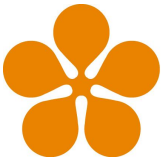


Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Diplomová práce

Vývoj nových modulů pro LMS Moodle

Vypracoval: Bc. Martin Drlík

Vedoucí práce: PhDr. Milan Novák, Ph.D.

České Budějovice 2013

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2011/2012**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Martin DRLÍK**
Osobní číslo:	**P11546**
Studijní program:	**N7503 Učitelství pro základní školy**
Studijní obory:	**Společný základ**
	Učitelství fyziky pro 2. stupeň základních škol
	Učitelství informatiky pro 2. stupeň základních škol
Název tématu:	**Vývoj nových modulů pro LSM Moodle**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Student analyzuje problematiku vývoje nových modulů pro LMS Moodle. Práce bude obsahovat teoretickou a praktickou část. V teoretické části práce student provede teoretický výzkum možností programovacího frameworku platformy MOODLE 2.0 a navrhne obecnou metodiku tvorby modulů. V praktické části student provede ověření navrhované metodiky vývoje modulů na konkrétním modulu. Modul bude představovat funkční instalaci pro LMS MOODLE 2.0 s dokumentací popisující jeho funkčnost, postup instalace, kritická místa a test přenositelnosti mezi dalšími verzemi LMS MOODLE.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **60**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

1. **MOODLE.CZ [online]. 2011. Dostupné z WWW: <http://moodle.cz/>.**
2. **MOODLE 2.0 [online]. 2011. MoodleDocs. Dostupné z WWW: <http://docs.moodle.org/20/en/Main_page>.**
3. **DE RAADT, Michael. Moodle 1.9 Top Extensions Cookbook book and eBook. US : PACKT Publishing, 2010. 324 s. ISBN 1849512167.**

Vedoucí diplomové práce: **PhDr. Milan Novák, Ph.D.**
Katedra informatiky

Datum zadání diplomové práce: **21. listopadu 2011**

Termín odevzdání diplomové práce: **30. dubna 2013**

Mgr. Michal Vančura, Ph.D.
děkan

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 21. listopadu 2011

Prohlášení

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně, pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 23. dubna 2013

Bc. Martin Drlík

Abstrakt

Práce se zabývá analýzou problematiky vývoje modulů pro LMS Moodle. Teoretická část je tvořena výzkumem možností programovacího frameworku platformy Moodle 2.0, na který navazuje metodika tvorby modulů založených na této platformě. Ověření metodiky je zajištěno v praktické části práce naprogramováním konkrétního modulu. Ten představuje platnou instalaci pro LMS Moodle 2.0, včetně dokumentace popisující jeho funkčnost, postup instalace, kritická místa a test přenositelnosti mezi dalšími verzemi tohoto systému.

Klíčová slova

LMS, Moodle, modul, eTask, hodnocení, úkoly.

Abstract

The aim of this diploma thesis is to analyze issues that come with the development of Moodle LMS modules. The theoretical part is devoted to the research of Moodle 2.0 programming framework capabilities, followed by the methodology of creating modules based on this platform. The verification of methodology is provided in the practical part by the development of a specific module, along with the effective installation of Moodle 2.0 LMS and documentation, that describes its functionality, installation procedure, critical points and a test of portability between other versions of this system.

Keywords

LMS, Moodle, module, eTask, evaluation, tasks.

Poděkování

Děkuji tímto panu PhDr. Milanu Novákovi, Ph.D. za odborné vedení této diplomové práce a za cenné rady, které mi při konzultacích předal.

Dále děkuji své rodině, která mne podporovala při studiu na vysoké škole a to nejen finančně.

V neposlední řadě patří mé díky lidem, kteří mi jsou na blízku a jsou mi oporou. Děkuji také všem, kteří se jakkoliv podíleli na zkvalitnění této diplomové práce.

Obsah

1 Úvod

LMS [1] (někdy též zmiňován jako VLE[2]) Moodle je jedním z nejpopulárnějších vzdělávacích prostředí na celém světě. Slovo Moodle je zkratkou z anglického Modular Object-Oriented Dynamic Learning Environment (modulární objektově orientované dynamické prostředí pro výuku) [1]. Jedná se o moderní a výkonný nástroj, který disponuje možností rozšiřovat jej o uživatelské moduly. Toho využívá tato diplomová práce, jejímž tématem je vývoj nových modulů pro tento systém.

Moodle je vyvíjen jako systém určený pro tvorbu elektronických kurzů v prostředí Internet. Je zdarma poskytován jako Open Source[3] software pod obecnou veřejnou licencí GNU[4]. Přes vztahující se autorská práva nabízí uživatelům značnou svobodu (možnost kopírování a upravování v souladu s licenčními podmínkami) [2].

Instalován může být na platformách podporujících webový server (nejčastěji Apache), interpret PHP[5] a SQL[6] databáze (např. MySQL, PostgreSQL a další). Instalace se skládá ze tří základních částí, které navzájem spolupracují [3]:

- adresář se zdrojovými kódy aplikace,
- adresář obsahující soubory nahrané uživateli (a některé další soubory kurzu) – tento adresář je zabezpečený tak, aby nebylo možné jej prohlížet přes webový prohlížeč [4],
- databáze (ukládá objekty a většinu informací v LMS Moodle).

[1] Learning Management System (systém pro řízení výuky)
[2] Virtual Learning Environment (virtuální prostředí pro výuku)
[3] Open source: `http://opensource.org/osd`
[4] GNU General Public License: `http://www.gnu.org/licenses/licenses.html#GPL`
[5] PHP: Hypertext preprocessor
[6] Structured Query Language (standardizovaný dotazovací jazyk)

2 Cíle práce

Cílem práce je analýza problematiky vývoje nových modulů pro LMS Moodle. Teoretický výzkum platformy Moodle 2.0, který bude proveden, je základem pro navržení obecné metodiky tvorby modulů. Její užitelnost bude následně prakticky ověřena naprogramováním konkrétního modulu. Cílem práce je na základě analýzy navrhnout a vytvořit modul eTask, který poskytuje přehlednou správu a hodnocení úkolů formou tabulkového výpisu. Nezbytnou součástí tohoto modulu je popis jeho funkčnosti (způsob instalace, možnosti nastavení, ovládání), dokumentace, kritická místa a také testování – nejen funkčnosti, ale i přenositelnosti mezi jednotlivými verzemi systému Moodle.

3 Metoda práce

Práce je členěna na dvě hlavní části – teoretickou a z ní vycházející praktickou. Teoretická část je tvořena výzkumem na základě obsahové analýzy, která je podkladem k sestavení obecné metodiky tvorby modulů pro LMS Moodle. V praktické části bude využito experimentálního ověření uvedené metodiky. Naprogramován bude modul eTask dle zadaných požadavků. Doplněn bude o dokumentaci a testy přenositelnosti. Modul bude publikován a nasazen v rámci systému Moodle. Důraz bude kladen na jednoduchost, intuitivní ovládání, přehlednost a funkčnost.

3.1 Východiska práce

Práce vychází z potřeby analyzovat vývoj nových modulů, na jehož základě bude sestavena obecná metodika. Nutností je navrženou metodiku ověřit – to bude zajištěno naprogramováním konkrétního modulu, který byl zvolen v důsledku nepřehledného hodnocení jednotlivých aktivit účastníků kurzu.

Na základě výše uvedených východisek práce byl stanoven modul, na kterém bude provedeno ověření metodiky. Jeho název je eTask. Jedná se o nástroj, který umožňuje intuitivní a přehlednou správu úkolů a jejich hodnocení v tabulkovém uspořádání. Na daný modul byly stanoveny požadavky a zároveň bylo ověřováno, zda již jeho obdoba není k dispozici v rámci pluginů, které Moodle poskytuje na svém serveru. Práce vychází též z požadavku Jihočeské univerzity na realizaci tohoto modulu.

3.2 Výzkumné problémy

V souvislosti s problematikou vývoje nových modulů pro LMS Moodle je nutné zjistit, jakým způsobem funguje framework dané platformy, jak se moduly tvoří. Zohledněna musí být též přenositelnost výsledného modulu mezi jednotlivými verzemi tohoto systému.

Proč modul eTask? Moodle je kvalitním prostředkem pro vytváření komplexních výukových kurzů. Je však dostačující i pro nejzákladnější kurzy, které si kladou za cíl pouze zadávat a hodnotit úkoly? Jako aktivní uživatel systému Moodle v roli učitele i studenta vyjadřuji svůj subjektivní názor, že pro správu těchto základních

aktivit je systém zbytečně složitý a tudíž i těžko využitelný. Důvodem je nepřehledná správa hodnocených úkolů. Podívejme se na následující obrázky vztahující se k hodnocení úkolů.

Úkol 01

Zadání úkolu 01

Celkem hodnocení

Účastníci	3
Odevzdáno	2
Nutno ohodnotit	1
Termín odevzdání	sobota, 9. únor 2013, 17.35
Zbývá	6 dny 23 hodin

Zobrazit/hodnotit všechny odevzdané úkoly

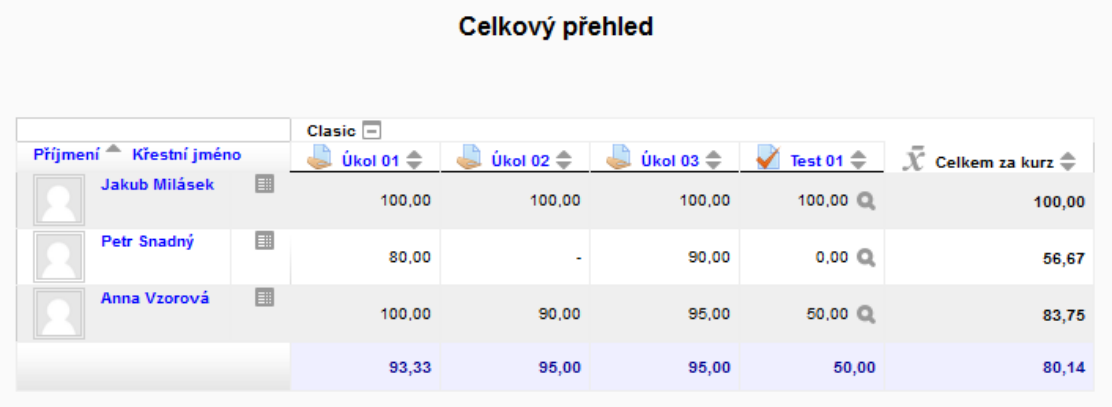
Obrázek 1: Základní informace o úkolu z pohledu učitele

Úkol 01

Výběr	Křestní jméno / Příjmení	Známka	Výsledná známka
	Anna Vzorová	100,00 / 100,00	100,00 / 100,00
	Petr Snadný	80,00 / 100,00	80,00 / 100,00
	Jakub Milásek	100,00 / 100,00	100,00 / 100,00

Obrázek 2: Zobrazení a hodnocení odevzdaného úkolu z pohledu učitele

Na obrázku 1 vidíme základní informace o úkolu z pohledu učitele – samotné hodnocení je však možné provádět až po zobrazení odevzdaných úkolů, jak vidíme na obrázku 2. Ten znázorňuje výpis studentů s patřičným ohodnocením odevzdané práce. Takové řešení je poměrně nesourodé a zbytečně zdlouhavé. Neměli bychom však opomenout existenci komplexní tabulky, která obsahuje hodnocení dílčích aktivit studenta. Vidět ji můžeme na obrázku 3 níže.



Obrázek 3: Známkování z pohledu učitele

Celkový přehled své uplatnění v systému Moodle zajisté má, ale hodnocení jednotlivých aktivit přes tuto tabulku nevyužívá svého potenciálu. Po zapnutí režimu úprav je sice umožněno hodnocení přímo upravovat, ale v souvislosti s požadavky na modul eTask nám však tento přehled neposkytuje požadované výsledky a efektivitu.

4 Analýza vývoje modulů v LMS Moodle

Tato část práce se zabývá analýzou vývoje modulů v LMS Moodle – tedy tím, jak jsou moduly vytvářeny a implementovány. Na úvod je nutné pochopit rozdíl mezi jednotlivými typy modulů. Podívejme se na ty nejzákladnější [1].

- Modul činností – modul činností umožňuje interaktivitu v rámci kurzu. Do této kategorie řadíme např. úkoly, fórum, chat, ankety či nahrávání souborů.
- Rozšíření typu úkol – ačkoliv úkoly spadají do modulu činností, lze rozlišovat jejich konkrétní typy. Patří sem např. online text či odevzdání souborů.
- Bloky – bloky patří mezi nejjednodušší typ modulů v LMS Moodle. Jelikož se snadno vytváří, nachází velké zastoupení v databázi modulů a pluginů. Jmenovat můžeme např. navigaci, kalendář, nadcházející události či prohledávání fóra.
- Uspořádání kurzu – určuje, jak bude stránka kurzu členěna. Typicky je využíváno uspořádání týdenní či tématické.
- Filtry – umožňují nahrazení určitých textů v rámci celého systému jiným obsahem. Filtry se využívají například pro lokalizaci systému, zobrazování rovnic či videa.
- Typy testovacích úloh – slouží k určení typu úlohy v rámci testu. Ta může být např. doplňovací, numerická či přiřazovací.

Chceme-li s jednotlivými typy modulů pracovat, je vhodné se zmínit o jejich umístění v rámci adresářové struktury LMS Moodle. Následující tabulka nám tuto informaci poskytuje (kořenový adresář máme pojmenován `moodle`) [1].

Typ modulu	Umístění
Modul činností	`moodle/mod/`
Rozšíření typu úkol	`moodle/mod/assignment/type/`
Bloky	`moodle/blocks/`
Uspořádání kurzu	`moodle/course/format/`
Filtry	`moodle/filter/`
Typy testovacích úloh	`moodle/question/type/`

Tabulka 1: Umístění modulů v adresářové struktuře systému

Kromě výše zmíněných základních typů modulů existuje celá řada dalších. Jejich komplexní přehled nalezneme v dokumentaci pluginů na webových stránkách systému Moodle [5].

Dříve než začneme samotný modul vyvíjet, je nutné si stanovit, co od modulu očekáváme, jakou má mít funkcionalitu a v neposlední řadě určit jeho typ. Dle výše uvedených informací si zvolíme takový, který nejvíce odpovídá naším požadavkům a představám o výsledném produktu.

4.1 Metodika vývoje modulů v LMS Moodle

Na základě obsahové analýzy byl proveden výzkum, který je východiskem pro vznik obecné metodiky vývoje modulů. V této kapitole se budeme zabývat tím, jak správně postupovat při tvorbě nového modulu. Postup bude následně ověřen aplikací na teoretickou část – vytvořen bude konkrétní modul, tedy eTask.

Různé typy modulů[7] mohou mít rozdílnou strukturu a postupy při jejich vývoji. Tato kapitola tedy poskytuje pouze obecnou metodiku jejich tvorby, což uvádějí i cíle práce (viz kapitola 2).

