Rozhraní pro internetové nákupy pomocí služby PayPal

Dominik Pfeffer

Bakalářská práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Dominik Pfeffer**
Osobní číslo:	**A14131**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**prezenční**
Téma práce:	**Rozhraní pro internetové nákupy pomocí služby PayPal**
Téma anglicky:	**An Online Purchases Interface using PayPal Services**

Zásady pro vypracování:

1. **V rámci teoretické části popište vlastnosti a architekturu vývojového frameworku Nette.**
2. **Stručně charakterizujte návhové vzory využívané v oblasti tvorby webových aplikací.**
3. **Prozkoumejte možnosti implementace rozhraní pro uskutečňování plateb služby PayPal do vlastní webové aplikace.**
4. **Implementujte webovou aplikaci, která bude integrovat platební systém PayPal, zaměřte se také na její bezpečnost.**
5. **Testovacím provozem ověřte funkčnost aplikace a komunikace s platebním portálem PayPal.**

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. PRETTYMAN, Steve. Learn php 7: object oriented modular programming using HTML5, CSS3, Javascript, XML, JSON, and MYSQL. New York: Apress, 2012. ISBN 9781484217290 SHERIF, Mostafa Hashem. Protocols for secure electronic commerce. Third edition. New York: CRC Press, 2016. ISBN 9781482203745
2. Nette framework [online]. Česká republika: Nette foundation, 2017 [cit. 2017-01-25]. Dostupné z: https://nette.org/
3. BÖHMER, Marian. Návrhové vzory v PHP: [23 vzorových postupů pro rychlejší vývoj]. Brno: Computer Press, 2012. ISBN 978-80-251-3338-5.
4. MILLER, Michael. The PayPal official insider guide to growing your business: make money the easy way. United States of America: Peachpit, 2012. ISBN 0321768523.
5. LECKY-THOMPSON, Ed a Steven D. NOWICKI. PHP 6: programujeme profesionálně. Brno: Computer Press, 2010. Programujeme profesionálně. ISBN 978-80-251-3127-5

Vedoucí bakalářské práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **24. února 2017**

Termín odevzdání bakalářské práce: **24. května 2017**

Ve Zlíně dne 24. února 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.
1

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 16.5.2017

.....
podpis diplomanta

ABSTRAKT

Tato bakalářská práce se zabývá návrhem a implementací rozhraní pro uskutečňování online nákupů a plateb pomocí služby PayPal. Aplikace je psána pomocí jazyku PHP ve vývojovém frameworku Nette. Práce zahrnuje zhodnocení možností implementace platebního systému PayPal a popis Nette frameworku a je rozdělena na teoretickou a praktickou část. V teoretické části se charakterizují pojmy typu framework, návrhový vzor, elektronické obchodování aj. Velká část je věnována samostatnému frameworku Nette a jeho vlastnostem. Dále také obsahuje srovnání s alternativními frameworky pro vývoj webových aplikací. V praktické části jsou převedeny znalosti z teorie do praxe a implementace samotného platebního rozhraní.

Klíčová slova: Netteframework, Nette, MVP, internetový obchod, PayPal, platební peněženka, PHP, e-shop

ABSTRACT

This bachelor thesis deals with the design patterns and implementation of an interface for making online purchases and payments using Paypal. The application is written by PHP in the Nette development framework. The thesis includes an assessment of implementation of the PayPal payment system and the description of Nette framework. The thesis is divided in to the theoretical and practical part. In the theoretical part are characterized concepts of the framework type, design pattern, electronic trading etc. A great part is devoted to Nette's framework and its features. This work compares Nette with alternative frameworks. In the practical part the application is made.

Keywords:Nette framework, Nette, MVP, E-shop, PayPal, PHP

Tímto bych chtěl poděkovat mému vedoucímu bakalářské práce Ing. Radkovi Valovi, Ph.D., za odborné vedení, konzultace, rady a především trpělivost.

„When you see a good move, look for better one.“

Emanuel Lasker

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Tématem bakalářské práce je návrh a implementace rozhraní pro uskutečňování on-line nákupů a plateb pomocí služby PayPal. Bakalářská práce neslouží jako návod pro vytvoření celého internetového obchodu. Může však pomoci při realizaci platebního systému PayPal.

Téma práce bylo vybráno kvůli osobnímu využití Nette framworku při pracovní činnosti a možnosti postupného začlenění platební peněženky do již připraveného internetového obchodu na žádost zaměstnavatele. Rozmach internetových obchodů je v dnešní době značný a dle osobního názoru autora je platba pomocí elektronické peněženky relativně bezpečný a pohodlný způsob nákupu.

V teoretické části bude objasněn pojem framework, návrhový vzor, MVC a MVP architektura. Ve třetí kapitole bude popsán vývojový framework Nette, jeho vlastnosti a zabezpečení. V dalším bodě bakalářské práce budou zmíněny možnosti debugování kódu pomocí tzv. laděnky. Následně bude přiblížen SSL certifikát a HTTPS protokol využitý pro šifrovanou komunikaci. Čtvrtá kapitola popisuje elektronické obchodování, přibližuje pojem internetový obchod, rozebírá platební peněženky. Závěr teoretické části se věnuje platebnímu systému PayPal, jeho výhodám, nevýhodám a zabezpečení.

Praktická část obsahuje popis technologie použité pro tvorbu internetového obchodu. Následně bude popsána adresářová struktura, generovaná pomocí nástroje composer. V další kapitole budou zmíněny požadavky na tvorbu funkční demo aplikace s cílem testování platebního systému PayPal. Pomocí ukázky kódu budou popsány důležité části aplikace. Implementace platebního systému PayPal je pro větší přehlednost popsána na závěr ve své vlastní kapitole.

I. TEORETICKÁ ČÁST

1 FRAMEWORK

Framework, neboli aplikační rámec, je skupina společných a předpřipravených softwarových bloků (sada tříd, knihoven a nástrojů), které můžou programátoři použít, rozšířit, anebo přizpůsobit dle vlastních potřeb. Jsou určené pro vykonávání základní funkcionality aplikace. S frameworkem programátor (vývojář) nemusí začínat programovat od nuly, umožňuje mu soustředit se na projekt namísto psaní neustále opakující se rutinní práce. Jde zejména o části, jež se vyskytují pravidelně, téměř v každém větším projektu např. přihlášení a odhlášení uživatelů, tvorba a zpracování formulářů apod. [2]

Framework přispívá k čitelnosti kódu. Určuje logickou strukturu a styl psaní, kterého by se programátor měl držet. V PHP se u větších projektů bez kvalitních vývojových nástrojů neobejdete.

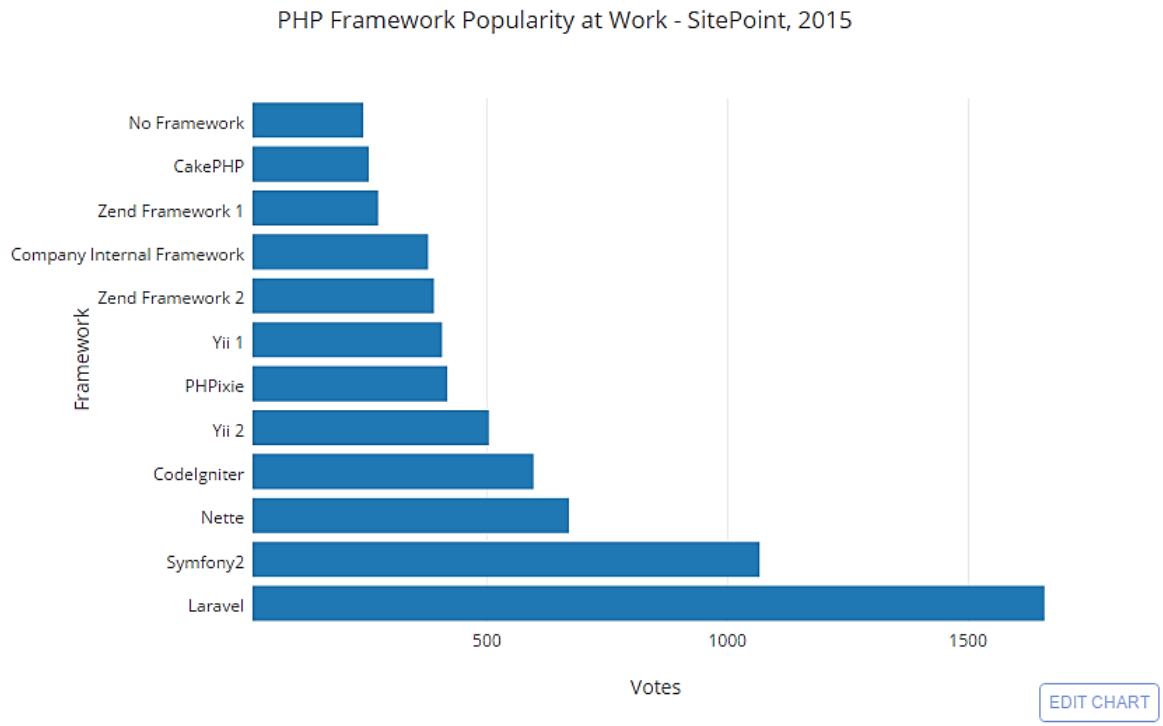
Hlavní důvody jsou:

1. I přes kvalitu PHP jazyka se časem stane, že programátor nalezne mezery v jeho standardních knihovnách. Některé důležité funkce v něm chybí nebo je s nimi špatná práce. V dalších případech zjistí, že určité věci programuje stále dokola.
2. Je obecně známo, že při programování v čistém PHP bez přidaných frameworků může být až polovina kódu nepotřebná nebo se stále opakuje. A s použitím kvalitních knihoven je práce napsána za polovinu času a s polovičním počtem řádků. Aplikace je lépe udržitelná. [1]
3. Díky znalosti vývojových frameworků programátor zvýší svou hodnotu na pracovním trhu.

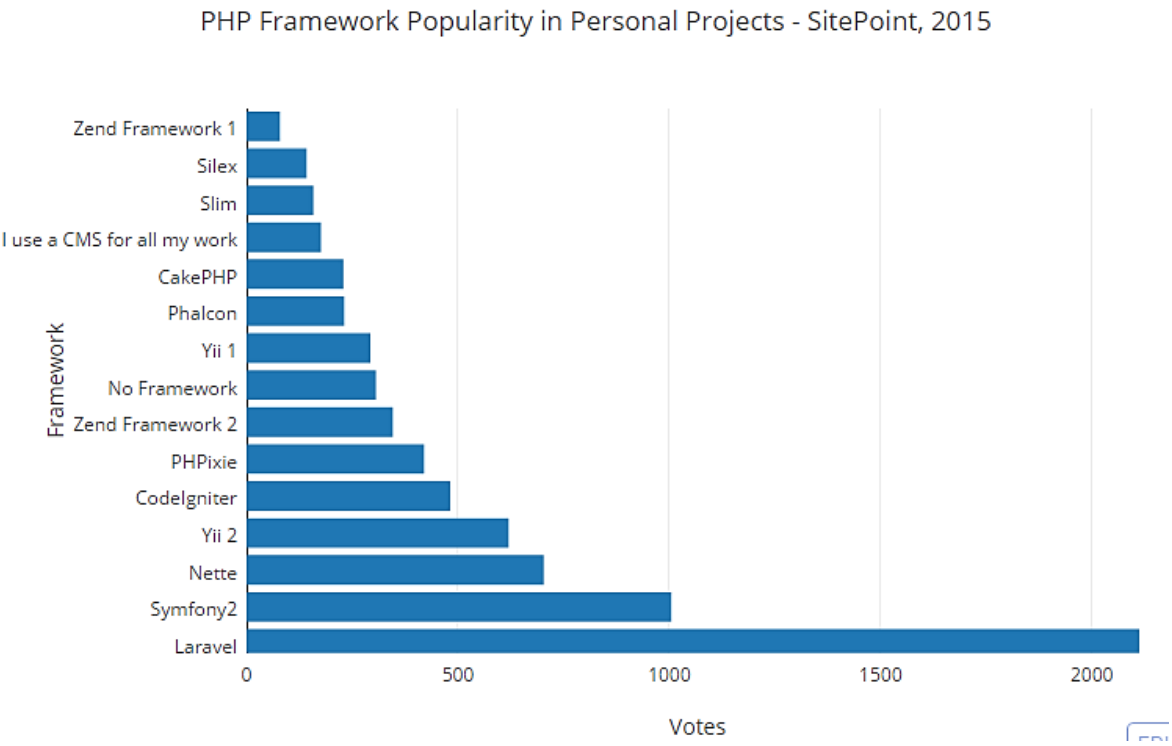
Nejznámější celosvětové PHP frameworky jsou například Laravel, Symfony, Zend, Phalcon, CodeIgniter. V České republice je neznámější a nejvíce používán Nette Framework.[5]

Více o něm bude popsáno v samostatné kapitole.

