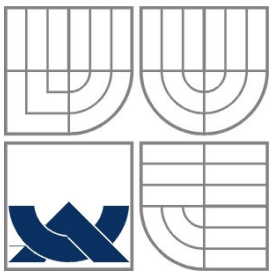
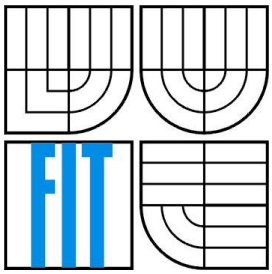


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ADAPTIVNÍ INTRANETOVÝ PORTÁL

ADAPTIVE INTRANET PORTAL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN PAVLÍČEK
AUTHOR

VEDOUCÍ PRÁCE — ING. RADEK BURGET, PH.D.
SUPERVISOR

BRNO 2009

Namísto tohoto listu
bude vloženo oficiální zadání
bakalářské práce

Abstrakt

Tato práce se zabývá analýzou potřeb společnosti ADDURRE s.r.o. Společnost potřebuje seskupovat a jednoduše manipulovat s daty potřebnými k chodu firmy. Této analýzy bude dále využito při návrhu řešení a následné implementaci nového systému. Nedílnou součástí práce je i nasazení systému do reálného firemního provozu a jeho následná technická podpora s údržbou.

Abstract

The first part of this thesis is focused on analysis of the company ADDURRE needs. The company wants to group its data from variety of systems and have easy access to their manipulation. The results of analysis will be used for projecting a proper solution, following by implementation of the new information portal. Start-up of portal, its technical support and maintenance are inseparable parts of this thesis.

Klíčová slova

Informační systém, intranetový portál, řízení toku dat.

Keywords

Information system, Intranet portal, data-flow controlling.

Citace

Pavlíček Jan: Adaptivní intranetový portál, bakalářská práce, Brno, FIT VUT v Brně, 2009.

Adaptivní intranetový portál

Prohlášení

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

……
Jan Pavlíček
11. 5. 2009

Poděkování

Rád bych poděkoval panu Ing. Radku Burgetovi, Ph.D. za odborné vedení při realizaci mé práce a inspirativní nápady. Dále bych rád poděkoval vedení společnosti ADDURRE, s.r.o., paní Mgr. Barboře Skalníkové za trpělivost a ochotnou spolupráci na vývoji samotného systému. V neposlední řadě chci také poděkovat zaměstnancům společnosti ADDURRE s.r.o. za jejich aktivní zpětnou vazbu při používání systému.

Obsah

Obsah 5
1 Úvod a cíl práce 7
2 Analýza současného stavu 8
2.1 Představení společnosti ADDURRE s.r.o. 8
2.2 Analýza firemní struktury 8
2.3 Analýza firemních procesů 9
2.3.1 Akvizice a hledání nových obchodních příležitostí 9
2.3.2 Skladování firemních dokumentů 10
2.3.3 Proces produkce zakázky 11
2.3.4 Správa webu a e-shopu 12
2.4 Doplňující požadavky na nový systém 12
3 Návrh koncepce nového systému 14
3.1 Funkčnost systému 14
3.1.1 Adresář 14
3.1.2 Dokumentový sklad 15
3.1.3 Správce zakázek 15
3.1.4 Správce obsahu webu 15
3.1.5 Správce e-shopu 15
3.1.6 Správa uživatelů, oddělení a jejich práv 16
3.2 Model případů užití 16
3.3 Použité technologie 17
3.3.1 Jazyk (x)HTML a CSS 17
3.3.2 Jazyk PHP 18
3.3.3 JavaScript a technologie AJAX 18
3.3.4 Webový a databázový server 19
4 Řešení informačního systému 20
4.1 Databázový model 21
4.2 Výkonné jádro aplikace 23
4.2.1 Logika výkonného jádra 23
4.2.2 Systémové třídy 24
4.3 Renderovací jádro aplikace 26
4.3.1 Logika renderovacího jádra 26
5 Implementace 28
5.1 Architektura modulů 28

5.2 Řízení přístupu 28
5.3 Import dat ze systému Varia 29
5.3.1 Proces vytváření kopie DBF a FPT do SQL tabulky 29
5.4 Implementace dokumentového skladu 31
5.4.1 Manipulace s adresáři 31
5.4.2 Manipulace se soubory 32
5.4.3 Schopnost propojování s jinými prvky v systému 33
5.5 Implementace správce webu a e-shopu 34
5.5.1 CMS webu 34
5.5.2 Propojení aplikace s e-shopem 34
5.6 Implementace JS a AJAX nástrojů 36
5.6.1 AJAXová manipulace s právy 36
5.6.2 Použité hotové nástroje 37
6 Závěr 38
7 Literatura 39
Příloha A – manuál k instalaci 40
Příloha B – CD se zdrojovými kódy 41

1 Úvod a cíl práce

Téma k této práci vzešlo z potřeby společnosti ADDURRE s.r.o. nakládat s firemními daty tak, aby se zefektivnily firemní procesy. Do této doby se k různým činnostem používal různý software, některé procesy byly zpomalené, zaměstnanci byli zmatení, ztráceli výkonnost a docházelo ke ztrátám dat. Hrozily tedy i ztráty reálných zakázek z důvodu neorganizovanosti dat.

Ke zlepšení situace bylo zapotřebí analyzovat současný stav a zmapovat potřeby společnosti. Toto probíhalo formou konzultace s vedením společnosti, analýzou vlastních zkušeností z provozu firmy, které jsem získal za dobu mého působení ve společnosti a aktivním monitoringem konkrétních firemních procesů.

Nasbírané informace posloužily k vytvoření modelu systému – návrhu databáze, výkonného jádra a formu frontendu. Postupné nasazení do reálného provozu přineslo další cenné informace od samotných uživatelů, kteří svými poznámkami přispěli k vylepšení a doladění funkčnosti aplikace.

Cílem práce je tedy z dostupné literatury a analyzovaných dat vytvořit aplikaci, která by byla uživatelsky jednoduchá, intuitivní, dokázala pracovat s tokem dat ze tří různých systémů a vhodně tyto systémy doplnit. Uživatelům bude poskytovat automatickou kontrolu procesů pomocí automaticky odesílaných e-mailů.

Systém by měl být plně rozšiřitelný – modulární – a administrovatelný bez větších zásahů do jádra systému.

2 Analýza současného stavu

V této kapitole bude provedena analýza samotné společnosti a čtyř základních firemních procesů. Zmapuji nedostatečnosti a díry v systému práce a navrhnu řešení pro jednotlivé problémy.

2.1 Představení společnosti ADDURRE s.r.o.

Společnost ADDURRE s.r.o. je ryze českou společností. Zabývá se marketingovou činností – v oblasti podpory prodeje (reklamní, prezentační či dárkové předměty), budování vztahu se zákazníky (CRM[1]), podporou podlinkových aktivit (teambuilding, brainstorming) a budování korporátních image (tvorba CI manuálu[2]).

Firma byla založena v roce 2004 Mgr. Barborou Skalníkovou, která využívá svých bohatých zkušeností ze svých předešlých praxí ze společnosti REDA a vzdělání v oblasti marketingové komunikace. Na českém trhu tímto rokem společnost působí již pět let a za tu dobu si vytvořila jméno spolehlivé, inovativní a kreativní agentury. Z důvodu rozšiřování kapacit a svého portfolia se společnost minulým rokem přestěhovala do nových prostor – nově zařízených kanceláří a skladového zázemí. Rovněž rozšířila své řady zaměstnanců téměř na dvojnásobek.

Mezi spokojené klienty patří společnosti RWE Transgas, a.s., Sekyra Group, a.s., APM Logistics Česká republika, a.s., Kancelář Zdravého města Brno nebo internetový portál ABC Českého hospodářství, a.s.

2.2 Analýza firemní struktury

Organizační struktura dělí společnost na oddělení. Každé oddělení je specifické svou činností a podle toho má různé pozice a úlohy ve firemním procesu. Společnost má celkem deset zaměstnanců, včetně majitelky společnosti.

- Vedení společnosti – *management firmy*
- Obchodní oddělení – *komunikace se zákazníky, akvizice, sjednávání zakázek*
- Marketingové oddělení – *propagace společnosti, reklamní kampaně*
- Nákupní oddělení – *komunikace s dodavateli, hledání dodavatelů, nákup zboží*

[1] CRM = customer relationship management

[2] CI manuál = corporate identity manuál, manuál pro používání firemního loga, barev či dalších motivů pro podporu firemní identity

- Produkční oddělení – *fyzická realizace zakázky, komunikace s dodavateli technologií*
- Technicko-hospodářské oddělení – *logistika, sklad, fakturace*
- IT oddělení a technická podpora – *správa počítačového hardware, sítě, firemních webů*

Každý zaměstnanec má své pracovní místo s počítačem. Na počítačích jsou nainstalovány operační systémy Windows XP nebo Windows Vista. Pro práci se používají programy z kancelářského balíku Office, verze 2003 a 2007. Pro evidenci faktur, objednávek, nabídek a skladové hospodářství používá společnost ekonomický software Varia[3].

Veškeré pracovní dokumenty a materiály jsou uloženy na firemním serveru, kde běží operační systém Linux Ubuntu, ke kterému, díky serverové službě Samba, přistupují zaměstnanci přes síťové disky.

2.3 Analýza firemních procesů

2.3.1 Akvizice a hledání nových obchodních příležitostí

Obchodní oddělení hledá nové zákazníky aktivním telemarketingem – průběžným telefonickým voláním na kontakty, které společnost získá buď výměnou vizitek na propagačních akcích, nebo výtahem ze zakoupené databáze firem ČR.

Tyto kontakty se zapisují do jediného souboru ve formátu MS Excel, na různé listy. Jednomu kontaktu patří jeden řádek, místo je vyhrazeno jen pro jednu kontaktní osobu. Kontaktní historie se zaznamenává do volných buněk – občas s nedostatečnou specifikací (chybějící data), občas chybí historie úplně (neexistují záznamy o interakcích s klienty). Interakce nelze rozumně spojovat s případnými dokumenty, které byly klientovi zaslány. Uspořádanost a přehlednost excelovské tabulky se exponenciálně snižuje počtem jejich uživatelů. Do databáze se zaznamenávají i kontakty, které s potenciálními klienty nemají nic společného – jsou to kontakty na dodavatele, reklamní agentury, chráněné dílny. K souboru mají přístup všichni zaměstnanci, je uložen na síťovém disku. Navíc kvůli omezení programu Excel může k souboru přistupovat zároveň jen jeden uživatel. Kontakty lze filtrovat jen v rámci možností programu Excel a kvalitě zadaných dat.

Rovněž v této tabulce nelze zaznačit jakoukoliv obchodní historii s klientem z ekonomického software Varia.

Tento firemní proces lze jednoznačně identifikovat jako naprosto neefektivní a navíc nebezpečný. Lehce lze ztratit důležitá data. Absence kvalitní databáze kontaktů, snadných

[3] Autor programu: Lubomír Pavlíček, Variasoft, http://www.variasoft.cz

operací nad ní a variabilní rozšiřitelností s doplňujícími daty firmu značně oslabuje a výrazně handicapuje na trhu.

2.3.1.1 Návrh řešení interní databáze kontaktů

V novém informačním systému bude vytvořen modul, který bude sdružovat firemní kontakty do přehledné databáze. Systém musí rozlišovat kontakty podle druhu kontaktu a podle přiděleného zaměstnance. Kromě rozšíření parametrů, které se budou ke kontaktu (firmě) vázat, bude systém umět ukládat neomezené množství kontaktních osob a záznamů o kontaktní historii (interakcích) s klientem. Systém interakcí také umožní nastavit uživateli připomenutí na další interakci.

Požadavek na načítání faktur a jejich položek ze software Varia bude vyřešen pravidelným importem dat z tohoto software do databáze systému. V kartě klienta se poté budou zobrazovat i tyto údaje.

Zaměstnanci obchodního oddělení budou mít rychlý přehled o kontaktech v této interní databázi; budou mít možnost je filtrovat podle nejrůznějších kritérií. V kartě klienta budou mít možnost vidět okamžitě kontaktní a obchodní historii. Ve spolupráci s dokumentovým skladem budou mít přístup k zaslaným nabídkám, vyhotoveným zakázkám a dalším dokumentům.

