ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická

Bakalářská práce

Home Cloud Platform

Tomáš Faflík

2011

Vedoucí práce: Ing. Miroslav Uller

Poděkování

Velmi rád bych poděkoval vedoucímu bakalářské práce ing. Miroslavu Ullerovi za poskytnutý čas při konzultacích a za nově nabité zkušenosti v oboru programování.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Tomáš F a f l í k

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný

Obor: Kybernetika a měření

Název tématu: Home Cloud Platform

Pokyny pro vypracování:

1. Domácí server je malý specializovaný počítač, který může sloužit jako centrum pro účely ovládání různých zařízení v domácnosti, správu sdílených domácích multimediálních souborů a/nebo jako domácí informační systém. Cílem této práce bude prozkoumat aktuální stav oboru a možné existující aplikace (dostupné systémy pro podporu „chytré" domácnosti).
2. Navrhněte a implementujte systém pro domácí server. Implementujte za použití technologií Java (Tomcat, J2EE), PHP nebo ASP.NET (serverová část), Javascript a AJAX. Systém navrhněte modulární, aby bylo možné ho snadno rozšiřovat o nové funkce.
3. Proveďte testování vytvořeného systému s alespoň 3 různými uživateli.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí bakalářské práce: Ing. Miroslav Uller

Platnost zadání: do konce zimního semestru 2010/2011

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 12. 2. 2010

Prohlášení

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

V Praze dne 6. 1. 2011

Anotace

Cílem práce je vytvoření základního frameworku, který by měl sloužit jako centrální bod pro digitalizaci moderní domácnosti. Framework bude provozován na levných a nízkoenergetických domácích serverech. V době, kdy klesající ceny úsporného hardwaru umožňují velmi levně sestavit nenáročné domácí servery, by měla tato práce zaplnit mezeru na trhu s obslužným softwarem pro tyto specializované počítače. Framework je rozšiřitelný pomocí externích aplikací - modulů. Součástí práce je několik ukázkových aplikací využívající tento framework: databáze domácího videa a program pro probouzení počítačů po lokální síti (*Wake-on-lan*). Veškerá práce se serverem probíhá pomocí webového prohlížeče.

Annotation

The goal of this work is to create a basic framework that should serve as a focal point for the digitization of the modern home. The framework is intended to run on cheap and energy-efficient home servers. Nowadays, the falling prices of energy efficient hardware allow for building inexpensive low-end home servers. By this work, we strive to fill a gap in the market for software for these specialized computers. The framework is extensible with external applications - modules. This work also describes several sample applications built upon this framework: a database of home videos and a program to wake up computers over local area network (Wake-On-LAN). The applications can be accessed via web browsers.

Obsah

Seznam obrázků

1. Cíl práce

Cílem této úlohy je vytvořit systém, který by měl sloužit jako digitální pomocník v moderní domácnosti. Toto softwarové vybavení by mělo pomoci postupnému nástupu digitalizace dnešních domácností. Práce je základním Frameworkem, který zaštiťuje a provozuje rozšiřující aplikace. Uživatelé tak mohou mít za pomoci internetu přehled o domácnosti – například o vlastní videotéce, či s jeho pomocí probouzet počítače v lokální síti, aby bylo možné s nimi vzdáleně komunikovat. Výčet možností je takřka neomezený a záleží pouze na modulech, které se do tohoto systému doprogramují a které si následně uživatelé v domácnostech stáhnou a jednoduchým postupem doinstalují. Důraz zde byl kladen na jednoduchost, aby s obslužným softwarem mohli komunikovat lidé, kteří mají pouze základní znalosti ovládání PC.
Jako programovací jazyk jádra programu byl zvolen jazyk PHP a JavaScript, rozšiřující komponenty pak mohou být programovány v různých jazycích a následně vyvolávány pomocí příkazové řádky ze základního frameworku.
Přístup k základnímu rozhraní systému probíhá pomocí webového prohlížeče, avšak k rozšiřujícím aplikacím (*extenderům*) – jako například databáze domácího videa - je možno přistupovat i z programů, jako je Windows Media Center (*za pomocí dálkového ovladače*).
Jednou ze základních funkcí programu je webová aktualizace. Jedná se o způsob, jak jádro systému udržet aktuální a zabezpečené proti případným hrozbám, nebo způsob, jak vyřešit případnou nekompatibilitu s novými prohlížeči, či novou verzí PHP.

Základní myšlenkou je tedy jeden centrální počítač (*server*) v domácnosti, který bude obsluhovat ostatní zařízení, u nichž nezáleží na tom, jestli se jedná o plnohodnotné PC, velmi jednoduché terminály nebo například zařízení typu smartphone. Pro přístup je jen potřeba webový prohlížeč nebo vhodná *extender* aplikace.

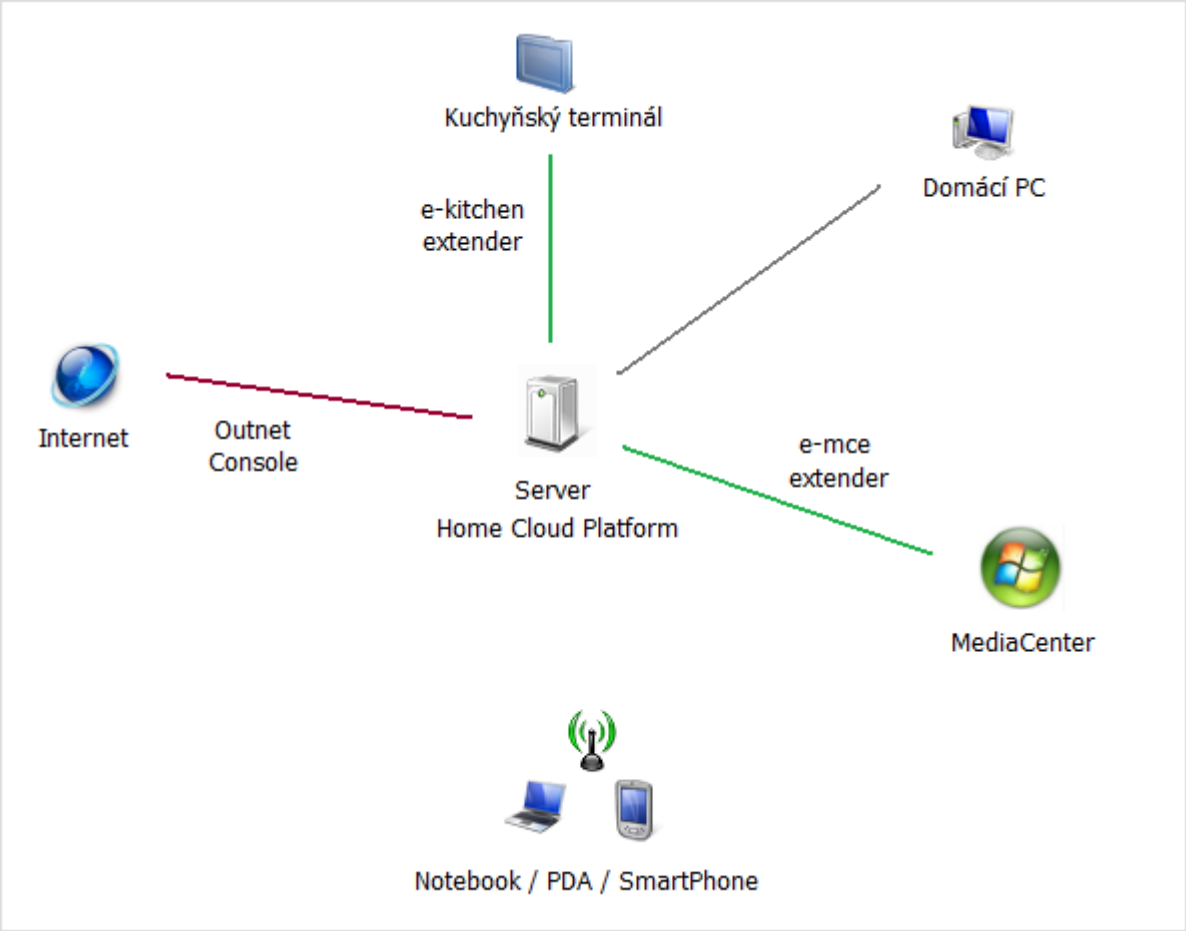
Každý z uživatelů (*členů domácnosti*) má vlastní složku pro osobní soubory, do které je umožněn i přístup z internetu – tímto je možné řešit problém velkého počtu lidí, kteří odesílají osobní soubory emailem, protože nemají přenosné médium při ruce.

Mezi základní balík rozšiřujících komponent patří:

- Domácí videotéka (s rozšířením pro aplikaci Windows Media Center, *e-mce remote extender*)
- Wake-on-lan – program pro probuzení počítačů v domácí síti

1.1 Konkurence

Konkurenční produkty se dají rozdělit do dvou základních kategorií – produkty, které jsou zdarma a placené produkty.
Z první kategorie je možné jmenovat systémy FreeNAS a Ubuntu Server Edition.
Jedná se zde o plnohodnotné operační systémy, které jsou *odnoží* serverových verzí pro velké společnosti, avšak s přehlednějším a jednodušším rozhraním (a v případě FreeNAS i velmi ořezaným systémem o nepotřebné komponenty, který je ovšem cílen do kategorie úložních zařízení – nejedná se tedy o rozšiřitelný software pro potřeby domácích serverů).
Ve druhé jmenované skupině je zástupcem Windows Home Server[i]. Jedná se taktéž o upravenou verzi původně firemního systému Windows Server 2003. Tento operační systém má také vypuštěny některé vlastnosti a naopak doprogramovány části jako například vzdálený přístup (taktéž pomocí webového prohlížeče), ovšem celkem nepřehledný, uzavřený a málo rozšiřitelný.
Tato práce byla tedy původně koncipovaná jako možnost nahrazení právě webového rozhraní v systému Windows Home Server, ovšem vzhledem k programovacímu jazyku, který byl zvolen, je nakonec možné Home Cloud Platform provozovat po drobných úpravách i na systémech Linux a prakticky na všech, kam je možné nainstalovat MySQL databázi a překladač jazyka PHP (*Apache + PHP modul*).
Systém se snaží oddělit normální uživatele od operačního systému, na kterém je provozována a ukazovat jen nezbytnou tvář tak, aby byla srozumitelná pro co nejvyšší okruh lidí.
Další z věcí, která prozatím většímu rozšiřování těchto domácích serverů „nepřeje“ je fakt, že většina lidí odmítá mít doma další počítač, který nebude zdánlivě používat.
Systém Windows Home Server je již dnes v Evropě velmi špatně dostupný. Samostatně ho nelze koupit – prodává se pouze jako OEM (Pouze k novým počítačům, nikoliv samostatně). Navíc od 1. kvartálu roku 2010 pouze partnerům, kteří tyto servery sestavují (například firma *Dell* apod.). Ve vývoji je jeho novější varianta založená na systému Windows Server 2008R2, ale zde ještě není známé datum vydání.



