Univerzální slevový systém pro e-shopy

Bc. Jan Januš

Diplomová práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

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

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

Jméno a příjmení:	**Bc. Jan Januš**
Osobní číslo:	**A15362**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Univerzální slevový systém pro e-shopy**
Téma anglicky:	**A Universal Discount-Sale System for E-shops**

Zásady pro vypracování:

1. **Prozkoumejte současnou situaci a možnosti implementace slevových systémů pro e-shopy.**
2. **Definujte požadavky a uživatelské cíle na slevový systém dle aktuálních marketingových trendů.**
3. **Popište technologie a nástroje potřebné k vytvoření slevového systému a seznamte se s nimi.**
4. **Navrhněte a implementujte vhodné řešení slevového systému podle definovaných požadavků.**
5. **Otestujte slevový systém na vybraném systému a zhodnoťte dosažené výsledky. Zaměřte se na bezpečnost.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **MACDONALD, Matthew, Adam FREEMAN a Mario SZPUSZTA. ASP.NET 4 a C# 2010 - KNIHA 1 - tvorba dynamických stránek profesionálně. Zoner Press, 2011. ISBN 978-80-7413-131-8.**
2. **CASTAGNETTO, Jesus, Harish RAWAT, Sascha SCHUMANN, Chris SCOLLO a Deepak VELIATH. PHP Programujeme profesionálně: 2. opravné a aktualizo-vané vydání. Vydání první. Hornocholupická 22, 143 00 Praha 4: Computer Press, a.s., 2002. ISBN 80-7226-310-2.**
3. **PROCHÁZKA, David. CSS a XHTML tvorba dokonalých WWW stránek krok za krokem - 2. vydání. Praha: GRADA, 2011. ISBN 978-80-247-3897-0.**
4. **CHAFFER, Jonathan a Karl SWEDBERG. Mistrovství v jQuery. Holandská 8, 639 00 Brno: Computer Press, a.s., 2013. ISBN 978-80-251-4103-8.**
5. **SACK, Joseph. Velká kniha T-SQL & SQL Server 2005 kompendium znalostí pro začátečníky i profesionály. Brno: Zoner Press, 2007. ISBN 978-80-86815-57-2.**

Vedoucí diplomové práce: **doc. Ing. Jiří Vojtěšek, Ph.D.**
Ústav řízení procesů

Datum zadání diplomové práce: **3. února 2017**

Termín odevzdání diplomové práce: **16. května 2017**

Ve Zlíně dne 3. února 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

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

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové 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 práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové 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 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 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 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 práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové 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 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 13.5.2017

.....
podpis diplomanta

ABSTRAKT

Cílem této diplomové práce je vytvoření univerzálního slevového systému pro e-shopy. Práce pracuje se stabilními technologiemi a novými trendy, na jejichž základě postupně buduje funkční a robustní aplikaci, která poskytuje univerzální rozhraní pro nastavování a následné automatizované uplatňování slev. Systém počítá s tím, že jeho provoz bude probíhat v cloudu. K systému lze připojit jakéhokoliv klienta, který je schopen prodávat a má přístup k Internetu. Řešení je postaveno primárně na Microsoft technologiích, které jsou velmi rozšířenou komoditou v korporátním světě. Systém umožňuje nastavit procentuální či hodnotové slevy na košík, které se uplatňují za stanovených podmínek, jejichž příkladem může být obrat košíku. Výhodou je, že vyhodnocení slevových akcí je prováděno on-line, zákazník má tedy ihned přehled o tom, kolik ho bude jeho nákup stát. Správce systému se může spolehnout na automatické mechanismy, které dle nastavených dat řídí platnost akcí či zahájí platnost nové akce. Jednou z dalších výhod je fakt, že aplikace běží na frontendovém frameworku Bootstrap, který si dobře poradí s mobilními zařízeními, tudíž je možné správu slevových akcí provádět jak na desktopu, tak na mobilních zařízeních (dotykové telefony, čtečky knih, tablety). V neposlední řadě lze zmínit jako výhodu jednoduchou implementaci ze strany klientské části, která je nezávislá na technologiích části serverové, společnou částí je pouze komunikace, která probíhá ve formátu JSON. Na základě vlastní implementace klientské části se lze domnívat, že komunikační rozhraní webového API je natolik univerzální, aby pokrylo širokou škálu e-shopových řešení.

Klíčová slova: slevový systém pro e-shopy, uplatnění slev, cloud, responzivní design, MVC návrhový vzor, Microsoft technologie.

ABSTRACT

The aim of this diploma thesis is to create a universal discount system for e-shops. The work is based on stable technologies and new trends, which gradually builds a functional and robust application that provides a universal interface for setting up and subsequently automating discounts. The system count that its operation will take place in cloud infrastructure. Any client who is able to sell and has access to the Internet can connect to the system. The solution is built primarily on Microsoft technologies, which are a very widespread commodity in the corporate world. The system allows you to adjust the percentage or value discounts on the

basket that apply under specified conditions, which may be the turnover of the basket. The advantage is that evaluation of discount events is done on-line, so the customer has an immediate overview of how much his purchase will cost him. System administrator can rely on automatic mechanisms that control the validity of the new action according to the set data. One of the other benefit is that app runs on Bootstrap front-end framework, which can handle mobile device well, so you can manage discount actions on desktop and mobile devices (touch phones, book readers, tablets). Last but not least, it is possible to mention the simple implementation of the client part, which is independent of the technologies of the server part, the common part is only the communication that takes place in JSON format. Based on own implementation of the client part, there is valid assumption, that the API web interface is so universal to cover a wide range of e-shop solutions.

Keywords: discount system for e-shops, discount evaluation, cloud, responsive design, MVC design pattern, Microsoft technologies.

Tímto bych chtěl poděkovat vedoucímu mé diplomové práce doc. Ing. Jiřímu Vojtěškovi, Ph.D. za vedení v průběhu práce, cenné rady pro tvorbu a vylepšení a za všechny starosti spojené s řádným odevzdáním této práce.

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

OBSAH

ÚVOD

Obchod je lidskou činností provozovanou již od pradávna. Obchodovalo se téměř se vším, co bylo poptáváno trhem a také s každým, kdo byl schopen něco nabídnout. Obchodníci se neustále snaží najít či zdokonalit způsob, jak nalákat své potencionální zákazníky k zakoupení jimi nabízeného zboží či služby. Ono nalezení způsobu a oslovení toho pravého zákazníka tím pravým zbožím či službou dalo za vznik moderní vědní disciplíně nazývané marketing.

Marketing ke své funkci využívá řadu nástrojů a metodik, které dopomáhají obchodníkovi ke zvýšení jeho prodejů. Marketing lze vnímat v nákupním centru dnešní doby tak, že si můžete všimnout dvou základních faktorů, na které dnešní maloobchodní prodejci sází. Tím prvním faktorem je propagace novinek a vyzvedávání kvality zboží. Zejména společnost Apple je nyní brána jako již zaběhnuté klišé v oblasti IT, kde nikdo nepochybuje o kvalitě a je tedy schopen si zboží pořídit i za relativně vysokou cenu. Druhým faktorem, který zastává většina ostatních maloobchodních prodejců je poskytování slev na zboží. Tento faktor hraje mnohdy významnou roli v přesvědčení zákazníka, zda koupit či nekoupit. Vznik internetu a vývoj nových technologií v oblasti internetové sítě umožnil obchodníkům přijít na trh s revolučním prodejním kanálem, a tím je internetový obchod, který stejně jako kamenná prodejna využívá uvedené faktory k tomu, aby došlo k přesvědčení zákazníka k dokončení nákupu.

V dnešní době dostávají slevy nový rozměr, a to zejména u velkých hráčů v oblasti digitálního prodeje, kteří jsou schopni vzhledem k velikosti obratu a dostatečné marži, zákazníkům nabídnout výrazné úlevy na cenách nabízeného zboží. Zavádějí se celosvětové dny slev, mezi které lze zahrnout např. Black Friday. Obchodníci mezi sebou soupeří, aby pokryli, co největší část trhu v daném oboru a hledají nové příležitosti a nástroje, které dopomůžou k dosažení požadovaných obchodních cílů. Jakmile navštívíte internetový obchod, stáváte se v danou chvíli potencionálním zákazníkem, a to je pro obchod velká zodpovědnost, proto se musí snažit vám poskytnout maximální zážitek z nákupu a maximální motivaci nákup dokončit. V souvislosti s uvedeným musí být prodejní systém chytře automatizován, zejména v oblasti procesu tvorby objednávky, aby byl schopen navodit zákazníkovi jednoduché a zcela intuitivní prostředí bez nutnosti zvýšené interakce ze strany zákazníka. Je dobré si uvědomit, že zákazník vnímá veškerý obsah, který je mu dostatečně výstižně zobrazen, a to včetně motivačních textů, které zákazníka motivují k realizaci objemnějšího nákupu, čímž

také dochází k naplnění stanovených cílů obchodníka, a to jsou stále vzrůstající tržby. Nový nástroj, jehož vývojem se bude diplomová práce zabývat, zavádí nové možnosti v oblasti automatizovaného poskytování slev na zboží při nákupu v internetovém obchodě. Obchodníkům nástroj poskytne jednoduchou správu slevových akcí, které jsou vyhodnocovány na základě obsahu košíku v e-shopu. V rámci práce bude implementována pouze základní podstata a princip tohoto systému. Vize pro jeho rozvoj jsou omezeny pouze aktuálně nabitými vědomostmi.

I. TEORETICKÁ ČÁST

1 SLEVOVÉ SYSTÉMY

Zkoumání ukázalo, že literární zdroje jsou v této oblasti značně omezené, proto bylo pro tuto část práce využito především webových zdrojů a reálných aplikací. V rámci webové sítě byl proveden průzkum registrovaných patentů v rámci evropské kanceláře patentů, dále byla prozkoumána nabídka trhu digitální komerce v oblasti poskytování slev na zboží.

V rámci průzkumu byly zaznamenány aplikace poskytující slevu zejména pomocí kupónů, tedy krátkých kódu, které jsou ve formátu textovém, číselném nebo jejich kombinace. Tyto kódy jsou schopny při použití v košíku internetového obchodu automaticky aplikovat slevu na zboží obsažené v košíku. Dále lze narazit na herní slevové systémy nebo moduly pro uplatňování slev, které jsou integrovány přímo do konkrétních internetových obchodů.

Během celé etapy zkoumání nebyl nalezen samostatný systém, který by agregoval téma slev do jednoho softwarového produktu a nabízel jej pomocí univerzálního on-line rozhraní internetovým obchodům k využití. Stejně tak nebyl nalezen systém, který by uplatňoval slevy automaticky v závislosti na obsahu košíku či dokonce inteligentně motivoval zákazníka k většímu nákupu.

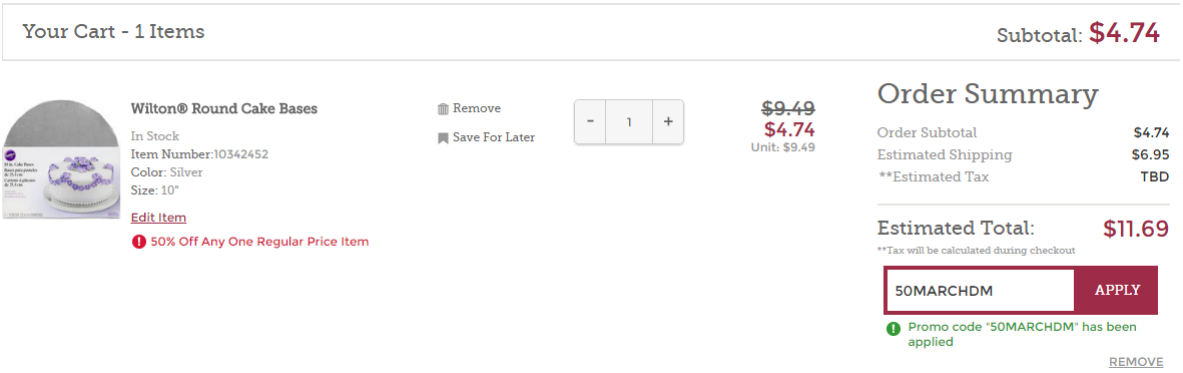
1.1 Kupónový slevový systém

Je jedním ze zkoumaných systémů a funguje na principu vložení slevového kódu do vstupního pole, které je k dispozici v košíku internetového obchodu, pak následuje potvrzení vloženého kódu, což zapříčiní jeho uplatnění na obsah košíku. Během procesu uplatnění je kód verifikován ze strany aplikace, ve které jsou příslušné kódy kupónu uloženy. Pokud jsou splněny všechny podmínky, pak je uplatněna příslušná sleva na košík. Podmínky mohou být doba platnosti kupónu, specifická skupina zákazníků, specifické kategorie zboží, konkrétní zboží a způsob platby nebo doručení.

Na podmínky kupónů lze také pohlížet z úhlu počtu použití, kdy může být definován kupón, který má neomezený počet použití či jednorázový kupón. Využití kupónu s neomezeným počtem použití je pak zejména v oblasti sociálních sítí a obecného marketingu. Typickým příkladem může být slevový kód „KONVICE“, jehož aplikací zákazník dosáhne na slevu uplatněnou pouze na zboží z kategorie varných konvic. Druhým typem je jednorázový kupón, který je ideální použít pro oblast automatizovaného marketingu např. nově registrovaní zákazníci, narozeninová sleva a jiné slavnostní příležitosti. Tyto kódy jsou generovány tak, aby bylo minimalizováno jejich zneužití např. formou hledání posloupnosti v kódech,

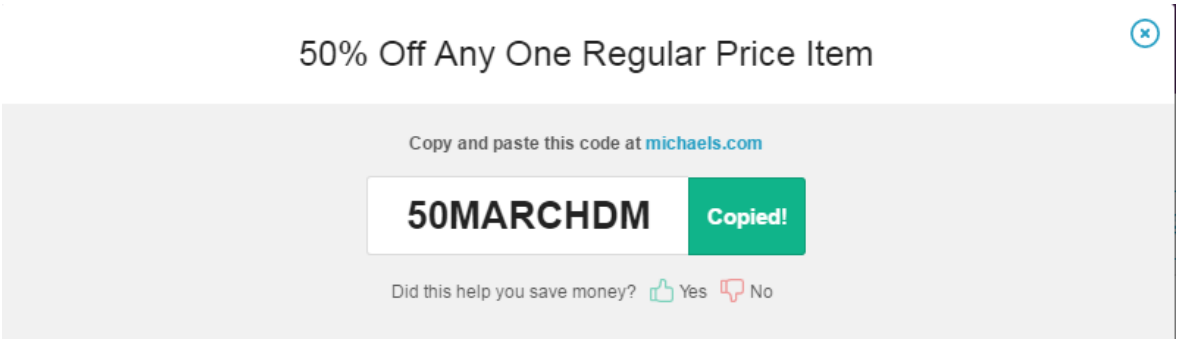
tedy pro jejich tvorbu je často využit náhodný generátor, který při celkovém počtu kombinací a době platnosti zneužití dostatečně minimalizuje.

Výsledkem uplatnění kupónu může být sleva na dopravu, případě doprava zdarma, sleva na košík, automatické přidání produktu zdarma do košíku a podobně. V případě, že je sleva hodnotová a nikoliv procentuální, pak je myšleno také na situace, že hodnota košíku může být menší než hodnota kupónu. V takovém případě je zamezeno jeho použití pomocí definice minimální hodnoty košíku. Ukázku použití kupónu znázorňuje níže uvedený obrázek (*Obr. 1*).



Obr. 1. Ukázka uplatnění kupónu v e-shopu.

Existují také systémy, které agregují univerzální slevové kupóny vydávané internetovým obchodem do jednoho portálu. V tomto portálu si může zákazník vybrat z nabízených slevových kupónů, zkopírovat je a následně být pomocí pár kliknutí přesměrován do internetového obchodu, kde je možné příslušný kupón uplatnit. Příkladem agregátoru, který je znázorněn na obrázku níže (*Obr. 2*), je *www.retailmenot.com*.



Obr. 2. Ukázka agregátoru slevových kupónů.

1.2 Herní slevový systém

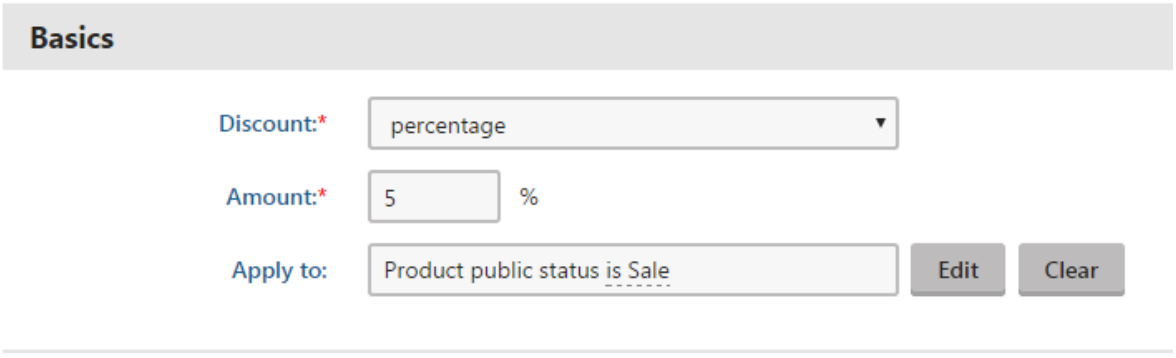
V souvislosti se slevami lze také narazit na koncepty, které uvádějí popis specifické hry pro internetový obchod, kdy postup do dalších úrovní je odměněn formou slev na vybrané zboží. Tento koncept lze brát za jistou formu motivace, jste motivováni být ve hře stále lepší a z tohoto důvodu navštívit častěji internetový obchod, čímž se také zvyšuje pravděpodobnost, že vás při návštěvě zaujme nabízené zboží a nikoliv pouze hra.

1.3 Integrované slevové systémy v internetových obchodech

Tyto slevové systémy jsou vždy součástí platformy internetového obchodu a umožňují použití katalogových slev pro motivaci vybraných zákazníků k uskutečnění nákupu. Například lze nastavit na zboží speciální Vánoční akci s výběrem zvolené kategorie zboží jako například Elektronika, Oblečení a podobně mezi 16. listopadem a 23. prosincem. Mohou být definována pravidla či podmínky, které mohou obsahovat cenové rozpětí produktů, výrobce produktu a podobně.

1.3.1 Kentico

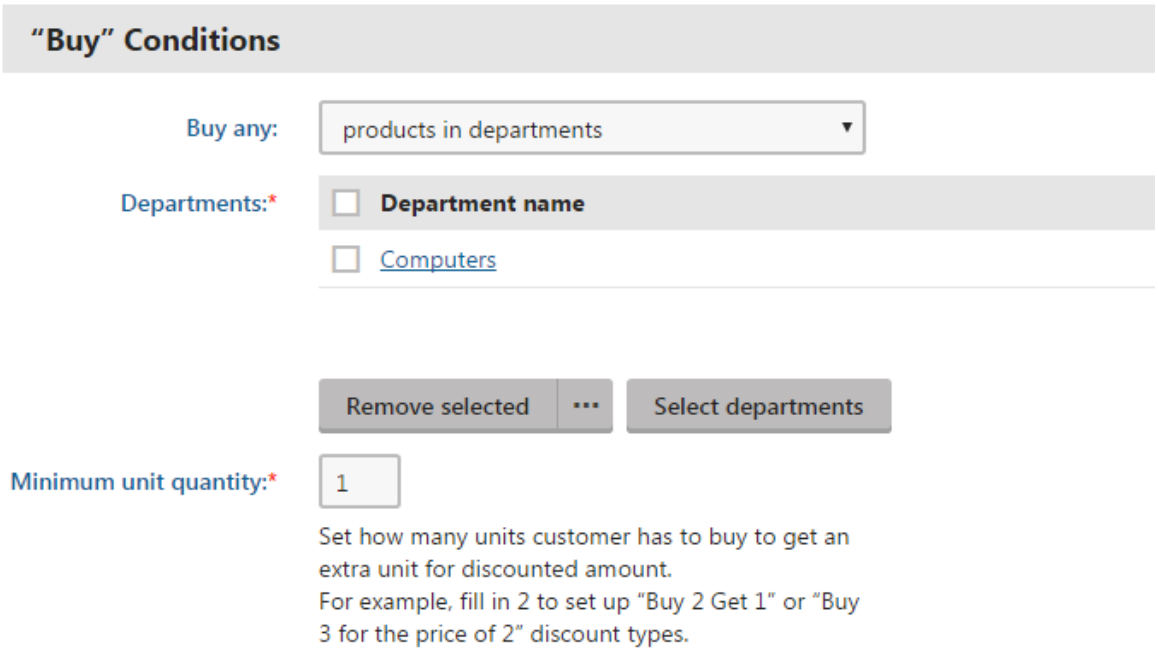
V rámci platformy internetového obchodu Kentico je možné kromě definice kupónů tvořit katalogové slevy, dále nastavit jejich platnost a definovat, na které zákazníky se sleva vztahuje, zda jsou to všichni zákazníci, registrovaní zákazníci nebo vybrané role zákazníků. *Obr. 3* níže znázorňuje nastavení katalogové slevy, kde je možné definovat také typ slevy, zda jde o procentuální či hodnotovou. Následně je k dispozici definice pomocí makro formule, kde je možné vybrat, na které zboží se katalogová sleva vztahuje [1].



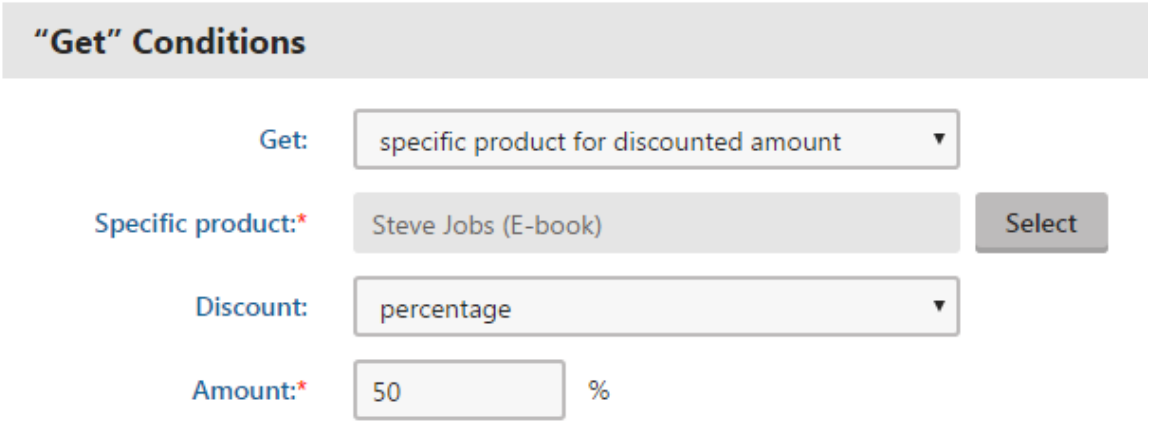
Obr. 3. Nastavení katalogové slevy v systému Kentico.

Dále platforma umožňuje vytvořit slevu typu „kup X a dostaň Y", což je marketingový způsob jak zvýšit prodej vybraných produktů. Slevy tohoto typu mohou být použity ve více scénářích. Ukázka nastavení pro zboží, které je potřeba pro splnění podmínek koupit je zobrazena na obrázku níže (*Obr. 4*). Ukázka nastavení pro zboží, které zákazník získá je následně zobrazena na obrázku pod ním (*Obr. 5*). Scénáře použití typu slevy „kup X a dostaň Y" jsou např. [1]:

- Nakup produkt a získej ten samý produkt zdarma.
- Nakup určitý počet stejného produktu a získej jeden z nich za zlevněnou částku.
- Nakup určitý počet produktů z kategorie a získej nejlevnější produkt zdarma.
- Nakup jeden nebo více produktů a získej vybraný produkt zdarma.



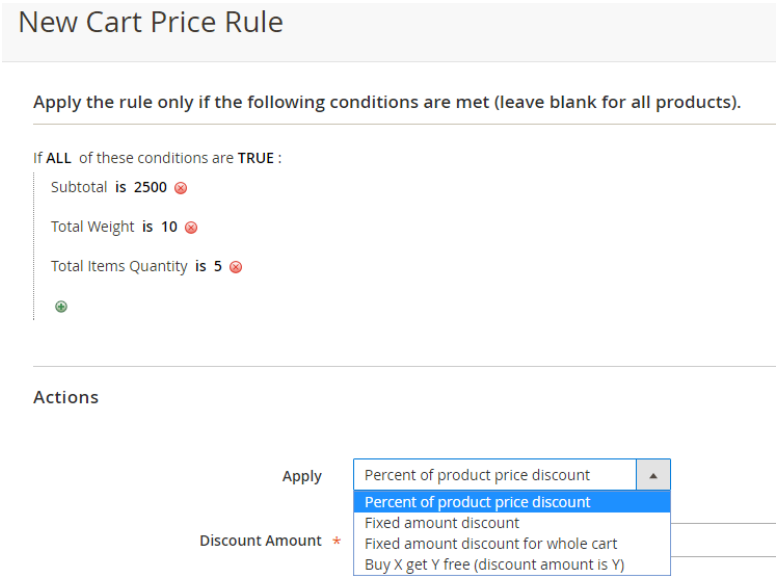
Obr. 4. Ukázka nastavení skupin zboží v nastavení slevy e-shopu Kentico.



Obr. 5. Ukázka nastavení zboží, na které bude uplatněna sleva v e-shopu Kentico.

1.3.2 Magento

Magento nabízí slevy pomocí použití tzv. pravidel, což není nic jiného než série logických rozhodnutí založených na specifikaci, kterou zadá uživatel. Jinými slovy lze Magentu říci, jakou slevu je zapotřebí nastavit v závislosti na příslušné situaci. Magentu lze také říci kolik zlevnit procent nebo zlevnit fixní částkou z celkové hodnoty zboží v košíku nebo celkové hodnoty zboží v košíku plus poštovné. Na *Obr. 6* níže lze vidět nabídku modulu marketingu v administraci internetového obchodu Magento a konkrétně jeho podsekci pro vytvoření nového cenového pravidla pro košík [2].



Obr. 6. Vytvoření cenového pravidla v e-shopu Magento.

2 WEBOVÉ TECHNOLOGIE

Pro vývoj slevového systému a jeho komponent byly využity technologie, při jejichž výběru byla věnována pozornost zejména standardizaci a vzájemné kompatibilitě. Uváděné ukázky kódu slouží zejména k pochopení fundamentálních principů pro následné využití v praktické části.

2.1 WWW

Na začátku devadesátých let hrstka talentovaných inženýrů v CERN laboratoři, vedená Timem Bernersem-Lee pracovala na konceptu, který se nazýval World Wide Web, jehož cílem bylo poskytnout výměnu dokumentů pomocí počítačové sítě. World Wide Web, který je také nazývaný pomocí zkratky WWW nebo W3, byl původně zaměřen na vědecké organizace po celém světě, tak aby měl každý autor vědeckého materiálu možnost jeho publikace a výměny mezi organizacemi. Koncem devadesátých let již Tim Berners-Lee pracoval na prototypu webového serveru a prohlížeče, který byl schopen vykreslovat dokumenty a pracovat s odkazy. V roce 1993 se CERN rozhodl, že WWW již nebude pouze chráněný, ale bude zveřejněn každému pro použití. Toto vedlo k vytvoření mezinárodního konsorcia známého jako World Wide Web konsorcium nebo také organizace W3C. W3C začalo formalizovat práci, která byla odvedena v CERNu. Výsledkem formalizace byly koncepty protokolů, které jsou nyní známé jako HTML, HTTP a URI. Tyto tři protokoly jsou základními stavebními kameny pro vznik revolučního fenoménu, který nyní známe pod pojmem Internet [3].

2.2 HTML

Pokud začnete procházet webové stránky, pak se zcela jistě setkáte s velkým množstvím obrázků, grafického obsahu, videa, textu a také audia. Každá webová stránka se zdá být unikátní, ale na pozadí toho všeho existuje jedna společná technologie a tou je HTML neboli Hypertext Markup Language. HTML obsahuje kromě stavebních bloků webových stránek také speciální instrukční sadu, která umožňuje pomocí klíčových slov textu odkazovat na informace v Internetu. Odkazy jsou také nazývány hyperlinky a jedná se o jeden ze základních stavebních kamenů World Wide Webu [3].

Je příhodné také zmínit, že HTML dokáže fungovat nezávisle na platformě operačního systému pracovní stanice, tedy je možné jej používat na operačních systémech Mac, Linux, Windows a dalších [3].

2.2.1 Struktura dokumentu

V době psaní této práce tj. květnu 2017 je nejnovější verzí HTML verze 5, jejíž struktura bude popsána. HTML dokument verze 5 začíná vždy *<!DOCTYPE HTML>*, což říká internetovému prohlížeči, že se jedná o dokument verze HTML 5 [3].

- Kořenový element se značí jako *<html></html>* a reprezentuje celý obsah dokumentu.
- Jako *<head></head>* značíme hlavičku dokumentu, která obsahuje metadata. Metadata mají vztah k celému dokumentu a může být pomocí nich definováno kódování, název autora, název dokumentu, popis, klíčová slova, titulek dokumentu nebo také kaskádové styly.
- Posledním základním prvkem je *<body></body>*, který zahrnuje obsah dokumentu.

2.3 CSS – kaskádové styly

Jedná se o jednoduchý grafický jazyk, který se používá k tvorbě webových stránek. Jazyk se používá konkrétně k definici stylů webových stránek, včetně jejich uspořádání, vzhledu a způsobu zobrazení na různorodých zařízeních a velikostech obrazovek. Definice stylu jsou typicky uloženy v externích souborech označených .css [4].

2.3.1 Syntaxe

Pokud se zaměříme na syntaxi, tak zjistíme, že je zcela jednoduchá a setkáme se v ní pouze se seznamem pravidel.

