UNIVERZITA PARDUBICE

DOPRAVNÍ FAKULTA JANA PERNERA

DIPLOMOVÁ PRÁCE

2020 Bc. Ondřej Mareš

Univerzita Pardubice

Fakulta Jana Pernera

Rozšíření redakčního systému o e-commerce modul

Bc. Ondřej Mareš

Diplomová práce

2020

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Ondřej Mareš**
Osobní číslo:	**D17446**
Studijní program:	**N3708 Dopravní inženýrství a spoje**
Studijní obor:	**Aplikovaná informatika v dopravě**
Téma práce:	**Rozšíření redakčního systému o e-commerce modul**
Zadávající katedra:	**Katedra informatiky v dopravě**

Zásady pro vypracování

Rozšíření redakčního systému o e-commerce modul.
Vytvořte rozšiřující modul redakčního systému, v teoretické části popište používané technologie, případně jejich alternativy, proveďte rešerši několika již existujících dostupných řešení.
V praktické části vytvořte PHP modul do redakčního systému Fork CMS, který rozšíří stávající redakční systém o možnosti vytváření a správu objednávek. Modul by měl odpovídat klasické struktuře modulů redakčního systému ForkCMS a využívat všechny možnosti které redakční systém nabízí, jako bloky, widgety, překlady. Popište postup vytváření modulu.

Rozsah pracovní zprávy: **40 normostran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. MAUTHE, Andreas a Peter THOMAS. Professional content management systems: handling digital media assets. Chichester: John Wiley, c2004. ISBN 0-470-85542-8.
2. LOCKHART, Josh. Modern PHP: new features and good practices. Sebastopol, CA: O'Reilly Media, 2015. ISBN 978-1-491-90501-2.
3. WELLING, Luke a Laura THOMSON. PHP and MySQL web development. Fifth edition. Hoboken, NJ: Addison-Wesley, [2017]. Developer's library. ISBN 978-0-321-83389-1.

Vedoucí diplomové práce: **Ing. Josef Šroll, Ph.D.**
Katedra informatiky v dopravě

Datum zadání diplomové práce: **28. listopadu 2019**
Termín odevzdání diplomové práce: **31. ledna 2020**

L.S.

doc. Ing. Libor Švadlenka, Ph.D.
děkan

doc. Ing. Karel Greiner, Ph.D.
vedoucí katedry

V Pardubicích dne 28. listopadu 2019

Prohlašuji:

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

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

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 23. 1. 2020

Ondřej Mareš

PODĚKOVÁNÍ

Chtěl bych poděkovat za podporu své rodině, všem vyučujícím z DFJP za vždy vstřícný a nápomocný přístup, panu Ing. Ondřeji Míčovi za typografické rady a panu Ing. Josefu Šrollovi, Ph.D. za rady a vedení této diplomové práce.

ANOTACE

Diplomová práce pojednává o vývoji e-commerce rozšíření pro open source redakční systém. V teoretické části je provedena rešerše existujících redakčních systémů a popsány použité technologie. V praktické části je popsán vytvořený modul pro redakční systém Fork CMS.

KLÍČOVÁ SLOVA

php, symfony, framework, e-commerce, open-source, forkcms

TITLE

Creating e-commerce extension for content management system

ANNOTATION

The diploma thesis deals with the development of extension e-commerce solution for content management system. In the theoretical part is done research of existing content management systems and described used technologies in practical part. The practical part describes the created module for content management system Fork CMS.

KEYWORDS

php, symfony, framework, e-commerce, open-source, forkcms

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

CMS	Content Management System
CVE	Common Vulnerabilities and Exposures
MIME	Multipurpose Internet Mail Extensions
Composer	Utilita určená pro správu knihoven.
CSV	Comma-Separated Values
Dependency Injection	Technika pro vkládání závislostí mezi komponentami programu.
ForkCMS	Redakční systém napsaný v programovacím jazyce PHP a frameworku Symfony.
JWT	JSON Web Token
MVC	Model View Controler – návrhový vzor
OSL	Open Software License
placeholder	Zástupný kód nebo znak.
RFC	Request for Comments
routa	Konkrétní URL překlad.
tag	prvek/element
wysiwyg	what you see is what you get – textový editor s formátováním
frontend	Části webu viditelné běžným návštěvníkům.
backend	Části webu skryté běžným návštěvníkům.

1 ÚVOD

Pro prezentaci svého nápadu, nebo podniku na internetu se často používají redakční systémy, které jsou specializované spíše na práci s textovým a multimediálním obsahem. Ve světě podnikání na internetu, zejména prodeji zboží, nebo výrobků jsou používány složitější redakční systémy, označované jako e-shopy. Tyto systémy umožňují spravovat obsah bez znalostí programovacích, nebo skriptovacích jazyků a díky své specializaci na určitou činnost mají své výhody a nevýhody. Jednou z nevýhod e-shopového systému je, že se nehodí pro prodej menšího počtu zboží, služeb a neumožnuje takovou správu obsahu webu jako klasické CMS. Jednoduché redakční systémy zase nejsou přímo zaměřeny na prodej, mnohdy lze docílit možnosti prodeje různými rozšířeními. Tato diplomová práce se zaobírá tvorbou e-commerce modulu určeného k prodeji menšího počtu produktů a služeb. Tento modul by měl být řešením pro projekty, kde řešení klasickým e-shopovým systémem je nevhodné, nadbytečné a časově náročné. Výsledkem by měl být modul, který by společně s redakčním systémem dokázal poskytnout návštěvníkovi webu možnost nakoupit a správci webu spravovat své objednávky.

2 CÍL PRÁCE

V teoretické části budou popsány použité technologie, případně jejich alternativy a provedena rešerše několika již existujících dostupných řešení.

Praktická část má za cíl vytvořit modul do redakčního systému ForkCMS, který rozšíří redakční systém o možnosti vytváření a správu objednávek. Modul by měl odpovídat klasické struktuře modulů redakčního systému ForkCMS a využívat všechny možnosti, které redakční systém nabízí, jako jsou bloky, widgety, překlady. Bude sepsán postup vytváření modulu.

Základní požadavky na modul:

- Správa objednávek.
- Správa zákazníků.
- Každá objednávka má svůj stav.
- Možnost, aby správce mohl přidávat a mazat stavy objednávek.
- Po odeslání objednávky bude odeslán zákazníkovi e-mail se sumarizací.
- Správa e-mailových šablon s možností dynamického doplnění dat.
- Možnost exportovat objednávky.
- Správa platebních metod.
- Správa přepravních metod.
- Vazba mezi platební metodou a přepravní metodou.
- Možnost vytvoření příplatkové služby.
- Jednoduchý nákupní proces.

3 REŠERŠE REDAKČNÍCH SYSTÉMŮ

3.1 Content management system (CMS)

CMS je označení pro softwarové aplikace, které slouží k vytváření a modifikaci digitálního obsahu bez znalosti programování. V dnešní době se tak nejčastěji označují webové aplikace, které jsou v česku také nazývány redakčními systémy. Existuje celá řada různých CMS, která se specializují na určitá odvětví, cílové skupiny, rozsah řešení, použité technologie a nabízejí tak jiné možnosti správy obsahu. Existují jak komerční řešení, tak i „open source“. Výhodou užití těchto systémů je efektivní správa uživateli, kteří nemusí mít znalosti programování a kódování. V této diplomové práci budou popsány a porovnány příklady klasických redakčních systémů s komplexními redakčními systémy určenými k prodeji, označovanými jako e-shopy. [1]

3.2 Open source – výhody a nevýhody

Open source neboli „otevřený software“ je označení pro software s otevřeným kódem, který můžeme volně upravovat a používat. Je důležité mít na paměti, že open source neznamená zároveň „free software“ i když rozdíl není mnohdy patrný. Rozdíl najdeme v licencích. Free software je distribuován za nulovou cenu a je možné si sním dělat cokoliv. U „open source“ to může být trochu jinak. Vývojář a majitel softwaru může podmínit do kdy je software zadarmo. Ukázkovým příkladem může být nový český framework Shopsys, který se zaměřuje na vývoj e-shopu. Komunitní edice je zdarma do doby, kdy je celkový roční objem tržeb e-shopu pod 12 milionů euro. Poté je nutné si zakoupit jednu z licencí. [2][3]

3.2.1 Výhody pro uživatele

Jedna z prvních otázek při rozhodování o vzniku webu, nebo e-shopu je, proč vlastně použít „open source“. První věcí, která asi nejvíce evokuje je pořizovací cena, protože tento software lze stáhnou zadarmo. Jelikož tyto systémy jsou většinou v podstatě funkční hned po instalaci, tak to vybízí u nenáročných podnikatelů ke správě e-shopu vlastními silami. To znamená i minimální náklady na provoz. Výhoda je to pro ty, co své podnikání nesoustřeďují primárně na internet a nechtějí do e-shopu moc investovat.

Dále je tu další skupina náročnějších podnikatelů, kteří mají představu o tom, co chtějí, potřebují jen konkrétní funkcionality, ale nechtějí platit za vývoj vlastního systému. Tito prodejci si tedy mohou najít firmu, která jim open source systém upraví a zkompletuje dle jejich potřeb. Mnohdy tyto firmy dělají údržby, aktualizace, technickou podporu a někdy i správu obsahu. Výhodou je také často dostupný trh s moduly. Moduly jsou univerzálním rozšířením systému. Tyto moduly je někdy také možné pořídit pro svůj obchod zadarmo, nebo za částku, která se mnohdy může

nákladově pohybovat ve zlomcích ceny vývoje modulu vlastního. Asi největší výhoda webu postaveného na open source tkví v tom, že v případě, kdy například majitel e-shopu není spokojený s firmou, která mu e-shop vytvořila, jednoduše může vzít celý svůj web a přejít k jiné firmě. [2][3]

3.2.2 Výhody pro prodejce e-commerce řešení

Častou otázkou bude asi: „Proč pracovat s cizím softwarem, který nemusí splňovat všechny mé požadavky a může mě licenčně omezit?“. Jako první odpověď si můžeme říci: „Proč vyvíjet něco, co už je vytvořeno“.

Dalším parametrem budou nízké vstupní náklady. Vlastní systém se musí navrhnout, nějakou dobu trvá, než se vytvoří první použitelná verze, systém se musí testovat a aktualizovat. Open source systém, který má silnou komunitní základnu, je většinou neustále vyvíjen a pravidelně aktualizován. Navíc, je možné nalákat uživatele těchto systémů a nabídnout jim vylepšení, nebo přebrat zákazníka od konkurence. [2][3]

3.2.3 Výhody pro vývojáře systému

„Proč investovat peníze do vývoje a pak to nabízet zadarmo?“ Odpovědí bude nejspíše více. Může to být proces budování značky, může to být důkaz, že svému produktu vývojáři věří a přitáhne jim nové zákazníky. Jako příklad můžeme vzít firmy, které nabízejí zdarma svůj e-shopový systém. Pokud je systém dobrý, vznikne komunita, která bude software používat, různě modifikovat, uživatelé systému mohou objevit různé chyby a nedostatky, takže se dá říci, že je to i forma testování softwaru. Lidé, kteří systém používají a jsou spokojeni se mohou jednodušeji stát zákazníky firmy a nechat si udělat komplexní řešení internetového obchodu. Povědomí o firmě a softwaru se může šířit bez investic do marketingu a může přilákat nového zákazníka. Distribuce vlastního systému jako open source může mít pozitivní vliv na prezentaci firmy a může přilákat silné obchodních partnery. V případě potřeby firmy najmout nového vývojáře, existuje možnost získat člověka, který má se systémem již nějakou zkušenost a tím snížit náklady na zaškolení. [2][3]

3.2.4 Nevýhody

Za největší nevýhodu by se dala považovat nutnost pravidelně aktualizovat. Když se používá systém s velkou komunitní základnou, najdou se i ti, kteří se budou snažit jiné poškodit. Na různých webových serverech se zveřejňují bezpečnostní problémy různých systémů a jedním

takovým je například server cvedetails.com[1]. Vývojáři většinou zveřejní záplatu, která problém vyřeší. Dá se říci, čím populárnější web, tím větší nebezpečí. Pokud web není dost rychle aktualizován, hackeři, kteří tyto bezpečnostní chyby sledují toho mohou využít. Může se jim podařit vyřadit web z provozu, falšovat cenu objednávek, nebo odcizit osobní údaje (například e-mailové adresy, nebo čísla platebních karet). Tedy cokoliv, co chyba umožní.

Další nevýhodou může být náročnost systému. Vývoj modulů a komponent pro redakční systém může být časově, ale i vědomostně náročný. Špatně se mohou hledat lidé, kteří dokážou systém efektivně upravovat.

Mnohdy tyto systémy obsahují části, které ne vždy nevyužijí, jen zabírají prostředky a ubírají na přehlednosti.

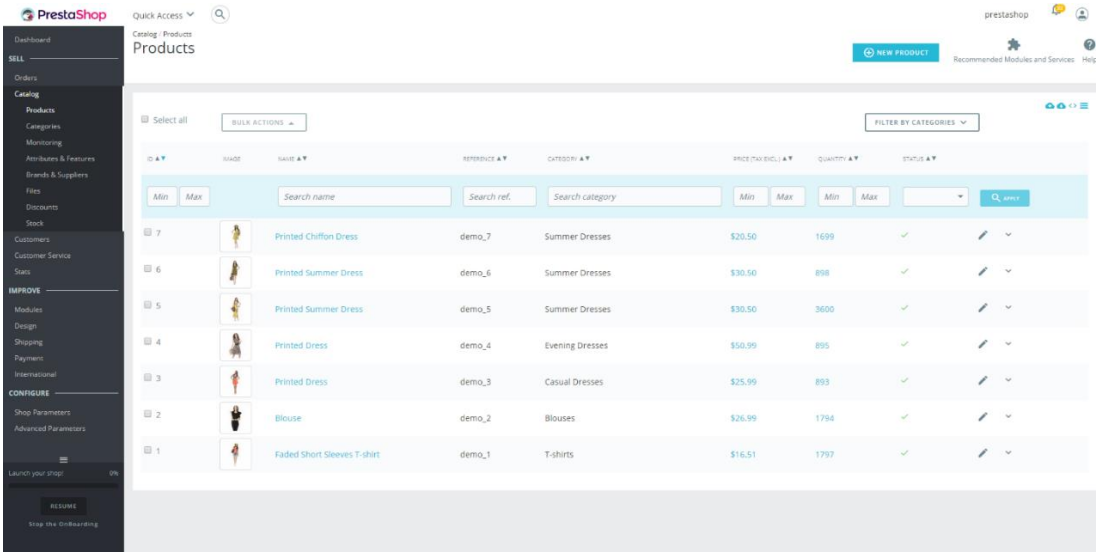
Mohou také nastávat problémy s uplatňováním licenčních podmínek. Například, že se uživatel systému nepřihlásí k nákupu licence, když již nesplňuje podmínky pro užití open source. [2][3][4]

3.3 Redakční systémy zaměřené na prodej

3.3.1 Prestashop

Prestashop je open source redakční systém distribuovaný pod OSL licencí. Vývoj systému započal v roce 2007. Systém je modulární a lze koupit již hotové moduly. Systém má efektivní správu objednávek, produktů, plateb, přeprav, výrobců, dodavatelů a skladů. Systém také umožňuje nahrávat a vytvářet vlastní šablony. Díky široké komunitě, lze nalézt mnoho informací pro řešení vzniklého problému. Dle informací udávaných vývojářem, existuje celosvětově více jak 300 000 obchodů běžících na Prestashopu. Dle aktuálních dat serveru buildwith.com[2] je 287 596 aktuálně běžících na Prestashopu, 526 808 webů, které někdy Prestashop používalo a z toho 5 914 v české Republice. Prestashop je 2. nejpopulárnějším CMS v České republice v kategorii e-commerce a stále nabývá na popularitě. Systém je přizpůsoben správě většího množství zboží. [20]

[1]Ukázka pro Wordpress - https://www.cvedetails.com/vulnerability-list/vendor_id-2337/product_id-4096/
[2] https://trends.builtwith.com/shop/PrestaShop

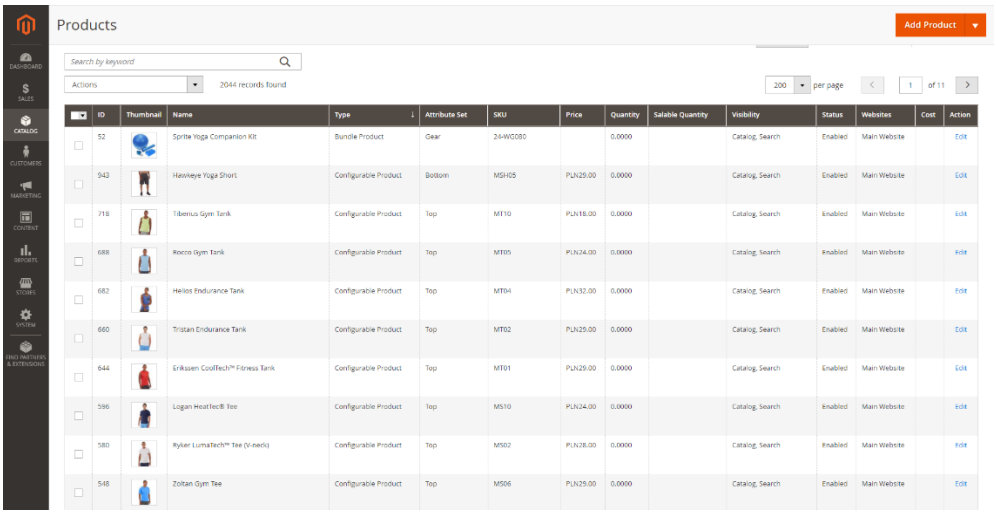


Obrázek 1 – Náhled na administraci systému PrestaShop. Zdroj: zpracování autora

3.3.2 Magento

Vývoj Magenta byl započat v roce 2008 a od té doby se stal jedním z nejpopulárnějších e-commerce systémů na světě. Systém je postaven na Zend frameworku a distribuován pod OSL licencí. Distribuované jsou dvě verze, „Community Edition“ a „Enterprise Edition“. Systém byl původně vyvíjen firmou Varien, Inc s asistencí dobrovolníků, nyní je vlastněn firmou Adobe. Systém je modulární, šablonovatelný, silně cachovaný, vícejazyčný a více doménový (možnost mít více domén a více obchodů s různou šablonou pod jednou instancí systému). V roce 2015 vyšla nová a přepracovaná verze Magento 2. Jelikož jsou verze Magento (1.9.2.4) a Magento 2 (2.2.3) nekompatibilní, nelze jednoduše přenášet moduly mezi verzemi a celkový přechod je poměrně komplikovaný. Pro první verzi existuje značné množství modulů, druhá verze zase řeší některé výkonnostní problémy. Používají se tedy zatím obě verze, ale výrobce letos oznámil že, v roce 2020 ukončí podporu pro verzi 1.9 a níže[3]. Jedním z nedostatků Magento je často vznikající problém s rychlostí načítání a musí se provádět různé optimalizační úpravy. [21][22]

[3]https://www.bounteous.com/insights/2019/04/16/magento-1-end-life-what-retailers-need-know/



Obrázek 2 – Náhled na administraci Magento. Zdroj: zpracování autora

3.3.3 Shopsys

Shopsys je novou open source e-commerce platformou uveřejněnou na začátku roku 2019 a je spíše frameworkem než konečným e-commerce řešením. Vývojářem systému je označován jako Framework Shopsys. Můžeme jej označit za takovou kostru e-shopu. Je to e-commerce platforma pro stavbu potenciálně velkých e-shopů. Instalace je náročnější než u předešle uvedených systémů. Shopsys obsahuje základní e-commerce funkce a v základu může stačit i pro méně náročný e-shop. Systém je poměrně dobře škálovatelný a výhodou je, že neobsahuje nadbytečné funkce, které mohou znepřehlednit správu a zkomplikovat potřebné modifikace systému. [23][24]

3.4 Běžné redakční systémy

3.4.1 WordPress

WordPress je open source redakční systém vyvíjený pod licencí GNU GPL. Dle dat serveru builtwith.com[4] má WordPress 83% podíl použitých open source distribucí na celém internetu, 34% podíl všech stránek na internetu. Systém je znám také pro svou velikou komunitu, modulárnost a velké množství dostupných šablon. [25]

3.4.2 Joomla

Open source redakční systém pro publikaci vlastního obsahu. První verze systému byla vydána v roce 2005 a celý projekt je vyvíjen pod licencí GNU GPL. Za svou dobu existence vyhrála řadu různých ocenění. Systém je postaven na frameworku s MVC návrhovým vzorem, který může být použit nezávisle na CMS a tím může dovolit vývojáři postavit silnou webovou

[4] https://trends.builtwith.com/cms/open-source/traffic/Entire-Internet

aplikaci. Systém je vhodný pro korporátní weby, intranety, portály, malé bussinesss weby, online magazíny, noviny, e-commerce, rezervace, stránky státních organizací, stránky neziskovek, komunitní weby a osobní stránky. VirtueMart[5] je open source e-commerce řešení postavené na redakčním systému Joomla. [26]

3.4.3 ForkCMS

Fork CMS je opensource redakční systém postaven na Symfony komponentách. Systém je modulární a na stránkách vývojářů je několik modulů ke stažení. Bohužel fork prochází poměrně náročnými změnami, kdy jsou odstraňovány závislosti na starých balíčkách třetích stran, které již nejsou updatovány nebo se v praxi ukázaly být nedostačující. Jednou z úprav je například odebrání z databázových tabulek typ enum. Enum používá ve svých tabulkách téměř každý modul do verze 4.10. Většina modulů nemá aktualizaci na změny a jsou tedy mnohdy téměř nepoužitelné. Mnoho uživatelů systému si vytváří vlastní moduly a stává se, že kvůli náročnosti aktualizace modulů neaktualizují verzi redakčního systému. [27]

Widget a Block

Fork je poměrně dobře administrovatelný, jedním z hlavních prvků, které tomuto redakčnímu systému dává sílu nad správou obsahu jsou Widgety a Bloky. Obě funkce systému mají téměř stejnou roli, liší se v případě použití. Widgety a bloky jsou vytvářeny programátorem a jsou součástí modulů. Jednotlivé widgety a bloky může uživatel systému umístit do různých částí stránky, jak sám potřebuje. [27]

3.5 Statistiky užití open source systémů

Dle statistik webového serveru buildwidth.com lze označit jako celosvětově nejpoužívanější open source redakční systém WordPress. V tabulce 1 jsou uvedeny některé redakční systémy, které byly zmíněny v této diplomové práci a jejich množství použití pro jeden milion nejlepších stránek dle návštěvnosti celosvětově.

