VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

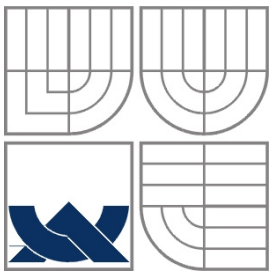
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TRANSFORMACE EDITAČNÍHO SYSTÉMU N.E.S.P.I. NA WEBOVÉ SLUŽBY

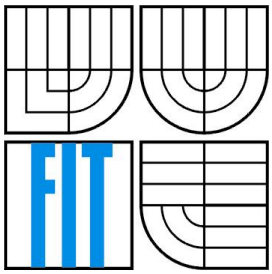
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. JAN FIALA
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TRANSFORMACE EDITAČNÍHO SYSTÉMU N.E.S.P.I. NA WEBOVÉ SLUŽBY

TRANSFORMATION OF N.E.S.P.I. SYSTEM INTO WEB SERVICES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. JAN FIALA
AUTHOR

VEDOUCÍ PRÁCE — ING. ŠÁRKA KVĚTOŇOVÁ, Ph.D.
SUPERVISOR

BRNO 2009

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2008/2009

Zadání diplomové práce

Řešitel: **Fiala Jan, Bc.**
Obor: Informační systémy
Téma: **Transformace editačního systému N.e.s.p.i. na webové služby**
Kategorie: Softwarové inženýrství

Pokyny:

1. Seznamte se s problematikou vytváření webových aplikací (HTML, XHTML, PHP) a architekturou orientovanou na služby (SOA).
2. Seznamte se s problematikou webových služeb (WS) - XML, SOAP, WSDL, HTTP, UDDI.
3. Proveďte detailní analýzu požadavků na transformaci editačního systému N.e.s.p.i. na webové služby (ve spolupráci se zadavatelem projektu).
4. Na základě provedené analýzy navrhněte vlastní koncepci systému. Zvolte vhodné implementační prostředí pro její realizaci.
5. Použitelnost systému demonstrujte na vhodně zvoleném vzorku dat.
6. Zhodnoťte dosažené výsledky, zejména použitelnost v reálném prostředí a diskutujte možnosti dalšího rozvoje vytvořeného produktu.

Literatura:

- Erl, T.: *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, 2005, ISBN 0-13-185858-0.
- Newcomer, E.: *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Addison-Wesley, 2002, ISBN 0-201-75081-3.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Bez požadavků.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Květoňová Šárka, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. července 2009
Datum odevzdání: 31. července 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Diplomová práce popisuje analýzu, specifikaci, vznik a transformaci stávajícího editačního systému N.E.S.P.I., jenž je duševním vlastnictvím firmy WebRex s.r.o. Hlavním cílem je oddělit administraci webových stránek od jejich vlastního obsahu viditelného na Internetu a docílit tak jednotného a komplexního administračního rozhraní pro zákazníky firmy. Zprovozněním služby není pouze úspora času a zjednodušení práce programátorů, ale také docílení toho, že editační systém nebude pouze duševním vlastnictvím firmy, ale také fyzickým, neboť zákazník bude vlastnit pouze webové stránky a možnost jejich administrace bude do budoucna poskytována jako služba. Dále by služba měla nabídnout zákazníkům moderní přístup k informačním technologiím a lepší podporu pro svoje podnikatelské plány a ideály.

Abstract

The diploma thesis describes the analysis, specification, origin and transformation of the current editing system N.E.S.P.I, which is the intellectual property of the company WebRex s.r.o. The main aim is to separate the administration of the web sites from their content, which can be seen on the Internet and by doing so to achieve the unified and complex administration interface for the customers. By bringing the service into work we do not only save time and simplify the work of the programmers, but we are also able to reach the stage when the editing system is not only the intellectual property of the company, but also its physical property, because the customer owns only the web sites, but the possibility to administrative them is offered as further service. The service should also provide the customers with modern access to informational technologies and propose enhanced assistance for their entrepreneurial plans and ideals.

Klíčová slova

ASP, editační systém, SOA, SOAP, XML, HTTP, CSS, PHP, SaaS, UDDI, WSDL

Keywords

ASP, edit system, SOA, SOAP, XML, HTTP, CSS, PHP, SaaS, UDDI, WSDL

Citace

Fiala Jan: Transformace editačního systému N.E.S.P.I. na webové služby, diplomová práce, Brno, FIT VUT v Brně, 2008

Transformace editačního systému N.E.S.P.I. na webové služby

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Šárky Květoňové, Ph.D.
Veškeré informace, požadavky a návrhy ohledně diplomové práce mi poskytl zadavatel projektu Dominik Debef a jeho zástupce Ivo Válal.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Bc. Jan Fiala
24-07-2009

Poděkování

Rád bych poděkoval všem, kteří mi jakýmkoliv způsobem pomohli při zpracování diplomové práce. Mé díky patří především Ing. Šárce Květoňové, Ph.D. za její ochotu, trpělivost a cenné připomínky, které mi poskytovala po celou dobu tvorby diplomové práce i za její veškerý čas, jenž mi věnovala. Dále bych také rád poděkoval mému konzultantovi a zadavateli projektu Dominiku Debefovi za poskytnutí informací týkající se vlastního zadání projektu, podněcování k novým a lepším vlastnostem systému ve fázi implementace, ale také za trpělivost při usměrňování mých myšlenkových pochodů. Nemalé díky patří i spolupracovníkům programátorům firmy za občasné rady i nápady k implementaci konkrétních problémů systému.

Obsah

1 Úvod

V posledních letech je čím dál více kladen důraz na využívání webových služeb (Web Services -WS), což je v dnešní době často skloňovaný pojem v oblasti informačních technologií (IT). Dnešní trend v informačních a komunikačních technologiích je přesun od prodeje softwarových produktů k poskytování služeb na internetu. Podle mnoha odborníků [26] představují služby standard pro novou generaci e-byznysu a to je podpořeno vývojem řadou špičkových firem, včetně takových gigantů jako jsou Microsoft, IBM či Novell.

Webové služby mají vhodné univerzální rozhraní pro přístup funkcionality a uložených dat v různých zdrojích, což mohou být různé informační systémy (IS) používané uvnitř společnosti i mimo ní. Toto řešení umožňuje integraci s informačními systémy jednoduše a efektivně. Realizační tým tak nemusí řešit problémy spojené s rozšiřováním informační infrastruktury, spočívající v nutnosti získaní znalostí různých technologií potřebných pro zajištění integrace. Dnes již WS představují standard pro komunikaci mezi online IS podporovanými mnoha výrobci softwarových produktů, za kterým stojí dostatečně velká komunita pro zajištění jeho bezproblémové podpory do budoucna. Řada IS již toto standardizované rozhraní WS podporuje implicitně, nebo je jeho volitelnou součástí. Některé softwarové balíky pak umožňují snadno doplnit rozhraní vlastním vývojem. Jako příklad informačních systémů umožňujících přístup ke svému obsahu pomocí webových služeb můžeme uvést kupř. SAP či Salesforce. Pomocí WS umožňuje přístup též převážná většina veřejně přístupných aplikací jako je eBay, Google, Amazon nebo Facebook [20].

Velkou výhodou webových služeb je umožnění bezproblémového spojení a spolupráci zcela různých systémů a nezávislost programovacího jazyka, technologie a platformy, které byly použity k provedení samotné aplikace a můžeme je vnímat jako další prvek vhodný při budování distribuovaných systémů. Z tohoto důvodu se v současné době velmi často a intenzivně využívají v prostředí podnikových IS pro realizaci komunikace mezi jejich jednotlivými aplikacemi, ale také jako způsob a prostředek integrace samostatných aplikací do společného IS. Přesto však standardně neřeší všechny požadavky kladené na informační systém.

Strukturovaně orientovaná architektura (SOA), jíž webové služby implementují, přitahuje zájem všech oblastí IT průmyslu a je poháněná moderními komunikačními standardy (XML, SOAP, UDDI, WSDL, XSD, BPEL a WS* standardy pro bezpečnost a transakčnost (viz kapitola 2.3.2). Díky nim rychle proniká do hlavních chodů aplikací zásadních pro plnění business operací. Dnes existuje velké množství technologií, které se snaží řešit různé potřeby WS. Problém je v tom, že mnohé technologie se překrývají ve své aplikační doméně, konkurují si navzájem a specifikace k některým z nich nejsou v konečné verzi. Další problém představují samotné implementace těchto technologií. Pro některé podpůrné technologie neexistuje implementace nebo jich je více. Další technologie jsou komerční a druhé jsou volně dostupné (Open source). Různé technologie jsou stabilní nebo naopak mají ještě řadu nedostatků resp. chyb a jsou ve stádiu vývoje. Najdou se i takové, jenž mají dodatečné požadavky, např. vyžadují speciální implementaci webových služeb resp. určitou verzi této implementace.

S webovými službami je také velmi provázaný pojem Aplication Service Providing (ASP), což lze přeložit jako "*poskytování aplikačních služeb*" [40]. Poskytování aplikačních služeb mění licence na službu, nikoliv jen na pronájem, jak se někdy ASP mylně interpretuje. Hovoříme o službě, neboť příjemce dostává k dispozici přímo aplikační výkon daných aplikací v definovaném rozsahu. V praxi vše funguje tak, že uživatelé si určité softwarové aplikace přímo nekupují, ani je sami neprovozují ve vlastní režii, ale pouze je používají. Veškeré činnosti spojené s péčí o řádný chod využívaných

aplikací se pak stanou odpovědností specializovaných poskytovatelů aplikačních služeb tzv. providerů [38].

Tato diplomová práce se zabývá problematikou transformace editačního systému N.E.S.P.I. (Nový Editační Systém Pro Internet) [44] na systém využívající principy právě technologie ASP (Application Service Providing) s využitím webových služeb. Vznik N.E.S.P.I. je datován do roku 2000 jako jednoduchý nástroj pro tvorbu a úpravu webových stránek. Nyní již ve třetí verzi jsou jeho možnosti technologicky překonány a cílem diplomové práce je vytvořit N.E.S.P.I. ve čtvrté verzi jako centralizovaný systém poskytující editaci webových stránek jako službu pro zákazníky firmy WebRex [44].

Druhá kapitola je zaměřena na teoretické pojetí technologií ASP, SOA a webových služeb, jak již bylo zjednodušeno v úvodu práce. Každou technologii si blíže představíme, řekneme si výhody a nevýhody a podíváme se i do jejich historie. V závěru této kapitoly jsou uvedeny nejznámější webové služby, kterými můžeme implementovat SOA.

Ve třetí kapitole se věnujeme analýze a specifikaci základních požadavků zadavatele na vyvíjený systém. Jsou zde popsány útoky, proti kterým má být systém odolný a poté jsou uvedeny E-R diagramy klientských stan a serverové části systému a také diagram použití.

Následující čtvrtá kapitola se věnuje samotné implementaci navrženého systému za použití popsaných technologií. S implementací do značné míry souvisí testování, jenž je uvedeno v 5. kapitole. Na závěr zhodnotíme dosažené výsledky a nastíníme možnosti dalšího budoucího rozvoje vytvořeného systému a částí, které nebyly součástí diplomové práce nebo nestihly být implementovány.

2 Teoretické pojetí webových služeb

V této kapitole se budeme zabývat teoretickým pojetím webových služeb a nejznámějšími technologiemi s nimiž WS souvisí. Nejdříve si popíšeme technologii ASP, na kterou je převážně zaměřena diplomová práce. Poté si podrobně popíšeme architekturu orientovanou na služby (SOA). A pak přijdou na řadu jednotlivé technologie webových služeb.

2.1 ASP – Application Service Providing

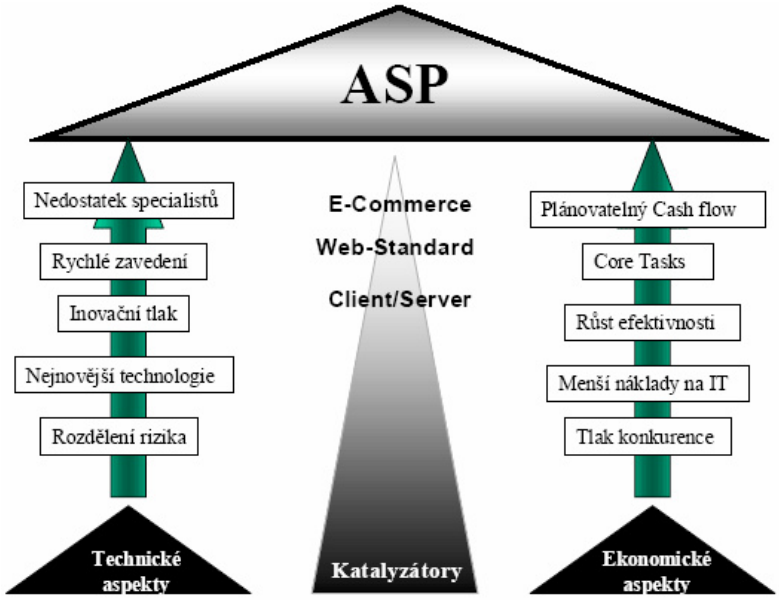
Jak již bylo zmíněno v úvodu diplomové práce, ASP je poskytování aplikačních služeb (software) prostřednictvím Internetu a jedná se o jednu z forem outsourcingu [18]. V praxi to znamená, že dodavatel zákazníkovi zajišťuje běh aplikací na vlastní nebo pronajaté softwarové a hardwarové infrastruktuře, a tím dodavatel zajišťuje dodávku software, hardware, síťových technologií, analytických, implementačních a softwarových služeb. Pojem ASP je vhodné spíše přirovnat ke známějšímu termínu Internet Service Provider (ISP), kdy si zákazník pronajímá připojení k Internetu [40]. Obdobně to funguje i v ASP, ale není to zase až tak jednoduché. Celou problematikou spojenou s ASP se budeme věnovat v této podkapitole.

2.1.1 Co je ASP?

Termín ASP je v obecném podvědomí znám především jako zkratka Active Server Pages, což je technologie ke generování dynamického obsahu WWW [11] stránek obdobně jako technologie PHP (viz Kapitola 4.1.3). My se ovšem budeme zabývat pojmem ASP jako zkratkou Application Service Providing, případně Application Service Provider a můžeme hovořit i o Application Service Provision. Nicméně se jedná o technologii, která má mnoho zajímavých vlastností, přičemž jednou z nich je i to, že není úplně triviální vysvětlit o co přesně se jedná.

Jak se v praxi ukazuje, jednou z největších překážek v prosazování ASP je nedostatek osvěty, nepochopení celé podstaty, nedocenění nabízených možností a nedostatečné podvědomí o výhodách a přínosech technologie ASP. Na druhé straně je nutno podotknout, že nízká úroveň povědomí o ASP je z důvodu neexistence jasné a srozumitelné, ale přesto přesné a vyčerpávající definice. Navíc v chápání ASP neexistuje všeobecný společný postoj o tom, co sem ještě patří a co již nikoli.

Jeden z pokusů o definici ASP [16] říká, že jde o jakousi směs či výslednici nových technologických možností a nových obchodních modelů. To naznačuje, že zde sehrávají významnou roli jak aspekty z oblasti IT, tak i aspekty z oblasti obchodních modelů a praktik, jak naznačuje následující Obrázek č. 1.



Obr. 1: Technické a ekonomické aspekty ASP [16]

Druhá, o něco delší, a také výstižnější definice [18] uvádí: „*Specializovaná firma (Application Service Provider) na vlastní nebo pronajaté informační a komunikační technologii provozuje služby, které nabízí k použití externím zákazníkům. Služby v sobě integrují software, hardware, síťové technologie a vhodné konzultační služby do jednoho balíku. K jejich využití je z technologického hlediska obvykle potřeba pouze připojení k Internetu a webový prohlížeč na osobních počítačích uživatelů zákazníka. ASP poskytují své aplikace velkým i malým podnikům i spotřebitelům.*“

S velkým rozmachem Internetu v poslední době došlo k tomu, že různé konkrétní činnosti a aktivity můžeme plnohodnotně a pohodlně vykonávat na dálku. Navíc mohou poskytovatelé obsluhovat více zákazníků, a tím jim umožnit koncentraci jejich kapacit a zdrojů na jednom místě a docílit tím vyšší efektivnosti. Dalším významným důsledkem je změna vztahu uživatele a jím používané aplikace. Díky ASP již není nutné, aby uživatelé kupovali jednotlivé aplikace a sami měli na starosti jejich správu, aktualizace atd. Dnes aplikace vlastní poskytovatel, který musí být vhodně vybaven po stránce infrastruktury, know-how i všeho ostatního. Poté již může nabízet používání aplikací svým zákazníkům jako svou službu.

Tím jsme si výstižně a ve stručnosti popsali základní aspekty technologie ASP a můžeme přejít k podrobnějšímu výkladu výhod a nevýhod.

2.1.2 Výhody a nevýhody ASP

Potřeba zavést ASP model se vyvinula z narůstajících nákladů na provoz specializovaných softwarových systémů, které překračovaly přijatelný cenový rozsah zejména pro malé a střední firmy. Rovněž narůstající komplexnost a složitost softwarových řešení vedla k velkým nákladům na distribuci software k uživatelům. Z tohoto důvodu je prioritní vlastností ASP podstatné snížení nákladů na pořízení aplikačního softwaru. Navíc se podstatně zjednodušil systém distribuce upgradů (nové verze), jelikož nemusí být zasílány k zákazníkovi a zde instalovány, protože aplikace jsou provozovány na serverech dodavatele přístupných z Internetu.

Nyní si podrobněji popíšeme nejvýznamnější vlastnosti ASP a ukážeme si, co zákazníci mohou získat se zavedením tohoto modelu a co na druhou stranu mohou ztratit tím, že opustí tradiční způsob provozu aplikace. Následující popis vlastností byl převzat z [18]. Nejdříve tedy přejděme k výhodám.

Výhody:

- **Menší pořizovací náklady –** vývoj vlastního IS je poměrně náročná činnost a je třeba mít poměrně hodně finančních prostředků a také znalostí. Kromě toho u ASP řešení není třeba kupovat licence, které jsou jinak při vlastním vývoji téměř nepostradatelné.
- **Rozšiřitelnost** – dodavatel je většinou schopen dodávat služby velkým i malým podnikům a podle toho rozšiřovat či snižovat kapacity. Provider je schopen lépe obstarat správu systému, protože se na tento obor specializuje.
- **Dostupnost** – k aplikačním službám se zajišťuje přístup přes Internet a dnes již není problém se dostat do systému přes jiné místo než jen ze sídla firmy a ne nutně z osobního počítače, ale i ze zařízení jako jsou PDA nebo chytré telefony (smartphony).
- **Zabezpečení** – dodavatel má k dispozici rozsáhlý tým IT specialistů, kteří zajišťují provoz aplikace, ale i její bezpečnost, což v dnešní době je jedna z klíčových vlastností softwarových produktů a u webových služeb to platí dvojnásob. Dále je nezbytné zálohování, antivirové ochrany a monitorování funkčnosti aplikace.
- **Aktuálnost** – tato vlastnost vyplývá z předešlých dvou. Při nutnosti změny aplikace, vyvolané legislativními požadavky nebo přidáním nového bezpečnostního kódu, je vždy aktualizace umožněna v co nejkratším čase a na jednom místě u poskytovatele. Tím samozřejmě odpadají servisní výjezdy k zákazníkovi a zrychluje se tak technická podpora.

Nevýhody:

- **(Ne)dostupnost** – jelikož tato vlastnost byla uvedena jako výhoda, musíme jí zároveň uvést i jako nevýhodu, protože připojení pomocí rychlého Internetu není vždy možné uskutečnit. Zároveň hrozí přehlcení komunikačních linek mezi poskytovatelem a dodavatelem a může tak dojít k výpadkům v síti. Důsledkem pomalejšího připojení mohou nastat zhoršené podmínky pro práci s aplikaci.
- **Závislost** – zákazníci jsou do značné míry závislí na poskytovateli aplikačních služeb a musí se spoléhat na jím dokonalé zajištění provozu a zákazník se tak dostává do vysoké závislosti na externím partnerovi a současně je poskytovatel pověřen řízením citlivých podnikových dat.
- **Bezpečnost** – i když má poskytovatel dostatečně zkušený personál, neznamená to, že v systému neexistují *„zadní vrátka“*. Mohou existovat různá bezpečnostní rizika (odposlouchávání dat po síti, potenciální ztráta dat). Jaká jsou další bezpečnostní rizika si přiblížíme v druhé kapitole.

Uvedené vlastnosti jsou pilíři modelu ASP a patří mezi nejvíce diskutované. V úvodu kapitoly jsme se zmínili o tom, že je ASP jednou z forem outsourcingu a nyní bychom si ukážeme vztah ASP a outsourcingu. Následující odstavec vychází z literatury [36].

Žádná společnost nevlastní licence aplikací z touhy je vlastnit, nýbrž z potřeby je užívat a v obrovské míře je ani nezajímá, na jakých serverech a operačních systémech jsou dané aplikace provozovány. Důležité jsou pouze účely, pro které je společnost má. Těmto požadavkům by klasický outsourcing v zásadě ještě vyhověl, pomineme-li fakt, že serverová infrastruktura se zpravidla nachází v prostorách zákazníka. Klasický outsourcing však stojí před problémem efektivnosti správy méně obvyklých aplikací. Jak má poskytovatel outsourcingu zajistit kvalitu služby, když se mu

nevyplatí vyškolit správce, kteří budou danou aplikaci udržovat? To nepřinese příjemci outsourcingu ani nižší cenu a ani vyšší kvalitu, než jakých by dosáhl sám.

Z předešlých odstavců víme, že ASP je v tomto ohledu dominantnější. Společnost poskytující ASP, je schopna snáze investovat do odbornosti svých pracovníků, neboť jsou určeny velkému počtu zákazníků. Může se rovněž více specializovat i na procesy související s problematikou správy a provozu konkrétních aplikací. Oproti poskytovateli ASP má outsourcer podstatně menší šanci, že se mu mezi klienty vyskytne nadkritické množství zájemců o danou aplikaci. Jinými slovy, poskytovatel ASP může nabídnout lepší poměr cena/kvalita, jelikož má přirozeně vyšší výnosy z rozsahu.

Tím jsme si výstižně a ve stručnosti popsali převažující výhody ASP a můžeme přejít k tématu, proč model vznikl a jakým vývojem prošel.

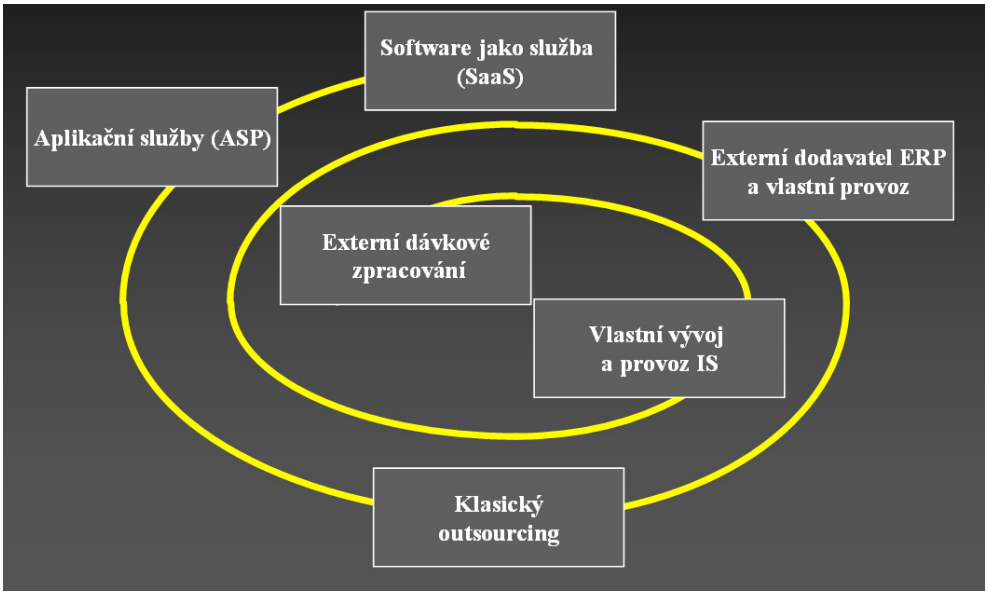
2.1.3 Historie a vývoj ASP

V této kapitole si přiblížíme jak ASP vzniklo a co bylo jeho předchůdcem, ale také pokračovatelem. Abychom to vzali postupně od začátku, musíme si přiblížit rozvoj IS, díky kterým získali webové služby značnou popularitu. Jak jsme předeslali v úvodu, představují WS standard pro komunikaci mezi IS. Pro pochopení celé problematiky IS a business procesů pro velké podniky si musíme definovat další důležitý pojem týkající se celé problematiky a to ERP - *Enterprise resource planning.*

ERP (Ekonomický účetní systém) je [38]: „*manažerský IS, který integruje a automatizuje velké množství procesů souvisejících s produkčními činnostmi podniku. Typicky se jedná o výrobu, logistiku, distribuci, správu majetku, prodej, fakturaci, a účetnictví. Za předpokladu, že systém je správně implementován, přináší řadu výhod. Především: zefektivnění a zrychlení ekonomických procesů, centralizaci dat, dlouhodobé úspory v investicích do informačních systémů a hardware. V konečném důsledku zvyšuje flexibilitu, takže i konkurenceschopnost. Výrobní firmy jsou stále více nuceny optimalizovat své výrobní procesy a zvyšovat celkové využití strojů, lidí a materiálů, což s sebou nese vysoké nároky na výrobní management z hlediska řízení a plánování výroby. Pro správné rozhodování je nutné mít informace o technologiích, postupech, kapacitách a kritických místech ve výrobě. Implementace informačního systému splňujícího výše uvedené požadavky na moderní řešení problematiky výroby je jedinou cestou jak dostat výrobní procesy pod kontrolu.*“

Systémy ERP bývají považovány za jádro celého firemního IS a nabízejí komplexní pohled na finanční zdroje podniku a pokoušejí se pokrýt základní funkce organizace, nehledě na typ organizace nebo její činnosti. Typický ERP systém využívá k dosažení integrace množství softwarových komponentů (modulů) a hardwarové infrastruktury. Klíčovou roli hrají unifikované databáze k ukládání dat, jenž pak používají různé moduly. ERP systémy dnes využívají nejen obchodní společnosti, ale také neziskové organizace, nevládní organizace, státní instituce a jiné velké entity.