4.1.1 Požadavky na modul

V první fázi vývoje modulu je nutné stanovit si požadavky, které by měl splňovat – provést analýzu řešení a navrhnout datový model. Tento počáteční krok je zcela

[7]Komplexní přehled pluginů: `http://docs.moodle.org/dev/Plugins`

zásadní a neměl by se v žádném případě podcenit. Na jeho základě si totiž musíme zvolit, jaký typ modulu chceme vyvíjet – zda si vystačíme s typem blok či zda budeme muset využít složitější konstrukce modulu činností anebo uspořádání kurzu.

Poté co si stanovíme typ modulu, není problém jej vhodně umístit do adresářové struktury, kterou jsme zmínili v tabulce 1. Většina modulů je umístěna v samostatném podadresáři, což je výhodné [1].

4.1.2 Šablona nového modulu

Pro vývoj nového modulu je vhodné vycházet ze šablony, která udává strukturu modulu. V zásadě máme dvě možnosti jak šablonu získat.

- Stažení šablony – Moodle poskytuje ke stažení šablonu pro tvorbu nového modulu[8]. Ovšem, jak stránky dokumentace uvádějí, tento balíček je poskytován pro verzi Moodle 1.9 – většina postupů by měla být aplikovatelná i na verzi 2.x (v době psaní této práce je aktuální verze 2.4) [6]. Pokud se rozhodneme pro možnost stáhnout si šablonu, měli bychom přeci jen využít novější balíček, který je určen pro verzi 2.x a je ke stažení na adrese `https://github.com/moodlehq/moodle-mod_newmodule`.

- Zkopírování existujícího modulu – duhou možností je, že si v adresářové struktuře systému Moodle vyhledáme nejdříve typ modulu, který chceme realizovat a následně si zvolíme konkrétního zástupce daného modulu. Jeho složku si nakopírujeme a přejmenujeme tak, aby odpovídala názvu našeho modulu. Tím opět získáme šablonu modulu ve smyslu struktury adresáře a souborů.

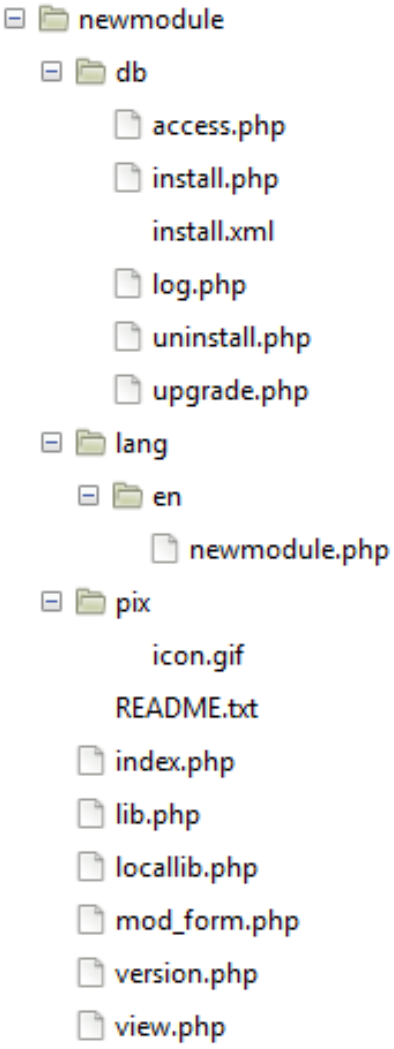
Je pouze na našem rozhodnutí, kterou z výše uvedených variant si zvolíme. Stažená šablona může obsahovat soubory, které nevyužijeme. Oproti tomu nevhodně zkopírovaný modul nemusí obsahovat všechny soubory, které potřebujeme pro svůj modul využít.

[8] Šablona modulu, verze 1.9: `http://download.moodle.org/plugins19/mod/NEWMODULE.zip`

4.1.3 Základní struktura modulu

Dříve než začneme programovat vlastní modul, podíváme se na obecnou strukturu nového modulu a na význam jednotlivých souborů. Vycházíme ze šablony modulu pro Moodle verze 2.x.

Nezapomínejme také na fakt, že soubory jsou již naplněny vzorovým zdrojovým kódem s komentáři – to nám ulehčí práci s nimi a umožní lépe se orientovat v použitých funkcích. Obrázek níže nám udává hierarchii souborů a složek obecného modulu.



Obrázek 4: Obecná struktura nového modulu

Popišme si nyní, jakou funkci jednotlivé soubory modulu plní, co by mělo být jejich obsahem [6, 7, 8].

- `db/` – adresář obsahující skripty, které zapisují data do databáze při instalaci či aktualizaci modulu.
  - ▷ `access.php` – tento soubor řídí, které činnosti mohou být v daném modulu pro uživatele dostupné. Jedná se o způsob přiřazování oprávnění pro jednotlivé akce a uživatelské role. Tato definovaná oprávnění jsou zapsána do databáze.
  - ▷ `install.php` – soubor dostupný od verze Moodle 2.0 a vyšší. Jedná se o post-instalační skript, který je proveden po vytvoření databázového schématu definovaného v `install.xml` (viz níže).
  - ▷ `install.xml` – XML definující strukturu databáze daného modulu. Pro vytvoření a úpravu tohoto souboru je doporučeno využít XMLDB editor. Jeho použití je následující [9].
    1. Do adresářové struktury umístíme vytvořený modul, který musí obsahovat podadresář `db` – např. `mod/newmodule/db`.
    2. Spustíme XMLDB editor, který je dostupný v administraci systému Moodle přes menu Správa stránek → Vývoj → editor XMLDB.
    3. Vyhledáme si záznam našeho modulu, tedy `mod/newmodule/db`, jak vidíme na obrázku 5 níže, a klikneme na odkaz `Vytvořit` (pokud není odkaz aktivní, ověříme, zda má webový server přístup pro zápis do adresáře modulu). Tím se vytvoří prázdný soubor `install.xml`.
    4. Následně klikneme na odkaz `Nahrát`. Soubor `install.xml` se načte do paměti a budeme s ním moci pracovat.
    5. V tomto kroku již můžeme vytvářet jednotlivé tabulky. Pro ukončení práce zvolíme odkaz `Uložit` (buď přímo v editačním okně nebo v hlavním přehledu).

Obrázek 5: Záznam modulu v XMLDB editoru

- ▷ `log.php` – vkládá do databáze záznamy během instalace či upgradu modulu. Pro pochopení logovacího souboru je vhodné seznámit se se strukturou databázové tabulky, do které jsou záznamy vkládány [10]. Ta je k dispozici v rámci dokumentace logovacího API.
- ▷ `uninstall.php` – soubor dostupný od verze Moodle 2.0 a vyšší. Je spuštěn, chceme-li odinstalovat modul. Provedení příkazů tohoto souboru předchází samotnému zrušení případných databázových tabulek.
- ▷ `upgrade.php` – využívá se, pokud je prováděna aktualizace modulu na novější verzi. Soubor obsahuje funkci `xmldb_newmodule_upgrade()`, kde `newmodule` je název modulu. V této funkci následně využíváme DDL[9] manažer a XMLDB třídy. Přehled funkcí, které můžeme využít, najdeme v API[10] pro definici dat – `http://docs.moodle.org/dev/DDL_functions`. Všechny funkce zmíněné v tomto API jsou určeny pro použití v systému Moodle verze 2.0 a vyšší [11].

- `lang/` – adresář obsahující jazykové balíčky. Při úpravě lokalizačního souboru musíme zvýšit verzi modulu v souboru `version.php`, aby mohla být provedena aktualizace.
  - ▷ `en/` – podadresář obsahující lokalizační soubor. Název adresáře je dvoumístný – jedná se o mezinárodně standardizovaný kód názvu jazyka dle normy ISO 639-1[11].
    - – `newmodule.php` – soubor obsahuje lokalizované řetězce v proměnné `$string[]`, jak je patrné z jeho zdrojového kódu. Proměnnou voláme metodou `get_string()`, která vrací lokalizovaný řetězec [12].

```
1 /* volání řetězce nazevmodulu v modulu newmodule vypíše Nový
2    modul */
3 get_string("nazevmodulu", "newmodule");
4
```

[9] Data Definition Language (jazyk pro definici dat)
[10] Application Programming Interface (rozhraní pro programování aplikací)
[11] Kompletní seznam jazykových kódů: `http://www.loc.gov/standards/iso639-2/php/code_list.php`

```
5 // řetězec v lokalizačním souboru
6 $string["nazevmodulu"] = "Nový modul";
```

Příklad 1: Lokalizace řetězce pomocí metody `get_string()`

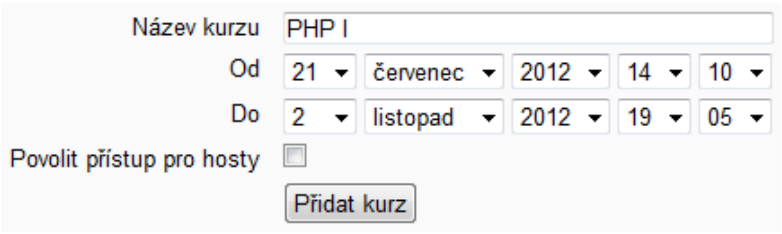
- `pix/` – adresář obrázků, respektive grafických prvků modulu.
  - ▷ `icon.gif` – soubor určující ikonu modulu. Tato ikona má standardní rozměr 16 × 16 pixelů.
- `README.txt` – do tohoto souboru uvádíme nejdůležitější informace, které by si měl uživatel přečíst před instalací modulu. Typicky uvádíme k čemu modul slouží a jak ho správně nainstalovat či ovládat.
- `index.php` – soubor obsahující seznam instancí aktivit obsažených v kurzu.
- `lib.php` – knihovna obsahující základní funkce[12], které modul využívá. Funkce začínají dle konvence názvem modulu, tedy řetězcem `newmodule_`.
- `locallib.php` – jedná se o lokální knihovnu. Oproti souboru `lib.php`, který obsahuje základní funkce, do něj umisťujeme funkce doplňkové – optimalizujeme tak výkon systému. Chceme-li obě knihovny propojit, potom do souboru `locallib.php` vkládáme `lib.php` a to pomocí příkazu `require_once`. Pokud jádro systému Moodle volá funkce ze souboru `lib.php`, nemusí do paměti načítat funkce lokální.
- `mod_form.php` – poskytuje formulář při vytvoření nové instance modulu. Umožňuje např. zadávat vstupy pro nastavení modulu. Formulář vytváříme pomocí rozšíření třídy `moodleform`, případně `moodleform_mod`. Třídy využívají knihovnu `formslib.php`. Přidávání jednotlivých elementů formuláře je popsáno v dokumentaci[13]. Nejčastěji se pro přidání formulářového prvku využívá funkce `addElement()`. K dispozici však máme i další, např. `setType()`, `addRule()` či `setDefault()`. Ukažme si princip implementace jednoduchého formuláře – výstup vidíme na obrázku 6 níže [13].

[12] Seznam základních funkcí knihovny `lib.php`: `http://docs.moodle.org/dev/NEWMODULE_Documentation#lib.php`

[13] Definice formulářů: `http://docs.moodle.org/dev/lib/formslib.php_Form_Definition`

```
1  // zpřístupníme knihovnu formslib.php v souboru mod_form.php
2  require_once($CFG->libdir . "/formslib.php");
3
4  // inicializujeme rozšíření třídy moodleform
5  class newmodule_form extends moodleform {
6     function definition() {
7        global $CFG;
8
9        $mform = $this->_form;
10       // přidáme elementy
11       $mform->addElement("text", "coursename", get_string("
            coursename", "newmodule"), 'size="40"');
12       $mform->setDefault("coursename", "PHP I");
13       $mform->addElement("date_time_selector", "coursefrom",
             get_string("from"));
14       $mform->addElement("date_time_selector", "courseto",
            get_string("to"));
15       $mform->addElement("checkbox", "hosts", get_string("
            hosts", "newmodule"));
16       $mform->addElement("submit", "addcourse", get_string("
            addcourse", "newmodule"));
17    }
18 }
19
20 // lokalizace řetězců v lokalizačním souboru!
21 $string["coursename"] = "Název kurzu";
22 $string["hosts"] = "Povolit přístup pro hosty";
23 $string["addcourse"] = "Přidat kurz";
24
25 // zobrazení formuláře
26 $mform = new newmodule_form();
27 $mform->display();
```

Příklad 2: Vytvoření formuláře pomocí třídy `moodleform`



Obrázek 6: Formulář vytvořený pomocí třídy `moodleform`

- `version.php` – soubor definující verzi modulu. Jedná se o nejjednodušší, ale zároveň nejdůležitější soubor. Spoustu akcí Moodle vykonává výhradně při instalaci modulu. Pouze zvýšení jeho verze vyvolá systémovou notifikaci v administraci systému a umožní tak aktualizaci modulu (např. změnu struktury databázových tabulek). Nezbytně nutné však je, abychom měli korektně připravený soubor `upgrade.php` zmíněný výše.
- `view.php` – jedná se o skript, který je při inicializaci modulu vykonán jako první. Zajišťuje zobrazení aktivit modulu a jejich rozhraní.

4.1.4 Práce s databází

V podkapitole 4.1.3 na straně 18 jsme popsali, jak využít XMLDB editor pro vytváření a úpravu databázových tabulek. Tento nástroj je užitečný pro správu souboru `install.xml` inicializovaného při instalaci či aktualizaci modulu.

Dále jsme v popisu souboru `upgrade.php` na straně 19 zmínili využití funkce `xmldb_newmodule_upgrade()` s odkazem na API pro definici dat. Tím získáváme k dispozici funkce, které nám umožňují pracovat se strukturou databáze, respektive databázových tabulek.

Chceme-li však s databází pracovat v interakci s modulem, musíme využít funkce, které nám Moodle nabízí. Jedná se o funkce určené pro práci s daty uloženými v databázových tabulkách – jejich získávání, vkládání, aktualizace. Tato kapitola se bude zabývat právě jimi. Zároveň se nacházíme ve fázi, kdybychom si měli vytvořit databázový model pro modul, který chceme vytvářet.

Databáze systému Moodle by měla být kódována znakovou sadou UTF8. Každá tabulka musí obsahovat pole `id` a je opatřena jednotným prefixem, který si volíme při instalaci systému – defaultně `mod_` [7].

Vycházet budeme z API pro manipulaci s daty, které nám poskytuje komplexní oporu v práci s databázovými záznamy. Připomeňme, že zmíněné DML[14] funkce jsou dostupné od verze Moodle 2.0 a vyšší[15], kde došlo ke změně databázové vrstvy vzhledem k podpoře některých nových funkcí. Podívejme se na základní informace o těchto funkcích [14].

