Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Ján Lučanský

Webový systém správy obsahu

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Michal Kopecký, Ph.D.

Studijní program: Informatika, Správa počítačových systémů

2006

Ďakujem vedúcemu mojej bakalárskej práce
RNDr. Michalovi Kopeckému, Ph.D.
za jeho pomoc, rady a zhovievavosť pri tvorbe tejto práce.

Prehlasujem, že som svoju prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím s požičaním práce.

V Prahe dňa 7. 8. 2006 Ján Lučanský

Obsah

Název práce: Webový systém správy obsahu
Autor: Ján Lučanský
Katedra: Katedra softwarového inženýrství
Vedoucí bakalářské práce: RNDr. Michal Kopecký Ph.D.
e-mail vedoucího: michal.kopecky@mff.cuni.cz

Abstrakt: V predloženej práci je popísaný návrh a implementácia webového systému správy obsahu (CMS podľa anglického Content Management System). Systém je zameraný na generovanie formulárov a ich overovanie. Ďalej sa zaoberá spracovaním dát z týchto formulárov, konkrétne ich ukladaním do databázy, ich úpravou a mazaním. Systém obsahuje ľahko modifikovateľné prostriedky na zobrazovanie týchto dát. Dokáže dáta exportovať do formátov PDF, XML a CVS. Takisto umožňuje generovanie koláčových grafov. Podstatnú časť tvoria nástroje na publikovanie týchto dát na verejné internetové stránky.

Kľúčové slová: systém správy obsahu, CMS, internetové stránky

Title: Web content management system
Author: Ján Lučanský
Department: Department of Software Engineering
Supervisor: RNDr. Michal Kopecký Ph.D.
Supervisor's e-mail address: michal.kopecky@mff.cuni.cz

Abstract: This work describes web content management systems (CMS). It discusses what CMS systems are and what are general characteristics and requirements of these systems. Work further analyzes existing CMS solutions. The main part focuses on CMS system which is a product of author of this work. The system aims primarily on generating forms and form validation. It also specializes on processing data from these forms. In particular it focuses on their creating, editing and erasing. System contains also easily administrable tools for presenting these data. System is also capable of exporting these data to PDF, XML and CSV formats. In addition, system enables user to generate pie charts. A considerable part of the solution comprises tools for presenting data on public web pages.

Keywords: content management system, CMS, web sites

Úvod

Systém Správy Obsahu - definícia

Systém správy obsahu (ďalej len CMS podľa anglického Content Management System) sa dá definovať ako technológia, ktorá zahrňuje správu, údržbu a distribúciu dát. CMS systémy sú prevažne nasadzované ako vnútropodnikové aplikácie, ktoré zvyšujú produktivitu a prehľad o dokumentoch, ktoré sa v rámci podniku vyskytujú. Zároveň by mal zvyšovať efektivitu práce s týmito dokumentmi. Okrem iného sa do práce s týmito dokumentmi nasadzujú rôzne podnikové procesy. Ako celok to má následok zvyšovania efektivity práce.

V poslednom období je stále dôležitejšia otázka distribúcie dokumentov po internete. Preto moderné CMS obsahujú nástroje, ktoré dokážu generovať internetové prezentácie či emaily, ktoré tieto dokumenty distribuujú. Neraz sa preto CMS stotožňuje s názvom redakčný systém.

Motivácia

Pre tvorbu CMS som sa rozhodol hneď z niekoľkých hľadísk.

Praktické hľadisko.
Veľkú časť voľného času trávim prácou, ktorá spočíva v tvorbe internetových stránok a menších intranetových aplikácii. Po niekoľkých projektoch som zistil, že požiadavky klientov majú veľa spoločného. Preto som sa rozhodol vytvoriť CMS, ktoré by mi pomohlo tvorbu stránok zjednodušiť a prácne časti zautomatizovať.

Technické hľadisko.
Súvisí s tým, že mám s tvorbou podobných aplikácií určité skúsenosti, ktoré by som rád prehĺbil a zároveň rozšíril rozsah vedomosti z danej problematiky.

Osobné hľadisko.
Súvisí s predošlými dvoma. A to v tom, že vždy ma veľmi bavilo tvoriť prezentácie a aplikácie, ktoré súviseli s webom. Tvorbou takého systému chcem zabezpečiť, aby bolo programovanie efektívnejšie a prehľadnejšie. Hlavne ale môžem spojiť školské a profesijné povinnosti.

Popis kapitol

Prvá časť kapitoly 1 rozoberá existujúce CMS systémy MAMBO (Joomla!), Magic WebDrive CMS, IBM WebSphere Portal a phpMyAdmin a vzájomné porovnanie ich vlastností. Druhá časť sa venuje požiadavkám na funkčnosť nového CMS systému Makaga. Poslednou časťou tejto kapitoly sú analýzy na výber najvhodnejšieho programovacieho jazyka a databáze. Kapitola 2 zoznamuje čitateľa s používaním nakonfigurovaného systému Makaga. Tu by mal čitateľ získať názornú predstavu, ako systém funguje a čo dokáže. Nasledujúca 3. kapitola obsahuje podrobný popis toho, ako sa systém inštaluje a ako sa pracuje s jeho konfiguračnými súbormi a šablónami. Kapitola číslo 0 popisuje jednotlivé funkčné celky programu. Ďalej obsahuje informácie o tom, ako možno systém vylepšovať. Posledná, 5. kapitola zhodnocuje dosiahnuté výsledky systému Makaga a polemizuje nad možnými rozšíreniami tohto programu.

1. Analýza

1.1. Technológie pre vývoj webových aplikácií

1.1.1. Programovací jazyk

PHP

Jazyk PHP [1] je veľmi rozšírený a obľúbený. Podporuje ho takmer každý webhosting. Ako komerčný, tak voľný. Jazyk PHP obsahuje rozsiahle funkcie pre prácu s poľom, textom, grafikou, súbormi, databázami a podporuje množstvo internetových protokolov (FTP, HTTP, IMAP, LDAP, POP3, SMTP, SNMP, ...) Ďalej je otázkou, ktorú verziu programovacieho jazyka zvoliť. Aj keď najnovšou verziou je verzia PHP 5, veľmi často je používaná verzia 4. Verzia 5 nie je spätne kompatibilná. Obsahuje ale väčšie programovacie možnosti (lepší objektový model, výnimky, ...) Interpreti jazyka PHP fungujú ako na serveroch Linuxu, tak na serveroch operačného systému Windows.

Ruby on Rails

Ruby [2] je skriptovací programovací jazyk. Aj keď je na poli skriptovacích jazykov veľká konkurencia, Ruby si získava stále viac na obľube. Písanie kódu v Ruby je veľmi intuitívne. Takisto obsahuje množstvo knižníc, ktoré sa každým dňom zväčšuje. Rails [3] je framework napísaný v Ruby a ten slúži práve na písanie webových aplikácií. Funguje ako MVC (Model-View-Controller). View má na starosti len zobrazovanie – obdoba šablónovacieho systému. Controller prijíma požiadavky od klienta a obsahuje základnú rozhodovaciu logiku. Model obsahuje jadro aplikácie, ako je komunikácia s databázou, obchodné pravidlá a podobne.

Bohužiaľ nevýhodou je, že zatiaľ neexistuje veľa literatúry, z ktorej by si mohol človek vybrať. Ďalšou nevýhodou je, že podpora hostingov je v súčasnosti takmer nulová.

ASP

Dá sa povedať, že ASP [4] je akousi konkurenciou pre PHP. Je to technológia od firmy Microsoft. Pre programovanie v ASP sa používali jazyky VBscript alebo JScript. ASP je súčasťou webového servera ISS (Internet Information Server) ASP je však možno prevádzkovať aj na serveri Apache.

ASP.NET

ASP.NET [5] opäť technológiou firmy Microsoft. Aj keď je odvodená od pôvodného ASP, je dosť odlišná. ASP.NET je založený nad CLR (Common Language Runtime). Ten je zdieľaný všetkými aplikáciami postavenými nad technológiou .NET. Preto je možnosť zvoliť z viacerých programovacích jazykov ako je C# či Visual Basic .NET. Princíp fungovania sa správa a podobne ako vývoj "okienkových" aplikácii. To znamená že jednotlivé časti stránok sa správajú ako objekty, na ktorých sa volajú a spracúvajú udalosti.

J2EE

J2EE [6] je technológia firmy SUN Microsystems. Obsahuje JSP (Java Server Page) a Java Servlets. JSP je založený na vkládaní príkazov do HTML kódu cez špeciálne značky. Potom tieto kódy server preloži do Java Servlets. Samotné JSP sa používa skôr vo väčších projektoch. Veľakrát je praxou, že Java skôr využíva na výstupy samotné PHP.

Pre mňa ako autora bola najprívetivejší jazyk **PHP** verzie 4. Predovšetkým z dôvodu jeho rozšíriteľnosti a hostingovej podpory. Ďalším dôvodom sú aj moje skúsenosti s týmto jazykom. Zvolil som verziu 4 kvôli jej väčšej podpore hostingov vzhľadom k verzii 5. Určujúce rozdiely ale medzi verziou 4 a 5 neexistujú.

1.1.2. HTML, CSS a Javascript

GUI v HTML, CSS a Javascripte je stále obľúbenejšie aj v rozsiahlejších systémoch. Výhodou je, že nie je potrebná žiadna inštalácia klienta. Ďalej existuje množstvo štandardov, ktoré popisujú prístupnosť a ergonómiu stránok. Nevýhodou je, že sa ťažšie spracúvajú udalosti od užívateľa (i keď technológia AJAX [7] a frameworky nad touto technológiou zastierajú aj tento problém). Ďalšou nevýhodou je to, že optimalizácia pre viaceré prehliadače je náročnejšia.

1.1.3. Databáza

Keďže programovacím jazykom bol zvolený jazyk PHP, pripadala v úvahu najmä trojica databázových systémov. MySQL [8], PostgreSQL [9] a Firebird [10]. Vzhľadom k požiadavkám na funkčnosť aplikácie, všetky databázy vyhovujú. Zvolil som ale databázu MySQL, lebo tá ma na webhostingoch najväčšiu podporu. Okrem iného väčšina literatúry rozoberá práve spojenie jazyka PHP s databázou MySQL.

Systém by mal vedieť pracovať s databázu MySQL verzie 3. Aj keď je táto verzia zastarala, ešte stále sa s ňou možno stretnúť v praxi. Ideálne však je, ak systém beží aspoň pod verziou 4.0. Tá už obsahuje cache na dotazy. Tým by mala byť práca s CMS rýchlejšia.

1.1.4. Prehliadač

Dôležitú úlohu zohráva aj prehliadač. Aj keď sa stránky front-endu majú optimalizovať pre čo najväčší počet prehliadačov stránky back-endu by mali vyžívať výhody jedného z nich. Možno voliť z prehliadačov ako je Internet Explorer, Mozilla Firefox, Opera, Konqueror alebo Apple Safari. Aj keď má Internet Explorer medzi užívateľmi najväčšiu obľubu, rozhodol som sa pre optimalizáciu pre Firefox. Dôvodom je jeho pomerne veľké rozšírenie a to, že ho možno nainštalovať a prevádzkovať pod operačnými systémami Microsoft Windows, Linux a Mac OS X.

1.1.5. Vývojové prostredie

Čo sa týka vývojového prostredia, možností je veľa. V podstate stačí akýkoľvek textový editor. Medzi profesionálne vývojové prostredia patrí Zend Studio. Tento produkt však nie je zadarmo. Medzi voľné prostredia, ktoré sú kvalitné patrí PhpEditorIDE či PSPad. Najrozumnejšie riešenie je však vývojové prostredie Eclipse[11] s pluginom PHP Eclipse-Plugin [12]. Eclipse je zdarma dostupné vývojové prostredie určené pre Javu. Stiahnuť sa dá množstvo pluginov, ktoré môžu samotné IDE vylepšiť. Príkladom môže byť priama podpora SVN (SubVersioN).

1.2. Už existujúce CMS

Ako bolo spomenuté, existuje veľa CMS. Z rady s GNU/GPL bude rozobratý systém *MAMBO(Joomla!)* [13] , z rady českých komerčných bude zástupcu predstavovať *MagicWebDrive CMS* [14] a z komerčných svetových platforiem *IBM Websphere Portal* [15]. Nakoniec bude spomenutý produkt *PHPMyAdmin* [16] , na ktorý sa budem neskôr odkazovať.

1.2.1. MAMBO (Joomla!)

Veľmi kvalitný open source projekt. Svedčí o tom aj to, že roku 2005 získal cenu *Best Open Source Solutions* na *LinuxWorld* v *San Franciscu*. Dnes je situácia taká, že vývojári systému *MAMBO* začali tvoriť tento produkt pod názvom *Joomla!*

Systém je zameraný hlavne na tvorbu dynamických webových stránok. Má veľmi príjemne užívateľské rozhranie. Základom celého systému je *obsah (content)*. Administrátor má možnosť vytvoriť si základné sekcie, v nich základné kategórie a k ním vzťahovať nejaký *obsah,* ktorý sa bude vo front-ende zobrazovať. Príklad: Sekcie *Obchod*, *Zamestnanci*. V rámci *Obchodu* existujú napríklad takéto kategórie: *O firme, Produkty, Servis, Kontakty, Novinky*,... Samotný obsah sa implicitne vytvára pomocou WYSIWIG javascriptovského editora (*TinyMCE*). Užívateľovi je poskytnuté množstvo nastavení obsahu. Napríklad radenie, dátum kedy sa má článok publikovať, sprievodný obrázok, meta tagy,... Ďalšou dôležitou zložkou je následné prideľovanie obsahu ku menu. Existuje rozhranie na vytvorenie štruktúry menu, kde sa jedným kliknutím priradí potrebný obsah. Ďalšou zaujímavou vymoženosťou systému sú tzv. *komponenty*. Príkladom komponenty *sú novinky, anketa, syndicate, správa bannerov, rozosielanie mailov*. Sú to tie časti front-endu, ktoré majú v back-ende špeciálnu administráciu. Napríklad pri *banneroch* sa dá nastaviť, ako dlho sa bude *banner* zobrazovať, akému *kontaktu* patrí, po koľkých kliknutiach sa má vypnúť a podobne. V *syndicate* je možnosť zobrazovať odkazy na rôzne verzie RSS výstupov.

Systém má možnosť voliť obsah pre užívateľov, ktorí sú prihlásení a ktorí nie. Má možnosť registrovať užívateľov. Funguje to na jednoduchom princípe, ale je to veľmi efektívne.

V systéme je oddeľovaná programová časť od zobrazovacej. Beckend je zobrazovaný pomocou tzv. *templates.* Množstvo *templates* je veľmi veľké. Ich kvalita je takisto výrazná. To robí *MAMBO* tak populárne.

Okrem iného je treba spomenúť, že do systému MAMBO existuje množstvo addonov. Sú to komponenty ako kniha návštev, internetový obchod, či kalendár.

Možno uznanlivo konštatovať, že MAMBO je veľmi kvalitný CMS, zameraný predovšetkým na budovanie stránok. Klasickému užívateľovi a väčšine firmám pokryje skoro všetky ich potreby s veľkým luxusom.

Je napísané v PHP a beží nad databázou MySQL. Tento systém je v určitých smeroch vynikajúcou inšpiráciou.

Veľkými konkurentmi MAMBO sú PostNuke alebo Xoops. Okrem týchto "veľkých" CMS existuje ešte rada systémov, ktoré sa sústreďujú len na určitú sféru webu. Napríklad blogy, fóra, internetové obchody a podobne.

1.2.2. magicWebDrive CMS

Spomedzi českých komerčných produktov je tento zaujímavý hneď z niekoľkých dôvodov. Po prvé, že mal tento produkt naozaj elegantné internetové stránky s množstvom informácií. Keďže získať prístup ku komerčným riešeniam je neľahká úloha, bolo toto hľadisko určujúce. Ďalším dôvodom boli ceny, ktoré sa k danému produktu vzťahovali. Je teda zaujímavé, čo môže zákazník zakúpiť.

Platforma sa predáva v troch edíciách. *Personal*, *Profesional* a *Entreprise*. Mimo to je v ponuke množstvo modulov, ktoré sa dajú dokúpiť. V nasledujúcej časti popíšem základné a rozširujúce systémy, ktoré je možno použiť.

- **Štruktúra stránky a obsah** – modul na základnú správu menu, obsahu k nemu pridelenému a "bonusy" ako formulár na pridávanie príspevkov k článku.
- **Súbory** – modul na manažovanie multimediálnych súborov.
- **Užívatelia** – modul na manažovanie užívateľov.
- **Práva** – modul pre manažovanie práv k jednotlivým *obsahom* a ich nastaveniam.
- **Databáza** – modul, ktorý dokáže vytvárať jednoduchým spôsobom tabuľky a relácie.
- **Rubriky a články –** modul**,** ktorý obsahuje správu článkov, čas ich publikovania a zmien.
- **Galéria obrázkov** – modul na manažovanie galérie obrázkov.
- **Galéria súborov** – modul na manažovanie galérie súborov.
- **Galéria odkazov** – modul na manažovanie galérie odkazov.
- **Kvízy** – modul na tvorbu kvízu.
- **Tlač** – modul obsahujúci zostavy pre tlač.
- **Export** – modul schopný exportu dát do XML alebo CSV.
- **Kalendár akcií** – modul pre kalendár akcií.
- **Kalendár akcií s rezervačným systémom** – modul pre kalendár akcií s rezervačným systémom.
- **Fulltextové vyhľadávanie** – modul, ktorý zaručí fulltextové vyhľadávanie v databáze.
- **Mapa serveru** – modul, ktorý dokáže generovať mapu servera.
- **Ankety** – modul na obsluhu ankiet.
- **Novinky** – modul na pridávanie, editovanie a mazanie noviniek.
- **Emailer** – modul, ktorý dokáže rozosielať hromadné maily.
- **Knihy návštev** – modul pre knihu návštev.
- **Otázky a odpovede** – modul pre manažovanie známeho FAQ.
- **Diskusia** – modul pre diskusie.
- **Štatistiky** – štatistika prístupov na jednotlivé *obsahy.*
- **Viacjazyčné prezentácie** – podpora viacjazyčnosti.
- **magicShop** – rozšírenie na internetový obchod.

Samotný produkt funguje na systéme podobnom MAMBO. Má veľmi príjemne užívateľské rozhranie. Systém sa pýši zaujímavými vlastnosťami, ktoré si zaslúžia pozornosť. Jednou z nich je to, že stránky sa cez šablóny negenerujú pri každom vzhliadnutí, ale vygenerujú sa iba raz pri vytvorení a vo front-ende sa načíta len ako statický text. To nepochybne zvyšuje rýchlosť na front-ende.

Ďalšou vymoženosťou je správa databáze. Iba v tomto produkte sa možno stretnúť so správou databáze na takejto úrovni. Obsahuje jednoduchú tvorbu tabuľky zadávaním názvu stĺpca a s dátovým typom číslo, text, email, html,...

Zaujme aj to, že jednotlivé stĺpce sa dajú zamykať pre jednotlivých užívateľov. To sa dá využiť napríklad pri tabuľke zamestnancov, kde ku stĺpcu plat bude mať prístup povedzme len manager a administrátor, ale bežný administrátor nie.

1.2.3. IBM WebSphere Portal

Na poli profesionálnych CMS zaujíma vážne postavenie systém *IBM WebSphere Portal*. Tento systém úspešne konkuruje CMS systémom s podobným zameraním a síce *BEA WebLogic Platform 8.1, SharePoint Portal Server 2003 od firmy Microsoft, OracleAS Portal 10g, Java system portal 6.2*

Takisto je potrebné v úvode poznamenať, že sa jedná o super výkonné, super kvalitné, ale zároveň o super drahé riešenie. Ich cena sa pohybuje v rádoch 10000 USD na jedno CPU.

IBM portálové riešenie umožňuje organizáciám, podnikom, rovnako ako oddeleniam v rámci veľkých podnikov, rýchlejšie zavádzať portály typu bussiness-to-bussines, bussiness-to-customer a bussiness-to-employee. Produkt je zameraný na minimálne nároky na správu a administráciu. Táto platforma obsahuje množstvo hotových služieb a komponent, pomocou ktorých je možné rýchlo postaviť ľubovoľné riešenie. Toto umožňuje rýchle reagovať na nové obchodné požiadavky, rýchle poskytnúť nové aplikácie a služby s minimalizovaním rizík.

Popisovaná platforma je neporovnateľná s voľnými riešeniami. Produkt ako tento je treba brať ako veľkú inšpiráciu a križovatku ciest toho, kam by sa vlastný projekt mohol uberať.