Obrázek 1 – Základní schéma sítě s využitím domácího serveru a softwaru Home Cloud Platform

Jak je vidět z obrázku, HCP server by měl být středem domácnosti, přes který bude probíhat komunikace. Pokud jste mimo domov a budete chtít například ukázat své fotografie z dovolené přátelům, pak je možné se k systému připojit pomocí internetu a dohledat fotografie, případně je stáhnout.

Jako příklad bych mohl uvést zjištění, jestli zlevněné DVD, které aktuálně vidím na stánku, již doma nemáme. Stačí si vzít mobilní telefon/smartphone vybavený prohlížečem a přístupem k internetu, přihlásit se k serveru HCP, podívat se na databázi videa a pokusit se daný film vyhledat. Maximálně do dvou minut bych měl být schopen znát odpověď.

Další z možností je například možnost úspory energie v době, kdy jsme na dovolené. Jak jsem již zmínil, aplikace psané například v jazyce C je možné spouštět přes příkazovou řádku. Pokud tedy budeme mít vhodně napsanou aplikaci, pak serveru můžeme pomocí webového prohlížeče určit čas, kdy má začít doma topit, abychom přijeli do vyhřátého bytu (*např. z dovolené*). Samozřejmě, že k příkazové řádce se nedostane domácí uživatel. Jedná se jen o prostředek, kterým může programátor zásuvných modulů komunikovat s okolím. Uživatel uvidí pouze grafické rozhraní.

1.2 Popis použitých technologií a jazyků

1.2.1 PHP

PHP, původně celým názvem Personal Home Page, je skriptovací jazyk vhodný především pro tvorbu webových stránek, ale v poslední době i pro rozmáhající se cloud computing. Jeho zápis se provádí přímo do *html* kódu a je oddělen speciálními znaky. Skripty se provádí na straně serveru a na lokální počítače je odeslána již vygenerovaná stránka. Veškerou výpočtovou zátěž tedy provádí server. Kód se nijak nekompiluje, pouze přímo píše do stránky a jeho přeložení a spuštění provádí webový server.
PHP od verze 5.0, vydané 13. 7. 2004, podporuje objektové programování.

1.2.2 Apache

Apache je základní software pro webový server, který překládá požadavky na požadované výstupy. V našem případě provádí příkazy jazyka PHP na webové stránce. Software je uvolněn pod volně šiřitelnou softwarovou licencí, která umožňuje ať komerční, tak i nekomerční volné využití. Licence byla napsána společností *Apache Software Foundation*[ii]. Velikou výhodou je fakt, že jej lze provozovat na všech dnes nejrozšířenějších platformách – Linux, Windows, Mac OS, UNIX i Solaris. V původní koncepci byl Apache dostupný pouze pro systémy Linux/UNIX, ale s postupem času se – s určitými omezeními – byly vydány verze pro systémy Windows i Mac OS X.
Nejznámější konkurenční webové servery jsou Internet Information Services a Google Web Server.
Bohužel první jmenovaný je pevně spjat se systémy Windows a druhý jmenovaný je naopak jen pro systémy Linux. Jazyk PHP a serverový systém Apache jsem pro tvorbu zvolil proto, že je možné na serverové straně provozovat prakticky jakýkoliv systém a programu Home Cloud Platform se změny dotknou pouze minimálně.

1.2.3 MVC (Model-View-Controller)

Jedná se o návrhový vzor, kdy jsou jednotlivé komponenty celkového programu logicky oddělené na tři základní části.

Model
- jedná se o reprezentaci informací, se kterými aplikace pracuje

View
- tato část v sobě ukrývá veškeré uživatelské rozhraní

Controller
- část, která reaguje na události a která odpovídajícím způsobem zajišťuje změny v modelu a view.

V celkovém programu se při takovémto návrhu dají například provádět úpravy uživatelského rozhraní, aniž by se nějak změnil chod samotného programu. Datový model ani reakce na dané události nebudou těmito změnami zasaženy.

1. Uživatel provede v programu nějakou akci (*například stisknutí tlačítka*)
2. Tento vstup zachytí controller, který vybere vhodnou akci, se kterou přistoupí a zaktualizuje datový model.
3. View část se poté aktualizuje na základě nových dat z komponenty model
4. Postup se opakuje

Tato architektura byla poprvé popsána v roce 1979 Trygvem Reenskaugem[iii], a poprvé využita v jazyce Smalltalk, vyvíjeným společností Xerox. Dnes tento model běžně využívají moderní složitější webové frameworky (například Zend Framework[iv], CakePHP[v]) a díky nim je možné urychlovat a zefektivňovat softwarový vývoj. V případě chyby je ve většině případů již předem možné odhadnout, ve které z komponent se nalézá.

1.2.4 JavaScript

JavaScript je skriptovací jazyk, který byl vyvinut ve společnosti Netscape. Standardizován byl v roce 1997. Skripty v tomto jazyce se píší přímo do kódu stránky, přičemž se provádí na klientské straně. Pro jejich provedení je třeba mít adekvátní prohlížeč s aktivovanou podporou JavaScriptu. Syntaxe jazyka je odvozena od jazyků C, C++ či Java.

1.2.5 AJAX

Technologie známá pod názvem AJAX (Asynchronní JavaScript a XML) je hojně využívaná pro tvorbu dynamických webových aplikací. Pro správnou funkčnost je třeba podpora ze strany prohlížeče, přičemž všechny dnešní nejpoužívanější prohlížeče AJAX plně podporují. AJAX je vhodné nasadit tam, kde potřebujeme průběžně komunikovat se serverem i poté, co je stránka načtena. Základní kámen této technologie byl položen roku 1996, zavedení prvku <IFRAME> do prohlížeče Internet Explorer 3.0. Tento prvek říká prohlížeči, že má vložit do stránky okno, ve které se načte jiná stránka. Díky JavaScriptu je možné s tímto vnořeným oknem pracovat – například ho obnovovat, nebo jej přesměrovat na zadanou adresu. Dalším požadavkem byla možnost přečtení vnořeného okna pomocí JavaScriptu, což taktéž již bylo

možné funkcí InnerHTML. V tuto chvíli je vytvořena základní možnost, jak komunikovat se serverem, skrze vnořené okno a přitom v původní stránce aktualizovat pouze některý text. Od páté verze Internet Exploreru, jakožto v té době nejrozšířenějším prohlížeči, se objevil prvek XMLHttpRequest, který nahrazuje vnořené okno. Výhodou je, že oproti vnořenému oknu prohlížeč nic neaktualizuje, jedná se o nový komunikační prvek prohlížeče. Drobnou nevýhodou této technologie je opožděná implementace v mobilních zařízeních. Například Internet Explorer Mobile 6.0 tuto implementaci dodnes nemá a vzhledem k nové platformě Windows Phone 7 ani už mít nebude. Ovšem je možné jej nahradit prohlížečem Opera Mobile, který je již k dispozici zdarma (více v základních požadavcích na přístup k serveru HCP).

1.2.6 MySQL databáze

MySQL je databázový systém, který je opět nezávislý na plaformě, na které je provozován. Komunikace s databází probíhá pomocí dotazovacího jazyka SQL. Tento systém je distribuován pod licencí GPL - je volně použitelný.
K použití MySQL jsem se rozhodl proto, že tento databázový systém je součástí volně dostupného balíku EasyPHP (o balíku EasyPHP více v kapitole *Hardwarová realizace serveru*). Alternativou bylo využití databáze SQLite, která je velmi užitečná především díky své minimální velikosti. Z výše uvedených důvodů jsem zvolil systém MySQL. V budoucnu by však neměl být problém umožnit komunikaci s databází SQLite, vzhledem k tomu, že základní příkazy jazyka SQL jsou v obou případech stejné. Při úvodním nastavení při instalaci frameworku by zde mohla být umožněna volba, jakou databázi bude framework používat.

1.3 Hardwarová realizace serveru

První věcí, kterou je třeba si ujasnit, je otázka *„Jak má být server výkonný?“*. Když se před běžnými uživateli řekne server, většina lidí si představí velké stroje s cenou přesahující i 100.000,-- Kč. Tak velký výkon není ovsem pro domácí podmínky většinou zapotřebí. Domácí server není primárně určen pro vyřizování miliónů požadavků za hodinu, není plánován jako web hosting velkých komerčních aplikací. Drobnou výjimku by mohl tvořit například elektronický obchod rodinného podniku, ovšem ani zde není předpokládaná návštěvnost srovnatelná s velkými obchody. Za takovýchto podmínek zjistíme, že po většinu času server nebude takřka dělat nic. Proto by měl být především kladen důraz na energetickou spotřebu takovéhoto domácího zařízení.

Dalším z aspektů by měla být i co nejnižší pořizovací cena serveru, aby bylo možné takovéto zařízení do domácnosti zakoupit. Základním problémem, jak již bylo v úvodu zmíněno, bude pravděpodobně psychologický problém uživatelů – „*Potřebuji další počítač, se kterým nebudu pracovat*?“. Otázka je samozřejmě oprávněná. Nikdo z uživatelů si fyzicky k serveru nesedne a nezačne na něm vytvářet grafy, či psát texty. Ovšem druhou stranou věci je fakt, že uživatel bude mít odkudkoliv přístupné své soubory na serveru uložené, bude mít přístup například k databázi domácího videa, aby věděl, jaký film si může v půjčovně vypůjčit, či jaký si má koupit.

Tyto požadavky na *úspornost* a *nízké pořizovací náklady* nám v dnešní době vyústí v hardware, který se používá v dnešních zařízeních známých pod slovem Netbook. Jako procesor je v naprosté většině užit Intel Atom, který byl také pro tyto účely navržen a v roce 2008 vydán. Jeho maximální tepelný výkon je u dvoujádrových modelů 8W, což v porovnání s úspornými procesory od konkurenční společnosti AMD se 45 W je méně než pětina. Úsporné modely svých procesorů ještě vyrábí firma VIA, u které je ovšem cena podstatně vyšší za velmi podobný výpočetní výkon.

Následující ceny jsou včetně DPH k datu 1. 12. 2010. Pro jejich vyhledání jsem použil server www.zbozi.cz s lokálním vyhledáváním pro hlavní město Praha a s řazením od nejnižší ceny (*jedná se pouze o nové zboží*).