- Všechny funkce, které budeme v této kapitole zmiňovat, jsou veřejnou metodou globálního objektu `$DB`, který je instancí třídy `moodle_database`. Chceme-li v našich funkcích pracovat s databází, musíme tento objekt inicializovat.

```
1 // inicializujeme globální objekt $DB
2 global $DB
```

Příklad 3: DML – inicializace globálního objektu `$DB`

- Parametr `$table` se udává bez prefixu.

```
1 // dotaz vrátí z tabulky user záznam id 5
2 $uzivatel = $DB->get_record("user", array("id" => "5"));
```

Příklad 4: DML – zápis parametru `$table`

- Parametr `$sql`, který využívají funkce končící na `_sql()`, musí obsahovat název tabulky bez prefixu a navíc uzavřený do složených závorek – je totiž součástí zápisu přímého SQL dotazu.

```
1 // dotaz vrátí z tabulky user záznam id 5
2 $uzivatel = $DB->get_record_sql("SELECT * FROM {user} WHERE
     id = ?", array(5));
```

Příklad 5: DML – zápis parametru `$sql`

[14] Data Manipulation Language (jazyk pro manipulaci s daty)

[15] DML funkce pro starší verze systému jsou k dispozici na `http://docs.moodle.org/dev/DML_functions_-_pre_2.0`; detailní popis změn pak nalezneme v migrační dokumentaci na stránce `http://docs.moodle.org/dev/DB_layer_2.0_migration_docs`

- Parametr `$conditions` je dán formou pole (ve tvaru `nazev_pole => hodnota_pole`).

```
1 /* dotaz vrátí z tabulky user záznam, kde firstname je Jakub
2    a lastname Milásek */
3 $uzivatel = $DB->get_record("user", array("firstname" => "
     Jakub", "lastname" => "Milásek"));
```

Příklad 6: DML – zápis parametru `$conditions`

- Parametr `$params` je dán formou pole, kde hodnoty slouží k vyplnění zástupných symbolů v SQL příkazu. Zástupným symbolem může být `?` nebo pojmenovaná hodnota. Pojmenované parametry musí být unikátní.

```
1 /* ? jako zástupný znak - dotaz v obou případech vrátí
2    z tabulky user záznam, kde firstname je Jakub a lastname
3    Milásek */
4 $uzivatel = $DB->get_record_sql("SELECT * FROM {user} WHERE
     firstname = ? AND lastname = ?", array("Jakub", "Milásek
     "));
5
6 // pojmenovaná hodnota jako zástupný řetězec
7 $uzivatel = $DB->get_record_sql("SELECT * FROM {user} WHERE
     firstname = :firstname AND lastname = :lastname", array("
     firstname" => "Jakub", "lastname" => "Milásek"));
```

Příklad 7: DML – zápis parametru `$params` s využitím zástupných symbolů

Funkce, pomocí kterých můžeme získávat data z databáze, lze klasifikovat do tří základních skupin (dotaz dle parametrů, dotaz dle klauzule `WHERE`, dotaz dle SQL) – každá z těchto skupin má zároveň tři návratové typy (`field`, `record`, `records`) podle toho, zda chceme získat hodnotu konkrétního názvu pole z jednoho záznamu, jeden kompletní záznam nebo více záznamů [7]. Parametry předané těmito funkcemi vytváří SQL dotaz typu `SELECT`.

Kromě funkcí pro získávání dat nesmíme opomenout ty, které zajišťují mazání, vkládání a aktualizaci záznamů. Parametry předané těmito funkcemi pak vytváří SQL dotazy typu `DELETE`, `INSERT` a `UPDATE`.

Abychom tyto funkce lépe pochopili, podívejme se nejprve na význam parametrů, které se v nich objevují [14, 15]:

- `$table` – název tabulky dle výše uvedené konvence,
- `$conditions` – pole podmínek; hodnoty jsou spojeny operátorem `AND` a užity do klauzule `WHERE`; parametr je buď volitelný nebo je mu nastavena defaultní hodnota `null`,
- `$fields` – seznam názvů polí oddělených čárkou, které mají být u daného záznamu vypsány; defaultní hodnota parametru je dána zástupným znakem `*` (vypíší se všechny názvy dostupných polí),
- `$strictness` – udává striktnost příkazu zadáním jedné z hodnot:
  - `IGNORE_MISSING` – tzv. kompatibilní režim; pokud dotaz nenalezne záznam, je vrácena hodnota `false`; pokud jich je nalezeno více, je vrácena chybová zpráva; hodnota je zároveň defaultní pro daný parametr,
  - `IGNORE_MULTIPLE` – pokud dotaz nalezne více záznamů, vrátí první z nich a zbytek ignoruje (tato hodnota se nedoporučuje),
  - `MUST_EXIST` – pokud dotaz nenalezne žádný záznam nebo naopak více záznamů, je vyvolána výjimka.
- `$select` – jedná se o fragment SQL kódu užitý do klauzule `WHERE`,
- `$params` – pole SQL parametrů (viz výše v souvislosti se zástupnými symboly); defaultní hodnota pole je nastavena na `null`,
- `$sql` – řetězec dotazu,
- `$values` – pole hodnot, kde jednotlivé hodnoty jsou hledány v názvu pole, které je určené parametrem `$field` (viz níže),
- `$field` – název pole v záznamu (neplést s parametrem `$fields`),
- `$sort` – defaultně prázdný parametr udávající název pole či polí, podle kterých je výsledek řazen,

- `$limitfrom`, `$limitnum` – parametry omezující počet vypsaných záznamů; vrací podmnožinu záznamů začínající od čísla v parametru `$limitfrom` o počtu záznamů v parametru `$limitnum` (např. `4, 20` bude znamenat, že od záznamu s indexem 4 se jich vypíše celkem 20, jsou-li k dispozici),
- `$return` – název pole, které má být vráceno dotazem,
- `$dataobject` – datový objekt s hodnotami pro jeden a více názvů polí v záznamu; podívejme se na příklad vytvoření objektu, který obsahuje základní údaje o uživateli:

```
1 // vytvoření objektu
2 $uzivatel = new stdClass();
3
4 // musíme zadat všechny názvy polí, které nemohou být NULL
5 $uzivatel->username = "milasj00";
6 $uzivatel->firstname = "Jakub";
7 $uzivatel->lastname = "Milásek";
```

Příklad 8: DML – inicializace objektu pro parametr `$dataobject`

- `$returnid` – logická hodnota vrací `id` vloženého záznamu; defaultně je parametr nastaven na `true`,
- `$bulk` – logická hodnota, která je ve většině implementací nevyužita; defaultně je parametr nastaven na `false` (`true` pokud jsou očekávány opakované aktualizace).

Výše jsme si uvedli základní výčet parametrů funkcí, které jsou součástí třídy `moodle_database`. Nyní se podíváme na jednotlivé funkce dle skupin, které jsme avizovali dříve. Zároveň si uvedeme příklady dotazování pomocí vybraných funkcí – princip však bude obdobný pro všechny ostatní [7, 15].

- `SELECT` dotazy dle parametrů – tyto funkce jsou založeny na pouhém zadání parametrů. Jsou vhodné v případech, kdy není nutné dotaz specifikovat částečným nebo úplným SQL dotazem.

- ▷ `get_field($table, $return, array $conditions, $strictness)`
- ▷ `get_record($table, array $conditions, $fields, $strictness)`
- ▷ `get_records($table, array $conditions, $sort, $fields, $limitfrom, $limitnum)`

```
1 /* dotaz vrátí z tabulky user hodnotu pole username
2    u záznamu, kde firstname je Jakub a lastname je Milásek
3    */
4 $uzivatelskeJmeno = $DB->get_field("user", "username", array
     ("firstname" => "Jakub", "lastname" => "Milásek"));
5
6 // vypíše milasj00
7 echo $uzivatelskeJmeno;
```

Příklad 9: DML – `get_field()`

- `SELECT` dotazy dle klauzule `WHERE` – jedná se o funkce, které končí řetězcem `_select`. Zároveň obsahují parametr `$select`, do kterého zadáváme část SQL dotazu užitého v klauzuli `WHERE`, jak jsme již zmínili výše.
  - ▷ `get_field_select($table, $return, $select, array $params, $strictness)`
  - ▷ `get_record_select($table, $select, array $params, $fields, $strictness)`
  - ▷ `get_records_select($table, $select, array $params, $sort, $fields, $limitfrom, $limitnum)`

```
1 /* dotaz vrátí z tabulky license hodnotu pole fullname
2    a source u záznamu, kde shortname je cc nebo fullname je
3    Creative Commons; porovnávané hodnoty jsme zapsali přímo
4    do dotazu - nevyužili jsme tedy zástupné symboly, tudíž
5    pole s parametry necháme prázdné */
6 $select = "shortname = 'cc' OR fullname = 'Creative Commons
     '";
```

```
7  $licence = $DB->get_record_select("license", $select, array
       (), "fullname, source");
8
9  // vypíše Creative Commons, http://creativecommons.org/
       licenses/by/3.0/
10 echo $licence->fullname . ", " . $licence->source;
```

Příklad 10: DML – `get_record_select()`

- `SELECT` dotazy dle SQL – funkce vychází z SQL dotazu užitého do klauzule `WHERE`, který předáváme parametrem `$sql`. Název tabulky v dotazu musí být bez prefixu a uzavřený do složených závorek, jak jsme již uvedli na straně 23.
  - ▷ `get_field_sql($sql, array $params, $strictness)`
  - ▷ `get_record_sql($sql, array $params, $strictness)`
  - ▷ `get_records_sql($sql, array $params, $limitfrom, $limitnum)`

```
1  /* dotaz vrátí z tabulky assign hodnoty polí name a intro
2     u záznamů, kde hodnota pole course je 2; výsledné pole
3     objektů bude obsahovat maximálně prvních 10 záznamů */
4  $sql = "SELECT name, intro FROM {assign} WHERE course = ?";
5  $ukoly = $DB->get_records_sql($sql, array(2), 0, 10);
6
7  // cyklem vypíšeme úkoly kurzu č. 2 ve tvaru Název - popis
8  foreach($ukoly as $ukol){
9     echo $ukol->name . " - " . $ukol->intro . "<br />";
10 }
```

Příklad 11: DML – `get_records_sql()`

- `DELETE` dotazy – umožňují nám smazat požadované záznamy. Využít můžeme dvě funkce.
  - ▷ `delete_records($table, array $conditions)`
  - ▷ `delete_records_select($table, $select, array $params)`

```
1 // dotaz odstraní tag z tabulky tag, kde userid je 5 a name
     je úkol
2 $DB->delete_records_select("tag", "userid = 5 AND name = '
     úkol'");
```

Příklad 12: DML – `delete_records_sql()`

- `INSERT` dotaz – slouží pro vkládání záznamů do databáze.

  ▷ `insert_record($table, $dataobject, $returnid, $bulk)`

```
1 // připavíme vkládaný objekt
2 $blok = new stdClass ();
3 $blok->name = "etask_settings";
4 $blok->version   = 2013021300;
5
6 // provedeme samotné vložení objektu do tabulky block
7 $DB->insert_record("block", $blok);
```

Příklad 13: DML – `insert_record()`

- `UPDATE` dotaz – aktualizuje požadovaný záznam. V datovém objektu musíme specifikovat jedinečnou hodnotu pole `id`. To určí, který záznam má být aktualizován. Jak jsme již uvedli na straně 22, toto pole je povinné pro každou databázovou tabulku systému Moodle.

  ▷ `update_record($table, $dataobject, $bulk)`

```
1 // připavíme aktualizační objekt
2 $kurz = new stdClass ();
3 $kurz->id = 2;
4 $kurz->fullname = "Programování v PHP I";
5
6 // provedeme samotnou aktualizaci v tabulce course
7 $DB->update_record("course", $kurz);
```

Příklad 14: DML – `update_record()`

K dispozici jsou některé další funkce, jako např. práce se sadou záznamů či transakce. Zmiňme také ty, které vrací počet záznamů dle zadaných kritérií nebo ověřující existenci záznamu.

Využít můžeme i pomocné funkce umožňující práci s operátorem `LIKE`, zjišťující délku řetězce a mnoho dalších – jejich popis nalezneme v dokumentaci na stránce `http://docs.moodle.org/dev/Data_manipulation_API`. Nejlepší cestou však je, podívat se přímo do souboru s těmito funkcemi – najdeme ho v adresáři `lib/dml/moodle_database.php`. Využít můžeme také dokumentaci třídy `moodle_database` [15].

4.1.5 Naprogramování modulu

V tuto chvíli byly realizovány následující kroky:

- stanovili jsme si požadavky na modul a udělali jeho návrh,
- zvolili si typ modulu a jeho šablonu,
- zorientovali se v základní struktuře souborů a jejich kódu,
- seznámili se s objekty pro práci s databází Moodle a navrhli datový model.

Zároveň jsme si uvedli rozšiřující odkazy na dokumentaci funkcí, které můžeme případně v našem modulu využít. Dostali jsme se tedy do etapy, kdy jedním z posledních kroků je naprogramovat modul, který jsme navrhli. Jeho realizace vychází z výše popsaných postupů.

V každém projektu je důležitý konzistentní styl kódování a to obzvláště, pokud se na něm podílí více vývojářů. Moodle zavádí standardní styl kódování. To umožňuje jednoduchost, čitelnost a porozumění kódu [16].

Pro ověření, zda je náš kód v souladu s pravidly stylu kódování systému Moodle, můžeme využít dva nástroje.

- Code checker[16]
- Moodle PHPdoc checker[17]

[16] `https://moodle.org/plugins/view.php?plugin=local_codechecker`
[17] `https://github.com/marinaglancy/moodle-local_moodlecheck`

Jedná se o standardní rozšíření, která umístíme do adresáře `moodle/local/` a provedeme klasickou instalaci těchto modulů ve správě stránek, jak je popsáno níže v kapitole 4.1.6. K dispozici jsou přes menu Správa stránek → Vývoj. Oba nástroje kontrolují zdrojový kód mírně odlišným způsobem, proto je doporučeno využít oba.

4.1.6 Instalace modulu

Z předchozího kroku bychom měli mít naprogramovaný modul – poskytovat ho budeme v komprimované složce. Podívejme se nyní na to, jak probíhá instalace nového modulu a ověřme si, zda jeho instalace proběhla správně [1].

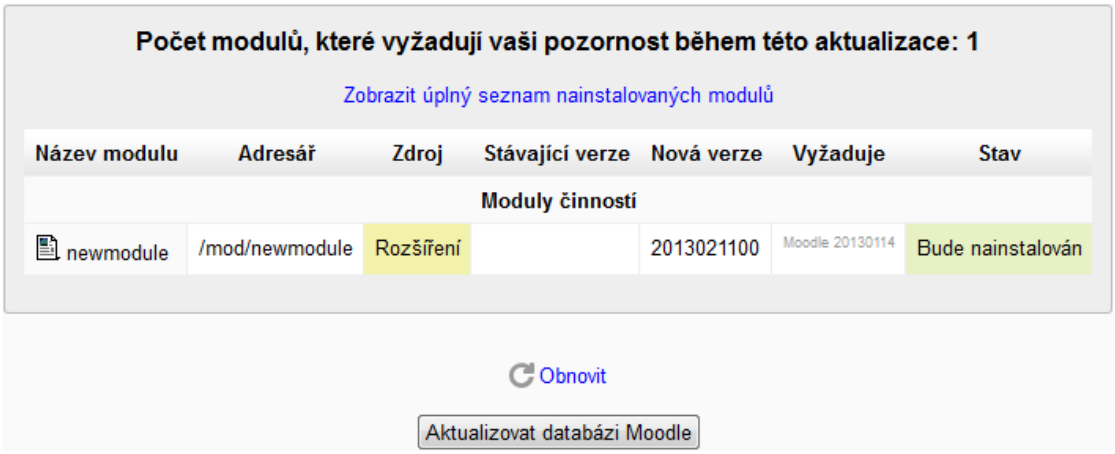
1. Extrahujeme komprimovanou složku s modulem, který chceme do systému Moodle instalovat.

