VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practise in the Company

2010

Matěj Glos

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 20. dubna 2010 .....

Rád bych na tomto místě poděkoval Janu Šmídovi, že mi umožnil absolvovat praxi ve firmě a za veškerý čas, který mi během praxe věnoval.

Abstrakt

Tato bakalářská práce popisuje průběh praxe vykonávané studentem ve firmě fv.cz, s.r.o. Firma fv.cz působí v oblastech vývoje internetových aplikací. Můžeme se zde dozvědět o různých principech a technologiích používaných při tvorbě internetových stránek. Bakalářská práce se zaměřuje hlavně na vývoj internetových stránek pod programovacím jazykem PHP, student ve firmě pracoval jako PHP programátor. V práci jsou rozebrány úkoly přidělené studentovi v rámci praxe. Je zde popsána implementace různých rozšíření internetového obchodu deju-jewelry.com. Dozvíme se jak implementovat platební bránu pro platbu platebními kartami v e-shopu, je zde rozebráno řešení hromadného odesílání e-mailů a část práce se zabývá i optimalizací internetových stránek.

Klíčová slova: fv.cz, Internetová stránka, Internetový obchod, PHP, SQL, HTML, e-mail, Framework, Knihovna ester, Nette Framework, Newsletter, Šablonovací systém, Platební brána, Platba platebními kartami, GP webpay, 3D Secure, Elektronický podpis, Certifikáty, Optimalizace, Cache

Abstract

The aim of this thesis is to describe the working practices of the student in fv.cz, s.r.o company. Fv.cz, s.r.o company is active in the development of Internet applications. We can learn there about different principles and technologies used for creating websites. This thesis focuses mainly on the development of websites under the PHP programming language student working in the company as a PHP programmer. In this thesis, the tasks assigned to a student are discussed, in relation to the experience in the implementation of the expansion of the e-shop deju-jewelry.com We can learn how implement a payment gateway for credit cards payments in an e-shop. And also there is analysed solution of mass sending of emails and part of work is also about optimalisation of websites.

Keywords: fv.cz, Internet page, Internet shop, PHP, SQL, HTML, e-mail, Framework, Library ester, Nette Framework, Newsletter, Template system, Payment gateway, Credit cards payments, GP webpay, 3D Secure, Electronic signature, Certificates, Optimalization, Cache

Seznam použitých zkratek a symbolů

PHP	–	PHP: Hypertext Preprocessor
SQL	–	Structured Query Language
HTML	–	HyperText Markup Language
XHTML	–	eXtensible Hypertext Markup Language
CSS	–	Cascading Style Sheets
GPL	–	General Public License
GNU	–	GNU's Not Unix!
WWW	–	World Wide Web
HTTP	–	Hypertext Transfer Protocol
HTTPs	–	Hypertext Transfer Protocol Secure
SVN	–	Subversion
GZIP	–	GNU ZIP
W3C	–	World Wide Web Consortium
RFC	–	Request For Comments

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Bakalářská práce pojednává o průběhu praxe absolvované ve firmě fv.cz, s.r.o. (dále jen fv.cz). Tato firma působí především v oblastech tvorby internetových stránek. Ve firmě jsem pracoval jako PHP programátor a byly mi přidělovány úkoly spjaté hlavně se servisem nebo rozšiřováním funkčnosti internetových stránek. V teto práci se hovoří hlavně o řešení úkolů na internetovém obchodě deju-jewelry.com, který mi byl dán na starost.

Nejprve se podrobněji seznámíme se zaměřením a působením firmy. Dále práce obsahuje přehled zadaných úkolů a postup jejich řešení. Na úplný závěr jsou shrnuty zkušenosti získané v průběhu řešení zadaných úkolů a také celkové zhodnocení průběhu praxe.

V této práci se můžeme seznámit s postupem při implementaci platební brány, je zde popsán komplexní systém na odesílání e-mailů a probráno řešení různých rozšíření prováděných na e-shopu deju-jewelry.com. Nakonec jsou zde rozebrány kroky pro optimalizaci rychlosti celého projektu.

2 Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta

Hlavni činnosti firmy fv.cz je tvorba www stránek na klíč. Na základě různorodých požadavků od klienta se vytvoří analýza a následně se zpracuje. Firma realizuje internetové prezentace od menších www stránek až po složitější řešení jako e-shopy, redakční systémy a portály.

Firma poskytuje také servisní služby: údržbu obsahu a designu www stránek, oprava a údržba aplikací 3. stran aj. Zabývá se také sestavováním a správou serverů a sítí.

Firma se zabývá programováním aplikací převážně v jazyce PHP. Ve firmě jsem pracoval jako PHP programátor. Mezi mou práci patřilo sestavování nových www stránek, rozšiřování jejich funkčnosti či servis již hotových stránek.

PHP je skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek. Při realizaci zadaných úkolů jsem nevyužíval jen jazyk PHP, ale i jiné jazyky s nim spojené jako SQL, XHTML a CSS.

SQL je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. Většina internetových aplikací využívá databázi MySQL, což je multiplatformní databáze dostupná jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí. XHTML je značkovací jazyk pro tvorbu hypertextových dokumentů v prostředí WWW. Je to nástupce jazyka HTML. Předpokládá se jeho užití společně s CSS, které nám umožní dosáhnout potřebného grafického zobrazení. CSS je tedy jazyk pro popis způsobu zobrazení www stránek.

3 Úkoly zadané studentovi v průběhu odborné praxe a zvolený postup řešení zadaných úkolů

V rámci praxe mi byla přidělena práce na e-shopu deju-jewelry.com. Tento projekt byl již rozpracován a mým úkolem bylo převzetí celého projektu, jeho servis a rozšíření o další funkce.

Jedná se o běžný elektronický obchod [2]. Obsahuje seznam výrobků rozdělených do kategorií s možností filtrování. Výrobky se přidávají do košíku a po vyplnění všech potřebných údajů se odešle objednávka k dalšímu zpracování.

Pro snadnější psaní www stránek má firma svou vlastní knihovnu nazvanou *ester* [3]. Jedná se o sadu funkcí a několika tříd pro zjednodušení tvorby menších a středních www stránek. Knihovna obsahuje šablonovací systém s podporou multi-jazyčnosti, mezivrstvu pro práci s databází, třídu pro práci s obrázky a plno dalších užitečných funkcí pro jednodušší vývoj www stránek. Tato knihovna je volně dostupná pod licencí GPL. Další vývoj knihovny byl ale už ukončen. Jedná se již o starší knihovnu napsanou převážně pro PHP verzi 4. Poslední verze knihovny byla vydána v roce 2008. Dnes firma přechází na framework Nette [4]. Nette je již plně objektově orientovaný framework napsaný v PHP 5. Je to český open source framework, který podporuje mnohé moderní technologie a koncepce.

Projekt, na kterém jsem pracoval, je napsán ještě nad knihovnou *ester*. Aplikace využívá také firemní redakční systém pro snadnou správu celého obsahu prezentace. Tento redakční systém je také postaven nad knihovnou *ester*.

V e-shopu bylo třeba dopracovat sekci „Můj účet“, určenou pro registrované zákazníky. Zde se měl zapracovat přehled všech objednávek pro jednotlivé zákazníky, přidávání výrobků do seznamu svých oblíbených výrobků, možnost přidávat přátele a sdílet s nimi své oblíbené výrobky a celková správa profilu.

