

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

E-learningový systém se zameřením na testování

Robert Soják

Vedoucí práce: Ing. Ondřej Macek

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

14. května 2012

Poděkování

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v příloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne

Abstract

Abstrakt

Obsah

1	Úvod	1
1.1	Cíl projektu	1
1.2	Využití	1
1.3	Obsah práce	2
2	Analýza	3
2.1	Popis problému	3
2.2	Požadavky na systém	3
2.2.1	Požadavky zadavatele	4
2.2.1.1	Funkční požadavky	4
2.2.1.2	Nefunkční požadavky	4
2.2.2	Rozšiřující požadavky	4
2.2.2.1	Funkční požadavky	4
2.2.2.2	Nefunkční požadavky	4
2.3	Rešerše existujících řešení	5
2.3.1	Google Formuláře	5
2.3.1.1	Tvorba formuláře	5
2.3.1.2	Typy otázek	6
2.3.1.3	Uživatelské rozhraní	7
2.3.2	Moodle	7
2.3.2.1	Otázky do testů	7
2.3.3	Studentova berlička	8
2.3.4	Závěr	8
2.4	Uživatelské role	8
2.5	Případy užití	9
2.6	Formuláře	10
2.6.1	Typy formulářů	10
2.6.2	Možnosti nastavení formulářů	11
2.7	Otázky	11
2.7.1	Typy otázek	11
2.8	Skripta	12
3	Návrh	13
3.1	Architektura systému	13
3.2	Datová vrstva	14

3.3	Business vrstva	14
3.4	Prezentační vrstva	14
3.5	Technologie a frameworky	14
3.5.1	C#ASP.NET MVC 4.0	14
3.5.2	Entity Framework	14
3.5.3	Javascript, JQuery	14
3.5.4	Wiki modul	14
4	Implementace	16
4.1	Datová vrstva	16
4.2	Práva uživatelů	18
4.3	Bezpečnost	19
4.3.1	Bezpečnost přenášených dat	19
4.3.2	Bezpečnost při práci v aplikaci	19
5	Testování	21
5.1	Unit testy	21
5.2	Selenium testy	21
5.3	Výkonové testy	21
5.3.1	Databáze	21
5.4	Pilotní provoz	21
6	Nasazení	22
7	Závěr	23
7.1	Další vývoj	23
7.2	Možná rozšíření	23
7.2.1	Otázky	23
7.2.2	E-mailové notifikace	23
7.3	Porovnání ASP.NET MVC a ASP.NET Web Forms	23
A	Seznam použitých zkratk	25
B	UML	26
B.1	Slovní popis případů užití	26
C	Instalační příručka	29
D	Uživatelská příručka	30
E	Unity Application Block tutoriál	31
F	Obrazová příloha	32
G	Obsah přiloženého CD	34

Seznam obrázků

2.1	Hlavní stránka Google Dokumentů	5
2.2	Ukázka Google Formuláře: tvorba vlevo, výsledek vpravo	6
2.3	Diagram užití	9
2.4	Stavový diagram formuláře	10
3.1	Architektura systému	13
3.2	Meta model	14
3.3	Návrhový model datové vrstvy	15
4.1	Entity Data Model	17
6.1	Diagram nasazení	22
F.1	Typy otázek v systému Moodle[6]	32
F.2	Architektura MVC	33

Seznam tabulek

2.1	Možnosti chování formulářů	11
-----	--------------------------------------	----

Seznam zdrojových kódů

4.1	Ukázka <code>UserPermissions</code> a souvisejících tříd – ukazuje implementaci oprávnění uživatelských rolí	18
4.2	Použití atributu <code>AuthorizeUserType</code>	19
4.3	Ověřování v business vrstvě	20
E.1	Předpis rozhraní <code>IUnityContainerAccessor</code>	31

Kapitola 1

Úvod

Tento projekt e-learningového¹ systému vznikl na základě poptávky neziskové organizace. Má sloužit jako nástroj pro naučné testování účastníků (dále *studentů*) kurzů, které organizace pořádá.

V současné době organizace využívá pro testování Google Formulářů² [2] s testovými otázkami, kde student vybírá právě jednu odpověď z několika možných. Skrze automaticky generovanou tabulku odpovědí jednotlivých studentů je sice koordinátor kurzu (dále *lektor*) schopen vyhodnotit správnost odpovědí, ale to je vše, co mu aplikace Formulářů umožňuje. Výsledky je nucen ručně vyhodnotit a danému studentovi nastítnit správné odpovědi nebo alespoň odkázat na související kapitolu v učebním materiálu (dále *skripta*).

Pro kategorizaci studentů a příslušných lektorů je v organizaci využito Google Skupin.

1.1 Cíl projektu

Cílem projektu je vyvinout portál pro podporu výuky. Hlavní funkcionalitou by měla být tvorba a vyplňování naučných testů určených skupinám studentů. Systém by měl také podporovat tvorbu a správu elektronických skript.

Jakožto bakalářská práce by měl projekt také ukázat mou schopnost samostatně řešit daný problém. Chtěl bych předvést nabyté znalosti a schopnost využít moderní technologie při jeho realizaci.

1.2 Využití

Výsledný systém by měl být nasazen do zmíněné poptávající neziskové organizace.

Má ale i širší využití, protože dává za vznik univerzálnímu e-learningovému portálu. Systém naučných testů (rozebrán ve 2.1 a 2.6.1) je zde svým způsobem originální a odlišuje portál od ostatních aplikací tohoto typu.

¹Vzdělávání za pomoci moderních elektronických informačních a komunikačních technologií

²Názvem Google Formulář zde značím formulář tabulkového procesoru z rodiny Google Dokumentů

Já osobně bych chtěl portál využít jako nástroj pro podporu výuky studentů informatiky na gymnáziu, kde působím jako učitel. Očekávám od něj efektivní pomoc při tvorbě a známkování testů a sdílení studijních materiálů.

Může být přínosem i pro další předměty jako jednoduše ovladatelný nástroj pro tvorbu tištěných testů či dotazníků, které jsou povětšinou těžkopádně tvořeny v textovém procesoru. Nastavil by se tím i jednotný vzhled, který má každý kantor odlišný.

1.3 Obsah práce

V této práci se dále věnuji analýze výsledného systému, kde ukazuji svůj postup při shromažďování požadavků, rešerši podobných systémů a promýšlení celkové funkcionality. Poté zde představuji návrh architektury systému a potřebných entit. Na to navazuji popisem implementace, kde zmiňuji některé postupy a konkrétní řešení některých částí aplikace. Věnuji se také testování a nasazení hotového portálu. Nakonec uvádím možná rozšíření a celkové zhodnocení.

Kapitola 2

Analýza

Při analýze zde nejdříve shromažďuji a popisuji požadavky na výsledný systém. Poté se věnuji rešerši již existujících systémů, abych získal přehled, co v současné době nabízejí a jak vypadají. To napomáhá k jasnějšímu pohledu na možná řešení pro splnění daných požadavků. Dále rozebírám konkrétní řešení mého systému.

2.1 Popis problému

Jak již vyplývá z úvodu, organizace potřebuje aplikaci pro tvorbu, vyplnění, okamžité vyhodnocení a okomentování testů. Ty zde značím jako *naučné testy*.

Lektor kurzu vytvoří šablonu testu s otázkami o několika možných odpovědích. Ke každé otázce by měl mít možnost napsat slovní komentář, ve kterém může nastínit, proč je daná odpověď správně nebo špatně a případně odkázat na studijní materiál, který danou problematiku vysvětluje. Každá otázka k sobě může vázat otázky alternativní, které test inovují při vyplňování stejného testu vícekrát.

Student by měl mít vždy přehled o dostupných testech, které jsou určené pro kurzy, do nichž je zařazen. Po konkrétním výběru se pro něj test vygeneruje z dostupných otázek a jejich alternativ. Vždy, když testovaný označí svou odpověď, zobrazení se vysvětlení správné odpovědi a dojde k jejímu viditelnému zvýraznění. Po vyplnění celého testu či pouze jeho části má student možnost odeslat svůj výsledek nadřazenému lektorovi. Ten má ve výsledku přehled, jak na tom jeho svěřenci se znalostmi jsou.

