VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

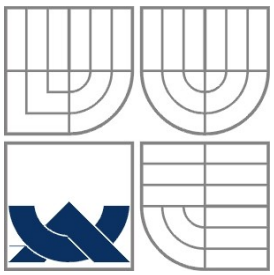
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# SYSTÉM ARCHIVACE DOKUMENTŮ

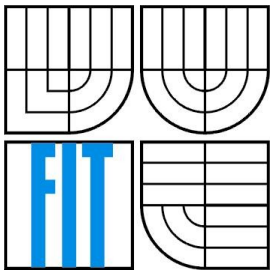
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE David Grochol
AUTHOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ VBRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# SYSTÉM ARCHIVACE DOKUMENTŮ

DOCUMENT ARCHIVING SYSTEM

**BAKALÁŘSKÁ PRÁCE**

BACHELOR'S THESIS

AUTOR PRÁCE — David Grochol
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jiří Ševcovic
SUPERVISOR

BRNO 2012

# Abstrakt

Cílem této bakalářské práce je vytvoření systému pro převod papírových archívů do digitální podoby, vytváření nových archívů a správa těchto archívů. Aplikace umožňuje získávání obrazových dat pomocí externího zařízení a ukládání do databáze. Důležitou požadovanou vlastností je práce v režimu online i offline. Implementace aplikace je v jazyce C#. Aplikace po spuštění spolupracuje se serverem MySQL a pro sdílení dat využívá aplikaci třetí strany.



# Klíčová slova

Archivace dokumentů, MySQL, C#, TWAIN



# Abstract

The goal of this thesis is to create a system for conversion of paper archives into digital form, creating new archives and management of these archives. Application allows to collect image data using external device and storing it into the database. Another important feature is working in online and offline mode. Application is implemented in C# programming language. After start it cooperates with MySQL server and it uses third party application for sharing data.



# Keywords

Document Archiving, MySQL, C#, TWAIN



# Citace

Grochol David: *Systém archivace dokumentů*, bakalářská práce, Brno, FIT VUT v Brně, 2012

# Systém archivace dokumentů

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jiřího Ševcovice.
Další informace mi poskytl Ing. Jiří Pokorný.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
David Grochol
16. května 2012

## Poděkování

Tímto bych chtěl poděkovat svému vedoucímu Ing. Jiřímu Ševcovicovi za vedení, nápady a podněty při vytváření práce. Dále bych chtěl poděkovat mým blízkým za podporu.

# Obsah

# 1. Úvod

Archivace dokumentů je nutná pro řadu osob, jak fyzických, tak právnických. V některých případech je přímo stanoveno legislativou státu po jakou dobu musí být dané dokumenty uchovány, v některých případech až 10 let. U osobních fotoalb je tato doba daleko delší. Možnosti jak archivovat dokumenty je více, jednou z možností je uchovávat všechny dokumenty v papírové podobě. Tato možnost je náročná na prostor, časovou stálost, ochranu před zničením a v neposlední řadě také lidské zdroje, které se starají o správu archívu. Další možností je tyto dokumenty převést do elektronické podoby. To umožňuje lepší správu celého archívu, rychlejší vyhledání požadovaného dokumentu. Snadnější zabezpečení proti poškození nebo zničení. Samozřejmě může dojít k poškození elektronického archívu vlivem přírodní katastrofy, nebo technického problému. Pokud je však prováděná pravidelná záloha dat, můžeme obnovit archiv během chvíle s minimální ztrátou dat. Oproti papírovému archívu, kde je záloha prakticky nemožná. Poškození archívu má většinou katastrofální následky na data. Z toho důvodu je výhodnější používat digitální archív nebo kombinaci obou typu archívu.

Cílem této práce je vytvořit systém, který umožňuje převod papírové databáze do digitální podoby, vytvářet nové a editovat tyto databáze. Dále možnost převodu digitální databáze do papírové podoby. Systém má fungovat v režimech online i offline vzhledem ke vzdálenému serveru.

Druhá kapitola obsahuje možnosti jakými je v dnešní době možno digitálně archivovat dokumenty a porovnání jejich výhod. Dále popis technologií, které jsou využity pro vývoj a běh systému. Ve třetí kapitole se věnuji analýze a specifikaci požadavků. Je zde definována cílová skupina uživatelů a možnosti aplikace. Další kapitola obsahuje návrh řešení, to obsahuje návrh databáze ve formě ER diagramu, návrh uživatelského rozhraní a návrh vnitřní struktury aplikace. Předposlední kapitola obsahuje popis implementace vybraných části aplikace.

# 2. Použité technologie

Na úvod této kapitoly bude představen vývoj sytému pro archivaci a dnešní možnosti. Následuje popis technologii pro vývoj aplikace a technologie, které aplikace využívá po spuštění. Aplikace bude implementována v jazyce C#, vyžadující .NET Framework pro běh aplikace. Pro získávání obrazových dat (převod dokumentu na obrázky) je využito protokolu TWAIN. Jako databázový systém je zvolen MySQL. Pro komunikaci s ním slouží jazyk SQL. Sdílení obrazových dat je realizováno pomocí služby třetí strany. V kapitole 2.6 jsou specifikovány požadavky aplikace na tuto službu a srovnání některých dnes dostupných služeb.

## 2.1 Archivace

Archivování dokumentů má dlouhou historii, nejstarší archív v českých zemích pochází ze 12. stol.[1] Postupem času se rozšiřoval rozsah a typy dokumentů, které se archívují. V posledních letech s rozšířením počítačů, vznikla možnost vytvářet digitální archívy. Aby bylo možno převést i stávající papírové archívy do digitální podoby, bylo nutno vyvinout zařízení pro převod dokumentů do digitální podoby, pro tento účel slouží skener [2], první byl vyvinut již v roce 1957. Další potřebnou technologií pro vytváření digitálních archívů je úložiště dat, k ukládání dat dnes především slouží pevné disky, dříve to byly také magnetické pásky. S postupem času se zlepšovaly technologie pro výrobu disků a zápis dat a dnes jsou běžné disky s kapacitou v řádech TB. Aby bylo možno digitální archív spravovat je nutno řadit data do určité hierarchie. K tomuto účelu slouží databáze [3], jejich vývoj začal v 50. letech 20. století. Dnes existuje celá řada databázi různého typu. Vývoj a vhodná kombinace těchto technologií umožnil vznik digitálních archívů.

Jak bylo řečeno v kapitole 1., archívy jsou dnes nutností nebo usnadněním v řadě oborů. Možnosti digitální archivace velice rozšiřuje rozvoj internetu, pomocí něj je možno archívy sdílet teoreticky s celým světem.

Možnosti jak vytvořit a spravovat vlastní digitální archiv jsou ve své podstatě dvě. Jednou z možností je využití služby a přenechat správu externí společnosti [4,5]. Tato možnost je v této práci uvedena pouze pro úplnost. Druhá možnost je zakoupení archivačního software [6,7] a spravovat archív samostatně. To má umožňovat výsledná aplikace této práce. Komerční aplikace mají řadu funkcí jako různé způsoby vkládání dat, úprava mazání. Mezi speciálnější a ne vždy podporované patří verzování dokumentů, převod obrazových dokumentů na text nebo mazání dokumentů po určité době. Aplikace také umožňují dokoupení pluginů s různou funkčností. Velkou nevýhodou zmíněných aplikací je, že neumožňují práci offline. To znamená, že uživatel musí mít vždy dostupný server s archívem a bez něj nemůže pracovat.

## 2.2 Protokol TWAIN

Tato kapitola obsahuje popis vývoje TWAIN [8], z jakých prvků se skládá, jak tyto prvky mezi sebou komunikují a popis uživatelského rozhraní. Tento popis je z teoretického pohledu. Popis funkci, které slouží pro komunikaci a nastavování protokolu jsou popsány v kapitole 5.1.

S příchodem skenerů, digitálních kamer a dalších zařízení zachycující obraz, uživatelé objevovali možnosti začlenění obrázků do jejich dokumentů a dalších prací. Zpočátku bylo získávání obrazových dat finančně náročné a každé zařízení pro získávání mělo své specifické rozhraní. Tato skutečnost prakticky znemožňovala tvorbu aplikací využívající různá zařízení tohoto typu, a proto bylo nutné vytvořit uživatelské rozhraní a ovladač pro širokou škálu zařízení. Proto vývojáři zařízení pro snímání obrazu a softwarových aplikaci připustili, že je nutné komunikaci mezi těmito zařízeními standardizovat. Standard měl umožňovat přístup k datům z těchto zařízení, přičemž nemá záležet na konkrétním typu zařízení a vytvořit jednoduché uživatelské rozhraní. TWAIN (Technology Without An Interesting Name – Technologie bez zajímavého názvu) byl vyvinut z důvodu této konzistence a zjednodušení.

