FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Migrace customizovaných e-shopů OpenCart 1.1
Student:	Bc. Tomáš Nováček
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem této práce je transformace zastaralého řešení e-shopů postaveného na platformě OpenCart 1.1 do moderního a snadno rozšiřitelného řešení. Cílem práce není realizovat frontend.

Postupujte v těchto krocích:
1. Analyzujte současné řešení e-shopů, které využívá společnost Jagu s.r.o.
2. Proveďte analýzu alespoň pěti současných nejvíce používaných řešení v komerční oblasti.
3. Na základě analýz zvolte a obhajte vhodnou formu realizace nového e-shopového řešení, které poskytuje funkcionality alespoň v rozsahu stávajících administrací e-shopů společnosti Jagu s.r.o., nebo se danému rozsahu nejvíce přibližuje.
4. Zajistěte možnost kompletní migrace dat starých e-shopů na vámi realizovanou platformu.
5. Pokud platforma neobsahuje veškerou již dříve využívanou funkcionalitu, navrhněte vhodná rozšíření a pokuste se o jejich realizaci.
6. Zhodnoťte rozsah a výsledek vaší práce, navrhněte vhodnou formu údržby, aby projekt nezastaral obdobným způsobem jako e-shopy společnosti Jagu s.r.o.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 11. října 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Migrace customizovaných e-shopů OpenCart 1.1

Bc. Tomáš Nováček

Katedra Softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

13. prosince 2018

Poděkování

Chtěl bych poděkovat svému vedoucímu Jiřímu Hunkovi za skvělé vedení a hlavně za možnost dělat diplomovou práci v jeho firmě. Zároveň bych chtěl poděkovat Janu Matouškovi za informace o projektu a asistenci při vývoji aplikace a Oldřichu Malcovi a Pavlu Kovářovi za technologické zázemí. Mé díky také patří Tereze Martinovské a Kryštofovi Slavíkovi za jazykové korektury. V neposlední řadě bych chtěl poděkovat Apakrychlím[1], že se uvolily být v pravém dolním rohu některých mých stránek.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. prosince 2018 .....

Odkaz na tuto práci

Nováček, Tomáš. *Migrace customizovaných e-shopů OpenCart 1.1*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Cílem této práce bylo zvolení nové platformy na tvorbu e-shopů a následná migrace e-shopů společnosti Jagu s.r.o. na tuto platformu.

Byl proveden průzkum nejpopulárnějších platforem na trhu a jejich porovnání s potřebami zákazníka. Analýza se zaměřovala převážně na pokrytí požadavků na nový systém a možnosti migrace na něj.

Následně byl vybrán a upraven modul na migraci databáze, při které byl kladen důraz na validaci dat, jejich čištění a převedení do nové databáze. Dále byl navržen postup implementace nových funkcí, aby úpravami platformy nebyly znemožněny její automatické aktualizace. Na základě tohoto návrhu byl vytvořen modul, který administraci PrestaShopu co nejvíce přiblížil potřebám zákazníka.

V závěru práce jsou uvedena další možná vylepšení platformy, mezi kterými je nejdůležitější postupný přepis modulu do Symfony frameworku, až to bude PrestaShop umožňovat. Dále je potřeba přemigrovat funkce, které byly do původního e-shopu v posledních letech přidány.

Klíčová slova e-shop, OpenCart, migrace, PrestaShop, databáze, modul

Abstract

The objective of this thesis was to select a new platform for e-shop creation and the follow-up migration of Jagu s.r.o. e-shops onto this platform.

An analysis of the most popular platforms on the market was made together with its comparison with the customer's needs. The analysis was focused on covering the requirements on the new system as well as the possibility of migration on it.

Subsequently, a migration module for the database was selected and modified. The emphasis was placed on data validation, cleaning and transfer to a new database.

Additionally, the process of implementing new features has been designed to make adjustments to the platform, so the adjustments do not prevent from automatic updates.

Based on this proposal, a module has been created to bring PrestaShop administration as close as possible to the needs of the customer.

At the end of the thesis are mentioned other possible improvements of the platform, among which the most important is the reimplementation of the module into the Symphony framework as soon as PrestaShop enables it. It is also necessary to reimplement the functions that were added to the original e-shop in recent years.

Keywords e-shop, OpenCart, migration, PrestaShop, database, module

Obsah

Seznam obrázků

Seznam tabulek

Seznam ukázek kódu

Úvod

I přes to, že je různých platforem na tvorbu e-shopů mnoho a jejich počet se každým dnem zvyšuje, jejich kvalita stále není ideální. E-shopy se vytváří s důrazem na frontend, na jeho líbivost a uživatelskou přístupnost, jako je přehledné zobrazení informací, dobré filtrování či jednoduché objednávání, nicméně kvalita administrace je na tom škodna. Často je backend příliš komplikovaný, neintuitivní, ale hlavně mu chybí modularita, možnost jednotlivé komponenty přidávat, odebírat či jednoduše upravovat.

To může být mnohdy problém, neboť každý e-shop je jiný, stejně tak potřeby jeho majitele a zákazníků. Je tedy běžné na e-shopu před jeho spuštěním (a často i po něm) dělat nějaké změny, aby co nejvíce vyhovoval potřebám všech.

Dnešní generace platforem na vytváření e-shopů na to není úplně připravená. Spousta z nich neumožňuje jednoduché úpravy vůbec, další pouze s velkým úsilím, v případě větších úprav dokonce za cenu ztráty automatických aktualizací.

V době, kdy se velké procento nákupů provádí přes internet, je nemožnost vytvoření customizovaných e-shopů velkým problémem. Ten bude v nejbližší době nutné řešit, pokud se má stát tvorba e-shopů alespoň trochu přístupnější jak pro vývojáře, tak pro běžné podnikatele.

Kapitola 1

Cíl práce

Cílem rešeršní části této práce bylo analyzovat současný stav customizovaných e-shopů OpenCart od firmy Jagu s.r.o. Bylo potřeba zjistit, jaké jsou jejich silné a slabé stránky, nejčastěji používané funkce, popřípadě funkce, které by uživatelé využili, ale v tuto chvíli je systém nepodporuje.

Dalším krokem byla analýza trhu, konkrétně alespoň pěti nejpoužívanějších platforem pro vytváření e-shopů a jejich porovnání s potřebami zákazníka. Dle nich bylo potřeba rozhodnout, zda vybrat a upravit jednu z analyzovaných platforem, či nový e-shop napsat od základů.

Cílem praktické části této práce bylo přemigrovat původní e-shopy na platformu zvolenou v rešeršní části. Pokud byla zvolena již existující platforma, bylo potřeba ji upravit tak, aby se funkcemi co nejvíce blížila původnímu e-shopu.

Při migraci bylo důležité zaměřit se převážně na korektní převod dat z jedné databáze do druhé, stejně tak jako zachování možnosti automatických aktualizací.

Kapitola 2

Současný stav

V této kapitole je probrána historie projektu administrace customizovaných e-shopů OpenCart 1.1 od jejího založení až po aktuální stav, v jakém mi byl projekt předán. Dále jsou rozebírány jednotlivé funkce aplikace, jak současné, tak plánované.

2.1 Historie

Firma Jagu s.r.o. je na trhu již přes deset let. Za tu dobu vytvořila mnoho projektů s různým zaměřením, nicméně pro potřeby této práce je mnohen důležitější jejich technologické zázemí a historie.

2.1.1 Jagu s.r.o

Firma Jagu s.r.o., založená 26. 7. 2008 [11], se zaměřuje na „webové stránky, internetové obchody, firemní informační systémy, aplikace, grafické práce“ [12]. Mezi jejich produkty patří takové weby jako Stylka, designový a stylový e-shop s domácí elektronikou, zaměřený na zdraví, péči o tělo a domácnost [12], či Můj Remington, internetový obchod s výrobky Remington [12].

Všechny její e-shopy jsou založené na platformě OpenCart [13], konkrétně verzi 1.1.1. Pro odlišení od originální verze OpenCartu budu v případě možné nejasnosti používat pro aktuální verzi vyvíjenou Jagu s.r.o. označení customizovaná verze administrace či customizovaná verze e-shopu.

2.1.2 OpenCart

OpenCart je bezplatný open-source management systém pro online obchody, napsaný v jazyku PHP s databází založenou na MySQL [13].

2.1.2.1 Vývoj OpenCartu

Nejdříve byl OpenCart vyvíjen v jazyce Perl, později byl však přepsán do již zmíněného PHP, ve kterém také vyšla jeho první stabilní verze 1.1.1 v roce 2009, tehdy běžící na PHP 5.2 a MySQL 5.0.17.

Dlouhou dobu byly pro OpenCart vydávány pouze malé patche, další velká verze, 2.0, vyšla až v roce 2014. Následovala ji verze 2.2 v březnu 2016, která byla z velké části vyvíjena OpenCart komunitou [14].

V červenci 2017 byla vydána zatím poslední stabilní verze 3.0.2.0.

2.1.2.2 Struktura

OpenCart poskytuje frontend i backend e-shopu, obojí je možné díky open-source kódu upravovat, popřípadě rozšiřovat pomocí již hotových modulů přes OpenCart Extension Store.

Backend i frontend jsou postaveny na návrhovém vzoru MVC-L – Model-View-Controller-Language – rozšíření klasického MVC návrhového vzoru. Odlišuje se podporou lokalizace, neboť jsou všechny texty ukládány do speciálních lokalizačních souborů, ze kterých se pak překlady získávají a zobrazují dle zvoleného jazyka e-shopu.

Uživatel tedy vidí View, kde každá jeho interakce s aplikací vyvolá akci v Controlleru, ten následně upraví/získá data skrz Model, který je již napřímo připojen do databáze. Tato modularita umožňuje rozdělit business logiku aplikace do více vrstev, popřípadě kteroukoliv z vrstev úplně vyměnit (například Model), aniž by se musel upravovat View či Controller.

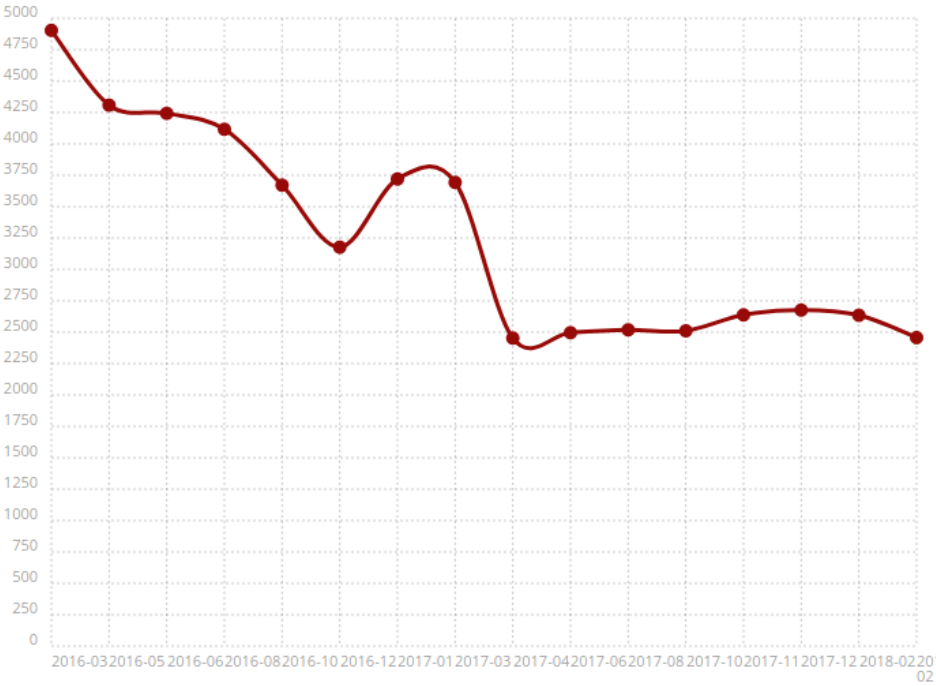
2.1.2.3 Podíl na trhu

OpenCart v tuto chvíli zabírá pouze 0.1 % trhu, přestože se za poslední rok počet e-shopů založených na OpenCartu zvýšil o 43 %. Na druhou stranu jeho podíl na milionu největších e-shopů za poslední dva roky klesl z 0.5 % na 0.25 % (obr. 2.1).

V porovnání s tím má například WooCommerce, e-commerce[1] systém založený na WordPressu, bez mála 25% podíl na trhu [3], což jej činí nejpoužívanější e-commerce platformou.

Na druhém místě je Shopify, které má 11.21% podíl [3], následovaný Squarespace s 7.21 % [3].

[1] E-commerce (někdy též e-komerce) je poměrně široký pojem používaný k označení veškerých obchodních transakcí realizovaných za pomoci internetu a dalších elektronických prostředků. E-commerce je tedy společně s dalšími „éčky“ (jako například e-learning) součástí e-business (elektronického podnikání). [15]



Obrázek 2.1: Graf podílu OpenCartu na milionu největších e-shopů za období 2/2016–2/2018 [2]

2.1.3 Vývoj administrace

Vývoj customizovaných OpenCart projektů byl primárně cílen na úpravy designu frontendu a vizuální stránka administrace zůstala prakticky stejná jako v původní verzi OpenCart 1.1.1.

Avšak ani funkce administrace nebyly opomíjeny. Z počátku šlo zejména o implementaci exportů do účetnictví a návrh a vývoj systému přetěžování metadat v závislosti na zvolené doméně.

Postupem času byly přidány lepší grafy (konkrétně byl vývoj zaměřen na filtry dat), a to v průběhu několika let ve dvou iteracích. Další velkou změnou byl nástroj pro hromadné přecenění spolu s importem dat o cenách konkurence z online nákupního rádce Heureka.

Také bylo několik customizovaných administrací propojeno s interním skladovým systémem Sysel a spolu s tím byl vytvořen funkční stavový automat objednávek.

Následovalo propojení s dopravcem Geis, tedy bylo implementováno vytváření zásilek, posílání balíků a tisk štítků na termotiskárně s protokolem EPL2[2]. To stejné bylo později implementováno pro Českou poštu a posléze

[2]EPL – Eltron Programming Language (EPL a EPL2) je jazyk na ovládání tiskáren, využívaný k tisku papírových štítků pro různé tiskárny modelu Eltron (nyní Zebra) [16].

i pro Zásilkovnu.

Přibližně ve stejné době byl přidán systém parametrů produktů, opět v několika iteracích (v první iteraci byly parametry vázané na kategorii, dnes už je vytváření parametrů volnější).

Průběžně byly také upravovány různé exporty, tabulky a přehledy, ale šlo spíše o drobnosti pro přehlednější a jednodušší práci než přidávání nové funkcionality.

V roce 2016 byl přidán speciální nástroj pro převod sortimentu do nových e-shopů s upraveným designem. Zároveň došlo k migraci systému z dříve využívaného FTP do git repozitáře.

O rok později bylo do systému implementováno EET v rámci příprav na druhou fázi závádění elektronické evidence tržeb.

Aktuálně celý e-shop běží na PHP 7.1, MariaDB 10.

2.2 Funkce aplikace

Administraci se od jejího nasazení v roce 2009 dostalo mnoho změn oproti originální verzi OpenCartu 1.1.1. Spousta z nich byla napsána tak, aby byla co nejdříve nasazena a schopna provozu, bez ohledu na prohřešky proti programovacím standardům. Kód byl tak na některých místech nepřehledný, neefektivní, neintuitivní či obsahoval duplicity, ať v rámci administrace či mezi backendem a frontendem.

Stále bylo mnoho změn plánovaných, ale zatím neimplementovaných, převážně kvůli nedostatku času. Některé změny byly kritické pro jednoduchou práci s aplikací, ale většina z nich byla pouze nice-to-have.

Zároveň zákazníci nevyužívali všechny funkce, které obsahovala původní verze OpenCartu, bylo tedy možné je odstranit a ulehčit tak uživatelskému rozhraní a kódu.

Vzhledem k počtu funkcí, které customizovaná administrace poskytovala, by bylo zbytečné je zde všechny vypisovat. Ze schůzek s Jiřím Hunkou vyplynulo, že všechny aktuální funkce v jakékoliv další verzi musí zůstat, pokud u nich nebude explicitně řečeno jinak.

Následuje tedy seznam funkcí, které by měly být v dalších verzích administrace přidány/odebrány. Požadavky označuji zkratkou PO a přidělil jsem jim číslo tak, jak jdou za sebou, abych se na ně mohl lépe odkázat v případě potřeby. Rozdělil jsem je na tři typy:

- **DEL** – funkce je v současném systému, ale není využívaná a může být smazána,
- **MUST** – funkce, která v systému není, ale musí být v jakékoliv další verzi,
- **NICE** – funkce, která v systému není, ale její implementace by zjednodušila práci s aplikací.

2.2.1 Obecné

Funkce, které se vztahují na celou aplikaci.

- **PO1** Základní nastavení e-shopu (titulek, název) pouze v kódu (ne v nastavení aplikace) [**NICE**]
- **PO2** Možnost mít více rolí pro jednoho uživatele [**NICE**]
- **PO3** Role superadmin [**NICE**]

2.2.2 Nástěnka (dashboard)

Nástěnka je stránka, která se zobrazí po přihlášení do administrace. Měl by na ní být přehled toho nejdůležitějšího z celého e-shopu.

- **PO4** Zobrazovat pouze důležité informace [**MUST**]
- **PO5** Omezit informace podle role [**MUST**]
- **PO6** Vizuální indikace nedokončených objednávek [**MUST**]
- **PO7** Přehled produktů, co se hodně prodávají, ale nejsou skladem [**MUST**]

2.2.2.1 Přehledový graf

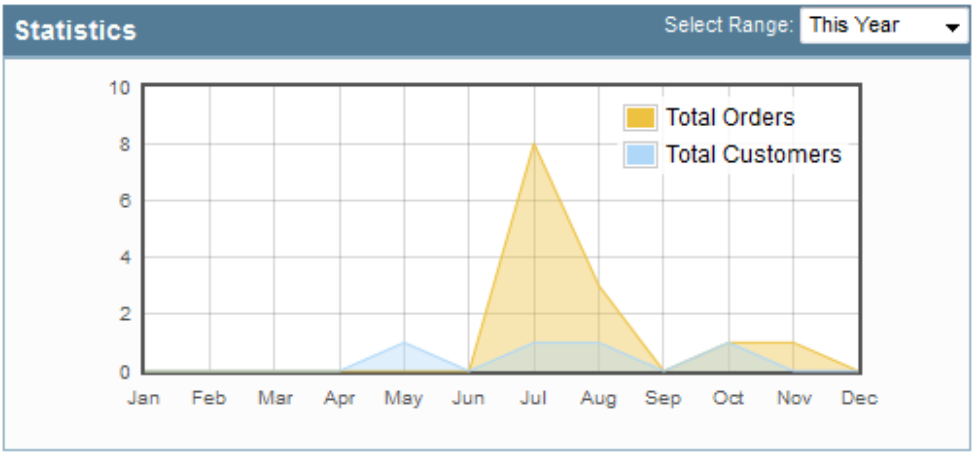
Dle diskuze s Jiřím Hunkou je přehledový graf jedním z nejdůležitějších a zároveň nejkomplikovanějších prvků celé administrace. Jde o graf zobrazující takové údaje, jako jsou prodeje či návštěvnost webu za určité období.

Od svého nasazení v roce 2009 se mu dostalo mnoho změn, jak je vidět při porovnání obrázků 2.2 (graf z původní administrace OpenCart 1.1.1) a 2.3 (customizovná verze). Graf z nejnovější verze OpenCartu 3.0.2 se tomu původnímu velmi podobá, jediná změna je totiž v moderním designu, funkcionalita však zůstává nezměněna.

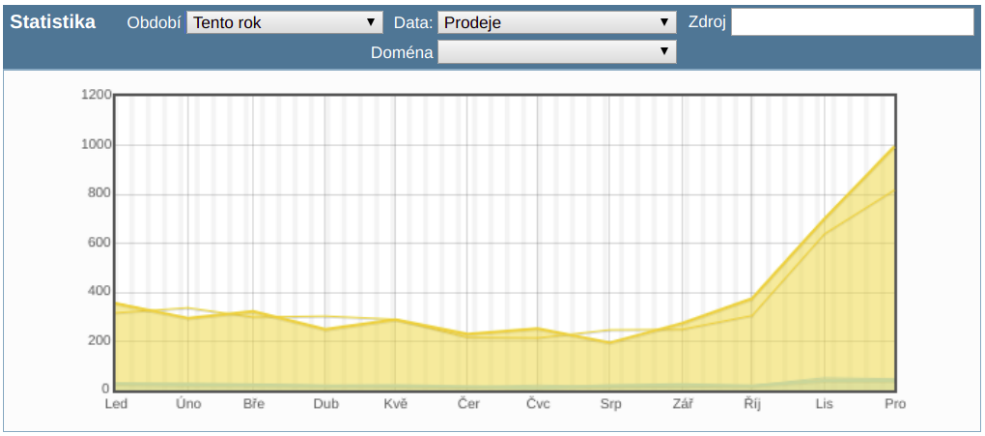
Při porovnání obrázků je vidět, že největší změnou customizované verze oproti původní OpenCart verzi prošly filtry.

Filtry

Období V původní administraci bylo možné zobrazit údaje pouze za poslední den (po hodinách), týden (po dnech), měsíc (po dnech) a rok (po měsících). Stejně tak tomu je v nejnovější verzi OpenCartu 3.0.2.0 (obr. 2.4). Oproti tomu customizovaná verze umožňuje zobrazit data za aktuální den, týden, měsíc, rok, rok (po týdnech), rok (dny s týdny), loňský rok, loňský rok (po týdnech), loňský rok (dny s týdny), desetiletí, denní dobu (za poslední měsíc) a denní dobu (za poslední rok).