2.3.2 Deklarace

Deklarace lze chápat jako dvojici vlastnosti a hodnoty, které provádí definici vizuálních stylů. Funkci vlastností si lze představit tak, že nám určují šířku, barvu pozadí nebo také typografii písma. Hodnoty definují již konkrétní šířku např. 500px, barvu, což může být např. černá, nebo také typ písma Arial. Specifikovanou deklaraci lze provést následujícím zápisem CSS kódu:

width: 500px; background-color: black; font-family: Arial;

2.3.3 Selektory

Selektory specifikují výběr části HTML dokumentu, které budou ovlivněny specifikovanými deklaracemi. Selektory lze v rámci CSS rozdělit na více typů tj. selektory elementů, selektory třídy, selektory identifikátoru, selektory následníka a selektory potomka, jejich kombinace a také zřetězení. Příkladem jednoho z typu tj. selektoru třídy je níže uvedený zápis CSS kódu [4]:

.form-control {display: block; width: 100%; height: 34px; padding: 6px 12px; font-size: 14px; line-height: 1.42857143; color: #555; background-color: #fff; background-image: none;border: 1px solid #ccc; border-radius: 4px;}

2.3.4 Způsoby implementace CSS

První metodou implementace může být použití style atributů. Každá deklarace specifikuje jméno vlastnosti, kterou chceme změnit. Lze použít více deklarací najednou a oddělit je mezi sebou středníkem. Implementovat můžeme jako níže uvedený CSS kód [5]:

* *

Atribut stylu je specifikován pouze pro *div* element a bude mít účinek pouze na tento element. Ostatní elementy zůstanou beze změny, i když jsou to také *div* elementy. Použití style atributu je jednoduché, nicméně aplikace probíhá pouze na úrovni jednoho elementu. Můžete použít style atribut pro každý element, který chcete změnit, ale pak se kód stane velmi složitým a jakékoliv budoucí revize kódu jsou zbytečně složité [5].

Existuje druhá metoda, která je v tomto směru flexibilnější, říká se jí vnořený styl. Stále používáme deklarace, i když jde o vnořené styly, ale jsou ohraničeny složenými závorkami *{}*. Vnořený styl můžeme implementovat např. následovně [5]:

<style> h1 {font-size: 48px; font-weight: normal; margin: 0; padding: 0 30px; line-height: 150px;}</style>

Je dobré dodržovat konvenci oddělení vnořených stylu, které ovlivňují vzhled od obsahu, kód je pak daleko přehlednější. Jinými slovy kód vnořeného stylu umístit do hlavičky HTML dokumentu tj. mezi tagy *<head></head>* a nikoliv do těla HTML dokumentu tj. mezi tagy *<body></body>* [5].

Třetí způsob implementace je externí soubor s CSS styly, kde není potřeba definovat *<style>* element. Definují se pouze napřímo selektory a deklarace. Následně lze použít *link* element pro přenos stylů do HTML dokumentu definicí následujícího příkazu [5]:

<link href="~/Content/checkboxes.css" rel="stylesheet" />

Lze linkovat více externích souborů s CSS styly. Pořadí v jakém se externí soubory linkují, je důležité, protože pokud se definují dva styly pro stejný selektor, pak je aplikován ten, který je načten jako poslední [5].

Klíčovou záležitostí pro pochopení stylů je to, jak se vrství a jak dědí. V HTML dokumentu může být několik zdrojů pro CSS vlastnosti, pak vrstvení a dědění v praxi znamená to, které z nich prohlížeč vyhodnotí pro zobrazení elementu. Dalšími styly, které existují, jsou styly prohlížeče a uživatelské styly. Styly prohlížeče jsou konvence stylů, které prohlížeč aplikuje na element, pokud na něj nebyl specifikován žádný jiný styl. Dále většina prohlížečů povoluje uživatelům definovat jejich vlastní uživatelské styly. Pořadí jakým prohlížeč vyhodnocuje zobrazení elementu je pak následující.

- Inline styly (styly, které se definují pomocí atributu)
- Vnořené styly (styly, které jsou uzavřeny v *style* elementu)
- Externí styly (styly, u kterých je důležité použití *link* elementu)
- Uživatelské styly (styly, které jsou definovány samotným uživatelem)
- Styly prohlížeče (konvence stylů aplikované samotným prohlížečem)

Pořadí může být změněno tím, že se použije u vlastnosti značka *!important*. Uvedenou značkou dojde k tomu, že prohlížeč vyhodnotí větší prioritu pro danou vlastnost [5].

2.4 JavaScript

Javascript je skriptovací jazyk vytvořený proto, aby bylo možné interagovat s webovými stránkami. Javascript tvořen následujícími třemi částmi.

- ECMASkript, který je definován ve standardu ECMA-262 a poskytuje funkcionalitu jádra.
- Dokument objekt model (DOM), který poskytuje metody a rozhraní pro práci s obsahem webové stránky.
- Browser objekt model (BOM), který poskytuje metody a rozhraní pro interakci s webovým prohlížečem.

Existují různé úrovně podpory pro uvedené tři části JavaScriptu skrze pěti hlavními webovými prohlížeči (Internet Explorer, Firefox, Chrome, Safari a Opera). Podpora pro ECMAScript 3 je dobrá skrze všemi prohlížeči a podpora pro ECMAScript 5 roste, zatímco podpora pro DOM se výrazně liší. Podpora BOM, která byla v poslední době zahrnuta do

HTML5 se může lišit prohlížeč od prohlížeče, ačkoli existují společné znaky, které jsou považovány za dostupné [6].

2.4.1 Syntaxe

Syntaxe ECMA skriptu dědí z jazyka C a dalších podobných jazyků jako je Java nebo Perl, tedy vývojář, který zná minimálně jeden z těchto jazyků, by neměl mít s JavaScriptem problém. Důležité je si uvědomit, že veškeré názvy proměnných, názvy funkcí a operátorů jsou case-sensitive, což znamená, že proměnná *variable* je rozdílná od proměnné *Variable* [6].

2.4.2 Identifikátory

Identifikátory mohou nabývat jednoho nebo více znaků ve formátu, kdy první znak musí být písmeno, podtržítko nebo značka dollaru. Všechny ostatní znaky mohou být písmena, podtržítka, značky dollaru nebo čísla. ECMAScript používá camel case konvenci tzn., že první písmeno prvního slova je malé a všechna další počáteční písmena ostatních slov jsou velká. Ačkoli tato konvence není povinností, tak se bere jako best-practise. Příkladem může být název funkce *showSecretKey()* [6].

2.4.3 Způsoby implementace JavaScriptu

Primární metodou je vkládání JavaScriptu do HTML stránky použitím <*script*> elementu. Jako zajímavost lze uvést, že tento element byl vytvořen společností Netscape a jeho první implementace proběhla v rámci Netscape Navigatoru 2. Později bylo přidáno do formální specifikace HTML [6].

Existují dva způsoby použití <*script*> elementu, a to jako vnořený JavaScript kód přímo do webové stránky, což je výhodné použít zejména při ladění funkcionality. V produkčním řešení se nedoporučuje tento způsob používat pro složité skripty vnořené ve webové stránce, a to kvůli zpomalení načítání stránky, protože veškerý takto umístěný skript musí být vyhodnocen, aby mohlo dojít ke kompletnímu načtení stránky. Druhým způsobem je uložení do externího souboru, který je výhodný hlavně z hlediska rychlosti, protože obecně webové prohlížeče cachují veškeré externí soubory linkované do HTML dokumentu [6].

Použití vnořeného JavaScriptu lze vysvětlit na níže uvedeném příkladu. Obsah JavaScriptové funkce je umístěn ve stejném souboru jako HTML kód [6]. Níže uvedený HTML kód zajišťuje zavolání JavaScriptové funkce *showSecretKey()* po události *onClick*:

```
<div>
<label id="secretKeyLabel">Secret Key <span title="Po kliknutí na tuto ikonu se zobrazí tajný klíč pro autorizovanou komunikaci s REST API" onclick="showSecret-Key(); return false" class="fa fa-question-circle-o"></span></label>
<input type="text" value="**********" class="form-control" id="editSecretKeyFake" disabled="disabled" />
@Html.TextBoxFor(x => x.SecretKey, new {
@class = "form-control hidden", id = "editSecretKey", disabled = "disabled";
})
</div>
<script>
function showSecretKey() {$("#editSecretKeyFake").hide();
    $("#editSecretKey").removeClass("hidden");
}
</script>
```

JavaScript je vyhodnocen již při prvotním načtení stránky a pak již dochází pouze k odchytávání události, která zajistí zobrazení skrytého tajného klíče [6].

Použití druhého způsobu, a to odkazu z externího souboru lze provést níže uvedeným způsobem [6]:

```
<script src="~/Scripts/modernizr-2.6.2.js"></script>
```

Obecně se za tímto příkazem neskrývá nic jiného než vložení odkazu na soubor, který se následně objeví v záhlaví HTML dokumentu [6].

2.4.4 jQuery

Při sílení zájmu veřejnosti o dynamické webové stránky se rozmáhají rovněž frameworky pro jazyk JavaScript. Aby knihovna jQuery mohla uspokojit všechny požadavky dnešního náročného uživatele, tak používá strategie, které využívají znalosti CSS. Své principy staví na selektorech jazyka CSS, čímž získává jednoduchou, avšak užitečnou možnost vyjádřit strukturu dokumentu. Výbornou záležitostí je také rozšiřitelnost knihovny, kdy se využívá principu zásuvných modulů, které je možné přidávat či odstraňovat a dostat se tak pouze k užitečnému minimu. Stojí také za to zmínit, že knihovna agreguje ošetření odchylek pro prohlížeče v rámci běžných úkonů, čímž se zmenšuje velikost zdrojového kódu a také se celý kód zjednodušuje. Knihovna jQuery pracuje se skupinami, tedy v případě, že chceme vyhledat všechny elementy *x* a provést nad nimi nějakou akci, pak není potřeba procházet

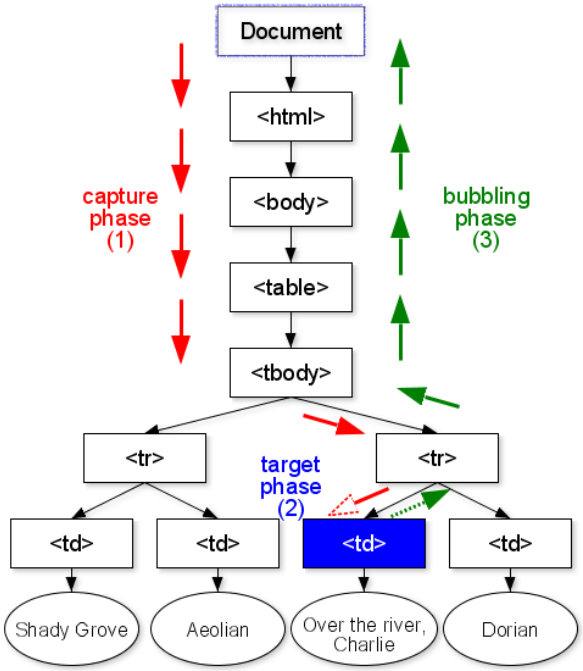
všechny elementy v cyklu, což se nazývá implicitní iterace. Obecně lze také použít provádění více akcí na jednom řádku, protože knihovna jQuery aplikuje na většinu svých metod techniku řetězení, která funguje tak, že volaná metoda objektu vrací samotný objekt, takže je s ním možné ihned pracovat a provést následující akci. Široké spektrum funkcí a komfortní implementace knihovny jQuery jsou původem z části jejího návrhu a části jejího postupného vývojového procesu, který ji neustále posiluje, hlavně díky rozrůstající komunitě vývojářů, kteří se snaží do jQuery vyvíjet. Navzdory tomu, že je za tímto robustním a flexibilním produktem obrovské množství práce, je konečná verze volně k dispozici [7].

2.4.5 Vyhledávání

Jedním ze základních cílů knihovny jQuery je usnadnit vyhledávání částí HTML dokumentu. To lze provést zavoláním funkce *$()*. Této funkci lze předat obvykle textový řetězec, který může obsahovat libovolný selektor jazyka CSS. Funkce *$()* vrátí novou instanci objektu knihovny jQuery, což je základní stavební kámen, s nímž lze následně pracovat. Vrácený objekt může obalovat více nebo žádný element modelu DOM a umožňuje k těmto elementům přistupovat nejrůznějšími způsoby. Lze použít tři typy selektorů, a to název elementu, identifikátor nebo třídu. Příkladem může být následující přístup k tlačítku pomocí identifikátoru *$('#btnCreateEshop')* [7].

2.4.6 Události

Existuje spoustu událostí, v nichž bychom mohli chtít provést nějakou úlohu. Jazyk JavaScript umožňuje kromě události načtení stránky pomocí atributu *onload* elementu *body* nebo vlastnosti *window.onload*, ale také kupříkladu události klepnutí myší *onclick*, úpravy formulářových polí *onchange* a změnu velikosti okna *onresize*. Pokud dojde k události, pak celý strom elementů modelu DOM dostane šanci, aby ji obsloužil. Pro každou událost existuje několik elementů, které na ni dokáží reagovat. Existují dvě strategie událostí, a to zachytávání a probublávání událostí. Schéma strategií znázorňuje obrázek uvedený níže (*Obr. 7*) [7].



Obr. 7. Schéma strategií událost [8].

Zachytávání událostí

První ze strategií, která umožňuje elementům reagovat na akce provedené uživatelem je zachytávání událostí. Pokud dojde k zachytávání událostí, pak získává událost nejdříve vnější element a pak specifické elementy [7].

Probublávání událostí

Druhou ze strategií je probublávání událostí. V této strategii dochází k tomu, že prohlížeč odešle událost specifičtějšímu elementu a až následně, co tento element získá příležitost na ni reagovat, předává ji nahoru (dochází k probublávání) k obecnějším elementům [7].

2.5 HTTP

HTTP je obecným aplikačním protokolem, pomocí něhož klient komunikuje se serverem systému WWW. Použití není vázáno jen na přenos dokumentů, ale lze jej použít také pro přenos libovolných binárních dat a pro obecnou komunikaci mezi klienty a aplikačními branami tj. proxy servery [3].

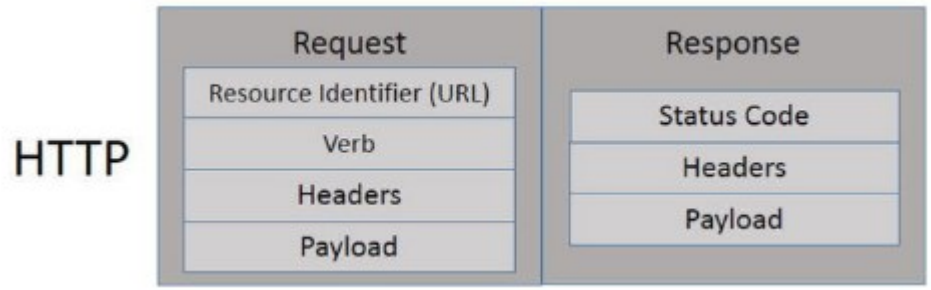
HTTP pracuje s HTTP dotazy a HTTP odpověďmi. Dotazy obsahují na prvním řádku klíčové slovo protokolu HTTP, URL a verze HTTP, vše oddělené mezerami. Někdy může být uváděna také verze HTTP Hlavička HTTP je založena na dvojici hodnot jméno/hodnota,

které jsou odděleny znakem : (dvojtečkou). Za hlavičkovými hodnotami je jeden prázdný řádek a pak začíná tělo, pokud nějaké je, což je znázorněno níže uvedenou ukázkou [3]:

POST /359fa719f8bacd2c533211d30eac9d12 HTTP/1.1
HOST: localhost:18874
content-type: application/json
content-length: 856

Zatímco HTTP odpověď začíná na prvním řádku se stavovým kódem a popisem, dále následuje HTTP hlavička opět s dvojicemi hodnot jméno/hodnota. Dále je zde prázdný řádek, který je následován tělem, pokud nějaké je. Strukturu HTTP dotazu a odpovědi znázorňuje uvedený obrázek (*Obr. 8*). Praktická forma odpovědi vypadá následovně:

200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/7.5
X-Aspnet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Sun, 19 Mar 2017 15:21:19 GMT
Content-Length: 357



Obr. 8. Struktura dotazu a odpovědi HTTP [3].

2.5.1 Metody HTTP/1.1

Protokol HTTP verze 1.1 definuje hned několik metod, které lze použít při komunikaci mezi klientem a serverem. Popsány budou pouze relevantní metody, resp. ty, které byly analyzovány v rámci práce.

GET

Metoda GET umožňuje vrátit jakoukoliv informaci, kterou lze identifikovat pomocí dotazovaného URI. Pokud dotazovaný URI odkazuje na data, pak jsou data vrácena v odpovědi [3].

POST

Metoda POST se používá při dotazu na server, kdy ve chvíli volání není známý přesný identifikátor zdroje, protože ve chvíli volání ještě neexistuje. Požadavek se od požadavku GET liší, protože nese náklad. Jako náklad (payload) lze chápat data, která chceme poslat na server a nechat je zpracovat [3].

2.6 JSON

JSON, nebo také JavaScript Object Notation je textově založený formát pro přenos strukturovaných dat. Byl navržen Douglasem Crockfordem a specifikován v doporučeních RFC 4627 [9].

Použitím jednoduché syntaxe a prostého textu lze jednoduše uložit čísla, textové řetězce, pole a objekty. Lze také tvořit vnořená pole a objekty, což umožňuje vytvoření komplexních datových struktur, které mohou být jednoduše posílány mezi dvěma webovými aplikacemi [9].

Datový formát JSON vznikl v době, kdy v oblasti internetu figuroval převážně formát XML, který trpěl nedostatky, jako jsou složitost nebo nutná detekce uzlů obsahujících pouze bílé znaky. Výhody JSONu je velmi jednoduchá interpretace, díky které jej mohou zapisovat a číst stroje i lidé. Datové struktury formátu JSON, lze také jednoduše mapovat na datové formáty většiny programovacích jazyků. Téměř všechny moderní programovací jazyky obsahují knihovny nebo funkce, které jsou schopny číst nebo zapisovat JSON struktury [9].

Během posledních několika let se tento formát zařadil mezi nejdůležitější formáty v oblasti webových technologií [9].

2.6.1 Struktury

JSON je založen na dvou základních strukturách. Jsou to kolekce párů název/hodnota a seřazený seznam hodnot [10].

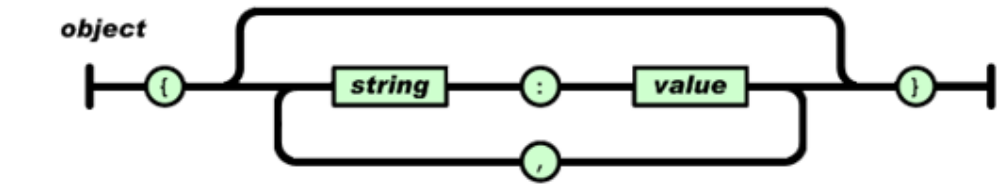
- Kolekce párů název/hodnota bývá v programovacích jazycích realizována různě, např. objektem, strukturou, slovníkem, hash tabulkou, asociativním polem nebo klíčovým seznamem.
- Seřazený seznam hodnot má většina programovacích jazyků realizován pomocí pole, seznamu, posloupnosti nebo vektoru.

2.6.2 Konstrukce

Struktury jsou v JSONu realizovány s využitím 5 konstrukcí. Jedná se o objekt, pole, hodnotu, řetězec a číslo [10].

Objekt

Objekt je neuspořádanou množinou párů název/hodnota. Základní poznávací značkou pro objekt je uvození znakem *{* (levé složené závorky) a ukončení znakem *}* (pravé složené závorky). Každý název je následován znakem *:* (dvojtečky) a páry název/hodnota jsou vždy odděleny znakem *,* (čárky). Schematicky včetně ukázky kódu je znázorněno na níže uvedeném obrázku (*Obr. 9*) [10].

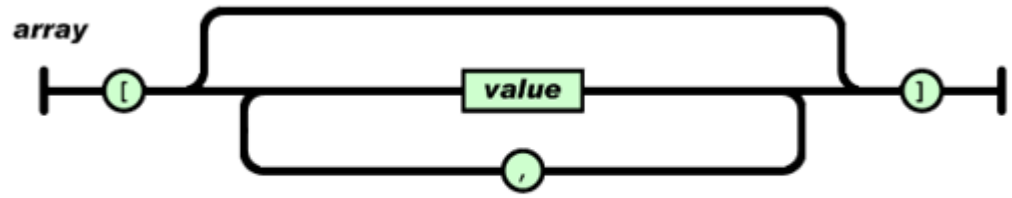



```
"customer": {
        "is_signed": true
}
```

Obr. 9. Schéma JSON objektu [10].

Pole

Pole je seřazená kolekce hodnot. Základní poznávací značkou pro pole je uvození znakem *[* (levé hranaté závorky) a ukončení znakem *]* (pravé hranaté závorky). Hodnoty jsou mezi sebou odděleny znakem *,* (čárky). Níže uvedené schéma a kód (*Obr. 10*) znázorňuje JSON pole [10].

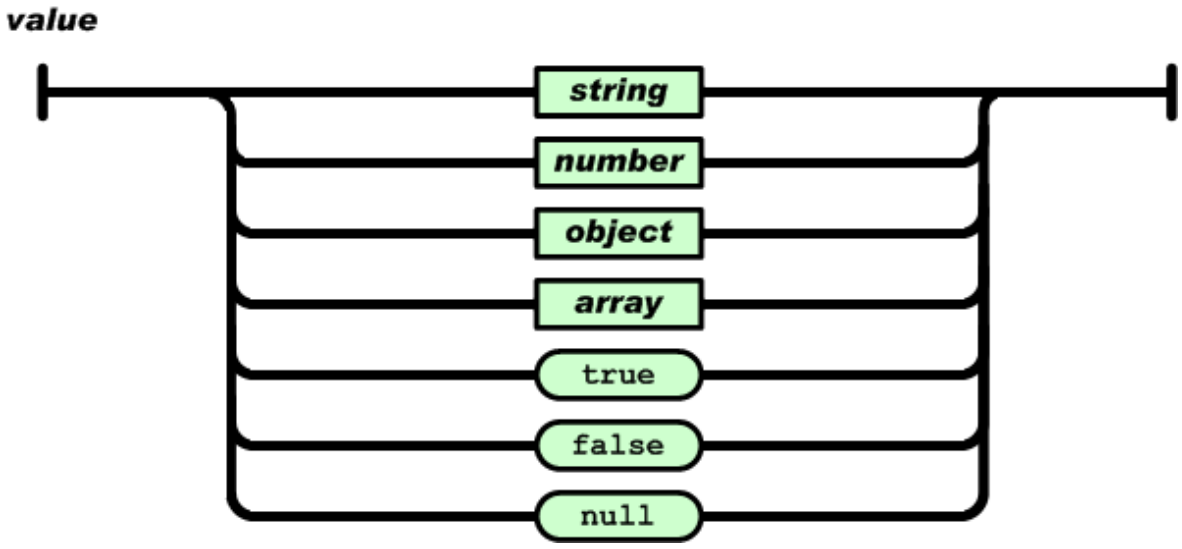



```
"general_lists": [{
                    "general_list_code": "category",
                    "general_list_item_code": "bundy"
                }, {
                    "general_list_code": "brand",
                    "general_list_item_code": "nike"
                }]
```

Obr. 10. Schéma JSON pole [10].

Hodnota

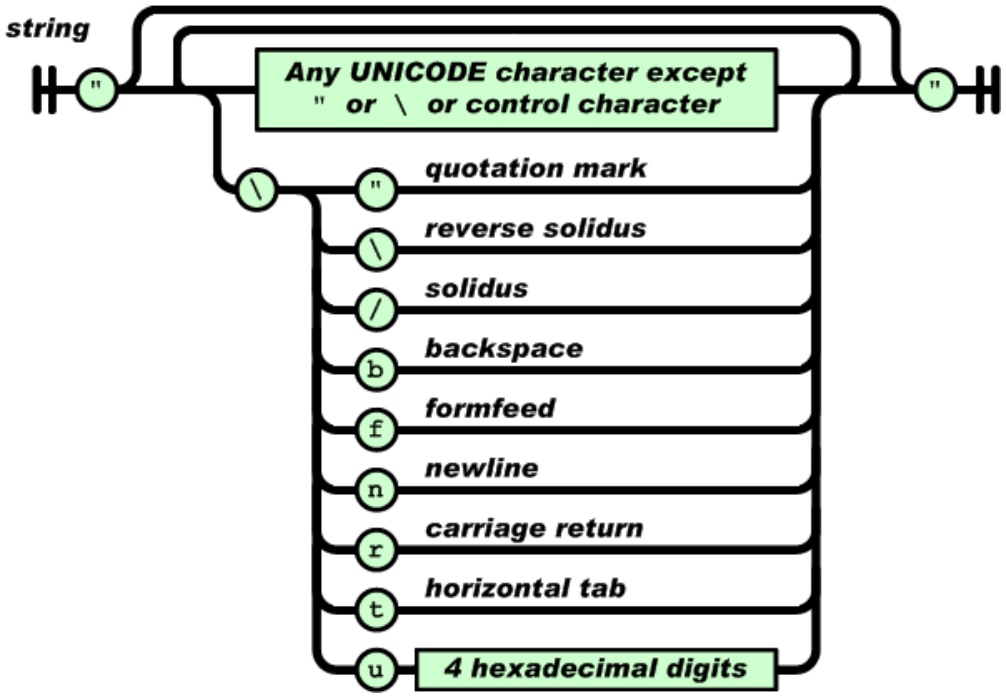
Hodnota je řetězec, který je uzavřen do dvojitých uvozovek. Hodnota může nabývat čísla, true, false, null, objektu nebo pole. Schematicky je JSON hodnota znázorněna na níže uvedeném obrázku (*Obr. 11*) [10].



Obr. 11 - Schéma JSON hodnoty [10].

Řetězec

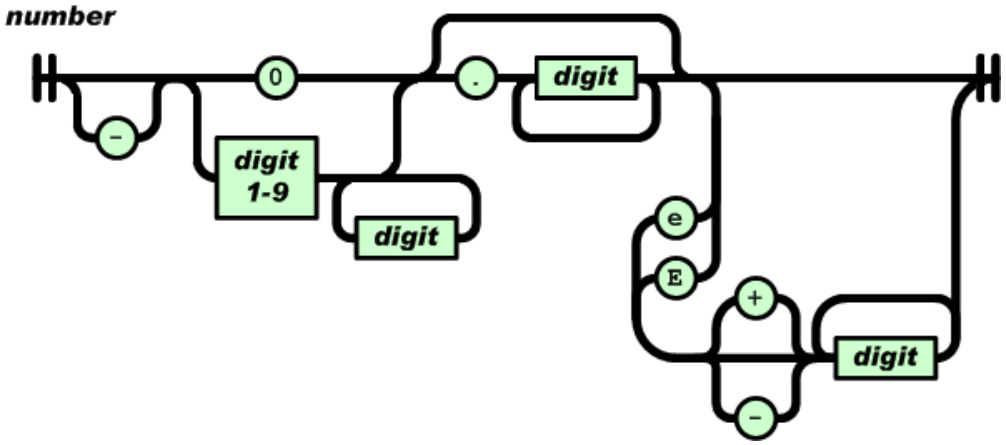
Řetězec je nula nebo více znaků kódování Unicode, uzavřených do dvojitých uvozovek a využívající únikových sekvencí (escape sequence) s použitím zpětného lomítka. Znak je reprezentován jako řetězec s jediným znakem. Schematicky je JSON řetězec znázorněn na níže uvedeném obrázku (*Obr. 12*) [10].



Obr. 12. Schéma JSON řetězce [10].

Číslo

Číslo je velmi podobné číslům z jazyků C a Java. Výjimkou je, že není používán oktalový ani hexadecimální zápis. Schematicky je JSON číslo znázorněno na níže uvedeném obrázku (*Obr. 13*) [10].



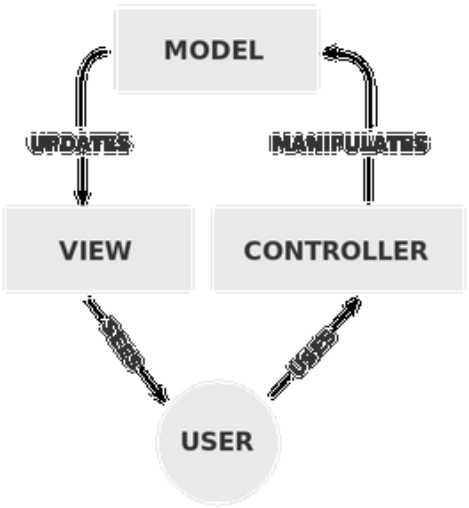
Obr. 13. Schéma JSON čísla [10].

3 NÁVRHOVÝ VZOR MVC

MVC znamená zkratku Model-View-Controller. Jedná se se o návrhový vzor, který byl publikován v sedmdesátých letech minulého století a dnes již existuje mnoho jeho modifikací. MVC se také jinak nazývá tří vrstvým návrhovým vzorem, nebo také trojúhelníkovým návrhovým vzorem. Zjednodušeně to znamená, že datový model tj. model a business logika tj. controller jsou odděleny od uživatelského rozhraní tj. view. Klasické MVC existovalo již v době, kdy každý ovládací prvek, který na obrazovce existoval, byl považován za hloupý a každý ovládací prvek byl spárován se svým vlastním controllerem pro řízení uživatelských interakcí, které na ovládacích prvcích probíhají, tedy pokud existuje deset view, pak musí existovat deset controllerů. Příchod Windows GUI tento princip zásadně změnil. Vztah jeden ovládací ovládací prvek a jeden controller se stal zastaralým. Ovládací prvky získali inteligenci k reakci na akce iniciované uživatelem. Ve světe Windows je view plochou, na které existují všechny ovládací prvky a pro které existuje pouze jeden controller. View může přijímat události a pomáhá tím controllerům provádět další zpracování [11].

3.1.1 Struktura návrhového vzoru

MVC návrhový vzor dělí aplikaci do 3 hlavních aspektů, a to Model, View a Controller viz uvedený obrázek (*Obr. 14*).



Obr. 14. MVC struktura [12].

Model

Model přijímá vstupy od controlleru a rozhoduje, co jsou potřeba za data, tak aby došlo ke splnění požadavku, který vznesl controller. Model je také schopen vytvářet transakce s databázemi, nebo jiným persistentním uložištěm, tak jak je zrovna potřeba. Pokud model obdrží požadovanou informaci, pak informuje view pomocí controlleru o nových datech vyvoláním příslušných událostí [11].

View

View zahrnuje všechny ovládací prvky uživatelského rozhraní, které jsou vyžadovány konečným uživatelem, aby mohlo dojít k interakci s aplikací. Jeho práce je monitorovat gesta koncového uživatele a posílat je do controlleru pro další zpracování. View zpracovává také avíza o události, které obdrží z modelu a může také požadovat nějaká nová data, tak aby mohlo dojít k jejich následnému vykreslení na obrazovce. Avšak v některých případech může být controller navrhnut tak, aby získával data z modelu, formátoval je a poslal je do view. Pak je jedinou úlohou view vykreslit data získaná z view bez nějakého dalšího zpracování [11].

