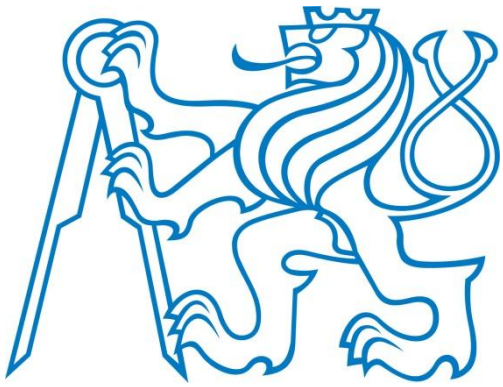
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra kybernetiky



BAKALÁŘSKÁ PRÁCE

Zobrazení dat uložených v Univerzálním úložišti

Pavel Kuchař

Vedoucí práce: Ing. Petr Novák, Ph.D.

duben 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Pavel K u c h a ř

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Zobrazení dat uložených v Univerzálním úložišti

Pokyny pro vypracování:

1. Seznamte se s tzv. „Univerzálním úložištěm" sloužícím pro snadné a formátované ukládání různých typů hodnot a průběhů společně s jejich základním popisem (název, typ, veličina, rozsah hodnot atd.). Jde o obdobu databáze, ale zahrnující pevnou strukturu pro ukládání dat, poznámek atd. a současně poskytující hledání záznamů, konverzi dat a další možnosti i bez aplikace, která data vytvořila a / nebo uložila.
2. Navrhněte, jaké další informace je vhodné k ukládaným datům do úložiště doplnit, aby bylo možné takto uložená data rovněž dostatečně vhodně zobrazovat / procházet i bez původní aplikace, která je pořídila a tedy do úložiště uložila.
3. Na tomto základě vytvořte aplikaci / komponentu pro zobrazení různých typů dat (hodnoty, body, seznamy, matice 2D/3D atd.) uložených v Univerzálním úložišti s využitím pouze zmíněných doplňkových informací.
4. Vytvořená aplikace / komponenta musí poskytovat schopnost uložení aktuálního zobrazení ve formě obrázku pro prezentace nebo články. Současně musí poskytovat možnost exportu některých typů dat do často využívaných souborů jako je TXT nebo CSV.

Seznam odborné literatury:

[1] Univerzální úložiště nejen pro lékařská data: https://nit.felk.cvut.cz/drupal/univerzalniuloziste
[2] Matthew MacDonald: Pro WPF 4.5 in C# Windows Presentation Foundation in .NET 4.5, Apress 2012
[3]Jack Xu, Practical: WPF Charts and Graphics, Apress 2009

Vedoucí bakalářské práce: Ing. Petr Novák, Ph.D.

Platnost zadání: do konce letního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 2. 10. 2014

Poděkování

Děkuji svému vedoucímu práce, panu Ing. Petru Novákovi, za cenné rady a připomínky, vstřícnost, ochotu a toleranci. Děkuji také své rodině a přátelům za velkorysou podporu.

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne …….. ……

Podpis autora práce

Abstract

This thesis deals with the displaying of data stored in the Universal storage, which creates user-friendly visual data form and data export to different formats suitable for machine processing. Universal storage is a form of the database used for the storing and subsequent work with the general data in poorly-defined structure. The work describes the structure and different data types, which are used by universal storage. The work designs and implements the universal way of presentation of selected data and advanced information. The solution is built on the advanced capabilities of WPF (Windows Presentation Foundation) for creating dynamic user interfaces.

Keywords

Universal storage, data visualisation, line chart, heat map, WPF

Abstrakt

Tato práce se zabývá zobrazováním dat uložených v Univerzálním úložišti uživatelsky přívětivou formou a jejich exportem do různých formátů vhodných pro strojové zpracování. Univerzální úložiště je forma databáze sloužící pro ukládání a následnou práci s obecnými daty špatně definované struktury. Práce popisuje strukturu úložiště a různé datové typy, se kterými úložiště pracuje. Navrhuje a implementuje univerzální způsob prezentace vybraných dat a jejich rozšiřujících informací. Řešení staví na pokročilých možnostech knihovny WPF (Windows Presentation Fondation) pro tvorbu dynamických uživatelských rozhraní.

Klíčová slova

Univerzální úložiště, vizualizace dat, čárový graf, teplotní mapa, WPF

Obsah

Kapitola 1
Úvod

Projekt Univerzálního úložiště (Novák, 2014), dále jen Úložiště, vznikl na základě potřeby ukládat data o špatně definované struktuře. Samotná data je třeba nejen ukládat, ale i procházet a opět získávat pro další zpracování („vyzvedávat je“ z Úložiště samotného). Touto problematikou se zabýval ve své bakalářské práci Martin Dubec (Dubec, 2009).

Před dalším zpracováním (strojovým anebo uživatelským) je nutno vyzvednutá data převést buď do uživatelsky přívětivé podoby (tabulka hodnot, graf, obrázek, 3D grafika …) nebo exportovat do některého z formátů běžných pro strojové zpracování (CSV, TXT, BMP …). Problematice uživatelsky přívětivého zobrazování a exportu dat vyzvednutých z Úložiště se věnuje tato práce.

Projekt Univerzálního úložiště původně vznikl převážně pro lékařské účely s cílem sjednotit ukládání různých typů dat z různých medicínských přístrojů společně s informacemi o těchto přístrojích, jednotlivých pacientech a jejich jednotlivých návštěvách lékaře. Toto předpokládané využití v lékařském prostředí také přirozeně generuje rámcovou strukturu dat, která bude využita i v této práci. Ukládaná data vyžadují, aby jednotlivé surové hodnoty byly naměřeny v rámci určitého měření, konkrétním měřicím přístrojem v průběhu dané návštěvy lékaře určitému pacientovi. Vzhledem k tomu, že samotné „Univerzální úložiště“ není závislé na této struktuře informací, tak ani navržené zobrazovací metody a celá architektura zobrazování informací uložených v Úložišti nesmí být závislá na této struktuře a musí být navržena s ohledem na všestrannost Úložiště samotného.

1.1 Cíle práce

- Seznámit se s projektem „Univerzálního úložiště“, které slouží pro ukládání různých typů naměřených veličin a souvisejících informací. Seznámit se s jeho stromovou architekturou ukládaní informací a s procesem získávání těchto dat z Úložiště.
- Analyzovat strukturu dat vyzvednutých komponentou vytvořenou v rámci předešlé práce na projektu, viz (Dubec, 2009).
- Analyzovat možnosti pro uživatelsky přívětivé zobrazení vyzvednutých dat s ohledem na univerzálnost dat samotných a navrhnout další položky, které by data měla obsahovat pro potřeby jejich zobrazování.
- Navrhnout různé možnosti zobrazení pro základní datové typy momentálně implementované v rámci projektu.
- Implementovat aplikaci/komponentu umožňující uživatelsky přívětivě zobrazit stávající základní datové typy a jejich export do různých formátů (PNG, CSV, BMP, TXT …).

1.2 Struktura a způsob zpracování práce

Nejprve bude v práci představeno „Univerzální úložiště“, analyzovány jeho momentální možnosti a určeny požadavky na jednotlivé komponenty popsané v rámci bodů zadání. V druhé polovině práce budou nastíněny koncepty využité k realizaci komponent vytvořených v rámci této práce a rozebrána jejich implementace.

1.3 Použité technologie

Celý projekt Universálního úložiště je postaven na platformě C# .NET Framework verze 4.0. Z tohoto důvodu i v implementační části této práce budou použity tyto technologie.

C# .NET Framework verze 4.0 je moderní platforma pro vývoj aplikací pro různá prostředí. Platformu vyvíjí společnost Microsoft jako nativní způsob vytváření moderních aplikací pro systémy Windows Vista a novější.

.NET Framework je rozsáhlá sada knihoven a technologií, které kromě základní knihovny jazyka C# umožňují například práci s databázemi, několik možností tvorby grafického uživatelského prostředí nebo vytváření webových aplikací.

Tato práce staví především na knihovně pro návrh a zobrazování uživatelského rozhraní Windows Presentation Foundation (WPF), která je nástupce starších Windows Forms. WPF slouží k vytváření propracovaných uživatelských rozhraní, která jsou vektorově renderovaná a tudíž nezávislá na rozlišení.

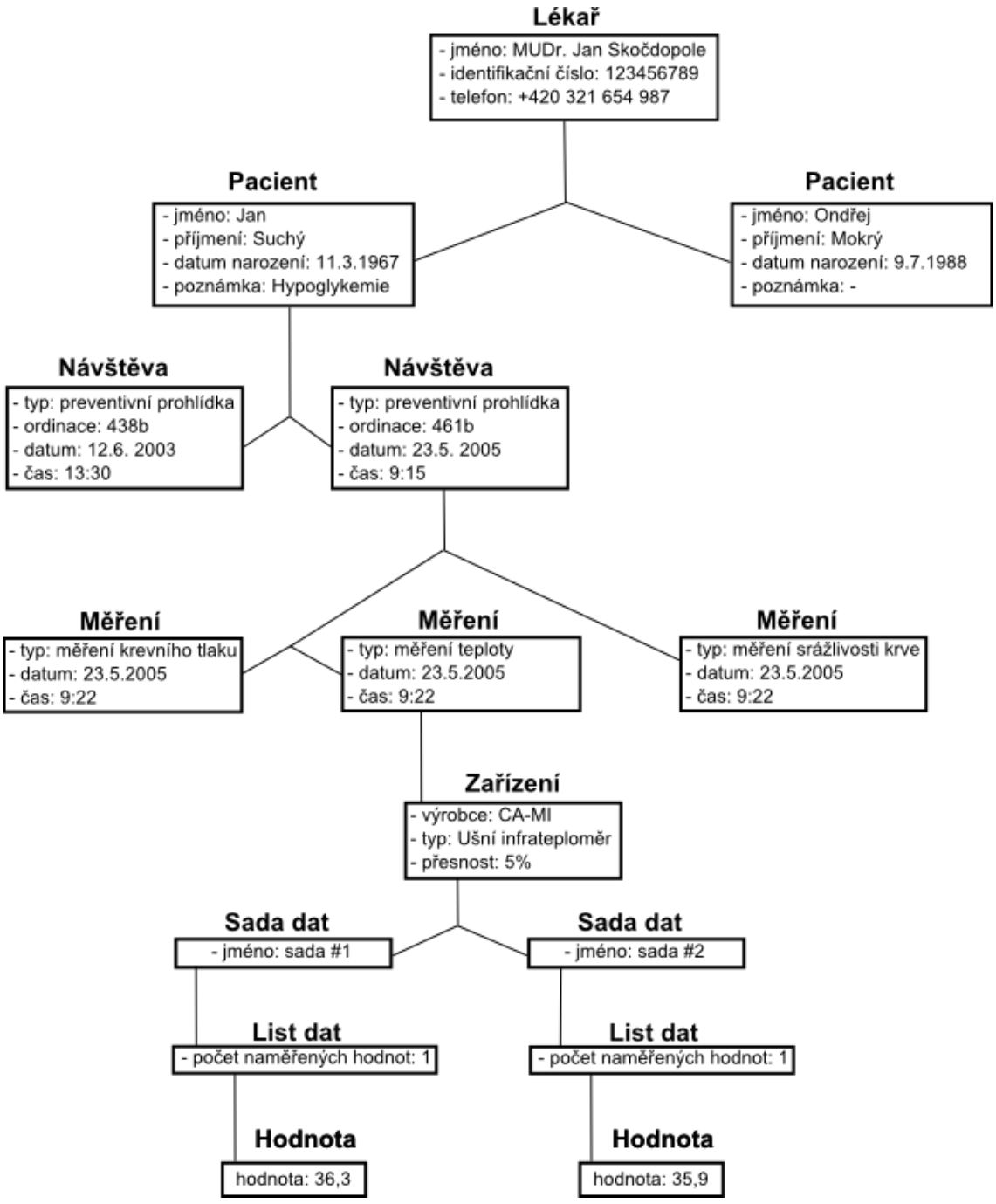
Přestože WPF umožňuje vytvářet uživatelská rozhraní imperativně jen pomocí C# kódu, záměrem celé technologie je umožnit vytváření komplexních uživatelských rozhraní pomocí deklarativního zápisu ve značkovacím jazyce XAML (Extensible Application Markup Language).

Kapitola 2
Univerzální úložiště

Univerzální úložiště původně vzniklo pro uchovávání lékařských dat (Dubec, 2009) a umožňuje data uchovávat v různých typech databází, v paměti počítače nebo jako souborový a adresářový strom v systému souborů. Verze, se kterou pracuje tato práce, byla proti původnímu návrhu rozšířena o další datové typy a je nastavena tak, aby data uchovávala pouze ve formě XML souborů vhodně uspořádaných v souborovém systému.

Hlavní výhodou Univerzálního úložiště je uchovávání všech dat v jednotném formátu bez ohledu na jejich původ. Data lze uchovávat v různých formách například jako databázi, souborový strom nebo v paměti počítače. Úložiště umožňuje jednotný přístup k těmto datům nezávisle na tom, v jaké formě se momentálně nacházejí.

Univerzální úložiště předpokládá, že ukládaná data lze rozčlenit do obecně hierarchické struktury a v této struktuře jsou také ukládána a následně vyzvedávána k dalšímu zpracování. Toto lze demonstrovat například na lékařských datech (viz Obr. 1). Každý konkrétní lékař má seznam pacientů, kteří ho navštívili. U každého pacienta lékař eviduje každou návštěvu, během které byla pacientovi provedena určitá měření. Každé toto měření mohlo být provedeno jedním nebo několika přístroji, které zaznamenaly jednu nebo několik sad dat, která každá obsahuje různé naměřené hodnoty uspořádané v jednotlivých seznamech.