2. Složku modulu nakopírujeme do adresáře, který je obvykle určen souborem `README.txt`.

3. Přihlásíme se do administrace systému, kde jsme obvykle informováni o dostupnosti nového modulu a možnosti jeho instalace, jak vidíme na obrázku 7 níže. Pokud na instalaci modulu nejsme upozorněni automaticky, nalezneme ji v menu Správa stránek → Informace. V závislosti na typu modulu může jeho instalace probíhat odlišně. Pokud aktualizace databáze Moodle proběhne v pořádku, jsme informováni o úspěšné instalaci modulu.

4. Po instalaci modulu jej můžeme plnohodnotně využívat. Je-li k dispozici nastavení modulu, lze jej modifikovat.

5. Případnou odinstalaci modulu provádíme z menu Správa stránek → Moduly → Přehled modulů. Složku modulu v adresářové struktuře pak musíme odstranit manuálně.



Obrázek 7: Kontrola zásuvných modulů

4.1.7 Testování

Nezbytnou součástí vývoje každého modulu je jeho testování. Dříve než uvolníme výsledný produkt ke stažení, musíme provést testování modulu. V závislosti na funkcionalitě modulu provedeme jeho patřičné testování.

Vše co modul má umět, musí umět v ideálním případě bez chyb. Otestovat také musíme jeho přenositelnost mezi různými verzemi systému Moodle. Konkrétně se zaměříme na přenositelnost mezi verzemi od Moodle 2.0 výš.

4.1.8 Dokumentace

Máme-li modul naprogramovaný a otestovaný, měli bychom sepsat jeho dokumentaci. Ta může být součástí souboru `README.txt`. Jednat se může také o externí a rozsáhlejší dokument. V rámci dokumentace je vhodné zohlednit právě problematiku kompatibility modulu mezi různými verzemi systému Moodle a případné možnosti řešení nekompatibility zde popsat.

Dokumentací se rozumí i řádné komentování funkcí zdrojového kódu a to podle standardního kódovacího stylu užívaného systémem Moodle, o kterém jsme se zmínili v kapitole 4.1.5 na straně 30. Z těchto komentářů lze snadno vygenerovat dokumentaci tříd a metod, která díky své přehlednosti usnadní další vývoj modulu. Zda máme funkce korektně dokumentovány můžeme zjistit díky nástroji Moodle PHPdoc checker uvedeném též na straně 30.

4.2 Modul eTask

Pro ověření metodiky vývoje modulů v LMS Moodle byl zvolen modul eTask, který bude vyvíjen v souladu s obecnou metodikou popsanou v kapitole 4.1. Tento modul bude zajišťovat přehledné hodnocení studentů v tabulkovém uspořádání.

Učitelé využívající systém Moodle často poukazují na problém s přehledností v oblasti hodnocení aktivit studentů, jako jsou např. úkoly. V kapitole 3.1 na straně 11 byla naznačena nesourodost hodnotícího systému, která způsobuje jeho nepřehlednost. Nedostatky, které zde systém Moodle vykazuje, odstraňuje modul eTask. Jeho analýzou a realizací se budeme zabývat v následující kapitole.

Filozofií modulu je poskytnout přehled hodnocení úkolů všech studentů. Tabulka, která toto umožňuje, bude umístěna na viditelném a snadno dostupném místě. Učitel i studenti uvidí komplexní přehled hodnocení. Nabízí se otázka, zda by měli mít studenti přístup ke sledování výsledků ostatních. Současné řešení známkování v Moodle vychází z faktu, že učitel vidí hodnocení všech studentů ve všech aktivitách, student vidí pouze své. Modul eTask je záměrně řešen tak, aby učitel i student viděli komplexní hodnocení úkolů. Důvodem je jednak přehlednost pro obě zmíněné role, ale především také motivační faktor pro studenty. Skutečnost, že mohou vidět, kdo úkol odevzdal a jaké má hodnocení, může mít významný vliv na jejich vnější motivaci. Modul eTask poskytuje signalizaci v rámci tabulkového výpisu. Buňka, která bude obsahovat hodnocení, může nabývat třech různých podbarvení – oranžové (úkol odevzdán), zelené (úkol odevzdán a úspěšně splněn) a červené (úkol odevzdán a nesplněn). Tabulkový výpis poskytuje okamžitou interaktivitu – učitel se může přímo dostat k prohlížení, editaci a hodnocení úkolu.

5 Návrh vlastního řešení

Tato kapitola se zabývá analýzou, návrhem a vlastním řešením modulu eTask. Postupy vychází z výše navržené metodiky a budou, společně s výsledky praktické činnosti na vývoji modulu, popsány v jednotlivých podkapitolách této části diplomové práce.

5.1 Analýza řešení

Modul eTask je nutné umístit tak, aby byl v rámci kurzu okamžitě viditelný a k dispozici. V zásadě není mnoho možností, jak toho docílit. Lze vytvořit modul typu blok – zde ovšem nastává zásadní problém. Modul by sice byl snadno dostupný, ale šířka tohoto bloku neumožňuje využití takového nástroje. Nevystačíme si ani s modulem činností, který by na stránku zobrazující modul eTask pouze odkazoval ze zvoleného tématu kurzu. Další otázkou je implementace samotné mapy zobrazující přehledné hodnocení. Je nutné analyzovat možnost využití modulu pro správu známek, který vidíme na obrázku 3 na straně 13.

5.1.1 Uspořádání kurzu

Po analýze jednotlivých řešení implementace modulu eTask, se jako jediné vhodné ukazuje užití modulu určujícího uspořádání kurzu. Toto uspořádání totiž definuje, jakým způsobem se bude zobrazovat hlavní (centrální) část stránky. Moodle nabízí čtyři základní formáty pro uspořádání kurzu: SCORM, diskusní, tématické a týdenní uspořádání. Poslední dvě jsou si podobná a zároveň nejvíce používaná [7]. Důležitý je ovšem fakt, že uspořádání kurzu umožňuje zasahovat do jeho centrální části. Tím dokážeme splnit podmínku stanovenou výše – aby byl modul okamžitě viditelný a k dispozici.

Realizace modulu eTask bude, na základě předchozí analýzy, vycházet z tématického uspořádání kurzu. Jednotlivá témata budou obsahovat aktivity zanesené do tabulkového přehledu vyvíjeného modulu. Využít by bylo možné i týdenní uspořádání – to je však méně časté než tématické. Ne každý kurz lze totiž realizovat v přesně definovaném týdenním intervalu a obvykle se jeho tvůrci uchylují právě k tématickému uspořádání, které je dané počtem témat, nikoliv týdnů.

Pro vytvoření nového tématického uspořádání můžeme využít metodu nakopírování a modifikace existujícího formátu kurzu. Následujícím postupem lze získat základ pro nový modul eTask [17].

1. Zkopírovat složku tématického uspořádání `course/format/topics/`.
2. Složku přejmenujeme názvem nového uspořádání kurzu, tedy `etask`[18].
3. Nahradíme původní název formátu novým ve všech skriptech.
4. Nové uspořádání kurzu je připraveno k další modifikaci.

Jak se však ukázalo, tato metoda nemusí být ideální. Důvodem je nekompatibilita uspořádání kurzu mezi jednotlivými verzemi Moodle. Značné změny se objevují ve verzích nižších 2.3, 2.3 a vyšších 2.4 – pokud bychom využili nakopírování tématického uspořádání kurzu, dostali bychom modul dostupný pro verzi 2.4. Verze nižší by musely být vytvořeny odpovídající modifikací tohoto modulu.

Další řešení vychází z možnosti, že si v našem modulu zavoláme soubor `format.php` umístěný v tématickém uspořádání kurzu[19]. Tím přebíráme aktuální uspořádání dostupné v dané verzi systému. Problém však nastává při použití navázaných souborů, jako jsou knihovny, javascript či styly – ty se implementují nekorektně. Podstatné jsou i názvy funkcí, které obsahují ve svém názvu `topics` místo `etask`. Zpočátku ideální řešení se po značném testování v rozdílných verzích systému Moodle ukázalo jako nepoužitelné. Nefunkčnost se týkala nejen vizuální složky, ale především funkční. To má na kvalitu modulu zásadní vliv. Využita tedy byla metoda nakopírování a modifikace existujícího formátu kurzu.

5.1.2 Modul pro správu známek a jeho využití

Mnoho aktivit v prostředí Moodle lze hodnotit bodově nebo dle dostupných škál. Každý z kurzů má přístup k nástroji, který toto hodnocení zajišťuje a poskytuje jeho přehled. Je-li určitá činnost studenta ohodnocena, objeví se i v celkovém přehledu. Přehled nalezneme přes menu Nastavení → Správa kurzu → Známky [18].

[18] Název modulu nesmí obsahovat velká písmena!
[19] `course/format/topics/format.php`

Odvolejme se opět na obrázek 3 na straně 13, který znázorňuje hodnocení účastníků v tabulkovém výpisu. Jedná se o modul známkování, který je standardní součástí systému Moodle. Strukturou a uspořádáním téměř vyhovuje požadavkům na modul eTask. Funkční hledisko ovšem nikoliv.

Základem výpisu je tabulka. Její záhlaví obsahuje v prvním řádku název kurzu. Ve druhém nalezneme příjmení a jméno účastníka kurzu, jeho e-mailovou adresu a jednotlivé aktivity, které jsou součástí hodnocení. Tabulka nám nabízí tři různá zobrazení [19]:

- úplný režim – zobrazí všechny známky včetně souhrnných,
- pouze souhrnné známky – skryje dílčí známky a zobrazí jen sloupec se souhrnnými známkami,
- pouze známky – zobrazí dílčí známky bez souhrnných známek.

Výše uvedená zobrazení spadají do celkového přehledu ve správě známek. Pokud však prozkoumáme adresář `grade/report/`, zjistíme, že kromě celkového přehledu existují ještě tzv. očekávané výstupy a výsledné známky ze všech kurzů. Cílem další analýzy je právě celkový přehled, nacházející se v adresáři `grade/report/grader/`.

Při analýze a testování modulu bylo zjištěno, že se jedná o rozsáhlou a poměrně složitou část aplikace. Zejména její závislosti a návaznosti funkcí nám neumožní snadnou implementaci v rámci modulu eTask, který vychází z tématického uspořádání kurzu. Pokročilé funkce, které tento modul nabízí, jsou pro nás nevyužitelné. Jejich využitelnost je opodstatnitelná v samotném modulu známkování. Cílem vyvíjeného modulu však není poskytovat tyto pokročilé funkce, ale umožnit jednoduchý a přehledný systém hodnocení úkolů účastníků kurzu a usnadnit tak práci lektorům a poskytnout motivační prvek studentům.

Zmiňme také otázku hodnocení, kdy modul eTask vyžaduje barevné odlišení odevzdaného, splněného a nesplněného úkolu. Modul pro správu známek obsahuje ve své databázové tabulce `grade_items`[20] pole nazvané `gradepass`, které udává mezní hodnotu pro splnění dané aktivity. K dispozici máme další dvě důležitá pole, kterými jsou `grademax` a `grademin`. Určují maximální a minimální možné hodnocení. Jejich hodnotou je číslo a všeobecně platí, že `grademin` $\leq$ `gradepass` $\leq$ `gra-`

[20] uvedeno bez prefixu tabulky, defaultně `mdl_grade_items`

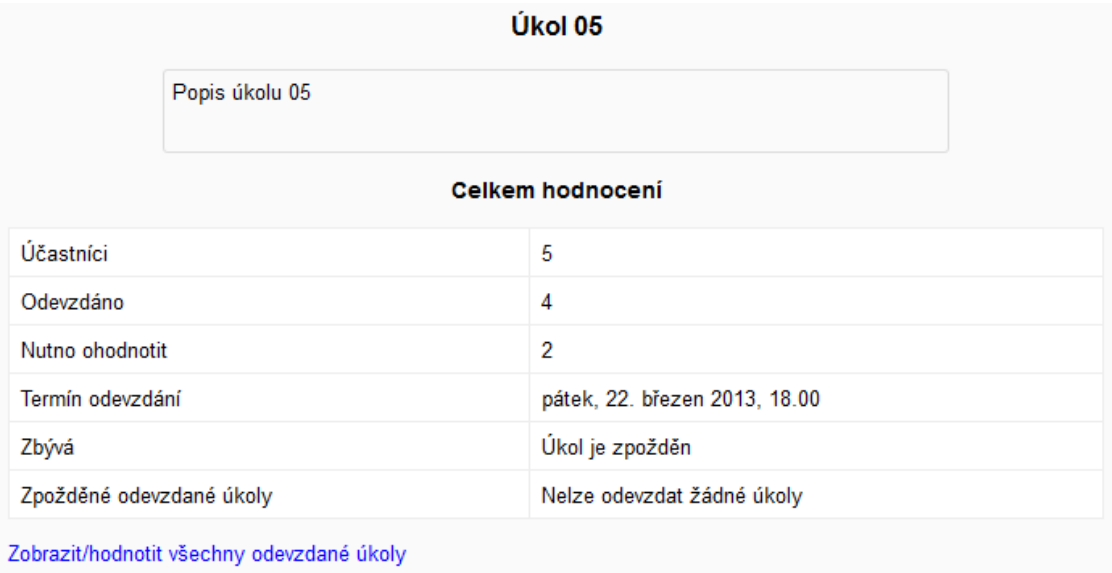
`demax` [20]. Problém ovšem nastává při použití jiných škál – vlastních. Uveďme si jednoduchý příklad. Vytvoříme si logickou škálu `Výborně, Velmi dobře, Dobře, Neprospěl`. Této škále odpovídají hodnoty `1`, `2`, `3` a `4`. To jde ovšem proti filozofii systému Moodle. Hodnotící systém totiž předpokládá, že čím vyšší hodnocení, tím lepší výsledek – ovšem u nás je hodnota `4` nejhorší, nikoliv nejlepší. Systém tedy nedokáže patřičně vyhodnotit, která hodnota je mezní pro splnění. V takto definované škále tedy není možné splnit podmínku $4 \leq 3 \leq 1$, která vychází z výše zmíněného principu. Řešení je snadné. Abychom docílili, že hodnocení `Neprospěl` je pro účastníka neúspěšným plněním aktivity, musí být škála řazena vzestupně, tedy `Neprospěl, Dobře, Velmi dobře, Výborně`. Tuto skutečnost však málokdo při vytváření škál bere v potaz i přesto, že je v lokální nápovědě vzestupné řazení popsáno.

Na základě uvedených skutečností budou využity dílčí funkce modulu pro správu známek. Jejich funkčnost usnadní vytvoření modulu eTask.

5.1.3 Hodnocení úkolů

Cílem je, aby hodnocení úkolů bylo co nejjednodušší. Jako neefektivní se jeví současné řešení, kdy je zapotřebí projít přes základní informace o úkolu do přehledu s hodnocením všech studentů a následně zobrazit samotné hodnocení studenta v daném úkolu. Tyto tři kroky vidíme postupně na obrázcích 8, 9 a 10. Na hodnocení studenta, jak jej vidíme na obrázku 10 na straně 39, se však chceme dostat přímým odkazem z tabulky modulu eTask.

