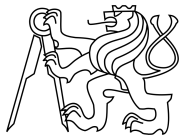
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Elektronický obchod pro framework Laravel
Student:	Tomáš Novotný
Vedoucí:	Ing. Josef Gattermayer
Studijní program:	Informatika
Studijní obor:	Web a multimédia
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Cílem práce je vyvinout balíček (rozšíření) pro framework Laravel, který přidá funkcionalitu jednoduchého e-shopu.

Pokyny:
1) Seznamte se s frameworkem Laravel.
2) Proveďte analýzu existujících balíčků e-shopu ve frameworku Laravel.
3) Proveďte analýzu existujících balíčků e-shopu v ostatních frameworcích.
4) Navrhněte funkcionalitu balíčku tak, aby splňoval požadavky pro klasický elektronický obchod:
 - vytvoření a správa objednávky,
 - správa produktů v nákupním košíku,
 - zadání dodacích a fakturačních údajů,
 - zadání způsobu platby a dopravy,
 - možnost platby přes zadanou platební bránu,
 - vytvoření jednoduché faktury,
 - export a import dat z účetního systému.
5) Balíček implementujte.
6) Balíček otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 9. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Elektronický obchod pro framework Laravel

Tomáš Novotný

Vedoucí práce: Ing. Josef Gattermayer

16. května 2016

Poděkování

Děkuji Ing. Josefu Gattermayerovi za cenné rady a připomínky. Dále děkuji všem, kteří se mnou měli trpělivost při tvorbě této práce.

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.

V Praze dne 16. května 2016 .....

Odkaz na tuto práci

Novotný, Tomáš. *Elektronický obchod pro framework Laravel*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Cílem této práce je vytvoření implementace balíčku pro PHP framework Laravel, který do aplikace přidá základní funkčnost klasického internetového obchodu.

Klíčová slova Laravel, PHP, ecommerce, elektronický obchod, eshop, webová aplikace, modul

Abstract

The goal of this work is to maintain a basic implementation of package for PHP framework Laravel, which will add functionality of clasic online shop into the application.

Keywords Laravel, PHP, ecommerce, online shopping, eshop, web application, package

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Nakupování zboží pomocí internetového obchodu (tzv. eshop), oproti nákupu v kamenné prodejně, přináší jak pro zákazníka, tak pro prodejce, mnoho výhod. Hlavní výhody jsou široká působnost a v podstatě neomezená otevírací doba oproti kamenné prodejně. Zákazník tak může, pouze s elektronickým zařízením a připojejím k internetu, nakupovat kdekoliv a kdykoliv. V neposlední řadě patří mezi výhody eshopů také nižší cena a širší nabídka zboží, než v kamenných prodejnách, jelikož není nutné všechno zboží vystavovat v prostorech prodejny, často s vysokým nájmem.

Trend zakládání nových, menších eshopů je v ČR stále velmi silný. V roce 2015 existovalo v ČR zhruba 37 000 eshopů [1]. Pro založení eshopu nejsou v ČR nutná žádná zvláštní povolení a podle zákona spadá toto podnikání do volných živností – obor číslo 48 Velkoobchod a maloobchod [2]. Internet je ovšem velmi velký trh, kde stále přicházejí noví hráči s novými technologiemi, postupy a trendy. Proto je nutné stále investovat do nových informačních technologií za účelem konkurenceschopnosti. Podnikatelé, kteří se rozhodnou své produkty či služby poskytovat na internetu prostřednictvím webové aplikace, si musí vybrat v jakém rozsahu chtějí podnikat. Můžou si totiž vybrat hned z několika způsobů jak svoje řešení implementovat.

Pro velmi omezené, ale bezplatné řešení, existuje mnoho možností jak aplikaci vytvořit pomocí tzv. free eshopů (např. PrestaShop[1]). Jedná se o hotový, volně šiřitelné software, tudíž jsou pořizovací náklady minimální. Aplikace ovšem nemusí vždy vypadat a fungovat podle představ majitele eshopu a jakékoliv úpravy a změny jsou jen velmi obtížně proveditelné a dlouhodobě neudržitelné.

Druhá možnost je zakoupení licence nějaké větší, robustnější, hotové aplikace (např. OxyShop[2]). Ty naopak podporují velké množství různých služeb, které jsou buď přímo součástí licence, či je lze jednoduše dokoupit a doinstalo-

[1] https://www.prestashop.com/
[2] https://www.oxyshop.cz/

vat. Nevýhodou stále zůstává jen velmi omezená možnost změn podle osobních požadavků majitele eshopu.

Nejlepší, ale zároveň často nejdražší řešení, je vytvoření vlastní webové aplikace na míru profesionální firmou. Tato aplikace je vytvořená přesně podle zadaných požadavků, a při správném návrhu aplikace je dlouhodobě udržitelná a upravitelná. Proto se jedná o nejvýhodnější řešení co se týče dlouhodobého vývoje eshopu a možnosti reakce na moderní trendy v nakupování na internetu.

Definice cílů

Cílem práce je navrhout a implementovat stažitelný balíček (rozšíření) pro PHP framework Laravel, který přidá funkcionalitu jednoduchého eshopu za použití moderních technologií. Tento balíček by měl velmi zjednodušit a urychlit tvorbu vlastní aplikace internetového obchodu a jeho následnou údržbu, jelikož vývojář aplikace nebude muset znovu navrhovat a implementovat části, které jsou v základu pro všechna eshopová řešení shodná, či velmi podobná, a bude se tak moci soustředit na ostatní požadavky.

Tato práce nejdříve popisuje použité technologie, a to zejména s framework Laravel. Následuje analýza existujících balíčků, které se snaží vyřešit stejnou problematiku, a to jak přímo vytvořené pro použití ve frameworku Laravel, tak i v ostatních frameworcích. Před vlastní implementací balíčku práce popisuje návrh výsledné funkcionality, tak aby splňovala požadavky pro klasický elektronický obchod dle zadání:

- Vytvoření a úprava objednávky
- Správa produktů v nákupním košíku
- Zadání dodacích a fakturačních údajů
- Zadání způsobu platby a dopravy
- Možnost platby přes zadanou platební bránu
- Vytvoření jednoduché faktury
- Možnost exportu a importu z účetního systému

Další část práce se zabývá samotnou implementací balíčku a poslední část vytvoření automatických jednotkových testů, pro zajištění stability aplikace.

Kapitola 1

Představení frameworku Laravel

1.1 Co je to framework

V této práci, dokonce i v samotném názvu práce, je velmi často používáno slovo framework. Framework lze popsat jako sada nástrojů, knihoven, konvencí a osvedčených postupů, které vytváří abstrakci nad rutinními úkoly do obecných modulů, které mohou být lehce znovu využity [3].

1.2 Základní informace

Laravel je PHP[3] framework pro webové aplikace. Autorem frameworku Laravel je sofwarový inženýr Taylor Otwell z amerického Arkansasu [4]. Od roku 2011, kdy byla publikována první demoverze, do konce roku 2014 byl vývoj frameworku Laravel pro Otwella jeho vedlejší činností. Od ledna 2015 se již věnuje dalšímu vývoji frameworku na plný úvazek [5].

Laravel se snaží především o čistý a výstižný kód s přehlednou vnitřní strukturou. To umožňuje i začínajícím programátorům, kteří nikdy předtím žádný framework nepoužívali, pustit se ihned do vývoje vlastní webové aplikace. Zároveň však umožnuje vytvořit vlastní složité návrhové vzory zkušenějšími uživateli pro rozsáhlé robustní projekty. Tuto snahu popisuje i oficiální text, kterým se framework Laravel prezentuje [6].

Laravel je webový aplikační framework s výstižnou a elegantní syntaxí. Věříme, že vývoj aplikací může být zábavná a tvůrčí činnost, která vývojáře opravdu baví. Laravel se snaží obtížnosti u vývoje ulehčit zjednodušením běžných funkčností většiny webových projektů, jako je např. autentizace, routování, práce se session nebo kešování.

Cílem frameworku Laravel je, aby byl proces vývoje aplikací příjemný pro vývojáře, aniž by to negativně ovlivnilo jejich funkčnost. Šťastní vývojáři tvoří ten nejlepší kód. Pro tento účel jsme se snažili zkombinovat to nejlepší, co jsme

[3]PHP: Hypertext Preprocessor

se naučili u ostatních frameworků, a to bez ohledu na programovací jazyk, jako např. Ruby on Rails, ASP.NET MVC a Sinatra.

Laravel je dostupný, ale výkonný framework, poskytuje nástroje pro velké robustní aplikace. Vynikající IoC container, přehledný migrační systém a pevně intergrovaná podpora pro jednotkové testování jsou nástroje, které potřebujete pro vývoj jakékoliv aplikace, kterou vytváříte.

Framework Laravel získává stále větší popularitu mezi PHP vývojáři po celém světě. V žebříčku oblíbenosti PHP frameworků se obvykle umísťuje na předních pozicích, a to jak v roce 2014 [7] tak v roce 2015 [8]. Jeho popularitu také potvrzují čísla počtu instalací PHP frameworků ze serveru Packagist[4], jak je znázorněno v tabulce 1.1.

Tabulka 1.1: Počet instalací PHP frameworků ze serveru Packagist

název	posledních 30 dní	celkově
Laravel	697 991	10 472 674
Symfony	530 073	11 496 828
Yii 2	88 446	1 350 814
CakePHP	54 105	725 553
Nette	17 972	490 214
CodeIgniter	3 703	19 999

1.3 Historie

Robustní framework, který je dobře otestovaný a připravený pro vývoj aplikací pro produkční nasazení, vyžaduje značnou dobu pro svůj vývoj. Oproti svým konkurentům je Laravel poměrně mladý framework. Jeho vývoj postupoval odlišněji než u ostatních zavedených frameworků a v poněkud rychlejším tempu, než je obvyklé (viz tabulka 1.2).

V srpnu v roce 2009 byla vydána nová verze PHP 5.3, která mimo jiné přinesla podporu pro jmenné prostory (*namespaces*) či možnost volání anonymních funkcí, tzv. uzávěry (*closures*). Tyto nové funkce umožnily vývojářům psát více a lépe objektově orientované PHP (OOP[5] PHP) aplikace. Ačkoliv nová verze PHP poskytovala mnoho výhod, ne všechny zavedené frameworky se na ně zaměřily. Místo toho se soustředily na podporu starších verzí PHP. V této době se na seznamu frameworků nejvíce vyskytovaly Symfony, Zend, Slim micro framework, Kohana, Lithium a CodeIgniter. Z nich byl pravděpodobně nejvíce známý PHP framework CodeIgniter. Vývojáři ho upřednostňovali pro jeho obsáhlou dokumentaci a jednoduchost. Kterýkoliv PHP programátor s ním mohl velmi rychle začít vyvíjet aplikaci. Framework

[4] Uložiště pro PHP balíčky – https://packagist.org

[5] Object-oriented programming

měl také okolo sebe velkou komunitu vývojářů a velmi dobrou podporu od svých tvůrců.

V roce 2011 ovšem ve frameworku CogeIgniter sházely funkčnosti, které Taylor Otwell považoval za základní při budování webové aplikace, jako například zabudovaná autentizace, či routování pomocí uzávěrů (*closure routing*). A tak byla 9. června 2011 vydána první beta verze frameworku Laravel 1. Podle svého autora byl tento projekt vytvořen pouze proto, aby vyřešil rostoucí obtížnost v používání frameworku CogeIgniter [9].

