České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra řídicí techniky

Diplomová práce

INFORMAČNÍ SYSTÉM PRO SPRÁVU A ANALÝZU LÉKAŘSKÝCH DAT

Praha 2010 Jan Baláš

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Jan Baláš**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Informační systém pro správu a analýzu lékařských dat**

Pokyny pro vypracování:

1. Seznamte se s technologií OLAP, a to včetně způsobu, jakým systém OLAP může získávat data.
2. Navrhněte strukturu relační databáze, která bude sloužit pro úschovu dat, a daný návrh implementuje.
3. Z důvodu snadné obslužnosti, aktualizace a portability vytvořte serverové řešení v kombinaci s tenkým klientem (webové rozhraní).
4. Navržený informační systém bude určen pro ukládání, správu a analýzu lékařských dat.

Při návrhu může diplomant využít znalostí, které získal během studia v předmětech: Manažerské informační systémy, Systémy pro podporu rozhodování a Návrhy automatizovaných zařízení.

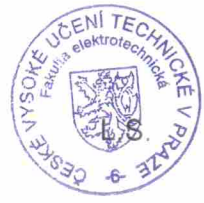
Seznam odborné literatury:

[1] A. J. Brust, S. Forte. Mistrovství v programování SQL Serveru 2005, 2007, Computer Press, ISBN: 978-80-251-1607-4.
[2] G. Spofford, S. Harinath, Ch. Webb, D. H. Huang, F. Civardi. MDX Solutions: With Microsoft SQL Server Analysis Services 2005 and Hyperion Essbase, 2005, Paperback, 2nd ed., ISBN: 978-0-471-74808-3.

Vedoucí: Ing. Jan Rusz

Platnost zadání: do konce letního semestru 2010/2011

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry



doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 26. 11. 2009

Poděkování

Rád bych na tomto místě poděkoval rodičům, kteří mě během celého studia usilovně podporovali a pomáhali překonat všechny problémy.

Také bych rád poděkoval Ing. Janu Ruszovi za veškerou pomoc při tvorbě této diplomové práce a v neposlední řadě také doc. Ing. Romanu Čmejlovi, CSc. za poskytnuté rady při tvorbě koncepce databáze a poskytnutí hardwaru, na kterém je výsledek této práce spuštěn.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne ……. …….

podpis

Anotace

Cílem této diplomové práce je navrhnout a implementovat informační systém (IS), který bude sloužit pro archivaci a správu audionahrávek z oblasti zpracování řeči a pro dokumentaci k těmto audionahrávkám. Systém bude také umožňovat archivaci a správu dalších naměřených biologických signálů, které byly pořízeny v posledních deseti letech, a zároveň poskytne prostředí pro vkládání nových dat, a to nejen lékaři či odbornými pracovníky, ale i studenty přímo v hodinách.

První část tohoto dokumentu se zabývá popisem technologií použitých při vývoji IS. Druhá část je zaměřena na implementaci výsledného systému na straně serveru, požadavky na vybavení uživatele a na rizika spojená s provozem IS. Třetí část je zaměřena na funkční strukturu a ovládání. Poslední – čtvrtá – část rozebírá návrh použité datové struktury.

Abstract

The primary goal of this thesis is to design and implement an information system (IS) suitable for archiving and management of various audio records used in speech processing and their documentation. System will also be able to archive and manage measurements of other biological signals that have been collected during past ten years. It will provide a suitable user interface usable for depositing new data by doctors, teachers and students.

The first section of this document deals with the description of technologies used during the development of the IS. The second part aims to describe implementation of the final product on server side. It also describes requirements that user must meet to be able to fully use the system. The third part focuses on the functionality structure and user interface. The last part analyses developed data structure used in the IS.

Obsah

Seznam zkratek

.NET Framework	Velké množství obecných knihoven řešících běžné úkony a virtuální stroj sloužící ke spouštění programů psaných pro tento framework.
ADO MD.NET	Rozšíření ADO.NET o podporu pro OLAP databáze.
ADO.NET	Část .NET frameworku určená pro přístup a datům a datovým službám.
AJAX	Asynchronous JavaScript And XML – technologie pro vývoj interaktivních webových stránek bez nutnosti postbacků.
ASP.NET	Webový aplikační framework - součást .NET frameworku.
C#	Objektově orientovaný jazyk vyvinutý pro platformu .NET.
C++	Objektově orientovaný programovací jazyk s přímou podporou v .Net frameworku.
CLI	Common Language Runtime – otevřená specifikace popisující vlastnosti proveditelného kódu a jeho prostředí pro běh (runtime environment) použité v .NET frameworku.
CLR	Common Language Runtime – Implementace CLI specifikace vytvořená firmou Microsoft (existuje ještě open source implementace Mono, která je vyvíjená převážně firmou Novel).
CLS	Common Language Specification – seznam pravidel, které musí každý CLI kompatibilní jazyk splňovat.
CSS	Cascading Style Sheets – kaskádní styly sloužící k popisu zobrazení dat.
CTS	Common Type System – standart specifikující, jak jsou definice typů a specifické hodnoty typů uloženy v paměti počítače.
DOM	Document Object Model – objektově orientovaná reprezentace XML dokumentu.
DTD	Document Type Definition – množina značek použitých pro značkování dokumentu vytvořeném pomocí značkovacího jazyka jako je XML nebo HTML.

FTP	File Transfer Protocol – nezabezpečený protokol aplikační vrstvy sloužící pro přenos souborů mezi počítači.
FTPS	FTP Secure – rozšíření FTP protokolu o podporu zabezpečení užitím SSL/TLS.
GAC	Global Assembly Cache – Cashe určená pro částečně zkompilovaný kód určený pro CLR, který má být sdílen s více aplikacemi. Cashe je jednotná pro celý počítač.
HTML	HyperText Markup Language – značkovací jazyk používaný pro tvorbu webových stránek.
HTTPS	Hypertext Transfer Protocol Secure – HTTP využívající k šifrování SSL/TLS protokol pro zajištění bezpečné kounikace.
IIS	Internet Information Services – webový server vyvinutý firmou Microsoft a určený pro operační systém Windows.
JIT	Just-In-Time compilation – dynamický překlad. Technika překladu do strojového kódu za běhu programu.
LINQ	Language Integrated Query – jazyk určený pro dotazování nad jakýmikoli daty. Je součástí .NET frameworku od verze 3.5.
MDX	Multidimensional Expressions – dotazovací jazyk podobný SQL určený pro OLAP datatabáze.
MSIL	Microsoft Intermediate Language, zvavý též Common Intermediate Language – nejnižší člověkem čitelný jazyk používaný v .NET.
OLAP	Online Analytical Processing – technologie uložení dat v databázi usnadňující hloubkovou analýzu.
SATA	Počítačová sběrnice určená k propojení paměťových zařízení.
SQL	Dotazovací jazyk používaný pro práci s daty v relačních databázích.
SŘDB	Systém řízení báze dat - programové vybavení, které řídí organizaci, ukládání, správu a získáváni uložených dat v databázi.
SSL	Secure Sockets Layer - Protokol zabezpečující bezpečnou komunikaci prostřednictvím šifrování.

SSMSE SQL Server Management Studio – aplikace určená pro práci a správu Microsoft SQL serveru.

TDS Tabular data stream – protokol aplikační vrstvy určený pro přenos dat mezi databázovým serverem a klientem.

TLS Transport Layer Security – protokol určený pro šifrování obsahu. Je určený pro zabezpečení komunikace po internetu.

VB.NET Programovací jazyk Visual Basic vystavěný nad platformou .NET frameworku.

W3C World Wide Web Consortium – mezinárodní standardizační organizace pro World Wide Web.

XHTML Extensible Hypertext Markup Language – nástupce HTML splňující podmínky XML.

XML Extensible Markup Language – obecný značkovací jazyk.

1 Úvod

V dnešní době informačních technologií dochází velkému nárůstu množství dat všech typů a lékařství není výjimkou, spíše naopak. S množstvím dat však vyvstává nový problém, a sice jak s takovým objemem dat pracovat. Od určitého množství se stává přechovávání „po šuplících“ velmi neefektivní a náchylné ke ztrátě dat, která bývají velmi cenná a nenahraditelná. V takové situaci přicházejí na řadu informační systémy (IS). Typický IS je softwarová aplikace určená pro ukládání, správu, analýzu a prezentaci rozličných typů dat. K IS má běžně přístup velké množství uživatelů. V lékařství je třeba - více než kde jinde - klást důraz na bezpečné uložení a zabránění neoprávněnému přístupu k citlivým informacím.

Cílem této diplomové práce je navrhnout a implementovat IS, který bude sloužit pro archivaci a správu audionahrávek z oblasti zpracování řeči a pro dokumentaci k těmto audionahrávkám. Systém bude také umožňovat archivaci a správu dalších naměřených biologických signálů, které byly pořízeny v posledních deseti letech, a zároveň poskytne prostředí pro vkládání nových dat, a to nejen lékaři či odbornými pracovníky, ale i studenty přímo v hodinách.

Prvotní návrh databáze bude využit přednostně pro řečové nahrávky pořízené na Foniatrické a Neurologické klinice 1. Lékařské fakulty University Karlovy v Praze. Ty se skládají z oblasti dysartrie u Parkinsonovy nemoci, dětské vývojové dysfázie, koktavosti, věkově závislých parametrů a dalších. Další využití databáze bude pro nahrávky pacientů s kochleárními implantáty. Ke každé skupině může být dle potřeb také pořízena kontrolní skupina zdravých mluvčích. Posledním v současnosti předpokládaným využitím databáze bude sběr různorodých biologických signálů studentů, kteří budou mít posléze k databázi přístup a budou tak moci v rámci výuky testovat navržené algoritmy na celé řadě těchto signálů. Navržený systém bude také velmi vhodný pro zálohu těchto jedinečných dat.

Součástí IS bude také samostatný exportní systém, který umožní exportovat nahrané soubory do zip archívu a zadaná data do běžných formátů jako je .doc, .csv, .xls nebo .pdf. Dále bude obsahovat možnost základní analýzy přímo prostřednictvím uživatelského rozhraní. Celý systém bude tvořen pomocí webový technologií, což umožní jeho snadnou rozšiřitelnost – uživatel bude potřebovat pouze webový prohlížeč, který je dostupný na drtivé většině osobních počítačů.

V první kapitole budou popsány technologie použité během vývoje aplikace.

Druhá kapitola se zaměří na skutečnou implementaci výsledného programu. Budou zde popsány požadavky na hardware a software, které musí být splněny pro správný chod aplikace, a to jak ze strany uživatele, který chce aplikaci využívat, tak požadavky na server, na kterém jsou data a samotná aplikace umístěny. Budou zde také popsány použité webové a SQL servery a důvody pro jejich nasazení. Dále zde budou zmíněna bezpečnostní rizika, která se při reálném nasazení vyskytují, a dále postupy použité pro jejich zamezení.

Ve třetí kapitole je popsána funkční struktura a logické členění výsledné aplikace. Jsou zde vysvětleny principy správy a jmenného zapouzdření aplikací, uživatelů, jednotlivých datových prvků (pacientů, kontrolních pacientů a vybavení) a k nim přiřazených souborů. Pro datové objekty je zde ukázán princip tvorby a správy jejich vlastností. Pro zadaná data je zde přiblížen způsob analýzy a exportu Jsou zde dále ukázány principy autentizace a autorizace k jednotlivým částem aplikace a jejím prvkům.

Čtvrtá kapitola se zaměřuje na strukturu použitou pro ukládání dat do databáze. V konceptuálním modelu jsou popsány základní entity a vazby mezi nimi.

2 Technologie použité pro vývoj

2.1 XHTML

XHTML (Extensible Hypertext Markup Language), jak již název napovídá, vychází ze dvou jazyků, a sice HTML (HyperText Markup Language) a XML (Extensible Markup Language). XHTML měl nahradit HTML 4.01, jehož vývoj měl být tímto ukončen. První verzi doporučila World Wide Web Consortium (W3C) roku 2000.

Hlavní výhodou nového formátu byla lepší spolupráce s ostatními datovými formáty. Toho bylo dosaženo právě díky užití XML, formátu navrženého čistě pro výměnu elektronických dat, bez ohledu na jejich formátování. XHTML stejně HTML vychází z SGML (Standard Generalized Markup Language), ovšem na rozdíl od svého předchůdce aplikuje zjednodušenou verzi, která je snažší na použití parserů.

Jelikož XML neobsahuje žádné předdefinované značky, které jsou třeba pro automatickou kontrolu, zda je dokument naformátován dle definice, a zároveň XHTML musí být řádně naformátované XML, je třeba používané značky vždy dodefinovat. Proto v každé stránce využívající XHML musí být obsažen DOCTYPE, který asociuje daný dokument DTD (Dokument Type Definition) souborem. Příslušné definice se nacházejí na stránkách W3C[1].

Mimo vývoj XHML začal roku 2007 souběžně vývoj HTML5, do jehož specifikace se od roku 2009 dostala i specifikace XHTML 5. Hlavní cíl vývoje HTML 5 je využití současných osvědčených pravidel používaných v XHTML a přidání nových značek, které omezí závislost na proprietálních zásuvných modulech jako jsou Adobe Flash, Java FX, nebo Microsoft Silverlight.

Více podrobností lze získat v literatuře [4], [12], [22], [23].

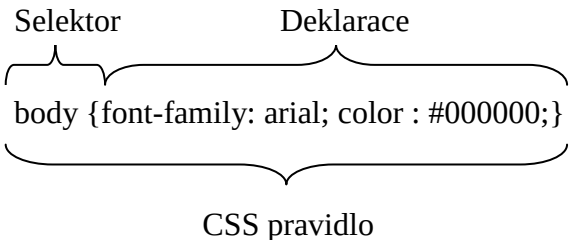
2.2 CSS

Specifikace pro CSS (Cascading Style Sheets), v češtině většinou označované jako kaskádní styly, byla vyvinuta a je udržována W3C konsorciem. Hlavní důvod pro tvorbu tohoto nového jazyka bylo oddělení vzhledu dokumentu od jeho dat a vnitřní struktury.

[1] http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd

Díky použití CSS pravidel lze každému objektu přiřadit vlastnosti, podle kterých bude následně prezentován v závislosti na prezentační platformě (tiskárna, webový prohlížeč, mobilní telefon atd.). Další výhodou CSS je výrazně snazší údržba stylu webové aplikace (při změně stylu se nemusí formátovat každý prvek separátně, ale pouze se aktualizuje příslušný CSS formát). CSS také na rozdíl od klasického HTML nabízí podstatně širší možnosti formátování.

CSS se skládá ze sad pravidel, kde každé pravidlo je složeno ze selektoru a bloku deklarací. Selektor označuje, na jakou část dokumentu budou aplikovány jednotlivé vlastnosti z bloku deklarací; v bloku deklarací jsou pak zobrazovací vlastnosti, oddělené středníky.



Obr. 1 Stavba kaskádového stylu

Jeden dokument může být ovlivněn více samostatnými pravidly. Pro jednoznačné určení výstupního stylu obsahuje CSS prioritní pravidla. Díky tomuto systému je na každý prvek dokumentu aplikováno pravidlo s nejvyšší prioritou – odtud vzniklo označení „kaskádní“.

Samotná pravidla mohou být psána třemi způsoby:

- přímo do prvku, jenž formátuje;
- do hlavičky stránky, ve které se jsou prvky, které formátuje;
- do separátního souboru s příponou .css, který musí být v hlavičce formátovaného souboru připojen pomocí elementu link.

Přes zjevné výhody má využití CSS stylů i značné nevýhody. Tou největší je rozdílná interpretace stylů v různých prohlížečích, což má za následek, že stejná stránka se může zobrazovat jinak např. v Internet Exploreru než v ostatních prohlížečích. Nutno podotknout, že v poslední době se situace výrazně zlepšuje, obzvláště s příchodem Internet Exploreru 8.

Více podrobností lze získat v literatuře [4], [12], [13], [21].

2.3 Témata

Témata - stejně jako CSS styly - mají na starost formátování dokumentu. Témata nejsou konkurencí stylů, spíše fungují jako jejich doplněk tam, kde možnosti stylů končí. Pomocí CSS lze dobře naformátovat obecné vlastnosti jako fonty, okraje nebo pozadí; specifické vlastnosti ASP.NET ovládacích prvků s nimi nicméně nastavit nelze. Aby bylo možné v rámci celé stránky unifikovat složitější ovládací prvky, je proto vhodné použít právě témata.

Tvorba témat je podobná tvorbě CSS stylů. Každé téma musí mít svůj vlastní soubor s koncovkou .skin, který je umístěn ve složce App_Themes. Ta se nachází ve složce aplikace, pro kterou je téma použito. Samotný .skin soubor obsahuje sadu tagů, jimiž je daný ovládací prvek definován. V tématu nemusí být použita všechna nastavení prvku; ta, která nejsou nastavena, používají defaultní hodnoty.

Více podrobností lze získat v literatuře [1], [4], [12].