[5] https://www.virtuemart.net/features/what-is-virtuemart

Tabulka 1 – Podíl užití CMS u 1 milionu nejúspěšnějších webů. Zdroj: https://trends.builtwith.com/shop/open-source

Redakční systém	Množství	Procentuální podíl
WordPress	359 941	35,99 %
Magento	13 510	1,35 %
PrestaShop	2 780	0,28 %
VirtueMart	811	0,08 %
Joomla	18 322	1,83 %

Pro Českou republiku je rozložení trochu jiné, nejpopulárnějším CMS je Joomla. Data uvedená v tabulce jsou z celkového počtu 61 856 detekovaných webů, postavených na open source.

Tabulka 2 – Podíl užití open source CMS u 61 856 detekovaných webů v ČR. Zdroj: https://trends.builtwith.com/shop/open-source/country/Czech-Republic

Redakční systém	Množství	Procentuální podíl
WordPress	12 469	20,19 %
Magento	718	1,8 %
PrestaShop	5 971	14,93 %
VirtueMart	1 122	2,81 %
Joomla	22 010	35,64 %

4 POUŽITÉ TECHNOLOGIE

4.1 HTTP protokol

HTTP protokol patří mezi nejpopulárnější internetové protokoly. Slouží pro přenos hypertextových dokumentů. Pomocí MIME je možné přenášet různé soubory s různým typem kódování. Využívá transportní protokol TPC. Protokol je schopen detekovat chyby při přenosu a zajistit jejich nápravu.

V době vypracovávání této diplomové práce, je HTTP2 použito na 41,8 % všech webových stránek[6] .

Tento protokol je sice univerzální, ale pomalu přestává vyhovovat požadavkům moderních webů. Například při použití HTTPS, které je dnes téměř na většině webů, vzniká větší potřeba výměny paketů. Což znamená větší časová režie. Z toho důvodu se Google rozhodl vytvořit úplně nový protokol. [28]

4.2 Webová služba (Web Service Architecture)

Webové služby jsou prostředky pro spolupráci mezi různými softwarovými aplikacemi, které mohou běžet na jiných platformách nebo frameworcích. WSA dokument má za cíl tuto architekturu definovat. Volný překlad definice: Webová služba je softwarový systém navržený k podpoře interoperability, komunikace stroje se strojem skrze síť. Typickou webovou službou je SOAP, který posílá zprávy ve formě serializovaného XML skrze HTTP protokol. Populárním je také REST API. [19]

4.3 JWT

JSON Web Token (JWT) je standardizovaný (RFC 7519) způsob bezpečného přenosu informací skrze objekt JSON. V JWT jsou zašifrovány informace pomocí tajného klíče s HMAC algoritmem, nebo pomocí public/private klíče. Tyto tokeny jsou vhodné jak pro autorizaci, tak i pro výměnu dat.

Token se skládá ze 3 částí, header, playload, signature, které jsou na sebe napojeny tečkou.

```
Header.playload.signature
```

Header obsahuje informaci o použitém algoritmu a typu tokenu.

[6] https://w3techs.com/technologies/details/ce-http2/all/all

Playload obsahuje přenášené informace, nejčastěji informace o uživateli. Tyto informace jsou trojího typu. Registrované – jsou to předdefinované, ale spíše doporučené položky informací (například exp – čas expirace). Public – položky si definuje uživatel JWT sám, ale měl by mít přehled o použití, aby nedošlo ke kolizi. Soukromé, ty se používají při komunikaci se stranami, u kterých proběhla domluva o jejich použití.

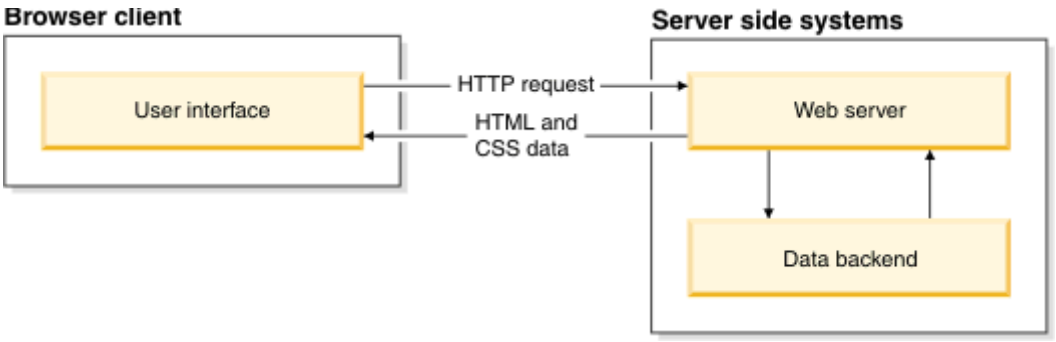
Signature slouží k prověření, zda zpráva nebyla po cestě změněna. [29]

4.4 AJAX

AJAX je označení skupiny technologií vývoje webových aplikací, které načítají obsah bez nutnosti znovunačtení celé stránky. Jedná se o asynchronní načítání a zpracování obsahu webové stránky. AJAX tvoří následující technologie:

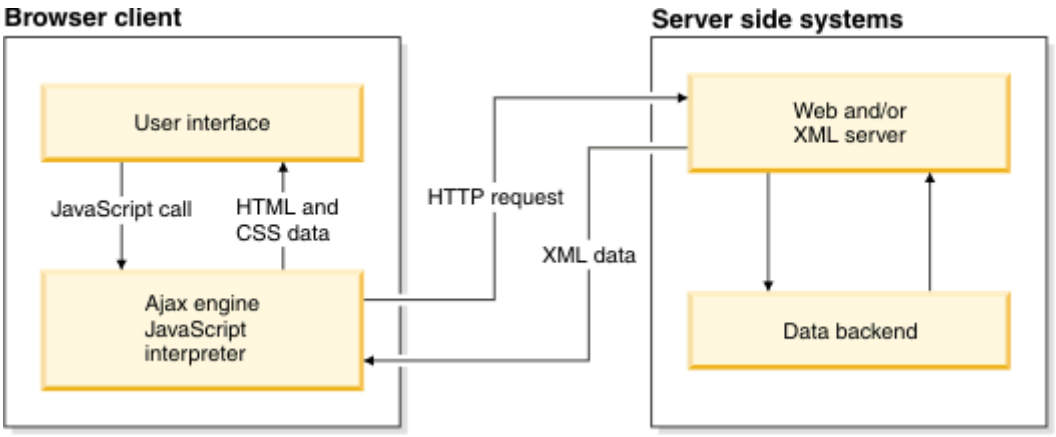
- HTML a CSS pro prezentaci dat,
- Document Object Model (DOM) pro dynamickou interakci,
- XMLHttpRequest objekt pro asynchronní komunikaci se serverem,
- JSON, XML, HTML, and XSLT pro datovou výměnu dat,
- JavaScript pro volání a zobrazení výsledku.

V klasické webové aplikaci, při přístupu na stránku, nebo v jakékoliv další interakci s jejím webovým rozhraním, je odeslán HTTP požadavek, který zpracuje webový server a odešle zpět odpověď ve formě HTML stránky. Během této komunikace je uživatel neschopen interagovat s webovou aplikací a čeká na načtení celé stránky. Celý proces je znázorněn na obrázku 3. [18]



Obrázek 3 – Klasická komunikace se serverem. Zdroj: [18]

V AJAXové webové aplikaci uživatel nemusí čekat na znovu načtení celé stránky, ale pouze toho obsahu, kterého se příslušná akce týká. Komunikace probíhá na pozadí, požadavky odesílá a zobrazuje JavaScript. Proces komunikace je znázorněn na obrázku 4. [18]



Obrázek 4 – Komunikace se serverem přes AJAX. Zdroj: [18]

4.5 JavaScript

JavaScript je scriptovací, nebo programovací jazyk, který umožňuje implementovat komplexnější požadavky na web. Je označován jako třetí vrstva standardu webových technologií (HTML, CSS a JavaScript). Pomocí JavaScriptu můžeme dynamicky měnit obsah, spravovat různá média, animovat obrázky a mnoho dalšího. Kód je psán v plain textu, jako HTML a CSS, takže nejsou zapotřebí žádné speciální nástroje pro psaní kódu. JavaScript je client-side, ale i server-site jazyk.

Client-side kód je kód, který je pouštěn v počítači uživatele. Při návštěvě stránky se kód stáhne k uživateli a je spouštěn webovým prohlížečem.

Server-side běží naopak na straně serveru a výsledky jsou staženy a zobrazeny v prohlížeči. JavaScript na straně serveru běží například v prostředí Node.js. [17]

4.6 PHP

PHP je scriptovací jazyk, interpretovaný na straně serveru. Nejčastěji se používá pro aplikace běžící na webovém serveru, ale může být taky použit jako program příkazové řádky (jako bash, Ruby, Phyton apod.). Autorem jazyka je Rasmus Lerdorf, který definoval původní význam PHP jako „Personal Home Page Tools".

Koncem roku 1998 bylo PHP3 nainstalováno na více jak 10 % procentech světových webových serverů. V současnosti se jazyk PHP rychle rozvíjí a o jeho vývoj se stará několik vývojářských týmů.

Postupem času se změnil i způsob vývoje aplikace, v minulosti bylo běžné napsat php soubor, nahrát jej na server pomocí FTP a zkoušet, zda je funkční.

Dnes pro vývoj existuje celá řada nástrojů, které usnadňují vývoj a údržbu programu. Vývoj probíhá na lokálních prostředích, která jsou identická se serverem. Jsou to například nástroje jako Vagrant, Ansible, Chef, Puppet, Git, Compser, PHP Unit.

PHP se postupem času transformovalo do moderního jazyka s možností namespace, trait, closure apod. V dnešní době čím dál méně programátorů sází na velké monolitické frameworky, ale více upřednostňuje menší specializované komponenty. Za tímto trendem stojí Composer manager, který změnil přístup k vývoji aplikací. Dovolí vývojáři využívat knihovny, které si sám dokáže kdykoliv poměrně jednoduše přidat do aplikace.

Framework se stává průvodcem programátora a vede ho k užívání určitých standardů a již zaběhlých postupů vývoje (zkušeností).

PHP běží na Zend Egine napsaném v jazyce C, nebo na novém enginu, zvaném HipHop Virtual Machine vyvíjeným Facebookem. Oba enginy by měly být navzájem kompatibilní. [12]

4.7 PHP framework

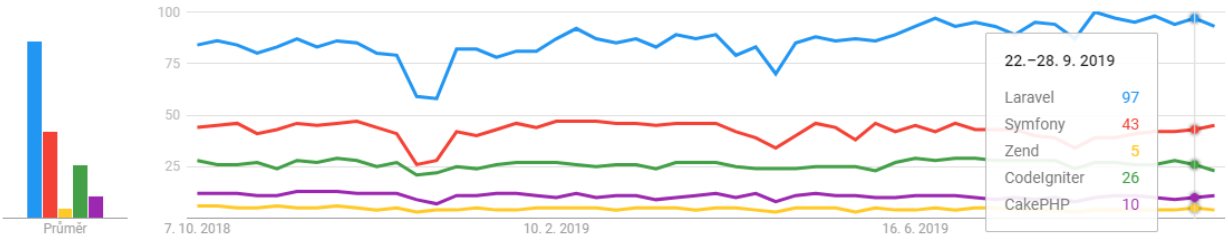
Framework je softwarová struktura, která shlukuje různé balíčky a návrhové vzory. Cílem frameworku je usnadnit a urychlit vývoj aplikace. Užitím frameworku si vývojář zkrátí dobu pro prvotní sestavování základních částí aplikace, jako je například napojení na databázi, implementace DI kontejneru, nebo automatická správa jmenných prostorů komponent. Díky určité struktuře a dodržování pravidel, nebude pro nového vývojáře problém zapojit se do probíhajícího projektu. Pravidelná aktualizace komponent pomáhá udržovat dobrou úroveň bezpečnosti systému.

Různé frameworky se specializují na různé typy aplikací, jako například klasická webová aplikace, e-shop, nebo třeba jen API pro aplikace třetích stran. Popularita frameworků je různá a nedá se určit, který je nejlepší, ale dá se uvažovat nad tím, které jsou nejpopulárnější. V tabulce číslo 3 je uvedeno několik nejznámějších PHP frameworků.

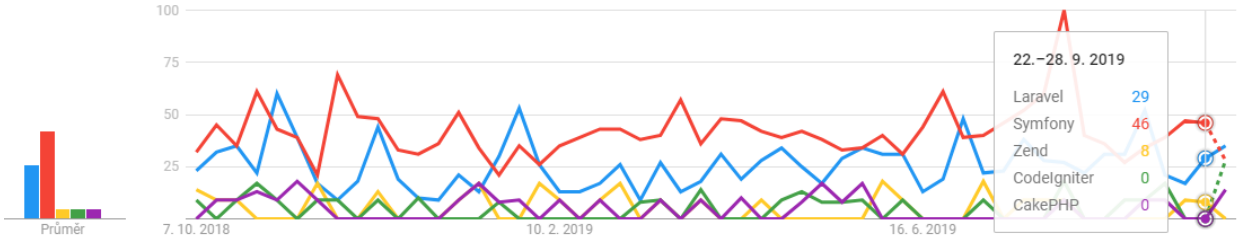
Tabulka 3 – PHP frameworky. Zdroj: https://fullstackgeek.blogspot.com/2019/05/best-php-frameworks.html

Framework	Rok vzniku	Oficiální webová stránka	Aktuální verze	licnce
Laravel	2011	www.laravel.com	5.8	MIT license
Symfony 2	2007	www.symfony.com	4.2	MIT license
Zend	2006	www.framework.zend.com	3	New BSD license
CodeIgniter	2006	www.codeigniter.com	3.1.10	MIT license
Cake PHP	2005	www.cakephp.org	3.7	MIT license

Pomocí několika webových nástrojů, můžeme porovnat výskyty jednotlivých názvů frameworků ve vyhledávání Googlu, nebo na stránkách stackowerflow.com. Nástroj pro průzkum hledaných výrazů na serveru google.com je Google Trends. Následující grafy ukazují popularitu vyhledávaných jednotlivých výrazů v rozmezí 7.10.2018 až 28.9.2019.



Obrázek 5 – Google Trends – frameworky – celosvětově. Zdroj: zpracování autora

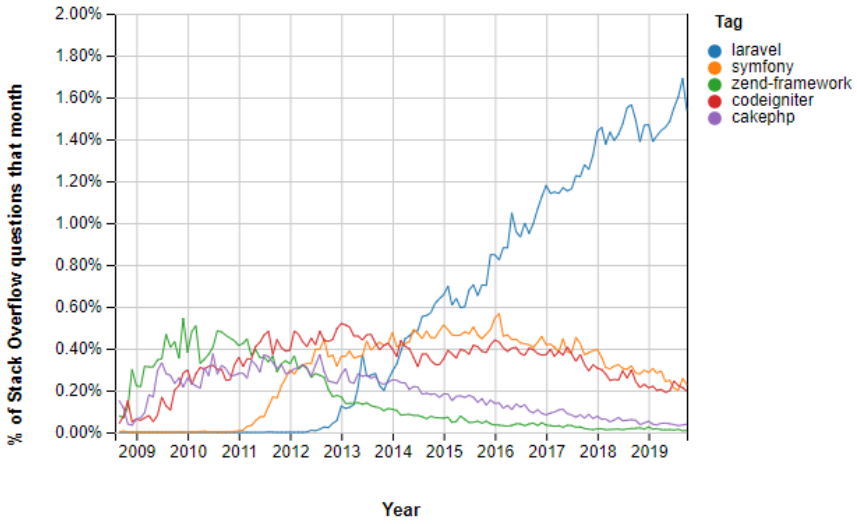


Obrázek 6 – Google Trends – frameworky – Česká republika. Zdroj: zpracování autora

Čísla v grafu představují relativní zájem ve vyhledávání vzhledem k nejvyššímu bodu grafu pro danou oblast a dobu (zájem v průběhu času). Tento zájem nabývá hodnoty od 0 až do 100. Hodnota 100 představuje nejvyšší popularitu výrazu, hodnota 0 znamená nízkou popularitu, nebo že nebyl k dispozici dostatek dat.

Uvedená data jsou čerpána ze zdrojů při vypracovávání této diplomové práce a v době čtení se mohou lišit[7].

Dalším zdrojem dat je server stackoverflow, který slouží developerům k dotazování a šíření svého způsobu řešení různých programovacích nesnází. Stackoverflow dává k dispozici nástroj, který zpřístupňuje informaci o výskytu určitých klíčových slov v dotazech. V následující tabulce, jsou zobrazeny procentuální výskyty názvů frameworků v otázkách za měsíc od počátku roku 2008 až do současnosti[8]. [15][16]



Obrázek 7 – Trendy dotazů. Zdroj: insights.stackoverflow.com.

4.7.1 Symfony

Symfony je kvalitně objektově navržený a světově velmi používaný PHP framework. Byl vydán v roce 2005 a za tu dobu ušel dlouhou cestu a získal si dobrou reputaci. Tento framework se skládá z mnoha komponent, které zjednodušují vývoj aplikace. Balíčky se stahují pomocí Composeru, lze je snadno aktualizovat, smazat, nebo stáhnout nové.

Framework je postaven na MVC architektuře. Šablony jsou psány šablonovacím jazykem Twig. Za největší benefity Symfony se dají považovat Bundly a Komponenty.

Bundle je něco jako plugin, nebo balíček PHP souborů, javascriptů, stylů a obrázků. Jeho výhodou je oddělenost od zbytku kódu. Tyto bundly je možné jednodušeji znovu použít na dalších projektech.

[7]https://trends.google.com/trends/explore?cat=5&q=%2Fm%2F0jwy148,%2Fm%2F09cjcl,Zend,%2Fm%2F02qgdkj,%2Fm%2F09t3sp#TIMESERIES
[8]https://insights.stackoverflow.com/trends?tags=laravel%2Csymfony%2Czend-framework%2Ccodeigniter%2Ccakephp

Komponenty jsou základní funkce, které mají za cíl redukovat rutinu a dávají prostor vývojáři, aby se zaměřil na konkrétní obchodní funkce. Symfony obsahuje několik desítek komponent, které usnadňují proces vývoje. Komponenty je možné rozšiřovat, přidávat vlastní moduly a taky je možné tyto moduly používat i na jiných frameworcích nebo čistém PHP. [14][15]

4.8 MySQL

MySQL je rychlý[9], robustní, víceuživatelský, relační databázový systém. Používá standardizovaný strukturovaný dotazovací jazyk (SQL). Historie vývoje sahá až do roku 1979, ale dostupný je teprve od roku 1996. Systém pracuje na principu klient-server a je určený pro různé operační systémy. Jedná se o nejoblíbenější open-source databázový systém, který mnohokrát vyhrál cenu čtenářů Linux Journal.

Je distribuovaný pod dvěma licencemi, open source (GPL) která je zadarmo za určitých podmínek a komerční pro případy, které nesplňují licenci GPL.

Jako výhody systému se dají považovat, vysoký výkon, nízká cena, jednoduchá konfigurace a edukace, přenositelnost, dostupnost zdrojového kódu, podpora a spousta dostupných řešení v diskuzích na internetu.

