VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

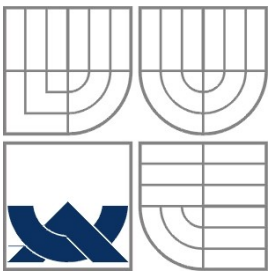
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TVORBA APLIKACÍ PRO IGOOGLE

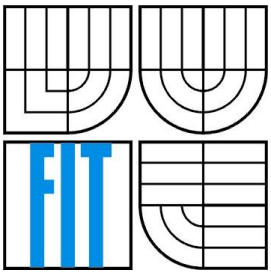
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE ZDENĚK HÁJEK
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TVORBA APLIKACÍ PRO IGOOGLE

DEVELOPMENT OF IGOOGLE APPLICATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

ZDENĚK HÁJEK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. JAROSLAV RÁB

BRNO 2008

Abstrakt

Tato práce se věnuje popisu technologie iGoogle, analýze, návrhu a samotnému vývoji aplikace, sloužící pro evidenci času stráveného nad úkoly nebo projekty. V technické zprávě jsou také uvedeny zajímavé poznatky z vývoje aplikace a na konci shrnuty výsledky práce s technologií iGoogle.

Klíčová slova

iGoogle, gadget, Google API, project time management, evidence času, JavaScript

Abstract

This thesis describes the iGoogle technology, analysis, design and developing of application for managing and tracking time spent on taks or projects. Thesis also involves interesting pieces of knowledge gathered during developing and by the end results of working with the iGoogle technology.

Keywords

iGoogle, gadget, Google API, project time management, time tracking, JavaScript

Citace

Zdeněk Hájek: Tvorba aplikací pro iGoogle, bakalářská práce, Brno, FIT VUT v Brně, 2008

Tvorba aplikací pro iGoogle

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jaroslava Rába. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum

Obsah

Úvod

Neustálý rozvoj webu přináší nové technologie. Uživatelům přestává stačit statický a neinteraktivní obsah webových stránek. Je požadována dynamičnost a interaktivita. Tento požadavek se velkou měrou naplnil s příchodem technologie nazývané jako Web 2.0.

Filosofií Webu 2.0 je pochopení webu jako platformy pro interaktivní aplikace, které můžou být sdíleny mezi uživateli. Kdokoliv je může vyvíjet a nabídnout ostatním. Web se tak rozšiřuje o jistý sociální fenomén, kdy obsah webu je tvořen uživateli.

Této myšlenky se ujala americká společnost Google a vyvinula tzv. iGoogle. Tato práce se věnuje tomu co vlastně iGoogle je a co nabízí jak uživatelům tak vývojářům. Je zde představen a popsán kompletní vývoj aplikace a také popis v jaké formě se aplikace šíří. Nezbytnou součástí je popis problémů a omezení, na které jsem při vývoji narazil. V jisté části se věnuji také popisu grafického rozhraní, které používají aplikace pro iGoogle.

Na závěr jsem shrnul své poznatky ze setkání s technologií iGoogle, zhodnotil výsledky své práce a popsal v čem spatřuji budoucnost interaktivních webových aplikací.

1 Technologie iGoogle

V této části se věnuji popisu technologie iGoogle a vysvětlení souvisejících pojmů. Je zde také popsána tato technologie jak z hlediska člověka, který ji používá, tak z hlediska vývojáře, tedy člověka, který pomocí ní aplikace tvoří.

1.1 Co je to iGoogle?

Firma Google se proslavila svým vyhledávačem. Webová stránka tohoto vyhledávače je vytvořena jednoduše a funkčně, v podstatě nabízí jen formulářové pole pro zadání hledaného výrazu. Obsahuje tedy spoustu nevyužitého místa. iGoogle umožňuje toto místo vyplnit různými interaktivními aplikacemi.

Jde tedy o velice chytrý tah ze strany společnosti Google, která tak využívá toho, že spousta uživatelů internetu má nastavenu jako svou výchozí stránku v prohlížeči právě Google. Pokud se vedle tohoto faktu dá skutečnost, že roste počet lidí požadujících od webu interaktivitu, získáme slova interaktivita a Google, tedy iGoogle.

1.1.1 Interaktivita

Interaktivní web si lze představit jako webovou stránku, která umožňuje, ba dokonce přímo vyžaduje zásahy ze strany uživatele.

V případě iGoogle je interaktivita zajištěna malými aplikacemi, které mohou být umístěny na titulní stránku vyhledávače Google. Takovéto malé aplikaci se říká gadget. Gadgety mohou plnit různou funkci. Kalkulačka, kalendář, plánovač, poznámky, měřič srdečního tepu, to jsou příklady některých gadgetů. Z toho je patrné, že nabídka je skutečně pestrá.

1.1.2 Gadget

Slovo gadget se do češtiny překládá doslovně jako důmyslný přístroj, pomůcka nebo vtipná věc. Nejvhodnějším termínem se v našem případě zdá býti slovo pomůcka.

Je nutno podotknout, že myšlenka použití gadgetů není nikterak nová. Sám gadget se totiž dá chápat také jako součást grafického uživatelského rozhraní. Poprvé se gadgety objevují jako doplňky desktopových prostředí některých uživatelsky zaměřených operačních systémů. Zde se také někdy vyskytují pod názvem widget. Dají se nalézt v operačním systému Apple Macintosh pod názvem Dashboard, v operačním systému Microsoft Windows Vista jako Microsoft gadgets a taktéž v operačním systému Linux kde existuje několik různých jako např. SuperKaramba, gDesklekts a další.

Průkopníkem je však firma Apple se svým operačním systémem, kde se poprvé objevuje jistá forma gadgetů a to v podobě tzv. Desk accesories [1].

Za zmínku také jistě stojí to, že Google poprvé použil gadgety ještě před samotným spuštěním iGoogle. Jednalo se o desktopovou aplikaci, která patřila do kategorie tzv. widget enginů a měla za cíl umožnit použití desktopových gadgetů tam, kde to nepodporoval operační systém nativně, například Microsoft Windows Xp. Google s tímto svým produktem slavil úspěch a tak toto know-how přenesl na web a tím splnil požadavek na interaktivitu webových stránek.

příklad iGoogle gadgetu

1.1.3 Personalizace

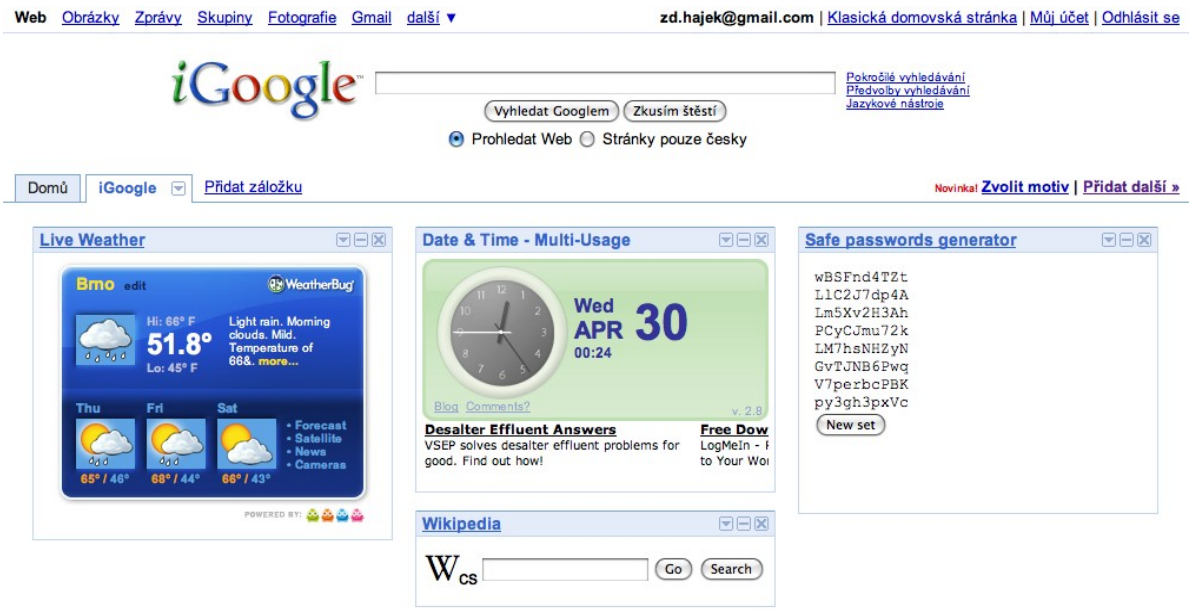
Interaktivní web tedy sestává z gadgetů, které si uživatelé umisťují na svou stránku. Zde chci zdůraznit fakt, že stránku iGoogle si každý uživatel sestavuje sám a podle individuálních přání.

