Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Návrh a vytvoření elektronického obchodu

Diplomová práce

2022 František Štefl

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2020/2021

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

Jméno a příjmení:	**Bc. František Štefl**
Osobní číslo:	**I19289**
Studijní program:	**N0613A140007 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Návrh a vytvoření elektronického obchodu**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Náplní diplomové práce je návrh a vytvoření optimálního elektronického obchodu pro firmu VKS Rynárec, která se zabývá prodejem krmných směsí. Na základě konzultace s firmou a zkoumání jiných implementací elektronických obchodů, dojde k vytvoření optimálního návrhu a řešení elektronického obchodu, který by pomohl firmu zviditelnit, a tak zvýšit počet zákazníků. Výsledný elektronický obchod se bude skládat z uživatelského a administrátorského rozhraní, kdy uživatelské rozhraní umožňuje prohlížení produktů, filtrování, vkládání produktů do košíku, objednání zboží a provedení platby. Administrátor umí vkládat, editovat a mazat produkty i jejich kategorie, zadávat akce i slevy a spravovat objednávky. Velkou předností bude použití moderních technologií Blazor a React.

Rozsah pracovní zprávy: **50 – 60 stran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

***MIKULÁŠKOVÁ, Petra a Mirek SEDLÁK. Jak vytvořit úspěšný a výdělečný internetový obchod. Brno: Computer Press, 2015. ISBN 978-80-251-4383-4.**
***ORR, Eli a Yehuda ZADIK. Programming with CodeIgniter MVC: build feature-rich web applications using the CodeIgniter MVC framework. Birmingham: Packt Publishing, 2013. ISBN 978-1-84969-470-4.**
***HIMSCHOOT, Peter. MICROSOFT BLAZOR: building web applications in .net. 2nd ed. New York: APRESS, 2020. ISBN 978-1484259276.**

Vedoucí diplomové práce: **PaedDr. František Smrčka, Ph.D.**
Vysoká škola polytechnická Jihlava

Datum zadání diplomové práce: **6. listopadu 2020**
Termín odevzdání diplomové práce: **15. května 2021**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

prof. Ing. Antonín Kavička, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 30. listopadu 2020

Prohlašuji:

Práci s názvem „Návrh a vytvoření elektronického obchodu“ jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

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

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

V Pardubicích dne 20.05.2022

František Štefl v.r.

PODĚKOVÁNÍ

V první řadě bych rád poděkoval svému vedoucímu práce, panu doktoru Františku Smrčkovi, za odborné vedení a rady při vytváření textové části této diplomové práce.

Dále bych rád poděkoval odborníkovi z praxe, Petru Musilovi, za cenné rady, bez kterých by vytvoření elektronického obchodu nebylo možné.

ANOTACE

Diplomová práce se primárně zaměřuje na návrh a vytvoření elektronického obchodu na míru pro firmu VKS Rynárec. Zabývá se též popisem použitých technologií při vytváření el. obchodu a také odůvodněním, proč byly tyto technologie zvoleny. Nadále je zde srovnání s konkurenčními technologiemi a obdobně zaměřenými elektronickými obchody podobného rozsahu. V druhé polovině diplomové práce je popsána analýza a návrh, popis zajímavých částí kódu a následně výsledný elektronický obchod, a nakonec testování a nasazení. Zajímavostí a unikátností je použití nového frameworku Blazor pro vytvoření administrátorského rozhraní, který umožňuje psaní frontendové části v programovacím jazyce C# oproti dnes standartně využívaném jazyce JavaScript.

KLÍČOVÁ SLOVA

ASP.NET Core, Blazor, e-shop, elektronický obchod, internetový obchod, programování, React, webová aplikace

TITLE

Design and implementation of e-shop

ANNOTATION

This diploma thesis deals with design and implementation of custom e-shop for company VKS Rynárec. It deals with description of used technologies for implementation of e-shop as well and lastly justification why these technologies were chosen. Furthermore, diploma thesis contains comparison of used technologies and competitive technologies and also comparison of e-shops with similar scope and purpose. In the second half of this diploma is described the analysis and design, description of interesting parts of code and final e-shop, and finally testing and deployment of the e-shop. Uniqueness of this diploma thesis is use of technology called Blazor for implementation of administrator interface. Blazor allows programmer to use C# programming language to implement frontend code instead of JavaScript which is commonly used nowadays.

KEYWORDS

ASP.NET Core, Blazor, e-shop, electronic shop, web store, programming, React, web application

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZDROJOVÝCH KÓDŮ

SEZNAM ZKRATEK A ZNAČEK

AI – Artificial Intelligence (umělá inteligence)

API – Application Programming Interface

GUI – Graphical User Interface (grafické uživatelské rozhraní)

HTML – Hypertext Markup Language

HTTP – Hypertext Transfer Protocol

JSON – JavaScript Object Notation (JavaScriptový objektový zápis)

JWT – JSON Web Token

ML – Machine Learning (strojové učení)

ORM – Object Relational Mapping (objektově relační mapování)

REST – Representational State Transfer

RPC – Remote Procedure Call (vzdálené volání procedur)

SOAP – Simple Object Access Protocol

SPA – Single Page Application (jednostránková webová aplikace)

SQL – Structured Query Language (strukturovaný dotazovací jazyk)

UI – User Interface (uživatelské rozhraní)

URL – Uniform Resource Locator (jednotný lokátor zdroje)

XML – Extensible Markup Language

ÚVOD

V dnešní době, kdy je internet nedílnou součástí našich životů, je velice oblíbené nakupování online v elektronických obchodech. Elektronické obchody jsou zejména populární díky své jednoduchosti, pohodlnosti, a hlavně rychlost nákupního procesu, protože v dnešní době je čas velmi cenný. Další významnou vlastností elektronických obchodů je i to, že dokážou přilákat nové zákazníky a při naplnění jejich potřeb si je i udržet a tím zvýšit dané firmě prodeje a zisky. Nonstop dostupnost elektronických obchodů je také jednou z velkých výhod oproti kamenným obchodům, které mají stanovenou otevírací dobu, které se musí zákazník přizpůsobit, pokud má zájem o nabízené zboží.

Firma VKS Rynárec, která se specializuje na výrobu a prodej krmných směsí pro zvířata, doposud neprovozovala žádný elektronický obchod, a proto jsem se s ní rozhodl navázat spolupráci. Firma se svou výrobou zaměřuje převážně na malé a střední odběratele, jelikož se jedná o malou rodinnou firmu. Cílem diplomové práce je návrh a vytvoření elektronického obchodu na míru, který by měl být jednoduchý a přehledný, aby se v něm zákazníci dobře orientovali. Důvodem je, že nepřehledný a složitý elektronický obchod snižuje důvěru zákazníka, a tedy i pravděpodobnost vytvoření objednávky zákazníkem.

Pro vytvoření elektronického obchodu byly zvoleny primárně technologie od společnosti Microsoft. Použita byla architektura ASP.NET Core a programovací jazyk C#. Administrátorská část elektronického obchodu je napsána v rozhraní Blazor a uživatelská část je napsána v moderní JavaScriptové knihovně React. Dále je v uživatelské části využít programovací jazyk TypeScript, který je nadstavbou nad jazykem JavaScript, jenž rozšiřuje o statické typování a další atributy pro lepší přehlednost a čitelnost kódu.

Diplomová práce se dá rozdělit na dvě stěžejní části. V první části práce je představen řešený problém a cíl práce. Dále jsou zde popsány konkurenční elektronické ochody, použité technologie a alternativní technologie k použitým technologiím. V druhé části diplomové práce je obsažena analýza firmy a firemních požadavků na základě kterých je představen návrh řešení tvorby elektronického obchodu. Dále jsou zde popsány zajímavé části implementované aplikace neboli elektronického obchodu a také představena výsledná podoba a funkcionality elektronického obchodu. Poslední kapitola druhé části diplomové práce je zaměřena na testování a nasazení elektronického obchodu.

1 PŘEDSTAVENÍ PROBLÉMU A CÍL PRÁCE

V první části této kapitoly je vysvětlen důvod vytváření webové aplikace a v druhé představeny cíle a požadavky webové aplikace.

1.1 Představení problému

VKS Rynárec je poměrně malá firma zaměřující se na výrobu a prodej krmných směsí pro domácí zvířata. Firma vlastní jednu kamennou prodejnu, kde vyrábí a zároveň prodává veškeré své produkty. Dále pak firma nabízí službu objednání zboží přes telefonní linku. Objednané zboží následně doručuje svými vozidly po blízkém okolí (cca 20 km v okruhu firmy). Firma neprovozuje vlastní elektronický obchod. Pouze statické webové stránky, na kterých je uvedeno pár základních informacích o firmě a zobrazen menší počet nabízeného zboží.

V dnešní době je nakupování online velice oblíbené, a proto by měla každá firma zvážit realizaci vlastního e-shopu. Pomocí e-shopu je firma schopna představit mnohem větší sortiment produktů, než by dokázala v kamenném obchodě, kam se mnohdy veškeré produkty nevejdou. Dále e-shopy mnohdy umožňují lepší vizualizaci produktů. Správně nafocené a hezky vypadající produkty jsou z velké části rozhodující faktor pro zakoupení onoho produktu. Jednou z největších výhod e-shopů je, že šetří čas, který je v dnešní uspěchané době velmi cenný. Stačí opravdu pár kliknutí a chtěný produkt je objednán. S tím se pojí i další vlastnost a to je, že nakupovaní online je příjemné. Člověk nemusí fyzicky dojít do kamenné prodejny, čekat ve frontě nebo hledat, kde se chtěný produkt v prodejně nachází.

Firma by ráda přilákala nové zákazníky ze svého okolí a těm stálým zpříjemnila využívání služeb firmy. Dále se chce firma za pomoci e-shopu více zviditelnit a celkově zlepšit své působení na trhu. Výslední e-shop chce firma využívat jako doplněk ke kamenné prodejně, nikoliv jako hlavní zdroj obživy firmy.

1.2 Cíl a požadavky e-shopu

Cílem diplomové práce je vytvoření interaktivního elektronického obchodu (e-shop) pro firmu VKS Rynárec za použití moderních technologií. Firma provozuje jednoduché webové stránky, které obsahují informace o firmě a menší počet nabízených produktů. Vytvořený e-shop by měl nahradit stávající webové stránky. Tímto krokem má firma za cíl se více zviditelnit a přilákat tak nové zákazníky.

Výsledný e-shop by měl být jednoduchý a přehledný, aby se v něm zákazník lehce orientoval. Zobrazované produkty by měly obsahovat obrázek, název produktu, cenu a ovládací prvek,

kterým se produkt vloží do košíku. Ve zmíněném košíku bude seznam produktů obsahující základní informace o produktech a ovládací prvky pro změnu počtu kusů v košíku a odebrání produktu z košíku. Dále zde bude zobrazena průběžná cena objednávky. Hlavní stránka s listem veškerých produktů bude umožňovat stránkování, řazení a filtrování produktů dle jejich kategorie a také vyhledávání produktů.

Jedním z požadavků firmy je neumožnění platby na dobírku při doručování zboží firemními vozidly. Firma se obává, že zákazníci si objednané zboží nebudou vyzvedávat, a tedy bude vznikat finanční ztráta za spotřebované palivo a čas. Dále si firma stanovila v doručovacích podmínka lhůtu doručení do 5 dnů. Důvodem tohoto rozhodnutí je vyvarování se doručování malých zásilek každý den. Úmysl firmy je naplnit kapacitu vozidla a až poté provést doručení objednávek, aby se co nejvíce minimalizovaly výdaje za uskutečněné doručení objednávek.

Součástí e-shopu bude i administrátorské rozhraní pro správu e-shopu. Toto rozhraní bude jednoduché a bude obsahovat základní funkce pro minimální správu e-shopu. Administrátor bude moci přidávat, upravovat a mazat jednotlivé produkty a kategorie a spravovat objednávky. Správa objednávek bude, po přidání veškerých produktů do e-shopu, nejpoužívanější funkcí. Pomocí této funkce budou zaměstnanci firmy připravovat vytvořené objednávky.

Kompletní e-shop bude implementován za použití moderních technologií a standardů. Konkrétně administrátorské rozhraní bude vytvořeno za pomoci nového frameworku Blazor od firmy Microsoft. Blazor umožňuje vytváření interaktivních uživatelských rozhraní (UI) za použití programovacího jazyka C#. Díky tomu je možné vytvářet klientskou část a serverovou část webové aplikace v jednom programovacím jazyce. Administrátorské rozhraní a serverová část e-shopu bude napsána v jazyce C# a pro komunikaci mezi admin. rozhraním a serverem bude použita další moderní technologie gRPC od firmy Google. gRPC je framework, který realizuje technologii vzdáleného volání procedur. Velkou výhodou gRPC je, že se jedná o multiplatformní framework, což umožňuje jeho použití pro komunikaci mezi platformami napsaných v různých jazycích.

Klientská část e-shopu bude vytvořena ve frameworku React, který používá programovací jazyk JavaScript. React umožňuje vytváření tzv. single page aplikací, kdy dochází k pře-renderování pouze měnící se části obsahu webové stránky a díky tomu šetří čas a velikost přenášených dat mezi klientem a serverem. Dále pak místo programovacího jazyka JavaScript bude použita jeho nadstavba TypeScript. TypeScript je programovací jazyk vyvinutý firmou

Microsoft, který rozšiřuje JavaScript o vlastnosti z objektově orientovaného programování jako jsou: statické typování, třídy, rozhraní a další. Pro komunikaci mezi klientem a serverem bude použita technologie REST API. Jedná se o programovací rozhraní, které je orientováno na data (typicky ve formátu JSON), užívané pro komunikaci mezi klientem a serverem za pomoci definování jedinečných HTTP adres na které klient přistupuje.

2 KONKURENČNÍ E-SHOPY

Náplní této kapitoly je prozkoumání konkurenčních webových stránek či aplikací. Jednotlivé webové stránky a aplikace budou popsány a porovnány s návrhem řešením elektronického obchodu (e-shop) pro firmu VKS Rynárec.

2.1 VKS Pohledští Dvořáci a.s.

První konkurenční webovou stránkou je vkshb.cz [1]. Jedná se o jednoduché webové stránky tvořené několika stránkami obsahující informace o firmě, firemních služeb a nabízeného zboží. Mezi jednotlivými stránkami lze přepínat pomocí navigačního menu umístěného v pravém horním rohu. Tyto webové stránky jsou celkem nepřehledné a obtížně se v nich orientuje. Pro zobrazení stránky s výpisem nabízených produktů je nutné, nalézt odkaz pro přesměrování na tuto stránku, v textu, který je umístěn na stránce Krmiva. Veškeré produkty jsou zobrazeny na jedné stránce po sebou. U jednotlivých produktů je zobrazen jeden obrázek, název, popis a cena produktu. Také se zde není možnost zobrazení bližších informací o vybraném produktu neboli není zde stránka s detailem produktu. Tyto webové stránky neumožňují vytvoření objednávky, a tedy se nejedná o e-shop.

Při srovnání těchto webových stránek s vytvářeným e-shope je zřejmé, že vytvářený e-shop obsahuje mnohem více funkcí. Webové stránky vkshb.cz slouží pouze ke zobrazení informací zákazníkovi, zatímco navrhovaný e-shop obsahuje navíc funkce pro vkládání produktů do košíku, vytváření objednávek, vytváření uživatelských účtů, filtrování produktů a další. Z tohoto důvodu nebyly získány žádné poznatky a užitečné informace z prozkoumání a porovnání webových stránek vkshb.cz

2.2 Ryhos, s.r.o.

Další konkurencí jsou webové stránky ryhos.cz [2]. Zde už jde o funkční e-shop, který umožňuje vkládání produktů do košíku a následné vytvoření objednávky. Tento e-shop je jednoduchý a přehledný, avšak není zde funkce pro řazení produktů. Je zde možnost filtrovat produkty podle kategorií a také možnost vyhledávání produktů. E-shop dále umožňuje registraci a přihlášení, ale neumožňuje zobrazení nebo úpravu profilu, prohlížení vytvořených objednávek a přidání nebo upravení dodacích adres. Nákupní košík je přehledný a obsahuje funkce pro odebrání produktu z košíku a změnu počtu kusů v košíku. Na jednotlivé produkty v košíku lze kliknout a dostat se tak na detail produktu. Košík jako takový je součástí procesu vytvoření objednávky, který je rozdělen na několik částí.

Zde se již jedná a vcelku designově příjemný e-shop, který obsahuje veškeré základní funkcionality pro zpříjemnění jeho používání. Možnost registrace účtu je zde naprosto zbytečná, neboť nepřináší žádné další výhody nebo účel. Zde vzniká výhoda vytvářeného e-shopu v rámci diplomové práce, protože registrovaný uživatel má možnost vytvářet, upravovat a mazat adresy, změnit své přihlašovací heslo a také prohlížet vytvořené objednávky. Další výhodou je funkce pro řazení produktů, která v e-shopu ryhos.cz není obsažena.

I když jsou oba e-shopy jednoduché a menších rozměrů, tak e-shop firmy VKS Rynárec, s.r.o. nabízí o poznání více funkcí, které zpříjemňují jeho využívání a dělají ho tak lepším e-shopem. Jednou z takových funkcí je držení si vyplněných uživatelských informací při vytváření objednávky. Pokud se zákazník vrátí z kroku vytváření objednávky do košíku a poté nazpět k vytváření objednávky, pole s adresou zůstanou vyplněny.

2.3 Agrotree

E-shop agrotree.cz [3] je velmi dobře zpracovaný, přehledný s příjemným grafickým designem a rozložením jednotlivých prvků. E-shop se skládá z horního menu, kde se nachází vstupní pole pro vyhledávání, telefonní číslo a otvírací doba, košík a odkazu pro přihlášení nebo rozbalující menu obsahující položky pro změnu údajů, zobrazení objednávek a odhlášení se. V levé části se nachází panel s kategoriemi a podkategoriemi produktů. Uprostřed stránky je umístěna obsahová část s produkty, která dále obsahuje posuvník pro filtrování dle ceny produktů a filtrování dle příznaku skladem. Dále se zde nachází tři tlačítka pro možnost řazení produktů. Ve spodní části stránky se nachází stránkování a zápatí, které obsahuje odkazy na různé funkce, kontaktní údaje, mapa sídliště firmy a odkazy na sociální sítě.

Tento e-shop obsahuje veškeré nezbytné funkce jako jsou: přihlášení a registrace, obnovení hesla, změnu osobních údajů (pouze doručovací a fakturační adresu), prohlížení objednávek, filtrování, stránkování a vyhledávání produktů a v neposlední řadě možnost vložit produkt do košíku a vytvořit objednávku. Registrace uživatele umožňuje a zároveň vyžaduje po uživateli zadat jeho fakturační adresu. Doručovací adresa a firemní údaje jsou při registraci volitelné. Proces vytváření objednávky je umístěn přímo do košíku, kde uživatel vidí jednotlivé produkty v košíku a níže je umístěn formulář s volbou dopravy a platby a vstupní pole pro zadání doručovacích údajů.

Většinu funkcionalit e-shopu agrotree.cz obsahuje i e-shop firmy VKS Rynárec, s.r.o. ale v některé funkce se odlišují. V e-shopu VKS Rynárec, s.ro. není umožněné filtrování produktů

podle jejich ceny a příznaku skladem. Dále zde není možnost provedení platby převodem skrze platební bránu GoPlay, kterou právě e-shop agrotree.cz obsahuje. E-shop VRK Rynárec, s.r.o. však nabízí možnost vytváření několika doručovacích adres a také možnost změnu hesla, která není v e-shopu agrotree.cz obsažena.

2.4 Kvidera

Kvidera [4] e-shop je po stránce obsažených funkcionalit srovnatelný s předešlým zmíněným e-shopem Agrotree. Obsahuje ale znatelně více interaktivních prvků a grafických elementů, což není nutně výhodou. V horní části elektronického obchodu se nachází odkazy s informativním obsahem o dopravě, firmě atd. Dále je zde umístěno vstupní pole pro vyhledávání, nákupní košík, odkazy pro přihlášení a registraci nebo tlačítko pro odhlášení se, zobrazení seznamu objednávek a úpravu údajů a také telefonní kontakt pro uskutečnění objednávky po telefonu. V poslední řadě je zde umístěno přehledné menu s kategoriemi nabízených produktů. Uprostřed se nachází seznam s produkty a také ovládací tlačítka pro řazení produktů a stránkování. Ve spodní části webu jsou sepsány kontaktní údaje, odkazy pro často hledané informace, a nakonec odkaz na sociální sít Facebook.

Ze všech předešle zmíněných a popsaných webových stránek a e-shopů je tento nejprofesionálnější. Grafický design a rozložený interaktivních prvků budí dojem kvality za cenu trochu nižší přehlednosti a horší orientace oproti e-shopu Agrotree. Dále je v e-shopu možné vytvoření účtu, možnost obnovení hesla, editace kontaktních údajů a doručovací adresy a změna přihlašovacího hesla. Proces registrace stejný jako je v e-shopu Agrotree a tedy je uživatel povinen vyplnit osobní údaje a doručovací adresu a zároveň může mít pouze jednu adresu. Rovněž je v e-shopu funkce pro vyhledávání produktů, jejich filtrování dle kategorie a řazení dle ceny nebo abecedně. Nakonec proces vytvoření objednávky je totožný s e-shopem Agrotree kde je vytváření objednávky sloučeno s košíkem.

E-shopy Kvidera a Agrotree jsou kvalitně zpracované. Oba tyto e-shopy obsahují stejné funkce, které se liší pouze v detailech. Např. v e-shopu Kvidera je uživateli umožněna změna hesla, kdežto v e-shopu Agrotree nikoliv. E-shop Agrotree oproti e-shopu Kvidera obsahuje funkci pro filtrování dle ceny a příznaku skladem.

2.5 Závěr porovnání

Z provedené analýzy konkurenčních nebo podobně zaměřených elektronických obchodů byly získány důležité informace a poznatky. V první řadě u volby barev jednoznačně dominuje

zbarvení do zelena. Volba této barvy je správná, neboť zelená barva je přirozená pro odvětví zaměřené na prodej zvířecích krmiv, zahradních doplňků apod. Dále se na většině webových stránkách nachází obrázky nebo ikony zvířat či rostlin, které vyplňují prázdná místa webové stránky a zvyšují pozitivní dojem z celkového grafického dizajnu webové aplikace.

Dalším výsledkem analýzy je obsahová část jednotlivých e-shopů. Analyzované e-shopy se shodují v obsažení menu kategorií nabízených produktů dle kterého lze produkty filtrovat. Toto se menu se nachází buďto v levé části stránky nebo v horní části nad seznamem produktů. Jednotlivé kategorie obsahují podkategorie maximálně 3. úrovně. Dalším způsobem filtrování produktů je možnost vyhledávání, filtrování dle ceny nebo příznakem dostupnosti zboží. Jednotlivé produkty lze i řadit dle jejich názvu či ceny. Nakonec proces vytváření objednávky je možné realizovat jako jeden krok, kdy formulář pro vyplnění adresy a uživatelských informací je přímo umístěn v košíku, nebo jej lze rozdělit na jednotlivé kroky.

Na základě provedené analýzy byly získány poznatky o tom, jak by mohl vytvářený elektronický obchod vypadat a jak by mohly být rozloženy jednotlivé části e-shopu (menu, obsahová část, obrázky atd.). Dále pak jaké funkcionality by měl obsahovat a jak může být realizován proces vytváření objednávky. Získané poznatky z analýzy konkurenčních e-shopů a webových stránek byly použity při navrhování řešení e-shopu pro firmu VKS Rynárec, s.r.o.

3 POUŽITÉ TECHNOLOGIE

Cílem této kapitoly je popsat veškeré použité technologie při implementaci e-shopu a odůvodnit jejich použití pro implementaci webové aplikace.

3.1 Visual Studio

Jedná se o integrované vývojové prostředí (IDE) od společnosti Microsoft, určené primárně pro operační systém Windows. V roce 2015 Microsoft uvolnil verzi Visual Studia i pro operační systémy Linux a MacOS. „Visual Studio je kreativní odpalovací rampa, kterou lze použít k editaci, debuggingu, vytváření kódu a následnému publikování aplikace. Dále toto IDE oproti ostatním integrovaným vývojovým prostředím nabízí kompilátory, nástroje pro dokončování kódu, grafický designér a mnoho dalších vlastností pro vylepšení vývoje softwaru[1]“ [5].

„Celé prostředí Visual Studio (VS) bylo v zásadě navrženo k tomu, aby v něm bylo možné použít kterýkoli programovací jazyk. VS je prostředí, jež nabízí všechny nástroje potřebné pro vytváření aplikací. Programovací jazyk je pak pouze jedním z nástrojů. Díky nástrojům ve VS je možné připojit se k databázi, prohlížet její objekty, číst informace, které nás zajímají, a dokonce je ukládat do objektů dostupných z kteréhokoli programovacího jazyka. Aby se vývoj aplikací zjednodušil, VS nabízí prostředí, které je společné všem programovacím jazykům“ [6, s. 22].

První verze tohoto vývojového prostředí byla vydána již v roce 1997 a šlo o první pokus vytvoření vývojového prostředí, které by podporovalo více než jeden programovací jazyk [7, s. 1]. Pro rok zveřejnění této diplomové práce je poslední verzí VS 17.1, která byla uvolněna 15. února 2022. VS je dostupné ve třech edicích, z nichž jedna je zdarma a zbylé dvě jsou placené.

Bezplatnou edicí VS je verze Community, která je určena především pro studenty, jednotlivce a malé týmy programátorů. Tato verze je bez jakéhokoliv omezení, a tedy obsahuje veškeré funkcionality jako zbylé dvě placené verze. Firmy v kategorii Enterprise (více než 250 počítačů, nebo roční výnos firmy přesahující 1 milión amerických dolarů) smí Community verzi využívat pouze ke studijním účelům, pracím na open source projektech anebo za účelem vyučování. Firmy, které nespadají do kategorie Enterprise, mohou využívat maximálně 5 kopií verze

[1] The Visual Studio IDE is a creative launching pad that you can use to edit, debug, and build code, and then publish an app. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to enhance the software development process.

Community. Pro ostatní zaměstnance musí firma dokoupit kopie ve verzi Professional nebo Enterprise. [8]

VS Professional je standardní placená edice pro komerční užití. Poskytuje navíc plnou implementaci funkce CodeLens (zobrazuje informace o poslední editaci metody, reference v kódu pro danou metodu, nebo zdali byly testy úspěšné) oproti verzi Community, kde je tato funkce implementována jen z části. [8]

Poslední edicí VS je verze Enterprise, která obsahuje veškeré funkcionality, jež jsou ve VS dostupné. Tato verze obsahuje navíc pokročilé funkce testování jako je například IntelliTest, které umožňuje generování unit testů. Dále pak vylepšené funkce ladění a diagnostiky, díky kterým lze snáze nacházet a opravovat chyby v kódu. Jednou z takových funkcí je funkce IntelliTrace, která umožňuje zaznamenávání určitých částí kódu, které je možné si prohlédnou po dokončení debuggingu, a tedy není nutné znovuspuštění aplikace, dále pak dovoluje zobrazení obsahu proměnných, anebo informací o volaných funkcích. Nakonec jsou zde i rozšiřující funkce, které zjednodušují a zpříjemňují práci při psaní programů apod.

Pro toto vývojové prostředí jsem se rozhodl, jelikož s ním mám zkušenosti a příjemně se mi v něm pracuje. Díky zaměstnání mám i přístup k verzi Enterprise, takže mohu plně využívat veškerých funkcionalit, která tato verze nabízí. Dalším důvodem je veliká podpora různých programovacích jazyků a platforem a možnost rozšiřitelnosti VS o další funkcionality. Příkladem známého doplňku je ReSharper, který ale osobně nepoužívám, neboť Microsoft velkou část funkcionalit, obsažených v tomto rozšiřujícím doplňku, převzal a implementoval přímo do VS. V neposlední řadě, jak již bylo zmíněno, VS podporuje velké množství programovacích jazyků, a tak se dá i pohodlně použít pro tvorbu frontendových částí webových aplikací, kde se především pracuje s jazykem JavaScript nebo TypeScript. Z tohoto důvod není nutné použití jiných vývojových prostředí (např. Visual Studio Code) pro tvorbu frontendové části, ale vše se dá pohodlně naimplementovat ve VS.