TWAIN definuje standard protokolu a API (aplikační programové rozhraní) pro komunikaci mezi aplikací a zařízením pro získávání obrazu.

### 2.2.1 Tvůrci a vývoj

TWAIN vytvořila skupina softwarových a hardwarových výrobců jako reakce na potřeby společností zabývajících se zpracováním obrazových dat. Cílem bylo vytvořit otevřené, multiplatformní řešení spojení vstupních zařízení pro získání obrázku a aplikací. Původní pracovní skupina se skládala z pěti společností: Aldus, Caere, Eastman Kodak, Hewlett-Packard a Logitech. Společnosti: Adobe, Hostem a Software Architects také významně přispěly k vývoji TWAIN.

Realizace TWAIN začala v lednu 1991. V roce 1998 byla oznámena dohoda mezi společností Microsoft a TWAIN pracovní skupinou o zařazení TWAIN podpory do Microsoft Windows 98 a Microsoft Windows NT 5.0. Aktuální verze TWAIN byla vytvořena členy pracovní skupiny ve složení: Adobe, Eastman Kodak, Fujitsu Computer Products of America, Hewlett-Packard Company, JFL Peripheral Solutions Inc., Ricoh Corporation, Xerox Corporation a Lizardtech Corporation.

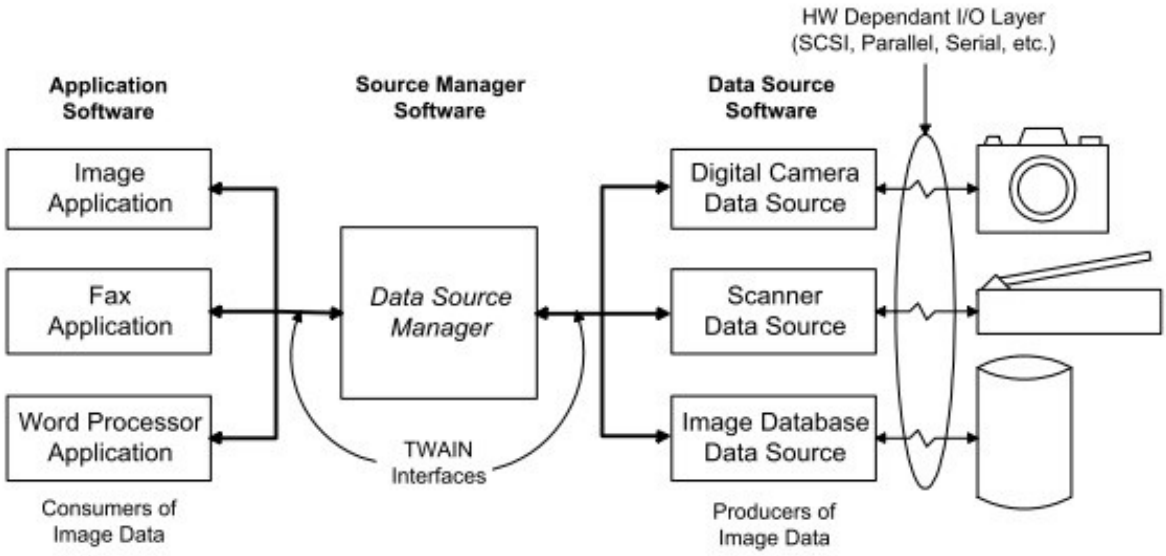
Při vývoji TWAIN se vývojová skupina držela následujících zásad:

- **Snadné přijetí:** Základem budoucího rozhraní by měla být implementace vyžadující minimální změnu v aplikaci.
- **Rozšiřitelnost:** Jedná se zejména o požadavek na multiplatformnost např. MAC OS, Microsoft Windows, Linux a různé klony UNIX. Dále také usnadnit výměnu dat mezi zařízením a aplikací. Dnes (verze 2.1) umožňuje přenášet pouze rastové typy, ale v budoucích verzích se počítá s přidáním dalších typů, jako text, vektorová grafika atp.

- **Integrace:** Jedná se o přímé začlenění do operačního systému. Klíčové prvky TWAIN jsou přímo jeho součástí.
- **Jednoduchá aplikace ↔ propojení zdrojů:** Přímočará identifikace zdroje a metoda výběru zdroje je přímo součástí TWAIN. Aplikace řídící tento mechanismus rovněž nastavuje data řídící vztahy mezi aplikací a zdrojem dat a umožňuje nastavení komunikace.
- **Zapouzdřené uživatelské rozhraní:** Nativní uživatelské rozhraní je požadováno v každém zdroji dat. Aplikace může přepsat uživatelské rozhraní.

### 2.2.2 Prvky

V této části si popíšeme prvky TWAIN protokolu, které jsou nutné pro správné fungování, k čemu tyto prvky slouží a kde jsou umístěny.




obr. 2.2.1 – Prvky TWAIN (převzato z [8])

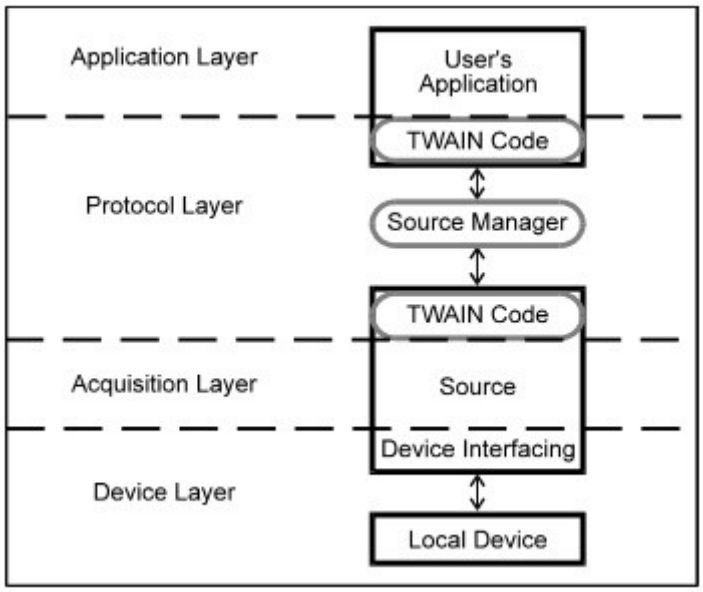
**Application Software (aplikace):** Nachází se přímo v aplikaci, která má využívat zařízení pro získávaní obrazu. Aplikace musí být modifikována, tak aby využívala protokolu TWAIN.

**Source Manager Software (TWAIN rozhraní):** Tento prvek je mezičlánkem mezi aplikací a zařízením pro získávaní obrazu. Přijímá požadavky od aplikace, ty transformuje na dotaz pro software aktuálně vybraného zařízení a dotaz mu pošle. Následně čeká na odpověď tu, pak zpět odešle aplikaci v daném formátu, na které se předem dohodli. Tento prvek je součástí operačního systému v podobě knihovny.

**Data Source Software (software zařízení):** Tato část je uložená přímo v zařízení pro snímání obrazu. Software musí splňovat standard TWAIN. Zpracovává příkazy z rozhraní TWAIN a převádí je na elementární úkoly pro zařízení. Tento prvek obsahuje nativní uživatelské rozhraní specifické pro dané zařízení.

### 2.2.3 Architektura

Architektura protokolu TWAIN se skládá ze čtyř vrstev, které využívají prvky TWAIN popsané v předešlé části.




obr. 2.2.2 – Architektura TWAIN (převzato z [8])

**Application Layer (aplikační úroveň):** Na této úrovni je uživatelská aplikace. TWAIN popisuje zásady uživatelského rozhraní pro vývojáře, přístup k funkcím a jak určitý zdroj vybrat. TWAIN neřeší jak je aplikace implementována a nemá vliv na vnitřní komunikační schéma aplikace.