- Zavedenie jednotného bezpečného modelu prístupu užívateľov ku všetkým typom informáciám a aplikáciám.
- Zaisťuje prispôsobenie obsahu portálu pre konkrétneho užívateľa alebo skupinu, pomocou obchodných pravidiel a profilov.
- Umožňuje koncovým užívateľom ľahko upravovať prostredie a obsah svojich portálov. Užívateľ si sám volí, ktoré informácie sú pre neho zaujímavé a dôležité a tieto umiestni podľa potreby v prostredí portálu.
- Uľahčuje užívateľom rýchle vyhľadať potrebné informácie na portáloch.
- Obsah portálov je dostupný z rôznych typov zariadení, napríklad mobilný telefón alebo PDA.
- Ponúka kompletnú sadu služieb pre teamovú spoluprácu a komunikáciu – prijíma komunikáciu, webové konferencie, prístupy k mailu atď.
- Obsahuje kancelárske komponenty (portlety), ktoré umožňujú koncovým užívateľom pracovať v prostredí portálu podobným spôsobom ako v prostredí textových a tabuľkových editorov vrátane podpory prezentácií. Užívateľ nepotrebuje mať na svojom počítači nainštalovaný žiadny iný software.
- Obsahuje kompletný robustný systém pre správu obsahu v prostredí portálu. Koncovým užívateľom umožňuje ľahko vkladať a publikovať nové informácie do portálu bez znalosti programovania. Správcom poskytuje prostredie na vytváranie šablón pre vkladanie správ, riadenie prístupu užívateľov ku konkrétnym oblastiam informácií. Správca takisto nastavuje procesy pre schvaľovanie a publikovanie informácií.
- IBM ponúka na stránkach katalóg hotových komponent, ktorých sú stovky. Podľa toho možno vytvoriť portál šitý na mieru. Podotýkam, že medzi tieto portlety patria aj systémy pre obchodné burzy, podpora práce s bankami, rôzne vstavané finančnícke aplikácie a podobne.
- Správa vlastného portálu sa prevádza priamo v prostredí webového prehliadača.

1.2.4. phpMyAdmin

phpMyAdmin je veľmi podarený a obľúbený klient pre správu databáze MySQL. Je naprogramovaný v jazyku PHP. Obsahuje rozumné a ergonomické rozhranie na tvorbu databáz a tabuliek. Dokáže vytvárať databázy a tabuľky na pár kliknutí. Dokáže pre vytvorené tabuľky generovať formuláre pre vkladanie dát. Dokáže dáta listovať, radiť, meniť, editovať. Dáta dokáže importovať a exportovať v rôznych formátoch. Na pár kliknutí dokáže zmeniť štruktúru tabuľky. Má pekné rozhranie pre vyhľadávanie dát. Dáta dokáže tlačiť. Skoro každý php webhosting túto aplikáciu obsahuje v základnom balíčku. Napriek tomu, že dokáže spravovať databázu, neposkytuje žiadne možnosti prezentácie dát navonok a pre bežného užívateľa sa nehodí. Pre programátora je ale neoceniteľnou pomôckou.

1.3. Porovnanie

	MAMBO (Joomla!)	**magicWeb Drive CMS**	**IBM WebSphere Portal**	**phpMyAdmin**
Generovanie webových stránok	**Áno**	**Áno**	**Áno**	**Nie**
Správa tabuliek	**Nie**	**Áno**	**Áno**	**Áno**
Podpora zobrazenia tabuľkových dát vo front-ende	**Nie**	**Nie**	**Áno**	**Nie**
Podpora správy súborov	**Nie**	**Áno**	**Áno**	**Nie** (len ako BLOB)
Podpora správy obrázkov	**Áno**	**Áno**	**Áno**	**Nie** (len ako BLOB)
Export do XML a CVS	**Nie**	**Áno**	**Áno**	**Áno**
Export dát do formátov určených pre tlač (PDF, LaTeX ...)	**Nie**	**Nie**	**Áno**	**Áno**
Cena	**Zdarma**	**9900 – 125000 CZK** Podľa zakúpených modulov	**Rádovo Státisíce CZK**	**Zdarma**

1.4. Základné požiadavky

Základné požiadavky na nový CMS, ktorý nesie názov *Makaga*, vychádzajú z existujúcich riešení. Nevýhoda systému MAMBO sa javí hlavne v tom, že nemá podporu správy databázových tabuliek. To znamená, že na obsah stránok nemožno „napojiť“ informácie z už existujúcej tabuľky. Väčšina firiem, ktoré sa rozhodnú pre CMS riešenie už nejakú databázu má. Minimálne nejaké excelovské tabuľky so svojimi produktmi. V praxi sa takisto stáva, že zákazník vyžaduje systém, ktorý spravuje neveľké množstvo tabuliek. Napríklad správa kontaktov, produktov, objednávok,... Preto som sa rozhodol, že *Makaga* nebude svoj dátový obsah určovať na základe entít ako je článok, novinka, FAQ a podobne, ale jeho obsah bude stáť na databáze a tabuľkách. V praxi to bude znamenať to, že systém “nebude vedieť”, že pracuje s novinkou, stránkou produktu, alebo stránkou FAQ. Bude len vedieť, že pracuje s istou entitou, ktorú rozpoznáva len na základe určitých vlastností. Týmto systém dostane abstraktnejšiu formu. Výhodou bude, že užívateľovi programu sa bude môcť ľahko nakonfigurovať samostatná databáza entít, s ktorými v praxi pracuje a vo front-ende zobrazovať veci, ktoré uzná za vhodné. Ďalšia výhoda, ktorá plynie z predošlého určenia, je tá, že takto môže designér presne a ľahko navrhnúť šablóny na mieru. Nevýhodou naopak bude, že Makagaa CMS nebude disponovať tak jednoduchou administračnou obsluhou ako MAMBO alebo magicWebDrive. Na rozdiel od ostatných CMS ale nechcem vytvoriť systém, kde užívateľ nebude potrebovať žiadne znalosti programovania. Naopak, predpokladá sa, že užívateľ tieto znalostí má a Makaga by mu mal pomôcť automatizovať tie častí vývoja, ktoré sú časté a zdĺhavé. Podľa mojich skúsenosti ide väčšinou o rozhranie pre tvorbu, editovanie a mazanie katalógových dát (produktov) a o tvorbu výstupov a ich spojenie s grafikou. Sila systému bude teda zameraná na správu dát, tvorbu menu a systém prepojenia webových stránok so spravovanými dátami.

1.5. Kompletné požiadavky

Nasledujúce popisujú jednotlivé nosné prvky systému Makaga.

1.5.1. Požiadavky na back-end

Back-endom je nazývané administračné rozhranie ku spravovaným katalógovým položkám. Požiadavky na toto rozhranie sú nasledovné.

Správa katalógových položiek

Správa umožňuje prezerať, vytvárať a upravovať položky. Toto rozhranie musí obsahovať jednoduché a ergonomické navigačné menu, ktoré sprístupňuje kategórie položiek. Ďalej musí obsahovať rozhranie na prezeranie týchto dát. S tým súvisí možnosť radenia, listovania a vyhľadávanie záznamov. Samotné vyhľadávanie by malo obsahovať klasické jednoduché vyhľadávanie, ale aj zložitejší filter podľa zvolených atribútov záznamu.

Správa súborov a obrázkov

Systém by mal obsahovať možnosť uploadovania súborov rôznych typov. Ďalej by mal vedieť spracovávať obrázkové formáty. Bolo by rozumné, ak by vedel robiť na obrázky náhľady.

Tvorba front-endu

Front-end predstavuje webovú prezentáciu, ktorá je spravovaná cez back-end. Základom by malo byť jednoduché rozhranie, ktoré umožní vytvárať štruktúrované položky menu. Hĺbka menu nesmie byť obmedzená. K jednotlivým položkám menu musí byť možnosť priradiť obsah.

Export

Systém by mal vedieť exportovať vyfiltrované dáta. Prvým formátom je **XML.** Je to rozšírený formát, ktorý tvorí štandard pre výmenu informácii. Neraz sa stane, že užívateľ potrebuje poslať svoje dáta v nejakom počítačom ľahko čitateľnom formáte svojim partnerom. Príkladom môže byť pravidelné posielanie adries odberateľov časopisu distribučnej firme, ktorá má na starosti rozosielanie výtlačkov. Ďalším formátom je **CSV**. Jedná sa o jednoduchý formát, s ktorým pracujú tabuľkové procesory ako je napríklad Microsoft Excel. V praxi sú tieto procesory veľmi rozšírené a preto je export do tohto formátu nevyhnutný. Posledným formátom je **PDF**. Tento formát slúži na ukladanie dát do grafickej podoby tak, aby tieto dáta boli na každej platforme zobrazované rovnako. Väčšinou slúži ako formát pre tlač. Preto firmy spravujú svoje cenníky práve v tomto formáte. Je teda rozumné, aby systém dokázal takéto cenníky aj sám generovať.

Grafy a štatistiky

Tie by mali poskytovať prehľad o dátach vo vnútri systému. Graf ma väčšiu výpovednú hodnotu ako tabuľka s číslami.

Bezpečnosť.

Systém musí obsahovať aspoň základnú bezpečnostnú politiku. Práva by sa mali vzťahovať na užívateľa, katalógovú kategóriu a akciu. Príkladom môže byť, že sekretárka bude mať právo prezerať kontakty, vytvárať nové, ale nebude ich môcť mazať a modifikovať. Rozlišovať sa budú 4 druhy akcií. Listovať položky, vytvárať nové položky, editovať položky a mazať položky.

1.5.2. Požiadavky na front-end

Zobrazenie menu

Systém vo front-ende musí vedieť zobraziť navigačné menu. Takisto by mal obsahovať možnosť meniť jeho vzhľad a rozloženie

Zobrazenie obsahu

V závislosti na menu sa musí zobraziť potrebný obsah.

1.5.3. Požiadavky na administráciu a konfiguráciu

Konfiguráciu by mal obsluhovať človek, ktorý vie programovať v jazyku PHP. Na jednej strane je nevýhoda časovo náročnejšieho nastavovania, na druhej strane systém takto ponúka väčšiu mieru konfigurácie. Takisto je potom lepšia možnosť dorábať špeciálne funkcionality, ktoré by užívateľ nakonfigurovaného systému mohol potrebovať.

Konfigurácia formulárových elementov.

Makaga by mala pracovať s konfiguračnými súbormi, ktoré obsahujú informácie o stavbe katalógovej položky. To znamená:

- Informáciu o HTML type elementu (checkbox, radio button, select, textarea,...).
- Pomocné informácie o tom, aké ďalšie vlastnosti zvolený element nesie (popis, veľkosť, výčet možností pri radio buttone,...).
- Informácia o tom, či je vyplnenie elementu povinné a prípadne podľa akého regulárneho výrazu sa má platnosť overovať.
- Systém by mal mať možnosť formulárovo spracovať aj závislosti medzi tabuľkami Príklad: Máme katalógovú položku "Projektanti", ktorá obsahuje zoznam ľudí. Ďalej máme katalógovú položku "Architektonický projekt", ktorá obsahuje informácie ako názov projektu, počet podlaží, cena projektu, samotný projekt v PDF a podobne. Mimo iné sa tam má nachádzať zoznam projektantov, ktorý sa na projekte podieľali.

Systém musí podporovať zadávanie zoznamu dát z inej tabuľky. Mal by však podporovať aj ďalšiu vlastnosť a to je pridávanie informácii ku vybraným položkám. Napríklad každému zvolenému projektantovi priradiť výšku odmeny. Ide o vzťahy *1:n* a *m:n*.

Konfigurácia obecných pohľadov

Pohľad by mal obsahovať informácie o tom, ako sa budú katalógové položky zobrazovať. To znamená:

- výčet stĺpcov, ktoré sa budú v danom pohľade zobrazovať
- informáciu o tom, koľko záznamov sa má zobraziť na jednu stranu
- prepínače, ktoré hovoria o použití alebo nepoužití, radenia, listovania, filtrovania, vyhľadávania záznamov
- pomocné informácie, ktoré sa týkajú exportov. Napríklad šírka stĺpcov v PDF súbore, logo, popisný text, ...
- umiestnenia zobrazovacích šablón pre hlavičku, telo, pätičku stránky, ...

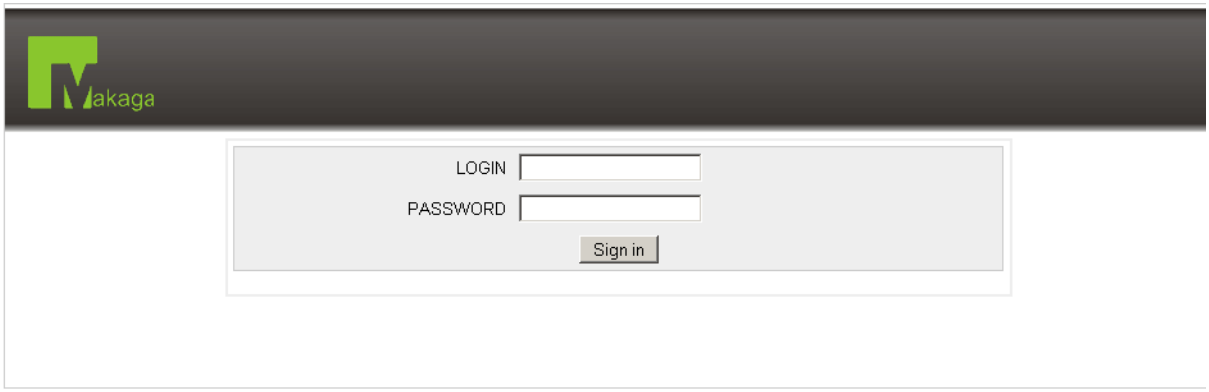
Konfigurácia šablón

Je dôležité, aby bola oddelená logika od grafiky. Preto systém musí obsahovať šablóny, ktoré umožňujú ľahkú modifikáciu grafiky, bez zbytočných zásahov do kódu.

2. Príručka užívateľa

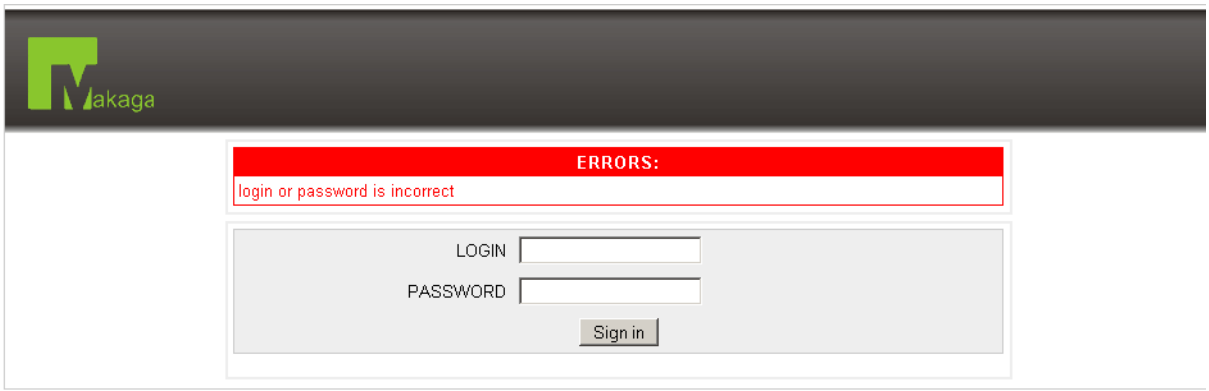
2.1. Prihlásenie

V prvom rade je potrebné, aby užívateľ do adresy prehliadača napísal `http://domena/admin/`, kde `domena` predstavuje URL, kde je systém Makaga nainštalovaný. Prehliadač ukáže nasledujúcu stránku:



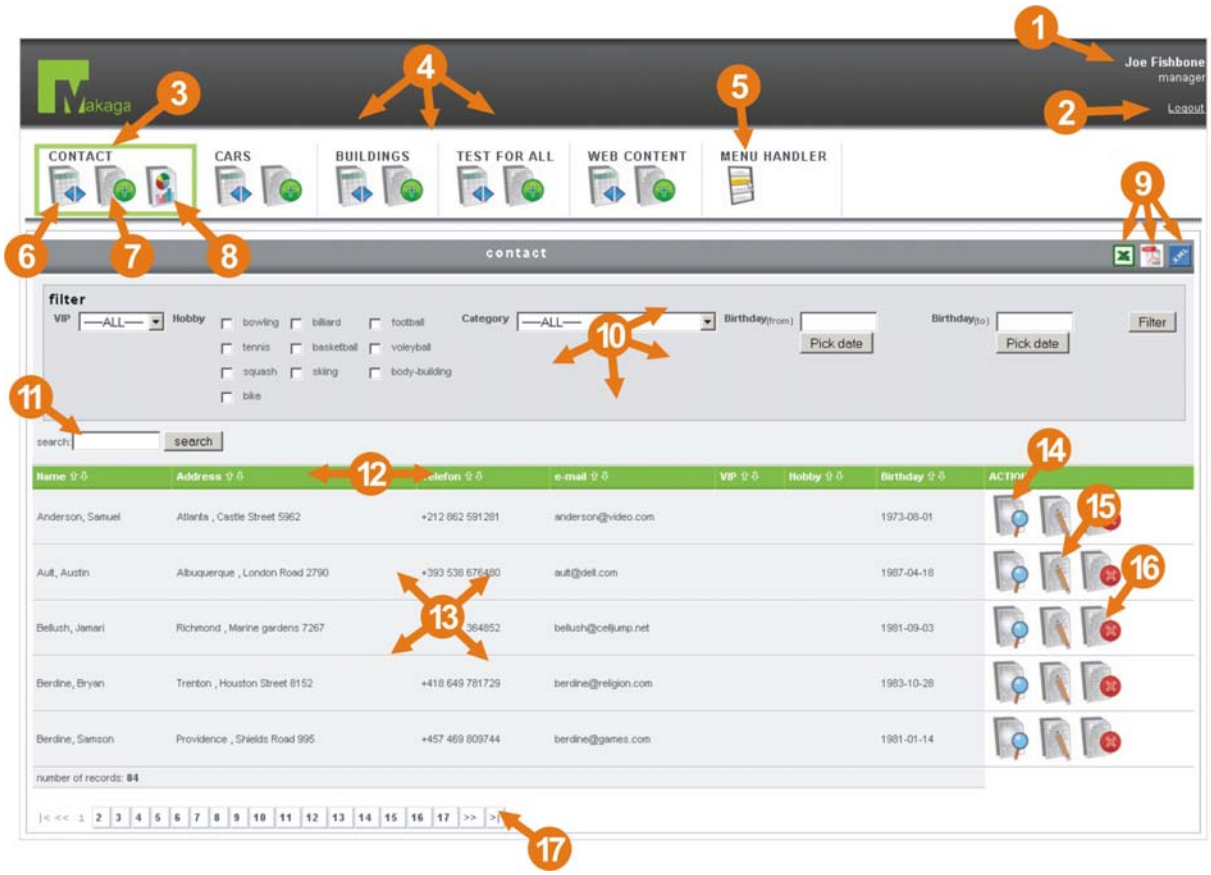
Obr. 1 Prihlasovacie okno aplikácie

Tu je užívateľ vyzvaný, aby napísal svoje užívateľské meno a heslo. Po úspešnom zadaní dôjde ku prihláseniu do systému. V prípade nesprávneho mena a hesla, prípadne nevyplnenia prihlasovacích údajov bude užívateľ upozornený nasledujúcim spôsobom:



Obr. 2 Zobrazenie chybových hlásení pri nesprávnom prihlásení

2.2. Pracovná plocha



Obr. 3 Pracovná plocha

Takto môže vyzerať pracovná plocha po prihlásení užívateľa. Na obrázku sa nachádzajú informačné šípky s číslami, ktoré znamenajú:

1. Užívateľské meno vyznačené tučne a pod ním skupina, kam užívateľ patrí.
2. Hypertextový odkaz. Po stlačení sa užívateľ zo systému odhlási a zobrazí sa obrazovka ako bolo ukázané v kapitole 2.1.
3. Aktuálne zvolená tabuľka je označená zeleným rámčekom. V rámčeku sa nachádza názov tabuľky a ikony znázorňujúce akcie, ktoré možno s tabuľkou vykonať. Ak užívateľ nemá práva na niektoré akcie, prípadne nie sú nad danou tabuľkou definované, žiadne ikony sa zobrazovať nebudú.
4. Ostatné tabuľky. Navzájom sú oddelené tenkou šedou čiarou.
5. Menu Handler, alebo v preklade "Menu Tvorca". Nástroj na tvorbu menu stránky, ktorá je prístupná verejne na adrese `http://domena/` . Ide o URL adresu, kde je Makaga nainštalovaná.
6. Ikona hárku s dvoma modrými šípkami. Po jej kliknutí dôjde ku prezeraní záznamov. Obrázok ukazuje túto akciu.
7. Ikona hárku s symbolom "+" v zelenom krúžku. Po kliknutí naň sa zobrazí formulár na pridávanie záznamov do tabuľky *CONTACT*.
8. Ikona hárku s naznačenými grafmi. Po kliknutí naň sa zobrazia koláčové grafy interpretujúce dáta v tabuľke *CONTACT*.
9. Ak sa v pruhu s názvom tabuľky nachádzajú ikony ako je naznačené, po kliknutí naň dôjde ku exportovaniu dát. Pôjde o export tých dát, ktoré sú momentálne odfiltrované. Prvá ikona predstavuje export do formátu CVS, ktorý možno použiť v tabuľkových procesoroch (Microsoft Excel, OpenOffice.org Calc,...). Druhou

ikonou je export dát do formátu PDF. Po kliknutí na poslednú tretiu ikonu dôjde ku exportu do súboru XML.