Obr. 1: Příklad hierarchicky strukturovaných dat

Univerzální úložiště je tedy koncipováno jako strom, jehož vnitřní uzly určují hierarchickou strukturu dat a jednotlivá data představují listy tohoto stromu. Každý vnitřní uzel stromu má nějaké základní vlastnosti (např. jméno, datum vytvoření, různé identifikátory, poznámky)

a může obsahovat libovolné množství podstromů, uzlů představujících další úroveň stromu (např. každý pacient absolvoval určité návštěvy). Tyto uzly mají přesně definovaný typ (např. návštěva, pacient, měření) podle úrovně, ve které se nacházejí. Vnitřním uzlům se říká Informační položky.

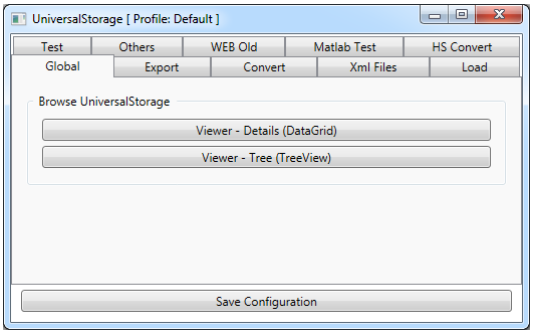
Každý vnitřní uzel – Informační položka - může obsahovat seznam listů, které rozšiřují informace o tomto uzlu a kterým se v Univerzálním úložišti říká InfoGrupy. Například uzel pacient může obsahovat InfoGrupy jako jméno, příjmení, rodné číslo, kontaktní adresa, telefon, zdravotní pojišťovna atd. Tyto InfoGrupy jsou uspořádávány do seznamů InfoGrup seskupujících související informace - např. InfoGrupa se základními informacemi o pacientovi – jméno, příjmení, datum narození a poznámka.

Poslední úroveň stromu představuje samotné naměřené hodnoty, které nazýváme Datové položky. Podobně jako Informační položky mohou mít i Datové položky některé společné vlastnosti, např. informace o platnosti hodnoty, datum a čas, jméno, identifikátor, informace o typu hodnoty aj. Tyto hodnoty reprezentují data různých typů, např. řetězec, celé číslo, seznam desetinných čísel, 3D matice, pole 2D matic.

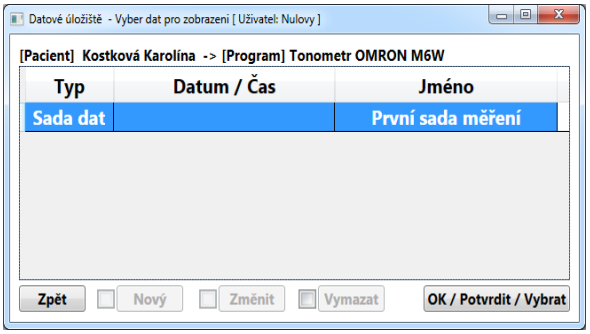
Z takto vystavěné struktury ukládání informací plyne snadná vertikální i horizontální rozšiřitelnost existujících položek, což je další velká výhoda Univerzálního úložiště.

Celý projekt, ač je primárně prezentován jako úložiště medicínských dat, dbá na univerzálnost použití. Po definici jednotlivých úrovní určujících strukturu stromu - v našem případě se jedná o úrovně pacient, návštěva, měření atd., lze Univerzální úložiště použít pro libovolná hierarchicky strukturovatelná data.

Pro interakci s Univerzálním úložištěm byly v této práci využity dva již existující dialogy. První (viz Obr. 2) umožňuje konfiguraci samotného Úložiště a základní zobrazování stromu uložených hodnot. Druhý dialog (viz Obr. 3) slouží k procházení stromem uložených dat a vyzvedávání jednotlivých Informačních a Datových položek z tohoto stromu. Struktura takto vyzvednutých dat je jednotná bez ohledu na nastavení Úložiště nebo data samotná a bude detailně rozebrána v kapitole 2.1. Druhý dialog též umožňuje přidávání, mazání a editaci položek uložených na nejvyšší úrovni stromu.



Obr. 2: Dialog pro konfiguraci Univerzálního úložiště



Obr. 3: Dialog pro výběr dat z Univerzálního úložiště

Univerzální úložiště také obsahuje některé pokročilejší možnosti pro práci s uloženými daty (např. různé formy řízení přístupu k datům) nebo možnosti pro export vybraných dat do prostředí Matlabu (Malaník, 2013).

2.1 Struktura vyzvedávaných dat z univerzálního úložiště

Po vyzvednutí dat z Úložiště pomocí dialogu pro získávání dat (viz Obr. 3) obdržíme pole vyzvednutých hodnot. Ke každé vyzvednuté hodnotě také získáme uspořádaný seznam vnitřních uzlů představující úplnou cestu od kořene stromu k této hodnotě.

Tato cesta plyne z hierarchické struktury uložených dat v Úložišti a lze podle ní přesně určit, jaká Datová nebo Informační položka byla vyzvednuta. Cesta obsahuje veškeré informace, které se týkají jednotlivých uzlů stromu v ní obsažených.

2.2 Informace obsažené ve vyzvednutých datech

Datové i Informační položky vyzvednuté z Úložiště mohou obsahovat kromě samotné naměřené veličiny nebo seznamu podstromů také další doplňující informace. Jde především o **Hodnotový** a **Datový typ** položky, **název** veličiny, **datum a čas pořízení** veličiny, její **validitu** a **komentáře** popisující možné anomálie a další informace.

Většina Datových položek je tvořena různými strukturami Hodnotových typů. Hodnotové typy (ValueType) jsou reprezentovány různými druhy hodnot, které obecně můžeme rozdělit do čtyř kategorií:

- **Boolean** – pravdivostní hodnoty

- **Byte**, **Int8**, **Int16**, **Int32**, **Int64 aj.** – celá čísla
- **Double**, **Single**, **Float** – desetinná čísla
- **String** – řetězce znaků

Tyto Hodnotové typy se skládají do struktur, tzv. Datových typů (DataType)

- **Primitive** – samotná jednoduchá hodnota naměřené veličiny
- **2DPoint** – bod v dvourozměrném prostoru
- **3DPoint** – bod v trojrozměrném prostoru
- **2DMatrix** – dvourozměrná matice
- **3DMatrix** – trojrozměrná matice
- **Field** – pole hodnot
- **List** – seznam hodnot
- **ItemsTimeSpanPrimitive** – seznam dvojic: Primitive – doba trvání
- **ItemsTimeSpanPoint2D** – seznam dvojic: Point2D – doba trvání
- **ItemsTimeSpanPoint3D** – seznam dvojic: Point3D – doba trvání
- **ItemsNamePrimitive** – seznam dvojic: Primitive - jméno

Kromě výše vyjmenovaných jsou definovány v Úložišti také speciální Datové položky, které nemají žádný Hodnotový typ. Mezi ně se například řadí:

- **Version** – verze
- **DeviceDesc** – popis zařízení
- **BitmapImage** – bitmapový obrázek
- **ImageBytes** – obrázek reprezentovaný jako pole bajtů
- **Color** – barva reprezentovaná RGB hodnotou
- **Line** – lomená čára reprezentovaná seznamem bodů
- **Rectangle** – obdélník

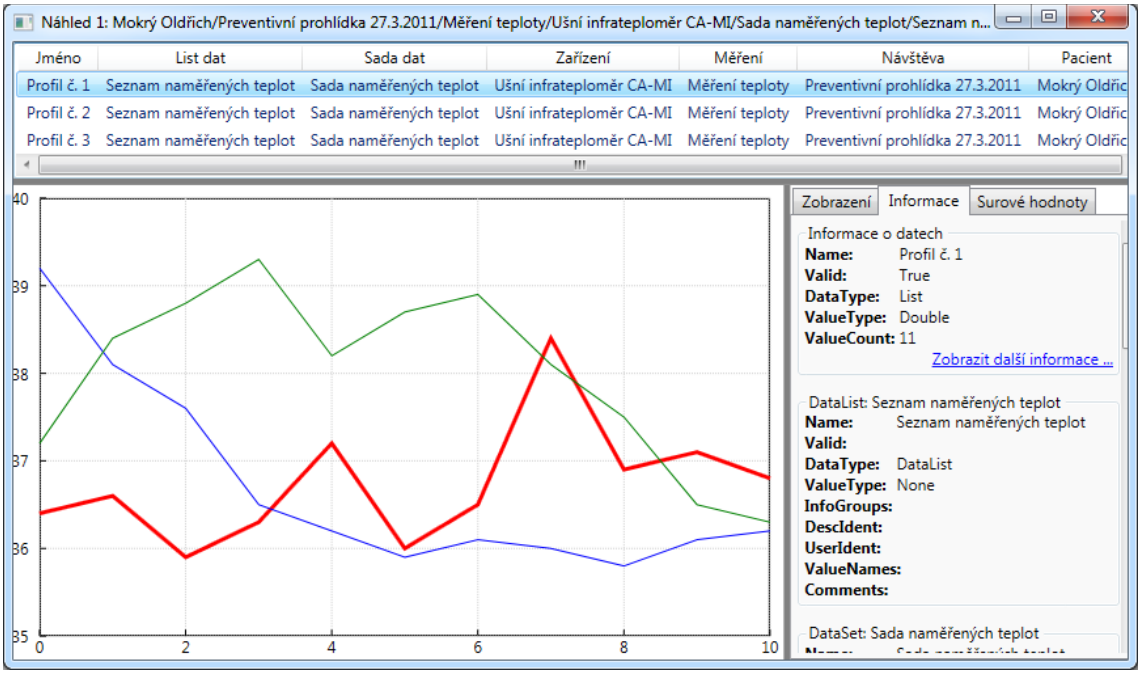
Různé Datové typy pak mohou zpřístupňovat kromě hodnoty naměřené veličiny i doplňující informace o struktuře samotné - například délky seznamů nebo velikosti matice. Detailnější popis všech struktur využívaných Univerzálním úložištěm a jejich hierarchie je rozpracován v příloze B.

Kapitola 3
Možnosti zobrazování vyzvednutých položek

V kapitolách 2.1 a 2.2 byla popsána struktura vyzvednutých dat a předvedeny všechny důležité rozšiřující informace obsažené v této struktuře. Tyto rozšiřující informace a především cesta k datům mohou mít zásadní vliv na interpretaci hodnoty vyzvednutých dat a měly by tedy být uživateli prezentovány společně s daty.

V rámci této práce jsem navrhl způsob, jak tyto informace prezentovat uživateli v ucelené a přehledné formě. Protože vyzvednutá data mohou nabývat velkého množství Datových a Hodnotových typů, byl kladen důraz na univerzálnost navržených zobrazovacích řešení pro více kombinací Datového a Hodnotového typu.

Pro modelovou situaci, kdy uživatel spustí aplikaci, která mu umožní procházet se strukturou Úložiště, vyzvedávat z ní jednotlivé položky a jejich náhledy následně zobrazovat v jednotlivých dialogových oknech, jsem navrhl jednoduché základní rozhraní aplikace (viz Kapitola 5) a dialogové okno náhledů pro vyzvednutá data (viz Obr. 4).



Obr. 4: Dialogové okno náhledů vyzvednutých dat

V horní části dialogového okna se nachází seznam se všemi vyzvednutými daty umožňující výběr jednotlivých položek k následnému zobrazení hodnot a dalších informací. Tento seznam zobrazuje cestu ke každé vyzvednuté položce.

Zbytek dialogového okna je rozdělen do dvou částí. V levé části se nachází grafický náhled hodnot vyzvednutých dat, který odpovídá dané kombinaci Hodnotového a Datového typu. Pravá část ve formě jednotlivých záložek zpřístupňuje možnosti daného zobrazení, zobrazuje další informace o položce aktuálně vybrané ze seznamu a zpřístupňuje zobrazení jejích hodnot v textové formě.

V následujících třech samostatných podkapitolách budou postupně popsány tyto tři části, na které je dialogové okno rozděleno.

3.1 Seznam vyzvednutých dat

Jednotlivé položky vyzvednuté ze struktury Univerzálního úložiště mají každá své jméno a seznam vnitřních uzlů vedoucích od vyzvednuté položky ke kořeni stromu – cestu k uložené položce. Jelikož na jedné úrovni podstromu mají jednotlivé uzly každý svůj unikátní název, tak lze názvů uzlů cesty k uložené položce využít k jednoznačnému rozlišení vyzvednutých položek.

Seznam vyzvednutých dat by tedy kromě názvů vyzvednutých dat, které se mohou opakovat, měl obsahovat i cestu k datům. Seznam by mohl být reprezentován tabulkou (viz Obr. 5), kde řádky představují položky vyzvednuté z Úložiště a sloupce postupně jména uzlů nacházejících se po cestě k těmto datům. První sloupec obsahuje jména samotných dat, druhý sloupec jména uzlů vyzvednutým datům nejblíže (uzly, které obsahují vyzvednutá data jako své přímé potomky), třetí sloupec jména uzlů v následující vyšší úrovni stromu, než jsou uzly z druhého sloupce, a tak dále až k poslednímu sloupci, ve kterém se nacházejí uzly přímo předcházející kořeni stromu. Kořen stromu již není vyzvedáván, neboť je totožný pro každou položku, a není tedy ani zobrazován.

Jméno	List dat	Sada dat	Zařízení	Měření	Návštěva	Pacient
#1	Seznam naměřených teplot	Sada naměřených teplot	Ušní infrateploměr CA-MI	Měření teploty	Preventivní prohlídka 27.3.2011	Mokrý Oldřich
#2	Seznam naměřených teplot	Sada naměřených teplot	Ušní infrateploměr CA-MI	Měření teploty	Preventivní prohlídka 27.3.2011	Mokrý Oldřich
#3	Seznam naměřených teplot	Sada naměřených teplot	Ušní infrateploměr CA-MI	Měření teploty	Preventivní prohlídka 27.3.2011	Mokrý Oldřich
#4	Seznam naměřených teplot	Sada naměřených teplot	Ušní infrateploměr CA-MI	Měření teploty	Preventivní prohlídka 27.3.2011	Mokrý Oldřich
#5	Seznam naměřených teplot	Sada naměřených teplot	Ušní infrateploměr CA-MI	Měření teploty	Preventivní prohlídka 27.3.2011	Mokrý Oldřich
#6	Seznam naměřených teplot	Sada naměřených teplot	Ušní infrateploměr CA-MI	Měření teploty	Preventivní prohlídka 27.3.2011	Mokrý Oldřich
#7	Seznam naměřených teplot	Sada naměřených teplot	Ušní infrateploměr CA-MI	Měření teploty	Preventivní prohlídka 27.3.2011	Mokrý Oldřich
#8	Seznam naměřených teplot	Sada naměřených teplot	Ušní infrateploměr CA-MI	Měření teploty	Preventivní prohlídka 27.3.2011	Mokrý Oldřich

Obr. 5: Seznam vyzvednutých dat

3.2 Zobrazování hodnot vyzvednutých dat

Data vyzvednutá z Úložiště ve formě seznamu dat lze uspořádávat do různých forem položkového zobrazení. Např. do tabulky nebo záložek. Tato položková zobrazení jsou popsána v kapitole 3.2.1.