2.2 Požadavky na systém

Požadavky na výsledný e-learningový systém sestávají ze základních požadavků zadávající organizace (2.1) a rozšiřujících požadavků. Rozšiřující požadavky byly sestaveny na základě obecných požadavků na systém takového typu a inspirací zmíněných v rešerši v následující sekci 2.3. Díky těmto rozšiřujícím požadavkům by měl vzniknout zmíněný komplexnější univerzální e-learningový systém.

2.2.1 Požadavky zadavatele

2.2.1.1 Funkční požadavky

- FP1. Tvorba testů s výběrovými otázkami s právě jednou správnou odpovědí
- FP2. Zobrazení výsledku testu a možnost jeho odeslání lektorovi kurzu
- FP3. Každá otázka může sestávat z několika variací
- FP4. Náhodný výběr variace otázky při generování testu
- FP5. Zobrazení komentáře k právě zodpovězené otázce (cca odstavcový)
- FP6. Neomezený počet možností daný test opakovat dokud nebude zodpovězen správně
- FP7. Možnost kategorizace studentů, lektorů a příslušných testů do skupin

2.2.1.2 Nefunkční požadavky

- NP1. Jednoduché ovládání, pochopitelné pro běžného uživatele i bez zaškolení

2.2.2 Rozšiřující požadavky

2.2.2.1 Funkční požadavky

- FP8. Rozšíření o více typů testů (obecně *formulářů*)
- FP9. Možnost výběru z více typů otázek
- FP10. Lektor může ohodnotit přijatý formulář
- FP11. Podpora exportu výsledků formulářů do Google Formulářů
- FP12. Podpora tisku prázdných i vyplněných formulářů
- FP13. Možnost tvorby elektronických skript
- FP14. Podpora importu studentů z Google Skupin [3]

2.2.2.2 Nefunkční požadavky

- NP2. Výsledný systém je webová aplikace postavená na ASP.NET MVC

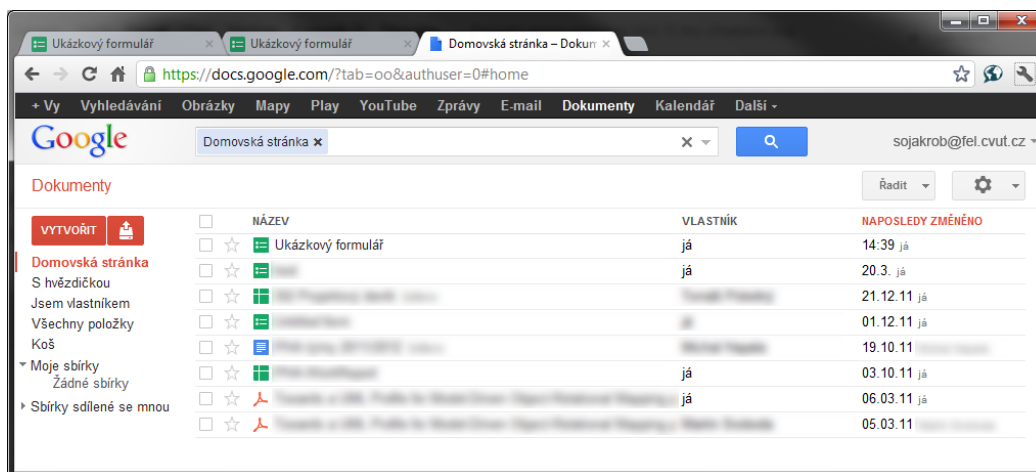
2.3 Rešerše existujících řešení

Rešerši existujících řešení e-learningových a tvorbou formulářů zabývajících se aplikací jsem neprováděl za účelem nalezení kandidáta na rozšíření o požadovanou funkcionalitu. Rešerši zde uvádím s cílem prozkoumat dostupná řešení, zjistit co poskytují za služby a třeba je využít pro inspiraci. Projekt byl již od začátku zamýšlen pro vznik takřka jíc na zelené louce. Jedním z důvodů je funkcionalita speciálního naučného testu (2.1, 2.6.1). Dalším důvodem je požadavek na jednoduché ovládání a zkušenost uživatelů ze zadávající společnosti s prostředím Google Formulářů. Hlavním důvodem je ale můj mojí snaha zapojit do vývoje znalosti softwarového inženýrství a touha vytvořit si systém vlastní. Nechtěl jsem jen vybrat z předpřipraveného řešení s hotovou architekturou a přibalit k ní plugin pro práci s naučnými testy.

2.3.1 Google Formuláře

Jako Google Formulář v této bakalářské práci značím formulář tabulkového procesoru z balíku Google Dokumenty[2] (na Obr. 2.1). Pomocí poskytovaného nástroje je uživateli umožněno vytvořit vlastní formulář z několika možných formulářových prvků. Hotový formulář lze poté zveřejnit (zcela nebo jen pro určitý okruh uživatelů) a nechat jej vyplnit. Odpovědi jsou automaticky zapisovány do tabulky, kde jsou spolu s dalšími informacemi (např. datum a čas vyplnění) připravené pro další analýzu.

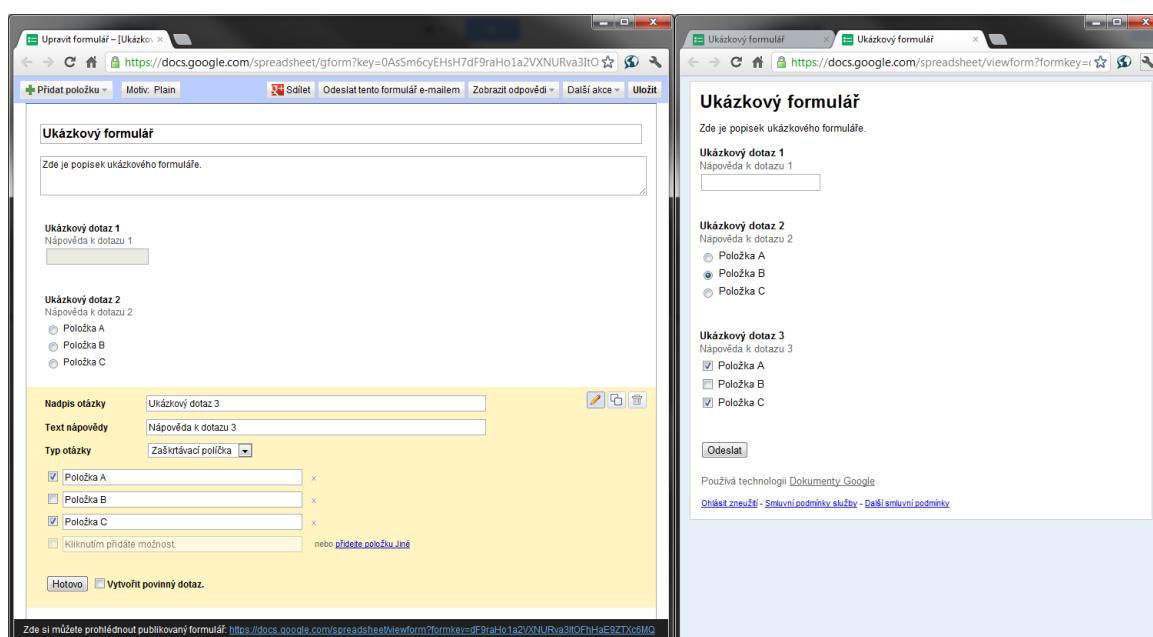
Základní myšlenka tohoto projektu vychází právě z Google Formulářů (lidé od zadavele jsou na ně zvyklí), proto zde jejich rozebrání věnuji více prostoru.



Obrázek 2.1: Hlavní stránka Google Dokumentů

2.3.1.1 Tvorba formuláře

Tvorba formuláře (viditelná na obr. 2.2) probíhá v, řekl bych, náhledovém režimu – uživatel vidí formulář téměř stejně jako koncový uživatel, jen nemůže vyplňovat odpovědi a



Obrázek 2.2: Ukázka Google Formuláře: tvorba vlevo, výsledek vpravo

otázky může jednotlivě upravovat. Úpravy se provádějí dynamicky pomocí Javascriptu a vše je automaticky ukládáno.

Otázka se skládá z nadpisu, textu nápovědy, volby typu otázky a zda je otázku povinné vyplnit. Nepovinný údaj je text nápovědy a dokonce i nadpis otázky, takže při nevyplnění vidí koncový uživatel jen prázdné pole či výčet odpovědí (v závislosti na typu otázky).

