VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

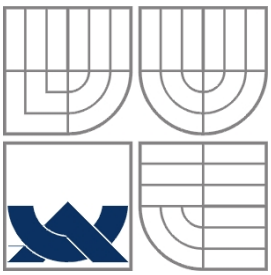
WEBOVÉ ROZHRANÍ PRO SPRÁVU DATABÁZE CACHÉ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

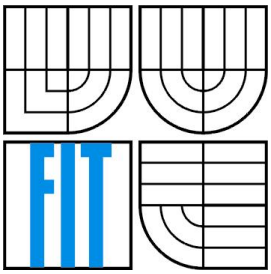
AUTOR PRÁCE
AUTHOR

PAVEL VALENA

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÉ ROZHRANÍ PRO SPRÁVU DATABÁZE CACHÉ

WEB ADMINISTRATION INTERFACE FOR CACHÉ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PAVEL VALENA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK BURGET, Ph.D.

BRNO 2015

Abstrakt

Úkolem této bakalářské práce je vytvořit webové rozhraní pro správu databáze Caché, které bude umožňovat přehledně a logicky zobrazovat obsah databáze, provádět změny v databázi, správu uživatelů a databází. Práce obsahuje popis návrhu a implementace tohoto webového rozhraní.

Abstract

The goal of this bachelor's thesis is to create web administration interface for Caché, where database data is clearly and logically displayed. Enabling user to edit database data, manage users and databases. The thesis describes the design and implementation process of this web interface.

Klíčová slova

Webové rozhraní pro správu databáze, globály, správa uživatelů, správa databáze, Caché, InterSystems, objektová databáze, postrelační databáze, ObjectScript

Keywords

Web administration interface, globals, user administration, database administration, Caché, InterSystems, object database, postrelational database, ObjectScript

Citace

Pavel Valena: Webové rozhraní pro správu databáze Caché, bakalářská práce, Brno, FIT VUT v Brně, 2015

Webové rozhraní pro správu databáze Caché

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Radka Burgeta, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Valena
27. května 2015

Poděkování

Děkuji svému vedoucímu Ing. Radku Burgetovi, Ph.D. za odborné vedení, vstřícnost a podněty, které mi při řešení tohoto projektu poskytl.

Obsah

1 Úvod

V době, kdy ve světě převládají relační databáze s neustále se zvětšujícím množstvím dat, s rostoucími rozměry tabulek a narůstajícími počty a vztahy mezi nimi, je práce s daty často komplikovaná. Databáze jsou nepřehledné, náročné na výkon a logika nakládání s daty je převážně či úplně od dat separována.

Databáze postrelačního typu čelí těmto problémům například objektovým přístupem. Práce s objekty je pro řadu aplikací přirozenější, data jsou logicky uspořádané a ve spolupráci s objektovými jazyky se objekty v databázi mohou takřka ztotožňovat s objekty v aplikacích. Pro mnoho použití představuje objektová databáze také zrychlení aplikace.

Správa objektové databáze je nedílnou součástí vývoje i údržby aplikací, které ji využívají. Data je potřeba jednoduše a logicky zobrazovat a manipulovat s nimi. Nedílnou součástí je i správa uživatelů a databází.

Nepřehlédnutelným zástupcem objektových databází je *Caché* od firmy *InterSystems*. V rámci dostupných webových nástrojů pro správu databáze *Caché* jsou data zobrazována značně nepřehledně – serializovaná, v jednom řádku a se značnou redundancí. Zcela chybí zobrazení objektové struktury dat. Je tedy zapotřebí nástroje nového, který by umožňoval přehledně, logicky a strukturovaně prohlížet a spravovat data v databázi *Caché*. Tento text se zaměřuje na návrh a vývoj webového rozhraní pro správu databáze *Caché*.

Čtenář bude nejprve stručně seznámen s prostředím databázového a aplikačního serveru *Caché* v kapitole druhé, a následně jsou v kapitole třetí popsány zvolené technologie pro tvorbu této webové aplikace.

Čtvrtá kapitola je věnována návrhu webového rozhraní, které si klade za cíl i možnost snadného rozšiřování funkcionality, možnost přizpůsobení vzhledu a v neposlední řadě modularitu – oddělením serverové a klientské části aplikace, což umožňuje např. použití některých funkcí serverové části jinými aplikacemi.

V páté kapitole popisuji zajímavé aspekty implementace daného webového rozhraní spolu se specifickými problémy, kterým jsem při implementaci čelil pro zachování flexibility a komfortnosti navrženého uživatelského rozhraní.

V šesté kapitole je porovnáno vytvořené webové rozhraní s vestavěným a testováno na přenositelnost mezi různými webovými prohlížeči.

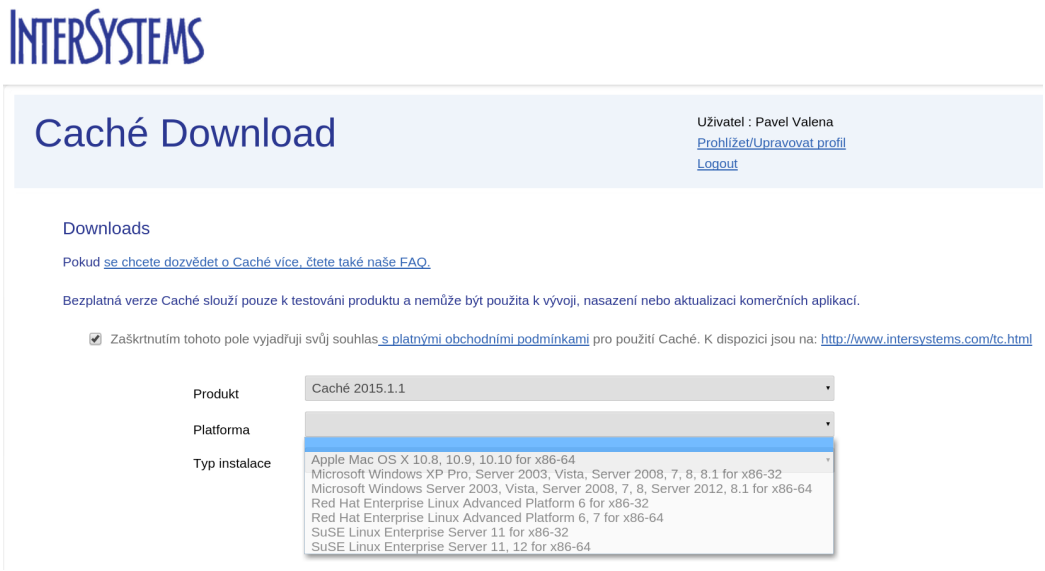
Sedmá kapitola patří shrnutí výsledků a možnostem dalších rozšíření.

2 Databázový a aplikační server Caché

Caché je komerční produkt, který je vyvíjen již od roku 1996. Je prezentován svojí rychlostí, škálovatelností a stabilitou [1]. Jedná se o poměrně obsáhlý balík aplikací, který obsahuje např. i nástavby nad *Caché*, jako je například *Zen* framework /knihovna/ (dále jen *Zen*), který je určený pro tvorbu webových aplikací. U *Caché* nalezneme také mnohá specifika, jako je např. jazyk *Caché ObjectScript*, který se používá pro tvorbu serverových aplikací či knihoven.

2.1 Obsah instalace

Caché je možné provozovat na různých platformách.



