UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Webová prezentace reklamní agentury s e-shopem

Martin Šára

Bakalářská práce
2012

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2011/2012**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin Šára**
Osobní číslo:	**I09269**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Webová prezentace reklamní agentury s e-shopem**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je vytvořit webovou prezentaci reklamní agentury obsahující přehledný internetový obchod pro prodej zábavné a reklamní techniky. Aplikace bude zobrazovat produkty v různých kategoriích včetně jejich obrázků, hodnocení a komentářů. Dále umožní obsluhovat nákupní košík, vytvářet objednávky a exportovat je, nabízet různé dodací a platební podmínky, realizovat nákup i bez registrace, zasílat notifikaci e-mailem. Administrátorská část bude obsahovat kompletní správu obchodu, statistiky a možnost exportu dat. Stránky budou optimalizovány pro vyhledávače, ošetřena bude problematika SQL injection.
V teoretická části se autor bude zabývat problematikou PHP frameworků, analýzou a porovnáním několika vybraných. Implementační část bude realizována pomocí PHP, MySQL, CSS stylování a Nette frameworku.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

SCHAFER, Steven M. HTML, XHTML a CSS : Bible pro tvorbu WWW stránek. Vyd. 1. Praha: Grada, 2009. 648 s. ISBN 978-80-247-2850-6.
GROFF, James R.; WEINBERG, Paul N. SQL : kompletní průvodce. Vyd. 1. Brno: CP Books, 2005. 936 s. ISBN 80-251-0369-2.
KUBÍČEK, Michal. Velký průvodce SEO : Jak dosáhnout nejlepších pozic ve vyhledávačích. Vyd. 1. Brno : Computer Press, 2008. 318 s. ISBN 978-80-251-2195-5.
LACKO, Luboslav. PHP a MySQL : Hotová řešení. Vyd. 1. Brno: CP Books, 2005. 299 s. ISBN 80-251-0397-8.

Vedoucí bakalářské práce: **RNDr. Iva Rulicová**
Katedra informačních technologií

Datum zadání bakalářské práce: **16. prosince 2011**
Termín odevzdání bakalářské práce: **11. května 2012**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 30. března 2012

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Poděbradech dne 9. 5. 2012 Martin Šára

Poděkování

Na tomto místě bych rád poděkoval RNDr. Ivě Rulicové za vedení práce a za její rady a připomínky. Dále bych chtěl poděkovat všem, kteří mě podporovali během celého studia a při zpracovávání této práce.

Anotace

Práce se zabývá problematikou frameworků. Podrobněji se zaměřuje na frameworky pro skriptovací jazyk PHP. Je poukázáno na výhody a nevýhody jednotlivých včetně jednoduchého porovnání práce s nimi. Pomocí vybraného frameworku je vytvořena webová prezentace s internetovým obchodem, který nabízí jak základní, tak i některé pokročilé funkce.

Klíčová slova

framework, PHP framework, internetový obchod, e-shop, Nette Framework, Zend Framework, Symfony

Title

Web presentation of advertising agency with e-shop

Annotation

This work deals with frameworks. It focuses on frameworks for PHP scripting language. There are highlighted advantages and disadvantages of each, including a simple comparison of work with them. Using the selected framework is created website with online store that offers both basic and some advanced features.

Keywords

framework, PHP framework, online store, e-shop, Nette Framework, Zend Framework, Symfony

Obsah

Seznam zkratek

API	Application Programming Interface
SQL	Structured Query Language
URL	Uniform Resource Locator
PHP	PHP: Hypertext Preprocessor
HTML	HyperText Markup Language
IIS	Internet Information Server
CGI	Common Gateway Interface
XML	Extensible Markup Language
OOP	Object Oriented Programming
XSS	Cross-Site Scripting
AJAX	Asynchronous JavaScript and XML
KISS	Keep It Simple, Stupid
DRY	Dont't Repeat Yourself
CSRF	Cross-Site Request Forgery
YAML	YAML Ain't Markup Language
ODBC	Open Database Connectivity
PDO	PHP Data Objects
WYSIWYG	What You See Is What You Get

Seznam obrázků

Úvod

Vizitka na Internetu se postupem času stala samozřejmostí téměř každé firmy. Sebelepší firma by bez jakékoliv propagace existovala jen velmi těžko. Kvalitní webová prezentace, která zaujme nejednoho návštěvníka, je významným krokem k úspěchu. Samotný web by na Internetu takřka nic neznamenal – skoro nikdo by jej neznal. Velmi důležitým faktorem úspěšnosti stránek je dobrá optimalizace pro vyhledávače. Právě když někdo něco hledá, nastává ta nejlepší situace dát o sobě vědět.

Obchodování na Internetu zažilo zejména v minulých letech obrovský rozmach. Vznikla řada nových čistě internetových obchodů, ale i spousta kamenných obchodů rozšířila svou působnost na celosvětovou síť. Důvodů je hned několik. Provoz e-shopu téměř nic nestojí – nejsou potřeba žádné drahé prodejny. Zákazníci uvítají pohodlnost nákupu z domova.

Cílem této práce je vytvořit webovou prezentaci s internetovým obchodem tak, aby se jednalo o jeden funkční celek. Důležité je poskytnout potenciálním zákazníkům nejen základní údaje o firmě, ale i podrobné informace o nabízeném sortimentu a jeho problematice. Právě při hledání zmiňovaných informací se naskytuje ideální chvíle pro nabídnutí jednotlivých produktů.

V rámci implementace bude použit velmi rozšířený skriptovací jazyk PHP. Zde se přímo nabízí využít některý framework. Vývoj aplikace se nejen zefektivní, ale automaticky bude ošetřena i bezpečnost. V teoretické části se proto nejprve přiblíží obecná problematika frameworků, poté budou vybrané PHP frameworky analyzovány a porovnány.

1 Framework

Definice pojmu framework není zcela jednoduchá, protože existuje více podobných výkladů. Obecně se jedná o sadu knihoven a nástrojů, které napomáhají při vývoji a údržbě celé řady aplikací. Například česká Wikipedie (2012) hovoří o „softwarové struktuře, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovny API, podporu pro návrhové vzory nebo doporučené postupy při vývoji.“ Naopak anglická Wikipedia (2012) popisuje framework jako „znovupoužitelnou sadu knihoven nebo tříd pro softwarový systém.“ Zatímco jedna knihovna je zaměřena na řešení specifické úlohy, framework má širší záběr. Nabízí komplexní řešení v rámci určité problematiky, čímž zvyšuje komfort při vývoji daných aplikací.

Cílem frameworků je převzít řešení běžně se opakujících problémů a usnadnit tak programátorům práci. Ti pak nemusí opakovaně řešit dílčí úlohy, ale mohou se plně věnovat vývoji dané aplikace.

Softwarových frameworků existuje velké množství. Liší se zejména svým zaměřením, použitými technologiemi a obsáhlostí. Pro některé projekty postačí menší frameworky, které vynikají nižšími nároky; pro jiné se hodí obsáhlé frameworky, které poskytnou veškerou potřebnou funkcionalitu.

1.1 Výhody používání

Používání frameworků přináší mnoho výhod. Jak již bylo zmíněno, osvobozují programátory od řešení běžných dílčích úloh, čímž šetří jejich čas a zvyšují efektivitu práce. Výsledná aplikace je zhotovena za výrazně menší čas, což šetří drahé prostředky.

Řešení dílčích úloh ve frameworku bývá efektivnější, než kdyby se programátor pokoušel danou problematiku řešit sám. Nejen že by ji musel podrobně zkoumat, ale pravděpodobně by bez dřívějších zkušeností ani nevytvořil příliš optimální řešení. Kód frameworku je většinou psán a testován větší skupinou odborníků, kteří se v dané problematice pohybují a rozumí ji. V závislosti na celkové kvalitě frameworku je kód spolehlivý a optimalizovaný pro řešení běžných úloh.

Vývoj softwaru v týmech může být s využitím frameworku snadněji rozdělen. V případě větších, unikátních projektů může část týmu vyvíjet vlastní framework, který poskytne potřebné funkce. Další část týmu využívá samotné rozhraní frameworku k vývoji vlastní aplikace. Zbytek se může soustředit na tvorbu grafického uživatelského rozhraní.

Frameworky většinou poskytují i jistou formu zabezpečení pro aplikace daného typu. Zabezpečení vyvíjené aplikace je ošetřeno právě frameworkem. Opět lze očekávat, že bezpečnost frameworku je na velmi vysoké úrovni, jelikož je zajišťována znalci v oboru.

Například u webových aplikací může být ošetřena autentizace[1] a autorizace[2] uživatelů, může být řešeno escapování[3] řetězců, nebo problematika SQL injection[4].

Další výhodou využívání frameworků je, že vedou vývojáře zavedenými a ověřenými postupy v dané oblasti. Může se jednat například o správu hesel nebo vhodný návrh grafického rozhraní. S tímto úzce souvisí i využívání návrhových vzorů, které pomáhají vést při správném objektovém návrhu aplikace. Dobře navržená aplikace je modulární a znovupoužitelná (DocForge 2011).

V závislosti na typu frameworku by se našly i jiné výhody, výše uvedené patří spíše k obecným. Hlavní výhody frameworků ve zkratce:

- rychlejší vývoj,
- spolehlivý a ověřený kód,
- zabezpečení,
- modularita a znovupoužitelnost,
- vede k nejlepším praktikám,
- podpora práce v týmu.

1.2 Nevýhody používání

Používání frameworků s sebou přináší i některé nevýhody. Částečnou nevýhodou je nutný počáteční čas investovaný pro pochopení fungování frameworku a naučení se s ním pracovat. Zezačátku je tedy vývoj aplikace pomalejší, ovšem s postupem času a s přibývajícími zkušenostmi se produktivita práce zvýší. Toto se týká pouze úplných začátečníků. Pokud má vývojář s daným frameworkem zkušenosti, nebo již s podobným pracoval, nebude zpočátku potřebovat tolik času pro osvojení si potřebných technik.

Další nevýhodou může být jistá degradace výkonu. Ta může být způsobena jednak použitím příliš obecného kódu pro řešení specifické úlohy, jednak režií frameworku samotného – například režií bezpečnostních mechanismů. Celkový výkon frameworku záleží na kvalitě implementace a na použitých technologiích.

Někdy může být potřeba funkcionalita, kterou framework přímo nenabízí. Vývojář pak musí danou funkcionalitu naprogramovat a vhodně ji do systému začlenit. Někdy to může být díky struktuře frameworku poněkud problematická záležitost. Záleží na konkrétních případech a na možnostech rozšiřování daného frameworku. Nejprve je nutné analyzovat požadavky aplikace, a podle toho vybrat vhodný framework, který poskytne potřebné funkce a případně umožní další rozšiřování. Volbou nevhodného frameworku se může vývoj značně zkomplikovat a zpomalit, proto se musí dbát na vhodný výběr.

[1] Autentizace – proces ověření identity.
[2] Autorizace – proces ověření práv.
[3] Escapování – nahrazení znaků se speciálním významem jinými znaky.
[4] SQL injection – technika napadení databázového systému.

Snad poslední nevýhodou je fakt, že i ve frameworku se mohou vyskytovat chyby. Ty se pak objevují ve všech aplikacích, které jsou pomocí daného frameworku vyvíjeny. Může se jednat o menší chyby při zpracovávání různých požadavků. V horším případě se mohou vyskytnout závažnější bezpečnostní rizika. Toto lze eliminovat pouze výběrem kvalitního a ověřeného frameworku. Zvážit by se měla i doba mezi nově vydávanými verzemi a rychlost oprav případných problémů (DocForge 2011).

Každé pro má i své proti, na poli frameworků to platí stejně. Spousta výhod a usnadnění je vykoupena některými omezeními, či nevýhodami. Je na zvážení, zda se vyplatí vyvíjet aplikaci s využitím frameworku, nebo bez něj. Záleží na náročnosti aplikace a možnostech uvažovaného frameworku. Možné nevýhody při používání frameworků jsou tyto:

- snížení výkonu,
- počáteční čas potřebný ke zvládnutí frameworku,
- nepodporovaná funkcionalita,
- možnost výskytu chyby ve frameworku.

1.3 Typy frameworků

Existuje mnoho typů frameworků, které jsou zaměřeny na zcela odlišné oblasti. V základě je lze rozdělit podle způsobu využití. Některé mohou usnadňovat návrh grafického uživatelského rozhraní, jiné mohou být zaměřeny pouze na práci s multimédii, další poskytují komplexnější API pro určitý programovací jazyk.

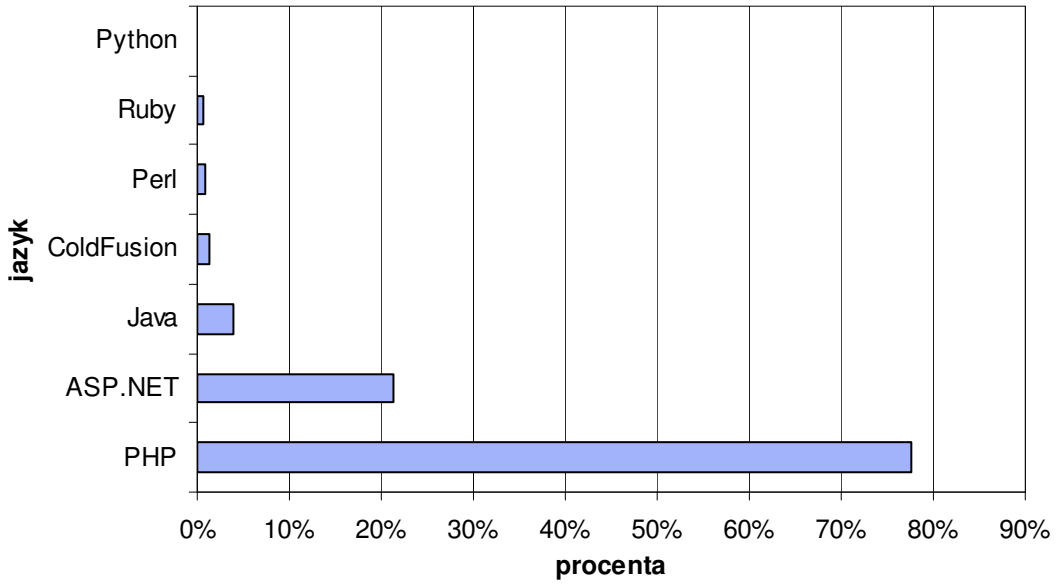
Velkou skupinu tvoří webové aplikační frameworky. Ty jsou určeny pro vývoj dynamických webových aplikací a služeb. Tyto frameworky lze podrobněji dělit podle jejich specializace. Některé jsou přizpůsobeny pouze pro tvorbu redakčních systémů, jiné poskytují komplexní podporu pro vývoj rozsáhlých webových aplikací. Může se jednat jak o frameworky pro skriptování na straně klienta, tak i na straně serveru.

Zaměření této práce je na PHP frameworky, tedy skriptování na straně serveru pomocí skriptovacího jazyka PHP.

2 PHP frameworky

2.1 PHP

PHP je široce používaný skriptovací jazyk pro vývoj dynamických webových aplikací. Název PHP je odvozen z rekurzivní zkratky – PHP: Hypertext Preprocessor. PHP skripty jsou zpracovávány na serveru, výsledky skriptů jsou vkládány do HTML kódu, a ten je poté odeslán uživateli. Jazyk PHP je platformě nezávislý, lze jej provozovat jak na operačních systémech Windows, tak i na systémech založených na Unixu. Obvykle se PHP nasazuje na webový server Apache. Lze jej provozovat i dalších webových serverech, například na IIS[5]. Instalace PHP na webový server je snadná – lze jej provozovat jako modul, nebo prostřednictvím CGI[6]. Snad právě díky jednoduchosti instalace a otevřené licenci je PHP tak rozšířené (PHP: Hypertext Preprocessor 2012a). Využívání PHP v oblasti webových aplikací ilustruje denně aktualizovaná statistika serveru W3Techs (2012), viz obrázek 1.