2.3.1.2 Typy otázek

Následuje výčet otázek, které lze na Google Formulář umístit, a použitelnost daného typu otázky ve vyvíjeném systému.

Text Krátký text, tázaný uživatel zapisuje odpověď do krátkého pole input¹, které už dále nemění svou velikost.

Text odstavce Pro tázaného zobrazen jako textarea, která se přizpůsobuje délce zapisovaného textu.

Více možností Jednoduchý výčet pomocí radio buttonů pro výběr jedné odpovědi.

Zaškrťávací políčka Výčet pomocí checkboxů, možnost vybrat žádnou či více odpovědí.

Vyberte ze seznamu Zobrazí uživateli combo box s výčtem možností. Vlastně stejná funkčnost jako Více možností, ale zabírá méně místa, jednotlivé odpovědi nejsou stále vidět.

¹Formulářový element HTML

Měřítka Umožňuje zvolení hodnoty ze zvoleného rozsahu, například 1–5. Tázanému je výběr umožněn pomocí radio buttonů.

Mřížka V podobě tabulky dává možnost zvolit pro každý řádek jednu hodnotu umístěnou ve sloupcích. Je zde ale omezení, sloupců může být max. 5 a pro jeden řádek nelze umožnit zvolení více hodnot najednou.

2.3.1.3 Uživatelské rozhraní

Protože uživatelé zadavatele jsou již zvyklí na rozhraní Google Dokumentů (na obr. 2.1 a 2.2) a toto rozhraní zároveň splňuje NP1, mělo by být grafické rozhraní e-learningového portálu podobné. Alespoň co se týče rozložení a umístění ovládacích prvků.

2.3.2 Moodle

Moodle [5] je asi nejznámější a celosvětově rozšířené prostředí pro elektronickou podporu výuky. Je to soubor mnoha modulů, které lze využít pro sestavení a nasazení vlastního výukového portálu. K dispozici je jako open source pod GNU GPL² licencí.

Jeho moduly umí snad vše, co se od takového systému očekává – od distribuce studijních materiálů přes tvorbu a vyhodnocování testů po diskuzní fóra. Jakékoli další moduly lze samozřejmě doprogramovat.

Z pohledu uživatele se mi zdá Moodle hodně složitý. Každá obrazovka chrlí velké množství různých nabídek, dat a možností. Nezkušený uživatel musí být doslova zavalen, nemluvě o vyučujících, kteří mají k dispozici podstatně více akcí a nastavení. Množství z nich je přístupné jen formou malých ikoněk u jednotlivých položek, což nutí orientovat se pomocí tooltipů. Začínající uživatel role studenta má ještě možnost po chvíli nalézt co hledá, ale lektor to má mnohem těžší. Běžní uživatelé, kteří mají problém používat akce textového procesoru, musí být v systému bez externí pomoci zcela ztraceni. Tomu se právě snažím v mém systému vyhnout.

2.3.2.1 Otázky do testů

V systému Moodle jsou otázky pojaty trochu jiným způsobem, než můžeme očekávat u přímočarého tvůrce formulářů (jako je např. výše popisovaný Google Formulář). Nejsou totiž vázány na jednotlivé testy, ale existují v tzv. otázkových bankách k určitému kurzu. Dále se dají řadit do kategorií a také sdílet s dalšími kurzy a vyučujícími.

Ve vyvíjení e-learningového systému se mi ale takový způsob nezdá být přínosem. Myslím si, že pro uživatele by bylo zbytečně složité řadit otázky do množství skupin a posléze každému formuláři tyto skupiny otázek nastavovat.

Systém obsahuje velké množství typů otázek, které vlastně rozšiřují základní běžně používanou množinu o specifitější (jako např. spojování). Výčet příkládám v příloze F.1.

²GNU General Public Licence, česky všeobecná veřejná licence GNU

2.3.3 Studentova berlička

Studentova Berlička je jeden z projektů studentů ČVUT FEL. Jedná se o informační systém pro podporu výuky na škole. Vývoj začal bakalářskou prací [4] v roce 2007 a poté byl systém dále rozvíjen a obohacován o další funkcionalitu. Zaměřil jsem se na modul pro generování testů [?].

Jednotlivé otázky jsou zde, podobně jako v Moodle, řazeny do tematických okruhů. Otázky do testu jsou vybírány z jednoho či více okruhů. Typy otázek jsou zde dva, slovní odpověď a výběrová.

Zajímavé je zde řešení postupu při tvorbě jednotlivých otázek. Otázka se může nacházet v jednom z těchto stavů:

- Rozpracované
- Můj lokál
- K validaci
- Ke schválení
- Schválené

Jak je již z výčtu stavů zřejmé, proces tvorby je uzpůsoben pro kooperaci více uživatelů, například cvičících a přednášejícího. Cvičící připravují otázky, které následně přednášející schvaluje.

Ačkoli je tento postup tvorby otázek pro vysokou školu určitě žádoucí, pro mnou vyvíjený systém by byl nadbytečný. V doméně uživatelů počítám s jedním vedoucím skupiny, který testové otázky sám vytváří. Z tohoto důvodu také v systému neřeším možné kolize při editování více uživateli naráz, což v tomto modulu Studentovi berličky řešeno je.

2.3.4 Závěr

V této sekci jsem popsal některé systémy, které jsem prostudoval za účelem zjištění nabízených služeb a způsobu jejich realizace. Hlavní inspirací pro základ e-learningového systému jsou pro mě Google Formuláře. Obsahují všechny základní typy otázek potřebné pro vytváření formulářů a jsou jednoduše ovladatelné. Naproti tomu Moodle mi ukazuje portál, kterému bych se blížít nechtěl. Obsahuje velké množství různých funkcí, kterých by v mé cílové skupině využilo možná procento uživatelů. Navíc je pro ně podle mě dosti složitý na ovládání.

2.4 Uživatelské role

Uživatelské role potřebné v systému jasně plynou již ze zadání, jsou to tyto tři role:

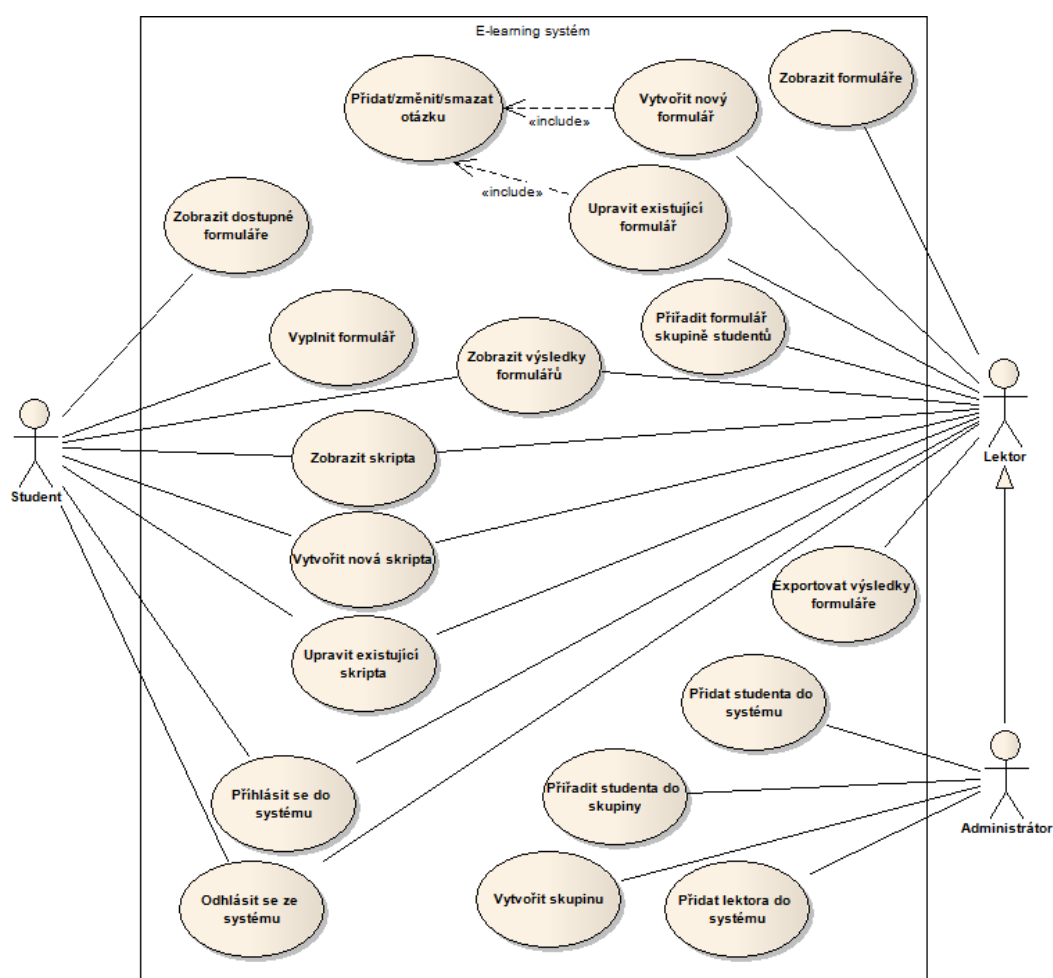
Student Představuje účastníka kurzu z domény zadavatele a obecně studenta využívajícího e-learningového systému. Z hlediska práv v aplikaci je na nejnižší úrovni, tzn. všechny jeho akce se vztahují pouze k němu samotnému. Mezi akce patří například vyplňování testů dostupné pro jeho skupinu a prohlížení výsledků svých testů (více v sekci 2.5).