Tabulka 1.2: Historie frameworku Laravel

verze	datum vydání
Laravel 1.0	20. června 2011
Laravel 2.0	24. listopadu 2011
Laravel 3.0	22. února 2012
Laravel 4.0	28. května 2013
Laravel 4.1	12. prosince 2013
Laravel 4.2	1. června 2014
Laravel 5.0	4. února 2015
Laravel 5.1	9. června 2015
Laravel 5.2	21. prosince 2015

Laravel 1 Hned ve svém prvním vydání v červnu 2011 obsahoval Laravel značné množství funkcí – vlastní zabudovanou autentizaci, lokalizaci, modely a jejich vzájemné vztahy, Eloquent ORM[6], jednoduché routovací mechanismus, práce se *session*, kešování, pohledy (*views*), rozšiřitelnost pomocí modulů a knihoven a další. Na první verzi se tak jednalo o velmi dobrou funkcionalitu, ale nedostupnost řadičů (*controllers*)[7] zabránila aby se stal plnohodnotným MVC[8] frameworkem (architektura MVC je popsána v kapitole 1.8.1). V následujících měsících byla přidána validační funkce, stránkování a podpora instalace komponent třetích stran (tzv. *packages*) pomocí příkazové řádky. Laravel přešel ze své verze 1 do verze 2 během necelých šesti měsíců.

Laravel 2 Druhá hlavní verze frameworku, vydaná 24. listopadu 2011, znamenala značné vylepšení od svého autora a od komunity vývojářů. Mezi hlavní vylepšení patřila implementace podpory řadičů, vlastní šablonovací systém Blade, IoC[9] kontejner. Naopak krokem zpět bylo odstranění podpory pro

[6] Objektově relační zobrazení, je programovací technika v softwarovém inženýrství, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem

[7] Jedna z komponent prezentační vrstvy architektury MVC

[8] Model-View-Controller

[9] Inversion of Control – neboli obrácené řízení, je návrhový vzor, který umožňuje uvolnit vztahy mezi jinak těsně svázanými komponentami

správu kompenent třetích stran. Přidáním řadičů se Laravel stal plnohodnotným MVC frameworkem. Během dalších dvou měsíců byla vydána další hlavní verze, Laravel 3.

Laravel 3 V pořadí třetí hlavní verze byla vydána v rekordním čase po předchozí verzi, a to 22. února 2012. Tato verze frameworku byla zameřená na intergraci jednotkových testů, rozhraní příkazové řádky Artisan CLI[10], migrací do databáze, jakožto možnost verzování pro databázovou vrstvu a hlavně obnovení podpory správy kompenent třetích stran nazvaných Bundles.

Laravel 3 také začal dotahovat ostatní velké PHP frameworky jako např. CodeIgniter a Kohana. Mnoho vývojářů tak začalo přecházet z ostatních frameworků na Laravel pro jeho výkonnost a výstižnou elegantní syntaxi. Laravel se začal dostávat do širšího povědomí vývojářů a začal se objevovat v recenzích, článcích, tutoriálech či postupech, jako novinka ve světě PHP.

Laravel 3 zůstal ve stabilním vydání po určitou dobu a s dostatečnou výkonností, aby byl použit pro řadu různých druhů aplikací, nabízející jednoduchost a extrémně krátkou křivku učení oproti ostatním frameworkům. Ale po zhruba pěti měsících se jeho autor rozhodl celý framework přepsat od začátku jako systém balíčků, distribuovaný pomocí manažeru závislostí Composer[11].

Laravel 4 Laravel 4 s krycím jménem Illuminate byl vydán 28. května 2013. Framework byl celý přepsán zcela od začátku jako soubor jednotlivých balíčků, které spolu interagují a tvoří tak výsledný framework. Správa těchto balíčků byla zajištěna pomocí nejlepšího dostupného PHP manažeru závislostí Composer. Tento způsob umožňoval lepší rozšiřitelnost o budoucí změny a úpravy. Narozdíl od svých předchozích verzí měl Laravel 4 pravidelně naplánované vydávání nových menších verzí (každých šest měsíců) a časté opravy chyb (kdykoliv bylo možné). S více jednotkovými testy, které pokrývaly 100 % funkcionalit frameworku, tak Laravel 4 sliboval stabilní a jednoduše rozšiřitelnou aplikaci.

Laravel 4 byl od dosavadních verzí vylepšen o nové funkcionality, které žádná dosavadní verze Laravelu, ani žádný jiný framework, neměl. Například způsob jak naplnit databázi potřebnými či testovacími daty (tvz. *seeding*), zabudované odesílání emailů pomocí různých knihovem podle vlastního nastavení, značně vylepšený Eloquent ORM, archivování smazaných záznamů v databázi (tvz. *soft deletes*) a mnohé další různé změny a vylepšení.

Laravel 5 Laravel 5 byl vydán 4. února 2015 z původně plánované verze 4.3. Tato verze přinesla zcela novou vnitřní souborovou strukturu pro vyvíjené aplikace. Společně s Laravel 5 byla vydána komponenta Socialite, která poskytovala zjednodušenou autentizaci pomocí jiných služeb (např. Facebook, či

[10] Command Line Interface

[11] Nástroj pro správu závislostí v jazyku PHP – http://getcomposer.org

Google+), či komponenta s názvem Scheduler, umožnující opakovaně spouštět naplánované úkoly. Mezi další nové funkcionality patří abstraktní vrstva Flysystem, která sloužila k využití vzdáleného souborového uložiště, ale ovládala se stejně jako lokální souborový systém. Dále pak Elixir – jednoduchý způsob pro správu, kompilování či spouštení testů souborů Less, Sass či CoffeScript pomocí nástroje Gulp.

Laravel 5.1, vydaný v červnu 2015, byl první verzí frameworku Laravel s dlouhodobnou podporou LTS[12] s naplánovanou opravou chyb po dva roky, a opravou bezpečnostních děr po tři roky. Také bylo rozhodnuto vydávat dlouhodobě podporované verze Laravelu každé dva roky.

Aktuální stabilní vydání frameworku Laravel je 5.2.29 (ke dni 18. dubna 2016).

1.4 Dokumentace

Oficiální webové stránky dokumentace pro framework Laravel se nacházejí na adrese https://laravel.com/docs.

1.4.1 Laracast

Laravel má take vlastní vzdělávací videoportál Laracast [10]. Nachazí se zde video návody k frameworku Laravel a k programovaní v jazyce PHP obecně. Některé návody jsou uveřejněny zdarma, ostatní jsou pak k dispozici za měsíční poplatek. Autorem webu a také hlavní autor většiny příspěvků je lektor Jeffrey Way.

1.5 Licence

Laravel je *open source*[13] PHP framework pro webové aplikace distibuovaný pod licencí MIT, která umožnuje volně využívat software, s jedinou podmínkou – zahrnutí textu licence do všech kopií softwaru [11].

1.6 Styly pro psaní kódu

Laravel dodržuje standardy PSR–0 [12], PSR–1 [13] od verze Laravel 4 a standard PSR–2 [14] od verze Laravel 5.1.

[12] Long-term support

[13] Svobodný software, který zaručuje uživatelům svobodu jej spouštět, kopírovat, distribuovat, studovat, měnit a zlepšovat

1.7 Instalace

1.7.1 Minimální požadavky

Framework Laravel ve verzi 5.2 pro svůj provoz vyžaduje PHP verze 5.5.9 či vyšší a naistalovaných několik PHP rozšíření:

- OpenSSL
- PDO
- Mbstring
- Tokenizer

1.7.2 Instalace frameworku

Framework Laravel využívá nástroj Composer pro správu svých závislostí – je tedy nutné mít ho před vlastní instalací frameworku naistalován. Poté existuje několik způsobů jak framework naistalovat.

1.7.2.1 Laravel instalátor

Nejdříve je nutné stáhnout instalátor pomocí příkazu

```
composer global require "laravel/installer"
```

Poté je možné vytvořit novou instalaci frameworku Laravel do uživatelem zadané složky příkazem

```
laravel new
```

1.7.2.2 Composer create-project

Další možnost jak framework naistalovat je přímo pomocí nástroje Composer příkazem

```
composer create-project --prefer-dist laravel/laravel
```

1.7.3 Konfigurace

Po instalaci frameworku není vyžadována prakticky žádná konfigurace. Pouze je nutné nastavit oprávnění pro některé potřebné složky. Podsložky ve `storage` a složka `bootstrap/cache` musít mít povoleny zápis. Také je vhodné vygenerovat klíč aplikace pro lepší zabezpečení šifrovaných dat. Pokud byla aplikace naistalována pomocí jedné z doporučených možností instalace uvedených v kapitole 1.7.2, byl tento klíč již vygenerován. Jinak je ho potřeba nastavit pomocí příkazu `php artisan key:generate`.

Nastavení různých částí aplikace je možné upravit v jednotlivých konfiguračních souborech, které jsou uloženy ve složce `config`. Nová instalace Laravel obsahuje předvyplněnou konfiguraci, kde je většinu možností předdefinována tak, aby bylo možné začít s vývojem aplikace s co nejmenším počátečním nastavením. Tento přístup je znám jako konvence nad konfigurací (*convention over configuration*).

1.7.3.1 Konvence nad konfigurací

Konvence nad konfigurací je softwarové návrhové paradigma, které se snaží snížit počet rozhodnutí a nastavení, které programátor musí provést, tím že předpokladá dodržení určitých konvencí [15]. Programátor musí určit nastavení jen pro ty části aplikace, které se od těchto konvencí liší. Tím je získána snadnější použitelnost aplikace, aniž by ztratila na flexibilitě.