Obrázek 1 – Využívání skriptovacích jazyků na straně serveru

Výhodou tohoto jazyka je i jeho jednoduchost pro vývojáře, kterou uvítají zejména úplní začátečníci. Během pár hodin jsou schopni se naučit základy jazyka a začít psát první jednoduché skripty. Ovšem i pokročilým programátorům je nabízen dostatek funkcí. Jmenujme širokou podporu databází, práci s obrázky a s textem, podporu pro poštovní protokoly, práci s XML[7] a další. Samozřejmostí je podpora OOP[8], která se s novými verzemi stále zlepšuje (PHP: Hypertext Preprocessor 2012b).

[5] IIS – webový server vyvíjený společností Microsoft pro operační systémy Windows.
[6] CGI – rozhraní pro připojení externích aplikací k webovému serveru.
[7] XML – rozšířitelný značkovací jazyk.

PHP pokrývá téměř všechny oblasti, které mohou být při webových aplikacích potřebné. Poskytuje spoustu elementárních funkcí, jejichž využíváním vývojáři dosáhnou požadovaných výsledků. Zde přichází na řadu frameworky, které poskytují řešení běžných problémů prostřednictvím komplexního API. Navíc mohou pomoci s odhalováním chyb, jelikož PHP je ve výchozím stavu velmi benevolentní jazyk. Další výhody PHP frameworků budou popsány v samostatné kapitole.

2.2 Architektura Model-View-Controller

Se softwarovou architekturou Model-View-Controller (dále jen MVC) se setkáme u většiny webových frameworků. V překladu se jedná o model, pohled (šablonu) a ovladač. Každá z těchto vrstev má na starost pouze určité úlohy. Jednoduše lze definovat jednotlivé vrstvy takto:

- model – datová struktura,
- view – uživatelské rozhraní,
- controller – řídící člen.

Účelem této architektury je oddělit jednotlivé vrstvy aplikace tak, aby na sobě byly co nejméně závislé. Kód se stane přehlednějším a snadněji upravitelným. Například u webových aplikací se nemíchá dohromady HTML s PHP kódem a databázovými dotazy. Úpravy aplikací, které jsou postaveny na MVC, jsou mnohem rychlejší a jednodušší, což ve výsledku vede k úspoře času a financí. Navíc se jednotlivé části mohou použít znovu při vývoji jiných aplikací. Rozdělením na jednotlivé vrstvy se podporuje i práce v týmu. Část vývojářů pracuje na aplikační logice, druhá část se věnuje reprezentaci výsledných dat.

Problematika architektury MVC je poměrně obsáhlá. V závislosti na typu aplikace se využívají různé variace této architektury. Ty se liší zejména vazbami mezi jednotlivými vrstvami a částečně i jejich úlohami. Webové frameworky tuto architekturu často využívají, resp. její konkrétní variace (Bernard 2009). Pro další práce je nutné mít alespoň obecnou představu o architektuře jako o celku.

2.2.1 Model

Model je datovou strukturou, kde se odehrává většina aplikační logiky. V modelu probíhají nejrůznější operace s daty, provádějí se výpočty, komunikuje se s databází. Model nabízí pevně dané rozhraní, které je využíváno ostatními vrstvami. O existenci controlleru a view model neví. Je funkční sám o sobě, a proto je možné stejný model používat u více podobných aplikací.

[8] OOP – objektově orientované programování.

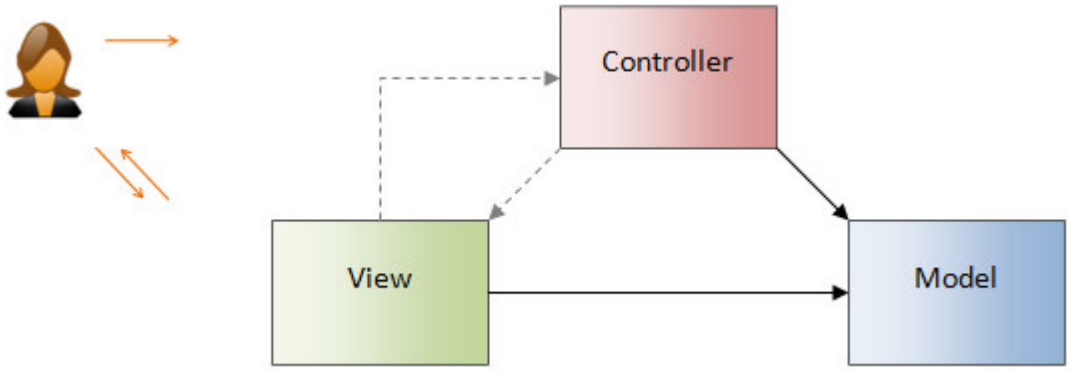
2.2.2 View

View má na starosti vykreslování dat. Většinou se jedná o šablonu, která je naplněna daty, a poté je vykreslena. V oblasti webových frameworků obvykle používá šablonovací systém, který usnadňuje výpis dat a přidává další užitečné funkce. Může se jít o možnost členění šablony do bloků, nástroje pro escapování řetězců, formátování data a času, nebo další pomůcky pro práci s řetězci apod. View je plněn daty z modelu. Zobrazení určitých dat ovlivňuje controller.

2.2.3 Controller

Controller je obecně provazujícím prvkem mezi modelem a view a stará se o celkovou funkčnost aplikace. V závislosti na konkrétních implementacích architektury MVC se úlohy controlleru mírně liší. Může se jednat o řídící část, která přímo zpracovává požadavky od uživatele. V jiné implementaci se o zpracovávání požadavků může starat view. V oblasti webových frameworků se požadavkem rozumí URL adresa. Na základě požadavku zavolá controller příslušnou metodu určitého modelu, a ten zprostředkuje data pro view.

Na obrázku 2 (Bernard 2009) je vidět, jak spolu jednotlivé vrstvy spolupracují, a jak probíhá interakce s uživatelem. Podle různých variací této architektury se vazby mezi jednotlivými vrstvami liší. Problematika MVC je poměrně obsáhlá a předmětem této práce není podrobně popisovat jednotlivá řešení. Pro pochopení fungování webových frameworků je však nutné mít alespoň obecnou představu o této architektuře a znát principy jejího fungování.



Obrázek 2 – Schéma architektury Model-View-Controller

2.3 Přínosy PHP frameworků

Na všem lze něco vylepšovat. I jazyk PHP je hodný spousty vylepšení. Proto vznikla řada frameworků (a stále se objevují nové), které mají za cíl usnadnit práci s PHP. Obecně jmenujme třeba nepřehledné výpisy chyb, žádné ladící nástroje, chybějící zabezpečení proti webovým útokům, ve výchozím stavu velkou volnost při používání proměnných,

pouze základní správu session apod. Tyto všechny oblasti mohou PHP frameworky řešit, a ušetřit tak programátorům práci.

PHP frameworky obvykle přináší následující nástroje a vylepšení:

- šablonovací systém,
  - usnadňuje a zpřehledňuje tvorbu výsledného HTML kódu,
  - zajišťuje escapování proměnných,
  - umožňuje členit šablonu na bloky,
  - podporuje vkládání komponent (samostatných funkčních celků),
- ladící nástroje,
  - zpřehledňují výpis proměnných a chybových hlášení,
  - umožňují měřit paměťovou a časovou náročnost aplikace,
  - obsluhují chyby na produkčním serveru,
- zabezpečení aplikace,
  - ochrana proti útokům, např. XSS[9], CSRF[10], problematika sessions…,
- vrstva pro práci s databází,
  - umožňuje pokládat efektivnější dotazy,
  - podpora cachování výsledků dotazů,
  - ochrana před SQL injection,
- podpora AJAXu[11,]
  - jednoduché zprovoznění AJAXu (i v rámci existujícího kódu),
- optimalizace pro vyhledávače,
  - hezké URL adresy,
  - kanonizace URL [12]adres,
- návrh pomocí MVC,
  - vede k přehledné výstavbě aplikace,
- správa session,
- správa cache,
- komplexní konfigurace celé aplikace,
  - globální nastavení aplikace z jednoho místa,
  - různé konfigurace pro vývojové a produkční prostředí,
- pomoc při lokalizaci aplikace,
- validace formulářů na straně serveru i klienta.

Oblastí, kde mohou frameworky pomoci, je opravdu mnoho. Dobrý PHP framework obsluhuje všechny výše zmíněné okruhy. Použitím vhodného frameworku se řešení výše uvedených problematik usnadní, či dokonce úplně zmizí.

[9] XSS – napadení webové aplikací pomocí neošetřených vstupů.
[10] CSRF – napadení webové aplikací odesláním formuláře z jiné webové stránky.
[11] AJAX – technologie umožňující měnit pouze části webových stránek bez nutnosti jejich znovunačtení.
[12] Kanonizace URL – přesměrování duplicitních stránek s různou URL adresou na jednu výchozí adresu.

2.4 Nasazení

PHP frameworky mohou být využity jak při tvorbě menších webových prezentací, tak při návrhu rozsáhlých webových aplikací. Podle náročnosti a rozsáhlosti je v základu možné rozdělit webové aplikace na:

- jednoduché webové prezentace o pár stránkách,
- složité webové aplikace poskytující mnoho sofistikovaných funkcí.

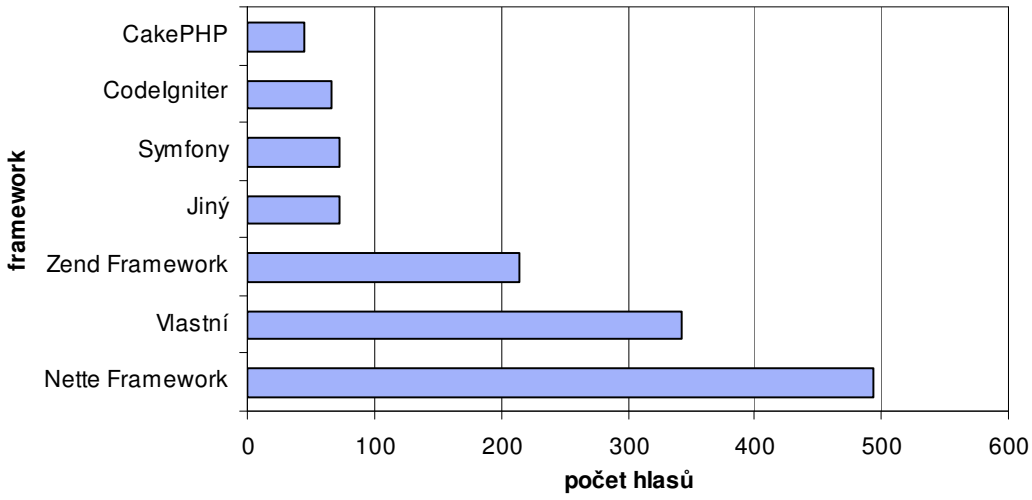
U jednoduchých webových prezentací, např. prezentace mále firmy, se může zdát nasazení frameworku naprosto zbytečné. I zde však framework nachází uplatnění. Může např. pomoci s validací a zabezpečením kontaktního formuláře, nebo při lokalizaci do více jazyků. U takto malých aplikací se nabízí sáhnout po malém a jednoduchém frameworku, který nabídne pouze základní funkce.

Hlavní využití frameworků je u složitých webových aplikací. Jmenujme třeba internetové obchody, systémy pro cestovní kanceláře, blogy, redakční a rezervační systémy, nejrůznější firemní systémy apod. Jak již bylo řečeno výše, usnadní, urychlí a zpřehlední vývoj celé aplikace.

V další kapitole budou podrobně rozebrány vybrané PHP frameworky a bude ukázáno na rozdíly v jejich vlastnostech a používání.

3 Vybrané PHP frameworky

Výběr porovnávaných frameworků byl proveden na základě ankety provedené na přelomu roku 2010/2011 na českém serveru Zdroják (2010). Výsledky ankety byly zejména na předních příčkách jasné, což znázorňuje obrázek 3. Vzhledem k jednoznačným výsledkům lze předpokládat, že se situace v oblasti PHP frameworků v České republice doposud nijak výrazně nezměnila.



Obrázek 3 – Oblíbený / používaný PHP framework

Mezi prvními třemi příčkami je znatelný rozdíl, zbylé pozice jsou téměř vyrovnané. Jasným vítězem se stal český Nette Framework. Na druhém místě dotazovaní volili používání vlastního frameworku. Velká většina vývojářů si podle svých požadavků vytvoří vlastní framework, který jim nejvíce vyhovuje. Na třetím místě se umístil Zend Framework, který je velmi populární nejen u nás.

Analyzován tedy bude Nette Framework, Zend Framework a framework Symfony.

3.1 Nette Framework

Nette Framework patří k nejpopulárnějším frameworkům v České republice. Podle výše zmíněné ankety je využíván naprostou většinou dotazovaných vývojářů. Framework byl založen v roce 2004 českým podnikatelem a programátorem Davidem Grudlem. Pro veřejnost byl ovšem uvolněn až v roce 2008. Od roku 2009 je framework vyvíjen skupinou jednotlivců sdruženou v Nette Foundation (2012), hlavním vývojářem však stále zůstává jeho zakladatel. Framework je šířen jako svobodný software a je dostupný buď pod licencí New BSD[13], nebo GNU GPL[14]. Mezi hlavní důvody stále větší popularity frameworku patří tyto:

[13] http://nette.org/cs/license#toc-new-bsd-license
[14] http://nette.org/cs/license#toc-gnu-general-public-license-gpl

- jedná se o výhradně český projekt,
- kvalitní framework s velkou aktivní komunitou.

3.1.1 Přednosti

Důraz je kladen zejména na dokonalé zabezpečení vyvíjených aplikací, jejich rychlost a jednoduchost. Framework se drží principů KISS a DRY, tedy snaží se udržovat kód jednoduchý a krátký, vede programátory k tomu, aby se neopakovaly.

- Bezpečnost

  Framework ochraňuje před všemi běžnými útoky na webové aplikace. K ošetření útoků XSS využívá technologii Content-Aware Escaping, která je součástí výchozího šablonovacího systému Latte. Technologie Content-Aware Escaping detekuje, v jakém kontextu je kód vypisován, a podle něj správně escapuje všechny speciální znaky. Tato technologie automaticky ošetřuje všechna nebezpečná místa v aplikaci, tudíž vývojáři se o tuto problematiku nemusí vůbec starat.