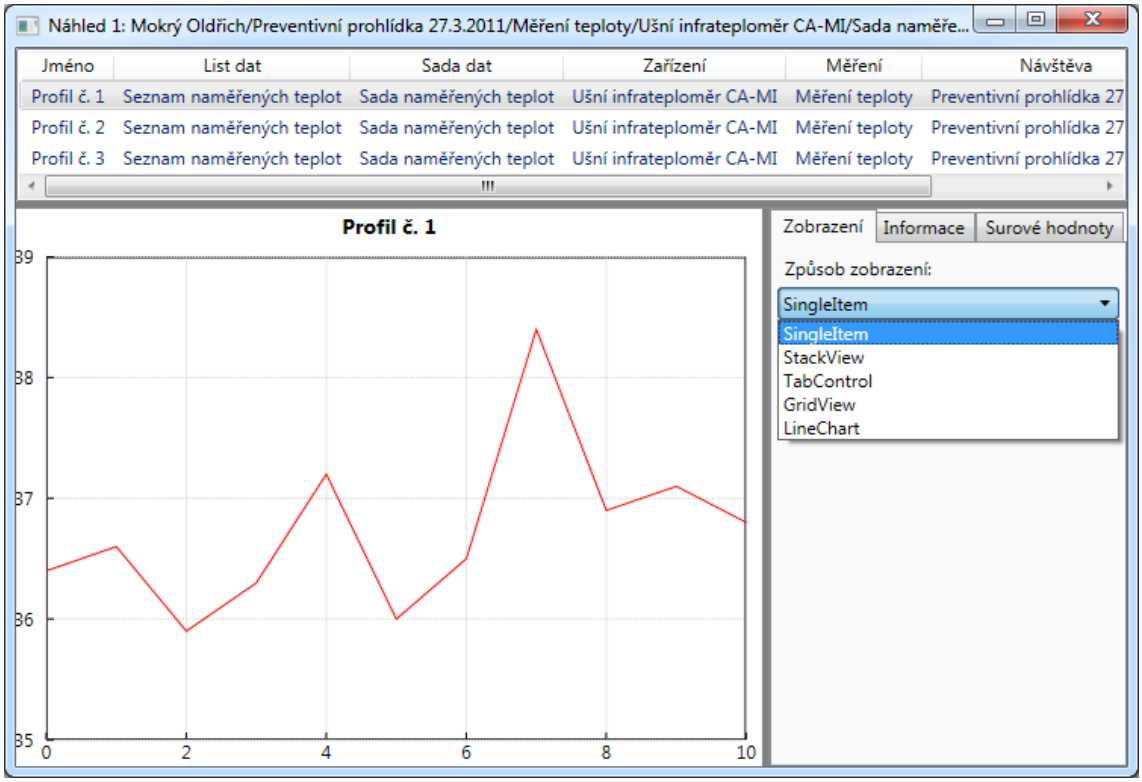
K zobrazování hodnot vyzvednutých dat jsem využil jejich typ, který je kombinací Hodnotového a Datového typu, jak bylo popsáno v kapitole 2.2. Jednotlivá navržená zobrazení jsou popsána v kapitole 3.2.2.

3.2.1 Položková zobrazení

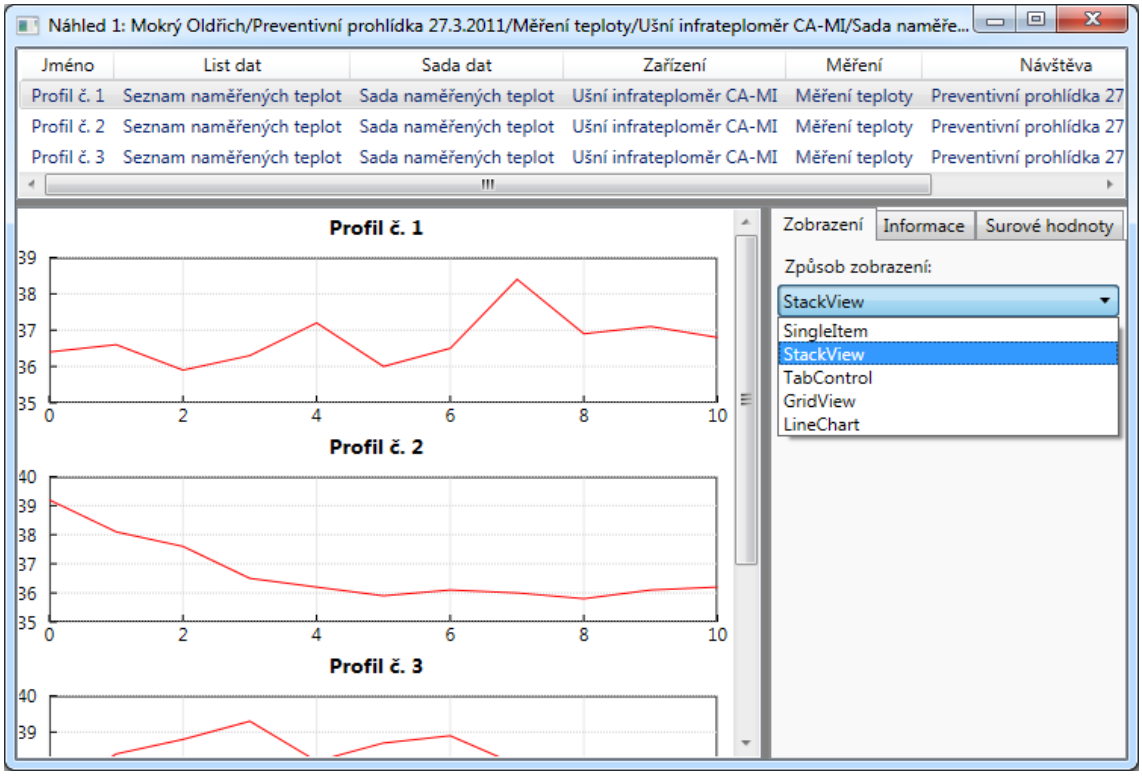
V rámci práce bylo navrženo několik možností uspořádání náhledů položek vyzvednutých z Univerzálního úložiště. Přitom některé náhledy jsou obecné, tedy přípustné pro všechny kombinace vyzvednutých dat. Jiné vyžadují, aby vyzvednuté položky měly určitou hodnotu Hodnotového a Datového typu.

Zpracovaná položková zobrazení (viz Obr. 6 - Obr. 10):

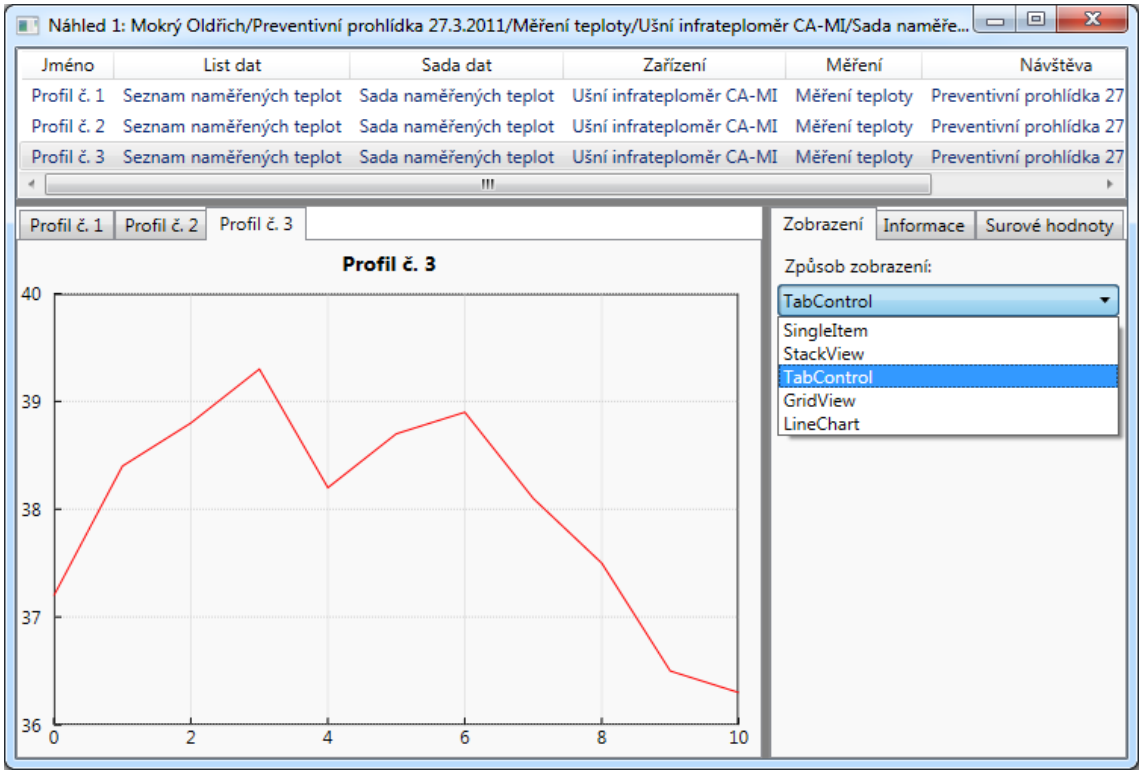
- **SingleItem** – zobrazení hodnoty datové položky vybrané ze seznamu vyzvednutých dat
- **StackView** – zobrazení všech datových položek formou seznamu
- **TabControl** – zobrazení jednotlivých hodnot formou záložek
- **GridView** – textové zobrazení jednotlivých hodnot formou tabulky
- **LineChart** – čárový graf všech vyzvednutých položek se zvýrazněním vybrané položky



