Automatické vyhodnocování testů

Automatic Processing Of Tests

Bc. Pavel Meluzín

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Pavel MELUZÍN**
Osobní číslo:	**A06660**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Automatické vyhodnocování testů**

Zásady pro vypracování:

1. **Navrhněte a vytvořte software pro automatické digitální vyhodnocování a vytváření testů splňující podmínky tohoto zadání níže.**
2. **Vyberte si programové prostředí pro realizaci zadání a proveďte stručný literární průzkum o tomto programovém prostředí, ve kterém budete software realizovat.**
3. **Na testy se bude odpovídat formou volby správného písmene tak, že pod písmenem správné odpovědi na papíře musí dotyčný zakřížkovat kolonku tvaru čtverce. V případě změny volby odpovědi se kolonka celá vytmaví a zatrhne se kolonka nová.**
4. **Na papíře budou vytištěny vodící znaky pro snadnější zaměření otázky, oblasti pro digitální zpracování a určení verze testu, maximální počet otázek je konstantní.**
5. **Autor testu vyplní své jméno a příjmení a přidělené číslo studenta. Krom toho se podepíše na test v části, která nebude určena pro digitální zpracování. Velikost obrázku, ve kterém bude skenovaný test, bude konstantní.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **ŽÁRA, Jiří. Moderní počítačová grafika. 1. vyd. Brno : Computer Press, a.s., 2004. 609 s. ISBN 80-251-0454-0.**
2. **MARTIŠEK, Dalibor. Matematické principy grafických systémů. 1. vyd. Brno : Littera, 2002. 278 s. ISBN 808-57-6319-2.**
3. **NAVRÁTIL, Pavel. Počítačová grafika a multimédia. 1. vyd. Kralice na Hané : Computer Media a.s., 2007. 112 s. ISBN 80-86686-77-9 .**
4. **NAVRÁTIL, Pavel. 50 příkladů v počítačové grafice. 1. vyd. Kralice na Hané : Computer Press, a.s., 2007. 50 s., CD-ROM. ISBN 978-80-86686-79-0.**
5. **ŽÁRA, Jiří. Počítačová grafika – principy a algoritmy. 1. vyd. Praha : Grada Publishing a.s., 1992. 446 s. ISBN 8085623005**
6. **NAGEL, Christian , et al. C sharp 2008 Programujeme profesionalně. Radek Hylmar; David Dirga. 1. vyd. [s.l.] : Computer press, 2008. 2 sv. (100, 772 s.). ISBN 978-80-251-2401-7.**
7. **SHARP, John. Microsoft Visual C sharp 2008 : Krok za krokem. 1. vyd. Brno : Computer Press, a.s., 2008. 592 s., 1 CD-ROM. ISBN 978-80-251-2027-9.**

Vedoucí diplomové práce: **Ing. Karel Perůtka, Ph.D.**
Ústav řízení procesů

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.
①

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

V diplomové práce se zabývám návrhem a vytvořením softwarového produktu, který bude jeho uživatelům sloužit pro vytváření a následnému vyhodnocování testů studentů z opticky zpracovaného formuláře testu.

V první teoretické části práce jsou objasněny všechny důležité pojmy z oblastí digitalizace obrazu, databázových systémů, zvolených technologií a vývojových prostředí, které jsem pro vlastní realizaci cíle práce využil.

Ve druhé praktické části jsou uvedeny poznatky o návrhu a realizaci řešení. Je zde podrobněji popsána vytvořená aplikace a její funkce. Čtenář zde nalezne také několik vybraných krátkých ukázek kódu, aby se případně mohl lépe zorientovat v celé struktuře projektu a ve zdrojových souborech, které jsou přílohou této práce.

Klíčová slova:

obraz, digitalizace, databáze, MS .NET Framework, MS Visual Studio, MS SQL Server, C#, XAML, LINQ, SQL

ABSTRACT

In my master thesis I employ myself in designing and developing of a software product, whose purpose is to enable their users create and evaluate student’s tests using optically processed test form.

In the first part of the thesis there are clarified all the important issues of digital image processing, database systems, technologies and development environments employed for the realization of the thesis objective.

In the second, practical part, there are stated findings following the practical realization of the objective. The created application and its functions are more widely described here. Several code examples are shown here to help a reader understand the whole structure of the project and its source codes, which make up the appendix of the thesis.

Keywords:

image, digitizing, database, MS .NET Framework, MS Visual Studio, MS SQL Server, C#, XAML, LINQ, SQL

Zde bych chtěl poděkovat mému vedoucímu diplomové práce panu Ing. Karlu Perůtkovi Ph.D. za všechny rady a připomínky, které mi při tvorbě této diplomové práce poskytoval. Dále bych zde chtěl poděkovat své rodině a kamarádům za podporu při psaní této práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

V současné době se již delší dobu běžně setkáváme s elektronickým zpracováním nejrůznějších typů dokumentů, formulářů a tiskopisů například na přepážkách pošt, bank a nejrůznějších úřadů. Díky celkovému rozvoji informačních technologií a výpočetní techniky, rostoucímu výkonu osobních počítačů a operačních systémů při snižování jejich ceny, došlo v posledních desetiletích k velkému rozvoji a rozšíření počítačové grafiky a s ní souvisejícímu oboru zpracování obrazu dokumentů - Document Image Processing (DIP).

Hlavním cílem této diplomové práce je navrhnout a vytvořit software, který bude umožňovat jeho uživateli - pedagogovi vytvářet písemné testy pro své studenty v papírové podobě. Vyhodnocení výsledků těchto testů se bude realizovat na principu optického zpracování obrazu dokumentu. Vstupními daty pro aplikaci budou soubory obrázků získané naskenováním vyplněného formuláře testu studenta. Takto získaná obrazová data budou programem zpracovávána vyhodnocována a informace o výsledku testu konkrétního studenta budou uloženy do databáze. Vytvořenou aplikaci můžeme svou funkcionalitou zařadit do kategorie software určeného pro zpracování obrazových dat a databázových aplikací.

Diplomová práci se dělí na teoretickou a praktickou část. V první kapitole teoretické části se věnuji všem důležitým pojmům z oblastí počítačové grafiky, reprezentace a digitalizace obrazu, vstupním a výstupním grafickým zařízením. V druhé kapitole teoretické části jsou popsány databázové systémy a teorie návrhu relačních databází. Třetí kapitola teoretické části seznamuje čtenáře s možnostmi vývoje aplikací pro operační systémy Windows, s platformou .NET Framework a s programovacím jazykem C#. Poslední čtvrtá teoretická kapitola je věnována vývojovým prostředí MS Visual Studio 2010 a MS SQL Server 2008, které jsem si pro vytvoření aplikace zvolil.

Úvodní kapitola praktické části se zabývá stručně analýzou zásad zadání práce a návrhu možného řešení. V další dvou kapitolách je podrobněji popsána struktura navržené databáze a navržený formulář testu a možnosti jeho vyplnění. Další kapitola se podrobněji zabývá použitými technologiemi .NET Framework a jinými, strukturou souborů aplikace a její instalací. Poslední kapitola praktické části popisuje funkce vytvořené aplikace a poslouží jako návod pro práci s aplikací.

Teoretická i praktická část byla doplněna krátkými příklady a celkově byla koncipována tak, aby si čtenář udělal lepší představu o vytvořené aplikaci a dokázal se zorientovat i ve zdrojových kódech a celé struktuře souborů projektu aplikace, které jsou přílohou práce na disku CD-ROM.

I. TEORETICKÁ ČÁST

1 DIGITALIZACE A ZPRACOVÁNÍ OBRAZU

1.1 Obraz v počítačové grafice

Definice obrazu se může lišit podle své aplikační oblasti a většina disciplín používá definice, které nejlépe vyhovují jejich potřebám. Obraz můžeme vnímat jako průmět reálného světa na sítnici oka, jako fotografii, obrazovku počítače, obrázek na papíře, či odraz v nehybné vodní hladině. Pro formální vymezení pak použijeme matematický model, kterým je spojitá funkce dvou proměnných tzv. *obrazová funkce* [Gonz87]:

$$z = f(x, y) \tag{1}$$

Definiční obor obrazové funkce můžeme zapsat jako kartézský součin dvou spojitých intervalů z oboru reálných čísel, které vymezují rozsah obrazu. Obrazová funkce realizuje zobrazení:

$$f : \left(\left\langle x_{\min}, x_{\max} \right\rangle \times \left\langle y_{\min,} y_{\max} \right\rangle\right) \to H \tag{2}$$

Reálná čísla x, y, kde $x : x_{\min} \geq x \geq x_{\max}$ a $y : y_{\min} \geq y \geq y_{\max}$ jsou souřadnice bodu ve dvou rozměrech, v nichž funkce nabývá nějaké hodnoty z z oboru hodnot, tj. $z \in H$. Hodnotou obrazové funkce může být jediné reálné číslo například jas, intenzita červené barvy,atp. Obvykle je to ale více hodnot například červená, zelená, modrá složka v barevném modelu RGB . Obecně funkce nabývá hodnot zapsaných jako uspořádaná n-tice údajů $z = [z_1, z_2, ..., z_n]$, což můžeme zapsat ve tvaru:

$$f : \left(\left\langle x_{\min}, x_{\max} \right\rangle \times \left\langle y_{\min,} y_{\max} \right\rangle\right) \to \left(H_1 \times H_2 \times ... \times H_n\right) \tag{3}$$

Při práci s obrazem v počítačové grafice máme zřídka k dispozici spojitý definiční obor funkce, ale většinou pracujeme v tzv. rastru, který si můžeme představit jako mřížku v dvojrozměrném prostoru, která je složená z obrazových bodů nebo elementů tzv. pixelů. Termín pixel je zkratkou z anglického picture element. Každý pixel má v rastru svou jednoznačnou adresu, podle které je identifikovatelný a díky které s pixely můžeme provádět požadované operace.[12]

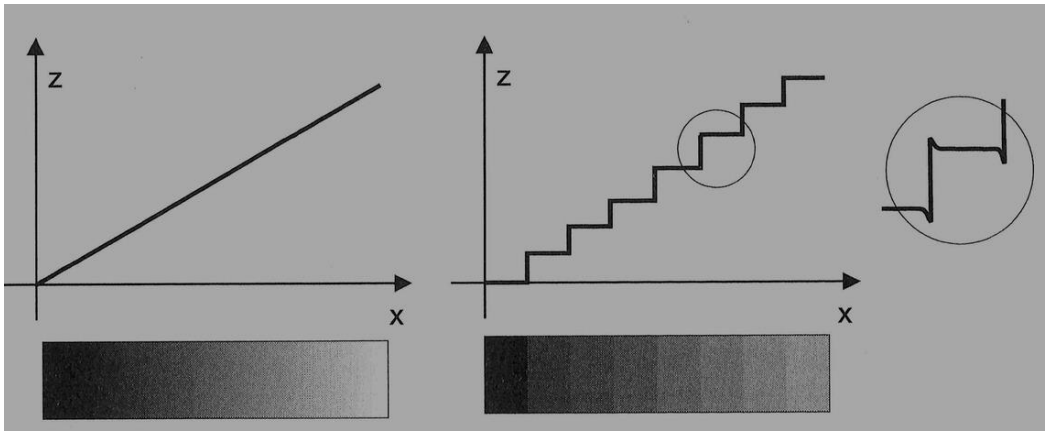
1.2 Digitalizace

V počítačové grafice lze většinu modelů charakterizovat nějakou spojitou funkci, ale obraz, který vidíme na obrazovce našeho monitoru je diskrétní. Základní děj spjatý se získáváním počítačového obrazu, je přechod od spojité funkce f(x,y) k diskrétní funkci $I_{i,j}$ jak v definičním oboru funkce f(x,y) tak v oboru jejích hodnot H. Jako příklad bych uvedl skenování obrázku, nebo jeho zachycení digitálním fotoaparátem nebo videozáznam pořízený digitální videokamerou. Obraz, který je snímán, má teoreticky nekonečný rozsah obrazových hodnot , stejně jako je možné ho přibližovat či vzdalovat téměř neomezeně. Bude však zobrazen v konečném množství pixelů a s konečným množstvím barev. Proces přechodu od spojitého obrazu k diskrétnímu se nazývá digitalizace a odehrává se ve dvou fázích, které se nazývají kvantování (quantization) a vzorkování (sampling).[12] [2]

1.2.1 Kvantování

Principem kvantování je rozdělení oboru hodnot obrazové funkce na intervaly, jimž je následně přidělena jediná zástupná hodnota. Při kvantování proto dochází ke ztrátě informací a tato ztráta se nazývá kvantizační chyba. Působí rušivě a projevuje se například v obraze jako náhlá změna barev nebo nepřirozené hrany. Podle způsobu rozdělení kvantované veličiny pak mluvíme o častěji používaném kvantování uniformním s konstantní délkou intervalu a kvantování neuniformním s proměnnou délkou intervalu. Délka intervalu se zvolí většinou jako průměr celého intervalu, vážený průměr, medián, nebo průměr z okrajů. Způsob výběru zástupné hodnoty závisí na použité aplikaci a požadovaných cílech. [12]

Kvantizační chyba ilustrující náhlou změnu barev a výskyt hran je dokonale ukázána na obrázku níže. Obrázek vznikl skenováním stolním skenerem obrázku z papíru z knihy při nastavení skenování v 256 odstínech šedé barvy a rozlišení 600 ppi a následujícími úpravami změn rozlišení, velikosti a formátů obrázku.



Obr. 1. Kvantizační chyby [12]

Na Obr. 1. si můžeme všimnout , že původně hladký barevný přechod (vlevo) je nahrazen skokovým přechodem(uprostřed). V obraze vznikají hrany, které v původním signálu nebyly přítomné. Lidské vnímání jejich subjektivní intenzitu zesiluje (zakroužkované hrany úplně vpravo).

Pro lepší představu zde uvádím příklad vtažený k následujícímu obrázku Obr. 2.. Počítač pracuje s diskrétními hodnotami. Obrazový snímač je analogový. Kvantování představuje proces, kdy je naměřené veličině přiřazena zástupná diskrétní hodnota. Analogová hodnota

$$f(x) = f(x_0 + i\Delta x) \tag{4}$$

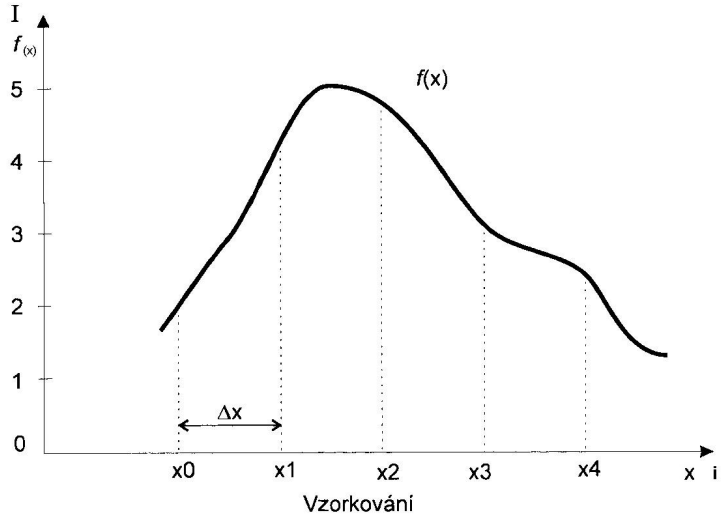
je tedy ze snímače převedena analogově digitálním (A/D) převodníkem na diskrétní hodnotu. Budeme-li uniformě kvantovat tak, že přiřadíme nejbližší celočíselnou hodnotu, bude na obrázku Obr. 2 funkční hodnotě $f(x_0)$ přiřazena úroveň 2, funkční hodnotě $f(x_1)$ úroveň 4, atd. [2]

1.2.2 Vzorkování

Obecně lze říct, že vzorkování signálu je proces jeho diskretizace v časové oblasti. V oblasti počítačové grafiky pod pojmem vzorkování spojité funkce f(x) rozumíme zaznamenávání hodnot tedy vzorků v předem daných intervalech, tak jak je naznačeno následujícím obrázku, kde jednorozměrnou funkci získanou pravidelným vzorkováním budeme označovat I_i a vzdálenost dvou vzorků označíme Δx. Původní spojitá funkce může být definována na libovolném intervalu $x \in \langle x_{0,} x_1 \rangle$. Vzorky budeme indexovat následujícím způsobem:

$$I_i = f(x_0 + i\Delta x), i = 0,1,2,... \quad (5)$$

V počítačové grafice je nejběžnější bodové vzorkování (point sampling), kdy je hodnota vzorku sejmuta v jediném bodě. Lze se však setkat ještě s plošným vzorkováním (area sampling), kdy je zaznamenána reprezentace hodnot celého vzorkovaného intervalu Δx a hodnota vzorku se určí měřením ze všech hodnot , například jako jejich průměr. Díky vyšší výpočetní náročnosti se v praxi plošné vzorkování aproximuje několika body a tato metoda se nazývá supersampling. [12]



Obr. 2. Vzorkování [2]

Převrácená hodnota vzdálenosti jednotlivých vzorků udává frekvenci vzorkování. Mějme rozteč obrazových bodů(vzorkovací interval) Δx , vzorkovací frekvence f_s potom vyjde

$$f_s = 1/\Delta x \quad (6)$$

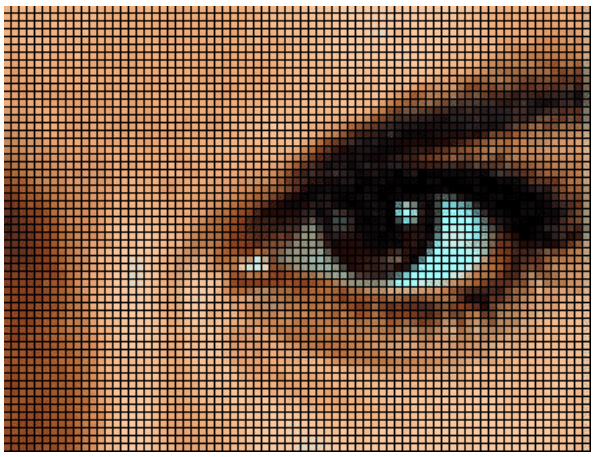
a její jednotkou je počet vzorků na jednotku času [Hz] či na jednotku vzdáleností [dpi]. Je zřejmé, že čím bude vyšší vzorkovací frekvence, tj. čím větší bude počet dosažených vzorků, v případě obrazu jeho rozlišení, tím bude paměťová náročnost reprezentace vyšší. Platí zde tzv. *Shannononův teorém*, který říká, že signál (obraz) lze rekonstruovat beze ztráty informace, pokud je vzorkován s frekvencí minimálně dvojnásobnou než je maximální frekvence signálu obsaženého v obraze. Z toho plyne že: [2]

$$f_{\max} < 1/(2\Delta x) \quad (7)$$

1.3 Reprezentace rastrového obrazu

Obrázky v počítačové grafice se dají rozdělit do dvou skupin. Jedna z nich se nazývá vektorová, kdy je obrázek poskládán z grafických primitiv, jaké jsou body, křivky nebo tvary, které jsou zadány pomocí matematických rovnic vektory. Druhá z nich se nazývá rastrová. Obraz rastrové grafiky, digitální obraz, nebo bitová mapa (bitmapa, pixmapa) je datový soubor nebo struktura reprezentována obecně dvourozměrnou maticí bodu (pixelů). Barva každého pixelu je individuálně definovaná. Kvalita a výsledná velikost rastrového obrázku v souboru je určena úplným množstvím obrazových bodů a množstvím informací (bitů) o každém obrazovém bodu nazývaném *hloubka barvy*.

Pokud je každý pixel popsán jediným bitem, říkáme, že je monochromatický (může obsahovat dvě barvy). Pokud mám obraz 256 barev, používá se pro jejich reprezentaci tzv. indexový mód. U takového obrazu nereprezentuje hodnota pixelu přímo barvu, ale je ukazatelem do barevné palety. Dalším případem je obraz, ve kterém je každý pixel reprezentován třemi barvami (tzv. true color) nejčastěji v barevném modelu RGB. Takový obraz obsahuje přímo barevné hodnoty jednotlivých pixelů. S obrazem jsou tedy spojeny tři palety, pro každý barevný kanál jedna. [2] [7] [12]



Obr. 3. Reprezentace rastrového obrazu [14]

1.3.1 Komprese rastrových obrazů

Nevýhodou rastrových obrazů je vysoká paměťová náročnost, proto se zavadí komprese obrazu, která slouží ke zmenšení objemu dat uložených na disku a může být bezeztrátová nebo ztrátová. Bezeztrátová komprese je taková kdy, komprimační metoda zachová v komprimovaném obraze všechnu informaci o obraze tak, že při zpětné

rekonstrukci dostaneme původní obraz. Ztrátová komprese je založena na nedokonalosti lidského zraku. Ztrátovou kompresi lze docílit většího kompresního poměru za cenu toho, že pokud obrázek zkomprimujeme a zpětně rekonstruujeme, nedostaneme původní hodnoty originálního obrázku a dojde ke ztrátě informace. [2]

1.3.2 Kompresní metody a rastrové grafické formáty

Existuje velké množství kompresních metod. Většina z nich je spojena s konkrétním grafickým formátem. Příčin rozsáhlosti kompresních metod a grafických formátů je několik, formáty odrážejí historický vývoj, konkrétní aplikační požadavky, vazby na programy, technické důvody, metody komprese. Zde jsou příklady základních kompresních metod a formáty.

Tab. 1. Příklady základních kompresních metod a grafických formátů [7]

Název kompresní metody	Zkratka	Druh komprese	Příklad formátu
Run Lenght encoding	RLE	bezeztrátová	PCX
Huffmanovo kódování	CCITT	bezeztrátová	TIFF
Lempel - Ziv - Welch	LZW	bezeztrátová	GIF, PNG, ZIP
Diskrétní kosinová transformace	DCT	ztrátová	JPEG

1.4 Vstupní a výstupní grafické zařízení

V této kapitole se jen stručně zmíním o nezbytných vstupních a výstupních grafických zařízení pro potřeby této diplomové práce jako je tiskárna a skener a uvedu zde jejich principy, parametry a základní rozdělení.

1.4.1 Tiskárna

Tiskárna (počítačová tiskárna) je výstupní grafické zařízení počítače, které slouží k přenosu dat uložených v elektronické podobě většinou na papír nebo také na jiné fyzické médium například průhledné fólie nebo kompaktní disky. Mnoho tiskáren je primárně používáno jako periférie počítače, který slouží jako zdroj dokumentu, může ale samozřejmě fungovat samostatně a data k tisku jí mohou být dodávaná z různých rozhraní např. USB či Bluetooth. Síťové tiskárny mohou mít vestavěná různá síťová rozhraní a jsou dostupné uživatelům počítačové sítě. Tiskárny můžeme rozdělovat do několik skupin podle různých kritérií. Základní rozdělení se odvozuje od způsobu přenosu barvy na médium. Podle

tiskových technologií tedy máme například tiskárny laserové (tonerové), inkoustové(termické, piezoelektrické, voskové), jehličkové, termální, plotry, tiskárny dopadu (tiskárny dopisní kvality a tiskárny matice) a jiné.

Základním parametrem u tiskárny je kvalita tisku, se kterou je spojena veličina DPI. DPI je zkratkou anglického Dot Per Inch, což přeloženo do češtiny znamená počet bodů na palec. Pokud se bavíme o výstupním zařízení jako je tiskárna, pak jde o to, jak jemně jsou poskládány jednotlivé obrazové body na médium. Což znamená, že čím vyšší je rozlišení, tím menší body s vyšší hustotou jsou vytvářeny. Pokud se budeme bavit o nějakém zařízení provádějícím digitalizaci obrazu viz. další kapitola skener, udává rozlišení v DPI to, jak jemně je rozlišována předloha a kolik obrazové informace je získáváno. Čím vyšší je rozlišení, tím menší detaily jsou v předloze rozlišovány a zaznamenávány. [7] [14] [15]

1.4.2 Skener

Skener (z anglického scanner) je vstupní grafické hardwarové zařízení, které slouží ke snímání neboli převedení fyzické 2D nebo 3D předlohy do digitální podoby pro další zpracování a využití většinou pomocí počítače. Skenery je možné dělit podle několika kritérií. Skenery můžeme například rozdělovat na skenery ruční a stolní nebo na černobílé a barevné. Pro potřeby této diplomové práce se budeme dále bavit o 2D skenování. Skener pracuje na principu digitalizace. V souvislosti s termínem kvantování se u skenerů a některých snímacích zařízení používá termín barevná hloubka jako parametr skeneru, např. u barevných skenerů v minulosti mohla být 24 bitů při 8 bitech na každou barvu, 36 bitů, tj. 12 bitů na každou barvu. Lepší skenery mají dnes 48 bitovou barevnou hloubku, tj. 16bitů na každou barvu. Barevná hloubka je dána kvalitou snímače a rozsahem A/D převodníku. Druhým důležitým parametrem u skeneru je rozlišovací schopnost neboli DPI, které dokáže scanner rozlišit. U dnešních běžných scannerů se optické rozlišení pohybuje od 300 x 300 dpi přes 600 x 1200 dpi a víc. Pomocí softwarové interpolace je možné dosáhnout vyššího rozlišení. U profesionálních a speciálních skenerů je možné vidět optické rozlišení 4800 x 4800 dpi či vyšší. Současné i ty nejlevnější skenovací zařízení bývají dodávány většinou se softwarovým vybavením, které disponuje základními nástroji z kategorie OCR, OMR a nabízí uživateli různé možnosti nastavení parametrů a předvoleb skenování nejrůznějších dokumentů. [2][7]

2 DATABÁZE A DATABÁZOVÉ SYSTÉMY

Databází (neboli datovou základnou) rozumíme soubor dat, který slouží pro popis reálného světa. Databáze ale nezahrnuje jen vlastní data, ale jejich vlastnosti, strukturu uspořádání dat, vztahy mezi daty a jejich omezení. Databáze je tedy určitá uspořádaná množina informací. S tímto těsně souvisí pojem databázový systém, který představuje množinu komponent určených pro definování, konstrukci a manipulaci s databází. Z širšího pohledu vnímáme databázový systém jako soubor technických prostředků, vlastních dat a programového vybavení neboli systému řízení báze dat (SŘBD, anglicky Data Base Management System - DBMS), který tvoří programovou vrstvu řešící operace databáze.