**Protocol Layer (protokolová úroveň):** Tento protokol je „jazyk“, kterým TWAIN mluví a rozumí mu. Používá instrukce a komunikační vlastnosti pro přenos dat. Vrstva protokolu obsahuje:

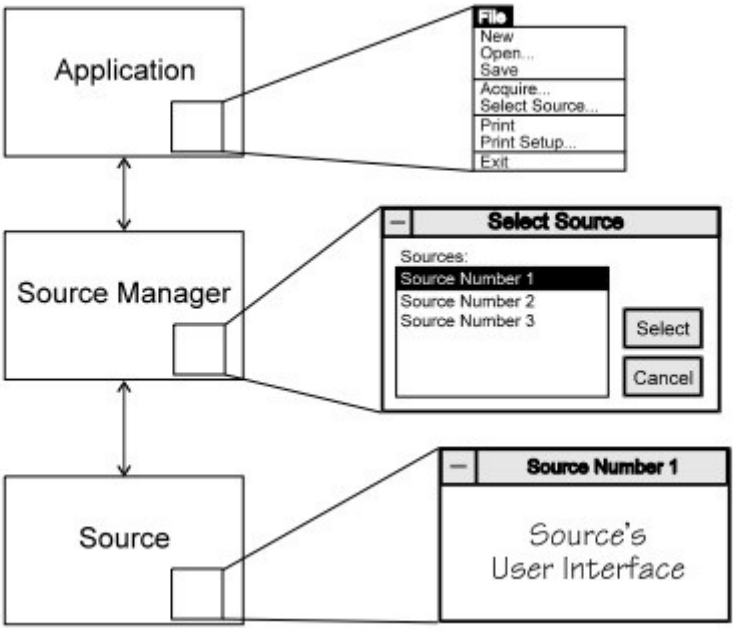
- Aplikační software poskytující rozhraní mezi aplikací a Source manager
- Source manager (vysvětlený v předešlé části)
- Software spojený se zařízením, který přijímá instrukce ze Source Manager a vrací data, nebo chybu a návratovou hodnotu.

**Acquisition Layer (Úroveň získávání):** Získávací zařízení může být fyzické (skener, digitální kamera…) nebo logické (databáze). Softwarové prvky pro získání dat se nazývají zdroje a vyskytují se zejména na této vrstvě. K předávání dat mezi zdrojem a aplikací, se používá formát a mechanismus, na kterém se aplikace a zdroj předem dohodli.

**Device Layer (Úroveň zařízení):** Jedná se o umístění nízko-úrovňových ovladačů zařízení. Na této vrstvě se konvertují specifické instrukce zařízení na hardwarové příkazy a akce specifické pro dané zařízení. Aplikace, které využívají TWAIN, již nevyžadují přikládání ovladačů zařízení k aplikaci, protože již jsou součástí zdroje (softwarový prvek pro získání dat). TWAIN se vůbec nezabývá vrstvou zařízení. Zdroje oddělují vrstvu zařízení od aplikace. Zdroje poskytují překlad z operací TWAINu a interakcí s uživatelským rozhraním, do příslušných příkazů pro ovladač zařízení, které zajišťují požadované chování.

### 2.2.4 Uživatelské rozhraní

Vyžaduje určité zásady, které je nutno dodržet při vývoji aplikace využívající TWAIN. Uživatelské rozhraní poskytované standardem TWAIN je možno přepsat a vytvořit své vlastní. Zde je schéma a popis integrovaného uživatelského rozhraní.




obr. 2.2.3 – Uživatelské rozhraní TWAIN (převzato z [8])

**Application (aplikace):** Uživatel si musí zvolit, ze kterého zařízení chce získat data. Rovněž potřebuje řídit, kdy se mají data získávat. K dosažení tohoto TWAIN doporučuje přidání dvou položek do menu výběr zdroje (Select Source) a získáváni dat (Acquire).

**Source Manager (TWAIN rozhraní):** Jestliže je vybrána položka výběr zdroje, zobrazí dialogové okno, v němž jsou zobrazeny všechny dostupné zdroje. Je nutno z nich jeden vybrat.

**Source (software zařízení):** Zdroj zobrazí integrované uživatelské rozhraní svého zařízení. Toto rozhraní může být přepsáno na aplikační úrovni. Každé zařízení má své specifické rozhraní, ale musí dodržovat standard.

V této kapitole byl popsán protokol TWAIN z několika pohledů. Je vysvětleno co je potřeba k plné funkčnosti protokolu a jak jednotlivé části protokolu mezi sebou komunikují. V poslední podkapitole je představeno integrované uživatelské rozhraní a jeho funkce na vrstvách TWAIN.

## 2.3 Jazyk C# a .NET Framework

Jazyk C# [9,10,11] jsem si zvolil pro implementaci z více důvodů. Zejména protože se jedná o objektově orientovaný jazyk, který umožňuje vhodně strukturovat aplikaci. Různé části této aplikace jsou pak znovupoužitelné. Další výhodou je velké množství knihoven pro práci s nejrůznějšími daty. Tyto knihovny jsou buď přímo jako součást .NET Frameworku [11,12], nebo mohou být vyvíjená nezávislými vývojáři. Poskytují se ve formě dynamických knihoven (dll). Jazyk C# je kompilován do podoby byte kódu (bytecode) a ten běží na platformě .NET Frameworku.

V dalších kapitolách se bude objevovat popis různých vnitřních konstrukcí aplikace a následující popis slouží pro nastínění možností jazyka C#.

### 2.3.1 .NET Framework

.NET Framework [11,12] je zastřešující název pro soubor technologií v softwarových produktech, které tvoří celou platformu. Tato platforma je dostupná pro Web, PC a mobilní zařízení. .NET Framework je prostředí, které umožňuje běh aplikací, nabízí spouštěcí rozhraní a potřebné knihovny.

Platforma je dostupná v různých podobách. Pro osobní počítače s operačním systémem Microsoft Windows od verze 98 je to Microsoft .NET Framework, pro kapesní počítače a mobilní telefony s operačním systémem Windows Mobile je to Microsoft .NET Compact Framework. Platforma určená pro embedded zařízení s ještě menší výpočetní kapacitou a většími omezeními je Microsoft .NET Micro Framework. Dále produkt Mono, který je určený pro operační systémy unixového typu, tento produkt je open source. Funkčnost aplikace na těchto systémech není zaručená, neboť Mono je vyvíjen nezávislými vývojáři, ne společností Microsoft.

Pro vývoj .NET aplikací vydal Microsoft Visual Studio .NET. Visual Studio obsahuje editor kódu, integrovaný debugger a mnoho dalších nástrojů jako designer formulářů, designer webů, tříd a databázových schémat. Je možno přidávat další rozšíření na téměř každé úrovni.

Platforma nepředepisuje použití jednoho jazyka. Zdrojový kód je vždy přeložen do mezijazyka CLI (Common Intermediate Language) a ten je následně spuštěn. K dispozici je celá řada programovacích jazyku .NET aplikací, např. C#, Visual Basic .NET, F#, J#, C++/CLI, JScript .NET a mnoho dalších.

### 2.3.2 Programovací jazyk C#

Jedná se o vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý společností Microsoft. Je standardizován komisemi ECMA (ECMA-334) a ISO (ISO/IEC 23270) [9]. Jazyk je založen na jazycích C++ a Java. C# zahrnuje následující programovací paradigmata: silné typování, je imperativní, deklarativní, funkcionální, objektově-orientovaný a komponentově orientovaný.

Velkou výhodou jazyka je rozdělení aplikace do několika projektů. Toto je vhodné využít pro oddělení formulářů a datové vrstvy. Formuláře tvoří samotnou aplikaci (exe) a jednotlivé projekty datové vrstvy jsou kompilovány do podoby dynamických knihoven. Je zde ovšem možnost různých nastavení kompilace výsledné aplikace. Dynamické knihovny jsou vhodné zejména pro aktualizace, není nutno po opravě chyb kompilovat celou aplikace a tu distribuovat, ale stačí zkompilovat knihovnu a tu následně distribuovat.

Definice jazyka dle standardu ECMA a několik vlastností jazyka.

**Standard ECMA definuje C# jako**:

- Jednoduchý, moderní, mnohoúčelový a objektově orientovaný programovací jazyk.
- Jazyk a jeho implementace poskytuje podporu pro principy softwarového inženýrství, jako jsou: detekce použití neinicializovaných proměnných, hlídaní hranic polí, automatický garbage collector a vlastnosti: robustnost, trvanlivost a programátorská produktivita.
- Jazyk je vhodný pro vývoj softwarových komponent distribuovaných v různých prostředích.
- Přenositelnost kódu je důležitá.
- Důležitá je i mezinárodní podpora.
- C# je navržen pro psaní aplikací, jak pro zařízení se sofistikovaným operačním systémem, tak pro zařízení s omezenými možnostmi.
- Přestože by aplikace psané v C# neměly plýtvat procesorovým časem a pamětí, nemohou se rovnat s aplikacemi napsanými v jazyce C nebo jazyce symbolických adres.

**Vlastnosti jazyka:**

- V C# neexistuje vícenásobná dědičnost, to znamená, že každá třída může mít právě jednoho přímého předka. Třída může implementovat libovolný počet rozhraní.
- Neexistují žádné globální proměnné, všechny funkce, metody musí být deklarovány uvnitř tříd. Náhradou za ně jsou statické metody a proměnné veřejných (public) tříd.
- C# je typově bezpečnější než C++, jediná výchozí implicitní konverze je rozšiřování typu integer (např. z 32 bitového na 64 bitový), nebo z odvozeného odvozeného typu na rodičovský. Neexistuje ani implicitní konverze z typu integer na boolean, ani mezi výčtovým typem a integer.
- Jazyk C# je case sensitive.