Základní deska + procesor + grafická karta + síťová karta **INTEL D410PT Packton**	1.384,-- Kč
Operační paměť **Kingston 4GB KIT DDR2 800MHz**	1.355,-- Kč
Pevný disk **WESTERN DIGITAL Caviar Green 1000GB**	1.439,-- Kč
Počítačová skříň + zdroj **In-Win BM-639 Black/Silver, 160W**	1.138,-- Kč
Součet	5.316,-- Kč

Celková pořizovací cena hardware nám tedy vychází na cenu lehce nad 5.000,-- Kč. K tomu je ještě třeba připočítat softwarovou výbavu, jejíž výbava je díky užitému jazyku PHP plně na uživateli. Systémy typu Linux jsou volně šířeny, ovšem pokud uživatel preferuje systémy Windows je plně na něm, zdali si dokoupí operační systém, který vyjde v nejlevnější variantě na cenu okolo 2.500,-- Kč.
Jelikož systémy Windows nemají integrovaný Apache server, jeví se mi jako nejvhodnější využít kompletní balík EasyPHP[vi]. Je distribuován zdarma a v základním nastavení je plně kompatibilní s Home Cloud Platform. Balíček obsahuje sever Apache, MySQL databázi a v poslední verzi podporuje PHP verze 5.2.
Co se týká odběru takovéhoto zařízení, špičkový odběr, z vlastních měření na starší variantě než je navržený hardware, nepřesáhne 60 W a průměrný odběr je řádově 20W při provozu se třemi běžnými pevnými disky.

Obrázek 2 – Velikosti a tvary některých domácích serverů

2. Vlastní systém

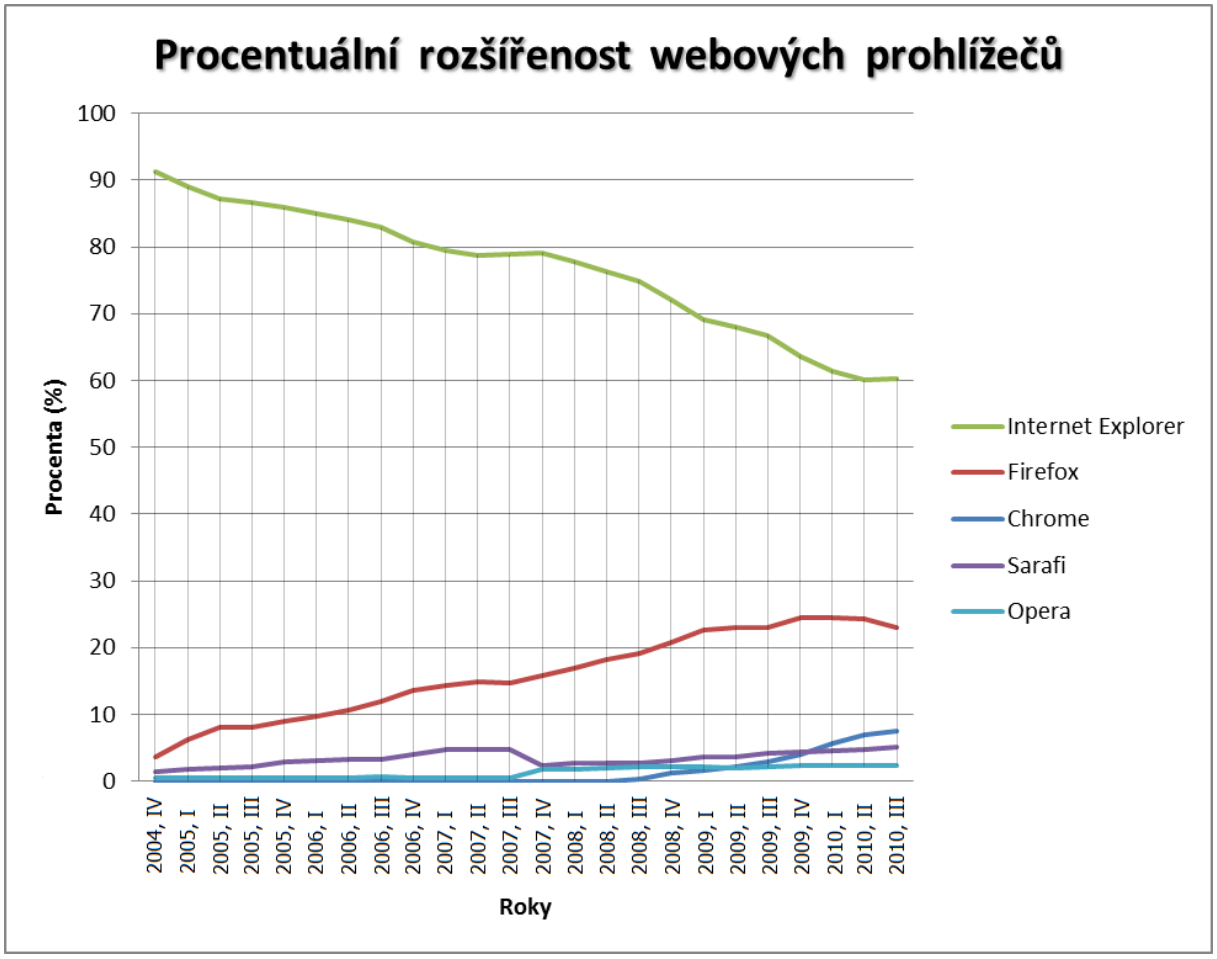
Na začátku jsem musel učinit základní rozhodnutí, zdali budu aplikaci programovat objektově a pomocí Model-View-Controller techniky, nebo původním neobjektovým stylem – jako bylo běžné dříve u PHP 4. Vzhledem k tomu, že jsem chtěl zároveň systém vytvořit moderním pro dnešní dobu a styl programování, ale současně rozumím i programátorům „odrostlým“ právě na PHP 4, tak jsem se rozhodl navrhnout Framework univerzálně. Systém si komponentu nejprve prohledá a zjistí způsob programování (zdali je dodržen styl MVC nebo ne) a podle toho k danému programu přistupuje. Volba objektového programování je pak na programátorovi komponenty samotné. Další z věcí, které mně k tomuto univerzálnímu rozhodnutí vedla, je kompatibilita starších aplikací z jiných projektů, které je možno velmi jednoduše převést do tohoto nového systému.

2.1 Minimální požadavky pro přístup

Pro přístup k serveru Home Cloud Platform je nutné splňovat požadavky na webový prohlížeč, případně speciální, dané specifickou *extender* aplikací.
Základní je požadavek na AJAX, akceptování Cookies a JavaScript.

Obrázek 3 – Dnešní nejrozšířenější desktopové internetové prohlížeče

Všechny z dnes nejpoužívanějších prohlížečů tyto základní požadavky splňují a tak jediný rozdíl je pouze v jejich vykreslovacím jádře. Ovšem v takovémto případě není narušena funkčnost systému, ale pouze grafická stránka. Ač systém plně splňuje podmínky pro kaskádové styly, drobné rozdíly v jednotlivých prohlížečích jsou přesto patrné. Návrh a optimalizaci jsem prováděl primárně pro prohlížeč Internet Explorer. Důvodem je jeho stálá nadvláda mezi ostatními programy.
Pro provoz na zařízeních PDA, smartphone či tablet je vhodné využít prohlížeč v závislosti na platformě. V systémech Apple je prohlížeč Safari, jež má velmi podobné jádro s desktopovou verzí, na zařízeních postavených na mobilní platformě Google Android také doporučuji ponechat výchozí prohlížeč Chrome Lite. Pouze pro zařízení Windows Mobile 5.0 a 6.x je nutné základní prohlížeč Internet Explorer Mobile vyměnit za Operu Mobile nebo Operu Mini, která zvládá technologii AJAX a JavaScript.



Obrázek 4 – Procentuální rozšířenost webových prohlížečů (*zdroj dat: Net Applications*[vii])

2.2 Instalační štít

Jednou z dalších komponent systému je instalační štít neboli zavaděč systému. Je to průvodce, pomocí kterého by měl i běžný uživatel nastavit systém. Funkce je následující: instalátor si sám stáhne z internetu nejaktuálnější soubory jádra systému, základní balík programů a Framework zavede spolu s prvními účty a spustí. Instalační štít je vytvořen univerzálně, kdy každý jeho krok je programován jako samostatná stránka, případně dvojstránka, pokud se jedná o prováděcí funkci. Dohromady pak instalátor drží *xml* soubor, kde jsou jednotlivé kroky popsány v pořadí, v jakém mají být vykonány.

Obrázek 5 - Obrázek instalačního štítu

Kroky instalačního štítu se dělí na 2 části – funkce „next“ (tato část slouží k ověření dostupnosti některých funkcí atp.) a funkce „execute“. U kroku typu „execute“ se při stisknutí tlačítka nedostaneme ihned na další stránku, ale provede se skript, který musí mít správný formát názvu (tedy například, pokud je soubor nazvaný *f1.php,* pak jeho spustitelná část musí mít název *f1-ex.php*). K tomuto názvu je poté nutno na konec souboru přidat 2 řádky:

```
<a href="?step=<?php echo $_SESSION['install_step']+1; ?>">dalsi</a>
<script>document.location.href = "?step=<?php echo $_SESSION['install_step']+1; ?>"</script>
```

Ukázka skriptu funkce execute

Tento skript zabezpečí, že daná funkce se provede a poté automaticky přeskočí na další krok instalačního štítu.

Jak je z ukázky kódu vidět, instalátor si ukládá jednotlivé kroky pomocí session. Při programování jsem uvažoval nad třemi možnostmi – využít *cookie, session* nebo jen adresní řádek s parametry. Volba *session* se mi zdála jako nejlepší, protože *cookie* nemusí mít každý ve svém prohlížeči povolené a psát kroky do adresního řádku není vhodná varianta – je zde možnost adresní řádek upravit a může nastat jakákoliv chyba. Naproti tomu *session* uživatel

upraví opravdu jen tím, že v instalačním štítu stiskne tlačítko a adresní řádek s vyvoláním daného skriptu se vždy ověřuje podle základního *xml* souboru.
Štít není navržen pouze pro potřeby frameworku HCP, ale je použitelný univerzálně, přičemž jednotlivé kroky (*funkce*) je možné různě kombinovat, upravovat a tím můžeme vytvořit zavaděč prakticky jakékoliv webové aplikace.
Zavaděč jsem do tohoto systému přidal proto, aby prvotní nastavení systému zvládl co největší počet lidí bez potřeby jakékoliv asistence. Jelikož jsou cílovým zaměřením domácnosti, pak by mělo být i spuštění intuitivně zvládnutelné.