Lektor Lektorem je koordinátor kurzu z domény zadavatele, vedoucí jedné či více skupin (neboli *kurzů*). Obecně se tedy jedná o učitele studentů, který má na starost tvorbu testů pro jeho skupiny a jejich následné ohodnocení.

Administrátor Uživatelská role známá asi ze všech systémů. Je to správce celého portálu a supervizor lektorů. Jako takový má práva všech lektorů a k tomu dostupnou funkcionalitu pro administraci aplikace včetně přidávání uživatelů a tvorby skupin.

2.5 Případy užití

Na diagramu 2.3 jsou zachyceny základní operace prováděné uživateli v daných rolích zastoupených aktéry případů užití. Slovní popis případů užití je umístěn v příloze B.1.



Obrázek 2.3: Diagram užití

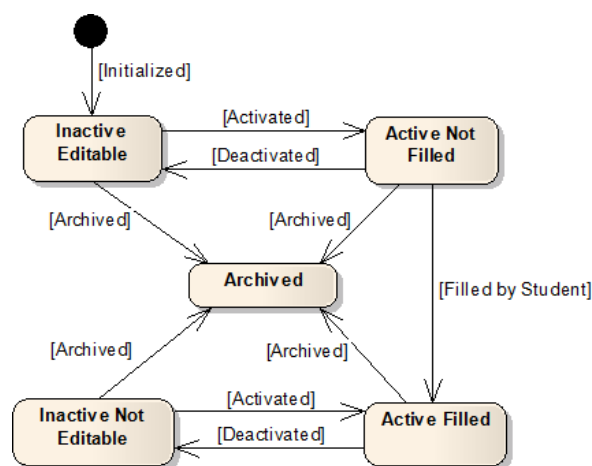
2.6 Formuláře

Formuláře představují základní nástroj tohoto e-learningového portálu. Slouží jak lektorům pro otestování znalostí jejich žáků, tak pro studenty samotné jako nástroj pro nauku a zpětnou vazbu.

Každý formulář je možné přiřadit do libovolného množství skupin. Vede-li lektor více skupin, má možnost určit jako cílové skupiny formuláře jejich podmnožinu. Studenti potom mají přístup ke všem formulářům z jím přiřazených skupin.

Mezi další možnosti, které formuláře obsahují, patří například nastavení časového limitu pro jeho vyplnění či zamíchání jednotlivých otázek a odpovědí. Více je popsáno v sekci 2.6.2.

Životní cyklus formuláře (viz. Obr. 2.4) začíná jeho založením lektorem a přechází do stavu *Neaktivní editovatelný*. V tuto chvíli jsou nastaveny možnosti formuláře a vytvářeny otázky. Aby byl viditelný pro studenty z přiřazených skupin, musí být uveden do stavu *Aktivní nevyplněný*. Pro opětovné pozastavení formuláře je možné přejít zpět do *Neaktivního* stavu. V případě, že byl již test někým vyplněn, pohybuje se mezi stavy *Aktivní vyplněný* a *Neaktivní needitovatelný*. V těchto stavech už není formulář možné editovat. Nakonec formulář přechází do stavu *Archivován*.



Obrázek 2.4: Stavový diagram formuláře

2.6.1 Typy formulářů

Základní množinu formulářů systému tvoří tyto tři typy:

Naučný test Je to netradiční typ testu vycházející z požadavků zadavatele. Ke každé otázce je možné přidat vysvětlení správné odpovědi (FP5), které je vyplňujícímu studentovi zobrazeno po označení odpovědi, kterou považuje za správnou. Test tedy plní dvě funkce, vypovídá o současných znalostech studenta a jeho připravenosti na možný zkouškový test. Zároveň ho opravuje a směřuje správnou cestou. Test je možné vyplnit vícekrát. Díky alternativním otázkám (FP3) není pokaždé stejný.

Zkouškový test Toto je běžný test pro ostré otestování studentových znalostí, jak ho známe ze všech vzdělávacích institucí. Může mít navíc určen například časový limit pro dokončení či možnost více pokusů pro správné vyplnění.

Dotazník Plní funkci běžného dotazníku. Každý ho smí vyplnit pouze jedenkrát.

2.6.2 Možnosti nastavení formulářů

Možnosti nastavení chování jednotlivých typů formulářů shrnuje následující tabulka [2.1](#).

	Naučný test	Zkušební test	Dotazník
Čas na vyplnění	✓	✓	✗
Zamíchání otázek	✓	✓	✗
Zamíchání nabízených odpovědí	✓	✓	✗
Vysvětlení správné odpovědi	✓	✗	✗
Možnost vyplnit vícekrát	✓	✓	✗

Tabulka 2.1: Možnosti chování formulářů

2.7 Otázky

Otázky tvoří náplň formulářů. Každá z otázek k sobě může vázat další, alternativní otázky. Když je pro studenta generován test, právě jedna alternativní otázka je vybrána. Jestliže tedy test obsahuje otázky s více alternativami, budou jednotlivé vygenerované testy rozdílné.

Jednotlivé otázky formuláře mohou být lektorem libovolně tvořeny, duplikovány, upravovány a mazány. Ovšem jen do okamžiku, kdy je formulář převeden do stavu *aktivní* a vyplněn některým ze studentů. V tom případě už otázky nesmějí být upravovány a mazány, aby měli všichni stejné podmínky a neměnily se jim otázky „pod rukama“.

Možnost duplikace otázek je zavedena hlavně kvůli výběrovým otázkám [2.7.1](#). Duplikací otázky tak odpadá nutnost opakovaného vkládání totožných odpovědí.

2.7.1 Typy otázek

Množina typů jednotlivých otázek je sestavena z potřeby zadavatele, čímž je otázka výběrová, a základních potřeb univerzálního e-learningového systému. Množina tedy obsahuje 5 typů otázek, z čehož 3 typy jsou základní. Možné další otázky vhodné pro rozšíření systému jsou nastíněny v sekci [7.2.1](#).

Výběrová otázka

Jedna správná odpověď Právě jedna z množiny nabízených odpovědí je správná.

Více správných odpovědí Žádná nebo více z nabízených odpovědí je správná.

Textová otázka

Krátká Od vyplňujícího studenta se očekává jednoslovná či jednovětná odpověď.

Dlouhá Zde se očekává vyčerpávající odpověď, tzn. jedna a více vět.

Rozsahová otázka Odpověď je třeba vybrat z definovaného rozsahu, např. známka 1–5, hodnocení 0–10. Tento typ otázky je hlavně pro potřebu dotazníku.

2.8 Skripta

Skripta hrají v systému roli elektronických učebních materiálů. Kategorizována jsou stejně jako formuláře, tzn. do skupin existujících v systému, takže studenti mají k dispozici materiály výhradně k jim přiřazeným kurzům. Skripta by měla být vytvářena převážně lektory pro studenty. Možnost jejich tvorby ale není studentům odepřena, mohou je využít například pro tvorbu osobních zápisků z hodin. Mají také možnost vytvořená skripta zveřejnit pro ostatní členy skupiny.