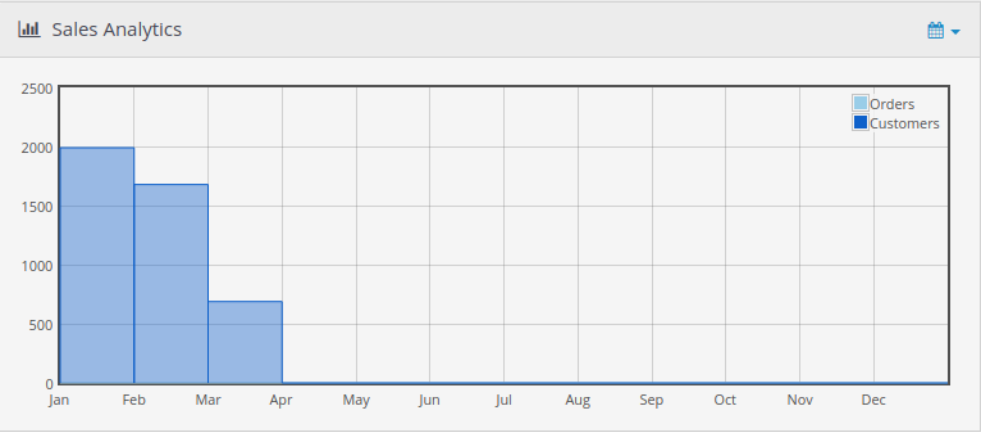
Obrázek 2.2: Přehledový graf prodejů za rok v OpenCart verzi 1.1.1



Obrázek 2.3: Přehledový graf prodejů za rok v customizované administraci

Data Aktuální verze customizované administrace umožňuje kromě nastavení filtrů také určit, jaká data se budou zobrazovat. Uživatelé si tedy mohou vybrat například mezi prodeji, ziskem, obratem či informacemi o balíčcích odeslaných jednotlivými logistickými společnostmi.

Zdroj Dále lze také v customizované administraci zobrazit data pouze z určitého zdroje. Pomocí fulltextového filtru je lze zúžit například pouze na Heureku, přičemž zadání *heureka* do filtru zobrazí data jak z obchodu heureka.cz, tak heureka.sk. Velkým problém je zde ale fakt, že zdrojů je velmi mnoho a filtr nenabízí žádné našeptávání (nápovědu). Uživatelé tak musí vědět, jaké všechny zdroje jsou k dispozici, aby je mohli vyhledat. Jinak musí vyhledávat metodou pokus-omyl.



Obrázek 2.4: Přehledový graf prodejů za rok v OpenCart verzi 3.0.2.0

Doména Posledním filtrem, který byl přidaný do nové administrace, je *Doména*. Jak už název napovídá, lze díky němu zobrazovat graf pro různé e-shopy, které administrace spravuje.

Toto je první krok k funkcionalitě, kterou původní OpenCart administrace nedisponuje – možnost mít jednu administraci pro více e-shopů, která by přes jedno rozhraní a samozřejmě jedno přihlášení umožňovala spravovat více projektů najednou.

Vylepšení

Jak je poznat z porovnání původní a customizované opencartové verze přehledového grafu, tak jedním z největších problémů je jeho zastaralost. Zdá se totiž, že na něm za posledních deset let neprobíhal žádný významný vývoj. Proto by do jakékoliv další verze customizované administrace bylo záhodno vybrat nějakou knihovnu na vykreslování grafů, která není ani moc nová (a tedy neobsahuje chyby) a zároveň není zastaralá (tedy vývoj na ní stále probíhá a nestane se, že by do roka přestala být podporována).

Také by bylo potřeba zvýšit uživatelskou přístupnost filtrů, například nějak lépe zobrazovat aktuálně filtrované hodnoty (někdy se při změně filtrů graf příliš neliší od předchozího, a tak není poznat, jestli se data změnila) či používat nějaké našeptávání při vyhledávání ve zdrojích.

Další potřebnou funkcí je možnost pamatovat si hodnoty filtrů grafu (teď se nastavení vymažou při opuštění stránky), nejlépe tak, aby primární nastavení bylo svázáno s uživatelovou rolí.

Možnost filtrovat si značky produktů na různých e-shopech by dle Jiřího Hunky byla také využívána.

Rozdělení požadavků do sekcí je tedy následující:

- **PO8** Moderní grafová knihovna [**MUST**]

- **PO9** Zobrazení informací podle role [**MUST**]
- **PO10** Indikace aktuálního filtru [**MUST**]
- **PO11** Pamatování si hodnot filtrů [**MUST**]
- **PO12** Filtry primárně dle rolí [**NICE**]
- **PO13** Filtry na produkty na různých doménách [**NICE**]

2.2.3 Objednávky

Objednávky jsou jednou z nejdůležitějších sekcí, proto je kritická jejich přehlednost a jednoduchost práce s nimi.

- **PO14** Indikace důvodu nedokončení objednávky [**MUST**]
- **PO15** Porovnávání nedokončené objednávky s následujícími dokončenými kvůli shodě [**MUST**]
- **PO16** Konzistence zobrazení dat mezi dokončenou a nedokončenou objednávkou [**MUST**]
- **PO17** Konfigurace sloupců v tabulce [**MUST**]
- **PO18** Vyhledávání objednávek s prázdným číslem faktury [**MUST**]
- **PO19** Možnost filtrovat jen podle hodnot, které dají nějaký výsledek [**MUST**]
- **PO20** Správnost dat (datum vytvoření objednávky je datum potvrzení objednávky, ne vytvoření košíku) [**MUST**]
- **PO21** Pohledy na tabulky podle rolí [**NICE**]

2.2.3.1 Detail objednávky

Detail objednávky by měl informace zobrazovat přehledně a kompaktně.

- **PO22** Přehlednější zobrazení údajů [**MUST**]
- **PO23** Jednodušší vystavování faktur [**MUST**]
- **PO24** Jednodušší vytváření kopie objednávky [**MUST**]
- **PO25** Jednodušší editace objednávky [**MUST**]
- **PO26** Možnost změnit dopravu [**MUST**]
- **PO27** Možnost upravit položky objednávky [**MUST**]
- **PO28** Potvrzení akcí (odeslání mailů, nastavení *nezaplaceno* atd.) [**MUST**]

2.2.4 Zákazník

Práce s informacemi o zákazníkovi není tak důležtá jako jiné sekce, ale i přes to se jí může dostat nějakých změn.

- **PO29** Věrnostní program [**NICE**]
- **PO30** Recenze produktů [**NICE**]
- **PO31** Integrovaný poštovní klient [**DEL**]

2.2.5 Zásilky

Pro jednoduchou práci s objednávkami je důležité i posílání zásilek, aby zákazník i správce obchodu neměli s balíky moc práce.

- **PO32** Vytváření zásilky bez objednávky [**MUST**]
- **PO33** Možnost přidat další dopravce [**MUST**]
- **PO34** Možnost měnit čas expedice a délku doručení pro různé dopravce [**MUST**]

2.2.6 Katalog

Přes katalog se upravují produkty a jejich kategorie, je to tedy důležitá sekce celé administrace.

- **PO35** Možnost hromadného nahrávání fotek k produktu [**MUST**]
- **PO36** Možnost popisu fotky produktu [**MUST**]
- **PO37** Jednodušší nastavování úvodní fotky produktu [**MUST**]
- **PO38** Barvu produktu zobrazovat nejen jako její hexadecimální hodnotu, ale i její grafickou reprezentací [**MUST**]
- **PO39** Jednodušší vytváření kopie produktu [**MUST**]
- **PO40** Řazení pomocí drag-and-drop[3] místo číslování [**NICE**]
- **PO41** Notifikace, které produkty se mají dokoupit, v závislosti na prodejích za poslední dobu [**NICE**]
- **PO42** Jednodušší nastavování hlavní kategorie [**NICE**]

[3]Drag-and-drop (táhni a pusť) je jeden ze způsobů interakce s uživatelským rozhraním. Jde o techniku, kde uživatel zmáčknutím levého tlačítka myši označí nějaký objekt (například fotografii), přetáhne jej na jiné místo na obrazovce a tam jej „pustí“ (uvolní tlačítko myši). Takto může jednoduše přesouvat třeba soubory v souborovém systému.

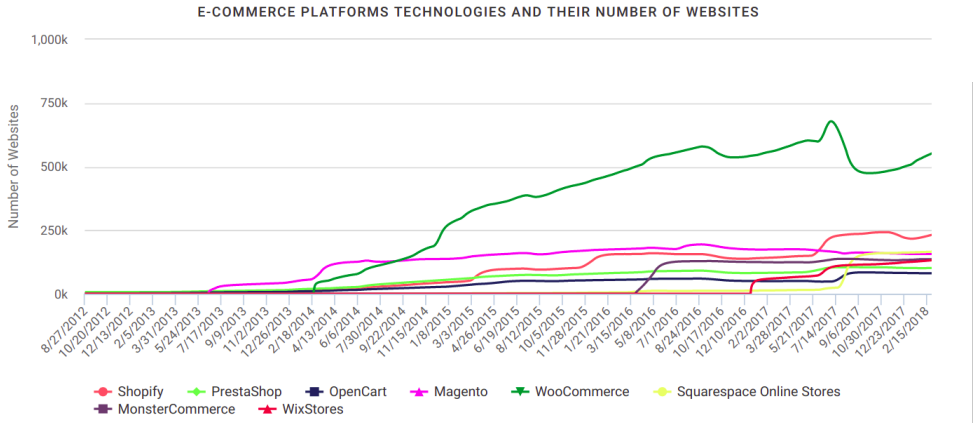
- **PO43** Akce [**DEL**]
- **PO44** Nehmotné produkty [**DEL**]
- **PO45** Obaly [**DEL**]
- **PO46** Stahování [**DEL**]

Kapitola 3

Průzkum trhu

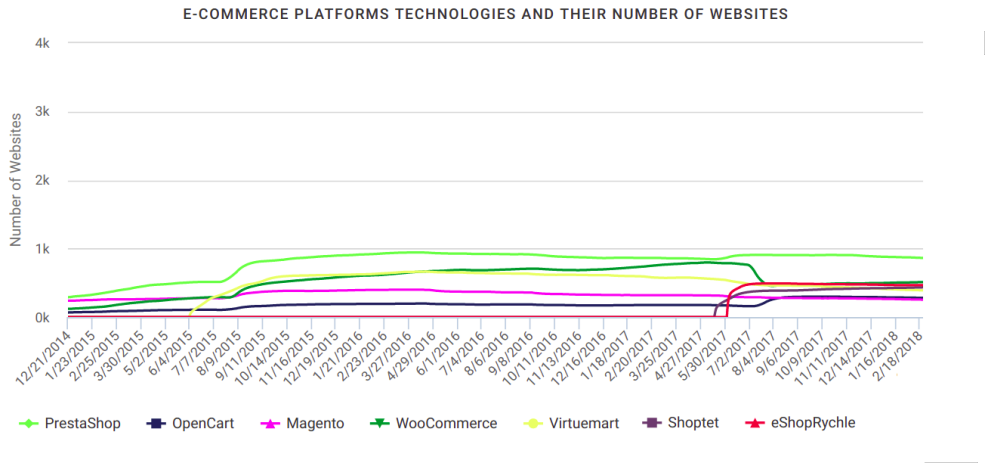
Na internetu je mnoho open-source e-shopů, které bylo užitečné si prohlédnout a zjistit, zda by se některý z nich nedal použít, či se jimi inspirovat při případném vytváření vlastní platformy. Následuje porovnání jednotlivých zkoumaných administrací.

Porovnával jsem nejpoužívanější e-shopy ze světového (obr. 3.1) a českého (obr. 3.2) trhu. Z nich jsem vybral tři komerčně nejpoužívanější (WooCommerce, Shopify a Squarespace), dále pak OpenCart (kvůli vztahu k původní administraci) a také platformy PrestaShop a Magento. Poslední dvě vybrané platformy jsou sice na světovém trhu v používanosti o pár procent pod WixStores, nicméně jsou úspěšnější na lokálním trhu a mají mnohem delší historii (WixStores vzniklo na konci roku 2016, PrestaShop i Magento jsou na trhu od začátku roku 2013).



Obrázek 3.1: Vývoj podílu nejpoužívanějších e-shopů na světovém trhu [3]

Zaměřoval jsem se převážně na použité technologie, podobnost workflow daného e-shopu a aktuální customizované administrace a do jaké míry e-shop splňuje body ze sekce Funkce aplikace 2.2. Procento pokrytí požadavků jsem



Obrázek 3.2: Vývoj podílu nejpoužívanějších e-shopů na českém trhu [4]

zjišťoval tak, že jsem procházel dema jednotlivých platforem a analyzoval jejich funkce. Pokud demo nebylo dostupné, hledal jsem informace v dokumentaci, případně na foru. Celkový rozpis pokrytí požadavků je pak možné najít v tabulce B.1.

Dalším důležitým bodem bylo, jestli stránka podporuje multisite – někdy překládáno jako síť webů, u e-shopů také označované jako multishop či multistore – tedy možnost přes jednu administraci spravovat více webů v rámci jednoho rozhraní, čistě pomocí přepínání mezi weby.

V potaz jsem bral také možnost jednoduché migrace z OpenCartu na cílový e-shop. Pokud by byla migrace a následná úprava administrace složitější, než napsat web úplně od základů dle potřeb zákazíka, bylo by rozhodnuto ve prospěch druhé možnosti.

Údaje o e-shopech jsou platné k únoru 2018.

3.1 OpenCart

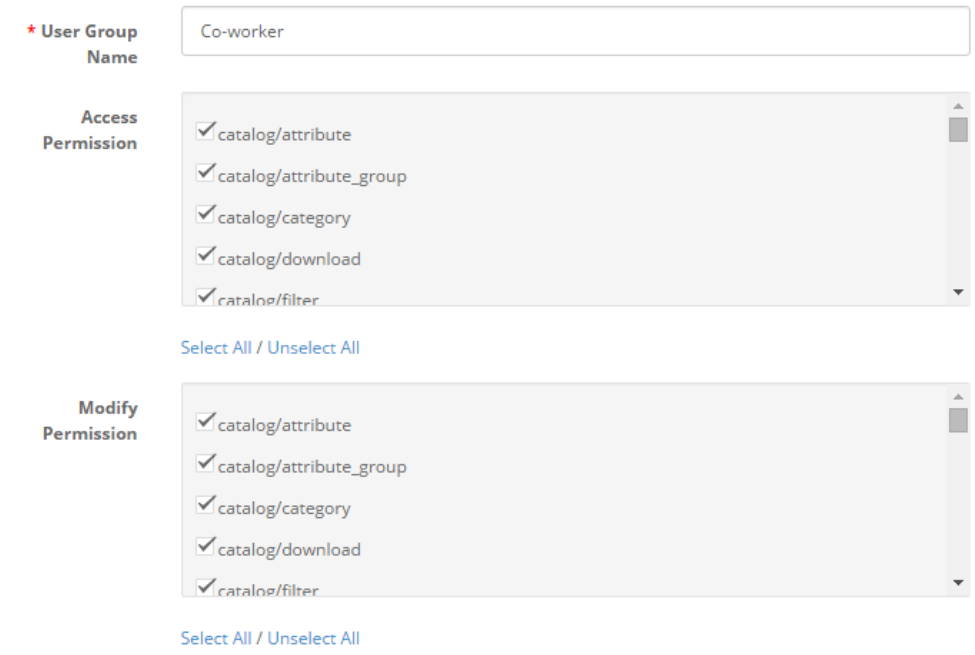
- **Nejnovější verze:** 3.0.2.0 (vydána 18. července 2017)
- **Technologie:** PHP 5.4+
- **REST API:** Ne
- **Multisite:** Ano
- **Rozšířitelnost:** OpenCart Extension Store
- **Web:** https://www.opencart.com/
- **Open-source:** Ano

- **Možná migrace z OpenCart:** Ano
- **Demo:** https://demo.opencart.com/
- **Pokrytí požadavků:** 20 %

Jak již bylo řečeno, tak OpenCart 3.0.2.0 i přes svůj moderní design zaostává prakticky ve všem ostatním.

V přehledovém grafu se dá stále filtrovat jen podle časového období, a to ještě velmi omezeně (obr. 2.4).

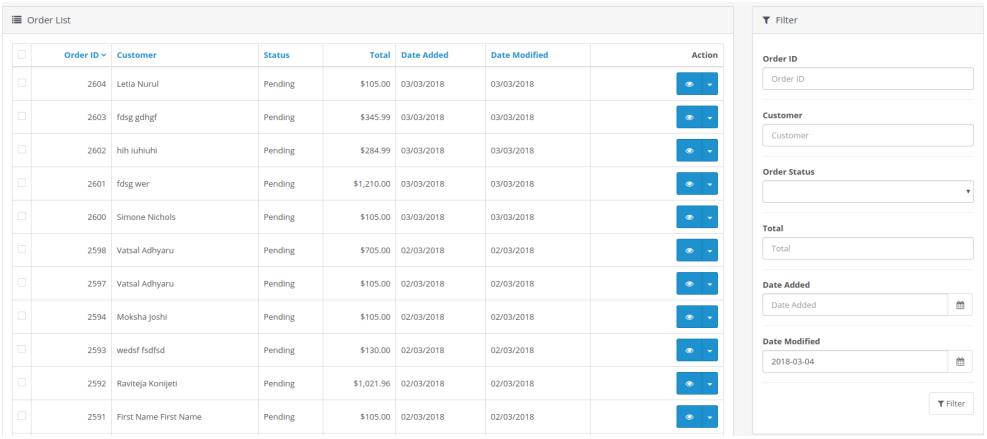
Nastavování rolí je stále komplikované, například pro nastavení práv na zobrazení a modifikování objednávky musí administrátor projít dva seznamy a v obou z nich ručně najít danou hodnotu (obr. 3.3), protože vyhledávání zde dostupné není. Kromě toho ani nelze nastavit uživateli více rolí zároveň.



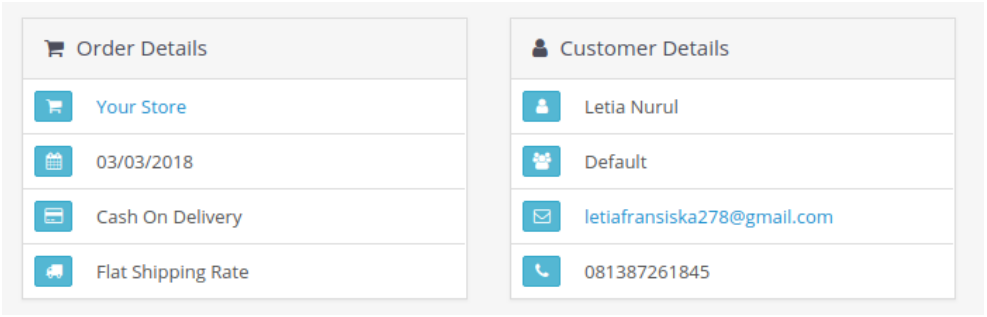
Obrázek 3.3: Nastavování práv v administraci OpenCart 3.0.2.0

Ani u objednávek nenastal žádný posun k lepšímu. Filtry v tabulce byly přesunuty netypicky na pravou stranu, čímž ubírají na šířce tabulce, která tak nemůže zobrazovat tolik dat. Neintuitivní skrytí tlačítek *Upravit* a *Smazat* za tlačítko *Detail* (ikonka oka) pouze přidává na počtu kroků, které musí uživatel udělat, aby dosáhl kýžené akce (obr. 3.4).

Detail objednávky dosáhl určitých změn, důležité informace jsou zobrazeny kompaktně a přehledně (obr. 3.5), nicméně ostatní informace jsou stále zbytečně rozprostřené po celé stránce. Stejně tak editace je nesmyslně rozdě-



Obrázek 3.4: Filtry objednávek v administraci OpenCart 3.0.2.0



Obrázek 3.5: Informace o objednávce v administraci OpenCart 3.0.2.0

lena na pět různých stránek, místo toho, aby bylo ušetřeno na mezerách mezi jednotlivými poli a související položky byly sdruženy.

Jediná výrazná změna v sekci *Produkty* byla oddělení SEO[4] informací do vlastní sekce.

Nahrávání fotek k produktům se nedostalo žádných změn a uživatelé je musí stále nahrávat po jedné. Stejně tak z nich nemohou jednoduše vybrat úvodní fotku produktu – tato akce se dělá pomocí opětovného procházení souborového systému (ať toho na serveru či na uživatelově počítači).

[4]Optimalizace pro vyhledávače (metatagy, klíčová slova atd.).

3.2 WooCommerce

- **Nejnovější verze:** 3.2.0 (vydána 11. října 2017)
- **Technologie:** PHP 5.6+
- **REST API:** Ano
- **Multisite:** Ano
- **Rozšířitelnost:** WooCommerce Extension Store
- **Web:** https://woocommerce.com/
- **Open-source:** Ano
- **Možná migrace z OpenCart:** Ano
- **Demo:** Ne
- **Pokrytí požadavků:** 52 %

I když je WooCommerce nejpoužívanější software pro e-shopy (25% podíl na trhu [3]), tak je dle mého názoru velmi nevhodný pro kohokoliv, kdo si chce kód byť jen lehce upravit.

WooCommerce sice nabízí velké množství rozšiřujících modulů, ale kvůli tomu, že celá platforma je pouze plugin do CMS[5] WordPress, jsou možnosti úprav kódu velmi omezené.

Každá aktualizace jakéhokoliv modulu totiž přepíše ruční změny v daných souborech. Člověk si tedy musí vybrat, jestli chce mít moduly a WordPress aktuální (nové aktualizace jsou často dostupné každý týden), jestli chce mít možnost si software upravit dle svých potřeb, či jestli mu stojí za to při každé aktualizaci jakéhokoliv modulu dělat ruční slučování obou kódů.