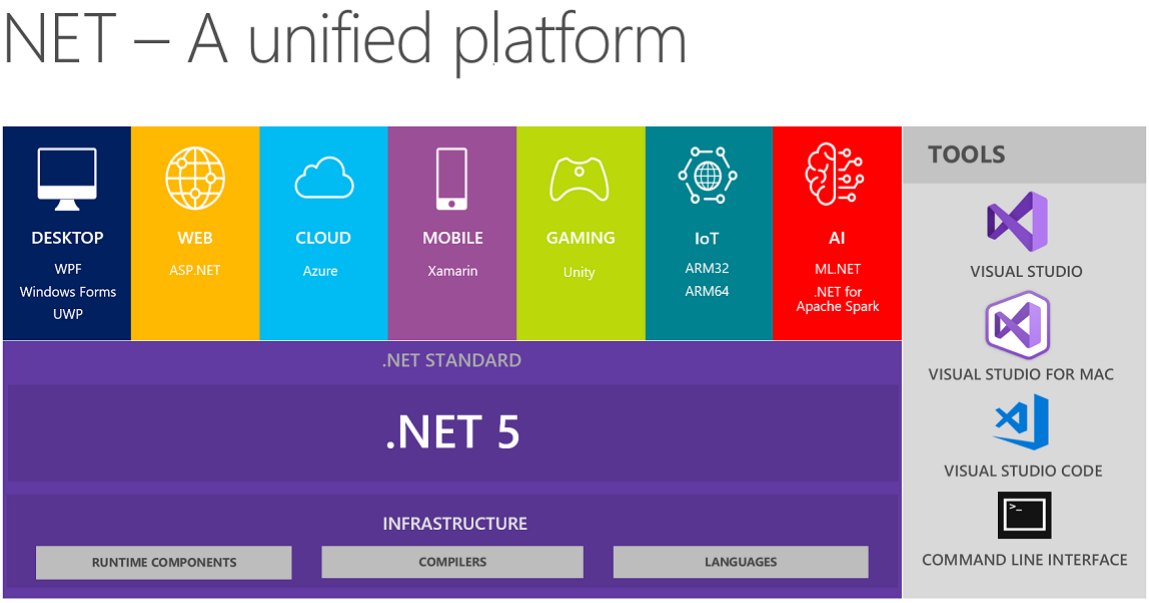
3.2 .NET

.NET je bezplatná, open-source platforma pro vytváření mnoha různých druhů aplikací, jako jsou například: webové aplikace, web API, mobilní aplikace, desktopové aplikace, hry, konzolové aplikace a další. .NET podporuje tři programovací jazyky kterými jsou: C#, F# a Visual Basic. [9]

„.NET integruje více jazyků, správu výjimek, služby pro ladění a profilování, obsahuje vylepšené zabezpečení, účinnější správu různých verzí aplikací a rozšiřování na jednotlivé pracovní stanice. Platforma .NET sjednocuje programovací model, takže volba programovacího jazyka je nyní víceméně záležitostí osobní volby. Členové jednoho programátorského týmu mohou s klidem používat různé jazyky, aniž by to nějak ovlivnilo opakovanou použitelnost kódu a integraci“ [10, s. 15].

První verze .NETu byla představena v roce 2002 a nesla název .NET Framework. Tato verze byla čistě zaměřena na operační systém Windows a Windows Server. Nejnovější, ale také poslední verzí .NET Frameworku je verze 4.8 vydaná v roce 2019. Další verzí, nebo spíše novou implementací .NETu je .NET Core, vydaný v roce 2016. Jeho aktuální verze je 6.0.3 (uvolněna 8. května 2022), která se označuje jednotným názvem .NET 6. .NET Core neboli .NET 6 je na rozdíl od .NET Frameworku podporován napříč všemi platformami. Jedná se o vylepšený .NET Framework, ať se to týká zabezpečení, výkonnosti, nebo optimalizace a podpory široké škály různých typů aplikací. Kvůli přehlednosti, a především kvůli sjednocení všech .NET frameworků (.NET Framework, .NET Core, .NET Standard …), došlo při vydání nové verze .NET Core ke změně názvu na samostatný název .NET [číslo verze]. Podle Microsoftu nový .NET nenahrazuje .NET Framework 4.8, který ale i nadále zůstává podporován a udržován. [11], [12]

Myšlenkou .NETu je sjednocení celého ekosystému. V roce 2019 byl vydán .NET 5, kde došlo k přejmenování předešlé verzi .NET Core 3.1 na .NET a sjednocení veškerých .NET elementů do jednoho balíčku vývojové platformy .NET 5. Následně v roce 2021 vyšla verze .NET 6, která toto sjednocení ekosystému dokončila.



Obrázek 1 – Organizace .NET platformy [11]

2002	2016	2020		
.NET Framework	.NET Core	.NET 5	→	Sjednocení pro desktopové aplikace, web. aplikace, cloud, mobile, hry, IoT, ML a AI
Pouze Windows	Cross Platform	Cross Platform		
	↓	↓		
	Windows, macOS a Linux	Windows, macOS, Linux, iOS, Android, tvOS, watchOS, WebAssembly		

Tabulka 1 – Vývoj verzí .NETu

3.3 ASP.NET Core

Nadstavba .NETu pro vývoj, debuging a nasazení dynamických webových aplikací, i vytváření back-endu pro mobilní aplikace. ASP.NET Core poskytuje přístup ke všem třídám a funkcím .NETu. V názvu zůstává slovo Core, aby nedocházelo k záměně s ASP.NET MVC.

ASP.NET Core je open-source framework pro vývoj webových aplikací podporovaný platformami macOS, Linux a Windows. Jedná se o nadstavbu .NET platformy (.NET Framework a .NET Core) a nástupce ASP.NETu. ASP.NET Core je napsán čistě od základu a spojuje dvě původně samostatné ASP.NET MVC a ASP.NET Web API do jednoho programovacího modelu, který je více zaměřen na modularitu než jeho předchůdci. Velkou výhodou ASP.NET Core je volitelná komponenta Blazor, která umožňuje psaní front-endové části v jazyce C#. [13], [14]

.NET 5.0
ASP.NET Core 5.0
↓
ASP.NET MVC Web Api
Signal R
Blazor
Razor Pages
gRPC

Tabulka 2 – Funkce ASP.NET Core 5.0

Důvodem použití .NETu a jeho nadstavby ASP.NET Core je, že se jedná o moderní technologie od firmy Microsoft, který obsahují funkce pro vytváření bezpečných a moderních webových aplikací. ASP.NET Core také obsahuje framework Blazor, který slouží k jednoduché a bezpečné implementací front-endové části aplikace. ASP.NET Core obsahuje i ORM nástroj Entity Framework, který zajišťuje automatickou konverzi dat mezi databází a aplikací.

3.4 C#

„C# je moderní, objektově orientovaný a typově bezpečný programovací jazyk. Tento jazyk umožňuje vývojářům vytvářet mnoho typů bezpečných a robustních aplikací, které běží na

.NET architektuře. Jedná se o programovací jazyk s větší mírou abstrakce[2], který má své kořeny v rodině C programovacích jazyků a díky tomu je jeho syntaxe hodně podobná programovacímu jazyku Java, JavaScript, C a C++[3]“ [15].

Výhoda C# spočívá především v množství programovacích technik, které podporuje, jako je nabídka objektově orientovaných funkcí, genericita a funkcionální programování. Podpora jak dynamického, tak statického typování, a především šikovný nástroj pro sestavování dotazů nad daty jakéhokoliv formátu zvaný Language Integrated Query (LINQ). V poslední řadě také podpora asynchronního programování. [16] (s. 2)

C# obsahuje hned několik funkcí, díky kterým je tento programovací jazyk jedním z nejpoužívanějších na světě. Jednou z takových funkcí je pokročilá správa paměti, o kterou se stará tzv. garbage collector (GC). GC automaticky uvolňuje paměť, kterou aplikace či program již nevyužívá nebo nepotřebuje. GC je dnes již běžnou součástí moderních programovacích jazyků, a i když jde o vítanou funkci, tak v některých případech může mít GC negativní dopad na výkonnost programovacího jazyka. Verze C# 7.2 vydaná v roce 2017 přinesla rozšiřující funkce pro lepší správu paměti, které zlepšují výkon jazyka na úkor jednoduchosti kódu se zachováním typové bezpečnosti. Toto dovoluje využití jazyka C# v aplikacích, které jsou čistě zaměřeny na výkon, kde se po dlouhá léta pouze využívaly méně bezpečné, ale rychlé jazyky C a C++. [16] (s. 2)

Další významnější funkcí C# je podpora typů s možnou null hodnotou. To umožňuje programátorům explicitně nastavit, zdali je možné přiřadit do proměnné hodnotu null či nikoliv. Tyto proměnné se nazývají nullable a non-nullable. Překladač následně uplatňuje odlišná pravidla pro tyto nullable a non-nullable proměnné. Pro nullable proměnné platí, že pouze v případě, že daná proměnná neobsahuje null hodnotu, může dojít k dereferenci této proměnné. Nullable proměnná může být totiž inicializována s null hodnotou, následně ji může být přiřazena non-null hodnota a v jiné části kódu může být proměnné znovu přiřazena hodnota null. Pokud jde o non-nullable proměnnou, tak v momentě její inicializace ji musí být přiřazena

[2] Zápis zdrojového kódu je bližší tomu, jak dané problémy chápe a zpracovává člověk.
[3] C# (pronounced "See Sharp") is a modern, object-oriented, and type-safe programming language. C# enables developers to build many types of secure and robust applications that run in .NET. C# has its roots in the C family of languages and will be immediately familiar to C, C++, Java, and JavaScript programmers.

non-null hodnota a zároveň ji nikdy, v jejím životním cyklu, nesmí být přiřazena hodnota null, jinak by došlo k vyvolání výjimky. [17]

Mnoho programovacích jazyků umožňuje rozšíření funkcionality skrze knihovny. Tato možnost rozšíření funkcionality je v dnešní době nezbytná a C# není výjimkou. Ano, existují programovací jazyky, které jsou zaměřeny pouze na specifické použití, ale v případě programovacích jazyků jako jsou Java, Python a C# je možnost rozšíření funkcionality skrze knihovny nezbytné. Pro C# existuje nespočetné množství knihoven, ať už se jedná o knihovny pro vývoj webových aplikací, nebo pomocné knihovny pro testování a debuggování, anebo knihovny jako je ReSharper, která značně usnadňuje programování pomocí našeptávání, generování kódu a kontrolování napsaného kódu.

Výše zmíněné vlastnosti tohoto jazyka jsou jedním z důvodů, proč jsem se rozhodl ho použít pro implementaci elektronického obchodu. Dalším důvodem je fakt, že se jedná o programovací jazyk vyvíjený firmou Microsoft. Zbylé použití technologie jsou také od této firmy a společně tvoří jednoduchý a fungující ekosystém, pro který existuje velké množství návodů a dokumentací. Hlavním důvodem je ale fakt, že se jedná o jeden z prvních programovacích jazyků, který jsem se začal učit a pro který jsem nabral i nové znalosti na Univerzitě Pardubice, tedy se jedná o programovací jazyk, se kterým mám nejvíce zkušeností. Nakonec je vývoj v tomto programovací jazyku velmi příjemný, přehledný, jednoduchý, a proto nebyl důvod přecházet na jiný programovací jazyk jako je např. Java, se kterým jsem se také seznámil na Univerzitě Pardubice.

3.5. ORM a Entity Framework

ORM je technologie, která řeší převod dat mezi programovacím jazykem a objektově relační databází. ORM se stará o převod dat mezi dvěma nekompatibilními systémy jako jsou například programovací jazyk C# a Microsoft SQL Server nebo Java a MySQL. Z pohledu programátora jsou pak jednotlivé tabulky reprezentovány datovými třídami a jeden řádek v tabulce je reprezentována instancí datové třídy a naopak. Při použití ORM se programátor osvobozuje od ručního vytváření SQL dotazů, protože je za něj generuje ORM. Zde vzniká nevýhoda ORM, kdy vygenerované dotazy nemusejí být optimální a neefektivní. [18]

Entity Framework je framework implementující technologii ORM. Dále je Entity Framework open source, multiplatformní a lightweight[4] a je obsažen v .NET platformě. Jak již bylo řečeno, Entity Framework jakožto implementace ORM zprostředkovává převod dat mezi databází a programovacím jazykem. Díky tomu se snižuje riziko vzniku chyb při manuálním psaní SQL dotazů a zároveň Entity Frameworkem, nebo jakýkoliv jiný ORM framework, poskytuje vyšší úroveň abstrakce, což zjednodušuje práci. Použití Entity Frameworku také do jisté úrovně snižuje potřebné množství kódu pro provedení akcí s daty jako jsou: načtení dat z databáze a uložení dat do databáze. [19], [20]

Pro použití Entity Frameworku jsem se rozhodl hlavně z důvodu použití .NET Core, který obsahuje tento ORM Framework. Navíc jen tento framework přehledný a jednoduchý na naučení se a použití. Dále je tento framework bezpečný na použití, protože automaticky provádí transakce při každém zásahu do databáze. Entity Framework je také lehce rozšiřitelný a v případě potřeby si programátor může lehce doimplementovat nebo upravit různé funkcionality.

3.6 Blazor

V dnešní době standartně existují webové aplikace, které se skládají z klientské a serverové části. Klientská část webové aplikace se typicky načte do prohlížeče uživatele, kde jsou vykonávány jednoduché funkce aplikace. To umožňuje webové aplikaci fungovat i offline v omezeném režimu[5]. Pro tvorbu klientské části webové aplikace se dnes nejčastěji používají knihovny a frameworky založeny programovacím jazyku JavaScriptu. Mezi takové patří například Angular, React nebo Vue. Pro server část se používají programovací jazyk jako je C#, Java, Python a další.

Zde vzniká problém, kdy je pro vývoj webových aplikací vyžadována znalost dvou a více programovacích jazyků pro psaní kódu na serverové části a pro psaní kódu na klientské části aplikace. Z tohoto důvodu se programátoři musí naučit programovací jazyk pro serverovou i klientskou část, a to stojí čas i úsilí. Proto vznikl framework Blazor, který umožňuje psaní klientské části aplikace v programovacím jazyku C#. Díky tomu se programátoři nemusí učit odlišné programovací jazyky pro serverovou a klientskou část aplikace. Tato výhoda

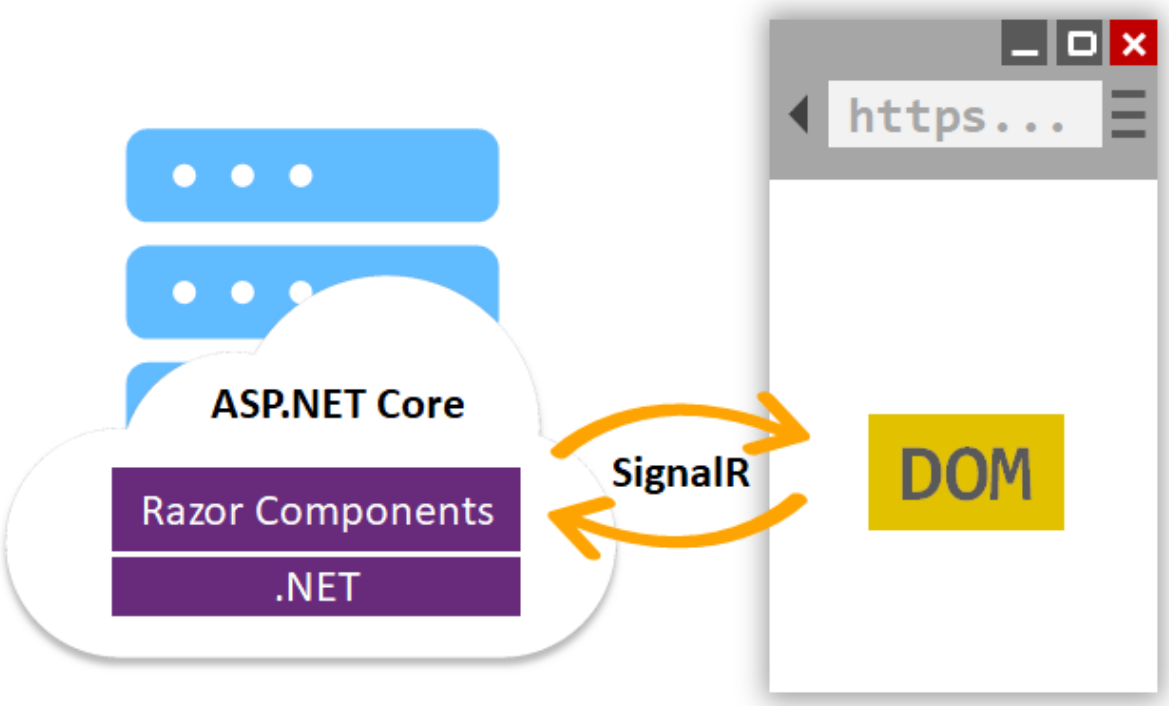
[4] Jednoduchý na použití a neobsahuje nadměrné množství metod a funkcí.
[5] Některá data nebudou aktuální a funkce implementované na serverové části nebudou k dispozici.

samozřejmě platí pouze pro programátory, kteří ovládají programovací jazyk C# nebo se ho chystají naučit.

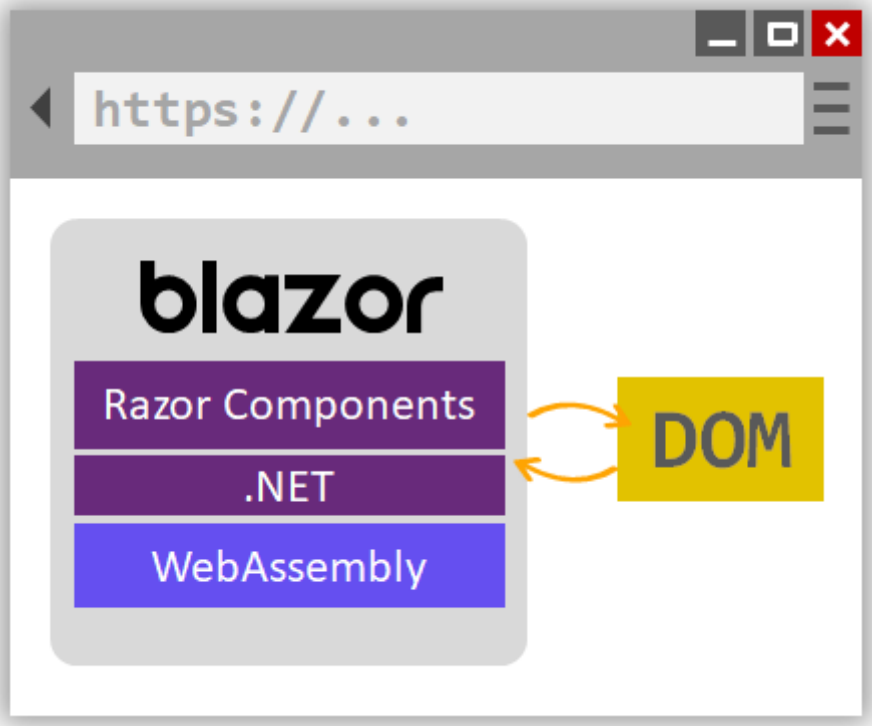
Blazor je framework určený pro vývoj single page[6] webových aplikací. Jelikož je Blazor postaven na .NET architektuře, tak lze vytvářet klientské části aplikace za použití programovacího jazyka C# namísto JavaScriptu a zároveň umožňuje použití .NET knihoven na klientské straně aplikace. [21]

Blazor aplikace se spouští buď na serverové části nebo přímo v prohlížeči uživatele. V případě možnosti běhu Blazor aplikace na serveru je zdrojový kód překládán a spouštěn na serveru a prohlížeč slouží pouze jako terminál pro zobrazování výsledků, nebo zadávání vstupů. Nevýhodou je nutnost udržování nepřetržitého spojení mezi serverem a klientem. Každý klient má vlastní instanci aplikace na serveru. Výhoda běhu kódu na serveru je určitě bezpečnost aplikace. Naproti tomu běh Blazor aplikace na klientské části spouští aplikaci přímo ve webovém prohlížeči uživatele. Při spuštění aplikace dojde ke stáhnutí zdrojového kódu, různých závislostí a .NET runtimu do prohlížeče, kde je následně spuštěn za pomoci WebAssembly. Klientská část aplikace funguje většinu času bez nutnosti komunikovat se serverem a ke komunikaci se serverem skrze WEB API, dochází pouze na vyžádání. Toto chování je velmi podobné aplikacím, které jsou vytvořeny v Reactu nebo Angularu. [21]

[6] Webová aplikace je rozdělena na komponenty, u které dochází k pře-renderování pouze komponent, ve kterých došlo k nějakým změnám. To vede k rychlejší odezvě webové aplikace.



Obrázek 2 – Blazor aplikace běžící na serveru [21]



Obrázek 3: Blazor aplikace běžící v prohlížeči [21]

Právě díky WebAssembly je možné spouštět kód napsaný v Blazoru přímo v prohlížeči. Přesněji řečeno WebAssembly překompiluje napsaný kód v Blazoru do nízko-úrovňového jazyka zvaný assembler, který je následně dále překompilován do binární podoby, která je spustitelná na všech moderních webových prohlížečích. Díky WebAssembly je možné spouštět v prohlížečích kód napsaný i v jiných programovacích jazycích než C#. [22]

Velkým důvodem použití tohoto framework je, že se jedná o slibnou alternativu ke knihovnám a framworkům založených na JavaScriptu, který umožňuje vytváření klientských částí webových aplikací za použití programovacího jazyka C#. Právě možnost psát logiku klientské části aplikace v jazyce C# je pro mě velkým usnadněním a zpříjemnění práce, i když do určité míry ovládám a rozumím jazyku JavaScript, který se i tak plánuji plně naučit. Posledním důvodem použití tohoto frameworku je kvalita a funkčnost microsoftího ekosystému, kdy jednotlivé technologie od Microsoftu spolu velmi dobře spolupracují a společně tvoří funkční prostředí.

3.7 React

Podobně jako Blazor je React technologie pro vývoj single page aplikací (SPA). Zde se ale jedná o JavaScript knihovnu, nikoliv framework, vyvíjený společností Facebook a komunitou individuálních vývojářů a firem. První verze nebo spíše prototyp Reactu (v té době FaxJS) byl vytvořen v roce 2011 jako komponenta Novinky na Facebooku. První oficiální vydání Reactu pro veřejnost se uskutečnilo až v roce 2013 a šlo o verzi 0.3.0. V dnešní době jde o jednu z nejpopulárnějších JavaScript knihoven pro vývoj webových aplikací. [23]

Rozdíl mezi knihovnou a frameworkem je ten, že framework má spoustu funkcionalit zabudovaných v sobě, kdežto knihovna obsahuje funkce jen na určitou věc. Pokud jde o čistý React, tak se jedná pouze o knihovnu, která nám umožňuje tvořit UI. React sám o sobě neobsahuje funkce pro routování, komunikaci se serverem, neumí držet jeden celistvý stav aplikace atd. Pro vytvoření kompletní webové aplikace se musí použít další knihovny, než jen samostatný React a to může být jak výhoda tak i nevýhoda. [23]

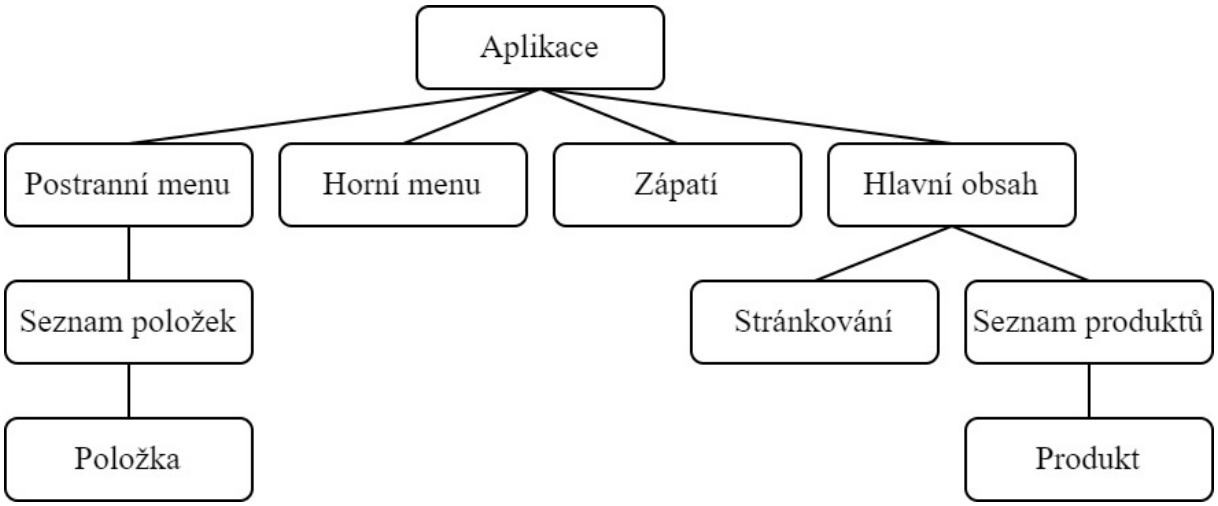
Pro nezkušené programátory jde určitě o nevýhodu, protože existuje spousta knihoven, které řeší stejný problém. Nezkušený programátor nemá zkušenosti a znalosti na to, aby byl schopen zvolit vhodnou knihovnu. Musí brát v potaz, zdali je knihovna stále spravována, zdali k ní existuje dostatečně obsáhlá dokumentace, zdali je vhodná pro řešení daného problém apod. Právě proto je velkou výhodou začít s vývojem ve frameworku, který v základu obsahuje velké

množství požadovaných funkcionalit. Programátor tak nemusí řešit, jaké knihovny má ještě obstarat, nemusí pročítat dokumentaci ke každé knihovně zvlášť apod. Stačí mu pouze nastudovat návody, kterých je pro populární frameworky nespočet a které většinou popisují krok po kroku vývoj webové aplikace. Dále jsou pak frameworky lépe udržovány a častěji aktualizovány. Nehrozí, že by ze dne na den skončila pro daný framework podpora. V případě zkušeného programátora, který se v tomto prostředí umí pohybovat nejde o nic složitého, a proto možnost, si zvolit knihovnu, kterou bude používat, je určitě výhoda.

React funguje na základě vytváření komponent, které jsou na sobě nezávislé, izolované a znovupoužitelné. Tyto komponenty se následně mezi sebou kombinují a tím vytvářejí interaktivní uživatelské rozhraní. Každá React aplikace má nejméně jednu root komponentu, která obsahuje zbylé komponenty. Výsledkem je pak strom komponent, které mezi sebou komunikují. Jednotlivé komponenty reprezentují část UI, které společně vytváří interaktivní webovou aplikaci. [24]

Název React odráží funkcionalitu této knihovny. Slovo react znamená v češtině reagovat, a to přesně React dělá. React si udržuje virtuální kopii Document Object Model elementu zvanou Virtual DOM, která je levná[7] na vytvoření. React následně, při každé změně stavu komponenty, porovná virtuální DOM s reálným DOMem a aktualizuje pouze tu část, kde došlo ke změně. Toto způsobuje, že nedochází k obnovení nebo aktualizaci celé webové stránky, ale pouze se aktualizuje změněná část na webové stránce. Díky tomu lze vytvářet SPA aplikace, které jsou mnohem rychlejší než klasické webové aplikace, u kterých při nějaké změně dochází, k znovunačtení a zobrazení obsahu celé stránky.

[7] Není náročná na výpočetní výkon procesoru ani paměť.



Obrázek 4 – Příklad rozložení komponent v Reactu

Horní komponenta je zmíněná root komponenta, která zastřešuje všechny ostatní komponenty. Komponenty jako například Stránkování lze znovupoužít na jiných místech v aplikaci, nebo dokonce v úplně jiných aplikacích za předpokladů, že jsou správně[8] implementovány. Znovupoužitelnost komponent je velmi užitečná a práci usnadňující vlastnost.

Početná komunita React vývojářů a velké množství návodů, rad a tipů jsou hlavnímy důvod, proč bude použita tato JavaScriptová knihovna pro impelmentaci klientské části webové aplikace. Dále pak její oblíbenost, snadná rozšiřitelnost (pomocí mnoha ověřených komponent) a do určité míry i jednoduchost, jsou dalšímy důvody použití této technologie. V neposlední řadě jsem byl s Reactem seznámem na předmětu Programování internetových aplikací, vyučovaný na Univerzitě Pardubice a z toho důvodu jsem se také rozhodl pro pužití této JavaScript knihovny.

