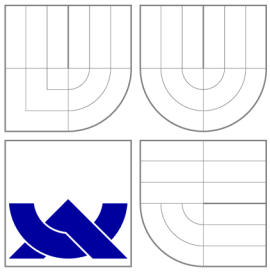
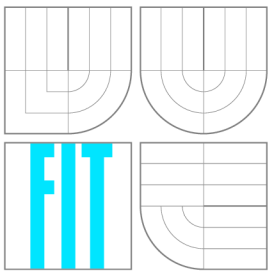


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

E-LEARNINGOVÝ KURS

E-LEARNING COURSE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. IVETA ŠENFELDOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ŠÁRKA KVĚTOŇOVÁ, Ph.D.

BRNO 2010

Abstrakt

Informační technologie jsou v dnešní době každodenní podstatnou součástí studentského života. Současné e-learningové technologie jsou založeny na klasickém vztahu učitel-student. Tato práce představuje nový směr, jak učinit učební proces efektivnější pomocí zvýšení interakce mezi studenty.

Abstract

Information technology is nowadays a essential part of student's everydays life. Current e-learning technologies are based on the basic teacher-student relationship. This work presents a new way how to make the learning process more effective by bringing the student's interaction to the next level.

Klíčová slova

e-learning, kurs, informační systém, Moodle, eFront, student, učitel

Keywords

e-learning, course, information system, Moodle, eFront, student, teacher

Citace

Iveta Šenfeldová: E-learningový kurs, diplomová práce, Brno, FIT VUT v Brně, 2010

E-learningový kurs

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením paní Šárky Květoňové

.....
Iveta Šenfeldová
25. května 2010

Poděkování

Ráda bych poděkovala p. Ing. Šárce Květoňové, Ph.D. za vedení mé diplomové práce.

Obsah

Kapitola 1

Úvod

Stále roste potřeba procesy co nejvíce automatizovat. Forma automatizace může představovat hardwarové nebo softwarové řešení v různých oblastech. Tato potřeba vznikla také v oblasti vzdělávání a dala tak počátek vzniku softwarové podpory výuky - e-learningu.

Již existující e-learningové systémy se od sebe příliš neliší a zejména pracují s klasickým dvouúrovňovým (tříúrovňovým, bere-li se v potaz administrátor) uživatelským modelem, kde učitel spravuje veškeré studenty. Tato práce si však klade za cíl představit nový model, který má především motivovat studenty a jejich spolužáky k lepším výsledkům pomocí podpory vzájemné interakce.

Ve druhé kapitole je obecně rozebrán e-learning, zejména výhody a nevýhody a technologie e-learningu. Třetí kapitola se zabývá požadavky kladenými na online webové aplikace, ale také specifickými požadavky na e-learningový systém. V prvním odstavci je zmíněn nový model a tento je popsán v kapitole čtvrté. V páté kapitole jsou rozebrány a zhodnoceny již existující řešení e-learningových systémů, jak komerční, tak nekomerční. Návrh aplikace a použité nástroje popisuje kapitola šestá a následná implementace, která z návrhu vyplývá, je rozebrána v kapitole sedmé. Následuje kapitola osmá, ve které jsou nastíněny potenciální možnosti rozšíření systému. Předposlední kapitola pojednává o testování a ladění aplikace, které je nedílnou součástí vývoje softwaru. Poslední kapitola je závěrečné zhodnocení. V příloze je ukázka testových otázek, detailní návrh databáze a use-case diagram společně se specifikací.

Kapitola 2

E-learning

Co to e-learning je? Neexistuje jednotná definice e-learningu, avšak názory expertů jsou například tyto:

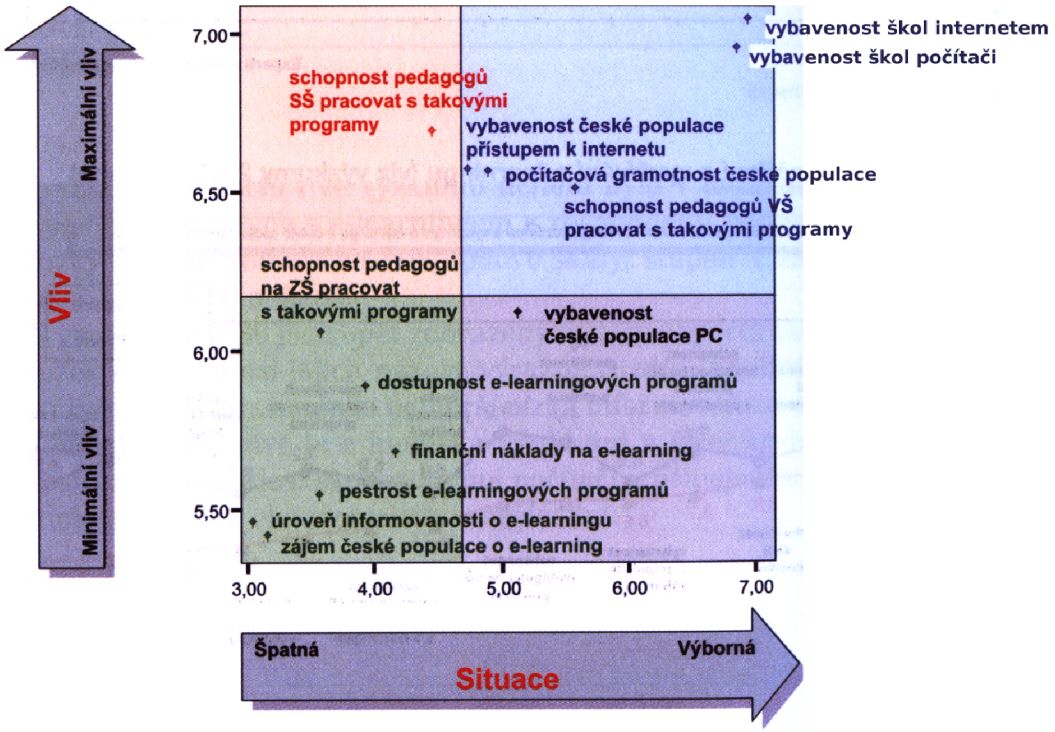
- „Forma vzdělávání využívající elektronická média a multimediální technologii (bez přímé účasti učitele se synchronní a asynchronní formou komunikace). Podstatu tvoří multimediální studijní materiály, zpětnovazební činnosti a funkce pro správu e-learningu.“[16]
- „Výuka, která využívá možností IT. Jedná se především o komunikaci, sdílení dat, organizaci studijního času, vyhodnocování a testování.“[16]
- „Elektronické výukové a vzdělávací systémy (zejména v on-line podobě).“ [16]

E-learning se postupem času mnohem více začleňuje do klasické vyučující metodiky. Může sloužit obecně nebo pro konkrétní případy, často je využíván ve firmách pro školení zaměstnanců, školami pro podporu výuky některých předmětů či výuka studentů na kombinovaném studiu nebo je lidé užívají pro svou osobní potřebu. Situaci v oblasti e-learningu znázorňuje obrázek 2.1.

2.1 Výhody e-learningu

K nejzásadnějším výhodám e-learningu patří tyto:

- čas - jde jednak o úsporu času, co se týče například cestování, ale také co se týče rozvrhu. Student se vzdělává tehdy, když mu to umožňuje jeho denní rozvrh. Další úspora času oproti knize je rychlejší vyhledávání informací.
- dostupnost - e-learning je stále „po ruce“. Tato výhoda do jisté míry souvisí s časovou výhodou, student si může spustit výukový program téměř kdykoli a kdekoli.
- individuální přístup - student se vzdělává dle svých dispozicí, upravuje si zejména učební tempo a lze kdykoli přestat.
- posílení soustředění - lze studovat z pohodlí domova a student tak není rušen ostatními studenty



Obrázek 2.1: Situace v oblasti e-learningu a vliv na e-learning [16]

2.2 Nevýhody e-learningu

K nejzásadnějším nevýhodám e-learningu patří tyto:

- ztráta osobního kontaktu - styk s lidmi má nepochybně lepší vliv na lidskou psychiku než když je člověk sám. V tomto případě je student ochuzen o zážitky, které se mohou odehrát pouze ve společnosti. Nový přístup popsaný v této práci by měl tento nedostatek zmírnit, protože nabádá skupinky studentů k úzké vzájemné interakci. Tyto skupiny mohou být tvořeny například na základě geografické lokality, takže systém napomáhá vzájemným setkáním.

- nedostatek kontroly - zpětná vazba zde nechybí, avšak není dostačující. Je vhodnější, když si je student okamžitě uvědomněn o svých chybách. Někteří studenti mohou také postrádat dozor, který posiluje jejich vůli. Také tento nedostatek by měl do jisté míry odstranit nový přístup popsaný v této práci.

- nelze prokázat totožnost studenta, pokud není přítomen v učebně s učitelem. Proto nelze e-learning použít na konání například závěrečných zkoušek z domu.

- nedostupnost - pro některé uživatele může být e-learning zcela nedostupný kvůli absenci technologického vybavení.

2.3 Technologie e-learningu

Níže uvedené pojmy definují přístup a práci s e-learningem.

2.3.1 Computer-Based Training (CBT)

Lze konstatovat, že Computer-Based Training je v podstatě učení se za pomocí počítače. Tím je myšleno, že počítač je v roli podpůrného prostředku v procesu učení. Příkladem je výukový kurz nebo materiály k výuce na cd-rom disku. Výuka tohoto typu dnes není tak populární, používá se WBT viz kapitola níže.

2.3.2 Web-Based Training (WBT)

Web-Based Training je jako Computer-Based Training s tím rozdílem, že u WBT je e-learning dostupný či spouštěn a vykonáván přes webový prohlížeč. V minulosti bylo použito právě spíše cd-romů, neboť připojení k internetu nebylo tolik rozšířeno, dnes už to ale problém není. WBT nemusí nutně znamenat e-learningový systém nějaké společnosti, ale obecně se dotýká toho faktu, že na internetu lze najít a zjistit téměř cokoli.

2.3.3 Learning Management System (LMS)

Learning Management System je ve skutečnosti řídicím systémem, je to strategické řešení pro plánování, dodávání a řízení všech vzdělávacích aktivit probíhajících v organizaci (zahrnuje on-line vzdělávání, virtuální třídy i školení vedená instruktorem) [1]. Nejedná se pouze o řízení kurzů, ale také správu uživatelů a správu studijních výsledků uživatelů daného systému. LMS dále také zpřístupňuje studentům učební obsah. Za běžné funkce systémů řízeného vzdělávání můžeme považovat následující moduly:

- testování a přezkušování žáků
- komunikační nástroje
- katalog výukových kurzů a objektů
- úložiště výukového obsahu [20]

LMS se nestará o obsah kurzů [1].

2.3.4 Learning Content Management System (LCMS)

Learning Content Management System má na starosti právě obsah kurzů a dále poskytuje autorům a návrhářům prostředky pro efektivní tvorbu obsahu. [1] Mezi běžné funkce LCMS patří:

- podpora výukových strategií e-learning
- týmový proces tvorby a úprav obsahu
- správa a znovu používání obsahu, sdílení, verzování, zamykání obsahu a zdrojů
- podpora vkládání řady typů multimedií (obrázky, animace, videa, zvuky, simulace) známých formátů, měnění jejich vlastností a programování jejich interakcí s okolím [19]

2.4 Dělení e-learningu

E-learning lze dělit z mnoha pohledů, dva základní jsou popsány níže.

2.4.1 Z pohledu připojení k internetu

- **On-line e-learning** - jde o takový systém, který vyžaduje neustálé připojení k internetu pro svou plnohodnotnou funkčnost.
- **Off-line e-learning** - takový systém nevyžaduje stálé připojení k internetu nebo jej nevyžaduje vůbec, například e-learningové kurzy na cd-rom.

2.4.2 Z pohledu komunikace mezi účastníky

- **Asynchronní e-learning** - v asynchronním e-learningu není nutné permanentní připojení k počítačové síti a uživatelé nekomunikují v reálném čase. Příkladem asynchronní komunikace je kupříkladu diskuzní fórum nebo komunikace formou e-mailových zpráv.
- **Synchronní e-learning** - v synchronním e-learningu je nutné permanentní připojení k počítačové síti a komunikace mezi uživateli probíhá v reálném čase. Takto funguje například chat, video nebo audiokonference a další.

2.4.3 Shrnutí

Existuje mnoho různých řešení e-learningu pro různé situace, použití a cílové skupiny. Tato práce je zaměřena na online LMS s podporou jak synchronní tak asynchronní komunikace mezi účastníky.

Kapitola 3

Požadavky na e-learningový systém

Při počáteční analýze každého budoucího projektu (nejen informačního systému) je nutno rozebrat a stanovit požadavky na daný systém. Požadavky lze pro lepší názornost rozdělit do tří kategorií:

- obecné požadavky
- specifické požadavky
- rozšiřující požadavky jako doplňující kategorie

Všechny požadavky uvedené v následujích kapitolách se týkají především on-line webové aplikace - e-learningového systému, avšak obecné požadavky se mohou svou nespecifičností týkat i jiných systémů.

3.1 Obecné požadavky

Za obecné požadavky na systém lze považovat tyto:

- použitelnost
- spolehlivost
- bezpečnost
- multijazyčnost
- softwarové a hardwarové zázemí
- možnost rozšiřování systému
- komunikace mezi uživately

Některé z výše uvedených požadavků mohou být považovány za samozřejmost, a i když se s nimi při návrhu systému často počítá, ne vždy se povede je splnit. Takovými mohou být kupříkladu přehlednost, jednoduchá ovladatelnost, bezpečnost a další. Proč tomu tak je, má nepochybně více důvodů. Těmi základními mohou být nedostatečná znalost oblasti (nebo problému) při návrhu či nepostačující softwarové nebo hardwarové vybavení. Následující kapitoly blíže popisují zmíňované požadavky.

3.1.1 Použitelnost

Aby se uživatel v systému dokázal snadno orientovat, měl by sytém splňovat základní principy použitelnosti. Použitelnost je relativní pojem a není možné jej exaktně vystihnout, nicmémě jsou určité konvence, na které jsou uživatelé zvyklí a těmi jsou například menu, které bývá často umísťováno v levé nebo v horní části systému nebo při přihlašování do systému je požadováno jako první uživatelské jméno a poté heslo, ne naopak. Uživatel by proto neměl být maten porušováním těchto konvencí, neboť by to mělo pouze negativní vliv, pokud by mělo jít o seriózní informační systém či jiný podobný produkt.

S použitelností nepochybně souvisí i design celého programu, který by měl být střídmý, nemělo by být použito mnoho barev a také by se neměly objevovat žádné blikající pozadí, které odvrací pozornost a spíše působí jako rušící element.

K použitelnosti se váže jednoduchá ovladatelnost. Cílem je uživateli usnadnit pohyb v systému. K tomu slouží například menu, mapa stránek, popisky různých funkcí nebo přímo nápověda, ve které je vše popsáno a vysvětleno. U složitějších programů se také používají tzv. tutoriály, které uživatele postupně provází systémem a předvádí jeho možnosti a způsoby ovládání. S použitelností ještě dále souvisí přístupnost. Ta vyjadřuje míru, s jakou dokážou s webovou prezentací pracovat různě omezení uživatelé. Přístupný web se dá ovládat třeba také bez myši, bez obrázků, na černobílém monitoru nebo z mobilu.[3]

3.1.2 Spolehlivost

Ke spolehlivému systému vede nepochybně jeho správný počáteční návrh, implementace, důkladné testování a dostatečně bezpečné provozní podmínky. Pokud vznikne v návrhu, implementaci nebo při provozu v systému zranitelné místo, může se stát, že ve spojení s hrozbou, kterou představuje například útočník z venku, bude takové zranitelné místo využito v jeho prospěch. Dále je důležité, aby server, na kterém je webová aplikace nasazena, splňoval požadavky, které podpoří spolehlivost celé aplikace. Spolehlivost lze do jisté míry definovat například tak, že je služba dostupná nad určitou procentuální hranici.

3.1.3 Bezpečnost

V dnešní době je bezpečnost, zvláště webových aplikací, velmi důležitá. U e-learningového systému jde především o ochranu osobních údajů a důvěryhodnosti systému pro zahrnutí do studijních výsledků. Jsou určitá opatření, která musí být součástí systému, a bez kterých by nebylo možno systém provozovat. S bezpečností souvisí tyto pojmy:

- spolehlivost
- zálohování
- autorizace a autentizace
- auditování

Bezpečnost, tak jako u spolehlivosti, je jak hardwarová, tak softwarová. Hardwarová bezpečnost je pro tuto práci méně podstatná, za zmínku ale stojí zálohování, které pro svou funkčnost potřebuje hardwarové vybavení v podobě diskových polí apod. Softwarově musí být zálohování samozřejmě také podporováno. Díky bezpečnosti je systém spolehlivý, pokud je vše správně ošetřeno. Hlavním aspektem bezpečnosti a také tím, s čím se běžný uživatel internetu denně setkává, je autentizace a autorizace. Pomocí autentizace se do systému

uživatel přihlásí a dle autorizace má buďto nějaká omezení, co smí a nesmí vykonávat nebo je neomezen (například administrátor). Obecně platí, že čím je systém složitější, tím větší je šance na nějakou „díru“ v systému, které může zneužít potenciální útočník. Pro větší přehled a snažší pochopení dění v systému se často používá tzv. logování. Obvykle jde o soubor (nebo více souborů), do kterých se zapisují systémové údaje. Takovými údaji mohou být přihlašování uživatelů, chybové stavy a další. Záznam musí obsahovat datum a čas a stručný popis události. Auditování neboli logování obvykle provádí systém automaticky, nicméně není problémem, aby uživatel (administrátor) mohl do jisté míry zadávat systémová hlášení ručně.

3.1.4 Multijazyčnost

Moderní české informační systémy dnes již disponují multijazyčností, tedy možností přepnutí systému do cizího jazyka. Pro tento konkrétní případ, kdy se jedná o e-learningový systém, který by měl vzdělávat zahraniční studenty, je multijazyčnost jednou ze základních podmínek. S multijazyčností se dnes počítá již ve všech hlavních frameworcích vyvinutých pro tvorbu webových aplikací, díky čemuž je vývoj snadnější. Multijazyčnost se dá rozdělit na dvě části. Jednak jde o národní nastavení, kdy lze přepínat měny, metriky a podobně a jednak jde o jazyk frontendu aplikace.

3.1.5 Softwarové a hardwarové zázemí

Tento požadavek je ze všech uvedených nejzákladnější. Je důležité mít zázemí pro vývoj, ale také pro provoz systému. Podstatné je na začátku vývoje si stanovit cíle a očekávání a podle toho zvolit softwarové a hardwarové zázemí tak, aby se byl schopen systém vypořádat s různými nepříznivými scénáři, které mohou nastat. Tento požadavek úzce souvisí se spolehlivostí a bezpečností. Správný výběr zázemí může v budoucnosti ušetřit spoustu problémů. Zde se také setkáváme s požadavkem omezení přírodních hrozeb, jakými jsou například oheň, voda atd. Proto jsou místnosti se servery, na kterých běží systém, speciální v tom smyslu, že se snaží minimalizovat hrozby tohoto typu a to kupříkladu tím, že sníží teplotu v místnosti, že je místnost ve vyšším patře a jiné.

3.1.6 Možnost rozšiřování systému

Systém by měl být do určité úrovně v budoucnu rozšiřitelný. Může vzniknout potřeba práce s novou technologií nebo se může stát systém nedostačujícím v určité oblasti.

K možnosti rozšiřitelnosti nepochybně přispěje dobrý návrh systému a používání standardních řešení a rozhraní. Toto do velké míry zajistí u webových aplikací framework daného skriptovacího jazyka, jež dopředu počítá s podobnými fakty.

Některé služby webové aplikace mohou být implementovány v odlišném programovacím jazyce, díky čemuž je služba potom kupříkladu rychlejší nebo výkonnější. Z tohoto pohledu lze také uvažovat o dalším možném rozšiřování, protože různé, v dnešní době více používané, programovací jazyky mají obvykle velké množství knihoven a přibývají nové, které podporují právě nově vytvořené technologie a postupy.

Podporu pro rozšiřování systémů lze zajistit pomocí webových služeb, kdy bude od aplikace oddělen aplikační server, který bude zpracovávat příchozí požadavky, které může přijímat od více různých aplikací.

3.1.7 Komunikace mezi uživateli

Je žádoucí, aby mohli mezi sebou uživatelé systému komunikovat. Komunikace může být dvojí. Online komunikace - například chat nebo případná audio či videokonference.

Další možností je komunikace uloženými zprávami. Ta se může odehrávat pomocí diskuzního fóra nebo pomocí zasílání správ v rámci systému.

Poslední formou komunikace je sjednávání schůzek a vzájemná organizace času. Systém by měl umožňovat jistý timemanagement skupin formou pozvánek s navrhovanými termíny.

Komunikace je, zvlášť v učebním procesu, velmi důležitá. Často napomůže k pochopení látky za mnohem kratší čas než kdyby byl student odkázán sám na sebe.