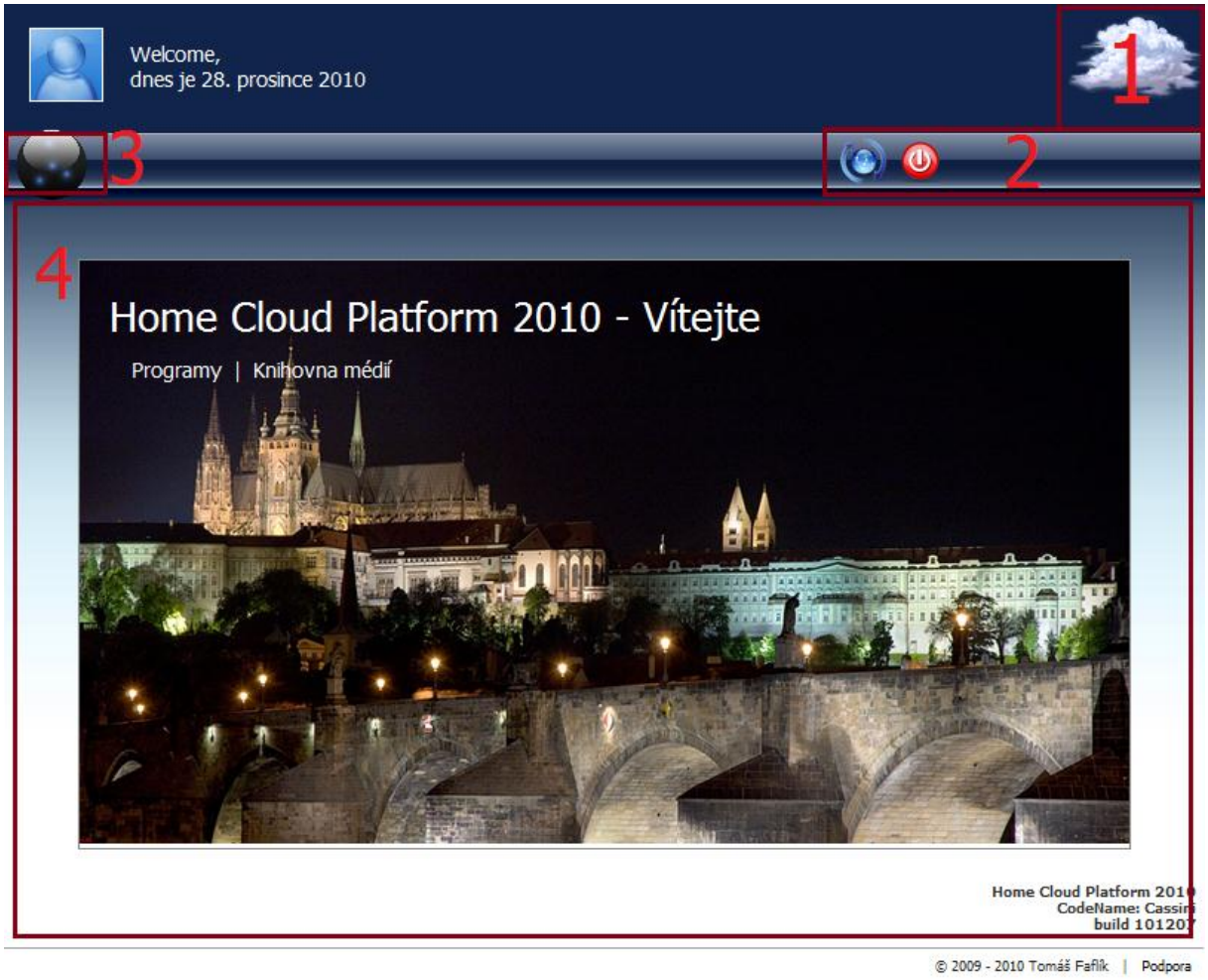
2.3 Přihlašovací obrazovka

Jedna z pilířů systému je bezpečnost. Pro autentifikaci uživatele jsem sáhl ke kombinaci cookie i session. To podle mého názoru zajišťuje bezpečnost na adekvátní úrovni, kdy se průběžně ověřuje, jestli je session i cookie stejná. Při úspěšném přihlášení se náhodně vygeneruje řetězec znaků, který se uloží jak na klientské části, tak na serveru a je málo pravděpodobné, že případný útočník zcizí obě části zároveň. Pokud se tyto údaje liší, tak systém uživatele automaticky odhlásí. Z každé IP adresy je povolené přihlášení jednou za 3 sekundy tím, že se neúspěšné pokusy ukládají do databáze. Základní přihlašovací obrazovka je pak vybavena skriptem, který vlastní přihlášení na tuto dobu pozdrží. Toto by mělo odvrátit útok hrubou silou na heslo tím, že další kontrola hesla proběhne až po 3 sekundách. Ač se zdánlivě jedná o zdržování, máme tím zaručeno, že jeden uživatel vyzkouší maximálně 1200 hesel za 1 hodinu. To spolu s kombinací silného hesla by mělo odolat útoku hrubou silou na prolomení hesla. Navíc, pokud se nepovede zadat heslo 10x za sebou správně, pak se do databáze vloží třiceti minutový *timeout* pro další přihlášení. Další z možností, pokud by byly servery vystavovány větším útokům, by mohla být služba v rámci jednoho z programů, který by zjišťoval neúspěšné pokusy z dané IP adresy a odesílal jej prostřednictvím komunikačního kanálu webových aktualizací na hlavní server. Odtud by se tato podezřelá adresa distribuovala na všechny připojené servery, které by ji mohli odmítat přímým zapisem do souboru *.htaccess*. Jedná se ovšem zatím o teoretický námět, který je možné doprogramovat a rozdistribuovat pomocí aktualizací.

Obrázek 6 – Přihlašovací obrazovka

2.4 Základní obrazovka

Při volbě vzhledu základní obrazovky, jsem se snažil o maximální jednoduchost a přehlednost. Základ by měl být stejný s tím, na co jsme běžně zvyklí z operačních systémů, či mobilních telefonů – základní uvítací obrazovka, panel pro spuštění programů, panel pro rychlý přístup k důležitým funkcím a podobně.



Obrázek 7 – Návrh základní obrazovky (aplikace .desktop)

Jak je z obrázku 4 patrné, základní obrazovka je rozdělena na 4 základní části.

1. Část pro miniaplikace – aktuálně je přístupná pouze „Zobrazovač aktuálního počasí“.
2. Část pro rychlý přístup k základním funkcím
3. Nabídka *Launch Bar* pro rychlý přístup k programům
4. Plocha pro aplikace

2.5 Adresářová struktura

Struktura adresáře, ve kterém se systém HCP nalézá, vypadá následovně:

core – Obsahuje celé funkční jádro systému. Aktualizuje se pomocí aktualizačního štítu.
programs – Tato složka je určena jako místo pro instalaci rozšiřujících programů.
users – Složka pro uživatelská nastavení jednotlivých účtů.
media – rozdělena na 3 základní podsložky
    music – složka hudby
    videos – složka filmů a videí
    pictures – složka obrázků a fotografií
settings – Složka pro umístění souborů s globálním nastavením

2.6 Systémové registry

Důležitou součástí systému jsou registry. Jsou to konfigurační informace o instalovaných modulech uložené v textové formě. Soubor se jmenuje *registry.php* a je uložen v systémové složce *core*. Základní konfigurační data obsahují údaje o názvu programu, jeho verzi a cestě. Do souboru registrů má přístup **instalátor**, který dokáže danou aplikaci nainstalovat, nebo odebrat.
Díky tomuto souboru může aktualizační štít, či **instalátor** okamžitě zjistit, jaké programy a v jakých verzích jsou nainstalovány. Programy se ještě podle názvu rozdělují na dvě skupiny – systémové programy (nepostradatelné pro správný běh systému) a volitelné aplikace.
První skupina se odlišuje tím, že má v názvu na prvním místě znak tečka. Instalátor pak ví, že takto označené aplikace nemůže odinstalovat a může je pouze zaktualizovat na novější verzi.
Registry v sobě uchovávají informace v následující struktuře:

```
Název Programu|číslo verze|cesta k programu
Hello World|1.0|programs/helloworld
```

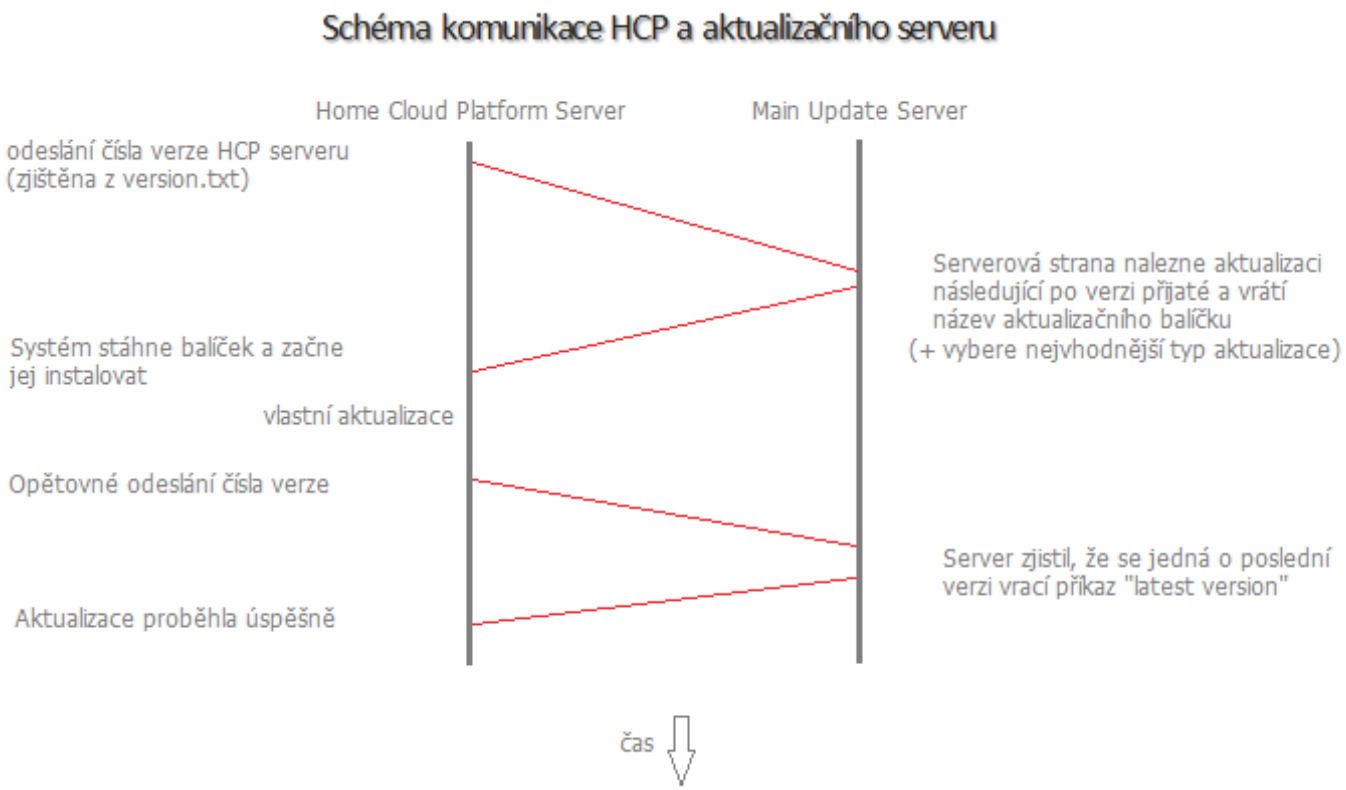
2.7 Aktualizační štít

Aktualizační štít je základní komponenta systému, která udržuje jádro systému aktuální. Aktualizace se provádí z centrálního serveru a pro jejich úspěšné provedení je třeba „svým způsobem“ snížit zabezpečení serveru. Do složky, kde se nachází HCP, je třeba nastavit plný přístup (777, nebo alespoň 755).
Z bezpečnostního důvodu by to ovšem problém být neměl, protože skripty, sloužící k odeslání souborů, jsou aktivní až po přihlášení do systému a nikdo jiný by v domácnosti přístup k serveru mít neměl. Proto by ani jiní uživatelé neměli mít do složky přístup.
Při těchto podmínkách může aktualizační štít pracovat, přičemž jeho práce se skládá z devíti základních kroků.