1.1 Srovnání PHP Frameworků



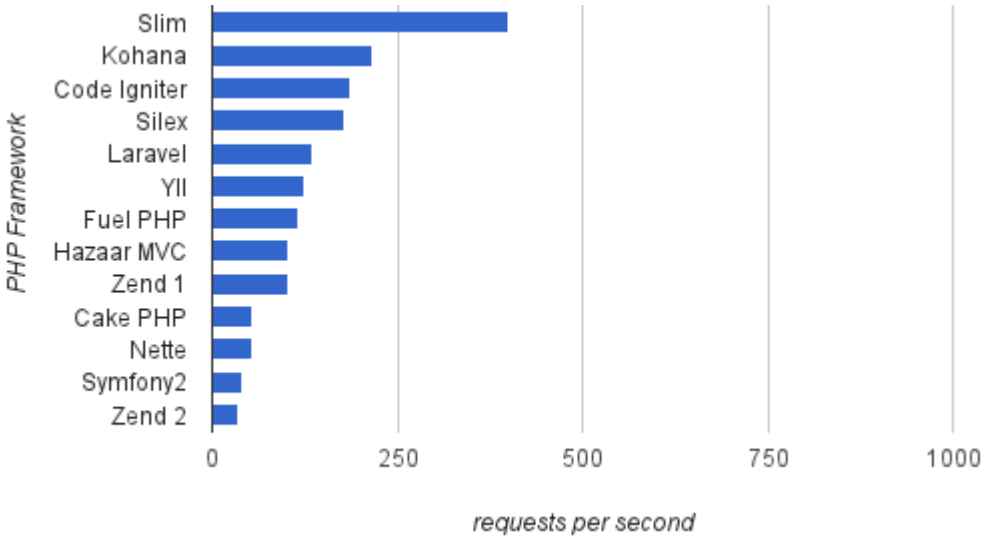
Obrázek 1: Srovnání PHP frameworkův práci[15]



Obrázek 2: Srovnání PHP frameworků, při užití na osobních projektech [15]

Na obrázku č. 1 a č. 2 lze vidět srovnání PHP frameworků v práci a na osobních projektech. Nejpopulárnější je Laravel, následuje Symfony a autorem využitý Nette si udržuje třetí místo mezi PHP giganty. PHP frameworků je velké množství, pro větší přehlednost je uveden pouze zlomek. [15]

1.1.1 Srovnání na základě výkonosti



Obrázek 3: Benchmark výkonosti PHP frameworků [16]

Nejlépe hodnocený z příkladových frameworků je Slim, a to právě z toho důvodu, že je to micro framework. Tento framework nepoužívá žádný šablonovací systém, což taky přispělo k jeho rychlosti. Avšak dle zkušených vývojářů by tato rychlost značně poklesla při reálném využití. [16]

Nette Framework je zhruba o třetinu rychlejší než Zend 2 a Symfony2, které patří mezi nejpoužívanější a největší na světě.

2 OBECNÉ NÁVRHOVÉ VZORY

Návrhový vzor je postup, podle kterého se program tvoří. Jsou to ověřené programovací postupy řešení pravidelně se vyskytujících situací, podle nichž vývojáři přistupují k problémům. Umožňují napsat čitelnější kód, v němž se budou orientovat i vývojáři, jež tento kód nepsali. [3]

Návrhový vzor se obecně skládá ze čtyř základních prvků:

1. **Název vzoru** je záhlavím, které je používáno k popisu návrhového problému, jeho řešení a důsledků. Popisuje se jedním nebo dvěma slovy.
2. **Problém** obecně popisuje, za jakých podmínek se má vzor použít. Slouží k vysvětlení problému a jeho kontextu.
3. **Řešení** slouží k popisu prvků, z nichž se návrh skládá. Avšak řešení nepopisuje konkrétní problém. Vzor slouží pouze jako šablona, jež je používaná v nejrůznějších situacích.
4. **Důsledky** jsou výsledky a kompromisy použití vzoru. Softwarové důsledky se ve většině případů zabývají prostorovými a časovými kompromisy. [6]

Návrhové vzory se dělí:

1. **Creational patterns** (vytvářející) má za úkol řešit problémy s vytvářením objektů v systému. Snahou těchto vzorů je popsat postup výběru třídy nového objektu. Jde o dynamická rozhodnutí učiněná za běhu programu.
2. **Structural patterns** (strukturální) jde o skupinu, jejíž vlastním zaměřením je uspořádání jednotlivých tříd nebo komponentů v systému. Hlavním cílem je zpřehlednění systému za pomocí strukturalizace kódu.
3. **Behavioral patterns**(chování) jsou založeny na třídách nebo objektech, zajímají se o chování systému. Zde se nejvíce využívá principu dědičnosti. [3]

2.1.1 MVC architektura

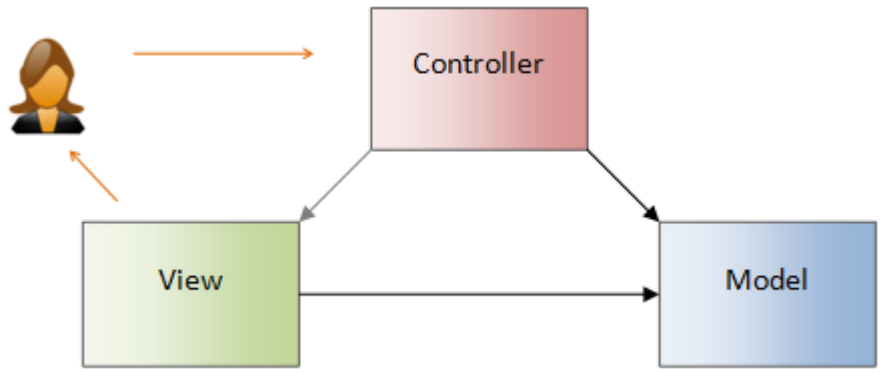
Model-view-controller je softwarový architektonický vzor, jehož uplatnění spočívá nejčastěji u aplikací klient/server (webových aplikací). Jeho první verze vznikla roku 1979. Původní autor návrhu je norský profesor Trygve Reenskaug. Tento vzor je jeden z nejvyužívanějších architektonických vzorů, ale ve své původní podobě už není používán.

Jeho využití můžeme nalézt u nejrůznějších frameworků, například Zend, ASP .NET, Ruby on Rail. Základní myšlenka MVC architektury je oddělit logiku od výstupu. Aplikace se dělí na komponenty nazývající se Model, View (pohled), Controller. [7]

Model je datový a funkční základ celé aplikace. Je to nejjednodušší část, obsahuje logiku aplikace, do které patří databázové dotazy, výpočty, validace apod. Funkce modelu je založena na přijetí a zpracování parametrů. Model o existenci view nebo controlleru neví.

View (pohled) má za úkol zobrazovat výstup uživateli. Obsahuje minimální množství logiky, která je nutná pro výpis. Vrstva view, stejně jako model, neví, odkud data přišla, stará se pouze o jejich výpis. U většiny frameworků se pro větší pohodlí uživatele obvykle používají šablonovací systémy pro vykreslení

Controller (kontroler) se chová jako prostředník, je to řídící vrstva. Zpracovává veškeré požadavky od uživatele a komunikuje s modelem i pohledem a vzájemně je propojuje. Obdobou controllerů v MVP architektuře jsou presentery. [7]



Obrázek 4: Schéma MVC [8]

Na obrázku výše můžeme vidět princip fungování MVC modelu. Uživatel vykoná akci v uživatelském rozhraní. Akce je zachycena pomocí controlleru. Controller upraví data v modelu nebo přímo ovlivní view. Následně view zobrazí změny uživateli. [8]

2.1.2 MVP architektura

Model-view-presenter je softwarový návrhový vzor, jenž je používaný v Nette frameworku, který bude popisován později. MVP architektura je založena na mnohem známější MVC architektuře, která byla popsána výše. [8]

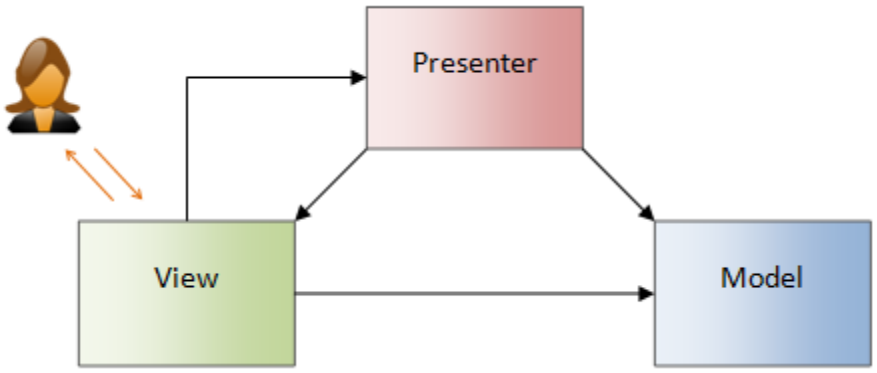
V rámci frameworku je logika rozdělena následovně:

Model, stejně jako u MVC, obsahuje logiku aplikace, zajišťuje přístup k datům a manipulaci s nimi. Model neví, že view nebo presenter existují.

View (pohled) je uživatelský vstup a výstup. Na rozdíl od MVC architektury plně kontroluje view. Převádí data reprezentovaná modelem do podoby vhodné k prezentaci uživateli. Pro vykreslení používá Latte šablony s HTML kódem. Latte je šablonovací jazyk navržený pro Nette, díky němu může programátor do HTML šablon vkládat data z PHP za pomocí speciálních značek. Podle potřeby programátora nemusí view vědět o modelu, anebo může přistupovat přímo k jeho datům, podle zvolené koncepce. [9]

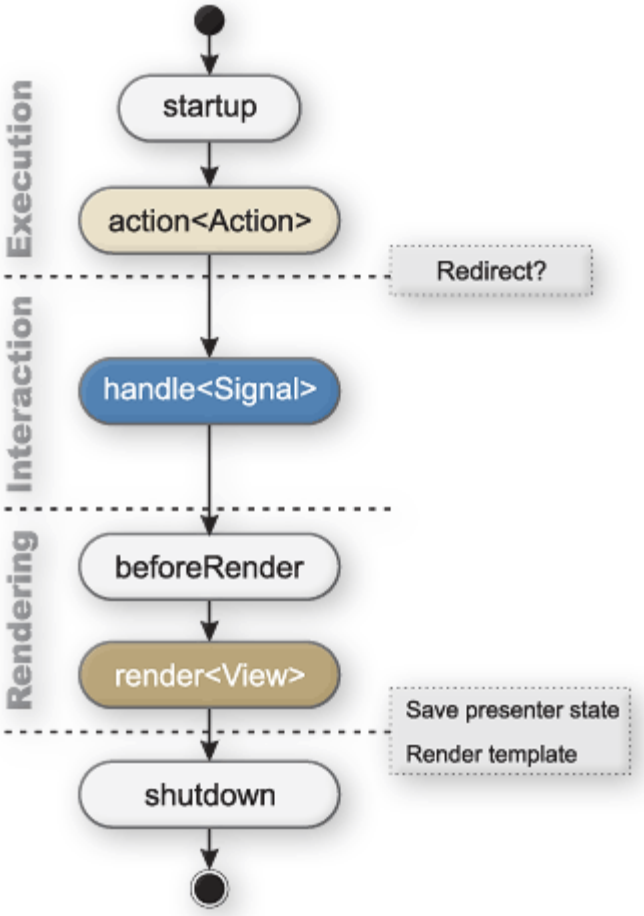
Presenter je komponent, který reaguje na události pocházející od uživatele a zajišťuje změny v modelu nebo pohledu. Tato vazba je mnohem silnější než u MVC architektury. Uživatelské akce se dělí do tří kategorií:

- změna view (nejčastěji),
- změna stavu (interakce v rámci aktuálního view),
- příkaz pro model.



Obrázek 5: Schéma MVP architektury [8]

Životní cyklus presenteru



Obrázek 6: Životní cyklus presenteru [9]

startup() – Tato metoda je volána ihned po vytvoření presenteru. Slouží k inicializaci proměnné nebo ověření uživatelského oprávnění. [9]

action<Action>() – Obdoba metody render<View>. Tato metoda má za úkol provést určitý úkon (přihlášení uživatele, zápis do databáze aj.) a poté přesměrovat jinam. Na rozdíl od metody render nic nevykresluje. [9]

handle<Signal> - Metoda zpracovává subrequesty. Určeno pro komponenty a zpracování AJAXových požadavků. [9]

beforeRender() – Většinou obsahuje nastavení šablony a předání proměnných společných pro více view. [9]

render<View> – Slouží k předání šabloně určitých dat. [9]

shutdown() – Je vyvolána při ukončení životního cyklu presenteru. [9]

3 NETTE FRAMEWORK

Nette framework je open source framework s plným využitím objektů (OOP – objektově orientované programování) a je založen pro tvorbu webových aplikací. [10]

Vznikl v roce 2004, autorem Nette frameworku je David Grudl, další rozvoj má na starost organizace Nette Foundation. Kolem Nette frameworku vyrostla jedna z největších a nejaktivnějších komunit českých PHP vývojářů. Aktuální verze Nette je 2.4. [11]

Nette framework je používán společnostmi jako jsou třeba T-Systems, GE Money, Mladá fronta, VLTAVA-LABE-PRESS, Internet Info, DHL, Logio, ESET [10]. Okolo Nette probíhá pravidelné školení, tzv. "Poslední soboty". Jde o setkání Nette vývojářů. [9]

3.1 Vlastnosti Nette

Základem Nette je opětovná použitelnost kódu, jeho základní jednotkou je komponenta. Má za cíl být zcela nezávislá na okolí (stránce či webu). Z toho vyplývá, že jednou napsanou komponentu můžeme používat stále dokola. [12]

Výhody:

- strmá křivka učení,
- ladící nástroje (Laděnka),
- zabezpečení,
- nejaktivnější komunita v České republice [12],
- SEO (Search Engine Optimalization).