Obrázek 1: Podporované platformy

Podporuje i platformy typu UNIX, vyžaduje však *RHEL (Red Hat Enterprise Linux*) nebo *SUSE Linux Enterprise Server*. Přenést pod jinou Linuxovou distribuci (na bázi *Debian GNU/Linux*) se mi instalaci bohužel nepodařilo. Pro vývoj webového rozhraní jsem tedy zvolil platformu Windows.

Instalace obsahuje několik nástrojů, především pro vývoj aplikací, správu a monitorování databáze a aplikačního serveru *Caché*. Klíčovým nástrojem je *Caché Studio* (dále jen *Studio*) – vývojové prostředí určené pro vývoj různých aplikací v prostředí *Caché*, které zahrnuje i možnost procházení zdrojových kódů tříd, včetně vestavěných. Při psaní kódu nabízí k automatickému doplnění názvy metod a tříd, ale i názvy parametrů a jejich typy, a to jak vlastního, tak vestavěného kódu.

K dispozici je dále webový server *Apache* (lze použít i jiného webového serveru [10]), na němž je provozován mimo uživatelských aplikací i *Management portál* (dále jen *Portál*) – více viz následující podkapitoly. Prostředí *Caché* je také otevřeno práci na příkazové řádce.

Databázový server Caché lze propojit /zapouzdřit/ s aplikacemi za pomocí JDBC, ODBC, tříd C++ a dalších [7].

2.2 Webové aplikace Caché

Logiku serverových aplikací, ale i samotné tvoření /generování/ obsahu webové aplikace lze provádět v rámci různých tříd na serveru. Lze využít čistě objektový přístup pro vytváření webových aplikací použitím *CLS*, které tvoří *HTML* za pomocí zabudovaných objektů. Alternativou jsou stránky *CSP* (*Caché Server Pages*) [15], které umožňují kombinovat dynamickou serverovou a statickou klientskou část (resp. dané části prokládat).

Zen umožňuje vytváření interaktivních webových aplikací. Z mého pohledu je však vhodnější pro rozsáhlejší aplikace. Také značně vymezuje stavbu /strukturu/ aplikace, tedy zamezuje plné flexibilitě.

Více o webových technologiích *Caché* ve 3. kapitole.

2.3 Aplikace a databáze

Caché umožňuje psaní serverových aplikací v některém z vlastních jazyků. Jde o nezávisle existující jazyky rozšířené o přístup k databázi. Jazyk *ObjectScript* přistupuje přímo ke *Globálům* – základní jednotce dat v databázi. Syntakticky se *globály* v jazyce *ObjectScript* odlišují pouze s přidaným znakem **^** před proměnou [2].

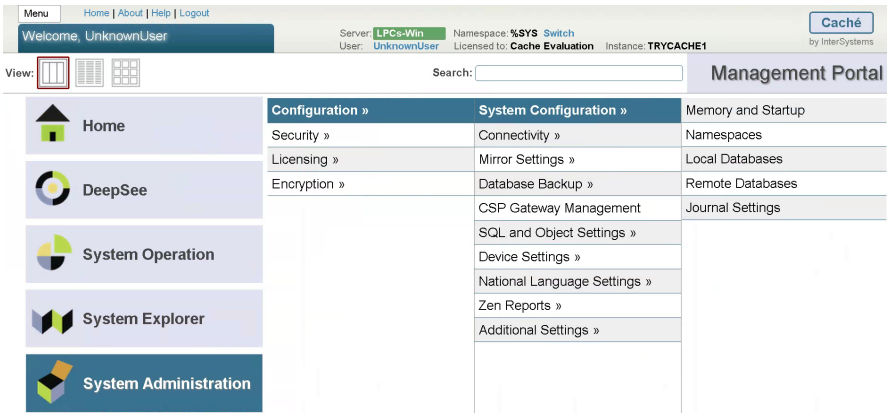
K obsahu databáze, konfiguraci a dalším lze přistupovat za použití instancí specializovaných tříd, jako je například `%ResultSet`, které načte uloženou proceduru (viz níže) třídy, která požadovanou informací disponuje a po provedení /execute/ lze procházet výsledky dotazu.

Provádět změny v konfiguraci lze za pomocí vestavěných perzistentních objektů (nebo třídních metod), které změny v konfigurací zapouzdřují logikou. Tato logika zajišťuje, že budou prováděny pouze korektní změny.

Aplikace mohou mimo čistě objektový přístup k databázi využít i zabudovaného rozšířeného jazyka *SQL* ihned v několika verzích – *Embedded, Dynamic,* a *Stored Procedure*. První je na bázi maker, druhá využívá instancí objektů a poslední z nich ukládá SQL v kompilovaném objektu.

2.4 Nástroje pro správu Caché

Vzhledem k vývoji webové aplikace jsem se zaměřil na nástroje, které jsou k dispozici z *Portálu*.



Obrázek 2: Management Portál

Je zde plnohodnotná správa uživatelů, databází, monitorovací nástroje, správa zálohování, zrcadlení serverů a další. Disponuje také správou g*lobálů*, která je jen zcela základní.

Více v 6. kapitole.

2.5 Zdroje informací

Dokumentace, dostupná z *Portálu* nebo online [2], je značně rozsáhlá – disponuje i návody, rejstříky, referencemi vestavěných tříd i různými příklady a výukovými materiály.

Bohužel se velice obtížně dohledávají požadované funkce, makra či třídy, za pomocí kterých lze dosáhnout požadovaného chování (více viz implementace). Dokumentace je však úplná, takže není nutné čerpat potřebné informace v rámci *Caché* z jiných zdrojů.

3 Webové technologie

Rapidní rozvoj webových technologií umožňuje psát aplikace různých, zcela odlišných typů. Tyto, uvažujeme vztah klient – server, si porovnáme v úvodních kapitolách. Dále si přiblížíme technologie, kterými disponuje *Caché* pro tvorbu webových stránek, a to část klientskou i serverovou.

3.1 Kombinace serveru a klienta

Aplikace, u kterých je klient úzce provázaný s jeho serverovou částí, využívají v naprosté většině případů různé knihovny a frameworky, obdobně jako *Caché Zen.* Tyto však někdy vývoj aplikace prodlužují, zvláště pokud jde o atypický či specializovaný projekt, nebo dokonce i značně komplikují, pokud například má zvolený framework pevně danou konstrukci a zároveň nedisponuje požadovanou funkcionalitou.

U značného množství těchto frameworků a především knihoven je výhodou, že jsou obecné a snadno použitelné. Mají tedy širokou škálu uplatnění, nikoliv však pro specializované aplikace. Pokud se tedy pokusíme o vývoj aplikace za použití frameworku, musíme počítat s dlouhým procesem výběru vhodného z nich.

3.2 Serverové aplikace

Aplikace, u nichž je veškerá logika, či její převážná část, prováděna na serveru, mají také znatelné zastoupení. Jednak z historických důvodů (u dlouho vyvíjených, nebo stále udržovaných aplikací), protože většina klientských strojů neměla dostatečný výkon, ale také pro řadu dalších výhod.

Mezi tyto výhody patří například centralizovaná logika aplikace a z hlediska kompatibility podpora většího počtu webových klientů – starších, či bez podpory, nebo s různou implementací využitých prostředků webového prohlížeče (dále jen *prohlížeč*). Není také zapotřebí instalace dodatečných (převážně grafických) komponent *prohlížeče*.

Nalézají však hojné uplatnění v případech, kdy je zapotřebí mít chráněnou aplikační logiku, ke které je potřeba (zvídavému) uživateli zamezit přístup, nebo kdy je serverová logika nepostradatelná – provádí se mnoho operací s databází aj..