10. V tejto tmavo šedej časti sa môže nachádzať filter. Ten môže obsahovať:
    a. Klasické input text boxy. Ak užívateľ zadá nejaký text do tohto boxu, výstupom budú tie záznamy, ktoré obsahujú v danom stĺpci zadaný podreťazec.
    b. Select box alebo radio button. Na výber je vždy možnosť *—ALL—*, čo značí všetky hodnoty. Ďalej obsahuje iné možnosti podľa toho, ako je systém nakonfigurovaný.
    c. V prípade filtrovania dátumu sa zobrazia dva text boxy s *dátumom od* a *dátumom do.* Takto možno vyfiltrovať dáta s dátumami v určitom intervale.
    d. Skupinu checkboxov. Ak užívateľ zaškrtne niektoré z nich a nechá dáta filtrovať, zobrazia sa tie položky, ktoré obsahujú aspoň jednu zo zaškrtnutých položiek.
11. Text box, kde užívateľ vyhľadáva fulltextovo. Stačí zadať kľúčové slovo a kliknúť na tlačidlo *search*.
12. Zelený pruh obsahuje názvy stĺpcov, ktoré sa zobrazujú. Za každým názvom sa môže vyskytovať dvojica šípok. Ukazuje smer hore, druhá smer dole. Kliknutím na šípky užívateľ docieli radenie dát podľa daného stĺpca vzostupne, resp. zostupne.
13. Časť, ktorá zobrazuje stĺpce s dátami danej tabuľky.
14. Pri každom zázname sa v stĺpci *Actions* nachádza ikona hárku so zväčšovacím sklom. Kliknutím naň užívateľ prejde na obrazovku, kde sa zobrazia detaily ku záznamu daného riadku.
15. Ikona hárku s ceruzkou sa nachádza pri každom zázname, ak užívateľ má právo modifikovať záznamy. Po kliknutí naň sa zobrazí pred vyplnený formulár s pôvodnými dátami a po odoslaní sa dáta modifikujú.
16. Hárok so znakom “X” v červenej kružnici. Kliknutím naň užívateľ vymaže záznam.
17. Tlačidlá, ktoré ovládajú listovanie záznamov.
    a. Tlačidlo so znakmi “|<”. Kliknutím na toto tlačidlo sa užívateľ dostane na prvú stránku.
    b. Tlačidlo so znakmi “<<”. Kliknutím tu sa užívateľ dostane o jednu stránku vzad.
    c. Tlačidla s číslom znamenajú presun na konkrétnu stránku
    d. Tlačidlo so znakmi “>>”. Kliknutím tu sa užívateľ dostane o jednu stránku vpred.
    e. Tlačidlo so znakmi “>|”. Kliknutím tu sa užívateľ dostane na poslednú stránku.

2.3. Prezeranie dát

Prezeranie dát prebieha veľmi intuitívne. Stačí klikať na ikony podľa toho ako bol ich význam vysvetlený v predošlej kapitole.

Obr. 4 Hypertextové odkazy

Záznamy sa zobrazujú v riadkoch pod sebou. Zvyčajne obsahujú klasický text. Môže však aj odkazovať hypertextový odkaz a to v dvoch prípadoch:

- Ak sa jedná o uploadovaný súbor. Potom sa zobrazuje hypertextový odkaz cez text, konkrétne názov súboru na serveri. Po kliknutí na odkaz (šípka 1 na obrázku) sa zobrazí dialógové okno, ktoré vyzýva na uloženie súboru na disk (šípka 2 na obrázku)
- Ak sa jedná o uploadovaný obrázok. V prezeraní záznamov sa zobrazí náhľadový obrázok (šípka A na obrázku). Po kliknutí naň sa v novom okne prehliadača zväčšený obrázok (šípka B na obrázku) Po kliknutí na zväčšený obrázok sa okno zatvorí.

2.4. Detail záznamu

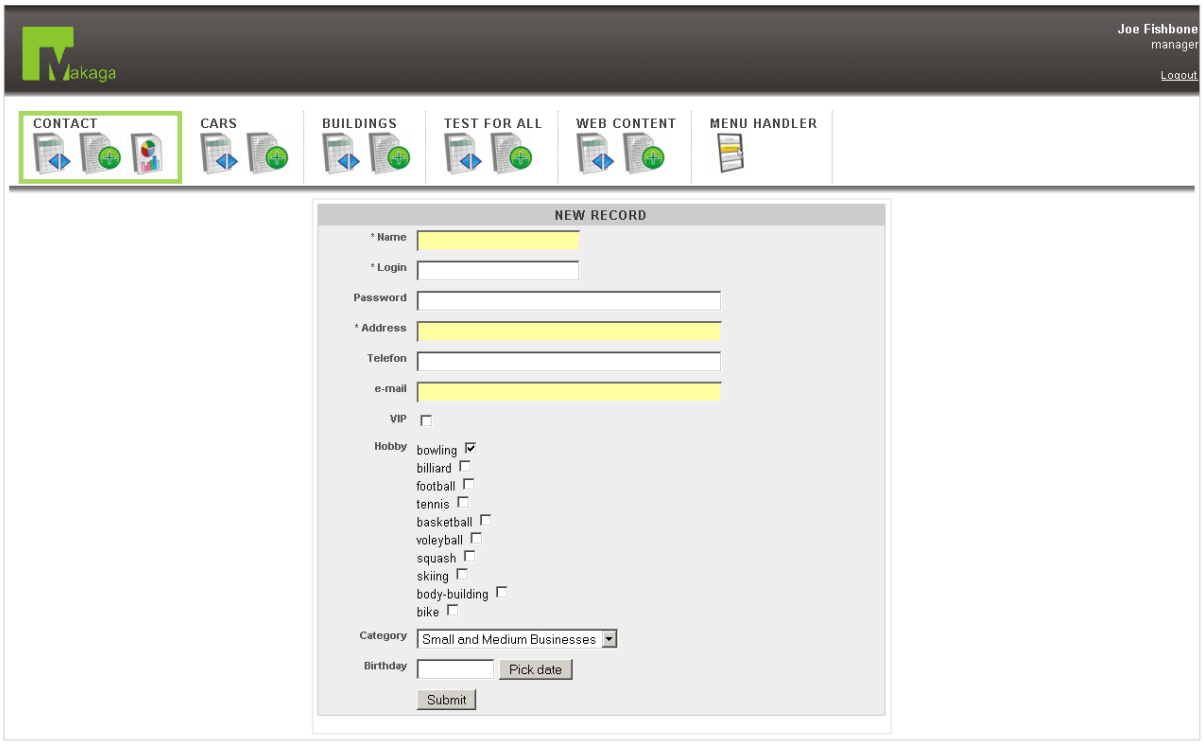
Detail záznamu môže vyzerať nasledovne:

Obr. 5 Náhľad na detail

Obsahuje záznamy pod sebou. Je odporúčane systémy konfigurovať tak, aby sa pri prezeraní dát zobrazovali len niektoré – najvýznamnejšie entity a pri detaile záznamu boli zobrazené entity všetky. Takisto ako v predošlej môžu detaily obsahovať hypertextové odkazy na súbory a obrázky.

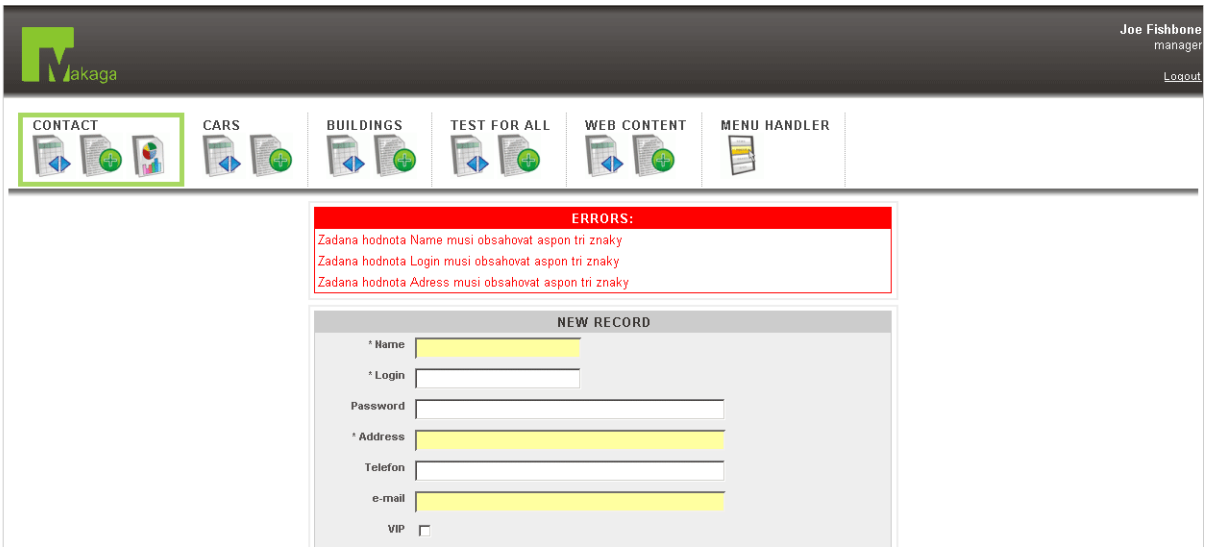
2.5. Nový záznam

Pridať nový záznam znamená vyplniť a odoslať formulár.



Obr. 6 Prázdny formulár

Ako vyplniť formulár vie väčšina bežných užívateľov internetu. Ak sa pred názvom entity nachádza šípka, je vyplnenie tejto hodnoty povinné. Pri nesprávnom vyplnení a odoslaní formulára môže dôjsť ku chybovému hláseniu, ako to demonštruje nasledujúci obrázok:



Obr. 7 Formulár po nekorektnom vyplnení dát

K takejto chybe dochádza po odoslaní formulára v týchto prípadoch:

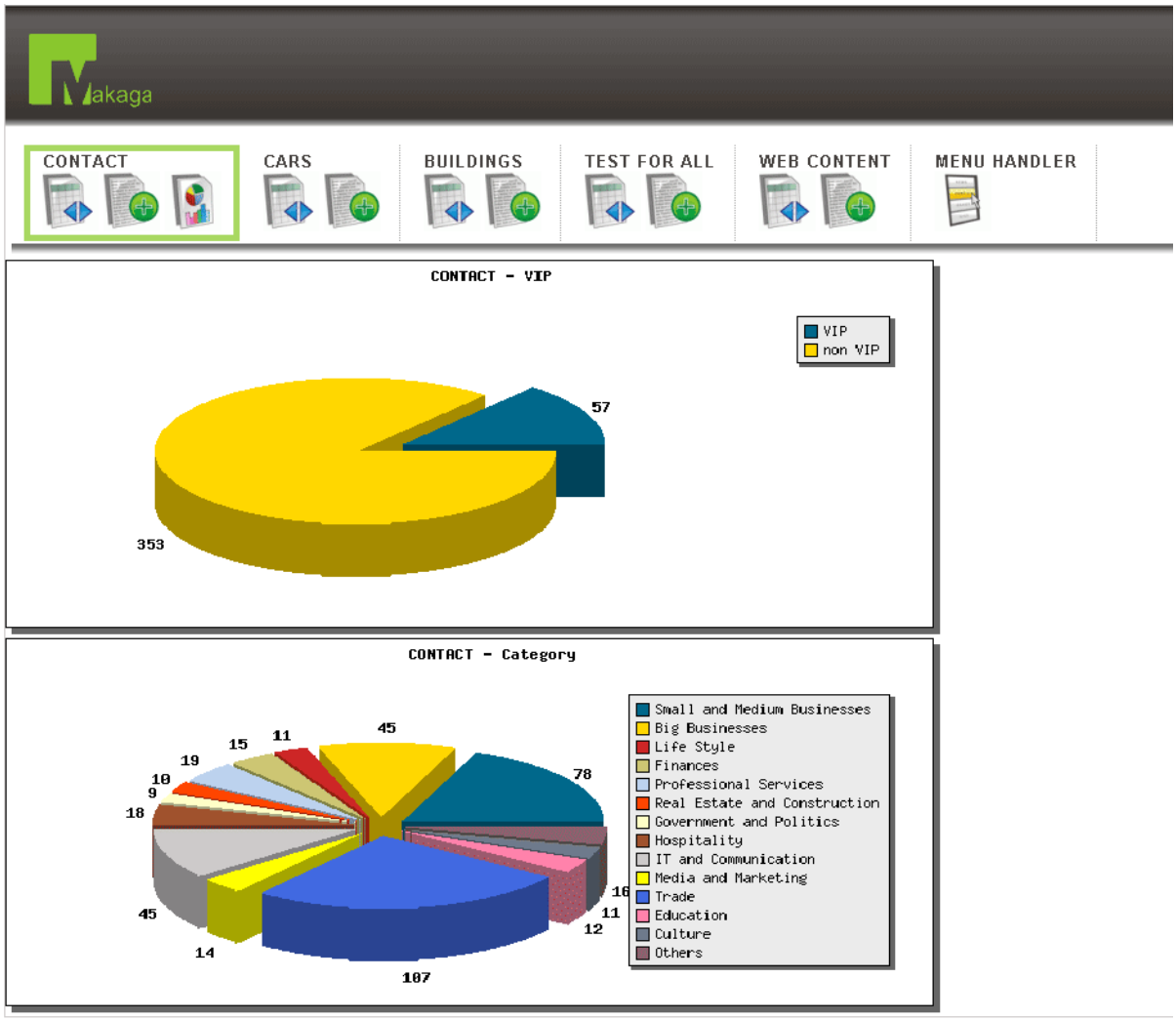
- Užívateľ nezadal povinnú položku.
- Zadaná položka má nesprávny tvar (nesprávny formát dátumu, nečíselná hodnota pri telefónnom čísle, ...)
- Ak uploadovaný súbor je väčší, ako systém dovolí.
- Ak uploadovaný súbor je iného typu, ako systém dovoľuje.

2.6. Editácia záznamov

Editácia dát je veľmi podobná zadávaniu dát a vizuálne sa od nej prakticky nelíši. Po kliknutí na ikonu hárku s ceruzkou sa zobrazí formulár, ktorý už má vyplnené dáta. Modifikácia sa správa odlišne v tom, že ak formulár obsahuje element password box, alebo element na upload súborov, tak vyplnenie týchto elementov je nepovinné. Ak užívateľ zanechá tieto polia prázdne, tieto položky ostanú po modifikácii nezmenené.

2.7. Grafy

Po kliknutím na ikonu s hárkom s grafmi sa užívateľovi zobrazí niekoľko koláčových grafov, ktoré hovoria o rozložení dát. Grafy sa môžu generovať len na tie stĺpce dát, ktoré boli zadávané cez select boxy, radio buttony a checkboxy. Takéto grafy môžu vyzerať nasledovne:

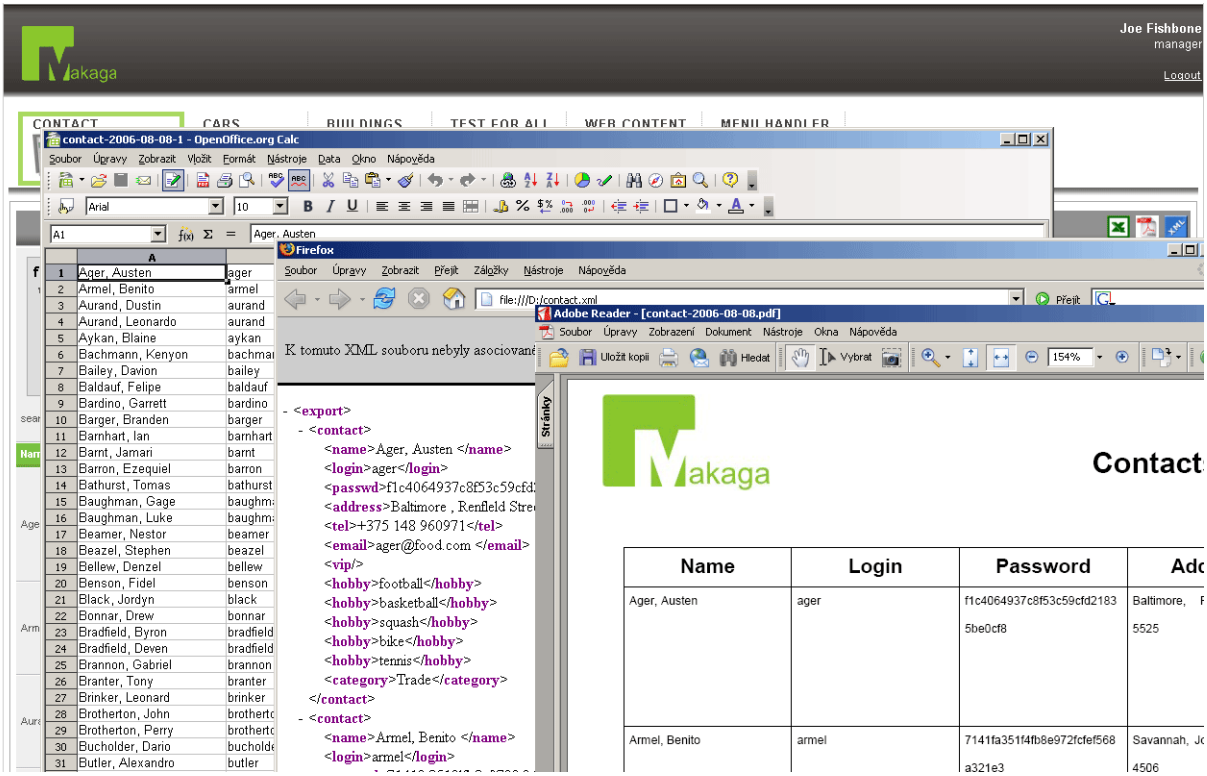


Obr. 8 Grafy

Grafy sú obrázky ohraničené rámčekom. V rámčeku sa nachádza koláčový graf s číselnými hodnotami vyjadrujúcimi počet tej ktorej položky. Vpravo od grafu sa nachádza legenda.

2.8. Exporty

Kliknutím na ľubovoľnú z troch ikon dôjde k tomu, že sa zjaví dialógové okno, ktoré dá možnosť vyexportovaný súbor uložiť alebo zobraziť.



Obr. 9 Náhľady exportovaných súborov

2.9. Web Content

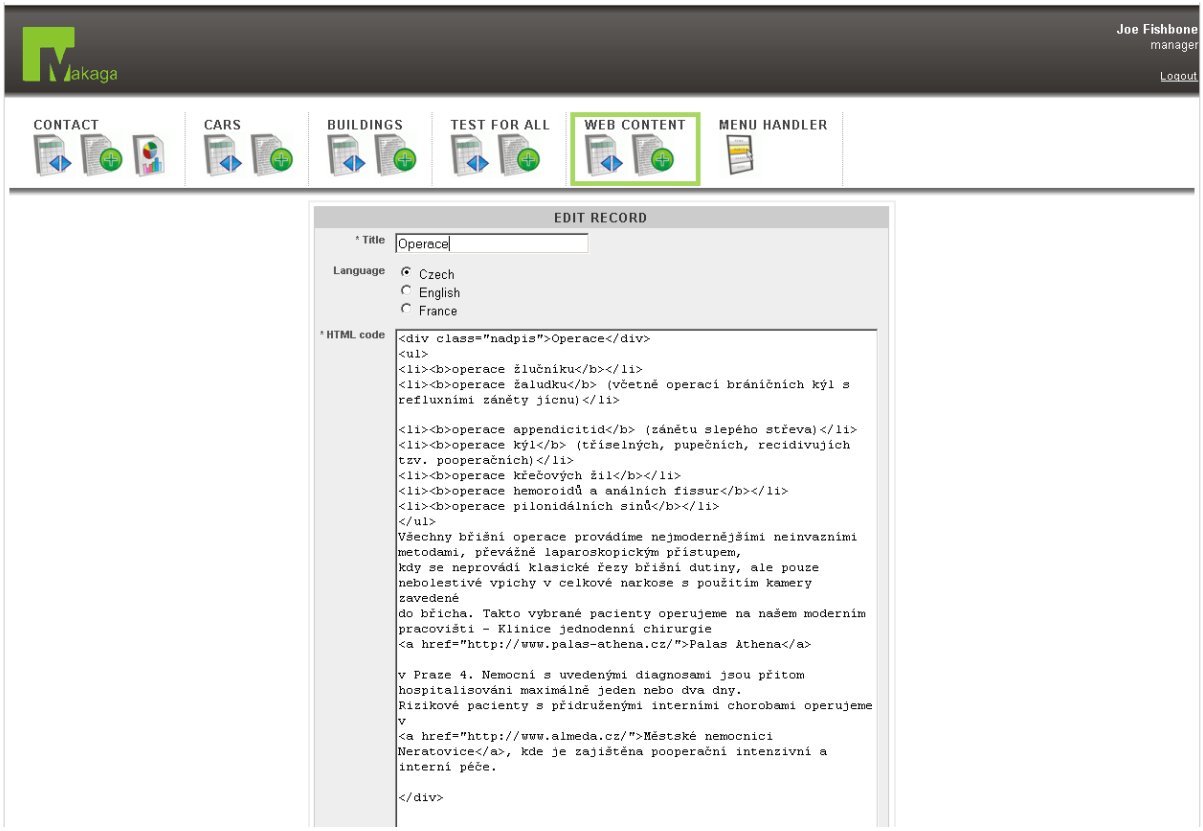
Web Content alebo Webový Obsah je jednou z tabuliek, ktoré by mal nakonfigurovaný systém obsahovať. Tabuľka obsahuje popis stránky a HTML kód obsahu stánky. V praxi ale býva zvykom že užívateľ dostane do správy nakonfigurovaný systém spolu vyplnenými záznamami pre Web Content. Prístup k tejto tabuľke by mal byť bežným užívateľom zamietnutý. Problémom je, že nepatrný preklep v HTML kóde môže spôsobiť rozhodenie celého front-endu.