2.3.2 Skladování firemních dokumentů

Firemní dokumenty, ať ty interní – směrnice, sdělení, grafické podklady, mustry formulářů, tak externí – prezentace zákazníkům, elektronické kupní smlouvy a nabídky, se skladují na firemním serveru. Díky službě Samba pak mají přístup k těmto dokumentům přes síťové disky.

Interní dokumenty mají vyhrazený prostor na jednom disku, každé oddělení má na serveru svou složku, do které si zaměstnanci ukládají příslušné dokumenty. Přístup na disk a do složek mají ovšem všichni zaměstnanci z firemní sítě (LAN nebo VPN).

Externí dokumenty se ukládají na disk s projekty, kde má každý zákazník svou projektovou složku.

Občas se stává, že se dokument poškodí (poté se vytahuje poslední známá verze z každodenních záloh), že se dokumenty někam „zašijí“ a trvá dlouho jejich vyhledání, nebo se dokumenty často duplikují. Neexistuje žádný systém revizí. Vyhledávání dokumentů probíhá pomalu přes službu Hledání ve Windows. Vzhledem k povahám dokumentů, jenž jsou většinou z balíku MS Office, může k jednomu souboru přistupovat zároveň jen jeden uživatel.

Je zřejmé, že tento firemní proces není řešen optimálně. S přihlédnutím k úrovni zabezpečení a neorganizovanosti dokumentů lze tento proces hodnotit jako neefektivní a těžkopádný.

2.3.2.1 Návrh řešení skladu dokumentů

Navrhuji implementaci modulu, který umožní vytvořit virtuální adresářovou strukturu, jakkoliv zanořitelnou, s možností ukládat do složek různé typy a druhy souborů. Uživatel by měl možnost soubory libovolně upravovat a nahrát do systému jeho revizi – bude se moci snadno vrátit ke starší verzi souboru. Uživatel bude moci dokumenty přiřadit k firmě z databáze kontaktů nebo k jakékoliv interakci s klientem či ke konkrétní zakázce.

Tvůrce adresářové struktury bude moci zamezit přístup některým uživatelům do některých složek snadnou editací jejich nastavení.

Adresářová struktura a informace o souborech budou uloženy v databázi, jednoduchou a okamžitou indexací bude možné vyhledat jakýkoliv soubor v neporovnatelně kratším intervalu.

2.3.3 Proces produkce zakázky

Předtím, než vznikne zakázka, nabízí obchodník klientovi cenovou nabídku – tu vytvoří v systému Varia. Poté, co zákazník souhlasí, vytvoří se ve Varii kupní smlouva, která jde ke klientovi k podepsání. Jakmile je s klientem podepsána kupní smlouva, vytváří obchodník v systému Varia zakázku. Všechny tyto tři dokumenty se ovšem musí zadávat ručně – nelze převádět cenovou nabídku do kupní smlouvy a kupní smlouvu do zakázky. Fakturační oddělení po dokončení zakázky vytvoří fakturu a dodací list z položek v zakázkovém listě (nelze to udělat z kupní smlouvy).

Podle povahy zakázky si následně zakázku rozděluje produkční a nákupní oddělení. Dle obchodních podmínek má společnost povinnost dodržovat jisté termíny – 2 pracovní dny od zadání musí předložit zákazníkovi grafický návrh, do 14 dnů proběhne fyzická výroba a do 21 dnů dodávka ke klientovi. Tyto termíny si kompetentní lidé hlídají sami – mají svůj pracovní šanon, který pravidelně prohlížejí a kontrolují stavy zakázek. Soubory k zakázkám ukládají zaměstnanci do odpovídajících složek na disk s projekty.

Občas se stane, že se na nějakou část zakázky zapomene, nebo se pozdrží (buď na straně společnosti, nebo na straně klienta). Také se stává, že se k zakázce vytvoří několik verzí souborů a zpětně se dohledává, která verze je vlastně aktuální. Při nemoci se zaměstnanec zastupuje jen těžko, jelikož většinou se ve své práci vyzná jen on sám.

Proces se dá hodnotit jako relativně vyhovující, ale začíná pokulhávat, jedná-li se o specifičtější zakázku, nebo má-li se zainteresovat více lidí. Neefektivní je zvláště, když vypadne jeden zaměstnanec z procesu (nemocí, nebo kvůli jiné činnosti). Jako nepraktický se dá označit systém ukládání souborů k zakázkám – ač všichni ví, kde je zhruba najdou, již si nejsou zcela jisti tím, zda-li je daná verze dokumentu aktuální. Nejvíce času ovšem obchodníci stráví při převodu jednotlivých dokumentů – všechny položky musí totiž zadávat ručně.

2.3.3.1 Návrh optimalizace procesu a začlenění do nového systému

Aby se zefektivnila práce obchodního oddělení a aby nemuseli vytvářet tři dokumenty, bude informační systém zastupovat systém Varia ve tvorbě kupních smluv a nabídek. Informační systém si z Varie bude periodicky importovat seznam zakázek. U každé zakázky bude moci obchodník následně vytvořit cenovou nabídku a kupní smlouvu (systém nabídne požadovanou tiskovou sestavu). Ke každé zakázce si pak udělá vlastní kalendář akcí, který bude sestaven na základě pevně zadaných dat z obchodních podmínek. V kartě zakázky bude možno lehce sledovat postup jejího vyřizování, každá akce se jednoduše v systému potvrdí, přidá se poznámka, nebo soubor se schváleným materiálem (kupní smlouva, grafický návrh) – tato funkce modulu bude úzce spjata s dokumentovým skladem. Bude-li nějaké stadium zakázky zpožděno, upozorní na to systém automatickou e-mailovou zprávou zodpovědnému oddělení.

2.3.4 Správa webu a e-shopu

Společnost provozuje svůj firemní web http://www.addurre.cz a webový e-shop http://www.reklamni-predmety.name s produkty standardizovaného katalogového zboží. Autorem obou webů je IT oddělení společnosti. Žádný z těchto webů nemá ucelené administrační rozhraní a jakékoliv změny – úprava zboží, správa objednávek, správa textů – je třeba provádět zadáním požadavku na IT oddělení.

Na jednu stranu je toto řešení rozumné, IT oddělení se ve svých systémech vyzná a teoreticky nemůže nic pokazit a aktualizaci dělá rychle. Nicméně, některé změny lze ponechat i na ostatních, kompetentních zaměstnancích, zvlášť, když se k němu dodá jednoduché administrační rozhraní.

2.3.4.1 Návrh administračního rozhraní webu a e-shopu

V novém informačním systému bude vytvořen modul suplující CMS[4] webu a e-shopu. Uživatel s odpovídajícími právy bude moci přidávat či měnit tiskové zprávy a novinky na webu. Dále bude mít možnost vyhledávat v produktech na e-shopu a tyto produkty upravovat.

V modulu budou dostupné i objednávky z webu. Obchodní oddělení bude mít možnost tyto objednávky prohlížet a měnit jejich stavy.

2.4 Doplňující požadavky na nový systém

Vedení společnosti si přeje, aby bylo v systému možnost definovat oddělení a přidávat/odebírat z něj uživatele. Chce mít možnost okamžité aktivace a deaktivace uživatelských účtů.

[4] CMS = content management system, systém spravující obsah

Také chce mít plnou kontrolu nad tím, jaká přístupová práva budou mít různá oddělení k jednotlivým modulům a jejich funkcím. Práva chce udělovat zvlášť také jednotlivým uživatelům.

Systém by měl být nenáročný na provoz a maximálně bezpečný proti vnějším zásahům a přístupům. Důraz se klade také na finanční stránku – systém by neměl vyžadovat nákladný upgrade jak na firemní hardware, tak software, preferuje se využití stávajících technologií tak, aby náklady na provoz byly co nejmenší. Systém by měl být rychlý, svižný a umožňovat okamžitý přístup více zaměstnanců v jednu chvíli do jedné agendy.

3 Návrh koncepce nového systému

Na základě analýzy byl pro společnost ADDURRE s.r.o. vypracován návrh nového informačního systému.

Základní koncepcí je vytvoření webové aplikace, naprogramované ve skriptovacím jazyce PHP. K zobrazování frontendu bude využito možností značkovacího jazyka HTML a stylistických prvků CSS. Data budou spravována databázovým systémem MySQL. Aplikace poběží na firemním serveru *adelka*, kde je nainstalován webový server Apache. K hotovému produktu budou zaměstnanci přistupovat přes jakýkoliv internetový prohlížeč zadáním adresy http://addis.addurre.cz, která bude u registrátora firemní domény nasměrována na firemní IP adresu. Budou moci také přistupovat z LAN nebo VPN přímým zadáním adresy http://adelka/addis/. Server je do Internetu připojen linkou 10/10 Mbit, takže ani externí zaměstnanci nebudou mít problém s rychlým přístupem do systému.

Zvolené řešení je pro společnost tím nejlevnějším. Náklady na provoz jsou téměř nulové, firemní server již dnes běží 24 hodin denně a momentálně je využit jen z 10% celkové kapacity. Aplikace bude platformově nezávislá, případná změna operačního systému na serveru nebude portálu vadit. Na uživatelských pracovních stanicích postačí k přístupu do systému obyčejný internetový prohlížeč podporující standardy XHTML, CSS a JavaScriptu.

3.1 Funkčnost systému

Do základní verze systému, vytvořenou v rámci této práce, byly navrženy následující funkční celky (resp. moduly).

3.1.1 Adresář

Adresář, jako stěžejní prvek celého systému, bude schopen přidávat, editovat a spravovat kontakty s nejrůznějšími údaji. Jednotlivé kontakty budou moci patřit do různých kategorií. Ke kontaktům bude možno přidávat kontaktní osoby. Důležitou funkcí bude správa interakcí s jednotlivými kontakty. Adresář bude schopen propojení kontaktů se soubory (např. obchodní podmínky, logo), nebo taktéž u konkrétních interakcí (např. zaslaná prezentace). Ze systému Varia si bude načítat obchodní historii (fakturaci).

Samozřejmostí je rozšířená filtrace výpisu kontaktů/kontaktních osob/interakcí podle nejrůznějších kritérií s vyhledávačem a export výpisů do formátu CSV pro další manipulaci v programu MS Excel.

3.1.2 Dokumentový sklad

Dokumentový sklad bude sloužit jako náhrada za jeden ze síťových disků. Bude schopen úplné správy složek ve formě souborového systému – vytváření, mazání, přesouvání a zanořování do (teoreticky) neomezené úrovně. Systém bude znát zhruba třicet nejpoužívanějších typů souborů. Jednotlivé složky je možné zamknout vůči různým skupinám uživatelů. U souborů bude možno zadat popisek a propojit jej s kontaktem, interakcí nebo zakázkou. Systém bude ukládat revize souborů – nemusí se nahrávat nový soubor, ale jen jeho nová revize, přičemž starší verze budou stále k dispozici.

Uživatel bude moci vyhledávat soubory v celém stromu souborového systému, exportovat seznam souborů a samozřejmě také filtrovat výpis podle různých kritérií.

Dokumentový sklad budou využívat i jiné funkční celky – např. *adresář* nebo *zakázky*.

3.1.3 Správce zakázek

Správce si bude z Varie periodicky načítat seznam všech zakázek. U každé bude možnost vidět seznam jejích položek a průběh práce – systém bude hlídat faktory zpracovávání zakázek (u různých typů zakázek různé faktory) a průběžně informovat kompetentní osoby o plnění daného bodu. U zakázek také systém nabídne vytvoření tiskové sestavy na cenovou nabídku a kupní smlouvu, kterou bude moci obchodník zaslat klientovi ke schválení. Rovněž bude možnost přiřadit k zakázce soubory (např. grafické návrhy, kupní smlouvy apod.). V případě, že se společnost (resp. IČO), která je u zakázky uvedena , nenachází v *adresáři*, nabídne její uložení.

3.1.4 Správce obsahu webu

Zde bude mít kompetentní uživatel možnost úpravy a vkládání článků na web do sekce *Tiskové zprávy*. Správce bude mít přístup do databáze článků. Pomocí editoru jednoduše vytvoří krátký a hlavní článek, který se ihned objeví na webu.

