VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

NÁVRH NEWSLETTER SYSTÉMU PRO ELEKTRONICKÝ OBCHOD

DESIGN OF NEWSLETTER SYSTEM FOR E-COMMERCE SOLUTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jan Kundrát
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jan Luhan, Ph.D.
SUPERVISOR

BRNO 2014

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2013/14
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jan Kundrát

Manažerská informatika (6209R021)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských, magisterských a doktorských studijních programů zadává bakalářskou práci s názvem:

Návrh newsletter systému pro elektronický obchod

v anglickém jazyce:

Design of Newsletter System for E-commerce Solution

Pokyny pro vypracování:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Seznam odborné literatury:

GILMORE, W. J. Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály. 3. vyd. Brno: Zoner Press, 2011. 736 s. ISBN 978-80-7413-163-9.
KOSEK, J. XML pro každého: podrobný průvodce. 1. vyd. Praha: Grada, 2000. 163 s. ISBN 80-716-9860-1.
KOTLER, P., V. WONG, J. SAUNDERS a G. ARMSTRONG. Moderní marketing. 4. vyd. Praha: Grada, 2007. 1041 s. ISBN 978-80-247-1545-2.
ÖGGL, B. a M. KOFLER. PHP 5 A MySQL 5. 1. vyd. Brno: Computer Press, 2007. 608 s. ISBN 978-80-251-1813-9.
ROSENBROCK, E. a E. FILSON. Linux, Apache, MySQL a PHP: instalace a konfigurace prostředí pro pokročilé webové aplikace. 1 vyd. Praha: Grada, 2005. 344 s. ISBN 80-247-1260-1.
TRAVIS, B. E. XML a SOAP: progamování serverů BizTalk. Vyd. 1. Praha: Computer Press, 2000. 418 s. ISBN 807226303X.

Vedoucí bakalářské práce: Ing. Jan Luhan, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2013/14.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA PODNIKATELSKÁ

L.S.

doc. RNDr. Bedřich Půža, CSc.
Ředitel ústavu

doc. Ing. et Ing. Stanislav Škapa, Ph.D.
Děkan

V Brně, dne 24. 3. 2014

ABSTRAKT

Tato práce se věnuje návrhu firemního newsletter systému pro rozesílání informačních a nabídkových emailů s nabídkou zboží zákazníkům. Systém bude určen pro velké a střední firmy, které provádějí obchodní činnost a nabízejí rozsáhlý sortiment zboží v internetových obchodech, kde vedou registrované zákazníky, kteří chtějí dostávat nabídkové emaily s aktuálními novinkami v obchodě.

ABSTRACT

The aim of this thesis is to describe a design of company newsletter system for sending information and e-mails with offers to customers. System is designed for large and medium companies which run a business and offer an extensive range of good in their online stores. In the database, which is connected with online stores, are registered many customers who want to receive emails with the latest news and offers in the store.

KLÍČOVÁ SLOVA

Newsletter systém, návrh newsletter systému, návrh informačního systému, systém pro rozesílání novinek, rozesílání emailů s novinkami.

KEYWORDS

Newsletter system, design of newsletter system, system for sending news, sending and creating newsletters, management of emails campaigns.

BIBLIOGRAFICKÁ CITACE

KUNDRÁT, Jan. *Návrh newsletter systému pro elektronický obchod*. Brno, 2014. Vysoké učení technické v Brně, Fakulta podnikatelská. Vedoucí práce Ing. Jan Luhan, Ph.D.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 28. května 2014

……..…

podpis studenta

PODĚKOVÁNÍ

Chtěl bych poděkovat firmě, pro kterou zpracovávám tuto práci, za podklady, veškeré náměty a pomocnou asistenci vedení firmy. Dále bych také rád poděkoval mým rodičům, kteří mi umožnili setkat se s výpočetní technikou a informačními technologiemi, a blízkým, kteří mě při práci podporovali. Děkuji také vedoucímu mé bakalářské práce panu Ing. Janu Luhanovi Ph.D. za vedení práce a veškeré rady.

OBSAH

ÚVOD

V současné době existuje mnoho obchodních řetězců, které obchodují s nejrůznějšími druhy zboží a disponují obrovským sortimentem. Mnoho obchodníků, kteří donedávna nabízeli své služby v obchodních centrech a obchodech umístěných na strategických lokacích nyní vlastní internetové obchody. Tyto internetové obchody běží jako informační systémy a jsou spravovány softwarovými firmami. Internetové obchody některých středních a velkých firem spravuje firma, pro kterou zpracovávám svou bakalářskou práci.

Firma potřebuje navrhnout a kompletně realizovat vlastní newsletter systém – systém, který by rozesílal emaily s nabídkami zákazníkům. Pro tyto účely má nyní funkční modul, který je součástí internetového obchodu. Tento modul však nesplňuje ani základní nároky zákazníků.

Pro navržení takové aplikace bude potřeba se seznámit se serverovým skriptovacím jazykem PHP a některým z frameworků, který využijeme pro naprogramování aplikace, a také s databázovým systémem. Dále je nutné podívat se na to, jak pracují SMTP servery pro odesílání emailů a nastudovat si webové služby, aby bylo možné komunikovat se systémem internetového obchodu.

Ve své práci tedy vytvořím návrh tohoto systému, který pak bude kompletně naprogramován, následně uveden do ostrého provozu a dále rozšiřován podle budoucích požadavků.

VYMEZENÍ PROBLÉMU A CÍLE PRÁCE

Firma, pro kterou zpracovávám bakalářskou práci, navrhuje, vytváří a používá pro realizaci internetových obchodů vlastní systém, který si staví na míru. Na tomto systému běží obchody všech jejich zákazníků.

Systém obsahuje také vestavěný modul, který si firma navrhla pro možnost hromadného odesílání emailů z prostřední administrace obchodu. Tento modul má však velmi omezené možnosti použití, je neohebný a není vhodné na něm dále stavět.

Vzhledem k tomuto problému vznikl ze strany vedení nový požadavek, a tím je potřeba vytvořit zcela nový systém, který bude robustní, naddimenzovaný (připraven až pro stovky tisíců zákazníků registrovaných v obchodech), a poběží jako samostatná webová aplikace.

Cílem práce bude tedy vytvořit návrh aplikace, přinést řešení, jak bude systém fungovat, jaké dílčí části bude mít, co bude nabízet zákazníkovi. Návrh bude potřeba dotáhnout tak daleko, aby bylo možno následně pokračovat kompletním naprogramováním systému a uvedením do ostrého provozu. Bude potřeba navrhnout databázové schéma aplikace, tvorbu emailových kampaní a emailů, správu příjemců a také odeslání navržených emailových zpráv.

Dle specifikace firmy bude systém napsán ve frameworku Symfony2 usnadňujícím tvorbu webových aplikací v programovacím jazyce PHP. Výsledná aplikace pak bude komunikovat prostřednictvím webových služeb s aplikací obchodu a bude z něj načítat seznamy emailových adres skupin registrovaných uživatelů.

Protože si firma přála nezveřejňovat své jméno v této práci, je v celé práci nahrazen její obchodní název zkratkou XYZ, s.r.o.

1 TEORETICKÁ VÝCHODISKA PRÁCE

Před přistoupením k navrhování systému je zapotřebí seznámit se s pojmy, které souvisejí s problematikou, od obecnějších jako je informační systém či elektronická pošta až ke konkrétnějším, zejména pak použité technologie, skriptovací jazyk a framework, na kterém budeme aplikaci stavět, dále pak také databázový systém.

1.1 Informační systém a ICT

Většina dnešních společností využívá při své činnosti informační systém. Informační systém je systém, jehož účelem je zajištění správných informací ve správnou chvíli na správném místě. Místem se rozumí v tomto případě lidé, kteří jsou uživateli informačního systému, a správností je plnění účelu byznys systému [1].

ICT je zkratka pro informační a komunikační technologie. Jsou důležité pro plnění účelu informačního systému. Často používáme zkratku IS/ICT pro informační systém podporovaný informačními a komunikačními technologiemi. ICT jsou hardwarové a softwarové prostředky pro sběr, přenos, ukládání, zpracování a distribuci informací a také pro komunikaci lidí a technologických komponent IS [1].

1.2 E-business, e-commerce, e-marketing

Pojmem E-business se nazývá internetové podnikání. Znamená to využít elektronických platforem pro provádění podnikatelské činnosti. Elektronickými platformami se rozumí internet, intranet, extranet. Veškeré tyto technologie nyní firmám umožňují urychlit podnikatelské aktivity a zároveň také je rozšířit v časovém i prostorovém rozsahu [2].

Mnoho firem se prezentuje na internetu prostřednictvím svých internetových stránek, pomocí nichž propagují své výrobky a služby, nechává si vytvořit informační systém pro vzájemnou komunikaci mezi zaměstnanci a jejich přístup k informacím ve firmě, vybudovaly extranet pro komunikaci s dodavately a distributory [2].

E-commerce znamená internetové obchodování, představuje proces nákupu a prodeje podporovaný elektronickými prostředky. Zahrnuje e-marketing a e-purchasing [2].

E-marketing je marketingová část e-commerce. Je to snaha společnosti informovat o svých výrobcích a službách a nabídnout a propagovat je na internetu [2].

1.3 Elektronická pošta

Elektronická pošta je základní komunikační služba na internetu. Každá firma, každý zaměstnanec v dnešní době zpravidla vlastní jednu či více emailových schránek, do kterých mu přichází nejrůznější zprávy. Email je v současné době velmi rozšířeným nástrojem pro komunikaci, proto jej mohou používat i obchodní firmy pro komunikaci se zákazníkem.

Pro odeslání a doručení emailů adresátovi jsou zapotřebí dva druhy programů – uživatelský poštovní agent (MUA) a přenosový poštovní agent (MTA). MUA se využívá pro čtení a odesílání pošty a MTA pro její doručení – přenos mezi servery. Struktura a cesta jsou podrobně popsány v dokumentech RFC 2821, ve kterém je definován SMTP protokol sloužící pro přenos pošty z jednoho serveru na druhý, a RFC 2822, který pak definuje formát zpráv [3].

1.3.1 Protokoly elektronické pošty

Mezi protokoly elektronické pošty patří POP3, IMAP a SMTP. První z nich POP3 znamená Post Office Protocol 3 – je to protokol pro přijímání pošty, běží na portu 110, slouží pro stahování e-mailů do lokálního počítače uživatele, umožňuje zanechat kopii emailu na serveru. Dále protokol IMAP je zkratkou Internet Mail Access Protocol, což je v překladu protokol přístupu k internetové poště. Tento protokol na rozdíl od protokolu POP3 zajišťuje, aby byly stejné zprávy uloženy na serveru i v lokálním počítači. Stará se tedy o celkovou synchronizaci a jeho implementace je podstatně složitější. Poslední z protokolů je SMTP, kterému bude věnována část dále [4].

1.3.2 SMTP protokol

SMTP protokol je standardem pro přenos elektronické pošty, který vznikl v roce 1982, kdy nahradil tehdejší protokoly, které nebyly vzájemně kompatibilní [5]. SMTP je zkratka pro Simple Mail Transfer Protocol (jednoduchý protokol pro přenos pošty). Používá se k odesílání, předávání a přijímání emailových zpráv na poštovní servery [3].

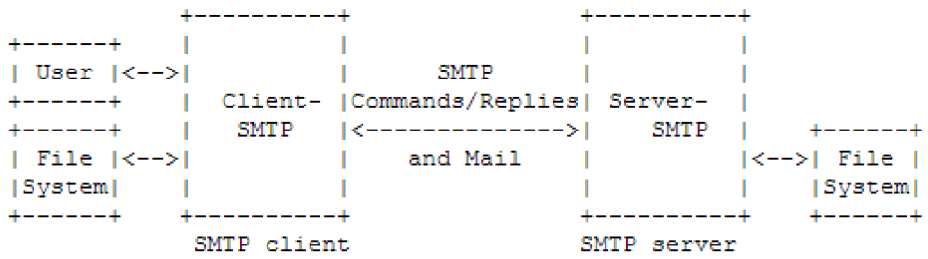
SMTP protokol tedy zajišťuje doručení emailu od odesilatele k příjemci (či více příjemcům) – na jednotlivé emailové adresy. Email je zpráva tvořená sekvencemi řádků. Emailové adresy pak mají tvar lokální-část@doména, kde doména je část za zavináčem a identifikuje server a lokální část značí schránku v doméně [3].

1.3.3 SMTP server a postup komunikace

Abychom mohli odesílat elektronickou poštu, musíme mít na počítači nainstalovaný SMTP server. SMTP server je program, který běží v systému serverového počítače obvykle jako proces na portu 25. Jeden z takových programů je např. Sendmail [5].

Komunikace s SMTP serverem

Postup komunikace s SMTP serverem je popsána v RFC 2821. Jakmile SMTP klient získá zprávu pro odeslání, přebírá odpovědnost za její doručení na jeden či více SMTP serverů [6]. Komunikaci ukazuje blíže obrázek 1.



Obrázek 1: Princip komunikace se SMTP serverem [6]

SMTP protokol se skládá z požadavků a odpovědí. Příkazy SMTP definují přenos pošty nebo funkci mailového systému požadované uživatelem. Příkazy SMTP jsou znakové řetězce ukončené <CRLF>. Odpovědi se skládají z kódu a zprávy oddělených znakem <SP> [6]. Některé příkazy a odpovědi jsou popsány v tabulkách 1 a 2.

