TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 - Elektrotechnika a informatika

Studijní obor: 1802T007 - Informační technologie

Systém online zpětné vazby uživatelů pro .NET a PHP aplikaci

Online system for user feedback .NET and PHP application

Diplomová práce

Autor: **Bc. Jan Paulík**

Vedoucí práce: Ing. Jan Kraus, Ph.D.

Konzultant: Ing. Pavel Štěpán

V Liberci 10. 5. 2013

Originální zadání

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 - školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, která vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum: 10. 5. 2013

Podpis:

Poděkování

Za odborné vedení mé diplomové práce, za cenné rady a připomínky, které mi byly velkým přínosem, bych rád poděkoval panu Ing. Janu Krausovi, Ph.D. . Poděkování patří také mému konzultantovi, který mi ochotně poskytl podrobné informace. Závěrem děkuji rodičům za finanční a hlavně psychickou podporu po celou dobu studia.

Abstrakt

Tato diplomová práce se zabývá návrhem systému technické podpory pro firmu KMB systems s.r.o.. Hlavní činností společnosti je vývoj produktů pro průmyslovou automatizaci a energetiku.

Systém technické podpory byl rozvržen na dvě aplikace, webovou a desktopovou. V rámci zpracování návrhu byly využity moderní metody programování. Webová aplikace obsahuje uživatelské a administrační prostředí a byla navržena pomocí HTML, PHP a CSS. Desktopová aplikace obsahuje pouze uživatelské rozhraní, které bylo naprogramováno v jazyce C#. Jelikož byly aplikace propojeny prostřednictvím databáze MySQL, byl zaručen přístup k aktuálním informacím. Pomocí přihlašovacího a dotazovacího formuláře byla technická podpora integrována do stávající webové prezentace firmy.

Vzniklá internetová aplikace byla nainstalována na testovací server společnosti, kde proběhly základní testy funkčnosti.

Klíčová slova: technická podpora, sociální sítě, webová aplikace, desktopová aplikace, databáze, MVC

Abstract

This thesis deals with the design of the technical support system for the company KMB Systems Ltd. The main activity of the company is to develop products for an industrial automation and power engineering.

The technical support system was divided into two applications, web and desktop. Within the designing process modern methods of programming were used. The web application contains the user and administration interface and was designed by means of HTML, PHP and CSS. The desktop application contains only the user interface which has been programmed in C#. Since the applications were connected through a MySQL database, an access to the latest information was guaranteed. Using the login and query forms the technical support was integrated into the existing website of the company.

The developed web application has been installed on the test server of the company where the basic functionality tests were tested.

Keywords: technical support, social networks, web applications, desktop applications, database, MVC

Obsah

Seznam obrázků

Seznam použitých symbolů a zkratek

PHP	Hypertext Preprocessor
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
API	Application Programming Inteface
CMS	Content Management Systems
XML	Extensible Markup Language
DOM	Document Object Model
XPath	XML Path Language
URL	Uniform Resource Locator
PKCS	Public Key Cryptographic Standars
OOP	Object-oriented Programming
SQL	Structured Query Language
MVC	Model-view-controller
URI	Uniform Resource Identifier
LGPL	Library General Public License

Úvod

V dnešní době, kdy o úspěchu v udržení na předních příčkách trhu produktů rozhodují maličkosti, ocení zákazníci nabízené služby nejen před prodejem, ale i během užívání. Jednou z nadstandardních služeb může být i online technická podpora. Ta představuje souhrn online nástrojů, díky kterým může zákazník řešit specifické problémy v co nejkratším čase. Tyto služby zahrnují celý proces spojený s nákupem a užíváním produktu.

Mezi společnosti, které nezahrnují tento druh podpory prodeje, patřila i firma KMB systems s. r. o.. Díky komunikaci s Chief R&D manažerem mi byla nabídnuta možnost vývoje online podpory. Společnost do dnešní doby disponovala pouze základní formou podpory v podobě emailové či telefonické komunikace.

Cílem této diplomové práce je vytvoření systému technické podpory. Ten zahrnuje doplněk do softwaru ENVIS a webovou aplikaci, která je propojena se současnou internetovou prezentací společnosti.

Obě aplikace by měly splňovat základní požadavky, které se případně mohou rozšířit o další funkce, jako je přehled dostupných ovladačů a dokumentace či odeslání chybových logů. Webové rozhraní by mělo zahrnovat administraci pro nastavení systému a uživatelské prostředí, které usnadní zákazníkovi komunikaci. Tato aplikace bude realizována pomocí HTML, PHP, CSS a bude využívat společnou databázi MySQL s desktopovou aplikací. Desktopová aplikace bude navržena jako doplněk do současného softwaru ENVIS, který slouží pro vyhodnocování naměřených výsledků z nabízených produktů a je napsán v jazyce C#.

Nejprve bylo nezbytně nutné stanovit základní parametry, které by technická podpora měla splňovat. Hlavním požadavkem bylo zahrnutí licenční politiky, která má za úkol zjednodušit zákazníkovi způsob licencování produktů. Dalším neméně důležitým prvkem bylo vytvoření přehledné správy ohlášení chyb a to v podobně zákaznických dotazů. Do systému byly implementovány další funkce, které výrazně usnadňují komunikaci mezi zákazníkem a prodejcem.

Diplomovou práci lze rozdělit do tří částí. V první části jsou popsány možné varianty technické podpory podléhající open source licenci. Dále jsou také nastíněny způsoby zabezpečení předávaných dat a souborů a metody registrace a přihlášení uživatelů do systému.

Druhá část se dělí na analýzu prostředků komunikace se zákazníky a na samotný návrh webové a desktopové aplikace. Dosavadní online styk se zákazníky byl prováděn prostřednictvím desktopové aplikací ENVIS a webových stránek, které jsou navrženy v redakčním systému Joomla. V této části jsou dále popsány základní funkce celého systému a prostředky, které jsou využity při návrhu. Jedná se především funkce pro registraci a přihlášení uživatelů využívající účtu sociální sítě. Dále pak o nástroje pro registraci produktů s následným procesem licencováním, který zahrnuje jak webovou, tak desktopovou aplikaci.

Závěrečná část je věnována možnostem rozšíření aplikace a samotnému testování, které zahrnuje funkčnost nejen během vývoje, ale také na budoucím hostingu, kam by aplikace měla být instalována.

Jelikož jsou v diplomové práci použity moderní metody a to převážně v oblasti registrace uživatelů, byly pro naprogramování využívány API jednotlivých poskytovatelů.

1 Technická podpora

1.1 Open source systémy zákaznické podpory

V současné době nabízejí moderní technologie velkou škálu prostředků pod licencí Open source, které řeší problematiku zákaznické podpory. Hlavní výhodou je snadná instalace a následná správa. Avšak ne každý systém vlastní atributy, které jsou požadované budoucím administrátorem. Poměrná část z nabízených se odlišuje svým zaměřením (pro správu jízdenek či hlášení chyb), ale všechny spojuje funkce pro komunikaci se zákazníkem prostřednictvím webových formulářů. Jelikož současné internetové stránky firmy KMB systems s.r.o. jsou založeny na redakčním systému Joomla 2.5, byla jako první uvažována možnost instalace doplňku splňující požadované funkce.

Huru Helpdesk

Huru helpdesk představuje doplněk v podobě komponenty, která byla inspirována open source kódem *ASP-based helpdesk Liberium*. Disponuje řadou funkcí, mezi které patří také diskuse o daném problému mezi uživatelem a pověřenou osobou na straně technické podpory. K textu lze připojit soubor, který blíže specifikuje daný problém, případně lze zobrazit dobu strávenou nad daným problémem. Výhodou této komponenty je, že řešení problémů se provádí ve front-endu. Pověřená osoba tak nemusí mít přístup do administrace systému.

Zmíněná komponenta však nesplňuje již základní požadavek, a to rozdělení uživatelů CMS Joomla od uživatelů technické podpory. Z důvodu časté aktualizace CMS Joomla a následné nutné aktualizaci jádra komponenty byly uvažovány systémy, které by pouze odkazovaly na firemní stránky. V úvahu připadalo několik variant. Od aplikací pro malé firmy až po rozsáhlé, které řeší i celou vnitropodnikovou komunikaci. Jedním z uvažovaných je systém Bugzilla.

Bugzilla

Představuje webovou aplikaci pro sledování závad „Defect Tracking System“ nebo chyb „Bug tracking System“. Bugzilla byla původně vyvinutá a používaná organizací Mozzila s licencí Mozzilla Public License, která se využívá v proprietárních i

open source projektech. Systém pro sledování chyb umožňuje jednotlivcům nebo skupinám vývojářů efektivně seskupovat nevyřešené chyby v jejich produktech. Hlášení chyb je prováděno prostřednictvím formulářů, kde je evidováno mnoho parametrů (popis, stav, závažnost) a lze k nim připojit komentář, video nebo screenshot konkrétního problému. Tento systém se nespecializuje pouze na evidenci chybových hlášení, ale i na návrhy nových funkcí.

Nabízená řešení technické podpory by byla vhodná v případě realizace základního systému, který by spočíval v pouhé komunikaci mezi zákazníkem a prodejcem. V případě podpory pro firmu KMB systems s.r.o. je však nutné zahrnout specifické požadavky v podobě licenční politiky a komunikaci s desktop aplikací. Proto je třeba navrhnout vlastní aplikaci, která bude vytvořena jako samostatný celek. Aplikace bude zahrnovat základní funkce ze všech uvažovaných systémů a navíc specifické funkce pro správu licencí a produktů. K tomuto systému bude nutné vytvořit databázi, která bude využívána k evidenci uživatelů a uchování záznamů o produktech či dotazech. Veškeré zpracování dat a jejich manipulace bude podléhat zabezpečení.

1.2 Možnosti pro registraci a přihlašování uživatelů

1.2.1 Přímá metoda registrace a přihlašování

Přímá metoda registrace a přihlašování ve většině případů využívá vlastní databázi, do které se uživatelé nejprve musí zaregistrovat. Hlavními údaji při registraci je uživatelské jméno (jednoznačný identifikátor), bezpečné heslo a emailová adresa. Přihlášení pak probíhá pomocí uživatelského jména popřípadě emailu a hesla. Na základě těchto údajů webová aplikace ověří jejich platnost a porovná je s databází uživatelů. Jestliže se vše shoduje, zákazník je přihlášen a může využívat rozšířené funkce aplikace.

Hlavní výhodou tohoto řešení je plné přizpůsobení potřebám konkrétní aplikace a tím i získání kontroly nad všemi uživatelskými účty. Tato varianta je z hlediska provozovatele téměř ideální.

Z pohledu uživatele to ovšem znamená, že pro přihlašování na odlišnou službu musí opět volit jednoznačné uživatelské jméno a pamatovat si další heslo. Pro uložení hesel se využívají různé pomůcky, které jsou implementovány v jednotlivých internetových prohlížečích. Avšak při přihlašování na jiném počítači musíme tyto

pomůcky synchronizovat, což vede ke komplikacím. Další nevýhoda je nepřenositelnost identity a její nejednoznačnost, která zabraňuje využít stejné přihlašovací údaje u jiné služby, protože na jiném serveru již může pod stejnou identitou vystupovat někdo jiný.

1.2.2 Registrace pomocí autentizačních metod

Nezávislé autentizační metody by měly splňovat základní myšlenku univerzálnosti, jednotnosti, unikátnosti a nezávislosti.

- univerzálnost – použitelnost pro jakoukoli službu
- jednotnost – pro všechny služby stejná
- unikátnost – pro prokazování identity oprávněného uživatele
- nezávislost – nepropojenost s poskytovatelem služby

Při přihlašování pomocí správy identit (providerem) nepotřebuje uživatel heslo ke každé službě, využívá pouze heslo k účtu u správce. Pod touto identitou pak může využít více služeb, které tuto metodu podporují. Provozovatel tak ztrácí část kontroly nad uživateli a nemusí se starat o problémy týkající se správou účtů (zabezpečený přenos údajů, ztráta hesla, bezpečný přenos při přihlašování).

User-centric (ověřují uživatele)

Při užívání těchto metod autentizace si uživatel založí u důvěryhodného poskytovatele účet, pomocí kterého se může přihlašovat k jiným službám. U OpenID si může uživatel vybrat z různých poskytovatelů, kterým bude důvěřovat a svěří jim své údaje. Jestliže si nevybere, je tu možnost, že jako důvěryhodná autorita bude jeho server. Oproti tomu u LiveID, OpenAuth, Facebook Connect, Google Federated Login a dalších je důvěryhodným subjektem firma, která danou metodu poskytuje.

OpenID

OpenID je otevřená a decentralizovaná metoda pro ověřování uživatelů. Její identity jsou spravovány jedním centrálním správcem, což umožňuje rozhodnutí, kterému poskytovateli uživatel svěří své údaje. OpenID identifikátor má tvar běžné URL, pomocí které jsou předávány klientovi dodatečné informace o uživateli (username, email, …). Jestliže uživatel souhlasí, jsou providerem poskytnuty.

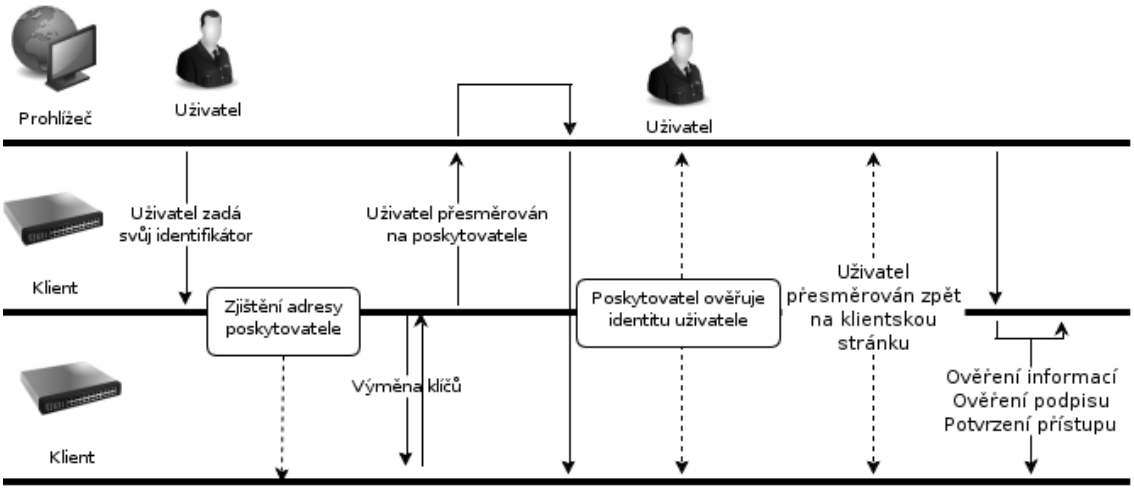
Nejnovější specifikace OpenID protokolu je verze 2.0, která nahradila značně neuspokojivou verzi 1.0 a pozdější 1.1. Specifikace OpenID 1.1 byla první verzí OpenID protokolu, která se dala běžně použít. Nejnovější specifikace nabízí několik vylepšení:

- jednotnou podporu pro rozšíření OpenID protokolu
- podporu větších požadavků a odpovědí, které by se nevešly do URL (předávání dat pomocí HTTP POST)
- možnost zadat identifikátor OpenID providera namísto identifikátoru uživatele
- podporu obecných rozšiřovacích atributů

OpenID 2.0 obsahuje identifikátory poskytovatelů. Jedná se o zástupný identifikátor, který dále odkazuje na poskytovatele. Ověřování probíhá v několika krocích a to mezi prohlížečem, poskytovatelem a klientem.

Uživatel nejprve sdělí svůj identifikátor (Identifier), který může být v podobě URI s protokolem http, https (URL) nebo XRI. Poté klient normalizuje přijatý identifikátor a zjišťovacím procesem (Discovery) se snaží získat adresu poskytovatele (OpenID provider). Provider představuje server, který umožňuje klientovi získat informace, zda daný uživatel vlastní konkrétní identifikátor. Klient a poskytovatel si mezi sebou vytvoří přiřazení a pomocí Diffie-Hellman algoritmu si vymění klíč, který poskytovatel bude používat při podpisu odpovědi a klient pro ověření pravosti. Tímto přiřazením odpadá nutnost dalších dotazů na ověřování podpisu při každém autentizačním požadavku nebo odpovědi. Po výměně klient přesměruje uživatelův prohlížeč a v URL předá požadavek pomocí autentizace.