## 2.4 Jazyk SQL

Univerzální databázový dotazovací jazyk SQL [13,14] (Structured Query Language), slouží k formulaci instrukcí pro databázový systém. Jsou to dotazy pro přidávání, změnu, mazání, čtení databázových objektů a mnoho dalších úkonů. Většina databázových systémů používá upravený jazyk SQL, MySQL používá standard SQL/PSM (SQL/Persistent Stored Module). Úprava jazyka SQL spočívá většinou ve speciálních příkazech.

Tento jazyk je v aplikaci využit pro práci s databází. Tato komunikace je jedna z nejdůležitějších částí aplikace.

### 2.4.1 Historie

V roce 1975 byl vytvořen jazyk SEQUEL (Structured English Query Language) a použit v Systém R. Jazyk se postupně začal rozšiřovat a začal se používat i v dalších databázových systémech. Proto bylo nutno jazyk standardizovat. V roce 1986 by standardizován ANSI a ISO. Jazyk byl přejmenován na SQL. Další vývoj jazyka byl v rozšiřování možností. Objevují se objektově-relační rysy (1999), podpora multimédií (2002), OLAP (2003) a XML dat (postupně 2003-2008). Jazyk se neustále vyvíjí a jeho použitelnost je dále rozšiřována.

### 2.4.2 Dotazy SQL

Podle účelu lze příkazy rozdělit do tří resp. čtyř hlavních skupin:

- **Pro práci s daty (Data Manipulation Language)**: SELECT, INSERT, UPDATE, DELETE a několik dalších. Tyto příkazy umožňují čtení, vkládání, změnu a smazání záznamu v tabulce.
- **Pro definici dat (Data Definition Language)**: CREATE TABLE, ALTER TABLE a podobné. Těmito příkazy můžeme vytvářet a měnit návrh databáze.
- **Pro řízení dat (Data Control Language)**: GRANT, REVOKE, START TRANSACTION, COMMIT, ROLLBACK a několik dalších. Těmito příkazy nastavujeme bezpečnostní mechanizmy MySQL a umožňují řízení transakcí.
- **Speciální příkazy**: do této skupiny spadají speciální příkazy, jako je nastaveni jazykového prostředí databáze, formátu data a času, přidání uživatele a mnoho dalších možností. Tato skupina není nijak standardizovaná a liší se podle databázového systému.

Příkazy jazyka SQL umožňují úplnou obecně kontrolu nad celým databázovým systémem. Je snaha, aby příkazy SQL byly syntakticky co nejblíže přirozenému jazyku (angličtině).

## 2.5 Databáze MySQL

Tato kapitola obsahuje popis databázového systému MySQL. [13,15] Databázi MySQL jsem si zvolil, protože se jedná o open source projekt a je po splnění určitých podmínek dostupná zdarma. Aplikace nevyžaduje příliš složitou databázi, a proto jsem zvolil tento jednoduchý databázový systém.

### 2.5.1 Historický vývoj

Databázový systém MySQL je vytvořen švédskou firmou MySQL AB, kterou nyní vlastní společnost Oracle Corporation. První verze MySQL 1.0 vyšla v roce 1995, jednalo se o verzi určenou pro operační systémy založené na Unixu. V roce 1998 byla vydána verze pro Microsoft Windows, dnes je databáze dostupná pro celou řadu operačních systémů.

Tento databázový systém byl od počátku optimalizován pro rychlost i za cenu některých zjednodušení. Postupem času však byli vývojáři nuceni na požadavek uživatelů postupně doplňovat funkčnost. Aktuální verze MySQL 5.5 je stále rychlá a obsahuje již mnoha rozšíření, avšak mnohé jí ještě chybí např. podpora XML.

### 2.5.2 Popis vlastností a licencí

MySQL je relační databázový systém. Relační databáze je založena na tabulkách. Nejtypičtějším rysem relačních databází je možnost vzájemného provázání tabulek v databázi podle jejich obsahu. Pro vazby mezi tabulkami se používá pojem relace. Databázový systém MySQL je díky své jednoduché implementaci multiplatformní. MySQL může být provozováno na MAC OS X, Microsoft Windows, Linux a mnoho dalších unixových klonech, jako FreeBSD, OpenBSD, Sun Solaris…

Architektura MySQL je založena komunikaci klient/server, kde klient zasílá serveru požadavky, ten je zpracuje a vrátí klientovi odpověď. Ke komunikaci mezi klientem a serverem se využívá jazyka SQL.

Existuje celá řada API a knihoven určených pro vývoj aplikací MySQL. Pro programování klientů můžeme použít C, C++, C#, Java, PHP a mnoho dalších.

MySQL je poskytováno pod dvojí licencí. MySQL je open source projekt s licencí GNU/GPL, pokud nechceme nebo nemůžeme u své aplikace splnit podmínky této licence, musíme použít druhou licenci. Komerční licence pro MySQL je poskytována ve více verzích s různou podporou. Obě tyto licence se vztahují jak na MySQL server, tak na API a knihovny pro tvorbu MySQL aplikací.

## 2.6 Technologie sdílení souborů

Data určená pro archivaci jsou v podobě obrázků získaných pomocí protokolu TWAIN. Tyto obrázky nejsou ukládaný přímo do databáze, ale na disk do hierarchie složek. Následně je použita technologie pro sdílení, aby mohla být tato hierarchie složek dostupná z více míst. Pokud bude aplikace využívaná pouze offline, není nutná tato technologie. Dnes je dostupná velká spousta takovýchto technologií. Liší se ve velké spoustě různých parametrů. Jeden z hlavních rozdílů je ve formě poskytování, buď je nabídnuta přímo celá služba i s úložným prostorem, nebo nástroj pro vytvoření této služby.

Aplikace má specifické požadavky na vlastnosti dané technologie. Nejdůležitější požadavek je, aby technologie umožňovala sdílet určitou složku. Další požadavky na šifrování dat během přenosu a na serveru jsou již na uživateli, jakou vyžaduje bezpečnost. Jednou z nejjednodušších možností jak sdílet tato data je připojení vzdáleného disku (mount). Zde je ovšem problém při nedostupnosti připojení k vzdálenému disku, není možná práce offline. Následuje tabulka (tab. 2.6.1) dnes dostupných služeb, které splňují podmínku pro funkčnost aplikace online, těchto služeb v poslední době přibývá. Srovnání slouží jako přehled služeb, nenutí tím uživatele zvolit některou z těchto služeb.

| služba | nezávislost poskytovatele | open source | velikost úložného prostoru (zdarma/max. placena) [GB] | operační systém (PC) |
|---|---|---|---|---|
| Dropbox | ne | ne | 2/100 | Windows, Linux, Mac |
| Sparke Share | ano | ano | vlastní | Windows, Linux, Mac |
| Syncany | ano | ano | vlastní | Windows, Linux, Mac |
| Ubuntu One | ne | ne(server) ano(klient) | 5/50 | Windows, Linux |
| openDrive | ne | ne | 5/1000 | Windows, Linux, Mac |
| SugarSync | ne | ne | 5/100 | Windows, Linux, Mac |

tab. 2.6.1 Srovnání technologií [16,17]

Srovnání je pouze v několika parametrech. Tyto služby mají velkou spoustu parametrů, ale ne všechny jsou důležité pro použití spolu s touto aplikací. Jeden z parametrů, který je poměrně důležitý pro aplikaci je počet uživatelů. U většiny aplikace je jeden uživatel, avšak je možný přístup z více zařízení, proto je služba s těmito parametry použitelná.

Podporované operační systémy uvádím z důvodu možné jejich budoucí podpory. Všechny tyto služby jsou podle uvedených parametrů použitelné pro použití s aplikací.

# 3. Analýza požadavků

Aplikace má sloužit k digitalizaci papírových databází (kartoték) nebo vytváření nových databází. Dále má umožňovat tyto digitální databáze převádět (tisknout) do papírové podoby. Aplikace by měla provozovat v režimu, jak online, tak offline, pokud aplikace pracuje v režimu offline je možnost přidávat nová data, která budou po přihlášení online nahrána do vzdálené databáze. Při práci online probíhá v určitých časových intervalech synchronizace do lokální databáze, pokud je dostupná. Ke sdílení dat (obrázků) je nutno pro práci online využít aplikaci třetí strany, možno vybrat z možností uvedených v tabulce tab. 2.6.1.

Aplikace by v ideálním případě měla být použitelná pro jakoukoli archivaci, v praxi ovšem dosažení tohoto cíle není možné, anebo by museli uživatele omezit možnosti na tuto aplikaci/databázi.