Nette Framework vede programátora k využívání následujících technologií a přístupů:

AJAX – Asynchronous Java Script and XML – Moderní webové aplikace běží napůl na serveru a napůl v prohlížeči. AJAX je klíčovým spojovacím prvkem. [13]

KISS – Keep it simple stupid („Zachovej to jednoduché, hlupáku!") – Jde o jednoduchý návrhový vzor. Tvůrce si klade za cíl udržet aplikaci co možná nejjednodušší a minimalistickou z důvodu snadné a dlouhodobé udržitelnosti. [14]

DRY – Don't repeat your self ("Neopakuj se") – Veškerý kousek logiky by měl mít v aplikaci pouze jednu jednoznačnou reprezentaci, což vede programátora ke znovuvyužití napsaného kódu. [14]

3.1.1 Zabezpečení

Čím je webová aplikace větší a významnější, tím častěji bývá napadena, proto je třeba si dávat velký pozor na bezpečnost. Každou chvíli si lze z různých zdrojů přečíst, že byla nalezena další bezpečnostní díra, kterou hackeři využili. Nette framework je ohledně bezpečnosti znatelně napřed oproti ostatním frameworkům. Zanedbání bezpečnosti může zapříčinit mnoho problémů, od ztráty dat až ke smazání celé webové stránky. [17]

Cross – Site Scripting (XSS)

Je metoda, která napadá webové stránky tak, že zneužívá neošetřené výstupy. Útočník zde může do stránky vložit vlastní kód, čímž může stránku pozměnit nebo získat citlivé údaje o uživatelích. Lze se bránit pouze ošetřením všech řetězců.

Nette framework přichází s revoluční technologií Context – Awareescaping, která tohle riziko řeší. Všechny výstupy ošetřuje automaticky, proto se nikdy nestane, že by kodér na něco zapomněl. [17]

```
{$message}
```

Obrázek 7: Ochrana výstupů [17]

Zápis {$message} znamená výpis proměnné. Ochrana spočívá v převádění všech nebezpečných znaků na jejich textové entity. Tomuto se říká tzv. escapování. V Nette programátor nemusí nic ošetřovat, makro pro výpis proměnných tuto funkci volá za programátora.

Cross – Site Request Forgery (CSRF)

Útok, který je založen na tom, že donutí uživatele navštívit stránku vykonávající útok na webovou aplikaci, kde je uživatel zrovna přihlášen. Hacker následně může pozměnit či smazat článek bez vědomí uživatele. Pro tento atak je nutné dobře znát strukturu napadené aplikace. Ochrana spočívá v generování a ověřování autorizačního tokenu. [17]

V Nette framework lze ochránit formulář za pomocí příkazu:

```
$form->addProtection();
```

Obrázek 8: Příkaz pro ochranu formuláře [17]

Pomocí příkazu na obrázku č. 8 je formulář chráněn pomocí jednorázového autorizačního tokenu. Token je přidán do skrytého formuláře, jenž obsahuje všechny parametry, které identifikují uživatelskou akci. Před vykonáním určitého požadavku je tento token odeslán. Pokud se odeslaný a očekávaný token shodují, je akce schválena a token zahozen. Je doporučeno takto ochránit všechny formuláře nacházející se v administrační části webové aplikace.

URL attack, control codes, invalid UTF 8

Všechny tyto pojmy souvisí se snahou útočníka podstrčit do webové aplikace škodlivý vstup. Ten pak může zapříčinit mnoho chyb, od poškození XML výstupu až po získání citlivých informací uživatele. Aby k tomu nedošlo, je nutné ošetření všech vstupů na úrovni jednotlivých bajtů. Jednou z nejčastějších ochran proti tomuto útoku je generování a ověřování autorizačního tokenu. Díky tomu má útočník velmi ztížen tzv. URL attack.

Nette framework naštěstí tohle řeší automaticky za Vás a všechny vstupy ošetří. [17]

Session hijacking, session stealing, session fixation

Správa session není nic jednoduchého a je známo několik typů útoků. Hacker může podstrčit uživateli své session ID, které zapříčiní, že získá přístup do webové aplikace bez toho, aby znal heslo uživatele. Lze se tomu vyhnout pouze správnou konfigurací serveru a PHP. [17]

Toto řeší Nette framework tak, že PHP nakonfiguruje automaticky za Vás. Jediné, co musí programátor udělat, je povolit funkci: ini_set()

Pro větší bezpečnost a pro odstranění bezpečnostních děr je nutné pravidelně aktualizovat Nette. Dále je doporučeno schovávat aktuální verzi frameworku použitou ve webové aplikaci. Další doporučení spočívá v nevkládání parametrů do SQL (Structured Query Language) a vytvoření white listů, kde je definované, co uživatel může zadat.

Nevýhoda frameworků je ta, že je lze velmi snadno odhalit, například pomocí error 500 (server error), pokud není upraven.

3.1.1.1 SSL certifikát a HTTPS protokol

SSL certifikát, respektive vrstva vložená mezi vrstvu transportní a aplikační, slouží k zabezpečení komunikace mezi klientem a severem. SSL funguje na principu asymetrické šifry. Každá ze stran, které vzájemně komunikují, má dvojici šifrovacích klíčů: veřejný a soukromý. [27]

Veřejný klíč je vystaven za pomocí certifikátu, který zveřejňuje Certifikační autorita. Pokud je tímto klíčem zašifrována nějaká zpráva, tak je zajištěno její rozkódování pouze soukromým klíčem majitele použitého veřejného klíče. [28]

Ustavení SSL spojení:

1. Klient zašle požadavek severu na SSL spojení.
2. Server na klientův požadavek zašle odpověď, která obsahuje certifikát serveru.
3. Na základě této odpovědi si klient ověří autentičnost serveru.
4. Díky dosud obdrženým informacím si klient vygeneruje základ šifrovacího klíče, pomocí kterého budou následující komunikaci šifrovat.
5. Server použije svůj soukromý klíč, aby rozšifroval základ klíče. Ze základu se vygeneruje hlavní klíč.
6. Server i klient navzájem potvrdí, že budou ke komunikaci používat tento klíč. [27]

SSL certifikát je možné poznat z pohledu uživatele tak, že URL serveru začíná se zkratkou HTTPS://.

HTTPS (Hypertext Transfer Protocol Secure) je internetový komunikační protokol, který chrání před odposloucháváním/sledováním nebo modifikací obsahu. HTTPS šifruje data mezi klientem a serverem. Odeslaná data za pomocí protokolu HTTPS jsou zabezpečena prostřednictvím protokolu TLS (Transparent Layer Security). [26]

TLS obsahuje tři hlavní vrstvy ochrany:

Šifrování – šifruje přenesená data proti odposlechu. Díky tomu může uživatel bezpečně sledovat webové stránky, aniž by někdo mohl sledovat jeho aktivitu, "odposlouchávat" jeho konverzaci nebo ukrást údaje.

Integrita dat – Nelze pozměnit nebo poškodit data během přenosu, aniž by to bylo detekováno.

Ověření – Jde o potvrzení, že uživatel opravdu komunikuje s požadovanými webovými stránkami.

Pro správné použití HTTPS musí vývojář správně nakonfigurovat server. Přesměrování všech adres lze pomocí souboru .htaccess, který se nachází v kořenovém adresáři aplikace. Je docíleno permanentního přesměrování s kódem 301.

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    ...
    RewriteCond %{HTTPS} off
    RewriteRule .* https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
    ...
</IfModule>
```

Obrázek 9: Ukázka přesměrováni HTTPS

3.1.2 SEO a kanonizace

Nette Framework přispívá k SEO (optimalizaci nalezitelnosti na internetu). Funguje to tím způsobem, že Nette zabraňuje existenci duplicitních URL adres vedoucích na stejný obsah. Pokud vede více cest k jednomu cíli, Framework první z nich určí za výchozí (kanonickou) a ostatní na ni přesměruje pomocí http kódu 301. Z toho důvodu vyhledávač stránky zaindexuje pouze jednou a nerozmělní jejich page rank. [25]

Výše popsanému postupu se říká kanonizace. Výchozí URL adresa je první cesta, kterou vygeneruje Router bez příznaku ONE_WAY. Příznak ONE_WAY značí jednosměrné cesty. Jednosměrné cesty se používají pro zpětnou kompatibilitu se starými URL adresami, protože tyhle cesty už neslouží pro generování URL adres.

Kanonizace probíhá za pomocí presenteru, ve výchozím nastavení je zapnutá.

3.2 Debugování a zpracování chyb

Nette Framework používá vlastní knihovnu Tracy na zpracování a debugování chyb. Tato knihovna postupem času zdomácněla pod jménem Laděnka. [29]

Laděnka má za cíl:

- rychle opravit či odhalit chyby,
- logovat chyby,
- vypisovat proměnné,
- měřit čas.

Je známo, že PHP dává vývojářům velkou volnost ohledně programování. Proto je důležité mít dobrý ladící nástroj. Tím je v tomhle případě Tracy. [29]

Instalace

Laděnka vyžaduje PHP verzi 5.4.4 nebo vyšší. Instalace probíhá pomocí nástroje composer přes příkazovou řádku zadáním následujícího příkazu:

composer require tracy/tracy

Použití

Laděnka se aktivuje přidáním do kódu, hned za načtení frameworku příkazem:

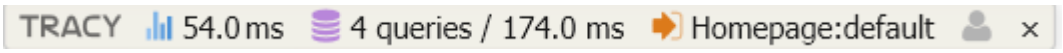
```
use Tracy\Debugger;