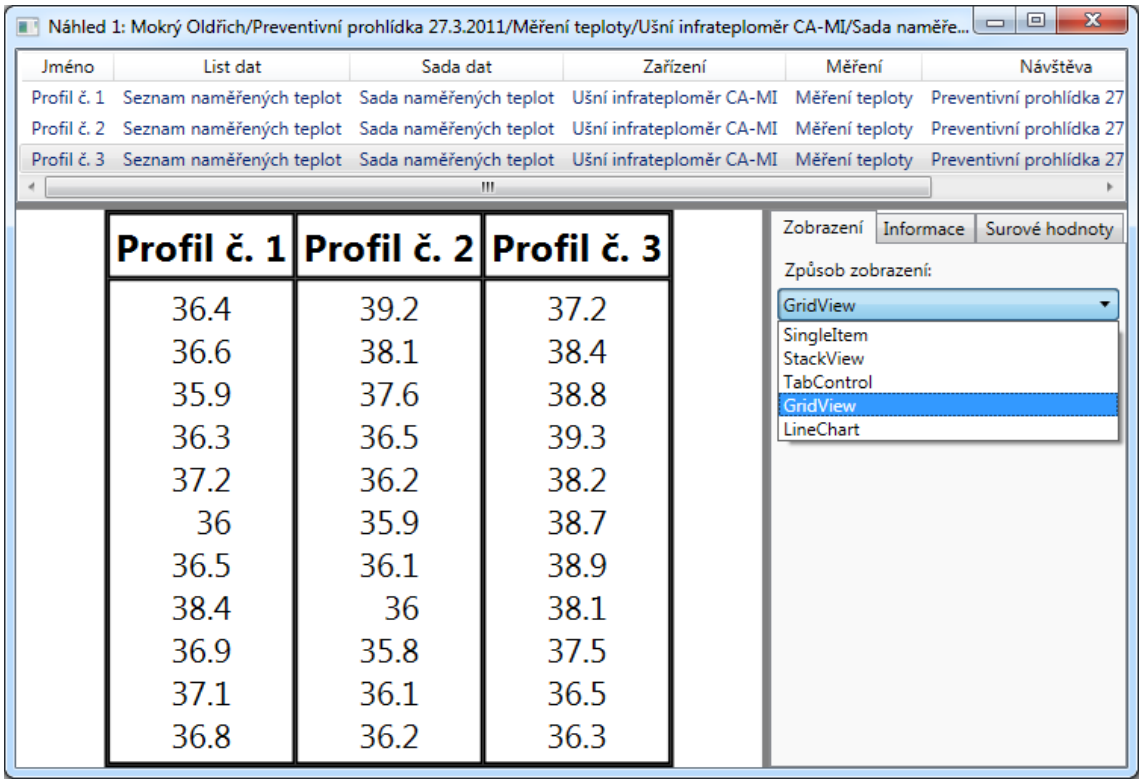
Obr. 6: Položkové zobrazení SingleItem



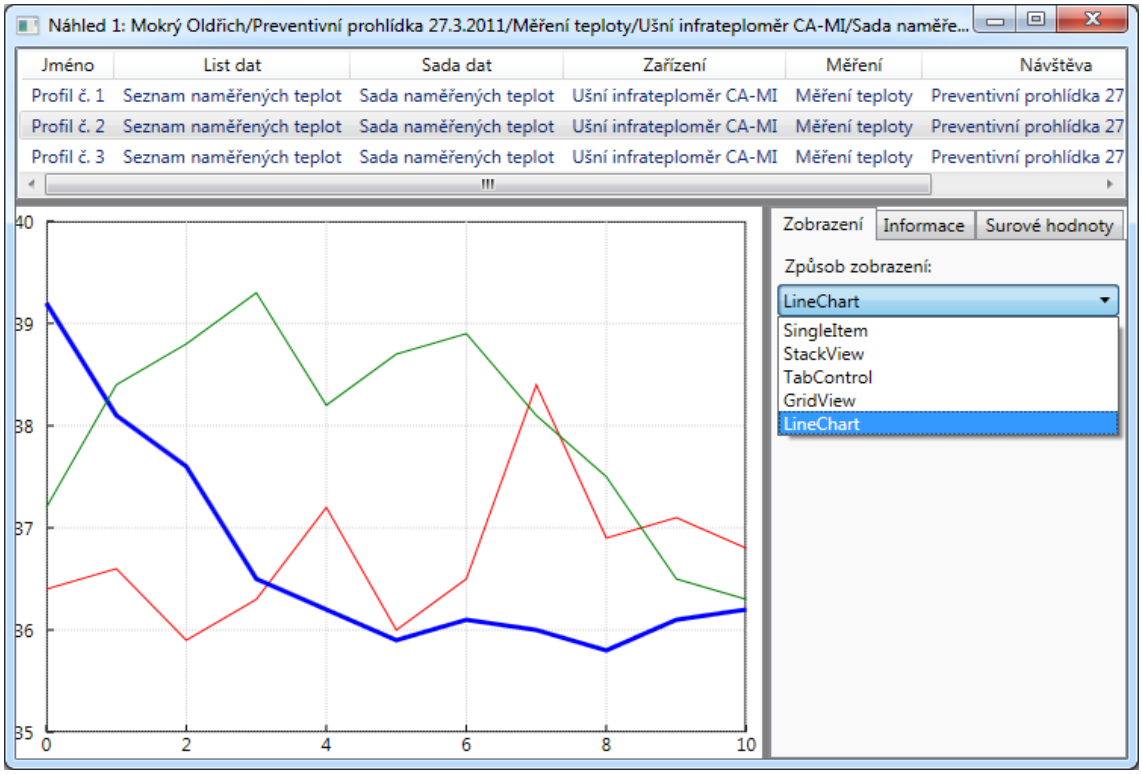
Obr. 7: Položkové zobrazení StackView



Obr. 8: Položkové zobrazení TabControl



Obr. 9: Položkové zobrazení GridView



Obr. 10: Položkové zobrazení LineChart

3.2.2 Zobrazení datových hodnot

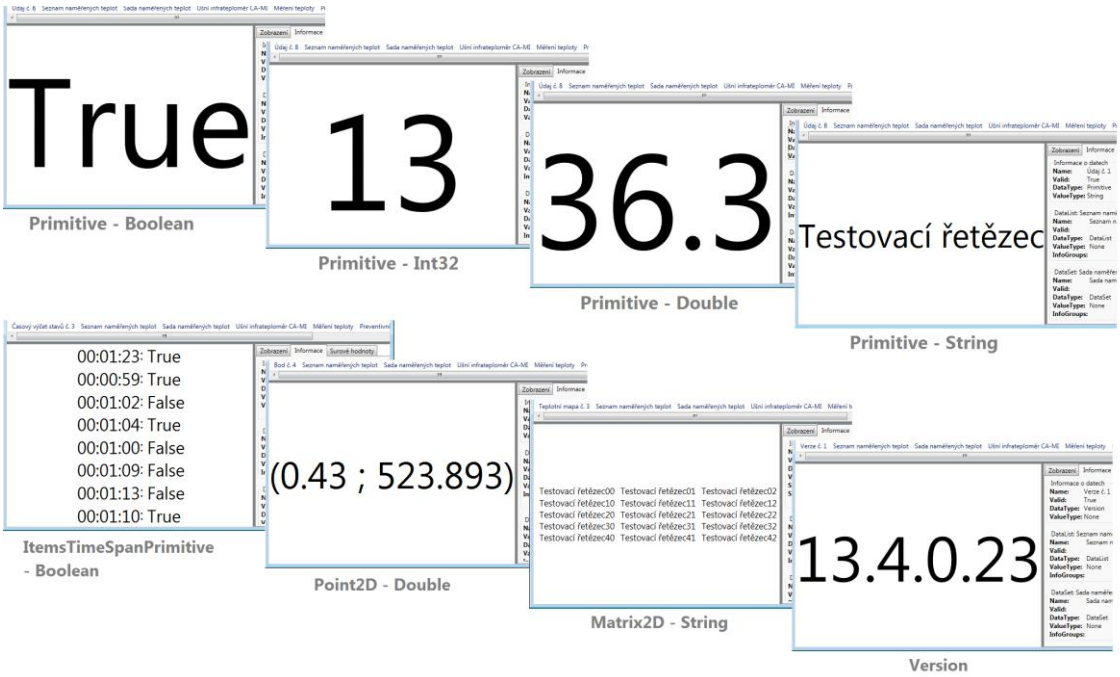
Jak bylo uvedeno, uložená data mají různé kombinace Datových a Hodnotových typů (viz kapitola 2.2), dohromady tvořících širokou škálu možností pro reprezentaci veličiny. Vytyčeným cílem je veličinu zobrazit uživateli v přístupné formě pro každou z těchto kombinací.

Za tímto účelem bylo vybráno několik základních podob vizualizace veličiny:

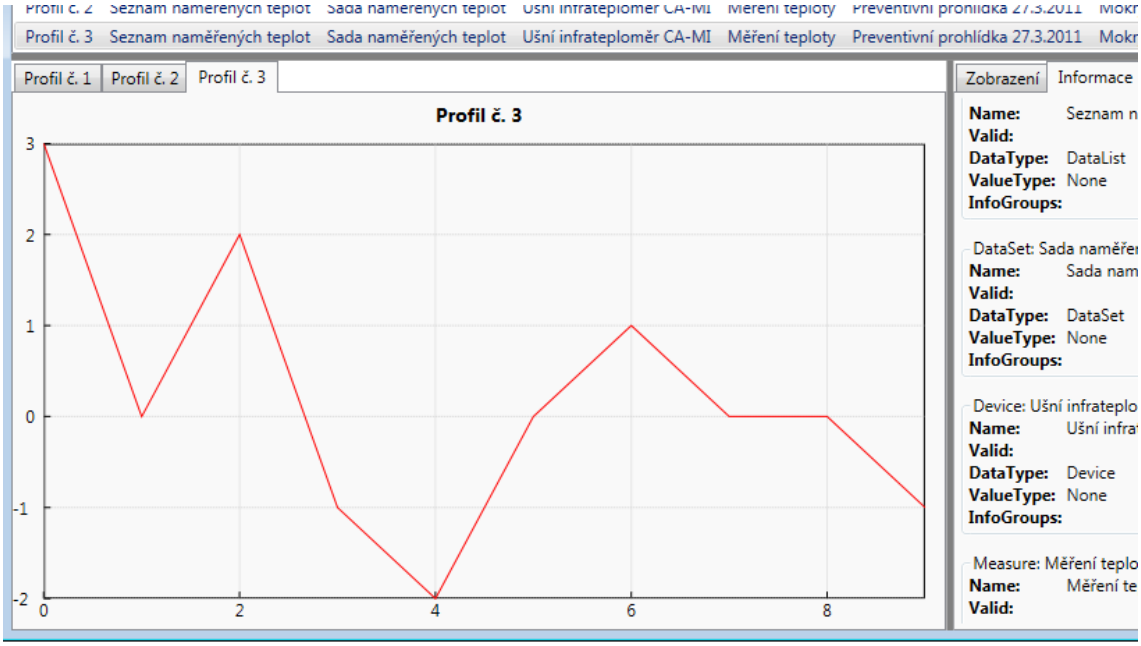
- **Textové forma** (viz Obr. 11) je použita pro vyzvednuté položky, pro které nedává smysl žádná grafická podoba. Například všechny položky Hodnotového typu Primitive, Point2D, ItemsNamePrimitive, Version a všechny položky s Datovým typem pravdivostní hodnoty nebo řetězce znaků.
- **Čárový graf** (viz Obr. 12) je použit pro vyzvednuté položky, které lze intuitivně převést na kolekci bodů v rovině. Například seznamy a pole celých nebo desetinných čísel.

- **Teplotní mapa** (viz. Obr. 13) je použita pro vyzvednuté položky, které lze intuitivně převést na dvourozměrnou matici čísel.

Tyto podoby vizualizací umožňují intuitivní zobrazení každé kombinace Datového a Hodnotového typu momentálně se v Úložišti nacházející.



Obr. 11: Textová forma vizualizace veličiny



Obr. 12: Čárový graf

Obr. 13: Teplotní mapa

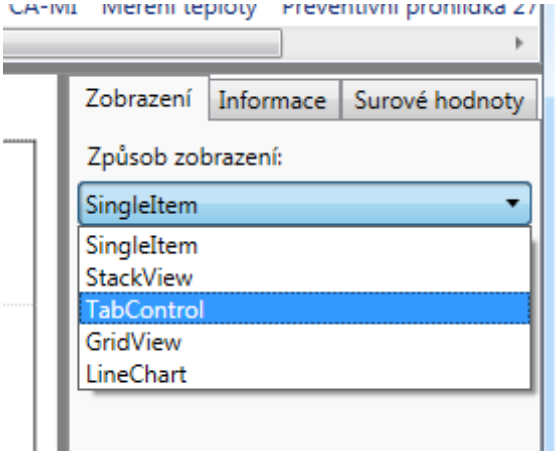
3.3 Dodatečné informace o datech a nastavení zobrazení

Poslední část dialogového okna náhledů na vyzvednutá data obsahuje následující záložky:

- Zobrazení
- Informace
- Surové hodnoty

3.3.1 Záložka Zobrazení

Záložka zobrazení (viz Obr. 14) umožňuje z roletového menu vybrat použitý styl položkového zobrazení (viz kapitola 3.2.1).

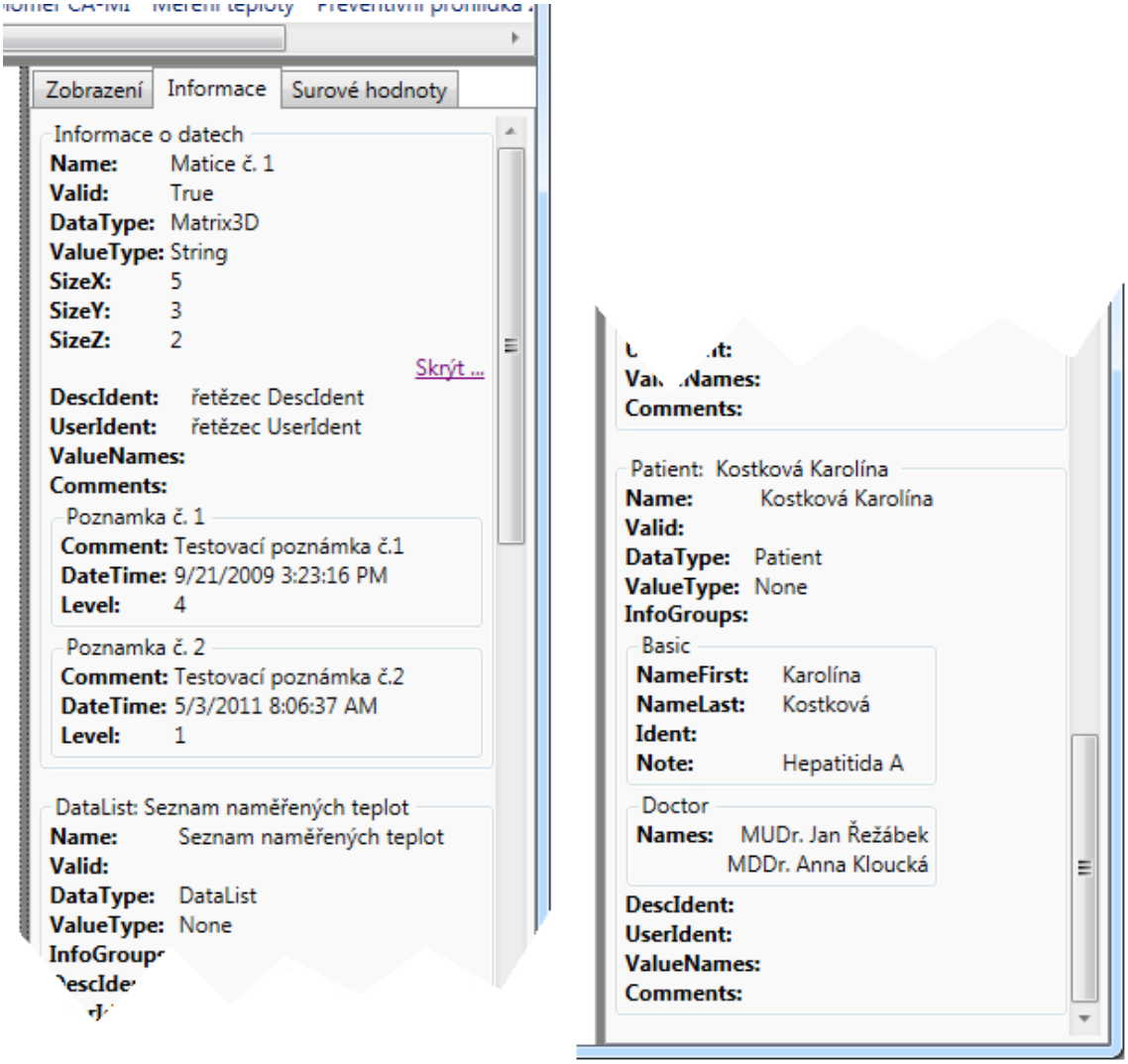


Obr. 14: Záložka zobrazení

3.3.2 Záložka Informace

Záložka informace (viz Obr. 15) je dynamicky propojena se seznamem vyzvednutých dat a obsahuje v textové formě veškeré informace o vybrané vyzvednuté položce a o všech uzlech nacházejících se na cestě k této položce.