3.8 JWT

Celým názvem JSON Web Token je standard používaný pro zabezpečenou výměnu informací v JSON formátu mezi dvěma stranami. Bezpečnost a důvěryhodnost přenášených informací je zajištěna digitálním podpisem. JWT se skládá ze tří částí oddělených tečkou: hlavička, data a digitální podpis. Model JWT tokenu je tedy následující: xxx.yyy.zzz. [25], [26]

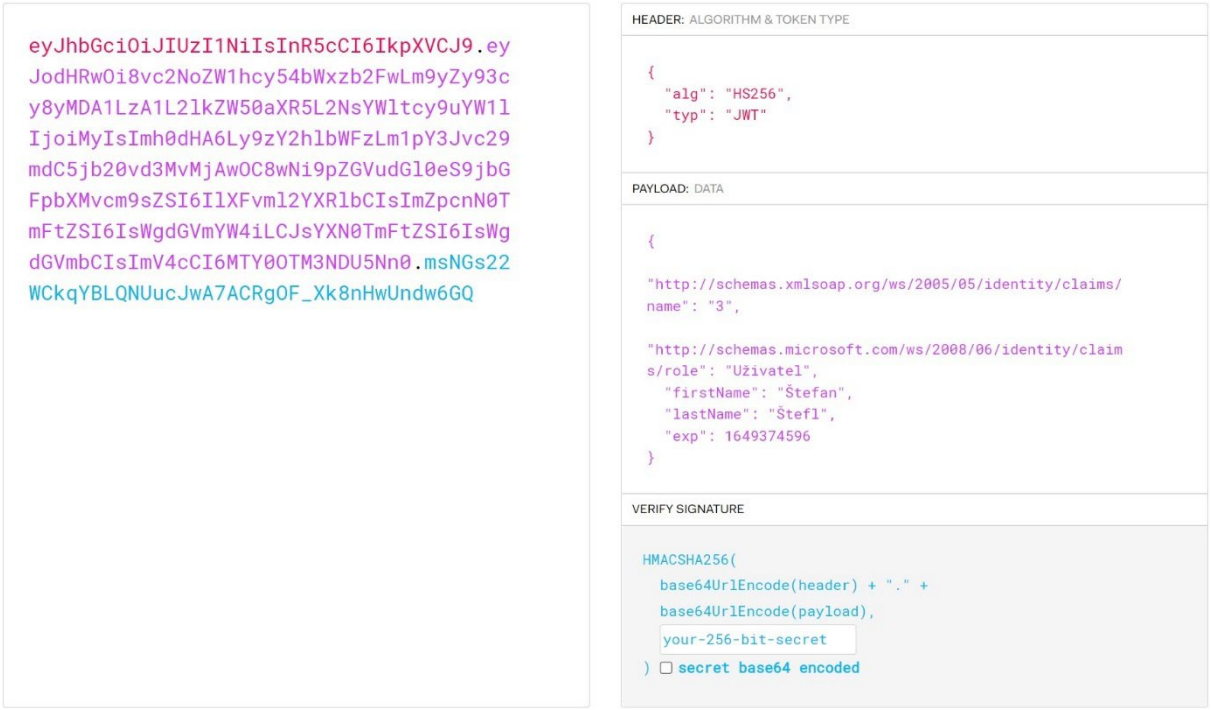
[8] Napsaný kód není vytvořený tak, aby fungoval jen pro specifické požadavky aplikace.

Hlavička se typicky skládá ze dvou částí. Jedna část obsahuje informaci, o jaký typ tokenu se jedná (JWT, SAML, SWT …) a druhá část hlavičky obsahuje informaci o použitém hashovacím algoritmu (HMAC, SHA256, RSA …). [25], [26]

Další částí JWT je obsahová část, která obsahuje tzv. claims. Claims popisují vlastnosti či informace o entitě. Entitou se většinou rozumí uživatel, který přistupuje do systému nebo aplikace komunikující s jinou aplikací. Claims se dále dělí na tři typy: reserved, public a private. Reserved je předdefinovaná kolekce, která není povinná, ale doporučuje se její použití. V reserved kolekci se nachází např.: iss (vydavatel/poskytovatel JWT), exp (doba platnosti JWT), sub (předmět JWT), aud (pole příjemců JWT) a další. Public a private jsou obojí soukromé claims definované programátorem, které jsou specifické pro danou aplikaci. Programátor definuje název claimu, kterému přiřadí hodnotu. Příkladem je claim s názvem UserName, který obsahuje uživatelské jméno uživatele. Jediný rozdíl mezi public a private je, že public claims musí mít jedinečné názvy. [25], [26]

Poslední částí JWT je digitální podpis, který slouží k ověření, že JWT nebylo změněno. Podpis se skládá z hlavičky, datové části JWT a tajného klíče. Defaultně se pro vytvoření podpisu používá symetrický hashovací algoritmus HS256, který pro šifrování a dešifrování používá jeden privátní klíč. Pomocí tohoto hashovacího algoritmu dojde k vytvoření digitální podpisu. Protože klient nějakým způsobem neupravuje JWT, tak nepotřebuje znát privátní klíč, který by jinak musel použít k vytvoření validního podpisu. [25], [26]

Při sjednocení všech výše zmíněných částí vzniknou tři řetězce oddělené tečkou, které lze snadno posílat/předávat a zahrnout do HTML a následně poslat v hlavičce pomocí HTTP protokolu. Příklad zašifrovaného a rozšifrovaného JWT lze vidět na přiloženém obrázku (obrázek 5).

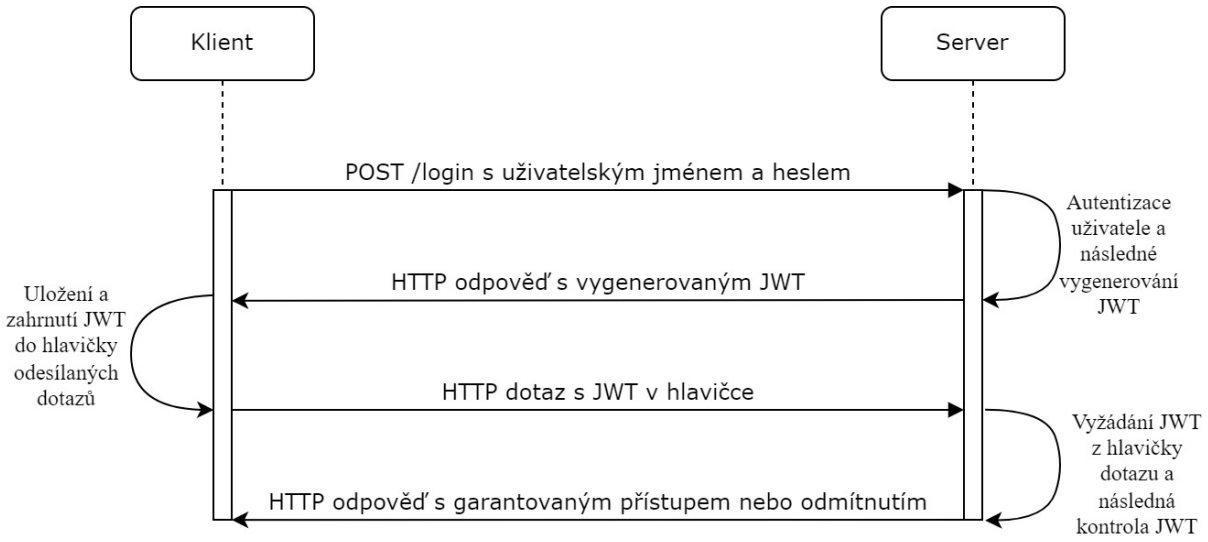


Obrázek 5 – JWT

JWT slouží pouze k autorizaci uživatele, nikoliv k jeho autentizaci. O autentizaci uživatele se stará proces přihlášení, při kterém dojde k ověření přihlašovacího jména a hesla. V případě existujícího přihlašovacího jména v databázi a odpovídajícího hesla, je uživatel úspěšně autentizován. Při úspěšné autentizaci, kdy se uživatel prokáže správný zadáním přihlašovacího jména a hesla, dojde k vygenerování JWT, který je poté serverem vrácen uživateli. Na klientské straně se obdržený token uloží do local storage nebo cookies a při každém dotazu na server se k dotazu připojí onen vygenerovaný JWT token. Typicky se JWT vkládá do autorizační hlavičky dotazu dle schéma: Authorization: Bearer <token>.

Obrázek 6 – Ukázka JWT v HTTP hlavičce

Tomuto procesu se říká bez-stavový autorizační mechanismus, protože nedochází k ukládání a držení informací o přihlášeném uživateli do paměti na serveru. Tento bez-stavový autorizační mechanismus funguje následovně (obrázek 7). Při jakémkoliv přístupu klienta na zabezpečené adresy dojde k ověření JWT tokenu serverem a v případě validního JWT je klientovi umožněný přístup, tedy dojde k autorizaci uživatele na základě jeho JWT. JWT tokeny jsou se označují jako soběstačné, protože obsahují veškeré nezbytně nutné informace pro autorizaci a díky tomu snižují nároky na výpočetní výkon oproti jiným metodám, kdy je např. nutnost držet data v paměti, provádět větší množství dotazů na databázi apod. [26]



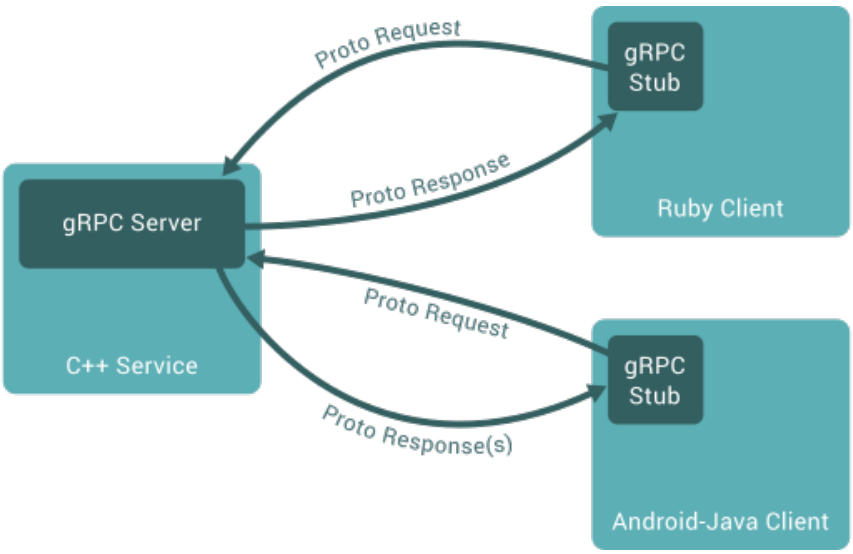
Obrázek 7 – Komunikace mezi klientem a serverem

Princip fungování JWT umožňuje ukládání nezbytně nutných dat pro autorizaci uživatele na straně klienta. Token je navíc šifrovaný a obsahuje ověřovací podpis, díky kterému server pozná, zdali nedošlo k editaci tokenu, a tedy je ten způsob autorizace uživatele bezpečný. Tyto dvě vlastnosti způsobu autorizace jsou hlavním důvodem, proč jsem se rozhodl použít JWT pro řešení autorizace ve webové aplikaci.

3.9 gRPC

RPC je protokol pro řešení komunikace mezi aplikacemi založených na klient-server modelu. Protokol funguje na principu volání servis ze vzdáleného serveru skrze síť bez detailního popisu sítě. Volání vzdálených servis je prováděno jako-by se vzdálené servisy na serveru nacházely lokálně u klienta. Stejně jako volání lokálních metod i volání vzdálených metod je synchronní proces při kterém dojde k pozastavení klienta do té doby, než obdrží odpověď od serveru. Pro specifikování API serveru se používá Interface Definition Language, který zprostředkovává komunikaci mezi klientem a serverem. [27]

gRPC je framework implementující vzdálené volání procedur vyvinutý společností Google jako novější generace RPC infrastruktury. Jedná se o moderní technologii, která využívá HTTP/2 protokol pro přenos dat, obsahuje pokročilou serializaci dat[9] a poskytuje funkce jako jsou: autentizace dat, obousměrné posílání zpráv, trasování nebo kontrola přehlcení při častém posílání zpráv/požadavků. Výhodou serializování zpráv do binární podoby je jejich malá velikost a vysoká rychlost přenosu. Na druhou stranu nejsou tyto zprávy čitelné člověkem, a to má za následek ztížení debugování. Dále gRPC podporuje velkou škálu programovacích jazyků a je nezávislé na zvolené platformě nebo operačním systému. Díky podpoře mnoha programovacích jazyků je možné mít servisy na serveru napsané např. v jazyce C++ a poskytovat služby klientům vyžívající jazyky: C#, Java, Python a další. [28], [29]



Obrázek 8 – gRPC komunikace klient-server různých platforem [30]

Technologie gRPC nabízí vysokou úroveň zabezpečení díky využívání HTTP/2 protokolu, který přenášená data mezi klientem a serverem šifruje a využívá TLS nebo SSL certifikáty. Dále poskytuje mnohonásobně vyšší rychlost při přenosu dat oproti klasickému RESTu a to díky rychlé serializaci dat jak na straně klienta tak i na straně serveru. V neposlední řadě je gRPC multiplatformní s podporou populárních programovacích jazyků jako jsou: Java, C#, PHP, Python atd. Jelikož se ale stále jedná o poměrně novou technologii uvolněnou v roce 2015, tak stále není 100 % podporována všemi internetovými prohlížeči. gRPC jako takové nenahrazuje REST. Jedná se především o velmi slibnou alternativu k RESTu a výběr mezi

[9] Posílané zprávy jsou serializovány do binární podoby.

těmito technologiemi záleží na podmínkách a požadavcích aplikace. gRPC jsem zvolil pro implementaci API administrátorského rozhraní aplikace, protože se chci s touto technologií blíže seznámit a porovnat ji s RESTem, který bude použit pro implementaci API uživatelského rozhraní.

3.10 REST API

Application Programming Interface je množina funkcí, knihoven, procedur a protokolů kterou programátor využívá v rámci tvorby aplikací. Účelem API je zprostředkování komunikace mezi dvěma platformami, které spolu komunikují a vyměňují si data. Díky API lze aplikaci rozšířit o nové funkcionality, díky čemuž šetří čas i peníze, protože programátor nemusí dané funkcionaity implementovat přímo pro danou aplikaci. API se netýká pouze webových aplikací, ale i běžných desktopových aplikací, které například skrze API komunikují s operačním systémem. [31]

Representational State Transfer je koncept pro design architektury rozhraní, který stanovuje určitá specifika a standardy při vytváření těchto rozhraní. REST jako takový je orientován na data, nikoliv na volání procedur jako například zmíněný framework gRPC. Pro přenos dat se zde využívá HTTP protokol, ale lze použít i jakýkoliv jiný protokol či metodu přenosu. Důvodem použití HTTP protokolu je ten, že HTTP protokol je dnes natolik používány a rozšířený že se málokdy vyplatí použití jiného způsobu přenosu dat. Jedním ze specifik RESTu je, že u HTTP protokolu by měl programátor využívat celou škálu nabízených metod, kterými jsou: GET, POST, DELETE a další. Dále REST stanovuje použití vícero tzv. přístupových bodů neboli rout. Pomocí přístupových bodů se následně přistupuje ke zdrojům, datům nebo stavům aplikace. Příklad koncových bodů: GET /products – při přístupu na takový koncový bod se očekává kolekce produktů, GET /product/1 – zde se očekává pouze jeden produkt s identifikačním číslem 1. Všechny názvy přístupových bodů by měly dodržovat předem stanovený standard pojmenování kvůli přehlednosti. U jednotlivých přístupových bodů je také nutné stanovit, která metoda HTTP má být použita při přistupování na daný koncový bod. [32]

Nejdůležitějším specifikem je udržet celé rozhraní bez-stavové. To znamená, že každý přístup na koncový bod musí obsahovat nezbytné informace pro jeho vykonání. Server si nesmí držet informace z předešlých požadavků a následně je použít při zpracování nově příchozích požadavků. Příkladem je ověření uživatele, kdy požadavek musí obsahovat nutná data pro ověření a rozpoznání, kdo daný požadavek poslal. [32]

Při dodržení těchto pravidel a standardů se vytvořené rozhraní nazývá jako RESTful API. V dnešní době jde o jeden z nejrozšířenějších způsobů/standardů vytváření webového rozhraní pro zajištění komunikace mezi různými platformami. [32]

REST patří mezi nejpoužívanější způsoby implementace API aplikace a je mnohými považován za standart při vytváření API aplikací. Právě oblíbenost, podpora a jednoduchost jsou jedny z hlavních výhod RESTu. Dále pak flexibilita, která dovoluje různým klientům vracet data ve formátech: JSON, XML, YAML atd., nebo jednoduchost vytváření a poskytování přístupových bodů, kdy stačí znát URL adresu nebo případě vlastnit API klíč, který většinou poskytuje tvůrce API. Jednoduchost, a především univerzálnost RESTu jsou důvody, proč jsem se rozhodl tuto technologii použít pro implementaci API uživatelského rozhraní.

4 ALTERNATIVNÍ TECHNOLOGIE

Tato kapitola se zaměřuje na popis technologií, které se dají brát jako alternativy k použitým technologiím, zmíněných v předešlé kapitole.

4.1 Spring Framework

Spring je open-source framework primárně užívaný pro tvorbu mikro servis a webových aplikací. Jde o jeden z nejpopulárnějších Java frameworků podporující jazyky Java, Kotlin a Groovy, za kterým stojí obrovská komunita vývojářů poskytující zpětnou vazbu ohledně reálných případů užití. Díky této zpětné vazbě byl framework Spring schopen se vyvinout do dnešní podoby. [33]

Podobně jako ASP.NET Core i Spring nabízí funkční základ, který lze jednoduše rozšiřovat pomocí tzv. modulů ze kterých se Spring framework skládá. Díky tomu se jedná o velmi flexibilní framework, který si mohou programátoři upravit tak jak potřebují. Programátoři se také nemusí zabývat vymýšlením řešení různorodých problémů, protože velké množství doplňujících knihoven a modulů s přehlednou dokumentací obsahují potřebné funkcionality, díky kterým je vývoj rychlejší, jednodušší a pohodlnější. Zabudovaný LiveReload[10] šetří o to víc čas, kdy programátor nemusí při každé změně kódu restartovat celou aplikaci, který nějaký ten čas trvá.

Kvůli veliké oblíbenosti frameworku existuje obrovská komunita, která pomáhá s rozvojem frameworku, tvoří různé návody a řeší společné, resp. běžné problémy při implementaci různých aplikací. Pro začátečníka je tak velice snadné začít s tímto frameworkem pracovat.

4.2 Java

Jedná se o objektově orientovaný programovací jazyk vyšší úrovně[11]. Java byla navržena tak, aby byl zkompilovaný kód spustitelný na všech platformách, které podporují Javu, bez nutnosti opakované kompilace. Syntaxe jazyka je podobná jazykům C a C++, které sloužili jako počáteční inspirace při vytváření. Na rozdíl od C++ je ale Java pouze objektově orientovaná, načež C++ obsahuje kombinaci syntaxe pro generické, strukturované a objektově orientované

[10] Promítání změn do webového prohlížeče v reálném čase při změně zdroje.
[11] Syntaxe čitelná a pochopitelná člověkem.

programování. Dále pak Java neumožňuje vícenásobnou dědičnost, kvůli přehlednosti a jednoduchosti, a přetěžování operátorů.

Java začala vznikat již v roce 1991, kdy její tvůrce James Gosling nebyl spokojený s programovacím jazykem C++ a tak začal tvořit vlastní programovací jazyk s názvem Oak (dub). V roce 1995 došlo k přejmenování programovacího jazyka na Java, protože jazyk s názvem Oak již existoval. Pří výběru nového názvu programovacího jazyka, chtěl vývojářský tým takový název, který by symbolizoval revolučnost, dynamiku, unikátnost, jednoduše se psal a snáze vyslovoval. Z několika návrhů byl nakonec zvolen název Java.

Při vývoji Javy se vývojáři řídili pěti zásadami. První z těchto zásad říká, že to[12] musí být jednoduché, objektově orientované a povědomé[13]. Druhá zásada říká, že by to mělo být robustní a bezpečné. Dále pák, že by to mělo být nezávislé na architektuře a přenositelné. Čtvrtá zásada říká, že by to mělo být výkonné a jako poslední zásada by to mělo být interpretované, dynamické a podporovat více vláken.

V dnešní době je Java jedním z nejpopulárnějších jazyků na světě. Java je velice populární v odvětví vytváření android aplikací, webových aplikací, a dokonce i ve velké většině nahrazuje programovací jazyk C++, který je sice výkonnější, ale Java nabízí větší bezpečnost, přenositelnost a udržovatelnost. C++ ale stále dominuje ve specifických případech užití.

4.3 Vue.js

Vue je open source JavaScript framework i knihovna pro vytváření uživatelských rozhraní nebo single page aplikací. Vue vytvořil Evan You, kdy v roce 2013 započal vývoj Vue a v roce 2014 byla vydána první verze. Vue je inspirováno JavaScriptovým frameworkem zvaným AngularJS od firmy Google. Cílem při vytváření Vue bylo vzít to nejlepší z frameworku Angular a vytvořit jednoduchý a přehledný framework.

Velikou předností Vue je možnost jej využít pouze jako doplňkovou knihovnu v určitých částech projektu, nebo jej využít jako komplexní framework pro vytváření single page aplikací. Vue se může skládat pouze z přístupné základní knihovny, která se soustředí pouze na

[12] To, myšleno jako programovací jazyk.
[13] Syntaxe jazyka podobná jiným syntaxím.

zobrazovací vrstvu, nebo lze využít kompletní ekosystém Vue pro komplexní vytváření a spravování webových aplikací. [34]

4.4 PHP

PHP patří k nejrozšířenějším open-source programovacím jazykům pro vytváření webových aplikací. PHP je nezávislé na volbě operačního systému, velice jednoduché na naučení a pochopení a díky své popularitě existuje veliké množství návodů a řešení obvyklých problémů. Na rozdíl od zmíněného Blazoru nebo Reactu, běží PHP na straně serveru nikoliv na straně uživatele. To znamená, že všechny napsané operace v PHP nejsou prováděny v uživatelském prohlížeči či PC, ale na straně serveru a výsledná data ve formě HTML jsou po zpracování odeslána uživateli. [35]

Jak již bylo zmíněno, díky své veliké popularitě existuje mnoho rozšiřujících knihoven pro práci s databází, soubory, zpracování textu, či grafiky a v neposlední řadě také podpora velkého množství internetových protokolů jako jsou HTTP, SMTP, FTP atd. PHP se nejčastěji využívá v kombinaci s operačním systémem Linux, webovým serverem Apache a databázovým systémem MySQL. Toto spojení se označuje zkratkou LAMP, která reprezentuje kombinaci všech čtyř zmíněných technologií pro vývoj webových aplikací.

PHP je oproti jiným programovacím jazykům velmi jednoduché na naučení se. Důvodem je jednoduše čitelná, logická a dobře organizovaná syntaxe PHP. Další výhodou PHP je jeho kompatibilita. PHP vzniklo již v roce 1995 a stále se jedná o velmi populární programovací jazyk pro vytváření webových aplikací, díky tomu je PHP kompatibilní s většinou operačních systémů nebo různými technologiemi jako jsou například Java, JavaScript atd. Největší výhodou PHP je fakt, že je kompletně zdarma a zároveň i kompletní ekosystém pro vytváření webových aplikací, který se skládá z operačního systému Linux, webového serveru Apache, databáze MySQL a PHP.

4.5 SOAP

Simple Object Access Protocol je protokol určený pro výměnu zpráv nebo přístup webových služeb za pomoci HTTP protokolu. SOAP, podobně jako gRPC, funguje na principu vzdáleného volání procedur. Komunikační postup v případě SOAPu je zaslání požadavku ve formě zprávy strukturované ve formátu XML na server, který vykoná příslušnou službu a zpět zašle odpověď s výslednými daty nebo chybovou zprávu. Přenášená data jsou pouze ve formátu XML, který je podobný značkovacímu jazyku HTML. Formát XML byl zvolen především

kvůli jeho rozšířenosti a dostupnosti různých vývojových nástrojů. Další výhodou XML je jeho syntaxe, která je snadno čitelná člověkem. Naopak nevýhodou XML je nutnost parsování, které je z důvodu obsáhlejší syntaxe XML značně náročnější na výpočetní výkon a paměť. [36], [37]

Zpráva SOAP protokolu se skládá z tzv. obálky, která obaluje celý obsah zprávy a určuje začátek a konec zprávy. V obálce se dále nachází hlavička[14], ve které mohou být obsaženy autentizační údaje nebo specifika komplexních datových typů přenášených dat. Přenášená data mohou být v základních datových typech jako jsou string nebo int, ale i ve složených datových typech, které jsou specifikována v hlavičce zprávy. V neposlední řadě se ve zprávě vyskytují posílaná data, které obsahují buďto požadavek klienta na určitá data, nebo odpověď serveru, a tedy data určená klientovy. [36], [37]

```
<xsd:complexType>
    <xsd:sequence>
        <xsd:element name="Tutorial Name" type="string"/>
        <xsd:element name="Tutorial Description" type="string"/>
    </xsd:sequence>
</xsd:complexType>
```

Kód 1 – Ukázka SOAP dotazu na server [36]

```
<soap:Body>
    <GetTutorialInfo>
        <TutorialName>Web Services</TutorialName>
            <TutorialDescription>
                All about web services
            </TutorialDescription>
    </GetTutorialInfo>
</soap:Body>
```

Kód 2 – Ukázka SOAP odpovědi ze serveru [36]

V dnešní době je SOAP celkově na ústupu, kvůli jeho nedostatkům a existenci lepších variant pro řešení a implementování API. SOAP je mnohými považován za komplexní protokol a zprávy formátované v XML mohou velmi často nabývat větší velikosti, což vede ke zpomalení přenosu. Obecně je v dnešní době využití tohoto protokolu jen ve specifických případech, kde je např. potřeba využít jiný protokol pro přenos než HTTP. SOAP umožňuje

[14] Hlavička může být ve zprávě jen jedna.

přenos i za pomoci protokolu SMTP nebo FTP. Dále SOAP nabízí zabudované funkce pro řešení bezpečnosti, které velmi spolehlivě chrání zprávy před zneužitím. [38]

5 ANALÝZA A NÁVRH WEBOVÉ APLIKACE

Tato kapitola obsahuje analýzu firmy a její původní webové stránky. Dále návrh webové aplikace[15] na základě provedené analýzy a konzultace s firmou.

5.1 Analýza firmy

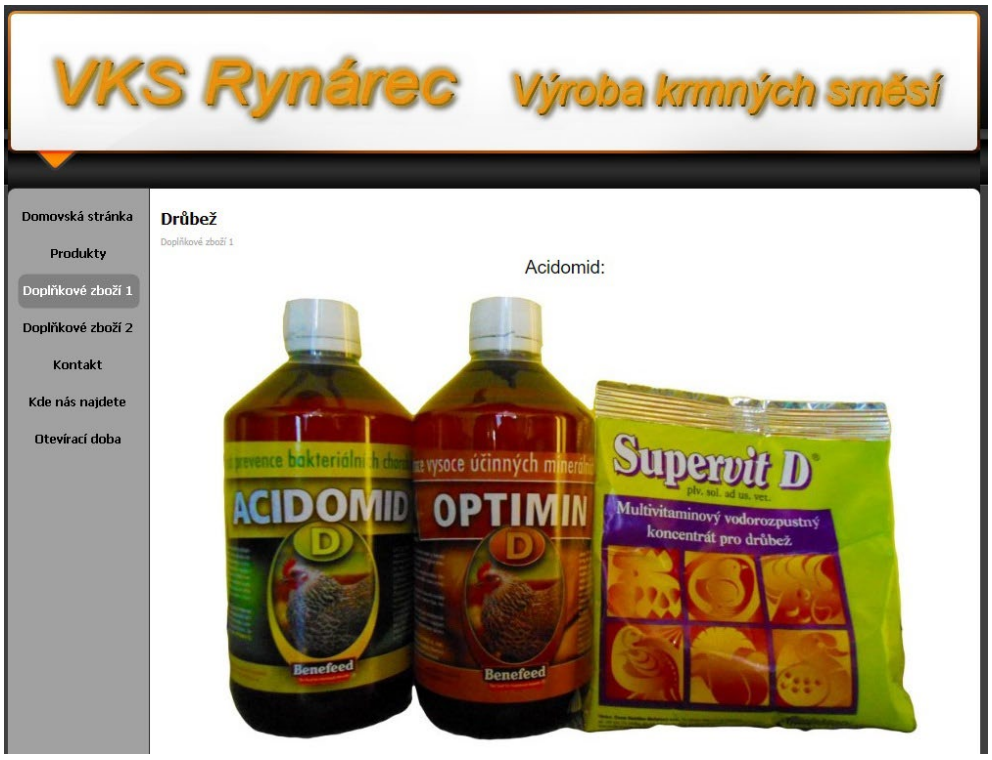
Firma VKS Rynárec byla založena v roce 1994 a jejich výrobní linka sídlí v Rynárci. Firma se specializuje na výrobu a prodej krmných směsí především pro malé a střední odběratele sídlící v blízkém okolí. Pro tyto odběratele nabízí firma rozvoz zdarma a množstevní slevu pět až deset procent při koupi krmných směsí v rozmezí váhy jeden a jeden a půl tuny. Součástí výrobní linky je i malá prodejna, kde lze i kromě vyrobených krmných směsí firmou také zakoupit různá krmiva, granule, pamlsky a hračky pro psy a kočky. Dále pak lizy, vitamíny a minerály pro koně, skot, ovce a další chovatelská zvířata. V neposlední řadě zde lze také zakoupit zeminy, rašeliny a substráty.