Debugger::enable(); // aktivujeme Laděnku
```

Obrázek 10: Přidání a povolení Laděnky

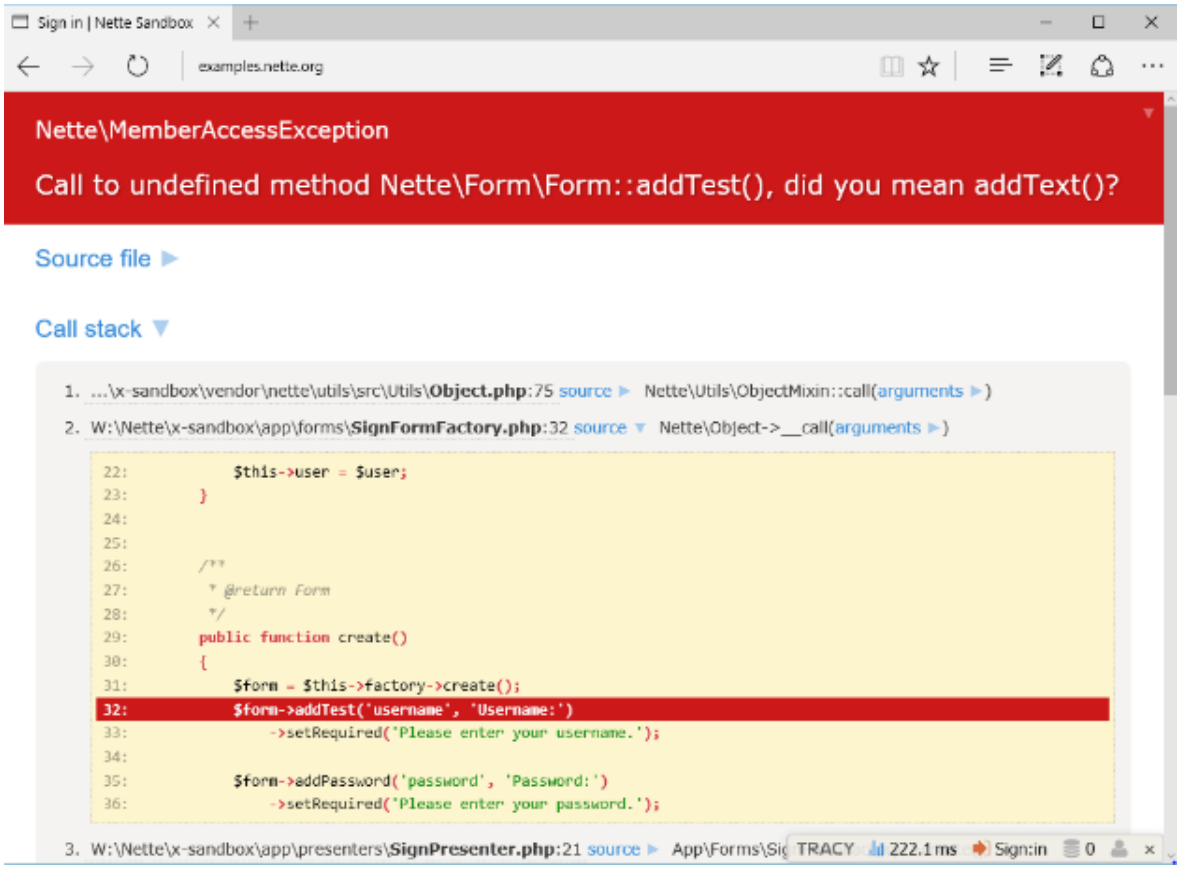
Debugger Bar

Je plovoucí panel, který se defaultně zobrazuje v pravém dolním rohu. Po přesunu myší na určité místo si svoji pozici zapamatuje a po znovu načtení stránky tam zůstane. [29]



Obrázek 11: Debugger Bar

V doplňcích se dají stáhnout rozšíření pro tento panel, případně si podle potřeby může uživatel napsat vlastní.



Obrázek 12: Nezachycená výjimka v provedení Tracy

Pokud má uživatel zapnutou Laděnku a v kódu se vyskytne nějaká chyba, tak se tato chyba nebo výjimka zobrazí v chybové hlášce, která je velmi důrazně zvýrazněna a označuje druh chyby a kde ji lze najít. Na obrázku č. 12 lze vidět, že je chyba zvýrazněná červeně na řádku č. 32. [29]

Zde jsou popsány jenom základy tohoto debugovacího nástroje, pro větší přehled je důležité si pročíst dokumentaci na stránkách https://nette.org/cs/.

4 ELEKTRONICKÉ OBCHODOVÁNÍ

Elektronické obchodování, neboli e-komerce, je souhrnné označení pro veškeré elektronické obchodování. Samostatné elektronické obchodování můžeme považovat za jednu ze součástí elektronického podnikání. Pomocí internetu lze prodávat fyzické zboží, digitální služby nebo informace.

Existují čtyři základní modely elektronického obchodování. Modely jsou určeny pomocí směru marketingových komunikací na:

- B2B (Business to Business) – od firem k firmám, jedná s o obchodování zaměřené na jiné firmy.
- B2C (Business to Customer) – od firem ke spotřebitelům, jedná se o obchodování zaměřené na konečné zákazníky.
- C2B (Customer to Business) – od spotřebitele k firmám.
- C2C (Customer to Customer) – od spotřebitele ke spotřebiteli (aukce, bazary).

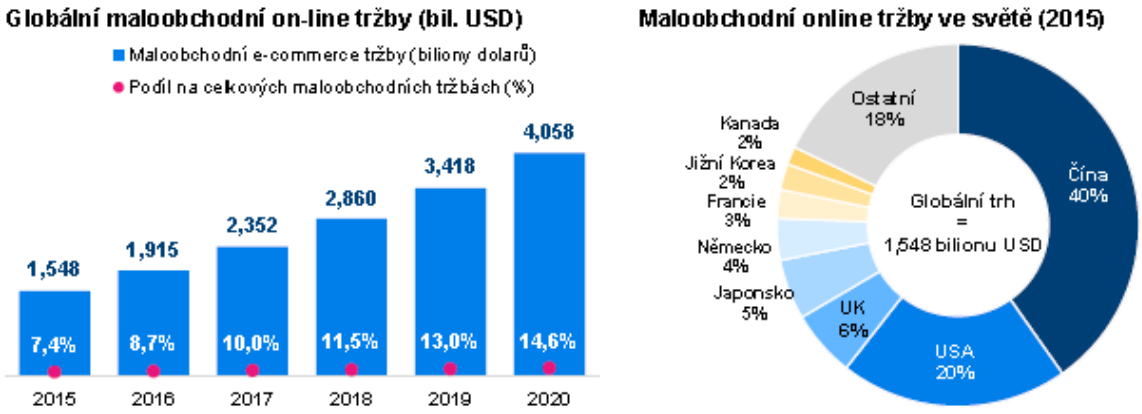
Mezi e-komerci patří i převážná většina činností spadajících pod elektronický marketing, například online reklama, email marketing. Do e-komerce lze zařadit i webové stránky nabízející konkrétní produkt či službu a umožňující jejich objednávku. [22]

4.1 Internetový obchod

E-shop je základní a zdaleka nejvíce používanou formou e-komerce. Internetový obchod je především zaměřen na prodej typu B2C, zřídka se vyskytují e-shopy zaměřené na B2B. Obchodník v internetovém obchodě nabízí zboží, zákazník si ho prochází, následně vkládá do imaginárního nákupního košíku a poté si vybrané zboží může objednat. [24]

Majitel musí splnit základní pravidla internetového obchodu. Na stránkách uvést údaje o prodávajícím, dále reklamační řád, nákupní řád, způsob a dobu dopravy zboží. Nesmí zapomenout na platební podmínky.

Nejčastějším způsobem platby je dobírka, české e-shopy ale nabízejí i platby bankovním převodem, platební kartou nebo elektronickou peněženkou (PayPal, GoPay). Internetový obchod má také za povinnost uvést, jakým způsobem bude zacházet s osobními údaji, které zákazník firmě sděluje. [23]



Obrázek 13: Podíl online nákupu na celosvětovém trhu [23]

Vlevo na obrázku č. 13 je zobrazen předpokládaný vývoj online trhu do roku 2020. Vpravo na obrázku č. 14 lze vidět podíl online nákupu na globálním obchodním trhu. Největším světovým online trhem je Čína. Na druhém místě se umístily USA, které měly o polovinu nižší tržby. [23]

4.2 Elektronická peněženka

Elektronická peněženka je služba, kde si uživatel vytvoří virtuální účet se svojí identitou. Na tento účet si uživatel nejprve převede peníze ze svého bankovního účtu, se kterými bude platit za služby či nakoupené zboží. Elektronické peněženky minimalizují nebezpečí odcizení údajů o platební kartě, jelikož při transakci se využívá pouze přihlašovacího jména a hesla. Platba prostřednictvím elektronické peněženky je zpravidla velmi rychlá. Uživatel má přístup do administračního rozhraní, kde vidí stav účtu či proběhlé transakce. Nejznámější elektronické peněženky jsou PayPal, GoPay, PaySec. V další kapitole bude podrobněji popisován platební systém PayPal, který byl využit pro účely této bakalářské práce. [22]

4.2.1 Platební systém PayPal

PayPal je internetový platební systém, který umožňuje přesuny peněz mezi účty PayPalu. Podle nejaktuálnějších zdrojů je to už více než deset let nejbezpečnější platební rozhraní. Tato platební peněženka je v dnešní době nejvýznamnější platební systém na celém světě, používaný ve 190 zemích světa a podporující platby ve 25 měnách. [18]

V České republice je PayPal zpřístupněn od roku 2006. PayPal byl založen roku 1998 v USA. Zakladatelů je více, nejznámější z nich se jmenuje Elon Musk. Tato společnost je nejvíce známa ve spolupráci s největším aukčním serverem eBay. [19]

Účty jsou propojeny s platebními kartami. Obecně řečeno jde tedy o prostředníka mezi kupujícím a prodávajícím, například e-shopem, na kterém je nakupováno. Tato platební síť je nejvíce vhodná pro internetové obchodníky, malé a střední podniky.

Systém umožňuje nastavit primární cenu, která bude uživatelem použita, mezi nimi je i česká koruna. Při platbě v jiné měně je částka přepočítána dle aktuálního kurzu. [20]

Výhody:

- Platby jsou velmi rychlé – peníze jsou na PayPal účet připsány téměř okamžitě.
- Bezpečnost – informace se přenášejí zašifrovaně a při jednotlivých platbách nejsou informace o platební kartě k dispozici.
- Jednoduchost.
- Nastavení limitu pro internetové platby.
- Intuitivnost při placeni.
- Mezinárodní podpora, celosvětová rozšířenost.

Nevýhody:

- Při krádeži účtu může daná osoba prostřednictvím účtu napadeného uživatele nakupovat, z toho důvodu je nutné použít silné heslo, popřípadě ho pravidelně měnit.
- Webové stránky ani uživatelské rozhraní nejsou poskytované v češtině.
- Vysoké poplatky.

Založení účtu je velmi jednoduché, je potřeba pouze e-mailová adresa, přihlašovací jméno a heslo. Dále zadáte platební kartu, pomocí níž můžete platit na internetu, a potřebné údaje pro online nákupy, aby nedošlo k opětovnému zadávání při každé transakci. [21]

PayPal má tři varianty platebních účtů:

Personal Account (osobní účet) – slouží k online nakupování, posílání a přijímání plateb. Na tento účet nelze přijímat platby, jež jsou provedeny debetní nebo kreditní kartou. Platební operace jsou bez poplatků.

Premier Account (zvýhodněný účet) – účet vhodný pro nákupy a prodej na aukčním serveru eBay, platby za zboží a služby v e-shopech. Pomocí účtu můžete přijímat všechny platby, avšak nachází se zde poplatky za přijaté platby.

Bussines Account (obchodní účet) – účet sloužící především pro online obchodování, akceptuje kreditní karty, debetní karty a platby z platebních účtů. Poplatky se nachází ve stej-

né výši jako u Premier účtu. Poplatky se platí pouze u přijímaní plateb, u odesílání se žádné neplatí. Informace o nich se nachází na oficiálních stránkách platební služby PayPal. Obecně platí, že se poplatky nachází v rozmezí 1,9 – 4,9 % dle velikosti peněžité částky, kterou prodávající obdržel. S vyšším prodejem se poplatky snižují. [21]

4.2.1.1 Zabezpečení

Platební rozhraní PayPal je zabezpečené pomocí SSL/HTTPS spojení a šifrování. SSL certifikát a HTTPS je popsán v kapitole 3.1.1.1. Zabezpečení samotného účtu proti napadení záleží na uživateli a bezpečnostní úrovni zvoleného uživatelského hesla. Dále je veřejně doporučováno nereagovat na e-maily, jež mají za cíl získat uživatelské heslo účtu. Uživatel reaguje na takový e-mail na vlastní nebezpečí. Majitel účtu, by měl při takovém počínání ihned informovat uživatelskou PayPal podporu. [21]

II. PRAKTICKÁ ČÁST

5 VÝBĚR TECHNOLOGIE

V teoretické části byly popsány vlastnosti, technologie a architektura vývojového frameworku Nette spolu se službou PayPal. V této části bude popsán postup vývoje rozhraní pro internetové nákupy pomocí služby PayPal.

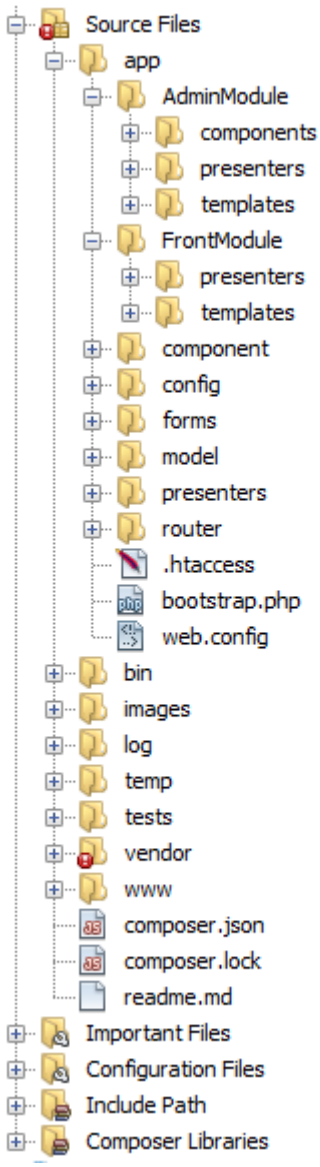
Jako hlavní nástroj pro programování internetového obchodu byl zvolen Nette framework. Tento framework byl zvolen díky jeho vysoké bezpečnosti, ale i přehledné dokumentaci v českém jazyce. Avšak hlavním důvodem bylo nedávné setkání autora s tímto typem frameworku v jeho pracovním životě, a také autorova touha po dalším zdokonalení se v této oblasti. Dále bude využit PayPal API (Application Programming Interface) pro implementaci a zprovoznění platebního systému PayPal.

Celá aplikace je psána ve vývojovém prostředí NetBeans, toto IDE (Integrated Development Environment) je zcela zdarma, použitelné i pro komerční využití. Jako softwarový webový server pro testování byl zvolen Apache. Webhosting, na němž aplikace běží, se jmenuje OneBit.

Byly použity následující technologie:

- MySQL 5.0.12
- PHP 7.0.13
- Nette 2.4

6 SANDBOX



Obrázek 14: Adresářová struktura
[Zdroj: vlastní]

Pomocí nástroje composer byl v konzoli nainstalován základní skeleton aplikace, známý pod jménem Sandbox, který již obsahuje Nette framework. V dalším kroku byly odstraněny zbytečné soubory.

Na obrázku č. 14 je možné vidět rozložení navrhované webové aplikace, která je rozdělena na administrační část (AdminModule) a uživatelskou část (FrontModule). V každé z částí jsou vlastní třídy presenterů a šablon.

Ve složce "config" se nacházejí konfigurační soubory, kde jsou nastaveny přístupy do databáze apod. Složka "router" obsahuje třídu, pomocí níž se nastavuje konfigurace URL

adresa. Složka "log" obsahuje chybové logy, "temp" obsahuje dočasné soubory (cache, session)."Vendor" je knihovna, kde se nachází Nette framework, popřípadě jiné knihovny pro webovou aplikaci. Adresář "www" slouží pro ukládání obrázků, CSS stylů a dalších souborů, jako jediný je přístupný z internetu.

7 POŽADAVKY PRO APLIKACI

Následující podkapitoly popisují nejenom požadavky pro reálnou aplikaci, ale i pro demo aplikaci popsanou v této bakalářské práci.

7.1 Funkční požadavky na demo aplikaci

V této kapitole jsou popsány scénáře a povolené uživatelské i administrátorské akce.

7.1.1 E-shop

Registrace uživatele – Uživatel, jehož cílem je nakoupit, se musí nejprve registrovat. Při registraci vyplní základní osobní údaje a adresu doručení.

Přihlásit uživatele – Přihlášení probíhá zadáním e-mailové adresy a hesla, které si uživatel zvolil při registraci. Každému registrovanému uživateli je přidělena role 1, která definuje uživatele jako klienta.

Zobrazit jednotlivé produkty – Přihlášený uživatel má možnost vybrat si na hlavní stránce produkt a zobrazit si jeho konkrétní údaje. V detailu produktu se zobrazí cena, název a popis produktu.

Vložit produkty do košíku – Pro nákup zboží klient musí nejprve vložit produkty do košíku pomocí tlačítka zobrazující se v detailu.

Zobrazit produkty v košíku – Přihlášený klient je po kliknutí na ikonu košík, jež se nachází v menu, přesměrován do nákupního košíku, kde je mu vykreslena cena a zvolený počet kusů. Zde si může vybrat z více postupů.

Mazat produkty v košíku – Uživatel může smazat produkty uložené v košíku, pokud si je tam nadále nepřeje mít uložené.

Rekapitulace objednávky – Pokud se uživatel nachází v košíku a přeje si dokončit objednávku, potvrdí tlačítko "Rekapitulace". Následně se mu zobrazí kontaktní údaje a detail objednávky, přitom se vytvoří objednávka v databázi.

Platba objednávky – Pokud si uživatel přeje zaplatit, klikne na tlačítko PayPal, které ho přesměruje na platební bránu PayPal. Klient zadá své přihlašovací údaje, potvrdí platbu, čímž ji dokončí a je znovu přesměrován na úvodní stránku internetového obchodu. Objednávka je v databázi označena jako zaplacená. Pokud by však platbu nedokončil a pomocí tlačítka "Cancel" zrušil, objednávka by se označila jako stornovaná.

Odhlásit se – Pokud si uživatel přeje odhlásit, klikne na svoje uživatelské jméno nacházející se v menu, zde se mu zobrazí submenu "Odhlásit se". Po kliknutí na tlačítko je uživatel odhlášen a přesměrován na úvodní stránku.

7.1.2 Administrace

Administrace je oddělena od uživatelského prostředí e-shopu. Na konec URL adresy se za lomítko napíše "admin". Celková URL adresa je tedy ve tvaru https://www.beeup.cz/admin.

Přihlásit administrátora – Přihlášení probíhá zadáním e-mailové adresy a hesla, které bylo administrátorovi přiřazeno v databázi. Každý administrátor má nastavenou roli 2.

Zobrazit produkty – Přihlášený administrátor si může zobrazit výpis všech produktů nacházejících se v databázi.

Přidat nový produkt – Pomocí tlačítka "Nový produkt" se vykreslí formulář, kde administrátor vyplní údaje nového produktu.

Upravit produkt – Po kliknutí na tlačítko "Upravit" lze aktualizovat staré produkty, například cenu, název nebo popis produktu.

Smazat produkt – Tlačítko "Smazat" slouží k odstranění produktu z databáze.

Zobrazit objednávky – V menu je možné vybrat "Objednávky", které slouží k vykreslení všech objednávek v databázi.

Zobrazit uživatele – Zobrazí všechny uživatele v databázi, včetně administrátorů.

Odhlásit se – Po dokončení činnosti v administraci se administrátor může odhlásit. Je to nutné zejména tehdy, když administrátor není přihlášený ze svého zařízení a kdokoliv by toho mohl zneužít.

7.2 Požadavky pro reálnou aplikaci

Pro reálnou aplikaci a plné využití administrace je nutné dodělat:

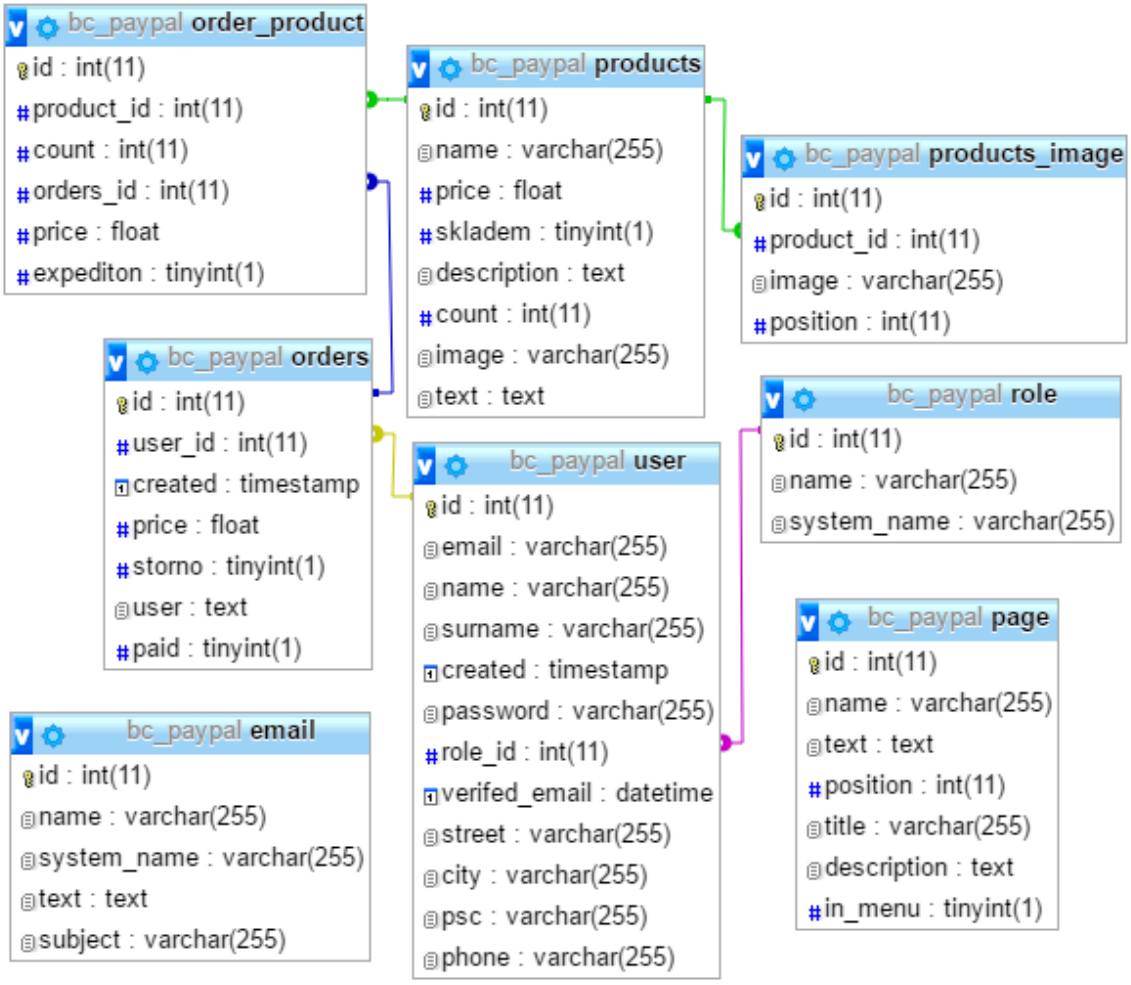
- přidat, smazat a upravit uživatele,
- ruční stornování, popřípadě úprava objednávky,
- generování elektronické faktury.

Legální elektronický obchod musí obsahovat:

- kontaktní formulář,
- obchodní podmínky,
- reklamační řád,
- zpracování osobních údajů,
- dodací podmínky,
- odstoupení od smlouvy.

8 NÁVRH DATABÁZE

Databáze byla navržena tak, aby splnila veškeré funkční požadavky a mohla simulovat základní procesy, které se dějí v internetovém obchodu, např. registrace uživatele či zaplacení za zboží. Databáze je navržena v prostředí phpMyAdmin.



Obrázek 15: Návrh databáze[Zdroj: vlastní]

Tabulka **user** – obsahuje informace o uživateli. Pro zjednodušení se počítá, že adresa doručení je stejná jako adresa bydliště. Tabulka obsahuje i statut uživatele, který je spojen pomocí cizího klíče s tabulkou statutu, ve kterém jsou konkrétní role definované. Vazba mezi uživatelem a objednávkou je 1:N, protože jeden uživatel může mít více objednávek.

Tabulka **role** – slouží k definici rolí použitých v aplikaci. Nastavené jsou pouze dvě – Administrátor a Klient. Pokud by se měla aplikace rozšířit, je možné přidat další, například administrátora, jehož práva by byla omezená za pomocí autentizace.

Tabulka **products** – ukládá veškeré produkty v internetovém obchodě, jež klient může nakoupit.

Tabulka **order_product** – slouží jako mezitabulka pro produkty a objednávku, kde se ID produktu přiřazuje k ID objednávky.

Tabulka **orders** – obsahuje vygenerované všechny objednávky. Vazba mezi objednávkou a produktem je typu M:N, jelikož jedna objednávka může obsahovat více produktů a zároveň, lze jeden produkt objednat ve více objednávkách.

Další tři tabulky jsou nevyužité, ale jsou obsahem této práce jako příklad, jak by návrh pokračoval.

Tabulka **page** – slouží k ukládání statických stránek, například stránky pro kontaktní formulář.

Tabulka **products_image** – Pokud má produkt více náhledových obrázků, jsou uloženy v "products_image". Vazba je typu 1:N, protože jeden produkt může mít více obrázků.

Tabulka **email** – slouží k zasílání internetové pošty uživatelům, například přihlašovací údaje, zapomenuté heslo, či informace o objednávce.

9 REALIZACE

Zde budou popsány důležité části vytvoření aplikace.

9.1 FrontModule

Nejdůležitějším presenterem je BasePresenter, od něhož dědí vlastnosti všechny ostatní presentery. O zobrazení produktů na úvodní stránce a o jeho detail se stará HomepagePresenter. Přidávání, mazání a o další práce s košíkem se stará tzv. BasketPresenter, v němž se nachází i komponenta vykreslující tlačítko PayPal, které má za úkol přesměrovat uživatele do platebního systému PayPal. Přihlášení a registraci uživatele má na starost UserPresenter.

9.1.1 Přihlášení a registrace

Uživatel musí být pro nákup v aplikaci přihlášený. Z toho důvodu bude popsána část kódu pro vytvoření přihlašovacího formuláře.