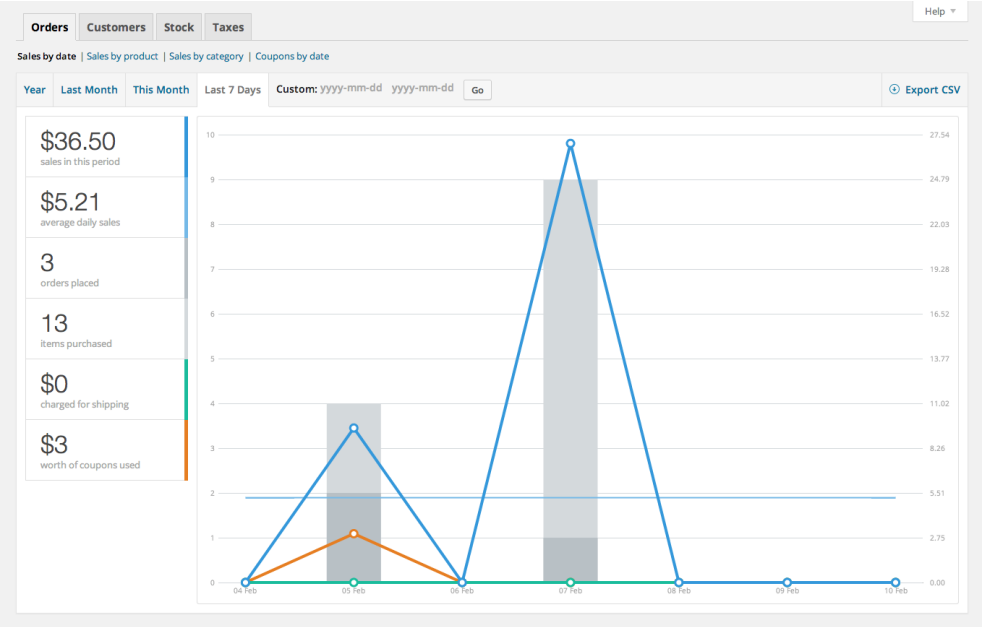
Sice by pro velkou většinu věcí, které firma Jagu s.r.o. potřebuje, bylo možné najít nějaký plugin, ale přidávání velkého množství rozšiřujících modulů má velký dopad na rychlost e-shopu a může ohrozit aplikaci co se bezpečnostních děr v kódu třetích stran týče. Zároveň je časté, že pluginy mezi sebou nejsou kompatibilní a jejich souběžná instalace buď není možná, nebo udělá celý e-shop neprovozuschopný.

Bohužel je WordPress kvůli své obecnosti[6] příliš nepřehledný a obsahuje spoustu funkcí či kódu, které jsou ne vždy využity. I kvůli tomu je velmi obtížné v něm dělat nějaké velké změny.

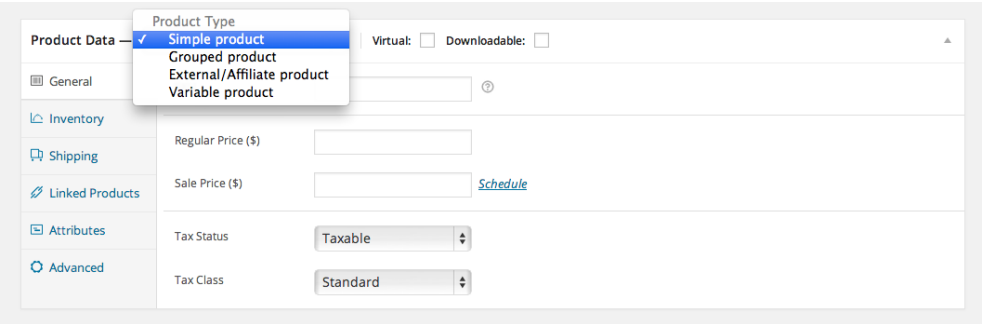
[5](CMS z anglického content management system) je software zajišťující správu dokumentů, nejčastěji webového obsahu. V dnešní době se jako CMS zpravidla chápou webové aplikace, někdy s případným doplňkovým programovým vybavením u klienta [17].

[6]Wordpress je využíván nejen na e-shopy, ale i na blogy, firemní stránky, online magazíny a spoustu dalších, v součtu má neskutečný 60.2% podíl na trhu [18].

Na druhou stranu má WooCommerce pěkně udělané přehledové grafy, jak je vidět na obrázku 3.6. Stejně tak se mi líbí kompaktnost a jednoduchost editace produktů (obr. 3.7), která je sice členěna na sekce, jako je to u nové verze OpenCartu, ale rozdělení zde působí přirozeněji a hlavně nezabírá na stránce tolik místa.



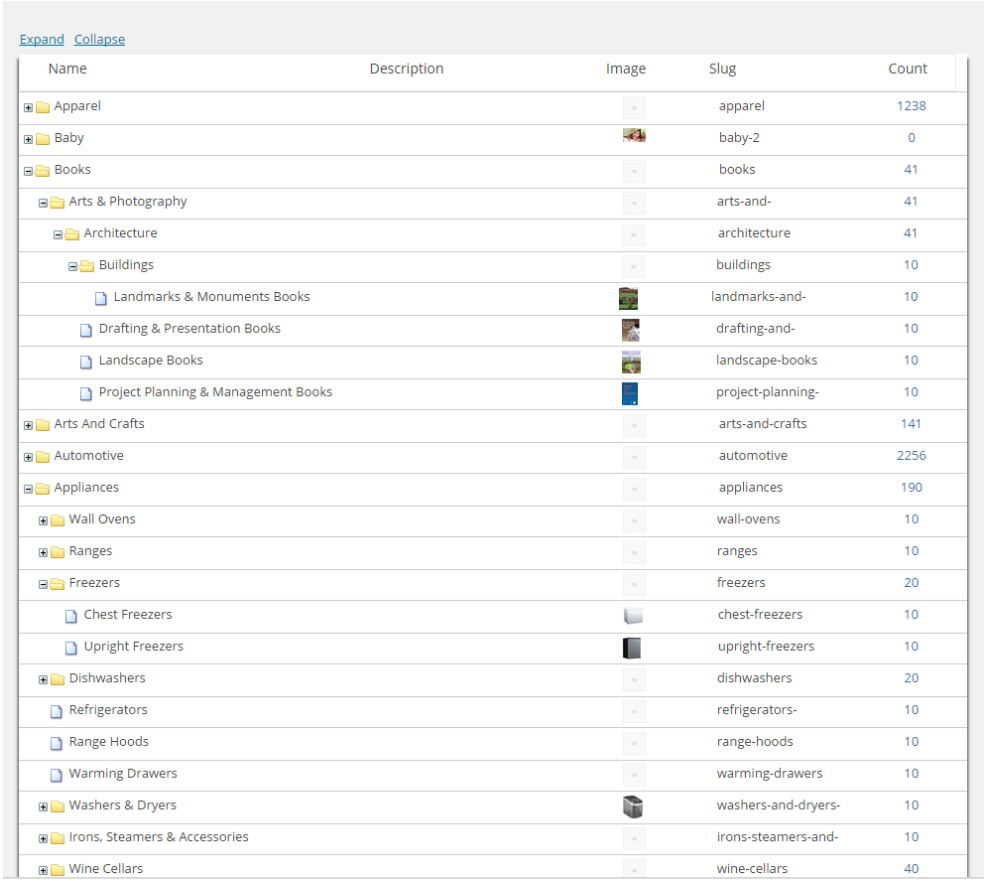
Obrázek 3.6: Přehledový graf v administraci WooCommerce [5]



Obrázek 3.7: Editace produktu v administraci WooCommerce [5]

Také má WooCommerce velmi dobře řešená práva uživatelů, která jsou zde nazývána jako Schopnosti. Ty může administrátor přidávat dle potřeby, například na zobrazení a úpravu jednotlivých stránek/produktů. Schopnosti mohou být přiřazeny skupinám a uživatelé pak patří do žádné, jedné nebo více skupin.

Jedním z často využívaných rozšíření pro WooCommerce je modul Category tree, který umožňuje klasické textové zobrazení kategorií zaměnit za stromovou vizualizaci adresářů, stylizovanou jako souborový systém. Ta sice není nijak výjimečně pěkně graficky realizovaná (obr. 3.8), ale svůj účel plní a je rozhodně jednodušší a přehlednější na práci než zobrazení kategorií u jiných administrací. Zároveň umožňuje drag-and-drop manipulaci s jednotlivými kategoriemi. Velkou nevýhodou zde ale je nemožnost v kategoriích vyhledávat.



Obrázek 3.8: Strom kategorií v administraci WooCommerce [6]

Fotografie k produktům se zde nahrávají buď pomocí výběru z prohlížeče souborového systému nebo drag-and-dropem, obojí podporuje nahrávání více souborů najednou. Nicméně zde opět není možnost vybrat z nahraných fotek úvodní fotku přímo, uživatel musí otevřít jiné menu.

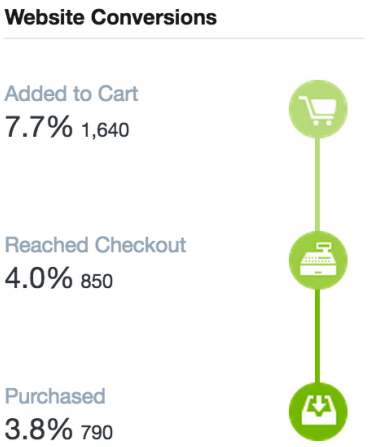
I když má WooCommerce spoustu silných stránek, tak její neflexibilnost co se ručních úprav týče ji činí pro složitější projekty nepoužitelnou.

3.3 Shopify

- **Nejnovější verze:** Neverzuje se
- **Technologie:** Ruby on Rails
- **REST API:** Ano
- **Multisite:** Ne
- **Rozšiřitelnost:** Shopify App Store
- **Web:** https://www.shopify.com/
- **Open-source:** Ne
- **Možná migrace z OpenCart:** Ano
- **Demo:** Ne
- **Pokrytí požadavků:** 43 %

Druhým nejpoužívanějším e-commerce systémem je Shopify (11.21% podíl na trhu [3]). Ten ale není pro potřeby naší administrace využitelný, protože se jedná o placenou službu, tedy není open-source. Cena za měsíční předplatné je 29 $ (cca 600 Kč) za Basic verzi, 79 $ (cca 1 630 Kč) za klasickou verzi a 299 $ (cca 6 150 Kč) za verzi Advanced[7] [19].

I tak je možné nějaké myšlenky z jejich administrace převzít. Konkrétně se mi líbilo pěkné zpracování uživatelských konverzí[8] (obr. 3.9), které přehledně zobrazuje uživatelovo chování.



Obrázek 3.9: Graf konverzí v administraci Shopify

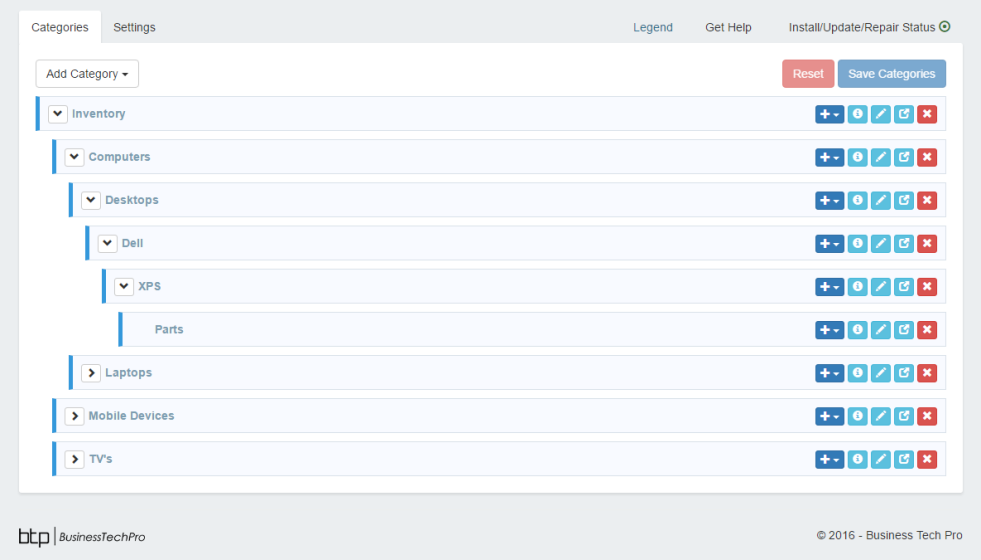
Dále se mi, stejně jako u WooCommerce (sekce 3.2), líbilo stromové uspořádání kategorií (obr. 3.10), které je zde lépe graficky zpracované než u zmíněného WooCommerce, i když opět pouze ve formě pluginu.

I když Shopify v samotném nahrávání fotek k produktům nijak nevyniká (jedná se o klasický drag-and-drop/procházení souborového systému), přece

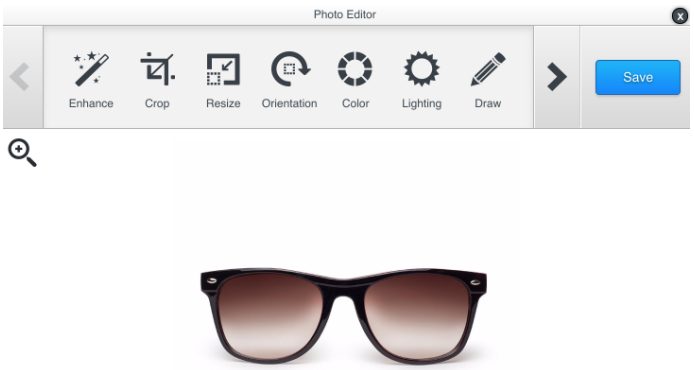
[7] Dle kurzu 1 americký dolar = 20,5888408 českých korun ze dne 4. března 2018.

[8] Konverze na internetu označuje situaci, kdy návštěvník webových stránek vykoná provozovatelem žádanou akci, jež má pro něj zejména obchodní užitek. Konverze je tak jedním z nástrojů internetového marketingu. Statistická pravděpodobnost, že se z návštěvníka stránky stane zákazník, se označuje jako konverzní poměr (angl. conversion rate), příp. míra konverze[20].

jen mají v tomto procesu jednu zajímavost. Po nahrání fotky systém umožňuje ji upravit, a to všemi možnými způsoby od oříznutí a překlápění až po úpravu jasu a dokreslování (obr. 3.11).



Obrázek 3.10: Strom kategorií v administraci Shopify [7]



Obrázek 3.11: Úprava obrázku v administraci Shopify [8]

3.4 Squarespace

- **Nejnovější verze:** 7 (vydána 7. října 2014)
- **Technologie:** Java
- **REST API:** Ano
- **Multisite:** Ano
- **Rozšířitelnost:** Squarespace integrations
- **Web:** https://www.squarespace.com/
- **Open-source:** Ne
- **Možná migrace z OpenCart:** Ne
- **Demo:** Ne
- **Pokrytí požadavků:** 30 %

Třetí nejpoužívanější e-commerce systém je Squarespace (7.21% podíl na trhu [3]), který je ale stejně jako Shopify placený, tudíž pro naši administraci nevhodný. Cena za Business verzi je 18 $ (cca 370 Kč)[9] [21].

Squarespace má pěkně zpracované grafy (obr. 3.12), které jsou rozdělené do sekcí:

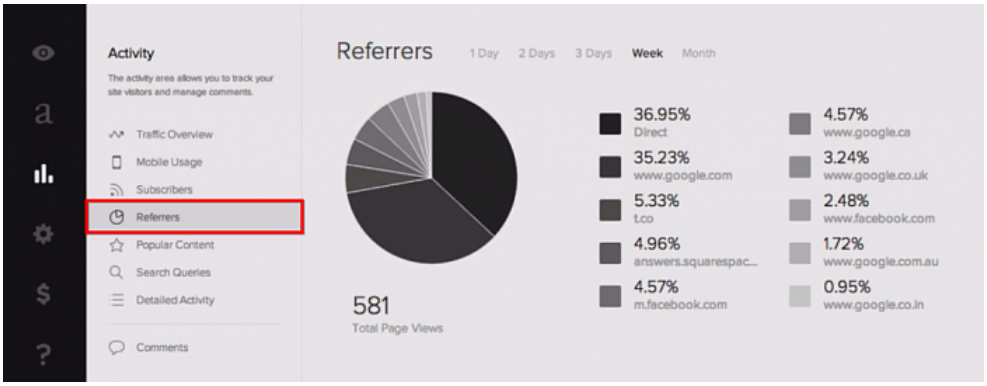
- **Traffic** – Návštěvnost,
- **Mobile usage** – Přístupy z mobilních zařízení,
- **Subscribers** – Odběratelé,
- **Referrers** – Odkazovatelé, tedy odkud návštěvníci přišli (napřímo, přes vyhledávání, přes odkaz atd.),
- **Popular content** – Populární obsah,
- **Search queries** – Vyhledávání.

Bohužel jediný filtr, který je zde implementovaný, je na časové období, a ještě k tomu obsahuje pouze omezené množství hodnot (v závislosti na typu grafu), nejčastěji jen den, týden a měsíc.

Uživatelské role jsou zde řešeny velmi jednoduše – existuje osm rolí:

- **Site Owner** – Majitel stránek,
- **Administrator** – Administrátor,

[9] Viz poznámku číslo 7



Obrázek 3.12: Graf Odkazovatelů v administraci Squarespace [9]

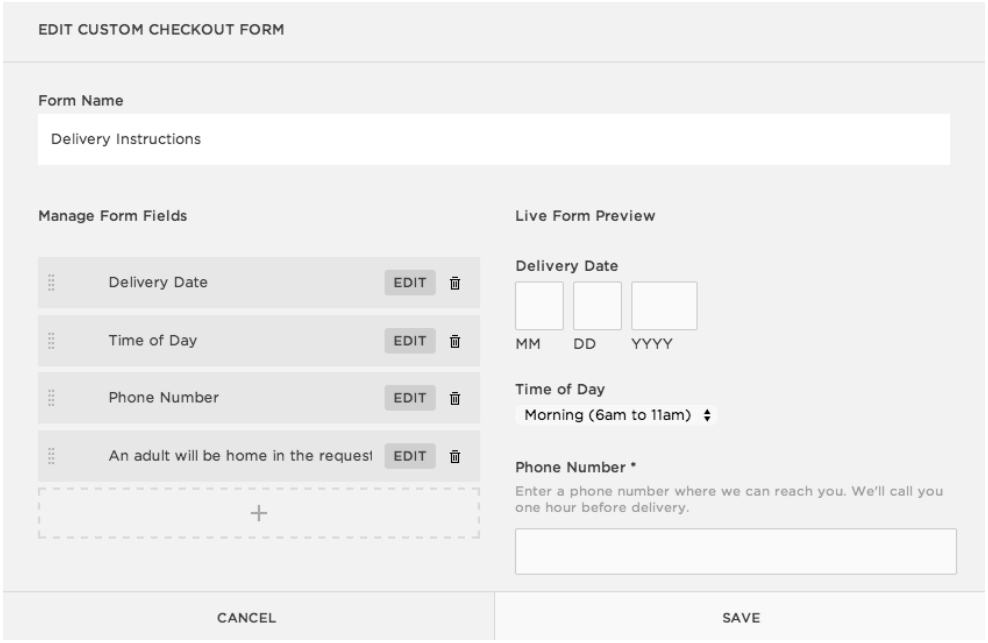
- **Content Editor** – Správce obsahu,
- **Billing** – Účetní,
- **Store Manager** – Správce obchodu,
- **Reporting** – Nahlašování,
- **Trusted Commenters** – Důvěryhodní komentující,
- **Comment Moderators** – Moderátoři komentářů.

Těmto rolím jsou jasně nastavena práva na jednotlivé stránky/akce a ty nelze nijak měnit, stejně tak nelze nové role přidávat. Tři poslední role se týkají pouze komentářů na frontendu a mají práva velmi limitována. Uživatelé však mohou mít přiřazeno více rolí.

Squarespace nepodporuje editaci či konkrétnější informace o objednávce (jako je třeba historie). Na druhou stranu mě velmi zaujala možnost si objednávkový formulář jednoduše upravit přidáváním či odebíráním konkrétních polí 3.13. Zákazník si pak například může nastavit, kdy je pro něj ideální čas dodání, jestli je objednávka myšlena jako dárek, bude-li v době převzetí doma někdo plnoletý atp.

Přiřazování kategorií produktům je velmi jednoduché, jedná se vlastně o štítky bez jakékoliv hierarchie.

Fotografie produktu jsou nahrávány pomocí klasického drag-and-dropu, jako hlavní fotografie je nastavena ta první z nich (jednoduchým přesunutím se dá nastavit jiná).