Příkaz	**Popis**
EHLO	Extended HELLO (HELO), příkaz určen pro identifikaci SMTP klienta k SMTP serveru
MAIL	Používá se k inicializaci transakce pošty
RECIPIENT (RCPT)	Používá se k identifikaci příjemce
DATA	Používá se pro přenos zprávy, ukončuje se tečkou

RESET (RSET)	Specifikuje, že se ruší daná transakce zprávy
QUIT	Specifikuje, že odesilatel nesmí uzavřít komunikační kanál před odesláním tohoto příkazu a musí vyčkat na odpověď

Tabulka 1: Přehled příkazů k SMTP serveru [6]

Kód	**Popis**
220	Služba připravena
221	Služba uzavírá přenosový kanál
250	Požadovaná akce v pořádku dokončena
354	Začátek posílání zprávy
500	Chyba příkazu, příkaz nebyl rozpoznán

Tabulka 2: Kódy odpovědí SMTP serveru [6]

1.4 Třívrstvá architektura Model-View-Controller

V současné době se čím dál častěji setkáváme s oddělením aplikační vrstvy od prezentační, tedy např. PHP kódu od HTML dokumentu. V PHP značky pro začátek a konec PHP kódu označují místa, které se budou interpretovat a jejich výsledek vypisovat do HTML dokumentu. Zdrojový kód se tak stává nepřehledný, míchá výstupní soubor a funkční část. Z tohoto důvodu je vhodné při vývoji aplikace oddělit aplikační část od části, která má na starost výstup [7].

Model MVC rozděluje aplikaci na 3 logické celky takovým způsobem, aby bylo možno jednotlivé vrstvy upravovat samostatně s minimálním dopadem na ostatní. První část – Model – zahrnuje práci s daty, druhá část – View – se zaměřuje na uživatelské rozhraní neboli výstup aplikace a třetí – Controller – řídí tok událostí a aplikační logiku [8].

Navrhovaný systém bude postaven na PHP frameworku Symfony 2, který staví právě na návrhovém vzoru MVC.

1.5 Webové technologie

Mluvíme-li o webovém obsahu, měli bychom si říct něco o technologiích, pomocí kterých lze takový obsah tvořit. Jedná se o technologie, mezi které patří např. značkovací jazyk HTML, skriptovací jazyk (Javascript), webový server, skriptovací serverový jazyk (PHP, Python) a databázový server (MySQL, MSSQL).

V počátečních dobách Internetu byly webové stránky statické – uživatel požádal o určitou stránku a server mu ji poskytl. S příchodem nových aplikací však narostly požadavky uživatelů [9].

Prvním řešením pro dynamický webový obsah bylo CGI, které umožnilo první funkce jako je přístup k databázi v okamžiku požadavku uživatele, avšak i na tomto bylo znát, že je stále co vylepšit. V roce 1995 přišla společnost Sun and Andreessen s programovacím jazykem Java. Ten umožnil první dynamický obsah pomocí appletů. Následoval vznik klientských skriptovacích jazyků jako je Javascript a serverových skriptovacích jazyků PHP a ASP. Uživatelé však chtěli plnohodnotné aplikace a vývojáři naopak vyhnout se šíření mnoha spustitelných souborů na tisíce počítačů. Reakcí na to byla poměrně nedávno technika Ajax [9].

Některé zmíněné technologie nyní probereme.

1.5.1 Ajax

Ajax je technika, která se opírá o svou hlavní komponentu – Javascript. První technologie, která se k tomu vztahuje, je objekt XMLHttpRequest implementovaný v roce 1999 do prohlížeče Internet Explorer 5, později se rozšířil do prohlížečů Mozilla i Safari. Ajax je přístup ze strany klienta a je na serveru nezávislý – schopen spolupracovat s jakýmikoli serverovými technologiemi (PHP, .NET, J2EE). Dnes je tato technika hojně používaná – používá ji např. Google ve svém vyhledávání [9].

Ajax umožňuje vývojáři webových aplikací asynchronně komunikovat se serverem. Termín byl původně použit jako zkratka pro asynchronní Javascript + XML, dnes již zahrnuje všechny technologie umožňující prohlížeči komunikovat se serverem bez nutnosti obnovení stránky. Technika Ajax nám tedy umožňuje dynamicky aktualizovat

určitý konkrétní obsah stránky, tedy její část, aniž bychom museli znovu načítat celou stránku [9].

Javascript a DOM

Javascript je skriptovací jazyk, který v roce 1995 vyvinula společnost Netscape. Byl vytvořen za účelem ušetřit práci vývojářům, kteří neovládali Javu. Je mocným nástrojem pro vytváření dynamických webových aplikací. Javascript byl původně vytvořen pro dynamickou změnu tagů ve stránkách, ukázalo se však, že stránku lze považovat za objekt, což vedlo k vzniku DOM (Document Object Model). DOM a Javascript byly zpočátku úzce spjaty, později z nich vznikly oddělené koncepty [9].

Document Object Model představuje objektově orientovanou reprezentaci stránky, kterou může skriptovací jazyk měnit. Standardizovalo jej W3C – World Wide Web Consorcium [9].

Dnes máme spoustu frameworků pro ulehčení práce s Javascriptem, takovým je například knihovna jQuery, která bude použita při implementaci systému.

1.5.2 PHP

PHP je skriptovací programovací jazyk, který byl vyvinut k tomu, aby se statické webové stránky obohatily o dynamické skriptování. Před příchodem PHP se pro tyto účely používalo rozhraní CGI, přes které se volaly programy, které by toto umožňovaly. Webový server tak musel spustit nový proces k tomu, aby mohl externí CGI program spustit, což bylo velmi zdlouhavé a značně zpomalovalo načtení stránky. PHP však, neběží-li jako CGI, je zabudován do webového serveru jako modul, podobně jako mohou být zabudovány programovací jazyky PERL, Python či Ruby, takže odpadají požadavky na čas [10].

Skriptování v PHP probíhá tak, že se dokumentu HTML vloží na libovolné místo PHP kód. Skript je pak PHP interpretrem zpracován a výsledkem je HTML dokument, který je serverem odesílán klientovi. Základem PHP kódu jsou značky pro začátek a konec PHP skriptu – ‚<?php‘ a ‚?>‘, tyto značky vymezují část dokumentu, ve které je PHP kód a který bude vyhodnocen. Zpravidla obsahuje různé výpočty, načítání dat z databáze a sestavení výsledného HTML kódu. PHP skript je tedy HTML dokument

obohacen o skripty v jazyce PHP, které mají na starosti dynamické načtení webového obsahu [10].

Vzhledem k tomu, že nároky na webové aplikace jsou čím dál tím vyšší, je vyžadována stále vyšší interaktivita a snadnost použití webových stránek, vývoj takovéto aplikace je také náročnější. Aby se tento vývoj usnadnil, vzniká spousta nástrojů jako nástavby nad PHP – tzv. frameworky, které v sobě nesou nejen spoustu ulehčení, ale také plno naprogramovaných modulů, které může programátor rovnou použít a nemusí je psát. Mezi nejpoužívanější frameworky patří CakePHP, Symfony a z českých např. Nette Framework [11].

1.5.3 Jazyk SQL

SQL je zkratka pro Structured Query Language. Je to databázový jazyk, který se používá pro formulování dotazů na databázový systém. Může se jednat o dotazy k získávání údajů (např. o uživatelích, objednávkách v eshopu, seznamu zboží), příkazy k vkládání a upravování záznamů v databázi nebo o jejich odstraňování [10].

SQL se zařazuje mezi tzv. deklarativní programovací jazyky, což znamená, že kód jazyka vkládáme do jiného programovacího jazyka, který je procedurální [12].

První částí jazyka je DDL – Data Definition Language, což je jazyk pro vytváření databázových schémat a katalogů, druhou SDL – Storage Definition Language – definuje způsob ukládání tabulek, třetí je VDL – View Definition Language – určuje vytváření pohledů, což si lze představit jako virtuální tabulku složenou z několika jiných tabulek, a konečně poslední část – DML – Data Manipulation Language – ta obsahuje základní příkazy pro manipulaci s daty [12].

Podíváme-li se podrobněji na práci s daty, nejčastěji použijeme příkazy SELECT pro získávání dat z databáze, INSERT pro vkládání, UPDATE pro aktualizaci a DELETE pro mazání údajů. Dále pak jazyk obsahuje příkazy pro manipulaci se strukturou tabulek, těmi jsou zejména CREATE TABLE, ALTER TABLE a DROP TABLE [10].

Mezi známé a používané databázové systémy patří MySQL, Oracle, MSSQL a PostgreSQL. Aplikaci postavím na databázovém systému MySQL, který je světově velmi rozšířený.

1.5.4 Další technologie a formáty

Probereme také další základní formáty a technologie pro nastavení konfigurace frameworku a pro tvorbu šablon.

Formát YAML

YAML je formát pro serializaci textových dat, tedy využívá se pro uložení textových dat a pro jejich opětovné načtení, z čehož vyplývá, že je také čitelný obyčejným textovým editorem [13].

YAML nabízí jednoduchost podobně jako textové soubory INI, ale navíc umožňuje v čistém textu vyjádřit i složitější konstrukce, jako struktury či pole. Užijeme-li porovnání s XML, má tu výhodu, že není potřeba tak rozsáhlých znalostí a že se nemusí pro jednoduchá data používat tak mocného jazyka [13].

Twig – šablonovací systém

Oddělení aplikační části od zobrazovací se většinou řeší pomocí šablonování. V Symfony2 můžeme vytvářet šablony jednak pomocí samotného jazyka PHP, ale zároveň se nám nabízí možnost využít vestavěný šablonovací systém. Do frameworku byl implementován šablonovací systém Twig, který značně zpřehledňuje tvůrci zdrojový kód.

Šablona Twig je textový soubor, který generuje různý typ obsahu, kterým může být např. HTML, XML, CSV či Latex. Twig definuje dva typy oddělovačů [14]:

- `{{ ... }}`: dvojité složené závorky se používají pro výpis hodnoty proměnných nebo tisk hodnoty výrazu [14]
- `{% ... %}`: řídí logickou část šablony, používá se pro příkazy v šabloně (podmíněné příkazy, cykly) [14]

1.6 Webové služby

Jestliže je potřeba vyřešit komunikaci mezi dvěma aplikacemi na internetu, je potřeba si tyto informace předávat takovým způsobem, aby si aplikace rozuměly. Pro příklad lze uvést dva internetové obchody, které si vzájemně chtějí předávat informace o

produktech, které nabízejí. Aby si tyto informace mohly předat a aby informacím předaným zdrojovou aplikací cílová aplikace rozuměla a dovedla by z dat vytáhnout smysluplné strukturované údaje, musí si je aplikace předat v určitém formátu. K tomuto účelu slouží tzv. webové služby. Webová služby jsou aplikace, které zajišťují zpřístupnění informací z webu dalším programům takovým způsobem, aby byla dostupná stejným způsobem, jako by byly uloženy někde v lokálním souboru [15].

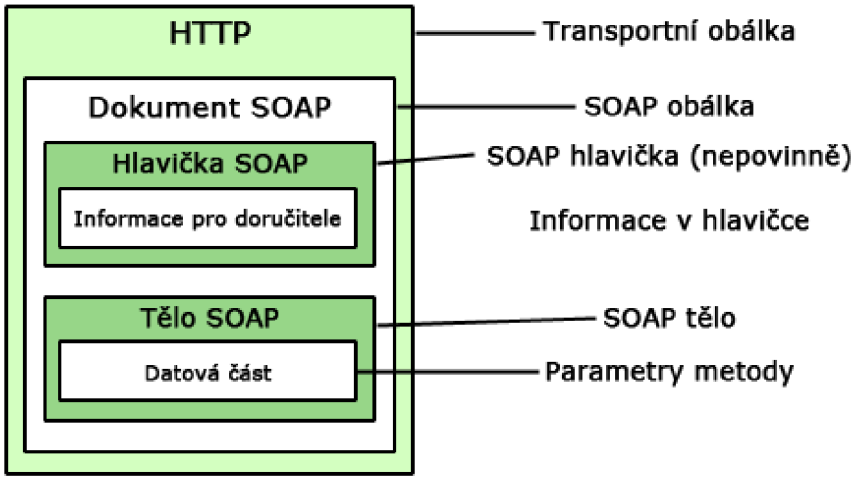
Komunikace počítačů probíhá přes různá rozhraní, které jsou většinou definována operačním systémem a programovacími jazyky, ve kterých jsou aplikace napsány. Na moderních objektově orientovaných platformách lze však psát aplikace, které mohou být využity jinými programy. Například veškeré části a dialogová okna operačního systému jsou psány jako komponenty a programátoři počítačových aplikací tyto komponenty mohou používat, aniž by museli tyto komponenty programovat. Tedy chce-li programátor zobrazit v určitou chvíli běhu aplikace chybovou hlášku, stačí, když použije systémovou komponentu, která chybu vypíše a o ostatní se nemusí starat [15].

Komponenty lze tedy jednoduše využívat ke zjednodušení práce, ovšem problém nastává ve chvíli, kdy je daná komponenta psaná pro konkrétní operační systém nebo pro konkrétní platformu. Na platformách operačního systému UNIX mají opakovatelně použitelné komponenty podobu sdílených objektů, které pracují stejným způsobem jako objekty v architektuře COM v systémech Windows. Programovací jazyk Java má svou architekturu EJB (Enterprise Java Beans). Program napsaný v Javě může používat objekty EJB, program napsaný pro Windows může používat své COM objekty, ale problém nastává ve chvíli, když mají komunikovat mezi sebou. Jestliže některé systémy spoléhají na objekty určité platformy, nazývají se těsně vázané, a pokud je část komunikace mezi aplikací a službou poškozena, může dojít k nepříjemným a nepředvídatelným situacím [15].