```
protected function createComponentFormLogin($name) {
    $form = new Form($this, $name);
    $form->addText('email', 'Email')
            ->addRule(Form::FILLED, 'Zadejte email!')
            ->addRule(Form::EMAIL, 'Musí exitovat')
            ->setAttribute('class', 'form-control')
            ->setAttribute('placeholder', 'Email');

    $form->addPassword('password', 'Heslo')
            ->addRule(Form::FILLED, 'Zadejte heslo!')
            ->setAttribute('placeholder', 'Heslo')
            ->setAttribute('class', 'form-control');

    $form->addSubmit('send', 'Přihlásit se');
    $form->onSubmit[] = [$this, 'submitFormLogin'];
    return $form;
}
```

Obrázek 16: Formulář pro přihlášení uživatele [Zdroj: vlastní]

Metoda createComponentFormLogin() vytváří formulář pro přihlášení uživatele. Vytváří instanci třídy UI/Form a přidává položky formuláře.

AddText() vytvoří textové pole pro zadání e-mailu, ke kterému jsou nastavena pravidla.

AddPasword() vytváří pole pro zadání hesla.

AddSubmit() vykreslí tlačítko pro odeslání formuláře.

OnSubmit() zavolá metodu pro zpracování formuláře.

V šabloně je formulář vykreslen pomocí makra:{control formLogin}.

```
public function submitFormLogin(Form $form) {
    $values = $form->getValues();
    try {
        $this->authenticator->login($values['email'], $values['password']);

        $this->redirect('Homepage:default');
    } catch (AuthenticationException $e) {
        $this->flashMessage('Špatné jméno nebo heslo', 'error');
    }
}
```

Obrázek 17: Metoda pro zpracování formuláře [Zdroj: vlastní]

Na obrázku č. 17 je vidět metoda pro zpracování formuláře. Metoda dostává jako parametr instanci formuláře. Pomocí metody getValues() získává data a pokusí se uživatele přihlásit, pokud se bude shodovat uživatelské jméno a heslo. Jestli je autentizace úspěšná, přesměruje uživatele na domovskou stránku, kde může vybírat produkty k nákupu.

V případě neúspěšného přihlášení, Nette zachytí autentizační výjimku a vypíše nastavenou zprávu.

```
public function processForm(Form $form) {
    $values = $form->getValues();

    $this->users->insert(array(
        'email' => $values['email'],
        'name' => $values['name'],
        'surname' => $values['surname'],
        'created' => new DateTime,
        'password' => Passwords::hash($values['password']),
        'role_id' => $this->roles->clientId,
        'street' => $values['street'],
        'city' => $values['city'],
        'psc' => $values['psc'],
        'phone' => $values['phone'] == '' ? null : $values['phone'],
    ));
    $this->authenticator->login($values['email'], $values['password']);
    $this->flashMessage('Jste přihlášen');
    $this->redirect('Homepage:default');
}
```

Obrázek 18: Metoda pro zpracování registračního formuláře [Zdroj: vlastní]

Vykreslení registračního formuláře pracuje na stejném principu, avšak ve zpracování probíhá nahrání hodnot, které metoda obdrží do databázové tabulky "User". Registrovaný uživatel je zároveň hned přihlášen do aplikace, aby neztrácel čas a mohl se věnovat nákupu.

9.1.2 Košík

Přidávání a mazání produktů z košíku nebo dokončení objednávky se nachází v BasketPresenteru, který dědí od BasePresenteru. V BasePresenteru se nachází metoda, která získává počet a obsah produktů a následně tento obsah posílá do šablony.

```
public function actionAdd($id) {
    $session = $this->getSession('basket');
    $product = $this->products->get($id);
    if (!$product) {
        $this->flashMessage('Produkt nenalezen');
        $this->restoreRequest($this->backLink);

        $this->redirect('Basket:default');
    }
    $productId = $id;

    if (!isset($session->basket[$productId])) {
        $session->basket[$productId] = 1;
    } else {
        $session->basket[$productId] ++;
    }
    $this->flashMessage('Produkt přidán do košíku');
    $this->restoreRequest($this->backLink);
    $this->redirect('Basket:default');
}
```

Obrázek 19: Metoda pro vložení zboží do košíku [Zdroj: vlastní]

Metoda přebírá parametr ID produktu, který je vkládán do košíku. Získá session s názvem basket. Načte si z databáze pomocí metody get() celý řádek, který odpovídá ID produktu. Následně zkontroluje, jestli daný produkt existuje, pokud ne, vypíše chybovou hlášku. Následuje kontrola, zda do session už byl přidat nějaký produkt, pokud ne, je vložen na pozici 1. Jestli už nějaký produkt košík obsahuje, tak se následující produkt vloží na další pozici.

9.2 AdminModule

Nejdůležitějším presenterem je BasePresenter, od něhož dědí vlastnosti všechny ostatní presentery, což jsou AdminPresenter, OrderPresenter, ProductPresenter, UserPresenter a SignPresenter. Každý z presenterů se stará o samostatnou část aplikace. Přihlášení funguje

na stejném principu jako u FrontModule s tím rozdílem, že role administrátora má ID nastavené na hodnotu 2.

