Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

World Wide Web aplikace

Marek Jalovec

Vedoucí práce: Ing. Pavel Šimek, Ph.D.

P r o h l a š u j i,

že jsem tuto bakalářskou práci na téma World Wide Web aplikace vypracoval zcela samostatně a veškerou použitou literaturu a podkladové materiály uvádím v Seznamu použitých zdrojů na konci práce.

V Praze dne ..... Marek Jalovec

Poděkování

Poděkování patří zvláště panu Ing. Pavlu Šimkovi, Ph.D., který svými cennými radami přispěl k úspěšnému dokončení celé práce, dále pak všem, kteří přispěli svou podporou, či radou ohledně vypracování, formy obsahu či principů prezentace.

World Wide Web aplikace

World Wide Web applications

Souhrn

Obsahem práce je popis nejpoužívanějších technologií dnešních moderních webových aplikací jak na straně klientské, tak na straně serverové, dále pak postupy vývoje a práce s klientem, jenž jsou klíčovým prvkem úspěšné práce na dynamických webových aplikacích. Práce obsahuje i několik praktických příkladů, sloužících pro ilustraci syntaxe a podoby jazyků PHP a SQL.

Summary

The content of this work is the description of most commonly used technology today's modern web applications on the client side, as well as on the server side, then procedures of development and work with clients who are a key elements of successful work on dynamic web applications. This work contains also several practical examples which had been served for illustrate the syntax and form of the languages PHP and SQL.

Klíčová slova: web, internet, dynamické aplikace, PHP, Apache, JavaScript, bezpečnost, šifrování

Keywords: web, internet, dynamic applications, PHP, Apache, JavaScript, security, cryptohraphy

Obsah

1. Úvod

Od roku 1996 zažil Internet masivní uživatelský boom, kdy během deseti let přibyla více než miliarda aktivních uživatelů. Tento boom nebyl ovšem doprovázen potřebnou osvětou a naprostá většina uživatelů dodnes neví o principu fungování Internetu vůbec nic a ikona webového prohlížeče je pro ně jen „brána do internetu“. Není tedy divu, že je Internet často zneužíván počítačovými piráty a hackery k získání jak osobních údajů uživatelů, tak finančních prostředků díky nezabezpečeným systémům a ukradeným heslům do bankovních služeb apod.

Snahou dneška je rychle a levně vytvářet dynamické aplikace a systémy přístupné komukoli odkudkoli bez zbytečných technologických omezení, ale tlak na nízkou cenu vývoje bývá prioritou tam, kde by mělo být myšleno hlavně na bezpečnost a kvalitu kódu. Nejdůležitější to je zejména u bankovních a pokladních systémů, komunitních sítí, elektronických aukčních síní a e-mailových služeb, kde je každá chyba draze zaplacena a může znamenat ztrátu peněz, nebo zcizení citlivých osobních informací. Nezbytné je tedy nezapomínat na praxí ověřené postupy a metody vývoje, které těmto rizikům zabrání.

„Na začátku bylo vše tak jednoduché. Původní „Internet“ – zpočátku propojující několik předních výzkumných ústavů v USA – byl určen ke sdílení výsledků vědeckých výzkumů. Ať už jste byli knihovníkem, nukleárním fyzikem nebo počítačovým vědcem, museli jste proniknout do poměrně komplexního systému – Firefox ani Internet Explorer nebyly ještě ani koncepty, když v roce 1962 J. C. Lickider z Massachusetského institutu technologií (MIT) poprvé představil svou myšlenku „Galactic Network“.

Lickider se stal vedoucím počítačového výzkumu úřadu DARPA (Defense Advanced Research Projects Agency), kde vyzdvihoval důležitost svých idejí o sítích. Přibližně ve stejnou dobu Leonard Kleinrock a Lawrence G. Roberts z institutu MIT pracovali na teorii přepínání packetů, klíčovém konceptu pro propojení počítačů do sítě. Roberts pokračoval na vývoji dále, aby roku 1965 společně s Thomasem Merrillem vytvořili první WAN (Wide Area Network), když pomocí vytáčeného spojení propojili síť TX-2 s Massachusetts se sítí Q-32 v Kalifornii.

Roberts přinesl ke konci roku 1966 výsledky svých experimentů do úřadu DARPA, kde vytvořil svůj projekt ARPANET (Advanced Research Projects Administration Network). Toho

času působil Kleinrock na univerzitě University of California – Los Angeles‘ Network Measurement Center, která byla zvolena jako první uzel projektu ARPANET a kde v roce 1969 Bolt Beranek a Newman (BBN) nainstalovali první paketové přepínače označované jako IMP (Interface Message Processors). Výzkumný ústav Stanford Research Center byl vybrán jako druhý uzel a v říjnu roku 1969 bylo uskutečněno první spojení host-to-host. Krátce na to se jako další uzly připojily University of California – Santa Barbara a University of Utah. Jednalo se tak o začátek toho, čemu dnes říkáme Internet.“ [Asleson, a další, 2006]

2. Cíl práce a metodika

Cílem bakalářské práce je zanalyzovat postupy vývoje world wide web aplikací a principy jejich bezpečnosti, která je životně důležitá pro zachování soukromí uživatelů. Postupy budou popisovány na aktuálně vznikající webové aplikaci Prácička.cz, která si klade za cíl stát se konkurencí webům monster.cz, jobs.cz a sprace.cz zejména byznys modelem, navrženým pro podporu mladých a malých firem přístupnými cenami založenými na ratingu inzerátu a počtu zájemců o konkrétní firmu místo vysokého paušálního poplatku.

Práce byla vypracována na základě analýzy dostupných zdrojů dat, dále je použita metoda dedukce, indukce a syntézy, v neposlední řadě pak vlastní poznatky a zkušenosti z oboru získané několikaletou praxí v pozici programátora, projekt-manažera a poskytovatele webhostingu.

Rešeršní kapitola „Používané technologie“ byla vypracována pomocí publikací s tématikou technologií webových aplikací od renomovaných autorů, zejména pana Jiřího Koska a zahraničních autorů jakými jsou například pánové Sverre Huseby, odborník na bezpečnost aplikací, a Baron Schwartz profesionál v oblasti balance-loadingu a administrace databází.

Kapitola „Vývoj a bezpečnost“ je zpracováním faktů o tvorbě nového internetového portálu, kladoucího si za cíl nabídnout nezaměstnaným (ale i pracujícím, hledajícím lepší pozici) snadnější přístup k nabídkám pozic v malých a mladých firmách, dále pak zkušeností získaných studiem zdrojových kódů aplikací a sledováním rozličných emailových fór a diskuzí.

Náplní kapitoly „Marketing a optimalizace“ jsou vlastní zkušenosti z aktivní správy reklamních kampaní na sociálních sítích včetně tvorby reklamních aplikací, dále pak často diskutovaný problém optimalizace kódu a textů stránek pro internetové vyhledávače. Zde jsou některá tvrzení podložena grafy, vytvořenými pomocí statistických nástrojů portálu FaceBook.

Poslední kapitolou je kapitola, věnující se nastavení vlastního serveru s OS Linux. Zde jsou popsány kroky, které byly učiněny při nastavení dedikovaného serveru pro webový portál zmiňovaný v kapitole „Vývoj a bezpečnost“ a které jsou důležité pro zachování bezpečí serveru i jeho uživatelů.

3. Používané technologie

První webové aplikace byly jen jednoduché statické stránky bez podpory dynamického obsahu a jediným pohyblivým prvkem byly animované obrázky formátu GIF. Byly psány v jazyce HTML a díky své jednoduchosti nenabízely mnoho prostoru pro chyby vývojářů a tvorbu bezpečnostních děr. Teprve v roce 1996 přibyl jazyk JavaScript, který umožňuje měnit stránku i po jejím načtení a zobrazení a v případě chyby prohlížeče nebo nepozornosti uživatele způsobit potíže. [Kosek, 1999]

Dnes již existuje skutečně široká paleta možností, jak uživateli přinést dostatečně interaktivní dynamický obsah, který je navíc zabalen do úhledného kabátku moderní grafiky a efektů. S možnostmi vývojářů ale roste i počet technologií, které v případě nedostatečné aktualizace a znalostí představují riziko nejen pro samotný počítač, ale i uživatele. A zde není radno riziko podceňovat.

3.1 PHP

3.1.1 Obecný popis

„Na počátku zrodu systému stál Rasmus Lerdorf. psal se rok 1994 a Rasmus si ve volném čase vytvořil v Perlu jednoduchý systém pro evidování přístupu k jeho stránkám. Jelikož neustálé spouštění interpretu Perlu velmi zatěžovalo WWW-server, přepsal autor systém do jazyka C.

Ačkoliv byl celý systém původně určen pro osobní Rasmusovo použití, zalíbil se i ostatním uživatelům serveru a začali ho používat. Systém se stal oblíbeným a používalo jej stále více uživatelů. Ti přicházeli s požadavky na vylepšení celého systému. Autor proto systém rozšířil a doplnil o dokumentaci a uvolnil jej pod názvem Personal Home Page Tools, který se později změnil na Personal Home Page Construction Kit. V téže době autor zprovoznil elektronickou konferenci, která sloužila jako prostor pro výměnu zkušeností mezi uživateli systému

Kromě zmíněného systému pro evidování přístupů ke stránkám vytvořil pan Lerdorf i nástroj, který umožňoval začleňování SQL-dotazů do stránek, vytváření formulářů

a zobrazování výsledků dotazu. Program, který umožnil zpřístupnění databází na Webu, se jmenoval Form Interpreter (FI).

Celosvětovou proslulost si získal systém PHP/FI 2.0. Tento systém vznikl spojením dvou předchozích programů autora. V této podobě se jednalo o jednoduchý programovací jazyk, který se zapisoval přímo do HTML-stránek. PHP/FI 2.0 se rozšířilo opravdu po celém světě a pracovalo i na mnoha českých severech.“ [Kosek, 1999]

PHP (PHP - Hypertext Preprocessor) je skriptovací jazyk určený pro vývoj webových aplikací. Jeho syntaxe je podobná jazykům Java nebo C++ - využívá objekty pro implementaci modelů, staví na rozdělení kódu na příkazy oddělené středníkem a uzavírání bloků kódu do složených závorek. Jeho možnosti jsou velice široké díky velkému množství zásuvných modulů, které jsou distribuovány buď přímo s jádrem, nebo v PEAR balíku (PHP Extension and Application Repository – Repositář PHP rozšíření a aplikací).

Jeho relativní pomalost (vůči například C++) způsobuje nutnost parsování a kompilace a teprve následné provádění. Tuto nevýhodu lze poměrně jednoduše eliminovat prostřednictvím tzv. akcelerátorů. Lze jmenovat například Zend Accelerator, vyvíjený firmou Zend a nebo open-source řešení eAccelerator. Oba tyto produkty staví na ukládání již kompilovaných skriptů do interní cache paměti, odpadá tak nutnost neustálého načítání, parsování a kompilování kódu. Lze tvrdit, že sebehorší kód, zpracovaný akcelerátorem, bude 1-10x rychlejší. Hovoří pro to jak praxe, tak hodnoty udávané samotnými výrobci. [SourceForge.net]

PHP si prošlo dlouhým a složitým vývojem. Ve verzi 3.0 byl celý parser jazyka kompletně přepsán pány Zeevem Suraskim a Andim Gutmansem, ve čtvrté verzi přibyl Zend engine, bylo značně zapracováno na bezpečnosti jazyka (zavedení superglobálních proměnných $_GET, $_POST, $_SESSION, $_SERVER, atd. a změna výchozích hodnot direktiv v konfiguračním souboru např. u „safe_mode“ na off) a v současné páté verzi byl kompletně přepracován objektový model, byly zavedeny jmenné prostory (shodné se jmennými prostory v C++ - možnost definice stejných názvů funkcí v různých segmentech aplikace). V šesté verzi je opět slibováno velké množství změn, z nichž nejzajímavější je nativní podpora UTF-8 kódování, která byla do teď řešena jen přes mb_string knihovnu a byla často řešena amatérským způsobem v každé aplikaci zvlášť. [Castagnetto, a další, 2001]

3.1.2 Ukázka okomentovaného kódu

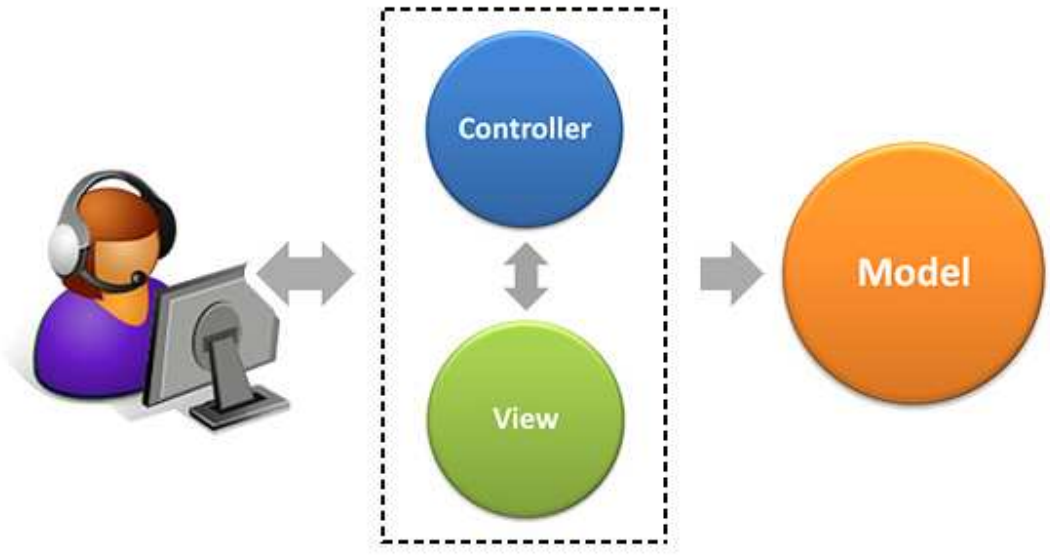
```
/**
 * Funkce pro ověření, zda je uzel $parent přímým rodičem uzlu $node
 * @param array $parent
 * @param array $node
 */

private function isParent($parent, $node) {
    // cyklus skrz všechny listy aktuální větve + volání rekurze
    foreach ($parent['tree'] as $key => $leaf) {
          // pokud je list, nebo nějaký jeho potomek hledaným uzlem
$node, vrátíme true, jinak cyklus proběhne a bude vráceno false
          if ($leaf['id'] == $node['p_id'] || $this->isParent($leaf,
$node)) return true
    }
    return false;
}
```

Strom má podobu uzlů, které mají své vlastnosti ID (identifikátor), název a další, pro tuto ukázku nepodstatné parametry, a na závěr pole „tree“ (strom), které může obsahovat 0 až X stejně konstruovaných uzlů.

Tento kód je využit v editoru stromů, který slouží pro konfigurátor vlastních dveří v nově vznikajícím e-shopu pro doménu interierovedvere.cz, provozovanou firmou CAG, jedním z největších výrobců interiérových a bezpečnostních dveří v České Republice.

3.1.3 „Model – View – Controller“

Architekturu MVC (Model – View – Controller) popsal poprvé Trygve Reenskaug v roce 1979. Jeho podstatou je rozdělení aplikace na tři na sobě nezávislé části, které jsou zcela oddělené od implementace ostatních dvou a zjednodušují tak práci celému vývojovému týmu: kodér může pracovat na šablonách souběžně s programátorem aplikační logiky, neboť programátor nepotřebuje zasahovat do šablony a kodéry naopak netrápí, odkud pochází zobrazovaná data. [Grudl, 2009].



Obrázek č. 1 - Struktura M-V-C architektury

[http://i.iinfo.cz/urs/nette-301-123698498933118.png]

První součástí je „model“. Ten je objektovým zpracováním nějaké problematiky, tedy například třída, mající na starost uživatele. Má určité rozhraní, které zajišťuje získání potřebných dat z databáze, cache, nebo jiného úložiště, manipulaci s nimi vzhledem ke zpracovávané problematice (přihlášení/odhlášení uživatele, změna hesla, atd.) a jejich následné předání controlleru, který si je vyžádal.

Druhou částí trojúhelníku je „view“, tedy pohled. Pohled bývá zpravidla v podobě šablon, tedy textových souborů, které slouží ke generování výstupu stránky. Snahou je oddělit aplikační logiku od zobrazovací, takže šablony obsahují jen značky pro výpis proměnných, jednotlivých objektů a komponent a jednoduché podmínky a cykly (např.: pro výpis tabulky, menu, atd.).

Posledním pilířem je „controller“ (např. v Nette Frameworku se tato část jmenuje „presenter“, jeho funkce však odpovídá schématu MVC), který zprostředkovává plnění šablon stránky z modelu. Zde najdeme volání metod jednotlivých modelů a práci s nimi. V ideálním případě by měl být controller zcela nezávislý na vlastní implementaci modelů; k tomu slouží pseudotřídy zvané „interface“, jež definují „ovládací rozhraní“ modelu a popisují, co všechno třída umí a s jakými parametry je potřeba ji volat.

3.1.4 Frameworky

Framework je softwarový balík, který má za cíl usnadnit programování předpřipravenými knihovnami, třídami, návrhovými vzory a API (Application Programming Interface – rozhraní pro programování aplikací) pro subsystémy (databáze, shell, atd.). Základní myšlenkou je dovolit programátorovi nezabývat se rutinními operacemi jako je tvorba GUI, navigace a celkového layoutu a zaměřit se naopak na důležité věci, jako je aplikační logika dle zadání a tím eliminovat zbytečné zdržení vývoje a bezpečnostní rizika. Snahou je zprostředkovat vrstvu mezi surovým jazykem a aplikací, která zapouzdřuje základní problematiku vývoje aplikací, což má na jednu stranu za následek nepatrné zpomalení provádění z důvodu parsování většího množství kódu, na druhou stranu umožňuje opakovaně používaný kód ukládat do cache a šetřit tak I/O operace disku.

