborom úloh, ktoré sú prezentované študentovi na vypracovanie. V administrácii chceme vytvárať obsah týchto zostáv takým spôsobom, ktorým učiteľ najprv vytvorí samotnú zostavu a potom ju označí ako otvorenú. Do takto otvorenej zostavy bude potom umožnené vkladať neobmedzený počet úloh (samozrejme, každú úlohu práve raz).

Aby sa otvorená zostava nemusela posielat medzi stránkami v url, je otvorená zostava implementovaná pomocou session dát. Model zostavy úloh bude implementovať metódy na označenie zostavy ako otvorenej a na načítanie otvorenej zostavy. Aby sa zabránilo otvoreniu neexistujúcej zostavy, označiť sa za otvorenú dá len tá inštancia modelu, ktorá obsahuje ne-NULL-ové ID.

Model zostavy tak isto implementuje dve metódy vkladajúce podmienky do dopytu na to, či zostava úloh má alebo nemá priradené úlohy. Na toto je snád' jediný krát použitý priamo textovo pridaný pod-dopyt, ktorý vyberie a spočíta počet úloh v relácii so zostavou.

## 6.2 Implementácia jazykových lokalizácií

Aplikácia používa niekoľko spôsobov na lokalizovanie textov. Jedným je klasický spôsob ako funguje lokalizácia v CodeIgniter a o ktorej sa čitateľ môže dozvedieť podrobnosti v manuáli k frameworku na jeho domovskej stránke. Zaujímavejšie z hľadiska vlastnej práce a návrhu sú druhé dva spôsoby, ktoré počítajú s ukladaním lokalizácie do dvoch rozdielnych databázových tabuliek.

### 6.2.1 Priama lokalizácia pomocou používateľských konštánt

Na tento spôsob lokalizácie sa používa tabuľka translations, ktorá má iba tri stĺpce: *constant*, *idiom* a *text*. Stĺpce *constant* a *idiom* spolu tvoria primárny kľúč. Na túto tabuľku existuje priamo model Translations rozširujúci klasický CI\_Model z CodeIgniter.

Model implementuje niekoľko metód. Jedna z nich umožňuje načítavať všetky konštanty a ich

texty do asociatívneho pola, pričom sa načítajú podľa zvoleného jazykového idiómu. Takéto pole sa potom ako parameter predá mnou pridanej metóde v triede `LIST_Lang` (zabezpečujúcej lokalizáciu) a tieto používateľské konštanty sa pridávajú k ostatným už načítaným konštantám z jazykových súborov.

Model ďalej štandardne implementuje metódy, ktoré vrátia celý zoznam konštant a textov pre všetky jazykové idiómy, kvôli zobrazeniu v editore lokalizácie. Tiež implementuje metódy na ukladanie (metóda vykonáva `INSERT` alebo `UPDATE` dopyt podľa potreby) a vymazávanie lokalizácie. Posledná implementovaná metóda kontroluje existenciu konštanty v lokalizačnej tabuľke, aby sa zabránilo vytvoreniu duplicitnej konštanty.

Spoločne s úpravami v knižnici jadra, `LIST_Lang`, je takto možné lokalizovať texty v používateľskom obsahu, pretože do lokalizačnej knižnice som pridal alternatívnu metódu, ktorá dostane vstupný text, a ak tento obsahuje prefix *lang:*, bude sa pokladať za jazykovú konštantu a nahradí sa textom z príslušnej konštanty pre aktívny jazyk.

### 6.2.2 Lokalizácia textov pomocou prekrytí

Mnou vytvorená trieda `LIST_Lang`, ktorá je rozšírením pôvodnej triedy jadra `CI_Lang`, implementuje lokalizačné funkcie pomocou prekrývania textov.

Text v databázových tabuľkách, ktorých sa takýto typ lokalizácie týka, je chápaný ako uložený vo východnom jazyku. Tento východzí jazyk však môže byť pre každý záznam iný. Reálna situácia, ktorá môže nastať:

Existuje kurz, ktorý navštevujú zahraniční študenti hovoriaci iba anglicky. Učitelia vytvárajú úlohy, no jeden učiteľ vytvorí úlohu popísanú po slovensky, druhý učiteľ ju napíše po anglicky. Tým je myslený východzí jazyk v jednotlivých záznamoch v tabuľke úloh. Aby sa obe úlohy zobrazovali lokalizované, dodajú učitelia k úlohám preklad do druhého jazyka formou prekrytia. Potom budú obe úlohy lokalizované do správnych jazykov.