Vznikají tedy webové stránky obsahující aplikace, které si uživatel sám vybral. Toto řešení je velice oblíbené a je velmi pravděpodobné, že řada uživatelů používá takto svou personalizovanou stránku jako svou homepage nebo-li výchozí stránku ve svém internetovém prohlížeči. Jednoduše také proto, že tato stránka splňuje přesně to co se od ní očekává a nemalou roli hraje jistě také skutečnost, že si ji vytvoří každý člověk sám a bere ji tak za vlastní.

Nepochybně obrovskou výhodou je fakt, že personalizovaná stránka iGoogle je dostupná všude tam kde je k dispozici internet a webový prohlížeč. Uživatelé jsou identifikováni pomocí svého e-mailu, který slouží zároveň jako přihlašovací jméno. Není tedy problém používat svou upravenou

stránku s gadgety na různých místech a také ji na různých místech libovolně upravovat. Díky přihlašování a potažmo identifikaci uživatelů, kterým je jako registrovaným, možné vyhradit datový prostor, je zajištěna perzistence vytvořené stránky.

Gadgety jsou umístěny ve veřejném repositáři a každý registrovaný uživatel k nim může přistupovat a přidat si je na svou stránku. Gadgety jsou v repositáři rozděleny do příslušných kategorií jako například Zprávy, Nástroje, Komunikace a podobně.



personalizovaná stránka iGoogle

1.2 Technické řešení

Jelikož se jedná o webovou technologii, používají se jako hlavní prostředky k realizaci takové nástroje a programovací jazyky, aby splňovaly nutnost běhu ve webovém prohlížeči, měly nízké nároky na síťový přenos a aby nemusel uživatel doinstalovávat jakékoliv potřebné knihovny nebo frameworky.

Každý gadget je sám o sobě tvořen pouze jedním jediným souborem. Jedná se o strukturovaný dokument značkovacího jazyka XML. Použití formátu XML je pochopitelné, protože je široce podporován ze strany browserů a je tedy zaručena kompatibilita.

Tento XML soubor by se dal rozdělit do tří významných částí. První částí jsou tagy (značky), které slouží k identifikaci gadgetu, deklaraci lokalizačních souborů a deklaraci uživatelských nastavení (user preferences). Tato první část se vyskytuje na začátku zdrojového souboru a jedná se svým způsobem o jistou deklarační hlavičku, která je nezbytná. Deklarační část obsahuje také

informace o tom jaké knihovny z iGoogle API se mají použít. Další částí je popis uživatelského rozhraní. Uživatelské rozhraní je tvořeno pomocí kombinace jazyka HTML a kaskádových stylů CSS. Třetí a velice významnou částí je samotná funkcionalita, která je zajištěna jazykem JavaScript. Popis uživatelského rozhraní a popis funkcionality jsou od hlavičky odděleny umístěním do XML tagu CDATA.

Když shrneme použité technologie, získáme XML, HTML, CSS a JavaScript. Z tohoto je patrné, že se nejedná o nijak výjimečné prostředky, ale o osvědčené stavební kameny webu.

1.2.1 Uživatelská nastavení

Jak jsem již uvedl, gadgety obsahují tzv. uživatelská nastavení (user preferences). Tyto jsou určeny pro ukládání různých uživatelských nastavení gadgetu. Pokud vývojář gadgetu umožní uživateli měnit například barvu gadgetu a chce aby tato barva byla při příštím načtení stránky zachována, použije k tomu právě uživatelská nastavení.

Těchto nastavení může být libovolný počet a každé uživatelské nastavení může být typu řetězec, číslo, booleovská hodnota (true nebo false), výčet (enum) a nebo jedním zvláštním typem, který je označen jako *hidden*. Jedná se tedy, svým způsobem, o obyčejné proměnné, které mají sloužit k ukládání nastavení. Tyto proměnné mají smysl zejména proto, že v rámci zdrojového souboru gadgetu k nim lze přistupovat z části popisující funkcionalitu, tedy z JavaScriptu. Všechny typy kromě *hidden* jsou dostupné z menu gadgetu a jsou určeny výhradně pro nastavení různých vlastností gadgetu.

Pokud jde právě o speciální typ *hidden*, ten je, jak již sám název napovídá, pro uživatele skrytý a není k němu žádná možnost přímého přístupu. Toto chování se může hodit v případě kdy gadget generuje data a ty je potřeba ukládat. Programátor k nim má přístup stejně jako k ostatním typům uživatelských nastavení.

Je tedy patrné, že možnost používání uživatelských nastavení (user preferences) je klíčovým prvkem k zajištění datové perzistence, jelikož se ukládají přímo na servery Googlu. Uživatel se díky tomuto může pohybovat mezi různými počítači nebo browsery a bude mít k dispozici stále svoje data.

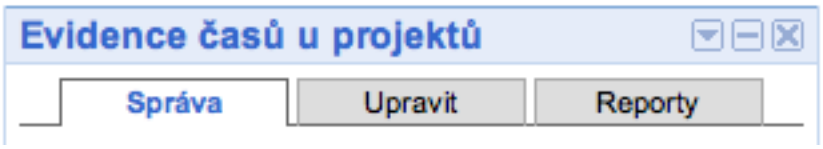
1.2.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní je stejně jako u klasických webových stránek vytvořeno pomocí jazyka HTML v kombinaci s kaskádovými styly CSS.

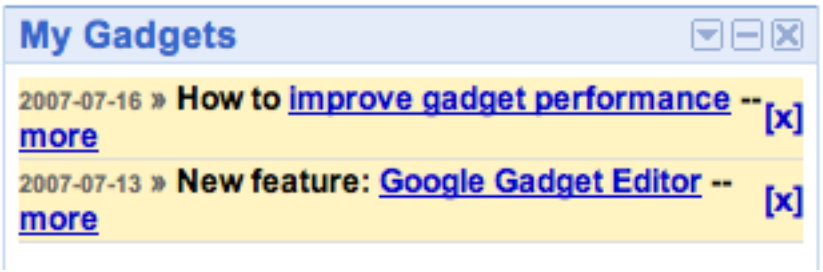
Programátor při tvorbě grafického rozhraní není nijak omezen a může použít veškeré elementy jako u klasické webové stránky včetně obrázků nebo například multimediální technologie FLASH. Jen je třeba mít na paměti, že vytvořený gadget může být používán velkým množstvím lidí,

proto je třeba uvážlivě používat rozsáhlejší grafické prvky kvůli tomu, že použití takových prvků může vytvářet velký síťový přenos.