Obecně lze frameworky, díky široké definici pojmu, rozdělit na dvě skupiny, kde jednou je ucelená aplikace, která tvoří nadstavbu nad PHP jádrem a která je ve webové aplikaci jen rozšířena a vhodně doplněna o aplikační logiku a druhou je jen balík knihoven a pluginů pro PHP jádro, který vlastně stojí stranou a používá se přikládáním těchto knihoven ke kódu aplikace.

Mezi nesporné výhody frameworků (tedy alespoň těch kvalitních) je znovupoužitelnost kódu, výrazné zkrácení nutného kódu pro vytvoření funkční aplikace až o několik řádů a bezpečnost. Mnohé frameworky totiž řeší problémy jako validnost vstupů a ochranu před Cross-Site Scriptingem sofistikovanými metodami přímo v jádře a umožňují tak programátorovi nemyslet a hlavně zapomenout na ochranu každého jednotlivého vstupu zvlášť – vše je řešeno automaticky.

Nevýhodou frameworků je na druhou stranu samozřejmě nutnost naučit se programovací jazyk téměř od nuly znovu, neboť na většinu věcí definují své vlastní funkce a metody, bez nichž není možné naplno využít potenciálu, který framework nese. To bohužel často znamená i neúmyslné vytváření bezpečnostních rizik. Navíc každý nový programátor v kolektivu potřebuje čas na zaučení, pokud není ve firmě použit nějaký jemu známý, ale proprietární framework. Proto je výhodné využít open-source frameworků, které mají jednak zpravidla kvalitní dokumentaci, komentovaný kód a hlavně velkou komunitu, která je schopna pomoci vyřešit vzniklý problém či s dopsáním nové funkce.

V Čechách mezi nejpoužívanější frameworky patří bezesporu Nette Framework autora Davida Grudla, Zend Framework, Kohana, Symfony či např. CodeIgniter. Každý z nich má svoje výhody a nevýhody a je jen na vývojářském týmu, který jim bude vyhovovat nejvíce svými možnostmi a strukturou.

Za Zend Frameworkem stojí velká společnost Zend, která ho vyvíjí jako komerční balík spolu s dalšími produkty, jakými jsou Zend Studio (vývojové prostředí pro vývoj PHP aplikací s podporou obarvování syntaxe, maker, automatického doplňování), Zend Server (webový server specializovaný pro Zend Framework), či například Zend Guard, který má nezastupitelnou pozici v ochraně kódu při předávání zákazníkovi. Tato široká škála produktů je na druhé straně vykoupena poměrně nepřehlednou strukturou, složitostí a nesmyslně dlouhými názvy funkcí, které by mohly být jednoduše rozděleny do „namespace“ (jmenných prostorů) nebo do jednotlivých objektů.

Pro českého programátora, kterému není objektové programování cizí, je patrně nejvhodnější Nette Framework, za kterým stojí léta vývoje a obrovská uživatelská česky mluvící základna. Má v sobě zabudovány ochrany proti všem nejběžnějším útokům od Cross-Site Scriptingu, přes SQL Injection až po SESSION hijacking a snaží se o dodržování všech dobrých programátorských zásad, kterými jsou DRY (Don’t Repeat Yourself – znovupoužitelnost kódu), KISS (Keep It Simple & Stupid – udržování jednoduchého a přehledného kódu) a MVC (Model – View – Controller – rozdělení aplikace na ucelené jednotky bez vzájemné závislosti). Přestože obsahuje téměř vše, co si lze pro práci na webové aplikaci představit, je pro něj připraveno velké množství pluginů, které usnadňují implementaci AJAXu (Asynchronní JavaScript a XML), zjednodušují práci s vícestránkovými formuláři (průvodci a dotazníky), tabulkami či nestandardními subsystémy. Již v základu však obsahuje automatický „lazy loading“ (načítání jen toho nejnutnějšího a potřebného) objektů, kvalitní šablonovací systém s inteligentním kontextovým escapováním řídících znaků a automatickým generováním odkazů, nadstavbu ReWrite módu v „Route“ systému či správu a generování HTML elementů a formulářů. Propracovaný cache systém s napojením na memcached eliminuje opakované načítání a provádění skriptů a napomáhá tak ještě vyšší rychlosti.

3.2 MySQL

3.2.1 Obecný popis

MySQL je dvoulicenčním databázovým softwarem, v současné době vlastněným Oracle Corporation. Je distribuován jako komunitní verze bez uživatelské podpory pod GNU/GPL licencí a jako shareware s plnou podporou a velkým počtem podpůrných nástrojů. Databázovými odborníky je většinou krizitován za špatnou podporu relací a uložených procedur, ale posledními verzemi tyto mezery dohání a stává se tak kvalitním a plnohodnotným produktem i pro náročné profesionální nasazení.

3.2.2 Pohledy

Pohledy jsou speciální „tabulky“, jejichž obsah se však načítá dynamicky a není nikde fyzicky uložen. Pokud pohled obsahuje operandy GROUP BY, UNION nebo agregační funkce SUM, MIN, MAX, COUNT, apod., není aktualizovatelný – není tedy možné na něj aplikovat příkazy INSERT, UPDATE a DELETE. S pohledy je bohužel spojena i nemožnost navázat na operace triggery.

Pro práci pohledu je možnost výběru ze tří algoritmů zpracování. První z nich je TEMPTABLE. Ten je velice pomalý, neboť nejprve vytvoří dočasnou („temporary“) tabulku se svým vlastním výsledkem a teprve nad ní provede uživatelský dotaz. Zde je zřejmé, že je vykonána operace navíc, která navíc v případě rozsáhlé tabulky může zabrat i dost paměti a času. Jediným důvodem, proč použít tento algoritmus je uvolnění uzamčení tabulky po vytvoření tabulky dočasné a zkrácení tak nedostupnosti tabulky při delším dotazu.

Druhým algoritmem je MERGE. Zde se MySQL server snaží spojit uživatelský dotaz s dotazem pohledu a vytvořit tak jen jeden SELECT příkaz se spojenými podmínkami v WHERE a HAVING částech. MySQL se při této operaci chová značně inteligentně a je schopno pospojovat i dotazy z různých úrovní podpohledů do jediného dotazu.

Třetí možností je „UNDEFINED“. Zde si MySQL vybere, co se mu pro konkrétní případ hodí, přičemž se stále snaží o výhodnější algoritmus „MERGE“. [Schwartz, a další, 2009] [DuBois, a další, 2010]

3.2.3 Uložené procedury a triggery

Uložený kód procedur je velmi efektivní způsob zpracování dat databáze, neboť šetříme přenosy při komunikaci s vnějším systémem a vše provádíme rovnou v databázi. Jazyk, který se k vytváření procedur využívá, je podmnožinou SQL/PSM a je definován v ISO/IEC 9075-4:2003 (E). Je bohužel velice jednoduchý a omezený a nedosahuje ani zdaleka kvalit procedurálních jazyků. Procedury mohou přebírat parametry a vracet hodnoty, triggery bohužel ovlivňovat nemohou. Důležité je myslet na oprávnění jednotlivých procedur, neboť při útoku SQL Injection, který se dostane až do stored procedury je možno napáchat nevratné škody na datech.

Shrnutí výhod podle Schwartze: [Schwartz, a další, 2009]

- Během v místě databáze eliminuje zbytečný síťový provoz.
- Aplikuje metodiku DRY – Don’t Repeat Yourself – zamezuje opakování kódu.
- Zjednodušuje údržbu.
- Poskytuje jemnější ladění přístupových práv pro ochranu před útoky.
- MySQL si ukládá výkonné plány procedur, je tak vysoce efektivní v jejich volání.
- Díky umístění přímo v databázi se snadno přenáší na další stroje v rámci replikace.
- Umožňuje rozdělit vývoj mezi aplikační programátory a databázové experty.

Shrnutí nevýhod podle Schwartze: [Schwartz, a další, 2009]

- MySQL neposkytuje dobré ladicí nástroje, což způsobuje obtížné debugování.

- Oproti aplikačním jazykům se jedná o pomalý a primitivní jazyk - počet dostupných funkcí je omezený, obtížně se dělají jakékoli složitější manipulace s řetězci.
- Při úspěšném napadení databáze jsou kódy procedur volně dostupné a mohou vést k napadení celé aplikace.
- Přesun zátěže na databázový server, který se hůře škáluje po výkonové stránce.
- MySQL neposkytuje kontrolu nad využitím zdrojů, které může uložený kód alokovat. Je limitován pouze hodnotou z konfiguračního souboru.
- Je těžké provádět profilaci kódu a v záznamech je jen CALL XY(‘heslo‘) – není tedy vidět kód procedury

Ukázka stored procedury, jejíž činnost je při 10 tisících vložených řádcích cca 2,8x rychlejší, než kód přepsaný do PHP: [Schwartz, a další, 2009]

```
DROP PROCEDURE IF EXISTS insert_many_rows;

delimiter //

CREATE PROCEDURE insert_many_rows (IN loops INT)
BEGIN
  DECLARE v1 INT;
  SET v1=loops;
  WHILE v1 > 0 DO
    INSERT INTO test_table values(NULL,0,
      ,qqqqqqqqqqwwwwwwwwweeeeeeeeeerrrrrrrrrttttttttt‘,
      ,qqqqqqqqqqwwwwwwwwweeeeeeeeeerrrrrrrrrttttttttt‘);
    SET v1 = v1 – 1;
  END WHILE;
END;
//

delimiter ;
```

Triggery jsou drobné programy, vykonávané před nebo těsně po provedení operace INSERT, UPDATE nebo DELETE. Současná implementace v MySQL má bohužel omezení jen jednoho triggeru na každou událost, je tedy nutno kód seskupit. To může vést k obtížnému ladění chování aplikace jako celku, dále při chybě zbytek kódu zůstane neproveden. Mezi výhody patří zejména úspora přenosu dat, neboť bez nich by případné navazující aktualizace dat musel vykonat znovu procedurální jazyk, který volal původní dotaz. Mezi nevýhody však patří více věcí a některé z nich jsou i poměrně závažné. Například nejde zrušit volání triggeru, přestože dotaz, který ho spustil, vrátil chybu. Stejně tak nejde zachytit chybu, kterou vyvolal trigger a tedy změnu, kterou měl provést a neprovedl. Mohou tak vznikat nepříjemné chyby v logice aplikace a hrozí nekonzistence dat. Obtížné je pak i dohledávání detailů chování aplikace, pokud nemáme přístup ke kódu triggerů. To ovšem neznamená, že používat triggery není výhodné. Využít je k aktualizaci skladu u pokladního a skladového systému nebo logů je doslova k nezaplacení a ušetří spoustu přenosu dat i výkonu aplikačního serveru. [Schwartz, a další, 2009]

3.2.3 Replikace

Replikace je nástroj SQL serveru určený pro distribuci dat na více strojů za účelem rozložení zátěže, základní zálohy pro případ výpadku primárního serveru, testovací účely či například geografické rozložení práce ve více pobočkách jedné firmy. Replikace však v žádném případě nelze považovat za náhradu zálohování a to je nutné řešit zvlášť. Částečnou funkčnost zálohy může replikace plnit pouze v extrémním případě umělého zpoždění slave serveru za masterem a neustálé kontroly všech dotazů na master serveru odborným personálem. Pak je možno slave server odpojit od mastera dříve, než na něj nějaký destrukční příkaz (např.: DROP DATABASE;) vůbec dojde. Stejně tak distribuce zátěže je sporná, protože zvýšený výkon je pouze u čtecích operací – zápis se provádí na každém slave serveru (otrok) zvlášť (s výjimkou využití tzv. BlackHole enginu – černé díry pro data, který vše rovnou zahazuje) a dochází tak dokonce k zvyšování nákladů na linkách a redundanci dat, stejně snadno může při sebemenší chybě nastavení dojít k pádu replikace, poškození či nekonzistenci dat. [Schwartz, a další, 2009]

MySQL umí provádět replikaci dvěma způsoby. Prvním je příkazová replikace, která byla do verze 5.1 i jedinou možností a je zpětně kompatibilní i se staršími verzemi (starší verze jako master, novější jako slave). Princip její funkce spočívá v zapisování příkazů, které master

instance provádí, do souboru zvaného binární log (zkráceně „binlog“). Zde je uloženo ID serveru, který příkaz poprvé vykonal kvůli ignoraci svých vlastních příkazů u replikací do kruhu nebo master – master, samotný příkaz a timestamp (časové razítko) provedení. Každý příkaz má také svoje číslo, které označuje jeho pozici. Ta je klíčová pro napojení replikačních slave serverů na správné místo binlogu. Binlog se pak přenáší na slave server I/O (input/output) vláknem, kde je SQL vláknem zpracován a podle konfigurace zapsán do svého vlastního binlogu (direktiva „log_slave_updates"), nebo zahozen. Pokud konfigurace slave serveru obsahuje direktivu „log_slave_updates“, stává se tak plnohodnotnou replikou a může snadnou posloužit jako master dalším slave serverům.

Největší nevýhodou příkazové replikace je asynchronní provádění a nemožnost zaručení stejných podmínek ve chvíli spuštění příkazu z binlogu jako na master serveru. Mnoho okolností může totiž způsobit zpoždění zpracování binlogu v řádu sekund, ale i několika hodin oproti master serveru. Další nevýhodou je riziko příkazů, které není možné správně provést nikde jinde, než na hlavním serveru (např.: změny v oprávnění, neexistujících tabulkách, atd.) a které mohou přerušit běh replikace. Toto je možné vyřešit doplněním konfiguračního souboru o ignoraci specifických chyb (např.: 1062 – zdvojený záznam v primárním klíči, 1064 – chyba v syntaxi příkazu a mnohé další), nebo filtrováním prováděných příkazů pomocí direktiv „replicate_do_table“, „replicate_ignore_table“, „replicate_do_database“, „replicate_ignore_database“ a jejich „wild“ alternativ, které k filtraci umějí využít regulárních výrazů pro název tabulky a databáze. Nastavení připojení slave serveru na master server se provádí následujícím příkazem:

```
CHANGE MASTER TO MASTER_HOST = ‘192.168.200.1‘, MASTER_USER =
‘replikace‘, MASTER_PASSWORD = ‘heslo‘, MASTER_PORT = 3306,
MASTER_LOG_FILE = ‘mysql-bin.000001‘, MASTER_LOG_POS = 0;
```

Nultá pozice označuje začátek binlogu, přestože se první příkaz na nulté pozici nikdy nenachází; tam jsou příkazy synchronizační. [Schwartz, a další, 2009]

Druhým způsobem je replikace řádková. Tu MySQL umí od verze 5.1 a je často používána v kombinaci s replikací příkazovou – MySQL server si sám určuje, kdy je výhodné použít kterou metodu. Řádková replikace do binlogu zapisuje přímo změněné řádky místo příkazů, které

ke změně vedly. To je výhodné, pokud je například výstupem dlouhého a náročného dotazu bylo jen pár řádků, naopak nevýhodné to je například v případě UPDATE operace přes celou tabulku s miliony řádků. Pak by binlog mohl narůst do velikosti mnoha gigabytů a zaplnit celý pevný disk na slave serveru. Další nevýhodou je, že není možné z binlogu dohledat příkaz, který ke změně vedl, což může někdy být pro obnovu poškozených nebo ztracených dat klíčové. Zbytek procesu replikace probíhá stejně, jako u replikace příkazové. [Schwartz, a další, 2009]

3.2.4 Zálohování

Problematika zálohování databází je poměrně rozsáhlé téma, které se navíc liší engine od enginu (MyISAM/InnoDB). Je ruku v ruce svázaná s problematikou zotavení po havárii, což většina webhostérů, i uživatelů včetně poloprofesionálů opomíjí - záloha je na nic, pokud z ní nelze po havárii server znovu uvést do provozu, je nedostupná nebo nečitelná.

Při plánování záloh je tak nutné myslet dopředu a uvažovat i nad možnostmi zotavení. Každodenní zálohy za běhu jsou k ničemu, pokud obsahují nedokončené příkazy a transakce a mohou být dokonce nečitelné, nemluvě o faktu, že znamenají ztrátu dat od momentu poslední zálohy do současnosti.

Zálohy se dělí na HOT, WARM a COLD. HOT záloha je záloha dělaná za běhu, bez dopadu na běh serveru. Zde je riziko poškození dat, pokud během kopírování dojde k zápisu.

WARM záloha je poněkud šetrnější. Využívá příkazu pro zapsání celého obsahu zásobníku příkazů a cache na disk a následného dumpu databáze, což zaručuje konzistenci dat:

```
FLUSH TABLES WITH READ LOCK;
```

Zde je nevýhodou to, že veškeré nové dotazy čekají na zpracování a dochází tak k prodlevě v práci aplikace. Nicméně je to pořád rychlejší a efektivnější, než záloha souborů při vypnutém MySQL (COLD záloha).

COLD záloha spoléhá na vypnutý MySQL server, tedy na surové zkopírování souborů databáze na nějaké bezpečné místo, což bohužel znamená poměrně dlouhý výpadek, který není často akceptovatelný pro business-critical aplikace. [Schwartz, a další, 2009]

Všechny tyto zálohy ale vyžadují, aby společně s nimi byl zálohován binární log. Jen tak je záruka obnovení všech dat – existuje kompletní obraz dat a záznam všech změn, které v nich byly provedeny. To je potřeba například i pro případ auditu, nebo vrácení změn, které byly provedeny omylem. Typickým příkladem je, že si zákazník smaže velké množství dat, která ale nezbytně potřebuje a je nutné je obnovit (emaily uložené v databázi, produkty v e-shopu, atd.). K tomu se využije záloha a binární log, ze kterého se vyjme příkaz mazání a který se následně aplikuje na zálohu. Tím máme všechna data do aktuálního okamžiku, jako by ke smazání nedošlo.