1. Vyhledání aktualizačního souboru.
2. Stažení aktualizačního souboru do dočasné složky „update“

3. Rozbalení souboru aktualizace
4. Spuštění skriptů před aktualizací (pokud jsou k dispozici)
5. Vlastní aktualizace (částečná (*part*) – nahrazení několika souborů, nebo úplná (*complete*) – výměna složky „*core*“ za novou verzi)
6. Spuštění skriptů po aktualizaci (pokud jsou k dispozici)
7. Vytvoření souboru version.txt, který uchovává informace o dané verzi systému
8. Vymazání dočasné složky update (včetně staré verze jádra, která se kvůli bezpečnosti aktualizačního procesu uchovává)
9. Znovunačtení stránky (případně pokračování v další instalaci).

Schéma komunikace mezi HCP serverem a aktualizačním serverem je na následujícím obrázku.

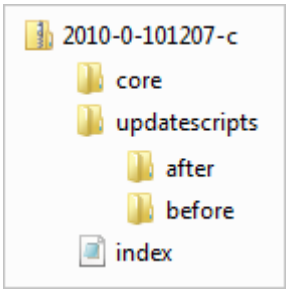


Obrázek 8 – Schéma komunikace serveru HCP a aktualizačního serveru

V této modelové situaci je k dispozici pouze jedna aktualizace – proto dojde během 4 kroků k ukončení aktualizace. Jiný případ by nastal, kdyby byla verze starší. Poté by se postup po opětovném odeslání čísla verze HCP opakoval do té doby, než HCP dostane příkaz „*latest version*“.

Číslování verzí systému se provádí podle data, kdy byla vydána. Podle toho se také jmenují aktualizační soubory: tedy například ze dne 24. 12. 2010 bude číslo verze „2010-0-101224“, kde první číslo udává hlavní verzi (*HCP 2010*), druhé číslo minoritní verzi a třetí číslo ve

formátu *rrmmdd* udává aktuální sestavení, podle data. Tento způsob číslování se mi jevil jako nejlepší, protože aktualizační serverová část jednoduše porovnává velikost čísel a z verze je jednoduše patrné, kdy byla vydána.
Aktualizační balíčky jsou distribuovány ve formě zip souboru, a jak již bylo zmíněno, jedná se o 2 typy aktualizačních balíčků: částečné a úplné. Oba typy souborů mají velmi podobnou strukturu.



Obrázek 9 – Schéma kompletního aktualizačního souboru

U typu *„complete“* (přípona –c.zip) je povinný soubor *index.php* a kompletní složka core s jádrem programu. Dále jsou zde volitelné složky *updatescripts/before* a *updatescripts/after*, kam je možné nahrát skripty libovolného názvu, které se vyvolávají před či po aktualizaci v pořadí dle názvu (umožňují například změnu v číslování verzí, změnu v databázi a podobně). U typu *„part“* (přípona –p.zip) je vše nepovinné – můžeme vynechat složku *core*, soubor *index.php*, či složku *updatescripts*. Složka *core* nemusí být kompletní, protože se přepíší pouze soubory, které se v ní nachází (případně se doplní, pokud jsou nové). Podmínkou ovšem je dodržet požadovanou strukturu – soubor vložit do podsložek, kde se reálně nachází. Soubor je tímto podstatně menší a rychleji aktualizace proběhne. Nevýhodou je, že pokud se neaktualizuje delší dobu, pak může počet balíčků značně narůst. Od toho jsou zde kompletní aktualizace, které jednu za určitou dobu shrnou předešlé částečné aktualizace (snažím se dodržovat cyklus, po deseti částečných aktualizacích vložit souhrnnou kompletní i vzhledem k jiným mým projektům, napojeným na tento aktualizační systém).
Vzhledem k tomu, že ve složce jádra a souboru *index.php* nachází prakticky celý funkční systém, je možné kompletní aktualizaci využít i k instalaci. Propojením s instalačním štítem tak vzniká možnost, jak jednoduchým způsobem nainstalovat systém a přitom mít jeho poměrně aktuální verzi. Instalační štít pak na rozdíl od aktualizací ještě doplní adresářovou strukturu a zavede prvotní uživatelské účty.

2.8 A-lang 2.0

A-lang je šablonovací jazyk, pomocí kterého se registrují akce v controlleru. Jeho zápis probíhá do view části a musí mít speciální syntaxi, která by se ve zbytku stránky neměla volně nacházet. Tento jazyk jsem původně vyvinul již v roce 2008 pro jiný projekt, ale jak se zde ukázalo, jedná se o velmi šikovnou metodu vkládání dynamického obsahu do jinak statické stránky. Proto jsem jazyk využil i zde. V této druhé verzi se dočkal značného vylepšení

v podobě vkládání funkcí s možnými argumenty. Pohledová část se pak může dynamicky přizpůsobovat v závislosti nejen na aktuálním stavu, ale i v závislosti na tom, jaký je vstup funkcí - například, jaký uživatel jej vyvolal. Skript tedy nahradí takto vypadající místo v textu jiným, který daná funkce vyvolá.
Jazyk umožňuje následující konstrukce:

```
[%promenna%]
[%funkce(argumenty)%]
```

Názvy proměnné nebo funkce podléhají standardním pravidlům programování, povoleny jsou čísla, znaky abecedy či znak "_" a je nutné dodržet velikost písmen. Definice výstupu se píši do příslušného controlleru, který je k aplikaci přidružen (pokud se nejedna o základní funkce, definované přímo v jádře programu - v AbstractController-u).
Informace o AbstractController-u jsou uvedeny v kapitole „Postup návrhu dalších modulů“.

Jako další existující šablonovací systémy je možné jmenovat Smarty[viii] či Templatepower[ix].

2.9 Uživatelské účty

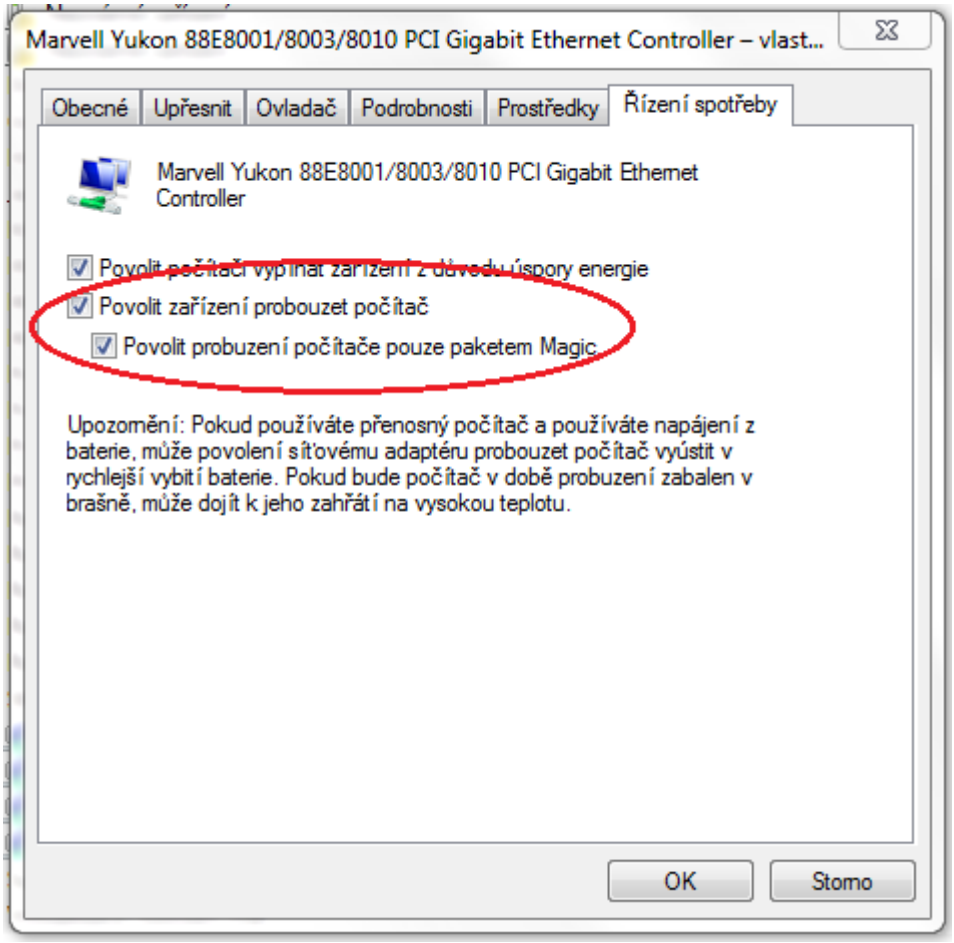
Každý z uživatelů má svoje vlastní přihlašovací údaje. Ty se skládají z přihlašovacího jména a hesla, na které je kladen požadavek délky alespoň pěti znaků. Přihlašovací jméno nesmí obsahovat diakritiku, naproti tomu heslo může. Heslo je šifrováno pomocí algoritmu MD5. Tento algoritmus neuchovává celé heslo, ale pouze jeho otisk (*tzv. hash*). Při proniknutí do databáze serveru a ukradení tohoto otisku tak není možné zpětně heslo dekódovat (*resp. není dnes reálné a výdělečné vzhledem k vynaloženému úsilí*). Každý z uživatelů má také vlastní složku pro soubory, která se jmenuje podle jeho přihlašovacího jména a je umístěna ve složce *users*. Takovouto službu poskytují servery třetích stran (například server *SkyDrive*[x]), na kterém má uživatel 5GB prostor na osobní data. Je zde však limitace jak ze strany celkové kapacity úložiště, tak i limitace velikosti jednoho souboru. Tyto limity nám na domácím serveru odpadají.
Každý z uživatelů navíc může měnit vzhled serveru HCP podle vlastního vkusu. Grafická témata jsou uložená ve složce jádra systému a aktualizují se spolu s aktualizacemi.
V základním balíku systému se nachází grafická témata „Classic Blue“, „Sunset“ a „Spring Japan“. V budoucnu budou nová témata přidávána a distribuována spolu s novými funkcemi systému pomocí aktualizací.

3. Základní dodávané aplikace

K systému jsou dodávané základní aplikace, pomocí kterých je možné demonstrovat funkčnost systému.