Systémy řízení báze dat uspořádávají a strukturují data tak, aby k nim uživatelé a aplikační programy mohli získat přístup a dále s nimi pracovat. Strukturu dat a techniky zpřístupňování dat, které poskytují specifické SŘBD, označujeme jako jejich datový model. Datový model vymezuje "charakter" jak SŘBD, tak i aplikací. [3]

Z hlediska datových modelů můžeme rozdělit databáze na několik základních typů:

- Hierarchická databáze
- Síťová databáze
- Relační databáze
- Objektová databáze
- Objektově relační databáze

2.1 Relační databáze

Popisovat jednotlivé datové modely by bylo nad rámec této práce, dále stručně popíši jen model relační databáze, který se mé práce týká.

Relační model dat byl navržen v 70. letech E. F. Coddem a byl založen na matematickém pojmu relačních množin, kde mají data pravidelnou strukturu a ukládají se v tabulkách a vztahy mezi daty se realizují pomocí operací relační algebry. (Přesnou definici relačního modelu uvedl E. F.Codd v roce 1985 v tzv. dvanácti Coddových pravidlech.) [3]

2.1.1 Tabulky relačních databází

Základem relačních databází jsou tedy databázové tabulky. Jejich sloupce se nazývají atributy nebo pole, řádky tabulky jsou pak záznamy. Data jsou v tabulkách uspořádaná jako kolekce tabulek:

- Každá tabulka má název tabulky, který ji jedinečně identifikuje.
- Každá tabulka má jeden nebo více sloupců, které jsou uspořádány ve specifickém pořadí směrem zleva doprava. Atributy mají určen svůj konkrétní datový typ - doménu.
- Každá tabulka má libovolný počet řádků (nemusí mít žádný), každý řádek obsahuje v každém sloupci jedinou datovou hodnotu. Řádek je řezem přes sloupce tabulky a slouží k vlastnímu uložení dat. Řádky nejsou uspořádané.

Tabulky jsou vzájemně propojeny prostřednictvím dat, která obsahují. Relační datový model používá k reprezentaci těchto relací mezi tabulkami primární a cizí klíče:

- Primární klíč je sloupec nebo kombinace sloupců v tabulce, jejichž hodnota (případně hodnoty) jedinečně identifikují každý řádek tabulky. Tabulka má pouze jeden primární klíč. Pole klíče musí obsahovat hodnotu, tzn. nesmí se zde vyskytovat nedefinovaná prázdná hodnota NULL.
- Cizí klíč je sloupec nebo kombinace sloupců v tabulce, jejichž hodnota (případně hodnoty) jsou hodnotou primárního klíče pro nějakou jinou tabulku. Tabulka může obsahovat více cizích klíčů, které ji spojují s jednou nebo více tabulkami.
- Kombinace primární klíč-cizí klíč vytvoří relaci rodič-potomek mezi tabulkami, které je obsahují. [3]

2.1.2 Relace mezi tabulkami

Relace neboli vztahy v relačním datovém modelu slouží ke svázání dat, která spolu souvisejí a jsou umístěny v různých databázových tabulkách. Rozlišujeme čtyři typy vztahů.

- **Žádný vtah** - mezi daty v různých tabulkách není žádná spojitost.

- **Jedna k jedné (1 : 1) -** používáme, pokud záznamu odpovídá právě jeden záznam v jiné databázové tabulce a naopak. Většinou není pádný důvod, proč takovéto záznamy neumístit do jedné databázové tabulky a proto se tento typ relací většinou využívá pro zpřehlednění rozsáhlých tabulek.
- **Jedna k více (1 : N) -** přiřazuje jednomu záznamu více záznamů z jiné tabulky. Jedná se o nejpoužívanější typ relace, jelikož odpovídá mnoha situacím v reálném životě.
- **Více k více (N : M)** - umožňuje několika záznamům z jedné tabulky přiřadit několik záznamů z tabulky druhé. Tento vztah bývá z praktických důvodů nejčastěji realizován kombinací dvou vztahů 1:N a 1:M, které ukazují do pomocné tzv. vazebné tabulky složené z kombinace obou použitých klíčů. [13] [20]

2.1.3 Normalizace a normální formy

Normalizace je proces dekompozice dat na jednotlivé tabulky a určení vztahů mezi nimi. Normální formy chápeme jako principy a pravidla vedoucí k dobře navrženému datovému modelu. Hlavním cílem je navrhnout databázové tabulky tak, aby obsahovaly minimální počet redundantních dat. Správnost navržení struktur lze ohodnotit některou z následujících normálních forem:

- **Nultá normální forma** - tabulka v nulté normální formě obsahuje alespoň jeden sloupec (atribut), který může obsahovat více druhů hodnot.
- **První normální forma** - tabulka je v první normální formě, pokud všechny sloupce (atributy) nelze dále dělit na části nesoucí nějakou informaci neboli prvky musí být atomické. Jeden sloupec neobsahuje složené hodnoty.
- **Druhá normální forma** - tabulka je v druhé normální formě, pokud obsahuje pouze atributy (sloupce), které jsou závislé na celém klíči.
- **Třetí normální forma** - tabulka je ve třetí normální formě, pokud neexistují žádné závislosti mezi neklíčovými atributy .
- **Čtvrtá normální forma** - tabulka je ve čtvrté normální formě, pokud atributy v ní obsažené popisují pouze jeden fakt nebo jednu souvislost.

- **Pátá normální forma** - tabulka je v páté normální formě, pokud by se přidáním libovolného nového sloupce atributu rozpadla na více tabulek. [13] [20]

2.1.4 Integritní omezení databáze

Pojem integrita dat se týká správnosti a úplnosti dat v databázi, což znamená, že data v ní uložená jsou konzistentní vůči definovaným pravidlům. Lze zadávat pouze data, která vyhovují předem definovaným kriteriím. Jednou z nejdůležitějších rolí relačního SŘBD je nejvyšší možná míra ochrany integrity dat. K zajištění integrity slouží integritní omezení:

- **Entitní integritní omezení** – povinné integritní omezení, které zajišťuje úplnost primárního klíče tabulky; zamezí uložení dat, která neobsahují všechna pole sdružená do klíče, nebo data, jež by v těchto polích byla stejná jako v nějakém jiném, již zapsaném, řádku tabulky.
- **Doménová integritní omezení** – zajišťují dodržování datových typů definovaných u sloupců databázové tabulky.
- **Referenční integritní omezení** – zabývají se vztahy dvou tabulek, kde jejich relace je určena vazbou primárního a cizího klíče.
- **Aktivní referenční integrita** – definuje činnosti, které databázový systém provede, pokud jsou porušena některá pravidla. [20]

2.1.5 Jazyk SQL

Jazyk SQL je nástroj pro organizování, správu a získávání dat uložených v relační databázi. Zkratka SQL znamená strukturovaný dotazovací jazyk (Structured Query Language). Jazyk SQL je základem databázových produktů všech největších světových softwarových firem jako je Microsoft, Oracle, IBM a dalších. Jazyk SQL se používá v kombinaci s procedurálními a objektově orientovanými jazyky. Není to pouhý dotazovací jazyk, nýbrž plnohodnotný nástroj pro veškeré aspekty práce v databázích. [3]

Příkazy SQL se v závislosti na své funkci dělí do kategorií. Podle názoru některých odborníků jsou tyto kategorie buď samostatnými jazyky, nebo jejich částmi. Všechny kategorie jazyka SQL však mají stejnou základní syntaxi a pravidla:

- **Data Definition Language, DDL** - jazyk pro definici dat, podmnožina příkazů SQL, které slouží k vytváření a odstraňování databází, k přidávání a odstraňování tabulek, pohledů na tyto tabulky, jejich mazání, vytváření indexů a k modifikaci všech těchto objektů apod..
- **Data Manipulation Language, DML** - jazyk pro manipulaci s daty podmnožina příkazů, které spolupracují s databázovým systémem a vlastní databází. Příkazy jazyka DML tak zajišťují vyhledávání, vkládání a odstraňování dat.
- **Data Control Language, DCL -** do jazyka DCL patří příkazy SQL, které správcům dovolují řídit přístup k datům v databázi přidělením práv skupinám či jednotlivým uživatelům a používat různá systémová oprávnění SŘBD.
- **Příkazy řízení transakcí** - databázová transakce je sada příkazů, kterou databázový uživatel požaduje zpracovat jako nedělitelnou jednotku. To znamená, že transakce musí být kompletně úspěšná nebo neúspěšná. Příkazy řídící databázové transakce přesně neodpovídají syntaxi příkazů jazyka SQL, ale mají značný vliv na chování těch příkazů SQL, které jsou součástí transakcí. [3] [13]

2.2 Modelování a návrh relačních databází

Plánování a návrh databáze bývá prvním a velice důležitým krokem při vývoji každé databázové aplikace. Relační databázové systémy udržují, zajišťují a využívají relace neboli vztahy mezi uloženými daty. Návrh relační databáze má tyto základní fáze:

- **Analýza systémových požadavků** - rozbor požadavků kladených na cílový systém. Mnohé z požadavků ovlivňují vývoj entitně relačního modelu(E-R model) poměrně zásadním způsobem, zatímco jiné jsou spíše obecnými systémovými požadavky, které se promítají do vlastního vývoje databázové aplikace. Jedním z důležitých úkolů při ER-modelování je proto správně vyčlenit požadavky, které mají skutečný vliv na návrh databáze, a oddělit je od požadavků, definujících funkční elementy systému.
- **Konceptuální návrh databáze** - po dokončení analýzy systémových požadavků se pokračuje takzvaným konceptuálním neboli myšlenkovým návrhem databáze. Aby návrh úplně splnil všechny operační požadavky mohou se vytvořit tyto tři modely databáze:

- Konceptuální model - vysokoúrovňové znázornění uživatelských a operačních požadavků. Pomáhá při procesu formování požadavků a naznačuje, jak data vnímají uživatelé.
- Logický model - podrobné znázornění softwarového modulu, který bude shromaždovat strukturovaná data, který eliminuje redundanci dat a zvyšuje integritu dat. Logické modely ukazují, jak data vnímají vývojáři.
- Fyzický model - podrobný výčet všech tabulek a sloupců databáze naznačující, jak bude server ukládat data.

Pro rozsah této diplomové práce je nejdůležitější a dostačující návrh fyzického modelu, při kterém postupujeme technikami modelování entit a vztahů (entity relationship modeling, E-Rmodeling).

Entity jsou objekty nebo věci, které můžeme blíže popsat určitými charakteristickými vlastnostmi - atributy. Určit správnou kardinalitu relace mezi entitami je nejobtížnějším úkolem v modelování entit a vztahů.

- **Vytvoření databáze** - Po úspěšném vytvoření kompletního přehledného modelu entit, následuje převod vytvořeného modelu do podoby příkazů SQL pro vytvoření správně fungující databáze.

[10]

3 VÝVOJ APLIKACÍ PRO OPERAČNÍ SYSTÉMY MS WINDOWS S PLATFORMOU .NET FRAMEWORK

3.1 Historie a vývoj aplikací pro MS Windows

V současnosti bývá jazyk C# a s ním související prostředí .NET Framework (vyslovujeme Dot Net Framewok) označováno za nejdůležitější a nejrozšířenější novou technologii pro vývojáře za posledních mnoho let. Návrh technologie .NET poskytuje nové vývojové prostředí, ve kterém lze vyvíjet skoro téměř libovolné aplikace určené pro operační systémy MS Windows a C# je relativně nový programovací jazyk, který byl navržen zejména pro spolupráci s technologií .NET. Společnost Microsoft od uvedení .NET Frameworku a jazyka C# očekávala, že zásadně ovlivní, změní a rozšíří způsob psaní programů pro své produkty operační systémů Windows jako nikdy předtím.

K lepšímu porozumění významu technologie .NET zde uvedu několik dalších minulých i současných možností programování pro systémy MS Windows, které se objevily v posledních dvaceti letech. Všechny operační systémy Windows od verze 3.1 z roku 1992 mají ve svém jádru rozhraní Windows API, u kterého přibývalo s novými verzemi mnoho nových funkcí. Win32API nemělo objektový základ a bylo souborem funkcí a datových struktur jazyka C. Technologie Microsoft Foundation Classes MFC měla za úkol Win32API doplnit o objektové prvky. MFC byla založena na objektové formě jazyka C++. Stejně tomu tak bylo u mnoha dalších technologií a platforem, pomocí nichž se vyvíjel systém Windows, jako například u modelu COM (Component Object Model), který vycházel ze služeb OLE (Propojování a vkládání objektů). Jeho nástupce byl DCOM (Distributed COM) a konečně COM+. U vývoje těchto rozhraní Windows se jednalo tedy o rozšiřování rozhraní ale ne o jeho nahrazení. Společnost Microsoft zvolila tento evoluční přístup k softwaru kvůli zpětné kompatibilitě. V průběhu let byla vyvinuta rozsáhlá programová základna pro systémy Windows. Kdyby v každé nové verzi Windows zavedla společnost Microsoft novou technologii, která by neumožnila používat existující programy, nedosáhly by její systémy takového úspěchu. Zpětná kompatibilita byla klíčovou vlastností technologií Windows avšak přinášela i velké nevýhody. Když se takto nějaká technologie vyvíjí a získává nové funkce, postupně se přitom komplikuje. Společnost Microsoft nemohla věčně rozšiřovat stejné vývojové nástroje a jazyky a neustále zvyšovat jejich složitost, aby uspokojila konfliktní požadavky podpory nejnovějšího hardwaru při

udržování zpětné kompatibility. Společnost Microsoft proto chtěla přijít s novou jednoduchou, ale přesto pokročilou sadou jazyků, prostředí a vývojových nástrojů, které vývojářům umožní snadnou tvorbu špičkových programů. Nový začátek pro ní představovala .NET platforma (aplikační rozhraní, API) k programování pro Windows s jazykem C#, který byl od základů navržen tak, aby spolupracoval s technologií .NET a zároveň využil veškerý pokrok ve vývojových prostředích a koncepcích objektově orientovaného programování za posledních dvacet let. Zpětná kompatibilita však zůstala zachována. Existující programy budou fungovat i nadále a technologie .NET byla navržena tak, aby se stávajícím softwarem spolupracovala. Komunikace mezi softwarovými komponentami v systému Windows je v současnosti téměř výhradně založena na modelu COM. S ohledem na tento fakt může technologie .NET poskytnout tzv. obálky kolem existujících komponent COM, aby s nimi mohly komunikovat komponenty .NET. [6] [8]

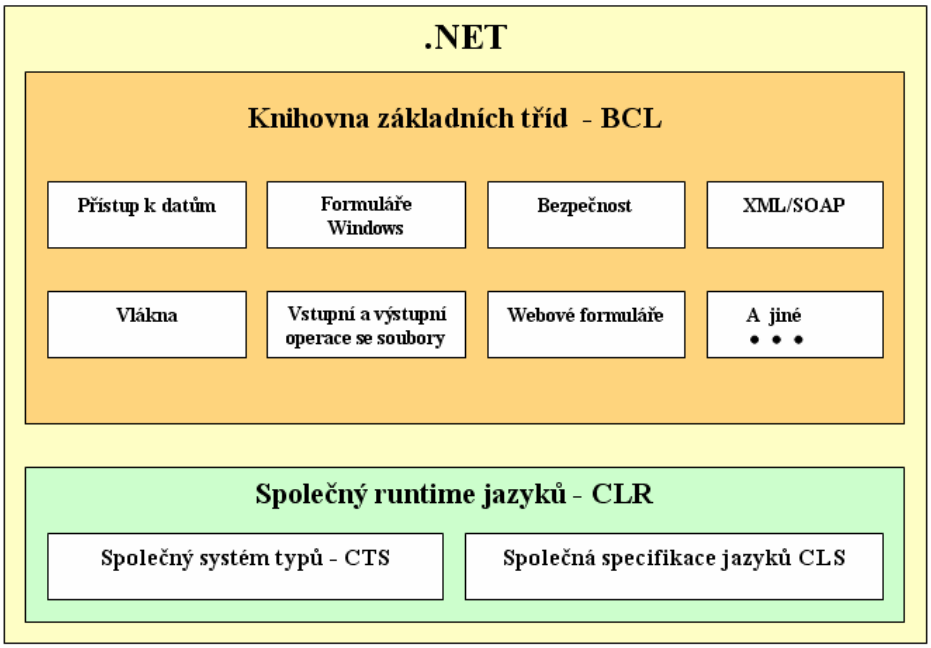
3.2 Technologie Microsoft .NET Framework

Odpověď na otázku: "Co je to vlastně technologie .NET?" nemusí být na první pohled tak úplně jasná, proto by jsme jí mohli popsat přirovnáním k otázce: "Čím je systém Windows pro vývojáře?" Kromě toho hlavního, že je Windows operační systém, je systém Windows především knihovnou. Je to množina všech funkcí v rozhraní Windows API, dostupných při programování aplikací. Tyto zmiňované funkce nabízejí společné rysy - zobrazováním dialogů či rozhraním pro jeden či více dokumentů a přístupem k základním funkcím, jakou jsou zabezpečení a služby komponent. Zadruhé je systém Windows prostředím, v němž jsou aplikace spouštěny. Stejně tak by jsme mohli rozdělit i prostředí .NET. Zaprvé je to stejně rozsáhlá a kompletní knihovna jako Windows API, kterou můžeme používat k volání stejných funkcí, které dříve poskytoval samostatný operační systém Windows, ale také v oblastech nových jako například přístup k databázím, připojení k internetu nebo zprostředkování webových služeb. Zadruhé .NET nabízí rovněž běhové operační prostředí (.NET Runtime), v němž jsou programy spouštěny.[8]

.NET je tedy zastřešující název pro soubor technologií a model pro budování systémů na rodině operačních systémů Windows, ale také na četných operačních systémech nepocházejících od společnosti Microsoft, jako jsou Mac OS X a nejrůznější distribuce Unix/Linux.

3.3 Architektura a hlavní stavební bloky platformy .NET Framework

Z pohledu programátora se dá .NET chápat jako prostředí běhu programů tzv. runtime CLR (Common Language Runtime) a vyčerpávající knihovna základních tříd BCL(Basic Class Library). Tuto architekturu znázorňuje následující obrázek.



Obr. 4. Vztah CLR, CTS, CLS a knihovnou základních tříd BCL [11]

3.3.1 Běhové prostředí CLR

Na vrstvu runtime se má správně odkazovat jako na společný runtime jazyků označovaný jako CLR(Common Language Runtime). Modul CLR je základní jádro platformy .NET, Kód spuštěný pod kontrolou modulu CLR se často označuje jako řízený kód (managed code).

Před spuštěním v modulu CLR však libovolný zdrojový kód(v jazyce C# nebo v některém jiném podporovaném) vyžaduje překlad, který v prostředí .NET probíhá ve dvou krocích:

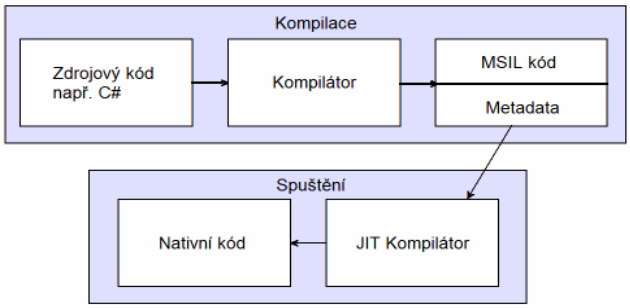
1. Překlad zdrojového kódu do jazyka IL.

2. Překlad jazyku IL do kódu specifického pro cílovou platformu pomocí modu CLR. [6]

3.3.2 Jazyk IL a překlad programu JIT

Jazyk IL(Intermediate Language) označován také jako CIL(Common Intermediate Language) nebo také MSIL (Microsoft Intermediate Language) hraje v platformě .NET Framework zásadní roli. MSIL je procesorově nezávislý kód podobný assembleru.

Dvoufázový proces překladu naznačen v předchozí kapitole je velmi důležitý, protože existence jazyka IL je základem mnoha výhod platformy .NET. To znamená, že stejný soubor s instrukcemi v bajtovém kódu lze přenést na libovolnou platformu. V době spuštění je možné snadno provést závěrečnou fázi překladu, která zajistí spuštění kódu na příslušné platformě. Jazyk IL se vždy překládá metodou Just-In-Time (JIT). Místo překladu celé aplikace v jednom průchodu (což by znamenalo dlouhý čas spouštění) překládá překladač JIT jednoduše každou z částí kódu v okamžiku jejího volání. Společnost Microsoft věří, že díky tomu dojde k navýšení výkonu aplikace, vzhledem k tomu, že závěrečná fáze překladu probíhá v době spuštění, kdy má překladač JIT přesné informace o typu procesoru, na kterém bude program spuštěn a výsledný spustitelný kód bude optimalizován tak, aby využil všech funkcí nebo instrukcí strojového kódu konkrétního procesoru. [11] [19]



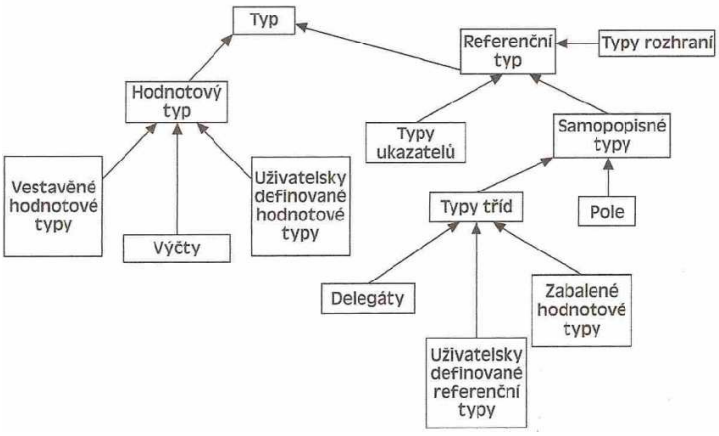
Obr. 5. Princip kompilace v prostředí .NET Framework [19]

Důležité vlastnosti jazyka IL:

- Objektová orientace a použití rozhraní
- Silné odlišení hodnotových a referenčních typů
- Silná typová kontrola dat
- Ošetření chyb pomocí vyjímek
- Uplatnění atributů [6]

3.3.3 Společný systém typů CTS

Dalším stavebním blokem platformy je společný systém typů CTS(Common Type System). Specifikace CTS plně popisuje všechny možné datové typy a programovací konstrukce, které runtime podporuje, specifikuje, jak mohou spolu tyto entity komunikovat, i podrobnosti o tom jak se mají reprezentovat ve formátu metadat .NET. Systém CTS neurčuje pouze základní datové typy, ale bohatou hierarchii typů. Součástí této hierarchie jsou kvalitně stanovené body, ve kterých smí kód definovat své vlastní typy. Hierarchická struktura systému CTS odráží objektově orientovanou metodiku jazyka IL s jednoduchou dědičností a odpovídá schématu na následujícím obrázku. [6]



Obr. 6. Hierarchie datových typů [6]

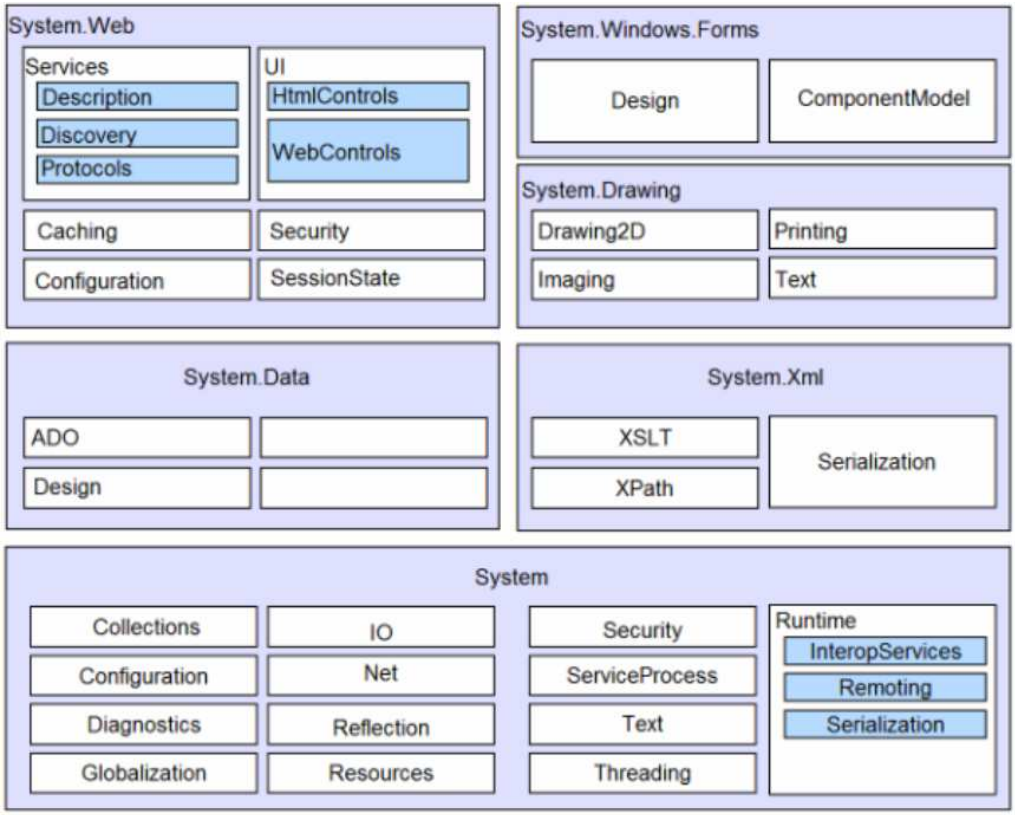
3.3.4 Spolupráce mezi jazyky CLS

Velkou výhodou platformy .NET je možnost vyvíjení aplikací v mnoha jazycích. Specifikace CLS (Common Language Specification) zajišťuje spolu se systémem CTS spolupráci jazyků. Specifikace CLS zahrnuje sadu minimálních standardů, které musí dodržovat všechny překladače určené pro platformu .NET. Díky CLS a jazyku IL .NET Framework podporuje celou řadu jazyků dodávaných s MS Visual Studiem NET jako jsou Visual Basic, C++, C#, J#. JScript ale také celou řadu programovacích jazyků třetích stran.[11]

3.3.5 Základní třídy

Knihovna základních tříd BCL (Basic Class Library) představuje rozsáhlou sbírku tříd řízeného kódu, které umožňují provádět téměř libovolné úkoly, k nimž se dříve využívalo

rozhraní Windows API. Uvedené třídy dodržují stejný objektový model jako jazyk IL, který je založen na jednoduché dědičnosti. To znamená, že můžeme buď vytvořit instance objektů libovolné základní třídy .NET, nebo od nich můžeme odvodit své vlastní třídy. Knihovny základních tříd přibližuje následující obrázek. [8]



Obr. 7. Knihovny systému NET Framework [19]

3.4 Verze platforem .NET Framework

Zde bych ještě chtěl zmínit, že verze .NET Frameworku se dají rozdělit na další podkategorie podle cílových platforem:

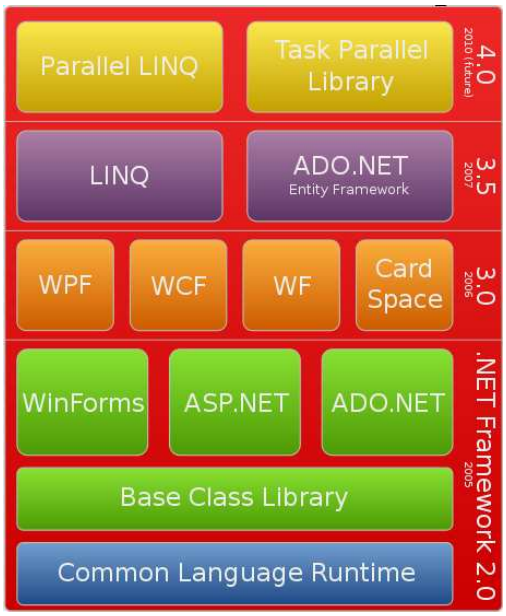
1) Microsoft .NET Framework je nejrozšířenejší platforma pro osobní počítače s operačním systémem Microsoft Windows od verze Windows 98.
2) Microsoft .NET Compact Framework je platforma určená pro kapesní počítače a mobilní telefony s operačním systémem Windows Mobile.
3) Microsoft .NET Micro Framework je platforma určená pro embedded zařízení, s ještě menší výpočetní kapacitou a většími omezeními, než kapesní počítače.
4) Verze .NET pro operační systémy nepocházejících od společnosti Microsoft:

- Mono - projekt uvedený společností Novell, jehož cílem bylo vytvořit sadu nástrojů kompatibilních prostředím .NET, které může běžet mimo operační systémy Microsoft také na operačních systémech UNIX, Linux, FreeBSD, MAC OS X, Solaris
- DotGNU - GNU obdoba .NET se nazývá DotGNU. Její část nazývaná DotGNU Portable.NET umožňuje spouštět všechny .NET aplikace na unixových platformách. [17]

Dále se budeme bavit jen o verzích týkající se této diplomové práce tedy .NET Framework pro systémy Windows.