Poměrně ideální je využít k zálohovacím účelům slave server replikace – nedochází k zatížení a zastavení master serveru ani při COLD záloze a není problém aplikovat ani hromadné uzamčení tabulek. Před vlastním zálohováním pozastavíme replikační slave a po zazálohování dat ho opět spustíme a zpracovávání binlogu pokračuje.

Pro zálohování je často znovu vynalézáno kolo a přitom existuje velké množství kvalitních aplikací a správců. Jedním z nich je například Zmanda Recovery Manager for MySQL, který zvládá jak inkrementální zálohy na binární úrovni, tak inteligentní zpracování binárních logů a má i přehledné rozhraní pro dohledání nějaké konkrétní operace.

3.3 JavaScript / AJAX

3.3.1 Obecný popis

JavaScript se ve světě webu objevil v roce 1996. Byl uveden firmou Netscape, která ho vyvinula pro svůj webový prohlížeč Netscape Navigator 3. JavaScript se zapisoval přímo do kódu stránky a sloužil zejména pro kontrolu formulářů na validní obsah. Výhodou tak bylo zmezení přenášených dat (stránka se nemusela načíst celá znova) a zvýšení uživatelského komfortu. Postupem času prošel JavaScript vývojem, který mu přidal podporu XML (AJAX - Asynchronní JavaScript a XML), další možnosti úpravy obsahu stránky rozšířením podpory DOM (Document Object Model) modelu, atd. [Kosek, 1999]

DOM model přišel v roce 1997 s Microsoft Internet Explorer 4.0, který zavedl DHTML (Dynamické HTML). Bylo možné přistupovat ke každému objektu na stránce, jeho vlastnostem a stylům a měnit je v reálném čase. Teprve tím se staly stránky skutečně dynamickými a interaktivními a umožnily řadu pěkných efektů. [Kosek, 1999]

Během roku 1996 si však firma Netscape uvědomila, že by bylo vhodné mít jednoduchý skriptovací jazyk i pro serverovou stranu. Vznikla tak technologie Mocha, později přejmenovaná na LiveWire a následně na SSJS (Server Side JavaScript) uvedená v balíku Enterprise Server 2.0, který se po akvizici firmou Sun přejmenoval na Sun Java System Web Server. Skripty se zapisovaly dovnitř párového tagu <SERVER>…</SERVER> a jejich výstupem byl HTML kód, který se zařadil do výstupu odesílaného klientovi do prohlížeče. [Kosek, 1999]

V současné době pro SSJS existuje více kvalitních enginů, k nejlepším (a nejpoužívanějším) můžeme zařadit například projekt Rhino vyvíjený v Javě a spravovaný Mozilla Foundation, který vznikl v roce 1997 pod hlavičkou firmy Netscape, nebo SpiderMonkey, který je napsán v C a je taktéž pod správou Mozilla Foundation. SpiderMonkey je obsažen například v APE serveru (AJAX Push Engine), zajišťujícím přenos dat od serveru klientovi bez nutnosti vyžádání dotazem. Rhino pracuje na bázi překladu JavaScriptu do objektového modelu Javy a ten je již efektivně interpretován JVM (Java Virtual Machine) v podobě zkompilovaného strojového kódu.

AJAX, případně AJAJ (Asynchronní JavaScript a JSON; v případě přenosu dat formátu JSON místo XML), je technologie používaná pro změnu části stránky bez načtení kompletního HTML dokumentu. Využívá k tomu přenosu klíčových informací, které jsou na klientské straně zpracovány JavaScriptem a dosazeny na správné místo. Výhodně se toho dá využít u listování v galerii, změny textu na stránce, rozbalování stromových menu, stránkování v tabulce, vytváření záložkového interface, na které jsou uživatelé zvyklí z operačního systému a aplikací na něm a mnohé další.

K přenosu AJAXových dat je využíván objekt XMLHttpRequest, který umí zprostředkovat komunikaci s JavaScriptem na straně klienta s nějakým zdrojem dat na straně serveru a obsahuje i objektový model pro zpracování přijatých dat.

„I když přístup, na kterém Ajax staví, není úplně nový, představuje důležitou změnu v náhledu na klasické paradigma požadavek/odpověď Internetu. Vývojáři webových aplikací nyní mohou svobodně asynchronně komunikovat se serverem, což znamená, že jsou schopni provádět mnoho úkolů, které byly dříve vyhrazeny klasickým klientským aplikacím. Když např. uživatel zadá poštovní směrovací číslo, můžete ho ověřit a případně s jeho pomocí automaticky vyplnit další položky formuláře, jako jsou město a stát. Již dříve jsme byli schopni tyto techniky napodobovat, s Ajaxem je to ale mnohem jednodušší.“ [Asleson, a další, 2006]

Metodou, která AJAX simulovala, je často používaný trik s neviditelným prvkem IFRAME. Tomu je pomocí JavaScriptu měněna adresa na nějaký skript na straně serveru včetně požadovaných parametrů a hodnot a zpátky je vracen opět JavaScriptový kód, který nějakým z výše uvedených způsobů pozmění rodičovské okno. Nevýhodou této techniky je nemožnost volat několik požadavků nezávisle na sobě, jako tomu je u plného AJAXu – IFRAME se musí stihnout načíst, aby se odpovědi nepřepsaly.

AJAX je v dnešní době velice populární metodou, jak oživit interface na straně klienta a vnést do něj trochu interaktivity. Mezi největší výhody patří bezesporu možnost přenášet jen nezbytná data místo celé stránky, což šetří datový tok, dále pak například možnost pracovat na jednom formuláři, zatímco druhý je právě na pozadí automaticky kontrolován a zpracováván –

je tak zajištěn výrazně plynulejší uživatelský zážitek z webové aplikace (blíží se desktopovým, což šetří čas).

3.3.2 Praktická ukázka jQuery frameworku

Ukázka implementace AJAXu pomocí JavaScriptového frameworku jQuery, mající na starost zavolání vzdáleného PHP skriptu a následnou tvorbu tabulky s požadovanými prvky (odkazy jsou generovány pomocí Latte filteru – součásti Nette Frameworku):

```
$('#summary').find('a').click(function(event) {  // vyhledá odkazy
v seznamu UL a jejich prokliku přiřadí akci
    event.preventDefault();  // blokace výchozí akce - prokliku
    $(this).addClass('active');  // označení prvku jako aktivního
    $.get({link loadSummary!}, function(data) {
          fillSummary(data.options);  // callback (zpětné volání) funkce
po zpracování dotazu pomocí XMLHttpRequestu
    })
});

function fillSummary(options) {
    $('#options').empty();  // vyprázdnění kontejneru
    var table = $('<table id="summaryTable"></table>');
    table.appendTo($('#options'));  // vytvoření tabulky
    $('<thead><tr><th
colspan="2">Sumarizace</th></tr></thead>').appendTo(table);
    // iterace polem, které přišlo od PHP skriptu a vytvoření
požadovaných řádků tabulky
    $.each(options, function(index, option) {
          tr = $('<tr></tr>').appendTo(table);
          $('<td class="th">' + option.name + '</td>').appendTo(tr);
          $('<td>' + (option.nameSel ? '<img src="' + (({link showImage,
'part' => 'PARAM_ID'}).replace('PARAM_ID', option.idSel)) + '" alt="' +
option.nameSel + '" title="' + option.nameSel + '" /><br />' +
option.nameSel : '<i>nevybráno</i>') + '</td>').appendTo(tr)
    });
}
```

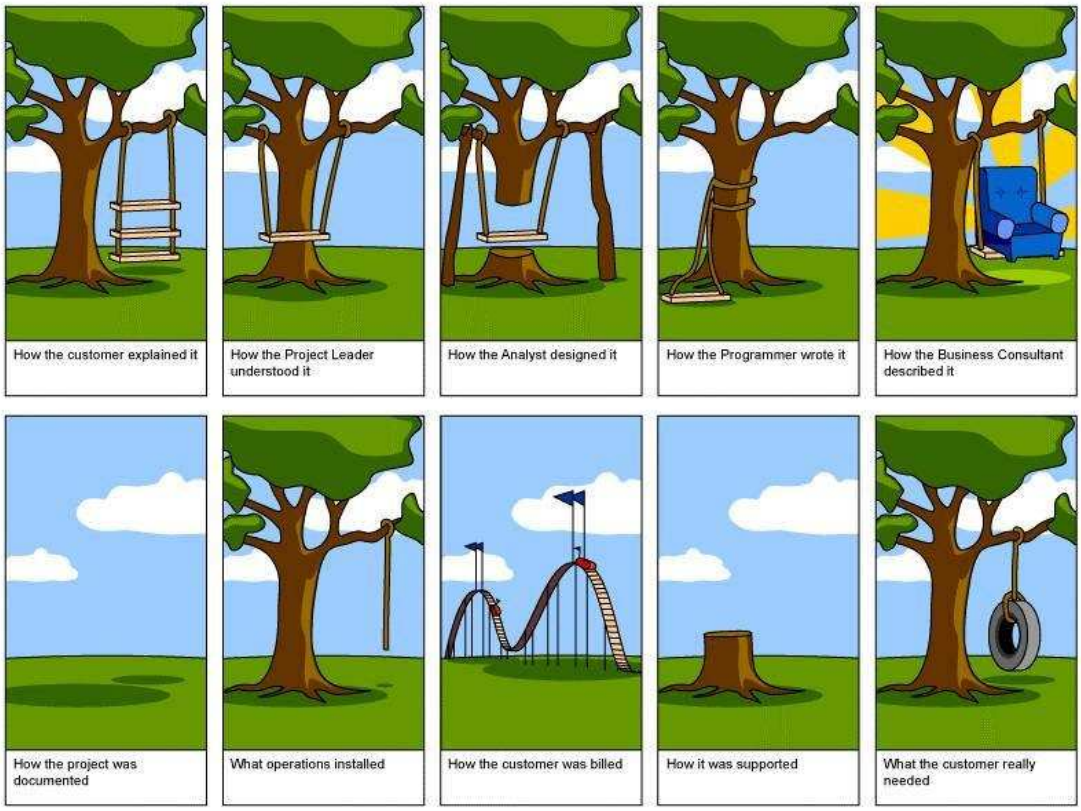
4. Vývoj a bezpečnost

Vývoj webových aplikací je komplikovaný proces, který se skládá z několika zásadních kroků a pravidel. Prvotní krok je důkladná komunikace s klientem, bez níž není možné s projektem vůbec začít. Druhým je brainstorming týmu, který pomůže při vývoji a odhalování slabin projektu. Třetím krokem je výběr vhodných technologií, neboť každý projekt má své specifické požadavky, které každá platforma je splňuje nějak jinak. Čtvrtým je samotná realizace následovaná konfrontací projektu s klientskými požadavky, ve které přichází ke slovu připomínkování a dolaďování projektu. Mnoho firem v této fázi nabízí jako doplňkovou službu zabezpečení projektu a SEO optimalizaci. Pokud klient na takovou firmu narazí, měl by zbystřit, protože tento postup svědčí o amatérizmu a snaze vytáhnout z klienta co nejvíce peněz. Zabezpečení aplikace by mělo být samozřejmostí již při vývoji a musí být automaticky začleněno do použitých technologií a frameworků. SEO optimalizaci je také nutno řešit rovnou – CMS (Content Management System – Systém pro správu obsahu) musí být schopen pracovat s více jazyky, editovat klíčová slova, popisek a titulek stránky.

4.1 Práce s klientem a sumarizace požadavků

Nejzákladnější, ale často opomíjenou a šizenou, částí projektu je sumarizace požadavků, které klient na projekt klade. Pokud v tom klient nemá zcela jasno, k čemu vlastně nový web potřebuje a co má být jeho obsahem, je naprosto nezbytné mu vhodnými otázkami pomoci si tyto otázky ujasnit. Bez základních informací může být výsledkem půlroční práce web, který je zmatečný, nepřehledný, který nemá pro uživatel žádný přínos a který byl zbytečnou investicí a tedy i špatnou referencí. V zásadě platí, že málo který zákazník ví, co vlastně chce a je proto potřeba odborníka, aby mu pomohl s ujasněním zadání.

V realizačním týmu je proto nezbytné mít kvalitního projekt-manažera, který může věnovat klientovi čas a získat tak podrobné informace sloužící pro realizaci. Tuto úlohu musí dělat člověk alespoň částečně znalý práce grafika i programátora a zároveň znalý moderních trendů v tvorbě webu a uživatelského rozhraní, aby nereálné, či zbytečně komplikované nápady klienta usměrnil a pomohl tak vytvoření jednoduchého, srozumitelného a snadno realizovatelného zadání, které povede k úspěšnému webu. Dobrý projekt-manažer splní požadavky klienta a zároveň maximálně usnadní svému týmu práci na projektu.



Obrázek č. 2 - Komunikační šum při vývoji projektu.

Častý problém je komunikační šum, který může výrazně změnit zadání na cestě od zákazníka k realizačnímu týmu. Popis obrázků z leva doprava, od shoda dolů: 1) Co zákazník popsal?, 2) Co projekt-manažer pochopil?, 3) Co analytik navrhl?, 4) Co programátor vytvořil?, 5) Co popsal obchodník?, 6) Jak vypadá dokumentace k projektu?, 7) Jak dopadla instalace?, 8) Za co byl zákazník fakturován?, 9) Jaká je uživatelská podpora projektu?. 10) Co vlastně zákazník potřeboval?
[http://andrewkcho.com/wordpress/wp-content/uploads/2009/12/what-the-customer-wanted.jpg]

U webu Prácička.cz bylo původní zadání klienta velmi kusé – požadavkem bylo předělat stávající web do funkční a konkurenceschopné podoby tak, aby byl graficky přitažlivý pro mladé lidi a byl schopen pojmout velké množství dat a návštěvníků při zachování uživatelské přívětivosti a rychlosti. Web, který na této doméně běží pátým rokem, nebyl v minulosti nikdy dodělán a umožňoval jen registraci uživatele a nenabízel žádný skutečný obsah. Přesto po prozkoumání databáze bylo zjištěno, že se do něj za poslední rok a půl zaregistrovalo více než šest desítek zájemců o práci.

Na základě tohoto zadání byla nejprve uskutečněna analýza konkurenčních webů monster.cz, sprace.cz a jobs.cz po stránce grafické, uživatelské a funkční. Cílem bylo sumarizovat chyby, kterých se autoři webů dopustili a vyvarovat se jich v nově vznikajícím portálu. Dále zjistit, jaké funkce jsou pro všechny shodné a tedy pravděpodobně důležité a potřebné.

4.1.1 Monster.cz

Web monster.cz (viz. Příloha A) má nepřehlednou grafiku založenou na zelené a fialové barvě a snaží se zaujmout logem infantilní „příšery". Grafika je poměrně mdlá, ale ani snaha o moderní styl jí příliš nepomáhá. Vešek je zaplněn úzkým vyhledávacím formulářem s možností rozkliku přes JavaScript, velkým animovaným bannerem a reklamními bloky partnerů, které jsou pro potencionální klienty z valné většiny nezajímavé. Spodní část stránky obsahuje dotazník určený ke zkvalitnění webu a následně odkazy na články, obsahující rady pro správné využívání webu a pohyb na pracovním trhu – ty by měly být umístěny na více viditelném místě a měly by být více zdůrazněné, neboť první dojem při pohovoru a kvalitní životopis jsou základními pilíři při žádání o práci.

Stránka vyhledávání také sama o sobě jinak přehledná není – sloupce v tabulce jsou u většiny nabídek nevyplněné a tak pouze zabírají místo, chybí sloupec s informativní částkou výplaty a případně požadovaný typ pracovního poměru. Vhodné by také bylo doplnit očekávané datum nástupu.

Největší slabinou webu, vedle nepřehlednosti a neucelenosti obsahu, je špatné zpracování přístupnosti, neboť při vypnutém JavaScriptu není možné plně využívat vyhledávacího formuláře a flashové bannery nemají zástupné obrázky – layout je tak poloprázdný a „rozsypaný". Dále celková obsahová chudost nevytváří dobrý dojem a budí dojem neaktualizovaného webu.

4.1.2 Jobs.cz

Jobs.cz (viz. Příloha B) je na tom v porovnání s monster.cz lépe. Stránky působí kompaktnějším dojmem, po obsahové stránce jsou bohatší a přehlednější a jsou postavené na líbivější grafice s uklidňující modro-fialovou barvou a kontrastní oranžovou. Panel sponzorů sice trochu zbytečně zabírá místo, které by bylo možné využít k zpřehlednění layoutu mezerami mezi jednotlivými prvky, ale celkově stránky vypadají dobře a budí důvěru u uživatele. K tomu napomáhá i graf zabudovaný do grafiky, nabuzující dojem burzy, či trhu.

Za výrazné obsahové plus pro studenty lze považovat vyhledávání ve stážích a praxích v odborných pozicích v zaběhlých firmách českého trhu, neboť s hledáním těchto pozic bývá problém – firmy požadují pracovníka s praxí, ale málo komu se chce riskovat přijetí člověka bez pracovní historie, pokud nemá dobré reference. Zajímavá je i část s blogovými příspěvky osobností z řad participujících firem, kde jsou shrnuty cenné myšlenky a informace, které nejsou vždy k dispozici tak snadno, jak by v ideálním případě měly.