  Pro ukázku:

  - naplnění proměnné *data*:

```
$this->template->data = '<script>alert("XSS");</script>';
```

  - výpis proměnné *data* v různých kontextech (JavaScript a HTML):

```
<script>alert({$data});</script>
{$data}
```

  - obsah proměnné data byl zcela automaticky a správně escapován podle kontextu:

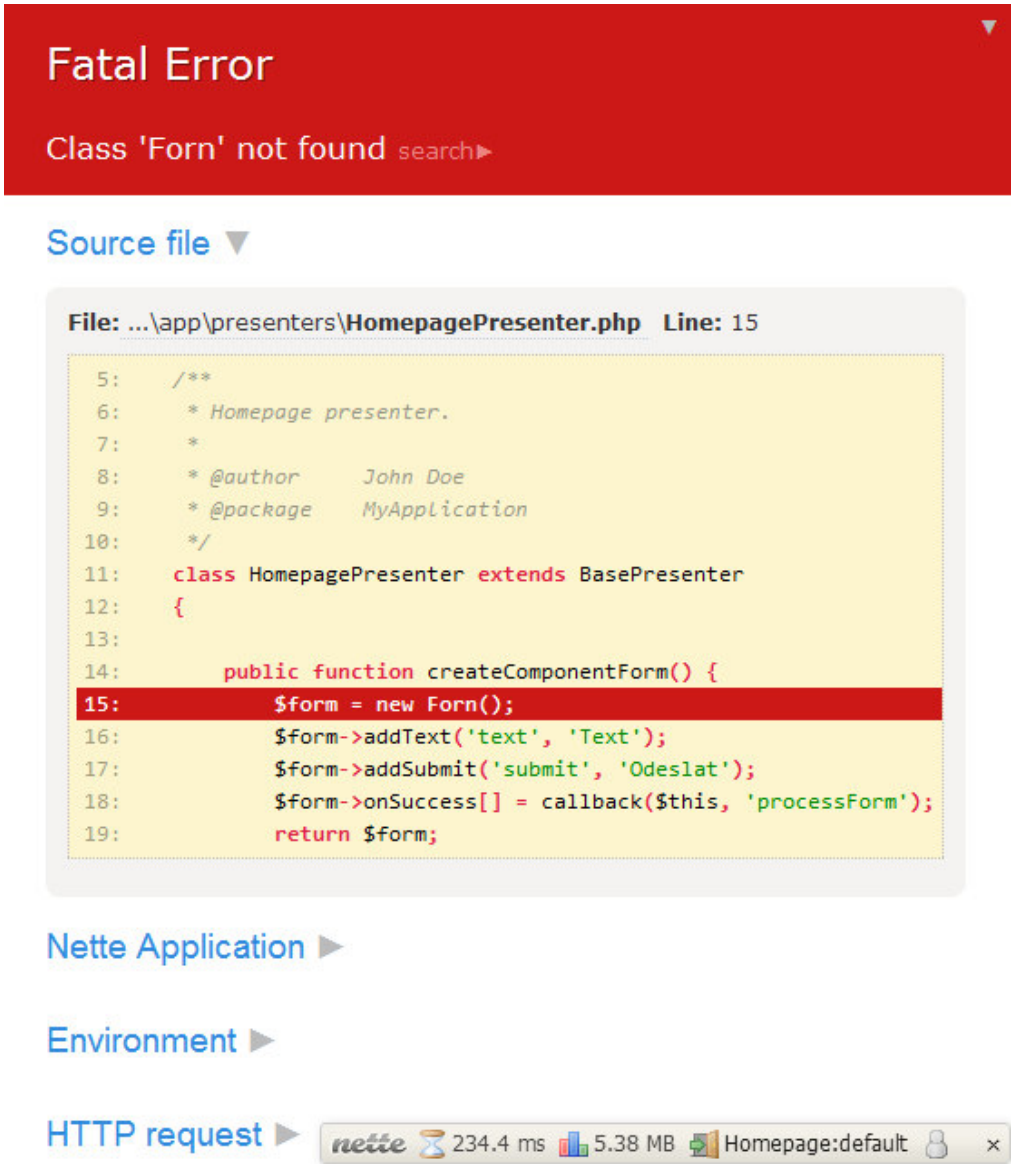
    JavaScript: `<script>alert("<script>alert(\"XSS\");<\/script>");</script>`
    HTML: `<script>alert("XSS");</script>`

  Proti útokům CSRF nabízí framework také jednoduché řešení. Slouží k tomu jednoduchá metoda `addProtection()`, které automaticky zabezpečí požadovaný formulář. V oblasti zabezpečení session framework také pomáhá, a to správným nastavením PHP.

- Ladící nástroje

  Samotné PHP není příliš přívětivé při výpisu chyb. K dispozici je nástroj, pro nějž se zažil název Laděnka (obrázek 4). Ta dokáže přehledně zobrazit řádek s chybou přímo v původním zdrojovém kódu. Při kliknutí na název souboru otevře daný soubor ve vývojovém prostředí a přejde automaticky na příslušný řádek. Laděnka varuje i při překlepech a používání neinicializovaných proměnných. Dále umožňuje okamžitě vyhledávat danou chybu na Internetu.

Další vymožeností je zobrazení obsahu všech lokálních proměnných a automatické logování chyb na produkčním serveru. Lze také zobrazit parametry aplikace, HTTP požadavku a další informace.



Obrázek 4 – Chybové hlášení v Nette Frameworku

Pro výše zmíněnou funkčnost je třeba, aby všechny třídy byly děděny od třídy Nette\Object. Tak jako většinu částí frameworku, lze i Laděnku používat zcela samostatně či v jiném frameworku.

- Výkon

  Dle nezávislého testu provedeného Petrem Daňkem (2008) je Nette Framework nejrychlejším frameworkem vůbec. Podle výsledků bylo rychlejší pouze použití samotného PHP. Není se čemu divit, minifikovaná verze Nette Frameworku do

jednoho souboru zabírá okolo 500 kB. Paměťové nároky jsou také výborné, oproti konkurenčnímu Zend Frameworku zabere výrazně méně paměti.

- Komunita

  Nette Framework je proslulý svou rozsáhlou komunitou, která je obrovskou oporou při vývoji. Komunita se částečně podílí i na vývoji frameworku. Pro začátečníky je obsáhlé fórum dobrou základnou pro řešení všech problémů. Komunita je opravdu velmi pružná, což prokazuje i aktivita na zmiňovaném fóru. Dále se konají setkání příznivců a vývojářů Nette Frameworku – Poslední Soboty. Většinou se prezentují nové technologie a funkce, řeší se problémy, ukazují se vzorová řešení, jedná se o vývoji samotného frameworku.

- Rozšíření

  Framework obsahuje pouze základní nástroje a pomůcky, které jsou využívané při vývoji většiny aplikací. Výsledné aplikace tedy nejsou brzděny zcela nepotřebnými knihovnami a rozšířeními. Framework si tak udržuje svoji lehkost a rychlost. Při potřebě dalšího nástroje či funkcionality je možné snadno využít některou komponentu. K dispozici je spousta doplňků a komponent, které jsou vytvářeny a aktualizovány zejména zmiňovanou komunitou.

3.1.2 Zápory

Malým záporem může být fakt, že Nette Framework byl založen jednotlivcem, a předně je jím i nadále vyvíjen. Za frameworkem tedy nestojí firma s více hlavními vývojáři. V praxi je však přirozené, že je projekt z velké části veden pouze několika málo předními vývojáři (Grudl 2009). Situaci navíc uklidňuje obrovská komunita, která by framework nenechala jen tak zmizet.

Hlavní nevýhoda tohoto frameworku spočívá v jeho dokumentaci, která určitě odradila nejednoho začátečníka. Ještě před půl rokem byla dokumentace nekompletní a rozepsaná na půl – pro starou a novou verzi. Člověk byl nucen dohledávat spoustu informací na fóru, čímž ztrácel spoustu času. Situace se s postupem času stále zlepšuje. Celá dokumentace se začala přepisovat a ujednocovat. Dnes už je na výrazně vyšší úrovni a stále se na ní pracuje.

3.1.3 Požadavky

Nette Framework v současné verzi vyžaduje alespoň PHP 5.2. Tato verze PHP ještě nepodporuje jmenné prostory. K dispozici je verze s prefixy (třídy frameworku začínají písmenem N), nebo verze bez prefixů (zde však může docházek ke jmenným konfliktům). Pro PHP 5.3 a vyšší je dostupná verze s plnou podporou jmenných prostorů.

Pro běh frameworku je vyžadována i jistá konfigurace webového serveru, která je podrobněji popsána na http://doc.nette.org/cs/requirements. Na otestování konfigurace serveru je dostupný nástroj Requirements Checker, který je standardní součástí stahovaného balíku.

3.1.4 Vývoj

Poslední stabilní verze 2.0.4 byla uvolněna 4. 4. 2012. Vývojový cyklus není příliš pravidelný. To lze požadovat za klad i zápor. Negativem je, že se na novou stabilní verzi může čekat i delší dobu. Naopak za pozitivní stránku věci lze brát fakt, že jsou uvolňovány pouze dotažené a vyladěné verze. Cílem není za každou cenu představit veřejnosti novou, sebeméně nedodělanou verzi.

Okomentovaná ukázka tvorby jednoduchého formuláře a zobrazení dat z něj se nachází v příloze A.

3.2 Zend Framework

Zend Framework je celosvětově známý a patří k nejpoužívanějším PHP frameworkům vůbec. Byl představen v roce 2005 na první Zend Conference. Za jeho vývojem stojí firma Zend Technologies, která byla založena v roce 1997 dvěma izraelskými programátory. Jejich jména jsou Zeev Suraski a Andi Gutmans. V současnosti se na vývoji podílí čtyři hlavní vývojáři (Grudl 2009). Firma Zend Technologies se také podílí na vývoji samotného PHP, dále vyvíjí např. Zend Studio (vývojové prostředí pro PHP) a Zend Server (webový aplikační server). Zend Framework je uvolňován pod licencí New BSD[15] (Zend: The PHP Company 2010). Důvody velké popularity tohoto frameworku jsou následující:

- framework je velice rozsáhlý a nabízí spoustu možností,
- za vývojem stojí firma, která nabízí i další podpůrné produkty a také podporuje vývoj jazyka PHP.

3.2.1 Přednosti

Zend Framework je zaměřen na tvorbu spolehlivých a moderních webových aplikací a služeb. Zaměřuje se i na práci se široce dostupnými API, např. od společnosti Google, Amazon, Yahoo!, Flickr a další. Framework dbá na rozšiřitelný a dobře testovaný kód a na flexibilní architekturu (Zend Framework 2012a).

- Rozsáhlost

  Svým rozsahem nemá framework konkurenci. V porovnání s ostatními PHP frameworky se opravdu jedná o obra. Pokrývá všechny potřebné oblasti při

[15] http://framework.zend.com/license

vývoji webových aplikací a služeb. Tyto oblasti jsou řešeny pomocí jednotlivých komponent frameworku. Jmenujme např. komponenty pro vývoj pomocí návrhového vzoru MVC, komponenty pro práci s databází, dále komponenty pro autentizaci a autorizaci, lokalizaci, správu sessions a vyhledávání. Výčet všech komponent je dostupný na adrese http://framework.zend.com/about/components.

- Firemní základna

  Narozdíl od většiny frameworků stojí za Zend Frameworkem firma. Hlavní vývojáři jsou placeni, což je velký rozdíl od podobných projektů (Grudl 2009). Díky firemní základně a placeným vývojářům je postaráno o průběžný vývoj a opravy. Jistou výhodou jsou i další produkty firmy. Jedná se zejména o Zend Server a Zend Studio, s kterými je Zend Framework úzce svázán.

- Dokumentace a literatura

  Framework má velmi kvalitní a obsáhlou dokumentaci, která je základem pro masové rozšíření podobných projektů. Díky celosvětové popularitě je dokumentace dostupná ve více jazykových mutacích. Další předností, která přispívá ke zmiňované popularitě, je dostupnost literatury. I na českém trhu se najdou publikace, které se Zend Frameworkem zabývají. Jmenujme třeba Zend Framework: Programujeme webové aplikace v PHP od Mariana Böhmera.

3.2.2 Zápory

Výchozí šablonovací systém Zend_View automaticky neescapuje proměnné. Pro bezpečný výpis proměnné je nutné výpis provádět pomocí `$this->escape(promenna)`. Stačí tedy jediné opomenutí a bezpečnostní díra je na světě. Navíc escapování je správné pouze v HTML kontextu (viz ukázka), pro správné escapování v rámci JavaScriptu se musí naprogramovat další metody. Další nevýhodou výchozího šablonovacího systému je jeho nepřehlednost. Výpis každé proměnné musí začínat kódem `<?php` a končit `?>`, dále je nutné použít funkci `echo` a navíc ještě funkci `escape`, viz následující zdrojový kód (Zend Framework 2012b):

```
1 <?php if ($this->books): ?>
2
3   <!-- A table of some books. -->
4   <table>
5     <tr>
6       <th>Author</th>
7       <th>Title</th>
8     </tr>
9
10    <?php foreach ($this->books as $key => $val): ?>
11    <tr>
12      <td><?php echo $this->escape($val['author']) ?></td>
13      <td><?php echo $this->escape($val['title']) ?></td>
14    </tr>
15    <?php endforeach; ?>
```

```
16
17    </table>
18
19 <?php else: ?>
20
21    <p>There are no books to display.</p>
22
23 <?php endif;?>
```

Pro znázornění je uveden totožný kód i v Nette Frameworku:

```
1  {if $books}
2
3     <!-- A table of some books. -->
4     <table>
5        <tr>
6           <th>Author</th>
7           <th>Title</th>
8        </tr>
9
10       <tr n:foreach="$books as $key => $val">
11          <td>{$val['author']}</td>
12          <td>{$val['title']}</td>
13       </tr>
14
15    </table>
16
17 {else}
18
19    <p>There are no books to display.</p>
20
21 {/if}
```

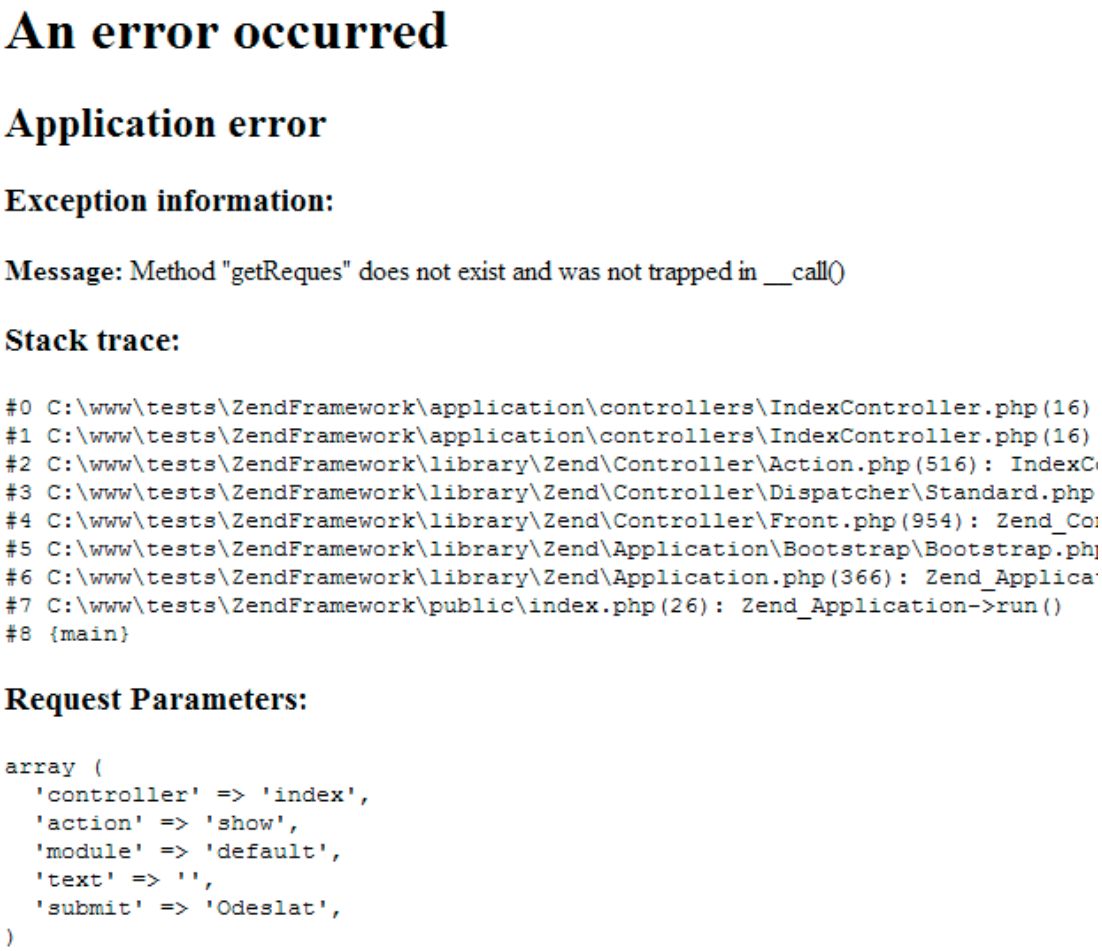
Výchozí šablonovací systém Zend Frameworku lze samozřejmě nahradit – třeba populárním šablonovacím systémem Smarty[16].

Vizualizace chyb také není na tak vysoké úrovni, jako například u konkurenčního Nette Frameworku. Zend Framework zobrazí podle typu chyby buď pouze strohý text:

> **Fatal error**: Class 'Application_Form_Tes' not found
> in **...\IndexController.php** on line **8**

případně trochu podrobnější popis chyby (obrázek 5).