S problémy mezi platformami také souvisí problémy, které mohou nastat při komunikaci mezi počítači. Microsoft vyvinul DCOM (Distributed COM), který řeší komunikaci mezi počítači, i přesto je ale volání služby na jiném systému obtížné. Například program spuštěný na počítači s operačním systémem Windows nemůže pomocí svých objektů DCOM volat objekt EJB a naopak. Řešením je v tomto případě SOAP protokol [15].

SOAP

SOAP – Simple Object Access Protocol – je protokol pro posílání zpráv při komunikaci mezi počítači. Pomocí SOAP můžeme vytvářet aplikace pro vzdálené volání metod objektů. Na komunikujících počítačích tedy nemusí běžet stejné systémy ani programy napsané ve stejném programovacím jazyce. Tento protokol je otevřenou standardní syntaxí pro volání metod. Informace plynoucí mezi komunikujícími aplikacemi jsou zapsány ve značkovacím jazyce XML (viz dále). Je to tedy prostý text posílaný prostřednictvím protokolu HTTP, který se používá pro přenos informací na webu [15].



Obrázek 2: Struktura SOAP (zdroj vytvořeno dle [17])

Struktura SOAP protokolu vypadá následovně: skládá se ze tří částí, první je SOAP obálka – ta definuje, co zpráva obsahuje, a také jak se bude zpráva zpracovávat, druhá část je soubor pravidel kódování – ty vyjadřují výskyt údajových typů charakteristických pro danou aplikaci, a nakonec třetí část – konvence pro reprezentaci vzdáleného volání procedur a odpovědí. Všechny části můžeme vidět na obrázku č. 1. Protokol se používá v kombinaci s protokolem HTTP, který zajišťuje přenos [15].

Podíváme-li se více na komunikaci klienta a serveru, skládá se z několika fází a vypadá následovně [15]:

1. Program s úlohou SOAP klienta vytváří XML dokument s údaji pro vzdálené volání metody objektu na externím systému. Klient SOAP je aplikace, která vznáší požadavek na SOAP server. Není to tedy klasický klient. Klient zabalí XML dokument do obálky SOAP.
2. Balíček je odeslán za pomocí HTTP protokolu.
3. SOAP server dostává zprávu. Obvykle se jedná o webový server, který vyřizuje požadavky SOAP, přičemž současně klasicky odpovídá požadavkům o webové stránky. Ten analyzuje došlý balíček a zavolá příslušný objekt, kterému předá parametry ze SOAP dokumentu.
4. Objekt provede požadovanou operaci a vrátí SOAP serveru získanou informaci, ten ji zabalí opět do obálky SOAP.
5. Obálka se odesílá zpět klientovi, je opět uschovaná do hlavičky HTTP.
6. Klient čeká na odpověď, jakmile dorazí, rozbalí obálku a dokument odešle aplikaci, ze které požadavek vzešel.

1.7 Jazyk XML

Jazyk XML je značkovací jazyk, který je určen pro uchovávání a zpracování textových dokumentů. Používá se pro výměnu a sdílení informací a je to otevřený formát, který není svázaný s žádnou platformou. Někdy se také můžeme setkat s úvahou, že se považuje za nástupce jazyka HTML, který je určen pro konstruování webových stránek. Pomocí jazyka XML můžeme snadno přenášet libovolná data z téměř libovolných datových struktur, jak z různých seznamů dat, tabulek, tak i např. relačních databázi. XML formát je založen na prostém textu obohaceném o tagy, kterých používá pro strukturalizaci dat, čímž lze snadno a přesně označit jejich význam [18].

Syntaxe

Jak již bylo řečeno, XML dokument je prostý textový dokument. Každý XML dokument obsahuje elementy, které se do sebe zanořují. Elementy se vyznačují prostřednictvím tagů, to jsou alfanumerické řetězce textu uzavřené do špičatých závorek, tedy znaků ‚<‘ a ‚>‘. Většina elementů v dokumentu obsahuje počáteční a

koncový tag, koncový se od počátečního liší použitím klasického lomítka (tedy ‚/') umístěného bezprostředně za otevírající špičatou závorkou [18].

```
<znacka>Obsah elementu</znacka>
```

Na příkladu vidíme element pojmenovaný „znacka", který obsahuje text „Obsah elementu".

Každý element může obsahovat vnořené elementy, pakliže obsahuje element vnořený nebo více vnořených elementů, nesmějí se jejich tagy křížit. Správnou syntaxi vnořeného elementu si můžeme ukázat následovně [18]:

```
<hlavni>
    Obsah hlavního elementu
    <vnoreny>obsah vnořeného elementu</vnoreny>
</hlavni>
```

Každý tag dále může obsahovat i atributy neboli parametry. Ty jsou připojeny do počáteční značky, od názvu značky jsou odděleny mezerou a udávají se jako název parametru poté rovnítko a následně hodnota parametru uzavřená do uvozovek nebo apostrofů [18]. Zde je uveden příklad zadání parametru:

```
<druh_zbozi velikost="M" barva="zelena">zelené tričko M</druh_zbozi>
```

Základní struktura

Každý XML dokument musí splňovat tyto základní pravidla [18]:

1) Musí obsahovat informaci o deklaraci a kódování dokumentu
2) Musí obsahovat právě jeden kořenový element, v něm pak jsou vnořeny všechny ostatní elementy

Deklarace typu dokumentu a jeho kódování začíná znaky ‚<?xml' a končí párem znaků ‚?>'. Uvnitř těchto znaků je uvedena verze XML dokumentu parametrem ‚version' a jeho kódování prostřednictvím parametru ‚encoding'. Celá deklarace tedy vypadá následovně [18]:

```
<?xml version="1.0" encoding="utf-8"?>
```

Nyní víme, jak vypadá deklarace dokumentu, víme, jak vypadá jeho tělo i jak se do něj zanořují elementy, ukažme si tedy ukázkový příklad celého XML dokumentu:

```
<?xml version="1.0" encoding="utf-8"?>
<sklad>
	<polozka>
		<nazev>Triko</nazev>
		<velikost pocetks="3">M</velikost>
		<velikost pocetks="8">L</velikost>
		<cena mena="CZK">190</cena>
	</polozka>
</sklad>
```

Probrali jsme tedy základy XML, které budeme potřebovat pro práci s webovými službami a pro načítání dat ze vzdáleného serveru.

2 ANALÝZA SOUČASNÉHO STAVU

Podíváme-li se na současné možnosti newsletter systémů, zjistíme, že se nám nabízejí dvě možnosti – vlastní, které firma poskytuje obchodníkovi na míru, a již hotové, mezi které patří projekty určené pro obchodníky, kteří se online zaregistrují.

Firma, pro která potřebuje navrhnout systém, má vlastní řešení newsletteru pracující jako modul začleněný do firemního systému s názvem XYZ Engine. Tento modul ale není dále dostatečně modifikovatelný, protože je funkčně vázaný na celý tento systém a jako takový tvoří pouze jeho komponentu. Je tedy potřeba navrhnout nové řešení zcela nezávislé na elektronickém obchodě.

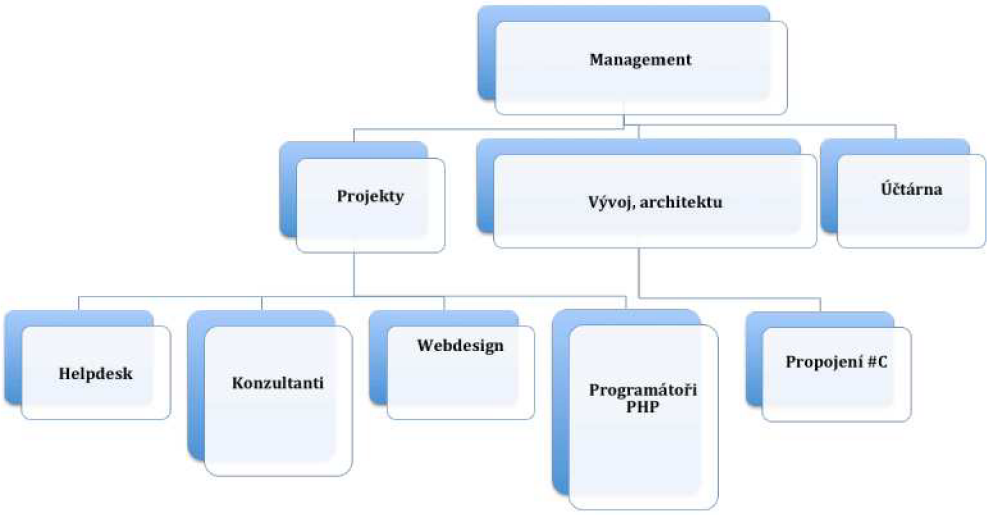
Co se týká hotových projektů, jedná se o systémy jako je např. zahraniční Mailchimp nebo český Smart Emailing. Tyto systémy si rozebereme v této části, podíváme se, co nabízejí, a shrneme si, proč není možné se s nimi spokojit. Také se podíváme, co nám nabízejí PHP frameworky a který je vhodné použít.

2.1 Informace o firmě

Společnost XYZ, s.r.o. je předním dodavatelem e-shopových řešení, dodává internetové obchody (B2B, B2C) vytvářené na míru, propojuje informační systémy Datec a Helios a orientuje se na dodávky středních a velkých implementací v oblasti eCommerce a IT.

2.1.1 Organizační struktura

V čele firmy je management, pod kterým jsou oddělení Projektů, pak Vývoje a nakonec Účtárna. Oddělení Projektů má pak pod sebou Helpdesk, Konzultanty, Webdesign a Programátory PHP. Ve Vývoji je již pak jen Propojení C#. Organizační strukturu firmy blíže ilustruje obrázek 3.



Obrázek 3 : Organizační struktura firmy (zdroj firma XYZ, s.r.o.)

2.2 Požadavky na newsletter systém

Pro vytvoření systému si firma stanovila své požadavky, kterým musí navržený systém odpovídat.

Základním požadavkem je, že systém musí být postaven na třívrstvé architektuře – musí být důsledně oddělena řídící vrstva od databázové a stejně tak od prezentační. Nesmí se stát, že bude v prezentační vrstvě funkcionalita databázové vrstvy – tedy načítání dat z databáze. Zároveň se musí počítat s budoucí možnou záměnou databázového systému (např. z MySQL na MSSQL), tedy musí být umožněna výměna nanejvýš pouhým zásahem do databázové vrstvy aplikace.

Původní požadavek byl, že systém musí být naprogramován v jazyce PHP a musí být využita databáze MySQL, přičemž bude pro celý systém použit některý z PHP frameworků. Po další domluvě bylo upřesněno, že bude využit framework Symfony2. Pod tímto frameworkem chce firma dále v budoucnu vyvíjet další aplikace.

Řešení bude nabízeno také zákazníkům ze zahraničí, veškeré kódy a programátorské řešení systému musí být v angličtině. Výsledný systém musí vícejazyčný – pro začátek především v angličtině a v češtině, následně pak také ve slovenštině a němčině. Musí být samozřejmě také připraven pro implementaci dalších jazyků. Mezi jazyky bude

možné se přepínat. Bude k dispozici tabulka textů nebo úložiště pro překlady. Ideální řešení překladů je použití PO souborů jako používá redakční systém Wordpress.

Systém musí být naddimenzovaný, musí se počítat s nemalým množstvím záznamů – mnoho kampaní pro zákazníka, přičemž každý email z kampaně bude odesílán až 700 000 zákazníkům. Musí se tedy počítat s počty záznamů pro odeslané emaily v řádu milionů.

Systém bude komunikovat s eshopem a pomocí webových služeb bude načítat skupiny příjemců, kteří si přejí dostávat newslettery. Pro tyto skupiny dále bude vracet seznamy příjemců – tedy jejich emailových adres případně jméno a příjmení. V budoucnu se musí počítat s rozšířením pro další systémy.

V newsletter systému bude správa obrázků – obrázky pak bude možno nahrávat do složek a používat pro grafický návrh emailové zprávy, tedy bude se vkládat do zvolené šablony.

Bude možno plánovat, kdy bude email odeslán, odeslání se bude řešit pomocí služby CRON.

Finální produkt bude jedna konkrétní aplikace, ve které bude možno zakládat emailové kampaně, vytvářet emaily, spravovat seznamy příjemců.

V budoucnu bude existovat databanka šablon, na kterou bude systém napojen a odkud bude si stahovat vytvořené optimalizované šablony.

Aplikace se bude instalovat pro každého zákazníka zvlášť na jeho vlastní server. Do aplikace se bude možno přihlásit pomocí uživatelského jména a hesla.

Každý zákazník bude mít své SMTP servery, které budou emaily odesílat. Ty budou uloženy v lokálním seznamu, který si bude moci sám spravovat, přidávat a mazat servery. Z tohoto seznamu pak bude aplikace určovat, na který z nich jej odešle. Serverů může být až desítky, v budoucnu bude možnost nastavovat serverům priority, podle kterých budou systémem použity pro odeslání.

Emaily budou odesílány v pravidelných dávkách na SMTP servery, tyto dávky bude možné individuálně nastavit pro každý uživatelský účet. Emaily se budou odebírat z konkrétní emailové fronty, do které je zařadí plánovač po načtení příjemců.

