

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

23. dubna 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
2	Popis problému, specifikace cíle	2
2.1	Cíl projektu	2
2.2	Využití	2
3	Analýza	4
3.1	Požadavky na systém	4
3.1.1	Požadavky zadavatele	4
3.1.1.1	Funkční požadavky	4
3.1.1.2	Nefunkční požadavky	4
3.1.2	Rozšiřující požadavky	4
3.1.2.1	Funkční požadavky	4
3.1.2.2	Nefunkční požadavky	5
3.2	Rešerše existujících řešení	5
3.2.1	Google Formuláře	5
3.2.1.1	Tvorba formuláře	5
3.2.1.2	Typy otázek	6
3.2.1.3	Uživatelské rozhraní	6
3.2.2	Moodle	7
3.2.2.1	Otázky do testů	7
3.2.3	Studentova berlička	8
3.2.4	Edubase	8
3.2.5	Závěr	8
3.3	Uživatelské role	8
3.4	Případy užití	8
3.5	Formuláře	8
3.5.1	Typy formulářů	9
3.5.2	Možnosti nastavení formulářů	10
3.6	Otázky	10
3.6.1	Typy otázek	11
4	Návrh	12
4.1	Architektura systému	12
4.2	Datová vrstva	12

4.3	Business vrstva	12
4.4	Prezentační vrstva	12
5	Implementace	14
5.1	Datová vrstva	14
5.2	Práva uživatelů	16
5.3	Bezpečnost	17
5.3.1	Bezpečnost přenášených dat	17
5.3.2	Bezpečnost při práci v aplikaci	17
6	Testování	19
6.1	Unit testy	19
6.2	Selenium testy	19
6.3	Výkonové testy	19
6.3.1	Databáze	19
6.4	Otestování v praxi	19
7	Nasazení	20
8	Použité technologie a frameworky	21
8.1	Entity Framework	21
8.2	Unity Application Block	21
8.3	Porovnání ASP.NET MVC a ASP.NET Web Forms	21
9	Závěr	22
9.1	Možná rozšíření	22
9.1.1	Otázky	22
A	Seznam použitých pojmů	24
B	Seznam použitých zkratk	25
C	UML diagramy	26
D	Instalace a uživatelská příručka	27
E	Obsah přílohy CD	28
F	Obrazová příloha	29

Seznam obrázků

3.1	Hlavní stránka Google Dokumentů	6
3.2	Ukázka Google Formuláře: tvorba vlevo, výsledek vpravo	7
3.3	Diagram užití	9
3.4	Stavový diagram formuláře	10
4.1	Architektura systému	12
4.2	Meta model	13
4.3	Návrhový model datové vrstvy	13
5.1	Entity Data Model	15
7.1	Diagram nasazení	20
F.1	Typy otázek v systému Moodle	29

Seznam tabulek

3.1	Možnosti chování formulářů	10
-----	--------------------------------------	----

Seznam zdrojových kódů

5.1	UserPermissions a související třídy	16
5.2	Použití atributu AuthorizeUserType	17
5.3	Ověřování v business vrstvě	18
8.1	Předpis rozhraní IUnityContainerAccessor	21

Kapitola 1

Úvod

Tento projekt e-learningového 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 Formulářů¹ s testovými otázkami, kde student vybírá právě jednu odpověď z několika možných. Skrze 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 nastínit 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.

¹Názvem Google Formulář zde značím formulář textového procesoru z rodiny Google Dokumentů

Kapitola 2

Popis problému, specifikace cíle

Jak již z výše nastíněného popisu vyplývá, organizace potřebuje aplikaci pro tvorbu, vyplnění, okamžité vyhodnocení a okomentování testů.

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 skupiny, 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.

2.1 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.

2.2 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 je zde svým způsobem originální a odlišuje portál od ostatních aplikací tohoto typu.

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 bývá test od testu odlišný.

Kapitola 3

Analýza

3.1 Požadavky na systém

Požadavky na výsledný e-learningový systém sestávají ze základních požadavků zadávající organizace a rozšiřujících požadavků, které byly sestaveny na základě obecných požadavků na systém takového typu a inspirací zmíněných v rešerši v následující sekci 3.2. Díky těmto rozšiřujícím požadavkům by měl vzniknout zmíněný komplexnější univerzální e-learningový systém.