[16] http://www.smarty.net/about_smarty



An error occurred

Application error

Exception information:

Message: Method "getReques" does not exist and was not trapped in __call()

Stack trace:

```
#0 C:\www\tests\ZendFramework\application\controllers\IndexController.php(16)
#1 C:\www\tests\ZendFramework\application\controllers\IndexController.php(16)
#2 C:\www\tests\ZendFramework\library\Zend\Controller\Action.php(516): IndexC
#3 C:\www\tests\ZendFramework\library\Zend\Controller\Dispatcher\Standard.php
#4 C:\www\tests\ZendFramework\library\Zend\Controller\Front.php(954): Zend_Co
#5 C:\www\tests\ZendFramework\library\Zend\Application\Bootstrap\Bootstrap.ph
#6 C:\www\tests\ZendFramework\library\Zend\Application.php(366): Zend_Applica
#7 C:\www\tests\ZendFramework\public\index.php(26): Zend_Application->run()
#8 {main}
```

Request Parameters:

```
array (
  'controller' => 'index',
  'action' => 'show',
  'module' => 'default',
  'text' => '',
  'submit' => 'Odeslat',
)
```

Obrázek 5 – Chybové hlášení v Zend Frameworku

Laděnka v Nette Frameworku přináší mnohem více informací a větší komfort, viz obrázek 4 v kapitole o Nette Frameworku.

Mírně nepříjemné mohou být i někdy velmi dlouhé názvy tříd, např:

```
class Zend_Application_Bootstrap_Bootstrap extends
Zend_Application_Bootstrap_BootstrapAbstract
```

Ušetřit psaní mohou nástroje pro příkazovou řádku. Ty dokážou názvy tříd automaticky generovat, doplňovat jednotlivé akce do controlleru a automaticky vytvářet příslušné soubory. Problém s dlouhými názvy by měl být vyřešen přechodem na novější verzi PHP, která již podporuje jmenné prostory.

3.2.3 Požadavky

K provozu Zend Frameworku je vyžadováno PHP 5.2.4 nebo vyšší. Zend Framework 2 bude vyžadovat alespoň PHP 5.3.3. Další požadavky na určitá PHP rozšíření závisí na

použitých komponentách. Na stránkách s požadavky[17] lze najít závislosti jednotlivých komponent a využívaná rozšíření.

3.2.4 Vývoj

Poslední stabilní verze 1.11.11 byla uvolněna 29. 9. 2011. Vývoj se však již v dnešní době soustředí na nový Zend Framework 2, který přináší spoustu nových věcí a částečně odstraňuje výše zmiňovaná negativa. K dispozici je zatím verze 2.0.0beta3 vydaná 2. 3. 2012. V práci tato verze nebyla uvažována, protože je stále ve fázi vývoje. Jelikož je framework vydáván firmou, kde za hlavním vývojem stojí placení programátoři, vývoj by měl probíhat stálým plynulým tempem.

Okomentovaná ukázka tvorby jednoduchého formuláře a zobrazení dat z něj se nachází v příloze B.

3.3 Symfony

Framework Symfony byl původně koncipován pro vývoj webových aplikací pro zákazníky francouzské firmy Sensio Labs. Veřejnosti byl publikován v 2005 pod licencí MIT[18]. Dnes patří k vedoucím PHP frameworkům. Vývoj frameworku je podporován firmou Sensio Labs a velkou komunitou. Symfony klade důraz na co nejmenší psaní kódu. Velká část kódu využívá konfigurační soubory, které jsou ukládány ve formátu YAML[19]. Od verze 2 je možné využívat i formát XML, PHP pole či anotace. Framework využívá řadu dalších projektů, jmenujme třeba knihovny pro práci s databází (Propel, Doctrine), testování pomocí PHPUnit, nebo šablonovací systém Twig (Symfony 2012a). Symfony vyniká:

- konfiguračními možnostmi,
- modularitou.

3.3.1 Přednosti

Symfony není pouze PHP framework, ale rozsáhlý projekt včetně vlastní filozofie a komunity. Vývojáři nejsou uvězňováni v prostředí Symfony, nejsou nuceni využívat předem poskytnuté technologie. Naopak, framework dává volnost pro využívání jiných knihoven a projektů. Symfony vše zastřešuje a sjednocuje.

[17] http://framework.zend.com/manual/en/requirements.introduction.html
[18] http://symfony.com/doc/current/contributing/code/license.html
[19] YAML – formát pro ukládání dat.

- Modularita

  Celá aplikace se skládá z částí nazývaných bundle. Jedná se strukturovanou sadu souborů (PHP, CSS, JavaScript, obrázky apod.), která řeší jednu určitou problematiku, např. blog, nebo fórum. Každý bundle disponuje vlastní konfigurací a je samostatným funkčním celkem. Takovéto části jsou snadno sdíleny s dalšími vývojáři.

- Konfigurace

  Symfony vyniká rozsáhlými možnostmi konfigurace. V podstatě se jedná o deklarativní programování. Po osvojení tohoto způsobu práce a konfiguračních možností roste celková produktivita (Stoupa 2008). Konfiguraci je možné psát v několika různých formátech, přičemž hlavním je formát YAML.

- Ladící nástroje

  Dobré ladící nástroje jsou základem rychlého vývoje. Symfony nabízí přehledné zobrazování chyb podobné Laděnce z Nette Frameworku. Propracovaný je také Symfony profiler, kde lze zobrazovat konfiguraci, informace o aktuálním požadavku, výjimky, logy, události, databázové dotazy a další informace. Kvalitou ladících nástrojů se řadí mezi Nette Framework a Zend Framework.

- Bezpečnost

  V Symfony se automaticky escapují proměnné, tudíž nehrozí napadení aplikace pomocí XSS útoku. Všechny formuláře jsou automaticky opatřovány kontrolním klíčem proti CSRF útoku.

3.3.2 Zápory

Framework vyznává poněkud odlišnou filozofii než většina PHP frameworků, na což je nutné si v začátcích zvykat. Pro začátečníky může být rozsáhlá konfigurace matoucí. Vytváření aplikací pomocí bundlů také znesnadňuje počáteční kroky. Z těchto způsobu se však postupem času stávají hlavní výhody.

Výchozí šablonovací systém neušetří moc psaní – zejména při tvorbě odkazů a vykreslování formulářů. Kód controlleru by také mohl být kratší. Předávání proměnných do šablony je poměrně zdlouhavé.

3.3.3 Požadavky

Symfony vyžaduje pro svůj běh alespoň PHP ve verzi 5.3.2. Straší verze s dlouhodobou podporou (končí v listopadu 2012) vyžaduje PHP 5.2.4 nebo vyšší. V distribuci je obsažen nástroj, který otestuje konfiguraci a nabídne i případná doporučení. Přehledně vypíše, kde

se má co nastavit. Dále také umožňuje pomocí webového rozhraní nastavit parametry aplikace, např. databázové připojení.

3.3.4 Vývoj

Aktuální verze 2.0.13 byla uvolněna 30. 4. 2012. Nové verze s opravami a novými funkcemi jsou vydávány v pravidelných intervalech, přibližně každý měsíc. Framework tedy opravdu žije. Za vývojem stojí zmiňovaná firma Sensio Labs, která existuje již přes 10 let (Symfony 2012b). Nic nenasvědčuje tomu, že by měl vývoj stagnovat, či být dokonce ukončen.

Okomentovaná ukázka tvorby jednoduchého formuláře a zobrazení dat z něj se nachází v příloze C.

3.4 Shrnutí

PHP frameworků je opravdu mnoho a vybrat si mezi nimi není vůbec jednoduché. Každý framework má jiné přednosti a jiná negativa. Nelze jednoznačně říci: „Tento framework je nejlepší ze všech a ostatní jsou k ničemu.“ Některý se hodí zejména na menší projekty, jiný si zakládá na maximální bezpečnosti, další pokrývá všechny potřebné oblasti. Pro každou aplikaci se lépe hodí jiný framework. Záleží také na vývojáři, jaký styl práce mu je osobní a jaké má preference.

4 Implementace

4.1 Webová prezentace s internetovým obchodem

S rozvojem Internetu se webová prezentace stala nedílnou součástí téměř všech firem. Každý, kdo chce být úspěšný, musí o sobě dát vědět. Tak, jako je důležitá propagace firmy v reálném světě, je důležitá propagace i ve světě virtuálním – Internetu. Základem je kvalitní a důvěryhodná webová prezentace, která dokáže zaujmout a přilákat nové zákazníky. Potencionální zákazníky je třeba nějakým způsobem oslovit. Ideální příležitost nastává v situaci, kdy člověk něco hledá. Proto je velmi důležité, aby byla webová prezentace optimalizována pro internetové vyhledávače. Čím lepší umístění ve vyhledávači, tím více potencionálních zákazníků. K další propagaci se nabízí i využití služeb nejrůznějších reklamních systémů.

Postupem času se stalo nakupování na Internetu velkým hitem. Dnes se dá prostřednictvím Internetu nakoupit téměř cokoli – od šroubku, přes zahraniční dovolenou, až po snídani. Internetový obchod přináší spoustu výhod. Majitel nemusí vlastnit žádnou prodejnu. Zákazník může provést nákup přímo z pohodlí domova.

Cílem této práce je vytvořit systém, ve kterém se webová prezentace a internetový obchod navzájem doplňují. Nejsou pevně dány hranice. Výhod má toto řešení několik. Zákazník se stále pohybuje v jednom stejném rozhraní, čímž si získáváme jeho důvěru. Velmi důležité je poskytnou i informace o prodávaných produktech a dané problematice. Potenciálnímu zákazníkovi se při hledání informací současně zobrazují i nabízené produkty, čímž se zvyšuje šance nákupu. Toto je řešeno tak, že každá kategorie s produkty může mít úvodní slovo – popis. Jednotlivé kategorie tedy slouží k popisu dané problematiky a zároveň k možnosti nákupu nabízených produktů.

4.2 Databázová vrstva

Databázová část aplikace využívá relační databázový systém MySQL. Jedná se o multiplatformní řešení, které lze provozovat nejen na operačních systémech Windows, ale i na systémech založených na Unixu. Na poli databázových systémů se jedná o velmi populární řešení. MySQL nabízí vysoký výkon, spolehlivost a snadnost používání. Většinou se nasazuje v kombinaci Linux, Apache, MySQL a PHP (LAMP). Právě k popularitě přispívá i licence. MySQL je dostupné buď pod licencí GNU GPL, nebo pod komerční licencí (MySQL 2012).

MySQL umožňuje pracovat s uloženými procedurami, triggery a pohledy. Databázový systém MySQL lze provozovat na různých úložných enginech. V závislosti na typu úložného enginu jsou k dispozici různé funkce. Může se jednat například o podporu transakcí nebo fulltextového vyhledávání.

V práci je využit úložný engine InnoDB a MyISAM. Naprostá většina tabulek využívá úložný engine InnoDB, který podporuje transakce. Engine MyISAM je využit pro fulltextové vyhledávání.

4.2.1 SQL injection

SQL injection je technika napadení databázového systému, která využívá pozměněné databázové dotazy. Princip této techniky spočívá v podstrčení upraveného SQL dotazu do neošetřeného vstupu aplikace. Při úspěšném SQL injection útoku může útočník získat například přístupová práva do aplikace, může vypisovat obsahy tabulek, aktualizovat je, nebo dokonce tabulky mazat. Výše zmíněné možnosti závisí na konkrétních dotazech a databázových oprávněních.

Pro názornost je uveden jednoduchý příklad. Mějme tabulku *uzivatele* obsahující informace o uživatelích. Při přihlašování bude prováděn dotaz:

```
SELECT * FROM uzivatele WHERE email='$email';
```

Zadá-li útočník do neošetřeného vstupního pole pro email řetězec:

```
' OR 1=1; --
```

Vznikne dotaz vypadající následovně:

```
SELECT * FROM uzivatele WHERE email='' OR 1=1; --';
```

Podmínka 1=1 je splněna pro všechny řádky tabulky, tudíž jsou všechny vypsány. Dvě pomlčky za středníkem slouží k ignorování zbytku dotazu.

Obranou proti SQL injection je důkladné escapování speciálních znaků používaných v SQL dotazech. Je nutné ošetřit všechny vstupní body aplikace – zejména formuláře a zpracovávané URL adresy. Jediným opomenutím vznikne v celé aplikaci bezpečnostní díra. Pokud bude útočníkem objevena, mohou být způsobeny fatální škody v aplikaci a odcizeny citlivé údaje.

Další částečnou obranou je omezit databázová práva aplikaci. Většina aplikací nepotřebuje mazat celé tabulky, a proto se doporučuje tyto dotazy aplikaci úplně zakázat. Pokud bude někde v aplikaci zapomenut neošetřený vstup, omezením práv se většinou nezabraná úniku informací, ale alespoň se může předejít jejich ztrátě.

PHP nabízí funkce pro escapování řetězců pro většinu databázových systémů. Ošetřovat každý vstup pomocí speciálních funkcí je nejen pracné, ale i nějaký vstup může být opomenut. Pohodlnějším, efektivnějším a bezpečnějším řešením je využívat databázovou vrstvu, která se o vše postará sama. Většina frameworků disponuje vrstvou pro práci s databází. K dispozici jsou i samostatné databázové vrstvy použitelné i mimo frameworky.

4.2.2 Knihovna dibi

Databázová vrstva dibi vznikla za účelem maximálně zjednodušit a zefektivnit práci s databází. O vývoj se stará Nette Foundation, které stojí i za vývojem výše zmiňovaného Nette Frameworku. Dibi podporuje práci s databázemi MySQL, PostgreSQL, SQLite, MS SQL, Oracle a Access. Přistupovat lze i pomocí rozhraní PDO nebo ODBC. Hlavní cíle a výhody používání jsou:

- jednoduchý a přehledný zápis SQL dotazů,
- řešení SQL injection,
- automatické formátování data, času a řetězců,
- řešení rutinních úkonů,
- sjednocení funkcí pro více databázových systémů.

K připojení k databázi stačí zadat pouze přístupové údaje a typ databáze. Po připojení se se všemi podporovanými databázemi pracuje téměř stejně. Při změně databázového systému pak není prakticky nutné do kódu vůbec zasahovat. Dibi automaticky řeší formátování SQL dotazů a escapování řetězců, což usnadňuje programátorovi práci a zvyšuje bezpečnost aplikace. Formátování řetězců je řešeno pomocí několika modifikátorů, podle kterých se řetězec automaticky upraví. Dibi nabízí i pokročilou práci s poli, která výrazně usnadňuje a zpřehledňuje SQL dotazy. Další užitečnou funkcionalitou jsou skládané a podmíněné dotazy, které se sestaví až za běhu aplikace, podle určitých podmínek. Pro získávání výsledků se nabízí řada funkcí, které umožňují načtení výsledku např. do asociativního pole, indexovaného pole, či načtení pouze prvního pole výsledku. Pro získání pouze určité části výsledku se hodí využití parametrů *limit* a *offset*. Samozřejmostí jsou nástroje pro ladění a logování dotazů (Dibi is Database Abstraction Library for PHP 5 2012).

Přestože je aplikace postavena na Nette Frameworku, pro prácí s databází využívá knihovnu dibi. V době vývoje aplikace byla databázová vrstva ve frameworku ještě v ranném vývoji a její používání bylo pouze experimentální.

Výsledky náročných a častých dotazů jsou ukládány do paměti cache, která je spravována frameworkem. Pro ještě efektivnější práci byly vytvořeny následující pomocné funkce:

```
protected function loadCache($namespace, $key) {
   $cache = new Cache($this->context->cacheStorage, $namespace);
   return $cache->load($key);
}

protected function saveCache($namespace, $key, $data) {
   $cache = new Cache($this->context->cacheStorage, $namespace);
   $cache->save($key, $data, array(Cache::TAGS => array($namespace)));
}

protected function cleanCache($namespace, $key = NULL) {
   $cache = new Cache($this->context->cacheStorage, $namespace);
```

```
    if ($key) {
        $cache->remove($key);
    } else {
        $cache->clean(array(Cache::TAGS => array($namespace)));
    }
}
```

Paměť cache je rozdělena do jmenných prostorů, aby nedocházelo ke konfliktům při ukládání dat. První dvě výše uvedené funkce se starají o načítání a ukládání dat. Třetí funkce umožňuje smazat pouze určitý klíč, nebo celý jmenný prostor.