Controller

Controller si lze představit jako rozšíření nebo osobního asistenta view, kdy jsou všechny události původem z view jsou zpracovány controllerem. Controller také informuje model v zastoupení view o tom, že se stala nějaká událost ve view a mohou být vyžadována nějaká nová data [11].

4 FRAMEWORKY

Framework je z obecného hlediska koncepční struktura, jejíž záměrem je sloužit jako podpora nebo návod pro vytvoření něčeho užitečného.

V počítačových systémech je framework často vrstvená struktura udávající informaci, jaký typ softwaru lze pomocí ní tvořit a jakým způsobem bude tento software vzájemně interagovat. Framework může být také sada funkcí uvnitř systému a popis toho jak tyto funkce vzájemně interagují, typicky vrstvy operačního systému, vrstvy aplikačního subsystému, nebo popis toho, jak by měla být standardizovaná komunikace na určité úrovni počítačové sítě a podobně [13].

Z programového hlediska je framework balík softwarových knihoven, kompilátorů, interpreterů, nebo API [13].

4.1 ASP.NET

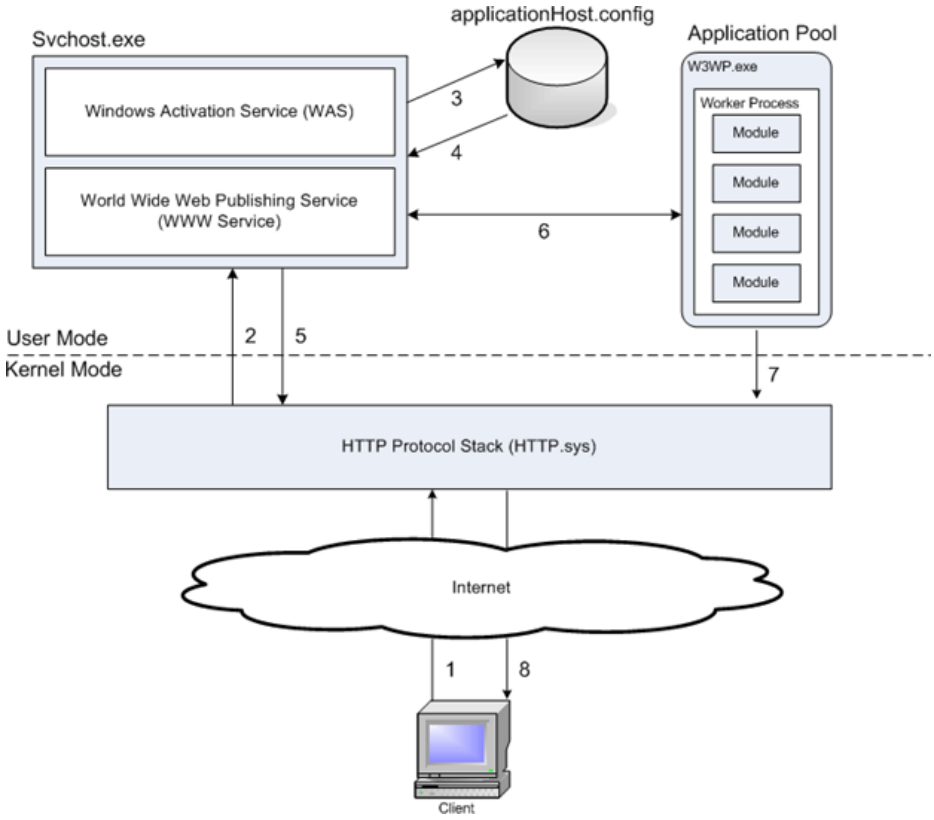
Je pododdílem .NET Frameworku vyhrazeným pro tvorbu webových aplikací. Ve své podstatě jde o sadu knihoven, které umožňují tvorbu webových aplikací v programovacím jazyce C#. Výhodou je, že knihovny obsahují již hotová řešení mnoha fundamentálních problémů, se kterými se lze v oblasti webových technologií setkat. Zejména je řešena bezpečnost, práce s databází nebo správa formulářů a podobně. Princip technologie je založen na architektuře klient-server, kdy ASP.NET běží vždy na straně serveru. Aplikaci v ASP.NET lze zjednodušeně chápat jako program jehož výstupem je HTML stránka [14].

První vydání technologie ASP.NET 1.0 předčilo všechna očekávání, dokonce ani lidé ze společnosti Microsoft netušili, jak nadšeným způsobem bude tato technologie přijata. ASP.NET se stalo velmi rychle zaběhnutým standardem a silným konkurentem v oblasti webových vývojových platforem [14].

4.1.1 ASP.NET MVC

Společnost Microsoft uvedla MVC jako jednu z alternativ k webovým formulářům. MVC je založeno na ASP.NET, jedná se se o framework, který umožňuje na základě znalostí ASP.NET vybudovat v efektivním čase robustní webové aplikace, které ctí principy MVC návrhového vzoru. V dnešní době se pro realizaci webových aplikací postavených na ASP.NET používá Microsoft MVC návrhový vzor a jeho modifikace jako jedny z nejčastějších [14].

Aby bylo možné pochopit funkci zpracování MVC požadavku, pak je nutné začít od úplného začátku, což je IIS. Níže uvedené schéma (*Obr. 15*) znázorňuje z IIS perspektivy toto zpracování HTTP požadavku, který vstupuje do MVC Frameworku [15].



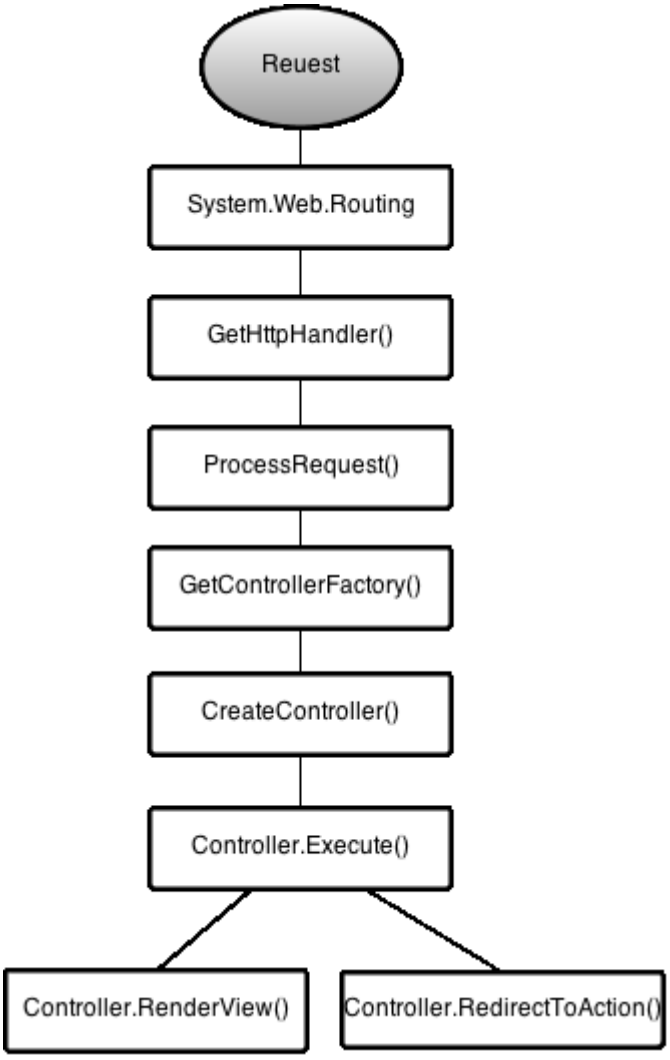
Obr. 15. Zpracování HTTP požadavku z hlediska IIS [15].

Zpracování ASP.NET lze popsat ve 4 jednoduchých krocích:

- Krok 1 (Směrování): Tento první krok se spustí vždy po prvním spuštění aplikace. UrlRoutingModule, pak zachycuje všechny požadavky. Chcete-li nechat MVC směrovací handler zpracovat požadavky, pak je nutná konfigurace směrovací tabulky když aplikace startuje. Defaultní směrovací konfigurace je poskytnuta souborem global.asax.
- Krok 2 (MvcHandler): Ve druhém kroku vytváří MvcHandler controller, předává controlleru ControllerContext a spouští controller.
- Krok 3 (Controller): MVC controller factory vyhledá a vytvoří controller v Create-Controller. Controller určuje, která metoda controlleru má být spuštěna, vytvoří seznam parametrů a spustí metodu. ControllerContext je předán do Execute () metody ve třídě controlleru.

- Krok 4 (Akce): Metody controlleru končí voláním metody RenderView () nebo RedirectToAction (). Metoda RenderView () je zodpovědná za zobrazování zobrazení (stránky) do prohlížeče. Metoda Controller.RenderView () deleguje svou práci na konkrétní ViewEngine.

Zjednodušeně si lze proces zpracování HTTP požadavku v ASP.NET MVC představit také jako níže uvedené schéma (*Obr. 16*). Znázorněné chování je defaultním, ale ASP.NET MVC umožňuje ladění procesu dle potřeb, dokonce lze zlepšit výkon použitím specializovaného controlleru, což může být např. asynchronní controller [15].



Obr. 16. Zjednodušené schéma HTTP zpracování [15].

ASP.NET MVC - Použití a výhody

Nosnou myšlenkou použití ASP.NET MVC je držet dobré oddělení mezi prezentací programu, což je user interface a zbytkem funkcionality. Oddělení umožňuje adresovat každý aspekt vrstvy samostatně. Oddělení view od zbytku aplikační logiky otevírá možnosti v budoucnu měnit logiku view. Toto se v praxi děje velmi často, kdy chce klient po nějaké době změnit design aplikace, protože se mu již jeví zastaralý a je potřeba jít ruku v ruce s aktuálními trendy. Pokud je view úzce spjato s business logikou, pak je velmi těžké tyto změny designu provést a vznikají tím další náklady na opravu bugů a testování.

Zaměříme-li se na další výhody použití ASP.NET MVC, pak lze jednoznačně označit za výhodu možnost snadné údržby prostředí, které zahrnuje různé technologie v různých místech. Díky organizační struktuře MVC lze jednodušeji škálovat robustní aplikace vytvořené v tomto návrhovém vzoru a také je snadněji udržovat vzhledem k čistějšímu oddělení úkolů. MVC dále nepoužívá formuláře založené na serveru, díky tomu je ASP.NET MVC ideální pro vývojáře, kteří chtějí mít plnou kontrolu nad chováním aplikace ASP.NET MVC [15].

Pokud bychom měli ASP.NET MVC porovnat např. vůči klasickým webovým formulářům, které se ve světe Microsoftu také používají, pak najdeme spoustu přednosti hrající ve prospěch ASP.NET MVC. Oddělení modelu, view a controlleru poskytuje mnohem lepší možnosti pro unit testy jakéhokoliv aspektu vytvářené aplikace, protože je možné model, view i controller testovat samostatně. MVC také umožní získat kontrolu nad drobnými detaily aplikace. Orientace v kódu aplikace a jeho správa je také mnohem jednodušší vzhledem k tomu, že existují logické celky kódu a nejedná se o tzv. špagetový kód, kde je práce s daty, zobrazením a aplikační logikou smíchána dohromady [14].

4.1.2 ASP.NET Web API

ASP.NET Web API je webový framework, který byl umístěn do verze Microsoft .NET 4.0 a výše, implementuje RFC 2616 (HTTP 1.1 specifikace), rovněž není konzervativní vůči REST. Lze jej použít k transformaci standardního webového projektu do robustního HTTP API, které umožní využít výhody, jako jsou práce s obsahem, cachování, souběžná validace a podobně. Co může mnohokrát zjednodušit implementaci, je šablona projektu dostupná ve Visual Studiu, podmínkou je však mít stažen minimálně ASP.NET MVC 4, kde je dostupná jak šablona, tak podstatné knihovny pro práci s ASP.NET Web API. Framework může být implementován pomocí tzv. NuGet balíčku, kdy musí být vybrán příslušný balíček a vytvořena reference tohoto balíčku na řešený projekt. Výběr balíčku závisí na problematice, která

je v projektu řešena. Každý z balíčků obsahuje podmnožinu ASP.NET Web API frameworku která je ušitá na míru pro specifický případ užití [3].

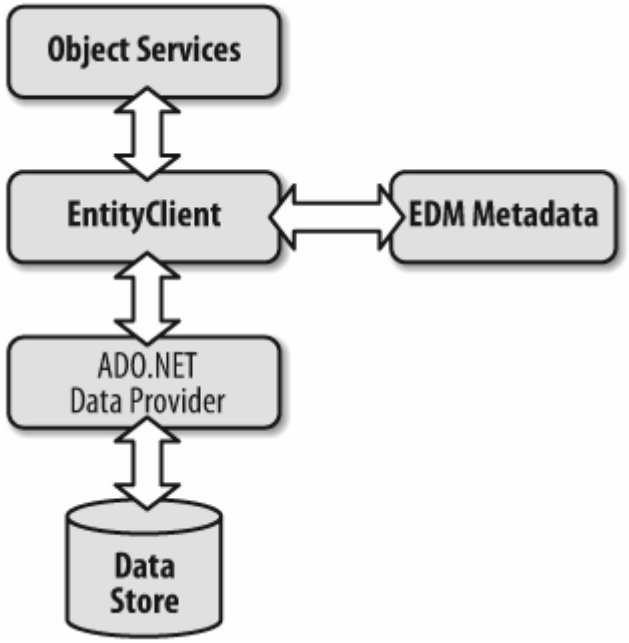
<package id="Microsoft.AspNet.WebApi.Client" version="5.2.3" targetFramework="net452" />

<package id="Microsoft.AspNet.WebApi.Core" version="5.2.3" targetFramework="net452" />

Jeden ze zajímavých aspektů ASP.NET Web API je, že se Microsoft rozhodl mít závislost na *Newtonsoft.Json*, což je open source knihovna. Ačkoliv se podobná záležitost stala v původní verzi MVC s jQuery, tak toto byl historicky první případ, kdy Microsoft začal mít závislost na open source .NET knihovně [3].

4.2 ADO.NET Entity Framework

Vývojář, který vyvíjí novou aplikaci, stráví vždy spoustu času tím, že přemýšlí, jakým způsobem bude řešen back-end tj. databáze, tabulky v databázi, vztahy mezi tabulkami, jak se tabulky budou jmenovat a jaké parametry se budou předávat uloženým procedurám spouštěným přes aplikační logiku. Microsoft Entity Framework kompletně mění celou hru pro všechny .NET vývojáře, takže není potřeba se již zajímat o detaily ukládání dat při psaní aplikační části. Pozornost je věnována spíše psaní aplikační logiky než způsobu přístupu k datům. ADO.NET Entity Framework je novou přístupovou platformou pro tvorbu .NET aplikací. Použitím data readerů a dalších podobných technologií umožňující přístup k databázi, čtení výsledků, sbírání bitů dat, kterými je potřeba nakrmit business třídy, tím vším vývojář ztrácí velké množství času, který by mohl být věnován práci na business logice. Použitím Entity Frameworku nedochází k přímému dotazování vůči databázovému schématu, ale spíše vůči schématu, které je odrazem business modelu. Pokud jsou data vrácena, pak vývojář není tlačen vytvořit z těchto dat objekty, protože řádky a sloupce jsou již jako objekty vráceny. Pokud přijde chvíle, kdy je potřeba uložit změny zpět do databáze, pak dochází pouze k uložení těchto objektů, tedy Entity Framework provede všechnu nezbytnou práci, konkrétně provede transformaci objektů na řádky a sloupce relační databáze. Popsaný princip lze znát pod názvem ORM (Object Relational Mapping). Níže uvedený obrázek (*Obr. 17*) znázorňuje princip funkce Entity Frameworku [16].



Obr. 17. Princip funkce Entity Frameworku [16].

Entity Framework používá Entity Data Model (EDM), který se vyvinul z Entity Relationship Modelingu (ERM), což je koncept, který je známý již po mnoho let. Entity, se kterými se rámci modelu pracuje, nejsou stejné jako objekty, definují schéma objektu, ale ne jeho chování. EDM je datovým modelem na straně klienta a je jádrem Entity Frameworku. Příkladem použití Entity Frameworku je kód znázorněný obrázkem (*Obr. 18*) níže [16].

```
1  //Repository
2  public Country FindCountry(int id) {
3   return _context.Countries.Find(id);
4  }
5  //DAL
6  public DbSet < Country > Countries {
7   get;
8   set;
9  }
10 //Structures
11 public class Country {
12  public int ID {
13   get;
14   set;
15  }
16  [StringLength(2)]
17  public string ISOCode {
18   get;
19   set;
20  }
21  [StringLength(50)]
22  public string CountryName {
23   get;
24   set;
25  }
26  public ApplicationUser User {
27   get;
28   set;
29  }
30 }
```

Obr. 18. Příklad použití Entity Frameworku.

4.3 BOOTSTRAP

Bootstrap je kolekce nástrojů pro tvorbu webových aplikací. Jedná se ve své podstatě o frontendový framework. Obsahuje HTML a CSS designové šablony pro typografii, formuláře, tlačítka, navigační menu a další rozhraní komponent jako menu, hlavičky a obrázkové carousely. Bootstrap začal jako interní projekt Twitteru, ale vývojáři, kteří jej vyvinuli, dostali povolení od managementu, poskytnout jej veřejnosti formou open source řešení. Důležitou věcí v Bootstrapu je to, že je designován jako „mobile first", což znamená, že prvky, které jsou vytvořeny pomocí Bootstrapu s cílem správného zobrazení v běžném prohlížeči, budou také vhodně zobrazeny na mobilním zařízení bez spousty práce navíc. Bootstrap je responzivní, což znamená, že zobrazení stránky automaticky přizpůsobí obrazovce zařízení, na kterém je stránka zobrazována. Existují 3 verze Bootstrapu, které jsou k dispozici pro stažení, a to zkompilovaná a minifikovaná verze, která obsahuje CSS, JavaScript a fonty, ale neobsahuje dokumentaci a zdrojové soubory, dále je to verze se zdrojovými soubory, která obsahuje Less, JavaScript, soubory fontů a dokumentaci a naposled jde o verzi SAAS. Je možné

také použít hostovanou verzi bez nutnosti Bootstrap stáhnout. Bootstrap je modulární a skládá se v podstatě ze série Less souborů, které implementují množství komponent tohoto nástroje [17].

4.4 Less

Less se řadí mezi tzv. preprocesory, což znamená, že transformuje jednu formu dat do druhé formy. Jako vstup se používá jazyk Less, který je dynamickým jazykem pro stylování a dělá údržbu komplexnějších stylopisů jednoduší, výstupem je pak CSS. Less rozšiřuje výchozí CSS syntaxi o mixiny, proměnné a vnořená pravidla. Vzhledem k jednoduchosti Lessu a faktu, že se jedná o open source jeho popularita stále stoupá. Zásadním rozdílem oproti ostatním CSS preprocesorům je to, že Less umožňuje kompilaci v reálném čase pomocí less.js uvnitř prohlížeče. Pro použití Bootstrapu není podmínkou perfektní znalost Lessu, tato znalost je nezbytná pouze v případě, že je potřeba upravovat Bootstrap. Příkladem definice proměnné a její použití v Lessu je níže uvedená ukázka kódu [18]:

```
@bs-datetimepicker-timepicker-font-size: 1.2em;
  .timepicker-hour, .timepicker-minute, .timepicker-second {
      width: 54px;
      font-weight: bold;
      font-size: @bs-datetimepicker-timepicker-font-size;
      margin: 0;
   }
```

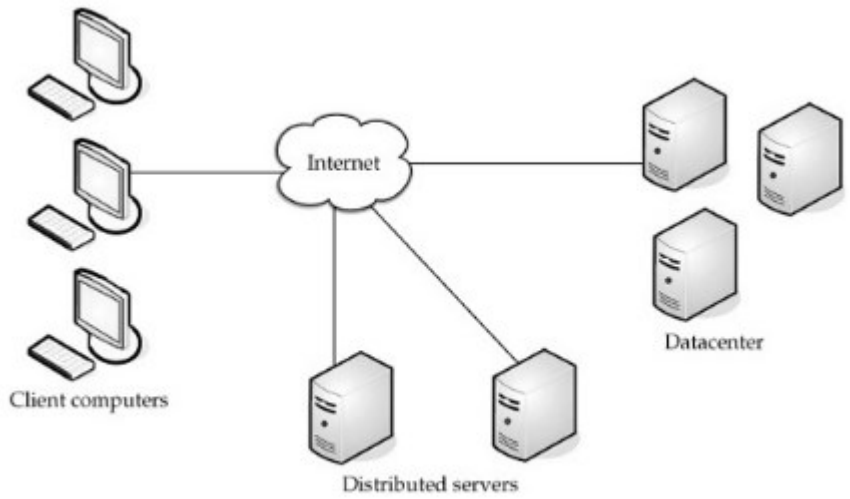
Výše uvedená ukázka kódu nastavuje velikost fontu pomocí proměnné za použití Less preprocesoru.

5 CLOUD COMPUTING

Cloud computing je dnes všude kolem nás. Cloud computing lze v základu chápat jako pojem, který vám umožňuje přistupovat k aplikacím, které jsou ve skutečnosti umístěny jinde než na vašem počítači nebo jiném zařízení připojenému k Internetu, nejčastěji se tedy jedná o vzdálené datacentrum. Je zde spousta výhod, které z tohoto plynou. První z nich je, že vaši aplikaci hostuje jiná společnost, což znamená, že tato společnost řeší náklady na infrastrukturu serverů, update software a v závislosti na tom jakou uzavřete s touto společností smlouvu, tak platíte méně než když je aplikace provozována na on premise řešení. Tím, že jsou vaše aplikace hostovány na vzdálených serverech, tak nemusíte kupovat nové servery při vzrůstajícím počtu uživatelů aplikace, stejně tak nemusíte platit za elektřinu, která servery napájí a zajišťuje jejich chlazení. Vše co má výhody má ovšem také své nevýhody, tedy v případě, že vzdálené servery budou nedostupné, tak se nepřipojíte ke své aplikaci. Další nevýhodou je integrace více aplikací dohromady, tedy za předpokladu, že je jedna aplikace provozována on premise a druhá v cloudu, pak se integrace stává komplikovanější, než kdyby byly obě aplikace provozovány jako on premise řešení [19].

5.1.1 Komponenty cloudu

Zjednodušeně, z topologického úhlu pohledu, je cloud computing řešení složeno z několika prvků, a to klientů, datacentra, a distribuovaných serverů viz *Obr. 19* níže. Každý prvek má smysl a hraje specifickou roli v dodávce funkční cloudově založené aplikace [19].



Obr. 19. Zjednodušená cloud topologie [19].

Klienti

Klienti jsou v cloud computing architektuře přesně to samé jako v typickém LANu. Jsou to typicky pracovní stanice, které jsou umístěny na kancelářském stole, ale mohou to být také laptopy, tablety, mobilní telefony nebo PDA zařízení. Jinými slovy se jedná o zařízení, se kterými interagují koncový uživatelé proto, aby dokázali spravovat jejich informace umístěné v cloudu. Klienti se dělí do třech kategorií, a to mobilní, tencí klienti a tlustí klienti [19].

- Mobilním klientem se rozumí všechna mobilní zařízení jako např. iPhone.
- Tenkého klienta můžeme chápat jako počítač, který nemá svůj vlastní hard disk a nechává veškerou práci na serveru. Zajišťuje tak pouze zobrazení zpracovaných informací.
- Tlustým klientem je běžný počítač, který používá webový prohlížeč k tomu, aby se připojil ke cloudu.

Datacentrum

Datacentrum je kolekcí serverů, kde je hostována aplikace, do které se přihlašujete. V praxi se může jednat o velkou místnost v podzemní části budovy nebo místnost na druhém konci světa, která je plná serverů, ke kterým lze přistupovat pomocí Internetu. Prakticky se využívá také virtualizace, kdy na jednom fyzickém serveru může běžet několik virtuálních serverů. Počet virtuálních serverů na jednom fyzickém serveru se odvíjí od rychlosti a velikosti tohoto fyzického serveru [19].

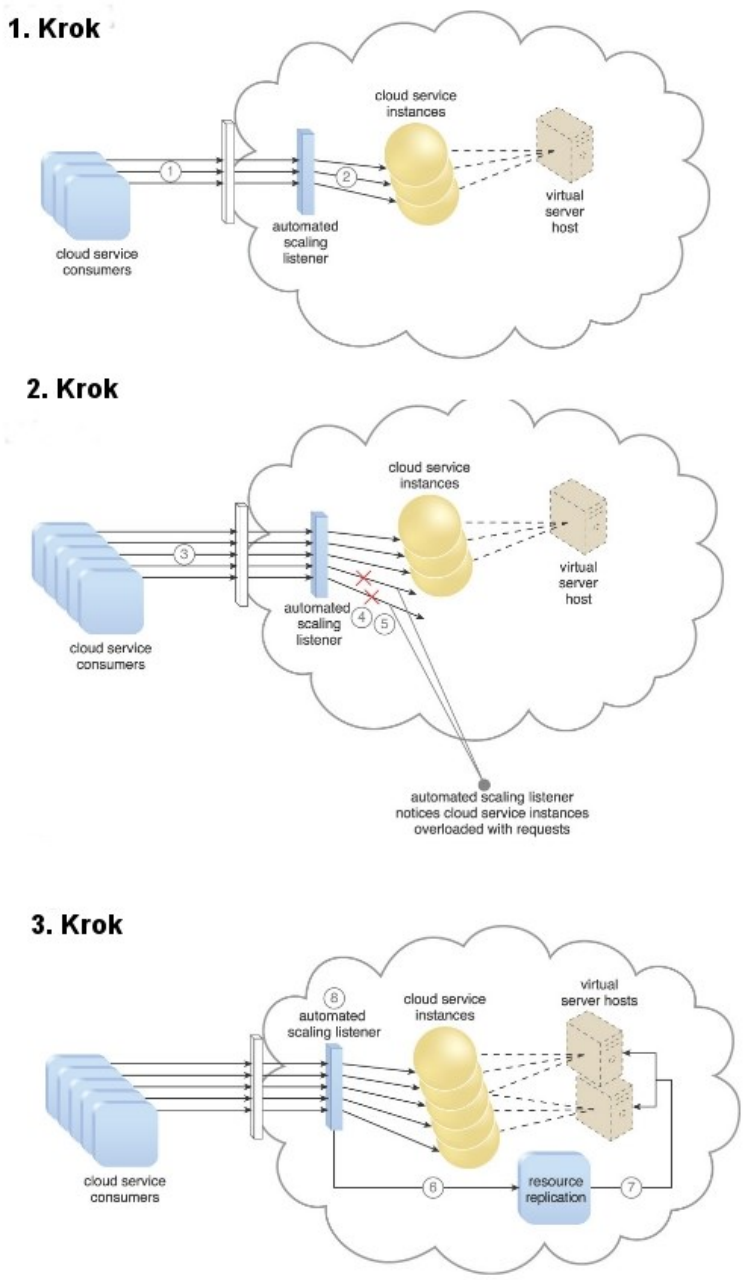
Distribuované servery

Servery nemusí být všechny hostovány na stejném místě. Často jsou servery geograficky rozmístěny do různých lokací. Toto dává poskytovateli cloudové služby pružnější možnosti v oblasti zabezpečení. Konkrétně Amazon má své servery rozmístěny po celém světě. Pokud nastane problém s dostupnosti jedné sítě, tak bude služba stále dostupná z další [19].

5.1.2 Horizontální škálování

Horizontální škálování (scale out) je způsob škálování, který umožňuje škálovat počet IT zdrojových instancí. Často se používá dynamický typ tohoto škálování, aby bylo možné se vypořádat s proměnlivou zátěží. Základem je komponenta, která naslouchá a monitoruje požadavky a pokud je zapotřebí škálovat, pak tato komponenta dává signál replikačnímu me-

chanismu, který spustí duplikaci IT zdroje podle požadavků a oprávnění. Pokud chceme využívat horizontální škálování, pak je zapotřebí tomuto také přizpůsobit kód hostované aplikace. Celý proces dynamického horizontálního škálování znázorňuje níže uvedený obrázek (*Obr. 20*), kdy první krok naznačuje klidový stav, následně ve druhém kroku naslouchací komponenta identifikuje přetížení cloudových služeb velkým množstvím dotazů a vydává signál k replikaci nového zdroje, což je naznačeno v kroku třetím [19].



Obr. 20. Proces dynamického horizontálního škálování [20].

5.1.3 Vertikální škálování

Vertikální škálování (scale up) lze chápat tak, že máme jednu instanci, kterou se snažíme výkonově posílit, což znamená, že namísto 16 GB paměti RAM zvedneme o dalších 16 GB. Stejně tak počet procesorových jader zvedneme např. z 6 na 12, ale stále se bude jednat o

jednu logickou jednotku, na které se škálování provádí. Z pohledu hostovaného software je vertikální škálování výhodnější v tom, že kvůli němu není zapotřebí měnit kód tohoto software. Nevýhodou je fakt, že v rámci vertikálního škálování existují limity, které jsou ovlivněny velikostí serveru. V případě vertikálního škálování se taktéž využívá dynamicky řízeného škálování směrem nahoru nebo dolů, dle počtu požadavků. Pokud je virtuální server zahlcen požadavky, pak může dojít k dynamickému navýšení jeho paměti nebo navýšení počtu procesorových jader [20].

6 SOFTWAROVÉ NÁSTROJE

Výběr softwarových nástrojů je nesmírně komplexní oblastí a jejich volba může výrazně ovlivnit čas, za který je vypracován konkrétní úkol. V této práci jsou použity softwarové nástroje, které za sebou mají letitou historii používání v oblasti vývoje software specializovaného zejména na webová řešení. Některé z uvedených nástrojů mají na platformě Microsoft jakousi exkluzivitu, protože jsou svázány s programovacími jazyky vyvinutými touto společností, tudíž pokud chcete vyvíjet profesionálně na platformě Microsoft, tak se těmto nástrojům nevyhnete. Jedná se především o MSSQL Server a Microsoft Visual Studio.

