

České vysoké učení technické v Praze  
Fakulta elektrotechnická  
Katedra počítačů



Bakalářská práce

## **E-learningový systém se zaměřením na testování**

*Robert Soják*

Vedoucí práce: Ing. Ondřej Macek

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

24. května 2012

## Poděkování

Na tomto místě bych chtěl poděkovat své rodině za neustálou podporu při studiu. Dále děkuji Ing. Ondřeji Mackovi za vedení práce, hlavně za pravidelné podnětné konzultace a poskytnutí zadání tohoto zajímavého projektu.

## 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 autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 24.5.2012

.....

# Abstract

This thesis deals with the analysis, development and testing of e-learning system to support teaching. The emphasis is on creating and filling tests with a special type added, a learning test. The portal also cooperates with some Google services, such as Docs and Groups. Result of the work is a web application that will be used for education by non-profit organization and for teaching in high school.

# Abstrakt

Tato práce se zabývá analýzou, vývojem a následným testováním e-learningového systému pro podporu výuky. Důraz je kladen na tvorbu a vyplňování testů s přidáním speciálním typem, naučným testem. Portál také spolupracuje s některými Google službami, jako jsou Dokumenty a Skupiny. Výsledkem práce je webová aplikace, která bude využívána ve vzdělávací neziskové organizaci a k výuce na střední škole.

# Obsah

<b>1</b>	<b>Úvod</b>	<b>1</b>
1.1	Cíl projektu . . . . .	1
1.2	Využití . . . . .	1
1.3	Obsah práce . . . . .	2
<b>2</b>	<b>Analýza</b>	<b>3</b>
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 . . . . .	11
2.6.2	Možnosti nastavení formulářů . . . . .	11
2.7	Otázky . . . . .	12
2.7.1	Typy otázek . . . . .	12
2.8	Skripta . . . . .	12
<b>3</b>	<b>Návrh</b>	<b>13</b>
3.1	Architektura systému . . . . .	13
3.2	Datová vrstva . . . . .	14

3.3	Business vrstva . . . . .	15
3.3.1	Rozpis balíčků . . . . .	15
3.4	ASP.NET MVC vrstva . . . . .	16
3.5	Technologie a frameworky . . . . .	17
3.5.1	C# ASP.NET MVC 4.0 . . . . .	17
3.5.2	ADO.NET Entity Framework . . . . .	17
3.5.3	HTML, CSS . . . . .	17
3.5.4	Javascript, jQuery . . . . .	17
3.5.5	MarkItUp a Wiki .NET Parser . . . . .	18
3.5.6	Google Data API . . . . .	18
<b>4</b>	<b>Implementace</b>	<b>19</b>
4.1	Datová vrstva . . . . .	19
4.2	Práva uživatelů . . . . .	21
4.3	Bezpečnost . . . . .	22
4.3.1	Bezpečnost přenášených dat . . . . .	22
4.3.2	Bezpečnost při práci v aplikaci . . . . .	22
<b>5</b>	<b>Testování</b>	<b>25</b>
5.1	Unit testy . . . . .	25
5.2	Selenium testy . . . . .	25
5.3	Výkonové testy . . . . .	26
5.3.1	Aplikace . . . . .	26
5.3.2	Databáze . . . . .	26
5.4	Pilotní provoz . . . . .	26
5.4.1	Testovací portál . . . . .	27
<b>6</b>	<b>Nasazení</b>	<b>29</b>
<b>7</b>	<b>Závěr</b>	<b>31</b>
7.1	Další vývoj . . . . .	31
7.2	Možná rozšíření . . . . .	32
7.2.1	Formuláře . . . . .	32
7.2.2	Otázky . . . . .	32
7.2.3	Skripta . . . . .	32
7.2.4	Reportování výsledků formuláře . . . . .	32
7.2.5	E-mailové notifikace . . . . .	32
<b>A</b>	<b>Seznam použitých zkratk</b>	<b>35</b>
<b>B</b>	<b>UML</b>	<b>37</b>
B.1	Popis případů užití . . . . .	37
<b>C</b>	<b>Instalační příručka</b>	<b>41</b>
<b>D</b>	<b>Unity Application Block tutoriál</b>	<b>45</b>

**E**   **Obrazová příloha**

**49**

**F**   **Obsah přiloženého CD**

**55**

# 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 . . . . .	11
3.1	Architektura systému . . . . .	13
3.2	Meta model . . . . .	14
3.3	Návrhový model datové vrstvy . . . . .	15
3.4	Balíčky business vrstvy . . . . .	16
3.5	Architektura MVC . . . . .	17
4.1	Entity Data Model . . . . .	20
4.2	Ukázka neoprávněného přístupu . . . . .	24
6.1	Diagram nasazení . . . . .	29
C.1	Volba konfigurace pro nasazení . . . . .	41
C.2	Nastavení databázového projektu . . . . .	42
C.3	Dialog pro publikování aplikace . . . . .	43
E.1	Typy otázek v systému Moodle . . . . .	49
E.2	Screenshot aplikace – Hlavní stránka studenta . . . . .	50
E.3	Screenshot aplikace – Vyplňování testu . . . . .	50
E.4	Screenshot aplikace – Vyplňování naučného testu . . . . .	51
E.5	Screenshot aplikace – Tvorba formuláře lektorem . . . . .	52
E.6	Ukázka tisku papírové verze formuláře . . . . .	53



# Seznam tabulek

2.1	Rozpis použitých definic pro jednotlivé případy užití . . . . .	10
2.2	Možnosti chování formulářů . . . . .	11

# Seznam zdrojových kódů

4.1	Ukázka <code>UserPermissions</code> a souvisejících tříd . . . . .	21
4.2	Použití atributu <code>AuthorizeUserType</code> . . . . .	23
4.3	Ověřování práv v business vrstvě . . . . .	23
D.1	Předpis rozhraní <code>IUnityContainerAccessor</code> . . . . .	45
D.2	Ukázka implementace <code>IUnityContainerAccessor</code> ve třídě <code>MvcApplication</code> . . . . .	45
D.3	Třída <code>UnityControllerFactory</code> . . . . .	46
D.4	Příklad registrace rozhraní pro injekci v <code>UnityContainerFactory</code> . . . . .	47
D.5	Příklad použití dependency injection v controlleru . . . . .	47

# Kapitola 1

## Úvod

Tento projekt e-learningového<sup>1</sup> 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 Formulářů<sup>2</sup> [12] 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 lektor nucen ručně vyhodnotit a danému studentovi nastínit správné odpovědi nebo ho 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. Jeho hlavní funkcí 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 a ukázat, jak umím uplatnit nabyté znalosti a využít moderních technologií při jeho realizaci.

### 1.2 Využití

Výsledný systém by měl být využit ve 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.

---

<sup>1</sup>Vzdělávání za pomoci moderních elektronických informačních a komunikačních technologií

<sup>2</sup>Názvem 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 většinou 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 nejprve věnuji analýze výsledného systému, kde ukazuji svůj postup při shromažďování požadavků, řešerši 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. Poté se věnuji testování a nasazení hotového portálu. Nakonec uvádím možná rozšíření a celkové zhodnocení.

# Kapitola 2

## Analýza

V této kapitole nejdříve shromážďuji a popisuji požadavky na výsledný systém. Poté se věnuji řešení již existujících systémů, abych získal představu, co v současné době nabízejí a jak vypadají. To napomůže k jasnějšímu pohledu na možná řešení pro splnění daných požadavků. Dále rozebírám konkrétní řešení mého systému.

### 2.1 Popis problému

Jak již vyplynulo 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 něho 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 z rozšiřujících požadavků. Rozšiřující požadavky byly sestaveny na základě obecných požadavků na systém takového typu a z inspirací zmíněných v řešení 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 variant
- FP4. Náhodný výběr varianty 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, zvládnutelné běžným uživatelem 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 [13]

### 2.2.2.2 Nefunkční požadavky

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

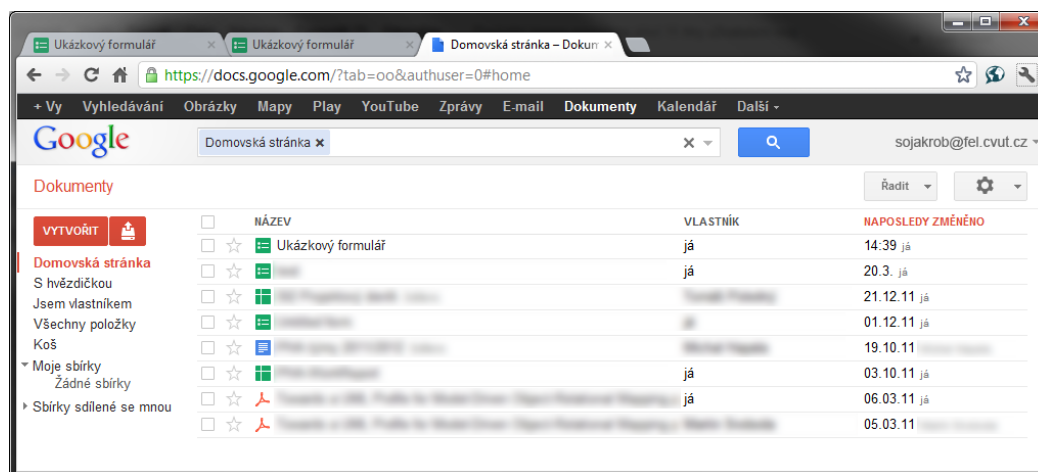
## 2.3 Rešerše existujících řešení

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 případně je využít pro inspiraci. Projekt byl již od začátku zamýšlen pro vznik „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 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.

### 2.3.1 Google Formuláře

Jako Google Formulář v této bakalářské práci značím formulář tabulkového procesoru z balíku Google Dokumenty [12] (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.