3.1.1 Požadavky zadavatele

3.1.1.1 Funkční požadavky

- FP1. Zobrazení komentáře k právě zodpovězené otázce (cca odstavcový)
- 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. Neomezený počet možností daný test opakovat dokud nebude zodpovězen správně
- FP6. Možnost kategorizace studentů a příslušných lektorů

3.1.1.2 Nefunkční požadavky

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

3.1.2 Rozšiřující požadavky

3.1.2.1 Funkční požadavky

- FP7. Rozšíření o více typů testů (obecně *formulářů*)

FP8. Možnost výběru z více typů otázek

FP9. Lektor může ohodnotit přijatý formulář

FP10. Podpora exportu výsledků formulářů do Google Formulářů

FP11. Podpora tisku prázdných i vyplněných testů

FP12. Možnost tvorby elektronických skript

3.1.2.2 Nefunkční požadavky

NP2. Výsledný systém je webová aplikace

3.2 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 od začátku, takříkajíc na zelené louce. Jedním z důvodů je funkcionalita speciálního naučného testu. Dalším důvodem je požadavek na jednoduché ovládaní a zkušenost uživatelů ze zadávající společnosti s prostředím Google Formulářů. Hlavním důvodem je ale myslím moje 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.

3.2.1 Google Formuláře

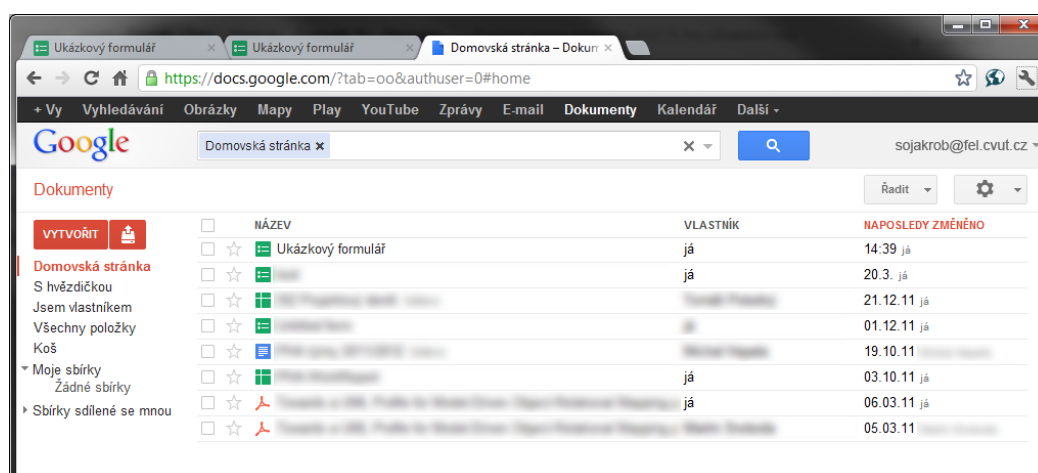
Jako Google Formulář v této bakalářské práci označuji formulář textového procesoru z balíku Google Dokumenty[2]. Pomocí poskytovaného nástroje je uživateli umožněno vytvořit vlastní formulář z několika možných formulářových prvků. Hotový formulář lze poté zveřejnit (zcela nebo jen pro určitý okruh uživatelů) a nechat uživateli 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, proto zde jejich rozebrání věnuji více prostoru.

3.2.1.1 Tvorba formuláře

Tvorba formuláře (viditelná na obr. 3.2) probíhá v, řekl bych, náhledovém režimu – uživatel vidí formulář téměř stejně jako koncový uživatel, akorát nemůže vyplňovat odpovědi a 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 odpovědi pro výběr (v závislosti na typu otázky).



Obrázek 3.1: Hlavní stránka Google Dokumentů

3.2.1.2 Typy otázek

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

Text Krátký text, tázaný uživatel zapisuje text do krátkého pole *input*, 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.