2.4 JavaScript

JavaScript[2] je objektově orientovaný skriptovací jazyk, původně vyvinut firmou Netscape. Přestože jej lze použít jako běžný programovací jazyk aplikací, jeho nejběžnější využití je přímo v (X)HTML kódu stránky, kde zajišťuje operace na straně klienta. Toto využití má velkou výhodu v tom, že není třeba odesílat data zpět na server, čímž se práce s webovou aplikací zrychlí. O interpretaci skriptu se stará klientův prohlížeč. JavaScript může reagovat na uživatelské události, tvořit vizuální efekty nebo provádět validaci vstupních údajů. Jelikož je JavaScript spouštěn přímo na uživatelově počítači, může sloužit jako přístupová cesta pro škodlivý kód. Aby se tomuto zabránilo, používají se dva základní ochranné prvky:

- v rámci prohlížeče běží skripty uvnitř tzv. sandboxu, kde mohou provádět pouze operace spojené s webem, nikoli běžné programovací operace, jako jsou například práce se soubory;

[2] Slovo Java v názvu je pouze marketingový trik, ve skutečnosti nemá JavaScript s Javou téměř nic společného.

- skript má přístup pouze k informacím pocházejícím ze stejného zdroje jako samotný skript – nemůže tedy přistupovat k heslům nebo cookies získaných z jiných webových stránek.

Více podrobností lze získat v literatuře [16], [20].

2.5 ASP.NET

ASP.NET je framework určený pro tvorbu dynamických webových aplikací. Byl vyvinut firmou Microsoft a poprvé představen v roce 2002 spolu s první verzí .NET frameworku. Historický předchůdcem tohoto jazyku je ASP (Active Server Pages), což je skriptovací jazyk vyvinutý pro IIS ve Windows NT4.0. ASP.NET; stejně jako jeho předchůdce ASP není vázán na žádný konkrétní jazyk. Pro vývoj může být použit jakýkoliv jazyk, který je podporován platformou .NET. Oficiálně podporované jsou C#, VB.NET, Managed C++ a JScript .NET, nicméně existují desítky dalších[3]. Framwork .NET navíc neposkytuje pouze nezávislost na použitém programovacím jazyku, ale umožňuje i jazykovou „integraci" – tedy dědit třídy, zachytávat výjimky, nebo využívat polymorfismu napříč různými jazyky.

Výše popsané interoperability lze dosáhnout tím, že všechny .NET komponenty dodržují specifikaci CTS (Common Type System). CTS obsahuje základní koncepty tříd, rozhraní delegátů, referenční a hodnotové typy a určuje, že každý objekt dědí z kořenové třídy System.Object.

Navíc .NET framework obsahuje specifikaci CLS (Common Language Specification), která poskytuje základní pravidla a povinnosti pro každý .NET kompatibilní jazyk. Díky tomu spolu mohou všechny compilery splňující tuto specifikaci spolupracovat.

Výsledné programy nejsou kompilovány do spustitelných souborů, jak je tomu zvykem například při kompilaci pomocí gcc. Místo toho je výsledkem kompilace tzv. MSIL[4] soubor (Microsoft Intermediate Language), který může být následně spuštěn v CLR[5] (Common Language Runtime). Velmi podstatné je, že výstupní MSIL soubor bude vždy shodný, nezávisle na tom, jaký .NET jazyk bude použit pro jeho výrobu.

[3] Podrobný seznam podporovaných jazyků lze najít zde:
http://www.dotnetpowered.com/languages.aspx
[4] Obdoba byte kódu známého z Javy
[5] Obdoba Java Runtime Environment

V okamžiku spuštění samotné aplikace dojde k opětovnému zkompilování MSIL souboru, nyní pomoci JIT (Just In Time) kompileru. Výsledkem této kompilace je již strojový kód. JIT kompiluje na vyžádání a kompiluje pouze kód, který pro svůj běh potřebuje. To způsobuje, že .NET aplikace mají tendenci fungovat s časem „svižněji“, protože výsledek již byl předem předkompilován.

Jistou nevýhodou užívání ASP.NET je jeho vázanost na společnost Microsoft a její operační systémy. Tuto závislost je možné z části obejít užitím open source projektu Mono, vyvíjeným převážně společností Novel. Mono poskytuje platformně nezávislý kompiler a CLR, částečně kompatibilní s aktuálním .NET frameworkem – aktuální implementace je přibližně na úrovni .NET 3.0.

Více podrobností lze získat v literatuře [1], [4], [11], [12].

2.6 LINQ

LINQ (Language Integrated Query) je univerzální dotazovací jazyk, určený pro tvorbu dotazů nad jakýmikoliv daty, pokud implementují interface IEnumerable<>, a to nezávisle na tom, zda se jedná o pole, list, XML, DOM, nebo vzdálený zdroj (např. tabulka v SQL databázi). Hlavní síla tohoto nástroje tkví v jeho univerzálnosti.

Typický zápis pro výběr vybraných hodnot v CheckBoxListu:

```
var vybranéHodnoty = Chbl.Items.AsEnumerable().Where(c => c.Selected=true)
.Select(c => new {c.Value});
```

Více podrobností lze získat v literatuře [8], [9].

2.7 Telerik ASP.NET AJAX Controls

Knihovna obsahující sadu ovládacích prvků, které rozšiřují základní prvky obsažené v ASP.NET. Volba na tuto knihovnu padla především díky její výborné dokumentaci, grafickému zpracování komponent, přehlednému kódu a v neposlední řadě i dobrému propojení s LINQ. Užitečnou vlastností je též podpora v linuxových distribucích pomocí projektu Mono, což může být užitečné v případě migrace webové části na jinou platformu než Windows.

Více podrobností lze získat v literatuře [16], [20].

2.8 SharpZipLib

SharpZipLib je open source knihovna umožňující kompresi a dekompresi formátů gzip a zip. Knihovna je vytvořena pro .NET platformu, přičemž její zdrojový kód je psán čistě v C#. Autor poskytuje jak zdrojové kódy, tak GAC na svých webových stránkách[6]. Jelikož jsou k dispozici kompilace pouze pro .NET verze 2 a nižší, je při vývoji použita vlastní kompilace určená pro .NET 3.5.

Více podrobností lze získat v literatuře [6].

2.9 User controls

User controls (uživatelské ovládací prvky) umožňují tvorbu vlastních ovládacích prvků v okamžiku, kdy ovládací prvky dostupné v ASP.NET nejsou dostačující. Jsou složeny z existujících ovládacích prvků, mohou obsahovat HTML kód, vlastní vlastnosti a metody obdobně jako ASP.NET webové stránky, ovšem na rozdíl od nich nemohou fungovat samostatně, ale musí být užity uvnitř ASP.NET stránky. V aplikaci jsou využity při generování ovládacích prvků pro různé vlastnosti objektů (viz. 4.9).

Více podrobností lze získat v literatuře [14].

2.10 ADO.NET

ADO.NET je technologie, kterou .NET aplikace využívají pro práci s databází. Třídy, které obsahuje, poskytují dvě základní funkcionality:

- správa a uchování dat - sem patří např. DataSet, DataTable, DataRow a DataRelation;
- připojení ke zdroji dat – příkladem je Connection, Command, a DataReader.

Třídy pro správu dat jsou kompletně generické, tj. nezávislé na zdroji dat. Jistou výjimku tvoří DataSet, který je navíc uzpůsoben pro relační data – podporuje nativně koncept řádků, sloupců, tabulek a relací mezi tabulkami.

Na rozdíl od správy dat se třídy zajišťující připojení ke zdroji dat navzájem výrazně liší. Pro každý zdroj dat existuje specifický data provider, který je upraven tak, aby co

[6] http://www.icsharpcode.net/OpenSource/SharpZipLib/Download.aspx

nejlépe využil všech možností zdroje dat (např. pro MS SQL server nativní provider poskytuje podporu pro TDS[7]).

Více podrobností lze získat v literatuře [4], [8], [9].

2.11 OLAP

OLAP (**o**n**l**ine **a**nalytical **p**rocessing) je databázová technologie určená k hlubokým analýzám strukturovaných dat. Zároveň je na rozdíl OLTP[8] určena spíše pro práci s historickými než aktuálními daty, protože pro aktualizaci datového zdroje je třeba provést aktualizaci příslušné OLAP krychle.

Základním pojmem v OLAPu je *krychle*, což je obdoba tabulky v relační databázi. Krychle se skládá z měr a dimenzí. *Míry* jsou číselná data, která chceme analyzovat (např. výsledky měření). *Dimenze* jsou oblasti, ve kterých jsou hodnoty hledány (např. typy měření). Jednotlivé dimenze mohou mít vlastní hierarchii s různými členy v jednotlivých úrovních (např. rok, měsíc, týden, den).

Pro přímou práci s krychlí je možné využít programu MS Excel, který podporuje práci s kontingenčními tabulkami. Na webu je možné použít ovládacích prvků PivotTable a ChartSpace (vyžadují podporu activeX). Je-li ovšem třeba pracovat s krychlí programově, musí se použít speciální jazyk vytvořený pro tvorbu vícerozměrných dotazů – MDX (**M**ulti**D**imensional e**X**pressions). Typický MDX vypadá takto:

```
SELECT
[sloupec] ON COLUMNS,
[řádky(dimeze)] ON ROWS
FROM [krychle]
WHERE [filtrovací podmínka]
```

Programové získávání dat z analytických služeb je téměř identické se získáváním dat z relačních databází díky řízenému poskytovateli dat ADO.NET (viz. 2.10) s názvem ADO MD.NET. S ADO MD.NET lze zacházet stejně jako s jakýmkoli jiným poskytovatelem ADO.NET, tj. spustit MDX příkazy přes objekty Connection a Command a pro práci s výsledky použít DataReader, DataAdapter, DataTable a DataSet. Navíc obsahuje další objekty specifické pro OLAP, kterými je možné podrobněji zkoumat jednotlivé krychle a pracovat s metadaty.

[7] Tabular data stream – protokol aplikační vrstvy určený pro přenos dat mezi databázovým serverem a klientem

[8] Online transaction processing – technologie uložení a správy dat založená na transakcích. Typicky zaručují rychlé vkládání dat a aktualizaci.

V průběhu vývoje aplikace se ukázalo, že hluboce strukturovaná data předem dané struktury nejsou nejvhodnějším způsobem zaznamenávání dat v rámci aplikace. Naopak se jako vhodnější ukázal návrh dynamických vlastností, u nějž je možné vlastnosti přidávat a měnit přímo za běhu; pro tuto strukturu dat je ovšem využití OLAPu krajně nevhodné. Prvním důvodem je skutečnost, že není možné vystavět krychli přímo nad databází, ve které jsou data uložena (logická struktura vlastností není dána tabulkami databáze, ale je uložena přímo s daty), ale musí být vystavěna samostatná databáze na základě vazeb mezi jednotlivými uloženými vlastnostmi. Tato pomocná databáze musí být vždy kompletně vystavěna pro každou aktualizaci krychle. Dalším důvodem bylo to, že dynamicky vytvářené vlastnosti nejsou na sobě závislé (jsou pouze řazeny do skupin), tudíž mizí výhoda tvorby hierarchických dimenzí a zůstává pouze výhoda rychlosti odpovědi na složitější dotazy vykoupená neaktuálností analyzovaných dat a jejich náročnou aktualizací. Z výše popsaných důvodů nebylo ve finální verzi využito OLAPu pro přístup k datům v databázi.

Více podrobností lze získat v literatuře [2], [18], [22].

3 Implementace

3.1 Hardwarové vybavení serveru

Na provoz aplikace byl pořízen server, který je nyní umístěn v serverovně katedry teorie obvodů K13131. Při návrhu konfigurace byl brán zřetel převážně na následující věci:

- objem dat (přestože velikost samotné webové aplikace je zanedbatelná, data, pro která je převážně určena, tedy audio- a videonahrávky dosahují již v současné době řádově stovky GB);
- cennost dat (nenahraditelná data získaná za mnoho let měření);
- výkon.

Výsledná sestava:

- procesor Core 2 Duo E8400;
- pevný disk 2x 1,5TB, 7200 otáček, SATA II;
- grafická karta GeForce 8400 GS;
- paměť 4GB DDR II;
- integrovaná síťová karta s maximální přenosovou rychlostí 1GB/s (full-duplex).

Pro zvýšení bezpečnosti byl pořízen externí síťový disk, určený pro pravidelné ukládání záloh.

3.2 Softwarové vybavení

3.2.1 Operační systém

Jelikož webový server, na kterém je aplikace provozována, je umístěn na stejném stroji jako databázový server, byl použit operační systém Windows Web Server 2008. Přestože je webová aplikace napsána v ASP.NET, jenž je určen pro operační systém Windows, je možné ji spustit pomocí projektu Mono i na jiných operačních systémech.

3.2.2 Databázový systém

Databázový systém se obecně skládá ze dvou samostatných částí, kterými jsou:

- systém řízení báze dat (SŘDB) - programové vybavení, které řídí organizaci, ukládání, správu a získáváni uložených dat;
- databáze – skutečná uložená data.

V tomto oddílu se budu věnovat výběru SŘDB. Pro výběr připadala v úvahu dvě velmi podobná řešení, a to řešení od firmy Oracle a Microsoft. MS SQL Server 2008 byl vybrán především z následujících důvodů:

3.2.2.1 SQL nativní klient

Díky existenci propracovaného nativního klienta je možné využívat všech funkcí databázového serveru přímo z kódu aplikace. K těmto funkcím patří například podpora všech datových typů podporovaných SQL serverem, podpora šifrování či podpora pro ADO.NET[9]. Další výhodou je zvýšená bezpečnost, jelikož veškerá přenášená data mezi SQL a web serverem jsou šifrována, a není tedy možné získat citlivé informace pouhým poslouchánim komunikace.

3.2.2.2 Nástroje

Pro plynulý a rychlý vývoj je třeba mít k dispozici kvalitní vývojářské nástroje.

- **SQL Server Management Studio** (SSMSE) - pro práci se samotnou databází a její následnou údržbu je velmi užitečná aplikace, která je dodávána přímo s SQL serverem. SSMSE obsahuje nástroje pro návrh databáze a její „živou“[10] úpravu. Mezi další potřebné vlastnosti patří náhledy do existujících tabulek a aktualizace jejich obsahu nebo export a import SQL skriptů, které se dají použít při zakládání nové databáze na dalším stroji. Dále poskytuje komplexní správu práv jak k samotnému databázovému serveru, tak k jednotlivým instancím databáze.
- **Visual Studio** nabízí přímé spojení s SQL Server 2008, včetně podpory při modelování databáze. Nespornou výhodou je přítomnost intellisence[11] při tvorbě SQL dotazů.

[9] Přístup k databázi skrz .NET framework
[10] Není třeba pracovat v offline režimu. SSMSE promítne navržené změny do databáze rovnou za běhu.
[11] Automatické doplňování kódu

3.2.3 Webový server

Pro vývoj i ostré nasazení byl použit webový server od firmy Microsoft IIS (Internet Information Services). IIS ve skutečnosti není pouze webový server, ale spíše kolekce programů, z nichž jedna část[12] poskytuje webové služby. Díky využití operačního systému Windows Web Server 2008 jsme získali přístup k IIS 7.0, který přináší mnohá vylepšení oproti předešlým verzím i oproti konkurenci. Seznam hlavních důvodů pro výběr právě tohoto webového serveru bude popsán na následujících řádcích.

Více podrobností lze získat v literatuře [7], [17].

3.2.3.1 Jednotná autentizace a autorizace

Od verze 7.0 došlo ke sjednocení autentizace a autorizace IIS a autentizace a autorizace prostřednictvím ASP.NET. Díky tomu je možné používat autentizací proces nad složkami a soubory přímo z ASP.NET. Odpadá tím problém autentizace souborů[13], kdy bylo nutné využívat Windows autentizaci, na kterou bylo potřeba vlastnit samostatnou licenci.

3.2.3.2 Snadná správa

IIS Manager pro IIS 7.0 kombinuje správu všech služeb jak pro samotný IIS, tak pro ASP.NET. Velmi podstatnou výhodou je možnost delegace rozličných administrátorských práv mezi správce serveru[14]. Delegace lze efektivně využít pro vzdálenou správu systému, kdy je možné vybrané zpřístupnit část aplikace nebo i celý server vzdálenému uživateli prostřednictvím HTTPS.

Přínos pro snadnou administraci je též v možnosti spravovat celý IIS skrze příkazový řádek. Pro tento druh správy slouží utilita AppCmd.exe. Hlavní přínos tohoto přístupu je především v možnosti dávkového zpracování, např. pomocí Power Shellu.

Příklad pro tvorbu nové webové stránky:

```
appcmd.exe add site /name:NovaStranka /id:999 /bindings:
”http/*:80:”/fyzickaCesta:”C:\inetpub\NovaStranka”
```

[12] Další poskytované služby jsou například FTP server se SSL podporou (vhodný pro aktualizaci dat na webovém serveru při vývoji), SMTP server a další

[13] Problém se vyskytoval převážně u PDF souborů.

[14] IIS poskytuje koncept IIS uživatelů – tedy uživatelů, kteří existují pouze v rámci samotného IIS, nikoliv na úrovni systému. Hlavní výhoda tohoto řešení je, že nejsou spotřebovávány placené klientské licence a zároveň takto vzniklý uživatel nemůže uškodit systému, na kterém IIS běží.