4.2.3 Normální formy

Normální formy tabulek slouží ke kvalitnějšímu návrhu databázových systémů. Práce s dobře navrženými tabulkami je snadnější a optimálnější. Informace nejsou ukládány nadbytečně a jsou dostupné atomicky. Čím je tabulka ve vyšší normální formě, tím je kvalitněji navržena.

- Nultá normální forma

  „Tabulka je v nulté normální formě právě tehdy, existuje-li alespoň jedno pole, které obsahuje více než jednu hodnotu.“ (Žák 2011)

- První normální forma

  „Tabulka je v první normální formě, jestliže lze do každého pole dosadit pouze jednoduchý datový typ (atributy jsou dále nedělitelné, tzv. atomické a tentýž atribut se neopakuje ve stejné tabulce).“ (Žák 2011)

- Druhá normální forma

  „Tabulka je ve druhé normální formě, jestliže je v první NF a navíc platí, že existuje klíč a všechna neklíčová pole jsou funkcí celého klíče (a tedy ne jen jeho částí).“ (Žák 2011)

- Třetí normální forma

  „Tabulka je ve třetí normální formě, jestliže každý neklíčový atribut není transitivně závislý na žádném klíči schématu neboli, je-li ve druhé normální formě a zároveň neexistuje jediná závislost neklíčových sloupců tabulky.“ (Žák 2011)

Existuje i čtvrtá a pátá normální forma. Designér databáze musí správně rozhodnout, podle které normální formy budou tabulky navrženy. Při používání vysokých normálních forem se může práce s tabulkami zkomplikovat, jelikož spousta tabulek bude obsahovat pouze číselné odkazy na skutečné informace. Použití určité normální formy záleží na konkrétním typu aplikace. V této prací jsou tabulky, až na pár výjimek, navrženy ve třetí normální formě.

4.2.4 Použité tabulky

Pokud není uvedeno jinak, všechny tabulky využívají úložný engine InnoDB a u primárního klíče každé tabulky je nastaven atribut AUTO_INCREMENT. V příloze E se nachází kompletní databázový model.

Role

Jednoduchá tabulka obsahující dostupné uživatelské role:

- admin – všechna práva,
- user – registrovaný zákazník,
- moderator – mazání komentářů,
- editor – správa zboží a kategorií,
- manager – správa objednávek, reklamací a skladu,
- guest – žádná práva (blokace uživatelského účtu).

Uzivatele

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_uzivatele	Int	YES	NO
FK	id_role	Int	YES	NO
	email	Varchar(70)	YES	YES
	heslo	Varchar(40)	YES	NO
	ip	Varchar(15)	YES	NO
	posledni_aktivita	Timestamp	YES	NO
	registrovan	Datetime	YES	NO
	heslo_token	Varchar(40)	NO	NO
	heslo_token_platnost	Datetime	NO	NO

Tabulka *Uzivatele* slouží k vedení uživatelů, kteří se mohou do obchodu přihlásit. Nachází se zde účty administrátorů i účty registrovaných zákazníků. Každý účet má přidělenou uživatelskou roli. Jako název uživatelského účtu je zvolen email, který musí být unikátní. Heslo je zkombinováno se solí, a poté zahashováno algoritmem SHA-1. Přihlašování probíhá ověřením emailu a hesla. U každého uživatele je zaznamenáván datum posledního přístupu, IP adresa a datum registrace. V případě zapomenutí hesla se využijí sloupce *heslo_token* a *heslo_token_platnost* pro jeho obnovu.

Zakaznici

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_zakaznika	Int	YES	NO
FK	id_uzivatele	Int	NO	NO

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
	jmeno	Varchar(30)	YES	NO
	prijmeni	Varchar(30)	YES	NO
	email	Varchar(70)	YES	NO
FK	id_adresy	Int	YES	NO
	firma	Varchar(50)	NO	NO
	ic	Int	NO	NO
	dic	Varchar(13)	NO	NO
	telefon	Varchar(15)	NO	NO
	mobil	Varchar(15)	NO	NO

V tabulce *Zakaznici* jsou uchovávány kontaktní a firemní údaje registrovaných i neregistrovaných zákazníků. Mohou zde být uchovávány i údaje o administrátorech. V případě neregistrovaných zákazníků je hodnota *id_uzivatele* NULL. U registrovaných zákazníků je pole *email* totožné s polem *email* v tabulce *Uzivatele*. V tomto případě by pole *email* v této tabulce mohlo být využito i pro jiný kontaktní email. Datový typ telefonních čísel je Varchar, aby bylo umožněno ukládání čísel i v mezinárodním formátu.

Adresy

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_adresy	Int	YES	NO
	ulice	Varchar(50)	YES	NO
	mesto	Varchar(50)	YES	NO
	psc	Mediumint	YES	NO

Tato tabulka se nachází pouze v nulté normální formě, jelikož atribut *ulice* obsahuje jak název ulice, tak i číslo popisné. Účelu internetového obchodu tento fakt nevadí. Pro splnění třetí normální formy by se tabulka dále musela rozdělit na dvě – adresy a města. Spojování by probíhalo přes primární klíč PSČ, nebo přes uměle vytvořené ID. Při vkládání dalších adres by se musela kontrolovat existence města s příslušným PSČ, což by vedlo ke komplikaci aplikace. Při použití primárního klíče PSČ by v situaci, kdy by dva různí zákazníci zadali stejné město s jiným PSČ (nebo naopak), docházelo ke konfliktu. Potřebám aplikace tato tabulka v této podobě vyhovuje.

Zpusob_Dopravy

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_dopravy	Int	YES	NO
	nazev	Varchar(30)	YES	YES
	cena	Float	YES	NO
	aktivni	Bool	YES	NO

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
	zmeneno	Timestamp	YES	NO

V této tabulce se nachází podrobnosti o dopravcích. Lze vkládat pouze nové dopravce a u stávajících měnit jejich cenu a aktivitu. Atribut *aktivni* indikuje, zda je dopravce stále nabízen a využíván, nebo je již archivován. Záznamy nelze mazat z důvodu vazby na tabulku *Objednavky*.

Zpusob_Platby

Tato tabulka má podobnou strukturu jako tabulka *Zpusob_Dopravy*, viz výše.

Stav_Objednavky

V tabulce se nachází všechny dostupné stavy objednávek:

- čeká na vyřízení – nové objednávky,
- čeká se na platbu – nezaplacené objednávky připravené k odeslání,
- vyřizuje se – položky objednávky se připravují (nebo byly objednány od dodavatele),
- vyřízená – dokončené objednávky,
- položky aktualizovány – objednané položky od dodavatele byly již naskladněny.

Objednavky

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_objednavky	Int	YES	NO
FK	id_zakaznika	Int	YES	NO
FK	id_dopravy	Int	YES	NO
FK	id_platby	Int	YES	NO
FK	id_dodaci_adresy	Int	YES	NO
	cena_dopravy	Float	YES	NO
	cena_platby	Float	YES	NO
	vytvoreno	Datetime	YES	NO
	zmeneno	Datetime	YES	NO
FK	id_stavu_objednavky	Int	YES	NO

Jedna z nejdůležitějších tabulek aplikace. Po odeslání objednávky v nákupním košíků je vytvořen záznam o nové objednávce a do tabulky *Zbozi_Objednavky* jsou vloženy veškeré položky objednávky. Každá objednávka má nastavenu určitou dopravu, platbu a stav. Dodací adresa se buď shoduje s adresou zákazníka, nebo má vytvořenou zcela jinou dodací adresu v tabulce *Adresy*. V době objednání jsou přeneseny aktuální ceny za dopravu a platbu. V atributu *vytvoreno* se udržuje datum a čas vytvoření objednávky a v atributu *zmeneno* poslední manipulace s objednávkou.

Zbozi_Objednavky

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_polozky	Int	YES	NO
FK	id_objednavky	Int	YES	NO
FK	id_zbozi	Int	YES	NO
FK	cislo_faktury	Int	NO	NO
	cena	Float	YES	NO
	procenta	Float	YES	NO
	pocet	Int	YES	NO
FK	id_stavu_polozky	Int	YES	NO

Výčet všech položek ke každé objednávce. V tabulce je vytvořen primární klíč *id_polozky* pro usnadnění práce. Atribut *cislo_faktury* indikuje, zda byla položka již fakturována (odeslána). Současná implementace však umožňuje expedovat pouze kompletní objednávky, tudíž se jedná pouze o přípravu pro tuto funkcionalitu. V době vytvoření objednávky je zkopírována aktuální cena zboží a sazba DPH.

Stav_Zbozi_Objednavky

V tabulce se nachází všechny dostupné stavy objednaného zboží:

- rezervováno – byl rezervován požadovaný počet kusů, nebo bylo zboží již naskladněno a zarezervováno,
- vyřizuje se – skladem nebyl dostatečný počet kusů, čeká se na objednání od dodavatele,
- objednáno – zboží bylo objednáno od dodavatele.

Faktury

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	cislo_faktury	Int	YES	NO
	datum_vystaveni	Datetime	YES	NO
	datum_splatnosti	Datetime	YES	NO
	datum_uhrady	Datetime	NO	NO

Jelikož současná implementace umožňuje odesílání pouze kompletních objednávek, tvoří se jedna společná faktura pro všechny položky objednávky. Atribut *cislo_faktury* je generován ve formě YYMM_____n, kde YY značí poslední dvojčíslí roku, MM číslo měsíce včetně počáteční nuly, a n pořadové číslo faktury v daném období. Atribut *datum_uhrady* indikuje, zda byla faktura zaplacena.

Reklamace

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_reklamace	Int	YES	NO
FK	id_polozky	Int	YES	NO
	email	Varchar(70)	YES	NO
	telefon	Varchar(15)	YES	NO
	vytvorena	Datetime	YES	NO
	prijata	Datetime	NO	NO
	vyrizena	Datetime	NO	NO
	popis_zavady	Text	YES	NO
	vyrozumeni	Text	NO	NO
FK	id_pozadovane_vyrizeni	Int	YES	NO
FK	id_vyrizeno_zpusobem	Int	NO	NO

Tato tabulka obstarává vyřizování reklamací, přičemž lze reklamovat pouze jedno zboží naráz. Do atributů *email* a *telefon* se ukládají kontaktní údaje – mohou být jiné, než v době nákupu. Pole *prijata* obsahuje datum, kdy se reklamace začala opravdu vyřizovat.

Zpusob_Vyrizeni_Reklamace

V tabulce se nachází všechny možné způsoby vyřízení reklamací:

- oprava,
- výměna za nové,
- vrácení peněz,
- zamítnuta.

Zbozi

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_zbozi	Int	YES	NO
	nazev	Tinytext	YES	NO
	popis	Text	NO	NO
FK	id_kategorie	Int	YES	NO
FK	id_sazby_dph	Int	YES	NO
	cena	Float	YES	NO
	ks_skladem	Int	YES	NO
	zobrazovat	Bool	YES	NO
FK	id_seo	Int	YES	NO
FK	id_stavu_zbozi	Int	YES	NO
	upraveno	Timestamp	YES	NO

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
	rating	Float	NO	NO
	rated	Int	NO	NO
	sum	Int	NO	NO
FK	id_obrazku	Int	NO	NO

Nejdůležitější tabulka, ve které se nachází veškerý sortiment. V tabulce se nachází i archivované zboží, což indikuje atribut *zobrazovat*. Atributy *rating*, *rated* a *sum* slouží k hodnocení produktu. Postupně obsahují: aktuální hodnocení, počet hodnocení a suma všech známek. *Id_obrazku* odkazuje na obrázek, který je použit jako hlavní (náhledový). Zboží se může nacházet pouze v jedné kategorii. Toto jisté omezení je řešeno zobrazováním daného zboží i ve všech nadřazených kategoriích.

Sazby_DPH

Jednoduchá tabulka obsahující pouze název sazby DPH a její procentuální vyjádření.

Stav_Zbozi

V tabulce se nachází všechny dostupné stavy zboží:

- skladem,
- není skladem,
- objednáno,
- nedostupné.

Komentare

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_komentare	Int	YES	NO
FK	id_zbozi	Int	YES	NO
	autor	Varchar(30)	YES	NO
	email	Varchar(70)	NO	NO
	nadpis	Varchar(50)	YES	NO
	zprava	Text	YES	NO
	vlozeno	Timestamp	YES	NO

Tabulka obsahuje jednotlivé komentáře k produktům. Ukládán je i email autora, ten však není zveřejňován.

SEO

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_seo	Int	YES	NO

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
	keywords	Tinytext	NO	NO
	description	Text	NO	NO
	url	Text	YES	NO

Tato tabulka uchovává informace potřebné k optimalizaci pro vyhledávače. Jedná se o klíčová slova, popis a hezkou URL adresu. Informace z tabulky jsou využívány pro zobrazování zboží a kategorií. Při každém přístupu na tyto stránky a při generování odkazů se překládají URL adresy, proto se tyto informace ukládají do paměti cache.

Obrazky

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_obrazku	Int	YES	NO
	popis_obrazku	Text	NO	NO
	cesta	Tinytext	YES	NO

V tabulce *Obrazky* se nachází všechny dostupné obrázky. Jeden produkt může disponovat několika obrázky. Současná implementace je připravena na používání jednoho totožného obrázku u více produktů. Přidáním vazební tabulky mezi tabulku *Obrazky* a *Kategorie* by se dále jednoduše umožnilo zobrazování obrázků prezentujících danou kategorii. U každého obrázku je zaznamenáván jeho popis a relativní cesta k souboru s obrázkem. Obrázky nejsou ukládány přímo do databáze, čímž se snižuje zatížení databázového systému. Ukládání obrázků na disk je však mírně komplikovanější, jelikož jsou potřeba další obslužné funkce.

Fulltext

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PFK	id_zbozi	Int	YES	NO
	nazev	Tinytext	YES	NO
	popis	Text	NO	NO

Tato tabulka využívá jako jediná úložný engine MyISAM. Jak již plyne z názvu, tento engine se využívá právě kvůli podpoře fulltextového vyhledávání. MySQL nabízí dva druhy vyhledávání:

- boolean vyhledávání – podporuje vyhledávací operátory,
- vyhledávání v přirozeném jazyce – automatické řazení dle relevance.

Boolean vyhledávání umožňuje používat speciální operátory ke zpřesňování výsledků. Jedná se například o operátory „+“ (a současně), „–“ (bez slova), „*“ (jakýkoliv znak) a další. Ve výchozím nastavení se ignorují běžná anglická slova a slova kratší než 4 znaky.

Oproti vyhledávání v přirozeném jazyce se neignorují slova, která se nacházejí ve více než 50% řádků. Tento typ vyhledávání je použit v práci.