## 3.1 Struktura ukládaných dat

Při analýze požadavků na ukládaná data jsem došel k závěru, že ukládání pouze obrázků je velice nepraktické. Z toho důvodu jsem navrhl strukturu, která umožňuje obrazová data logicky řadit v databázi a shlukovat je do skupin. Tyto skupiny bude možno doplnit popisem pro lepší orientaci v archívu.

Databáze bude obsahovat záznamy, kde budou obsaženy informace o logickém celku dat. Pro ještě specifičtější řazení obrazových dat bude možno vytvářet podsložky (souhrny). Poslední částí jsou samotná obrazová data, k nim bude možno přidat textový popis. Tato data mohou být přiřazeny přímo k záznamu, nebo do podsložek.

## 3.2 Model případů užití

Diagram případů užití popisuje možnosti, které mohou provádět jednotliví účastníci. Diagram je zobrazen na obrázku 3.2.1. Následně jsou jednotlivé případy popsány.


obr. 3.2.1 Diagram případů užití

### 3.2.1 Popis případu užití pro účastníka – uživatel

Pro zpřístupnění funkcí uživatele je nutno, aby se uživatel přihlásil. K tomu je potřeba přihlašovací jméno, heslo, dále je nutno znát jméno databáze a adresu databázového serveru. Po úspěšném přihlášení již má uživatel k dispozici všechny mu určené funkce.

**Přidání, smazání a editování** všech položek ukládaných do databáze. Obsah jednotlivých položek a jejich propojení je popsán v kapitole 4.1 Návrh databáze. Další funkcí je možnost **výběru zařízení,** ze kterého se budou získávat obrazová data. Pokud uživatel nevybere zařízení, je použito výchozí zařízení nastavené v systému. Uživatel rovněž může **prohlížet položky** uložené v databázi. K prohlížení je možno využít více způsobů, zejména klávesnice a myši.

Případ užití **tisk a náhled tisku** zpřístupňuje uživateli možnost převodu digitální databáze do papírové podoby, tisknout určité dokumenty atp. Je možno tisknout celé záznamy se všema podsložkama a daty v nich obsažené, nebo jednotlivá data (dokumenty). Režimy tisku jsou obdobné jako v jiných aplikacích. Tedy rychlý tisk bez možnosti nastavení, nebo s možností nastavení, kde je možno vybrat tiskárnu, kvalitu atd. Při volbě zobrazení náhledu je zobrazeno okno s náhledem tisku.

Uživatel má také možnost nastavit určité parametry aplikace. **Nastavení komprese** a **umístění dat** umožňuje uživateli nastavit parametry získávaných dat. Jedná se o velikost komprese získaných dat v procentech (výchozí nastavení na 50%) a nastavení umístění hierarchie složek obrazových dat z databáze (výchozí nastavení do C:/archiv).

### 3.2.2 Popis případu užití pro účastníka – administrátor

Pro zpřístupnění funkcí administrátora je nutno, aby se administrátor přihlásil. K tomu je potřeba přihlašovací jméno, heslo, dále je nutno znát adresu databázového serveru. Po úspěšném přihlášení již má administrátor k dispozici všechny mu určené funkce.

Možnosti administrátora lze rozdělit do dvou sekcí, správa uživatele a databáze. Uživatele může administrátor jak **vytvořit**, tak **smazat**. Existujícímu uživateli je možno **přidávat oprávnění** pro jednotlivé databáze a tím mu umožnit do této databáze přistupovat a pracovat s ní. Uživatelé, kteří byli vloženi na server z jiného umístění, se mohou přihlásit lokálně až po té co administrátor provede **synchronizaci uživatelů na lokální server**. Tento krok vyžaduje administrátorské oprávnění, a proto ho musí provádět administrátor a není možno jej provádět automaticky.

Další možnosti administrátora jsou spojeny se správou databáze. Administrátorovi je umožněno **přidávat a mazat databáze**. Další možností je **vložení databáze na lokální server**. Databáze na vzdáleném serveru, která není vložena lokálně, musí být lokálně vložená administrátorem. Tato operace totiž vyžaduje administrátorské oprávnění. Administrátor také může **synchronizovat libovolnou databázi** ze vzdáleného serveru na lokální. Synchronizována databáze musí být na lokálním serveru vytvořena. Na rozdíl od uživatele může synchronizovat libovolnou databázi, uživatel může synchronizovat pouze databázi, ke které je přihlášen.

### 3.2.3 Popis případu užití pro účastníka – čas

Čas je speciální případ účastníka. Jemu jsou přiřazeny úlohy, které přímo nemůže uživatel ovládat. Oba tyto případy užití jsou spouštěny na jiném vlákně než samotná aplikace. Uživatel tak může normálně pracovat. **Nahrání dat na server z cache paměti** se provede vždy po přihlášení uživatele ke vzdálenému serveru. Data, která se nepovede nahrát, budou zpět uložena do cache.

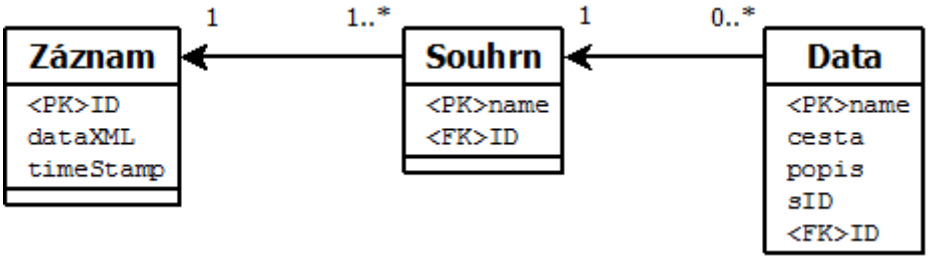
Druhou funkcí je **automatická synchronizace** databáze. Synchronizace probíhá pouze, jestliže je dostupná lokální databáze. Poprvé proběhne ihned po přihlášení uživatele, následně pak periodicky v nastaveném časovém intervalu (ve výchozím stavu nastaveno na 2 hodiny).

# 4. Návrh řešení

Na základě předchozí analýzy požadavků jsem navrhl aplikaci. Návrh obsahuje model databáze, uživatelské rozhraní a vnitřní strukturu aplikace.

## 4.1 Návrh databáze

Návrh datového modelu databáze reprezentuje ER diagram. Databáze je navržená tak, aby bylo možno ukládat data různého charakteru. Různorodost dat umožňuje ukládání informací ve formátu XML, MySQL nemá podporu XML formátu, tyto data jsou v tabulce ukládaná ve formě textu a na úrovni aplikace konvertována do formátu XML a dále zpracovávaná. Omezení primárních klíčů je řešeno na úrovni databáze a je kontrolováno na úrovni aplikace. Dále jsou popsány atributy XML souboru, které určují typy dat a další možnosti, všechny atributy jsou u všech elementů.




obr. 4.1.1 ER diagram

**Atributy XML souboru**

primary – atribut označuje primární klíč. Hodnota atributu pro primární klič je „true“, všechny ostatní položky musí mít hodnotu „false“.

name – označuje položku, která bude tvořit název ve stromovém seznamu položek. Položka označená jako primární klíč je součástí názvu automaticky, je přiřazena nakonec. Hodnota „true“ znamená použití v názvu, „false“ nikoli.

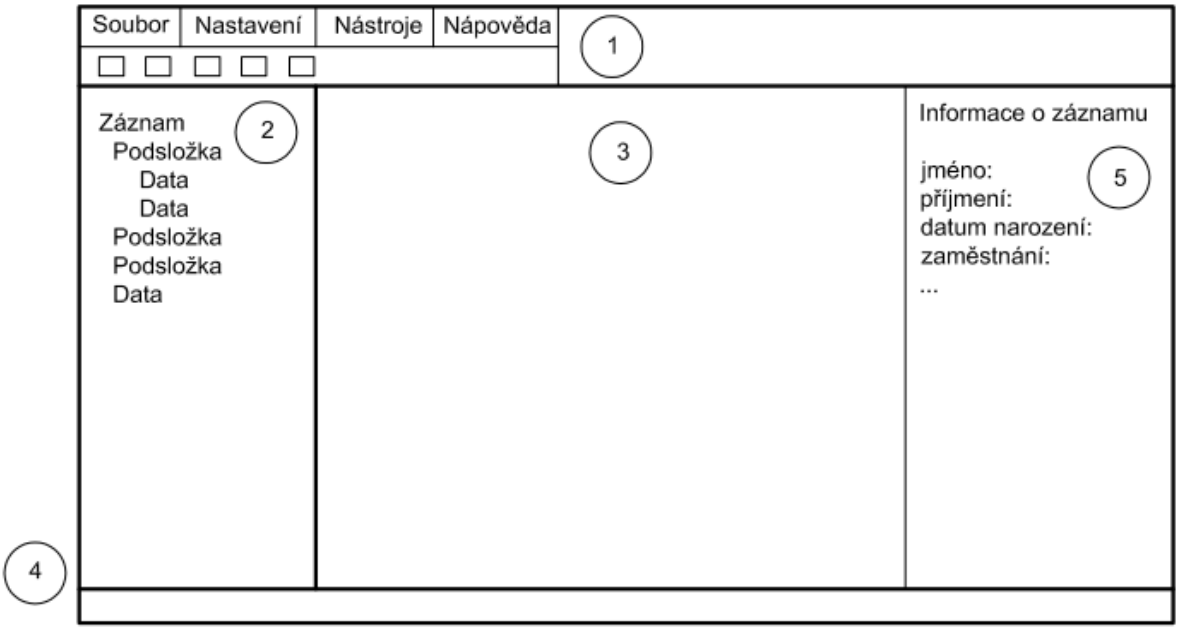
type – jedná se o atribut označující typ dat. Aplikace podporuje několik typů. Pokud není vyžadována typová kontrola, je použito klíčové slovo „undefined“. Primární klíč tabulky musí obsahovat celočíselný typ.