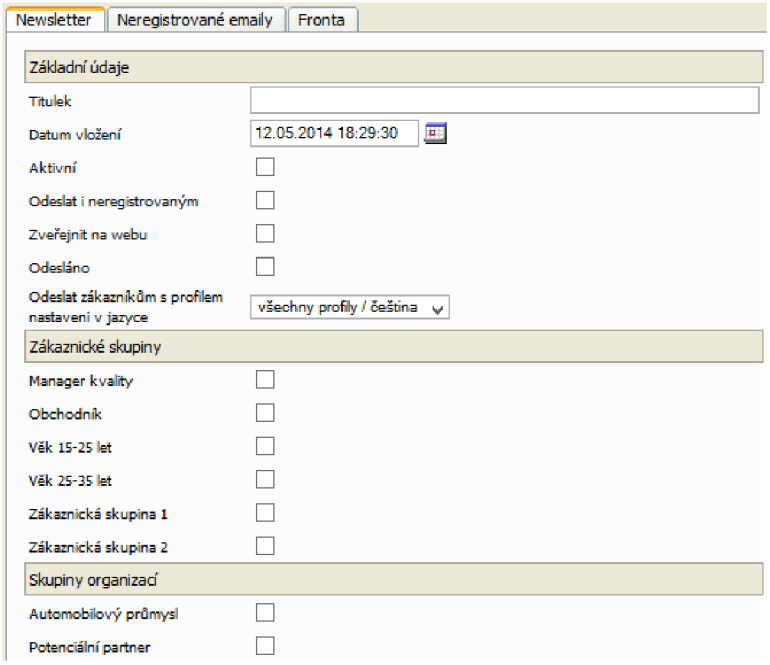
Dále v budoucí rozšířené verzi systému se budou sbírat informace, zda byl email doručen, kolik zákazníků na něj zareagovalo navštívením odkazů v emailu.

Co se týká grafického návrhu systému, není potřeba věnovat mnoho času, finální grafický návrh bude dělat designér, styly a HTML dokument bude tvořit kodér.

2.3 Současné řešení firmy

Pro odesílání newsletterů nyní firma nabízí modul s názvem **Mail centrum**, který je integrovaný v jejich systému pro správu internetového obchodu. Kompletní ukázku tvorby newsletteru v tomto modulu znázorňuje obrázek v příloze č. 1. Jak lze zpozorovat, modul je velmi jednoduchý a zároveň má jen velmi strohé možnosti nastavení.

Vytvoření newsletteru v modulu není rozděleno do jednotlivých kroků, vše je v jednom



Obrázek 4: Formulář pro tvorbu newsletteru v modulu firmy (zdroj vlastní)

rozsáhlejším formuláři, zřejmě protože možností pro jeho vytvoření není tolik, aby byl zapotřebí průvodce. Část formuláře pro tvorbu můžeme vidět na obrázku 4.

Systém umožňuje nastavit základní údaje, jako je název newsletteru, zákaznické skupiny, které budou newslettery přijímat, dále nabízí volbu pro výběr obrázku, který bude v mailu zobrazen, a poté umožňuje vyplnit text zprávy, který bude přiložen.

V další části umožňuje modul přidávat manuálně emailové adresy a seznam adres dále spravovat. K tomu potom dále obsahuje frontu emailů, které byly odeslány, čímž má uživatel přehled, které emaily jsou přichystány pro odeslání a kolik emailů zbývá.

Celý modul je zpracován velmi jednoduše a nedává tvůrci emailu žádné rozšířené možnosti. Uživatel nemá možnost vybírat z hotových šablon, na základě kterých by pak tvořil emailovou zprávu z jednotlivých částí. V podstatě zde ani neexistuje optimalizovaný layout zprávy, jedná se jen o jednoduchou zprávu, která se vytvoří prostřednictvím textového editoru, obohacenou o obrázek.

Modul je tedy pro stávající zákazníky absolutně nedostatečný a neposkytuje žádné pokročilé funkce pro tvorbu interaktivních grafických emailových zpráv. Zapotřebí je tedy navrhnout systém, který by poskytoval zákazníkům firmy mnohem rozsáhlejší možnosti, jako je například vytvoření emailu poskládáním z textů a obrázků do již hotových šablon, které by bylo možno do systému přidávat a přímo pro systém tvořit, systém, který by splňoval také nároky náročnějších zákazníků a bylo by možné jej dále vyvíjet dle potřeby nezávisle na řešení internetového obchodu.

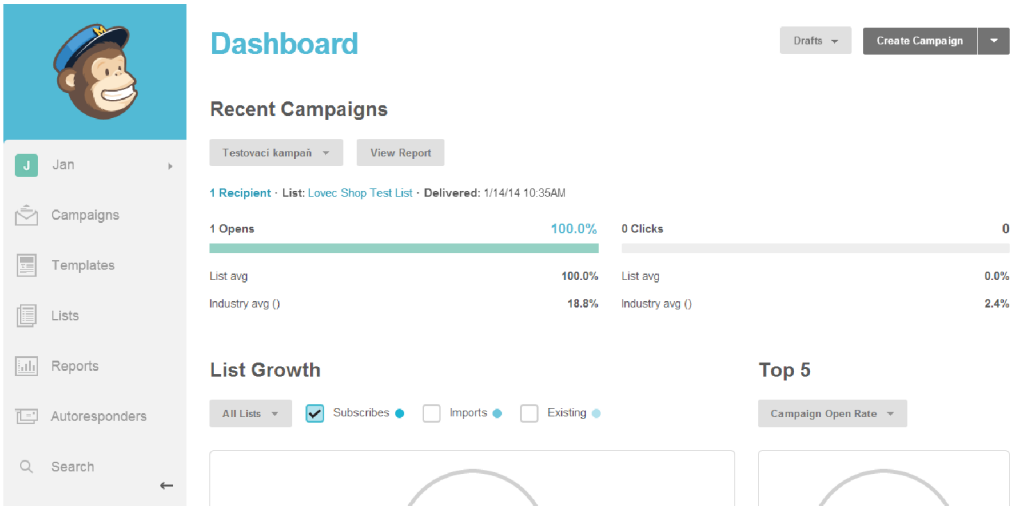
2.4 Současná dostupná řešení

Odesílání emailů s novinkami zákazníkům je v internetovém podnikání velmi vyhledávaným řešením, které napomáhá získání nových a udržení stávajících zákazníků. Proto existují služby, které nabízí obchodníkům rozesílání newsletterů.

2.4.1 Mailchimp

První z hotových řešení newsletter systémů, o kterém se zmíním, je služba Mailchimp. Je k dispozici na webové adrese www.mailchimp.com. Je to systém zaměřený na

tvoření a rozesílání emailových kampaní, který lze rozesílat na vlastní seznamy příjemců.



Obrázek 5: Administrační rozhraní služby Mailchimp.com (zdroj vlastní)

Mailchimp nabízí svým registrovaným zákazníkům tvorbu emailových kampaní. Pro tvorbu kampaní nabízí velmi propracovaný průvodce rozdělený do několika kroků, jimž předchází možnost volby, zda se bude jednat o klasickou kampaň tvořenou HTML emailem spolu s alternativní verzí čistého textu nebo prostou kampaní tvořenou emailem s čistým textem či kampaní s možností odeslat emailem obsah RSS čtečky.

Průvodce tvorbou kampaně umožňuje v prvním kroku nastavit seznam příjemců, na který bude vytvořený email odeslán. Tento seznam lze jednoduše vytvářet v sekci Lists. Druhým krokem je nastavení základních údajů, jako je předmět doručovaného emailu či odesilatel, pak třetím krokem výběr nejrůznějších layoutů emailu či šablon, na základě kterých bude email sestaven. Lze importovat i vlastní šablonu. Dále pak následuje editace obsahu šablony a nakonec poslední krok – potvrzení všech zadaných údajů, čímž je kampaň kompletní.

Výhody Mailchimpu jsou, že nabízí možnost zpětné vazby, tedy sledovat, jak příjemci reagovali na email, nabízí tzv. Reporty. Pro každou kampaň existuje report, v němž se dozvíme, kolika příjemcům email dorazil, kolik z nich si jej prohlédlo a kolik kliklo na odkazy v něm. Reporty lze stahovat v CSV formátu, kde jsou všechny detailní informace shrnuty.

V systému existuje správa šablon – lze importovat vlastní šablonu v ZIP archivu, správa seznamů příjemců a mnoho dalších funkcí. Nabízí také možnost importovat příjemce do seznamů např. v CSV formátu.

Tato služba je pro nás nevhodná především ze tří důvodů. První z nich je, že Mailchimpem lze odeslat zprávy 2000 odběratelům a limit je 12000 emailů měsíčně, vyšší limit je již za poplatek. Firma má pod sebou velké obchodní řetězce, které mají až několik set tisíc zákazníků, výše uvedený limit některé z nich mnohonásobně překročí.

Druhým důvodem je, že v minulosti byly přicházející emaily údajně některými internetovými portály blokované, protože z Mailchimpu údajně chodily uživatelům emailových schránek nevyžádané emaily.

Třetím důvodem, proč je pro nás řešení nevhodné, je, že se jedná o projekt třetí strany a firma, která spolupracuje se zahraničními partnery a má několik tisíc zákazníků u nás i v zahraničí, kterým poskytuje vlastní software šitý na míru, nemůže toto řešení do své nabídky zahrnout. Odtud také ještě plyne další nevýhoda, kterou je nemožnost jakékoli modifikace systému a jakákoli budoucí škálovatelnost na přání zákazníků. Navíc je Mailchimp v angličtině, a proto jej nemůže firma snadno nabízet všem vnitrostátním a zahraničním zákazníkům.

2.4.2 Smart Emailing

SmartEmailing je česká služba, která umožňuje obchodníkům rozesílat newslettery. Najdeme ji na adrese smartemailing.cz.

Tato služba je určena především pro české obchodníky, kteří si chtějí navrhovat své newslettery a rozesílat zákazníkům. Nabízí základní funkce –správu kontaktů, přípravu emailu, statistiky, autorespondery, personalizaci a další.

Systém má velmi jednoduché ovládání, propracovaný návrh emailu, umožňuje přehledně spravovat seznamy příjemců a nabízí možnost importu kontaktů. Podrobné zpracování statistik, které jsou přínosné pro každého obchodníka, je rovněž zajímavou

Obrázek 6: Služba SmartEmailing (zdroj vlastní)

funkcí systému a nabízí přehled o tom, kolik zákazníků email přijalo a kolik shlédlo.

Největším problémem tohoto systému je, že je použitelný zdarma na 7 dní na vyzkoušení, poté se již platí za použití systému poplatky. Firma nemůže svým zákazníkům a zahraničním partnerům nabídnout službu dostupnou pro všechny české obchodníky, ale vedení chce vlastní systém, který bude firma spravovat.

2.4.3 Další možnosti a zhodnocení

Kromě dvou výše uvedených možností řešení existují i další. Lze zakoupit licence k dalším softwarům třetích stran. Opět ale narážíme na problém softwaru, který se musí zakoupit a již není produktem firmy, která dodává vlastní software.

Možných řešení pro rozesílání newsletterů je dnes k dispozici mnoho, vybrat a používat z nich některý může být vhodné pro jednu nezávislou firmu. Poskytovat však toto řešení firma nemůže, protože je dodavatel vlastního softwaru šitého na míru.

Podíváme-li se ještě na výše uvedené systémy, které jsme vybrali pro analýzu, jsou zajímavým řešením, avšak pro nás nedostačující. Produkty Mailchimp a SmartEmailing jsou hotové služby dostupné pro firmy, které nelze jakkoli modifikovat na přání zákazníka. Každý zákazník této firmy by měl mít k dispozici svůj systém, který bude možné dále upravovat a přímo vývojáři firmy spravovat.

U obou systémů je problémem také vícejazyčnost. Mailchimp je v angličtině a SmartEmailing v češtině. Systémy poskytované zákazníkům je potřeba překládat do různých jazykových mutací, což u těchto systémů není možné. Mailchimp má propracované nastavení, možnost správy příjemců, importy a exporty příjemců a výsledků kampaní, avšak jakékoli zásahy do systému jsou nemožné.

2.5 Přehled PHP frameworků

Nyní nastíním, jak je to se situací s PHP frameworky – jaké jsou dnes k dispozici, jak jsou využívané a co nám nabízejí, proč byl pro vývoj vybrán daný framework.

Mezi PHP frameworky patří k světově nejvyužívanějším Laravel, Phalcon, Symfony2, CodeIgniter, Yii, Kohana a CakePHP [19]. Zástupcem mezi českými frameworky je Nette Framework. Obrázek č. 2 ukazuje světovou oblíbenost frameworků koncem roku 2013.

Při výběru vhodného PHP frameworku nás budou zajímat parametry jako je rychlost, bezpečnost, stabilita, pravidelné aktualizace, zda daný Framework využívá ORM model pro mapování entit v databázi na objekty, dále pak šablonovací systém, který používá k zobrazování výsledného obsahu, možnosti validace dat a také licence.

CakePHP

Framework CakePHP je zaměřen na rychlý vývoj prostřednictvím návrhového vzoru MVC. Jeho přednostmi jsou propracované nástroje na důkladnou validaci vstupních dat a má srozumitelně uspořádanou strukturu. Dalšími výhodami tohoto balíku je jeho propojení s nejpoužívanějšími databázemi, pro které využívá tzv. Active Records [20].

Požadavkem tohoto frameworku je mimo HTTP server také PHP ve verzi 5.2.8, a co se databáze týká, pracuje s MSSQL, PostgreSQL, MySQL a SQLite [20]. Produkt je vydáván pod licencí MIT, v době psaní této práce byla vydaná stabilní verze 2.4.9.

CodeIgniter

CodeIgniter vyvíjí společnost EllisLab. Mezi přednosti frameworku patří srozumitelná struktura, použití MVC modelu, je dobře rozšiřitelný o práci s emaily, XML apod. [20].