Jazykové prekrytia sú uložené v tabuľke `lang_overlays`, ktorá má päť stĺpcov: *table*, *table\_id*, *column*, *idiom* a *text*. Prvé štyri stĺpce tvoria primárny kľúč. Tabuľka teda ukladá text prekrytia pre konkrétny jazyk, pre konkrétny riadok a stĺpec v konkrétnej cieľovej tabuľke. Čiže ak mám v tabuľke *tasks* stĺpec *text*, ktorý je lokalizovaný pomocou prekrytia, a mám úlohu s ID 10, budem mať v tabuľke prekrytia napríklad takýto záznam kvôli lokalizácii:

table	table_id	column	idiom	text
tasks	10	text	english	English version of text.

Trieda `LIST_Lang` implementuje niekoľko metód, ktoré efektívnym spôsobom pracujú s týmito

prekrytiami. Efektivita je mienená hlavne k počtu dopytov na databázu, ktoré sa snažím minimalizovať na pokiaľ možno jeden.

Trieda si ukladá všetky načítané prekrytia, aby nebolo nutné v tom istom session ich znovu načítavať z databázy, ak sa používajú viackrát. Aby sa tiež nemuseli pre každý záznam z tabuľky, v ktorej sa nachádza pôvodný text vo východnom jazyku viackrát načítavať prekrytia pre jednotlivé riadky, implementoval som metódu, ktorá dostane názov tabuľky, už načítané riadky tejto tabuľky (tie z pôvodnými textami) a zoznamom stĺpcov, pre ktoré môže existovať prekrytie. Táto metóda potom načíta tieto prekrytia a interne si ich uloží do členskej premennej. Tie prekrytia, ktoré sa nenájdu, sú označené ako neexistujúce aby sa pri pokuse zobrazit' ich knižnica opätovne nepokúšala ich načítať z databázy.

Knižnica potom vie štandardne vrátiť prekrytie podľa parametrov (tabuľka, id, stĺpec a jazyk) a tiež vykonať uloženie či vymazanie prekrytí. Keďže niektoré objekty je treba klonovať, ako sú úlohy a zostavy úloh, a tieto objekty majú práve prekrytiami lokalizované texty, je implementovaná aj metóda na klonovanie príslušných prekrytí, ktorej stačí dať názov tabuľky a starý s novým primárnym kľúčom.

Všetky databázové operácie s prekrytiami sú tentoraz implementované priamo v knižnici a nie v samostatnom modeli.

### 6.2.3 Vyhľadávanie a zorad'ovanie výsledkov dopytov podľa lokalizovaných stĺpcov

Aby bolo možné realizovať SQL dopyty s LIKE podmienkou na stĺpce lokalizované pomocou buď konštant alebo prekrytí, a tiež aby bolo možné realizovať ORDER BY klauzulu na takýchto stĺpcoch, vytvoril som rozšírenie pre DataMapper s názvom DMZ\_Translations, ktorí implementuje niekoľko metód umožňujúcich dosiahnuť reálne zorad'ovanie a vyhľadanie.

Implementácia je na báze vytvorenia subquery v ORDER BY alebo LIKE klauzule a vybrať jeden riadok buď z príslušnej prekladovej tabuľky, alebo použije aktuálnu hodnotu stĺpca, ak nebol nájdený záznam v prekladovej tabuľke. Subquery sú realizované pomocou UNION s LIMIT 1, takže vždy vráti len jeden riadok.

Keďže samotný DataMapper podporuje zorad'ovanie výsledku dopytu podľa hodnoty stĺpca v niektorej z tabuliek v relácii s týmto stĺpcom, implementoval som podobné metódy aj do tohto rozšírenia, s využitím metód základnej triedy DataMapper.

## 6.3 Bezpečnosť systému

Väčšina akcií vo všetkých ovládačoch je zabezpečená proti prístupu neautorizovaných používateľov. Existujú dva autentifikačné body:

- students/login pre prihlásenie študenta,

- admin\_teachers/login pre prihlásenie učiteľa.

Vstupné formuláre na prihlásenie sú zabezpečené pred SQL injection aj útokom hrubou silou.

Vďaka tomu, že autentifikačná informácia o názve účtu je e-mailová adresa študenta alebo učiteľa, a pri autentifikácii je kontrolovaná správnosť zadania tohto údaju (regulárnym výrazom), je nemožné toto pole zneužiť na SQL injection útok. Údaj o hesle je na rovnako kontrolovaný na rozsah dĺžky a pred vložením do SQL dopytu je hashovaný pomocou SHA1, teda ani tento údaj nie je možné zneužiť na SQL injection útok. Formuláre boli rovnako otestované na SQL injection penetračným testom, v ktorom obstáli.