6.1 XAMPP

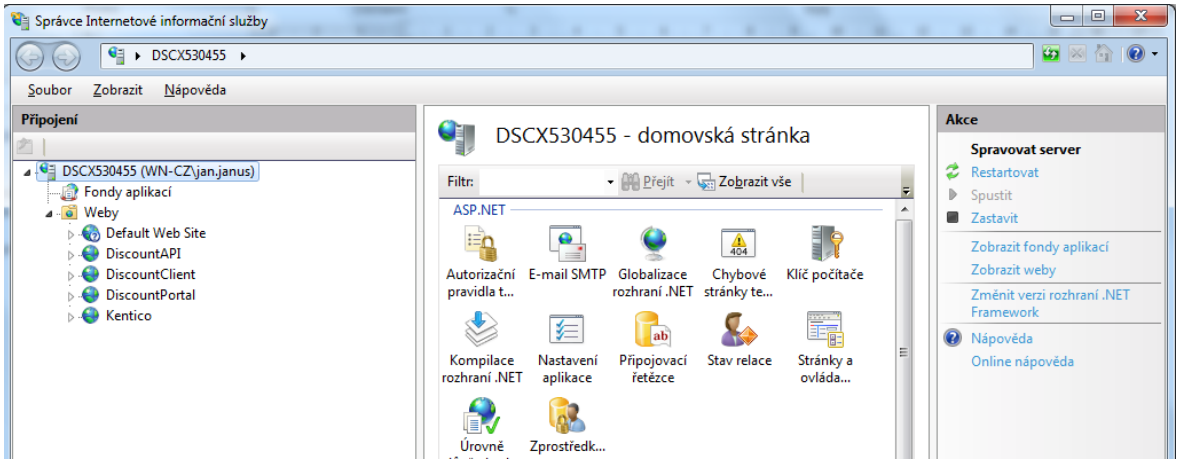
XAMPP je volně dostupné open source multiplatformní řešení webového serveru a dalších komponent vytvořených komunitou Apache Friends. XAMPP se skládá z webového serveru Apache, MySQL databáze (od verze 5.5.30 a 5.6.14 se používá MariaDB), PHP a Perlu. Tvůrci XAMPPu jej oficiálně chtěli použít pouze jako vývojářský nástroj, který by umožnil webovým tvůrcům a vývojářům otestovat jejich práci na jejich vlastních počítačích bez nutnosti přístupu k Internetu. Většina důležitých bezpečnostních funkcí je ve výchozím stavu deaktivována, aby bylo použití XAMPPu, co nejjednodušší. XAMPP byl využit především z důvodu zprovoznění internetového obchodu Magento a následném průzkumu jeho marketingových funkcí. Logo aplikace XAMPP je zobrazeno na níže uvedeném obrázku (*Obr. 21*) [21].

Obr. 21. Logo XAMPP [21].

6.2 IIS (Internet Information Services)

IIS je webový server v prostředí Windows, který přijímá požadavky od vzdálených klientských stanic a vrací požadovanou odpověď na požadavek. Tato funkcionalita umožňuje webovému serveru sdílet a doručovat informace v rámci počítačové sítě. Web server umožňuje doručit informace uživateli v několika formách, a to např. formou statické webové stránky kódované v HTML, přes výměnu souborů jako je download a upload, textové soubory nebo obrazové soubory a další.

IIS funguje v rámci několika standardních jazyků a protokolů. HTML je používáno především k vytvoření elementů, jako jsou texty, tlačítka, obrázky a hyperlinky. HTTP se používá jako základní komunikační protokol k výměně informací mezi serverem a uživateli. Dále lze využít HTTPS, tedy HTTP za použití SSL nebo TLS pro šifrovanou komunikaci, což lze zajistit vložením certifikátu do IIS serveru. Prostředí IIS serveru je znázorněno na níže uvedeném obrázku (*Obr. 22*) [22].



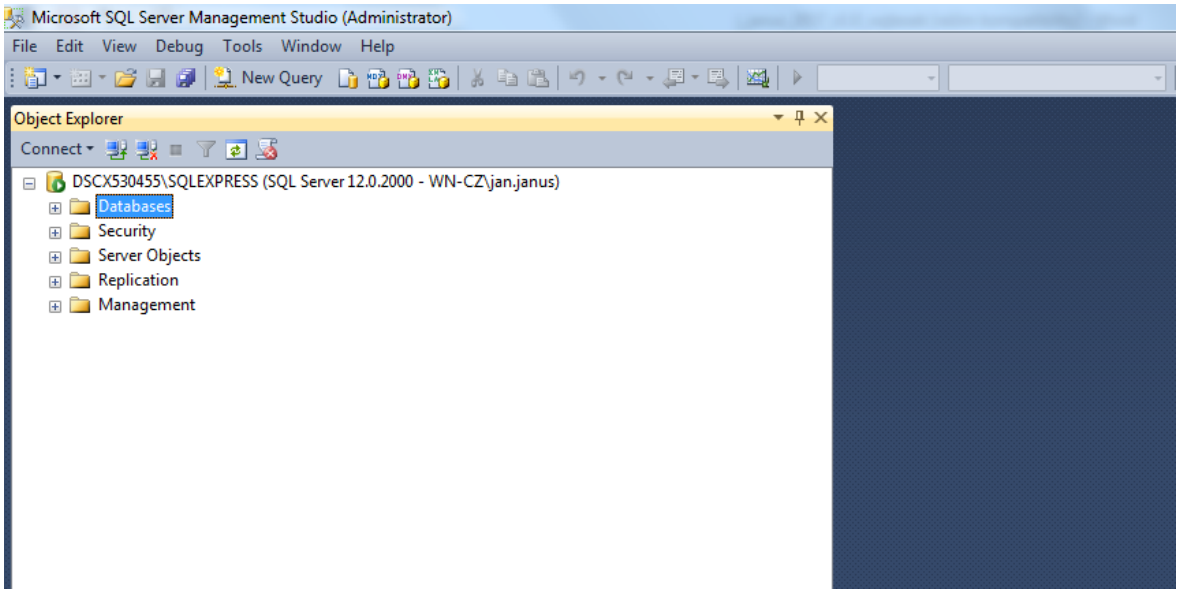
Obr. 22. Prostředí IIS serveru.

IIS serveru je využito pro hostování všech vyvinutých aplikací v rámci této práce. Současně IIS umožnilo zprovoznit řešení internetového obchodu Kentico, na kterém došlo k analýze jeho marketingových funkcí.

6.3 MSSQL Server

Microsoft SQL Server je systém pro správu relačních databází vyvinutý společností Microsoft. Jakožto databázový server se jedná o softwarový produkt, který slouží primárně k ukládání a získávání dat z databáze na základě požadavku jiných softwarových aplikací, které mohou běžet na stejném serveru, kde je hostován SQL Server nebo na jiném napříč počítačovou sítí. Microsoft nabízí minimálně tucet rozdílných verzí Microsoft SQL Serveru, které jsou zaměřeny na různé uživatele a pracovní zátěž, tedy od malé aplikace až po velké aplikace, které čelí velkému množství současně připojených uživatelů. Pro účely tohoto projektu byla použita verze Express, která je zdarma a obsahuje databázový engine. Zatímco tato verze nemá žádnou limitaci na počet databází nebo uživatelů, tak je limitována na použití jednoho procesoru, 1 GB paměti a 10 GB databázových souborů.

Datové uložiště je databáze, která shromažďuje kolekci tabulek s různými typy sloupců. SQL Server podporuje různé datové typy, včetně primárních typů jako jsou *Integer*, *Float*, *Decimal*, *Char*, *Varchar*, *binary*, *Text* a další. Uložené místo alokované pro databázi je rozděleno do sekvenčně číslovaných stránek, každá o velikosti 8 KB. SQL Server ukládá stránky do paměti RAM, aby minimalizoval zápis a čtení z disku. Ke správě databázového serveru se využívá softwarového nástroje Microsoft SQL Server Management Studio (*Obr. 23*) [23].



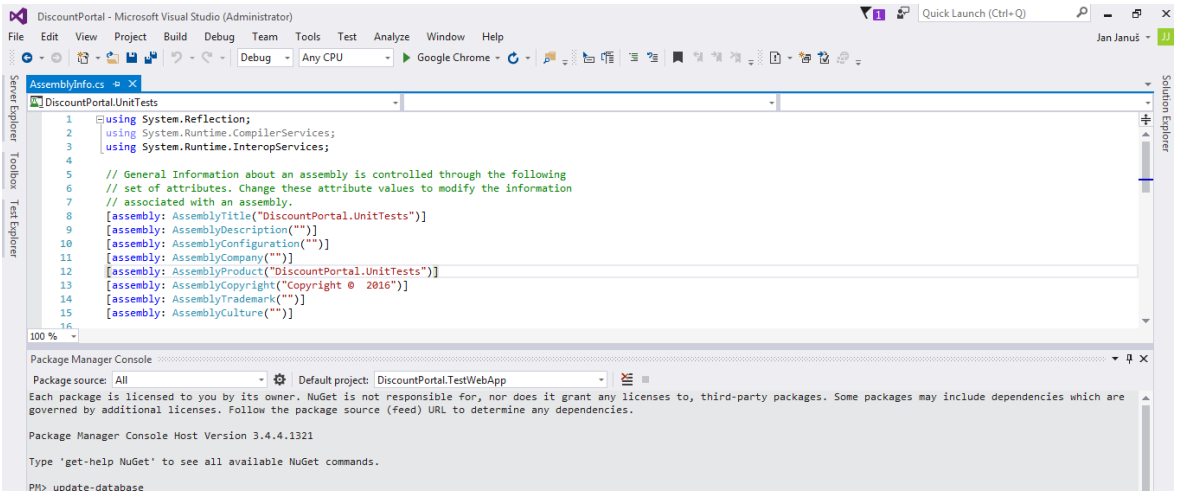
Obr. 23. Prostředí Microsoft SQL Server Management Studio.

6.4 Microsoft Visual Studio

Microsoft Visual Studio je integrované vývojové prostředí (IDE) od společnosti Microsoft. Jeho uplatnění najdeme zejména v oblasti vývoje software pro OS Microsoft Windows, stejně tak webových stránek, webových aplikací, webových služeb či mobilních aplikací. Visual Studio využívá Microsoft vývojové platformy jako je Windows API, Windows Forms a podobně.

Visual Studio obsahuje editor kódu, který podporuje funkci IntelliSense, stejně tak funkci pro refaktorizaci kódu. Integrovaný debugger funguje jako debugger na úrovni kódu a debugger na úrovni stroje. Další zabudované nástroje obsahují např. webový návrhář, návrhář tříd, a návrhář databázového schématu. Podporu mají také různorodé pluginy, které rozšiřují funkcionalitu téměř ve všech směrech, obsahem takového pluginu může být podpora pro verzování kódu.

Visual Studio podporuje různé programovací jazyky a umožňuje editoru a debuggeru podporovat (v různé míře) téměř libovolný programovací jazyk za předpokladu, že existuje specifická jazyková služba. Zabudované jazyky tvoří například C, C++, C#, TypeScript. Microsoft poskytuje zdarma verzi Visual Studia, která se nazývá Community edition, která podporuje také pluginy a není za ni potřeba nic platit. Jako zajímavost lze uvést, že software pro správu databáze SQL Server Management Studio byl naprogramován právě pomocí Visual Studia. Ukázku prostředí aplikace Visual Studio lze vidět zobrazenou na obrázku níže (*Obr. 24*) [24].



Obr. 24. Ukázka prostředí Visual Studio.

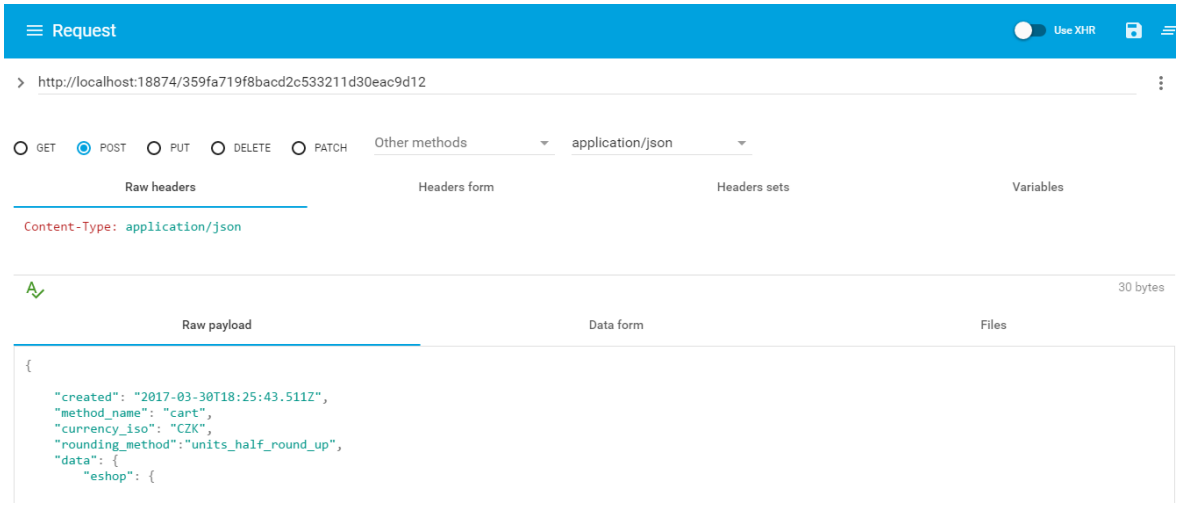
Pro účely tohoto projektu byl nástroj využit v plném rozsahu. Veškerá aplikační logika včetně datových struktur byla tvořena za pomoci tohoto softwarového nástroje.

6.5 Enterprise Architect

Sparx Systems Enterprise Architect je modelovací a tvůrčí nástroj založený na OMG UML. Platforma podporuje návrh a konstrukci softwarových systémů, modelování business procesů a modelování průmyslových domén. Modelování systémů pomocí UML poskytuje základ pro modelování všech aspektů organizační architektury spolu se schopností poskytnout základy pro navrhování a implementaci nových systémů nebo změnu stávajících systémů. Aspekty, které lze pokrýt tímto typem modelování, zahrnují organizační nebo systémové architektury, architektury orientované na služby a webové modelování. Enterprise Architect pokrývá základní aspekty životního cyklu vývojových aplikací, od řízení požadavků až po fázi návrhu, konstrukce, testování a údržby [25].

6.6 Advanced REST Client

Advanced REST Client API je využíván k otestování interakce mezi různými úrovněmi softwarových komponent. Testování se provádí pomocí koncových bodů API, které provádí validaci na základě návratové hodnoty. Odpověď může být úspěšná *true* a nebo úspěšná *false*. Ve volání Advanced REST Client API se provádí metody GET a POST. V metodě GET jsou předávány klíčové hodnoty spolu s adresou URL, zatímco v metodě POST jsou informace přenášeny skrytě. V metodě POST jsou páry název/hodnota uzavřeny v těle požadavku HTTP, což činí mnohem kompaktnější a bezpečnější adresu URL, tudíž metodu POST lze označit za bezpečnější než GET. To je také jeden z důvodu, proč byla zvolena jako primární metoda pro komunikaci metoda POST. Ukázku dotazu z prostředí aplikace lze vidět na obrázku níže (*Obr. 25*) [26].



Obr. 25. Ukázka dotazu z prostředí aplikace Advanced REST Client API.

II. PRAKTICKÁ ČÁST

7 ANALÝZA A MAPOVÁNÍ

Hlavním záměrem analýzy je definovat požadavky na nový slevový systém, které budou pokrývat jak technickou tak marketingovou oblast. Ještě než začneme definovat požadavky funkčního charakteru, tak je dobré zdůvodnit, proč byla pro tvorbu systému zvolna platforma ASP.NET od společnosti Microsoft. Tato platforma má širokou podporou dokumentací a tutoriálů na řešení konkrétních problémů. Prostředí operačního systému Windows, který tuto platformu hostuje, umožňuje v kombinaci s dalšími nástroji velmi provázanou integraci a koncepční vývoj. Platforma je také velmi oblíbená u velkých korporací, které sáhnou raději po ASP.NET řešení, než po open source, navzdory tomu, že je Microsoft řešení z hlediska financí podstatně dražší.

Neméně důležité je také zmínit důvod výběru šablony sb-admin2 pro administraci slevového portálu. Sb-admin2 je administrační šablona založená na frameworku Bootstrap. Šablona je otevřená modifikacím, jak na úrovni jQuery, kdy je možné přidávat další pluginy či upravovat stávající funkce, tak na úrovni souborů Less. Fundamentální funkcí je responzivní design, který umožňuje zobrazení, jak na mobilních zařízeních, tak na desktopech.

Pro znázornění jednotlivých funkčních i nefunkčních požadavků bylo využito znázornění ve formě UML diagramu. Jednotlivé požadavky jsou rozepsány do uživatelských scénářů, aby mohlo dojít k hlubšímu pochopení problematiky.

7.1 Definice cíle

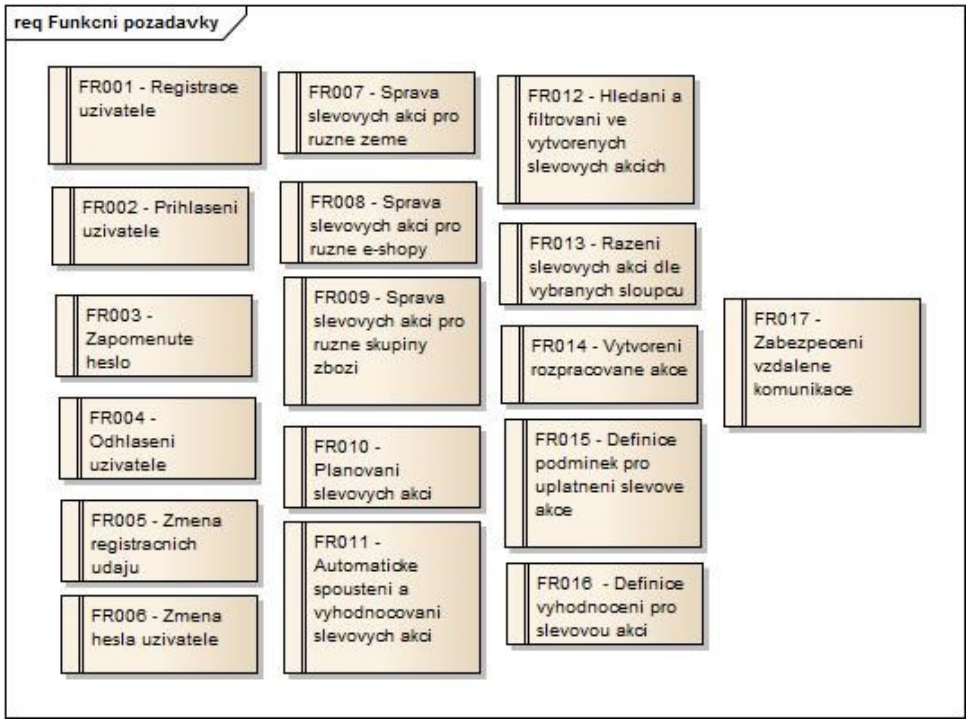
Cílem tohoto projektu je vytvoření serverové aplikace, která bude komunikovat s klienty skrze API rozhraní. Systém bude jako celek sloužit pro nastavení, plánování, monitorování slevových akcí. Výsledný produkt bude cílen na střední až velké společnosti a bude poskytován jako nezávislá aplikace instalovaná v cloudovém prostoru. Produkt bude navenek prezentován jako aplikace slevového portálu s přihlášením a registrací. Po přihlášení bude k dispozici spektrum klíčových funkcí definovaných ve funkčních požadavcích a blíže popsaných v případech užití.

7.2 Funkční požadavky

Funkční požadavky definují základní požadavky na aplikaci a stanovují hranice funkcí aplikace, tudíž je možné si na první pohled uvědomit jaké oblasti aplikace bude řešit a jaké už

ne. Jednotlivé funkční požadavky jsou definovány níže a znázorněny formou diagramu na obrázku níže (*Obr. 26*).

- Systém umožní registraci nového uživatele.
- Systém umožní přihlášení registrovaného uživatele.
- Systém umožní funkci zapomenutého hesla.
- Systém umožní funkci odhlášení uživatele.
- Systém umožní dodatečnou změnu registračních údajů uživatele. Nebude však možné změnit e-mail uživatele.
- Systém umožní dodatečnou změnu hesla uživatele.
- Systém umožní spravovat slevové akce pro různé země.
- Systém umožní spravovat slevové akce pro různé e-shopy.
- Systém umožní spravovat slevové akce pro různé množiny zboží.
- Systém umožní plánovat slevové akce.
- Systém umožní automatické vyhodnocování a spouštění slevových akcí.
- Systém umožní hledání a filtrování ve vytvořených slevových akcích.
- Systém umožní řazení slevových akcí dle vybraných sloupců.
- Systém umožní vytvořit rozpracovanou slevovou akci.
- Systém umožní definici podmínek pro uplatnění slevové akce (obrat košíku, registrovaný zákazník, virtuální skupiny zboží).
- Systém umožní definovat jedno vyhodnocení pro jednu slevovou akci.
- Vzdálená komunikace se systémem bude zabezpečena.

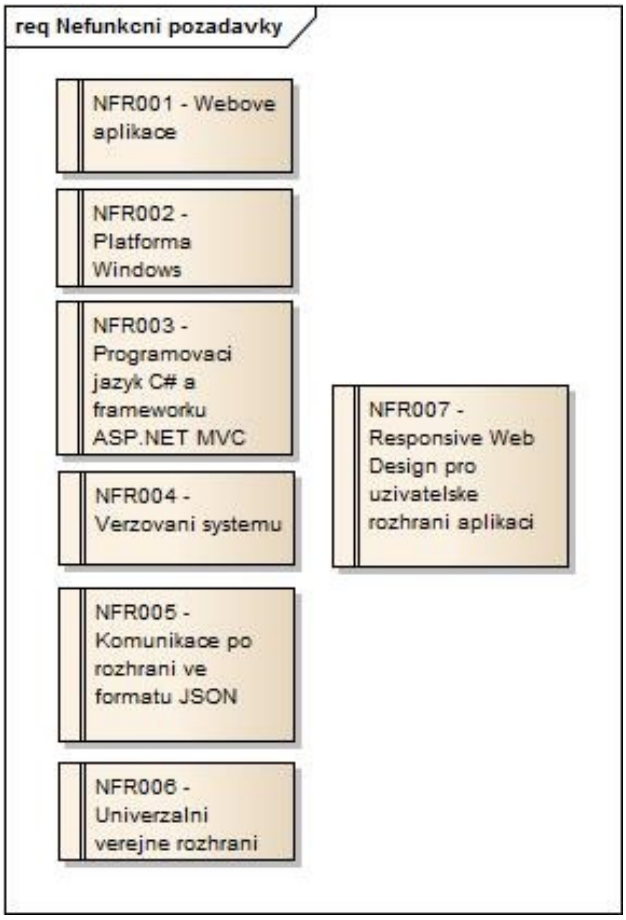


Obr. 26. Diagram funkčních požadavků.

7.3 Nefunkční požadavky

Nefunkční požadavky definují architekturu a prostředí, ve kterém bude aplikace provozována. Také definují formáty výměny dat a údržbu systému. Jednotlivé nefunkční požadavky jsou definovány níže a znázorněny formou diagramu na obrázku níže (*Obr. 27*).

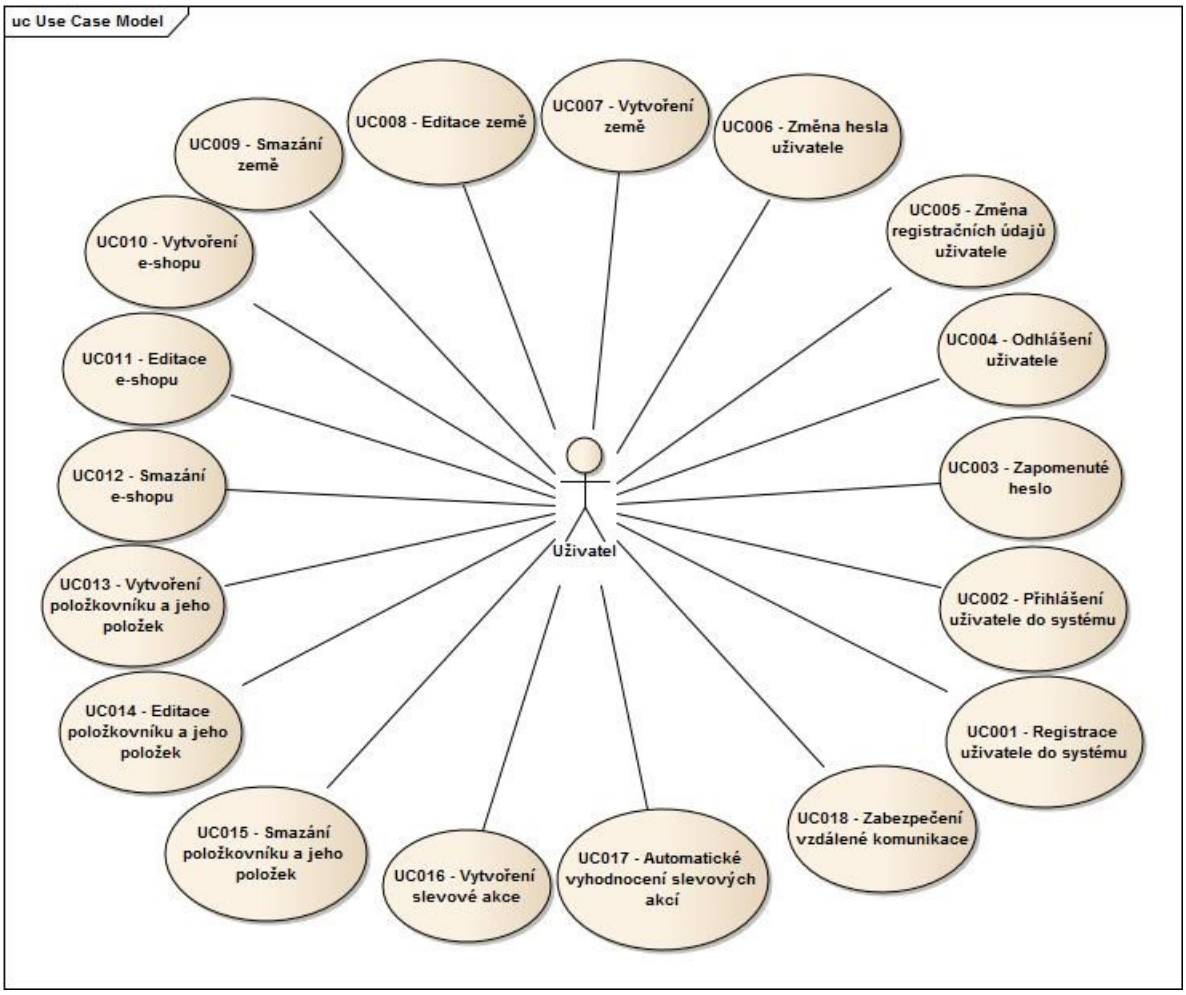
- Systém bude dostupný jako webová aplikace.
- Systém bude provozován na platformě Windows.
- Systém bude naprogramován v jazyce C# a frameworku ASP.NET MVC.
- Systém bude verzován ve vazbě na opravy chyb či implementaci rozšiřujících funkcionalit.
- Komunikace skrze rozhraní mezi klientem a serverem bude probíhat ve formátu JSON.
- Univerzální veřejné rozhraní. Kód klientské části může být implementován třetí stranou nezávisle na programovacím jazyce a platformě internetového obchodu.
- UI bude navrženo v technologii RWD (Responsive Web Design).



Obr. 27. Diagram nefunkčních požadavků.

7.4 Případy užití (use cases)

Případy užití neboli use case realizují jednotlivé funkční požadavky. Díky případům užití bylo možné rozkrýt detail implementace a vyhnout se tak případným nečekaným překvapením v jejím průběhu. Může existovat více uživatelských scénářů k jednomu funkčnímu požadavku. Vzhledem ke komplexnosti práce byly popsány pouze základní případy užití. Jednotlivé případy užití jsou zobrazeny v diagramu na *Obr. 28* níže. Podrobný popis 18-ti základních případů užití je uveden v příloze práce jako příloha P2.

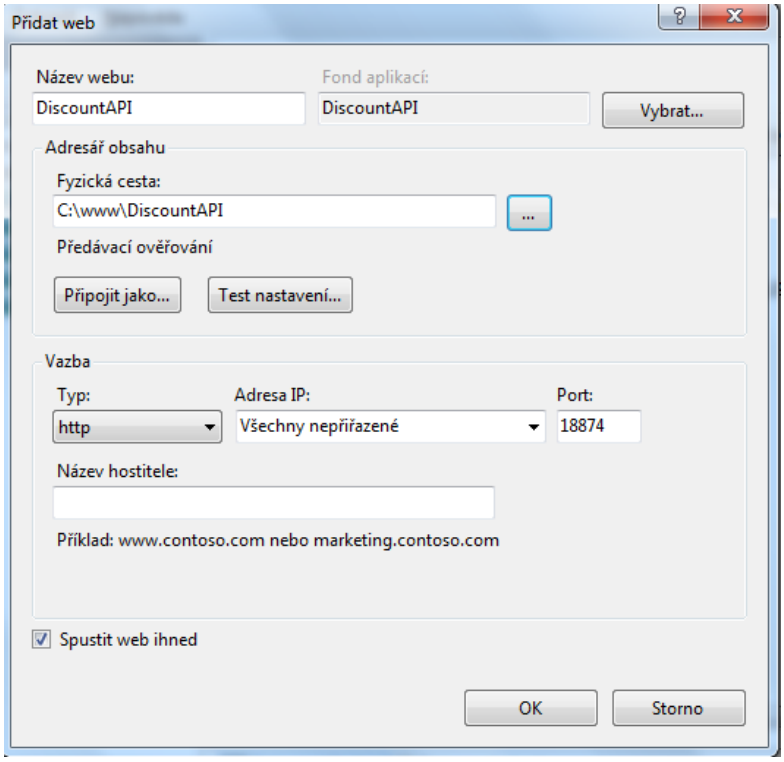


Obr. 28. Use case model diagram.