Co by se dalo hodnotit negativně, je fakt, že tento framework nepoužívá šablonovací systém, takže se programátor musí spokojit při psaní vrstvy View s čistým PHP, které může kód značně znepřehlednit, nehledě na to, že tím přichází o komfort většinou pohodlného krátkého zápisu. Alternativou k tomuto může být také využití existujících šablonovacích systémů prostou implementací do frameworku [20].

Tento PHP Framework vyžaduje opět HTTP server, PHP verze 5.1.6 a databáze – MySQL (4.1+), MSSQL, PostgreSQL, SQLite [20]. Verze frameworku je nyní 2.1.4.

Laravel

Tak jako všechny zmíněné frameworky, i tento využívá návrhový vzor MVC. Jádro frameworku se skládá jen ze základních funkcí, přesto však poskytuje rozsáhlou funkcionalitu, navíc se dá rozšířit o velké množství přídavných balíčků.

Velkou předností Laravelu je propracovaná, rozsáhlá dokumentace, která je přehledně uspořádaná a je obohacena také o spoustu příkladů. Dokumentace frameworku je rozšířena i o možnost zakoupení knihy, tímto zakoupením zároveň vývojář podpoří vývoj [21].

PHP framework Laravel požaduje ke své funkci PHP ve verzi 5.3+ a některou z rozšířených databází jako je opět MSSQL, MySQL, PostgreSQL a SQLite [21]. Nyní se nabízí verze 4.1.

Nette

Za vydáním Nette Frameworku stojí český vývojář, jedná se tedy o český produkt. Framework využívá MVC návrhový model a zaměřuje se především na bezpečné

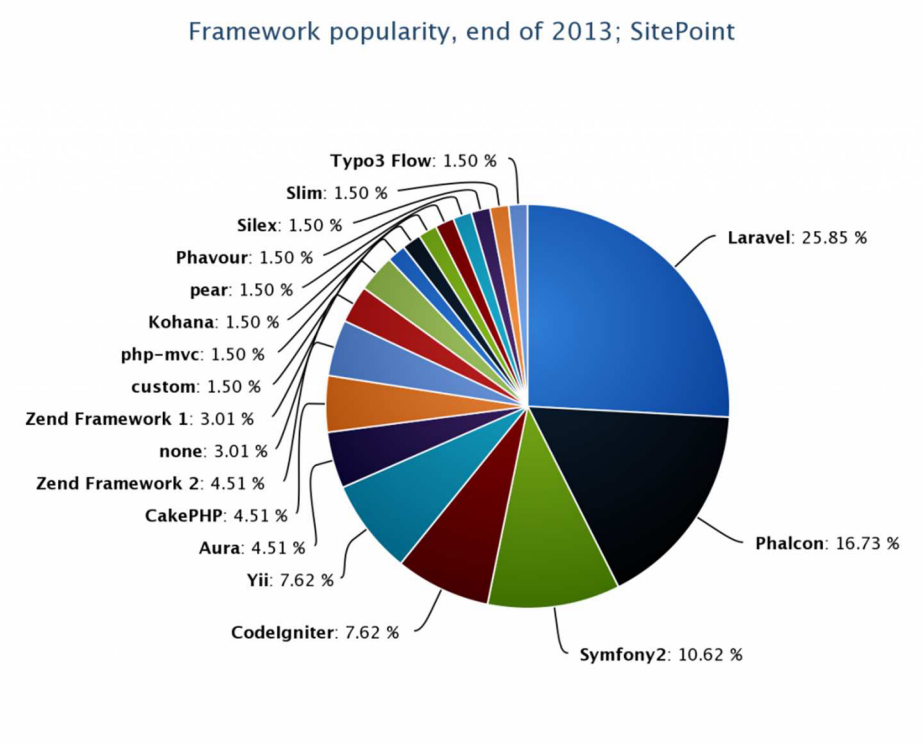
webové aplikace a velmi rychlý vývoj. S databází pracuje prostřednictvím tzv. Nette\Database, NotORM nebo pomocí Dibi knihoven [22].

Výhody frameworku, které si můžeme přečíst i na oficiálních stránkách, jsou propracované ladící nástroje, vysoký výkon, neustále se rozrůstající nabídka doplňků, ale kromě toho také aktivní komunita v České republice [23].

Nette omezuje vývojáře pouze na použití PHP 5.3+ běžící samozřejmě na HTTP serveru [22]. Poslední uvolněná verze je 2.1.2.

2.6 Výběr frameworku a hodnocení

Vzhledem k tomu, že pro české vývojáře je k dispozici český framework, věnovala se výběru frameworku firemní diskuse, ve které se rozhodovalo mezi Nette a Symfony, přičemž zvolen byl Symfony. Jedna z věcí, která k tomu vedla je, že je tento framework zahraniční produkt a firma v budoucnu spolupracovat se zahraničními partnery. Je tedy lepší se neomezovat na český produkt, který navíc nemá zdaleka tak propracovanou dokumentaci jako produkt od zahraničního výrobce.

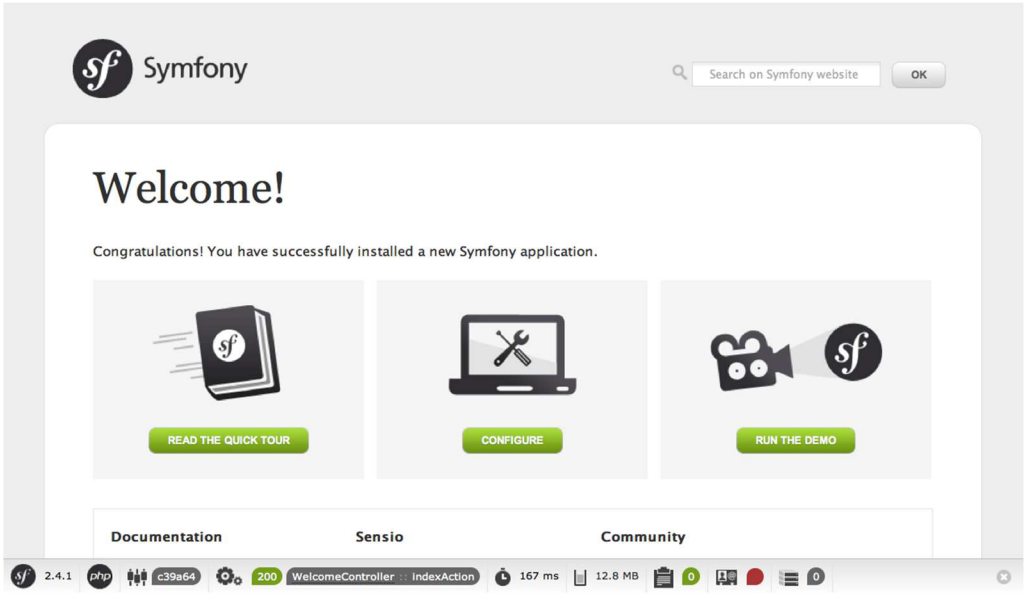


Graf 1: Zastoupení PHP frameworků koncem roku 2013 [19]

2.7 Symfony2 framework

Jedním z nejpoužívanějších PHP frameworků dnes je Symfony2 od firmy SensioLabs. Byl uveden v roce 2005 a za dobu své existence se vyvinul ve velmi silný a stabilní framework, kolem kterého se utvořila obrovská komunita profesionálních vývojářů, a na tomto nástroji vyrostlo mnoho velkých internetových projektů, jako jsou Yahoo! nebo Opensky.com [24].

Symfony plní požadavky, které se od takového frameworku očekávají – rychlost, flexibilita, znovupoužitelnost komponent, ale také to, že je používán nesčetným množstvím stále narůstajícího počtu programátorů, takže se začátečník nemusí obávat, že by se mu naskytly problémy, které by nebyl schopen vyřešit. Kromě tohoto je k dispozici online rozsáhlá dokumentace plná příkladů a řešení [24].



Obrázek 7: Spuštění první aplikace v Symfony2 frameworku (zdroj vlastní)

Symfony2 framework je založen na modelu MVC, tedy odděluje od sebe aplikační logiku od uživatelského rozhraní a od datového modelu. Vrstvu zvanou model zde zastupuje Doctrine, což je vrstva pro práci s údaji uloženými v databázi, dále vrstvu View zde zastupuje šablonovací systém Twig a mezi těmito vrstvami stojí Controller. Controller je PHP metoda či funkce, která zpracovává přicházející požadavky (Requests) a vrací odpovědi (Responses), kterými je nejčastěji HTML soubor. Tvůrce aplikace si může definovat vlastní Controller založený na vestavěné třídě Controller [25].

Ve frameworku Symfony2 je veškerý kód, který programátor napíše, organizovaný v tzv. balíčcích - Bundles. Bundly jsou shromáždění PHP skriptů, CSS stylů, obrázků, Javascriptů, ale také konfiguračních souborů. Konfigurační soubory jsou základem pro nastavení frameworku a komponent, ale také se používají pro určení, který Controller obslouží kterou událost. Nadále jsou pak řešeny buďto pomocí skriptů PHP, anebo pomocí XML souborů, avšak výchozím konfiguračním formátem je YAML, který je oblíbený pro svou jednoduchost a snadnou přehlednost [25].

V této části práce jsem ukázal stávající řešení firmy, které je nedostačující a koncovým zákazníkům plně nevyhovuje. Prošel jsem současné systémy, které se dnes nabízí k využití, a probral jejich klady a zápory a také to, proč je potřeba navrhnout vlastní řešení. Dále provedl analýzu PHP frameworků, které se nám nabízí jako vhodné nástroje pro vývoj newsletter systému, a rozepsal se o Symfony. Nyní přejdu k části, ve které se budu věnovat kompletnímu návrhu – od architektury až po jednotlivé dílčí komponenty.

3 VLASTNÍ NÁVRHY ŘEŠENÍ

V této části na základě poznatků v předchozích dvou částech práce provedu návrh systému. Ukážu také na princip, jakým se budou emaily odesílat, navrhnu architekturu systému, databázovou strukturu, sestavení emailových zpráv a další.

Návrh systému se bude skládat z několika dílčích kroků:

1) Základní návrh aplikace, funkce systému
2) Databázová struktura
3) Konfigurace Symfony frameworku
4) Řešení tvorby kampaní, příjemci, volby odeslání
5) Tvorba emailu – šablony a emailový návrhář
6) Princip odesílání emailů, plánovač, komunikace s SMTP servery
7) Překlad aplikace
8) Načítání příjemců pomocí webových služeb

V následující části práce se podíváme na jednotlivé kroky podrobněji.

3.1 Základní návrh aplikace

Jak bylo napsáno v požadavcích firmy, základní úlohou navrhovaného newsletter systému je vytvořit a spravovat emailové kampaně, které budou rámce pro emaily, které se budou odesílat na sestavený seznam příjemců pomocí SMTP serverů.

Emailový obsah, který bude odesílán na cílové emailové schránky příjemců, bude tvořen z emailových šablon konstruovaných z obrázků a kódu HTML jazyka, kaskádových stylů a dále z pomocí textů, které budou pro daný email uloženy.

V systému budou vystupovat tyto objekty:

- Emailová kampaň
- Email
- Šablona emailu
- Text emailu
- Příjemce

- Odeslané emaily
- Uživatel
- SMTP server
- Jazyk lokace systému
- Obrázek
- Složka obrázku

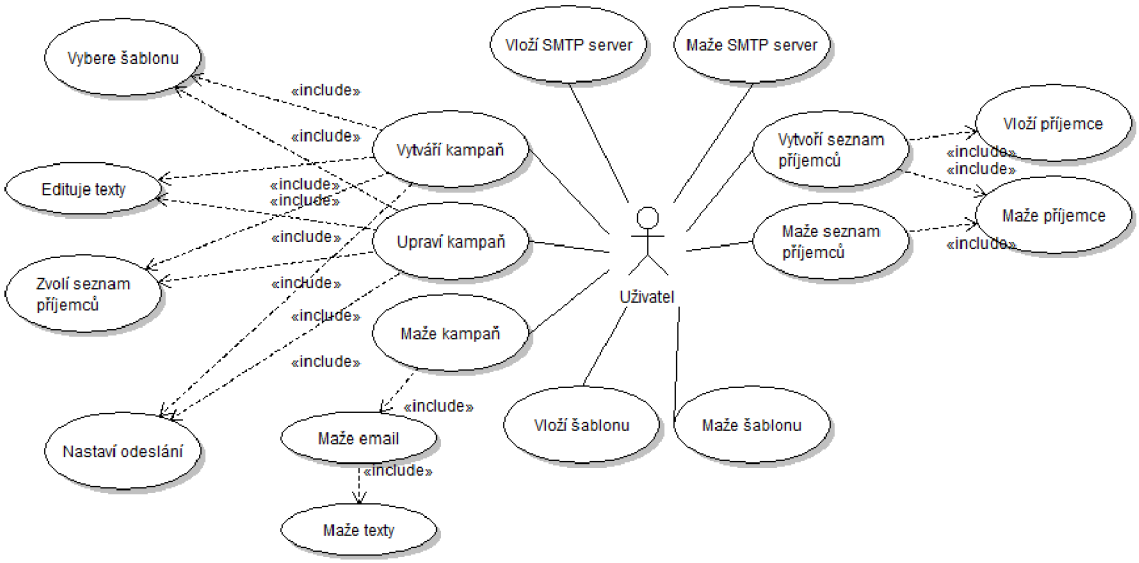
Do emailové kampaně bude patřit email, dále do ní budou vloženi příjemci, email bude odeslán na SMTP server, který emaily odešle na daný server (adresu).