V tabulce jsou definovány celkem tři fulltextové indexy – dva nad sloupci *nazev* a *popis* a třetí společný nad oběma sloupci. Dva samostatné indexy se využívají k řazení výsledku, přičemž váha hledaného slova v názvu je 5x vyšší než v popisu. Podle třetího společného indexu se vyhledává (Vrána 2006). Sloupce mají nastaveno porovnávání *utf8_general_ci*, tudíž lze totožně vyhledávat text s diakritikou i bez ní. Při nenalezení žádných výsledků (krátké slovo) se zkouší vyhledávání ještě pomocí LIKE, viz následující kód:

```
public function search($text) {
      $result = $this->database->query(
'SELECT products.* FROM shop_fulltext
 JOIN products USING (id_zbozi)
 WHERE MATCH (shop_fulltext.nazev, shop_fulltext.popis) AGAINST (%s IN BOOLEAN MODE)', $text,
'ORDER BY 5 * MATCH (shop_fulltext.nazev) AGAINST (%s) +
                  MATCH (shop_fulltext.popis) AGAINST (%s) DESC', $text, $text,
'LIMIT 100');
      if (!$result->fetchAll()) {
         $result = $this->database->query(
'SELECT products.* FROM shop_fulltext
JOIN products USING (id_zbozi)
WHERE shop_fulltext.nazev LIKE %~like~ OR shop_fulltext.popis LIKE %~like~', $text, $text,
'LIMIT 100');
      }
      return $result;
}
```

Kategorie

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_kategorie	Int	YES	NO
FK	id_nadkategorie	Int	NO	NO
	nazev	Tinytext	YES	NO
	popis	Mediumtext	NO	NO
	zobrazovat	Bool	YES	NO
FK	id_seo	Int	YES	NO

V této tabulce je uložen strom kategorií. Každá kategorie disponuje atributem obsahující ID nadřazené kategorie. Kategorie na nejvyšší úrovni mají tento atribut roven NULL. Tento přístup nabízí velice jednoduché upravování a vkládání nových kategorií. Problém nastává při získávání celého stromu, které se provádí pomocí rekurzivního volání. Jelikož se jedná o poměrně náročnou a často prováděnou operaci, je získaný strom kategorií automaticky uložen do paměti cache a odsud je nadále načítán.

Při výpisu zboží v dané kategorii je nutné prohledávat i všechny dceřiné kategorie, aby byly zobrazeny produkty z celého podstromu. Opět se jedná o poměrně náročnou operaci, pro se výsledky těchto dotazů ukládají do paměti cache.

Dodavatele

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_dodavatele	Int	YES	NO
	nazev	Tinytext	YES	NO
	email	Varchar(70)	NO	NO
	telefon	Varchar(15)	NO	NO
FK	id_adresy	Int	YES	NO

Tabulka obsahuje dodavatelé, kteří zásobují internetový obchod. Vedeny jsou pouze kontaktní údaje.

Dodavatele_Zbozi

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id	Int	YES	NO
FK	id_dodavatele	Int	YES	NO
FK	id_zbozi	Int	YES	NO
	nakupni_cena	Float	YES	NO
	dodaci_doba	Int	NO	NO

Vazební tabulka uchovávající informace o dodávaném zboží. Jeden dodavatel může dodávat více produktů a zároveň jeden produkt může být dodáván více dodavateli. Vedena je nákupní cena a orientační dodací doba. Atribut *id* byl přidán kvůli jednodušší manipulaci s nákupním košíkem ve skladu.

Dodavky

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	id_dodavky	Int	YES	NO
FK	id_dodavatele	Int	NO	NO
	datum_vytvoreni	Datetime	YES	NO
	datum_prijeti	Datetime	NO	NO
	predmet_zpravy	Tinytext	NO	NO
	zprava	Text	NO	NO

V této tabulce jsou vedeny realizované objednávky u dodavatelů. Do atributů *predmet_zpravy* a *zprava* se ukládá obsah odesílaného emailu spolu s objednávkou.

Dodavky

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PFK	id_zbozi	Int	YES	NO

Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PFK	id_dodavky	Int	YES	NO
	cena	Float	YES	NO
	pocet	Int	YES	NO
	dodaci_doba	Int	NO	NO
	naskladneno	Int	NO	NO

Vazební tabulka mezi tabulkou *Dodavky* a *Zbozi*. Obsahuje jednotlivé položky realizovaných objednávek u dodavatelů. Do atributů *cena* a *dodaci_doba* jsou přeneseny informace z tabulky *Dodavatele_Zbozi*. Atribut *naskladneno* umožňuje naskladnit jiný počet zboží, než byl objednaný (např. v případě chybějících nebo poškozených kusů). Bez této volby by bylo nutné (v případě potřeby) ručně zasahovat do počtu kusů skladem.

Nastaveni

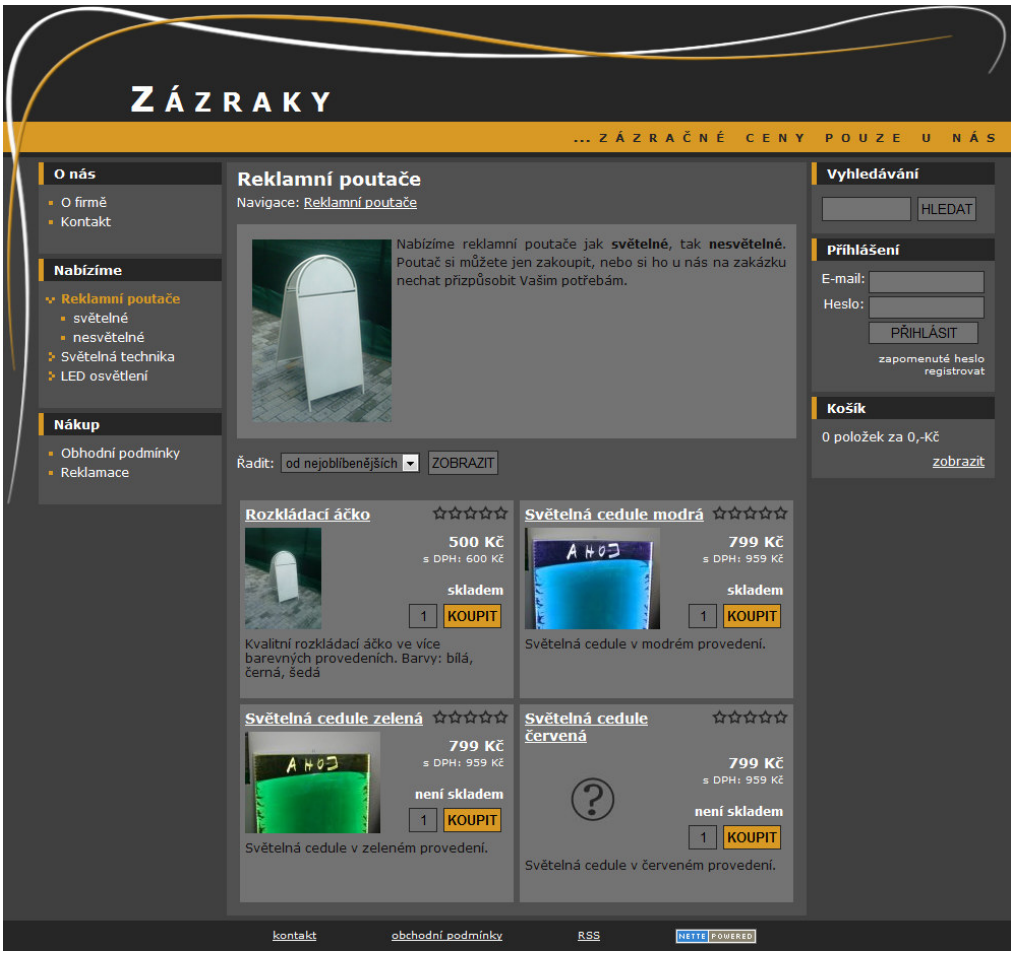
Klíč	Sloupec	Datový typ	Nenulový	Unikátní
PK	klic	Varchar(50)	YES	NO
	hodnota	Text	NO	NO

Tabulka určená k ukládání obecného nastavení obchodu. Nastavení se ukládá ve formě klíč a jeho hodnota.

4.3 Webová vrstva

Webová část aplikace je postavena na populárním Nette Frameworku. Framework pokrývá veškerou potřebnou problematiku. Hlavní důvody pro výběr tohoto řešení byly: bezpečnost výsledné aplikace, pokročilé ladící nástroje, přehledný a snadno rozšířitelný návrh, podpora moderních technologií, možnost využívat celou řadu doplňků (komponent) a rozsáhlá aktivní komunita.

Hlavní layout (viz obrázek 6) a všechny šablony využívají jazyk HTML 5. Pro zpětnou kompatibilitu se staršími verzemi Internet Exploreru se využívá JavaScriptový kód *html5shiv*[20], který zajišťuje správné zobrazení HTML 5 elementů. Grafická úprava stránek je řešena pomocí kaskádových stylů. Pomocí CSS je také řešena optimalizace pro tisk.



Obrázek 6 – Layout webu

4.3.1 Optimalizace pro vyhledávače

Optimalizace pro vyhledávače, neboli Search Engine Optimalization (SEO), slouží k dosažení co nejlepších pozic ve výsledcích vyhledávačů. Umístění je určováno na základě SERP (Search Engine Result Page). Konkrétní pozice je ovlivněna několika

[20] http://code.google.com/p/html5shiv/

faktory. Záleží na umístění hledaného slova ve stránce, v titulku a v nadpisu, v metaznačkách keywords a description. Dalším faktorem je i hodnocení stránky podle vyhledávače (PageRank). Důležitá jsou dobře vybraná klíčová slova, vhodně stylizované texty a správně strukturovaný dokument (Kubíček 2008). Problematika SEO je velmi obsáhlá, a tudíž zde nebude rozebírána do podrobností.

O SEO se z velké části stará Nette Framework routováním URL adres. Jedná se o překlad URL adresy na konkrétní akci presenteru a naopak. Tento mechanismus zajišťuje kanonizaci adres, čímž se zabraňuje vzniku shodných stránek s různou URL adresou. Framework automaticky přesměruje duplicitní stránky na jednu hlavní (kanonickou) adresu. Nedochází tak k indexaci totožného obsahu a nemění se hodnocení stránky.

Pomocí routování lze poměrně jednoduše a dynamicky vytvářet hezké URL adresy, které také přispívají k lepšímu umístění ve výsledcích. V aplikaci byly vytvořeny dva routery komunikující s databází, které využívají tabulku SEO. Jeden z nich obstarává přepis URL adres u kategorií, a druhý u produktů. Z důvodu jednoznačné identifikace je adresa vždy složena ve tvaru: `ID-pekna-url-adresa`.

Další optimalizaci zajišťují metaznačky v hlavičce HTML kódu. Konkrétně se jedná o keywords a desscription. Obě hodnoty lze u produktů a kategorií libovolně měnit. K další optimalizaci slouží vhodně členěný HTML kód, pečlivě vybíraná klíčová slova a hodnotné a unikátní texty na stránkách.

4.3.2 AJAX

Technologie AJAX (Asynchronní JavaScript a XML) se využívá při tvorbě moderních interaktivních webových aplikací. Využití AJAXu umožňuje načítat a měnit pouze požadované části stránek, aniž by došlo k obnovení celé stránky. Při správné implementaci je zvýšen uživatelský komfort a snížena zátěž webového serveru – nemusí se přenášet celé stránky, ale pouze její požadované části.

Tato technologie s sebou přináší i jistá úskalí. Pro chod AJAXu je vyžadován moderní prohlížeč se zapnutou podporou JavaScriptu. Pokud nelze zpracovat požadavek pomocí AJAXu, musí se požadavek zpracovat normálně. Nette Framework dokáže typ požadavku detekovat, a podle toho se příslušným způsobem zpracovává.

Dalším rizikem je použití AJAXu tam, kde se dbá na optimalizaci pro vyhledávače. Jelikož se informace vypisují až na základě uživatelské akce, nejsou tyto informace pro vyhledávací roboty viditelné. Proto není vhodné využít AJAX pro stránkování produktů, nebo dokonce k jejich zobrazování. V aplikaci je tato technologie použita pouze tam, kde je vhodná a nemá žádný dopad na SEO. Překreslování částí stránek a odesílání AJAXových požadavků je řešeno pomocí *jQuery*[21].

[21] http://addons.nette.org/cs/jquery-ajax

4.3.3 Kategorie

Strom kategorií může být libovolně zanořený. Zobrazovaná data jsou načítána z paměti cache, čímž se odlehčuje zátěž databázového serveru. Pokud data v paměti cache nejsou, nebo byla změněna, je strom kategorií vytvořen rekurzivním voláním funkce. O samotné zobrazování stromu se stará komponenta *Navigation*[22], která navíc umožňuje zobrazování drobečkové navigace. Výsledný strom se vypisuje pomocí zanořovaného nečíslovaného seznamu. Komponenta byla upravena tak, aby se při přihlášení administrátora zobrazovaly ikony pro úpravu kategorií.

Zobrazování celého stromu není příliš praktické a přehledné, pro se ve výchozím stavu zobrazuje strom sbalený. Pří zobrazení konkrétní kategorie nebo produktu se rozbalí pouze daný podstrom. Pomocí šipek lze strom libovolně rozbalovat a sbalovat, aniž by se přešlo na příslušnou kategorii. Tuto funkcionalitu zajišťuje JavaScriptová knihovna *Simple Tree Menu*[23].

4.3.4 Vyhledávání

Vyhledávání v internetovém obchodě patří mezi nejdůležitější funkce. Nemělo by být závislé na používání diakritiky a na velikosti písmen. Nalezené záznamy očekáváme seřazené dle jejich relevance. Uživatel by také měl mít možnost vyhledávat i pomocí velmi krátkých klíčových slov, např. název výrobce LG. Vyjmenované vlastnosti zajišťuje použitý databázový systém a správně nastavené fulltextové tabulky. Podrobnosti jsou uvedeny u databázové tabulky *Fulltext*.

4.3.5 Nákupní košík

Nákupní košík je ukládán do sessions po dobu maximálně 7 dnů. Obsah košíku tedy závisí na konkrétním prohlížeči a stroji. Pokud by se požadovalo přenášení košíku v rámci uživatelského účtu, muselo by se přistoupit k implementaci s využitím databáze.

O informativní výpis košíku na každé stránce (počet položek a jejich cena) se stará komponenta *Basket*, ve které jsou uloženy všechny informace. Celkovou cenu vrací následující metoda, přičemž se provádí pouze jeden databázový dotaz. Zda bude vrácena cena bez DPH, nebo včetně, ovlivňuje parametr *$dph*.

```
public function totalPrice($dph = TRUE) {
    $items = $this->getItems();

    if (!$items)
      return 0;
```

[22] http://addons.nette.org/cs/navigation
[23] http://www.dynamicdrive.com/dynamicindex1/navigate1.htm

```
        $ids = array_keys($items);
        $products = $this->template->items = $this->presenter->models->ProductModel->getProduct($ids)
            ->fetchAll();

        $sum = 0;
        foreach ($products as $product) {
            $sum += $items[$product->id_zbozi] * ($dph ? $product->cena_vcetne_dph : $product->cena);
        }

        return $sum;
    }
```

Vkládání produktů do košíku je realizováno pomocí AJAXu, tudíž se nepřekresluje celá stránka, ale pouze se zobrazí informativní zpráva o vložení zboží do košíku.

Obsluhu košíku a následný nákup zajišťuje *BasketPresenter*. Aplikace umožňuje provádět nákupy i bez registrace, přičemž jsou vyžadovány alespoň základní kontaktní údaje. V prvním kroku objednávky je nabídnut způsob dopravy a platby a volitelně lze zadat jinou dodací adresu. Následuje rekapitulace objednávky, kterou je stále možné upravit. Po kontrole je objednávka odeslána. Pokud se v době odeslání objednávky nachází na skladě dostatečný počet kusů zboží, tak je automaticky rezervováno. V opačném případě se v administrační části skladu objeví informace o nutnosti objednat požadované zboží.

V příloze D se nachází diagram aktivit znázorňující celý proces nákupu.

4.3.6 Správa skladu

Správa skladu přímo v internetovém obchodě usnadňuje celkové vedení skladu. Dává kompletní přehled o skladových zásobách; umožňuje jednoduše objednat zboží přímo od dodavatele; upozorňuje na požadované zboží, které není skaldem.

Na hlavní stránce skladu se na samém začátku zobrazuje zboží, které bylo objednáno zákazníkem a požadovaný počet kusů nebyl dostupný. Tvorba objednávek pro dodavatelé je řešena formou nákupního košíku. Lze zobrazovat zboží pouze od určitého dodavatele, nebo konkrétní zboží od všech dodavatelů. Po kliknutí na ikonu nákupního košíku se zobrazí formulář pro zadání požadovaného počtu zboží. Po potvrzení se zboží přidá na rozepsanou objednávku. Pro vyšší komfort se využívá technologie AJAX. Po přidání všech položek je možné objednávku odeslat. Volitelně lze zadat zprávu pro dodavatele, poté je automaticky odeslán email na adresu dodavatele včetně obsahu celé objednávky.