```
public function createComponentFormNew($name) {
    $form = new Form($this, $name);
    $form->addText('name', 'Název')
            ->addRule(Form::FILLED, 'Název musí být vyplněn');
    $form->addText('price', 'Cena')
            ->addRule(Form::FILLED, 'Text musí být vyplněn');
    $form->addCheckbox('skladem', 'Skladem');
    $form->addTextArea('description', 'Popis')
            ->addRule(Form::FILLED, 'Popis musí být vyplněn');
    $form->addText('count', 'Pocet');
    $form->addUpload('image','Obrázek');
    $form->addTextArea('text', 'Text');
    $form->addSubmit('send', 'Odeslat');

    $form->onSuccess[] = [$this, 'submitFormNew'];
    $form->addProtection('Vypršel časový limit, odešlete formulář znovu');
    return $form;
}
```

Obrázek 20: Vkládání nového produktu [Zdroj: vlastní]

Na obrázku výše je kód pro vytvoření formuláře, který zajišťuje vkládání produktu do databáze. Metoda createComponentFormLogin() vytváří formulář pro přihlášení uživatele. Vytváří instanci třídy UI/Form a přidává položky formuláře, ke kterým je přidána validace.

```
public function submitFormNew(Form $form) {
    $values = $form->getValues();

    $this->row = $this->products->insert(array(
        'name' => $values['name'],
        'price' => $values->price == '' ? null : (float) $values->price,
        'skladem' => $values->skladem,
        'description' => $values['description'],
        'count' => $values->count == '' ? null : (float) $values->count,
        'image' => $values['image'] == '' ? NULL : $values['image'],
        'text' => $values['text'],
    ));
    $name = 'pic' . $this->row['id'];
    $this->saveImage($values);
    $this->row->update(array(
        'image' => $name . '.jpg',
    ));
    $this->flashMessage('Údaje vloženy');
    $this->redirect('Product:viewProducts');
}
```

Obrázek 21: Zpracování formuláře pro nový produkt [Zdroj: vlastní]

Metoda pro zpracování formuláře dostává jako parametr instanci formuláře. Poté získá data z vyplněného formuláře a vloží je do databáze. Do proměnné je uloženo jméno produktu, které je tvořeno z jeho názvu a ID. Pomocí funkce je obrázek uložen a následně je aktualizováno jeho jméno v databázi. Po aktualizaci vypíše požadovaný text a přesměruje na defaultní stránku pro vykreslení produktů.

9.3 Zabezpečení HTTPS

```
<IfModule mod_rewrite.c>
        RewriteEngine On

        RewriteCond %{HTTPS} off
        RewriteRule .* https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

        RewriteRule ^$ www/ [L]

        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteCond %{REQUEST_FILENAME} !-d
        RewriteCond %{REQUEST_URI} !^www/
        RewriteRule ^(.*)$ www/$1
</IfModule>
```

Obrázek 22: Přesměrování HTTPS [Zdroj: vlastní]

Pro používání HTTPS protokolu je nutné mít správně nakonfigurovaný server. Přesměrování všech adres lze docílit pomocí souboru .htaccess, který se nachází v kořenovém adresáři aplikace.

Ve třetím a čtvrtém řádku je vidět kód, který slouží k přesměrování s kódem 301. V druhém řádku probíhá kontrola, jestli je HTTPS zapnuté. Pokud není, tak je stránka přesměrována.

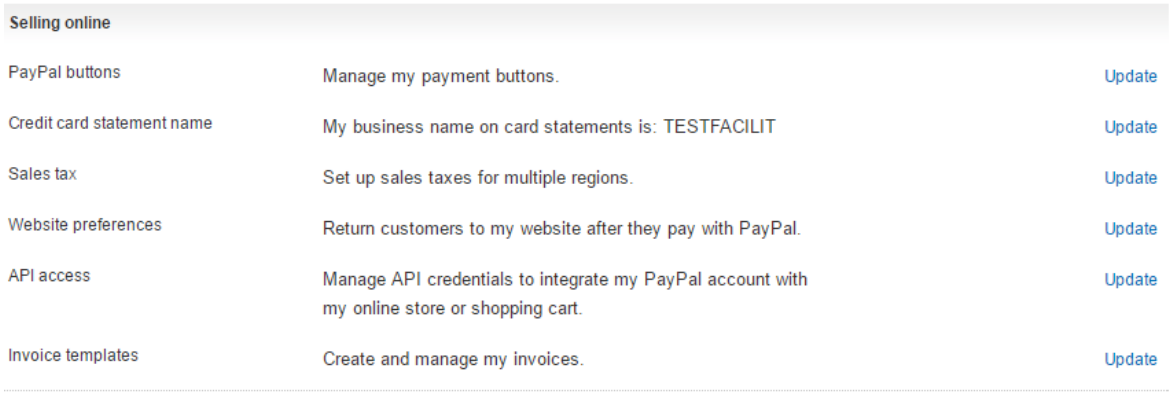
Přesměrování 301 znamená, že je stránka na adresu přesunuta permanentně. HTTPS zabezpečení je pro internetové obchody nutné z hlediska bezpečnosti přihlášení a registrace uživatele. Detailní popis tohoto protokolu je popsán v teoretické části, v kapitole 3.1.1.1.

9.4 Implementace PayPal

Implementace PayPal lze provést dvěma způsoby: buď pomocí API (Application Programming Intefrace) nebo pomocí služby PayPal, kdy je administrátorovi automaticky vygenerované tlačítko.

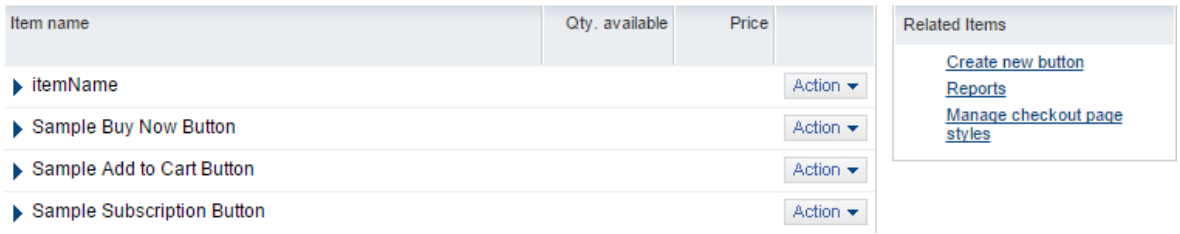
9.4.1 PayPal tlačítko

Nejprve je nutné si vytvořit Business účet na webových stránkách elektronické peněženky. Po přihlášení se zobrazí úvodní stránka, na ní v bočním panelu uživatel vybere „My selling tools“. Poté se zobrazí panel pro nejrůznější nastavení.



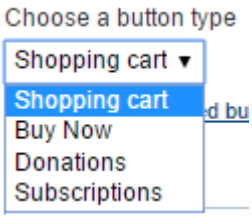
Obrázek 23: Selling tools [32]

Pro vygenerování tlačítka pro spárování internetového obchodu se službou PayPal klikneme na PayPal buttons „Update“.



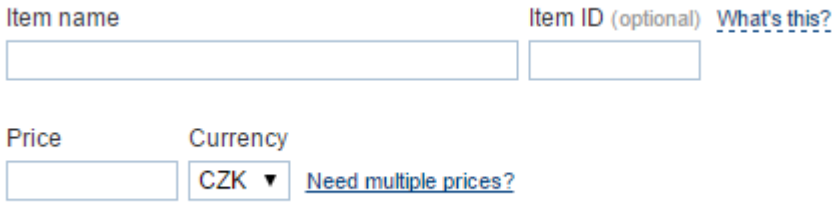
Obrázek 24:Selling tools Update [32]

Na obrázku č. 24 vybereme „Create new button“. Zobrazí se stránka pro vytvoření tlačítka. Nachází se zde nejrůznější nastavení.



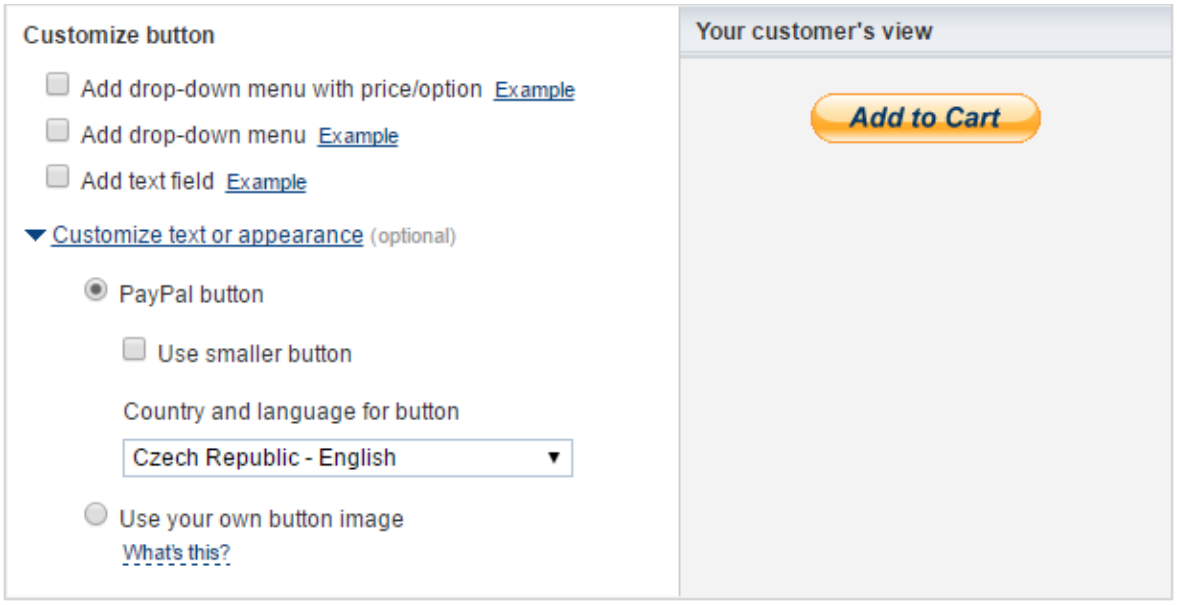
Obrázek 25: Výběr typu tlačítka [32]

Pro vytvoření nákupního košíku vybereme „Shopping cart“.



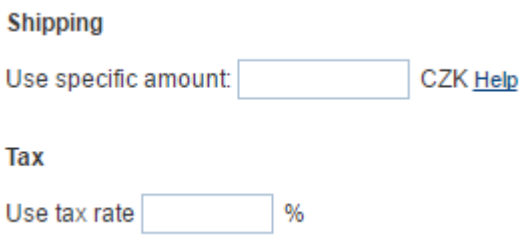
Obrázek 26: Jméno a cena produktu [32]

Do kolonky „Item name“ se nastaví název produktu, do kolonky „Price“ požadovaná cena a měna, ve které chce majitel internetového obchodu prodávat.



Obrázek 27: Volitelné nastavení [32]

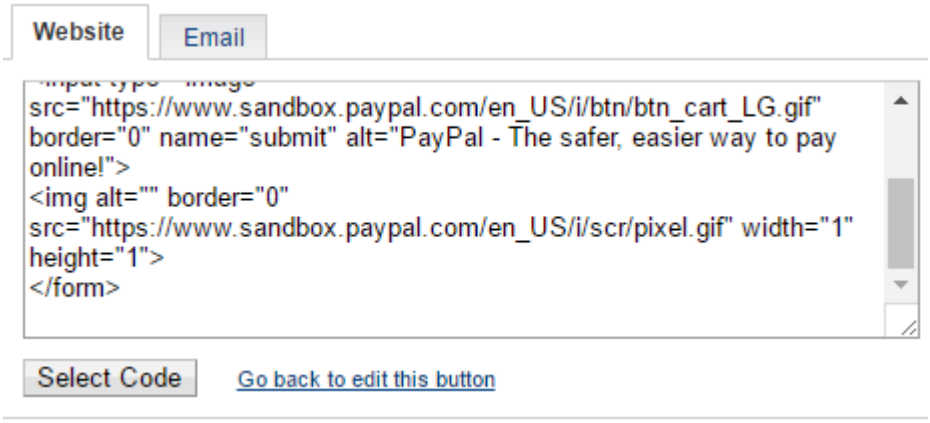
Na obrázku č. 27 vidíme volitelné nastavení, kde uživatel může nastavit například vyskakovací menu pro nastavení ceny, použití menšího tlačítka nebo jazyk pro nápis vykreslující se na tlačítku a další.



Obrázek 28: Cena za doručení a DPH [32]

Na obrázku výše lze nastavit cenu za doručení objednávky a DPH v procentech. Obě tyto možnosti jsou volitelné.

Nastavení potvrdíme kliknutím na tlačítko „Create Button“. Následně se vygeneruje HTML (Hyper Text Markup Language) kód, který stačí zkopírovat do webové aplikace na místo, které je požadováno pro nákup.



Obrázek 29: Vygenerované HTML pro tlačítko [32]

Výhody této implementace:

- jednodušší.
- rychleji implementovatelné.

Nevýhody:

- menší kontrola nad platbou.

9.5 Implementace API

Vybrané API je stáhnuto ze stránky https://github.com/MetisFW/PayPal. Toto API bylo doporučeno na nejrůznějších fórech zabývající se platebním systémem PayPal. Autor API uvádí i přehlednou dokumentaci v anglickém jazyce.

Toto API je použito ve webové aplikaci, jež je právě popsána. Rozšíření je nainstalováno pomocí composeru. Pro instalaci je nalezena za pomocí konzole cílová složka, ve které se zadá příkaz composer require metisfw/paypal. Všechny potřebné nástroje a knihovny se stáhnou do složky "vendor".

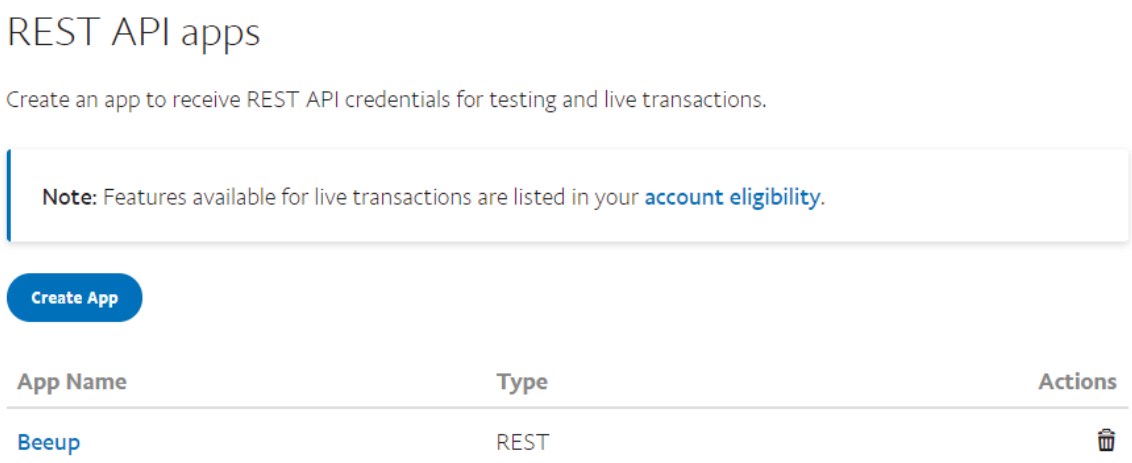
```
payPal:
  clientId: AUCNf6HcPcOZ-EWRk70zQPbl1WpFLu...YOUR_CLIENT_ID
  secret: EOB_ka6eAcDlsAHJJBu8kWO3G2r7wL_5...YOUR_SECRET_KEY
  currency: CZK
  # optional Payment Experience Profile ID
  # https://developer.paypal.com/docs/api/payment-experience/
  # experienceProfileId: XP-AAAA-BBBB-CCCC-DDDD
  sdkConfig:
    mode: sandbox
    log.Enabled: true
    log.FileName: '%tempDir%/PayPal.log'
    log.LogLevel: DEBUG
    validation.level: log
    cache.enabled: true
    # 'http.CURLOPT_CONNECTTIMEOUT' => 30
    # 'http.headers.PayPal-Partner-Attribution-Id' => '123123123'/