Vzhled aplikace

Vzhled systému bude navržen pro nejjednodušší ovládání s ohledem na to, aby byly hlavní funkce co nejrychleji dostupné. V horní části webu bude volba jazyků a možnost odhlášení z aplikace. V menu vlevo budou odkazy pro vstup do kampaní, šablon, seznamů příjemců, emailové fronty a nastavení aplikace. V pravé části pak detaily vybrané sekce. Obrázek grafického návrhu je vidět v příloze 4. Bude se jednat o návrh, který si pak převezmou a dotáhnou designéři.

Funkce systému z pohledu uživatele

Systém bude nabízet jednotlivým uživatelům funkce, které znázorňuje následující use case diagram.



Obrázek 8 : Use case diagram uživatelských funkcí (zdroj vlastní)

Jak vyplývá z diagramu výše, systém může být rozdělen do několika částí – tedy na část, která řeší správu kampaní, dále na část správy seznamu příjemců, správu emailových šablon a správu seznamu SMTP serverů.

Podíváme-li se podrobněji na správu kampaní, bude umožňovat vytvoření nové kampaně, upravení neodeslané kampaně a nakonec smazání kampaně. Součástí vytvoření a úpravy kampaně bude výběr z šablony pro emaily, poté taky editace textu pomocí vhodného editoru a také vložení seznamu příjemců. Poslední funkcí pak bude nastavení odeslání, které bude umožňovat uživateli nastavit datum a čas, kdy bude email odeslán na SMTP server, případně možnost odeslat ihned.

3.2 Databázová struktura

Vzhledem k funkci systému bylo potřeba zavést entity uvedené v tabulce 3.

Entita	**Popis**
campaign	Data kampaní
email	Data emailu kampaně
email_images	Vazba mezi emailem a obrázky
email_text_data	Vazba mezi emailem a texty
im_folders	Složky ve Správci obrázků
im_images	Obrázky zobrazované ve Správci obrázků
language	Jazyky aplikace
lists	Vlastní seznamy příjemců
list_subscriber	Členové seznamu příjemců
queue	Emailová fronta
sent_emails	Archiv odeslaných emailů
settings	Nastavení a parametry aplikace
smtp_servers	Seznam SMTP serverů s přístupovými údaji
subscriber	Seznam příjemců emailu kampaně
template	Šablony emailů
users	Uživatelé systému a jejich údaje (hesla)

Tabulka 3: Přehled entit v databázové vrstvě

Kompletní databázové schéma je uvedeno v příloze 2. **Entity jsou** pak **mapovány** pomocí **ORM mapovače na objekty**. Mapování popisují konfigurační soubory, které mohou být buďto přímo součástí PHP, nebo v XML souborech či také v YAML formátu. Pro svou jednoduchost byl zvolen formát YAML. Mapování entity campaigns znázorňuje kód v příloze 3.

3.3 Konfigurace Symfony2

Před psaním aplikace v Symfony2 je potřeba provést instalaci a nastavení frameworku. Framework stáhneme ze stránek http://symfony.com/download a extrahujeme ze ZIP archivu např. do kořenové složky našeho HTTP serveru. Dalším krokem je spuštění instalace, což provedeme zadáním adresy [nazev_http_serveru]/[cesta_k_symfony]/web/config.php do webového prohlížeče. Poté nás přivítá jednoduchý instalátor, který nás seznámí s tím, zda máme v pořádku konfiguraci serveru a PHP. Můžeme zde provést konfiguraci, která se skládá z vyplnění údajů pro přístup k databázi.

Konfiguraci můžeme provést také prostou úpravou konfiguračního souboru parameters.yml, který se nachází v [symfony_root]/app/config/. Zde se nachází veškeré **konfigurační soubory**. Jsou ve formátu YAML, mohou být však i ve formátu XML či samotném PHP.

Po této instalaci již můžeme přistoupit k psaní vlastní aplikace tím, že si založíme **vlastní balíček**. Ten pak bude umístěn ve složce [symfony_root]/src/[libovolna_slozka]/[nazev_balicku]/.

V konfiguračním souboru routing.yml pak také nastavujeme veškeré cesty v aplikaci. Příklad **nastavení cesty** v **YAML formátu** pro seznam kampaní může vypadat následovně:

```
campaigns:
    path:     /campaigns/
    defaults: { _controller: MainSystemBundle:Campaign:list }
```

Tím zajistíme, že požadavek na url uvedenou v atributu path převezme námi napsaný program v daném Controlleru.

3.4 Řešení tvorby emailových kampaní

V systému najdeme Správu kampaní, ve které bude seznam všech kampaní, které jsme realizovali. Bude možnost je upravovat, ale to jen pokud již nebudou potvrzeny k odeslání, a mazat. Nemůže chybět možnost vytvářet nové. Tvorba i úprava budou řešeny podobně, a to pomocí jednoduchého **průvodce**, který bude rozdělen do **6 kroků**:

1. základní informace
2. výběr šablony
3. tvorba obsahu
4. výběr příjemců
5. potvrzení
6. nastavení odeslání

Prvním krokem je formulář, který ukládá základní informace o kampani. Těmito základními informacemi jsou: jméno kampaně, předmět emailu, emailová adresa odesilatele, jméno odesilatele.

V kroku **Výběr šablony** je umožněno uživateli zvolit si HTML předlohu, na které bude navrhovat email. Šablony se načítají z vlastního seznamu, kde bude možnost později importovat vlastní navržené šablony. V sekci je tlačítko pro výběr šablony, po kliknutí se zobrazí box pro výběr šablony.

Tvorba obsahu je realizována komponentou zvanou emailový návrhář. Tato komponenta bere šablonu a v ní určená místa pro přepis textu ze šablony vlastním textem. Po kliknutí na text v šabloně se uživateli objeví formulář pro editaci zprávy. Zde může zapsat svůj text a ten uložit. Systém poté pošle ajaxový požadavek na úpravu textu v databázi.

Čtvrtým krokem průvodce kampaní je **Vložení příjemců**. Uživatel smí vložit do databáze příjemce z vlastního seznamu, který si vytvořil, nebo ze seznamu, který se načte z externího zdroje – tím bude systém XYZ Engine – eshop obsahující určité skupiny příjemců, ze kterých je možno vybírat.

Předposledním krokem kampaně je krok **Potvrzení**. Tato část sestává z kontroly předchozích kroků – tedy, zda jsou platně vyplněné základní údaje o kampani, zda je

vybrána šablona, zda jsou vyplněny všechny texty a zda jsou vloženi příjemci. Pokud je vše v pořádku, uživatel může potvrdit kampaň.

Poslední krok je **Odeslání** a skládá se z nastavení specifikující čas, kdy bude kampaň odeslána. K dispozici jsou dvě možnosti – odeslat ihned a odeslat v daný časový okamžik. Po dokončení tohoto kroku může uživatel kampaň potvrdit a tím ji přesunout do odeslaných. Po tomto kroku již kampaň nebude moci upravovat.

Vytvoření objektu kampaň

Kampaň jako taková je tvořena objektem, se kterým se v programu manipuluje. Pro vytvoření nové kampaně je určena metoda CampaignControlleru **createAction**, která ze systémových služeb načte UserController – řídící část pro správu uživatele, z které použije metodu pro nalezení objektu přihlášeného uživatele. Poté přímo vytvoří novou kampaň pro daného uživatele s aktuálním datumem, přičemž zapíše její údaje do databázové tabulky campaigns. Následně pak přesměruje dění aplikace na první krok formuláře, kde se nastavují potřebné údaje. Zde je znázorněn kód, který nám řeší založení nové kampaně:

```
public function createAction()
{
        $this->init();

        // get a logged user object
        $user = $this->get('user_controller')->getLoggedUser();

        // create campaign
        $camp_id = $this->rp->createNewEmptyCampaign($user);

        // redirect to step 1
        return $this->redirect($this->generateUrl('step1_basic',
                array('camp_id' => $camp_id)));
}
```

3.5 Tvorba emailu

K vytvoření výsledného emailového obsahu je potřeba sestavit email z přednastavené šablony a doplnit jej vlastním textovým obsahem a vlastními obrázky.

3.5.1 Šablona

Šablona emailu je tvořena zdrojovým kódem v jazyce HTML. S přihlédnutím k tomu, že je komplikované navrhnout grafický emailový obsah tak, aby se zobrazil stejně na všech klientech pro zobrazování elektronické pošty, jedná se o klasický HTML dokument, který nemá žádné složitější nastavení. Je to tedy dokument bez rozšířeného nastavení a bez zvláštních stylů.

Samotná šablona je tedy tvořena HTML dokumentem, hlavička dokumentu obsahuje pouze tag <title> a <meta> s nastavením typu dokumentu a znakové sady – Content-type. V těle je potom část dokumentu určující zobrazení, tedy obsahuje layout realizovaný tabulkami v HTML. Do parametrů tagů jsou pak přímo vkládány atributy style pro stylování dokumentu.

V dokumentu jsou dvě možnosti značení proměnného obsahu:

1. **Textový obsah** – textový obsah v dokumentu je uzavřen do sekvencí textových řetězců {% n} a {%}, kde n značí číslo textu v rámci šablony.
2. **Obrázky** – HTML tagy obrázku jsou uzavřeny do sekvencí řetězců {%i n} a {%}, kde n je číslo obrázku v rámci šablony.

Import šablony

Šablony do systému bude možno nahrávat prostřednictvím formuláře. Bude se jednat o ZIP archiv obsahující dokument template.html s HTML dokumentem šablony bez jakýchkoli externích zdrojů s výjimkou obrázků. Obrázky budou přiloženy v tomto archivu přímo v kořenovém adresáři, přičemž budou pojmenovány čísly a příponou dle typu obrázku.

V archivu bude dále přiložen soubor info.xml, ve kterém bude zahrnut název šablony, autor a zdroj, odkud pochází.

3.5.2 Emailový návrhář

Pro vytvoření obsahu emailové zprávy je v systému tzv. emailový návrhář. Ten prochází HTML kód šablony a hledá sekvence znaků, podle kterých pozná, že se jedná

o část textu, která je určena pro přepsání. Tuto oblast označí a umožní ji upravit prostřednictvím HTML editoru.

Jako HTML editor je použit jeden z volně dostupných editorů s názvem TinyMCE. Je to HTML WYSIWYG editor nezávislý na platformě šířený pod licencí GPL, který jednoduše zaměňuje textové pole (textarea) za pole editoru, kde lze vkládat formátovaný text.

Emailový návrhář najde veškeré výskyty modifikovatelných textů a obrázků

- `{% n}modifikovatelný text{%}`
- `{%i n}<img src="..." style="...">{%}`

a tyto nahradí za:

- `<div id="text[n]>modifikovatelný text</div>`
- `<img id="i[n]" src="..." style="...">`

Konverze je řešena PHP skriptem. Poté již přebírá vizuální část práce program napsaný ve frameworku jQuery v Javascriptu a ten při kliknutí na text vyvolá proceduru, která vytvoří instanci editoru TinyMCE a vloží do něj text k editaci. Pro obrázky se otevře Správa obrázků (popsaná dále).

Po dokončení editace bloku lze text uložit. Text se odešle Javascriptem pomocí Ajaxu do PHP skriptu, který text uloží. PHP skript zároveň také při konverzi sám zjišťuje, zda existuje již text pro daný text z šablony a sám jej nahradí.

3.5.3 Správa obrázků

Kromě náhrady textu je potřeba do emailové zprávy vkládat vlastní obrázky, proto bude v systému existovat modul s názvem Správce obrázků pro jejich správu. Modul bude umožňovat třídit obrázky také do jednotlivých složek. Celý modul bude vyjma uploadu obrázků uveden jako ajaxová aplikace. Každé otevření složky a načtení obsahu nebude obnovovat celou stránku.

Správce obrázků bude rozdělen vizuálně na dvě části. Vpravo bude seznam složek, které bude možno přidávat či mazat a po kliknutí na některou z nich se vygeneruje její obsah do levé části.

Zároveň bude možno uploadovat obrázky ve formátu JPEG, GIF a PNG. Správce obrázků nám uploadované obrázky sám upraví a prostřednictvím PHP knihovny SimpleImage vygeneruje náhled obrázku o maximálních rozměrech 140 x 140 pixelů pro rychlé načítání.

3.6 Princip odeslání emailu

Pro odeslání emailu budeme mít **frontu** realizovanou prostřednictvím databázové tabulky **queue**. Seznam odeslaných emailů bude veden v tabulce **sent_emails**, do které se budou přesouvat emaily z fronty po odeslání z SMTP serveru.

Emailová fronta **queue** s emaily připravenými k odeslání bude načítána skriptem, který se bude spouštět pomocí systémové služby CRON v pravidelných intervalech. Tento skript bude vybírat emaily z fronty a odesílat.

Informace o stavu emailu bude vedená v systému v sekci **Emailová fronta** a uživatel systému bude moci pravidelně sledovat současný stav emailů ve frontě. S přihlédnutím do archivu bude vědět, které emaily se již podařilo v pořádku odeslat a které ne. Emaily, jejichž odeslání skončilo chybou, budou označeny příznakem **failed**.

Odeslání emailu bude umožněno ve dvou volbách:

1. Odeslat ihned
2. Odeslat v určité datum a určitý čas

V obou případech bude obsluhovat odeslání opět systémová služba CRON, která spustí PHP skript. V prvním případě budou mít kampaně nastavený příznak **straight_away**, ve druhém nastavený datum a čas odeslání. Všechny nevyřízené kampaně budou mít nastavený příznak **in_queue**. Skript vybere kampaně s nenastaveným příznakem in_queue a současně s nastaveným příznakem straight_away nebo datumem nižším, než který bude v danou chvíli. Vybere příjemce emailu a ty vloží do fronty.