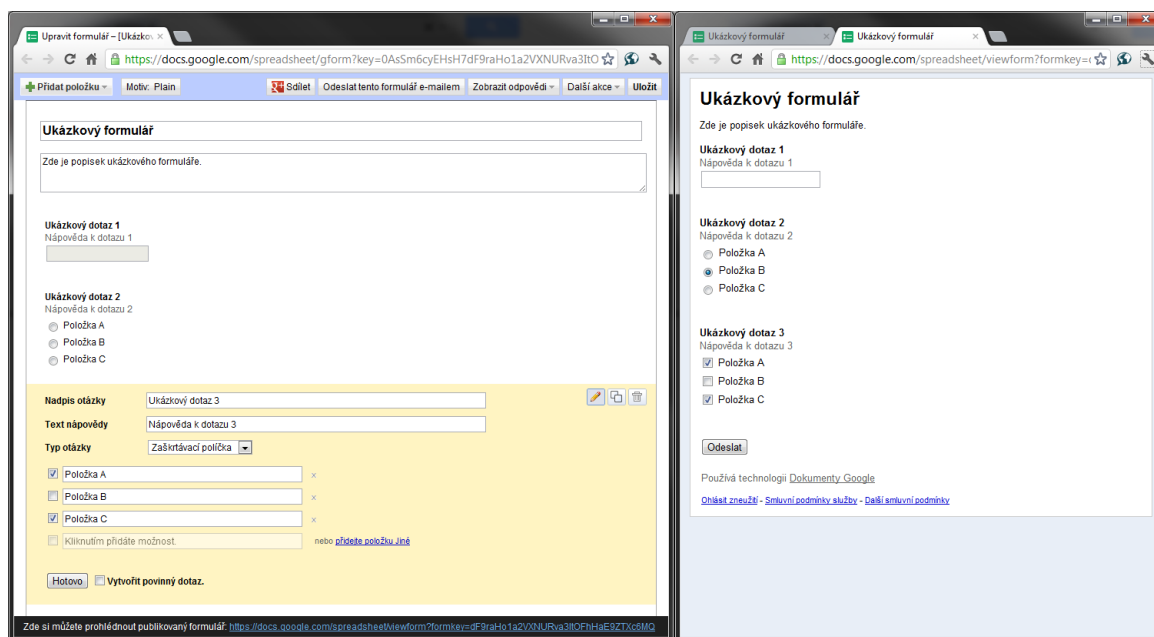
Základní myšlenka tohoto projektu vychází právě z Google Formulářů (lidé od zadavele jsou na ně zvyklí), proto zde jejich rozboru věnuji více prostoru.



Obrázek 2.1: Hlavní stránka Google Dokumentů – zde je vidět přehledný layout a design stránek

#### 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



Obrázek 2.2: Ukázka Google Formuláře: tvorba formuláře vlevo, výsledek je vidět vpravo

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 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, kde tázaný uživatel zapisuje odpověď do krátkého pole input<sup>1</sup>, které už dále nemění svou velikost.

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

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

**Zaškrťovací políčka** Výčet pomocí checkboxů, je možné 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.

<sup>1</sup>Formulářový 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 [15] je asi nejznámější, 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 GPL<sup>2</sup> licenci.

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

Systém Moodle pojímá otázky 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 E.1.

---

<sup>2</sup>GNU 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 [3] v roce 2007 a poté byl systém dále rozvíjen a obohacován o další funkcionalitu. Zaměřil jsem se na modul pro generování testů [5].

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 Studentovy 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 blížil nechtěl. Obsahuje velké množství různých funkcí, kterých by v mé cílové skupině využilo možná procento uživatelů. Navíc je pro ně podle mne dosti složitý na ovládání.

## 2.4 Uživatelské role

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

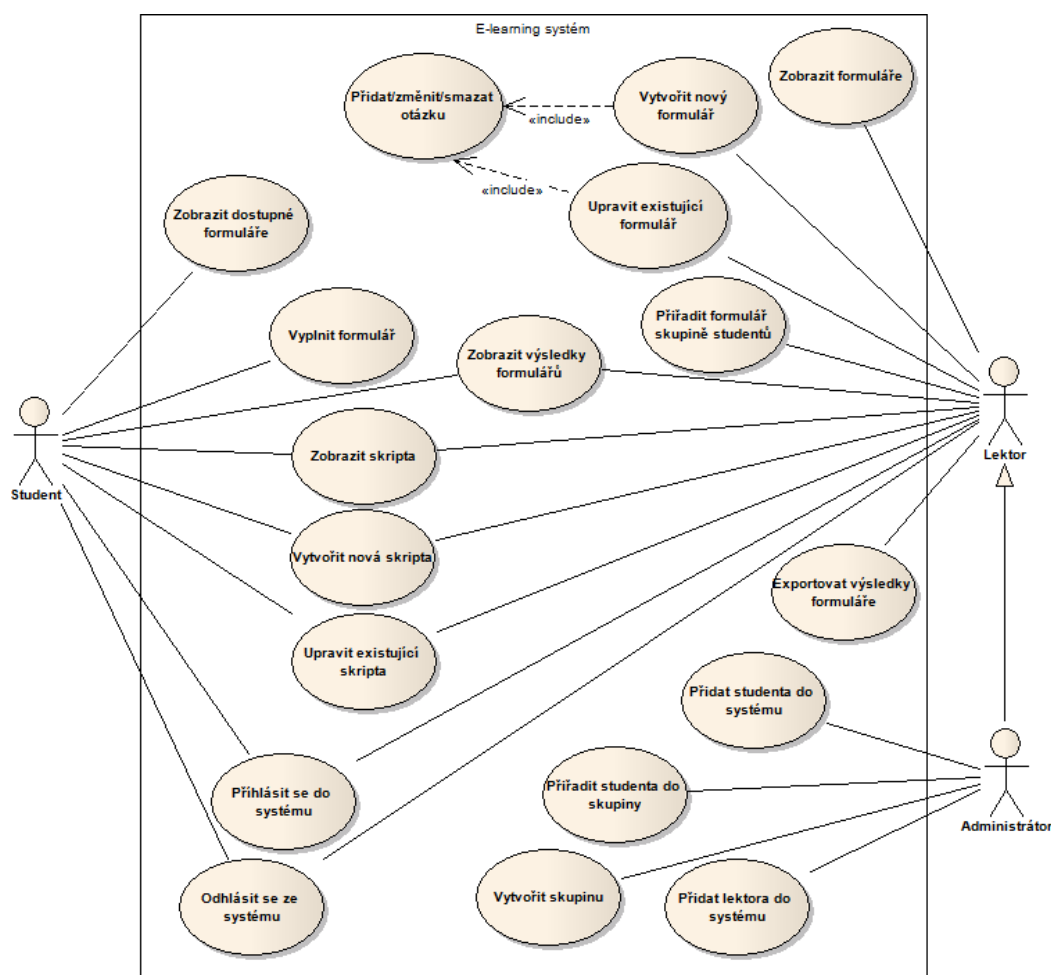
**Student** Představuje účastníka kurzu z domény zadavatele a obecně studenta využívajícího e-learningového systému. z hlediska práv v aplikaci je na nejnižší úrovni, tzn. všechny jeho akce se vztahují pouze k němu samotnému. Mezi akce patří například vyplňování testů dostupné pro jeho skupinu a prohlížení si 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 vybraných případů užití je umístěn v příloze B.1. Ostatní jsou realizovány pomocí Selenium testů, které jejich scénáře jasně vystihují. Výpis je umístěn v tabulce 2.1.



Obrázek 2.3: Diagram užití – zachycuje základní operace prováděné aktéry v daných rolích

Případ užití	Textový popis (B.1)	Selenium test (5.2)
Přihlásit se do systému	X	X
Odhlásit se ze systému		X
Zobrazit formuláře		X
Vytvořit nový formulář	X	X
Upravit existující formulář		X
Přidat/změnit/smazat otázku	X	
Přiřadit formulář skupině		X
Exportovat výsledky formuláře	X	
Zobrazit dostupné formuláře		X
Vyplnit formulář	X	X
Zobrazit výsledky formulářů		X
Zobrazit skripta		X
Vytvořit nová skripta		X
Upravit existující skripta		X
Přidat studenta do systému		X
Přiřadit studenta do skupiny		X
Vytvořit skupinu		X
Přidat lektora do systému		X

Tabulka 2.1: Rozpis použitých definic pro jednotlivé případy užití – některé případy jsou definovány textovým popisem, některé scénáři Selenium testů

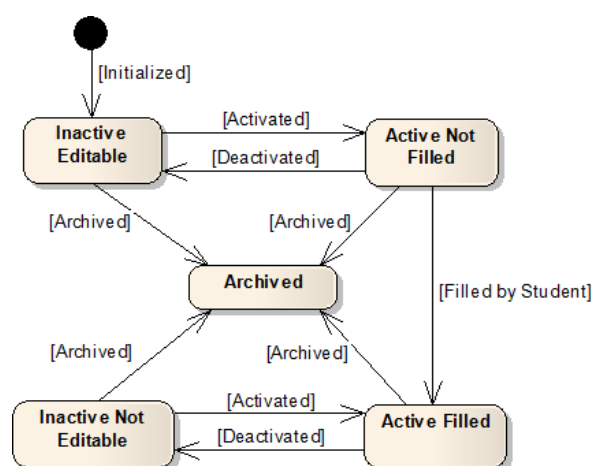
## 2.6 Formuláře

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 jim 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 zobrazující jeho životní cyklus

### 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 2.2.

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

Tabulka 2.2: Možnosti nastavení chování formulářů systému

## 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, tou je otázka výběrová, a ze 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 7.2.2.

#### 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 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 dalšího. k tomuto účelu by měl být k dispozici editor používající Wiki značkování<sup>3</sup>, 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.

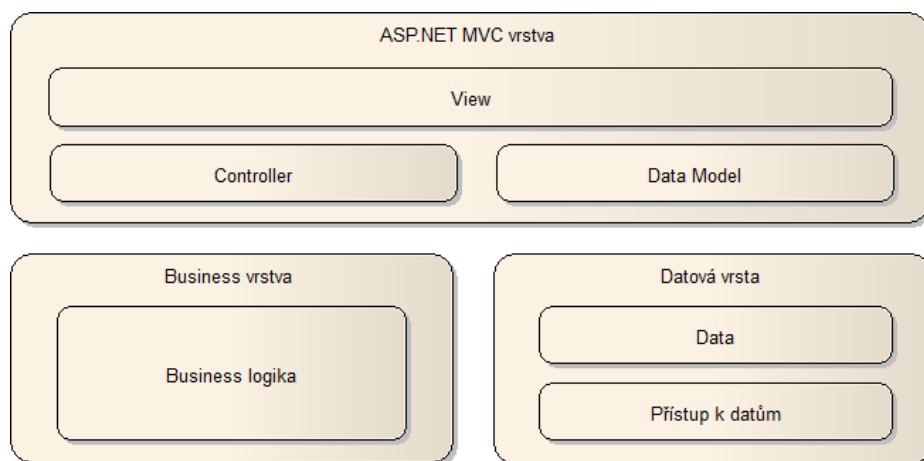
---

<sup>3</sup>Syntaxe a klíčová slova využívaná k formátování textu na stránce

# Kapitola 3

## Návrh

### 3.1 Architektura systému



Obrázek 3.1: Architektura systému – třívrstvá architektura s vrstvou datovou, bussines a „prezentační“ vrstvou postavenou na MVC vzoru

Architektura systému (Obr. 3.1) je spojením architektury třívrstvé a MVC<sup>1</sup> (Obr. 3.5). Použití MVC jasně vyplývá již z NP2. Ovšem architektura MVC sama o sobě podle mne o celkové struktuře systému nevypovídá mnoho. Pod pojem Model totiž skrývá několik různých komponent systému, konkrétně například business logiku (jejíž část ale bývá také v Controlleru) a datové uložení. 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.