Díky dostupnosti zdrojového kódu MySQL, je možné jej modifikovat pro svůj vlastní projekt. Tato možnost není pro většinu uživatelů zásadní, ale dává tím jistotu stávajícím uživatelům, že v případě nutnosti mají možnost modifikovat kód pro svůj projekt. Aktuálně již existuje několik větví a náhrad za MySQL, jako je například MariaDB. [13]

4.9 HTML

HTML je značkovací jazyk využívaný pro tvorbu formátovaného obsahu, nejen webových stránek. Základním prvkem jsou tagy, ty se dělí na párové a nepárové. Tyto tagy se používají pro formátování dokumentu, jsou předem definované a mají určité vlastnosti a atributy. Pro zobrazení formátovaného výsledku je potřeba html soubor otevřít ve webovém prohlížeči. Zdrojový kód zase lze otevřít v jakémkoliv jednoduchém textovém editoru, nebo vývojovém prostředí jako je například Netbeans, nebo WebStorm. [6][7]

V roce 2018 byla zveřejněna verze HTML 5.3, která je doposud nejnovější verzí[10].

[9] http://www.mysql.com/ why-mysql/benchmarks/.
[10]https://www.w3.org/TR/html53/

Základní struktura html souboru:

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Titulek</title>
 </head>
 <body>
  <h1>Nadpis</h1>
  <hr />
  <p>Odstavec textu</p>
  <a href="/odkaz">Odkaz</a>
</body>
</html>
```

HTML spadá spolu s CSS a JavaScriptem do front-end web developmentu. HTML má na starosti obsah, CSS vzhled a JavaScript zase interaktivitu. [8]

4.10 TWIG

Šablonovací systém vytvořený Fabienem Potencierem, zakladatelem frameworku Symfony. Twig je rychlý díky sestavování kódu do optimalizovaného PHP kódu, režie oproti běžnému PHP kódu byla snížena na minimum. Bezpečný, díky sandbox módu pro vyhodnocení nedůvěryhodného kódu. Flexibilní, díky lexeru a parseru, které dovolí vývojáři vytvořit vlastní tagy a filtry. Twig je snadno naučitelný, testovaný pomocí Unit tesů, dobře zdokumentovaný, a je připraven k použití na větších projektech. [9]

Ukázka syntaxe for cyklu s možností kombinace s klauzulí else:

```
{% for user in users %}
    {{ user.name }}
{% else %}
    <p>Uživatelé nenalezeni</p>
{% endfor %}
```

4.11 CSS

CSS v češtině taky známé jako Kaskádové styly je jazyk pro definování vzhledů html elementů. Smyslem vzniku tohoto jazyka je oddělení vzhledu od obsahu. První verze publikovaná v roce 1996 byla poměrně jednoduchá, druhá verze publikovaná v roce 1998 byla už více striktně definovaná a dávala vývojáři více možností. Jelikož druhá verze byla již mnohokrát větší než první verze, stala se nepraktickou. Bylo rozhodnuto, že CSS musí být rozděleno na moduly. Ve verzi 2.1 byly opraveny chyby a bylo oznámeno rozdělení na moduly, rozdělení následovalo v CSS 3. Moduly byly rozděleny do úrovní 1, 2 a 3, nyní je nejnovější 4. úroveň, ale CSS4 nemá zatím žádnou definici.

Pro usnadnění práce se dnes často používají CSS frameworky, které mají již předpřipravené styly s podrobnou dokumentací. [6][11]

4.12 GIT

GIT je software určený k verzování soborů. Tento systém je navržen tak, aby uchovával informaci o každé provedené změně v souboru. Každý developer pracující na projektu má k dispozici celou historii projektu a může upravovat různé části kódu bez starosti ztráty, nebo přepisu. Git repozitář se často udržuje na serveru třetích stran, jako je GitHub, Bitbucket nebo Gitlab. Přes webové rozhraní těchto serverů je možné vytvářet větve, sjednocovat větve, dávat požadavky na sjednocení, nahlížet do souborů, procházet commity a mnoho dalšího bez nutnosti mít projekt zprovozněný lokálně.

Git se vyplatí používat pro projekty, kde do kódu přispívá více vývojářů, ale taky se hodí i v případě, kdy pracujeme na vlastním projektu. Pomocí pár příkazů, můžeme v případě potřeby vrátit kód do stavu jakéhokoliv předcházející verze. Verzování nám taky vytváří jistý druh dokumentace, například díky commit zprávě, je k dispozici informace s odůvodněním k provedeným změnám, ve webové aplikaci je možné dávat komentáře k různým částem kódu.

Git je tedy nástrojem, který nejenom zabezpečuje stav kódu, ale i zefektivňuje vývoj. Developer s přístupem k repozitáři může jednoduše data stáhnou, upravit, kdykoliv aktualizovat, odeslat zpět do repozitáře a to pomocí pár příkazů nebo kliků v IDE. [5]

4.13 GoPay

Gopay je platební brána, která zprostředkovává bezpečnou internetovou platbu. Platební brána je vhodná pro e-shopy, prodejce služeb, digitálního obsahu, vstupenek, jízdenek, a mobilních aplikací. Brána by měly být vhodná pro jakýkoliv druh byznysu. Gopay nabízí 3 druhy platebních bran, a to „Inline brána“, „Redirect brána“ a „Mobile brána“. Umí přes 55 platebních metod, přijímá 9 měn, umí opakované platby, dokáže nosit logo a barvy klienta. [10]

5 VÝVOJ E-COMMERCE MODULU

5.1 Verze Fork CMS a závislosti

Vyvíjený modul je napsán pro verzi redakčního systému 4.1.0. a je doporučeno mít nainstalovanou tuto konkrétní verzi. Modul má návaznost na dalších modulech, jako je modul „Profiles“ a „Catalog“.

Modul „Profiles“ je součástí základního balíku modulů v redakčním systému Fork a rozšiřuje systém o možnost vytváření a správu uživatelských účtů. Modul „Catalog“ nabízí správu produktů.

5.2 Modul

Fork CMS je modulární systém a většina požadavků na nějakou funkčnost je řešena vytvořením nového modulu, nebo přizpůsobením již existujícího modulu. Za základní zobrazovací prvky modulu můžeme považovat Widgety a Bloky. Moduly jsou psány v návrhovém vzoru MVC, kdy „Model“ se nachází v adresáři Engine, „View“ v adresáři Templates a „Controller“ v adresářích Actions a Widgets. V této diplomové práci je tvořen modul s názvem Orders.

5.2.1 Obecná struktura modulu

Základní struktura a umístění modulů jsou poměrně jednoduché. Soubory modulů se nachází v adresáři v src. V adresáři src se modul rozděluje do adresářů Frontend a Backend.

V adresáři Frontend můžeme najít adresáře Cache, Core, Files, Modules, Themes. Do adresáře Cache jsou ukládány soubory pro cachování šablon frontendu, pro modul zde není nic potřeba přidávat, adresář používá cachovací systém Forku. V adresáři Core se nacházejí třídy Jádra systému, které se mnohdy používají v modulech, ale pro vlastní modul sem ve většině případů nic nedáváme. Adresář Files je adresářem, ve kterém se uchovávají soubory nahrané pomocí modulu a taky se na ně odkazuje v šablonách v případě, že je soubor určen ke stažení. Adresář Themes obsahuje šablony webu, není zde potřeba ukládat soubory modulu. Do Themes nahráváme šablony modulu pouze, pokud je potřeba je přetížit. Zásadním adresářem pro moduly je adresář Modules, ve kterém se nacházejí adresáře modulů, které nesou jejich název. Například modul Orders zde bude mít adresář pojmenovaný Orders. V adresáři konkrétního modulu se mohou nacházet adresáře Action, Widgets, Ajax, css, js, Layout, Engine. V adresáři Actions jsou třídy, které lze chápat jako actions u controlleru s rozdílem, že nemají definovanou „routu“, ale jsou prvkem modulu, označovaný jako block. V adresáři Widget jsouobdobné třídy jako v Adresáři Action, s rozdílem, že jsou prvkem označovaným jako widget. Do adresáře Layouts

se ukládají šablony pro widgety a bloky. V Adresáři Engine jsou uloženy Modely a různé pomocné třídy. V adresáři Ajax jsou třídy volané Ajaxovým voláním javascriptu, který je většinou ukládán v adresáři js. V adresáři css jsou ukládány kaskádové styly, ale není to úplně běžné, jelikož každý vzhledový šablonový balíček má své vlastní styly a mnohdy není žádoucí, aby jiné styly ovlivňovaly vzhled. Většinou se totiž styl načte až po načtení konkrétního blocku, nebo widgetu modulu, takže tyto styly mnohdy rozhodí celkový vzhled po nějaké úpravě obsahu v administraci.

Adresář Backend má s drobnými rozdíly obdobnou strukturu, jako adresář Frontend, ale mnohdy je obsáhlejší. Nachází se zde adresáře Actions, Ajax, Command, DependencyInjection, Engine, EventSubscriber, Installer, Js, Layout, Resources. V adresáři Action jsou třídy označované jako „controllery“, které spravují konkrétní backendové stránky jako výpis, vytvoření a editace. Do adresáře Command se ukládají třídy, které registrují příkazy pro příkazovou řádku a pouštějí potřebné scripty. DependencyInjection obsahuje třídu, která definuje cestu ke konfiguračnímu souboru pro Dependency Injection. Adresář Engine obsahuje stejně jako na frontendu třídy načítající obsah z databáze a pomocné třídy. Adresář EventSubscriber obsahuje třídy a metody, které jsou volány při vyvolaných událostech. Adresář Layout obsahuje šablony backendu, adresář Resources obsahuje konfigurační soubor pro DI. Adresář Installer, patří mezi důležitější adresáře, nachází se zde instalační soubor modulu spolu se soubory obsahující překlady a SQL příkazy.

Tuto strukturu není zcela nutné dodržovat a je možné definovat vlastní adresáře, nicméně pro přehlednost a funkčnost všech funkcí Forku je doporučeno tuto popsanou strukturu dodržovat. Pro správný chod funkcí Forku je vhodné minimálně udržet strukturu adresářů Installer, Actions, Widgets a Layout.

5.3 Use Case diagram

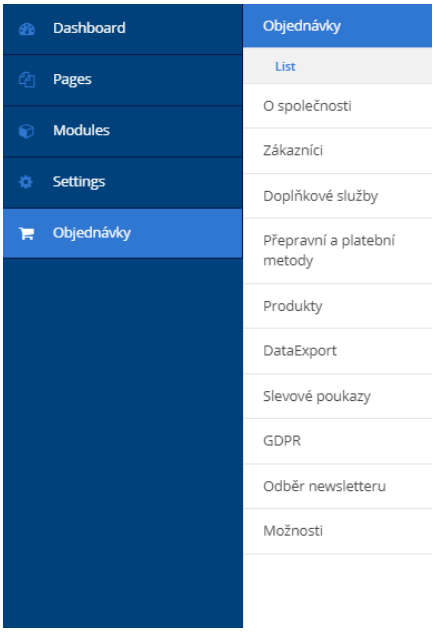
Use Case diagram, který zobrazuje chování systému (modulu) z pohledu rolí je přiložen na konci této diplomové práce, jako příloha D. Jeho účelem je popsat funkcionalitu modulu, tedy co má modul umět.

5.4 Modul Orders – Backend

Po instalaci modulu do systému se do první úrovně navigace administrace přidá prvek „Objednávky“. Většina běžných modulů je zařazena do navigace pod prvek „Modules“, kde se na druhé úrovni navigace nachází prvek s názvem modulu. Jelikož je tento modul obsáhlý

a je očekávané, že se bude jednat o modul s častým přístupem, tak bylo rozhodnuto, že přístup k modulu bude již z první úrovně navigace.

Po kliknutí na navigační prvek „Objednávky“ se dostaneme do výpisu objednávek a v navigační části se zobrazí navigační prvky pro tento modul. Navigaci můžeme vidět na obrázku 8.



Obrázek 8 – Ukázka navigace ve forku pro modul objednávek. Zdroj: zpracování autora

V navigaci najdeme prvky „Objednávky“, „O společnosti“, „Zákazníci“, „Doplňkové služby“, „Přepravní a platební metody“, „Produkty“, „DataExport“, „Slevové poukazy“, „GDPR“, „Odběr newsletteru“ a „Možnosti“. V následujících podkapitolách budou tyto prvky popsány.

5.4.1 Objednávky

Jednou ze zásadních vlastností modulu je správa objednávek. Výpis objednávek je první stránkou, která se zobrazí při přístupu do modulu přes navigaci. Tabulka s objednávkami je vytvořena pomocí komponenty DataGridDB, která je součástí jádra systému Fork. Tato komponenta umí zobrazit data v tabulce, která lze různě řadit a stránkovat. Při vytváření instance třídy DataGridDB stačí zadat jako parametr konstruktoru SQL dotaz a pak jen vložit do šablony proměnou.

Pro možné vyhledávání v objednávkách, byl vytvořen filtr na jméno, e-mail a číslo objednávky. DataGridDB má inicializační vstup pouze jako jeden SQL dotaz a parametry. Ukázalo se, že neumožňuje manipulaci s daty nad databází, ale pouze nad načtenými daty. Potřeba je ale prohledat všechny objednávky za krátký čas. Alternativou by mohla být komponenta DataGridArray která má vstup pole s daty. Data by se načetly z databáze mimo DataGrid

a následně by byly filtrovány přímo v action. Toto řešení není správné, jelikož objednávek může být velké množství, mohou vyčerpat všechnu dostupnou paměť a načtení skončí chybou. Proto se jako správná možnost jeví dynamické sestavení SQL dotazu.

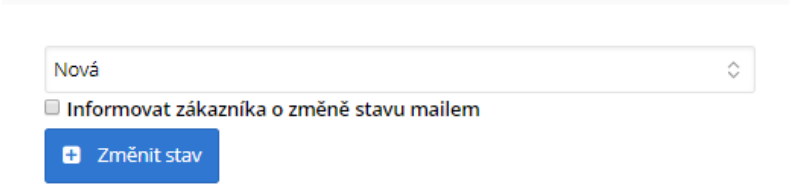
V následující ukázce kódu, jsou dvě metody z action třídy Index, které skládají dotaz pro DataGridDB.

```
private function buildQuery($parameters = null)
{
    $setWhere = true;
    $this->sqlQuery = "
SELECT i.id, i.order_number AS number, concat(oa.name, ' ', oa.surname) AS name,oa.email,
i.currency, UNIX_TIMESTAMP(i.created_on) AS created_on, concat_ws( ' ',
i.final_sum_without_vat + i.final_vat, i.currency) as price , os.title AS status
                    FROM orders AS i
                    JOIN order_address AS oa ON oa.id = i.order_address_id
                    JOIN order_status AS os ON os.id = i.order_status_id";

    if (isset($parameters))
    {
        if (isset($parameters['name']) && trim($parameters['name']) != "")
        {
            $this->sqlQuery .= $this->returnWhere($setWhere);
            $this->sqlQuery .= " CONCAT(oa.name,' ',oa.surname) LIKE '%".
$parameters['name'] ."%' ";
           // $this->queryParameters = $parameters['name']; //not working with LIKE %%
        }
        if (isset($parameters['email']) && trim($parameters['email']) != "")
        {
            $this->sqlQuery .= $this->returnWhere($setWhere);
            $this->sqlQuery .= " oa.email LIKE  '%". $parameters['email'] ."%' ";
           // $this->queryParameters = $parameters['email'];
        }

        if (isset($parameters['number']) && trim($parameters['number']) != "")
        {
            $this->sqlQuery .= $this->returnWhere($setWhere);
            $this->sqlQuery .= " i.order_number LIKE  '%". $parameters['number'] ."%' ";
            //$this->queryParameters = $parameters['number'];
        }
    }
}