Přístupnost webu je lepší, stránky jsou použitelné i s vypnutými obrázky, CSS a JavaScriptem, problém ovšem nastává s čitelností článků a dalších textů psaných tmavým písmem. Zde by bylo potřeba jako pozadí prvků v kaskádových stylech definovat i barvu v případě nenačtení obrázku, pak by problém odpadl. Problémy by také mohl zrakově postiženým působit menší font na spodní části titulky a stejně jako u monster.cz chybí přepínání velikosti textu, které se dnes u přístupných webů stává samozřejmostí.

4.1.3 Sprace.cz

Sprace.cz je projektem firmy Seznam, která v Čechách zastřešuje snad všechny typy internetového byznysu, které si lze představit (od realit, přes veřejné emaily, přes mapy, až právě k pracovnímu portálu). Je tak možné očekávat jistou kvalitu zaběhlé firmy s dlouholetou praxí v oboru a tedy i odpovídající množství nabídek firem i potencionálních zaměstnanců.

Stránky jsou velice svižné, grafika je jednoduchá, bez zbytečných ozdobných grafických prvků a obsahuje jen nejnutnější informace. Je tak poměrně přehledná i pro nového návštěvníka a nepředstavuje překážku kvalitnímu obsahu. Trochu zmatečné je množství oborů, které

na člověka vypadne hned při vstupu na stránku, nebylo by od věci zařadit pozice do několika málo kategorií s proklikem. Vyhledávání nabízí hodně parametrů a umožňuje dostat se velice rychle k požadovanému inzerátu bez zbytečného procházení několika stránek irelevantních výstupů.

Mezi největší přínos oproti konkurenčním webům lze řadit plnou funkčnosti při vypnutém JavaScriptu a vypnutém načítání obrázků, stránka tak funguje perfektně i na mobilních telefonech a PDA zařízeních s pomalým připojením. Veškeré texty jsou krásně kontrastní a není tak problém je přečíst i na tmavém monitoru nebo bez brýlí.

4.1.4 Prace.cz

Prace.cz vypadá ze všech konkurenčních webů nejlépe a patřila i k přání klienta o podobný layout a typ obsahu. Prošla před necelým rokem kompletním redesignem, který přinesl přehlednější grafiku a pestřejší obsah vzhledem ke zpracovávanému tématu. Stránka působí velice čistě, vyhledávání je přehledné a titulka obsahuje vše potřebné v jasné a ucelené podobě. Pruh s partnery nepůsobí rušivě a nepřekáží jako u jobs.cz. Je využito přehledného záložkového layoutu a celkově není webu moc co vytknout, krom občas pomalejších reakčních časů.

Výsledky vyhledávání jsou v přehledné tabulce s odstupy, tvořícími přirozené optické linie mezi jednotlivými informacemi. V tabulce jsou všechny potřebné informace zahrnující datum zadání, název zaměstnavatele, typ pracovního poměru a krátký perex, který výtečně slouží k rychlé orientaci ve výsledcích. Trochu spornou vlastností je možnost personifikované stránky pro zaměstnavatele, která sice boří stereotyp, ale zanáší nepravidelnost, znemožňující rychlé procházení nabídek – je nutno každou nabídku důkladně prohlédnout, neboť jednotlivé části mohou být zpřeházeny.

Celá stránka funguje i bez JavaScriptu a obrázků a nabízí bohatý a aktualizovaný obsah. Mezi výhody patří i newsletter uživatelům se statistikami platu na jejich pozici v lokalitách ČR, který je hodnotnou informací při sebeoceňování. Za plus je bráno i přihlašování do privátní sekce se zadáváním životopisu prostřednictvím zabezpečeného připojení HTTPS.

4.2 Brainstorming

Brainstorming je metoda sloužící k vymýšlení funkcí a vlastností nového projektu, využívající faktu, že více lidí více vymyslí, neboť nápad jednoho může být inspirací druhému a může vést k nápadům, které by skupina jednotlivců nevymyslela ani za násobně delší čas.

Pří brainstormingu se posadí celý vývojový tým do jedné místnosti a nezávisle na sobě říkají vše, co je k danému projektu napadne. Nehledí se na konkrétní přínos jednotlivce, jde jen o vygenerování a zapsání co největšího množství nápadů, z nichž se vyberou ty nejlepší, které poslouží při realizaci projektu jako osnova.

Předpokladem pro úspěch brainstormingu je uvolněná a přátelská atmosféra, neboť strach z vyslovení neobvyklé myšlenky by mohl být kontraproduktivní, základem je tak nevyjadřovat se k nápadům ostatních – nekritizovat a nehodnotit. Jen se nechat inspirovat a přinést co nejzajímavější nápad.

Při tvorbě webu Prácička.cz mezi nejzajímavější nápady patřilo rozdělení webu na dvě domény Prácička.cz a Práčičky.cz, obohacení vyhledávacího formuláře o tlačítko „HLEDEJ podobné“ a přidání operátora na horké lince.

Rozdělení webu mezi dvě domény, které jsou obě v držení klienta, bylo klientem přijato kladně a bylo zařazeno do plánu vývoje. Prácička.cz obsahuje nabídky práce, zatímco Práčičky.cz bude určena pro nabídky brigád a praxí a celý web je koncipován jako přepínatelný pomocí grafického prvku „ohnuté stránky papíru“ v pravém horním rohu stránky.

Obohacení vyhledávacího formuláře o tlačítko „HLEDEJ podobné“, které se stará o vyhledávání, narozdíl od tlačítka „HLEDEJ“, podle volnější definice zadaných parametrů včetně rozmezí platů a bere v potaz drobné rozdíly v rámci jednotlivých krajů (volné pozice, cenová hladina, vzdělání populace, …) bylo taktéž přijato a bylo zapracováno do celkové koncepce webu.

Pozice operátora bude mít na starost schvalování inzerátů, které automatický filtr vyhodnotí jako podezřelé, což zaručí kvalitní obsah bez nevyžádané reklamy a jeho nezávadnost dle platné legislativy (xenofobní či nenávistné texty, propagace zakázaných organizací apod).

4.3 Výběr vhodné platformy a technologií

V počátku vývoje bylo vybíráno mezi LAMP (Linux – Apache – MySQL - PHP) kombinací a konkurenčním balíkem firmy Microsoft sestávající z aplikačního serveru IIS (Internet Information Service) s podporou skriptovacího jazyka ASP a databáze MSSQL. Obě platformy mají své výhody i nevýhody a tak rozhodování proběhlo na základě zkušeností týmu s technologiemi a požadavků klienta.

Vzhledem k požadavku klienta na vlastní dodaný dedikovaný server a k požadavku na nízké vstupní náklady byla zvolena open-source kombinace LAMP, která nabízí dobré možnosti konfigurace se zachováním výborného výkonu, který je možné v případě potřeby jednoduše zvýšit přechodem na serverový software nginx s FastCGI enginem pro podporu skriptovacího jazyka PHP. Nginx je vysoce výkonný server s omezenými možnostmi rozšiřitelnost, ale oproti Apache s několikanásobnými možnostmi zatížení – zvládne i několik tisíc přístupů za vteřinu bez ztráty reakční doby díky propracovanému systému cachování a díky optimálně napsanému kódu serveru.

Výběr platformy Linux víceméně předurčil skriptovací jazyk na PHP, jež je výkonným jazykem s možností objektového programování a s širokou podporou veřejnosti – naprostá většina problémů, se kterými se programátor může potýkat je řešena na některém z mnoha odborných fór a nehrozí tak riziko zabrzdění vývoje díky „bugu“ (chybě) samotného jádra jazyka – je velice nepravděpodobné, aby se tým dostal při práci se stabilní verzí do situace, kdy narazí na neohlášený bug jádra. Plusem je i existence databází open-source a freeware knihoven phpbuilder.com, phpclasses.org a hotscripts.com, které obsahují řešení a implementaci téměř všech běžných problémů od odesílání mailů po práci s obrázky – tím se redukuje množství kódu, které je potřeba napsat při tvorbě aplikace.

4.4 Základní chyby při vývoji

Během vývoje je možné udělat několik zásadních chyb. Můžeme je rozdělit na chyby technologické a na chyby právně-komunikační, vzniklé ze špatné dohody s klientem a na špatně sepsanou smlouvu.

K chybám technickým patří špatný návrh databáze, neodpovídající normálním formám (0NF – každá tabulka; 1NF – buňky jsou dále nedělitelné (adresa rozdělena na ulici, PSČ, město, stát), 2NF – existuje unikátní identifikátor, který pojmenovává celý záznam, 3NF – všechny sloupce záznamu jsou vzájemně nezávislé, 4NF – tabulka popisuje jen jeden fakt, nebo obor hodnot, 5NF – tabulka je natolik jednoduchá, že již není možné ji rozložit na dvě jednodušší bez ztráty vazeb, informací, nebo kontextu), nevhodný výběr technologií, nepřehledné programování, které není zřejmé po krátké odmlce ani samotnému autorovi kódu (zde jsou vhodným pomocníkem komentáře), případně výběr nevhodné hardwarové platformy. Poslední chyba, nevhodný výběr hardwaru, je nejméně problematická, pokud není problémem investice do jiného stroje. Měnit ovšem programovací jazyk, framework, databázový software nebo databázový návrh, či začít přepisovat zdrojový kód od nuly v půlce vývoje, možné není bez vysokých investic v podobě času vůbec možné. Zpravidla by to znamenalo vše, co je již hotové, zahodit a začít zcela znovu. A to není u naprosté většiny projektů přijatelné.

Mezi chyby právně-komunikační lze bez pochyby řadit špatně sepsanou smlouvu a nedostatečnou komunikaci ohledně požadavků a kontroly průběhu vývoje. Ve smlouvě nesmí chybět harmonogram vývoje včetně dead-line (poslední možný čas na dokončení) jak pro vývojáře, tak pro klienta na kontrolu jednotlivých milestone (milní kámen – označení ukončeného vývoje do nějakého konzistentního bodu). Dále pak samozřejmě celková částka za odvedenou práci, podmínky víceprací a úprav po schválení zadání, neboť tyto mohou vývoj protáhnout a tím i prodražit. Součástí smlouvy by mělo být, co by příloha, i kompletní zadání včetně diagramů a grafického návrhu, čímž se pak lze odvolat na jednotlivé prvky funkčnosti, která byla předem dohodnuta.

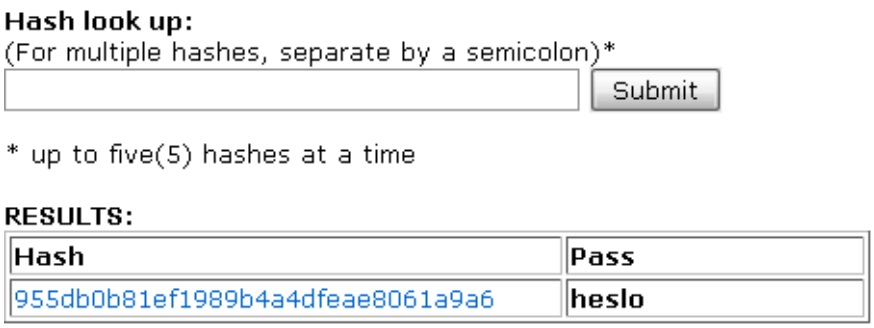
4.5 Šifrování, hashe a hesla

Základní věda o šifrování se nazývá kryptologie, která zastřešuje dva klíčové obory, a sice kryptografii, zabývající se tvorbou šifer, a kryptoanalýzou, která má naopak za úkol šifrovací algoritmy analyzovat a luštit zašifrované zprávy. Šifry byly v historii využívány zvláště pro vojenské účely, kdy je první šifrovací metoda známa již od Caesara. Ta spočívala v posunu každého písmene zprávy o pevný počet pozic v abecedě, vznikl tak zdánlivě nesmyslný text. Takovéto šifry jsou ale zranitelné vůči sémantické analýze textu podle výskytu písmen – známe-li jazyk zprávy, je poměrně jednoduché aproximovat konstantu posunu a zprávu dešifrovat. Jediným algoritmem, který spoléhá na posun znaků a je nerozluštitelný, je Vernamova šifra, kde je každý znak posunut o zcela náhodný počet pozic, vzniká tak zcela náhodný soubor znaků, na který nelze úspěšně aplikovat žádnou analytickou metodu. Nevýhodou takovéto šifry je nutnost přenést také dešifrovací klíč, který má stejnou délku jako samotná zpráva a možnost použít ho jen jednou – při použití na dvě odlišné zprávy by už bylo možno vyhledat pravidelnost a šifru prolomit.

Moderní šifrovací algoritmy jako DES, Blowfish a jiné jsou výrazně náročnější na prolomení, ale s dostatečnou výpočetní kapacitou je reálné je prolomit i za relativně krátký čas (natolik aby získané informace byly ještě relevantní). Šifra DES například používá 64 bitový klíč, který je pro ochranu před výkonem dnešních počítačů nedostačující a proto existuje norma TripleDES, která spoléhá na 3 násobnou aplikaci metody DES s různými klíči.

Šifry se dají obecně dělit na jedno a obousměrné. Jednosměrné se nazývají hashe a jejich cílem je poskytnout jednoznačnou identifikaci objektu, na který byly aplikovány, bez ohledu na jeho délku nebo složitost. Základem každého kvalitního hashovacího algoritmu jsou následující požadavky: pro každý vstup musí existovat nějaký výstup, z výstupu nesmí být možno ani přibližně odhadnout co bylo vstupem a změna byť jen jednoho bitu musí vést k naprosto odlišnému výstupu. I přes tyto vlastnosti jsou však algoritmy náchylné na slovníkové útoky a na útoky hrubou silou („brute-force“ útoky). Ty spočívají ve volání hashovací funkce s nějakými generovanými hodnotami a následným porovnáváním těchto hashů s hashem hledaným. Proto je v aplikacích nutno nespoléhat jen na samotnou hashovací funkci a doplnit ji vhodným algoritmem, který slovníkovou metodu a metodu hrubé síly znesnadní (v případě

neprozrazení algoritmu). Nejjednodušší je metoda zvaná „salting“ (solení), tedy přimíchání nějakého dalšího řetězce do hesla a až následné hashování, které i v případě úspěchu prolomení hashe znemožní úspěšné přihlášení. I při těchto ochranách ze strany programátorů by mělo v ideálním případě platit, že heslo je alespoň šestimístné, mělo by obsahovat malá a velká písmena, čísla a nějaký speciální znak (zavináč, křížek, vykřičník, tečku, …) a že by člověk neměl používat stejné heslo pro dvě různá místa. Často se pak stane, že u nějakého slabě chráněného webu či fóra dojde k průniku útočníků a krádeži přihlašovacích údajů. Pak mají útočníci k dispozici přístup k mnoha dalším webům prostřednictvím ukradených hesel. Toto doporučení naráží na problém, že lidský mozek není schopen zapamatovat si větší množství zcela náhodných řetězců a mnoho lidí by pak aplikovalo „papírkovou metodu“ – hesla na papírcích, umístěných poblíž počítače a to opět vytváří riziko krádeže hesel.



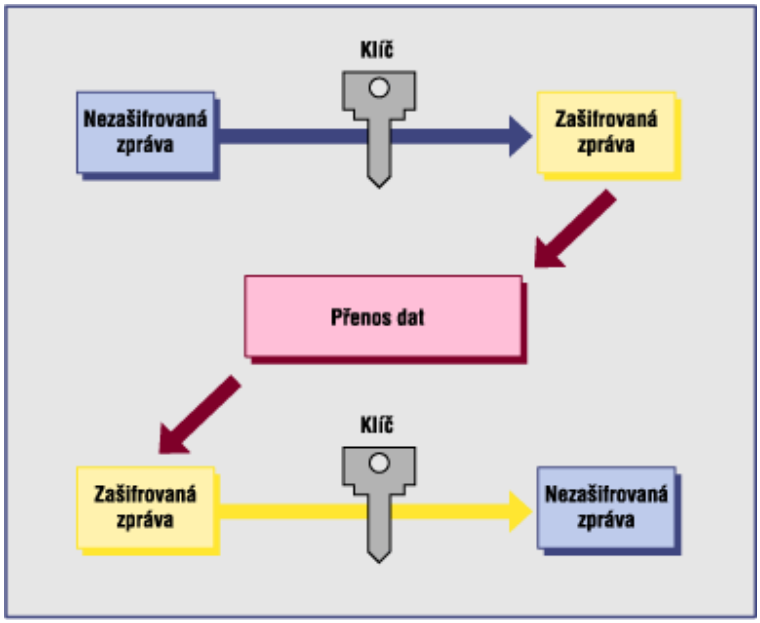
Obrázek č. 3 - Ukázka prolomeného hashe ve veřejně dostupné knihovně MD5 hashů.

[http://gdataonline.com/seekhash.php]

Obousměrné šifry slouží k utajenému přenosu informací, které mají být čitelné až pro autorizovaného příjemce. Zde existují tři možnosti, jak tuto komunikaci provést: symetrické šifrování, asymetrické šifrování a hybridní šifrování.