**Podporované typy:**

- **integer** – celočíselný datový typ
- **numeric** – typ obsahující číselný typ, nejsou podporovaná komplexní čísla
- **date** – typ obsahující datum
- **email** – typ obsahující emailovou adresu, kontroluje se pouze správnost zadání, nikoli existence

## 4.2 Návrh uživatelského rozhraní

Uživatelské rozhraní je z pohledu uživatele nejdůležitější část aplikace. Je to část, se kterou pracuje. Je navrženo co nejjednodušší a intuitivní. Rozhraní zpřístupňuje uživateli všechny případy užití specifikované v kapitole 3.1.




obr. 4.2.1. – Uživatelské rozhraní

Dále je v bodech popis oblastí uživatelského rozhraní. Je popsán význam a obsah těchto částí. Čísla bodů odpovídají číslům uvedeným v obrázku 4.2.1.

1) Kompletní menu pro práci s aplikací, poskytující přístup ke všem případům užití. V horní části hlavní menu obsahující všechny možnosti spojené s nastavením aplikace, pod ním je rychlé menu obsahující položky pro práci s aplikací a některé další často používané prvky.
2) Zobrazení seznamu záznamů obsažených v aplikaci. V rychlém menu v horní části jsou možnosti pro prácí se záznamy, souhrny a daty v databázi. Nebo je zde zobrazeno textové pole pro vyhledávání v záznamech s možností určení klíče pro vyhledávání. Níže je zobrazena stromová struktura položek obsažených v databázi.
3) Zobrazovací oblast dat (obrázků), ve druhé záložce zobrazení popisu dat s možností editace.
4) Oznamovací oblast. Jsou zde zobrazeny informace o aktuální relaci. Jako název databáze, režim přihlášení uživatele, Adresa vzdálené databáze a umístění obrázku na disku.
5) Zde se nachází informace o aktuálně vybraném záznamu v podobě tabulky. V dolní části je možnost povolení editace.

## 4.3 Vnitřní návrh aplikace

Před samotnou implementací je důležitá dekompozice problém. Jelikož je C# objektově orientovaný toto rozdělení je v podobě třídního návrhu. Jazyk C# umožňuje projekt rozdělit na více projektů. Využitím této možnosti jsem projekt rozdělil na uživatelské rozhraní, které obsahuje všechny zobrazované formuláře a projekty datové vrstvy. Každý z těchto projektů se zabývá řešením jiného problému. Celkem projekt obsahuje pět datových projektů. Projekt obsahující formuláře je jednoduchý a v dalším textu nebude podrobněji popsán. Podrobnější popis některých tříd se nachází kapitole 5.1. Všechny datové projekty jsou kompilovány do podoby dynamických knihoven (dll). Toto je výhodná možnost při opravě chyb, nemusí se aktualizovat celá aplikace, ale stačí pouze nakopírovat novou verzi knihovny. Tyto knihovny je také možná využívat i v jiných projektech.

Zde je stručný popis jednotlivých datových projektů.

### confDatabase

Tento projekt bude sloužit k získání informací o databázích. Jeho obsahem bude formát XML dat pro vytváření nových záznamů v databázi a seznam položek pro vyhledávání. Pokud bude potřeba přidat nový typ databáze, bude potřeba editovat tento projekt. Výslednou upravenou dll knihovnu následně distribuovat uživatelům.

### dataSource

Tento projekt bude obsahovat dvě třídy. Jednu pro práci s XML daty a druhou pro zobrazení stromové struktury dat uložených v databázi. Tyto zobrazována data mohou být také filtrována po zadání parametrů.

### mySqlLib

Tento projekt bude pokrývat velkou část aplikace. Bude poskytovat kompletní možnosti pro přístup k databázi na uživatelské i administrátorké úrovni. Také možnost šifrování dat ukládaných do cache paměti, možnosti nahrání dat cache paměti do vzdálené databáze a nástroje pro synchronizaci databáze v určitých časových intervalech.

### twain

Zde bude kompletní obsluha zařízení pro získávání obrazových dat a možnost převodu obrázku do daného formátu včetně možnosti komprese.

### printer

Projekt *printer* jak již název napovídá, bude umožňovat tisk dat. Podpora bude pro standardní funkce pro tisk a náhled. Dále možnost vytvářet obrázky z textu a datových tabulek (dataGrindView). Další možností bude možnost změny velikosti obrázku pro tisknutelnou plochu.

# 5. Implementace

Obsahem této kapitoly je popis implementace zajímavých částí aplikace. Vývoj aplikace probíhal ve vývojovém prostředí Microsoft Visual Studio 2010. Jako nápověda při vývoji aplikace sloužily webové stránky[10] věnované standardním knihovnám C# a obsahující mnoho dalších užitečných informací o programování v jazyce C#.

Při zapnutí aplikace je nutno se přihlásit a to buď jako uživatel, nebo administrátor. Po úspěšném přihlášení jako uživatel jsou dostupné všechny funkce přístupné uživateli. Ihned po přihlášení je zobrazen seznam záznamů v levé části aplikace a ve stavové liště jsou zobrazeny informace o aktuálním připojení. Od této chvíle má uživatel možnost prohlížet, přidávat, mazat data a obecně všechny možnosti vyplývající z diagramu případů užití zobrazenému na obrázku 3.2.1. Po přihlášení jako administrátor je zobrazeno dialogové okno se všemi možnostmi administrátora, specifikované v diagramu případů užití. Aby mohl administrátor pracovat jako běžný uživatel, musí toto okno uzavřít a přihlásit se jako běžný uživatel.

Následující podkapitoly jsou věnovány popisu implementace, a jak tyto části fungují. Dále jsou popsány možnosti automatické konfigurace a na závěr postup instalace aplikace.

## 5.1 Použití protokolu TWAIN

Celá práce s protokolem TWAIN je součástí projektu „twain“, který je kompilován do dll knihovny a je možno jej využít i dalšími aplikacemi. Základ tohoto projektu tvoří převzatý kód ze stránek The code project od autora NETMaster, tento kód je zveřejněn pod licenci „A Public Domain dedication“, která umožňuje použití a úpravu tohoto kódu, použitím této licence se autor vzdává autorských práv k dílu. [18]

Tento kód jsem upravil pro potřeby aplikace. Celou práci spojenou se získáváním obrazových dat jsem rozdělil do tří tříd. TwainLib.cs a TwainDefs.cs jsou převzaté a pictureTransver.cs je mnou vytvořená. TwainDefs.cs obsahuje definice výčtových typů (enum) určujících možnosti nastavení protokolu a struktur obsahující aktuální nastavení komunikace.

Pro nastavení a práci protokolu TWAIN slouží třída TwainLib.cs. Po vytvoření instance třídy je nutno tuto instanci inicializovat pomocí metody Init() s parametrem Handle, který identifikuje aplikaci. Po inicializaci je nutno vybrat zařízení které bude sloužit pro získávání obrazových dat, jinak je použito výchozí nastavení. Následně již může dojít k samotnému získávání. Po dokončení získávání je nutno převést obrázky a ukončit získáváni. Převod obrázků probíhá ve dvou fázích, nejprve jsou převedeny do pole obsahující odkazy na obrázky a pak pomocí třídy pictureTransver je potřeba převést na klasické obrázky v podobě jpeg.

Funkce pro ovládání zařízení pomocí protokolu TWAIN jsou z knihovny *twain_32.dll*. Tato knihovna není přímo určena pro .NET Framework, proto je nutno na tyto funkce vytvořit odkazy do knihovny. To je umožněno pomocí funkce DllImport. Tímto způsobem se dostaneme k funkcím obsaženým v knihovně. Tento postup je také použit na některé funkce z knihoven *gdi32.dll* a *kernel32.dll*.