Další úloha CRON dále bude spouštěna cca každých 10 minut a bude vybírat emaily z fronty po určitých dávkách a ihned odesílat na SMTP server k odeslání. Podaří-li se email v pořádku odeslat, přesune email do odeslaných.

SMTP serverům bude možno nastavit příznak **active**, který bude značit, že je daný server aktivní a lze jej použít k odeslání.

3.6.1 Odeslání prostřednictvím SMTP serveru

Odeslání emailu proběhne skriptem, který načte třídu pro odesílání emailu prostřednictvím SMTP serveru. Třída napsaná v PHP naváže komunikaci se serverem a bude na něj postupně v cyklu odesílat zprávy. Celá komunikace bude pracovat následujícím způsobem:

1) Navázání spojení:
   a) Provede se pokus o navázání spojení pomocí PHP funkce fsockopen(), v případě SSL zabezpečení se před adresu serveru přidá řetězec pro protokol – sekvence znaků „ssl://“. Očekává se odpověď serveru s kódem 220.
   b) Pošle se EHLO, očekává se odpověď s kódem 250.
   c) V případě TLS zabezpečení se provede autentifikace – pošle se příkaz STARTTLS, očekává se kód 220, poté se znovu pošle EHLO, očekává se odpověď 250.
   d) V případě autentifikace (existuje jméno a heslo pro daný server):
      i) Pošle příkaz AUTH LOGIN, očekává se kód 334.
      ii) Pošle se uživatelské jméno kódované v base64, očekává se kód 334.
      iii) Pošle se heslo v base64, očekává se odpověď kód 235.
2) Odeslání seznamu emailů:
   a) Pošle se kód s odesilatelem zprávy MAIL FROM: <email_from>, očekává se kód 250.
   b) Pošle se kód s příjemcem zprávy RCPT TO: <email_to>, očekává se odpověď s kódem 250.
   c) Pošle se příkaz DATA, očekává se kód 354.
   d) Pošle se obsah zprávy (kompletní obálka), nakonec se pošle tečka se závěrečnými znaky (CR, LF) pro ukončení, očekává se odpověď 250.
3) Komunikace končí odesláním příkazu QUIT, přijetím odpovědi a následným ukončením navázaného spojení PHP funkcí fclose().

V případě jakékoli chyby výše (např. nesprávná odpověď serveru) se odesílání daného emailu ukončí a pokusí se odeslat další.

3.7 Překlad aplikace

Překlad aplikace bude řešen pomocí knihovny **GNU gettext**, která je k dispozici i pro PHP. Řešení sestává z uložení překladů do souborů s příponou .PO, zpracování je jednodušší, protože na to již připraven i Symfony2 framework.

Lokalizační soubory uložíme do složky [aplikace]/src/Main/SystemBundle/ Resources/translations/. Překlady jsou pak rozděleny, jak říká specifikace PO souborů, do samostatných souborů dle lokalizace – např. en.po pro angličtinu, cs.po pro češtinu. V tomto případě budu umisťovat české texty do souboru messages.cs.po, texty v angličtině pak analogicky do messages.en.po.

Překladové soubory se budou generovat pomocí programu Poedit, který lze volně stáhnout z adresy http://www.poedit.net/download.php. Příklad souboru messages.cs.po s českými překlady je v příloze 5.

Informace o **nastavení jazyka** uživatele budeme ukládat do databáze **k danému uživatelskému účtu**. Tato nastavená lokalizace se bude načítat v Symfony pomocí tzv. **listeneru** a nastavovat jako lokalizace prostředí.

Pro nastavení jazyka uživatel v **pravé horní části systému** najede na nastavenou lokaci a v dynamicky zobrazeném menu bude moci vybrat lokaci. Odkaz ho přeměruje na adresu uživatele /set_lang?lang=[jazyk], kterou již zpracuje Symfony a převede celou akci na UserController, kde vyvolá metodu setLangAction() s parametrem jazyka. Ta se již postará o uložení tohoto nastavení k přihlášenému uživateli.

Veškeré texty v šablonách Twigu pak budou upraveny modifikátorem **|trans**, který zajistí, že se PHP knihovna gettext pro daný text pokusí najít v překladech alternativní

```
3  {% block title 'welcome'|trans %}
4
5  {% block content_header '' %}
6
7  {% block content %}
8      <h1>{{ 'Dashboard'|trans }}</h1>
9
10     <h2>{{ 'Get started'|trans }}</h2>
11
12     <div class="dashb-opt-box">
13         <a href="{{ path('camp_create') }}">{{ 'Create a new campaign'|trans }}</a>
14         <div>
15             {{ 'Campaigns are emails, that are sent'|trans }}
16             {#<a href="{{ path('camp_create') }}">{{ 'get more'|trans }}</a>>#}
17         </div>
18     </div>
```

Obrázek 9: Řešení překladů v šabloně Twig (zdroj vlastní)

text podle identifikátoru **msgid** v PO souboru pro aktuálně nastavený jazyk. Příklad šablony v enginu Twig znázorňuje obrázek 9.

3.8 Načítání seznamu příjemců pomocí webových služeb

Systém bude umět spolupracovat zpočátku s aplikací internetového obchodu **XYZ Engine**, v budoucnu i s aplikacemi českých a zahraničních partnerů firmy, mimo jiné také se systémy společnosti Helios. Z externích zdrojů bude systém prostřednictvím webových služeb načítat skupiny zákazníků a pro dané skupiny pak kontakty, na které se bude newsletter odesílat.

Mechanismus načítání bude mít dvě fáze. První fáze načítání bude probíhat při tvorbě kampaní. V kroku **Výběr příjemců** bude dispozici volba **Načíst z externího zdroje**, odkud se budou načítat skupiny zákazníků v eshopu, ze které bude moci tvůrce kampaně vybrat. Druhá fáze načítání proběhne v době odeslání newsletteru. Spustí se opět webová služba, tentokrát pro načtení konkrétních kontaktů – emailových adres, jména a příjmení. Webová služba nám bude vracet XML soubor, který bude obsahovat jméno a příjmení příjemce a jeho emailovou adresu. Vzor souboru je níže:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Recipients from XYZ Engine 6.0 -->
<recipients>
      <recipient name="Jan" surname="Kundrát">
            jan@jankundrat.cz</recipient>
```