1.8 Architektura

1.8.1 Model–View–Controller

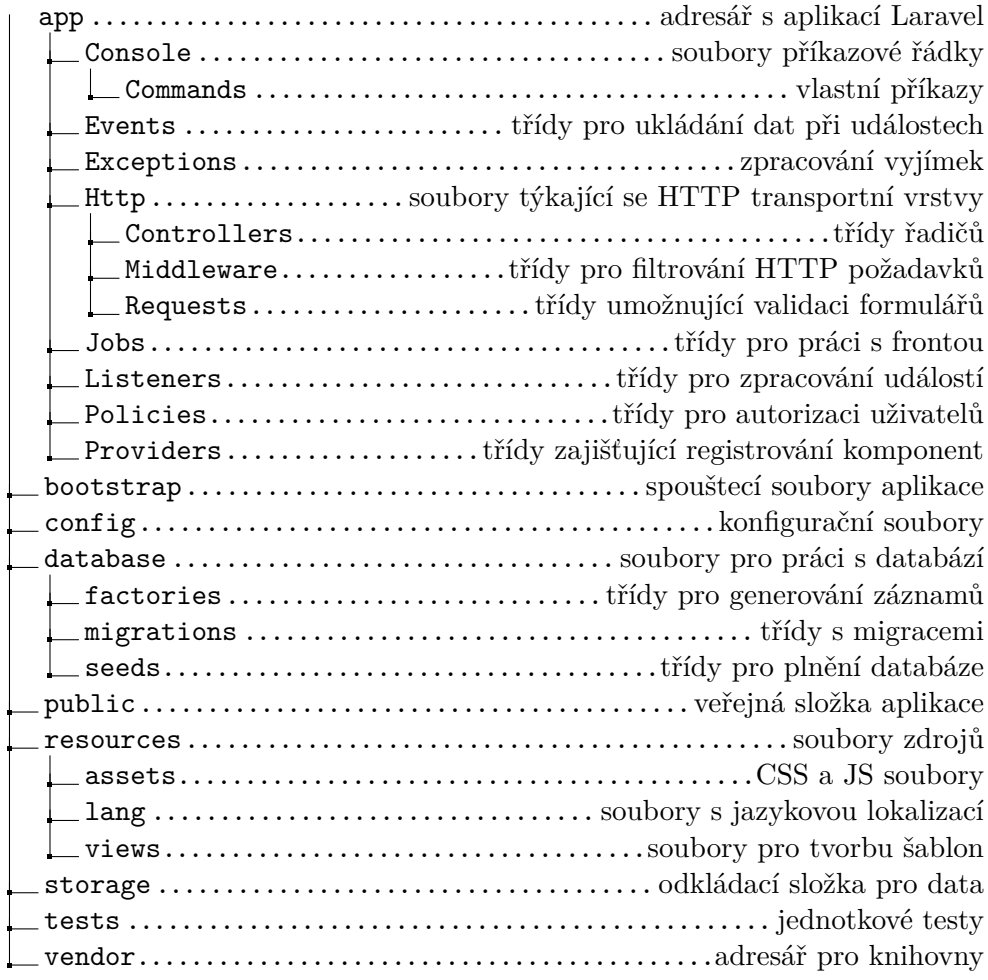
Laravel využívá návrhový vzor[14] *Model–View–Controller* (MVC). Tento vzor představil Tryfve Reenskaug již v roce 1979 [16]. MVC rozděluje kód aplikace do tří nezávislých částí, tak aby modifikace některé z nich, měla jen minimální vliv na ostatní. Toto rozdělení zpřehledňuje aplikaci, a lze ji v budoucnu jednodušeji vyvíjet a upravovat. Jednotlivé části lze také testovat zvlášť [17].

- **Model** – doménově specifická reprezentace infromací, s nimiž aplikace pracuje.
- **View (pohled)** – převádí data reprezentovaná modelem do podoby vhodné k interaktivní prezentaci uživateli, obvykle s využitím šablonovacího systému.
- **Controller (řadič)** – reaguje na události a zpracovává požadavky, typicky pocházející od uživatele, a zajišťuje změny v modelu nebo v pohledu.

1.8.2 Adresářová struktura

Laravel má od své verze 5.0 novou adresářovou strukturu. Tato struktura poskytuje lepší základ pro vývoj robustních aplikací a podporuje standard automatického načítání souborů PSR–4 [18]. Výchozí adresářová struktura frameworku Laravel po instalaci je zobrazena na obrázku 1.8.2.

[14] Někdy je MVC pro svou povahu také označováno jako architektonický vzor



Obrázek 1.1: Adresářová struktura frameworku Laravel

Kapitola 2

Základní komponenty frameworku Laravel

V této kapitole jsou popsány hlavní komponenty frameworku Laravel, zejména pak ty, které budou použity v implementaci balíčku v kapitole 5. Framework má mnoho dalších funkcionalit, které ovšem nejsou pro samotnou implementaci potřebné. Zbytek komponent lze nalézt v dokumentaci frameworku Laravel [19].

2.1 Artisan CLI

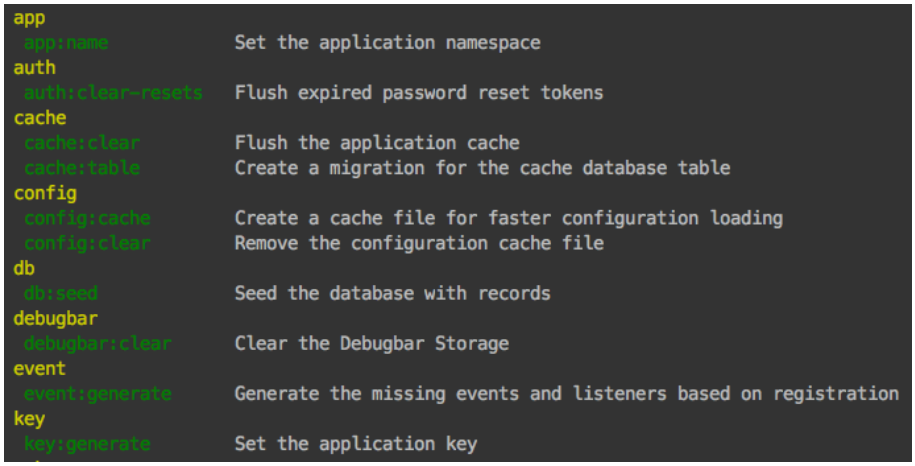
Artisan je příkazová řádka (CLI), která je postavena na komponentě Symfony Console. Obsahuje příkazy pro zrychlení a ulehčení opakujících se činností prováděných při tvorbě webových aplikací. Artisan umí např. zobrazovat seznam URL routování aplikace, pracovat s migracemi, přepínat stav aplikace, generovat kostry částí aplikace, pracovat s frontou úloh, atd. Jak příkazová řádka vypadá, je znázorněno na obrázku 2.1.

2.1.1 Vlastní příkazy

Laravel umožňuje vytvářet vlastní příkazy, které jsou přístupné přes příkazovou řádku Artisan. Je tak možné vytvořit vlastní příkaz, který ještě více usnadní práci s aplikací.

2.2 Facades

V rámci frameworku Laravel je *Facade* třída, která poskytuje přístup k objektu voláním statických metod. Jedná se tedy v podstatě o „aliasy“, které při svém zavolání vykonají odpovídající metodu objektu, který je uložen v kontejneru aplikace. Použití tohoto způsobu volání metod lze docílit především zpřehlednění, kdy je na první pohled zřejmé, co daný kus kódu dělá.



Obrázek 2.1: Ukázka příkazové řádky Artisan

Facades se nastavují v hlavním konfiguračním souboru `config/app.php`, kde se určuje název a odpovídající třída. Například pro třídu odpovídající za správu kešování by se jednalo o

```
'Cache' => Illuminate\Support\Facades\Cache::class
```

Využití *Facades* nám také přináší možnost budoucích úprav aplikace. V případě kdy není vyhovující třída, kterou má framework registrovanou jako daný alias, stačí nám pouze v konfiguraci změnit třídu pro alias, která implementuje stejné metody. Zbytek kódu aplikace při využití *Facades* není potřeba měnit.

2.3 Fronta

Fronta umožňuje odložit provedení časově náročného procesu, jako např. odesílání mailu či importování velkého množství dat. Tento způsob značně urychlí běh aplikace. Framework Laravel nabízí jednotné API[15] pro různé nástroje implementující práci s frontou. Nastavit jaký nástroj se bude o správu fronty starat lze upravit v souboru `config/queue.php`. Je možné vybrat některou z podporovaných služeb – pomocí databáze, Beanstalkd, Amazon SQS, Redis či synchronní provedení fronty (zejména vhodné pro vývojové prostředí).

2.4 Databáze

Framework Laravel má pro práci s databázemi hned několik nástrojů – Schema Builder, třídu DB, Query Builder a Eloquent ORM.

[15] Application Programming Interface

2.4.1 Podporované databáze

Laravel podporuje tyto databázové systémy: MySQL, Postgres, SQLite a SQL Server.

2.4.2 Schema Builder

Schema Builder je třída obsahující metody pro definici a úpravy struktur databázových tabulek. Schema Builder pracuje nad všemi podporovanými databázemi.

2.4.3 Migrace

Migrace jsou jakýmsi verzovacím systémem nad databází. Umožňují upravovat databázovou strukturu, aniž by ohrozily chod aplikace, ale i samotnou databázi. Migrace jsou třídy, které se ukladají do složky `database/migrations`. Obsahují dvě metody – `up` a `down`. Metoda `up` slouží k přidávání nových tabulek, sloupců nebo indexů do databáze, zatímco metoda `down` provádí přesný opak. V těchto metodách se obvykle využívá Schema Builder.

2.4.4 Plnění databáze daty

Laravel poskytuje způsob, jak naplnit databázi počátečními daty. Používá pro tento účel třídy (tzv. *Seeders*), které jsou uloženy ve složce `database/seeds`.

2.4.5 Eloquenty ORM

Eloquent ORM poskytuje jednoduchou implementaci návrhového vzoru Active Record[16]. Každá hlavní databázová tabulka má vlastní model, který se stará o interakci s touto tabulkou. Tento model může mít definovány vztahy s dalšími modely. Tyto vztahy pak odpovídají tabulkám v databázi (v případě vztahu N:M), které tuto funkčnost zajišťují.

2.4.5.1 Polymorfní vztahy

Kromě klasických vztahů mezi modely umožňuje Eloquent ORM jednoduchý zápis vztahu s více než jednou tabulkou – tzv. polymorfní vztahy.

Například příspěvky (`posts`) a jejich komentáře (`comments`) lze opatřit ohodnocením (`likes`). Využitím polymorfních vztahů lze vytvořit návrh, který tuto umožní využití jedné společné tabulky pro oba případy. Jednoduchá ukázka, jak takový příklad vypadá, je na obrázku 2.2.

Dva důležité atributy zde jsou `likeable_id` a `likeable_type`. Atribut `likeable_id` bude obsahovat primární klíč příspěvku, či komentáře, zatímco `likeable_type` bude obsahovat název třídy. Buď celý název včetně jmenných

[16] Architektonický návrhový vzor pro práci s datovými zdroji

Obrázek 2.2: Ukázka polymorfního vztahu mezi tabulkami

prostorů, či zkrácený název pomocí mapovací tabulky. Tyto dva atributy tak dohromady určují přesně jeden komentář, či příspěvek, kterému patří.

Tento návrh se hodí pro zjednodušení, a zmenšení počtu tabulek, které by plnily stejný účel, a obsahovaly stejné informace, až na jiný cizí klíč.

2.5 Tvorba balíčku

Balíčky jsou hlavním způsobem jak do frameworku Laravel přidávat novou funkcionalitu. Pro využití nějakého z balíčků ho stačí přidat do požadavků v souboru `composer.json`. Existuje několik druhů balíčků, některé jsou nezávislé a fungují tak s jakýmkoliv frameworkem (např. Carbon[17] či Behat[18]). Další jsou pak vytvořeny speciálně pro framework Laravel. Tyto balíčky pak mohou obsahovat vlastní registrované stránky, řadiče, šablony a vlastní konfiguraci uzpůsobenou pro použití s frameworkem Laravel.

2.5.1 Service Provider

Třída *Service provider* spojuje framework Laravel a balíček. Je zodpovědná za registrování všech tříd a s souborů do hlavního kontejneru Laravelu, a určuje tak odkud má framework načíst soubory jako šablony, konfiguraci, či lokalizované překlady.