Pro vývojáře je praktické využít zabudovaný FTP server pro rychlou aktualizaci či pro umístění nového obsahu[15]. Od verze IIS 7 je možné využít FTP over SSL, čímž se výrazně zvyšuje bezpečnost přenosu dat. Je možné využít shodný certifikát pro FTPS i pro IIS.

3.2.3.3 Diagnostika

Ve verzi 7.0 byla poprvé představena tato velmi užitečná vlastnost pro vývojáře, tak pro správce. Díky Request tracing modulu je možné nastavit logování a trasování jakéhokoli obsahu nebo výsledku běhu kódu[16].

Mezi typické příklady patří událost File Not Found (chybový kód 404). Pokud nastane tato událost, každý dotaz ze strany klienta a odpověď ze strany serveru jsou zalogovány včetně příslušného času.

3.3 Požadavky na tenkého klienta

V rámci bezpečnosti probíhá veškerá komunikace mezi klientem a serverem šifrovaně za použití protokolu HTTPS (viz 3.4.1). Je proto nutné, aby klientský prohlížeč tento protokol podporoval. Dále je třeba, aby certifikát serveru byl na straně uživatele přidán mezi důvěryhodné certifikáty. Pro plnohodnotné využití aplikace je nutné, aby měl klientský prohlížeč povoleno využití JavaScriptu. Při použití Internet Exploreru navíc musí být aktivováno cashování a skriptování Pro absolvování testů navíc musí být nainstalovaný plugin schopný přehrát streemované audio – ideálně Windows Media Player 11[17]. Pro systémy od firmy Apple je možné použít rozšíření Flip4Mac[18]k pro QuickTime. Pro různé linuxové distribuce existují pluginy VLC[19] nebo Totem[20].

Pro správné fungování je třeba. aby použitý prohlížeč splňoval minimální požadavky pro zobrazení ASP.NET stránek a požadavky pro správné fungování komponent Telerik. Podporované prohlížeče, které těmto požadavkům vyhovují, jsou zobrazeny v Tab. 1.[21]

[15] Přístupové účty jsou shodné s účtu, které se používají pro správu samotného IIS
[16] Nastavení sledování lze nastavit globálně na úrovni IIS, nebo pouze na úrovni dané aplikace
[17] http://www.microsoft.com/mediaplayer/
[18] http://www.apple.com/downloads/macosx/video/flip4macwindowsmediacomponentsforquicktime.html
[19] https://addons.mozilla.org/cs/firefox/addon/14730
[20] http://projects.gnome.org/totem/#download
[21] Zdroj: http://www.telerik.com/products/aspnet-ajax/resources/browser-support.aspx

Prohlížeč	Windows	Mac OS	Linux
Internet Explorer	6.0+	Ne	Ne
Firefox	2.0+	2.0+	2.0+
Google Chrome	2.0+	Ne	Ne
Opera	9.0+	9.0+	Ne
Safari	3.0+	3.0+	Ne
Netscape	9.0+	9.0+	9.0+

Tab. 1 Seznam podporovaných prohlížečů

Výsledná aplikace byla testována na prohlížečích Google Chrome 4.1, Firefox 3.6 a Internet Explorer 8. U netestovaných prohlížečů se může stát, že aplikované CSS styly nebudou zobrazeny dle předpokladů.

3.4 Bezpečnost

3.4.1 SSL

SSL (Secure Socket Layer) je šifrovací protokol pracující mezi třetí (transportní) a čtvrtou (aplikační) vrstvou architektury TCP/IP. Jeho cílem je zajistit bezpečnou komunikaci mezi komunikujícími stranami (v našem případě mezi klientským prohlížečem a web serverem) v nezabezpečeném prostředí, jako je např. internet.

Šifrování SSL funguje na principu asymetrické šifry, kdy existují dva šifrovací klíče – veřejný a soukromý. Veřejný klíč je dostupný pro všechny uživatele a je použit pro šifrování odchozích zpráv klienta při handshakeu[22]. Soukromý klíč je použit pro rozkódování zprávy od klienta během handshaku a má ho k dispozici pouze web server. Při handshaku je na straně serveru i klienta vygenerován na základě získaných informací unikátní klíč, který je používán pro kódování a dekódování následné komunikace.

Využití šifrované komunikace vyžaduje větší nárok na výpočetní výkon na straně klienta i serveru. Jelikož data uložená ve webové aplikaci mohou obsahovat citlivé údaje o pacientech včetně autentických záznamů, je šifrovaná komunikace i přes zvýšené nároky na hardware vyžadována pro veškerou komunikaci mezi klientem a web serverem. Pokusí-li se klient přistoupit stránku prostřednictvím nešifrovaného HTTP protokolu, je automaticky přesměrován na přihlašovací stránku, která využívá protokol HTTPS.

[22] Handshake je proces, který proběhne mezi entitami, které spolu chtějí komunikovat před faktickým začátkem komunikace. Během handshaku jsou nastaveny parametry komunikačního kanálu, které budou dále využity při komunikaci.

3.4.2 Přístup k souborům

Během práce s aplikací dochází ke zpracovávání velkého množství souborů, které mohou mít citlivý obsah, proto je třeba, aby se k souborům nedostal nikdo neoprávněný. Aby nebylo možné soubor získat pouhým zadáním URL do webového prohlížeče, je adresářová struktura, do které jsou soubory ukládány (viz Obr. 5), umístěna mimo adresní prostor webové aplikace. Tím je zajištěno, že přístup k souborům je možný pouze prostřednictvím kódu spuštěného na straně serveru.

Druhý stupeň ochrany adresáře, v němž jsou uloženy soubory včetně všech podadresářů, je zajištěn využitím rozdílných Application Pool Users pro autentizované uživatele a pro neautentizované uživatele. Application Pool Users jsou uživatelské účty, které IIS využívá[23] pro přístup ke zdrojům serveru, jako je disk, registry či síť. Nepřihlášení uživatelé využívají účet, který nemá právo čtení ani zápisu na disku, což zabrání úniku dat i v případě, že se útočníkovi podaří spustit škodlivý kód na straně serveru[24]. Díky maximálně omezeným právům tohoto účtu není možné spuštěním škodlivého kódu ohrozit ani samotný server.

Třetí úroveň ochrany existuje pro přihlášené uživatele. Ani přihlášený uživatel nemá k souborům přímý, ale pouze zprostředkovaný přístup. Během tohoto přístupu je ověřováno, zda uživatel, který žádá přístup k souborům, k nim má dostatečná práva (viz 4.6.1).

3.4.3 Práce s databází

3.4.3.1 Přístup k databázi

Pro přístup web serveru do databáze je využito SQL serverové autentizace[25]. Účet, který IIS využívá, má práva připojení k databázi (connect), čtení (db_datareader) a spouštění uložené procedury (execute). Tento profil neumožňuje přímý zápis do databáze ani manipulaci s tabulkami. Díky takto nastaveným bezpečnostním pravidlům nemůže

[23] IIS pro práci se systémem využívá pracovní proces w3wp.exe. To je také proces, který je třeba připojit v případě, že je nutné webovou aplikaci debugovat.
[24] Kód snažící se získat přístup je systémem zakázán, jelikož uživatel, který se jej snaží spustit, nemá práva k disku přistupovat.
[25] Druhou možností bylo využít Windows autentizace, která ovšem vyžaduje, aby server i uživatel, který se snaží získat přístup, pocházeli ze stejné Windows server domény.

útočník zničit uložená data ani v případě, že se mu podaří poslat škodlivý SQL kód prostřednictvím IIS.

3.4.3.2 Zápis do databáze

Zápis do databáze je nejnáchylnějším místem k poškození dat, jelikož zde dochází k fyzické změně dat v databázi, během čehož se může databáze dostat do nekonzistentního stavu v případě hardwarové chyby nebo špatně provedeného SQL příkazu. Aby k tomuto stavu nedošlo, je pro úpravu databáze (aktualizace i přidávání nových dat) využito transakcí.

Transakce je metoda, v rámci které vývojář definuje „pracovní jednotku" – jednotku, po jejímž provedení se databáze nachází v konzistentním stavu. Každá transakce tvoří jednu jednotku a musí splňovat čtyři základní podmínky:

- *Atomicita – veškeré* modifikace v rámci jedné transakce musí být úspěšně zapsány do databáze, nebo *žádná* změna nesmí být provedena.
- *Spojitost* – v okamžiku aplikování (commit) všech změn nebo vrácení se k výchozímu stavu (rollback) se musí databáze nacházet v konzistentním stavu včetně zachování integrity dat.
- *Izolovanost* – modifikace provedené v rámci jedné transakce musí být izolované od modifikací provedených v rámci jakékoliv jiné transakce. Jinými slovy, data zpracovávaná během transakce mohou být během zpracování viditelná pouze v rámci transakce, ve které jsou zpracovávána.
- *Stálost* – jakmile transakce dokončí svou činnost, musí být veškerá data trvale uložena do databáze – chyba hardwaru nebo softwaru nesmí mít na tuto skutečnost vliv.

Veškeré úpravy databáze jsou prováděny prostřednictvím uložených procedur (stored procedures). Podrobnosti je možné nalézt v [2], [3] nebo [14]. Hlavní důvody pro jejich užití byly:

- centralizovaná správa kódu užívaného pro úpravu databáze;
- možnost využití logických a procedurálních podmínek[26] při stavbě SQL příkazu;
- rychlejší spuštění díky předgenerovanému plánu spuštění (execution plan);

[26] Podmínky typu IF ELSE nebo SLECECT CASE. Díky přenesení části programové logiky ze strany web serveru na stranu SQL serveru je možné výrazně snížit počet otevření a zavření připojení k databázi, čímž je zvýšena přístupnost databáze a sníženo vytížení spojení mezi web a SQL serverem.

- možnost hromadné aktualizace nebo zápisu dat díky podpoře uživatelských tabulkových proměnných.

3.4.3.3 Čtení z databáze

Pro získávání dat z databáze bylo využito dynamických SQL příkazů spouštěných přímo z web serveru. V případech, kdy bylo třeba pro tvorbu příkazu využít logických funkcí nebo zpracovávat výsledky více samostatných dotazů, byly využity uživatelské funkce[27] (user-defined functions). Hlavní rozdíl mezi uživatelskou funkcí a uloženou procedurou je v tom, že uložená funkce umožňuje vrátit nejen skalární hodnotu, ale i tabulku, což umožňuje její využití uvnitř jiného SQL dotazu. Uživatelské funkce byly použity také jako součást uložených procedur.

3.4.3.4 SQL injection

SQL injection útok, známý též jako SQL insertion útok, je podskupina code insertion útoků, během kterých se útočník snaží zadáním falešných dat získat data, ke kterým nemá přístup, nebo poškodit databázi. Nejčastěji útok probíhá při tvorbě dynamických SQL dotazů, kde se útočník snaží s využitím escape literálů změnit původní smysl SQL dotazu. Jako obrana proti těmto útokům jsou použity dvě metody:

- validace vstupních řetězců;
- použití parametrických příkazů při tvorbě SQL dotazu.

Stěžejní roli pro bezpečnost hraje právě využití parametrických příkazů, kdy je místo SQL dotazu s přímo zakomponovaným vstupem použit tzv. placeholder. Díky tomuto přístupu jsou parametry a samotný SQL dotaz zpracovávány samostatně a nemůže dojít k tvorbě alternativního příkazu užitím upraveného parametru.

Typický příklad použití parametrů při dynamické tvorbě SQL příkazu může vypadat následovně:

```
string select = "SELECT * FROM dbo.Users WHERE  UserName = @UserName";
SqlCommand cmd = new SqlCommand(select, new SqlConnection(conString));
cmd.Parameters.AddWithValue("@UserName ", tbUserName.Text);
```

Parametry jsou využity i v uložených procedurách. Na rozdíl od dynamických SQL příkazů je třeba u parametru uvést, jakého je typu a zda se jedná o parametr vstupní,

[27] Uživatelské funkce fungují obdobně jako systémové funkce používaných v T-SQL. Např. SELECT GETDATE().

výstupní či vstupně výstupní. Uvedené hodnoty se musí shodovat s parametry uložené procedury. Samotné zpracování parametrů uvnitř procedury probíhá stejně jako v případě dynamických SQL příkazů.

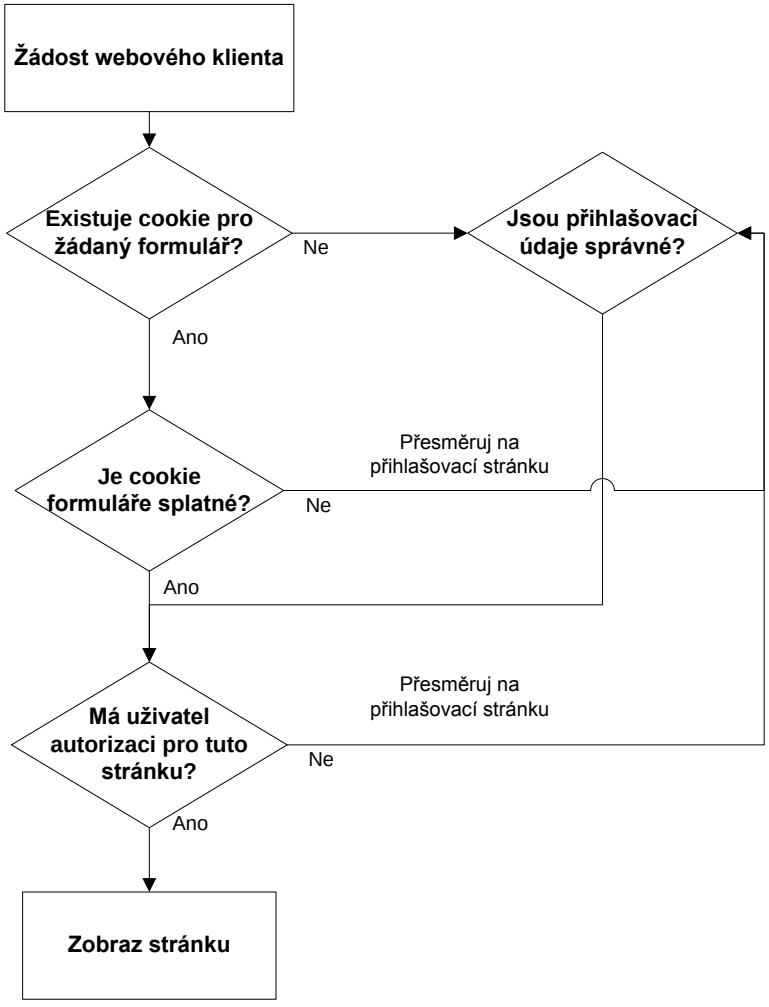
Typický příklad využití parametrů v uložené proceduře je následující:

```
SqlConnection connection = new SqlConnection(connectionString);
SqlCommand command = new SqlCommand("CreateUser", connection);
SqlParameter usrname;
usrname = new SqlParameter("@UserName", SqlDbType.NVarChar);
usrname.Direction = ParameterDirection.Input;
command.Parameters.Add(tbUserName.Text);
...
```

4 Funkční struktura

4.1 Autentizace a autorizace

Autentizace a autorizace jsou postupy, které mají zajistit, že pouze oprávněný uživatel dostane přístup k informacím a funkcím systému. Diagram znázorňující použité postupy je zobrazen na Obr. 2. Jednotlivé operace budou podrobněji popsány v následujících kapitolách.



Obr. 2 Princip zabezpečení přístupu k aplikaci

4.1.1 Autentizace

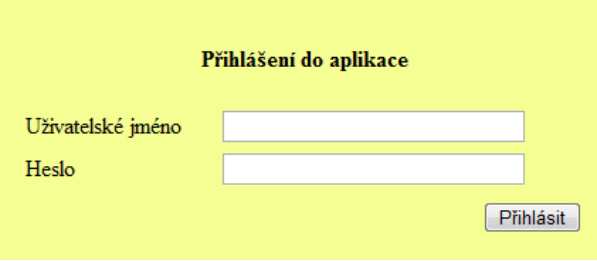
Autentizace je proces ověření skutečné identity uživatele, který se pokouší vstoupit do systému. Toto prověření se provádí prostřednictvím uživatelského jména[28] a hesla, které

[28] Uživatelské jméno musí být unikátní v rámci celého systému.

se zadávají na přihlašovací stránce. Tato stránka je také jediná dostupná pro nepřihlášené uživatele. V rámci aplikace se pro autentizaci užívá proces zvaný „forms authentization“. Tento proces, včetně jeho zabezpečení, je implementován přímo v .NET frameworku.