private function returnWhere(&$bool)
{
    if ($bool)
    {
        $bool = false;
        return ' WHERE ';
    } else
    {
        return ' AND ';
    }
}
```

5.4.2 Detail objednávky

Kliknutím na tlačítko detail, se dostaneme na výpis všech potřebných informací týkající se objednávky. Nachází se zde, číslo objednávky, fakturační adresa, dodací adresa, zvolená

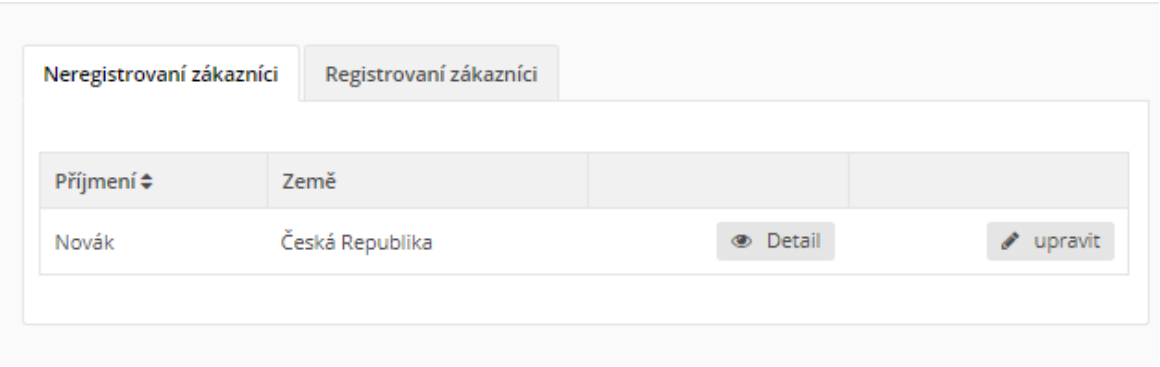
přepravní metoda, zvolená platební metoda, seznam koupených služeb, seznam koupených produktů katalogu a celkové shrnutí. V detailu objednávky se nastavuje i stav aktuálně zobrazené objednávky. V pravém rohu se nachází formulář, ve kterém lze vybrat stav. O změně stavu lze informovat zákazníka zaškrtnutím checkboxu „Informovat zákazníka o změně stavu mailem“ a následným kliknutím na tlačítko „Změnit stav“. O zobrazení detailu a zpracování formuláře pro změnu stavu objednávky se stará action ViewOrder.php a šablona ViewOrder.twig.php.



Obrázek 9 – Formulář změny stavu objednávky. Zdroj: zpracování autora

5.4.3 Zákazníci

Výpis zákazníků se nachází v sekci modulu pod prvkem navigace „Zákazníci“. Výpis zákazníků je vypsán pomocí komponenty Forku DataGridDB a je rozdělen do dvou tabulek. Jenda vypisuje neregistrované zákazníky, jedná se v podstatě o seznam vytvořený z fakturačních údajů. Druhá tabulka vypisuje registrované uživatele. Tento výpis je zpracováván action Třídou Customers a šablonou Customers.html.twig.



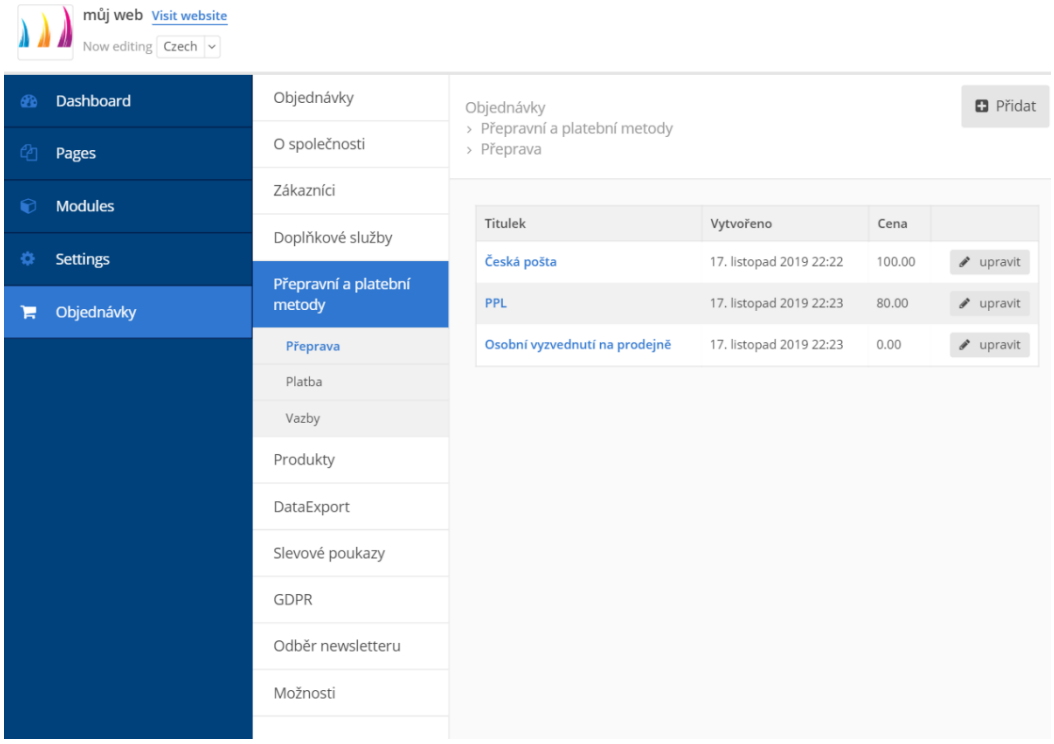
Obrázek 10 – Výpis zákazníků. Zdroj: zpracování autora

Detail zákazníka obsahuje podrobnější informace, jako je adresa a seznam objednávek. Jelikož neregistrovaní zákazníci nemají vytvořený účet, tak vytvořené objednávky nemají vazbu ke konkrétnímu uživateli. Výpis objednávek je tedy zprostředkován SQL dotazem na objednávky obsahující stejnou e-mailovou adresu. Výhodou je, že i když zákazník vytvoří několik objednávek bez přihlášení, tak můžeme zjistit jeho přibližnou historii objednávek.

Detail objednávky je zpracováván action ViewOrder a šablonou ViewOrder.html.twig.

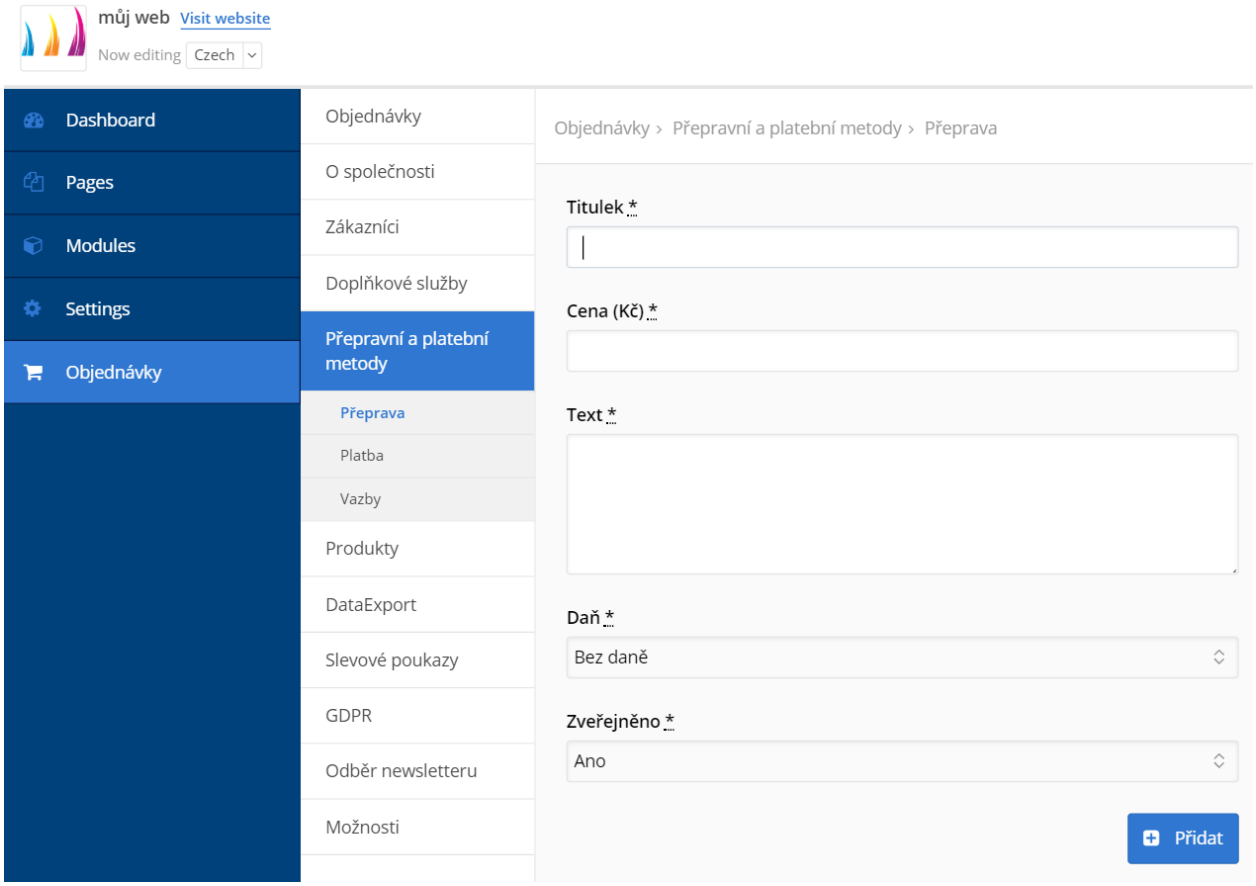
5.4.4 Přepravní a platební metody

Správa platebních metod se nachází v navigaci Objednávky → Přepravní a platební metody. Nachází se zde tři části, což jsou Přeprava, Platba a Vazby.



Obrázek 11 – Přepravní a platební metody. Zdroj: zpracování autora

Stránka s přepravou vypisuje vytvořené přepravní metody, jako příklad byly vytvořeny metody „Česká pošta“, PPL a „Osobní vyzvednutí na prodejně“. Přepravní metody je možné vytvářet kliknutím na tlačítko přidat, v pravém horním rohu. Již vytvořené přepravní metody lze upravovat kliknutím na tlačítko upravit u konkrétní přepravní metody vypsané v tabulce. Každá platební metoda má titulek, cenu, text, daň a možnost zobrazit, nebo nezobrazit.



Obrázek 12 – Formulář pro vytvoření platební metody. Zdroj: zpracování autora

Správu přepravy zpracovávají Action:

- ShippingMethods.php,
- EditShippingMethod.php,
- AddShippingMethod.php.

Šablony:

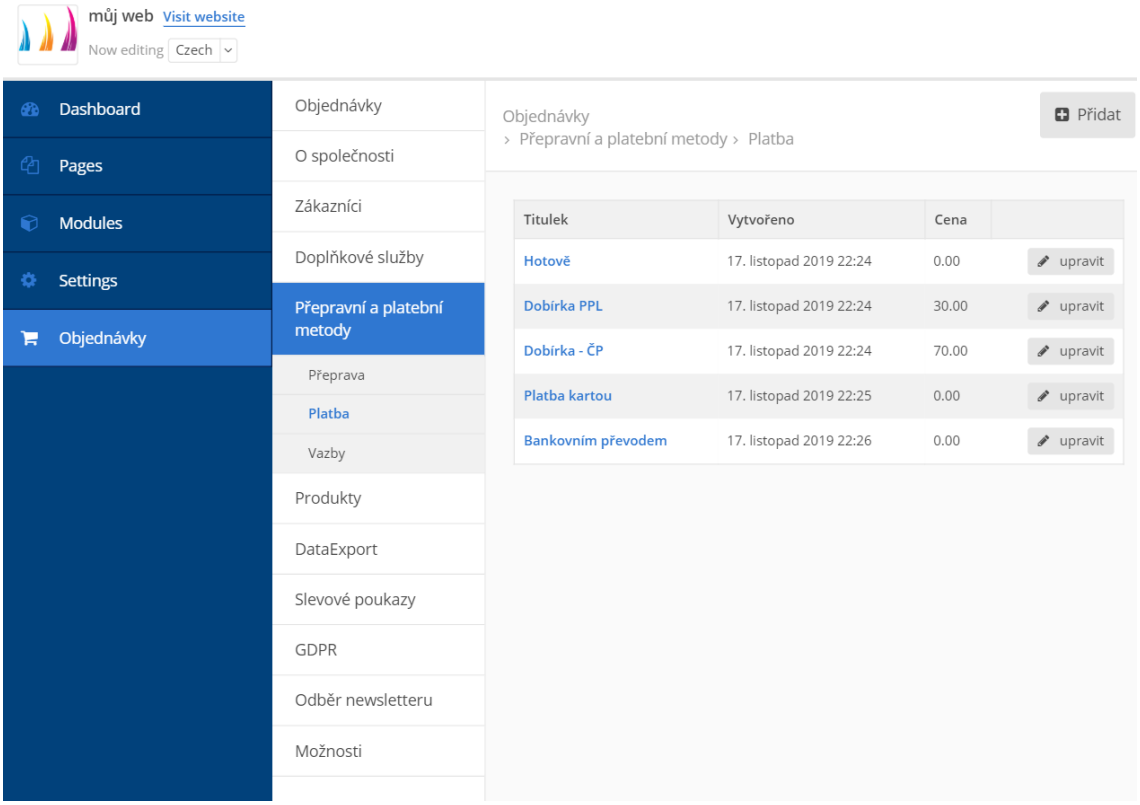
- ShippingMethods.html.twig,
- EditShippingMethod.html.twig,
- AddShippingMethod.html.twig.

Data přepravních metod se ukládají do databáze, a to do tabulky *orders_shipping_methods*.

```
CREATE TABLE IF NOT EXISTS `orders_shipping_methods` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `published` enum('N','Y') COLLATE utf8mb4_bin NOT NULL DEFAULT 'Y',
  `language` VARCHAR(5) COLLATE utf8mb4_bin NOT NULL,
  `title` VARCHAR(255) COLLATE utf8mb4_bin NOT NULL,
  `price` DECIMAL(10,2) DEFAULT NULL,
  `currency_id` INT(11) NULL DEFAULT NULL,
  `vat_id` INT(11) NULL DEFAULT NULL,
  `text` text COLLATE utf8mb4_bin,
  `created_on` datetime NOT NULL,
  `sequence` INT(11) NOT NULL,
```

```
  PRIMARY KEY (`id`, `language`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AUTO_INCREMENT=1;
```

Správa platebních metod, tedy výpis, vytvoření a editace platby jsou téměř identické s přepravou. Rozdíly najdeme až v použití na frontendové části modulu.



Obrázek 13 – Výpis platebních metod. Zdroj: zpracování autora

Správu platby zpracovávají action:

- PaymentMethods.php,
- EditPaymentMethod.php,
- AddPaymentMethod.php.

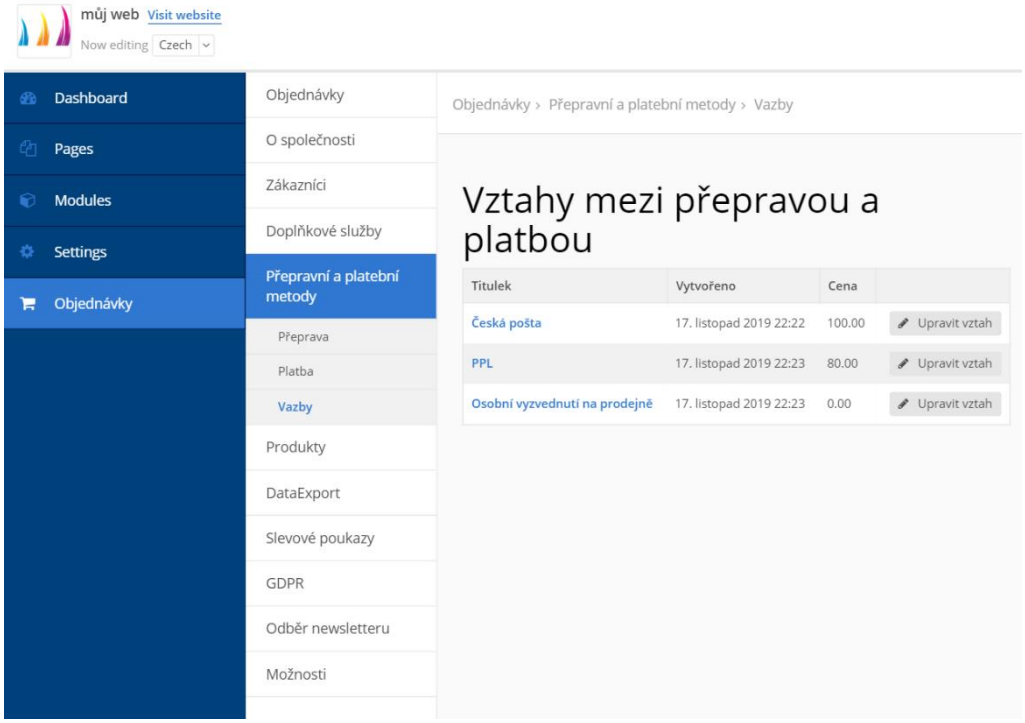
Šablony:

- PaymentMethods.html.twig,
- EditPaymentMethod.html.twig,
- AddPaymentMethod.html.twig.

Data platebních metod se ukládají do databáze, a to do tabulky *orders_payment_methods*.

```
CREATE TABLE IF NOT EXISTS `orders_payment_methods` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `published` enum('N','Y') COLLATE utf8mb4_bin NOT NULL DEFAULT 'Y',
  `language` VARCHAR(5) COLLATE utf8mb4_bin NOT NULL,
  `title` VARCHAR(255) COLLATE utf8mb4_bin NOT NULL,
  `price` DECIMAL(10,2) DEFAULT NULL,
  `currency_id` INT(11) NULL DEFAULT NULL,
  `vat_id` INT(11) NULL DEFAULT NULL,
  `text` text COLLATE utf8mb4_bin,
  `created_on` datetime NOT NULL,
  `sequence` INT(11) NOT NULL,
  `catalog_order_payment_log_id` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`id`, `language`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Mezi přepravou a platbou existují určité vazby. Například rozdíl v cenách za určité přepravní služby, kdy například příplatek za dobírku u dopravce A je 30 korun a u dopravce B 40 korun. Dalším příkladem může být, kdy různé způsoby přepravy, jako donáška kurýrem, doručení na pobočku, nebo i osobní převzetí na prodejně umožňují jiné způsoby platby. Z těchto důvodu je tu možnost vytvoření vazby.



Obrázek 14 – Vztahy mezi přepravou a platbou. Zdroj: zpracování autora

Na stránce „Vazby“ je připraven výpis přepravních metod, kliknutím na tlačítko „Upravit vztah“ u přepravní metody se dostaneme na stránku, kde je možné spárovat tuto přepravní metodu s platebními metodami. Platební metodu přiřadíme přetažením prvku ze sekce „Platební metody“ do sekce „Přiřazené platební metody“. Pro funkci drag-and-drop je použita knihovna jquery-sortable.js.

Správu platby zpracovávají action:

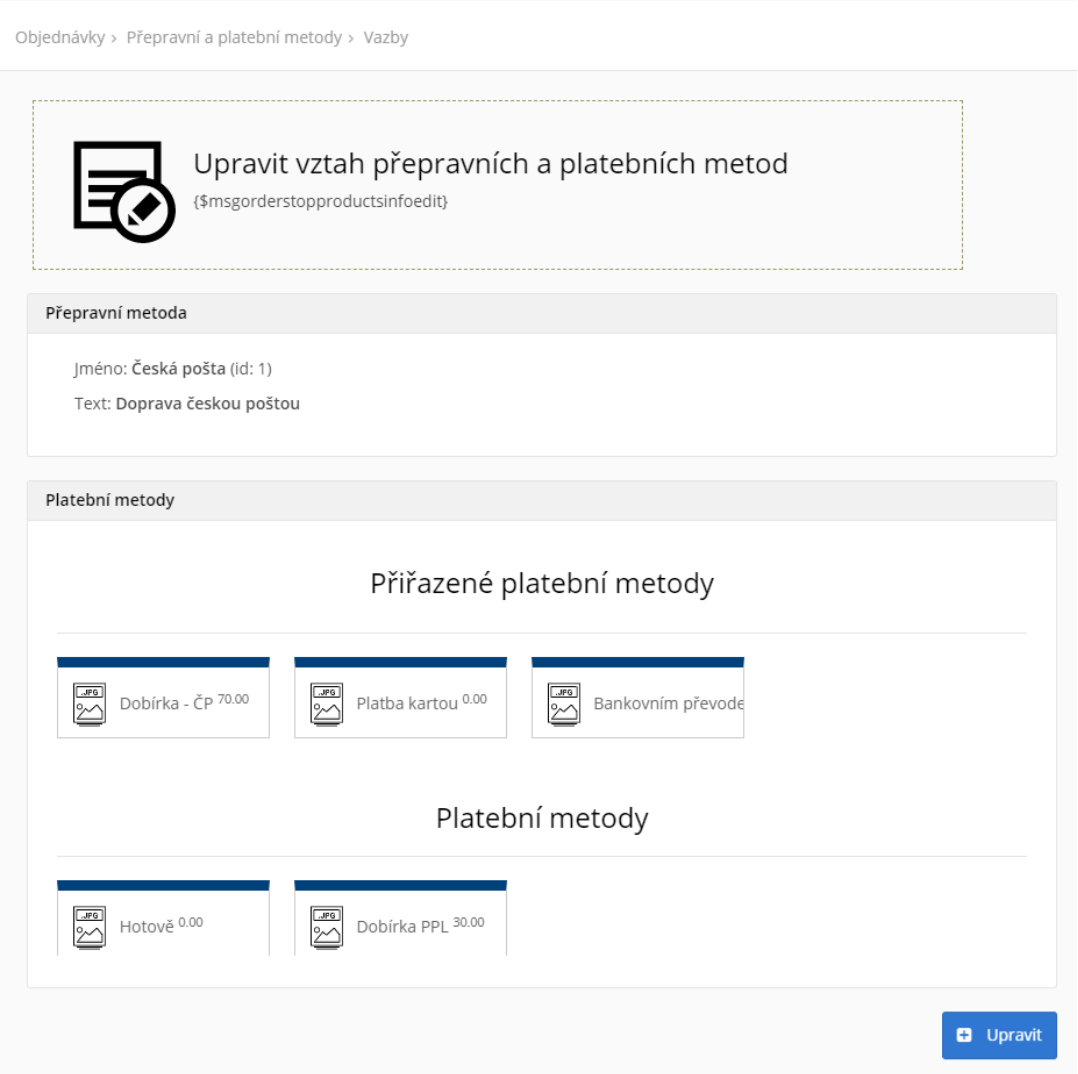
- EditShippingPaymentMethodRelations.php,
- ShippingPaymentMethodRelations.php.

Šablony:

- EditShippingPaymentMethodRelations.html.twig,
- ShippingPaymentMethodRelations.html.twig.

Data vazeb mezi přepravními metodami a platebními metodami se ukládají do databáze, a to do tabulky *catalog_orders_shipping_payment_relations*.

```
CREATE TABLE IF NOT EXISTS `catalog_orders_shipping_payment_relations` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `payment_method_id` INT(11) NOT NULL,
  `shipping_method_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci AUTO_INCREMENT=1;
```



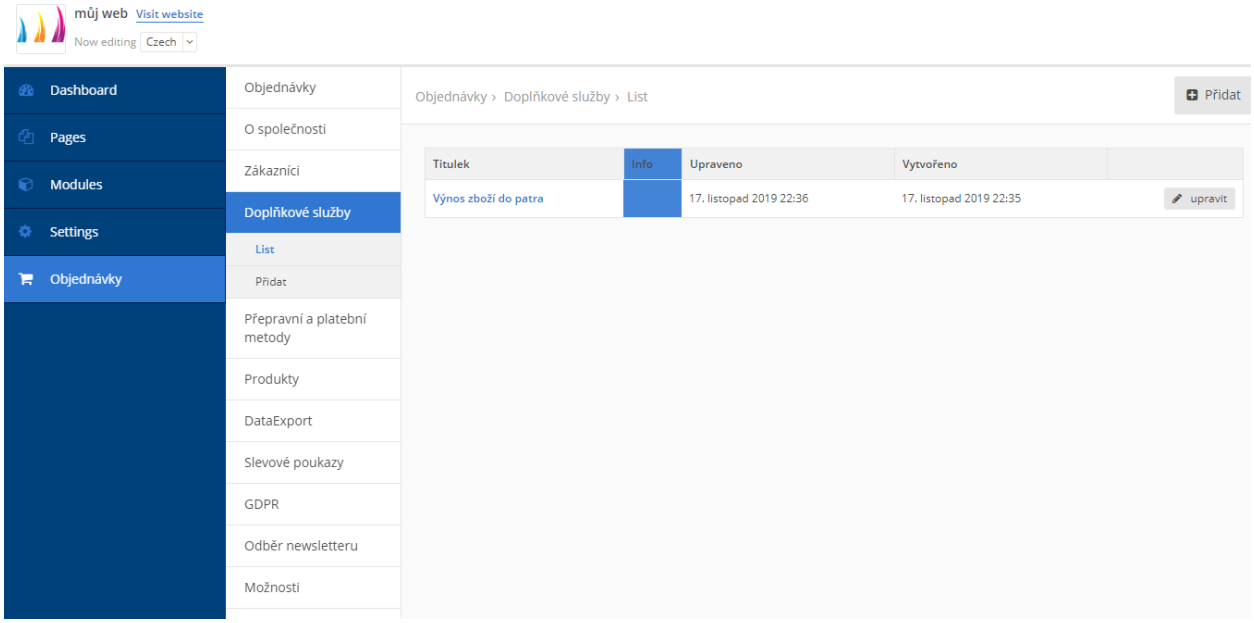
Obrázek 15 – Úprava vztahu mezi přepravou a platbou. Zdroj: zpracování autora

5.4.5 Produkty

Produkty nejsou součástí tohoto modulu, ale součástí modulu Catalog, který není vyvíjen v této diplomové práci. Jedná se pouze o propojení s modulem a využití prvku „Produkt“ z modulu Catalog. V případě kliknutí na prvek menu „Produkty“ bude uživatel přesměrován na modul Catalog. Objednávka má vazbu na produkty pouze v případě, že jsou součástí objednávky a byly zvoleny zákazníkem. V tomto případě jsou součástí objednávky ID produktů. Aby objednávku šlo odeslat, existence produktů není potřeba, detailnější popis vytvoření objednávky je popsán ve frontendové části.