Zaškrtnutí políčka Výčet pomocí checkboxů, možnost vybrat žádnou a více odpovědí.

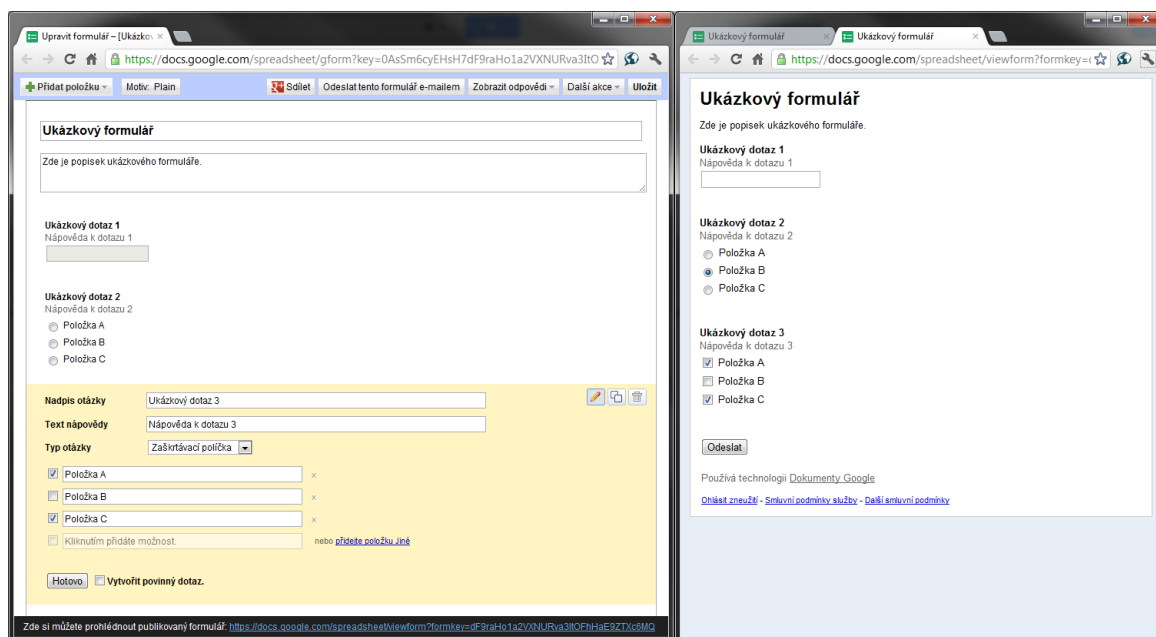
Vyberte ze seznamu Zobrazí uživateli combo box s výčtem možností. Vlastně stejná funkčnost jako Více možností, ale zabírá méně místa, jednotlivé odpovědi nejsou stále vidět.

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 nelze pro jeden řádek umožnit zvolení více hodnot najednou.

3.2.1.3 Uživatelské rozhraní

Protože uživatelé zadavatele jsou již zvyklí na rozhraní Google Dokumentů (na obr. 3.1 a 3.2) a toto rozhraní zároveň splňuje NP1, mělo by být grafické rozhraní e-learningového portálu podobné. Alespoň co se týče rozložení a umístění ovládacích prvků.



Obrázek 3.2: Ukázka Google Formuláře: tvorba vlevo, výsledek vpravo

3.2.2 Moodle

Moodle [3] je asi nejznámější a celosvětově rozšířený prostředek pro elektronickou podporu výuky. Je to vlastně soubor mnoha modulů, které lze využít pro sestavení a nasazení vlastního výukového portálu. K dispozici je jako open source pod GNU GPL 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 všemožných akcí a nastavení. Množství z nich je přístupné jen formou malých ikoněk u jednotlivých položek. 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.

3.2.2.1 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 vyvíjeném e-learningovém 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 skupiny otázek nastavovat.

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

3.2.3 Studentova berlička

3.2.4 Edubase

3.2.5 Závěr

3.3 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 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 viz. sekce [3.4](#)).

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 jako takový má práva všech lektorů a k tomu dostupnou funkcionalitu pro administraci aplikace včetně přidávání uživatelů a tvorby skupin.