Forms authentization spočívá v tvorbě tzv. lístku v okamžiku, kdy se uživatel úspěšně přihlásí do systému. Tento lístek je poté udržován[29] po celou dobu, kdy je uživatel přihlášen a slouží k jeho identifikaci v rámci aplikace. Lístek zůstává platný po 30 minut od poslední komunikace klienta se serverem. Délku platnosti je možné změnit v web.config úpravou nastaveni vlastnosti „timeout“.

```
<authentication mode="Forms">
    <forms loginUrl="Login.aspx"
    protection="All"
    timeout="30"
    name=".ASPXAUTH"
    path="/"
    requireSSL="false"
    slidingExpiration="true"
    defaultUrl="Default.aspx"
    cookieless="AutoDetect"
    enableCrossAppRedirects="false" />
</authentication>
```



Obr. 3 Přihlášení do aplikace

4.1.2 Autorizace

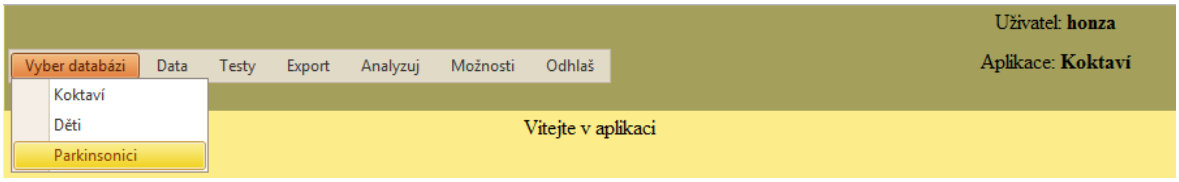
Autorizace je proces, při kterém dochází k přidělování práv autentizovanému uživateli. Tento proces probíhá ihned po přihlášení a následně při každém načtení nové stránky. Pokud se uživatel snaží získat přístup k obsahu, ke kterému nemá práva, např. přímým zadáváním url, je automaticky odhlášen a přesměrován na přihlašovací stránku. Podrobnosti o struktuře a fungování práv jsou popsány v kapitole 4.8.

Při procesu autorizace je také tvořeno dynamické menu, které slouží uživateli jako rozcestník (viz Obr. 4). V prvním kroku je zjištěno, ke kterým aplikacím uživatel má práva. V druhém kroku je uživatel přihlášen k aplikaci. Má-li uživatel práva k více

[29] Jsou-li na prohlížeči povoleny cookies, je lístek udržován v nich. V opačném případě je udržován pomocí dotazových textů.

aplikacím, je přihlášen k první, která je v databázi nalezena. Následně je podle práv v aplikaci, ke které je přihlášen, sestaveno menu, které umožní navigovat na všechna přístupná místa aktuální aplikace.

V menu „Vyber databázi“ je přístupný seznam všech aplikaci, které jsou pro přihlášeného uživatele dostupné. V tomto menu se mezi nimi může uživatel přepínat. Při přepnutí je znovu vygenerováno menu odpovídající právům uživatele v aplikaci, do které se přepnul. Informace o přihlášeném uživateli a aplikaci, ve které se právě nachází, lze nalézt v pravém horním rohu menu.



Obr. 4 Menu aplikace

4.2 Adresářová struktura

Aplikace používá pro ukládání všech dat v rámci všech aplikací jasně definovanou adresářovou strukturu. Nastavení jejích pevných prvků se provádí prostřednictvím konfiguračního souboru web.config před prvním spuštěním[30]. Část popisující adresářové rozložení vypadá následovně:

```
<FilePath>
  <add key="RootPath" value="D:\Files" />
  <add key="ApplicationDirectory" value="Application" />
  <add key="PatientsDirectory" value="Patients" />
  <add key="EquipmentDirectory" value="Equipment" />
  <add key="UsersDirectory" value="Users" />
  <add key="ControlPatientDirectory" value="ControlPatients" />
</FilePath>
```

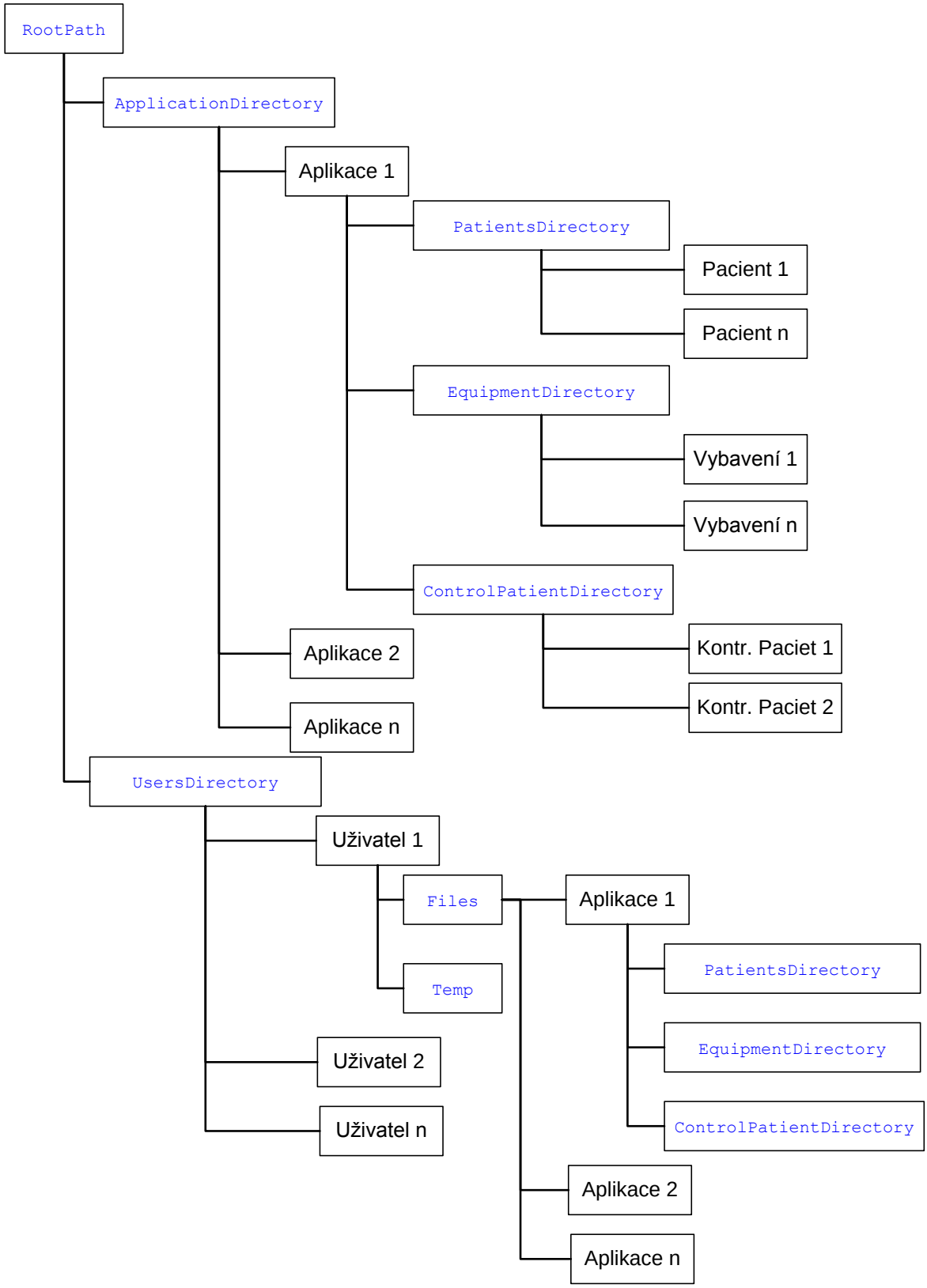
Skutečné rozmístění na disku poté probíhá podle schématu zobrazeného na Obr. 5. Modře vyznačená textová pole budou nahrazena hodnotami z konfiguračního souboru (Files a Temp jsou neměnná), černě vyznačená pole budou vyplněna názvy příslušných prvků:

- Aplikace – název konkrétní aplikace;
- Uživatel – uživatelské jméno uživatele;

[30] Je-li třeba změnit adresářovou strukturu, je nutné zapsat novou strukturu do web.config a zkopírovat existující data podle nově definované struktury. (při provedení pouze jednoho bodu systém ztratí informace o fyzickém umístění uložených soborů!)

- Pacient - ID pacienta;
- Kontrolní paciet - název kontrolního pacienta;
- Vybavení – název vybavení.

U veškerých hodnot užívaných pro tvorbu adresářů je z tohoto důvodu kontrolována jejich jedinečnost v rámci aplikace. Uživatelské jméno a název aplikace musí být unikátní globálně.



Obr. 5 Adresářová struktura

4.3 Správa uživatelů

Seznam všech uživatelů ve webové aplikaci, jejich vlastností a práv je zcela nezávislý na konkrétní aplikaci. Přidávat nové uživatele a spravovat existující mohou pouze uživatelé s právem „Spravovat uživatele“.

Odkaz pro správu uživatelů je umístěn v menu „Možnosti“, submenu „Uživatelé“. Na odkazované stránce je možné přidávat nové uživatele i aktualizovat nastavení existujících uživatelů, stejně jako informace o nich.

4.3.1 Tvorba nových uživatelů

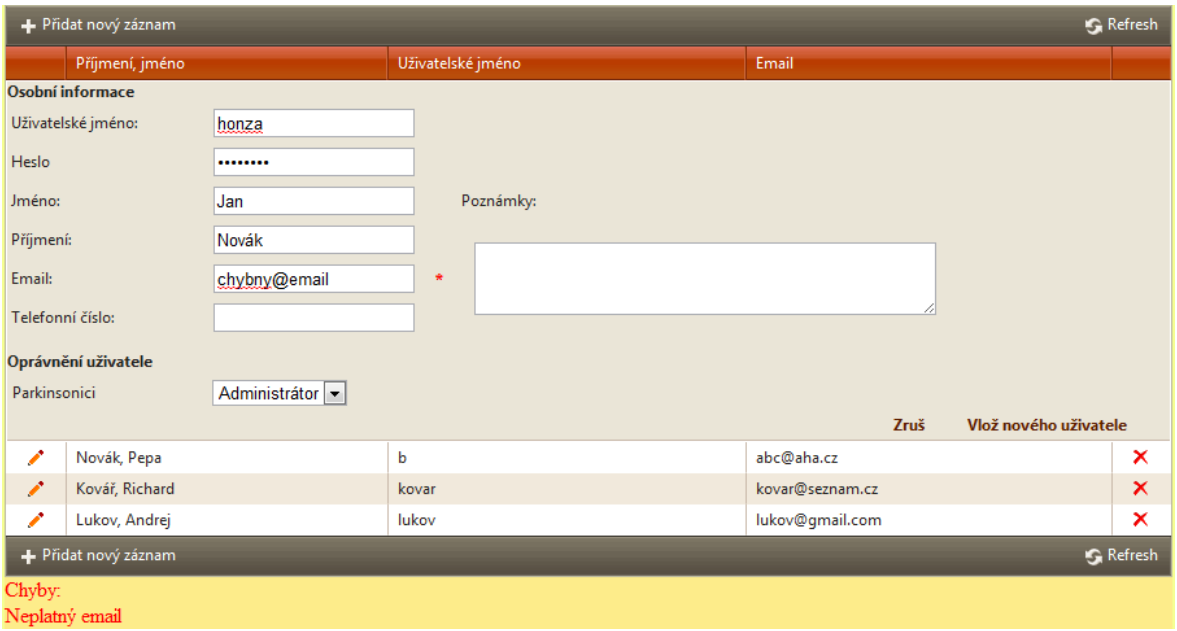
Nový uživatel se přidává přes webový formulář zobrazený na Obr. 6 kliknutím na symbol „plus“ a vyplněním formuláře se základními informacemi o uživateli.

- Uživatelské jméno – může obsahovat pouze alfanumerické znaky nebo „_“ a „-“
- Heslo
- Jméno
- Příjmení
- Kontaktní email – musí splňovat syntax pro email ve formě {text}@{text}.{text}

Další nepovinnou položkou je telefonní číslo a role uživatele v aktuální aplikaci.

Po odeslání vyplněného formuláře probíhá na straně serveru kontrola, zda uživatel se shodným uživatelským jménem již neexistuje. Proběhne-li kontrola úspěšně, je uživatel zapsán do databáze. Po zapsání uživatele do databáze se automaticky na serveru vytváří uživatelova složka dle struktury zobrazené na Obr. 5.

Nově vytvořený uživatel nemá žádná práva. Práva může uživatel získat buď přímým zadáním pro konkrétního uživatele, nebo přiřazením role, která již má práva nadefinována. Více o struktuře práv v kapitole 4.8.



Obr. 6 Tvorba nového uživatele

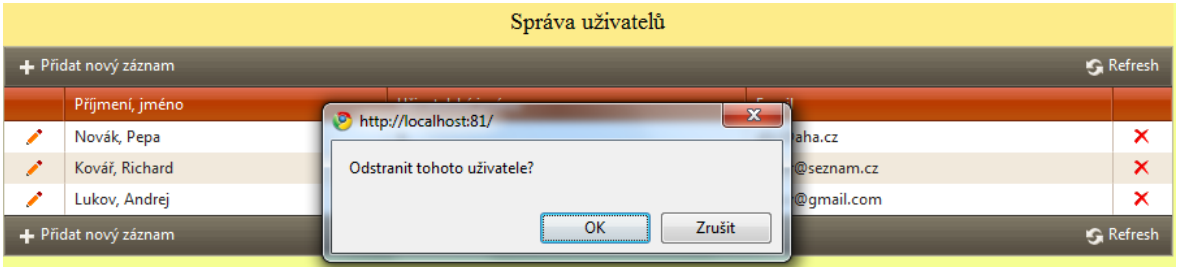
4.3.2 Aktualizace existujících uživatelů

Aktualizace existujícího uživatele probíhá kliknutím na editační tlačítko „tužky“. Následně zobrazený formulář je shodný s formulářem pro vytvoření nového uživatele; jedinou odlišností oproti tvorbě nového uživatele je skutečnost, že formulář je již vyplněn informacemi editovaného uživatele, pouze heslo zůstává nevyplněné. Po zadání aktualizovaných informací se zadané hodnoty (pokud projdou všemi kontrolami) zapíší do databáze. Není-li třeba aktualizovat heslo, stačí ponechat pole prázdné.

Dojde-li ke změně uživatelského jména, přejmenuje se odpovídajícím způsobem i uživatelský adresář.

4.3.3 Mazání uživatelů

Pro smazání uživatele stačí zmáčknout znak „křížku“ v pravé části tabulky. Po zmáčknutí se objeví okno žádající potvrzení daného úkonu (viz. Obr. 7). Pro zastavení procesu mazání slouží tlačítko „Zrušit“, pro potvrzení tlačítko „OK“. Smazáním dojde k odebrání uživatele ze všech rolí a smazání všech souborů uživatel, které nebyly označeny jako veřejné. Více o typech souborů v kapitole 4.5.



Obr. 7 Mazání uživatele

4.4 Správa aplikací

Spravovat aplikace může každý uživatel s právem „Přidat aplikace“. Odkaz na správu aplikací je umístěn v menu „Možnosti“, submenu „Aplikace“.

4.4.1 Tvorba nové aplikace

Tvorba aplikace z uživatelského hlediska spočívá pouze v kliknutí na tlačítko „Přidat novou aplikaci“ a ve vyplnění jména aplikace v následném formuláři. Jméno se nesmí shodovat s žádnou existující aplikací. Je-li toto pravidlo splněno, vytvoří se nová aplikace se svým vlastním jmenným prostorem. Během tvorby aplikace se vytvoří aplikační složky dle struktury zobrazené na Obr. 5. Zároveň se uživateli, který aplikaci vytvořil, přidají veškerá práva k nově vzniklé aplikaci.

4.4.2 Správa aplikace

Po vzniku je aplikace pouze prázdná slupka, pouze s jedním uživatelem, zakladatelem. Po kliknutí na editační tlačítko „tužky“ se zpřístupní menu se třemi záložkami:

- Obecné,
- Práva,
- Role.

V záložce „Obecné“ je možné změnit název aplikace. Nový název se opět nesmí shodovat s žádnou již existující aplikaci. Po přejmenování aplikace dojde i přejmenování aplikačních adresářů.

V záložce „Role“ se nachází seznam rolí, které jsou dostupné v rámci editované aplikace. Přidávání, editování a odebírání probíhá obdobným způsobem jako u úpravy aplikace.

V záložce „Práva“ je možné měnit práva uživatelů v aplikaci. Záložka obsahuje dvě podzáložky:

- Uživatelé,
- Role.

V podzáložce „Uživatelé“ se nachází seznam všech uživatelů v rámci celé webové aplikace. Zde je možné přes symbol tužky upravit práva uživatele v rámci aplikace, která je momentálně editována. Stejným způsobem je možné upravit práva role. V rámci uživatelů i rolí je možné využít fulltextového vyhledávání. Stačí do textového pole v horní části tabulky (viz Obr. 8), zadat hledaný výraz a zmáčknout „enter“. Vyhledávání není závislé na velikosti písmen.

Obr. 8 Správa práv

4.4.3 Odstranění aplikace

Odstranění se provádí přes symbol křížku. Před odstraněním se objeví okno vyžadující potvrzení, obdobně jako u mazání uživatelů v kapitole 4.4.3. Odstraněním aplikace dojde ke smazání všech dat, která byla v rámci aplikace uložena!

4.4.4 Práva v aplikaci

Přístup k jednotlivým funkcím aplikace je podmíněn vlastnictvím příslušného práva. Přihlášenému uživateli jsou dostupné pouze ty nabídky, k nimž má dostatečné oprávnění.