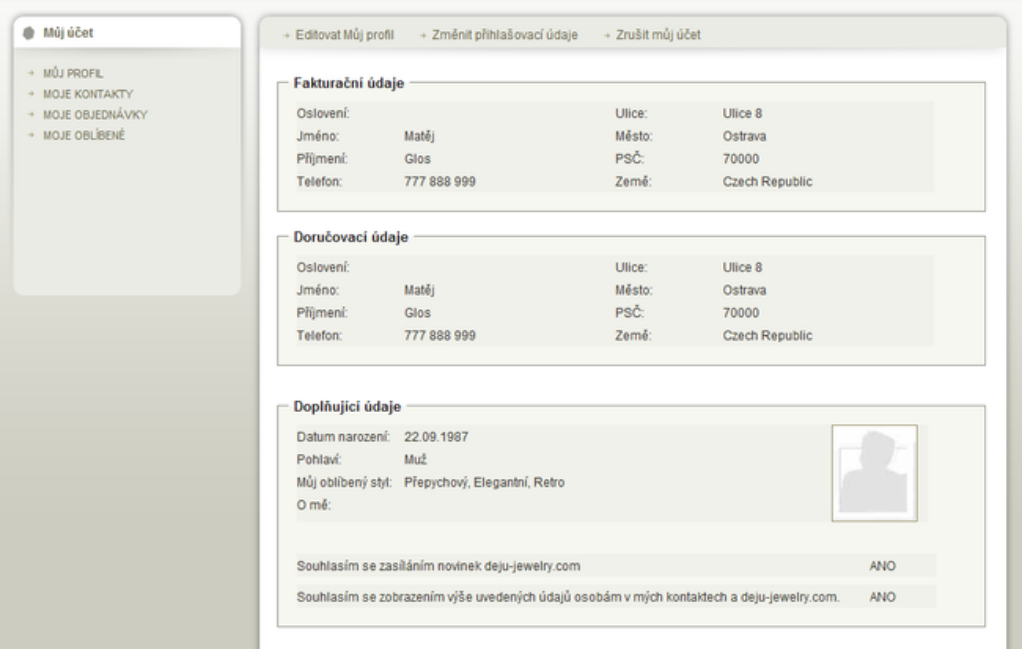
Dalšími úkoly bylo:

- vytvoření systému pro odesílání novinek tzv. newsletteru
- zapracování tzv. E-dárku, což je náhrada slevového kupónu pro nakupování zboží
- zapracování platební brány pro platbu platebními kartami v e-shopu.

Posledním, pro mě zajímavým, bodem byla celková optimalizace celé aplikace. Už když jsem projekt přebíral, velkým problémem byla velmi nízká rychlost načítání stránky.

3.1 Můj účet

Sekce „Můj účet“ je určena pro registrované uživatele e-shopu. Sekce se skládá z několika částí.



Obrázek 1: Profil uživatele

3.1.1 Profil uživatele

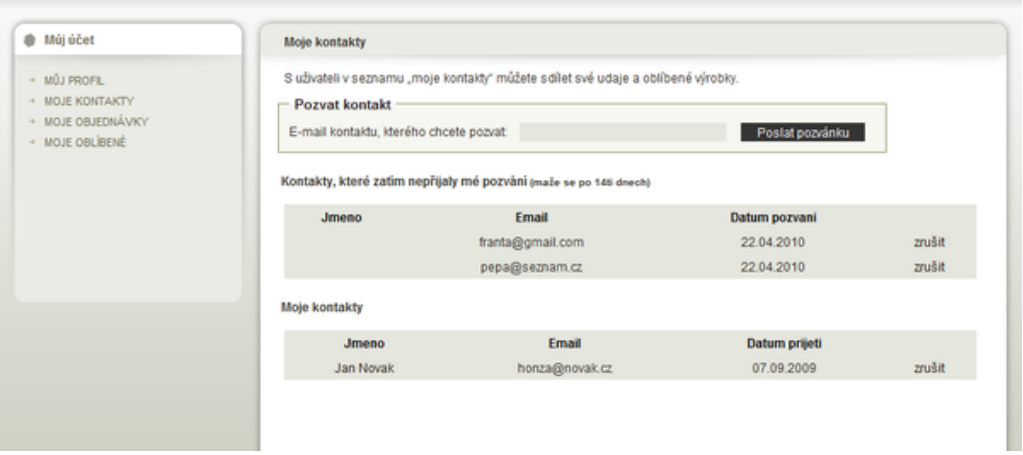
Profil uživatele měl být plně editovatelný. Nacházejí se zde fakturační údaje, doručovací údaje a doplňující údaje uživatele. V doplňujících údajích si uživatel může vložit svoji fotografii a určit bližší informace o sobě, jako datum narození, pohlaví apod. V profilu se také uživatel může přihlásit k odebírání novinek a určit zda chce zobrazovat své údaje svým přátelům.

Samozřejmostí je i změna přihlašovacích údajů, hesla a přihlašovacího jména. Požadavkem bylo použit jako přihlašovací jméno e-mail uživatele. Každý uživatel si může i svůj profil zrušit. Ukázka uživatelského profilu lze vidět na obrázku 1.

Všechny tyto údaje jsou uložené v databázi v tabulce uživatelů. V aplikaci dále bylo třeba globálně určit, které údaje uživatele se zobrazují jeho přátelům. Toto nastavení je uloženo v konfiguračním souboru celé aplikace.

3.1.2 Přátelé

Přátelé mohou v aplikaci mezi sebou sdílet své údaje a oblíbené výrobky. Nového přítele si uživatel může přidávat na základě e-mailu. Uživatel vyplní e-mail svého přítele a zvolí odeslat. Příteli se odešle e-mail vygenerovaný z připravené šablony a výzva se uloží do databáze.



Obrázek 2: Přátelé uživatele

Přítel má potom dvě možnosti:

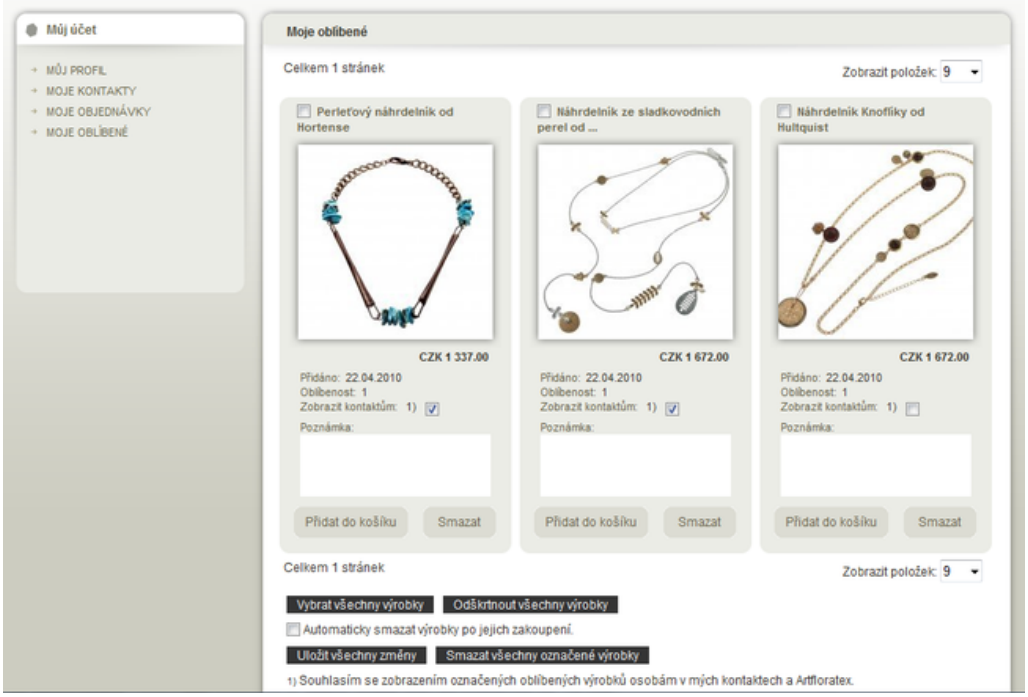
1. Přijmout pozváni kliknutím na odkaz obsažený v e-mailu. Zde se funkčnost dělí na další dvě možnosti.
   (a) Pokud je již přítel registrován v systému, přesměruje se na stránku s přihlášením do svého profilu. Po přihlášení přítel může přijmout žádost a oba uživatelé se stanou přáteli. Do databáze se uloží datum a čas přijetí žádosti a stránka se přesměruje na profil nového přítele.
   (b) Uživatel není registrován. Potom se přítel přesměruje na stránku s registrací a přítel se bude muset registrovat. Až po registraci je možné provést „spřátelení" obou uživatelů.
2. Přítel muže smazat nebo ignorovat e-mail. Potom se po určitém časovém úseku, nastaveném v konfiguraci, automaticky smaže výzva ze systému. Tento čas je nastaven v konfiguračním souboru aplikace.