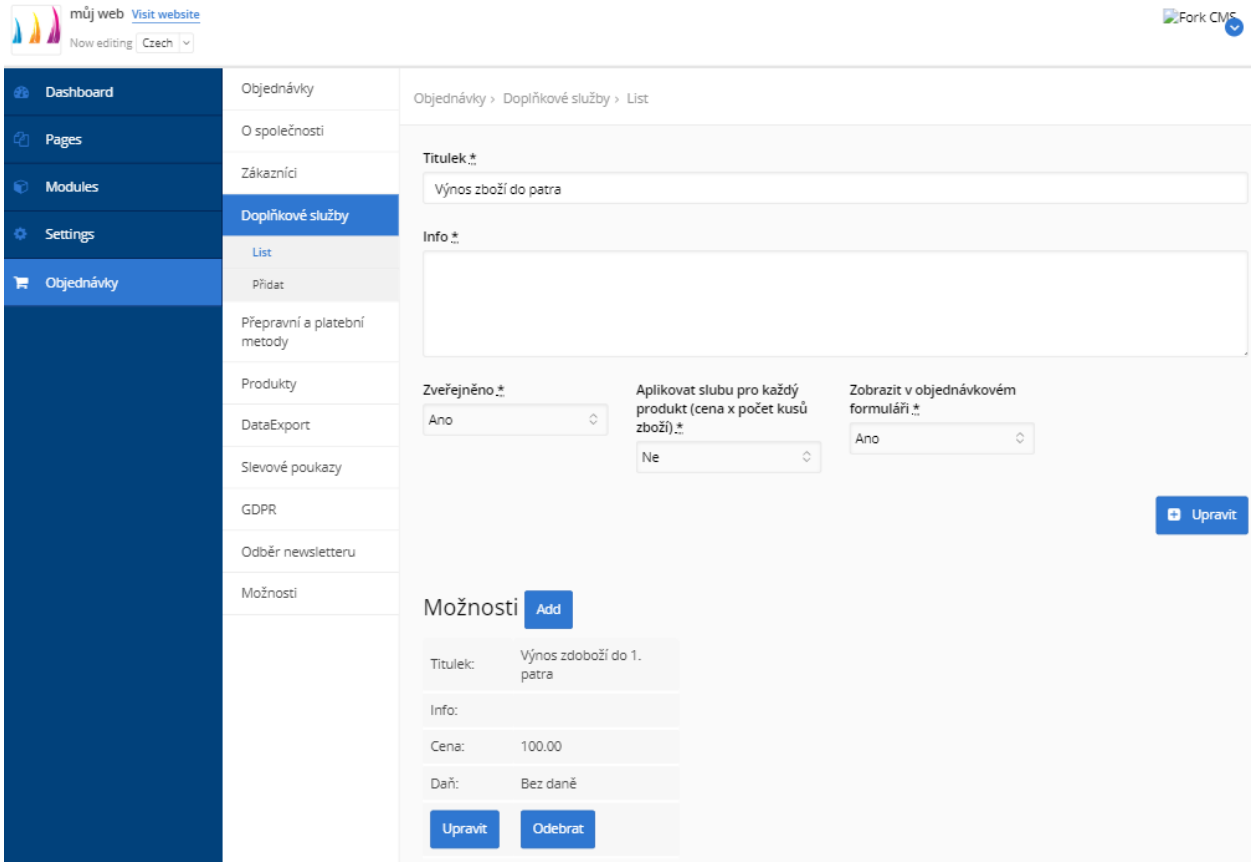
5.4.6 Doplňkové služby

Jako doplňkovou službu můžeme například chápat výnos do patra u prodejce nábytku, nebo úpravu, či zabalení zboží za příplatek. Správu služeb najdeme v Objednávky → Doplňkové služby.



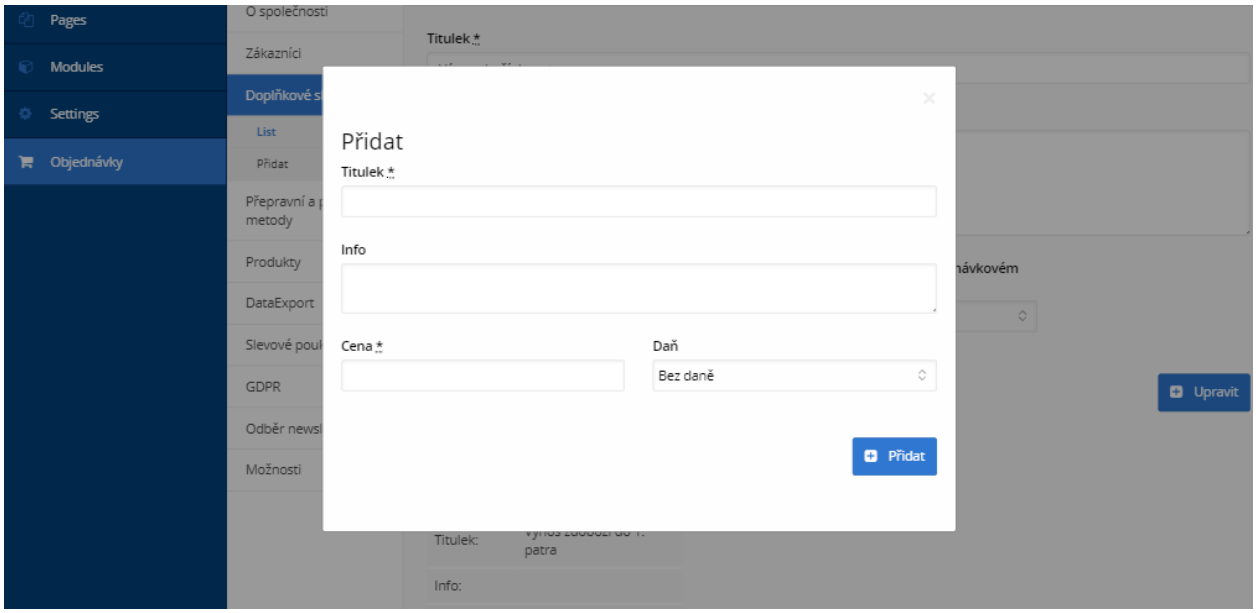
Obrázek 16 – Doplňkové služby. Zdroj: zpracování autora

Při vytvoření, nebo editaci služby nastavujeme titulek, informační text, vlastnosti, zda je služba zveřejněná, zda se má aplikovat na každý produkt, zda se má zobrazovat v objednávkovém formuláři, a hlavně zde přidáváme možnosti. Jedna z věcí, která uživatele napadne je, jaký je rozdíl mezi „Zveřejněno“ a „Zobrazit v objednávkovém formuláři“. Při použití správcem systému se může zdát, že se obě vlastnosti chovají stejně, ale ve skutečnosti mají konkrétní využití. Pokud je služba nezveřejněná, nelze ji vybrat v objednávce a ani nemůže být objednávkou odeslána. Pokud zvolíme, že nechceme zobrazovat službu v objednávkovém formuláři, tak služba nebude ve formuláři zobrazena, ale může být za nějaký podmínek automaticky vybrána a v objednávce odeslána. Příklad použití: Naším produktem je služba zpracování nějakých dat, nebo vytvoření nějakého posudku. Na webu vyplním formulář a po přístupu do fáze dokončení objednávky již nechci mít žádný výběr služeb, ale mít již zvolenou službu za vyhodnocení formuláře.



Obrázek 17 – Editace doplňkové služby. Zdroj: zpracování autora

Důležitým prvkem služby jsou možnosti. Možnost je možné zadávat až po založení služby. Součástí objednávky musí být alespoň jedna možnost, jinak služba nemůže být součástí objednávky. Jako příklad se dá zase uvést „Výnos zboží do patra“, kdy pater pro výnos může být více, ale dopravce si za různý počet poschodí určuje jiné částky. Tedy můžeme mít například výnos od prvního do druhého patra za 500 korun, od třetího do čtvrtého za 1000 korun, od pátého a výš 2000 korun. V tomto případě vytvoříme službu „Výnos zboží do patra“ a přidáme pro každé rozmezí jednotlivou možnost. Kliknutím na tlačítko přidat možnost vyskočí modální okno, kde je možné zadat název možnosti, informační text, cenu a zvolit daň.



Obrázek 18 – Modální okno pro přidání možnosti služby. Zdroj: zpracování autora

Zpracování přidání možnosti je prováděno přes AJAX. Po vyplnění formuláře a kliknutím na tlačítko přidat, se data odešlou pomocí javascriptu na server, ten odešle informaci o zpracování a následně javascript aktualizuje výpis možností.

V následujícím uvedeném kódu je ukázka javascriptové funkce, která je volána pro odeslání dat na server.

```
sendServiceOption: function (formData, callback) {
    $.ajax({

        type: 'POST',
        data: $.extend({}, {
            fork: {
                module: 'Orders',
                action: 'ServiceOption',
            }
        },
                formData
                ),
        success: function (response) {


            if (typeof callback == "function") {
                callback(response.data);
            }

        }
    });
},
```

Objekt Fork, který posíláme přes AJAX nese název modulu a název Action na kterou budou data odeslána. Odeslání dat probíhá metodou Post. Tento způsob použití AJAXU je definován

redakčním systémem Fork, ale je možné tento způsob obejít a použít vlastní způsob. Pro zachování jednoty s jinými moduly bylo použito řešení navržené tvůrci systému Fork.

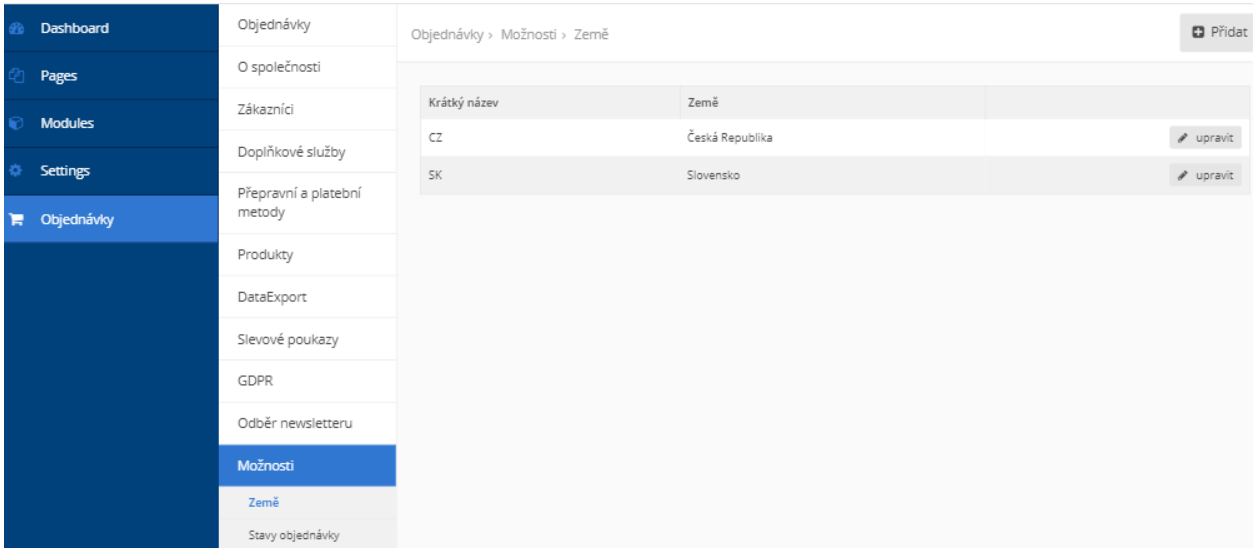
Data jsou odeslána třídě ServiceOption v adresáři Ajax, která provede potřebnou operaci a odešle zpět odpověď.

5.4.7 Možnosti a nastavení

Část modulu „Možnosti“ je určena ke konfiguraci modulu uživatelem systému, jsou to možnosti jako nastavování číselníků, e-mailů, parametrů platební brány a podobně. Konfiguraci, kterou není potřeba často upravovat a nechceme ji zpřístupňovat běžnému uživateli je v „Settings“, kde je nastavení i pro ostatní moduly.

Možnosti země

V možnostech se nachází definice zemí, pro které je možné vytvořit objednávku na frontendu. Jedná se o výběrový prvek objednávkového formuláře, který dává na výběr volbu země u doručovací adresy. Přidání je velmi jednoduché, kliknutím na tlačítko přidat se načte formulář s již předpřipravenými zeměmi. Stačí tedy vybrat a uložit.



Obrázek 19 – Nastavení země. Zdroj: zpracování autora

Možnosti stavů objednávky

Je běžně zaběhnutá praxe, že každá objednávka má nějaký stav. Daly by se definovat 3 základní stavy: nová, dokončená a storno. Mnohdy je ale proces zpracování a vyřízení objednávky komplikovanější a z toho důvodu je vytvořena možnost definovat vlastní stavy. V základu jsou předpřipravené stavy: nová, přijatá, vyřízená, storno. Tyto stavy můžeme smazat, upravit nebo

přidat nový. Jediná věc, co je potřeba, je mít vždy definovaný počáteční stav. To je stav, který bude mít objednávka po odeslání zákazníkem. Výchozí objednávku nastavíme tak, že ve formuláři zaškrtneme možnost, že je výchozí a uložíme.

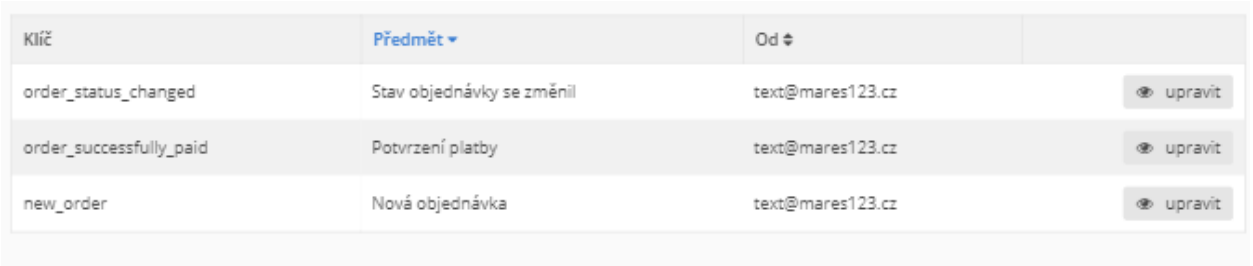
Obrázek 20 – Stavy objednávky. Zdroj: zpracování autora

Stav „výchozí“ může být pouze jeden, a to poslední nastavený jako výchozí. O jednotnost se stará přímo metoda v modelu, která v případě nastavení stavu jako výchozího, resetuje před uložením tuto hodnotu u všech existujících stavů.

```
public function updateStatus($data)
{
    if ($data['is_default'])
    {
        $this->resetDefaultStatus();
    }
    return $this->database->update('order_status', $data, ' id = ?', $data['id']]);
}
```

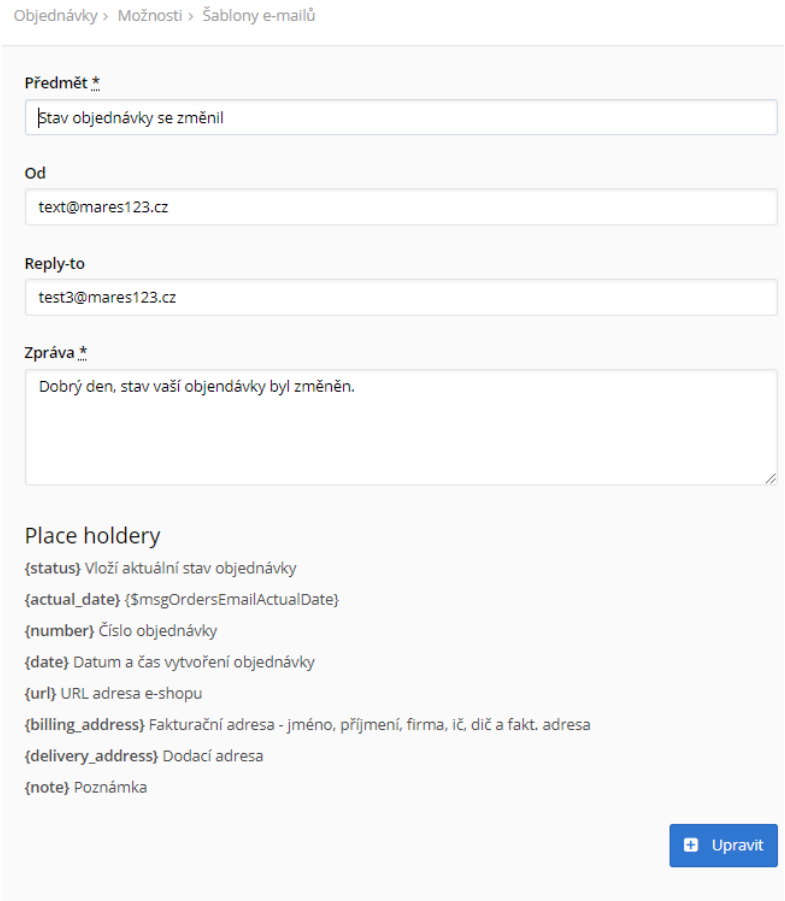
Možnosti šablon e-mailů

Modul objednávek umí zasílat informační e-maily z určitých událostí, proto je tu i správa jejich šablon. Předdefinované jsou 3 zprávy u kterých máme možnost upravovat jejich obsah.



Obrázek 21 – Seznam e-mailových šablon. Zdroj: zpracování autora

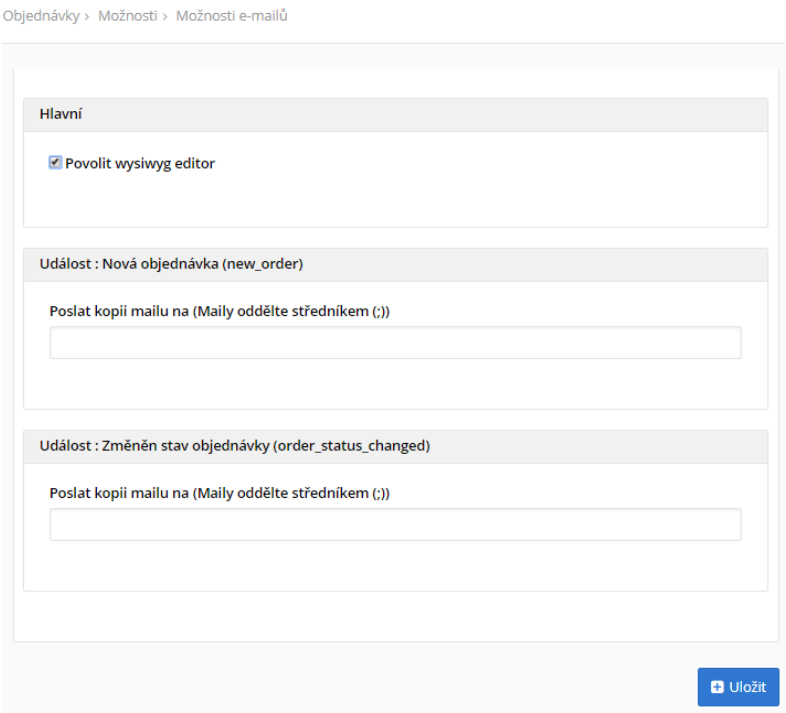
V editaci šablony je možné nastavit předmět, e-mail odesílatele, e-mail pro odpověď a obsah zprávy. Ve zprávě lze používat „place holdery“, které budou dynamicky nahrazeny konkrétním obsahem. Seznam placeholderů je uveden pod textovým polem pro zprávu.



Obrázek 22 – Úprava e-mailové šablony. Zdroj: zpracování autora

Možnosti e-mailů

Na další stránce v možnostech e-mailů je možné povolit wysiwyg editor pro editaci zprávy a nastavit si e-mailové adresy pro zasílání kopií zpráv. Například pro e-maily s klíčem „new_order“ stačí uložit e-mailovou adresu, nebo více e-mailových adres oddělených středníkem a následně po vyvolání události pro odeslání emailu se odešle emailová zpráva jak originálnímu příjemci, tak i na adresy uložené v nastavení.



Obrázek 23 – Nastavení e-mailů. Zdroj: zpracování autora

5.4.8 Nastavení

Do nastavení modulu se dostaneme přes navigaci „Nastavení“, „Moduly“ a „Objednávky“.

Lze zde nastavit počáteční číslo objednávky. Tato hodnota má účinnost do doby, než je vytvořena první objednávka. Pokud již existuje objednávka, změna této hodnoty nebude mít žádný vliv. Dále je možné povolit komentář v objednávce, odběr newsletteru v objednávce, zobrazení daně v souhrnu objednávky, povolení odesílání objednávky bez uživatelského účtu.