[17] http://carbon.nesbot.com/
[18] http://docs.behat.org/en/v3.0/

Kapitola 3

Analýza existujících rešení

3.1 Řešení pro frameworku Laravel

Na serveru Packagist i Packalyst[19] existuje velké množství balíků, které zastupují jen část logiky nákupního procesu, a to především jednoduché napojení nákupního košíku, který implementuje přidávání či odebírání zboží. Z nich je nejstahovanější balíček Laravel Shopping Cart (59 774 stažení). Oproti tomu existuje jen několik balíčků, které řeší problematiku implementace jednoduchého internetového obchodu. Jedná se především o balíčky Laravel shop (2 086 stažení) a Aimeos Laravel (3 312 stažení).

3.1.1 Shopping Cart

Jelikož se jedná o nejvíce stahovaný balíček týkající se eshopů, a to s výraznou převahou, je zřejmé, že mnoho uživatelů vyžaduje pro svoji aplikaci obdobnou funkcionalitu. Návrh aplikace tedy bude uzpůsoben tak, aby umožňovala implementovat funckcionalitu košíku samostatně.

3.1.2 Laravel shop

Jedná se o balíček pro Laravel 5.1, který přidává základní funkcionalitu eshopu, a nejvíce odpovídá definovaným cílům této práce. Je však vytvářen pro jiný trh, kde jsou zvyklosti fungování eshopu odlišné. Jedná se např. o implicitní podpoře platební brány PayPal, která není na našem trhu příliž rozšířená. Samotný proces implementace platebních bran je velmi kvalitní, a může se tak stát inspirací pro vlastní implementaci balíčku.

Dále není balíček uzpůsoben k okamžitému nasazení (poskytuje pouze jednotlivé funkce, nikoliv hotový proces objednávky) a programátora, který tento balíček implementuje do své aplikace, tak čeká značný kus práce, než je možné provést první nákup.

[19] Uložiště pro PHP balíčky frameworku Laravel – http://packalyst.com/

3.2 Řešení pro jiné frameworky

Na serveru Packagist není k nalezení mnoho balíčků, které by implentovaly funkcionalitu internetového obchodu. Naopak existuje veliká nabídka bezplatných, či licencovaných řešení eshopů, které poskytují veškerou funkčnost.Tyto aplikace ale nenabízí, či přímo zakazují, možnost vlastních úprav, nebo jsou tyto úpravy velmi obtížné. Neumožňují tedy vývoj vlastní aplikace internetového obchodu podle vlastních požadavků.

3.2.1 Sylius

Sylius je rešení pro PHP framework Symfony2, které implementuje funkční internetový obchod. Tento systém je tvořen serií jednotlivých komponent, které dohromady tvoří výsledný produkt. Rozdělení kódu do několika menších komponent značně zpřehledňuje celý projekt, a umožňuje lepší správu, testování a vývoj jednotlivých částí. To zatím není potřeba v návrhu aplikace řešit, ale pro budoucí vývoj se jedná o jednu z možných cest.

Kapitola 4

Návrh systému

Cílem této kapitoly je navrhout stažitelný balíček (rozšíření) pro PHP framework Laravel, který přidá funkcionalitu jednoduchého eshopu. Tento balíček by měl velmi zjednodušit a urychlit následnou tvorbu vlastní aplikace internetového obchodu a jeho následnou údržbu, jelikož vývojář aplikace nebude muset znovu navrhovat a implementovat části, které jsou v základu pro všechna eshopová řešení shodná či velmi podobná a bude se tak moci soustředit na ostatní požadavky.

Balíček by neměl implementovat funkcionalitu, která se sice internetováho obchodu týká (či je vyžadována), ale je řešena jiným balíčkem, či přímo samotným frameworkem Laravel, jako například autentizace a autorizace uživatele. Systém by měl být navržen tak, aby umožňoval celý proces objednávky, jak ze strany zákazníka, tak majitele eshopu, samostatně ihned po instalaci. Zároveň musí být dostatečně jednoduchý, aby bylo možné tento balíček využít jako základ při budování vlastního internetového obchodu, který bude obsahovat velké množství dalších funkcí. Dále musí být dostatečně flexibilní a konfigurovatelný, aby umožňoval napojení na již existující systém.

4.1 Funkční požadavky

Balíček rozšiřuje aplikaci o funkcionalitu jednoduchého eshopu. Musí tak spňovat následující požadavky:

- Jednoduchá administrace produktů
- Jednoduchá administrace kategorií produktů
- Jednoduchá administrace objednávek
- Možnost zadání slevy na produkt či celou kategorii
- Možnost více druhů měn

- Možnost více jazykových lokalizací
- Vložení a správa položek v nákupním košíku
- Zadaní dodacích a fakturačních údajů
- Zadaní způsobu platby a dopravy
- Možnost platby přes platební bránu GoPay
- Vytvoření jednoduché faktury
- Odeslání potvrzujícího e-mailu zákazníkovi
- Import produktů ve formátu XML
- Export objednávek ve formátu XML

4.2 Uživatelské role

Internetový obchod s vlastní webovou administrací vyžaduje autentizaci i autorizaci uživatelů. Ačkoliv balíček nabízí i jednoduchou administraci obchodu, neimplementuje vlastní systém rolí uživatelů. Využívá pouze zabudovaný systém autentizace z frameworku Laravel. Zabezpečení přístupu stránek administrace je na programátorovi, který balíček využívá, a možnosti jak tohoto zabezpeční dosáhnout jsou popsány v kapitole 5.7.5.

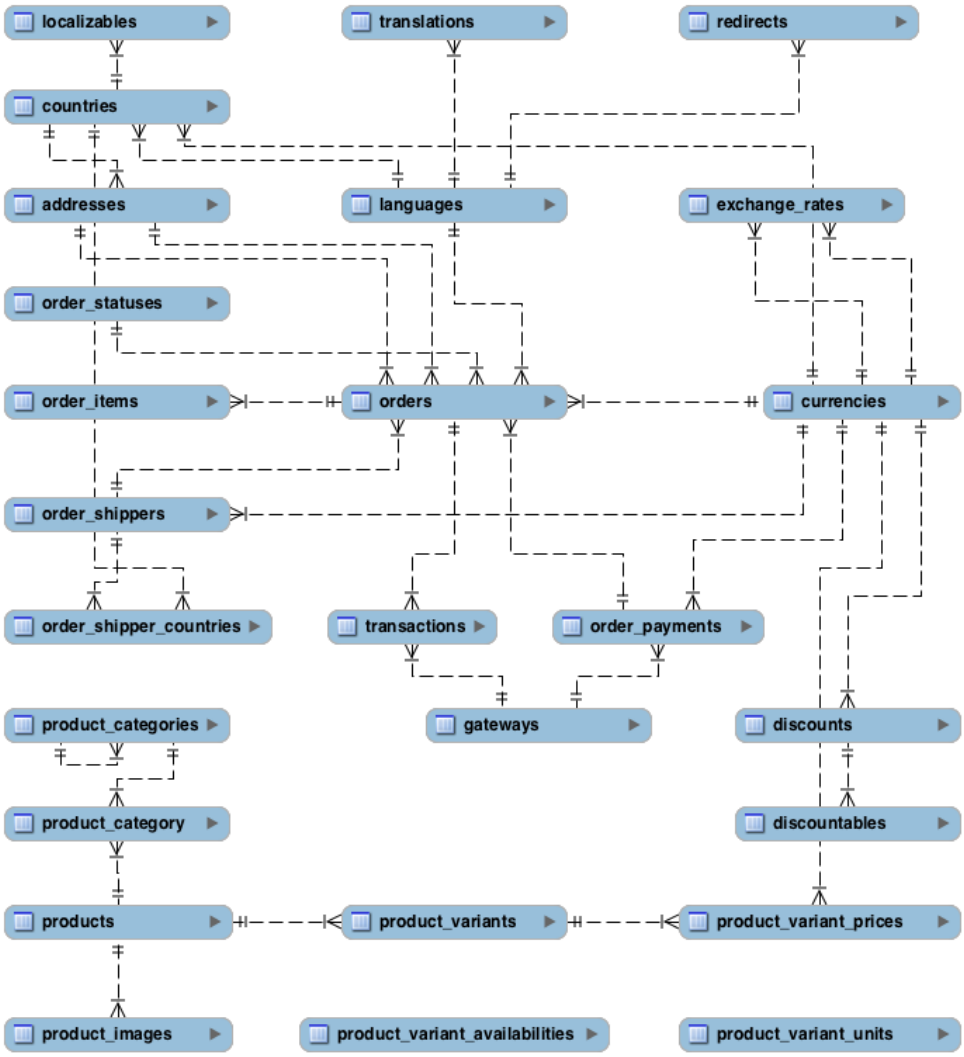
4.3 Datový model

Návrh datového modelu balíčku počítá s co možná největší škálovatelností aplikace, tak aby zajistil bezproblémovou implementaci dalších funkcí a rozšíření. Jedná se především o lokalizaci všech textů zobrazovaných zákazníkovi, možnost více variant produktu, možnost zadávat více cen v různých měnách, implementace více ceníků pro uživatelské skupiny atd. Některé tyto funkce vlastní implementace v aktualním stavu nevyužívá, ale zajišťuje si tak možnost budoucího růstu, bez nutnosti přepracovávat datový model aplikace.

Všechny tabulky, které zodpovídají za ukládaní dat o entitách (např. produkty, objednávky, kategorie) mají stejné dva atributy, které Eloquent ORM vyžaduje pro některé své funkce. Nebudou tedy uváděny u návrhu každé tabulky zvlášť. Jedná se o atributy:

- `created_at` – čas, kdy byl záznam vytvořen
- `updated_at` – čas, kdy byl záznam naposledy upraven

Celý datový model se skládá z 26 tabulek a je včetně vzájmených vztahů mezi tabulkami zobrazen na obrázku 4.1.



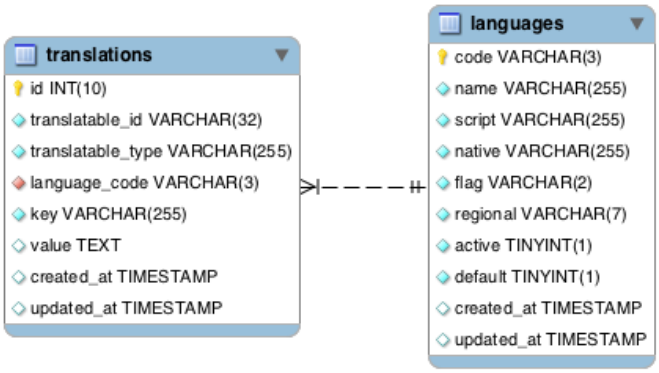
Obrázek 4.1: Datový model

4.3.1 Lokalizace textů

Framework Laravel nabízí jednoduchý systém pro lokalizaci textů [19]. Lze tedy snado vytvářet multijazyčné aplikace. V případě eshopu, kde majiteli jde o co největší návštevnost a počet objednávek, a tím větší zisk, by bylo nevhodné, aby se omezil jen na jednojazyčný trh. Návrh balíčku tak počítá s možností lokalizace textů entit, aby bylo možné vytvořit kompletně lokalizované verze aplikace.