Nyní můžeme přejít k tomu, jak se dostat od systémů ERP až k ASP a jeho následovníkovi SaaS. Vývoj řešení podnikových IS naznačuje následující Obrázek č.2. Následující text podkapitoly je převzat z internetového portálu Centra pro Výzkum Informačních Systémů [33].



Obr. 2: Vývoj řešení podnikových IS [41]

Koncem 90. let, s rozmachem Internetových služeb do firem, převzali dodavatelé aktivitu ve vývoji IS do svých rukou. Časové intervaly mezi zásadními změnami v nabídce podnikových IT řešení se výrazně zkrátily, a to bez podstatné změny v poptávce zákazníků. Přelom nového tisíciletí by bylo možno z hlediska vývoje nabídky ERP systémů charakterizovat třemi, rychle po sobě následujícími fázemi.

První z nich představuje tradiční způsob implementace ERP systémů, který spočívá v budování, resp. upravování podnikových aplikací podle individuálních potřeb zákazníků. Ta představuje snahu uspořit vysoké náklady na úpravy softwaru, při nichž je nutné využít služeb programátorů (customizaci). Opakovatelná podoba podnikových aplikací kromě úspor přináší také prvek standardizace a nabídku nejlepších praktik, pokud jsou tato přednastavená řešení založena na dlouhodobých praktických zkušenostech výrobce v jednotlivých odvětvích.

Další fázi vývoje představuje pronájem ERP systémů po Internetu. ASP nabídlo novou cestu, jak zpřístupnit špičková softwarová řešení především menším organizacím, které si nemohly dovolit jejich pořízení. Tento trend byl do velké míry způsoben tzv. *Internetovou horečkou* (anglicky „dot-com bubble“) [28], jenž motivovala řadu firem přesunout své obchodní aktivity na Internet s vynaložením velkých investic. V roce 2001 nízká výnosnost těchto investicí způsobila ztrátu důvěry akcionářů a to mělo za následek krach mnoha společností a termín ASP dostal mezi potencionálními zákazníky velmi špatný zvuk. Pro mnoho společností zůstalo důvěryhodnější pořízení ERP systému formou klasického nákupu software.

Od roku 2005 se objevuje opětovný zájem dodavatelů o zavedení progresivnějších modelů dodávky a provozu ERP systému a dalších aplikací, zejména CRM systémů (Customer Relationship Management - řízení vztahů se zákazníky), kterou charakterizuje pojem SaaS – Software as a Service. S neúspěchem Internetové horečky se technologie ASP dostala do pozadí a po opětovném získání důvěry v Internet a IT se začal používat právě termín SaaS. Ten bývá často ztotožňován z ASP, a to z čistě marketingových důvodů, neboť ASP bylo kompromitováno příliš těsnou vazbou na „dot-com bubble“. Oba modely mají podobné charakteristiky, ale existují však mezi nimi jisté rozdíly: [28].

- Model SaaS je vždy 1:N („one-to-many“) ve vztahu k zákazníkovi a je o něco složitější ho implementovat, hlavně co se týká upgradů a customizace.

- V modelu SaaS platí, že poskytovatel aplikace je současně také jejím výrobcem, což u ASP vždy neplatí. Poskytovatel mnohdy kupoval aplikaci od výrobce a nabízel ji k pronajmutí.
- Aplikace SaaS jsou založeny na Internetu a jeho protokolech. Oproti tomu u ASP platí tvrzení, že jde o aplikaci „klient-server“.
- Model SaaS se snaží implementovat standardy a adoptovat se více na Service-Oriented Architecture (SOA).

Přibližně ve stejné době s nástupem SaaS se objevuje stále více ERP systémů postavených na bázi právě servisně orientované architektury SOA. Oba uvedené trendy, SaaS i SOA, by mohly z dlouhodobého hlediska zcela změnit celou řadu věcí v praxi podnikové informatiky, zejména pak obchodní modely dodávky. Problematiku a podrobné seznámení se SOA nám nabídne další podkapitola.

2.2 SOA – Service-Oriented Architecture

Oblasti výpočetní techniky se obrací směrem k distribuovaným výpočetním systémům. Prvkem držícím tyto systémy pohromadě musí být vhodná architektura. Tou nejčastěji zmiňovanou je SOA (Servisně Orientovaná Architektura), která je všeobecně chápána a přijímána jako další fáze budování podnikových IS [8]. Architektura SOA umožňuje odstranit nevhodné podnikové systémy a jejich procesy, jenž vzdorují změnám, a mohou včas učinit správná rozhodnutí v organizaci. Výsledkem je akceschopný podnik, který může rychleji reagovat na změny na trhu, v reálném čase přizpůsobit svou činnost okolnostem a nabízet nové produkty a služby rychleji než konkurence, a tak zvýšit výkonnost podniku při současném snížení nákladů na IT a zlepšení flexibility podnikových procesů [2].

Termín SOA lze slyšet z úst manažerů i odborníků v IT, pokud se jich však zeptáme co SOA přesně znamená, tak jen hrstka z dotázaných dokáže odpovědět správně a navíc má každý na věc jiný názor. V této kapitole si blíže vysvětlíme pojem SOA a k čemu je vhodné architekturu použít.

2.2.1 Co víme o SOA

Počátky SOA spadají do počátků prvních informačních systémů (tj. zhruba do 60. let dvacátého století). Myšlenka je tedy již poměrně stará a v době jejího vzniku nebylo k dispozici dostatečné technické vybavení (rychlá počítačová síť) a začíná se prosazovat až v současné době, kdy je technické vybavení na dostatečné úrovni. Analytici se shodují na tom, že velké monolitické celopodnikové systémy stále více brání změnám podnikových procesů, protože neumožňují efektivně a za krátký čas přidat nový proces nebo připojit aplikaci, která nový proces realizuje. Další problém je, že tyto systémy neumí zavést výrobek, který se bude prodávat jinak, než se předpokládalo v době zavádění systému [9].

Tento nedostatek řeší SOA díky servisně orientovanému přístupu k organizování IT zdrojů pomocí jednotného řešení, které snadno kombinuje stávající starší a nové technologie, a to má za cíl maximální zvýšení flexibility managementu v podniku. Tímto lze rychle propojit podnikové procesy, což dává podnikům schopnost dostatečně reagovat na měnící se podmínky trhu. Dále dovoluje získávat informace mnohem snadněji a v mnohem přístupnější podobě. Koncový uživatel si může zvolit formu výstupních dat a tato data dále zpracovávat a prohlížet na různých zařízeních (webový portál, aplikační klient, mobilní telefon) [8].

Kvůli globálním trhům jsou firmy nuceny k dosahování vyšší výkonnosti, produktivity, rychlého zavádění produktů. To má za následek velké nároky na IT oddělení, aby vyráběla jednoduché a flexibilní systémy při nižších nákladech. Největším problémem je oddělení jednotlivých systémů podniků a to např. již zmiňovaný ERP (viz kapitola 2.1.3) nebo CRM (Customer Relationship Management - řízení vztahů se zákazníky), které pracují samostatně a sloučení informací poskytovaných těmito systémy je velmi problematické. Dříve byla uplatňována časově náročná manuální řešení nebo byl problém řešen pomocnými programy, jenž snižovaly požadovanou flexibilitu a udržovatelnost systémů.

Jak již bylo zmíněno, jsou webové služby základním pilířem Architektury SOA, jenž je postavena na konceptech distribuovaného a modulárního programování. Řekli jsme si, že WS implementují SOA, ale není to pravidlo. SOA můžeme realizovat jakýmkoliv způsobem založeným na službách. SOA na rozdíl od tradičních architektur využívá silně spolupracující aplikační služby, které kooperují na základě jejich formálního popisu a ten je nezávislý na programovacím jazyku. Z toho vyplývá, že je zmiňovaná architektura nezávislá na vývojové platformě. Zajímavý je fakt, že jednotlivé komponenty mohou být implementovány na odlišných platformách, ale ty již jsou slabě vázané na daný operační systém a programovací jazyk. SOA odděluje funkce do jednotek, které mohou být distribuované přes síť a jejich kombinací a znovupoužitím umožňují vytváření business aplikací. Služby pak svou vzájemnou komunikací a koordinací vytvářejí služby nové. Tyto služby pak označujeme jako business procesy, které jsou pak doménou firmy a významnou měrou podporují firemní softwarovou infrastrukturu a umožňují manažerům a business specialistům podílet se na jejich návrhu. Tím mohou snadno a rychle měnit požadavky v závislosti na trhu [14].

2.2.2 K čemu slouží SOA?

Následující podkapitola vychází z literatury [14]. Metodika SOA byla navržena k použití v oblasti businessu a podnikových systémů, proto má zde úspěšné použití. Podnikové aplikace jsou velmi často specializované a z tohoto důvodu se mezi sebou špatně integrují. Proto existuje řešení v podobě SOA, jenž nabízí integraci a využití již dostupných zdrojů tím, že jednotlivé funkce zabalí do služeb. A z těch lze posléze skládat procesy a vytvářet vyšší přidanou hodnotu podnikové infrastruktury. Další významnou vlastností servisně orientované architektury je schopnost integrovat do procesu i další služby, jenž jsou nejsou vlastnictvím dané organizace, ale jejich obchodních partnerů .

Pojem služba má ve vnitropodnikové organizaci značný význam, neboť je to termín dostatečně abstraktní k tomu, aby jej používali manažeři a lidé z business oddělení. Na druhou stranu je i dostatečně konkrétní k tomu, aby byla implementována lidmi z IT. Služba je tak nezbytný krok k zefektivnění pracovního cyklu a navíc se vytvoří snadné spojení mezi businessem a IT.

SOA také umožňuje v kombinaci s Business Process Managementem (BPM) dynamicky aplikovat řízení business procesů v rámci firemní infrastruktury, která je postavená na znovupoužitelných službách a dojde-li ke změně business procesu, pak realizace takovéto změny je díky SOA mnohem rychlejší a jednodušší, což má za následek vyžadovanou flexibilitu.

V následující podkapitole se budeme zabývat modelem zralosti SOA. Účelem tohoto modelu je poskytnout IT manažerům rámec pro posouzení strategické hodnoty zavedení SOA v organizaci.

2.2.3 Model zralosti SOA

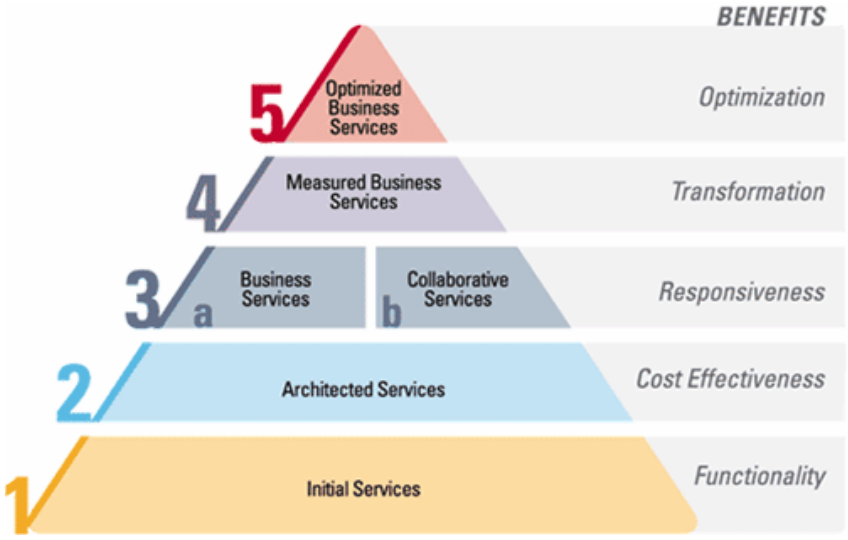
Myšlenky hodnocení úrovní zralosti jsou aplikovány na oblast SOA již koncem roku 2005. Důvodem zavedení tohoto modelu je poskytnout IT manažerům představu pro posouzení strategické hodnoty zavedení SOA ve společnosti. Model zralosti (Maturity Model SOA) ukazuje zvyšující se přínosy

pro podnikání při zavedení vyšší úrovně zralosti SOA a ukazuje, jak se na jednotlivých úrovních zralosti mění cíle, charakteristiky a předpoklady vlivu SOA na podnikání.

SOA MM pro každou úroveň zralosti definuje cíle, rozsah, vliv na podnikání, důležité průmyslové standardy, klíčové praktiky a kritické faktory úspěchu, jak technologické, tak organizační. Dále poskytuje návod, jak zavést SOA a měřit pokrok jeho zavádění a specifikuje potřeby pro zavedení dané úrovně zralosti SOA:

- Metodiky pro analýzu, návrh a implementaci SOA.
- Modelovací a vývojové nástroje, které umožní specifikaci a vytváření služeb a jejich propojení na byznys procesy.
- Enterprise Service Bus (ESB) pro poskytování spolehlivé, rozšiřitelné, distribuované komunikace a transformace dat mezi službami.
- Registr a repository služeb a politik jako jediné místo pro uspořádání a řízení SOA informací včetně katalogu dostupných služeb, definice jejich rozhraní a politik pro používání služeb.
- Nástroje pro provoz a řízení infrastruktury včetně monitorování využívání služeb a zjišťování metrik.

Obrázek č.3 ukazuje pět úrovní SOA zralosti a klíčový vliv na podnikání, včetně mapování na úrovně zralosti CMMI (Capability Maturity Model - Stupňovitý model zralosti).



Obr. 3: Model zralosti SOA [32]

Jednotlivé úrovně mají následující klíčové charakteristiky:

- **SOA MM úroveň 1 – Počáteční služby (Initial Services).** Tato úroveň zralosti reprezentuje fázi učení a počáteční fáze zavádění SOA, kde jsou typické projekty zaměřeny na implementaci funkcionality pomocí nových technologií a testování SOA technologií v laboratorním prostředí. Zavedení SOA je iniciováno ze strany vývojářské organizace a SOA je zpravidla součástí projektu na integraci aplikací. Na této úrovni zralosti se implementují základní standardy pro služby (viz Následující kapitola 2.3), formují se nové dovednosti potřebné pro vývoj služeb a definují se základní metriky hodnocení.
- **SOA MM úroveň 2 – Služby zasazené do architektury (Architected Services).** Na této úrovni zralosti jsou již zavedeny technologické standardy SOA a zavádění SOA je kontrolované a řízené

architektonickou organizací. Klíčovým přínosem na této úrovni je snížení nákladů vývoje a nasazení díky využití SOA infrastruktury a komponent v porovnání s jednotlivými projekty.

- **SOA MM úroveň 3 - Business Services a Collaborative Services.** Předností vrstvy je spojení podnikových procesů s IT. Tato úroveň je složena ze dvou částí. První jsou podnikové služby (Business Services), které se zaměřují na zlepšení interních podnikových procesů a druhou jsou spolupracující služby (Collaborative Services), jenž se zabývají zlepšením spolupráce s externími partnery.
- **SOA MM úroveň 4 – Měřené podnikové služby (Measured Business Services).** Úroveň zralosti 4 se zaměřuje na měření výkonnosti procesů zavedených na předešlé úrovni a jejich vlivu na podnikání. Součástí vrstvy je monitorování procesů, které umožňují uživatelům měnit způsob reakce na podnikové události.
- **SOA MM úroveň 5 – optimalizované podnikové služby (Optimized Business Services).** Úroveň zralosti 5 přidává automatickou reakci na měření zavedené na úrovni 4. Tím se SOA IS stává podnikovým nervovým systémem a reaguje automaticky na události objevující se na podnikové úrovni podle pravidel optimalizovaných pro plnění podnikových cílů.

Uvedená kapitola 2.2.3 vychází ze článku [3]. V následující podkapitole si ukážeme jaké přínosy a jaké problémy mohou nastat při zavedení SOA do podnikové infrastruktury.

2.2.4 Přínosy SOA

SOA umožňuje přebudovat IT infrastrukturu, odstranit redundance a zrychlit projekty a to znovupoužitím aplikačních služeb. IT technologie společnosti se může díky servisně orientované architektuře rychleji přizpůsobovat měnícím se podnikovým potřebám, rychleji a s menšími náklady realizovat nové IT projekty a omezovat výdaje na administrativu. Na základě existujících aplikací se vytvářejí služby (zákaznické, objednávkové, skladové apod.) odpovídající potřebám podniku. Poté je nad těmito službami vybudována v oddělené architektonické vrstvě aplikační logika odpovídající danému procesu. Takové sladění IT s podnikovými procesy pak přináší následující výhody [29]:

- Vytvořené služby se mohou použít v mnoha různých projektech. Omezuje se jejich redundance a v nových projektech není nutné přepracovávat funkcionalitu. Tak lze urychlit projekty a snižovat průběžné náklady.
- Oddělená aplikační logika se může snadno měnit, aktualizovat nebo dokonce zcela nahradit tak, aby odpovídala změněným potřebám podnikání. Umožňuje také vytvářet různé kombinace služeb bez toho, aby se měnily nebo přepisovaly. Výsledkem je větší akceschopnost podniku a snížení nákladů.
- SOA mění celou dynamiku IT, protože projekty, aplikace a celé IT týmy propojuje vzájemnými závislostmi a vazbami. Dřívější autonomie týmů omezila jejich externí interakce, zvýšila jejich odpovědnost a usnadnila správu rizik.

Nyní si ukažme přínosy SOA, kvůli nimž se v podnicích často využívá a to z hlediska provozní výkonnosti, efektivity podnikání a vývoje [29]:

- Omezení nákladů na infrastrukturu a správu.

- Lepší přehled, zabezpečení a kontrola celkových (end-to-end) podnikových procesů.
- Zjednodušení analýzy základních příčin problémů a automatizace určování priorit.
- Zrychlení doby nutné pro uvedení produktů a služeb realizovaných kritickými podnikovými aplikacemi a procesy na trh.
- Jednodušší dosažení souladu s politikami (bezpečnostními, businessovými či regulačními) ze všech funkčních oblastí.
- Udržování kontinuity podnikání v celém podniku včasnou reakcí na snížení bezpečnosti, výkonnosti nebo dostupnosti služeb.
- Sledování byznys metrik v reálném čase.
- Zvýšení opakované použitelnosti služeb a omezení nákladů na vývoj.
- Zkrácení doby potřebné pro vývoj nových služeb.
- Omezení nákladů na integraci a její složitosti využitím oborových standardů.
- Bezproblémovou spoluprací s klíčovými infrastrukturními řešeními třetích stran, jako jsou systémy pro správu identit, adresářové služby, systémy pro správu podniku, a využití jejich možností.

Z uvedených přínosů vyplívají zásadní výhody, nicméně tou hlavní výhodou SOA je, že změny podnikového IT se dají uskutečnit za kratší dobu a implementace první SOA aplikace bude obvykle trvat stejně dlouho nebo i déle, než kolik času by zabrala tvorba stejné aplikace v monolitickém návrhářském stylu. Avšak všechny následující SOA aplikace a jejich změny budou obvykle rychlejší a méně nákladné, protože se při nich může využít dříve vybudovaná SOA infrastruktura a služby.

SOA je spojována s technologií webových služeb a ty jsou pro ní jedním ze základních kamenů a hrají jasnou úlohu při realizaci SOA. V následující kapitole si podrobně popíšeme webové služby a standardy, které k nim neodmyslitelně patří [9].

2.3 Webové služby

Webové mohou být popsány jako sada programů spolupracujících po síti (nejčastěji po Internetu) a není vyžadována explicitně účast člověka během transakcí na rozdíl od současného webu, jenž je orientovaný téměř výhradně na uživatele. Podle standardizační organizace W3C je webová služba definována následovně [1]: „*softwarová aplikace, identifikovaná pomocí URI, jejíž rozhraní a vazby je možno definovat, popsat a prozkoumat pomocí nástrojů XML. Webová služba podporuje přímé interakce s jinými aplikacemi za použití zpráv, založených na XML a předávaných pomocí Internetových protokolů*.“ Webové služby jsou tedy systémem pro podporu součinné spolupráce počítačů v síti. Komunikace se odehrává mezi dvěma počítači, při níž má jeden funkci poskytovatele webové služby (provider) a druhý zastává funkci klienta [1]:

- **Služby** provádějí na požádání určitou akci (např. poskytují data z registru, přebírají zaslaná data a vkládají je do databází apod.). Charakteristickou vlastností služeb je naslouchání (zpravidla permanentní) požadavkům klientů a plnění jejich cílů. Každá služba musí buď veřejně, nebo alespoň pro svůj okruh klientů jasně deklarovat, jaká jsou pravidla pro využívání služby. Proto musí podporovat přístup k informacím o formátu předávaných zpráv, způsobu předávání zpráv a o pravidlech zpracování zpráv.
- **Klientské aplikace** jsou programy, žádající služby o akci (poskytnutí dat, vložení zaslaných dat apod.). Klientské aplikace pouze vznášejí požadavky na služby a nenaslouchají ostatním účastníkům (s výjimkou definovaných případů, např. při očekávání odpovědi služby či jiné klientské aplikace). Pokud chtějí využívat konkrétních služeb, musejí při komunikaci s nimi striktně dodržovat pravidla, stanovená službou.

Webové služby představují vhodné a univerzálně použitelné rozhraní pro zpřístupnění funkcionality a uložených dat, avšak WS může ve své výsledné formě nabývat různých podob, zejména v závislosti na použitých technologiích. Existuje však skupina pravidel, které z obecné služby tvoří právě službu webovou a měly by být pro všechny formy závazné. Obecně je WS prostředek pro přístup k aplikacím, který je [13]:

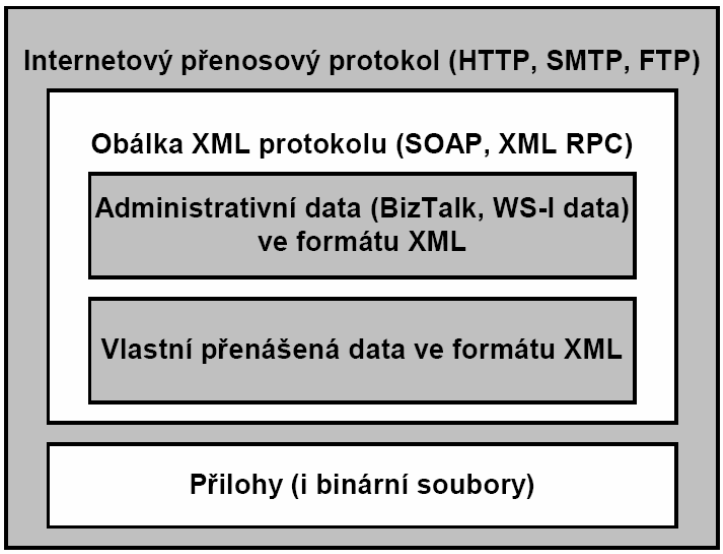
- Dostupný přes Internet nebo privátní intranet.
- Používá standardizovaný systém zpráv založený na XML.
- Nezávislý na platformě (operačním systému), programovacím jazyku, síťovém prostředí a na konkrétním softwarovém produktu (databázi, aplikaci).
- Rozhraní je popsáno strojově zpracovatelnou XML gramatikou.
- Existuje jednoduchý vyhledávací mechanismus pro nalezení služby.