Je potrebné si uvedomiť, že Web Content v sebe neobsahuje kompletnú stránku v HTML, ale len obsahovú časť. Čiže hlavičku, menu, pätičku neobsahuje. To sa v praxi ukázalo ako najefektívnejšie riešenie. Tým je aj znížená deštrukčná sila vyššie spomenutých preklepov v kóde HTML.

Samotné zadávanie HTML kódu je vo Web Contente obohatené o špeciálny tag `visual`. Ten obsahuje atribút `tpl`. Tag `visual` sa vo front-ende zamení za tabuľkové dáta. Čiže slúži na to, aby sa dáta z tabuliek zadávaných v back-ende mohli sprístupniť užívateľom prezerajúcim si front-end.

To aké dáta sa náhradou tagu `visual` zobrazia závisí od hodnoty `tpl`. Užívateľ má na výber len nejaké konkrétne hodnoty, ktoré vznikajú konfiguráciou systému. Čiže každá inštancia konkrétneho CMS Makaga spravidla obsahuje vlastné hodnoty atribútu `tpl`, ktoré sa zobrazujú len u tejto inštancie. Užívateľ, ktorý je poverený správou tabuľky Web Content, by

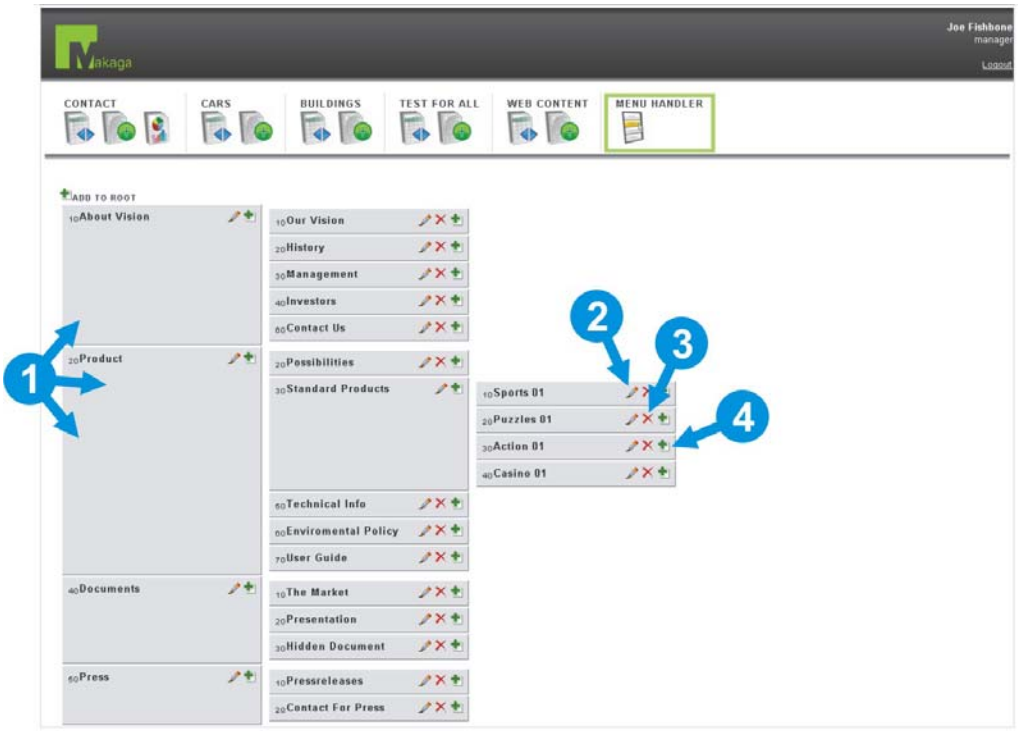
mal byť uvedomený o existencií tohto tagu a výčtom možných hodnôt atribútu `tpl` od osoby, ktorá systém konfigurovala. Ďalej by mal byť oboznámený s tým, ktoré šablóny (o šablónach viac na v kapitole 3.5) sa pri zámene tagu na front-ende používajú.



Obr. 10 Formulár pre Web Content

2.10. Menu Handler

Menu handler je špeciálny modul, ktorý dokáže vytvárať štruktúru menu. Podobne ako v Web Content z predošlej kapitoly, aj menu naplnenie dát Menu Handlera by malo byť súčasťou konfigurácie. Príklad pracovnej plochy Menu Handlera vyzerá takto:



Obr. 11 Menu Handler

Menu Handler sa skladá z viacerých častí (na obrázku znázornených šípkou):

1. Tu sú znázornené jednotlivé bunky menu. Sivý rámček predstavuje jednu položku menu.
2. Po kliknutí na ikonku ceruzky sa užívateľovi zjaví editačný formulár, kde zvolí názov položky menu a priradí obsah stránky. Ako poslednú zadá prioritu bunky menu. Čím je priorita nižšia, tým je položka v rámci podmenu umiestnená vyššie.
3. Po kliknutí na ikonku s červeným znakom "X" dôjde ku zmazaniu príslušnej bunky menu.
4. Po kliknutí na zelenú ikonu menu sa objaví formulár, ktorý po vyplnení a odoslaní vytvorí novú položku podmenu. Ide o ten istý formulár ako pri zadávaní.

2.11. Front-end

Prístup ku front-endu je na `http:/domena/index.php`. Čo sa ovládania týka, to je väčšinou individuálne v každej inštancií a takisto grafika je väčšinou iná. Bežnému užívateľovi front-end predstavuje internetovú stránku. Je na osobe, ktorá systém konfiguruje, aby bola stránka ergonomicky prístupná.

Obr. 12 Náhľady front-endov troch inštancií

3. Inštalácia konkrétnej webovej aplikácie

Inštalácia v skratke zahrňuje prekopírovanie súborov z priloženého CD, nastavenie prístupu na databázu a spustenie SQL skriptov.

Konfigurácia spočíva v nastavení troch konfiguračných súborov, príprava grafiky pre front-end v podobe tvorby šablón a nakoniec vloženie záznamov do Web Contentu (kapitola 2.9) a Menu Handlera (kapitola 2.10)

3.1. Inštalácia: 1. Kopírovanie

Prvou časťou inštalácie je kopírovanie súborov z adresára `install/basic/` na priloženom CD do htdocs webového servera. Adresárová štruktúra je nasledovná:

admin/	V tomto adresári sa nachádzajú súbory back-endu. Obsahuje spúšťací `index.php` a kaskádové štýly pre administračný mód.
classes/	Tu sa nachádzajú triedy, ktoré program využíva.
config/	Konfiguračné súbory.
constants/	Súbory s konštantami.
export/	Skripty, ktoré generujú dáta pre export.
images/	Uploadnuté obrázky.
javascript/	Javascript súbory.
pic/	Obrázky**,** ktoré využíva back-end.
template/	Adresáre a súbory so šablónami.
upload/	Uploadnuté súbory.

3.2. Inštalácia: 2. Nastavenie databázy

Nastavenie databázy spočíva v editácii súbor `config/dbconf.php`. Ten obsahuje definíciu štyroch konštánt:

- **SQL_CONNECT_SERVER** – URL databázového servera
- **SQL_CONNECT_USER** – užívateľské meno
- **SQL_CONNECT_PASSWORD** - heslo
- **SQL_CONNECT_DB** – názov databázy

Správne nastavenie tohto súboru zaručí napojenie na databázu, čo je pre chod aplikácie nevyhnutné. Príklad definície:

```
<?php
define ('SQL_CONNECT_SERVER',           'localhost');
define ('SQL_CONNECT_USER',             'sqlusr');
define ('SQL_CONNECT_PASSWORD',         'l6.tz4}0eIlb@');
define ('SQL_CONNECT_DB',               'portal');
?>
```

3.3. Inštalácia: 3. Tvorba tabuliek

3.3.1. Tabuľky užívateľov a ich práva

V kapitole 2.1 je návod na prihlásenie do systému za pomoci užívateľského mena a hesla. Konfiguráciu prístupu k týmto údajom cez Makagu je potrebné riešiť s veľkou

opatrnosťou. Odporúča sa to len v prípade, že práva vytvárať položky má len jeden užívateľ vo zvláštnej skupine. Inak by mohlo dochádzať ku nekorektným operáciám, ako zmena práv bežného užívateľa na administrátora a opačne. Mazanie užívateľov by nemalo byť povolené nikomu.

Na začiatku je potrebné vytvoriť tabuľky cez rozhranie databázy (napr. pomocou aplikácie *phpMyAdmin* – kapitola 1.2.4) Samotný sql skript pre tvorbu potrebných tabuliek sa nachádza na priloženom CD v adresári `install/basic/makaga.sql`. Ten zároveň vloží potrebné údaje do tabuliek. Vytvorí skupinu s názvmon *manager* a užívateľa s prihlasovacím menom *admin* a heslom *betasystem.* Skript obsahuje príkazy na tvorbu tabuliek *users*, *groups* a *rights*. Príkaz na tvorbu tabuľky *users* vyzerá následovne:

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(40) default NULL,
  `login` varchar(40) default NULL,
  `passwd` varchar(40) default NULL,
  `group_id` int(11) NOT NULL default '0',
  PRIMARY KEY  (`id`),
  UNIQUE KEY `login` (`login`)
) TYPE=MyISAM AUTO_INCREMENT=1 ;
```

Tabuľka *users* obsahuje informácie o užívateľovi. Jeho meno, login, heslo zahashované funkciou MD5 a skupinu, pod ktorú patrí. Skupiny sú v tabuľke *groups*, ktorej definícia vyzerá nasledovne:

```
CREATE TABLE `groups` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(40) default NULL,
  PRIMARY KEY  (`id`)
) TYPE=MyISAM AUTO_INCREMENT=1 ;
```

Tabuľka *groups* je naplnená informáciami o názve skupiny a jej identifikátore. Samotné práva sú uložené v tabuľke *rights* s touto štruktúrou:

```
CREATE TABLE `rights` (
  `group_id` int(11) NOT NULL default '0',
  `table_name` varchar(255) NOT NULL default '0',
  `mask` int(4) NOT NULL default '0',
  PRIMARY KEY  (`group_id`,`table_name`)
) TYPE=MyISAM;
```

Tabuľka obsahuje identifikátor skupiny, názov tabuľky, ku ktorej sa práva skupiny vzťahujú a masku.

Maska je riešená nasledujúcim spôsobom: Na každú tabuľku existuje 5 druhov práv:

- právo prezerať s hodnotou 1
- právo prezerať detaily záznamov s hodnotou 2
- právo editovať záznamy s hodnotou 4
- právo pridávať záznamy s hodnotou 8
- právo mazať záznamy s hodnotou 16

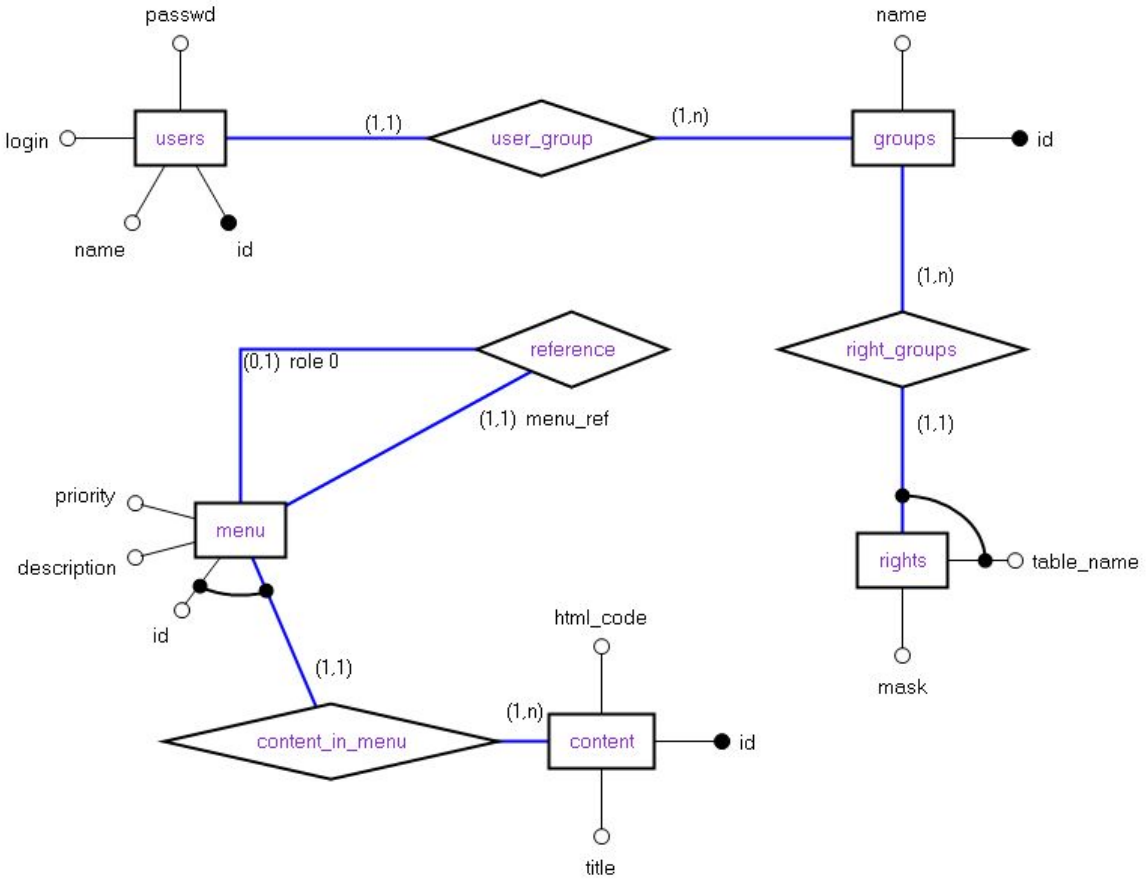
Maska obsahuje súčet hodnôt práv. Príklad: Ak chceme skupine s názvom *administrators* prideliť všetky práva, jej maska bude 1+2+4+8+16 = 31. A ak skupine *secretary* chceme prideliť len právo na prezeranie a vytváranie nových záznamov, musíme jej prideliť masku 1+8=9.

3.3.2. Tabuľky Web Content

Tabuľku pre Web Content by mala štandardne obsahovať každá konfigurácia. Spomínaný adresár `install/basic/makaga.sql` na CD obsahuje aj definíciu tejto tabuľky. Je ju možno administrovať cez rozhranie CMS, ako bolo napísané v kapitole 2.9.

3.3.3. Tabuľky Menu Handler

S tabuľkou pre Web Content súvisí aj tabuľka pre Menu Handler. Skript opäť v adresári `install/basic/makaga.sql` na CD. Tvorba konkrétneho menu a prideľovanie obsahu pre bunky menu sa vytvorí podľa návodu v kapitole 2.10.



Obr. 13 ER diagram tabuliek *users*, *groups*, *rights*, *menu* a *content*

3.3.4. Tabuľky s dátami

Obmedzenie na tabuľky s dátami spočíva v predpoklade, že každá tabuľka obsahuje primárny kľúč *id* s vlastnosťou auto_increment. Príklad:

```
CREATE TABLE `car` (
  `id` int(11) NOT NULL auto_increment,
  `type` varchar(255) default NULL,
  `lic_number` varchar(255) default NULL,
  `owner` int(11) default NULL,
  PRIMARY KEY  (`id`)
) TYPE=MyISAM;
```

3.3.5. Prepojovacie tabuľky

Pre prepojovacie tabuľky platí dodatočné obmedzenie, že musia obsahovať stĺpec s názvom *join_col*, ktorý predstavuje cudzí kľúč tabuľky, ktorá prepojovaciu tabuľku využíva. Príklad:

```
CREATE TABLE `projectants` (
  `id` int(11) NOT NULL auto_increment,
  `join_col` int(11) default NULL,
  `id_contact` int(11) default NULL,
  `payment` int(11) default NULL,
  PRIMARY KEY  (`id`)
) TYPE=MyISAM ;
```

V stĺpci *join_col* sa nachádza povedzme cudzí kľúč tabuľky architektonických projektov a v stĺpci *id_contact* sa nachádza cudzí kľúč kontaktov na osoby.

3.4. Inštalácia: 4. Konfiguračné súbory a ich vlastnosti

V nasledujúcej časti budú popísané dátové štruktúry, ktoré popisujú *formuláre, elementy* a *pohľady*.

Element popisuje jeden konkrétny HTML element formuláru. To znamená checkbox, input text box, radio button, ... Obsahuje informácie o tom, či je vyplnenie elementu povinné, do ktorého stĺpca tabuľky sa majú dáta zapísať a podobne.

Pohľad združuje informácie o jednom konkrétom zobrazení tabuľkových dát. Zobrazenie musí mať definované šablóny a vlastnosti navigačných prvkoch ako sú listovanie, radenie, vyhľadávanie, filter,...

Formulár je základná jednotka, ktorá združuje informácie ako názov tabuľky a k nej prislúchajúce *elemenenty* a *pohľady*

Konfiguračné súbory sa nachádzajú v adresári `/config`. Ten obsahuje nasledujúce súbory:

- `config_forms.php` – súbor obsahuje súhrnné informácie o *formulároch*.
- `config_elements.php` – v tomto súbore sa nachádzajú polia popisujúce *elementy* formulára.
- `config_view.php` – v tomto súbore zase polia, ktoré popisujú *pohľady.*

3.4.1. config_forms.php

Najprv bude uvedený príklad kódu zo súboru `config_forms.php` a potom rozobrané jednotlivé konfiguračné možnosti a vlastnosti.

```
...
...
$form[2] = array (
      FORM_LABEL => 'BUILDINGS',
      FORM_TABLE => 'building',
      FORM_TYPE  => FORM_TYPE_NORMAL,
      FORM_ELEMENTS => &$building,
      FORM_VIEW_LIST => &$view_building,
      FORM_VIEW_DETAIL => &$view_building,
      );

$form[3] = array (
      FORM_LABEL => 'PROJECTS',
      FORM_TABLE => 'project',
      FORM_TYPE  => FORM_TYPE_JOIN,
      FORM_ELEMENTS => &$project,
      FORM_VIEW_LIST => &$view_project,
      FORM_VIEW_DETAIL => &$view_project,
      );

...
...