3.1.5 Správce e-shopu

Správce bude mít přístup do databáze e-shopu. Uživatel bude mít v základním zobrazení přehled o objednávkách. U každé objednávky pak bude možnost ručně nastavit její stav. Systém sám páruje webovou objednávku se zakázkou z Varie a bude řídit její stav (nalezne-li zakázku v systému, nastaví stav „Vyřizuje se“, pokud je zakázka dokončena, nastaví stav „Vyřízeno“). Nenalezne-li v *adresáři* IČ objednávajícího, nabídne jeho uložení do databáze, a zároveň do první interakce vloží odkaz na webovou objednávku.

Další funkcí správce e-shopu bude správa nabízeného sortimentu. Uživatelé budou moci procházet sortiment, přidávat a měnit zboží i s obrázky. Uživatel také bude mít přehled o obratu výrobku a také kolik návštěvníků e-shopu si zobrazilo detail produktu.

V dalším zobrazení bude mít uživatel možnost vidět statistiku hledaných slov a frází ve vyhledávači e-shopu. V přehledu lze vidět kolikrát lidé hledali v určitém období danou frázi. Byl-li navíc uživatel přihlášen, objeví se u zákazníka, o co měl zájem. Z takového přehledu pak lze vytvářet analýzu zájmů návštěvníků webu.

Rovněž je možnost filtrace aktuálního výpisu podle různých kritérií a jejich export do formátu CSV.

3.1.6 Správa uživatelů, oddělení a jejich práv

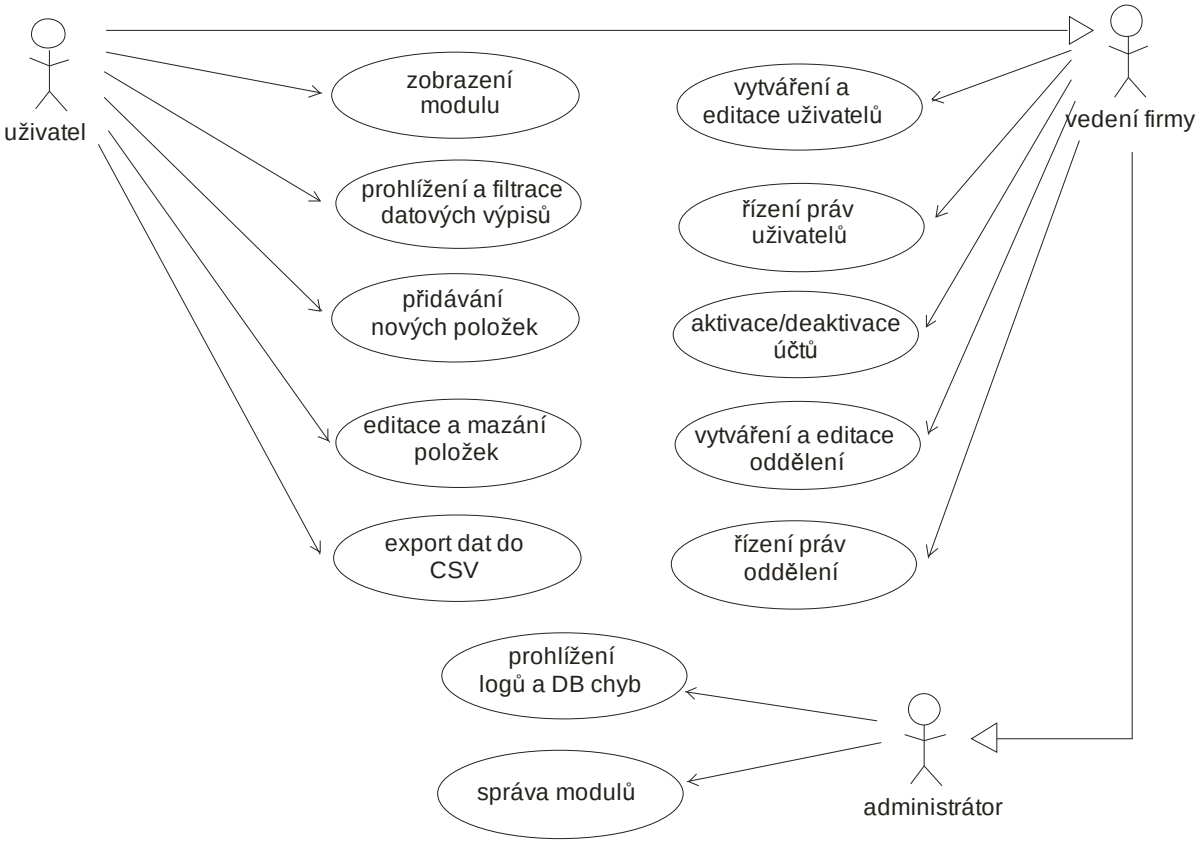
Samostatnou funkční jednotkou je administrace uživatelů informačního systému, jejich zařazení v odděleních a především jejich práva k jednotlivým funkčním celkům.

Administrátor bude moci libovolně aktivovat/deaktivovat, přidávat či editovat uživatele. To samé bude moci dělat se skupinami uživatelů (oddělení). Ve správě práv mu systém nabídne seznam všech dostupných práv pro jednotlivé funkce – standardně stačí nastavit právo pro oddělení. Bude-li ovšem administrátor chtít explicitně zakázat/povolit některou funkci přímo uživateli, má i tuto možnost. Informace o odděleních uživatelů dále využívají i jiné funkční celky (např. Dokumentový sklad).

3.2 Model případů užití

V systému budou existovat tři základní role uživatelů. *Běžný uživatel* má právo na běžné uživatelské akce, má-li k tomu odpovídající právo v daném modulu. Nad uživatelem stojí *vedení společnosti*, které dědí akce po uživateli a zároveň může vykonávat akce vytváření/editace/aktivace samotných uživatelů, manipulovat s jejich právy a ty samé akce může provádět s odděleními. Třetím typem uživatele je *administrátor* systému, který dědí akce po vedení společnosti. Oproti vedení má navíc právo akce správy modulů a prohlížení databázových chyb a systémového logu.

Na obrázku [Obrázek 1] je naznačen návrh use-case diagramu pro informační systém.



Obrázek 1 – Use-case diagram informačního systému

3.3 Použité technologie

3.3.1 Jazyk (x)HTML a CSS

Jazyk HTML[1] vznikl na začátku 90. let 20. století při potřebě interpretovat dokumenty v informačním systému švýcarského CERNu. O jeho vznik se zasloužili Tim Berners-Lee a Robert Caillau. Zároveň s ním vznikl standard protokolu HTTP[5] pro přenos těchto dokumentů v počítačové síti. Brzy na to byl stejnými autory vyvinut první prohlížeč podporující tento standard s názvem WorldWideWeb, dnes běžně užívaná zkratka pro celosvětovou informační dálnici.

K jazyku HTML a jeho novější variantě XHTML[6] existuje přesná specifikace. Základní struktura dokumentu se skládá z hlavičky a těla dokumentu. V hlavičce se specifikují informace o kódování, stylovém souboru, klíčových slovech a popisek dokumentu. V samotném těle stránky se definuje obsah, který chceme uživateli interpretovat.

[5] HTTP = HyperText Transfer Protocol, protokol na přenos hypertextových protokolů

[6] XHTML = Extensible HTML, rozšířitelný HTML

Samotný obsah dokumentu se skládá z *prvků*, *atributů*, *entit* a *komentářů*. Prvky (zažitějším výrazem jsou někdy označovány jako *tagy*) mají svou počáteční a koncovou značku (u staršího HTML existují výjimky nepárových značek) uvozenou špičatými závorkami – < a >. Prvek ve své deklaraci obsahuje jméno a může obsahovat (povolené) atributy. Koncová značka je definována lomítkem a názvem prvku, který uzavíráme.

V HTML existují jisté zásady, které pomáhají k lepší orientaci v kódu a kompatibilitě s prohlížeči a vyhledávači internetových dokumentů. Platí, že v prvku můžou být definovány další prvky, nebo samotný interpretovaný text, entity nebo komentáře. Podle charakteru prvků se tyto dělí na blokové a řádkové a platí, že blokové prvky se nesmí umisťovat do řádkových. Prvky by se neměly křížit (např. <prvek1><prvek2></prvek1></prvek2> je špatně).

Pro definici fontů, grafické efekty objektů či jejich umístění a další stylizování je využito kaskádových stylů specifikace CSS[2]. Stylizovat prvky HTML můžeme použitím selektorů, identifikátorů, pseudotříd nebo jejich kombinací. Seznam stylů se uvozuje složenými závorkami a jednotlivé deklarace se oddělují středníkem.

Kaskádové styly se mohou zapisovat přímo do atributů prvku (tzv. inlineový zápis) nebo definovat v hlavičce HTML dokumentu. Nicméně nejlepší volbou v rozsáhlém projektu je umístit tyto styly do zvláštního souboru s příponou *css*, který se poté připojí do HTML hlavičky.

Kód informačního systému bude plně validní XHTML za plné podpory CSS stylů.

3.3.2 Jazyk PHP

PHP[3] je skriptovací jazyk, určený pro programování dynamických internetových aplikací. Může se začlenit přímo do struktury jazyka HTML, nebo může sám generovat kód HTML.

PHP skript se kompiluje a provádí na straně serveru. Ve skriptu se uvozuje znaky <? a ?>. Vývojáři jazyka se v syntaxi inspirovali více programovacími jazyky, podobnosti lze nalézt s jazykem C, Java i Perl. Od páté verze navíc PHP plně podporuje objektový model, inspirovaný jazykem C++, a lze využít většinu známých technik OOP.

Nově vytvářený informační systém bude založen právě na tomto jazyce a bude se maximálně opírat o myšlenku objektového programování.

3.3.3 JavaScript a technologie AJAX

Javascript je rovněž skriptovací jazyk, který se používá převážně pro vylepšení dynamičnosti a interaktivnosti webových stránek. Jsou jím obvykle ovládány DOM[7] prvky na frontendu. Vkládá se přímo do HTML kódu, většinou se však, opět jako CSS, připojuje do HTML hlavičky jako externí soubor.

[7] Document Object Model = objektový model dokumentu

Javascript je ryze objektový. Syntaxe jazyka je inspirována jazyky C, C++ a Java. Je prováděn na straně klienta integrovaným interpretem v prohlížeči.

AJAX[4] využívá javascriptové práce s DOM a objektu XMLHttpRequest, který vytvořila společnost Microsoft s pátou verzí prohlížeče Internet Explorer. V dnešní době se těší čím dál tím větší oblibě, jelikož stránkám dodávají nový rozměr dynamičnosti. Stránky se nemusí při každé operaci znovu složitě načítat. Aplikace založené na AJAXu s přidanou hodnotou multimediálních schopností, např. přehrávání videa, se dnes označují jako Web 2.0. Mezi nejznámější aplikace patří např. Facebook, webové aplikace Google, nebo videoserver YouTube.

V novém informačním portálu budou javascriptové a AJAXové funkce použity spíše okrajově a budou sloužit zejména ke kontrole formulářů či rozšíření formuláře na základě zadaných dat.

3.3.4 Webový a databázový server

Jako webový server bylo zvoleno řešení od Apache Software Foundation, webový server Apache ve své druhé verzi. Výhodou je opět jeho multiplatformost – běží stejně spolehlivě jak na OS Linux, tak na OS Windows, dostupnost – řešení je k dispozici pod licencí GPL[5], důvěryhodnost – používá se na více jak 50% webových serverech ve světě[8,6] a variabilita – pomocí zásuvných modulů a knihoven, které se kompilují do výkonného jádra, dokáže zpracovávat a generovat zdrojové kódy z nejrůznějších skriptovacích jazyků. Nabízí také nepřeberné množství konfiguračních nastavení a bezpečnostních protokolů.

Správcem databáze byl zvolen relační databázový systém MySQL[7], vytvořený společností MySQL AB. Ta jej distribuuje rovněž pod licencí GPL. Komunikace s databází probíhá pomocí jazyka SQL, jenž je dialektem procedurálního databázového jazyka PL/SQL. Tento systém je pro svou přenositelnost, výkonnost, snadnou konfigurovatelnost a v neposlední řadě také finanční dostupnost tím nejvhodnějším výběrem pro nový informační portál.

Jelikož budou nový systém využívat jen zaměstnanci, zhruba do 20 lidí, bude webový i databázový server na jednom počítači.