Seznam práv:

1. Spravovat pacienty
2. Spravovat vybavení
3. Spravovat kontrolní pacienty
4. Přidat pacienty
5. Přidat kontrolní pacienty
6. Přidat vybavení
7. Přidat uživatele
8. Spravovat uživatele
9. Přidat a spravovat místa
10. Přidat role
11. Spravovat role
12. Přidat aplikace

13. Spravovat práva aplikací
14. Tvořit testy
15. Absolvovat testy

4.5 Správa (kontrolních) pacientů a vybavení

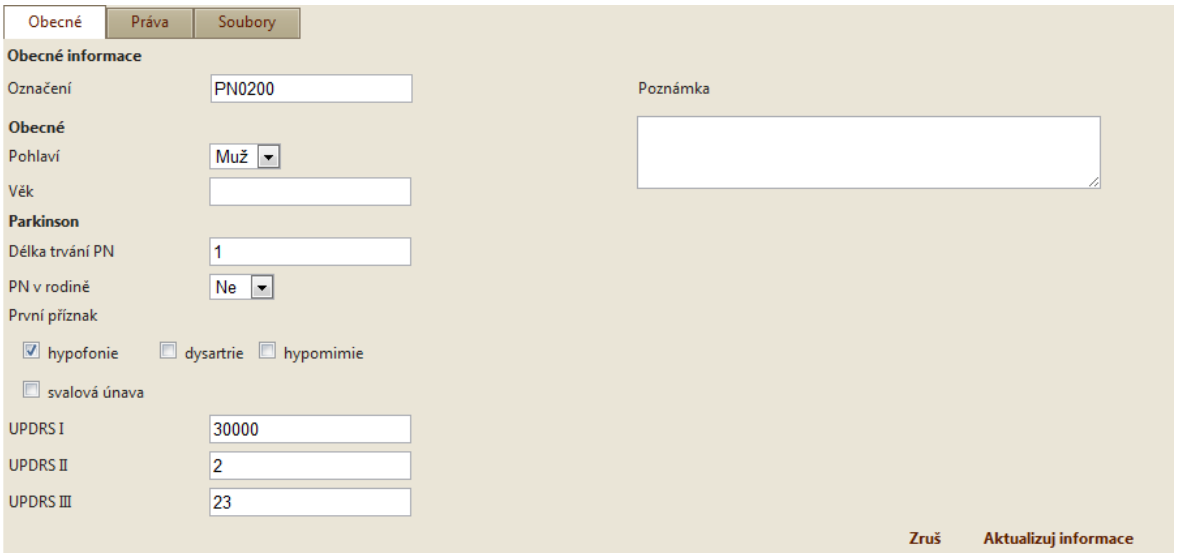
Přístup na stránku správy je přes menu „Data“, submenu „Pacienti“, respektive „Kontrolní pacienti“ nebo „Vybavení“. Protože správa pacientů, kontrolních pacientů a vybavení probíhá z uživatelského pohledu shodně, bude zde popsána pouze z pohledu pacienta.

4.5.1 Správa pacientů

Formulář pro tvorbu nového pacienta je přístupný přes tlačítko „Přidat nový záznam“ (viz Obr. 10). Při tvorbě nového pacienta je uživateli zpřístupněna pouze karta „Obecné“ (viz Obr. 9). Zde je třeba vyplnit informace o pacientovi. Aby mohl být pacient úspěšně vytvořen, je třeba, aby byly řádně vyplněny všechny vlastnosti s nastaveným parametrem povinná vlastnost (viz kapitola 4.9.4). Po vyplnění všech povinných vlastností je možné pacienta přidat zmáčknutím tlačítka „Vytvoř pacienta“. Po zapsání nového pacienta do databáze je pro pacienta vytvořena na straně serveru Vlastní složka, která bude sloužit pro ukládání veřejných souborů, které budou v budoucnu k pacientovi nahrány (viz kapitola 4.6). K nově vytvořenému pacientovi má veškerá práva pouze uživatel, který pacienta vytvořil.

Formulář pro aktualizaci pacienta je přístupný přes editační „tužku“, která je umístěna v tabulce pacientů napravo od označení pacienta (viz Obr. 10). Během aktualizace je možné upravit všechny vlastnosti. Jediná podmínka je, stejně jako u tvorby nového pacienta, aby všechny povinné vlastnosti byly řádně vyplněny v okamžiku aktualizace. Pokud je změněno označení pacienta, jsou zároveň přejmenovány adresáře, ve kterých jsou uloženy soubory nahrané k pacientovi, a to jak adresáře v rámci aplikace, tak v rámci uživatelů, pokud existují. V kartě „Práva“ lze měnit přístupová práva k pacientovi. V kartě „Soubory“ je možné nahrát soubory k pacientovi (viz kapitola 4.6).

Smazat pacienta je možné pomocí „křížku“ (viz Obr. 10). Při mazání pacienta dojde k odstranění všech jeho záznamů v databázi, všech souborů, které k němu byly nahrány, i výsledků testů, kterými byly hodnoceny pacientovy soubory.



Obr. 9 Tvorba nového pacienta

Obr. 10 Správa pacientů

4.5.2 Práva pacientů

Práva k pacientovi je možné nastavit na kartě „Práva“ (viz Obr. 10). Práva je možné nastavit separátně pro každého uživatele a pro role, které existují ve stejné aplikaci jako pacient. Je možné nastavit následujících pět práv:

- smazat pacienta,
- vidět veřejné soubory pacienta,
- přidat veřejné soubory,
- změnit práva,
- změnit vlastnosti.

4.6 Nahrávání souborů

Soubory se dají nahrát pouze k existujícím instancím pacientů, kontrolních pacientů nebo vybavení. Pro nahrání nového souboru k dané instanci musí mít přihlášený uživatel právo spravovat pacienta (resp. kontrolního pacienta nebo vybavení).

Formulář pro nahrání souborů se nachází v záložce „Soubory“, podzáložce „Nahraj“ (viz Obr. 11). Před nahráním je třeba vyplnit základní informace o souboru. Je-li třeba nahrát více souborů se stejnými vlastnostmi najednou, může uživatel přidat pole pro další soubory pomocí tlačítka „Add“. Jediné omezení pro nahrávané soubory je maximální souhrnná velikost 200MB[31]. Doba nahrávání nesmí přesáhnout 1 hodinu.

Obr. 11 Nahrávání souborů

Nahrané soubory jsou k dispozici na kartě „Zobraz“, kde lze zobrazit buď pouze soubory, které nahrál současný uživatel (karta „Moje“), nebo všechny soubory, ke kterým má práva (karta „Vše“).

4.6.1 Práva souboru

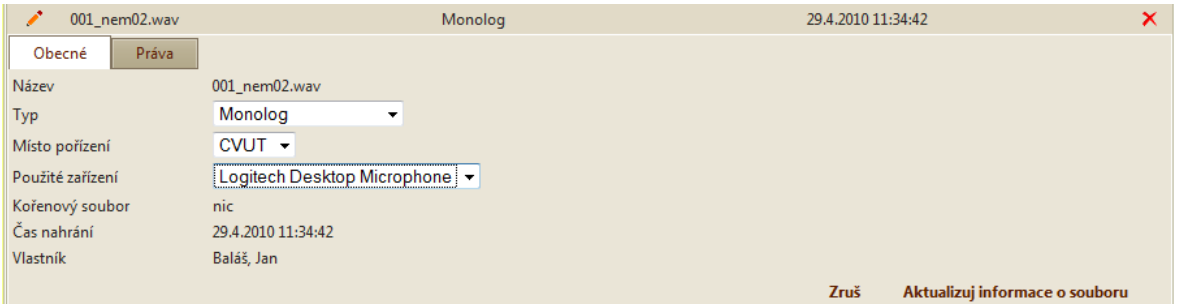
Při nahrávání nového souboru je možné pomocí přepínače zvolit, zda soubor bude přístupný všem uživatelům, kteří mají právo vidět veřejné soubory, nebo pouze uživateli, který soubor nahrál a dalším jím vybraným uživatelům. Je-li soubor nastaven jako veřejný, je na serveru uložen ve složce aplikace. Je-li soubor soukromý, je uložen ve složce uživatele, který ho nahrál – viz Obr. 5. Další vlastnosti a práva souboru lze změnit v kartě „Zobraz“ (viz Obr. 13), kde si pomocí „tužky“ zobrazí vlastnosti souboru (viz Obr. 14).

Ve vlastnostech souboru jsou dvě karty. V kartě „Obecné“ lze měnit data, která jsou o souboru uložena v databázi (typ, místo pořízení, použité zařízení) – viz Obr. 12. V kartě „Práva“ je možné nastavit práva k souboru. Opět je možné nastavit práva separátně pro jednotlivé uživatele a pro role.

[31] Velikost je možné zvýšit, maximálně však na 1GB, změnou nastavení ve web.config `<httpRuntime maxRequestLength="202400" executionTimeout="3600" />`

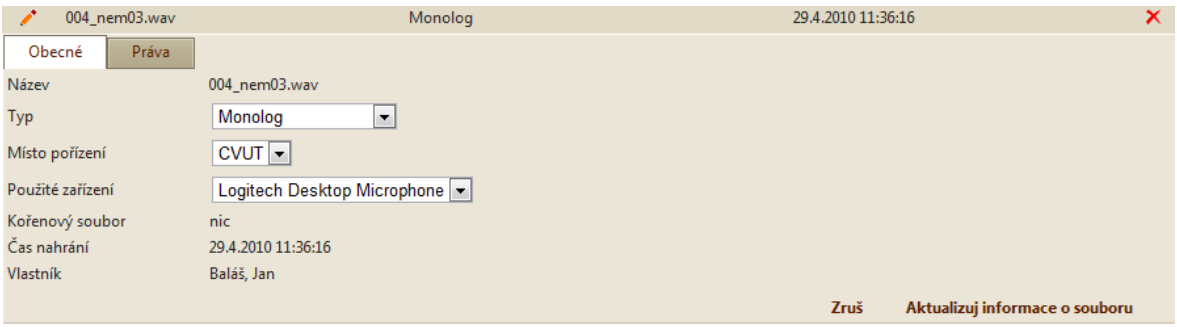
Práva, která lze pro soubor nastavit, jsou:

- Smazat – nutné pro smazání souboru;
- Číst – nutné pro export souboru, nebo přidání do vlastního testu;
- Měnit vlastnosti – nutné pro změnu dat v kartě „Obecné“;
- Měnit práva – nutné pro změnu práv k souboru.



Obr. 12 Obecné vlastnosti souboru

Obr. 13 Seznam souborů

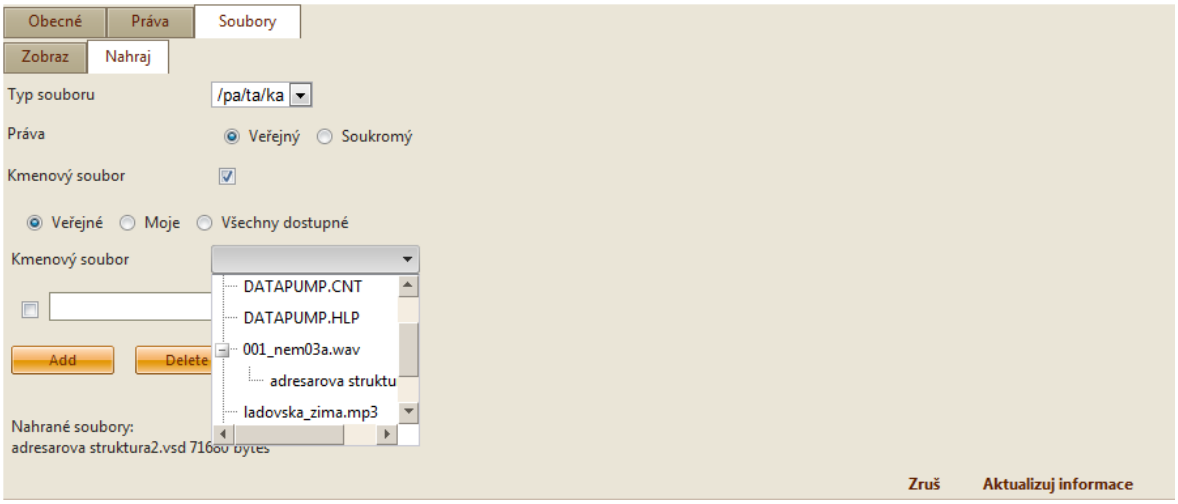


Obr. 14 Vlastnosti souboru

4.6.2 Vázanost souboru

Každý soubor může být navázaný na jiný soubor nahraný v minulosti. Typický případ takovéto provázanosti je např. audionahrávka a její textový přepis. Pro přiřazení nového souboru k již existujícímu souboru je třeba zaškrtnout „Kmenový soubor“. Následně se objeví rozbalovací seznam, ve kterém je seznam souborů, ke kterým se nový

soubor může přiřadit. Seznam zobrazených souborů se dá omezit pomocí přepínače umístěného nad seznamem. Jak je vidět z Obr. 15, propojení mezi soubory se může vrstvit libovolně do hloubky. Soubor, který je vázaný k jinému souboru, je na straně serveru uložen ve složce se shodným jménem, jako je nadřazený soubor, pouze „.“ je zaměněna za „_“. Tato adresářová provázanost je dodržena i při exportu – viz 4.7.



Obr. 15 Hierarchie nahrávání

4.7 Export

4.7.1 Testy

Exportovat výsledky testů je možné z menu „Export“ – „Testy“. Na stránce exportu jsou k dispozici dvě záložky. V záložce „Vlastnosti“ je možné vybrat, do jakého formátu mají být výsledky exportovány. V záložce „Export“ se vybírají[32] testy, jejichž výsledky mají být exportovány. Na výběr jsou pouze testy, u nichž má přihlášený uživatel právo úpravy testu. Samotný export se provede tlačítkem „Exportuj“, po jehož zmáčknutí se uživateli nabídne možnost stáhnout soubor „Testy“ s příponou dle zvoleného formátu. Exportovaný soubor obsahuje vždy data platná v době zmáčknutí tlačítka „Exportuj“. Formát výstupního souboru je zobrazen v Tab. 2. Jsou-li k exportu vybrány testy, které dosud nebyly nikým absolvovány, nejsou sloupce s hodnocením vyplněny, nicméně test je do výstupního dokumentu zařazen.

[32] Výběr je možné provést zaškrtnutím políček na levé straně tabulky nebo držením levého tlačítka myši tažením, jak je zvykem v desktopových aplikacích.

Podporované formáty pro export[33]:

- Microsoft Excel - .xls[34];
- Microsoft Word - .doc;
- Pdf.

Export testů			
Název testu: název prvního testu			
Testová otázka: testová otázka prvního testu			
Název souboru	Uživatel 1	Uživatel 2	Uživatel n
1. soubor v 1. testu	Hodnocení	Hodnocení	Hodnocení
n. soubor v 1. testu	Hodnocení	Hodnocení	Hodnocení
Název testu: název druhého testu			
Testová otázka: testová otázka druhého testu			
Název souboru	Uživatel 1	Uživatel 2	Uživatel n
1. soubor v 1. testu	Hodnocení	Hodnocení	Hodnocení
n. soubor v 1. testu	Hodnocení	Hodnocení	Hodnocení

Tab. 2 Formát exportu testů

4.7.2 Pacienti, kontrolní pacienti, vybavení

Exportovat pacienty, kontrolní pacienty a vybavení je možné z menu „Export“ – „Pacienty“ (resp. „Kontrolní pacienty“, „Vybavení“). Obdobně jako při exportu testů jsou na stránce dvě karty – „Export“ a „Vlastnosti“. V kartě vlastnosti je třeba vybrat, zda chce uživatel vyexportovat informace o pacientovi (kontrolním pacientovi, vybavení), nebo příslušné soubory.

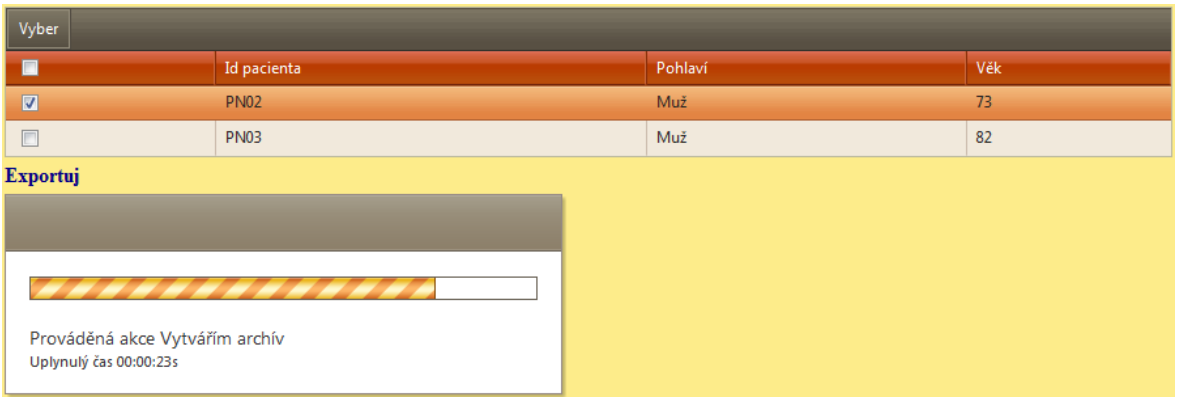
4.7.2.1 Export souborů

Aby bylo možné exportovat větší počet souborů najednou, jsou před exportem všechny soubory zabaleny do zip archívu. Pro tvorbu archívu je využita vlastní kompilace knihovny SharpZipLib. Výhodou této knihovny je možnost tvorby adresářové struktury

[33] Formát .csv nepodporuje složitější strukturu výstupního dokumentu.
[34] Při otevření exportovaného souboru s MS Excel 2007 objeví se varování, že soubor je v jiném formátu než určuje přípona. To je dáno tím, že pro tvorbu není použit nativní binární formát, ale zdrojový HTML. Nejedná se o chybu, pouze o větší přísnost ohledně MIMO typů v MS Office 2007. Podrobnosti o této problematice lze nalézt zde:
http://blogs.msdn.com/vsofficedeveloper/pages/Excel-2007-Extension-Warning.aspx
http://devblog.grinn.net/2008/06/file-you-are-trying-to-open-is-in.html

uvnitř zip archívu. Díky této vlastnosti jsou všechny soubory umístěny ve stejné adresářové struktuře jako na serveru – viz Obr. 5. Samotný archiv se tvoří v „Temp“ adresáři uživatele, který si jej vyžádal.