U většiny webových aplikací však není tento přístup vhodný, protože s počtem uživatelů značně rostou nároky na výkon serveru. Pro provedení každé operace /činnosti uživatele/ se čeká na odezvu serveru, a v důsledku toho je aplikace málo interaktivní.

3.3 Klientské aplikace

Jde o aplikace vyvíjené v prostředí webového klienta, kód (aplikační logika) je prováděna na klientovi – přímo u uživatele. V porovnání se serverovým přístupem, kdy prostředí může být značně

přizpůsobené /na míru/ a aplikace může využívat i specializovaných prostředků hostitelského prostředí, je klientská aplikace závislá (vyžaduje) prostředky, které jsou dostupné v libovolném (kompatibilním) prostředí.

Výhodou je, že aplikace může využívat velice interaktivní nástroje pro zobrazení grafického uživatelského rozhraní (dále jen *GUI*), jako je např. Adobe Flash Player nebo HTML5 v kombinaci s *JavaScriptem*, pro vytvoření poutavého rozhraní. Oproti klasickým metodám (viz další odstavec) jsou však náročnější na vývoj.

Dosáhnout značné interaktivity lze ale i použitím dostupných prostředků *XHTML* (Extensible Hypertext Markup Language; jde o přísnější, přeformulovaný, standard HTML [8]), které umožňuje pracovat se stromem *DOM* [3], použitím *JavaScriptu*, a dynamicky tak měnit obsah webové stránky (dále jen *dokument)*. Můžeme tak s menší náročností vyvíjet interaktivní a plnohodnotné specializované aplikace.

Dále za pomocí objektu *XMLHttpRequest* [4], opět v rámci *JavaScriptu*, lze uskutečnit HTTP [9] komunikaci se serverem bez nutnosti načíst znovu celý obsah *dokumentu.* Odesíláme dotazy /příkazy/ na serverovou část aplikace, a to typu HTTP *GET* či *POST*. Server může následně vypisovat /vracet/ klientovi strukturovaná data např. v *XML* nebo *JSON*, která mohou být navíc nezávislá na implementaci klienta.

Použitím výše uvedených technologií (především v předešlých 2 odstavcích) můžeme vyvíjet takzvané „tlusté“ klientské aplikace [5]. Výhodou takové aplikace je například větší flexibilita a nižší zatížení serveru (především oproti implementaci logiky na serveru, viz 3.2).

I na v klientské části aplikace lze pro implementaci použít řadu knihoven. Mezi ně patří např. jQuery, Prototype a underscore.js.

3.4 Serverová část – Caché

Server *Caché* poskytuje stránky *CSP* nebo *CLS* pro implementaci serverové části webové aplikace. Využívá přitom nejvíce jazyka *Caché ObjectScript*, který přistupuje k uloženým třídám, objektům nebo přímo do databáze. Těmto podstatným, a pro implementaci webového rozhraní serveru *Caché* nepostradatelným technologiím jsou věnovány následující podkapitoly.

3.4.1 Jazyk ObjectScript

ObjectScript je nadmnožinou jazyka *MUMPS*, který je zároveň i ANSI standardem, jenž *Caché* ve svém jádru implementuje [6]. Jazyk umožňuje přímý přístup k obsahu databáze, ale umožňuje provádět i pokročilé serverové operace za pomoci tříd a objektů.

Jazyk *ObjectScript* užívá následující, poměrně atypické, notace [11]:

- všechny příkazy jsou uvozeny **`set`** nebo **`do`**
- jména systémových /vestavěných/ tříd jsou uvozeny **`%`**
- vestavěné funkce jsou uvozeny **`$`**
- makra jsou uvozeny **`$$$`**

- speciální příkazy jsou uvozeny **##**, z nižch podstatný je **##class(**třída**)**, sloužící pro invokaci (volání) třídy
- **.**název (tečka) – předá proměnou zvolené metodě nebo funkci skrze referenci
- **..**název (dvě tečky) – odkazuje na instanční proměnou (uvnitř třídy)

Pro naprostou většinu klíčových slov a funkcí má navíc zkrácenou notaci, např:

- **s – set**
- **d – do**
- **$lb – $listbuild**

Poslední zmíněná položka (`$lb`) vytvoří *seznam*, který např. můžeme přiřadit k proměnné, nebo přímo uložit v databázi. *Seznam* umožňuje seskupení více položek (např. i dalších seznamů), ke kterým lze následně přistupovat pomocí `$LIST`.

Pokud nevíme, zdali se v proměnné nachází *seznam*, můžeme použít `$LISTVALID`, který vrací hodnotu `True`, pokud se jedná o *seznam*. Je však potřeba zvlášť ošetřit prázdný řetězec, jelikož ten je také vyhodnocen jako validní *seznam* [17].

3.4.2 Třídy

Caché obsahuje řadu vestavěných tříd i jejich typů, skrze které lze přistupovat do databáze k datům i konfiguraci. Vestavěné třídy jsou stavěné hierarchicky, odvozují od tříd základních většinou abstraktních. Řada tříd je specializovaných pro konkrétní účel (pro přístup přes terminál, webové rozhraní, *Studio*, aj.). Ke konfiguraci či datům tak lze přistupovat skrze rozhraní několika tříd.

Caché umožňuje přístup k těmto třídám zcela shodně přes libovolné komunikační rozhraní. Tvoří tak vlastní *API*, čímž zajišťuje dosažení obdobných výsledků při použití libovolného nástroje (např. pro konfiguraci). Konfigurace samotná je uložená spolu s daty mezi *globály*. Třídy ji tedy zapouzdřují logikou tak, aby docházelo pouze ke korektním úpravám.

3.4.3 Caché Server Pages

Neboli *CSP* [15]. Jedná se o propojení statického *dokumentu* s logikou prováděnou na serveru (jako je tomu např. u PHP [12]). Obsah lze dynamicky generovat při každém vyžádání *dokumentu*, např. spolu s odesláním hlavičky *HTTP GET*. Alternativou *CSP* jsou *CLS*. *CLS* je třída reprezentující a generující *CSP* stránku, viz ukázka:

```
Class WebStranky.Uvod Extends %CSP.Page { … }
```

Tato třída potom pomocí zděděných a nově vytvořených metod vykresluje obsah stránky. *CSP* na rozdíl od *CLS* může obsahovat pouze statický *dokument*.

CSP umožňuje několik způsobů vytváření obsahu do *dokumentu*. Jednak v rámci elementu *script*, viz následující ukázka:

```
<script language="cache" method="OnPreHTTP" arguments="" returntype="%Boolean"> … </script>
```

Nebo druhým způsobem zápisu:

```
#( … )#
```

V obou případech se dosadí namísto tří teček kód *ObjectScriptu*. V případě prvním se kód, v důsledku uvedeného `method="OnPreHTTP"`, bude provádět ještě před zahájením odpovědi *HTTP*, a uživatel tak např. místo zobrazení stránky může být přesměrován na jinou.

3.4.4 Jmenné prostory

Namespace, neboli jmenný prostor, je virtuální prostředí oddělující od sebe logicky nesouvisející data (jako jsou např. *třídy* a *globály*) a zároveň slučující databáze, které mohou být lokalizovány v jiných fyzických umístěních [13].