3.4 Případy užití

Na diagramu [3.3](#) jsou zachyceny základní operace prováděné uživateli v daných rolích zastoupených aktéry případů užití.

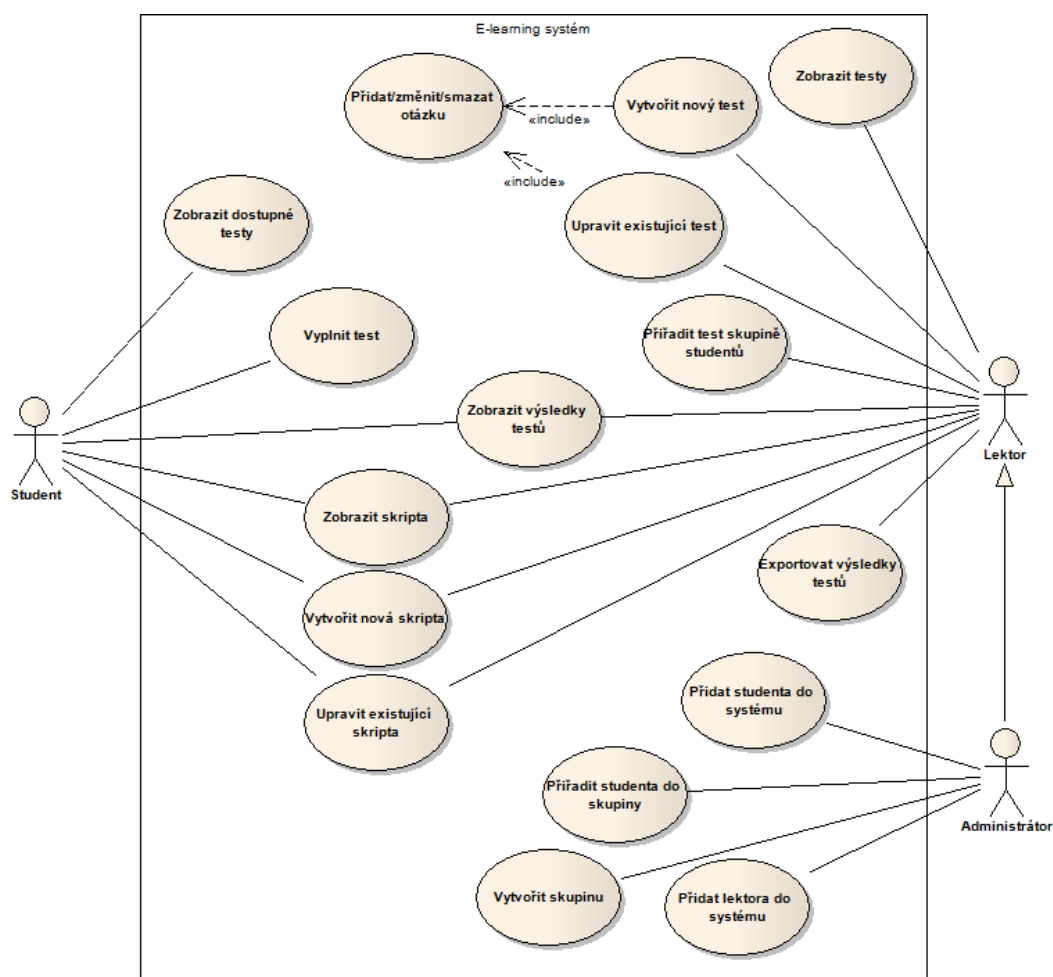
3.5 Formuláře

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 tím tedy možnost určit jako cílové skupiny formuláře jen 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 [3.5.2](#).