Služba poskytuje přístup k datům a dotazující se klienti nemají žádné informace o vnitřní struktuře aplikace, která je ukryta za službou. Dále služby striktně oddělují komunikaci a technické stránky komunikujících aplikací, díky čemuž je vždy zajištěna kompatibilita různých informačních systémů. WS jsou tak především nástrojem pro zvýšení interoperability. Navíc využívají současné technologie a zahrnují silný aparát pro přesnou a standardizovanou definici služby.

Oproti velké výhodě WS v nezávislosti na platformě, dané aplikaci nebo síťovém rozhraní je velkou nevýhodou webových služeb jejich závislost na standardech, a to kvůli komunikaci uvnitř rozsáhlých aplikací i mezi nimi. Proto by měly být WS založeny na obecně přijímaných mezinárodních standardech, pocházejících z respektovaných standardizačních organizací a sdružení (ISO, IETF, W3C, OASIS) [1].

2.3.1 Technologie webových služeb

Webové služby založeny na komunikaci pomocí zpráv (XML protokoly), jenž jsou přenášeny některým z Internetových přenosových protokolů (HTTP, SMTP, FTP), jak můžeme vidět na Obrázku č.4. Stejně jako v předchozí kapitole zde uvedeme, že webové služby jsou technologicky nezávislé.



Obr. 4: Struktura zpráv webových služeb [1]

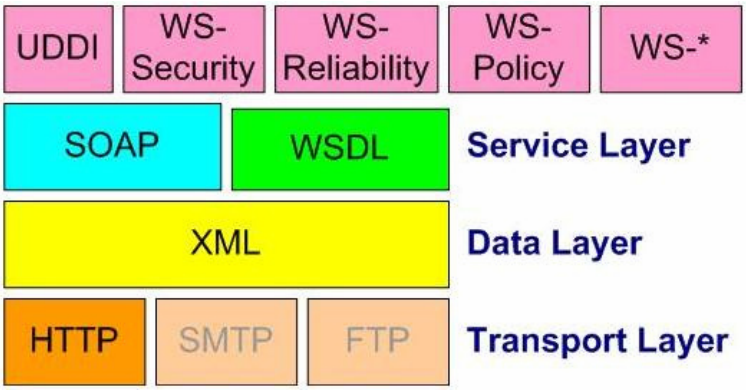
Filosofií webových služeb je v maximální možné míře využít stávající technologie. Jako standard pro výměnu XML zpráv je doporučen protokol SOAP, ale může jím být i jiný protokol. Součástí základního návrhu WS jsou technologie a specifikace komunikačních protokolů postavených na XML, které umožní [13]:

- Výměnu zpráv (zpravidla protokol SOAP).
- Popis webových služeb (zpravidla jazyk WSDL).
- Publikování a vyhledávání (zpravidla jazyk UDDI).

V následující kapitole si tyto uvedené protokoly a standardy webových služeb, ale i jiné probereme v následujících podkapitolách.

2.3.2 Protokoly webových služeb

Protokoly webových služeb jsou stále vyvíjeny a nyní se rozdělují do čtyř hlavních vrstev, jak můžeme vidět na Obrázku č.5. Jednotlivé vrstvy a protokoly budou předmětem následujících podkapitol.



Obr. 5: Vrstvy webových služeb [39]

2.3.2.1 HTTP

Zkratka pochází z anglického HyperText Transfer Protocol, což je Internetový protokol aplikační vrstvy určený pro výměnu hypertextových dokumentů ve formátu HTML a definuje tvar přenášených informací, možnosti a náležitosti požadavku i odpovědi. Z názvu vyplívá, že je určen pro přenos hypertextových dokumentů, ale má i velké využití pro přenos libovolných binárních dat (obrázky a soubory). Protokol pracuje na principu dotaz-odpověď (klient-server), kdy veškerou aktivitu musí být vyvolána klientskou stranou. Ta pošle serveru prostý text, který označuje určitý požadavek (požadovaný HTML dokument, binární data). Server poté předá data klientovi opět pomocí textu označující výsledek dotazu (typ požadovaného dokumentu, existenci požadovaných binárních dat) a za kterými následují data samotného požadovaného dokumentu. Poté již komunikace mezi klientem a serverem končí a spojení se uzavírá. Pokud požaduje klient data opakovaně, navazuje se nové spojení a data se posílají znovu. Protokol si předchozí informace o předchozích spojeních neukládá, proto je HTTP protokol bezstavový [19].

Protokol HTTP prošel během několika let své existence nezanedbatelným vývojem. Jeho první verze (označená 0.9) vznikla při vývoji systému WWW ve středisku CERN a byla velmi primitivní. Pro přenos hypertextových dokumentů byl požadován jednoduchý protokol, bez složitého spojování, autorizace a signalizace. Původní protokol klientovi umožňoval pouze navázat spojení se serverem, zaslat požadavek ve tvaru GET <URL>, přečíst odpověď a uzavřít spojení. Server pouze přijal požadavek, zpracoval jej a odeslal požadovaný dokument nebo zprávu o chybě [19].

Dnešním standardem je verze 1.0 (RFC 1945) a hlavní rozdíle oproti 0.9 je možnost přidáním k dotazu i odpovědi doplňující informace (typ dokumentu, dobu vzniku, schopnosti či požadavky klienta). Několikaleté používání této verze však odhalilo jisté slabiny a proto vývoj pokračoval verzí 1.1 (RFC 2616). Verze 1.1 není takovým krokem vpřed, jako svého času 1.0. Nicméně přináší několik užitečných rozšíření. Zaměřují se na zvýšení efektivity HTTP, lepší práci vyrovnávacích pamětí a serverů (proxy cache) a pro nás užitečnou zdokonalenou podporu neanglických jazyků [31].

HTTP definuje několik dotazovacích metod, které se mají provést nad uvedeným objektem/dokumentem [5]:

- **GET** - Požadavek na poslání dokumentu určeného URL zaslaný společně s dalšími případnými daty (proměnné prohlížeče, session id, …). Výchozí metoda při požadavku na zobrazení hypertextových stránek, RSS feedů aj. Jedná se o nejpoužívanější metodu.
- **HEAD** - To samé jako metoda GET, ale server už nemusí předávat tělo odpovědi. Poskytne pouze metadata o požadovaném cíli (velikost, typ, datum změny, atd.) a dále se používá k testování hypertextových linek, jejich dostupnosti a poslední modifikaci.

- **POST** - Odesílá uživatelská data na server. Používá se například při odesílání formuláře na webu. S předaným objektem se pak zachází podobně jako při metodě GET. Data může odesílat i metoda GET, ale metoda POST se používá pro příliš velká data (víc než 512 bajtů, což je velikost požadavku GET) nebo pokud není vhodné přenášená data zobrazit jako součást URL (data předávaná metodou POST jsou obsažena v HTTP požadavku).
- **PUT -** Požadavek na uložení posílaných dat na server, která poté budou dostupná pomocí GET. Používá se velmi zřídka, pro nahrávání dat na server se běžně používá FTP nebo SCP/SSH.
- **DELETE** - Smaže uvedený objekt (uveden v URL) ze serveru. Jsou na to potřeba jistá oprávnění stejně jako u metody PUT.
- **TRACE** - Používá se k testování originálního serveru, ten má vrátit klientovi kladnou odpověd bez dat.
- **OPTIONS** - Dotaz na server, jaké má podporuje možnosti komunikace.
- **CONNECT** - Spojí se s uvedeným objektem přes uvedený port. Používá se při průchodu skrze proxy pro ustanovení kanálu SSL.

Dnes se již v mnoha případech pro transport dat po Internetu pro větší bezpečnost před odposloucháváním (viz kapitola X.X) či potvrzením dat používá zabezpečená verze HTTPS. Nejedná se o jiný protokol pouze přenášená data pomocí http jsou šifrována pomocí SSL nebo TLS, což zaručuje ochranu proti odposlechnutí dat nebo man-in-the-middle útokům [10]. HTTPS implicitně komunikuje prostřednictvím TCP portu 443 (u HTTP je to port 80). Pro komunikaci pomocí HTTPS musí nejdříve server vlastnit certifikát, který musí být podepsán certifikační autoritou (CA). Ta potom zaručuje, že vlastník certifikátu se nevydává za nikoho jiného. Webové prohlížeče jsou většinou vybaveny podpisovými certifikáty největších podpisových autorit. Organizace si mohou certifikáty podepsat samy a nemusí platit CA za podpis certifikátu nemalé peníze. Ovšem poté webové prohlížeče při načítání stránek varují uživatele před nedůvěryhodným zdrojem. Úroveň zabezpečení závisí na správnosti implementace v prohlížeči, na serveru a na použitém šifrovacím algoritmu [43].

2.3.2.2 XML

Co se týče formátu přenosu dat, je zřejmé, že vedoucí postavení si postupně vydobyl formát XML (eXtensible Markup Language - rozšiřitelný značkovací jazyk) jako formát nezávislý na platformě, síťovém prostředí a programovém vybavení. XML je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Původní jazyk pro publikování HTML již přestal vyhovovat především pro svou složitost, která vznikla jeho postupným (a svévolným) rozšiřováním. Jazyk XML nemá žádné předdefinované značky (tagy, názvy jednotlivých elementů) a také jeho syntaxe je podstatně přísnější, než HTML. Jazyk XML je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Jazyk umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, nezabývá se sám o sobě vzhledem dokumentu nebo jeho částí. Prezentace dokumentu (vzhled) se potom definuje připojeným stylem. Další možností je pomocí různých stylů provést transformaci do jiného typu dokumentu, nebo do jiné struktury XML [46].

První verze doporučení XML byla zveřejněna v únoru 1998. V říjnu roku 2000 byla zveřejněna revize tohoto doporučení, kterou známe jako XML 1.0 (Dnes již existuje čtvrtá revize). Doporučení definuje, co je to XML dokument, co je prvek (element), jeho počáteční a koncové ohraničení, značka, atributy i obsah prvku. Určuje pravidla pro volbu názvů prvků (značek a atributů). Stanoví správné formátování dokumentu a také, kdy je dokument platný (validní). Aktuální verze XML je 1.1

(od 16. srpna 2006). Obě verze se liší v požadavcích na použité znaky v názvech elementů, atributů atd. Verze 1.0 dovolovala pouze užívání znaků platných ve verzi Unicode 2.0, která obsahuje většinu světových písem, ale neobsahuje později přidané sady (Mongolština). Verze XML 1.1 zakazuje pouze řídící znaky, což znamená, že mohou být použity jakékoli jiné znaky. Je třeba poznamenat, že omezení ve verzi 1.0 se vztahuje pouze na názvy elementů a atributů. Jinak obě verze dovolují v obsahu dokumentu jakékoli znaky. Verze 1.1 je tedy nutná, pokud potřebujeme psát názvy elementů v jazyku, který byl přidán do Unicode později. Další změna ve verzi 1.1 je, že řídící znaky se mohou vkládat jen jako escape sekvence (ESC se používá pro rozšíření ASCII kódu pro různé účely), a se speciálními znaky (form-feed) se musí zacházet jako s bílými znaky [46].

XML neobsahuje předdefinované tagy, je třeba definovat vlastní značky, které budeme používat. Tyto značky je možné (nepovinně) definovat v souboru DTD (Document Type Definition). Potom je možné automaticky kontrolovat, zda vytvářený XML dokument odpovídá této definici. Program, který tyto kontroly provádí, se nazývá parser. Při vývoji aplikací můžeme parser použít, a ten za nás detekuje většinu chyb v datech. DTD není jediný definiční jazyk pro XML. Neobsahuje možnost kontrolovat typy dat (čísla, měnové údaje, údaje o datu a čase). To je vlastnost, která chybí při zpracování dat databázového charakteru. Pro různé standardní aplikace byla postupně vytvořena schémata, která definují značky (názvy elementů) pro konkrétní typy dokumentů. Příkladem může být DocBook, který definuje struktury pro vytváření knih, článků, vědeckých publikací a podobně. Výhodou takových aplikací je, že současně s definičními soubory DTD je dodávána sada stylů (XSL souborů) pro následné zpracování a přípravu pro tisk, nebo pro převod do jiných standardních tvarů (PostScript, HTML atd.). Další vlastností XML je, že v jednom dokumentu můžeme používat najednou nezávisle na sobě několik druhů značkovaní pomocí jmenných prostorů (namespaces). To umožňuje kombinovat v jednom dokumentu několik různých definic ve formě DTD nebo schémat bez konfliktů v pojmenování elementů [7].

Přijetí doporučení W3C o XML spustilo lavinu dalších aktivit. Firma Microsoft stála při zrodu XML a vynaložila velké úsilí pro rozšíření technologie XML tak, aby byla využitelná při tvorbě webových služeb.

2.3.2.3 SOAP

V této podkapitole si popíšeme protokol SOAP, jenž je považován za základní pilíř webových služeb, a převážně čerpá informace z literatury [13].

SOAP se ve webových službách používá místo vzdáleného volání procedur (RPC). Jedna aplikace pošle v XML zprávě požadavek druhé aplikaci, ta požadavek obslouží a výsledek dotazu předá opět jako XML zprávu dotazující se aplikaci. V tomto případě bývá webová služba vyvolána webovým serverem, který čeká na požadavky klientů a v okamžiku, kdy přes HTTP přijde soapová zpráva, spustí webovou službu a předá jí požadavek. Výsledek služby je pak předán zpět klientovi jako odpověď [15].

Definice protokolu současné verze 1.2 neurčuje, čeho je SOAP zkratka. V prvním případě vycházíme, že jeho primárním účelem je vzdálené volání procedur, pak je SOAP všeobecně vnímanou zkratkou pro Simple Object Access Protocol. Druhý výklad zkratky SOAP je založen na faktu, že je prezentován jako protokol servisně orientované architektury a v této souvislosti je zkratka SOAP vykládána jako Service Oriented Architecture Protocol.

SOAP obsahuje nástroje pro vlastní popis obsahu zprávy, způsobu jejího zpracování, množinu pravidel pro vyjádření vlastních datových typů a konvence pro vlastní reprezentaci vzdáleného volání procedur (RPC – Remote Procedure Call). Jedna aplikace pošle v XML zprávě požadavek druhé aplikaci, ta požadavek obslouží a výsledek zašle jako druhou zprávu zpět původnímu iniciátorovi komunikace. V tomto případě bývá webová služba vyvolána webovým serverem, který čeká

na požadavky klientů a v okamžiku, kdy přes HTTP přijde SOAP zpráva, spustí webovou službu a předá jí požadavek. Výsledek služby je pak předán zpět klientovi jako odpověď. SOAP je nástupce XML-RPC, ačkoliv si zapůjčuje jeho způsob přenosu dat a další vlastnosti.

SOAP byl původně navržen v roce 1998 jako XML-RPC (jednoduché a méně flexibilní) za podpory firmy Microsoft. První verze (1.0) protokolu SOAP vznikla na konci roku 1999 jako výsledek společné práce firem DevelopMentor, Microsoft a UserLand, které chtěly vytvořit protokol pro vzdálené volání procedur založený na XML. V průběhu roku 2000 se k podpoře přihlásila i firma IBM a nová verze SOAPu 1.1 byla zaslána W3C konsorciu. SOAP je dnes ve verzi 1.2, která je nejpoužívanější. Součástí standardu SOAP 1.2 je jeho vazba na protokol HTTP.

SOAP se ve webových službách dá použít dvěma způsoby a podle toho se dělí i označení webových služeb:

- **SOAP RPC Web services** – webové služby využívající SOAP v souladu se svým původním účelem – tzn. využívají zabudovaných funkcí SOAPu k volání vzdálených procedur a funkcí. Klient volá webovou službu na základě specifikace konkrétní operace a množiny parametrů. V odpovědi se vrací výsledné hodnoty. Definice operací, požadovaných parametrů a návratových hodnot je specifikována ve WSDL dokumentu – proto dokumentace RPC služby musí být ve formátu WSDL.
- **SOAP Web services** – SOAP je zde použit pouze jako „obálka“ pro přenos libovolného obsahu uvnitř SOAP zprávy. Tento způsob aplikace SOAP se někdy označuje jako „document-oriented“ nebo „document-style“. Narozdíl od vzdáleného volání funkcí, posílá klient službě jako parametr obecný XML blok, jehož definice může být součástí WSDL dokumentu. Výhody protokolu SOAP oproti přímému posílání XML bloků přes HTTP metodou POST/GET jsou především formalizace datových typů, zaručení jednoznačnosti XML elementů a možnost formálního zápisu operací ve WSDL. Další z výhod je možnost použití některého předefinovaného vzoru pro výměnu zpráv (MEP), začínající jednosměrnou komunikací a končící sofistikovaným systémem pro výměnu sekvencí zpráv. Dokumentace služby musí být ve formátu WSDL.

2.3.2.4 WSDL

Web Services Description Language (jazyk popisu Webových služeb) označuje, co nabízí webová služba za funkce a způsob, jak se jí na to zeptat. Soubor WSDL je dokument XML popisující sadu zpráv SOAP a způsob, jakým se zprávy vyměňují. Jelikož WSDL má formát XML, je čitelný a snadno upravovatelný, ale většinou je generován a využíván softwarem. Klientský program připojující se k webové službě umí číst WSDL, aby zjistil dostupné funkce na serveru. Dále jsou ve WSDL souboru uložené použité speciální datové typy. Podporované operace a zprávy jsou popsány abstraktně a potom se omezují na konkrétní síťový protokol a formát zprávy. WSDL poskytuje dokumentaci pro distribuované systémy a služby jako návod pro automatizaci detailů obsažených komunikačních aplikacích. Je určený pro popis webových služeb a přístupu k nim. Je třeba poznamenat, že WSDL není novým typem definičního jazyka [23].

WSDL dokument definuje služby jako kolekci koncových bodů v síti nebo protokolů. Abstraktní definice koncového bodu a zpráv je oddělena od jejich konkrétní podoby nebo datových vazeb. To umožňuje opětovné použití abstraktních popisů. Port je definován přiřazením síťové adresy s opětovně použitelnou vazbou. Kolekce portů definuje službu. WSDL dokument tedy používá následujících elementů pro popis síťových služeb [22]:

- Types - container pro definici typu dat používající nějaký typový systém (např. XSD).
- Message - abstraktní typová definice předávané zprávy.
- Operation - abstraktní popis akce podporované službou.
- Port Type - abstraktní sada operací podporovaných jedním nebo více uzly.
- Binding - konkrétní protokol a specifikace formátu dat pro běžný typ portu.
- Port - uzel definovaný jako kombinace binding a síťové adresy.
- Service - kolekce souvisejících uzlů.

„Mnoho současných nástrojových kitů SOAP obsahuje nástroje ke generování souborů WSDL z existujících programovacích rozhraní. Existuje i několik nástrojů k přímému psaní WSDL, přesto není nástrojová podpora WSDL tak kompletní, jak by mohla být. Nemělo by trvat dlouho, aby nástroje k tvorbě souborů WSDL a následnému generování proxy objektů se staly součástí většiny implementací SOAP. WSDL se tak stane preferovaným způsobem tvorby rozhraní SOAP pro Webové služby XML.“ [5]

2.3.2.5 UDDI

Technologie Universal Discovery Description and Integration (UDDI) slouží pro vyhledávání Webových služeb a díky ní můžeme získat informace o nabízených službách nebo vyhledat konkrétní službu od různých poskytovatelů. Pro nabízení vlastních WS musíme být registrování v UDDI, pokud tomu tak není je malá pravděpodobnost, že se k naší nabízené službě dostane větší množství zákazníků. Nabízené služby organizace v registru UDDI popisuje soubor XML. Ten se dělí na tři části. První slouží pro popis samotné organizace (název, adresa,…). Druhá část zahrnuje průmyslové kategorie založené na standardech a poslední část popisuje rozhraní ke službě [35]. Způsob, jakým jsou služby definovány je uveden v dokumentu UDDI nazvaném Type Model nebo tModel. V mnoha případech obsahuje tModel soubor WSDL, který popisuje rozhraní SOAP k Webové službě XML, ale tModel je dostatečně pružný, aby mohl popsat téměř jakýkoli druh služby [5].

Adresář UDDI rovněž obsahuje několik způsobů, jak vyhledávat služby potřebné ke stavbě našich aplikací. Můžeme například hledat poskytovatele služby v určeném geografickém místě nebo podnik určitého typu. Adresář UDDI pak dodá informace, kontakty, odkazy a technická data, podle kterých vyhodnotíme, které služby splňují vaše požadavky. UDDI umožňuje vyhledat podniky, od kterých můžeme získat určité Webové služby. Specifikace WS-Inspection umožňuje procházet množinu Webových služeb XML nabízených na určitém serveru. Zde můžete zjistit, která služba splňuje vaše požadavky. Technologie UDDI (Universal Description, Discovery and Integration) je po WSDL druhou fundamentální součástí webových služeb. Hlavním účelem této technologie je, aby byla snadno k nalezení. Jedná se o soubor technologií pro budování distribuovaného adresáře webových služeb, jak na veřejné úrovni, tak i pro vnitřní účely organizací. Technologii UDDI v současnosti tvoří [5]:

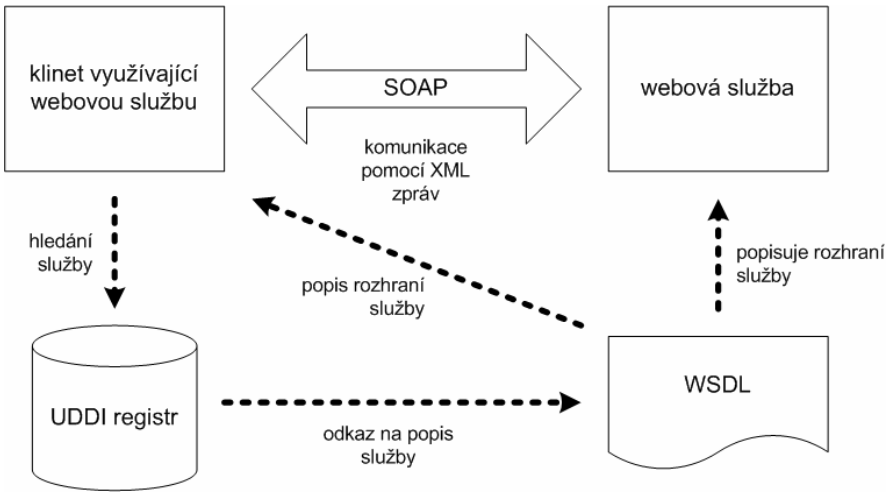
- **Specifikace UDDI** - definuje XML formát pro uložení dat a metadat o službách a společnostech (hierarchická XML struktura uložení metadat je předmětem obr. 6) a definuje API rozhraní pro manipulaci s nimi. Dále zahrnuje aparát pro replikaci veřejných adresářů a s příchodem verze UDDI 3.0 i komplexní pravidla pro komunikaci s privátními UDDI registry.

- **Veřejné UDDI adresáře –** tzv. UBR (UDDI Business Registry) – plně funkční veřejně přístupné implementace specifikace UDDI (v současnosti verze 2.0).

2.3.2.6 Vztah tří základních technologií webových služeb

V této kapitole se zaměříme jak spolupracují uvedené technologie webových služeb budeme vycházet z literatury [15]. Jak jsme již uvedli, webové služby umožňují jednoduchou komunikaci mezi aplikacemi ve velmi heterogenním prostředí, protože komunikace je založena na platformě nezávislých standardech. Aplikace si mezi sebou posílají XML zprávy, které přenášejí dotazy a odpovědi jednotlivých aplikací. Celá infrastruktura webových služeb je založena na třech základních technologiích, které jsme si uvedli v předešlých podkapitolách.