Na obrázku 4.2 jsou naznačeny tabulky, které zodpovídají za uložení přeložených textů. Samotná tabulka `languages` zodpovídá za seznam jazyků, které může aplikace používat. Je tak možné vytvořit jednojazyčnou aplikaci pou-

ným omezením na jeden jazyk v této tabulce. Zároveň tak rozšíření aplikace o lokalizované texty do dalšího jazyka pouhým přidáním nového záznamu. Pro samotné překlady není tato tabulka nutná (ty si vystačí pouze s kódem jazyka, který poskytuje framework Laravel), pokud tedy uživatel již využívá jiný systém pro ukládání dostupných jayzyků, není nutné tuto tabulku využívat, či dokonce vytvářet.



Obrázek 4.2: Datový model lokalizace

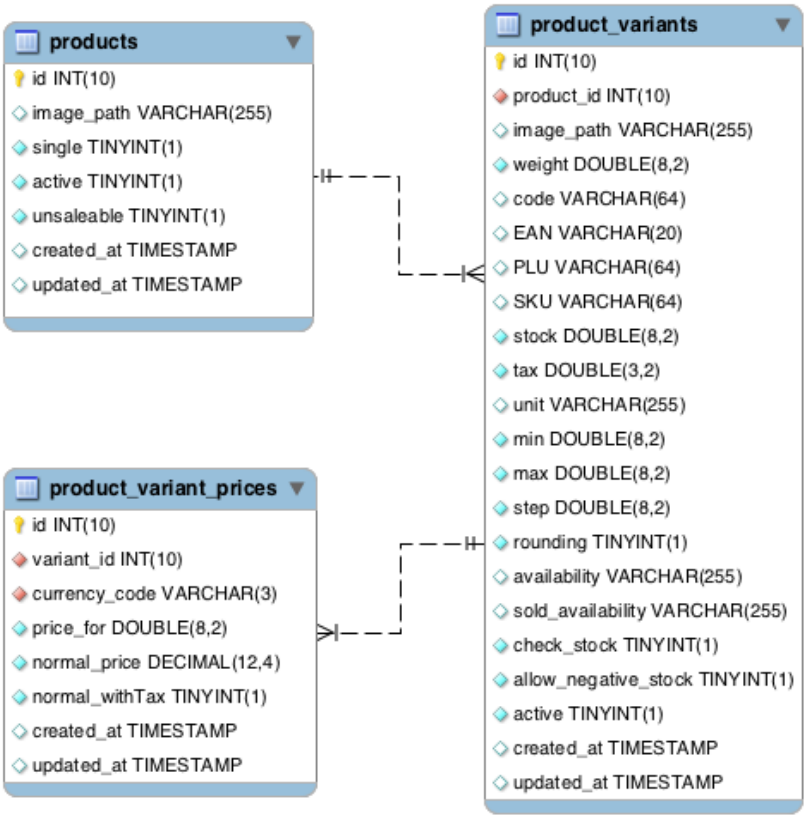
Pro vztah s entitami, které mají lokalizované texty slouží tabulka `translations`. Tato tabulka obsahuje dva atributy (`translatable_id` a `translatable_type`), které dohromady tvoří cizí klíč entity, které daný záznam patří. Napojení na tuto tabulku se následně řeší velmi jednoduše pomocí Eloquent ORM tzv. polymorfním vztahem (viz kapitola 2.4.5.1).

- `translatable_id` – cizí klíč lokalizované entity
- `translatable_type` – třída lokalizované entity
- `language_code` – cizí klíč jazyka překladu
- `key` – typ překládané položky (např. název, popis)
- `value` – přeložený text pro daný typ

4.3.2 Produkt

Entita produktu je hlavní část celého eshopu. Samotný produkt tvoří tři tabulky – samotný produkt, jeho varianty, a ceny těchto variant, jak je znázorněno na obrázku 4.3. V případě kdy aplikace využívá pouze jednu variantu a cenu pro každý produkt, je tento návrh zbytečně složitý a mohl by být vyřešen pomocí jedné tabulky. Použitý návrh na druhou stranu poskytuje možnost pro snadné rozšíření aplikace o varianty produktu, či ceníky uživatelů. Je tak výhodnější navrhnout modely tak, aby byly na tuto možnost připraveny,

a ušetřily tak budoucí práci, kdy by musel být pracně předěláván již fungující systém. To je ostatně jeden z hlavním cílů tohoto balíčku.



Obrázek 4.3: Datový model produktu

Stejně tak nejsou v tabulce produktu uloženy texty jako název, popis, atd. Tyto atributy jsou uloženy pomocí tabulky `translations`, jak je popsáno v kapitole 4.3.1.

Tabulka `product` má tak jen několik atributů, zbylé informace jsou umístěny v tabulkách, které danému produktu příslušejí.

- `image_path` – uložená adresa obrázku
- `single` – zda se jedná o produkt s variantami
- `unsaleable` – zda je daný produkt možné prodávat
- `active` – zda je produkt viditelný na eshopu zákazníkem

4.3.2.1 Varianta produktu

Tabulka `product_variants`, která ukládá data pro jednotlivé varianty produktů, obsahuje většinu informací. Každá varianta přísluší některému produktu. Produkt tak může mít několik variant, ale pro správné fungovaní aplikace musí mít vždy alespoň jednu.

- `image_path` – uložená adresa obrázku
- `weight` – váha v kilogramech
- `code` – obecný unikátní kód
- `EAN` – čárový kód
- `PLU` – identifikačních číslo zboží
- `SKU` – unikátní kód typicky používaný obchody a obchodníky
- `stock` – skladové zásoby
- `tax` – hodnota daně
- `unit` – jednotka produktu (např. ks, kg, ml)
- `min` – minimální množství, které je možné nakoupit
- `max` – maximální množství, které je možné nakoupit v jedné objednávce
- `step` – množství, po kterém je produkt přidáván do objednávky
- `rounding` – zda se výsledné množství zaokrouhluje na celá čísla
- `availability` – zobrazená dostupnost produktu (např. skladem)
- `sold_availability` – zobrazená dostupnost při vyprodání produktu
- `check_stock` – zda se zakoupené množství odečítá ze skladových zásob
- `allow_negative_stock` – povolené záporné množství skladových zásob
- `active` – zda je varianta viditelná na eshopu zákazníkem

4.3.2.2 Cena varianty produktu

Tabulka `product_variant_prices` zodpovídá za ukládání informací o ceně produktu. Každá cena přísluší některé variantě produktu. Varianta produktu může mít více různých cen. Je tak možnost rozdílných cen pro různé měny (pokud je systém nechce pouze přepočítávat pomocí měnového kurzu), či implementace ceníků pro uživatelské skupiny.

Dále je tabulka navržena tak, aby umožňovala přidání dalších atributů cen, jako například nákupní cena, při co nejnižších požadavcích na změny v samotné implementaci.

- `currency_code` – kód měny
- `normal_price` – hodnota ceny
- `normal_withTax` – zda je hodnota uvedena včetně daně
- `price_for` – jakému množství zásob cena odpovídá

4.3.3 Kategorie

Základním systémem pro třídění produktů je systém kategorií produktů. Tabulka `product_categories` je navržena tak, aby umožňovala tvorbu nekonečně hlubokého stromu kategorií. Každá kategorie tak může mít libovolné množství podřazených kategorií. Produkty jsou na kategorie napojeny pomocí vztahu N:N (viz obrázek 4.4), produkt tedy může patřit do libovolného množství kategorií.

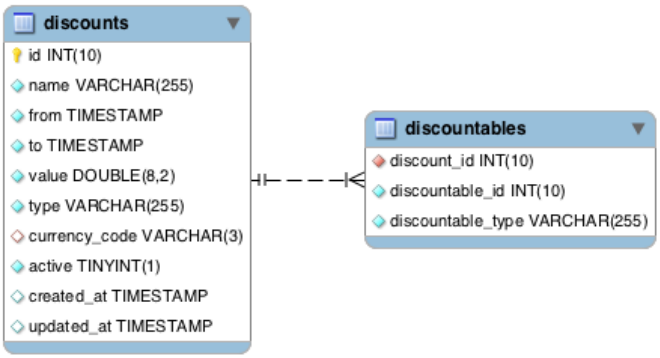
Obrázek 4.4: Datový model kategorie

V tabulce, stejně jako u produktů, nejsou uloženy lokalizované texty. Tyto atributy jsou uloženy pomocí tabulky `translations`. Tabulka `product_categories` má tak jen několik atributů.

- `image_path` – uložená adresa obrázku
- `order` – pořadí kategorie při výpisu
- `parent_id` – cizí klíč nadřazené kategorie

4.3.4 Slevy

Aby byl balíček co nejvíce rozšiřitelný, je vztah entit s tabulkou slev vyřešen pomocí polymorfního vztahu (viz kapitola 2.4.5.1). Tabulka `discountables` tedy obsahuje dva atributy (`discountable_id` a `discountable_type`), které dohromady tvoří cizí klíč entity, kterému daný záznam patří.



Obrázek 4.5: Datový model slev

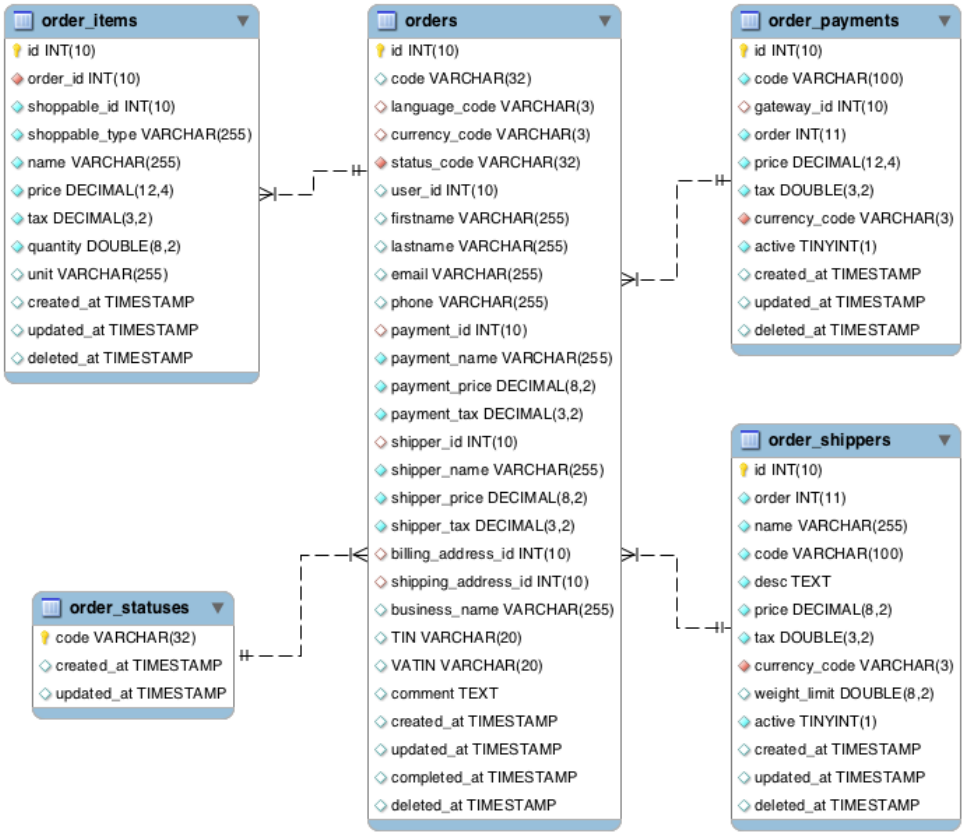
Tímto rešením je možné přiřadit slevy i jiným entitám, než jen produktům. Je tedy možné napojit na slevy celé kategorie, či naopak pouze určitou variantu produktu. Také je možné určit slevu pro daný typ způsobu dopravy, apod.