Firma od roku 2012 provozuje vlastní webové stránky. Jde o jednoduché webové stránky s postranním menu pro přepínání mezi jednotlivými stránkami. Z postranního menu lze přepnout na stránku pro zobrazení produktů, který zde není mnoho. Produkty jsou zobrazeny ve formě obrázku produktu a názvu. Jde pouze o zlomek produktů, které firma nabízí a u kterých není uvedena cena a popis. Nakonec poslední uvedené informace jsou kontakty firmy, její stručná historie, mapa se sídlem firmy a otevírací doba.

Závěrem jde o velmi jednoduché stránky, které sice obsahují nějaké nabízené zboží firmou, ale nejde o kompletní nabízeny sortiment. U produktů není uvedena cena a popis. Srovnáním se jedná o podobné webové stránky jako VKS Pohledští Dvořáci a.s., které jsou popsány v kapitole 2. Ne webu vkhsb.cz je však u jednotlivých produktů uvedena cena a krátký popis.

[15] Označení pro elektronický obchod z pohledu programátora.

Obrázek 9 – Ukázka webu VKS Rynárec, produkty



Obrázek 10 – Ukázka webu VKS Rynárec, doplňkové zboží

5.2 Konzultace s firmou

Při konzultaci s firmou se došlo k závěru, že elektronický obchod bude sloužit jako doplněk k jejich stávající kamenné prodejně. Firma se takto rozhodla, protože nemá žádné zkušenosti s provozováním vlastního e-shopu a dále si nemůže finančně dovolit provozovat nákladný e-shop, který nemusí okamžitě generovat zisky. Hlavní příjmy firmy budou nadále přicházet z prodejů v kamenné prodejně a z rozvozu krmných směsí zákazníkům. Tohoto přístupu se firma plánuje držet do té doby, než získá potřebné zkušenosti z provozování vlastního e-shopu a také, než začne e-shop přinášet znatelné množství nových zákazníků a zakázek. Firma je do budoucna ochotna vložit další a větší investici do e-shopu, pokud bude e-shop z části výdělečný. Pro prvotní verzi e-shop chce ale firma co nejnižší možné výdaje na zpracování a provoz.

5.2.1 Požadavky firmy na elektronický obchod

Požadavky stanovené firmou na elektronický obchod byly minimální, protože jak bylo výše zmíněno, firma nemá žádnou předešlou zkušenost s provozováním e-shopu. Veškeré požadavky se tedy týkaly financí, kdy firma požadovala, aby celkové náklady na tvorbu a zprovoznění elektronického obchodu nepřesáhly 20 000 Kč. Dále pak firma stanovila, aby měsíční náklady na údržbu nepřesahovaly 5 000 Kč. Jediné požadavky na funkcionalitu aplikace byly, zaprvé u jednotlivých krmných směsí firmy by mělo být zobrazeno složení a zadruhé neumožnit platbu dobírkou z důvodu možného nepřevzetí zboží zákazníkem. Další požadavky na e-shop firma neměla a byla otevřena návrhům autora této práce.

5.2.2 Funkční a nefunkční požadavky

Funkční požadavky popisují jednotlivé funkcionality systému. Na základě provedené analýzy firmy a konzultace s firmou jsou zde sepsány funkční požadavky na webovou aplikaci.

Funkční požadavky:

- E-shop musí obsahovat košík pro vkládání produktů.
- E-shop musí umožňovat vytvoření objednávky.
- E-shop musí zaslat emailovou notifikaci uživatele při vytvoření objednávky nebo při její změně.
- E-shop musí umožňovat vkládání, editování a mazání produktů a produkt kategorií.
- E-shop musí umožňovat správu objednávek.

- E-shop musí umožňovat vytváření uživatelských účtů.
- E-shop musí umožňovat přihlašování a odhlašování.
- E-shop musí umožňovat změnu hesla.
- E-shop musí umožňovat obnovení zapomenutého hesla.
- E-shop musí umožňovat filtrování, řazení a vyhledávání produktů.

Nefunkční požadavky:

- Grafické rozhraní e-shopu by mělo být atraktivní, přehledné a jednoduché.
- Obrázky produktů by měly být uchovávány v dostatečné velikosti a dobré kvalitě.
- E-shop by měl být vždy přístupný.
- E-shop by měl být dobře zabezpečený, aby byl odolný vůči známým metodám útoků.
- Česká lokalizace e-shopu.

5.3 Návrh řešení

Tato kapitola se věnuje analýze a návrhu implementace webové aplikace. U jednotlivých návrhů řešení jsou popsány a diskutovány jejich alternativy.

5.3.1 Use case diagram

Tento diagram slouží ke znázornění chování systému tak, jak ho vidí uživatel. Cílem diagramu je graficky zobrazit funkcionalitu systému, kterou po něm klient vyžaduje nebo očekává. Use case diagram v žádném případě nezkoumá, jak budou jednotlivé funkcionality systému řešeny a implementovány. Z tohoto důvodu se use case diagram vytváří na úplném začátku návrhu informačního systému, kdy je důležité upřesnění všech funkcionalit systému a až v další fázi návrhu řešit, jak budou jednotlivé funkcionality řešeny a implementovány. [39]

Use case diagram se skládá z aktéra a tzv. případu užití. Aktér je role, která nějakým způsobem interaguje s případem užití. Aktér reprezentuje uživatele nebo externí systém a dále se rozděluje na aktivního a pasivního aktéra. Aktivní aktér je aktér, který inicializuje nějaký případ užití, například uživatel vloží položku do košíku. Aktivní aktér se zakresluje do pravé části diagramu. Naproti tomu pasivní aktér je zpravidla inicializován případem užití. Příkladem je externí

server, který vytváří a odesílá emailové notifikace je iniciován případem užití Poslat email. Pasivní aktér se zakresluje do pravé části diagramu. [39]

Případ užití se skládá ze sady několika procesů, jejichž provedení vede k dosažení určitého cíle. Příkladem případu užití je například vytvoření objednávky, registrace uživatele nebo změna hesla. Případ užití tedy reprezentuje funkcionalitu, kterou by měl navrhovaný systém obsahovat. Tato funkcionalita se většinou dále skládá z dalších procesů, které se v diagramu neznázorňují. Například případ užití Změna hesla se dále skládá z procesu validace nově zadaného hesla a ověření původního hesla. Tato vnitřní logika se v use case diagramu nezachycuje, neboť to není pro uživatele či zákazníka podstatné. [39]

Aktéři:

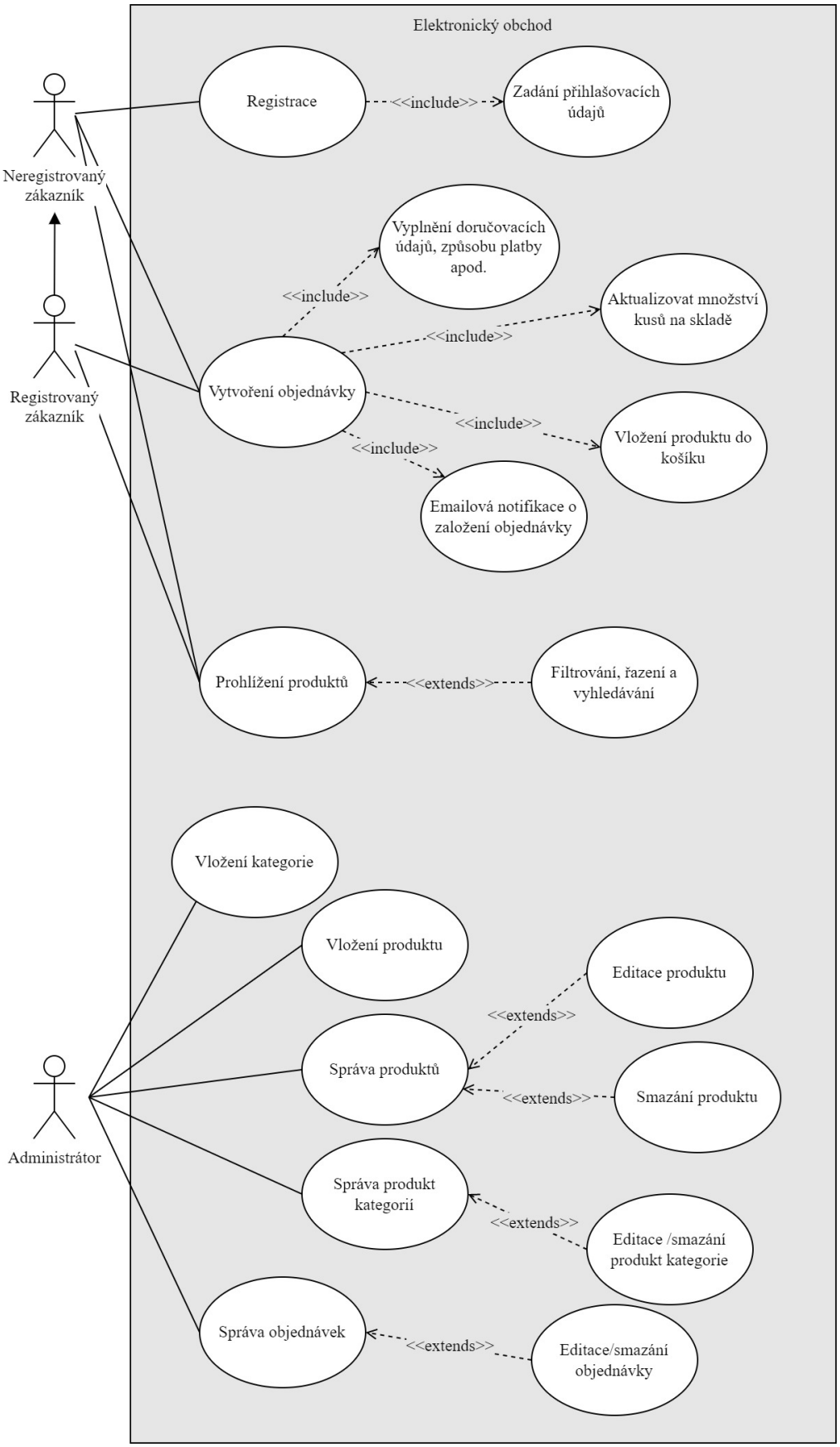
- Neregistrovaný zákazník
- Registrovaný zákazník
- Administrátor

Hlavní činnosti zákazníka

- Prohlížení produktů
- Vytvoření objednávky
- Filtrování produktů

Hlavní činnosti administrátora

- Spravování produktů
- Spravování kategorií
- Spravování objednávek



Obrázek 11 – Use case diagram

Přiložený Use Case diagram nezobrazuje veškeré scénáře případu užití, neboť některé případy užití jsou nepodstatné/nezajímavé a zároveň by výsledný diagram nemusel být čitelný a přehledný, pokud by obsahoval veškeré případy užití.

5.3.2 Datový model

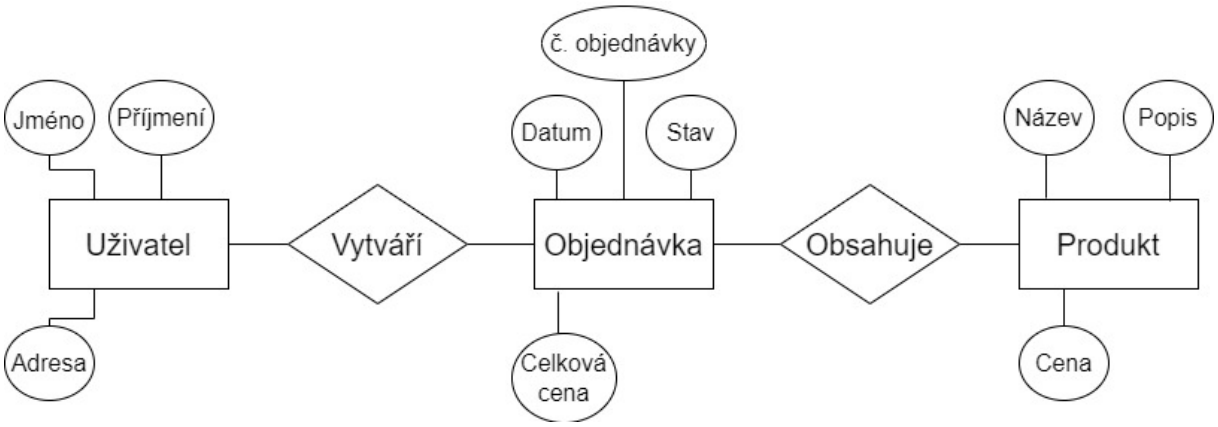
Datový model (DM) popisuje a znázorňuje data, se kterýma program nebo aplikace pracuje. DM se často využívá jako prostředek pro komunikaci mezi zákazníkem, který definuje požadavky na aplikaci a firmou či programátorem, který aplikaci vyvíjí. Proces vytváření DM je založeno na tzv. principu tří architektur (P3A). [40]

Princip tří architektur rozděluje proces tvorby DM na tři mentálně zvládnutelné části, které reprezentují různé úrovně abstrakce. Jednotlivé vrstvy se zaměřují na jeden z hlavních aspektů vyvíjené aplikace: obsah (konceptuální úroveň), technologie (technologická úroveň) a implementační specifika (implementační úroveň). Výsledkem každé ze tří vrstev je konkrétní model pro danou vrstvu. Vrstvy dále představují postup shora dolů, kde je model konceptuální úrovně nejobecnější a model implementační úrovně nejdetailnější. [40]

5.3.2.1 Konceptuální úroveň

Na konceptuální úrovni dochází ke znázornění a popisu informací (dat), se kterými bude vyvíjená aplikace pracovat. Veškeré informace se dělí na jednotlivé objekty (entity) a mezi jednotlivými objekty se dále popisují jejich vztahy (vazby). Každý objekt se dále skládá z vlastností (atributy). Výsledný model popisuje veškerá data a vazby mezi nimi tak, aby byla srozumitelná pro zákazníka. Nejedná se o popis chování a podobu dat přímo v počítači. [40]

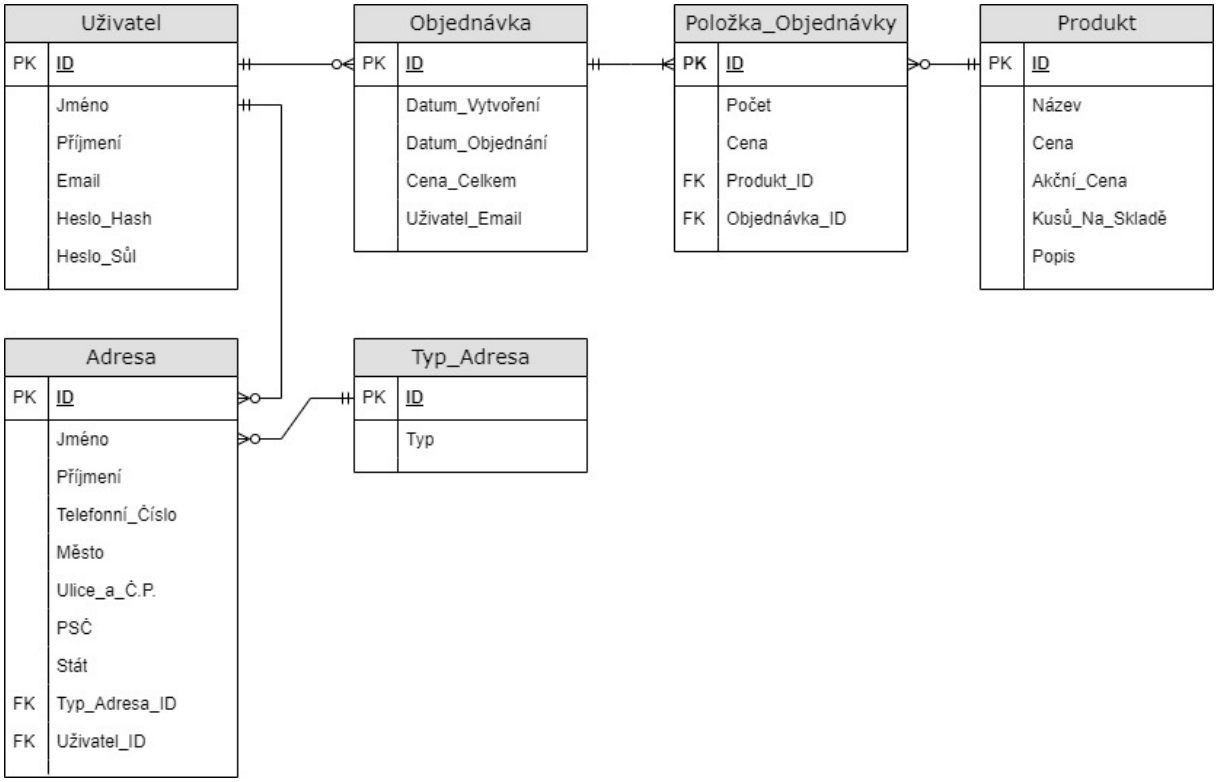
- Entita: uživatel, objednávka, produkt
- Vazba: vytváří, obsahuje
- Atributy: jméno, příjmení, adresa, č. objednávky, datum, stav atd.



Obrázek 12 – Modelu konceptuální úrovně

5.3.2.2 Technologická úroveň

Na konceptuální úroveň navazuje technologická úroveň, která konceptuální model dále rozšiřuje. Logický model se skládá z jednotlivých tabulek, které reprezentují entity. Tabulky dále obsahují názvy sloupců, kterým odpovídají názvy atributů každé entity. V každé tabulce jsou vyznačeny primární i cizí klíče a mezi tabulkami jsou detailněji znázorněny jejich vztahy mezi sebou. Z logického modelu lze implementovat výslednou podobu databáze ve zvolené technologii, avšak logický model je na zvolené technologii nezávislý. [40]



Obrázek 13 – Modelu technologické úrovně

5.3.2.3 Implementační úroveň

V poslední úrovni je vytvořen fyzický model systému dle zvolených technologií[16]. Předešlé nezávislé modely na technologii jsou v tomto kroku přetransformovány do fyzických modelů, které vyplívají z konkrétních specifik zvolených technologií. Podle fyzických modelů lze sepsat skripty pro vygenerování databáze, protože popisují detailní strukturu dat v aplikaci. Fyzický model si lze představit jako logický model, rozšířený o informace specifické pro zvolený databázový systém[17].

5.3.2.4 Popis datového modelu

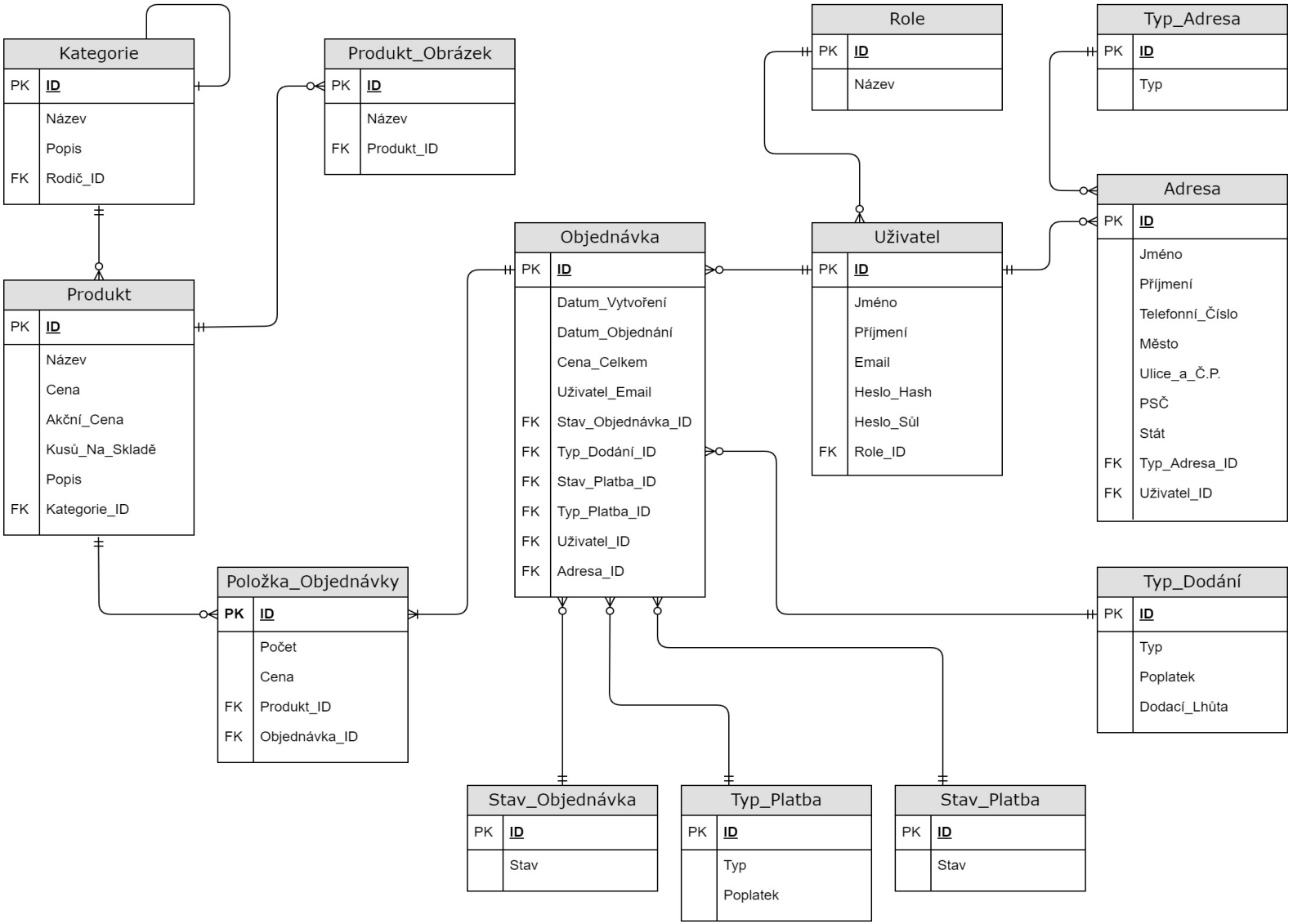
Návrhu logického datového modelu (LDM) je důležité věnovat dostatek času. Veškeré tabulky s argumenty a vazby mezi tabulkami by měly být detailně popsány a pečlivě promyšleny, aby se v pozdější fázi vývoje nenarazilo na nedostatky nebo jakékoliv chyby. Změny LDM jsou v pokročilejší fázi vývoje aplikace velmi náročné na realizaci a v případě rozsáhlejších projektů mnohdy ani nerealizovatelné. Z těchto důvodů vyplívá, že je navrhnutí LDM velmi důležité a mělo by se zde nic uspěchat, aby se v budoucí fázi vývoje neobjevily zbytečné problémy, kterým se dá předejít.

Při navrhování datového modelu došlo k přeskočení konceptuální úrovně. Model na konceptuální úrovni se primárně používá pro konzultace a pro stanovení přesné definice zadání před podepsáním smlouvy se zákazníkem. Zároveň není nutné vytvářet konceptuální model pro aplikace menších rozměrů. V případě této diplomové práce se jedná o aplikaci menších rozměrů a zároveň neproběhla detailní analýza požadavků se zákazníkem, ani vytvoření a podepsání smlouvy. Z těchto důvodů autor diplomové práce usoudil, že není potřeba se zaobírat tvorbou modelu konceptuální úrovně a rovnou přešel na tvorbu LDM.

Navrhování LDM se odvíjelo od požadavků na aplikace. Jak již bylo zmíněno v kapitole 5.2.1, firma stanovila minimum požadavků na systém, a proto byly další požadavky vytvořeny autorem diplomové práce dle znalostí a zkušeností. LDM byl navrhnut tak, aby byl šlo o optimální model pro tento daný projekt. Výsledná podoba LDM je znázorněna na obrázku 14.

---

[16] Jedná se o konkrétní technologie zvolené pro implementaci informačního systému.
[17] Specifické informace jako jsou datové typy atributů, vlastnosti sloupců atd.



Obrázek 14 – Datový model

Objednávka

Jedná se o hlavní entitu, která popisuje objednávky e-shopu.

Položka_Objednávky

Jedná se o tzv. vazební entitu, která realizuje vztah M:N[18] mezi entitou Produkt a Objednávka.

Produkt

Entita popisující a uchovávající produkty. Tato entita má vazbu na entitu Kategorie a Produkt_Obrázek.

Kategorie

Tato entita obsahuje kategorie produktů. Entita Kategorie má unární vazbu, která umožňuje přiřazení kategorii nadřazenou kategorii.

Produkt_Obrázek

Entita popisující a uchovávající vygenerované názvy pro jednotlivé fotky produktů.

Uživatel, Role

Entita Uživatel popisuje uživatelské účty e-shopu a je propojena s entitou Role, která obsahuje názvy uživatelských rolí.

Adresa, Typ_Adresa

Entita Adresa uchovává uživatelské adresy a entita Typ_Adresa obsahuje dva typy adres, kterými jsou: Doručovací a Fakturační.

Typ_Dodání, Typ_Platba, Stav_Objednávka, Stav_Platba

Všechny tyto entity se označují jako číselníky a obsahují statická data, která popisují nějaký stav.