3.2 Specifické požadavky

Specifické požadavky na systém mohou být tyto:

- databáze učebních textů
- statistiky a přehled studia
- zpětná vazba studentů
- podpora pro vkládání multimediálních souborů

Pokud se systém skládá z modulů (tak jako například Moodle), je možné v budoucnu požadovanou funkcionalitu doimplementovat a zaintegrovat do systému. V této podkapitole jsou uvedeny specifické požadavky na řešený e-learningový systém. Ač jsou specifické, s některými z nich se lze setkat v mnoha jiných systémech, nejen e-learningových. Kupříkladu statistiky a zpětná vazba uživatelů jsou velmi přínosné funkce pro chod víceuživatelského systému.

3.2.1 Databáze učebních textů a otázek

Součástí e-learningového systému musí být databáze učebních textů a zkušebních otázek. Studenti musí mít přístup k takové databázi, aby mohli z daných textů čerpat, a tak se připravovat na testy jednotlivých kurzů. Texty, a jiné učební pomůcky, by měly být uloženy ve standardních formátech, obecně je to však jedno.

Zkušební otázky by měly být vhodně ohodnoceny body tak, aby byl student po nastudování potřebných materiálů schopný správně odpovědět. Je vhodné, aby otázek bylo větší množství, aby byl, pokud možno, každý test unikátní. Unikátnost lze do jisté míry zajistit tak, že se pro každý test k určitému předmětu vyberou otázky z databáze náhodně a také možnosti odpovědi by měly být různě přehazovány. V takovém případě se do určité míry zamezí nechtěnému efektu, kdy si studenti zaznamenají testové otázky a pak se pouze naučí správné odpovědi.

Dalším problémem u testových otázek bývá to, že studenti náhodně vybírají své odpovědi, když neznají odpovědi. Demotivací k tomuto je bezpochyb taková bodová srážka za špatnou odpověď, že se studentům vynaložené úsilí nevyplatí.

3.2.2 Statistiky a přehled studia

Je žádoucí mít přehled o úspěšnosti jednotlivých testů, celých kurzů a podobně. K tomu účelu slouží statistiky a různé přehledy, díky kterým lze zjistit například obtížnost kurzu. V případě, že by byly statistiky podrobné, je možno vyhodnocovat, na jaké otázce studenti často chybují nebo v jaké oblasti a tím zjistit, kterou oblast předmětu je potřeba posílit. Je samozřejmě možné uvažovat i o sběru jiných dat. Takovými může být počet stáhnutí různých textů a jiné.

3.2.3 Zpětná vazba studentů

K možnému vylepšení systému může pomoci právě zpětná vazba studentů, protože právě studenti jsou hlavními uživateli e-learningu a jejich připomínky jsou v tomto smyslu cenné. Ke každému kurzu je vhodné přiřadit formulář s obecnými otázkami, avšak neméně pomůže také textové pole, kde lze napsat poznámku, konstruktivní kritiku nebo jiný komentář. Odeslání formuláře se následně započítá do statistik, které jsou zmíňovány výše. Zpětnou vazbu lze provádět i neutomatizovaně a to tak, že student může komunikovat s učitelem pomocí diskuzního fóra, e-mailu či osobně.

3.2.4 Podpora pro vkládání multimediálních souborů

Multimediální soubory, jako jsou zejména audio, video a obrázky, mohou výrazným způsobem napomoci k pochopení látky. Proto je vhodné, aby systém umožňoval podporu pro vkládání takových souborů a jejich přehrávání či zobrazení, například přímo v testu nebo v elektronickém učebním textu.

3.3 Možné rozšiřovací požadavky

Každý systém je možné nějakým směrem vylepšit nebo rozšířit. Záleží na tom, jaký přijde požadavek a zda je důležité jej splnit či nikoliv. Pokud je požadavek splnitelný a povede ke zkvalitnění systému, pak je samozřejmě vhodné jej implementovat. Rozšiřovací požadavky se různí dle systému, o který se jedná. V případě informačního systému pro e-learningové kurzy mohou být požadavky například tyto:

- podpora offline studia
- propojení s jiným systémem

3.3.1 Podpora offline studia

Může nastat situace, že student nebude disponovat stálým připojením k internetu (například z důvodu výpadků sítě). V takovém případě bude pro studenta velkou výhodou, pokud bude schopen studovat offline. Takovou podporu však není lehké vybudovat. Nabízí se tyto možnosti: dostupnost systému nebo pouze materiálů na přenosném médiu nebo schopnost systému pracovat offline po zjištění, že je síť nedostupná. Obojí má své výhody a zejména nevýhody.

V prvním případě - přenosné médium - je jednoznačná výhoda ta, že lze systém spustit na kterémkoliv počítači a je stále poruce, nebo formou vytištěných materiálů. Problémem, a tedy nevýhodou, je, že informace bude časem nutné aktualizovat. Ideální formou offline

systému by byl systém, který by se synchronizoval s úložným médiem a v případě, kdy by přešel systém do online módu, provedl by synchronizace s centrálním serverem.

Druhou možností je, že bude systém schopný rozpoznat, že není dostupná síť a s takovou záležitostí se vypořádá. Výhodou systému je transparentnost pro studenta,vše probíhá automaticky. To platí i pro aktualizaci systému, která, pokud bude k dispozici, proběhne po opětovném připojení do sítě nebo kdykoliv v průběhu práce se systémem. Nevýhodou je, že pro offline práci musí být k dispozici na soukromém disku studenta systém s databází. Databáze by měla obsahovat pouze takové informace, které jsou nutné pro práci se systémem. V tom případě je na disku vyžadováno místo a pokud není explicitně podpora pro přenos systému na jiný počítač, pak je dostupný s daty studenta jen na jeho počítači. Tento princip podporuje například služba Google Gears.

3.3.2 Propojení s jiným systémem

Pro zautomatizování určitých operací prováděných ručně zaměstnancem může být požadováno, aby byl e-learningový systém propojitelný s jiným systémem do budoucna. Takovým může být například informační systém fakulty, kde se ukládají veškerá hodnocení studenta. Pak by takové propojení systémů mohlo přímo aktualizovat studentovy dosažené výsledky, aniž by je musel do systému zadáv at učitel. To je pouze jeden z příkladů, které mohou v praxi nastat. Aby byl systém schopný takového případného propojení, měl by být navržen a implementován dle standardních postupů a s pomocí rozšířených a podporovatelných nástrojů nebo by měl být snadno škálovatelný.

3.3.3 Shrnutí

Na systém je kladeno mnoho požadavků, které je potřeba zahrnout do návrhu výsledné aplikace. Další požadavky vznikají novým pohledem na e-learning, kterým se zabývá tato práce.

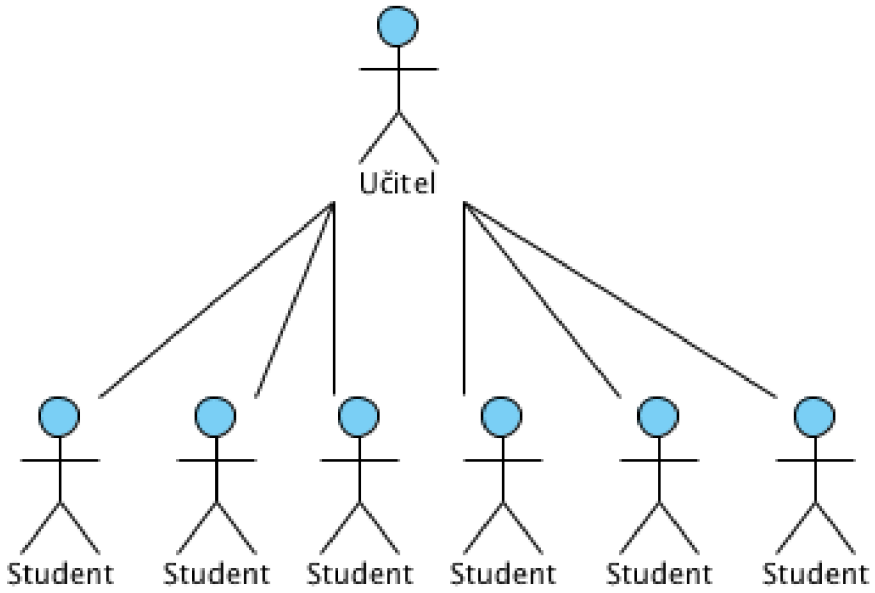
Kapitola 4

Nový pohled na e-learning

Z výše rozebraného teoretického úvodu je patrné, že pokud vznikne požadavek na e-learningový systém nebo jen podporu výuky určitého kurzu, je výhodnější využít stávajících řešení, kde je případně možno cokoliv doplnit (nový modul u open source systému). Avšak přijde-li požadavek příliš specifický a to v takovém smyslu, že by nabourával celý model dnes existujících řešení, pak není možno tato řešení využít resp. je efektivnější navrhnout a naimplementovat nový systém.

4.1 Stávající model e-learningu

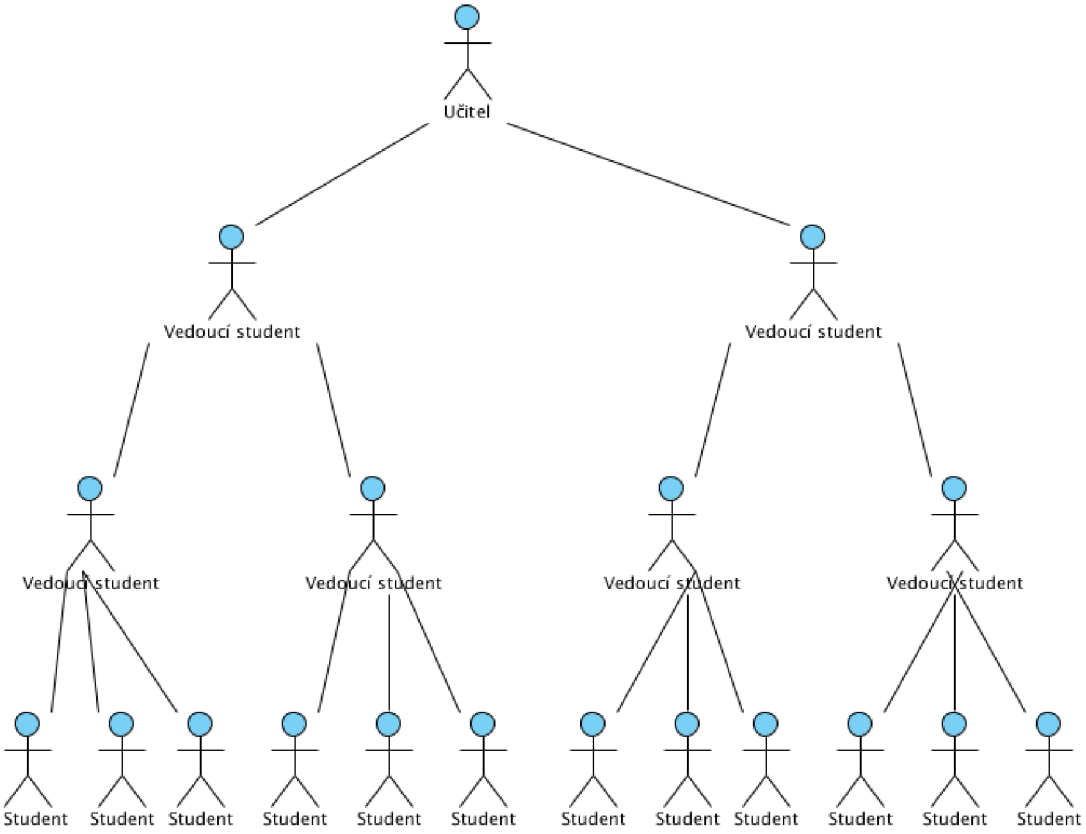
Klasický e-learning tak, jak je dnes znám, má třívrstvou hierarchii ve smyslu uživatelském. Tedy na nejnižší úrovni a s nejnižšími právy je student, nad ním je učitel a nad učitelem se všemi právy je administrátor. Takto to v podstatě funguje i v reálném světě, zejména tedy vztah studenta a učitele. Tento model je osvědčený, nemusí být však optimální pro všechny situace.



Obrázek 4.1: Hierarchie uživatelských vrstev ve stávajících e-learningových systémech

4.2 Nový model e-learningu - hierarchie studentů

Nový model e-learningu využívá hierarchie studentů. Studenti tvoří skupiny, které mají ve správě další studenti. Student je motivován vést skupinu bonusovým bodovým hodnocením závislým na úspěšnosti své skupiny. Vedoucího studenta určí učitel, nebo sám systém podle aktuálního bodového hodnocení. Začátek studia proto může fungovat stejně, jako klasický e-learningový model a hierarchie se začne budovat až v pokročilejších fázích, kdy lze určit vhodné vedoucí studenty.



Obrázek 4.2: Hierarchie uživatelských vrstev v novém modelu

4.2.1 Co se mění pro studenty

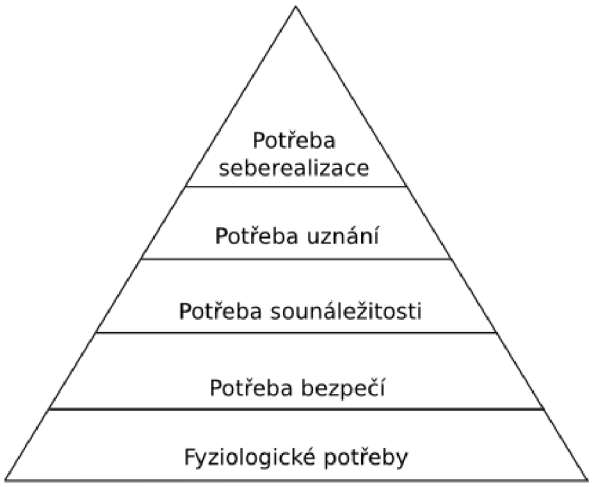
Na počátku kurzu je každý student veden pouze učitelem. Postupem času získává student body za aktivitu, testy apod., na základě určitého počtu bodů se student může stát vedoucím studentem. Takovým způsobem se dostanou výše bystřejší studenti, kteří následně dostanout přidělenou skupinku studentů. Těmto studentům pak s dotazy a učivem pomáhají výše postavení studenti. Tito studenti mohou být motivování k dobrým výsledkům celé skupiny například bodovým hodnocením nebo penězi (podobně jako studenti s dobrým studijním průměrem získávají stipendia). Pak je v jejich zájmu, aby studentům pod sebou předávali správné informace. V případě negativních výsledků celé skupiny pak klesá v hodnocení i student ve vedení. Skupin je více a může mezi nimi probíhat určité soutěžení, které také napomůže k dosažení lepších výsledků. Studenti se mohou díky tomuto systému seberealizovat. Tento systém si vyžaduje úpravu práv v systému. Student však

nikdy nedosáhne na pozici či práva učitele.

Motivace z pohledu psychologie

Abraham Maslow přišel roku 1943 s teorií pyramidové skladby lidských potřeb. Dle jeho názoru se lidská osobnost soustředí na vyšší stupeň dosažení cílu jen tehdy, pokud jsou splněny potřeby na nižších úrovních.

Nejzákladnější lidskou potřebou jsou fyziologické potřeby, mezi které se řadí například dýchání, potravu a tekutiny a další. Pokud je splněna tato potřeba, přichází na řadu potřeba bezpečí, čili potřeba zajištění a uchování existence. Následuje potřeba sounáležitosti, která se váže na lásku a přátelství, obecně začlenění se do větší skupiny. Potřeba uznání zahrnuje respekt a ocenění jednotlivce ze strany ostatních. Nejvyšší lidskou potřebou je tedy potřeba seberealizace, která představuje potřebu jednotlivce využít své schopnosti. [21]



Obrázek 4.3: Hierarchie lidských potřeb podle Maslowa

4.2.2 Role učitele v novém modelu e-learningu

Nové pojetí procesu výuky této práce pomáhá delegovat některé pravomoci na studenty. Tato schopnost sytému umožňuje snížit časovou náročnost předmětu a vytvořit prostor vyučujícímu ke zkvalitnění předmětu. Například není vyloženě nutné přímo kontrolovat vedoucí studenty, jaké předávají svým studentům informace, neboť je v jejich zájmu dobré hodnocení celé skupiny. Učitel má však možnost zasahovat do hierarchie studentů viz kapitola Use-case diagram níže.

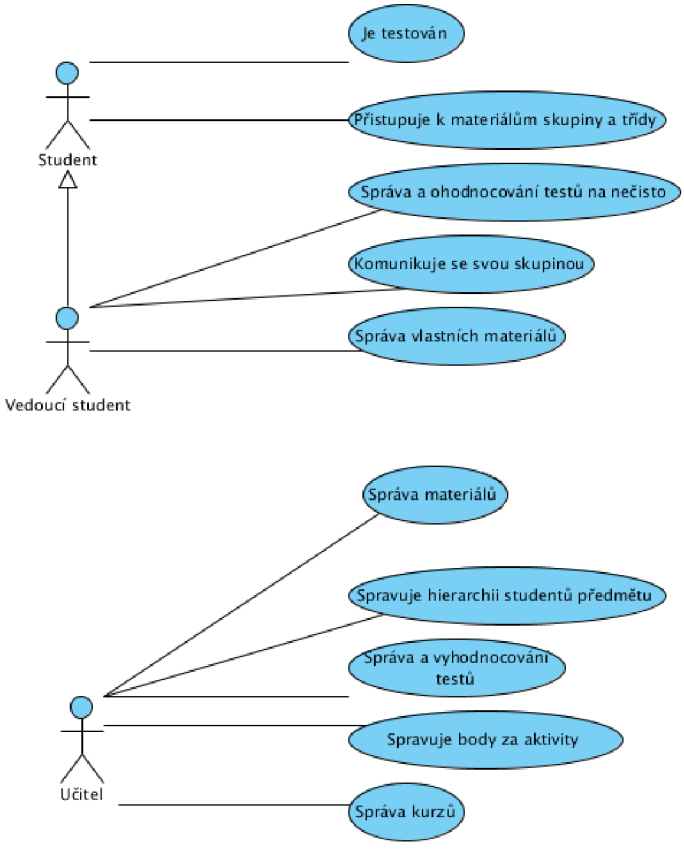
4.2.3 Use-case diagram

Use-case diagram znázorňuje, že aktér vedoucí student je specializací aktéra studenta. Vedoucí student má určitá privilegia navíc. Co je společné pro všechny studenty je, že jsou testováni testy, které spravuje učitel a dále mají přístup k materiálům přiřazeným skupině nebo třídě. Vedoucí student má pak navíc přístup ke správě vlastních testů na nečisto, nejen pro svou potřebu, ale zejména pro pomoc při výuce jemu přiřazených studentů. Dále má vedoucí student možnost komunikovat se svou skupinou především pomocí diskuzního fóra, kde má vyhrazený prostor pro svou skupinu a je zároveň moderátorem. Zde má vedoucí

student možnost pomáhat objasňovat případné nejasnosti. Vedoucí student může také vytvářet a sdílet vlastní studijní materiály, které mají za cíl napomoct k pochopení určité látky.

Aktér učitel má za úkol, jako i v klasických e-learningových systémech, spravovat a vyhodnocovat testy pro studenty a spravovat kurzy. Ke správě se přidávají i materiály, kde má učitel možnost ohodnotit vedoucího studenta body za zpracovaný materiál, má možnost zpřístupnit vedoucím studentem zpracovaný materiál ostatním či vybraným skupinám a dále musí vytvořené materiály nejdříve schvalovat než jsou zpřístupněny ostatním studentům. Dalším podstatným případem použití učitele je, že má možnost měnit hierarchii studentů v předmětu. Jakmile dosáhne student (nebo vedoucí student) další úrovně v hierarchii, je povýšen učitelem. Avšak student může být ze své funkce sesazen, pokud se nedaří celé skupině nebo v případě prohřešků (toto záleží na učiteli). Student nebo vedoucí student může získávat body nejen za testy a dobrý prospěch celé skupiny, ale také za různé aktivity. Tyto aktivity vypisuje učitel nebo je vyplňuje zpětně.

Tento use-case diagram nemá za cíl popisovat veškeré případy použití. Je zřejmé, že uživatelé e-learningového systému se do něj budou přihlašovat, budou mezi sebou komunikovat a jiné, z použití systému, vyplývající pžípady použití. Cílem však je naznačit odlišnost nového modelu od stávajících, vyzdvihnout nové případy použití, které se v běžných e-learningových systémech nevyskytují.



Obrázek 4.4: Use-case diagram

Kapitola 5

Existující řešení

E-learningové systémy jsou v dnešní době čím dál více používány a do budoucna se budou nejspíše objevovat ve větší míře a to z toho důvodu, že je snaha učební a testovací proces co nejvíce automatizovat. Díky tomuto odpadá proces opravování testu po studentovi a případné zadávání dosažených bodů do systému. Student si může navíc ihned ověřit správnost svých odpovědí.

Existující řešení lze rozdělit do dvou částí: komerční řešení a nekomerční řešení. Komerční řešení, tedy taková, která nejsou volně šířitelná za účelem zisku, představují větší množinu oproti nekomerčním, otevřeným řešením.

5.1 Nekomerční řešení

Níže uvedená nekomerční řešení jsou zároveň Opensource řešeními, tedy je případně možno i poupravit jejich zdrojové kódy či dopsat plugin dle potřeby.