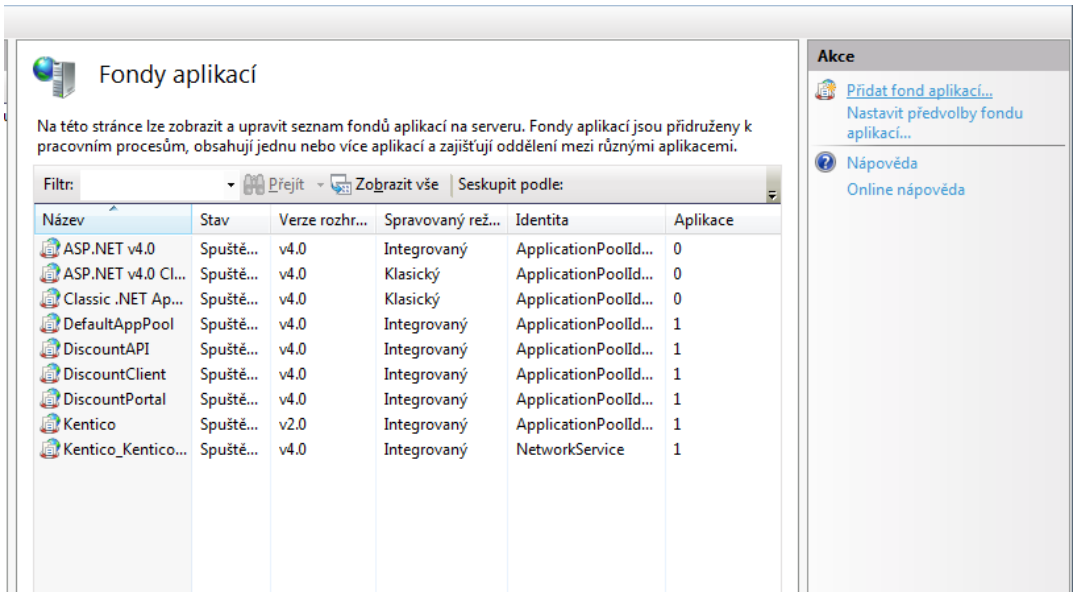
8 INSTALACE APLIKACÍ DO IIS

Před započetím vývoje každé aplikace byla vygenerována šablona webové aplikace ve Visual Studio. Byl vytvořen první build, jehož obsah byl nasměrován do IIS serveru, který bylo potřeba nainstalovat, protože v základní konfiguraci Windows webový server IIS instalován není.

Prvním důležitým krokem po instalaci IIS bylo vytvoření aplikačního fondu, kterému bylo potřeba nastavit verzi rozhraní na .NET Framework v4.0.30319. Každá z vyvinutých aplikací běží na .NET Frameworku 4.5.2, tudíž je potřeba mít instalovanou minimálně tuto verzi .NET Frameworku. Ve druhém kroku bylo nutné přidat nový web do správy webů, kterému bylo potřeba vyplnit název, nastavit fyzickou cestu a také nastavit volný port, na kterém webová aplikace poběží. Přidání webu znázorňuje níže uvedený obrázek (*Obr. 29*). Správu aplikačních fondů, pak znázorňuje obrázek pod ním (*Obr. 30*).



Obr. 29. Přidání webu do IIS serveru.



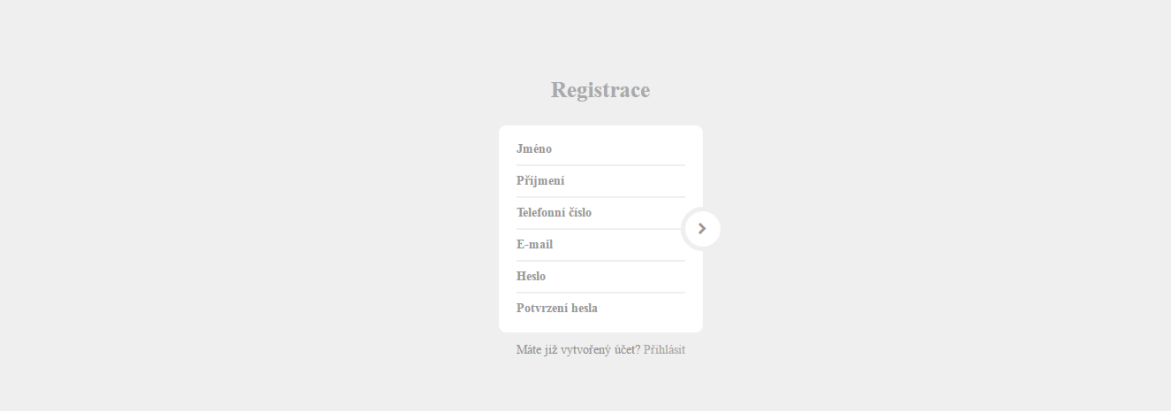
Obr. 30. Správa aplikačních fondů.

9 WEBOVÁ APLIKACE SLEVOVÉHO PORTÁLU

Aplikace slevového portálu je navržena jako cloudová aplikace, která počítá do budoucna s dynamickým horizontálním škálováním, tudíž se vzrůstajícím počtem uživatelů bude možné data těchto uživatelů distribuovat mezi infrastrukturu důkladně zabezpečených serverových řešení. Systém bude schopen samostatně vyhodnotit, zda potřebuje další server k dispozici. Pokud ano, pak zavolá funkci, která zajistí přípravu nového serveru.

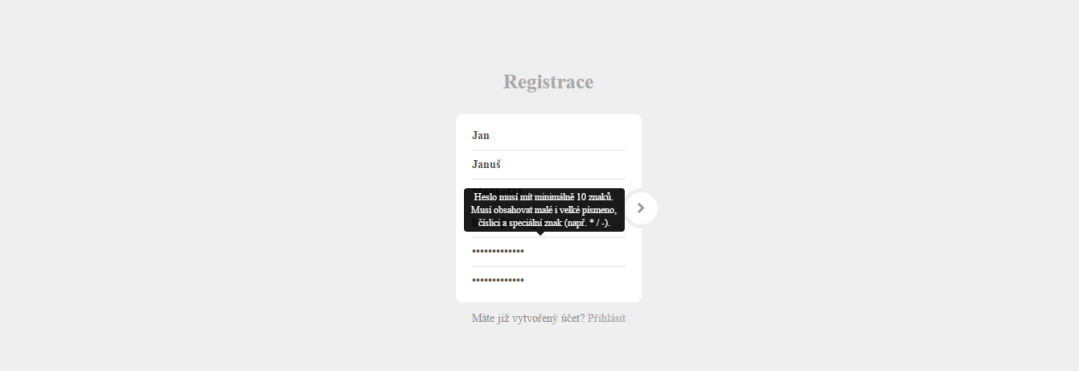
9.1 Registrace

K používání slevového systému je potřeba se nejdříve zaregistrovat do webové aplikace portálu. Pro registraci byl vytvořen pouze jednoduchý formulář se základními údaji o uživateli, tak aby byla registrace pro uživatele, co nejjednodušší. Níže uvedený obrázek (*Obr. 31*) znázorňuje registrační formulář do portálu.



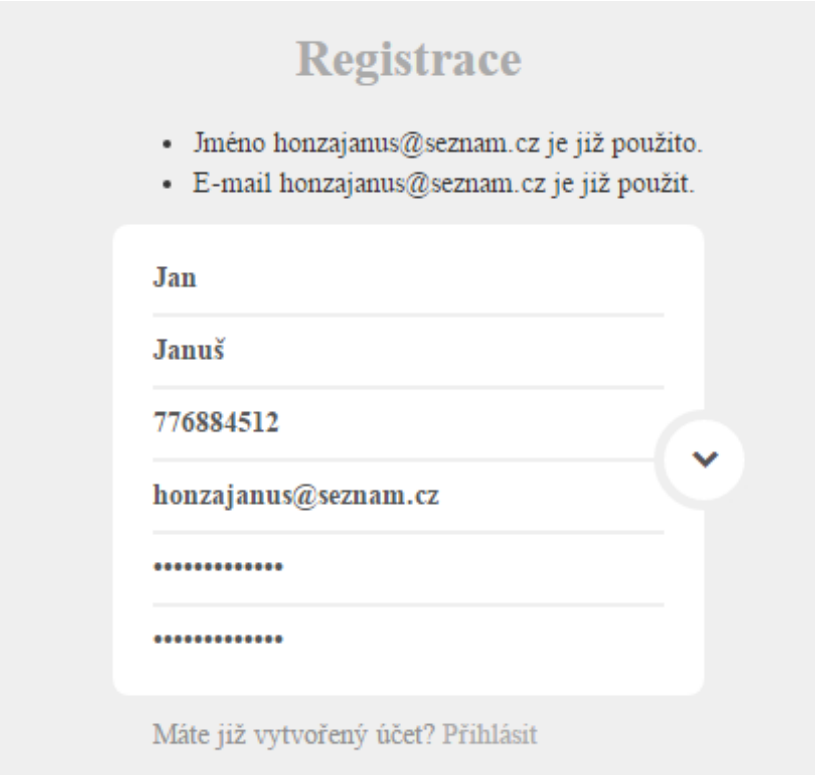
Obr. 31. Registrace uživatele do portálu.

Údaje je nutné vyplnit, tak aby byly splněny všechny validace formuláře, tedy zejména forma hesla uživatele. Níže uvedený obrázek (*Obr. 32*) znázorňuje zobrazení validačního hlášení při nesprávném tvaru hesla.



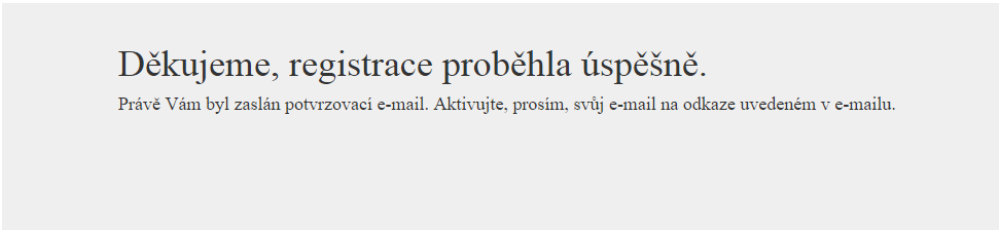
Obr. 32. Zobrazení validačního hlášení při nesprávně zadaném hesle.

Pokud v databázi již existuje zadaný e-mail, pak je uživatel upozorněn o této skutečnosti chybovým hlášením, což lze pozorovat na níže uvedeném obrázku (*Obr. 33*).



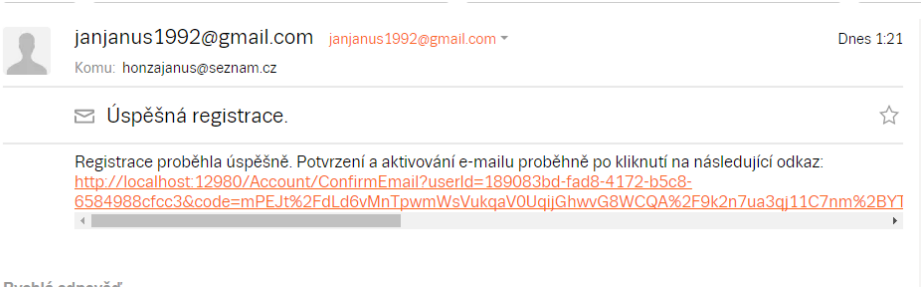
Obr. 33. Zobrazení chybového hlášení při existujícím e-mailu.

Po úspěšné registraci je uživateli zaslán aktivační e-mail. Tato forma byla zvolena především vzhledem k aktuálním trendům bezpečnosti v oblasti webů. *Obr. 34* níže znázorňuje stav úspěšně dokončené registrace.



Obr. 34. Úspěšné dokončení registrace.

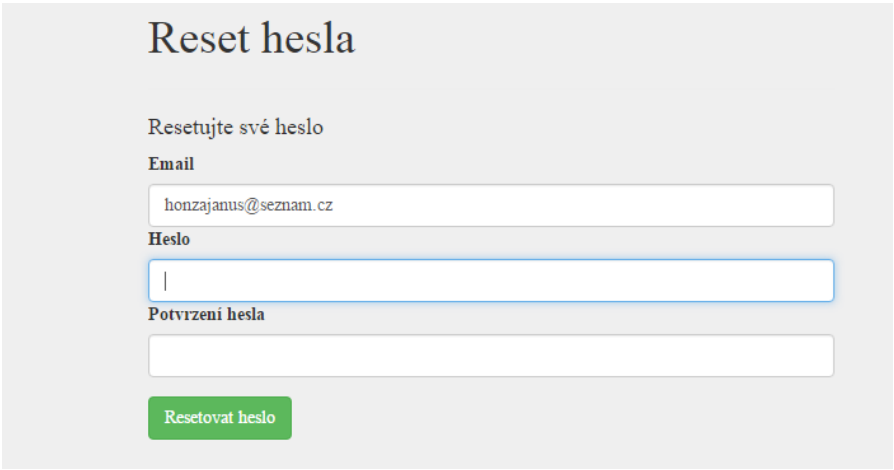
Aktivační link, který je zaslán v každém aktivačním e-mailu se skládá z unikátního id uživatele a dále kódu, do kterého je vygenerován unikátní textový řetězec. Tímto je zamezeno zneužití aktivačního linku. Forma aktivačního linku je znázorněna na níže uvedeném obrázku (*Obr. 35*).



Obr. 35. Aktivační link pro aktivaci účtu.

Jakmile dojde k potvrzení aktivačního linku, tak je účet aktivní a je možné jej začít využívat, což znázorňuje následující obrázek (*Obr. 36*). Stejný princip lze využít při funkci zapomenuté heslo, kde je taktéž vygenerován unikátní link do e-mailu a pokud jej uživatel navštíví, pak má možnost změnit si svůj přístupový údaj viz níže uvedený obrázek (*Obr. 37*).

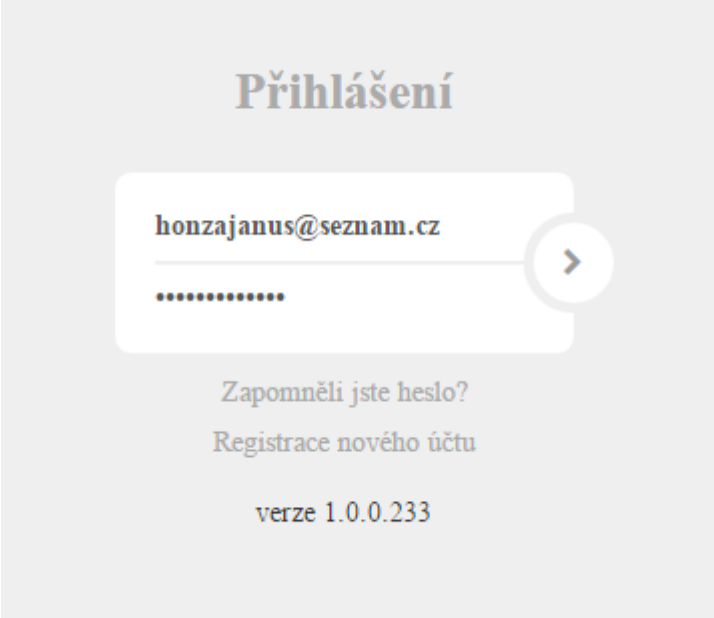
Obr. 36. Úspěšná aktivace.



Obr. 37. Zapomenuté heslo.

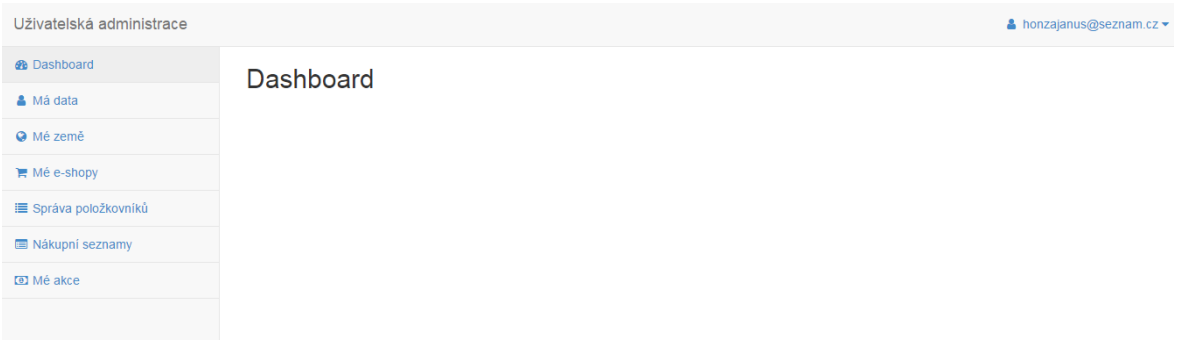
9.2 Přihlášení

Pokračovat lze kliknutím na link pro přihlášení, kde lze provést zadání přihlašovacích údajů a potvrdit vstup do aplikace. Lze si také všimnout, že se zde zobrazuje verze aplikace, která je generována prozatím manuální změnou v kódu aplikace. Přihlašovací formulář je znázorněn na obrázku níže (*Obr. 38*).



Obr. 38. Přihlašovací formulář.

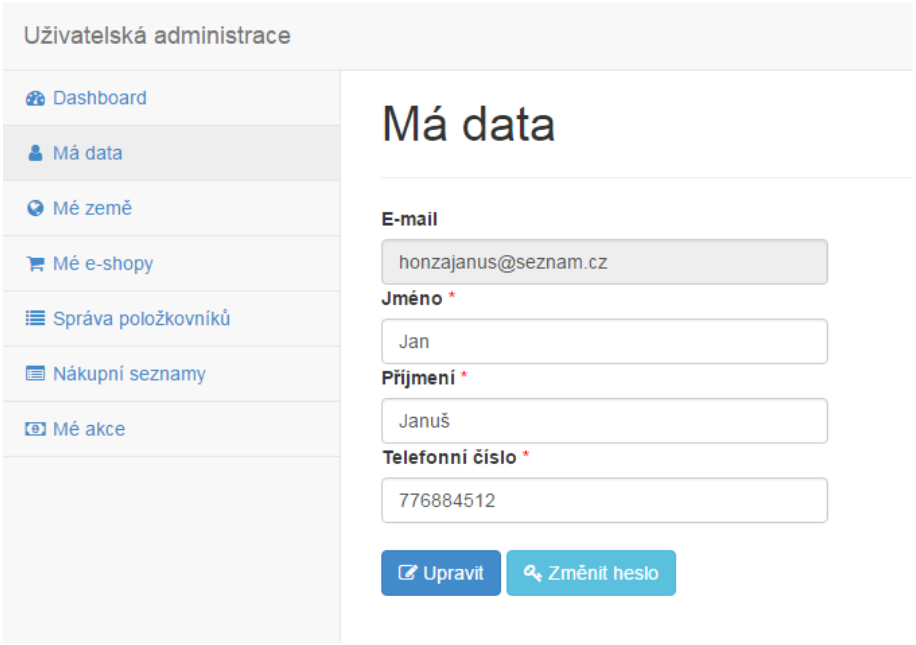
Po přihlášení do aplikace je zobrazena první položka menu (*Obr. 39*), která bude v budoucnu obsahovat ucelený statistický přehled týkající se aplikovaných slev či aktuální licenční úrovně a dalších informací spojených s čerpáním licence. Aktuálně nemá žádný obsah.



Obr. 39. Zobrazení dashboard.

9.3 Má data

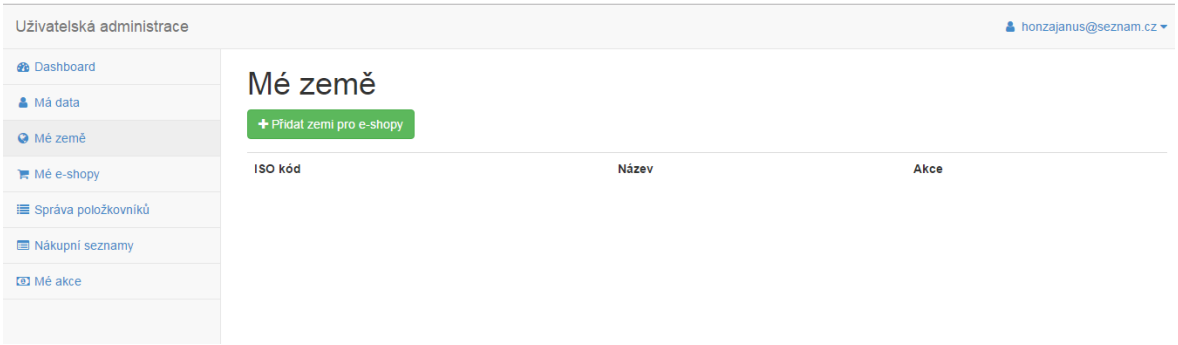
Pokud klikneme na položku menu *Má data*, pak lze upravovat uživatelská data a také lze provést změnu hesla. Pole e-mail zůstává needitovatelné, jelikož se jedná o jednoznačný identifikátor uživatele, který nelze dodatečně měnit. Obsah modulu *Má data* je znázorněn na níže uvedeném obrázku (*Obr. 40*).



Obr. 40. Zobrazení modulu Má data.

9.4 Mé země

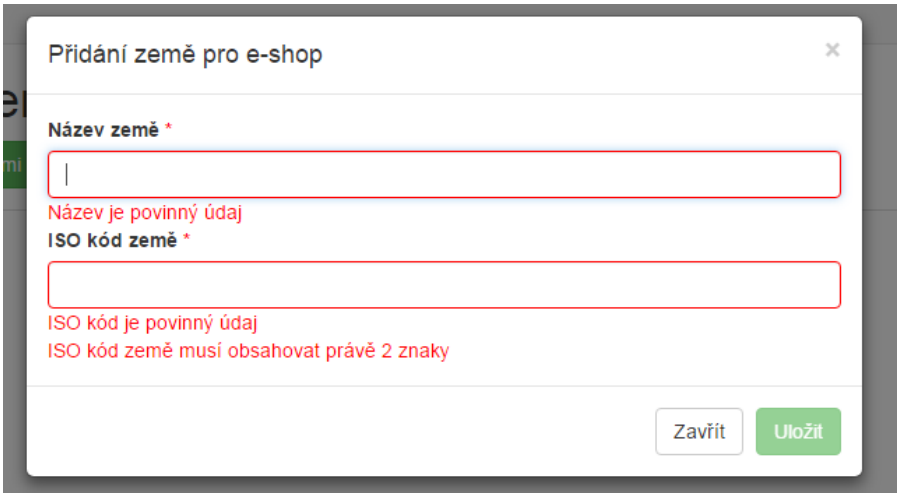
Jedním z primárních nastavení, bez kterého nelze dále pokračovat, je nastavení zemí. Pro nastavení zemí je zapotřebí kliknout na položku menu *Mé země,* viz níže uvedený obrázek (*Obr. 41*). Dále je zapotřebí pokračovat kliknutím na tlačítko *Přidat zemi pro e-shopy*. Jak si lze všimnout, tak je tlačítko v zelené barvě s ikonkou plus a je umístěno nalevo od menu. Tímto způsobem jsou navržena všechna tlačítka pro přidávání záznamů. Uživatel má vše dostatečně blízko a díky jednotné úpravě bude po několika hodinách přesně vědět, kde požadovanou funkci hledat.



Obr. 41. Zobrazení modulu Mé země.

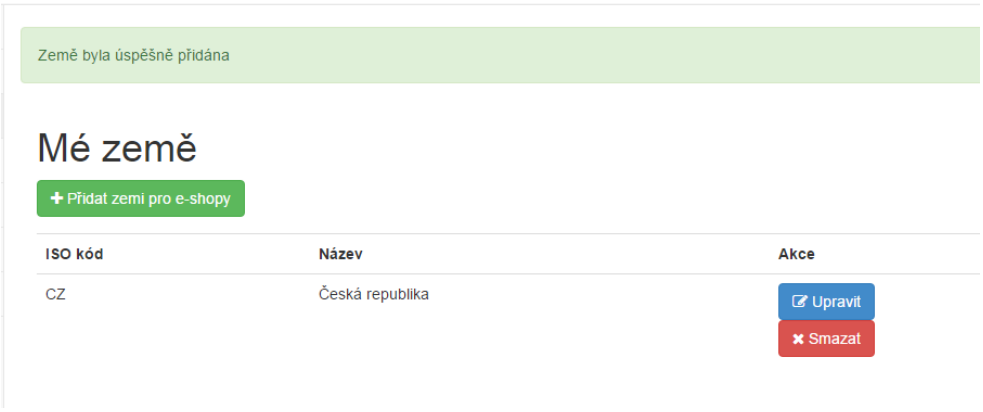
Událost, která se spustí po kliknutí na tlačítko, zajistí zobrazení modálního okna s možností zadání ISO kódu země. ISO kód následuje standard ISO Alpha-2, který umožňuje pouze 2 znaky, kvůli tomuto omezení je přizpůsobena také validace pole v modálním okně, jak je

znázorněno na *Obr. 42* níže. Je taktéž ošetřeno uložení bez správně vyplněných údajů na úrovni tlačítka *Uložit*, které je neaktivní do doby, než je vše v pořádku vyplněno. Tato kontrola se děje na straně serveru.



Obr. 42. Validace modálního okna přidání země.

Po korektním vyplnění všech údajů a uložení země se země objeví mezi seznamem založených zemí. Vytvořenou zemi lze editovat nebo smazat pomocí tlačítek uživatelských akcí. Možnosti editace a smazání existující země jsou znázorněny níže uvedeným obrázkem (*Obr. 43*).

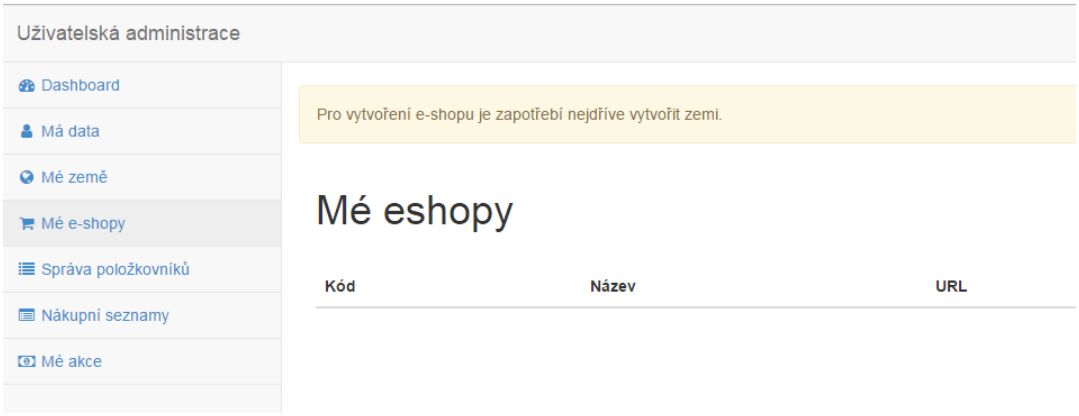


Obr. 43. Úspěšné založení země.

9.5 Mé e-shopy

Dalším důležitým krokem, který je v nastavení nutné provést je vytvoření nového e-shopu. Princip tohoto úkonu je téměř shodný s vytvořením země, nicméně rozdíl je ve vytvoření relace na zemi, tudíž z toho vyplývá, že e-shop lze vytvořit pouze za předpokladu, že existuje minimálně 1 země. Tato situace je uživatelsky podchycena formou upozorňujícího hlášení

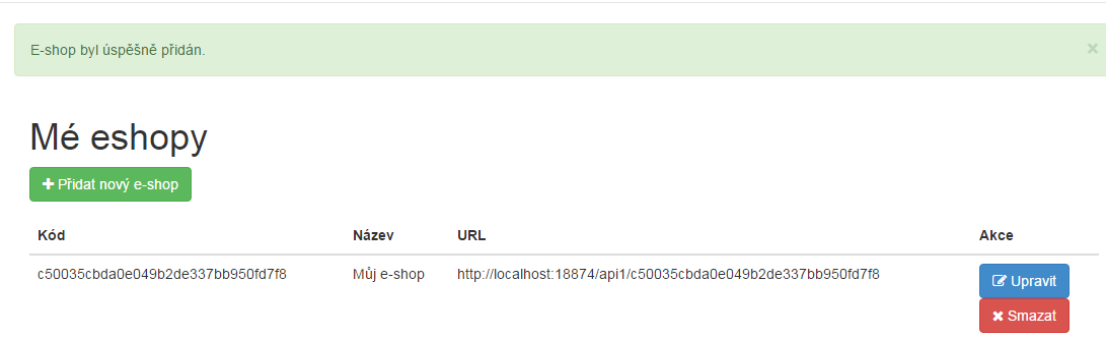
(*Obr. 44*), které uživatele navádí nejdříve vytvořit zemi a pak se vrátit k vytvoření e-shopu. Níže uvedený obrázek (*Obr. 45*) znázorňuje modální okno, které slouží pro založení e-shopu.



Obr. 44. Upozornění na nutnost vytvoření e-shopu.

Obr. 45. Založení e-shopu.

Pokud dojde k vytvoření e-shopu, pak je vygenerována URL adresa API a kód e-shopu. URL adresa API se odvíjí od domény, na které je aplikace portálu hostována. Dále je součástí URL adresy verze API, která má svůj význam z hlediska budoucího vývoje a zpětné kompatibility. Zobrazení vytvořeného e-shopu znázorňuje níže uvedený obrázek (*Obr. 46*).

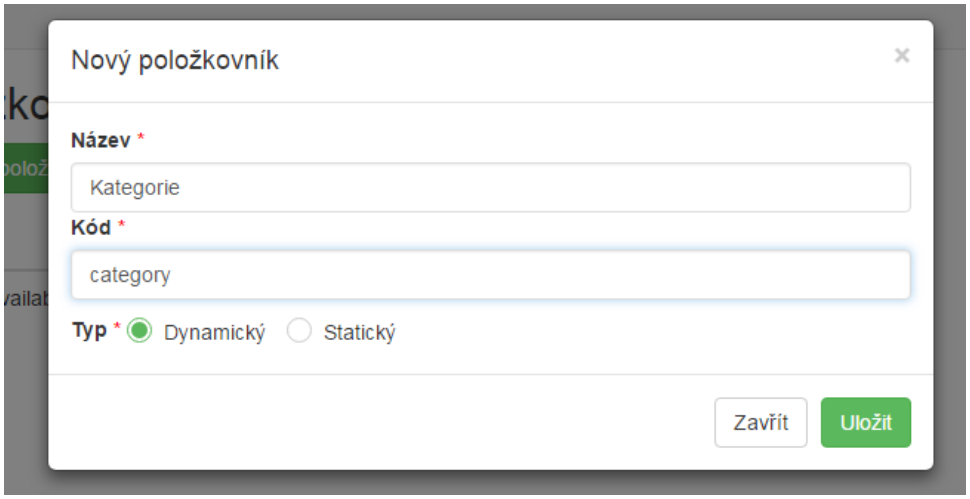


Obr. 46. Zobrazení vytvořeného e-shopu.

Vytvořený e-shop lze smazat nebo jej lze editovat pomocí tlačítek uživatelských akcí. Pokud existuje vytvořený e-shop, pak jsou splněny všechny základní předpoklady k nastavení první slevové akce.