Příklad:

```
[DllImport("twain_32.dll", EntryPoint="#1")]
        private static extern TwRC DSMparent( [In, Out] TwIdentity origin, IntPtr zeroptr, TwDG dg,
TwDAT dat, TwMSG msg, ref IntPtr refptr );
```

## Popis některých funkcí pro práci s protokolem TWAIN:

**DSMparent** – využívá se při inicializaci protokolu TWAIN, pro použití aplikací.

Příklad použití:

```
DSMparent( appid, IntPtr.Zero, TwDG.Control, TwDAT.Parent, TwMSG.OpenDSM, ref hwndp )
```

appid – identifikace protokolu TWAIN, zde bude inicializována

Další 3 parametry slouží pro nastavení funkce

TwMSG – otevření/zavření možnosti použití TWAIN. OpenDSM – Povolení možnosti použití protokolu TWAIN, CloseDSM – v metodě Finish(), ukončí použití TWAIN aplikací, nové použití je možné až po nové inicializaci.

hwndp – identifikace aplikace, v C# je při volání Init() použit jako parametr this.Handle, který je použit jako tento parametr

**DSMident** – slouží pro výběr zařízení, které bude provádět získávání. Při inicializaci je vybráno výchozí zařízení. Tento výběr je možno změnit pomocí metody Select(). Tato metoda zobrazí dialogové okno, vyvolané právě touto funkcí, ve kterém jsou zobrazena všechna zařízení podporující TWAIN.

Příklad použití:

```
DSMident( appid, IntPtr.Zero, TwDG.Control, TwDAT.Identity, TwMSG.UserSelect, srcds );
```

appid, srcds – identifikace zařízení, struktura TWAIN identity

Další 3 parametry nastavují protokol

TwMSG – určuje co se bude provádět. UserSelect – objeví se dialogové okno a uživatel vybere zařízení, GetDefault – automaticky se vybere výchozí zařízení a apod.

**DSuserif** – tato funkce zajišťuje samotný proces získávání obrazových dat. V této funkci lze pomocí parametru nastavit možnost zobrazení uživatelského rozhraní TWAIN.

Příklad použití:

```
DSuserif( appid, srcds, TwDG.Control, TwDAT.UserInterface, TwMSG.EnableDS, guif )
```

appid, srcds – identifikace zařízení na kterém bude probíhat získávání

Další 3 parametry slouží pro nastavení přenosu

guif – struktura obsahující nastavení uživatelského rozhraní, guif.ShowUI = 0 – uživatelské rozhraní se nezobrazí a ihned začne probíhat skenování podle posledního nastavení, jestliže guif.ShowUI = 1 – zobrazí se integrované uživatelské rozhraní TWAIN.

## 5.2 Komunikace se serverem MySQL

Pro připojení k MySQL serveru je využito poskytovaného konektoru (knihovny) *MySql.Data.dll* společnosti Oracle. Tato knihovna je určena pro .NET Framework a umožňuje kompletní práci s MySQL serverem resp. MySQL databází.

Komunikace se servery je obsažena v projektu *mySqlLib*. Tento projekt poskytuje kompletní vybavení pro komunikaci s MySQL serverem. Základní třída *sqlClient* uchovává informace o aktuálním přihlášeném klientovi a umožňuje provádět dotazy nad danou databázi. Další důležitou třídou je *sqlAdmin* ta umožňuje přihlášení administrátorovi a pomocí ní administrátor realizuje práci s databází. V neposlední řadě je důležitou funkcí možnost synchronizace a nahrání dat z cache paměti na server.

Při zapnutí aplikace je vytvořena instance třídy *sqlClient* ta je uchovávaná po celou dobu běhu aplikace. Pokud dojde k odhlášení a opětovnému přihlášení je objekt smazán a vytvořen nový. Po zadání přihlašovacích údajů je testována správnost připojení k databázi, nejprve je proveden pokus o připojení k vzdálené databáze, při neúspěchu je proveden pokus o připojení k lokální, pokud je neúspěšný i ten je vrácená chyba. Pokud proběhne úspěšné připojení k jedné z databází, jsou uchovány informace o přihlášení. Po úspěšném přihlášení ke vzdálené databázi je otestován přístup k lokální a v případě úspěšnosti je spuštěna synchronizace. Následně je spuštěn pokus o nahrání dat z cache paměti a otevírá se možnost pracovat s databází. Pak má uživatel možnost vládat, měnit, mazat a vyhledávat ve všech třech tabulkách aktuální databáze, tyto operace probíhají nad aktuální databází (vzdálená nebo lokální). Pokud je uživatel přihlášen offline, jsou operace vkládaní, změna a mazání ukládány do cache paměti pro pozdější nahrání dat.

Pro přihlášení a celé sezení administrátora je vytvořena instance třídy *sqlAdmin*. Tato třída je velice podobná třídě *sqlClient* v podobě uchování dat o přihlášeném uživateli. Tato třída slouží pro administraci databáze a umožňuje provádět všechny operace specifikované v diagramu případů užití zobrazenému na obrázku 3.2.1.

Dále projekt obsahuje podpůrné třídy. Třída *connect* umožňuje testování připojení a vykonávaní dotazů. Třída *cryptor* slouží pro šifrování dat, která jsou ukládaná do cache paměti nebo k ukládání konfigurace. Další tři třídy umožňují synchronizaci databáze a upload dat, jedna obsahuje metody pro tuto činnost, další dvě spouští nová vlákna pro běh na pozadí, aby uživatel mohl během těchto operací již pracovat.

## 5.3 Tisk

Aplikace umožňuje, jak je obvyklé, dva druhy tisku a náhled tisku. Třída pro tisk je obsažena v projektu „printer“ objekt umožňující tisk se jmenuje *print*. Třída obsahuje tři metody nezávislé na tisku a slouží pro přípravu před tiskem. Jedna z metod vytvoří ze zobrazených dat v tabulce obrázek, který je následně tištěn, druhá slouží pro úpravu velikosti obrázku, pokud obrázek přesahuje tisknutelnou plochu, je zmenšen na velikost této plochy. Zmenšení může být jak na šířku, tak na výšku. Poslední metoda umožňuje vytvořit z textového řetězce obrázek, který může být následně tištěn.

Všechny tři metody pro obsluhu tisku očekávají jako parametr pole obrázku určených pro tisk. Všechny tři metody jsou si velice podobné. Umožňují standardní funkce pro tisk, což jsou zobrazení náhledu, tisk s možnosti nastavení tiskárny a tzv. rychlý tisk. Tyto funkce jsou obdobné jako ve většině programů.

Další speciální metoda je vyvolána před samotnou akcí tisku (zobrazení náhledu), a ta plní tiskový zásobník stránkami určenými pro tisk a následně je vyvolána událost tisku (zobrazení náhledu).

## 5.4 Konfigurace aplikace

Při prvním spuštění a přihlášení do systému, je vytvořen konfigurační soubor pro uchování nastavení posledního přihlášení. Uložení těchto dat usnadňuje uživateli práci s aplikací. Pokud uživatel využívá jednu databázi na jednom serveru, pro přihlášení stačí zadat pouze heslo (to nelze uložit v žádném případě). V případě využití jiné databáze je nutno přihlašovací údaje změnit. Do souboru se ukládá přihlašovací jméno, adresa serveru, jméno databáze, cesta k souborům a velikost komprese dat. Data jsou ukládaná ve formátu XML, do souboru s názvem *settings.xml*, data jsou v souboru šifrována pro zvýšení bezpečnosti. Tento soubor je vytvořen po prvním přihlášení, při následujícím přihlášení je aktualizován, takže není možno si pamatovat více databází. Soubor je aktualizován vždy při změně některé z ukládaných položek. Pokud dojde ke smazání souboru, aplikace automaticky vytvoří nový při následujícím spuštění, nebo při pokusu o aktualizaci, s aktuálními daty.

## 5.5 Instalace

Samotná aplikace nevyžaduje instalaci, avšak pro úplnou funkčnost tj. práci offline je potřeba nainstalovat MySQL server. Ten je po splnění licenčních podmínek dostupný na domovských stránkách MySQL, v sekci download.[15] Aplikace je testována na verzi 5. Je potřeba mít také nainstalován vzdálený server (pokud uživatel potřebuje). Při instalaci vzdáleného serveru je potřeba povolit vzdálený přístup. Po instalaci je nutno přidat pomocí SQL skriptu administrátorovi práva pro vzdálené přidávání uživatelů, tento úkon je potřeba provést lokálně, protože vzdáleně tento skript selže pro nedostatečná práva. Pro přidání tohoto skriptu je potřeba se přihlásit jako administrátor pomocí konzole MySQL (dostupná u nainstalovaného serveru). Bez přidání těchto práv administrátor přihlášený vzdáleně, nemůže přidělovat práva ostatním uživatelům. Na administrátorský účet se vztahuje také omezení, že administrátor musí mít stejné heslo na lokálním i vzdáleném serveru. Pokud nebude tato podmínka splněna, dojde k chybě u operací vyžadující obě databáze.