5.1.1 Moodle

Moodle je softwarový balíček pro tvorbu výukových systémů a elektronických kurzů na internetu. Tento projekt se neustále vyvíjí, je navržený na základě sociálně konstruktivistického přístupu k vzdělávání. [11]

Je to Opensource řešení spadající pod obecnou veřejnou licenci GPL (General Public License). Moodle je naprogramovaný v PHP a podporuje různé typy databází, zejména pak PostgreSQL a MySQL.

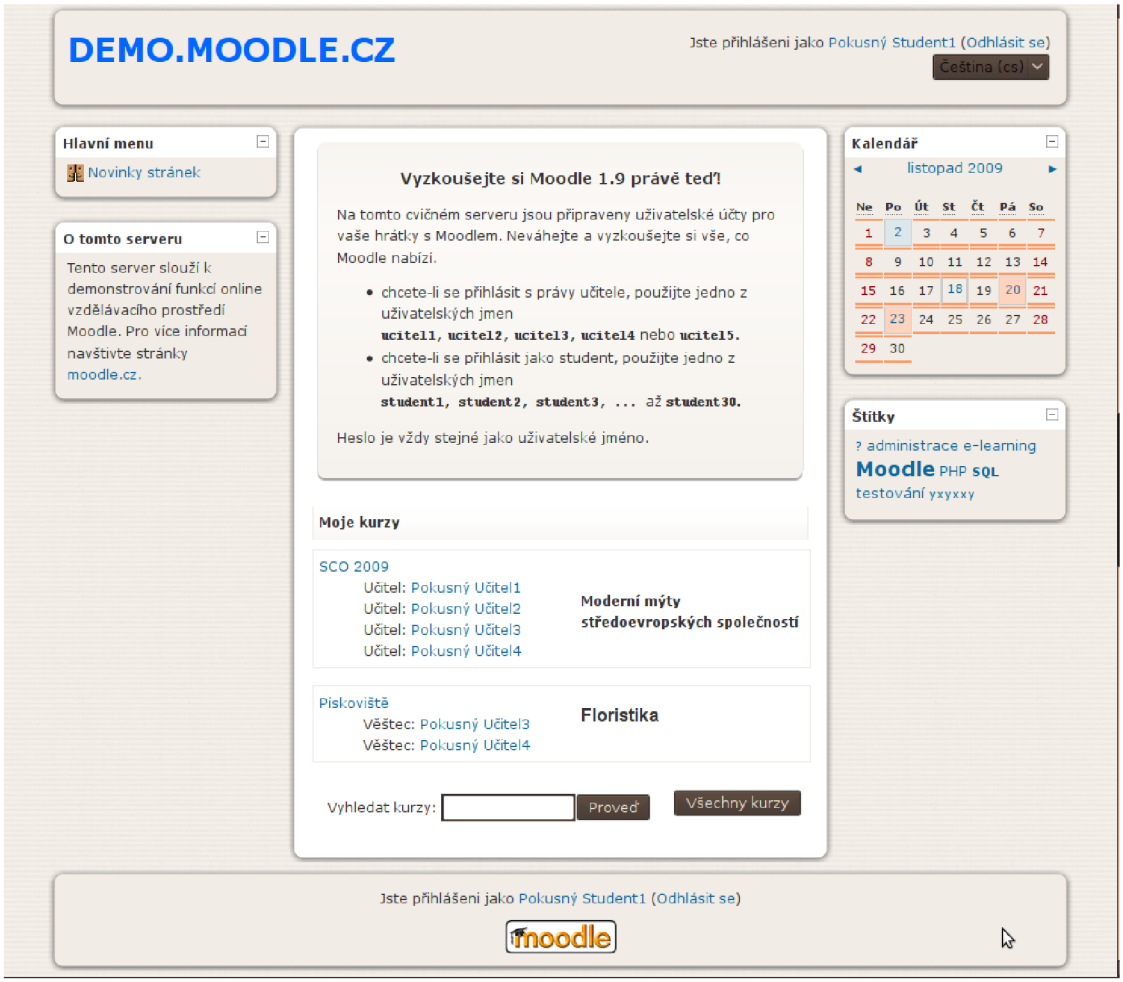
Původně bylo slovo Moodle akronymem pro Modular Object-Oriented Dynamic Learning Environment (Modulární objektově orientované dynamické prostředí pro výuku). [11]

Do systému Moodle se lze přihlásit třemi různými rolemi. Těmi jsou administrátor, který spravuje systém, učitel, který vytváří kurzy, testové otázky a jiné a student, jež je zkoušen.

Moodle má svoji wiki, na které jsou doplňující informace, zejména jsou pak zde rozepsané všechny tři role podrobněji. Nechybí dokumentace ke stažení a další rozšiřující moduly. Prostředí systému Moodle je na obrázku 5.1.

Řešení komunikace mezi uživateli

Moodle nabízí tři typy komunikace mezi účastníky, což je velmi pozitivní fakt. První možnozstí, která je Moodle dostupná od jeho prvopočátku, je diskuzní fórum. Jedná se o asyn-



Obrázek 5.1: Ukázka prostředí e-learningového systému Moodle

chronní komunikaci, kde může komunikovat každý s každým a takové fórum (nebo jeho určitou část) pak moderuje učitel.

Druhou možností komunikace je zaslání soukromé zprávy přímo vybranému uživateli v rámci systému Moodle. Jedná se také o asynchronní komunikaci, pouze však mezi dvěma uživateli.

Poslední, třetí, možností je komunikace pomocí chatu. Jde o synchronní komunikaci, obvykle menší skupiny (ne každý s každým jako u diskuzního fóra). Taková forma komunikace probíhá v reálném čase a je zapotřebí stálého připojení k počítačové síti.

Z předešlých uvedených možností komunikace tedy vyplývá, že Moodle je v tomto směru dobře vybaven. Komunikace je ve vzdělávání velmi podstatnou částí, která podporuje proces učení.

Testový modul

Testy lze vytvářet učitelem pouze v režimu úprav. Testy se skládají z jednotlivých úloh, které jsou předem vytvořeny a přiřazeny do určitého kurzu. Typů úloh poskytuje Moodle celou řadu. Od klasických výběr 1 z N az po doplňovací úlohu (tzv. Cloze), ve které student musí například doplnit text, vybrat jednu z možností nebo zadat číselný údaj. Tyto úlohy však zatím nemají pro jejich tvorbu grafické rozhraní, a tak je nutno zadávat text úlohy spolu s řídicími příkazy pomocí vestavěného editoru nebo pomocí předem připrvených souborů. [10]

Testový modul neslouží pouze pro závěrečné testy, ale také pro výuku studentů. Je možno, dle nastavení, testy opakovat a případně zobrazit správné odpovědi nebo učitelovy komentáře k jednotlivým otázkám. Díky získaným výsledkům z testů má učitel také přehled o obtížnosti látky.

Výukový modul

Pro práci se studijními materiály musí učitel uvést systém do režimu úprav. V této fázi může být vložen studijní materiál k danému kurzu. Moodle podporuje řadu různých typů souborů. Jako studijní materiál může být do kurzu vložen i odkaz na webovou stránku.

Výuku pak podporují další činnosti, které se v Moodlu nachází a těmi jsou například přednáška, diskuzní fórum nebo wiki, díky které mají studenti možnost pracovat na projektech společně.

Jak už je zmíněno výše, pro podporu výuky slouží rovněž i testový modul, kde si mohou studenti i opakovaně spustit testy a tak si ověřit již získané znalosti.

Zhodnocení

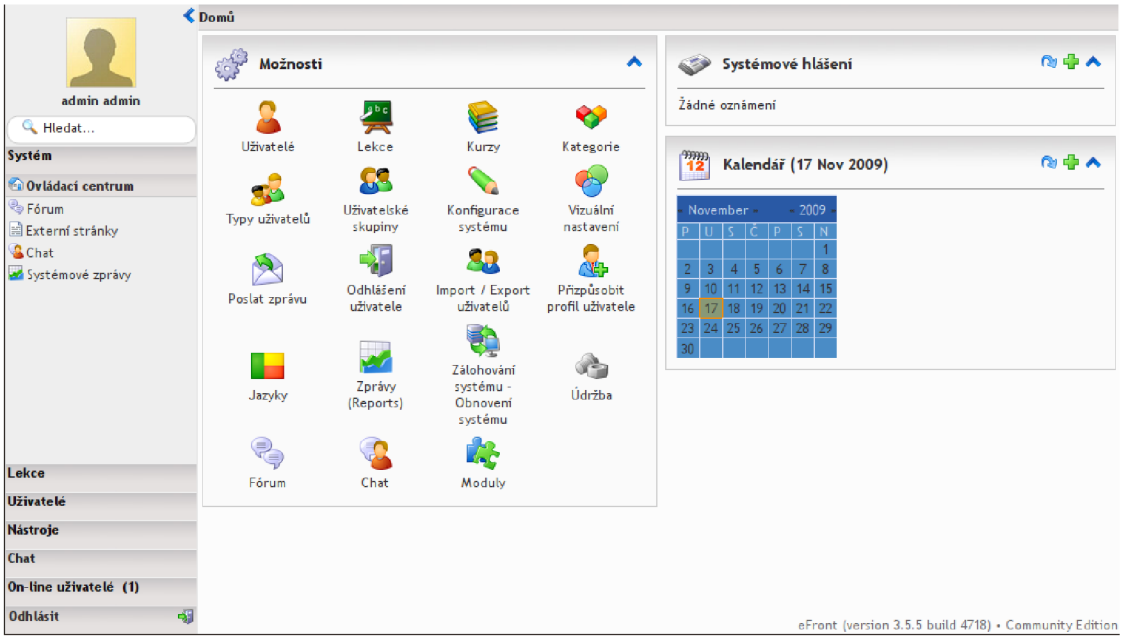
Moodle je klasický systém pro správu kursů, který je poměrně hodně rozšířen. K tomu nepřispívá pouze fakt, že jde o nekomerční řešení, ale také to, že v základu má spoustu dobrých a použitelných modulů pro vzdělávání studentů a pokud je potřeba tento systém určitým způsobem rozšířit, exituje spousta doplňkových modulů, které přispívají ke spokojenosti uživatelů Moodle.

Na druhou stranu má Moodle poměrně neintuitivní grafické rozhraní, ve kterém se špatně orientuje. Uživatelé Moodlu postrádají tutoriály a česká dokumentace není dokončena. Například k vytvoření testu je potřeba znát dobře funkčnost systému. Běžný uživatel, který s Moodlem nemá žádné zkušenosti na to snadno nepřijde.

5.1.2 eFront

E-learningový systém eFront je dostupný ve více variantách a těmi jsou: eFront Enterprise, eFront Educational a eFront Community Edition. Nekomerčním řešením je z uvedených pouze eFront Community Edition, který používá licenci CPAL (Common Public Attribution License). Vývoj eFrontu započal už v roce 2001, Opensource řešení ale vzniklo až v roce 2007. V dnešní podobě je eFront značně komplexní a bytelný systém. Lze jej vužít jako:

- učební nástroj
- komunikační nástroj
- vyhodnocovací nástroj
- certifikační nástroj
- nástroj pro sdílení a správu souborů [5]



Obrázek 5.2: Ukázka prostředí e-learningového systému eFront

Mezi vlastnosti eFrontu a také výhody patří to, že je multiplatformní a lze jej užívat v jakémkoli internetovém prohlížeči, dále je multijazyčný, podporuje češtinu a využívá moderních webových technologií, kterou je například Ajax. Je velmi přehledný a jednoduchý na ovládání. Podobně, jako u systému Moodle, existují v eFrontu tři role pro přihlášení do systému: administrátor, učitel a student. eFront má také svou wiki, dokumentaci ke stažení a doplňky. Pro vznik eFrontu byl použit volně dostupný Zend framework v PHP a jako webový server lze použít Apache nebo XAMPP. Jak vypadá prostředí systému eFront lze vidět na obrázku 5.2.

Zhodnocení

eFront je založen na stejné podstatě jako Moodle, avšak oproti Moodlu má rozhodně přehlednější a intuitivnější grafické rozhraní. To může být způsobeno tím, že eFront existuje

i jako komerční řešení, a proto se možná dostává kvalitnější péče i nekomerčnímu řešení v rámci jedné skupiny, která má tento systém na starosti.

5.2 Komerční řešení

Existující komerční řešení jsou zaměřena stejným směrem jako například Moodle. Do systému se přihlašují klasicky tři možné role (student, učitel, administrátor), student má k dispozici studijní materiály a testy, učitel je vytváří a administrátor systém spravuje.

Příkladmi takových řešení jsou AES (Asynchronní E-learningový Systém), eLeap nebo, jak už je zmíněno výše, eFront, který je ve formě komerční i nekomerční.

5.3 Rozdíly mezi komerčním a nekomerčním řešením

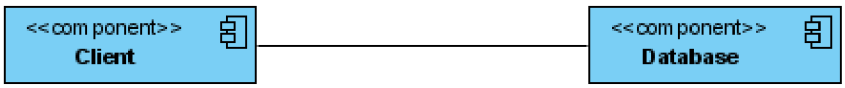
Výhody nekomerčního opensource jsou v možnosti mít kompletní náhled do funkce systému a v případě potřeby doimplementovat chybějící funkčnost. Nevýhodou je chybějící podpora ze strany dodavatele a nutnost spoléhat se jen na komunitu.

Komerční řešení jsou většinou uzavřená a neposkytují mnoho možností pro vlastní rozšíření systému. Je proto obvykle nutné toto rozšíření vyvinout na žádost u dodavatele. Výhodou tohoto přístupu je přenos zodpovědnosti na kvalitu systému na stranu dodavatele.

Kapitola 6

Návrh aplikace

Aplikace je navržena jako jedna celistvá aplikace (klient) komunikující s databází 6.1. Architektura klienta je rozdělena do tří vrstev: Model, View, Controller (MVC) viz kapitola 6.3.4. Aplikace spravuje persistentní data jedné databáze. Celkový proces vývoje tak, jak probíhal od začátku do konce, je popsán v kapitole 6.1. Instalace, použité a potřebné nástroje jsou popsány v následujících kapitolách.



Obrázek 6.1: Client jako jeden celek napojený na databázi

6.1 Proces vývoje

Celkový proces vývoje této aplikace je poměrně rozsáhlý a velmi náročný na počet člověkohodin. Systémy této velikosti jsou v komerční oblasti většinou navrhovány a implementovány v týmech o několika programátorech, analyticích a softwarových inženýrech.

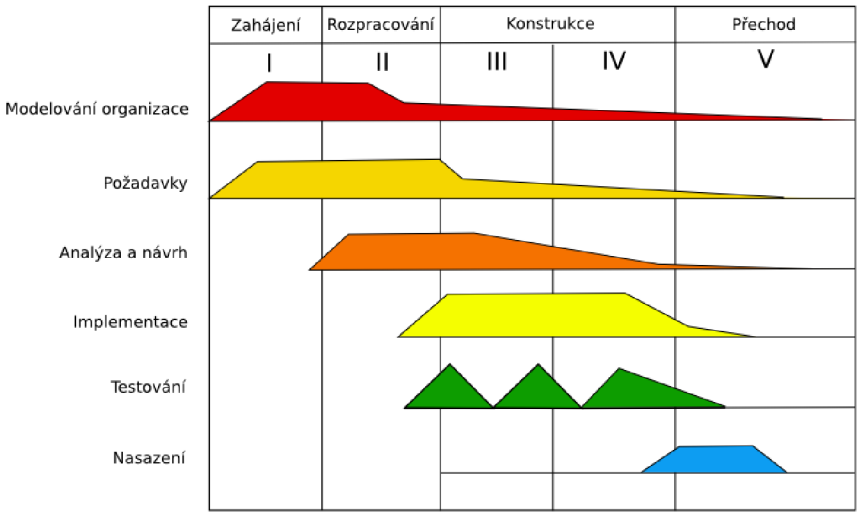
Vývoj lze rozdělit do několika iterací, počínaje návrhem, přes implementaci, konče testováním a laděním. Počátky návrhu vznikaly nejdříve jako náčrtky na papíře. Takto se dal návrh snadno měnit a rychle překreslovat. Po dostatečným promyšlení byl návrh překreslen do UML diagramů v elektronické podobě. Dle hotového návrhu započaly práce na implementaci, která byla pomyslně rozdělena do iterací tak, aby byl na počátku vytvořen základ aplikace, na kterém se dá postupně dále stavět. Tento základ představoval vývoj administrace, ve které lze spravovat uživatele a kurzy.

Po dokončení administrace byla dále implementována část pro učitele, která je rozšířením pro ostatní dvě role: student a vedoucí student. Pro otestování správné funkčnosti částí role učitel byla následně implementována část pro studenta, ve které tak bylo možno otestovat zejména spouštění testů a výpis získaných bodů. Další iterací pak byla část vedoucího studenta, která je v podstatě rozšířením role student a některé části má velmi podobné s učitelem. Po dokončení této iterace přišlo na řadu testování a ladění, díky kterému se odstranily nalezené chyby. Poslední implemetační částí pak bylo vytvoření designu aplikace, což probíhalo společně s dalším testováním a laděním aplikace.

I když byl projekt vypracováván jedním člověkem, bylo použito zjednodušeného projektového řízení Unified Process.

6.1.1 Řízení projektu na bázi Unified Process

Projekt byl rozdělen do pěti iterací, které znázorňuje obrázek 6.2.
Jedntolivé iterace probíhaly takto:
I - říjen - prosinec 2009
II - leden - únor 2010
III a IV - březen - duben 2010
V - květen 2010



Obrázek 6.2: Jednotlivé iterace projektu

6.2 Databáze

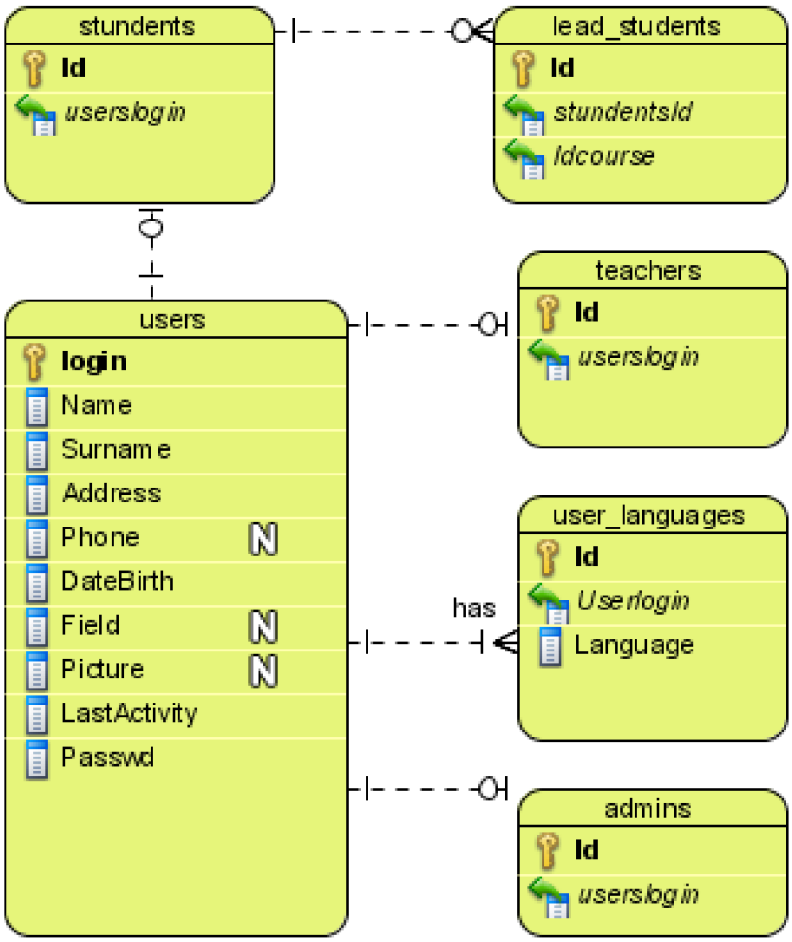
Pro správu perzistentních dat, které jsou částečně generovány a částečně získávány interakcí s uživatelem, je nutné použít určitá forma databáze. V tomto projektu byla využita relační databáze, která tvoří nižší vrstvu pro standardní úložiště dat rámce Ruby on Rails - Active Record. Pro tyto účely je zde použita databáze MySQL, která je popsána v následující kapitole.

Čas strávený nad návrhem databáze v době návrhu aplikace by němel být podceňován. V případě rámce Ruby on Rails je databáze hlavním nosníkem modelové vrstvy, která drží kostru aplikace. Změny v modelu projektu v době implementace či dokonce v pozdější fázi projektu jsou pak tvrdě vykoupeny režií reimplmenetace vlastností projektu.

Tabulky 6.1 a 6.2 zobrazují rozdíl v nákladech modelového projektu v případě vynechaných respektive zahrnutých příprav v projektech vypracovávaných iterativně.

Celkový pohled na návrh databáze je v příloze. Některé části ER diagramu jsou popsány v následujících odstavcích.

Uživatelé jsou sdružováni v jedné tabulce *users*, která má vazby 1:1 na tabulky s názvy jednotlivých rolí. Výjimkou jsou vedoucí studenti, kteří nemají vazbu přímo na tabulku *users*, ale na tabulku studentů. Každému uživateli je také přiřazen alespoň jeden jazyk. Uživatele a jejich jazyky sdružuje tabulka *user_languages* viz obrázek 6.3.