[8] Na základě výzkumu společnosti Netcraft z roku 2007.

4 Řešení informačního systému

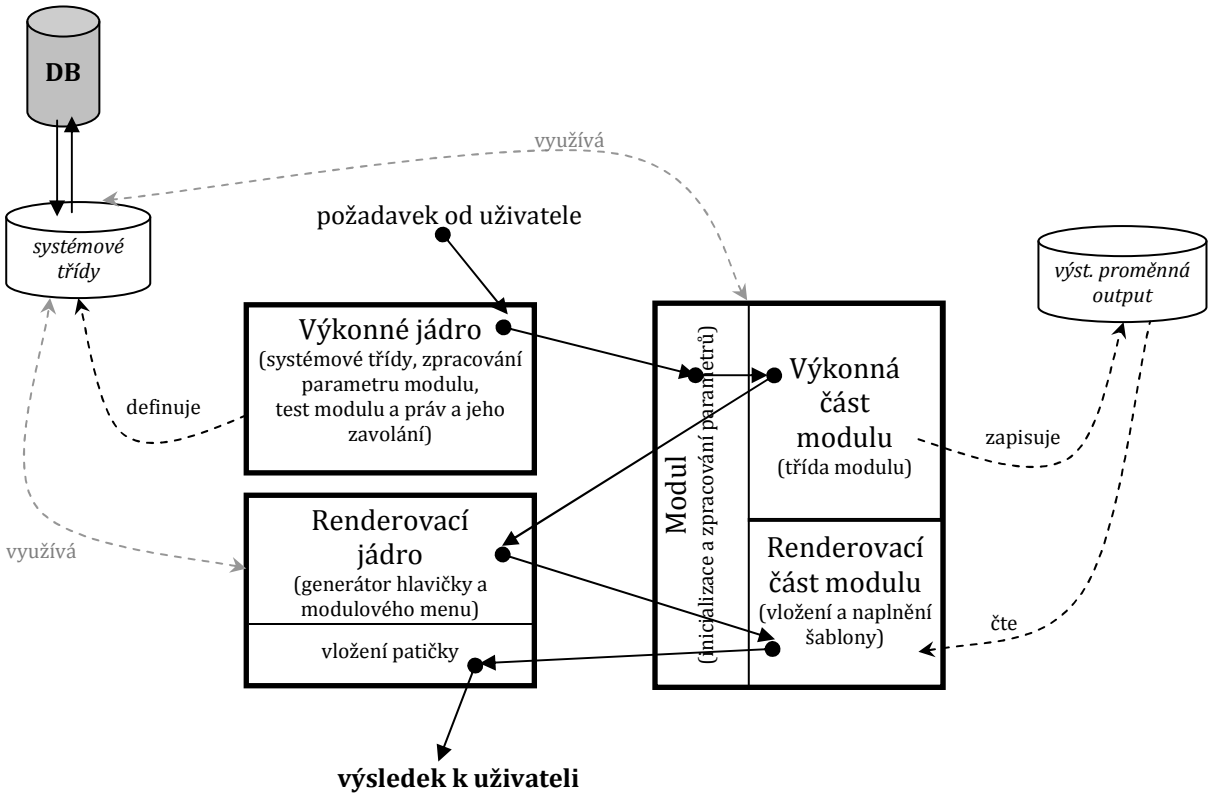
Základními stavebními kameny nového informačního systému, který byl nazván ADDIS (ADDurre Informační Systém), jsou databáze, výkonné jádro, renderovací jádro a samotné moduly. Jádra jsou volána spouštěcím skriptem *index.php*, který po úvodní kontrole přítomnosti spouštěcích souborů nejdřív zavolá jádro výkonné a poté renderovací.

Výkonné jádro vyřizuje všechny požadavky uživatele. Stará se o zavedení systémových tříd, zjišťuje dostupné moduly, testuje zvolený modul (práva přihlášeného uživatele a zda-li jsou všechny soubory modulu v pořádku). Druhá část práce výkonného jádra se odehrává v inicializátoru a řídící třídě modulu, která zpracuje požadavky uživatele a naplní výstupní proměnnou *output*.

Renderovací jádro zpracuje vyhodnocené parametry a výstupní data a vygeneruje výsledný kód na frontend.

Modulárnost systému je důležitá vlastnost pro snadnou implementaci funkčních celků (modulů). Každý modul funguje samostatně, nezávisle na ostatních, ovšem existuje možnost propojení mezi sebou. Moduly pro svou práci využívají systémových tříd výkonného jádra. Dělí se na výkonnou část, která spolupracuje s výkonným jádrem, naplňuje výstupní proměnnou a renderovací jádro, které naopak z této proměnné čte.

Průběh práce a předávání řízení aplikace je naznačeno na schématu [Obrázek 2].



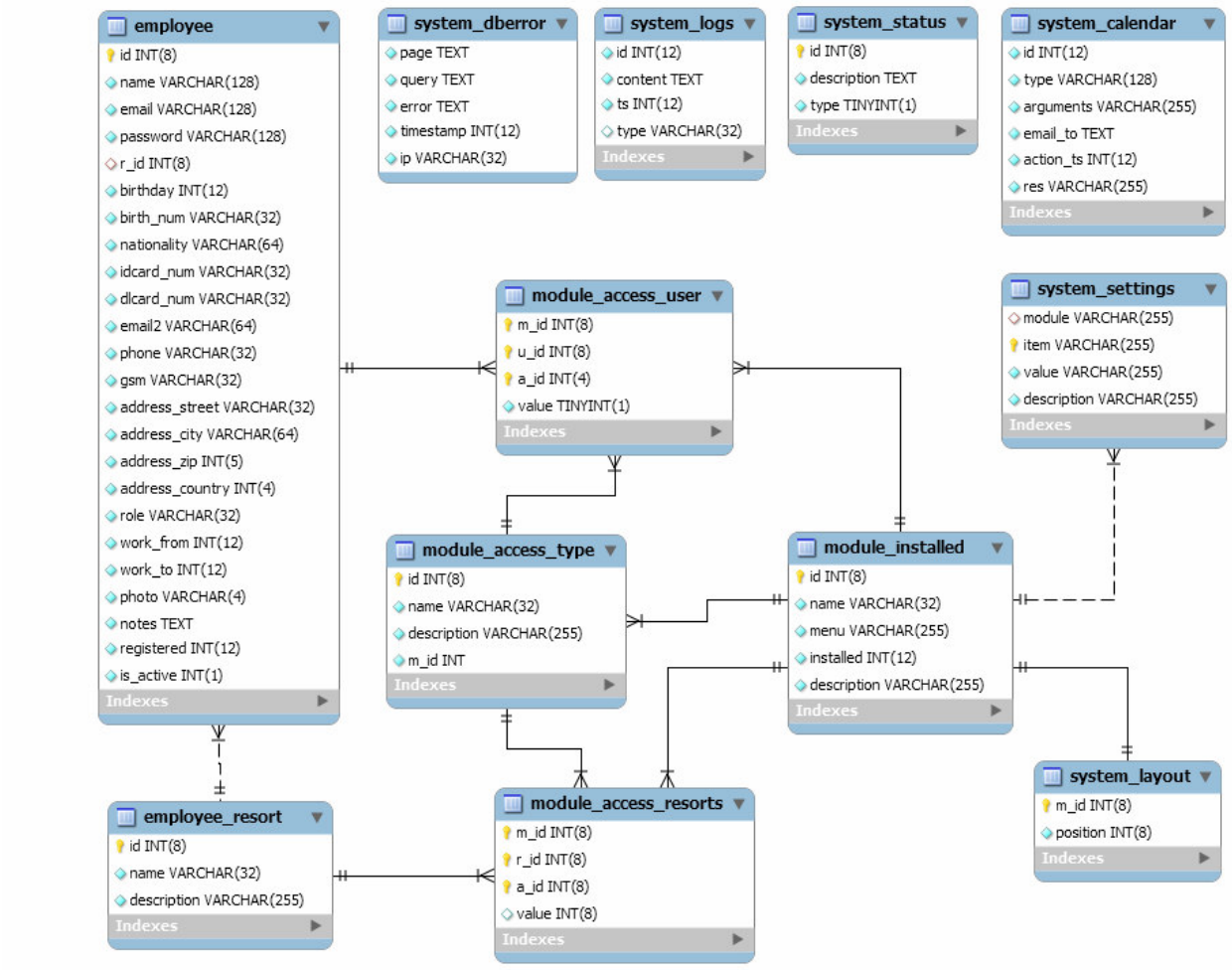
Obrázek 2 – Schéma práce systému

Před samotnou implementací systému bylo však nejdříve důležité navrhnout databázový model. S databází pracuje jen *výkonné jádro*.

4.1 Databázový model

Databáze je rozdělena na dvě jednotky. Jedna je systémová, která řídí systém jako takový – obsahuje informace o uživatelích, modulech, systémových nastaveních a operacích. Každý modul pak má svůj vlastní systém databázových tabulek, které ve většině případů tahají data ze systémových. Rovněž moduly mohou mít vhodně propojené tabulky.

Následuje přehled systémových tabulek vysvětlující ER diagram [Obrázek 3].



Obrázek 3 – ER diagram databáze systému

- Tabulka *employee* obsahuje informace o uživatelích systému, včetně MD5 hashe[9] jejich hesel a definici oddělení, do kterého patří.

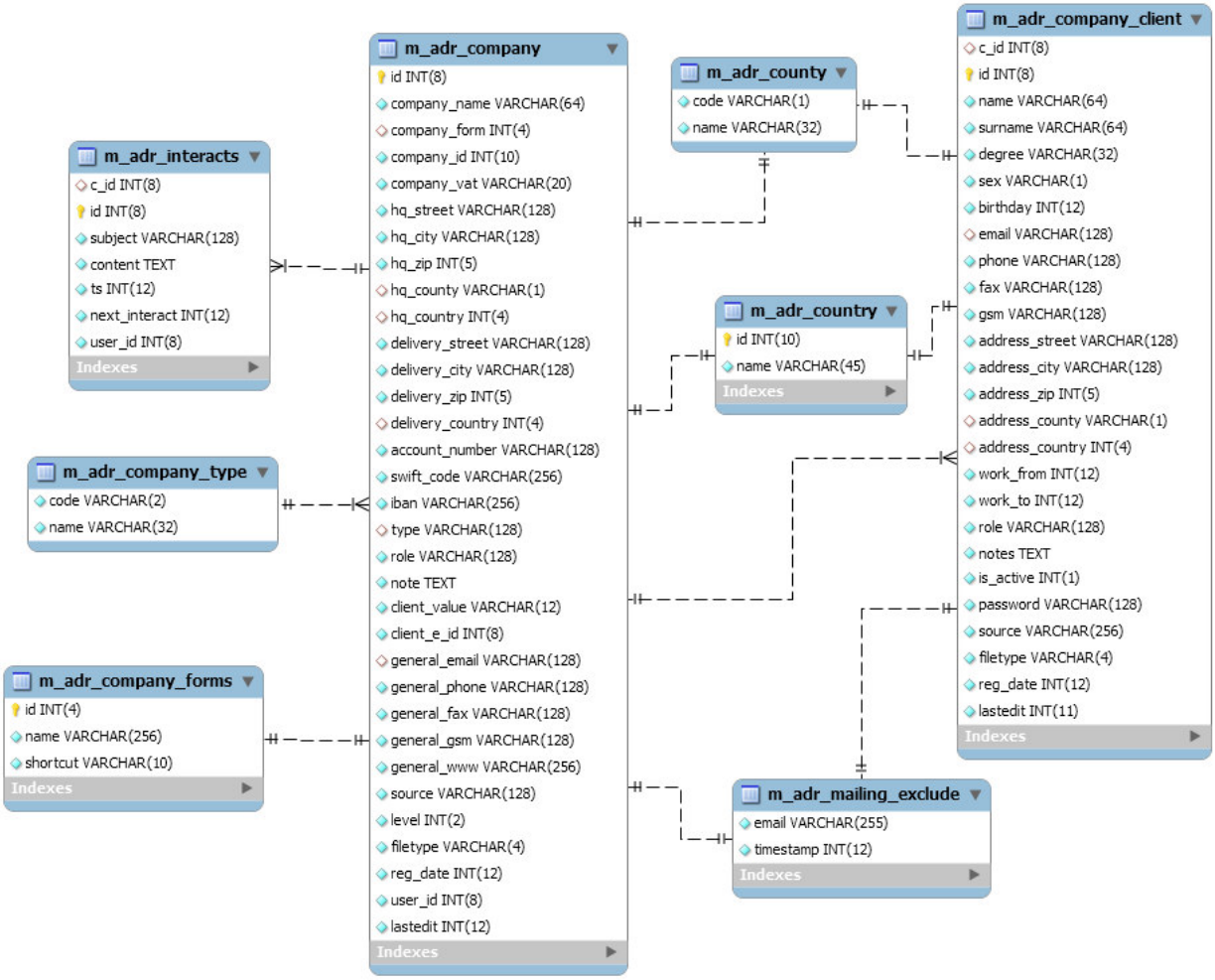
[9] MD5, šifrovací algoritmus Message-Digest, vytváří 32bitový hexadecimální otisk zadaného řetězce