Vzájemné vztahy mezi těmito třemi technologiemi jsou zachycené na Obrázku č. 6. Ke každé webové službě by měl být k dispozici její formální popis v jazyce WSDL. Z tohoto popisu již jde automaticky vygenerovat požadavek SOAP. Ve větších systémech nebo přímo v Internetu se popis služby může zaregistrovat do UDDI registru, jenž slouží jako seznam, který umožňuje vyhledávání služeb s určitými parametry. Klient, který chce využít WS, ji získá přes UDDI nebo přímo její popis. Z něj je jasné, jakou strukturu má mít zpráva SOAP a kam se má webové službě poslat, aby ji rozpoznala.



Obr. 6: Vztah základních technologií webových služeb [15]

Ostatní standardy jako WSDL a UDDI vznikly až později po uvedení SOAPu a jen dále rozšiřují jeho možnosti a snadnost použití. SOAP umožňuje zaslání XML zprávy mezi dvěma aplikacemi a pracuje tedy na principu peer-to-peer. Zpráva je jednosměrný přenos informace od odesílatele k příjemci, ale díky kombinování několika zpráv můžeme pomocí SOAPu snadno implementovat běžné komunikační scénáře.

2.3.3 Zhodnocení webových služeb

V této kapitole zhodnotíme technologie webových služeb. Podíváme se na pozitiva a negativa webových služeb, které mohou přijetí webových služeb zpomalit. Celá kapitola je přebrána z literatury [1].

2.3.3.1 Pozitiva webových služeb

Jak již jsme uvedli v úvodu práce, je bezesporu největší výhodou webových služeb univerzálnost a nezávislost na informačních technologiích, která umožňuje komunikaci mezi různými aplikacemi. Ty pak mohou být napsaný v jiném programovacím jazyce, běžet pod jinými operačními systémy i na rozdílných fyzických počítačích. Lze pak např. dosáhnout toho, že spolu komunikují aplikace napsaná v Javě provozovaná na Linuxu s aplikací napsanou v .NET v OS Windows. S implementací WS nedochází ke znehodnocení, již vytvořených softwarových komponent (IS, ERP, CRM), neboť je pouze obalují.

Již víme, že webové služby vychází ze vzdáleného volání procedur (RPC-XML). Tyto procedury předali cílové aplikaci pouze metody objektů a docházelo tak k těsné vazbě mezi aplikacemi. Tento postup byl později nahrazen metodou předávání XML dokumentů, což umožnilo daleko širší použití. Jedním z nich je, že XML zprávy jsou ve formě textu a odpadají tak problémy spojené s binárními protokoly.

Za významné pozitivum považujeme dobře definovanou sémantiku dat, která se přenášejí mezi více komunikujícími stranami. Ta má za následek hladký tok informací a menší chybovost, způsobenou špatnou interpretací dat.

2.3.3.2 Negativa webových služeb

Stejně jako v kapitole 2.1.2 o technologii ASP musíme zde uvést, že výhody webových služeb jsou i zároveň nevýhodami. Jelikož jsou webové služby založeny na XML, tedy forma prostého textu, jednoduše procházejí firewally a to má za následek bezpečnostní riziko a je pouze otázkou času, kdy bude proveden úspěšný útok pomocí WS. S formátem XML souvisí i další negativum, což je mnohonásobné zvětšení značkovaných dat oproti původním a tím pádem velké nároky na systémové zdroje při analyzování přenášených dat. Další bezpečnostní riziko je při deklaraci struktur volně vázaných zpráv, kdy můžeme jako datový element předávat SQL dotaz pro koncovou databázi, který může mít destruktivní účinky.

Významným záporem WS je neexistence stabilních standardů pro bezpečnou komunikaci a volně vázané transakce (rollback, pravidla pro doručení). Významným omezením je podmíněná shoda komunikujících stran na sémantice a syntaxi zpráv. Sémantická standardizace je velmi obtížná a je to nejslabší místo webových služeb.

Dalším diskutabilním problémem WS je jejich spolehlivost. Přidáme-li do svého systému cizí webovou službu, musíme mít zaručenou její dostupnost. Postupem času by měla mít služba svůj životní cyklus pro určení vhodnosti jejího použití a způsobu implementace.

3 Analýza a specifikace požadavků

V kapitole si nejprve řekneme, jak probíhala specifikace požadavků pro vývoj editačního systému N.E.S.P.I., jako systému založeného na technologii ASP. Budeme se zabývat bezpečností systému, dále pak uživateli, jenž budou systém využívat a také si uvedeme diagram případů použití, diagram tříd a samozřejmě E-R diagramu, jenž patří datovému modelování. Analýza a specifikace požadavků pro vývoj software je důležitou fází v životním cyklu jeho vývoje. Bylo jí tudíž určeno velké pozornosti, neboť neúspěšné transformování stávajícího systému s vynaložením finančních prostředků a nemalých časových zdrojů, by mělo negativní vliv na současné i nové zákazníky, neboť ještě před dokončením samotného systému bylo vynaloženo velké úsilí na jeho propagaci v rámci marketingu firmy před konkurencí.

3.1 Sběr informací

Největší sběr informací o vyvíjeném systému proběhlo dne 6. října 2008 v zázemí firmy WebRex s.r.o. s majitelem a zadavatelem Dominikem Debefem a hlavním programátorem firmy Ivo Válalem, který vedl po celou implementace odborný dohled nad projektem. V pravidelných dvoutýdenních intervalech pak probíhaly porady nad aktuálním stavem projektu a byly doplňovány různé drobnosti. Lze tedy říci, že systém byl, je a bude dále vyvíjen dle životního cyklu softwaru jako model s přírustky.

3.2 Bezpečnost systému

Nejvíce diskutovanou otázkou plánovaného systému je jeho bezpečnost, neboť na ní stojí a padá působení celé firmy. Při nabourání nepovolané osoby do systému by mohlo dojít ke ztrátě dat nebo dojít k uveřejnění nevhodných a kompromitujících informací. Webové stránky všech zákazníků jsou samozřejmě pravidelně zálohovány. Ovšem vzhledem k tomu, že firma se stará o cca. 400 zákazníků, trvalo by obnovení dat delší dobu, což je z hlediska zákazníků nemyslitelné a vedlo by ke ztrátě jejich důvěry a pravděpodobně i k nalezení zázemí u konkurence. V následujících podkapitolách si ukážeme typy útoků, proti kterým by měl být systém odolný.

3.2.1 XSS – Cross-site scripting

První formou útoku je cross-site scripting nebo-li skriptování napříč servery, což je metoda narušení WWW stránek využitím bezpečnostních chyb ve skriptech. Útok je založen na injektování kódu na vstup webové aplikace. Útočník díky těmto chybám v zabezpečení webové aplikace dokáže do stránek podstrčit svůj vlastní kód (javascript, html, php atd.). Ten může využít k poškození vzhledu stránky, její nefunkčnosti, získání citlivých údajů návštěvníků stránek, obcházení bezpečnostních prvků aplikace a phishingu. Existují 3 typy útoků [38]:

- **Lokální (DOM based)** - lze využít i na statických stránkách a jde o neošetřené přenesení proměnné z URL adresy do javascriptu.
- **Neperzistentní** - je postaven na úpravě části URL, která se interpretuje do stránky jako její součást. Pokud do URL přidáme svůj kód, který není před interpretací upraven, zachová se kód

na stránce v prohlížeči, jako její normální součást. Tato zranitelnost se týká především stránek s generovaným obsahem. Je to problematikou URL adres (editují se hodnoty proměnných), vyhledávacích formůlářů, radiobuttonům, inputů atd. (zadaný řetězec se neukládá a pouze se pošle na výstup bez ošetření).

- **Perzistentní** - jde o nejnebezpečnější možnost, protože na takto napadené stránky nemusíme vstoupit přes upravený link. Vzniká, pokud je obsah stránky generován z databáze. Náš javascript jednoduše vložíme třeba jako součást komentáře. Spolu s ním se uloží do databáze a je následně zobrazen všem lidem, kteří si takovýto komentář zobrazí. Útok je problematikou diskusních fór, knih návštěv a podobných aplikací, které uchovávají data zadaná na vstup třeba v databázi nebo v souboru. Tím je zaručeno, že dojde k opětovnému spuštění skriptu po každém načtení stránky do prohlížeče.

3.2.2 SQL Injection

Další bezpečnostní slabina napadení webových stránek je metoda SQL injection. Ta spočívá v napadení databázové vrstvy programu vsunutím (injection) kódu přes neošetřený vstup a vykonání SQL dotazu. Pokud útočník zná strukturu tabulky (nebo celé databáze), které svými proměnnými ovlivní, má vše daleko jednodušší, než když musí odhadovat, jaké sloupce jsou použity, jaké mají datové typy a jak se jmenují. V nejběžnějším případě je útok na internetové stránky prováděn přes neošetřený formulář, manipulací s URL nebo třeba i podstrčením zákeřně upravené cookie. Na Internetu je stále velké množství webů, spravovaných převážně nezkušenými programátory, kteří o této technice útoku neví a tuto kritickou chybu opomíjejí. Útočníci můžou udělat následující nežádoucí úkony [6]:

- Získat přístup k citlivým datům (například uživatelská hesla, skryté emaily atd.).
- Získat přístup k jakémukoliv, například administrátorskému účtu na webu.
- Získat přístup ke všem účtům naráz.
- Smazat všechna data co máme v tabulkách (proto je doporučeno pravidelně zálohovat obsah databází).

Toto nechtěné chování vzniká při propojení aplikační vrstvy s databázovou vrstvou a zabraňuje se mu pomocí jednoduchého escapování potencionálně nebezpečných znaků.

3.2.3 Replay

Tento útok spočívá v odposlouchávání části komunikace mezi dvěma autentizujícími se stranami a následném použití odchycených dat k autentizaci útočníka. Bránit se útokům poslouchání dat na síti lze jednoduše pomocí techniky výzva-odpověď nebo použitím časové pečetě. Pro odposlouchávání posílaných dat v síti se používají analyzátory paketů tzv. „*sniffery*“. To jsou zařízení pracující v pasivním režimu (síťový adaptér je v promiskuitním režimu), které zachycují síťové pakety a původně sloužili k analyzování provozu v síti. Analyzátory paketů se mohou lišit různými funkcemi a také mohou sledovat v síti pouze provoz jednotlivých protokolů. Většina dnešních snifferů analyzuje minimálně protokoly Ethernet, TCP/IP a NetBIOS. Ačkoliv se jedná o nejznámější útok, je nutno podotknout, že obranu proti němu není zas jednoduché implementovat v aplikaci [17].

3.3 Neformální specifikace

Budoucí systém musí být uživatelsky příjemný a jeho rozhraní musí být přehledné a snadno ovladatelné. Přístup k jednotlivým informacím musí být co nejintuitivnější. Podmínkou zadavatele je, že vyvíjený systém musí zachovávat současné rozložení prvků systému a jeho jednoduché ovládání, neboť jsou na něj zákazníci zvyklí a na něm stojí jeho úspěšnost a oblíbenost. Jiné složité ovládání by mohlo zákazníky odradit, ačkoliv určité změny jsou již nyní ve fázi specifikace nevyhnutelné. Ty by se ovšem měly běžného zákazníka dotknout minimálním způsobem a spíše mu umožnit efektivnější prácí se systémem. Pouze necelé jedno procento z nich má ke stávajícímu ovládání výhrady a chtělo by prvky uspořádat jinak. S tímto souvisí i další požadavek, že editační systém bude napodobovat vzhled jednotlivých stránek zákazníků jako je tomu dosud. Systém se tedy bude stylovat dle webových stránek zákazníka, aby byla zajištěna jejich spokojenost a docílilo se efektu u zákazníků, že pracují skutečně se svými webovými stránkami.

Systém bude prozatím vyvíjen v českém jazyce, ale systém počítá s přidáním dalších jazykových verzích, které pro své webové stránky zákazníci vyžadují. Systém bude od počátku s touto variantou počítat a bude snadné upravovat a přepínat jednotlivé verze systému. Samotný systém má umožnit i spravování jazykových verzí webů samotných zákazníků a docílit tím lepší efektivity v úspoře času, neboť stávající systém musel být na jiné verze předěláván ručně a to ještě velmi nevhodným způsobem.

Systém by měl obsahovat funkci testování, která zjistí správný provoz určených částí systému pro webové stránky zákazníků. Testování bude automatické, které se může použít při spouštění nového webu a snadno tak zjistit, zda je všechno v pořádku. Nebo může být testování specifikováno pouze na určité části systému. Nedílnou součástí systému bude logování uživatelských akcí pro lepší přehled o činnostech uživatelů v systému. Do systému musí dále být zakomponována funkce snadného použití optimalizace pro webové vyhledávače (SEO - Search Engine Optimization) [45].

Nová verze editačního systému by měla počítat i se zabudováním Internetového obchodu (e-shopu), popřípadě intranetu pro určité zákazníky. Po té bude zákazníkům umožněno spravovat své webové stránky, e-shopy a intranety na jednom místě. Produkty si zákazníci budou vkládat různými cestami např.: importem XML nebo rovnou z vlastních IS (Money, Navision, Hélios, Pohoda, atd.). V první fázi to bude zajištění komunikace přes XML a SOAP. Další IS budou následovat, neboť to bude vyžadovat dostatek času pro jejich zavedení.

Na vývoj systému by měly být použity nejmodernější technologie. Jelikož se jedná o webovou aplikaci, bude nutností použít MySQL (>5.x) s kombinací volně šiřitelného skriptovacího programovacího jazyka PHP (>5.x) anebo interpretovaného skriptovacího jazyku RUBY, který je zároveň objektově orientovaný. Vize do budoucna je, že by systém pracoval nejen s databázovým systémem MySQL, který tvoří naprostou většinu webových stránek, ale i s jinými databázovými systémy jako je PostgreSQL, MSSQL atd. Další moderní technologií, která by měla zefektivnit a přinést užitek je standard XHTML 1.1 a systém musí mít ve všech svých částech validní kód a aplikace musí být podporována všemi majoritními webovými prohlížeči.

3.4 Uživatelé systému

Systém bude obsluhovat dva typy uživatelů, samozřejmostí je zajistit bezpečné přihlášení a dále možnost měnit přihlašovací údaje. Prvním typem uživatelů jsou zákazníci, kteří si pomocí systému budou editovat svůj web, e-shop a nebo intranet (dále označované e/i), a to v jejich libovolné kombinaci, již zmíněným způsobem v předešlé kapitole. Zákazníci se budou dělit podle rolí

na administrátory, kteří budou mít plnou moc systémem, a obyčejné uživatele, kteří budou mít omezené pole působnosti podle toho, co jim administrátor povolí. Tím máme na mysli, že uživatel bude moci spravovat a editovat tu část webu (e/i), kterou dostane na starosti, tzn. určité části webu, jako jsou novinky, akce a nebo celou jazykovou verzi webu.

Druhými uživateli systému budou zaměstnanci firmy WebRex [44], kteří se dále budou dělit na programátory a manažery. Ti budou mít stejně jako administrátoři plnou moc nad webem (e/i), ovšem s tím rozdílem, že mohou administrovat jakéhokoliv zákazníka. Dále budou moci měnit a editovat různá nastavení pro weby (e/i), přidávat a upravovat jazykové verze, šablony, přidávat novinky o systému N.E.S.P.I. a další specifické operace. Manažeři dále budou mít navíc v systému možnosti, které jim jako manažerům náleží např.: vystavování faktur zákazníkům a do budoucna jiné marketingové záležitosti.

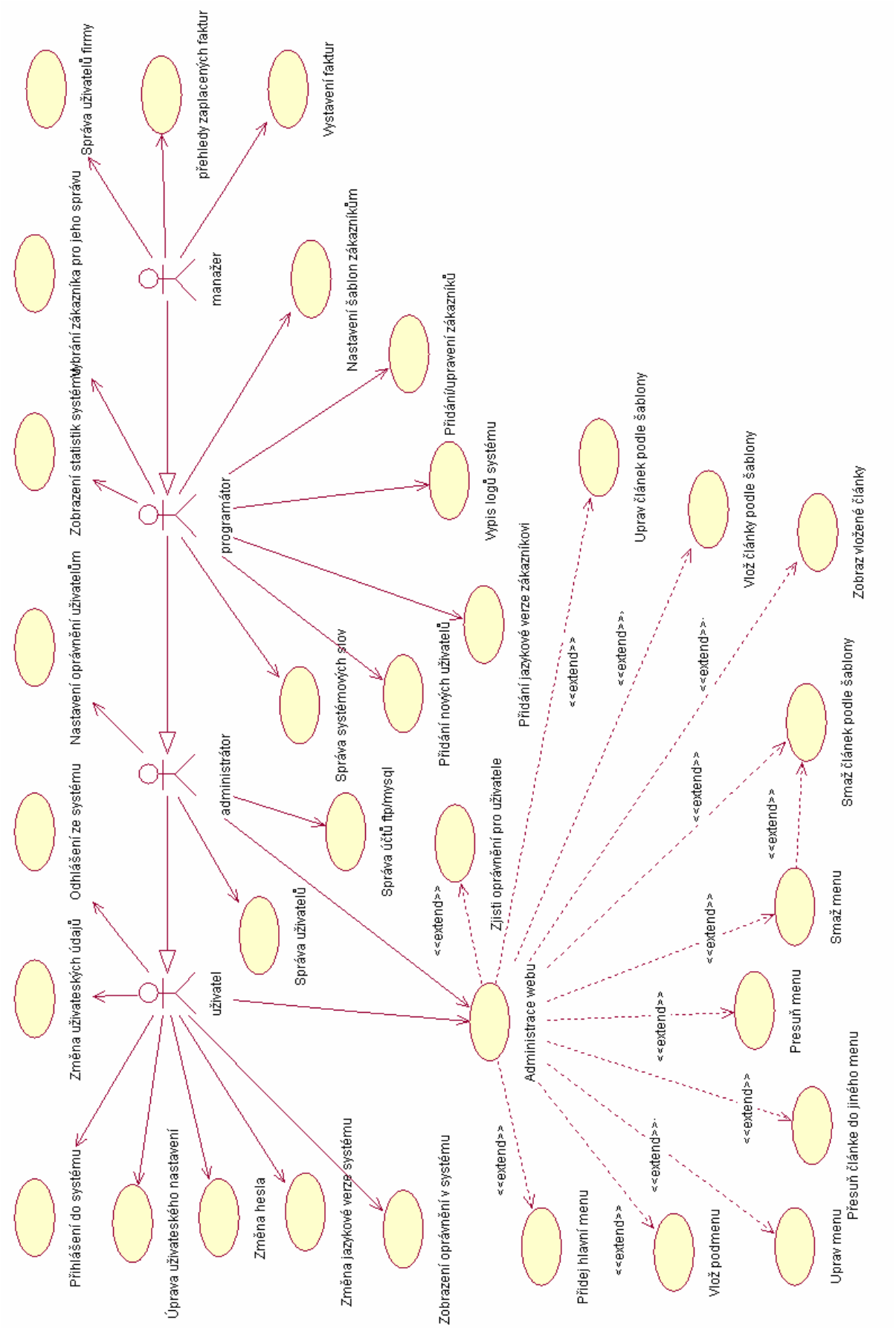
Pro lepší pochopení rolí a jejich funkcí v systému poslouží následující model případů užití.

3.5 Model případů užití

Modely případů užití slouží k nalezení hranic systému. Jsou psány z pohledu zákazníka a podávají první představu o rozsahu projektu. Slouží k lepší komunikaci se zákazníkem a později jsou použity při implementaci. Při řešení téměř všech projektů je prvním krokem vytvoření takovýchto modelů. Nezabýváme se zde technologickými problémy a snažíme se navrhnout funkční podobu systému co nejsrozumitelněji pro zákazníka. Zjišťujeme, které procesy má systém podporovat a jací uživatelé ho budou používat.

K modelování případů užití (use case diagramů) bylo využito jazyka UML (Unified Modeling Language), který je jednotný modelovací jazyk s bohatou sémantikou a syntaxí, který usnadňuje návrh a vizualizaci různých typů aplikací. Je obecným standardem pro specifikaci, vizualizaci, tvorbu a dokumentování jednotlivých objektů softwarových systémů. Ve svém názvu má slovo unifikovaný, protože jedním z jeho cílů je sjednocení používaných výrazových prostředků. Při vývoji softwaru zjednodušuje komplexní proces návrhu, tvorby a popisu objektů výsledného programu. Při návrzích databází umožňuje návrhářům tento jednotný jazyk komunikovat společně s tvůrci aplikací a uživateli. UML podporuje objektově orientovaný přístup k analýze, návrhu a popisu programových systémů. UML neobsahuje způsob, jak se má používat, ani neobsahuje metodiku(y), jak analyzovat, specifikovat či navrhovat programové systémy.

Následující Obrázek č. 7 zachycuje diagram případů použití pro vytvářený editační systém.

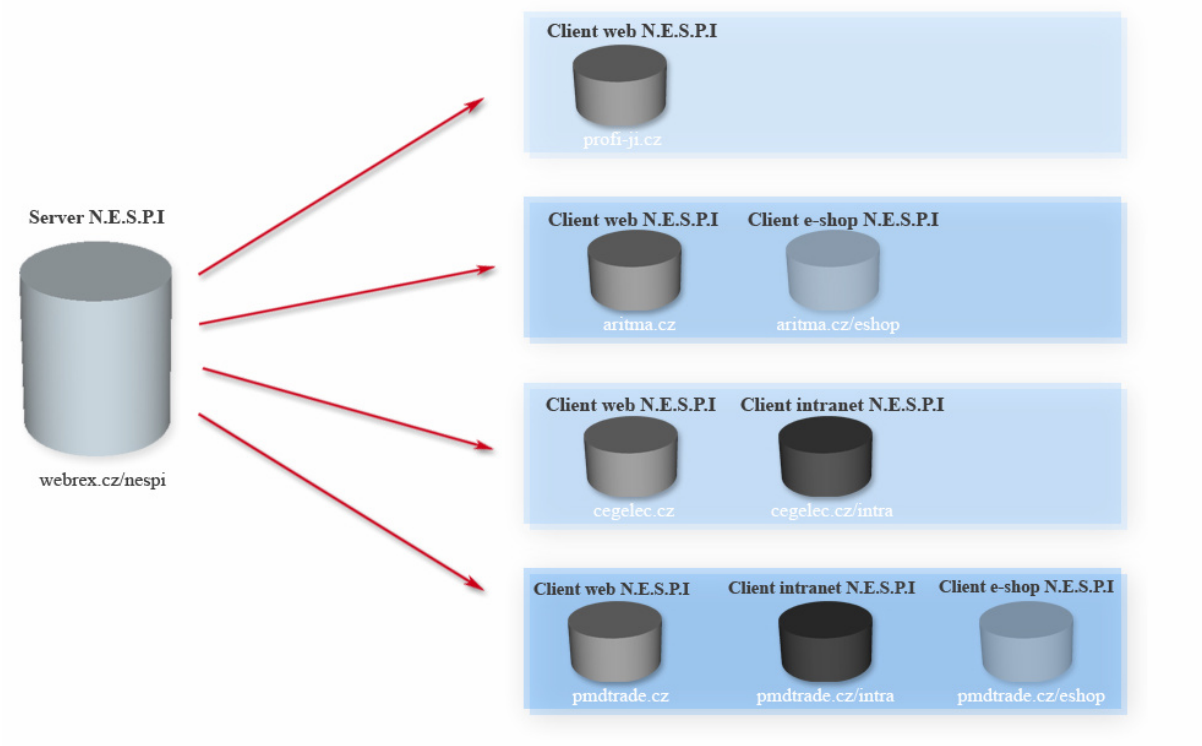


Obr. 7: Model případů použití (use case diagram)

3.6 Konceptuální datový model (ER-model)

Konceptuální model je pokusem umožnit vytvoření popisu dat v databázi nezávisle na jejich uložení - jsou formalizovaným popisem modelované reality. Sémantické modely slouží obvykle k vytvoření schémat s následnou transformací na databázové schéma. Klasickým způsobem prezentace modelovaného světa je E-R model, nebo-li entitně-relační model, pracující s pojmy entita (objekt), vztah (relationship), atribut (vlastnost). Jeho základem je převedení komplexních struktur modelované skutečnosti do dvourozměrných tabulek a nalezení vztahů mezi nimi. Právě snadný převod do tabulek ho činí vhodným pro relační databáze. Dále se používá pro vizuální reprezentaci dat. Má nezastupitelnou úlohu nejen při návrhu databázových aplikací, ale i při jejich optimalizaci a odstraňování chyb. Efektivní datový model přesně a úplně popisuje a vyhovuje nárokům zadání a je použitelný pro tvůrce databáze, eliminuje redundanci dat a je nezávislý na hardwaru a softwaru.