3.4.5 Globály

Jazyk *ObjectScript* umožňuje přímo v kódu, s drobnou změnou – přidaným znakem **^** před proměnou, pracovat totožně s *globály*, které jsou uloženy v databázi, jako s běžnými proměnnými. *Globály* ovšem nejsou pouze ve vztahu 1:1, tedy 1 proměnná : 1 hodnota.

Globály mohou mít mnoho *rozměrů* (maximum se mi bohužel nepodařilo dohledat). Ukažme si na příkladu, co je myšleno *rozměrem globálu*:

```
s ^MujGlobal(1, 2, 3, 4, 5) = "hodnota"
```

Čísla 1 až 5 jsou zde rozměry *globálu*. V databázi jsou řazeny veškeré *globály* podle názvu a následně rozměru od levého po pravý. Namísto čísel by se samozřejmě daly použít řetězce, nebo kombinace čísel a řetězců.

Hodnotou může být kromě základních typů také odkaz na objekt, či objekt samotný. Jde o tzv. *Swizzling* [18], což je automatické načítání perzistentních objektů z databáze v případě, že je vyžadovaný objekt referencován z jiného objektu.

3.4.6 Zabezpečení

Caché má zabudovaný systém uživatelů a rolí. Včetně jejich přihlašování na webové rozhraní, správy přístupu k datům a odhlášení po uplynutí sezení. Není tedy třeba implementovat žádné dodatečné mechanismy pro ochranu dat, či autorizaci přístupu.

4 Návrh webového rozhraní

Cílem je interaktivní, flexibilní webové rozhraní s oddělenou serverovou a klientskou částí, podporou snadného rozšiřování funkcionality a možností přizpůsobení vzhledu. V první řadě jde však o vytvoření rozhraní, které bude umožňovat přehledně a logicky prohlížet a spravovat *globály*, ale také i uživatele a databáze.

Pro návrh tohoto rozhraní jsem zvolil logiku aplikace provádět na klientské části vytvořením tzv. „tlustého“ klienta (viz 3.3). Klientskou část jsem implementačně oddělil od serverové, která je minimální. Serverová část provádí pouze příkazy zaslané /vyžádané/ klientem (dále jen *akce*), a vypisuje /navrací/ zpět vyžádaná data, nebo pouze stav skrze strukturovaná data.

Klient následně data získaná *akcí* interpretuje (resp. zobrazuje) a zpracovává je pro další odesílání dotazů či příkazů zvolených uživatelem.

Pevně stanovené jsou a*kce*, které serverová část poskytuje /provádí/, spolu s informací, o jaký *typ akce* se jedná. U nastavovaných *hodnot* (viz 4.2.2) je předem známo klientské části aplikace, které položky lze měnit (resp. nastavit), ale pouze ve formě pořadového čísla položky.

Rozšiřování funkcionality lze provádět na klientské i serverové části aplikace nezávisle. Díky minimální implementaci serveru je ve většině případů vytvoření *akce* jen napojením názvu akce na odpovídající třídu v *API Caché*.

V klientské části klienta jsem pro možnost snadného rozšíření funkcionality zvolil generického přístupu, kdy je prováděn tentýž kód nezávisle na konkrétní *akci*.

Možnosti přizpůsobení vzhledu lze dosáhnout oddělením aplikační logiky od samotného *stylování*. Více viz 4.3.3.

4.1 Zvolené technologie

Pro implementaci serverové části aplikace jsem zvolil vytvoření nové třídy *ObjectScriptu*, jejíž činnost bude volána z *CSP*.

Pro implementaci klienta jsem zvolil *XHTML dokument,* jehož obsah bude dynamicky měněn pomocí *JavaScriptu*, který bude se serverovou částí komunikovat pomocí *XMLHttpRequest*. Vzhled bude určovat samostatný soubor *CSS*.

Komunikace probíhá ze strany klientské části pomocí odesílání požadavků v *HTTP GET* a serverová část vždy vrací zpět /vypisuje/ *JSON*.

4.2 Serverová část

Serverová část je zcela oddělená od činnosti klienta, jeho chování i způsobu zpracování (resp. zobrazení) dat. Serverová část je bezstavová, takže každá *akce* musí pro úspěšné provedení obsahovat veškeré potřebné informace – *parametry akce* (více viz 4.2.2).

4.2.1 Typy akcí

Rozlišujeme dva typy *akcí*:

- Datové akce mají klientovi poskytnout data z databáze. Tyto akce by neměly selhat, pokud klientská část aplikace odešle validní vstup, který již předem zná (název *akce*), nebo který získá za chodu (názvy g*lobálů*).
- Příkazové akce mají za cíl provést požadovaný příkaz a sdělit klientské části pouze to, zdali příkaz uspěl či neuspěl. V případě chyby vrací serverová část chybovou hlášku.

Platný název akce je vyžadován vždy. K vybraným *akcím* jsou zapotřebí platné *parametry*, které musí klient uvést pro úspěšné provedení akce.

4.2.2 Parametry

Parametry jsou děleny do několika *polí*:

- Prostor – ve kterém jmenném prostoru manipulovat s daty (požadováno u všech *akcí* s globály)
- Určení – klíčový parametr, který přesně určuje data, která mají být navrácena nebo pozměněna
- Hodnota – obsahuje data, která mají být uložena či nastavena

4.2.3 Datové Akce

Server odpovídá na (resp. provádí) následující *datové akce*:

- Prostory – navrací seznam prostorů
- Globály – navrací seznam globálů
- Data – navrací veškerá data uložená ve všech položkách /rozměrech/ *určeného* globálu
- Uživatelé – navrací seznam uživatelů
- Databáze – navrací seznam databází

4.2.4 Příkazové Akce

Server provádí následující *příkazové akce*:

- Nastav globál – nastaví *hodnotu určeného* globálu; neexistující *globál* či jeho *rozměr* vytvoří
- Smaž globál – smaže *určený* globál
- Pozměň uživatele – nastaví *hodnoty* u *určeného* uživatele
- Smaž uživatele – smaže *určeného* uživatele
- Vytvoř uživatele – vytvoří *určeného* uživatele se zadanými *hodnotami*

- Pozměň databázi – nastaví *hodnoty* u *určené* databáze
- Smaž databázi – smaže *určenou* databázi
- Zkopíruj databázi – zkopíruje *určenou* databázi na novou databázi zadanou *hodnotou*
- Vytvoř databázi – vytvoří *určenou* databázi se zadanými *hodnotami*

4.3 Klientská část

Klientská část je se skládá ze tří podčástí:

- Kostra – obsahuje elementy označené příslušným *ID*, které *Logika* dynamicky mění (resp. mění obsah elementů)
- Logika – kód který je obeznámen s *ID* elementů, *akcemi,* způsoby jejich interpretace aj. (viz 4.3.2)
- Vzhled – styl definující výsledný vzhled aplikace, obeznámen s *ID i* typy elementů a užitými atributy označujícími vzhled

4.3.1 Kostra

Kostra aplikace je členěná do tří hlavních sekcí:

- Hlavní menu – obsahuje navigaci mezi Globály, Uživateli a Databázemi, tak aby bylo možné se mezi nimi přepínat; dále zobrazuje přihlášeného uživatele a možnost odhlášení
- Stromové / výběrové menu – obsahuje strukturovaný (stromový) výčet Uživatelů / Databází / Globálů / Obsahu globálu
- Datový obsah – zobrazuje zvolená data spolu s volbami akcí – tedy pohled na jednotlivé položky

4.3.2 Logika

Řídící logika, která se řídí vstupy od uživatele, a následně volá *akce* /zasílá je/ serverové části. Odpovědi vyhodnocuje a zobrazuje uživateli.