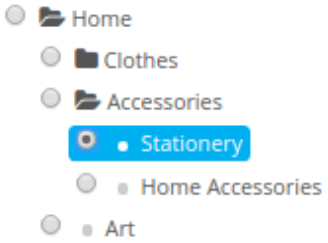
Obrázek 3.13: Nastavování polí objednávkového formuláře v administraci Squarespace [10]

3.5 PrestaShop

- **Nejnovější verze:** 1.7.3.0 (vydána 28. února 2018)
- **Technologie:** PHP 5.4+
- **REST API:** Ano
- **Multisite:** Ano
- **Rozšířitelnost:** PrestaShop Addons Marketplace
- **Web:** https://www.prestashop.com/
- **Open-source:** Ano
- **Možná migrace z OpenCart:** Ano
- **Demo:** http://demo.prestashop.com/en/
- **Pokrytí požadavků:** 65 %

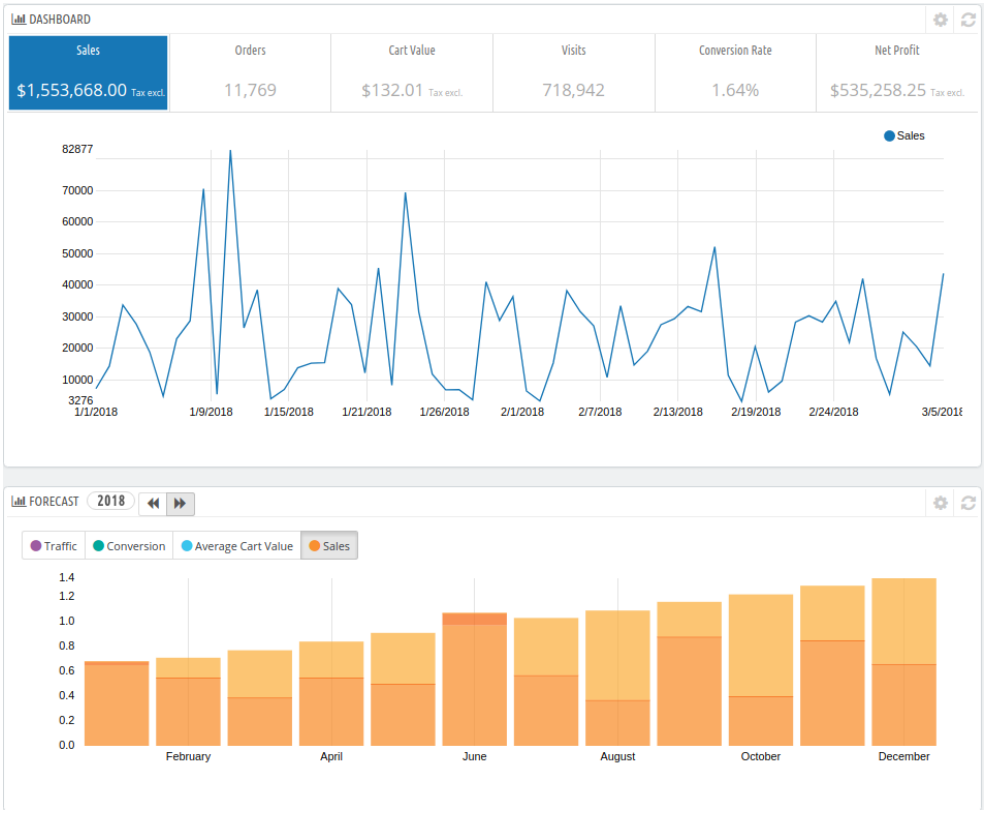
I když PrestaShop není nijak často používaným systémem (má podíl na trhu pouhých 4.05 % [3]), nabízí velmi mnoho funkcí v moderním designu.

Příkladem jsou opravdu pěkně řešené grafy, kde se kromě časového období dá filtrovat podle různých hodnot, jako jsou prodeje či uživatelské konverze. Také nabízí kromě zobrazení dat z historie i předpověď budoucích návštěv či prodejů (obr. 3.15).



Obrázek 3.14: Nastavování rodičovské kategorie v administraci PrestaShop

Nastavování kategorií je stylizováno jako souborový systém, stejně jako u konkurenčního WooCommerce (obr. 3.8), grafika zde ale vypadá mnohem lépe, neboť se odprostila od klasického Windows [22] designu složek a souborů a poskytuje přehledný způsob, jak do kategorií produkty přiřazovat (obr. 3.14).

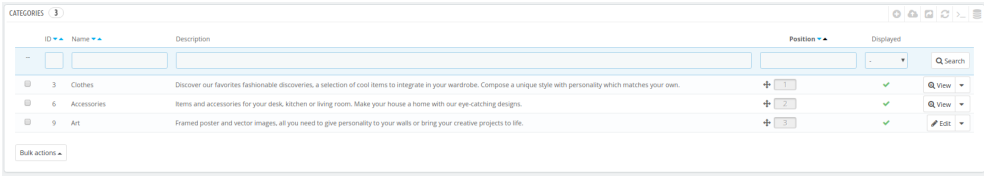


Obrázek 3.15: Přehledové grafy v administraci PrestaShop

Bohužel celkový přehled kategorií je řešený trochu nešťastně (obr. 3.16), protože se jedná o tabulku. Ta sice nabízí vyhledávání, což byla velká nevýhoda konkurence, ale vyhledávání funguje pouze na jedné úrovni kategorií a nezanořuje se hlouběji do stromu. Ani není možné jednoduše rozbalit danou větev kategorie – uživatel sice může kliknout na daný řádek v tabulce, ale to

má za následek překreslení celé stránky a zobrazení opět pouze jedné úrovně kategorií. Pro návrat o úroveň výše je potřeba kliknout na tlačítko *Zpět*, což opět překreslí celou stránku.

Další nevýhodou je zobrazování celého popisu kategorie přímo v tabulce, což má za následek její velikou šířku a tudíž ne úplně dobrou přehlednost.



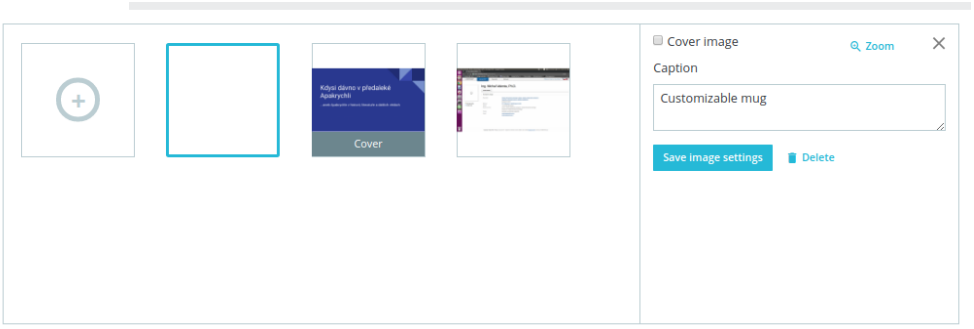
Obrázek 3.16: Tabulka kategorií v administraci PrestaShop

Uživatelské role jsou zde řešeny klasicky pomocí přiřazování práv na jednotlivé akce k rolím, nicméně uživatel nemůže mít více rolí najednou. Na rozdíl od OpenCartu (obr. 3.3) je zde přiřazování práv mnohem jednodušší. Je totiž řešeno pomocí tabulky, kde v řádcích jsou jednotlivé stránky a ve sloupcích akce (obr. 3.17). Opět zde ale není možno vyhledávat, pokud tedy nebereme v potaz implicitní vyhledávání pomocí webového prohlížeče.

	View	Add	Edit	Delete	All
»**Dashboard**	☐	☐	☐	☐	☐
»**Sell**	☑	☑	☑	☑	☑
» Orders	☐	☐	☐	☐	☐
» Orders	☐	☐	☐	☐	☐
» Invoices	☐	☐	☐	☐	☐
» Credit Slips	☐	☐	☐	☐	☐
» Delivery Slips	☐	☐	☐	☐	☐
» Shopping Carts	☐	☐	☐	☐	☐
» Catalog	☑	☑	☑	☑	☑
» Products	☑	☑	☑	☑	☑
» Categories	☑	☑	☑	☑	☑
» Monitoring	☐	☐	☐	☐	☐
» Attributes & Features	☐	☐	☐	☐	☐
» Attributes	☐	☐	☐	☐	☐
» Features	☐	☐	☐	☐	☐
» Brands & Suppliers	☐	☐	☐	☐	☐

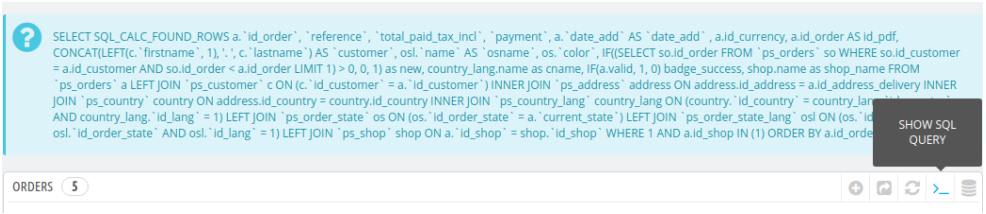
Obrázek 3.17: Tabulka uživatelských práv v administraci PrestaShop

Velmi intuitivní je ovšem nahrávání obrázků k produktům, které je možné buď pomocí výběru ze souborového systému či drag-and-dropem přímo do okna s editací. Obojí samozřejmě povoluje nahrávání více souborů najednou. Úvodní fotka produktu se pak vybírá jednoduše pomocí zaškrtávátka u vybrané fotky (obr. 3.18). Na stejném místě může uživatel i změnit popis jednotlivých nahraných fotek.



Obrázek 3.18: Nahrávání fotek produktů v administraci PrestaShop

Velmi zajímavá (i když pro obyčejného uživatele asi nijak výjimečně důležitá) mi přijde možnost si při práci s tabulkami nechat vypsat SQL dotaz, který se provedl při získávání dat pro danou tabulku (obr. 3.19). Zároveň si uživatel může dotaz upravit a uložit do SQL Manageru[10]. Díky tomu je pak jednoduché jej kdykoliv znovu provést.



Obrázek 3.19: SQL dotazy v administraci PrestaShop

3.6 Magento

- **Nejnovější verze:** 2.2 (vydána 26. září 2017)
- **Technologie:** PHP 5.6+
- **REST API:** Ano
- **Multisite:** Ano
- **Rozšiřitelnost:** Magento Marketplace
- **Web:** https://magento.com/
- **Open-source:** Ano

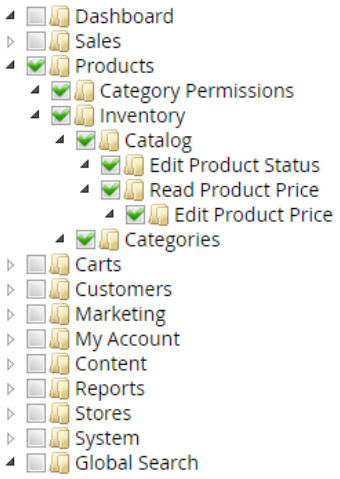
[10]SQL Manager je nástroj PrestaShopu, který umožňuje spravovat SQL dotazy, které uživatel chce častěji využívat. Tato funkce je ale z bezpečnostních důvodů omezena pouze na příkaz typu SELECT.

- **Možná migrace z OpenCart:** Ano
- **Demo:** http://magento2.demo.ubertheme.com
- **Pokrytí požadavků:** 56 %

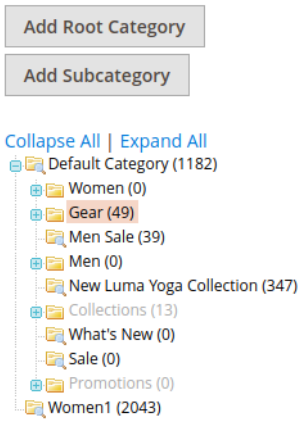
Magento, ač dle mého názoru není nijak významně zajímavě udělané či pěkné na pohled, přináší pár nápaditých způsobů, jak řešit některé problémy.

Například role uživatelů řeší pomocí stromu práv, jak je vidět na obrázku 3.20. Tento způsob se sice logikou velmi podobá tabulce práv v konkurenčním PrestaShopu (obr. 3.17), nicméně řešení pomocí adresářového stromu je bezpochyby přehlednější. Vyhledávání ve stromu opět není dostupné, ale troufám si tvrdit, že v tomto řešení je filtrace mnohem méně potřeba než u zmíněného PrestaShopu. Na druhou stranu, kdyby uživatel opravdu vyhledávat potřeboval, bude tak moci udělat mnohem snáze v HTML tabulce pomocí klávesové zkratky CTRL+F v prohlížeči než v tomto řešení. Zde by totiž musely být všechny větve rozbalené, aby takovéto vyhledávání fungovalo, a tím se pak ztrácí krása, kompaktnost a přehlednost stromu.

Podobně je v Magentu provedeno zobrazení kategorií, které má ale v porovnání s PrestaShopem mnohem horší design (obr. 3.21).



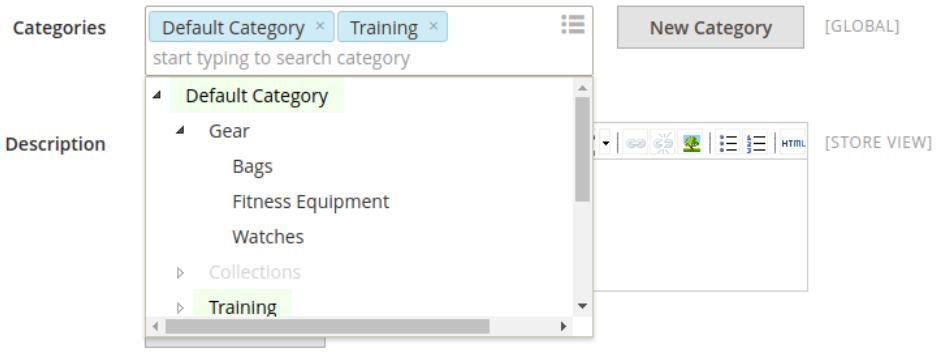
Obrázek 3.20: Nastavování práv uživatelů v administraci Magento



Obrázek 3.21: Strom kategorií v administraci Magento

Na druhou stranu se mi líbí přiřazování kategorií přímo k produktům, které je implementováno jako textové pole s našeptáváním, kde našeptávání je ve formě stromu (obr. 3.22).

Magento má také v pěkném provedení nahrávání a editaci fotek u produktů (obr. 3.23), kde uživatel může buď fotky zvolit pomocí vyskakovacího okna se souborovým systémem či přes drag-and-drop. Poté může nastavit, jestli je



Obrázek 3.22: Přiřazování kategorií k produktu v administraci Magento

fotka úvodní, zda-li se má zobrazovat miniatura a další. Velkou výhodu vidím i v jednoduché změně pořadí fotek, která je opět realizována pomocí drag-and-dropu.

Zároveň je možné všechny tyto úkony dělat nejenom s fotkami, ale i s videosoubory, dokonce je i nastavit jako úvodní video (místo úvodní fotografie).

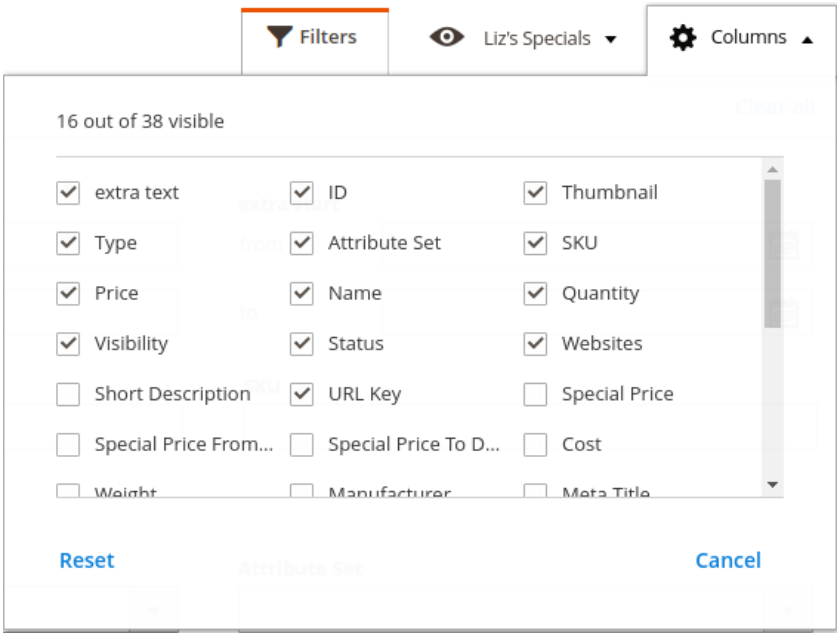
Jako chybu v uživatelském rozhraní ale vidím chybějící potvrzovací okno při mazání fotek. Dále bych jako uživatel ocenil možnost nastavit úvodní fotku, aniž bych musel zobrazit zvětšeninu dané fotky.

Obrázek 3.23: Nahrávání fotek produktů v administraci Magento

Jako jediný e-shop, na který jsem se ve své diplomové práci zaměřil, umožňuje pokročilou práci s tabulkami. Kromě dobře uděleného filtrování je zde totiž implementováno jak skrývání sloupců, tak možnost uživatelských pohledů, ovlivňujících nastavení filtrů a zobrazených sloupců (obr. 3.24).

3.7 Shrnutí

V této kapitole jsem rozebral šest různých administrací e-shopů, vybraných kvůli jejich oblíbenosti či známosti na českém a světovém trhu. Každá z nich



Obrázek 3.24: Nastavování zobrazení tabulek v administraci Magento

měla svoje klady i zápory, jak co se týče obecné použitelnosti, tak splnění požadavků na budoucí verzi customizované administrace (viz sekce 2.2).

Veškerý rozebíraný software má moderní grafiku, která je dnes již samozřejmostí, ale ne vždy je s ní pracováno tak, aby uživatelova interakce s rozhraním byla jednoduchá a intuitivní.

Funkcionalita se však často diametrálně liší, protože každá administrace přistupuje k různým problémům jinak, ale žádná z nich nesplňuje přesně představy ideálu.

Všechny zjištěné informace jsem sepsal do tabulky B.1, kterou je možné nalézt v příloze B, zde dávám pouze shrnutí dané tabulky (tabulka 3.1).

Pro přehlednost zde používám zkratku OS – open-source.

	Pokrytí PO	OS	Migrace	Multisite	API	Jazyk
OpenCart	20 %	Ano	Ano	Ano	Ne	PHP 5.4+
WooCommerce	52 %	Ano	Ano	Ano	Ano	PHP 5.6+
Shopify	43 %	Ne	Ano	Ne	Ano	Rails
Squarespace	30 %	Ne	Ne	Ano	Ano	Java
PrestaShop	65 %	Ano	Ano	Ano	Ano	PHP 5.4+
Magento	56 %	Ano	Ano	Ano	Ano	PHP 5.6+

Tabulka 3.1: Shrnutí jednotlivých administrací e-shopů

Před finálním verdiktem bych chtěl podotknout, že dostupných adminis-

trací je mnohem více a je možné, že nemálo z nich dosahuje stejných či vyšších kvalit jako některé zde rozebírané. Stejně tak i těch několik mnou vybraných poskytuje obrovské množství funkcí a mohlo se stát, že jsem některé z nich přehlédl, i když jsem se snažil o co nejdůkladnější průzkum. Aby člověk poznal takto komplexní systémy a věděl o každé jejich funkci, musel by s nimi pracovat po mnoho dnů a týdnů, což v rámci této práce není možné.

Nejlepší ze zmíněných administrací je dle analýzy PrestaShop, který řeší většinu problémů elegantně a logicky a zároveň pokrývá velkou část požadavků vypsaných v sekci 2.2.

Proto jsem se rozhodl zaměřit se detailněji na možnosti migrace z OpenCartu na PrestaShop, abych porovnal, jestli migrace ze staré verze OpenCartu, spolu s již hotovými úpravami databáze, není časově náročnější než napsat celý systém od nuly.

3.8 Možnosti migrace na PrestaShop

Možností migrace z OpenCartu na PrestaShop bylo několik, dle očekávání byly všechny zpoplatněny.

Největším problémem migrace byla velikost databází – customizovaná administrace čítala 104 tabulek (10–15 jich ovšem bylo prázdných) a databáze PrestaShopu jich měla 250, takže ať bych se rozhodnul pro jakoukoliv možnost, úspěšná migrace by nebyla jednoduchá.

Protože bylo oproti OpenCartu 1.1.1 v customizované verzi mnoho tabulek přidaných (dle dodaných informací jich bylo 25–30), musela by se i v případě využití nějakého nástroje třetí strany na automatické migrace velká část databáze přemigrovat ručně.

Počítal jsem, že pokud by ruční migrace jedné tabulky zabrala 20 minut, tak i v případě použití automatizovaného externího skriptu by bylo potřeba migrací přidaných tabulek strávit alespoň 10 hodin (když připočítám potřebu propojení s automaticky zmigrovanými tabulkami). Zároveň bylo nutné data zkontrolovat a náležitě otestovat, jestli migrace proběhla v pořádku, na to jsem vyhradil dalších 5 hodin. Dohromady by tedy tyto dodělávky zabraly 15 hodin.

Pak bylo samozřejmě potřeba doimplementovat a napojit na databázi funkce, které byly přidány do současné customizované administrace oproti původní OpenCart verzi.

Důležitým parametrem při vybírání migračního nástroje byla nutnost zmigrovat několik OpenCart e-shopů. Bylo tedy potřeba vybrat takový, u kterého se neplatí za každou migraci zvlášť.

Dále jsem musel vzít v potaz celkový objem migrovaných dat, neboť některé tabulky (převážně tabulky historie objednávek) měly desetitisíce záznamů. Proto u každé ze zvažovaných možností uvádím, kolik by stála migrace všech záznamů, popřípadě pouze dat od začátku roku 2017.

V tabulce 3.2 pro přehlednost uvádím data za daná období na všech 14 e-shopech spravovaných firmou Jagu s.r.o. Protože produkty a kategorie se musely přemigrovat všechny, je pro obě časové období stejný údaj. Údaje jsou platné k 3. dubnu 2018.

	2017–2018	Celkem
# objednávek	12 134	46 639
# záznamů historie objednávek	57 619	207 339
# aktivních zákazníků	785	3 981
# kategorií	308	308
# produktů	1 751	1 751

Tabulka 3.2: Data na e-shopech za dvě různá období

3.8.1 MigrationPro

- **Web:** https://addons.prestashop.com/en/data-migration-backup/27664-migrationpro-opencart-to-prestashop-migration-tool.html

MigrationPro je modul přímo do PrestaShopu, který dle popisu umožňuje za jednotnou cenu migraci všech dat (produkty, objednávky, zákazníky, pravidla na ceny v katalozích, zaměstnance, košíky, komunikaci se zákazníky, hesla zákazníků, SEO (metadata), CMS stránky, vlastní pole, recenze, atributy, kombinace, daně, obrázky a další).

Hodnocení i komentáře uživatelů měl pozitivní, za poslední čtvrtletí roku 2017 a první čtvrtletí roku 2018 byl hodnocen čtyřikrát, z toho třikrát pěti a jednou čtyřmi hvězdičkami (z pěti). Uživatelé nástroj povětšinou hodnotili jako jednoduchý na práci. Zároveň je podle PrestaShop Addons Marketplace vyvíjen ověřeným týmem vývojářů, měl by tedy poskytovat kvalitní služby.