Gadgety navíc nabízí podporu pro některé zajímavé prvky grafického uživatelského rozhraní. Jedná se o záložky (tabs), drag&drop elementy a tzv. MiniMessages. Tyto uvedené se dají vytvářet pomocí JavaScriptu, jelikož jsou součástí iGoogle API, které nabízí příslušné objekty a metody.



záložky (tabs)



MiniMessages

1.2.3 Funkcionalita

K tomu aby gadget vykonával nějakou činnost je potřebné aby obsahoval kód, který bude tuto činnost provádět. Tímto kódem je programovací jazyk JavaScript. Jazyk JavaScript je objektově orientovaný skriptovací jazyk, který má v současnosti pevné místo v klientsky zaměřených webových aplikacích.

Google gadgets API obsahuje několik knihoven pro JavaScript. Hlavní z nich je tzv. *core* knihovna, která obsahuje některé základní funkce. Tato knihovna je načítána pokaždé a programátor ji nemusí explicitně uvádět v hlavičce. Dalšími knihovnami jsou knihovny pro práci s výše uvedenými prvky grafického uživatelského rozhraní nebo např. důležitá knihovna pro práci s uživatelskými nastaveními (user preferences). Podrobný popis knihoven je uveden v Google Gadgets API Reference [2].

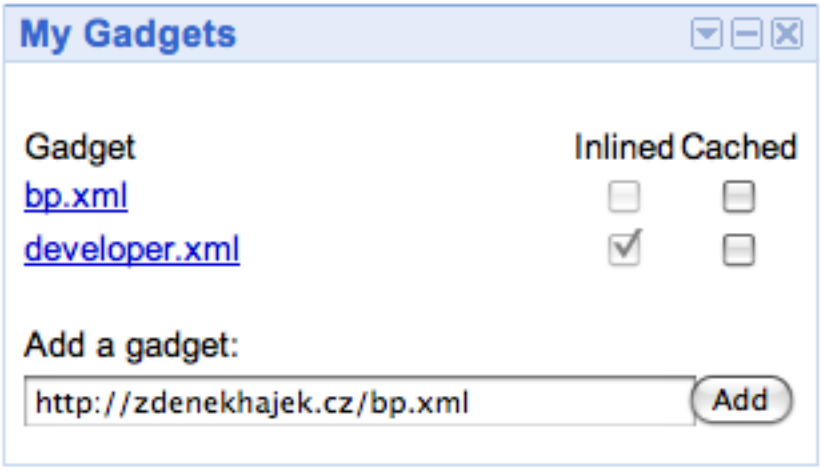
1.2.4 Vývoj gadgetu

Z toho co jsem doposud uvedl je patrné, že gadget je tvořen jedním XML souborem, a který používá dříve zmíněnou kombinaci XML, HTML, CSS a JavaScript. Tyto technologie nepožadují použití

žádného speciálního vývojového prostředí, ale stačí obyčejný textový editor se zvýrazňováním syntaxe.

Vyvíjený gadget se umístí na webu na místo, ke kterému je možný přístup (webhosting). Je možné využít i například služby Google Code, která funguje jako hosting pro různé open source projekty a umožňuje dokonce správu verzí (revision control) pomocí Subversion.

Aby nebylo pokaždé, za účelem otestování gadgetu, nutné nahrávat svůj výtvor do repositáře gadgetů, je k dispozici tzv. Developer Gadget, který si vývojář umístí na svou stránku. Tento gadget mu pak umožní přidávat gadgety z libovolné URL. Jen je nutné si při vývoji dát pozor na checkbox *Cached*, který zajišťuje uložení gadgetu do cache a následné změny provedené ve zdrojovém souboru by se při obnovení v prohlížeči neprojevily.



Developer gadget

Nezbytnou součástí vývoje jakéhokoliv softwaru je ladění (debugging). Google vývojářům neposkytuje žádný nástroj pro ladění a proto je nutné si pomoci externí aplikací. Může to být například produkt, který nese jméno Firebug a který je distribuován jako rozšíření (extension) pro prohlížeč Mozilla Firefox. Tato neocenitelná pomůcka umožňuje například krokování JavaScriptového kódu, sledování obsahu proměnných, sledování požadavků (GET/POST) na stránce a především upozornění na případné chyby ve zdrojovém kódu včetně zobrazení čísla řádku s chybou. Těžko si lze představit vývoj komplexnějších aplikací bez tohoto nástroje.

2 Analýza

V této kapitole podrobně rozeberu a analyzuji zadání. Proces analýzy spočíval především ve vyjasnění si jednotlivých požadavků se současným přihlédnutím na možnosti, které programátorovi iGoogle poskytuje.

2.1 Rozbor zadání

Hlavním cílem je vytvořit iGoogle gadget, který bude sloužit podobně jako aplikace, které jsou označovány jako *project time management* nebo *project time tracking*. Tato aplikace bude umožňovat také generování hlášení (reportů) za určité období.

2.1.1 Project time management

Abych mohl důsledně zpracovat zadání bylo nutné nejprve přesně zjistit jak pracují *project time management* programy a co nabízejí uživateli, který je používá.

Obecným principem, na základě kterého tyto programy pracují, je sledování času, který strávil uživatel u počítače. Konkrétněji u projektů, které tvořil za pomocí počítače. Jedná se tedy o případy kdy člověk, který se věnuje například programování, psaní dopisu nebo jakékoliv jiné činnosti na PC, potřebuje vědět kolik této aktivitě věnoval času. Postup ke zjištění požadovaného času bývá takový, že si uživatel spustí *project time management* program, do něj uvede název projektu nebo činnosti, které se bude věnovat a stisknutím spouštěcího tlačítka se začne počítat čas. Během počítání se uživatel dál věnuje své práci. Po ukončení jen stiskne tlačítko pro ukončení počítání času a uplynulý čas se uloží.

Toto je klíčovým principem všech programů na sledování času a zároveň také hlavním principem mého iGoogle gadgetu.

2.1.2 Technologie pro perzistentní uložení dat

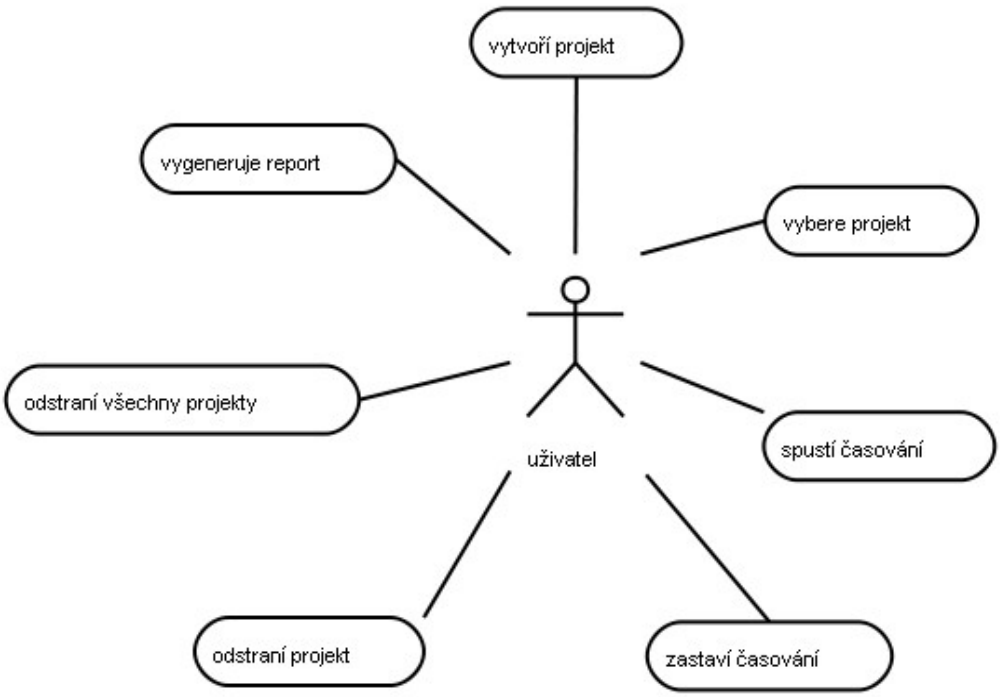
Z výše uvedeného je zřejmé, že výsledná aplikace generuje jisté množství dat, u kterých je potřeba perzistentního uložení. V zadání je požadováno využití některých dostupných technologií Google jako např. Google Spreadsheet, Google Notebook nebo Google Base. Tyto služby mají zveřejněné své programátorské rozhraní (API). V tomto API je zveřejněno jak k těmto službám přistupovat a jak je využívat z různých programovacích jazyků včetně JavaScriptu, který je v mém případě hlavním programovacím jazykem.