Tvorba zip archívu může být časově velmi náročná, v závislosti na celkové velikosti exportovaných souborů, jejich typu[35] a celkovém vytížení serveru, na kterém je aplikace spuštěná. Aby uživatel věděl, jak je jeho požadavek zpracováván, je během celé doby zpracování zobrazen „progress bar“[36]. Po dokončení archivace „progress bar“ zmizí a uživateli je nabídnut odkaz pro stažení.



Obr. 16 Tvorba archívu

4.7.2.2 Export informací

Export informací je obdobný exportu testů, pouze poskytuje navíc možnost exportu do souboru .csv[37]. Struktura výstupního souboru je zobrazena v Tab. 3. Pro zpřehlednění výběru prvků k exportu je možné využít podrobného filtrování. Filtrovat je možné přes všechny vlastnosti. Při tvorbě filtru je možné tvořit jakkoli strukturovaný logický výraz pomocí běžných logických prvků[38]. Pro stavbu filtrovacího výrazu je možné využít jak matematických operátorů pro číselné vlastnosti, tak fulltextového vyhledávaní pro vlastnosti reprezentované textem. Filtr se aplikuje stisknutím tlačítka „Vyber“. Ukázka tvorby filtru je vidět na Obr. 17.

Před exportem je třeba vybrat v seznamu „Vlastnosti k zobrazení“, jaké vlastnosti mají být exportovány. Při prvním zobrazení stránky pro export jsou zobrazeny pouze

[35] Archivace souborů, které již byly zkomprimovány, trvá výrazně déle.
[36] Knihovna SharpZipLib nenabízí možnost sledovat průběh tvorby archivu, což trvá nejdelší dobu, proto se může zdát, že se zpracování „zaseklo“ na 80%. Ve skutečnosti to znamená, že byla vytvořena struktura archivu včetně souborů a probíhá archivace.
[37] Comma-separated value – hodnoty oddělené čárkami.
[38] AND, OR, negace.

vlastnosti s nastavením „Zobrazovat vlastnost“ (viz kapitola 4.9). Po změně výběru v seznamu „Vlastnosti k zobrazení“ jsou zobrazovány pouze vlastnosti, které jsou zde zaškrtnuté – pouze ty jsou také exportovány – viz Obr. 17. Pouze zobrazené vlastnosti jsou vyexportovány do výstupního souboru.

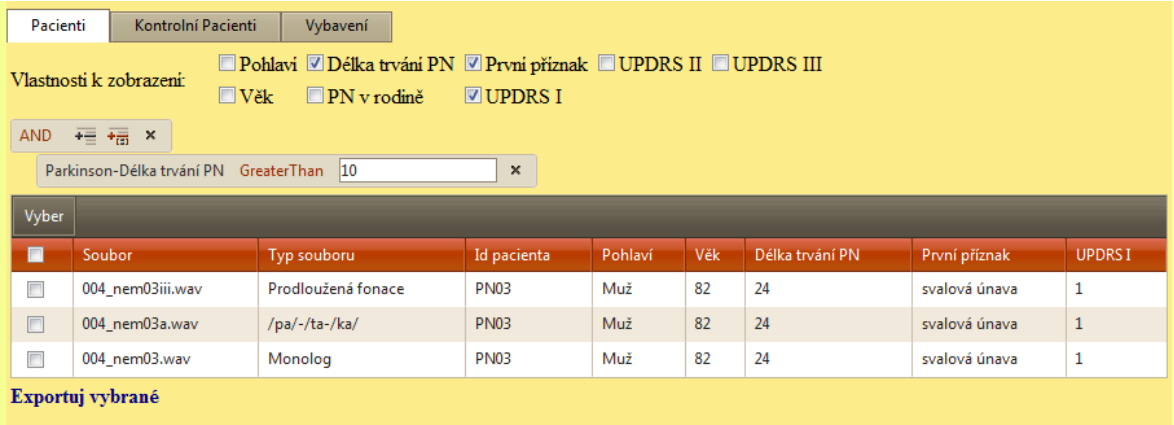
Id pacienta	Vlastnost 1	Vlastnost 2	Vlastnost n
Pacient 1	Hodnota	Hodnota	Hodnota
Pacient n	Hodnota	Hodnota	Hodnota

Tab. 3 Struktura souboru exportu pacientů

Obr. 17 Export pacientů

4.7.3 Soubory

Exportovat konkrétní soubory je možné z menu „Export“ – „Soubory“. Na této stránce jsou k dispozici tři karty – „Pacienti“, „Kontrolní pacienti“, „Vybavení“. Pod každou z karet je možné provést export odpovídajících souborů. V tabulce pro export jsou vždy zobrazeny všechny soubory, ke kterým má uživatel přístup (právo číst nebo měnit). Jelikož se zde může nacházet velké množství souborů, je dobré využívat možností filtrů, jak bylo popsáno v kapitole 4.7.2.



Obr. 18 Výběr souborů pro export

4.8 Práva

Přístup ke všem funkcím, informacím i datům je chráněn systémem práv tak, že uživatel má přístup pouze k oblasti, ve které má dostatečná práva. Práva může uživatel získat dvěma způsoby:

- přidáním práva konkrétnímu uživateli;
- přidáním práva roli, ve které je uživatel přiřazen.

Výsledné přístupové právo uživatele vznikne sjednocením individuálních práv uživatele a práv role, ve které je uživatel přiřazen.

Příklad:

Název práva	Práva uživatele	Práva role uživatele	Celkové právo uživatele
Tvořit testy	ANO	NE	ANO
Absolvovat testy	NE	ANO	ANO
Přidat pacienty	NE	NE	NE

Tab. 4 Princip práv

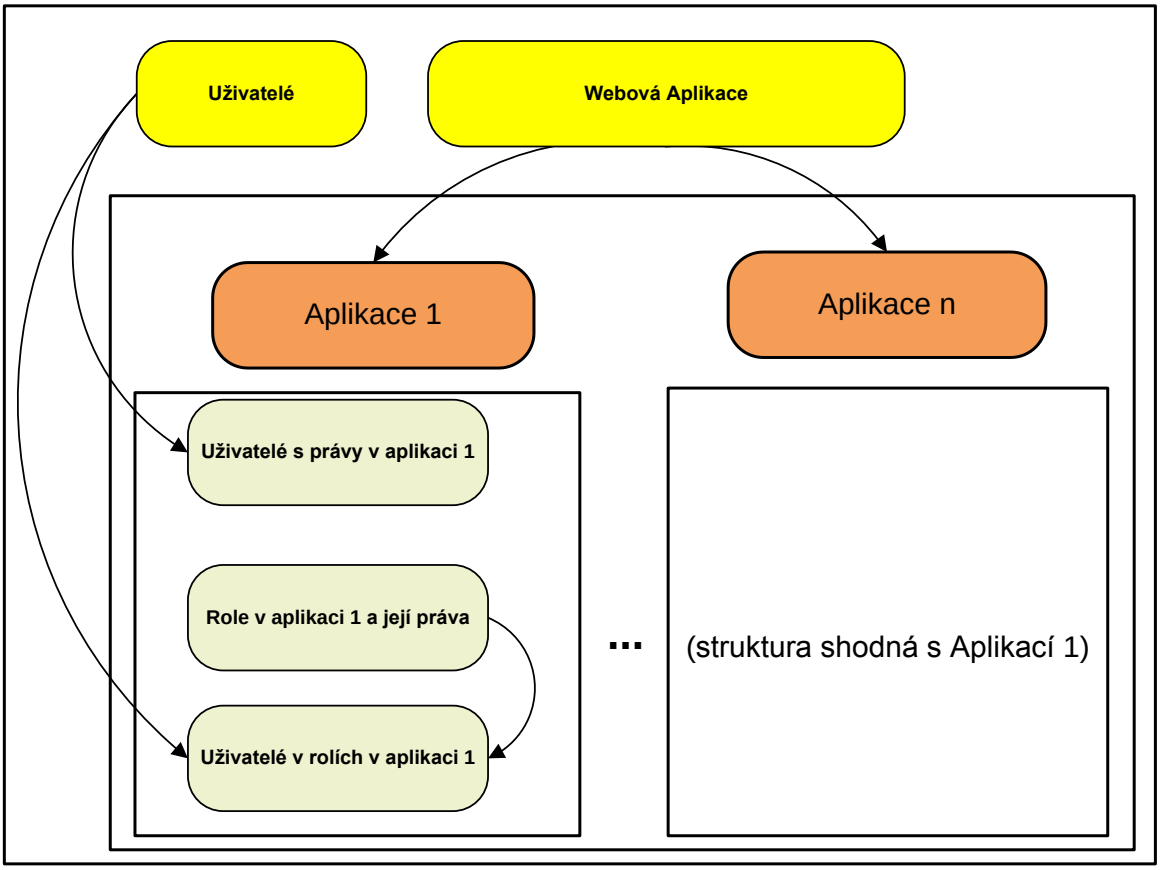
Veškerá práva a role jsou platné v rámci konkrétní aplikace[39], což je vidět ve schematickém přehledu na Obr. 19. Úpravou role jsou ihned[40] ovlivněni všichni uživatelé, kteří jsou k dané roli přiřazeni. Je-li role smazána, všichni přiřazení uživatelé mají do

[39] Role Student v aplikaci Parkinsonici tudíž může mít zcela odlišná práva od role Student v aplikaci Koktaví. Zároveň nemohou existovat dvě shodně pojmenované role v rámci jedné aplikace.
[40] Z pohledu uživatele se změna projeví při první žádosti o nová data ze serveru.

doby, než je jim přiřazena jiná role, pouze svá uživatelská práva a v nastavení uživatele jsou zobrazeni jako uživatelé bez přiřazené role.

Veškerá práva lze rozdělit do čtyř kategorií:

- práva v rámci aplikace (viz kapitola 4.4.4);
- práva k pacientům, kontrolním pacientům a vybavení (viz kapitola 4.5.2);
- práva k testům (viz kapitola 4.11.3);
- práva k souborům (viz kapitola 4.6.1).



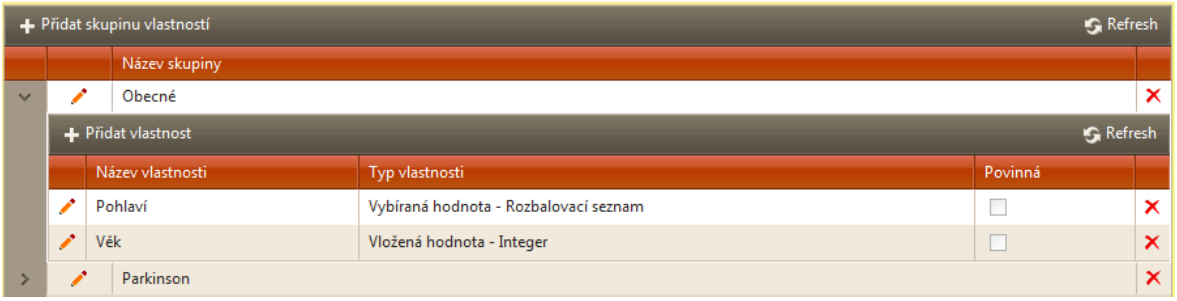
Obr. 19 Struktura práv v aplikaci

4.9 Vlastnosti (kontrolních) pacientů a vybavení

Základní funkcí informačního systému je uchovávat potřebné informace o pacientech. Definice slova potřebné se může lišit aplikace od aplikace (např. u Parkinsoniků je třeba uchovávat jiná data než u Koktavých) a může se vyvíjet i v čase. Z tohoto důvodu byl vytvořen systém, který umožňuje spravovat vlastnosti v rámci jednotlivých aplikací.

4.9.1 Tvorba skupin a vlastností

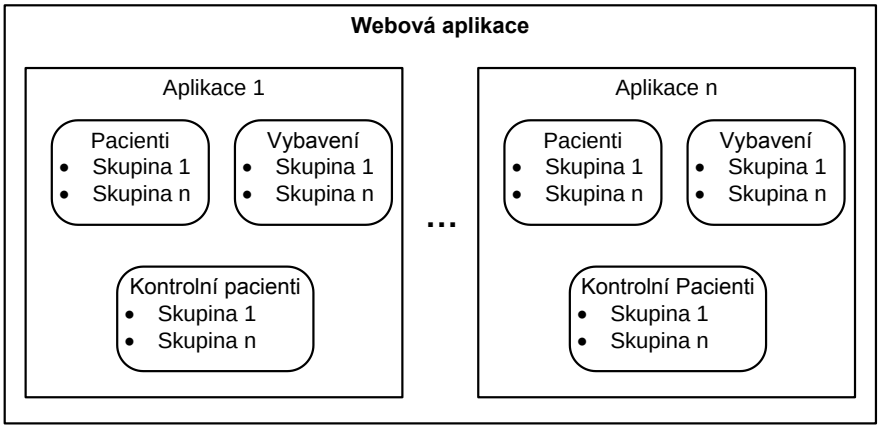
Skupina vlastností se přidává přes tlačítko „Přidat skupinu vlastností“ a edituje přes příslušnou editační „tužku“. Pro přidání vlastnosti do skupiny nebo editaci existující vlastnosti je třeba rozbalit skupinu přes boční šipku (viz Obr. 20) a použít vnořenou tabulku obdobně jako při tvorbě skupiny. Podrobnosti o pravidlech pro tvoření skupin a vlastností viz 4.9.2.



Obr. 20 Skupiny a vlastnosti

4.9.2 Jmenný prostor

Vlastnosti v rámci pacientů, kontrolních pacientů i vybavení (dále jen *objektů*) se pro větší přehlednost dělí do samostatných skupin. Skupina vlastností může obsahovat libovolné množství konkrétních vlastností, přičemž názvy vlastností v rámci jedné skupiny musí být unikátní. Unikátnost názvu vlastnosti není závislá na typu vlastnosti (např. není možné mít v jedné skupině vlastnost „věk“ typu celé číslo a vlastnost „věk“ typu reálné číslo). Objekt může obsahovat libovolné množství skupin. Názvy skupin v rámci objektu musí být unikátní (objekt je unikátní v rámci aplikace – různé aplikace tedy mohou obsahovat objekty stejného typu se shodně pojmenovanými skupinami). Výše popsaný způsob zapouzdření jmenných prostorů je graficky znázorněn na Obr. 21.



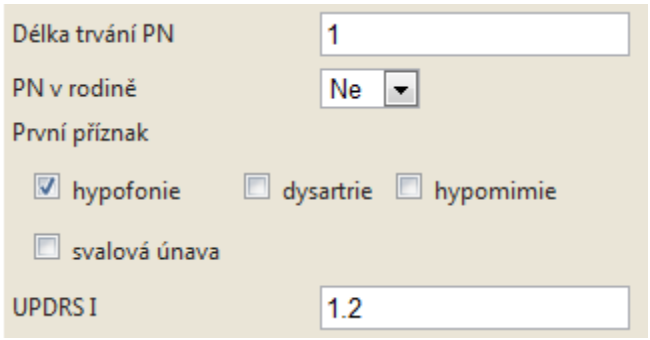
Obr. 21 Jmenný prostor vlastností

4.9.3 Typy vlastností

Při tvorbě vlastnosti je možné vybrat ze čtyř typů. Podle vybraného typu vlastnosti jsou posléze v zadávacích formulářích generovány ovládací prvky, přes které lze vlastnostem přiřadit konkrétní hodnoty. Ovládací prvky jednotlivých typů vlastností jsou vidět na Obr. 22. V závislosti na vybraném typu se též liší validační podmínky, které je nutné splnit pro aktualizaci nebo vložení nových hodnot vlastností do databáze.

Typy vlastností jsou:

- celé číslo,
- reálné číslo[41],
- vybíraná hodnota – rozbalovací seznam (vybírá se jedna hodnota ze seznamu),
- vybíraná hodnota – checkbox (vybírá se libovolné množství hodnot ze seznamu).