3.4.1 Historie verzí .NET Framework

Historický vývoj .NET Framework je stručně popsán v této podkapitole. Hlavní přínosy jednotlivých verzí .NET Framework jsou přehledně znázorněny na následujícím obrázku.



Obr. 8. Verze .NET Framework [18]

Oficiální uvedení první verze platformy .NET Framework 1.0 v lednu roku 2002 se setkalo s velkým nadšením . Řadu dalších změn a vylepšení a také verzi .NET Compact Framework pro mobilní zařízení přinesla v roce 2003 verze .NET Framework 1.1 .

Verze .NET Framework 2.0, se kterou přišlo největší rozšíření platformy, je považována za klíčovou a byla vydána o dva roky později v roce 2005. Tato verze není zpětně kompatibilní s verzí 1.0 a 1.1. Za zmínku u této verze určitě stojí podpora 64bitových

procesorů a technologie ASP.NET(Aktive Server Pages), která umožňuje vytvářet webové stránky s dynamickým obsahem a technologie ADO.NET (ActiveX Data Objects), která představuje množinu tříd nabízejících služby pro přístup k datům a tvorbu databázových aplikací.

.NET Framework verze 3.0 byla uvedena s příchodem nového operačního systému Windows Vista v roce 2006 a přinesla nový pohled na vývoj aplikací v podobě čtyř nových knihoven:

- Windows Presentation Foundation (WPF) je knihovna pro práci s uživatelským rozhraním, která umožňuje oddělení návrhu vzhledu aplikace od samotného programového kódu s využitím jazyka XAML(Extensible Application Markup Language), kde jsou uživatelské prvky rozhraní definovány vektorově.
- Windows Communication Foundation (WCF) umožňuje komunikaci mezi jednotlivými aplikacemi uplatňující se hlavně v oblasti webových služeb.
- Windows WorkFlow Foundation obsahuje nástroje pro definování průběhu práce. Umožňuje jednoduše vytvářet schémata procesů jak v tradičních programovacích jazycích, tak i v jazyku XAML.
- Windows CardSpace přinesl jednotný přístup pro ověřování identity uživatele.

.NET Framework 3.5 vychází o rok později a pokračuje v rozvíjení nových funkcí doplněných ve verzi 3.0. Přináší doplněné třídy v základní knihovně tříd (BCL). Největším přínosem této verze je integrace technologie LINQ (Language Integrated Query) a informovanosti o datech. Tato nová funkce umožňuje vytvářet kód napsaný v jazycích s podporou technologie LINQ, který bude zajišťovat filtrování, práci s výčty a vytváření projekcí různých typů dat SQL, kolekcí, dat XML a sad DataSet s použitím jednotné syntaxe.

Dne 12.4.2010 spolu s vývojovým prostředím MS Visual Studio 2010 oficiálně vyšla i verze .Net Framework 4.0. Přínosem této verze .NET je podpora paralelismu a paralerního programování pro multiprocesorové a distribuované systémy v knihovně Task Parallel Library a využití těchto možností i v technologii PLINQ (Paraller LINQ). NET 4.O dále přináší plnou podporu pro jazyky IronPython, IronRuby, and F#.

Vývoj .NET Framework a nových verzí vývojových prostředí společnosti Microsoft Visual Studio. NET je přehledně znázorněna v následující tabulce.

[17] [18]

Tab. 2. Přehled vývoje verzí MS .NET Framework a vývojových nástrojů Visual Studio.NET

Verze	Datum vydání	Podpora Visual Studia	Základní součástí verze Windows
1.0	13.2.2002	Visual Studio .NET	
1.1	24.2.2003	Visual Studio .NET 2003	Windows Server 2003
2.0	7.11.2005	Visual Studio 2005	
3.0	3.11.2006		Windows Vista, Windows Server 2008
3.5	19.11.2007	Visual Studio 2008	Windows 7, Windows Server 2008 R2
4.0	12.4.2010	Visual Studio 2010	

3.5 Programovací jazyk Microsoft Visual C#

C# (anglická výslovnost "see sharp") je vysokoúrovňový objektově orientovaný a typově bezpečný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework, později schválený standardizačními komisemi ECMA (ECMA-334) a ISO (ISO/IEC 23270).

3.5.1 Porovnání jazyka C# s jinými jazyky

Z určitého hlediska lze jazyk C# považovat za jakousi revoluci mezi programovacími jazyky, jakou v prostředí Windows představuje technologie .NET. Současně s tím, jak společnost Microsoft přidávala v posledních desetiletí další funkce do systému Windows a do rozhraní Windows API, rozrostly se i jazyky Visual Basic 2008 a C++. Jazyky Visual Basic a C++ sice díky tomu získaly mimořádné možnosti, ale oba jazyky také kvůli způsobu svého vývoje trpěly různými omezení a problémy. Společnost Microsoft se rozhodla poskytnout vývojářům jazyk C# navržený speciálně pro technologie .NET a vytvořený zcela nově. Společnost Microsoft oficiálně popisuje jazyk C# jako „jednoduchý moderní, objektově orientovaný a typově bezpečný programovací jazyk, který je odvozen od jazyků C a C++“. Většina nezávislých pozorovatelů k těmto dvěma jazykům ještě dodává jazyk Java, ke kterému se přirovnává obtížnost naučit se jazyk C# . Struktura jazyku C# lépe vyhovuje moderním vývojovým prostředí než oba jazyky C a C++ a byl navržen tak, aby programátorům současně poskytl jednoduchost používání jazyka Visual Basic a v případě potřeby vysoký výkon a nízkoúrovňový přístup k paměti jako jazyk C++.

Díky tomu, že jazyk C# byl od začátku navržen pro spolupráci s technologií .NET, je jeho podpora funkcí této platformy kompletnější a dostupná na základě vhodnější syntaxe, než je tomu u výše uvedených jazyků. Jednou z oblastí, pro které tento jazyk není určen, je vývoj časově kritického nebo extrémně zatěžovaného kódu, tj. kódu, kde skutečně záleží na tom zda, průběh smyčky vyžaduje 1000 nebo 1050 cyklů procesoru a je potřeba uvolnit prostředky v milisekundě. [6] [9] [11]

3.5.2 Vztah jazyka C# k technologii .NET

C# je relativně nový programovací jazyk, který je významný ze dvou hledisek:

- Je specificky navržen a určen k použití s platformou .NET Framework společnosti Microsoft.
- Jedná se o jazyk založený na moderní metodice objektově orientovaného návrhu. Při jeho vývoji se společnost Microsoft využila zkušeností se všemi podobnými jazyky, které se objevily v průběhu přibližně dvaceti let od prosazení objektově orientovaných principů.

Je ale nutné zdůraznit důležitý fakt, že jazyk C# je samostatným jazykem. Je sice navržen tak, aby generoval kód určený pro prostředí .NET, ale není sám o sobě součástí platformy .NET. Tato platforma poskytuje některé funkce, které jazyk C# nepodporuje a jisté funkce jazyka C# nejsou naopak podporovány platformou .NET. C# je však zaměřen na práci s prostředím .NET. [8]

3.5.3 Základní vlastnosti a výhody jazyka C#

- Patří mezi Case-sensitivy jazyky (rozlišování malých a velkých písmen).
- Využívá automatickou správu paměti tzv. Garbage collection - funkce platformy .NET zajišťující automatické uvolňování dynamicky přidělované paměti.
- Plná podpora tříd a objektově orientovaného programování, včetně dědičnosti rozhraní i implementace, virtuálních funkcí a přetížení operátorů.
- V C# neexistuje vícenásobná dědičnost - to znamená, že každá třída může být potomkem pouze jedné třídy, aby se předešlo komplikacím a přílišné složitosti,

která je spojena s vícenásobnou dědičností. Třída ale může implementovat libovolný počet rozhraní.

- Možnost označit třídy nebo metody uživatelsky definovanými atributy.
- Jazyk je vhodný pro vývoj softwarových komponent distribuovaných v různých prostředích.
- Konzistentní a vhodně definovaná sada základních typu.
- Integrovaná podpora automatického generování dokumentace ve formátu XML.
- Plný přistup ke knihovně základních tříd .NET a také snadná dostupnost rozhraní Windows API.
- V případě potřeby jsou dostupné ukazatele a přímý přístup do paměti, ale jazyk je navržen takovým způsobem, že lze bez nich pracovat téměř ve všech situacích
- Vedle členských dat a metod využívá vlastnosti a události.
- C# neobsahuje a ani nepotřebuje dopřednou deklaraci - není důležité pořadí deklarace metod. [6]

3.5.4 Stručná historie a verze jazyka C#

C# 1.0

Zprávy o novém jazyku od společnosti Microsoft začaly vyplývat na povrch v roce 1998, kdy se tento vznikající jazyk označoval za COOL a říkalo se, že bude velmi podobný Javě. Tyto zprávy ale tehdy Microsoft ještě dementoval. V červnu roku 2000 ukončila společnost Microsoft všechny spekulace uvedením specifikací nového jazyka označovaného za C# a následovala předběžná verze sady SDK rámce .NET s kompilátorem C#. Nový jazyk vytvořil Anders Hejlsberg, Scoty Wlitamuth a Peter Golde. První oficiální verze byla vydaná v roce 2002 společně s .NET Framework 1.0. První verze obsahovala základní podporu objektového programování, ve které se vycházelo z jazyka C++ a zkušeností s jejich aktualizací v jazyce Java.

C# 2.0

Na další verzi se čekalo až do konce roku 2005. Mezi její nové vlastnosti patřila především nativní podpora generik vycházející z podpory na úrovni CLI, nová forma iterátorů

poskytující funkčnost generátoru, částečné a statické třídy, anonymní metody pro pohodlnější užívání delegátů(odkazů na metody), a Nullovatelné hodnotové typy.

C# 3.0

Tato verze byla vydána na konci roku 2007 společně s .NET Frameworkem 3.5 a Visual Studiem 2008. Obsahuje poměrně revoluční změny, které však nevyžadují změnu podkladového IL, takže aplikace v něm psané půjdou spouštět i na počítačích vybavených .NET Framework 2, ponesou-li si s sebou patřičné knihovny. Hlavní přínosy jazyka C# 3.0 jsou:

- Integrovaný dotazovací jazyk LINQ (Language Integrated Query), který přináší nový způsob pro dotazování nad jakýmikoliv daty, usnadňuje jejich tvorbu, třídění a vyhledávání v nich.
- Anonymní třídy umožňující např. rychlé vytvoření objektů přenášejících informace vyžádané z databáze přes LINQ.
- Lambda výrazy, jež si berou inspiraci z funkcionálního programování, umožňují tvořit anonymní metody, které obsahují jeden výraz nebo několik příkazů. Jednoduše řečeno lambda výraz je výraz, který vrací metodu. Pro potřebu lambda výrazů byl uveden nový operátor =>. Ten se nazývá „přechází v“.

C# 4.0

Konečná oficiální verze C# 4.0 byla vydána s .NET Frameworkem 4.0 a Visual Studiem 2010 v první polovině dubna roku 2010 . Nová verze se zaměřuje hlavně na spolupráci s dynamickými aspekty programování a to jsou:

- Dynamicky typované objekty
- Volitelné parametry a parametry určené jménem
- Vylepšená interoperabilita COM
- Typové parametry generik
- Kovariance a kontravariance

[16]

3.5.5 Ukázka základní struktury programu v C#

V následující ukázce zdrojového kódu je uveden jednoduchý program tzv. „Hello World“ v programovacím jazyku C#. Jedná se o konzolovou aplikaci, která pouze vypíše do okna konzole pozdrav „Hello World!“.

```
1     // Podle dohodnuté konvence mají soubory C# příponu *.cs
2     using System;
3
4     namespace MůjPrvníProgram
5     {
6           class Program
7           {
8                 static void Main(string[] args)
9                 {
10                      Console.WriteLine("Hello world!");
15                }
16          }
17    }
```

Na řádku 1 vidíme ukázku jednořádkového komentáře uvozenou znaky //(víceřádkové komentáře se používají tímto způsobem: /* zde může být zapsán komentář na více

řádcích */).

Třídy, základní jednotky objektového programování, jsou v C# rozděleny pro lepší orientaci a jednoznačnost názvů do jmenných prostorů. Na počátku zdrojového kódu řádku 2 jmenujeme příkazem using jmenné prostory, jež budeme používat. Zde je to using *System*, který obsahuje objekt *Console* použitý na řádku 10.

Příkazem namespace na řádku 4 říkáme, že chceme zařadit kód vymezený následujícími složenými závorkami (řádek 5 a 17) do jmenného prostoru MůjPrvníProgram.

Hned poté na řádku 6 definujeme klíčovým slovem class třídu Program, jejíž obsah bude opět vymezen dalšími složenými závorkami(řádek 7 a 16).

Na řádku 8 daklarujeme statickou metodu Main bez návratové hodnoty přijímající argument pole řetězců (string[] args). Každá aplikace musí obsahovat metodu Main (řádek 8-15), která v tomto případě obsahuje jen jeden příkaz pro výpis textu do okna konzole (Console.WriteLine("Hello world!");) na řádku 10.

4 POUŽITÉ VÝVOJOVÉ NÁSTROJE

Vývojové prostředí (IDE, anglicky Integrated Development Environment) je obecně software usnadňující práci programátorů, v němž probíhá celý komplexní vývoj aplikací. Většinou bývá zaměřené na jeden konkrétní programovací jazyk. Tento software poskytuje mnoho nástrojů pro vývoj, úpravu, překlad a ladění softwaru a obsahuje tedy editor zdrojového kódu, kompilátor, případně interpret, debugger. Cílem IDE je shrnout schopnosti nástrojů programovacího jazyka do takové ucelené podoby, která teoreticky snižuje čas potřebný k porozumění jazyku a zvyšuje produktivitu vývojáře. Těsná integrace činností může také ještě více přispět ke zvýšení produktivity. [22]

4.1 MS Visual Studio 2010

MS Visual Studio 2010 je nejnovější plně integrované bohatě vybavené vývojové prostředí od společnosti Microsoft. Může být použito pro vývoj konzolových aplikací a aplikací s grafickým rozhraním spolu s Windows Forms aplikacemi, databázovými aplikacemi, webovými stránkami, webovými aplikacemi a webovými službami ale také pro vývoj mnoha dalších typů aplikací. Dovoluje vytvářet také projekty, které kombinují moduly zkompilované pomocí různých programovacích jazyků. Visual Studio podporuje jazyky prostřednictvím jazykových služeb, což umožňuje, aby editor kódu a debugger podporoval jakýkoliv programovací jazyk. Mezi vestavěné jazyky patří C#(zde Visual C#), C++(Visual C++), Basic (Visual Basic), F# (Visual F#). Podpora dalších jazyků lze doinstalovat zvlášť. MS Visual Studio není samozřejmě jediné vývojové prostředí pro aplikace v jazyku C# , ale je určitě nejrozšířenější. Z řad alternativních vývojových prostředí bych jmenoval například SharpDevelop, Turbo C# Explorer a Monodevelop.[6]

4.1.1 Základní poskytované funkce

Při prvním spuštění Visual Studia se vývojové prostředí ideálně přizpůsobí pro práci ve zvoleném programovacím jazyku. Microsoft Visual Studio nabízí mimo mnoha dalších tyto základní funkce:

- **Textový editor kódu** - velice důmyslný editor kódu automaticky uspořádává kód pomocí odsazovacích čar, zvýrazňuje odpovídající začáteční a koncové závorky bloků a barevně odlišuje klíčová slova. V průběhu psaní kódu provádí některé

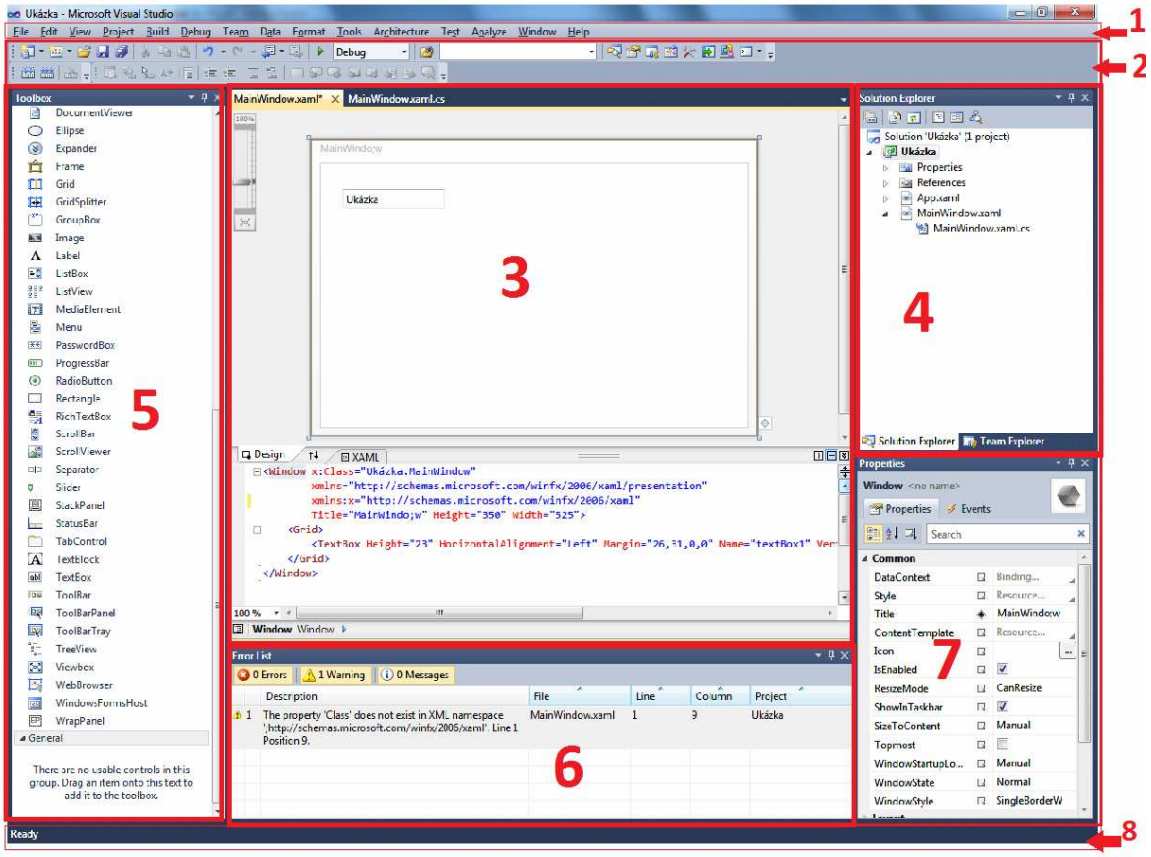
kontroly syntaxe a podtrhuje místa způsobující chyby v překladu takže umožňuje ladění v době návrhu. Praktická vlastnost IntelliSense automaticky nebo na zavolání zobrazuje nabídku jmen tříd, polí, metod jakmile začneme psát jejich počáteční písmena. Editor kódu Visual Studia také podporuje refaktorování včetně změny pořadí parametrů, přejmenování proměnných a metod, extrakci rozhraní a zabalení členů třídy uvnitř vlastností mezi ostatní a mnohé další funkce.

- **Editor návrhu vzhledu** - umožňuje rozvrhnout ovládací komponenty uživatelského rozhraní a přístupu k datům projektu a automaticky přidává do zdrojových souborů nezbytný kód pro vytvoření instancí těchto tříd.
- **Pomocná okna** - umožňují prohlížet a měnit všechny aspekty projektu jako třídy nebo vlastnosti komponent a jejich výchozí hodnoty jak webových formulářů tak formulářů Windows.
- **Možnost překladu v rámci prostředí -** integrovaný překladač, kterému jsou předány všechny potřebné parametry včetně detailního seznamu sestavení na které projekt odkazuje.
- **Integrovaný ladící program -** vhodně provázané prostředí s ladícím programem (debuggerem) umožňující různé stupně krokování a sledování hodnot proměnných.
- **Přístup k jiným programům -** v prostředí je možno také spouštět množství dalších užitečných programů umožňující například měnit nastavení počítače nebo sítě a kontrolovat běžící systémové služby například databázová připojení.
- **Integrovaná nápověda MSDN -** MSDN (Microsoft Developer Network) je program společnosti Microsoft určený především vývojářům. Součástí programu je Czech MSDN Blog, elektronický .NET zpravodaj, MSDN web, vývojářská fóra, MSDN Connection, MSDN RSS okénko a další. Většina služeb je poskytována zdarma. Spolu s vývojovým prostředí je možnost nainstalovat si velice rozsáhlou dokumentaci MSDN Library for Visual Studio a nebo jí využívat přes internet. Nápověda obsahuje kompletní nápovědu k Visual Studiu, platformě .NET, programovacím jazyků a spoustu praktických příkladů použití. Při označení klíčového slova či chybového hlášení a stisku klávesy F1 nám nápověda vyhledá související témata a řešení.

- **Práce s různými verzemi platformy .NET -** od verze Visual Studia 2008 je možnost zvolit si verzi platformy .NET, se kterou chceme pracovat a to .NET framework 2.0, 3.0, 3.5 či nejnovější verzí 4.0. Použitím nástroje pro převod projektu ze starší verze Visual Studia se samotný kód nezmění a je nám umožněno použít funkce Visual Studia novějšího.
- **Další funkce -** jen jmenovitě zde uvedu další užitečné funkce Visual Studia jako jsou průzkumník objektů, průzkumník otevřených panelů, průzkumník řešení, průzkumník týmů, průzkumník dat, průzkumník serveru(například MS SQL SERVERU). [6] [8] [21]

4.1.2 Okno aplikace

Pro popis základního okna aplikace jsem zvolil automaticky vytvořenou šablonu projektu z kategorie Visual C# typu WPF Aplication.(Windows Presentation Foundation). Na následujícím obrázku můžeme vidět červené očíslované a orámované základní bloky okna aplikace MS Visual studia 2010(zde verze Visual studia 2010 Ultimate RC) popsané níže.



Obr. 9. Okno aplikace MS Visual Studia 2010 RC

Blok 1 obsahuje hlavní panel nabídek (menu bar) Visual Studia, jehož struktura je stručně popsána takto:

- **File (Soubor) -** typická nabídka pro práci se soubory a projekty Visual Studia.
- **Edit (Upravit) -** zde najdeme možnosti práce se schránkou, funkce krok zpět a vpřed, vyhledávání a nahrazování textu, vkládání souboru, formátování dokumentu. Za zmínku zde stojí další možnosti funkce IntelliSense jako například nástroje pro generování nových metod, rozhraní, tříd.
- **View (Zobrazit) -** nabídka pro nastavení vzhledu Visual Studia, rozvržení jednotlivých bloků například editorů kódu a grafického navrháře, zobrazení bohaté sady panelů nástrojů a dalších pomocných oken jako jsou například průzkumník databázového serveru, průzkumník Solution Explorer (blok 4) a jiné.
- **Refactor (Refaktorizace)** - možnosti vyextrahování metod z kódu a vymazání čí přeuspořádání parametrů.
- **Project (Projekt) -** nastavení vlastností projektů a přidávání dalších komponent do projektu, jako jsou formuláře, třídy a jiné.
- **Build (Sestavení) -** konfigurace a vlastní funkce pro sestavení projektů.
- **Debug (Ladění) -** možnosti nastavení a vlastní ladění programu, krokování a správa vyjimek.
- **Data (Data) -** správa datových zdrojů, databází a SQL
- **Tools (Nástroje) -** nástroje pro nastavení a připojení k serverům, databází, vytváření GUID, manažer rozšíření funkcí Visual Studia, externí nástroje, importování a exportování nastavení.
- **Architecture (Architektura) -** nástroje pro vytváření grafů a UML diagramů.
- **Test (Test) -** nástroje pro testování běhu a ladění programu.
- **Analyze (Analýza) -** nástroje pro analýzování kódu.
- **Window (Okno) -** nastavení a rozložení jednotlivých oken projektu a souborů.
- **Help (Nápověda) -** zobrazení a nastavení nápovědy, ukázek programů, přístup k MSDN, technické podpoře, kontrola vylepšení, registrování produktu.