---

<sup>1</sup>Model View Controller

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

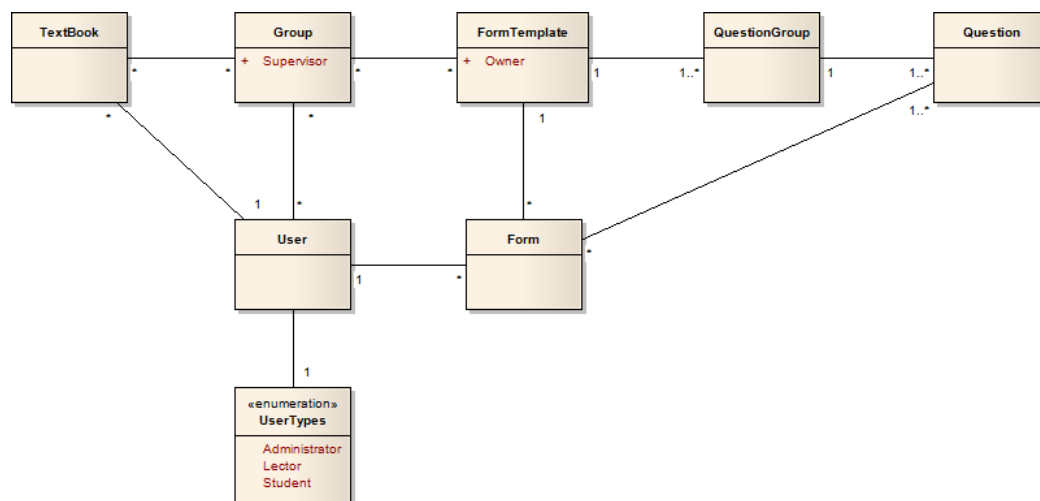
Datová vrstva obsahuje celý datový model systému a také komponenty potřebné pro CRUD<sup>2</sup> operace s těmito daty.

## 3.2 Datová vrstva

Na Meta modelu (Obr. 3.2) jsou vidět základní entity modelovaného systému a vztahy mezi nimi. Za zmínku stojí **FormTemplate** a **QuestionGroup**. První jmenovaná entita zastupuje šablonu formuláře, na jejímž základě jsou generovány jednotlivé instance (neboli konkrétní sestavy) formuláře, představující unikátní formulář určený k jednomu vyplnění studentem. **FormTemplate** obsahuje kolekci druhé jmenované entity **QuestionGroup**, což je skupina otázek (otázka a její alternativy, viz. 2.7). Do instance formuláře se vybírá právě jedna otázka z této množiny.

Návrhový model na obrázku 3.3 ukazuje podrobnější návrh objektů datové vrstvy. u jednotlivých objektů jsou vypsány jejich základní vlastnosti, se kterými by měl systém umět pracovat pro dosažení požadované funkcionality analyzované v kapitole 2.

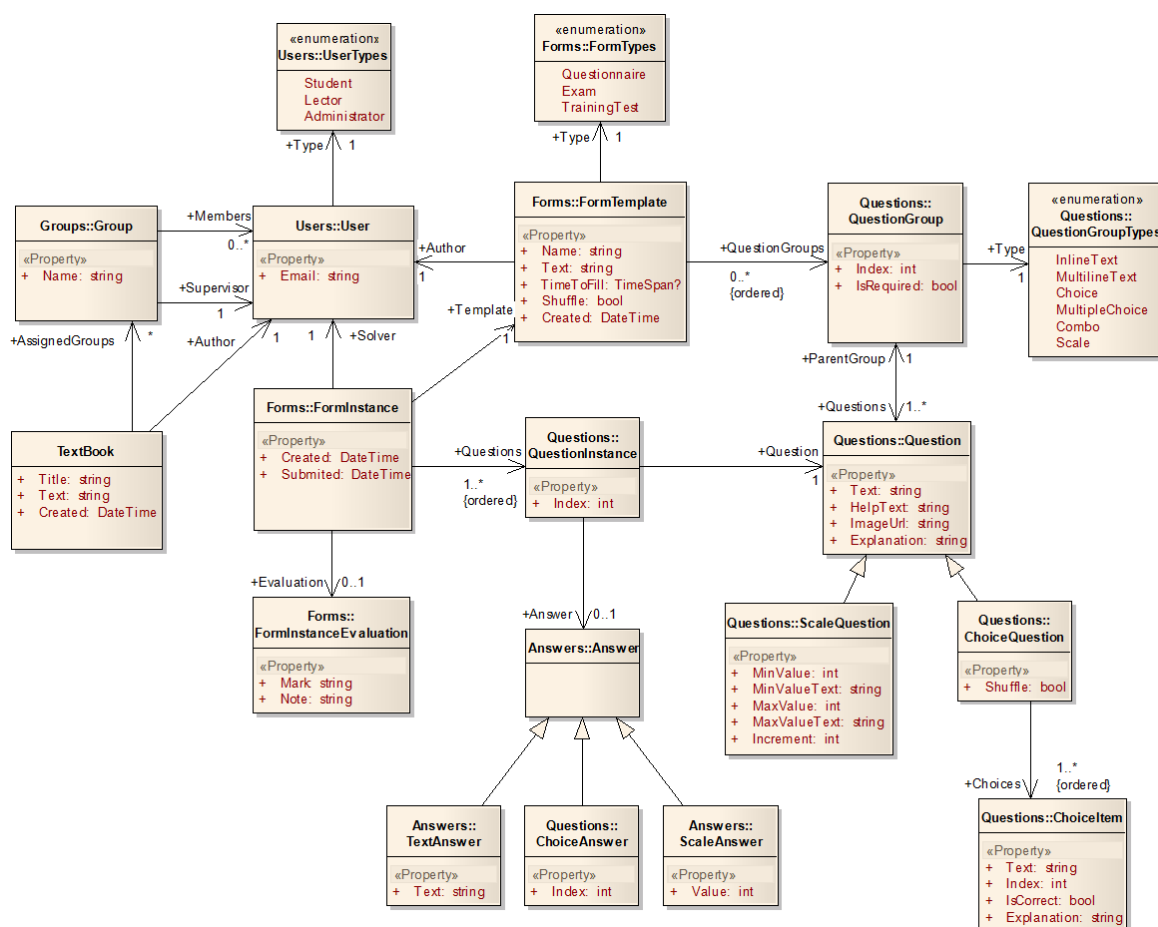
Mimo jiné bylo nutné přidat vlastní objekt instance otázky (**QuestionInstance**), který je třeba pro zachycení pozice otázky ve formuláři (jelikož je možné při generování instance testu nechat otázky promíchat). Další zanesenou entitou je **Answer** zastupující odpověď na otázku a **FormInstanceEvaluation** tvořící ohodnocení instance formuláře. Přibyl také objekt **ChoiceItem**, což je položka možné odpovědi na otázku typu výběrová (2.7.1).



Obrázek 3.2: Meta model zachycující základní entity modelovaného systému a vztahy mezi nimi

<sup>2</sup>Create, Read, Update & Delete - čtyři základní operace datového uložště





Obrázek 3.3: Návrhový model datové vrstvy ukazující podrobnější návrh objektů datové vrstvy

### 3.3 Business vrstva

Jak již bylo řečeno výše, business vrstva obsahuje veškerou logiku tohoto e-learningového systému a operace pro práci s daty získané pomocí datové vrstvy. Jelikož obsahuje různorodé komponenty, je rozdělena do několika balíčků, které jsou i se svým obsahem patrné z diagramu na obrázku 3.4.

#### 3.3.1 Rozpis balíčků

**Managers** Balíček obsahující třídy, které nazývám „manažery“. Mají za úkol poskytovat funkcionalitu pro práci s jím svěřenou datovou entitou z datové vrstvy, například **TextBookManager** obsahuje implementaci spojenou se skripty a **GroupManager** se skupinami.

**Permissions** Zde je místo pro práva-definující třídy. o implementaci s nimi spojené je více popsáno v sekci 4.2.

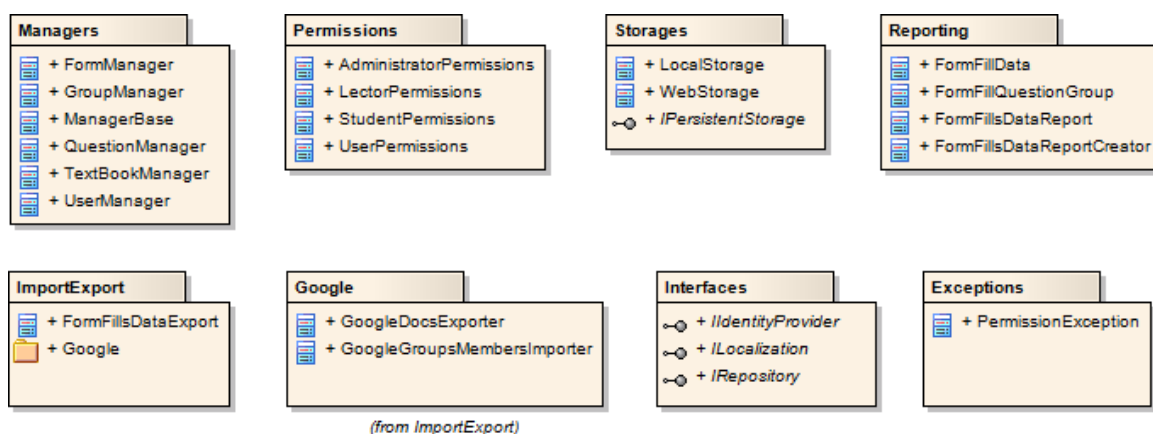
**Storages** V tomto balíčku jsou umístěny třídy týkající se ukládacích prostorů. Lokální pro odkládání dat např. při exportu a **WebStorage** pro zapouzdření přístupu k datům v databázi ve webovém prostředí.

**Reporting** Balíček sloužící k reportování výsledků vyplněných formulářů.

**ImportExport** V ImportExport balíčku se nacházejí třídy pro import dat do systému a export ze systému. Vnořený balíček Google se věnuje předávání dat s Google službami.