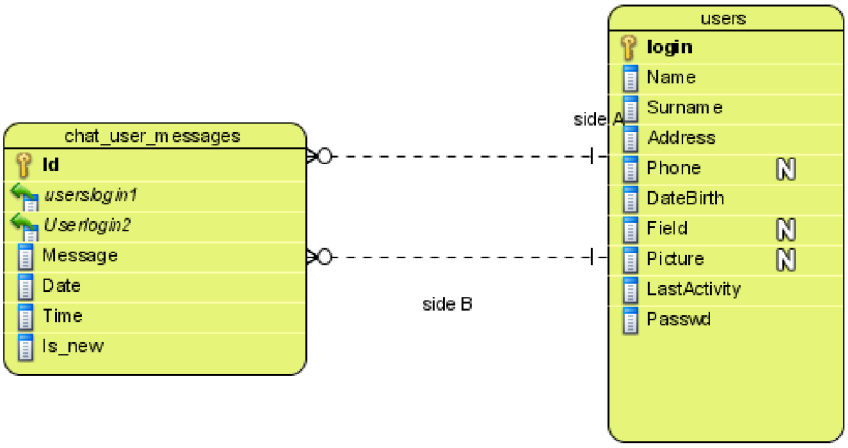
Obrázek 6.3: Databázové tabulky pracující s uživateli

Ajaxový chat využívá pro komunikaci dvou tabulek a to *users* a *chat_user_messages*, ve které jsou jako cizí klíče využity dva loginy uživatelů pro budoucí výpis historie chatu. Tato struktura je zobrazena na obrázku 6.4.

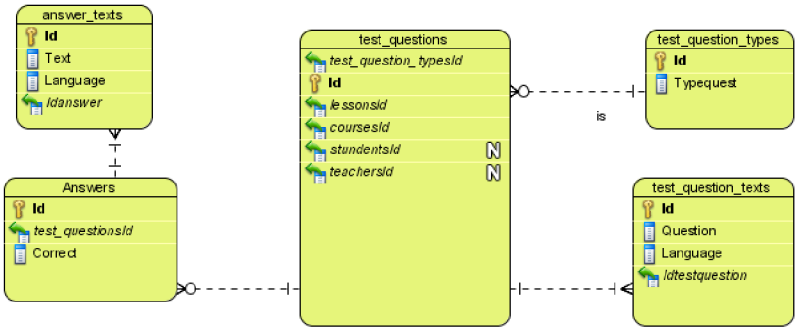
Další zajímavou částí jsou testové otázky, které jsou co do počtu tabulek poměrně rozsáhlé viz obrázek 6.5. Tabulka pro testové otázky je sama o sobě tvořena primárním klíčem a cizími klíči, které se odkazují na tabulky pracující s dalšími daty. Otázky mohou nabývat tří typů a tyto jsou uloženy v tabulce *test_question_types*. Každá otázka má znění, které však může být ve více jazycích a proto je zde vytvořena speciální tabulka *test_question_texts*, která tyto znění v různých jazycích sdružuje. Stejně je řešena tabulka *answers* a *answer_texts* uchovávající odpovědi k otázce, která není fulltextová. Fulltextová otázka nemá v databázi žádnou vazbu na tabulku s odpověďmi.

6.2.1 MySQL

Systém řízení báze dat MySQL společnosti Oracle je velmi často používán v souvislostí s webovými aplikacemi a projekty menšího a středního rozsahu. Základní verze systému je volně k použití i pro komerční produkty. „Ve srovnání s jinými databázovými systémy patří MySQL spíše k těm jednoduchým, ale zase je nenáročný na zdroje počítače a u některých operací nabízí zvýšení rychlosti.“ [14]



Obrázek 6.4: Databázové tabulky pracující s chatem



Obrázek 6.5: Databázové tabulky pracující s testovými otázkami

Databázový systém MySQL je postaven na principu platformy, na které jsou postaveny různé typy datových úložišť. Datová úložiště mají různé vlastnosti zaměřené na specifické či obecné použití. Dvě základní úložiště jsou popsány v následujících odstavcích.

Úložiště MyISAM

Na některých platformách je toto úložiště jako výchozí. MyISAM je velmi výhodné použít tam, kde je potřeba jednoduchých vysokorychlostních operací, které nevyžadují záruku integrity transakcí. Rychlejší odezva systému je také zajištěna možností umísťovat soubory s daty a s indexy do samostatných adresářů a je zde podpora indexů sloupců BLOB a TEXT.[18]

Úložiště InnoDB

Tento typ úložiště byl použit v tomto projektu proto, že MyISAM nepodporuje cizí klíče. Na úkor rozšířených vlastností je InnoDB oproti MyISAM v jistých operacích pomalejší. Dalšími benefity jsou rozšířené možnosti pro souběžnou práci více uživatelů a dále úložiště disponuje možností zamykání řádků, a ne pouze celých tabulek. InnoDB je vhodné použít tam, kde se provádějí kritické transakce a je také vhodné pro ukládání velmi rozsáhlých dat. Tabulky InnoDB potřebuji obecně více diskového prostoru než tabulky MyISAM. [18]

Tabulka 6.1: Výsledek přeskočení etapy příprav v iteračních projektech [9]

Status projektu	Náklady na práci	Náklady na přepracování
20%	100 000 Kč	75 000 Kč
40%	100 000 Kč	75 000 Kč
60%	100 000 Kč	75 000 Kč
80%	100 000 Kč	75 000 Kč
100%	100 000 Kč	75 000 Kč
Přepracování na konci projektu	0 Kč	0 Kč
Celkem	500 000 Kč	375 000 Kč
Celkový součet		857 000 Kč

Tabulka 6.2: Výsledek zahrnutí příprav do iteračních projektů [9]

Status projektu	Náklady na práci	Náklady na přepracování
20%	100 000 Kč	10 000 Kč
40%	100 000 Kč	10 000 Kč
60%	100 000 Kč	10 000 Kč
80%	100 000 Kč	10 000 Kč
100%	100 000 Kč	10 000 Kč
Přepracování na konci projektu	0 Kč	0 Kč
Celkem	500 000 Kč	50 000 Kč
Celkový součet		550 000 Kč

Dostupná omezení

V MySQL existuje několik omezení, díky kterým lze zvýšit výkon a také zajistit integritu dat. Ke kontrole nastavených omezení dochází obvykle při operacích INSERT, UPDATE a DELETE, neboť dochází ke změně dat. Omezení použita v tomto projektu jsou následující:

- PRIMARY KEY - představuje jedinečný identifikátor řádku, ve skutečnosti je druhem omezení UNIQUE, které kontroluje, zda vkládaná hodnota není již obsažena v daném sloupci. Při vytvoření primárního klíče je zároveň vytvořen index (PRIMARY) a ten zajistí, že dva řádky nemají v tomto sloupci stejnou hodnotu.
- FOREIGN KEY - definuje a zajišťuje vztahy mezi databázovými tabulkami a tím zajišťuje integritu dat. Kromě této funkce může být cizímu klíči definována akce (nebo více akcí), které se provedou při úpravách či odstranění řádků. Typickou akcí, použitou v tomto projektu, je ON DELETE CASCADE, což zajistí odstranění záznamu z tabulky, pokud byl odstraněn primární klíč v tabulce, na níž je vazba definována.
- NOT NULL - do sloupce s tímto omezením musí být vždy vložena data [18]

Výhody používání těchto omezení mohou být následující:

- konzistence dat - je podstatné udržet databázi v konzistentním stavu pro správnou funkčnost aplikace.
- výkon - omezení jsou vykonávaná na straně serveru, což je obvykle efektivnější, než kdyby byla vykonávaná na straně klienta. Dále omezení pomáhá optimalizátoru dotazů systému řízení báze dat zefektivnit proces čtení.

- čas a produktivita vývojářů - definováním omezení se snižuje čas potřebný pro vývoj, systém je jednodušší a funguje do jisté míry také jako prevence chyb. [18]

6.2.2 Webový server

Volba webového serveru

Existuje mnoho webových serverů určených přímo pro spolupráci s programovacím jazykem Ruby, potažmo frameworkem Ruby on Rails. Pro potřeby diplomové práce bylo zvažováno mezi třemi implementacemi webového serveru: WEBrick, Mongrel a Thin. WEBrick je knihovna v Ruby, která vykonává HTTP požadavky aplikace. Mongrel je malý, rychlý HTTP server pro Ruby a JRuby. Posledním ze tří serverů je Thin, který je dle mnoha dostupných výkonnostních měření nejrychlejší. Je složen ze tří navzájem provázaných komponent, které byly v minulosti vyvinuty samostatně pro jiné účely. Tyto celky byly za svou relativně dlouhou historii velmi zoptimalizovány a odladěny, a dle názorů autorů Thinu jsou to nejlepší knihovny ve webové historii programovacího jazyka Ruby.

- mongrel parser - součástí webového projektu Mongrel
- event machine - síťová I/O knihovna s extrémně vysokou škálovatelností, výkonem a stabilitou
- rack - minimalistické rozhraní mezi webovými servery a Ruby frameworky [8]

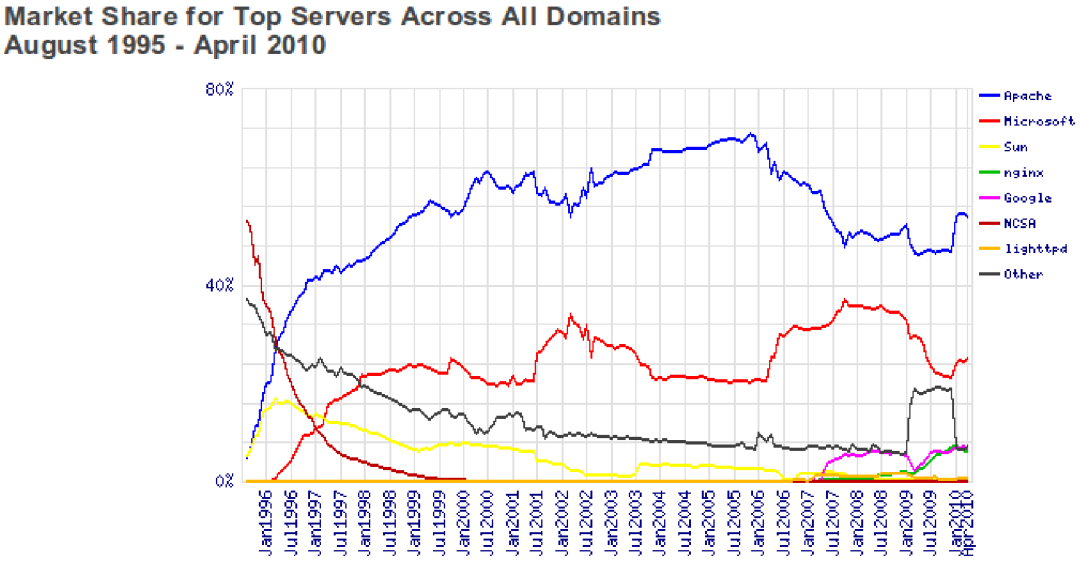
Webový server Apache

Webový server Thin v předcházející kapitole byl použit v době vývoje. V případě nasazení do skutečného provozu by bylo využito webového serveru Apache. Tento webový server je dlouhodobě světově nejrozšířenějším webovým serverem, za jehož vznikem stojí organizace Apache Software Foundation. Je velmi pravděpodobné, že by byl e-learningový systém provozován ve stejném prostředí, ve kterém běží ostatní studijní podpůrné informační systémy. Tyto systémy budou běžet vzhledem k vysokému procentuálnímu zastoupení webového serveru Apache ve světě pravděpodobně také na tomto systému, a proto je vhodné snížit náklady na provoz a údržbu zavedením jednotného systému. Další výhodou webového serveru Apache je jednoduchá konfigurace a malé nároky na specifické vlastnosti operačního systému. To dovoluje běh v prostředí jak Unix-like systémů, MS Windows OS tak mnoha dalších. Je proto velmi vhodný k integraci do stávajícího prostředí.

Rozšířenost Apache dokumentuje aktuální graf 6.6

Populárnost Apache má mnoho důvodů. Některými z nich jsou:

- cena - Apache je zdarma, kvalitativně se však cena na systému neprojevuje. Organizace Apache Software Foundation těží z modelu svobodného softwaru.
- je multiplatformní - dá se provozovat jak na různých architekturách a operačních systémech. Server podporuje jak často zastoupené procesory Intel, tak Alpha, RISC, SPARC a další
- rozšiřitelnost - vzhledem k velké rozšiřitelnosti nabízí spoustu modulů, díky čemuž nabízí velké množství funkcí
- dynamické stránky - je vhodný pro tvorbu dynamicky generovaných stránek díky rozhraní CGI skriptů nebo také díky dalším rozšiřujícím modulům [15]



Developer	March 2010	Percent	April 2010	Percent	Change
Apache	112,747,166	54.55%	110,752,854	53.93%	-0.62
Microsoft	50,572,540	24.47%	51,284,570	24.97%	0.50
Google	14,592,133	7.06%	13,749,829	6.70%	-0.37
nginx	12,673,962	6.13%	12,977,486	6.32%	0.19
lighttpd	1,657,584	0.80%	1,078,403	0.53%	-0.28

Obrázek 6.6: Podíl na trhu webových serverů všech domén [12]

6.3 Nástroje a znovupoužitelný kód

Aby bylo možné vytvořit projekt v rozumném čase, je nutné využít znovupoužitelný kód a pomocné nástroje pro zjednodušení vývoje. Aplikace diplomové práce přímo využívá vrsty webového rámce - frameworku Ruby on Rails, který zajišťuje vrstvu obsluhy uživatelských požadavků (requests). Webový framework je napsán v Ruby a aplikace, které jej využívají musí být také psány ve skriptovacím jazyce Ruby. Pro dynamické chování na straně tenkého klienta - webového prohlížeče - využito skriptovacího jazyka JavaScript. Veškeré tyto nástroje jsou popsány v následujících kapitolách.

6.3.1 Potřebné programy a instalace na operačním systému Linux

Diplomová práce byla vyvíjená na operačním systému Linux/Ubuntu, která je založena na Debian distribuci. Prvním základním kamenem instalace projektu je programovací skriptovací jazyk Ruby, který je volně ke stažení buď na stránkách www.ruby-lang.org nebo ze standardního zdroje distribuce. V případě Ubuntu a jiných na Debianu postavených distribucích lze Ruby nainstalovat příkazem:

```
sudo apt-get install ruby
```

Jazyk Ruby disponuje mnoha knihovnami a rámec Ruby on Rails je jednou z nich. Instalace knihoven zjednodušuje projekt RubyGems, který slouží podobně jako repozitář

Debian distribuce. RubyGems lze nainstalovat buď ze stránek projektu www.rubygems.org, nebo ze standardního zdroje linuxové distribuce. V případě Ubuntu nebo jiné na Debianu postavené distribuce jde o příkaz:

```
sudo apt-get install rubygems
```

Pro běh aplikace je potřeba instalace frameworku Ruby on Rails (RoR) a klienta MySQL databáze pro Ruby. Ruby on rails lze stáhnout na adrese: www.rubyonrails.org nebo jej lze nainstalovat přes RubyGems následovně:

```
sudo gem install rails
```

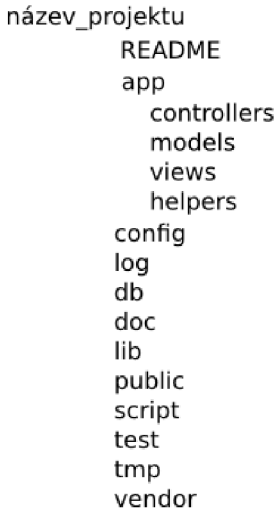
Databázi MySQL je možno najít na adrese: www.mysql.com nebo rovněž nainstalovat pomocí RubyGems:

```
sudo gem install mysql
```

A na závěr webový server thin, jehož instalace probíhá:

```
sudo gem install thin
```

Po instalaci všech balíků se pak vytvoří projekt jednoduše pomocí příkazu „rails nazev_projektu“ vykonáním v příkazové řádce. Takto vytvořený projekt má pak adresářovou strukturu znázorněnou na obrázku 6.7



Obrázek 6.7: Adresářová struktura nově vytvořeného projektu v Ruby on Rails

6.3.2 Ruby

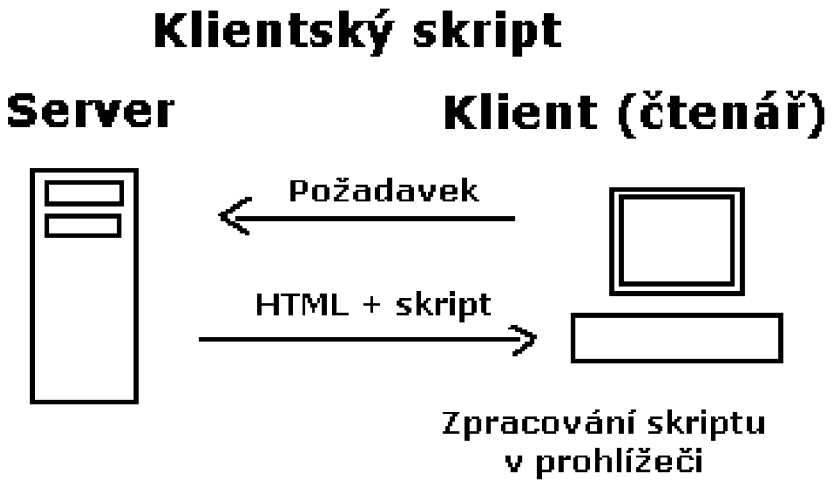
Autorem jazyka Ruby je Yukihiro Matsumoto. Ruby je oproti svým konkurentům mladým jazykem, který se začal dostávat do popředí až v posledním desetiletí. Je to dynamický skriptovací jazyk pro všeobecné použití a je objektově orientovaný. V Ruby je objektem všechno. Například řetězcový literál „string“ je také objekt a má některé základní metody, jako například *length*, což vrátí délku řetězce.

Velkou výhodou programovacího jazyku Ruby je v neposlední řadě velmi obsáhlá dokumentace, která nabízí nejen oficiální reference, ale také zkušenosti uživatelů. Vzhledem k masivnímu rozšíření v poslední době, kdy jazyk Ruby velmi nabývá na popularitě, je snadné najít s pomocí internetových vyhledávačů řešení problémů, s kterým se již některý z programátorů setkal.

Pro otestování správné funkčnosti metody nebo pouze ověření existence některé metody v Ruby existuje interaktivní Ruby prostředí *irb*. Jde o konsolový program, který po spuštění očekává zadání vstupu a na základě tohoto vstupu vypíše odpovídající výstup.

6.3.3 JavaScript

JavaScript je objektově orientovaný programovací jazyk vykonávaný na straně tenkého klienta, webového prohlížeče. Postupem času získává JavaScript stále větší popularitu a má zaručenou budoucnost ve standardu HTML5. 6.8



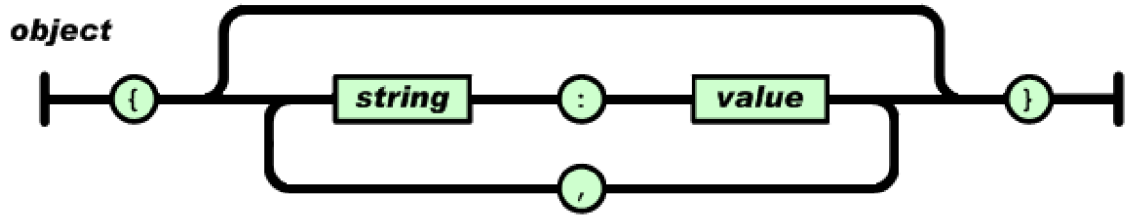
Obrázek 6.8: Zpracování požadavku [4]

JavaScript je jazyk:

- interpretovaný - nemusí být kompilován
- objektový - využívá objektů prohlížeče a zabudovaných objektů
- závislý na prohlížeči - funguje ale ve většině prohlížečů
- case sensitivní - záleží na velikosti písem v zápisu
- syntaxí je podobný jazykům C, Java. [4]

Pro práci s JavaScriptem bylo v tomto projektu využito knihovny jQuery, která zjednodušuje zápisy funkcí a obsahuje často používané obraty ve formě standardních funkcí. JQuery je zdarma a je to multiplatformní, open source řešení. S pomocí jQuery je dále možno vyvíjet nové pluginy. Jedná se o pměrně mladou knihovnu, která byla vydána roku 2006. Pro načítání dat z databáze do JavaScriptu, potažmo webového prohlížeče, který následně zobrazí data na výstup uživateli, bylo využito technologie JSON (Javascript Object Notation). JSON je textový formát, který je nezávislý na jazyku, avšak používá konvence,

které jsou známé programátorům jazyka C, C++, Java a další. JSON je postaven na dvou strukturách: množina párů jméno/hodnota a seřazený seznam hodnot. JSON však může představovat také objekt, což je neseřazená sada párů jméno/hodnota. [7] Tato struktura je naznačena na obrázku 6.9.