Nyní je vhodná doba, abychom si graficky znázornili (Obrázek č.8) jak bude systém fyzicky vypadat, což přispěje k pochopení celé problematiky. Poté již přestoupíme k samotným diagramům pro serverovou a klientskou část aplikace.

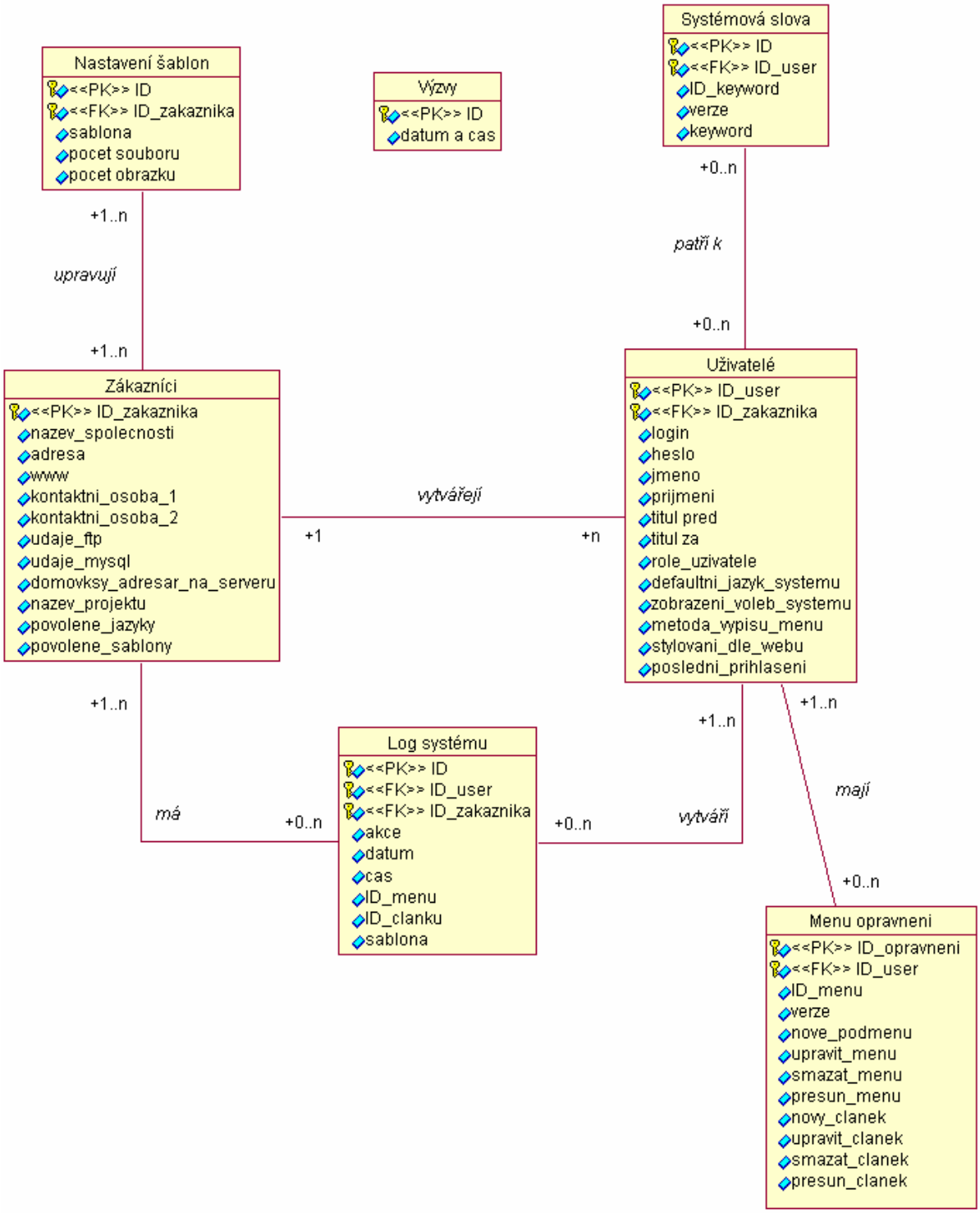


Obr. 8: Fyzická struktura celého editačního systému N.E.S.P.I.

Z nákresu je patrné, že samotný editační systém (dále ho budeme označovat zkráceně pouze jako server) bude umístěn na jiném místě než jeho klienti. Toto je opravdu radikální změna oproti původnímu editačnímu systému, neboť ten samotný byl součástí webové prezentace zákazníků. Nyní již máme oddělenou administraci webových stránek od jeho vlastního obsahu. Samotný web (e-shop) zákazníků pak bude obsahovat několik málo skriptů, které budou mít na starosti pouze vykreslování vlastního obsahu a tento obsah bude editován (spravován) právě ze serveru. Navíc tím docílíme toho, že všichni zákazníci budou používat stejné prostředí, čímž bude umožněno efektivnější vyvíjení systému a různá vylepšení budou k dispozici okamžitě všem uživatelům. Toto je hlavní myšlenka a cíl mé diplomové práce. Nyní již přejdeme k samotnému datovému modelování obou částí systému.

3.6.1 ER diagram pro editační systém - serverová část

Na Obrázku č. 9 je vyobrazena struktura databáze pro editační systém. Na první pohled je zřejmé, že databáze není příliš složitá. Je to způsobeno tím, že samotný systém ke svému běhu nepotřebuje příliš mnoho informací. Důležité je to, jak systém obsluhuje klientské strany.



Obr. 9: ER diagram pro serverovou část systému

Nyní si ve stručnosti popíšeme jednotlivé tabulky systému a popíšeme si významné atributy v nich.

- **Uživatelé** – dle mého názoru je to nejdůležitější tabulka v systému. Udržuje osobní informace u uživatelích (jméno, příjmení atd.), položky nastavení systému (role uživatele, výchozí jazyk systému atd.) a uživatel má zde uloženou i roli, pod kterou v systému vystupuje. Pro přihlášení má každý uživatel login a heslo, které je uloženo jako otisk hashovaní funkce md5.
- **Zákazníci** – tato tabulka udržuje informace o zákaznících systému. U každého zákazníka jsou potřeba znát jeho kontaktní údaje (název společnosti, kontaktní osoby atd.), speciální nastavení pro jednotlivé zákazníky: přístupy k databázím a ftp klienta, povolené šablony pro vytvoření článků a povolené jazyky na webu pro zákazníka.
- **Log systému** – zde se uchovávají informace o tom, kteří uživatelé prováděli určité akce v systému (přihlášení do systému, mazání článku a menu, jejich úprava atd.). Tato funkce je běžná pro vetší systémy a slouží k odhalení nedostatků způsobené lidským faktorem. Propojení tabulky do `Uživatelé` je z důvodu uchovaní dané akce pro jednotlivého uživatele systému. Nicméně je nutné i propojení do tabulky `Zákazníci`, neboť když programátoři a manažeři firmy pracují s webem určitého zákazníka je nutné tuto informaci poznamenat.
- **Systémová slova** – tabulka slouží výhradně k uchování názvů položek v systému (názvy sekcí v systémů, položky formulářů atd.) pro různé jazykové verze.
- **Nastavení šablon** – v tabulce je uloženo pro každého zákazníka kolik souborů a obrázků může vložit k určité šabloně pro vytváření článků. Např. některé šablony mohou obsahovat soubory i obrázky a některé pouze soubory anebo obrázky. Více si ukážeme v následující podkapitole.
- **Menu oprávnění** – zde se uchovávají nastavení oprávnění pro obyčejné uživatele systému. Zde administrátoři svým uživatelům nastaví, které úkony mohou v systému vykonávat (vkládat, upravovat a mazat menu nebo články).
- **Výzvy** – z ER diagramu vidíme, že tabulka není propojena s jinou tabulkou. Důvodem je, že tato tabulka slouží výhradně k bezpečnosti systému, jenž bude popsána v kapitole 4.5.1.

3.6.2 ER- diagram pro klientskou část systému – web

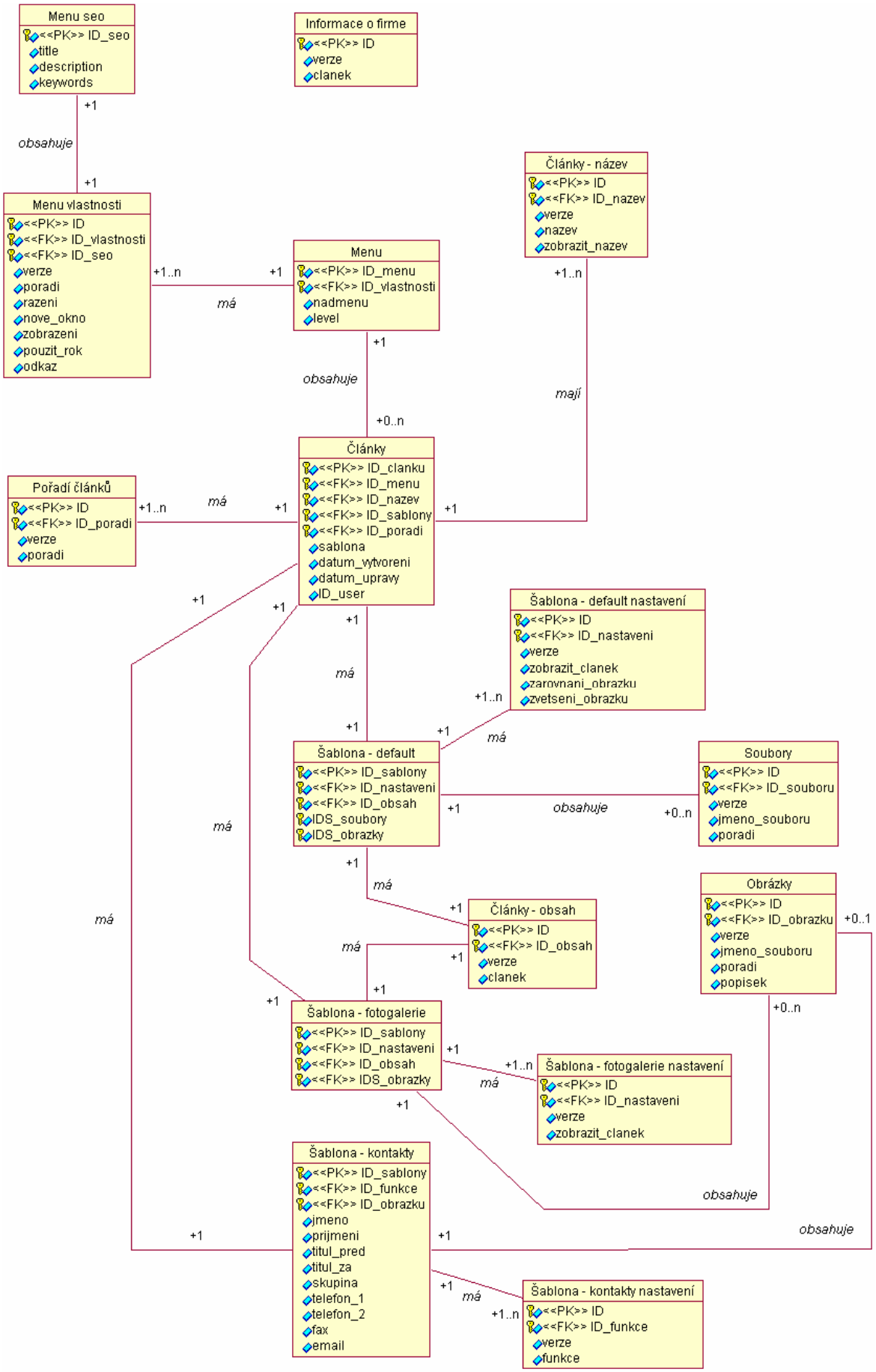
Na následujícím Obrázku č. 10 je nakreslena struktura datového modelu pro klientské části systému. Tato datová struktura bude uložena na klientských stranách a s touto bude pracovat (editovat) serverová část systému. Nyní si popíšeme jednotlivé tabulky a jejich účel.

- **Menu** – pravděpodobně nejdůležitější tabulka na klientské straně, neboť na ní se odvíjí spoustu dalších věcí. Každé menu na webu představuje určitý obsah, který se skládá z různého počtu článků podle různých šablon. Ke každému menu se váže několik článků, jenž jsou vytvořeny podle určitých šablon. Menu dále obsahuje položku nadmenu, díky tomu můžeme vytvořit velkou hierarchii menu a podmenu do několika úrovní. To je potřeba u webu firem, který mají svůj web zaměřený na prezentaci svých produktů. Např. u produktů `procesory` by byly podmenu `Intel` a `AMD` a ke každému z nich by byly další podmenu `dvoujádrové` a `čtyřjádrové`.

- **Menu vlastnosti** – tabulka schraňuje informace o každém menu pro každou jazykovou verzi. Důležitá položka je především název menu. Další položky slouží k nastavení vlastností daného menu: pořadí (určuje pořadí menu v hiearchii), zobrazení (zda se na webu menu zobrazí) atd.
- **Menu seo** – aby bylo možné webové stránky zákazníků nalézt ve webových vyhledávačích, je nutné každé webové stránce přidat informace pro tyto vyhledávače. Jsou to titulek stránky (title), popis stránky (description) a klíčová slova (keywords). Všechny tyto informace o každém menu uchovává právě tato tabulka.
- **Články** – obsahuje údaje o článcích, které jsou vytvořené pro dané menu. Podle které šablony byl článek vytvořen, datum a čas vytvoření a poslední změny.
- **Články název –** zde jsou uloženy názvy pro všechny články pro každou jazykovou verzi.
- **Články pořadí –** určuje pořadí jednotlivých článků pro dané menu.
- **Soubory –** obsahuje jména souborů, které jsou vloženy k některým šablonám a samozřejmě pro všechny jazykové verze.
- **Obrázky –** obdobně jako soubory obsahuje jména vložených obrázků a navíc ještě mají popisek obrázku.

Nyní jsme si prošli základní strukturu klientské části a nyní se podrobněji podíváme na jednotlivé šablony pro vytváření článků. Prozatím uvažujeme pouze 3 šablony. Další budou přidány až z postupem času, nutno podotknout, že jejich přidání do systému nebude příliš složité.

- **Šablona default –** tato šablona, jak již název napovídá, slouží jako všestranně použitelná pro tvorbu článků. Pomocí této šablony může být vložen obsah na web a k němu může být přiloženo několik souborů a obrázků.
- **Šablona default nastavení –** zde je uloženo nastavení defaultní šablony. Zda se zobrazí obsah článku, kam se zarovnají obrázky a zda se obrázky po kliknutí zvětší.
- **Šablona fotogalerie –** slouží pro vkládání většího množství obrázků.
- **Šablona fotogalerie nastavení –** opět nastavení pro šablonu fotogalerie, zde je pouze nastavení, jestli se zobrazí obsah článku na webu zákazníka.
- **Šablona kontakty –** tabulka pro uchování kontaktních údajů osob, jenž potřebují být uvedeny na zákaznickém webu.
- **Šablona kontakty nastavení –** zde se udržuje informace o funkci určité osoby v podporovaných jazykových verzích.



Obr. 10: ER diagram pro webovou klientskou část systému

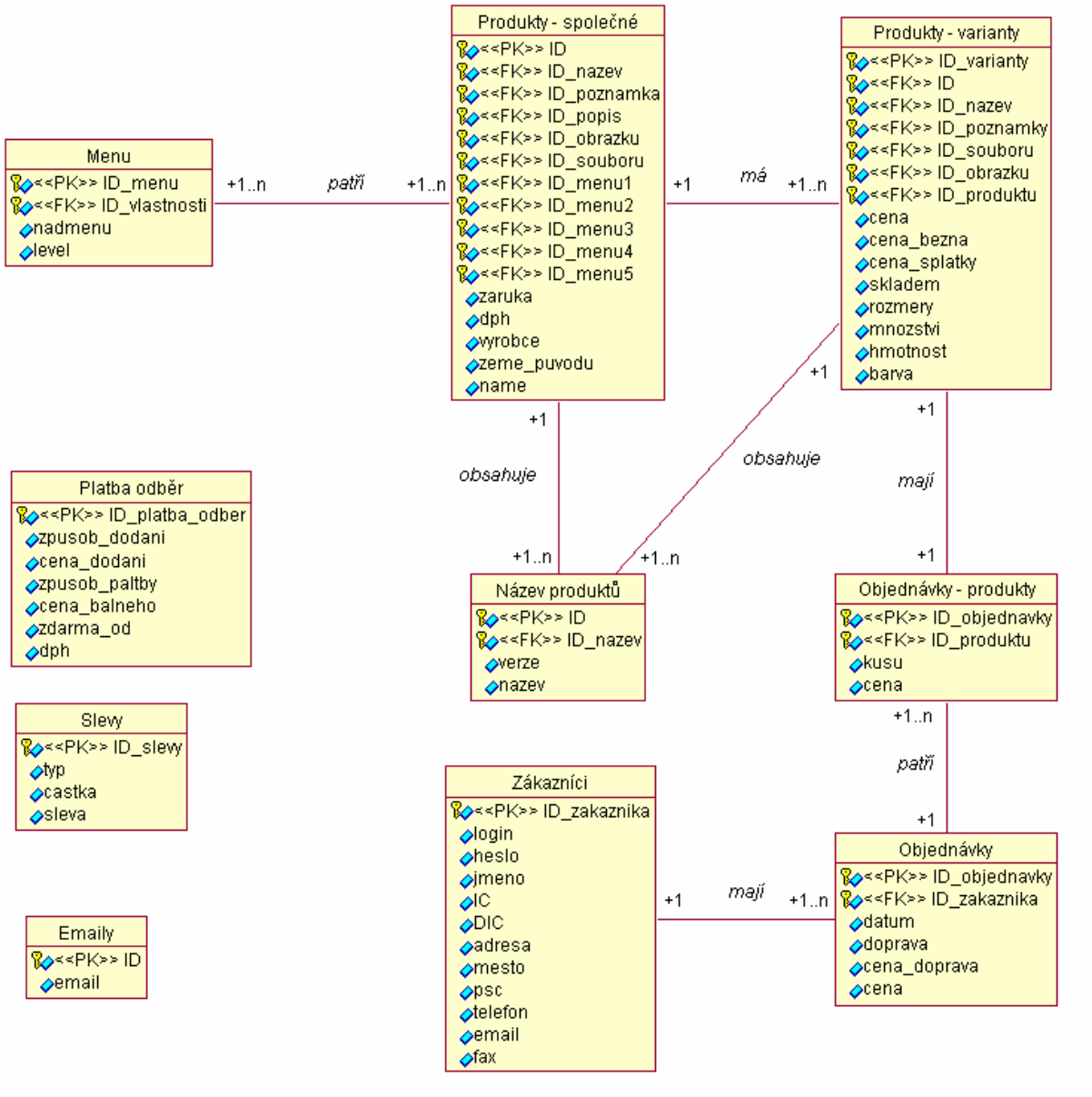
3.6.3 ER- diagram pro klientskou část – eshop

Na následujícím Obrázku č. 11 vidíme diagram pro klientskou část aplikace zaměřenou na správu internetových obchodů zákazníků. Nejdříve si ovšem popíšeme jednotlivé tabulky, abychom pak měli lepší přehled v diagramu.

- **Menu –** tuto tabulku jsem si již popsali v předešlé podkapitole. Na tuto tabulku se váže i část strany e-shopu. Struktura menu určuje strukturu produktů a ty se pak zobrazují podle zařazení do podmenu.
- **Produkty společné –** tabulka obsahuje společné informace pro produkty (výrobce, země původu atd.) a dále tento produkt může mít několik variant v e-shopu např. jinou barvu nebo jinou fotku. Dále je zde více `ID_menu`, a to z důvodů, že určitý produkt může být zobrazen ve více menu.
- **Produkty varianty –** zde jsou uloženy varianty daného produktu.
- **Produkty název –** zde jsou uloženy názvy všech produktů pro každou jazykovou verzi.
- **Objednávky produkty –** uložené informace o objednaných produktech v daném množství.
- **Objednávky –** zde již máme uloženy konkrétní objednávky pro určitého zákazníka.
- **Zákazníci –** tabulka udržuje informace o registrovaných uživatelích internetového obchodu. Uloženy jsou obdobné informace jakou u uživatelů systému, tedy přihlašovací údaje a osobní údaje.

Nyní si popíšeme tabulky, které slouží pro nastavení e-shopu:

- **Platby odběr –** udržuje informace o odběrech a platbách za produkty .
- **Emaily** – zde jsou uloženy emailové adresy, na které se posílají objednávky.
- **Slevy** – zde jsou uloženy informace o slevových akcí v e-shopu.



Obr. 11: ER diagram klientské části systému pro internetový obchod

3.7 Návrh struktury databáze

Při tvorbě databáze je nejdůležitější její návrh. Se špatným návrhem totiž padá celá databáze a tím pádem i celý vyvíjený systém. Mnoho vývojářů dělá základní chyby již na začátku, když opomíjí jakýkoliv předběžný návrh. Takto se dají tvořit jednoduché databáze, u nichž se nepředpokládá vysoké zatížení, ale pokud má být databáze pilířem aplikace, bude programátor dříve nebo později řešit vážné problémy. Při návrhu tedy bereme ohled zejména na předpokládané požadavky (zejména dotazy). Důležité je, aby v námi navržené databázi šly rychle realizovat vyhledávací a aktualizační dotazy. Dobré je ujasnit si, jak budou data do databáze vložena, aby se nevytvářela zbytečná duplicita záznamů a je také dobré při návrhu databáze omezit velikost datových typů, aby velikost databáze byla co nejmenší.

Systém využívá relační databázi MySQL, více o ní se dozvíte ve 4. kapitole. Databáze se skládá z tabulek, které odpovídají návrhu databáze v ER-modelu v předchozí podkapitole. Jednotlivé vztahy a atributy entit jsou navrženy také podle tohoto diagramu.

Ve všech tabulkách najdeme atribut ID (automatické číslo) a jednoznačně odlišuje každý záznam, což je dobré zejména při vyhledávacích a aktualizačních dotazech. U těchto údajů je uveden údaj auto_increment, kterým si systém sám generuje unikátní číselné hodnoty. Pro řetězce v databázi je použito datového typu VARCHAR, pro číselné atributy je použit datový typ INT nebo SMALLINT , pro časové údaje speciální datový typ DATE a pro obsah článku TEXT.

Podrobný návrh databáze pro serverovou a klientskou stranu webu i e-shopu najdete na přiloženém CD.

3.8 Návrh uživatelského rozhraní

Jelikož se jedná o webovou aplikaci pro zákazníky firmy, bude nutné vytvořit nejenom funkční systém, ale pro efektivní schopnost práce zákazníků a též samotných zaměstnanců, je velmi důležité brát ohled na celkový vzhled aplikace. V kapitole 3.3 jsme si řekli, že systém musí zachovávat stávající rozložení prvků a napodobovat vzhled webových stránek. Musí především ponechat jednoduché a intuitivní a přehledné zacházení, doplněné o drobné funkce, které výrazným způsobem nenaruší hlavní funkce systému.

Předběžný návrh aplikace, který byl vytvořen programátorem v rámci semestrálního projektu, vidíme na Obrázku č. 12. Nutno podotknout, že se jedná o návrh spíše z hlediska funkčnosti a rozmístění prvků systému. Nejsou zde umístěny prvky jazykových verzí a další funkce, které systém bude mít. O finální grafickou podobu se postará profesionální grafik firmy. Ještě se zmíníme o tom, že vzhled bude mít každý zákazník jiný, což je výrazná změna oproti běžným systémům, které vypadají většinou stejně. Ovšem díky XHTML 1.1 a odděleným kaskádovým stylům (CSS viz Kapitola 4.1.2) to nebude žádný problém.

Obr. 12: Předběžný návrh aplikace

Nyní jsme provedli specifikaci a analýzu požadavků na systém a můžeme přejít k další kapitole, jenž se zabývá následující fází vývoje software a tou je implementace systému.

4 Implementace systému

V následující kapitole si postupně ukážeme všechny náležitosti, jak byl systém implementován. Nejprve probereme technologie a softwarové prostředky, kterými byl systém implementován.Dále se budeme věnovat zabezpečení systému a podpoře prohlížečů k systému. Na závěr kapitoly si přiblížíme strukturu systému a jeho uživatelské prostředí. Rád bych zde podotknul důležitou věc. Ačkoliv diplomová práce pojednává o transformaci editačního systému, je nutné podotknout, že systém byl implementován celý úplně od začátku tzv. „*na zelené louce*“ a neobsahuje žádné původní zdrojové kódy. Nově vytvořená struktura systému a zdrojové kódy obsahují pouze myšlenku původního editačního systému N.E.S.P.I.