**Interfaces** Obecný balíček pro různá rozhraní, která není nutné dělit do větších kontextů.

**Exceptions** Zde se nacházejí specifické výjimky systému.



Obrázek 3.4: Balíčky business vrstvy obsahující různorodé komponenty pro obstarávání veškeré aplikační logiky

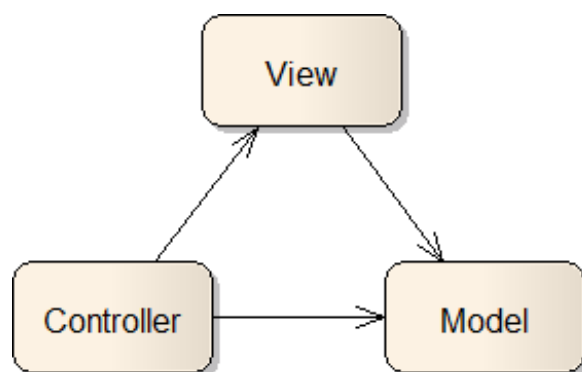
### 3.4 ASP.NET MVC vrstva

Tato vrstva staví na modelu MVC (na Obr. 3.5) tvořícím podstatu technologie ASP.NET MVC. Základem je zde tedy view, controller a model.

Pohledy (views) jsou odpovědné za tvorbu grafického rozhraní pro uživatele, které tvoří na základě dat získaných od controlleru. v ASP standartně odpovídá jméno pohledu jménu metody controlleru, kterou je pohled vyvoláván. Většina pohledů sestává z několika parciálních (částečných) pohledů, které jsou využívány na několika místech.

Controllery zde zastávají funkci prostředníka mezi pohledy a business vrstvou. Obstarávají pro pohledy relevantní data a při vyvolání akce uživatelem tuto akci propagují do aplikační logiky.

Model z MVC trojice v této vrstvě není zcela standartní, jak jsem již přiblížil ve 3.1. Podle MVC terminologie pod pojem model patří má business i datová vrstva. Proto v této vrstvě mluvím spíše o Data modelu (což je vidět na diagramu 3.1), který obsahuje datové entity určené k předávání mezi pohledem a controllerem. Jejich smyslem je mimo jiné odstínit závislost pohledů na konkrétních vlastnostech datových typů datové vrstvy a vytvořit další vlastnosti plněné aplikační logikou.



Obrázek 3.5: Architektura MVC – View, Controller, Model a znázornění komunikace mezi nimi

## 3.5 Technologie a frameworky

### 3.5.1 C# ASP.NET MVC 4.0

Tento projekt vyvíjím v jazyce C# ve vývojovém prostředí Visual Studio 2010. Protože se jedná o webové orientované řešení, je využito ASP.NET [9]. Ze tří<sup>3</sup> přístupů pro tvorbu s ASP.NET používám MVC verze 4.0.

### 3.5.2 ADO.NET Entity Framework

Entity Framework je ORM<sup>4</sup> framework z rodiny ADO.NET. Slouží pro práci s daty v relačním databázovém uložišti. Vývoj zcela odlišuje například od ručního psaní SQL dotazů a připojování k DB. Framework je velice jednoduše a rychle použitelný. Doporučil bych jeho využití v malých a středně velkých aplikacích, kde není až tak velký počet dotazů na DB a tyto dotazy nemusí být optimalizovány do posledního klíčového slova. Více o využití EF v tomto projektu uvádím v sekci 4.1.

### 3.5.3 HTML, CSS

Jelikož je projekt webová aplikace, základ uživatelského rozhraní tvoří HTML a CSS. Kaskádové styly jsou optimalizovány pro prohlížeč Google Chrome, avšak kontrolovány byly i ve Firefoxu a Opeře. Dodatek k nynějšímu stavu viz. závěr 7.1.

### 3.5.4 Javascript, jQuery

Pro zajištění dodatečné interaktivity a operací aplikace na straně uživatele je využito Javascriptu a nadstavbového frameworku jQuery [14]. Pro zjednodušení práce s Cookies

<sup>3</sup> Jsou to přístupy Web Pages, Web Forms a MVC. Více lze nalézt například na [10]

<sup>4</sup> Objektově relační mapování, technika pro převod dat mezi relační databází a objekty objektově orientovaného programovacího jazyka

(využité např. pro uchování seznamu srolovaných otázek při editaci formuláře) je využito pluginu Cookie [2].

### 3.5.5 MarkItUp a Wiki .NET Parser

Formátování skript (sekce 2.8) je řešeno pomocí Wiki značkování. Aby měl uživatel značkování co nejvíce ulehčené, je využito editoru MarkItUp! [6]. Je to velice přizpůsobitelný a flexibilní editor jakéhokoli kódu. v tomto případě upravený pro úpravu Wiki syntaxe.

Pro převod stránky s Wiki značkováním do HTML je využit Wiki .NET Parser II [4]. Jeho použití je velice jednoduché, stačí zavolat metodu `ConvertToHTML` s parametrem Wiki-formátovaného textu. Vracenou hodnotou je pak připravený HTML kód.

### 3.5.6 Google Data API

K exportu výsledků formuláře do Google Formulářů (FP11) se předpokládá využití Google API. Pro tento účel existuje knihovna Google Data API .NET Client Library [11], která obsahuje funkcionalitu pro volání Google API z prostředí .NET.

## Kapitola 4

# Implementace

### 4.1 Datová vrstva

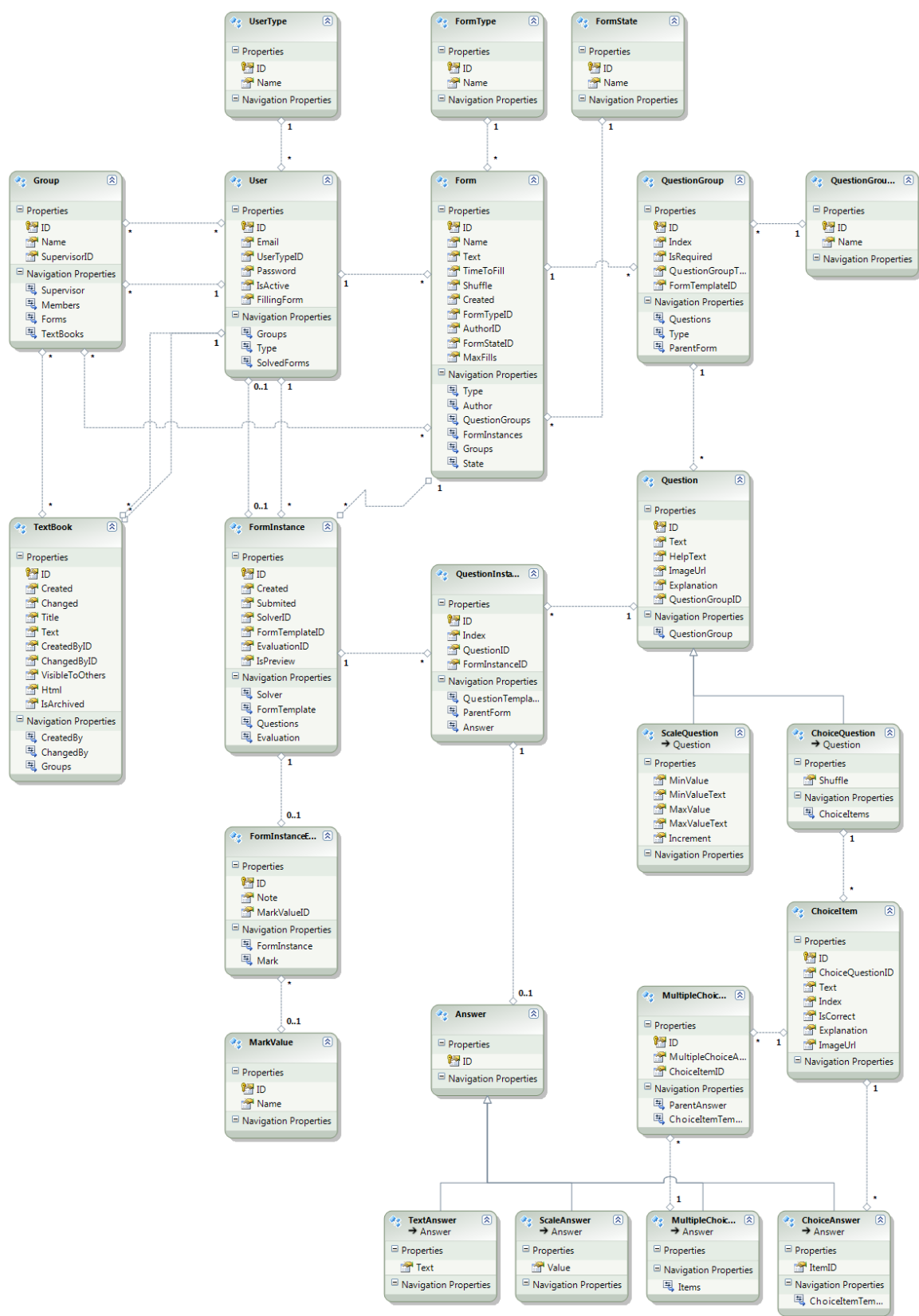
Pro implementaci datové vrstvy aplikace jsem použil Entity Framework (viz. 3.5.2). Zvolil jsem přístup Model First Development. Podle mne je to výborný kompromis mezi Code First a Database First řízením vývoji. Mám díky němu plnou kontrolu nad strukturou vytvářené databáze a zároveň nad generovanými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

Model First Development je jedním z postupů při vývoji datové vrstvy. Zaplňuje mezeru mezi Code First a Database First vývoji. Při Code First postupu se nejdříve napíše kód všech datových entit a Entity Framework poté automaticky vygeneruje strukturu potřebné databáze. Naopak při Database First vývoji se nejdříve vytvoří kompletní databáze, na jejímž základě vygeneruje EF odpovídající třídy v kódu. Při Model First se nevytváří ani struktura DB, ani samotný kód, ale vše se zakresluje do diagramu Entity Data Modelu (4.1). Vývojář je tedy po celou dobu v roli architekta systému, protože se vlastně stále pohybuje v UML diagramu. Přesto má možnost, díky množství nastavení, ovlivňovat strukturu generované DB i generovaného kódu tříd.

Všechny možné nedostatky skriptu databáze lze vyřešit přidáním databázového projektu a importem vygenerovaného skriptu do něho. 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 mne každé nasazení databáze stojí pouhé dvě kliknutí myši.

Potřebné úpravy generovaného kódu tříd lze jednoduše provést pomocí částečných (partial) 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 nutné provádět „ručně“ v business vrstvě. Je to hlavně z důvodu zamezení vzniku různých 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 – diagram celé datové vrstvy vytvořený pomocí Entity Frameworku

## 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 v systému

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 řešeno 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 MITM<sup>1</sup>), 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`, která má za následek zobrazení varovného hlášení (Obr. 4.2). Běžně by se uživatel na takovou url vůbec dostat neměl, protože stejná politika je implementována ve views, čímž adresa takové url není uživateli dostupná.