4.5.1 Symetrické šifrování

Symetrické šifrování je vhodné, pokud je reálné předem domluvit šifru a přenést ji po nějakém bezpečném kanále. Vhodným kanálem je například osobní předání na nějakém médiu, nebo již vytvořený bezpečný tunel internetem (čehož je využito u hybridního šifrování). Výhodou těchto algoritmů je vysoká rychlost šifrování i dešifrování při zachování přijatelné obtížnosti prolomení, navíc s délkou klíče náročnost prolomení exponenciálně roste – u 256bitového klíče se již dá mluvit o vysoké bezpečnosti. Princip fungování zobrazuje následující obrázek – zpráva je zašifrována symetrickým klíčem, přenesena a příjemcem opět rozkódována pomocí stejného klíče. [Huseby, 2006]



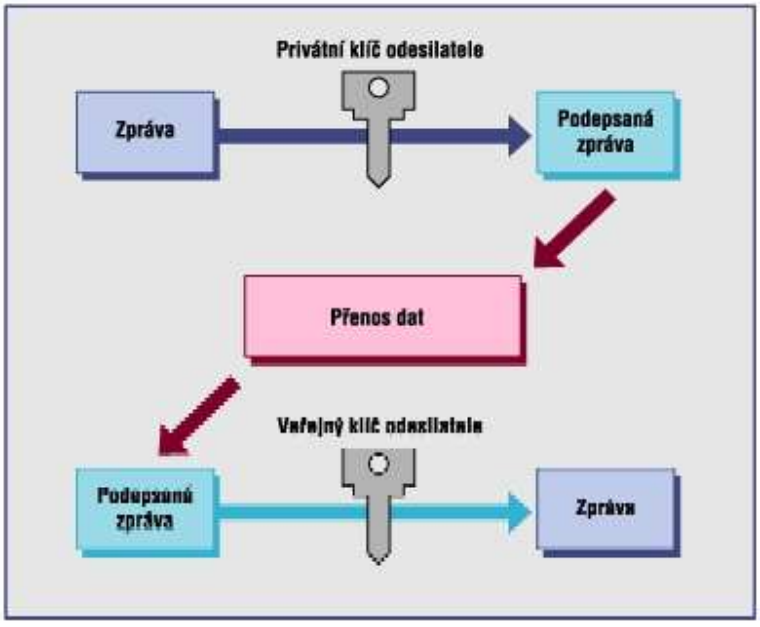
Obrázek č. 4 - Symetrické šifrování dat

[http://st.vse.cz/~XRENP01/sym_soubory/image002.gif]

4.5.2 Asymetrické šifrování

U asymetrického šifrování vždy existují na straně odesílatele i příjemce dva certifikáty – privátní a veřejný. Privátní klíč je nezbytné udržet v tajnosti, klíč veřejný se naopak distribuuje neomezeně a na pořádání, slouží jako párový k privátnímu a je určen pro dešifrování zpráv zašifrovaných privátním klíčem, či naopak k zašifrování zpráv, jež budou následně dešifrovány klíčem privátním. Klíče jsou navzájem inverzní, nicméně při znalosti jednoho není možné odvodit druhý.

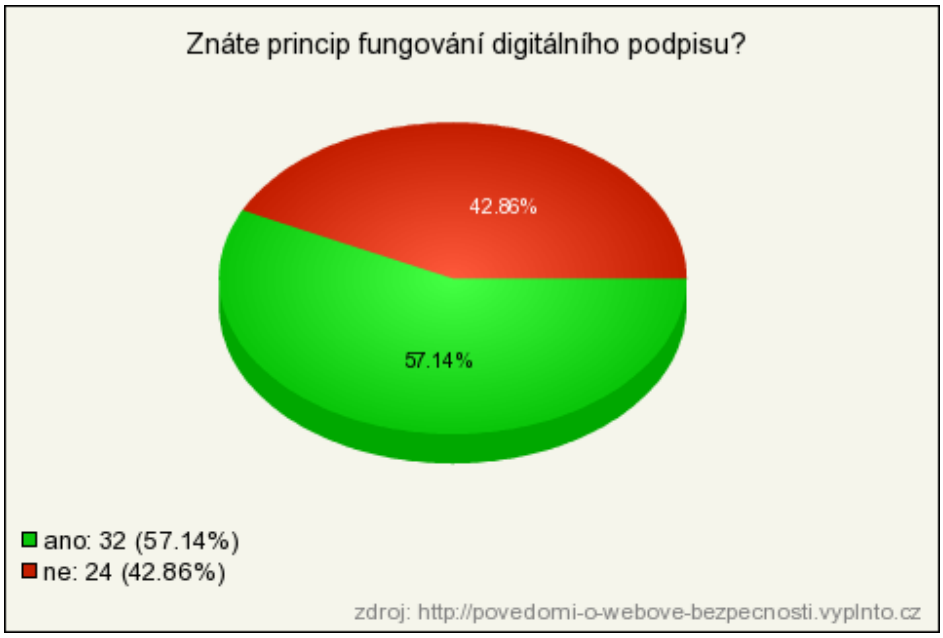
Jsou celkem 3 způsoby využití asymetrických klíčů: první je použít klíče k vytvoření podepsané zprávy odesílatelem. Odesílatel zašifruje zprávu svým privátním klíčem, čímž ověří, že zpráva pochází opravdu od něj a příjemce ji dešifruje veřejným klíčem odesílatele – bez tohoto klíče není čitelná, nicméně dešifrovat ji může každý, kdo si vyžádá veřejný klíč příjemce. [Huseby, 2006]



Obrázek č. 5 - Zpráva podepsaná privátním klíčem odesílatele.

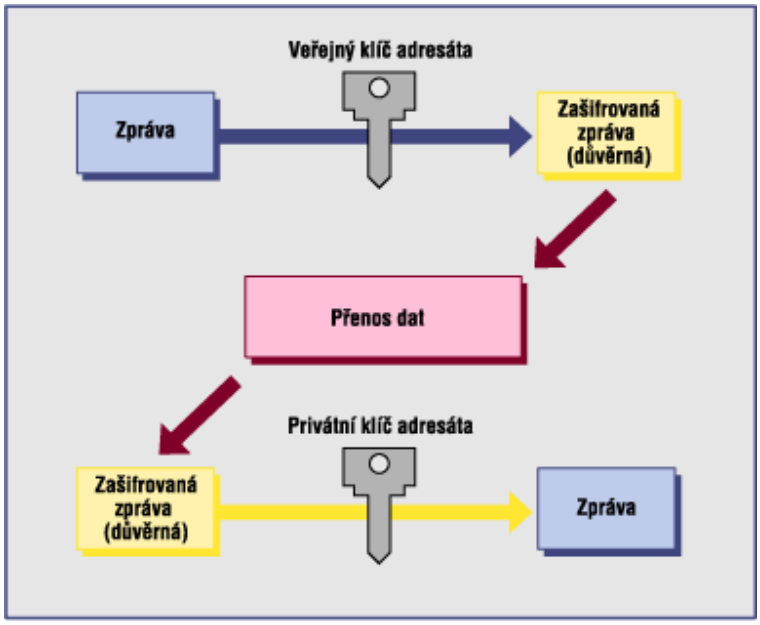
[http://st.vse.cz/~XRENP01/asym_soubory/image002.jpg]

Znalost digitálního podpisu je poměrně dobrá díky propagaci státní správou (v rámci elektronického podání daňového přiznání a elektronických datových schránek) a médii. Je to často probíraný termín v souvislosti s bezpečnosti a soukromím, které jsou dnes klíčovými tématy a je potřeba je brát vážně. Technologie již umožňují přepis mluveného slova do textové podoby a jeho překlad zcela automaticky, jediným způsobem, jak ochránit své soukromí, je tak komunikaci s další osobou zcela skrýt.



Obrázek č. 6 - Znalost principu digitálního podpisu mezi náhodnými respondenty.

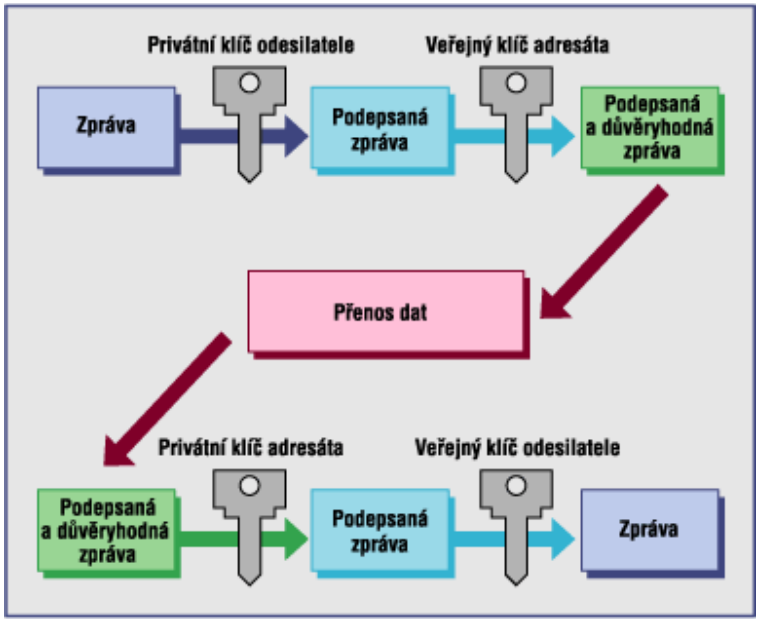
Druhou akcí, kterou lze provést je zašifrování veřejným klíčem příjemce na straně odesílatele a následné dešifrování příjemcovým privátním klíčem. Tím je zaručena důvěryhodnost a původnost zprávy, neboť je přenášena v podobě čitelné až po aplikace privátního klíče a její případná změna během přenosu by způsobila chybu při dešifrování.



Obrázek č. 7 - Důvěryhodná zpráva přenášena v zašifrované podobě.

[http://st.vse.cz/~XRENP01/asym_soubory/image003.gif]

Poslední metodou je kombinace dvou předchozích, čímž vznikne zpráva jak ověřená, tak zároveň důvěryhodná. Metoda spočívá v zašifrování zprávy privátním klíčem odesílatele a následném zašifrování veřejným klíčem příjemce. Zpráva tak není čitelná během přenosu a zároveň je u ní doložen autor. Příjemce po přijetí na zprávu aplikuje nejdříve svůj privátní klíč a následně odesílatelův veřejný a získá původní informace s garancí původu.



Obrázek č. 8 - Zašifrovaná zpráva s ověřeným vlastníkem.

[http://st.vse.cz/~XRENP01/asym_soubory/image004.gif]

4.5.3 Hybridní šifrování

V praxi se často používá kombinace metod symetrického a asymetrického šifrování. Je tak spojena výhoda rychlosti a nenáročnosti na výpočetní výkon s bezpečností přenosu klíče ověřenou metodou. Tohoto je využito například u metody PGP („Pretty Good Privacy“), kterou v roce 1991 uvedl Phil Zimmerman, který byl za uveřejnění této šifry na území USA trestně stíhán. Zpráva je při aplikaci metody PGP nejprve zkomprimována, k čemuž je využit otevřený algoritmus PKZIP. Tím dojde k zmenšení přenášeného objemu dat. Zkomprimovaná informace je poté rozdělena na bloky o velikosti dělitelné 64bity a je zašifrována symetrickým klíčem metodami DES, AES nebo IDEA, čímž je využito rychlosti symetrického šifrování. Před odesláním je k ní připojen symetrický klíč šifrovaný veřejným klíčem příjemce – asymetricky je tak šifrováno jen velmi malé množství dat a celý proces je tak relativně nenáročný na výkon procesoru.

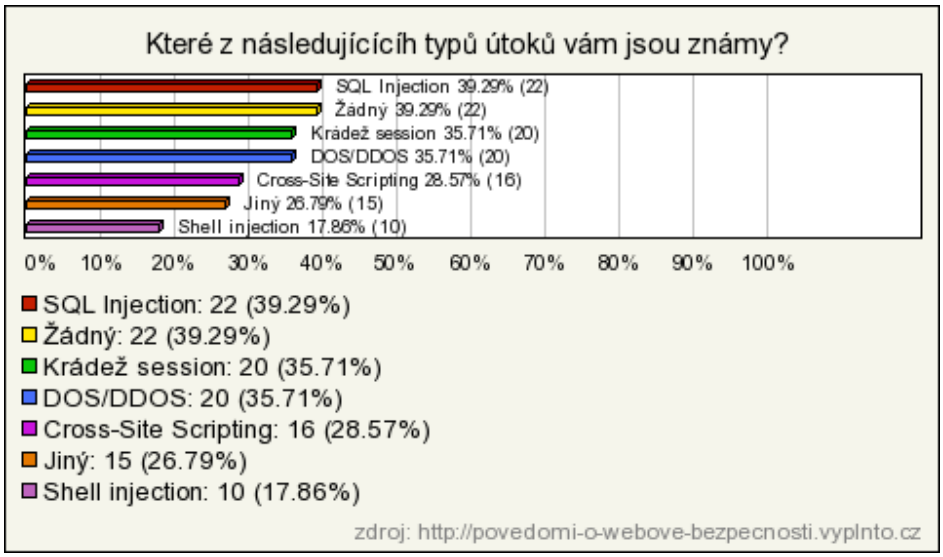
Další implementací hybridního šifrování je protokol SSH u *NIXových OS, kdy je nejprve navázána komunikace asymetrickými klíči, pomocí které je přenesen unikátní klíč symetrický. Asymetrická komunikace je přerušena a komunikace běží přes symetrické klíče rychle, nenáročně a bezpečně.

PGP je součástí i dalších softwarových produktů, například emailového klienta The Bat!, bezpečnostních řešení společnosti SkyNet, a.s., či open-source řešení GnuPG, které je k dispozici i pro operační systém Windows.

4.6 Typy útoků na webovou aplikaci

Typů útoků na webové aplikace je celá řada a je na programátorovi, aby se s nimi vypořádal a ochránil jak server a samotnou aplikaci, tak uživatele, který na web přijde. Důležité je hlídat všechny vstupy do aplikace, neboť žádným datům od klienta není možné věřit a mohou být škodlivá. Všechna data, včetně hlaviček HTTP požadavků je možno pozměnit či cíleně vytvořit se záměrem poškodit aplikaci, nebo k ní získat neoprávněný přístup a zajistit si tak prospěch nebo poškodit provozovatele.

Obecná znalost útoků je poměrně dobrá, nicméně pořád platí, že téměř polovina lidí žádný typ nezná. Z toho lze i odvodit, že neví, jak se jim bránit a jsou proti nim zranitelní.



Obrázek č. 9 - Znalost útoků na webovou aplikaci.

4.6.1 SQL a Shell injection

Útoky typu SQL a Shell injection spočívají v podstrčení dat, která mají za úkol změnit stav systému SQL nebo shellu operačního systému, který je webovou aplikací využíván. Je využíváno přepnutí parseru jazyka z kontextu hodnoty na kontext příkazu pomocí metaznaku (uvozovka, dvojitá uvozovka a zpětné lomítko) a tím vykonání operace, se kterou se při tvorbě aplikace nepočítalo.

Příkladem může být třeba vyřazení ověření hesla při přihlašování:

```
$res = mysql_query("SELECT * FROM `uzivatele` WHERE `jmeno` =
'".$jmeno."' AND `heslo` = '".$heslo."'");
```

Při zadání hodnoty „' OR 1 = 1; --" do pole hesla, dojde k vyřazení podmínky využitím logické funkce OR (nebo), vždy platného výrazu 1 = 1 a zakomentování zbytku příkazů sekvencí zdvojené pomlčky.

PHP má zabudovanou ochranu (zapnutelnou konfigurační direktivou „magic_quotes_gpc"), která se snaží útoku tohoto typu předejít a přidává tzv. „escape" znak zpětného lomítka před běžné metaznaky, které přijdou od klienta v parametru GET, nebo POST požadavku. Escape sekvence změní metaznak na jeho původní význam, tedy na normální ASCII znak, který již přepnutí významu v příkazu nezpůsobí. Přesto tato ochrana není dostatečná, neboť útok lze provést i bez těchto znaků, případně je lze zadat formou ASCII kódu v hexadecimálním tvaru, který ochranou projde.

U webu Prácička.cz je k ochraně využito open-source MySQL knihovny Davida Grudla jménem „dibi". Tato knihovna je implementací vrstvy, starající se o zprostředkování správného příkazu do běžně používaných databázových strojů (MySQL, PostgreSQL, MSSQL, SQLITE a dalších) a chránící databázi proti útoku typ SQL Injection hlídáním typů předávaných dat. Využívá syntaxi podobnou PHP funkci „sprintf()", kdy pro náhradu proměnných využívá znakových konstant, označujících typ vkládané hodnoty. Využít lze například následující konstanty:

Konstanta	Popis
%s	Řetězec
%sN	řetězec, ale prázdný se přeloží jako NULL
%bin	binární data
%b	logická hodnota
%i %u	celé číslo
%iN	celé číslo, ale 0 se přeloží jako NULL
%f	číslo s desetinným rozvojem
%d	datum (očekává řetězec nebo celé číslo)

Tabulka č. 1 - Příklad konstant dibi knihovny pro náhradu obsahu.

Výhodou tohoto přístupu je snadnost a čitelnost zápisu a nemožnost na ochranu nějaké hodnoty zapomenout. Příklad kódu, využívajícího dibi, sloužící k načtení a vypsání obrázku z databáze do vstupního proudu:

```
if ($res = dibi::fetch("SELECT * FROM [part_images]
WHERE [pi_part] = %i", $part)) {
        header('Content-Type: image/jpeg');
        print $res['pi_image'];
}
```

Zde je proměnná „$part“ celočíselného typu přebírána z AJAXového požadavku a dibi se stará o její interpretaci validním způsobem i v případě, že se uživatel pokusí podstrčit hodnotu řetězcovou. Je pak jen na programátorovi, zda chybu vrácenou příkazem zobrazí uživateli, nebo si ji třeba jen zapíše do záznamu pro pozdější šetření chybného chování. Vždy by mělo být snahou chyby zachytávat a později analyzovat. Je možné tak předejít rozšiřování chyby do dalších dat vznikajících postupným užíváním aplikace.