3.1.1 Wake-on-lan

Wake on LAN je technologie, pomocí které je možné dálkově zapnout počítač v lokální síti typu LAN. Standard byl navržen v roce 1997 firmou IBM. Pro správnou funkčnost je potřeba splňovat minimální hardwarové požadavky a mít správně nastavené lokální stanice. Je nutné vlastnit ATX formát základní desky, který umožňuje softwarové zapnutí napájení, které není, na rozdíl od staršího AT standardu, fyzicky odděleno přepínačem typu „zapnuto/vypnuto“. Dále je potřeba síťová karta s podporou této technologie a BIOS, který ji podporuje a dodává síťové kartě energii i ve vypnutém stavu počítače. Síťová karta je poté i po vypnutí napájena a naslouchá komunikaci okolní sítě. Pokud zaznamená tzv. *Magic Packet,* pak vyšle signál a nastartuje se napájení, případně se systém probudí ze spánkového režimu. Magický paket je šest konstantních hexadecimálních bajtů *FF:FF:FF:FF:FF:FF*, za kterými následuje 16x opakující se MAC adresa této síťové karty. Pro odeslání se používá protokol UDP na portu 9.
MAC adresa je fyzická adresa síťové karty, která je dána pevně z výroby a teoreticky by se nemělo stát, aby byly síťové karty se stejnou adresou. Pokud tuto adresu neznáme, pak je možné si jí v systémech Windows vyhledat pomocí příkazu *ipconfig –all* v příkazovém řádku.
Pokud se na lokálních stanicích nachází systém Windows je ještě potřeba povolit funkci probuzení v operačním systému, který je schopný požadavek na probuzení odmítat. Toto se provede ve Správci zařízení, kde na dané síťové kartě aktivujeme možnost „*Povolit probuzení počítače pouze paketem Magic*“ (*viz obrázek 10*). Minimální požadavky na funkci Wake-on-LAN splňují všechny dnes prodávané počítače. Jedinou výjimkou samozřejmě jsou bezdrátové sítě typu WiFi, kde tato funkce z energetických důvodů povolena není.
Práce probíhá na spojové vrstvě OSI/ISO modelu. Není zde tedy třeba IP adresa dané stanice a stačí pouze zmíněná MAC adresa. Proto je funkce nezávislá na použitém operačním systému (*nepotřebujeme pro funkci žádné speciální ovladače apod.*).



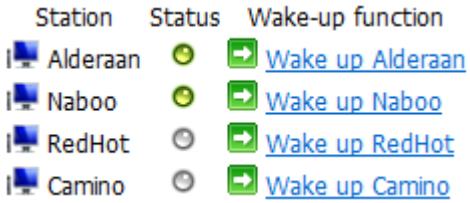
Obrázek 10 – Aktivace funkce Wake On Lan v systémech Windows

Funkce aplikace na straně serveru je poté jednoduchá. Program nevyužívá databázi a počítače s jejich MAC adresami si ukládá do souboru *computerlist.ini*, který se nachází ve složce aplikace. Struktura souboru je

```
[computers]
Nazev_PC_nebo_IP_Adresa = MAC:adresa
192.168.0.1 =11:22:33:44:55:66
```

Jednotlivé počítače jsou na jednotlivých řádcích. Takováto struktura souboru dovoluje jednoduchý způsob načtení (*parsování*), pomocí funkcí přímo v jazyce PHP.

Při načtení aplikace se zobrazí seznamu všech připojených počítačů s ikonou jejich statusu – zelená pro zapnuté počítače, šedá pro vypnuté počítače. Zjištění stavu PC se provádí pomocí příkazového řádku, vyvolaného skriptem spuštěným přes AJAX. Využívá se zde jednoduché funkce PING, která je do 1 sekundy ukončena. Pokud do 1 sekundy stanice odpoví, což vzhledem k běžným odpovědím v sítích typu LAN je dostatečně dlouhá doba, pak víme, že stanice je aktivní. Pokud nastane opačný případ, pak je stanice vypnutá. Příkaz se na každou stanici preventivně odesílá každých 5 vteřin, aby došlo k vyloučení chyby, kdy je stanice aktivní, ale například z důvodu zahlcení nestihá odpovědět, či se odpověď ztratí v síti. Toto by mělo rychle ukázat skutečný stav PC.

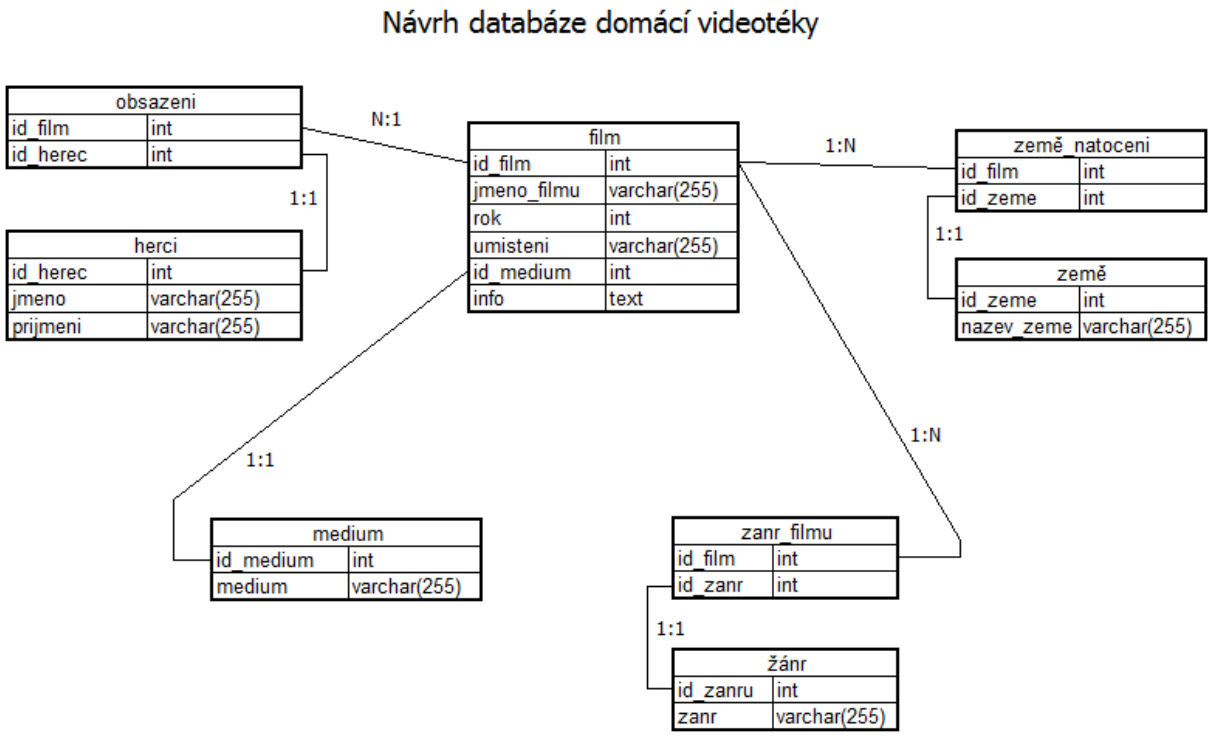


Obrázek 11 - Náhled aplikace Wake-on-lan

Dále se nám v programu nachází možnost Wake-up. Právě tato funkce provede skript, který sestaví Magický paket a posléze jej pomocí protokolu UDP odešle do lokální sítě. Pro správnou funkci je ještě nutné mít na serveru v souboru *php.ini* povolenou práci se sokety.

3.1.2 Databáze domácího videa

Aplikace potřebuje pro svou činnost aktivovanou databázi. Databáze je navržena dle následujícího E-R diagramu:



Jako další vlastnost ke každému filmu je možnost přidat obrázek (*plakát*). Tento obrázek však není uložen v databázi, ale při přidávání filmu se obrázek vybere a při nahrání se automaticky pojmenuje jako ID filmu s povolenou příponou (*PNG* nebo *JPG*). Složka s těmito obrázky se nachází ve složce této rozšiřující aplikace – pojmenovaná „*posters*“.
Z bezpečnostních důvodů bude možné provádět veškeré úpravy databáze pouze z lokální sítě za pomoci lokálních PC, notebooků nebo zařízení smartphone / PDA. Z internetu je však možné, po řádném přihlášení, databázi prohlížet.

Díky této databázové konstrukci je možné provádět vyhledávání v databázi pomocí různých filtrů – například si vyhledat procentuální zastoupení filmů podle země natočení.

3.2 Rozšíření základních aplikací

Rozšíření základních aplikací (*neboli extender-y*) jsou speciálně upravené grafické výstupy z daného modulu pro použití externími programy, které nejsou pro HCP primárně navrženy.