Generuje samotný obsah a formuluje *parametry* a *akce* odesílané na server. Vzhled obsahu je však přizpůsoben dle definice *vzhledu*, protože obsahuje pouze *označení* příslušného *elementu* spolu s jeho typem.

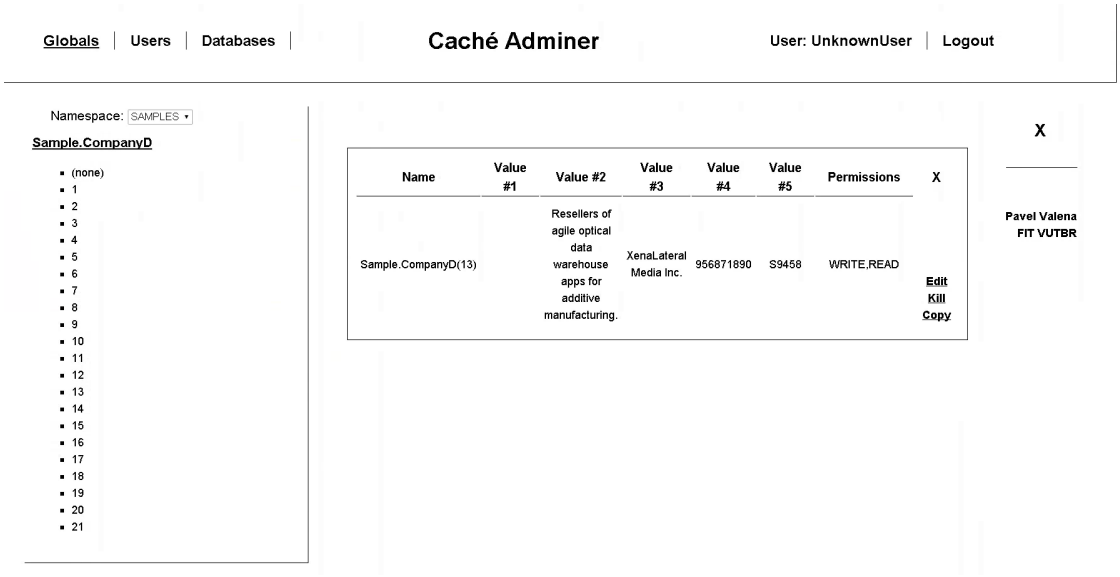
Výběrem v *stromovém menu* uživatel zvolí, které položky se mu zobrazí v *datovém obsahu*. Do *datového obsahu* se položky přidávají a může tak nezávisle na sobě být zobrazen libovolný počet položek.

Každá položka obsahu k sobě obsahuje volby *akce,* které s ní lze provádět.

4.3.3 Vzhled

Vzhled aplikace je vázán na elementy, jejich *ID* a *označení*. Lze tak měnit vzhled aplikace bez zásahu do jiných částí klienta (či serveru). Kód tedy bude pouze obsahovat označení (*class*), o jaký prvek (*element*) se jedná.

4.3.4 Výsledek návrhu



Obrázek 3: Výsledek návrhu

4.3.4.1 Rozvržení GUI

V části horní lze přepnout zobrazení na Globály, Uživatele nebo databáze. Dále v levé části je *stromové menu* pro výběr požadovaného globálu a uprostřed data zvoleného *globálu – seznam o* 5 prvcích – a v pravé části jsou zobrazeny *akce* k tomuto globálu.

V pravé horní části, pod *horním menu*, je dále **X** pro zrušení zobrazení veškerého obsahu, jelikož při výběru dalších položek z menu se data přidají do horní části *datového obsahu* a stávající se posunou pod ně. Lze je také rušit jednotlivě.

Vzhled byl navržen jako minimální, s ohledem na možnost jeho přizpůsobení.

5 Implementace

Serverová a klientská část komunikují za pomoci *HTTP GET* prostoru protokolu ze strany klienta a *JSON* ze strany serveru. Klientská část zasílá *akci,* jmenný prostor, *určení akce* a *hodnoty.* Serverová část zasílá strukturovaná data a návratový stav požadavku.

5.1 Serverová část

Serverovou část jsem pro jednoduchost zapouzdřil do jedné třídy *Caché* objektu typu `%RegisteredObject` – serverová část je bezstavová, takže postačí pouze instance neperzistentního objektu.

Pro každou *akci* se volá odpovídající *metoda*, která implementuje požadované chování. *Metoda* si dle potřeby vyžádá načtené *parametry* a po jejich případném zpracování je předá instanci vestavěné třídy nebo perzistentního objektu, která uskuteční požadované chování. Následně jsou vypsány případná vyžádaná data a navrácena odpověď – obojí zapouzdřené v *JSON*.

Metoda vypadá například následovně:

```
Method GlobKill() As %Boolean
{
 s St = ##Class(%Studio.Global).Kill("^" _ ..Qry)

 q ..Error(St)
}
```

Tato *Metoda* implementuje operaci zrušení *globálu*. Volá metodu `Kill` třídy `%Studio.Global` s parametrem, který obsahuje název *globálu* v třídní proměnné `..Qry` (*určení*).

Výsledek volání je typu `%Status`, který třídní metoda `Error` vyhodnotí a navrátí úspěch či neúspěch, který se předá *metodě*, která `GlobKill` volala. Ta vypíše /vrací/ odpověď klientské části v podobě zasílaného stavu. Metoda `Error` navíc v případě chyby vypíše ve formátu *JSON* chybovou hlášku.

5.1.1 Převod do JSON

V případě *datové akce* serverová část vypisuje /vrací/ v rámci prvního pole *JSON* (*array*) názvy sloupců. V každém dalším poli je vždy jeden datový celek (např. jeden uživatel). Klient tedy předem neví jaký obsah mu bude doručen – názvy sloupců i jejich počet se mohou dynamicky měnit.

5.1.2 Převod dat globálu do JSON

Mimo statických částí generovaného *JSON* je také potřeba zapouzdřit i data, která mohou být v případě *globálů* N-rozměrná, a to v podobě *seznamu* (viz 3.4.1). Pro tento účel se data načítají

v režimu pro *Caché* nativním /datovém/ (a nikoliv v režimu pro zobrazování), v němž lze pracovat s obsahem globálu jako při přímém přístupu (obdoba použití `^` před názvem proměnné, viz 2.3).

V tomto režimu lze rozpoznat, zdali se jedná o *seznam* použitím funkce `$LISTVALID`, a rekurzivně se provede převedení na pole *JSON* i pro všechny hodnoty v seznamu. Pro výpis /vrácení/ hodnot v *JSON* je nutné převést hodnoty *globálu* (které mohou být např. i binární) do formátu pro zobrazení uživateli. Převedení může vypadat například takto:

```
##Class(%Studio.Global).format(Cache_Value, .Text_Value)
```

V ukázce se volá metoda `format` třídy `%Studio.Global` s parametry `Cache_Value` – proměnná obsahující převáděnou hodnotu a `.Text_Value` – proměnná předaná referencí, která po provedení *metody* obsahuje převedenou hodnotu.