4.1 Použité implementační informační technologie

Nyní se obeznámíme s informačními technologiemi, které byly použiti pro implementaci editačního systému N.E.S.P.I. jako technologii ASP (viz kapitola 2.1).

4.1.1 XHTML

Dříve byly webové stránky formátovány prostřednictvím vlastních HTML atributů a později se toto formátovaní nahradilo promocí kaskádových stylů CSS (viz kapitola 4.1.2) kvůli oddělení vlastního obsahu HTML od jeho vzhledu a tak původní HTML do značné míry mnoho zjednodušuje. Vývoj HTML byl ukončen verzí 4 a jeho pokračování je XHTML (Extensible Hypertext Markup Language), jenž vychází z jazyka XML (viz kapitola 2.3.2). XHTML stejně jako HTML je vyvinuto konsorciem W3C a jeho původním cílem bylo HTML úplně nahradit, ale v roce 2007 došlo k založení skupiny, jenž vytváří novou verzi HTML, označenou jako HTML5. Naproti tomu je stále ve vývoji nová verze XHTML 2.0 [37].

Nyní si uvedeme některé rozdíly mezi HTML a XHTML:

- všechny tagy musí být ukončené a to včetně nepárových (
, <hr/>, <img/>, <input/>,atd.).
- všechny tagy a jejich atributy musí být zapsány malými písmeny a to z toho důvodu, že jsou takto deklarované v odkazované DTD (Dokument Type Definition).
- všechny hodnoty atributů musí být uzavřeny do uvozovek nebo apostrofů.
- jsou zakázány některé atributy pro formátování stránky (font,b,i, center, atd.).

4.1.2 CSS

CSS je zkratka pro anglický název Cascading Style Sheets, česky "kaskádové styly" a jejich využití je poměrně široké. Kaskádové, protože se na sebe mohou vrstvit definice stylu, ale platí jenom ta poslední. Je to kolekce metod pro popis způsobu grafického zobrazení stránek napsaných v jazycích HTML, XHTML nebo XML. Pomocí CSS je možné měnit cokoliv od velikosti, druh fontu a barvy textu po mezery mezi znaky a řádky, okraje a mezery kolem prvků, přesné umístění na stránce atd.

Existuje několik verzí CSS, přičemž stále nejpoužívanější je CSS2, neboť poslední revize CSS3 není plně podporována webovými prohlížeči [12].

Z předešlé kapitoly víme, že nejdůležitějším smyslem CSS je oddělení vzhledu dokumentu od jeho struktury a obsahu. Design webových stránek se pak načítá ze souboru/ů s příponou css, čímž je dokonale oddělen obsah od vzhledu a umožňuje přehledné, jednoduché, rychlé a efektivní změny vzhledu stránky.

Bohužel je zatím nevýhodou CSS stále špatná podpora v majoritních prohlížečích (IE, Mozilla a Opera). Různé prohlížeče neinterpretují CSS kód stejně a je někdy složité napsat jej tak, aby se na v různých prohlížečích zobrazovali hypertextové dokumenty stejně. Ovšem výhod při použití kaskádových stylů je více:

- rozsáhlejší možnosti formátování (padding, margin).
- konzistentní styl, tzn. že všechny nadpisy(h1..h6), odkazy atd. jsou stejné. Bez CSS se museli v HTML všechny zvlášť nastavit.
- již zmíněné oddělení vzhledu od obsahu.
- uložení CSS v externím urychluje i načítání stránek.

4.1.3 PHP vs. Ruby

V neformální specifikaci (viz kapitola 3.3) jsme uvedli, že systém bude vyvíjen v programovacím jazyce PHP nebo Ruby. O použití jazyka byla vedena dlouhá diskuze mezi manažery firmy i programátory. PHP je ve firmě zavedené a umí ho všichni programátoři a je k němu mnoho referenčních materiálů a tutoriálů na internetu. Na druhou stranu je Ruby nepříliš známou technologií, která nabízí nevídané možnosti, ovšem za cenu neznalosti a nevelké uživatelské podpory. A je o mnoho pomalejší než právě PHP. To je asi hlavní příčina proč se N.E.S.P.I., které je napsané v Ruby příliš neprosadilo. Nutno také podotknout, že neexistuje mnoho hostingů, které podporují Ruby. Z uvedených důvodu byl pro projekt vybrán PHP. V následujících podkapitolách si oba jazyky blíže popíšeme.

4.1.3.1 PHP

Je všestranný, nejrozšířenější a nejoblíbenější skriptovací jazyk pro programování webových aplikací (dynamických stránek). Nejčastěji se začleňuje přímo do struktury jazyka (X)HTML, což je velmi výhodné při jejich vytváření. Zkratka znamená rekurzivní PHP: Hypertext Preprocessor, ale původem je to **Personal Home Page**. PHP skripty jsou prováděny na straně serveru (server-side programming) a k uživateli jsou zobrazeny hypertextové dokumenty, jenž jsou jejich výsledkem. Jazyk PHP je multiplatformní a patří mezi jazyky, kde například nemusíme dopředu definovat typ proměnných a navíc proměnná může během vykonávání programu změnit svůj datový typ. Existuje několik verzí jazyka od verze 1.0 (1995) do stabilní 5.2 a budoucí verze 6.0 [47].

4.1.3.2 Ruby

Abychom byli přesní pro vývoj systému jsme neměli na mysli samotný jazyk Ruby, ale framework Ruby on Rails (RoR), který se používá pro vývoj webových aplikací napojených na databázi. Tento framework používá návrhový vzor MVC (Model View Controller), který odděluje části aplikace zodpovědné za čtení a ukládání dat včetně manipulace s nimi (model), za zobrazení grafického rozhraní aplikace (view) a za část přijímající vstupy od uživatele a řídící zobrazení dat na výstupu

(controller). Vše v RoR je založeno na jazyce Ruby, od Ajaxu v šablonách (view), přes požadavek a odpověď v controllerech po vnitřní architekturu aplikace v modelech obalujících databázi.

V RoR je vše založeno na jazyce Ruby, což je plně objektově orientovaný skriptovací jazyk, který je napsán v jazyce C a je inspirován především Perlem. Ruby byl od začátku navržen jako snadno čitelný, přehledný, lehce pochopitelný na jedné straně, ale patřičně silný gramaticky a sémanticky na straně druhé. Syntaxe je velmi podobná jako v jazyce v C++, což značně ulehčuje přechod z prostředí C++ [21].

4.1.4 JavaScript

Javascript je programovací jazyk, který vychází z objektově orientovaného modelu, a na rozdíl oproti PHP je vykonáván na klientské straně (ve webovém prohlížeči). Výhodou je menší zatížení serveru, ale na druhou stranu z toho plyne nevýhoda v podobě bezpečnostních omezení. JavaScript na straně klienta neumožňuje čtení a zapisování souborů, aby tím neohrozil uživatele. Nebyl by pak problém jednoduchým prográmkem naprosto znehodnotit obsah pevného disku a další nebezpečnou věcí je možnost načtení jakéhokoliv dokumentu zadaného v URL. Podrobněji se otázce bezpečnosti, která je v současné době velmi důležitá, budu věnovat později.

Základ jazyka Javascript (syntaktická podobnost z C++ a Javou) byl vložen do webových prohlížečů, aby umožnil vložit do HTML stránek proveditelný obsah a docílit tak dynamického chování. Stránky pak mohou obsahovat různé programy, zajišťující kontrolu dat ve formulářích a různých dynamických designových efektů. JavaScript je jazyk bez typové kontroly, což znamená, že proměnné nemusí mít specifikovaný typ a bohužel nemusí program v JavaScriptu v různých webových prohlížečích korektně fungovat. Navíc je JavaScript je čistě interpretovaný jazyk, na rozdíl od kompilovaných C/C++ a Javy, která je převáděna do bajtového kódu [28].

4.1.5 MySQL

MySQL je relační databáze typu DBMS (database managment system) a vychází z deklarativního programovacího jazyka SQL (Structured Query Language). MySQL je k dispozici pod komerční licencí i bezplatnou licencí GPL (GNU General Public License). Díky vysokému výkonu MySQL a multiplatformnímu použití (Windows, Linux, OS X) má vysoký podíl na dnes používaných databázích, především ve webových aplikacích. Od počátku vývoje MySQL byl kladen důraz především na rychlost, a to i za cenu některých zjednodušení (jednoduchá záloha dat, triggery, uložené procedury, …). Od verze 5.x se MySQL stala plnohodnotnou databází se všemi náležejícími funkcemi, ovšem na jejich úkor ztratila svůj prvotní účel a to již řečenou rychlost. MySQL je projektován pro jednoduché databáze, byť třeba s obrovskou spoustou údajů [24].

Databáze MySQL je jeden z prvních hojně rozšířených systémů a spolupracuje s ním mnoho dalších technologií: C, C++, Java, Perl, PHP, Python, Ruby, Tcl, Visual Basic nebo .NET atd. Za účelem našeho systému je MySQL nejlepší a nejjednodušší volbou.

4.2 Výběr platformy

Pro implementaci systému jsme si vybrali technologie, které jsou na platformě operačního systému nezávislé. Pro nás je důležité, že vyvíjený systém bude schopný provozu na strojích z OS Windows, Linux i Mac OS.

Systém byl implementován a testován lokálně na Windows XP sp 3 a skutečné nasazení systému bylo provedeno na 64-bitové linuxovém serveru od společnosti Český server. Na obou

platformách byla ověřena požadovaná a totožná funkčnost. Byl jsem velmi překvapen, že s nasazením systému na hosting nedošlo k výraznému zpomalení a odezvy systému.

4.3 Použitý software pro implementaci systému

Důležitou roli při implementaci hrají nejen znalosti vývojářů, ale také to, do jaké míry používají kvalitní software pro tvorbu vlastních aplikací. Pro psaní zdrojových kódů aplikace bylo použito freewareového programu **Notepad++ v5.3.1.**, což je velmi efektivní nástroj pro psaní programů. Podporuje zvýraznění syntaxe pro velké množství programovacích jazyků, uživatel si může barevně libovolnou syntaxi přizpůsobit, doplňovaní atributů funkcí a proměnných a mnoho jiných vymožeností. Díky otevřenému kódů programu existuje také mnoho pluginů, jako jsou zvýraznění tagů, přímá podpora připojení k ftp atd. Musím zde tento program zmínit, neboť mi pomohl dosáhnout mnohem větší efektivity v programování.

Jak již bylo zmíněno v předešlé kapitole, byl systém vyvíjen na lokálním počítači. Aby zde mohl být spuštěn webový server, použil jsem opět volně dostupný program **EasyPHP 3.0**, jenž obsahuje následující technologie.

Pro editování obsahu článků v systému byl použit další volně dostupný program, a to WYSIWYG editor **TinyMCE**. Ten bude v budoucnu vyměněn za lepší a placený editor Innova.

- MySQL 5.0.51a
- PHP 5.2.8
- phpMyAdmin 3.1.1
- Apache 2.2.11

Ještě pro upřesnění zde uvedu konfiguraci serveru pro chod našeho systému:

- MySQL 5.0.22
- PHP 5.2.8
- phpMyAdmin 2.1.6
- Apache 2.2.4

4.4 Podpora Prohlížečů

Ve specifikaci od zadavatele bylo, že systém musí být validní podle standardu XHTML 1.1. Všechny moduly systému proto byly validovány a případné nedostatky byly odstraněny. Většinou se jednalo o zapomenuté uzavřené tagy. Horší už bylo vytvoření validního kódu při použití několika formulářů společně s tabulkou, zde je standard 1.1 velmi nekompromisní.

Další požadavek, podpora většiny prohlížečů, už tak jednoduchý nebyl, ale i přesto byl splněn. Toto je obecný požadavek na vývoj webových aplikací, nicméně stále mnoho vývojářů tuto podmínku nedodržuje. Dnešní moderní prohlížeče podporující nejnovější standardy a se správným (očekávaným) zobrazením potíže nemají, problém ovšem nastává při použití starších prohlížečů, na mysli máme především ten od Microsoftu Internet Explorer 6.0. Je sice vynikající, že nové

moderní prohlížeče nemají problémy se správným vykreslením, bohužel stále mnohou uživatelů Internetu používá staré a nezabezpečené prohlížeče. O tom vás jistě přesvědčí následující Obrázek č. 13 [42]. Kdy ještě na konci roku 2007 byla převaha IE6 více jak třetina.

2009	IE7	IE6	IE8	Fx	Chrome	S	O
April	23.2%	15.4%	3.5%	47.1%	4.9%	3.0%	2.2%
March	24.9%	17.0%	1.4%	46.5%	4.2%	3.1%	2.3%
February	25.4%	17.4%	0.8%	46.4%	4.0%	3.0%	2.2%
January	25.7%	18.5%	0.6%	45.5%	3.9%	3.0%	2.3%
2008	**IE7**	**IE6**	**IE5**	**Fx**	**Chrome**	**S**	**O**
December	26.1%	19.6%		44.4%	3.6%	2.7%	2.4%
September	26.3%	22.3%		42.6%	3.1%	2.7%	2.0%
August	26.0%	24.5%		43.7%		2.6%	2.1%
April	24.9%	28.9%	1.0%	39.1%		2.2%	1.4%
January	21.2%	32.0%	1.5%	36.4%		1.9%	1.4%
2007	**IE7**	**IE6**	**IE5**	**Fx**	**Moz**	**S**	**O**
November	20.8%	33.6%	1.6%	36.3%	1.2%	1.8%	1.6%
July	20.1%	36.9%	1.5%	34.5%	1.4%	1.5%	1.9%
May	19.2%	38.1%	1.6%	33.7%	1.3%	1.5%	1.7%
January	13.3%	42.3%	3.0%	31.0%	1.5%	1.7%	1.5%
2006	**IE7**	**IE6**	**IE5**	**Fx**	**Moz**	**N7/8**	**O**
November	7.1%	49.9%	3.6%	29.9%	2.5%	0.2%	1.5%
July	1.9%	56.3%	4.2%	25.5%	2.3%	0.4%	1.4%
March	0.6%	58.8%	5.3%	24.5%	2.4%	0.5%	1.5%
January	0.2%	60.3%	5.5%	25.0%	3.1%	0.5%	1.6%
2005	**IE6**	**IE5**	**Fx**	**Moz**	**N7**	**O8**	**O7**
November	62.7%	6.2%	23.6%	2.8%	0.4%	1.3%	0.2%
September	69.8%	5.7%	18.0%	2.5%	0.4%	1.0%	0.2%
July	67.9%	5.9%	19.8%	2.6%	0.5%	0.8%	0.4%
March	63.6%	8.9%	18.9%	3.3%	1.0%	0.3%	1.6%

Obr. 13: Statistiky webových prohlížečů za roky 2005-2009

Systém byl bez jakýchkoliv nedostatků prověřen v následujících prohlížečích: Internet Explorer (6.0, 7.0, 8.0), Opera (8.5, 9.5, 10.00 Beta), Firefox (3.0.10, 3.5 Beta 4), Gogole Chrome (1.0, 2.0) a Konqueror (3.5.4). Systém byl vyzkoušen i v mobilním prohlížeči Opera Mini (4.2), kde byl systém plně funkční, ale jeho zobrazení bylo z části odlišné (přizpůsobené pro mobilní zařízení).

4.5 Bezpečnost systému

Jelikož se jedná o webovou aplikaci s přístupem pro mnoho uživatelů, byl při implementaci brán velký zřetel na její bezpečnost. Ta je důležitá především z hlediska důležitosti spravovaných dat. Při neoprávněném přístupu do aplikace by mohly být kompromitovány jednotlivé webové stránky nebo e-shopy jednotlivých zákazníků a při získání administrátorských práv dokonce všech zákazníků! V kapitole 3.2 jsme si popsali útoky, proti kterým byla aplikace ochráněna, nyní si ukážeme, jak bylo zabezpečení dosaženo.

4.5.1 Zabezpečení proti odposlechnutí hesla

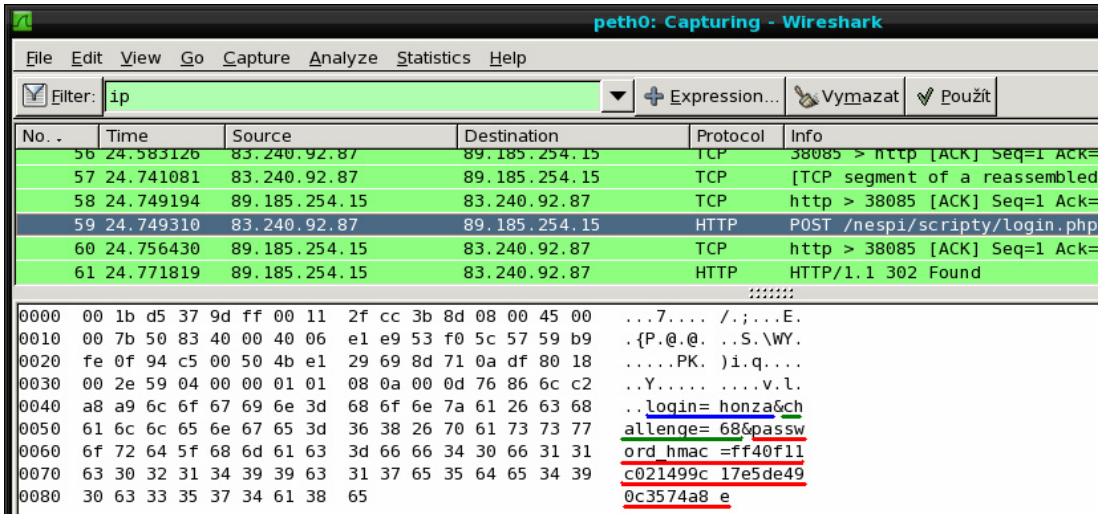
Toto je asi nejsnadnější způsob, jak se do systému neoprávněně přihlásit, pokud není systém proti tomuto útoku chráněn. Hlavní náročnost tohoto útoku spočívá v tom, že odposlouchávat data lze jen v místech, kterými data proudí. Útočník musí získat přístup k nějakému síťovému prvku mezi obětí a serverem, aby mohl sledovat probíhající komunikaci a mít zařízení pro odposlech, typicky nějaký program. Odposlouchávání dat se dá zabránit i při použití protokolu HTTPS (viz kapitola 2.3.2), kdy se šifruje přenos všech přenášených dat. To ovšem u větších aplikací zpomaluje chod systémů a ne všechny hostingy tento typ protokolu podporují. Pro náš systém není nutné použít HTTPS, neboť kromě přihlašovacích údajů systém neposkytuje citlivá data ke zneužití. Nyní si popíšeme techniku, jak dosáhnout bezpečného přihlášení s odolností na útok replay.

Jedná se o techniku výzva-odpověď, a ta funguje tak, že server vygeneruje pro klienta výzvu. Ten k ní připojí své heslo a serveru pošle otisk tohoto spojení. Server na své straně provede totéž, a pokud výsledky souhlasí, tak je uživatel přihlášen, v druhém případě samozřejmě odmítnut [30]. Někoho jistě napadne, proč to dělat tak složitě, když pouze stačí heslo zahashovat a je bezpečnost zajištěna. To je ovšem mylná domněnka, neboť útočníkovi je jedno, jestli zadá otevřené heslo přímo do přihlašovacího formuláře nebo pošle na server rovnou hash hesla, neboť oběma způsoby se dostane dovnitř systému. Bezpečnost v metodě výzva-odpověď spočívá v tom, že server každou výzvu pošle pouze jednou a pokud útočník zachytí odpověď klienta k ničemu mu to neposlouží, neboť server již takovou výzvu nikdy nepošle.

K implementaci této techniky je potřeba funkce, která realizuje výpočet otisku hesla spojeného s výzvou. K tomu slouží kód HMAC (keyed-Hash Message Authentication Code), jenž pro výpočet otisku používá kryptografické hashovací funkci, v našem případě MD5. A jako výzva pro otisk slouží poslední vložená hodnota položky ID v tabulce Výzvy (viz kapitola 3.6.1). Aby již nedošlo k opakovanému použití (zneužití) této výzvy a otisku, je po ověření uživatele tato výzva z tabulky vymazána a útočníkovi je tak znemožněno přihlášení do systému. Pro výpočet této funkce slouží na straně klienta volně dostupná javascriptová knihovna [25]. A na straně serveru následující funkce v php [30]:

```
function hmac_md5($key, $data){
     $blocksize = 64;
     if (strlen($key) > $blocksize) {
        $key = pack("H*", md5($key));
     }
     $key = str_pad($key, $blocksize, chr(0x00));
     $k_ipad = $key ^ str_repeat(chr(0x36), $blocksize);
     $k_opad = $key ^ str_repeat(chr(0x5c), $blocksize);
return md5($k_opad . pack("H*", md5($k_ipad . $data)));}
```

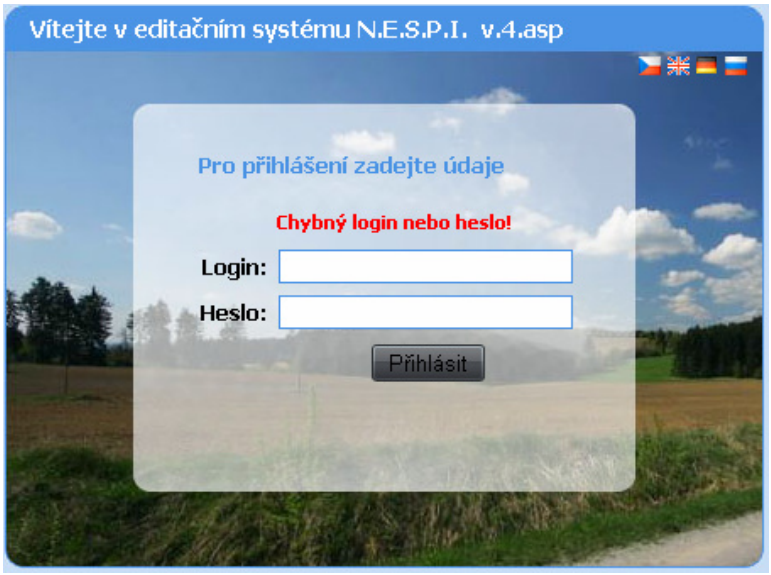
Následující Obrázek č. 14 zobrazuje data, jenž posílá přihlašovací formulář systému (Obrázek č. 15). Pro názornou ukázku byl použit program Wireshark a na vyznačených hodnotách můžeme spatřit login, výzvu (challenge) a otisk kódu hmac_md5, tedy žádné viditelné heslo.



Obr. 14: Data poslaná přihlašovacím formulářem

Při implementaci přihlašování jsem bral v potaz i útok hrubou silou (Brute Force attack), což je pokus o neoprávněné přihlášení bez znalosti hesla. Jedná se o systematické testování všech možných kombinací nebo omezené podmnožiny všech kombinací. Problémem je, když si uživatelé volí slabé heslo, což jsou kombinace různých slovních spojení nebo příliš krátká hesla. Systém uživatele o této skutečnosti informuje. Proti tomuto útoku je systém chráněn následujícím jednoduchým způsobem. Pokud nesouhlasí otisk výzvy a hesla ze serveru a klienta, je ve skriptu vložena časová prodleva. Nevylučuje to ovšem nemožnost úspěchu, ale je tak zaručena dostatečně dlouhá doba na to, aby se útočníkovi útok vyplatil.

Následující Obrázek č.15 znázorňuje přihlašovací formulář do editačního sytému N.E.S.P.I., který požaduje login a heslo. Jelikož je systém vícejazyčný, má uživatel možnost si i tento formulář zobrazit ve svém jazyce.



Obr. 15: Přihlašovací formulář pro editační systém N.E.S.P.I.

4.5.2 Zabezpečení proti SQL injection

Na tento typ útoku jsem bral z počátku implementace velký zřetel, nicméně jsem zjistil, že tento útok na systém není příliš reálný. V administrátorské části by tento útok mohli provést zaměstnanci, ale ti

nemají důvod to provádět, neboť mají jednodušší (přímý) přístup k databázi. A v části systému pro zákazníky je to opět nepravděpodobné, neboť si zákazníci spravují svoji databázi a mají opět přímý přístup k svojí databázi. Jediné potenciální a reálné riziko bylo při vytvoření nového uživatele, kdy by útočník mohl podvodně nastavit roli a získat tak přístup k celému systému a všem zákazníkům. Tato možnost ovšem byla řádně ošetřena.