---

<sup>1</sup>Man In The Middle – útočník, který odposlouchává a případně mění komunikaci



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

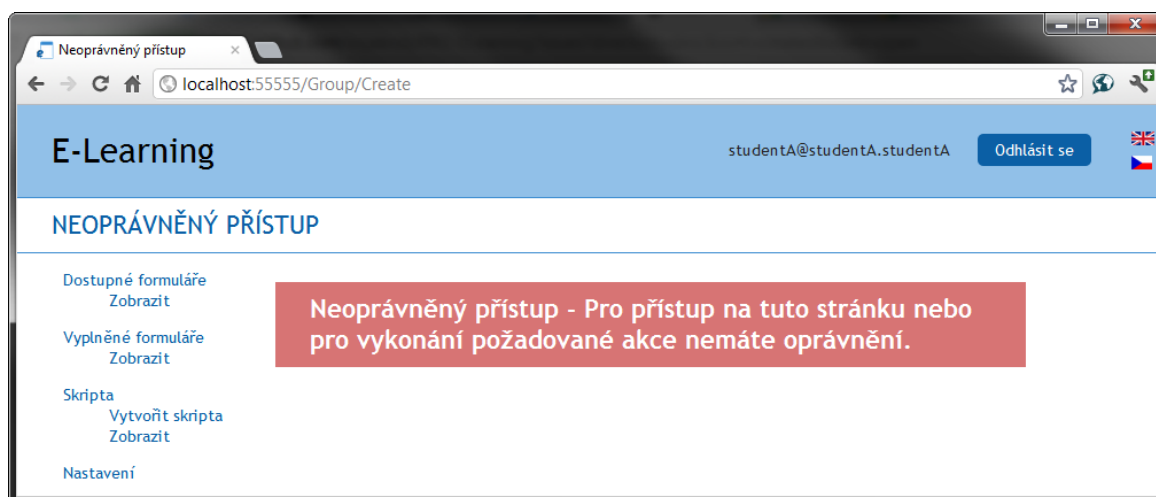
Zdrojový kód 4.2: Použití atributu `AuthorizeUserType` na akci controlleru

Druhým stupněm je kontrola oprávnění v samotné business vrstvě. Všechny operace nad daty zde obsahují ověření práva na vykonání pomocí `UserPermissions`, testují zda požadovanou položku uživatel vytvořil či patří do jeho skupiny apod. 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 oprávnění přihlášeného uživatele v business vrstvě – při neoprávněném přístupu je v aplikaci vyvolána chyba typu `PermissionException`



Obrázek 4.2: Ukázka neoprávněného přístupu – takto vypadá sdělení o neoprávněném přístupu na stránku či provádění neoprávněné akce následované vznikem výjimky `PermissionException`

## Kapitola 5

# Testování

### 5.1 Unit testy

Ověření funkčnosti hlavních tříd a jejich metod je provedeno pomocí Unit testů. Protože se testují hlavně metody pro práci s daty, je vytvořena testovací databáze speciálně pro tyto účely, aby nedocházelo k možným nekonzistentním datům a zásahům do databáze ostré.

Pro implementaci Unit testů využívám nástrojů integrovaných ve Visual Studiu.

### 5.2 Selenium testy

Pomocí testovacího systému Selenium [18] 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. Testy jsem zapojil do testů podporovaných Visual Studiem. To přineslo obrovskou výhodu, protože testy stačí spustit stejně jako Unit testy a tím 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<sup>1</sup> testy, takže je ho třeba naportovat pro testy integrované do VS.

---

<sup>1</sup>Open source framework pro testování v .NET [17]

## 5.3 Výkonové testy

### 5.3.1 Aplikace

V současné době vychází výkonové testování aplikace pouze z pozorování a zpětné odezvy od uživatelů při pilotním provozu, popsáným níže v 5.4.

Při pilotním provozu systém obsahoval 46 aktivních uživatelů, z toho bylo vždy cca 15 studentů online. Jejich aktivitou vzniklo 79 instancí formulářů s 538 instancemi otázek a 518 odpověďmi. Při takovémto nízkém počtu dat nelze očekávat snížení výkonu dotazy do databáze, ale lze pozorovat celkovou plynulost aplikace při dotazech na server a běhu skriptů na klientském počítači. Z mého pohledu nedocházelo k žádným problémům a potvrdili mi to i uživatelé, na škále 1-5 (jako známkování ve škole) je výsledný průměr hodnocení 1,3.

### 5.3.2 Databáze

Pro testování a nalezení neoptimalizovaných a zbytečných dotazů do databáze je dobré využít SQL Server Profiler [19], který je součástí SQL Server Performance Tools. Při použití Entity Frameworku se stává, že dotazy do databáze nejsou vždy zcela optimální. Důvodem je, že EF neporozumí přesně záměru programátora (což třeba ani nemůže), a například namísto jednoho dotazu s JOIN pošle zvlášť dotaz pro každý záznam v tabulce. Díky profileru lze tyto dotazy jednoduše najít a odpovědný kód upravit.

V tomto projektu jsem zatím SQL profileru nevyužil, ale je to jedna z věcí, které se budu v brzké době věnovat. V tuto chvíli není totiž optimalizace dotazů do DB stěžejní. Na jednotlivých stránkách aplikace teoreticky není nic, co by mohlo i v nejhorším případě systém viditelným způsobem zpomalit, což se potvrdilo při pilotním testování. Ovšem lepší je všechny dotazy optimalizovat ještě před tím, než se bude systém dále rozrůstat a plnit velkým množstvím dat.

## 5.4 Pilotní provoz

Pilotní provoz probíhal za asistence mých studentů gymnázia (věk 16-18 let). Někteří vyzkoušeli připravený naučný test s tématem Základy programování a někteří vyplnili ostrý test ze základů počítačových sítí, který jim byl následně oznámkován. Nakonec byli všichni požádáni o vyplnění dotazníku ohledně názoru na systém.

Studentům byly poskytnuty informace o adrese portálu a přihlašovacích údajích. Poté byli studenti stručně instruováni aby vyplnili dostupný test. Myslím, že se v systému bez problému zorientovali a dostali se ke spuštění vyplňování formuláře. Zde ale nastal menší problém, který vyžadoval mou asistenci. Firefox totiž blokoval otevírané okno s formulářem, takže bylo nutné říci studentům o nutnosti odblokovat otevírání oken a jak to skrze nenápadnou a podivnou nabídku provést. Samotné vyplňování formulářů už bylo zcela bezproblémové.

Po vyplnění ostrého testu se studentům líbilo, že si ihned mohli prohlédnout svůj výsledek (po zobrazení vyplněného testu viděli správné odpovědi vyznačené zeleně a špatné červeně).

Výsledky dotazníku jsou myslím pro systém dobré. Z výsledků vyplývá, že je potřeba zapracovat na designu stránek, zatímco uživatelskou přívětivost (zde pro roli studenta) se, myslím, podařilo splnit. Dotazník byl vyplněn 37 studenty, otázky s jejich výsledky jsou zde:

**Otázka:** Jak hodnotíte uživatelskou přívětivost? Tzn. jak se v portálu vyznáte, jak jsou jasné názvy položek menu, jednoduchost stránek, ...

**Možné odpovědi:** 1, 2, 3, 4, 5 (hodnocení jako ve škole)

**Výsledek:** průměr 1,84

**Otázka:** Jak hodnotíte design stránek a testů?

**Možné odpovědi:** 1, 2, 3, 4, 5 (hodnocení jako ve škole)

**Výsledek:** průměr 2,4

**Otázka:** Jak hodnotíte rychlost aplikace? Tzn. jak rychle se načítají jednotlivé stránky, jak je plynulá celková práce s aplikací.

**Možné odpovědi:** 1, 2, 3, 4, 5 (hodnocení jako ve škole)

**Výsledek:** průměr 1,38

**Otázka:** Chtěl/a byste tento e-learningový systém používat při výuce?

**Možné odpovědi:** Ano, Nevím, Ne

**Výsledek:** Ano 62%, Nevím 35%, Ne 3%

**Otázka:** Máte radši papírové nebo elektronické testy/písemky?

**Možné odpovědi:** Papírové, Elektronické

**Výsledek:** Papírové 32%, Elektronické 68%

#### 5.4.1 Testovací portál

Testovací portál je dostupný na adrese <http://elearning.robertsojak.cz/>, kde bude dostupný nejméně do obhajoby této práce. Přihlašovací údaje jsou následující:

- Administrátor – E-mail **admin@admin.admin**, heslo **Pass123**
- Lektor – E-mail **lector@lector.lector**, heslo **Pass123**
- Student ve skupině lektora – E-mail **studentL@studentL.studentL**, heslo **Pass123**

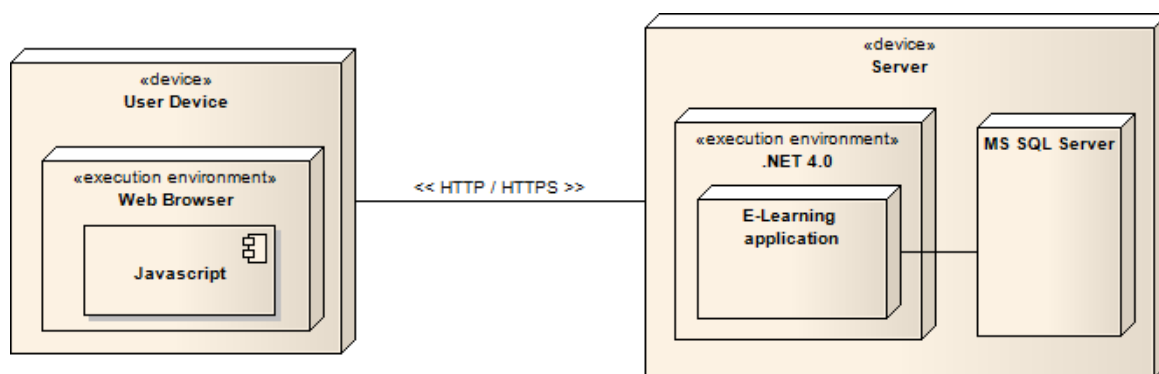
## Kapitola 6

# 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 – klientské zařízení s webovým prohlížečem a server s .NET frameworkem a SQL databází

# Kapitola 7

## Závěr

V této bakalářské práci jsem vytvořil funkční základ e-learningového portálu s některými specifickými požadavky zadavatele (2.2.1). Systém byl poté úspěšně otestován v pilotním provozu (5.4), ve kterém byl shledán provozuschopným.