Při analýze řešení je nutné brát v potaz fakt, že na samotné hodnocení je vázáno mnoho dalších procesů – propojení na modul známkování, samotné akce v modulu úkolů či zasílání notifikací e-mailem. Není tedy vhodné vytvářet pro tuto činnost vlastní formulář.



Úkol 05

Popis úkolu 05

Celkem hodnocení

Účastníci	5
Odevzdáno	4
Nutno ohodnotit	2
Termín odevzdání	pátek, 22. březen 2013, 18.00
Zbývá	Úkol je zpožděn
Zpožděné odevzdané úkoly	Nelze odevzdat žádné úkoly

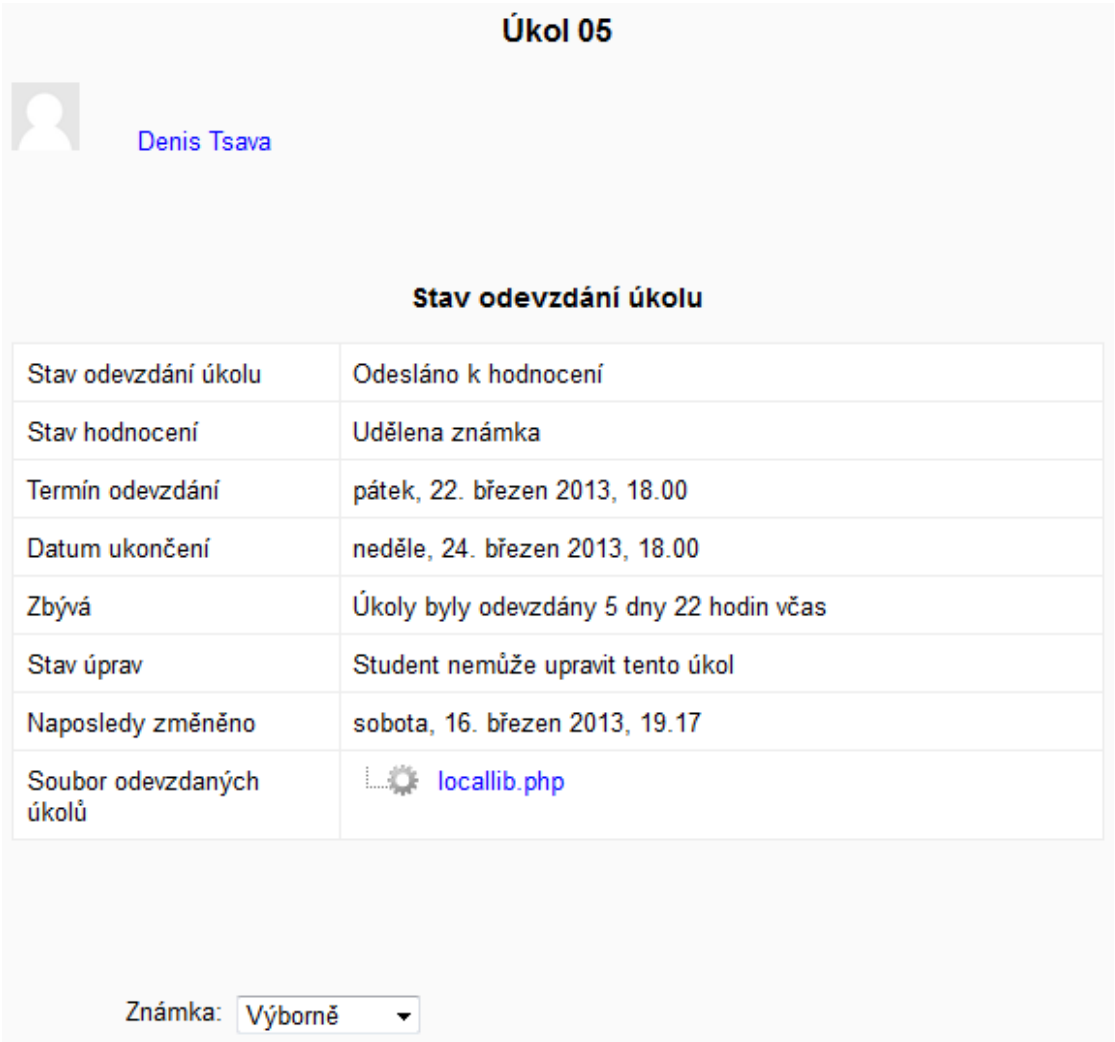
Zobrazit/hodnotit všechny odevzdané úkoly

Obrázek 8: Základní informace o úkolu

Úkol 05

Výběr	Křestní jméno / Příjmení	Známka	Výsledná známka
	Martin Drlík	Výborně	Výborně
	Jakub Milásek	-	-
	Petr Snadný	-	-
	Denis Tsava	Výborně	Výborně
	Anna Vzorová	-	-

Obrázek 9: Přehled s hodnocením všech studentů v daném úkolu



Obrázek 10: Hodnocení studenta v daném úkolu

Nabízí se dvě základní varianty řešení. První z nich je odkazovat patřičnou buňku tabulky v modulu eTask přímo na hodnocení úkolu konkrétního uživatele. Druhou možností je vytvořit překryvné okno, které převezme obsah hodnotící stránky dle parametrů adresní řádky. Překryvné okno lze realizovat více způsoby – přímým vložením do stránky, tzv. popup oknem nebo pomocí elementu `iframe`.

Ve všech případech využíváme URL stránky, které určuje hodnoceného studenta a úkol. Zde nastává první zásadní problém. Podíváme-li se na tabulku s hodnocením studentů, vidíme, že není primárně řazena dle příjmení uživatele, jak by bylo očekáváno. Defaultní řazení se neprakticky řídí parametrem `id` v databázové

tabulce `user`. To má zásadní vliv na určení uživatele, kterého chceme hodnotit. Při podrobnější analýze totiž zjistíme, že odkaz na hodnocení úkolu má následující tvar: `/mod/assign/view.php?id=16&rownum=3&action=grade`. Parametr `id` určuje instanci úkolu. Pro nás je však podstatný parametr `rownum`. Ten udává index řádku tabulky – počítán je od nuly. V modulu eTask jsme také schopni indexovat řádky tabulky, nicméně řazení studentů probíhá podle příjmení a nelze jej měnit. Rozdílné řazení v těchto dvou tabulkách zapříčiní nesouhlasné indexy řádků a tím nekorektní přechod na hodnotící stránku, neboť indexy řádků zůstávají, ale pořadí uživatelů se mění. Skutečnost, že uživatel není určen parametricky, je v zásadě nepochopitelná. Řazení tabulky se uchovává v objektu, který je uložen v proměnné `SESSION`. Stránka s hodnotícím formulářem následně obsahuje skryté pole, kde jsou patřičně řazené číselné identifikátory všech studentů daného úkolu – index řádku pak určuje index identifikátoru v poli. Uveďme si příklad. Skryté pole formuláře obsahuje řetězec `2, 3, 4, 5, 6` – jedná se o pole uživatelů určených pomocí `id` v databázové tabulce `user`. Máme-li v parametru `rownum` hodnotu `3`, znamená to, že budeme hodnotit uživatele, jehož `id` odpovídá číslu `5`[21]. Problém vyřešíme, když budeme uvedenou tabulku defaultně řadit podle příjmení – tím sjednotíme řazení studentů a indexy řádků v hodnotící tabulce a tabulce modulu eTask.

V rámci konzistentnosti modulu pro správu úkolů bylo stanoveno, že problematika hodnocení úkolů nebude řešena překryvným ani popup oknem. Stanovisko vzešlo po dlouhodobé analýze jednotlivých řešení – ta se ukázala jako neefektivní z funkčního i časového hlediska. Moodle v současné době neposkytuje dostatečnou oporu pro uvedená řešení – tomu nasvědčuje fakt, že využitelnost překryvných oken v rámci systému je takřka nulová. Javascriptová knihovna YUI[22], která je systémem implementována, vykazuje navíc značnou nesourodost v těch částech aplikace, které překryvná okna využívají. Vzhled těchto prvků by měl být sjednocen od verze systému 2.5. Vzhledem k výše uvedeným skutečnostem bude přechod na hodnocení úkolu realizován přímým odkazem z patřičné buňky tabulky. Další vývoj aplikace ukáže, do jaké míry budou realizovatelná řešení, která jsou nyní neefektivní.

[21] Při řazení studentů podle příjemní může pole vypadat takto: `2, 5, 4, 6, 3` – pokud tedy opět bude mít parametr `rownum` hodnotu `3`, hodnotíme uživatele s `id 6`!

[22] `http://yuilibrary.com/`

5.1.4 Východiska realizace

Nový modul, vzhledem k výše uvedeným možnostem a jejich analýze, bude vycházet z metody nakopírování a modifikace existujícího formátu kurzu, kterým je týdenní uspořádání. To sice přináší problém s kompatibilitou mezi jednotlivými verzemi, nicméně jde o řešení, které klade důraz na kvalitu a korektní funkčnost celého modulu.

Jako nevhodné se ukázalo řešení přejmout a modifikovat část modulu pro správu známek. Jako ideální řešení se nabízí realizovat tuto část od prvopočátku s případným využitím dílčích funkcí celkového přehledu známek.

Přes možná rizika při zadávání vlastních škál bude využito existující databázové pole pro určení mezní hodnoty splnění dané aktivity. Modul eTask tedy bude založen na existujícím databázovém schématu bez nutnosti vytvářet dodatečné databázové tabulky.

Hodnocení úkolu bude zajištěno přímým odkazem na stránku s hodnocením studentského úkolu tak, jak ji poskytuje systém Moodle ve svém modulu.

5.2 Datový model

Datový model systému Moodle je značně rozsáhlý[23] a v současné verzi 2.4 jej tvoří 305 databázových tabulek. Modul eTask využívá pouze minimální počet z celkového množství.

Při realizaci modulu jsme vycházeli z tématického uspořádání kurzu, implementovali dílčí funkce modulu známkování a využili stávající hodnotící systém modulu pro správu úkolů. Nebudeme zde však popisovat samotný formát kurzu – jeho uspořádání, z kterého vycházíme. Soustředit se budeme právě na modul eTask, který je do tohoto tématického uspořádání implementován. Jeho filozofie vychází z již existujících databázových tabulek a polí, které systém Moodle nabízí. Samotný modul tedy nebude, při své instalaci či aktualizaci, vytvářet žádné nové databázové tabulky.

Modul eTask využívá ve svých funkcích přímý přístup k následujícím databázovým tabulkám[24]:

[23] Databázové schéma Moodle 2.2 – `http://www.examulator.com/er/moodle22_erd.png`

[24] Název tabulek je uveden s defaultním prefixem `mdl_`

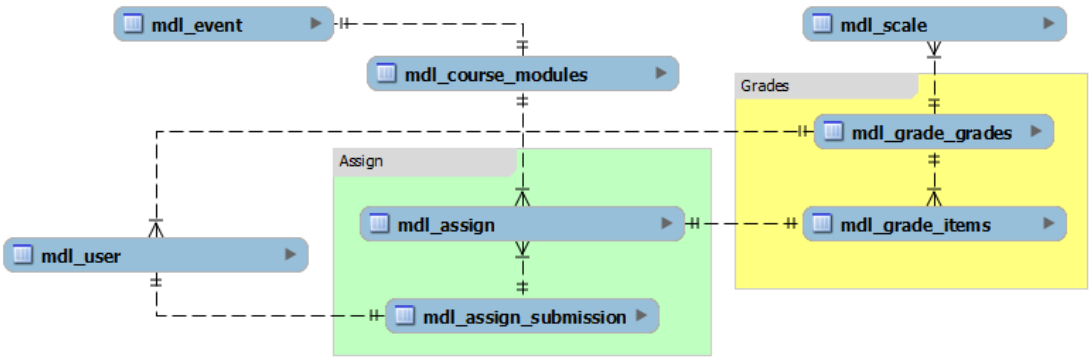
- `mdl_assign` – obsahuje základní informace o úkolu,
- `mdl_assign_submission` – udává status odeslání úkolu,
- `mdl_course_modules` – poskytuje `id` úkolu podle jeho `instance`,
- `mdl_event` – umožní zjistit viditelnost úkolu,
- `mdl_grade_items` – možnost nastavit pole `gradepass` (potřebnou známku),
- `mdl_scale` – předkládá uživatelem definované hodnotící škály.

Další části datového modelu jsou zajištěny pomocí funkcí, které systém Moodle nabízí – jejich využití v rámci modulu eTask bude popsáno v kapitole 5.4 na straně 44. Jde především o tabulku poskytující informace o hodnocení studenta v rámci daného úkolu a dále o tabulku pro získání uživatelů v kurzu:

- `mdl_grade_grades` – poskytuje hodnocení studentů,
- `mdl_user` – umožní získat informace o uživatelích v kurzu.

Zmiňme také fakt, že hodnocení úkolů je zároveň obsaženo v databázové tabulce `mdl_assign_grades`. Využití tabulky `mdl_grade_grades` je však z hlediska užitelnosti efektivnější – poskytuje nám komplexnější informace a lze ji zpracovat předdefinovanou funkcí systému Moodle.

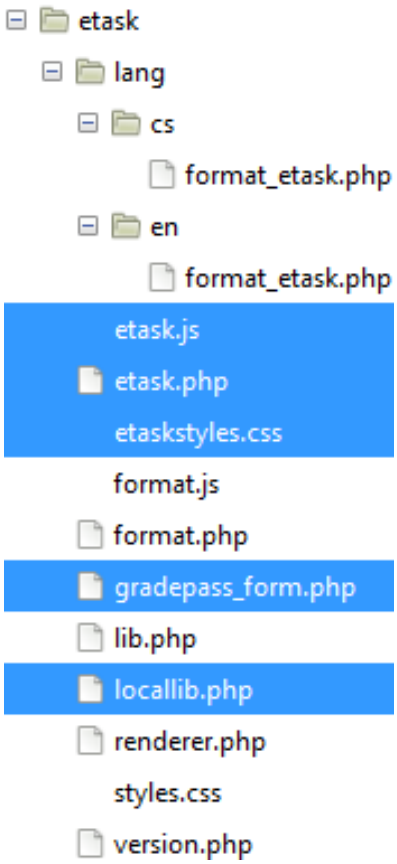
Výše uvedené informace vedou k následujícímu základnímu datovému modelu, který znázorňuje závislosti mezi jednotlivými databázovými tabulkami modulu eTask.



Obrázek 11: Základní datový model

5.3 Adresářová struktura

Umístění modulu v rámci adresářové struktury je dáno jeho typem, jak znázorňuje tabulka 1 na straně 15. Modul eTask vychází z uspořádání kurzu a umístěn je v adresáři `moodle/course/format/etask/`[25]. Odpovídá mu následující adresářová struktura.



Obrázek 12: Struktura modulu eTask

Pokud nebereme v úvahu lokalizační soubory, výše uvedená adresářová struktura obsahuje v zásadě dva typy souborů. Jsou to jednak původní soubory převzaté s formátem uspořádání kurzu a dále pak vlastní soubory modulu eTask, který je do tématického uspořádání implementován a jeho skripty jsou zvýrazněny. Toto řešení přispívá k přenositelnosti modulu mezi staršími verzemi systému Moodle i přes nutnost dílčích modifikací.