V záložce objednávky je možné procházet všechny realizované objednávky. Nenaskladněné objednávky se prostřednictvím této stránky přijímají do skladu. Kvůli možným neshodám v objednávce (např. rozbité zboží) je možno upřesnit přijímaný počet kusů do skladu. Pokud bylo naskladněno zboží, na které nějaká objednávka čekala, je automaticky správce objednávek o této události informován. Notifikace se provádí prostřednictvím změny stavu objednávky.

Pomocí stránky dodavatelé se mohou přidávat a upravovat dodavatelé. Na totožné stránce se přidává i konkrétní dodávané zboží. Samozřejmě může jeden dodavatel dodávat více produktů a jeden produkt může být dodáván více dodavateli.

4.3.7 Správa reklamací

Reklamace zboží je přístupná bez nutnosti přihlášení, což umožňuje reklamovat zboží i neregistrovaným zákazníkům. Na začátku reklamačního procesu je zákazník vyzván k zadání čísla faktury a data jejího vystavení. Při shodě se vypíše seznam objednaného zboží, které se nacházelo na dané faktuře. Po výběru vadného zboží, preferovaného způsobu vyřízení a popsání závady se reklamace zavede do systému.

V administraci se objeví nový záznam. Reklamaci je nutné přijmout, což později indikuje, zda se již zpracovává. V detailu se nacházejí podrobnosti o reklamaci. Zde je i formulář pro její vyřízení. Pro případ výměny zboží se zobrazuje i aktuální počet kusů skladem. Při vyřízení reklamace je zákazník automaticky informován emailem.

4.3.8 Správa produktů a kategorií

Jak již bylo řečeno v popisu databázových tabulek *Zbozi* a *Kategorie*, kategorie mohou být libovolně zanořovány. Jeden produkt se však může nacházet pouze v jedné kategorii. Produkt ovšem zobrazuje i ve všech nadřazených kategoriích. U každého produktu a kategorie lze pro vyhledávače nastavit klíčová slova, popis a hezkou URL adresu. Zobrazování jednotlivých kategorií a produktů se může vypnout tak, aby nebyly viditelné pro návštěvníky. Popisy se zadávají pomocí WYSIWYG [24]editoru *TinyMCE*[25].

4.3.9 Práce s obrázky

Nahrávání obrázků v administraci zajišťuje komponenta *MultipleFileUpload*[26] (MFU). Komponenta není výhradně určena pouze pro nahrávání obrázků. Umožňuje najednou nahrávat více souborů na server za podpory AJAXu. MFU zastřešuje několik utilit, které samotné nahrávání vykonávají. V rámci dané utility se ještě může volit používaná technologie. Pokud se z jakéhokoli důvodu nedaří soubory nahrát, je možné využít standardního HTML formuláře.

V aplikaci je preferována utilita Plupload s využitím technologie Flash. Zatímco jsou všechny obrázky nahrávány, může být doplňován podrobný popis. Po úspěšném odeslání obrázků a uložení informací je nabídnuta stránka s novým obrázky, kde se může doplnit jejich popis.

[24] WYSIWYG – editor pro vkládání formátovaného textu pomocí HTML značek.
[25] http://www.tinymce.com/
[26] http://addons.nette.org/cs/multiplefileupload

Pro zobrazování obrázků u produktů se využívá JavaScriptová knihovna *FancyBox*[27], která umožňuje procházet všechny obrázky v podobě prezentace. Obrázky jsou automaticky zmenšovány na zobrazitelnou plochu.

4.3.10 Hodnocení produktů

Hodnocení produktů obstarává komponenta *RatingControl*. Započítání hlasu je prováděno pomocí AJAXu. Hlasovat lze pro každý produkt pouze jednou denně. Ochrana se řeší uložením času do session. Zobrazení aktuálního hodnocení a vizuální efekty při hodnocení jsou realizovány pomocí kaskádových stylů – *CSS Star Rating Redux*[28].

4.3.11 Komentáře produktů

O zobrazování a přidávání komentářů se stará komponenta *CommentsControl*. Manipulace s komentáři se rovněž provádí pomocí technologie AJAX. Samozřejmostí je podpora stránkování. Vložit nový komentář lze pouze po správném opsání ověřovacího kódu, který generuje komponenta *CaptchaControl*[29]. Jedná se o jednoduchý Turingův test, který zabraňuje vkládání komentářů roboty.

4.3.12 Správa uživatelů

Aplikace využívá celkem 5 uživatelských rolí, viz databázová tabulka *Role*. V závislosti na přidělené roli jsou v uživatelském menu po přihlášení dostupné různé části administrace. O zobrazení přihlašovacího formuláře a uživatelského menu se stará komponenta *LoginControl*. Přihlašovací formulář je odesílán na zabezpečenou stránku (HTTPS), tudíž nelze odposlechnout přihlašovací údaje.

Pro vyšší uživatelský komfort je při registraci „živě“ pomocí AJAXu ověřována existence zadávaného emailu. Pokud je email již použit, je uživatel ihned informován zprávou.

Pokud uživatel zapomene heslo, nabízí aplikace proces pro jeho obnovu. Stačí zadat registrační email, na který je odeslán odkaz pro obnovu hesla. Odkaz obsahuje vygenerovaný klíč, kterým se ověří totožnost uživatele. Platnost klíče vyprší po jednom dni. Pokud je klíč platný, zobrazí se formulář pro zadání nového hesla (Vrána 2010).

4.3.13 Generování faktur

Obchod nabízí export vystavených faktur do populárního PDF formátu. K exportu je využita komponenta *InvoiceControl*[30], která interně využívá knihovnu mPDF. Šablona faktury je naplněna potřebnými údaji – dodavatel, odběratel a objednané položky.

[27] http://fancybox.net/
[28] http://www.komodomedia.com/blog/2007/01/css-star-rating-redux/
[29] http://github.com/PavelMaca/CaptchaControl
[30] http://github.com/OndrejBrejla/Nette-InvoiceControl

Knihovna mPDF vytvoří ze šablony dokument PDF, který je možné stáhnout či ihned zobrazit v prohlížeči.

4.3.14 Stránkování a řazení

Pro stránkování je využita komponenta *VisualPaginator*[31], která automaticky uchovává aktuální stránku a stará se o posun mezi jednotlivými stránkami. Řazení obstarává databázový systém, nicméně je nutné zařídit zpracování parametru z URL adresy a předat jej ve vhodné formě databázi, resp. knihovně dibi. To zajišťuje následující metoda:

```
protected function getOrdering($param, $possible) {
    $order = Strings::match($param, "/(?<key>($possible))[-](?<order>(asc|desc))/");
    return $order ? array($order['key'] => $order['order']) : NULL;
}
```

V parametru *$possible* jsou svislítkem oddělené sloupce, podle kterých se může řadit. Pokud parametr *$param* vyhoví regulárnímu výrazu, je vráceno pole s klíčem „název sloupce“ a hodnotou buď „asc“, nebo „desc“. Stránkování včetně řazení může být použito takto:

```
$orderBy = $this->getOrdering($this->order, 'rating|cena|nazev');
$result = $this->models->ProductModel->getProductIn($id, $orderBy, $show);

$paginator = $this['paginator']->getPaginator();
$paginator->itemCount = $result->getRowCount();

$products = $result->fetchAll($paginator->offset, $paginator->itemsPerPage);
```

4.3.15 RSS kanál

Aplikace nabízí informační RSS kanál s nejnovějšími produkty. Zákazníci se mohou přihlásit k odběru novinek a zůstat tak stále informováni o nejnovějším sortimentu. Ke generování RSS kanálu je použita komponenta *RssControl*[32], která usnadňuje tvorbu výsledného XML souboru.

4.4 Instalace aplikace

4.4.1 Požadavky

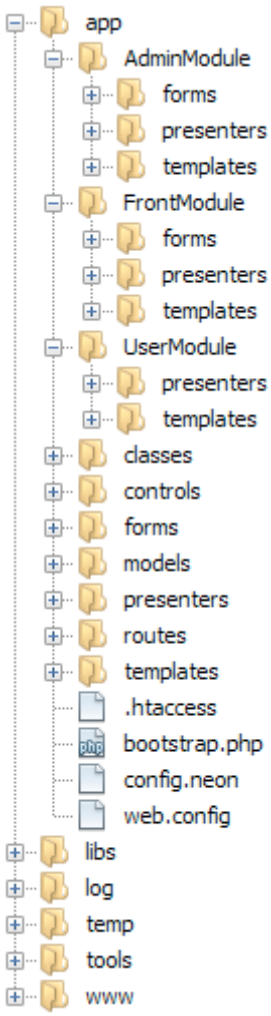
Aplikace běží na Nette Frameworku verze 2, tudíž je vyžadováno alespoň PHP 5.3. Podrobnější nastavení webového serveru lze otestovat přiloženým nástrojem Requirements-Checker, který se nachází ve složce *tools*. Pro jeho spuštění je nutné povolit přístup do tohoto adresáře pomocí souboru *.htaccess*. Aplikace musí mít povolen zápis do adresářů *temp* a *log* pro ukládání dočasných souborů a chybových zpráv. Ke správnému

[31] http://addons.nette.org/cs/visualpaginator
[32] http://addons.nette.org/cs/rsscontrol

zobrazení je vyžadován moderní webový prohlížeč se zapnutou podporou JavaScriptu. Databázová část vyžaduje databázový systém MySQL verze 5 a vyšší.

4.4.2 Struktura aplikace

Aplikace využívá předpřipravené struktury Nette Frameworku. Celá aplikace se nachází ve složce *app*. Aplikace je rozdělena do tří modulů:



- FrontModule – veřejná část,
- UserModule – uživatelská část,
- AdminModule – administrátorská část.

Každý modul obsahuje vlastní presentery, šablony a formuláře.

Zbylé složky obsahují třídy používané všemi moduly:

- classes – pomocné třídy (autentikátor, autorizátor…),
- controls – používané komponenty,
- forms – formuláře společné pro více modulů,
- models – vrstva pro práci s databází,
- presenters – obsahuje pouze *BasePresenter*, od kterého jsou odvozovány další presentery v jednotlivých modulech,
- routes – třídy pro překlad URL adres,
- templates – layout a další společné šablony (email).

Soubory *.htaccess* a *web.config* slouží k nastavení použitého webového serveru. *Config.neon* obsahuje veškeré nastavení a *bootstrap.php* slouží ke spuštění celé aplikace.

Další složky na úrovni složky *app* mají následující význam:

- libs – složka pro knihovny; obsahuje Nette, Nette-minified, dibi a mPDF,
- log – slouží k zápisu chyb a výjimek na produkčním serveru,
- temp – složka pro dočasné soubory, úložiště paměti cache,
- tools – obsahuje nástroj Requirements-Checker pro otestování konfigurace,
- www – jediný adresář dostupný z webu, obsahuje *index.php*, CSS styly, obrázky a další používané soubory.

4.4.3 Kroky instalace

Nejprve je nutné obsah archivu *website.zip* zkopírovat na server. Direktivu DocumentRoot nastavit do složky *www*. Adresáře *temp* a *log* musí mít nastavená práva pro zápis.

V dalším kroku se musí vytvořit databáze, ve které se spustí skript *install.sql*. Pro ukázková data je ještě nutné spustit skript *test_data.sql*. Tyto kroky lze udělat pomocí správce databáze Adminer, který se nachází v kořeni webu ve složce *adminer*.

V konfiguračním souboru *config.neon*, který se nachází ve složce *app*, se musí nastavit přístupové údaje k databázovému serveru a název vytvořené databáze (*common: parameters: database:*). Pro správnou funkčnost aplikace je také potřeba nastavit odesílaní emailů. To se provádí v totožném souboru. Ve výchozím stavu se využívá PHP funkce `mail()`. Pro využití vlastního SMTP serveru se musí odkomentovat řádek *smtp:* a nastavit příslušné údaje.

V případě potíží se doporučuje smazat obsah adresáře *temp* (vyjma souboru *.htaccess* a *web.config*). V závislosti na konfiguraci webového serveru může být potřeba odkomentovat řádek RewriteBase v souboru *www/.htaccess*.

Produkční a vývojový režim se nastavuje automaticky. Pro vynucení produkční režimu je potřeba odkomentovat řádek `$configurator->setProductionMode(TRUE)` v souboru *app/bootstrap.php*.

Závěr

V teoretické části byly nastíněny výhody a nevýhody používání frameworků obecně. Poté se přešlo k problematice webových frameworků, konkrétně pro skriptovací jazyk PHP. Z desítek PHP frameworků byly tři nejpopulárnější a nejpoužívanější frameworky v České republice porovnány a analyzovány. Jedná se o český nejpopulárnější Nette Framework, celosvětově rozšířený Zend Framework a francouzský framework Symfony. Každé řešení nabízí zcela jiné možnosti. Neliší se pouze svým rozsahem a funkcemi, ale mají i odlišnou filozofii. Každý framework se lépe hodí k vývoji jiných aplikací. Záleží také na preferencích vývojáře, jaký styl práce mu je osobní. Nelze jednoznačně říci: „Tento framework je ze všech nejlepší a ostatní jsou naprosto zbytečné."

V implementační části se podařilo vytvořit webovou prezentaci s internetovým obchodem dle předpokladů. Obě zdánlivě různé části jsou úzce propojeny v jeden funkční celek – žádná tlačítka ve smyslu vstupte do obchodu apod. Výsledné webové stránky mají jednoduchý a přehledný design. Fyzicky je aplikace samozřejmě rozdělena na veřejnou a administrátorskou část. Prakticky se však uživatel stále nachází ve stejném rozhraní. Kromě běžných funkcí internetového obchodu nabízí aplikace správu skladu a reklamací. V případě reklamací se jedná o poměrně jednoduché řešení. Komplexnější je již zmiňovaná správa skladu. Ta automaticky upozorňuje jeho správce, že je potřeba objednat zboží. Po přidání všech potřebných produktů na objednávku se vytvoří zpráva, která je zaslána dodavateli. Po obdržení objednaného zboží se pomocí rozhraní skladu produkty naskladní. Automaticky jsou přičteny počty kusů skladem a současně jsou aktualizovány objednávky čekající na dané zboží. Veškeré funkce obchodu se zpřístupňují podle uživatelských rolí, kterých je celkem 5. Nechybí role pro administrátora a zákazníka. Další role jsou určeny pro správu produktů a kategorií; mazaní komentářů a poslední role pro správu objednávek, skladu a reklamací.

Vývoj propracovaného internetového obchodu, který bude nabízet spoustu funkcí, je běh na velmi dlouhou trať. Stávající řešení je plně provozuschopné. Samozřejmě se najdou části, kde by se dala přidat další funkcionalita. Na některá rozšíření je obchod již připraven. Jedná se třeba o dodávání objednávek po částech, nebo vkládání více obrázků ke kategoriím.

Pro mě, jako pro autora celé práce, se jednalo o velmi zajímavé a přínosné téma. Podrobněji jsem se seznámil s vývojem webových aplikací pomocí frameworků. Tři vybrané frameworky jsem blíže poznal a zjistil, jaké mají přednosti, či nedostatky. Výsledná webová aplikace byla vytvořena pomocí Nette Frameworku, tudíž jsem si práci s tímto frameworkem osvojil a získal jsem drahocenné zkušenosti. Díky obsáhlosti frameworku se bude skoro vždy čemu učit a mít se v čem zdokonalovat. Kromě Nette Frameworku jsem také nabyl zkušenosti v CSS stylování, návrhu databáze a optimalizaci stránek pro vyhledávače.