$form[8]=array (
      FORM_LABEL => 'MENU HANDLER',
      FORM_TABLE => 'menu',
      FORM_TYPE  => FORM_TYPE_MENU_HANDLER,
      FORM_ELEMENTS => $menuhand,
      FORM_VIEW_LIST => null,
      FORM_VIEW_DETAIL => null,
);
...
...
```

V konfiguračnom súbore sa nachádza pole `$form`. Každá položka poľa predstavuje jeden *formulár*. Prvý rozmer poľa `$form` určuje poradové číslo *formulára*. Druhým rozmerom je jedná z konštánt a určuje už vlastnosť konkrétneho formulára:

- **FORM_TABLE** - kľúč, ktorý definuje názov tabuľky, s ktorou *formulár* pracuje
- **FORM_LABEL** - kľúč, ktorého hodnota definuje popisný názov *formulára*
- **FORM_ELEMENTS** - kľúč poľa, kde hodnotou je referencia na *množinu elementov* (detaily ďalej v kapitole 3.4.2)
- **FORM_VIEW_LIST** - kľúč poľa, ktorý predstavuje odkaz na *pohľad*, ktorý sa bude zobrazovať pri prezeraní záznamov.
- **FORM_VIEW_DETAIL** - kľúč poľa, kde hodnota predstavuje odkaz na *pohľad*, ktorý sa bude zobrazovať pri detailnom náhľade na záznam. (o pohľadoch viac kapitola 3.4.3)
- **FORM_VIEW_EXPORT** - definuje *pohľad* na dáta, ktoré sa budú exportovať
- **FORM_TYPE** - predstavuje odkaz na typ *formuláru*. Môže ísť o následné možnosti:
  - **FORM_TYPE_NORMAL** - klasický typ formulára, ktorý sa zobrazuje v menu
  - **FORM_TYPE_JOIN** - ide o pomocný typ formulára, ktorý sa v menu nezobrazuje a slúži ako prepojovacia tabuľka.
  - **FORM_TYPE_MENU_HANDLER**- ide o špeciálny formulár, ktorý zobrazí rozhranie pre tvorbu štruktúry menu pre front-end (ide o typ popisovaný v 2.10).
- **FORM_GRAPH** – kľúč poľa, ktorého hodnota predstavuje pole *elementov*, na ktoré sa majú generovať koláčové grafy

3.4.2. config_elements.php

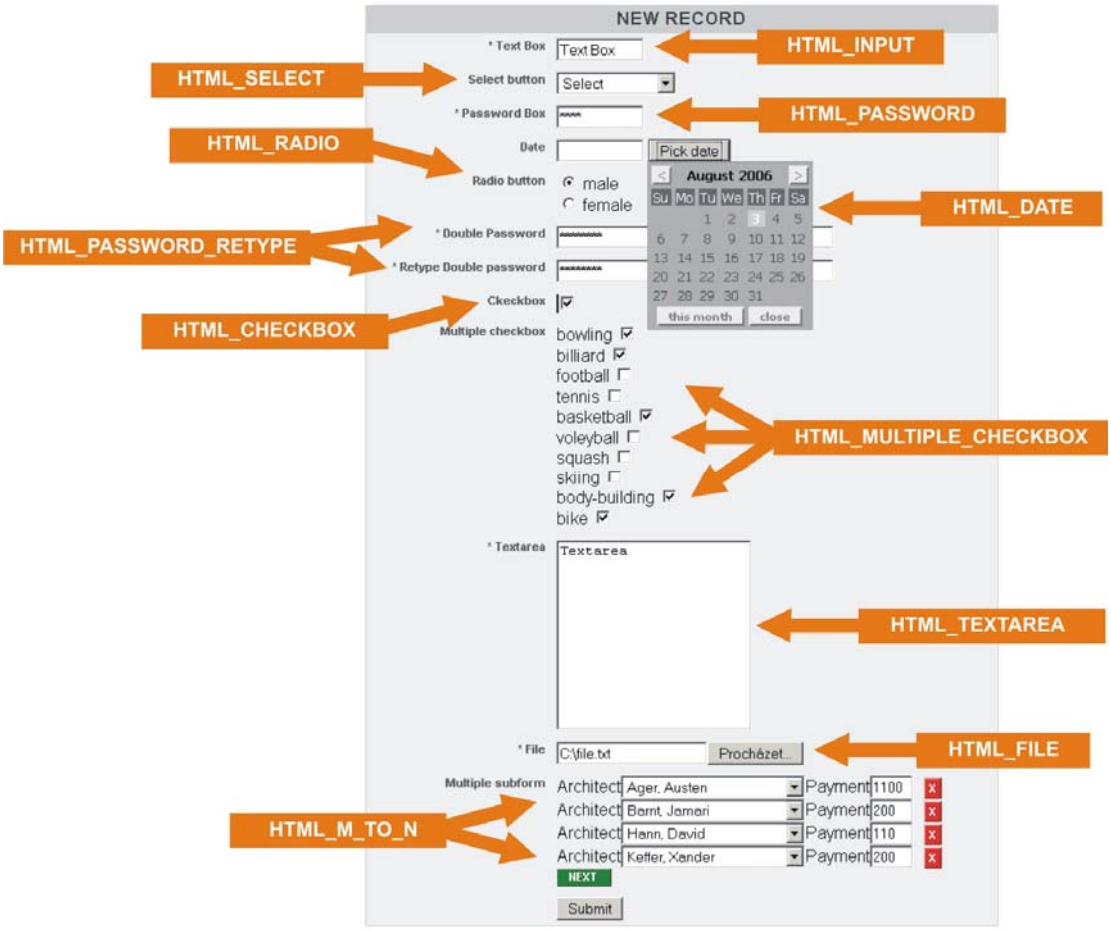
V `config_elements.php` sa nachádzajú popisy formulárových *elementov*. Každé pole v tomto súbore predstavuje *množinu elementov*, ktoré prislúchajú jednému *formuláru*. Každá položka poľa predstavuje jeden konkrétny *element*. Príklad dvoch elementov formulára:

```
$building[] = array(
      FMAP_HTML_LABEL           => 'Building',
      FMAP_HTML_TYPE            => HTML_INPUT,
      FMAP_HTML_NAME            => 'name',
      FMAP_HTML_INPUT_SIZE      => 30,
      FMAP_SQL_COL              => 'name',
      FMAP_SQL_APOSTROPHE       => true,
      FMAP_SQL_DUTY             => true,
      FMAP_SQL_REGEXP           => REGEXP_STRING,
      FMAP_SQL_REGEXP_IF_ERROR  => 'Building name is empty'
      );

$building[] = array(
      FMAP_HTML_LABEL           => 'Year',
      FMAP_HTML_TYPE            => HTML_INPUT,
      FMAP_HTML_NAME            => 'year',
      FMAP_HTML_INPUT_SIZE      => 4,
      FMAP_SQL_COL              => 'year',
      FMAP_SQL_APOSTROPHE       => false,
      FMAP_SQL_DUTY             => true,
      FMAP_SQL_REGEXP           => REGEXP_NUMBER,
      FMAP_SQL_REGEXP_IF_ERROR  => ' Year is empty'
      );
```

Nasleduje vysvetlenie jednotlivých kľúčov a hodnôt *elementov.* Zároveň je rozobrané, ako pri konkrétnom *elemente* kombinovať dané kľúče a hodnoty.

- **FMAP_HTML_LABEL** – využíva sa pri všetkých *elementoch*. Vyjadruje popis elementu.
- **FMAP_HTML_TYPE** – predstavuje *typ elementu*. Nadobúda jednu z hodnôt:
  - HTML_INPUT – jednoduchý input text box
  - HTML_DATE – input text box s tlačidlom pre voľbu dátumu
  - HTML_HIDDEN – skrytá položka vo formulári
  - HTML_TEXTAREA – textové pole
  - HTML_SELECT – select box
  - HTML_RADIO – radio button
  - HTML_CHECKBOX - checkbox
  - HTML_CHECKBOX_MULTIPLE – skupina checkboxov
  - HTML_PASSWORD – text box so zámenou textu za znak hviezdička
  - HTML_PASSWORD_RETYPE – zdvojený predošlý typ s kontrolou na rovnako zadané hodnoty
  - HTML_FILE – formulárový element na upload súborov
  - HTML_IMAGE – podobný predošlému typu, ale určený na upload obrázkov
  - HTML_M_TO_N – špeciálny typ, ktorý dokáže priradzovať k jednému elementu viacero riadkov iného formulára. Bližšie informácie a konfigurácia bude rozobraná neskôr.



Obr. 14 Príklad typov HTML elementov

- **FMAP_HTML_NAME** – unikátny identifikátor, ktorý určuje atribút name v html kóde v príslušnom tagu.
- **FMAP_HTML_INPUT_SIZE** – zadáva *typ elementu* HTML_INPUT, HTML_DATE, HTML_PASSWORD a HTML_PASSWORD_RETYPE. Hodnota nastavuje atribút size pri formulárovom tagu. Čo v praxi značí na koľko znakov je nastavené šírka elementu.
- **FMAP_HTML_TEXTAREA_COLS** – v prípade že *typ elementu* je HTML_TEXTAREA, je potrebné zadať túto hodnotu, ktorá nastavuje atribút cols v HTML tagu textarea. Ten určuje počet riadkov.
- **FMAP_HTML_TEXTAREA_ROWS** – v prípade že *typ elementu* je HTML_TEXTAREA, je potrebné zadať túto hodnotu, ktorá nastavuje atribút rows v HTML tagu textarea. Ten určuje počet stĺpcov.
- **FMAP_SQL_COL –** nachádza sa v každej definícii *elementu* a určuje názov stĺpca tabuľky s ktorým je *element* spätý.
- **FMAP_SQL_APOSTROPHE** – takisto obsahuje každá definícia *elementu*. Hodnotou je *true*/*false* podľa toho, či sa po odoslaní formuláru má ukladať zadaná hodnota do databáze ako reťazec alebo nie.
- **FMAP_SQL_DUTY**– opäť obsahuje každá definícia *elementu* a hovorí o tom, či je vyplnenie tohto elementu užívateľom povinné.
- **FMAP_SQL_REGEXP –** ak je predošlá hodnota nastavená na *true*, je potrebné zadať regulárny výraz, podľa ktorého sa bude hodnotiť platnosť odoslanej hodnoty.

- **FMAP_SQL_REGEXP_IF_ERROR** – táto hodnota obsahu je reťazec, ktorý sa má vypísať užívateľovi, ak hodnota odoslaného formulára neodpovedá regulárnemu výrazu zadaného v FMAP_SQL_REGEXP.
- **FMAP_FILE_MIME –** v prípade, že *typ elementu* je HTML_FILE alebo HTML_IMAGE je definovanie tejto vlastnosti povinnosťou. Hodnotou je pole, ktorého hodnoty sú MIME typy podporovaných súborov.
- **FMAP_FILE_MAX_SIZE –** v prípade uploadu súboru je potrebné nastaviť maximálnu veľkosť súboru v bajtoch.
- **FMAP_IMAGE_THUMB –** ak je *typ elementu* HTML_IMAGE, tento prepínač nastavuje, či ma systém automaticky vygenerovať zmenšený náhľad obrázku.
- **FMAP_IMAGE_THUMB_WIDTH –** ak je predošlá hodnota *true*, je potrebné touto vlastnosťou zadať šírku zmenšeného obrázku.
- **FMAP_REFERENCE –** v prípade, že *typ elementu* je HTML_SELECT, alebo HTML_RADIO, je potrebné nastaviť výčtové hodnoty, ktoré tieto typy obsahujú. Príklad:

```
$alltest[] = array(
      FMAP_HTML_LABEL     => 'Gender',
      FMAP_HTML_TYPE      => HTML_RADIO,
      FMAP_HTML_NAME      => 'gender',
      FMAP_SQL_COL        => 'gender',
      FMAP_REFERENCE      => array (

                          0 => array (
                                 ENUM_LABEL    => 'male',
                                 ENUM_VALUE    => 'male',
                                 ENUM_SELECTED => true),
                          1 => array (
                                 ENUM_LABEL    => 'female',
                                 ENUM_VALUE    => 'female',
                                 ENUM_SELECTED => false),
                                 ),

      FMAP_SQL_APOSTROPHE        => true,
      );
```

Každá položka poľa s kľúčom FMAP_REFERENCE predstavuje jednu hodnotu výčtu. Každá položka výčtu obsahuje vnorené pole s týmito kľúčmi:

  - **ENUM_LABEL –** popis výčtovej položky
  - **ENUM_VALUE –** hodnota položky, ktorá sa uloží do databáze
  - **ENUM_SELECTED –** prepínač. Hodnota *true* sa zadáva vtedy, ak má byť daná položka pri zobrazení formulára nastavená implicitne. Hodnotu *true* by mala obsahovať len jedna položka výčtu.

- **FMAP_HTML_VALUE –** ak ide o *typ elementu* HTML_HIDDEN alebo HTML_CHECKBOX, je potrebné zadať hodnotu, ktorú má formulár po odoslaní preniesť. HTML_HIDDEN totiž prenáša dáta, ktoré sú nastavené v atribúte *value* tagu `<input type=″hidden″`. Aj keď sa checkbox správa dvojstavovo, v skutočnosti prenáša len jedinú hodnotu, a to vtedy, ak je checkbox zaškrtnutý. Nejedná sa ale o hodnotu true, ale o hodnotu, ktorá je nastavená v atribúte *value* HTML tagu checkbox.
- **FMAP_HTML_LABEL_RETYPE –** tento kľúč a hodnota sa nastavuje pri *type elementu* HTML_PASSWORD_RETYPE. Tento typ pod sebou zobrazí dvakrát HTML_PASSWORD. Užívateľ musí oba vyplniť zhodným reťazcom. Uvedený typ sa často využíva na vytvorenie nového hesla užívateľovi. Ten ho musí zadať dvakrát, aby v prípade preklepu nedošlo k nastaveniu nežiaduceho hesla. Je vylepšený o to, že po

odoslaní formulára dochádza ku kontrole oboch zadaných hesiel na zhodu. FMAP_HTML_LABEL_RETYPE hovorí o popise druhého input boxu.

- **FMAP_HTML_NAME_RETYPE –** súvisí s predošlým kľúčom. Tu sa jedná o HTML_NAME pre druhý input box.
- **FMAP_SQL_NOT_MATCH_ERROR -** opäť súvisí s *typom elementu* HTML_PASSWORD_RETYPE. Ide o chybový reťazec, ktorý sa zobrazí užívateľovi, ak došlo k nezhode hesiel.
- **FMAP_DEPENDENCY_TYPE –** Táto položka obsahuje 2 hodnoty:
  - **DEPENDENCY_1_TO_N**
  - **DEPENDENCY_M_TO_N**

Tieto položky sa definujú v troch prípadoch:

a.) Prvým prípad súvisí s DEPENDENCY_1_TO_N. Ten v princípe hovorí o závislosti jednotlivých formulároch. V požiadavkách stojí, že systém má podporovať prepájanie formulárov. Demonštrácia, ako takúto závislosť nastaviť:

```
$car[] = array(
     FMAP_HTML_LABEL           => 'Owner',
     FMAP_HTML_TYPE            => HTML_SELECT,
     FMAP_DEPENDENCY_TYPE      => DEPENDENCY_TYPE_1_TO_N,
     FMAP_TARGET_MAP           => &$contact,
     FMAP_TARGET_TABLE         => 'contact',
     FMAP_TARGET_COLS          => array(0,3),
     FMAP_TARGET_SEPARATOR     => ', ',

     FMAP_HTML_NAME            => 'owner',
     FMAP_SQL_COL        => 'owner',
     FMAP_SQL_APOSTROPHE => false,
     );
```

Do definície *elementu* v tomto prípade pribudli kľúče:

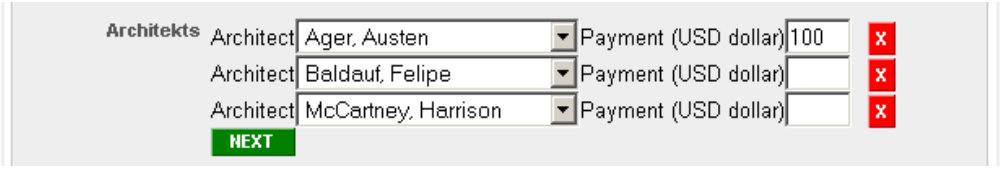
- FMAP_TARGET_MAP – ide o referenciu na pole cudzích *elementov*
- FMAP_TARGET_FORM_ID – kľuč cudzieho *formulára*
- FMAP_TARGET_TABLE – tabuľka, kde sa dáta cudzích *elementov* nachádzajú
- FMAP_TARGET_COLS – pole identifikátorov cudzích elementov, ktoré sa neskôr zobrazia vo výčte ako ENUM_LABEL
- FMAP_TARGET_SEPARATOR – oddeľovač, ktorý bude jednotlivé stĺpce z predošlého elementu oddeľovať

b.) Druhý prípad súvisí s DEPENDENCY_M_TO_N a *typom elementu* HTML_CHECKBOX_MULTIPLE. Tento typ indukuje, že na zapamätanie odoslaných dát bude potrebná prepojovacia tabuľka. Názov tejto tabuľky sa nachádza v hodnote s kľúčom **FMAP_TARGET_JOIN_TABLE.**

Príklad konfigurácie MULTIPLE_CHECKBOX:

```
$contact[] = array(
     FMAP_HTML_LABEL           => 'Hobby',
     FMAP_HTML_TYPE            => HTML_CHECKBOX_MULTIPLE,
     FMAP_HTML_NAME            => 'hobby',
     FMAP_DEPENDENCY_TYPE      => DEPENDENCY_TYPE_M_TO_N,
     FMAP_TARGET_JOIN_TABLE  => 'hobby',
     FMAP_REFERENCE            => &$enum_hobby,
     FMAP_SQL_COL        => 'hobby',
     FMAP_SQL_APOSTROPHE => false,
     FMAP_SQL_DUTY       => false,
     );
```

c.) Posledný tretí prípad sa vzťahuje ku DEPENDENCY_M_TO_N a *typu elementu* HTML_M_TO_N. V tomto prípade sa správa element tak, že po stlačení tlačidla *next* sa automaticky pridá riadok formulára na ktorý sa konfigurácia odkazuje. Po kliknutí na tlačidlo *X* zase príslušný riadok zmizne. Obrázok 5 ukazuje zobrazenie troch riadkov cudzieho formulára.



Obr. 15 Príklad multiple subform

Samotná konfigurácia vyzerá takto:

```
$building[] = array(
FMAP_HTML_LABEL              => 'Architekti',
FMAP_HTML_TYPE               => HTML_M_TO_N,
FMAP_HTML_NAME               => 'architekti',
FMAP_DEPENDENCY_TYPE         => DEPENDENCY_TYPE_M_TO_N,

FMAP_TARGET_JOIN_FORM_MAP => &$form[3],
FMAP_SQL_COL                 => 'architects',
FMAP_SQL_APOSTROPHE          => false,
FMAP_SQL_DUTY                => false,
);
```

V tejto konfigurácií pribudol kľúč **FMAP_TARGET_JOIN_FORM_MAP,** kde hodnota je referenciou na nejaký *formulár* z `config_forms.php` (kapitola 3.4.1). Typ tohto formulára musí byť nastavený na TYPE_JOIN.

Ak má dôjsť ku plnohodnotnej konfigurácií m:n, je nutné, aby sa vo *formulári* v FMAP_TARGET_JOIN_FORM_MAP nachádzala mapa elementov, ktorá obsahuje element s typom závislosti DEPENDENCY_1_TO_N.

3.4.3. config_view.php

V `config_view.php` sú popísané vlastnosti *pohľadov*. Príklad *pohľadu* môže vyzerať takto:

```
$view_contact = array (
     VIEW_COLS           =>    array(0,3,4,5,6,7,9),
     VIEW_PER_PAGE       =>    20,

     VIEW_IS_ORDER       =>    true,
     VIEW_IS_LISTING     =>    true,
     VIEW_IS_SEARCH      =>    true,
     VIEW_IS_FILTER      =>    true,

     VIEW_SEARCH_COLS    =>    array (0,1,2),
     VIEW_FILTER_COLS    =>    array (6,7,8,9),

     VIEW_START_ORDER     =>   1,
     VIEW_START_DIRECTION =>   DIRECTION_ASC,

     VIEW_START_LISTING    =>  0,

     VIEW_TPL_SEARCH  =>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'search.php',
     VIEW_TPL_ORDER   =>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'order.php',
     VIEW_TPL_BODY    =>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'body.php',
     VIEW_TPL_HEAD    =>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'head.php',
     VIEW_TPL_LISTING =>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'list.php',

);
```

Jednotlivé kľúče znamenajú:

- **VIEW_COLS** – predstavuje kľúče *elementov* z `config_elements.php`, ktoré sa majú do pohľadu načítavať.
- **VIEW_PER_PAGE –** definuje, koľko záznamov sa má zobraziť (na jednu stránku). Aj v prípade, že stránkovanie nie je zapnuté, zobrazí sa len nastavený počet záznamov.
- **VIEW_IS_ORDER** – nastavuje sa na *true*, ak je aktivované radenie záznamov, inak sa nastaví na *false*.
- **VIEW_IS_LISTING** – nastavuje sa na *true*, ak je aktivované listovanie záznamov, inak sa nastaví na *false*.
- **VIEW_IS_SEARCH** – nastavuje sa na *true*, ak je aktivované vyhľadávanie záznamov, inak sa nastaví na *false*.
- **VIEW_IS_FILTER** – nastavuje sa na *true*, ak je aktivovaný filter záznamov, inak sa nastaví na *false*.
- **VIEW_SEARCH_COLS** – pole kľúčov elementov, nad ktorými prebieha vyhľadávanie. Zadáva sa, ak je hodnota VIEW_IS_SEARCH nastavená na *true.*
- **VIEW_FILTER_COLS** – pole kľúčov *elementov*, na ktoré sa ma vygenerovať filtrovací formulár.
- **VIEW_START_ORDER –** v príde, že je povolené radenie záznamov podľa stĺpcov nastavením VIEW_IS_ORDER na *true*, túto vlastnosť určuje stĺpec, podľa ktorého sú dáta radené implicitne.
- **VIEW_START_DIRECTION –** súvisí s predošlým nastavením a určuje smer radenia. Na výber sú dve možnosti:
  - **DIRECTION_ASC –** vzostupné radenie
  - **DIRECTION_DESC –** zostupné radenie
- **VIEW_START_LISTING –** ak je nastavený prepínač VIEW_IS_LISTING na *true*, ktorý povoľuje listovanie záznamov, táto vlastnosť nastaví stránku, ktorá je nalistovaná pri zobrazení pohľadu ako prvá.
- **VIEW_TPL_SEARCH –** určuje adresu šablóny pre vyhľadávanie.
- **VIEW_TPL_ORDER –** adresa šablóny pre listovanie.
- **VIEW_TPL_BODY –** adresa šablóny pre telo zobrazovaných záznamov.

- **VIEW_TPL_HEAD -** šablóna hlavičky pohľadu.
- **VIEW_TPL_LISTING –** šablóna pre listovanie položiek.

3.5. Inštalácia: 5. Šablóny

3.5.1. Šablóny všeobecne

Šablóny by mali zaisťovať oddelenie grafiky (HTML, CSS) od funkčnosti. To znamená, že funkčná logika by sa mala nachádzať v iných skriptoch, ako zobrazovacia časť. Celkový proces prípravy dát a ich zobrazovanie by malo prebiehať v týchto krokoch:

- Funkčná logika aplikácie (výber dát z databáze, generovanie hlásení, ...)
- Príprava premenných pre šablónu. Jedná sa o medzistupeň. Pripravujú sa tu premenné s tými názvami, ktoré používa šablóna.
- Načítanie šablóny. V šablóne sa nachádza HTML kód so zámenou premenných za ich hodnoty. Väčšinou však šablónovacie systémy musia obsahovať podporu dynamických blokov, vetvenia a volanie funkcií aby mohlo dôjsť ku korektnému zobrazeniu. Hlavne u graficky náročnejších stránok.

Príklad, ktorý načrtá šablónové riešenie:

```
<?php
/*
 * skript show-page.php
 */

//funkcna logika
$q = 'SELECT `title`,`content` FROM `page` WHERE `id` = 7';
$page_data = $conn->query_fetch_assoc_all($q);

...
...