Poskytovatel ověří, jestli uživatel může používat daný identifikátor a jestli si to opravdu přeje. Pro ověřování je možné využít jméno a heslo, SMS kód nebo elektronický klíč. Dále je uživatelův prohlížeč přesměrován na klientovy stránky a zároveň v URL předá informace o stavu autentizace, jestli je potvrzena nebo zamítnuta. Na závěr klient ověří informace předané poskytovatelem. Tento způsob autentizace je v současné době poskytován portály: AOL, BBC, IBM, MySpace, PayPal, Yahoo! a Google.



Obr. 1.1 - OpenID 2.0 protokol

Výhody standardu OpenID:

- odpadá vyplňování registračních formulářů (pomocí jednoho uživatelského jména a hesla se přihlásíme na více internetových stránkách)
- změna osobních údajů se provádí na jednom místě
- vyšší bezpečnost citlivých dat u poskytovatelů
- je otevřené a decentralizované – veřejně přístupná specifikace protokolu
- využití pokročilých autorizačních metod – k přihlašování nemusí být nutné pouze heslo, lze využít i autorizační klientský certifikát, SMS zprávu

LiveID

LiveID pochází od firmy Microsoft a nabízí přihlášení ke službám Live. Identifikátor má tvar emailové adresy Windows Live nebo Hotmail a na rozdíl od OpenID klient nedostává od poskytovatele osobní údaje, ale pouze alfanumerický hash, který je unikátní pro daného uživatele a službu. Jestliže by chtěl klient využívat další údaje, musí si o ně zažádat a poté spárovat s daným hashem LiveID.

OpenAuth

OpenAuth je jako OpenID nebo LiveID autentizační metoda, která uživatele přesměrovává na stránky providera (v tomto případě AOL), kde se přihlásí pod svým jménem a heslem. Při přihlašování může využít OpenID identifikátor nebo svou službu u AOL (ICQ). Pomocí OpenAuth se lze přihlásit k webovým i desktopovým aplikacím včetně Flex/AIR aplikací.

Facebook Connect

Facebook Connect je aplikační programové rozhraní sociální sítě Facebook, pomocí kterého se propojují profily uživatelů s požadovanou webovou stránkou, která tak získá informace o daném uživateli. Pro autentizaci a autorizaci využívá ověřovací protokol OAuth 2.0.

Přihlášení probíhá prostřednictvím Login Dialogu a má několik bezpečnostních funkcí, které chrání soukromí.

- **Základní informace** - Při přihlášení do aplikace jsou poskytnuty pouze základní údaje o uživateli a to: id, jméno, příjmení, uživatelské jméno, pohlaví, lokace, odkaz (url adresa kde se nachází profil).
- **Rozšířené informace** - Poskytuje oprávnění k citlivějším informacím a povolením, které jsou schopné publikovat a mazat záznamy.
- **Open Graph oprávnění –** Možnost publikování akcí na Open Graph a získání akce publikované jinými aplikacemi.
- **Stránka oprávnění** – Umožňuje aplikaci načíst access_tokens pro stránky a aplikace, které uživatel spravuje.
- **User and Friend oprávění** – Tyto oprávnění poskytují informace, které uživatel zveřejňuje svým přátelům (např. přístupy k seznamu činností, k datům narození, k událostem, ke stránkám, co se uživateli líbí).

Při prvním přihlášení přes Facebook Connect jsou vypsány údaje a konkrétní oprávnění, které bude daná aplikace využívat.

Google Federated Login

Mezi nejvyužívanější poskytovatele OpenID patří Google, který mimo protokolu OpenID 2.0 podporuje následující rozšíření:

- OpenID Atribute Exchange 1.0 – možnost přístupu vývojářů k uživatelským informacím získaných registrací u Google se souhlasem uživatele
- OpenID User Interface 1.0 – podpora alternativních prostředí pro ověření autorizace

- OpenID + OAuth Hybrid protokol – umožňuje kombinovat žádosti OpenID s *OAuth authentication* (dokument popisující proces autentizace) a zjednodušuje proces při souhlasu s ověřováním
- PAPE (Provider Authetication Policy Extension) – poskytuje mechanismy při ověřování uživatele, informuje poskytovatele, jaké autentizační prostředky byly použity, klient může požadovat, aby uživatelé byli ověřováni pomocí metod *phishing-resistant* nebo *multi-factor*

Proces ověřování OpenID

OpenID ověření při přihlášení pro webové aplikace zahrnuje několik kroků mezi webovou aplikací a Google uživatelem.

Obr. 1.2 – Ověření přístupu

1. Webová stránka požádá, aby se uživatel přihlásil prostřednictvím login formuláře přes svůj Google účet.
2. Uživatel se přihlásí pomocí standardního formuláře.
3. Webová aplikace zjišťovacím procesem (Discovery) odešle požadavek společnosti Google za účelem získání informací k ověření přihlášení a koncového bodu (endpoint).

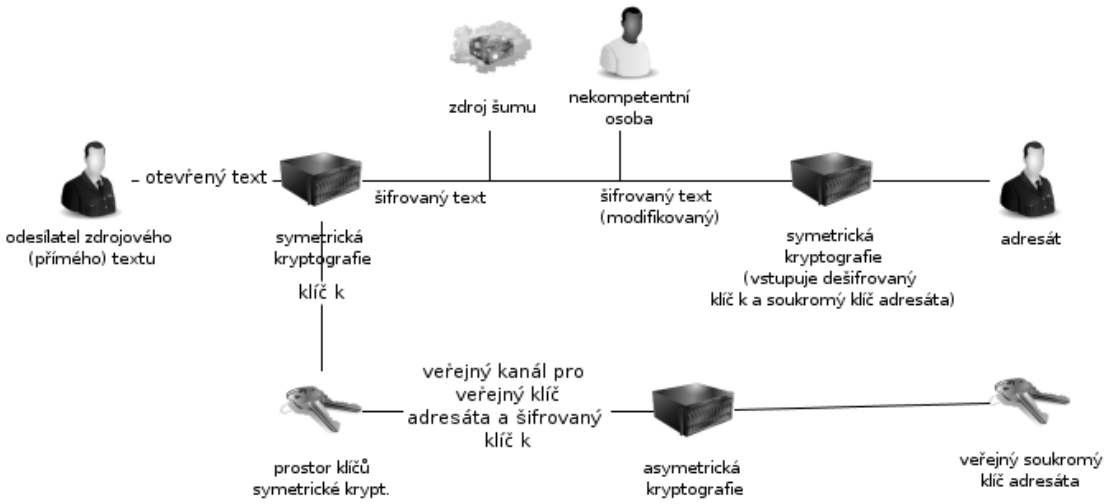
4. Google vrací XRDS dokument, který obsahuje adresu koncových bodů.
5. Webová aplikace odešle přihlašovací požadavek na ověření adresy koncového bodu Google.
6. Uživatel je přesměrován na přihlašovací stránku Google, aby se přihlásil.
7. Jakmile se přihlásí, Google zobrazí stránku a upozornění, že aplikace jiného výrobce žádá o ověření. Uživatel musí souhlasit/zamítnout propojení svých uživatelských údajů s Google účtem. Pokud je již uživatel přihlášen ke svému účtu, nebo již v minulosti schválil přihlašovací údaje na této stránce, je tento krok přeskočen.
8. V případě že uživatel souhlasí s ověřením, Google ho vrátí zpět pomocí URL uvedené v *oppid.return_to* na stránku se žádostí.
9. Webová stránka aplikace používá získaný Google identifikátor, pomocí kterého rozpozná uživatele a umožní mu přístup k datům a aplikacím.

1.3 Možnosti šifrování dat

V současné době je velice důležité použití kryptografických postupů, např. k zašifrování nebo k ochraně citlivých dat při přenosu přes nebezpečné prostředí. Takovým prostředím může být např. Internet či jiná veřejná datová síť. Existují dva druhy šifrování, a to šifrování s tajným klíčem (symetrické) a šifrování s veřejným klíčem (asymetrické). Jejich rozlišení souvisí s počtem a způsobem použití pro daný algoritmus. Symetrické šifrování využívá stejného klíče pro šifrování i dešifrování, oproti tomu asymetrické šifrování využívá minimálně dva různé klíče.

1.3.1 Symetrické šifrování

Symetrické šifrování je založeno na jediném utajovaném šifrovacím klíči, který musí znát obě strany. Je nutné, aby se příjemce i odesílatel zprávy předem dohodli na jednom klíči, který si musí bezpečným způsobem předat po důvěryhodném kanále i s dalšími údaji (např. typ algoritmu) a jenž budou znát jen oni. Pro výměnu klíče přes veřejný kanál je použito asymetrické šifrování (Obr. 1.3).



Obr. 1.3 – Hybridní kryptosystém

Jedná se především o blokové šifry, kde se pracuje s datovými bloky, které mají pevně stanovenou délku. Jestliže jsou data větší než stanovená délka, rozdělí se na více bloků. Při šifrování se každý blok zašifruje algoritmem, který je řízen utajovaným klíčem.

Hlavní výhody a nevýhody symetrického šifrování:

+ rychlost algoritmu
+ vhodné pro méně výkonné počítače
- předání tajného klíče
- pro každou komunikující stranu jiný klíč

Šifra AES (Rijndael)

Šifra Rijndael je iterační symetrická bloková šifra, která může mít proměnou délku bloku a klíče. Velikost bloku a klíče může být 128, 192 nebo 256 bitů. Jelikož kombinuje požadavky na bezpečnost, výkonnost, jednoduchost a flexibilitu, implementace byla vybrána organizací NIST za nový šifrovací standard.

Přestože šifra podporuje větší bloky, je délka vstupního a výstupního bloku definována jako 128 bitů. Základem je datový typ *word,* který disponuje velikostí 4 bajty (32 bitů). Pro její návrh jsou použity algebraické operace různých struktur.
Jestliže jsou šifrovaná data delší, jsou zpracovávány po jednotlivých blocích. V případě kratšího datového řetězce (poslední blok se zbytkem dat) je potřeba doplnit data na

odpovídající délku. Doplnění bloku (tzv. padding) lze provést několika algoritmy od primitivního, který blok doplní nulami, až po složitější schémata. Nejvyužívanějším mechanismem dle skupiny standardů pro kryptografii s veřejným klíčem navrženým a publikovaným společností RSA Security, je PKCS#7, který je podporován v .NET Frameworku i v PHP. Při implementaci je důležité rozhodnout, co se bude šifrovat a zvolit správný režim.

- ECB (Electronic Codebook) – vhodný pro šifrování náhodných dat
- CBC (Cipher Block Chaining) – vhodný pro šifrování souborů
- CFB (Cipher Feedback) – vhodný pro šifrování datového proudu
- OFB (Output Feedback, 8 bitů) – srovnatelná s CFB, použití v aplikacích, kde není možné tolerovat chyby, není bezpečný, protože pracuje pouze v 8 bitovém režimu
- NOFB (Output Feedback, n bitů) – srovnatelný s OFB, ale je bezpečnější, protože pracuje s celým blokem
- RAND (System Random Number Generator)

Jelikož tato šifra slouží ve vyvíjející aplikaci k šifrování souborů přenášejících se na server, je použit režim CBC. Tento režim funguje tak, že před zašifrováním se odpovídající blok otevřeného textu XORuje předcházejícím blokem zašifrovaného textu. Z toho vyplývá, že jednotlivé bloky na sobě závisejí a při dešifrování jednoho konkrétního se musí nejprve dešifrovat všechny předcházející bloky. V případě dešifrování první datové části se náhodně vygeneruje předchozí blok, který se přidá k zašifrovaným datům jako „nultý blok“, jenž následně slouží k dešifrování prvního bloku.

Implementace v jazyce C# se aplikuje na pole bajtů (pro vstup i výstup) a provádí se za pomoci třídy *System.Web.Security.Cryptography.RijndaelManaged*. Jestliže je potřeba převést textový řetězec, je využívána třída *Encoding.UTF8.GetBytes*, případně *Encoding.ASCII.GetBytes.*

V případě programovacího jazyka PHP jsou využívány funkce *stext()* a *ustext()*, které zadaným klíčem převedou požadovaný řetězec dle algoritmu CBC.

```
function stext($key,$text){
$iv_size = mcrypt_get_iv_size(MCRYPT_RIJNDAEL_256, MCRYPT_MODE_CBC);
$iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);
$crypttext = mcrypt_encrypt(MCRYPT_RIJNDAEL_256, $key, $text, MCRYPT_MODE_CBC, $iv);
return $sifrovane;}
```

```
function ustext($key,$crypttext){
$iv_size = mcrypt_get_iv_size(MCRYPT_RIJNDAEL_256, MCRYPT_MODE_CBC);
$iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);
$desifrovane=mcrypt_decrypt(MCRYPT_RIJNDAEL_256,$key,$crypttext, MCRYPT_MODE_CBC, $iv);
return $desifrovane;
}
```

Šifra AES zakládá část své bezpečnosti na dosud méně známých aritmetických operacích, které by mohly v budoucnu přinést ohrožení v podobě jejich prolomení. Disponuje vysokou rychlostí a flexibilitou, která přináší možnost implementace na Smart Card a použitelnost při konstrukci hashovacích funkcí.

1.3.2 Asymetrické šifrování

Naopak **asymetrické šifrování** používá dva klíče, veřejný a privátní, které si uživatel vygeneruje pomocí některého z dostupných programů a stává se tak jejich jediným majitelem. Jestliže nelze prakticky odvodit jeden klíč z druhého, je šifrování označováno jako šifrování s veřejným klíčem. Pro tento způsob platí pravidlo, že pokud je zpráva zašifrována jedním typem klíče, lze ji dešifrovat pouze druhým typem. Velkou výhodou je skutečnost, že jeden z klíčů může být poskytnut komukoli. Odesílatel tak může zašifrovat zprávu veřejným klíčem, kterou pak může přečíst pouze vlastník privátního klíče. Jestliže adresát chce na danou zprávu odpovědět, musí použít veřejný klíč odesílatele, aby zprávu nemohl dešifrovat někdo jiný.

Hlavní výhody a nevýhody asymetrického šifrování:

+ privátní klíče jsou pouze u majitelů, dostupné jsou pouze veřejné klíče
- velmi náročné na matematické operace a na výkon počítače

Algoritmus RSA

RSA je založena na asymetrickém šifrování, které umožňuje generovat dvojici klíčů. Každý zúčastněný subjekt má vlastní soukromý klíč a jemu odpovídající veřejný klíč. Jeden je použit pro šifrování a jeden pro dešifrování. Pro ochranu komunikace mezi více subjekty je potřeba 2*n klíčů. Algoritmus využívá klíče o velikosti 517-4096 bitů. Pro větší bezpečnost zahrnuje matematické operace v podobě rozkladu velkých čísel na prvočísla, což představuje vysokou výpočetní náročnost.

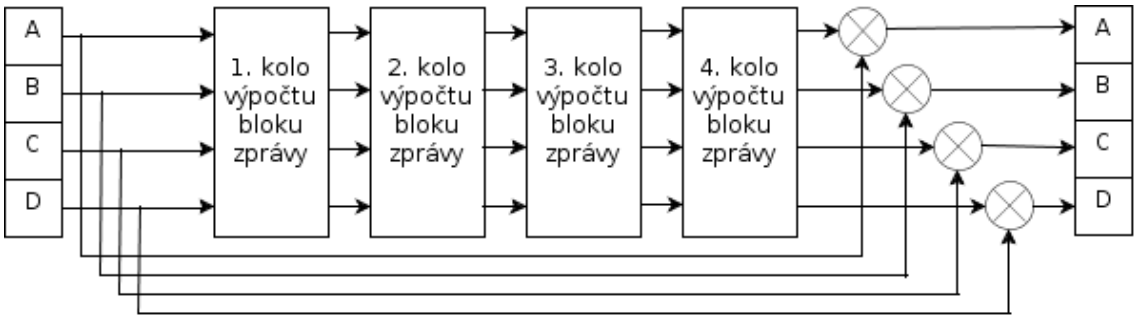
1.3.3 Hashovací funkce

Hashovací funkce je jednosměrná matematická funkce, do které vstupuje blok proměnné délky a vystupuje blok pevné délky (obvykle 128 nebo 160 bitů), který se nazývá hash. Výsledek je totožný s originálem, ale při výpočtu dochází k redukci množství původní informace, proto jej nelze rekonstruovat. Využívá se jako součást protokolů pro digitální podpisy, kryptografických protokolů, certifikátů, ale i při bezpečném ukládání a kontrole hesel v operačních systémech.