Tabulka `discounts` tentokrát přímo obsahuje atribut pro název, jelikož se zobrazuje pouze interně a není nutné tento název překládat. Opět je tu ale možnost model drobně upravit, aby lokalizovaný název podporoval. Mimo název tabulka obsahuje atributy:

- `from` – datum, od kdy je sleva aktivní
- `to` – datum, do kdy je sleva aktivní
- `type` – druh slevy (např. absolutní, procentuální)
- `value` – hodnota slevy
- `currency_code` – kód měny
- `active` – zda je sleva aktivní

4.3.5 Objednávky

Objednávky jsou nejdůležitejší součástí internetového obchodu. Musí také kopírovat velké množství informací, aby bylo zajištěno, že objednávka zůstane stejná i po změnách jiných entit. Každá objednávka může mít libovolný počet položek, jeden aktuální stav objednávky a určený jeden způsob dopravy a platby. Tyto vztahy znázorňuje obrázek 4.6.



Obrázek 4.6: Datový model objednávky

Jelikož balíček umožňuje nákup i neregistrovaným zákazníkům, je nutné aby tabulka `orders` obsahoval i všechny dodací a fakturační údaje o zákazníkovi. Není totiž možné tyto informace vždy přebírat z tabulek uživatelů.

- `code` – kód objednávky (čitelnější zápis než primární klíč)
- `user_id` – cizí klíč na uživatele

4.3.5.1 Položky objednávky

Seznam položek v objednávce je tvořen záznamy v tabulce `order_items`. Každá položka tak přísluší jedné objednávce a jednomu produktu. Aby bylo umožněno napojit systém na již hotové rešení, není vztah s produktem rešen pomocí klasického vztahu s využitím cizího klíče, ale polymorfním vztahem s tzv. prodejnou entitou. Tato entita může být reprezentována jakýmkoliv modelem, v případě vlastního datového systému se jedná o tabulku `product_variants`. Kromě toho obsahuje následující atributy:

- `name` – název produktu
- `price` – cena za jeden kus bez daně
- `tax` – daň produktu
- `quantity` – počet objednaných kusů
- `unit` – jednotka produktu

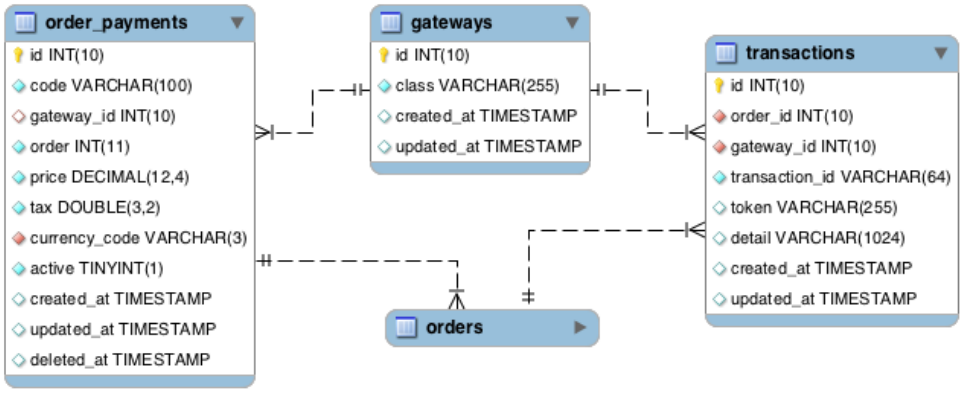
4.3.5.2 Stav objednávky

Možné stavy objednávky jsou uvedeny v tabulce `order_statuses`. Jeden ze statusů pak určuje, že se jedná o objednávku v nákupním procesu – nákupní košík. Statusy mají pouze svůj jednoznačný kód, podle kterého se dají identifikovat. Název, či popis jednotlivých stavů je rešen pomocí lokalizovaných textů v tabulce `translations`. Hodnoty stavů v balíčku jsou:

- `IN_PROGRESS` – Nákupní košík
- `PLACED` – Objednávka odeslána zákazníkem
- `PENDING_FOR_PAYMENT` – Objednávka čeká na platbu
- `DISPATCHED` – Objednávka byla odeslána zákazníkovi
- `PAID` – Objednávka byla zaplacena
- `PAYMENT_FAILED` – Platba byla neúspešná
- `DONE` – Objednávka je úspešně vyřízena
- `STORNO` – Objednávka byla stornována

4.3.5.3 Platební brány

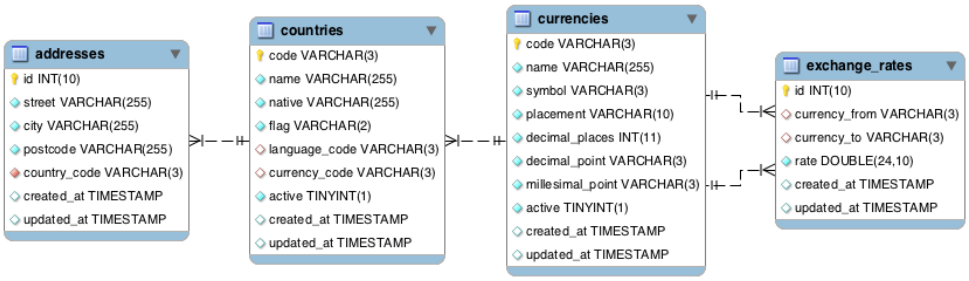
Návrh tabulky `order_payments` umožnuje napojení na platební bránu. Může tak mít vztah s jednou platební bránou v tabulce `gateways`. Ta má atribut s celým jménem třídy, která za danou platební bránu zodpovídá (viz. kapitola 5.5). Tato třída následně vytváří nové záznamy o provedených platbách, ať úspěšných, či neúspešných. Tyto informace se ukládají do tabulky `transactions`. Vztahy mezi jednotlivými tabulkami jsou znázorněny na obrázku 4.7



Obrázek 4.7: Datový model platební brány

4.3.6 Další modely

Balíček využívá další modely a jim odpovídající tabulky. Jedná se například o měny či jejich měnové kurzy. Balíček také implementuje vlastní modely pro adresy a státy. Tyto modely lze jednoduše nahradit vlastním rešením, či jiným balíčkem.



Obrázek 4.8: Další datové modely

4.4 Facades

Balíček má dvě hlavní třídy, ke kterým lze mimo jiné přistupovat pomocí tzv. *Facades* (viz kapitola 2.2). Tyto třídy jsou `Cart` a `Shop` a jsou hlavním přístupovým bodem pro práci s balíčkem, a tak obsahují všechny potřebné funkce.

4.4.1 Třída Cart

Třída `Cart` zodpovídá za nákupní košík a práci s ním. Mimo jiné implementuje následující funkce:

- `current` – nákupní košík zákazníka
- `price` – cena aktuálního nákupního košíku
- `total` – počet kusů v aktuálním nákupním košíku
- `items` – položky aktuálního nákupního košíku
- `addItem` – přidá novou položku do nákupního košíku
- `hasItem` – zkontroluje, zda je položka v košíku
- `getItem` – vratí položku z košíku
- `updateItem` – aktualizuje položku
- `deleteItem` – odstraní položku z košíku
- `clearItems` – odstraní všechny položky z košíku
- `updateDeliveryInformation` – upraví dodací informace
- `updateShippingAndPayment` – upraví způsob platby a dopravy
- `placeOrder` – odešle objednávku

4.4.2 Třída Shop

Třída `Shop` obsahuje funkce potřebné pro chod eshopu. Jedná se o výpis dat, změnu konfigurace obchodu, exporty a importy dat, registrace potřebných URL pro správný běh obchodu a mnohé další. Jednotlivé funkce jsou následující:

- `products` – vrátí všechny produkty
- `productsInCategory` – vrátí všechny produkty v dané kategorii
- `product` – vratí jeden produkt

- `categories` – vrátí všechny kategorie
- `mainCategories` – vrátí všechny hlavní kategorie
- `category` – vrátí danou kategorii
- `redirectTo` – přesměruje z aliasu
- `setCurrency` – nastaví měnu
- `convert` – převede na jinou měnu dle kurzu
- `payments` – vrátí všechny dostupné způsoby platby
- `shippers` – vrátí všechny dostupné způsoby dopravy
- `lastOrder` – vratí poslední objednávku uživatele
- `exportOrder` – exportuje danou objednávku
- `invoice` – vytvoří fakturu pro danou objednávku
- `importProducts` – importuje produkty
- `productFeed` – vytvoří XML feed pro produkty
- `getOrder` – vratí danou objednávku
- `gatewayCallback` – zpracuje odpoveď z platební brány
- `routes` – zaregistruje připravené URL stránky obchodu
- `adminRoutes` – zaregistruje připravené URL stránky administrace

4.5 Katalog produktů

Balíček implementuje jednoduchý katalog produktů. Ten tvoří stránky pro kategorie, s výpisem produktů a detail produktu. Tyto stránky zobrazují všechny potřebné informace o produktech a umožňují vložení produktu do nákupního košíku.

4.6 Nákupní proces

Balíček implementuje i základní nákupní proces. Tento proces je velmi jednoduchý a využívá jednotlivých funkcí ze třídy `Cart`. Umožní uživateli vyzkoušet si funkčnost balíčku ihned po instalaci, a názorně tak předvede použití balíčku v praxi. Proces nákupního košíku se sestává ze čtyř (resp. pěti) kroků.

1. **Nákupní košík** – Zobrazí položky v nákupním košíku a umožní změnu jejich objednaného množství a případné celkové vymazaní z nákupního košíku.

2. **Dodací údaje** – Zobrazí formulář pro vyplnění dodacích a fakturačních údajů zákazníka. Pokud je uživatel přihlášen, budou některé údaje předvyplněny z údajů zákazníkova profilu.

3. **Způsob dopravy a platby** – Zobrazí výběr z dostupných možností způsobu platby a dopravy včetně výsledné ceny.

4. **Souhrn** – Zobrazí souhrn objednávky včetně výsledné ceny za objednané produkty a vybraný způsob dopravy a platby. Zákazník může doplnit komentář k objednávce. Zákazník zde zkontroluje správnost údajů a objednávku odešle k vyřízení.

5. **Dokončení** – Zobrazí stránku s potvrzením odeslání objednávky a případně informace o dalším průběhu objednávky (očekávaný způsob zaplacení, datum doručení objednávky).

4.7 Administrace

Balíček implementuje jednoduchou administraci. Tato administrace je realizována pomocí jednotlivých REST[20] řadičů pro jednotlivé modely. Umožňují tedy zobrazit stránky pro přehled, vytvoření a změnu jednotlivých modelů. Administrace umožňuje spravovat produkty, kategorie, slevy a objednávky.

Tato administrace je velmi jednoduchá, a slouží hlavně k možnosti vyzkoušet si celkovou funkčnost balíčku ihned po instalaci. Pro produkční využití je doporučeno vytvořit vlastní, rozsáhlejší administraci, ale k základním procesům tato stačí, a je možné ji využít jako inspiraci či základ pro robustnější řešení.