[25] Kořenový adresář instalace je `moodle`.

5.4 Aplikační logika

V této kapitole se budeme zabývat jednotlivými soubory, které jsou součástí adresářové struktury modulu eTask uvedené v podkapitole 5.3 na předchozí straně.

Veškeré skripty modulu eTask se řídí tzv. kódovacím stylem systému Moodle, který jsme zmínili v kapitole 4.1.5 na straně 30 a přispívá k celkové kvalitě kódu.

- `lang/` – adresář s lokalizačními soubory, které jsou umístěny v podadresářích jednotlivých jazykových balíčků. Modul je plně lokalizován do českého a anglického jazyka.
  - ▷ `cs/format_etask.php` – podadresář obsahující soubor s českou lokalizací řetězců uspořádání kurzu a modulu eTask.
  - ▷ `en/format_etask.php` – podadresář obsahující soubor s anglickou lokalizací řetězců uspořádání kurzu a modulu eTask.
- `etask.js` – soubor obsahuje funkce založené na javascriptové knihovně YUI. Význam funkcí popisuje podkapitola 5.4.1.
- `etask.php` – jedná se o hlavní soubor implementace modulu eTask. Vkládá potřebné knihovny – externí (v rámci systému Moodle) i lokální (v rámci modulu eTask). Stará se o zobrazení modulu v uspořádání kurzu. Samotný soubor je volán v hlavním skriptu uspořádání kurzu – `format.php` (viz níže).
- `etaskstyles.css` – soubor zajišťující vzhled modulu eTask. Žádným způsobem nezasahuje do standardního vzhledu uspořádání kurzu, které určuje soubor `styles.css` (viz níže).
- `format.js` – soubor poskytující dynamické funkce založené na javascriptové knihovně YUI. Neobsahuje funkce modulu eTask – ty jsou vedeny odděleně v souboru `etask.js` (viz výše).
- `format.php` – slouží pro zobrazení kurzu v tématickém uspořádání, které je tvořeno jednotlivými moduly.
- `gradepass_form.php` – soubor obsahující třídu a metody vykreslující formulář pro nastavení potřebné známky u úkolů modulu eTask.

- `lib.php` – knihovna obsahující hlavní třídu se základními funkcemi pro dané uspořádání kurzu.
- `locallib.php` – knihovna lokálních funkcí, které zajišťují korektní chod modulu eTask a jeho vykreslení. Jednotlivé funkce této knihovny jsou popsány v podkapitole 5.4.2 na straně 47.
- `renderer.php` – renderer zajišťující výstup pro daný formát kurzu.
- `styles.css` – základní soubor určující vzhled uspořádání kurzu.
- `version.php` – standardní soubor určující verzi modulu, požadovanou verzi systému Moodle, od které je možné modul instalovat a samotný název modulu.

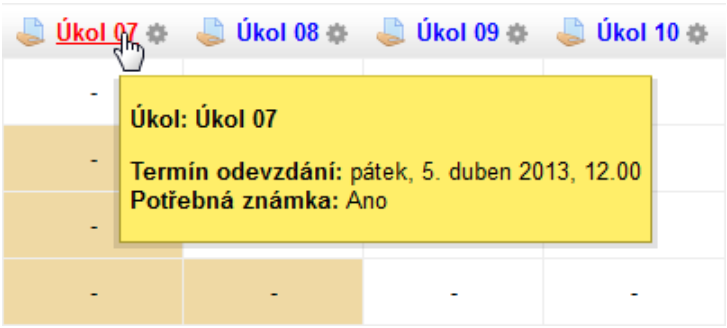
5.4.1 Funkce knihovny `etask.js`

Soubor obsahuje dvě funkce, které modul eTask bezpodmínečně využívá – obě jsou založeny na jádře knihovny YUI, které jsou součástí systému Moodle.

- První z nich vytváří tzv. tooltip. Ten je aplikován na atribut `title` požadovaných odkazů. Funkce využívá YUI Tooltip widget, který je rozšířením základní třídy Widget [21, 22].

```
1 /* Nastaví tooltip na atribut title v odkazech hlavičky
     eTask tabulky */
2 YUI().use("yui2-yahoo-dom-event", "yui2-animation", "yui2-
     container",
3     function(Y) {
4     var YAHOO = Y.YUI2;
5     var elements = YAHOO.util.Dom.getElementsByClassName("
        etasktooltip", "a");
6     var tooltip = new YAHOO.widget.Tooltip("tooltip",
7         {context: elements, autodismissdelay: 60000
8     });
9 });
```

Příklad 15: YUI – Tooltip widget

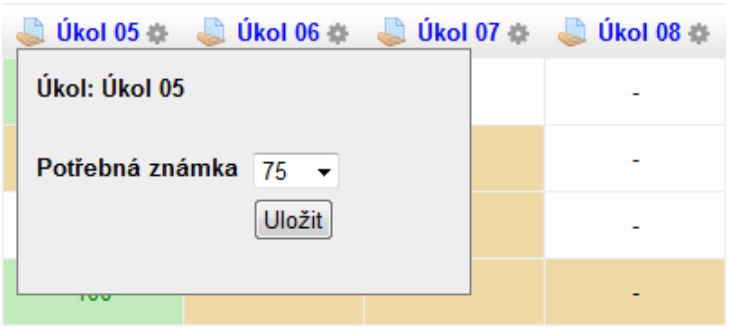


Obrázek 13: YUI – Tooltip widget

- Druhá funkce, `toggle(id)`, využívá metodu `toggleView()` knihovny YUI, která je součástí třídy `NodeList`. Funkce zobrazí nebo skryje uzel, tedy formulář nastavení potřebné známky pro splnění úkolu. Obalující prvek formuláře je jednoznačně identifikován pomocí atributu `id`, který je předán parametrem funkce. Akce je vyvolána na kliknutí [23].

```
1  // Zobrazí a skryje formulář potřebné známky
2  function toggle(id){
3      var id = id;
4      var style = document.getElementById(id).style.display;
5      YUI().use("node", function(Y) {
6          if(style=="none"){
7              Y.all(".gradesettings").hide();
8          }
9          Y.one("#"+id).toggleView();
10     });
11 }
```

Příklad 16: YUI – `toggle(id)` s využitím metody `toggleView()`



Obrázek 14: YUI – `toggle(id)` s využitím metody `toggleView()`

5.4.2 Funkce knihovny `locallib.php`

Lokální knihovna `locallib.php` je jádrem modulu eTask. Zajišťuje jeho korektní funkčnost a zobrazení. Soubor obsahuje 13 komplexních funkcí, které v této podkapitole zmíníme. Dostupné jsou od verze systému Moodle 2.0, pokud není stanoveno jinak. Funkce zde uvádíme bez parametrů a rozebíráme jejich podstatu. Kompletní dokumentace je k dispozici jako příloha této práce.

- `get_etask_users()` – abychom mohli s modulem eTask začít pracovat, potřebujeme seznam studentů – účastníků daného kurzu. Jejich získání realizujeme pomocí funkce `load_users()`, která je součástí knihovny `grade/report/grader/lib.php`. Studenty zapsané v kurzu lze získat také funkcí `get_role_users()` – ta je součástí knihovny `lib/accesslib.php` a využívat ji nebudeme.

- `get_formated_etask_user()` – funkce vytvoří formátovaný odkaz na uživatele, který je účastníkem kurzu. Odkaz směřuje na detail uživatelského profilu a obsahuje profilový obrázek, jméno a příjmení uživatele.

- `get_etask_assignments()` – chceme-li zobrazovat hodnocení studentů v jednotlivých úkolech, musíme tyto úkoly získat. Funkce slouží pro `SELECT` viditelných úkolů daného kurzu z databázové tabulky `mdl_assign` pomocí DML funkce `get_records_sql()`, kterou jsme zmínili v kapitole 4.1.4 na straně 28 a jsou definovány v knihovně `lib/dmllib.php`. Viditelnost úkolů v kurzu je řízena záznamy v tabulce `mdl_course_modules`. Ta nám zároveň umožní

získat `id` úkolu podle jeho `instance`. Takto získané `id` budeme následně využívat jako parametr URL identifikující úkol. Funkce je dostupná až od verze systému Moodle 2.3, kde došlo ke změně modulu pro správu úkolů. Pro starší verze by musela být funkce mírně modifikována v závislosti na užité databázové tabulce.

- `get_scale()` – modul eTask bere v potaz i uživatelsky definované škály. Uchovávají se v databázové tabulce `mdl_scale` formou řetězce – jednotlivé hodnoty jsou odděleny čárkou. Připomeňme, že škála musí být řazena vzestupně, jak jsme uvedli v kapitole 5.1.2. Abychom z řetězce obsahujícího škálu vytvořili pole indexované od `1`[26], využijeme funkci `make_menu_from_list()`, která je definována knihovnou `lib/moodlelib.php`.

- `get_gradepass_form()` – funkce vytvoří formulář pro nastavení potřebné známky u daného úkolu. Využívá třídu `gradepass_form` souboru `gradepass_form.php`, který je součástí modulu eTask. Konstruktor `moodleform` umožňuje předávat celou řadu parametrů, včetně uživatelských dat [13]. Při inicializaci nového formuláře je nutné nastavit jeho atribut `action` a dále mu předat parametry úkolu, `id` kurzu a položku, která bude přednastavena v elementu `select`.

- `get_formated_etask_assignment()` – ve výstupní eTask tabulce chceme úkoly zobrazovat jako formátované odkazy s možností nastavení potřebné známky. To nám zajišťuje právě tato funkce. Odkaz na úkol se liší dle uživatelských oprávnění a také podle toho, zda je zapnutý režim úprav kurzu. Režim úprav kurzu ověříme funkcí `user_is_editing()`, oprávnění pak pomocí `has_capability()`. Dále využíváme již zmíněné funkce `get_scale()` a `get_gradepass_form()`. Pokud není aktivní režim úprav, směřuje odkaz na detail úkolu – to platí pro roli učitele i studenta. Na titulek odkazu úkolu je aplikován tooltip widget, který jsme zmínili v kapitole 5.4.1 na straně 45. Student nemůže aktivovat režim úprav, což znamená, že odkaz bude vždy směřovat na zadání úkolu. Pokud má uživatel možnost editovat kurz a aktivuje režim úprav, odkaz bude směřovat na editaci úkolu. Zároveň se objeví

[26] Index prvního prvku škály musí mít hodnotu `1` – každý prvek škály je přepočítávána numericky. První má hodnotu `1`, druhý `2`, atd. – z toho plyne fakt vzestupného řazení škál!

editační ikona u každého názvu úkolu – po kliknutí na ni se zobrazí formulář pro nastavení potřebné známky daného úkolu. Zobrazení formuláře zajišťuje funkce `toggle()` zmíněná v kapitole 5.4.1 na straně 46.

- `get_etask_grade()` – funkce získá hodnocení studentů v jednotlivých úkolech. Realizace je možná dvěma způsoby. První z nich využívá databázovou tabulku `mdl_assign`, kde jsou záznamy o úkolech. Nesmíme opomenout propojení s tabulkou `mdl_course_modules`, abychom zajistili pouze viditelné úkoly. Následně přes tabulku `mdl_assign_grades` získáme hodnocení studentů v úkolech. Musíme však také brát v potaz hodnocení přes uživatelské škály, které se nachází v tabulce `mdl_scale`. Tato varianta řešení je však značně komplikovaná. Využili jsme druhý způsob realizace – funkci `grade_get_grades()`, která je k dispozici v knihovně `lib/gradelib.php` a vrátí objekt s hodnocením všech studentů v daném úkolu. Tyto objekty si uložíme do pole tak, abychom získali hodnocení pro všechny úkoly. Důležité je podotknout, že pro korektní funkčnost je nutné implementovat knihovny `grade/lib.php` a `grade/report/grader/lib.php`.

- `get_formated_etask_grade()` – hodnocení úkolu je v databázi uchováváno formou desetinného čísla. Tabulka modulu eTask desetinná čísla nevyužívá. Funkce změní formát z desetinného na celé číslo – aplikována je zaokrouhlovací PHP funkce `round()`.

- `is_submitted()` – modul eTask rozlišuje tři stavy, podle kterých podbarvuje buňky tabulky – jedním z nich je informace o odevzdání úkolu. Ke zjištění, zda byl úkol odevzdán, využíváme DML funkci `get_record_sql()`, která získá potřebná data z databázové tabulky `mdl_assign_submission`. Úkol je odevzdán, pokud pole `status` odpovídá hodnotě `submitted`. V závislosti na nastavení úkolu může být požadováno odevzdání potvrzené odesílacím tlačítkem – pokud je úkol odevzdán a není potvrzen tlačítkem, pole `status` nabývá hodnoty `draft`. Takový úkol však není akceptována jako odevzdaný, neboť student může provádět modifikace řešení. Důležité je upozornit, že tato funkce je opět dostupná až od verze systému Moodle 2.3 – stejně jako `get_etask_assignments()` výše.

- `get_cell_color()` – funkce určuje atribut `class` pro buňku tabulky, která je následně podbarvena užitím CSS.

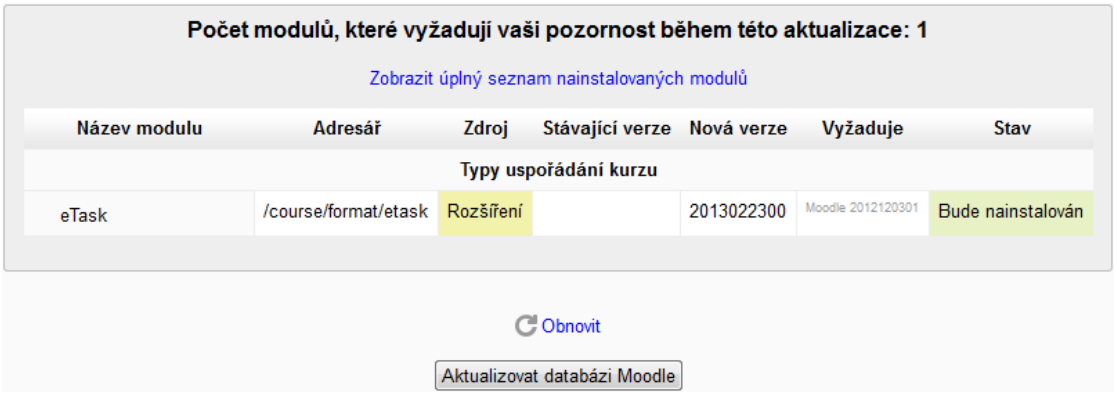
- `mod_assign_grading_sort_by()` – v kapitole 5.1.3 na straně 39 jsme se zabývali problematickým řazením studentů v hodnotící tabulce. To bylo v rozporu s řazením tabulky modulu eTask a docházelo ke kolizím při odkazování na hodnocený úkol. Funkce tento problém řeší – umožňuje nastavit, podle kterého databázového pole se má tabulka defaultně řadit. Tímto polem je `lastname`, tedy příjmení studenta.