5.1.3 Ukládání hodnot

Veškeré klientem zaslané *hodnoty*, které obsahují více položek (*seznam*), se zasílají jako řetězec, který server následně interpretuje, a tím vytvoří *seznam*, který již lze procházet po jednotlivých položkách, nebo jej lze celý uložit jako *globál* do databáze. Převod řetězce obsahujícího *seznam* v textové podobě na *seznam* je realizován následovně:

```
X ("(A) s A = " _ ..Retezec, .Seznam)
```

Kde **`X`** je příkaz **`XCUTE`**, který naváže vnitřní pomocnou proměnou **`A`** na proměnou `Seznam` a řetězec v proměnné `Retezec` je interpretován a přiřazen do proměnné `Seznam`.

5.1.4 Výjimky

Korektnost parametrů či jejich případná absence není před voláním *metody* vestavěné třídy nebo objektu nijak ošetřována a může tak dojít k lokální chybě (v rámci navrácené hodnoty typu `%Status`), nebo také může dojít ke globální výjimce, kterou serverová část odchytí. V obou případech je navrácen *JSON* se stavem indikujícím chybu a na místě dat je chybová hláška.

5.2 Klientská část

Tato část je z hlediska implementace nejobsáhlejší, protože zahrnuje veškeré nakládání s daty, jejich zpracování, zobrazování uživateli a zasílání interakcí serveru.

Implementace *logiky* má jasně vymezený *interface* – rozhraní (ovládací prvky a k nim příslušné funkce), které slouží pro interakci uživatele. Je to z toho důvodu, aby bylo zřejmé, jaká činnost bude provedena.

Pro implementaci jsem využil některých funkcí knihovny `underscore.js` [14] a dále pomocné funkce pro zrušení objektu v *DOM* podle jeho `id` [16].

5.2.1 Generičnost

Veškeré zobrazované /vykreslované/ prvky – *datový obsah*, *interface* (uživatelské akce) i *stromové menu* – jsou vytvářeny genericky, tedy bez ohledu na *akci*, která je prováděna. Lze však specifikovat dodatečné parametry pro vykreslení k dané *akci*.

5.2.1.1 Specifikace jména akce

Mějme příklad:

```
Akce[10] = 'UserList';
```

Jedná se o *akci* výpisu seznamu uživatelů (viz 4.2.3 – *uživatelé*). Textové označení `'UserList'` slouží pouze pro komunikaci se serverovou částí. Interní označení je `10`. Výše uvedená definice je plně postačující pro vygenerování platného dotazu a odeslání jej serverové části, a to voláním jedné z příslušných funkcí, jako je např. `GetData(10)`.

5.2.1.2 Stromové menu

Pro vykreslení získaného obsahu ve *stromovém menu* je zapotřebí např. definice:

```
Strom[10] = ['.',      0,          0,          1];
```

První z hodnot určuje oddělovač, který bude použit pro stromové větvení. Druhá z hodnot je příznak, zdali se jedná o tvar uzlu globálu (je zpracován odlišně, protože obsahuje výčet rozměrů v závorkách). Třetí hodnota značí, kolik úrovní stromu má být při načtení rozbaleno. Poslední z hodnot je příznak, zdali ke stromu vykreslovat *akci* /možnost/ přidání nového prvku. Tím je *stromové menu* vygenerováno.

Zároveň také při stisknutí libovolného listu *stromu* je uživateli v části *datového obsahu*, již bez bez další komunikace se serverovou částí, vykreslena tabulka s hodnotami.

5.2.1.3 Funkce

Pro přidání možnosti *akce* s daty je třeba definovat *funkce*, např.:

```
Funkce[10] = [    ['Edit',   'Uprav',   'Ulož'     ],
                  ['Kill',   'Smaž'     ]          ];
```

Zde jsme definovali 2 *funkce*, které z pravidla vedou (ale nemusí) na odeslání *akce* serverové části – co pole /řádek/, to definice jedné *funkce*. Lze definovat libovolný počet *funkcí*. Tyto *akce* mají interní označení `11` a `12`, protože navazují na číslo aktuální *akce*.

První z hodnot pole definice jedné *funkce* obsahuje název *činnosti* – *JavaScriptové* funkce, která bude volána pro provedení *funkce*. Druhá z hodnot je popisek tlačítka a třetí z hodnot je

alternativní popisek. Alternativní popisek slouží pro *činnost*, která má dvě fáze (např. upravit a uložit).

Činnosti jsou z pravidla generické (provádí stejné chování pro různé *akce*), takže můžeme využít stejné *činnosti* pro smazání globálu, uživatele, i databáze, přestože *činnost* nemá žádnou specializaci pro danou *akci*.

5.2.1.4 Úpravy

Pro *globály* lze implicitně všechny hodnoty upravovat. Pokud mají mít sloupce (vyjma globálů) možnost úprav v rámci klientské části, je potřeba definovat např.:

```
Upravy[11] = [1, 2, 3, 4, 5, 7, 8, 19, 20];
```

Jedná se o seznam sloupců (číslovaných od nuly), které půjdou upravit (v rámci klientské části). Po provedení změn odesílá klient serverové části sloupce pod stejnými indexy, pod kterými mu byly data zaslány.

Obdobným způsobem se specifikují i pole pro nové položky, včetně sloupců, které se u zobrazovaných dat nevyskytují, protože server zasílá více popisků sloupců, než je sloupců dat – ty nebudou při zobrazování dat využity.

5.2.2 Zpracování globálů

Globály mohou obsahovat *seznam*, který server převede na *JSON* (viz 5.1.2). Na místě libovolné z hodnot *globálu* může být řetězec, nebo pole hodnot. Z hlediska implementace jsem uvažoval 2 zanoření (2d). Zbylé pole se převedou zpět na *seznam.*

5.2.2.1 Zobrazování globálů

Zobrazení je potřeba realizovat 2 průchody – v jednom se určí maximální šířka pole a v druhém dojde k samotnému vypsání /vykreslení/ dat.

5.2.2.2 Ukládání globálů

Ukládání je realizováno převedením všech hodnot buněk na řetězec obsahující *seznam*, který je následně odeslán serverové části, která řetězec interpretuje (viz 5.1.3).

5.2.2.3 Kopírování globálů

Kopírování globálů není definováno žádnou *akcí* v serverové části. *Činnost* kopírování totiž pouze zapouzdřuje činnost ukládání /upravení/ globálu. A to tak, že nabídne uživateli změnit název, který dále předává *činnosti* uložit jako *určení*.

Lze tak dosáhnout vytvoření libovolného nového, nebo přepsání libovolného existujícího globálu.