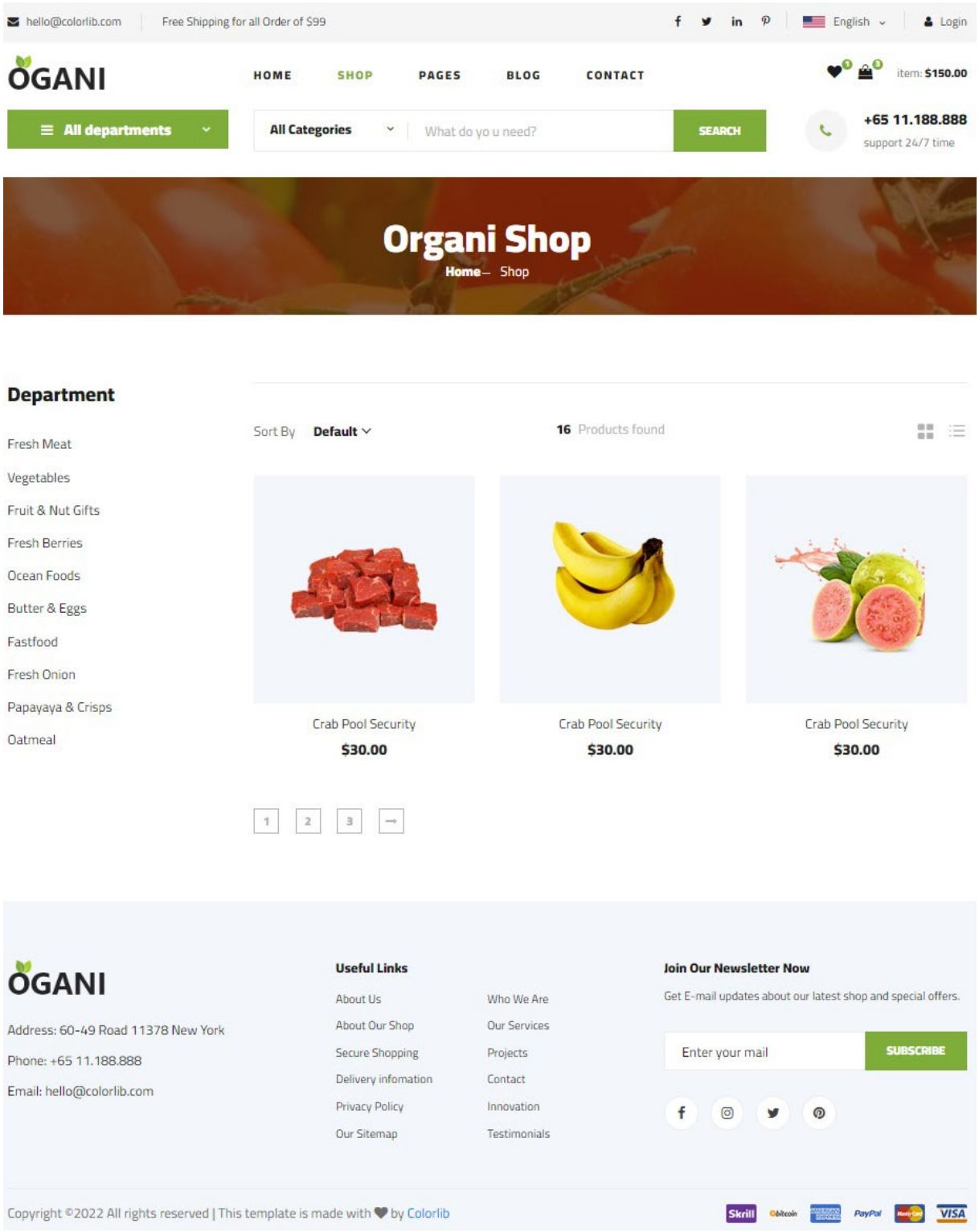
5.3.3 GUI e-shopu

Prvním krokem při návrhu GUI e-shopu je vytvoření tzv. wireframe neboli drátěného modelu. Jedná se o podklad pro zpracování grafického návrhu e-shopu, který odráží přesný pohled na

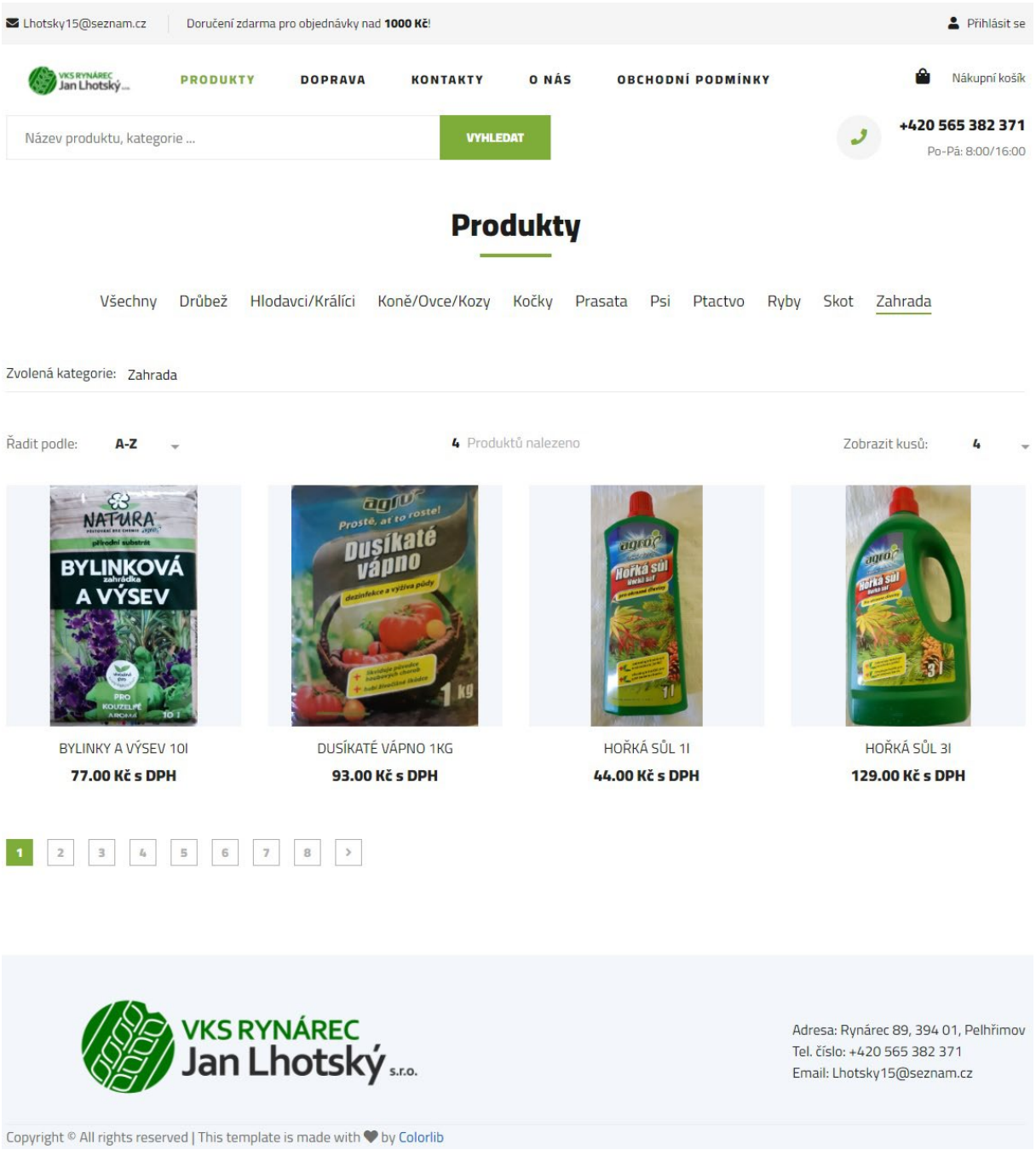
[18] Umožňuje každému záznamu z jedné tabulky přiřadit libovolný počet záznamů z druhé tabulky.

rozmístění jednotlivých ovládacích prvků na stránce e-shopu. Wireframe se vytváří pro jednotlivé stránky e-shopu jako jsou například hlavní stránka, detail produktu, jednotlivé kroky vytváření objednávky atd. Na vytvořených wireframech se následně navrhuje grafika e-shopu. Při návrhu grafiky je důležité brát ohled na to, aby jednotlivé barvy společně ladily a také aby e-shop nebyl až příliš barevný. Grafika by měla být navržena tak, aby zákazníkovi pomohla při orientaci na e-shopu. [41]

Vytváření wireframu a návrh grafiky nebylo při vytváření této práce realizováno. Namísto toho se vybrala vhodná šablona, která byla následně upravena pro potřeby vyvíjeného e-shopu. Při výběru šablony pro uživatelské rozhraní byl kladen důraz na jednoduchost a barevné rozložení. Firma primárně prodává krmné směsi pro zvířata a zahradní hnojiva. Z tohoto důvodu by se mělo jednat o šablony, které obsahují především zelené prvky, neboť zelená barva je přirozená pro výrobky z rostlin a zvěř. Co se šablony pro administrátorské rozhraní týče, zde byl kladen důraz na jednoduchost šablony s nejlépe jen jedním menu pro umístění položek sloužící k přepínání mezi funkcemi a zbytek stránky, aby sloužil ke zobrazení obsahu. U administrátorského GUI nezáleží tolik na celkovém vzhledu a dojmu, protože slouží čistě k o vládání aplikace.



Obrázek 15 – Původní vzhled šablony uživatelské části



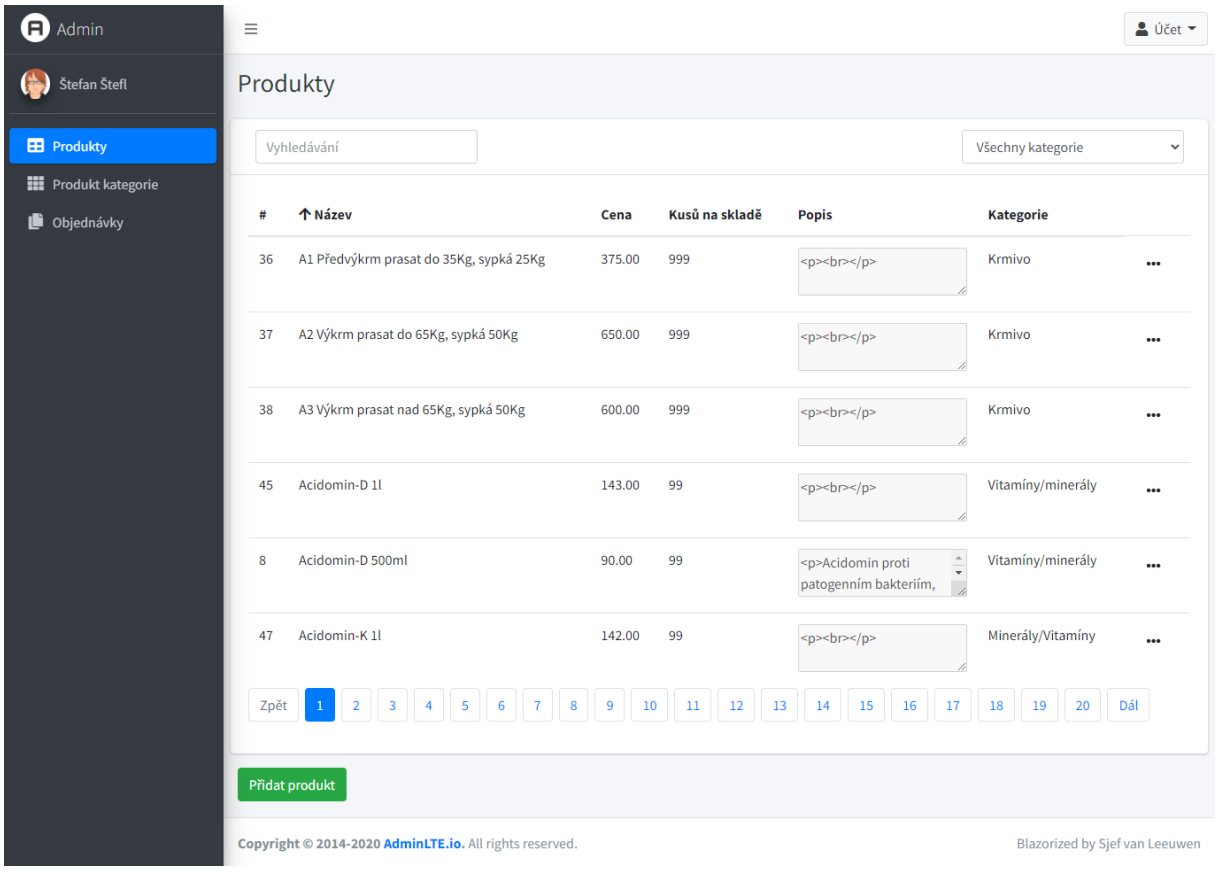
Obrázek 16 – Upravená šablona uživatelské části

Zvolená šablona pro uživatelské rozhraní splňuje veškeré požadavky. Šablona je jednoduchá, přehledná, a i když je pozadí bílé a texty černé, jsou zde prvky zbarvené do zelena. Vše dohromady působí jednoduše, čistě a hlavně přehledně. Samozřejmostí je dynamicky se upravující rozložení stránky při změně velikosti, které zajišťuje responzivitu i na menších zařízeních jako jsou mobilní telefony. Při zmenšování šířky okna automaticky dochází ke zmenšování vybraných prvků nebo k jejich úplnému skrytí či změně vzhledu. V tabulce níže

jsou vypsány hraniční velikosti šířky, při kterých dochází ke změně rozložení nebo změně velikosti jednotlivých prvků na stránce.

Šířka okna	Určení	Popis
Méně než 576px	Většinou malá mobilní zařízení	Zobrazení pouze jednoho produktu na řádek
577px až 767px	Široká mobilní zařízení	Zobrazení maximálně dvou produktů na řádek
768px až 991px	Tablety	Skrytí záhlaví a horního menu pro přepínání mezi záložky stránky
992px a více	Počítače	Vše zobrazeno v nativní velikosti

Tabulka 3 – Jednotlivé body responzivity



Obrázek 17 – Šablona administrátorské části

Šablona pro administrátorské rozhraní se skládá z hlavního postranního menu, horní lišty a obsahové části. Jedná se o jednoduchou Blazor šablonu, která vyhovuje veškerým výše

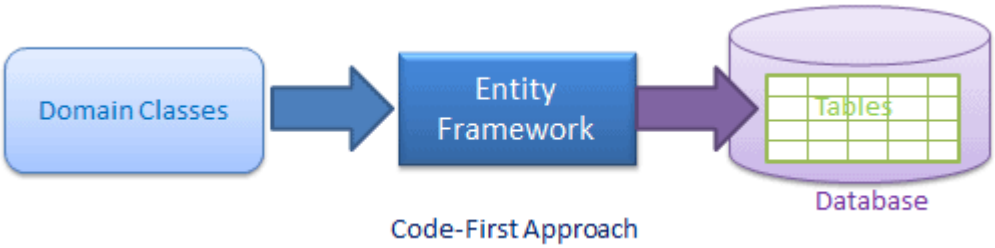
zmíněným požadavkům ohledně jednoduchosti a rozložení. Pro účely spravování elektronického obchodu je tato šablona vyhovující a v případě budoucích rozšíření funkcionalit dostačující.

6 VÝVOJ WEBOVÉ APLIKACE

Po konzultaci s firmou a stanovení veškerých požadavků na elektronický obchod následoval proces implementace e-shopu. Protože se autor diplomové práce zaměřuje především na vývojové technologie od firmy Microsoft, které jsou společně s odůvodněním jejich použití popsány v kapitole 3, není zde zahrnuta kapitola zaměřena na volbu technologií pro vývoj e-shopu.

6.1 Databáze

Databáze je generována webovou aplikací. O generování databáze se stará tooling Entity Frameworku, který na základě doménového modelu vygeneruje migrace s pseudokódem, podle kterého Entity Framework vytváří SQL příkazy, které jsou nakonec spouštěny nad databází. Tento postup se nazývá Code First. Jde o postup, při kterém se nejprve vytvoří doménové třídy s vlastnostmi na základě, kterých je vytvořena databáze namísto postupu, kdy se jako první vytvoří databáze, podle které jsou následně vytvořeny modelové třídy podle jednotlivých tabulek databáze [42].



Obrázek 18 – Code-First postup [42]

Na základě navrženého datového modelu byly, podle jednotlivých tabulek, vytvořeny odpovídající doménové třídy ve webové aplikaci. Tyto třídy obsahují properties (vlastnosti), které reprezentují jednotlivé sloupce tabulky v databázi. Pro jednotlivé properties lze stanovit definice, které se aplikují pro sloupce v databázi a zároveň se podle těchto definicí validují data na serverové straně.

```
public class Product
{
      public int ID { get; set; }

      [Required]
      [MaxLength(200)]
      public string Name { get; set; }

      public int AvailableQuantity { get; set; }

      [Column(TypeName = "decimal(18,2)")]
      public decimal Price { get; set; }

      [Column(TypeName = "decimal(18,2)")]
      public decimal? ReducedPrice { get; set; }

      [MaxLength(5000)]
      public string Description { get; set; }

      public ProductCategory ProductCategory { get; set; }

      public int? ProductCategoryID { get; set; }

      public List<ProductImage> ProductImages { get; } = new();

      public DateTime Created { get; set; }
      public DateTime? Deleted { get; set; }
}
```

Kód 3 – Doménová třída Product

Příklad doménové třídy Product, pomocí kterého tooling Entity Framework vygeneruje migraci na základě, které ORM Entity Framework dále vygeneruje SQL příkaz pro vytvoření tabulky Product v databázi. U property Name jsou v hranatých závorkách stanoveny definice[19] pro sloupec v databázi. Required značí not null hodnotu ve sloupci a MaxLength maximální délku řetězce ve sloupci. Pomocí property `ProductCategory` se v doménové třídě zapisuje vazba na tabulku ProductCategory a property `ProductCategoryID` slouží k vytvoření sloupce pro cizí klíč produkt kategorie[20]. Poslední zajímavostí je property `Deleted`. Touto property je řešeno mazání produktu příznakem, kdy se do sloupce Deleted v databázi uloží čas smazání

[19] Tyto definice jsou v ASP.NET Core nazývány jako data annotation attributes.
[20] Vytvořený sloupec obsahuje Id produkt kategorie.

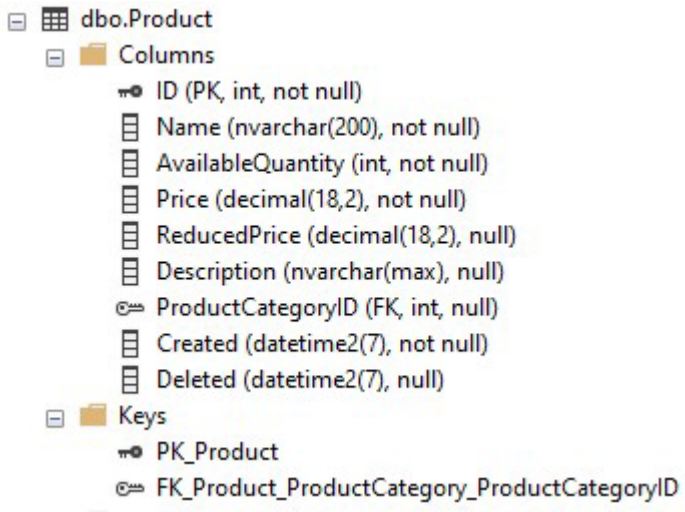
produktu v momentě jeho odstranění. Tento produkt bude stále v databázi, avšak u všech výchozích[21] dotazů na databázi dojde k vyfiltrování záznamů, které obsahují časové razítko ve sloupci Deleted. Důvod použití tohoto řešení je vazba mezi tabulkou Produkt a tabulkou Objednávka. Pokud by došlo k trvalému odstranění záznamu produktu z databáze, muselo by dojít i k trvalému odstranění záznamu dané objednávky, kvůli vazební závislosti. Zákazník by poté nemohl zpětně prohlížet objednávky s produktem, neboť oba záznamy by byly z databáze odstraněny. Z tohoto důvodu bylo zvoleno řešení mazání určitých záznamů v databázi příznakem.

```
migrationBuilder.CreateTable(
    name: "Product",
    columns: table => new
    {
        ID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        Name = table.Column<string>(maxLength: 200, nullable:
        false),
        AvailableQuantity = table.Column<int>(nullable: false),
        Price = table.Column<decimal>(type: "decimal(18,2)",
        nullable: false),
        ReducedPrice = table.Column<decimal>(type:
        "decimal(18,2)", nullable: true),
        Description = table.Column<string>(maxLength: 5000,
        nullable: true),
        ProductCategoryID = table.Column<int>(nullable: true),
        Created = table.Column<DateTime>(nullable: false),
        Deleted = table.Column<DateTime>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_Product", x => x.ID);
        table.ForeignKey(
            name:"FK_Product_ProductCategory_ProductCategoryID",
            column: x => x.ProductCategoryID,
            principalTable: "ProductCategory",
            principalColumn: "ID");
    });
```

Kód 4 – Vygenerovaný pseudokód podle doménové třídy Product

[21] Výchozí dotazy jsou generovány Entity Frameworkem a jde o dotazy jako GetAll, Delete, GetObject atd.

Vygenerovaná migrace podle doménové třídy Product, podle které Entity Framework sestaví SQL příkaz. Vytvořený příkaz vytvoří v databázi tabulku Product se sloupci ID, Name, AvailableQuantity, …, Created a Deleted.



Obrázek 19 – Výsledná podoba tabuly Product v databázi

V datovém modelu je mezi tabulkou Produkt a Objednávka znázorněn vztah M:N. Tento vztah se v relační databázi řeší za pomoci tzv. vazební tabulky, která obsahuje cizí klíče spojovaných tabulek. Pro vytvoření této vazební tabulky je nutné vytvořit vazební doménovou třídu, dle které bude vytvořena vazební tabulka v databázi. Výsledkem je doménová třída OrderItem, která reprezentuje jednotlivé položky v objednávce. V novějších verzích Entity Frameworku není nutné vytváření vazebních doménových tříd. Stačí pouze do odpovídajících doménových tříd, mezi kterými je vazba M:N, vložit property pro cizí klíč a kolekci.

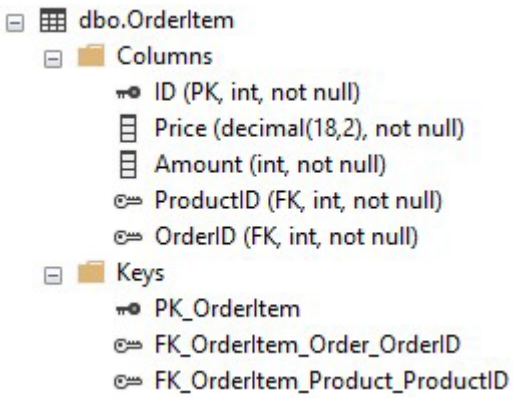
```
public class OrderItem
{
      public int ID { get; set; }

      [Column(TypeName = "decimal(18,2)")]
      public decimal Price { get; set; }

      public int Amount { get; set; }

      public Product Product { get; set; }

      public int ProductID { get; set; }

      public Order Order { get; set; }

      public int OrderID { get; set; }
}
```

Kód 5 – Doménová třída OrderItem

Datová třída OrderItem obsahuje vazbu na tabulku Product a Order a cizí klíče obou těchto tabulek. Dále pak property `Price`, k evidování aktuální cena produktu, protože se může v průběhu času měnit a Amount pro evidování počtu kusů produktu.

```
migrationBuilder.CreateTable(
    name: "OrderItem",
    columns: table => new
    {
        ID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        Price = table.Column<decimal>(type: "decimal(18,2)",
        nullable: false),
        Amount = table.Column<int>(nullable: false),
        ProductID = table.Column<int>(nullable: false),
        OrderID = table.Column<int>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_OrderItem", x => x.ID);
        table.ForeignKey(
            name: "FK_OrderItem_Order_OrderID",
            column: x => x.OrderID,
            principalTable: "Order",
            principalColumn: "ID");
        table.ForeignKey(
            name: "FK_OrderItem_Product_ProductID",
            column: x => x.ProductID,
            principalTable: "Product",
            principalColumn: "ID");
    });
```

Kód 6 – Vygenerovaný pseudokód podle doménové třídy OrderItem



Obrázek 20 – Výsledná podoba tabuly OrderItem v databázi

Tímto postupem byla vytvořena databáze webové aplikace. Z důvodu obsáhlosti nebyly zahrnuty obrázky a popisy ostatních doménových tříd, migrací a následně výsledné podoby tabulek v databázi. Pro znázornění byly zvoleny tabulku Produkt a vazební tabulka OrderItem,

na kterých jsou ukázány a popsány veškeré zajímavé části ohledně vytváření databáze za pomoci ORM Entity Frameworku.

6.2 Zabezpečení aplikace

6.2.1 Ukládání hesel

Jedním ze způsobů je ukládání zadaných hesel jako prostý řetězec. Toto se může jevit jako dobré řešení, neboť v přídě, že zákazník zapomene heslo, mu je jednoduše posláno na emailovou adresu. Ovšem tento způsob je obrovské bezpečností riziko. Hesla jsou totiž viditelná a čitelná kýmkoliv, kdo získá přístup do databáze a v případě, že se útočníkovi podaří proniknout do databáze, získá přístup ke všem uživatelským účtům. Jediná menší výhoda tohoto řešení, je jeho jednoduchost, ale kvůli velkému bezpečnostnímu riziku, se tato metoda nepoužívá.

Způsob použití v této webové aplikaci je hashování hesla[22] a uchovávání hashe[23] v databázi. Výhodou toho způsobu je určitě nečitelnost hesla, nedochází k ukládání hesla v podobě prostého řetězce a odolnost vůči prolomení hesla. Odolnost prolomení závisí na zvolené hashovací funkci. Zvolená hashovací funkce pro hashování hesel je SHA512[24]. Tento algoritmus ze vstupního řetězce vytvoří výstup, též nazýván jako otisk, fixní délky. Pro jeden řetězec je vždy stejný výstup. Velkou výhodou SHA512 je, že je prakticky nemožné z výstupu rekonstruovat vstupní řetězec.

Pro ještě větší zabezpečení hesel je v aplikaci použita metoda tzv. solení hesla. Jak již bylo v předešlém odstavci zmíněno, hashovací funkce SHA512 vygeneruje pro jeden řetězec vždy stejný výstupní řetězec. V případě, kdy má několik uživatelů stejné heslo, budou mít i stejný hash. Aby se tomuto zabránilo a snížilo se riziko prolomení stejných hesel, je ke každému heslu připojen náhodně vygenerovaný řetězec neboli sůl. Tento řetězec se většinou přidává na začátek hesla. Takto se zajistí, že i v případě výskytu stejných hesel bude mít každé jedinečný hash.

6.2.2 Autorizace uživatele

Autorizace uživatelů je řešena pomocí JSON Web Token neboli JWT. Jedná se o bez-stavový způsob řešení autorizace. Bez-stavový znamená, že na straně serveru nedochází k ukládání

[22] Hashováním hesla se rozumí proces použití hashovací funkce pro vytvoření hashe ze vstupního řetězce.
[23] Hash je označení pro řetězec písmen a číslic vytvořený hashovací funkcí.
[24] Číslo 512 označuje délku výstupního řetězce v bitech.

informací o autentizovaném uživateli, které jsou následně používány pro autorizaci. V případě JWT dojde k vygenerování tokenu pro autentizovaného uživatele, kterému je token v zašifrované formě předán. Uživatel následně token přikládá ke všem dotazům na server a server je tak schopen provést autorizaci a povolit nebo zamítnout uživateli přístup.

```
private string CreateToken(string name, string role,
                           string firstName, string lastName)
{
     byte[] key = Encoding.ASCII.GetBytes(jwtConfiguration.Key);
     SymmetricSecurityKey securityKey = new(key);

     JwtSecurityToken token = new(
          null,
          null,
          new[]
          {
               new Claim(ClaimTypes.Name, name),
               new Claim(ClaimTypes.Role, role),
               new Claim("firstName", firstName),
               new Claim("lastName", lastName),
          },
          expires: DateTime.Now.AddHours(1),
          signingCredentials:
           new SigningCredentials(
            securityKey, SecurityAlgorithms.HmacSha256)
           );

     JwtSecurityTokenHandler tokenHandler = new();

     return tokenHandler.WriteToken(token);
}
```

Kód 7 – Způsob generování JWT

Vygenerovaný token obsahuje celkově čtyři Claimy neboli vlastnosti. Vlastnosti Name a Role jsou tzv. reserved vlastnosti, které jsou defaultně obsaženy v JWT a programátor je může využít. Do vlastnosti Name se ukládá Id uživatele[25] a do vlastnosti Role se vloží role uživatele. Vlastnosti firstName a lastName jsou tzv. public vlastnosti, které jsou definované autorem diplomové práce. Do těchto vlastností jsou uloženy jméno a příjmení uživatele. Dále se tokenu nastaví doba expirace, která je v tomto případě nastavena na 1 hodinu, a nakonec se vytvoří

[25] Jde o jedinečný identifikátor uživatele.

podpis JWT. Podpis JWT je vytvořen z hlavičky, obsahové části a bezpečnostního klíče, který je známý pouze serveru. Pro vytvoření podpisu je použit symetrický algoritmus HS256. Vygenerovaný token je poté poslán uživateli.