V této záložce jsou uvedeny v jednotlivých boxech postupně informace o každém uzlu, jeho název, platnost, Datový a Hodnotový typ, identifikátory, komentáře a položky z InfoGrup v případě Informačních položek nebo informace o struktuře u Datové položky.



Obr. 15: Záložka Informace

3.3.3 Záložka Surové hodnoty

Záložka surové hodnoty (viz Obr. 16) je dynamicky propojena se seznamem vyzvednutých dat a zobrazuje v textové podobě veličiny, zdrojové hodnoty využité pro uživatelské zobrazení (viz kapitola 3.2.2).

Obr. 16: Záložka Surové hodnoty

Kapitola 4
Koncept řešení

Pro vypracování jednotlivých částí aplikace pro zobrazování dat získaných z Univerzálního úložiště (viz Kapitola 3) jsem v souladu s předešlou prací na projektu použil programovací jazyk C# a framework .NET 4.0.

Návrh dynamického uživatelského rozhraní, které je jedním z úkolů této bakalářské práce, jsem postavil na technologii WPF. V rámci WPF existuje vždy mnoho způsobů, jak dosáhnout požadovaného vzhledu a chování jednotlivých prvků uživatelského rozhraní (Nathan, 2010). Snahou bylo vytvořit elegantní, modulární a jednoduše rozšiřitelné řešení, které umožní integraci nových funkcí a zobrazovacích možností.

V rámci této kapitoly budou představeny některé základní koncepty, které nám knihovna WPF nabízí a na kterých je následně postaveno vypracované řešení.

4.1 Hierarchie tříd WPF

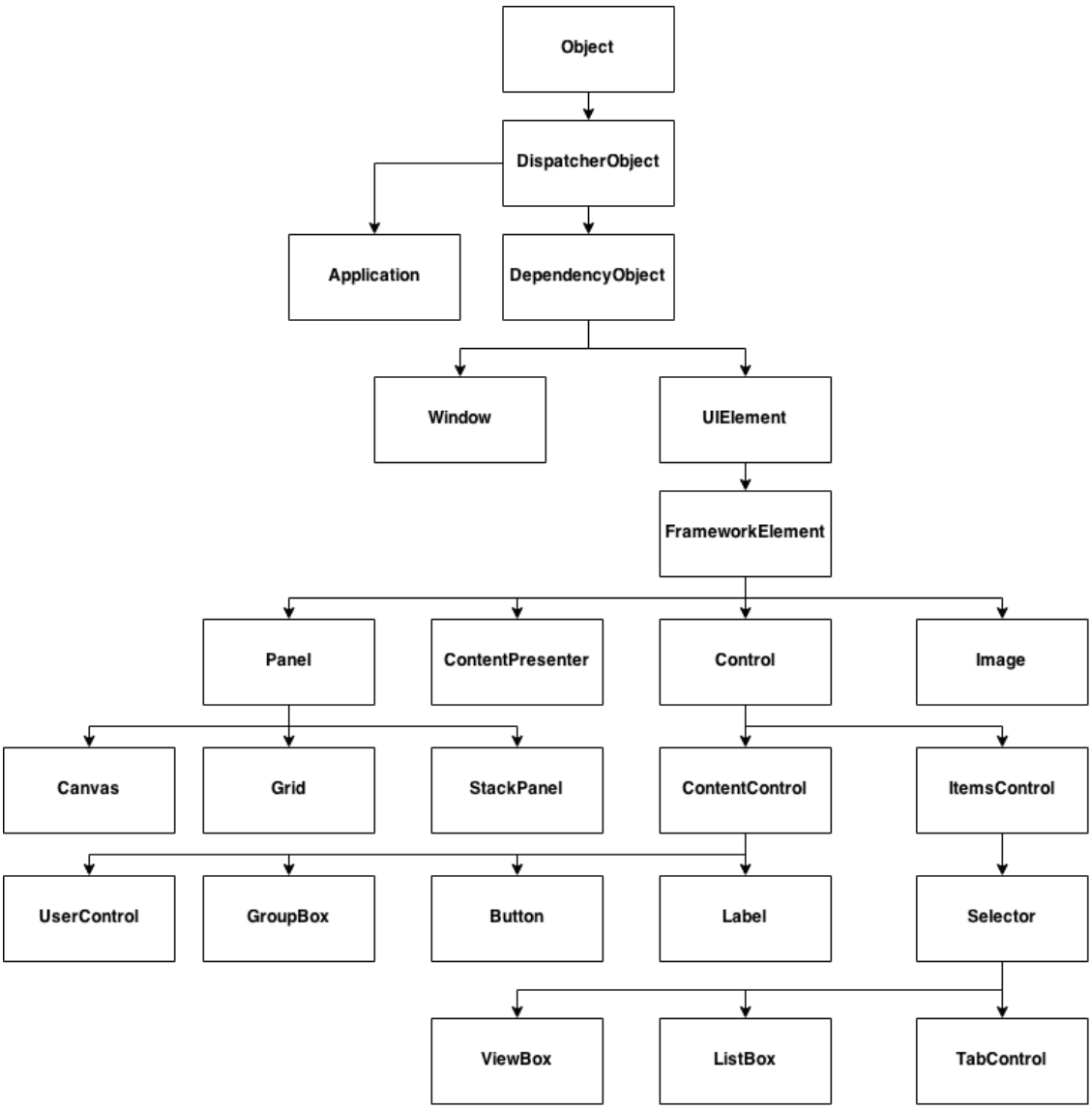
Knihovna WPF zakládá na hluboké dědičnosti jednotlivých tříd. Hierarchie základních tříd je schematicky znázorněna na Obr. 17.

Třída `Application` zapouzdřuje aplikaci založenou na WPF a je odvozena od abstraktní třídy `DispatcherObject`, která je předkem pro každý objekt, který má být přístupný jen ve vlákně, které ho vytvořilo.

Třída `DependencyObject` je základovou třídou pro třídy podporující dependency properties a attached properties (viz kapitola 4.3). Třída `Window` představuje okno aplikace založené na knihovně WPF a třída `UIElement` je předkem pro všechny elementy tohoto okna.

Od abstraktní třídy `FrameworkElement` jsou odvozeny jednotlivé elementy, ze kterých se skládá uživatelské rozhraní. Jde o komponenty zpřístupňující různé formy obsahu a kontejnery (potomky třídy `Panel` – např. `Canvas`, `StackPanel`, `Grid`) do nichž lze jako logické potomky (viz kapitola 4.2) vkládat komponenty uživatelského rozhraní.

Komponenty zpřístupňující různé formy obsahu se dají rozdělit na skupinu prvků zobrazujících typově speciální obsah (např. obrázek – třída `Image`, textové pole – třída `TextBlock` nebo 3D grafika – třída `Viewport3D`) a na početnou skupinu prvků, odvozených od třídy `Control`, využívajících šablony pro definici svého vzhledu (viz kapitola 4.4).



Obr. 17: Hierarchie základních tříd WPF

4.2 XAML

XAML (Extensible Application Markup Language) je značkovací programovací jazyk vhodný pro konstrukci a inicializaci objektů, založený na XML (Extensible Markup Language). XAML na rozdíl od XML mapuje své jednotlivé objekty na elementy knihovny WPF a tím umožňuje deklarativním způsobem vytvářet grafická uživatelská rozhraní. Definování objektu pomocí XAML je ekvivalentní k definici objektu pomocí defaultního konstruktoru. Proto každý element deklarovaný v XAML lze napsat i jen pomocí procedurálního kódu (toto ovšem neplatí naopak).

XAML soubory stejně jako všechny XML soubory jsou strukturovány ve formě stromu. Z toho tedy vyplývá, že jednotlivé prvky mohou obsahovat jednoho nebo více následovníků a s výjimkou kořenového prvku má každý prvek svého rodiče. Tato struktura představuje logický strom, ze kterého se v aplikacích postavených nad knihovnou WPF následně generuje uživatelské rozhraní. Uživatelské rozhraní je reprezentováno grafickým stromem vzniklým dekompozicí logického stromu až na základní elementy. Logický strom je tedy zjednodušením grafického stromu.

Přestože kód aplikace může být ve speciálních případech zcela napsán v XAML a následně zobrazen například pomocí prohlížeče Internet Explorer, tak pro reálné aplikace je nutné vedle XAML použít i procedurální kód. Mísit XAML a procedurální kód lze dvěma způsoby. Prvním způsobem je dynamické nahrání a parsování XAML a druhým způsobem je jeho kompilace. Kompilace probíhá ve třech krocích – nejdříve se kód převede do speciálního binárního formátu (tzv. BAML – Binary Application Markup Language), v druhém kroku se připojí převedený kód k assembly (sestavení aplikace) jako binární zdroj a nakonec je XAML provázán s procedurálním kódem.

4.3 Závislé vlastnosti

Závislé vlastnosti (dependency properties) a přidružené závislé vlastnosti (attached properties) jsou nové typy vlastností užívané u elementů knihovny WPF. Jsou definovány na abstraktní třídě `FrameworkElement`.

Dependency properties přidávají k běžným vlastnostem tříd pokročilou funkcionalitu – především upozorňování na změny (change notification), dědičnost hodnoty (property value inheritance) a podporu pro více poskytovatelů hodnoty (support for multiple providers). Nejsilnější stránkou dependency properties je možnost jejich svazování (Binding) s vlastnostmi a jinými dependency properties na rozdílných prvcích uživatelského rozhraní. Tento koncept umožňuje dynamicky aktualizovat libovolnou závislou vlastnost prvku v návaznosti na hodnotě libovolné závislé vlastnosti jiného prvku a tedy i v návaznosti na akce uživatele. Aktualizované vlastnosti můžeme přiřadit buď hodnotu svázané vlastnosti nebo tuto hodnotu libovolně změnit pomocí konvertoru (třídy implementující rozhraní `IValueConvertor`).

Koncept attached properties umožňuje přidružit k existujícím třídám odvozeným od `FramevorkElement` nové dependency properties, které mohou měnit stav prvku pomocí jeho veřejných metod a vlastností.

4.4 Šablony

Prvky odvozené od třídy `Control` (viz kapitola 4.1) definují svůj vzhled na základě šablon – speciálních tříd odvozených od `FrameworkTemplate`. Vzhled prvku lze rozdělit do dvou částí. První část definuje vizuální podobu celého prvku jako kontejneru na svůj obsah a druhá část popisuje, jak budou data (obsah prvku) vykreslena. Z tohoto rozdělení plyne existence dvou druhů šablon.

Šablony typu `ControlTemplate`, které definují vizuální podobu prvku, popisují jeho zobrazení v uživatelském rozhraní a jsou svázány jen se samotným uživatelským rozhraním.

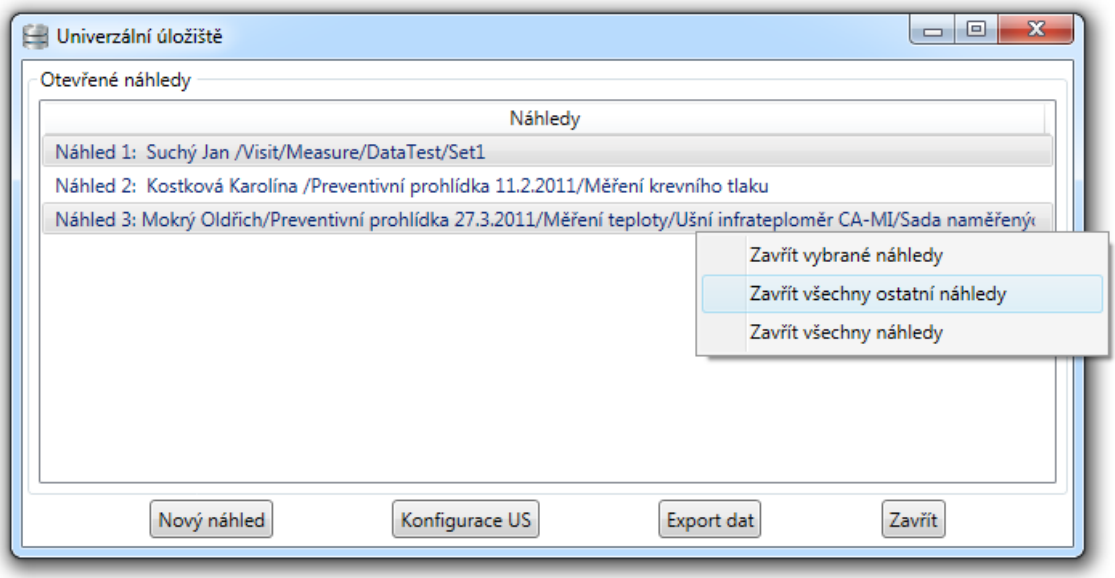
Šablony typu `DataTemlpate` jsou svázány se zobrazovanými daty a popisují, jak mají být data reprezentována. Tyto šablony se aplikují prostřednictvím objektu třídy `ContentPresenter`, který se pro přiřazená data na základě jejich typu pokusí najít vhodnou šablonu a tu pak aplikovat pro vytvoření grafického stromu. Pokud `ContentPresenter` nenalezne šablonu odpovídající přiřazenému obsahu, tak obsah zobrazí v textové podobě (`TextBlock`) pomocí řetězce získaného metodou `ToString`.

Jednotlivé objekty tříd odvozených od `Control` mohou využívat více šablon k sestavení svého grafického stromu. Například pro zobrazení objektu třídy `ItemsControl,` sloužící k zobrazování seznamu prvků, jsou použity čtyři různé šablony:

- `ItemsPanel` – šablona definující kontejner pro zobrazení jednotlivých prvků
- `ItemTemplate` – šablona definující vizuální strom pro zobrazení jednotlivých prvků
- `ItemContainerStyle` – styl definující vzhled elementu zapouzdřujícího prvky seznamu
- `Template` – šablona definující podobu objektu `ItemsControl`

Kapitola 5
Základní dialogové okno

Po spuštění samotné aplikace se zobrazí Základní dialogové okno (viz Obr. 18). To má za úkol přímočaře zpřístupnit uživateli jednotlivé části aplikace a umožnit pokročilou správu otevřených dialogů s náhledy dat. Tento přístup se ukázal výhodný zejména při práci s vysokým počtem otevřených dialogových oken, kdy usnadňuje uživateli orientaci mezi jednotlivými dialogy.