Každý uživatel má přehled o všech přátelích a výzev k přátelství. Zobrazí se mu přehledný seznam všech přátel s informaci, zda přítel výzvu o přátelství přijal či ještě ne a datum přijetí žádosti. Tento přehled přátel je vidět na obrázku 2.

Po rozkliknutí již potvrzeného přítele uživatel vidí profil tohoto přítele a jeho oblíbené výrobky. Uživatel může také rušit své přátelství. Při zrušení přátelství nebo po přijetí žádosti o přátelství je uživatel vždy informován e-mailem.

3.1.3 Oblíbené

Jedná se o seznam oblíbených výrobků uživatele. Každý registrovaný uživatel si může přidávat výrobky z e-shopu do svého seznamu oblíbených. Přidání výrobku je možné v detailu každého výrobku. V profilu uživatele se potom zobrazí všechny oblíbené výrobky



Obrázek 3: Oblíbené výrobky uživatele

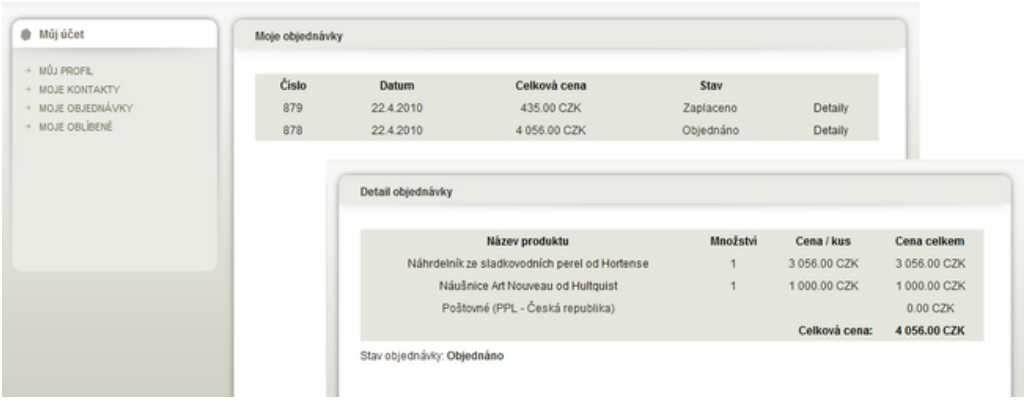
s možnosti jejich úpravy. Ke každému výrobku je možné si uložit poznámku a určit zda se má zobrazit přátelům. V tomto přehledu je možné také přidat výrobek do košíku, zobrazení detailu výrobku a smazáni výrobku z oblíbených. Sekci oblíbených vidíme na obrázku 3.

3.1.4 Přehled objednávek

Bylo potřeba také zapracovat přehled všech objednávek pro každého registrovaného uživatele. V profilu každého uživatele je možnost zobrazit přehlednou tabulku objednávek. V tabulce je zobrazen datum objednávky, celková cena, stav, způsob dopravy. Po rozkliknutí objednávky se zobrazí souhrn dané objednávky. Zobrazí se soupis všech výrobků s jejich cenou, zda byl použit slevový kupón a způsob doručení. Seznam objednávek i s detailem jedné objednávky je zobrazen na obrázku 4.

3.2 Newsletter

Dalším úkolem bylo vytvořeni systému pro zasílání novinek uživatelům pomocí e-mailu, zasílání tzv. newsletterů.



Obrázek 4: Objednávky uživatele

3.2.1 Odesílání e-mailů v aplikaci

V rámci newsletterů byl požadavek vytvořit i nějaký komplexní systém na odesílání e-mailů v celé aplikaci. Pro odesílání e-mailu existuje v knihovně *ester* jednoduchá funkce email_send. Tato funkce zajišťuje správné zakódování a odeslaní e-mailu. Požadavkem ale bylo uchovávat historii všech odeslaných e-mailu a mít jednotný, snadno editovatelný design všech e-mailu. Navíc by systém měl umět zpracovat odeslání velkého množství e-mailu, což se využije právě u newsletterů.

Pro tento účel bylo třeba vytvořit třídu pro zpracování všech těchto úkolů při odesílání e-mailu. Třída implementuje šablonovací systém, dědí ze třídy template z knihovny *ester*. Tím je zajištěno jednoduché šablonování e-mailu jednotné se šablonováním v celé aplikaci. Třída je nazvána etemplate.

K čemu slouží šablonovací systém? U větších projektů je dobré mít oddělenou aplikační vrstvu aplikace od té prezentační. HTML kodér potom není závislý na programátorovi. Může zasahovat do designu stránek, aniž by potřeboval spoluprací programátora. Kodér tedy pracuje jen s šablonami. V těchto šablonách se nacházejí tzv. tagy, značky, které jsou potom nahrazovány v PHP skriptech programátorem potřebnými daty. Toto nahrazování nám zjednodušuje právě šablonovací sytém.

Šablony jsou obvykle ukládány v souborech. U e-mailu bylo třeba editovat šablony v administraci, proto šablony e-mailu ukládáme do databáze. V administraci máme vytvořenou sekci pro správu všech šablon e-mailu.

Pro odeslání e-mailu třídou etemplate, ji musíme nejprve předal identifikátor šablony. Podle tohoto identifikátoru se šablona načte z databáze. Dále třídě předáme data, kterými chceme šablonu naplnit. Určíme příjemce, odesilatele a metodou send e-mail odešleme. Jednoduchá ukázka použití třídy etemplate je znázorněna v přikladu 3.1.

Příklad 3.1

```
// vytvoříme instanci třídy etemplate
// v konstruktoru předáme identifikátor šablony, kterou chceme použít
$mail = new etemplate('nazev_sablony');

// šablonu naplníme nějakými daty
// metodě replace předáváme název tagu a řetězec, kterým chceme tag nahradit
$mail->replace('JMENO', $jmeno);
$mail->replace('PRIJMENI', $prijmeni);
$mail->replace('TEXT', $text);

// e-mail odešleme
$mail->send($to, $from);
```

Výpis 1: Ukázka použití třídy etemplate pro odeslání e-mailu

■

E-maily se však ve třídě neodesílají ihned. Všechny e-maily jsou ukládány do speciální tabulky v databázi. Z této tabulky jsou potom na pozadí dávkově odesílány pomocí cron scriptu.

Poznámka 3.1 Cron je softwarový démon, který automatizovaně spouští v určitý čas nějaký příkaz resp. proces. Slouží jakožto plánovač úloh, jenž umožňuje opakované spouštění periodicky se opakujících procesů (nějakých skriptů).

Tato tabulka s e-maily slouží jako fronta e-mailů. Cron script tuto frontu postupně vyprazdňuje. V tabulce dále ukládáme datum vytvoření, datum odeslání e-mailu a stav e-mailu. E-mail se může nacházet ve třech stavech:

1. e-mail ještě nebyl odeslán
2. e-mail v pořádku odeslán
3. e-mail se nepodařilo odeslat

Na základě těchto stavů se v cron scriptu načtou e-maily z databáze a odešlou. Podle výsledku odeslání cron script změní stav daného e-mailu a uloží čas odeslání. Tím máme evidovanou celkovou historii všech odeslaných e-mailů. Tento přehled je dostupný v administraci aplikace. Zde můžeme také měnit stavy e-mailů a odesílat e-maily ručně.

Odesílání e-mailů v cron skriptu by mělo být co nejrychlejší. PHP skripty mají nastaven časový limit pro zpracování, obvykle je nastaven na 30s. Do tabulky proto vkládáme již zpracované a zakódované e-maily, aby se jich stihlo odeslat co nejvíce. Nejlepší by bylo uložit si rovnou všechny parametry PHP funkce **mail** [5]. Cron skript tyto údaje jen vytáhne z tabulky a pomocí této funkce hned odešle.