Aby se zvýšila bezpečnost při ukládání hashe, využívá se takzvané solení, které spočívá v přidání dalšího řetězce na vstup – sůl. Sůl může představovat cokoli a nemusí být tajná, ale požadavkem je, aby se lišila mezi jednotlivými uživateli. Sůl se pro porovnání musí posílat společně s hashovaným řetězcem. Proto je vhodné při hashování využít uživatelské jméno, pro které není nutné vytvářet speciální sloupec v databázi.

MD5

Funkce MD5 zpracovává soubory s libovolnou délkou a jejím výstupem je hashovací funkce s délkou 128 bitů. Při hashovaní je soubor rozdělen do 512 bitových bloků. Jestliže soubor není možné rozdělit do uceleného počtu 512 bitových bloků, poslední blok je náhodně doplněn na konečnou délku 512 bitů. Dále je každý 512 bitový blok rozdělen do 16 proměnných o 32 bitech. Tyto proměnné jsou rozděleny do čtveřic A, B, C, D(Obr. 1.4), jejichž spojení vytvoří jednu 128 bitovou hašovací hodnotu. V první fází výpočtu se do proměnných A, B, C, D vloží inicializační hodnoty. Výpočet funkce probíhá, dokud nejsou zpracovány všechny bloky.



Obr. 1.4 – Hashovocí funkce MD5

Hashovací funkce MD5 se kvůli zprávám o jejím prolomení přestává využívat jako samostatný nástroj pro šifrování hesel v databázi. Její uplatnění lze však nalézt při

šifrování dat, které nepodléhají velkému utajení. V aplikaci technické podpory se pomocí této funkce generuje řetězec, který identifikuje licencované soubory.

SHA

SHA (Secure Hash Algorithm) je hashovací funkce vytvářející ze vstupních dat výstup (otisk), který má pevně danou délku. Tento otisk (kontrolní součet) disponuje vlastností, kdy při nepatrné změně na vstupu dojde k velké změně na výstupu. SHA obsahuje pět algoritmů SHA-1, SHA-224, SHA-256, SHA-384 a SHA-512 (předchozí čtyři se souhrnně nazývají SHA-2).

SHA-1 vytváří otisk o velikosti 160 bitů. U algoritmů spadajících pod SHA-2 je výstupní hodnota rovna jejich označení v bitech.

Hashovací algoritmy SHA jsou relativně bezpečné, protože útočník, který by chtěl algoritmus prolomit, by musel nejprve najít zprávu, která odpovídá danému otisku nebo najít dvě rozdílné zprávy, které mají stejný otisk.

Dešifrování algoritmu lze provádět dvěma již zmiňovanými způsoby. V prvním případě může být použita hrubá síla neboli vzorový útok (preimage attack), tedy při hledání 2^L výpočtů, kde L je velikost bitů v otisku zprávy. Druhý případ, kdy se hledají dvě rozdílné zprávy se stejným otiskem, se nazývá kolize. Ta vyžaduje pouze $2^{L/2}$ výpočtů k provedení tzv. narozeninového útoku (birthday attack).

Hashe jsou nejvíce využívány při šifrování hesel v databázi, proto aby nebylo možné při vstupu do databáze zjistit, jaké heslo uživatel použil.

Z důvodu bezpečnosti byl pro šifrování hesel ve vyvíjené aplikaci využit hash SHA-2, který je implementován jak v desktopové aplikaci pomocí jazyka C#, tak ve webové aplikaci ve scriptovacím jazyce PHP.

Implementaci v C# obstarává střída *System.Security.Cryptography.SHA512*, pomocí které je vytvořena a následně volána metoda pro ověřování hesla.

```
public static string CreatePasswordHash(string _password, string _salt)
{
string saltAndPwd = String.Concat(_password, _salt);
SHA512 sha512 = new System.Security.Cryptography.SHA512Managed();
byte[] sha512Bytes = System.Text.Encoding.Default.GetBytes(saltAndPwd);
byte[] cryString = sha512.ComputeHash(sha512Bytes);
string hashedPwd = string.Empty;
        for (int i = 0; i < cryString.Length; i++)
        {
           hashedPwd += cryString[i].ToString("x2");
        }
return hashedPwd;
}
string password = CreatePasswordHash(pass, email);
```

Využití SHA-2 ve webové aplikaci zajišťuje funkce *hash()*, jejíž parametry určují způsob šifrování daného řetězce (hesla).

```
$password = hash("sha512", $password . $email);
```

Návrh technické podpory pro KMB systems s.r.o.

„*Společnost KMB systems se od roku 1991 zabývá vývojem elektroniky zejména pro průmyslovou automatizaci a energetiku. V současnosti nabízí zejména produkty pro kompenzaci jalového výkonu NOVAR, široké spektrum vestavěných měřících přístrojů řady SM a přenosných analyzátorů kvality napětí SIMON. Na základě požadavků zákazníků vyvíjí a dodává i jiná speciální elektronická zařízení pro energetiku, průmysl i jiná odvětví.*“[1]

2 Analýza současné zákaznické (technické) podpory

V současné době je zákaznická podpora společnosti KMB systems s.r.o. implementována do internetových stránek a desktopové aplikace ENVIS. V obou případech se jedná o minimální podporu, která je řešena prostřednictvím telefonátů, emailů, Skype komunikací a vyplněním kontaktního formuláře, který je umístěn na webové prezentaci. Uživatel ani centrum zákaznické podpory tak nemají rychlý přehled o konkrétních problémech a jejich řešení. Jsou odkázáni na vyhledávání v historii odeslaných či doručených emailů.

Internetové stránky jsou vytvořeny v redakčním systému Joomla, který disponuje velkou škálou rozšíření. Struktura stránek je koncipována do formy katalogu. Zákazník má tak přehled o všech nabízených produktech a jejich vlastnostech. Na stránkách jsou také k dispozici dokumenty, které obsahují manuály k jednotlivým produktům.

V případě užívání aplikace ENVIS není poskytnut zákazníkovi přímý kontakt se zákaznickou podporou. Podporu v této aplikaci představuje položka „Nápověda“, kde uživatel získá základní informace o aplikaci a její aktualizaci, o dokumentaci aplikace a možnostech zobrazení chybových logů.

[1] KMB systems s. r. o. [online] [cit.2013-03-06]. Dostupný z WWW:< http://www.kmb.cz>

3 Popis navrhovaného systému

Aktuální webová prezentace firmy je založena na redakčním systému (CMS) Joomla 2.5.9, aplikace technické podpory bude navržena jako samostatný systém, který bude následně propojen pomocí doplňků.

Rozhodnutí o návrhu samostatného systému vyplynulo ze specifických požadavků firmy. Cílem bylo oddělení uživatelů redakčního systému od uživatelů technické podpory a bezproblémový chod aplikace po aktualizaci redakčního systému.

Aplikace technické podpory bude rozdělena na dvě části. První část je tvořena webovou aplikací, kde jsou mimo jiné implementovány administrátorské funkce. Druhou část tvoří doplněk pro desktopovou aplikaci ENVIS, který rozšíří současnou nevyhovující podporu pro zákazníky.

Funkční požadavky:

- Systém bude umožňovat registraci uživatelů přes sociální sítě
- Systém bude mít tři základní role uživatelů (administrator, responder, user).
- Uživatelé si budou moci zaregistrovat zakoupené přístroje a po té je licencovat.
- Dotazy týkající se technické podpory mohou zasílat jak registrovaní, tak neregistrovaní uživatelé (možnost připojit soubor).
- Přihlášení uživatelé budou mít přehled o aktuálních ovladačích pro svůj registrovaný produkt.
- Přihlášení uživatelé budou mít možnost zasílat logy z aplikace ENVIS pro řešení problémů,

Nefunkční požadavky:

- Hlavní část systému bude realizována pomocí PHP, HTML a CSS.
- Doplněk pro desktopovou aplikaci bude navržen ve Visual Studiu 2012 na platformě C# .NET.
- Všechny záznamy budou ukládány do databáze MySQL.

3.1 Použité technologie

3.1.1 Návrhový vzor MVC (Model View Controller)

Návrhový vzor rozděluje aplikaci na tři základní části: *mode, view* a *controller*. Do datové vrstvy **(model)** je možné zahrnout logiku aplikace, strukturu dat a potřebné přístupy k nim. Uživatelské rozhraní **(view)** obsahuje výstupy v uživatelsky přívětivé podobě a aplikační logika **(controller**) sloužící k řízení chodu celé aplikace. Všechny části mají svoji vlastní úlohu a i přesto, že jsou mezi sebou propojeny, lze je modifikovat samostatně.

Model

Model představuje množinu funkcí a tříd, které prioritně vykonávají operace s daty (načítání, zpracování, ukládání). Kromě manipulace s daty a zajištění jejich přístupů (např. komunikace s databází) je zde veškerá business logika, která zodpovídá za chod celé aplikace.

Model jako celek neví nic o *view* a *controlleru*. Jestliže je správně navržen, měl by být využitelný v jakékoliv aplikaci a to beze změny.

View

View vyjadřuje data, která zastupují model. Jeho úkolem jsou výstupy aplikace např. v grafické či textové podobě. Přístup k datům zajišťuje *controller* nebo jiné rozhraní, které model nabízí. Data jsou zobrazována v různých podobách, jak ve značkovacích jazycích HTML, XHTML či XML, tak i v podobě obyčejného textu.

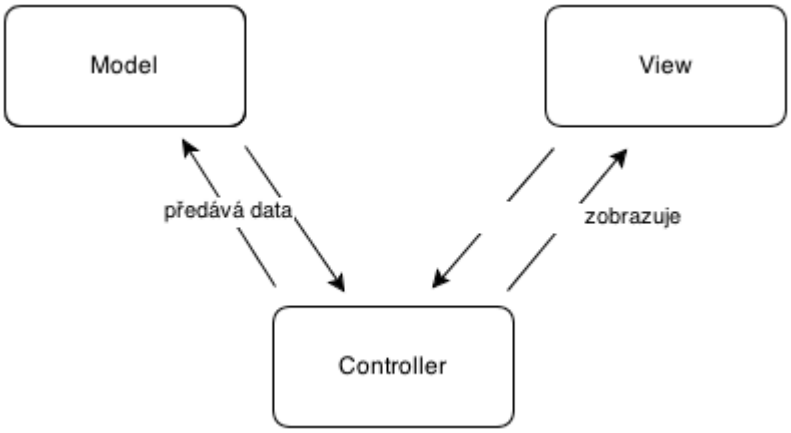
Controller

V tomto návrhovém vzoru plní *conroller* funkci řídící jednotky, která reaguje na uživatelské požadavky a zabezpečuje změny ve *view* nebo modelu. Na základě žádostí rozhoduje, jaké funkce se v modelu vykonají a jak se zobrazí.

U webové aplikace *controller* představuje skripty, ke kterým aplikace přistupuje pomocí metod GET a POST.

Obecný postup MVC

Architektura MVC obsahuje vlastnosti jednotlivých komponent (model, view, controller) a definuje vztahy mezi nimi. Realizace MVC je závislá na způsobu komunikace mezi jednotlivými komponentami, avšak obecně platí:



Obr. 3.1 – Architektura MVC

1. Uživatel provede nějakou akci v uživatelském rozhraní (např. stisknutí tlačítka a odeslání formuláře).
2. *Controller* obdrží oznámení o této akci z objektu uživatelského rozhraní.
3. *Controller* přistoupí k modelu a v případě potřeby ho zaktualizuje na základě provedené uživatelské akce (např. uloží dotaz).
4. Model zpracuje změněná data (např.: přepočítá aktuální počet dotazů).
5. *View* použije zaktualizovaný model pro zobrazení nových dat (např. zobrazí počet dotazů) a *view* získává data přímo z modelu nebo je předá *controlleru.*
6. Uživatelské rozhraní čeká na další akci uživatele, která celý cyklus zahájí znovu.

Důležitou úlohu zde hrají použitá rozhraní. Čím obecnější a kvalitnější jsou, tím mají větší flexibilitu (jsou však pomalejší).

Výhody a nevýhody MVC

Výhody MVC oproti běžnému návrhu webových aplikací [7][12]:

- **Opětovné použití kódu, který je součástí modelu**
  Oddělené komponenty *model* a *view* umožňují využít logiku modelu v jakémkoliv zobrazení. Protože *model* není závislý na *controlleru* ani *view*, jednotlivé části modelu se snáze modifikují, testují a udržují.
- **Snadná rozšiřitelnost**
  Neomezené přidávání *controllerů* a *view* bez ovlivnění funkčnosti aplikace. Pokud je zachováno společné rozhraní, je možné využívat funkcionalitu komponent.
- **Přehlednost návrhu**
  Přehlednost kódu zajišťuje lepší orientaci a snižuje tak možnost duplicity.
- **Snadnější podpora nových typů klientů**
  Při rozvoji aplikace o nové uživatele, stačí vytvořit nové *view*, případně *controller*. Ty však budou přistupovat k datům pomocí stávajícího modelu.
- **Nezávislost při vývoji komponent**
  Jednotlivé prvky architektury mohou být navrhovány nezávisle.

Nevýhody

Architektura MVC se v nárůstu rozsáhlých aplikací snaží eliminovat neuspořádanou stavbu a zpřehlednit jejich strukturu, a to rozdělením aplikace na části s podobným zaměřením. Z tohoto rozdělení plynou následující nevýhody:

- Nárůst zdrojového kódu.
- Jestliže platforma nenabízí žádné vestavěné řešení, musí se navrhnout vlastní MVC.
- Znalost principů MVC pro správnou implementaci systému.

3.1.2 HTML a xHTML

HTML (HyperText Markup Language) je hlavním značkovacím jazykem, který vytváří obsah webových stránek a je aplikací jazyka SGML (Standar Generalized Markup Language). Značkovacím jazykem je označován jazyk pro vytváření dokumentů obsahující jak text, tak i instrukce pro zpracování. Zpracování HTML a xHTML se provádí v prostředí webového prohlížeče.

3.1.3 CSS

CSS (Cascading Style Sheet) neboli kaskádové styly jsou nadstavbou pro značkovací jazyky HTML, XHTML nebo XML a popisují způsob zobrazení stránek. K zobrazení webové stránky nejsou styly potřebné, ale díky nim lze vytvořit jedinečný vzhled stránek a aplikovat tak návrh grafika.

Hlavní výhody použití kaskádových stylů:

- možnost nastavení jednoho dokumentu pro více výstupních zařízení (monitor, tiskárna, mobilní telefon)
- oddělení stylů od obsahu zaručuje lepší přehlednost
- soubor css se načítá do paměti CACHE a tak má dokument menší velikost a výsledná stránka se načítá rychleji
- využití dědičnosti při úpravách jednotlivých elementů

3.1.4 PHP

PHP (Hypertext Preprocessor) je programovací jazyk primárně určený k programování dynamických webových stránek a zahrnuje se do struktury HTML či xHTML.

PHP scripty jsou prováděny na straně serveru, uživatel vidí až jejich výsledek. Jazyková syntaxe je odvozena od několika programovacích jazyků (Perl, C, Pascal a Java) a je nezávislá na platformě. PHP skripty fungují bez nutnosti větší modifikace na mnoha různých operačních systémech.

Nejrozšířenější verze PHP 5 nabízí využití ZEND Enginu II. generace, který mimo jiné obsahuje vylepšenou podporu pro objektově orientované programování (OOP). Dále vlastní doplňky pro explicitní konstruktory a destruktory, pro klonování objektů a abstraktní třídy, které zlepšující využití OOP v PHP. PHP 5 také nabízí dokonalejší podporu XML založenou na knihovně *libxml2* a zavedení *Simple XML,* který usnadňuje využití při čtení a manipulaci s XML.

Scriptovací jazyk PHP má svobodnou Open Source licenci a je podporován na většině hostingů. Je tedy primárně určen pro vývoj webových stránek

3.1.5 JavaScript a jQuery

JavaScript je interpretovaný programovací jazyk, který disponuje základními objektově orientovanými schopnostmi. Díky programovaným konstrukcím, příkazů if a smyčce while připomíná jádro jazyka programovací jazyky C, C++ a Javu. JavaScript však nevyužívá typové kontrolky, což znamená, že proměnné nemusí mít specifikovaný typ.

JavaScript umožňuje nadstavbu v podobě frameworku jQuery, pomocí něhož je usnadněná práce s elementy DOM a k výběru elementu lze použít CSS a XPath. jQuery je rychlá a přesná JavaScript knihovna, která umí jednoduše procházet HTML elementy a pracovat s událostmi či animacemi. Hlavními výhodami je malá velikost, podpora všech hlavních webových prohlížečů a podpora CSS pravidel.