Dále zde můžeme přizpůsobit výpis objednávek a zákazníků. Například zde můžeme nastavit počet záznamů na stránku a zvolit si sloupce, které chceme zobrazovat.

5.4.9 Instalátor

V backendové části modulu se nachází sekce, kterou využívá instalační mechanizmus systému Fork k instalaci modulu. Jedná se o adresář „Installer“, který by měl být součástí každého modulu. Adresář obsahuje soubor Instaler.php a adresář Data. V adresáři Data se nachází soubor Instal.sql, který obsahuje databázové scripty a soubor locale.xml který obsahuje překlady modulu. Soubor Installer.php obsahuje třídu Installer, která dědí z třídy Backend\Core\Installer\ModuleInstaller. Tato třída Installer musí mít definovanou metodu „install“, která je volána systémem Fork při instalaci modulu.

Takto popsaná struktura je používána u většiny modulů, ale zároveň není úplně ideálním řešením. Například zde, ani přímo v redakčním systému Fork není definován žádný aktualizující nebo mazací mechanizmus. Proto z důvodu potřeby aktualizovat již běžící projekt, vznikly „deltascripty“. V adresáři Data se nachází adresář Delta, ve kterém se nacházejí soubory s pojmenováním v jednotném tvaru.

```
000_nazev_popisujici_upravu.sql
```

Tento soubor obsahuje SQL scripty pro aktualizaci databáze. Tyto soubory se nenačítají automaticky, pro spuštění při instalaci, je potřeba definovat jejich spuštění v třídě Install. Příklad přidání delta scriptu do třídy Install:

```
//deltascripts
$this->importSQL(dirname(__FILE__) . '/Data/delta/001_gopay.sql');
```

Hlavním smyslem je, že pokud došlo ke změnám a je potřeba aktualizovat databázi, tak nemusíme hledat změny v hlavním instalačním SQL scriptu a v podstatě ručně zkopírujeme obsah scriptu a pustíme jej nad databází. Tento postup ušetří spousty práce a určitě by se dal dalším vývojem automatizovat.

Script uninstall.sql obsahuje příkazy pro odstranění modulu z databáze. Fork cms, neobsahuje žádnou odstraňovací funkci, proto z této potřeby vznikl tento script. Spuštění není umožněno modulem ani systémem, spuštění provádí vývojář sám. Příklad odinstalčního scriptu pro odebrání modulu z databáze:

```
DROP TABLE orders;
DROP TABLE order_associations;
DROP TABLE order_associations_module;
DROP TABLE catalog_order_items;
DROP TABLE order_address;
DROP TABLE order_shipping_address;
DROP TABLE orders_shipping_methods;
DROP TABLE orders_payment_methods;
DROP TABLE catalog_orders_shipping_payment_relations;
DROP TABLE order_status;
DROP TABLE order_allowed_countries;
DROP TABLE order_payment_log;
DROP TABLE order_subscription;

DELETE FROM `modules` WHERE `name` = 'Orders';
DELETE FROM `modules_extras` WHERE `module` = 'Orders';
DELETE FROM `modules_settings` WHERE `name` = 'Orders';
```

```
DELETE FROM `modules_tags` WHERE `module` = 'Orders';
DELETE FROM `search_modules` WHERE `module` = 'Orders';
DELETE FROM `backend_navigation` WHERE `label` = 'Orders';
DELETE FROM `backend_navigation` WHERE `url` LIKE '%orders/%';
```

5.5 Modul Orders – Frontend

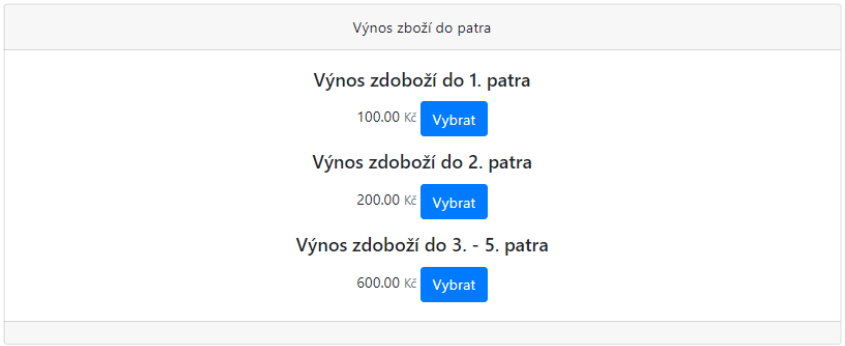
Frontendová část má obdobnou strukturu jako „backendová“, rozdíly jsou v použití Actions a Widgets. Actions spravují prvek modulu zvaný jako „block“, Widgets spravují prvky zvané jako „widgets“. Oba dva jsou to prvky, které v redakčním systému Fork přidáváme na stránky. Nejvýraznější rozdíl je v použití, block jde přidat pouze jedenkrát na stránku na které se žádný jiný block nenachází. Widget lze přiřadit na jakoukoliv stránku, a to kolikrát je potřeba.

Pro vzhled formulářů byl využit CSS framework Bootstrap 4.

5.5.1 Actions

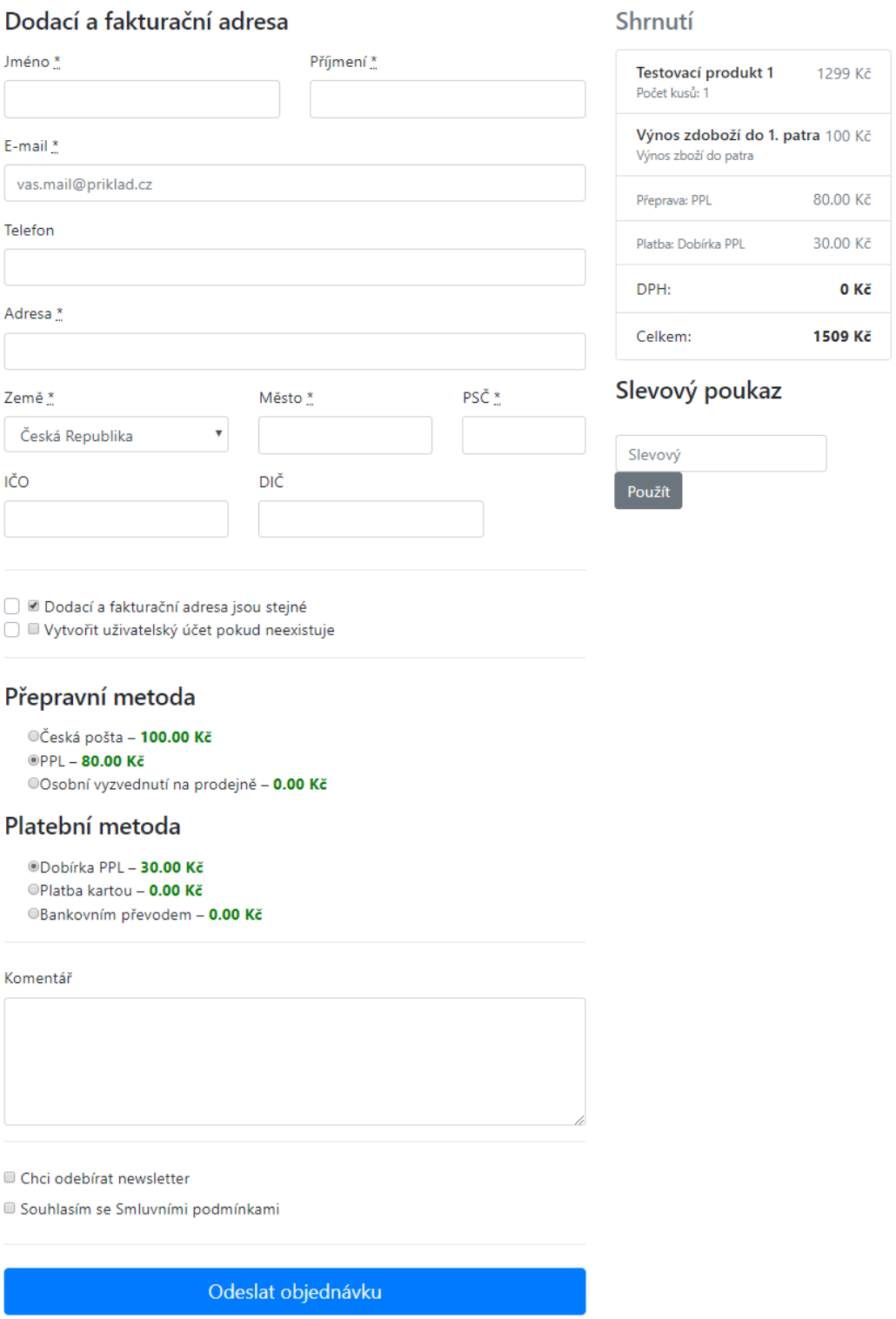
Základní action modulu je Order, ve které se nachází základní objednávací formulář. Formulář obsahuje vstupy pro fakturační údaje, pokud není přepravní adresa stejná, tak odškrtnutím pole „Dodací a fakturační adresa jsou stejné“ se objeví pole pro zadání druhé adresy. Následuje výběr z přepravních metod. Výběrem přepravní metody se pomocí AJAX odešle zvolená metoda na server, kde je uložena do session a načtou se platební metody pro zvolenou přepravní metodu. Dále se ve formuláři nachází pole pro komentář a checkbox pro odběr novinek, v administraci lze nastavit, aby se tyto prvky nezobrazovaly. Poslední checkbox je potvrzení souhlasu se zpracováním osobních údajů.

Další možností, kterou zde můžeme zobrazit jsou služby. Tato možnost se zobrazuje v případě, že jsou nějaké služby vytvořeny. Z každé služby lze zvolit pouze jednu možnost, opakovaný výběr stejné služby výběr ruší. Výběr služby je odeslán pomocí AJAX na server a uložen do session.



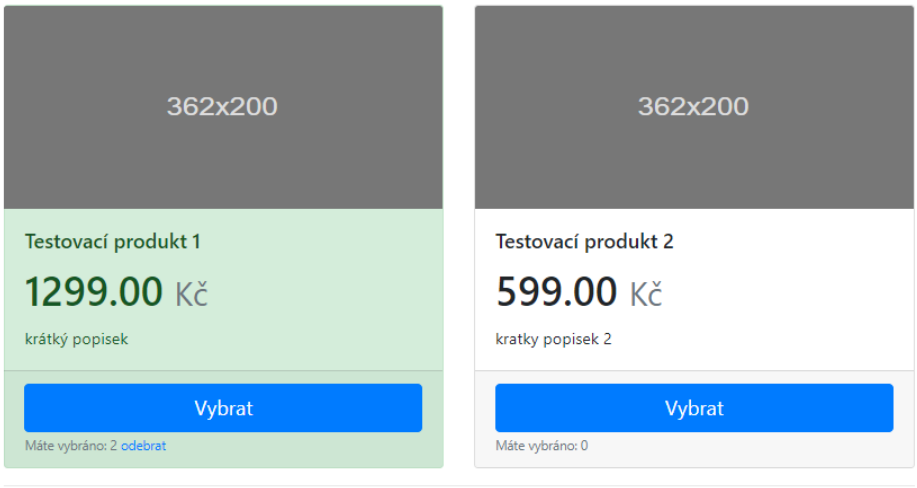
Obrázek 24 – Výběr služby. Zdroj: zpracování autora

V pravém sloupci, vedle formuláře se nachází shrnutí objednávky, kde jsou uvedeny ceny za služby, přepravu, platbu a celkový součet. Při změně přepravy, platby a služeb se shrnutí znovu načte pomocí AJAXU.



Obrázek 25 – Objednávkový formulář. Zdroj: zpracování autora

Action CatalogOrder obsahuje stejné funkce jako action Order, ale nabízí navíc možnost objednávky produktu modulu Catalog. Výběr označí vybraný produkt, výběrem již vybraného produktu se inkrementuje počet kusů produktu. Kliknutím na tlačítko odebrat, se zruší výběr produktu. Oproti službě, je možné zvolit více produktů. Při přidání i odebrání je volán AJAX, který uloží do session zvolený produkt, další AJAX volání zase aktualizuje shrnutí objednávky.



Obrázek 26 – Výběr produktů. Zdroj: zpracování autora

Action ProfileOrdersList slouží k výpisu objednávek uživatele. Tato část je napojena na modul „Profiles“, který není vyvíjen v této diplomové práci, ale je součástí základního balíčku redakčního systému Fork CMS. Po správné instalaci modulu a nastavení widgetů, můžeme na stránku pro uživatele přidat blok ProfileOrdersList, který zobrazí tabulku s objednávkami uživatele. Tyto objednávky musí být vytvořeny pod přihlášeným uživatelem. Uživatel zde má k dispozici datum vytvoření objednávky, číslo objednávky a stav objednávky.

Action ThanksPage se umisťuje na stránku, na kterou bude zákazník přesměrován po odeslání, nebo zaplacení objednávky. Stránka má funkci děkovné stránky, kde je poděkováno zákazníkovi za vytvořenou objednávku a zákazník je informován, zda objednávka byla úspěšně odeslána. Další funkce děkovné stránky je v měření konverzí objednávek použitím scriptů třetích stran.

Děkovná stránka má také v případě použití platební brány notifikační funkci. Je to stránka, na kterou jste přesměrování po provedení, nebo zrušení platby v platební bráně GoPay. Na základě dat poslaných z platební brány, je u objednávky změněn stav a je zákazníkovi zobrazena předpřipravená zpráva. Po zaplacení objednávky je vyvolána událost, která odešle e-mail zákazníkovi o úspěšném zaplacení. Obsah tohoto e-mailu je nastavitelný v administraci.

Administrátor má možnost nastavit zprávu, která bude na děkovné stránce zobrazována. Jedná se o výpis obsahového bloku, který bude načten dle nastavených id v nastavení. Obsahové bloky, jsou modulem systému Fork, tento modul není součástí této diplomové práce, jedná se pouze o využití z důvodu jednoduché správy obsahu uživatelem systému.

5.5.2 E-maily

Jednou z očekávaných vlastností objednávkového systému je, že při vytvoření objednávky je zaslán souhrn objednávky e-mailem. Proto v tomto modulu tato možnost nesmí chybět. Jak již bylo zmíněno, v administraci objednávek se nachází správa e-mailových šablon.

Aby bylo možné jednotně implementovat a jednoduše spravovat obsah e-mailů, byl navržen následující postup. V kódu, kde bude zpracován požadavek, ve kterém vznikne stav, při kterém je potřeba odeslat e-mailovou zprávu, umístíme metodu, která vyvolá událost a odešle potřebná data. Tato data budou zpracována třídou MailModel, který načte všechny šablony z databáze dle klíče. Klíč je znakový řetězec, pomocí kterého určujeme, pod jakou událost šablona spadá. Například pro novou objednávku máme klíč 'new_order'. S tímto klíčem můžeme mít vytvořenou šablonu s rekapitulací objednávky, s fakturou, poděkováním při zakoupení v nějaké marketingové akci, dodatečné informace v případě omezení služeb, nebo konání inventury a podobně. Ve stručnosti to znamená, že po odeslání objednávky, budou odeslány e-maily dle šablon, které mají klíč 'new_order'. Pro každou mutaci se definují vlastní šablony a odeslány jsou e-maily pouze v té mutaci, ve které byla událost vyvolána.

Tabulka, ve které jsou uloženy e-mailové šablony.

```
CREATE TABLE IF NOT EXISTS `order_mail_templates` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `key` varchar(255) NOT NULL COLLATE utf8mb4_bin,
  `subject` varchar(255) NOT NULL COLLATE utf8mb4_bin,
  `from` varchar(255) NOT NULL COLLATE utf8mb4_bin,
  `reply_to` varchar(255) NOT NULL COLLATE utf8mb4_bin,
  `message` text NOT NULL COLLATE utf8mb4_bin,
  `language` varchar(5) COLLATE utf8mb4_bin NOT NULL,
  PRIMARY KEY (`id`, `language`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
AUTO_INCREMENT=1;
```

V Action objednávky, v kódu po uložení objednávky je odeslána událost skrze třídu Events. Následující kód je volán na konci objednávky:

```
$this->events->sendEventNewOrder([
    'order' => $order,
    'summary' => $summary,
    'email' => $fields['email']
]);
```

Třída Events je obalovou třídou a zatím obsahuje tři metody sendEventNewOrder, beforeGetSummary a sendEventOrderPaid. Každá z metod odesílá událost, skrze třídu EventDispatcher která je součástí balíčku Symfony. Událost je instancí třídy GenericEvent, která je také součástí balíčku Symfony.

Následující kód, definuje třídu Events.

```
class Events
{

    /**
     *  @var $eventDispatcher \Symfony\Component\EventDispatcher\EventDispatcher
     */
    private $eventDispatcher;

    public function __construct($eventDispatcher)
    {
        $this->eventDispatcher = $eventDispatcher;
    }

    public function sendEventNewOrder($data)
    {
        $event = new GenericEvent($data);
        $event->setArgument('orderData', $data);
        $this->eventDispatcher->dispatch('order.send', $event);
    }

    public function beforeGetSummary($data)
    {
        $event = new GenericEvent($data);
        $event->setArgument('summaryData', $data);
        $this->eventDispatcher->dispatch('summary.before', $event);
    }

    public function sendEventOrderPaid($data)
    {
        $event = new GenericEvent($data);
        $event->setArgument('orderData', $data);
        $this->eventDispatcher->dispatch('order.paid', $event);
    }

}
```

Události, třída a metody kterou budou volány se definují v konfiguračním souboru service.yml.

V následujícím kódu je ukázka definice služby „fork.modules.orders.eventSubscriber.mail“ pro třídu EmailEventSubscriber. Argumenty služby jsou další služby, potřebné ve třídě EmailEventSubscriber, které jsou zprostředkované skrze Dependency Injection. Tags určují listener, událost a metodu, kterou bude událost pouštět.

```
fork.modules.orders.eventSubscriber.mail:
    class: Backend\Modules\Orders\EventSubscriber\EmailEventSubscriber
    arguments:
        - '@fork.modules.orders.mail model'
        - '@request_stack'
    tags:
        - { name: kernel.event_listener, event: order.send, method: onOrderSend }
        - { name: kernel.event_listener, event: order.status_changed, method:
onOrderStatusChanged }
        - { name: kernel.event_listener, event: order.paid, method: onOrderPaid }
```

Třída EmailEventSubscriber obsahuje metody, které jsou pouštěny při zachycení události. Tyto metody většinou pouštějí metody třídy MailModel pro odeslání e-mailové zprávy.

```
/**
 *
 * @param GenericEvent $event
 */
public function onOrderSend(GenericEvent $event)
{
    $data = $event->getSubject();
    $this->mailModel->sendNewOrderMail($data);
}
```

V následují ukázce, je metoda `sendOrderPaidMail` z třídy MailModel, která odesílá e-maily objednávky.

```
    /**
     *  Method send email to customer and copy to selected mail in admin options.
     *  parameters ['summary' => , 'order' => , 'email' => ]
     * @param array $orderData
     */
    public function sendNewOrderMail($orderData)
    {
        $key = 'new_order';
        $templatePath = '/Orders/Layout/Templates/Mails/NewOrder.html.twig';
        $email = $orderData['email'];

        $data = $this->orders->getEmailTemplates($key);
        if (!empty($data))
        {
            foreach ($data as $value)
            {
                $subject = isset($value['subject']) ? $value['subject'] : null;
                $from = isset($value['from']) ? $value['from'] : null;
                $replyTo = isset($value['reply_to']) ? $value['reply_to'] : null;
                $message = isset($value['message']) ? $value['message'] : null;
                $toDisplayName = null;
                $toEmail = $email;

                //finalize message fill placeholders
                $variables['message'] = $this->buildMessage($message, $orderData);
                $variables['settings'] = $this->settings;

                $this->sendMail($subject, $variables, $toEmail, $from, $replyTo,
$templatePath, $toDisplayName);

                // send copy
                $sendCopy = $this->prepareMails(isset($this-
>settings['email_new_order_copy_to']) ? $this->settings['email_new_order_copy_to'] :
null);
                foreach ($sendCopy as $value)
```

```
                {
                    $this->sendMail($subject, $variables, $value, $from, $replyTo,
$templatePath, $toDisplayName);
                }
            }
        }
    }
```

5.5.3 Integrace platební brány GOPAY do modulu

Pro integraci platební brány do systému bylo využito SDK gopay/payments. Balíček se dá stáhnout pomocí Composeru nebo přímo z Gitu. K dispozici je i podrobná dokumentace[11].

Příkaz pro stažení balíčku:

```
composer require gopay/payments-sdk-php
```

V modulu objednávek je vytvořena třída GoPayModel, která má na starosti autentizaci, vyvolání platební brány, zpracování stavu platby a logování všech platebních operací. Tato třída používá SDK gopay/payment pro komunikaci s platební bránou.