Obr. 22 Ovládací prvky různých typů vlastností

[41] Desetinná čárka je reprezentována „ . “ i „ , “

4.9.3.1 *Číselné vlastnosti*

Ovládací prvky pro oba typy číselných vlastností jsou stejná textová pole. Hlavní rozdíl spočívá v kontrolním filtru a v následném zpracování hodnoty z textového pole. Aby byla vlastnost zpracována, musí vstupní řetězec z textového pole splňovat následující regulární výraz:

- `ValidationExpression="^-?\d+"` pro celé číslo
- `ValidationExpression="^-?\d+[\.\,]?\d*"` pro číslo reálné.

Rozsah pro číselné vlastnosti je -1,79E +308 až 1,79E +308.

4.9.3.2 *Seznamové vlastnosti*

Vlastnost typu seznam je po vytvoření pouze prázdná schránka, kterou je třeba naplnit hodnotami, ze kterých je možné vybírat. Pro zadání těchto hodnot je třeba editovat danou vlastnost zmáčknutím editační „tužky“ příslušné vlastnosti. V následném formuláři je kromě běžných nastavení vlastnosti možnost spravovat položky seznamu. Položky lze přidávat a upravovat stejným způsobem jako jakoukoli jinou vlastnost. Dojde-li k přejmenování položky v seznamu, přejmenují se i hodnoty, které byly již dříve vyplněny v jednotlivých instancích objektu.

Obr. 23 Tvorba seznamové vlastnosti

4.9.4 Parametry vlastností

Pro každou vlastnost je možné nastavit rozšiřující nastavení

- povinná vlastnost;
- zobrazovat vlastnost.

Je-li vlastnost nastavena jako povinná, je k validačním pravidlům platným pro daný typ vlastnosti přidána povinnost vyplnit hodnotu pro tuto vlastnost. Je-li vlastnost ponechána nevyplněná, bude uživatel upozorněn na nutnost jejího vyplnění. Dokud nebude řádně vyplněna, nebude možné odeslat formulář zadávající nová data nebo aktualizující současná data v systému. Je-li vlastnost nastavena jako povinná, v okamžiku, kdy již existují instance příslušného objektu, vyžaduje vlastnost vyplnění pro každou novou instanci a při každé editaci existující instance.

Je-li vybráno nastavení „Zobrazovat vlastnost“, ukážou se tato vlastnost a její hodnoty při každém zobrazení seznamu instancí objektů, dokud nejsou uživatelem vybrány jiné vlastnosti k zobrazení – viz 4.7.2.2. Vlastnosti s tímto atributem jsou též primárně nabídnuty k exportu.

4.10 Analýza

4.10.1 Kruhový diagram

Kruhový graf je využíván pro zobrazení poměrného zastoupení jednotlivých prvků. Jak je vidět z Obr. 24, je použit pro zobrazení zastoupení jednotlivých hodnot dané vlastnosti. Vlastnosti je možné analyzovat pro vybrané:

- pacienty,
- kontrolní pacienty.

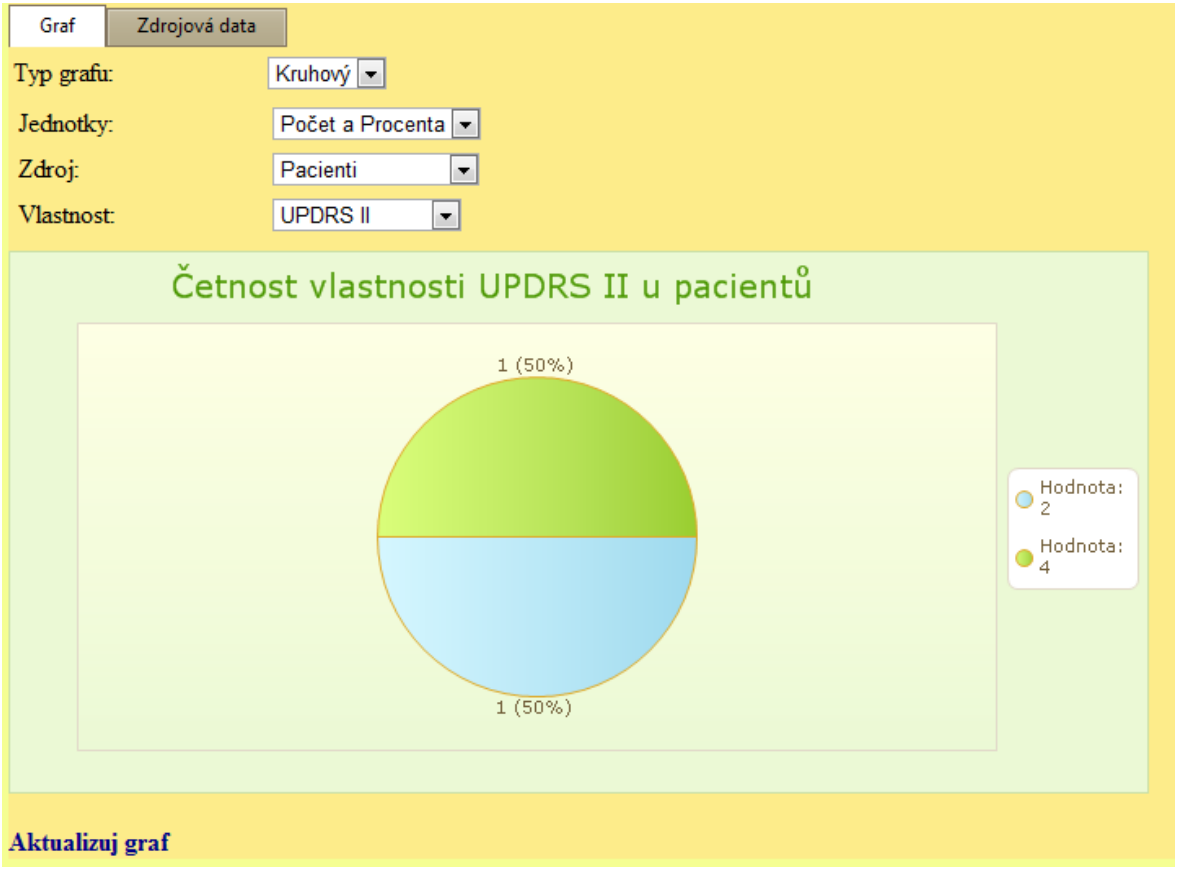
Zdrojové pacienty/kontrolní pacienty je možné vybrat v kartě „Zdrojová data“ pouhým označením. Při výběru je možné použít filtrování, jak již bylo popsáno dříve v kapitole 4.7.2.2. Pokud uživatel vybere novou skupinu pacientů k analýze, je třeba zmáčknout tlačítko „Aktualizuj graf“[42]. Je-li vybrána jiná vlastnost k analýze, nebo jiné jednotky, je graf aktualizován automaticky.

Pro zobrazení hodnot v grafu je možné vybrat mezi:

- absolutní počet výskytů,
- procentuelní zastoupení,
- kombinace počtu a procentuelního zastoupení (Obr. 24).

[42] Toto nepropojení je dáno tím, že aktualizace grafu při každé změně vybraného seznamu pacientů by působila velké množství postbacků a tím, z pohledu uživatele, velmi zpomalila aplikaci.

V pravé části grafu je zobrazena legenda se všemi hodnotami vlastností, které se v grafu nacházejí, včetně jejich barevného zastoupení.



Obr. 24 Kruhový diagram

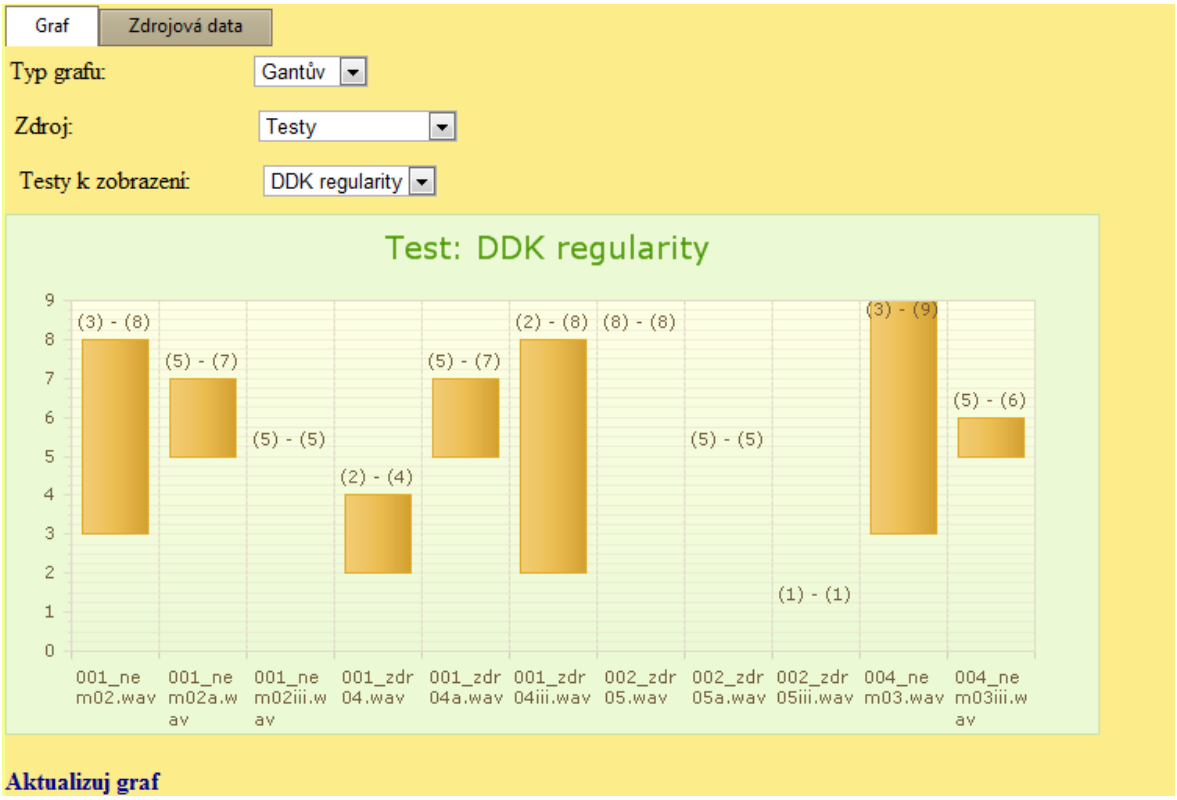
4.10.2 Sloupcový diagram

Sloupcové digramy jsou v analýze využity k zobrazení rozsahů, v jakých se nacházejí vybrané hodnoty. Lze je využít k analýze tří objektů:

- pacienti,
- kontrolní pacienti,
- testy.

Při analýze pacientů a kontrolních pacientů je třeba vybrat vlastnosti, které chce uživatel analyzovat (všechny vybrané se zobrazí v grafu). Výběr pacientů, pro které se analýza provádí, je shodný s výběrem zdrojových dat u kruhových diagramů. Jelikož je analyzován rozsah, je možné analyzovat pouze číselné vlastnosti.

Při analýze testů je třeba vybrat pouze test, který chce uživatel analyzovat (Obr. *25*). Na výběr jsou všechny testy, které má přihlášený uživatel právo editovat. Do analýzy jsou zahrnuta všechna aktuální hodnocení vybraného testu.



Obr. 25 Sloupcový diagram

4.11 Testy

Testy jsou určeny k hodnocení audionahrávek pacientů a kontrolních pacientů více uživateli nebo odborníky. Takto získaná data je dále možno samostatně exportovat (viz 4.7.1) nebo použít pro rychlou analýzu (viz 4.10).

4.11.1 Správa

Stránka pro správu všech dostupných testů je přístupná přes menu „Testy“, submenu „Správa testů“. Nový test se vytvoří zmáčknutím tlačítka „Přidat nový test“ a vyplněním názvu testu a testové otázky. Po vytvoření testu je možné v kartě „Práva“ nastavit přístupová práva k testu.

V kartě „Soubory“ lze test naplnit soubory, které mají být testovány. Do testu je možné přidat jakýkoliv audiosoubor[43] objektu pacient nebo kontrolní pacient. Přidat soubory je možné v kartách „Pacienti“ a „Kontrolní skupina“. Soubory, které již jsou v testu, se nacházejí v kartě „Soubory v testu“ (viz Obr. 26). Výběr, filtrování a přidávání souborů funguje na stejném principu jako export souborů (viz 4.7.2.1).

Odebrat soubory z testu je možné v kartě „Soubory v testu“ přes křížek. Je-li odebrán z testu soubor, který již byl absolvován, jsou spolu se souborem odebrána všechna hodnocení daného souboru.

Obr. 26 Správa souborů v testu

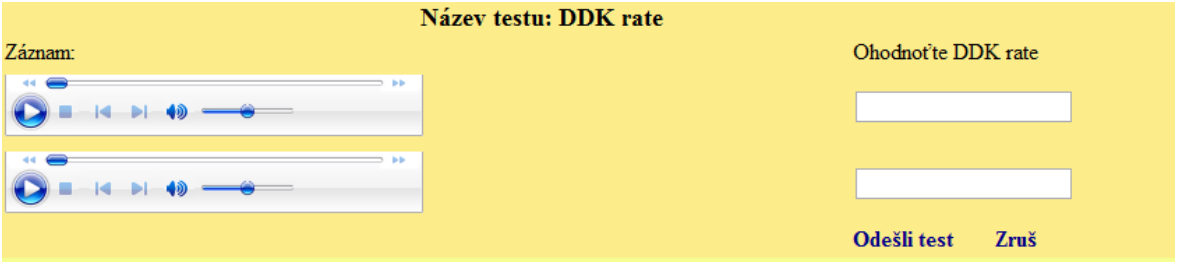
4.11.2 Absolvování

Všechny testy, které má uživatel právo absolvovat, jsou přístupné přes menu „Testy“, submenu „Absolvované testy“. Na této stránce je možné spravovat testy, které uživatel již absolvoval, nebo podstoupit zatím neabsolvované testy. Pro absolvování testu stačí zmáčknout tlačítko „Absolvuj test“ (viz Obr. 27). Následně je uživatel přesměrován na stránku, kde je pro každý testovaný soubor zobrazen samostatný přehrávač a kolonka pro vyplnění číselného hodnocení nahrávky (viz Obr. 28). Pro odeslání testu a uložení odpovědí do databáze slouží tlačítko „Odešli test“. Odeslání proběhne pouze tehdy, pokud jsou řádně vyplněna všechna textová pole odpovědí. Pro návrat na stránku výběru testů bez uložení výsledku testů slouží tlačítko „Zruš“. Je-li test úspěšně odeslán, přesune se ze složky „Spustitelné testy“ do složky „Absolvované testy“. Uživatel může test absolvovat

[43] Podmínkou správného fungování je, aby byl Windows Media Player schopen tento formát přehrát. Je-li dodán soubor, jehož typ není podporován, bude soubor v testu zobrazen, ale nepůjde přehrát.

pouze jednou. Chce-li stejný test absolvovat znovu, je třeba nejprve smazat výsledky předchozího absolvování.

Obr. 27 Testy k absolvování



Obr. 28 Absolvování testu

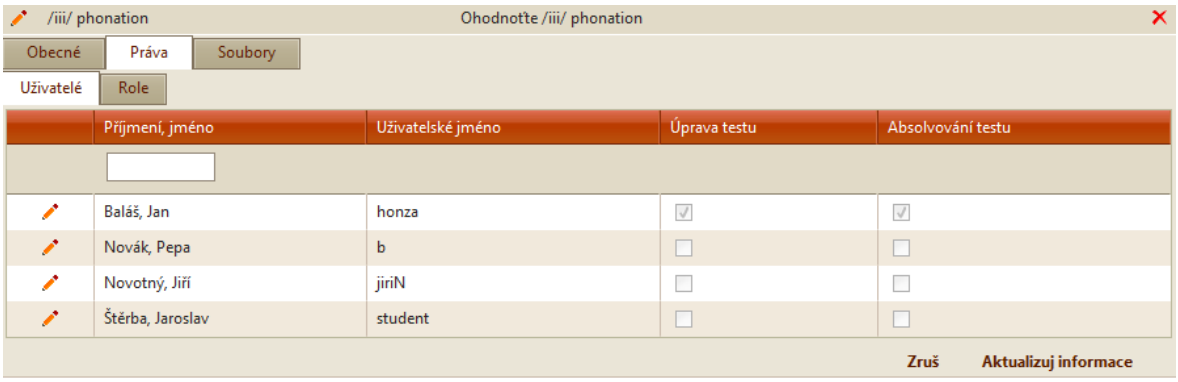
Obr. 29 Testy k absolvování

4.11.3 Práva

Práva testu je možné editovat v záložce „Práva“ viz Obr. 30. Editovat test a měnit jeho práva může pouze uživatel, který má právo úpravy testu, nebo je v roli, která toto právo má.

Pro práci s testem jsou dostupné pouze dva typy práv:

- právo absolvování testu;
- právo úpravy testu.