//priprava premennych
$tpl_title = $page_data[0]['title'];

...
...


//zobrazenie template
include('template/tpl_page.php');

?>
```

```
<!-- tpl_page.php -->
<html>
<head>
<title> <?=$tpl_title?> </title>
</head>
<body>
<b> hello world </b>
...
...
```

Systém Makaga nevyužíva žiadny už hotový šablónovací engine, ako sú napríklad *Smarty* [17] či *TinyButStrong* [18]. Využíva ale klasické PHP skripty, kde sa však nevykonáva nič okrem zobrazovania. Samotné šablóny sú teda uložené v súboroch s koncovkou *.php*. Platí konvencia, že zápis riadiacich štruktúr PHP kódu v šablóne je v tzv. alternatívnej syntaxi.

Jedná sa o štruktúry *if*, *while*, *for*, a *foreach*. Základom tejto syntaxe je zámena otváracej zloženej zátvorky za dvojbodku (znak '{' za ':'). Pravá zátvorka je zámena za *endif;, endwhile;, endfor;, endforeach;* Do PHP blokov sa uzatvárajú osobitne všetky častí PHP kódu. Naopak žiadna časť HTML kódu by nemala byť interpretovaná príkazom *echo* v PHP. Názorný príklad:

```
Vsetky polozky pola:
<? foreach ( $array as $item) : ?>
     Moja polozka: <?=$item;?>
<? endforeach; ?>
```

Výhody takéhoto zápisu sú nasledujúce:

- Hneď na prvý pohľad je zrejmé, či sa jedná o šablónu, alebo o klasický PHP skript.
- Pre designéra je tento kód ďaleko prehľadnejší. Stačí mu vedieť, že nemá zasahovať do častí medzi <? … ?> Takto mu odpadnú starosti so zloženými zátvorkami.
- Využitie všetkých výhod jazyka. Niekedy je totiž nutné použiť v šablóne zložitejšie podmienky, ako klasické šablónovacie systému povoľujú.
- Rýchlosť. Keďže kód sa nemusí interpretovať dvakrát (*Smarty*).

Nevýhody:

- Designér musí ovládať časť jazyka PHP, ktorý je syntakticky zložitejší oproti konštrukciám, ktoré poskytujú šablónovacie systémy typu *Smarty.*
- PHP kód môže ľahko znížiť prehľadnosť celej šablóny.

3.5.2. Rozdelenie šablón v systéme

Systém obsahuje množstvo šablón, ktoré sú rozdelené do rôznych skupín podľa funkcie. Všetky sa nachádzajú v adresári `template/`, ktorý tieto adresáre:

`export/`	šablóny pre export súborov
`form_in_filter/`	šablóny *elementov* zobrazovaných vo filtri
`form_in_row/`	šablóny *elementov* zobrazovaných pri *type elementu* HTML_M_TO_N (detaily v kapitole 3.4.2)
`form_in_table/`	šablóny *elementov* zobrazovaných v klasických formulároch
`front_end/`	šablóny front-endu
`menu_handler/`	šablóny Menu Handleru (viac kapitola 2.10)
`system/`	systémové šablóny ako hlavičky, chybové hlásenia,...
`view/`	šablóny *pohľadov* back-endu

Osoba, ktorá je poverená konfigurovaním systému, bude modifikovať (prípade vytvárať) len šablóny front-endu. Tie možno rozdeliť do troch skupín:

- **Šablóny layoutu -** tie zobrazujú a definujú layout ako taký. To znamená, že obsahujú grafiku pre hlavičku stránky, pätičku,...
- **Šablóny menu front-endu -** ide o šablóny, ktoré majú na starosti zobrazenie navigačného menu.
- **Šablóny pohľadov front-endu –** šablóny, ktoré vychádzajú zo šablón *pohľadu* back-endu. Tie by mali slúžiť na zobrazovanie tabuľkových dát. Pôjde o šablóny, ktoré sa nahradia za špecialný tag visual, ktorý bol spomenutý už v kapitole 2.9

3.5.3. Šablóny front-endu: layoutu

Tieto šablóny obsahujú HTML kód, ktorý nesie hlavný design front-endu. Väčšinou obsahujú len HTML a len zriedka zobrazenie nejakej premennej. Systém má kvôli jednoduchosti prednastavených niekoľko šablón:

- `header.php` – hlavička stránky. Väčšinou ide o nejaký obrázok alebo logo firmy na vhodnom pozadí. Môže obsahovať aktuálny dátum, kto má meniny, motto, …
- `before_content.php` – je šablóna, ktorá predstavuje časť stránky medzi `header.php` a Web Contentom.
- `footer.php` - je šablóna, ktorá zobrazí pätičku.

Poradie zobrazovanie je také, že sa najprv zobrazí `header.php`, potom sa zobrazí menu, potom `before_content.php`, za ním samotný obsah a nakoniec `footer.php`.

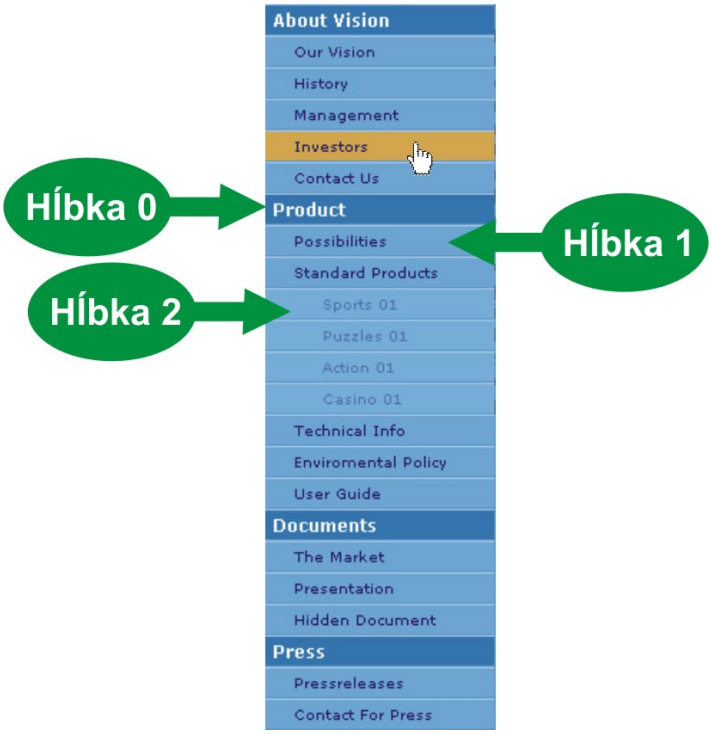
3.5.4. Šablóny front-endu: menu

Šablóna, ktorá zobrazí navigačné menu pre front-end. Implicitne sa nachádza pod názvom `menu.php` Šablóna má na starosti vizuálne spracovať obsah poľa `$menu`. Pole má nasledujúcu štruktúru:

Prvý rozmer poľa určuje poradové číslo položky. Druhý rozmer obsahuje viaceré konštanty. Pre šablóny sú zaujímavé tieto:

- **MENU_COL_ID** – Identifikátor položky menu. Ten je potrebné použiť pri hypertextových odkazoch v menu. Tým sa dosiahne toho, že po kliknutí na položku menu dôjde ku zobrazeniu príslušnej podstránky. Príklad:
  `<a href="index.php?page_id=<?=$menu[$j][MENU_COL_ID]?>">`
- **MENU_COL_DESCRIPTION –** hodnota je názvom bunky menu
- **MENU_COL_LEVEL –** určuje hĺbku podstránky

Záznamy sú radené tak, že za každou položkou menu nasleduje jej podmenu. Na jeden prechod poľa je možno zobraziť bezproblémovo celé menu v správnej štruktúre. Na základe MENU_COL_LEVEL sa má v šablóne rozhodovať či použiť štýl pre menu alebo podmenu.



Obr. 16 Príklad zobrazeného menu vo front-ende

3.5.5. Šablóny front-endu: pohľady a tag visual

Ide o najnáročnejšiu prípravu šablón. Ich význam súvisí s použitím špeciálneho tagu `visual`, ktorého použitie bolo spomenuté v kapitole 2.9. Ku korektnému nastaveniu šablón je potrebné vykonať tri kroky:

1. Prvým krokom je vytvoriť nový *pohľad*. Návod je v kapitole 3.4.3.
2. Druhým krokom je modifikovať (prípadne vytvoriť nanovo) šablóny nadefinované v *pohľade* z kroku 1.
3. V súbore `config/config_visualtag.php` vytvoriť (doplniť) pole s názvom `$visual_tag_arr`. Argumentom je reťazec. Ak tento reťazec bude obsahovať atribút `tpl` tagu `visual`, tak sa vo front-ende miesto tagu zobrazí práve pohľad vytvorený v kroku 1.

Hlavnou témou tejto kapitoly je popísať šablóny používané v pohľadoch. Ide o hodnoty kľúčov konfiguračných polí pohľadov v `config_view.php` (kapitola 3.4.3) pod kľúčmi:

VIEW_TPL_SEARCH
VIEW_TPL_HEAD
VIEW_TPL_ORDER
VIEW_TPL_BODY
VIEW_TPL_LISTING

Vo vyššie spomínaných šablónach sa požívajú tieto premenné(polia):

- `$url` – URL adresa práve spusteného skriptu.
- `$elements` – *množina elementov* aktuálne spracovávaného *formulára* z `config_elements.php` (kapitola 3.4.2).
- `$view` – práve spracovávaný *pohľad* z `config_view.php` (kapitola 3.4.3).
- `$data` – pole s dátami, ktoré sa majú pri pohľade zobraziť. Prvý rozmer poľa je poradové číslo zobrazovaného záznamu. Druhý argument je názov stĺpca tabuľky v databáze.
- `$num_records` – celkový počet záznamov, ktoré sa majú zobraziť.
- `$count_pages` – počet stránok pri listovaní.
- `$act_page` – aktuálna stránka v rámci listovania záznamov.
- `$idview` – je poradie pohľadu v rámci práve zobrazeného pohľadu. Jeho využitie spočíva v tom, že na základe tejto premennej dokáže systém zobrazovať a spracovávať viac pohľadov naraz. Použitie viacerých pohľadov je bežné. Napríklad zobrazovanie noviniek v jednej časti front-endu a listovanie produktov v druhej.

Ďalšiu vec, ktorú musí osoba poverená konfiguráciou vyriešiť je správne nastavenie hypertextových odkazov, aby mohlo dôjsť ku korektnému ovládaniu v rámci *pohľadu*. Odkazy obsahujú čísla pri listovaní, šípky pri radení záznamov a podobne. Odkazy sa definujú nasledovne:

```
<a href="<?=$url?>?<?=KEY_URL_...?><?=$idview?>=<?=$hodnota?>">...
```

KEY_URL_... je jedna konštánt s prefixom KEY_URL_, konkrétne môže ísť o:

- KEY_URL_ORDER – identifikátor stĺpca, podľa ktorého sa radí. Identifikátor je poradie *elementu*, ktorý predstavuje daný stĺpec.
- KEY_URL_DIRECTION – smer radenia. Nadobúda hodnoty DIRECTION_ASC alebo DIRECTION_DESC.
- KEY_URL_LIST – číslo stránky, ktoré sa má po kliknutí na odkaz zobraziť.
- KEY_URL_SEARCH – hodnotou je kľúčové slovo, ktoré sa má vyhľadávať.

Pri tvorbe šablón je rozumné vychádzať zo šablón v `/template/view`.

3.6. Inštalácia: 6. Front-end a jeho modifikácia

Rozdelenie šablón layoutu do `header.php`, `before_content.php` a `footer.php` nie je určujúce a je možné toto rozdelenie ľahko zmeniť. Tu je však už potrebný zásah do kódu. Konkrétne do `/index.php`, čo je spúšťacia stránka front-endu. Hlavná časť tohto skriptu po kóde s načítaním potrebných súborov vyzerá takto:

```
// naviazanie spojenia s databazou
$conn = new class_Db();
if (! $conn->connect() )  { echo ERROR_DB_CONNECT; die(); }

//ziskam data pre menu a aktualnu stranku
$front_handle = new class_FrontEnd($conn, TABLE_MENU, TABLE_CONTENT );
$menu = $front_handle->genMenu();
$page = $front_handle->getContent();

//nastavim aktualnu adresu skriptu
$url = basename( $_SERVER['PHP_SELF'] );

//nacitam sablony
include(TEMPLATE_DIR.'front_end/header.php');
include(TEMPLATE_DIR.'front_end/menu.php');
include(TEMPLATE_DIR.'front_end/before_content.php');

//zobrazim web content
$front_handle->visualTagReplace($page['htmlcode'],$visual_tag_arr);

include(TEMPLATE_DIR.'front_end/footer.php');
```

Z toho je zrejmé, že zmena šablón layoutu je len otázkou modifikácií niekoľkých riadkov. Konkrétne tých, ktoré pomocou príkazu `include` načítavajú obsah šablón.

3.7. Návod ako pridať formulár

Táto kapitola obsahuje konkrétny návod, ako vytvoriť a nakonfigurovať konkrétny formulár. Ako príklad je uvedený formulár pre správu kontaktov. Ten sa nachádza aj na priloženom CD v adresári `/install/example1/`, ktorý obsahuje konfiguračné súbory a SQL skripty.

- Prvým krokom je tvorba tabuľky v databáze. Ide o tabuľku, ktorá obsahuje primárny kľúč id, meno, adresu, poznámku či je kontakt VIP, kategória do ktorej osoba patrí a dátum narodenia.

```
CREATE TABLE `contact` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(255) default NULL,
  `address` varchar(255) default NULL,
  `vip` varchar(255) default NULL,
  `category` varchar(255) default NULL,
  `birthday` varchar(10) default NULL,
  PRIMARY KEY  (`id`)
)
```

- Druhým krokom je nastavenie príslušných práv. Rozhodneme sa pre nastavenie plných práv pre skupinu *manager* (ktorá má identifikátor id číslo 1) Plné práva predstavuje maska 31. Masky boli vysvetlené v kapitole 3.3.1

```
INSERT INTO `rights` VALUES (1, 'contact', 31);
```

- Tretím krokom je pridanie mapy elementov do súboru `config_elements.php`, ktoré popisujú formulárové prvky kontaktov. Ten vyzerá nasledovne:

```
   $contact[] = array(
        FMAP_HTML_LABEL           => 'Name',
        FMAP_HTML_TYPE            => HTML_INPUT,
        FMAP_HTML_NAME            => 'name',
        FMAP_HTML_INPUT_SIZE      => 25,
        FMAP_SQL_COL              => 'name',
        FMAP_SQL_APOSTROPHE       => true,
        FMAP_SQL_DUTY             => true,
        FMAP_SQL_REGEXP           => REGEXP_STRING,
        FMAP_SQL_REGEXP_IF_ERROR   => 'Zadana hodnota Name musi obsahovat
aspon tri znaky',
        );

   $contact[] = array(
        FMAP_HTML_LABEL           => 'Address',
        FMAP_HTML_TYPE            => HTML_INPUT,
        FMAP_HTML_NAME            => 'address',
        FMAP_HTML_INPUT_SIZE      => 50,
        FMAP_SQL_COL              => 'address',
        FMAP_SQL_APOSTROPHE       => true,
        FMAP_SQL_DUTY             => true,
        FMAP_SQL_REGEXP           => REGEXP_STRING,
        FMAP_SQL_REGEXP_IF_ERROR=> 'Zadana hodnota Adress musi obsahovat
aspon tri znaky'
        );

   $contact[] = array(
        FMAP_HTML_LABEL           => 'VIP',
        FMAP_HTML_TYPE            => HTML_CHECKBOX,
        FMAP_HTML_NAME            => 'vip',
        FMAP_HTML_VALUE           => 'VIP',
        FMAP_SQL_COL              => 'vip',
        FMAP_SQL_DUTY             => false,
        FMAP_SQL_APOSTROPHE       => true,
        FMAP_SQL_DUTY             => false,
        );

   $contact[] = array(
        FMAP_HTML_LABEL           => 'Category',
        FMAP_HTML_TYPE            => HTML_SELECT,
        FMAP_HTML_NAME            => 'category',
        FMAP_REFERENCE            => &$enum_category,
        FMAP_SQL_COL              => 'category',
        FMAP_SQL_APOSTROPHE       => true,
        );

   $contact[] = array(
        FMAP_HTML_LABEL           => 'Birthday',
        FMAP_HTML_TYPE            => HTML_DATE,
        FMAP_HTML_NAME            => 'birthday',
        FMAP_HTML_INPUT_SIZE      => 10,
        FMAP_SQL_COL              => 'birthday',
        FMAP_SQL_APOSTROPHE       => true,
        FMAP_SQL_DUTY             => false,
        );
```

- Predposledným krokom je nastavenie pohľadov pre kontakty. Dovedna pôjde o tri pohľady. Pohľad pre prezeranie kontaktov, pohľad pre detail kontaktu a pohľad pre export. Nastavenie pohľadov znamená pridanie polí do `config_view.php`.