V následujícím kódu, je metoda checkPaymentStatus třídy GoPayModel, která kontroluje stav platby, mění stav objednávky, loguje stavy platby a odesílá událost pro odeslání e-mailu o úspěšné platbě.

```
public function checkPaymentStatus($id)
{
    $response = $this->gopay->getStatus($id);

    if ($response->hasSucceed())
    {
        //log
        $this->logPaymentAction([
            'order_number' => $response->json['order_number'],
            'state' => $response->json['state'],
            'amount' => $response->json['amount'],
            'currency' => $response->json['currency'],
            'created_on' => BackendModel::getUTCDate(),
            'response' => $response,
        ]);

        // cahnge order status
        $statusId = $this->getOrderStateId($response->json['state']);
        $this->orders->setOrderStatus($response->json['order_number'], $statusId);

        if ($response->json['state'] == "PAID" && !$this-
>isOrderPaidInLogMoreThenOnce($response->json['order_number']))
        {
            $email = $response->json['payer']['contact']['email'];
            $order = $this->orders->getOrderByOrderNumber($response-
>json['order_number']);
            $this->events->sendEventOrderPaid([
                'order' => $order,
                'email' => $email,
```

[11] https://doc.gopay.com/en/

```
                ]);
            }
        }
        return $response;
}
```

Kontrola stavu platby a komunikace mezi modulem a platební bránou probíhá při přístupu na stránku s action GopayNotification a ThanksPage.

5.5.4 Export

Objednávky lze exportovat do csv souboru. Export můžeme spustit na stránce export kliknutím na odkaz exportovat. Export zprostředkovává třída Export v adresáři Engine. Metoda exportOrdersToCSV spouští export objednávek, data jsou načítána po 100 řádcích a v datovém proudu ukládána do souboru, který je stahován prohlížečem.

```
public function exportOrdersToCSV()
{
        $totalCount = $this->model->getOrdersCount();
        $baseLimit = 100;
        $data = $this->model->getOrders();
        $this->startCsv($data);

        for ($index = 1; $index < $totalCount + 1; $index++)
        {
            $offset = ($index - 1) * $baseLimit;
            $data = $this->model->getOrders($baseLimit, $offset);
            $this->writeCsv($data);
        }
        $this->endCsv();
 }
```

5.6 Soubory modulu

Soubory backendové části modulu:

```
Actions
-- AddCountry.php
-- AddCustomer.php
-- AddPaymentMethod.php
-- AddService.php
-- AddShippingMethod.php
-- AddService.php
-- AddShippingMethos.php
-- AddShippingPaymentMethodRelations.php
-- AddStatus.php
-- AllowedCountries.php
-- Customers.php
-- DeleteCustomers.php
-- DeletePaymentMethod.php
-- DeleteShipppingMethod.php
-- DeleteStatus.php
-- EditCountry.php
```

```
-- EditCustomer.php
-- EditEmailTemplate.php
-- EditOrder.php
-- EditPaymentMethod.php
-- EditService.php
-- EditShippingMethod.php
-- EditShippingPaymentMethodRelations.php
-- EditStatus.php
-- EmailOptions.php
-- EmailTemplates.php
-- Export.php
-- Gdpr.php
-- Index.php
-- OrderStatus.php
-- PaymentGatewayOptions.php
-- PaymentMethods.php
-- RemoveCountry.php
-- Service.php
-- Settings.php
-- ShippingMethods.php
-- ShippingPaymentMethodRelations.php
-- Subscription.php
-- ViewCustomer.php
-- ViewOrder.php

Ajax
-- InvoiceGenerate.php
-- ServiceOption.php

Data
 locale
-- cs.php

DependencyInjection
-- OrdersExtension.php

Engine
-- Events.php
-- Helper.php
-- MailModel.php
-- Model.php
-- SubscriptionModel.php

EventSubscriber
-- EmailEventSubscriber.php

Installer
 Data
  Delta
  -- 001_gopay.sql
 -- install.sql
 -- locale.xml
 -- uninstall.sql
-- Installer.php

Js
-- jquery-sortable.js
-- Orders.js

Layout
 Css
 -- shipping-relations.css
 Templates
 -- AddCountry.html.twig
 -- AddCustomer.html.twig
 -- AddPaymentMethod.html.twig
 -- AddService.html.twig
 -- AddShippingMethod.html.twig
```

```
 -- AddStatus.html.twig
 -- AllowedCountries.html.twig
 -- Customers.html.twig
 -- EditCountry.html.twig
 -- EditCustomer.html.twig
 -- EditEmailTemplate.html.twig
 -- EditPaymentMethod.html.twig
 -- EditService.html.twig
 -- EditShippingMethod.html.twig
 -- EditShippingPaymentMethodRelations.html.twig
 -- EditStatus.html.twig
 -- EmailOptions.html.twig
 -- EmailTemplates.html.twig
 -- Export.html.twig
 -- Gdpr.html.twig
 -- Index.html.twig
 -- OrderStatus.html.twig
 -- PaymentGatewayOptions.html.twig
 -- PaymentMethods.html.twig
 -- Service.html.twig
 -- ServiceOption.html.twig
 -- Settings.html.twig
 -- ShippingMethods.html.twig
 -- ShippingPaymentMethodRelations.html.twig
 -- Subscription.html.twig
 -- Summary.html.twig
 -- ViewCustomer.html.twig
 -- ViewOrder.html.twig

Resources
 config
 -- services.yml

-- Config.php
-- info.xml
```

Soubory frontendové části modulu:

```
Actions
-- CatalogOrder.php
-- GopayNotification.php
-- Order.php
-- ProfileOrderList.php
-- ThanksPage.php

Ajax
-- DiscountCoupon.php
-- PaymentMethods.php
-- RemoveProduct.php
-- SelectProducts.php
-- SelectService.php
-- Summary.php

Engine
-- Events.php
-- GoPayModel.php
-- Model.php
-- OrderSession.php
-- SubscriptionModel.php

Js
-- order.php

Layout
  Templates
    BackendAjax
```

```
  -- ServiceOption.html.twig
  Mails
    Blocks
    -- BillingAddress.html.twig
    -- ShippingAddress.html.twig
    -- Summary.html.twig
    -- SummaryPaymentItem.html.twig
    -- SummaryProductItem.html.twig
    -- SummaryServiceItem.html.twig
    -- SummaryShippingItem.html.twig
  -- Default.html.twig
  -- NewOrder.html.twig
  -- OrderPaid.html.twig
  -- StatusChanged.html.twig
  Summary
  -- Summary.html.twig
  -- SummaryPaymentItem.html.twig
  -- SummaryProductItem.html.twig
  -- SummaryServiceItem.html.twig
  -- SummaryShippingItem.html.twig
-- CatalogOrder.html.twig
-- GopayNotification.html.twig
-- Order.html.twig
-- ProductItem.html.twig
-- ProfileOrdersList.html.twig
-- ServiceItem.html.twig
-- ThanksPage.html.twig
```

5.7 API

Pro napojení aplikací třetích stran, jako jsou například mobilní aplikace, nebo jiná webová stránka je vytvořeno API s pár základními metodami.

Připraveny jsou metody pro načtení všech objednávek, načtení konkrétní objednávky, vložení objednávky, získání všech stavů objednávky a aktualizace stavu objednávky.

V této diplomové práci nebyla vytvořena další aplikace, která by s touto API komunikovala, komunikace s API probíhala pouze skrze aplikaci Postman. V modelovém příkladě, je tato API využívána aplikací používanou expeditorem, který odbavuje jednotlivé objednávky a potřebuje mít k dispozici jejich detail a možnost změnit jejich stav.

API je postaveno dle architektury REST a celá zdrojová část jsou mimo e-commerce modul. Jedná se o prvek Bundle frameworku Symfony, který se nachází v adresáři src/ApiBundle. Bundle je postaven na MVC návrhovém vzoru a skládá se z následujících souborů:

```
Controller
  -- ApiAbstractController.php
  -- OrdersController.php
DependencyInjection
  -- ApiExtendion.php
Model
  -- Orders.php
Resource
  Config
     -- routing.yml
-- ApiBundle.php
```

Při vytváření této části se ukázal problém s routingem, kde se nepodařilo přidat vlastním konfiguračním souborem routy konkrétních metod API. Z toho důvodu jsou routy přidány do konfiguračního souboru pro routing redakčního systému ForkCMS. Jedná se o soubor:

```
app\config\routing.yml
```

Jelikož ForkCMS má již vlastní API, ale postavenou na jiné architektuře, tak základní klíč URL je tvořen názvem api2.

V následujícím kódu můžeme vidět definici rout pro jednotlivé metody API v yaml souboru.

```
orders_api.insert_order:
    path: /api2/orders/insert-order/{locale}
    defaults:
        _controller: ApiBundle:Orders:insertOrder
    methods:  [POST]
orders_api.login:
    path: /api2/orders/login
    defaults:
        _controller: ApiBundle:Orders:login
    methods:  [POST]
orders_api.validate_token:
    path: /api2/orders/validate-token
    defaults:
        _controller: ApiBundle:Orders:validateToken
    methods:  [POST]
orders_api.get_all_states:
    path: /api2/orders/get-all-states
    defaults:
        _controller: ApiBundle:Orders:getAllStates
    methods:  [GET]
orders_api.get_order:
    path: /api2/orders/get-order/{id}
    defaults:
        _controller: ApiBundle:Orders:getOrder
    methods:  [GET]
orders_api.get_orders:
    path: /api2/orders/get-orders/{limit}/{offset}
    defaults:
        _controller: ApiBundle:Orders:getOrders
    methods:  [GET]
orders_api.update_order_status:
    path: /api2/orders/update-order-status
    defaults:
        _controller: ApiBundle:Orders:updateStatusOfOrder
    methods:  [POST]
```

Aby data objednávek nebyla zneužita třetí stranou, jsou přístupy k metodám ošetřeny autentizací. Pro volání metod je potřeba znát JWT, který je možné získat po přihlášení.

Zasláním přihlašovacích údajů (parametr login a password) dotazovací metodou POST na následující URL můžeme získat autorizační token.

```
/api2/orders/login
```

V případě správných přihlašovacích údajů nám bude zaslána zpráva spolu s tokenem. Následující kód je ukázka úspěšného přihlášení.

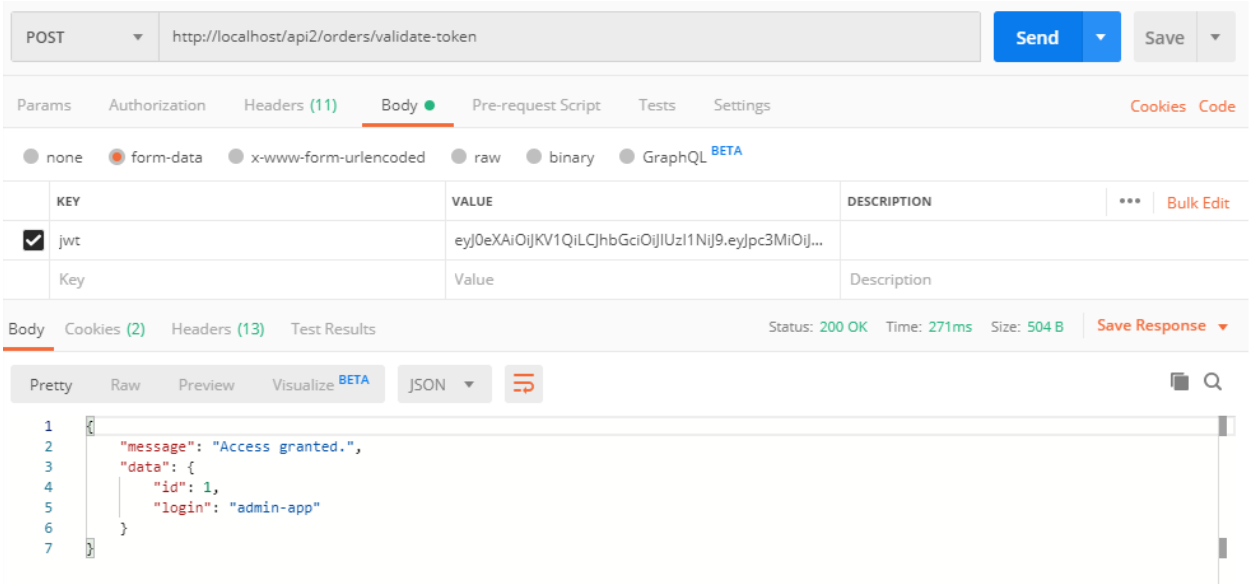

```
{
    "message": "Successful login.",
    "jwt": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOlwvXC9sb2NhbGhvc3QiLCJhdWQiOiJodHRwOlwvXC9sb2NhbGhvc3QiLCJpYXQiOjEzNTY5OTk1MjQsIm5iZiI6MTM1NzAwMDAwMCwiZGF0YSI6eyJpZCI6MSwibG9naW4iOiJhZG1pbi1hcHAifX0.eJEPFwuHM19birQvCsFeRc5g_5suJ-aCJm7TI9oiMko"
}
```


Tento token je potřeba mít v těle každé volané API metody jako parametr „jwt“.

Platnost jwt si můžeme ověřit zasláním tokenu na url:

```
/api2/orders/validate-token
```

Vrácen nám bude dešifrovaný obsah tokenu.



Obrázek 27 – Kontrola validity tokenu skrze aplikaci Postman. Zdroj: zpracování autora

Pro generování a dekódování tokenů byla použita PHP knihovna JWT. Stáhnout tuto knihovnu pomocí composeru následujícím příkazem.

```
composer require firebase/php-jwt
```

5.8 Rozdíly oproti klasickému modulu redakčního systému Fork

Při vyváření modulu „Orders“ byly některé praktiky používané v systému Fork nahrazeny jinými. Například je běžné, že třídy jsou statické, a to se netýká jen tříd modulů. Běžně se tedy definuje statická třída Model v adresáři Engine, jejíž metody lze volat přímo na třídě, bez vytvoření instance, a to pomocí operátoru čtyřtečky. Metody a atributy nepatří instanci, ale patří třídě. Třída Model má na starosti dotazování do databáze a v tomto případě statika zásadně

nevadí, jenže jsou potřeba i jiné metody a operace s daty. Dá se konstatovat, že statika narušuje objektový model a není dobré ji nadměrně používat. Součástí tohoto modulu nejsou statické třídy ale klasické třídy, které jsou registrovány jako služby a spravovány skrze techniku Dependenci Injection.

Redakční systém Fork neumí aktualizovat ani odinstalovat modul. Pro ulehčení aktualizace, vznikly delta scripty, které rozdělují databázové změny do samostatných souborů, které jsou pouštěny postupně při instalaci, nebo jej jednoduše může pustit programátor, který modul aktualizuje. Pro rychlé odebrání modulu vznikl odinstalační script uninstall.sql, který programátor ručně pustí nad databází.

Další nezvyklostí u většiny modulů je například absence cizích klíčů v tabulkách databáze. U tabulek se nacházejí slupce pro identifikátor záznamu z jiné tabulky, ale není definován mezi nimi žádný vztah. Důvodem je neunikátnost primárního klíče, u většiny modulů má příspěvek pro každou mutaci vlastní záznam se stejným ID a mutace je rozlišována skrze sloupec language.

5.9 Postup při vytváření modulu

Pro vytvoření vlastního nebo podobného modulu je vhodné dodržovat podobné struktury dle kapitoly 5.2.1. Začít lze sestavováním modulu z backendové, nebo frontendové části. Nejvhodnější je začít instalačním scriptem v backendové části. Je potřeba definovat strukturu navigace v backendu a frontendové prvky jako Widgety a Actions. Po instalaci modulu již není možné znovu spustit upravený instalační script, takže instalaci je vhodné pouštět až v době kdy víme, že instalační script odpovídá návrhu modulu. V případě, že je potřeba aktualizovat modul, tak je potřeba buď to modul odinstalovat připraveným odinstalačním scriptem, nebo provedené změny ručně přidat do databáze.

Pro multijazyčnost modulu, doporučuji dodržovat v šablonách strukturu placeholderů pro překlady Forku a rovnou je definovat v souboru locale.xml. Pokud překlad neexistuje nebo není překlad importovaný do systému, tak se budou zobrazovat placeholdery.

Po každé úpravě je potřeba překlad importovat buď přes administraci nebo následujícím příkazem.

```
php app/console locale:import --module=<nazev modulu>
```

Při vývoji se vyskytne s velkou pravděpodobností spousta chyb, v developerském režimu bude chyba vypsána v prohlížeči. V produkčním režimu lze informaci o chybě zjistit jen z logu. Logy

nalezneme v adresáři app/logs, ve vývojářském režimu se chyby zapisují do souboru dev.log a v produkčním režimu se chyby zapisují do prod.log.

Vypsat informace z logu můžeme například pomocí příkazu:

```
less app/logs/dev.log
```

Při vývoji je dobré si uvědomit, že Fork poměrně hodně cachuje, obzvlášť v produkčním režimu. Mazat cache je potřeba například při změně v konfiguračních souborech. V produkčním režimu je potřeba mazat cache pro jakoukoliv změnu v šabloně. Cache paměť lze smazat pomocí příkazu:

```
php app/console cache:clear --env=<dev/prod>
```

5.10 Instalace

5.10.1 Instalace Redakčního systému

Instalace redakčního systému je poměrně jednoduchá, nakopírujeme soubory redakčního systému do adresáře, pokud používáme git tak uděláme git clon. Potřeba je také v příkazovém řádku spustit příkaz „*composer install*“, který stáhne všechny potřebné balíčky a jejich závislosti. Poté přistoupíme na URL adresu, která vede na kořenový adresář, ve kterém jsou soubory umístěny. Při správném postupu se v prohlížeči zobrazí instalační formulář, pokud váš server nesplňuje konfigurační požadavky, budou vám zde vypsány nedostatky, které je potřeba opravit. V instalaci zvolíte, zda web bude vícejazyčný nebo jedno jazyčný, zadáte registrační údaje pro administrátora a přístupové údaje k databázi.

5.10.2 Instalace modulu

Modul by měl být sestavený od kořenového adresáře, tak aby jeho podadresáře odpovídaly správnému umístění souborů modulu.

Adresář s modulem (src/) nakopírujeme do kořenového adresáře redakčního systému. Pokud modul máme v gitu, vytvoříme v adresáři novou složku, například „external_modules“ uvnitř pustíme příkaz „*git clone <repozitář>* “, který stáhne modul z gitu. Následně adresář /src nakopírujeme do kořenového adresáře systému. Modul je sice nyní v systému, ale není nainstalován. Je potřeba jít do administrace a zvolit Nastavení → Moduly. Na stránce se vypíše seznam instalovaných a nenainstalovaných modulů. Najdeme modul „Orders“ a klikneme na instalovat. Pokud instalace proběhla úspěšně, ve vrchní části stránky bude zobrazen zelený pruh s informací, že modul byl úspěšně nainstalován.

Po úspěšné instalaci by se v navigaci měla objevit položka „Objednávky“. Pokud v navigaci není položka „Objednávky“, je potřeba smazat cache paměť redakčního systému. Dále už je jen potřeba nakonfigurovat, služby, dopravy, platby, e-maily, informace o prodejci a dosadit block/widget na stránku, na které se má objednávkový formulář zobrazovat.

6 ZÁVĚR

V této diplomové byla provedena rešerše redakčních systémů, byly popsány výhody a nevýhody vývoje pro open source redakční systémy. V druhé části byly popsány použité technologie a porovnání PHP frameworků. V poslední části byl vytvořen a popsán modul pro redakční systém Fork CMS.

Modul má rozsáhlou administraci, která umožňuje uživateli systému vypsat objednávky, filtrovat objednávky, zobrazit detailní informace o objednávce, změnit stav objednávky, notifikovat zákazníka o změně stavu objednávky, vypsat všechny zákazníky, zobrazit všechny objednávky zákazníka, definovat přepravní metody, definovat platební metody, definovat vztahy mezi platebními a přepravními metodami, spravovat doplňkové služby k prodeji, definovat destinaci prodeje, definovat a spravovat stavy objednávky, spravovat obsah automatických e-mailů, konfigurace pro přizpůsobení výpisu objednávek a zákazníků, konfigurace údajů platební brány.

Frontendová část modulu umožňuje přidat na stránku objednávkové formuláře v různých variantách. Tyto formuláře mají AJAXové prvky, které dokáží reagovat na změny provedené zákazníkem a zobrazovat mu přesné informace o objednávce.

Modul je připraven pro integraci platební brány.

Po dokončení modulu se dá konstatovat, že vývoj modulu pro redakční systém není jednoduchý, sice nám redakční systém dává spoustu možností, které usnadní vývoj, ale zároveň vytváří pro vývojáře různá omezení. Pro vývoj je potřeba znát základní prvky využívaného systému a dokázat je použít. Také se ukázalo, že některé praktiky nejsou úplně ideální a byly částečně nahrazeny modernějším způsobem.

Závěrem je také, že tento modul nemůže konkurovat větším e-shopovým redakčním systémům, ale může být vhodný tam, kde velký e-shopový systém nemá moc smysl. To může být například prodej služeb, nebo malého množství produktů.

V době vývoje e-commerce modulu, systém Fork CMS již pomalu zaostává nad moderními technologiemi a metodami vývoje, ale pro svoji jednoduchost a administrovatelnost si nejspíš po nějakou dobu udrží své uživatele.