- Tabulka *employee_resort* obsahuje informace o odděleních. Do jednoho oddělení může samozřejmě patřit více uživatelů.
- Tabulka *module_installed* obsahuje informace o nainstalovaných modulech, jejich slovního i číselného identifikátoru.
- Tabulka *system_layout* je jen pomocnou tabulkou k modulům, která určuje pořadí nabídky modulů v menu.
- Tabulky *module_access_user* a *module_access_resorts* obsahují informace o právech jednotlivých uživatelů, resp. oddělení. Identifikátor uživatele, resp. oddělení, identifikátor modulu, společně s cizím identifikátorem typu práva z tabulky *module_access_type* tvoří unikátní kombinaci klíče pro právo dané akce. Typy práv můžou být zadány globálně (typ práva pro všechny moduly, např. právo přidávání položek do DB), nebo specificky pro modul – v tom případě je zadán identifikátor modulu *m_id*.
- Tabulka *system_settings* obsahuje nastavení aplikace. Rovněž je možné specifikovat, ke kterému modulu se hodnota nastavení vztahuje.
- Tabulka *system_dberror* slouží k uchovávání databázových chyb.
- Tabulky *system_logs* slouží k zapisování vybraných systémových operací (loginy, informace o odeslaných systémových e-mailech, nespecifikované výjimky systému)
- Tabulka *system_status* obsahuje popis stavů operací, společně s identifikátorem, jestli je typ operace úspěšný nebo neúspěšný.
- Do tabulky *system_calendar* si systém, potažmo moduly, ukládají plánované akce, o kterých informuje uživatele automatickými e-mailovými zprávami.

Jak bylo naznačeno výše, moduly mají svou vlastní databázovou strukturu; pravidlo pro větší přehlednost zní, že názvy tabulek daného modulu začínají sekvencí „*m_XXX*“, kde XXX jsou první tři písmena slovního identifikátoru modulu. Z tabulek jednoho modulu můžou číst i jiné moduly.

Pro praktickou ukázku byla vybrána struktura databáze modulu adresáře [Obrázek 4].

Data o firmách se ukládají do tabulky *m_adr_company*, které odkazují do systémových tabulek *employee* klíči *user_id* (uživatel zadávající kontakt) a *client_e_id* (zaměstnanec, který má kontakt na starosti). Ke každé firmě lze přiřadit kontaktní osoby a interakce. Kontakt může mít vícero typů (klient/potenciální klient/dodavatel aj.), specifikace typů jsou uvedeny v tabulce *m_adr_company_type* a v tabulce firem se ukládá do kolonky *type* – jednotlivé typy se oddělují znakem zavináče.



Obrázek 4 - Ukázka struktury databáze modulu adresáře

Data tabulek celé databáze jsou uloženy v kódování *utf8_czech_ci*.

4.2 Výkonné jádro aplikace

Jádro celé aplikace je uloženo v adresáři *core*, hlavními skripty pro běh jádra jsou *init.php* a *core.php*.

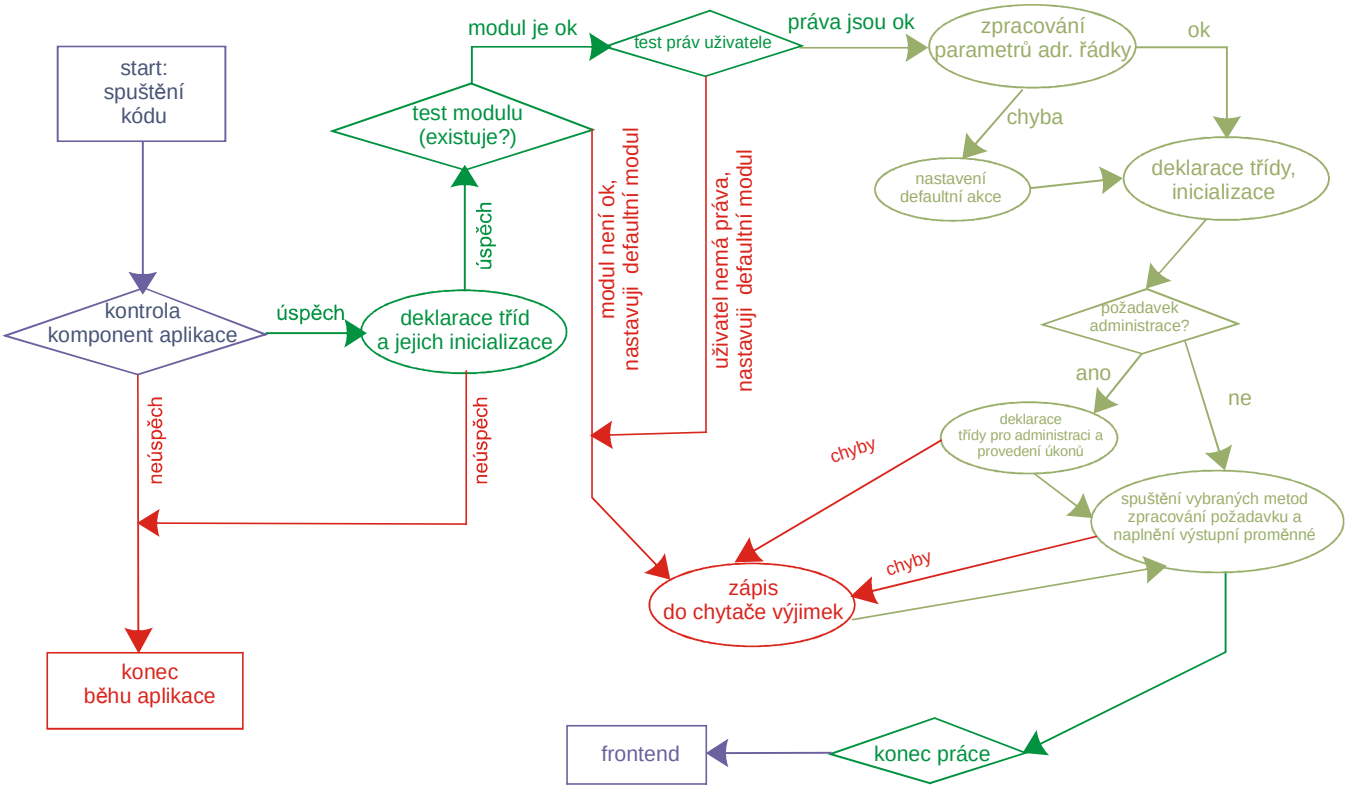
4.2.1 Logika výkonného jádra

Jádro pracuje v několika fázích.

V první fázi se volá skript *core.php*, který deklaruje systémové konstanty a systémové třídy. Předává práci inicializátoru jádra, který tyto systémové třídy naplní a připraví k práci. Skript dále zpracuje povinný parametr *module* z adresové řádky – pokud neexistuje, nastaví defaultní modul, jímž je hlavní stránka. Je-li modul nastaven, zkontroluje přístupová práva a otestuje připravenost modulu.

V druhé fázi se volá inicializátor modulu, který deklaruje a inicializuje modulovou třídu. Zpracovává povinný parametr *action* (není-li nastaven, nastaví se defaultní akce, která je různá u všech modulů) a další zadané parametry. Sled těchto operací poté určí, které metody z modulové třídy budou spuštěny. Modul pak vykoná svou práci – zpracuje uživatelovy požadavky a ukládá výsledky své práce do výstupní proměnné typu pole *output*.

Příklad práce výkonného jádra na diagramu [Obrázek 5].



Obrázek 5 – Diagram práce výkonného jádra

4.2.2 Systémové třídy

Systémové třídy řídí systémové operace a poskytují podporu modulům. Inicializují se a naplňují při spuštění aplikace. Jejich metody slouží k uskutečnění databázových operací, hlídání práv uživatelů, zachytávání výjimek a stavů operací, parsování nastavení a základní operace s moduly.

4.2.2.1 Třída pro práci s databází a nastavení připojení

Třída s databázovými operaci *Database* zařizuje komunikaci s databází. Konstruktor databáze vyžaduje vyparsované hodnoty nastavení připojení, kterou poskytuje jiná třída. Nastavení připojení k databázi je uloženo v souboru XML a má pevně danou strukturu [Kód 1].

Třída je schopná udržovat spojení s více databázemi naráz v jeden moment, pro různá připojení je pak definováno více XML souborů s nastavením.

Metody této třídy využívají prakticky všechny moduly, i jiné systémové třídy. Třída si ukládá poslední dotaz na databázi i poslední výsledek databázové operace, lze tak zpětně přistoupit k poslednímu výsledku, bez nutnosti znovu vykonávat dotaz. Mezi důležité metody patří *query*, konající klasický dotaz na databázi, *query_num* která zjišťuje počet vrácených řádků posledního dotazu a *query_result*, která vrací určitou hodnotu z databáze. Dotazy jsou před samotným posláním ošetřeny proti SQL injection (pokusům o databázový hack) a nahradí klíčová slova v dotazu aktuálními hodnotami (např. klíčové slovo CURRENT_TIMESTAMP se nahradí aktuálním unixovským timestampem).

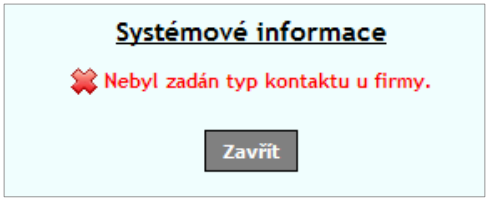
Třída také zkoumá, zda-li nevrátila MySQL databáze nějakou chybu – v takovém případě volá třídu na zpracování těchto chyb.

```
<?xml version='1.0' standalone='yes'?>
<database>
	<hostname>host</hostname>
	<username>user</username>
	<password>pass</password>
	<database>db_name</database>
	<encoding>utf8</encoding>
</database>
```

Kód 1 –XML s nastavením připojení k databázi

4.2.2.2 Třída zachytávání výjimek a operačních stavů

Zajímavou třídou je *Status_Exception_Catcher*; zpracovává veškeré chyby v komunikaci s databází a také výsledky operačních stavů. Veškeré tyto operace mají v systému přiřazený svůj návratový kód. Tyto kódy třída zpracuje a přidá do interního seznamu výjimek. Na konci práce celé aplikace se tyto výjimky metodou *writeEvents* objeví na frontend, aby měl uživatel přehled o tom, co se v systému událo [Obrázek 6].



Obrázek 6 – ukázka výsledku práce třídy Status_Exception_Catcher na frontendu

4.2.2.3 Třída operující s nastavením

Třída *Settings* si při inicializaci nahraje z databáze do interní proměnné veškeré nastavení. K těmto nastavením poté můžou přistupovat ostatní třídy a moduly pomocí metody *get*. Typickým globálním nastavením je např. URI systému, která je potřebná pro atribut *src* u obrázků, nebo atributu *href* u odkazů.

4.2.2.4 Třída práv

Třída *Rights* je pro práci se systémem velmi důležitá. Jak již bylo dříve naznačeno, vedení chce mít plnou kontrolu nad právy oddělení a jednotlivých uživatelů. Konstruktor třídy si zavolá interní metodu *load*, která nejdříve zjistí, zda-li je přihlášený uživatel.