Obrázek 6.9: Struktura pro z8pis objektu v JSONu [7]

6.3.4 Framework Ruby on Rails

Ruby on Rails je navržen zejména pro tvorbu webových aplikací podložený praxí společnosti 37signals. Společnost hledala vhodný nosný jazyk pro svůj rámec, který by usnadnil vývoj stále se opakujících postupů tvorby webových aplikací. Jako nejvhodnější kandidát výběrového řízení se ukázal skriptovací jazyk Ruby.

Architektura je založena na paradigmatu MVC a je popsána v následující kapitole. Návrh Ruby on Rails se řídí dvěma zásadními koncepty, těmi jsou:

- DRY „Don't repeat yourself" - Základní myšlenka tohoto přístupu je - každá část kódu má své místo - každá část kódu by měla by být umístěna právě a pouze tam. Tohoto je docíleno také principem jazyka Ruby, takže v aplikaci založené na tomto frameworku se objevuje velmi málo duplicit.

- konvence před konfigurací - tím je myšleno, že framework obsahuje na mnoha místech implicitní hodnoty, které není většinou nutné předefinovávat. V případě, kdy je uživatel rámce nucen opustit implicitní chování systému, Ruby on Rails to jednoduše umožní.[17]

Scaffolding - pomocné generátory

Ruby on Rails obsahuje pomocné skripty, které jsou uloženy v adresáři rámce Ruby on Rails. Tyto scripty vygenerují kostru často se opakujícího kódu za programátora. Tento princip nijak nenarušuje koncepci DRY - Don't Repeat Yourself. Ba naopak, programátor se vyhne opakování při akcích spojených při vytváření souborů a dalších systémových akcí. Vygenerovaný kód je předurčen k následné modifikaci. Pokud například programátor vytváří novou stránku, použije script, který vytvoří na všechna důležitá místa soubory a zaregistruje stránku do rámce. Programátor pak soubory otevře a doplní nosnou logiku aplikace.

Helpers - zkrácené příkazy pro prezentační vrstvu

Helper můžeme v rámci Ruby on Rails chápat jako zástupnou funkci pro útržek kódu, který bychom jinak museli psát stále znovu. Pro diplomovou práci je důležitý zejména form helper, který napomáhá jednoduššímu vytváření formulářů. Díky form helperu je pak

možno jednoduše formuláře validovat. Někdy ale dochází k výjimce a validace nejde tímto způsobem implementovat. Tento problém musí být pak ošetřen programátorem standardní cestou bez využití helperů.

Ostatní rozšíření

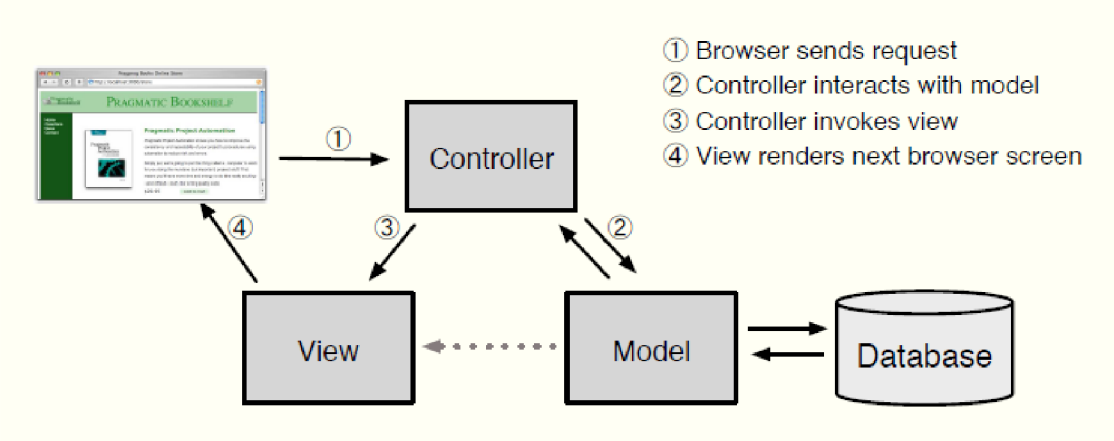
Ruby on Rails čítá mnoho rozšiřujících knihoven, které se ale obvykle instalují až dle potřeby. Zde je, pro srovnání, například rozdíl mezi PHP Zend frameworkem, který obsahuje po instalaci všechny dostupné knihovny. Další ulehčení práce v Ruby on Rails je zajištěno také zavedenými konvencemi - například co se týče pojmenování databázových tabulek.

Ruby on Rails je agilní. Řídí se těmito zásadami:

- jednotlivci a interakce před procesy a nástroji
- funkční software před komplexní dokumentací
- spolupráce se zákazníkem před vyjednáváním smlouvy
- reakce na změny systému před vykonáváním plánu [17]

Model-View-Controller (MVC)

Model-View-Controller je třívrstvá architektura, která rozděluje kód logicky tak, aby byla jednak odstíněna logika od zobrazování a dále aby bylo možno pohodlně komunikovat s databází. Umožňuje snažší psaní a údržbu kódu. Vzhledem k tomu, že je Ruby on Rails na této architektuře postaveno, bylo ji v tomto projektu využito. Každá vrstva má svůj úkol, za který je zodpovědna a případně zasílá nebo příjmá požadavky vrstvy jiné. Toto paradigma prezentuje obrázek 6.10



Obrázek 6.10: Model-View-Controller architektura [17]

MVC - Model

Vrstva modelu zajišťuje veškerou komunikaci s databází a tím zodpovídá za stav aplikace. Z pohledu uživatele se jedná o nejspodnější vrstvu z uvedených, která vykonává CRUD

operace nad databázovými tabulkami. V modelu se definují vazby na jiné databázové tabulky, což umožňuje především případné spojování (join) tabulek na sebe. Vazby, které lze definovat jsou:

- one to one
- one to many
- many to many

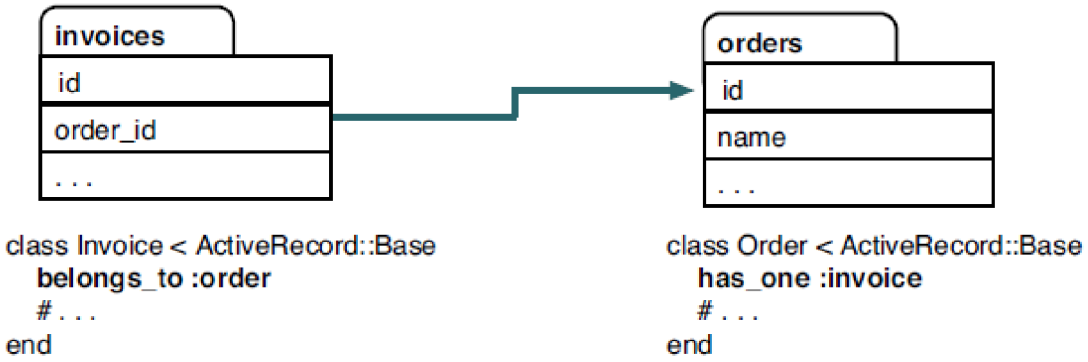
Použití klíčového slova v definici modelu se dále odvíjí od toho, zda tabulka obsahuje cizí klíč nebo ne. Pokud tabulka obsahuje cizí klíč (tedy primární klíč jiné tabulky) a jedná se o vazbu one to one, pak je tato definována jako

```
belongs\_to :nazev\_tabulky\_s\_timto\_primarnim\_klicem
```

a tabulka na druhé straně má definovanou vazbu

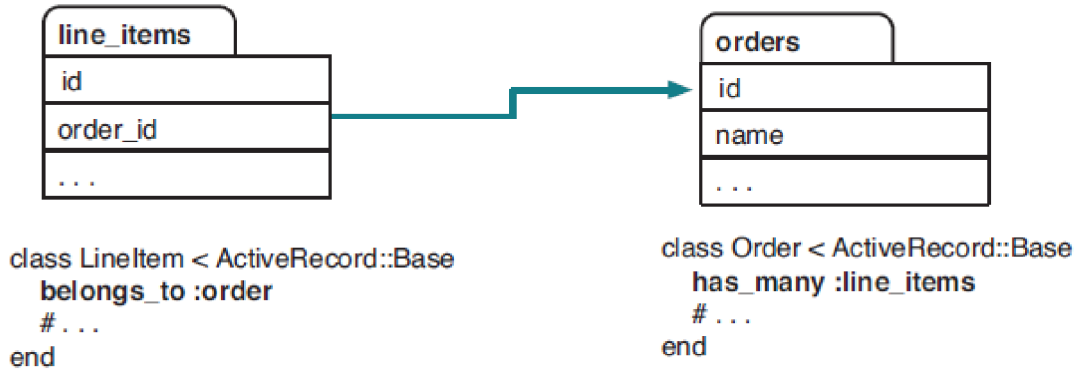
```
has\_one :nazev\_tabulky\_s\_cizim\_klicem.
```

V případě, že se jedná o vazbu *one to many*, pak tabulka obsahující cizí klíč má definovanou vazbu *belongs_to* a tabulka, která má primární klíč definuje vazbu *has_many*. Posledním případem je vazba *many to many*, která je komplikovanější. Pro tento případ se používá v obou modelech stejná vazba a to: has_and_belongs_to_many. Pro tento typ vazby musí v databázi existovat vazební tabulka, která obsahuje cizí klíče obou tabulek. Zde je nutno podoktnout, že RoR má určitou konvenci, co se pojmenování vazební tabulky týče. RoR očekává, že vazební tabulka mezi těmito dvěma tabulkami bude složena z názvu obou tabulek, v množném čísle a k tomu navíc v abecedním pořadí, čili například pokud existuje vazba many to many mezi tabulkami lessons a tests, pak vazební tabulka ponese název lessons_tests. Příklad těchto vazeb je znázorněn na obrázcích 6.11, 6.12 a 6.13.

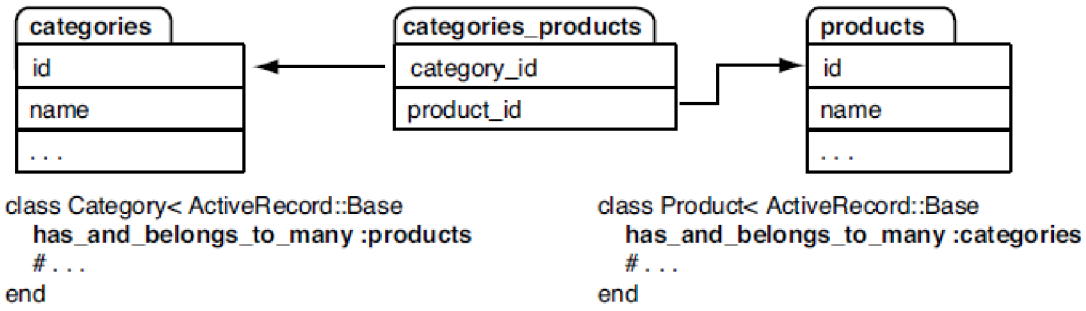


Obrázek 6.11: Příklad vazby one to one v Ruby on Rails [17]

K modelům se váží další zavedené konvence. První důležitou konvencí je ten fakt, že název třídy (modelu) bude začínat velkým písmenem, pokud se skládá z více slov, pak nebudou oddělena žádním znakem, pouze budou začínat velkým písmenem a bude v jednotném čísle. Naopak v databázi se očekává, že tabulka bude mít název zapsaný malými písmeny, pokud obsahuje více slov, budou odděleny podtržítkem a bude zapsána v množném čísle. Tyto pravidla se však týkající anglických názvů tabulek. V případě, že jsou názvy v jiném



Obrázek 6.12: Příklad vazby one to many v Ruby on Rails [17]



Obrázek 6.13: Příklad vazby many to many v Ruby on Rails v Ruby on Rails [17]

jazyce nebo tyto konvence nelze splnit, je pak možnost názvy tabulek předefinovat. Obrázek 6.14 ilustruje příklad těchto pojmenování.

Vzhledem k tomu, že RoR upřednostňuje konvence před konfigurací, je implicitně očekáváno, že databázová tabulka má jako primární klíč sloupec s názvem id. To nemusí platit ve všech případech, avšak je možno primární klíč předefinovat. Problém ovšem nastává, pokud je zapotřebí složeného primárního klíče. Bohužel RoR toto nepodporuje nativně, a tak je nutno použít například vždy sloupec id, který bude primárním klíčem nebo je možnost rozšiřujícího pluginu, který by měl tuto funkčnost zajišťovat.

V modelu se dále řeší validace formulářů, avšak funkce validátorů jsou svázané s používáním form helperů. Pokud je formulář v šabloně zapsán například čistě v html, pak nelze odeslaná data tímto způsobem validovat. V případě form helperu se plní speciální proměnná *error_messages* chybovými hláškami, pokud některá validace neprošla. Odeslaný formulář je dostupný přes speciální proměnnou params, do které se ukládají post a get požadavky. Celý formulář vytvořený pomocí form helperu pak bude v params uložen jako pole hashů (asociativních polí), které je možno přímo vložit do funkce pro vytvoření objektu, který bude následně uložen (nebo upraven) do databáze. Při ukládání do databáze přichází na řadu validace připravené v modelu a pokud některá z nich selže, neprojde podmínka při ukládání (kterou musí zadat programátor) a uživateli se zobrazí chybové hlášky. Při použití form helperu se programátor nemusí starat o znovuplnění formuláře, pokud došlo k chybě. Objekt je znovunačten a uživatel tak nepřijde o vyplněná data.

Class Name	Table Name	Class Name	Table Name
Order	orders	LineItem	line_items
TaxAgency	tax_agencies	Person	people
Batch	batches	Datum	data
Diagnosis	diagnoses	Quantity	quantities

Obrázek 6.14: Konvence pojmenování tabulek v Ruby on Rails [17]

MVC - Controller (řadič)

Řadič, neboli controller je mezivrstvou mezi modelem a prezentační vrstvou (view). Controller přijímá události vyvolané uživatelem v prohlížeči, interaguje s modelem a plní prezentační vrstvu daty, které se mají zobrazit uživateli. [17] Každý controller je specializací třídy ApplicationController. Třída controlleru je pojemovaná následujícím způsobem: NazevcontrolleruController. Odpovídající soubor tomuto controlleru je pojmenován jako: nazevcontrolleru_controller.rb. Třída potom obsahuje definice metod, které mohou být buď pomocné nebo se jedná o metody, které se přímo volají při spuštění akce uživatelem. Takto vytvořený controller je pak jednoduše v prohlížeči zavolán jako:

`http://host:3000/nazevcontrolleru/akcecontrolleru`

, čímž se vykoná požadovaná akce v daném controlleru.

I v controlleru se očekává použití určitých konvecncí u pojmenování metod. Metody začínají malým písmenem. Pokud obsahuje název metody více slov, pak jsou odděleny podtržítkem. Názvy tříd začínají vždy velkým písmenem.

MVC - View (pohled, prezentační vrstva)

View, neboli prezentační vrstva je zodpovědná za generování uživatelského rozhraní. Jakmile jsou data zobrazna, práce view končí. [17]. View je spjato s controllerem, avšak controller nemusí mít žádné view nebo naopak hned několik. Jedno view odpovídá jedné akci v controlleru. Když je pak akce vyvolána, view se zobrazí v prohlížeči. Pro přehlednost jsou všechny view patřící k jednomu controlleru v jednom adresáři, který nese název daného controlleru. Mohou také existovat akce, které může uživatel vyvolat a není potřeba view. Může se jednat například o odstranění položky v html tabulce čili po vykonání akce dochází k načtení view původní akce.

Aby bylo možno do určité míry používat logiku i ve view, používá se v RoR šablonovací systém Embedded Ruby (ERb). Mohlo by se zdát, že je tímto porušen koncept MVC, ale aplikační logika je zde používaná velmi málo a jen tam, kde je opravdu nutná. Příkazy v ruby se v šabloně zapisují mezi znaky <% a % >. Pokud vznikne potřeba vypsat obsah ruby proměnné přímo do šablony, pak je tato proměnná zapsána mezi znaky <%= % >.

Routování

Routování v RoR představuje práci s URL aplikace. Po vytvoření projektu v RoR je vytvořen také soubor routes.rb, který se nachází ve složce config. V tomto souboru jsou implicitní nastavení routes (cest) a zde je může programátor předefinovat. Základní route je

map.root :controller =>„login“, :action =>„index“, která nastavuje root aplikace, tedy jaký controller a jaká akce v něm se má vykonat po spuštění aplikace. V tomto případě se jedná o controller s názvem login a akcí index. Další implicitní nastavení, které lze v souboru najít, jsou tyto routes: map.connect ':controller/:action/:id', map.connect ':controller/:action/:id.:format', které dovolují do url vložit id (parametr) a dále formát, např (xml). Protože je tato aplikace multijazyčná, byla použita ještě jedna route a to: map.connect ':controller/:action/:lang', která definuje takovou route, ve které je zobrazen vybraný jazyk uživatelem.

Objektově-relační mapování (ORM)

Objektově-relační mapování je použito takovým způsobem, že objekty objektově orientovaného programovacího jazyka jsou mapovány na relační databáze. Databázové objekty se tak jeví jako objekty programovacího jazyka. [2] V RoR mapují ORM knihovny databázové tabulky na třídy viz model v popisu MVC v kapitole 6.3.4. Jeden řádek tabulky představuje jeden objekt, jehož atributy jsou názvy sloupců tabulky. Takto lze číst hodnoty sloupců nebo je měnit. V RoR existuje vrstva, která řeší ORM. Jedná se o ActiveRecord. Třída, která má pak za úkol mapovat tabulku, je zapsaná tímto způsobem: class NazevTabulky <ActiveRecord::Base end.

Pokud se nevyužívá ORM, pak se dotazy zapisují jako řetězce, které se následně vykonají zavoláním patřičné funkce (nebo je řetězec přímo součástí funcke). V RoR je možno použít i tento způsob, je však vhodnější používat ORM, které zpřehledňuje zápis, má funkce navíc atd. Klasické SQL operace jako SELECT, INSERT, UPDATE a DELETE jsou v ActiveRecord definovány jako statické metody, kde SELECT je find, INSERT je create (nebo také new a save), UPDATE je update a DELETE je delete. Vložení nového záznamu do databáze může být provedeno dvěma způsoby. První možností je zmíněná funkce create, která vytvoří nový objekt představující řádek v databázové tabulce a poté ho uloží do databáze. Naproti tomu existuje druhá možnost, kde se pomocí metody new vytvoří nejdříve objekt a pak jej musí programátor sám uložit do databáze pomocí metody save. Tato druhá možnost je vhodná právě tam, kde se řeší validace formulářů viz kapitola 6.3.4 výše - popis modelu v architektuře MVC.

Díky vhodnému návrhu ORM v RoR je zajištěna ochrana před útoky zvanými sql injection. Pomocí sql injection se snaží útočník podvrhnout (obvykle formulářová) data a pozměnit tak sql dotaz. V případě úspěchu je možno nakonec i odstranit celou databázi. Pokud je v RoR dodržován správný zápis metod dotazujících do databáze, pak je ochrana před tímto typem útoku zajištěna. Příklad potenciálně nebezpečného dotazování: Course.find(:all, :conditions =>[„name = #params[:name]“]). Takto se přímo vkládá řetězec do dotazu. Příklad bezpečného dotazování: Course.find(:all, :conditions =>[„name = ?“, params[:name]])

Návrhové vzory

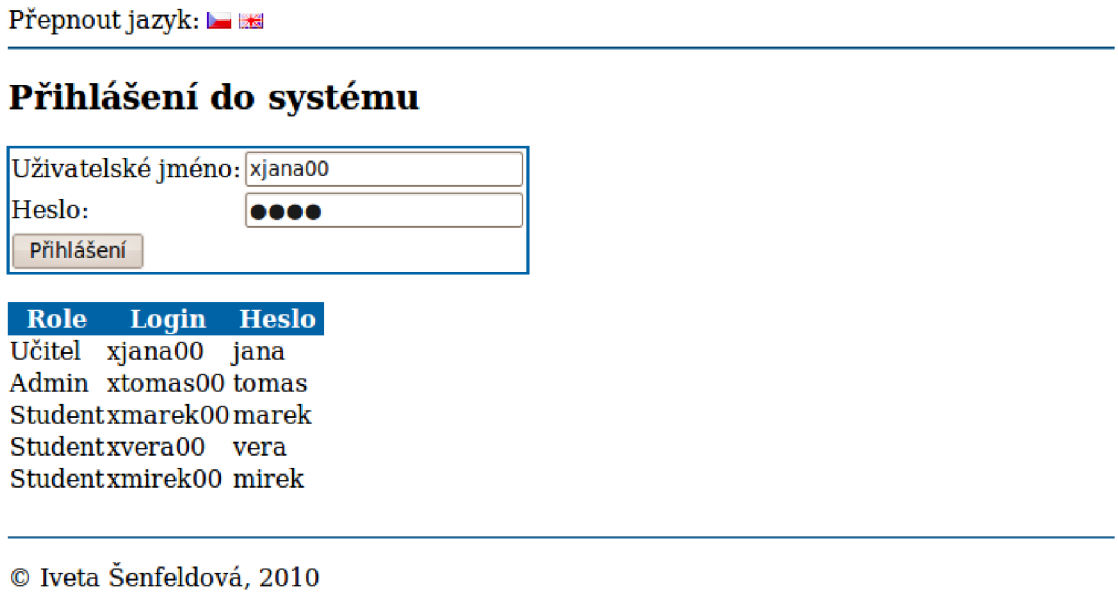
Framework Ruby on Rails obsahuje některé návrhové vzory. Jedním z nich je například návrhový vzor *Adaptér*, který souvisí s ORM. Při vykonávání dotazu do databáze se musí systém připojit k databázi. Databází je ale více druhů a proto musí existovat třída - adaptér - která načítá konfiguraci databáze z konfiguračního souboru a poté je schopna se na danou databázi připojit a vykonat dotaz. Čili úkolem adaptéru je přizpůsobit existující rozhraní objektu takovému rozhraní, které je potřeba. [13].