6 Testování

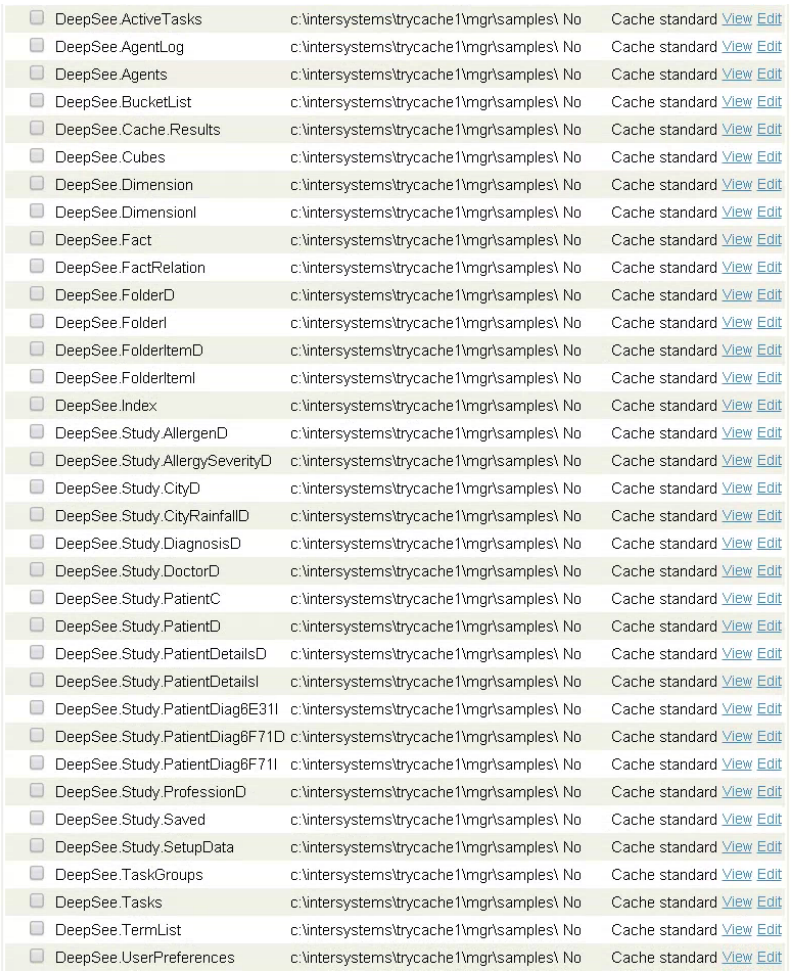
V první podkapitole porovnáme vytvořené webové rozhraní s vestavěným, v kapitole druhé jej otestujeme na přenositelnost mezi různými webovými prohlížeči.

6.1 Porovnání vytvořeného rozhraní a Portálu

Vzájemně porovnáme obě rozhraní, přičemž se zaměříme především na *globály*, protože právě nepřehledné zobrazení globálů a jejich nevhodná editace v *Portálu* vedly k nutnosti vzniku tohoto webového rozhraní.

6.1.1 Přehled globálů

Nejprve se podívejme na příklad zobrazení všech *globálů* pro daný jmenný prostor.



DeepSee.ActiveTasks	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.AgentLog	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Agents	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.BucketList	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Cache.Results	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Cubes	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Dimension	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.DimensionI	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Fact	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.FactRelation	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.FolderD	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.FolderI	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.FolderItemD	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.FolderItemI	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Index	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.AllergenD	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.AllergySeverityD	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.CityD	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.CityRainfallD	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.DiagnosisD	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.DoctorD	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.PatientC	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.PatientD	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.PatientDetailsD	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.PatientDetailsI	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.PatientDiag6E31I	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.PatientDiag6F71D	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.PatientDiag6F71I	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.ProfessionD	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.Saved	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Study.SetupData	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.TaskGroups	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.Tasks	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.TermList	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit
DeepSee.UserPreferences	c:\intersystems\trycache1\mgr\samples\	No	Cache standard View Edit

Obrázek 4: Přehled globálů – Management Portál

Výše je přehled z *Portálu*, který je součástí distribuce *Caché*. Jména globálů jsou v prvním sloupci. Znatelný je často se opakující prefix souvisejících *globálů.* Nyní tento přehled porovnejme s vytvořeným webovým rozhraním.

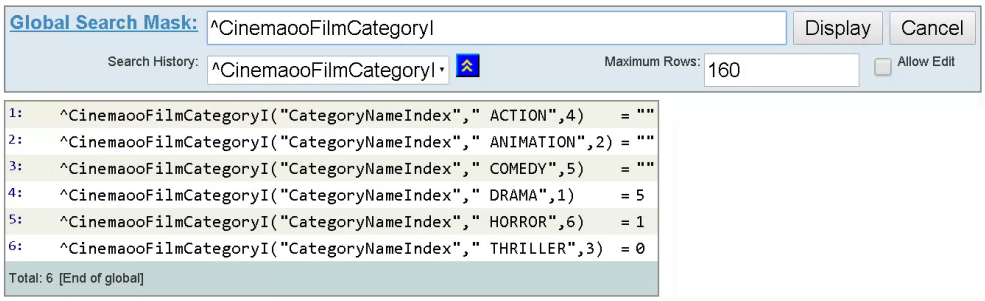
Obrázek 5: Přehled globálů – vytvořené webové rozhraní

Jednotlivé související *globály* se řadí do stromové struktury, kterou lze postupně rozbalovat i zpětně sbalovat a při zvolení listu tohoto stromu se zobrazí příslušná data.

V úvodním zobrazení je celé menu sbalené a umožňuje tak lepší přehled o *globálech.* Dále při jejich postupném rozbalování umožňuje snazší navigaci k hledanému *globálu*.

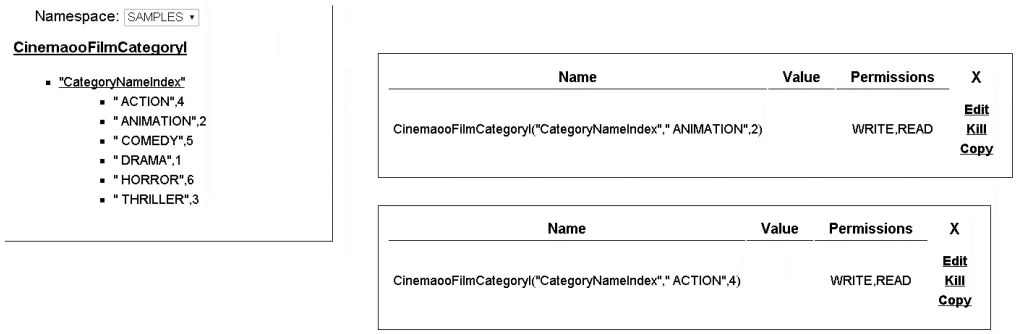
6.1.2 Zobrazení dat globálu

Dále se podívejme na příklad zobrazení uložených dat *globálu.*



Obrázek 6: Data globálu – Management Portál

Na obrázku 5 již na první pohled vidíme značnou redundanci v zobrazení, což je mnohem znatelnější u rozsáhlejších *globálů*, jenž mají mnoho rozměrů. Také zde chybí možnost editace dat (ta je ve zvláštním výběru v přehledu *globálů*, viz obrázek 3).



Obrázek 7: Data globálu – vytvořené webové rozhraní

Na obrázku 7 v *stromovém menu* vlevo není takřka žádná redundance, zobrazují se přímo listy a uzly stromu. V pravé části (*datovém obsahu*) jsou zobrazeny 2 položky, ke kterým jsou zde 3 možnosti práce s daty – editace, odstranění a vytvoření kopie.

6.1.3 Editace dat globálu

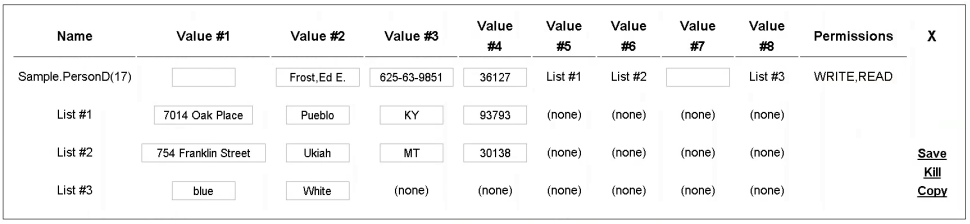
Zaměřme se nyní na editaci dat *Globálu.* V rámci *Portálu* vypadá procházení položek k editaci totožně, jako při jejich zobrazování. Jediným rozdílem je, že pokud se *rozměr globálu* někde větví, tak nabídne výběr větví, podobně jako v nově vytvořeném webovém rozhraní, ale zcela neinteraktivně a s těžko znatelným kontextem. Vypisuje pouze jednu (pod)úroveň jednoho uzlu na jednu webovou stránku. Ty jsou navíc velice špatně rozeznatelné od hodnot.