Obr. 18: Základní dialogové okno aplikace

Základní dialog ve spodní části okna nabízí několik tlačítek pro základní úkony:

- **Nový náhled** – zobrazí dialog pro vyzvednutí dat z Úložiště (viz Kapitola 2) a v případě úspěšného výběru dat následně zobrazí Zobrazovací dialog (viz Kapitola 3)
- **Konfigurace US** - zobrazí dialog pro konfiguraci Univerzálního úložiště (viz Kapitola 2)
- **Export dat** – zobrazí dialog pro export dat (viz Kapitola 7)
- **Zavřít** – zavře celou aplikaci včetně všech náhledů

Horní část okna obsahuje interaktivní seznam všech zobrazených náhledů. Tento seznam ve formě kontextového menu spravuje otevřené náhledy. Po vybrání položek ze seznamu se

po stisknutí pravého tlačítka myši zobrazí menu umožňující různými způsoby zavírat jednotlivé náhledy s vyzvednutými daty. Konkrétně zavřít vybrané náhledy, zavřít všechny náhledy krom vybraných a zavřít úplně všechny náhledy. Po dvojkliku levým tlačítkem myši se zobrazí žádaný náhled v popředí. Vybírat více položek ze seznamu lze dle zvyklostí operačního systému (Shift + šipky, Ctrl+Shift+označování myší, Ctrl+označování jednotlivých položek myší apod.).

Kapitola 6
Popis řešení

Pro zobrazování vyzvednutých dat byly využity technologie popsané v Kapitole 4. V následujících podkapitolách budou popsány implementační detaily vytvořené aplikace důležité k pochopení fungování jednotlivých částí aplikace (komponent) a umožňující integraci těchto komponent v rámci možné navazující práce na projektu Univerzálního úložiště.

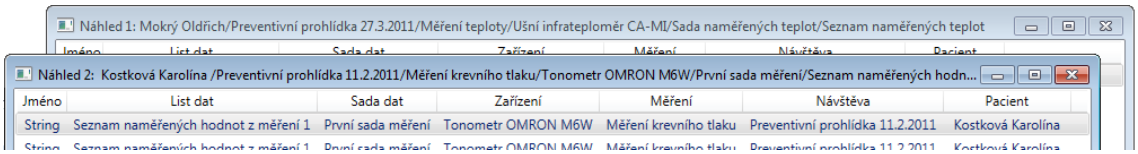
Řešení vychází z podoby a chování jednotlivých částí aplikace navržených v Kapitole 3.

6.1 Dialogové okno náhledů vyzvednutých dat

Jedná se o klasické okno vytvořené ve WPF, které je uspořádané do mřížky (`Grid`), jejíž jednotlivé buňky jsou odděleny posuvnými rozdělovači (`GridSplitter`).

Titulek okna (viz Obr. 19) je dynamicky generován ze seznamu vyzvednutých dat pomocí konvertoru (`StorageDataValueInfoToWindowTitleCorverter`), který převádí pole vyzvednutých dat na řetězec sestavený z pořadového čísla náhledu a jmen uzlů nejdelší společné cesty vyzvednutých dat.

Tento titulek je také využit k rozlišení jednotlivých dialogových oken v Základním dialogovém oknu aplikace (viz Kapitola 5).



Obr. 19: Titulek dialogového okna náhledů vyzvednutých dat

6.2 Seznam vyzvednutých dat s informacemi o cestě

Seznam vyzvednutých dat (`USDataValueInfoListViewControl`) popsaný v kapitole 3.1 je implementován jako potomek třídy `ListView` (viz kapitola 4.1) s překrytou metodou `OnItemsSourceChanged` (viz Výpis programu 1), která při změně obsahu seznamu dynamicky vygeneruje sloupce tabulky (`GridView`) v závislosti na obsažených uzlech.

Jména obsažená v jednotlivých buňkách jsou získávána jako jméno daného uzlu cesty pomocí konvertoru s parametrem (StorageDataValueInfoToPathStringCorverter).

```
protected override void OnItemsSourceChanged(IEnumerable oldValue, IEnumerable newValue)
{
    var dataCollection = newValue as UniData.StorageDataValueInfo[];
    // Definice nového tabulkového zobrazení
    GridView gridView = new GridView();
    this.View = gridView;

    //Vygenerování prvního sloupce se jménem vyzvednuté položky
    gridView.Columns.Add(new GridViewColumn
    {   //definice hlavičky sloupce
        Header = "Jméno",
                 DisplayMemberBinding = new Binding(".")
                   { Converter = new StorageDataValueInfoToNameStringCorverter() }
    });

    //Generování sloupců na základě cesty k vyzvednuté položce
    int i = 0; //proměnná uchovávající polohu uzlu v cestě
    foreach (var col in dataCollection[0].path)
    {
        gridView.Columns.Add(new GridViewColumn
        {   //Definice hlavičky sloupce
            Header =
                 DataValueTypeToString(dataCollection[0].path[i].dataValue.DataValueType),
                 DisplayMemberBinding = new Binding(".")
                    { //konvertor mapující polohu v cestě na jméno uzlu
                      Converter = new StorageDataValueInfoToPathStringCorverter(),
                                  ConverterParameter = i
                    }
        });
        i++;
    }
}
```

Výpis programu 1: Automatické generování sloupců seznamu vyzvednutých dat (zjednodušeno)

6.3 Grafický náhled hodnot vyzvednutých dat

Grafického náhledu hodnot vyzvednutých dat popsaných v Kapitole 3 jsem docílil implementací několika komponent pro grafickou vizualizaci vyzvednutých dat a komponenty umožňující prezentaci jednotlivých zobrazení v několika typech položkového zobrazení.

Jednotlivé komponenty pro grafickou vizualizaci dat jsou zpřístupněny prostřednictvím prvku DataValueControl, který je přímým potomkem třídy Control. Tento prvek stanovuje, jakým způsobem budou data zobrazena. Určuje tak pomocí datových šablon (DataTemplate –

viz kapitola 4.4) pro každou datovou třídu obsaženou v Univerzálním úložišti, která je charakterizována kombinací Hodnotového a Datového typu.

Kvůli správnému provázání uložených hodnot s jednotlivými prvky šablony jsem pro některé datové třídy vytvořil konvertory (viz kapitola 4.3). Např. `Array2DtoListOfArraysConverter` pro konverzi vícerozměrných polí (z Multidimensional array na Jagged array) za účelem správné interpretace hodnoty při svázání s vlastností `ItemsSource` třídy `ItemsControl`.

6.3.1 Položková zobrazení

Položková zobrazení jsou implementována jako jednotlivé šablony (`CotrolTemplate` – viz kapitola 4.4) prvku `ItemsControl`, mezi kterými se přepíná v závislosti na stavu attached property `ViewHelper.ViewType` (viz kapitola 4.3), jejíž hodnota je odvozena od seznamu vyzvednutých dat (viz kapitola 6.4.1.) za pomoci konvertoru (`DataToViewTypeConverter`). Tato přidružená závislá vlastnost může nabývat několika hodnot, které odpovídají použitému zobrazení:

- **`None`** – nepodařilo se určit žádný vhodný způsob pro zobrazení vyzvednutých dat
- **`SingleItem`** – zobrazen jediný prvek typu `DataValueControl`, jehož obsah je svázán s aktuálně vybranými daty ze seznamu vyzvednutých dat (viz kapitola 6.2)
- **`GridView`** – zobrazena tabulka všech vyzvednutých hodnot v textové formě pomocí `RawDataValueControl` (viz kapitola 6.4.3)
- **`StackView`** – jednotlivé prvky `DataValueControl` pro každou vyzvednutou položku řazeny do seznamu pod sebou
- **`TabControl`** – jednotlivé prvky `DataValueControl` umístěny každý v samostatné záložce
- **`LineChart`** – vyzvednuté položky reprezentovány čárovým grafem (viz kapitola 6.3.3)

6.3.2 Textová forma

Textová forma zobrazení vyzvednutých dat je pro jednoduché Hodnotové typy (Primitive, Point2D, Point3D atd.) implementována pomocí šablony obsahující jedno nebo více textových polí (`TextBlock`) pro zobrazení hodnoty veličiny - např. pro data typu Point2D viz Výpis programu 2.

```
<!-- Šablona pro zobrazení datových tříd odvozených od DataValuePoint2DBase -->
<DataTemplate DataType="{x:Type UniDataValues:DataValuePoint2DBase}">
    <Viewbox><!--prvek zajišťující škálovatelnost textového pole-->
        <StackPanel Orientation="Horizontal"><!--kontejner na textová pole-->
            <TextBlock Text="(" />
            <TextBlock Text="{Binding Path=ValueX}" /><!--X-ová hodnota -->
            <TextBlock Text=" ; " />
            <TextBlock Text="{Binding Path=ValueY}" /><!--Y-ová hodnota -->
            <TextBlock Text=")" />
        </StackPanel>
    </Viewbox>
</DataTemplate>
```

Výpis programu 2: Šablona pro třídy odvozené od DataValuePoint2DBase

Pro výčtové Hodnotové typy (např. List položek Datového typu String) je v šabloně obsažen prvek `ItemsControl`, který ve formě textových polí zobrazí jednotlivé hodnoty veličiny - např. pro 2DMatrix datového typu String viz Výpis programu 3.

```
<!-- Šablona pro zobrazení datové třídy DataValueMatrix2DString -->
<DataTemplate DataType="{x:Type UniDataValues:DataValueMatrix2DString}">
    <Viewbox><!--prvek zajišťující škálovatelnost textového pole-->
        <ItemsControl ItemsSource="{Binding Path=Values}"><!--sloupce matice-->
            <ItemsControl.ItemsPanel>
                <ItemsPanelTemplate><!--kontejner pro sloupce matice-->
                    <StackPanel Orientation="Horizontal" />
                </ItemsPanelTemplate>
            </ItemsControl.ItemsPanel>
            <ItemsControl.ItemTemplate>
                <DataTemplate>
                    <ListBox ItemsSource="{Binding}" /><!--řádky matice-->
                </DataTemplate>
            </ItemsControl.ItemTemplate>
        </ItemsControl>
    </Viewbox>
</DataTemplate>
```

Výpis programu 3: Šablona pro třídu DataValueMatrix2DString

6.3.3 Čárový graf

Pro znázornění posloupností číselných hodnot se jako nejvhodnější grafické znázornění jeví čárový graf. Pro reprezentaci čárového grafu jsem odvodil od třídy `ListBox` novou třídu `LineChartControl`, která rozšiřuje třídu `ListBox` o několik dependency properties:

- `Xmin, Xmax, Ymin, Ymax` – reprezentují rozsah přiřazených hodnot
- `Title, IsTitle` – definují titulek grafu

- `XLabel, IsXLabel, YLabel, IsYLabel` – definují popisky X-ové a Y-ové osy

Třídu `ListBox` jsem určil jako nejvhodnějšího kandidáta pro předka, neboť dědí od třídy `Selector`, a díky tomu bude umožněna synchronizace momentálně vybrané datové řady (posloupnosti 2D bodů) s vybranou položkou ze seznamu vyzvednutých hodnot (viz kapitola 6.2). Povědomí o momentálně vybrané položce následně umožňuje její zvýraznění.

Vizuální podoby čárového grafu jsem docílil použitím vhodných šablon pro grafickou interpretaci jednotlivých elementů definovaných na třídě `ListBox`. Tyto šablony jsou definovány v defaultním stylu pro reprezentaci objektu třídy `LineChartControl`, který se standardně nachází v `Themes/Generic.xaml`.

Základem vizuální podoby čárového grafu je šablona přiřazená vlastnosti `Template` určené k definování nového vzhledu prvku (viz Výpis programu 4). Vlastnost `Template` je definovaná předkem `Control`.

V rámci šablony `Template` jsem použil nově vytvořenou třídu `GridlineCanvas`, která je potomkem kontejneru `Canvas`. `GridlineCanvas` má za úkol vkládat do grafu osy, číselné popisky os a mřížku. Za tímto účelem zpřístupňuje několik vlastností (např. `NoXticks` – počet vodících čar mřížky na X-ové ose, `GridLineThickness` – síla čar mřížky, `AxisTextColor` – barva číselných popisků os aj.). Samotný návrh na vkládání „hezkých“ popisků os (obvykle celá sta, celé desítky, celé desetiny, čísla končící číslovkami 1, 2 nebo 5 apod.) je implementován podle algoritmu uvedeného v článku *Nice numbers for graph labels* (Glassner, 1993).

```
<!--Control template čárového grafu-->
<ControlTemplate TargetType="localLineChart:LineChart">
  <DockPanel><!-- kontejner obsahující názvy jednotlivých os a plochu grafu  -->
    <DockPanel.Resources><!-- konvertor z pravdivostní hodnoty na hodnotu viditelnosti -->
      <BooleanToVisibilityConverter x:Key="BooleanToVisibilityConverter" />
    </DockPanel.Resources>
    <TextBlock DockPanel.Dock="Top<!-- Titulek grafu -->
         Visibility="{TemplateBinding IsTitle, <!-- zobrazení upravuje propery IsTitle -->
                                Converter={StaticResource BooleanToVisibilityConverter}}"
         Text="{TemplateBinding Title}"/>
     <TextBlock DockPanel.Dock="Bottom"<!-- Název X-ové osy -->
          Visibility="{TemplateBinding IsXLabel,<!--zobrazení upravuje propery IsXLabel-->
                                Converter={StaticResource BooleanToVisibilityConverter}}"
          Text="{TemplateBinding XLabel}"/>
     <TextBlock DockPanel.Dock="Left"<!-- Název Y-ové osy -->
          Visibility="{TemplateBinding IsYLabel, <!--zobrazení upravuje propery IYLabel-->
                                Converter={StaticResource BooleanToVisibilityConverter}}"
          Text="{TemplateBinding YLabel}"
          RenderTransformOrigin="0.5,0.5"><!-- nastavení počátku pro otočení popisku -->
        <TextBlock.LayoutTransform><!-- otočení popisu Y-ové osy o 90° -->
            <RotateTransform Angle="-90" />
        </TextBlock.LayoutTransform>
      </TextBlock>
       <!-- prvek zobrazující mřížku, osy a popisky os -->
      <localLineChart:GridlineCanvas Margin="2" x:Name="textCanvas"
                              Xmin="{TemplateBinding Path=Xmin}"
                              Xmax="{TemplateBinding Path=Xmax}"
                              Ymin="{TemplateBinding Path=Ymin}"
                              Ymax="{TemplateBinding Path=Ymax}">
          <ItemsPresenter /><!-- prvek vkládající jednotlivé datové řady -->
      </localLineChart:GridlineCanvas>
    </DockPanel>
</ControlTemplate>
```

Výpis programu 4: ControlTemplate čárového grafu

Datové řady reprezentované jednotlivými linkami čárového grafu jsou vytvářeny pomocí datové šablony přiřazené k vlastnosti `ItemTemplate`, která odpovídá za zobrazování jednotlivých položek ve třídách odvozených od třídy `ItemsControl`.