U vývojářů jsem si zároveň ověřil, že je možné pomocí MigrationPro zmigrovat i customizovaný OpenCart založený na verzi 1.1.1. Dle očekávání však uměl nástroj pracovat jenom se základními OpenCart tabulkami, neuměl přemigrovat žádné nově přidané.

Další výhodou bylo neomezené použití balíčku, tedy jedna licence šla použít na více migrací, nezávisle na počtu obchodů. Muselo ovšem jít pouze o jeden cílový obchod, tedy bylo možné vytvořit multisite PrestaShop z několika OpenCart e-shopů za cenu jedné licence.

Služba stála 96,79 € (cca 2 450 Kč)[11], v ceně balíčku byla zahrnuta i tříměsíční podpora od tvůrců, samozřejmostí byla také dokumentace.

MigrationPro se tedy zdál být slibným nástrojem na migraci, který poskytoval mnoho funkcí za malé peníze.

[11] Dle kurzu 1 euro = 25,3671 českých korun ze dne 4. března 2018

3.8.2 LitExtension

- **Web:** http://litextension.com/prestashop-migration-tool/opencart-to-prestashop.html

LitExtension je nástroj třetí strany, který umožňuje migraci informací o produktech, kategoriích, zákaznících, objednávkách, výrobcích, SEO (metadata), vlastních polích, recenzích a daních.

Jako velkou nevýhodu jsem viděl omezený počet řádků, které bylo možné migrovat. V základní verzi za 69 $ (cca 1 410 Kč) bylo možné zpracovat pouze 500 řádků z každé tabulky. Pokud by bylo potřeba převést celou historii objednávek (přes 80 000 řádků), musel bych si pořídit odpovídající verzi LitExtension, která by v tomto případě stála 699 $ (cca 14 400 Kč). Za přemigrování SEO URL všech kategorií a produktů bylo dalších 99 $ (cca 2 000 Kč).

Další velkou nevýhodou bylo jednorázové použití nástroje. Vzhledem k tomu, že jsme měli v plánu migrovat několik e-shopů, znamenalo by to zaplatit danou částku vícekrát, dohromady tedy 4 186 $ (cca 86 753 Kč). Teprve pak bychom měli přemigrována data do základních databázových tabulek PrestaShopu, bez jakýchkoliv přidaných či upravených tabulek. Zároveň by migrace dle webu trvala minimálně dva dny.

LitExtension na druhou stranu podporoval i upravené databáze, tedy uživatel si mohl nastavit, jaká pole byla přidaná či upravená, a ta se následně zmigrovala také.

V rámci balíčku byla roční technická podpora a dokumentace.

3.8.3 Cart2Cart

- **Web:** https://www.shopping-cart-migration.com/shopping-cart-migration-options/877-opencart-to-prestashop-migration

Cart2Cart podporuje migraci z/do více než 80 e-shopů, jednou z možností je samozřejmě i migrace z OpenCart do PrestaShopu. Skript přenáší informace o produktech, kategoriích, výrobcích, daních, zákaznících, objednávkách (i s historií), kupónech a recenzích.

Uživatel si může připlatit za dodatečné úpravy migrace, jako je zachování ID objednávek (49 $ (cca 1 000 Kč)), vytvoření automatického přesměrování z původního e-shopu (59 $ (cca 1 215 Kč)), migrace SEO URL pro kategorie a produkty (59 $ (cca 1 215 Kč)), zachování ID produktů (39 $ (cca 800 Kč)), zachování ID zákazníků (49 $ (cca 1 000 Kč)) či migrování obrázků produktů a kategorií (49 $ (cca 1 000 Kč)).

Finální cena se odvíjela od počtu zákazníků, objednávek a produktů a dala se spočítat pomocí jejich online nástroje na naceňování migrací. Z komunikace s technickou podporou vyplynulo, že pro takto velkou zakázku by nám udělali nějakou speciální nabídku (např. množstevní slevu). Pro správné nacenění

však požadovali administrátorské přístupy do všech e-shopů, což mi přišlo nesmyslné, a tak jsem migraci nacenil sám pomocí již zmíněného nástroje.

Finální cena za migraci všech e-shopů za použití Cart2Cart mi vyšla na 1 766 $ (cca 36 100 Kč) za data od roku 2017 a 1 956 $ (cca 40 000 Kč) za všechna data.

3.8.4 Vlastní migrační skript

Další možností samozřejmě bylo napsat si vlastní migrační skript, který by data z OpenCartu do PrestaShopu převedl.

Ač by tento postup byl zdlouhavý a komplikovaný, alespoň by se tím ulehčila migrace originálních OpenCart tabulek, které byly změněny, či nově přidaných tabulek.

Počítal jsem s již zmíněnými 20 minutami na migraci jedné tabulky, pro aktuálních 104 tabulek to činilo 2 080 minut, tedy 34.5 hodiny na přemigrování všech základních OpenCart tabulek, i těch později přidaných. S dalšími 5–6 hodinami na otestování, že vše funguje tak, jak má, jsem se dostal na cca 40 hodin práce pouze na migraci.

Velkou výhodou by byla 100% kontrola nad migrací jako takovou, stejně tak žádné externí náklady.

3.8.5 Shrnutí

Pro jednoduché srovnání jsem vytvořil tabulku 3.3 se všemi důležitými informacemi.

	MigrationPro	**LitExtension**	**Cart2Cart**	**Skript**
Vlastní tabulky	Ne	Ano	Ne	Ano
Počet běhů omezen	Ne	Ano	Ano	Ne
Data omezena	Ne	Ano	Ano	Ne
Cena 2017 (Kč)	2 450	73 282	36 100	0
Cena vše (Kč)	2 450	86 753	40 000	0
Čas (h)	15	65	20	40

Tabulka 3.3: Shrnutí možností migrace databáze

Z tabulky vyplývá, že ač je psaní vlastního migračního skriptu časově náročné, je to nejlevnější a nejefektivnější způsob, jak data z jedné databáze do druhé přemigrovat. Přesto jsem zkusil modul MigrationPro, neboť mohl práci velmi ulehčit, i když by byl schopný přemigrovat jen nezměněné tabulky. Nabízí totiž mnoho služeb za malé peníze.

Kapitola 4

Migrace databáze

V této kapitole je popsán postup, pomocí kterého jsem dosáhl úspěšné migrace customizované OpenCart 1.1 databáze na PrestaShop databázi. Popisuji zde jak migraci původních OpenCart tabulek za použití upraveného MigrationPro modulu pro PrestaShop, tak migraci tabulek, které vývojáři do databáze přidali v průběhu vývoje e-shopu.

Před započetím prací jsem musel podepsat dohodu o mlčenlivosti a zároveň si zašifrovat celý disk na počítači, na kterém jsem vyvíjel, protože jsem v pozdější části projektu pracoval i s osobními daty zákazníků.

4.1 Migrace OpenCart tabulek

Po dohodě s Jiřím Hunkou byla zakoupena licence MigrationPro, abych mohl vyzkoušet, do jaké míry mi může být nástroj při migraci prospěšný.

Nejdříve jsem zkusil přemigrovat testovací data, která mi vývojáři customizované administrace poskytli pro seznámení se s jejich systémem. Šlo o anonymizovaná data, která obsahovala nejdůležitější části databáze, jako jsou zákazníci, objednávky či produkty. Na druhou stranu chyběly například údaje o dopravcích či některé minoritní informace o objednávkách. Šlo pouze o to, abych měl představu o jejich závislostech a případných úskalích migrace.

4.1.1 Testování modulu

První spuštění migrace dle očekávání neprošlo, neboť v databázi byla nevalidní data (například prázdné jméno a příjmení zákazníka či cizí klíče s hodnotou 0). Prozatím jsem tyto problémy vyřešil SQL dotazem, který data přetransformoval na validní. Nešlo mi totiž o korektní migraci dat, ale abych zjistil, jakou část změněné databáze dokáže modul přemigrovat. Po úpravě dat již migrace doběhla v pořádku. Jedinou další změnou, kterou jsem musel udělat, byla změna maximální délky příjmení zákazníka, která sice v databázi byla nasta-

vena na 250 znaků, ale v PrestaShop entitě *Customer* byla omezena pouze na 35 znaků.

Dalším krokem bylo procházení kódu modulu, při kterém jsem se snažil pochopit propojení jednotlivých komponent, komunikaci mezi zdrojovým a cílovým e-shopem a způsob zpracování dat.

4.1.2 Propojení e-shopů

Propojení zdrojového a cílového e-shopu funguje velmi jednoduše – při nainstalování MigrationPro si kromě samotného modulu musí uživatel stáhnout i propojovací skript napsaný v PHP, který nahraje do předem dané složky zdrojového OpenCart e-shopu. Skript dále funguje jako prostředník – poskytuje MigrationPro modulu přístupový bod, se kterým může cílový e-shop komunikovat. Jediná věc, kterou je potřeba ve skriptu nastavit, je přístupový token, kterým se cílový e-shop při komunikaci autentifikuje. Stejný kód se pak zadává při konfiguraci modulu po jeho nahrání do PrestaShopu.

MigrationPro pak při svém běhu zasílá na propojovací skript požadavky, na které mu v případě shodujících se tokenů zdroj odpovídá požadovanými daty, ze kterých jsou poté vytvořeny příslušné objekty (Zákazník, Objednávka atd.). Ty se zpracují, případně se přes prostředníka dotáží na chybějící data, zvalidují se a vloží do cílové databáze.

4.1.3 Slabé stránky kódu MigrationPro

Samotný modul byl (jako celý PrestaShop) napsaný v PHP a využívá mnoho prestashopových tříd. Bylo na něm však vidět, že jeho vývoj probíhal nejednotně a neorganizovaně, neboť i méně zkušený programátor by v kódu hned po prvním projití našel několik míst, která nebyla pěkně řešená.

Jednalo se především o volání deprecated funkcí, kterých nebylo málo (což bylo způsobeno tím, že modul byl lehce pozadu za aktuální verzí PrestaShopu) či předávání více parametrů, než funkce požaduje. Pěkným příkladem je kód 4.1 ze souboru *OPImport.php*.

```
protected $image_path = '/image/';
public function setImagePath()
{
    $this->image_path = '/image/';
}
```

Ukázka kódu 4.1: Funkce setImagePath z modulu MigrationPro

Daná metoda se následně volala na mnoha místech jako *$import ->setImagePath($this->image_manufacturer);* apod. Tedy nejenom že metoda sama o sobě nastavovala proměnnou stále dokola na její defaultní hodnotu, ale ještě ignorovala nastavování správné cesty k obrázku. Při procházení

kódu jsem si často říkal, na jaké úrovni seniority vývojáři jsou, neboť na takovéto chyby upozorňuje každé lepší IDE.

Stejně tak kód obsahoval několik metod, které nikdy nebyly volány, popřípadě try-catch bloků, kde byla odchytávána výjimka typu *PrestaShopException*, která v dané části kódu nikdy nemohla být vyhozena.

Mnoho metod vracelo hodnotu pouze v některých cestách, často návratová hodnota chyběla úplně.

Na přehlednosti také ubíralo velké množství zakomentovaného kódu. V největší třídě modulu, která čítala cca 1 800 řádků, bylo přes 100 řádků zakomentovaných, tedy 5,5 %, což není málo, o mnoha TODOs ani nemluvě. Dle mého názoru by měl být kód, který je dostupný zákazníkům, od takovýchto věcí co nejvíce vyčištěn.

Také by celému modulu rozhodně prospělo přidání PHP anotací. Tím se kód také zpřehlední, protože IDE následně vývojářům našeptávají například dostupné metody a členské proměnné jednotlivých objektů, a sníží se tak pravděpodobnost překlepů či nesprávného volání.

Vývojáři také velmi bojovali s (vlastními) programovacími konvencemi. Například proměnná, ve které byl uložen objekt *Customer*, se jmenovala *customerObj*, o dvacet řádků pod ní se proměnná s objektem *Product* jmenovala *productObject* a o dalších dvacet řádků pod ní se proměnná s objektem *Attribute* jmenovala *AttributeModel*.

Na to, že celý modul obsahoval pouze cca deset tříd, tam bylo podobných chyb a nesrovnalostí velmi mnoho, což jeho pochopení o něco ztěžovalo.

4.1.4 Úprava modulu

Dle licence nesměl být modul jakkoliv upravován bez svolení autora. Protože jsem ale modul upravit potřeboval, abych mohl jednodušeji přemigrovat přidané a upravené sloupečky do customizované databáze OpenCart, kontaktoval jsem tvůrce modulu. Vysvětlil jsem mu svoji situaci a poprosil jej o svolení kód upravit a použít dle svých potřeb.

Svolení jsem dostal hned další den, a tak jsem využil příležitosti a tvůrcům nabídl, že nepěkné části kódu opravím, sjednotím vše potřebné a výsledný kód jim zašlu.

Zde je nutno poznamenat, že úpravy migračního modulu byly iterativní prací a musel jsem se k nim v průběhu migrace celého e-shopu několikrát vracet. Postupně jsem totiž zjišťoval různé nedokonalosti či chybějící data, ať v závislosti na následném propojování s administrací PrestaShopu, tak na konzultacích s vývojáři z Jagu.

4.1.4.1 Rozdíly ve struktuře

Migrací originálních OpenCart tabulek jsem strávil několik měsíců, neboť se customizovaná databáze od základní OpenCart databáze velmi lišila. Zároveň

MigrationPro modul některé OpenCart tabulky ignoroval úplně (jako jsou třeba převodníky hodnot mezi různými systémy, například palce na centimetry, či celou kolekci tabulek navázanou na atributy produktů).

Bohužel se mi nepodařilo získat původní strukturu OpenCart databáze, a tak jsem musel procházet databázi tabulku po tabulce, sloupeček po sloupečku a porovnávat ji jak s MigrationPro skriptem, tak s PrestaShop databází, a změny do nové databáze převádět ručně a po jedné.

Také bylo potřeba doplnit mnoho metod na získání dat ze zdrojového e-shopu, zpracovat data do podoby, se kterou pracuje PrestaShop, a následně je uložit. V rámci toho jsem musel upravit/doplnit třídy prestashopových entit, neboť každá tabulka má svůj vlastní model, který danou entitu reprezentuje[12].

4.1.4.2 Migrace informací o obchodech

Musel jsem si při migraci dát také pozor na provázanost jednotlivých dat, protože bylo potřeba korektně přemigrovat cizí klíče. Jedním z požadavků zákazníka totiž bylo, aby klíče všech entit zůstaly stejné, a tak jsem musel jejich správnou migraci ohlídat.

Důležitým bodem migrace bylo také korektní zpracování provázanosti dat na jednotlivé obchody. OpenCart ve verzi 1.1.1 totiž multisite nepodporoval, a tak si jej vývojáři customizované verze administrace museli doimplementovat sami. Bohužel se zdá, že vývoj této funkcionality probíhal ad-hoc, takže jednotlivé záznamy o obchodech nemají žádnou strukturu a nejsou tedy nijak provázany. Jde vždy o jeden sloupeček v tabulce, jejíž data mají být navázána na nějaký konkrétní e-shop(y). Ve sloupečku je pak uložena jedna nebo více URL obchodů, ke kterému daný záznam patří. V případě více obchodů jsou adresy odděleny novým řádkem.

PrestaShop pracuje s více propojenými obchody mnohem praktičtěji – v jedné tabulce uchovává informace o jednotlivých e-shopech (název, URL, jestli je obchod aktivní atd.) a v ostatních tabulkách se na tuto tabulku pouze odkazuje za pomoci cizích klíčů. Kde je potřeba spojit záznam s více obchody (například produkt je dostupný na více doménách), tam se využívá vztahová m:n tabulka.

Jako první krok migrace tedy bylo potřeba přemigrovat právě informace o obchodech, aby na ně mohlo být odkazováno v dalších tabulkách. Problémem bylo, že se seznam e-shopů přes všechny tabulky neshodoval – někde bylo záznamů více, někde jich bylo méně, ale za to byl mezi nimi záznam, který se jinde nevyskytoval. Nestačilo tedy získat seznam domén pouze z jedné tabulky, ale ze všech a výsledky následně sloučit (viz kód 4.2).

Bohužel ani uložení domén nebylo napříč jednotlivými tabulkami jednotné, částečně to bylo způsobené případem užití (u Dopravců bylo ve sloupečku *do-*

[12] V modelech se u každého atributu (reprezentujícího sloupeček v databázi) určují i metadata, například jestli je atribut povinný, jakého je datového typu či jestli na něm má před uložením proběhnout nějaká validace.

mains více domén, v Objednávkách byla ve sloupečku *domain* doména jen jedna), ale někde byla nejednotnost způsobena nedodržováním interních pravidel. Například v již zmíněné tabulce s Dopravci byly u většiny záznamů jednotlivé domény v seznamu odděleny novým řádkem (`\n`), v jednom případě ale byly domény odděleny čárkou. Podobných nesrovnalostí bylo v celé databázi více a čištění dat (či alespoň rozumné zpracování nevalidních dat) mi zabralo nemálo času.

Ze získaných dat pak bylo potřeba vytvořit odpovídající záznamy v PrestaShop databázi. Stačilo tedy naparsovat URL domény a získat z ní název e-shopu.

```
private $domainTableNames = [
        "order" => "domain",
        "category" => "domains",
        "category_banner" => "domains",
        "product" => "domains",
        "customer" => "domain",
        "manufacturer" => "domains",
    ];

private function getDomains() {
  $result = [];
  foreach ($this->domainTableNames as $table => $column) {
    $this->client->setPostData(
          $this->query->shops($table, $column)
        );
    $this->client->query();
    /* @var array $domains*/
    $domains = $this->client->getContent();
    foreach ($domains as $domainGroup) {
      $splitDomains=explode("\n",$domainGroup["domains"]);
      if (count($splitDomains) == 1) {
        $splitDomains=explode(",",$domainGroup["domains"]);
      }

      foreach ($splitDomains as $domain) {
        $domain = str_replace("\r", "", $domain);
        if (strlen($domain) &&
         !in_array($domain, $result)) {
          $result[] = $domain;
        }
      }
    }
  }
  return $result;
}
```

Ukázka kódu 4.2: Získání seznamu domén

Samozřejmě jsem celý seznam i jména domén mohl na požádání získat od někoho z Jagu, ale chtěl jsem, aby skript byl znovupoužitelný na jakékoliv další sadě e-shopů se stejnou (či alespoň podobnou) strukturou.

4.1.4.3 Migrace daní a stavů objednávek

Dále bylo potřeba přemigrovat informace o daních a stavech objednávek. Jagu chtělo obojí zanechat tak, jak to bylo v customizované verzi OpenCartu, nicméně MigrationPro modul samotné daně a stavy přemigrovat neuměl – uměl pouze namapovat zdrojová data na cílová. Musel jsem tedy dané nastavení přemigrovat ručně.

Problémem zde byla opět jiná struktura databáze, neboť OpenCart využíval jednu tabulku na stavy objednávek a dvě na typy daní, kdežto PrestaShop měl na stavy tabulky dvě a na daně jich měl dokonce pět. Musel jsem tedy zanalyzovat jednotlivé vztahy mezi tabulkami a data při vkládání příslušně rozdělit.

4.1.4.4 Migrace produktů

Migrační modul nefungoval moc dobře ani u produktů, kde úplně chybělo například migrování vlastností produktů, propojení produktů (produkt A souvisí s produktem B, například náhradní břity do holicího strojku) či varianty produktů (produkt A je variantou produktu B, například má jinou barvu). Bohužel zrovna u těchto dat nebyla migrace úplně přímočará, protože databázová struktura vlastností a variant byla mezi e-shopy velmi rozdílná.

Vlastnosti produktů byly v OpenCart uloženy bez jakékoliv návaznosti na jazyk – jak názvy vlastností, tak jejich hodnoty (pokud jsou textové) byly v češtině. Zato tam existovala určitá (nepovinná) vazba na kategorii, která dle slov vývojářů byla již v té době volnější a více se využívala vazba vlastností přímo na produkty. Zde jsem tedy při transformaci dat byl schopný rozdělit vlastnosti i jejich hodnoty do příslušných struktur, jak s nimi pracuje PrestaShop. Zapracoval jsem i rozdělení na jednotlivé jazykové mutace (i když byly texty pro všechny jazyky prozatím v původním jazyku, tedy češtině).

Uložení variant produktů bylo bohužel tak rozdílné, že migrace bez zásahu do struktury nebyla možná. Oba e-shopy totiž varianty pojímaly jinak, a tak i když se nazývaly stejně, nebyly úplně ekvivalentní. OpenCart totiž bral varianty jako rozdílné produkty, které byly propojené přes vztahovou m:n tabulku. Při procházení dat jsem zjistil, že jednotlivé produkty ani nemusely být od stejného výrobce, ale mohlo jít čistě o produkt s velmi podobnými specifikacemi. Z pohledu PrestaShopu byly varianty produktu chápány jako konkrétní produkt, u kterého se mění jeden nebo více parametrů. Po analýze problému jsem došel k závěru, že migrace tedy možná nebyla, aniž by nebyl porušen aktuální usecase. Musel jsem tedy v PrestaShopu vytvořit novou

vztahovou tabulku, která variantní vztahy ukládá, a data do ní přemigrovat bez jakékoliv transformace.

4.1.4.5 Migrace zákazníků

Migrace zákazníků byla víceméně přímočará. Všechny informace, které jsem potřeboval v PrestaShopu, byly uložené i v databázi OpenCartu. Chyběly pouze dvě drobnosti – PrestaShop ukládá i informace o pohlaví a datu narození. Ani jedna z informací v původní databázi obsažena nebyla, a tak jsem hledal způsob, jak problém vyřešit. Zpracování data narození jsem jednoduše v nastavení PrestaShopu vypnul, neboť původně bylo v registračním formuláři vyžadováno. Protože Jagu e-shopy již mají mnoho klientů a u nich se tato informace nesbírá, přišlo mi logické, že nebude potřeba tento údaj vyžadovat ani nadále. Tím jsem tedy docílil toho, že datum narození nemusím řešit ani při migraci, neboť bych jej neměl z ostatních údajů jak odvodit.