[20] Representational state transfer

Kapitola 5

Implementace

5.1 Použité technologie

5.1.1 Framework Laravel

Balíček je vytvořen speciálně pro použití s frameworkem Laravel. Přímo využívá funkce tohoto frameworku, a použití s jiným frameworkem nikdy nebude optimální.

5.1.2 Twitter Bootstrap

Jelikož vytvoření šablon, či přímo designu není cílem balíčku, je použito co nejméně vlastního množství stylování. Šablony jsou tak tvořeny především pomocí moderního CSS frameworku Twitter Bootstrap 4.

5.1.3 Form Builder

Balíček obsahuje i velké, složité formuláře, jako např. formulář pro úpravu produktu v administraci. Psát a udržovat tyto formuláře přímo v HTML kódu by bylo velmi obtížné. Pro ulehčení práce při tvorbě složitých formulářů je využito balíčku Form Builder[21]. Ten je inspirován obdobnou komponentou z frameworku Symphony. Tento balíček generuje samotný HTML kód formuláře namísto vývojáře. Implicitně podporuje tvorbu formulářů ve formátu Twitter Bootstrap 3.

5.1.4 DomPDF

Balíček umožňuje generovat faktury objednávek. Je tak nutné využít službu, která zajistí správné generování PDF[22] souboru. Bylo rozhodnuto využívat

[21] https://github.com/kristijanhusak/laravel-form-builder
[22] Portable Document Format – přenosný formát dokumentů

knihovnu DomPDF[23], přesně pak balíček DomPDF Wrapper for Laravel 5[24], pro snazší intergraci s frameworkem Laravel.

5.2 Service Provider

Před samotným vývojem balíčku je nutné vytvořit třídu, která ho bude propojovat s frameworkem Laravel – tzv. *Service provider* (viz kapitola 2.5.1). Tato třída, v tomto případě `ShopServiceProvider`, obsahuje kód pro registrování migrací, překladů, konfiguračního souboru, šablon a vlastních příkazů.

5.3 Modely

Balíček implementuje modely, které využívají Eloquent ORM pro perzistenci dat s databází vytvořenou podle návrhu datového modelu v kapitole 4.3. Modely mají funkce, které implementují příslušné vztahy mezi entitami. Modely se nacházejí ve jmenném prostoru `Novott20\Shop\Entities`.

5.3.1 Interfaces

Balíček nevyžaduje využití vlastních modelů. Programátor tak může využít již existující modely (např. ze stavájícího řešení) a napojit je na objednávkový proces balíčku. Aby bylo zajištěno, že tyto modely budou obsahovat všechny potřebné metody pro správné fungování všech procesů balíčku, musí implementovat rozhraní (*interface*), které specifikuje potřebné metody. Tato rozhraní jsou např.:

- `Shoppable` – Toto rozhraní umožňuje přeměnit jakýkoliv model, na tzv. prodejný model – model, který lze vkládat do nákupního košíku. Toto rozhraní implementuje třída varianty produktu `Product\Variant`.
- `Discountable` – Třídě správně implementující toto rozhraní je umožněno využívat vytvořených slev. Toto rozhraní implementují třídy produktu `Product`, varianty produktu `Product\Variant` a kategorie produktu `Product\Category`.

5.3.2 Traits

Některé složitější vztahy, jako např. propojení modelů s jejich lokalizovanými texty v tabulce `translations`, jsou implementovány pomocí tzv. *traits* [20]. To umožňuje jednoduché napojení dalších modelů, které vyžadují stejnou funkcionalitu, aniž by vznikaly redundatní kusy kódu.

[23] https://github.com/dompdf/dompdf
[24] https://github.com/barryvdh/laravel-dompdf

- `Translations` – Stará se o správné přístupové metody k atributům lokalizovaných textů, tak aby umožňovaly stejnou syntaxi jako pro přístup k vlastním atributům modelu pomocí Eloqent ORM. Pro výpis názvu produktu, jehož instance je uložená v proměnné `$product`, jsou možné všechny následující způsoby:

  ```
  $product->translate('cz')->getTranslation('name');
  $product->getTranslation('name');
  $product->cz->name;
  $product->name;
  ```

  Pro výpis názvu přeloženého do angličtiny, který není v aplikaci nastavený jako základní jazyk, jsou možné způsoby:

  ```
  $product->translate('en')->getTranslation('name');
  $product->en->name;
  ```

- `Shoppable` – Správně implementuje všechny vyžadované metody ze stejnojmenného rozhraní.

- `CanApplyDiscount` – Přidává modelu metody, které dokáží určit výslednou slevu a tuto slevu aplikovat.

- `CanConvertPrices` – Přidává modelu metody, které dokáží konvertovat cenu modelu do určené měny.

5.4 Pomocné třídy

Ačkoliv framework Laravel nabízí velké množství pomocných tříd a komponent, bylo nutné vytvořit i některé vlastní. Tyto třídy se nacházejí ve jmenném prostoru `Novott20\Shop\Support`.

- `Repository` – Obecný repozitář, který umí pracovat s jakýmkoliv modelem. Tato třída je jakási nadstavba nad Eloquent ORM. Implementuje rozšířené metody pro vytváření, úpravu a mazaní modelu. Umožňuje také jednoduché filtrování či hledání nad modelem pomocí zřetězených funkcí. Například pro výběr prvních pěti produktů v dané kategorii a daným názvem se jedná o následující sekvenci metod.

  ```
  $repository->inCategory(1)->search('Test')->limit(5)->all();
  ```

  Pro jednotlivé modely lze vytvářet vlastní repozitáře, které obsahují specifické metody pro daný model. Stačí, když tyto třídy rozšiřují původní třídu `Repository`.

- `RESTController` – Tento řadič implementuje všechny metody (`index`, `create`, `store`, `show`, `edit`, `update`, `destroy`) vyžadované pro vytváření, úpravu a mazaní modelu. Tento řadič využívá především administrační část aplikace.

5.5 Platební brány

V balíčku je podporována platební brána GoPay. Napojení balíčku na tuto bránu implementuje třída `Novott20\Shop\Gateway\GoPay`. Je však možné jednoduše přidat podporu jiné platební brány. Přidání nové platební brány může být provedeno vytvořením PHP třídy, která bude implementovat rozhraní (*interface*) `Novott20\Shop\Contracts\Gateway`.

5.6 Artisan CLI

Balíček rozšiřuje příkazovou řádku Artisan o tři nové příkazy.

- `shop:migrations` – Zkopíruje migrace z balíčku do správné složky aplikace (`database/migrations`). Tyto migrace obsahují třídy pro vytvoření databáze potřebné pro fungování internetového obchodu.
- `shop:seed` – Nahraje počáteční data do databáze, aby nebylo nutné vždy záznamy vytvářet ručně. Jedná se o základní jazyky, měny, stavy objednávky, či testovací produkty a kategorie.
- `shop:setup` – Tento příkaz provede všechny potřebné úkony, které je potřeba provést po instalaci balíčku, aby byl internetový obchod plně funkční. Je tak nejjednoduším způsobem, jak balíček zprovoznit.

5.7 Instalace

Balíček má repozitář na serveru GitHub[25] na adrese `https://github.com/novott20/shop`. Je také registrován na serveru Packagist pod stejným jménem. Postup jak balíček naistalovat a úspešně napojit na aplikaci ve frameworku Laravel je taktéž popsáno na hlavních stránkách projektu na obou zmíněných stránkách.

5.7.1 Instalace balíčku

Balíček lze naistalovat pomocí příkazu `composer require novott20/shop`, či přidáním `"novott20/shop": "dev-master"` do souboru `composer.json` do pole závislostí aplikace a následně příkazem `composer install` či `composer update`.

[25] https://github.com/

5.7.2 Registrace balíčku

Registrace balíčku je provedena přidáním cesty ke třídě `ShopServiceProvider` balíčku do pole `providers` v hlavním konfiguračním souboru `config/app.php`.

```
Novott20\Shop\Providers\ShopServiceProvider::class,
```

Je také možné registrovat aliasy pro *Facade* třídy balíčku do pole `aliases` ve stejném souboru.

```
'Shop' => Novott20\Shop\Facades\ShopFacade::class,
'Cart' => Novott20\Shop\Facades\CartFacade::class,
```

5.7.3 Konfigurace balíčku

Publikování konfiguračního souboru balíčku, lokalizovaných překladů a šablon do příslušných složek aplikace lze provést pomocí příkazu:

```
php artisan vendor:publish --provider="Novott20\Shop\Facades\
ShopServiceProvider"
```

5.7.4 Nastavení databáze

Nejjednoduším způsobem, jak zprovoznit balíček, je s využitím příkazu:

```
php artisan shop:setup
```

Ten provede všechny potřebné úkony – zkopírování migrací a provedení migrace, nahrání počátečních dat do databáze.

5.7.5 Registrace routování

Balíček sám o sobě neregistruje stránky katalogu produktu či objednávkového procesu do aplikace. Pokud chce uživatel využít tyto předpřipravené funkcionality, stačí pak vložit

```
\Shop::routes();
\Shop::adminRoutes();
```

do souboru `app/Http/routes.php`. Balíček nabízí odělené funkce pro routování zákaznické a administrační části. Umožňuje tak, aby administrační část podléhala autorizaci.

Kapitola 6

Testovaní

Při vývoji aplikace velmi nastává moment, kdy je velmi obtížné, zbytečně zdlouhavé a prakticky nereálné testovat aplikaci manuálně. Proto se pro kontrolu korektního fungování aplikace využívá automatické testování. Automatické testování se sestává z testů, které ověřují, že zdrojový kód aplikace pracuje správně a dle očekávání programátora [21]. Využití automatických testů má i vedlejší efekt, kdy se stává zdrojový kód přehlednější, jelikož je nutné dodržovat určité standardy pro psaní testovatelného kódu [22].

Automatické testování je možné rozdělit do několika úrovní – od nejnižší možné úrovně jednotkových testů, které testují jen jednu izolovanou třídu, přes integrační testy, až po funkční testy, které testují korektní chování celé aplikace [22].

6.1 Jednotkové testy

Jednotkové testy jsou nejnižší možnou úrovní automatického testování. Jednotkové testy obvykle testují pouze vybranou konkrétní jednotku, která je izolovaná od ostatních částí programu. Z pohledu procedurálního programování je jednotkou program, funkce či procedura. Z pohledu objektově orientovaného programování (OOP) je jednotkou obvykle třída, či konkrétní metoda [23]. U jednotkových testů se kontrolují především správné návratové hodnoty metod v závislosti na vstupních parametrech.

Jelikož se testuje pouze vybraná jednotka (třída), mělo by se při jednotkovém testování snažit o maximální izolaci testované jednotky. Testovaná třída by neměla volat metody jiných skutečných tříd, přistupovat k síti, využívat skutečné databáze či využívat filesystém. Pro simulaci těchto činností se používájí falešné objekty, které splňují požadované rozhraní, ale nevykonávají žádnou skutečnou činnost. Tyto objekty se označují jako *mocks* či *stubs*.

6.2 Integrační testy

Integrační testy testují, jak spolu jednotlivé třídy aplikace navzájem spolupracují. Testují tedy, zda jedna třída splňuje požadavky jiné. Oproti jednotkovým testům je možné využívat připojení ke skutečné databázi či filesystému, protože jde především o testování spolupráce.