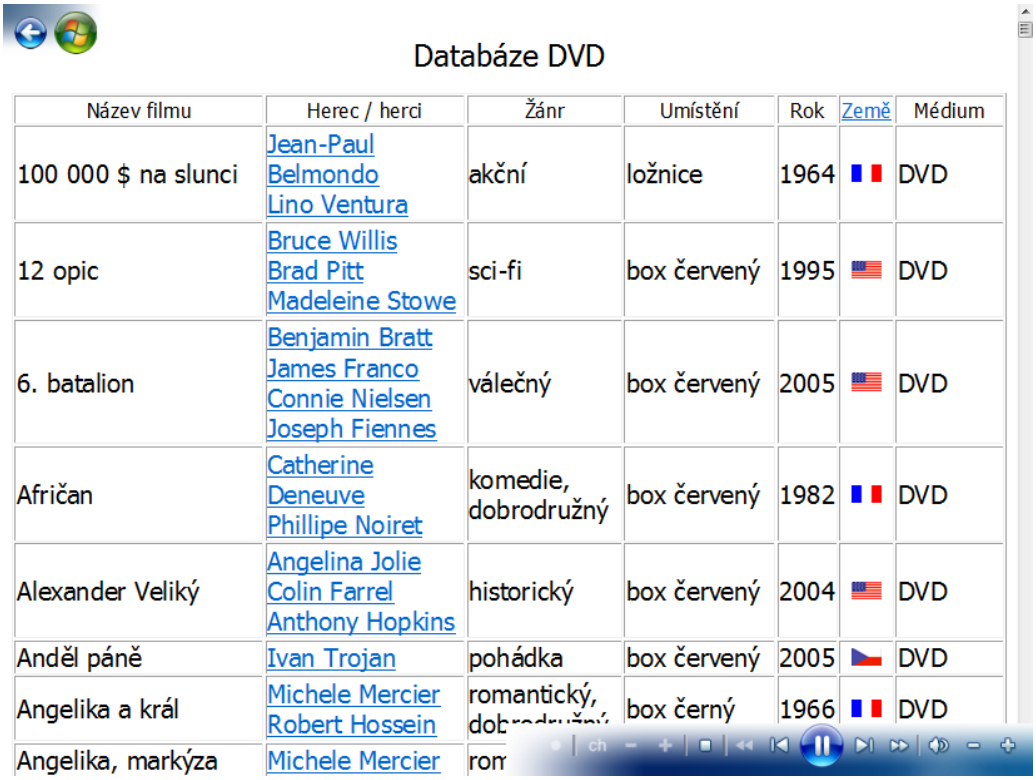
3.2.1 e-mce remote extender

Jedná se o rozšíření pro domácí zařízení typu HTPC (*Home Theatre PC*). S tímto rozšířením je možné s pomocí dálkového ovládání přistupovat k databázi domácího videa přímo z aplikace Media Center. Aplikace Windows Media Center [xi] se objevila v systémech Windows poprvé v roce 2005, kdy firma Microsoft uvolnila speciální vydání systému Windows XP – Media Center Edition 2005. Jednalo se o jádro systému Windows XP Professional, které mělo nadstavbu v podobě aplikace, která uměla přistupovat k multimediálnímu obsahu na disku PC či po lokální síti. Dále měla aplikace univerzální rozhraní pro televizní karty, které ovšem museli splňovat požadavky na certifikaci přímo z Windows Hardware Quality Labs (WHQL[xii]). Vlastní systém Windows XP MCE se v domácnostech neujal, protože nebyl prodáván samostatně a jako OEM verze byl dražší, než domácí verze XP Home Edition. Při příchodu následných Windows Vista tuto aplikaci Microsoft přiložil k variantám systému Home Premium a Ultimate. U poslední verze Windows 7 je pak aplikace již přítomná u všech prodávaných edicí systému (výjimkou je jen verze starter, která se nedá samostatně pořídit a je distribuovaná pouze s velmi levnými notebooky).
Vlastní aplikace Media Center může využívat pro vykreslování obsahu jádra Internet Exploreru, který je v systému přítomný. Díky této vlastnosti je jednoduché připravit *rozšiřující prvek - extender aplikaci*, která zpracuje stránku a vypíše jej s požadovanou velikostí písmen tak, aby byla viditelná a bez problému čitelná na televizi z větší dálky.
Databázi je však nutno plnit pomocí přístupu z lokálního PC. Z bezpečnostních důvodů je úprava/doplňování filmů do databáze omezena pouze na přístup z lokální sítě. Prohlížení je však možné provádět i z internetu.

Obrázek 12 – *e-mce remote extender* přidaný do aplikace Windows Media Center

Důležitou součástí nástavbového programu Windows Media Center je Dálkový ovladač. Je to univerzální dálkové ovládání, které musí splňovat náležité parametry, aby mohlo komunikovat s jednotným IR-přijímačem. Je tedy možné vlastnit IR přijímač jiné značky než ovladač. Důvodem je, že k tomuto přijímači je možné pořídit i bezdrátovou klávesnici, která umožní ovládat PC v plném rozsahu. Dálkový ovladač je však navržen tak, aby bez problémů dokázal plně obsluhovat veškeré části vlastní aplikace.

Při návrhu *extender* aplikace *e-mce remote* jsem vycházel z faktu, že všechny dálkové ovladače se musí chovat stejně a mají stejné funkční prvky. Při orientaci v domácí videotéce je tedy možné využívat tlačítka 1 až 9 jako písmen, jak je uvedeno na obrázku 14. Po stisknutí se obrazovka rovnou posune na písmeno, kterému dané tlačítko odpovídá.



Obrázek 13 – Vlastní zobrazení aplikace přímo na televizi. Posouvání se provádí pomocí dálkového ovládání

Obrázek 14 – Příklad dálkového ovladače pro aplikace Windows Media Center

4. Postup návrhu dalších modulů

Jak již bylo zmíněno, rozšiřující aplikace jsou důležitou komponentou systému. Jejich vytváření se provádí pomocí jazyka PHP, na který může být navázán program napsaný v jiném jazyce. Ten je možno, díky vlastnostem jazyka PHP, vyvolat za pomoci příkazové řádky, kdy je dokonce možné dostat z příkazové řádky výstup a s jeho pomocí dále reagovat dle potřeb.
Programy je možné vytvářet za pomoci dvou základních architektur. První možností je psaní kódu v posloupnosti bez jakýchkoliv objektů. Tento styl programování v jazyce PHP 4.0 převažoval, a proto na něj mohou být někteří programátoři navyklí. Druhou možností je využití základních objektů pro tvorbu MVC aplikace. Následující návody ukáží, jak vytvořit vlastní jednoduchou aplikace „Hello World!“.

4.1 Základní neobjektový návrh

Základní složku s aplikací nazveme „*helloworld*“. Do této složky vložíme soubor „*index.php*“

```
<?php
 if (is_online())
 {
  /* kód stránky */
  echo "Hello world!";
 }
?>
```

Funkce is_online() je definovaná v jádře programu. Vyvolává se bez parametrů a výstupem je hodnota *true* v případě, že je uživatel přihlášen a přihlášení je stále aktivní, nebo *false* v opačném případě. Jak je z tohoto kódu patrné, je tím zabezpečeno, že do programu má přístup pouze uživatel, který je řádně přihlášený. Pokud se někdo pokusí vyvolat program přímo, pak PHP server odpoví, že funkce is_online() není definovaná a skript se opět neprovede.

Program i se složkou umístíme do adresáře „*programs*“ a pro správnou funkci jej buď nainstalujeme, nebo ručně doplníme soubor registrů (*viz kapitola registry*) o řádek

```
Hello World|1.0|programs/helloworld
```

V tuto chvíli je možné program vyvolat z nabídky programy.

4.2 Základní objektově orientovaný návrh

Vytvoříme složku „*helloworld*“. Do této složky umístíme soubory „*HelloworldController.php*“, „*HelloworldView.php*“ a „*view.phtml*“. Jak nám napovídají návrhy stránek, tak soubor view.phtml je šablona, ve které se vytváří vzhled vykreslené stránky. Zapisují se sem také tagy šablonového systému A-Lang 2.0 (*viz dále*).
V jádře systému se nachází základní třídy pro *controller* i *view*. Těchto tříd tedy můžeme s výhodou využít. Jmenují se AbstractView a AbstractController a jsou to jakési předlohy, které obsahují základní funkce, kterých můžeme využít.

Soubor HelloworldController.php:

```
<?php
include_once dirname(__FILE__). "/../../core/AbstractController.php";
class HelloworldController extends AbstractController
{
 public function HelloworldController()
 {
  include_once dirname(__FILE__). "/HelloworldView.php";
  parent::AbstractController(new HelloworldView(dirname(__FILE__)."/view.phtml"));
 }
 public function indexAction($args)
 {
  $view = $this->getView();
  return $view->render();
 }
}
?>
```

Funkce getView() je definována v AbstractControlleru a vrací nám soubor pohledu (*view*). Funkce render() nám zase vytvoří finální stav stránky, který následně vypíšeme. Vymění šablonové tagy za reálný přepis a předá hotový text k vykreslení. V souboru *AbstractView* je navíc schován základní balíček funkcí jazyka A-Lang 2.0, který je možné si libovolně rozšířit, podle potřeb v našem *HelloworldView*.

```
<?php
include_once dirname(__FILE__). "/../../core/AbstractView.php";
class HelloworldView extends AbstractView
{
 public function HelloworldView($filename)
 {
  parent::AbstractView($filename);
 }
 public function hello()
 {
  return "Hello World!";
 }
}
?>
```

V tomto souboru můžeme vidět definici vlastní funkce pro šablonovací systém. Jedná se o jednoduchou funkci hello(), udávanou bez argumentů, která nahradí text za „Hello World“! Vlastní šablona stránky poté vypadá jednoduše:

```
[%hello()%]
```

5. Zkušenosti uživatelů

1. Uživatel 1, Žena 21 let, studentka, v oboru IT začátečník

   „*Ovládání systému je jednoduché, velmi mi vyhovuje možnost změny vzhledu a všeobecně hezké grafické zpracování. Aplikace pro probuzení PC je velmi šikovná.*“

2. Uživatel 2, Muž 32 let, práce v telekomunikační spol., v oboru IT expert

   „*Až na drobné chyby v podobě špatného kódování češtiny a nekompatibilitu mezi prohlížeči je systém graficky a ovladatelně zdařilý. Chybí mi větší množství aplikací, které by ovšem neměl být problém, při správné dokumentaci, doprogramovat.*“

3. Uživatel 3, Muž 25 let, práce v letectví, v oboru IT pokročilý

   „*Systém je hezky zpracovaný a myslím si, že má potenciál pro nasazení v domácnostech díky lehkému ovládání. Pouze postrádám větší množství aplikací.* “

6. Závěrem

V době kompletace této práce se začínaly objevovat první prohlížeče, které plně podporují návrh standardu HTML5. Ten rozšiřuje původní jazyk HTML o nové značky a především, za pomoci adekvátních prohlížečů, podporují i off-line aplikace. Znamená to tedy, že s jazykem HTML můžeme vytvořit aplikaci, která bude funkční i tehdy, nejsme-li připojeni k internetu. Dalšími novinkami budou tagy <*video*> a <*audio*>, které nám dovolí přímo v prohlížeči přehrát multimediální obsah bez jakékoliv speciální aplikace – tzv. *pluginu* (například Adobe Flash [xiii] nebo Microsoft Silverlight[xiv]). Další z velkých novinek bude vykreslování grafiky pomocí Direct2D – součást grafických funkcí DirectX. Toto platí přirozeně jen pro systémy Windows, ovšem ostatní hráči na poli prohlížečů aktuálně experimentují s nasazením speciální verze OpenGL zvanou WebGL[xv], která má krom rychlého vykreslování přinést dokonce i 3D grafické rozhraní pro prohlížeče bez nutnosti *pluginů*. Znamená to, že grafika bude vykreslována přímo grafickým adaptérem a tedy, že webové stránky se svou rychlostí začnou pomalu přibližovat dnešním desktopovým aplikacím. Tohoto by bylo v budoucnu

možné u projektu Home Cloud Platform využít, například pro přehrávání hudby z domácího serveru při jízdě autem a podobně.
Můžeme zde pozorovat, jak současně se snižováním ceny přístupu k internetu se podobné aplikace začínají rozšiřovat. Již dnes je například možné využívat textových editorů bez nutnosti jakékoliv instalace – například Google Docs, Office Live, nebo Oracle Cloud Office, které jsou již dostupné i pro zařízení iPhone. Je tedy vidět, že internet bude stále více prorůstat do životů běžných lidí a pokud někdo chce mít svá data dostupná a současně nechce nechat svá osobní data a dokumenty na serverech velkých společností, pak může být software Home Cloud Platform skvělou alternativou ke jmenovaným službám.

6.1 Splnění cílů vytyčených touto prací

1. **Cílem práce bude prozkoumat aktuální stav oboru a existující aplikace**
   - Existující produkty, které by měly spadat do stejného oboru, jsou popsány v kapitole *Konkurence*. Většinou se však jedná o systémy, které svým využitím směřují mimo rozsah této práce. Jedinou přesnou konkurencí je tedy systém Windows Home Server, který k datu vyhotovení této práce již není na trhu dostupný.