- `get_etask_gradebook()` – renderuje tabulku modulu eTask za využití tříd `html_table`, `html_table_row` a `html_table_cell`. Ty jsou součástí knihovny `lib/outputcomponents.php` [24]. Jedná se o funkci kompletující modul za využití ostatních zmíněných funkcí knihovny `locallib.php`. Volána je v souboru `etask.php`.

- `update_gradepass()` – slouží pro aktualizaci pole `gradepass` v databázové tabulce `mdl_grade_items` pomocí DML funkce `update_record()` zmíněné v kapitole 4.1.4 na straně 29. Abychom aktualizaci mohli provést, je nutné zjistit si `id` aktualizovaného úkolu. Následně ověříme, zda byla data formulářem skutečně odeslána, zda byl potvrzen řetězec `sesskey`, který je předáván skrytým formulářovým polem a zda má uživatel k této akci oprávnění. Odpovídající funkce, uvedené ve stejném pořadí, jsou `data_submitted()`, `confirm_sesskey()` a `has_capability()`. Po ověření těchto závislostí je vytvořen aktualizační objekt a provedena samotná aktualizace hodnoty.

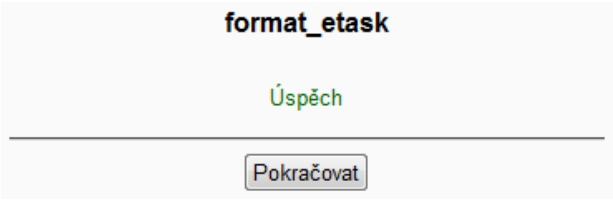
5.5 Instalace a konfigurace

Modul eTask se řídí standardním instalačním procesem systému Moodle. Modul, který je přílohou této práce, extrahujeme a následně nakopírujeme do adresáře `course/format/` v kořenovém adresáři instalace. Po přihlášení do administrace systému je nový modul automaticky detekován a jsme vyzváni k jeho instalaci. Pokud tomu tak není, přejdeme do menu Správa stránek → Informace. Modul je dostupný pro verzi systému Moodle 2.4 a vyšší.



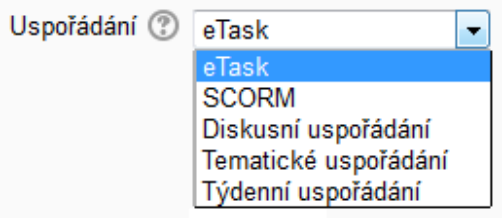
Obrázek 15: Instalace modulu eTask – kontrola zásuvných modulů

Poté, co provedeme aktualizaci databáze Moodle, jsme informováni o průběhu instalace. Pokud proběhne v pořádku, je vypsána zpráva o úspěchu. Tlačítkem `Pokračovat` se vrátíme zpět do administrace systému.



Obrázek 16: Instalace modulu eTask – upgrade na novou verzi

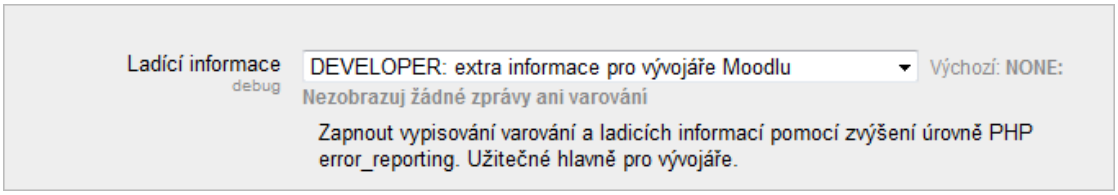
Po instalaci je modul připraven k použití. Aplikovat ho můžeme na nově zakládané kurzy, ale i na ty již existující. V nastavení kurzu stačí změnit uspořádání na hodnotu eTask.



Obrázek 17: Modul eTask – nastavení kurzu

5.6 Testování

Testování modulu probíhalo především v průběhu jeho vývoje. Abychom však mohli případné syntaktické či logické chyby pohodlně odhalit, je nutné hned v prvopočátcích nastavit režim ladění. To provedeme v administraci systému Moodle přes menu Správa stránek → Vývoj → Ladění. Položku `Ladící informace` nastavíme na hodnotu `DEVELOPER: extra informace pro vývojáře Moodle`, jak vidíme na obrázku 18 níže.



Obrázek 18: Nastavení režimu ladění

Tento režim ladění nastaví chybové výpisy[27] tak, aby byly zobrazovány veškeré chyby a varování v PHP skriptech. Dále je aktivován interní DEBUG MODE systému Moodle.

Výše zmíněné nastavení ladícího režimu je určeno především pro vývojáře! Jakmile je aplikace či modul odladěn, je důrazně doporučováno vrátit položku `Ladící informace` zpět na hodnotu `NONE: Nezobrazuj žádné zprávy ani varování`, abychom případnými chybovými výpisy neposkytli hackerům informace o nastavení našich stránek [25].

V rámci testování modulu byly využity dva nástroje zmíněné v kapitole 4.1.5 na straně 30. Jedná se o nástroj Code checker, který zajišťuje korektnost kódu z hlediska stylu kódování určeného systémem Moodle. Dále jsme aplikovali nástroj Moodle PHPdoc checker – ten provede kontrolu kódu z hlediska jeho dokumentace. Dle výsledků, které nám uvedené nástroje poskytly, modul eTask nevykazuje kódové nedostatky omezující jeho funkčnost. Kód je korektně dokumentován a je možné z něj vygenerovat plnohodnotnou dokumentaci, která je přílohou této práce.

Pro plnohodnotné ověření funkčnosti naprogramovaného modulu byl sestaven testovací scénář. Testované položky a jejich výsledky shrnuje tabulka 2 na následující straně. Testování probíhalo ve webových prohlížečích Firefox 19, Google

[27] `error_reporting` v PHP nastaví na `E_ALL` – maximální úroveň chybových výpisů.

Chrome 25, Internet Explorer 9, Opera 12 a Safari 5. Užitá verze systému Moodle 2.4. Operační systém Windows 7.

No.	Testovaná položka	Výsledek
1	Instalace/odinstalace modulu	Úspěch
2	Nastavení uspořádání kurzu na hodnotu `eTask`	Úspěch
3	Zobrazení eTask tabulky bez zapsaných účastníků kurzu	Úspěch
4	Zobrazení eTask tabulky se zapsanými účastníky kurzu	Úspěch
5	Funkčnost odkazu na uživatelský profil	Úspěch
6	Zobrazení viditelných úkolů kurzu v eTask tabulce	Úspěch
7	Funkčnost odkazu na úkol s oprávněním student	Úspěch
8	Funkčnost odkazu na úkol s oprávněním vyšším než student	Úspěch
9	Funkčnost odkazu na editaci úkolu (aktivní režim úprav)	Úspěch
10	Korektnost tooltip widgetu aplikovaného na odkaz úkolu	Úspěch
11	Zobrazení formuláře nastavujícího potřebnou známku	Úspěch
12	Funkčnost formuláře nastavujícího potřebnou známku	Úspěch
13	Odpovídající hodnocení student vs. úkol	Úspěch
14	Správné podbarvení buněk tabulky dle stavu úkolu	Úspěch
15	Korektnost odkazu na hodnocení studenta v daném úkolu	Úspěch
16	`overflow` zobrazení tabulky při nadměrném počtu úkolů	Úspěch
17	Anglická lokalizace modulu	Úspěch

Tabulka 2: Výsledky testování modulu

V průběhu testování dle výše uvedeného scénáře se objevovaly chyby v prohlížečích Internet Explorer 9 a Opera 12. Neměly však vliv na funkčnost modulu – jednalo se pouze o rozdílné zobrazování komponent modulu. Nedostatky byly odstraněny úpravou CSS. Optimalizace pro Internet Explorer vyžadovala zvláštní pozornost díky nekorektní podpoře standardů.

5.7 Přenositelnost

Modul eTask není přenositelný do starších verzí systému Moodle. Metoda, podle které bylo při realizaci postupováno, neumožňuje tuto přenositelnost zajistit. Přesto byla zvolena jako nejoptimálnější pro dané řešení, jak uvádí kapitola 5.1.

Systém Moodle jeví zásadní změny mezi jednotlivými verzemi, které se přímo dotýkají vytvářeného modulu. Ten vychází z tématického uspořádání kurzu. Pro jeho užití byla zvolena metoda nakopírování skriptů a jejich modifikace, respektive rozšíření. Již tento princip znemožňuje samotnou přenositelnost. Tématické uspořádání kurzu se liší ve verzi nižší 2.3, 2.3 a vyšší 2.4. Implementace modulu eTask ovšem zajišťuje jeho snadnou modifikovatelnost pro starší, respektive novější verze systému.

V kapitole 5.3 na straně 43 jsme uvedli, že skripty modulu eTask jsou patřičně separovány od tématického uspořádání, do kterého téměř nezasahují. To nám umožňuje velmi snadnou modifikaci pro ostatní verze systému Moodle – zásahy jsou minimální!

Nesmíme opomenout fakt, že od verze systému Moodle 2.3 došlo ke změně modulu pro správu úkolů. Původní modul `assignment`, který je v administraci veden jako položka `Úkol (2.2)`, byl nahrazen novým – `assign`. Poznámka uvádí, že je u něj vyžadován nástroj pro upgrade, tzv. `tool_assignmentupgrade`, který je standardní součástí nových verzí systému a provede aktualizaci původního modulu. Z hlediska přenositelnosti má pro nás nová verze úkolů zásadní význam a to především z důvodu odlišnosti databázových tabulek. Původní modul využíval tabulku `mdl_assignment_submissions`, nový však `mdl_assign_submission`. Tohoto problému v přenositelnosti se týkají pouze dvě funkce, které modul eTask využívá. První z nich, `get_etask_assignments()`, slouží k získání viditelných úkolů daného kurzu. Druhá, `is_submitted()`, zjišťuje dle statusu, zda byl úkol odevzdán. Řešení problému není opět náročné a modul lze velmi snadno a rychle modifikovat pro odlišné verze systému Moodle.

Dílčím cílem této diplomové práce bylo naprogramovat modul eTask a provést test jeho přenositelnosti – nikoliv však vytvořit modifikace modulu pro starší verze systému Moodle. Součástí přílohy je tedy modul, který je dostupný pouze pro verzi 2.4 a vyšší. Pro ostatní verze jsou nutné dílčí modifikace.

5.8 Uživatelské rozhraní

Uživatelské rozhraní modulu eTask se liší dle oprávnění. V zásadě se jedná o dvě skupiny uživatelů, které mají rozdílné možnosti, jak s modulem manipulovat. Jednak jde o uživatele, kteří jsou účastníky kurzu a mají roli `student`. Dále pak o skupinu uživatelů, kteří mají oprávnění aktualizovat kurz – jedná se především o role `teacher` a `manager`.

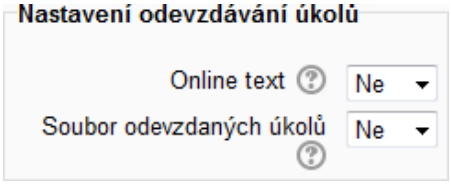
Tato kapitola se bude zabývat právě uživatelským rozhraním pro výše uvedené dvě skupiny uživatelů systému Moodle. Nejprve představíme rozhraní z pohledu učitele, respektive manažera a následně z pohledu studenta.

Rozhraní pro obě skupiny uživatelů splňuje požadavky na jednoduchost a intuitivní ovládání, jak bylo stanoveno v dílčích cílech této práce. Modul disponuje anglickou a českou lokalizací.

5.8.1 Rozhraní učitele

Základním prvkem modulu eTask je hodnotící tabulka. Hodnoceny jsou úkoly zadávané prostřednictvím modulu pro jejich správu. Využití je však mnohem bohatší než pouhý přehled hodnocených úkolů.

Podle obrázku 19 vidíme, že zadat můžeme také úkoly, které se neodevzdávají[28] – snadno tak lze definovat položku, která bude zobrazovat výsledky testů či jiných aktivit určených k hodnocení, aniž by bylo nutné odevzdávat jakýkoliv text či soubory[29].



Obrázek 19: Úkol – nastavení offline činnosti

Obrázek 20 na následující straně definuje rozhraní modulu eTask pro uživatele v rolích učitel, respektive manažer – berme v úvahu deaktivovaný režim úprav.

[28] Nastavované hodnoty nalezneme při zadávání či aktualizaci úkolu.

[29] Jedná se o obdobu úkolu `Offline činnost`, který byl dostupný ve starší verzi modulu pro správu úkolů.

	U1	t1	U2	P1	t2	T1	t3	U3	t4	U4	T2
Martin Drlík	Ano	9	Ne	-	-	87	-	Ano	-	-	-
Jakub Milásek	Ano	6	-	Ano	3	-	5	-	-	-	-
Petr Snadný	Ne	7	Ano	-	7	56	6	-	-	-	-
Denis Tsava	Ano	10	-	-	10	100	10	-	10	-	-
Anna Vzorová	-	4	-	Ano	6	18	8	-	-	-	-

Legenda: Odevzdáno Splněno Nesplněno

Obrázek 20: Modul eTask – rozhraní učitele s vypnutým režimem úprav

Záhlaví tabulky zobrazuje názvy úkolů tak, jak je definujeme při vytváření či editaci úkolu[30]. Je vhodné tyto názvy uvádět co nejstručněji, aby tabulka nenabývala do šířky a neztrácela tak na přehlednosti! Úkoly je možné vytvářet v libovolných sekcích tématického uspořádání kurzu. Modul eTask zobrazuje úkoly v pořadí, v jakém byly vytvořeny a lze je skrývat standardními nástroji systému Moodle. Jejich počet v rámci jednotlivých sekcí není omezen. Popis úkolu, který společně s názvem tvoří povinné položky, je zobrazen v jeho detailu – většinou jde o zadání či instrukce k úkolu.

Jednotlivé názvy úkolů v eTask tabulce jsou realizovány formou odkazu. Je-li vypnutý režim úprav, směřuje odkaz na detail úkolu, jinak přímo na jeho editaci. Odkazy jsou opatřeny tzv. tooltip widgetem, který je znázorněn obrázkem 13 na straně 46. Poskytuje nám informace o názvu úkolu, termínu odevzdání a potřebné známce – poslední dvě položky nemusí být nastaveny.

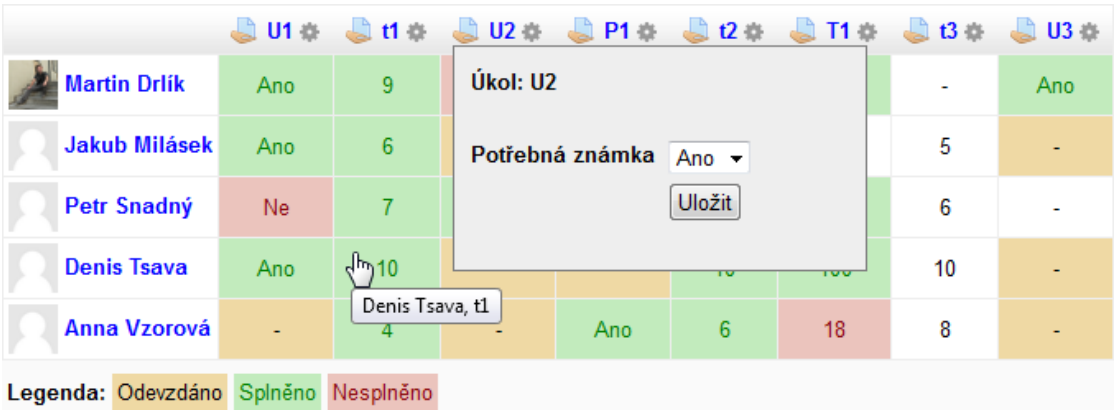
První sloupec tabulky definuje účastníky kurzu, včetně profilového obrázku. Obrázek i jméno jsou opatřeny odkazem, který směřuje na profil uživatele systému Moodle.