Na základě pilotního provozu se domnívám, že systém splňuje specifikované požadavky (2.2) a očekávání. Jedním z hlavních požadavků bylo jednoduché ovládání, ozn. jako NP1. Jak vyplynulo z ohlasu uživatelů, byl splněn na lepší dvojku (známkování jako ve škole). Osobně si myslím, že hodnocení bylo negativně ovlivněno chováním otevíraného okna formuláře v použitém prohlížeči (popsáno v sekci 5.4), protože pro úspěšné vyplnění dostupných formulářů a prohlédnutí si výsledků nebyly nutné žádné další pokyny. Nepříjemnost s otevíranými okny se tedy pokusím vyřešit a poté už, myslím, nic nebrání dostat se na jedničku.

Při práci na tomto projektu jsem se naučil vyvíjet s použitím technologie ASP.NET MVC (3.5.1), což pro mne byla zajímavá změna oproti ASP.NET Windows Forms. Také jsem si osvojil využívání Selenium testování (5.2), které bude zcela jistě výborným pomocníkem ve všech mých starých i nových webově orientovaných aplikacích.

### 7.1 Další vývoj

S dalším vývojem systému se chystám použít metodu dog fooding<sup>1</sup>. Během užívání při výuce se budou jistě stále rodit nové požadavky a nápady, které se objevovaly již i během analýzy a vývoje. Také je pravděpodobné, že při ostrém provozu nebudou některá chování systému přesně odpovídat chování požadovanému, které při analýze bez testování v provozu na větším počtu uživatelů nelze nikdy předpovídat.

Zcela určitě bude žádoucí například vylepšení rozhraní pro vytváření formulářů technologií AJAX [8] a dalšími klientskými skripty, aby nebylo nutné při každé úpravě otázky potvrzovat změnu stiskem tlačítka. Také se chci věnovat notifikacím pomocí e-mailu, které mohou být využity k informování o novém dostupném formuláři či udělené známce.

V implementaci systému v tuto chvíli chybí řádné logování vzniklých chyb. To je důležité pro každou aplikaci, protože ať se programátoři a testéři snaží sebevíc, nikdy nebude zcela bez chyb. Jejich zaznamenávání alespoň vede k neustálému dohledu.

---

<sup>1</sup>Metoda vývoje, kdy programátoři používají vyvíjený software v roli koncového uživatele

Při dalším vývoji také počítám s vylepšením celkového designu portálu a úpravou některých problematických míst v kaskádních stylech. Hlavně v Internet Exploreru totiž některé stránky nevypadají zcela správně. Optimalizace pro všechny prohlížeče přesahovala rozsah dosavadního vývoje a je mnohem produktivnější svěřit kódování vzhledu stránek specializovanému vývojáři.

## 7.2 Možná rozšíření

Zde uvádím výčet možných rozšíření a nápadů, které by v budoucnu mohly systém vylepšit a rozšířit na víceúčelový e-learningový portál.

### 7.2.1 Formuláře

- Přidání vlastnosti Generovaný počet otázek – instance formuláře bude vygenerována jen s určitým počtem náhodně zvolených otázek
- Možnost zadání stupnice pro automatické známkování – instance formuláře by tak mohla být automaticky oznámkována ihned po vyplnění (např. podle procentuální úspěšnosti)
- Možnost nastavit rozsah platnosti formuláře – formulář bude dostupný pouze v zadaném časovém rozsahu
- Umožnění vyplnění formuláře i neregistrovanému uživateli – tím by vznikla možnost využít systém například pro výzkum pomocí anonymních dotazníků

### 7.2.2 Otázky

- Možnost přiřadit k textu otázky a vysvětlení správné odpovědi obrázek
- Nastavení bodového ohodnocení za správnou/špatnou odpověď

### 7.2.3 Skripta

- Možnost nahrání obrázku nebo jiného materiálu, např. prezentace

### 7.2.4 Reportování výsledků formuláře

Reportování celkových výsledků formuláře je v současné době možné pomocí externích nástrojů po exportu do Google Dokumentů. Určitě by bylo dobré generovat reporty také přímo v aplikaci. Solidní základ je již v implementaci připravený, stačí jen na základě dostupných údajů report vygenerovat a zobrazit.

### 7.2.5 E-mailové notifikace

O e-mailových notifikacích jsem se již zmínil výše. Jistě by velkou měrou přispěly k efektivitě práce. Studentovi by odpadla nutnost neustále sledovat změny v dostupných testech, udělených známkách a další.



# 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] HARTL, K. Cookie — jQuery plugin pro práci s Cookies. Dostupné z: <<https://github.com/carhartl/jquery-cookie>>. stav k 4.5.2012.
- [3] HUNKA, J. *Studentova berlička I*. ČVUT FEL, 2007. Bakalářská práce.
- [4] KÖHLER, R. Wiki .NET Parser II — převodník Wiki do HTML. Dostupné z: <<http://www.codeproject.com/Articles/74160/Wiki-NET-Parser-C>>. stav k 6.4.2012.
- [5] KRAML, F. *Studentova berlička III — Generátor testů a písemek II*. ČVUT FEL, 2010. Bakalářská práce.
- [6] SALVAT, J. MarkItUp! — univerzální značkovací jQuery editor. Dostupné z: <<http://markitup.jaysalvat.com/home/>>. stav k 17.5.2012.
- [7] TORDEUR, K. ASP.NET MVC and Unity. Dostupné z: <<http://blogs.realdolmen.com/experts/2011/04/23/asp-net-mvc-and-unity/>>. poslední návštěva 16.4.2012.
- [8] web:AJAX. AJAX — technologie pro vývoj interaktivních webových aplikací. Dostupné z: <[http://en.wikipedia.org/wiki/Ajax\\_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming))>. stav k 22.5.2012.
- [9] web:ASP. ASP.NET — Oficiální stránky. Dostupné z: <<http://www.asp.net/>>. poslední návštěva 22.5.2012.
- [10] web:ASPInfo. ASP.NET — Informace o možnostech vývoje. Dostupné z: <<http://www.asp.net/get-started>>. poslední návštěva 22.5.2012.
- [11] web:GoogleAPI. Google Data API .NET Client Library – .NET knihovna pro využití Google API. Dostupné z: <<http://code.google.com/p/google-gdata/>>. poslední návštěva 17.5.2012.
- [12] web:GoogleDocs. Google Dokumenty — hlavní stránka. Dostupné z: <<http://docs.google.com/>>. poslední návštěva 14.5.2012.
- [13] web:GoogleGroups. Google Skupiny — hlavní stránka. Dostupné z: <<http://groups.google.com/>>. poslední návštěva 14.5.2012.

- [14] web:jQuery. jQuery — JavaScript knihovna rapidně rozšiřující jeho funkcionalitu. Dostupné z: <<http://jquery.com/>>. stav k 12.4.2012.
- [15] web:Moodle. Portál věnovaný systému Moodle. Dostupné z: <<http://moodle.org/>>. poslední návštěva 14.5.2012.
- [16] web:MoodleOtazky. Výčet typů otázek Moodle. Dostupné z: <[http://docs.moodle.org/22/en/images\\_en/2/23/Manage\\_question\\_types.png](http://docs.moodle.org/22/en/images_en/2/23/Manage_question_types.png)>. stav k 12.5.2012.
- [17] web:NUnit. NUnit — Open source framework pro testování v .NET. Dostupné z: <<http://www.nunit.org/>>. návštěva 23.5.2012.
- [18] web:Selenium. Selenium — balík nástrojů pro automatizaci testů skrze webové prohlížeče. Dostupné z: <<http://seleniumhq.org/>>. poslední návštěva 15.5.2012.
- [19] web:SQLProfiler. SQL Server Profiler — nástroj pro sledování dotazů do databáze. Dostupné z: <[http://technet.microsoft.com/cs-cz/library/ms181091\(v=sql.100\).aspx](http://technet.microsoft.com/cs-cz/library/ms181091(v=sql.100).aspx)>. stav k 20.5.2012.

## 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

**EF** Entity Framework

**UML** Unified Modeling Language

**DB** Databáze

**ORM** Object/Relational Mapping

**SQL** Structured Query Language

**HTML** HyperText Markup Language

**CSS** Cascading Style Sheets

**FTP** File Transfer Protocol

**AJAX** Asynchronous JavaScript and XML

**XML** Extensible Markup Language

⋮

# Příloha B

## UML

### B.1 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

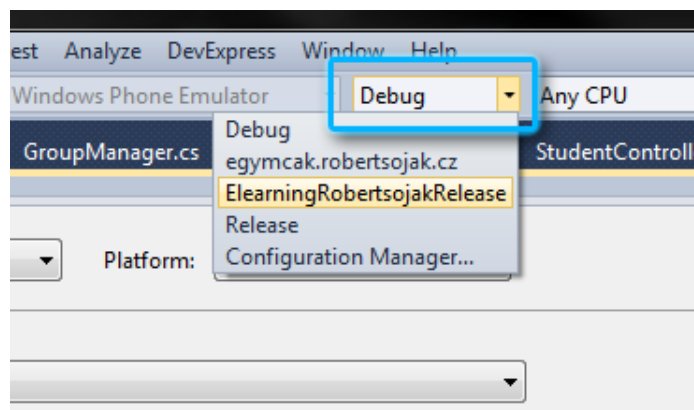
---

## Příloha C

# Instalační příručka

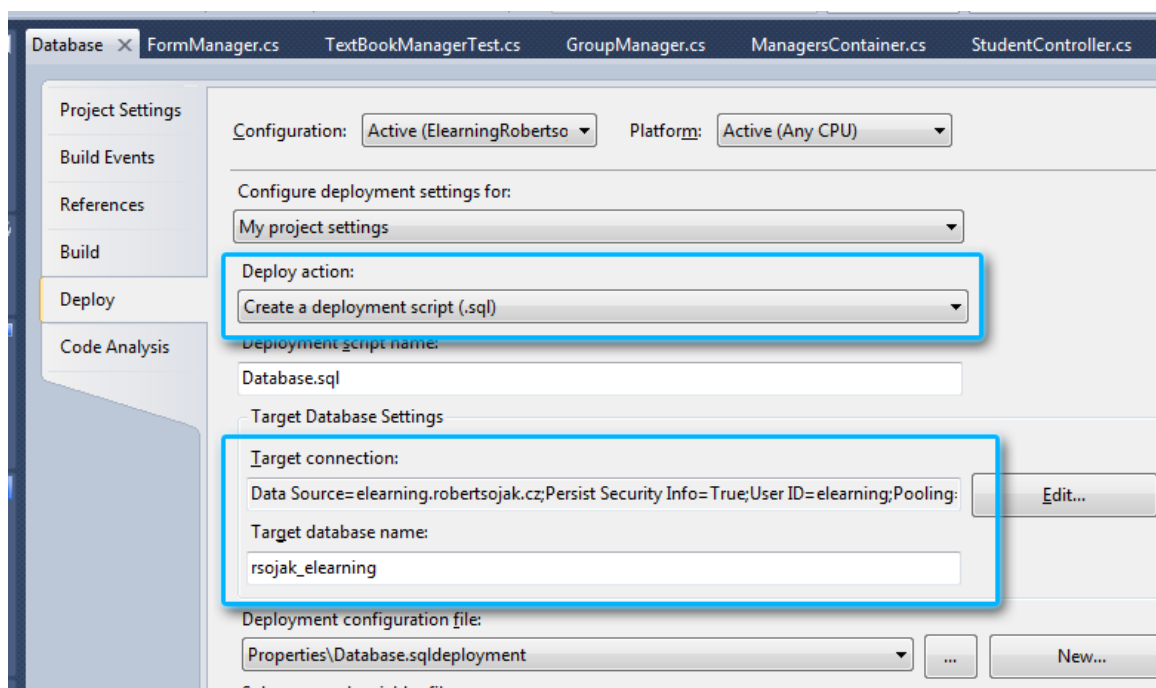
Zde uvádím popis instalace (nasazení) aktuální verze aplikace asi nejjednodušším způsobem, a to za využití Visual Studia. Pro úspěšné nasazení je třeba mít k dispozici aplikační a databázový server, jak je uvedeno v kapitole 6.