Funkce **mail** je definována takto:

```
bool mail ( string $to , string$subject , string $message [, string $additional_headers [,
    string $additional_parameters ]] )
```

Výpis 2: Definice funkce mail

Všechny tyto parametry by měly být správně zakódovány a e-mail by měl také obsahovat správně sestavenou hlavičku. Přesný formát e-mailu definují standardy RFC [13]. Pro vygenerování všech těchto parametrů jsem využil velice rozšířenou open source třídu PHPMailer [6]. Tato třída slouží pro jednoduché odesílání e-mailu v PHP a implementuje v sobě generování všech potřebných parametrů.

Třída etemplate umí také pracovat s přílohami v e-mailech. Dokáže zpracovat jak klasické přílohy, tak i tzv. vložené přílohy. Vložená příloha se od té klasické liší tím, že je vnořena přímo do zprávy e-mailu. Tohle je používáno především u obrázků. V e-mailu se na takovou přílohu odkazujeme pomocí speciálního identifikátoru. Pokud známe tento identifikátor můžeme obrázek zobrazit v e-mailu např. takto:

```
<img src="cid:d47844673f2db74d78da8687d794523d" alt="vnořený obrázek"/>
```

Výpis 3: Ukázka zobrazení vnořeného obrázku v e-mailu

3.2.2 Newsletter

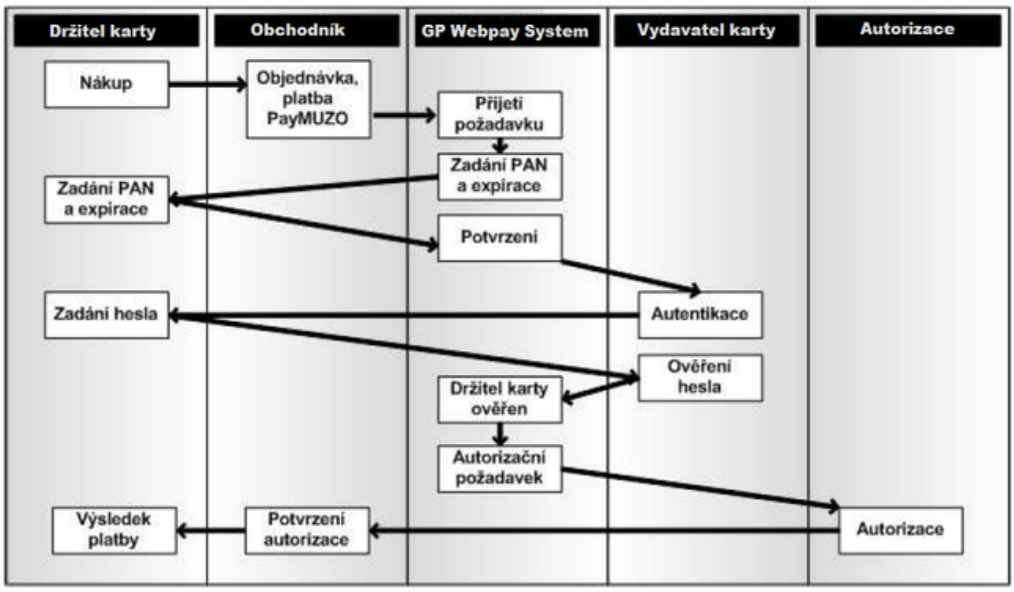
Odesílání samotných newsletteru je řešeno přes administraci. Vyberou se skupiny, na které chceme newslettery odeslat, vyplní se šablona e-mailu a po potvrzení se e-maily odešlou pomocí naší třídy etemplate, tzn. uloží se do databáze a čekají na odeslání.

Registrovaní uživatelé si mohou ve svém profilu aktivovat zasílání novinek a vybrat si skupiny newsletteru, které je zajímají. Pro odběr novinek se může přihlásit i neregistrovaný uživatel. K tomu slouží formulář na úvodní stránce e-shopu. Do formuláře se zadá jen e-mail, kam novinky zasílat, a uživatel se přidá do zvláštní skupiny. Uživatel se také může kdykoli jednoduše odhlásit ze zasílání newsletteru. To provede buď opět ve svém profilu, nebo jednoduše kliknutím na k tomu určený odkaz v samotném e-mailu.

3.3 Platební brána

Bylo potřeba zapracovat do e-shopu platební bránu pro on-line platbu platebními kartami. Pro tento účel byl vybrán systém GB webpay od společnosti Global Payments Europe Inc. [7].

Systém GB webpay je internetová platební brána, která umožňuje elektronickým obchodům přijímat platby uskutečněné platebními kartami v prostředí Internetu. Systém plně podporuje bezpečnostní standard 3D Secure.



Obrázek 5: 3D Secure

3.3.1 3D Secure

3D Secure je ochranný mechanismus pro ověření držitele platební karty. Tento standard je definován asociací VISA a MasterCard. Mechanismus zajišťuje bezpečnost tím, že údaje o své kartě nakupující neposkytuje obchodníkovi, ale přímo bance. Systém zajistí také zakódovaný přenos dat. Obchodníkovi se pouze sdělí výsledek transakce, zda byla povolena či nikoliv.

Tím je chráněn jak obchodník, tak majitel karty. Je odstraněna možnost zneužití platební karty obchodníkem, od kterého se zboží kupuje a volným přenosem nechráněných dat. Pokud 3D systém ověří totožnost držitele karty, vydavatel karty se tímto zavazuje, že nebude popírat platnost této transakce a nebude požadovat vrácení peněz od obchodníka.

Při přijetí požadavku na provedeni platby GP webpay předá požadavek na ověření autentičnosti do 3D systému. Na základě obdržení výsledku povoluje/zamítá další zpracování objednávky. Zabezpečeni 3D Secure zobrazuje obrázek 5.

3.3.2 GP webpay

Systém vytváří objekt nazvaný Objednávka, který obsahuje všechny informace nezbytně nutné pro vytváření finančních transakcí.

Komunikace s platební branou GP webpay může probíhat několika způsoby:

1. Přes dodané administrační rozhraní systému. Na určených stránkách se nachází webové rozhraní, ve kterém máme přehled všech objednávek. Objednávky můžeme vyhledávat a déle s nimi pracovat. Uhrazovat objednávky, rušit, reverzovat, stornovat aj.
2. Pomocí webových služeb - Web Services. Web Services je technologie pro vzdálené volání procedur. Standard Web Services je definován organizací W3C [8]. Do GP webpay zašleme administrativní požadavek, systém požadavek zpracuje a zašle zpět výsledek zpracování požadavku. Pomocí tohoto systému je možné integrovat kompletní funkčnost standardně poskytovaného webového administrativního rozhraní přímo do naší aplikace.
3. Vytvářet objednávky zasláním POST nebo GET požadavku do systému GP webpay. GP webpay požadavek zpracuje a vrátí nám výsledek. Mým úkolem bylo implementovat pravé tuto funkčnost.

Podrobnější popis systému, s návodem na implementaci i s ukázkou praktických příkladů, se nachází v dokumentaci dodané poskytovatelem služby.

3.3.2.1 Vytváření objednávek platebního systému Pokud se zákazník rozhodne uhradit objednávku pomocí platební brány, přesměrujeme ho na stránky GP webpay. Ukázka této stránky je na obrázku 6. Zde nakupující vyplní údaje o své kartě a potvrdí transakci. Do naší aplikace se vrátí výsledek této transakce.

Požadavek na stránku GP webpay posíláme metodou GET nebo POST. Požadavky probíhají přes zabezpečený HTTPS kanál, za použití serverového certifikátu společnosti.