V modelu lze definovat tzv. callback funkce, které se zavolají při určité akci. S těmito callbacky lze provádět komplexní validace, mapovat hodnoty slopupců tak jak jsou ukládány nebo vybírány z databáze a dokonce lze jimi zabránit v provedení určitých operací. [17] Příklad takové callback funkce je: *before_destroy*, za kterou následuje název metody, která se má vykonat před tím, než se bude mazat z databáze. Toto chování odpovídá návrhovému vzoru *Observer* - *Pozorovatel*, který upozorňuje na změnu stavu objektu nebo na přítomnou událost.

Kapitola 7

Implementace aplikace

Implementace vychází z návrhu a dostupných možností, které poskytují vybrané nástroje. Aplikace se skládá ve své podstatě z pěti částí. První částí je přihlašování do systému, které je na obrázku 7.1 a ostatní čtyři představují oblasti pro dostupné role v systému.



Obrázek 7.1: Přihlášení do systému

Role administrátor má dva zásadní úkoly a to: správa uživatelů a správa kurzů. Administrátor jako jediný může měnit na požádání uživateli informace, ostatní role si své osobní informace mohou jen prohlížet. Administrátor dále přiřazuje učitele do kurzů.

Učitel se stará zejména o testový a výukový modul. Čili správa materiálů a správa testů. Dále však může zapisovat studenty do kurzu nebo je z nich odhlašovat a spravovat jejich bodya zejména pak student povyšovat a přiřazovat k nim studenty. Učitel je také moderátorem diskuzního fóra.

Student má ze všech rolí nejméně privilegií. Především jsou pro něj vytvářeny testy a vkládány soubory. Testy může spouštět a soubory stahovat na disk.

Vedoucí student je role, která je v podstatě specializací studenta. Oproti studentovi má možnost vytvářet testy a má svou část diskuzního fóra.

V této kapitole jsou popsány části aplikace, které jsou významné nebo řešeny způsobem, který je vhodné popsat.

7.1 ApplicationController

ApplicationController je controller, který je generalizací všech controllerů v aplikaci. V tomto controlleru se proto definují veškeré metody a nastavení, které pak budou využívat všechny ostatní controllery, jež jsou jeho specializací. Následující tři kapitoly popisují doplněnou logiku určenou všem controllerům.

7.1.1 Aktualizace aktivity

Aby bylo možno zjistit, kdo z uživatelů je právě přihlášen v systému, je nutno tuto informaci uchovávat v databázi. Pro tyto účely slouží sloupec s názvem *lastactivity* v tabulce *users*. Sloupec je typu *datetime*. Kdyby sloupec uchovával pouze aktuální čas bez datumu, pak by se mohlo stát, že pokud by se uživatel přihlásil například ve tři hodiny odpoledne a pak se týden nepřihlásil, každý den v tuto dobu by se zobrazoval jako přihlášený v systému. K tomu, aby bylo zjištěno, zda je uživatel v systému aktivní se tak musí aktualizovat jeho lastactivity při každé akci, kterou vykoná. Vzhledem k tomu, že se controller chatu dotazuje na aktuální data co pět sekund, pak je tak často aktualizována aktivita přihlášeného uživatele. Avšak test, zda je uživatel přihlášený, je vyhodnocen jako pravda, pokud ještě neuběhlo pět a více minut od posledního záznamu aktivity. Tuto funkčnost zajišťuje metoda s názvem *set_last_activity*.

7.1.2 Nastavení jazyka

Aplikace je přeložena v době odevzdání této práce do dvou jazyků. Uživatel má možnost mezi těmito jazyky přepínat. Změna jazyka se projevuje v proměnné *params[:lang]*. Aby tato změna byla trvalá, ukládá se do pole session a ze session se opět znovuplní params[:lang], neboť controllery pracují s touto proměnnou. Tuto poměrně jednoduchou funkčnost zajišťuje metoda s názvem *set_locale*.

7.1.3 Testování práv

Každá role v systému má vyhrazenou oblast, po které se může pohybovat. V aplikaci musí být zajištěno, aby se žádná z rolí nedostala do oblasti role jiné. Proto má každá role svou sadu controllerů založenou právě na roli a díky tomu lze jednoduše testovat, jaká role požaduje jakou stránku. Pokud by například chtěl student přistoupit na stránku *teachertests*, díky opatření v *ApplicationController* se mu vypíše hlášení, které mu sdělí, že nemá práva pro zobrazení této stránky. Metoda tedy porovnává roli přihlášeného uživatele a výskyt

role v požadovaném controlleru. Pokud tato podmínka neprojde, uživateli se stránka nezobrazí. V aplikaci však existují výjimky a těmi jsou například: ChatController, PersonalinfoController, které se nemusí kontrolovat, neboť pracují s uživatelovými daty uloženými v globálním poli session. Testování práv zajišťuje metoda *check_rights*.

7.2 Layouty

V Ruby on Rails se vytváří takzvané layouty, které slouží pro rozmístění prvků na stránce. Jedná se o klasické HTML stránky, kde je možno vkládat přídavnou logiku a případně JavaScript. Jejich použití tak výrazně snižuje opakování kódu, který by se tak musel psát v každé šabloně pro každou stránku. Layouty obsahují speciálně označené místo <%= yield %>, do kterého se bude vkládat obsah šablony, která koresponduje s požadovaným controllerem a akcí. Pro účel tohoto projektu byly vytvořeny tři layouty. Jeden je použit při zobrazování formuláře pro přihlášení do systému a druhý je použit v popup oknech a třetí je ve všech ostatních případech. Vzhledem k tomu, že název layoutu má odpovídat názvu controlleru, což nelze splnit, pokud více controllerům má odpovídat jeden layout, bylo použito předefinování tohoto implicitního nastavení. Předefinování je velmi prosté, stačí zápis v třídě controlleru layout „nazev_layoutu“.

Layout s názvem main, který slouží pro zobrazování veškerých stránek po přihlášení v aplikaci kromě popup oken, má následující strukturu:

- head - zde se v sekci title načítá aktuální titulek prohlížeče. Dále se zde přilinkovává soubor pro design stránek a javascriptová knihovna jQuery
- body - po sekci head následuje body, ve které se načítá header, který obsahuje souhrnné informace o uživateli, jako například jméno, příjmení, roli atd.
- menu - po headeru následuje sekce menu, ve které se načítají položky menu v závislosti na tom, jakou roli má přihlášený uživatel. Pokud má uživatel roli učitel, bude se načítat menu z konfiguračního souboru s názvem menuteacher.yml a následně se budou generovat odkazy na na stránky, kde název controlleru je složen z názvu role a položky v konfiguračním souboru. Příklad takového vygenerovaného odkazu může být: <a href=„/teachertests“>Testy . Tento odkaz vede na controller teachertests a akci index.
- contact list - po načtení menu je sekce pro načtení contact listu chatu
- obsah - předposlední sekcí je sekce, kde bude vkládán obsah načítaných šablon dle požadované stránky
- patička - poslední částí je patička informující o autorovi aplikace

7.3 Konfigurační soubory

Konfigurační soubory slouží pro jednoduché načtení a případné uložení (aktualizaci) dat. Jsou uloženy v adresáři config a jedná o se o soubory: fields.yml, languages.yml, menu s patřičnou rolí a roles.yml. YAML (YAML Ain't Markup Language) formát je podobný xml formátu. Soubor fields.yml uchovává obory, které mohou být přiřazeny studentům. Jednoduchým vložením nového oboru do tohoto souboru se přidá možnost studentovi přiřadit

další obor. Stejně tak lze manipulovat se souborem languages.yml. Pokud je přidán nový jazyk (zkratka), je automaticky načítán a připraven k použití. Pro kompletní funkčnost nového jazyka však musí být doplněny překlady viz níže v této sekci. Roles.yml obsahuje dostupné role v systému. Lze přidat nový záznam, avšak role je provázána s konfiguračním souborem menu + název_role a pokud by se chtěl do systému přihlásit uživatel s nově přidanou rolí, pak by musel existovat i soubor pro načtení menu a controllery, které by měly v názvu danou roli. V případě přidání položky do menu souboru pak musí být vytvořen i patřičný controller a šablona pro zpracování této položky. Z toho vyplývá, že přidávání rolí a položek do menu je úkol pro programátora.

Překlady jazyků se nachází v souborech pojmenovaných patřičnou zkratkou jazyka v adresáři config/locales. Po přidání nového jazyka v do souvoru languages.yml by pak musel být zde vytvořen nový soubor s danou zkratkou a doplněny překlady.

7.4 Ajaxový chat

Pro možnost synchronní komunikace mezi uživateli byl implementován ajaxový chat znázorněný na obrázku 7.2. Ajax (Asynchronous JavaScript and XML) mění obsah stránky (nebo její části) na základě stavu serveru tak, aniž by musela být celá stránka znovu načtena. Funkčnost chatu je rozdělena do více souborů. Pro přístup k chatu slouží contact list načtený v menu. Javascriptový kód je umístěn přímo v layoutu hlavní části aplikace, v sekci menu. Na tomto místě probíhá načítání contact listu chatu, to znamená načítání přihlášených (online) uživatelů. Za přihlášeného uživatele se považuje ten, jehož záznam „lastactivity“ v databázi nepřesáhl dobu pěti minut. Toto znovunačítání contact listu se dá přirovnat k pojmu „polling“, což znamená pravidelné (v určitém zvoleném časovém intervalu) se dotazování do databáze, který uživatel je zrovna přihlášený. Pro načtení uživatelů je opětovně volán controller Chat pro načtení dat, které jsou následně vloženy do JSON pole a v layoutu jsou pak vypsány loginy uživatelů, jež jsou zároveň odkazy. Tento odkaz vede na stránku pro chatování s uživatelem, kde je zobrazena hisotrie, pokud existuje. Z historie je vypsáno posledních deset záznamů.

Historii chatu, tedy její načítání řeší controller ChatHistory, který je taktéž opětovně volán a to co čtyři sekundy. Takto dochází k aktualizaci stránky a tedy možnosti komunikace mezi dvěma uživateli v reálném čase. Odeslání zprávy zpracovává controller ChatSendMessage, který pouze ukládá data do databáze. Ukládají se informace o tom, kdo zprávu odeslal, kdo je příjemce, samotná zpráva, informace o tom, že je zpráva nová, datum a čas. Po uložení zprávy je načtena controllerem ChatHistory a zobrazena na stránce. Pokud příjemce není v moment odeslání zprávy na stránce pro komunikaci s odesílatelem, pak se mu u odesílatele v menu, v contact listu zobrazí blikající vykřičník, který značí, že mu byla poslána zpráva od daného uživatele. Ikona je z menu odstraněna v moment, kdy příjemce vstoupí na stránku chatu a je mu zobrazena historie chatu s odesílatelem. V tomto okamžiku dochází k aktualizaci atributu nových zpráv „is_new“ na hodnotu false. Pro ukládání komunikace chatu mezi dvěma uživateli je v databázi tabulka s názvem chat_user_messages. K této tabulce je pak vytvořena třída (model) pro vykonávání dotazů. Tato třída obsahuje jednu statickou metodu, která zjišťuje přítomnost nových zpráv pro přihlášeného uživatele od ostatních přihlášených uživatelů. Údaj, zda je v databázi přítomna nová zpráva, je načítán do pole spolu s contact listem a pak je tento atribut testován při výpisů přihlášených uživatelů a na základě jeho hodnoty je pak zobrazena nebo nezobrazena upozorňující ikona.

Obrázek 7.2: Ajaxový chat

7.5 Popup okna

Popup okna v aplikaci slouží pro rozšířený výběr objektů, kdy nám nepostačují klasické formulářové prvky. V aplikaci jsou popup okna použity na dvou místech. Jednak v administraci a jednak v sekci pro roli učitel. V administraci se jedná o přidávání učitele do kurzu. Zobrazení nového okna vyvolá odkaz pro přidání nebo odebrání učitele z kurzu. Je zobrazen seznam učitelů s checkboxy, kde se jednoduše vybere nebo odebere požadovaný učitel. Po provedení této akce se vykoná JavaScript, který obnoví původní stránku a okno se zavře. Obnovením původní stránky se automaticky načtou vybraní učitelé (nebo se naopak odstraní z výpisu odebraní učitelé). Stejně tak funguje popup okno v roli učitele, kde se přidávají nebo odebírají studenti z kurzu. Zde však není potřeba obnovování původní stránky, neboť tyto data se načítají až v detailu studenta.

Popup okna mají svůj vlastní layout.

7.6 Práce se soubory

Práce se soubory probíhá na více místech. Učitel má možnost vkládat materiály ke kurzu, ale také k lekci a dále je spravovat. Takto vložený soubor je uložen na disk do adresáře

public/data a do databáze se jen ukládají dodatečné informace jako název souboru a jeho *content-type*. Soubory jsou na disku pojmenované dle id sloupce v databázi. Takto lze zajistit případný konflikt v názvech souborů, které by se mohly jmenovat stejně, a přitom by měly jiný obsah. Vložený materiál se pak zobrazuje studentům v daném kurzu nebo lekci. Název souboru je odkazem a ten vede na stažení souboru. Tuto funkcionalitu zajišťuje controller *DownloadanddeleteController*, ve kterém jsou dvě metody. Jedna metoda pro stažení souboru a druhá pro jeho odstranění z disku.

Každý uživatel má možnost mít v systému svou fotografii. Nahrávání probíhá při vytváření uživatele nebo při jeho editaci. Pokud je fotografie přiložen, pak je rovněž uložena na disk. Avšak fotografie uživatelů mají vlastní adresář a to *public/data/users*. Fotografie se ukládají s názvem loginu uživatele, neboť je v systému unikátní.

7.7 Povyšování studentů

Povyšování studentů probíhá v detailu kurzu v roli učitele, který je zobrazen na obrázku 7.3. V detailu kurzu jsou načteni všichni zapsaní studenti. Každý student má u svého jména ikonu prázdné hvězdy. Odkaz této ikony vede na povýšení studenta. V takovém případě je studentovi převedena role student na vedoucí student a je zobrazen nad všemi studenty v daném seznamu. Ikona u vedoucího studenta je vyplněna žlutou barvou a její odkaz vede na změnu role vedoucího studenta na studenta. Jakmile je dostupný alespoň jeden vedoucí student, je mu možno přiřadit skupinu studentů, kterým bude následně pomáhat. Toto přiřazení probíhá výběrem studentů, výběrem vedoucího studenta a potvrzení.

Při povýšení studenta zároveň dochází k vytvoření jeho sekce na diskuzním fóru. Sekce je pojmenovaná loginem studenta. Pokud by byl student povýšen i v jiném kurzu, pak má stejný prostor pro diskuzi v obou kurzech. S odstraněním vedoucího studenta ze všech kurzů je odstraněn i jeho prostor v diskuzním fóru. Vzhledem k tomu, že je login studenta unikátním, pak nemusí být v tabulce *forum_parts* (obsahující části fóra) další informace, která by se vázala na studenta.

Při povýšení studenta je zapsán záznam do tabulky *lead_students* a při přihlášování pak musí být proveden test, zda se přihlašuje student a zda není v některém z kurzu vedoucím studentem. Pokud ano, je mu automaticky načteno patřičné menu z konfiguračního souboru. Pokud je student vedoucím studentem v jednom kurzu, avšak zapsaný ve více kurzech, privilegia vytváření testů apod. se váží pouze na kurz, ve kterém je veden jako vedoucí.

7.8 Testy

Testy jsou důležitou částí této aplikace. Mohou být vytvářeny učitelem nebo vedoucím studentem. Testy vedoucího studenta mají pouze charakter procvičení si otázek vytvořených vedoucím studentem, testy vytvořené učitelem mají mnohem větší váhu a student za ně získává bodové hodnocení. Vytvořené testy se studentům zobrazují v detailu kurzu nebo v detailu lekce, ke které náleží. Testy mohou obsahovat více lekcí a pak je takový test zobrazen jen u té nejvyšší lekce, neboť u nižších lekcí nemají studenti potřebné znalosti. Učitel má možnost testy plně spravovat, pokud byl však test alespoň jednou absolvován, je uzamčem pro editaci.

7.8.1 Vytváření testu

Při vytváření testu je na učiteli, aby zadal potřebné parametry. Musí být zadán alespoň jeden název testu ve vybraném jazyce, kurz, ke kterému se bude test vázat, lekce kurzu (jedna nebo více), počet otázek, které se budou vybírat z databáze, typy otázek a parametr „skrytý“, který říká, zda se bude test studentům zobrazovat nebo ne. Popis testu nemusí být zadán, ale při spouštění testu studentem mu pak nebudou o testu dostupné žádné další informace. Typy otázek mohou být tři a to: 1, což znamená „právě jedna z N“, M znamená „M z N“ a fulltext znamená, že student zadá test, který bude vyhodnocen později učitelem.

Po zadání potřebných parametrů a jejich validaci je učiteli zobrazena stránka, na které jsou náhodně vybrány testové otázky z databáze odpovídající zadaným parametrům. Nejdříve se do pole načtou veškeré otázky odpovídající zadaným parametrům a následně se ve smyčce vybírají náhodně otázky tak dlouho, dokud není vybrán zadaný počet otázek. Je ošetřeno, že se v testu neobjeví dvě stejné otázky. Pokud učitel zadal více otázek než je dostupno, pak je o tom informován a musí parametry testu změnit. V opačném případě jsou otázky načteny a připraveny k bodovému ohodnocení. Je nutno zadat kladné i záporné hodnocení, které je implicitně nastaveno na nula. Pokud není učitel spokojen s vybranými otázkami, má možnost odeslat požadavek na jejich znovuvybrání. Jakmile je spokojen, odešle požadavek pro uložení testových otázek k danému testu.

7.8.2 Spouštění testu

Spouštění testů probíhá v rolích student a vedoucí student. Studenti se na spuštění testu dostanou přes odkaz v názvu testu. Následně je zobrazena stránka, která zobrazuje popis testu a odkaz vedoucí na spuštění testu. Po spuštění testu jsou načteny otázky a odpovědi nebo místo odpovědí pouze formulářové pole pro zadání fulltextové odpovědi. V průběhu vyplňování testu má student možnost uložení stavu testu do databáze pro případ pádu webového prohlížeče nebo jiných nepříznivých okolností. Takto uložený test má nastavený stav *inprocess* a dokud nezadá vyhodnocení testu, stav se nemění.

7.8.3 Vyhodnocení testu

Při vyhodnocení testu je pozměněn stav testu, pokud byl rozpracovaný test uložen nebo je vytvořen nový záznam o absolvovaném testu v databázi. Pokud test neobsahoval žádné fulltextové otázky, pak je okamžitě po odeslání formuláře vyhodnocen. Vyhodnocení se promítne studentovi tak, že je mu zobrazen celkový získaný počet bodů za test a dále jsou zelenou barvou znázorněny odpovědi, které jsou správně. Zobrazení správných odpovědí je založeno na změně css stylu. Po vyhodnocení testu jej student nemá možnost opakovat a už se nedostane na jeho spuštění.