6.3 Administrátorské API

API pro administrátorskou část webové aplikace je implementováno za použití frameworku gRPC, které funguje na způsobu vzdáleného volání procedur. Pro zajištění funkčnosti gRPC je nutné, aby klient a server měli identické .proto soubory které definují metody a zprávy. Obsah .proto souborů vypadá následovně:

```
syntax = "proto3";

service Greeter {
     rpc SayHello (HelloRequest) returns (HelloReply);
}
message HelloRequest {
     string name = 1;
}
message HelloReply {
     string message = 1;
}
```

Kód 8 – Proto soubor definující metodu Greeter

Na prvním řádku je specifikováno, která verze syntaxe je použita. V případě nespecifikování se použije defaultní verze syntaxe proto2. Service Greeter odpovídá servise Greeter na serveru s metodou SayHello, která má vstupní parametr HelloReqeust a vrací objekt HelloReply. Nakonec message reprezentuje podobu posílaných nebo přijímaných dat. Uvnitř složených závorek se definují buďto jako primitivní datové typy nebo složené datové typy spolu s názvem a pořadím, které se píše za rovnítko. Tímto způsobem jsou popsány servisy, jejich metody a podoba vstupních a výstupních parametrů. Tyto .proto soubory následně slouží jako definice volání serverových služeb pro klientským RPC nástroje. [29]

Na serverové straně aplikace je nutné zaregistrovat přístupové body API, na kterých bude server naslouchat a obsluhovat příchozí požadavky od klientů. Registrování přístupových bodů je umístěno ve třídě Startup.cs, která obsahuje ConfigureServices metodu pro konfiguraci veškerých servis aplikace. Právě v této třídě dojde k zaregistrování přístupových bodů gRPC.

```
app.UseEndpoints(endpoints =>
{
     endpoints.MapRazorPages();
     endpoints.MapControllers();
     endpoints.MapFallbackToFile("index.html");

     endpoints.MapGrpcService<AddressFacade>();
     endpoints.MapGrpcService<DeliveryTypeFacade>();
     endpoints.MapGrpcService<OrderFacade>();
     endpoints.MapGrpcService<OrderStateFacade>();
     endpoints.MapGrpcService<PaymentStateFacade>();
     endpoints.MapGrpcService<PaymentTypeFacade>();
     endpoints.MapGrpcService<ProductFacade>();
     endpoints.MapGrpcService<ProductCategoryFacade>();
     endpoints.MapGrpcService<SecurityService>();
});
```

Kód 9 – Registrace přístupových bodů na straně serveru

Protože serverová i klientská část jsou obě implementovány v .NET Core, je použita metodika Code-first gRPC. Díky použití této metody není nutné vytváření .proto souborů. Místo toho je použita .NET anotace pro definování podoby metod a zpráv. V aplikaci jsou vytvořeny datové třídy a interface neboli rozhraní. Datové třídy definují podobu posílaných zpráv pomocí gRPC a rozhraní definuje podobu metod spolu s návratovými typy a atributy metody. [43]

```
[DataContract]
public class ProductDto
{
      [DataMember(Order = 1)]
      public int ID { get; set; }

      [DataMember(Order = 2)]
      public string Name { get; set; }

      [DataMember(Order = 3)]
      public decimal Price { get; set; }

      [DataMember(Order = 4)]
      public int AvailableQuantity { get; set; }

      [DataMember(Order = 5)]
      public decimal? ReducedPrice { get; set; }

      [DataMember(Order = 6)]
      public string Description { get; set; }

      [DataMember(Order = 7)]
      public int? ProductCategoryID { get; set; }

      [DataMember(Order = 8)]
      public List<int> ProductImageIDs { get; set; } = new();
}
```

Kód 10 – Třída definující formát gRPC zprávy

```
[ServiceContract]
public interface IProductFacadeContract
{
      Task<ProductDto> CreateProductAsync(ProductIM inputModel);

      Task<ProductDto> GetProductAsync(GetProductRequest
       request);

      Task<List<ProductDto>> GetAllProductsAsync();

      Task UpdateProductAsync(ProductDto inputModel);
}
```

Kód 11 – Rozhraní popisující třídu a její metody

O komunikaci mezi klientem a gRPC přístupovými body se stará code-first gRPC klient, který využívá servisní kontrakty pro volání metod ze serveru. Aby klient mohl provádět vzdálené

volání procedur, je potřeba jej zaregistrovat spolu a předat mu rozhraní, které definuje podobu třídy na serveru. Dále mu je předána adresa serveru, na který budou posílány dotazy.

```
builder.Services.AddCodeFirstGrpcClient<IProductFacadeContract>(
      o => o.Address = new Uri(builder.HostEnvironment.BaseAddress)
).ConfigurePrimaryHttpMessageHandler(
      () =>new GrpcWebHandler(
            GrpcWebMode.GrpcWeb, new HttpClientHandler())
);

builder.Services.AddCodeFirstGrpcClient<IProductCategoryFacadeContract>(
      o => o.Address = new Uri(builder.HostEnvironment.BaseAddress)
).ConfigurePrimaryHttpMessageHandler(
      () => new GrpcWebHandler(
                  GrpcWebMode.GrpcWeb, new HttpClientHandler())
);
```

Kód 12 – Registrace code-first gRPC klienta spolu s odpovídajícím rozhraním

IProductFacadeContract je rozhraní zobrazené na obrázku výše, které code-first gRPC klientu udává podobu třídy na serveru. Pomocí builder.HostEnvironment.BaseAddress se nastaví adresa, na které operuje server a nakonec pomocí GrpcWebMode.GrpcWeb se nastaví způsob volání v podobě HTTP dotazů.

```
PagedQueryResult<ProductDto> data = await ProductFacadeContract
      .GetProductListAsync(pagedQuery)
      .ConfigureAwait(false);
products = data.Data;
```

Kód 13 – Voládní vzdálené procedury

Příklad vzdáleného volání procedury GetProductListAsync (kód 13), které se je zapsáno stejným způsobem jako volání metody třídy nacházející se ve stejném adresním prostoru.

6.4 Uživatelské API

Uživatelské API je implementováno podle architektury rozhraní zvané REST. Na rozdíl od administrátorského API je uživatelské API orientováno na data, nikoliv na volání vzdálených procedur. Data jsou mezi klientem a serverem přenášena ve formátu JSON, který je jedním z nejrozšířenějších formátů. ASP.NET obsahuje zabudované komponenty, které vytváření REST API značně usnadňují a zlehčují. Tyto komponenty se automaticky starají o deserializaci příchozích HTTP dotazů a serializaci návratových dat do HttpResponse.

```
[Route("api/product")]
public class ProductController : ControllerBase
{
      [HttpGet("getProducts")]
      public async Task<List<ProductVM>> GetProductsAsync()
      {
            return await productFacade.GetAllProductsAsync();
      }

      [HttpPost("getProducts")]
      public async Task<PagedQueryResult<ProductVM>>
       GetProductsAsync([FromBody]
       PagedQueryRequest<ProductListQueryFilter> pagedQueryRequest)
      {
            return await
             productFacade.GetPagedProductsAsync(pagedQueryRequest);
      }

}
```

Kód 14 – API přístupové body uživatelské části

Na serverové straně jsou v tzv. Controllerech umístěny a definovány přístupové body ke zdrojům[26]. V prvním řádku ve složených závorkách je definována základní část URL adresy pro ProductController. První metoda GetProductsAsync je zavolána v případě, kdy na server přijde HTTP dotaz s metodou Get. Metoda HTTP dotazu a doplňující název URL jsou definovány ve složených závorkách nad touto metodou. Aby byla tato metoda zavolána a vykonána, musí klient poslat HTTP dotaz s Get metodou na adresu: https://adresaServeru/api/product/getProducts.

Druhá metoda má stejný název URL adresy jako ta první. Odlišují se pouze metodou HTTP dotazu a zároveň jsou zde očekávána data v obsahové části HTTP dotazu, která jsou následně předána jako parametr do metody GetPagedProductsAsync.

Oba zmíněné přístupové body vrací kolekci produktů, které jsou následně zobrazeny uživateli. První přístupoví bod vrací veškeré produkty, které se nachází v databázi a druhý přístupový bod produkty filtruje a seřazuje dle požadavků, které jsou obsaženy v těle příchozího HTTP dotazu.

[26] Zdroje představují návratová data, stavy aplikace apod.

```
[HttpPost("updateUserDetails")]
[Authorize(Roles = nameof(Role.Entry.User))]
public async Task<IActionResult>
UpdateUserDetailsAsync([FromBody] UserDetailsDto inputModel)
{
     if (int.TryParse(User.Identity.Name, out int userId) &&
         await accountFacade.UpdateUserDetailsAsync(userId,
         inputModel))
     {
          return Ok();
     }

     return BadRequest();
}
```

Kód 15 – API přístupový bod s vyžadovanou autorizací

V případě zabezpečeného přístupového bodu je nad danou metodou ve složených závorkách specifikováno provedení autorizace. Autorizace je řešena pomocí JWT, který se přikládá do hlavičky HTTP dotazu. JWT je na straně serveru rozšifrováno a je zkontrolována jeho validace. V případě nevalidního JWT nebo nenalezení JWT v hlavičce HTTP dotazu je přístup na tento přístupový bod zamítnut. Tímto způsobem jsou zabezpečeny i ostatní přístupové body v aplikaci.

7 POPIS WEBOVÉ APLIKACE

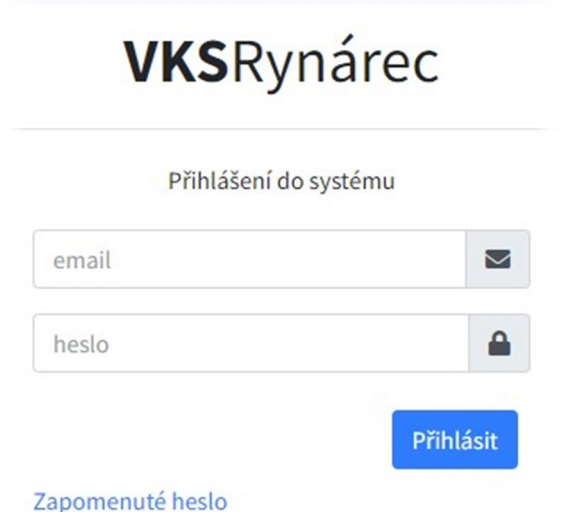
V této kapitole je popsána výsledná webová aplikace. Kapitola je rozdělena na dva oddíly, kde v prvním oddíle je popsána administrátorská část aplikace a v druhém oddíle uživatelská část aplikace.

7.1 Administrátorské rozhraní

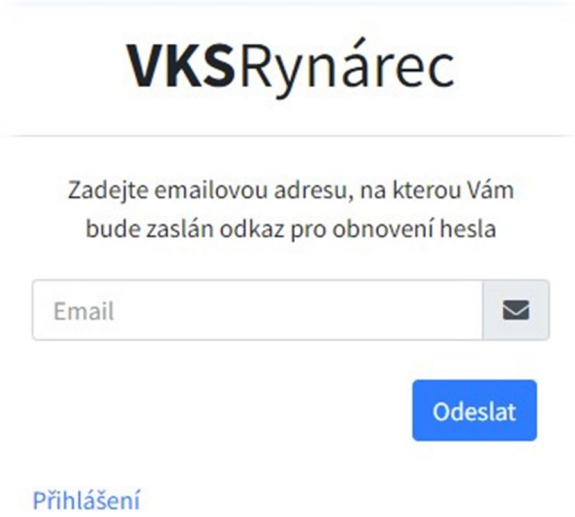
Administrátorské rozhraní umožňuje slouží ke spravování elektronického obchodu. Jde o část aplikace, která umožňuje vkládání, upravování a mazání jednotlivých produktů a produkt kategorií. Dále pak umožňuje správu objednávek v systému. V levé části administrátorského rozhraní je postranní navigační menu, pomocí kterého lze přepínat mezi jednotlivými stránkami pro správu produktů, kategorií nebo objednávek. V horní části je umístěn sloupec, kde se nachází rozbalovací menu s možnostmi pro změnu hesla a odhlášení se ze systému. Nakonec uprostřed rozhraní je umístěna obsahová část, kde zobrazují veškerá data jako seznam produktů či objednávek.

7.1.1 Přihlašovací obrazovka

Přihlašovací formulář (obrázek 22) je první věcí, která je zobrazena při přístupu do administrátorského rozhraní. Zde je uživatel nucen zadat přihlašovací emailovou adresu a heslo, aby mohla být provedena jeho autentizace. Při neúspěšné autentizaci je nad vstupním polem pro emailovou adresu vypsána varovná zpráva o neúspěšném pokusu o přihlášení do systému. V přihlašovacím formuláři se také nachází odkaz Zapomenuté heslo pro přepnutí na formulář (obrázek 21), kde uživatel zadá přihlašovací email, na který mu je následně poslán odkaz pro resetování hesla.



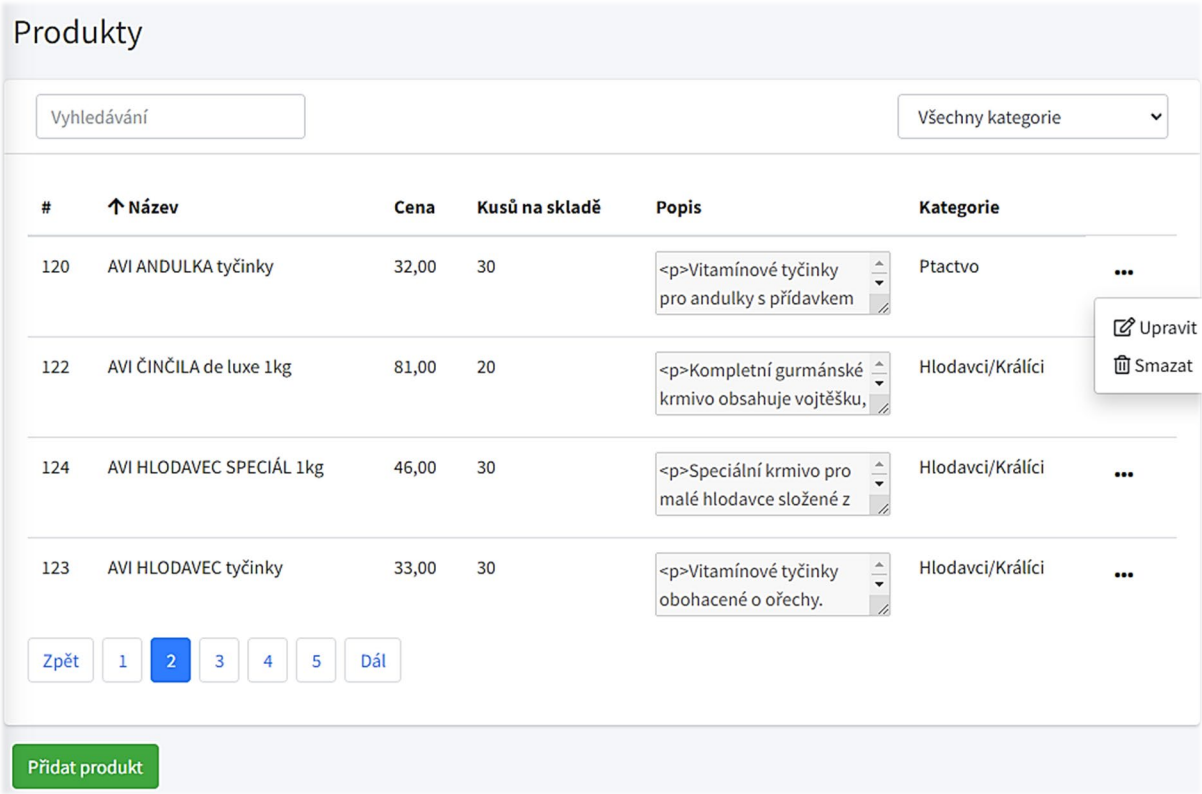
Obrázek 22 – Přihlašovací okno admina



Obrázek 21 – Obnovení hesla

7.1.2 Správa produktů

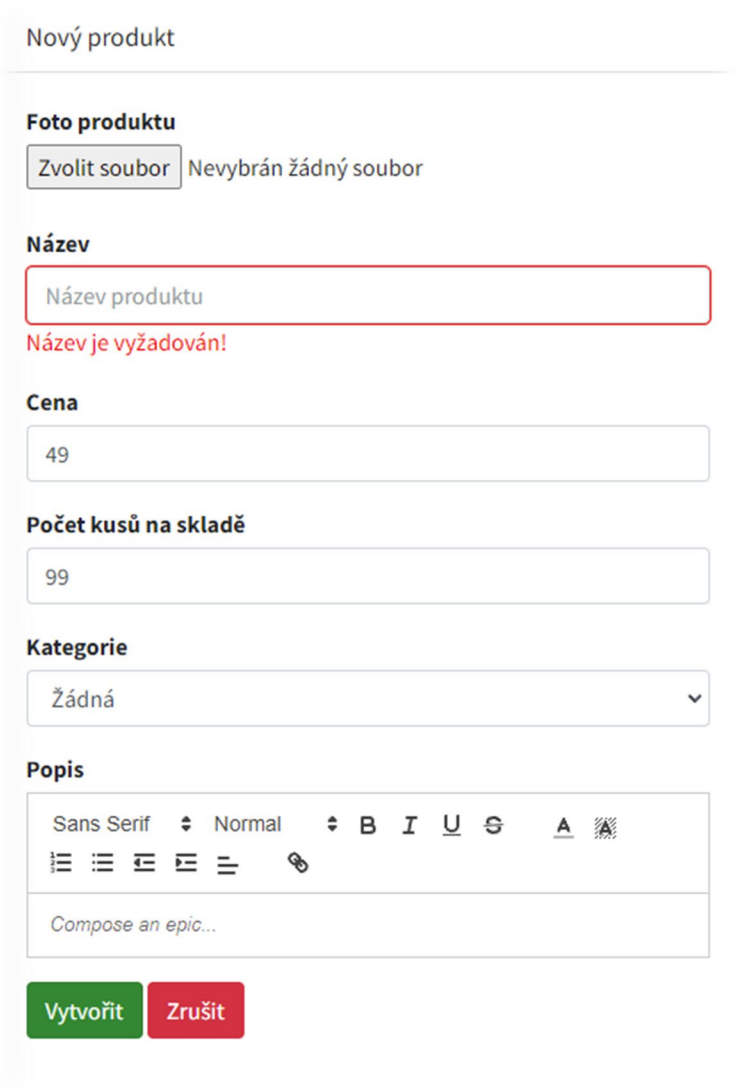
Správa produktů jako taková se skládá z listu produktů a stránkování, které je umístěno ve spodní části listu. V listu produktů se dále nachází tlačítko pro vložení nového produktu a jednotlivé řádky s produktem obsahují rozbalovací menu, které dále obsahuje možnosti pro úpravu a smazání produktu. V horní části listu se nachází názvy sloupců, podle kterých lze list produktů řadit. Nad listem produktů je umístěno pole pro vyhledávání produktů dle jména, kategorie nebo popisu. Také je zde umístěno rozbalovací menu kategorií, které slouží k filtrování produktů dle jejich kategorie.



Obrázek 23 – Seznam produktů

Po kliknutí na tlačítko pro vložení nového produktu se místo listu produktů zobrazí formulář pro vložení nového produktu do systému. Formulář obsahuje pole s názvy, do kterých uživatel zadává vyžadované i nevyžadované informace o vkládaném produktu např.: název produktu, cena, popis atd. Formulář dále obsahuje dvě funkční tlačítka, jedno pro zrušení vkládání produktu, které zpátky zobrazí list produktů a druhé pro potvrzení vložení nového produktu. Při potvrzení vložení produktu dojde k validaci jednotlivých polí a v případě vyplnění nevalidních dat, dojde ke zobrazení červeného varovného textu pod chybně vyplněnými poli, aby uživatel věděl, jaké pole nevyhovují validaci a také proč vyplněné hodnoty nevyhovují

validaci. Pokud jsou veškerá vložená data validní, dojde ke vložení nového produktu do databáze a uživateli je znovu zobrazen list produktů, který obsahuje i nově vložený produkt.

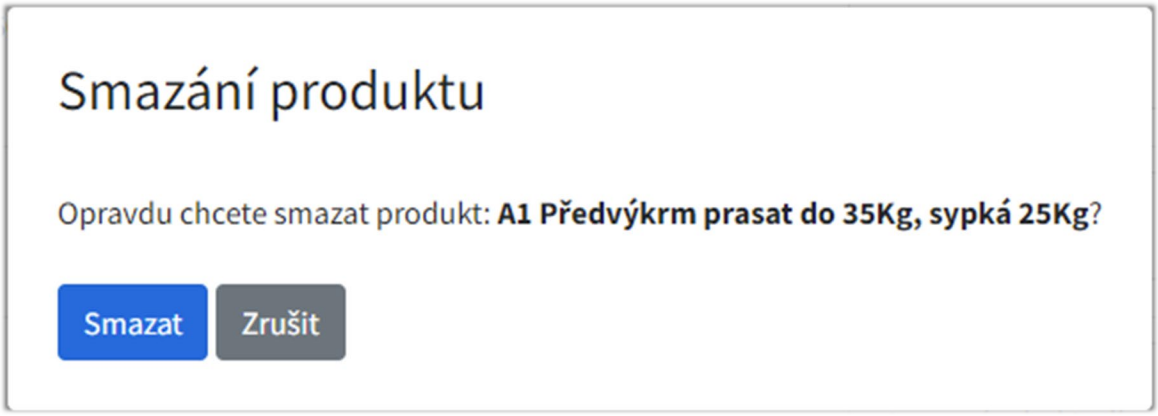


Obrázek 24 – Vložení produktu do systému

Formulář pro úpravu existujícího produktu vypadá stejně jako ten pro vložení nového produktu. Spolu s formulářem jsou navíc zobrazeny obrázky produktu (obrázek 25), pokud daný produkt nějaké obrázky má. Jednotlivé obrázky mají v pravém horním rohu vykreslený červený křížek, který představuje funkci smazání obrázku. Stejně jako u vkládání nového produktu je i zde validace vložených hodnot a v případě úpravy stávajících hodnota na nevalidní hodnoty se zobrazí varovný text a nevalidní hodnoty se neuloží do databáze.

Obrázek 25 – Fotky produktu na stránce editace produktu

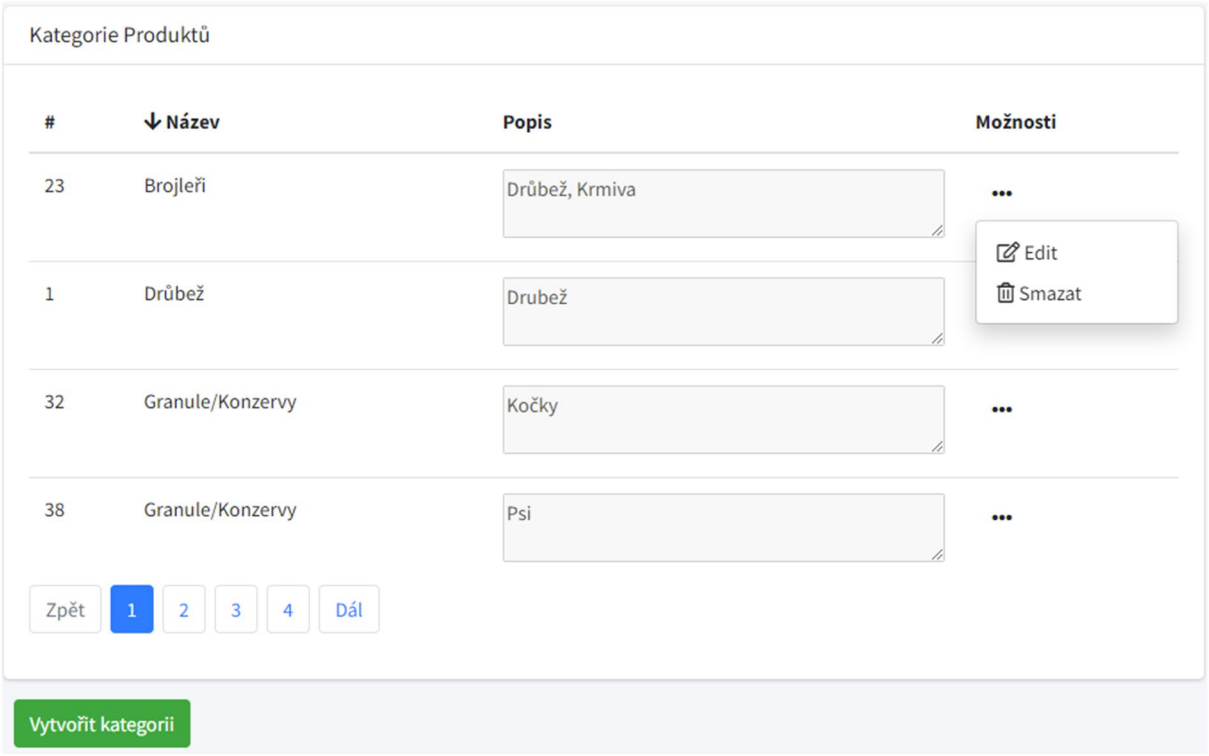
Poslední ze zmíněných funkčností je smazání produktu. Pokud uživatel vybere v rozbalovacím menu možnost smazání produktu, dojde ke zobrazení modálního okna (obrázek 26), které se uživatele zeptá, zdali si opravdu přeje smazat vybraný produkt. Uživatel má následně na výběr, zdali potvrdí smazání a tím dojde ke smazání produktu příznakem nebo smazání produktu zruší, a tedy se modální okno zavře a neprovede se žádná další akce.



Obrázek 26 – Potvrzení smazání produktu

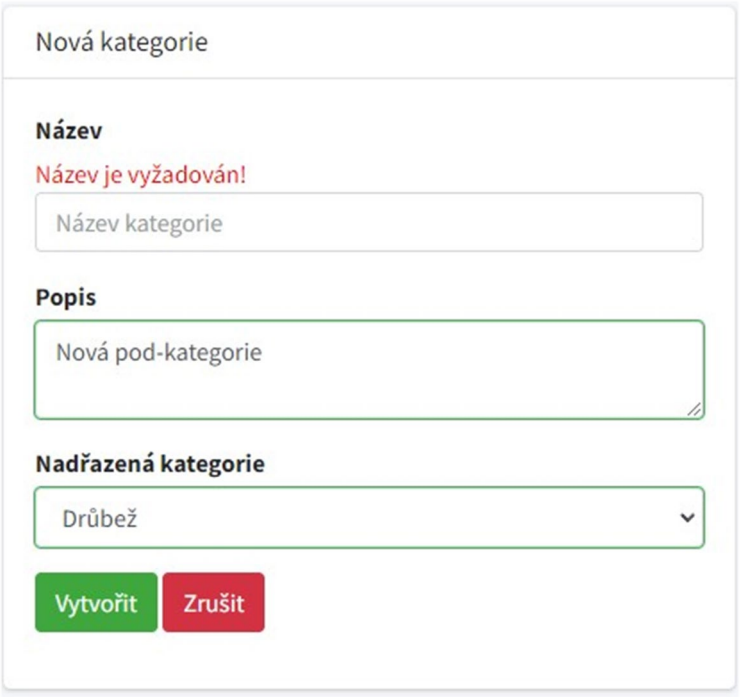
7.1.3 Správa kategorií produktů

Stránka se správou kategorií (obrázek 27) je téměř identická jako stránka správy produktů. Liší se pouze počtem a názvy sloupců v listu kategorií. U jednotlivých kategorií je zobrazen jejich název, popis a ovládací tlačítko pro zobrazení rozbalovacího menu s možnostmi úpravy nebo smazání dané kategorie. Pod listem kategorií je umístěno tlačítko pro vložení nové kategorie.



Obrázek 27 – Seznam produkt kategorií

Formulář pro vytvoření nové kategorie je, oproti formuláři pro vložení nového produktu, méně obsáhlý. Při vkládání nové kategorie se zadává jen jméno kategorie, popis a z rozbalovacího menu nadřazená kategorie. Ani zde nechybí validace vkládaných hodnot. Validován je název kategorie, který je povinný a v případě jeho nevyplnění nedojde k odeslání požadavku na server a uživateli je zobrazen varovný text, aby toto pole vyplnil. Formulář pro vložení nové kategorie (obrázek 28) je identický s formulářem pro úpravu kategorie i co se validace týče.