Osobním přínosem pro mne je práce s frameworkem Symfony, využití AJAX při práci s obsahem a získání přehledu o komplikovanosti vytváření e-commerce řešení.

7 POUŽITÁ LITERATURA

[1] **MAUTHE, Andreas a Peter THOMAS.** *Professional content management systems: handling digital media assets.* Chichester: John Wiley, c2004. ISBN 0-470-85542-8.

[2] **STALLMAN, Richard.** Why Open Source misses the point of Free Software. In: *gnu.org* [online]. [cit. 03.11.2019]. Dostupné z: http://www.gnu.org/philosophy/open-source-misses-the-point.html.

[3] **FOGEL, Karl.** Producing Open Source Software: How to Run a Successful Free Software Project. In: *producingoss.com* [online]. 2005–2019 [cit. 03.11.2019]. Dostupné z: https://producingoss.com/en/producingoss-a4.pdf.

[4] *CVE Details* [online]. ©2010 [cit. 10.10.2019]. Dostupné z: https://www.cvedetails.com/

[5] **BELL, Peter a Brent BEER.** *Introducing GitHub: a non-technical guide.* Beijing: O'Reilly, [2014]. ISBN 978-1-491-94974-0.

[6] **FRAIN, Ben.** *Responsive web design with HTML5 and CSS3: learn responsive design using HTML5 a CSS3 to adapt websites to any browser or screen size*. Birmingham: Pack Publishing, c2012. ISBN 978-1-84969-318-9.

[7] W3C. *The global structure of an HTML document.* [online]. c2009 [cit. 03.11.2019]. Dostupné z: https://www.w3.org/TR/html401/struct/global.html

[8] **WOOD, Adam.** HTML5: What's New in The Latest Version of HTML? In: *html.com* [online]. 21.7.2019 [cit. 03.11.2019]. Dostupné z: https://html.com/html5/.

[9] TWIG. *The flexible, fast, and secure template engine for PHP*. [online]. © 2010-2019 [cit. 3.11.2019]. Dostupné z: https://twig.symfony.com/

[10] GOPAY. *Platební brána pro váš bussiness*. [online]. © 2019 [cit. 3.11.2019]. Dostupné z: https://www.gopay.com/cs/co-brana-umi.html

[11] **VEROU, Lea**. *CSS secrets: better solutions to everyday web design problems.* Sebastopol, CA: O'Reilly Media, 2015. ISBN 978-1-449-37263-7.

[12] **LOCKHART, Josh.** *Modern PHP: new features and good practices. Sebastopol*, CA: O'Reilly Media, 2015. ISBN 978-1-491-90501-2.

[13] **WELLING, Luke a Laura THOMSON**. *PHP and MySQL web development*. Fifth edition. Hoboken, NJ: Addison-Wesley, [2017]. Developer's library. ISBN 978-0-321-83389-1.

[14] Introducing Symfony: Who Made Symfony and Why. SENSIO LABS. Symfony [online]. [cit. 2019-11-07]. Dostupné z: http://symfony.com/blog/the-30-most-usefulsymfony-bundles-and-making-them-even-better

[15] **HANSEN, Steven.** *7 Good Reasons to Use Symfony Framework for Your Project.* In: hackernoon.com [online]. 28.11.2017 [cit. 07.11.2019]. Dostupné z: https://hackernoon.com/7-good-reasons-to-use-symfony-framework-for-your-project-265f96dcf759.

[16] **JAIN, Ayush.** *Best PHP Frameworks (2019)*. In: dzone.com [online]. 12.6.2019 [cit. 11.11.2019]. Dostupné z: https://dzone.com/articles/best-php-frameworks-2019. Path: Homepage; Web dev; 12.6.2019.

[17] MDN Web docs. *What is JavaScript?* [online]. 20.7.2019 [cit. 13.11.2019]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[18] IBM – United States. IBM Knowledge Center. *What is Ajax?* [online]. Copyright IBM Corp. ©2017. [Cit. 15.11.2019]. Dostupné z: https://www.ibm.com/support/knowledgecenter/SS8PJ7_9.6.1/com.ibm.etools.webtoolscore.doc/topics/cajax.html

[19] W3C. Web Services Architecture. *W3C Working Group Note 11 February 2004* [online]. World Wide Web Consortium, ©2004. [Cit. 16.11.2019]. Dostupné z: https://www.w3.org/TR/ws-arch/

[20] PrestaShop [online]. PrestaShop SA, corporation company: ©2019 [cit. 16.11.2019]. Dostupné z: https://www.prestashop.com/en

[21] **UL HASAN, Syed Muneeb**. Things To Consider For Magento Optimization and Better Performance In: *magenticians.com* [online]. 7.5.2019 [cit. 16.11.2019]. Dostupné z: https://magenticians.com/magento-optimization/.

[22] Magento An Adobe Company. Tech Resources [online]. Magento, Inc.: ©2019 [cit. 16.11.2019]. Dostupné z: https://magento.com/technical-resources

[23] **ANDREA, Michal.** ByznysPark w/ Petr Svoboda: Budoucnost je Open Source. In*: zoom.rba.cz* [online]. 13.7.2017 [cit. 16.11.2019]. Dostupné z: https://zoom.rba.cz/prednasky/byznyspark-w-petr-svoboda-budoucnost-je-open-source

[24] Shopsys Framework [online]. Shopsys s.r.o., ©2003-2019 [cit. 16.5.2018]. Dostupné z: https://www.shopsys.cz/co-je-shopsys-framework

[25] Blogovací nástroj, platforma pro publikování a CMS [online]. Automattic Inc., [cit. 16.11.2019]. Dostupné z: https://cs.wordpress.org/

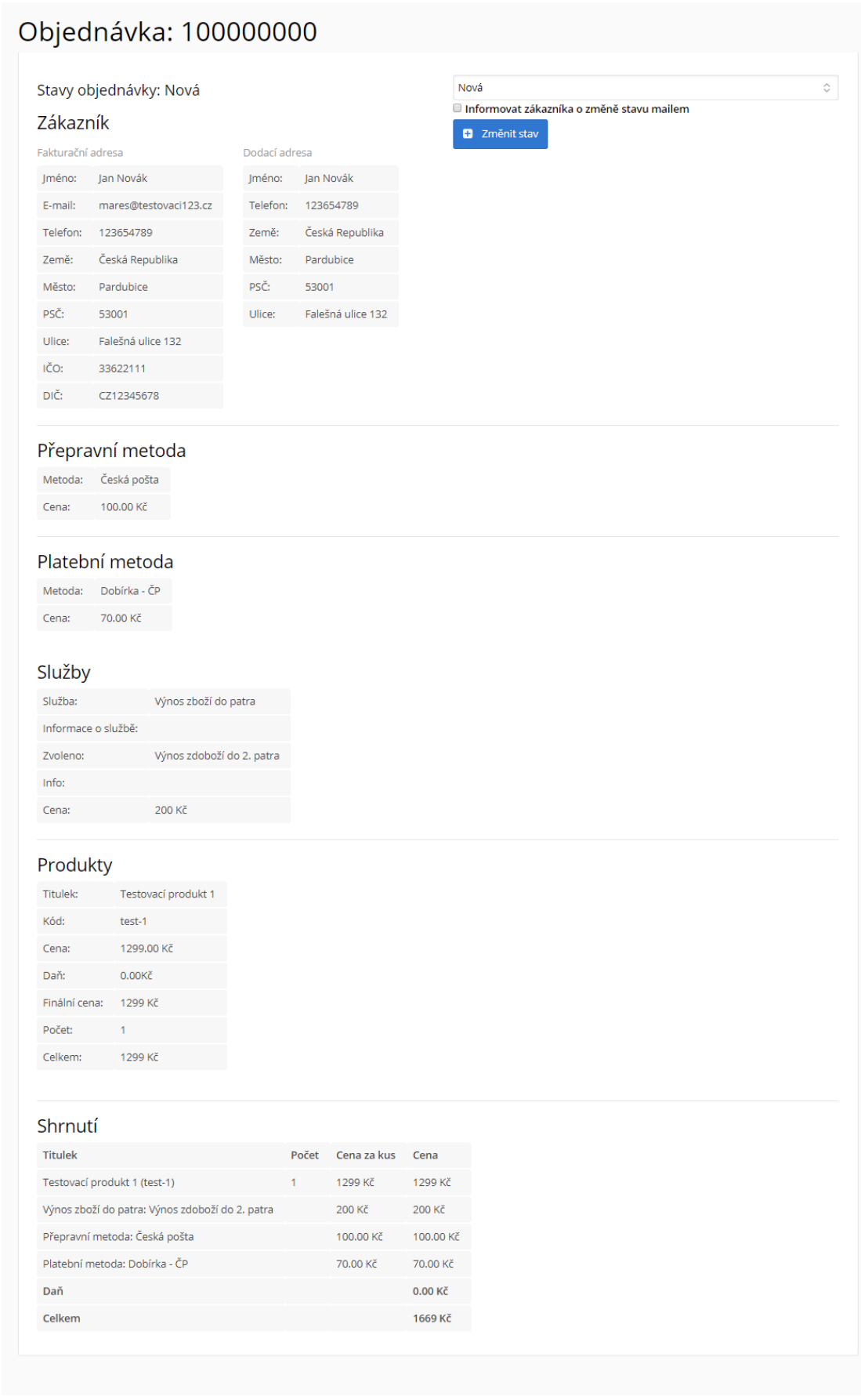
[26] Joomla! - Content Management System to build websites & apps [online]. Open Source Matters, Inc. ©2005-2019 [cit. 17.11.2019]. Dostupné z: https://www.joomla.org/about-joomla.html

[27] The Fork CMS [online]. Fork CMS. [cit. 17.11.2019]. Dostupné z: https://www.fork-cms.com/features

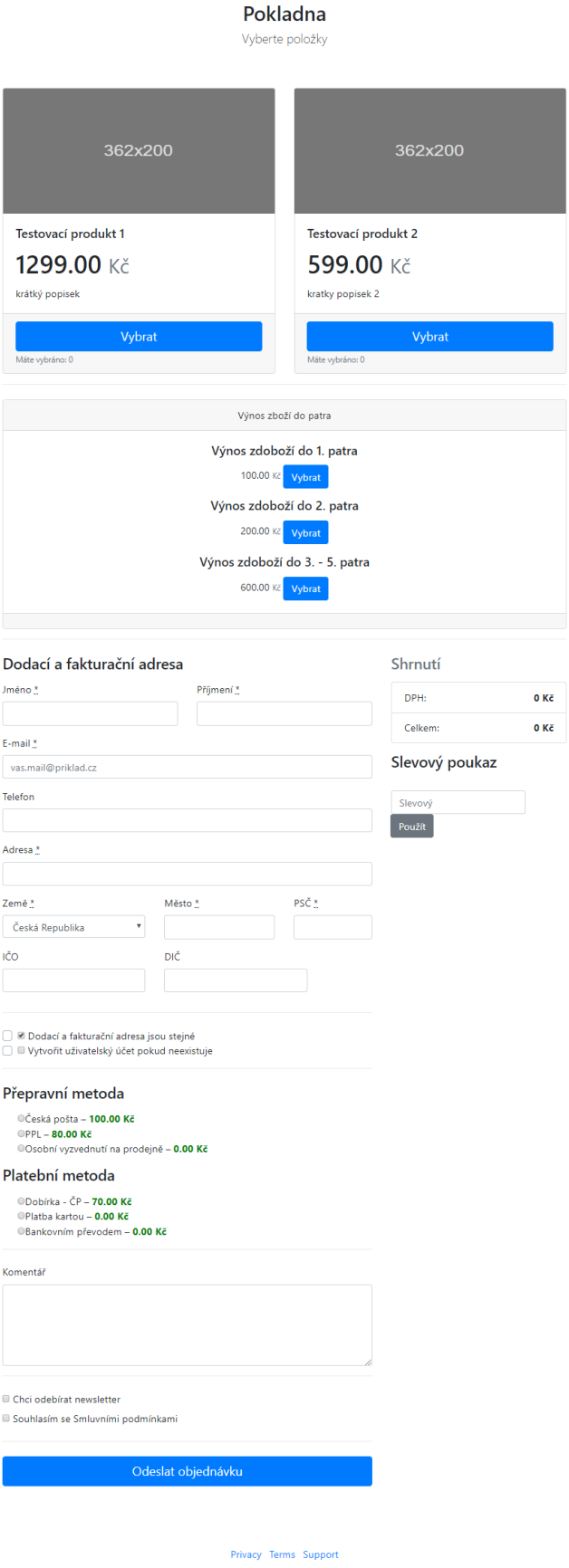
[28] **KRČMÁŘ, Petr.** HTTP/3 nebude postavené na TCP, základem bude QUIC používající UDP. In: *Root.cz* [online]. 14.11.2018 [cit. 5.8.2019]. Dostupné z: https://www.root.cz/clanky/http-3-nebude-postavene-na-tcp-zakladem-bude-quic-pouzivajici-udp/.

[29] Auth0. *JSON Web Tokens*. JSON Web Token Introduction - jwt.io. [online]. Auth0, Inc. ©2019 [cit. 12.1.2020]. Dostupné z: https://jwt.io/introduction/.

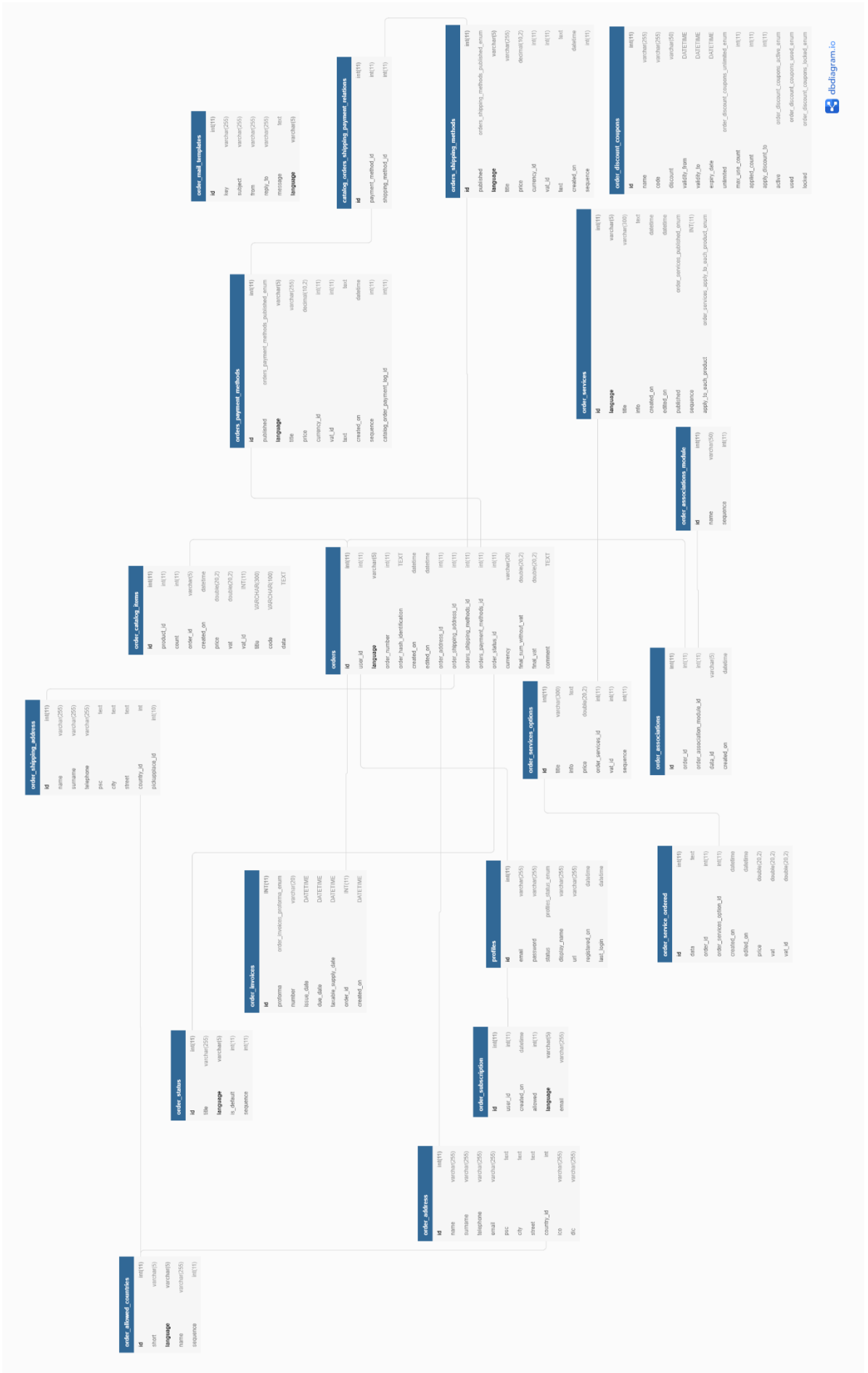
8 PŘÍLOHY



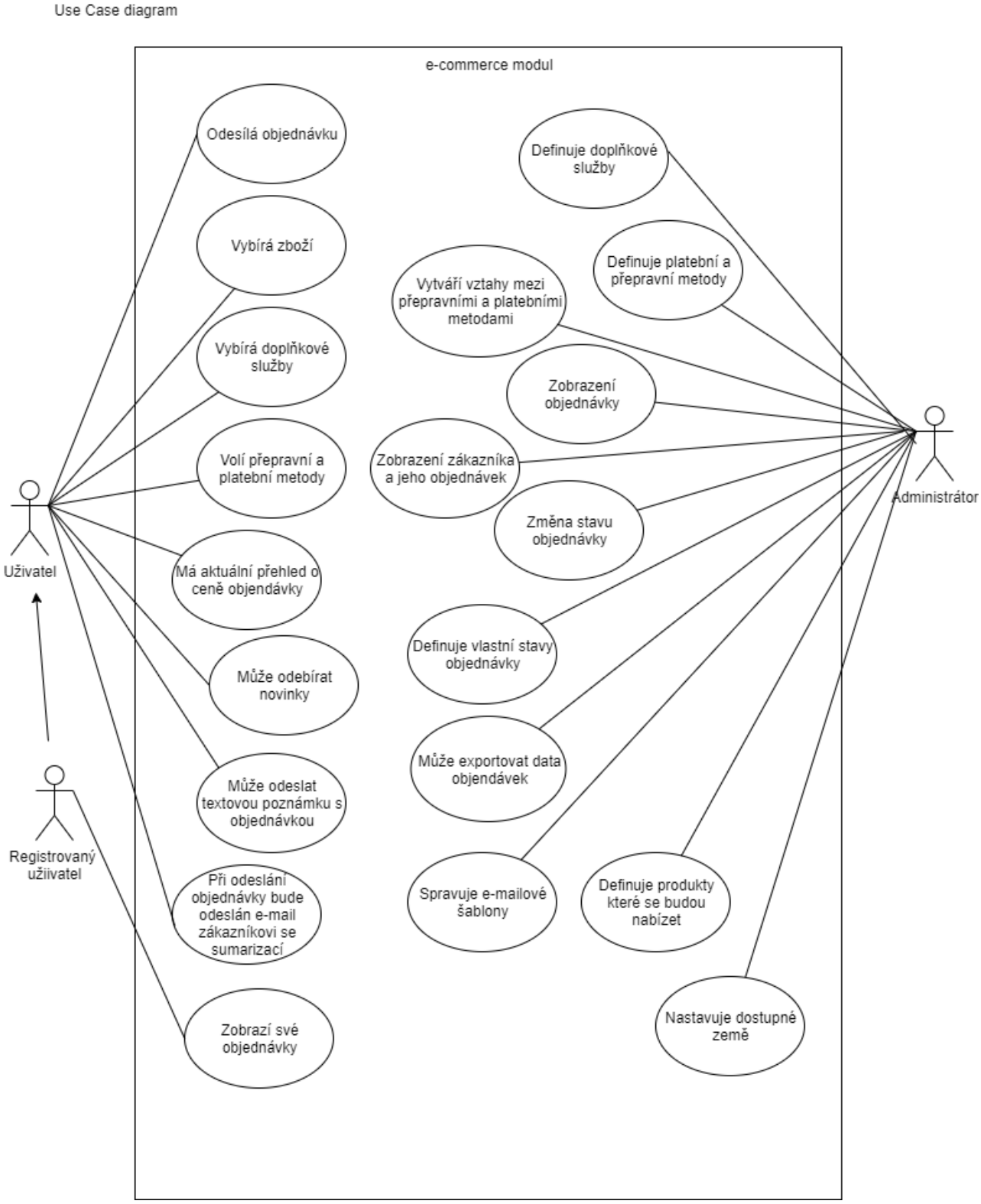
Příloha A – Detail objednávky. Zdroj: zpracování autora



Příloha B – Náhled na objednávkový formulář. Zdroj: zpracování autora



Příloha C – Diagram databáze. Zdroj: zpracování autora



Příloha D – Use Case diagram. Zdroj: zpracování autora