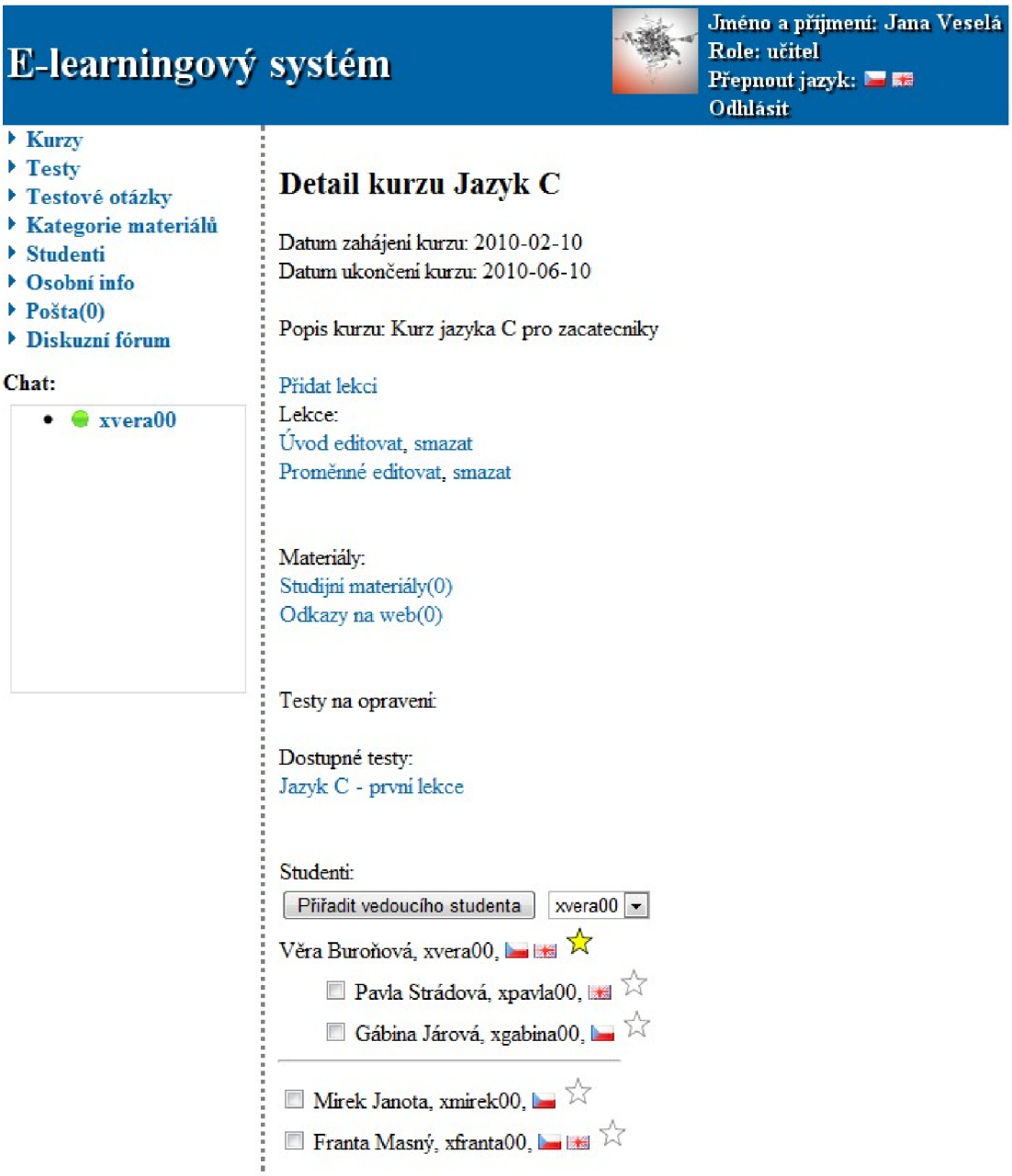
7.9 Získávání statistik a jejich zobrazení

Statistická data se zpracovávají v detailu testu a v kurzu po jeho ukončení. Pro zobrazení statistik se nejdříve musí data zpracovat do řetězců. Ty se pak použijí jako součást html kódu (url) obrázku, který je stahován z internetu - Google Chart Tools. Příklad url pro vygenerování statistiky: http://chart.apis.google.com/chart?
cht=p3& tento paramter udává typ grafu - v tomto případě jde o koláčový typ
chs=250x100& udává velikost grafu
chd=t:60,40& statistické hodnoty

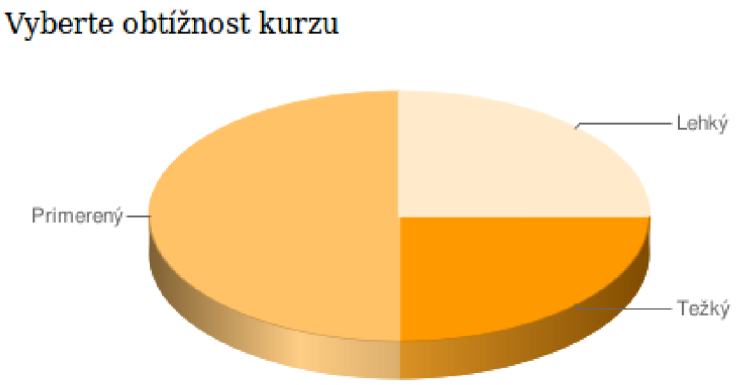
chl=Hello—World hodnoty v pořadí, v jakém přísluší uvedeným statistickým hodnotám [6] Parametry chd a chl jsou získávány z databáze a upravovány do řetězců. V případě detailu testu jsou načítány možné bodové zisky a četnost získaných jednotlivých bodových zisků. V případě statistik ohodnocení kurzu studenty jsou v databázi připraveny tři otázky s odpověďmi ve dvou jazycích pro prezentaci. Tyto statistiky se generují na základě četnosti odpovědí k jednotlivým otázkam. Toto ilustruje obrázek 7.4.

7.10 Shrnutí

Mimo výše uvedené implementační kroky bylo potřeba vyřešit mnoho dalších více či méně složitých postupů, jak dodat finální funkční systém. Některé vlastnosti nebylo možno v rámci rozsahu diplomové práce implementovat. Tato rozšíření nejsou potřebná k základní funkcionalitě systému, ale mohou razantně navýšit schopnosti systému. O možných rošířeních pojednává další kapitola.



Obrázek 7.3: Detail kurzu



Obrázek 7.4: Statistiky po ohodnocení kurzu studenty

Kapitola 8

Možnosti rozšíření

Implementace této aplikace je spíše do šířky než do hloubky. To znamená, že je v systému více funkcí, ale na méně detailní úrovní. Z toho plyne, že aplikace je do budoucna snadno rozšiřitelná o případné detaily, ale i o nové funkce.

8.1 Server

Aplikace je nyní vytvořena jako jeden celek napojený na databázi. To by mohla být v budoucnu nevýhoda, pokud by se na systém měla připojit jiná služba. Bylo by proto vhodné rozdělit aplikaci na dvě části. Jedna část, klient, by řešila především interakci s uživatelem a druhá část, server, by přijímala požadavky od klienta, zpracovávala je a odesílala data zpět klientovi. Z toho plyne, že databáze by byla až za částí server, viz obrázek 8.1.

Obrázek 8.1: Návrh aplikace se serverovou částí

Server by se choval jako webová služba, abychom zajistili standardizované rozhraní. Takto navržený server by pak implementoval návrhový vzor *Fasáda*[13], jež by se skládal z jedné hlavní třídy, která by volala metody dalších specifických tříd. Specifické třídy by v tomto případě představovaly jednotlivé modely, čili třídy představující jednotlivé databázové tabulky a metody, definované v modelu, by pracovaly s daty z databáze. Server by byl dalším projektem frameworku Ruby on Rails, avšak oproti clientovi by nebyla potřeba žádných controllerů a viewů. Možnosti formátů dat zasílaných serverem je více. Nejčastěji se však používá XML.

S pomocí serveru by bylo možno připojit například informační systém pro správu bodů studentů - WIS. Aby byla data ve WISu aktualizovaná, musel by se tento systém dotazovat serverové části na aktuální data. Pak by se studentům po vykonání testu zobrazily body ihned v kurzu ve WISu a učitel by je nemusel zadávat ručně, čímž by odpadla manuální práce a snížila by se časová náročnost při správě bodových aktivit studentů.

Framework Ruby on Rails je na tuto možnost připraven. Existuje třída ActiveResource, která zajišťuje rozhraní, na které se mohou připojovat aplikace třetích stran.

8.2 Testy

Důležitou součástí e-learningového systému je testový modul. Tento modul by bylo možno vhodně v budoucnu rozšířit a to zejména o další parametry. V současné podobě má student možnost absolvovat test pouze jednou a poté je pro něj test uzamčen. V rámci procvičování by však mohl učitel vytvářet testy takové, které by bylo možno studenty opakovaně spouštět. Dalším možným parametrem, který se nabízí, je časové ohraničení testu. Před spuštěním testu by byl student informován o tom, zda je test časově omezen a případně jak a po jeho zahájení by započal odpočet nastaveného času.
Při vytváření testu se vybírají náhodně otázky z databáze, které odpovídají parametrům testu. Tento seznam otázek je následně zobrazen na stránce, na které má možnost učitel otázky ohodnotit a uložit nebo vybrat nové. Zde by bylo možno práci s otázkami rozšířit do flexibilnější podoby. Učitel by mohl například vkládat manuálně otázky ze seznamu, různě je prohazovat a podobně.

8.3 Povyšování studentů

povysovani studentu do vice vrstev Současná implementace dovoluje povýšit studenta o jednu úroveň. Rozšíření systému by mohlo nabízet možnost povyšování studentů do více vrstev tak, že vedoucí student by mohl mít nadřazeného vedoucího studenta. V takovém případě by byly více viditelné (nejen znalostní) rozdíly mezi jednotlivými vedoucími studenty. Tato možnost by pak vyžadovala lepší grafické znázornění aktuální hierarchie.

8.4 Chat

Komunikace přes ajaxový chat probíhá v této aplikaci mezi dvěma uživateli. V budoucnu by mohla být komunikace rozšířena na takovou úroveň, že by mohly existovat kanály, pojmenované například jako dostupné kurzy v systému, a zde by diskutovalo více uživatelů najednou. Uživatelé by případně mohli vytvářet nové kanály.

Kapitola 9

Testování a ladění

Důležitou fází v modelu životního cyklu software je také testování a ladění aplikace. Každá nově naimplementovaná část by měla být otestována na svou správnou funkčnost, a i když obvykle je, může se stát, že bude v budoucnu odhalena chyba, která je sémantického charakteru. Těmto chybám se snaží předejít správný návrh aplikace, avšak nelze vše na začátku simulovat nebo projít dopodrobna a nejen proto je určeno testování. Z obrázku 6.2 v kapitole 6.1 je patrné, že testování probíhá po celou dobu implementace. Je to proto, aby se zapříčinilo zanášení chyb do nových funkčnich celků, aby se otestovala funkčnost podpůrných prostředků a další.

U rozsáhlých projektů není možné pouze v jedné iteraci testování nalést všechny chyby a je obvyklé, že již nabízené programy obsahují chyby. Pro tyto účely existují programy jako například Bugzilla nebo Mantis, do kterých se zapisují nalezené chyby, které jsou následně přiřazovány prográmtorům, jenž mají za úkol je opravit. Bugzilla ve firmě obvykle slouží pro tým testerů, bez kterých se softwarová firma neobejde, avšak možnost zadávání chyb do programu mají u některých produktů možnost i uživatele (jedná se například o open source programy).

Testování může být do určité míry také automatizováno. Za automatizaci mohou být považovány skripty, které jsou spouštěny a vyhodnocovány uživatelem nebo přímo program, který s testerem prochází jednotlivé funkční bloky a očekává zápis výsledku jednotlivých testů (Unit Testy).
Program, který pracuje s daty a je jejich potřeba při testování, mají obvyklé testovací databáze. Obecně celé testování programu probíhá na testovacích serverech a verze, se kterou komunikují uživatelé (zákazníci) je provozována na jiném serveru.

Jakmile proběhne testování určitého celku programu, za který je zodpovědný určitý programátor a jsou nalezeny chyby, je mu tento fakt ohlášen a přichází fáze oprav. Jakmile jsou opravy hotovy, přichází na řadu další testování a tento cyklus se opakuje, dokud není funkčnost dle požadavků.

9.1 Shrnutí

Při testování této práce testerem bylo zjištěno, že ne všechny prvky jsou intuitivně rozmístěny. Na základě těchto zkušeností byly provedeny úpravy, které zvýšily použitelnost

webového rozhraní. Některé uživatelské nedokonalosti zůstaly zachovány, protože je nebylo možné v reálném časovém horizontu opravit. Tyto opravy by vyžadovaly zásah do základu aplikace, a v budoucnu by byly řešeny novou verzí po rozsáhlem refaktoringu.

Kvalitní uživatelské rozhraní by bylo možné vytvořit po delších provozních zkušenostech v postupných iteracích, nebo na základě zkušeností experta pro návrh uživatelského rozhraní.

Kapitola 10

Závěr

Tento projekt rozšiřuje možnosti klasických e-learningových systémů novým pohledem na proces studia, který jsem navrhla a implementovala v této práci. Proces studia novou metodou značně ulehčuje zde navrhnutý e-learningový systém, který řídí výměnu informací mezi studenty, jejich hodnocení a ostatní aspekty studia. Výsledkem je také užší spolupráce mezi studenty. Vyučující benefitují hlavně snížením časové náročnosti vyhodnocování testů a nižší potřebě konzultací ze strany studentů - studenti se obracejí s dotazy primárně na vedoucí studenty své skupiny.

Potenciál e-learningového systému tkví hlavně v propojení s hlavním fakultním informačním systémem. Vzájemná symbióza obou systémů by v budoucnu mohla přinést zjednodušení studia jak pro dálkové studenty, tak pro studenty denního programu.

Literatura

[1] Barešová, A.: *E-learning ve vzdělávánní dospělých*. VOX, 2003, iSBN 80-86324-27-3.

[2] Douglas K. Barry: Object-relational mapping. `http://www.service-architecture.com/object-relational-mapping/`, 2010 [cit. 2010-05-13].

[3] Dušan Janovský: Jak psát web. `http://www.jakpsatweb.cz/pouzitelnost.html`, 2009.

[4] Dušan Janovský: Jak psát web. `http://www.jakpsatweb.cz/javascript/javascript-uvod.html`, 2010 [cit. 2010-05-13].

[5] eFront prezentace: eFrontPresentation2008 English. `http://www.scribd.com/doc/3166070/eFrontPresentation2008-English`, 2009.

[6] Google: Google Chart Tools. `http://code.google.com/intl/cs/apis/chart/docs/making_charts.html`, 2010 [cit. 2010-05-15].

[7] Kolektiv autorů: JSON. `http://www.json.org/`, 2010 [cit. 2010-05-13].

[8] Marc-André Cournoyer: Thin. `http://code.macournoyer.com/thin/`, 2010 [cit. 2010-05-13].

[9] McConnell, S.: *Dokonalý kód*. Computer press, 2006, iSBN 80-251-0849-X.

[10] Moodle: Demo Moodle. `http://demo.moodle.cz/help.php?module=quiz&file=multianswer.html`, 2009.

[11] Moodle: Moodle. `http://docs.moodle.org/cs/Co_je_Moodle`, 2009.

[12] Netcraft Ltd: April 2010 Web Server Survey. `http://news.netcraft.com/archives/2010/04/15/april_2010_web_server_survey.html`, 2010.

[13] Pecinovský, R.: *Návrhové vzory*. Computer press, 2007, iSBN 978-80-251-1582-4.

[14] Ponkrác, M.: *PHP a MySQL bez předchozích znalostí*. Computer press, 2007, iSBN 978-80-251-1758-3.

[15] Pošmura, V.: *Apache - Příručka správce www serveru*. Computer press, 2002, iSBN 80-7226-696-9.

[16] Sak, P.: *Člověk a vzdělání v informační společnosti*. Portál, 2007, iSBN 978-80-7367-230-0.

[17] Sam Ruby, D. H., Dave Thomas: *Agile Web Development with Rails*. The Pragmatic Programmers LLC, 2009, iSBN-13: 978-1-9343561-6-6.

[18] Schneider, R. D.: *MySQL - Oficiální průvodce tvorbou, správou a laděním databází*. Grada Publishing, 2006, iSBN 80-247-1516-3.

[19] Wikipedia: Wikipedia - LCMS. `http://cs.wikipedia.org/wiki/Lcms`, 2009.

[20] Wikipedia: Wikipedia - LMS. `http://cs.wikipedia.org/wiki/LMS`, 2009.

[21] Řepa, V.: *Podnikové procesy - Procesní řízení a modelování*. Grada Publishing, 2007, iSBN 978-80-247-2252-8.

Dodatek A

Ukázka testových otázek pro předmět jazyk C

Výběr odpovědi 1 z N

Který operátor se používá pro dělení se zbytkem?
a) /
b) MOD
c) %
Správná odpověď c), +2 body

Výběr odpovědí 1 až N

Vyberte, které datové typy pracují s celými čísly
a) float
b) int
c) char
d) unsigned int
Správná odpověď b), d), +3 body

Písemná odpověď

Zapište cyklus for vypisující na standardní výstup čísla 1 až 10
- textové pole pro písemnou odpověď, která bude hodnocena učitelem
Správná odpověď +10 bodů

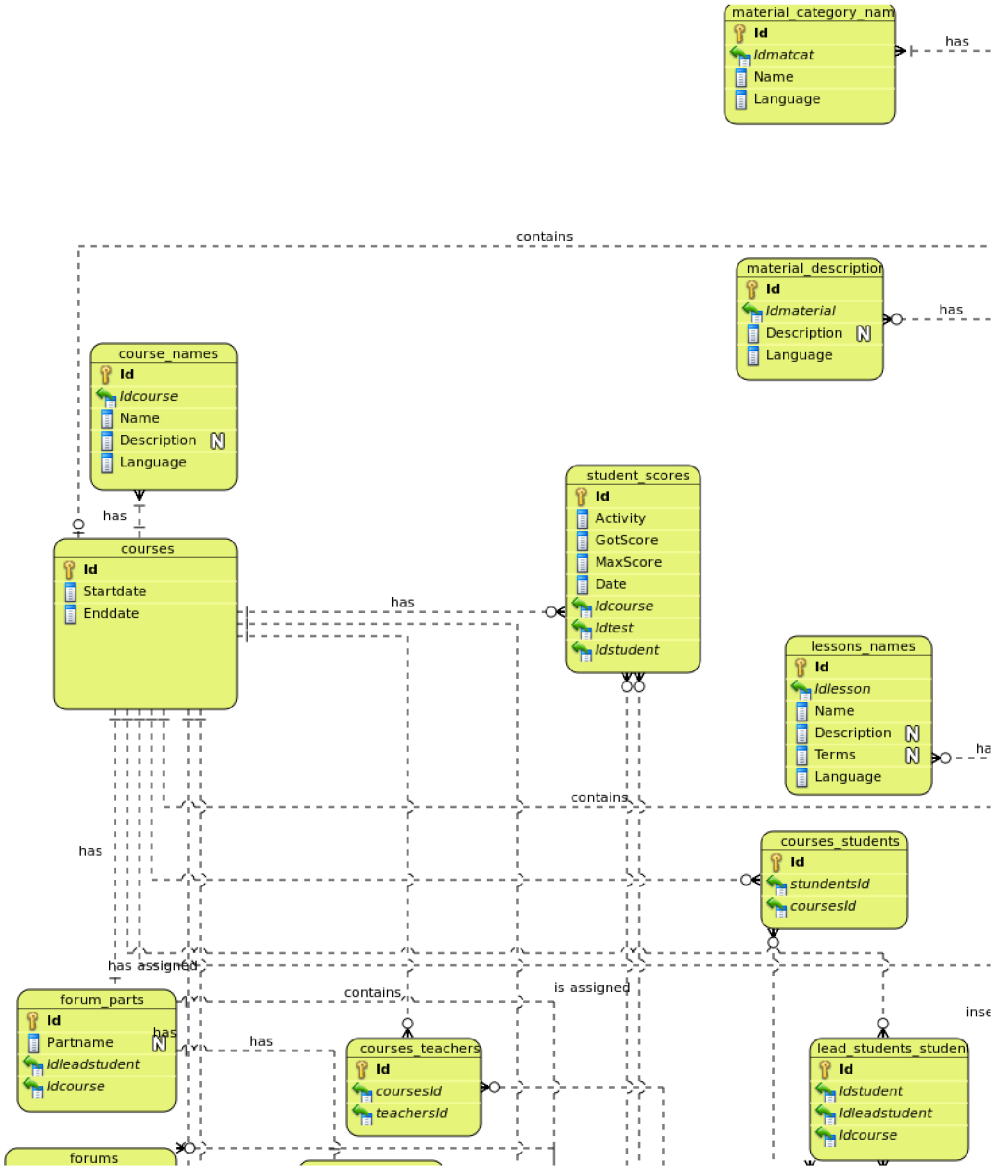
Doplňovací úloha
- připravený text (nebo kód) s prázdnými textovými poli pro doplnění správných hodnot.
Např. :