Informace o pohlaví sice v OpenCart databázi také uložená nebyla, ale dala se (alespoň částečně) odvodit z ostatních dat, konkrétně tedy ze jména a příjmení zákazníka. K tomu jsem využil jednoduchou funkci, kterou jsem léta testoval a vylepšoval ve webové aplikaci Seznamovák, která slouží pro účastníky a organizátory seznamovacího soustředění pro nově nastoupivší studenty FIT ČVUT.

Funkce čistě na základě jména a příjmení (konkrétně jejich posledních písmen) dokáže určit pohlaví s velkou úspěšností. Její výhodou je, že na rozdíl od většiny knihoven třetích stran, se kterými jsem se setkal, k tomu nepotřebuje databázi jmen. Po letech analýzy jmen a příjmení se mi podařilo vylepšit funkci tak, že při určování pohlaví účastníků Seznamováku 2018 měla úspěšnost 99.53 %. Jediné špatné určení ještě k tomu bylo způsobeno zadáním domácí podoby jména, protože účastník do registračního formuláře zadal „Honza“ místo „Jan“ a algoritmus jej tedy mylně považoval za dívku, neboť mužská jména v češtině na „a“ nekončí.

Algoritmus (kód 4.3) jsem musel pro potřeby migrace trochu poupravit, protože ne všechny údaje byly ve správném formátu nebo validní obecně. Často obsahovaly nadbytečné bílé znaky, byly napsané kapitálkami či byl ve jméně i titul.

Je nutno poznamenat, že algoritmus je zaměřený na jména českého a slovenského původu, nicméně funguje i pro některá ruská. Se jmény jiného původu nebude pravděpodobně fungovat tak dobře, ale pro aktuální potřeby byl dostačující, neboť drobné chyby se daly velmi jednoduše opravit ručně – stačilo prostě změnit pohlaví přímo v administraci či v uživatelově profilu.

Testování kvality algoritmu jsem bohužel mohl provádět pouze tak, že jsem ručně porovnával jména zákazníků a k nim přiřazená pohlaví. Zkontroloval jsem takto všech 1 753 záznamů, ze kterých ale bylo 46 nevalidních – buď byly v databázi místo jména/příjmení uloženy náhodné sekvence znaků, nebo byl místo příjmení vyplněn třeba název firmy. Zbylo tak 1 707 záznamů, které

šlo korektně zpracovat. Z nich bylo 9 cizích jmen, u kterých jsem většinou nebyl schopný určit pohlaví ani nestrojově (například asijská jména).

Ze zbylých 1 698 záznamů algoritmus špatně určil pohlaví jenom u jednoho, jeho správnost je tedy 99.94 %.

```
private function getGender($surname, $name) {
  $surname = $this->sanitizeName($surname);
  $name = $this->sanitizeName($name);
  return (preg_match('/á$/', $surname)
    || preg_match('/enko$/', $surname)
    || preg_match('/ova$/', $surname)
    || preg_match('/a$/', $name)
    || (
       preg_match('/ina$/', $surname) &&
       preg_match('/a$/', $name)
     )
   ) == true?2:1;
  // Muž má v PrestaShopu ID 1, žena 2
}
function sanitizeName($name) {
  $name = preg_replace('/([\S]+\.)/', '', $name);
  $name = strstr($name, ',', true) ?: $name;
  $name = trim($name);
  return mb_strtolower($name);
}
```

Ukázka kódu 4.3: Určení pohlaví podle jména

4.2 Migrace přidaných tabulek

Protože modul MigrationPro byl primárně zaměřen na migraci OpenCart databázových tabulek (jejichž název začíná prefixem *oc_*), bylo potřeba, abych si kód na přemigrování tabulek, které byly přidány vývojáři customizovaného e-shopu, napsal sám. Důležité bylo rozhodnutí, jestli jejich migraci doimplementovat do modulu MigrationPro, nebo udělat nějaký externí skript, nezávislý na zbytku migrace.

4.2.1 Migrace přes externí skript

Nejdříve jsem se rozhodl pro externí skript, který jsem naimplementoval v Bashi[13]. Využíval jsem v něm již zmíněného pojmenovávání tabulek, díky kterému jsem si mohl v databázi vyhledat všechny tabulky, co nepatří OpenCartu (tedy jejich název nezačíná prefixem *oc_*). Do souboru jsem pak vyexportoval jejich strukturu a data a soubor poté naimportoval do prestashopové databáze.

[13]Bash je jeden z unixových interpretů pro vytvoření příkazového řádku.

Skript fungoval dobře pro základní migraci, ale měl několik zásadních chyb, které jsem si uvědomil až později. Zaprvé počítal s tím, že všechny primární klíče zůstanou stejné i v cílové databázi (a tedy budou správně fungovat propojení přes cizí klíče), což nemusí být vždy chtěné. Zadruhé by skript nefungoval při migraci více zdrojových e-shopů do jednoho cílového, protože se v rámci něho vždy prováděl DROP na dané tabulky a data by tak byla ztracena. I kdybych tuto možnost vypnul, opět by mohl nastat problém se zachováním správných cizích klíčů. Skript jsem tedy využil pouze na vytvoření databázové struktury v cílové databázi a přemigrování dat, která byla stejná přes všechny databáze (například základní informace o dopravcích atd.).

Zároveň jsem později přidával příkaz, který měnil názvy některých sloupečků, abych dodržel štábní kulturu PrestaShopu. OpenCart totiž pojmenovává sloupečky s primárními klíči jako *table_name_id*, zato PrestaShop vkládá *id* na začátek názvu, tedy *id_table_name*. Musel jsem tedy ve vygenerovaném souboru s SQL příkazy provést záměnu jak ve struktuře, tak v datech.

4.2.2 Migrace přes migrační modul

Pak jsem také zjistil, že ne všechny tabulky začínající prefixem *oc_* jsou původní OpenCart tabulky – některé přidali vývojáři custominované verze e-shopu. Tyto tabulky tedy bylo potřeba přemigrovat také.

Jedna z tabulek, pro kterou jsem migrace musel přidávat, byla tabulka overridů, tedy přepsání určitých dat. Šlo například o případy, kdy produkt měl na jednom obchodě jiná metadata pro vyhledávače než na jiných obchodech. Vzhledem k tomu, že původní OpenCart 1.1.1 funkcionalitu více obchodů nepodporoval, nebylo vůbec jednoduché tyto drobné změny v datech nějak rozumně ukládat a spravovat. Jedinou možností byly právě tabulky *oc_domain_override* a *oc_url_alias*.

4.2.2.1 Tabulka oc_domain_override

Tabulka oc_domain_override ukládala právě již zmíněná metadata, jako například klíčová slova produktu či popis kategorie. Každý řádek obsahoval doménu, ke které se záznam vztahoval, identifikátor jazyka, a pak hodnoty a identifikátory přepisovaného záznamu.

PrestaShop s více obchody již počítá, a tak je jeho databáze na podobné případy připravena. Objekty, u kterých to je potřeba (například produkty, kategorie), mají kromě základní tabulky ještě tabulku s postfixem *_lang*, ve které jsou uložené hodnoty specifické pro různé jazky a obchody. Například je možné nastavit meta tagy pro produkt tak, aby byly různé pro jednotlivé obchody, a mít pro ně i překlady pro všechny podporované jazyky.

Nevýhodou nicméně je, že počet záznamů v tabulce tak velmi rychle narůstá. Pokud spravuji šest obchodů, které podporují tři jazyky, je pak pro jeden produkt v tabulce *ps_product_lang* 18 záznamů. To je v porovnání

s řešením v OpenCartu, kde byly záznamy 3 pro každý jazyk, opravdu velký nárůst. Samozřejmě vše má své výhody a nevýhody. PrestaShop sází na flexibilitu úprav oproti počtu záznamů, což je vzhledem k tomu, jakých rychlostí dosahují dnešní databáze a stroje, na kterých běží, dle mého názoru rozumný krok.

Při migraci tabulky *oc_domain_override* jsem tedy mohl jednoduše určit, jaký záznam se má přepsat jakými daty pro určitý jazyk a doménu, aniž bych kvůli tomu musel vytvářet speciální tabulku. Tím také odpadla potřeba se do ní dotazovat při každém přístupu uživatele na danou stránku.

4.2.2.2 Tabulka oc_url_alias

Tabulka oc_url_alias byla na přemigrování výrazně složitější. V této tabulce nebyla uložena metadata, ale informace o tom, jaká se mají provádět přesměrování při přístupech na určité stránky. Informace byly opět vázané na jednotlivé domény. Příkladem tedy může být záznam, který znamená: „pro doménu *www.stylka.sk* přesměruj *www.stylka.sk/category_id=10* na *www.stylka.sk/zastrihavace-chlpkov*“.

Protože ale bylo potřeba URL občas měnit, měl každý záznam ještě příznak *moved*, který (pokud byl nastaven) udával ID záznamu v téže tabulce, na který měla být URL dále přesměrována. Může se tedy stát, že při přístupu na některou ze starých URL může být přesměrování (nebo alespoň rekurzivních vyhledávání v tabulce) provedeno několik. I s tím jsem musel při migraci počítat.

Vzhledem k tomu, že jedním z požadavků bylo, aby všechna URL (produktů, kategorií. . .) zůstala stejná, tedy stále platily odkazy například z vyhledávačů, bylo kritické, aby údaje z této tabulky byly přemigrovány bez chyby.

Bohužel jsem při práci opět narazil na problém nekonzistencí a nevalidity. Některé záznamy měly totiž příznak *moved* nastaven na ID záznamu, které v tabulce neexistovalo. Dále pak data ve sloupečku *domain*, který opět sloužil k určení obchodu, ke kterému se daný záznam vztahuje, nebyla konzistentní. Z ostatních tabulek jsem vypozoroval, že pokud byl sloupeček nazván *domain*, byla v něm vždy uložena jedna URL ve formátu www.xyz.cz. pokud se jmenoval *domains*, mohlo v něm být uloženo více URL v daném formátu, většinou oddělených odřádkováním. Ve sloupečku *domain* v této tabulce však byly uloženy záznamy nejen ve formátu „www.xyz.cz“, ale i „http://www.xyz.cz/“, „www.xyz.cz**,**“ a „www.xyz.cz,www.abc.cz“. Musel jsem tak před zpracovávání dat provést také jejich čištění a validaci.

Následně jsem vzal vždy nejaktuálnější hodnotu URL (tedy tu, co neměla nastavený příznak *moved*) a pokud šlo o produkt, kategorii či výrobce, tak ji uložil do příslušné tabulky s postfixem *_lang* do sloupečku *link_rewrite*, který určuje, pod jakou URL bude daný objekt dostupný. Tím jsem zaručil, že všechny URL objektů budou stejné i po spuštění nové verze e-shopů.

Nicméně bylo potřeba uložit i staré podoby jednotlivých URL, aby došlo ke správnému přesměrování. Bohužel PrestaShop tuto funkcionalitu nepodporoval, a tak jsem příslušnou tabulku musel vytvořit. Dotazování se do této tabulky při reálném běhu je však otázkou frontendu, tedy je mimo rozsah této diplomové práce.

4.2.2.3 Tabulka oc_domain_setup

Další tabulkou s nastavením hodnot pro různé domény byla například tabulka *oc_domain_setup*, kde nejdůležitějším údajem byly informace o dopravcích. Původní e-shopy spolupracují s takovými firmami jako je Česká pošta či Geis, pro které si obchody ukládají různé nastavení, které bylo potřeba také přemigrovat. Pro PrestShop sice existují nějaké moduly, které tyto dopravce do administrace přidávají, ale Jagu má v plánu si tuto funkcionalitu vytvořit samo. Migrování těchto údajů do nějakých speciálních tabulek by tedy prozatím bylo zbytečnou prací, neboť v danou chvíli nebylo jasné, jak bude nová struktura dopravců vypadat. Rozhodl jsem se proto tyto údaje prozatím přemigrovat tak, jak byly uložené v původním e-shopu.

4.3 Problémy při migraci

Největším problémem při celé migraci byla nevalidita dat. Částečně tak bylo proto, že v databázi byla uložena i data z doby, kdy ještě na původním e-shopu validace při odesílání formulářů nebyly vůbec, nebo pouze v omezené míře. To mělo za následek ku příkladu data, kde ve sloupečku PSČ byla uložena písmena, jako telefonní číslo uživatel zadal svůj email či jako příjmení své telefonní číslo.

S takovýmito daty se velmi těžko pracovalo, protože je nešlo opravovat jinak než ručně – projít záznam po záznamu a změnit vše potřebné. To bohužel bylo velmi časově náročné a po dohodě s hlavním vývojářem customizovaných e-shopů jsem opravil data, která šla opravit jednoduše, a zároveň jsem zatím omezil podmínky validity pro jednotlivá pole. Se spoustou dat se totiž již nepracuje, a tak nevadí, že jsou nevalidní.

Druhý typ nevalidních dat nebyl způsoben chybou, ale záměrně. Šlo například o objednávky s prázdným jménem a příjmením zákazníka či údaje, kde jako cizí klíč byla nula (a tedy klíč referencoval neexistující záznam). Nebylo tedy potřeba (a ani záhodno) je nijak řešit, splňovaly totiž usecase firmy.

Bohužel se několikrát stalo, že migrace typu dat, která již fungovala, přestala fungovat s novými daty. Pokud například přibyl nový zákazník se jménem, které nebylo validní, musel jsem přidat další čištění záznamů, aby migrace prošly. Takto jsem se k již hotovým migracím vracel velmi často, abych vůbec mohl v práci pokračovat.

Dalším problémem byl objem dat, protože některé tabulky obsahovaly desetitisíce záznamů či byly provázány s mnoha dalšími tabulkami. Migrační

modul sice umožňoval nastavit velikost dávek, po jakých se budou data migrovat, ale všechny možnosti mají své výhody a nevýhody. Musí totiž vždy dojít k připojení na zdrojový server, získání dat a následné zpracování. Pokud se data získávají po malých dávkách, tak se dotazů musí udělat mnoho a migrace trvá dlouho. Pokud se data získávají po velkých dávkách, trvá dlouho jejich transformace, a tak může na jednom ze serverů dojít k timeoutu a celý proces se ukončí. Bylo tedy potřeba najít zlatou střední cestu, která byla bohužel závislá na síle serveru.

Když jsem migraci testoval na svém počítači, tak jak zdrojový, tak cílový e-shop byly na jednom stroji, a tak přenášení dat bylo mnohem rychlejší než na výsledném produkčním prostředí. Tam jsem musel dávky snížit oproti testovacímu na čtvrtinu, jinak se migrace zasekávala právě kvůli timeoutům. Zpracovávání objemnějších struktur, na které byla navázána spousta dalších tabulek a u kterých bylo potřeba provádět složitější transformace, totiž mohlo zabrat i několik sekund.

Snížení velikosti dávek vedlo k tomu, že doba běhu skriptu nakonec přesáhla i deset hodin. Kvůli tomu se celý skript velmi obtížně testoval, neboť jsem musel často několik hodin čekat, až se migrace dostane do fáze, kde vzniká nějaký problém.

Zpomalení migrace na produkčním prostředí bylo také způsobeno faktem, že se při zpracování dat z databáze kopírovaly i obrázky spojené s příslušným záznamem (například náhledy produktů). To v případě mnoha souborů prodloužilo migraci o jednotky sekund na každý záznam. V případě již zmíněných produktů, kde bylo ke každému záznamu obrázků více, znamenalo celkové prodloužení doby běhu i o několik hodin.

Samozřejmě šlo spustit migraci například pouze pro Objednávky, ale aby se vše správně propojilo přes cizí klíče a já mohl zkontrolovat, že jsou data v pořádku, musel jsem zároveň přemigrovat i Produkty a Zákazníky (a s nimi spojené pomocné tabulky).

Jeden z problémů jsem si způsobil dokonce sám. I když jsem s vedoucím předem prošel celý e-shop a eliminoval velkou část tabulek a sloupečků, které nebylo potřeba migrovat, jejich detailní seznam jsem si nechal zkontrolovat až v průběhu migrace. Našlo se tak několik sloupečků a tabulek, které jsem již přemigroval, ale v původních e-shopech se nepoužívaly, takže jsem musel kód na migrování opět smazat.

Celkově mi migrace dat nakonec zabrala 193 hodin, což se od mého odhadu v sekci 3.3 liší o skoro pětinásobek. Musím uznat, že mě časová náročnost migrace překvapila, už z toho důvodu, že jsem vkládal docela veliké naděje do MigrationPro. I když mi modul bezpochyby v práci velmi pomohl, pořád mu k dokonalosti chybělo mnoho a paradoxně jsem strávil více času úpravami modulu, díky kterým jsem mohl přemigrovat původní OpenCart tabulky, než přidáváním nové funkcionality pro migraci tabulek přidaných do původní databáze.

Počet řádků kódu pouze v PHP třídách modulu se úpravami zvedl z 5 677

na 8 062 (tedy nárůst o 42 %). I z toho je vidět, že modul nebyl na migraci této databáze dobře uzpůsoben a bylo na něm potřeba udělat opravdu hodně práce.

Kapitola 5

Upravení PrestaShopu

Když jsem měl migraci hotovou, byl čas zaintegrovat přidané sloupečky a tabulky do administrace. Při rozhodování, jak administraci upravit, jsem se řídil administrací customizovaného e-shopu, ale také vše konzultoval s jeho vývojáři. Ne všechna přemigrovaná data bylo potřeba zobrazovat, na druhou stranu byly i informace, které se v customizovaných e-shopech nezobrazovaly, ale pouze z důvodu, že se nikdo nedostal k implementaci příslušných sekcí.

Jedním z bodů zadání bylo také zajištění, že půjde PrestaShop dále jednoduše aktualizovat, aby projekt rychle nezastaral jako předchozí customizovaná verze OpenCartu. Projekt by se tak za pár let opět dostal do stejných problémů. Jednotlivé možnosti aktualizace byly naštěstí popsané ve vývojářské dokumentaci PrestaShopu [23].

5.1 Problémy při rozšiřování platformy

Velkým problémem při implementaci nových funkcí byl fakt, že PrestaShop od verze 1.7 postupně přecházel z jejich PrestaShop frameworku (týmem PrestaShop nyní nazývaný legacy), který používal Smarty šablony, na framework Symfony (týmem PrestaShop nazývaný modern) a jeho renderování přes Twig. Bohužel z technických a časových důvodů probíhal přechod velmi pomalu, a tak byly některé části platformy napsány v legacy a části v modern. Musel jsem se tedy naučit oba dva frameworky, abych mohl upravit vše potřebné. Strávil jsem několik dní pouze pročítáním vývojářské dokumentace jak PrestaShopu, tak samotného Symfony, stejně tak Smarty a Twig šablon.

Bohužel postupný přechod také znamenal, že jakákoliv má změna v legacy kódu může přijít v brzké době vniveč, protože daná třída/šablona může být s dalším updatem nahrazena její modern verzí a mnou udělané úpravy bude potřeba udělat znova, tentokrát v Symfony. Výhodou ale je, že jednotlivé aktualizace PrestaShopu přichází z docela velkými rozestupy a samotné přepisování do modern verze trvá dlouho (migrace na Symfony probíhala v době psaní této

práce již dva roky a stále nebyla hotová), a tak nebudou nároky na úpravu mého kódu vysoké.

Nicméně i tak byla práce na úpravách PrestaShopu náročná, neboť nebylo vždy jednoduché poznat, který kód patří ke které stránce a co všechno je potřeba upravit, aby propojení fungovalo správně. I v modern kódu byl totiž občas volán nějaký legacy kód, protože ještě neexistovala jeho Symfony podoba, a tak se mohly některé funkce zdát zmatečné a záměry vývojářů PrestaShopu těžké na pochopení. Věřím ale, že postupným přemigrováváním do nového frameworku se přehlednost celého projektu zvýší a kód bude lehčí na upravování.

Zároveň se nabízí otázka, jestli těmto problémům nebylo možné předejít, kdybych při průzkumu trhu neporovnával jen funkce, použitelnost a design dané platformy, ale také interní specifikace, jako je právě členění a přehlednost kódu, stejně tak jako obtížnost rozšiřitelnosti platformy.

Při snaze odpovědět si na tuto otázku jsem došel k názoru, že pokud bych měl zkoumat kód všech analyzovaných platforem více dopodrobna, přidalo by mi to desítky hodin práce jenom na rešerši, ze které by s velkou pravděpodobností nakonec beztak vyšel nejlépe PrestaShop. Ten totiž přes ne úplně šťastné období přerodu z legacy frameworku na modern framework nabízí největší pokrytí požadovaných funkcí, čímž snižuje počet úprav kódu, které jsou potřeba udělat. Nedostatky platformy při jejím rozšiřování tedy nebudou tak omezující, jako by mohly být u jiných platforem.

V potaz je také potřeba vzít i fakt, že ani ostatní platformy by nebyly v rozšiřitelnosti dokonalé. E-shopy často bývají velké projekty se stovkami tříd a statisíci řádky kódu, na kterých pracují desítky lidí (někdy i z komunity, vzhledem k tomu, že ty nejpoužívanější e-shopy jsou open-source), takže je náročné vyvíjet i samotný jejich základ. Možnost kód dále rozšiřovat, například pomocí modulů, jak to dělá PrestaShop, klade na vývojáře velmi velké nároky, které je nutí psát kód co nejvíce modulární a zároveň umožňovat třetím stranám jej bez větších problémů rozšířit, aniž by přišly o základní funkce.