4.5.3 Zabezpečení proti XSS

Dalším zabezpečením systému bylo proti útoku XSS. Obdobně jako v předešlé podkapitole je tento typ útoku na systém opět nepříliš reálný. Administrátorskou část mají na starosti zaměstnanci firmy, kteří nemají zájem vkládat škodlivý kód a pokud by tak chtěli učit, bylo by pro ně jednodušší vložit ho přímo do některých ze zdrojových kódů. V administraci webových stránek je obdobná. Jelikož se jedná o editační systém, je vkládání skriptů a HTML tagů nutně vyžadováno. Pokud administrátor nechce, aby k jednotlivým článkům a menu měl vybraný uživatel přístup, jednoduše mu to v systému zakáže pomocí oprávnění.

Pokud by byl do budoucna požadavek zadavatele, i přes nepravděpodobnost SQL injection i XSS, vytvořit proti těmto útokům obranu, bylo by nutné v případě XSS dané atributy kontrolovat na nežádoucí znaky a v případě SQL injection slashovat předávané parametry nebo jednoduše omezit jejich délku.

4.6 Struktura systému

Systém byl vyvíjen s ohledem na jeho snadné rozšíření už od počátku projektu, neboť se do budoucna počítalo s jeho rozšířením, které nebude v rámci diplomové práce. V kapitole 2.3.2 jsme si uvedli, že protokol HTTP je bezstavový a slouží ke komunikaci mezi www serverem a prohlížečem. To je dost velká nevýhoda, hlavně pro vývoj dynamické webové aplikace, protože občas (dost často) potřebujeme znát hodnotu proměnné odeslané formulářem před několika stránkami zpět. Pokud nepotřebujeme uchovat mnoho proměnných, lze je předávat v odkazech nebo pomocí skrytých formulářových prvků. Ale pokud chceme uchovat více údajů pro webovou aplikaci, což je náš případ, musíme se poohlédnout po jiné metodě. K tomuto účelu v PHP existují superglobální proměnné tzv. sessions. Díky nim mohl být systém vystavěn jako značně modulární a snadno rozšiřitelný o další funkce a v systému se nepřenáší žádné viditelné atributy v URL.

Systém má pro lepší přehlednost v adresářích oddělené skripty pro různé použití a tuto strukturu si nyní popíšeme:

- **design** – zde jsou umístěny soubory potřebné pro design systému.
- **e-shop** – budoucí úložiště pro implementaci internetového obchodu do systému.
- **main** – adresář obsahující všechny skripty samotného jádra systému.
- **scripty** – tady jsou umístěny soubory pro obsluhu souborů z části main.
- **tiny_mce** – editor pro editování článků zákazníků.
- **web** – samotný modul pro editování zákaznických webových stránek.
- **web/scripty** – opět obsluhující soubory pro formuláře z části web.
- **web/sablony** – uložené šablony pro vytváření článků na webových stránkách zákazníků.

- **zakaznici_data** – adresář slouží jako úložiště dat pro každého zákazníka. Především je zde uložen soubor kaskádových stylů (CSS), pro zobrazení systému napodobující web. Dále je zde skript zobrazující logo každého zákazníka a to z důvodu, že každý zákazník potřebuje vykreslit své logo různým způsobem (jeden nebo více obrázků, flash, flash s obrázekm atd.) a samozřejmě soubory potřebné pro vykreslení loga zákazníka.

V budoucnu se do systému zakomponuje i samotný intranet firmy a ten bude opět v odděleném adresáři. Nyní si ukážeme některé důležité skripty pro chod systému.

- **index.php** – tradiční název souboru, na který se webový server podívá v případě, že není specifikováno jméno souboru. Na tento soubor jsou směrovány všechny obsluhující skripty. Pokud není nastavena sesions přihlášení, zobrazí se přihlašovací formulář (kapitola 4.5.1), v druhém případě samotný systém, který bude popsán v následující podkapitole.
- **head.php** – zde je hlavička dokumentu, která obsahuje metadata pro systém (title, stylesheets, jazyk dokumentu, atd.) a dále jsou zde uloženy všechny javascriptové kódy.
- **main.php** – soubor, do kterého se vkládají veškeré moduly (skripty) a slouží tedy jako jádro systému pro vykreslení obsahu.
- **nastav_sekci.php** – aby mohli, být v main.php (hlavní části) zobrazeny požadované moduly je nutné nastavit sessions podle, kterých se tyto moduly spustí (zobrazí). Tuto činnost má na starosti tento skript.
- **login.php/logout.php** – jedná se o skripty starající se o přihlášení/odhlášení do/ze systému.
- **ostatní skripty** – zajišťují funkčnost uvedenou ve specifikaci: správa uživatelů a zákazníků, logování událostí, statistiky návštěvnosti atd.

Nyní přejdeme k popisu struktury editace webových stránek zákazníků. Do této části systému se z role zaměstnance firmy (programátor, manažer) dostaneme po vybrání určitého zákazníka (získáním jeho ID) a administrátor nebo uživatel se do editace webových stránek dostane rovnou po přihlášení do systému. Pro samotnou editaci webových stránek slouží následující skripty:

- **administrace.php** – obdobně jako main.php zajišťuje tento skript zobrazení veškerého obsahu pro editování zákaznických webů.
- **nastav_administraci.php** – skript opět nastavuje sessions pro vykreslení správného obsahu v editačním systému.
- **vypis_menu_click.php** – slouží pro vypsání struktury hierarchie menu pro web. Slouží jako navigace pro vkládání článků.
- **sprava_menu.php** – má na starosti přidání, úpravu a mazaní menu pro různé jazykové verze.
- **sprava_clanku.php** – opět slouží pro vkládaní, úpravu a mazaní článků přidruženým k jednotlivým menu.
- **opravneni.php** – z názvu je patrné, že administrátoři zde nastaví uživatelům, jaké mohou v editačním systému provádět akce.

- **default.php** – skript pro vykreslení formulářů defaultní šablony pro vytváření článku (nový, upravit a smazat).
- **default_script.php** – obslužný skript pro defaultní šablonu, který ukládá informace do databáze a přenáší přiložené obrázky a soubory k článku. Můžeme říct , že je to v podstatě nejdůležitější skript pro samotnou editaci článků. Ostatní šablony dále uvádět nebudeme, neboť z této vychází a liší se pouze v jiných údajích pro dané šablony. Implementoval jsem šablonu *fotogalerie*, jenž přikládá k článku větší množství fotografií a šablonu *kontakty*, která slouží pro vložení kontaktních údajů zaměstnanců konkrétní firmy na webu.
- **smaz_menu.php** – poslední z důležitých skriptů, jenž má na starosti při mazání menu smazat dané menu a jeho články a zároveň všechny jeho podmenu a jejich články. Tento soubor byl řádně testován, aby se opravdu smazalo to co se má. Musím zde podotknout že tento skript by případným útočníkům sloužil jako nástroj pomsty.
- **ostatní skripty** – dále je zde mnoho jednoduchých, nicméně nutných skriptů zajišťujících různá nastavení a funkce pro chod editačního systému: přesunutí menu a článku, nastavit oprávnění, skripty pro chod hierarchie menu atd.

V návrhu jsme provedli i analýzu internetového obchodu, bohužel zůstalo jen u návrhu, neboť v rámci časových možností nebyl e-shop implementován. Bude vyroben, až do firmy nastoupím na stálý úvazek a budu pokračovat v dalším rozvoji systému. Nyní jsme si popsali strukturu systému a prostředí pro editaci webových stránek a v následující kapitole si ukážeme vytvořené celkové uživatelské prostředí.

4.7 Uživatelské rozhraní

Vzhled systému byl vytvořen v první fázi implementace a od jeho počátku se příliš nezměnil. Abych to uvedl na pravou míru, nezměnilo se rozložení prvků systému, protože, jak již víme ze specifikace, vzhled systému je pro každého zákazníka jiný. Následující Obrázek č. 16 zachycuje systém z pohledu programátora a s již vybraným zákazníkem PROfi pro editování jejich webových stránek. Nejdříve si popíšeme hlavní 3 části, ze kterých se sytém skládá. Tou první je horní část, kde nejdůležitější částí je oblast pro zobrazování log zákazníků a dále pak menu pro akce v systému. Prostřední část logicky slouží jako obsah systému (main.php) a spodní část sloužící jako logické ohraničení samotného obsahu aplikace tzv. patička (footer).

Obr. 16: Systém s vybraným zákazníkem PROfi v editačním prostředí

- **1 - horní lišta** – slouží jako informační panel v systému. Je zde uveden aktuální datum a čas (CET) a údaje o posledním přihlášení uživatele. Obě položky lze v systému vypnout. Nejdůležitější úlohou v této části je možnost přepnout jazykovou verzi systému a zaměstnanci firmy navíc mají funkci editování a přidávaní systémových slov přímo v aplikaci. Systémovými slovy máme na mysli textové položky v systému: popisy tlačítek, názvy menu v systému, informativní texty atd.
- **2 - logo zákazníků** – oblast pro logo zákazníků.

- **3 - uživatelská lišta** – zde jsou informace o přihlášeném uživateli, který zde má možnost odhlášení, nastavení systému a při slabém hesle možnost ho změnit.
- **4 - webrex menu** – navigační panel pro zaměstnance firmy. Do této oblasti se zákazníci nedostanou. Mohli by při znalosti jmen proměnných pro nastavení dané sekce, ale tato potenciální hrozba byla u všech skriptů dostupných pouze pro zaměstnance firmy ošetřena.
- **5 - zákaznické menu** – slouží jako navigace pro zákazníky, mohou zde přepínat mezi administrací webových stránek a e-shopu, spravovat své uživatele a pokud mají administrátorská práva, tak uživatelům nastavit oprávnění pro manipulaci s jednotlivými položkami webu.
- **6 - nabídka webu** – zde jsou ještě funkce pro nastavení editování webových stránek. Pokud uživatel nemá oprávnění provádět dané akce, vypíše se mu upozornění o této skutečnosti.
- **7 - hierarchie menu** – vyobrazení struktury menu a podmenu zákaznického webu, slouží jako přístupový bod k článkům daného menu.
- **8 - nabídka editačních funkcí** – slouží pro obsluhu samotné editace menu a článků.
- **9 - výpis vložených článků** – zde jsou již vložené články k vybranému menu. Z obrázku to není dostatečně patrné, ale samotný obsah článku a přiložené soubory a obrázky jsou uživateli pomocí javascriptu zobrazeny až po kliknutí na název článku. Tato funkce slouží k přehlednosti, neboť samotný název článku moc nevypoví o obsahu.
- **10 - patička systému** – slouží pro logické oddělení obsahu aplikace. Zatím zde není nic praktického, ale do budoucna by tu mohly být další informace, např. čas vykreslení stránky, počet provedených dotazů pro vykreslení stránky atpod.

4.8 Dokumentace

Součástí každého většího projektu bývá i dokumentace, jež obsahuje ovládání programu a také samotný popis programu. Dokumentace o ovládání systému bude jeho nedílnou součástí a to zvlášť v části pro administraci webových stránek a internetového obchodu. Programová dokumentace obsahující popis algoritmů a popis jednotlivých softwarových součástí nebyla z časových důvodů realizována. Nicméně jsem sepsal prozatím manuál k projektu popisující činnost jednotlivých skriptů v systému pro snadnější hledání požadovaných skriptů k jejich úpravě a pochopení struktury celého systému. Pokud jste v předchozí kapitole 4.5 nenašli skript popisující činnost systému zadanou ve specifikaci naleznete ho v tomto manuálu, který naleznete na přiloženém CD v souboru `manual.txt`.

5 Testování

Jelikož je systém vytvářen na zakázku pro firmu WebRex s.r.o. [44] a bude nasazen do reálného prostředí, bylo při jeho vývoji nutné zajistit i dostatečné testování, kterým se odstranilo velké množství chyb. Použili jsme několik technik testování, ty si popíšeme a zhodnotíme jejich výsledky v následujících podkapitolách. Dále poznáme skutečnost, jak je testování důležitou fází životního cyklu vývoje software a je i přímou součástí fáze implementace.

5.1 Zátěžový test

Typické zátěžové testování spočívá ve vytvoření stavu, kdy s aplikací pracuje v určitou dobu požadovaný počet uživatelů a následné měření doby odezvy systému. Cílem zátěžového testování není kontrolovat funkčnost nebo správnost systému, ale zjistit, zda bude systém dostatečně rychlý i při větším počtu připojených uživatelů. Z výsledků těchto testů se poté provádějí optimalizace jednotlivých částí aplikací a opakované testy ověřují jejich úspěšnost.

Zátěžový test na vyvíjený systém byl proveden za účasti 12 uživatelů. Každý prováděl sledy úkonů, které se v systému normálně provádějí (přidání menu, správa článků, uživatelů atd.). Bylo současně editováno mnoho tabulek na straně serveru i na straně klienta. Nutno říci, že byly prováděny úkoly, které vyžadovaly větší komunikaci z databází na obou stranách. Nikdo nezaznamenal výrazné zvýšení odezvy aplikace nebo jakékoliv chybové stavy a nebylo tak nutné přistoupit k optimalizaci a provádět další zátěžový test. Dále podotýkám, že optimalizace zdrojových kódů jednotlivých částí systému byla brána na vědomí již při implementaci systému.

Výsledný průběh testu jsem předpokládal, neboť webové aplikace jsou rychlé, výkonné a nevyžadují mnoho systémových prostředků. Ke zpomalení nebo omezenější funkčnosti by mohlo nastat v případě připojení několika tisíců uživatelů do systému. Ale jelikož má firma okolo 400 zákazníků a tedy potenciálních uživatelů okolo 2000, je velmi pravděpodobné, že systém nabídne všem uživatelům rychlé a bezproblémové prostředí pro správu jejich webů.

5.2 Testování funkcionality

Funkcionální testování (neboli analytické testování) je proces, jehož cílem je zjistit, zda funkcionalita systému odpovídá požadavkům ve specifikaci a ověřit tak, že systém dělá to, co má dělat. Tento druh testování byl proveden metodou černé skříňky (black-box). Tento název plyne z neznalosti testujícího obsahu zdrojových kódů, tedy toho, jak se systém chová uvnitř. Ověřuje pouze výstupy, zda se shodují s očekávaným výsledkem.

Funkcionální testy byly také prováděny po celou dobu implementace jednotlivých částí systému, kdy sloužili pro odhalování chyb. Testování funkcionality celého systému bylo provedeno několika osobami a to z řad programátorů i obyčejných uživatelů a nebyly odhaleny žádné závažné chybové stavy. Všechny výstupy byly obdobné stejně jako ve staré verzi editačního systému.

5.3 Jednotkové testování

V tomto druhu testu se jedná o otestování každé komponenty zvlášť, a to na co nejvíce možnostmi, jenž mohou nastat. Předem si musíme říci, že tímto způsobem nebyl testován celý systém, ale jen některé jeho části a to především formuláře. Ty jsou pro činnost celého systému zcela zásadní, neboť komunikace s databází probíhá výhradně přes ně a je důležité, aby korektně zkontrolovaly naplnění všech údajů a na jejich základě vytvořily SQL dotaz, anebo zanechali systém v konzistentním stavu. Pro tento druh testování je výše zmíněná metoda černé skříňky nedostatečná a je třeba vycházet ze znalostí zdrojového kódu, tedy samotné implementace, a postupně plnit formuláře různými daty a sledovat jejich činnost.

K tomuto účelu existuje metoda bíle skříňky (white-box), kdy testující zná vnitřní logiku programu a jeho cílem je vykonání všech příkazů programu tak, aby se zjistily chyby v řídící logice. Tato metoda je ovšem spíše teoretická a v praxi jí nemůžeme nikdy dosáhnout, neboť i relativně malé programy obsahují velké množství cest, které by bylo třeba projít. Tento požadavek můžeme tedy snížit na to, abychom provedli každý případ alespoň jednou.

Důležité je zkontrolovat prázdná pole, a také, jestli jsou hodnoty vyplněné správně (např. jestli údaje odpovídají datovým typům náležejícím sloupcům určité tabulky, pro kterou je formulář určen). Bohužel vždy existuje mnoho možností, jak formulář vyplnit nesprávně a počet možností roste exponenciálně se složitostí formuláře a je tedy nemožné ověřit všechny. Je nutné aspoň ověřit prázdná pole a náležející datové typy a jejich velikost.

Daný druh testování odhalil velké množství chyb, které vznikly v průběhu implementace různých částí systému a bylo tak vytvořeno korektní chování webové aplikace. Uvedené testování považuji za nejdůležitější z uvedených testů.

5.4 Explorativní testování

Tato metoda testování je velmi účinná a úspěšně odhaluje mnoho chyb v softwarových produktech. Zvolená testovací osoba je podobná koncovému uživateli, aby se zjistilo,zda je aplikace přehledná, logicky uspořádaná a nekladla příliš vysoké nároky na uživatele, což je nežádoucí. Tester také musí být nestranný, objektivní, neměl by to být nikdo z programátorů a také by měl testovat osamocen, aby nedocházelo k jeho ovlivňování. Tato osoba je v rychlosti seznámena se systémem a poté již se systémem pracuje a zkouší dělat úkony, které bude dělat cílový uživatel. Zaznamenává veškeré chyby nebo nepřesnosti a rozpory se specifikací, na než během testování přijde.

Tímto způsobem byl systém testován během fáze implementace. Vždy se důkladně otestovala dokončená část systému, aby bylo zjištěno, že neobsahuje chyby a nemusely se zpětně opravovat. Nakonec byl systém testován jako celek.

Testování prováděly osoby vyhovující výše uvedeným podmínkám a vykonávaly v editačním systému následující úkony: přidávání menu a jejich správa, přidávání článků a jejich správa, různá nastavení v systému. V průběhu testování byly zaznamenány jisté nedostatky. Jednalo se o logické rozdělení aplikace, nedostatky v kódu a samozřejmě se vyskytly i problémy se zobrazením aplikace v různých prohlížečích, ale nevyskytla se žádná závažná chyba. Všechny zaznamenané nedostatky byly po ukončení testování odstraněny a nebylo již třeba test opakovat.

6 Závěr

6.1 Zhodnocení výsledků

V práci byla popsána problematika služeb poskytovaných na Internetu a různých technologií, které je využívají. Byly probrány technologie ASP, SOA a jednotlivé webové služby. Každou technologii jsme si podrobně popsali a rozebrali jsme jejich výhody a nevýhody. Výsledkem diplomové práce je vytvoření editačního systému poskytovaného zákazníkům jako webovou službu pro správu jejich webových stránek. Proces vývoje tohoto systému jsme si ukázali od samotného počátku, což byla analýza a specifikace, až po řádné testování naimplementovaného systému.

Byly implementovány všechny požadované funkce od zadavatele projektu uvedené ve specifikaci a analýze projektu. Systém je vícejazyčný, bylo zachováno rozložení prvků původního editačního systému a byla implementována funkce bezpečného přihlášení do systému. Vytvořené editační prostředí splňuje všechny požadavky pro tuto činnost a byly přidány funkce pro efektivnější a snadnější práci. Systém obsluhuje více rolí uživatelů a obyčejným uživatelům je možné nastavovat práva, co smějí v systému vykonávat. Systém dále umožňuje správu všech uživatelů a zákazníků, všechny jeho části jsou validní pro standard XHTML 1.1 a je kompatibilní s majoritními webovými prohlížeči.

Transformací editačního systému jsme dosáhli vyšší flexibility a úspory času s vytvářením a hlavně upravováním webových stránek zákazníků. Nyní již samotný editační systém funguje jako služba a je pouze vlastnictvím firmy WebRex s.r.o.

6.2 Budoucí vývoj systému

Nejdůležitějším krokem pro budoucí vývoje systému je rozšíření o správu zákaznických Internetových obchodů, neboť v rámci diplomové práce byl proveden pouze návrh na tuto klientskou stranu. Ve specifikaci jsme mluvili i o jednotném administračním prostředí pro intranety firem, ale v rámci diplomové práce jsme se jím téměř nezabývali. Se zadavatelem projektu jsme rozhodli, že tato část systému bude odsunuta do pozadí a budeme se plně věnovat pouze administraci webů a e-shopů. Je zde i otázka, zda-li vůbec půjde vytvořit nějaké jednotné rozhraní pro intranety různorodých firem.

V budoucnosti bude pokračovat vývoj pro editování webových stránek a to především v přidání více šablon pro vytváření článků. Postupem času se také jistě vyskytnou nějaké drobné připomínky od zákazníků, které budou muset být vyřešeny.

Dalším následujícím krokem ve vývoji systému bude integrování samotného intranetu firmy, jenž slouží pro správu projektů a dalších firemních aktivit, jenž je potřeba udržovat. S jeho vytvořením se tak systém stane multifunkčním nástrojem, jenž zajistí lepší a efektivní práci zaměstnancům firmy i jejich zákazníkům.