```
$view_contact = array (
     VIEW_COLS           =>           array(0,1,2,3,4),
     VIEW_PER_PAGE       =>           15,

     VIEW_IS_ORDER       =>           true,
     VIEW_IS_LISTING     =>           true,
     VIEW_IS_SEARCH      =>           true,
     VIEW_IS_FILTER      =>           true,

     VIEW_SEARCH_COLS    =>           array (0,1,2),
     VIEW_FILTER_COLS    =>           array (0,2,3,4),

     VIEW_START_ORDER          =>           1,
     VIEW_START_DIRECTION      =>           DIRECTION_ASC,
     VIEW_START_LISTING        =>           0,

     VIEW_TPL_SEARCH=>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'search.php',
     VIEW_TPL_ORDER=>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'order.php',
     VIEW_TPL_BODY=>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'body.php',
     VIEW_TPL_HEAD=>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'head.php',
     VIEW_TPL_LISTING=>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'list.php',
);

$view_contact_detail = array (
     VIEW_COLS           =>           array(0,1,2,3,4),
     VIEW_PER_PAGE       =>           1,

     VIEW_IS_ORDER       =>           false,
     VIEW_IS_LISTING     =>           false,
     VIEW_IS_SEARCH      =>           false,
     VIEW_IS_FILTER      =>           false,

     VIEW_SEARCH_COLS    =>           array (0,1,2),
     VIEW_FILTER_COLS    =>           array (0,2,3,4),

     VIEW_START_ORDER          =>           1,
     VIEW_START_DIRECTION      =>           DIRECTION_ASC,
     VIEW_START_LISTING        =>           0,

     VIEW_TPL_HEAD=>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'dtl_head.php',
     VIEW_TPL_BODY=>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'dtl_body.php',
);


$view_contact_export = array (
     VIEW_COLS           =>           array(0,1,2,3,4),
     VIEW_PER_PAGE       =>           20,

     VIEW_IS_ORDER       =>           true,
     VIEW_IS_LISTING     =>           true,
     VIEW_IS_SEARCH      =>           true,
     VIEW_IS_FILTER      =>           true,

     VIEW_SEARCH_COLS    =>           array (0,1,2),
     VIEW_FILTER_COLS    =>           array (0,2,3,4),

     VIEW_START_ORDER          =>           1,
     VIEW_START_DIRECTION      =>           DIRECTION_ASC,
     VIEW_START_LISTING        =>           0,

     VIEW_TPL_SEARCH=>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'search.php',
     VIEW_TPL_ORDER=>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'order.php',
     VIEW_TPL_BODY=>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'body.php',
     VIEW_TPL_HEAD=>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'head.php',
     VIEW_TPL_LISTING=>'../'.TEMPLATE_DIR.VIEW_SUB_DIR.'list.php',
```

```
        VIEW_PDF_COLS_WIDTH => array(30,60,20,40,20),
        VIEW_PDF_HEADER_LOGO => '../pic/pdf_logo_makaga.jpg',
        VIEW_PDF_HEADER_TEXT => 'Contacts',
        VIEW_PDF_FONT_SIZE => 6,
        VIEW_PDF_LINE_SIZE => 5,
);
```

- Posledným krokom je vytvorenie konfiguračnej mapy formulára. Mapy pre formulár sa nachádzajú v `config_forms.php`.

```
$form[2] = array (
        FORM_LABEL => 'CONTACT',
        FORM_TABLE => 'contact',
        FORM_GRAPH => array(2,3),
        FORM_TYPE  => FORM_TYPE_NORMAL,
        FORM_ELEMENTS => &$contact,
        FORM_VIEW_LIST => &$view_contact,
        FORM_VIEW_DETAIL => &$view_contact_detail,
        FORM_VIEW_EXPORT => &$view_contact_export,
        );
```

Toto bol obecný príklad toho, ako vytvoriť formulár. Predstavme si ďalší formulár o autách. Ten by obsahoval značku auta, SPZ a osobu vlastníka. (kompletný príklad na priloženom CD v adresári `/install/example1/`). Element vlastník by mal nasledujúce vlastnosti:

```
$car[] = array(
        FMAP_HTML_LABEL             => 'Owner',
        FMAP_HTML_TYPE              => HTML_SELECT,
        FMAP_DEPENDENCY_TYPE        => DEPENDENCY_TYPE_1_TO_N,
        FMAP_TARGET_MAP             => &$contact,
        FMAP_TARGET_FORM_ID         => 2,
        FMAP_TARGET_TABLE           => 'contact',
        FMAP_TARGET_COLS            => array(0,3),
        FMAP_TARGET_SEPARATOR       => ', ',

        FMAP_HTML_NAME              => 'owner',
        FMAP_SQL_COL                => 'owner',
        FMAP_SQL_APOSTROPHE         => false,
        );
```

Ďalej by sme mali nastavené mapy formulárov a pohľadov podobne ako v príklade s kontaktmi. Cieľom ďalších riadkov bude ukázať, ako v pohľade vytvoriť automaticky hypertextový odkaz, ktorý by zo stĺpca *owner* ukazoval priamo na náhľad danej osoby.

Riešenie sa týka iba nepatrnej zmeny šablóny `template/view/body.php`.

```
<?if (isset($elements[$col_id][FMAP_DEPENDENCY_TYPE]) &&
$elements[$col_id][FMAP_DEPENDENCY_TYPE] == DEPENDENCY_TYPE_1_TO_N) :?>
<a href="<?=$url?>?
<?=KEY_URL_FORM?>=<?=$elements[$col_id][FMAP_TARGET_FORM_ID]?>&
<?=KEY_URL_ACTION?>=<?=ACT_SHOW_RECORD?>&
<?=KEY_URL_RECORD?>=<?=$data[$j][PREFIX_ID.$dkey]?>">
<?=$cell?>
</a>