for (–doplnte–; i=5; i++) {

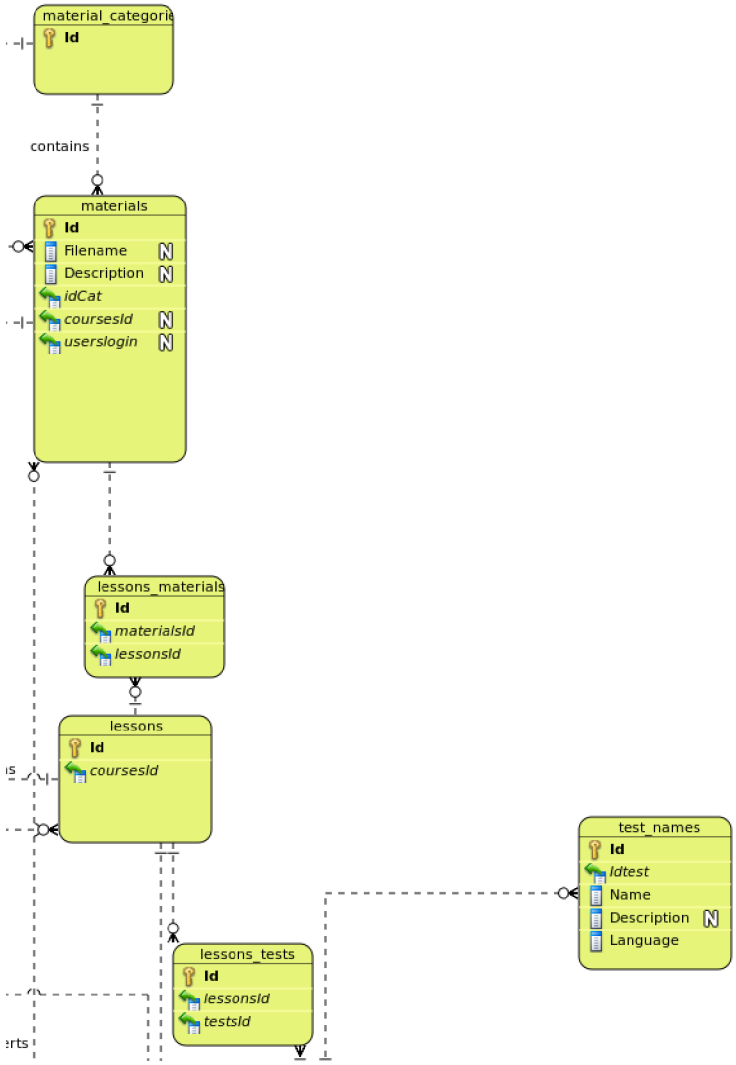
}
Správná odpověď +10 bodů

Dodatek B

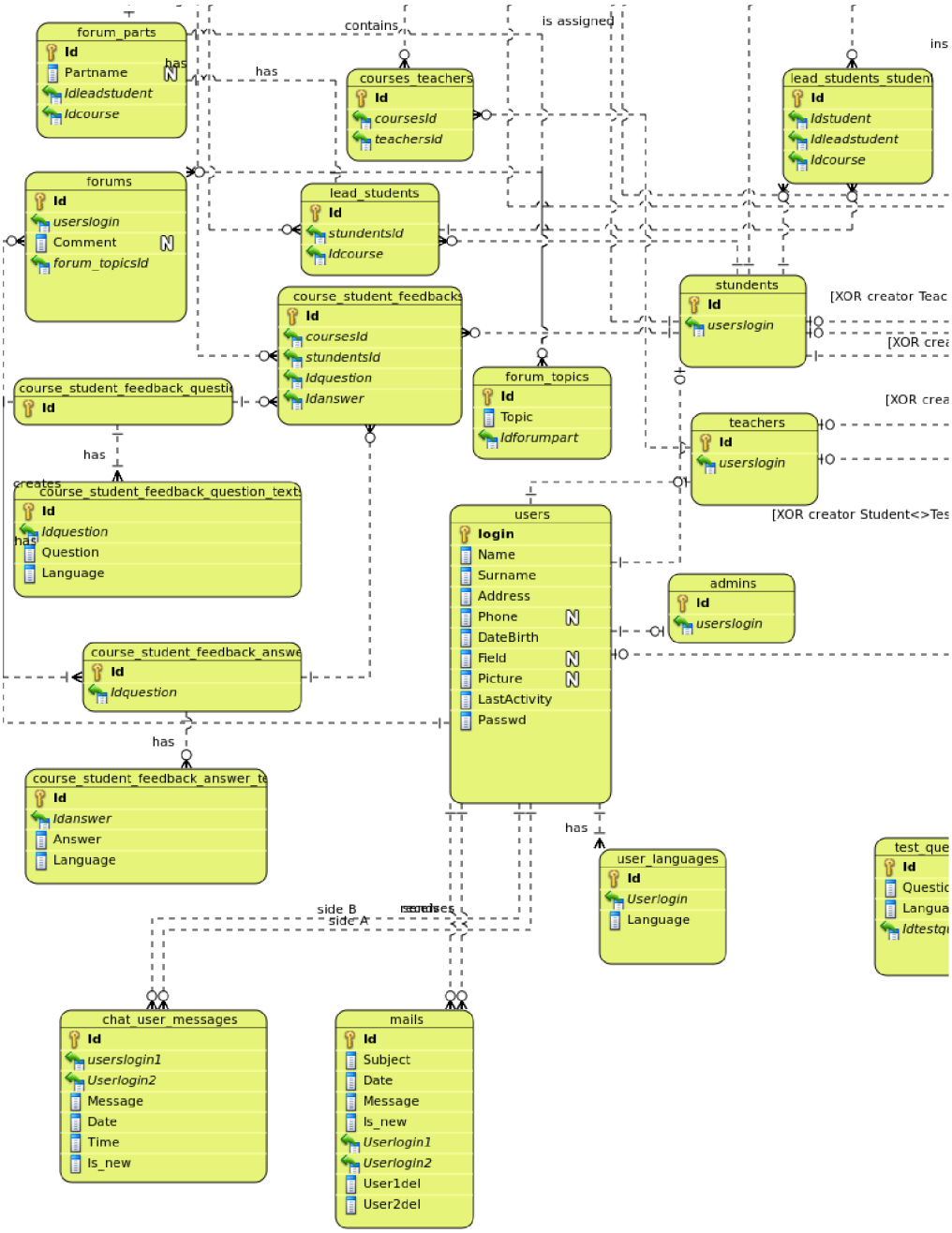
Návrh databáze



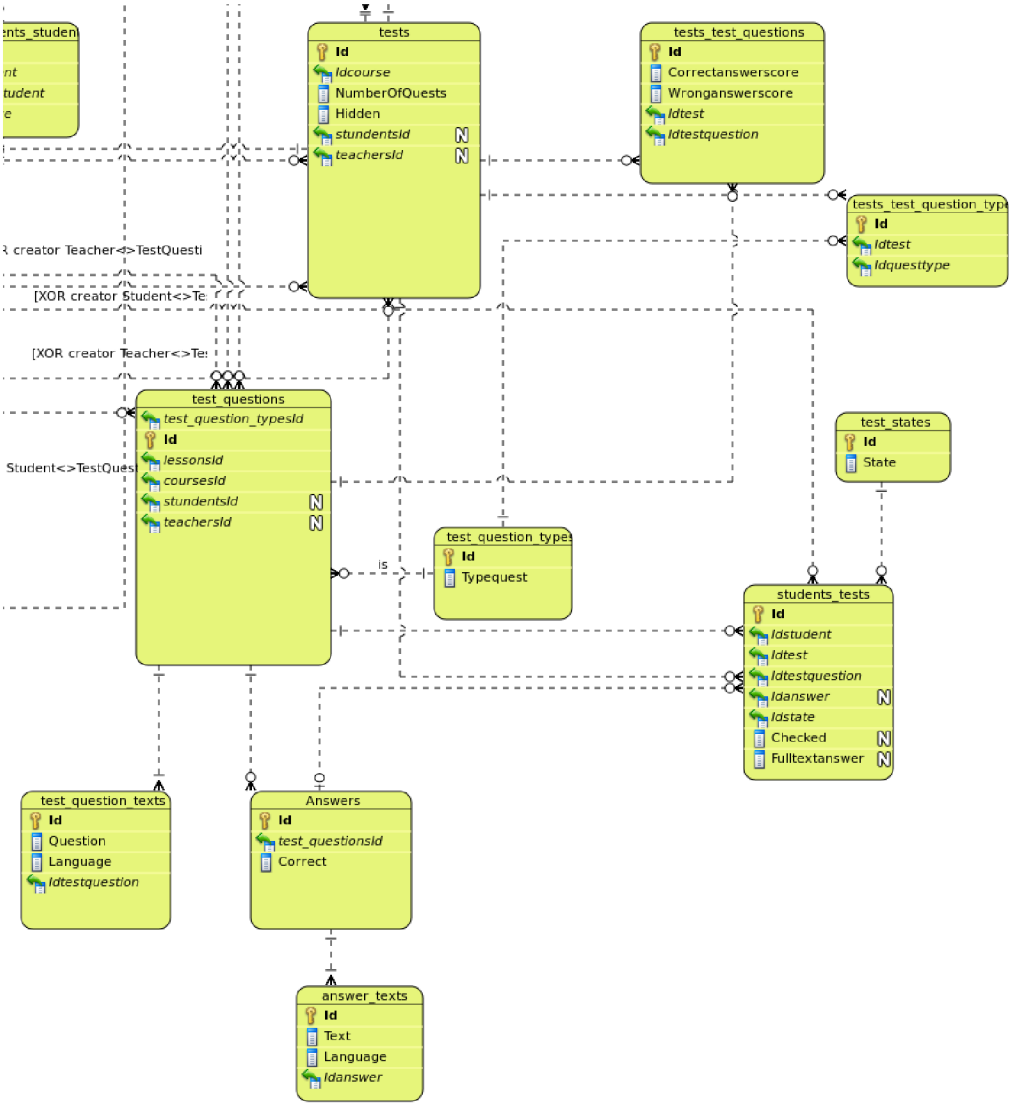
Obrázek B.1: Návrh databáze - část první



Obrázek B.2: Návrh databáze - část druhá



Obrázek B.3: Návrh databáze - část třetí



Obrázek B.4: Návrh databáze - část čtvrtá

Dodatek C

Use-case diagram

Obrázek C.1: Use-case diagram

C.1 Specifikace use-case diagramu

ID:	1
Název:	Hodnocení kurzu
Vytvořeno:	Iveta Šenfeldová
Popis:	Zhodnocení absolvovaného kurzu
Primární aktéři:	Student, Vedoucí student
Sekundární aktéři:	
Předpoklady:	Kurz je ukončen
Následné podmínky:	Odpovědi jsou zaznamenány v databázi
Akce pro spuštění:	Student vybere „Ohodnotit kurz“
Hlavní tok:	1.Systém zobrazí prázdný dotazník ke kurzu 2.Student vybírá odpovědi na otázky z nabízených možností a/nebo doplní textovou poznámku 3.Student vybere „Odeslat hodnocení“ 4.Návrat na předešlou obrazovku
Alternativní toky:	
Výjimky:	Selhání operace, Selhání systému
Frekvence:	Zřídka

ID:	1.E.1
Název:	Hodnocení kurzu - selhání operace
Vytvořeno:	Iveta Šenfeldová
Popis:	Případ použití není dokončen
Primární aktéři:	Student, Vedoucí student
Sekundární aktéři:	
Předpoklady:	1.Systém nedokončil případ použití 2.Systém se neukončil
Následné podmínky:	Data z dotazníku nebyla uložena do databáze
Akce pro spuštění:	Selhání v průběhu vykonávání hlavního toku
Tok:	1.Systém informuje studenta o chybě v průběhu vykonávání 2.Systém se navrátí na původní místo
Frekvence:	Zřídka

ID:	1.E.2
Název:	Hodnocení kurzu - selhání systému
Vytvořeno:	Iveta Šenfeldová
Popis:	Systém je nečekaně nedostupný
Primární aktéři:	Student, Vedoucí student
Sekundární aktéři:	
Předpoklady:	1.Výpadek proudu 2.Útok na systém 3.a jiné
Následné podmínky:	Hodnocení nebylo uloženo do databáze
Akce pro spuštění:	Selhání systému v průběhu vykonávání hlavního toku
Tok:	1.Systém informuje studenta o selhání
Frekvence:	Zřídka

ID:	2
Název:	Přidání/odebrání materiálů
Vytvořeno:	Iveta Šenfeldová
Popis:	Přidání nebo odebrání materiálu do kategorie
Primární aktéři:	Učitel
Sekundární aktéři:	
Předpoklady:	Žádné
Následné podmínky:	Vložený materiál je uložen na disk a informace jsou uloženy do databáze a je dostupný studentům, odstraněný materiál je odstraněn z disku i databáze
Akce pro spuštění:	Učitel vybere „Přidat“ nebo „Odstranit“ v dané kategorii
Hlavní tok:	1.Učitel vybere „Přidat“ nebo „Odstranit“ materiál 2.Při přidávání je zobrazen formulář pro výběr souboru, při odstraňování je zapotřebí pouze odeslat požadavek na odstranění přes odkaz
Alternativní toky:	
Výjimky:	Selhání operace, Selhání systému
Frekvence:	Zřídka

ID:	2.E.1
Název:	Přidání/odebrání materiálů - selhání operace
Vytvořeno:	Iveta Šenfeldová
Popis:	Případ použití není dokončen
Primární aktéři:	Učitel
Sekundární aktéři:	
Předpoklady:	1.Systém nedokončil případ použití 2.Systém se neukončil
Následné podmínky:	Materiál nebyl přidán do databáze nebo z ní nebyl odstraněn
Akce pro spuštění:	Selhání v průběhu vykonávání hlavního toku
Tok:	1.Systém informuje vedoucího studenta o chybě v průběhu vykonávání 2.Systém se navrátí na původní místo
Frekvence:	Zřídka

ID:	2.E.2
Název:	Přidání/odebrání materiálů - selhání systému
Vytvořeno:	Iveta Šenfeldová
Popis:	Systém je nečekaně nedostupný
Primární aktéři:	Učitel
Sekundární aktéři:	
Předpoklady:	1.Výpadek proudu 2.Útok na systém 3.a jiné
Následné podmínky:	Přidávaný materiál nebyl uložen do databáze a/nebo na disk, odstraňovaný materiál nebyl odstraněn z databáze a/nebo z disku
Akce pro spuštění:	Selhání systému v průběhu vykonávání hlavního toku
Tok:	1.Systém informuje vedoucího studenta o selhání
Frekvence:	Zřídka

ID:	3
Název:	Prohlížení zapsaných kurzů
Vytvořeno:	Iveta Šenfeldová
Popis:	Zobrazení informací souvisejících se zapsanými kurzy
Primární aktéři:	Student, Vedoucí student
Sekundární aktéři:	
Předpoklady:	Student je zapsán alespoň do jednoho kurzu
Následné podmínky:	Student má zobrazené informace o kurzu a má možnost prohlížet lekce kurzu, stahovat materiály a spouštět testy
Akce pro spuštění:	Student/Vedoucí student vybere název kurzu, který chce prohlížet
Hlavní tok:	1.Student/Vedoucí student vybere název kurzu 2.Je zobrazena stránka kurzu s veškerými dostupnými informacemi 3.Student/Vedousí student může dále pokračovat případy použití: Prohlížení lekcí, Stahování materiálů, Spouštění testů
Alternativní toky:	
Výjimky:	Selhání operace, Selhání systému
Frekvence:	Často

ID:	3.1
Název:	Prohlížení zapsaných kurzů - Stahování materiálů
Vytvořeno:	Iveta Šenfeldová
Popis:	Student/Vedoucí student má možnost prohlížet a zejména stahovat dostupné materiály kurzu
Primární aktéři:	Student, Vedoucí student
Sekundární aktéři:	
Předpoklady:	Ke kurzu byl vložen alespoň jeden materiál
Následné podmínky:	Student/Vedoucí student má zobrazené dostupné materiály nebo dále uložené na disku
Akce pro spuštění:	Student/Vedoucí student vybere kategorii materiálu, ve které se nachází soubory (odkazy)
Hlavní tok:	1.Student/Vedoucí student vybere kategorii materiálů, která uvádí neprázdný stav 2.Zobrazí se seznam dostupných materiálů 3.Pokud se jedná o elektronické texty či jiné soubory, může si je student uložit na svůj disk
Alternativní toky:	
Výjimky:	Selhání operace, Selhání systému
Frekvence:	Často

ID:	3.2
Název:	Prohlížení zapsaných kurzů - Prohlížení lekcí
Vytvořeno:	Iveta Šenfeldová
Popis:	Student/Vedoucí student má možnost prohlížet detaily lekcí
Primární aktéři:	Student, Vedoucí student
Sekundární aktéři:	
Předpoklady:	V kurzu je dostupná alespoň jedna lekce
Následné podmínky:	Student/Vedoucí student má zobrazený detail vybrané lekce
Akce pro spuštění:	Student/Vedoucí student vybere lekci
Hlavní tok:	1.Student/Vedoucí student vybere lekci, kterou chce prohlížet 2.Následně je zobrazen detail lekce
Alternativní toky:	
Výjimky:	Selhání operace, Selhání systému
Frekvence:	Často

ID:	4
Název:	Správa vlastních testových otázek
Vytvořeno:	Iveta Šenfeldová
Popis:	Vedoucí student spravuje své testové otázky
Primární aktéři:	Vedoucí student
Sekundární aktéři:	
Předpoklady:	Žádné
Následné podmínky:	Je provedena jedna nebo více z CRUD operací nad otázkami vedoucého studenta
Akce pro spuštění:	Vedoucí student vybere seznam otázek/editaci otázky/přidání otázky/odstranění otázky
Hlavní tok:	1.Pro vytvoření nové testové otázky vybere vedoucí student Vytvořit novou otázku 1.1 Po vyplnění formuláře a odeslání hodnot je otázka vytvořena a uložena do databáze 2.Pro odstranění otázky vybere vedoucí student Odstranit otázku a ta je následně odstraněna z databáze 3.Pro editaci otázky se načte formulář s vyplněnými hodnotami a po odeslání úprav se tyto promítnou v databázi
Alternativní toky:	
Výjimky:	Selhání operace, Selhání systému
Frekvence:	Zřídka

ID:	5
Název:	Změna hieararchie studentů
Vytvořeno:	Iveta Šenfeldová
Popis:	Povýšení studenta na pozici Vedoucího studenta nebo jeho sesazení
Primární aktéři:	Učitel
Sekundární aktéři:	
Předpoklady:	Rozhodnutí učitele
Následné podmínky:	Student je povýšen na Vedoucího studenta/Vedoucí student je sesazen o jednu nebo více úrovní
Akce pro spuštění:	Učitel si zobrazí stránku kurzu, kde má zobrazenou hierarchii studentů
Hlavní tok:	1.Pro povýšení studenta vybere učitel studenty a následně vybere Přiřadit vedoucího studenta 2.Pro sesazení vedoucího studenta jej učitel označí a vybere Sesadit vedoucího studenta
Alternativní toky:	
Výjimky:	Selhání operace, Selhání systému
Frekvence:	Často

ID:	6
Název:	Správa bodů studentů
Vytvořeno:	Iveta Šenfeldová
Popis:	Učitel spravuje veškerá bodová hodnocení studentů v jeho kurzech
Primární aktéři:	Učitel
Sekundární aktéři:	
Předpoklady:	Vznikl požadavek na změnu v bodových hodnoceních studenta
Následné podmínky:	Bodové hodnocení studenta je upraveno
Akce pro spuštění:	Učitel vstoupí do kurzu přes detail studenta
Hlavní tok:	1.Pro zapsání bodu učitel vybere Zapsat body 1.1 Je zobrazeno nová stránka s formulářem pro vytvoření bodového hodnocení 1.2 Po jeho potvrzení je zobrazena původní stránka, kde je již promítnuto nové hodnocení 2.Pro editaci bodového hodnocení je zobrazena stránka s formulářem, ve kterém jsou již načte data 2.1 Tyto data učitel upraví 2.2 Potvrdí změny a je zobrazena původní stránka s hodnocením studenta 3.Pro odstranění hodnocení učitel vybere Odstranit na příslušném řádku 3.1 Před odstraněním je učitel ještě dotázán, zda si opravdu přeje hodnocení odstranit a po potvrzní se tato změna ihned projeví
Alternativní toky:	
Výjimky:	Selhání operace, Selhání systému
Frekvence:	Často

ID:	7
Název:	Zobrazení studentů
Vytvořeno:	Iveta Šenfeldová
Popis:	Učitel má možnost zobrazení všech nebo pouze svých studentů
Primární aktéři:	Učitel
Sekundární aktéři:	
Předpoklady:	Žádný
Následné podmínky:	Je zobrazen seznam studentů
Akce pro spuštění:	Učitel vybere v menu položku Studenti
Hlavní tok:	Po vybrání položky v menu se učiteli zorbazí stránka se seznamem studentů
Alternativní toky:	
Výjimky:	Selhání operace, Selhání systému
Frekvence:	Často

ID:	7.1
Název:	Zobrazení studentů - Zapsání odebrání studenta z/do kurzu
Vytvořeno:	Iveta Šenfeldová
Popis:	Učitel má možnost zapsat studenta do svého kurzu nebo studenta z kurzu odebrat
Primární aktéři:	Učitel
Sekundární aktéři:	
Předpoklady:	Vznikne požadavek na zápis studenta do kurzu nebo na jeho odebrání z kurzu
Následné podmínky:	Vybraný student je přidán/odebrán do/z kurzu
Akce pro spuštění:	Učitel vybere Zapsat do kurzu nebo Odstranit z kurzu u vybraného studenta
Hlavní tok:	1.Po výběru Zapsat do kurzu nebo Odstranit z kurzu se zobrazí nové okno, ve kterém učitel vybere kurz, do kterého bude student zapsán nebo z kterého bude odstraněn 2.Po potvrzení se projeví změný v databázi
Alternativní toky:	
Výjimky:	Selhání operace, Selhání systému
Frekvence:	Zřídka

ID:	8
Název:	Chatování
Vytvořeno:	Iveta Šenfeldová
Popis:	Možnost online komunikace v systému
Primární aktéři:	Kdokoliv z přihlášených uživatelů
Sekundární aktéři:	
Předpoklady:	V systému jsou přihlášeni alespoň dva uživatelé
Následné podmínky:	Je zobrazena stránka s výpisem historie chatu
Akce pro spuštění:	Uživatel klikne na přihlášeného uživatele v contact listu v menu
Hlavní tok:	1.Po zobrazení stránky pro chat s vybraným uživatelem je zobrazena historie chatu 2.Po napsání zprávy a jejím odeslání je zobrazena na stránce 3. Zpráva je doručena příjemci
Alternativní toky:	
Výjimky:	Selhání operace, Selhání systému
Frekvence:	Zřídka