Zranitelné jsou hlavně databáze PostgreSQL a MSSQL, které umožňují zadání více než jednoho příkazu najednou – u MySQL vrátí PHP chybu a zabrání provedení příkazu. Lze pak využít ukončení textové konstanty, třeba výše zmíněného hesla, a druhého příkazu pro vyprázdnění tabulky uživatelů - „‘; DELETE FROM `uzivatele`; --“.

```
SELECT * FROM `uzivatele` WHERE `jmeno` = ‘pepa.novak’ AND `heslo` = ‘‘;
DELETE FROM `uzivatele`; --‘;
```

Nebo například pro vypnutí celého MSSQL serveru:

```
SELECT * FROM `uzivatele` WHERE `jmeno` = ‘pepa.novak’ AND `heslo` = ‘‘;
SHUTDOWN; --‘;
```

4.6.3 Session hijacking

Naprostá většina serverů využívá k udržení sezení klienta tzv. „session“. Narozdíl od cookies jsou tyto soubory uloženy na straně serveru a ukládání dat do nich je tak výrazně bezpečnější – klient je nemůže svévolně měnit. K identifikaci dat je využitý identifikátor SESSID, který se odesílá klientovi prostřednictvím cookie, nebo parametru GET požadavku (u PHP přibude do adresy řetězec „&PHPSESSID=“ a nějaký alfanumerický identifikátor) při založení sezení. Data jsou tak bezpečně v kontrolovatelném úložišti a klient jen při každém požadavku připojí cookie s identifikátorem, podle nějž server potřebná data načte a zprostředkuje je aplikaci.

Problémem je, pokud aplikace vystaví nové sezení klientovi již při prvním načtení stránky bez přihlášení a ponechá mu ho i po přihlášení – vygenerování nového identifikátoru si musí programátor explicitně vyžádat na API jazyka a to bývá opomíjeno. Pokud je v této chvíli identifikátor sezení ukraden útočníkem, může si ve své cookie identifikátor změnit na identifikátor oběti a získat tak jeho přihlášení bez znalosti hesla. Způsobu zcizení je mnoho – Cross-Site Scripting, „Man in the middle“ (prostředník) nebo třeba podstčení identifikátoru prostřednictvím GET požadavku.

U podstrčení probíhá útok tak, že útočník přiměje oběť k návštěvě stránky odkazem obsahujícím jeho PHPSESSID v parametru požadavku. Oběti se tak uloží do cookie SESSID útočníka a jakákoli akce provedená na stránce bude prováděna jako by pod útočníkovým sezením. Pokud se tedy oběť přihlásí a nedojde k obnovení identifikátoru sezení, přihlášení oběti bude zprostředkováno i útočníkovi a ten bude mít k dispozici celý účet oběti.

5. Marketing a optimalizace

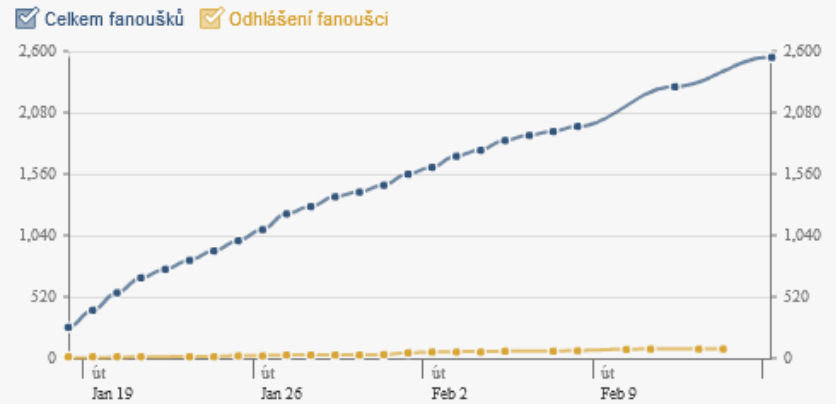
5.1 Komunitní sítě jako marketingový nástroj

V dnešní době se rozmáhá využívání komunitních sítí, jakými jsou FaceBook, nebo Twitter, k marketingovým účelům. Častým případem je bohužel zneužívání možností, jaké tyto sítě poskytují, například nalákání uživatel do falešné skupiny, její následné přejmenování a prodej. Nalákat naivní uživatele je poměrně jednoduché, stačí jen dobrý nápad. V minulosti byly velice úspěšné skupiny slibující neexistující výhru po dosažení určitého počtu fanoušků nebo nějaké výhody oproti ostatním uživatelům – statistiky běžně uživatelům nedostupné (kauza s „aplikací“ „Zjisti, kdo si prohlíží tvůj profil”, která slibovala detailní statistiku přístupů na uživatelův profil), nebo změnu vzhledu. Vzhledem k snadnému přejmenování skupiny pak někteří uživatelé skupiny slibující telefon iPhone zjistili, že jsou nyní členy skupiny „Mladí lidé volí Jiřího Paroubka!”, se kterou by asi ne každý z nich souhlasil.

Mimo praktiky porušující pravidla slušného chování a licenci FaceBooku, jsou „fanpage“ (skupiny fanoušků) velmi populární u českých médií a společností. Televizní stanice od ledna roku 2010 masivně využívají fanpage na FaceBooku jako portál pro diskuzi diváků s politiky v diskusních pořadech, což je výhodou pro běžného občana – má možnost reálně politikovi položit otázku, která ho zajímá a na kterou by možná těžko dostal odpověď.

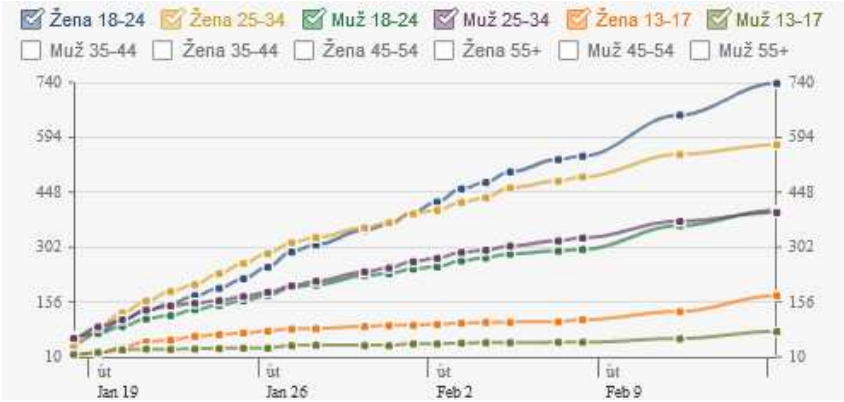
Firma EKO-KOM provozuje například skupinu „Má to smysl, třídím odpad“, která má za účel osvětu v problematice třídění odpadu, výhledové také shromáždění dostatečného množství lidí pro reálnou akci – například dobročinnou kulturní akci. Společně s touto skupinou provozuje i další aplikace, jakou byl například kvíz „Jaká jsi popelnice?“, která měla nepopiratelný úspěch – v době od 21. 1. 2010 do 21. 2. 2010 si ji vyzkoušelo 648 osob.

Graf nárůstu počtu fanoušků skupiny „Má to smysl, třídím odpad“ je, jak vidno z grafu, poměrně rovnoměrný a stabilní, množství fanoušků, kteří se ze skupiny odebrali je víceméně zanedbatelné.



Obrázek č. 10 - Nárůst počtu fanoušků fanpage firmy EKO-KOM k 16.2.2010

Výhodou takovéto skupiny je snadná distribuce informací jak vzdělávací povahy (v případě této skupiny), tak komerčního charakteru a protože Facebook zprostředkovává i demografické statistiky fanoušků, je ideálním materiálem pro cílení vhodné reklamy.



Obrázek č. 11 - Graf věku a pohlaví fanoušků fanpage firmy EKO-KOM.

Ze zjištěných údajů vyplývá, že skupiny na sociálních sítích jsou pro šíření povědomí o firmě a produktu vhodné a je nutno je brát na ně zřetel. Díky nízkým vstupním nákladům je mohou využít i ty nejmenší firmy a živnostníci k propagaci svých produktů a zviditelnění sebe sama.

5.2 Kvalita kódu

Kvalita kódu měla váhu hlavně před několika lety, kdy parsery vyhledávačů neuměly správně rozpoznat kontext jednotlivých textů a v nesmyslných konstrukcích se ztrácely. Pravými zabijáky pak byly stránky v tabulkovém layoutu bez dalšího členění na nadpisy a odstavce. Snad nejhorším, co si lze v tomto ohledu představit, je webová stránka tvořená v Microsoft Word, která přímo oplývá zbytečným kódem. Ani v dnešní době nelze čistotu kódu a jeho validnost zanedbávat. Existují normy, které obecně platí a které při optimalizaci webu pro prohlížeče mají své nezastupitelné místo. Bez jejich dodržování nelze počítat s dobrým umístěním, neboť vyhledávače nezvládnou správně rozpoznat kontext textu.

Nadpisy jsou v (X)HTML párové značky H1-H5, které označují nadpis v jeho postupně snižující-se důležitosti. H1 by měla obsahovat název webu a nějaký slogan, sloužící k marketingovým účelům. H2 je pak určena nadpisům jednotlivých segmentů stránky, případně jako titulek obsahové části. H3 je nadpisem podkapitol a H4 s H5 se již moc nevyužívají. S výhodou lze využít i obrázkového obsahu, neboť parametr „alt“ je pak využit místo něj. Rozmístění nadpisů H1 a H2 na titulní stránce Prácička.cz je vidět na následujícím obrázku:

Obrázek č. 12 - Rozmístění nadpisů H1 a H2 na titulce Prácička.cz

Seznamy jsou prvky určené pro seznamy vlastností, parametrů a hodnot, jsou také často využívány k tvorbě menu, neboť i to lze považovat za výčet hodnot. Výhodou je automatické formátování včetně odrážek a odsazení vnořených bloků, nevýhodou je nepříliš estetické ztvárnění odrážek u nečíslovaných seznamů. Naštěstí není problém tyto odrážky s využitím kaskádových stylů zaměnit za libovolný grafický prvek, či je zcela vynechat.

Tabulky jsou prvky v (X)HTML zcela specifické svým chováním. Díky němu byly v minulosti často využívány k tvorbě layoutů. Klíčovými vlastnostmi jsou: automatické dopočítání šířky a výšky buněk, snadná stylovatelnost a využitelnost vedle sebe ležících buněk včetně zarovnání – toto chování lze například u DIVů s nastaveným obtékáním emulovat jen velmi těžko pomocí rodičovského elementu a ani to není dokonalé.

Specifické místo mají i tagy P, STRONG, či SMALL, které označují text, resp. jeho důležitost. P (podle „paragraph“) je označením obecného odstavce textu bez zvláštního významu, STRONG (silný/ě) označuje důležitý výraz, nebo část, kterou je potřeba zdůraznit, aby ji uživatel nepřehlédl. Takovým blokům je pak přikládána větší důležitost i při indexování obsahu vyhledávačem. SMALL je opakem STRONGu, označuje například poznámky pod čarou, nebo jiné, ne úplně podstatné, informace.

5.3 ReWrite mód - čisté URL

ReWrite mód je plugin Apache serveru, sloužící k přepisu webových adres z formy klasického GET požadavku, obsahujícího řetězec proměnných spojených pomocí znaku „&“ („and“) do formy přehledné a snadno zapamatovatelné. Je možné simulovat virtuální adresářovou strukturu, která je vyhledávači snadněji indexována, neboť dává URI (Uniform Resource Identifier – jednotný identifikátor na zdroj; dříve se užívalo označení URL – Uniform Resource Locator – jednotný ukazatel na zdroj) jednotný styl s pevnými pravidly. Příklad přepisu jednoduchého odkazu:

```
http://www.pracicka.cz/index.php?presenter=clanky&action=detail&id=45
```

ReWrite mód dává vzniknout odkazu výrazně hezčímu a zapamatovatelnějšímu:

```
http://www.pracicka.cz/clanky/detail/45
```

Tento URI je vhodný pro běžné použití, neboť nezatěžuje uživatele názvy proměnných a je i, jak je na první pohled patrno, kratší. V Nette Frameworku je problematika pěkných adres řešena obousměrným routerem (směrovačem), který se stará na straně výstupu o generování čitelných adres pomocí automatického generátoru odkazů v Latte filteru, na straně vstupu se stará naopak o dekódování adresy a předání parametrů tam, kam je třeba, aby aplikace vrátila požadovaný výstup. Výhodou je možnost skládání více rout za sebe a vytvoření systému překladu, který se postará například o dvě různé verze adres (starý a nový web), nebo o vícejazyčnou aplikaci (pomocí automatického překladu). Doporučenou metodikou je skládat routy od nejvíce specifických po obecné, čímž je zaručen správný překlad – systém bere v potaz prioritu routy podle pořadí zápisu. Příklad zápisu jednoduchého routeru, který si poradí s adresou z předchozího příkladu:

```
$router[] = new Route('<presenter>/<action>/<id>', array(
    'presenter' => 'Homepage',
    'action' => 'default',
    'id' => NULL,
));
```

6. Správa systému

6.1 Nastavení serveru - LAMP

Zkratka LAMP je zkrácením kombinace Linux - Apache - MySQL - PHP. Jde o kombinaci svobodného a open-source software šířeného z větší části pod licencí GNU/GPL. Výhodou jsou téměř nulové pořizovací náklady a vysoký výkon daný nízkou režijí systému.

Pří instalaci dedikovaného serveru pro portál Prácička.cz byla provedena čistá instalace z „netinstall“ (minimální verze se základními knihovnami pro instalaci z internetvého zdroje) CD Debianu verze 5.04, který díky své stabilitě a výkonu odpovídá nejlépe požadavkům klienta. Nevýhodou Debianu je občasná neaktuálnost balíků, protože základním požadavkem pro zařazení do distribude je stabilnost verze a její odzkoušení uživateli ve verzi „testing“. Občas se tak stane, že mezi verzemi některé balíky zmizí, neboť jim skončí podpora ze strany autorů. S tím je při aktualizaci nutno počítat a dostupnost potřebných balíků si ověřit

Hned po instalaci byl doplněn balík SSHD, sloužící pro vzdálenou správu protokolem SSH. Doporučit lze změnu konfigurační direktivy „AllowUsers“ a „Port“. „AllowUsers“ je seznam uživalů, kteří mají povoleno přihlášení pomocí SSH na konzoli stroje a „Post“ je portem, na kterém SSH démon naslouchá. Změnou těchto dvou hodnot je výrazně zvýšena bezpečnost serveru, neboť není tak jednoduše napadnutelný brute-force metodou útoku.

Pro FTP spojení byl použit server ProFTPD, který je ověřenou aplikací s vysokým výkonem. Typ instance byl nastaven na „inetd“. FTP server tedy nebude na pozadí běžícím démonem, ale bude při každém požadavku spouštěn centrálním serverem XINETD. I zde je možno nastavit perzistentní běh procesu, důvodem spouštění přes XINETD je, že ProFTPD neumí kontrolovat IP adresu požadavku vůči tabulce „hosts.allow“ a „hosts.deny“, kde jsou white-listy (seznamy povolených) a black-listy (seznamy zakázaných) IP adres a je tak zranitelný vůči útoku typu brute-force. Tyto tabulky jsou používány skriptem Blockhosts, které spolupracují se serverem XINETD a analyzují systémové logy každého připojení. Pokud nějaká IP adresa překročí počet neplatných pokusů o přihlášení, nastavený v konfiguračním souboru, je automaticky přidána do seznamu „hosts.deny“ na dobu, kterou si správce zvolí (vhodná penalizace je například 3 až 6 hodin – většina automatických skriptů při zablokování automaticky

server opouští, ale je potřeba umožnit oprávněnému uživateli, který jen zapoměl heslo, opětovný přístup). Tím je zaručena ochrana pro brute-force útokům. Ukázka výstupu ze skriptu Blockhosts:

```
web:~# blockhosts.py --dry-run –verbose
blockhosts 2.4.0 started: 2010-03-25 09:20:11 CET
 ... loaded /etc/hosts.allow, starting counts: blocked 0, watched 3
 ... loading log file /var/log/auth.log, offset: 2871508
 ... loading log file /var/log/proftpd/proftpd.log, offset: 2195598
 ... discarding all host entries older than  2010-03-25 01:20:11 CET
 ... final counts: blocked 0, watched 3
#---- BlockHosts Additions
#bh: ip:   121.9.210.247 :   3 : 2010-03-25 08:54:19 CET
#bh: ip:  117.240.234.46 :   1 : 2010-03-25 07:33:54 CET
#bh: ip:  125.163.239.56 :   1 : 2010-03-25 06:26:47 CET

#bh: logfile: /var/log/auth.log
#bh: offset: 2877946
#bh: first line:Mar 21 06:27:59 web CRON[4750]: pam_unix(cron:session):
session closed for user root

#bh: logfile: /var/log/proftpd/proftpd.log
#bh: offset: 2195598
#bh: first line:Mar 01 07:33:46 web proftpd[12460] web.core.hostium.cz
(222.252.4.144[222.252.4.144]): FTP session opened.

#---- BlockHosts Additions

sshd, proftpd, in.proftpd: ALL: spawn (/usr/bin/blockhosts.py --verbose -
-echo "%c-%s" >> /var/log/blockhosts.log 2>&1 )& : allow
```

Instalace Apache byla ponechána na výchozích hodnotách, s vyjímkou nastavení cesty k souborům z /var/www na složku, která je uložena na RAID-1 poli. V současných distribucích je Apache velmi dobře nastaven a krom vyjímečných případů není potřeba do konfigurace zasahovat. Co změnu vyžaduje naopak témeř vždy, je konfigurace PHP. Zde bylo potřeba doinstalovat knihovny pro manipulaci s obrázky, s IMAP spojením k mailu a práci s databází

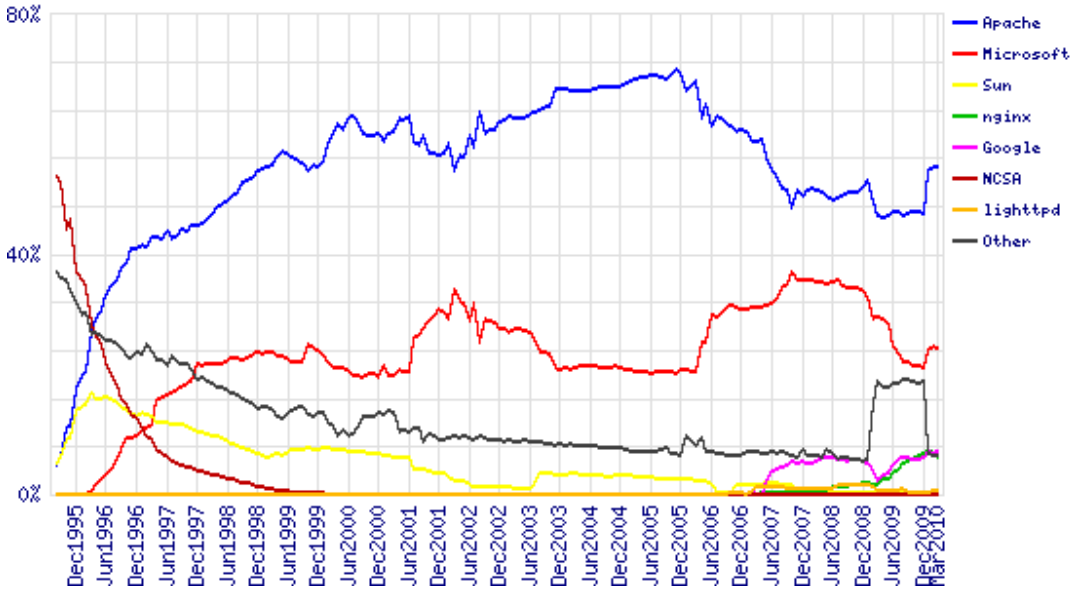
MySQL, které se s jádrem automaticky neinstalují. Dále bylo změněno nastavení maximální povolené velikosti paměti a výpisu chyb, aby se chybové hlášky nedostaly k uživateli.