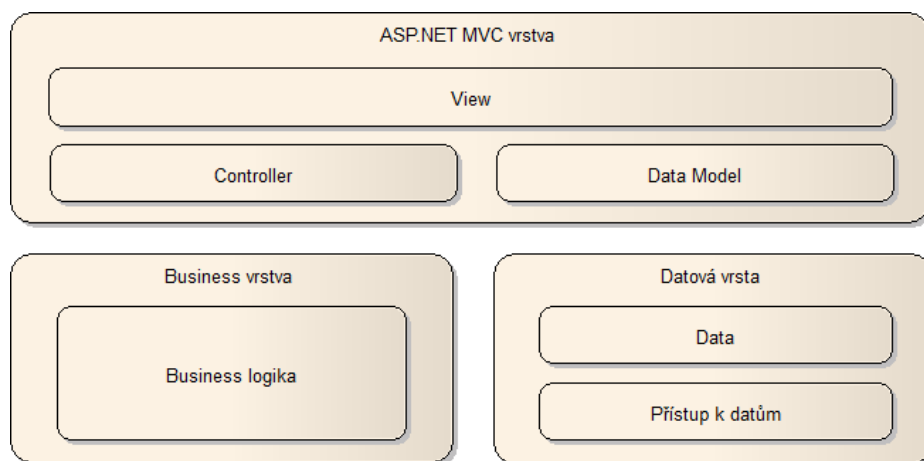
Skripta je možné formátovat do strukturované podoby, tzn. možnost tvorby nadpisů, odstavců, odkazů, tabulek, možnost volby stylu písma a další. K tomuto účelu by měl být k dispozici editor používající Wiki značkování³, která je snad pro většinu uživatelů pochopitelná a lehce naučitelná. S pomocí tlačítek pro automatické formátování vybraného textu nemusí mít ani uživatel potřebu pročítat dokumentaci k používání daného Wiki značkování.

³Syntaxe a klíčová slova využívaná k formátování textu na stránce

Kapitola 3

Návrh

3.1 Architektura systému



Obrázek 3.1: Architektura systému

Architektura systému (Obr. 3.1) je spojením architektury třívrstvé a MVC¹ (Obr. F.2). Použití MVC jasně vyplývá již z NP2. Ovšem architektura MVC sama o sobě podle mě o celkové struktuře systému nevypovídá mnoho. Pod pojem Model totiž skrývá několik různorodých komponent systému, konkrétně například business logiku (jejíž část ale bývá také v Controlleru) a datové uložště. A protože Model, View i Controller bývají v praxi velmi provázané, je nemožné jednu z komponent vyměnit za jinou s rozdílnou implementací bez zásahu do ostatních.

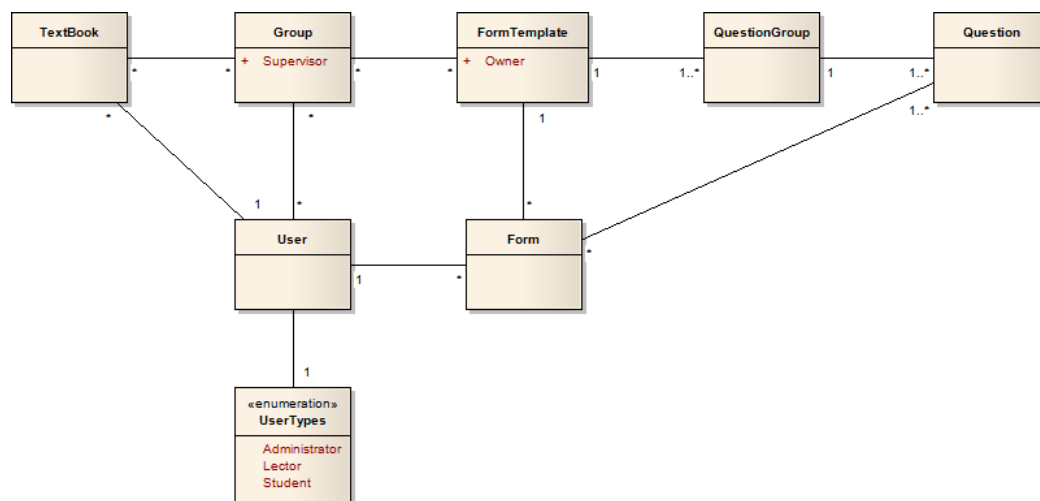
Já tedy využívám MVC jako „prezentační“ vrstvu z pohledu třívrstvé architektury. Mohu totiž tuto vrstvu kdykoli vyměnit například za Windows Forms, čímž vznikne desktopová aplikace využívající totožnou business a datovou vrstvu.

¹Model View Controller

Business vrstva je již standartní vrstva třívrstvé architektury, která obstarává veškerou aplikační logiku a zprostředkovává prezentační vrstvě zpracovaná data.

Datová vrstva obsahuje celý datový model systému a také komponenty potřebné pro CRUD² operace s těmito daty.