SQL příkaz pro přidání práv administrátorovi:

```
GRANT ALL PRIVILEGES ON *.* TO 'login'@'%' WITH GRANT OPTION;
```

Aplikace vyžaduje k běhu .NET Framework 4. Tato součást systému je většinou již nainstalována, neboť ji potřebuje velká škála aplikací, v opačném případě, je nutno ji doinstalovat. Pro operační systémy založené na UNIX nutno doinstalovat patřičnou verzi Mono.

Pro práci se zařízením pro získávání obrazových dat, je většinou potřeba nainstalovat ovladače daného zařízení. Pro některé zařízení to není nutno a TWAIN je podporuje přímo. Bohužel takovéto zařízení jsem doposud neobjevil, abych na něm mohl tuto skutečnost otestovat.

# 6. Závěr

Cílem této práce bylo vytvořit systém, který umožňuje převádět stávající papírové archívy do elektronické podoby v podobě obrázků. Vytvářet nové archívy a provádět jejich správu. Toto téma jsem si vybral, protože mě zajímal princip získávání obrazových dat a možností jejich archivace.

Práce začala studováním požadavků pro archivaci dokumentů a převod z papírové podoby do elektronické. Požadavky jsem zjišťoval z již existujících systému umožňující archivaci dokumentů. Tyto komerční systémy jsou velice obsáhlé a mají spoustu různých funkcí, které jsou někdy pro uživatele naprosto zbytečné. Proto jsem se rozhodl vytvořit systém, který poskytuje základní možnosti vytváření a správy archívu. Výsledný systém je zaměřen spíše na data, které je nutno trvale uchovávat.

Na základě těchto zjištění jsem přešel ke konkrétní analýze požadavků. Tu tvoří specifikování cílové skupiny uživatelů a diagram případů užití, který odhaluje možnosti aplikace. Na základě analýzy je vytvořen datový model databáze, vyjádřený ER diagramem. Po té proběhl návrh uživatelského rozhraní, zaměřený na snadné a intuitivní použití a na konec byl vytvořen návrh vnitřní struktury aplikace. Na základě tohoto návrhu jsem přistoupil k implementaci aplikace.

Výsledná aplikace splňuje požadavky zadání. Možnosti dalšího rozšíření aplikace, je zejména v bezpečnosti uchovávání a přenosu dat. V této oblasti by nejlepším rozšířením bylo ukládání obrázků přímo do databáze, to by pravděpodobně vyžadovalo změnu databáze. MySQL neumožňuje ukládání obrázků, ale ukládá se jako binární obraz dat. Aplikace by za běhu online vyžadovala rychlejší přístup k databázi, protože by se zvýšil objem přenášených dat z databáze. Dalším rozšířením by mohla být automatická skartace (mazání) dat po určité době. To by vyžadovalo mírné upravení databáze, které by spočívalo v přidání sloupce do tabulek s informací o expiraci dat a vytvoření agenta, který by hlídal toto expirační datum a po vypršení doby by položku smazal. Jiná rozšíření by mohla vznikat na požadavek uživatelů.

Vhodným rozšířením by také mohlo být rozdělení uživatelů do více skupin, aktuálně administrátor a uživatel. Rozdělení by mohlo být, jak v rámci administrátorů (pro určité databáze, specifická práva apod.), tak uživatelů (pouze prohlížení apod.).

Tvorba této aplikace, jak z pohledu návrhu, tak implementace, byla jeden z největších projektů, na kterých jsem pracoval. Již zpočátku jsem si uvědomil jak je důležitý správný návrh, proto jsem této části věnoval velké úsilí, ale i přesto byla nutná občasná úprava návrhu a s tím v pozdějších fázích souvisela re-implementace. Uvědomil jsem si, že návrh kvalitní aplikace je velmi náročný a důležitý. Velkým přínosem při vytváření této aplikace pro mě byla práce s protokolem TWAIN. Tato technologie mě velice zaujala, možností použití s různými zařízeními různých výrobců a multiplatformností. Tato práce mě velice zaujala a rád bych se jí věnoval i nadále například i v diplomové práci.

# Literatura


[1] Rudolf Kaďorek, Vznik a vývoj Státního ústředního archivu v Praze, in: Aby na nic a na nikoho nebylo zapomenuto. K jubileu ústředního archivu českého státu 1954-2004, Praha 2004, s. 13 - 44.

[2] Scanner. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2012, 24.2.2012 [cit. 2012-05-02]. Dostupné z: http://cs.wikipedia.org/w/index.php?title=Scanner&oldid=8183400

[3] Database. In*: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2012, 29.4.2012 [cit. 2012-05-02]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Database&oldid=489761972

[4] XEROX CZECH REPUBLIC S.R.O. *XEROX | Digitalizace, správa dokumentů a zpracování dat* [online]. [cit. 2012-04-16]. Dostupné z: http://www.dokumentyefektivne.cz/

[5] OCÉ-ČESKÁ REPUBLIKA, s.r.o. *Océ PRISMAarchive - Řešení pro archivaci velkých objemů nativních, tiskových i skenovaných dat* [online]. [cit. 2012-04-16]. Dostupné z: http://www.oce.cz/produkt/prismaarchive

[6] DMS INOVIO Digitální archiv: *INOVIO - workflow software, zrychlení vašeho řízení*. TACTICA MANAGEMENT, s.r.o. DMS INOVIO Digitální archiv [online]. [cit. 2012-05-03]. Dostupné z: http://www.inovio.cz/a/dms-inovio-digit%C3%A1ln%C3%AD-archiv

[7] DAXIA: digitální archiv | | digitalizace archivace | skenování | verzování dokumentů. YDS S.R.O. DAXIA [online]. [cit. 2012-05-03]. Dostupné z: http://www.yds.cz/Produkty/Sprava-dokumentu-DMS-/Digitalni-archiv-Daxia/

[8] TWAIN WORKING GROUP. *TWAIN Specification* [online]. 2009 [cit. 2012-04-15]. Dostupné z: http://www.twain.org/component/attachments/download/3559.html

[9] C Sharp (programming language). In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2012, 14 April 2012 08:24 UTC [cit. 2012-04-15]. 487306861. Dostupné z: http://en.wikipedia.org/w/index.php?title=C_Sharp_(programming_language)&oldid=487306861

[10] MICROSOFT. *MSDN Library* [online]. [cit. 2012-04-15]. Dostupné z: http://msdn.microsoft.com/en-us/library

[11] WATSON, Ben. *C# 4.0*: *řešení praktických programátorských úloh*. Vyd. 1. Brno: Zoner Press, 2010, 656 s. ISBN 978-80-7413-094-6.

[12] .NET Framework. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2002, 14 duben 2012 04:10 [cit. 2012-04-15]. 487285322. Dostupné z: http://en.wikipedia.org/w/index.php?title=.NET_Framework&oldid=487285322

[13] KOFLER, Michael. *Mistrovství v MySQL 5*. Vyd. 1. Překlad Jan Svoboda, Ondřej Baše, Jaroslav Černý. Brno: Computer Press, 2007, 805 s. ISBN 978-80-251-1502-2.

[14] SQL. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2012, 16.4.2012 [cit. 2012-04-16]. Dostupné z: http://en.wikipedia.org/wiki/SQL

[15] ORACLE CORPORATION. *MySQL :: The world's most popular open source database* [online]. [cit. 2012-04-16]. Dostupné z: http://www.mysql.com/

[16] MACICH ML., Jiří. Cloudová úložiště pro vaše zálohy: račte si vybrat. *Lupa.cz* [online]. 2012[cit. 2012-04-15]. Dostupné z: http://www.lupa.cz/clanky/cloudova-uloziste-pro-vase-zalohy-racte-si-vybrat/

[17] ALTERNATIVETO. *Backup & Sync* [online]. [cit. 2012-04-15]. Dostupné z: http://alternativeto.net/category/backup-and-sync/

[18] NETMASTER. *.NET TWAIN image scanner* [online]. 2002, 13.5.2002 [cit. 2012-04-15]. Dostupné z: http://www.codeproject.com/Articles/1376/NET-TWAIN-image-scanner

# Seznam příloh

Přiložené CD s kompletními zdrojovými kódy a ZIP archív „archivace.zip“ obsahující zkompilovanou funkční aplikaci.

Přiložené CD obsahuje následující soubory a adresáře

- src — adresář obsahující kompletní zdrojové texty
- archivace.zip — archív obsahující spustitelnou aplikaci
- BakalarskaPrace.pdf — tento dokument ve formátu pdf