Není-li uživatel přihlášen, načítá jen moduly, ke kterým byl zvolen veřejný přístup. V opačném případě nahraje z *tabulky práv oddělení* do veřejné proměnné práv všechna zadaná práva oddělení, do kterého uživatel patří. Poté se podívá do *tabulky práv uživatelů* a jsou-li specifikována dodatečná práva, přepíše jimi založenou veřejnou proměnnou. K této proměnné poté přistupují moduly a administrační skripty vždy, když je třeba zkontrolovat, zda-li má uživatel k dané akci právo.

4.2.2.5 Třída správy modulů

O kontrolu a výpis modulů se stará třída *Modules*. Konstruktor třídy nejdříve do své interní proměnné nahraje všechny nainstalované moduly. Metoda *createMenu* poté vypíše nabídku modulů na frontend. Metoda *test* testuje, zda-li jsou přítomny všechny nezbytné soubory zvoleného modulu a zda-li má aktuální uživatel k modulu přístup.

4.3 Renderovací jádro aplikace

Po ukončení práce výkonného jádra spouští *index.php* zobrazovací, které vykreslí výsledky operace na frontend. Uloženo je v adresáři frontend, hlavním skriptem je *render.php*. Jádro generuje validní XHTML kód za podpory CSS v kódování UTF-8.

4.3.1 Logika renderovacího jádra

V první části práce renderovací jádro nejdříve samo vykreslí HTML hlavičku. V ní si vezme z výstupní proměnné *output* název stránky, skládající se z názvu aplikace, aktuálního modulu a aktuální stránky v modulu. Dále pomocí metody *createMenu* třídy *Modules* vykreslí menu modulů.

Poté zobrazí lokální menu dané stránky. Používá k tomu zpracované parametry, podle kterých zobrazí nabídku akcí či možnosti zobrazení dané stránky. Společně s tímto menu zobrazí i aktuální agendu a přihlášeného uživatele. Ve spodní části panelu s lokálním menu vypíše práva pro aktuálního uživatele v aktuálním modulu.

Dále vykresluje hlavní obsah. Opět používá zpracované parametry a zkoumá, jestli je pro požadovanou stránku připravená šablona – každá akce musí mít připravenou šablonu, v opačném případě vypíše chybovou hlášku [Kód 2].

```
switch(parametr_akce) {

    case ‘A‘:
      // definice šablon pro danou akci
      možnéObjekty = pole(’XX’,’YY’);

      if parametr_stranky je v poli možnéObjekty
        // existuje-li definovaná akce a šablona pro ni, vlož a naplň ji
        vložŠablonu();
      else
        // existuje-li definovaná akce, ale ne šablona pro ni, vypiš chybu
       tiskni ’Neexistující stránka’;
       vložVýjimku();

    break;

    default:
        // neexistuje-li definovaná akce zobraz chybu
        tiskni ’Neexistující akce’;
        vložVýjimku();
    break;
}
```

Kód 2 – Ukázka determinace výběr šablony (pseudokód)

V poslední fázi se zobrazí patička, s odkazem na nápovědu pro daný modul a uzavře se HTML struktura stránky.

Grafické rozložení prvků na webové stránce ukazuje [Obrázek 7].

Obrázek 7 – Vzhled a rozvržení prvků na stránce

5 Implementace

V této kapitole se pojednává o detailech implementace. Bude představena architektura modulů, řízení přístupu uživatelů, architektura javascriptů, import dat z Varie a detailně popsány vybrané moduly včetně práce s e-shopem.

5.1 Architektura modulů

Moduly tvoří funkční celky systému. Pro větší přehlednost a snadné programování dalších funkcí v budoucnu jsou moduly implementovány zvlášť, každý do svého adresáře. Každý modul pak má i své tabulky v databázi. Moduly plně využívají nabízených systémových tříd. Programátorovi k implementaci nového modulu tedy postačí základní znalost metod těchto tříd pro práci s databází, právy a chytačem výjimek a dodržet několik zásad při programování.

Základem je modulová třída *main.php*, která obsahuje veškeré metody zpracovávající databázové operace. Všechny výstupy by měly směřovat do public proměnné *output*, která je typu pole. Dále naprogramuje inicializátor modulu *init.php*, který zpracuje adresovou řádku, nastaví lokální proměnné a inicializuje třídu. Povinný je také skript *menu.php*, obsahující definice lokálních menu ke každé možné stránce v modulu. Nezbytnou součástí je také definice frontendu ve skriptu *frontend.php*. V ní se podle zpracovaných parametrů volají šablony jednotlivých stránek. Šablony stránek tohoto modulu umístí do adresáře *pages*. Podle potřeby také doimplementuje administrační třídu *admin.php*, která vyřizuje všechny požadavky z formulářů.

Všechny soubory nahraje programátor do adresáře *modules/identifikátor*. Slovní identifikátor modulu by měl být bez diakritiky a mezer (např. *address_book* pro adresář). V databázi pak programátor vytvoří potřebné tabulky pro svůj modul.

V administraci ADDISu už jen jednoduše tento modul nainstaluje a okamžitě může přidávat či ubírat práva jednotlivým oddělením a uživatelům. Stejně pak lze modul odstranit tak, že jej v administraci jednoduše deaktivuje, nebo přímo odinstaluje.

5.2 Řízení přístupu

Po vstupu na hlavní stránku aplikace se návštěvník musí do systému přihlásit. Zadá-li správné uživatelské jméno (resp. e-mail) a heslo, založí ADDIS *sessions*[8], do kterého si načte data o uživateli – jeho jméno, identifikátor, e-mail a číslo oddělení. Znovu zavolá metodu pro načítání

práv, jelikož samotná kontrola uživatele probíhá až po inicializaci systémové třídy *Rights*, ve které se tím pádem nacházejí neaktuální data.

Práva se načítají při každém požadavku na aplikaci – neukládají se do sessions, aby se případná simultánní změna práv administrátorem mohla okamžitě projevit.

Při delší nečinnosti (na serveru je nastaveno 10 minut) systém automaticky ruší dané sezení a je třeba se znovu přihlásit. Zrušit sezení lze i ručně, a to odhlášením ze systému opět na úvodní stránce.

Od původního záměru filtrovat přístupy pro přihlášené uživatele i podle místa připojení bylo nakonec upuštěno, jelikož byl navržen dostačující systém řízení práv.

5.3 Import dat ze systému Varia

Ekonomický software Varia ukládá svá data v centrálním úložišti na firemním serveru. Pro ukládání dat používá databázový souborový systém dBase; tabulky tedy mají formát souboru DBF s připojenými FPT soubory, které obsahují poznámky.[9]

Data v dBase formátu mohou nabývat několika typů – typ *integer* reprezentuje 4bajtové celé číslo; typ *float* reprezentuje desetinné (až) 20bajtové číslo; typ *character* obsahuje až 254bajtový řetězec; typ *memo* obsahuje pointer na blok v souboru FPT, kde je uložena textová poznámka libovolné délky; typ *logical*, který obsahuje informaci typu TRUE/FALSE, a typ *date* či *datetime*, který obsahuje časový údaj. V systému dBase, zvláště v jeho novějších verzích, existuje celá řada dalších datových typů, v systému Varia jsou ovšem využity jen výše zmíněné.

Při manipulaci s tabulkami si program Varia, nainstalovaný u zaměstnanců na pracovních stanicích, potřebnou tabulku vždy stáhne k sobě, změní a nahraje zpět na server. Varia tedy nefunguje jako standardní server-klient aplikace.

Natahování dat ze systému Varia do ADDISu probíhá dvěma fázemi. První fáze je záležitostí serveru. Jelikož se data systému Varia nacházejí na témže počítači jako systém ADDIS, je na něm nastavena naplánována úloha (cron), která každých *deset minut* zkopíruje DBF a FPT soubory do adresáře *files/tmp* aplikace ADDIS. Poté se spustí druhý skript, který zkoumá tabulku DBF a složí SQL dotaz, kterým vytvoří přesnou kopii do databáze ADDISu.

5.3.1 Proces vytváření kopie DBF a FPT do SQL tabulky

V základním balíku PHP existuje třída funkcí pro práci s *dBase* formátem. Nicméně nedokáže pracovat s FPT soubory, které obsahují obsah položek typu *memo* a navíc tyto položky při výpisu úplně ignoruje. Byla tedy vytvořena speciální metoda, která načte celý obsah souboru DBF do proměnné *dbaseFile* a tuto pak čte postupně po bajtech.

V hlavičce DBF souboru [Tabulka 1] určuje první bajt typ tabulky a zda-li existuje připojený FPT soubor pro poznámky. Je-li tento bajt nastaven na přítomnost FPT souboru, tak se načte do proměnné *memoFile* jeho obsah. Význam pro převod má dále čtvrtý až sedmý bajt, který nás informuje o počtu položek v tabulce.

B	0	1	2	3	4	5	6	7
0	Typ DBF (1 B)	Datum posledního update tabulky ve fomátu YYMMDD (3 B)			Počet položek v tabulce (4 B)			
8	....							

Tabulka 1 – Struktura DBF hlavičky (prvních 8 bajtů)

Od 68. bajtu hlavičky pak následují 48bajtové bloky, které obsahují informace o jednotlivých sloupcích tabulky. Jejich struktura je naznačena v [Tabulka 2]. Kromě názvu sloupce je pro nás důležitý datový typ sloupce, délka hodnoty v bajtech v případě typu *character* a počet číslic v případě typu *integer*. Kopírovací skript z těchto informací složí SQL dotaz na vytvoření tabulky.

B	0	1	2	3	4	5	6	7
0	Název sloupce (32 B)							
8								
16								
24								
32	Datový typ (1 B)	Délka (1 B)	Počet dec. čísel (1 B)	Neobsazeno (2 B)		Indexace MDX (1 B)	Neobsazeno (2 B)	
40	Další autoinkrement (4 B)				Neobsazeno (4 B)			

Tabulka 2 – Struktura popisovače sloupců

Samotná data následují hned za hlavičkou. Každému řádku předchází vždy jeden bajt; ten může být nastaven na 0x2A (hvězdičku) v případě, že je řádek smazán, jinak obsahuje vždy hodnotu 0x20 (mezeru). Smazané řádky se do zkopírované SQL tabulky vůbec nedostanou. Kopírovací skript čte postupně všechny buňky všech řádků, provede příslušnou konverzi kódování (Varia používá kódování Windows-1250[10], před vložením do SQL použije skript funkci *iconv* na konverzi do UTF-8) a datových typů (z *date* na *unixovský timestamp*). V případě, že skript narazí na buňku datového typu *memo*, musí najít její obsah v připojeném souboru.

K načtení položky typu *memo* z externího souboru FPT je třeba znát hodnotu buňky, která odkazuje na externí soubor, tzv. pointer na blok. Pointer je 4bajtové číslo, uložené v Little-

[10] Poznámka: Informaci o kódování nalezneme také v hlavičce DBF souboru, a to na 29. bajtu. Jelikož však tuto informaci známe z dřívějšího studia struktury tabulek Varie, nemusíme ji skriptem znova zkoumat.

Endian notaci a označuje pořadové číslo bloku, na kterém poznámka začíná v FPT souboru. [10] Každá poznámka má svou hlavičku, která na prvních čtyřech bajtech informuje o typu poznámky (text/obrázek) a na posledních čtyřech bajtech informuje o její délce (v notaci Big-Endian). Od 9. bajtu pak následuje samotný obsah poznámky. Standardně má jeden blok velikost 512 bajtů, obsah poznámky se však může rozkládat na více blocích. Nevyužitý zbytek bloku je vyplněn nulami. Informace o nejbližším volném bloku jsou uloženy v hlavičce FPT souboru. Kopírovací skript přečte celý blok, resp. více bloků s poznámkou, obsah převede do UTF-8 a vrací se do DBF tabulky.

Celý proces tak vytvoří dva dotazy na databázi – jeden smaže starou tabulku a vytvoří novou a druhý ji naplní. Pro přehlednost mají tabulky z Varie v MySQL název *m_var_YYY*, kde YYY je název tabulky Varie. Kopírovací proces se vykoná pro tabulky faktur vydaných *(fv.dbf),* položek faktur *(obraty.dbf),* zakázek *(zakazky.dbf),* položek zakázek *(za_hod.dbf),* vydaných objednávek *(objednav.dbf)* a položek vydaných objednávek *(obratyo.dbf)* s jejich příslušnými FPT soubory.