Tento úkol je dle mého názoru skoro až nemožný, neboť napsat program, který umožňuje 100% rozšiřitelnost bez zásahů do core souborů, je v takovém měřítku, v jakém jsou zdrojové kódy e-shopů, projektem sám o sobě. Jeho řešitel by si jistě vysloužil nemalé uznání všech vývojářů, kteří kdy pracovali na nějakém open-source kódu o takovéto velikosti.

5.2 Zachování možnosti aktualizace PrestaShopu

Aktualizace takto rozsáhlého kódu, který může být kvůli modulům a ručním změnám velmi rozdílný mezi jednotlivými e-shopy, není jednoduchá. Při některých aktualizacích dochází pouze k opravám funkcí či optimalizaci SQL dotazů, jiné ale mažou třídy či mění strukturu databáze. Proto samotní vývo-

jáři PrestaShopu v současnosti rozlišují dva možné typy aktualizace – upgrade a migraci [24].

5.2.1 Upgrade

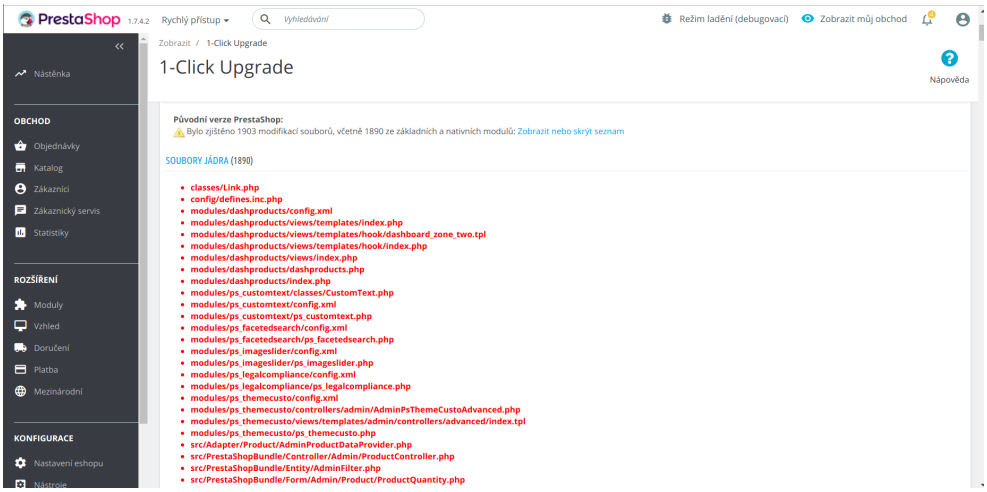
Upgrade se u PrestaShopu používá při aktualizaci minor verze produktu, tedy například přechod z verze 1.6.0 na 1.6.1 nebo z verze 1.7.1 na 1.7.2. Tomuto druhu se také říká in-place upgrade, neboť celý proces probíhá na jednom obchodu (serveru), aniž by uživatel (vývojář) potřeboval nějakou další kopii celého obchodu.

Upgrade minor verze přináší pouze drobné změny či opravy chyb a neměl by mít dopad na funkčnost modulů či vést ke ztrátě dat.

Tento typ aktualizace je velmi jednoduchý na provedení, stačí totiž spustit proces přes oficiální PrestaShop modul 1-Click Upgrade, který celou aktualizaci provede. Zároveň před začátkem procesu udělá zálohu celého obchodu, a tak při jakékoliv chybě může vrátit celý PrestaShop do původního stavu [25].

V průběhu psaní této práce vyšla nová verze PrestaShopu (vývoj začínal na verzi 1.7.4.0, v červenci 2018 byla vydána verze 1.7.4.2), a tak jsem si mohl upgrade rovnou vyzkoušet v praxi ve svém vývojářském prostředí.

Před upgradem modul 1-Click Upgrade ukáže seznam souborů, které byly ručně změneny (obr. 5.1), takže vývojáři vědí, o které části kódu můžou teoreicky přijít. Proto je důležité neupravovat soubory přímo, ale za pomoci modulů či overridů, jak bude popsáno na následujících stránkách.



Obrázek 5.1: Seznam změněných souborů při práci s modulem 1-Click Upgrade

Při upgradu samotném jsou pak na stránku vypsány všechny provedené změny, jak v souborech (tam pouze jejich jména), tak jednotlivé SQL příkazy, které byly provedeny. Díky tomu jsem mohl změny databáze zanést i do mých vývojářských migračních skriptů, jinak by se stalo, že by na serveru sice byly

aktualizované soubory (které by se tam přenesly přes git), ale databáze by stále byla původní.

5.2.2 Migrace

Migrace je již o něco složitější a provádí se hlavně při přechodu na novou major verzi produktu (aktuálně z verze 1.6 na 1.7). Jde o out-of-place migraci, tedy migruje se z obchodu se starou verzí PrestaShopu na novou verzi PrestaShopu, která již běží paralelně s původní na jiném serveru.

Při této migraci je velmi pravděpodobné, že se některé moduly či funkce stanou nedostupné, stejně tak že se nějakým výrazným způsobem změní databáze, a tak je často potřeba udělat při přechodu mezi verzemi změny v kódu či v datech. Proto migrace probíhá out-of-place, dává tak vývojářům čas dodělat případné změny. Kupříkladu verze 1.7 je oproti verzi 1.6 z části postavena na PHP frameworku Symfony, byl předělán systém práv uživatelů, stejně jako šablony obchodu.

S tímto typem aktualizace samozřejmě souvisí i migrace dat z původního obchodu na nový. Ta bohužel nemusí být úplně triviální a vždy záleží na tom, jaké změny byly zrovna v databázi provedeny. Pomocníkem tak může být další produkt MigrationPro (jejichž modul jsem využil při migraci dat z OpenCartu na PrestaShop), tentokrát modul na migraci z PrestaShopu na PrestaShop.

Migraci také mohou vývojáři provést ručně, buď zdlouhavým ETL[14] přes CSV soubory, nebo více technickým způsobem, a to přes PrestaShop API, jak je popsáno ve vývojářské dokumentaci. Sami vývojáři PrestaShopu doporučují migraci právě přes API, protože PrestaShop webservice se za poslední roky prakticky nezměnila, a tak je mezi verzemi velmi kompatibilní [26]. Stejně tak API poskytuje jednoduchý přístup jak ke všem databázovým objektům a jejich položkám, tak k jejich jazykovým mutacím.

5.3 Aktualizace customizovaného kódu

V tomto ohledu je naštěstí PrestaShop velmi silným nástrojem, protože na rozdíl od OpenCartu se změnami v kódu počítá, a tak má speciální strukturu jak adresářů, tak tříd, která umožňuje jednoduché úpravy designu i funkcí.

V kořenu adresářové struktury prestashopové aplikace je složka *classes*, ve které jsou uloženy třídy reprezentující jak jednotlivé stránky frontendu (například seznam produktů či Košík), tak entity reprezentující jednotlivé databázové tabulky (*Address*, *Order*, *Product*...).

Na stejné úrovni je složka *overrides*, kam mohou vývojáři ukládat své úpravy kódu. Pokud cesta ke třídě ve složce *override* odpovídá cestě ke třídě

[14]ETL neboli Extract-Transform-Load je technika, při které jsou data nejdříve vytažena z databáze, pak jsou upravena dle potřeby (sloupečky jsou smazány, přidány atd.) a pak znovu nahrána do databáze (v tomto případě na cílovém obchodě).

ve složce *classes* a zároveň jsou třídy pojmenované stejně, tak jsou metody v originální třídě přepsány/doplněny metodami z přepisující třídy.

Například chci přepsat třídu *Order* ze složky *classes/Order/Order.php*, která vypadá takto:

```
class OrderCore extends ObjectModel
{
    /** @var int Delivery address id */
    public $id_address_delivery;

    /** @var int Invoice address id */
    public $id_address_invoice;

    //...další atributy...

    public static $definition = array(
        'table' => 'orders',
        'primary' => 'id_order',
        'fields' => array(
            'id_address_delivery' =>
                array('type' => self::TYPE_INT,
                'validate' => 'isUnsignedId'),
            'id_address_invoice' =>
                array('type' => self::TYPE_INT,
                'validate' => 'isUnsignedId'),
        //...další definice databázových sloupečků...
    )

    /**
     * @see ObjectModel::getFields()
     * @return array
     */
    public function getFields()
    {
        if (!$this->id_lang) {
            $this->id_lang = Configuration::get(
             'PS_LANG_DEFAULT',
             null,
             null,
             $this->id_shop);
        }

        return parent::getFields();
    }
}
```

Ukázka kódu 5.1: Příklad třídy Order

Ve složce *overrides/classes/Order/* vytvořím soubor *Order.php*, který může vypadat následovně:

```php
class OrderCore extends ObjectModel
{

    /** @var Date order date */
    public $order_date;

    public function __construct(
          $id = null,
          $id_lang = null) {
        self::$definition['fields']['order_date'] = array(
         'type' => parent::TYPE_DATE,
         'validate' => 'isDATE');
        parent::__construct($id, $id_lang);
    }

    public function findSimilar() {
         ..implementace metody findSimilar..
    }
}
```

Ukázka kódu 5.2: Příklad rozšíření třídy Order

Výsledná třída *Order* vygenerovaná ClassLoaderem by vypadala asi takto:

```php
class OrderCore extends ObjectModel
{
    /** @var Date order date */
    public $order_date;

    /** @var int Delivery address id */
    public $id_address_delivery;

    /** @var int Invoice address id */
    public $id_address_invoice;

    //...další atributy...

    public static $definition = array(
        'table' => 'orders',
        'primary' => 'id_order',
        'fields' => array(
            'id_address_delivery' =>
                 array('type' => self::TYPE_INT,
                 'validate' => 'isUnsignedId'),
            'id_address_invoice' =>
                 array('type' => self::TYPE_INT,
                 'validate' => 'isUnsignedId'),
            'order_date' =>
                 array('type' => parent::TYPE_DATE,
                 'validate' => 'isDATE'),
```

```
        //...další definice databázových sloupečků...
    )

    public function findSimilar() {
         ..implementace metody findSimilar..
    }

     /**
     * @see ObjectModel::getFields()
     * @return array
     */
    public function getFields()
    {
        if (!$this->id_lang) {
            $this->id_lang = Configuration::get(
             'PS_LANG_DEFAULT',
             null,
             null,
             $this->id_shop);
        }

        return parent::getFields();
    }
}
```

Ukázka kódu 5.3: Příklad výsledné třídy Order

PrestaShop takto vývojářům umožňuje mít systém vždy připravený na aktualizaci na novou verzi a zároveň nedochází ke konfliktům s již upraveným kódem.

5.4 Rozšíření PrestaShopu

I při samotném rozšiřování PrestaShopu je možné vybrat si z více cest:

- přímé upravování souborů,
- override tříd a přímá úprava šablon,
- vlastní modul.

Nejjednodušší způsob je upravovat soubory PrestaShopu přímo, ale jak již bylo řečeno, při této možnosti by bylo velmi těžké udržet možnost aktualizace, a tak jsem tuto možnost zavrhl ihned. Nemluvě o tom, že od této možnosti odrazují i sami vývojáři PrestaShopu.

Override tříd, jak byl popsán v předchozí sekci, je silnou a slibnou možností, ale bohužel tento způsob funguje jen pro PHP třídy a renderovací šablony je nutné upravovat ručně. Problém aktualizací by tak byl vyřešen pouze částečně a dlouhodobě by mohl přinášet problémy.

Rozhodl jsem se tedy napsat vlastní modul, který bude všechny potřebné funkce do PrestaShopu přidávat. Moduly totiž využívají nejenom overrides, ale i takzvané hooky (hooks)[15], které umožňují vkládat/upravovat kusy šablon při jejich vykreslování.

5.4.1 Hooky

Hooky jsou v Symfony frameworku eventy, k jejichž odběru se můžou různé služby (například právě moduly) přihlásit. V případě vyvolání daného hooku o tom budou přihlášené služby notifikovány. V administraci PrestaShopu se zároveň dá jednoduše nastavit, v jakém pořadí budou odběratelé daného hooku informováni o vyvolání dané akce, takže se dá upravit například pořadí vykreslení části šablon či volání funkcí na zpracování dat.

Nevýhodou hooků je vyšší náročnost na naprogramování, protože se tím do kódu přidává nemálo logiky. Například pokud je potřeba v přehledové tabulce Produktů přidat nový sloupeček, tak místo prostého upravení daného kódu je potřeba vytvořit celou novou strukturu souborů a volání funkcí.

Nejdříve musí vývojář detekovat, jaké hooky jsou na stránce se seznamem produktů volány. Při tom velmi pomůže Symfony Profiler [27], tedy ladicí nástroj, který (kromě jiného) detekuje, jaké šablony byly vykresleny skrz které hooky a volání. Následně je potřeba zjistit, které z hooků potřebujeme odebírat a jejich volání zaregistrovat v námi vytvářeném modulu. Bohužel registrování hooků se provádí při instalaci modulu, což znamená, že je nutné pro přidání nového hooku modul odinstalovat a znovu nainstalovat, jinak k úspěšnému odběru nedojde.

Poté je potřeba vytvořit v modulu funkci, která bude daný hook zpracovávat (například vytáhne potřebná data z databáze či je naopak do databáze uloží).

Ve srovnání s prostým upravením souborů je složitost mnohonásobně vyšší. Zároveň je problémem také to, že ne všechny stránky jsou přepsané do Symfony, a tak Profiler není dostupný všude. Bohužel ani u již přepsaných stránek hooky nepokrývají všechny potřebné části renderování, a tak je často nutné vytvořit si hooky vlastní a ty potom volat v šabloně dané stránky. Kvůli tomu je potřeba šablonu přetížit pomocí overridů.

Overridy bohužel nejsou možné pro všechny třídy – ani v dokumentaci, ani na PrestaShop foru jsem nenašel způsob, jak některých rozšíření dosáhnout, a tak jsem se místy musel uchýlit k úpravám souborů napřímo. Takto jsem musel upravit pouze tři soubory, všech ostatních změn se mi podařilo dosáhnout pouze díky hookům a overridům tříd a šablon.

[15] V české verzi PrestaShopu je funkce přeložena jako „háky“, ale tento výraz mi nepřijde úplně šťastný, a tak budu nadále používat anglickou variantu.

5.4.2 Implementace modulu

Overridy, jak jsou popsány v sekci 5.3, je možné mít na dvou místech – přímo ve složce *override* v kořeni projektu, nebo ve složce *modules/module_name/overrides*, pro jejíž strukturu platí stejná pravidla. Problémem zde ale je, že PrestaShop overridy z této složky nenačítá – načítá je pouze ze složky v kořeni projektu. Při instalaci modulu totiž dochází k překopírování všech override souborů typu .php do hlavní *override* složky. To bohužel velmi ztěžuje vývoj, neboť při každé změně nějaké z tříd v dané složce je potřeba modul odinstalovat a znovu nainstalovat, což může zabrat i několik desítek sekund, nezávisle na tom, jestli programátor zrovna přidal několik tříd, nebo upravil jeden řádek kódu. Proto jsem vyvíjel vše ve složce *override* v kořeni projektu a pak až přesunul všechny soubory do modulu.

Bohužel základní funkce na instalaci modulu nejsou dostačující, protože přesouvají ze složky *override* pouze soubory .php, nicméně ve složce mohou být například i soubory .tpl (Smarty šablony pro vykreslení obsahu). Musel jsem tedy přidat instalační a odinstalační funkci, která složku *overrides* u modulu projde a všechny .tpl soubory překopíruje do cílové *override* složky (popřípadě je z ní smaže při odinstalaci). Důležité ale bylo, aby při kopírování zůstala zachována struktura adresářů, jinak by přetížení souborů nefungovalo.

Stejný problém byl u modulové složky *classes*, ve které jsou uloženy třídy, sloužící jako entity pro databázové objekty. Z této složky se načítají třídy pouze do jmenných prostorů náležících k modulu, ale pro zbytek PrestaShopu dostupné nejsou. Proto jsem musel přidat další funkci, která při instalaci soubory z této složky překopíruje do hlavní *classes* složky v kořeni projektu, stejně tak funkci, která je při odinstalaci opět smaže. O zavedení tříd do projektu se pak postará ClassLoader, který načítá všechny třídy z hlavní složky *classes*.

Dalším problémem bylo, že PrestaShop ignoruje některé návratové hodnoty při instalaci modulu. Pokud třeba nemá uživatel, pod kterým e-shop běží, práva na čtení složky s modulem, tak se sice z kopírovací funkce vrátí chyba, ale instalátor ji nebere v potaz. Tím pádem je zobrazena zpráva, že vše bylo úspěšně nainstalováno, i když se nic nepřekopírovalo.

Při implementaci modulu jsem se nejdříve zaměřil na pochopení toho, jak moduly fungují. K tomu mi sloužila vývojářská dokumentace a zároveň i moduly, které jsem v PrestaShopu již měl nainstalované a mohl jsem se u nich v některých věcech inspirovat.

Poté jsem procházel přemigrované tabulky a sloupce a vytvářel pro ně potřebné formuláře či přidával pole do již existujících formulářů. Největším problémem vždy bylo zjistit, která část aplikace se vytváří v které třídě, protože projekt obsahuje statisíce řádků kódu a i po přečtění dokumentace mi nebylo hned jasné, co kde hledat. Postupně jsem ale pochopil celou strukturu systému, a tak s každým dalším formulářem byla práce lehčí.

PrestaShop má pěkně řešené vytváření přehledových tabulek a formulářů, jak v legacy verzi, tak v modern verzi. Jde vždy pouze o vybrání správných

předem připravených komponent, které se pak samy naplní daty a vyrenderují. Při psaní některých složitějších komponent (jako je třeba formulář na přidávání variant produktů) je však potřeba data předzpracovat. U této komponenty jsem se ale mohl jednoduše inspirovat komponentou na přidávání souvisejících produktů, která funguje úplně stejně, pouze zpracovává data z jiné tabulky.

Bohužel jsem se při práci nemohl úplně spoléhat na dokumentaci, protože ne vždy odpovídala aktuální verzi PrestaShopu. Nejenom že byla místy zastaralá, někdy dokonce popisovala funkce, které ještě nebyly naimplementované (v době, kdy nejnovější vydaná verze PrestaShopu byla 1.7.4.4, tak některé části dokumentace popisovaly verzi 1.7.5). Ani oficiální PrestaShop forum nebylo úplně nápomocné, protože i když už se někdo potýkal se stejnými problémy jako já, tak buď k jeho otázce na foru nebyla žádná odpověď, nebo byla otázka i odpověď třeba z roku 2013. Od té doby se PrestShop velmi změnil, a tak řešení nebylo použitelné pro aktuální verzi.

Při implementaci modulu jsem si však nebyl jistý, jak nakládat se změnami struktury databáze. PrestaShop moduly využívají vlastních instalačních a upgrade postupů, v rámci kterých jsou SQL příkazy uloženy ve složce modulu a při instalaci, upgradu a odinstalaci se provádějí dle předem daných pravidel. Nicméně všechny Jagu produkty využívají software Phinx [28], který je velmi dobrým nástrojem na změny databázové struktury.

Phinx jsem využíval i já při práci na tomto projektu ještě předtím, než jsem byl rozhodnut, že budu psát modul. Musel bych tak buď kombinovat Phinx skripty a SQL skripty modulu, nebo všechny již hotové databázové změny přesunout do modulu. Ani jedno z řešení mi nepřipadalo úplně rozumné, první by bylo nelogické a nepřehledné, druhé by bylo těžké na udržitelnost, protože databázových změn (hlavně na začátku projektu) bylo velmi mnoho. Dalším důvodem bylo také to, že vývojáři starého projektu administrace, kteří budou pravděpodobně časem pracovat i na tomto projektu, jsou již na Phinx zvyklí. Proto jsem se rozhodl mít dále oddělené databázové skripty od modulu a využívám na ně Phinx.

5.4.3 Překlad modulu

PrestaShop má velmi pěkně vymyšlené překlady celého e-shopu – v kódu (jak v PHP třídách, tak v renderovacích šablonách) jsou speciální značky (ať jde o speciální zápis volání funkcí či makra v šablonách), díky kterým se určí texty k překladu. Ty se pak zobrazí v administraci, kde je možné daný text přeložit do všech jazyků, které jsou pro e-shop zpřístupněné. Jednotlivé texty se v administraci zobrazí podle jejich domény, ať se jedná o pojmenování sekcí, názvy polí formulářů, chybové hlášky nebo texty spojené s modulem.

I když je tento koncept opravdu pěkný, bohužel realita byla o něco horší. Nejenže systém nebyl vždy úplně intuitivní (například parser, který určoval, co se má přeložit, bral v potaz pouze texty v jednoduchých uvozovkách a ty v dvojitých zahazoval), ale texty v overridech ignoroval úplně. Což mi, vzhle-

dem k tomu, že většina mnou přidaných tříd mezi overridy patřila, práci úplně neulehčovalo. Musel jsem tedy přetížit samotný systém na překlady, abych vůbec byl schopný překlad modulu začít řešit.

Druhou vadou celého systému byl opět jeho postupný přerod z legacy do modern frameworku, protože v tuto chvíli se na překlady používaly dva různé způsoby, každý někde jinde a každý nějak jinak. Legacy framework sázel na již zmíněný parser, kde v rámci PHP tříd hledal volání funkce *l*, kde prvním parametrem je překládaný text a druhým (nepovinným) parametrem je třída, ke které text náleží. V Smarty šablonách se text určený k překladu obalí do speciálního bloku – *{l s='Add' d='jagu'}*, kde *s* je překládaný text a *d* je doména[16].

S přechodem na Symfony ještě do systému přibyla funkce *trans* třídy *Translator*, která se v PHP třídách volá *$this->translator->trans('Quantity', [],'Admin.Catalog.Feature')*, kde první parametr je překládaný text, druhý případné parametry překladu (množná čísla atd.) a třetí je opět doména. Ve Twig šablonách je pak syntaxe následující: *{{ 'Module to configure' | trans({},'Admin.Catalog.Feature') }}*.