Blok 2 obsahuje nástrojovou lištu s širokou paletu zvolených nástrojů.

Blok 3 je největší a také hlavní pracovní plocha Visual Studia, kde je v tomto případě umístěn grafický návrhář aplikace, jinak zde bývá nejčastěji zobrazen textový editor zdrojového kódu konkrétního souboru. Jelikož v tomto případě se jedná o aplikaci WPF, tak se grafický návrhář skládá ze dvou částí, v horní části vidíme náhled grafického návrhu aplikace a pod ní vidíme její zdrojový kód v jazyce XAML. Mezi jednotlivými okny a zdrojovými soubory pracovní plochy se přepínáme pomocí záložek.

Blok 4 v pravé horní části obsahuje Průzkumník řešení (Solution Explorer) se strukturou souborů vytvořeného projektu.

Blok 5 v levé části obsahuje panel nástrojů Toolbox se základními ovládacími prvky formulářů.

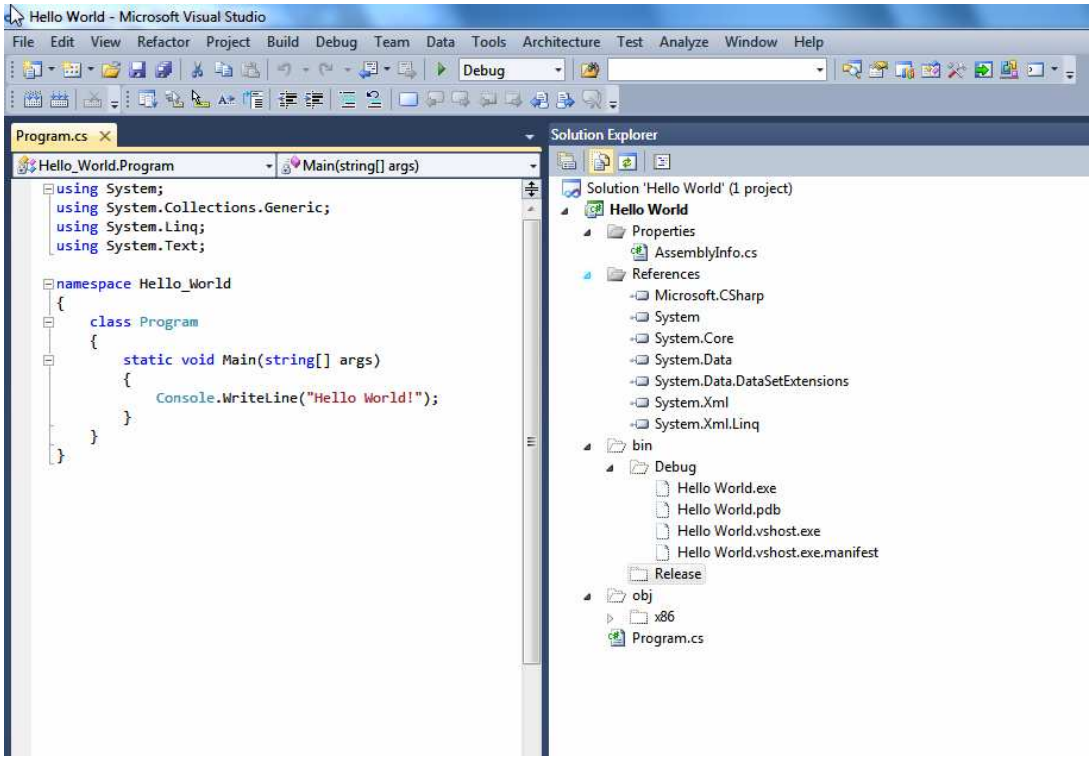
Blok 6 vpravo dole se nazývá Properties (Vlastnosti) a usnadňuje nám rychlé a přehledné nastavení vlastností jednotlivých objektu zde například vlastnosti okna vyvíjené aplikace.

Blok 7 pod hlavní pracovní plochou obsahuje seznam chyb, varování a informační zprávy (Error List).

Blok 8 dole je klasický stavový řádek prostředí.

4.1.3 Ukázka struktury aplikace ve Visual Studiu 2010

Zde uvádím praktickou ukázku programu z kapitoly 3.5.5., která poslouží k pochopení základní struktury aplikace v C# v prostření Visual Studia. Po vytvoření projektu s názvem "Hello World" pomocí šablony kategorie Visual C# s názvem Console aplication se může zobrazit projektu tak, jak můžeme vidět na následujícím obrázku. Je zde doplněn pouze jeden příkaz Console.WriteLine("Hello World!"); a program je následně přeložen a spuštěn.



Obr. 10. Struktura projektu aplikace Hello World

V levé části v textovém editoru kódu vidíme celý obsah souboru Program.cs. V pravé části vidíme kompletní rozbalený obsah okna průzkumníka řešení (Solution Explorer), který kromě dalších položek zobrazuje soubory projektu:

- **Solution 'Hello World' Project** – Jedná se o soubor řešení (solution file) na nejvyšší úrovni projektu. Předpokládejme, že celý projekt je uložen v adresáři \\Hello World . Pokud tento adresář otevřeme uvidíme zde, že tento soubor se ve skutečnosti jmenuje Hello World.sln. Každý soubor řešení obsahuje odkazy (reference) na jeden či více souborů projektu.
- **Hello World** – toto je soubor projektu v jazyku C#. Každý soubor projektu se odkazuje na jeden či více souborů se zdrojovým kódem a jinými položkami daného projektu. Celý nazev tohoto souboru je Hello World.csproj a je uložen na adrese \\Hello World\Hello World.
- **Properties** – jedná se o složku \\Hello World\Hello World , kde najdeme soubor s názvem AssemblyInfo.cs. Jde o speciální soubor, který lze použít k přidání atributů programu např. datum publikace programu nebo atributů, pomocí nichž můžeme upravit způsob spuštění programu.

- **References** – jedná se o složku, která obsahuje odkazy na zkompilovaný kód, která aplikace může použít.
- **bin a obj** – tyty položky odpovídají přímo složkám \\Hello World\Hello World\bin a \\Hello World\Hello World\obj a byly vytvořeny při sestavení aplikace a obsahují spustitelnou verzi programu společně s některými dalšími soubory, které se používají pro sestavení a ladění aplikace. Ve složce \\Hello World\Hello World\bin\debug se nacházejí čtyři soubory. Soubor Hello World.exe je přeložený program a spouštěcí soubor aplikace. Zbylé soubory Hello World.pdb, Hello World.vshost.exe , Hello World.vshost.exe.manifest obsahují informace, které Visual Studio využívá v případě spuštění programu v režimu ladění.
- **Program.cs** – zdrojový soubor jazyky C#. [9]

4.2 MS SQL Server 2008 Express

MS SQL Server 2008 je nejnovější verze relačního databázového systému (relational database management systém, RDBMS) od společnosti Microsoft. SQL Server je komplexní datová platforma, která dokáže obsluhovat požadavky na ukládání, manipulaci a prezentaci dat a je k dispozici v několika různých vydání přizpůsobeným funkčním požadavkům různých nasazení.

Pro rozsah této práce jsem si zvolil jednu ze základních verzi MS SQL Server 2008 Express with Advanced Services.

4.2.1 Přehled základních komponent MS SQL SERVER 2008

Obr. 11. Vize společnosti Microsoft
pro datovou platformu [2]

Základními komponentami MS SQL SERVER 2008 jsou:

- Databázový modul - klíčová služba systému SQL Server, která dovoluje ukládat, načítat, zpracovávat a zabezpečit data.

[2] Obrázek převzat z webu společnosti Microsoft: http://www.microsoft.com/cze/sqlserver2008/images/diag-sql2008-lg.gif

- Modul úložiště - řídí ukládání dat na disk a jejich zpřístupnění aplikacím.
- Podsystém zabezpečení - efektivní správa konfigurace funkcí zabezpečení, silné ověřování a řízení přístupu, výkonné možnosti šifrování a správy klíčů a zdokonalené auditování.
- Programovací rozhraní - základem je výkonný jazyk Transact-SQL(T-SQL, transakční SQL), díky modulu CLR je možnost použít i Visual C#.
- Service Broker - integruje do datové platformy SQL Serveru systém řazení zpráv do fronty.
- SQL Server Agent - modul plánování a výstrah.
- Replikace - modul replikace synchronizuje všechny změny kopií dat a jejich vzájemné synchronizace s hlavní sadou dat v rámci celé databázové struktury.
- Analysis Services - modul pro vytváření komplexních analytických řešení.
- Bussiness Inteligence - modul pro optimalizaci procesů, zkoumání a analyzování dat, filtrování a transformování dat, automatizované sestavy. [4] [10]

4.2.2 Přehled nástrojů pro správu MS SQL SERVER 2008

Systém SQL Server 2008 se dodává s osmi samostatnými nástroji, které umožňují konfigurovat, spravovat a sledovat databázové služby a také návrh objektů sytému SQL Server a spouštění kódu. Nejvšestrannějším nástrojem je SQL Server Management Studio ale pro úplnost zmíním i ostatní základní nástroje jako OSQL, SQLCMD, Tablediff, BCP, SQLDiag, Resource Governon, SQL Server Configuration Manager, Database mail. Základním dnes již zastaralým nástrojem SQL Serveru už od verze z roku 2000 je nástroj OSQL pro příkazový řádek umožňující připojení k instanci SQL Serveru a zadávání dotazů bez režie, která je spojena se zobrazením grafického rozhraní. Od verze SQL Serveru 2005 přišel nástroj SQLCMD nabízející rozhraní dotazů příkazového řádku a umožňuje nahrazení zastaralého OSQL. Nástroj Tablediff slouží k porovnávání struktury a dat dvou tabulek. Nástroj BCP se používá k exportu a importu dat tabulek. Ke shromažďování diagnostických informací SQL Serveru slouží nástroj SQLDiag. Nástroj Resource Governon dovoluje v databázi konfigurovat pravidla přidělování prostředků. Nástroj Database Mail umožňuje instancím systému SQL Serveru odesílat poštovní zprávy.

Důležitým nástrojem pro správu služeb a protokolů je nástroj SQL Server Configuration Manager, který je využíván zejména k těmto úkolům: [4]

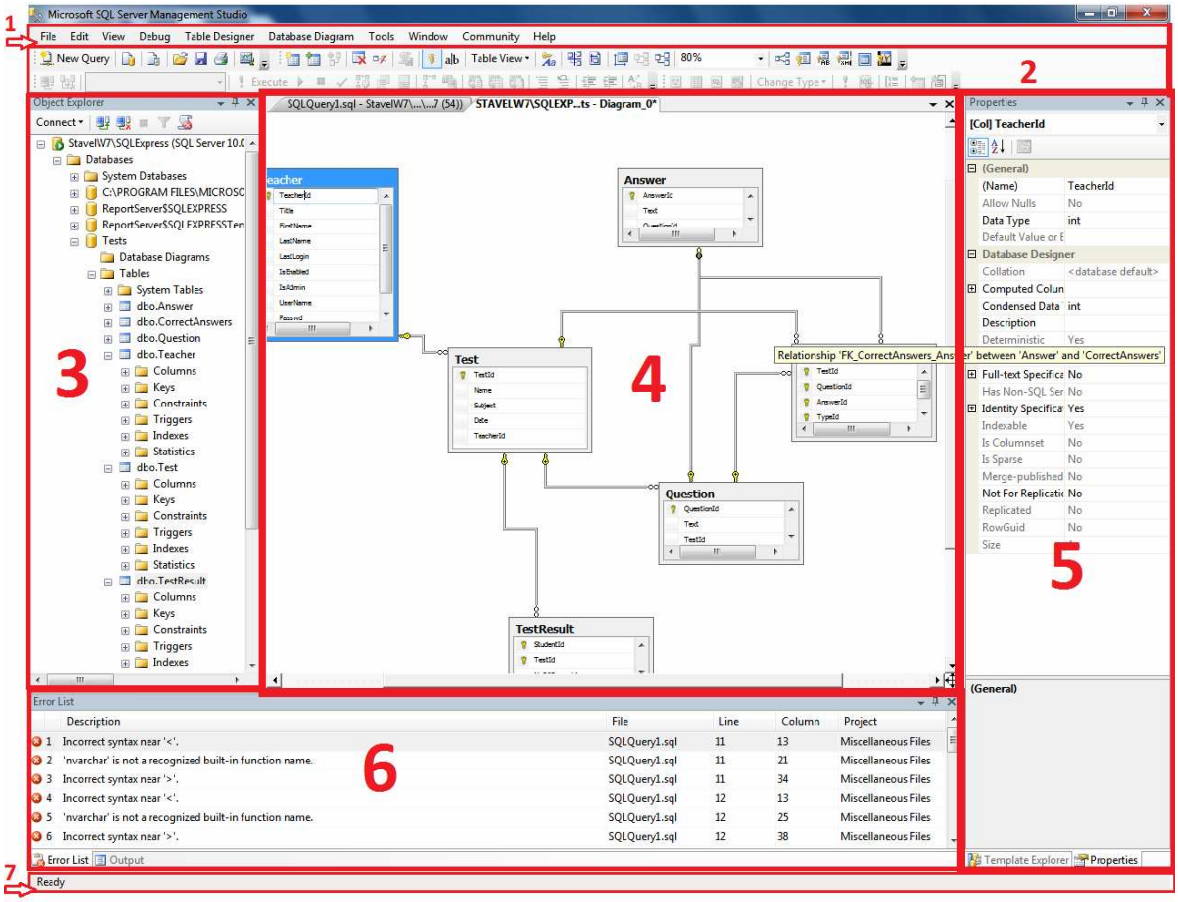
- Spuštění, zastavení, pozastavení a restartování služby
- Změna účtu služeb a hesel
- Správa režimu spuštění služby
- Konfigurace parametrů spuštění služby

4.2.3 SQL Server Managment Studio

SQL Server Managment Studio zkratkou SSMS je klíčový nástroj poskytující všechny možnosti správy systému SQL Server. Tento grafický nástroj je obsažen v SQL Serveru od verze 2005 a je vystaven na novém integrovaném vývojovém prostředí, které mimo jiných sdílí i MS Visual Studio. Mezi nejčastější základní úlohy, ke kterým využíváme SSMS patří:

- Vytváření databází.
- Správa databázových objektů – tabulky, schémata, procedury, funkce, triggery, indexy.
- Správa uživatelských účtů, přihlášení a rolí.
- Tvorba a správa automatizovaných úloh.
- Zálohování a obnova.
- Nastavení parametrů serveru – jazyk, využití zdrojů, výchozí umístění souborů.
- Psaní a ladění SQL skriptů, včetně jejich organizace do projektů. [10]

Pro lepší představu o práci s nástrojem SSMS je věnován následující obrázek Obr. 12. popisující strukturu a rozvržení bloků standardního okna SSMS. Po spuštění nástroje SSMS a následnému přihlášení k vybranému lokálnímu nebo vzdálenému SQL Serveru se zobrazí hlavní okno aplikace. Jednotlivé bloky popsané níže jsou na obrázku červeně orámovány a označeny příslušným číslem.



Obr. 12. Ukázka a rozvržení standardního okna nástroje SSMS

Blok 1 obsahuje hlavní panel nabídek SSMS, jehož struktura se mění v závislosti na tom, s čím aktuálně pracujeme. Stálá nabídka je tato:

- **File (Soubor)** - typická nabídka pro práci se soubory a projekty, nastavení a připojení k SQL Serverům.
- **Edit (Upravit)** - zde najdeme možnosti práce se schránkou, vyhledávání a nahrazování textu, vkládání souboru, formátování dokumentu.
- **View (Zobrazit)** - nabídka pro nastavení vzhledu, rozvržení jednotlivých bloků, zobrazení sad panelů nástrojů a dalších pomocných oken.
- **Debug (Ladění)** - možnosti nastavení a vlastní ladění, krokování a správa vyjímek dotazů nad databázi.
- **Tools (Nástroje)** - možnost využití externích nástrojů.
- **Window (Okno)** - nastavení a rozložení jednotlivých oken projektu a souborů.

- **Community (Komunita) -** nabízí možnosti spolupráce, položení a odpovídání otázek skupině vývojářů online.
- **Help (Nápověda) -** zobrazení a nastavení nápovědy, přístup k MSDN, technické podpoře, a dokumentaci. Archiv Books Online podrobně vysvětluje každou funkci SQL serveru s ukázkami kódu.

Na obrázku vidíme další aktuální nabídky pro práci s vlastnostmi tabulek **(Table Designer)** a práci s vlastnostmi diagramu databáze **(Database Diagram)** podle kontextu aktuální hlavní pracovní plochy bloku 4.

Blok 2 obsahuje volitelnou nástrojovou lištu s bohatou paletou nástrojů.

Blok 3 obsahuje okno Object Explorer pro správu kteréhokoliv objektu SQL Serveru. Zde jsou ve stromové struktuře zobrazeny všechny databáze, u databáze Tests její jednotlivé tabulky a další atributy.

Blok 4 je největší a také hlavní pracovní plocha SSMS, kde je v tomto případě umístěn grafický databázový diagram databáze Tests týkající se této diplomové práce.

Blok 5 obsahuje podokno Properties (Vlastnosti) usnadňující nám rychle a přehledně měnit vlastností jednotlivých objektu.

Blok 6 obsahuje seznam chyb (Error List), varování a informační zprávy.

Blok 7 stavový řádek SSMS.

II. PRAKTICKÁ ČÁST

5 NÁVRH ŘEŠENÍ

Ze zásad zadání práce plyne, že hlavním cílem této diplomové práce je vytvoření komplexního softwarového produktu, který by měl pedagogům umožnit jak vytváření, tak zpětné automatické vyhodnocení písemných testů studentů z opticky zpracovaného papírového formuláře. Nejvíce podmínek je kladeno právě na samotný papírový formulář, do kterého studenti vyplní určité údaje o své osobě a požadovaným způsobem do formuláře zanesou správné odpovědi na testové otázky, aby vyhodnocení jejich odpovědí bylo možno realizovat automaticky softwarově.

Protože nejrozšířenějšími operačními systémy ve školství jsou různé verze operačních systémů od společnosti Microsoft Windows, rozhodl jsem se vytvořit software, který bude určen pro použití právě na těchto operačních systémech. Pro efektivní, bezpečnou a spolehlivou správu dat uživatelů systému, informací o výsledcích testů studentů, samotných zadání testových otázek a odpovědí, jsem se rozhodl navrhnout takovou aplikaci, která bude využívat databáze.

Pro vývoj aplikace jsem si zvolil platformu .NET a programovací jazyk C#. V souvislosti s touto volbou jsem si vybral nejnovější vývojové prostředí MS Visual Studio 2010 a databázový systém MS SQL Server 2008. V teoretické části práce jsem se také snažil popsat nejdůležitější vlastnosti jazyka C#, platformy .NET a zvolená vývojová prostředí a informace doplnil vhodnými obrázky s ukázkami pracovních prostředí, zdrojových kódu a strukturou souborů projektů. V druhé kapitole praktické části se podrobněji zabývám návrhem a vytvořením vhodné databáze, na které bude aplikace postavena. Ve třetí kapitole rozebírám požadavky pro návrh vhodného testového formuláře. Čtvrtá kapitola podrobněji pojednává o použitých technologiích .NET i jiných, struktuře souborů projektu, hlavnímu algoritmu zpracování obrazových informací. Poslední pátá kapitola praktické části se zabývá instalací, funkcemi vytvořené aplikace a poslouží také jako návod pro práci s aplikací.

Pro vytvořenou aplikaci jsem zvolil název TestStudio.

6 DATABÁZE APLIKACE TESTSTUDIO

Prakticky žádný větší program pracující s nějakými složitějšími daty se neobejde bez databáze. Správně navržená databáze poskytuje uživateli přístup k aktuálním a přesným informacím. Pro dosažení požadovaných výsledků při práci s databází je správný návrh nezbytný, v opačném případě by mohla být výsledkem nefungující nebo neefektivní nebo složitě spravovatelná databáze.

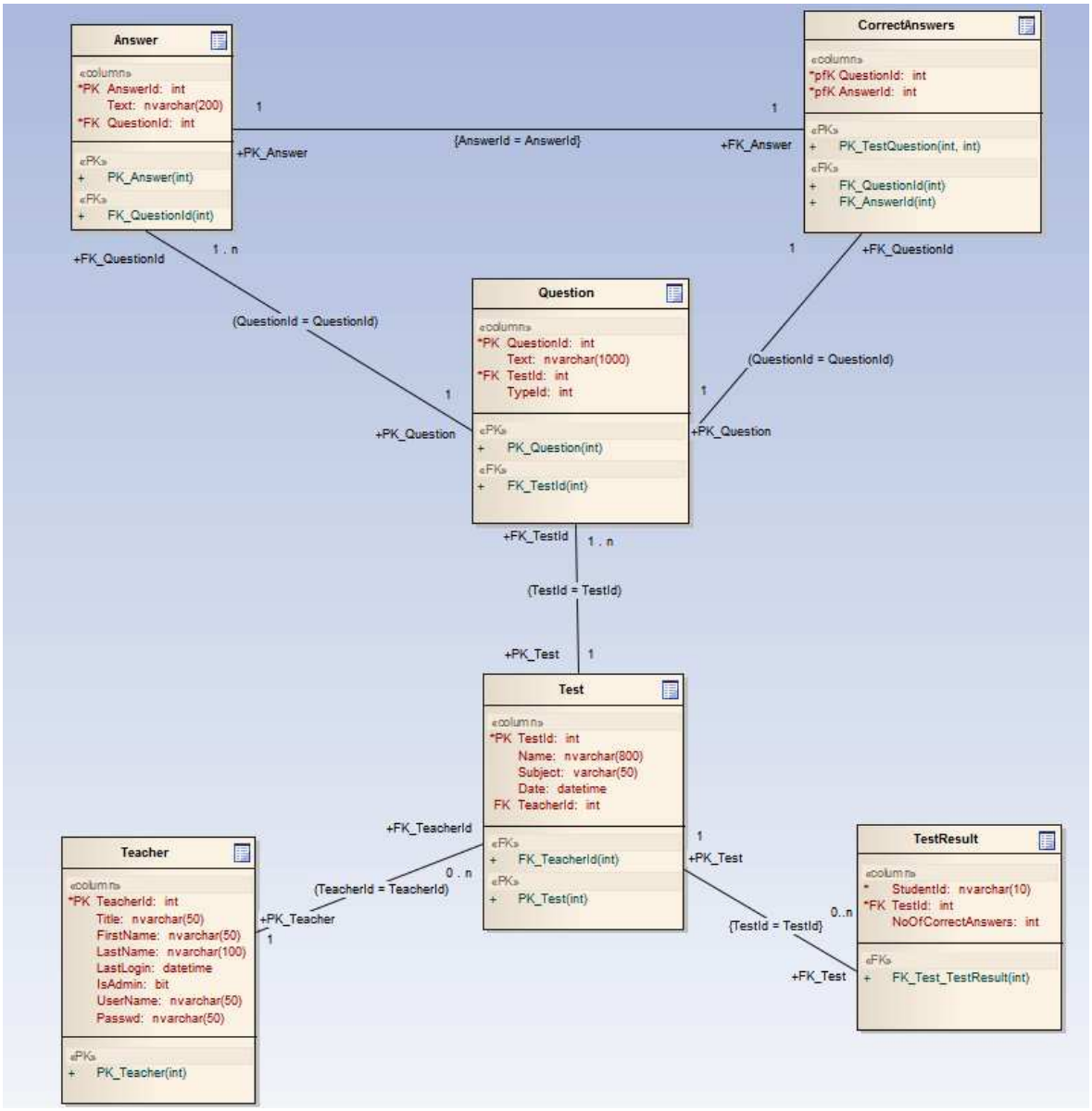
6.1 Analýza požadavků a konceptuální návrh databáze

Na začátku návrhu databáze je důležité si plně uvědomit samotný účel databáze, tedy kdo bude databázi využívat a k jakým účelům. Ze zásad zadání této diplomové práce je jasné, že uživateli aplikace a tedy i její databáze budou učitelé, kteří ji bude využívat k vytváření testů pro své studenty a v databázi se budou pak shromažďovat informace o výsledcích testů jednotlivých studentů. V tuhle chvíli máme tedy jasno o třech základních entitách a tedy i tabulkách databáze. První je entita učitel, další entitou je test, a třetí entitou je výsledek testu. Entita test je ale v databázi široký pojem, protože s každým testem je spojeno příliš mnoho dalších informací, abychom všechny informace mohli jednoduše ukládat do jedné tabulky. Testy podle zásad zadání budou mít určitý maximální počet testových otázek, kde každá otázka bude mít určitý počet odpovědí a zde je dále potřeba rozlišit, která odpověď bude správná. Jeden test bude dále v několika variantách, kde budou otázky a odpovědi v různém pořadí, aby se tak zamezilo studentům v možném podvádění opisováním odpovědí z vyplněného formuláře testu studentů sedících v těsné blízkosti. Z tohoto důvodu jsem do návrhu databáze zahrnul tři další tabulky jako je testová otázka, odpověď na testovou otázku a správná odpověď na testovou otázku.