3.2 Datová vrstva



Obrázek 3.2: Meta model

3.3 Business vrstva

3.4 Prezentační vrstva

3.5 Technologie a frameworky

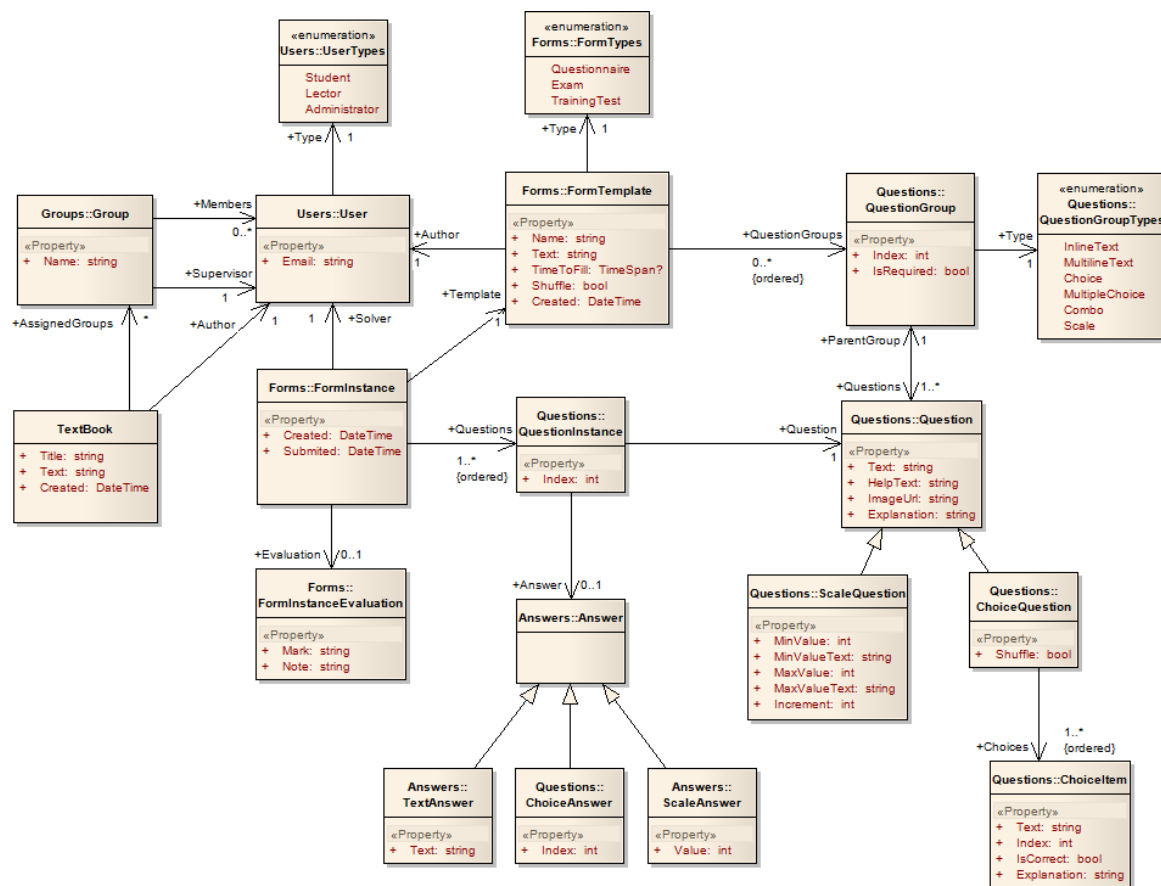
3.5.1 C#ASP.NET MVC 4.0

3.5.2 Entity Framework

3.5.3 Javascript, JQuery

3.5.4 Wiki modul

²Create, Read, Update & Delete - čtyři základní operace datového uložště



Obrázek 3.3: Návrhový model datové vrstvy

Kapitola 4

Implementace

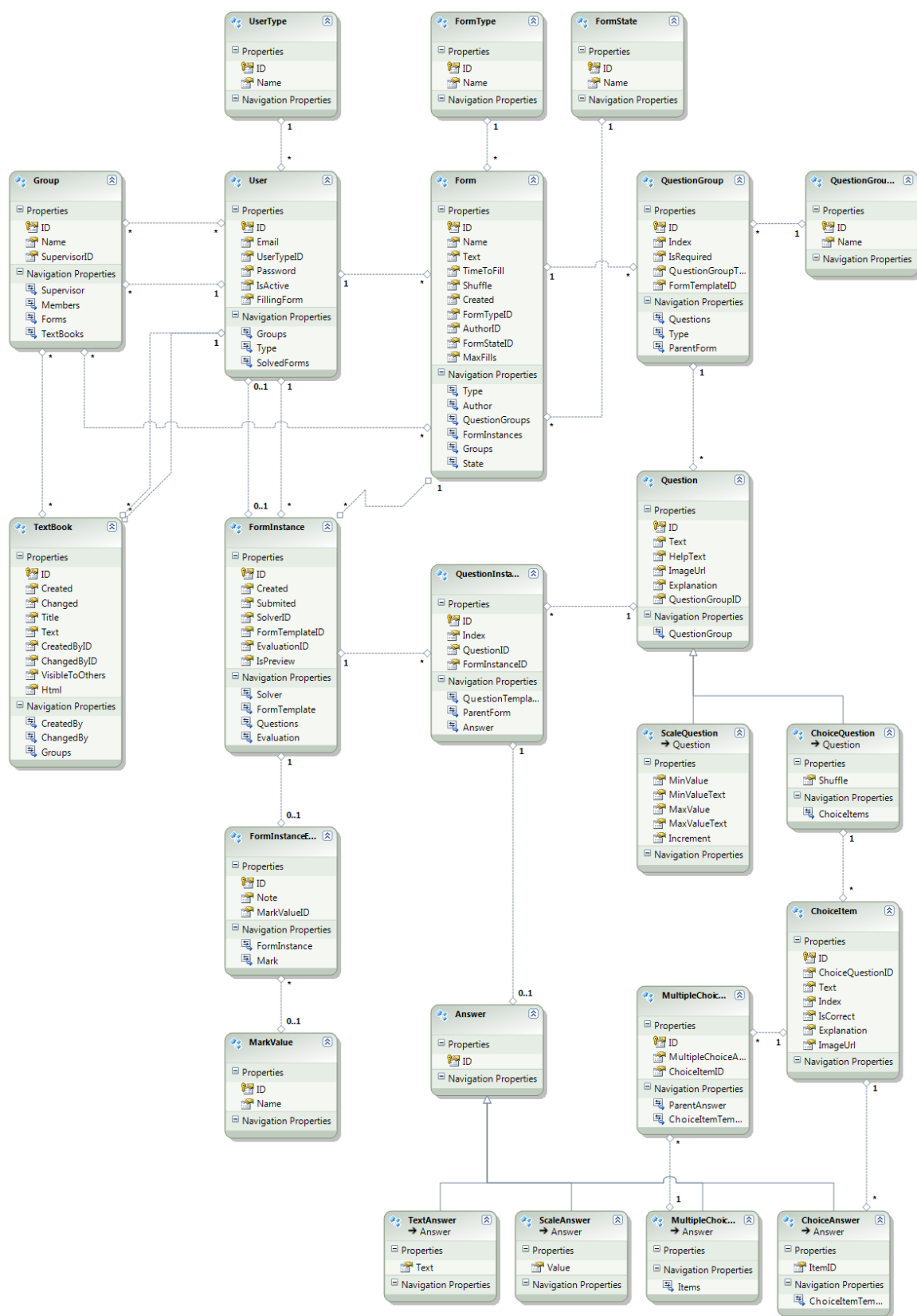
4.1 Datová vrstva

Pro implementaci datové vrstvy aplikace jsem použil Entity Framework (viz. 3.5.2). Zvolil jsem přístup Model First Development. Podle mě je to výborný kompromis mezi Code First a Database First řízením vývoji. Mám díky němu plnou kontrolu nad strukturou vytvářené databáze a zároveň nad generovanými třídami.

Všechny možné nedostatky skriptu databáze lze vyřešit přidáním databázového projektu a importem vygenerovaného skriptu do něj. V takovém projektu lze mít další skripty například pro spuštění před a po vyvolání hlavního skriptu. Post-deployment skriptu využívám také pro automatické vytvoření základních záznamů (číselníky, administrátor), čímž mě každé nasazení databáze stojí pouhé dvě kliknutí myši.

Úpravy generovaného kódu tříd lze jednoduše provést pomocí částečných tříd. Například u entit obsahujících hodnotu číselníku toho využívám pro navázání enumerátoru v kódu, představujícího daný databázový číselník. Nikde jinde v kódu pak tedy nemusím řešit žádné číselníkové objekty a identifikátory vztahující se k databázi.

Ve své implementaci datové vrstvy nevyužívám možnosti nastavit kaskádní mazání položek v databázi, všechna taková mazání je tedy nutné provádět „ručně“ v business vrstvě. Je to hlavně z důvodu zamezení vzniku všelijakých chyb z nepozornosti a neprostudování všech možných relací před zásahem do kódu. Je tedy nutné mít v metodách pro mazání pár řádků kódu navíc, ale tím jsou všechny vykonávané akce patrné a případně lze nějaké další akce při mazání vyvolat.



Obrázek 4.1: Entity Data Model

4.2 Práva uživatelů

Práva uživatelů se v tomto systému odvíjí od role, ve které se daný uživatel nachází. Není zde potřeba nastavovat každému uživateli specifická práva, protože role přesně definují jeho vztah k informacím v systému a operacím s nimi. To je rozdíl například od vnitrofiremních IS, kde je potřeba každému uživateli měnit práva pro přístup k datům a jednotlivým modulům.

Přesto jsou ale práva implementována tak, aby v případě budoucí potřeby (a zde také pro ukázkové akademické účely) bylo velmi jednoduché přiřadit každému uživateli práva nezávisle na jeho systémové roli. Abych dále mohl uvést podrobnější detaily, uvádím zde ukázkou implementace oprávnění uživatele:

```
namespace ELearning.Business.Permissions
{
    public class UserPermissions
    {
        // ...
        public virtual bool Group_List_All { get { return false; } }
        public virtual bool Group_CreateEdit { get { return false; } }
        public virtual bool Group_Delete { get { return false; } }
        // ...
        internal static UserPermissions Get(UserTypes userType)
        {
            switch (userType)
            {
                // ...
                case UserTypes.Administrator:
                    return new AdministratorPermissions();
                // ...
            }
        }
    }
    public class StudentPermissions : UserPermissions { /* ... */ }
    public class LectorPermissions : StudentPermissions { /* ... */ }
    public class AdministratorPermissions : LectorPermissions
    {
        // ...
        public override bool Group_List_All { get { return true; } }
        public override bool Group_CreateEdit { get { return true; } }
        public override bool Group_Delete { get { return true; } }
        // ...
    }
}
```

Zdrojový kód 4.1: Ukázka `UserPermissions` a souvisejících tříd – ukazuje implementaci oprávnění uživatelských rolí

Jak je z kódu 4.1 patrné, všechny dílčí práva (v ukázce práva na operace s uživatelskými skupinami) jsou umístěna do vlastní property. Vracení správné hodnoty v závislosti na roli uživatele je řešené pomocí dědičnosti. Pro úpravu práv, aby fungovala pro jednotlivé uživatele nezávisle na rolích, je potřeba jen přidat vlastní třídu pro vrácení hodnoty získané ne „natvrdo“, ale z relevantního záznamu z databáze.