Nejdříve je třeba zvolit konfiguraci pro zveřejňování, tzn. Release nebo některou z mnou předpřipravených. Na obrázku C.1 je zachycena volba konfigurace pro nasazení na ukázkový server projektu (5.4.1).



Obrázek C.1: Volba konfigurace pro nasazení na ukázkový server v prostředí Visual Studia

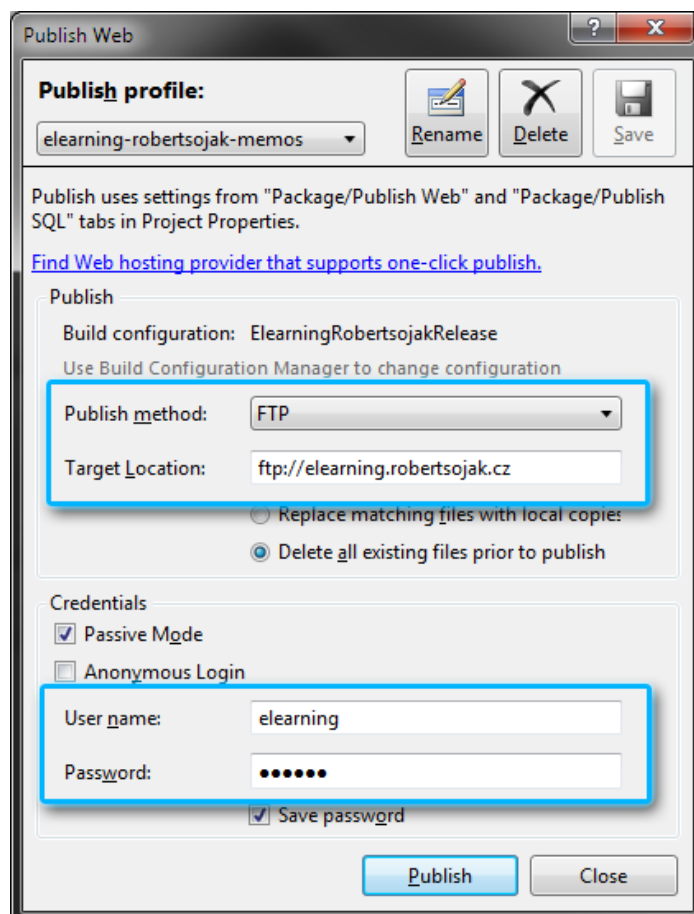
Pro nasazení databáze je třeba nejprve zkontrolovat nastavení databázového projektu. Akce pro Deployment by měla být nastavena na vytvoření skriptu a automatické vytvoření databáze, jak je vidět na obrázku C.2. v poli *Target connection* by měly být vyplněny údaje nutné k připojení k požadovanému DB serveru a v poli *Target database name* jméno vytvářené databáze. Po kontrole údajů stačí jen vybrat v menu *Build* → *Deploy Database* a celá databáze i s počátečními daty bude automaticky nasazena.



Obrázek C.2: Stránka pro nastavení databázového projektu s vyznačeným nastavením, které vyžaduje pozornost

Nasazení samotného portálu je stejně jednoduché. Po vybrání projektu *ELearning* je v menu *Build* zpřístupněna akce *Publish ELearning*. Vyvolané dialogové okno je vidět na obrázku C.3. k nasazení je využito metody *FTP*, která nahrává aplikační soubory přes FTP spojení na zadanou adresu. Po vyplnění přístupových údajů a stisku tlačítka *Publish* dojde k nasazení aplikace.





Obrázek C.3: Dialog pro publikování aplikace s vyznačenou volbou metody publikace a přístupových údajů

## Příloha D

# Unity Application Block tutorial

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

Dependency injection je návrhový vzor pro napojení závislostí třídy zvenčí, což je opak k tomu, když své závislosti řeší třída sama. K předávání těchto závislostí se využívá specializované třídy, což umožňuje při záměně konkrétní implementace provést výměnu pouze na jednom místě. Konkrétní objekty se závislé třídě většinou předávají v konstruktoru nebo pomocí veřejných vlastností.

Pro využití Unity Application Block je nejdříve nutné nainstalovat do projektu Unity NuGet balíček (v menu Visual Studia 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 D.1: Předpis rozhraní `IUnityContainerAccessor` sloužící pro přístup k Unity bloku

Vytvořený interface poté lze přidat do implementace třídy aplikace `MvcApplication`, která je umístěna v souboru `Global.asax.cs`. Způsob implementace je patrný z tohoto kódu:

```
public class MvcApplication : System.Web.HttpApplication, ←
    IUnityContainerAccessor
{
    public static IUnityContainer UnityContainer
    {
        get { return _unityContainer; }
    }
    private static IUnityContainer _unityContainer;
    ...
}
```

```

protected void Application_Start()
{
    ...
    InitUnity();
}
...
private static void InitUnity()
{
    var unityContainerFactory = new UnityContainerFactory();
    _unityContainer = unityContainerFactory.CreateContainer();
    ControllerBuilder.Current.SetControllerFactory(typeof(←
        UnityControllerFactory));
}
...
IUnityContainer IUnityContainerAccessor.UnityContainer
{
    get { return _unityContainer; }
}
}

```

Zdrojový kód D.2: Ukázka implementace `IUnityContainerAccessor` ve třídě `MvcApplication` zajišťuje globální dostupnost Unity bloku

Dále je třeba vytvořit potomka třídy `DefaultControllerFactory` s přetíženou metodou `CreateController` a `GetControllerInstance`, který nahradí základní controller factory [7]. Právě ten bude zajišťovat dependency injection pro každý controller aplikace. v metodě `CreateController` je využito instance aplikace k získání instance `UnityContainer`:

```

public class UnityControllerFactory : DefaultControllerFactory
{
    protected IUnityContainer _unityContainer;
    ...
    public override IController CreateController(System.Web.Routing.←
        RequestContext requestContext, string controllerName)
    {
        IUnityContainerAccessor accessor = requestContext.HttpContext.←
            ApplicationInstance as IUnityContainerAccessor;
        _unityContainer = accessor.UnityContainer;

        return base.CreateController(requestContext, controllerName);
    }

    protected override IController GetControllerInstance(System.Web.←
        Routing.RequestContext requestContext, Type controllerType)
    {
        if (controllerType == null)
            return null;

        return _unityContainer.Resolve(controllerType) as IController;
    }
}

```

```
}
```

Zdrojový kód D.3: Třída `UnityControllerFactory`, která nahradí výchozí controller factory

Posledním krokem je vytvoření samotné `UnityContainerFactory`. v té je třeba provést registraci jednotlivých konkrétních implementací požadovaných tříd a rozhraní pro injekci. Následující kód ukazuje příklad takové registrace:

```
public class UnityContainerFactory
{
    ...
    public virtual IUnityContainer CreateContainer()
    {
        UnityContainer result = new UnityContainer();
        // Registers injection of class WebAuthenticationContext in
        // place of interface IAuthenticationContext
        result.RegisterType<IAuthenticationContext, ←
            WebAuthenticationContext>();

        return result;
    }
    ...
}
```

Zdrojový kód D.4: Příklad registrace rozhraní pro injekci v `UnityContainerFactory`, třídy lze registrovat stejným způsobem



























Tím je celý Unity blok zanesen do aplikace a připraven k použití. Příklad použití v controlleru uvádím v následující ukázce:

```
public class BaseController : Controller
{
    [Dependency]
    public IAuthenticationContext AuthenticationContext { get; set; }
    ...
}
```

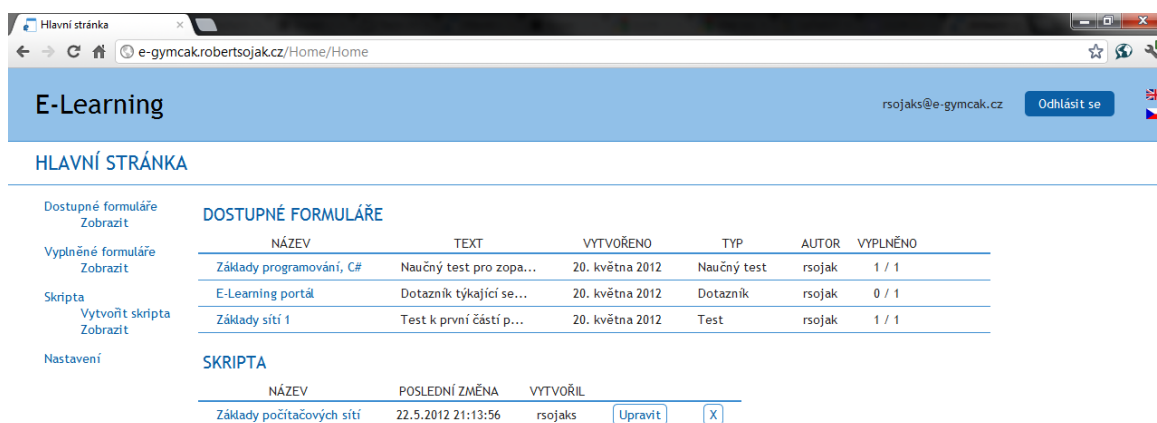
Zdrojový kód D.5: Příklad použití dependency injection v controlleru pro injekci implementace rozhraní `IAuthenticationContext`