Dalším krokem tedy bude shromáždění všech konkrétních typů údajů, které jednotlivé entity ponesou. Do každé tabulky proto přiřadíme jednotlivé sloupce - atributy. Každý atribut má určitý název a datový typ, který nám určuje, jaká data můžeme do sloupce ukládat, například zda to bude číslo, text, datum či jiné. Je nutné si dobře předem promyslet, jaké pro tabulky zvolit primární a cizí klíče pro vytvoření relací mezi daty v tabulkách. Návrh databáze je složitý proces. Konceptuální návrh databáze se sám o sobě může dělit na několik modelů vedoucí k vytvoření vlastní databáze (které jsem jmenoval v kapitole 2.2). Nejdůležitější je návrh fyzického modelu databáze. Já při navrhování

databáze využil software Enterprise Architect (EA) od společnosti Sparx Systems, který nabízí prostředí a výkonné flexibilní nástroje pro modelování, vývoj a testování software, založené na jazyku UML (Unified Modeling Language), což je grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů. Po založení nového projektu jsem si vytvořil tzv. data model, kde jsem si nastavil typ databáze SQL Server 2005 a postupně vytvořil databázový model, který můžete vidět na následujícím obrázku a který nám dává velice přesnou představu o navržené databázi, jejich entitách, atributech, klíčích, relacích. EA dokonce umožňuje vygenerovat příkazy jazyku SQL pro vytvoření takové databáze. Já ale použil po drobných úpravách vlastní viz. další kapitola.

Pro lepší orientaci v ukázce databázového diagramu zde popíši tabulku Teacher (entita učitele) propojenou s tabulkou Test (entita test). Tabulka Teacher je rozdělena na tři bloky, horní nese název tabulky Teacher, ve střední části tabulky vidíme všechny atributy tabulky(červené písmo) a jejich datové typy. Primární klíče jsou označeny PK, cizí klíče FK, poslední spodní blok nese informace o primárních a cizích klíčích (zelené písmo). Vlastní relace mezi tabulkami představuje čára propojující tabulky. Na straně tabulky Teacher vstupuje do relace primární klíč PK_Teacher (TeacherId) , na straně tabulky Test je to cizí klíč pojmenovaný FK_TeacherId (TeacherId). Uprostřed vidíme jejich rovnost. Jedná se o relaci typu 1 : n, jak jde vyčíst z popisu čáry na straně tabulky Teacher (1) a na straně tabulky Test (0 . n) .



Obr. 13. Entitně-relační modelování návrhu databáze s aplikací Enterprise Architect[3]

[3] Software dostupný z adresy: http://www.sparxsystems.com/products/ea/

6.2 Vytvoření databáze

V průběhu vývoje aplikace se databáze několikrát drobně upravovala jako například některé atributy tabulek. Konečná podoba kódu v jazyku SQL pro vytvoření tabulek databáze a vzájemných relací je tato:

```
1     CREATE TABLE Teacher (
2           TeacherId int IDENTITY(1,1),
3           Title nvarchar(20),
4           FirstName nvarchar(50),
5           LastName nvarchar(100),
6           LastLogin datetime DEFAULT GETDATE(),
7           IsAdmin bit DEFAULT(0),
8           UserName nvarchar(50) NOT NULL,
9           Passwd nvarchar(50) NOT NULL,
10          CONSTRAINT pk_teacher PRIMARY KEY CLUSTERED (TeacherId)
11    )
12    ;
13
14    CREATE TABLE Test (
15          TestId int IDENTITY(1,1),
16          Name nvarchar(200),
17          Subject nvarchar(50),
18          Date datetime DEFAULT GETDATE(),
19          TeacherId int NOT NULL,
20          CONSTRAINT pk_test PRIMARY KEY CLUSTERED (TestId
21          )
22    ;
23    CREATE TABLE TestResult (
24          StudentId nvarchar(10) NOT NULL,
25          TestId int NOT NULL,
26          NoOfCorrectAnswers int DEFAULT(0),
27          )
28    ;
29
30    CREATE TABLE Question (
31          QuestionId int IDENTITY(1,1),
32          Text nvarchar(1000),
33          TestId int NOT NULL,
34          TypeId int NOT NULL,
35          CONSTRAINT pk_question PRIMARY KEY CLUSTERED (QuestionId)
36    )
37    ;
38    CREATE TABLE Answer (
39          AnswerId int IDENTITY(1,1),
40          Text nvarchar(300),
41          QuestionId int NOT NULL,
42          CONSTRAINT pk_answer PRIMARY KEY CLUSTERED (AnswerId)
43    )
44    ;
45
46    CREATE TABLE CorrectAnswers (
47          QuestionId int NOT NULL,
48          AnswerId int NOT NULL,
49          CONSTRAINT pk_correctAnswer PRIMARY KEY CLUSTERED (
50          QuestionId, AnswerId)
51    )
52    ;
53
54    ALTER TABLE Test ADD CONSTRAINT FK_Test_Teacher
55          FOREIGN KEY (TeacherId) REFERENCES Teacher (TeacherId)
56    ;
```

```
57    ALTER TABLE TestResult ADD CONSTRAINT FK_TestResult_Test
58          FOREIGN KEY (TestId) REFERENCES Test (TestId)
59    ;
60    ALTER TABLE Question ADD CONSTRAINT FK_Test_Question
61          FOREIGN KEY (TestId) REFERENCES Test (TestId)
62    ;
63    ALTER TABLE Answer ADD CONSTRAINT FK_Answer_Question
64          FOREIGN KEY (QuestionId) REFERENCES Question (QuestionId)
65    ;
66    ALTER TABLE CorrectAnswers ADD CONSTRAINT
67          FK_CorrectAnswers_Question
68          FOREIGN KEY (QuestionId) REFERENCES Question (QuestionId)
69    ;
70    ALTER TABLE CorrectAnswers ADD CONSTRAINT FK_CorrectAnswers_Answer
71          FOREIGN KEY (AnswerId) REFERENCES Answer (AnswerId)
72    ;
```

Na očíslovaných řádcích 1 - 52 jsou příkazy SQL postupně vytvořeny všechny tabulky databáze: Teacher, Test, TestResult, Question, Answer, CorrectAnswers s jejich příslušnými atributy, datovými typy, primárními klíči a omezeními. Na řádcích 54 - 72 jsou následně vytvořeny cizí klíče, které jsou propojeny s primárními klíči jednotlivých tabulek a definují nám způsob propojení a svázání dat konkrétních tabulek, tedy vlastní relace mezi tabulkami. Tabulky dodržují všechny normální formy a relace mezi jednotlivými tabulkami jsou typu 1 : n a 1 : 1. Podrobnějším popisem vlastností takto vytvořených tabulek se zabývá následující podkapitola.

6.3 Přehled tabulek, atributů a jejich relací

Tab. 3. Tabulka Učitel - Teacher

Název tabuky: Teacher

Název sloupce	Význam	Datový typ	Klíč - relace
TeacherId	Jedinečné identifikační číslo učitele	int	primární klíč- tabulka Test: TeacherId
FirstName	Křestní jméno	nvarchar(50)	
LastName	Příjmení	nvarchar(100)	
LastLogin	Datum posledního přihlášení	datetime	
IsAdmin	Práva administrátora systému	bit	
UserName	Uživatelské jméno pro přihlášení	nvarchar(50)	
Passwd	Heslo pro přihlášení	nvarchar(50)	

Tab. 4. Tabulka Test - Test

Název tabuky: Test

Název sloupce	Význam	Datový typ	Klíč - relace
TestId	Jedinečné identifikační číslo testu	int	primární klíč- tabulka Test: TeacherId
Name	Název testu	nvarchar(200)	
Subject	Název předmětu testu	nvarchar(50)	
Date	Datum vytvoření testu	datetime	
TeacherId	Identifikační číslo tesu	int	cizí klíč - tabulka Teacher:TeacherId

Tab. 5. Tabulka Výsledek testu - TestResult

Název tabuky: TestResult

Název sloupce	Význam	Datový typ	Klíč - relace
StudentId	Identifikační číslo studenta	nvarchar(10)	
NoOfCorrectAnswers	Počet správných odpovědí	int	
TestId	Identifikační číslo testu	int	cizí klíč - tabulka Test:TestId

Tab. 6. Tabulka Otázka - Question

Název tabuky: Question

Název sloupce	Význam	Datový typ	Klíč - relace
QuestionId	Jedinečné identifikační číslo otázky	int	primární klíč - tabulka Test: TeacherId
Text	Text otázky	nvarchar(1000)	
Testid	Identifikační číslo tesu	int	cizí klíč - tabulka Test:TestId
TypeId	Verze varianty testu	int	

Tab. 7. Tabulka Odpověď - Answer

Název tabuky: Answer

Název sloupce	Význam	Datový typ	Klíč - relace
AnswerId	Jedinečné identifikační číslo odpovědi	int	primární klíč - tabulka Questiont:QuestionId
Text	Text odpovědi	nvarchar(300)	
QuestionId	Identifikační číslo otázky	int	cizí klíč - tabulka Question:QuestionId

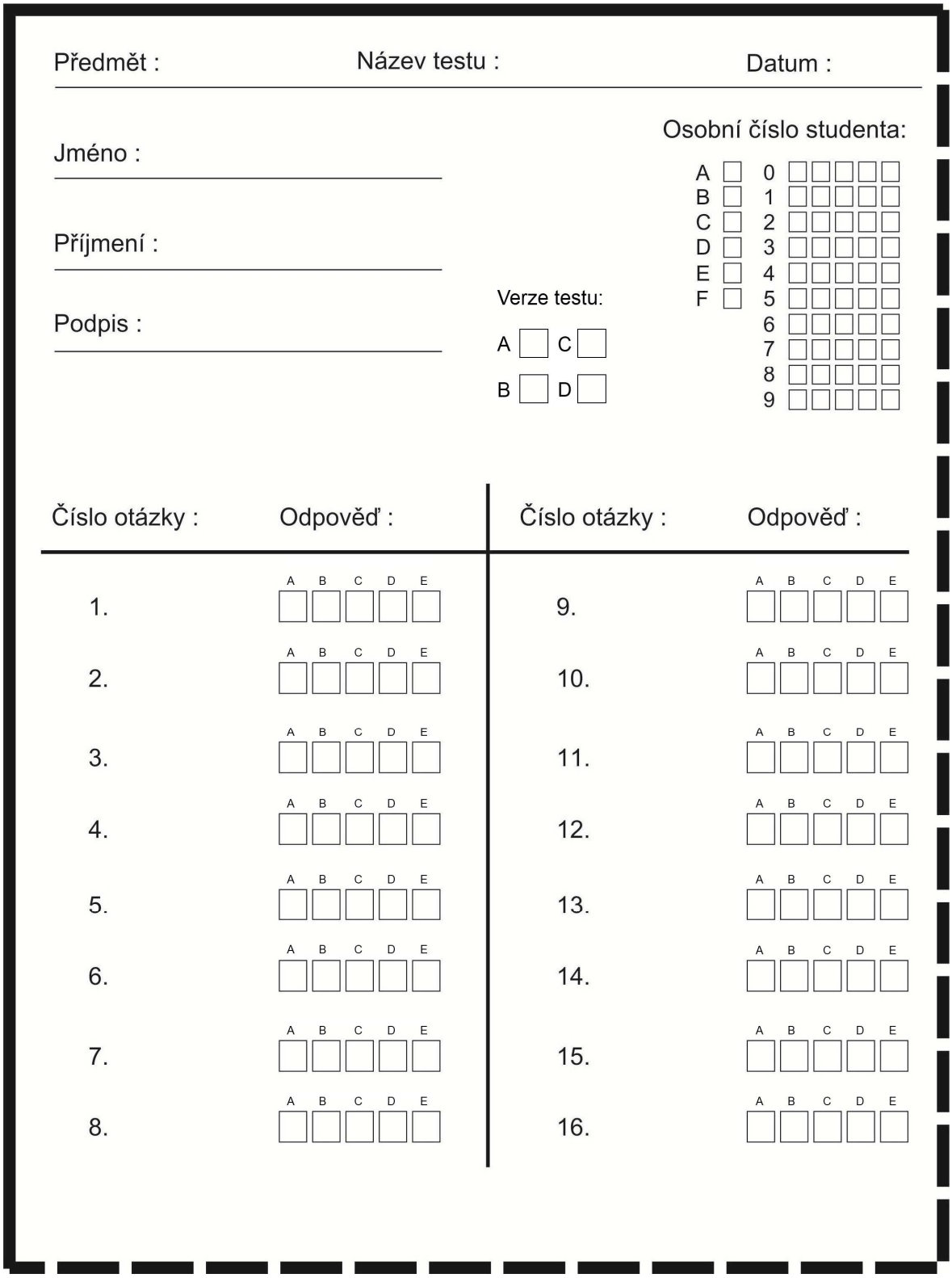
Tab. 8. Tabulka Správná odpověď - CorrectAnswer

Název tabuky: CorrectAnswers

Název sloupce	Význam	Datový typ	Klíč - relace
QuestionId	Identifikační číslo otázky	int	primární klíč(QuestionId, AnswerId) cizí klíč - tabulka Test:TestId cizí klíč - tabulka Question:QuestionId
AnswerId	Identifikační číslo odpovědi	int	

7 NÁVRH FORMULÁŘE TESTU A JEHO ZPRACOVÁNÍ

Na následujícím obrázku je umístěna šablona navrženého testového formuláře.



Obr. 14. Navržený formulář testu

7.1 Vlastnosti šablony

Šablona byla uložena jako soubor typu .jpeg s rozlišením 2255x3028 pixelů při barevné hloubce 24bit a celkové velikosti 1337kB. Pro vytvoření obrázku šablony byl použit freewarový nástroj PAINT.NET v 3.5.4[4].

Šablona obsahuje oblasti pro digitální zpracování a oblasti, které takto zpracovány nebudou. Jedná se o informace o testu, jako je název testu, předmět, datum konání testu, a informace o studentovi: jméno, příjmení a jeho podpis.

Pro snadnější zaměření oblastí pro digitální zpracování bylo navrženo orámování tvaru obdélníku. Levá a horní strana je souvislá černá čára. Pravá a dolní strana je přerušovaná čára. Toto orámování hraje důležitou roli při zpracování obrazových informací testu aplikací viz. kapitola 8.3, 8.4.

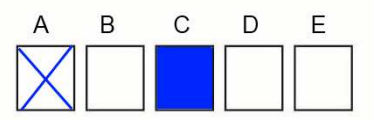
V šabloně máme tři oblasti, které se budou dále digitálně zpracovávat. Jedná se o tzv. osobní číslo studenta, skládající se z šesti znaků. Inspirací byla studentská čísla přidělována UTB ve Zlíně. První znak je písmeno určující fakultu studenta a za ním následuje pět číslic. Tato kombinace písmena a čísel jednoznačně identifikuje studenta. Další oblast je verze testu. Aplikace je navržena pro čtyři verze testu označené písmeny A, B, C, D, aby se tak snížila možnost podvádění mezi studenty opisováním. Největší část formuláře zabírají kolonky odpovědí na jednotlivé otázky. Nad každou kolonkou odpovědi je nadepsáno písmeno odpovědi A, B, C, D, E .

7.2 Způsoby vyplnění testového formuláře

Základním elementem testového formuláře je kolonka tvaru čtverce. Všechny kolonky v dané oblasti mají stejnou velikost. Podle zásad zadání diplomové práce bodu 3. se bude na testy odpovídat formou volby správného písmene tak, že pod písmenem správné odpovědi na papíře musí dotyčný zakřížkovat kolonku tvaru čtverce. V případě změny volby odpovědi se kolonka celá vytmaví a zatrhne se kolonka nová. V průběhu řešení této diplomové práce jsem dospěl k závěru, že mnohem vhodnější a flexibilnější by bylo

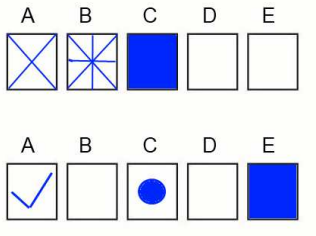
[4] Informace o softwaru Paint.NET v3.5.4 na adrese: http://www.getpaint.net/

vyhodnocování platné kolonky odpovědí nebo její korekce(ale nejen kolonky odpovědi ale i kolonek osobního čísla a verze testu) provádět lepším způsobem. Algoritmus vyhodnocování jsem proto navrhl tak, že student zakřížkuje kolonku správné odpovědi a v případě, že si svou odpověď rozmyslí, ponechá zakřížkovanou kolonku beze změny a novou kolonku správné odpovědi celou vytmaví. Zatím jednoduše řečeno, vybrání označené správné kolonky zde funguje tak, že jako správnou kolonku vybereme ze skupiny kolonek tu, která má nejmenší vypočítanou průměrnou hodnotu jasu (brightness), neboli čím tmavější tím správnější, jak vidíme na obrázku Obr.15. Nejprve zde byla studentem jako správná odpověď vybrána a označena zakřížkováním kolonka A, pak se student rozmyslel a jako správnou odpověď zvolil odpověď C tak, že označil vytmavením celou kolonku odpovědi C.



Obr. 15. Použitý způsob vyhodnocení

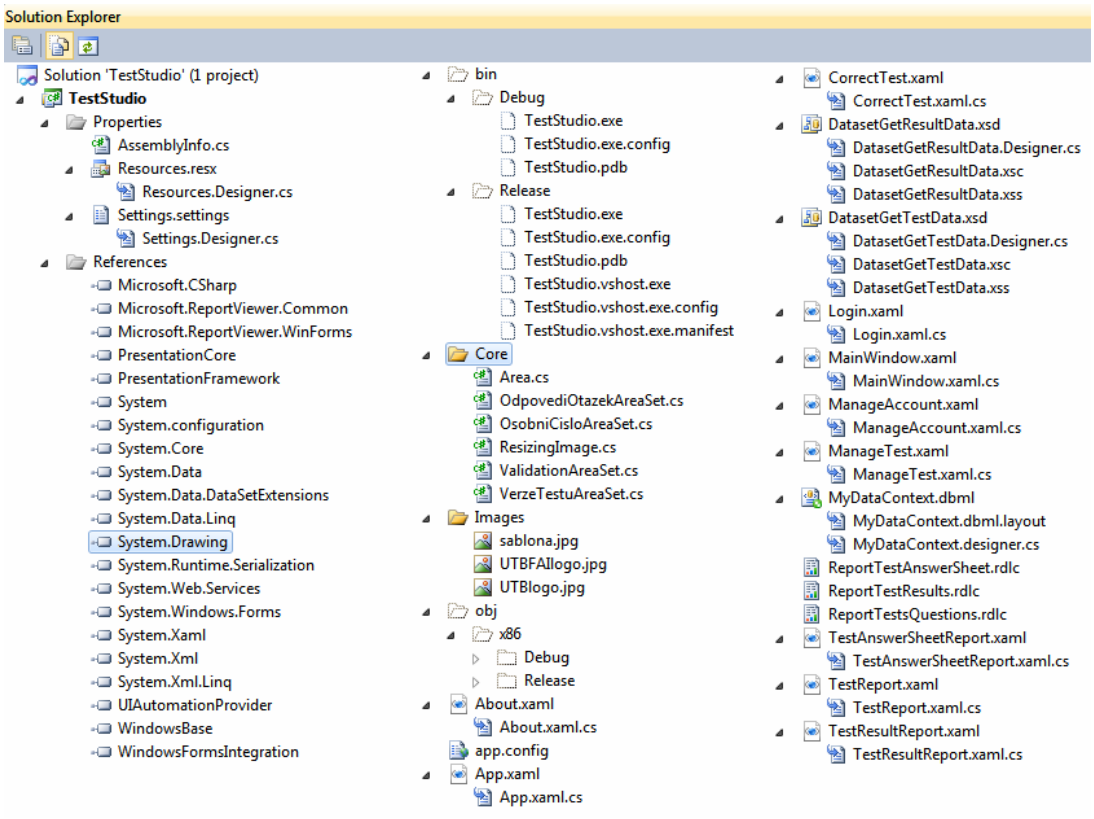
Takto navržený způsob vyhodnocování nám ale prakticky dává více možností zadat studentům podmínky, jakým způsobem se mají správné kolonky vyplňovat a opravovat. Pro označení kolonky by se nemusel používat vždy jen křížek, ale mohly by být použity například i puntíky a tzv. fajfky a korekce by se mohly provádět opět vytmavením, jak vidíme ve spodní řadě kolonek na obrázku Obr. 16. Nebo by se mohly provádět například i dvě korekce, tak jak vidíme v horní řadě kolonek stejného obrázku. Nejprve zde student označil jako správnou odpověď kolonku A křížkem, pak si to rozmyslel za správnou odpověď vybral kolonku B, kterou označil hvězdičkou, pak si to opět rozmyslel a jako správnou odpověď označil vytmavením kolonku C.



Obr. 16. Další možnosti vyhodnocení

8 STRUKTURA APLIKACE A POUŽITÉ TECHNOLOGIE .NET FRAMEWORK A JINÉ

Tato kapitola se věnuje nejpodstatnějšímu algoritmu vyhodnocení testu a technologiím .NET Framework použitým při vývoji aplikace TestStudio a pro lepší představu o nich rozebírám některé příklady použití. Tyto příklady byly vybrány proto, že nejsou příliš obsáhlé a zorientovat se v nich nebude složité. Všechny další soubory stejného typu jsou v projektu jsou vybudovány obdobně, jen jsou mnohem více obsáhlejší a jsou k nalezení na disku CD-ROM, který je součástí této diplomové práce. Aplikace TestStudio byla vytvořena ve Visual Studiu 2010 jako nový projekt typu WPF application z kategorie Visual C#, podkategorie Window Aplications. Propojení projektu s databází na SQL Serveru je realizováno přidáním knihoven LINQ to SQL classes do projektu. Hlavní jádro aplikace se nachází v složce pojmenované Core. Najdeme zde několik C# souborů, které realizují nejpodstatnější operace zpracování a vyhodnocení obrazových informací. Pro tvorbu předloh zadání testů a výsledkových listů byly využity reportovací nástroje MS Visual Studia. Na následujícím obrázku vidíme přehlednou strukturu souborů projektu. Podrobnějšímu popisu se věnují následující podkapitoly.



Obr. 17. Struktura souborů projektu aplikace TestStudio v MS Visual Studiu 2010

Na předchozím obrázku Obr. 17 v levém sloupci ve složce Properties nalezneme několik C# souborů obsahující základních informací o zdrojích, sestavení a nastavení projektu TestStudio, ve složce References jsou zobrazeny názvy použitých knihoven. Zorientovat se ve složce bin a obj by jsme měli díky informacím z kapitoly 4.1.3, všechny ostatní typy zdrojových souborů jsou popsány v následujících podkapitolách.

8.1 WPF Formuláře aplikace

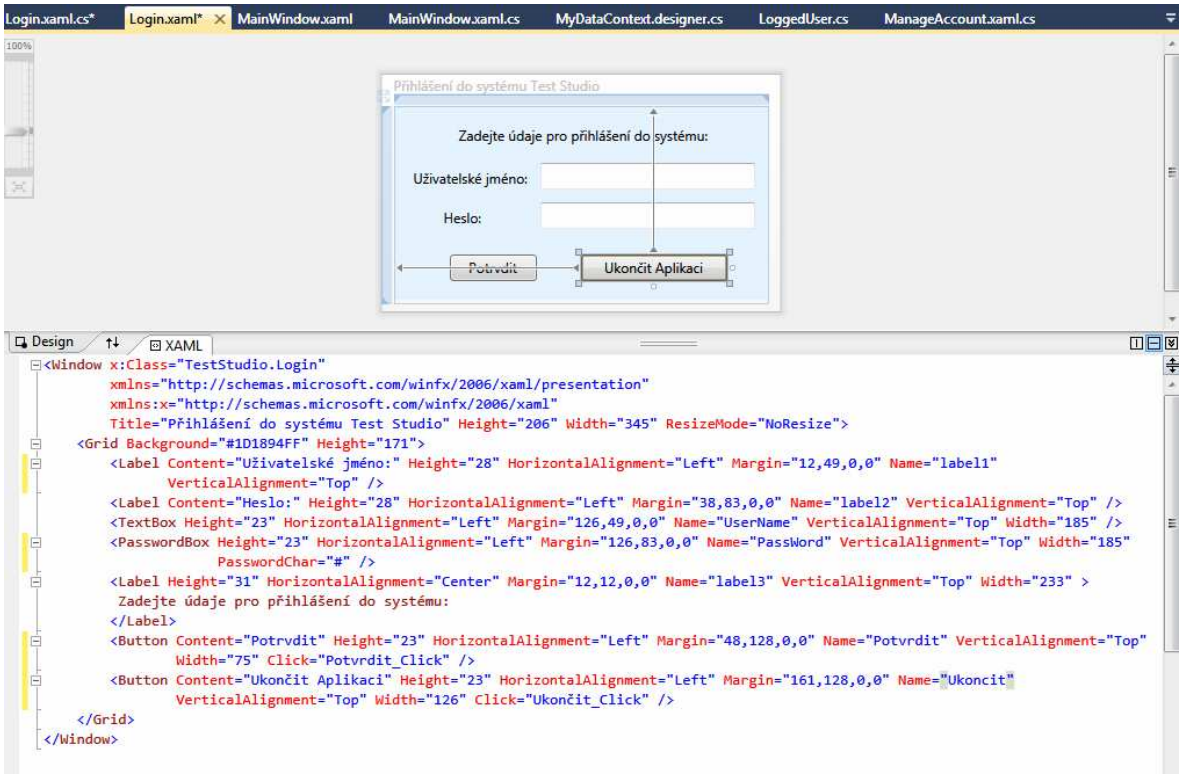
Všechny formuláře aplikace jsou založeny na technologii knihoven Windows Presentation Foundation (WPF), která přišla už s vydáním .NET Framework 3.0. Knihovna slouží k vytváření uživatelského rozhraní klientských aplikací. Zatímco ovládací prvky Window Forms jsou nativní ovládacími prvky systému Windows, které užívají popisovače okna založené na pixelech, WPF je založeno na rozhranní DirectX a pyšní se tím, že umožňuje oddělit návrh vzhledu aplikace od samotného programového kódu zde jazyka C# s využitím jazyka XAML. XAML je zkratka slov XML for Applications Markup Language (značkovací jazyk XML pro aplikace) a používá se při definování hierarchické struktury uživatelského rozhraní.