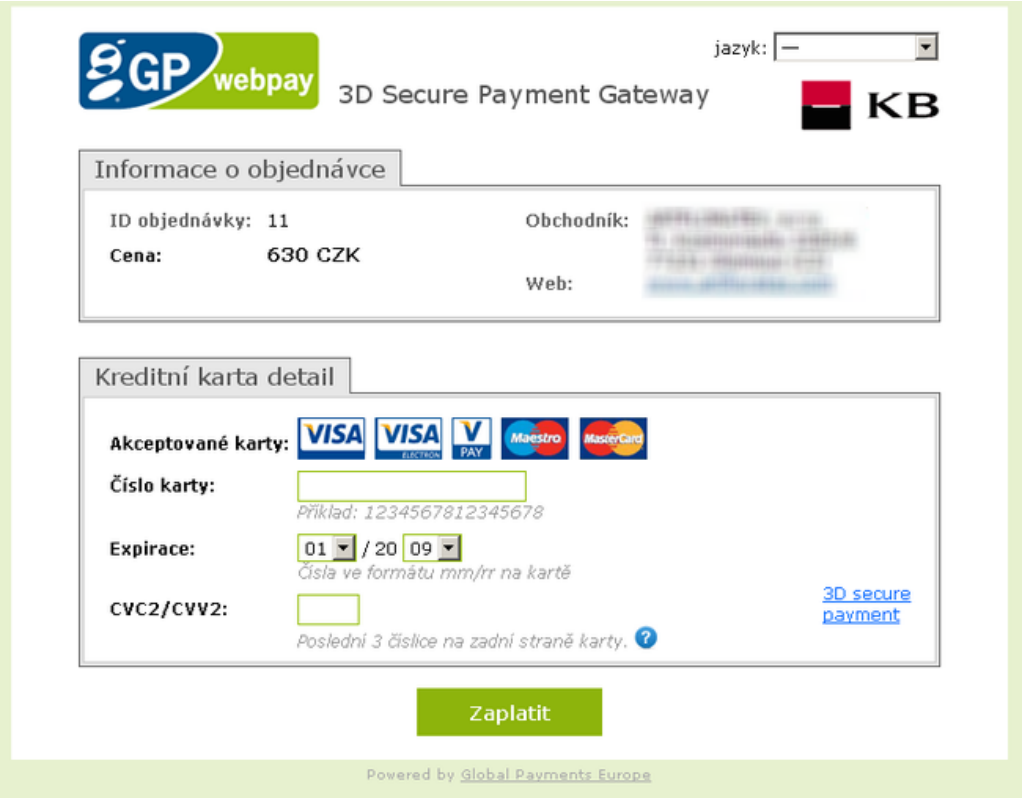
Požadavek musí obsahovat několik údajů. Pro představu uvádím jen některé:

1. přidělené číslo obchodníka, sloužící pro identifikaci našeho obchodu
2. částka k zaplacení a měna
3. URL v e-shopu, kam chceme aby nám GP webpay poslalo odpověď s výsledkem požadavku
4. kontrolní podpis, který vznikne zřetězením všech zaslaných údajů. Výpočet tohoto podpisu je popsán níže v kap. 3.3.2.2.

GP webpay zkontroluje platnost zadaných údajů. Vyhledá obchodníka, zkontroluje zaslaný podpis požadavku a pro všechny prvky zkontroluje platnost obsahu (délka, typ, hodnota). Na předanou adresu odešle odpověď s výsledkem požadavku.

Odpověď obsahuje:

1. návratové kódy s textovým popisem
2. další nezbytná data
3. samozřejmostí je opět kontrolní podpis pro ověření doručených dat



Obrázek 6: Ukázka platební brány GP webpay

3.3.2.2 Podepisování zpráv Elektronický podpis slouží k ověření, zda požadavek byl poslán oprávněným subjektem. Obsah podpisu je vypočten na základě zaslaných dat, tím je prokázáno, že obsah jednotlivých polí nebyl cestou změněn, a soukromého klíče, tím je prokázáno, že požadavek pochází od daného subjektu.

Za pomocí přiloženého programu od poskytovatele služby GP webpay si vygenerujeme soukromý a veřejný klíč s potřebnými parametry. Soukromý klíč si bezpečné uložíme. Tento klíč používáme pro podepisování našich požadavků. Veřejný klíč poskytneme poskytovateli služby GP webpay. Při přijetí libovolného požadavku GP webpay pomoci tohoto klíče zkontroluje, zda byl požadavek podepsán opravdu námi za pomoci našeho soukromého klíče.

Pro ověření zaslaných odpovědí z platební brány nám poskytovatel poskytne jeho veřejný klíč. Tímto klíčem si zkontrolujeme, že odpověď opravdu přišla od GP webpay.

Dále je naší povinností uchovávat všechny informace o úspěšných i neúspěšných verifikací podpisu. V systému musíme logovat veškeré údaje nutné k ověření podpisu. Jedná se o samotný podpis, pole která byla využita pro jeho vytvoření a výsledek ověření. V případě chybějících nebo nekompletních záznamů nám nebude možné uznat autentičnost provedených transakcí.

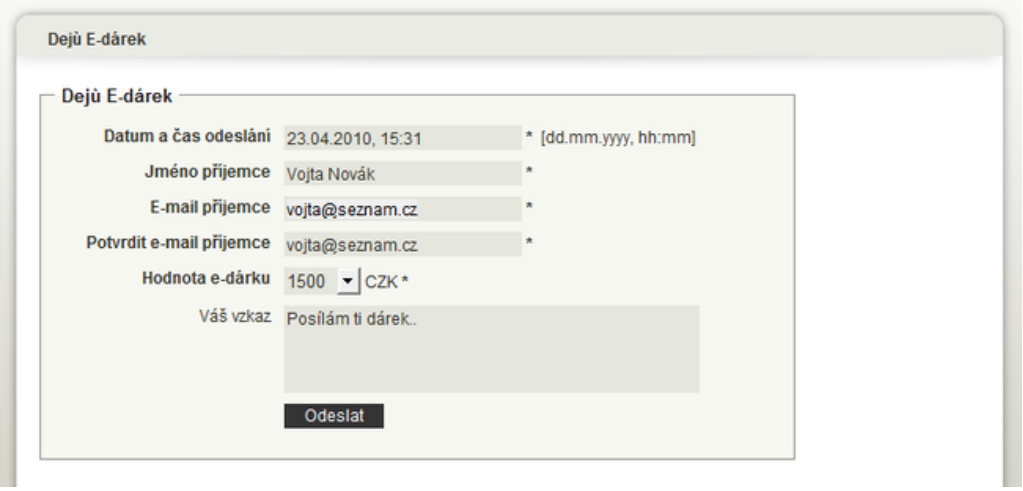
3.3.3 Popis implementace

Pro vytváření a přijímání požadavku platební brány jsem vytvořil dvě třídy. První třída je nazvaná webpay_send. Tato třída nám vygeneruje URL adresu platební brány se všemi potřebnými parametry. Na tuto adresu přesměrujeme zákazníka pro vyplnění informací o své platební kartě a následné zaplacení. Druhou třídou je webpay_receive. Tato třída se nám postará o ověření příchozí odpovědi z platební brány.

Třída webpay_send si automaticky načte všechny potřebné parametry s konfigurace aplikace. Při vytváření instance třídě stačí předat jen číslo objednávky a výše částky, kterou chceme uhradit. V případe potřeby můžeme měnit i zbylé parametry požadavku. Před vygenerováním potřebné URL třída automaticky ošetří všechna předaná data a vytvoří elektronický podpis. Zajisti nám také uložení požadavku do databáze pro zpětné dohledání a uchování historie plateb. Použití je tedy velmi jednoduché a odpadá nám hodně starostí při vytváření plateb.

Na stránce pro přijetí odpovědi využijeme druhé třídy webpay_receive. Třídě předáme všechny přijaté parametry od platební brány. Třída zkontroluje přijata data a ověří nám elektronický podpis odpovědi. Nakonec nám vrátí výsledek, zda transakce proběhla v pořádku nebo ne. Všechny odpovědi, ať už ověřené nebo neověřené, automaticky ukládá do databáze.

V administraci u každé objednávky můžeme vidět přehlednou historii všech plateb.



Obrázek 7: Formulář pro objednání E-Dárku

3.4 E-Dárek

E-Dárek slouží jako náhrada slevového kupónu při objednání zboží. E-Dárek mohou odesílat pouze registrovaní uživatelé.