Hlavní oblast tabulky je hodnotící. Obsahuje hodnocení jednotlivých uživatelů v daných úkolech. Podbarvení buňky v tabulce je určeno stavem úkolu, jak jej definuje legenda. Hodnocení úkolu, které nemá nastavenou potřebnou známku, zůstává nepodbarvené, respektive s bílým podbarvením. Stejně tomu je i u neodevzdaného úkolu. Pokud je odevzdání úkolu vázáno potvrzovacím tlačítkem, musí být toto

[30]Položka `Název úkolu` v sekci `Obecná nastavení`.

potvrzení provedeno – jinak není úkol detekován jako odevzdaný. Přechod na hodnocení úkolu je realizován kliknutím na patřičnou buňku tabulky, které odpovídá úkol, respektive uživatel. Buňka je v tomto smyslu, pro lepší přehlednost, opatřena titulkem, který uvádí jméno studenta a název úkolu – viz obrázek 21 níže.

Nyní se podívejme na možnosti modulu, pokud uživatel s oprávněním učitel či manažer aktivuje režim úprav.



Obrázek 21: Modul eTask – rozhraní učitele se zapnutým režimem úprav

Jak vidíme na obrázku 21 výše, u každého odkazu s názvem úkolu přibyla editační ikona. Ta na kliknutí vyvolá formulář pro nastavení potřebné známky daného úkolu. Formulář skryjeme opětovným kliknutím na ikonu. Pokud chceme zobrazit formulář u jiného úkolu, není nutné zavírat původně otevřený – stačí aktivovat editační ikonu jiného úkolu a původně otevřený formulář je skryt. Formulář nabízí dostupné hodnocení pro daný úkol – buď číselné nebo uživatelsky definované škály. Potřebná známka nemusí být nastavena. Modul eTask však bez této mezní hodnoty nedokáže korektně podbarvit buňky tabulky a rozlišit tak stav `Splněno` a `Nesplněno` – tento případ je znázorněn na obrázku 21, kde úkol s názvem `t3` nemá nastavenou potřebnou známku. Připomeňme, že při aktivním režimu úprav směřuje odkaz názvu úkolu na jeho editaci.

Tabulka modulu eTask má nastavené tzv. přetékání. Počet hodnocených aktivit není tedy omezen šířkou stránky – pokud je překročena, pod tabulkou se zobrazí horizontální scrollbar.

5.8.2 Rozhraní studenta

Uživatelské rozhraní z pohledu studenta má spíše informativní a motivační charakter. Student vidí tabulku s komplexním hodnocením všech studentů v daných úkolech. Rozložení tabulky je totožné pro všechna oprávnění – modul se liší pouze dostupnými funkcemi.

Odkazy na jednotlivé úkoly, které tvoří záhlaví tabulky, směřují na detail úkolu – zde student odevzdává řešení. Pokud se však jedná o tzv. offline činnost, kterou jsme zmínili v kapitole 5.8.1 na straně 55, úkol se neodevzdává ani nepotvrzuje.

První sloupec tabulky, který zobrazuje účastníky kurzu, je pro všechny role totožný. Odkazy na uživatelské profily jsou, pro registrované a přihlášené uživatele systému Moodle, veřejně dostupné – není tedy nutné zavádět speciální oprávnění.

Hodnotící oblast neposkytuje studentům žádnou interaktivitu. Pro lepší přehlednost je opět každá buňka tabulky opatřena titulkem se jménem studenta a názvem úkolu. Ze strany studentů může docházet k nekorektnímu odevzdávání úkolů. Pokud učitel u úkolu nastaví položku `Požadovat, aby studenti klikli na tlačítko Odeslat` na hodnotu `Ano`, musí student tlačítkem potvrdit odevzdání – dává tím najevo, že odevzdaná verze řešení je finální. Pokud při daném nastavení uživatel nepotvrdí odevzdání, úkol má status konceptu a v eTask tabulce není veden jako odevzdaný. Úkoly, u kterých není vyžadováno potvrzené odevzdání, mají automaticky status odevzdaných.

6 Závěr

Diplomová práce se zabývá vývojem nových modulů pro LMS Moodle. Členěna je do dvou částí. Teoretická část, která je tvořena obecnou metodikou vývoje modulů a praktická, která ověřuje obecnou metodiku – výstupem je modul eTask.

Teoretický výzkum možností programovacího frameworku platformy Moodle 2.0, který byl proveden, vedl ke zpracování obecné metodiky tvorby nových modulů. Důležité je zmínit, že se jedná o obecnou metodiku. Vzhledem k množství typů modulů, které systém Moodle umožňuje implementovat (viz tabulka 1 na straně 15 zobrazující přehled těch nejzákladnějších), není možné tuto problematiku uchopit a zpracovat komplexně. Metodika vychází ze šablony pro vývoj nového modulu – ta je obecná a ve svém adresáři nabízí soubory, které jsou součástí většiny typů modulů. Každý soubor má svá specifika a je definováno, k čemu slouží. Popsány jsou principy a funkce podstatné pro korektní funkčnost modulu. Součástí práce jsou také praktické příklady – jmenujme např. lokalizaci řetězců, vytváření formulářových prvků, manipulaci s databázovými záznamy či práci s javascriptovou knihovnou YUI. Důraz je kladen na správnost kódu, včetně dokumentace, a způsob jeho ladění. Znázorněn je postup instalace a poukázáno na nutnost testování. V závěru teoretické části uvádíme základní požadavky na modul eTask, který bude prakticky ověřovat vzniklou metodiku.

Praktická část slouží pro experimentální ověření obecné metodiky, která je navržena v první části této práce. Ověření bylo zajištěno naprogramováním modulu eTask a poukázalo na fakt, že tuto metodiku nelze využít ani chápat jako manuál a v tomto slova smyslu k vývoji modulu nepostačí. Obecnost metodiky vnáší do vývoje modulů didaktické hledisko. Aby byl uživatel schopen vyvinout a naprogramovat vlastní modul, se zcela specifickými požadavky, musí si sám dohledávat a nastudovat dodatečné materiály a postupy – tím si je osvojuje. Navržená metodika, navzdory značné individualizaci při vytváření jednotlivých modulů, poskytuje dostatečný základ pro jejich vývoj. Východiska realizace modulu eTask poskytla provedená detailní analýza řešení. Modul je založen na existujícím datovém modelu a nevytváří žádné dodatečné databázové tabulky. Do tématického uspořádání kurzu, z kterého modul vychází, je implementována hodnotící tabulka. Popsána je adresářová struktura a aplikační logika včetně užitých funkcí. Opomenuta není

instalace modulu, jeho testování a přenositelnost. Funkční nedostatky byly odstraněny při dílčích testováních. Finální testování proběhlo bez funkčních chyb, avšak objeveny byly nedostatky v zobrazení modulu různými prohlížeči. Zásadní problém způsoboval prohlížeč Internet Explorer 9, kdy se po umístění kurzoru myši na odkaz pohyboval obsah stránky pod tabulkou směrem dolů – mezi tabulkou a zbytkem stránky tak vznikala zvětšující se mezera. Zobrazovací nedostatky byly odstraněny. Závěrečné kapitoly se věnují popisu uživatelského rozhraní modulu eTask – z pohledu učitele a studenta.

Za dílčí neúspěch při vývoji modulu lze považovat jeho nedostupnost pro starší verze systému Moodle. Modul vychází z principů, které možnost přenositelnosti vylučují. Je však realizován způsobem, který umožňuje jeho snadnou modifikaci pro starší verze. Modul je dostupný pro verzi Moodle 2.4 a vyšší. Dále se nepodařilo zajistit pohodlné hodnocení úkolů prostřednictvím dialogového okna, jak bylo původně zamýšleno. Modul pro správu úkolů a jejich hodnocení je v současné době řešen způsobem, který neumožňuje jeho snadnou implementaci. Problémy nastaly i při vytváření dialogového okna javascriptovou knihovnou YUI, kterou systém Moodle defaultně využívá. Z hlediska efektivity bylo hodnocení úkolů přes dialogové okno zamítnuto. Současné řešení využívá přímého přechodu na hodnotící stránku. Zde však připomeňme problém, který jsme již vyřešili, s rozdílným řazením záznamů v hodnotící tabulce a tabulce modulu eTask. Detailně jsme se problematikou zabývali v kapitole 5.1.3 na straně 37.

Přes dílčí problémy, které vývoj modulů pro LMS Moodle provázely, se podařilo splnit cíle uvedené v úvodu této diplomové práce. Na základě teoretického výzkumu byla vytvořena metodika tvorby nových modulů a ta následně ověřena naprogramováním konkrétního modulu. Modul byl otestován a zdokumentován.

Použitá literatura a zdroje

[1] RAADT, Michael de. *Moodle 1.9 top extensions cookbook*. Birmingham: Packt Pub, 2010. ISBN 18-495-1216-7.

[2] MoodleDocs: About Moodle. *Moodle* [online]. 21 November 2012 [cit. 2013-02-02]. Dostupné z: http://docs.moodle.org/24/en/About_Moodle

[3] GUTIÉRRE, Eladio, María A. TRENAS, Julián RAMOS, Francisco CORBERA a Sergio ROMERO. A new Moodle module supporting automatic verification of VHDL-based assignments. *Computers* [online]. 2010, roč. 54, č. 2, s. 562-577 [cit. 2013-02-02]. ISSN 03601315. DOI: 10.1016/j.compedu.2009.09.006. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0360131509002462

[4] RICE, William H. *Moodle 2.0 e-learning course development: a complete guide to successful learning using Moodle*. 1st pub. Birmingham: Packt, 2011, viii, 321 s. Community experience distilled (Packt). ISBN 978-1-84951-526-9.

[5] MoodleDocs: Plugins. *Moodle.org* [online]. 4 December 2012 [cit. 2013-02-04]. Dostupné z: http://docs.moodle.org/dev/Plugins

[6] MoodleDocs: NEWMODULE Documentation. *Moodle.org* [online]. 6 December 2011 [cit. 2013-02-05]. Dostupné z: http://docs.moodle.org/dev/NEWMODULE_Documentation

[7] MOORE, Jonathan a Michael CHURCHWARD. *Moodle 1.9 extension development: customize and extend Moodle by using its robust plugin systems*. Birmingham, U.K.: Packt Open Source, 2010, xi, 298 p. ISBN 978-1-847194-24-4.

[8] MoodleDocs: Upgrade API. *Moodle.org* [online]. 7 July 2012 [cit. 2013-02-05]. Dostupné z: docs.moodle.org/dev/Upgrade_API

[9] MoodleDocs: XMLDB editor. *Moodle.org* [online]. 22 June 2011 [cit. 2013-02-05]. Dostupné z: http://docs.moodle.org/dev/XMLDB_editor

[10] MoodleDocs: Logging API. *Moodle.org* [online]. 25 January 2012 [cit. 2013-02-05]. Dostupné z: http://docs.moodle.org/dev/Logging_API#Mod.2F.2A.2Fdb.2Flog.php_Files

[11] Data definition API: MoodleDocs. *Moodle.org* [online]. 25 January 2012 [cit. 2013-02-07]. Dostupné z: http://docs.moodle.org/dev/DDL_functions

[12] MoodleDocs: String API. *Moodle.org* [online]. 13 March 2012 [cit. 2013-02-16]. Dostupné z: http://docs.moodle.org/dev/String_API

[13] MoodleDocs: lib/formslib.php Form Definition. *Moodle.org* [online]. 9 February 2013 [cit. 2013-02-16]. Dostupné z: http://docs.moodle.org/dev/lib/formslib.php_Form_Definition

[14] MoodleDocs: Data manipulation API. *Moodle.org* [online]. 7 January 2013 [cit. 2013-02-09]. Dostupné z: http://docs.moodle.org/dev/Data_manipulation_API

[15] Docs For Class moodle_database. *Moodle.org* [online]. 15 Nov 2011 [cit. 2013-02-12]. Dostupné z: http://phpdocs.moodle.org/HEAD/core/dml/moodle_database.html

[16] MoodleDocs: Coding style. *Moodle.org* [online]. 13 December 2012 [cit. 2013-02-20]. Dostupné z: http://docs.moodle.org/dev/Coding_style

[17] MoodleDocs: Course formats. *Moodle.org* [online]. 5 February 2013 [cit. 2013-02-20]. Dostupné z: http://docs.moodle.org/dev/Course_Format

[18] BARRINGTON, Rebecca. *Moodle gradebook: set up and customize the gradebook to track student progress through Moodle.* 1st ed. Birmingham: Packt., 2012, iii, 113 s. Community experience distilled. ISBN 978-1-84951-814-7.

[19] MoodleDocs: Gradebook. *Moodle.org* [online]. 22 October 2012 [cit. 2013-02-26]. Dostupné z: http://docs.moodle.org/24/en/Gradebook

[20] MoodleDocs: Grades. *Moodle.org* [online]. 2 November 2011 [cit. 2013-02-27]. Dostupné z: http://docs.moodle.org/dev/Grades

[21] HOLE, Alastair. *Moodle JavaScript cookbook: over 50 recipes for making your Moodle system more dynamic and responsive with JavaScript*. Birmingham: Packt, 2011, iii, 165 s. Qick answers to common problems (Packt). ISBN 978-1-84951-190-2.

[22] YUI Library: Example: Creating a Simple Tooltip Widget With Extensions. *YUI Library* [online]. © 2006-2013 [cit. 2013-03-31]. Dostupné z: http://yuilibrary.com/yui/docs/widget/widget-tooltip.html

[23] YUI Library: NodeList Class. *YUI Library* [online]. © 2006-2013 [cit. 2013-03-30]. Dostupné z: http://yuilibrary.com/yui/docs/api/classes/NodeList.html

[24] MoodleDocs: Output API. *Moodle.org* [online]. 1 October 2012 [cit. 2013-04-01]. Dostupné z: http://docs.moodle.org/dev/Output_API

[25] MoodleDocs: Debugging. *Moodle.org* [online]. 20 July 2012 [cit. 2013-04-01]. Dostupné z: http://docs.moodle.org/23/en/Debugging

Seznam obrázků

Seznam tabulek

Seznam příkladů

Přílohy

1. DVD – na přiloženém DVD se nachází plné znění diplomové práce pod názvem souboru `drlik_diplomova-prace.pdf`, dále jsou v komprimované složce `etask.zip` přiloženy zdrojové kódy naprogramovaného modulu. Součástí je také adresář `dokumentace`, který obsahuje dokumentaci funkcí modulu eTask.