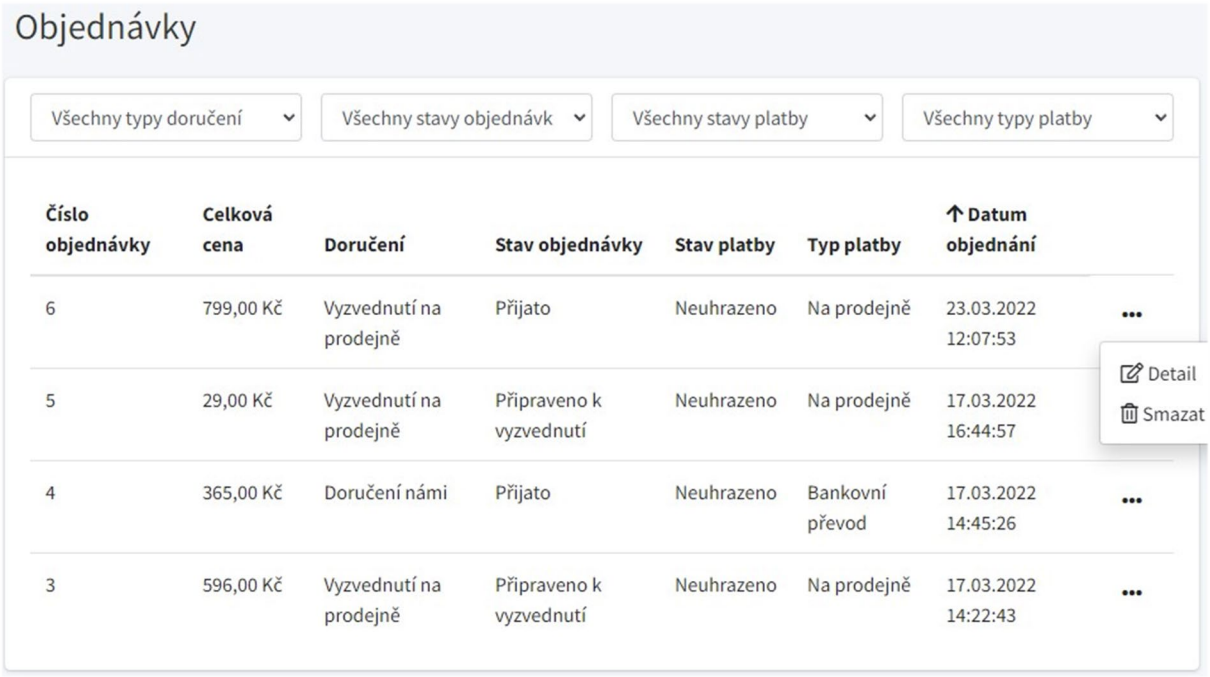


Obrázek 28 – Vložení produkt kategorie do systému

V neposlední řadě zmíněná funkce pro smazání kategorie funguje stejně jako u smazání produktu. Uživateli je zobrazeno modální okno se zprávou, názvem dané kategorie a možností potvrzení nebo zrušení smazání kategorie.

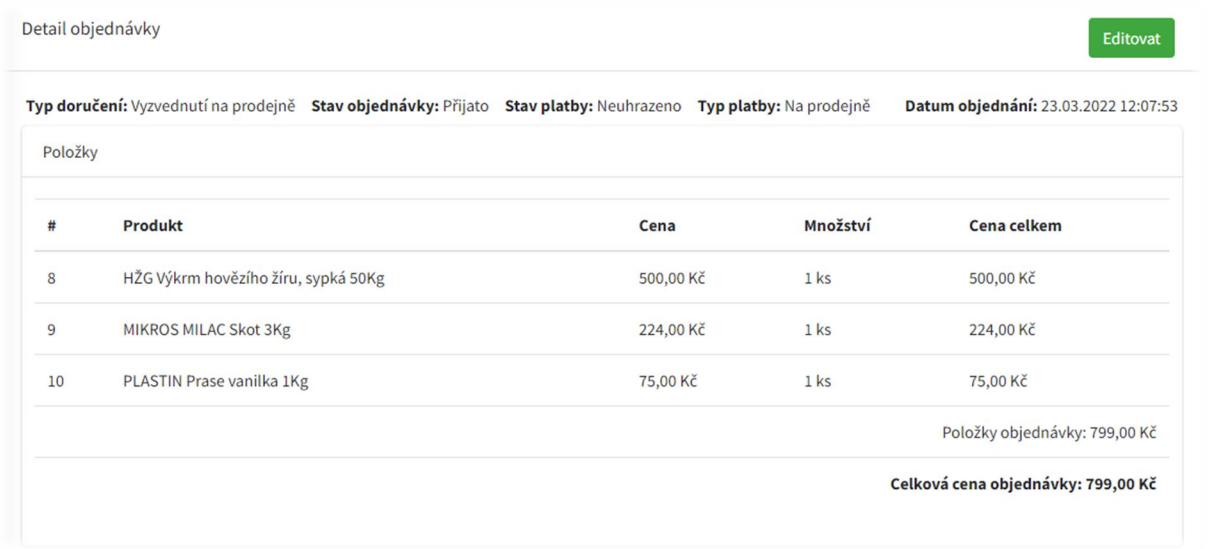
7.1.4 Správa objednávek

Nejzajímavější a zároveň nejpoužívanější funkcí aplikace je správa objednávek. Spravování nových, ale i existujících objednávek je hlavní funkcionalitou administrátorské části aplikace. Stránka správy objednávek se skládá z listu objednávek (obrázek 29), kde se v horní části nachází rozbalovací menu typ doručení, stav objednávky, typ platby a stav platby. Dle zmíněných rozbalovacích menu je uživatel schopen filtrovat objednávky. Dále je možné objednávky řadit dle velikosti celkové ceny nebo data vytvoření. Jednotlivé řádky objednávek obsahují rozbalovací menu, které obsahuje možnost pro zobrazení detailu objednávky a možnost pro smazání objednávky.



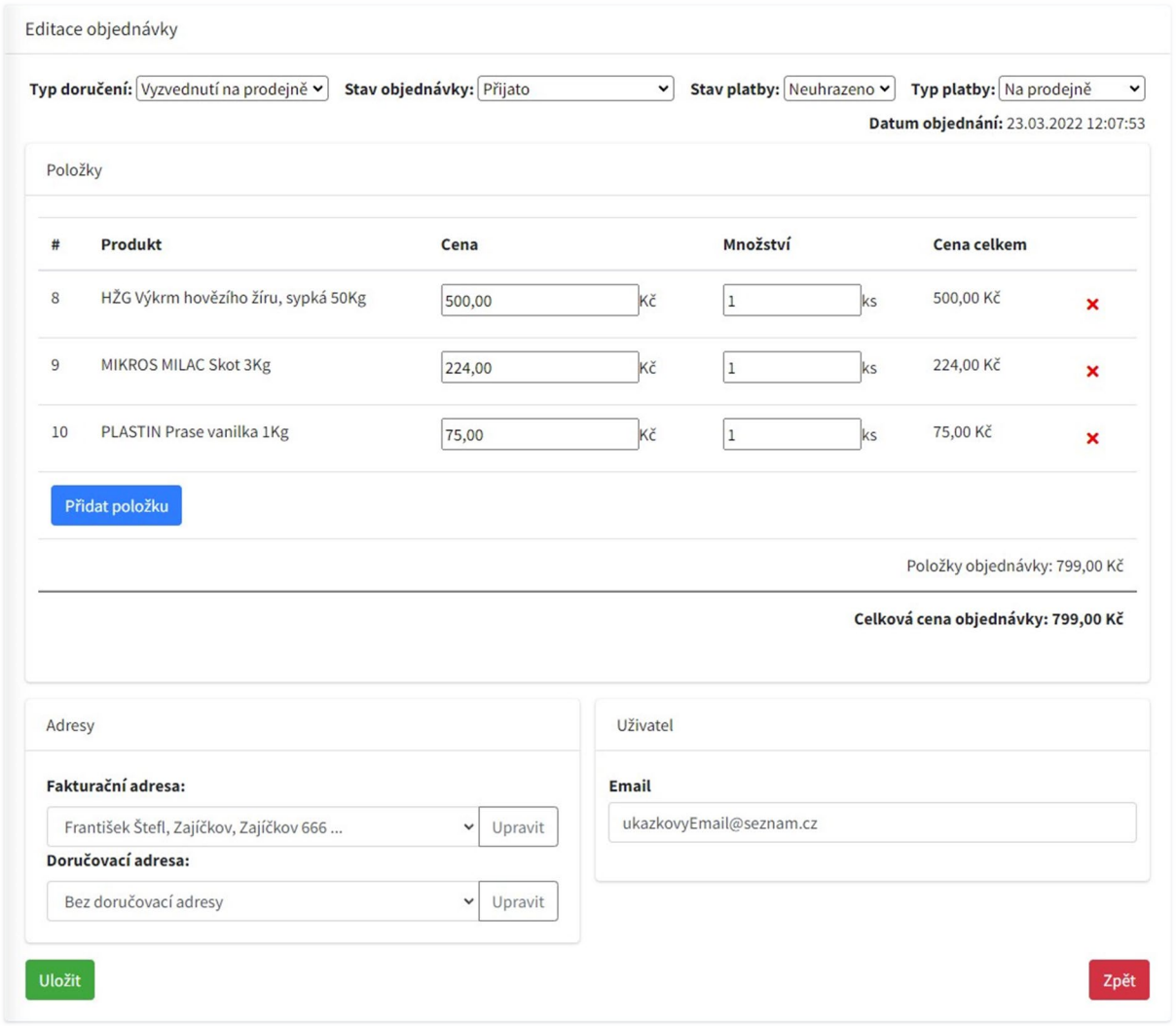
Obrázek 29 – Seznam objednávek

Na stránce detail objednávky (obrázek 30) jsou zobrazeny veškeré důležité informace o dané objednávce. V horní části se nachází informace o typu doručení, stavu objednávky, typu platby, stavu platby a datum vytvoření objednávky. Poté v prostředí části je seznam produktů objednávky. U jednotlivých produktů je uveden jejich název, cena platná pro produkt v době vytvoření objednávky, počet kusů a celková cena. Nakonec ve spodnější části se nachází fakturační adresa a informace o zákazníkovi. Pro editaci objednávky je v pravém horním rohu umístěno tlačítko pro přesměrování na editační stránku objednávky.



Obrázek 30 – Detail objednávky

Na stránce pro editaci objednávky (obrázek 31) je administrátor schopen změnit zmíněné typy doručení, platby a stavy objednávky a platby. Dále je pak možné přidat nové produkty do objednávky nebo jednotlivé produkty z objednávky odebrat. Uživatel také může změnit cenu a množství u jednotlivých produktů. V neposlední řadě je administrátor schopen změnit fakturační adresu na jinou, nebo aktuální fakturační adresu editovat. Doručovací adresu je taktéž možno změnit nebo editovat, ale i vytvořit novou doručovací adresu, která bude následně přidělena uživateli, který objednávku vytvořil. Editace jednotlivých adres je implementována tak, že dochází k editaci adresy, která je přímo přiřazena k objednávce, nikoliv k editaci adresy, kterou má zákazník uloženou. Pokud má registrovaný uživatel uloženou adresu a administrátor ji v editaci objednávky, jakkoliv pozmění, dojde k vytvoření duplicitní adresy, která je následně změněna dle zadaných změn administrátorem a přiřazena k objednávce. Uložená adresa uživatele se tedy nezmění. U doručovací adresy lze nastavit stav bez doručovací adresy. Tento stav znamená, že objednávka bude vyfakturována a doručena na jednu adresu, kterou je fakturační adresa.



Obrázek 31 – Editace objednávky

7.2 Uživatelské rozhraní

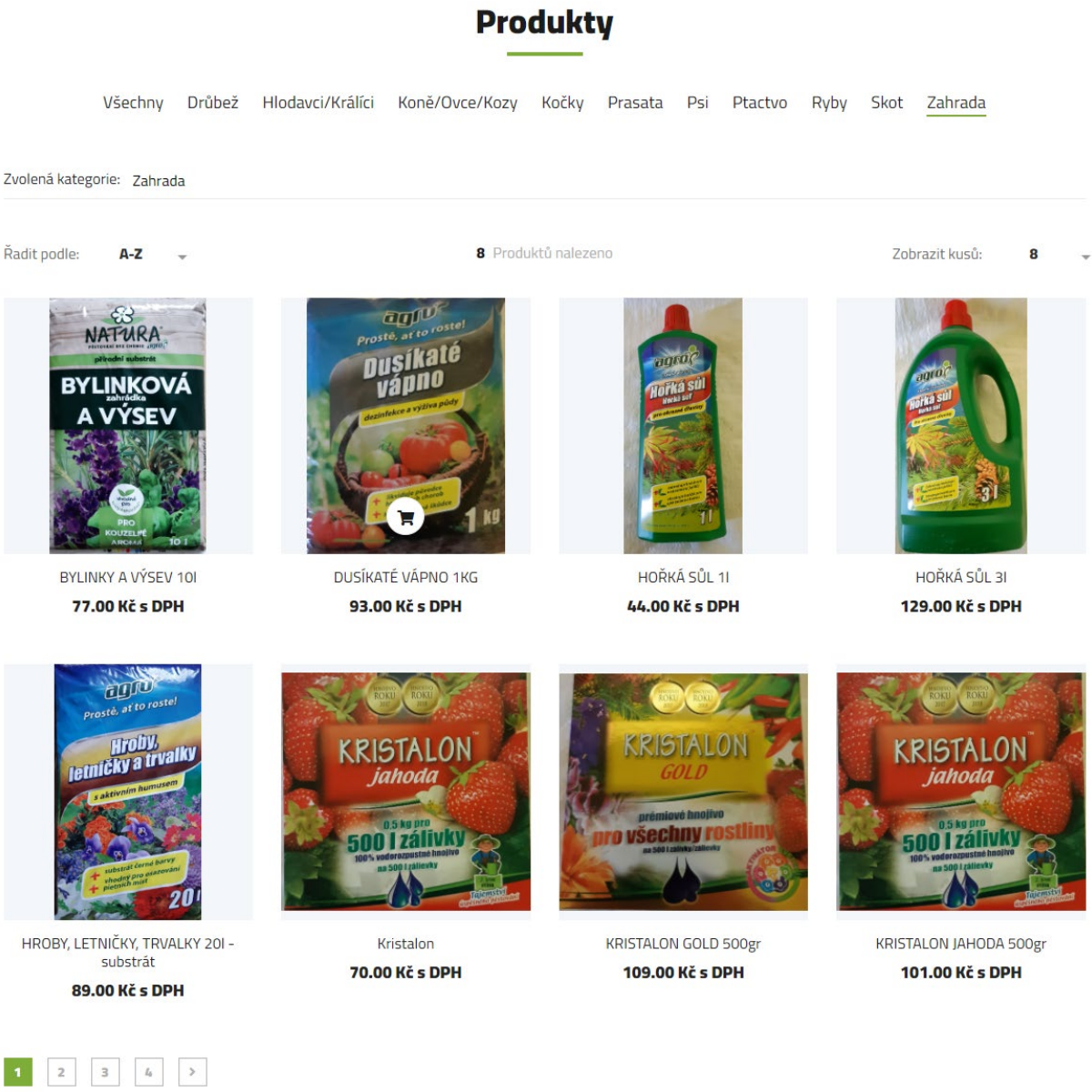
Uživatelské rozhraní je část aplikace, která je určena pro zákazníky k prohlížení nabízeného zboží a vytváření objednávek. Uživatelské rozhraní se sestavuje z horní lišty, kde se nachází menu se záložky, tlačítko nákupního košíku a tlačítko pro přihlášení uživatele. Pod horní lištou je umístěn řádek s polem pro vyhledávání a kontaktní telefonní číslo spolu s otevírací dobou firmy. Dále níž je obsahová část, kde se dynamicky vykresluje obsah na základě zvolené záložky, např.: produkty, obchodní podmínky, informace o dopravě apod. Nakonec ve spodní části je umístěno zápatí stránky, které obsahuje adresu firmy.

7.2.1 Seznam produktů

Seznam produktů je hlavní stránkou uživatelského rozhraní. Jde o stránku, která se zobrazí zákazníkovi při vstupu na webové stránky po zadání adresy bez parametrů. Jelikož jde o první

stránku, kterou nový zákazník uvidí, tak by měla působit atraktivně, a tedy nesmí zákazníka odradit svým vzhledem a uspořádáním. Z tohoto důvodu bylo snahou navržení seznamu produktů a ostatních prvků webové stránky tak, aby vše působilo jednoduše a bylo lehce čitelné a přehledné.

V horní části seznamu produktů se nachází menu s kategoriemi produktů, díky kterému je zákazníkovi umožněno filtrování na základě jednotlivých kategorií. V jeden okamžik může být filtrováno pouze jednou kategorií, tzn. jednotlivé kategorie se neprolínají. Pod menu kategorií se v levém okraji nachází rozbalovací menu pro řazení produktů. Produkty lze řadit dle názvu, a to vzestupně nebo sestupně, anebo podle ceny a taktéž vzestupně či sestupně. Naproti tomu v pravé části se nachází rozbalovací menu, kde si zákazník může zvolit počet aktuálně zobrazených produktů na stránku. Nakonec uprostřed řádku se vypisuje informace o aktuálním počtu zobrazených nebo nalezených produktů. Pod těmito ovládacími a informativními prvky se zobrazují produkty. Produkty jsou vykreslovány ve stylu tzv. mřížky, kdy jeden řádek obsahuje maximálně 4 sloupce s produkty. V jednotlivých buňkách mřížky je zobrazen obrázek produktu, pod obrázkem název produktu a pod názvem je cena produktu. Při najetí kurzoru myši na produkt se na obrázku produktu zobrazí tlačítko s ikonkou košíku, které slouží pro vložení jednoho produktu do košíku. Po kliknutí na toto tlačítko se v pravém horním rohu stránky zobrazí upozornění o úspěšném vložení produktu do košíku, nebo upozornění, že daný produkt již není skladem. Kliknutím na obrázek nebo název produktu se zákazníkovi zobrazí detail produktu.

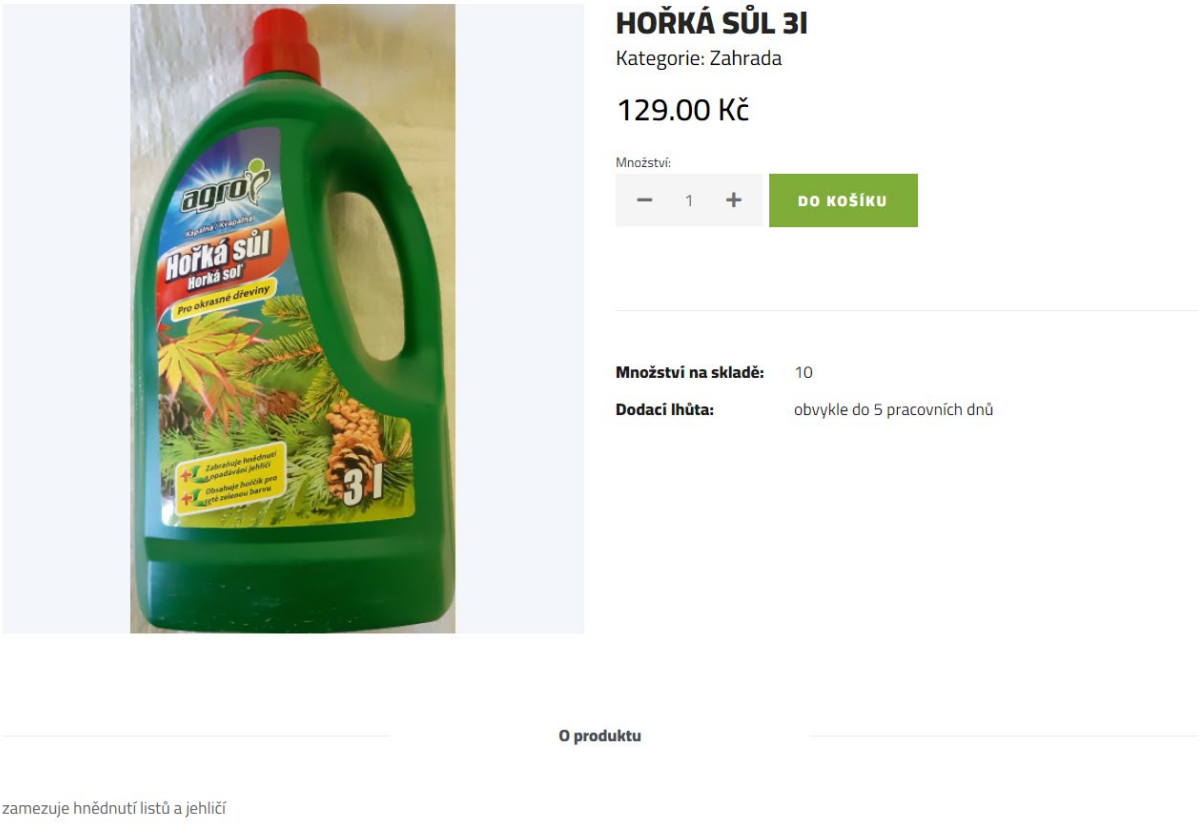


Obrázek 32 – Seznam produktů

7.2.2 Detail produktu

Slouží k podrobnějšímu zobrazení informací o produktu zákazníkovi. Je zde důležité poskytnou zákazníkovi zajímavé fotografie produktu v dobré kvalitě, aby měl představu o tom, co kupuje. Také je podstatné mít u produktu výstižný popis. Popis by měl být lehce čitelný a měl by obsahovat dostatečné informace o daném produktu jako například: pro co je produkt určený, jak produkt použít a dávkovat, složení, nutriční hodnoty atd. Nakonec zde nesmí chybět název produktu, jeho kategorie, cena a ovládací prvky pro výběr množství a vložení produktu do košíku.

Rozložení stránky s detailem produktu je rozděleno na tři části: fotografie, ovládací prvky a popis produktu. Posuvník s obrázky produktu je umístěn do levé části stránky a napravo od něj je umístěn název produktu, kategorie, ovládací prvky a pár základních informací o produktu jako je dostupnost, dodací lhůta a velikost nebo váha. Oba tyto elementy jsou vedle sebe na jednom řádku a pod nimi je umístěn obsáhlejší textový popis produktu. Posuvník s obrázky produktů automaticky zobrazuje jednotlivé obrázky v určitém časovém intervalu. Toto se děje pouze v případě, že má produkt přiřazen více než jeden obrázek, mezi kterými může i uživatel sám přepínat pomocí ovládacích šipek na levém a pravém okraji obrázku. Na jednotlivé obrázky lze kliknout a tím je zvětšit. Dále možnost měnění množství produktu je implementována pomocí dvou tlačítek, každé z jedné strany číslovky udávající aktuální zvolené množství produktu. Tlačítka obsahují matematické znaky plus a mínus a po kliknutí se buďto zvýší nebo sníží množství produktu o jeden kus. Množství nemůže nabýt nulové nebo záporné hodnoty. Tlačítko „Do košíku“ následně vloží navolené množství produktu do košíku.



Obrázek 33 – Detail produktu

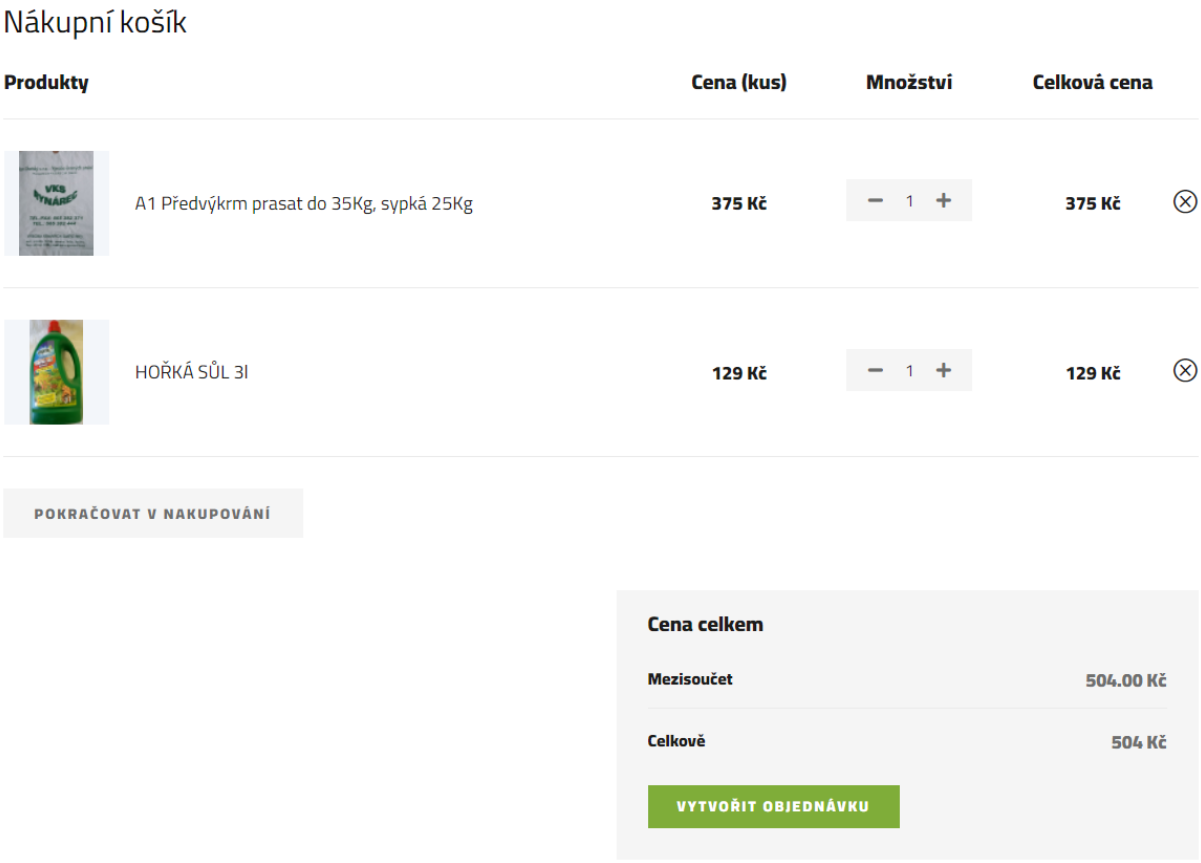
Proces kontroly počtu objednávaného množství a počtu dostupného množství produktu se u nepřihlášeného zákazníka provádí před vstoupením do procesu objednání zboží. Tedy před

vstupem do procesu vytvoření objednávky se odešle dotaz na server, aby se zjistilo, jestli je vyžádané množství produktu dostupné. Pokud požadované množství u jednoho nebo více produktů přesahuje dostupné množství na skladě, tak je uživateli zobrazeno upozornění o nedostupnosti zvoleného množství produktu. U registrovaného a přihlášeného zákazníka se vytvoří objednávka ve stavu nákupní košík a při jakékoliv změně množství objednávaného produktu probíhá kontrola dostupnosti. Finálně, odečtení počtu produktů na skladě proběhne až v případě úspěšného objednání objednávky[27].

7.2.3 Nákupní košík

Nákupním košíku slouží pro zobrazení produktů, které sem byly vloženy uživatelem, a také pro zobrazení ceny všech produktů v košíku. Produkty jsou zobrazeny jako seznam produktů, který je seřazen pod sebe dle názvu produktů. V seznamu je u jednotlivých produktů zobrazena cena za kus, množství a celková cena za všechny kusy produktu. Pod seznamem produktů se v levé části stránky nachází tlačítko „Pokračovat v nakupování“, které uživatele přesměruje na procházení produktů a v pravé části stránky se nachází tabulka s celkovou cenou všech produktů v košíků a tlačítko „Vytvořit objednávku“ pro zahájení procesu vytvoření objednávky. Jak již bylo zmíněno, při kliknutí na tlačítko „Vytvořit objednávku“ dojde ke kontrole dostupnosti zvoleného množství produktů.

[27] Objednávka je zaevidována v systému a nachází se ve stavu přijato.



Obrázek 34 – Nákupní košík

U jednotlivých produktů se také nacházejí ovládací prvky pro zvětšení množství nebo pro odebrání produktu z košíku. Při kliknutí na název nebo obrázek produktu je následně zobrazen detail produktu.

7.2.4 Proces vytvoření objednávky

Tento proces se dá implementovat více způsoby. Celý proces vytváření objednávky se dá implementovat na jednu stránku, nebo rozdělit na jednotlivé kroky, mezi kterými může zákazník libovolně přepínat. V případě jednotlivých kroků jsou po zákazníkovi postupně vyžadovány jednotlivé informace jako jsou: způsob dopravy, typ platby a fakturační a doručovací adresa. Před odesláním objednávky se ještě většinou zobrazuje souhrn objednávky, kde si zákazník může zkontrolovat, zdali zadal veškeré vyžadované informace správně. Jak již bylo zmíněno, jednotlivé kroky nemusí být nutně rozděleny do jednotlivých stránek, ale lze je sloučit a mít tak vše na jedné stránce. Při použití toho řešení je důležité dbát na přehledné rozložení stránky, aby se v ní zákazník zvládl orientovat a neztrácel se v ní.

V této webové aplikaci je implementován proces vytvoření objednávky na jednu stránku. Stránka vytvoření objednávky je rozložena tak, aby vše bylo přehledné a viditelné. V horní části je umístěna mapa s oblastí doručení a výběr dopravy a způsob platby pomocí tzv. radio tlačítek. Jde o skupinu možností, ze kterých lze kliknutím zvolit pouze jednu z možností. V tomto případě tedy zákazník volí jednu možnost ze způsobu dopravy a jednu možnost ze způsobu platby. V případě, kdy zákazník nezvolí z nabízených možností, je mu po kliknutí na tlačítko „Dokončit objednávku“ zobrazen varovný text, aby zvolil způsob dopravy a platby.