2.1.3 Generování reportů

Jelikož má aplikace generovat hlášení za určitý časový úsek (reporty) jejichž obsahem má být přehledné vyjádření čemu se uživatel v jednotlivých dnech věnoval, bude nutné vytvořit prvek pro práci s časem. Tento prvek by měl být také především k dispozici pro měření časů strávených u projektů, tedy časového intervalu mezi stisknutím tlačítka pro spuštění a zastavení počítání. Bude tedy nutné implementovat jistou formu časovače, který zajistí dostatečně přesné měření času.

2.2 Případy užití

Z analýzy zadání vyplynula množina činností, které může uživatel provádět. Tato množina činností je shrnuta v následujícím diagramu případů užití (use case diagram).



use case diagram

3 Dekompozice a návrh

Analyzované zadání je dále nutné rozdělit na jednotlivé části, které budou řešeny postupně. Výsledná aplikace bude tvořena právě spojením těchto dílčích části.

Nejvhodnějším rozdělením rozboru zadání se mi jevilo rozdělení na tři významné části. Jedná se o grafické uživatelské rozhraní, interní práce s daty a interní práce s časem.

3.1 Grafické uživatelské rozhraní

Protože aplikace pracuje v grafickém prostředí je nutné vytvořit grafické rozhraní, které zprostředkovává kontakt mezi uživatelem a programem.

Grafické rozhraní by mělo být jednoduché, přehledné a bez komplikací zpřístupňující všechny dostupné funkce gadgetu. Při vývoji bude také důležité zohlednit skutečnost, že grafické rozhraní je vytvářeno jen na základě webových technologií HTML a CSS.

Jelikož aplikační rozhraní nabízí možnost použití tzv. záložek (tabs), rozhodl jsem se rozdělit gadget do tří částí, které jsem pojmenoval Správa, Upravit a Reporty. Toto řešení jsem zvolil z důvodu prostorové šetrnosti, kterou nabízí. Není nutné mít všechny prvky v jedné velké části, ale díky záložkám je možné mít několik menších částí a mezi nimi přepínat.

3.1.1 Správa

Tato část by měla být hlavní částí aplikace a měla by zpřístupňovat klíčové ovládací prvky. V mém případě se tedy jedná především o výběr projektu a o tlačítka pro spuštění a zastavení počítání času. Dále by mělo být obsaženo viditelné znázornění právě vybraného projektu a také nepochybně celkový čas strávený na právě zvoleném projektu.

3.1.2 Upravit

Tato záložka by se také dala nazvat Editační. Z názvu je zřejmé, že tato část je určena různým úpravám, které může uživatel provádět. Do této sekce jsem zahrnul vytváření nových projektů a smazání projektů. Smazat lze buď jeden vybraný projekt nebo všechny projekty.

3.1.3 Reporty

Tato sekce bude sloužit k možnosti vygenerování reportů. Může také obsahovat další statistické informace jako například celkový objem uživatelských dat, celkový počet projektů nebo také celkový čas strávený na projektech.

3.2 Interní práce s daty

Tato část je uživateli skryta, takže s ní nemůže přímo pracovat. Nicméně základní povinností bude vytvořit určitou strukturu dat tak aby se daly pohodlně zpracovávat. Tato struktura by měla zajistit potřebné členění a vytvořit hierarchii, ze které by bylo jasně patrné rozdělení na projekty a jejich časy. Datová struktura by měla být tedy nejlépe buď přímo XML, což je jazyk pro ukládání strukturovaných dat nebo velice blízká XML.

Aby bylo možné s touto strukturou dat pracovat, bude důležitou součástí vytvoření dostatečně robustních funkcí, které umožní ukládání, modifikaci a odstraňování dat z této struktury.

3.2.1 Omezení

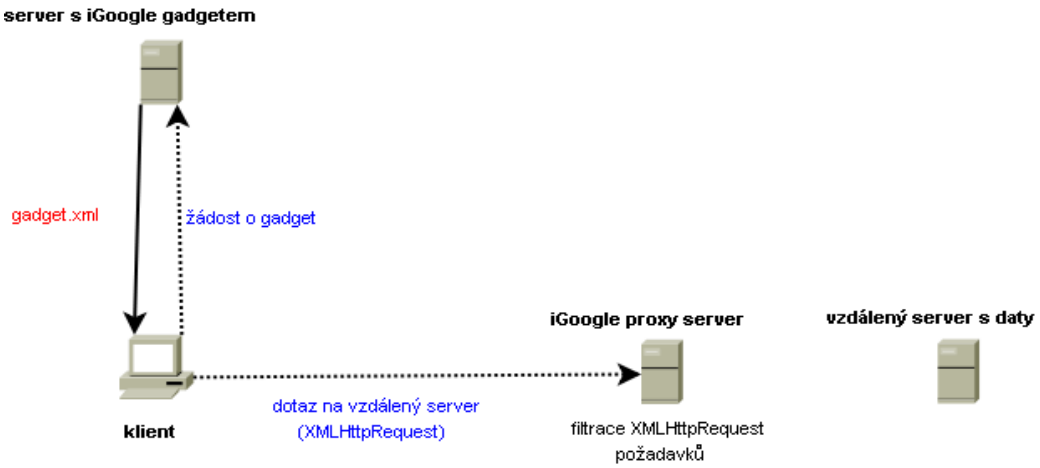
Na tomto místě se chci pozastavit a upozornit na velice důležité omezení, na které jsem při návrhu aplikace narazil.

V zadání je původně pro ukládání uživatelských dat požadováno použití některé dostupné technologie jako např. Google Spreadsheet, Google Notebook nebo Google Base. Jak jsem již dříve uvedl, tak jsou k dispozici aplikační rozhraní (API) k některým těmto službám. Například rozhraní Google Base, což je ve své podstatě klasická databáze, je dobře zdokumentováno i pro případ použití v programovacím jazyku JavaScript. A skutečně u běžných webových aplikací ho lze používat, ale bohužel nikoliv v případě iGoogle. Pokusím se vysvětlit proč.

Získávání dat pomocí JavaScriptu z technologií jako např. Google Base se děje užitím technologie, která se nazývá AJAX nebo-li Asynchronous JavaScript and XML tedy Asynchronní JavaScript a XML. Princip této technologie spočívá v tom, že je možné z webové stránky pomocí JavaScriptu, resp. použitím JavaScriptového objektu, který se nazývá XMLHttpRequest, získávat data ze vzdáleného serveru.