Datová šablona předpokládá jednotlivé položky typu `PointCollection`, které pak převádí pomocí konvertoru (`PointsToNormalizePointsConverter`) do normalizované podoby vzhledem k rozměrům zobrazované plochy grafu a souřadnicím daných bodů. Z normalizovaných `PointCollection` jsou konstruovány jednotlivé lomené čáry (`Polyline`). Barva lomené čáry je definována v závislosti na poloze (pořadí) generující položky v seznamu hodnot pomocí konvertoru (`ItemToBrushConverter`), který implementuje rozhraní `IMultiValueConverter`.

Na každé lomené čáře je definováno několik událostí měnících zvýraznění lomené čáry v závislosti na akci uživatele (např. najetí myší nebo vybrání datové řady).

Jednotlivé lomené čáry jsou vkládány do vizuálního stromu grafu prostřednictvím objektu `ItemsPresenter` (viz Výpis programu 4). Čáry jsou zabalené v kontejneru definovaného vlastností `ItemsPanelTemplate` na třídě `ItemsControl`. V případě čárového grafu jde o komponentu `Canvas`.

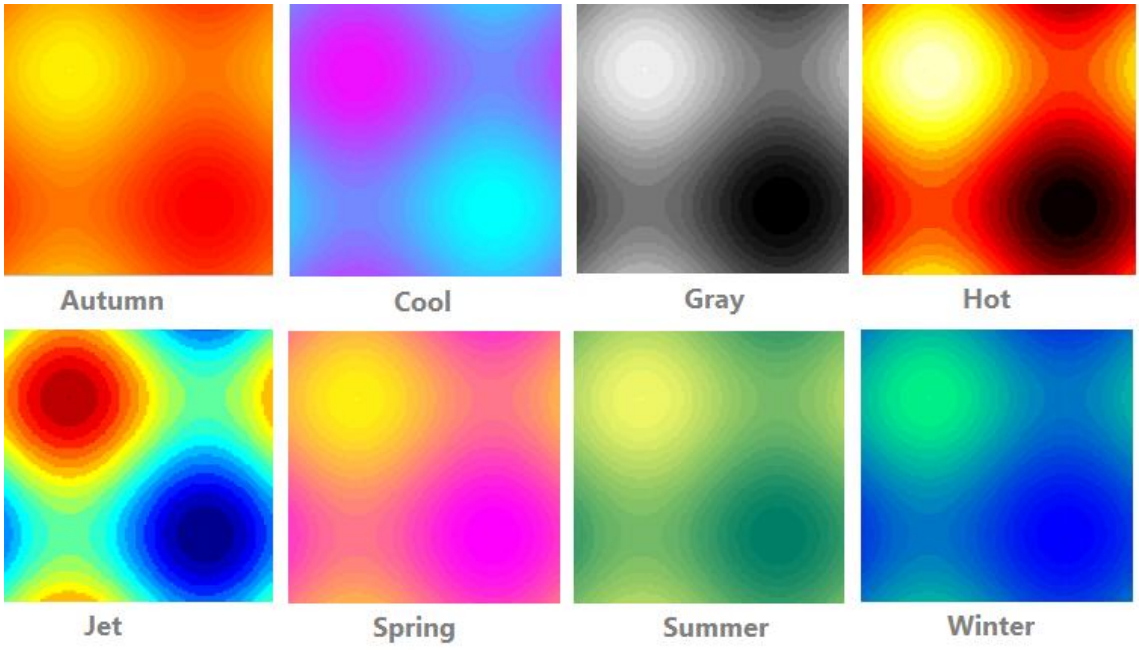
6.3.4 Teplotní mapa

Pro znázornění dvourozměrných matic celých a desetinných čísel byla vybrána teplotní mapa jako implicitní grafické znázornění. Teplotní mapu jsem implementoval jako potomka třídy `UserControl`, která slouží jako základová třída pro vytváření uživatelských komponent bez nutnosti definovat vizuální styl prvků pomocí šablon. `UserControl` využívá konceptu mísení XAML s procedurálním jazykem (Code-Behind) stejným způsobem, jako probíhá návrh hlavního okna aplikace.

Pro reprezentaci teplotní mapy je použit kontejner `Canvas`, na který se v závislosti na změnách vlastnosti `DataContext` dynamicky generují po celé ploše obdélníky, jejichž barva reprezentuje hodnotu veličiny v dané buňce matice.

Barvy jsou generovány na základě povědomí o největším a nejmenším prvku matice a dané barevné škále zpřístupněné vlastností `ColormapStyle` (viz Obr. 20).

Pro generování barvy na základě číselné hodnoty v třídě `Colormap` bylo postupováno podobně, jak je popsáno v knize *Practical WPF Charts and Graphics* v kapitole *Custom Colormap and shading* (Xu, 2009).



Obr. 20: Barevné škály teplotních map

6.4 Detailní informace o vyzvednutých datech a nastavení zobrazení

Jak bylo popsáno v kapitole 3.3, detailní informace o vyzvednutých položkách a nastavení jednotlivých zobrazení (viz kapitola 6.3) jsou implementována ve formě záložek (`TabControl`) – **Zobrazení**, **Informace**, **Surové hodnoty**.

6.4.1 Zobrazení

Záložka Zobrazení obsahuje dva prvky. Prvním prvkem je výběrové pole (`ComboBox`), které zpřístupňuje kolekci jednotlivých zobrazení vhodných pro vyzvednutá data (viz kapitola 6.3.1). Kolekce je generována pomocí konvertoru (`DataToViewTypeConverter`), který ji vytváří na základě Hodnotových a Datových typů všech vyzvednutých položek (viz Výpis programu 5).

```
public object Convert(object value, Type targetType, object parameter,CultureInfo culture)
{
    var dataCollection = value as StorageDataValueInfo[];
    //seznam všech možných zobrazení pro všechny položky
    List<ViewHelper.ViewType> possibleViews = new List<ViewHelper.ViewType>();

    //Iniciace seznamu možných zobrazení pro všechny položky
    possibleViews.Add(ViewHelper.ViewType.SingleItem);
    possibleViews.Add(ViewHelper.ViewType.StackView);
    ...

    //Projití vyzvednutých dat a určení možných zobrazení dle Hod. a Dat. typu položky
    foreach (var data in dataCollection)
      {
         //seznam možných zobrazení pro konkrétní položku
         List<ViewHelper.ViewType> possible = new List<ViewHelper.ViewType>();
         possible.Add(ViewHelper.ViewType.SingleItem);//vše umíme zobrazit jako SingleItem
         possible.Add(ViewHelper.ViewType.StackView); //vše umíme zobrazit jako StackView
         possible.Add(ViewHelper.ViewType.TabControl);//vše umíme zobrazit jako TabControl
         possible.Add(ViewHelper.ViewType.GridView); //vše umíme zobrazit jako GridView

         switch (data.value.dataValue.DataType)
         {
            case EnumDataValueType.Primitive: break;
            case EnumDataValueType.Point2D: break;
            ...
            case EnumDataValueType.Field: //Field čísel umíme zobrazit jako SnglAxLnChart
              if(ValueType!=EnumDataValueType.Bool && ValueType!=EnumDataValueType.String)
                 possible.Add(ViewHelper.ViewType.SingleAxisLineChart);
              break;
            case EnumDataValueType.List: //List čísel umíme zobrazit jako SnglAxLnChart
              if(ValueType!=EnumDataValueType.Bool && ValueType!=EnumDataValueType.String)
                 possible.Add(ViewHelper.ViewType.SingleAxisLineChart);
              break;
         }
 //ponechá v seznamu všech možných zobrazení je ty, které podporují zobrazení dané položky
         possibleViews = new List<ViewHelper.ViewType>(possibleViews.Intersect(possible));
      }
//vrátí seznam možných zobrazení podporujících všechny položky
    return possibleViews;
}
```

Výpis programu 5: Metoda Convert třídy DataToViewTypeConverter

6.4.2 Informace

Záložka Informace je tvořena položkovým seznamem (`ItemsControl`), jehož položky představují jeden uzel cesty vybrané vyzvednuté datové položky. Každá položka má formu boxu (`GroupBox`), v rámci kterého se nachází obsahový prvek (`ContentControl`), na nějž se stejně jako v kapitole 6.3 aplikují v závislosti na typu dat vybraných ze seznamu jednotlivé šablony

(např. pro `DataValueInfoPatient` viz Výpis programu 6). Tyto šablony následně v textové formě zobrazují všechny InfoGrupy a důležité vlastnosti vyzvednutých dat.

```
<DataTemplate DataType="{x:Type UniDataValuesInfos:DataValueInfoPatient}">
   . . . <!--vynechána definice layout a umisteni prvku -->
   <TextBlock Text="Name: " />
   <TextBlock Text="{Binding Path=Name}" />
   <TextBlock Text="Valid: " />
   <TextBlock Text="{Binding Path=Valid}"  />
   . . . <!--vynechána další textová pole s informacemi -->
   <ItemsControl ItemsSource="{Binding Path=InfoGroupsXML}"> <!—seznam InfoGrup-->
      <ItemsControl.ItemTemplate>
        <DataTemplate DataType="UniDataValuesInfGroups:InfoGroups">
          <GroupBox Header="{Binding Path=Name}">
            <ContentControl Content="{Binding Path=.,
                        Converter={StaticResource InfoGroupsToGridValuesConverter}}" />
          </GroupBox>
        </DataTemplate>
      </ItemsControl.ItemTemplate>
    </ItemsControl>
   <!--stavové tlačítko zobrazující/skrývající další informace -->
   <ToggleButton x:Name="ToggleMoreInfoButton">
   . . . <!--definice stylu stavového tlačítka – styl se mění pro každý stav -->
   </ToggleButton>
   . . . <!--vynechána definice o layoutu skrývané části -->
   <TextBlock Text="DescIdent: />
   <TextBlock Text="{Binding Path=DescIdent}"  />
   . . . <!--vynechána další textová pole s informacemi -->
   <ItemsControl ItemsSource="{Binding Path=Comments.Comments>
     <ItemsControl.ItemTemplate> <!--seznam komentářů-->
       <DataTemplate DataType="UniDataValuesComments:CommentItem">
         <GroupBox Header="{Binding Path=Key}">
           <TextBlock Text="Comment: " />
           <TextBlock Text="{Binding Path=Value />
           . . . <!--vynechána další textová pole s informacemi -->
         </GroupBox>
       </DataTemplate>
     </ItemsControl.ItemTemplate>
   </ItemsControl>
</DataTemplate>
```

Výpis programu 6: DataTemplate pro informace o pacientovi

6.4.3 Surové hodnoty

Záložka Surové hodnoty obsahuje jediný prvek (`RawDataValueControl`), který je potomkem třídy `Control` a který pomocí prvku `ContentPresenter` a datových šablon pro jednotlivé typy vyzvednutých dat (podobně jako `DataValueControl` viz kapitola 6.3) zobrazuje hodnotu dat v textové podobě (viz kapitola 6.3.2).

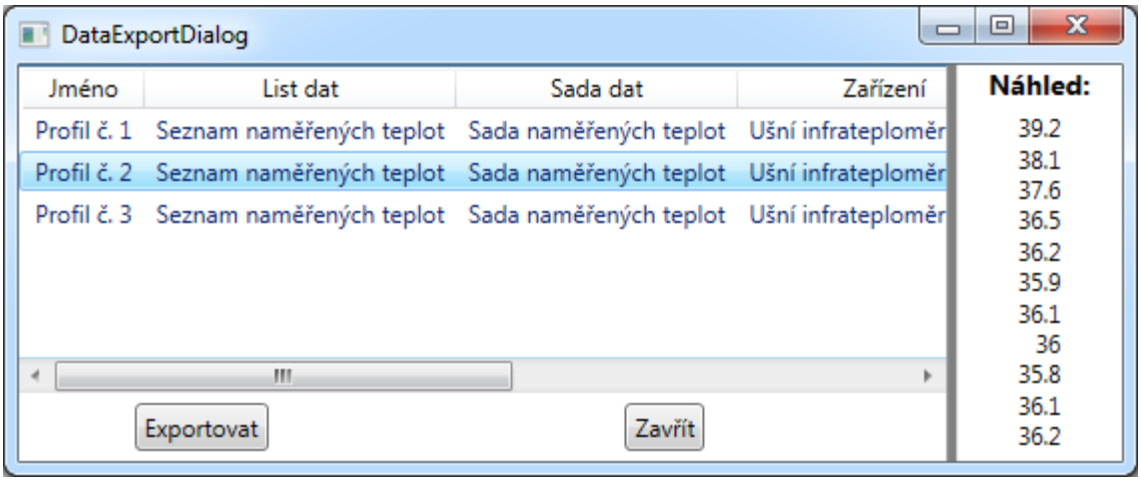
Kapitola 7
Dialog pro export dat

Položky vyzvednuté z úložiště lze pomocí dialogu pro export (viz Obr. 21) uložit do různých formátů. Okno dialog exportu dat je rozdělen na dvě části. Levá část obsahuje seznam s vyzvednutými daty (viz kapitola 3.1) a tlačítka pro export vybraných dat ze seznamu a uzavření dialogu. Pravá část prezentuje náhled na data vybraná v seznamu vyzvednutých dat v textové formě (viz kapitola 3.3.3).