Životní cyklus formuláře (viz. Obr. [3.4](#)) začíná jeho založením lektorem. V tuto chvíli jsou nastaveny možnosti formuláře a vytvářeny otázky. Aby byl viditelný pro studenty z



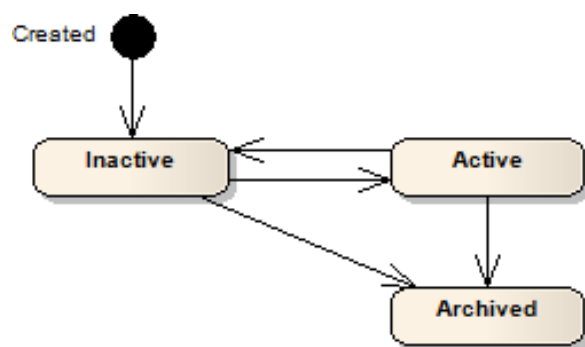
Obrázek 3.3: Diagram užití

přiřazených skupin, musí přejít do stavu *Aktivní*. Pro opětovné pozastavení formuláře je možné přejít zpět do *Neaktivního* stavu, kde už ale není povolena editace. Nakonec formulář přechází do stavu *Archivován*.

3.5.1 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 (FP1), které je vyplňujícímu studentovi zobrazeno po označení odpovědi, kterou považuje za správnou. Test tedy plní dvě funkce, vypovídá o současných znalostech studenta a připravenost na možný zkouškový test, a zároveň ho opravuje a směřuje správnou cestou. Test je možné vyplnit libovolněkrát. Díky alternativním otázkám (FP3) není pokaždé stejný.



Obrázek 3.4: Stavový diagram formuláře

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.

3.5.2 Možnosti nastavení formulářů

Možnosti nastavení chování jednotlivých typů formulářů shrnuje následující tabulka 3.1.

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

Tabulka 3.1: Možnosti chování formulářů

3.6 Otázky

Otázky tvoří náplň formulářů. Každá z otázek může k sobě do skupiny vázat další, alternativní otázky. Když je pro studenta generován test, právě jedna otázka je z této skupiny vybrána. Jestliže tedy test obsahuje otázky i s jejich alternativami, budou jednotlivé vygenerované testy rozdílné nejen z hlediska vzhledu (z důvodu možného promíchání otázek), ale také z hlediska obsahu.

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 3.6.1, aby byly zadané možnosti jednoduše použitelné i pro další otázku.

3.6.1 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 9.1.1.

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.

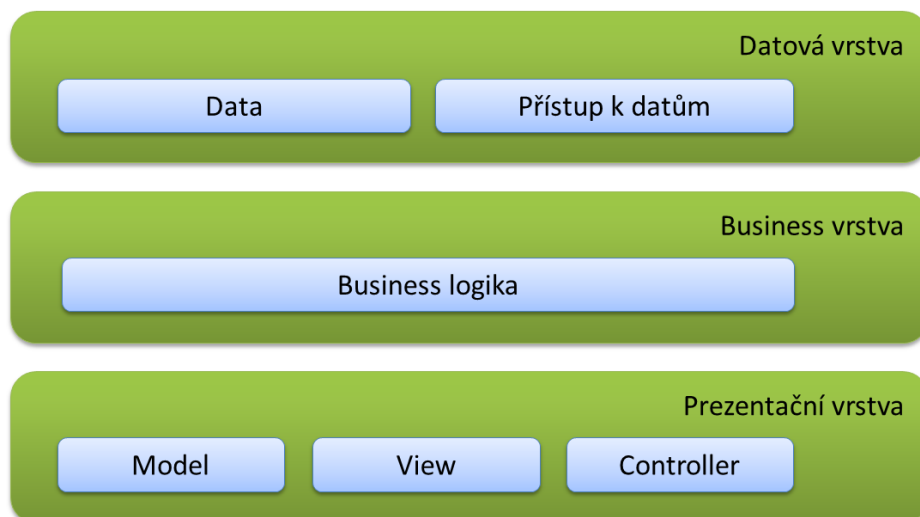
Dlouhá Zde se očekává vyčerpávající odpověď, tzn. jedna a více vět.