Zjednodušeně se tedy jedná o žádost klientského JavaScriptu o zaslání dat ze vzdáleného serveru. Právě v tomto vězí hlavní problém. Veškeré tyto požadavky musí v případě iGoogle procházet přes proxy server Googlu. Tento proxy server pracuje jako prostředník mezi klientem a serverem. Překládá tedy požadavky od klienta serveru a sám vůči serveru vystupuje jako klient [3]. Toto by samo o sobě nevadilo, ovšem další činností tohoto proxy serveru pro iGoogle je filtrování některých požadavků z bezpečnostních důvodů. Nejvýznamnějším bezpečnostním důvodem je ochrana proti tzv. Cross-site skriptování.

Cross-site skriptování je metoda narušení webových stránek [4]. Toto narušení se děje zejména tím, že útočník podvrhne svůj kód v JavaScriptu, který nic netušící klient spustí a útočník se tak může dostat například k citlivým údajům.



schema průběhu požadavků

A právě kvůli tomuto omezení, které znemožňuje asynchronní požadavky na vzdálené servery, je nemožné využít pro uložení dat jakýkoliv jiný cizí server. Důvodem je tedy skutečnost, že se tyto požadavky ke vzdáleným serverům vůbec nedostanou. Tento fakt pro mě znamenal změnu celé koncepce ukládání dat.

3.2.2 Řešení omezení

Z výše uvedeného důvodu není možné použít externí úložiště pro uživatelská data. Jediným řešením, které se nabízí jsou již dříve zmíněné uživatelské nastavení (user preferences), které slouží jako jakési perzistentní proměnné přímo v rámci gadgetu. Nevyskytuje se u nich tedy žádné volání vzdáleného serveru, ale vše je řešeno v rámci iGoogle.

Jistou nevýhodou je absence možnosti vytvoření jakékoliv dynamické struktury, která je pro ukládání dat nutná. Krom jednoduchých typů jako celé číslo nebo řetězec je k dispozici výčtový typ (enum), který sice nabízí určité prvky struktury, avšak tato struktura musí být pevně definována v hlavičce a takto vytvořenou ji není již možno v průběhu programu měnit.

Požadavek na dynamickou strukturu, kterou by bylo možné v průběhu modifikovat jsem chtěl vyřešit použitím skrytého uživatelského nastavení typu řetězec, který bude obsahovat klasickou XML strukturu. Ve finále by tedy uvnitř JavaScriptu bylo možné pracovat s XML objektem a s veškerými metodami, které JavaScript poskytuje, přičemž při uložení by se tento objekt serializoval na řetězec, který by se uložil do skrytého uživatelského nastavení. Bohužel i zde jsem částečně narazil, tentokrát na omezení velikosti uživatelských nastavení, které je omezeno na 2 kB. Toto omezení je zřejmě způsobeno tím, že se uživatelská nastavení získávají pomocí GET požadavku z URL. Právě kvůli neschopnosti některých webových prohlížečů a zřejmě i webových serverů pracovat s URL delší než 2048 znaků je patrně omezena velikost uživatelských nastavení. Jelikož jsou 2 kB velice málo, zavrhl

jsem použití formátu XML a rozhodl se použít jiný strukturovaný formát, který bude méně náročný na místo a tím pádem stručnější než formát XML. Odpověď na tento požadavek jsem nalezl ve formátu JSON.

3.2.3 JSON

JSON je zkratka pro JavaScript Object Notation a jedná se o strukturovaný formát pro výměnu dat, který je v textové podobě a je tak přímo čitelný [5]. Textová podoba umožňuje převedení objektové struktury do prostého textového řetězce a jeho následné uložení.

Oproti XML má především tu výhodu, že je stručnější a tudíž vhodnější pro můj případ kdy jsem omezen velikostí uložených uživatelských dat. Tato výhoda je dána především tím, že XML je daleko komplexnější než JSON zejména v tom směru, že XML je značkovacím jazykem (markup language).

Nevýhodou použití JSON je nulová nativní podpora ze strany webových prohlížečů. Veškerou podporu je tedy nutné dodat pomocí externích JavaScriptových knihoven.

3.3 Interní práce s časem

Součástí práce s časem by mělo být vytvoření způsobu, který by umožnil jednoduchou a přitom pohodlnou práci s časy a také daty. Zároveň by měl zvolený formát počítat s tím, že se bude ukládat do datové struktury k jednotlivým projektům a tudíž by měl být ve všech funkcích jednotný.

Pro co nejjednodušší manipulaci s časem je v návrhu zvolena reprezentace času ve vteřinách. Tímto je usnadněna práce, jelikož takový čas je zapsán jen prostřednictvím jednoho čísla. Použití této reprezentace času ve vteřinách bude obnášet také vytvoření funkce, která převede tento čas do čitelnější podoby a to konkrétně do formátu HH:MM:SS, tedy např 14:12:54.

Čas je nutné také nějakým způsobem počítat. Nezbytnou součástí aplikace tedy musí být část, která zajistí měření uplynulého času mezi dvěma okamžiky. Tuto funkci by měl zastávat časovač, který zajistí počítání času. Výhodou takového časovače by byla vlastnost kdy by bylo možné kdykoliv v průběhu počítání provést znovunačtení (reload) iGoogle stránky a tedy i obnovení gadgetu a přitom by nebylo přerušeno počítání.

Pro vytváření reportů bude důležité vypočítávání časových intervalů. V těchto intervalech pak bude zobrazena délka práce na projektu. Pokud bude interval např. týden, bude nutné zajistit vytvoření funkcí, které provedou rozdíl dvou různých dat a zjistí zda jejich rozdíl je v délce týdne.

4 Implementace

Tato část je věnována popisu implementací jednotlivých částí tak jak jsou uvedeny v návrhu a taktéž případným problémům. Na samý úvod uvedu bližší popis programovacích jazyků, nástrojů a prostředků, které byly při vývoji použity.

4.1 Programovací jazyky

Jak jsem již dříve uvedl, gadgety jsou postaveny na třech jazycích – HTML, CSS a JavaScript. Tento výčet jazyků je pochopitelný, protože samotný vývoj gadgetu není nepodobný vývoji jakékoliv jiné webové aplikace resp. aplikaci běžící ve webovém prohlížeči

4.1.1 HTML

HTML (HyperText Markup Language) je jazykem pro výtváření webových stránek, který je založen na značkovacím jazyku SGML (Standard Generalized Markup Language). Skupina značkovacích jazyků založených právě na SGML tvoří majoritní část jazyků, které jsou v současnosti používány pro tvorbu webových stránek. HTML při svém vývoji prošlo několika verzemi, přičemž poslední stabilní je verze 4.01 [6].

HTML slouží v případě gadgetů k vytvoření grafického uživatelského rozhraní a potažmo k vytvoření různých částí, které budou určeny k zobrazování výstupních hodnot z funkční části gadgetu. Při implementaci jsem dodržoval specifikaci HTML 4.01, která je se ukázala jako dostatečná k tvorbě rozhraní a je u ní jistota kompatibility u drtivé většiny současných webových prohlížečů. Je pochopitelně možné použít i novější jazyk XHTML, který je odvozen od XML, nicméně v případě gadgetů nejsou rozdíly mezi těmito jazyky pro vývojáře až tak zásadní. Toto je zejména z toho důvodu, že se u gadgetů neuvádí hlavička se specifikacemi, tzn. tag `<head>`, ale používá se jen tělo (`<body>`).

4.1.2 CSS

CSS (Cascading Style Sheets) nebo-li kaskádové styly. Jedná se o prostředek k určení formátování rozhraní vytvořeného pomocí jazyka HTML a tedy i k vytvoření jistého vizuálního stylu.