3.4.1 Objednání E-Dárku

Pro objednání E-Dárku slouží v e-shopu speciální formulář, viz. obrázek 7. Zde se vyplní e-mailová adresa uživatele, kterému chceme E-Dárek zaslat, finanční výše E-Dárku a datum/čas, kdy chceme E-Dárek odeslat příjemci. Výše E-Dárku nemůže byt libovolná. V konfiguraci je definována minimální a maximální hodnota E-Dárku.

Po potvrzení se přejde na souhrnnou stránku, kde se pro jistotu zeptáme, zda opravdu chceme E-Dárek s těmito údaji a touto hodnotou odeslat. Pote přejdeme na klasickou objednávku jako při objednávání běžného zboží. E-Dárek nelze přidávat do košíku ani nakupovat současně s jiným zbožím. Požadavkem také bylo umožnit placení E-Dárku pouze platební kartou. Po odeslání objednávky se hned přejde k jeho zaplacení.

E-Dárek má tyto stavy:

1. nezaplacen
2. k odeslání/zaplacen
3. odeslán
4. přijat
5. odmítnut
6. uplatněn
7. stornován

V okamžiku potvrzení objednávky se E-Dárek uloží do databáze se stavem 1. Pokud platba proběhne v pořádku, změní se stav na stav 2 a E-Dárek je připraven k odeslání.

Pro odeslání E-Dárku využíváme třídu etemplate popsanou v kapitole 3.2.1. Tato třída musela byt rozšířena o další parametr, parametrem je čas odeslání e-mailu. Při úspěšné platbě se E-Dárek ihned odešle pomoci této třídy. Je ji však navíc předán čas, kdy chceme e-mail odeslat. E-mail se vygeneruje a s předaným časem se uloží do databáze. Na základě tohoto času cron script e-maily odesílá. E-mail se odešle jen pokud již tento čas nastal. Při úspěšném odeslání navíc změní i stav odesílaného E-Dárku na stav „odeslán".

3.4.2 Výběr E-Dárku

Příjemci E-Dárku v určený čas přijde e-mail s E-Dárkem. Aby příjemce mohl E-Dárek využít, musí ho nejprve vyzvednout a potvrdit přijetí. Příjemce může také E-Dárek odmítnout. Po potvrzení přijetí E-Dárku se uživateli vygeneruje náhodný slevový kód. Tento kód se uloží do databáze a změní se stav na „přijat". Nyní je E-Dárek připraven k použití. Pokud se příjemce rozhodne E-Dárek odmítnout, změní se stav na „odmítnut". Uživatel, který E-Dárek odeslal, je vždy informován o novém stavu e-mailem.

3.4.3 Uplatnění E-Dárku

V okamžiku objednání zboží v e-shopu je možnost vložit slevový kód. Po vložení kódu do objednávky se uplatní výše E-Dárku jako sleva z celkové ceny objednávky. Zde mohou nastat stavy:

1. **Objednané zboží v objednávce má vyšší cenu než E-Dárek se slevovým kódem.**

   Pote se úplatní celá hodnota E-Dárku, tj. hodnota se odečte z celkové ceny.

2. **Objednané zboží v objednávce má nižší cenu než E-Dárek se slevovým kódem.**

   Potom se sleva uplatní a provede se vytvoření nového E-Dárku ve výši rozdílu ceny. Tato situace je vidět i na obrázku 8. V konfiguraci je definována minimální hodnota E-Dárku. Pokud zbylá hodnota je menší než tato minimální hodnota, nový E-Dárek již není vytvořen a zbylá hodnota propadne.

Při uplatnění E-Dárku se změní stav na „uplatněno" a slevový kód již není možné dále použít.

3.4.4 Administrace

V administraci aplikace máme novou sekci s přehledem všech E-Dárků. Zde můžeme vytvářet a odesílat nové E-Dárky, upravovat je nebo stornovat.

Obrázek 8: Uplatnění E-Dárku v objednávce

3.5 Optimalizace

Velkým problémem již při přebírání tohoto projektu byla rychlost. Načítání stránky bylo extrémně pomalé. Mým úkolem bylo najít problémová místa v aplikaci a snažit se je opravit.

Optimalizace byla rozdělena do několika kroků:

1. optimalizace HTML
2. optimalizace SQL dotazů
3. optimalizace PHP skriptů

3.5.1 Optimalizace HTML

Prvním problémem byla příliš velká velikost celé stránky. Úvodní strana aplikace měla kolem 230kB, a to bez obrázků.

První možností jak zmenšit velikost načítané stránky je odstranění přebytečného kódu. Z kódu se odstraní všechny komentáře a přebytečné mezery. Minimální velikost dosáhneme úplným odstraněním všech nepotřebných bílých znaků. To se ale vždy nehodí, kód se potom stane velice nepřehledným a špatně upravitelným.

Této minimální verze se využívá především u nějakých externích knihoven. Tvůrce poskytne obvykle dvě verze své knihovny, jednu klasickou a druhou minimalizovanou.

Klasickou verzi použijeme při vývoji aplikace, kde máme k dispozici přehledný kód pro snadnější ladění aplikace. Do produkčního prostředí potom nasadíme minimalizovanou verzi.

Po odstranění přebytečného kódu a především použití minimalizovaných JavaScriptových knihoven jsem docílil zmenšení stránky na cca 150kB z původních 230kB.

Další možností je komprimovat stránku. Pomocí PHP můžeme pro kompresi použít program GZIP. Na začátku stránky, kterou chceme komprimovat zavoláme

```
ob_start ("ob_gzhandler");
```

Tato funkce nám zajistí, že po vygenerování celého obsahu stránky se tento obsah předá callback funkci ob_gzhandler, která obstará kompresi. Aby vše fungovalo, webový prohlížeč musí podporovat kompresi stránky. Funkce ob_gzhandler z hlavičky zjistí zda prohlížeč kompresi podporuje. Pokud ne, funkce vrátí **false** a stránka se komprimovat nebude. Dnes už ale veškeré prohlížeče kompresi stránek podporují.

Výsledky použití GZIP komprese:

Název souboru	Původní velikost	Velikost po kompresi
index.php	35kB	5.3kB
css.php	54kB	9.6kB

Tabulka 1: Výsledky nasazení GZIP komprese

Tím jsme docílili celkové zmenšení stránky na nějakých 80kB, což je už slušný výsledek.

Pro další zrychlení načítaní HTML stránky, bychom se měli řídit těmito pravidly:

- Sloučit všechny externí soubory do jednoho souboru. Jedná se o JavaScripty a CSS styly. Čím více souborů je třeba pro načtení stránky, tím vzniká více HTTP požadavků a načítaní se tím zpomalí. Tyto soubory déle minimalizujeme, viz. výše.
- Všechny přímo vnořené JavaScripty a CSS styly v HTML souborech přemístit do externích souborů. Externí soubory jsou většinou cachovány prohlížečem, ukládány do mezipaměti. Tyto soubory potom není třeba načítat ze serveru pokaždé, ale jen tehdy pokud došlo ke změně v souborech.
- Zapnout celkové cachování celé aplikace. O tom, že chceme stránku cachovat se prohlížeč dozví z hlavičky požadavku. Některé stránky, kde se často mění obsah, však není vhodné cachovat.

3.5.2 Optimalizace SQL

Cílem bylo snížit čas provádění všech SQL dotazů na jedné stránce pod 1s.

Pro zjištění problémových SQL dotazů bylo zapnuto jejich logování. Při každém dotazu se měřila jeho rychlost a ukládala se do databáze. Z této tabulky jme mohli zjistit nejnáročnější dotazy, na které bychom se měli zaměřit.