Podívejme se tedy rovnou na samotnou editaci *dat globálu.*

Obrázek 8: Editace dat globálu – Management Portál

Zde jsou data opět jen v serializované textové podobě, zcela nevhodné pro editaci, především u *globálů*, jenž mají jako hodnotu *seznam*, který obsahuje další *seznam*.

Podívejme se na editaci ve vytvořeném webovém rozhraní.



Name	Value #1	Value #2	Value #3	Value #4	Value #5	Value #6	Value #7	Value #8	Permissions	X
Sample.PersonD(17)		Frost,Ed E.	625-63-9851	36127	List #1	List #2		List #3	WRITE,READ	
List #1	7014 Oak Place	Pueblo	KY	93793	(none)	(none)	(none)	(none)		
List #2	754 Franklin Street	Ukiah	MT	30138	(none)	(none)	(none)	(none)		Save Kill
List #3	blue	White	(none)	(none)	(none)	(none)	(none)	(none)		Copy

Obrázek 9: Editace dat globálu – vytvořené webové rozhraní

Editace zde probíhá po jednotlivých položkách, stejně tak, jak byla data zobrazována. Při editaci dat tak nehrozí, že by se narušila jejich struktura.

6.2 Přenositelnost webového rozhraní

Webové prohlížeče mají často různé způsoby implementace vykreslování či chování. Bylo testováno, zdali se některý z prohlížečů odchyluje od běžného chování.

Test byl proveden v následujících prohlížečích:

- Google Chrome – verze 42.0.2311.152
- Opera – verze 29.0.1795.47
- Light, a light firefox. – verze 35.0
- Internet Explorer – verze 9.0.8112.16421
- Midori – verze 0.5.10

V žádném z testovaných prohlížečů nebyla nalezena žádná odchylka od běžného chování.

7 Závěr

V následujících kapitolách zhodnotím bakalářskou práci a nastíním možnosti dalšího rozšíření webového rozhraní.

7.1 Shrnutí

Cílem této bakalářské práce bylo navrhnout a implementovat webové rozhraní pro správu databáze *Caché*. Hlavním cílem bylo přehledně a logicky prohlížet a spravovat *globály*.

Testování ukazuje, že jsou tyto cíle splněny a že lze pracovat v řadě různých webových prohlížečích.

Práce si mimo jiné kladla za cíl oddělit implementačně klientskou a serverovou část, možnost přizpůsobení vzhledu a snadné rozšiřování funkcionality.

I tyto aspekty vyvinuté webové rozhraní splňuje. Výsledkem je aplikace, jenž může být přizpůsobena a zasazena jako zásuvný modul jiné aplikace.

Nebo také lze využít serverové části pro zasílání příkazů z jiného klienta, či použít klientskou část aplikace pro jiný objektový databázový server.

7.2 Možnosti dalšího vývoje

Možností dalšího vývoje je celá řada. Patří mezi ně především přidání popisů a značení povinných hodnot u formulářů upravujících nebo přidávajících uživatele a databáze, možnost upravovat velikosti *seznamů*, či vytvoření *seznamu* na místě hodnoty.

Dále také možnosti překladu, pokročilejší zpracování chyb serveru, jenž uživateli zobrazí chybovou hlášku. Využití by zajisté nalezlo vyhledávání v globálech, nebo uložení části databáze do klientské části pro práci offline (bez možnosti editace).

Literatura

[1] InterSystems Corporation: Caché Overview [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://www.intersystems.com/our-products/cache/cache-overview/> [anglicky]

[2] InterSystems Corp.: Documentation: Variables [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://docs.intersystems.com/cache201511/csp/docbook/DocBook.UI.Page.cls?KEY=GBAS_variables> [anglicky]

[3] W3C. World Wide Web Consortium. Document Object Model [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://www.w3.org/DOM/> [anglicky]

[4] W3Schools. The XMLHttpRequest Object [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://www.w3schools.com/xml/xml_http.asp> [anglicky]

[5] Wikipedia, the free encyclopedia. WIKIMEDIA FOUNDATION, Inc. Fat client [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Fat_client> [anglicky]

[6] Wikipedia, the free encyclopedia. WIKIMEDIA FOUNDATION, Inc. Caché ObjectScript [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Cach%C3%A9_ObjectScript> [anglicky]

[7] InterSystems Corp.: Caché Technology Guide [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://www.intersystems.com/assets/CacheTechnologyGuide.pdf> [anglicky]

[8] Wikipedia, the free encyclopedia. WIKIMEDIA FOUNDATION, Inc. XHTML [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://en.wikipedia.org/wiki/XHTML> [anglicky]

[9] Wikipedia, the free encyclopedia. WIKIMEDIA FOUNDATION, Inc. Hypertext Transfer Protocol [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol> [anglicky]

[10] InterSystems Corp.: Documentation: CSP Gateway Operation and Configuration [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://docs.intersystems.com/ens20151/csp/docbook/DocBook.UI.Page.cls?KEY=GCGI_oper_config> [anglicky]

[11] InterSystems Corp.: Documentation: Symbols Used in Caché ObjectScript [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://docs.intersystems.com/ens20151/csp/docbook/DocBook.UI.Page.cls?KEY=RCOS_symbols> [anglicky]

[12] Wikipedia, the free encyclopedia. WIKIMEDIA FOUNDATION, Inc. PHP [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://en.wikipedia.org/wiki/PHP> [anglicky]

[13] InterSystems Corp.: Documentation: Creating Caché Databases and Namespaces [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://docs.intersystems.com/ens20151/csp/docbook/DocBook.UI.Page.cls?KEY=GMSM_createdatabases> [anglicky]

[14] Underscore.js. ASHKENAS, Jeremy. DOCUMENTCLOUD INC. Underscore.js [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://underscorejs.org/> [anglicky]

[15] InterSystems Corp.: Class Reference: %CSP.Page [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://docs.intersystems.com/ens20151/csp/documatic/%25CSP.Documatic.cls?APP=1&LIBRARY=%25SYS&CLASSNAME=%25CSP.Page> [anglicky]

[16] Johan Dettmar: Remove element by id [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://stackoverflow.com/a/18120786/4900139> [anglicky]

[17] InterSystems Corp.: Caché ObjectScript Reference: $LISTVALID [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://docs.intersystems.com/ens20151/csp/docbook/DocBook.UI.Page.cls?KEY=RCOS_flistvalid> [anglicky]

[18] InterSystems Corp.: Using Caché Basic: Using Objects [online]. [cit. 24.5.2015]. Dostupný z WWW: <http://docs.intersystems.com/ens20151/csp/docbook/DocBook.UI.Page.cls?KEY=GBAS_objects> [anglicky]

Seznam příloh

Příloha 1: DVD

Obsahuje následující soubory a adresáře:

- `src/` – složka se zdrojovými kódy
- `cache/` – instalační soubory *Caché*
- `install.txt` – návod k instalaci
- `technicka-zprava.pdf` – tato technická zpráva
- `technicka-zprava.odt` – zdrojový soubor k této technické zprávě