Nespornou výhodou přístupu s použitím CSS je oddělení samotné struktury uživatelského rozhraní od jejího vzhledu. Toto dále umožňuje nastavovat vzhled z jednoho místa ve zdrojovém kódu. Při tvorbě kaskádových stylů jsem se nesetkal s problémy, které by byly způsobeny

nekompatibilitou mezi prohlížeči, nicméně při použití složitějších postupů a efektů je důležité míti na paměti, že určité problémy mohou nastat právě díky kompatibilitě, která není absolutní.

4.1.3 JavaScript

JavaScript je skriptovací jazyk, který se ve většině případů používá při vývoji webových aplikací jako klientský jazyk. Jeho časté nasazení je především proto, že je možné jeho kód jednoduše vložit do souboru s webovou stránkou.

Jelikož se jedná o jazyk, který se provádí až u klienta, je logické, že interpret JavaScriptu musí být součástí webového prohlížeče. Tento fakt často vede k problémů s nekompatibilitou mezi prohlížeči a to proto, že jednotlivé webové browsery mají vlastní implementace tohoto interpretu. Pokud je vývojář nucen využívat funkce, které nejsou mezi prohlížeči vzájemně kompatibilní, nezbývá nic jiného než tvořený kód realizovat dvakrát, pro každý prohlížeč vlastní. Tomuto problému se mi naštěstí podařilo vyhnout mimochodem i díky tomu, že nepracuji s XML, u kterého jsou rozdíly v práci pod Internet Explorerem a Mozillou Firefox.

4.2 Nástroje a prostředky

Výše zmíněné jazyky jsou zapsány v jednom XML souboru, který tvoří zdrojový kód gadgetu. Svůj XML soubor s gadgetem jsem umístil na webhosting a editaci a tvorbu kódu jsem prováděl přímo na webhostingovém serveru. Vyvíjený gadget jsem umístil na iGoogle stránku pomocí dříve zmíněného Developer Gadgetu. Tato kombinace umožňuje okamžité testování napsaného kódu v prohlížeči. K psaní kódu mi postačil editor se zvýrazněním syntaxe, konkrétně Notepad++.

Ladění probíhalo za pomoci již uvedeného Firebugu, který se instaluje jako rozšíření do webového browseru Mozilla Firefox. Tento nástroj se při vývoji ukázal jako velice užitečný a v podstatě těžko si lze bez něj představit tvorbu jakéhokoliv komplexnějšího gadgetu.

Gadget využívá několik druhů různých knihoven. Tyto druhy se dají rozdělit do následujících dvou částí. V prvním případě se jedná o knihovny dodané Googlem, určené a uplatnitelné výhradně při vývoji iGoogle gadgetů. Tyto knihovny se uvádějí v deklarační části zdrojového XML souboru, pomocí tagu `<Require>`. Takže volání některé z těchto knihoven se zapíše např. jako `<Require feature="setprefs" />` [7]. Druhou skupinou jsou externí knihovny, které se připojují jako externí JavaScript pomocí tagu `<script src="">`. Deklarace těchto knihoven se již však neumisťuje do hlavičky, ale do samotného obsahu gadgetu. Ve své aplikaci využívám, krom některých iGoogle knihoven, i dvě externí. Jedná se o specifikaci formátu JSON a o open source knihovnu JSONER, která zpřístupňuje funkce pro práci s tímto formátem [8, 9].

4.3 Uživatelské rozhraní

Při implementaci jsem se snažil dosáhnout toho aby kvůli překreslení nebo obnově stavu jednotlivých komponent nemuselo docházet k obnovení celé iGoogle stránky.

Kvůli rozdělení grafického uživatelského rozhraní do tří záložek je i HTML struktura popisující toto rozhraní rozdělena do tří částí. Každá z těchto částí je tvořena tagem `<div>`, přičemž je u těchto tagů nastavena vlastnost `display:none`. To má svůj význam kvůli použití knihovní funkce *tabs* pro tvorbu záložek. V každé z těchto jednotlivých částí se potom nacházejí elementy tvořící samotné rozhraní jednotlivých záložek.

Důležitým prvkem uživatelského rozhraní jsou prostředky pro zadávání vstupních informací. HTML nabízí v podstatě jen jeden element, tím je vstupní pole formuláře `<input>`. Pomocí těchto elementů jsou zajištěny veškeré potřebné uživatelské vstupy. Předání těchto informací skriptu se provádí pomocí formulářových tlačítek, která mají nastaveny po kliknutí provedení patřičné JavaScriptové funkce. U vstupních polí bylo pochopitelně nutné ošetřit korektní vstup, takže například v případě nevyplnění názvu projektu je uživatel upozorněn.

Dalším nezbytným elementem jsou roletová menu. Tyto menu slouží k výběru projektu, na kterém se právě pracuje nebo k výběru jednotlivého projektu, který se má smazat. Zde kvůli požadavku pro stále aktuální data bez nutnosti obnovování stránky bylo potřebné vytvořit funkci, která vykreslí roletové menu s aktuálními hodnotami. Funkce, která provádí tuto činnost je definována pod názvem `dropDownProjectsList`. Tato funkce je poté volána ve všech nutných případech, např. po přidání nového projektu. Mimo to záložky, jako součást grafického uživatelského rozhraní iGoogle gadgetů, umožňují vytvoření tzv. *callback* funkce, která se provede vždy při přepnutí z jedné záložky na druhou. Takže i v této funkci se provádí překreslení roletového menu

Tzv. *core* knihovna, která je připojována automaticky ke každému gadgetu umožňuje zestručnění některých JavaScriptových funkcí. Jedná se například o často používanou funkci `getElementById`, kterou je možné nahradit stručnějším zápisem `_gel` [7].

Jelikož Google podporuje u gadgetů lokalizaci, bylo provedeno vytvoření české a anglické verze. Jednotlivé lokalizace jsou tvořeny XML soubory. V hlavičce zdrojového souboru se poté uvede URL na tyto lokalizované soubory pomocí tagu `<Locale>`. Soubor s lokalizací obsahuje dvojice identifikátor – lokalizovaná zpráva. Ve zdrojovém kódu je poté možné na libovolná místa vkládat jednotlivé zprávy pomocí metody `prefs.getMsg("id_zpravy"))`. Konkrétní jazyk se určuje podle jazyka, který má uživatel nastaven ve svém iGoogle profilu.

4.4 Práce s uživatelskými daty

Jak jsem již uvedl, pro ukládání uživatelských dat je používán formát JSON, který je uložen jako řetězec ve skrytém uživatelském nastavení. Aby bylo vůbec možné pracovat s datovým typem JSON, je nejprve nutné připojit knihovnu, která je dostupná na http://www.json.org/.