6.3 Laravel a automatické testování

Framework Laravel je po instalaci nastaven pro testování testovacím frameworkem PHPUnit[26]. Kromě frameworku PHPUnit využívá Laravel také komponenty Symfony HttpKernel, DomCrawler a BrowserKit [24]. Tyto komponenty umoňují při testování prohledávat a obsluhovat pohledy (*views*) aplikace, pomocí simulací webového prohlížeče.

6.4 PHPUnit

PHPunit je standard pro testování v PHP. Byl vytvořen pro tvorbu jednotkových testů, ale lze jej použít i pro tvorbu integračních testů či funkčních testů [25]. Integrační testy jsou oproti jednotkovým testům širší a mohou testovat více částí (tříd) aplikace současně [21].

PHPUnit pro ověření, zda má funkce výsledek dle očekávání, využívá tvrzení, tzv. asercí (*assertions*). Například pro otestování, zda určitá hodnota odpovídá hodnotě `true`, lze použít tvrzení `assertTrue`.

PHPUnit má k dispozici mnoho typů asercí. Kompletní seznam asercí je popsán v oficiální dokumentaci frameworku PHPUnit [26].

6.5 Testování balíčku

Pro zajištění stability kontroly budoucích změn implementuje balíček sérii integračních testů, které kontrolují správné fungování hlavních *Facade* tříd `Shop` a `Cart` určených v kapitole 4.4. Testování provádí celkem 31 testů. Náhled, jak vypadá výsledek testování, je znázorněn na obrázku 6.1.

```
PHPUnit 4.8.24 by Sebastian Bergmann and contributors.

...............................

Time: 7.14 seconds, Memory: 49.50Mb

OK (31 tests, 71 assertions)
```

Obrázek 6.1: Výsledek automatických integračních testů tříd `Shop` a `Cart`

[26] https://phpunit.de/

Závěr

V práci se povedlo navrhnout a implementovat stažitelný balíček pro PHP framework Laravel. Tento balíček přidává funkcionalitu jednoduchého eshopu. Umožňuje tedy vytvoření a úpravu objednávky, správu produktů v nákupním košíku, zadaní dodacích a fakturačních údajů, zadání způsobu platby a dopravy, možnost platby přes vybranou platební bránu, vytvoření jednoduché faktury a export a import z účetního systému. Balíček tedy implementuje všechny požadavky, které byly definovány na začátku práce. Dokonce implementuje některé funkce navíc, jako např. možnost zadání absolutních a procentuálních slev, podporu více měn či podporu jazykových lokalizací. Kód dodržuje vnitřní filozofii frameworku Laravel a snaží se o čistý a přehledný kód, ve kterém se lehce zorientuje i začínající programátor. Zároveň umožňuje snadné rozšíření do robustní velké eshopové platformy, která může konkurovat stávajícím freewarovým řešením.

Literatura

[1] *Počet e-shopů v Česku vzrostl během pěti let o 80 více nakupují oblečení a obuv [online].* [cit. 2016-04-18]. Dostupné z: http://finexpert.e15.cz/pocet-e-shopu-v-cesku-vzrostl-behem-peti-let-o-80--cesi-stale-vice-nakupuji-obleceni-a-obuv

[2] Lukášová, J.: *Jak si založit vlastní e-shop? [online].* [cit. 2016-04-18]. Dostupné z: http://www.podnikatel.cz/clanky/jak-si-zalozit-vlastni-e-shop/

[3] Croft, J.: *Frameworks for Designers [online].* 2007, [cit. 2016-04-18]. Dostupné z: http://alistapart.com/article/frameworksfordesigners

[4] Otwell, T.: *Twitter účet Taylora Otwella [online].* [cit. 2016-04-18]. Dostupné z: https://twitter.com/taylorotwell

[5] Otwell, T.: *On laravel's Future: Part 2 [online].* [cit. 2016-04-18]. Dostupné z: https://laravel-news.com/2014/11/laravels-future-part-2/

[6] Otwell, T.: *Laravel Philosophy [online].* [cit. 2016-04-18]. Dostupné z: https//laravel.com/docs/4.2/introduction

[7] Skvorc, B.: *Best PHP Frameworks for 2014.* Prosinec 2013, [cit. 2016-04-18]. Dostupné z: http://www.sitepoint.com/best-php-frameworks-2014/

[8] Skvorc, B.: *The Best PHP Framework for 2015: SitePoint Survey Results.* Duben 2015, [cit. 2016-04-18]. Dostupné z: http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

[9] Surguy, M.: *History of Laravel PHP framework, Eloquence emerging [online].* [cit. 2016-04-18]. Dostupné z: http://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging/

[10] Way, J.: *Webový portál Laracast [online].* [cit. 2016-04-18]. Dostupné z: https://laracasts.com

[11] *The MIT License (MIT) [online].* [cit. 2016-04-18]. Dostupné z: https://opensource.org/licenses/MIT

[12] *PSR-0: Autoloading Standard [online].* [cit. 2016-04-18]. Dostupné z: http://www.php-fig.org/psr/psr-0/

[13] *PSR-1: Basic Coding Standard [online].* [cit. 2016-04-18]. Dostupné z: http://www.php-fig.org/psr/psr-1/

[14] *PSR-2: Coding Style Guide [online].* [cit. 2016-04-18]. Dostupné z: http://www.php-fig.org/psr/psr-2/

[15] Chen, N.: *Convention over Configuration [online].*

[16] Reenskaug, T.: *Models-Views-Controllers [online].* 1979, [cit. 2016-04-18]. Dostupné z: https://heim.ifi.uio.no/~trygver/2007/MVC_Originals.pdf

[17] *MVC aplikace & presentery [online].* [cit. 2016-04-18]. Dostupné z: https://doc.nette.org/cs/2.2/presenters

[18] *PSR-4: Autoloader [online].* [cit. 2016-04-18]. Dostupné z: http://www.php-fig.org/psr/psr-4/

[19] Otwell, T.: *Oficiální dokumentace frameworku laravel [online].* [cit. 2016-04-18]. Dostupné z: https://www.laravel.com/docs

[20] Vrána, J.: *PHP 5.4 přináší změny a nové jazykové konstrukce [online].* 20011, [cit. 2016-04-18]. Dostupné z: https://www.zdrojak.cz/clanky/php-5-4-traits/

[21] Way, J.: *Laravel Testing Decoded [online].* Leanpub, 2013.

[22] Zamrzla, J.: *Testování v PHP: tvorba testovatelného kódu [online].* 2013, [cit. 2016-04-18]. Dostupné z: http://www.zdrojak.cz/clanky/testovani-v-php-tvorba-testovatelneho-kodu/

[23] Hunt, A.; Thomas, D.: *Programátor pragmatik: jak se stát lepším programátorem a vytvářet kvalitní software*, ročník 1. Brno: Computer Press, 1990, 266 s.

[24] *Laravel Testing [online].* [cit. 2016-04-18]. Dostupné z: https://laravel.com/docs/5.2/testing

[25] *Testing [online].* [cit. 2016-04-18]. Dostupné z: http://silex.sensiolabs.org/doc/testing.html

[26] *Appendix A. Assertions [online].* [cit. 2016-04-18]. Dostupné z: `https://phpunit.de/manual/current/en/appendixes.assertions.html`

Příloha **A**

Seznam použitých zkratek

ACL (Access Control List). Seznam oprávnění na objekt.

AJAX (Asynchronous JavaScript and XML). Označení pro techniku využívající několik technologií, které umožňují měnit obsah webové stránky na pozadí.

CI (Continuous Integration). Označení pro nástroje, které mají za úkol ověřovat kvalitu software při vývoji. Integrace změn má být podle definice průběžná.

CLI (Command-line Interface). Uživatelské rozhraní ovládané přes příkazovou řádku.

CMS (Content Management System). Anglické označení pro redakční systém.

CSS (Cascading Style Sheets). Kaskádové styly upravují způsob zobrazení (X)HTML dokumentů.

DI (Dependency Injection) je technika předávání závislostí mezi službami. V momentu inicializace objektu dochází k předávání závislostí pomocí veřejného rozhraní třídy (například pomocí konstruktoru).

DIC (Dependency Injection Container). Označení pro kontejnér služeb, který se využívá v návrhovém vzoru Dependency injection.

FTP (File Transfer Protocol). Protokol pro přenos souborů v počítačové síti.

FTPS (File Transfer Protocol over SSL). Zabezpečená varianta FTP protokolu využívající kryptografické protokoly TLS a SSL.

GUI (Graphical User Interface) Představuje uživatelské rozhraní ovládané přes grafické prvky.

HTML (HyperText Markup Language). Značkovací jazyk využívaný při vývoji webových stránek.

HTTP (Hypertext Transfer Protocol). Internetový protokol pro přenos dokumentů ve formátu (X)HTML .

IDE (Integrated Development Environment). Software pro vývoj aplikací integrující veškeré potřebné funkce pro pohodlnou práci s určitým jazykem.

JS (JavaScript). Skriptovací jazyk nejčastěji využívaný pro programování problémů ve webových aplikací na straně uživatele.

MVC (Model-view-controller) Architektura rozdělující aplikaci do tří vrstev. Na vrstvu datovou, pohledovou (často zastoupena šablonovacím jazykem) a logickou.

OOP (Object-oriented Programming). Technika vývoje software, která využívá koncepci objektů. Nejčastějšími stavebními prvky OOP jsou rozhraní, třídy a jejich metody.

ORM (Object-relational mapping), neboli objektově relační mapování je vrstva, která zajišťuje mapování dat z relační databáze na objekty programovacího jazyka a naopak.

PDO (PHP Data Objects). Rozšíření, které umožňuje pracovat jednotně s různými databázemi.

PHP (PHP: Hypertext Preprocessor). Skriptovací jazyk primárně využívaný pro programování dynamických webových stránek. oduchých HTTP volání.

REST (Representational state transfer). Způsob jak jednoduše vytvořit, číst, editovat nebo smazat informace ze serveru pomocí jednoduchých HTTP volání.

SFTP (SSH File Transfer Protocol over SSL). Protokol pro přenos souborů po síti využívající SSH-2.

SQL (Structured Query Language). Standardizovaný dotazovací jazyk využívaný v relačních databázích.

URL (Uniform Resource Locator). Řetězec pevně daného tvaru, který popisuje přesnou lokaci zdroje na internetu.

VPS (Virtual Private Server), neboli virtualizovaný server, který nabízí výhody vyhrazeného serveru na sdíleném stroji.

WYSIWYG (What you see is what you get). Editory nabízející uživateli editaci textu obdobnou formou jako u kancelářských balíků.

XML (Extensible Markup Language). Standardizovaný značkovací jazyk, který vznikl jako zjednodušená varianta staršího jazyka SGML.

Příloha **B**

Obsah přiloženého CD

- `implementation` ..... zdrojové kódy implementace
- `thesis`
  - `img` ..... obrázky
  - `biblography.bib` ..... literární zdroje
  - `BC_Novotny_Tomas.pdf` ..... výsledná forma práce ve formátu PDF
  - `BC_Novotny_Tomas.tex` ..... zdrojová forma práce ve formátu LaTeX