```
        <recipient name="Josef" surname="Kundrát">
              josef@jankundrat.cz</recipient>
</recipients>
```

Externí zdroj bude nastavitelný v nastavení aplikace, kde bude adresa serveru a přístupové údaje.

3.9 Shrnutí

Provedl jsem tedy návrh celého systému. Návrh řeší veškeré hlavní části a postupy, jak implementovat newsletter systém.

Základem je instalace frameworku, na kterém bude aplikace dále vyvíjena. Je popsáno nastavení, instalace a psaní prvního programu pro tento framework. Dalším stěžejním bodem je pak databázové schéma. Následuje pak princip tvorby kampaní a emailů a postup odeslání sady emailů.

Důležitá informace je, že vstup do rozhraní aplikace bude pod přihlašovacími údaji – uživatelské jméno a heslo. V aplikaci bude seznam uživatelů, kteří se budou moci do systému přihlašovat. Prozatím neřeším odlišování práv uživatelů, ale aplikaci chci přizpůsobit možnosti, že by v budoucích verzích systému bylo možno vytvářet uživatele hlavním uživatelem – administrátorem. Tito uživatelé by pak měli různá práva, např. vytvářet kampaně, mazat kampaně, spravovat SMTP servery, nastavovat hlavní parametry aplikace, či pouze sledovat reporty, o které by aplikace v budoucnu měla být rozšířena.

Současný hotové návrh tedy popisuje hlavní části, které budou realizovat kompletní systém. Navrhuji jednak základní objekty, které v systému budou figurovat, přes databázové schéma aplikace, tedy propojení a vazby mezi objekty, kompletní řešení tvorby kampaně a návrhu emailu, import šablon, vkládání obrázků. Detailně se zaměřuji na odeslání seznamů emailů na jednotlivé příjemce od rozdělení fronty na dávky až po samotné příkazy na SMTP server. Vzhledem k tomu, že aplikace bude vícejazyčná, neopominul jsem ani návrh, jak vícejazyčnost realizuji pomocí vestavěných prostředků jazyka PHP. Ke konci je pak stručně popsáno rozhraní, které umožní propojení aplikace s informačními systémy a prostřednictvím tohoto rozhraní načítat příjemce.

3.10 Ekonomické zhodnocení

Cílem firmy je poskytnout svým zákazníkům kompletní software, tedy nejen jeho návrh, ale kompletní produkt vč. implementace. Cena, kterou firma zaplatí za tento systém, se odvíjí od celkového času potřebného pro vývoj systému.

Návrh systému by za běžných podmínek prováděl zkušený pracovník s měsíčním platem cca 40 000 Kč. Řešil by, jak by celý systém vypadal, náplní jeho práce by bylo zhotovit projekt podobný obsahu této práce. Samotný návrh odhaduji na 20 pracovních dní, přičemž za pracovní den považuji 8 odpracovaných hodin.

Jednotlivé části systému by programoval PHP programátor. Jeho plat je cca 35 000 Kč. Náplní práce programátora by bylo uvést do funkce všechny části systému a přivést na svět kompletní funkční produkt. Doprogramovat jednotlivé části systému může trvat 1 až 2 měsíce v závislosti na zkušenostech a praxi programátora, tedy cca 30 pracovních dní.

Poslední částí by bylo řešení instalace systému, tedy vytvoření samoinstalačního balíčku, což odhaduji u zkušeného softwarového vývojáře na 10 pracovních dní.

Tabulka 4 informuje o celkových nákladech na realizaci systému.

Činnost	**Potřebný čas (hod)**	**Cena (Kč)**
Návrh systému	160 hod.	40 000 Kč
Implementace řešení, programování	240 hod.	52 500 Kč
Řešení instalace softwaru	80 hod.	17 500 Kč
Celkem:	**480 hod.**	**110 000 Kč**

Tabulka 4: Celkové náklady na realizaci systému

ZÁVĚR

Cílem práce bylo vytvořit návrh systému pro rozesílání newsletterů. Tento cíl se podařilo splnit, návrh byl realizován, systém je připraven na implementaci, na které se nyní pracuje, a měl by být v následujících měsících uveden do provozu.

Návrh firmě přinesl potřebné materiály pro kompletní realizaci systému, o který mají zájem její zákazníci. Navržený systém bude zákazníkům firmy, tedy obchodním řetězcům, poskytovat nástroj pro rozesílání emailů s novinkami a akčními nabídkami jejich zákazníkům.

Dokončený systém bude umožňovat tvorbu emailových kampaní s návrhem emailu tvořeného emailovou šablonou a dynamickými texty a obrázky. Dále bude shromažďovat seznamy příjemců, které bude možno v systému vytvářet a také je získávat z jiných informačních systémů. Nabídne uživateli odesílání emailů na rozsáhlé seznamy příjemců prostřednictvím jeho vlastních SMTP serverů.

Bakalářská práce tedy přinesla firmě návrh systému, jehož vývoj nyní probíhá a bude v následujících měsících dokončen – k dispozici jsou první testovací verze. Systém bude zákazníkům firmy poskytovat žádaný nástroj pro zkvalitnění jejich nabídky zboží, což by také mohlo zvýšit jejich zisk.

SEZNAM POUŽITÉ LITERATURY

[1] BRUCKNER, Tomáš. *Tvorba informačních systémů: principy, metodiky, architektury.* 1. vyd. Praha: Grada, 2012, s. 15. Management v informační společnosti. ISBN 978-80-247-4153-6.

[2] KOTLER, Philip. *Moderní marketing: 4. evropské vydání.* 1. vyd. Praha: Grada, 2007, s. 181-182. ISBN 978-80-247-1545-2.

[3] LEVINE, John R. *Qmail: správa unixových poštovních systémů.* 1. vyd. Překlad Martin Kysela. Praha: Grada, 2007, s. 3-9. ISBN 978-80-247-1698-5.

[4] ROSENBROCK, Eric a Eric FILSON. *Linux, Apache, MySQL a PHP: instalace a konfigurace prostředí pro pokročilé webové aplikace.* 1 vyd. Praha: Grada, 2005, s. 152 - 155. ISBN 80-247-1260-1.

[5] KYSELA, Martin. *Linux: kapesní průvodce administrátora.* Praha: Grada Publishing a.s., 2004, s. 158. ISBN 80-247-6249-8.

[6] RFC 2821. *Simple Mail Transfer Protocol*. AT&T Laboratories, 2001. Dostupné z: http://www.rfc-base.org/txt/rfc-2821.txt

[7] VRÁNA, Jakub. *Šablony. PHP triky* [online]. 2005 [cit. 2014-03-25]. Dostupné z: http://php.vrana.cz/sablony.php

[8] BERNARD, Borek. Úvod do architektury MVC. *Zdroják* [online]. 2009 [cit. 2014-03-12]. Dostupné z: http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/

[9] ASLESON, Ryan. *AJAX: vytváříme vysoce interaktivní webové aplikace*. Vyd. 1. Překlad Jakub Zemánek. Brno: Computer Press, 2006, 269 s. ISBN 80-251-1285-3.

[10] KOFLER, Michael a Bernd ÖGGL. *PHP 5 a MySQL 5: průvodce webového programátora*. Vyd. 1. Brno: Computer Press, 2007, 607 s. ISBN 978-80-251-1813-9.

[11] STOUPA, Václav. 2008. Přehled a vývoj PHP frameworků. *Root.cz* [online]. [cit. 2014-03-25]. Dostupné z: http://www.root.cz/clanky/prehled-a-vyvoj-php-frameworku/

[12] SKŘIVAN, Jaromír. 2000. Databáze a jazyk SQL. *Interval.cz* [online]. [cit. 2014-03-12]. Dostupné z: http://interval.cz/clanky/databaze-a-jazyk-sql/

[13] MALÝ, Martin. YAML: Serializační formát pro ukládání dat. *Zdroják* [online]. 2009 [cit. 2014-05-19]. Dostupné z: http://www.zdrojak.cz/clanky/yaml-serializacni-format-pro-ukladani-dat/

[14] Twig for Template Designers. *TWIG - Documentation* [online]. 2011 [cit. 2014-05-19]. Dostupné z: http://twig.sensiolabs.org/doc/templates.html

[15] TRAVIS, Brian E. *XML a SOAP: programování serverů BizTalk.* Vyd. 1. Praha: Computer Press, 2000, 418 s. ISBN 807226303X.

[16] SOAP Version 1.2 Part 0: Primer (Second Edition). [online]. Dostupné z: http://www.w3.org/TR/2007/REC-soap12-part0-20070427/#L1244

[17] HOROVČÁK, Pavel. Webové služby - třetí generace internetu. [online]. [cit. 2014-05-19]. Dostupné z: http://www.systemonline.cz/clanky/webove-sluzby-treti-generace-internetu.htm

[18] KOSEK, Jiří. 2000. *XML pro každého: podrobný průvodce.* 1. vyd. Praha: Grada, 163 s. ISBN 80-716-9860-1.

[19] SKVORC, Bruno. Best PHP Frameworks for 2014. *Sitepoint* [online]. 2013 [cit. 2014-04-29]. Dostupné z: http://www.sitepoint.com/best-php-frameworks-2014/

[20] JAKOUBĚ, Jaroslav. ORM test PHP frameworků - CakePHP, CodeIgniter. *Zdroják* [online]. 2013 [cit. 2014-05-08]. Dostupné z: http://www.zdrojak.cz/clanky/test-php-frameworku-cakephp-codeigniter/

[21] JAKOUBĚ, Jaroslav. ORM test PHP frameworků - Kohana, Laravel. *Zdroják* [online]. 2013 [cit. 2014-05-08]. Dostupné z: http://www.zdrojak.cz/clanky/orm-test-php-frameworku-kohana-laravel/

[22] JAKOUBĚ, Jaroslav. ORM test PHP frameworků - Nette, Prado. *Zdroják* [online]. 2013 [cit. 2014-05-08]. Dostupné z: http://www.zdrojak.cz/clanky/orm-test-php-frameworku-nette-prado/

[23] NETTE FOUNDATION. *Nette Framework* [online]. 2014 [cit. 2014-05-08]. Dostupné z: http://nette.org/cs/

[24] 6 good reasons to use Symfony. SENSIOLABS. *Symfony* [online]. 2014 [cit. 2014-05-08]. Dostupné z: http://symfony.com/six-good-reasons

[25] The Big Picture (current). SENSIOLABS. *Symfony* [online]. 2014 [cit. 2014-05-08]. Dostupné z: http://symfony.com/doc/current/quick_tour/the_big_picture.html

SEZNAM OBRÁZKŮ

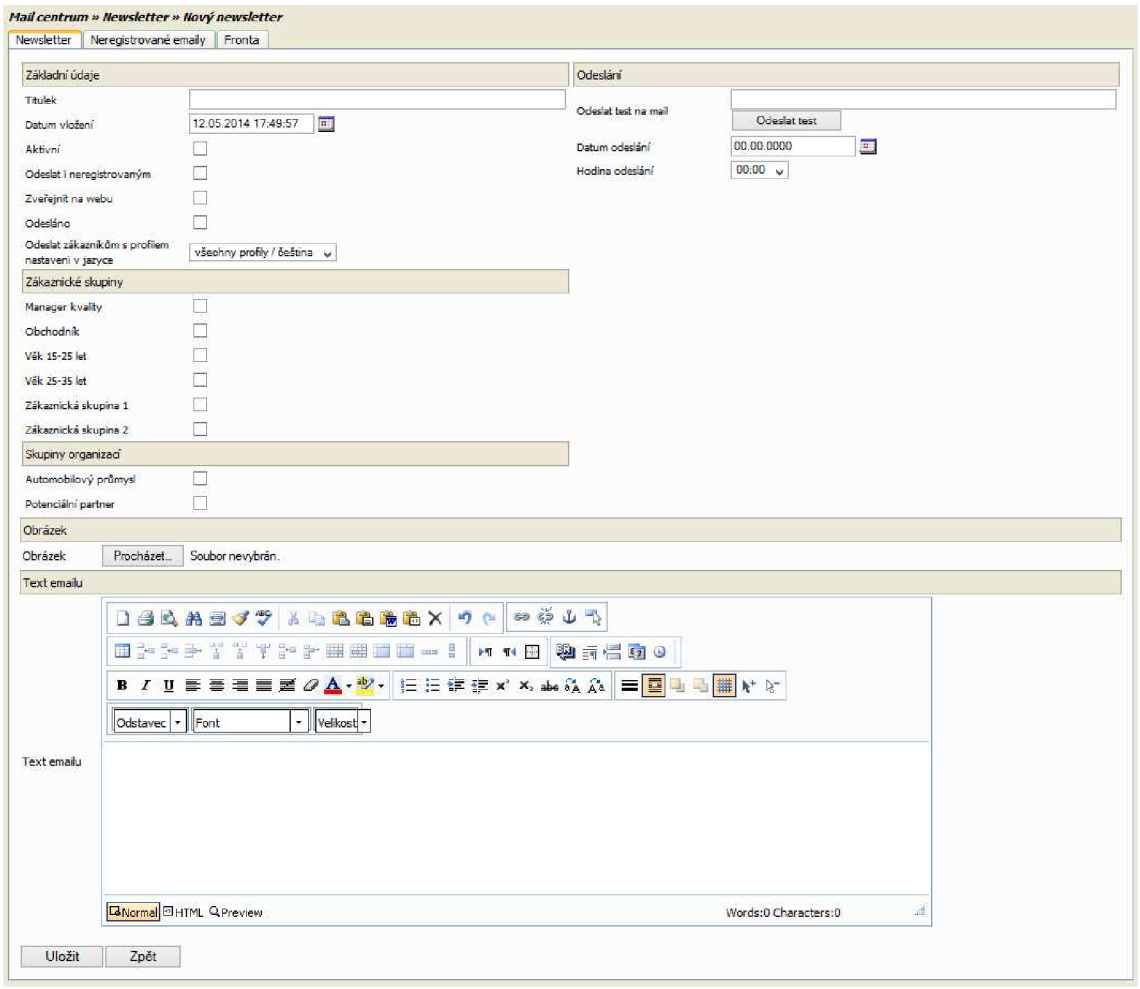
SEZNAM TABULEK

SEZNAM GRAFŮ

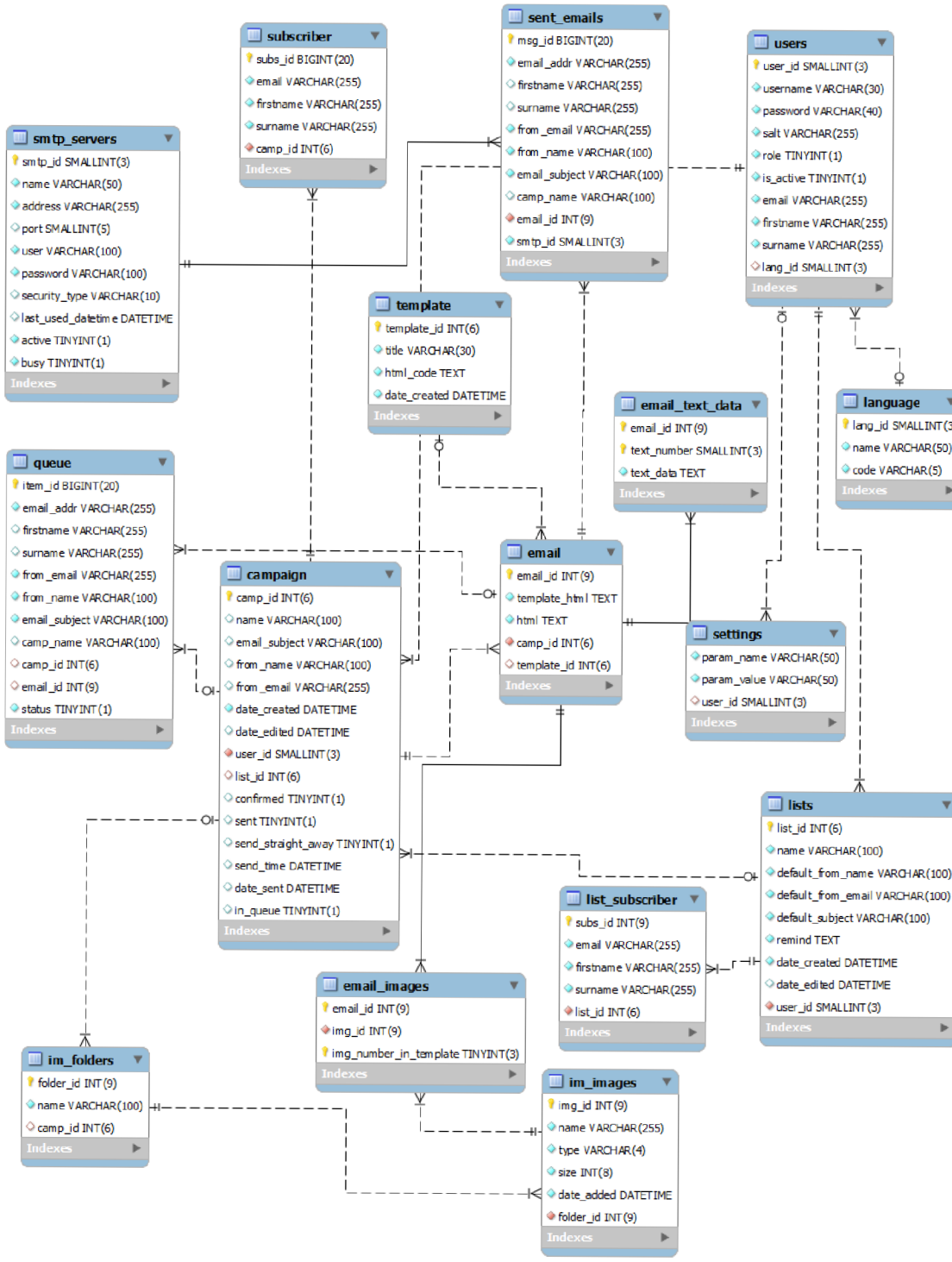
SEZNAM PŘÍLOH

PŘÍLOHY

Příloha 1: Současné řešení firmy pro odesílání newsletterů (zdroj vlastní)



Příloha 2: Databázové schéma aplikace (zdroj vlastní)



Příloha 3: Mapování entity campaigns do objektu Campaign

```
Main\StoreBundle\Entity\Campaign:
    type: entity
    table: campaign
    indexes:
        user_id:
            columns:
                - user_id
    repositoryClass: Main\SystemBundle\Repository\CampaignRepository
    id:
        campId:
            id: true
            type: integer
            column: camp_id
            generator:
                strategy: IDENTITY
    fields:
        name:
            type: string
            length: 100
            nullable: true
        emailSubject:
            type: string
            length: 100
            nullable: true
            column: email_subject
        fromName:
            type: string
            length: 100
            nullable: true
            column: from_name
        fromEmail:
            type: string
            length: 255
            nullable: true
            column: from_email
        dateCreated:
            type: datetime
            nullable: false
            column: date_created
        dateEdited:
            type: datetime
            nullable: true
            column: date_edited
        confirmed:
            type: boolean
            nullable: true
        sent:
            type: boolean
            nullable: true
        sendStraightAway:
            type: boolean
            nullable: true
            column: send_straight_away
        sendTime:
            type: datetime
            nullable: true
            column: send_time
        dateSent:
            type: datetime
            nullable: false
            column: date_sent
        inQueue:
            type: boolean
            nullable: true
            column: in_queue
    manyToOne:
        user:
            targetEntity: Main\StoreBundle\Entity\User
            cascade: {  }
            mappedBy: null
            inversedBy: null
            joinColumns:
                user_id:
                    referencedColumnName: user_id
            orphanRemoval: false
        list:
            targetEntity: Main\StoreBundle\Entity\RList
            cascade: {  }
            mappedBy: null
            inversedBy: null
            joinColumns:
                list_id:
                    referencedColumnName: list_id
            orphanRemoval: false
    lifecycleCallbacks: {  }

```

Příloha 4: Návrh vzhledu systému (zdroj vlastní)

Příloha 5: PO soubory s překlady

```
msgid ""
msgstr ""
"Project-Id-Version: Newsletter\n"
"POT-Creation-Date: \n"
"PO-Revision-Date: \n"
"Last-Translator: \n"
"Language-Team: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"X-Generator: Poedit 1.6.4\n"
"Plural-Forms: nplurals=3; plural=(n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2;\n"
"Language: cs\n"

# Copyright (C) 2014 Newsletter System

# === login
#: login.html.twig
msgid "Username"
msgstr "Uživatel"

#: login.html.twig
msgid "Password"
msgstr "Heslo"

#: login.html.twig
msgid "Login"
msgstr "Přihlásit"

```