Pro úplnost je třeba dodat, že se v databázi ADDISu uchovávají stará data z minulých let – z důvodu zachování historie jednotlivých agend. Tyto tabulky byly importovány jednorázově, mají příponu příslušného letopočtu, např. *m_var_fv_2008*, a jsou v databázi nastálo, jejich obsah se už neaktualizuje.

Moduly můžou k těmto hotovým tabulkám volně přistupovat. Vždy z nich však jen čtou, neboť při každé aktualizaci se obsah zkopírovaných tabulek maže. Doplňující data k jednotlivým dokumentům z Varie si ukládají do svých vlastních tabulek, pro lepší identifikaci a propojení s nimi většinou používají nějaký klíč. (Např. u faktur nebo zakázek to jsou jejich čísla, která jsou vždy unikátní.)

5.4 Implementace dokumentového skladu

Modul imituje standardní souborový systém, který je udržován čistě na databázové úrovni. Šetří se tím hlavně systémové prostředky a zefektivňuje se tím proces operace s adresáři a soubory, včetně vyhledávání, které je velmi rychlé.

5.4.1 Manipulace s adresáři

Adresář představuje jeden řádek v tabulce *m_dok_folders*. Obsahuje svůj identifikátor (id), id nadřazeného adresáře (v případě, že je v kořeni, pak je tento id roven nule) a také, v jaké úrovni se adresář nachází.

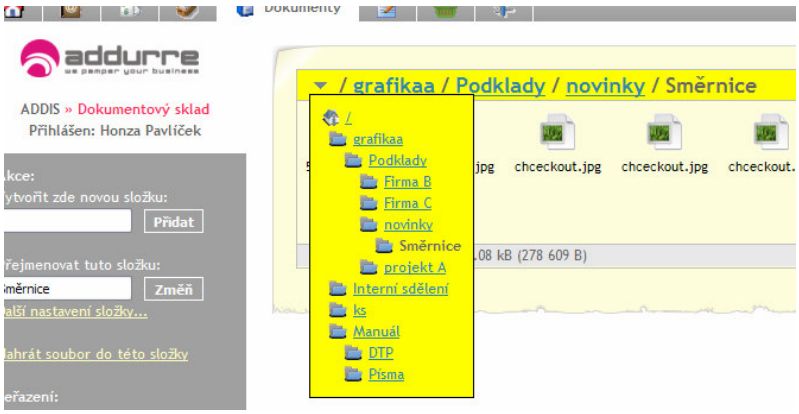
V případě, že uživatel s patřičnými právy zamezí přístup do složky nějakému oddělení, vloží se do tabulky *m_dok_folders_access_forbid* řádek s informací o tom, které oddělení má

přístup zakázán, id zakázaného adresáře a id adresáře, u kterého je toto omezení nastaveno. Poslední zmíněná informace je důležitá pro případ, že omezovaný adresář obsahuje další podadresáře (libovolně zanořené) – omezení totiž automaticky platí i na ně a musíme tedy tuto informaci zaznamenat do databáze. V tabulce tedy bude tolik řádků, pro kolik oddělení toto omezení platí násobeno počtem podadresářů. Pokud tedy omezujeme adresář, ve kterém je dalších 10 různě zanořených podadresářů, a to pro 4 oddělení, bude v tabulce celkem 4 + 10×4 řádků, tj. 44 záznamů.

Omezením adresáře se zamkne i jeho celková vnitřní struktura. Není tedy možné přesunovat zanořené složky a jednotlivým podadresářům nelze měnit přístupová práva, jelikož jsou podřízena adresáři, u kterého toto omezení bylo nastaveno.

Algoritmus, který prochází jednotlivé složky si udržuje informaci o nadřazených adresářích, ví tedy zpětně o všech rodičích a jejich omezeních. Průchod stromem se děje rekurzivně v privátní metodě *_walk* – té je zaslán seznam všech adresářů, které jsou rodiči; ty poté prochází a sestavuje strom do výstupní proměnné. Díky uchovávání „historie rodičovství" má metoda k dispozici okamžitou informaci, jestli se daná položka nenachází pod zakázaným adresářem pro aktuálního uživatele – takový adresář pak do stromu vůbec nezapisuje.

Algoritmus lze zhodnotit jako velmi efektivní a rychlý, dotaz na databázi je totiž proveden za celou dobu jen jednou a zbytek procesu probíhá ve zmíněné metodě. Hotový strom posléze může na frontendu vypadat jako na obrázku [Obrázek 8]. Na obrazovce se objeví aktuálně otevřená cesta ke složce, po kliknutí na šipku se objeví adresářový strom. Opětovným kliknutím na šipku se strom zavře.



Obrázek 8 – Ukázka zobrazení adresářového stromu a obsahu adresáře

5.4.2 Manipulace se soubory

Všechny soubory, které jsou do systému vkládány, včetně jejich revizí, jsou uloženy v jediném adresáři informačního systému – *files/docs*. Každý nový soubor po nahrání získává jednoduchým algoritmem [Kód 3] unikátní 40znakový hexadecimální hash, pod kterým je následně uložen v uvedeném adresáři. Systém podle přípony souboru určí jeho typ ze své

vlastní databáze – v základní verzi zná systém 29 přípon, resp. typů souborů, které jsou ve společnosti nejčastěji využívány. Dojde-li k pokusu o nahrání souboru, který není v seznamu přípon, odmítne systém tento soubor nahrát.

Revize souboru jsou ukládány pod stejným zahashovaným jménem jako originální soubor, akorát s dodatkem čísla revize (např. *„...ab3_r1“*). V databázi se uchovávají veškeré informace o souboru – jejich velikost, typ (podle přípony), druh (podle účelu souboru) datum nahrání a konečně hash, podle kterého systém může k souboru fyzicky přistoupit, je-li třeba.

```
private function _generuj_hash(nazev_souboru) {
	// vygeneruje náhodné čtyřmístné číslo
	náhodnéČíslo = rand(1000,9999);
	// vytvoří pomocí algoritmu sha1 hash ze spojeného stringu skládajícího se
z timestampu, aktuálního času milisekund, názvu souboru a vygenerovaného náh. čísla
	hash = sha1(aktuálníTimestamp.aktuálnímiliSekundy.nazev_souboru.náhodnéČíslo);
	// kontrola, jestli stejný hash už v DB existuje
	if hash je v databázi
		// pokud ano, rekurzivně znovu zavolat generátor
		_generuj_hash(nazev_souboru);
	else
		// v opačném případě vrať hash, pod kterým se soubor uloží
		return hash;
}
```

Kód 3 – Algoritmus generování názvu souboru

V editačním formuláři lze libovolně měnit zařazení, popisek, druh a dokonce i název souboru, který je nabídnut při jeho stažení. Soubor a jeho revize lze stáhnout v detailu souboru. Smazáním souboru se smažou i veškeré jeho revize, jednotlivé revize lze mazat i zvlášť.

Při manipulaci se soubory a výpisu obsahu adresářů se vůbec nepřistupuje k fyzickým souborům, vše probíhá na databázové úrovni až do doby, kdy si chce uživatel soubor fyzicky stáhnout. K tomu slouží metoda *download*, která z databáze složí název souboru, připraví informaci o MIME typu a velikosti souboru pro hlavičku HTTP a až v tento moment přistoupí k fyzickému souboru a nabídne jej ke stažení. Prohlížeč pak zobrazí stahovací dialog, vše se děje bez reloadu stránky.

5.4.3 Schopnost propojování s jinými prvky v systému

Výbornou ukázkou spolupráce s jinými moduly je schopnost propojovat dokumenty s jinými prvky v systému. Je připravena možnost propojení s *kontaktem z adresáře*, *interakcí s vybraným kontaktem* nebo se *zakázkou*. Při nahrávání souboru uživatel jednoduše zvolí z menu druh propojení a systém pomocí ajaxové funkce nahraje konkrétní nabídku (např. seznam kontaktů, nebo zakázek).

Toto propojení se vkládá do tabulky *m_dok_relations*, kde se ukládá informace o id souboru, typu propojení a argument propojení, který většinou obsahuje identifikátor prvku, se

kterým se soubor propojuje. Propojený soubor pak ve svém detailu obsahuje odkaz na propojený prvek a naopak – např. v interakci se objeví seznam propojených souborů.

Při smazání souboru se smaže i propojení s prvkem. Naopak to ale neplatí – je-li smazán prvek, informace o propojení a samotný soubor zůstávají. Ovšem, zjistí-li modul při přípravě zobrazení detailu souboru, že propojení již není aktuální, tak tuto informaci smaže automaticky, k uživateli se tedy tato neaktuálnost vůbec nedostane.

5.5 Implementace správce webu a e-shopu

Jelikož se web i e-shop nacházejí na stejném serveru jako systém ADDIS, je propojení s databázemi jednoduché. Moduly sloužící jako správci webového obsahu si při inicializaci vytváří druhé databázové spojení, resp. vytvoří si nový objekt databázové systémové třídy. Ta pozná, že se nejedná o databázi ADDISu, a tak věnuje pozornost tomu, do které databáze zapisovat interní databázové chyby (chyby, které vrátí databáze e-shopu musí zapsat do databáze ADDISu).

5.5.1 CMS webu

V základním zobrazení tohoto jednoduchého rozhraní správy obsahu webu addurre.cz nabídne uživateli seznam dostupných článků. Články se nacházejí v tabulce *articles* a obsahují informaci o nadpisu, krátkém článku (který se zobrazuje na seznamu článků) a hlavní obsah. Uživatel může okamžitě vytvořit nový článek. Zobrazí se mu dva WYSIWYG[11] editory – použitý engine je FCKEditor2. Do nich napíše krátký článek a obsah. Po odklepnutí se článek okamžitě objeví na webu.

Obdobně to je s editací článku – uživatel s odpovídajícími právy má po kliknutí na detail článku možnost okamžité editace. Změny probíhají okamžitě a jsou zapsány do tabulky ADDISu *m_web_changes* pro pozdější kontrolu – ukládá se zde starý a nový text.

5.5.2 Propojení aplikace s e-shopem

V základním zobrazení zobrazuje modul e-shopu seznam vytvořených objednávek z tabulky *objednavky*. Seznam je standardně řazen od nejnovější, k dispozici jsou různé druhy filtrování či seřazení. V detailu objednávky lze manipulovat se stavem objednávky, ovšem systém je schopen se stavem manipulovat automaticky.

[11] WYSIWYG = What You See Is What You Get editor, při úpravě textu uživatel rovnou vidí výsledek

5.5.2.1 Automatická manipulace s objednávkami, propojení se zakázkou

Založí-li registrovaný klient novou objednávku v e-shopu, přichází informace o objednávce na e-mail příslušnému oddělení. Ti poté ručně vystaví v systému Varia zakázku. Zakázky se automaticky načítají do ADDISu. Při načítání zakázek systém automaticky kontroluje, zda-li nebyla objednávka pořízena z e-shopu a pokud ano, tak ji automaticky spáruje s touto objednávkou. Spárování si uloží do tabulky *m_esh_orders_pair*. Při tomto spárování označí objednávku stavem „Vyřizuje se“. Pokud se spárovaná zakázka dokončí, zapíše si tuto informaci do tabulky a označí objednávku stavem „Vyřízeno“. V seznamu objednávek se poté u čísla webové objednávky objeví i číslo spárované zakázky.

5.5.2.2 Manipulace se zbožím

Na e-shopu nabízí společnost standardní sortimentní zboží, které je rozděleno do pěti katalogů. Produkt na e-shopu je tedy vždy zařazen do konkrétního katalogu a do konkrétní kategorie. Zboží je uloženo v tabulce *katalog_zbozi*, katalogy pak v tabulce *katalog_nazvy*.

Při zobrazení seznamu produktů lze zboží vyhledávat a filtrovat podobně jako na e-shopu. Uživatel má v detailu výrobku možnost editovat veškeré jeho údaje - název, cenu, barvu, zařazení do katalogu, kategorie i podkategorie, typ potisku a klíčová slova. Přidávat a upravovat lze i obrázek k výrobku. Pomocí *GD knihovny*[11] se vytvoří automaticky náhled obrázku. Oba obrázky se nahrají přímo do adresáře e-shopu.