Literatura

Framework. In: *Wikipedie: Otevřená encyklopedie* [online]. 2002–2012, Datum poslední revize 11. 04. 2012 [cit. 24. 4. 2012]. Dostupné z: http://cs.wikipedia.org/w/index.php?title=Framework&oldid=8388377

Framework. In: *Wikipedia: The free encyclopedia* [online]. 2001–2012, Datum poslední revize 9. 04. 2012 [cit. 24. 4. 2012]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Framework&oldid=486421642

Framework. In: *DocForge* [online]. 24. 8. 2011 [cit. 27. 4. 2012]. Dostupné z: http://docforge.com/wiki/Framework

What is PHP?. *PHP: Hypertext Preprocessor* [online]. 27. 4. 2012a [cit. 27. 4. 2012]. Dostupné z: http://www.php.net/manual/en/intro-whatis.php

Usage of server-side programming languages for websites. *W3Techs*: *World Wide Web Technology Surveys* [online]. 27. 4. 2012 [cit. 27. 4. 2012]. Dostupné z: http://w3techs.com/technologies/overview/programming_language/all

What can PHP do?. *PHP: Hypertext Preprocessor* [online]. 27. 4. 2012b [cit. 27. 4. 2012]. Dostupné z: http://www.php.net/manual/en/intro-whatcando.php

BERNARD, Borek. Úvod do architektury MVC. *Zdroják*: *o tvorbě webových stránek a aplikací* [online]. 7. 5. 2009 [cit. 27. 4. 2012]. Dostupné z: http://www.zdrojak.cz/clanky/nette-framework-mvc--mvp/

Výsledky ankety: mezi čtenáři vede klasika – XHTML, PHP a MySQL. *Zdroják*: *o tvorbě webových stránek a aplikací* [online]. 20. 12. 2010 [cit. 27. 4. 2012]. Dostupné z: http://www.zdrojak.cz/clanky/vysledky-ankety-mezi-ctenari-vede-klasika-xhtml-php-a-mysql/

ZDROJÁK. *Výsledky ankety*: *mezi čtenáři vede klasika – XHTML, PHP a MySQL* [online]. 20. 12. 2010 [cit. 27. 4. 2012]. Dostupné z: https://spreadsheets.google.com/viewanalytics?formkey=dEl1ZDJhbk1hWEFLbU43Skl0eDVDSnc6MQ#chart#4

Nette Foundation [online]. 2012 [cit. 28. 4. 2012]. Dostupné z: http://nettefoundation.com/cs/

Nette Framework [online]. 2012 [cit. 28. 4. 2012]. Dostupné z: http://nette.org/cs/

DANĚK, Petr. Velký test PHP frameworků: Zend, Nette, PHP a RoR. *Root.cz*: *informace nejen ze světa Linuxu* [online]. 11. 9. 2008 [cit. 28. 4. 2012]. Dostupné z: http://www.root.cz/clanky/velky-test-php-frameworku-zend-nette-php-a-ror/

GRUDL, David. Docela šokující čísla, že?. *PhpFashion* [online]. 2. 6. 2009 [cit. 28. 4. 2012]. Dostupné z: http://phpfashion.com/

PHP and Zend Engine. *Zend*: *The PHP Company* [online]. 2010 [cit. 29. 4. 2012]. Dostupné z: http://www.zend.com/en/community/php/

About Zend Framework. *Zend Framework* [online]. 2012a [cit. 29. 4. 2012]. Dostupné z: http://framework.zend.com/about/overview

Programmer's Reference Guide: Zend_View. *Zend Framework* [online]. 2012b [cit. 29. 4. 2012]. Dostupné z: http://framework.zend.com/manual/en/zend.view.introduction.html

Symfony: *High Performance PHP Framework for Web Development* [online]. 2012a [cit. 6. 5. 2012]. Dostupné z: http://symfony.com/

STOUPA, Václav. Přehled a vývoj PHP frameworků. *Root.cz*: *informace nejen ze světa Linuxu* [online]. 28. 3. 2008 [cit. 6. 5. 2012]. Dostupné z: http://www.root.cz/clanky/prehled-a-vyvoj-php-frameworku/

Symfony: *Open-Source PHP Web Framework* [online]. 2012b [cit. 6. 5. 2012]. Dostupné z: http://www.symfony-project.org/

MySQL: *The world's most popular open source database* [online]. 2012 [cit. 3. 5. 2012]. Dostupné z: http://www.mysql.com/

Quick Start. *Dibi is Database Abstraction Library for PHP 5* [online]. 2012 [cit. 3. 5. 2012]. Dostupné z: http://dibiphp.com/cs/quick-start

ŽÁK, D. *Databázové systémy I.* Přednáška č. 10. Pardubice: Univerzita Pardubice, 2011.

VRÁNA, Jakub. Fulltextové vyhledávání v MySQL. *PHP triky*: *Weblog o elegantním programování v PHP pro mírně pokročilé* [online]. 13.9.2006 [cit. 4. 5. 2012]. Dostupné z: http://php.vrana.cz/fulltextove-vyhledavani-v-mysql.php

KUBÍČEK, Michal. *Velký průvodce SEO*: *Jak dosáhnout nejlepších pozic ve vyhledávačích*. 1. vyd. Brno: Computer Press, 2008. ISBN 978-80-251-2195-5.

VRÁNA, Jakub. Poslání zapomenutého hesla. *PHP triky*: *Weblog o elegantním programování v PHP pro mírně pokročilé* [online]. 30.6.2010 [cit. 5. 5. 201]. Dostupné z: http://php.vrana.cz/poslani-zapomenuteho-hesla.php

SCHAFER, Steven M. HTML, XHTML a CSS: *Bible pro tvorbu WWW stránek*. Vyd. 4. Praha: Grada, 2009. 648 s. ISBN 978-80-247-2850-6.

OPPEL, Andy. *Databáze bez předchozích znalostí*. Vydání první. Brno: Computer Press, a.s., 2006. ISBN 80-251-1199-7.

Příloha A – Ukázka v Nette Frameworku

Funkční aplikace se nachází na přiloženém CD ve složce *examples/NetteFramework*.

HomepagePresenter.php

```php
<?php
class HomepagePresenter extends BasePresenter // výchozí presenter
{
    public function createComponentForm() { // továrnička na vytvoření formuláře
        $form = new Formular(); // nová instance formuláře
        $form->onSuccess[] = callback($this, 'processForm'); // nastavení metody
                                                // pro zpracování
        return $form;
    }

    // metoda zpracovávající formuláře
    public function processForm(\Nette\Application\UI\Form $form) {
        $this->setView('show'); // nastavení šablony 'show'

        // naplnění proměnných v šabloně
        $this->template->text = 'Text:';
        $this->template->data = $form->values->text;
    }

}
```

Formular.php

```php
<?php
class Formular extends \Nette\Application\UI\Form { // odvození formuláře

    public function __construct(Nette\ComponentModel\IContainer $parent = NULL, $name = NULL) {
        parent::__construct($parent, $name);

        $this->addText('text', 'Text'); // přidání textového vstupu
        $this->addSubmit('submit', 'Odeslat'); // přidání tlačítka na odeslání
    }

}
```

Homepage/default.latte

```
<!DOCTYPE html>
<html lang="cs">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Nette Framework</title>
    </head>

    <body>
        {control form} {* zobrazení formuláře (volá metodu createComponentForm()
                          v HomepagePresenter.php))*}
    </body>
</html>
```

Homepage/show.latte

```
<!DOCTYPE html>
<html lang="cs">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Nette Framework</title>
    </head>

    <body>
        {$text} {* výpis proměnné text *}
        <br />
        {$data} {* výpis proměnné data, escapování podle kontextu *}
        <script>alert({$data});</script>
        <br />
        <a n:href="default">zpět</a> {* tvorba odkazu pomocí makra n:href *}
    </body>
</html>
```

Příloha B – Ukázka v Zend Frameworku

Funkční aplikace se nachází na přiloženém CD ve složce *examples/ZendFramework*.

IndexController.php

```
<?php
class IndexController extends Zend_Controller_Action // výchozí controller
{

   public function indexAction() // výchozí akce
   {
      $form = new Application_Form_Form(); // instancování formuláře
      $this->view->form = $form; // předání formuláře do šablony
   }

   public function showAction() // akce show
   {
      $this->view->text = 'Text:'; // nastavení proměnné v šabloně

      $request = $this->getRequest(); // získání aktuálního požadavku
      $form = new Application_Form_Form(); // instancování formuláře

      if ($this->getRequest()->isPost()) { // formulář odeslán?
         if ($form->isValid($request->getPost())) { // formulář správně vyplněn?
            $this->view->data = $form->getValue('text'); // nastavení proměnné
                                                         // v šabloně hodnotou
                                                         // z formuláře
         }
      }
   }
}
```

Form.php

```
<?php
class Application_Form_Form extends Zend_Form // odvození formuláře
{

   public function init()
   {
      $this->setMethod('post'); // nastavení metody

      $this->addElement('text', 'text', array( // textové pole
         'label' => 'Text:'
      ));

      $this->addElement('submit', 'submit', array( //odesílací tlačítko
         'ignore'   => true,
         'label'    => 'Odeslat',
      ));
   }
}
```

index/index.phtml

```
<!DOCTYPE html>
<html lang="cs">
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <title>Zend Framework</title>
   </head>

   <body>
      <?php
      // nastavení akce formuláře
      $this->form->setAction($this->url(array('action' => 'show')));
      // výpis formuláře
      echo $this->form;
      ?>
   </body>
</html>
```

index/show.phtml

```
<!DOCTYPE html>
<html lang="cs">
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <title>Zend Framework</title>
   </head>

   <body>
      <?php echo $this->escape($this->text); // výpis proměnné text?>
      <br />
      <!-- výpis proměnné data, escapování jen podle HTML -->
      <?php echo $this->escape($this->data); ?>
      <script>alert("<?php echo $this->escape($this->data); ?>");</script>
      <br />
      <!-- tvorba odkazu -->
      <a href="<?php echo $this->url(array('action'=>'index')); ?>">zpět</a>
   </body>
</html>
```

Příloha C – Ukázka ve Frameworku Symfony

Funkční aplikace se nachází na přiloženém CD ve složce *examples/Symfony*.

DefaultController.php

```php
<?php
namespace Sara\TestBundle\Controller;
use Sara\TestBundle\Form\Formular;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\RedirectResponse;

class DefaultController extends Controller
{
   public function indexAction() // výchozí akce
   {
      $form = $this->createForm(new Formular()); // vytvoření formuláře
      // vykreslení požadované šablony s předanými proměnnými
      return $this->render('TestBundle:Default:index.html.twig',  array('form' => $form->createView()));
   }

   public function showAction() // akce show
   {
      $form = $this->createForm(new Formular()); // vytvoření formuláře

      $request = $this->get('request'); // získání požadavku
      if ($request->getMethod() == 'POST') {  // kontrola použité metody
         $form->bindRequest($request); // naplnění formuláře daty z požadavku
                                       // formulář může být vázán na objekt, ten by byl také naplněn
         if ($form->isValid()) { // kontrola validity
            $values = $form->getData(); // získání dat; nepracuje se s objektem
            $data = $values['Text:'];
            // vykreslení šablony s parametry
            return $this->render('TestBundle:Default:show.html.twig',  array('data' => $data, 'text' => 'Text:'));
         }
      }
      return $this->redirect($this->generateUrl('default')); // přesměrování na výchozí stránku
   }
}
```

Formular.php

```
namespace Sara\TestBundle\Form;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilder;

class Formular extends AbstractType
{
   public function buildForm(FormBuilder $builder, array $options) // továrna na formulář
   {
      $builder->add('Text:', 'text', array('required' => false)); // přidání textového pole
   }

   public function getName() // vrací jedinečné jméno
   {
      return 'formular';
   }
}
```

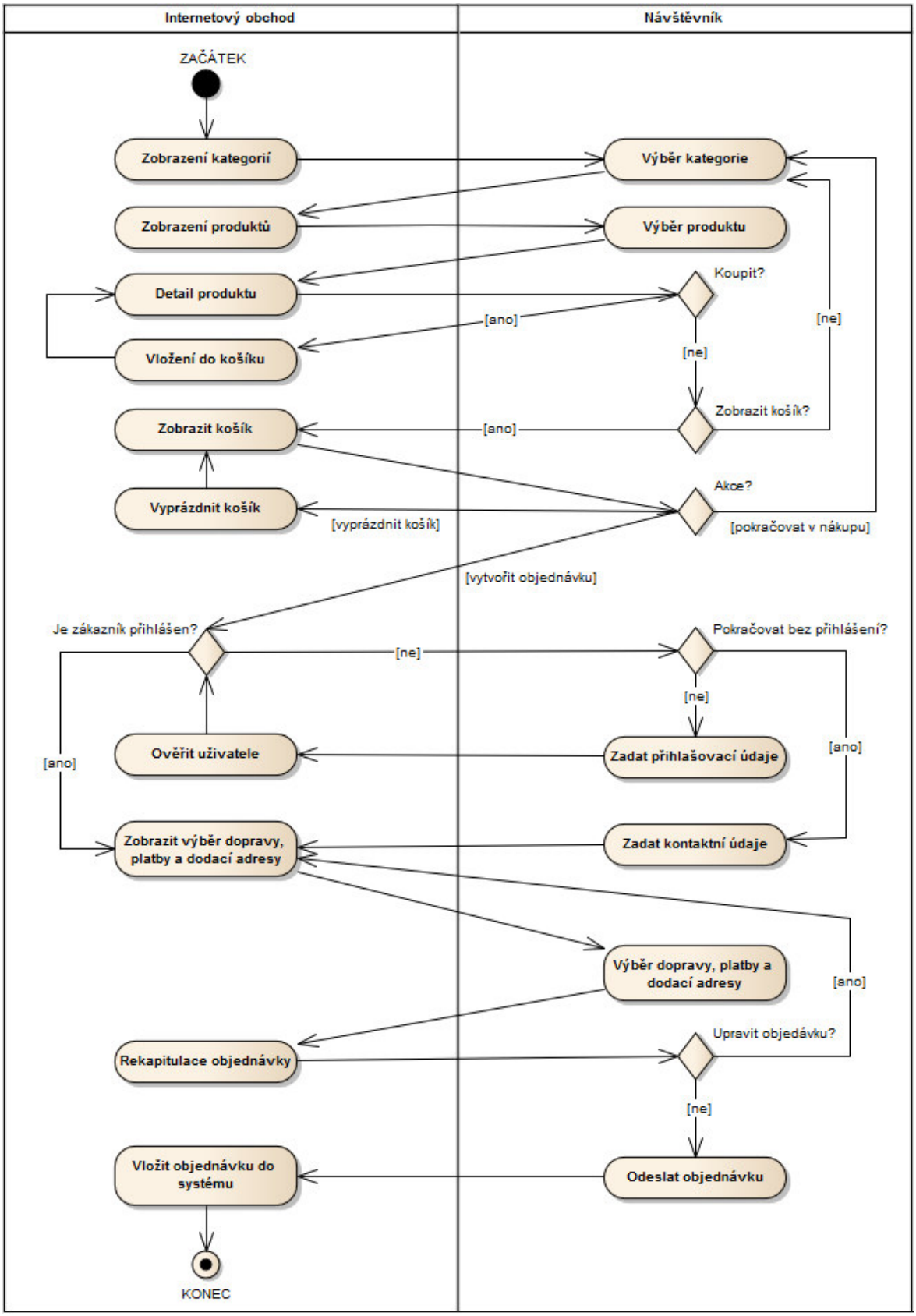
Index.html.twig

```
<!DOCTYPE html>
<html lang="cs">
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <title>Framework Symfony</title>
   </head>

   <body>
      <!-- zobrazení formuláře -->
      <form action="{{ path('_show') }}" method="post">
         {{ form_widget(form) }}
         <input type="submit" value="Odeslat" />
      </form>
   </body>
</html>
```

show.html.twig

```
<!DOCTYPE html>
<html lang="cs">
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <title>Framework Symfony</title>
   </head>

   <body>
      {{ text }} <!-- výpis proměnné -->
      <br />
      {{ data }} <!-- výpis proměnné data, escapování je automatické -->
      <script>alert('{{ data }}');</script>
      <br />
      <a href="{{ path('_default') }}">zpět</a> <!-- tvorba odkazu podle názvu routy -->
       </body>
</html>
```

Příloha D – Diagram aktivit – proces nákupu



Příloha E – Databázový model