6.2 Využití Linuxu a serveru Apache

Dle průzkumu organizace Netcraft je server Apache využíván na více než 54,4% hostovaných webech a je tak nejvyužívanějším softwarovým produktem na tomto poli. Následuje ho IIS od Microsoftu s 24,6%. Poměrně zajímavé je nízké využití ostatních produktů, zejména služeb firmy Google (cca 6,9%) a serveru nginx (cca 6,7%), z nichž se žádný, i přes své nesporné kvality, nedostal přes 10% zastoupení.

Server	Únor '10	Procenta	Březen '10	Procenta	Změna
Apache	112,903,926	54.46%	112,747,166	54.55%	0.09
IIS	50,928,226	24.57%	50,572,540	24.47%	-0.10
Google	14,315,464	6.91%	14,592,133	7.06%	0.16
nginx	13,978,719	6.74%	12,673,962	6.13%	-0.61
lighttpd	1,097,685	0.53%	1,657,584	0.80%	0.27

Tabulka č. 2 - Zastoupení serverového software z globálního hlediska.



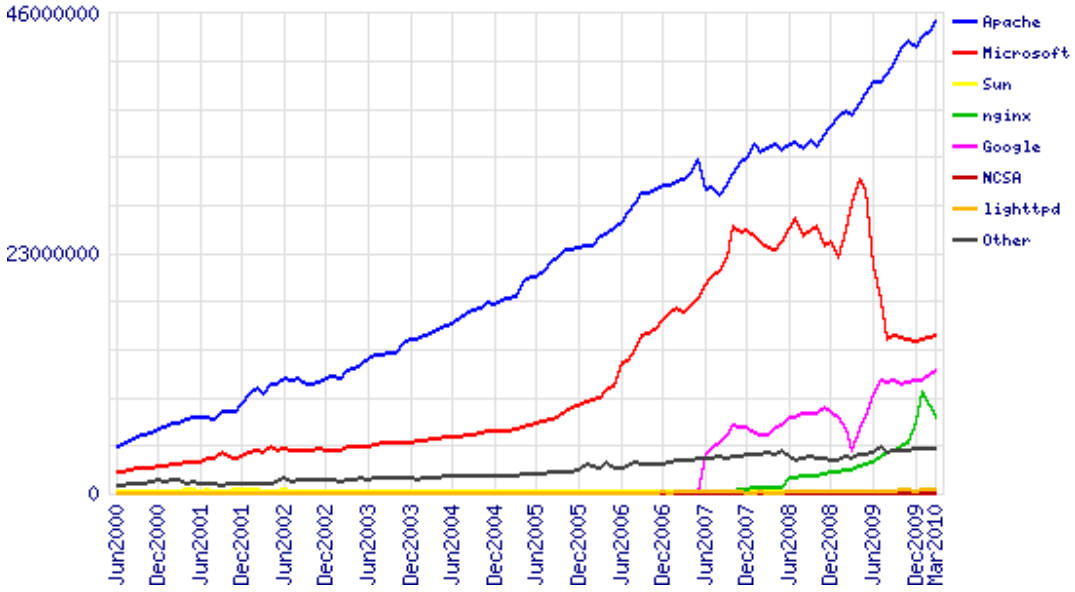
Obrázek č. 13 - Podíl webových serverů na trhu z celosvětového hlediska

[http://news.netcraft.com/archives/2010/03/overallc.png]

Pořadí aktivních instancí webových serverů odpovídá procentuálnímu zastoupení hotovaných domén – vede Apache, následovaný IIS firmy Microsoft a servery Googlu a vysokorychlostním nginxem. Ze statistiky je ale vidět, že každá instance Apache hostuje v průměru výrazně více webů než konkurenční servery. [Netcraft Ltd, 2010]

Server	Únor '10	Procenta	Březen '10	procenta	Změna
Apache	44,293,169	52.83%	45,127,654	53.73%	0.90
IIS	14,959,249	17.84%	15,065,206	17.94%	0.09
Google	11,475,629	13.69%	11,807,466	14.06%	0.37
nginx	8,412,148	10.03%	7,206,107	8.58%	-1.45
lighttpd	290,503	0.35%	292,2	0.35%	0.00

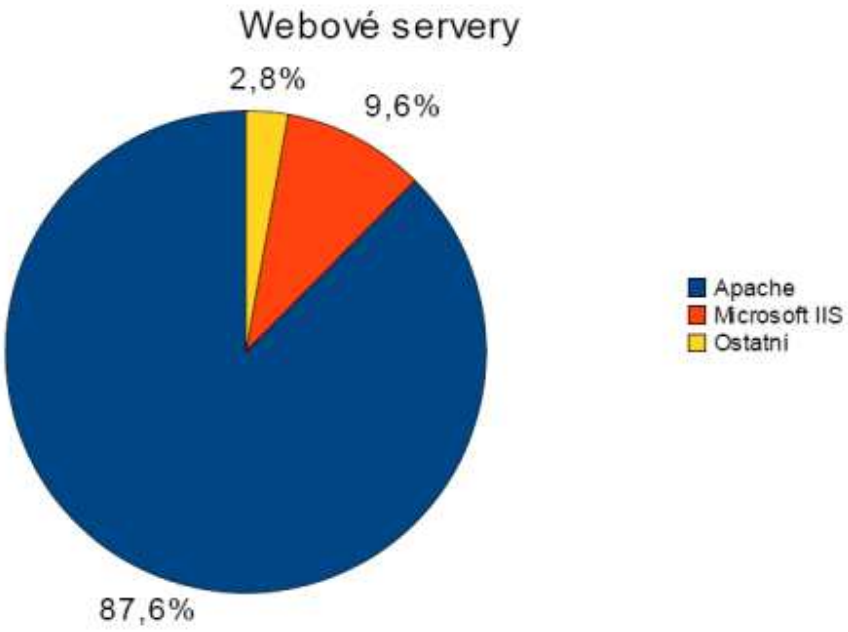
Tabulka č. 3 - Počet instancí serverového software z globálního hlediska.



Obrázek č. 14 - Podíl OS na trhu z celosvětového hlediska

[http://news.netcraft.com/archives/2010/03/overalld.png]

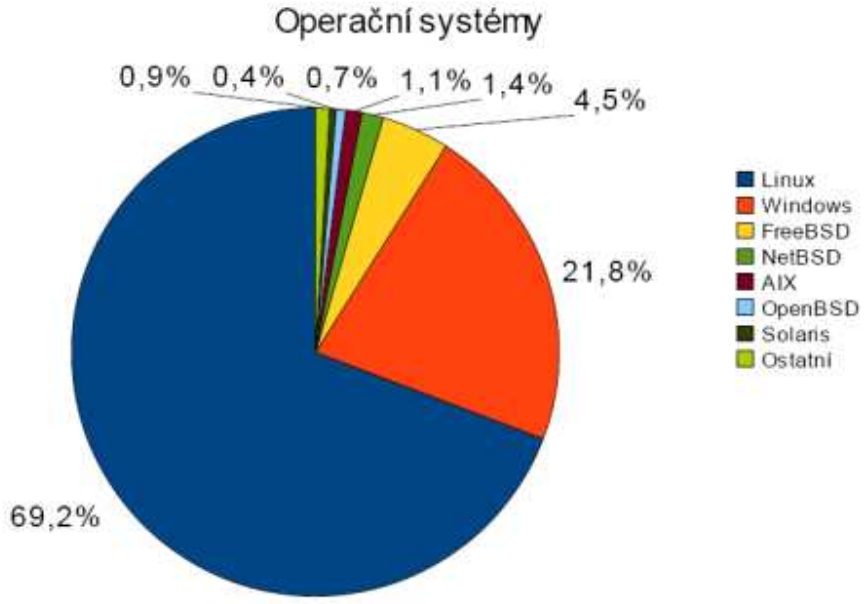
Dle průzkumu serveru root.cz je situace v Čechách značně odlišná a Apache získává drtivou převahu s téměř 88% trhu. IIS má 9,6% a ostatní 2,8% - zde jde zejména o soukromé servery a projekty; drobné webhostingové společnosti si často nemohou dovolit pracovat s nestandardním řešením kvůli malému množství klientů a následné nerentabilitě služeb, v Čechách je navíc relativně malé množství schopných programátorů specializujících se na technologie firmy Microsoft – zde se opět jedná spíše o soukromé síly pracujících na intranetových službách.



Obrázek č. 15 - Podíl webových serverů na českém trhu

[http://i.iinfo.cz/urs/Webservery-121637634032738.png]

Poměrně zajímavý výsledek tohoto průzkumu je, že Linux je jen na necelých 70% serverů (77,6% při započtení systému BSD postavených na UNIXovém jádře), je tedy dost poskytovatelů webhostingu, kteří provozují server Apache pod OS Windows, místo pod domovským Linuxem. Vést k tomu může jednoduchá instalace a grafické rozhraní, které je pro většinu začínajících administrátorů přijatelnější, než textový režim Linuxu. Linux sice také nabízí vzdálenou správu s grafickým výstupem přes systém VNC, ale stále platí, že jde o „nestandardní“ operační systém, který vyžaduje diametrálně odlišný přístup a znalosti, než správa „bezúdržbových“ Windows, kde je většina nastavení na defaultních hodnotách dostatečná pro zprovoznění webové aplikace. [Krčmář, 2008]



Obrázek č. 16 - Podíl OS na českém trhu

[http://i.iinfo.cz/urs/Systemy-121637479692055.png]

6.3 Aktualizace

U serverových aplikací platí, že sebemenší mezera bývá využita útočníky k nainstalování nežádoucího software či k poškození dat. Je tak tedy nezbytné mít veškerý software na serveru aktualizovaný a prozáplatovaný. U Linuxových distribucí je vhodným řešením automatická instalace kritických záplat software pomocí CRONu (démon, mající na starost spouštění naplánovaných úloh), u Microsoft Windows existuje systém aktualizací Windows Update, případně Microsoft Update.

*NIXové systémy mají nespornou výhodu ve frekvenci aktualizací – narozdíl od firmy Microsoft, Linuxová komunita vydává opravy kritických zranitelností v okamžiku jejich vyřešení a nedrží se plánu vydávání záplat druhé úterý v měsíci. Server je tak lépe chráněn již pár hodin po odhalení chyby a nečeká se několik týdnů, během kterých je systém zranitelný.

U systému, založených na Debianu, se využívá balíkový správce Aptitude, který narozdíl od jiných (například Yast u SuSE) nemá vlastní GUI (Graphic User Interface - grafické uživatelské rozhraní). Provedení příkazu pro aktualizaci seznamu balíků („apt-get update“) vypadá následovně (včetně výstupu):

```
web:~# apt-get update
Hit http://debian.sh.cvut.cz lenny Release.gpg
Ign http://debian.sh.cvut.cz lenny/main Translation-en_US
Hit http://debian.sh.cvut.cz lenny Release
…
Fetched 90.2kB in 0s (330kB/s)
Reading package lists... Done
```

Tím je lokální verze seznamu balíků aktuální a Aptitude může vyhledat aktualizace pro software nainstalovaný na serveru. Aktualizaci lze vyžádat příkazem „apt-get upgrade“:

```
web:~# apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  dpkg dpkg-dev libapache2-mod-php5 libcups2 libcupsimage2
libmysqlclient15-dev libmysqlclient15off libpango1.0-0 libpango1.0-common
libpango1.0-dev
  linux-image-2.6.26-2-amd64 linux-libc-dev mysql-client-5.0 mysql-common
mysql-server mysql-server-5.0 php-pear php5 php5-cli php5-common php5-
curl
  php5-dev php5-gd php5-imap php5-mcrypt php5-mysql php5-xmlrpc sudo
28 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 77.5MB of archives.
After this operation, 122kB disk space will be freed.
Do you want to continue [Y/n]?
```

Potvrzením se provede stažení balíků z aktualizačního serveru a jejich zařazení do jádra – jsou spuštěny potřebné instalační skripty, které se postarají o zařazení CRON událostí a automatického spouštění aplikací při zapnutí systému, pokud je to nezbytné.

Výhodou Linuxu je balíková architektura, která umožňuje jednoduchou aktualizaci každé části zvlášť a v případě nekompatibility vyřadit jen nezbytně nutné balíky. Snadno se pak udržuje konzistentnost systému – každý balík v sobě nese informace o závislosti, je tak velmi nepravděpodobné, že by nastal konflikt verzí, nebo kritická chyba aplikace kvůli chybějící knihovně – Aptitude se o dodržování závislostí úzkostlivě stará.

7. Závěr

Bakalářská práce se zabývá postupy, které byly provedené při tvorbě portálu Prácička.cz, od prvotní konzultace s klientem na návrhu a funkčnosti, přes výběr vhodných technologií až po nastavení serveru. Tyto postupy jsou ověřené praxí jako vhodné pro nové webové projekty a jsou univerzálně použitelné pro všechny malé reklamní kanceláře a vývojáře k upevnění jistot při vývoji a vylučují nepříjemná překvapení, vyplývající z nejasné smlouvy. Mezi nejdůležitější body začátku vývoje tedy patří: shrnutí, ujasnění požadavků klienta a následné uzamčení (jakožto ochrana proti neustálým změnám), jejich analýza a porovnání s trhem, pro který bude web určen a výběr vhodných technologií, na kterých bude web postaven. Samotný vývoj je relativně jednoduchým procesem, který spočívá v rozdělení úkolu na atomické části, které jsou snadno realizovatelné a postupně navazující jedné na druhou. Velké usnadnění v tomto procesu nabízejí frameworky, které umožňují vytvořit jednoduchý web v řádu minut až pár hodin. Z v práci vypsaných se nejvíce osvědčil český Nette Framework autora Davida Grudla, který nabízí vynikající výkon, vlastní implementaci šablon s cachováním a ochranu před útoky.

Z krátkého dotazníku o webové bezpečnosti, který byl publikován na serveru Vyplň.to, vyplývá, že většina uživatelů internetu si je alespoň částečně vědoma rizik, která z něj hrozí a jsou seznámeni s akcemi, které jsou potřeba pro zajištění soukromí a jsou ochotni je provádět i přes drobné nepohodlí. Znají digitální podpis, umějí se vypořádat s propadlými certifikáty serverů a jsou opatrní v otevírání odkazů a souborů, které jim přijdou přes instant messaging, či elektronickou poštu. V dnešní době se navíc výrobci prohlížečů předhání v bezpečnostních prvcích, jakými jsou automatická blokace podezřelých stránek, zvýraznění domény v řádku adresy kvůli jednodušší kontrole správnosti adresy či ochrana proti Cross-Site Scriptingu. Přes to je nutné se aktivně přičinit o svou bezpečnost a nezjednodušovat útočníkům práci rizikovým chováním.

Webové aplikace jsou trendem budoucnosti a vývoj webových aplikací je byznysem, ve kterém se točí vysoké částky a je potřeba ho brát vážně. Technologie jsou dnes na špičkové úrovni a je jen na vývojářích a projekt-manažerech, jak jich dokáží využít pro kvalitní produkty, které budou obohacením pro český internet a české uživatele.

Seznam použitých zdrojů

1. **Asleson, Ryan a Schutta, Nathaniel T. 2006.** *AJAX.* Brno : Computer Press, a.s., 2006. stránky 17, 29. 80-251-1285-3.