Formát ukládaných dat volí uživatel z nabídky standardního dialogu pro ukládání, který umožňuje exportovanou položku uložit do vhodných formátů generovaných na základě Datového a Hodnotového typu položky.

Podporované formáty jsou:

- **TXT** – textový formát – umožňuje uložit libovolnou datovou položku
- **CSV** – hodnoty oddělené čárkou, tabulková data – umožňuje uložit např. List, 2DMatrix
- **BMP** – bitmapový obrázek – umožňuje uložit např. 2DMatrix
- **PNG –** rastrový obrázek – umožňuje uložit např. 2DMatrix



Obr. 21: Dialog pro export dat

Kapitola 8
Závěr

Cílem této bakalářské práce bylo vytvořit pro projekt Univerzálního úložiště aplikaci, která umožní vizualizaci uložených dat v uživatelsky přívětivé formě a jejich export do některého z formátů vhodných pro strojové zpracování.

Univerzální úložiště slouží pro ukládání, procházení a opětovné získávání špatně strukturovaných dat. Bylo původně navrženo pro lékařské účely s cílem sjednotit ukládání různých typů dat z různých medicínských přístrojů společně s informacemi o těchto přístrojích, jednotlivých pacientech a jejich jednotlivých návštěvách lékaře. Úložiště však umožňuje práci i se zcela obecnými daty.

Po analýze stávajících možností Úložiště, které zprostředkovávaly ukládání a vyzvedávání uložených dat, jsem navrhl několik způsobů grafické vizualizace pro prohlížení dat uživatelem. Neidentifikoval jsem potřebu doplnit pro zobrazování k ukládaným datům další informace.

Pro správnou interpretaci zobrazených dat bylo nutné navrhnout způsob zobrazení dalších informací o datech a jejich struktuře. Navrhl jsem tedy také komponentu umožňující uživateli zobrazit ke každé položce vyzvednuté z Univerzálního úložiště vyčerpávající výčet detailních informací.

V rámci další části bakalářské práce jsem navržené způsoby zobrazení vytvořil jako jednotlivé komponenty aplikace pro zobrazování dat uložených v Univerzálním úložišti. Snahou bylo vytvořit modulární a jednoduše rozšiřitelné řešení, které umožní integraci nových funkcí a zobrazovacích možností. Součástí aplikace je také komponenta umožňující ukládání dat vyzvednutých ze struktury Úložiště do některého z formátů vhodných pro strojové zpracování.

Výsledkem práce je tedy aplikace umožňující uživateli exportovat data a zobrazovat náhledy dat uložených v Univerzálním úložišti. U jednotlivých komponent aplikace jsem kladl důraz na intuitivní ovládání, srozumitelnost a přehlednost. Komponenty řeší elegantním způsobem volbu správného zobrazení v závislosti na typu vyzvednutých dat a jsou navrženy s ohledem na možné samostatné použití v rámci další práce na projektu Univerzálního úložiště.

Reference a citace

.NET Framework Reference Source. [Online] http://referencesource.microsoft.com/.

Dorman, Scott. 2010. *Sams Teach Yourself Visual C# 2010 in 24 Hours: Complete Starter Kit.* Indianapolis : Sams Publishing, 2010.

Dubec, Martin. 2009. Univerzálna databáza pre lekárske data. *Bakalářská Práce.* 2009.

Glassner, Andrew S. 1993. *Graphics Gems.* Orlando : Academic Press, Inc. , 1993. 0122861655.

Malaník, Jakub. 2013. Vzájemné využití MATLAB a Microsoft .NET Framework (C#). *Bakalářská práce.* 2013.

Nathan, Adam. 2010. *WPF Unleashed.* Indiana : Sams publishing, 2010.

Novák, Petr. 2014. Možnosti zobrazení časových průběhů pro visuální analýzu. *Nature Inspired Technologies Group.* [Online] 2014. https://nit.felk.cvut.cz/drupal/moznostizobrazeni.

Novák, Petr. 2014. Univerzální úložiště nejen pro lékařská data. *Nature Inspired Technologies Group.* [Online] 2014. https://nit.felk.cvut.cz/drupal/univerzalniuloziste.

Podila, Pavan a Hoffman, Kevin Scott. 2009. *WPF Control Development Unleashed: Building Advanced User Experiences.* Indianapolis : Sams Publishing, 2009. 0768695481.

Virius, Miroslav. 2011. *Programování pro .NET.* Praha : České vysoké učení technické, 2011. 978-80-01-04864-1.

Xu, Jack. 2009. *Practical WPF Charts and Graphics.* New York : Apress, 2009.

A. Seznam použitých zkratek a symbolů

2D – dvourozměrný

3D – trojrozměrný

BMP – bitmapový obrázek

C# – vysokoúrovňový objektově orientovaný programovací jazyk

CSV – Comma-separated values

PNG – Portable Network Graphics

TXT – textový soubor

RGB – barevný model Red, Green, Blue

WPF – Windows Presentation Foundation

XAML – Extensible Application Markup Language

XML – Extensible Markup Language

B. Přehled datových typů používaných v Univerzálním úložišti

Datové typy používané v Univerzálním úložišti jsou vybudované na základech hluboké dědičnosti. Základní třída jakéhokoli datového typu je `DataValueBase`, kterou následně rozšiřuje `DataValueExtBase`. Přehled jejich nejdůležitějších vlastností je vypsán v následující tabulce Tab. 1.

`DataValueBase`	`Name – unikátní pojmenování`
	`ValueType – hodnotový typ`
	`DataType – datový typ`
`DataValueExtBase`	`DateTime – datum spojené s hodnotou`
	`Valid – platnost hodnoty`
	`Comments - poznámky`

Tab. 1: Přehled základních tříd s nejdůležitějšími vlastnostmi

Datové typy se dále dají rozdělit na typy **hodnotové** nacházející se ve jmenném prostoru `UniversalDataV3.Values`, **informační** nacházející se v `UniversalDataV3.ValuesInfos`, **seznamové** v `UniversalDataV3.ValuesItems`, **grafické** v `UniversalDataV3.ValuesGraphics` a typy **ostatní** v `UniversalDataV3.ValuesOthers`.

Hodnotové typy

Základní hodnotové datové typy (viz Tab. 2) dědí přímo od `DataValueExtBase`. Hodnotové typy představují listy na nejnižší úrovni stromu uložených hodnot a jsou tedy určeny k uchovávání hodnoty – jedné nebo i více - samotného měření. Konečný typ - třída, kterou jsou reprezentována samotná data - je odvozena od jednotlivých hodnotových typů na principu generické dědičnosti podle daného typu uchovávané hodnoty (řetězec znaků, procento, různé reprezentace celých a desetinných čísel).

To znamená, že na uchování celého čísla typu `UInt32` použijeme třídu `DataValuePrimitiveBase<UInt32>`, resp. potomka této třídy `DataValuePrimitiveBaseUInt32`, který jí de facto jen přejmenovává. Obdobný princip funguje i pro všechny ostatní datové typy – například existují třídy `DataValuePrimitiveBool`, `DataValuePrimitiveInt64`,

DataValuePoint2DDouble, DataValuePoint3DByte, DataValueMatrix2DUInt16, DataValueMatrix3DSingle, DataValueFieldInt8, DataValueListString a jiné.

DataValuePrimitiveBase	Value – jednoduchá hodnota
DataValuePoint2DBase	ValueX – hodnota X-ové souřadnice bodu
	ValueY – hodnota Y-ové souřadnice bodu
DataValuePoint3DBase	ValueX – hodnota X-ové souřadnice bodu
	ValueY – hodnota Y-ové souřadnice bodu
	ValueZ – hodnota Z-ové souřadnice bodu
DataValueMatrix2DBase	Values – dvojrozměrné pole hodnot
	SizeX – počet sloupců matice
	SizeY – počet řádků matice
DataValueMatrix3DBase	Values – trojrozměrné pole hodnot
	SizeX – velikost matice podle X-ové osy
	SizeY – velikost matice podle Y-ové osy
	SizeZ – velikost matice podle Z-ové osy
DataValueFieldBase	Values – pole hodnot
	ValueCount – počet hodnot v poli
DataValueListBase	Values – seznam hodnot
	ValueCount – počet hodnot v seznamu
DataValueItemsAnyValueBase	ValueCount – počet hodnot v seznamu hodnot

Tab. 2: Přehled Hodnotových tříd (UniversalDataV3.Values) s nejdůležitějšími vlastnostmi

Informační typy

Informační typy představují vnitřní uzly stromu uložených hodnot v struktuře Univerzálního úložiště. Jak bylo popsáno v Kapitole 2, tak každý vnitřní uzel má jasně definovaný typ odvíjející se od patra stromu, ve kterém se nachází, a libovolný počet potomků typu odpovídajícímu následujícímu patru stromu.

Pro danou databázi lékařských dat tyto typy odpovídají postupně patrům pacient, návštěva, měření, zařízení, sada dat a seznam dat. Jednotlivé Informační typy (DataValueInfoPatient, DataValueInfoVisit, DataValueInfoMeasure, DataValueInfoDevice, DataValueInfoSet, DataValueInfoList) jsou pak odvozeny na principu generické dědičnosti od třídy DataValueInfoHighBase (viz Tab. 3), která je nepřímým potomkem DataValueExtBase.

DataValueInfoHighBase	InfoGroups – seznam infogrup
	Relations – seznam závislostí
	Items – seznam podpoložek

Tab. 3: Přehled Informačních typů (UniversalDataV3.ValuesInfos) s nejdůležitějšími vlastnostmi

Seznamové typy

Seznamové typy (viz Tab. 4) ukládají do seznamu vždy dvojici hodnot. Tato dvojice hodnot je reprezentována některým z položkových typů (viz Tab. 5). Každý položkový typ obsahuje právě jednu hodnotu některého z vybraných Hodnotových typů spolu s určitou položkou (datum, jméno nebo čas).

Konečný typ, který ukládá samotná data, je podobně jako u Hodnotových typů vytvořen na principu generické dědičnosti podle typu uchovávané hodnoty s následným přejmenováním. Příklady tříd pak jsou:
DataValueItemsTimeSpanPrimitiveString
DataValueItemsTimeSpanPoint3DByte
DataValueItemsNamePrimitiveDouble
DataValueItemsDateTimePrimitiveInt64.
Seznamové typy dědí od ItemAnyValueBase, která neobsahuje žádné užitečné informace a není ani potomkem žádné jiné třídy.

DataValueItemsDateTimeValueBase	Values – seznam hodnot ItemDateTimeValueBase
DataValueItemsNameValueBase	Values – seznam hodnot ItemNameValueBase
DataValueItemsTimeSpanValueBase	Values – seznam hodnot ItemTimeSpanValueBase

Tab. 4: Přehled Seznamových typů (UniversalDataV3.ValuesItems) s nejdůležitějšími vlastnostmi

ItemDateTimeValueBase	DateTime – datum a čas
	Value - hodnota
ItemNameValueBase	Name - jméno
	Value - hodnota
ItemTimeSpanValueBase	Time – doba trvání
	Value - hodnota

Tab. 5: Přehled položkových typů s nejdůležitějšími vlastnostmi

Grafické typy

Grafické typy (viz Tab. 6) reprezentují vždy barvu nebo grafický objekt (čáru, obdélník). Úkolem těchto typů je především demonstrovat možnosti Úložiště. Grafické typy jsou potomky `DataValueGraphicsBase`, která dědí od `DataValueExtBase`.

`DataValueGraphicsColor`	`Red` – červená složka barvy
	`Green` – zelená složka barvy
	`Blue` – modrá složka barvy
`DataValueGraphicsLines`	`Points` – body
	`Count` – počet bodů
`DataValueGraphicsRectangle`	`Left` – velikost odsazení od levého okraje
	`Top` – velikost odsazení od horního okraje
	`Width` - šířka
	`Height` - výška
`DataValueImageBytes`	`Value` – bajtové pole obrazu
	`BytesSize` – délka bajtového pole
`DataValueBitmapImage`	`Value` – bitmapový obraz

Tab. 6: Přehled grafických typů s nejdůležitějšími vlastnostmi

Ostatní typy

Ostatní typy (viz Tab. 7) představují hodnoty, které nemohly být reprezentovány žádným z předcházejících typů. Tyto typy stejně jako grafické typy mají za úkol nastínit možnosti Úložiště pracovat s nejrůznějšími formáty dat. Všechny Ostatní typy jsou přímými potomky `DataValueExtBase`.

`DataValueVersion`	`Major – hlavní verze`
	`Minor – vedlejší verze`
	`Build – sestavení`
	`Revision – revize`
`DataValueDeviceDesc`	`Manuf – jméno výrobce`
	`Product – název produktu`
	`Model – název modelu`
	`ResX – rozlišení X-ové osy`
	`ResY – rozlišení Y-oné osy`
	`SizeX – X-ová velikost`
	`SizeY – Y-ová velikost`

Tab. 7: Přehled ostatních typů s nejdůležitějšími vlastnostmi

C. Obsah doprovodného CD

CD / Application – spustitelná verze aplikace

CD / Application / USTestData – testovací data

CD / Source – zdrojové kódy

CD / Sources / Controls – elementy UI

CD / Source / Controls / HeatMap – teplotní mapa

CD / Source / Controls / LineChart – čárový graf

CD / Source / Resources – pomocné prostředky

CD / Source / Resources / Converters - konvertory

CD / Source / Resources / Helpers – pomocné třídy

CD / Source / Themes – defaultní šablony

CD / Text – text práce