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

Kapitola 4

Návrh

4.1 Architektura systému

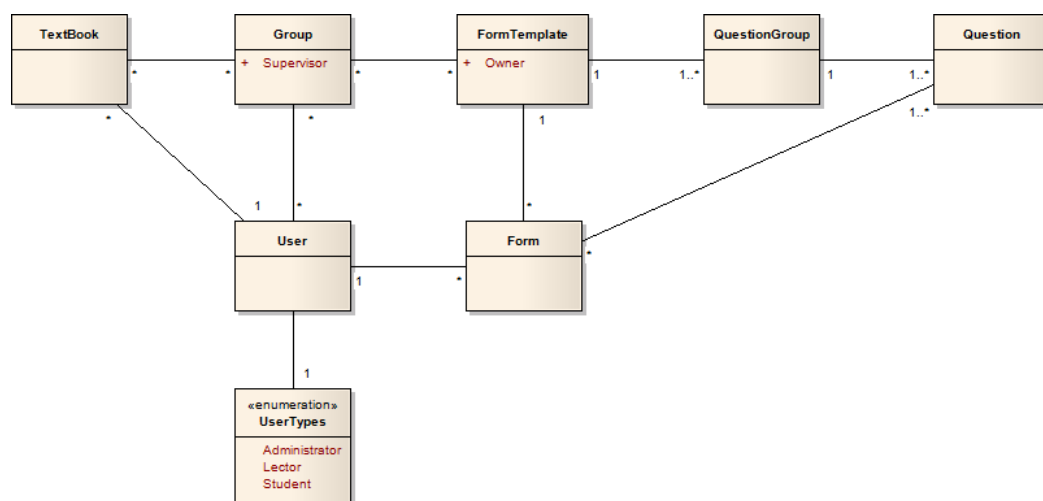


Obrázek 4.1: Architektura systému

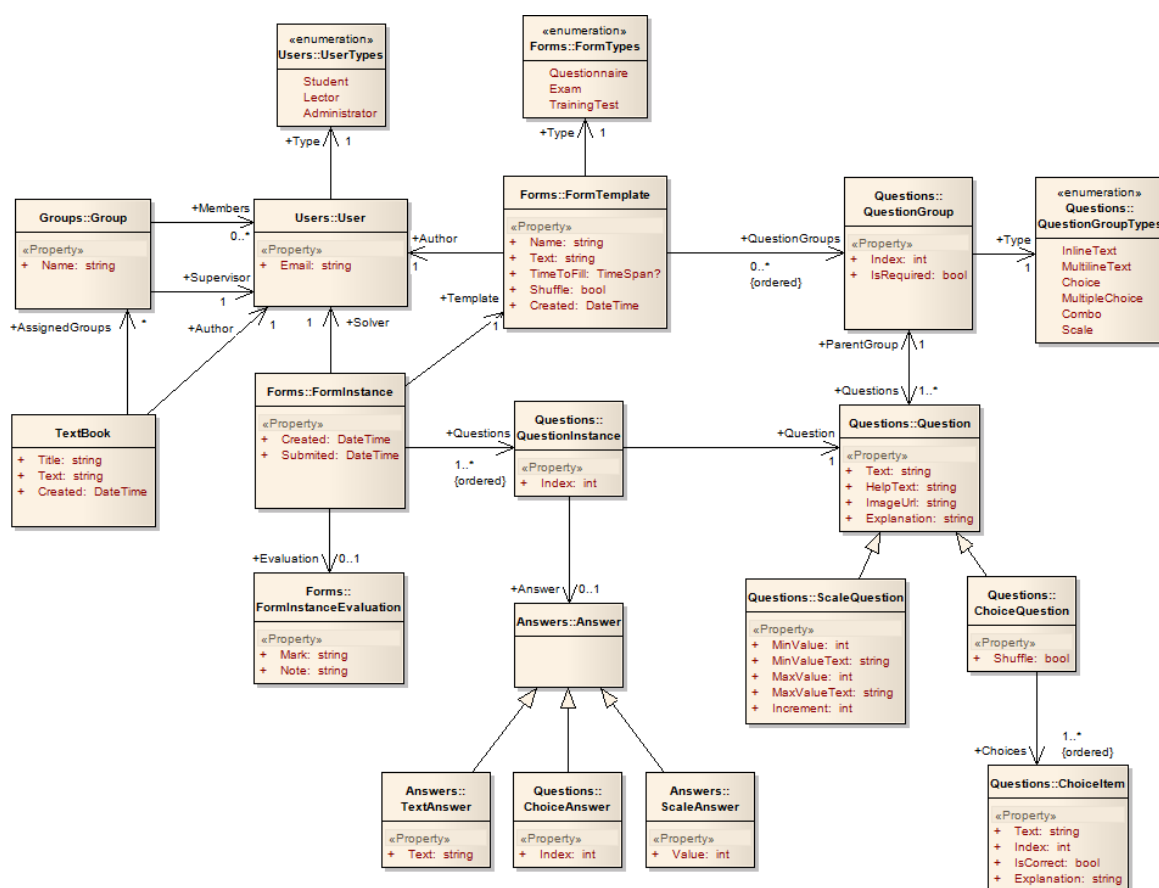
4.2 Datová vrstva

4.3 Business vrstva

4.4 Prezentační vrstva



Obrázek 4.2: Meta model



Obrázek 4.3: Návrhový model datové vrstvy

Kapitola 5

Implementace

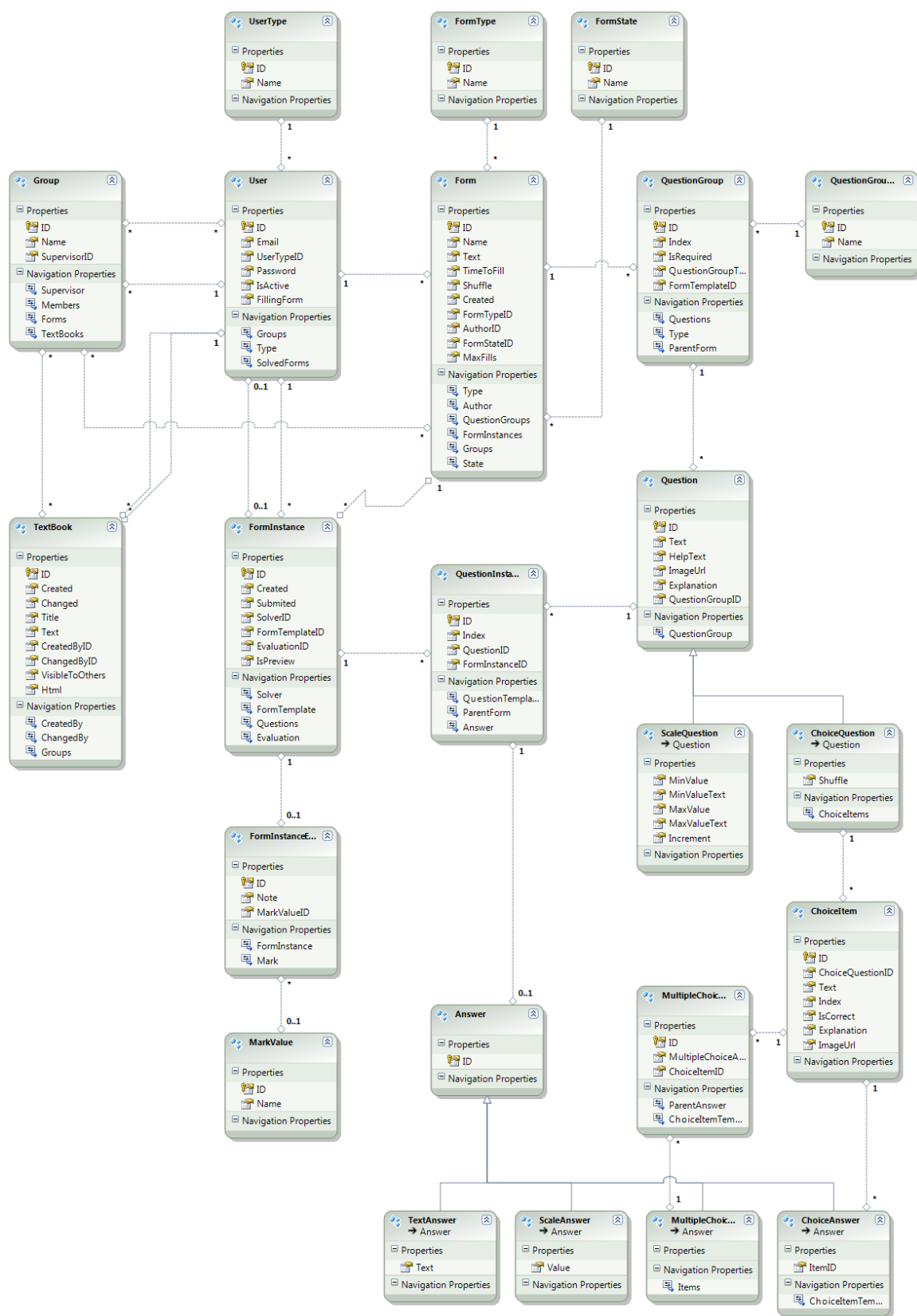
5.1 Datová vrstva

Pro implementaci datové vrstvy aplikace jsem použil Entity Framework (viz. 8.1). 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. 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 5.1: Entity Data Model

5.2 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 5.1: `UserPermissions` a související třídy

Jak je z kódu 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.

5.3 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.