4.3 Bezpečnost

Na bezpečnost této webové aplikace se dívám ze dvou směrů, a to z hlediska bezpečnosti přenášených dat mezi klientem a serverem a z hlediska bezpečnosti při práci uživatele v aplikaci. Jsou zde možná i další hlediska, například komunikace webového serveru s databázovým serverem, ale tu zde myslím není třeba rozebírat. Pro jednoduchost předpokládám, že aplikace i databáze běží na totožném serveru.

4.3.1 Bezpečnost přenášených dat

Jedná se o bezpečnost přenosu veškerých dat, která se přenášejí po síti (ať už po LAN či po internetu), když uživatel interaguje s aplikací. Nejdůležitější je asi moment, kdy se uživatel do aplikace přihlašuje. Odesílá své uživatelské jméno a heslo, díky čemuž je systémem identifikován a autorizován k provádění jednotlivých akcí.

Když přihlašovací údaje zachytí útočník (útokem známým jako MITM¹), získává tím identitu pravého uživatele a přístup ke všem jeho datům v systému. To dělá tento útok nejnebezpečnějším ze všech. Když se navíc útočníkovi podaří získat přihlašovací údaje administrátora systému, dostane se rázem k veškerým datům v systému. Navíc pro export do Google Formulářů musí uživatel zadat údaje ke svému Google účtu, takže v případě odposlechnutí dojde k narušení bezpečnosti i ve zcela jiném systému než je tento.

I když se ale útočníkovi nepodaří zachytit přihlašovací údaje, stále má možnost se hodně věcí dozvědět. Při zachycení komunikace vidí všechny informace, které si uživatel v danou chvíli prohlíží či zadává.

Naštěstí se dá útokům lehce zabránit, a to zavedením komunikace přes HTTPS namísto HTTP, což se řeší na úrovni webového serveru. Při komunikaci přes HTTPS je využito protokolu SSL, který s využitím serverového certifikátu veškerou komunikaci šifruje.

4.3.2 Bezpečnost při práci v aplikaci

Zde se jedná o bezpečnost interakce uživatele s rozhraním aplikace, tzn. zobrazování informací, přidávání a úprava dat, to vše podmíněné stupněm oprávnění uživatele a jeho přiřazením do skupin.

V aplikaci je implementováno „dvoustupňové“ zabezpečení. Prvním stupněm je kontrola oprávnění uživatele na úrovni prezentační vrstvy. Přístup na jednotlivé stránky je regulován pomocí zavedeného atributu `AuthorizeUserType`, který jsem připojil ke všem metodám controllerů. Předáním parametru `UserType` je určen nejnižší stupeň oprávnění pro přístup na danou url zastupovanou názvem metody. Při neoprávněném pokusu o přístup na url je atributem vyhozena výjimka typu `PermissionException`. Běžně by se uživatel na takovouto url vůbec dostat neměl, protože stejná politika je implementována ve views, čímž adresa takové url není uživateli dostupná.

```
[AuthorizeUserType(UserType = UserTypes.Lector)]  
public ActionResult Create()
```

¹Man In The Middle – útočník, který odposlouchává a případně mění komunikaci

```
{  
    // ...  
}
```

Zdrojový kód 4.2: Použití atributu `AuthorizeUserType`

Druhým stupněm je kontrola oprávnění v samotné business vrstvě. Všechny operace nad daty zde obsahují ověřují práva na vykonání pomocí `UserPermissions`, testují jestli požadovanou položku uživatel vytvořil či patří do jeho skupiny atd. Pokud práva nemá, vznikne opět výjimka `PermissionException`.

```
public Form GetForm(int id)  
{  
    var result = GetSingle(f => f.ID == id);  
    if (result == null)  
    {  
        if (Context.Form.Count(f => f.ID == id) > 0)  
            throw new PermissionException("Form_Get");  
        else  
            throw new ArgumentException("Form not found");  
    }  
  
    return result;  
}
```

Zdrojový kód 4.3: Ověřování v business vrstvě

Kapitola 5

Testování

5.1 Unit testy

Ověření funkčnosti hlavních tříd a jejich metod je provedeno pomocí Unit testů.

5.2 Selenium testy

Pomocí testovacího systému Selenium?? testuji aplikaci z pohledu uživatele. Selenium je systém, který ve spuštěném webovém prohlížeči vyvolává předem definované akce, jako je klikání na tlačítka a vkládání textů do příslušných polí. Lze také ověřovat stavy jednotlivých html elementů a mnoho dalšího. Tento testovací systém tedy simuluje chování uživatele při užívání systému. Díky těmto testům je ověřen jak obsah uživatelského rozhraní (dostupné akce a údaje), tak i samotné výsledky vyvolaných akcí.

Selenium mimo jiné umožňuje přímou integraci do projektu vyvíjeného systému. Poskytuje totiž C#knihovny, skrze které lze události v prohlížeči vyvolávat. Já jsem testy zapojil do testů podporovaných Visual Studiem. To přineslo obrovské výhody, protože testy stačí spustit stejně jako Unit testy, čímž se vytvoří instance prohlížeče a započne simulace chování uživatele. Ihned je tvořen report s úspěchy a neúspěchy jednotlivých dílčích testů.

Samotné scénáře Selenium testů není nutné psát celé ručně v kódu. Základ lze velice jednoduše vygenerovat automaticky – stačí zapnout nahrávání akcí ve webovém prohlížeči a chovat se jako uživatel. Nahraný scénář události se poté vyexportuje do jednoho z podporovaných jazyků, v mém případě do C#. Jediný malý nedostatek je, že kód je generován pro NUnit ?? testy, takže je ho třeba naportovat pro testy integrované do VS.

5.3 Výkonové testy

5.3.1 Databáze

5.4 Pilotní provoz

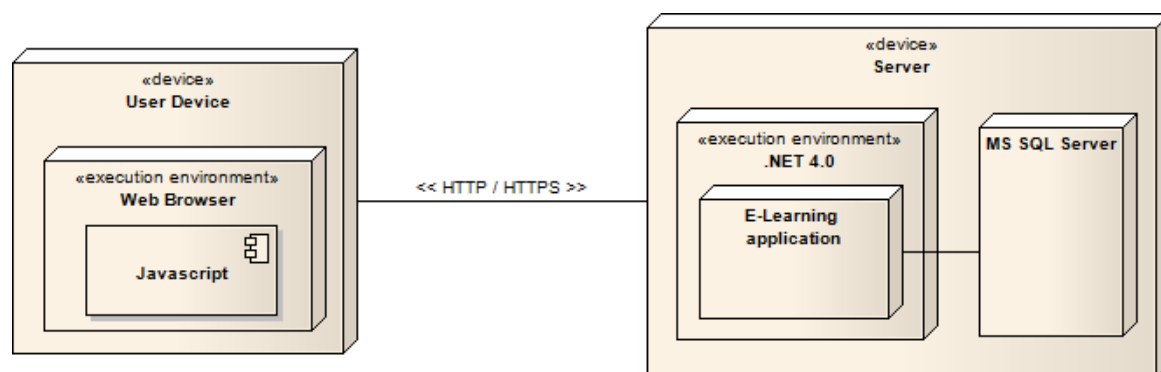
Kapitola 6

Nasazení

Nasazení aplikace je patrné z obrázku 6.1. Uživatel potřebuje internetový prohlížeč se zapnutou podporou Javascriptu. Jeho pomocí se připojuje k serveru, na kterém je nasazena aplikace.

Pro správný běh aplikace musí být na serveru nainstalován .NET Framework verze alespoň 4.0. Jako datové uložisko aplikace požaduje databázový server Microsoft SQL Server verze alespoň 2005. Databázový server může být nasazen i na vzdáleném serveru.

Popis postupu nasazení a instalace uvádím v instalační příručce v příloze C.



Obrázek 6.1: Diagram nasazení

Kapitola 7

Závěr

7.1 Další vývoj

S dalším vývojem systému počítám metodou dog fooding¹. Během užívání při výuce se budou jistě stále rodit nové požadavky a možné změny stávající funkcionality.

7.2 Možná rozšíření

7.2.1 Otázky

Možnost přiřadit k textu otázky obrázek.