5.5.2.3 Seznam hledaných slov

Při každém hledání zapíše vyhledávač požadovanou frázi do tabulky *search_queries*. ADDIS si tyto fráze seskupuje a sestavuje z ní pro uživatele podrobnou statistiku. Ukázka takového seznamu je na obrázku [Obrázek 9].

Obrázek 9 – Ukázka seznamu hledaných frází na e-shopu v systému ADDIS

Zajímavou funkcí hledacího algoritmu na e-shopu, který byl vytvořen speciálně pro tuto práci, je vyhledání různých tvarů a synonym zadaného slova. Kontaktuje se přitom server http://www.pravidla.cz a http://www.synonyma-online.cz a vyparsuje se jejich odpověď.

E-shop tak hledá více vhodných výrazů a návštěvníci nacházejí více výrobků. I tato statistika se započítává do seznamu hledaných výrazů a zobrazuje se v systému ADDIS.

5.5.2.4 Manipulace s registrovanými uživateli

Registrovaní uživatelé, resp. právnické osoby, se ukládají do tabulky *uzivatele*, každý uživatel má jeden řádek. Do ADDISu se kromě údajů o uživatelích načítají také jejich obraty a statistika navštívených stránek – lze tak jednoduše analyzovat, který druh zboží je zajímá a marketingové oddělení může na základě nasbíraných dat vytvářet cílené nabídky.

Opět platí, že pokud systém nenalezne uživatele v *Adresáři*, nabídne jeho uložení.

5.6 Implementace JS a AJAX nástrojů

Jak již bylo naznačeno v kapitole 3.2.3, je v systému použito několik jednoduchých javascriptových funkcí. Ty jsou uloženy v adresáři *scripts/general* v souboru *scripts.js*.

Pro účely zobrazování a skrývání objektů na základě kliknutí na určitý prvek byla vytvořena funkce *show_unshow*. Té se předávají dva parametry – první je ID objektu, který chceme zobrazit (nastavit CSS vlastnost *display* na *blok*) a druhé je ID objektu, který chceme skrýt (*display* na *none*). Pokud jsou tyto parametry shodné (odkazují na stejný objekt), nastaví opačnou viditelnost požadovaného objektu.

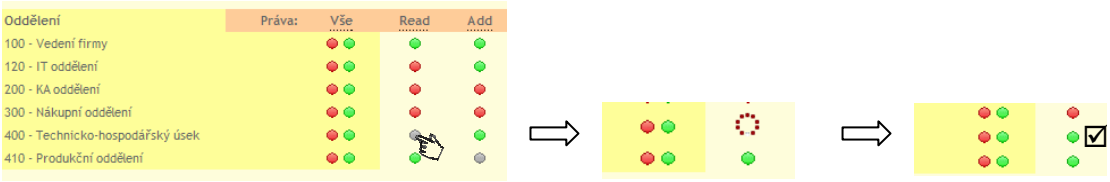
Dalším souborem javascriptových funkcí je handler AJAXových požadavků *ajax.js*. Na základě akce *onclick* či *onchange* u vybraných objektů v systému se asynchronně zašle požadavek na server – zavolá se skript *ajax.php*, uložený v kořenovém adresáři a ten se postará o spuštění příslušného skriptu, který vyhodnotí požadavek. Tyto skripty jsou uchovávány v adresáři *single/ajax/název_funkce/main.php*. Javascriptu se poté vrátí výsledek a ten jej zobrazí do zadaného objektu.

Funkce si pro svou práci volají výkonné jádro, mají tedy plný přístup k systémovým třídám.

5.6.1 AJAXová manipulace s právy

Zajímavou AJAXovou funkcí je manipulátor s právy. Stránka se zadáváním práv obsahuje desítky formulářů, každý formulář v podstatě nastavuje jeden typ práva jednomu uživateli v jednom modulu. Časté reloadování stránky by v tomto případě bylo krajně neefektivní, a proto byl zvolen jiný způsob aktivace práv. Na obrázku [Obrázek 10a] lze vidět ukázku této

stránky – každý puntík je jeden formulář. Po kliknutí na něj se nejdříve zobrazí ikonka průběhu [Obrázek 10b] a asynchronně se zavolá funkce *change_rights*, která zpracuje skryté parametry z formuláře. Výsledek posílá okamžitě zpět a javascript se postará o zápis nového formuláře, tentokrát s možností odebrání daného práva [Obrázek 10c].



Obrázek 10a (vlevo) – ukázka stránky s možností měnit práva

Obrázek 10b (uprostřed) – zpracování požadavku na pozadí bez nutnosti reloadu stránky

Obrázek 10c (vpravo) – zobrazení výsledku operace a nového formuláře

5.6.2 Použité hotové nástroje

V aplikaci jsou použity tři hotové nástroje, všechny jsou uloženy v adresáři *scripts* pod svými názvy.

Prvním z nich je FCKEditor2, který slouží k editaci článků v modulu správy webu. Volá se vložením jeho jádra pro PHP. Pro více informací viz dokumentaci k tomuto nástroji. [11]

Druhým z nástrojů je Greybox, který slouží k zobrazování obrázků a HTML stránek v nenásilné verzi pop-up okna. Chceme-li tento nástroj použít, stačí do příslušného elementu vložit atribut *rel* a klíčové slovo. Pro více informací viz dokumentaci k tomuto nástroji. [12]

Posledním z použitých nástrojů je knihovna Tooltip od Waltera Zorna. Slouží k informování uživatele, či zobrazení mini-nápovědy bublinkou, která se plynule objeví po najetí myši nad inkriminovaný objekt. Výhodou oproti často používanému atributu *title* jsou jeho značně rozšířené možnosti – podbarvení, formátování textu a schopnost zobrazovat v bublince i obrázky. Pro více informací viz dokumentaci k tomuto nástroji. [13]

6 Závěr

Práce měla za cíl zefektivnit procesy ve společnosti ADDURRE s.r.o., poskytnout zaměstnancům společnosti kvalitní informační systém, ve kterém by měli možnost spravovat data z jiných systémů (ekonomický systém Varia, webové stránky a e-shop), propojit jejich informační tok v jeden celek a nabídnout nové možnosti zejména ve správě firemní databáze kontaktů a dokumentů.

Součástí práce bylo i nasazení a instalace systému s databází na firemní server. Musel se také přesunout e-shop a webové stránky na ten samý server. Rovněž proběhlo i zaškolení všech zaměstnanců. Během testovacího období bylo sledováno mj. i využívání naimplementovaných funkčních celků a sběr zpětné vazby od uživatelů. Zejména tato zpětná vazba přispěla k průběžným úpravám jednotlivých funkcí a frontendu.

Po zhruba měsíčním testování v ostrém provozu lze s potěšením konstatovat, že výkonnost zaměstnanců se neporovnatelně zvýšila. ADDIS je využíván každý pracovní den minimálně 2 hodiny každým zaměstnancem[12]. Systém si pochvalují nejen zaměstnanci, kterým ulehčuje práci v řízení projektů a akvizici nových obchodních příležitostí, ale i samotné vedení společnosti, které má okamžitý přehled o činnosti svých podřízených. Vděčnou funkcí se stalo automatické ohlašování průběhu zakázky či plánovaných interakcí s klienty. Uživatelé se díky intuitivnosti systému naučili s ADDISem rychle pracovat.

Závěrem je vhodné podotknout, že tento systém bude po uzavření této práce i nadále vyvíjen a zdokonalován.

[12] Tato informace vyplynula ze záznamů o přihlašování uživatelů a časů jednotlivých akcích prováděných v ADDISu , které tento systém aktivně sleduje a zapisuje do databáze.

7 Literatura

[1] HTML [online]. HyperText Markup Language, *Wikipedia, the free encyclopedia*, dostupné na Internetu <http://en.wikipedia.org/wiki/HTML>.

[2] CSS [online]. Cascading StyleSheets, *Wikipedia, the free encyclopedia*. Dostupné na Internetu <http://en.wikipedia.org/wiki/CSS>.

[3] PHP [online]. PHP: General information. 2001-2009. Dostupné na Internetu: <http://cz.php.net/manual/en/faq.general.php>.

[4] AJAX [online]. Ajax (programming), *Wikipedia, the free encyclopedia*. Dostupné na Internetu <http://en.wikipedia.org/wiki/AJAX_(programming)>.

[5] GPL [online], GNU General Public License, *Wikipedia, the free encyclopedia*. Dostupné na Internetu <http://en.wikipedia.org/wiki/GNU_General_Public_License>.

[6] Web Server Surveys December, 2007 [online], Netcraft. Dostupné na Internetu: <http://news.netcraft.com/archives/2007/12/29/december_2007_web_server_survey.html>.

[7] MySQL [online]. MySQL: General information. 1995-2009. Dostupné na Internetu: <http://dev.mysql.com/doc/refman/5.0/en/introduction.html>.

[8] Sessions [online]. PHP: Sessions. Dostupné na Internetu: <http://cz.php.net/session>.

[9] DBF [online]. Xbase File Format Description Dostupné na Internetu: <http://www.clicketyclick.dk/databases/xbase/format/index.html>.

[10] FPT [online]. Visual FoxPro Reference - Memo File Structure (.FPT). Dostupné na Internetu: <http://msdn.microsoft.com/en-us/library/8599s21w(VS.71).aspx>.

[11] GD FUNCTIONS [on-line]. PHP : GD Functions. Dostupné na Internetu: <http://cz.php.net/manual/en/ref.image.php>.

[12] FCKEditor Docs [online]. Dostupé na Internetu: <http://docs.fckeditor.net/>.

[13] Greybox Online Documentation [online], Orangoo Labs. Dostupné online: <http://orangoo.com/labs/GreyBox/Documentation/>.

[14] Tooltip [online], DHTML and Javascript Libraries. Dostupné na Internetu: <http://www.walterzorn.com/tooltip/tooltip_e.htm>.

Příloha A – Manuál k instalaci

V této příloze je popsán stručný návod k instalaci systému na webový a databázový server. Na CD se nachází verze systému 0.9.0519 ze dne 19.5.2009.

1. Na přiloženém CD se nachází adresář *www* obsahující zdrojové kódy aplikace. Celý adresář se umístí na webový server.
2. Běží-li webový server na OS Linux je třeba nastavit zapisovací práva ke složce *files*, *files/tmp* a *files/docs* příkazem *chmod 0777*.
3. Na databázovém stroji se vytvoří databáze pro systém – porovnávání nastavit na *utf8_czech*_ci.
4. Na CD se v adresáři *database* nachází soubor *structure.sql*. V něm je série dotazů na databázi, která vytvoří strukturu tabulek. Obsah tohoto skriptu se spustí v administračním rozhraní MySQL na vytvořenou databázi.
5. Dále se v adresáři *database* nachází soubor *sample-data.sql*. V něm je třeba upravit hodnotu nastavení pro URI adresu systému; v souboru se vyhledá řetězec URI_ADDRESS a nahradí se za plnou adresu systému s lomítkem na konci (např. *http://addis.addurre.cz/*) a uloží. Skript se poté spustí na databázi.
6. Posledním krokem je nastavení připojení k databázi. To se nachází v souboru *db_connection.xml* v adresáři *core/config*. Jsou-li potřeba moduly pro práci s webem addurre.cz a reklamni-predmety.name, je třeba nastavit také připojení k databázím těchto webů do odpovídajících souborů.
7. Do systému se dá nyní přihlásit pomocí už. jména *admin(@addurre.cz)* a s heslem *admin*. Tento uživatel má plný přístup k administraci systému.
8. Pro kopírování tabulek ze systému Varia je nutné na serveru nastavit automatickou úlohu, která kopíruje soubory, popsané v kapitole 5.3.1 do složky *files/tmp* a následné automatické spouštění skriptu *cron_varia.php* v kořenovém adresáři. Rovněž je vhodné nastavit automatické spouštění kalendáře v *cron_calendar.php* jednou denně vždy na 7. hodinu ranní.

Příloha B – CD se zdrojovými kódy

Přiložené CD v obálce obsahuje:

- adresář *www* se zdrojovými kódy systému, soubory se styly a veškerými grafickými prvky se nacházejí v adresáři *www/styles*.
- adresář *database* se skripty pro vytvoření databáze a naplnění vzorkem dat pro spuštění aplikace
- adresář *docs* obsahující tento dokument v elektronické podobě a manuál pro instalaci v PDF