Literatura

[1] Benda P: *Webové služby – opatrný optimismus* [online]. [cit. 28-12-2008]. Dostupné na URL: <http://si.vse.cz/archiv/clanky/2003/05_benda.pdf>

[2] Bloomberg J.: *Servisně orientovaná architektura* [online]. [cit. 06-07-2009]. Dostupné na URL: <http://www.progress.com/cz/soa/index.ssp>

[3] Buchalcevová A., Gála L.: *Modely zralosti SOA* [online]. [cit. 08-07-2009]. Dostupné na URL: <http://si.vse.cz/archive/proceedings/2006/modely-zralosti-soa.pdf>

[4] Churý L.: *Základy XML webových služeb* [online]. [cit. 11-07-2009]. Dostupné na URL: <http://programujte.com/?akce=clanek&cl=2005081704-zaklady-xml-webovych-sluzeb>

[5] Cpress.cz: *HTTP protokol* [online]. [cit. 10-07-2009]. Dostupné na URL: <http://www.cpress.cz/knihy/tcp-ip-bezp/HTTP/http1-1.htm>

[6] Dočkal D.: *SQL injection : Princip a ochrana* [online]. [cit. 05-01-2009]. Dostupné na URL: <http://investice.ihned.cz/c4-10122900-19732020-i00000_d-sql-injection-princip-a-ochrana>

[7] Dvořák J.: *Využití technologie AJAX v univerzitním informačním systému* [online]. [cit. 11-07-2009]. Dostupné na URL: <http://is.mendelu.cz/dok_server/slozka.pl?id=16028;download=13271>

[8] Erl T.: *Servisně orientovaná architektura.* Computer Press, Brno, 2009. ISBN 978-80-251-1886-3

[9] Gregor J., Petrjanoš V.: *Když se IT stává službou*[online]. [cit. 22-12-2008]. Dostupné na URL: <http://www.progress.com/progress_software/worldwide_sites/cz/docs/soa/070913e.pdf>

[10] Halley M.: *Denial of Service útoky: man in the middle, distribuované DoS* [online]. [cit. 11-03-2009]. Dostupné na URL: <http://www.lupa.cz/clanky/denial-of-service-utoky-vyuziti-mitm-utoku/ >

[11] Holub T.: *WEB technologie - Active Server Pages* [online]. [cit. 11-12-2008] Dostupné na URL: <http://herakles.zcu.cz/local/manuals/asp/index.html>

[12] Janovský D.: *CSS - Kaskádové styly* [online]. [cit. 11-03-2009]. Dostupné na URL: <http://www.jakpsatweb.cz/css/>

[13] Jirouš V.: *Webové služby v knihovnictví* [online]. [cit. 10-07-2009]. Dostupné na URL: <http://www.akvs.cz/akp-2005/06-jirous.pdf>

[14] Kianička T.: *Princip vývoje software postavený nad průbežnou integrací na bázi SOA* [online]. [cit. 06-07-2009]. Dostupné na URL: <https://dip.felk.cvut.cz/browse/pdfcache/kianit1_2008bach.pdf>

[15] Kosek J.: *Využití webových služeb a protokolu SOAP při komunikaci* [online]. [cit. 30-12-2008]. Dostupné na URL: <http://www.kosek.cz/diplomka/html/websluzby.html>

[16] Kraus M.: Software ze zásuvky [online]. [cit. 15-12-2008]. Dostupné na URL:<http://si.vse.cz/archiv/clanky/2001/kraus.pdf>

[17] Loveček T.: *Nejznámější útoky v síti Ethernet* [online]. [cit. 05-01-2009]. Dostupné na URL: <http://connect.zive.cz/node/714>

[18] Lukeš O., Šťastný P.: *Trh ASP v ČR a ve světě* [online]. [cit. 11-12-2008]. Dostupné naURL: <http://nb.vse.cz/~vorisek/FILES/4IT417_materialy_k_predmetu/Vybrane_seminarni_prace/2007/Stastny-Lukes_Trh_ASP_v_CR_a_ve_svete.pdf>

[19] Merta M.: *WWW server* [online]. [cit. 10-07-2009]. Dostupné na URL: <http://www.fi.muni.cz/~kas/p090/referaty/2002-podzim/skupina14/apache.html>

[20] Mlynář S., Vosáhlo T. : Nové *možnosti automatizace workflow* [online]. [cit. 11-03-2009]. Dostupné na URL: <http://www.lbms.cz/Nastroje/Business-Mashups/pdf/0805-ITS-SM+TV-Nove-moznosti-automatizace-workflow.pdf>

[21] Molič J.: *Ruby on Rails: Úvod* [online]. [cit. 11-03-2009]. Dostupné na URL: <http://www.root.cz/clanky/ruby-on-rails-uvod/>

[22] Müller M.: *Web Services Description Language (WSDL)* [online]. [cit. 11-07-2009]. Dostupné na URL: <http://nb.vse.cz/~zelenyj/it380/eseje/xmulm01/wsdl.htm>

[23] Odstrčil J.: *Web Services Introduction* [online]. [cit. 11-07-2009]. Dostupné na URL: <https://dsn.felk.cvut.cz/wiki/_media/vyuka/cviceni/x36mti/prezentace2007/xodstrci-doc.pdf>

[24] Osuchowski M.: *MySQL* [online]. [cit. 16-07-2009]. Dostupné na URL: <http://www.osuchowski.cz/wwwstranky/mysql.php>

[25] Paj's Home.: *MD5 Javascript* [online]. [cit. 06-05-2009]. Dostupné na URL: <http://pajhome.org.uk/crypt/md5/md5.js>

[26] Peterka J.: *ASP, rok a něco poté…*[online]. [cit. 11-12-2008]. Dostupné na URL: <http://www.earchiv.cz/b01/b0900006.php3>

[27] Plánička P.: *JavaScript* [online]. [cit. 16-07-2009]. Dostupné na URL: <http://home.pf.jcu.cz/~pepe/Diplomky/planicka.doc>

[28] Polanský O., Voříšek J.: *SaaS model dodávky aplikací a z něho vyplívající transforamce IT průmyslu* [online]. [cit. 06-07-2009]. Dostupné na URL: <http://nb.vse.cz/~vorisek/FILES/Clanky/2008_Polansky-Vorisek_SaaS_a_transformace_IT_prumyslu.pdf>

[29] Progress Software: *Proč a jak zavádět architekturu SOA* [online]. [cit. 22-12-2008]. Dostupné na URL: <http://www.progress.com/progress_software/worldwide_sites/cz/docs/soa/070913b.pdf>

[30] Root.cz.: *Bezpečné přihlašování uživatelů* [online]. [cit. 06-05-2009]. Dostupné na URL: <http://www.root.cz/clanky/bezpecne-prihlasovani-uzivatelu/>

[31] Satrapa P.: *HTML v příkladech - Část 9 - HyperText Transfer Protocol* [online]. [cit. 10-07-2009]. Dostupné na URL: <http://www.fi.muni.cz/~kas/p090/referaty/2002-podzim/skupina14/apache.html>

[32] Schekkerman J.: *SOA maturity model* [online]. [cit. 19-12-2008]. Dostupné na URL: <http://www.sonicsoftware.com/solutions/service_oriented_architecture/soa_maturity_model/index.ssp>

[33] Sodomka P.: *Aktuální trendy vývoje českého ERP trhu (závěrečná část)* [online]. [cit. 06-07-2009]. Dostupné na URL: <http://www.cvis.cz/hlavni.php?stranka=novinky/clanek.php&id=661>

[34] soom.cz: *Pokročilé techniky XSS* [online]. [cit. 04-01-2009]. Dostupné na URL: <http://www.soom.cz/index.php?name=articles/show&aid=485>

[35] Tomek M.: *UDDI aneb jak sehnat webovou službu* [online]. [cit. 11-03-2009]. Dostupné na URL: <http://nb.vse.cz/~zelenyj/it380/eseje/xtomm19/uddi.htm>

[36] Tomíšek J., Máčel M.: *Jak uspět na trhu ASP v ČR* [online]. [cit. 06-07-2009]. Dostupné na URL: <http://si.vse.cz/archive/proceedings/2006/nastal-cas-asp.pdf>

[37] Trna M., Petrák J.: *HTML a XHTML: Často zodpovídané dotazy* [online]. [cit. 11-03-2009]. Dostupné na URL: <http://dsic.zapisky.info/XML/XHTML/FAQ/>

[38] Tuesday Business Network: Podpůrné vnitrofiremní systémy pro MSP, poskytované prostřednictvím vysokorychlostního internetu. [online]. [cit. 06-07-2009]. Dostupné na URL: <http://redakce.abchistory.cz/download-ke-stazeni/2006-broadband-studie-02-asp.pdf>

[39] Vdfwiki: *Web services* [online]. [cit. 28-12-2008]. Dostupné na URL: <http://www.vdfwiki.com/index.php?title=Web_Services>

[40] Voříšek J., Pavelka J.,Vít M. a kolektiv: *Aplikační služby IS/ICT formou ASP*. Grada, Praha, 2004. Management v informační společnosti. ISBN 80-247-0620-2

[41] Voříšek J: *Na cestě k ASP* [online]. [cit. 15-12-2008]. Dostupné naURL: <http://nb.vse.cz/~vorisek/FILES/Clanky/2002_Vorisek-Feuerlicht-Na_ceste_k_ASP.ppt>

[42] W3schools.: *Browser Statics* [online]. [cit. 06-05-2009]. Dostupné na URL: <http://www.w3schools.com/browsers/browsers_stats.asp>

[43] Waleek L.: *HTTPS* [online]. [cit. 11-07-2009]. Dostupné na URL: <http://www.abclinuxu.cz/slovnik/https>

[44] WebRex s.r.o. : *Výroba Internetových stránek* [online]. [cit. 11-12-2008]. Dostupné na URL: <http://www.webrex.cz/>

[45] Weiss P.: *Search Engine Optimization* [online]. [cit. 11-07-2009]. Dostupné na URL: <http://interval.cz/clanky/seo-search-engine-optimization/>

[46] Young J.M.: *XML - krok za krokem*. Computer Press, Brno, 2006. ISBN 80-251-1070-2

[47] Ždárek R.: *PHP příručka a manuál, php scripty ke stažení* [online]. [cit. 11-03-2009]. Dostupné na URL: <http://tvorba-webu.zdarek.com/php/>

Seznam příloh

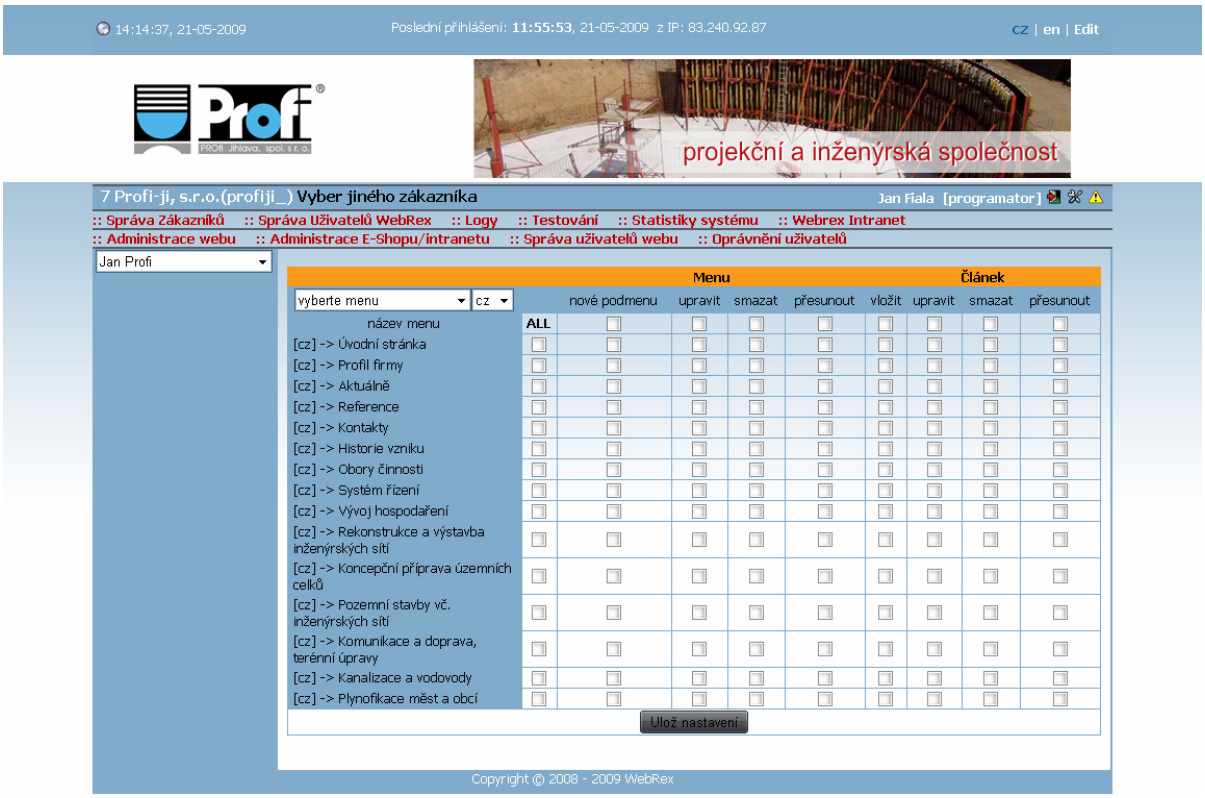
Příloha 1

Součástí diplomové práce je i přiložené CD, na kterém se nachází následující adresářová struktura s dokumenty

- obrazky – zde jsou uloženy všechny obrázky použité v technické zprávě.
- sw – programy použité při implementaci systému.
- www/server – zdrojové kódy pro serverovou část systému.
- www/klient - zdrojové kódy pro klientskou část systému.
- zdroje – uložené elektronické zdroje, z kterých jsem čerpal informace pro technickou zprávu.
- tato technická zpráva v elektronické podobě.

Příloha 2

Obr. 2-1: Systém bez vybraného uživatele



Obr. 2-2: Nastavení oprávnění pro uživatele

Obr. 2-3: Vybraný článek pro editace

Příloha 3

Databázová struktura pro server

```
CREATE TABLE IF NOT EXISTS `aaa_nespi_server_challenges` (
  `id` int(11) NOT NULL auto_increment,
  `created` datetime NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=24;

CREATE TABLE IF NOT EXISTS `aaa_nespi_server_jazyky_systemu` (
  `ID` int(20) NOT NULL auto_increment,
  `ID_jazyk` int(20) default NULL,
  `verze` varchar(5) collate utf8_bin default NULL,
  `keyword` varchar(255) collate utf8_bin default NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=27 ;

CREATE TABLE IF NOT EXISTS `aaa_nespi_server_logy_systemu` (
  `ID` int(20) NOT NULL auto_increment,
  `ID_user` int(11) NOT NULL,
  `role_uzivatele` varchar(255) collate utf8_bin default NULL,
  `ID_zakaznika` int(11) default NULL,
  `akce` varchar(255) collate utf8_bin default NULL,
  `datum` date NOT NULL,
  `cas` time NOT NULL,
  `ID_menu` int(20) default NULL,
  `ID_clanku` int(20) default NULL,
  `sablona` varchar(20) collate utf8_bin default NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=87 ;

CREATE TABLE IF NOT EXISTS `aaa_nespi_server_nastaveni_sablon` (
  `ID` int(20) NOT NULL auto_increment,
  `ID_zakaznika` int(20) default NULL,
  `sablona` varchar(20) collate utf8_bin default NULL,
```

```
  `pocet_souboru` smallint(6) default NULL,
  `pocet_obrazku` smallint(6) default NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=8 ;

CREATE TABLE IF NOT EXISTS `aaa_nespi_server_opravneni` (
  `ID` int(20) NOT NULL auto_increment,
  `ID_user` int(20) NOT NULL,
  `ID_menu` int(20) NOT NULL,
  `verze` varchar(5) collate utf8_bin NOT NULL,
  `nove_podmenu` smallint(6) NOT NULL,
  `uprav_menu` smallint(6) NOT NULL,
  `smaz_menu` smallint(6) NOT NULL,
  `presun_menu` smallint(6) NOT NULL,
  `novy_clanek` smallint(6) NOT NULL,
  `uprav_clanek` smallint(6) NOT NULL,
  `smazat_clanek` smallint(6) NOT NULL,
  `presun_clanek` smallint(6) NOT NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `aaa_nespi_server_users` (
  `ID_user` bigint(20) NOT NULL auto_increment,
  `ID_zakaznika` bigint(20) NOT NULL,
  `login` varchar(255) collate utf8_bin NOT NULL,
  `heslo_hash` varchar(255) collate utf8_bin NOT NULL,
  `slabe_heslo` varchar(255) collate utf8_bin NOT NULL,
  `prijmeni_uzivatele` varchar(255) collate utf8_bin NOT NULL,
  `jmeno_uzivatele` varchar(255) collate utf8_bin NOT NULL,
  `titul_pred` varchar(255) collate utf8_bin default NULL,
  `titul_za` varchar(255) collate utf8_bin default NULL,
  `role_uzivatele` varchar(255) collate utf8_bin NOT NULL,
  `default_jazyk` varchar(255) collate utf8_bin NOT NULL,
  `zobrazeni` varchar(255) collate utf8_bin NOT NULL,
  `zobraz_datum_cas` varchar(255) collate utf8_bin NOT NULL,
  `zobraz_prihlaseni` varchar(255) collate utf8_bin NOT NULL,
  `volba_vypisu_menu` varchar(255) collate utf8_bin NOT NULL,
```

```
  `stylovat_dle_webu` varchar(255) collate utf8_bin NOT NULL,
  `zmena_clanku` varchar(255) collate utf8_bin NOT NULL,
  `sekce` varchar(255) collate utf8_bin NOT NULL,
  `last_visit_date` date NOT NULL,
  `last_visit_date_new` date NOT NULL,
  `last_visit_time` time NOT NULL,
  `last_visit_time_new` time NOT NULL,
  `from_ip` varchar(255) collate utf8_bin NOT NULL,
  `from_ip_new` varchar(255) collate utf8_bin NOT NULL,
  PRIMARY KEY  (`ID_user`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=20 ;

CREATE TABLE IF NOT EXISTS `aaa_nespi_server_zakaznici` (
  `ID_zakaznika` bigint(20) NOT NULL auto_increment,
  `jmeno_zakaznika` varchar(255) collate utf8_bin NOT NULL,
  `adresa_ulice` varchar(255) collate utf8_bin default NULL,
  `adresa_mesto` varchar(255) collate utf8_bin default NULL,
  `adresa_psc` varchar(255) collate utf8_bin default NULL,
  `adresa_stat` varchar(255) collate utf8_bin default NULL,
  `kontakt_jmeno_1` varchar(255) collate utf8_bin NOT NULL,
  `kontakt_jmeno_2` varchar(255) collate utf8_bin NOT NULL,
  `kontakt_email_1` varchar(255) collate utf8_bin default NULL,
  `kontakt_email_2` varchar(255) collate utf8_bin default NULL,
  `kontakt_telefon_1` varchar(255) collate utf8_bin default NULL,
  `kontakt_telefon_2` varchar(255) collate utf8_bin default NULL,
  `www` varchar(255) collate utf8_bin NOT NULL,
  `ftp_host` varchar(255) collate utf8_bin NOT NULL,
  `ftp_user` varchar(255) collate utf8_bin NOT NULL,
  `ftp_password` varchar(255) collate utf8_bin NOT NULL,
  `ftp_adresar` varchar(255) collate utf8_bin NOT NULL,
  `sql_host` varchar(255) collate utf8_bin NOT NULL,
  `sql_user` varchar(255) collate utf8_bin NOT NULL,
  `sql_password` varchar(255) collate utf8_bin NOT NULL,
  `sql_database` varchar(255) collate utf8_bin NOT NULL,
  `sql_phpadmin` varchar(255) collate utf8_bin default NULL,
  `home_directory` varchar(255) collate utf8_bin NOT NULL,
  `nazev_projektu` varchar(255) collate utf8_bin NOT NULL,
```

```
  `ikona_plus` varchar(255) collate utf8_bin NOT NULL,
  `ikona_minus` varchar(255) collate utf8_bin NOT NULL,
  `povolene_jazyky` varchar(50) collate utf8_bin default NULL,
  `povolene_sablony` varchar(255) collate utf8_bin NOT NULL,
  PRIMARY KEY  (`ID_zakaznika`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=8 ;
```

Databázová struktura pro webového klienta

```
CREATE TABLE IF NOT EXISTS `profiji_clanek_nazev` (
  `ID` int(20) NOT NULL auto_increment,
  `ID_nazev` int(20) NOT NULL,
  `verze` varchar(5) collate utf8_bin NOT NULL,
  `nazev` varchar(255) collate utf8_bin NOT NULL,
  `zobrazit_nazev` enum('Ano','Ne') collate utf8_bin NOT NULL default 'Ano',
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=17 ;

CREATE TABLE IF NOT EXISTS `profiji_clanek_obsah` (
  `ID` int(20) NOT NULL auto_increment,
  `ID_obsah` int(20) NOT NULL,
  `verze` varchar(5) collate utf8_bin NOT NULL,
  `clanek` longtext collate utf8_bin NOT NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=15 ;

CREATE TABLE IF NOT EXISTS `profiji_clanky` (
  `ID` int(20) NOT NULL auto_increment,
  `ID_menu` int(20) NOT NULL,
  `ID_sablony` int(20) NOT NULL,
  `sablona` varchar(30) collate utf8_bin NOT NULL,
  `ID_nazev` int(20) NOT NULL,
  `ID_poradi` int(20) NOT NULL,
  `datum_vytvoreni` date default NULL,
  `datum_zmeny` date default NULL,
  `cas_vytvoreni` time default NULL,
  `cas_zmeny` time default NULL,
```

```
  `ID_user` int(20) NOT NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=10 ;

CREATE TABLE IF NOT EXISTS `profiji_clanky_poradi` (
  `ID` int(20) NOT NULL auto_increment,
  `ID_poradi` int(20) NOT NULL,
  `verze` varchar(5) collate utf8_bin NOT NULL,
  `poradi` int(20) NOT NULL,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=17 ;

CREATE TABLE IF NOT EXISTS `profiji_info_o_firme` (
  `ID` int(20) NOT NULL auto_increment,
  `verze` varchar(5) collate utf8_bin default NULL,
  `clanek` text collate utf8_bin,
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=6 ;

CREATE TABLE IF NOT EXISTS `profiji_menu` (
  `ID_menu` int(20) NOT NULL auto_increment,
  `ID_vlastnosti` int(20) NOT NULL,
  `nadmenu` int(11) NOT NULL,
  `level` int(11) NOT NULL,
  PRIMARY KEY  (`ID_menu`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=18 ;

CREATE TABLE IF NOT EXISTS `profiji_menu_seo` (
  `ID_seo` int(20) NOT NULL auto_increment,
  `title` varchar(255) collate utf8_bin NOT NULL default '',
  `description` varchar(255) collate utf8_bin NOT NULL default '',
  `keywords` varchar(255) collate utf8_bin NOT NULL default '',
  PRIMARY KEY  (`ID_seo`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=2 ;

CREATE TABLE IF NOT EXISTS `profiji_menu_vlastnosti` (
  `ID` int(20) NOT NULL auto_increment,
```

```
  `ID_vlastnosti` int(20) NOT NULL,
  `ID_seo` int(20) default NULL,
  `verze` varchar(5) collate utf8_bin NOT NULL,
  `nazev` varchar(150) collate utf8_bin NOT NULL,
  `poradi` smallint(6) NOT NULL default '1',
  `razeni` enum('ASC','DESC') collate utf8_bin NOT NULL default 'ASC',
  `nove_okno` enum('Ano','Ne') collate utf8_bin NOT NULL default 'Ne',
  `zobrazeni` enum('Ano','Ne') collate utf8_bin NOT NULL default 'Ano',
  `pouzit_rok` enum('Ano','Ne') collate utf8_bin NOT NULL default 'Ne',
  `odkaz` varchar(255) collate utf8_bin NOT NULL default '',
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=38 ;

CREATE TABLE IF NOT EXISTS `profiji_obrazky` (
  `ID` int(20) NOT NULL auto_increment,
  `ID_obrazku` int(20) NOT NULL,
  `verze` varchar(5) collate utf8_bin NOT NULL,
  `jmeno_souboru` varchar(80) collate utf8_bin NOT NULL default '',
  `poradi` smallint(6) NOT NULL default '0',
  `popisek` varchar(255) collate utf8_bin default '',
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=20 ;

CREATE TABLE IF NOT EXISTS `profiji_sablona_default` (
  `ID_sablony` int(20) NOT NULL auto_increment,
  `ID_nastaveni` int(20) NOT NULL,
  `ID_obsah` int(20) NOT NULL,
  `soubory` varchar(50) collate utf8_bin NOT NULL default '',
  `obrazky` varchar(50) collate utf8_bin NOT NULL default '',
  PRIMARY KEY  (`ID_sablony`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=9 ;

CREATE TABLE IF NOT EXISTS `profiji_sablona_default_nastaveni` (
  `ID` int(20) NOT NULL auto_increment,
  `ID_nastaveni` int(20) NOT NULL,
  `verze` varchar(5) collate utf8_bin NOT NULL,
  `zobrazit_clanek` enum('Ano','Ne') collate utf8_bin NOT NULL default 'Ano',
```

```
  `zarovnani_obrazku` enum('left','right','top','bottom') collate utf8_bin NOT NULL default 'right',
  `zvetseni_obrazku` enum('Ano','Ne') collate utf8_bin NOT NULL default 'Ano',
CREATE TABLE IF NOT EXISTS `profiji_sablona_fotogalerie` (
  `ID_sablony` int(20) NOT NULL auto_increment,
  `ID_nastaveni` int(20) NOT NULL,
  `ID_obsah` int(20) NOT NULL,
  `obrazky` varchar(50) collate utf8_bin NOT NULL default '',
  PRIMARY KEY  (`ID_sablony`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `profiji_sablona_fotogalerie_nastaveni` (
  `ID` int(20) NOT NULL auto_increment,
  `ID_nastaveni` int(20) NOT NULL,
  `verze` varchar(5) collate utf8_bin NOT NULL,
  `zobrazit_clanek` enum('Ano','Ne') collate utf8_bin NOT NULL default 'Ano',
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `profiji_sablona_kontakty` (
  `ID_sablony` int(20) NOT NULL auto_increment,
  `jmeno` varchar(80) collate utf8_bin NOT NULL default '',
  `prijmeni` varchar(80) collate utf8_bin NOT NULL default '',
  `titul_pred` varchar(25) collate utf8_bin default NULL,
  `titul_za` varchar(25) collate utf8_bin default NULL,
  `ID_funkce` int(20) NOT NULL,
  `skupina` varchar(80) collate utf8_bin NOT NULL,
  `telefon1` varchar(20) collate utf8_bin default NULL,
  `telefon2` varchar(20) collate utf8_bin default NULL,
  `mobil1` varchar(20) collate utf8_bin default NULL,
  `mobil2` varchar(20) collate utf8_bin default NULL,
  `fax` varchar(20) collate utf8_bin default NULL,
  `email` varchar(30) collate utf8_bin NOT NULL default '',
  `obrazek` varchar(30) collate utf8_bin default '',
  PRIMARY KEY  (`ID_sablony`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=7 ;

CREATE TABLE IF NOT EXISTS `profiji_sablona_kontakty_nastaveni` (
```

```
  `ID` int(20) NOT NULL auto_increment,
  `ID_funkce` int(20) NOT NULL,
  `verze` varchar(5) collate utf8_bin NOT NULL,
  `funkce` varchar(120) collate utf8_bin NOT NULL default '',
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=3 ;

CREATE TABLE IF NOT EXISTS `profiji_soubory` (
  `ID` int(20) NOT NULL auto_increment,
  `ID_souboru` int(20) NOT NULL,
  `verze` varchar(5) collate utf8_bin NOT NULL,
  `jmeno_souboru` varchar(80) collate utf8_bin NOT NULL default '',
  `poradi` smallint(6) NOT NULL default '0',
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=15 ;
```