extensions:
  payPal: MetisFW\PayPal\DI\PayPalExtension
```

Obrázek 30: Nastavení v souboru config.neon [Zdroj: vlastní]

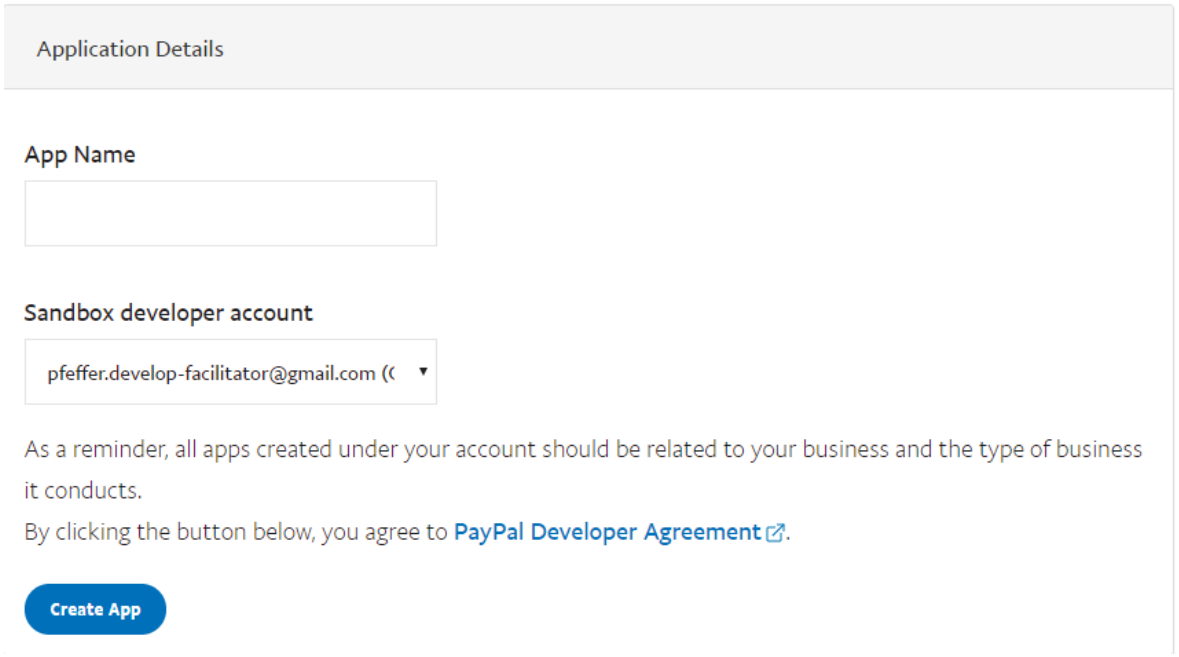
Do souboru config.neon, nacházející se ve složce "config", jsou přidány řádky z obrázku č. 31.

Přihlásíme se do vývojového prostředí platební služby PayPal, jež se nachází na stránce https://developer.paypal.com/ pomocí údajů, kterými jsme se registrovali. Na této stránce získáme „clientId“ a „secret“, měnu si můžeme nastavit dle libosti. Zkratky jsou uvedeny v dokumentaci k API.



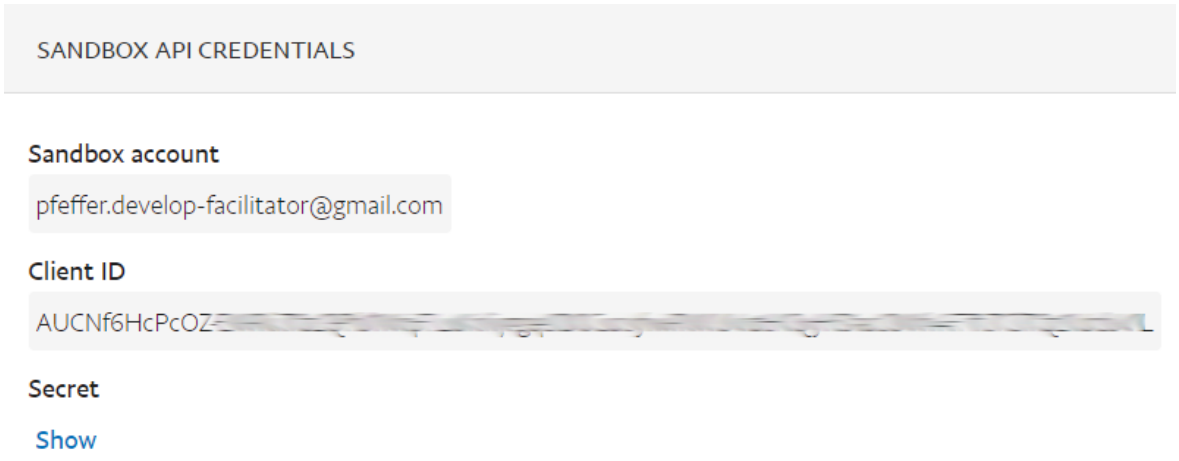
Obrázek 31: REST API [33]

Na úvodní stránce se zhruba v polovině nachází tlačítko „Create App“, na které klikneme.



Obrázek 32: Nastavení emailu a jména API [33]

Zobrazí se stránka, kde se nastavuje jméno API a business účet, na který se budou zasílat peníze po obdržení platby. Klikneme na tlačítko „Create App“.



Obrázek 33: ClientId a secret [33]

Zde vidíme vygenerované „ClientId“ a „Secret“. Oba údaje se zkopírují a vloží na příslušné místo do souboru config.neon, jak bylo uvedeno výše. „Secret“ se zobrazí po stisknutí odkazu „Show“. Toto jsou údaje vygenerované pro testování tzv. „sandboxu“.

Obrázek 34: Přepnutí na Live verzi [33]

Pro přejití na tzv. Live verzi, kde dochází k reálným platbám, stačí přepnout pomocí přepínače „Sandbox/Live“. V pravém horním rohu přepneme na tlačítko „Live“, jak je vidět na obrázku výše. Údaje jsou rozdílné, proto si uživatel musí dávat pozor, jestli chce pouze testovat platby, anebo platit reálnými penězi.

```php
public function createComponentPayPalPaymentButton() {
    $sessionBasket = $this->getSession('basket');
    $allPrice = $this->orders->getPrice($sessionBasket->basket);
    $operation = $this->factory->create('Objednávka-Beeup', $allPrice);
    $control = new PaymentControl($operation);

    //called after successfully completed payment proccess
    $control->onSuccess[] = function(PaymentControl $control, Payment $paid) {
        $order = $this->orders->get($this->getOrderId());
        $order->update(array(
            'paid' => true,
        ));
        $sessionBasket = $this->getSession('basket');
        $sessionBasket->basket = array();
        $this->flashMessage('Platba proběhla úspěšně');
    };