7.2.2 E-mailové notifikace

7.3 Porovnání ASP.NET MVC a ASP.NET Web Forms

¹Programátoři používají vyvíjený software v roli koncového uživatele

Literatura

- [1] FINK, G. How To Use Unity Container In ASP.NET MVC Framework. Dostupné z: <<http://www.codeproject.com/Articles/99361/How-To-Use-Unity-Container-In-ASP-NET-MVC-Framewor>>.
- [2] Google Documents. Google Dokumenty — hlavní stránka. Dostupné z: <<http://docs.google.com/>>. poslední návštěva 14.5.2012.
- [3] Google Groups. Google Skupiny — hlavní stránka. Dostupné z: <<http://groups.google.com/>>. poslední návštěva 14.5.2012.
- [4] HUNKA, J. *Studentova berlička I*. ČVUT FEL, 2007. Bakalářská práce.
- [5] Moodle org. Portál věnovaný systému Moodle.
- [6] Výčet typů otázek Moodle. Dostupné z: <http://docs.moodle.org/22/en/images_en/2/23/Manage_question_types.png>. stav k 12.5.2012.

Příloha A

Seznam použitých zkratek

LAN Local Area Network

MITM Man In The Middle

HTTP HyperText Transfer Protocol

HTTPS HyperText Transfer Protocol Secure

IS Informační Systém

GPL General Public License

MVC Model View Controller

CRUD Create Read Update Delete

VS Visual Studio

⋮

Příloha B

UML

B.1 Slovní popis případů užití

Název: Přihlásit se do systému

Aktéři: Student, Lektor, Administrátor

Hlavní scénář:

1. Uživatel otevře stránky systému pomocí webového prohlížeče
2. Uživatel zvolí Přihlásit se
3. Systém zobrazí formulář pro zadání přihlašovacích údajů
4. Uživatel zadá přihlašovací údaje a potvrdí stiskem tlačítka Přihlásit se
5. Systém zobrazí domácí obrazovku uživatele

Alternativní scénář 1: Uživatel zadá špatné přihlašovací údaje v bodě 4

5. Systém zobrazí chybové hlášení
6. Pokračuje se bodem 4

Výsledek: Uživatel je přihlášen do systému

Název: Vytvořit nový formulář

Aktéři: Lektor, Administrátor

Vstupní podmínky:

- Uživatel je přihlášen do systému

Hlavní scénář:

1. Uživatel zvolí Vytvořit nový formulář

2. Systém zobrazí formulář pro zadání parametrů vytvářeného formuláře
3. Uživatel zadá požadované parametry a potvrdí tlačítkem Vytvořit
4. Systém vytvoří nový formulář na základě zadaných parametrů
5. Systém zobrazí obrazovku pro práci s otázkami vytvořeného formuláře

Alternativní scénář 1: Uživatel zadá špatné parametry v bodě 3

4. Systém zobrazí chybové hlášení o špatně vložených parametrech
5. Pokračuje se bodem 3

Alternativní scénář 2: Uživatel zruší vytváření formuláře zvolením jiné akce

Výsledek: Systém obsahuje nový formulář

Název: Přidat otázku

Aktéři: Lektor, Administrátor

Vstupní podmínky:

- Uživatel je přihlášen do systému
- V systému existuje formulář ve stavu Neaktivní editovatelný nebo Aktivní nevyplněný

Hlavní scénář:

1. Uživatel zvolí formulář, ke kterému chce otázku přidat
2. Systém zobrazí obrazovku pro práci s otázkami zvoleného formuláře
3. Uživatel zvolí požadovaný typ otázky a stiskne tlačítko Přidat
4. Systém přidá do kolekce otázek formuláře novou otázku zvoleného typu

Výsledek: Formulář obsahuje novou otázku

Název: Vyplnit formulář

Aktéři: Student

Vstupní podmínky:

- Uživatel je přihlášen do systému
- V systému existuje formulář ve stavu Aktivní nevyplněný nebo Aktivní vyplněný
- Formulář je přiřazen do stejné skupiny jako uživatel
- Formulář byl uživatelem vyplněn méněkrát než je jeho maximální počet vyplnění

Hlavní scénář:

1. Uživatel zvolí formulář k vyplnění
2. Systém zobrazí detailní informace o vybraném formuláři
3. Uživatel spustí vyplňování stiskem tlačítka Start
4. Systém vygeneruje instanci formuláře na základě jeho parametrů
5. Systém skryje hlavní ovládací prvky a zobrazí vygenerovaný formulář
6. Uživatel vyplní své odpovědi na otázky formuláře
7. Uživatel potvrdí odpovědi stiskem Odeslat
8. Systém odpovědi zaznamená, potvrdí přijetí a opět zobrazí hlavní ovládací prvky

Alternativní scénář 1: Vypršel časový limit formuláře během vyplňování v bodě 6

7. Systém ukončí vyplňování formuláře
8. Pokračuje se bodem 8 hlavního scénáře

Výsledek: Systém obsahuje novou instanci formuláře s odpověďmi na otázky

Název: Exportovat výsledky formuláře do Google Dokumentů

Aktéři: Lektor, Administrátor

Vstupní podmínky:

- Uživatel je přihlášen do systému
- V systému existuje formulář

Hlavní scénář:

1. Uživatel zvolí zobrazení výsledků formuláře k exportu
2. Systém zobrazí přehled vyplnění formuláře
3. Uživatel stiskne tlačítko Export výsledků do Google Dokumentů
4. Systém zobrazí formulář pro zadání přihlašovacích údajů Google účtu
5. Uživatel vyplní údaje a potvrdí export stiskem Exportovat
6. Systém exportuje výsledky do nového dokumentu v zadaném Google účtu

Výsledek: Uživatelův Google účet obsahuje nový dokument s výsledky formuláře

Příloha C

Instalační příručka

Příloha D

Uživatelská příručka

Příloha E

Unity Application Block tutoriál

Pro dependency injection využívám NuGet balíčku Unity Application Block. Zde bych chtěl přiblížit, jak jej začlenit do projektu.

Nejdříve je nutné do projektu nainstalovat Unity NuGet balíček (v menu Visual Studio zvolit Tools > Library Package Manager > Manage NuGet Packages..., v dialogovém okně vyhledat Unity a potvrdit instalaci).



























Pro přístup k Unity definuji podobně jako v [1] interface `IUnityContainerAccessor` s následujícím předpisem:

```
public interface IUnityContainerAccessor
{
    IUnityContainer UnityContainer { get; }
}
```

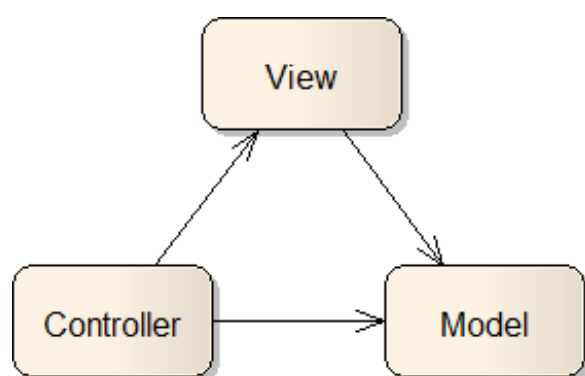
Zdrojový kód E.1: Předpis rozhraní `IUnityContainerAccessor`

Příloha F

Obrazová příloha

Question type	No. questions	Version	Requires	Available?	Delete	Settings
 Multiple choice	311 (+3 hidden)	2010090501		 ↑ ↓		
 True/False	67	2010090501		 ↑ ↓		
 Short answer	347	2010090501		 ↑ ↓		
 Numerical	10 (+1 hidden)	2010090501	Short answer	 ↑ ↓		
 Calculated	8	2010090501	Numerical	 ↑ ↓		
 Essay	15	2010090501		 ↑ ↓		
 Matching	32	2010090501		 ↑ ↓		
 Random short-answer matching	1	2010090501	Short answer	 ↑ ↓		
 Embedded answers (Cloze)	26	2010090501	Short answer, Numerical, Multiple choice	 ↑ ↓		
 Description	11	No database		 ↑ ↓		
 Random	69 (+2 hidden)	No database		↑ ↓		
 Missing type	1	No database		↑ ↓		
 Calculated multichoice	0	No database	Multiple choice	 ↑ ↓	Delete	
 Calculated Simple	0	No database	Numerical	 ↑ ↓	Delete	

Obrázek F.1: Typy otázek v systému Moodle^[6]



Obrázek F.2: Architektura MVC

Příloha G

Obsah přiloženého CD