Obecná pravidla pro psaní SQL dotazů:

- V SELECT dotazech nepoužívat $*$ pro načtení všech sloupců. Obvykle nepotřebujeme vypisovat všechny sloupce tabulky. Vypisujeme jen opravdu ty, které potřebujeme. Pokud je třeba vypsat všechny sloupce, použijeme v dotazu také výpis jednotlivých sloupců. Databáze si potom nemusí zjišťovat seznam sloupců tabulky.
- Vyhnout se použití klauzule LIKE, pro vyhledávání v řetězcích. Zamyslet se, zda není možné použit pro vyhledávání jinou metodu.
- používat co nejméně klauzule IN nebo NOT IN
- používat klauzule typu LIMIT. Nevybírat zbytečně všechny záznamy v tabulce, pokud je nepotřebujeme.
- V klauzuli WHERE dávat na začátek obecnější podmínky. Na začátek dáváme podmínky, po kterých vypadne ze seznamu nejvíce záznamů. Systém na začátku odstraní co nejvíce řádků, které se potom v dalších podmínkách již nezkoumají.
- Používat indexy.

3.5.2.1 Indexy Index je databázová struktura, která uchovává informace o indexovaných sloupcích, které budou potom použity pro vyhodnocení dotazu. Indexy by se měly používat u všech sloupců, podle kterých se vyhledává, třídí nebo podle kterých se spojují tabulky. Data v tabulce nejsou obvykle nijak setříděny. Při vyhledávání je potřeba projít sekvenčně celou tabulku, což je časově náročné. Pro rychlejší vyhledávání si vytvoříme index nad sloupcem, podle kterého vyhledáváme.

Indexy výrazně ovlivňují výkon především u větších tabulek. U malých tabulek obsahujících řádově desítky záznamů je jejich význam zanedbatelný. Při vytváření indexu musíme brát v potaz, že při vkládání a úpravě záznamů do tabulky se musí obnovovat i indexy. To stojí určitou režii. Indexy bychom proto neměli používat u tabulek, do kterých se převážně vkládá a méně čte.

Index můžeme vytvořit nad jedním nebo několika sloupci tabulky. Při používání indexů bychom si měli dávat pozor na používání funkcí v SQL dotazech. Indexy neumí pracovat s funkcemi. Pokud použijeme v podmínce dotazu funkci, která nějak upravuje obsah sloupce (např. LOWER, UPPER), index tohoto sloupce se nepoužije.

Pro zjištění zda tabulka opravdu využívá námi vytvořené indexy můžeme v MySQL databázích použít příkaz EXPLAIN. Tento příkaz nám vypíše plán provádění daného dotazu. Ukázka je vidět v příkladu 3.2. Podrobný popis všech hodnot vygenerovaného plánu najdeme v dokumentaci [9].

Příklad 3.2
Vypíšeme si plán dotazu nad tabulkou *clanek*. Tato tabulka má vytvořený index *index1* nad sloupci *idmenu* a *publikovat*.

```
EXPLAIN SELECT nazev, obsah FROM clanek WHERE idmenu = 12 AND publikovat = 1 ORDER
 BY datum DESC;
```

vypíše plán:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	clanek	ref	index1	index1	4	const	29	Using where; Using filesort

Tabulka 2: Ukázka vygenerovaného plánu SQL dotazu

Sloupec **type** s hodnotou *ref* nám říká, že se tabulka neprochází celá. Počet řádků, které se v tabulce procházejí, je ve sloupci **rows**. V **possible_keys** je seznam indexů, které je možné na dotaz použít. Ve sloupci **key** potom máme indexy, které se opravdu použili. *Using filesort* ve sloupci **Extra** znamená, že se bude muset výsledek ještě ručně setřídit. Zde by bylo dobré přidat do indexu tabulky ještě sloupec *datum*. Potom se již výsledek dotazu nemusí třídit, setříděná data se vezmou z indexu.

■

3.5.3 Optimalizace PHP

Největší podíl na rychlost aplikace mělo zpracování PHP kódu. Bylo to dáno z části serverem, na kterém stránka běžela. Optimalizace probíhala na produkčním serveru, kde měla stránka běžet při nasazení. Tento server neměl moc dobré parametry a rychlost aplikace se na tom podepsala. Do běhu naší aplikace se počítala také zátěž i z jiných projektů, hostovaných na tomto serveru.

Pro optimalizaci PHP existuje spoustu metod a doporučení. Některé PHP funkce jsou pomalejší a doporučuje se použití jiných postupů pro dosažení výsledku. Např. je známo, že řetězce s apostrofy se zpracovávají rychleji než řetězce vložené do uvozovek. Tyto úpravy však nemají téměř žádný vliv na zrychlení. Také zmenšení velikosti PHP skriptů, pročištění kódu o zbytečné mezery a komentáře, může zrychlit zpracování PHP skriptů. Přednost by ale měla mít vždy přehlednost a čistota kódu.

Zrychlení můžeme docílit především používáním efektivních algoritmů a správným navržením databáze. Doporučuje se také vyhýbat regulárním výrazům. Nahradit je, pokud je to možné, funkcemi pro práci s řetězci. Používat vestavěné funkce namísto psaní svých vlastních - v podstatě vždy je uživatelská funkce pomalejší.

3.5.3.1 Akcelerátor Nejjednodušším a nejúčinnějším řešením by bylo nasazeni na PHP skripty nějaký akcelerátor [12]. PHP je interpretovaný programovací jazyk, na spuštění PHP skriptů je potřeba interpreta. Ten umožňuje vykonávat (interpretovat) program přímo z jeho zdrojového kódu. Tenhle způsob spouštění aplikaci má spousty výhod i nevýhod.

Jednou nevýhodou je právě menší rychlost spouštění aplikace. PHP akcelerátor funguje tak, že udržuje v paměti již předkompilované verze skriptů. Tím se značně zvýší rychlost jejich spouštění. Interpret již nemusí zpracovávat celý zdrojový kód skriptů.

PHP akcelerátory však nejsou primárně dostupné na všech hostingových serverech. Na serveru, kde běží náš projekt, akcelerátor taktéž neposkytují. Zrychleni PHP skriptů se tedy muselo docílit jinak.

3.5.3.2 Cache V aplikaci byly nalezeny dvě hlavní problémové části, které ji zpomalovaly. Tyto části zůstali v aplikaci po mém předchůdci. Jednalo se o generování menu a generování formuláře pro filtrování výrobků v e-shopu. U obou těchto částí se vytvářelo velké množství SQL dotazů a mnoho vnořených cyklů. Pro vytvoření menu bylo třeba zavolat kolem 80 SQL selectů, u filtrování to bylo dokonce něco kolem 200 dotazů.

Nejlepším řešením by bylo vymyslet lepší, efektivnější algoritmus těchto částí. Tím by se musela přepsat velká část kódu a investovat do projektu další čas. To nebylo cílem této optimalizace. Dalším řešením se nabízelo cachování.

Pro cachování byla využita třída Cache z Nette Frameworku [10]. Výhodou tohoto frameworku je, že některé jeho části můžou fungovat i samostatně.

Cachování probíhá přibližně takto:

```
// získání instance cache
$storage = new FileStorage('tmp');
$cache = new Cache($storage);

// ověření, zda jsou data v cachi
if (isset($cache['mojeCacheData'])) {

    // pokud data v cachi jsou, načtou se z cache
    $data = $cache['mojeCacheData'];

} else {

    // cache je prázdná, vygenerujeme data
    ...

    // uložíme data do cache a nastavíme expiraci
    $cache->save('mojeCacheData', $data, array(
        'expire' => time() + 60 * 10,
    ));
}
```

Výpis 4: Ukázka cachování s třídou Cache v Nette Frameworku

Do cache můžeme ukládat jakákoliv data. V našem případě je nejlepší ukládat celé HTML s vygenerovaným menu a HTML s již vygenerovaným formulářem pro filtrování. Pokud se vygenerované stránky najdou v cachi, jen se zobrazí a nemusíme je již dále zpracovávat. Výsledek cachování se nachází v tabulce 3.