## Příloha E

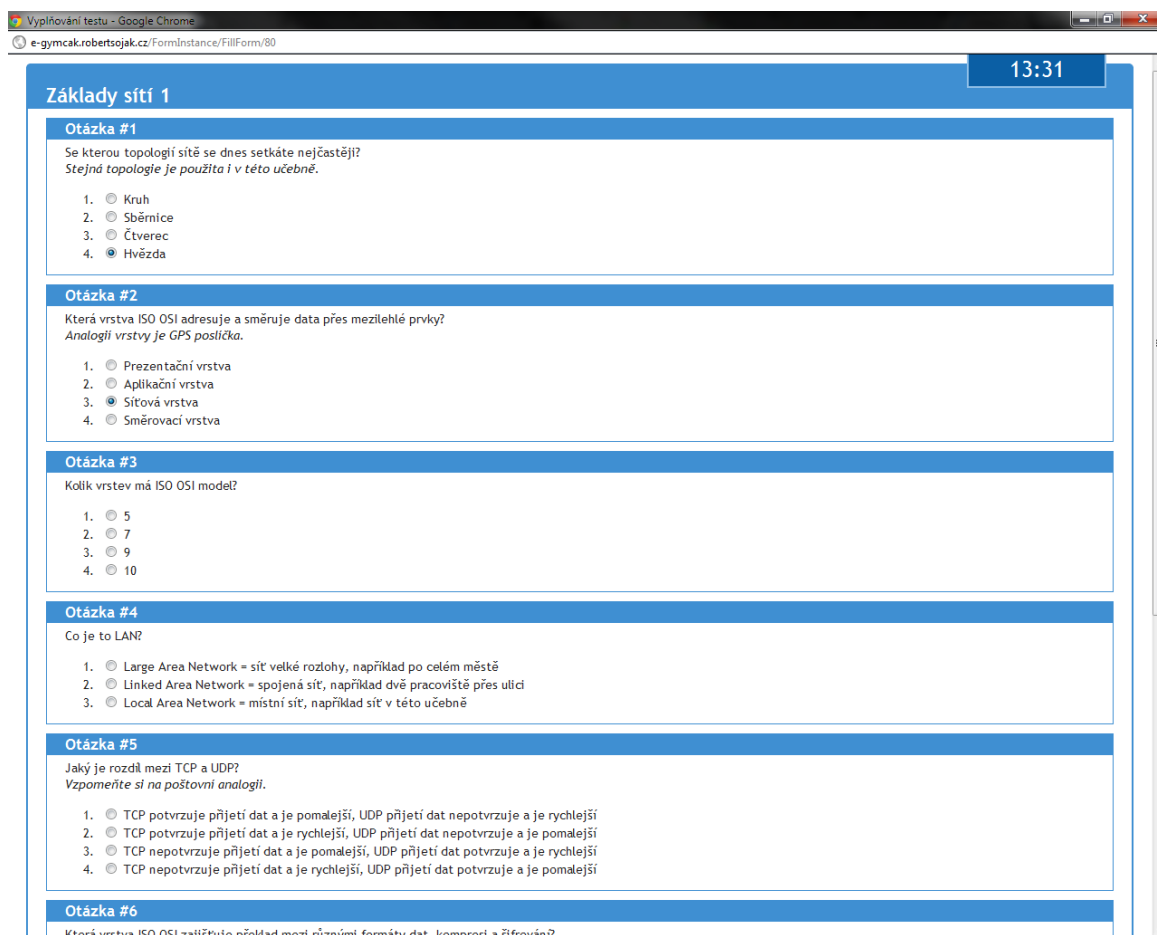
### Obrazová příloha

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	 ↑ ↓	<a href="#">Delete</a>	
 Calculated Simple	0	No database	Numerical	 ↑ ↓	<a href="#">Delete</a>	

Obrázek E.1: Typy otázek v systému Moodle [16]



Obrázek E.2: Screenshot aplikace – Hlavní stránka studenta



Obrázek E.3: Screenshot aplikace – Vyplňování testu

Wyplňování testu - Google Chrome  
e-gymcak.robertsojak.cz/FormInstance/FillForm/81

12:43

### Základy programování, C#

**Otázka #1**  
Kolik stojí základní verze Visual Studio (Express edice), ve které můžeme programovat své programy?

- ☒ Je zdarma, stačí se bezplatně registrovat
- ☐ Pro studenty asi 399 Kč
- ☐ 399 Kč pro všechny
- ☐ 19 999 Kč pro všechny

**Otázka #2**  
Jak vypisujeme text do konzole?

- ☐ Write.ToConsole(text);
- ☒ Console.WriteLine(text);
- ☐ string text = Console.ReadLine();
- ☐ Console.WriteLine(text);

Vypsat je anglicky Write. Naopak přečíst je Read.

**Otázka #3**  
Co je vlastně výsledný program, který můžeme poslat do světa?

- ☒ Soubor s příponou .exe
- ☐ Soubor s příponou .cs
- ☐ Složka s projektem, který lze otevřít ve Visual Studiu

Jistěže .exe - spustitelný program. Najdeme ho ve složce Debug (testovací verze) či Release (verze pro uživatele).

**Otázka #4**  
Jak v C# vytvoříme celočíselnou proměnnou "x" s počáteční hodnotou 0?

- ☐ float x = 0;
- ☐ string x = 0;
- ☒ int x = 0;
- ☐ float x = 0;
- ☐ int x = 1;

Celočíselná proměnná se značí "int", z anglického slova integer. Float je proměnná pro desetinná čísla, string pro řetězec (text). Nezapomínejte na středník za každým příkazem.

**Otázka #5**  
Co znamená v C# "a != b"?

- ☒ Testování, zda se proměnná a nerovná proměnné b
- ☐ Testování, zda se proměnná a rovná proměnné b
- ☐ Říká programu, aby si dal pozor na proměnnou a a b
- ☐ Přikazuje programu, aby do proměnné a uložil proměnnou b

Znakem "!" v C# negujeme (obracíme) logickou hodnotu výrazu.

Obrázek E.4: Screenshot aplikace – Vyplňování naučného testu

The screenshot shows a web browser window with the URL `e-gymcak.robertsojak.cz/Form/CreateEditQuestions/1`. The page has a blue header with the text "E-Learning" and a user email `rsojak@e-gymcak.cz` with a login button "Odhlásit se". Below the header, there is a sidebar on the left with a "TEST" section and a list of navigation links: "Formuláře", "Skripta", "Studenti", "Skupiny", and "Nastavení", each with sub-links for "Vytvořit" and "Zobrazit".

The main content area is titled "Základy sítí 1" and includes a "Test" tab, a question count of 8, a time limit of 15 minutes, and buttons for "Upravit", "Náhled", and "Tisk". There are also buttons for "Rozbalit všechny otázky" and "Sbalit všechny otázky".

The test content is organized into a table with columns for question number, type, question text, and actions. The first question is a multiple-choice question about network topologies. The second question is a text question about "token". Below the text question, there is a "Nápověda" section with a text box and a "Volby" section with a list of options and a "Zamíchat" checkbox. At the bottom, there is a "Nová otázka" section with a dropdown menu and a "Přidat" button.

#	Volba	Text	Nápověda
#1	Volba	Se kterou topologií sítě se dnes setkáte nejčastěji?	
#2	Volba	Co je to "token"?	
#3	Volba	Které tvrzení se NETÝKÁ fyzické vrstvy ISO OSI?	
#4	Volba	Která vrstva ISO OSI adresuje a směřuje data přes mezilehlé prvky?	
#5	Volba	Jaký je rozdíl mezi TCP a UDP?	
#6	Volba	Která vrstva ISO OSI zajišťuje překlad mezi různými formáty dat, kompresi a šifr...	
#7	Volba	Co je to LAN?	
#8	Volba	Kolik vrstev má ISO OSI model?	

Obrázek E.5: Screenshot aplikace – Tvorba formuláře lektorem



Základy sítí 1		Jméno <input type="text"/>
<b>Otázka #1</b>		
Co je to LAN?		
<ol style="list-style-type: none"><li><input type="radio"/> Local Area Network = místní síť, například síť v této učebně</li><li><input type="radio"/> Linked Area Network = spojená síť, například dvě pracoviště přes ulici</li><li><input type="radio"/> Large Area Network = síť velké rozlohy, například po celém městě</li></ol>		
<b>Otázka #2</b>		
Co je to "token"?		
Vzpomeňte si na obrázek na tabuli.		
<ol style="list-style-type: none"><li><input type="radio"/> Zdobený dřevěný kůl (sloup) náboženského významu. Vyskytuje se u domorodých kmenů, dnes hlavně v Severní Americe, Austrálii a v Africe.</li><li><input type="radio"/> Jiné označení pro balíčky dat proudící po síti.</li><li><input type="radio"/> Hra na PlayStation 1, 2, 3 a PlayStation Portable.</li><li><input type="radio"/> Speciální zpráva, kterou si předávají počítače umístěné v kruhové síti. Kdo má token, může komunikovat.</li></ol>		
<b>Otázka #3</b>		
Jaký je rozdíl mezi TCP a UDP?		
Vzpomeňte si na poštovní analogii.		
<ol style="list-style-type: none"><li><input type="radio"/> TCP nepotvrzuje přijetí dat a je pomalejší, UDP přijetí dat potvrzuje a je rychlejší</li><li><input type="radio"/> TCP potvrzuje přijetí dat a je rychlejší, UDP přijetí dat nepotvrzuje a je pomalejší</li><li><input type="radio"/> TCP nepotvrzuje přijetí dat a je rychlejší, UDP přijetí dat potvrzuje a je pomalejší</li><li><input type="radio"/> TCP potvrzuje přijetí dat a je pomalejší, UDP přijetí dat nepotvrzuje a je rychlejší</li></ol>		
<b>Otázka #4</b>		
Kolik vrstev má ISO OSI model?		
<ol style="list-style-type: none"><li><input type="radio"/> 5</li><li><input type="radio"/> 7</li><li><input type="radio"/> 9</li><li><input type="radio"/> 10</li></ol>		
<b>Otázka #5</b>		
Které tvrzení se TÝKÁ fyzické vrstvy ISO OSI?		
Vzpomeňte si na analogii vrstvy.		
<ol style="list-style-type: none"><li><input type="radio"/> Překlad mezi různými formáty dat</li><li><input type="radio"/> Neví co přenáší</li><li><input type="radio"/> Detekuje a opravuje chyby</li></ol>		

Obrázek E.6: Ukázka tisku papírové verze formuláře

## Příloha F

# Obsah příloženého CD

[bin] – adresář s aplikací určenou k distribuci

[doc] – dokumentace ke zdrojovému kódu aplikace

[help] – adresář s uživatelským manuálem

[src] – zdrojový kód aplikace pro prostředí Visual Studio 2010

[text] – elektronická verze tohoto dokumentu a jeho zdrojové soubory

readme.txt – popis instalace aplikace a obsah CD