Jako ukázkový příklad toho jak je vlastně myšleno oddělení návrhu vzhledu od programového kódu, uvádím grafickou ukázku konkrétního formuláře a s ním spjatého zdojového kódu v jazyce XAML a v jazyce C#. Ve struktuře souborů projektu TestStudio na obrázku Obr.17 vidíme soubory s názvem Login.xaml a Login.xaml.cs. Obsah těchto souborů je znázorněn tak, jak je zobrazí Visual Studio na následucí straně. Na prvním obrázku je soubor Login.xaml, v horní části okna vidíme náhled v grafickém editoru, jak bude okno přihlášení do aplikace vypadat, pod ním vidíme samotný kód jazyka XAML. V následující ukázce kódu se zaměřím na deklaraci tlačítka pojmenovaného "Potvrdit" jehož reprezentace v XAML je následující:

```
<Button Content="Potrvdit" Height="23" HorizontalAlignment="Left" Margin="48,128,0,0"
Name="Potvrdit" VerticalAlignment="Top" Width="75" Click="Potvrdit_Click" />
```

Za každým elementem jazyka XAML vždy stojí třída .NET. Prostřednictvím atributů a podřízených elementů nastavíme hodnoty vlastností a definujeme metody pro zacházení s událostmi. Zde na příkladu tlačítka vidíme jeho definované vlastnosti jako jsou textový obsah, výška, horizontální zarovnání, poloha, jméno, vertikální zarovnání, délka a jako

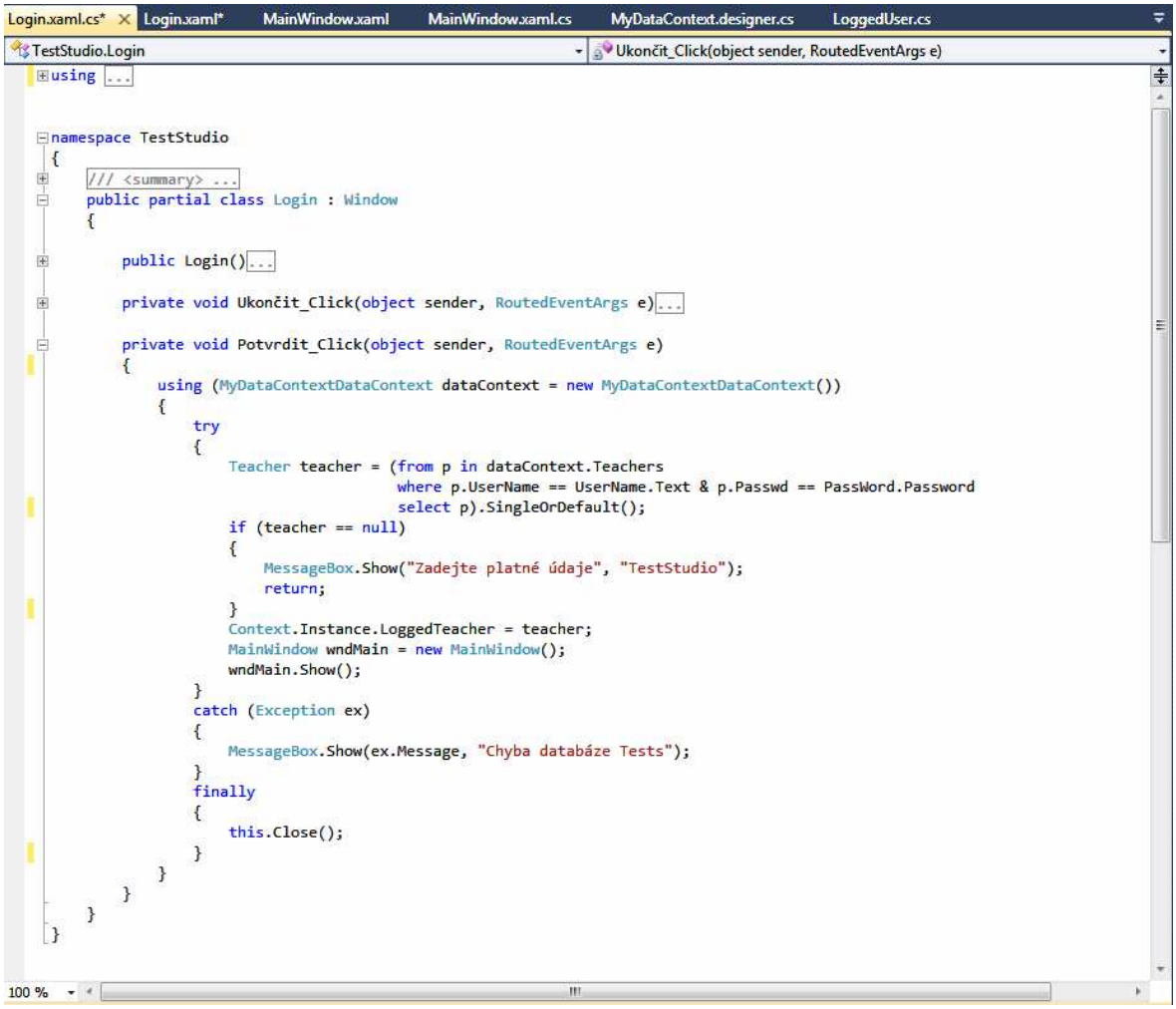
poslední je definována metoda `Click="Potvrdit_Click"` pro obsluhu události, která vznikne po stisknutí tlačítka.



Obr. 18. Ukázka souboru Login.xaml

Metoda Potvrdit_Click je ale umístěna již v dalším souboru Login.xaml.cs, jehož obsah je na následujícím obrázku. Po vyplnění správných údajů pro přihlášení uživatele aplikace a stisknutí tlačítka Potvrdit, se aplikace pokusí připojit k databázi pomocí datakontextu vytvořeného pomocí tříd LINQ to SQL (viz další podkapitola), dojde ke kontrole uživatelského jména a hesla s daty v databázi a v případě úspěchu se nastaví kontext pro aplikaci přihlášeného uživatele využitím programovacích technik Singletonu[5]. Nakonec se otevře hlavní okno aplikace TestStudio a okno pro přihlášení a připojení k databázi se uzavře.

[5] Více informací na adrese http://cs.wikipedia.org/wiki/Singleton



Obr. 19. Ukázka souboru Login.xaml.cs

8.2 Propojení aplikace s databází MS SQL Serveru

Vlastní propojení projektu z databází běžící na SQL Serveru je založeno na využití knihoven LINQ to SQL classes. Jak jsem již stručně zmínil v kapitole 3.4.1 s příchodem verze .NET Framework 3.5 přišlo také prostředí jazyka integrovaných dotazů v .NET (.NET Language Integrated Query Framework) tedy dotazovací jazyk LINQ. Existuje několik různých typů aplikace LINQ podle typu dat, s nimiž chceme pracovat jako LINQ pro objekty, pro datové množiny, pro entity, pro XML. Mě však v téhle kapitole zajímá využití jazyka LINQ pro SQL v prostředí Visual Studiu 2010(dostupné ale už od verze 2008). Přidáním knihoven LINQ to SQL do projektu získáme možností využívat silně typové rozhraní pro práci s databází na SQL Serveru, které nám umožňuje navrhovat třídy, se kterými můžeme dále pracovat. LINQ pro SQL se netýká pouze dotazování se na data, ale umožňuje také provádět všechny potřebné operace vkládání, aktualizace, mazaní a jiné.

Na obrázku Obr. 17 v průzkumníku řešení vidíme soubor s názvem MyDataContext.dbml. Jedná se o soubor typu LINQ to SQL classes, který obsahuje dvě komponenty MyDataContext.dbml.layout a MyDataContext.designer.cs. Užitečným nástrojem Visual Studia je nástroj objektově-relační mapovací návrhář pracující se souborem MyDataContext.dbml, s jehož pomocí můžeme graficky navrhovat objekty pro mapování do databáze. Kód C# vygenerovaný tímto návrhářem je uložen v souboru MyDataContext.designer.cs. Databáze jako kompletní systém tabulek, pohledů, procedur, všeho co jí tvoří, je tak převedena na prvek v jazyku LINQ typu DataContext. K vlastním atributům tabulek se pak přistupuje jako vlastnostem. Další databázové prvky jako jsou tabulky, pohledy, vztahy jsou převedeny do jazyka LINQ jako třídy a kolekce a vnořené kolekce.

Informace týkající se připojovacího řetězce k databázi se načítají z konfiguračního souboru aplikace. To nám dává možnost měnit konfiguraci připojení k databázi aplikace, aniž by jsme aplikaci museli znovu sestavovat.

8.3 Konfigurace aplikace

Pro lepší zorientování v konfiguraci aplikace zde uvádím stručný popis struktury a obsahu konfiguračního souboru aplikace app.config. Jedná se o soubor typu xml, který je rozdělen do několika hlavních částí a sekcí. Zde uvádím jako ukázku jen velmi zkrácenou verzi obsahu konfiguračního souboru. Prvky stejného typu jsou nahrazeny znaky ". . . ". V horní části <configSections> máme definované skupinu sekcí pojmenované "Areas". Jedná se o oblasti, které bude aplikace zpracovávat a vyhodnocovat. V kapitole 7 jsme si prohlédli šablonu formuláře testu. Šablona je orámovaná vlevo a nahoře černou čarou a vpravo a dole přerušovanou černou čarou určité šířky. Tento rám je v konfiguračním souboru reprezentován sekcemi pojmenovanými "ValidationLeftTop" a "ValidationRightBottom". Nastavení limitů hranic pro přijetí obrázku aplikací podle jeho rámu se nachází v sekci <appSettings> a jedná se o průměrnou hodnotu jasu v rozsahu <0 - hodnota jasu černé bárvy, 1 - hodnota jasu bíle barvy> pro plnou a pro přerušovanou část rámu testového formuláře. Dále zde najdeme sekce "StudentIdArea", "TestVersionArea" a "QuestionArea". Z názvů je už patrné, že se jedná o sekce, kde budou definované všechny oblasti nesoucí informace o studentském čísle, verzi testu a odpovědích na otázky. Jako příklad zde uvedu jednu kolonku odpovědi E u otázky číslo 16, která je reprezentována dvěma obrazovými

body se souřadnicemi levého horního rohu x_1 = 656, y_1=1400 a pravého dolního rohu x_2 = 724, y_2 = 1475.

```
<add key="QuestionArea16.5" value="656;1400;724;1475" />
```