9.6 Správa položkovníku

Tento modul je velmi důležitou součástí pro nastavení slevových akcí cílených na konkrétní skupinu zboží. Uživatel má možnost vytvořit nový položkovník pomocí tlačítka *Přidat položkovník*. Kliknutím na tlačítko je spuštěna událost pro zobrazení modálního okna, které obsahuje pole pro zadání názvu, kódu a možnosti výběru typu položkovníku, viz obrázek níže (*Obr. 47*).

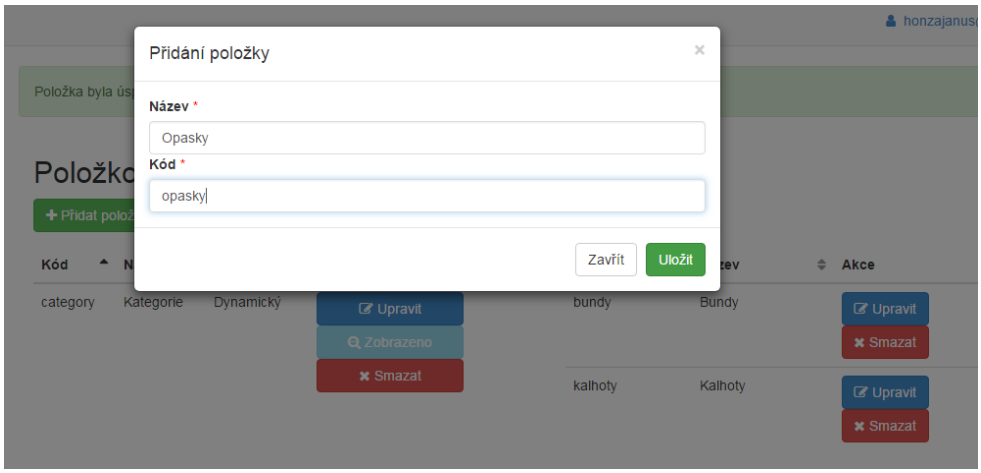


Obr. 47. Založení nového položkovníku.

Pole *Název* slouží pouze pro uživatelskou identifikaci skrze systémem, naopak pole *Kód* je používáno jako primární identifikační údaj při vyhodnocování zboží vloženého zákazníkem do košíku. Výběr *Typ* je používán k možnosti definice funkčního charakteru položkovníku. Pokud se jedná o dynamický typ, pak se pracuje s hodnotami, které definují skupiny zboží na základě společných parametrů např. kategorie Bundy nebo materiál Goretex. Pokud se

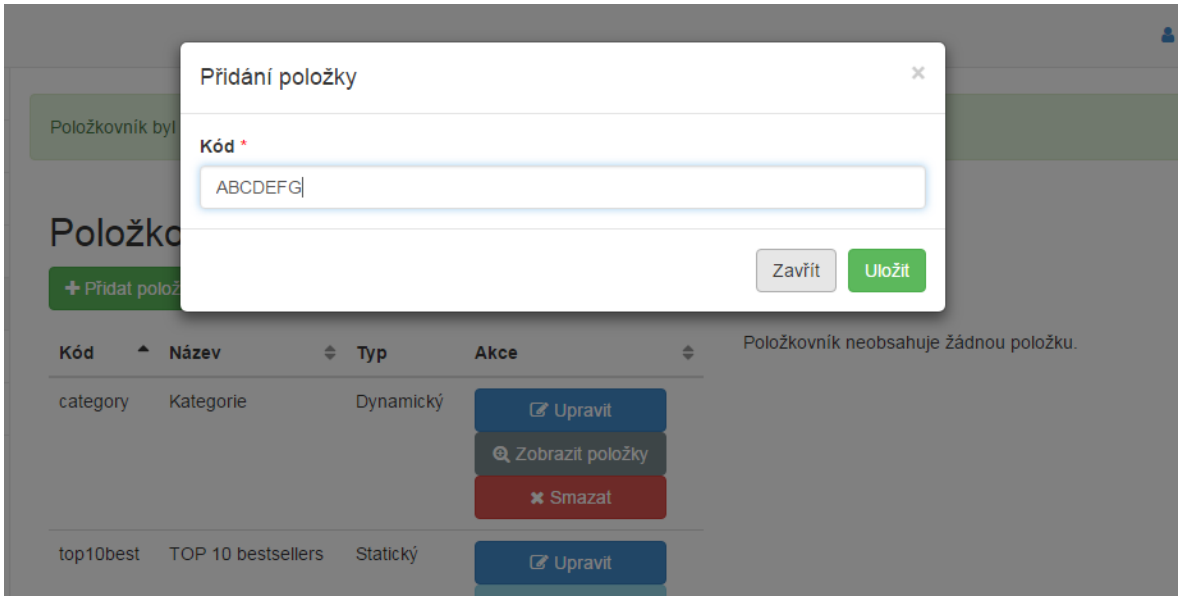
jedná o statický typ, pak položkovník počítá s tím, že jeho obsahem budou kódy produktů, které mohou být vyhodnoceny např. nějakým dataminingovým mechanismem v systému třetí strany např. TOP 10 nejprodávanějších produktů. Po vyplnění všech povinných údajů je možné položkovník uložit pomocí tlačítka *Uložit*.

Pro přidávání položek do položkovníku je nutné nejdříve vybrat položkovník, do kterého chcete přidat položky. Toto lze provést pomocí tlačítka *Zobrazit*, které se nachází mezi tlačítky pro *Upravit* a *Smazat*. Po stisknutí tlačítka se v pravé části obrazovky zobrazí tlačítko *Přidat položku* a je možné přidávat jednotlivé položky do položkovníku. Princip je zde opět podobný jako při vytváření položkovníku. Primárním identifikátorem, se kterým pracuje algoritmus vyhodnocování slevových akcí, je opět pole *Kód*. Při návrhu bylo myšleno na to, aby tento proces probíhal bez rušivých vjemů, tudíž je vše realizováno v technologii ajax, což znamená, že uživatel není rušen neustálým načítáním webové stránky, ale naopak vše se děje asynchronně. Níže uvedený obrázek (*Obr. 48*) znázorňuje přidání položky do vybraného položkovníku.



Obr. 48. Přidání položky do dynamického položkovníku.

Vytváření položky do statického položkovníku se z uživatelského hlediska liší tím, že je potřeba zadat pouze pole *Kód*, do kterého je zapotřebí vyplnit kód produktu. Níže uvedený obrázek (*Obr. 49*) znázorňuje přidání do statického položkovníku.

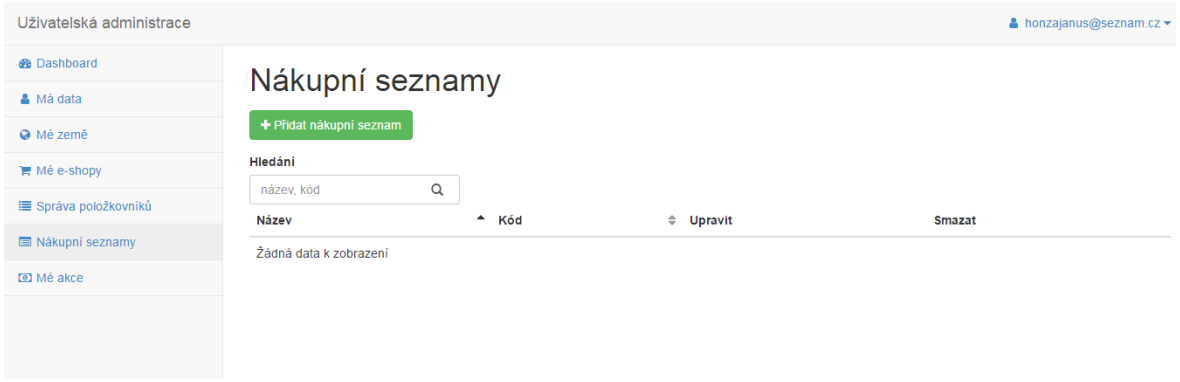


Obr. 49. Přidání položky do statického položkovníku.

Pokud máme vytvořeny položkovníky naplněné položkami, pak je lze využívat v dalších modulech, a to *Mé akce* a také modulu *Nákupní seznamy*.

9.7 Nákupní seznamy

Tento modul slouží pro agregaci jednotlivých položkovníku do jednoho nákupního seznamu, na který lze následně uplatňovat slevové akce. Uživatel má jedinečnou možnost vytvářet virtuální množiny zboží složené z kombinace dynamických a statických položkovníků. Příkladem takového nákupního seznamu může být například seznam Bundy Adidas a TOP 10 nejprodávanějších produktů. Modul nákupních seznamů znázorňuje níže uvedený obrázek (*Obr. 50*).

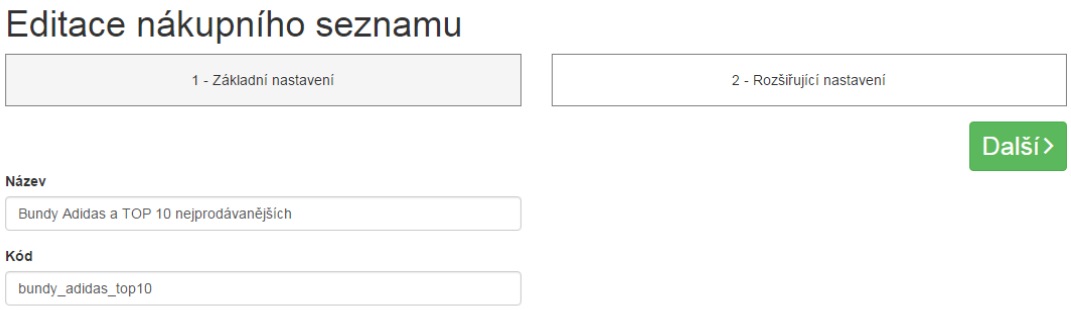


Obr. 50. Zobrazení modulu Nákupní seznamy.

Pokud uživatel klikne na tlačítko *Přidat nákupní seznam*, pak se spustí řízený průvodce pro vytvoření nákupního seznamu.

9.7.1 První krok

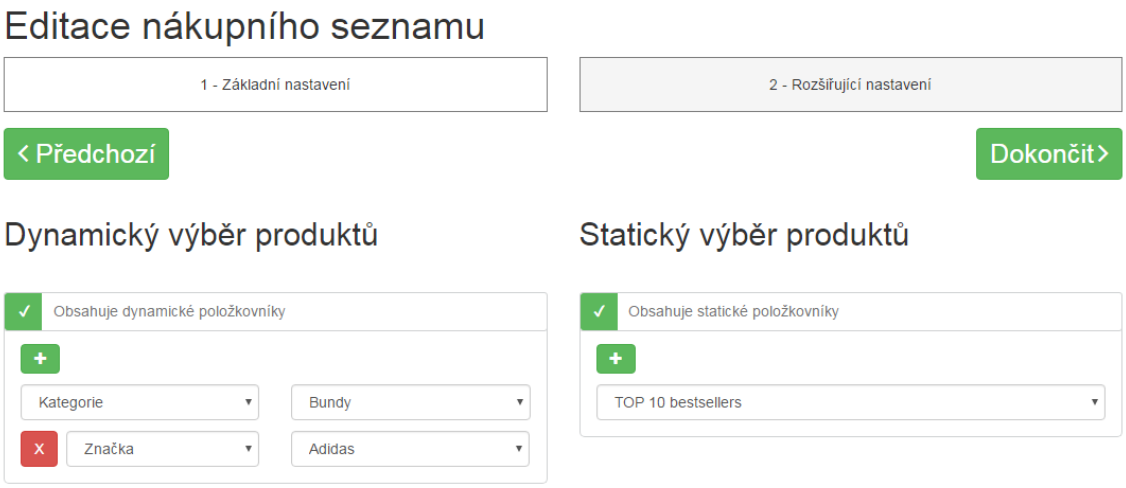
V prvním kroku průvodce je vyžadováno vyplnit pole *Název* a *Kód*. Pokud jsou tato pole vyplněna, pak je možné pokračovat do dalšího kroku. V opačném případě, je uživatel vyzván k vyplnění příslušného nevyplněného pole. První krok je znázorněn na níže uvedeném obrázku (*Obr. 51*).



Obr. 51. První krok tvorby nákupního seznamu.

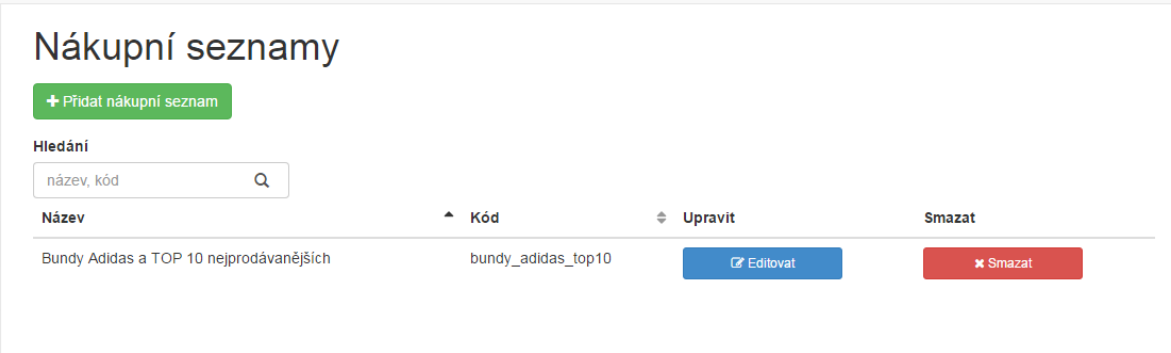
9.7.2 Druhý krok

Druhý krok je rozdělen na dvě poloviny a to dynamický a statický výběr produktů viz níže uvedený obrázek (*Obr. 52*). V levé části, tedy v dynamickém výběru je možné tvořit relace mezi položkami dynamických položkovníků a nákupním seznamem. Je navrženo tak, aby bylo možné přidávat neomezené množství záznamů. Přidávání záznamů je možné zajistit tlačítkem označeným jako *plus* a odstranění je řízeno tlačítkem *křížku*. Pravá strana je prostorem pro přidávání statických položkovníků. Vzhledem k tomu, že se jedná o seznamy kódů produktů, tak se nejde do úrovně jednotlivých položek, ale vybírá se celý položkovník. Pokud je vybrán alespoň jeden dynamický nebo alespoň jeden statický položkovník, pak je možné pokračovat tlačítkem *Dokončit.*



Obr. 52. Druhý krok tvorby nákupního seznamu.

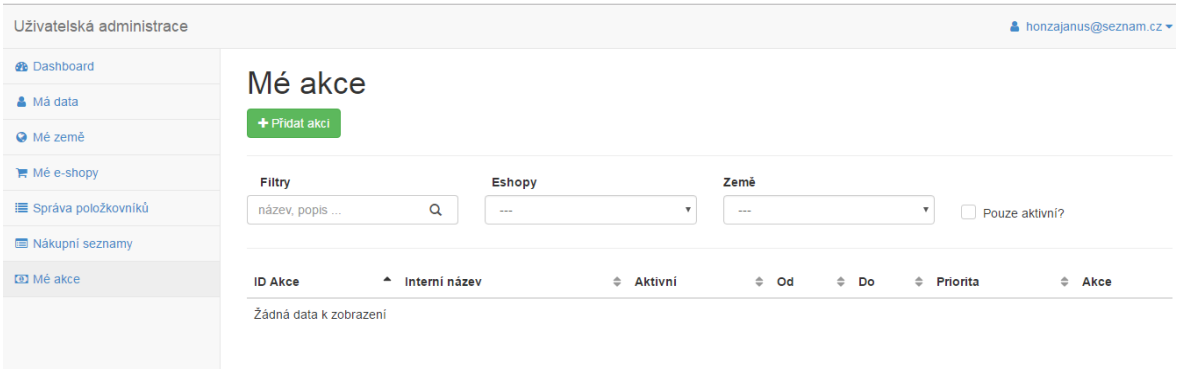
Po stisknutí tlačítka dochází k vytvoření nákupního seznamu, se kterým je dále možné pracovat v modulu *Mé akce* při tvorbě či editaci konkrétní akce. Ve vytvořených nákupních seznamech je možné využít hledání řešeného technologií ajax. Samozřejmostí je také editace či smazání konkrétního nákupního seznamu viz níže uvedený obrázek (*Obr. 53*).



Obr. 53. Zobrazení vytvořeného nákupního seznamu.

9.8 Mé akce

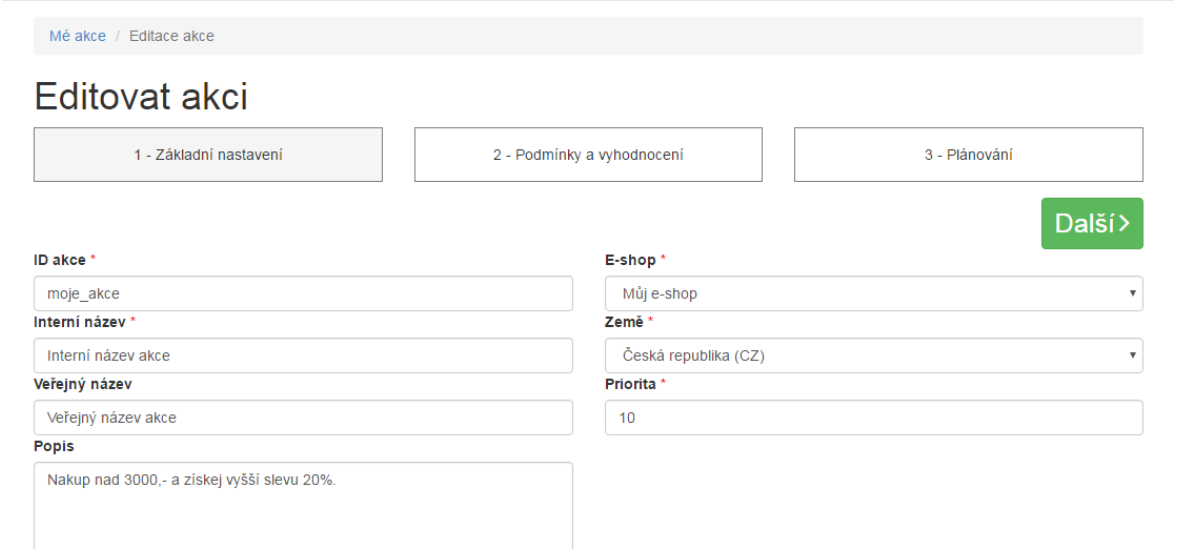
Modul *Mé akce* slouží k tvorbě a správě všech vytvořených slevových akcí, které může uživatel nastavit pro svůj e-shop a konkrétní zemi, ve které svůj e-shop provozuje. To znamená, že mohou být nastaveny různé akce pro různé země a e-shopy. Níže uvedený obrázek (*Obr. 54*) znázorňuje zobrazení modulu *Mé akce*.



Obr. 54. Zobrazení modulu Mé akce.

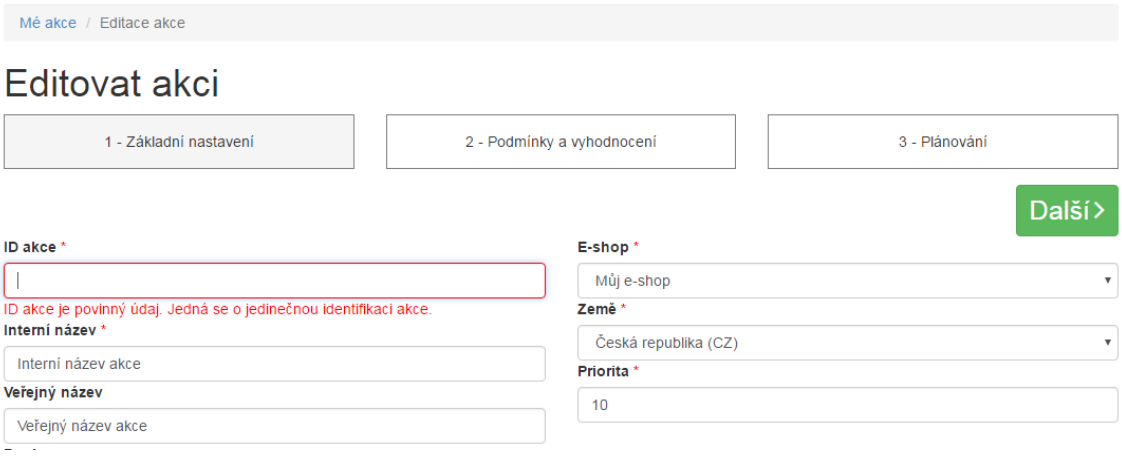
9.8.1 První krok

Kliknutím na tlačítko *Přidat akci* spustíme událost pro vytvoření akce a systém zobrazí první z kroků řízeného průvodce pro tvorbu slevové akce, viz níže uvedený obrázek (*Obr. 55*). Vyplněním všech polí označených jako povinné (*) splníme podmínku pro pokračování do dalšího kroku. Aplikace si hlídá, zda jsou opravdu všechna pole korektně vyplněna. Především je také realizována validace na existující prioritu v rámci kombinace e-shop a vybraná země, jelikož se jedná o parametr, který je vyhodnocován algoritmem uplatnění akcí a nemůže ve stávajícím stavu dojít k situaci, kdy by na jeden košík bylo uplatněno více slevových akcí najednou.



Obr. 55. První krok průvodce pro založení akce.

Pokud uživatel zapomene vyplnit některý z údajů, tak je systémem upozorněn, že je nutné příslušný údaj vyplnit viz níže uvedený obrázek (*Obr. 56*).

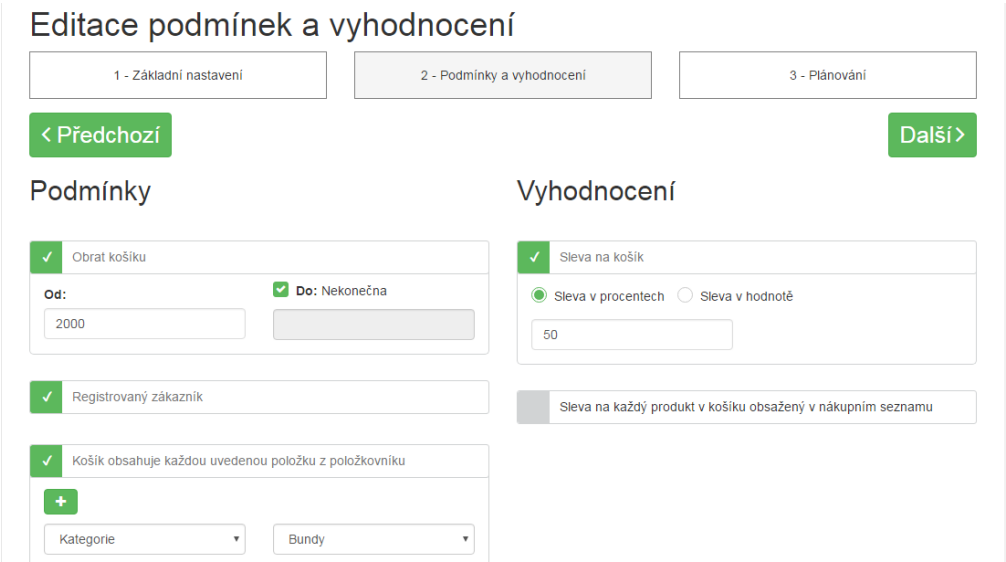


Obr. 56. Validace nevyplněných povinných polí.

Pokud jsou všechny údaje korektně vyplněny, pak lze pokračovat kliknutím na tlačítko *Další*.

9.8.2 Druhý krok

Systém umožní v dalším kroku řízeného průvodce nastavení podmínek a vyhodnocení akce viz níže uvedený obrázek (*Obr. 57*). Podmínky lze chápat jako pravidla, která musí být splněna, aby byla akce uplatněna. Naopak vyhodnocení lze chápat jako pravidla, která se uplatní v případě, že jsou splněny podmínky, nicméně vzhledem ke komplexnosti řešení se ve vyhodnocení také prolíná nastavení podmínek, a to zejména sleva, která je uplatněna na nákupní seznamy.



Obr. 57. Druhý krok průvodce pro založení akce.

Podmínky

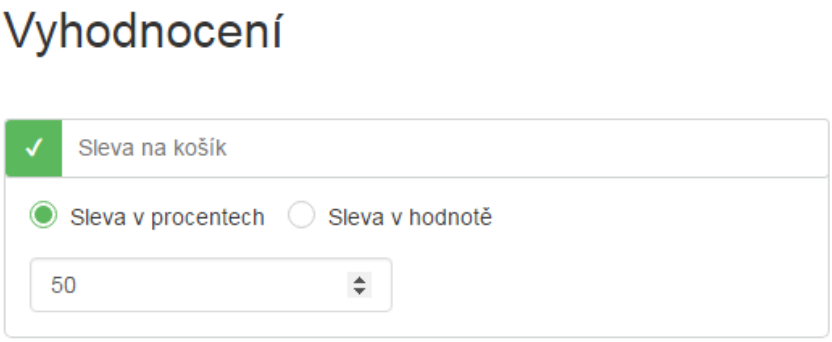
Systém aktuálně umožňuje práci se třemi různými typy podmínek. Jedná se o obrat košíku, registrovaného zákazníka a podmínku typu košík obsahuje každou uvedenou položku z položkovníku.

- **Obrat košíku** - umožní uživateli nastavit jakého obratu musí zákazník dosáhnout, aby byla sleva uplatněna. Je možné využít jak intervalového nastavení, tak je možné řešit nastavení nákupu nad určitou částku. Pro tento případ je k dispozici zaškrtávátko *Do nekonečna*.
- **Registrovaný zákazník** - umožňuje nastavit podmínku, zda se akce vztahuje pouze na registrovaného zákazníka nebo je aktivní i pro neregistrované zákazníky.
- **Košík obsahuje každou uvedenou položku z položkovníku** - umožňuje identifikovat skupiny zboží, které musí být vloženy do košíku, aby byla splněna podmínka. Je taktéž možné přidávat neomezeně počet řádků, kdy mezi jednotlivými řádky je logika AND.

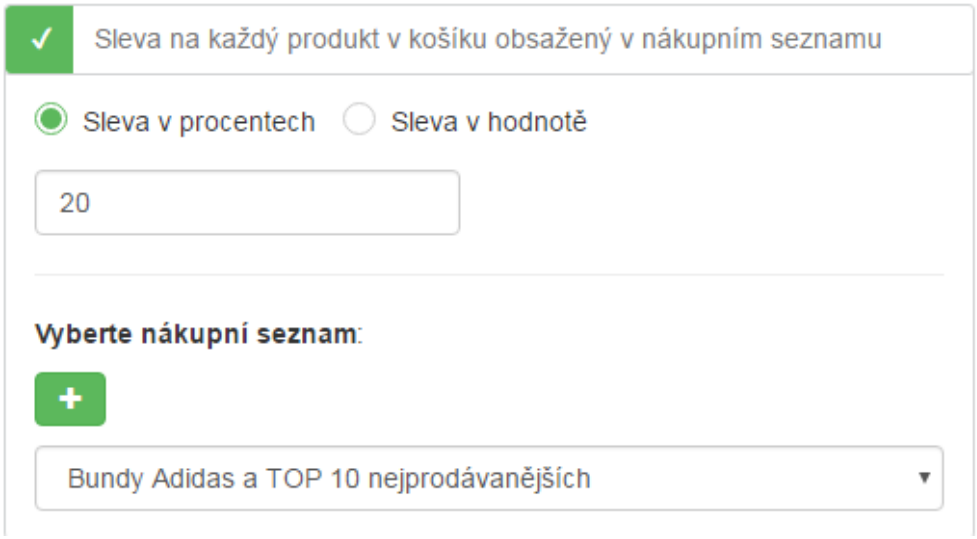
Vyhodnocení

Systém aktuálně umožňuje práci se dvěma různými typy vyhodnocení, které dále umožňují dílčí nastavení. Aktuálně dostupná vyhodnocení jsou sleva na košík, kterou lze definovat v procentech nebo jako hodnotovou. Dále je k dispozici nastavení slevy na každý produkt v košíku obsažený v nastaveném nákupním seznamu.

- **Sleva na košík** - umožní po splnění podmínek akce uplatnit definovanou slevu na celý košík. Níže uvedený obrázek (*Obr. 58*) znázorňuje nastavení tohoto vyhodnocení.
- **Sleva na každý produkt v košíku obsažený v nákupním seznamu –** umožní po splnění podmínek uplatnit definovanou slevu pouze na množinu zboží, která je zahrnuta v nákupním seznamu. V případě, že zboží, které je obsaženo v nákupním seznamu, není zrovna vloženo do košíku a jsou splněny podmínky, pak k uplatnění nedojde. Níže uvedený obrázek (*Obr. 59*) znázorňuje nastavení tohoto vyhodnocení.



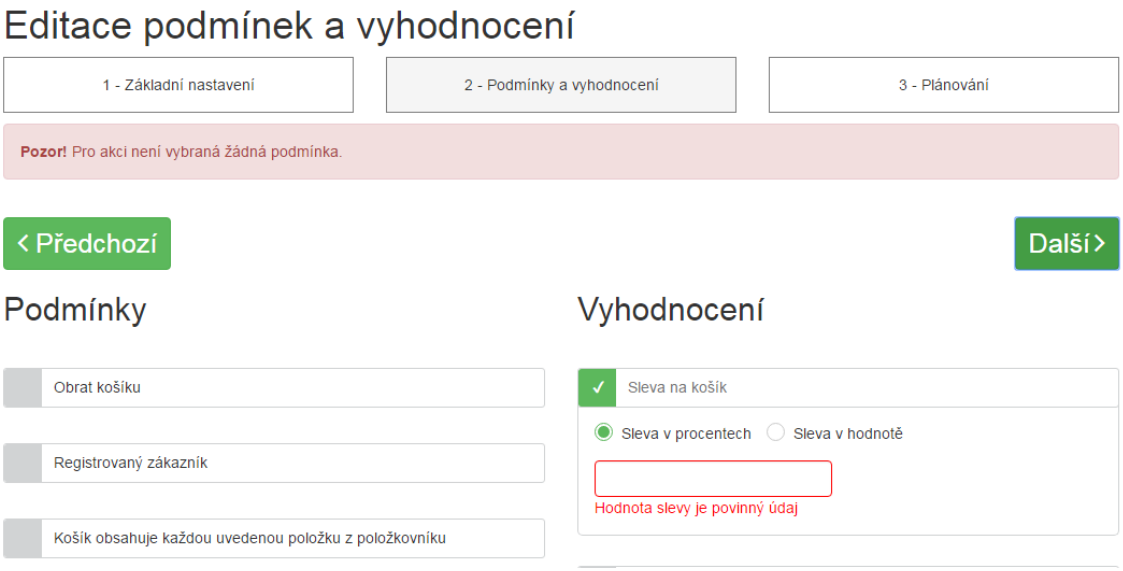
Obr. 58. Zobrazení vyhodnocení Sleva na košík.