    //called when user cancelled payment process
    $control->onCancel[] = function(PaymentControl $control) {
        $order = $this->orders->get($this->getOrderId());
        $order->update(array(
            'storno' => true,
        ));
        $this->flashMessage('Objednávka byla zrušena');
    };
    return $control;
}
```

Obrázek 35: Vytvoření tlačítka pro zaplacení [Zdroj: vlastní]

Na obrázku č. 35 je zobrazena ukázka kódu z BasketPresenteru, kde je vytvořeno tlačítko pro zaplacení objednávky.

Získáme session, ve které máme uložené produkty. Zavoláme metodu getPrice0 a získáme aktuální cenu objednávky. Předáme parametry továrničce pro vytvoření tlačítka. Následně vytvoříme tlačítko. Pokud je platba úspěšná, tak v metodě onSuccess[] nahrajeme do databáze příslušné objednávky do kolonky "paid" hodnotu 1 a vypíšeme text „Platba proběhla úspěšně“.

Jestli uživatel zruší platbu, tak v metodě onCancel[] nastavíme v databázi u příslušné objednávky v kolonce "storno" hodnotu 1 a vypíšeme, že objednávka byla zrušena.

Tlačítko se vykresluje v příslušné šabloně, za pomocí makra {control payPalPaymentButton}

Výhody:

- větší kontrola nad aplikací,
- mnohem více možností zpracování.

Nevýhody:

- složitější,
- zdlouhavé.

ZÁVĚR

Cílem bakalářské práce bylo vytvořit rozhraní pro uskutečnění on-line nákupů a plateb pomocí služby PayPal v jazyce PHP a vývojového frameworku Nette.

Práce se skládá ze dvou částí. V první části byly popsány základní pojmy a vlastnosti související s Nette frameworkem. Byly charakterizovány návrhové vzory používané u webových aplikací. Konkrétně byla rozebrána architektura MVC a MVP, bylo poukázáno na jejich rozdíly a podobnosti. Jedna z kapitol se věnuje přímo Nette frameworku a popisuje jeho vlastnosti, výhody a zabezpečení, které je na vysoké úrovni. V samostatné kapitole došlo ke srovnání PHP frameworků na základě četnosti použití a rychlosti. Závěr teoretické části se zabývá elektronickým obchodováním, konkrétně internetovým obchodem. Podkapitola se věnuje elektronickým peněženkám, konkrétně systému PayPal, popisuje jeho výhody, nevýhody, typy účtů a zabezpečení.

V druhé, praktické části byly popsány technologie, který byly použity pro tvorbu internetového obchodu. Představeny byly i užité nástroje. Ukázána byla adresářová struktura, která byla použita pro samotný vývoj, a důležité soubory nacházející se uvnitř. Před samotnou tvorbou se uskutečnil sběr požadavků pro demo aplikaci. Samozřejmostí je popis požadavků, které musí obsahovat reálný obchod v internetovém provozu. Pomocí ukázky kódu byly popsány důležité části aplikace, jako je například přihlášení, přidávání produktů do košíku apod. Implementace platebního systému PayPal proběhla ve vlastní kapitole, rozdělena je na dvě části. První část se věnuje popisu generování samostatného tlačítka pro košík pomocí platebního systému PayPal. V druhé části je popsán způsob implementace za pomocí API, který byl v rámci ukázkové aplikace také zvolen.

Dle obou metod implementace platebního systému bylo shledáno, že pro malé internetové obchody, kde uživatel nepotřebuje mít absolutní kontrolu nad platebním systémem, je dostačující použít PayPal vygenerované tlačítko. Tato metoda je rychlejší a jednodušší. Pro implementaci jsou dostačující pouze základní znalosti PHP a HTML. Naopak, pokud uživatel stojí o téměř absolutní kontrolu nad platebním systémem, měl by použit API, popřípadě napsat si vlastní, podle svých požadavků na funkčnost. Tato metoda je náročnější a delší. Pro zabezpečení internetového obchodu je nutno využít protokolu HTTPS. To se provádí v souboru .htacess. Ač je postup jednoduchý, z hlediska nutnosti byl důkladně popsán. Výsledek bakalářské práce je dočasně zpřístupněn na adrese www.beeup.cz.

SEZNAM POUŽITÉ LITERATURY

[1] Úvod do Nette frameworku pro PHP. *IT network* [online]. 2015 [cit. 2017-05-04]. Dostupné z: https://www.itnetwork.cz/php/nette/zaklady/uvod-do-php-frameworku-nette/

[2] What is a framework. *Codeproject* [online]. United States, 2003 [cit. 2017-05-04]. Dostupné z: https://www.codeproject.com/Articles/5381/What-Is-A-Framework

[3] Seriál návrhových vzorů – 1. díl. *Programujte.com* [online]. Czech republic, 2012 [cit. 2017-05-04]. Dostupné z: http://programujte.com/clanek/2012032900-serial-navrhovych-vzoru-1-dil/

[4] Návrhové vzory v PHP / Nette. *Egoblog.cz* [online]. Czech republic, 2016 [cit. 2017-05-04]. Dostupné z: http://www.egoblog.cz/navrhove-vzory-v-php-nette/

[5] 10 nejlepších PHP frameworků pro vývojáře. *Interval.cz* [online]. Czech republic, 2016 [cit. 2017-05-04]. Dostupné z: https://www.interval.cz/clanky/10-nejlepsich-php-frameworku-pro-vyvojare/

[6] BÖHMER, Marian. Návrhové vzory v PHP: [23 vzorových postupů pro rychlejší vývoj]. Brno: ComputerPress, 2012. ISBN 978-80-251-3338-5.

[7] MVC architektura. *IT network* [online]. Česká republika, 2014 [cit. 2017-05-07]. Dostupné z: https://www.itnetwork.cz/navrhove-vzory/mvc-architektura-navrhovy-vzor/

[8] Prezentační vzory z rodiny MVC. *Zdroják.cz* [online]. Česká republika, 2009 [cit. 2017-05-07]. Dostupné z: https://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/

[9] Model-View-Presenter (MVP). *Nette* [online]. Česká republika, 2010 [cit. 2017-05-07]. Dostupné z: https://doc.nette.org/cs/2.4/presenters#toc-zivotni-cyklus-presenteru

[10] Nette. *PhpFashion* [online]. Česká republika [cit. 2017-05-15]. Dostupné z: https://phpfashion.com/category/nette/

[11] Nette Framework: zvyšte svoji produktivitu. *Zdroják.cz* [online]. Česká republika, 2009 [cit. 2017-05-07]. Dostupné z: https://www.zdrojak.cz/clanky/nette-framework-zvyste-svoji-produktivitu/

[12] Nette-zaciname-aneb-motivace. *PhpFashion* [online]. Česká republika, 2006 [cit. 2017-05-07]. Dostupné z: https://phpfashion.com/nette-zaciname-aneb-motivace

[13] Ajax. *Nette* [online]. Česká republika, 2015 [cit. 2017-05-07]. Dostupné z: https://doc.nette.org/cs/2.4/ajax

[14] Slovníček. *Péháprkáři* [online]. Česká republika [cit. 2017-05-07]. Dostupné z: https://pehapkari.cz/slovnicek/

[15] The Most Popular Framework of 2015. *Sitepoint* [online]. Česká republika, 2015 [cit. 2017-05-07]. Dostupné z: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

[16] Performance benchmark of popular PHP frameworks. *Systems architect* [online]. USA, 2013 [cit. 2017-05-07]. Dostupné z: https://systemsarchitect.net/2013/04/23/performance-benchmark-of-popular-php-frameworks/

[17] Zabezpečení před zranitelnostmi. *Nette* [online]. Česká republika, 2015 [cit. 2017-05-07]. Dostupné z: https://doc.nette.org/cs/2.4/vulnerability-protection

[18] WILLIAMS, Damon. Pro PayPal E-Commerce. New York, N.Y.: Distributed to the-Book trade worldwide by Springer-Verlag, c2007. ISBN 1-59059-750-8.

[19] PayPal krok za krokem. *Penize* [online]. Česká republika, 2014 [cit. 2017-05-07]. Dostupné z: http://www.penize.cz/nakupy/293651-paypal-krok-za-krokem-jak-jednoduse-platit-na-internetu

[20] PayPal recenze. *Forex srovnávač* [online]. Česká republika, 2017 [cit. 2017-05-07]. Dostupné z: http://www.forexsrovnavac.cz/paypal

[21] *MILLER, Michael. The PayPal official insider guide to growing your business: make money the easy way. ISBN 9780321768520.*

[22] E-Commerce. *Management mania* [online]. Česká republika, 2016 [cit. 2017-05-07]. Dostupné z: https://managementmania.com/cs/e-commerce

[23] E-COMMERCE: SVĚTOVÝ OBCHOD ONLINE. *EDotace* [online]. Česká republika, 2016 [cit. 2017-05-07]. Dostupné z: http://www.edotace.cz/clanky/e-commerce-svetovy-obchod-online

[24] Slovník pojmů. *Shoptet* [online]. Česká republika, 2015 [cit. 2017-05-15]. Dostupné z: https://www.shoptet.cz/slovnik-pojmu/?letter=E

[25] Routování URL *Nette* [online]. Česká republika, 2010 [cit. 2017-05-07]. Dostupné z: https://doc.nette.org/cs/2.4/routing#toc-seo-a-kanonizace

[26] HTTPS. *Google* [online]. 2016 [cit. 2017-05-08]. Dostupné z: https://support.google.com/webmasters/answer/6073543?hl=cs

[27] Co je to SSL certifikát? *Domain Master* [online]. 2015 [cit. 2017-05-08]. Dostupné z: https://www.domainmaster.cz/napoveda/caste-dotazy/ssl-certifikaty/co-je-ssl-certifikat-108/

[28] *OPPLIGER, Rolf. SSL and TLS: theory and practice. Boston: Artech House, c20016. Artech House information security and privacy series.*

[29] Debugování a zpracování chyb. *Nette* [online]. Česká republika, 2016 [cit. 2017-05-08]. Dostupné z: https://tracy.nette.org/cs/

[30] LECKY-THOMPSON, Ed a Steven D. NOWICKI. PHP 6: Programujeme profesionálně. Brno: ComputerPress, 2010. ISBN 978-80-251-3127-5

[31] PRETTYMAN, Steve. Learn php 7: object oriented modular programming using HTML5, CSS3, Javascript, XML, JSON, and MYSQL. New York: Apress, 2012. ISBN 9781484217290 SHERIF, Mostafa Hashem. Protocols for secure electronic commerce. Third edition. New York: CRC Press, 2016. ISBN 9781482203745

[32] *Sandbox PayPal* [online]. [cit. 2017-05-15]. Dostupné z: https://www.sandbox.paypal.com/businessprofile/mytools

[33] *Developer PayPal* [online]. [cit. 2017-05-15]. Dostupné z: https://developer.paypal.com/developer/applications/

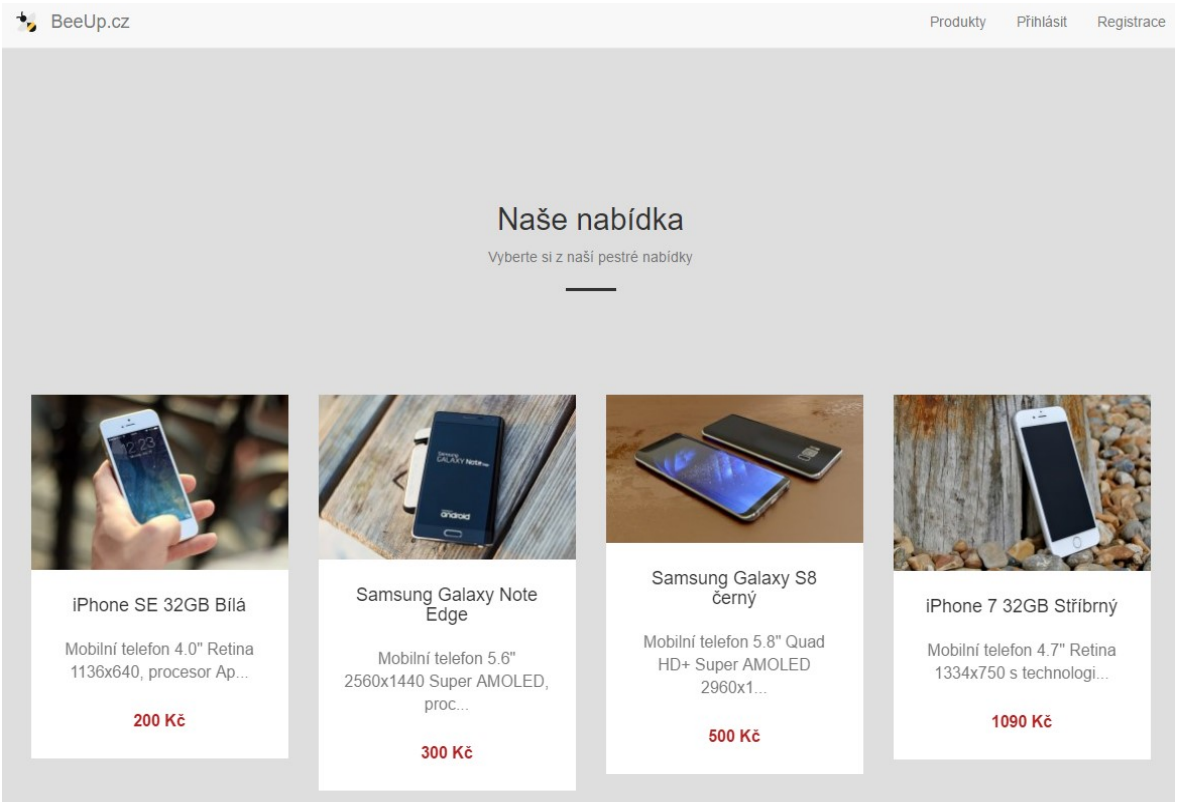
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PHP	Hypertextový preprocesor.
MVC	Model-view-controller.
MVP	Model-view-presenter.
HTML	Hyper Text Markup Language.
SEO	Search Engine Optimalization.
AJAX	Asynchronous Java Script and XML.
CSRF	Cross-site Request Forgery.
ID	Identification Number.
URL	Uniform Resource Locator.
XSRF	Cross-site Request Forgery.
KISS	Keep It Simple, Stupid.
DRY	Don’t Repeat Yourself.
IDE	Inregrated Development Environment.
API	Application Programming Interface.
DPH	Daň z přidané hodnoty.

SEZNAM OBRÁZKŮ

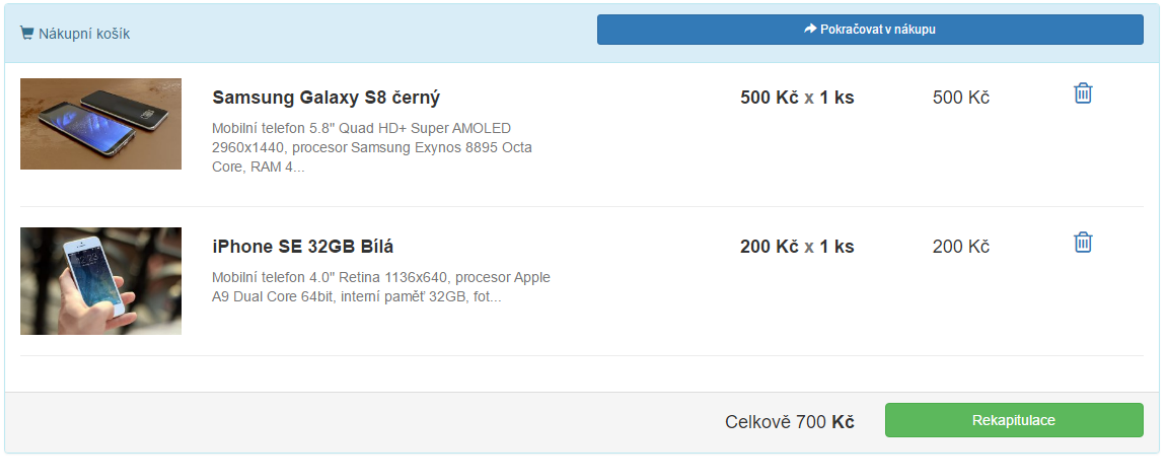
SEZNAM PŘÍLOH

PŘÍLOHA PII: OBRÁZEK ÚVODNÍ STRANY E-SHOPU



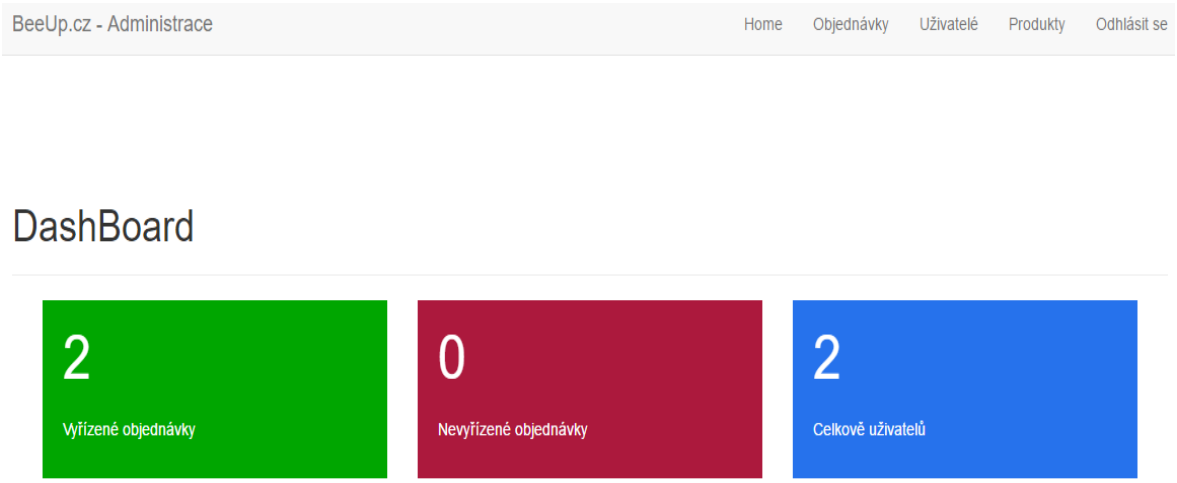
Obrázek 36: Úvodní strana e-shopu [Zdroj: vlastní]

PŘÍLOHA PIII: OBRÁZEK NÁKUPNÍHO KOŠÍKU



Obrázek 37: Nákupní košík [Zdroj: vlastní]

PŘÍLOHA P IV: OBRÁZEK ÚVODNÍ STRANY V ADMINISTRACI



Obrázek 38: Úvodní strana po přihlášení v administraci [Zdroj: vlastní]