Proti útoku hrubou silou je zabezpečená aplikácia kontrolou špeciálnej bezpečnostnej tabuľky, do ktorej si zapisuje informácie o každom neúspešnom ale oprávnenom pokuse o prihlásenie. Pomocou nastavení časového intervalu a počtu opakovaní je používateľ, ktorí je neschopný sa autentifikovať do systému, na nastavený čas odstavený od možnosti sa prihlásiť.



## 7 Záver

Motiváciou tejto bakalárskej práce bolo zhodnotenie existujúceho systému LaMSfET, vyhodnotenie jeho nedostatkov, jak v zdrojovom kóde tak v jeho funkcionalite, s následným návrhom opatrení, ktoré by mali pôvodný systém opraviť. Inou možnou alternatívou bolo navrhnúť nový systém.

Po vyhodnotení nedostatkov pôvodného systému LaMSfET bolo rozhodnuté o vytvorení nového projektu, ktorým by bol nový systém s rovnakým účelom skladu úloh a nástroja na podporu výuky. Takto vznikol návrh systému Long-term Internet Storage of Tasks, alebo LIST. Návrh nového systému bol založený na maximálnom využití existujúcich a dobre zdokumentovaných open-source riešení, aby sa vyhol hlavnému nedostatku systému LaMSfET, ktorým bola nezrozumiteľnosť a neprehľadnosť zdrojového kódu. Pôvodný systém LaMSfET implementoval iba jediné open-source riešenie, PHP template engine Smarty.

Ďalším cieľom tejto práce bola snaha o implementáciu väčšiny funkcionality z návrhu nového systému LSIT, pričom tento cieľ bol dosiahnutý. Niekoľko ďalších chýbajúcich častí systému bude implementované počas letných prázdnin, aby bolo možné systém nasadiť od septembra 2013.

Vďaka novému systému LIST máme teraz nástroj, ktorý je možné ďalej vyvíjať rapídny tempom, nakoľko implementuje množstvo známych open-source riešení, ako je CodeIgniter, Smarty, DataMapper, TinyMCE, Plupload, jQuery, jQuery-UI, FancyBox a ďalšie. Ďalší vývojár tak má poruke všetku potrebnú dokumentáciu týchto riešení a tiež API dokumentáciu zdrojového kódu vygenerovanú pomocou phpDocumentor.

Do budúcnosti by bolo zaujímavé systém LIST rozšíriť o ďalšie funkčné časti, ako sú:

- samostatný druh zadání typu projekt, z ktorých by si študent mohol vyberať a prihlasovať sa na ne,
- integrovať systém na kontrolu plagiátov v zdrojovom kóde odovzdaných riešení úloh,
- pridať niektoré bežné funkcie LMS systémov, ako sú diskusné boardy alebo aspoň komentáre k zadaniam úloh, prípadne pridať možnosť priamej komunikácie s učiteľmi pomocou súkromných správ.

## 8 Použitá literatúra a prílohy

### 8.1 Literatúra

- [1] Peter Jurčo: Dlhodobý viacúčelový sklad úloh na cvičenia, 2009, bakalárska práca, FMFI UK Bratislava
- [2] JUnit framework api, junit.sourceforge.net, dostupné na: <http://junit.sourceforge.net/javadoc/>
- [3] Smarty template engine, [www.smarty.net](http://www.smarty.net), dostupné na: <http://www.smarty.net/docs/en/>
- [4] TinyMCE, Moxiecode Systems AB, dostupné na: <http://www.tinymce.com/>
- [5] Moodle LMS, [www.moodle.org](http://www.moodle.org), dostupné na: <https://moodle.org/>
- [6] Chamilo LMS, Chamilo Association, dostupné na: <http://www.chamilo.org/>
- [7] JQuery, JQuery team, dostupné na: <http://www.jquery.com/>
- [8] JQueryUI, JQuery team, dostupné na: <http://www.jqueryui.com/>
- [9] CodeIgniter, EllisLab Inc., dostupné na: <http://ellislab.com/codeigniter>
- [10] GeSHi, Nigel McNie a Benny Baumann, dostupné na: <http://qbnz.com/highlighter/index.php>
- [11] DataMapper ORM, Wanwizard, dostupné na: <http://datamapper.wanwizard.eu/>
- [12] Plupload, Moxiecode Systems AB, dostupné na: <http://www.plupload.com/>
- [13] FancyBox, FancyBox.net, dostupné na: <http://fancyapps.com/fancybox/>

### 8.2 Prílohy

- (1) API dokumentácia zdrojového kódu, Andrej Jursa, Bratislava, 2013, dostupné na priloženom CD-ROM
- (2) Návrhu systému, Andrej Jursa, Bratislava, 2013, dostupné na priloženom CD-ROM