Obr. 59. Zobrazení vyhodnocení Sleva na každý produkt v košíku obsažený v nákupním seznamu.

Nastavení podmínek a vyhodnocení

Pro pokračování do posledního kroku je opět nutné splnit veškeré předpoklady, a to mít zaškrtnutou minimálně jednu podmínku a vždy jedno vyhodnocení. Dále je nutné mít vyplněny příslušné hodnoty v zaškrtnutých záložkách. Aplikace opět počítá s kontrolou na zadávané hodnoty, tedy nedovolí zadat cokoliv jiného než kladné číslo a v případě intervalu, taktéž kontroluje, zda není hodnota *Do* menší než hodnota *Od*. Taktéž kontroluje opačný stav. Pokud je vše korektně vyplněno, pak lze pokračovat do dalšího kroku, jinak je uživatel upozorněn zřetelným chybovým hlášením viz *Obr. 60* níže.

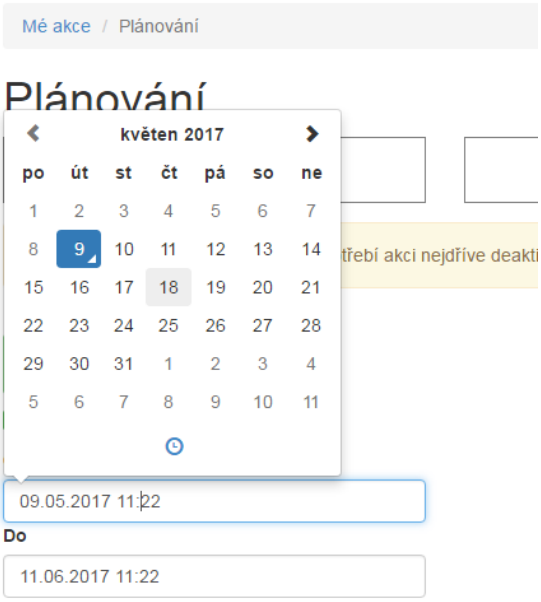


Obr. 60. Validace chybně vyplněných polí ve druhém kroku průvodce založení slevy.

9.8.3 Třetí krok

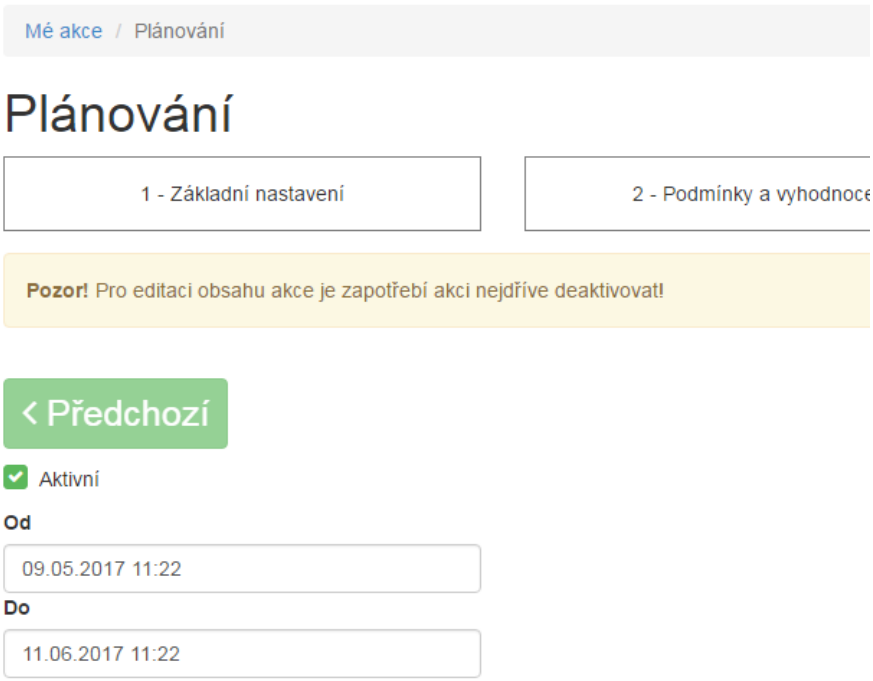
Třetí krok, který je posledním krokem pro vytvoření akce slouží k naplánování platnosti. Je možné nastavit datum *Od*, které určuje, kdy začíná akce platit. Stejně tak je potřeba nastavit koncové datum platnosti *Do*. Obě možnosti umožňují nastavení jak na úrovni data, tak na úrovni času. Dalším důležitým prvkem je zaškrtávátko *Aktivní*, které slouží pro aktivaci či deaktivaci dané akce. Toto je komfortní především pro uživatele, který může akci kdykoliv jednoduše deaktivovat či aktivovat. Nastavení času znázorňuje níže uvedený obrázek (*Obr. 61*) a nastavení data pak obrázek (*Obr. 62*) pod ním.

Obr. 61. Nastavení času platnosti.



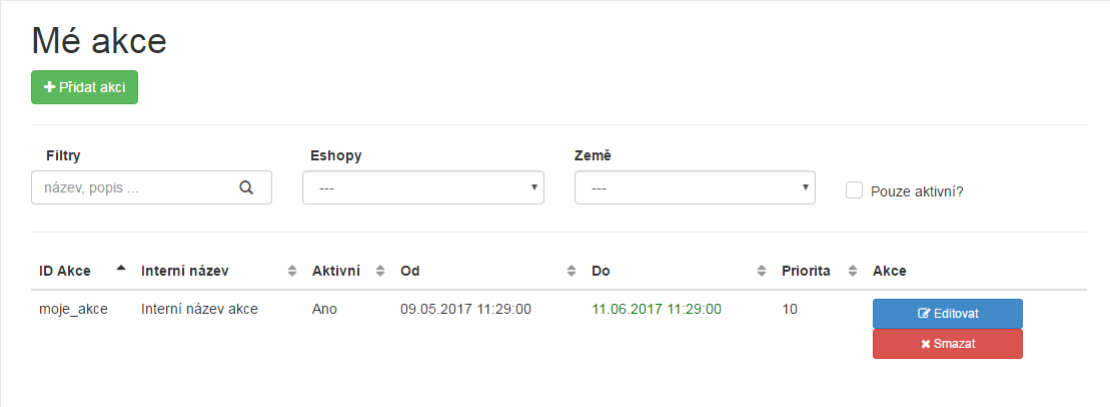
Obr. 62. Nastavení data platnosti.

Ve chvíli, kdy je akce aktivována, tak je zobrazeno upozorňující hlášení informující o nemožnosti akci editovat do doby, než dojde k deaktivaci, viz *Obr. 63* níže. Funkcionalita je k dispozici z toho důvodu, že při procházení jednotlivými kroky je vždy část akce uložena do databáze, tak aby uživatel neztratil vyplněná data a mohl případně nastavení akce pouze rozpracovat a dokončit později, tudíž není záhodno provádět editaci obsahu nad akcí, která je spuštěna v produkčním režimu.



Obr. 63. Aktivace akce.

Jakmile jsou vyplněny obě pole tj. pole *Od* a *Do* při aktivní akci, pak je možné akci dokončit jako aktivní. Za předpokladu, že je akce deaktivována, pak není nutné vyplnit obě pole s datem a je možné akci dokončit rozpracovanou. Po dokončení založení akce se zobrazí akce v přehledu založených akcí, viz níže uvedený obrázek (*Obr. 64*). Pokud máme založeny desítky akcí, pak je možné použít rozšířené filtrace včetně hledání, které hledá v názvu a popisu akce. Ve filtrech lze použít filtraci dle země, e-shopu nebo lze filtrovat pouze aktivní akce. Samozřejmostí je kombinace jednotlivých filtrů. Veškerá filtrace je realizována technologií ajax, tudíž není uživatel vyrušen novým načtením HTML dokumentu. Neméně důležitou funkcí z uživatelského hlediska je také možnost řazení dle jednotlivých sloupců, které je realizováno pomocí šipek u každého sloupce a je možné řadit vzestupně nebo sestupně.



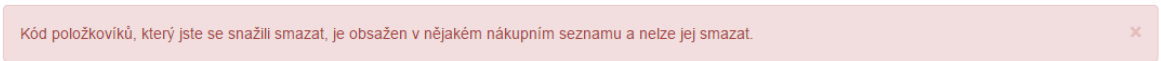
Obr. 64. Zobrazení založené akce.

9.8.4 Rozšířené validace

Aplikace je navrhnuta tak, aby dokázala zamezit jakýmkoliv nekonzistentním stavům. Jsou realizovány validace, jak na databázové úrovni, tak na úrovni aplikační.

Validace na relace

Pokud existuje relace mezi dvěma entitami, pak nemůže dojít ke smazání jedné nebo druhé entity. Uživatel je schopen čehokoliv, tedy aplikace musí myslet za něj, co lze povolit a co už nikoliv. V tomto případě se jedná především o kontroly probíhající při mazání záznamů. Před smazáním vždy dojde ke kontrole, zda neexistuje relace na daný záznam, který se uživatel snaží smazat. Pokud taková relace existuje, tak je uživateli zobrazeno chybové hlášení, které informuje, že je nejdříve nutné tuto relaci zrušit, aby bylo možné daný záznam smazat. Znázornění chybového hlášení lze vidět na níže uvedeném obrázku (*Obr. 65*) a obrázku (*Obr. 66*) pod ním.



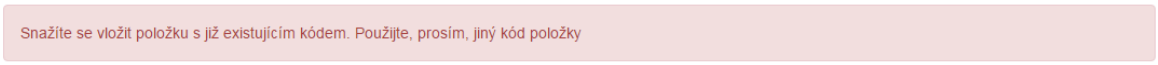
Obr. 65. Zobrazení chybového hlášení informujícího o existující relaci.

Obr. 66. Zobrazení chybového hlášení informujícího o existující relaci.

Validace na duplicitní hodnoty

Vzhledem k použití kódů, které jednoznačně specifikují danou entitu, bylo taktéž nutné použít validaci na duplicitní hodnoty. Systém při snaze uložit zadávanou hodnotu zkontroluje,

zda již neexistuje v databázi a pokud ano, pak zobrazí uživateli chybové hlášení, které jej o této skutečnosti informuje, viz níže uvedený obrázek (*Obr. 67*).



Obr. 67. Zobrazení chybového hlášení informujícího o duplicitě.

10 WEBOVÁ APLIKACE API

Návrh API byl jedním z nejsložitějších úkonů tohoto projektu, a to zejména z toho důvodu, že se jedná o rozhraní, které prezentuje aplikaci navenek a musí umožnit jednoduchou a dostatečně rychlou komunikaci mezi klientem a serverem. Vzhledem k charakteru navrženého API je možné také do budoucna počítat s možností použití load balanceru s distribucí zátěže na více aplikací API hostovaných na různých serverových řešeních.

10.1 Forma komunikace

Nejdříve se bylo potřeba zamyslet, v jakém formátu bude API komunikovat. Po pečlivém promyšlení všech pro a proti byl vybrán formát JSON, který umožňuje přenášet minimalizovanou informaci v jednoduše zpracovatelné a čitelné formě. Dále se bylo potřeba zamyslet nad tím, jakým způsobem bude API interpretována veřejnosti zejména z hlediska URL adresy a dále způsobu volání metod. Byla zvolena strategie, kdy URL adresa obsahuje identifikaci domény, verze API a unikátního API klíče, který identifikuje konkrétní e-shop daného uživatele uvnitř cloudového řešení. Volání jednotlivých metod API není řešeno formou URL adresy, ta zůstává stále neměnná, nicméně řeší se uvedením názvu volané metody do kolekce hodnot v zaslaném JSON dotazu.

10.1.1 Dotaz

V rámci návrhu formy dotazu byla neustále na mysli představa košíku e-shopu, do kterého zákazník vkládá produkty různých druhů, je schopen měnit množství a vidí příslušné informace o ceně za celý košík i jednotlivé položky, které jsou do něj vloženy. Z této základní myšlenky vychází návrh dotazu na API, který byl dále obohacen o přídatné parametry, které vycházejí z logiky slevového portálu. Technicky dotaz počítá s tím, že nedochází nikdy k vynechávání parametrů, zasílá se vždy v kompletní formě. Toto bylo navrženo zejména z důvodu jednotné formy dotazu a stejně tak je navržena i forma odpovědi. Pokud je některý parametr v dotazu vynechán, pak je API připravena tuto záležitost klientovi sdělit formou validační funkce, která vrátí informaci o nesprávně formě dotazu s konkrétním prvkem, který je nevyplněn nebo zcela chybí. Ukázku plné verze dotazu lze vidět na níže uvedeném obrázku (*Obr. 68*).

```
{
    "currency_iso": "CZK",
    "rounding_method": "none",
    "data": {
        "eshop": {
            "country_iso": "CZ"
        },
        "customer": {
            "is_signed": false
        },
        "cart": {
            "items": [
                {
                    "product_code": "ABCDEFG",
                    "quantity": 3,
                    "unit_sales_price": 1000,
                    "general_lists": [
                        {
                            "general_list_code": "category",
                            "general_list_item_code": "bundy"
                        },
                        {
                            "general_list_code": "brand",
                            "general_list_item_code": "nike"
                        }
                    ]
                }
            ]
        }
    },
    "created": "2017-05-09T18:26:00Z",
    "method_name": "cart"
}
```

Obr. 68. JSON dotaz na metodu cart.

Pokud se zaměříme detailně na strukturu dotazu, tak zjistíme, že je rozdělen do dvou základních logických částí, a to je část s externími parametry a tělo (data). Část s externími parametry lze chápat jako část, ve které jsou uvedeny parametry, které slouží pro nastavení chování API k dynamickým datům a ovlivňují průtok dotazu vyhodnocovacím algoritmem na straně serveru. V rámci těla dotazu jsou specifikovány jednotlivé objekty *eshop*, *customer*, *cart*, které jsou do budoucna připraveny na sdružování souvisejících kolekcí hodnot, tedy objekt *customer* bude sdružovat všechny informace o zákazníkovi apod. Objekt *cart* obsahuje informace o košíku, se kterým aktuálně zákazník pracuje. Pole *items* následně interpretuje výčet produktů v konkrétním množství, ceně a s konkrétními parametry produktů, které jsou definovány v rámci pole *general_lists*. Detailní popis parametrů v dotazu popisuje níže uvedená tabulka (*Tab. 1*).

Parametr (Kolekce)	**Popis**
Created	Čas vytvoření dotazu dle mezinárodního standardu UTC.
method_name	Slouží k vložení názvu volané metody. Dle volané metody dochází také k validaci schématu dotazu. Aktuálně k dispozici pouze metoda *cart*.
currency_iso	ISO kód dle mezinárodního standardu ISO 4217 slouží pro práci s korektním počtem desetinných míst, zejména při zaokrouhlování.
rounding_method	Slouží k volbě metody zaokrouhlení vypočtených částek. Aktuálně k dispozici *units_half_round_up* a *none*.
country_iso	ISO kód dle mezinárodního standardu ISO 3166-1 alpha-2. Identifikuje zemi (jazykovou verzi e-shopu), ve které zákazník nakupuje. Musí odpovídat s nastavenou zemí ve slevovém portálu.
is_signed	Slouží k identifikaci události přihlášení zákazníka do e-shopu resp. k indentifikaci registrovaného zákazníka. Může nabývat hodnoty *true* (registrovaný zák.) nebo *false* (neregistrovaný zák.).
product_code	Slouží k identifikaci kódu produktu. Tento parametr souvisí s nastavením statických položkovníků v aplikaci slevového portálu.
quantity	Slouží k definici aktuálně vloženého množství do košíku pro daný produkt.

unit_sales_price	Jedná se o jednotkovou prodejní cenu daného produktu.
general_list_code	Kód položkovníku.
general_list_item_code	Kód položky obsažené v položkovníku.

Tab. 1. Popis parametrů JSON dotazu na metodu cart.

10.1.2 Odpověď

Forma odpovědi byla navržena speciálně tak, aby pokryla obsahem veškeré výpočty, tudíž klient nebude potřebovat nikdy nic počítat a pouze přebere příslušné parametry, které se mu budou hodit. Odpověď se může lišit v závislosti na tom, zda je uplatněna akce nebo není. Signalizaci, zda je akce uplatněna či nikoliv provádí parametr *was_action_applied*, který může nabývat hodnoty *true* nebo *false*. Tento způsob byl zvolen pro jednoznačnou identifikaci, zda jako klient mám začít pracovat s vrácenou odpovědí nebo ji mám ignorovat. Pokud by parametr neexistoval, pak by bylo možné identifikaci provést pouze na základě nulových hodnot v parametrech definujících výši slevy, což nebylo považováno za elegantní řešení. Odpověď se dělí do tří základních logických celků, a to externích parametrů, parametrů uplatněné akce a těla (cart). Ukázku odpovědi lze vidět znázorněnou na níže uvedeném obrázku (*Obr. 69*).

```
{
    "was_action_applied": true,
    "action": {
        "action_id": "moje_akce",
        "public_name": "Veřejný název akce",
        "description": "Nakup nad 3000,- a získej vyšší slevu 20%.",
        "cart": {
            "total_calculated_price": 2700,
            "total_calculated_discount": 300,
            "total_calculated_price_rounded": 2700,
            "rounding_difference": 0,
            "items": [
                {
                    "product_code": "ABCDEFG",
                    "unit_calculated_price": 900,
                    "total_item_calculated_price": 2700,
                    "unit_calculated_discount": 100,
                    "total_item_calculated_discount": 300
                }
            ]
        }
    },
    "created": "2017-05-09T16:35:25Z"
}
```


Obr. 69. JSON odpověď metody cart.

První z těchto celků tj. externí parametry je navržen tak, aby obsahoval parametry, které se nevztahují přímo k uplatněné akci, ale k odpovědi jako takové. Parametry akce, jakožto druhý celek obsahuje již parametry, které se vztahují přímo k uplatněné akci, kterou nastavil marketingový specialista ve slevovém systému. Poslední část, která je obsahem objektu *cart* již obsahuje výpočty vykonané v závislosti na vstupních datech zaslaných v dotazu a na uplatněné akci. Detailní popis parametrů v odpovědi popisuje níže uvedená tabulka (*Tab. 2*).

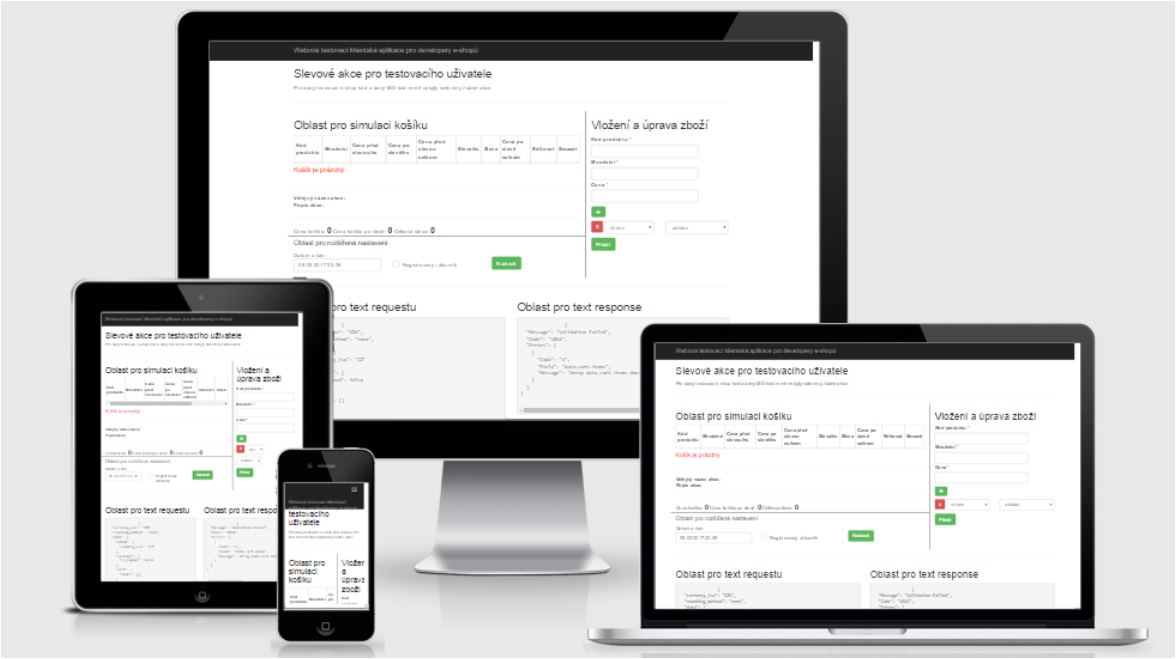
Parametr (Kolekce)	**Popis**
created	Čas odpovědi na dotaz dle mezinárodního standardu UTC.
was_action_applied	Slouží k jednoznačné identifikaci, zda došlo k uplatnění slevy či nikoliv. Může nabývat dvou hodnot *true* nebo *false*.
action_id	Slouží k jednoznačné identifikaci uplatněné akce.
public_name	Veřejné jméno uplatněné akce, které může e-shop zobrazit v košíku.

total_calculated_price	Celková cena košíku po slevách.
total_calculated_discount	Celková sleva na košík.
total_calculated_price_rounded	Celková zaokrouhlená cena košíku.
rounding_difference	Rozdíl zaokrouhlení.
product_code	Kód produktu pro jednoznačné spárování zlevněného zboží.
unit_calculated_price	Jednotková cena produktu po slevách.
total_item_calculated_price	Celková cena produktu po slevách.
unit_calculated_discount	Celková jednotková sleva na produkt.
total_item_calculated_discount	Celková sleva na produkt.

Tab. 2. Popis parametrů JSON odpovědi metody cart.

11 WEBOVÁ APLIKACE KLIENTSKÉHO SYSTÉMU

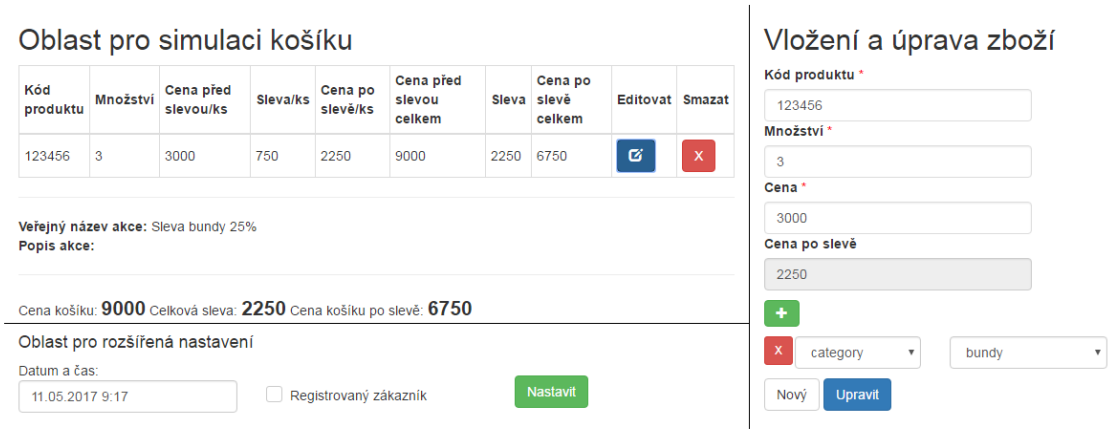
Tato aplikace byla vyvinuta jako poslední a byla na ní ověřena kompletní funkčnost vyvinutého API a taktéž provázanost se slevovým portálem. V etapě vývoje, kdy tato aplikace ještě neexistovala, byla využita aplikace třetí strany tj. Advanced REST Client pro simulaci komunikace s API a ladění v průběhu vývojového cyklu. Testovací klient je aplikace, která simuluje reálný košík e-shopu a umožňuje nastavovat parametry, které ovlivňují uplatnění či neuplatnění nastavených slevových akcí. Mimo to aplikace dotahuje aktivní slevové akce od účtu uživatele, který je nastaven v konfiguraci této klientské aplikace. Pokud dojde k uplatnění některé z akcí, pak je vizuálně znázorněno zelenou barvou, viz *Obr. 71* níže. Součástí aplikace je také možnost monitorování komunikace mezi košíkem a API, která probíhá po každé změně nastavení či položky v košíku. Zobrazení klientské aplikace na různých rozlišeních, znázorňuje níže uvedený obrázek (*Obr. 70*).



Obr. 70. Zobrazení klientské aplikace pro různá rozlišení.

Obr. 71. Zvýraznění uplatněné akce v klientské aplikaci.

Aplikace umožňuje vkládat a upravovat obsah košíku stejně jako v e-shopu. K vkládání zboží slouží pravý panel znázorněný na obrázku níže (*Obr. 72*), ve kterém je zapotřebí vyplnit povinné parametry produktu, bez kterých by produkt nebylo možné do košíku vložit. Zboží lze z košíku odstranit pomocí červeného tlačítka s křížkem nebo jej lze editovat, kdy v editačním módu se lze pomocí tlačítka *Nový* přepnout do módu vložení nového produktu. Košík také znázorňuje jednotlivé vypočtené částky a texty vztahující se k uplatněné akci. Součástí je také panel s rozšířeným nastavením, který umožňuje pomocí data pohyb na časové ose nebo možnost nastavení simulace registrovaného zákazníka. Na níže uvedeném obrázku (*Obr. 73*) je znázorněn dotaz a odpověď při komunikaci s API.



Obr. 72. Oblasti správy košíku v klientské aplikaci.

Obr. 73. Komunikace s API.

12 TESTOVÁNÍ

Testování aplikací bylo potřeba provádět po každém nově sestaveném buildu, tak aby bylo možné odstranit nalezené chyby, co nejrychleji. Testování probíhalo ve dvou úrovních, a to unit testy, které automatizovaně testují jednotlivé stavební bloky aplikace, především ty složitější, tudíž náchylnější k chybovosti. Druhou úrovní bylo manuální testování, které posloužilo zejména v rámci funkčních testů a integračního testování, tedy vzájemné synergie mezi vyvinutými aplikacemi.

12.1 Unit testy

Vývojář musí dbát na základní návyky, jedním z nich jsou unit testy, které jsou tvořeny již od začátku vývojového cyklu. Díky architektuře MVC je možné jednoduše testovat business logiku, kde našli jednotkové testy největší uplatnění. Na níže uvedeném obrázku (*Obr. 74*) probíhá testování výpočtu slev, tudíž v případě budoucích úprav výpočetního algoritmu je možné jednoduše otestovat, zda funguje výpočet stále stejně nebo nastala změna. Pokud nastala změna, pak bude test označen jako neprůchozí a my budeme mít signál k tomu, abychom upravený kód ještě jednou zkontrolovali a nalezli příčinu tohoto rozdílu.

```
[TestMethod]
public void ComputePercentageDiscountForItem()
{
    double itemPrice = 1000;
    int percentageDiscount = 15;

    double resultDiscount = CartDiscountComputer.ComputeNewPriceAfterPercentageDiscount(itemPrice, percentageDiscount);

    Assert.AreEqual(850, resultDiscount);
}

[TestMethod]
public void ComputePercentageValueBasedOnCartPriceAndDiscount()
{
    double cartTotalPrice = 4000;
    int valueDiscount = 150;

    double cartValueAfterDiscount = CartDiscountComputer.ComputePercentValueBasedOnCartoTotalPriceAndValueDiscount(cartTotalPrice, valueDiscount);

    Assert.AreEqual(3.75, cartValueAfterDiscount);
}
```

Obr. 74. Testování výpočtu slev.

12.2 Funkční a integrační testování

V rámci funkčních a integračních testů se podařilo odhalit několik chyb, a to zejména chyby související s návratovou odpovědí API. Jedna z nich, která nebyla v průběhu vývoje v aplikaci podchycena bylo uplatnění slevy do mínusových hodnot. Košík se následně tvářil, že je jeho celková cena v mínusu, což bylo následně ošetřeno metodou, která se volá nad jednotlivými proměnnými, které vracejí hodnoty do kolekcí v JSON odpovědi, viz *Obr. 75* níže.

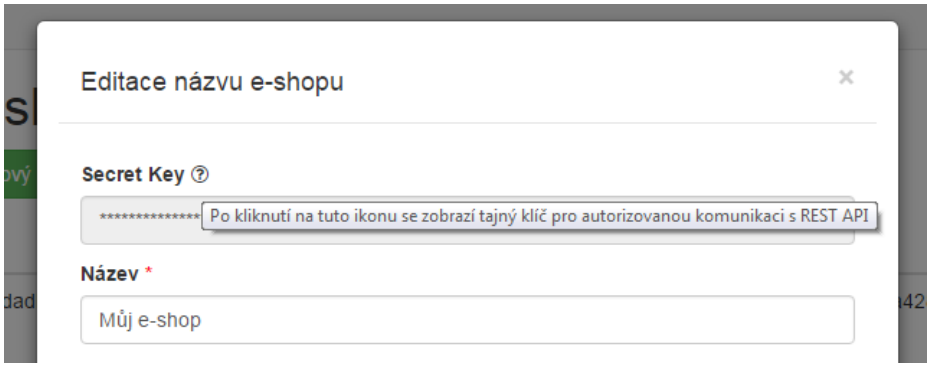
```
public static double ReturnZeroIfLowerThanZero(double value)
{
    if (value < 0) return 0;
    else return value;
}
```

Obr. 75. Metoda řešící chybu návratu mínusových hodnot.

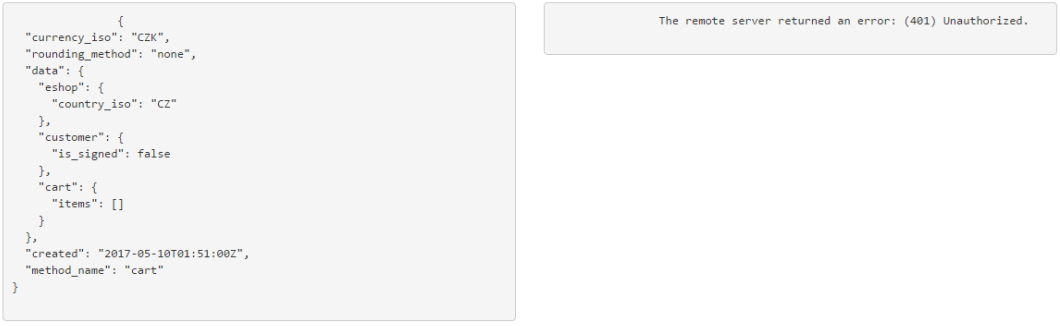
13 ZABEZPEČENÍ SLEVOVÉHO SYSTÉMU

Zabezpečení slevového systému jako celku byla věnována rovněž pozornost. Po předchozích zkušenostech s vývojem webových aplikací byl vypnut HTML atribut autocomplete na přihlašovacích i registračních polích. Tento atribut standardně zajišťuje automatické vyplňování pro jednotlivé inputy.