3.1.6 MySQL

MySQL je relační databázový systém typu DBMS (Database Managment System), který vlastní společnost Oracle. Databáze je tvořena tabulkou, které se skládá ze sloupců a řádků. Sloupce označují datové typy jednotlivých záznamů obsažených v řádcích. Práce s databází se provádí prostřednictvím dotazů, které vychází z deklarativního programovacího jazyka SQL. Pro správu databáze přes webové rozhraní se využívá grafická nadstavba phpMyAdmin, která umožňuje snadnou práci s danou databází.

3.2 Návrh databáze

Veškerá zpracovaná data aplikací technické podpory jsou uchována v databázi, jejíž struktura je postavena na relačním databázovém systému MySQL. Tímto je zajištěna lepší manipulace s daty a souběžný přístup více uživatelů k datům ve stejnou dobu.

Databáze obsahuje 15 tabulek, z nichž většina podléhá 3. normální formě.

Popis databázových objektů

Tabulka users uchovává údaje o všech registrovaných uživatelích, kteří mohou plně využívat nástroje technické podpory. Atributy `email` a `password` slouží k přihlášení uživatele do systému.

Tabulka authentications uchovává záznamy o uživatelích, kteří se přihlašují prostřednictvím sociální sítě, jejíž jméno je uloženo v atributu `provider` a identifikováno `provider_uid`. Součástí tabulky je neméně důležitý atribut `profile_url`, který obsahuje přímou adresu profilu konkrétního uživatele.

Tabulka userType je výpisem uživatelských rolí. Obsahuje tři hodnoty (administrator, user, responder). Uživatelským rolím je věnována samostatná kapitola.

Tabulka product_reg je asociativní tabulka a řeší vztah M:N mezi tabulkami `users`, `produkt` a `licence`. Obsahuje informace o všech produktech, které si jednotliví uživatelé zaregistrovali. Hlavními atributy jsou zde `pocet_lic` (počet licencí daného produktu), `hash` (slouží pro kontrolu, při licencování přes internetový prohlížeč a zároveň je využitý jako část názvu dočasného licenčního souboru) a `newhash` (tvoří část názvu souboru při uložení konečného licenčního souboru).

Tabulka licence je výpisem všech stavů, které mohou nastat při vytváření licenčního souboru.

- register – produkt je zaregistrován uživatelem
- net_waiting – je odeslána licence prostřednictvím internetového prohlížeče a čeká na konečné zpracování v desktopové aplikaci
- wait – licence je zpracována v desktopové aplikaci a je postoupena ke konečnému zpracování
- download – licence je dokončena a uživatel si může stáhnout licenční soubor

Tabulka licence_cat uchovává záznamy o licenčních skupinách

Tabulka product obsahuje informace o podporovaných produktech

Tabulka product_cat uchovává názvy kategorií, do kterých jsou produkty rozděleny.

Tabulka dotazy je asociativní tabulkou a řeší vztah M:N mezi tabulkami `users`, `userType` a `dotazy_cat`. Jsou zde uloženy záznamy o dotazech od registrovaných uživatelů. Atributy `user1read` a `user2read` slouží ke zjištění, zdali zaslaný dotaz či odpověď byly přečteny.

Tabulka dotazy_cat uchovává kategorie, které jsou nastaveny pro dotazy.

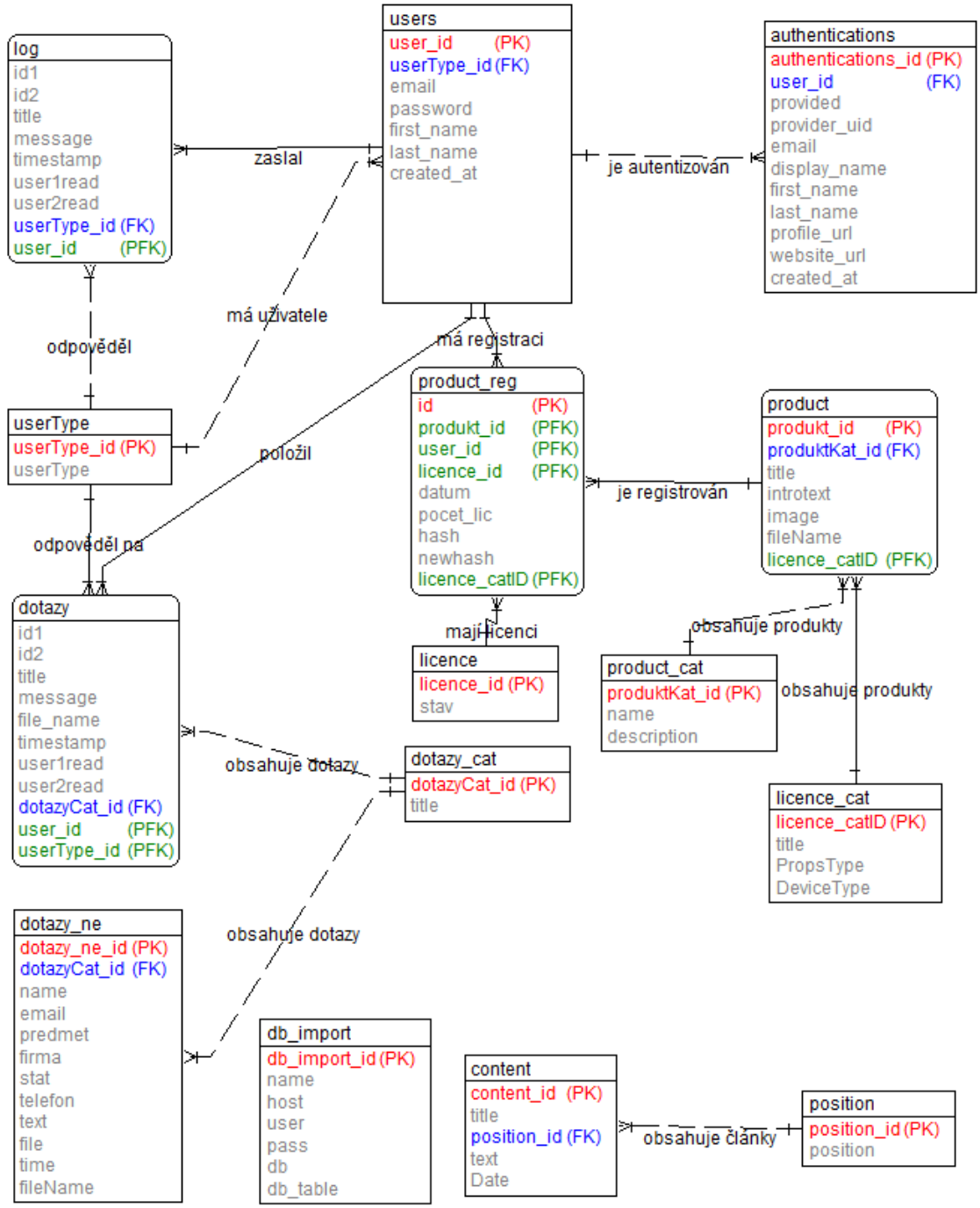
Tabulka dotazy_ne je obdoba tabulky dotazy, která slouží pro uložení informací od uživatelů, kteří nejsou registrovaní.

Tabulka db_import slouží pro nastavení přihlašovacích údajů k jednotlivým databázím pro import produktů.

Tabulka log je asociativní tabulkou, která řeší vztah M:N mezi tabulkami `users` a `userType`. Obsahuje záznamy o chybových logů.

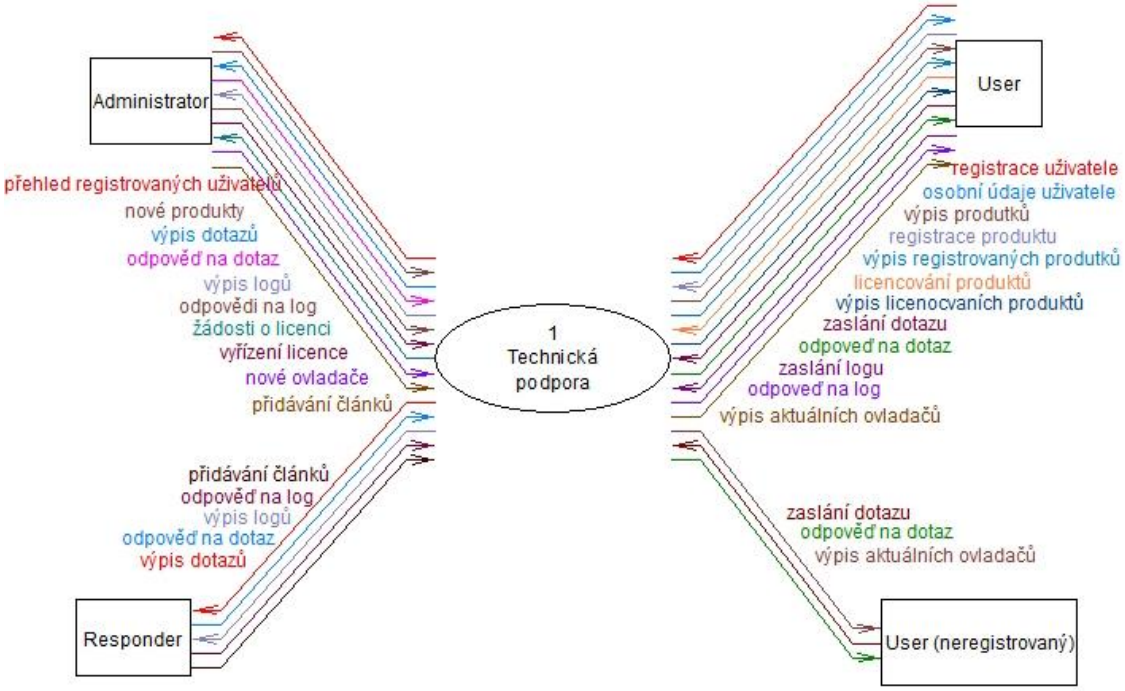
Tabulka concent zahrnuje záznamy o článcích (textech), které jsou vypsány uživateli na stránkách technické podpory.

Tabulka position uchovává pozice, dle kterých jsou zobrazovány jednotlivé články.



Obr. 3.2 - E-R diagram

Databázový systém MySQL, který se stará o uchovávání dat pro aplikaci technické podpory, využívá formát uložiště dat (storage engine) InnoDB. Tento typ uložiště mimo jiné v rámci relační databáze umožňuje využít výhody referenční integrity, která zajišťuje korektnost při zadávání údajů s cizími klíči.



Obr. 3.3 – Kontextový diagram celého systému

3.3 Struktura webové aplikace

Struktura webové aplikace je koncipována dle návrhového modelu MVC, kde platí striktní rozdělení prezenční, aplikační a doménové logiky v systému. Rozdělení aplikace na moduly *model*, *view* a *controller* určuje odpovědnost jednotlivých zdrojových kódů. Vztahy mezi komponentami přesně vymezují procesy, které se odehrávají během zpracování uživatelova požadavku a odezvy na tento požadavek.

Pro nastavení MVC slouží `application`, `model` a `conroller`. Tyto třídy obsahuje soubor *base.php*, kde třída `application` načte příslušný *controller* pro zpracování dat. V případě potřeby je načten *model*, pomocí kterého lze získat přístup k funkcím pracujících se záznamy získaných nejen z databáze.

Jednotlivé třídy jsou koncipovány do stejnojmenných souborů, které jsou obsaženy v adresářích pojmenovaných dle návrhového modelu. Toto rozdělení má za úkol snazší orientaci při úpravě jednotlivých tříd. Jako hlavní třídy systému webové

aplikace jsou považovány třídy nesoucí názvy uživatelských rolí v jednotlivých modulech (Obr. 3.4). Pro modul *controllers* to jsou `admins`, `users` a `responder`, k nim příslušné třídy v modulu *models* (`admin`, `responder` a `user`).

Třídy **controllers** obecně obsahují funkce pro řízení dat mezi *view* a *modelem*. Funkce ve většině případů mají následující podobu. Ve funkci je nejprve nutné vytvořit odkaz na *model*, který daná třída bude využívat.

```
$user = $this->loadModel( "user" ); //při využití modelu user
```

V případě webových formulářů nebo při předávání hodnot mezi jednotlivými stránkami jsou načítány hodnoty metodou $_POST nebo $_GET. Metoda $_POST slouží k přenášení dat pomocí proměnných, které nejsou vidět. Data se přenáší skrytě mezi *view* a *controllerem*. Oproti tomu metoda $_GET přenáší data prostřednictvím URL adresy a není vhodná pro přenos choulostivých dat.

```
if( count( $_POST ) ){ $id = $_GET["id"]; …
```

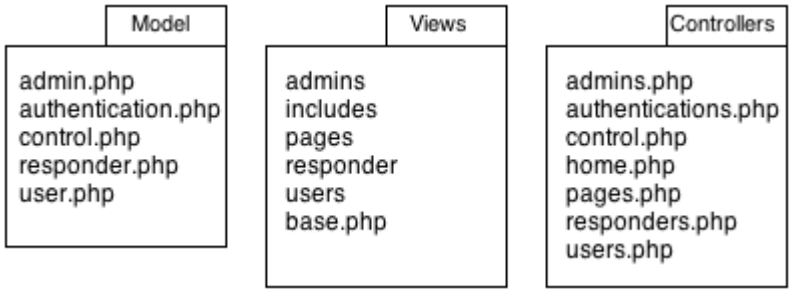
Načtené hodnoty se dále předají načtenému *modelu* prostřednictvím příslušné funkce. Popřípadě se načtou hodnoty do objektového pole a předají se k následnému zobrazení ve *view*.

```
$setRead = $user->setRead($id,'dotazy');
$getDotaz['data']= $user->getDotaz($id);
$this->loadView("users/myDotazyRead",$getDotaz);
```

Ve *view* neboli pohledu je opět adresářová struktura rozdělena na složky s názvy uživatelských rolí. Každý adresář obsahuje soubory, v nichž je zapsána výsledná podoba webové stránky, která je řízena kaskádovými styly (CSS).

3.3.1 Adresářová struktura

Adresářová struktura aplikace je koncipována dle návrhového vzoru MVC. Základní složky představují jednotlivé komponenty *model*, *view* a *controllers*, kde jsou definovány jejich třídy. Použitá struktura odděluje složky jádra aplikace od složek, které s aplikací jen komunikují (knihovny) nebo slouží k uchování záznamů (licence, logy, dotazy). Důležitou součástí je složka „public“, která obsahuje vizuální podobu webové aplikace za pomocí kaskádových stylů.



Obr. 3.4 – Adresářová struktura MVC komponent

Popis důležitých složek:

- `application/` - třídy vztahující se k chodu aplikace
- `controllers/` - akční controllery, které řídí aplikaci a obsluhují požadavky uživatele
- `models/` - třídy, které řeší doménovou logiku aplikace
- `public/` - obsahuje soubory pro vizuální podobu stránek uživatelů systému
- `view/` - obsahuje layout stránek
- `libs/` - knihovny PHP kódu (třídy, funkce)
- `dotazy/` - dotazy od registrovaných i neregistrovaných uživatelů
- `licence/` - šifrované licenční soubory
- `logy/` - chybové logy registrovaných uživatelů

3.4 Uživatelské role

Uživatelé technické podpory lze rozdělit do dvou kategorií. První kategorie obsahuje běžné uživatele, kteří mají přístup pouze k veřejnému obsahu stránek, kde naleznout popis produktů, základní ovladače k přístrojům a publikované texty.

Druhá kategorie zahrnuje uživatele, kteří mají v rámci aplikace přidělenou uživatelskou roli a mohou tak plně využít nabídku služeb. V rámci aplikace existují tři základní role, které se liší rozsahem oprávnění.

- **Administrator** je hlavní a nejdůležitější uživatelskou rolí. Kromě přístupu ke všem uživatelským účtům a k jejich nastavení má také přístup k vytváření a kontrole licencí. Stejně jako *Responder* má i *Administrator* možnost odpovídat na položené dotazy a doručené logy. Funkce, které má administrátor k dispozici jsou znázorněny v kontextovém diagramu celého systému (Obr. 3.3).

- **Responder** přejímá některá oprávnění od *administratora*. Kromě možnosti odpovídat na dotazy a zaslané logy od uživatelů může vytvářet a editovat publikované texty.
- **User** je základní role v aplikaci. Znázorňuje zákazníka, který využívá všechny služby a funkce rozšířené jeho registrací do systému. Nejdůležitější službou, kterou může využít, je licencování zakoupeného produktu.