Co bylo cachováno	Průměrný čas před cachovaním [s]	Průměrný čas po cachování [s]
menu	0,237187741	0,001233089
filtrování	0,294066517	0,001585817

Tabulka 3: Výsledky cachování

Data v cachi se můžou v průběhu fungování stránky měnit. Klient v administraci může upravit položky v menu nebo změnit kategorie, podle kterých se v e-shopu filtrují výrobky. Naší cachi si proto nastavíme vhodný čas, kdy chceme data vyexpirovat - zahodit z cache a načíst nová data.

3.5.4 Výsledek optimalizace

Při optimalizaci bylo zjištěno, že největší dopad na rychlost měl PHP kód aplikace. Ostatní části aplikace měly na rychlost zanedbatelný vliv. Při optimalizaci SQL bylo upraveno pár dotazů a vytvořeno několik indexů. Databáze aplikace ale není zase tak rozsáhlá, aby tyto změny měly výrazný vliv na zrychlení aplikace. Celkové zmenšení načítané stránky, popsané v kapitole 3.5.1, zrychlilo o něco načítání. Největší čas si však bralo zpracování PHP skriptů.

Celkový výsledek optimalizace můžeme vidět v tabulce 4. Veškeré testování probíhalo na produkčním serveru, tedy za podmínek, ve kterých by aplikace normálně měla fungovat. Na každé stránce bylo provedeno 100 měření, v tabulce je uveden průměrný čas těchto měření.

Měřená stránka	Čas před optimalizací [s]	Čas po optimalizaci [s]
úvodní strana	1,673802528	0,878705518
přehled výrobků	2,348721000	0,931291280
detail výrobků	1,086256639	1,065637400
klasicky článek	0,781423082	0,671412077

Tabulka 4: Výsledky optimalizace

Na zrychlení se promítlo především cachování. Stránky, na kterých se nezobrazuje menu e-shopu ani filtrování, nevidíme žádnou výraznou změnu. Jedná se o detail výrobku a klasický článek na stránce. Rozdíl je vidět už na úvodní stránce, kde je použito cachování menu. Ještě větší zrychlení jsme docílili v přehledu seznamu výrobků. Zde se z cache načítá jak menu, tak i formulář pro filtrování.

4 Teoretické a praktické znalosti a dovednosti získané studentem v průběhu odborné praxe

V rámci praxe jsem se určitě mnohé naučil. Rozšířil jsem si znalosti o různých principech a technologiích využívaných v internetových stránkách a v Internetu vůbec. Především to byla ale velká zkušenost. Získal jsem představu, jak chodí vývoj aplikací v praxi.

Před řešením všech zadaných úkolů jsem se musel naučit pracovat z knihovnou *ester*, kterou firma používá a z části i s Nette Frameworkem. Zjistil jsem jak knihovny a frameworky mohou usnadnit programování a umožňují efektivně a rychle vyvíjet software. Programátor nemusí řešit základní úlohy spojené např. s navigací ve stránce, s bezpečností, s prací s databází apod. O to se postará framework. Programátor se může zaměřit přímo na řešení zadání. Má také k dispozici plno užitečných podpůrných funkcí a tříd, které by jinak musel sám psát.

Dále bylo třeba se seznámit s verzováním projektu, tzv. Subversion (zkráceně SVN) [11]. Pro vývoj projektu se ve firmě používaly SVN repozitáře. Verzování je způsob uchovávání historie všech změn v projektu. V systému se uchovává kdo a kdy provedl jaké změny v projektu. V případě potřeby se můžeme vždy vrátit k předchozím verzím projektu. Dále SVN umožňuje práci na jednom projektu více lidem najednou. Dokáže řešit případné kolize - situace, kdy více programátorů současně změní stejné části zdrojového kódu.

V rámci řešení úkolu s odesíláním e-mailu, popsaného v kapitole 3.2.1 na straně 11, jsem se musel podrobněji seznámit se strukturou obsahu e-mailu. Musel jsem si nastudovat vytváření těla zprávy a hlaviček e-mailu. Zajistit správné kódování, správné zobrazování českých znaků. Seznámil jsem se s prací s přílohami a celkovým postupem při odesílání e-mailu.

Při implementaci platební brány jsem se blíže dozvěděl jak probíhají platby na Internetu a problémy s tím spojené. Dozvěděl jsem se více o bezpečnosti přenosu dat na Internetu, s vytvářením a prací s certifikáty.

Mnoho nových znalostí mi dalo řešení optimalizace. Seznámil jsem se s různými postupy jak zrychlit a zefektivnit aplikaci. Naučil jsem se pracovat s různými nástroji pro měření a testování optimalizace. Pronikl jsem podrobněji do problematiky vykonávání SQL dotazů a získal přehled o principech, jak správně psát optimalizovaný PHP kód.

5 Závěr

V období trvání praxe jsem se naučil plno nového a získal mnoho zkušeností, jak to v praxi může fungovat. Zjistil jsem také v jakých oblastech bych se měl zlepšovat a jaké znalosti mi chybí.

Ve firmě mi byla hlavně vyčítána nepřehlednost napsaného kódu. Kód by měl mít vždy nějakou ucelenou strukturu. Různým programátorům vyhovují různé styly psaní kódu. Měla by se však dodržovat nějaká základní pravidla. Dále mi bylo vyčítáno nedotahování úkolů do úplného konce. Vždy byla odhalena nějaká drobná chyba. Věřím, že absolvováním této praxe a získáním větších zkušeností, se do budoucna těchto nedostatku vyvaruji.

Celkově hodnotím průběh praxe dobře. Při řešení úkolů jsem vždy dosáhl kýžených výsledků, i když by samozřejmě bylo mnohé co zlepšovat. Funkční se stala sekce „Můj účet“ i s potřebnými podsekcemi, byl nasazen systém na odesílání e-mailů, bez problému jsme zprovoznili placení platebními kartami v e-shopu a slušného výsledku jsme docílili i u optimalizace rychlosti celé aplikace.

6 Reference

[1] *PHP: Hypertext Preprocessor* [online]. 2001 [cit. 2010-04-22].
Dostupné z WWW: <http://php.net/>.

[2] *Internetový obchod In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation [cit. 2010-04-22].
Dostupné z WWW: <http://cs.wikipedia.org/wiki/Internetový_obchod>.

[3] *Ester* [online]. 2008 [cit. 2010-04-22].
Dostupné z WWW: <http://devel.fv.cz/ester.demo/>.

[4] *Nette Framework* [online]. 2008 [cit. 2010-04-22].
Dostupné z WWW: <http://nettephp.com/>.

[5] *PHP: mail - Manual* [online]. 2001 [cit. 2010-04-22]. PHP: Hypertext Preprocessor.
Dostupné z WWW: <http://php.net/manual/en/function.mail.php>.

[6] *PHPMailer* [online]. 1999 [cit. 2010-04-22].
Dostupné z WWW: <http://phpmailer.codeworxtech.com/>

[7] *Global Payments Inc. Europe* [online]. 1999 [cit. 2010-04-22].
Dostupné z WWW: <http://www.globalpaymentsinc.com/Europe/Czech/>

[8] *World Wide Web Consortium (W3C)* [online]. 2010 [cit. 2010-04-22].
Dostupné z WWW: <http://www.w3.org/>

[9] *7.2.1 Optimizing Queries with EXPLAIN* [online]. 2010 [cit. 2010-04-22]. MySQL Documentation: MySQL Reference Manuals
Dostupné z WWW: <http://dev.mysql.com/doc/refman/5.0/en/using-explain.html>

[10] *Nette\Caching* [online]. 2010 [cit. 2010-04-22]. Nette Framework
Dostupné z WWW: <http://doc.nettephp.com/cs/nette-caching>

[11] *Apache Subversion* [online]. 2010 [cit. 2010-04-22].
Dostupné z WWW: <http://subversion.apache.org/>

[12] *PHP accelerator In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation [cit. 2010-04-22].
Dostupné z WWW: <http://en.wikipedia.org/wiki/PHP_accelerator>.

[13] *RFC 2822 (rfc2822) - Internet Message Format* [online]. 2009 [cit. 2010-04-22]. Internet RFC/FYI/STD/BCP Archives
Dostupné z WWW: <http://www.faqs.org/rfcs/rfc2822.html>