Každá URL adresa API je svým způsobem unikátní, protože se liší API klíčem, který je ekvivalentní ke kódu e-shopu, tudíž není možné odvolat API bez nutné znalosti kódu e-shopu daného uživatele. Jako další prvek zabezpečení byla implementována HTTP Basic Autentizace, což v praxi znamená to, že pokud API obdrží požadavek od anonymního zdroje, pak je vrácen chybový HTTP status kód 401. HTTP autentizace využívá tajného klíče, který je vygenerován hashovou funkcí po založení e-shopu viz *Obr. 76* níže a ve výchozím stavu je nečitelný. Lze jej přečíst pouze po nutné interakci uživatele. Neprůchozí dotaz lze vidět na níže uvedeném obrázku (*Obr. 77*).



Obr. 76. Tajný klíč pro zabezpečenou komunikaci.



Obr. 77. Neprůchozí HTTP autentizace.

ZÁVĚR

Cílem této diplomové práce bylo vyvinout nový produkt, který bude možné představit široké veřejnosti. Tento produkt je specifický svým zařazením, jelikož dle studie neexistuje mnoho řešení, které se touto problematikou zabývají. Základní myšlenkou bylo vytvořit unikátní produkt, který zaujme svou robustní funkcionalitou a který bude možné do budoucna neomezeně vylepšovat. V začátcích nebyly jasné detaily, které pomohla specifikovat část analýzy a mapování, tudíž bylo možné na základě rozkrytých souvislostí navrhnout architekturu výsledného produktu. Celkový postup byl takový, že se nejdříve nakreslila představa jednotlivých modulů na papír, tak aby bylo možné si vizualizovat jednotlivé moduly, dále pole v jednotlivých modulech. Na základě náčrtku došlo k návrhu databázového schématu včetně definice datových typů pro jednotlivá pole v tabulkách, dle jejich funkčního charakteru, k tomuto výrazně dopomohl Entity Framework, díky kterému nebylo nutné psát všechny T-SQL skripty ručně, ale vygenerovala je aplikace na základě sestavení modelu. Některé z T-SQL skriptů bylo přesto potřeba napsat ručně, tudíž byl využit systém databázových migrací, které obsluhuje metoda *Seed* a jejichž spuštění lze vyvolat pomocí *Package Manager Console*.

V průběhu implementace došlo k několika zásadním zvratům, které zapříčinily postupný vývoj datového modelu, který na začátku figuroval jako součást projektu slevového portálu až postupně se z něho vyvinula samostatná datová struktura, která je nyní sdílena mezi všemi aplikacemi. Výsledkem je eliminace přebytku duplicitního kódu. Architektura jako celek byla komplexní záležitostí a vůbec jednou z nejsložitějších částí této práce. Naprosto zásadní je nezávislost business logiky slevového portálu na API, tudíž při budoucích úpravách nedojde k porušení již funkčních částí v obou aplikacích najednou. Pokud dojde k porušení funkční části, pak je možné odhalit eventuální příčinu pomocí unit testů v konkrétním projektu.

Výsledná aplikace nabízí uživateli možnost registrace do slevového portálu. Ve chvíli, kdy je vytvořen nový aktivní účet, pak je možné se přihlásit a začít využívat naplno funkcí slevového portálu jakožto cloudové aplikace. Při registraci byl brán ohled na soukromí uživatele, jsou vyžadovány pouze nezbytně nutné údaje. Přihlášený uživatel může začít nastavovat prostředí svého účtu přesně podle jeho potřeb, tedy pro začátek spravovat své e-shopy a země, ve kterých dané e-shopy působí. Uživatel má následně možnost vytvářet položkov-

níky a nákupní seznamy, které byly navrhnuty a implementovány z důvodu zavedení možnosti přesného zaměření slev na konkrétní množiny produktů. Díky úzkému provázání virtuálních množin zboží s modulem slevových akcí je možné spouštět automatizovaně rozmanité slevové akce, které ovlivňují různé části e-shopem prodávaného sortimentu. Pomocí několika kliknutí je možné nastavit slevovou akci s konkrétní dobou platnosti a spustit ji pro produkční režim, uživatel se nestará o nic jiného než o nastavení potřebných akcí a jejich aktivaci, vše ostatní systém zajistí. Výsledkem je, že se na e-shopu ihned po aktivaci akce začne projevovat její vyhodnocení, tedy pouze za předpokladu, že zákazníkův košík splňuje podmínky pro uplatnění. Slevové akce je možné naplánovat dopředu a systém sám zajistí jejich automatické spuštění dle nastavené doby spuštění a zároveň slevovou akci sám ukončí v závislosti na termínu ukončení. V neposlední řadě lze zmínit, že správu může uživatel provádět na jakémkoliv zařízení, protože je celá aplikace navržená v technologií responzivního designu.

Projekt byl zakončen integračními a funkčními testy aplikace jako celku, které dopomohly odhalit skryté vady, aby následně mohlo dojít k nápravě. Vývoj na systému však stále není zastaven a dochází k vývoji nových funkcí, které budou součástí základního balíčku pro prezentaci na trhu nových IT technologií v České republice.

Celkově projekt výrazně zdokonalil mé zkušenosti v oblasti architektonického návrhu aplikací a posunul mé myšlení v oblasti synergie IT technologií a marketingových technik diametrálně kupředu. Každého vývojáře těší, když může vyvíjet něco nového, co zanechá nějakou stopu, na rozdíl od přebírání již existujících aplikací, které již někdo před ním vymyslel. K tomuto přispívá i fakt, že výsledná aplikace nemá dle aktuálně dostupných informací celosvětově konkurenci, tedy myšlenka univerzální slevové aplikace je zcela unikátní.

SEZNAM POUŽITÉ LITERATURY

[1] *Kentico E-commerce* [online]. [cit. 2017-05-11]. Dostupné z: http://www.kentico.com/cs/produkt/e-commerceg.

[2] Magento: How To Apply Shopping Cart / Checkout Discounts. *Shero* [online]. 2014 [cit. 2017-05-11]. Dostupné z: https://www.sherodesigns.com/magento-how-to-apply-shopping-cartcheckout-discounts/

[3] UGURLU, Tugberk, Alexander ZEITLER a Ali KHEYROLLAHI. *Pro ASP.NET Web API: HTTP Web services in ASP.NET* [online]. Apress, 2013 [cit. 2017-05-11]. ISBN 978-1-4302-4726-5.

[4] CROFT, Jeff, Ian LLOYD a Dan RUBIN. Mistrovství v CSS: Pokročilé techniky pro webové designéry a vývojáře. Vydání první. Holandská 8, 639 00 Brno: Computer Press, a.s., 2007. ISBN 978-80-251-1705-7.

[5] HOWE, Shai. *Learn to code HTML & CSS: Develop & Style websites* [online]. New riders, 2014 [cit. 2017-05-11]. ISBN 978-0-321-94052-0.

[6] ZAKAS, Nicholas C. *Professional JavaScript for Web Developers* [online]. Wrox, 2005 [cit. 2017-05-11]. ISBN 978-1118026694.

[7] FREEMAN, Adam. *Pro jQuery 2.0* [online]. 2. Apress, 2013 [cit. 2017-05-11]. ISBN 978-1-4302-6389-0.

[8] Unable to understand useCapture attribute in addEventListener. *Stackoverflow* [online]. 2016 [cit. 2017-05-11]. Dostupné z: http://stackoverflow.com/questions/7398290/unable-to-understand-usecapture-attribute-in-addeventlistener.

[9] JSON Basics: What You Need to Know. *JSON Basics: What You Need to Know* [online]. 2011 [cit. 2017-05-11]. Dostupné z: http://www.elated.com/articles/json-basics/.

[10] Úvod do JSON. *Úvod do JSON* [online]. [cit. 2017-05-11]. Dostupné z: http://www.json.org/json-cz.html.

[11] EMMATTY, John T. Differences between MVC and MVP for Beginners. *Codeproject.com* [online]. 2011 [cit. 2017-05-11]. Dostupné z: https://www.codeproject.com/Articles/288928/Differences-between-MVC-and-MVP-for-Beginners.

[12] ZEESHAN, Afzaal Ahmad. Understanding ASP.NET MVC using real world example, for beginners and intermediate. *Codeproject.com* [online]. 2015 [cit.

2017-05-11]. Dostupné z: https://www.codeproject.com/Articles/871375/Understanding-ASP-NET-MVC-using-real-world-example.

[13] Framework definition. *Whatis.com* [online]. [cit. 2017-05-11]. Dostupné z: http://whatis.techtarget.com/definition/framework.

[14] MACDONALD, Mathew, Adam FREEMAN a Mario SZPUSZTA. *ASP.NET 4 a C# 2010: Tvorba dynamických stránek profesionálně*. 2. Brno: Zoner Press, 2011. ISBN 978-80-7413-145-5.

[15] IQBAL CHOWDHURY/GALIB, Shahriar. Why(s) & How(s) of Asp.Net MVC Part 1. *Codeproject.com* [online]. 2013 [cit. 2017-04-27]. Dostupné z: https://www.codeproject.com/Articles/552846/Why-s-How-s-of-Asp-Net-MVC-Part.

[16] LERMAN, Julia. *Programming Entity Framework: Building Data Centric Apps with the ADO.NET Entity Framework* [online]. 1. O'REILLY, 2009 [cit. 2017-05-11]. ISBN 978-0596520281.

[17] FORBES, Alan. *The Joy of Bootstrap: A smarter way to learn the world's most popular web framework* [online]. 2. CreateSpace Independent Publishing Platform, 2015 [cit. 2017-05-11]. ISBN 978-1522792819.

[18] PRABHU, Anirudh. *Beginnning CSS Preprocessors: With Sass, Compass, and Less* [online]. 1. Apress, 2015 [cit. 2017-05-11]. ISBN 978-1-4842-1347-6.

[19] ERL, Thomas. *Cloud Computing Design Patterns* [online]. The Prentice Hall Service Technology Series from Thomas Erl. Prentice Hall, 2015 [cit. 2017-05-11]. ISBN 978-0133858563.

[20] Dynamic Scalability Architecture. *Informit.com* [online]. 2013 [cit. 2017-05-11]. Dostupné z: http://www.informit.com/articles/article.aspx?p=2093407&seqNum=3.

[21] INSTALLING, CONFIGURING, AND DEVELOPING WITH XAMPP. *Http://dalibor.dvorski.net* [online]. Ontario, 2007 [cit. 2017-05-11]. Dostupné z: http://dalibor.dvorski.net/downloads/docs/InstallingConfiguringDevelopingWithXAMPP.pdf.

[22] Web Server (IIS). *Technet.microsoft.com* [online]. 2009 [cit. 2017-05-11]. Dostupné z: https://technet.microsoft.com/en-us/library/1c64b5ab-d155-4654-b3d1-4f265749e3fc.

[23] SACK, Joseph. *Velká kniha T-SQL & SQL Server 2005 kompendium znalostí pro začátečníky i profesionály*. Brno: Zoner Press, 2007. ISBN 978-80-86815-57-2.

[24] JOHNSON, Bruce. *Professional Visual Studio 2015* [online]. 1. Wrox, 2015 [cit. 2017-05-11]. ISBN 978-1119068051.

[25] Enterprise Architect. *Http://www.sparxsystems.com/* [online]. [cit. 2017-05-11]. Dostupné z: http://www.sparxsystems.com/.

[26] Advanced REST client in API Testing. *Http://www.zyxware.com* [online]. 2016 [cit. 2017-05-11]. Dostupné z: http://www.zyxware.com/articles/5518/advanced-rest-client-in-api-testing.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

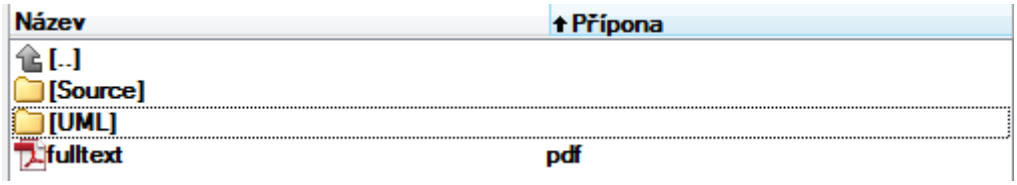
SAAS	Software As A Service.
CSS	Cascade Style Sheets
HTML	HyperText Markup Language
MVC	Model View Controller
LAN	Local Area Network
IDE	Integrated Development Environment
REST	Representational State Transfer
UI	User Interface
JSON	JavaScript Object Notation
AJAX	Asynchronous JavaScript and XML
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
RAM	Random Access Memory
IT	Information Technology
GB	GigaBajt

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: ADRESÁŘOVÁ STRUKTURA NA CD



- **Source** - Uvedený adresář obsahuje zdrojové kódy.
- **UML -** Uvedený adresář obsahuje projekt s UML diagramy.

PŘÍLOHA P II: PŘÍPADY UŽITÍ

UC001 - Registrace uživatele do systému

Vstupní podmínky

Uživatel má ve webovém prohlížeči zobrazenou webovou stránku slevového portálu a má k dispozici odkaz pro registraci.

Hlavní úspěšný scénář

1. Uživatel klikne na odkaz pro registraci nového účtu.

2. Systém zobrazí formulář s poli Jméno, Příjmení, Telefonní číslo, E-mail, Heslo a Potvrzení hesla a tlačítko pro potvrzení zadaných údajů ve formuláři. Systém zobrazí také odkaz pro na stránku pro přihlášení.

3. Uživatel vyplní pole požadovanými údaji a kliká na tlačítko pro potvrzení registrace.

4. Systém kontroluje, zda heslo odpovídá bezpečnému formátu hesla, to znamená, že obsahuje minimálně 10 znaků, obsahuje malé i velké písmeno, číslici a speciální znak např. *.

5. Systém zjistí, že je heslo dostatečně bezpečné.

6. Systém kontroluje, zda již zadaný e-mail neexistuje v databázi.

7. Systém zjistí, že zadaný e-mail není registrován v databázi a zakládá uživatelský účet do databáze. Systém účet označí jako neaktivní a odesílá aktivační odkaz na uživatelem zadaný e-mail. Systém zobrazí uživateli informační hlášení o zaslání aktivačního e-mailu.

8. Uživatel aktivuje svůj účet kliknutím na odkaz v e-mailu.

9. Systém provede aktivaci účtu a vyzve uživatele k přihlášení.

10. Funkce končí.

Alternativní scénáře k hlavnímu úspěšnému scénáři

5a Systém zjistí, že heslo není dostatečně bezpečné

1. Systém zjistí, že heslo není dostatečně bezpečné.

2. Systém zobrazí validační hlášení informující uživatele o bezpečném formátu hesla.

3. Funkce končí.

7a Systém zjistí, že zadaný e-mail již v databázi existuje

1. Systém zjistí, že zadaný e-mail již v databázi existuje.

2. Systém zobrazí chybové hlášení informující uživatele o existujícím zadaném e-mailu.

3. Funkce končí.

UC002 - Přihlášení uživatele do systému

Vstupní podmínky

Uživatel má ve webovém prohlížeči zobrazenou webovou stránku slevového portálu a má k dispozici formulář pro přihlášení.

Hlavní úspěšný scénář

1. Uživatel zadá přihlašovací údaje (E-mail, Heslo) a potvrdí tlačítkem pro přihlášení.

2. Systém kontroluje, zda je e-mailová adresa ve validním formátu.

3. Systém zjistí, že je e-mailová adresa ve validním formátu.

4. Systém kontroluje, zda je zadaný účet registrován a aktivován.

5. Systém zjistí, že je zadaný účet registrován a aktivován.

6. Systém provede přihlášení uživatele.

7. Funkce končí.

Alternativní scénáře k hlavnímu úspěšnému scénáři

3a Systém zjistí, že e-mailová adresa není ve validním formátu

1. Systém zjistí, že e-mailová adresa není ve validním formátu.

2. Systém zobrazí validační hlášení informující uživatele o nevalidním formátu e-mailové adresy.

3. Funkce končí.

5a Systém zjistí, že zadaný účet není registrován nebo aktivován

1. Systém zjistí, že zadaný účet není registrován nebo aktivován.

2. Systém zobrazí uživateli chybové hlášení informující o neplatném přihlášení.

3. Funkce končí.

UC003 - Zapomenuté heslo

Vstupní podmínky

Uživatel má ve webovém prohlížeči zobrazenou webovou stránku slevového portálu a má k dispozici odkaz pro zapomenuté heslo.

Hlavní úspěšný scénář

1. Uživatel kliká na link pro funkci zapomenutého hesla.

2. Systém zobrazí formulář s jedním polem pro zadání e-mailové adresy a potvrzovacím tlačítkem.

3. Uživatel zadá e-mailovou adresu a potvrdí.

4. Systém kontroluje, zda je e-mailová adresa ve validním formátu.

5. Systém zjistí, že je e-mailová adresa ve validním formátu.

6. Systém pošle do e-mailu odkaz pro změnu hesla.

7. Uživatel klikne na odkaz v e-mailu.

8. Systém přesměruje uživatele na stránku s možností zadat nové heslo.

9. Uživatel zadá z hlediska zvýšené bezpečnosti opětovně svůj e-mail, opakovaně vloží nové heslo a potvrdí tlačítkem pro reset hesla.

10. Systém kontroluje, zda heslo odpovídá bezpečnému formátu hesla, to znamená, že obsahuje minimálně 10 znaků, obsahuje malé i velké písmeno, číslici a speciální znak např. *.

11. Systém zjistí, že je heslo dostatečně bezpečné.

12. Systém provede reset hesla.

13. Funkce končí.

Alternativní scénáře k hlavnímu úspěšnému scénáři

5a Systém zjistí, že e-mailová adresa není ve validním formátu

1. Systém zjistí, že e-mailová adresa není ve validním formátu.

2. Systém zobrazí validační hlášení informující uživatele o nevalidním formátu e-mailové adresy.

3. Funkce končí.

11a Systém zjistí, že heslo není dostatečně bezpečné

1. Systém zjistí, že heslo není dostatečně bezpečné.

2. Systém zobrazí validační hlášení informující uživatele o bezpečném formátu hesla.

3. Funkce končí.

UC004 - Odhlášení uživatele

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má k dispozici funkci odhlášení.

Hlavní úspěšný scénář

1. Uživatel kliká na funkci odhlášení.

2. Systém provádí odhlášení uživatele ze systému.

3. Systém zobrazí přihlašovací formulář.

4. Funkce končí.

UC005 - Změna registračních údajů uživatele

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má k dispozici modul pro změnu registračních údajů.

Hlavní úspěšný scénář

1. Uživatel kliká na modul správy svých registračních údajů.

2. Systém zobrazí modul správy registračních údajů.

3. Uživatel mění své registrační údaje.

4. Systém provede aktualizaci registračních údajů nad účtem uživatele.

5. Systém načte změněné údaje z databáze a zobrazí je uživateli.

6. Funkce končí.

UC006 - Změna hesla uživatele

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má otevřen modul pro změnu registračních údajů.

Hlavní úspěšný scénář

1. Uživatel kliká na tlačítko změna hesla.

2. Systém zobrazí formulář pro zadání současného hesla a nového hesla.

3. Uživatel vyplní současné a nové heslo a potvrdí tlačítkem pro změnu.

4. Systém kontroluje, zda heslo odpovídá bezpečnému formátu hesla, to znamená, že obsahuje minimálně 10 znaků, obsahuje malé i velké písmeno, číslici a speciální znak např. *.

5. Systém zjistí, že je heslo dostatečně bezpečné.

6. Systém aktualizuje heslo uživatele.

7. Funkce končí.

Alternativní scénáře k hlavnímu úspěšnému scénáři

5a Systém zjistí, že heslo není dostatečně bezpečné

1. Systém zjistí, že heslo není dostatečně bezpečné.

2. Systém zobrazí validační hlášení informující uživatele o bezpečném formátu hesla.

3. Funkce končí.

UC007 - Vytvoření země

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má k dispozici modul pro správu zemí.

Hlavní úspěšný scénář

1. Uživatel kliká na tlačítko pro vytvoření země.

2. Systém zobrazí formulář pro zadání názvu země a jejího ISO kódu dle mezinárodní normy ISO 3166-1 alpha-2. Tlačítko pro uložení je označeno jako neaktivní.

3. Uživatel vyplní správnou délku ISO kódu a název země.

4. Systém provede aktivaci tlačítka pro uložení země.

5. Uživatel uloží zemi.

6. Systém provádí kontrolu duplicitních záznamů dle ISO kódu země.

7. Systém zjistí, že zadaný ISO kód ve vztahu k danému uživateli neexistuje.

8. Systém vytvoří nový záznam země v databázi.

9. Funkce končí.

Alternativní scénáře k hlavnímu úspěšnému scénáři

7a Systém zjistí, že zadaný ISO kód ve vztahu k danému uživateli již existuje

1. Systém zjistí, že zadaný ISO kód ve vztahu k danému uživateli existuje.

2. Systém zobrazí uživateli chybové hlášení informující o tom, že zadaný ISO kód již existuje.

3. Funkce končí.

UC008 - Editace země

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má k dispozici modul pro správu zemí, kde je založena alespoň 1 země.

Hlavní úspěšný scénář

1. Uživatel kliká na tlačítko pro editaci země.

2. Systém načte uložené údaje země z databáze a zobrazí uživateli formulář pro jejich editaci.

3. Uživatel provede změnu.

4. Systém provádí kontrolu duplicitních záznamů dle ISO kódu země.

5. Systém provede aktualizaci záznamu země.

6. Funkce končí.

UC009 - Smazání země

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má k dispozici modul pro správu zemí, kde je založena alespoň 1 země.

Hlavní úspěšný scénář

1. Uživatel kliká na tlačítko pro smazání země.

2. Systém zobrazí dotaz, zda si je uživatel jistý smazáním země.

3. Uživatel potvrdí dotaz souhlasně.

4. Systém kontroluje, zda neexistuje vazba.

5. Systém zjistí, že na zemi neexistuje vazba.

6. Systém provede smazání země.

7. Funkce končí.

UC010 - Vytvoření e-shopu

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má k dispozici modul pro správu e-shopů.

Hlavní úspěšný scénář

1. Uživatel kliká na tlačítko pro vytvoření e-shopu.

2. Systém zobrazí formulář pro zadání názvu e-shopu a přiřazení země. Tlačítko pro uložení je označeno jako neaktivní.

3. Uživatel vyplní název e-shopu a přiřadí k němu alespoň 1 zemi.

4. Systém provede aktivaci tlačítka pro uložení e-shopu.

5. Uživatel uloží e-shop.

6. Systém vytvoří nový záznam e-shopu s vazbou na zemi a generuje unikátní kód, který bude sloužit jako API klíč. Systém rovněž generuje tajný klíč pro zabezpečenou komunikaci přes rozhraní.

7. Funkce končí.

UC011 - Editace e-shopu

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má k dispozici modul pro správu e-shopů, kde je založen alespoň 1 e-shop.

Hlavní úspěšný scénář

1. Uživatel kliká na tlačítko pro editaci e-shopu.

2. Systém načte uložené údaje e-shopu z databáze a zobrazí uživateli formulář pro jejich editaci. Také zobrazí tajný klíč, který bude možné zobrazit kliknutím na tlačítko pro zobrazení.

3. Uživatel provede změnu.

4. Systém provede aktualizaci záznamu e-shopu.

5. Funkce končí.

UC012 - Smazání e-shopu

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má k dispozici modul pro správu e-shopů, kde je založen alespoň 1 e-shop.

Hlavní úspěšný scénář

1. Uživatel kliká na tlačítko pro smazání e-shopu.

2. Systém zobrazí dotaz, zda si je uživatel jistý smazáním e-shopu.

3. Uživatel potvrdí dotaz souhlasně.

4. Systém provede smazání e-shopu.

5. Funkce končí.

UC013 - Vytvoření položkovníku a jeho položek

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má k dispozici modul pro správu položkovníků.

Hlavní úspěšný scénář

1. Uživatel kliká na tlačítko pro vytvoření položkovníku.

2. Systém zobrazí formulář pro zadání názvu položkovníků a jeho kódu. Tlačítko pro uložení je označeno jako neaktivní.

3. Uživatel vyplní název položkovníku, kód a zvolí jeho typ (dynamický, statický).

4. Systém provede aktivaci tlačítka pro uložení položkovníku.

5. Uživatel uloží položkovník.

6. Systém kontroluje, zda již neexistuje kód položkovníku ve vztahu k uživateli v databázi.

7. Systém zjistí, že kód položkovníku neexistuje.

8. Systém vytvoří nový záznam položkovníku v databázi.

9. Uživatel označí příslušný položkovník a kliká na tlačítko pro vytvoření nové položky.

10. Systém zobrazí formulář dle typu položkovníku. Pro dynamický zobrazí pole kód a název. Pro statický položkovník pouze pole kód. Tlačítko pro uložení je označeno jako neaktivní.

11. Uživatel vyplní pole.

12. Systém provede aktivaci tlačítka pro uložení položky.

13. Uživatel uloží položku.

14. Systém kontroluje, zda ve zvoleném položkovníku neexistuje kód ukládané položky.

15. Systém zjistí, že kód položky ve zvoleném položkovníku neexistuje.

16. Systém vytvoří novou položku v databázi a současně vytváří vazbu na položkovník.

17. Funkce končí.

Alternativní scénáře k hlavnímu úspěšnému scénáři

7a Systém zjistí, že kód položkovníku v databázi existuje

1. Systém zjistí, že zadaný kód položkovníku ve vztahu k danému uživateli existuje.

2. Systém zobrazí uživateli chybové hlášení informující o tom, že zadaný kód již existuje.

3. Funkce končí.

15a Systém zjistí, že ve zvoleném položkovníku existuje kód ukládané položky

1. Systém zjistí, že ve zvoleném položkovníku existuje kód ukládané položky.

2. Systém zobrazí uživateli chybové hlášení informující o tom, že zadaný kód položky již existuje.

3. Funkce končí.

UC014 - Editace položkovníku a jeho položek

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má k dispozici modul pro správu položkovníků.

Hlavní úspěšný scénář

1. Uživatel klikne na tlačítko pro editaci položkovníku nebo položky.

2. Systém zobrazí formulář pro editaci údajů.

3. Uživatel upraví údaje a ukládá pomocí tlačítka uložit.

4. Systém kontroluje duplicitu kódu položkovníku nebo položky.

5. Systém zjistí, že kód položkovníku nebo položky neexistuje v databázi.

6. Systém aktualizuje položkovník nebo položku.

7. Funkce končí.

UC015 - Smazání položkovníku a jeho položek

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má k dispozici modul pro správu e-shopů, kde je založen alespoň 1 položkovník naplněný položkami.

Hlavní úspěšný scénář

1. Uživatel kliká na tlačítko pro smazání položkovníku nebo položky.

2. Systém zobrazí dotaz, zda si je uživatel jistý smazáním položkovníku nebo položky.

3. Uživatel potvrdí dotaz souhlasně.

4. Systém kontroluje, zda neexistuje vazba.

5. Systém zjistí, že na položku nebo položkovník neexistuje vazba.

6. Systém provede smazání položkovníku se všemi položkami nebo smazání konkrétní položky.

7. Funkce končí.

UC016 - Vytvoření slevové akce

Vstupní podmínky

Uživatel je přihlášen v aplikaci slevového portálu a má k dispozici modul pro správu slevových akcí. Uživatel má založen e-shop a k němu přiřazenou zemi.

Hlavní úspěšný scénář

1. Uživatel kliká na tlačítko pro vytvoření akce.

2. Systém zobrazí první krok průvodce pro vytvoření akce.

3. Uživatel vyplní pole ID akce, Interní název, Veřejný název, Popis, vybere e-shop, vybere zemi a nastaví prioritu.

4. Uživatel pokračuje do dalšího kroku průvodce.

5. Systém kontroluje, zda jsou vyplněna všechna povinná pole.

6. Systém kontroluje, zda je priorita unikátní v rámci kombinace země a e-shop.

7. Systém zjistí, že jsou vyplněná všechna povinná pole a priorita je unikátní.

8. Systém umožní pokračovat do druhého kroku průvodce a ukládá data z předchozího kroku do databáze.

9. Uživatel zaškrtne alespoň jednu podmínku a zaškrtne jedno vyhodnocení. Vyplní obsah zaškrtnutých podmínek a vyhodnocení.

10. Uživatel pokračuje do dalšího kroku.

11. Systém kontroluje, zda je zaškrtnuta alespoň jedna podmínka a je zaškrtnuto vyhodnocení.

12. Systém zjistí, že je zaškrtnuta podmínka a vyhodnocení.

13. Systém pokračuje do posledního kroku průvodce, ukládá data z předchozího kroku do databáze.

14. Uživatel vyplní pole Od a Do a provede označení akce jako aktivní.

15. Systém zablokuje předchozí kroky.

16. Uživatel dokončí založení akce.

17. Systém ukládá data ze třetího kroku do databáze. Pole Od a Do jsou uloženy ve formátu datum a čas. Akce je v produkčním režimu.

18. Funkce končí.

UC017 - Automatické vyhodnocení slevových akcí

Vstupní podmínky

Systém obdrží on-line dotaz skrze rozhraní.

Hlavní úspěšný scénář

1. Systém vyhodnotí on-line dotaz pomocí algoritmu vyhodnocení slev.

2. Systém vrací výstup vyhodnocený algoritmem skrze rozhraní.

3. Funkce končí.

UC018 - Zabezpečení vzdálené komunikace

Vstupní podmínky

Systém obdrží on-line dotaz skrze rozhraní.

Hlavní úspěšný scénář

1. Systém provede proces autentizaci klienta, který dotaz zasílá.

2. Systém zjistí, že byla autentizace provedena úspěšně.

3. Systém vrací odpověď na dotaz.

4. Funkce končí.

Alternativní scénáře k hlavnímu úspěšnému scénáři

2a Systém zjistí, že byla autentizace provedena neúspěšně

1. Systém zjistí, že byla autentizace provedena neúspěšně.
2. Systém vrací odpověď na dotaz s informací o neprovedené autentizaci.
3. Funkce končí.