4 Funkce systému

4.1 Přihlášení a registrace uživatelů

4.1.1 Standardní přihlášení

K aplikaci technické podpory se mohou uživatelé přihlásit několika způsoby. Prvním způsobem je klasický přístup pomocí emailu a hesla. Tato možnost přihlášení nejprve vyžaduje registraci na internetových stránkách technické podpory za využití databázových tabulek `users`, `userType` popřípadě `authentications`. Registraci zajišťuje funkce *register()*. Pro zajištění bezpečnosti byla použita PHP funkce hash() s algoritmem sha512, kde zabezpečení zajišťuje tzv. solení hesla (salt), které spočívá v přidání řetězce k heslu.

```
function register (){
…
//nastavení hesla
$password = hash("sha512", $password . "??????");
// uložení uživatele
$new_user_id = $user->create( $email, $password, $first_name, $last_name);
$_SESSION["user"] = $new_user_id;
//přesměrování na profil uživatele
$this->redirect( "users/profile" );
…
}
```

Jestliže uživatel vlastní účet na sociální síti Facebook, Google+, Twitter nebo Linkedin, může využít přihlášení prostřednictvím jeho sítě. Jakmile se přihlásí, proběhne autentizační proces, který ověří, zda uživatel je již v databázi či nikoli.

Pokud uživatel v databázi není (nebyla nalezena emailová adresa), je dotázán, zdali chce svůj účet spojit se stránkami technické podpory. Pokud ano, funkce *create()* vytvoří nového uživatele, který bude ověřován identifikátorem *provider_uid*. Tím jsou získány základní informace o uživateli, které jsou uloženy v databázi a slouží k ověření.

V případě, že se uživatel v databázi již nachází a ověřovací token souhlasí s *provider* a *provider_uid*, je přihlášen pomocí své sociální sítě.

4.1.2 Přihlášení přes sociální sítě

- **Facebook login**

Pro využití přihlašování prostřednictvím Facebook účtu, bylo nutné na stránkách http://developers.facebook.com/ vytvořit a zaregistrovat aplikace, které budou zajišťovat komunikaci jak pro webovou aplikaci, tak pro doplněk v desktop aplikaci. Tyto jsou aplikace identifikovány pomocí *App ID* a *App Secret*.

Jelikož aplikace jsou navrženy pomocí PHP a C# .NET, byly využity *Facebook PHP SDK*[8] a Facebook SDK for .NET[9], které nabízí metody pro autentizaci prostřednictvím protokolu OAuth 2.0.

Nejprve bylo nutné vygenerovat URL adresu přístupu, která obsahuje parametry *app_id, app_secret, redirec_uri, responose_type* a *scope*. Nejdůležitějšími parametry jsou *redirect_url* (stránka volaná po úspěšném přihlášení - callback) a *scope. Scope* představuje souhrn práv a informací (email, user_about_me, user_webcsite, read_stream, offilen_access), které uživatel poskytne webové aplikaci. Některé údaje jsou vedeny jako veřejné, a proto u nich nemusí být žádné oprávnění. Jelikož aplikace požaduje unikátní email pro přihlášení, bylo nutné zajistit přístup k osobnějším údajům.

Vygenerovaná URL adresa pro autentizaci:

```
https://www.facebook.com/dialog/oauth?client_id=170017139821589&redirect_uri=https://www
.facebook.com/connect/login_success.html&response_type=token&display=popup&scope=email,u
ser_about_me,read_stream
```

Jakmile uživatel prostřednictvím Login Dialogu přístup schválí, webová aplikace při *callbacku* obdrží autorizační token. Informace obsažené v autorizačním tokenu jsou součástí Facebook Graph API, pomocí něhož jsou získány požadované záznamy o uživateli, které jsou načteny do proměnných a následně uloženy do databáze.

```
$this->user->profile->identifier = (array_key_exists('id',$data))?$data['id']:"";
$this->user->profile->displayName = (array_key_exists('name',$data))?$data['name']:"";
$this->user->profile->profileURL = (array_key_exists('link',$data))?$data['link']:"";
$this->user->profile->email = (array_key_exists('email',$data))?$data['email']:"";
```

Obr. 4.1 – Facebook – autorizace údajů

- **Linkedin a Twitter**

Pro autentizaci pomocí Linkedin a Twitter účtu byla jako u předešlého poskytovatele nutná registrace webové aplikace. Na základě registrace bylo přiděleno *Api key/consumer_key* a *Secret key/consumer_secret*, které slouží pro propojení webové a desktopové aplikace s účty.

Na základě dostupných informací a z důvodu ověřené kompatibility je autentizační proces ověřován prostřednictvím protokolu OAuth 1.0, kde bylo implementováno rozhraní pro programování aplikací LinkedIn[6] a Twitter[18]. Autorizace probíhá opět prostřednictvím URL adresy, kde jsou nastaveny příslušné parametry a následná oprávnění.

URL pro autentizaci LinkedIn

```
http://api.linkedin.com/",oauth_consumer_key="oa4uq8m42hl0",oauth_token="fd3a8434-fbd7-4cca-8a87-d8878a1e3933",oauth_signature_method="HMAC-SHA1",oauth_signature="VCDhagWugOmDkzplsEkdxZgMTK8%3d",oauth_timestamp="1365427960"
```

Získání informací:

```
https://api.linkedin.com/v1/people/~:(id,first-name,last-name,public-profile-url)
```

Získané informace o uživateli jsou předány v XML formátu, který je nutno načíst a posléze jednotlivé elementy uložit do příslušných proměnných k dalšímu použití.

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"yes\"?>
<person>
<id>LinkedIn ID</id>
<first-name>Jméno</first-name>
<last-name>Příjmení</last-name>
<public-profile-url>Adresa profilu</public-profile-url>
</person>
```

Linkedin ani Twitter API prostřednictvím tokenu nepodporují získání emailové adresy, kterou aplikace technické podpory využívá jako identifikátor uživatele. Jak uvádí Kirsten Jones[13] ze společnosti Linkedin, je to z důvodu zachování soukromí uživatele a emailová adresa se musí získat od uživatele jiným způsobem. Na základně této skutečnosti je uživatel při procesu autorizace vyzván, aby zadal prostřednictvím formuláře emailovou adresu, kterou bude využívat. Tato adresa se společně s *provider_id* a ostatními získanými záznamy o uživateli uloží do databáze pro opětovné přihlášení.

- **Google**

Autentizace pomocí Google je prováděna pomocí API přístupu, který je identifikován podle přiděleného *Client ID* a *Client secret*. Každý přístup dále obsahuje adresu přesměrování. Ověřovací proces je reprezentován protokolem OAuth 2.0[14], který zajišťuje přenos uživatelských oprávnění, s kterými může webová aplikace nakládat. Podobně jako u Facebookového přihlášení byly z autorizačního tokenu získány požadované informace o uživateli a pomocí funkce *getUserProfile()* postoupeny k dalšímu zpracování.

4.1.3 Správa produktů

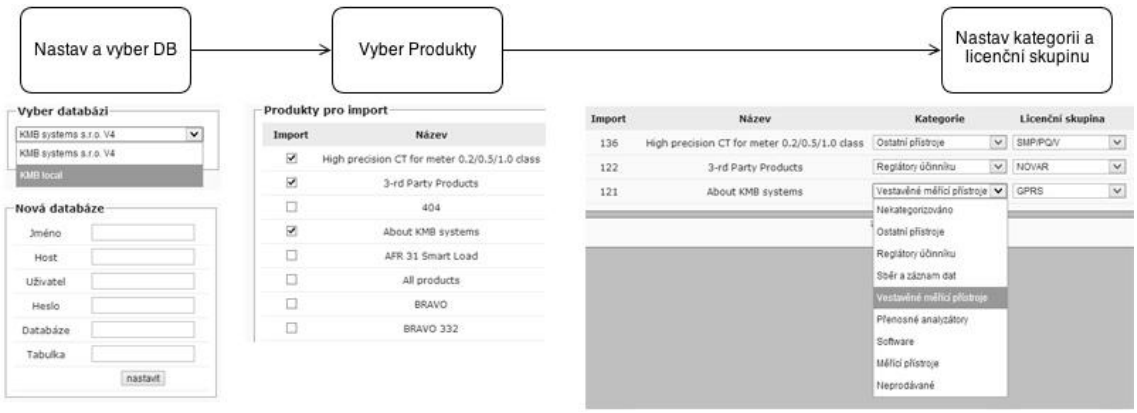
Správa produktů je jednou z hlavních součástí technické podpory. Všechny informace o produktech jsou uchovávány v databázi v tabulce `product` a jejich kategorie v tabulce `product_cat`.

Přidání produktů

Nové produkty lze přidávat pouze prostřednictvím uživatelského účtu s rolí *administrator,* který má dvě základní možnosti přidání.

První možnost představuje vyplnění formuláře pro nový produkt, kde se nastaví základní informace, tj. název, kategorie, kategorie licence, popis a příslušný obrázek. Další možnost, která se bude využívat především pro přidání více produktů současně, je **import**.

Import produktů pracuje s tabulkou `db_import`. Proces importu probíhá ve třech krocích (Obr. 4.2). V prvním kroku se nejprve nastaví informace o databázi, ze které se budou produkty načítat a uloží se do databáze, kterou využívá technická podpora. Poté se zobrazí nabídka uložených databází, ze kterých si administrátor vybere. V dalším kroku se vypíší všechny produkty[2]. Poslední krok slouží pro nastavení kategorií, do kterých budou produkty zařazeny a k nastavení licenčních skupin.



Obr. 4.2 – Import produktů

Registrace a licencování produktů

Při nákupu má zákazník možnost zaregistrovat si daný produkt na stránkách technické podpory a tím získat přístup k rozšířeným službám. Pro tyto účely jsou v databázi vyčleněny tabulky `product_reg` a `licence`, které v sobě uchovávají informace o registrovaných a licencovaných produktech.

Proto, aby mohl uživatel produkt plně využívat, musí ho nejprve zaregistrovat a posléze licencovat. Registrace produktu se provádí dvěma způsoby. První možností je registrování prostřednictvím internetových stránek technické podpory. Druhou možností je využití doplňku technické podpory, který je implementován do desktopové aplikace ENVIS. V obou případech je volána funkce *setProduct_reg()* případně metoda *setProduct_reg(int product_id),* díky které se do databáze vloží informace o daném produktu a uživateli do databáze (tabulka `product_reg`). Ten má dále možnost zobrazit si přehled všech registrovaných produktů.

[2] V případě CMS Joomla se vypíší všechny články v kategorii produkty.

Webová aplikace – PHP

```
function setProduct_reg(){
$user = $this->loadModel( "user" );
$reg_data = $user->setProductReg( $_SESSION["user"], $_GET["product_id"]);
$produkt['data'] = $user->getMyProduct_reg($_SESSION["user"]);
$this->loadView("users/myProduct",$produkt);
}
```

Deskotp aplikace - C#

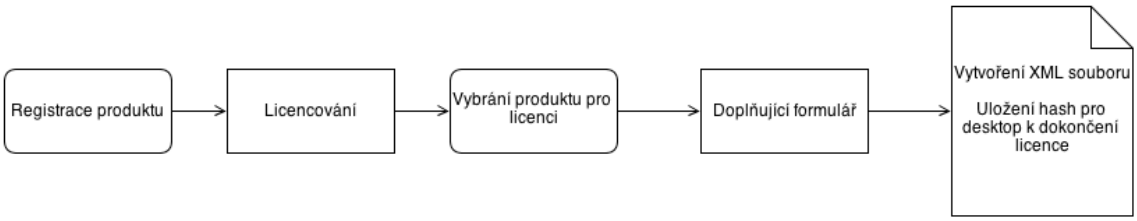
```
public void setProductReg(int product_id) {
string query = "INSERT INTO product_reg ( user_id, licence_id, product_id, datum)
                VALUES ('" + user_id + "','1','" + product_id + "', NOW() ) ";


if (this.OpenConnection() == true) {
MySqlCommand cmd = new MySqlCommand(query, conn);
Form1.NaseProduktyControl1.btn_registerProduct.Visible = false;
Form1.NaseProduktyControl1.lbl_productReg.Visible = true;
cmd.ExecuteNonQuery();

this.CloseConnection();
 }
}
```

Licencování produktů se může provést prostřednictvím doplňku v desktop aplikaci ENVIS, případně započít přes webovou aplikaci. Licenční proces se skládá z několika kroků (Obr. 4.3 a Obr. 4.4).

Jestliže je licencování prováděno prostřednictvím webové aplikace, uživatel si vybere produkty, nastaví příslušný počet licencí, případně doplní informace (název společnosti, email, tel. číslo a číslo faktury) a požadavek odešle. Požadavek ve formě formuláře je následně zpracován funkcí *setSend()*, v níž je vytvořena licence a aktualizována tabulka pro správu registrovaných produktů `product_reg` v databázi. Zde je mimo jiné zapsán pomocný hash, který nese název dočasného licenčního souboru. Hash neboli kód je odeslán žadateli k dalšímu kroku. Funkce *setSend()* obsahuje také odkaz na funkci, která vytváří dočasný licenční XML soubor, který uchovává informace o licencovaných produktech.

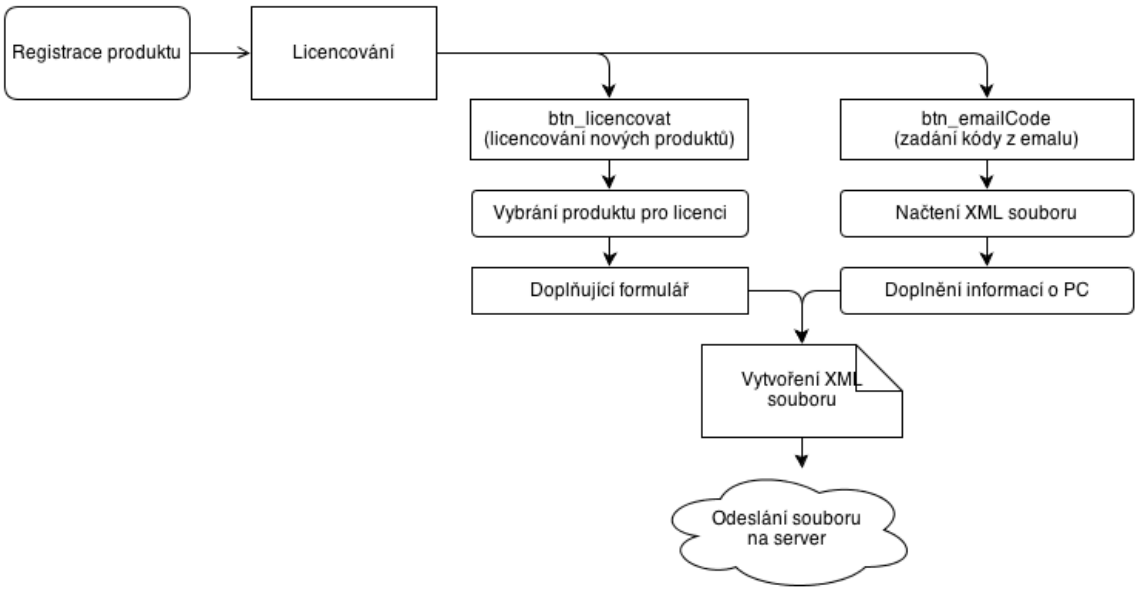
Má-li zákazník ve svém počítači nainstalovanou aplikaci ENVIS s doplňkem technické podpory, může licencování provést skrze ni. Po přihlášení a zvolení položky „Licencování“ jsou zákazníkovi nabídnuty dvě možnosti.

První možností je dokončení licenčního procesu, který byl zahájen přes webovou aplikaci a zákazník tak musí zadat kód, který obdržel emailem. Pomocí metody *updateXML()* a tohoto kódu je načten dočasný XML soubor a do něj připsány informace o počítači, pro který je licencován. Aktualizovaný soubor se uloží na server.



Obr. 4.3 – Licenční proces přes internetovou aplikaci

Druhá možnost spočívá v opětovném vybrání produktů, které chce zákazník licencovat a doplnění informací s pojenými s licencí. Pomocí metody *setLicence()* je opět aktualizována tabulka `product_reg` a vytvořen XML soubor, který také obsahuje informace o licencovaném počítači. Soubor je opět zašifrován a uložen na server.



Obr. 4.4 – Licenční proces přes desktop aplikaci

Vytvoření a uložení licenčního souboru na server

Soubory, které jsou určeny k licencování nebo k jejich zprostředkování, jsou vytvářeny jak z webové aplikace (dočasný licenční soubor), tak i prostřednictvím doplňku technické podpory. Ten je instalován do desktop aplikace ENVIS.

Je-li XML soubor vytvořen přes webovou aplikaci, je použito rozhraní DOM[15] (Document Object Model), které je standardním rozhraním pro práci s dokumenty XML tvořené konsorciem W3C. V tomto rozhraní jsou definovány způsoby, jakými se XML dokumenty mapují na hierarchii objektů v paměti. Každá část dokumentu (element, atribut či textový uzel) odpovídá v paměti jednomu objektu, který

je instancí třídy *DOMDocument*. Struktura dočasného souboru obsahuje tři hlavní elementy.

Element ReqDevice obsahuje potomky *ArrayOfUnsignedInt* a *unsignedInt*, ve kterých jsou uloženy informace o licencovaných produktech. Informace typu *DeviceType, PropsType* představuje hexadecimální hodnotu zápisu dle typu produktu, který je zařazen do určité licenční skupiny (např. Pro produkty NOVAR – PropsType:0x0040; DeviceType:0x0000) a převeden do desítkového formátu. Všechny licenční skupiny jsou uložené v tabulce `licence_cat` v databázi. *UnsignedInt* element dále obsahuje počet licencí produktu, které zákazník požaduje.

Element RemovedLicenses, obsahuje záznamy o počtu odebraných licencí v poslední žádosti o změnu či přidělení licence.

Element ListVersionInfo představuje seznam elementů *string*, který uchovává informace o společnosti žádající o licenci.

Celý soubor je z důvodu bezpečnosti zašifrován algoritmem Rijndael a uložen na serveru, kde čeká na další zpracování. Při využít aplikace ENVIS k licencování či dokončení licence, jsou použity třídy *XMLDocument* a *MemoryStream*[10], pomocí kterých je vytvořen nový licenční soubor nebo aktualizován dočasný. V případě provádění updatu dočasného souboru, je soubor načten do datového proudu a do příslušného elementu se vloží informace k dokončení licencování. Výsledný soubor je opět zašifrován a odeslán na server, což obstarává metoda *UploadFile()* s definicí třídy *FtpWebRequest*.

Struktura XML souboru:

```
<?xml version="1.0" encoding="utf-16"?>
<LicStruct
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <ReqDevices>
    <ArrayOfUnsignedInt>
      <unsignedInt>4376</unsignedInt>
      <unsignedInt>80</unsignedInt>
      <unsignedInt>1</unsignedInt>
    </ArrayOfUnsignedInt>
  </ReqDevices>
  <LicensedDevices />
  <RemovedLicenses>0</RemovedLicenses>
  <ListVersionInfo>
```

```
    <string>#$COMPANY$#Název firmy</string>
    <string>#$EMAIL$#firma@firma.cz</string>
    <string>#$PHONE$#+420000111222</string>
    <string>#$INVOICENR$#0987654</string>
  </ListVersionInfo>
</LicStruct>
```

4.1.4 Zpracování licenčních souborů

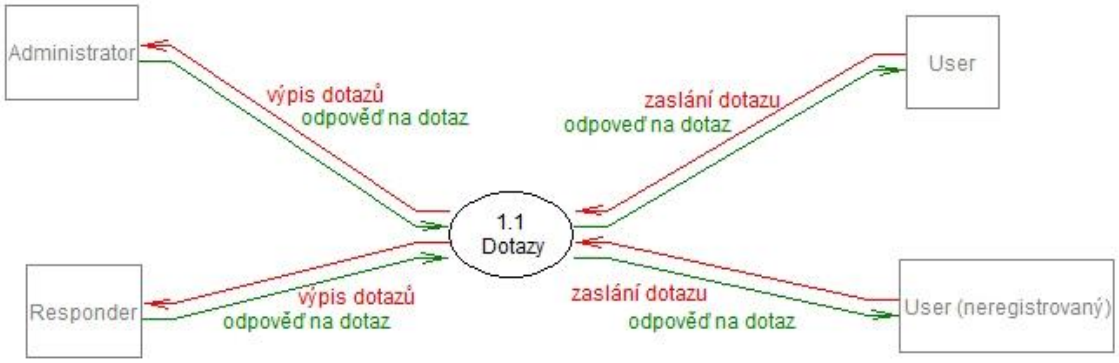
Zpracování licenčních souborů je prováděno přes administrátorské prostředí v sekci *Registrace a licence*, kde má administrátor přehled o registrovaných produktech a o průběhu licencování, které je znázorňováno statusy:

- **register** – produkty registrované zákazníkem
- **net_waiting** – žádost o licenci, která je odeslána přes webovou aplikaci a čeká na dokončení v desktop aplikaci
- **wait** – žádost o licenci odeslanou přes desktop aplikaci, popřípadě dokončení licenčního procesu pomocí zaslaného kódu, zákazník čeká na vyřízení licence
- **download** – žádost o licenci je vyřízena a zákazník si může stáhnout licenční soubor

V případě, že administrátor obdrží email, že byl zaslán požadavek na licencování a status nabývá hodnoty *wait,* může administrátor stáhnout zákazníkem vytvořený soubor. Jestliže jsou data v souboru o licenci korektní, je vytvořen licenční soubor, který opět nahrán prostřednictvím webové aplikace na server. Odtud si jej může zákazník pomocí webové či desktopové aplikace stáhnout.

4.2 Správa dotazů

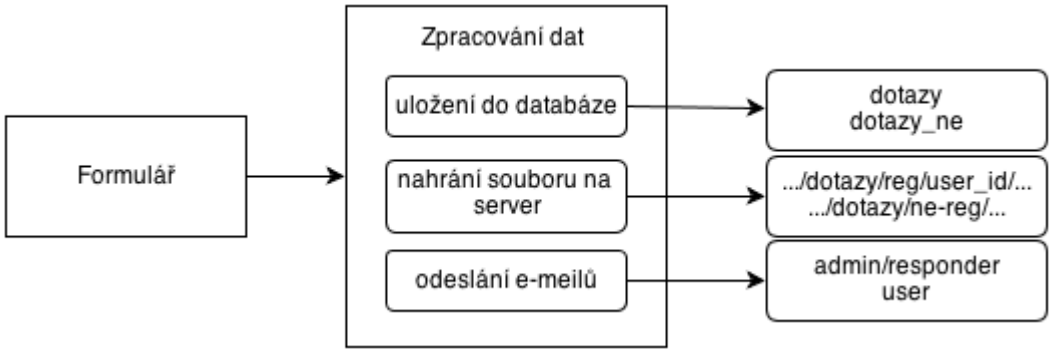
Mezi další funkce, které mohou zákazníci využívat, patří odesílání dotazů. Jednotlivé údaje jsou ukládány do tabulek `dotazy` nebo `dotazy_ne`(neregistrovaní uživatelé) v databázi. Funkce pro zasílání dotazů zahrnuje všechny uživatelské role systému technické podpory (Obr. 4.5), kde hlavní nastavení (založení nových kategorií) a kontrolu provádí administrator. Pro zodpovězení dotazů je v systému vytvořena speciální role responder.



Obr. 4.5 – DF diagram 2. úrovně - Dotazy

Odesílání dotazů lze provést jak z webové, tak z desktopové aplikace. V případě nutnosti lze k dotazu připojit soubor, který bude blíže specifikovat dotaz. Jestliže zákazník využívá webovou aplikaci, je odeslání prováděno prostřednictvím formuláře, kde je rozlišováno, zdali je zákazník registrován a přihlášen či nikoli. Na základě této skutečnosti je použita tabulka v databázi.

V případě, že uživatel není registrován, může využít pouze webovou aplikaci, kde musí vyplnit kontrolní kód z obrázku (CAPCHA). Při následném odeslání dotazu jsou jeho osobní údaje uloženy do databáze. Registrovaní uživatelé disponují značnou výhodou, jelikož jejich dotazy jsou koncipovány do formy diskuse, mají přehled o průběhu řešení.



Obr. 4.6 – Proces zpracování dotazů

Proces zpracování dotazů (Obr. 4.6) pro registrované a neregistrované uživatele je totožný, až na místo uložení připojeného souboru a tabulky v databázi pro uložení záznamů. Větší rozdíly jsou především ve způsobu odeslání souborů na server přes desktopovou aplikaci.

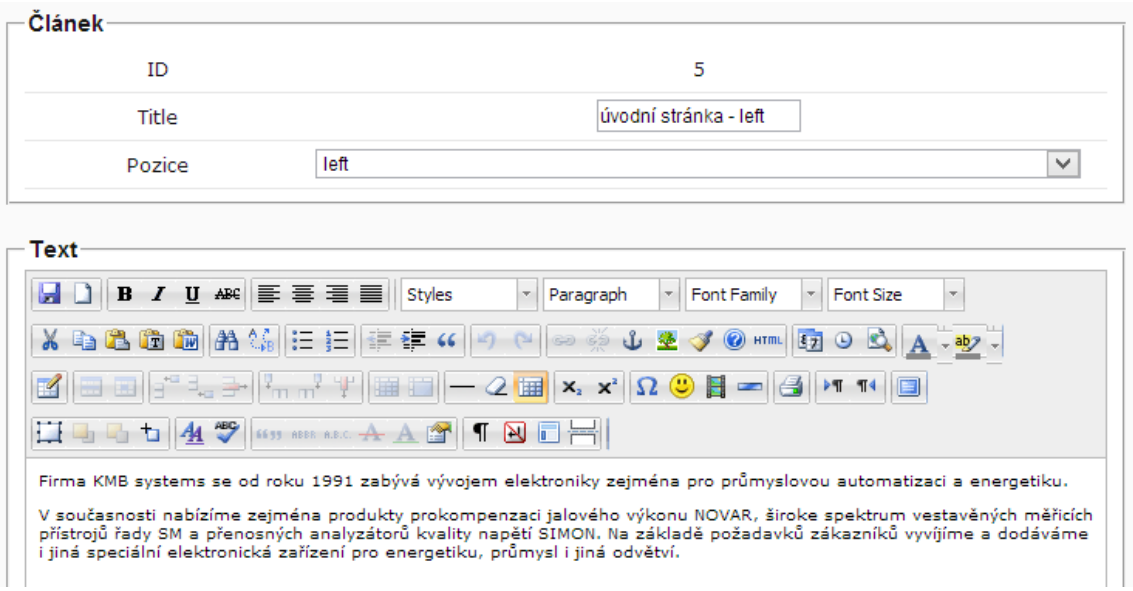
- **Webová aplikace** – Po odeslání formuláře jsou data předána třídě `users` představující *controller*. Zde za pomoci funkce `dotazNew()` jsou data

zpracována. Je-li připojen k dotazu soubor, vytvoří se uživatelská složka (jestli doposud nebyla vytvořena) na serveru, která bude využívána pro úschovu souborů uživatele. V dalším kroku je soubor přejmenován na název, který je tvořen datem a uživatelským id. Po úspěšném nahrání souboru na server je přidán záznam do databáze a odeslán oznamovací email responderovi.

- **Desktopová aplikace** využívá, až na rozdíly v implementaci, stejného principu jako aplikace webová. Ovšem oproti ní, kdy uložení souborů probíhá na stejném serveru, musí desktopová aplikace vytvořit připojení k FTP serveru. Spojení je uskutečňováno za pomoci třídy `FtpWebRequest`, kde je nutné pro přístup nastavit uživatelské jméno a heslo.

4.3 Články

Články v systému technické podpory představují souvislé texty, které jsou obsaženy na internetových stránkách nebo v desktopové aplikaci a jsou uloženy v databázi v tabulce `content`. Články může vytvářet a aktualizovat administrátor a responder za pomoci WYSIWYG editoru TinyMCE. Tento editor je platformě nezávislý JavaScript HTML editor textu a založený na myšlence WYSIWYG s open source licencí vydanou LGPL u Moxiecode Systems AB [17]. Pro využití editoru v HTML kódu je nejprve zavolán Javascript soubor obsahující zdrojový kód TinyMCE. Dále je provedena inicializace, která má za úkol nastavení editoru (nastavení jazyka, přidání pluginů pro správu textu).



Obr. 4.1 – Editace článku s TinyMCE editorem

Po vytvoření článku je nutné přiřadit pozici, na které se bude text zobrazovat. Systém umožňuje výběr ze 14 pozic pro webovou a 7 pozic pro desktopovou aplikaci. Jelikož jsou články do databáze ukládaný prostřednictvím TinyMCE editoru i s HTML tagy, bylo nutné vybrat vhodný způsob načítání pro korektní načítání článků v desktop aplikaci. Jako nejlepší řešením se ukázalo využít komponentu webBrowser, která nenačítá text tak, jak je uložen v databázi, ale při načítání akceptuje HTML tagy.

4.4 Odeslání chybových LOG souborů

Funkce odeslání chybových logů představuje speciální případ dotazů. V desktopové aplikaci je možné vygenerovat LOG soubor, který obsahuje záznamy o činnosti programu. Jestliže aplikace vykazuje chybu, zákazník má možnost tento soubor odeslat prostřednictvím webového či desktopového formuláře. Vizualizace a struktura výpisu jednotlivých LOG souborů je totožná s výpisem dotazů.

4.5 Nabídka aktuálních ovladačů a dokumentů

Stránky technické podpory nabízejí přehled aktuálních ovladačů a dokumentů k příslušnému produktu. Jejich správu zajišťuje současná webová prezentace firmy KMB systems s.ro. přesněji, doplněk redakčního systému Joomla Phoca Download. Tento doplněk v podobě download manageru obsahuje komponenty, moduly a pluginy. Technická podpora využívá adresáře `phocadownload` uloženého na serveru, který v podsložkách obsahuje jednotlivé dokumenty.

Výpis dokumentů zajišťuje funkce `getFileByName()` s parametry `filepath` a `fileName`, která prochází všechny soubory a složky uložené v požadovaném umístění. V případě, že název souboru obsahuje řetězec s názvem produktu, je tento soubor vypsán.

4.6 Integrace do stávající webové prezentace

Současné internetové stránky společnosti KMB systems s.r.o. jsou provozovány na redakčním systému Joomla 2.5.9. Proto bylo nutné navrhnout takové doplňky, které by byly v případě aktualizace CMS kompatibilní. Do internetových stránek byl implementován doplněk pro přihlášení uživatele k aplikaci technické podpory a dále formulář pro rychlé zasílání dotazů.

Integrovaný přihlašovací formulář je tvořen souborem *login.php*, který obsahuje script pro přihlášení přes sociální sítě a možnost založení nového účtu při přesměrováním na webovou aplikaci technické podpory. Pro jeho zobrazení je nutná instalace komponenty *Jumi*, která umožňuje zobrazení PHP a HTML scriptů ze zadaného umístění.

V případně doplňku pro zasílání rychlých zpráv byl upraven současný formulář, který umožňuje odeslání krátké zprávy na přednastavený email. Funkce formuláře byla rozšířena o práci s databází. Jestliže zákazník odešle vyplněný formulář, administrátor popřípadě responder obdrží email obsahující údaje z formuláře, které jsou také uloženy k dalšímu zpracování do tabulky `dotazy_ne` v databázi. Práci s databází v CMS Joomla umožňuje třída *JFactory*, která v sobě obsahuje metodu *getDbo()* pro připojení k databázi a implementaci rozhraní *JDatabaseDriver*.

5 SEO optimalizace webové aplikace

SEO (Search Engine Optimization), neboli optimalizace stránek pro vyhledávače, je sbírka doporučení, která se snaží webovou stránku maximálně zviditelnit v internetových vyhledávačích. Názory na efektivitu optimalizace jsou značně rozlišné. Ovšem moderní internetové stránky by měly obsahovat tyto doporučení alespoň v základní podobě.

Základní optimalizace je tvořena z viditelných faktorů na stránce, mezi které patří nejen titulek stránky, její popis a klíčová slova, ale také tvorba samotného obsahu stránek. Tyto činitele může nastavit administrátor v „backhandu“ aplikace. Editoři by měli dodržovat přednastavené kaskádové styly, aby zamezili nárůstu velikosti stránky.

Validita

Validita webových stránek zvyšuje předpoklady ke správnému zobrazení obsahu většímu počtu návštěvníků, včetně těch, co využívají nestandardní prohlížeče či zařízení (mobilní telefon, PDA). Pro roboty vyhledávačů (Google, Seznam a jiné) jsou přístupnější pro indexaci a následné zařazení do databáze. Stránky, které jsou validní, se vyznačují svoji rychlostí pří načítání, kompatibilitou s budoucími verzemi prohlížečů a především srozumitelností i při nesprávném načtení obrázků nebo kaskádových stylů.

Duplicita obsahu

Duplicitní obsah vzniká, jestliže více různých URL adres směřuje na totožnou stránku nebo na stránky s podobným obsahem. Vyhledávače si obvykle vybírají pouze jednu URL a ostatní i se zpětnými odkazy ignoruje. Častým typem duplicitního obsahu jsou totožné stránky, na které je odkazováno z různých míst např.:

- `http://kmbsystems.cz`
- `http://kmbsystems.cz/index.php`
- `http://www.kmbsystems.cz`
- `http://www.kmbsystems.cz/index.php`

Ošetření duplicit je u webové aplikace v první řadě realizováno v souboru .htaccess, který slouží k úpravě vlastností serveru bez zásahu správce. Nejprve je zajištěno, že před URL se vždy zapíše „www“. Druhá část kódu pak zajišťuje zobrazení úvodní stránka ve tvaru domény.

- `RewriteCond %{HTTP_HOST} ^kmbsystems\.cz`
- `RewriteRule (.*) http://www.kmbsystems.cz/$1 [R=301,L]`
- `RewriteCond %{HTTP_HOST} kmbsystems \.cz/ index.php?route=users/login`
- `RewriteRule (.*) http://www.kmbsystems.cz/ [R=301,L]`

Další ošetření je implementováno pouze na některé stránky. Jeho nastavení se provádí přímo v hlavičkách prostřednictvím meta tagu, který vyhledávačům naznačí, že daná stránka nemá být indexována a nemá následovat žádný z odkazů obsažených na stránce.