5.3.1 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.

5.3.2 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 přiřazení 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 5.2: 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 5.3: Ověřování v business vrstvě

Kapitola 6

Testování

6.1 Unit testy

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

6.2 Selenium testy

6.3 Výkonové testy

6.3.1 Databáze

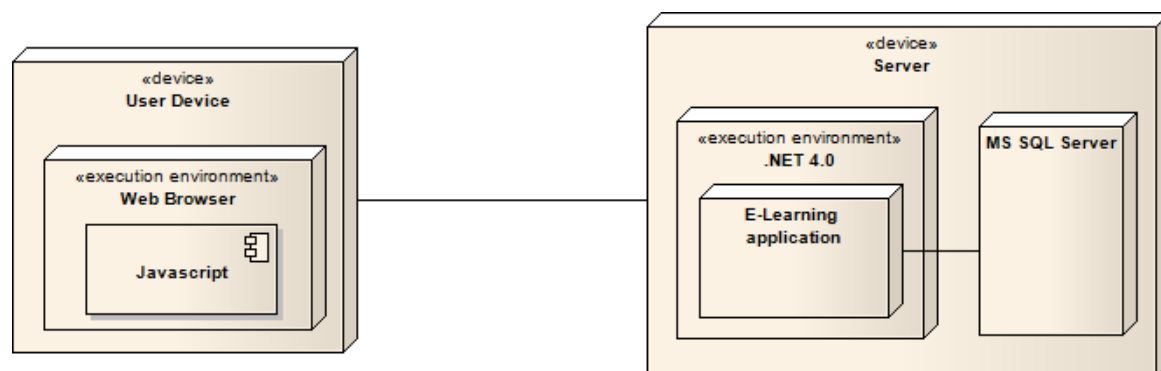
6.4 Otestování v praxi

Kapitola 7

Nasazení

Nasazení aplikace je patrné z obrázku 7.1. 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.



Obrázek 7.1: Diagram nasazení

Kapitola 8

Použité technologie a frameworky

8.1 Entity Framework

8.2 Unity Application Block

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 8.1: Předpis rozhraní `IUnityContainerAccessor`

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

Kapitola 9

Závěr

9.1 Možná rozšíření

9.1.1 Otázky

Možnost přiřadit k textu otázky obrázek.

Literatura

- [1] FINK, G. How To Use Unity Container In ASP.NET MVC Framework.
<http://www.codeproject.com/Articles/99361/How-To-Use-Unity-Container-In-ASP-NET-MVC-Fram>
stav z 22.4.2012.
- [2] web:GoogleDocs. Google Dokumenty — hlavní stránka.
<http://docs.google.com/>.
- [3] web:MoodleCZ. Portál věnovaný systému Moodle.

Příloha A

Seznam použitých pojmů

E-learning

Vzdělávání za pomoci moderních elektronických informačních a komunikačních technologií

⋮

Příloha B

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

⋮

Příloha C

UML diagramy

Příloha D



























Instalační a uživatelská příručka

Příloha E

Obsah přílohy E

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