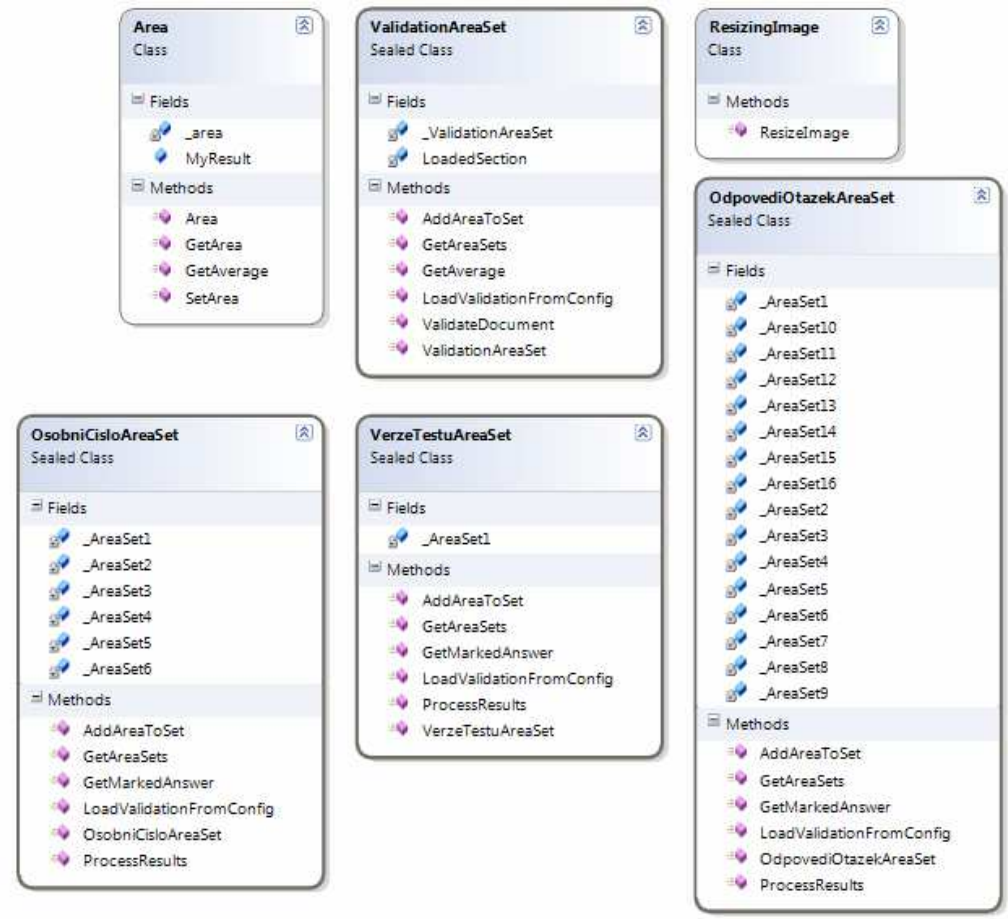
V poslední části konfigurace nalezneme sekci <connectionString>, která obsahuje všechny informace (adresa SQL Serveru, název databáze, uživatelské jméno a heslo) připojovacího řetězce k SQL Serveru.

```
<?xml version="1.0" encoding="utf-8" ?>

<configuration>
  <configSections>
    <sectionGroup name="Areas">
      <section name="ValidationLeftTop" type="System.Configuration.NameValueSectionHandler"/>
      <section name="ValidationRightBottom" type="System.Configuration.NameValueSectionHandler"/>
      <section name="StudentIdArea" type="System.Configuration.NameValueSectionHandler"/>
      <section name="TestVersionArea" type="System.Configuration.NameValueSectionHandler"/>
      <section name="QuestionArea" type="System.Configuration.NameValueSectionHandler"/>
    </sectionGroup>
  </configSections>
  <Areas>
    <ValidationLeftTop>
     ...
      <add key="ValidationArea2" value="0;0;2250;22" />
    </ValidationLeftTop>
    <ValidationRightBottom>
      ...
      <add key="ValidationArea4" value="0;3009;2248;3025" />
    </ValidationRightBottom>
   <QuestionArea>
                ...
              <add key="QuestionArea16.5" value="656;1400;724;1475" />
    </QuestionArea>
    ...
  </Areas>

<appSettings>
    <add key="TolerationAreas/ValidationLeftTop" value="0,28" />
    <add key="TolerationAreas/ValidationRightBottom" value="0,45" />
</appSettings>

<connectionStrings>
        ...
    <add name="TestStudio.Properties.Settings.db890ConnectionString"
      connectionString="Data Source=sql3.aspone.cz;Initial Catalog=db890;Persist Security
Info=True;User ID=db890;Password=dp1meluzin2x"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

8.4 Zpracování a vyhodnocení obrazových dat

Ve složce projektu Core (jádro) nalezneme tyto C# soubory:

Area.cs, ValidationAreaSet.cs, OsobniCisloAreaSet.cs, VerzeTestuAreaSet.cs, OdpovediOtazekAreaSet.cs, ResizingImage.cs. Soubory obsahují třídy a metody, jejichž přehled je na následujícím obrázku.



Obr. 20. Diagram vytvořených tříd realizujících zpracování a vyhodnocení obrazových informací testového formuláře

Po zadání vlastních obrazových dat (viz. kapitola 9.5) aplikace nejprve zjistí vlastní rozlišení obrázku, pokud je jiné než požadované (2255x3028 pixelů), což v drtivé většině případů samozřejmě bude, dojde ke změně rozlišení obrázku na tuto požadovanou hodnotu. Tuto změnu rozlišení realizuje třída ResizingImage.cs. Poté následuje vlastní validace obrázku, jejíž výsledek rozhodne zda bude obrázek vůbec aplikací přijat k dalšímu zpracování. Orámování formuláře testu bylo záměrně navrženo tak, aby souvislé černé rovné hrany a z jedné strany přerušované hrany byly skenerem jednoduše rozpoznatelné a software skeneru je dokázal automaticky oříznout až na vnější hranici orámování. Při tisku nebo následném skenování obrázku může dojít ke zkosení hran celého orámování testového formuláře. Nebo by mohl nastat případ, že by test byl naskenován celý převráceně. U procesu skenování vyplněného testového formuláře předpokládáme, že vlastní software dodávaný ke konkrétnímu skenovacímu zařízení standardně umožňuje

nastavit automatické ořezání skenovaného dokumentu. Nemusí to ale být vždy pravidlem nebo výsledek nemusí být ideální, proto v těchto případech se počítá s tím, že uživatel obrázek upraví manuálně s využitím nějakého vhodného grafického nástroje (postačí i MS Paint, který je součástí MS Windows). Validace rámu testu je realizována třídou ValidationAreaSet. Rám má celkem čtyři validační oblasti. Jedná se vlastně o geometrické objekty - obdélníky vyplněné černou barvou vlevo a nahoře a o dva obdélníky, které jsou vyplněny převážně černou ale přerušovaně také bílou barvou vpravo a dole. Na základě souřadnic bodů v obrázku načtených z konfiguračního souboru aplikace (levý horní a pravý dolní rohový bod) dojde k vypočtení průměrné hodnoty jasu barvy v dané ploše. Výsledkem budou dvě desetinné čísla. První číslo nese průměrnou hodnotu jasu celkové plochy levého a horního obdélníku a druhé číslo průměrnou hodnotu jasu pravého a dolního obdélníku. Tyto hodnoty se porovnají s nastavenou limitní hodnotou v konfiguračním souboru TolerationAreas/ValidationLeftTop a TolerationAreas/ValidationRightBottom (viz. kapitola 8.3). Pokud budou vypočtené hodnoty větší než tyto limity, obrázek nebude aplikací vůbec přijat a to bude nejčastěji z důvodů přílišného zkosení celého obrázku, ke kterém by mohlo dojít při procesu tisku a skenování, nebo z důvodu špatného ořezání obrázku či převrácení celého obrázku. Limitní hodnoty pro přijetí obrázku aplikací byly po dlouhém testování nastaveny na takové hodnoty, při kterých bude zaručeno správné zaměření všech dalších oblastí v obrázku, které se budou dále zpracovávat.

Třída Area je základním stavebním blokem a je využívána všemi dalšími třídami (kromě třídy ResizingImage) a realizuje nastavení souřadnic všech oblastí tvaru čtverce a výpočet průměrné hodnoty jasu v nich. Další třídy VerzeTestuAreaSet, OsobniCisloAreaSet, OdpovediOtazekAreaSet jsou si velmi podobné a dále zde popíšu jen třídu OsobniCisloAreaSet. Osobní číslo studenta se skládá celkem z šesti znaků. První znak je písmeno, které nabývá hodnot A, B, C, D, E, F. Dalších pět znaků jsou číslice, kde každá může nabývat hodnot 0, 1, . . . , 9. Třída CisloStudentaAreaSet má tedy celkem šest kolekcí listů oblastí pojmenované AreaSet1 až 6. Metody třídy slouží pro nastavení souřadnic každé oblasti (kolonka tvaru čtverce) z konfiguračního souboru, zařazení oblasti do listu oblastí AreaSet, vypočtení průměrné hodnoty jasu barvy všech oblastí v AreaSet, setřídění výsledků a určení oblasti s nejmenší hodnotou jasu.

Celý algoritmus zpracující obrazové informace (číslo studenta, verze testu, odpovědí) získané z naskenovaného formuláře testu a vyhodnocení správných odpovědí konkrétní varianty testu podle zvoleného testu z databáze najdeme v souborech CorrectTest.xaml.cs a CorrectTest.xaml.

8.5 Reporty

Pro tvorbu předloh pro tisk zadání testů, testového formuláře a výsledků testů jsem se rozhodl využít reportovacích nástrojů, které nabízí MS Visual Studio 2010 (dostupné i v starších verzích) a pro vlastní práci s vytvořenými reporty jsem se rozhodl využít komponentu Microsoft Report Viewer určenou pro aplikace běžících v prostření .NET Framework 4.

Do projektu (Obr. 17) bylo přidáno několik zdrojových souborů reportů pojmenovaných TestReport.rdlc, TestResultReport.rdlc a TestFormReport.rdlc. Soubory s příponou rdlc (z anglického Client Report Definition file) obsahují informace o rozvržení a použitých datech zobrazovaných v reportech, které jsou zpracovány prohlížečem MS ReportViewerem. Soubor rdlc je sám o sobě souborem typu XML. Pro první dva reporty, u kterých jsou zdrojová data načítaná z databáze, byly v projektu vytvořeny další zdrojové soubory dat tzv. datasety pojmenované DataSetGetTestData.xsd a DataSetGetResultData.xsd. Soubory s příponou xsd jsou taky soubory typu XML. Pro práci s oběma typy souborů lze ve Visual Studiu využít grafického návrháře či textového editoru. S každým datasetem je ještě spojen zdrojový soubor C# DataSetGetTestData.Designer.cs a DataSetGetResultData.Designer.cs. Pro naplnění datasetu požadovanými daty z tabulek databáze byly v databázi vytvořeny dvě uložené procedury.

V posledním souboru reportu TestFormReport.rdlc je zabudován obrázek ze složky Images\sablona.jpg. Nutné bylo zabezpečit nastavení poměrné výšky a šířky obrázku v reportu, aby nedocházelo při tisku či exportu dat v MS ReportVieweru ke změně velikosti obrázku, což by mělo za následek špatné vyhodnocení testu.

9 PŘEDSTAVENÍ APLIKACE TESTSTUDIO

Tato kapitola se zabývá v první části instalací a hardwarovými a softwarovými nároky pro instalaci a spuštění aplikace TestStudio. V dalších podkapitolách se podrobněji věnuji popisu možností práce a funkcí aplikace.

9.1 Softwarové nároky, instalace a spuštění aplikace

Před samotnou instalací a spuštěním aplikace TestStudio je nutné zmínit, že aplikace pro svůj běh potřebuje mít v počítači s operačními systémy MS Windows XP či novějšími MS Vista nebo MS Windows 7 nainstalováno prostředí Microsoft .NET Framework verze 4.0[6] a komponentu Microsoft ReportViewer[7], kterou aplikace TestStudio využívá pro práci s reporty. Zdroje pro instalaci MS ReportViewer nalezneme též přiložené ve složce *Instalace\ReportViewer* a instalaci spustíme souborem ReportViewer.exe, licenční podmínky .NET Framework 4 mi neumožňují přiložení instalačních zdrojů ,ale je možno je volně získat na webových stránkách[6] společnosti Microsoft.

Nejjednodušší způsob jak nainstalovat, spustit a otestovat aplikaci TestStudio bez jakékoliv konfigurace je následující. Na přiloženém disku CD-ROM v adresáři *Instalace\TestStudio* nalezneme soubor TestStudio.zip, který stačí rozbalit do libovolné složky na disku počítače. Po rozbalení souboru zip se nám v adresáři objeví dva soubory: TestStudio.exe a TestStudio.exe.config. Tyto nekomprimované soubory ale nalezneme na disku CD-ROM ve složce *Instalace\TestStudio\TestStudio*, odkud si je můžeme přímo zkopírovat do počítače. Aplikaci spustíme souborem TestStudio.exe. V tomto případě je ale nutné mít přístup k internetu, protože aplikace se připojuje k databázi umístěné na vzdáleném SQL Serveru v síti internet. Proto jsem si zřídil SQL Server hosting a pro připojení k tomutu SQL Serveru jsem aplikaci nakonfiguroval.

---

[6] MS .NET Framework 4 je možno volně získat na adrese:

http://www.microsoft.com/downloads/details.aspx?FamilyID=0a391abd-25c1-4fc0-919f-b21f31ab88b7&displaylang=en

[7] MS ReportViewer je možno volně získat na adrese:

http://www.microsoft.com/downloads/details.aspx?FamilyID=a941c6b2-64dd-4d03-9ca7-4017a0d164fd&displaylang=en

Pro spuštění aplikace bez přístupu k internetu je nutné mít v PC nebo lokální sítí nainstalovaný databázový server MS SQL Server 2008 Express[8] (či nějakou vyšší distribuci), do něj obnovit zálohu databáze, kterou aplikace TestStudio používá. Tato databáze se nachází na přiloženém disku CD-ROM v adresáři *Instalace\Databáze* v souboru *db890.bak*. Poté je nutno upravit konfigurační soubor aplikace *TestStudio.exe.config* podle konfigurace použitého SQL Serveru. Podrobnější informace o postupu konfigurace a obnovení databáze pro SQL Server jsou uvedeny v příloze P3. Tato příloha vznikla pro případ, aby bylo možno vytvořenou aplikaci kdykoliv v budoucnosti použít. V příloze nalezneme několik postupů pro obnovení databáze, konfigurace aplikace i návod pro instalaci samotného SQL Serveru.

Dalším způsobem, jak otestovat aplikaci TestStudio nad rámec jejího uživatele je prozkoumat celý projekt TestStudio přímo v nejnovějším vývojovém prostředí MS Visual Studio 2010[9], ve kterém jsem aplikaci vyvíjel. Celý projekt, všechny zdrojové kódy a soubory se nacházejí na disku CD-ROM v adresáři *Zdrojové kódy\TestStudio\TestStudio*. Projekt otevřeme souborem TestStudio.sln. Pro lepší orientaci v projektu vycházíme z kapitol této diplomové práce 7., 8., 9. Aplikace je zde opět nakonfigurována pro připojení ke vzdáleném SQL Serveru v síti internet a pro její sestavení a spuštění není třeba nic měnit. V případě využití jiného SQL Serveru, postupujeme podle návodu v příloze P3.

[8] MS SQL Server 2008 Express je možno získat na adrese:
http://www.microsoft.com/downloads/details.aspx?FamilyID=B5D1B8C3-FDA5-4508-B0D0-1311D670E336&displaylang=en

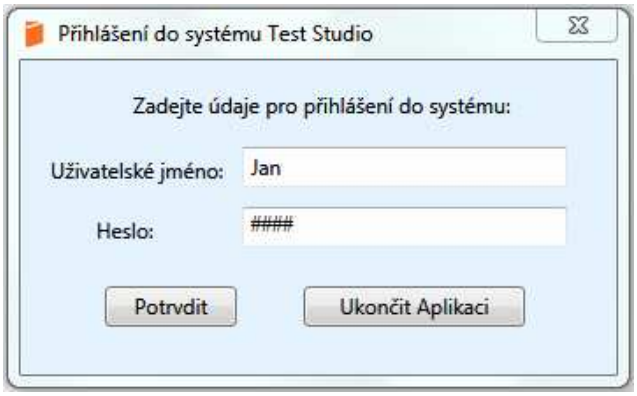
[9] MS Visual Studio 2010 je možno získat na adrese:
http://www.microsoft.com/downloads/details.aspx?FamilyID=06a32b1c-80e9-41df-ba0c-79d56cb823f7&displaylang=en

Po prvním spuštění aplikace zadejte následující údaje pro přihlášení. Tento uživatelský účet má práva administrátora aplikace.

Informace pro přihlášení uživatele po spuštění aplikace:

Uživatelské jméno: ***Jan***

Heslo: ***test***

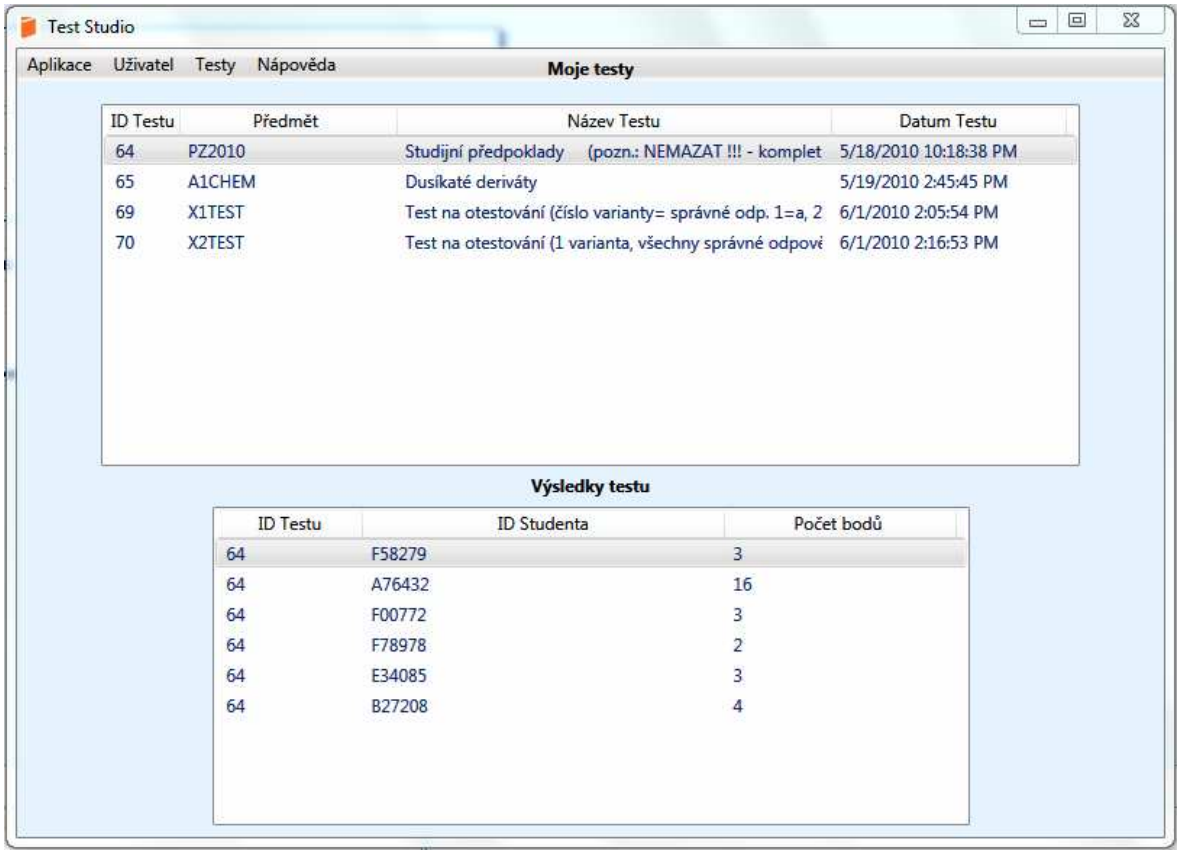


Obr. 21. Přihlašovací okno po spuštění aplikace[10]

9.2 Hlavní okno aplikace

Hlavní okno aplikace TestStudio se objeví až po úspěšném přihlášení a ověření uživatele v databázi (viz. ukázka v kapitole 8.1 a 9.1). Vidíme zde hlavní menu aplikace, jehož struktura a funkce jsou popsány v dalších podkapitolách. Po přihlášení uživatele se v horní tabulce *Moje Testy* načtou z databáze všechny testy přihlášeného uživatele. V dolní tabulce *Výsledky testu* vidíme všechny výsledky testů studentů označeného testu v tabulce *Moje testy*. Obrázek hlavního okna aplikace se nachází na další straně.

[10] Zdrojové soubory: Login.xaml , Login.xaml.cs



Obr. 22. Ukázka hlavního okna aplikace[11]

9.2.1 Struktura menu

Menu aplikace má následující strukturu položek

- Aplikace:
    - Odhlásit se
    - Ukončit aplikaci
- Uživatel:
    - Změna údajů (Změna všech informací a přihlašovacích údajů uživatele)
    - Vytvořit nový účet (Přidání nového uživatele do systému)
    - Vymazat účet (Odstranění uživatele ze systému)

[11] Zdrojové soubory: MainWindow.xaml , MainWindow.xaml.cs

- Testy:
    - Vytvořit nový test
    - Opravit test
    - Vymazat výsledek testu
    - Vymazat test
    - Předloha zadání testu
    - Předloha výsledků testu
    - Předloha formuláře testu
- Nápověda:
    - O aplikaci (Stručné informace o aplikaci - Obr.23)

Obr. 23. Základní informace o aplikaci[12]

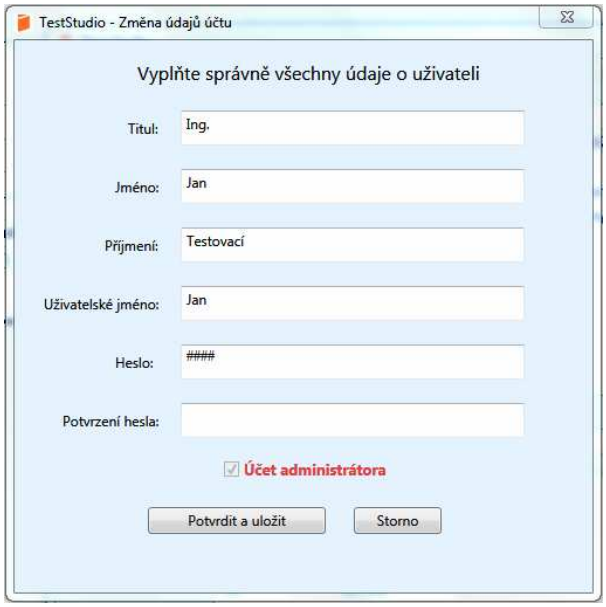
[12] Zdrojové soubory: About.xaml , About.xaml.cs

9.3 Správa uživatelů aplikace

Položka menu - *Uživatel* - *Vytvořit nový účet* je v systému k dispozici pouze uživatelům s právy administrátora. Takovýto uživatel může do systému přidávat nové uživatele, kterým může stejné oprávnění rovněž přidělit.

Položka menu - *Aplikace* - *Odhlásit se* umožní odhlášení uživatele ze systému, po kterém se zobrazí opět startovací okno pro přihlášení do systému.

Položka menu - *Aplikace* - *Ukončit aplikaci* jak název napovídá umožňuje ukončit aplikaci.



Obr. 24. Formulář pro změnu údajů uživatele
nebo přidání nového účtu [13]

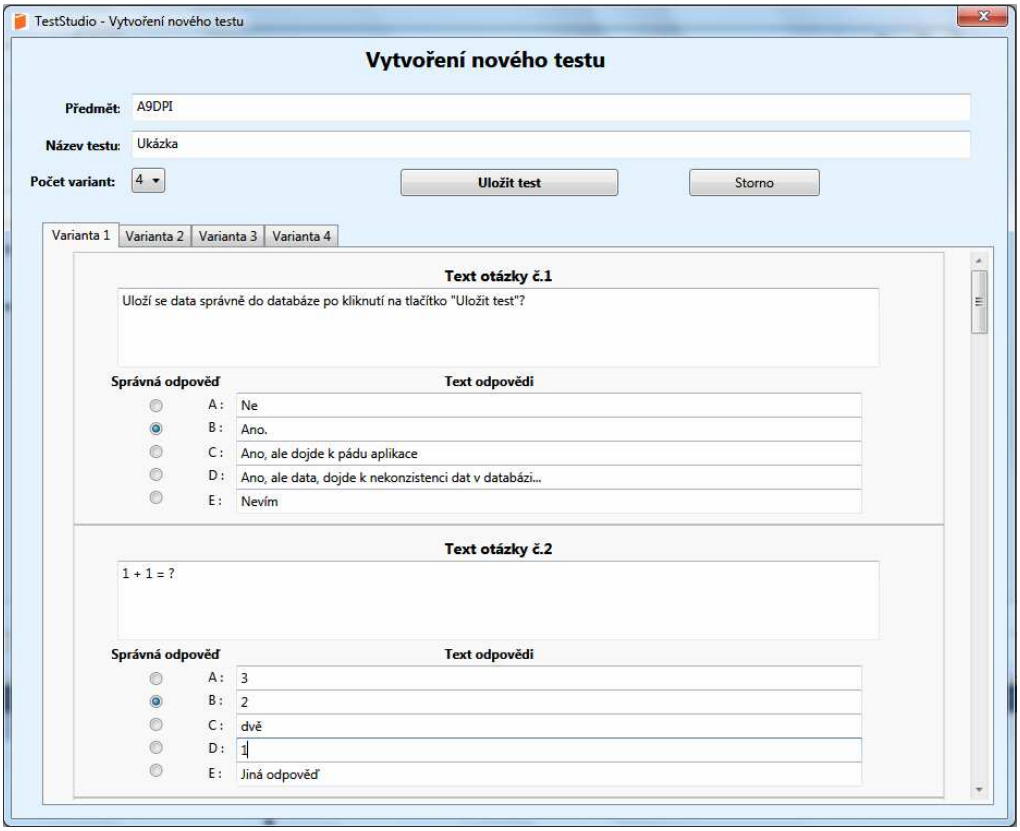
Po kliknutí myši na položku menu - *Uživatel* - *Změna údajů* se nám otevře nové okno obsahující formulář s předvyplněnými informacemi z databáze o přihlášeném uživateli včetně uživatelského jména a hesla. Nedají se zde měnit pouze oprávnění administrátora, která se dají přidělit pouze při vytvoření nového účtu uživatelem s administrátorským oprávněním. Odstranit uživatele ze systémů může pouze uživatel s administrátorským oprávněním po kliknutí na položku Menu - *Uživatel* - *Vymazat uživatele*. V listu uživatele

13 Zdrojové soubory: ManageAccount.xaml , ManageAccount.xaml.cs

označí, konkrétního uživatele (bez práv administrátora), kterého chce odebrat ze systému a klikne na tlačítko *Vymazat uživatele.* Po potvrzení dojde k vymazání uživatele a všech s ním souvisejících dat (testy, otázky, odpovědi, výsledky testů) z databáze.

9.4 Vytváření testů

Po kliknuti na položku *Menu - Test - Vytvořit Nový test* se otevře nové okno[14], kde si uživatel může vybrat počet variant testu. Minimální počet otázek v každé variantě testu je jedna otázka. Po vyplnění otázek, odpovědí a označení správné odpovědi a stisknutí tlačítka *Uložit test* dojde k uložení testu do databáze. Test se poté objeví v listu *Moje testy* a je možné s ním dále pracovat.

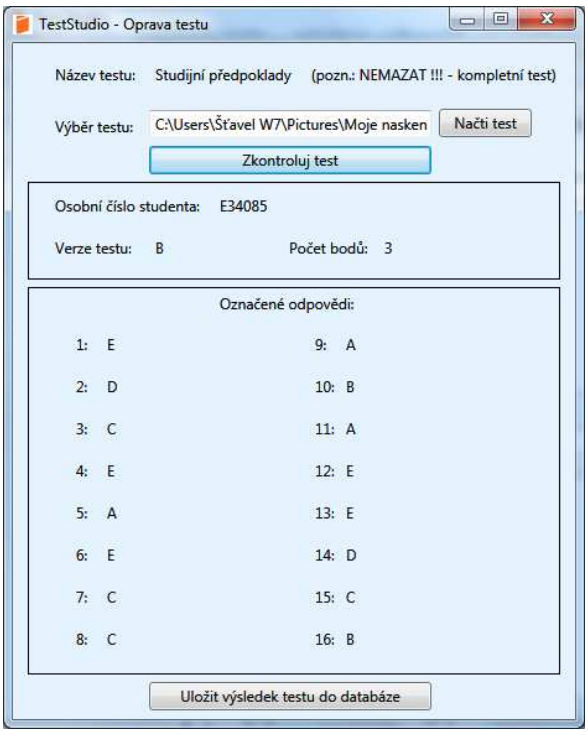


Obr. 25. Formulář pro vytvoření nového testu[14]

[14] Zdrojové soubory: ManageTest.xaml , ManageTest.xaml.cs

9.5 Vyhodnocení testu

Po označení konkrétního testu v tabulce *Moje testy* a kliknutím na položku Menu - *Test* - *Opravit test* se otevře nové okno aplikace, které je na následujícím obrázku Obr. 26. Po stisku tlačítka *Načti test* se objeví další dialog pro vybrání konkrétního souboru naskenovaného testu (je možné zobrazit soubory konkrétního typu obrázek JPG, obrázek BMP, obrázek TIF, obrázek PNG, obrázek GIF, nebo zvolit zobrazení všech typů souborů). Po vybrání souboru se zpřístupní tlačítko *Zkontroluj test*, jehož stisknutím se spustí vlastní vyhodnocení testu. Pokud zvolený test úspěšně projde validací dokumentu a varianta testu bude existovat v databázi, zobrazí se v okně studentské číslo, varianta testu , zatržené odpovědi studenta a celkový počet dosažených bodů(1 správná odpověď = 1 bod) v testu a zpřístupní se tlačítko *Uložit výsledek testu do databáze*, jehož stisknutím se výsledek uloží. Výsledky testů studenta a i celé testy lze z databáze vymazat tak, že označíme kliknutím myši konkrétní položku v tabulce *Moje testy* nebo *Výsledky testu* a z menu klikneme na položku *Test* - *Vymazat test* nebo *Vymazat výsledek testu*.

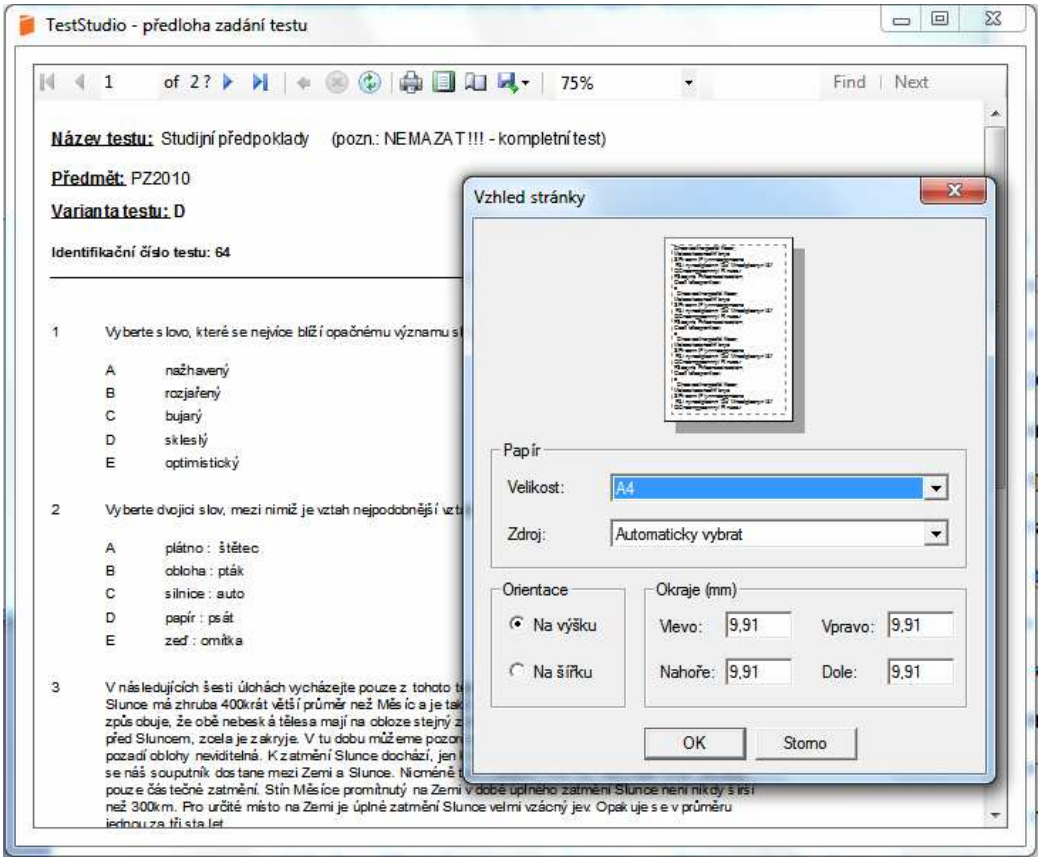


Obr. 26. Okno pro vyhodnocení testu[15]

[15] Zdrojové soubory: CorrectTest.xaml , CorrectTest.xaml.cs

9.6 Tisk a export dat

Po označení konkrétního testu v tabulce *Moje testy* a kliknutím na položku Menu - *Test* - *Předloha zadání testu* nebo *Předloha výsledků testu* či *Předloha formuláře testu* se otevře nové okno aplikace, které nám umožňuje tvorbu tiskových sestav zadání testů nebo výsledků testu či testového formuláře. Panel nástrojů umístěný nahoře okna nám kromě tvorby tiskových sestav nabízí další užitečné nástroje pro práci s konkrétní předlohou. Můžeme využít tiskových náhledů, nastavení vlastnosti dokumentů, vlastností pro tisk, textového vyhledávání v dokumentu, ale také export předloh do souborů formátů doc pro aplikaci MS WORD, xls pro MS EXCEL, či do přenosného formátu dokumentů pdf.

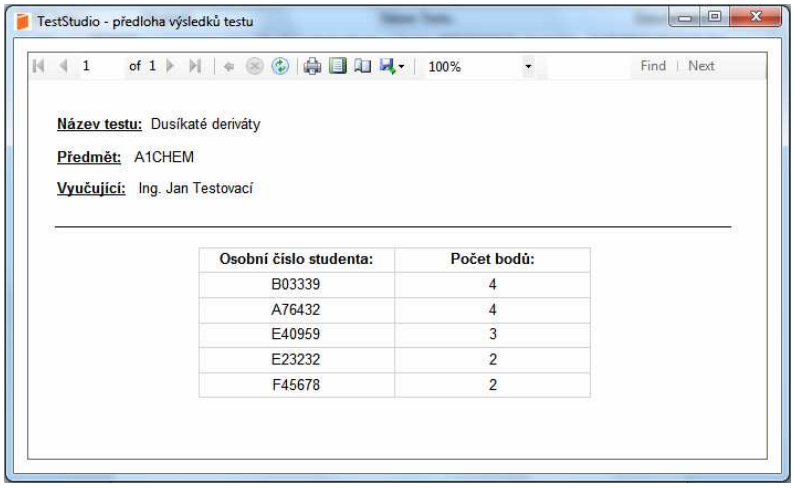


Obr. 27. Předloha zadání testu[16]

[16] Zdrojové soubory:

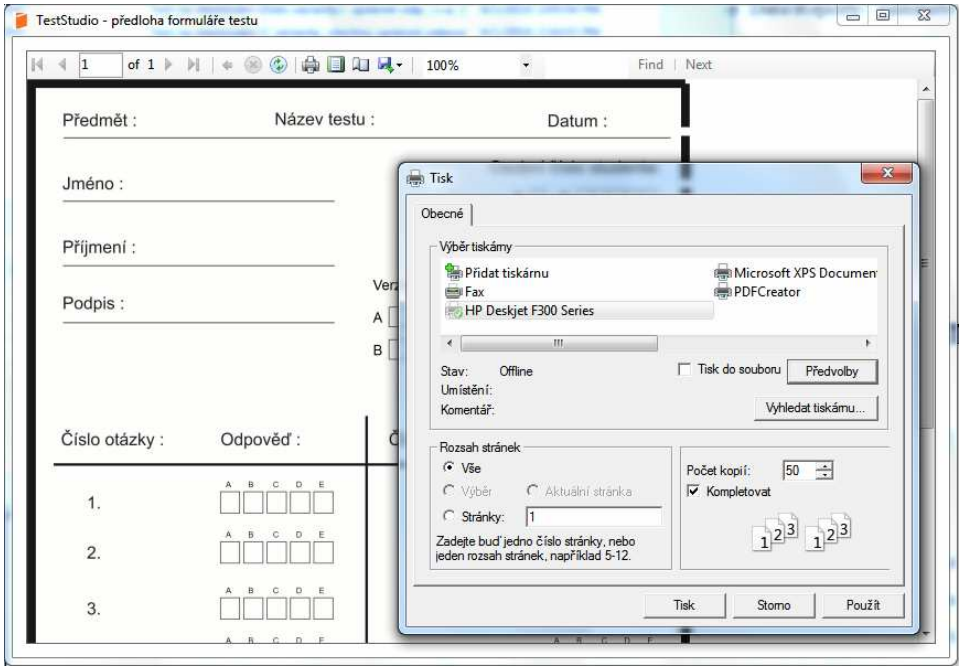
Předloha zadání testu - TestReport.xaml, TestReport.xaml.cs, ReportTestsQuestions.rdlc, DatasetGetTestData.xsd

Předloha výsledků testu - TestResultReport.xaml, TestResultReport.xaml.cs, ReportTestResults.rdlc, DatasetGetTestResult.xsd



Obr. 28. Předloha výsledků testu[16]

Vzhledem k různé délce textu otázek a odpovědí doporučuji předlohu zadání testu před samotným tiskem vyexportovat do dokumentu typu doc či xls a případně drobně upravit v aplikaci MS Word či EXCEL, aby nedocházelo k případům oddělení otázky a odpovědí na dva listy papíru. Pro tisknutí testových formulářů nedoporučuji export do dokumentu xls pro aplikaci MS EXCEL, protože poměr stran formuláře zde není zachován.



Obr. 29. Předloha testového formuláře [17]

[17] Předloha formuláře testu - TestFormReport.xaml, TestFormReport.xaml.cs, ReportTestAnswerSheet.rdlc

9.7 Data pro otestování aplikace

Pro otestování funkčnosti aplikace doporučuji pro přihlášení uživatelský účet s přihlašovacími údaji (uživatelské jméno: Jan, heslo: test) viz. kapitola 9.1. K tomuto účtu byly vytvořeny dva kompletní testy ve 4 variantách po 16 otázkách s přeházeným pořadím otázek i odpovědí, které se nazývají: Studijní předpoklady a Dusíkaté deriváty. Ke každému testu už bylo přidáno pár výsledků studentů. Databáze obsahuje i další uživatelské účty, testy a výsledky, na kterých je možno vyzkoušet operace vymazáni. Pro kompletní otestování aplikace doporučuji si vytvořit nový uživatelský účet(z právy administrátora), kompletní testy, vytisknout je na tiskárně, vyplnit, naskenovat, (případně oříznout okraje obrázku až na orámování formuláře testu, pokud to software dodávaný ke skeneru neumožňuje nastavit v předvolbách skenování nebo pokud jsou výsledky nepřesné) a zkontrolovat správné vyhodnocení aplikací TestStudio. Na přiloženém disku CD-ROM ve složce *Data\Vzory naskenovaných testů* naleznete naskenované obrázky vzorových testů. Jedná se o soubory typu: jpg, png, tif, bmp, gif. Obrázky byly skenovány s různým rozlišením 75, 100, 150, 200, 300 a 600 dpi. Obrázky jsou v různém rozlišení a byly skenovány jako obrázky nebo dokumenty s předvolbami skenování většinou v barevném režimu, ale taky černobílém režimu a odstínech šedi. Testy jsem tiskl a skenoval na několika zařízeních. U skenování testu doporučuji nastavit rozlišení 200dpi, skenovat v barevném režimu a nastavit automatické ořezávání dokumentu. Záleží na možnostech softwarového vybavení každého skeneru. Vzorové testy byly vyplněny různými psacími potřebami různých barev. Ve složce *Data\ Vzory naskenovaných testů \neakceptovatelné dokumenty* naleznete také několik souborů obrázků vytisknutých nebo naskenovaných příliš zkoseně, nebo špatně oříznutých, či převrácených, které aplikace TestStudio odmítne. Ve složce *Data\Reporty* naleznete ukázky vyexportovaných dokumentů aplikace vzorového testu ve formátech doc, xls, pdf.

ZÁVĚR

Hlavním cílem této diplomové práce bylo vytvořit software, který usnadní pedagogům práci s tvorbou písemných testů v papírové podobě a jejich automatickým vyhodnocením z opticky zpracovaného testového formuláře. V rámci řešení praktické části práce jsem tedy vytvořil aplikaci, která umožňuje vytvoření až čtyř různých variant testových textových otázek s možností celkem pěti odpovědí, z nichž je jedna jediná správná. Takto vytvořený test umožňuje aplikace archivovat v databázi, přímo vytisknout nebo exportovat do souboru dokumentů různých formátů. Zkoušený student dostane zadání testu spolu s prázdným testovým formulářem v papírové podobě. Do hlavičky testu formuláře napíše student své jméno, příjmení, datum, název testu, předmět testu a požadovaným způsobem zakřížkováním patřičných kolonek s možností korekce odpovědi vytmavením kolonky zanese do formuláře všechny své odpovědi na otázky, verzi testu a identifikační osobního čísla studenta. Takto vyplněné a podepsané formuláře testů studentů projdou procesem skenování a výsledné soubory obrázků budou vstupními daty aplikace, která podle označené varianty testu srovná vybrané odpovědí se správnými odpověďmi z databáze, spočítá celkový počet dosažených bodů a spolu s označeným číslem studenta uloží záznam do databáze. Výsledky všech studentů určitého testu umožňuje aplikace opět vytisknout nebo vyexportovat do souboru. Veškerá data související se správou testů a účtů uživatelů aplikace jsou ukládána do databáze.

Nejdůležitějším cílem při tvorbě aplikace bylo zaručit spolehlivé vyhodnocení údajů z opticky zpracovaného papírového testového formuláře. Tohoto cíle bylo dosaženo, ale je nutné zmínit, že pro soubory obrázků předkládané aplikaci platí předpoklad, že hranice obrázku musí s danou mírou tolerance odpovídat hranici orámování testového formuláře. Orámování testového formuláře bylo záměrně navrženo tak, aby byly tyto hrany softwarem skenovacího zařízení spolehlivě detekovatelné a mohlo dojít k automatickému ořezání oblastí mimo tento rám. V případě, že by nebyl tento předpoklad splněn, musí zasáhnout uživatel a obrázek upravit manuálně.

Pro vývoj aplikace jsem si zvolil programovací jazyk C# a s ním související platformu .MS NET Framework. Jako vývojové prostředí jsem si zvolil nejnovější verzi MS Visual Studio 2010 a jako databázový systém MS SQL Server 2008. Vytvořená aplikace je v souvislosti se zvolenými technologiemi určena pouze pro operační systémy

MS Windows a byla vyvíjena a testována v operačních systémech MS Windows XP, MS Windows Vista, MS Windows 7.

V teoretické části práce jsem se věnoval popisu všech pojmů bezprostředně souvisejících s charakterem vytvořené aplikace z oblasti počítačové grafiky a databázových systémů. Byly zde popsány možnosti vývoje aplikací pro operační systémy MS Windows se zvolenými vývojovými technologiemi .NET Framework, použité programovací jazyky a vývojové prostředí MS Visual Studio 2010 a MS SQL Server 2008.

V praktické části jsem shrnul a popsal poznatky o vytvořeném softwarovém řešení, vytvořenou databázi, formulář testu, nejdůležitější algoritmy zpracování obrazových dat a použité technologie .NET Framework a jiné. Teoretická i praktická textová část byla doplněna několika příklady ukázek zdrojových kódů a byla koncipována tak, aby usnadnila orientaci ve struktuře zdrojových souborů projektu vytvořené aplikace, které jsou přiloženy na disku CD-ROM. Závěrečná kapitola praktické části pak rozebírá HW/SW nároky, instalaci vytvořené aplikace a poslouží také jako jednoduchý návod pro práci s aplikací.

ZÁVĚR V ANGLIČTINĚ (CONCLUSION)

The main goal of this master thesis was to develop a software product, helping educators to create paper tests, which would be automatically optically processed. In the practical part of the thesis I created an application, which enables creating up to four versions of text test questions with possibility of up to five answers and only one is correct. A test creating this way can be stored within the application database, directly printed or exported into various document formats. Student, taking the test, gets the test submission and a blank test form. The student fills the form header with his first and second name, date, title of the test, subject, then taking the test with answers by filling in the cells, chooses version of the test and enters his/her personal ID. Forms filled this way are scanned and consequent image files make up input data for the application, which compares the filled in answers with the correct answers in the database according to the test variant and along with the student's ID sends the final point count to the database. Test results for all students taking a specific test can be printed or exported to a file. All the data related to test and user account management are saved in the database.

The most important matter in the application development was to ensure reliable evaluation of values from the optically processed paper test form. This goal has been achieved, but it is necessary to mention, that the images used by the application, must have their border matching the border line of the test form within certain toleration. Borders of the test form have been designed on purpose to be detected by every scanning device and therefore the areas outside the borderlines can be automatically cropped. If this condition is not met, the user must interfere to correct the image manually.

I have chosen C# programming language and its related .NET Framework platform, the newest MS Visual Studio 2010 as development environment and MS SQL Server 2008 as database system. The application can be therefore used only in MS Windows operating system and was developed and tested in MS Windows XP, MS Windows Vista, and MS Windows 7.

In the theoretical part of the thesis I focused on description of all the terms related closely to the nature of the application in the field of computer graphics, database systems, MS Windows application development, .NET Framework technology, programming languages and MS Visual Studio 2010 and MS SQL Server environments.

In the practical part I described issues of the software solution, the database, test form, the most important algorithms of image processing, .NET Framework technology etc. Both theoretical and practical parts were complemented by some examples of source code of the application, which are placed on CD-ROM disc. The final chapter of the practical part resolves HW/SW requirements, installation of the application and serves as a simple manual for the application.

SEZNAM POUŽITÉ LITERATURY

Monografie:

[1] **BAYER , Jürgen. C# 2005 Velká kniha řešení . 1. vyd.** Brno : Computer Press, a.s., **2007**. 813 s., 1 CD-ROM. **ISBN 978-80-251-1620-3.**

[2] **DOBEŠ , Michal.** ***Zpracování obrazu a algoritmy v C#.*** **1. vyd.** Praha : BEN - technická literatura, **2008.** 144 s. **ISBN 978-80-7300-233-6.**

[3] **GROFF, James R.; WEINBERG, Paul N.** ***SQL kompletní průvodce***. 1.vydání. Brno : CP Books, a.s., **2005**. 936 s. **ISBN 80-251-0369-2.**

[4] **HOTEK, Mike**. ***Microsoft SQL Server 2008 : Krok za krokem***. 1. vydání. Brno : Computer Press, a.s., **2009**. 488 s. ISBN **978-80-251-2466-6**.

[5] **MARTIŠEK, Dalibor. Matematické principy grafických systémů**. 1. vyd. Brno : Littera, **2002**. 278 s. **ISBN 808-57-6319-2.**

[6] **NAGEL, Christian , et al. C# 2008 Programujeme profesionalně**. 1. vyd. [s.l.] : Computer press, **2008**. 2 sv. (100, 772 s.). **ISBN 978-80-251-2401-7**.

[7] **POKORNÝ Pavel. Základy počítačové grafiky. 1. vydání.** Zlín: Univerzita Tomáše Bati ve Zlíně. **2004**. 120 s. ISBN **80-7318-161-4**

[8] **ROBINSON, Simon**, et al. ***C# Programujeme profesionálně***. 1.vydání. Brno : Computer Press, a.s., **2003**. 1130 s. **ISBN 80-251-0085-5**.

[9] **SHARP, John. Microsoft Visual C# 2008 : Krok za krokem**. 1. vyd. Brno : Computer Press, a.s., **2008**. 592 s., 1 CD-ROM. **ISBN 978-80-251-2027-9**.

[10] **Solid Quality Learning.** ***Microsoft SQL Server 2005 : Základy databází - krok za krokem***. 1.vydání. [s.l.] : Computer Press, a.s., **2007**. 320 s. **ISBN 978-80-251-1524-4**.

[11] **TROELSEN, Andrew.** ***C# a .NET 2.0 Profesionálně***. 1.vydání. Brno : Zoner Press, **2006**. 1197 s. **ISBN 80-86815-42-0**.

[12] **ŽÁRA, Jiří.** ***Moderní počítačová*** *grafika*. 1. vyd. Brno : Computer Press, a.s., **2004**. 609 s. **ISBN 80-251-0454-0.**

Akademické práce:

[13] **MELUZÍN, Pavel .** ***Informační systém firmy***. Zlín, 2006. 55s, VIII s. Bakalářská práce. UTB ve Zlíně, FAI.

Internetové zdroje:

[14] **BŘEZINA, Jan. Co znamená jednotka DPI a kde se s ní můžeme setkat?.** *Grafika Publishing* [online]. **2002,** 1, [cit. 2010-04-01]. Dostupný z WWW: <http://www.grafika.cz/art/photoshop/DPI_pojmy.html>**. ISSN 1212-9569**.

[15] **Computer printer In *Wikipedia*** *: the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, , [cit. 2010-04-04]. Dostupné z WWW: **<http://en.wikipedia.org/wiki/Computer_printer>**

[16] **C Sharp In *Wikipedia :*** *the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, , [cit. 2010-04-07]. Dostupné z WWW: **<http://cs.wikipedia.org/wiki/C_Sharp>.**

[17] **.NET Framework In *Wikipedia*** *: the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 18. 11. 2005, [cit. 2010-04-04]. Dostupné z WWW: **<http://cs.wikipedia.org/wiki/.NET_Framework>.**

[18] **.NET Framework In *Wikipedia*** *: the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, , [cit. 2010-04-04]. Dostupné z WWW: **<http://en.wikipedia.org/wiki/.NET_Framework>.**

[19] **HORVÁTH, Tomáš.** *Programujet.com* [online]. 10. 12. 2008 [cit. 2010-04-05]. **.NET Framework**. Dostupné z WWW: **<http://programujte.com/?akce=clanek&cl=2008120700-net-framework>.**

[20] **Relační databáze In *Wikipedia*** *: the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, , [cit. 2010-04-11]. Dostupné z WWW: **<http://cs.wikipedia.org/wiki/Relační_databáze>.**

[21] **Visual Studio In *Wikipedia*** *: the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, , [cit. 2010-05-01]. Dostupné z WWW: **<http://cs.wikipedia.org/wiki/Microsoft_Visual_Studio>.**

[22] **Vývojové prostředí In *Wikipedia*** *: the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, , [cit. 2010-04-04]. Dostupné z WWW: **<http://cs.wikipedia.org/wiki/Vývojové_prostředí>.**

Ostatní obecné internetové zdroje z oblasti programování:

[23] ***Microsoft Developer Network*** [online]. 2010 [cit. 2010-05-09]. Dostupné z WWW: **<http://msdn.microsoft.com/cs-cz/vcsharp/default%28en-us%29.aspx>.**

[24] ***C# Corner*** [online]. 2010 [cit. 2010-05-09]. Dostupné z WWW: **<http://www.c-sharpcorner.com/>**

[25] ***CodeGuru*** [online]. 2010 [cit. 2010-05-09]. Dostupné z WWW: **<http://www.codeguru.com/>.**

[26] ***The Code Project*** [online]. 2010 [cit. 2010-05-09]. Dostupné z WWW: **<http://www.codeproject.com/>.**

[27] ***Programujte*** [online]. 2010 [cit. 2010-05-09]. Dostupné z WWW: **<http://programujte.com/>.**

[28] ***Vývojář.cz : Vývojáři sobě...*** [online]. 2010 [cit. 2010-05-09]. Dostupné z WWW: **<http://www.vyvojar.cz/>.**

Dvě monografie se seznamu odborné literatury ze zadání práce (strana 3) použity nebyly:

3. NAVRÁTIL, Pavel. *Počítačová grafika a multimédia*. 1. vyd. Kralice na Hané : Computer Media a.s., 2007. 112 s. ISBN 80-86686-77-9 .

4. NAVRÁTIL, Pavel. *Počítačová grafika a multimédia*. 1. vyd. Kralice na Hané : Computer Press, a.s., 2007. 50 s., CD-ROM. ISBN 978-80-86686-79-0.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Rozhraní pro programování aplikací (zkratka **A**pplication **P**rogramming **I**nterface)
CLR	Běhové prostředí .NET(zkratka **C**ommon **L**anguage **R**untime)
CLS	Společná specifikace jazyků(zkratka **C**ommon **L**anguage **S**pecifiacation)
CTS	Společný typový systém(zkratka **C**ommon **T**ype **S**ystem)
DBMS	Systém řízení báze dat (zkratka **D**ata **B**ase **M**anagement **S**ystem)
DIP	Obrazové zpracování dokumentů(zkratka **D**ocument **I**mage **P**rocessing)
dpi	Jednotka používaná k určení množství obrazových bodů, které se vejde do jednoho palce z anglického **D**ots **P**er **I**nch.
doc	Typ souboru pro aplikace MS WORD.
ppi	Jednotka používaná k určení rozlišení rastrového obrazu nebo zobrazovacího zařízení (např. monitoru nebo scanneru) - pixely na palec z anglického **P**ixels **P**er **I**nch)
RGB	Barevný model RGB z anglického **R**ed-**G**reen-**B**lue neboli červená-zelená-modrá, je to aditivní způsob míchání barev používaný v zobrazovacích zařízení.
USB	Označení univerzální sériové sběrnice z anglického **U**niversal **S**erial **B**us
XML	Značkovací jazyk, zkráceně XML (zkratka **E**xtensible **M**arkup **L**anguage)
SQL	Strukturovaný dotazovací jazyk (zkratka **S**tructured **Q**uery **L**anguage)
ECMA	Mezinárodní standardizačními komise pro informační a komunikační systémy
C#	Programovací jazyk C# (sharp)
LINQ	Integrovaný jazyk pro dotazování (zkratka **L**eanguage **In**tegrated **Q**uery)
XAML	Značkovací jazyk XML pro aplikace (zkratka **X**ML for **A**pplications **M**arkup **L**anguage)
OMR	Optické rozpoznávání znaků (zkratka **O**ptical **M**ark **R**ecognition)
OCR	Optické rozpoznávání znaků (zkratka **O**ptical **C**haracter **R**ecognition)

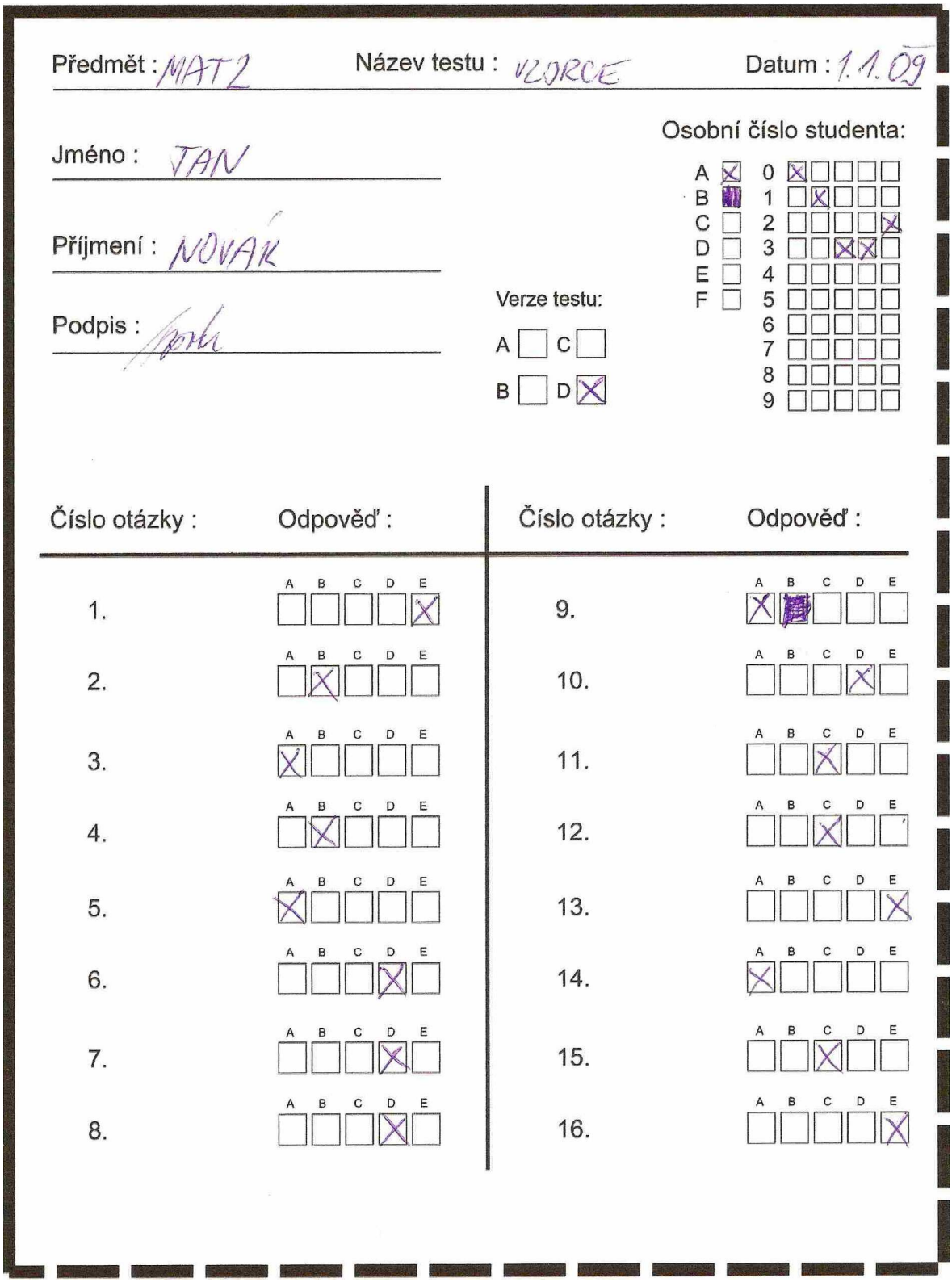
MS	Zkratka společnosti **M**icro**s**oft
MSDN	(zkratka **M**icro**s**oft **D**eveloper **N**etwork)
MB	Jednotka informace (zkratka **M**ega **B**yte)
IDE	Integrované vývojové prostředí (zkratka z **I**ntegrated **D**evelopment **E**nvironment
SŘBD	**S**ystém **ř**ízení **b**áze **d**at
ISO	Mezinárodní organizace pro normalizaci (zkratka **I**nternational **O**rganization for **S**tandardization)
EA	Zkratka softwaru **E**nterprise **A**rchitekt (http://www.sparxsystems.com)
pdf	Soubor typu **P**ortable **D**ocumet **F**ormat(přenosný formát dokumentu nezavislý na software i hardware vyvinutý firmou Adobe)

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P 2: UKÁZKA VYPLNĚNÉHO FORMLÁŘE TESTU



Předmět : MAT 2 Název testu : VZORCE Datum : 1.1.09

Jméno : JAN

Příjmení : NOVÁK

Podpis : Novák

Osobní číslo studenta:

A ☒	0 ☒	☐	☐	☐	☐
B ☒	1 ☐	☒	☐	☐	☐
C ☐	2 ☐	☐	☐	☐	☒
D ☐	3 ☐	☐	☒	☒	☐
E ☐	4 ☐	☐	☐	☐	☐
F ☐	5 ☐	☐	☐	☐	☐
	6 ☐	☐	☐	☐	☐
	7 ☐	☐	☐	☐	☐
	8 ☐	☐	☐	☐	☐
	9 ☐	☐	☐	☐	☐

Verze testu:

A ☐ C ☐

B ☐ D ☒

Číslo otázky :	Odpověď : A B C D E	Číslo otázky :	Odpověď : A B C D E
1.	☐ ☐ ☐ ☐ ☒	9.	☒ ☒ ☐ ☐ ☐
2.	☐ ☒ ☐ ☐ ☐	10.	☐ ☐ ☐ ☒ ☐
3.	☒ ☐ ☐ ☐ ☐	11.	☐ ☐ ☒ ☐ ☐
4.	☐ ☒ ☐ ☐ ☐	12.	☐ ☐ ☒ ☐ ☐
5.	☒ ☐ ☐ ☐ ☐	13.	☐ ☐ ☐ ☐ ☒
6.	☐ ☐ ☐ ☒ ☐	14.	☒ ☐ ☐ ☐ ☐
7.	☐ ☐ ☐ ☒ ☐	15.	☐ ☐ ☒ ☐ ☐
8.	☐ ☐ ☐ ☒ ☐	16.	☐ ☐ ☐ ☐ ☒

PŘÍLOHA P 3: OBNOVENÍ DATABÁZE NA SQL SERVERU A KONFIGURACE APLIKACE (1. ČÁST)

Následující příloha poslouží jako návod pro změnu konfigurace připojení aplikace k SQL Serveru, návod pro obnovení databáze na SQL Serveru a návod pro instalaci SQL Serveru, aby bylo aplikaci možno kdykoliv v budoucnosti použít.

Změna konfigurace

Vytvořená aplikace je standardně nakonfigurována pro připojení k SQL SERVER 2008 provozovaného k webhostingu: http://pavelmeluzin.cz.windows4.aspone.cz/

Aplikace využívá několik různě pojmenovaných připojovacích řetězců nesoucích ale stejné informace o připojení. Pro přihlášení k SQL existují dva způsoby autorizace uživatele: 1. SQL Server authentication pro připojení ke vzdálenému SQL Serveru v lokální síti či síti internetu a 2. Windows Authentication pro připojení k lokálnímu SQL Serveru nainstalovaného v počítači. Zde vidíme použitý způsob pro SQL Server Autentitication.

Úplná konfigurace připojení v souboru TestStudio.exe.config (app.config) je následující:

```
<connectionStrings>
    <add name="DefaultCS" connectionString="Data Source=sql3.aspone.cz;Initial
Catalog=db890;Persist Security Info=True;User ID=db890;Password=dp1meluzin2x"
      providerName="System.Data.SqlClient" />
    <add name="TestStudio.Properties.Settings.DefaultCS" connectionString="Data
Source=sql3.aspone.cz;Initial Catalog=db890;Persist Security Info=True;User
ID=db890;Password=dp1meluzin2x"
      providerName="System.Data.SqlClient" />
    <add name="TestStudio.Properties.Settings.db890ConnectionString"
      connectionString="Data Source=sql3.aspone.cz;Initial Catalog=db890;Persist Security
Info=True;User ID=db890;Password=dp1meluzin2x"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
```

Pokud se rozhodneme databázi provozovat na SQL Serveru s autorizací SQL Server Autenthication musíme změnit tyto údaje(označeny červeným písmem pro každý připojovací řetězec zvlášť) podle použité konfigurace:

```
connectionString="Data Source=sql3.aspone.cz;Initial Catalog=db890;Persist Security
Info=True;User ID=db890;Password=dp1meluzin2x"
```

Adresa SQL serveru:	Data Source =**sql3.aspone.cz**
Název databáze na serveru:	Initial Catalog = **db890**
Nastavení zabezpečení:	Persist Security Info = True
Identifikační uživatelské jméno:	ID = **db890**
Heslo uživatele:	Password = **dp1meluzin2x**

PŘÍLOHA P 3: OBNOVENÍ DATABÁZE NA SQL SERVERU A KONFIGURACE APLIKACE (2. ČÁST)

Pokud se rozhodneme databázi provozovat na lokálním SQL Serveru, nejspíš použijeme druhý způsob Windows Authentication. (Samozřejmě ale záleží jaký způsob ověřování nastavíme, můžeme použít i SQL Server Authentication).

V následující ukázce konfigurace je použita konfigurace k lokálnímu SQL Serveru v počítači, předpokládáme autorizaci typu Windows Authentication:

```
<connectionStrings>
    <add name="DefaultCS"
connectionString="Server=LENOVO\SQLEXPRESS;Initial
Catalog=db890;Trusted_Connection=True;"
      providerName="System.Data.SqlClient" />
    <add name="TestStudio.Properties.Settings.DefaultCS"
connectionString="Server=LENOVO\SQLEXPRESS;Initial
Catalog=db890;Trusted_Connection=True;"
      providerName="System.Data.SqlClient" />
    <add name="TestStudio.Properties.Settings.db890ConnectionString"
      connectionString="Server=LENOVO\SQLEXPRESS;Initial
Catalog=db890;Trusted_Connection=True;"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
```

Zaměřme opět pozornost na sekci:
`connectionString="Server=LENOVO\SQLEXPRESS;Initial Catalog=db890;Trusted_Connection=True;"`
kde musíme změnit údaje podle nastavené konfigurace (označeny červeným písmem):

Název SQL Serveru: Server = **LENOVO\SQLEXPRESS**

(Název počítače\Název SQL Serveru)

Název databáze: Initial Catalog = **db890**

Nastavení zabezpečení: Trusted_Connection = True

Je nutné změnit všechny připojovací řetězce.

Výše uvedené ukázky konfigurace považuji za nejčastěji používané. Samozřejmě záleží na zvolené konfiguraci SQL Serveru a na názvu databáze. Všechny možnosti konfigurace připojovacího řetězce lze nalézt na adrese:

http://www.connectionstrings.com/sql-server-2008.

PŘÍLOHA P 3: OBNOVENÍ DATABÁZE NA SQL SERVERU A KONFIGURACE APLIKACE (3. ČÁST)

Instalace SQL Serveru 2008Express with Advanced Services:

Podrobný postup instalace SQL Serveru zde popisovat nebudu, ale uvedu zde jen několik spolehlivých odkazů včetně videoprůvodce.

Videoprůvodce instalací SQL Serveru v českém jazyce:

(Základní kroky při instalaci SQL Server 2008. Určeno pro úplné začátečníky bez zkušeností s produktem SQL Server 2008.)

http://www.mstv.cz/vyvojari/videos/104/SQL-Server-2008---instalace

http://www.aspnet.cz/Articles/253-video-instalace-microsoft-sql-serveru-2008.aspx

Na disku CD-ROM ve složce Instalace\SQL Server 2008 Express\ je umístěn dokument:

SQL2008_Express_install.doc - podrobný postup instalace v českém jazyce

(internetový zdroj: https://forum.helios.eu/orange/forum/download/file.php?id=143&sid=4b17998a5d2f91ca0b2d4de1beb52d41)

Oficiální stránky společnosti MS:

http://msdn.microsoft.com/cs-cz/library/dd981045%28v=SQL.100%29.aspx

Obnovení databáze ze souboru db890.bak

Obnovení databáze lze provést několika způsoby, podrobné informace jsou uvedeny v odkazech.

Obnovení databáze pomocí SQL Server Management Studio:

http://msdn.microsoft.com/cs-cz/library/ms177429.aspx

http://mersite.cz/BlogDetail.aspx?StringId=blogBackupSQL

Obnovení a přesun databáze pomocí Database Publishing Wizard:

http://wiki.aspone.cz/Jednoduch%C3%BD_p%C5%99enos_datab%C3%A1ze_z_Va%C5%A1eho_po%C4%8D%C3%ADta%C4%8De_na_ASPone_SQL_server