```
<meta name=“robots“ content=“noindex, nofollow“ />
```

Poslední metodou ochrany aplikace proti duplicitě je vyřazení složek a to pomocí vzorů v souboru robots.txt, který obsahuje příkaz „Disallow“ jenž zamezuje robotu vyhledávačů přístup do adresářů, kde je prováděno nastavení systému.

RSS

Technologie RSS kanálů umožňuje zákazníkům odebírat novinky ze stránky s výpisem produktů, aniž by byl jejím návštěvníkem. Generování kanálu produktů je prováděno dynamicky funkcí `createRSS()`, která obsahuje titulek, odkaz a prvních 200 znaků z informačního textu.

6 Zabezpečení systému

V dnešní době, kdy se zvyšují znalosti programovacích jazyků a následně tak roste počítačová kriminalita, si především mladí lidé chtějí dokázat své znalosti a tak se snaží proniknout do cizích informačních systémů – tzv. script-kiddeis. Útokům neuniknou ani kontaktní formuláře, které napadají „komentářové spamy“ zanechávající nevyžádané příspěvky v podobě reklamních odkazů či nesmyslných příspěvků.

Aby se předešlo napadení systému technické podpory, bylo použito několik forem zabezpečení.

6.1 Zabezpečení proti komentářovému spamu

Každý den jsou webové stránky navštěvovány různými naprogramovanými roboty. Nejběžnější úkol robotů je indexace obsahu stránek, ale stránky. Stránky mohou být také navštívit roboty, kteří se snaží různými způsoby zanechat na stránkách nesmazatelnou stopu. Nejběžnější způsob je zanechání různých URL adres v diskusích, popřípadě odeslání nežádoucí zprávy přes formulář. URL adresy tak mají zvýšit PageRang nebo návštěvnost stránek tvůrce robota.

Na webových stránkách technické podpory je proto použita metoda CAPTCHA – Completely automated public Turing test (plně automatický veřejný Turingův test k odlišení počítačů a lidí), která spočívá v zobrazení deformovaného textu na obrázku. Jestliže chce neregistrovaný uživatel odeslat formulář, musí opsat správný kód vygenerovaný na obrázku. Ten je následně ověřen a formulář je odeslán.

Generování obrázku probíhá pomocí funkce *captcha()*, kde je definováno pole obrázků, ze kterých je vygenerováno pozadí a přípustné znaky, které může výsledný obrázek zobrazit.

Výsledná podoba a hodnota je pro ověření a zobrazení uložena do *$_SESSION['captcha']*. V případě odeslání formuláře se porovná hodnota ze vstupního pole s hodnotou uloženou v $_SESSION['captcha']['code']. Jestliže se hodnoty shodují, formulář je odeslán.

```
Nastavení CAPTCHA
$captcha_config = array(
        'code' => '',
        'min_length' => 5,
        'max_length' => 5,
        'png_backgrounds'=>
```

```
array(dirname(__FILE__)./captcha.png',dirname(__FILE__).'/captcha2.png'),
'fonts' => array(dirname(__FILE__) . '/times_new_yorker.ttf'),
'characters' => 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789',
'min_font_size' => 24,
'max_font_size' => 30,
'color' => '#000',
'angle_min' => 0,
'angle_max' => 15,
'shadow' => true,
'shadow_color' => '#000',
'shadow_offset_x' => -2,
'shadow_offset_y' => 2
);
```

6.2 Zabezpečení před XSS

XSS neboli Cross Site Scripting je narušení webové aplikace v důsledku shromažďování neošetřených dat od uživatelů. Data jsou nejčastěji odeslána pomocí formulářů nebo přes URL, kdy je útočník modifikuje tak, že se v jejich kontextu provede podstrčený JavaScriptový kód.

Zabezpečení před XSS je implementováno pomocí PHP funkce *htmlspecialchars()* na stránky, které zobrazují výstupy předávané metodou POST či GET. Funkce tak převede řídící znaky na odpovídající HTML entity.

6.3 Zabezpečení před SQL injection

SQL injection je schopnost uživatele upravovat SQL dotaz prostřednictvím předávaných dat. Touto úpravou je možné ovlivňovat data, která se ukládají do databáze. Častým způsobem je „přilepení“ apostrofu či uvozovek a tím vznikne škodlivý kód modifikující původní dotaz. Apostrof či uvozovky změní odeslaná data tak, že řetězec, který se nachází za apostrofem, není považován za data, ale za samotný kód dotazu k provedení.

Kontrola zda-li je $id číslo