Poté je již možné pracovat s tímto typem, převádět ho z objektu na řetězec nebo naopak vyhodnocovat řetězec a tím ho převádět zpět na objekt. Na úvod práce s uživatelskými daty bude jistě vhodné uvést jak vypadá použitá struktura pro ukládání veškerých dat, tedy informací o názvech projektů, dnů kdy bylo na projektech pracováno a délek časů, který byl jednotlivým projektům nebo úkolům věnován. Minimální kostra datové struktury vypadá následovně:

```
{ps:[]}
```

Jedná se o minimální možnou JSON strukturu, která je používána v mém gadgetu. Identifikátor `ps` je zkráceným zápisem slova projects. Jednotlivé identifikátory jsou záměrně zkráceny na co nejmenší velikost kvůli efektivnímu využití datového prostoru. Tato uvedená struktura se používá jako inicializační a je nastavena uživateli pokud si přidá gadget na svou iGoogle stránku. Datová struktura naplněná daty může vypadat takto:

```
{"ps":[
      {"p":{"n":"Bakalářská práce",
            "t":[{"w":"09.05. 2008","d":7}]}},
      {"p":{"n":"Java",
            "t":[{"w":"09.05. 2008","d":11}]}}
]}
```

Vysvětlení jednotlivých identifikátorů:

- ps – projects, sdružuje pod sebe všechny projekty
- p – project, jednotlivý projekt
- n – name, název projektu
- t – time, sdružuje v sobě údaje o čase kdy bylo na projektu pracováno
- w – when, datum aktivity
- d – duration, délka aktivity ve vteřinách

Aby bylo možné s takovouto strukturou pracovat, tedy modifikovat v ní data, bylo nutné vytvoření několika nezbytných funkcí. Jedná se především o funkce pro přidání a odstranění projektů a pro přidání časové aktivity k projektu. Nejprve, ale bylo potřeba zajistit si nějaké aplikační rozhraní pro práci s objektem JSON. Samotná knihovna specifikující tento formát nemá žádné funkce pracující s daty v JSON, proto bylo nutné získat nějaký rámec funkcí, který umožní práci s těmito daty. Dají se nalézt různé knihovny, které by splnily tento požadavek, ale bohužel pro jazyk JavaScript je jich naprosté minimum. Po pečlivém hledání byla zvolena open source knihovna JSONER. Tato knihovna značně zpříjemňuje práci s formátem JSON zejména díky tomu, že nabízí funkce pro vyhledávání, vkládání, počítání různých elementů apod. [9]. S použitím této knihovny bylo možné sestavit funkce pro vkládání a odstranění projektů, pro přidávání časových záznamů k projektům atd.

Tyto uvedené funkce jsou prováděny na proměnné reprezentující datovou strukturu v části zdrojového kódu v JavaScriptu, to znamená, že tato proměnná není perzistentní a bude smazána např. s obnovením iGoogle stránky. Proto je nutný převod této proměnné do textového řetězce, který je pak následně uložen do skrytého uživatelského nastavení. Převedení proměnné, která je instancí objektu JSON se provádí knihovní metodou `JSON.stringify`. Výsledkem je prostý řetězec, vhodný pro uložení. Tento proces převodu a uložení se děje tehdy když je provedeno přidání nebo smazání projektů nebo když je k projektu uložen čas (po stisknutí tlačítka STOP).

Ovšem aby byl vůbec možný přístup k uživatelským nastavením z JavaScriptového kódu, je nutné tyto nejprve získat. Děje se tak vytvořením instance objektu, který představuje uživatelská nastavení. Vytvoří se tedy objekt, např. `var prefs = new _IG_Prefs()` a poté je k dispozici přístup k uživatelským nastavením pomocí proměnné `prefs`.

4.5 Čas

Implementace práce s časem v sobě zahrnovala vytvoření všech potřebných funkcí pro práci s časem jako s obsahem proměnných, vytvoření stabilního časovače pro měření času a napsání funkce pro generování reportů.

Dříve bylo uvedeno, že veškerý čas je reprezentován jedním číslem. Toto číslo je vyjádření času ve vteřinách. Tento způsob v sobě skrývá tu výhodu, že je možné takto daný čas jednoduše ukládat do proměnných celočíselného typu. Aby bylo sledování času dostatečně uživatelsky příjemné, byla vytvořena funkce, která tento čas převádí do formátu HH:MM:SS. Tato funkce se nazývá `secToTime`. Její implementaci provázelo vytvoření dvou pomocných funkcí. Jednou z nich byla funkce pro celočíselné dělení (`div`), protože JavaScript sám o sobě tuto operaci nepodporuje a druhou byla pomocná funkce `zeroCheck`. Tato zajišťuje doplnění nuly pro čísla menší než 10. Díky ní je tedy možné mít čas v podobě 07:01:04, která je přijatelnější než 7:1:4.

Časovač, prostředek, kterým se dá měřit čas, je implementován takovým způsobem aby u něj nedocházelo ke zpožďování nebo ke zrychlování počítací smyčky a také aby byl trvanlivý vůči obnovení stránky (reload). První požadavek, zajištění přesnosti, využívá rozdílu času v době zahájení a ukončení práce na projektu. Po spuštění počítání času uživatelem, se zjistí systémový čas, rozdělí se do tří hodnot na hodiny, minuty a vteřiny, provede se uložení těchto tří hodnot do tří skrytých uživatelských nastavení a spustí se časovač. Uložením tohoto počátečního systémového času do uživatelských nastavení je zároveň splněn druhý požadavek na odolnost proti obnovení iGoogle stránky. Časovač je tvořen smyčkou, která v periodě 1000 ms provádí rozdíl mezi aktuálním systémovým časem a mezi časem uloženým v době spuštění. Tento rozdíl je roven času, který uplynul od doby spuštění počítání. Měření času je tedy založeno na systémovém čase každého uživatele.

Vygenerování reportů je provedeno po stisknutí příslušného tlačítka v záložce Reporty. Stisknutí vede k otevření nového okna, do kterého jsou provedeny patřičné výpisy za časová období. Toto řešení otevírání reportů v novém okně je zvoleno především kvůli prostoru, které nové okno poskytuje. Reporty mohou být samy o sobě celkem rozsáhlé, proto je lepší je nevměstnávat do těla gadgetu, ale raději je zobrazovat v novém okně. Veškeré procedury s tím spojené jsou implementovány ve funkci `generateReport`. Tato funkce ve své činnosti vytváří jeden dlouhý řetězec, který obsahuje veškerý obsah otevřeného okna, kód okna s reporty je tedy tvořen HTML elementy. Z hlediska práce s časem je nutné zjistit zda jsou záznamy u jednotlivých projektů v rozsahu delším než týden. Pokud ano, je po intervalu o délce sedm dní proveden součet délek všech aktivit v daném týdnu a tento součet je vyznačen a uveden v reportu. Toto v sobě vyžadovalo podobný přístup jako v případě měření času, tedy princip rozdílu dvou hodnot. Byla vytvořena funkce `dateToValue`, která převede datum z formátu DD.MM. YYYY na prosté číslo. Díky tomu, je možné provádět rozdíly dvou dat a zjišťovat tak zda jejich rozdíl je právě sedm dní. Celková činnost funkce pro generování je tedy průchod celé uživatelovy datové struktury, procházení jednotlivých projektů u kterých se kontrolují týdenní intervaly a postupné přidávání zjištěných hodnot do výsledného řetězce. Tento řetězec se po analýze všech projektů použije jako obsah nově otevřeného okna.

4.6 Testování

Pro co nejširší míru kompatibility byl vyvíjený gadget v průběhu vývoje testován v prohlížečích Internet Explorer 6, Mozilla Firefox 2.0.0.14 a Opera 9.25. Po dokončení byly testovány ještě prohlížeče Epiphany 2.22 a Safari 3.1.1. U všech těchto vyjmenovaných prohlížečů lze zaručit kompatibilitu výsledné aplikace.

5 Závěr

V této kapitole chci zhodnotit výsledky, kterých jsem dosáhl, shrnout zde zajímavé poznatky, na které jsem při vývoji narazil a nastínit jakým směrem se může ubírat vývoj nejen mého projektu, ale i celé platformy.

Cílem, kterého jsem měl dosáhnout mělo být nastudování problematiky spojené s vývojem aplikací pro iGoogle a vývoj samotné aplikace. Nastudování problematiky spočívalo v obeznámení se s možnostmi, které jsou nabízeny jak uživateli, tak vývojáři. Pouze toto by samo o sobě nestačilo, proto jsou v technické zprávě zahrnuty také poznatky, se kterými jsem se jako programátor potýkal. Právě tyto poznatky považuji za velmi cenné, protože v oficiálních specifikacích jsou uvedeny buď velmi povrchně nebo nejsou uvedeny vůbec. Nezbytnou součástí těchto poznatků je také návrh řešení daného problému a uveřejnění následné implementace.