   <?else: ?>
   <?=$cell?>
   <?endif;?>
```

Pôvodná časť `<?=$cell?>` bola nahradená blokom kódu hore. V pôvodnej šablóne sa len zobrazoval text bunky tabuľky, v novej už bude hypertextový odkaz. V kóde pribudla

podmienka, ktorá zisťuje, či išlo o položku so závislosťou. Ak áno, vytvoril sa hypertextový odkaz, ktorý nastavuje tri premenné zvonku. A síce *formulár* kontaktov, *akciu* zobraz detail a konkrétny záznam *záznam*.

4. Programátorská časť

4.1. Rozdelenie do logických vrstiev

Ako sa Makaga ovláda, inštaluje a konfiguruje bolo naznačené v kapitolách 2 a 3. Táto kapitola popisuje, ako systém funguje.

Funkčnosť systému možno rozdeliť do nasledujúcich *logických celkov:*

- Databázové rozhranie
- Prihlásenie a práva
- Správa akcií
- Generovanie formulárov
- Správa pohľadov
- Správa front-endu
- Šablóny

4.1.1. Databázové rozhranie

Významný celok, ktorý zaisťuje komunikáciu s databázou. Obsahuje dve vrstvy:

- Nižšia vrstva má na starosti vykonávanie samotných SQL dotazov. Tie sú predávané tejto vrstve vo forme reťazcov.
- Vyššia vrstva zaisťuje generovanie SQL dotazov pre nižšiu vrstvu. Vstupom tejto vrstvy sú špeciálne polia, ktoré popisujú jednotlivé časti SQL dotazu. Ide o polia, ktoré obsahujú:
    - názvy stĺpcov a ich aliasov
    - názvy tabuliek
    - štruktúru popisujúcu podmienku WHERE
    - štruktúru popisujúcu informácie o radení záznamov
    - informácie o GROUP BY príkaze

O detailoch týchto polí sa čitateľ môže dozvedieť v technickej dokumentácii na CD v adresari `install/documentation/`.

Dôvodom zavedenia dvoch vrstiev je hneď niekoľko. Prvým dôvodom je zjednodušenie práce programátorom. Všetky časti programu, ktoré potrebujú vybrať dáta z databáze, generujú len polia s dátami, čo je veľakrát jednoduchšie ako skladať samotné SQL dotazy. Samozrejmá je aj jednoduchá znovu použiteľnosť polí. Ďalším dôvodom je to, že ak by bola požiadavka na zmenu databázového servera na taký, ktorý má odlišnú syntax, nebude potrebné modifikovať program ako celok, ale len spomínané vrstvy databáze. V prípade rovnakej syntaxe stačí zmeniť nižšiu vrstvu. Posledným dôvodom je zvýšená bezpečnosť proti SQL injection, lebo reťazce potencionálnej hrozby sú tak sústredené v jedinej štruktúre.

4.1.2. Prihlásenie a práva

Ide o celok, ktorý má na starosti prihlasovanie a odhlasovanie užívateľov. Obsahuje triedy, ktoré majú za úlohu zaisťovať autorizáciu a autentizáciu. Princíp fungovania je jednoduchý. Programu sa po vyplnení logovacieho formulára dostanú premenné zvonku, ktoré obsahujú prihlasovacie meno a heslo. Ďalej dôjde ku napojeniu na databázu. Zistí sa, či daný užívateľ existuje a detekuje sa zhoda zahashovaného hesla so zahashovaným heslom v databáze. Nakoniec sa nastavia užívateľské údaje a práva do super globálnych premenných session.

4.1.3. Správa akcií

Jedná sa o celok, ktorý má na starosti prijímať požiadavky na *akcie* od užívateľa, vyhodnocovať ich a spracovávať. Požiadavka na *akciu* znamená:

- užívateľ chce vložiť nový záznam
- užívateľ chce prezerať záznamy
- užívateľ chce prezerať detaily záznamov
- užívateľ chce modifikovať záznam
- užívateľ chce vymazať záznam
- užívateľ chce zobraziť koláčové grafy
- užívateľ chce exportovať dáta
- užívateľ chce zobraziť Menu Handler

Program spozná že došlo ku požiadavke na *akciu* tak, že zaznamená zmenu *premennej zvonku*. *Premenné zvonku* dostáva PHP skript z URL adresy (alebo z formulára odoslaním cez metódu GET). Nech sa v adrese prehliadača nachádza takáto adresa: *www.makaga.cz/index.php?meno=jozef&vaha=88*. Potom sú skriptu `index.php` predané *premenné zvonku* `$_GET[meno]` a `$_GET[vaha]` s hodnotami `jozef` a `88`. To znamená, že úložiskom premenných zvonku je pole `$_GET`. Ďalšou možnosťou, ako inicializovať *premennú zvonku* je odoslanie formulára metódou POST. Vtedy je úložiskom týchto premenných pole `$_POST`. Metódy POST a GET sa líšia tým, že metóda GET prenáša dáta v adrese prehliadača (ako bolo naznačené). Technicky dôjde ku nastaveniu *premenných zvonku,* ktoré sú určené na *správu akcií* tým, že užívateľ klikne na hypertextový odkaz v rámci systému Makaga.

V ďalšom kroku sa zistí, či má prihlásený užívateľ príslušné práva k danej *akcií*. V prípade že nie, musí dôjsť ku upozorneniu užívateľa. Pozornému čitateľovi neunikne, že *akcií* je viac ako práv. Jedno právo sa môže vzťahovať k viacerým *akciám*. Napríklad právo prezerať detaily záznamov obsahuje aj zobrazenie koláčových grafov.

S *akciou* súvisia však aj pomocné informácie. Ide napríklad o *formuláre*, s ktorými akcia súvisí. Napríklad pri vložení nového záznamu musí systém správne určiť ktorý formulár sa má zobraziť. Ďalej je veľakrát potrebné identifikovať konkrétny záznam v rámci formulára. Napríklad ak ide o mazanie či editovanie záznamov. Tieto pomocné dáta musí správa akcií správne rozlišovať a ukladať.

Posledným krokom správy je volanie tried podľa toho, ku akej *akcií* došlo. To znamená, že správa ma zaistiť volanie tried, ktoré zobrazujú formuláre či *pohľady*.

Správa obsahu má na starosti aj zobrazenie hlavičky, menu back-endu, systémových hlásení a pätičku.

4.1.4. Generovanie formulárov

Samotné generovanie formulárov prebieha na prvý pohľad pomerne jednoduchým spôsobom. Na základe zvoleného *formulára* si systém vezme mapu *elementov* z `config_elements.php` (viac kapitola 3.4.2). Program prechádza konfiguračné pole a na základe dát v ňom uložených vykreslí šablóny prázdnych elementov, ktoré obsahujú jednotlivé elementy.

V prípade editácie záznamov sa najprv vyberie príslušný záznam z databáze a potom sa v cykle, podobne ako v predošlom prípade, volajú šablóny elementov.

Po odoslaní formulára systém znovu použije mapu *elementov,* za pomoci ktorej kontroluje, či sú povinné položky vyplnené správne. V prípade že nie, tak sa zobrazí vyplnený formulár s odoslanými dátami a chybovými hláseniami. Ak sú položky vyplnené správne, dôjde

k uloženiu dát do databáze. Ako vygenerovať príslušné polia pre SQL dotaz vie systém opäť na základe mapy *elementov*.

4.1.5. Pohľady

Konfigurácia pohľadov sa nachádza v súbore `config_view.php`, ktorý bol predstavený v kapitole 3.4.3. Ak dôjde ku akcií, ktorá má zobraziť záznamy, správa akcií zariadi, aby sa zobrazil príslušný *pohľad*. Zobrazenie pohľadu prebieha v týchto krokoch:

- Najprv musí systém zistiť, ktorú stránku pohľadu má zobraziť, podľa ktorého stĺpca sa má radiť, ktorým smerom,… Takže v prvom kroku sa pozrie do *premenných zvonku* a zistí, či nedošlo ku zmene aktuálnej stránky, zmene stĺpca podľa ktorého sa radí, zmene smeru radenia, zmene filtrovacích kritérií, zmene vyhľadávaného slova. Ide o tzv. *stavy pohľadu*.
- Ďalší krok spočíva v tom, že na základe *stavov pohľadu* musí systém správne vygenerovať SQL dotaz, ktorý potrebné dáta vytiahne z databázy. V tomto kroku sú využívané aj dáta z mapy *elementov* príslušného formulára.
- V poslednom kroku dôjde ku vykresleniu tých šablón, ktoré boli v pohľade definované.

4.1.6. Správa front-endu

Princíp fungovania front-endu už bol naznačený v kapitole 2.11. Správa front-endu prebieha v týchto krokoch:

- Systém načíta dáta z tabuľky Menu Handlera a pripraví tieto dáta pre šablónu, ktorá menu vo front-ende zobrazuje.
- Systém zistí na základe určitej *premennej zvonku* identifikátor aktuálnej webovej stránky. Ten sa mení vždy po kliknutí na nejakú položku menu.
- Na základe identifikátora stránky sa z databáze vytiahne príslušný záznam Web Contentu.
- Posledný krok spočíva v zobrazení Web Contentu a náhradou tagov `visual` za *pohľady*. To sa deje tak, že text sa najprv rozparsuje podľa regulárneho výrazu určeného tagom `visual`. Zobrazí sa časť HTML kódu pred tagom. Následne sa zistí aký *pohľad* sa má zobraziť (podľa atribútu `tpl`). V cykle opäť dôjde ku zobrazeniu časti HTML kódu pred ďalším tagom `visual`, atď.

4.1.7. Šablóny

Aj šablóny už boli spomenuté (kapitola 3.5). Šlo však o šablóny *pohľadov* a back-endu. Mimo ne existuje viacero typov šablón, ktoré sa načítavajú len pri chode back-endu:

- **Šablóny HTML elementov formulára**. Tieto šablóny sú volané pri generovaní formulárov. Pre každý typ HTML elementu formulára existuje jedna šablóna pre prázdny element a jedna pre predvyplnený element. V systéme Makaga sa šablóny HTML formulárových elementov nachádzajú v troch sadách. Ide o súbory s rovnakými názvami a funkciou. Líšia sa len v tom, ako sú samotné elementy naštýlované a napozícované. Sady sa nachádzajú v nasledujúcich adresároch:
  - `template/form_in_table/` - je to sada šablón, ktorá sa zobrazuje pri formulároch pre vytváranie a editáciu záznamov.
  - `template/form_in_filter/` - šablóny v tejto sade sa zobrazujú ako filtrovací formulár.

- `template/form_in_row/` - táto sada šablón sa zobrazuje pri type elementu HTML_M_TO_N.

- **Systémové šablóny**. Tie sa zobrazujú v back-ende. Nachádzajú sa v adresári `template/system/` Ich výčet a úlohy sú tieto:
  - `backend_head.php` – šablóna, ktorá predstavuje hlavičku back-endu
  - `backend_foot.php` – šablóna pätičky back-endu
  - `menu.php` – šablóna, ktorá vykreslí navigačné menu back-endu
  - `error.php` – šablóna, ktorá zobrazuje chybové hlásenie v back-ende
  - `success.php` – šablóna, ktorá vypisuje hlásenia o úspechu prevedenia akcie
  - `no_permission.php` – šablóna, ktorá sa zobrazí, ak užívateľ chce vykonať akciu, na ktorú nemá právo
  - `index_head.php` – šablóna s hlavičkou úvodnej prihlasovacej stránky back-endu
  - `login.php` - šablóna tela úvodnej prihlasovacej stránky back-endu

4.2. Triedy

4.2.1. Popis tried

V tejto časti budú popísané triedy. Triedy sú najvyššie *funkčné celky*. Pochopenie ich významu je nevyhnutné pre pochopenie fungovanie celé systému. Nasledujúca časť popisuje, čo je funkciou jednotlivých tried a čo sa ich modifikáciou dá docieliť.

Class_State

Táto trieda patrí medzi často inštanciované a jej hlavnou úlohou je zachovávať a určovať stavy aplikácie. Jej základ fungovania spočíva v tom, že jedna inštancia uchováva jeden stav nepretržite. K tomu sú využívané superglobálne premenné, tzv. sessions. Tie na rozdiel od klasických premenných udržiavajú hodnotu "neustále" aj po ukončení skriptu. Pri opätovnom spustení ostane premenná stále inicializovaná s rovnakou hodnotou.

Konštruktor dostane na vstup zoznam možných stavov, kľúč poľa premenných zvonku a kľúč poľa session.

K zmene stavu, ktorý sa nachádza v session, dôjde vtedy, ak existuje premenná zvonku s príslušným kľúčom a zároveň je hodnota jednou z možných stavov.

Inštancie tejto triedy sa používajú napríklad na zachovávanie stavu popisujúceho aktuálnu stránku pri listovaní, stĺpec radenia, smer radenia, kľúčové slovo, aktuálny *formulár*, aktuálnu stránku vo front-ende, akciu,...

Class_Mysql

Predstavuje nižšiu vrstvu databázového rozhrania (kapitola 4.1.1). Hlavným cieľom tejto triedy je tvorenie spojenia s databázou MySQL a jej základná obsluha. To znamená vykonávanie SQL dotazov, ktoré sú predávané v reťazcoch, vrátenie počtu záznamov po použití príkazu SELECT, posledný vložený kľúč a podobne.

Modifikáciou je možné zmeniť databázu z MySQL na nejakú inú. Napríklad na PostgreSQL. K tomu by došlo tak, že sa príkazy na obsluhu MySQL databáze by sa zmenili na ekvivalenty obsluhujúce databázu PostgreSQL.

Class_Db

Predstavuje vyššiu vrstvu databázového rozhrania (kapitola 4.1.1). Dedí z triedy Class_MySQL. Jej úlohou je generovať SQL dotazy na základe špeciálnych polí, ktoré popisujú časti SQL dotazu. Tie sú predávané metódam v argumentoch.

Rozšírením tejto triedy je možno docieliť toho, že program dokáže spracovávať náročnejšie SQL dotazy. V tejto podobe systému to potrebné nie je, ale pri rozšírení by sa to mohlo hodiť. Mohlo by ísť povedzme o podporu vnorených SELECT príkazov (podporuje až MySQL verzie 4.0)

Class_Form

Je nosnou triedou logického celku *generovania formulárov* popísaného v kapitole 4.1.4. Všetky informácie o tom, ako má formulár vyzerať získava systém z mapovacieho poľa z `config_elements.php` (kapitola 3.4.2). Metódy tejto triedy dokážu vygenerovať základné typy formulárových položiek, plus niektoré zložené.

Okrem generovania prázdnych, či vyplnených formulárov táto trieda zisťuje platnosť dát po odoslaní formulára a tieto dáta ukladá do databázy. Okrem iného obsahuje metódy na uploadovanie súborov a tvorbu náhľadových obrázkov.

Modifikáciou tejto triedy sa dá docieliť pridanie ďalších typov formulárových položiek. Vezmime si príklad, kde by sme chceli vytvoriť nový typ formulárovej položky, v ktorej by si užívateľ mal vyberať farbu. Šlo by o typ podobný HTML_DATE, kde sa ale miesto okna pre výber dátumu zobrazilo okno s paletou, ako ho poznáme z grafických editorov. Kliknutím myšou na farbu by sa do input text boxu vygeneroval kód farby.

Tvorba takéhoto typu formulárového prvku by pozostávala z nasledujúcich krokov:

- Prvým krokom by bolo vytvorenie novej konštanty pre daný typ – HTML_COLOR. Tá by sa mala nachádzať v `config/sysconst.php`.
- Druhým krokom by mala byť tvorba nových šablón pre tento typ elementu a ich začlenenie do sád (kapitola 4.1.7).
- Posledným tretím krokom by mala byť modifikácia tejto triedy. Šlo by o doplnenie metód *showEmptyElement* a *showFilledElement*, kde by sa doplnilo odchytenie nového typu formulárového prvku a načítanie príslušných šablón z druhého kroku. Ak by malo pre daný typ dochádzať ku špeciálnej validácii odoslanej hodnoty tohto formulárového prvku, je potrebné modifikovať aj metódu *isValidElementData*. Kde sa opäť odchytí daný typ prvku a časť kódu s podmienkami a nastavením chybového hlásenia, ak dáta podmienkami neprejdú.

Class_View

Ide o nosnú triedu logického celku *Pohľady* (viac kapitola 4.1.5). Trieda Class_View má na starosti obsluhu *pohľadov*. Ako bolo spomenuté, *pohľad* je kolekciou grafických a ovládacích prvkov, ktoré prezentujú dáta. Konfigurácia pohľadu je zaznamenaná v mapovacom poli elementov (detaily kapitola 3.4.3).

Na správu stavov systém využíva inštancie triedy Class_State (aktuálna stránka, stĺpec podľa ktorého sa práve listuje, smer radenia, filter, ...).

Funkčnosť tejto triedy možno zhrnúť do týchto častí:

- Správa stavov. Úlohou je zistiť správne načítanie stavov podľa toho, ktoré časti zobrazovania pri pohľade sú nastavené (filter, radenie, listovanie, ...).
- Generovanie polí pre SQL dotazy. Tie sa generujú na základe vnútorných stavov.
- Načítanie šablón. To predstavuje už samotné zobrazovanie.

Class_ActionHandler

Ide o hlavnú časť správy akcií (kapitola 4.1.3). Spúšťací skript administračnej časti Makagy komunikuje výhradne s touto triedou. Obsahuje tri inštancie Class_State. Jedna sníma akcie, druhá tabuľku a tretia konkrétny záznam.

Podľa zvolenej akcie a tabuľky sa zobrazujú formuláre a pohľady. Okrem týchto hlavných úloh zobrazuje systémové šablóny, ktoré obsahujú administračné menu, exporty do CSV, XML a PDF, či generujú grafy.

Modifikáciou tejto triedy možno vytvoriť ďalšie akcie. Povedzme, že chceme vytvoriť akciu tvorbu stĺpcových grafov. Tvorba tejto akcie by pozostávala z týchto krokov:

- Vytvorenie konštanty pre novú akciu.
- Modifikácia šablón navigačného menu. Čiže "pridanie novej ikonky" so správnym odkazom.
- Odchytenie nového typu akcie v metóde *handle.*
- Ďalší krok spočíva v tvorbe metódy, ktorá dokáže túto *akciu* vykonávať. Pri zložitejších typoch *akciách* je možno vytvoriť samostatnú triedu, ktorá by obsahovala všetky potrebné funkcionality pre danú *akciu.*
- Príprava šablón pre *akciu* samotnú.

Class_MenuHandler

Je časťou správy akcií (kapitola 4.1.3). Ide o časť systému na generovanie menu a priradzovanie obsahu. V prvom rade musí byť nastavená tabuľka Web Contentu s príslušnými mapami elementov a pohľadov. Táto trieda obsahuje metódy ktoré dokážu generovať menu ľubovoľnej hĺbky, vytvárať potrebné mapovacie polia elementov a následne cez Class_Form tieto dáta ukladať.

Ku každej bunke menu je možno priradiť obsah. Ďalej je možno priradiť prioritu (čím nižšie číslo, tým je položka v podmenu vyššie) a popis.

Ďalej dokáže vygenerovať vnorenú štruktúru polí, ktorá reprezentuje strom menu. Využíva sa to hlavne kvôli šablónam, ktoré menu zobrazujú.

Modifikáciou možno meniť zobrazenie, prípadne pridať nové položky k bunke menu. Napríklad autor, dátum, ... Ide ale skôr o modifikáciu máp ako zásah do kódov.

Class_Login

Keďže v požiadavkách stojí potreba práv užívateľov systému na určité akcie a tabuľku, bolo potrebné vytvoriť triedu, ktorá tieto dokáže tieto práva manažovať. Trieda jednak zaisťuje zobrazenie prihlasovacích okien a jednak potrebuje zistiť, či má užívateľ právo na jednotlivé *akcie* a *formuláre*. Podľa toho zobrazí príslušné akcie a dovolí akcie vykonať. V opačnom prípade musí naopak vykonanie *akcie* zamedziť.

Modifikáciou je možno pridať ďalšie overovacie či bezpečnostné prvky. Napríklad pri formulári generovať obrázok, ktorý obsahuje číselný token. Číslo z obrázku by užívateľ musel opísať do text boxu. Tým by bola zvýšená bezpečnosť voči útokom hrubou silou.

Class_Timer

Class_Timer je triedou úzko spätou s Class_Timer. Jej hlavou úlohou je zaisťovať automatické odhlásenie užívateľa, ak po určitú dobu nevykonal žiadnu akciu.

Class_Dependency

Systém sa musí vedieť vyrovnať aj so závislosťami v tabuľkách. Jak vo formulároch, tak v pohľadoch. Class_Dependency slúži práve na modifikáciu a tvorbu máp elementov. Trieda musí zaistiť:

- Zmenu tých formulárových elementov, ktoré nesú odkazy na prepojovacie tabuľky. Príklad: Majme tabuľku *zamestnancov* a tabuľku *firemných automobilov*. V mape elementov u firemných automobilov sa nachádza jedna položka *vodič,* ktorá hovorí o tom, že je typu HTML_SELECT. Problém je, že reálne sa v tabuľke na pozícii vodiča nachádza cudzí kľúč na prepojovaciu tabuľku. Preto je potrebné tento element

doplniť o všetkých zamestnancov, ktorý sa nachádzajú v tabuľke. A presne k tomu slúži táto metóda.

- Vytvorenie elementov pre filter. Samotná mapa elementov pre filter vychádza z mapy pre formulár. Ale opäť je potrebné zaistiť doplnenie mapy o závislosti. Niektoré elementy je potrebné zameniť. Napríklad dvojstavový checkbox je potrebné zameniť za trojstavový radio button (dať na výber zaškrtnuté, nezaškrtnuté a všetky), dátum duplikovať a vytvoriť *dátum od* a *dátum do*.

Class_Graph

Táto trieda slúži na generovanie grafov. Využíva knižnice JpGraph [19]. Úlohou triedy je pripraviť dáta pre spomínanú knižnicu, ktorá vygeneruje koláčové grafy pre výčtové typy select box, radio button a checkbox.

Class_PDF

Trieda generujúca PDF dokumenty. K svojej činnosti používa knižnicu FPDF Library [20].

Class_Debug

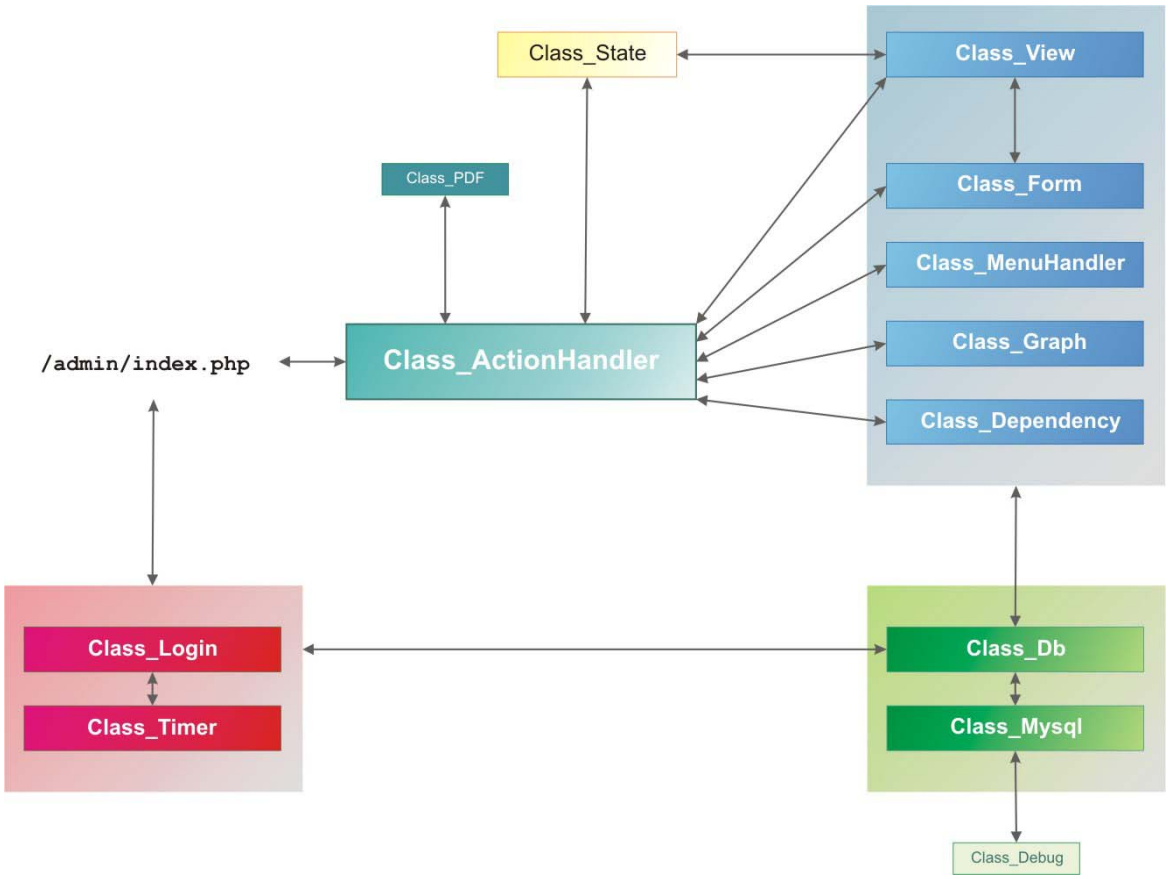
Pomocná trieda, ktorá zobrazuje vykonávané SQL dotazy. Pre správny beh aplikácie sa odporúča vypnúť jej chod (programátorská príručka na CD).

Class_FrontEnd

Táto trieda je volaná pri zobrazení front-endu. Má na starosti uchovávať stav aktuálne zobrazenej stránky inštanciou Class_State. Spracovávať informácie týkajúce sa menu a jeho zobrazenie. A ako posledné vedieť zobraziť obsah stránky. Do obsahu stránky je možno v back-ende vložiť nie len klasické značky HTML, ale aj špeciálny tag `<visual tpl="xxxx">`. Tento tag je nahrádzaný vo front-ende *pohľadom*, na ktorý tento tag odkazuje. Presne týmto spôsobom je možné elegantne dostať informácie z back-endu do front-endu.

4.2.2. Vzťahy tried

Popis vzťahov medzi triedami je posledným krokom k tomu, aby čitateľ získal čo najlepšiu predstavu o fungovaní celého systému. V tejto kapitole je zobrazený systém volaní tried v dvoch prípadoch. V back-ende a front-ende.

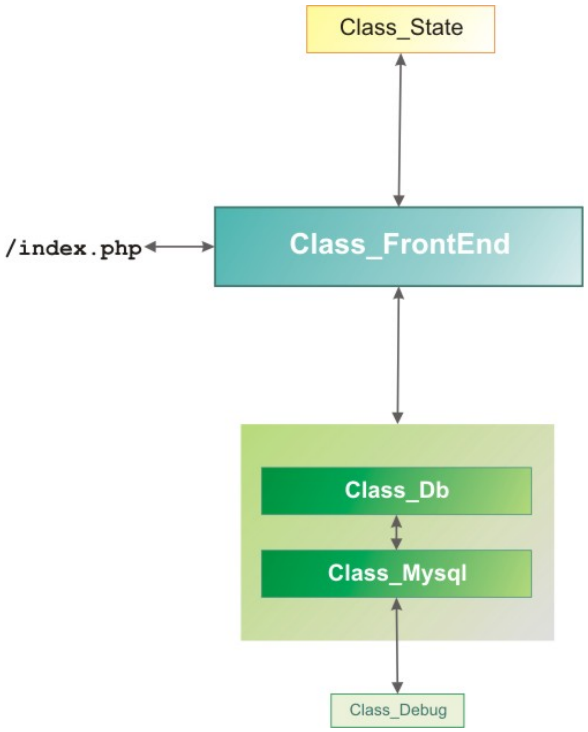


Obr. 17 Vzťahy tried pri back-ende

Na Obr. 177 je naznačené fungovanie back-endu. Cesta k back-endu je `http://domena/admin/index.php`. Skript `index.php` načíta potrebné triedy, konštanty a konfiguračné súbory.

Systém najprv zobrazí prihlasovaciu obrazovku a pomocou triedy Class_Login sa zaisťuje správne prihlásenie užívateľa. Dáta o užívateľovi sa získavajú z databáze pomocou triedy Class_Db.

Po nalogovaní sa v `index.php` vytvorí inštancia triedy Class_ActionHandler. Tá má na starosti zobrazenie šablón back-endu. Class_ActionHandler obsahuje inštancie tried Class_State, ktoré zaznamenávajú aktuálnu *akciu*, *formulár* a záznam. Class_ActionHandler tieto stavy vyhodnocuje a na základe toho generuje formuláre cez Class_Form, zobrazuje záznamy pomocou Class_View, generuje grafy cez Class_Graph, generuje rozhranie pre tvorbu menu cez Class_MenuHandler. V poslednom rade na prekompilovanie mapovacích polí používa Class_Dependency, tak ako to bolo spomenuté pri popise tejto triedy (predošlá kapitola).



Obr. 18 Vzťahy tried pri front-ende

Na Obr. 18 sú naznačené vzťahy tried pri front-ende. `index.php` obsahuje inštanciu triedy Class_FrontEnd. Tá si drží stav aktuálne zobrazovanej stránky. Podľa toho sa volajú jej metódy na zobrazovanie navigačného menu a obsahu stránok za pomoci Class_Db.

4.3. Bezpečnosť

Prístup

Pre prístup do administračného systému je vyžadované zadanie užívateľského mena a hesla. Heslá nie sú uložené v databáze ako plaintext, ale sú zahashované funkciou MD5. Tým je stupeň ochrany vyšší, lebo aj ten kto by sa dostal k zahashovanému heslu, pôvodne heslo nemá možnosť spätne dopočítať. Ďalším bezpečnostným prvkom je, že k samotným prístupovým menám a heslám z administračného systému prístup nie je. Posledným bezpečnostným prvkom je už spomenuté automatické odhlásenie užívateľa po dlhšej nečinnosti. Nakoniec je možno zabezpečiť prístup nastavením súboru `.htaccess` Tam je možno nastaviť prístup len z určitých IP adries.

Sniffing

Sniffing alebo odpočúvanie je technika, kde útočník odchytáva pakety putujúce po sieti a snaží sa z nich vyčítať iným užívateľom zadané mená a heslá, čím dôjde ku kompromitácií.

Rozumnou ochranou voči odpočúvaniu je zabezpečenie administračnej časti tým, že komunikácia medzi klientom a serverom prebieha cez protokol SHTTP.

Cross-site scriptig

Cross-site scriptig [21] je technikou, kde útočník vloží zámerne do objektu ktorý sa zobrazuje všetkým užívateľom (napr. fórum,...) škodlivý kód. Tento kód môže narušiť design stránky, ale aj spôsobiť až ukradnutie session. Proti tomuto útoku by mali byť zabezpečenia implementované v šablónach. Nemusí však ísť o nevyhnutnosť. To z dôvodu, že vkladanie dát prebieha len v administračnom móde, ktoré je zabezpečené heslom a prístup k nemu majú len autorizované osoby. Ďalším dôvodom je, že niekedy je nevyhnutné využívať

prostriedky, ktoré takéto útoky umožňujú. Napríklad pri obsahu stránok na front-ende. V princípe je obrana jednoduchá a to vhodné použitie funkcie `htmlspecialchar()`.

SQL injection

Technika útočníka, v ktorej sa snaží vpašovať podvrhnuté dáta až do SQL dotazu. Deje sa tak väčšinou cez premenné v URL adrese. Ochrana v tomto prípade je rozdelená do viacerých častí kódu. Najvyššie je Class_State, ktorá podľa potreby dokáže akceptovať len hodnoty premenných z určitej množiny. Najnižšie zabezpečenie sa nachádza v Class_Mysql. Tam je využívaná funkcia `mysql_escape_string()`, ktorá zamedzuje takýto útok. V tomto smere má MySQL výhodu, že dokáže vykonať naraz len jeden SQL dotaz.

Podstrčenie premenných

Obecná technika, ako zamieňať resp. podstrkovať premenné. Na obranu je nevyhnutné každú premennú pred použitím inicializovať. Takisto je potrebné vždy pristupovať k premenným zvonku cez pole `$_GET`, `$_POST`.

Download

Ochrana uploadnutých súborov spočíva v tom, že útočník by nemal možnosť stiahnuť uploadovaný súbor na základe toho, že pozná URL uploadnutého súboru. Riešenie spočíva v tom, že prístup k súborom je zamedzený systémovými prostriedkami a práva čítania má len užívateľ, pod ktorým beží webový server. Potom exportný skript načítava súbory a predáva ďalej užívateľovi. Takto je prístup k súborom kontrolovaný.

5. Záver

Cieľom práce bolo vytvoriť systém správy obsahu, zameraný na správu katalógových položiek. Každá z požiadaviek na systém bola splnená. Systém poskytuje nástroje, ktoré sa stávajú vynikajúcou pomôckou každého správcu webu. Hlavným cieľom je zvýšenie efektivity práce a vyhovieť zákazníkovi. Makaga umožňuje vytvoriť zložitejší webový portál v priebehu niekoľkých hodín. Bez neho by vývoj trval ďaleko dlhšie. Praktický význam práce môžem ako jeho autor potvrdiť tým, že už počas vývoja sa systém zaviedol niekoľko krát do praxe.

Nie je v silách jedinca, aby mohol sám v rozumnom čase vytvoriť systém správy obsahu, ktorý disponuje všetkými vlastnosťami ako konkurenčné riešenia plus kľúčové vlastnosti navyše. Preto som sa snažil systém navrhnúť tak, aby obsahoval práve tie nástroje, ktoré vývoj webového portálu najviac posilnia. Ďalej bolo dôležité aj to, aby kód programu mohol byť ľahko modifikovateľný a aby ho bolo možno ľahko dopĺňať o nové vlastnosti. Spomeniem tie, ktoré by som pri vylepšení systému odporúčal vyvíjať ako prvé:

- Vylepšenie práv tak, aby sa v prípade potreby mohli prideľovať práva až na samotné záznamy.
- Ďalším vylepšením by boli tzv. Wizardi. Šlo by o sled formulárov, ktorých vyplnením by došlo k automatickým vyplneniam konfiguračných súborov a vytvoreniu databázových tabuliek.
- Štatistika prístupov. Koľko ľudí si prezrelo front-end, z akých vyhľadávačov sa ku stránke dostali, cez aké kľúčové slová a podobne.
- Členenie front-endu medzi bežných a prihlásených užívateľov. Ich registrácia.
- Podpora a správa často používaných prvkov webových stránok, ako sú ankety, fóra a príspevky návštevníkov ku článkom.
- Nástroje pre internetový obchod. Nákupný košík, nástroje na prácu s cenami. Následné generovanie faktúr a dodacích listov.

Verím, že Makaga si nájde svoje miesto medzi užívateľmi a že jej vývoj bude naďalej pokračovať.

Použitá literatúra

[1] PHP Documentation Group (2005): „Manuál PHP“, http://www.php.net/manual/cs/

[2] Dave Thomas, David Hunt (2004): „Programming Ruby”, The Pragmatic Bookshelf

[3] Dave Thomas, David Heinemeier Hansson (2005): „Agile Web Development with Rails“, The Pragmatic Bookshelf

[4] Active server pages tutorial: http://www.asptutorial.info/learn/Introduction.asp

[5] The official Microsoft ASP.NET Site: http://asp.net

[6] Java 2 Platform, Enterprise Edition: http://java.sun.com/j2ee/index.jsp

[7] Ryan Asleson, Nathanien T. Schutta(2006): „Foundation of Ajax“, Apress

[8] MySQL AB (2004): „MySQL Manual“, http://dev.mysql.com/doc/mysql/en/index.html

[9] The PostgreSQL Global Development Group (2005): „PostgreSQL 8.1.4. Documentantion“, http://www.postgresql.org/docs/8.1/interactive/index.html

[10] Helen Borrie(2005): „Firebird 1.5 Quick Start Guide“, http://www.firebirdsql.org/pdfmanual/Firebird-1.5-QuickStart.pdf

[11] Eclipse: http://www.eclipse.org/

[12] PHP Eclpise-Plugin: http://sourceforge.net/projects/phpeclipse/

[13] Joomla!: http://www.joomla.org/

[14] magicWebDrive CMS: http://www.magicwebdrive.cz/

[15] Websphere software (2003): „Guide to WebSphere Portal 5.0“, ftp://ftp.software.ibm.com/software/dw/wes/pdf/portal5whitepaper.pdf

[16] phpMyAdmin: http://www.phpmyadmin.net/

[17] Smarty: http://smarty.php.net/

[18] TinyButStron: http://www.tinybutstrong.com/

[19] JpGraph: http://www.aditus.nu/jpgraph/

[20] FPDF Library: http://www.fpdf.org/

[21] Ilia Alshanetsky(2005): „Guide to PHP Security“, php|architect nanobooks