```
if(!is_numeric($cont_id)){
        $pages = $this->loadModel( "pages" );
        $error = 'Nesprávná hodnota!!!';
        $this->loadView("pages/error",$error);
        exit;
}
```

7 Testování a rozvoj aplikace technické podpory

7.1 Testování

Testování systému technické podpory bylo prováděno jak pro desktopovou aplikaci, tak pro webovou. Internetová aplikace byla vyvíjena na localhostu prostřednictvím balíčku XAMPP, který obsahuje základní prvky pro vývoj webových aplikací (PHP, phpMyAdmin, Apache HTTPD, MySQL, FileZilla FTP Server). Testování aplikace nejprve probíhalo při samotném návrhu, po nainstalování na serveru společnosti KMB systems s.r.o..

Testování při vývoji

Nastevení - Server: localhost
Verze PHP: 5.4.7
Databáze: MySQL 5.5.27

Toto testování je především zaměřeno na korektní vykonávání jednotlivých funkcí, přesné zobrazování a ošetření přístupu dle uživatelských rolí. Protože samotné PHP nenabízí možnosti přímého ladění, byly nejvíce využívány funkce pro výpis `echo` a `print_r()`. Tyto funkce zaručují rychlý výpis dat v případě kontroly při předávání mezi jednotlivými funkcemi. Výpisy dat jsou však pro běžného uživatele skryty. Jestliže nastane neočekávaná chyba za běhu aplikace, uživateli je zobrazena chybová stránka s popisem. Případné chyby, jako jsou odmítnutí přístupu (403), neexistence stránky (404) nebo vnitřní chyba serveru (500), jsou přesměrovány pomocí souboru .htaccess opět na chybovou stránku.

Testování na hostingu

Nastavení - Server: www.kmbsystems.cz
Verze PHP: 5.2.17
Databáze: MySQL 5.5.29

Pro spuštění webové aplikace na hostingu společnosti KMB systems s.r.o. bylo nutné provést základní nastavení funkcí, domovských adresářů a přesměrování.

V případě přihlašování přes sociální sítě bylo nutné vytvořit nové aplikace (Facebook) či API přístupy (Google), které jsou přesměrovány na URL adresu současné aplikace.

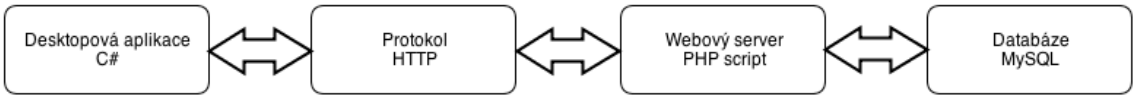
Do testování byly zahrnuty základní faktory ovlivňující webové stránky, jako je validita stránek a datová velikost. Z výsledků získaných (X)HTML validátorem na stránkách *www.w3.org*, vyplívá, že stránky neobsahují chyby a jsou validní.

Datová velikost představuje dobu, po kterou bude zákazník stránku stahovat. Protože se předpokládá, že webovou aplikaci technické podpory budou navštěvovat i uživatelé, kteří nevlastní výkonné počítače nebo dobré připojení k internetu, byla aplikace navržena tak, aby se načítala co možná nejrychleji.

Dle nástroje pro zjišťování rychlosti načítání stránek WEBPAGETEST (www.webpagetest.org) byl proveden test, který stanovil rychlost načtení úvodní stránky na čas 1.300 sekund a opětovné načtení za 0.614 sekund. Tento výsledek umožňuje rychlé načtení i u slabšího výkonu počítače. Zjištěné hodnoty ale mohou být však irelevantní, jelikož se stránky stále aktualizují.

Dalším faktorem pro testování by mohl být i výsledek ze SEO analyzátoru. Avšak využití analyzátorů může být pouze marketingový krok společností, které si tímto krokem chtějí zvýšit návštěvnost stránek. Získané procento kvality stránek, nemusí tedy odpovídat skutečnosti. Analyzátor lze tedy využít pouze jako informativní složku, dle které je možné nastavit chybějící části ve zdrojovém kódu. Pro hodnocení těchto faktorů byla využita analýza na stránkách www.seo-servis.cz, kde aplikace byla ohodnocena 93%. Tento výsledek je převážně ovlivněn nízkým počtem slov na stránce.

Testování desktopové aplikace bylo díky vývoji ve Visual studiu 2012 prováděno prostřednictvím debuggeru, který detekoval chyby k ošetření. Jelikož aplikace využívá komunikaci s databází MySQL a data ukládá na lokální server, podstatná část chybových hlášení byla výsledkem špatné komunikace nebo chybného nastavení oprávnění. Vzhledem k tomu, že desktopová aplikace byla uvažována jako možnost rozvoje softwaru ENVIS, její vývoj probíhal pouze s využitím lokální databáze. V případě, že by se aplikace stala součástí softwaru, bylo by nutné upravit třídu, která zajišťuje práci s databází. Její hlavní úkoly jsou připojení a následné odeslání a zpracování dotazů. Tyto funkce by nahradily PHP scripty uložené na serveru, prostřednictvím kterých by byly odesílány a zpracovávány dotazy. Jednou z možností, jak zajistit komunikaci mezi desktop aplikací a webovým serverem, je využití třídy `HttpWebRequest`.



Obr. 7.1 – Návrh připojení k databázi

Tato třída by mocí metod POST a GET zajišťovala předávání dat, které by z důvodu bezpečnosti musely být v prvé řadě zašifrovány. Pro zpracování požadavků lze implementovat další *controller*, který umožní komunikaci s již vytvořenými *modely* vykonávající databázové dotazy.

7.2 Možnosti rozvoje technické podpory

7.2.1 Mobilní aplikace

Vývoj celého konceptu služeb technické podpory by měl splňovat aktuální požadavky zákazníků, kteří v dnešní době začínají ve velké míře využívat mobilní aplikace. Ta by měla usnadnit a značně snížit dobu řešení problému při komunikaci mezi zákazníkem a firmou.

Aplikace by mohla obsahovat veřejnou část, kde by zákazník našel nejen aktuální přehled nabízených produktů, ale také část neveřejnou, která by byla zpřístupněna na základě registrace. Pro přihlášení do aplikace by pak zákazník využíval přihlašovací údaje z webové či desktopové aplikace s možností využití účtu sociální sítě.

Kromě přehledu registrovaných a licencovaných produktů, by mohl zákazník také využít on-line podporu prostřednictvím chatu. A to za předpokladu, že by byl příslušný zaměstnanec firmy připojen. K odeslané zprávě by mohl zákazník připojit dokument nebo fotografii znázorňující konkrétní problém.

Aplikace by využívala stávající MySQL databázi a administrátorské prostředí webové aplikace technické podpory.

Možnosti neregistrovaného uživatele:

- přehled aktuálních produktů
- přehled dokumentů
- zaslání dotazu
- přehled prodejců

Obr. 7.2 – Grafický návrh mobilní aplikace

Možnosti registrovaného uživatele:

- výpis registrovaných produktů uživatelem
- výpis licencí uživatele
- výpis logů
- zaslání dotazů či odpověď z předešlé diskuse
- možnost on-line řešení problému

Závěr

Cílem této diplomové práce bylo vytvoření technické podpory pro společnost KMB systems s.r.o.. Dále pak nevrhnout doplněk pro nabízený produkt ENVIS v podobě desktopové aplikace, který zahrnuje služby technické podpory. Dalším úkolem bylo rozšířit stávající webovou prezentaci společnosti o nástroje technické podpory.

Pro aplikaci ENVIS, která slouží k vyhodnocení naměřených dat z produktů nabízených společností, byl v prostředí Visual studia 2012 (C#) navržen doplněk. Toto rozšíření zahrnuje uživatelské nástroje technické podpory a je provázáno s webovou prezentací.

U rozvoje webové prezentace byla nejprve uvažována možnost využít jeden z nabízených open source projektů, které již obsahují prvky komunikace se zákazníky a to v podobě zadávání dotazů a jejich zpracování. Tyto systémy se však ukázaly díky specifickým požadavkům jako nevyhovující. Z tohoto důvodu byla navrhnuta aplikace založená na architektuře MVC, jejíž návrh probíhal za použití HTML, PHP, CSS a databáze MySQL. Tato aplikace nabízí uživatelské a administrační prostředí, kde je prováděna kontrola a nastavení systému. Jak doplněk pro software ENVIS, tak webová aplikace technické podpory využívají společnou databázi, což umožňuje uživateli mít svůj účet stále pod kontrolou.

Navržený systém splňuje nejen základní požadavky v podobě možnosti zasílání dotazů a hlášení chyb prostřednictvím logů, ale i specifické požadavky, které zahrnují licenční politiku. K maximálnímu využití aplikací technické podpory je nutná registrace uživatelů, kterou umožňuje moderní způsob přihlášení prostřednictvím účtů sociálních sítí Facebo, Google, LinkedIn a Twitter.

Samotný návrh byl místy doprovázen chybovými zprávami, které způsobovaly nefunkčnost aplikací. A to nejen v případě testování, ale i při vývoji. Pro návrh řešení byla využívána především diskusní fóra, kde bylo možné nastínit detailně vzniklé problémy. Určité komplikace nastaly také při testování, kdy byla webová aplikace přenesena z localhostu na hosting firmy KMB systems s.r.o. a to z důvodu odlišně nastaveného serveru. Chybová hlášení byla postupně úspěšně odstraňována a nyní je aplikace plně funkční. Prozatím ji firma využívá v testovacím provozu.

Aby systém technické podpory byl maximálně využit, měl by se dále rozvíjet. Jednou z možností by mohl být vývoj mobilní aplikace, která by se specializovala na přímou komunikaci mezi zákazníkem a firmou.

Literatura

[1] SHARP, J. *Microsoft Visual Studio C# 2008:Krok za krokem*, 1.Vyd. Brno: Computer Press 2008, 592 s. ISBN 978-80-251-2027-9

[2] JAGGER, J., SHARP, J. *Microsoft Visual C# .NET*, 1.Vyd. Mobil Media a.s. 2002, 655 s. ISBN 80-86593-27-4

[3] SIROVICH, J., DARIE, C. *SEO v PHP: Programujeme profesionálně*, 1.Vyd. Brno: Computer Press 2008, 380 s. ISBN 978-80-251-2083-5

[4] GILMORE, JASON W. *Velká kniha PHP a MySQL 5 – kompendium znalostí pro začátečníky i profesionály*, 1.Vyd. Znore Press 2007, 864 s. ISBN 80-86815-53-6

[5] VRÁNA, J. *1001 tipů a triků pro PHP*, 1.Vyd. Brno: Computer Press 2012, 465 s. ISBN 978-80-251-2940-1

Internetové zdroje

[6] APIs | LinkedIn Developer Network [Online] 2013 [cit. 2013-03-05]. Dostupný z WWW:< https://developer.linkedin.com/apis>

[7] Baray, Cristobal. 1999. The model-view-controller (MVC) design pattern. [online] 1999. [cit. 2012-04-12]. Dostupný z < http://cristobal.baray.com/indiana/projects/mvc.html>

[8] Facebook SDK for PHP – Vývojáři společnosti Facebook [online] 2012 [cit. 2012-12-05]. Dostupný z WWW:< https://developers.facebook.com/docs/reference/php/>

[9] Facebook SDK for .NET [online] 2012 [cit. 2012-12-05]. Dostupný z WWW:< http://facebooksdk.net/>

[10] Knihovna MSDN [online] 2013 [cit. 2013-04-02]. Dostupný z WWW:< http://msdn.microsoft.com/library/default.aspx>

[11] Kryptologie [online] [cit. 2012-02-19]. Dostupný z WWW:< http://kryptologie.uhk.cz/idea_cz.htm>

[12] Tichý, Jan. 2004. Programová podpora tvorby webových aplikací. [online] 25. 8 2004. [cit. 2012-04-12]. Dostupný z < http://www.jantichy.cz/diplomka>

[13] How get Email from user profile? | LinkedIn Developer Network [Online] 2013 [cit. 2013-03-05]. Dostupný z WWW:< http://developer.linkedin.com/thread/3265>

[14] OAuth2 – google-api-php-client [online] 2012 [cit. 2013-03-05]. Dostupný z WWW:< https://code.google.com/p/google-api-php-client/wiki/OAuth2>

[15] PHP: DOMDocument::__construct - Manual[online] 2013 [cit. 2013-03-27]. Dostupný z WWW:< http://php.net/manual/en/domdocument.construct.php>

[16] Šifrovací standartd AES [online] 1999 [cit. 2012-02-19]. Dostupný z WWW:<http://crypto-world.info/klima/1999/chip-1999-11-64-65.pdf>

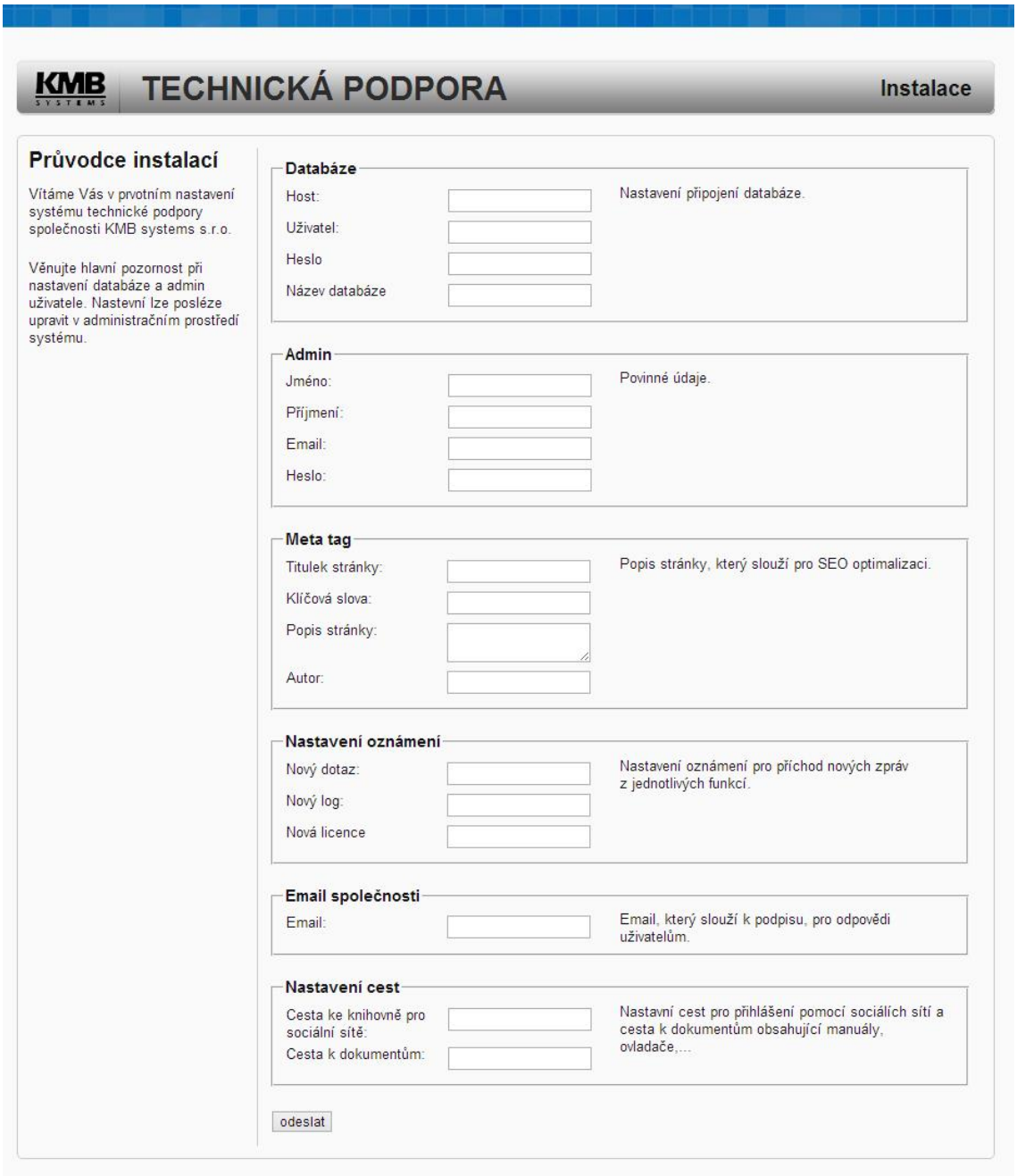
[17] TinyMCE - Home [online] 2003 [cit. 2013-05-06]. Dostupný z WWW:<http://www.tinymce.com >

[18] Twitter Developers [online] 2003 [cit. 2013-05-06]. Dostupný z WWW:<http://dev.twitter.com >

Příloha A – Instalace systému

Průvodce instalací webové aplikace:

1. Přenesení složky „technicalsupport“ pomocí FTP klienta na server.
2. Přejít na adresu http://www.kmbsystems.cz/technicalsupport/.
3. Vyplnění formuláře pro základní nastavení systému.
4. Odstranění instalačního adresáře „install“.
5. Aplikace je nainstalována a připravena k použití.



Obr. A.1 – Grafický návrh mobilní aplikace

Instalace webové aplikace je přizpůsobena pro běžného uživatele. Po přenesení celé aplikace na server a spuštění adresy, na kterou má být aplikace instalována, je načten formulář. Tento formulář představuje základní nastavení systému.

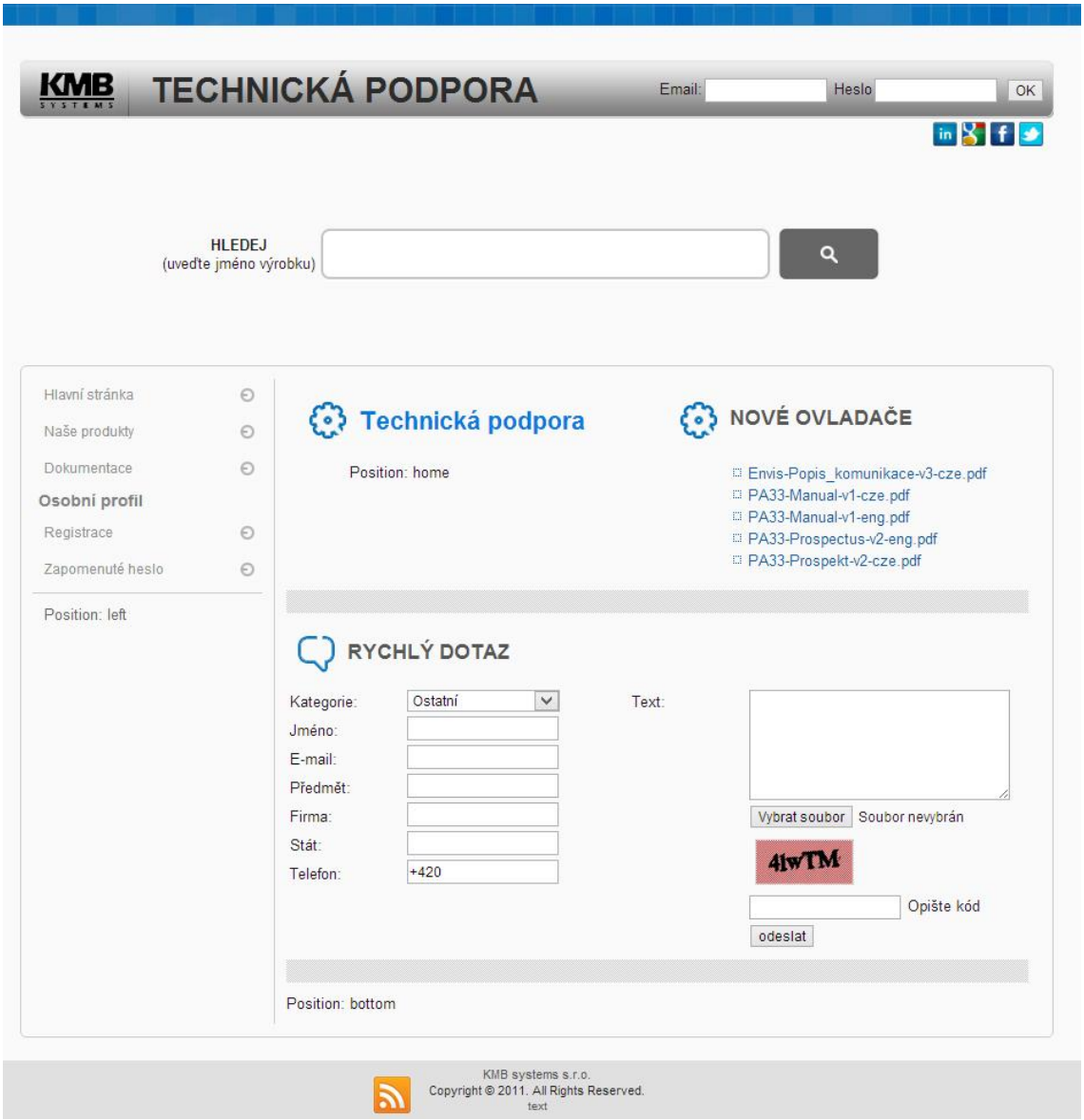
Zpracování dat zajišťuje třída `fn,` kde jsou implementovány funkce `createFile()` a `createDatabase()`. Jak vyplívá z názvu těchto funkcí, úkolem je vytvoření konfiguračního souboru *configuration.php* a příslušných tabulek v databázi, které bude využívat technická podpora.

Konfigurační soubor je uložen v kořenovém adresáři aplikace a obsahuje třídu `TConfig()` s nastavenými parametry získanými z formuláře.

```
Class TConfig {
        //pristup do databaze
        public $database_host = "'.$database_host.'";
        public $database_user = "'.$database_user.'";
        public $database_pass = "'.$database_pass.'";
        public $database_name = "'.$database_name.'";

        //cesta ke knihovne pro social login
        public $base_url = "'.$social_path.'";

        //cesta pro download souboru
        //public $filepath = "'.$file_path.'";

        //meta
        public $sitename = "'.$sitename.'";
        public $keywords = "'.$keywords.'";
        public $description = "'.$description.'";
        public $author = "'.$autor.'";

        //nastaveni oznameni
        public $newDotaz ="'.$notificDotaz.'";
        public $newLog ="'.$notificLog.'";
        public $newLic ="'.$notificLic.'";

        public $email = "'.$email.'";
}
```

V případě, že instalace proběhne v pořádku, zobrazí se informační stránka o úspěšné instalaci. Po té je nutné odstranit instalační adresář a přejít dle navigace na úvodní stránku aplikace.

Příloha B – Základní vizualizace webové aplikace

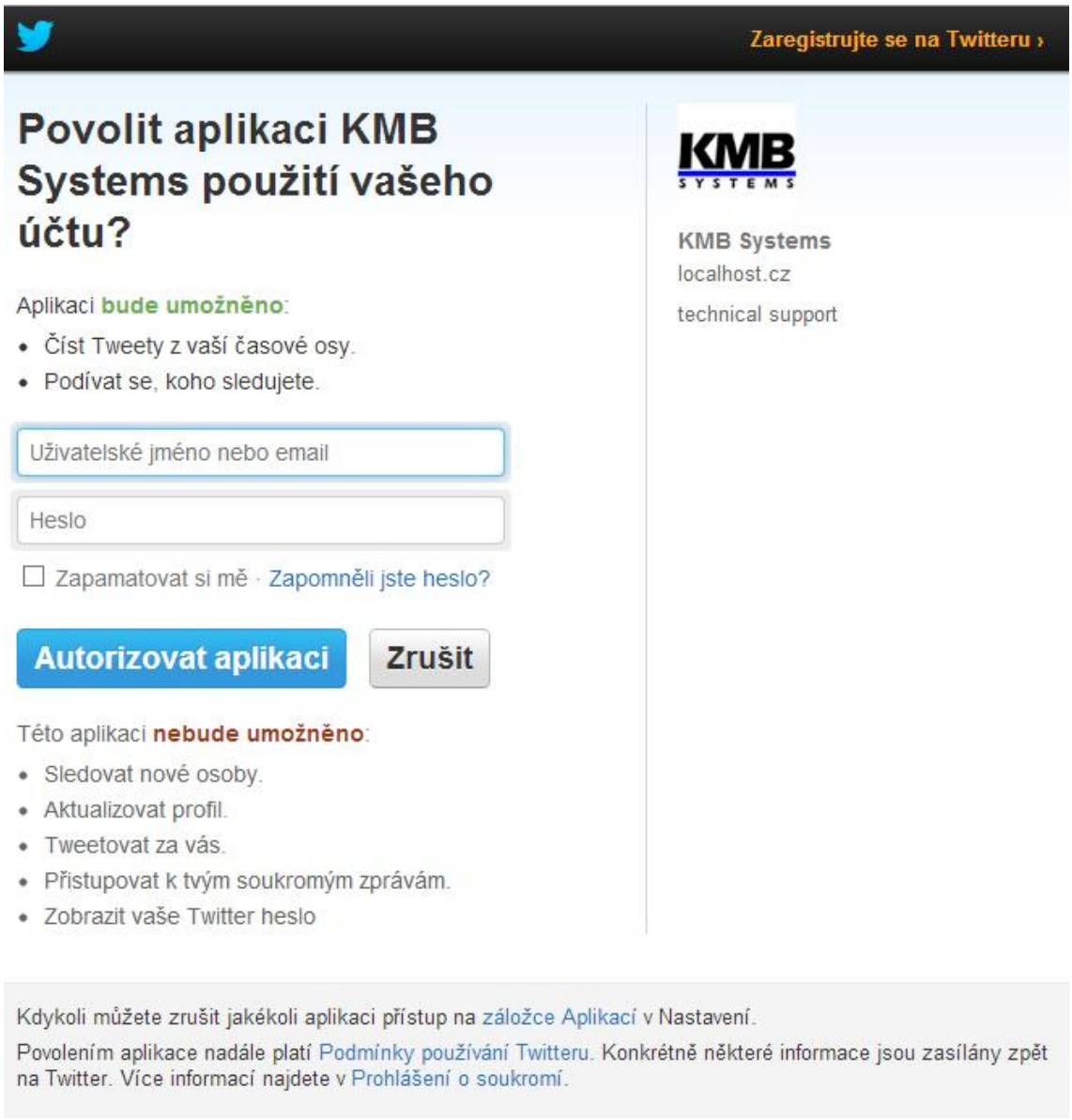


Obr. B.1 – Grafický návrh úvodní stránky webové aplikace

Příloha C – Autorizace údajů Google, LinkedIn, Twitter

Obr. C.1 – Autorizace aplikace Google

Obr. C.2 – Autorizace aplikace Linkedin



Obr. C.3 – Autorizace aplikace Twitter