2. **Castagnetto, Jesus, a další. 2001.** *PHP Programujeme profesionálně.* Brno : Computer Press, a.s., 2001. 80-7226-3102.

3. **DuBois, Paul, a další. 2010.** MySQL 5.0 Reference Manual :: 18.4.2 View Processing Algorithms. *MySQL.* [Online] Oracle Corporation, 2010. [Citace: 22. Leden 2010.] http://dev.mysql.com/doc/refman/5.0/en/view-algorithms.html.

4. **Grudl, David. 2009.** Nette Framework: MVC & MVP. *Zdroják.* [Online] Internet Info, s.r.o., 24. Březen 2009. [Citace: 3. Leden 2010.] http://zdrojak.root.cz/clanky/nette-framework-mvc--mvp/. 1803-5620.

5. **Huseby, Sverre H. 2006.** *Zranitelný kód.* Brno : Computer Press, a.s., 2006. stránky 132, 133. 80-251-1180-6.

6. **Kosek, Jiří. 1999.** *PHP - Tvorba interaktivních internetových aplikací.* Praha : Grada Publishing, 1999. stránky 19-21. 80-7169-373-1.

7. **Krčmář, Petr. 2008.** Exkluzivně: Linux je na 70 % českých serverů, Apache na 88 %. *Root.cz.* [Online] Internet Info, s.r.o., 21. Červenec 2008. [Citace: 22. Leden 2010.] http://www.root.cz/clanky/exkluzivne-linux-je-na-70-serveru-apache-na-88/. 1212-8309.

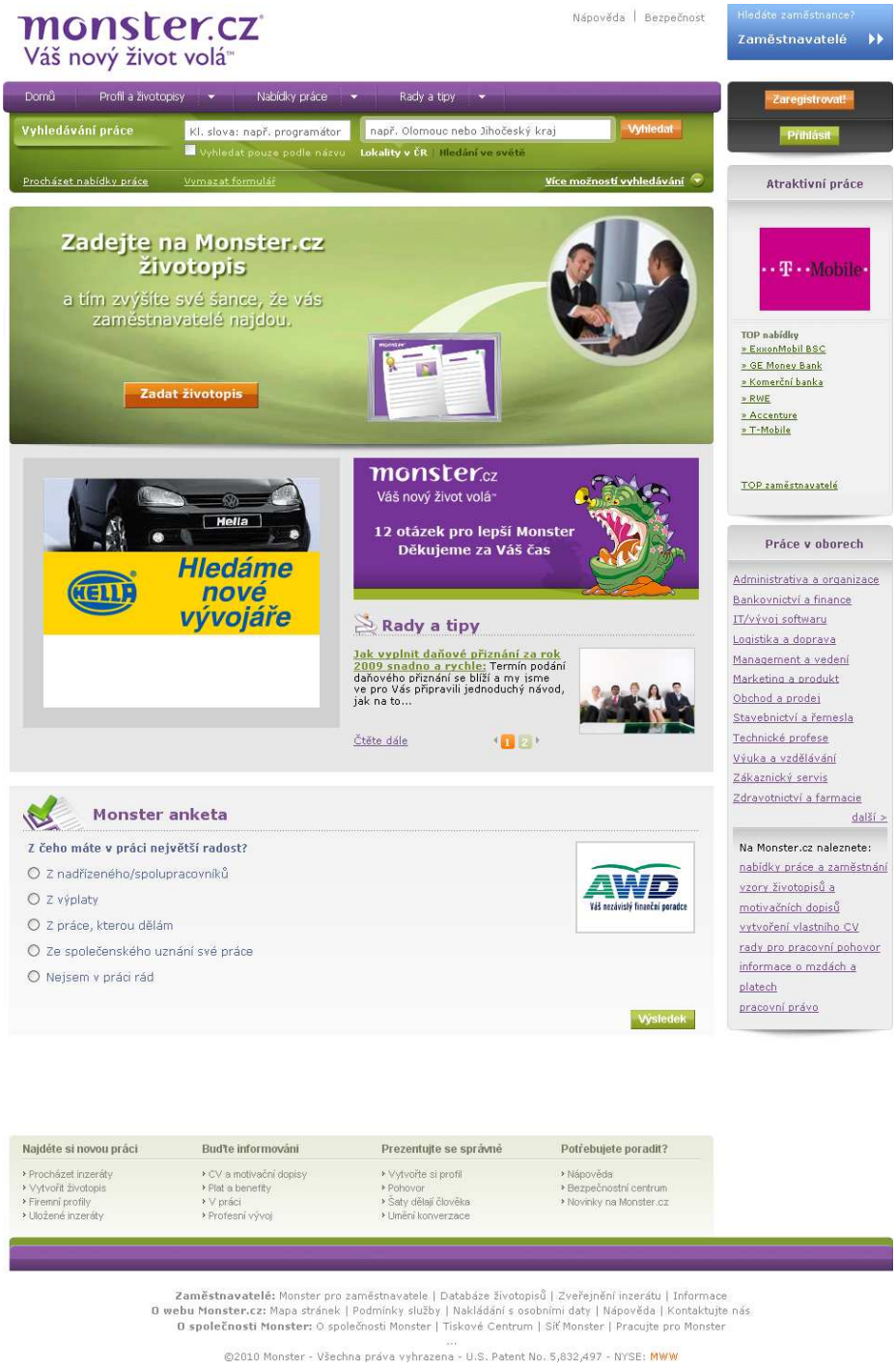
8. **Netcraft Ltd. 2010.** March 2010 Web Server Survey. *Netcraft.* [Online] Netcraft Ltd, 17. Březen 2010. [Citace: 18. Březen 2010.] http://news.netcraft.com/archives/2010/03/17/march_2010_web_server_survey.html.

9. **Schwartz, Baron, a další. 2009.** *MySQL profesionálně.* Brno : Zoner Press, 2009. stránky 243-246, 259, 260, 371-372, 381, 498. 978-80-7413-035-9.

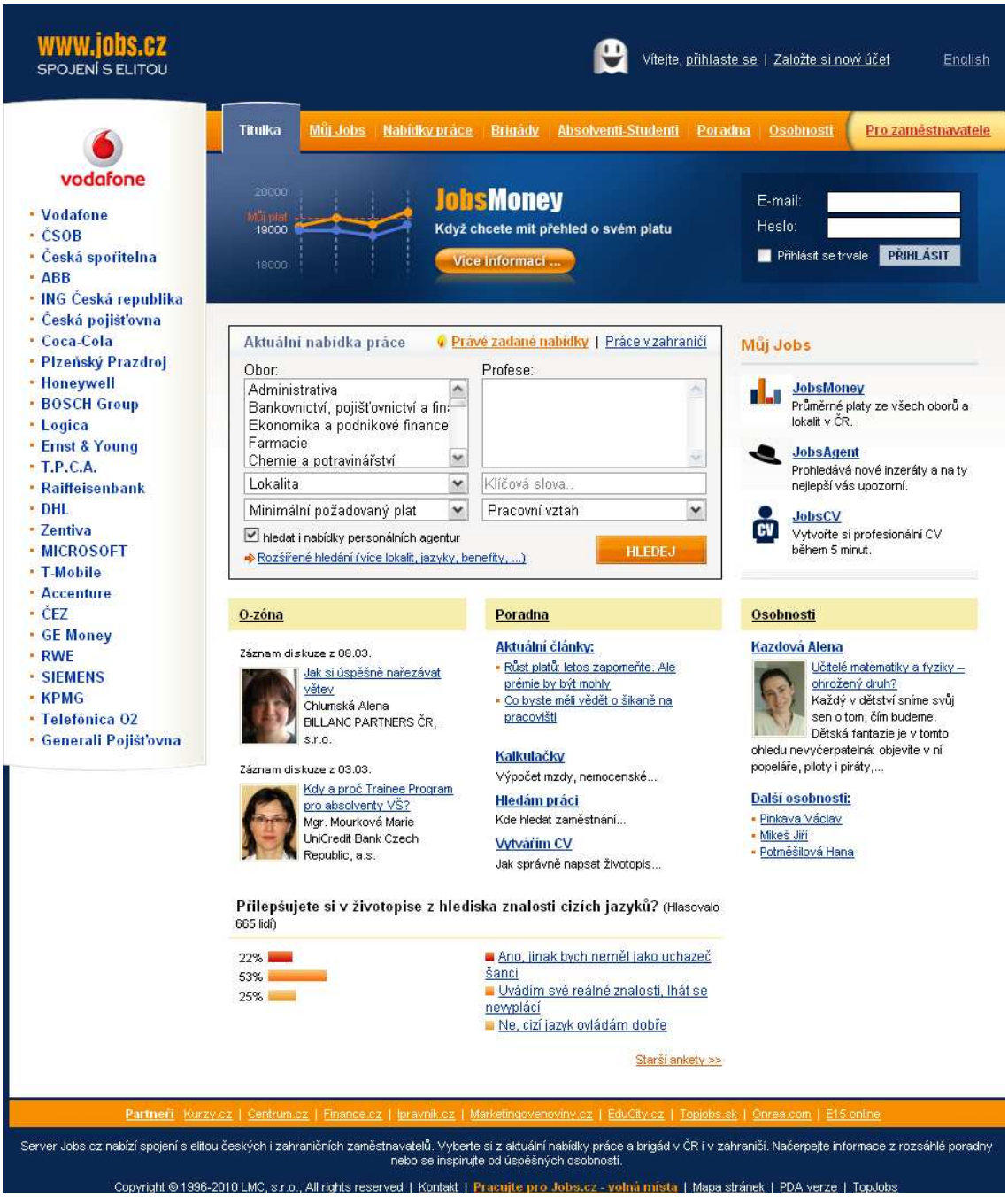
10. **SourceForge.net.** eAccelerator. *eAccelerator.* [Online] SourceForge.net. [Citace: 1. Březen 2010.] http://www.eaccelerator.net/

Přílohy

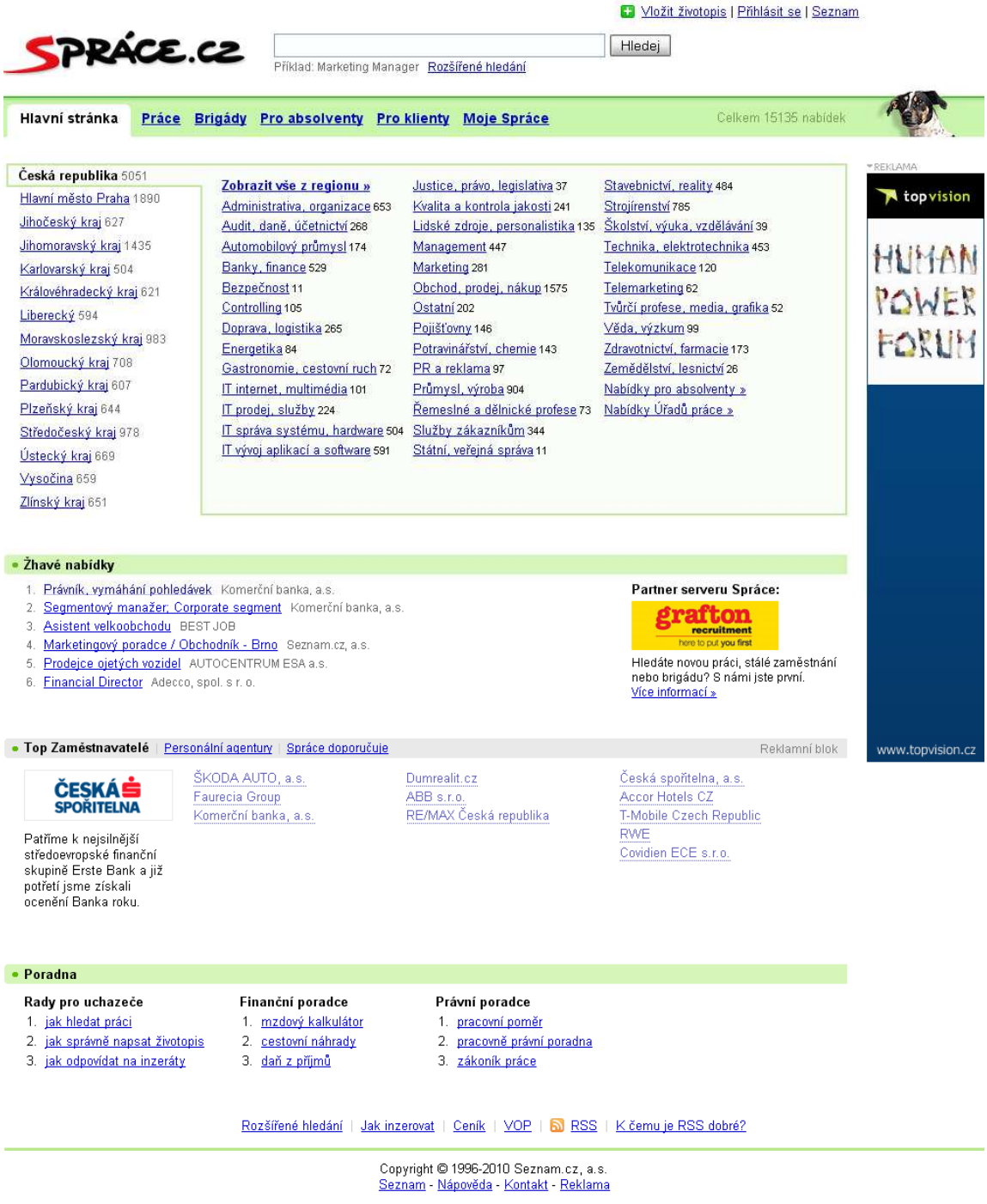
Příloha A – Titulka webu monster.cz



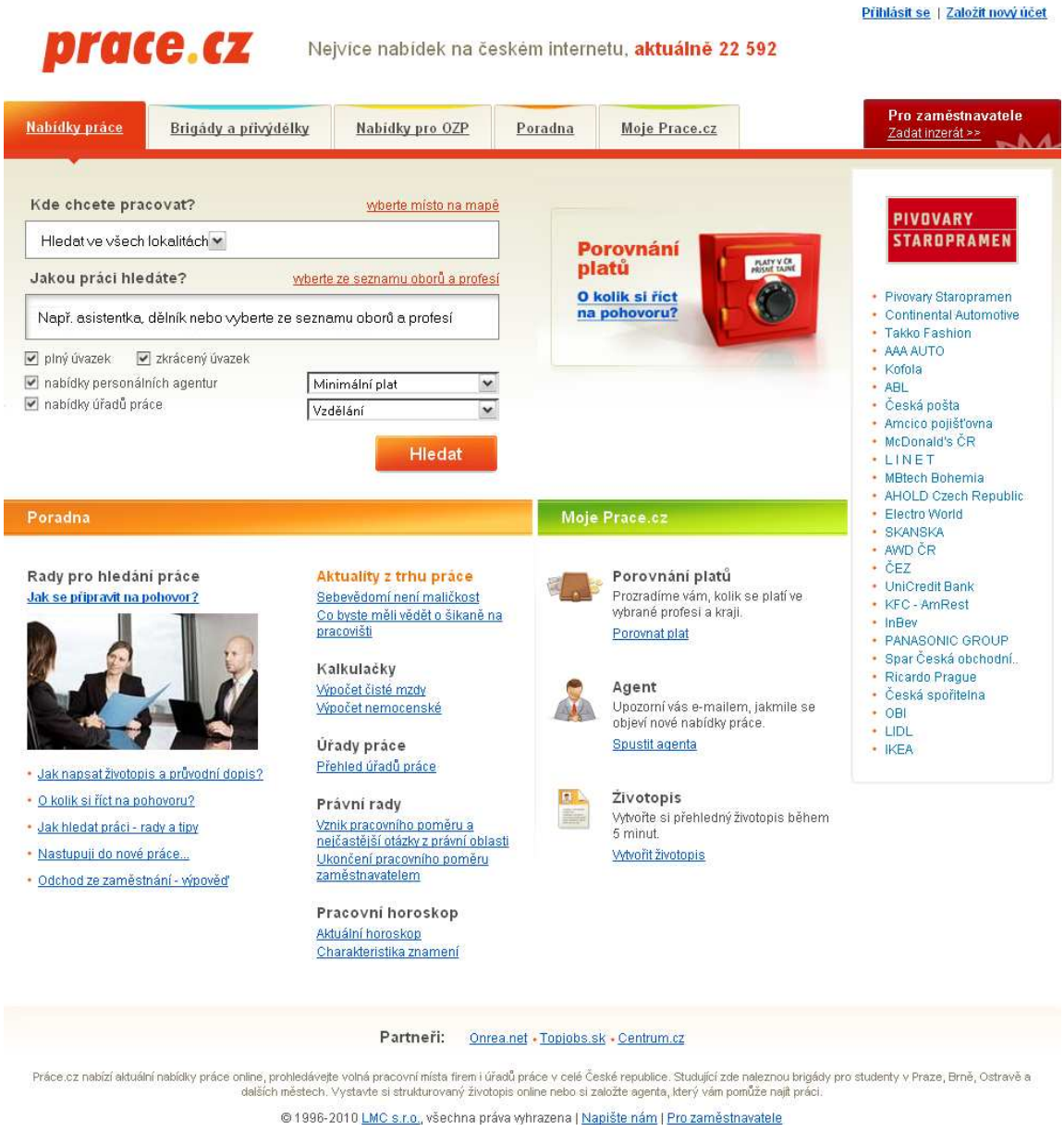
Příloha B – Titulka webu jobs.cz



Příloha C – Titulka webu sprace.cz



Příloha D – Titulka webu prace.cz



Příloha E –Návrh titulky webu pracicka.cz

Seznam tabulek

Seznam obrázků