Přínosem je nepochybně samotný popis vývoje aplikace pro iGoogle a uvedení všech potřebných nástrojů. Zejména výčet nástrojů použitých při vývoji může býti užitečným, protože například uvedený nástroj použitý pro ladění je velice cenným pomocníkem.

Pokud mám zhodnotit vytvořenou aplikaci, tak věřím, že se mi podařilo naplnit stanovené zadání. Co se týká její budoucnosti, tak doufám, že moje celá práce bude zdrojem inspirace pro další vývojáře, ukáže jim cestu jakým způsobem se dají řešit některé neduhy technologie iGoogle a jistě je také důkazem faktu, že se dají i s omezenými možnostmi vytvářet komplexnější aplikace.

Na samotný závěr chci shrnout svoje dojmy z práce s platformou iGoogle. Myslím, že tato technologie dozajista ukazuje směr jakým se bude v nejbližších letech ubírat vývoj celého webu. Jde tedy o zvýšení interaktivity a také o uplatňování nových možností jak reagovat na požadavky zákazníků. Díky této platformě je možný relativně rychlý vývoj nových webových funkcí, které se distribuují ve formě gadgetů. Pokud se odstraní některé nepříjemnosti, např. zmiňované problémy s u-kládáním většího množství dat, tak myslím, že by se technologie iGoogle mohla stát vskutku velice zajímavou, příjemnou a snadno použitelnou platformou pro všechny vývojáře.

Literatura

[1] *Wikipedia: The Free Encyclopedia: Widget engine* [online]. ©2008 [citováno 9. 05. 2008].
Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Widget_engine&oldid=210930453>

[2] *Getting Started - Google Gadgets* [online]. Google Code. Google Inc.
Dostupný z WWW: <http://code.google.com/apis/gadgets/docs/gs.html>

[3] *Wikipedie: Otevřená encyklopedie: Proxy server* [online]. ©2008 [citováno 9. 05. 2008].
Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Proxy_server&oldid=2478676>

[4] *Wikipedie: Otevřená encyklopedie: Cross-site scripting* [online]. ©2008 [citováno 9. 05. 2008].
Dostupný z WWW:
<http://cs.wikipedia.org/w/index.php?title=Cross-site_scripting&oldid=2491438>

[5] *Wikipedia: The Free Encyclopedia: JSON* [online]. ©2008 [citováno 9. 05. 2008].
Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=JSON&oldid=210748112>

[6] *Wikipedie: Otevřená encyklopedie: HyperText Markup Language* [online]. ©2008 [citováno 9. 05. 2008].
Dostupný z WWW:
<http://cs.wikipedia.org/w/index.php?title=HyperText_Markup_Language&oldid=2538808>

[7] *Google Gadgets API Reference - Google Gadgets* [online]. Google Code. Google Inc.
Dostupný z WWW: <http://code.google.com/apis/gadgets/docs/reference.html#JS_Ref>

[8] *JSON - JavaScript Object Notation* [online].
Dostupný z WWW: <http://json.org/>

[9] *JSONER - High quality Web and Java development offshore outsourcing services from Ukraine* [online]. SoftAMIS.
Dostupný z WWW: <http://www.soft-amis.com/jsoner/>

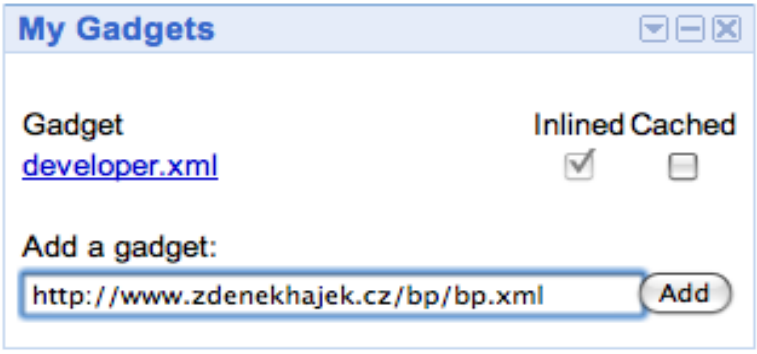
Seznam příloh

Příloha 1 Návod na otestování

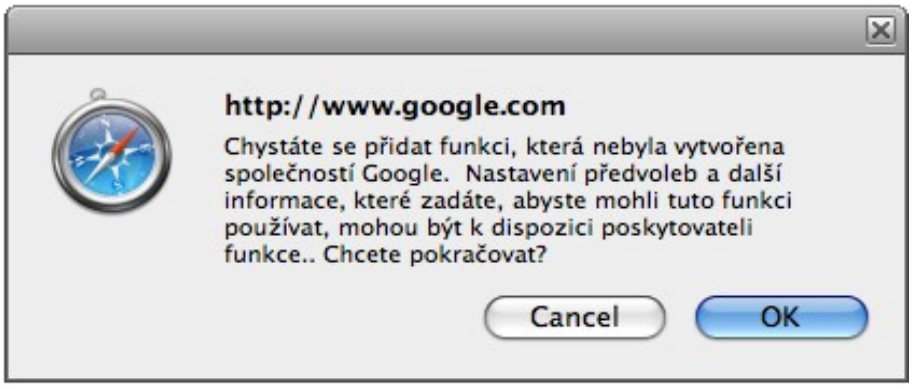
1) Přihlaste se, případně se zaregistrujte na http://www.igoogle.com/

2) http://www.google.com/ig/adde?moduleurl=www.google.com/ig/modules/developer.xml – na této adrese naleznete Developer Gadget, přidejte ho na svou iGoogle stránku kliknutím na „Přidat na Google“

3) Přes Developer Gadget zadejte adresu nového gadgetu http://www.zdenekhajek.cz/bp/bp.xml a klikněte na tlačítko „Add“

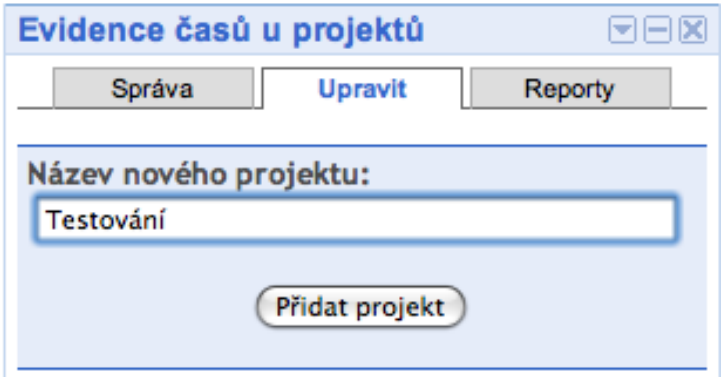


4) Budete dotázáni zda chcete skutečně přidat gadget z cizí stránky. Pro přidání klikněte na „OK“



5) Gadget se Vám vloží na Vaši iGoogle stránku a můžete s ním začít pracovat

6) Přidání nového projektu provedete v záložce Upravit, zadáte název projektu a kliknete na tlačítko „Přidat projekt“



7) Po přepnutí do záložky Správa můžete vybrat projekt a stisknutím tlačítka „Start“ spustit měření času

8) Po dokončení činnosti se počítání zastaví stiskem tlačítka „Stop“. Spočítaný čas se přičte k projektu

9) Vygenerování reportů je možné na záložce Reporty stisknutím tlačítka „Vygenerovat reporty“. Reporty pak mohou vypadat např. následovně