Jak je vidět, tak oba způsoby jsou trochu jiné. Bohužel, na některých místech, jako jsou třeba Twig šablony v modulech, nefungoval ani jeden z nich. I když měl být systém nových překladů plně funkční již od verze PrestaShop 1.7 [29], o skoro dva roky později vše stále nefungovalo tak, jak mělo. Bylo tedy v tuto chvíli nemožné překládat texty v šablonách modulů třetích stran ani jedním způsobem, a tak jsem si musel vytvořit v modulu speciální metody. Ty překládaly požadované texty v PHP třídách starým způsobem za použití funkce *l*, a pak předávaly pole s překlady do šablony těsně před renderováním. Tento způsob sice neodpovídá původnímu návrhu překladů, jak byl zamýšlen vývojáři PrestaShopu, ale byl to funkční a udržitelný způsob, jak modul překládat.

[16]Doménou zde není myšlena internetová doména, ale oblast, které text náleží. Může jít třeba o *Admin.Actions*, tedy texty k akcím v administraci, či *Module.Jagu.General*, tedy obecné texty k modulu Jagu.

Kapitola 6

Shrnutí a budoucnost projektu

Začít pracovat na projektu, který je už nějakou dobu ve vývoji, nikdy není jednoduché. Trvá vždy nějaký čas, než se programátor sžije s novým prostředím, pochopí programátorské standardy, propojení jednotlivých komponent či důvody, proč jsou některé části kódu napsané tak, jak jsou. Často je také velmi důležité znát zákazníka a hlavně rozumět všem jeho požadavkům, jinak některé části projektu nemusí dávat smysl. Mohou být napsané pro splnění konkrétního usecase, který je pro firmu důležitý, i když pro někoho nezasvěceného se může zdát nepochopitelný.

V rámci diplomové práce jsem musel pochopit projekty dva – jak customizovaný OpenCart e-shop firmy Jagu s.r.o, tak e-shop postavený na platformě PrestaShop. Oba projekty obsahují statisíce řádků kódu a spoustu komponent, které bylo potřeba pochopit, abych mohl projekty dále rozvíjet.

U projektu customizovaných administrací byla pro mě největším problémem jeho komplexnost, protože projekt prochází vývojem již od roku 2009 a za tu dobu vývojáři stihli udělat opravdu mnoho změn. Tím se e-shop velmi odklonil jak od původní funkcionality, tak původní struktury. To mělo za následek různé nekonzistence mezi jednotlivými částmi kódu, stejně tak jako degradaci architektury. Zároveň byla v databázi spousta sloupečků a tabulek, které se nevyužívají, ale nikdo je nesmazal, což ubíralo na čitelnosti kódu a tím pádem celého projektu. Na druhou stranu jsem se vždy mohl s dotazy obrátit na vývojáře customizované administrace, kteří mi vždy byli schopni poradit, a tak jsem ani nepotřeboval dokumentaci.

Projekt PrestaShop měl také své neduhy, které byly z velké části způsobeny pomalým přepisem z jedné technologie na jinou. Proces přepisu byl v době psaní této práce přibližně v polovině, a tak spolu musel často komunikovat nový a starý framework, aby se dosáhlo kýženého výsledku. V případě problémů bohužel nebyla ani dokumentace moc nápomocná, protože ne vždy popisovala aktuální stav. Přesto je PrestaShop velmi slibnou platformou, protože nabízí spoustu funkcí a hlavně velkou modularitu. Až vývojáři dokončí přepisování do Symfony, bude PrestaShop bezpochyby nejlepší platformou pro

tvorbu e-shopů na trhu.

V rámci práce jsem se zaměřil na korektní migraci dat z customizovaného OpenCart e-shopu na PrestaShop. I přesto, že jsem jako základ využil modul na migrace, musel jsem na něm udělat mnoho změn, protože nebyl dostačující. Přidával jsem migraci OpenCart tabulek, které byly modulem ignorovány, a zároveň jej upravoval tak, aby byl schopný přemigrovat i tabulky a sloupečky, které byly do customizované OpenCart databáze přidány. Při tom jsem se nejvíce potýkal s nekompatibilitou databází co se struktury dat týče, stejně tak s nevaliditou migrovaných dat, která jsem musel kromě transformace také čistit.

Aktuálně je tedy modul schopný přemigrovat všechna data z původního e-shopu, aniž by ztratil jediný záznam. Stejně tak je možné migrační skript pustit opakovaně, a dokonce i na dalších obchodech se stejnou databázovou strukturou. Velkou silou skriptu je tedy jeho znovupoužitelnost. Zároveň umožňuje korektní transformaci dat, aby byla co nejvíce zachována původní struktura PrestaShop databáze, tedy aby data odpovídala požadovanému rozdělení do příslušných tabulek, ale také aby se korektně zpracovaly i přidané sloupečky a tabulky.

Nový systém pokrývá 65 % z požadavků definovaných v sekci 2.2, proto byla také platforma PrestaShop zvolena jako základ. Bohužel po skončení této diplomové práce zůstává toto číslo stejné – v rámci práce byl totiž čas pouze na migraci dat a některých funkcí z customizované administrace, k implementaci nových věcí jsem se nedostal. Dle zadání jsem se zaměřoval na funkce, které chyběly novému systému oproti starému. V prvním kroku vývoje nové administrace byla důležitá hlavně korektní a úplná migrace databáze.

Avšak migrace dat byla pouhým začátkem celého procesu migrace e-shopů. To hlavní, co dělá customizované e-shopy OpenCart pro Jagu s.r.o. tak zásadní, je již zmíněná funkcionalita. Tedy know-how firmy skryté v kódu, v uživatelském rozhraní, v ulehčení pracovního procesu a ve spoustě dalších věcí, na kterých firma v posledních devíti letech pracovala.

V rámci této práce jsem tedy naimplementoval modul, který zatím umožňuje alespoň přímou práci s daty v databázi. Velká část dat, kterou jsem přemigroval, je tedy upravitelná, či alespoň zobrazitelná v administraci. Přidával jsem pole na úpravu dat i tam, kde u starého e-shopu z nedostatku času nebyla. U některých migrovaných dat to však potřeba nebylo, šlo totiž o tabulky, se kterými pracuje aplikace v pozadí, ale uživatel k nim přístup nepotřebuje.

Ač PrestaShop ze základu pokrýval 65 % požadavků definovaných v sekci Funkce aplikace 2.2, tak funkcí, které chce Jagu s.r.o. mít ve výsledném e-shopu, je mnohem více. Velkou většinu z nich však nelze nalézt v žádné platformě, protože je specifická pro danou firmu a byla vyvinuta jejich vývojáři.

Jenom samotná migrace a psaní modulu mi zabraly stovky hodin práce. Přemigrování všech funkcí ze starého e-shopu do nového by jistě bylo na další stovky hodin, pokud by je přepisoval někdo, kdo systémům rozumí. Komuko-

liv jinému by jistě práce zabraly bezpochyby ještě více času, neboť funkcí je opravdu hodně a spousta z nich je velmi komplexních. Jsem si jistý, že dostat nový e-shop do stavu, který se dá považovat alespoň za prototyp finálního, by jistě mohlo být obsahem dalších několika diplomových prací.

Mezi funkce, které jsou na starém e-shopu, ale zatím nejsou na novém, patří:

- sledování doručované zásilky,
- tisk štítků na zásilky,
- feedy pro vyhledávače Heureka, Zboží a Google Nákupy,
- porovnávání cen,
- detailní přehledové grafy,
- exporty do csv,
- a spousta dalších.

Takto rozsáhlý projekt, jako jsou customizované e-shopy OpenCart, nebylo možné obsáhnout v jedné diplomové práci, aby výstup obsahoval vše potřebné a zároveň byl lehce spravovatelný. Přes to, že vetšina práce na projektu stále není hotova a zabere ještě spoustu času, než bude projekt připraven ke spuštění, věřím, že se mi podařilo vypracovat kvalitní základ platformy, na které se bude dobře stavět. Pokud bude dodržen postup vývoje nových funkcí navržený v této práci, tedy bude platforma rozšiřována za pomoci modulu, může projekt časem obsahovat veškeré potřebné funkce. Zároveň díky automatickým aktualizacím nezastará, jako se stalo u původního e-shopu.

Závěr

Cílem této diplomové práce bylo přemigrování customizovaných e-shopů OpenCart verze 1.1 na novou platformu.

Jedním z výsledků mé práce je detailní rozbor aktuálně nejpoužívanějších platforem, ze kterých jsem vybral PrestaShop. Ten splňoval požadavky zákazníka z 65 % a zároveň byl velmi modulární.

Dále jsem v rámci diplomové práce naimplementoval pro PrestaShop dva moduly. První z nich vychází z modulu MigrationPro a slouží k migraci customizovaných databází OpenCart na databáze PrestaShop e-shopů. Při jeho implementaci jsem se zaměřil na validaci a čištění dat, stejně tak jako jejich korektní transformaci pro správné uložení do cílové databáze.

Druhý z modulů slouží k úpravě PrestaShop administrace tak, aby co nejvíce splňovala potřeby zákazníka. Při jeho vývoji jsem se zaměřil na jednoduchost dalších úprav a na zachování možnosti automatických aktualizací PrestaShopu.

V budoucnu je potřeba některé části modulu postupně přepsat do Symfony frameworku, až do něj budou přepsané i původní části administrace. Stejně tak by bylo záhodno uzpůsobit překlady modulu podle nového PrestaShop standardu, ale i zde se čeká na potřebné úpravy ze strany vývojářů PrestaShopu.

Z analýzy aktuálního stavu původních e-shopů také vyplynuly nějaké požadavky zákazníka, jejich implementace ale byla nad rámec rozsahu této práce. I tam je tedy prostor pro další rozvoj.

Nejdůležitějším úkolem do budoucna je však migrace funkcí, které byly přidány do customizovaného OpenCartu firmy Jagu s.r.o., ale nejsou zatím naimplementované do nového e-shopu.

Literatura

[1] Nováček, T.: Apakrychle – Co je Apakrychle? *Apakrychle.ninja [online]*, 2018, [cit. 2018-1-12]. Dostupné z: http://apakrychle.ninja/

[2] Builtwith: OpenCart – podíl trhu. *builtwith.com [online]*, 2018, [cit. 2018-2-26]. Dostupné z: https://trends.builtwith.com/shop/OpenCart

[3] Datanyze: E-commerce – podíl na trhu. *Datanyze [online]*, 2018, [cit. 2018-2-26]. Dostupné z: https://www.datanyze.com/market-share/e-commerce-platforms

[4] Datanyze: E-commerce – podíl na českém trhu. *Datanyze [online]*, 2018, [cit. 2018-2-26]. Dostupné z: https://www.datanyze.com/market-share/e-commerce-platforms/Czech%20Republic/

[5] Wordpress: WooCommerce. *WordPress [online]*, 2018, [cit. 2018-4-3]. Dostupné z: https://cs.wordpress.org/plugins/woocommerce/

[6] Codecanyon: WooCommerce – Category tree plugin. *Envato Market [online]*, 2018, [cit. 2018-4-3]. Dostupné z: https://codecanyon.net/item/woocommerce-admin-category-tree-/10876618

[7] Shopify: Categories by BTP – Shopify. *App Store [online]*, 2018, [cit. 2018-4-3]. Dostupné z: https://apps.shopify.com/categories-by-btp

[8] Shopify: Using the image editor – Shopify. *Shopify help center [online]*, 2018, [cit. 2018-5-3]. Dostupné z: https://help.shopify.com/manual/productivity-tools/image-editor

[9] CMS Critic: How Useful is Squarespace as an Analytics Tool? *CMS Critic [online]*, 2018, [cit. 2018-5-3]. Dostupné z: https://www.cmscritic.com/how-useful-is-squarespace-as-an-analytics-tool/

[10] Squarespace: Creating a custom checkout form. *Squarespace support [online]*, 2018, [cit. 2018-5-3]. Dostupné z: https://support.squarespace.com/hc/en-us/articles/206540907-Creating-a-custom-checkout-form

[11] Peníze.cz: Jagu s.r.o – výpis z obchodního rejstříku. *Peníze.cz [online]*, červenec 2008, [cit. 2018-2-26]. Dostupné z: http://rejstrik.penize.cz/28426126-jagu-s-r-o

[12] Jagu s.r.o.: Jagu s.r.o. *jagu.cz [online]*, 2017, [cit. 2018-2-26]. Dostupné z: http://jagu.cz/

[13] OpenCart: OpenCart. *OpenCart.com [online]*, 2018, [cit. 2018-2-26]. Dostupné z: https://www.opencart.com/

[14] Wikipedia: OpenCart – Wikipedia. *Wikipedia [online]*, 2018, [cit. 2018-2-26]. Dostupné z: https://en.wikipedia.org/wiki/OpenCart

[15] Adaptic: E-commerce – definice. *Adaptic [online]*, 2018, [cit. 2018-2-26]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/e-commerce/

[16] Wikipedia: Eltrontitle – Wikipedia. *Wikipedia [online]*, 2018, [cit. 2018-5-3]. Dostupné z: https://en.wikipedia.org/wiki/Eltron_Programming_Language

[17] Wikipedia: CMS – Wikipedia. *Wikipedia [online]*, 2018, [cit. 2018-4-3]. Dostupné z: https://cs.wikipedia.org/wiki/Syst%C3%A9m_pro_spr%C3%A1vu_obsahu

[18] W3Techs: W3Techs – Web Technology Survey. *W3Techs [online]*, 2018, [cit. 2018-4-3]. Dostupné z: https://w3techs.com/technologies/history_overview/content_management

[19] Shopify: Shopify – Pricing. *Shopify [online]*, 2018, [cit. 2018-4-3]. Dostupné z: https://www.shopify.com/pricing

[20] Wikipedia: Konverze – Wikipedia. *Wikipedia [online]*, 2018, [cit. 2018-4-3]. Dostupné z: https://cs.wikipedia.org/wiki/Konverze_(marketing)

[21] Squarespace: Squarespace – Pricing. *Squarespace [online]*, 2018, [cit. 2018-4-3]. Dostupné z: https://www.squarespace.com/pricing/

[22] Microsoft: Windows – oficiální web. *Microsoft.com [online]*, 2018, [cit. 2018-5-12]. Dostupné z: https://www.microsoft.com/cs-cz/windows

[23] PrestaShop: Module development – PrestaShop DevDocs. *PrestaShop DevDocs [online]*, 2018, [cit. 2018-16-11]. Dostupné z: https://devdocs.prestashop.com/1.7/modules/

[24] PrestaShop: Keeping up-to-date – PrestaShop DevDocs. *PrestaShop DevDocs [online]*, 2018, [cit. 2018-21-9]. Dostupné z: https://devdocs.prestashop.com/1.7/basics/keeping-up-to-date/

[25] PrestaShop: 1-Click Upgrade Module. *PrestaShop Official Addons Marketplace [online]*, 2018, [cit. 2018-13-12]. Dostupné z: https://addons.prestashop.com/en/data-migration-backup/5496-.html

[26] PrestaShop: Keeping up-to-date – PrestaShop DevDocs. *PrestaShop DevDocs [online]*, 2018, [cit. 2018-13-12]. Dostupné z: https://devdocs.prestashop.com/1.7/basics/keeping-up-to-date/migration/#other-migration-options-for-developers

[27] Symfony: Symfony profiler – Symfony documentation. *Symfony documentation [online]*, 2018, [cit. 2018-19-11]. Dostupné z: https://symfony.com/doc/3.4/profiler.html

[28] Phinx: Phinx. *Phinx [online]*, 2018, [cit. 2018-30-11]. Dostupné z: https://phinx.org/

[29] PrestaShop: New translation system – PrestaShop 1.7. *PrestaShop Build [online]*, 2018, [cit. 2018-15-11]. Dostupné z: http://build.prestashop.com/news/new-translation-system-prestashop-17/

Příloha **A**

Seznam použitých zkratek

PHP Hypertext Preprocesor

EPL Eltron Programming Language

FTP File Transfer Protocol

EET Elektronická evidence tržeb

SEO Search Engine Optimization

URL Uniform Resource Locator

REST Representational state transfer

API Application Programming Interface

CMS Content Management System

SQL Structured Query Language

IDE Integrated Development Environment

ETL Extract-transform-load

CSV Comma separated values

Příloha B

Kompletní tabulka srovnání e-shopů

Pro přehlednost používám zkratky OC – OpenCart, WC – WooCommerce, SS – Squarespace, PS – PrestaShop. Zkratka R u požadavků pak znamená, že jde o bezplatné rozšíření. U verzí PHP je vždy myšlena daná verze nebo vyšší.

	OC	**WC**	**Shopify**	**SS**	**PS**	**Magento**
Jazyk	PHP 5.4	PHP 5.6	Rails	Java	PHP 5.4	PHP 5.6
Open-source	Ano	Ano	Ne	Ne	Ano	Ano
Migrace z OC	Ano	Ano	Ano	Ne	Ano	Ano
Multisite	Ano	Ano	Ne	Ano	Ano	Ano
REST API	Ne	Ano	Ano	Ano	Ano	Ano
PO1 2.2.1	Ne	Ne	Ne	Ne	Ne	Ne
PO2 2.2.1	Ne	Ano (R)	Ne	Ne	Ne	Ne
PO3 2.2.1	Ano	Ano (R)	Ne	Ne	Ano	Ano
PO4 2.2.2	Ne	Ano	Ne	Ne	Ne	Ne
PO5 2.2.2	Ne	Ne	Ne	Ne	Ano	Ano
PO6 2.2.2	Ne	Ano	Ano	Ne	Ano	Ne
PO7 2.2.2	Ne	Ne	Ne	Ne	Ano	Ne
PO8 2.2.2.1	Ne	Ano	Ano	Ano	Ano	Ano
PO9 2.2.2.1	Ne	Ne	Ne	Ne	Ano	Ne
PO10 2.2.2.1	Ne	Ano	Ano	Ne	Ano	Ano
PO11 2.2.2.1	Ne	Ne	Ne	Ne	Ano	Ne
PO12 2.2.2.1	Ne	Ne	Ne	Ne	Ne	Ne
PO13 2.2.2.1	Ne	Ne	Ne	Ne	Ne	Ne
PO14 2.2.3	Ne	Ne	Ne	Ne	Ne	Ne
PO15 2.2.3	Ne	Ne	Ne	Ne	Ano	Ne
PO16 2.2.3	Ano	Ano	Ano	Ano	Ano	Ano

	OC	**WC**	**Shopify**	**SS**	**PS**	**Magento**
PO17 2.2.3	Ne	Ne	Ne	Ne	Ne	Ano
PO18 2.2.3	Ne	Ne	Ne	Ne	Ne	Ano
PO19 2.2.3	Ne	Ne	Ne	Ne	Ne	Ne
PO20 2.2.3	Ano	Ano	Ano	Ano	Ano	Ano
PO21 2.2.3	Ne	Ne	Ne	Ne	Ne	Ano
PO22 2.2.3.1	Ne	Ne	Ne	Ano	Ano	Ano
PO23 2.2.3.1	Ano	Ano	Ne	Ano	Ano	Ano
PO24 2.2.3.1	Ne	Ano	Ano	Ne	Ne	Ano
PO25 2.2.3.1	Ne	Ano	Ano	Ne	Ano	Ne
PO26 2.2.3.1	Ne	Ne	Ne	Ne	Ano	Ne
PO27 2.2.3.1	Ne	Ano	Ne	Ne	Ano	Ne
PO28 2.2.3.1	Ne	Ne	Ano	Ne	Ne	Ano
PO29 2.2.4	Ano	Ano	Ano	Ne	Ne	Ano
PO30 2.2.4	Ano	Ano	Ano	Ne	Ano (R)	Ano
PO31 2.2.4	Ano	Ano	Ano	Ano	Ano	Ano
PO32 2.2.5	Ne	Ne	Ne	Ne	Ne	Ano
PO33 2.2.5	Ne	Ne	Ne	Ne	Ano	Ne
PO34 2.2.5	Ne	Ne	Ne	Ne	Ano	Ne
PO35 2.2.6	Ne	Ano	Ano	Ano	Ano	Ano
PO36 2.2.6	Ne	Ano	Ne	Ano	Ano	Ano
PO37 2.2.6	Ne	Ne	Ano	Ano	Ano	Ano
PO38 2.2.6	Ne	Ne	Ne	Ne	Ano	Ano
PO39 2.2.6	Ne	Ano	Ano	Ano	Ano	Ne
PO40 2.2.6	Ne	Ano	Ano	Ne	Ano	Ano
PO41 2.2.6	Ne	Ne	Ne	Ne	Ne	Ne
PO42 2.2.6	Ne	Ano	Ano	Ne	Ano	Ne
PO43 2.2.6	Ne	Ano	Ano	Ano	Ano	Ano
PO44 2.2.6	Ano	Ano	Ano	Ano	Ano	Ano
PO45 2.2.6	Ano	Ano	Ano	Ano	Ano	Ano
PO46 2.2.6	Ne	Ano	Ano	Ano	Ano	Ano
# MUST (z 30)	3	13	11	9	22	17
% MUST	10 %	43 %	37 %	30 %	73 %	57 %
# NICE (z 11)	3	6	4	0	3	4
% NICE	27 %	55 %	36 %	0 %	27 %	36 %
# DEL (z 5)	3	5	5	5	5	5
% DEL	60 %	100 %	100 %	100 %	100 %	100 %
# celkem (max 46)	9	24	20	14	30	26
% celkem	20 %	52 %	43 %	30 %	65 %	56 %

Tabulka B.1: Kompletní tabulka srovnání administrací e-shopů

Příloha C

Obsah přiloženého DVD

- `readme.txt` ..... stručný popis obsahu DVD
- `zdrojove-soubory`
  - `aplikace` ..... zdrojové kódy implementace
  - `text` ..... zdrojová forma práce ve formátu LaTeX
- `prace.pdf` ..... text práce ve formátu PDF