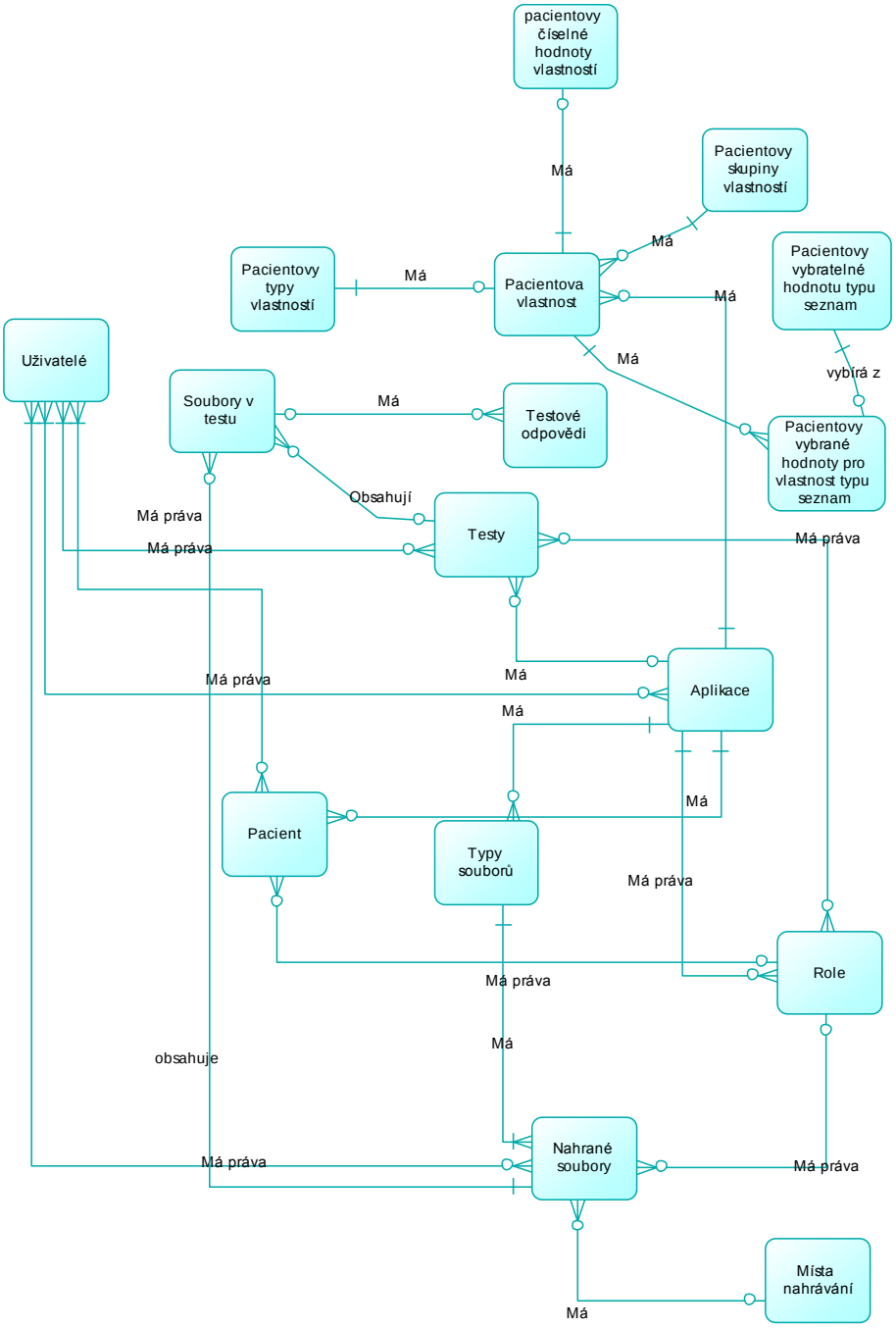
Obr. 30 Správa práv testů

Uživatel s právem úpravy testu má k dispozici všechna nastavení, která test umožňuje. Jmenovitě jsou to: správa práv, název testu, testová otázku a seznam souborů, které budou testovány.

Uživatel s právy pro absolvování testu má právo daný test absolvovat. Podrobnosti o absolvování testů viz 4.11.2.

5 Datová struktura

Zjednodušený konceptuální model zobrazený na Obr. 31 zobrazuje entity použité při tvorbě datového modelu a vztahy mezi nimi. Pro jednoduchost nejsou zobrazeny entity vybavení a kontrolních pacientů a entity na ně navázané, jelikož jsou shodné s entitami pacientů.



Obr. 31 Zjednodušený konceptuální datový model

5.1 Význam jednotlivých entit a vztahů mezi nimi

Aplikace: Představuje jednotlivé aplikace v rámci webové aplikace. Každá aplikace má své vlastní role, pacienty, kontrolní pacienty, vybavení, jejich vlastnosti, skupiny vlastností, soubory, místa nahrávání a testy.

Pacient: Reprezentuje kontejner, který obsahuje instance souborů a vlastností. Ke každému pacientovi může získat práva uživatel nebo role.

Uživatel: Účet, pod kterým je možné se přihlásit k aplikaci. Obsahuje základní informace o uživateli. Uživatel existuje nezávisle na ostatních entitách, pouze k nim může získat právo.

Nahrané soubory: Seznam všech souborů, které byly nahrány v rámci jednotlivých aplikací k pacientům, kontrolním pacientům a vybavení. Obsahuje základní informace o souboru, jako je jméno, vlastník (uživatel, který ho nahrál), čas nahrání, název aplikace, v jejímž rámci byl nahrán, a k jakému objektu patří (pacient, kontrolní pacient, vybavení), dále zda je soubor veřejný a na jaký soubor je vázaný. Ke každému souboru může uživatel nebo role získat práva. Každý soubor s nastaveným atributem pacient nebo kontrolní pacient může být přidán do testu.

Místa nahrávání: Seznam míst, kde byly soubory (nejčastěji nahrávky) pořízeny.

Pacientova vlastnost: Seznam všech vlastností, které mohou být pro pacienta vyplněny. Pro každou vlastnost obsahuje odkaz na typ vlastnosti, zda je nutné vlastnost vyplnit a zda je vlastnost defaultně zobrazována.

Pacientova skupina vlastností: Představuje seznam skupin, v rámci kterých jsou vlastnosti řazeny. Každá vlastnost musí být přiřazena právě v jedné skupině. Samotná skupina může existovat i bez vlastností.

Pacientovy možné hodnoty typu seznam: Seznam hodnot, ze kterých může uživatel vybírat u vlastnosti typu seznam (viz 4.9.3.2). Každé vlastnosti jsou přiřazeny hodnoty, kterých může nabývat, a každá taková hodnota je očíslována. Číslování hodnot se provádí pro každou vlastnost separátně (vlastnost A má hodnoty 1 až n, vlastnost B má hodnoty 1 až m).

Pacientovy vybrané hodnoty pro vlastnost typu seznam: Zde je seznam vybraných hodnot.

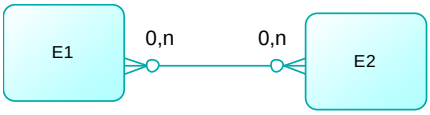
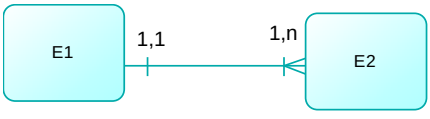
Pacientovy číselné hodnoty vlastností: V této entitě jsou číselným vlastnostem pacientů přiřazeny konkrétní hodnoty.

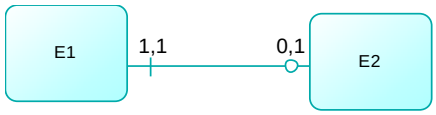
Pacientovy typy vlastností: Seznam typů vlastností, které mohou existovat (celé číslo, reálné číslo, seznam atd.). Podle typu jsou vlastnosti přiřazeny hodnoty z tabulky číselných vlastností, nebo seznamových vlastností.

Role: Objekt, kterému mohou být přiřazena práva stejně jako uživateli. Uživatelé, kterým je přiřazena role, nabývají navíc ke svým uživatelským právům všechna práva, která role má.

Testy: Seznam testů a otázek, na které se v testu odpovídá. Každému testu mohou být přiřazeny soubory, které spadají pod pacienty nebo kontrolní pacienty v rámci stejné aplikace jako samotný test.

Testové odpovědi: Odpovědi na testové otázky provedené uživateli. Každá odpověď je přiřazena konkrétnímu souboru v testu a uživateli, který odpovídal na testovou otázku. Uživatel může zadat pouze jednu odpověď ke každému souboru v testu.

Zobrazení	Kardinalita	Parcialita
	1:N	Záznam z E1 se může vázat k více záznamům z E2. Záznam z E2 se může vázat maximálně k jednomu záznamu z E1
	N:M	Záznam z E1 se může vázat k více záznamům z E2. Záznam z E2 se může vázat k více záznamům z E1
	1:N	Záznam z E1 se může vázat k více záznamům z E2. Záznam z E2 se může vázat k maximálně jednomu záznamu z E1
	1:N	Záznam z E1 se může vázat k více záznamům z E2. Záznam z E2 se musí vázat právě k jednomu záznamu z E1
	1:N	Záznam z E1 se může vázat k více záznamům z E2, minimálně však k jednomu. Záznam z E2 se musí vázat právě k jednomu záznamu z E1

	1:1	Záznam z E1 se může vázat maximálně k jednomu záznamu z E2. Záznam z E2 se musí vázat právě k jednomu záznamu z E1
	N:M	Záznam z E1 se může vázat k více záznamům z E2, minimálně však k jednomu. Záznam z E2 se může vázat k více záznamům z E1

Tab. 5 Použité značení

5.1.1 Fyzický datový model

Fyzický datový model byt vytvořen na základě konceptuálního datového modelu. V přiloženém diagramu jsou zobrazeny pouze tabulky a jejich provázanost na základě primárních a cizích klíčů. Další omezení v rámci tabulek je možné vyčíst z přiložených zakládacích skriptů. Jelikož datový model je příliš rozsáhlý na to, aby byl zobrazen v těle tohoto dokumentu, je dodán jako samostatná tištěná příloha.

6 Závěr

Diplomová práce se skládá ze tří hlavních částí, kterými jsou: (i) návrh a implementace datové struktury, (ii) tvorba poskytovatelů, kteří zajistí propojení databáze a uživatelského prostředí a (iii) tvorba samotného uživatelského prostředí. Výsledkem návrhu datové struktury je sada zakládacích skriptů určených pro MS SQL Server 2008. Vytvoření univerzální poskytovatelé jsou dostupní prostřednictvím sestavené knihovny MyLibrary.dll a mohou být použiti pro přístup k datové struktuře vytvořené v prvním bodě, a to prostřednictvím libovolného programovacího jazyka kompatibilního s platformou .NET.

Z původně navrhovaného systému s pevně danými vlastnostmi se během návrhu přešlo na vícemodulární pojetí, kdy systém samotný je pouze jakási nevyplněná šablona, kterou je třeba dotvořit dle potřeb konkrétní skupiny, která jej bude využívat. S vizí uživatelsky sestavitelného systému byl místo fixních vlastností vytvořen systém vlastností dynamických, které mohou být nastaveny skrz uživatelské prostředí, a to bez nutnosti zásahu programátora. Dynamicky tvořené vlastnosti tím umožní spravovat typy informací ukládané o objektech v průběhu času. Díky dynamickým vlastnostem je možné vytvořit samostatné aplikace s vlastními objekty a vlastními sledovanými vlastnostmi, a tím poskytovat pouze relevantní data uživatelům, kteří s daty dále zacházejí; to je pro data, s nimiž bude náš IS pracovat, velmi praktické – u různých pacientů bude možné sledovat různé parametry.

I přes důkladné testování je možné, že se při provozu objeví drobné chyby, které nebyly dosud odhaleny. Nejpravděpodobnější chybou, která může nastat, je jiné formátování ovládacích prvků, než bylo původně zamýšleno. To je dáno především tím, že různé prohlížeče používají odlišné způsoby vykreslování a není prakticky možné otestovat aplikaci na všech dostupných prohlížečích, jejich verzích a různých operačních systémech. Nicméně žádná taková chyba by neměla závažně omezit základní funkčnost IS jako celku.

Do budoucna by bylo vhodné rozšířit systém o zadávání časově závislých dat, díky kterému by bylo možné přidávat průběžně získávaná data o pacientech v závislosti na datu pořízení. Toto rozšíření by zároveň umožnilo zvýšit možnosti analýzy dat. Dále by bylo praktické vytvořit rozšíření, díky kterému by bylo možné automaticky importovat data

přímo z měřicích přístrojů včetně jejich interpretace a výpočtu základních určujících hodnot.

Tato práce měla za cíl vytvořit informační systém, který usnadní správu existujících a sběr nových dat. Díky důkladným testům vznikla stabilní aplikace splňující požadavky, které byly při zadání vytyčeny. Zda byl zvolený postup návrhu správný a výsledné uživatelské prostředí bude praktické, však budou muset posoudit až uživatelé během praktického provozu.

Použitá literatura

[1] *ASP Network*. [Online] [Citace: 12.5.2010]. Dostupný z WWW: <http://www.aspnet.cz/>. ISSN: 1801-9447.

[2] BRUST, A. J. a FORTE, S.. 2007. *Mistrovství v programování SQL Serveru 2005*. Brno : Computer Press, a.s., 2007. ISBN: 978-80-251-1607-4.

[3] DEWSON, R. *Beginning SQL Server 2008 for Developers: From Novice to Professional*. New York : Apress, 2008. ISBN: 978-1-59059-958-7.

[4] DUCKETT, J. *Beginning Web Programming with HTML, XHTML, and CSS*. Wiley Publishing, Inc., 2004. eISBN: 0-7645-7813-8.

[5] MACDONALD, M. *Beginning ASP.NET 3.5 in C# 2008: From Novice to Professional, Second Edition*. Wiley Publishing, Inc., 2007. ISBN: 978-1-59059-891-7.

[6] HERNANDEZ, M. J. a VIESCAS, J. L. 2004. *Myslíme v jazyku SQL*. Praha : GRADA, 2004. ISBN: 80-247-0899-X.

[7] *IC#Code*. [Online] [Citace: 12.5.2010]. Dostupný z WWW: <http://www.icsharpcode.net>.

[8] *Internet Information Services*. [Online] [Citace: 12.5.2010]. Microsoft Corporation, Dostupný z WWW: <http://www.iis.net/>.

[9] JENNINGS, R. *Professional ADO.NET 3.5 with LINQ and the Entity Framework*. Indianapolis : Wiley Publishing, Inc., 2009. ISBN: 978-0-470-18261-1.

[10] RATTZ, J. *Pro LINQ Language Integrated Query in C# 2008*. New York : Apress, 2008. ISBN: 978-1-59059-789-7.

[11] KELLENBERGER, K. *Beginning T-SQL 2008*. New York : Apress, 2009. ISBN: 978-1-4302-2461-7.

[12] LIBERTY, J. *Programming C#*. 2nd edition. O'Reilly Media Inc., 2002. ISBN: 0-596-00309-9.

[13] MCFARLAND, D. S. *CSS: The Missing Manual*. 2nd edition. Sebastopol : O'Reilly Media Inc., 2009. ISBN: 978-0-596-80244-8.

[14] *Microsoft Developer Network*. [Online] [Citace: 12.5.2010]. Microsoft Corporation Dostupný z WWW: < http://msdn.microsoft.com>.

[15] POKORNÝ, J., HALAŠKA, I. *Databázové systémy*. Praha : ČVUT, 1998. ISBN: 80-01-01724-9.

[16] RICE, N., RHODES, A. *Telerik RadControls for ASP.NET AJAX - A Step by Step Learning Guide*. Telerik Corp., 2008.

[17] SCHAEFER, K. a další. *Professional IIS 7.0*. Indianapolis : Wiley Publishing, Inc., 2008. ISBN: 978-0-470-09782-3.

[18] SPOFFORD, G. a další. *MDX Solutions. With Microsoft SQL Server Analysis Services 2005 and Hyperion Essbase*. 2nd edition. Indianapolis : Wiley Publishing, Inc., 2006. ISBN: 978-0-471-74808-3.

[19] *SQL Server*. [Online] [Citace: 12.5.2010]. Microsoft Corporation. Dostupný z WWW: < http://www.microsoft.com/sql/>.

[20] *Telerik*. [Online] [Citace: 12.5.2010] Dostupný z WWW: <http://www.telerik.com>

[21] *W3Schools*. [Online] [Citace: 12.5.2010] Dostupný z WWW: <http://www.w3schools.com>.

[22] *Wikipedia*. [Online] [Citace: 12.5.2010] Dostupný z WWW: <http://wikipedia.org/>.

[23] *World Wide Web Consortium*. [Online] [Citace: 12. 5. 2010] Dostupný z WWW: <http://www.w3.org>.

Přílohy

A. Fyzický datový model

Vzhledem k velikosti datového modelu je dodán jako samostatný dokument, přiložený na konci této práce.

B. Obsah přiloženého CD

Adresář		Obsah
Doc		Obsahuje elektronickou podobu tohoto textu a elektronickou podobu datového modelu.
Sources	Projects	Obsahuje zdrojové kódy využívané pro chod ASP.NET stránek.
	WebSites	Obsahuje zdrojové kódy ASP.NET stránek a zkompilované binární soubory zbytku programu.
	SQL	Obsahuje zakládací skripty pro databázi.