2. **Navrhnout modulární implementaci za použití technologií Java, PHP, nebo ASP.NET + JavaScript a AJAX**
   - Pro implementaci serverové části byl finálně vybrán jazyk PHP, pro implementaci klientské části JavaScript a AJAX. Konečný produkt je funkční a nyní je pouze na programátorech, jaké komponenty k systému vytvoří – systém splňuje slibovanou flexibilitu.

3. **Provést testování s alespoň třemi uživateli**
   - Z reakcí uživatelů je možné usoudit, že systém splnil požadavek na jednoduchost uživatelského rozhraní.

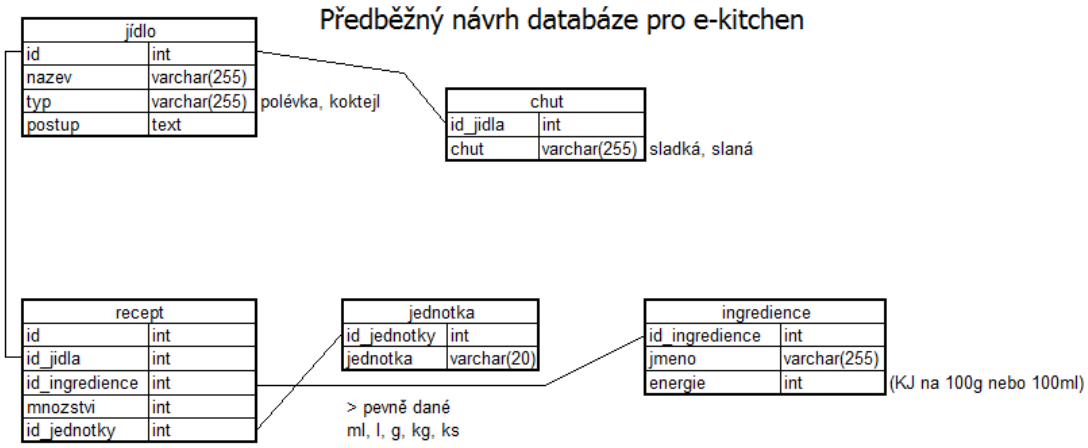
Díky této práci se mi podařilo nahlédnout do moderního způsobu programování. Dozvěděl jsem se o technice Model-View-Controller, která mi do této doby byla neznámá. Taktéž jsem se začal lépe orientovat v objektovém programování, i když zde ještě budu muset zapracovat.
Jedním z velkých problémů bylo vytvoření grafického rozhraní, protože rozhraní, které by splňovalo interaktivitu přesně podle mých představ, vyžaduje v dnešní době nástroje Adobe Flash či MS SilverLight. U návrhu uživatelského rozhraní jsem strávil dlouhou dobu i díky tomu, že jsem vyvíjel verzi, která by se pokusila implementovat alespoň ty funkce, které dnešní prohlížeče z HTML 5 podporují. Bohužel je zde problém nekompatibility mezi prohlížeči, kdy stejné funkce různí výrobci implementují prozatím rozdílně.

6.2 Budoucí možnosti rozšíření

V původním návrhu systému bylo několik užitečných modulů, na které se v současné implementaci nedostalo. Uvádím zde jeden spolu s rozšířením jako příklad budoucích možností rozšíření tohoto systému.

6.2.1 Databáze receptů (e-kitchen)

Tato databáze je navržena dle následujícího předběžného E-R diagramu:



Program by měl být schopen vyhledávat jídla podle kategorií (*prozatím označeno jako typ*) a podle jednotlivých ingrediencí vypočítat celkovou energetickou hodnotu jídla. Další z možností by mělo být zadání ingrediencí, které máme aktuálně k dispozici a kdy program sám vyhledá v databázi, které recepty je možné z těchto věcí vytvořit. Recepty budou uložené v databázi, přičemž bude možné vytváření vlastních receptů.
Jako další zlepšení této služby by byla možnost připojení HCP serverů k jedné centrální službě, která by tyto recepty synchronizovala a rozesílala jej na všechny ostatní. Jednalo by se tedy o externí službu a aplikaci, které by nijak nezasahovaly do jádra systému ani do aktualizací, protože by využívaly vlastní aktualizační (*resp. synchronizační*) komunikační kanál.

6.2.2 e-kitchen remote extender

Jednalo by se o rozšíření předchozí aplikace e-kitchen pro kuchyňské terminály. Základní myšlenkou je vyměnit desítky papírových kuchařek za jeden terminál, který umožní zobrazit všechny recepty, které se na serveru nachází. V dnešní době, kdy přichází elektronický inkoust je navíc možné takovéto terminály provozovat takřka s nulovými energetickými nároky, jelikož *e-ink*[xvi] potřebuje energii pouze na překreslení obrazu. Poté se obraz stane statický a energie již není pro obnovení potřeba – jak známe z technologie LCD. Spojením

této zobrazovací techniky spolu s dotykovou fólií vzniká jednoduchý terminál, který může v kuchyni velmi efektivně sloužit. Další nespornou výhodou technologie *e-ink* je lepší čitelnost textu spolu s velmi vysokým kontrastem.
V dnešní době se objevují první čtečky elektronických knih právě s dotykovým ovládáním, bohužel cenově jsou zatím pro takovéto využití nedostupné.

Příloha A - Application Programming Interface

bool **is_online()**

- Bezparametrová funkce, která nám vrací hodnotu *true* nebo *false* podle toho, zdali je uživatel řádně přihlášený.

```
<?php
 If (is_online())
 {
   /* akce pro přihlášeného uživatele */
 }
 else
 {
   /* akce pro nepřihlášeného uživatele */
 }
?>
```

string **getUser()**

- Bezparametrová funkce vracející nám uživatelské jméno přihlášeného uživatele.

```
<?php
 echo „Vítám uživatele jménem “.getUser();
?>
```

mixed **database([**string *$query*] **)**

- Funkce provede připojení k databázi.
- Pokud je zadán parametr v podobě MySQL dotazu, pak jej provede a na výstupu je pole, ve kterém jsou výstupní prvky.

  Příklady použití

  - Předpokladem tohoto příkladu je MySQL databáze, ve které je tabulka *users* s položkou *usernames*. Ta obsahuje 2 položky – „Alice“ a „Bob“.

    Příklad 1 (*užití pouze pro připojení k databázi*):

```
<?php
 database();
 $q = mysql_query(„SELECT usernames FROM users“);
 $q = mysql_fetch_row($q);
 mysql_close();
?>
```

- Tento kód nám udělá řadu (jednorozměrné pole) o 2 prvcích, kde $q[0] = „Alice“, $q[1] = „Bob“.
- Výstupem funkce je hodnota *true* nebo *false* v závislosti na tom, zdali se podařilo k databázi přistoupit.

Příklad 2 (*stejný příklad užití*)

```
<?php
 $q = database(„SELECT usernames FROM users“);
?>
```

- Tento kód provede stejnou operaci – vytvoří jednorozměrné pole o 2 prvcích, kde opět $q[0] = „Alice“, $q[1] = „Bob“.
- Krom připojení k databázi se taktéž odpojí a prvky automaticky seřadí do jednorozměrného pole.
- Výstupem může být však hodnota *false* v případě, že se k databázi nepodařilo připojit.

- Tato funkce využívá souboru *opendatabase.php*, který je umístěn ve složce *settings* a obsahuje v sobě příkazy *mysql_connect* a *mysql_select_db* s potřebnými parametry.

AbstractController
Třída, která vyvolává akce, které jsou k dané aplikaci (*a tedy i její šabloně*) přidruženy.

- **$view**
  - Proměnná, do které se ukládá pohled, se kterým se pracuje.
  - Je skrytá, tudíž není možné do ní nic zapisovat mimo objekt (*private*).

- **AbstractController($view)**
  - Funkce je konstruktor.
  - Funkce, která se provede při inicializaci objektu a zapíše pohled, se kterým se pracuje do proměnné **$view**.

- string **getView()**
  - Bezparametrová funkce, která vrací obsah proměnné view.

- array **dispatch(**string *$action*, array *$args***)**
  - Funkce volající danou akci s požadovanými argumenty.

AbstractView
Třída pracuje s načtenou šablonou, která je uložena v textovém souboru typu HTML a využívající šablonovacího jazyka *A-lang 2*. Výstupem je vyhotovená stránka.

- **$templateFileName**
    - Proměnná, do které se ukládá název souboru pohledu – *tzv. view teplate*.
    - Je skrytá, tudíž není možné do ní nic zapisovat mimo objekt (*private*).

- **$template**
    - Proměnná, která uchovává obsah souboru pohledu (šablonu), (*opět private).*
- **AbstractView($filename)**
    - Funkce je konstruktor.
    - Funkce, která se provede při inicializaci objektu a zapíše název souboru pohledu do proměnné **$templateFileName**.

- string **getTemplate()**
    - Pokud není načtená pohledová šablona, pak nám jí funkce načte.
    - Výstupem je právě tato šablona.

- string **loadTemplate(**string *$filename***)**
    - Funkce načte obsah šablony do proměnné **$template**.

- string **fillTemplate()**
    - Funkce načte šablonu, vyhledá řetězce náležící šablonovému jazyku *A-lang*, provede jejich výměnu za požadované výstupy a převede výstupní řetězec na *utf-8* kódování.
    - Výstupem je hotová stránka připravená pro vykreslení.

- string **render()**
    - Vyvolá funkci **fillTemplate()**.
    - Výstupem je hotová stránka připravená pro vykreslení.

- array **getPlaceholdersFromTemplate(**string *$template***)**
    - Výstupem funkce je pole všech prvků jazyka *A-lang*, které jsou v dané šabloně přítomné.

- string **substitutePlaceholders(**string *$template*, array *$placeholders***)**
    - Výstupem funkce je vyhotovená stránka v kódování, ve kterém je uložený soubor s šablonou.

Reference

[i] Windows Home Server - http://www.microsoft.com/windows/products/winfamily/windowshomeserver/default.mspx
[ii] Apache - http://www.apache.org/
[iii] MVC architektura - http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html
[iv] Zend Framework - http://framework.zend.com/
[v] CakePHP - http://cakephp.org/
[vi] EasyPHP - http://www.easyphp.org/
[vii] Net Applications - http://www.netapplications.com/
[viii] Šablonovací jazyk Smarty - http://www.smarty.net/
[ix] Šablonovací jazyk TemplatePower- http://templatepower.codocad.com/
[x] Microsoft SkyDrive - http://www.skydrive.com
[xi] Windows Media Center – http://windows.microsoft.com/cs-CZ/windows7/products/features/windows-media-center
[xii] Windows Hardware Quality Labs - http://www.microsoft.com/whdc/winlogo/default.mspx
[xiii] Adobe Flash - http://www.adobe.com/products/flash/
[xiv] Microsoft Silverlight - http://www.silverlight.net/
[xv] WebGL - http://www.khronos.org/webgl/
[xvi] E-ink - http://www.eink.com/