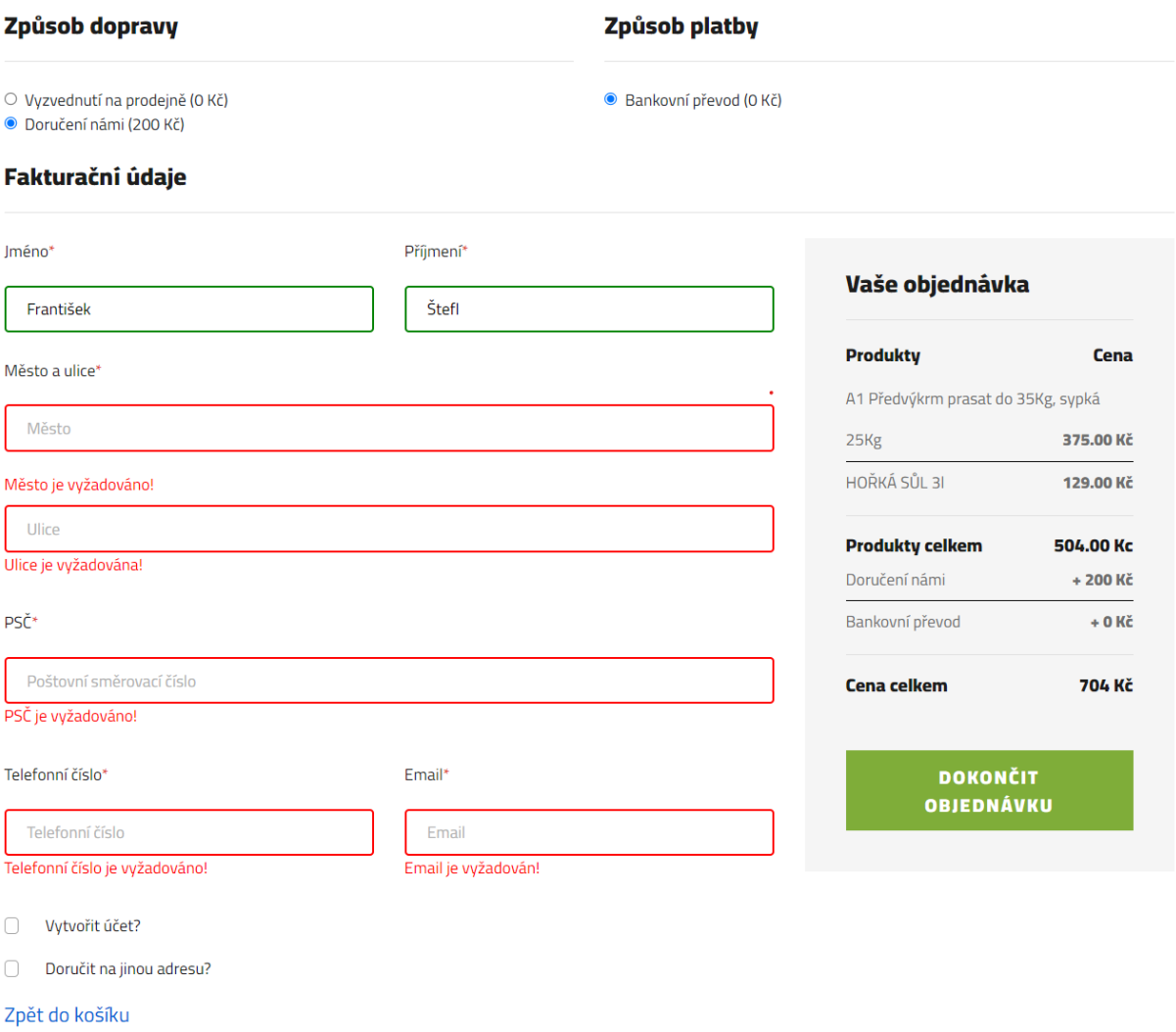
Pod výběrem způsobu dopravy a platby se nachází formulář pro vyplnění fakturační adresy objednávky spolu s tabulkou, kde jsou vypsány jednotlivé produkty objednávky s jejich cenou, celkovou cenou objednávky a tlačítkem pro dokončení objednávky. Pro registrovaného a přihlášeného zákazníka se zde navíc zobrazuje rozbalovací menu s uloženými adresami, ze kterých si zákazník může vybrat požadovanou adresu a ta se mu automaticky předvyplní do příslušných polí ve formuláři. Jednotlivá pole mají nad sebou zobrazený název, který říká, co se do daného pole má vyplnit. Vyžadovaná pole ve formuláři mají za svým názvem červenou hvězdičku[28].

Níže pod formulářem adresy je umístěno jedno nebo dvě zaškrtávací políčka. V případě nepřihlášeného zákazníka se zde nachází zaškrtávací políčko „Vytvořit účet?“, po jehož zaškrtnutí se zobrazí dvě vstupní pole pro vložení hesla a potvrzení hesla. S touto možností se následně, při dokončení objednávky, vytvoří nový účet. Druhé zaškrtávací políčko slouží ke zobrazení formuláře pro vyplnění jiné doručovací adresy. Toto umožňuje doručení objednávky na jinou adresu, než je fakturační. Pod zaškrtávacími políčky je umístěno tlačítko „Zpět do košíku“ pro vrácení se do košíku.

Co se validace hodnot týče, tak veškerá vstupní pole jsou validována jak na straně klienta, tak na straně serveru. Při každé interakci s polem dojde ke zkontrolování zadané hodnoty. Pokud je hodnota validní, tak je vstupní pole ohraničeno zelenou barvou. V případě zadání nevalidní hodnoty je pole ohraničeno červenou barvou a pod polem dojde ke zobrazení červeného textu informující zákazníka, proč je vložená hodnota nevalidní. Tato validace je řešena na straně klienta za pomoci funkcí psaných v programovacím jazyce JavaScript, který lze na straně klienta zakázat a tím tedy i znemožnit validaci dat na straně klienta. Z toho důvodu je validace

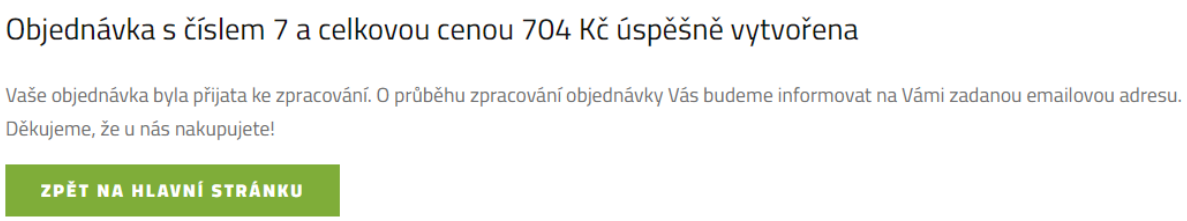
[28] V dnešní době standard označování vyžadovaných polí.

dat řešena i na straně serveru, kde se veškerá příchozí data od klienta validují před jejich dalším zpracováním. Dalším důvodem validace dat na straně serveru je, že obdržená data od klienta můžou být pozměněna útočníkem, který komunikaci meze klientem a serverem odposlouchává.



Obrázek 35 – Vytvoření objednávky

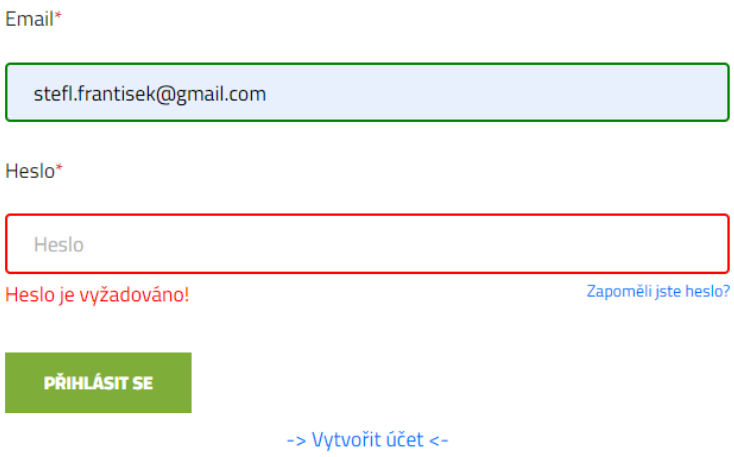
Po úspěšném dokončení objednávky se zákazníkovi zobrazí zpráva o úspěšném přijetí objednávky a na zadanou emailovou adresu mu přijde potvrzení o přijetí objednávky. Pokud zákazník zvolil jako způsob platby bankovní převod, přijde mu i email s požadavkem o uhrazení platby a číslem bankovního účtu, kam danou částku zaslat.



Obrázek 36 – Potvrzení vytvoření objednávky

7.2.5 Přihlašování a registrace uživatele

Zákazníci mají možnost si pomocí procesu registrace vytvořit uživatelský účet. Při registraci nového účtu je po uživateli vyžadována validní emailová adresa, heslo a potvrzení hesla. Doplňující informace jako jsou: jméno a příjmení, si může uživatel později doplnit ve správě profilu. Po úspěšné registraci se může zákazník přihlásit na svůj účet za pomoci přihlašovacího formuláře, do kterého zákazník vyplní emailovou adresu a heslo a následně dojde k autentizaci uživatele. V případě zapomenutí hesla si může uživatel zažádat o obnovení hesla. Proces obnovení hesla je zahájen odesláním formuláře s vyplněnou emailovou adresou účtu na server, kde dojde k ověření existence dané uživatelského učtu. Pokud je zadaná emailová adresa evidována v systému jako registrovaný účet, je na tuto adresu odeslán email s odkazem obsahující validační token, který přesměruje uživatele na stránku pro zadání nového hesla. Vygenerovaný validační token má danou dobu expirace, kdy po vypršení doby expirace nelze přistoupit na stránku pro vložení nového hesla.



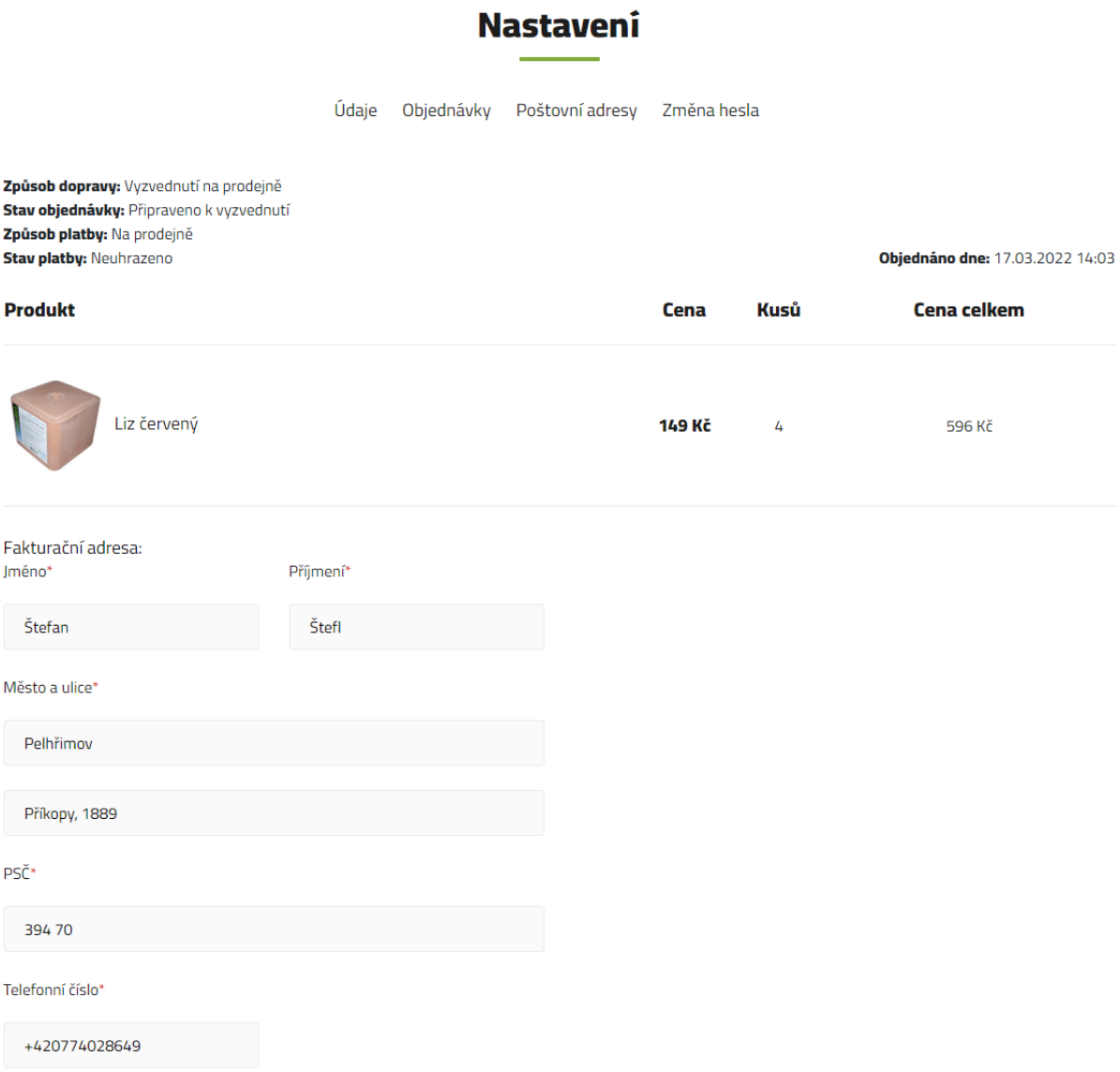
Obrázek 37 – Přihlašovací formulář

7.2.6 Správa profilu

Profil uživatele se skládá ze čtyř segmentů: údaje, objednávky, poštovní adresy a změna hesla. Tyto jednotlivé segmenty umožňují správu dat, která odpovídají názvům zmíněných segmentů.

První ze segmentů, správa údajů, umožňuje uživateli nastavit jméno a příjmení. Jedná se o velmi jednoduchý formulář, jehož vstupní pole nejsou vyžadována. Zajímavějším formulářem je změna hesla, který se nachází v segmentu změna hesla. Tento formulář obsahuje tři vstupní pole: aktuální heslo, nové heslo a potvrzení hesla. Všechna tři vstupní pole musí být vyplněna, pokud chce uživatel provést změnu hesla. Po odeslání formuláře na server se nejprve porovná odeslané aktuální heslo s evidovaným heslem v systému. Pokud odeslané heslo odpovídá tomu evidovanému v systému, provede se změna hesla a uživateli je zobrazena zpráva o úspěšném změnění hesla. V opačném případě, kdy se vyplněné aktuální heslo a uložené heslo v databází neshodují, se neprovede žádná operace, ale zpráva o úspěšném změnění hesla se i tak zobrazí uživateli.

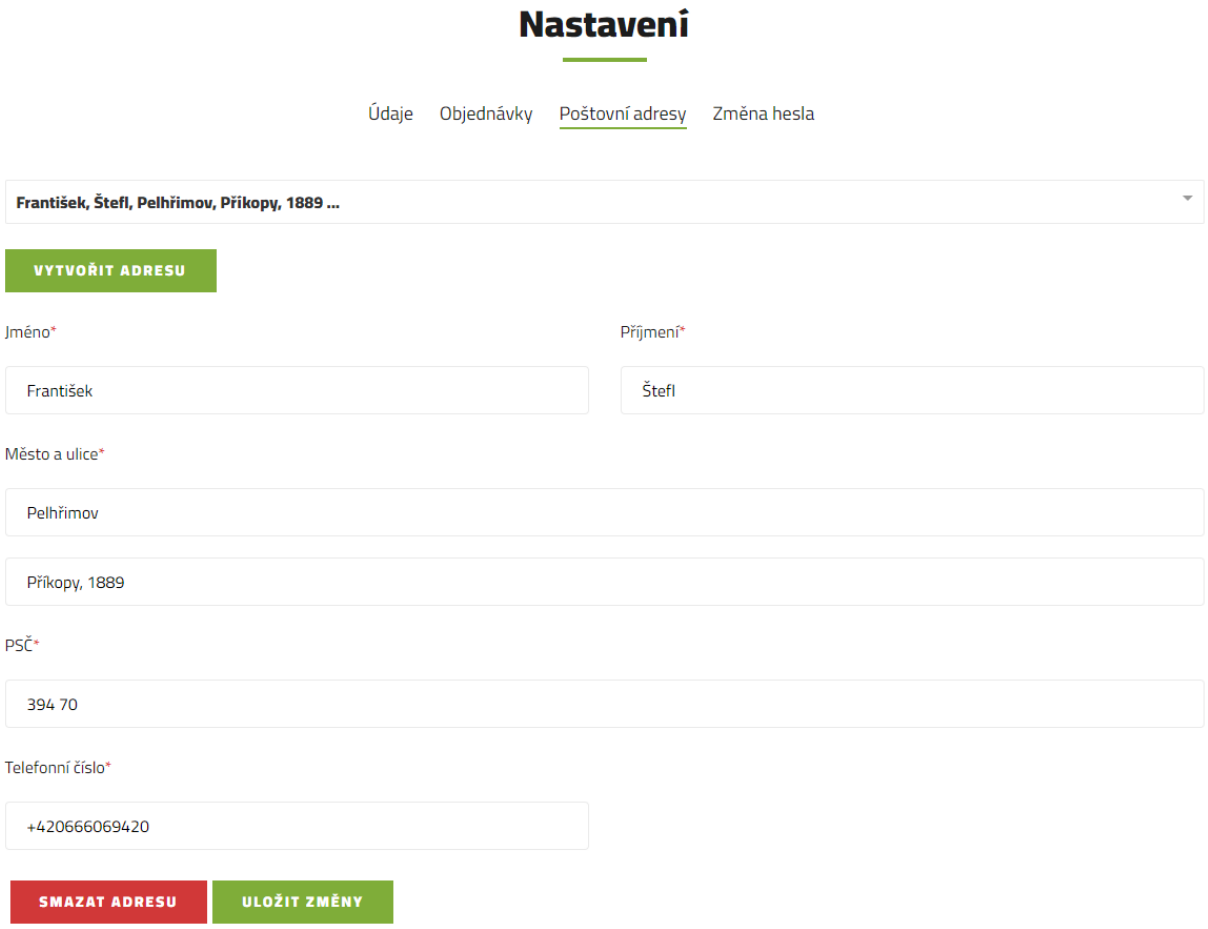
Dále se v profilu uživatele nachází správa objednávek, která slouží pro zobrazení vytvořených objednávek uživatelem. Jde o tabulku se záhlavím, kde jednotlivé řádky reprezentují objednávky. Objednávky v tabulce jsou v základu řazeny dle data vytvoření objednávky, ale záznamy lze dále řadit i podle celkové ceny objednávky. Po kliknutí na řádek v tabulce dojde k přesměrování na detail objednávky. Na stránce s detailem objednávky jsou zobrazeny veškeré důležité informace o objednávce jako jsou: stavy objednávky, adresa/y (fakturační a popř. i doručovací) a jednotlivé produkty objednávky. Kliknutím na jednotlivé produkty objednávky dojde k přesměrování na detail produktu.



Obrázek 38 – Detail objednávky uživatele

Posledním segmentem je správa adres, která umožňuje vytváření, upravování a mazání doručovacích či fakturačních adres. Stránka se skládá z rozbalovacího menu, které obsahuje existující adresy (pokud si uživatel nějaké vytvořil) a tlačítka pro vytvoření nové adresy. Kliknutím na tlačítko pro vytvoření nové adresy dojde ke zobrazení modálového menu s formulářem pro vyplnění údajů adresy. Jednotlivá pole formuláře jsou vyžadována a hodnoty v nich validovány. Pokud uživatel nezadá nebo zadá nevalidní hodnoty, dojde ke zobrazení varovného textu a formulář nepůjde odeslat. Po úspěšném vytvoření a uložení adresy do systému lze adresu vybrat ve zmíněném rozbalovacího menu. Následně se pod tlačítkem pro vložení nové adresy zobrazí vyplněný formulář s hodnoty zvolené adresy. Takto lze adresu upravit nebo smazat ze systému. Při upravování adresy jsou veškerá pole validována a v případě

zadání nevalidních hodnot se zobrazí varovný text a adresa s nevalidními hodnoty nepůjde uložit. Při zvolení možnosti smazání adresy se na serveru provede kontrola, zdali na tuto adresu neexistuje objednávka. Pokud se v systému nachází objednávka s touto adresou, provede se tzv. soft delete[29]. V opačném případě je adresa zcela smazána z databáze.



Obrázek 39 – Úprava uživatelské adresy

[29] Smazání záznamu v databázi příznakem. Záznam je stále uchován v databázi.

8 TESTOVÁNÍ A NASAZENÍ

Testování je nedílnou součástí procesu vývoje aplikace a patří do tzv. dobrých praktik vývoje softwaru, ke kterým dále patří objektové programování, používání vícevrstvých architektur apod. Na úvod je dobré zmínit, že testování je praktikou, kterou není nutné dodržovat, ale bezpochyby jde o důležitou a přínosnou praktiku, která vede k vytváření lepšího a funkčního kódu. V případě elektronického obchodu by se mělo provádět pravidelné testování i po jeho nasazení. [44]

V průběhu vývoje se testování zaměřuje především na správnou funkcionalitu jednotlivých částí aplikace či správný zápis kódu. Zde se např. používají jednotkové testy. Tyto testy především testují třídy a jejich metody, kdy jednotlivé metody vykonají s požadovanými, ale od sebe se lišícími parametry a kontrolují, zdali jsou výstupy těchto metod korektní. Tyto testy nemá smysl vytvářet pro metody, které provádí např. jednoduché dotazy do databáze a kontrolovat, zdali došlo k vrácení očekávaných dat. Smysl má testovat např. metodu, která se stará o kalkulaci celkové ceny objednávky. Takový test otestuje, zdali metoda provádí správný výpočet celkové ceny pro různé varianty objednávek. Díky těmto testům je programátor schopen zjistit, zdali všechny metody fungují správně po provedení nějaké úpravy kódu či refaktorování. [44]

Dalším typem testů jsou akceptační testy, které testují případy užití jako je např. registrace zákazníka nebo vytvoření objednávky. Tyto testy tedy testují specifickou logiku aplikace a testuje se výstup aplikace nikoliv výstup jedné specifické části kódu. Dále integrační testy, které se zaměřují na testování správné funkčnosti komponent jako jsou např. infrastruktura nebo databáze. Tyto testy se např. používají pro testování REST API. V daném testu se provede vytvoření dotazu na odpovídající přístupový bod aplikace a ověření správnosti vrácených dat. Posledním typem testů jsou systémové testy, které testují aplikaci jako funkční celek. Tento typ testů je používán v konečné fázi vývoje a dochází při něm k ověření aplikaci z pohledu zákazníka. Jednotlivé testy simulují kroky, které mohou v praxi nastat. [44]

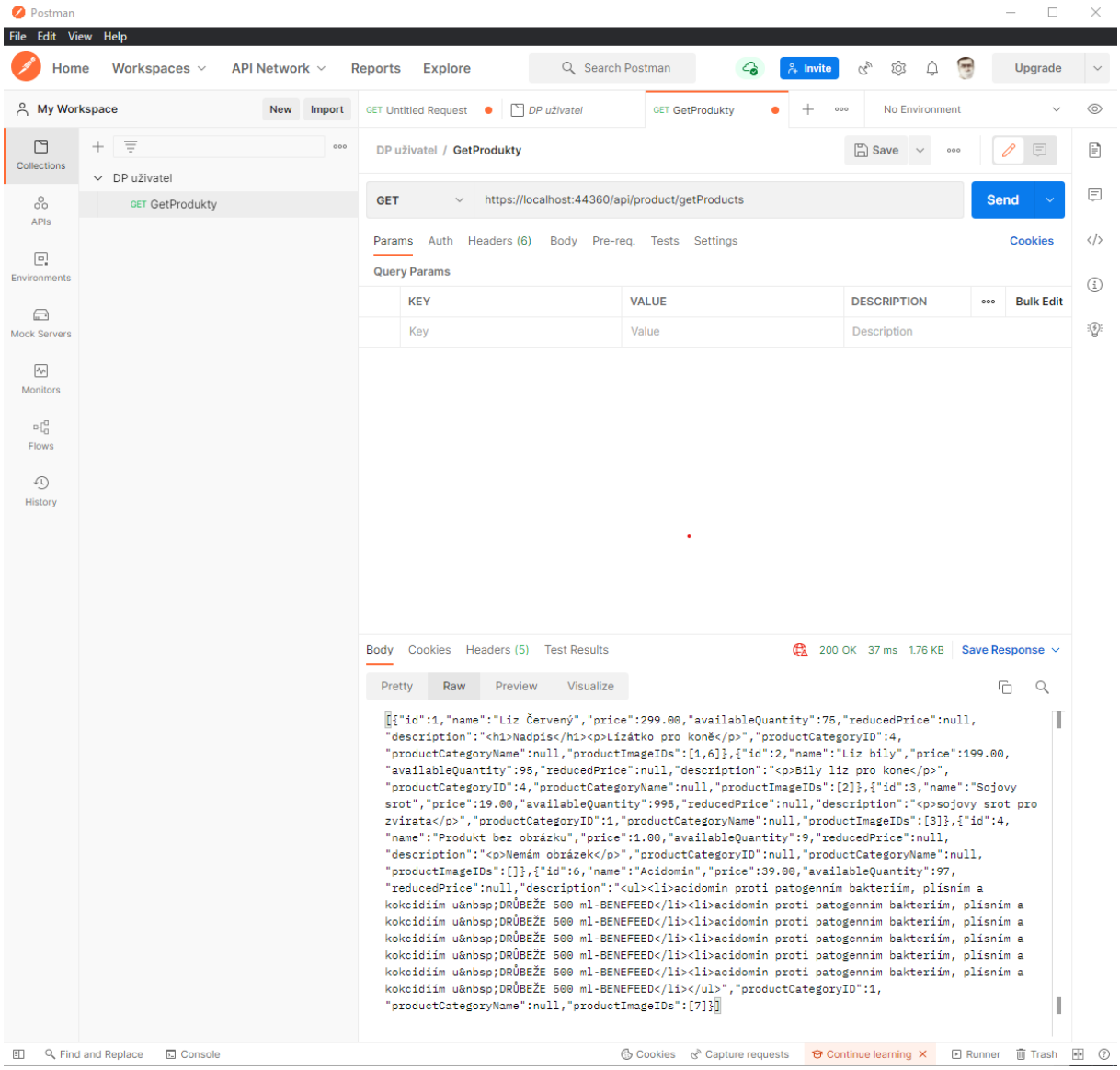
Z důvodu menší rozsáhlosti a důrazu na včasné dokončení elektronického obchodu nejsou v aplikaci vytvořeny žádné ze zmíněných testů. Veškeré testování a ověřování správné funkčnosti kódu probíhalo za pomocí debugovacího režimu. V případě testování API aplikace byly použity nástroje Postman a Fiddler.

8.1 Postman

Postman je multiplatformní aplikace, která umožňuje návrh API, monitorování HTTP dotazů a testování. Postman nabízí velmi přehledné grafické rozhraní, zpříjemňuje a usnadňuje jeho používání. Dále obsahuje velké množství jednoduchých ale i pokročilých funkci pro navrhování API, monitorování příchozích i odchozích HTTP dotazů, testování a vytváření základních ale i pokročilejších HTTP dotazů. Tento nástroj byl především použit k vytváření HTTP dotazů, které byly následně posílány na odpovídající přístupové body REST API aplikace. Tímto způsobem bylo testováno API aplikace.

Při vytváření HTTP dotazu je možné si zvolit dotazovací metodu Get, Post, Delete atd. a vložit autorizační hlavičku dotazu u které se dále volí typ autorizace[30], anebo přidat data do těla dotazu[31]. Hotový HTTP dotaz je možné v Postmanu uložit do kolekce pro budoucí opakované použití jako šablonu nebo odeslání. Po odeslání dotazu dojde následně ke zobrazení odpovědi ze serveru nebo vypsání odpovídajícího chybového hlášení.

[30] API Key, Bearer Token a další.
[31] Zde je možnost si zvolit formát dat.

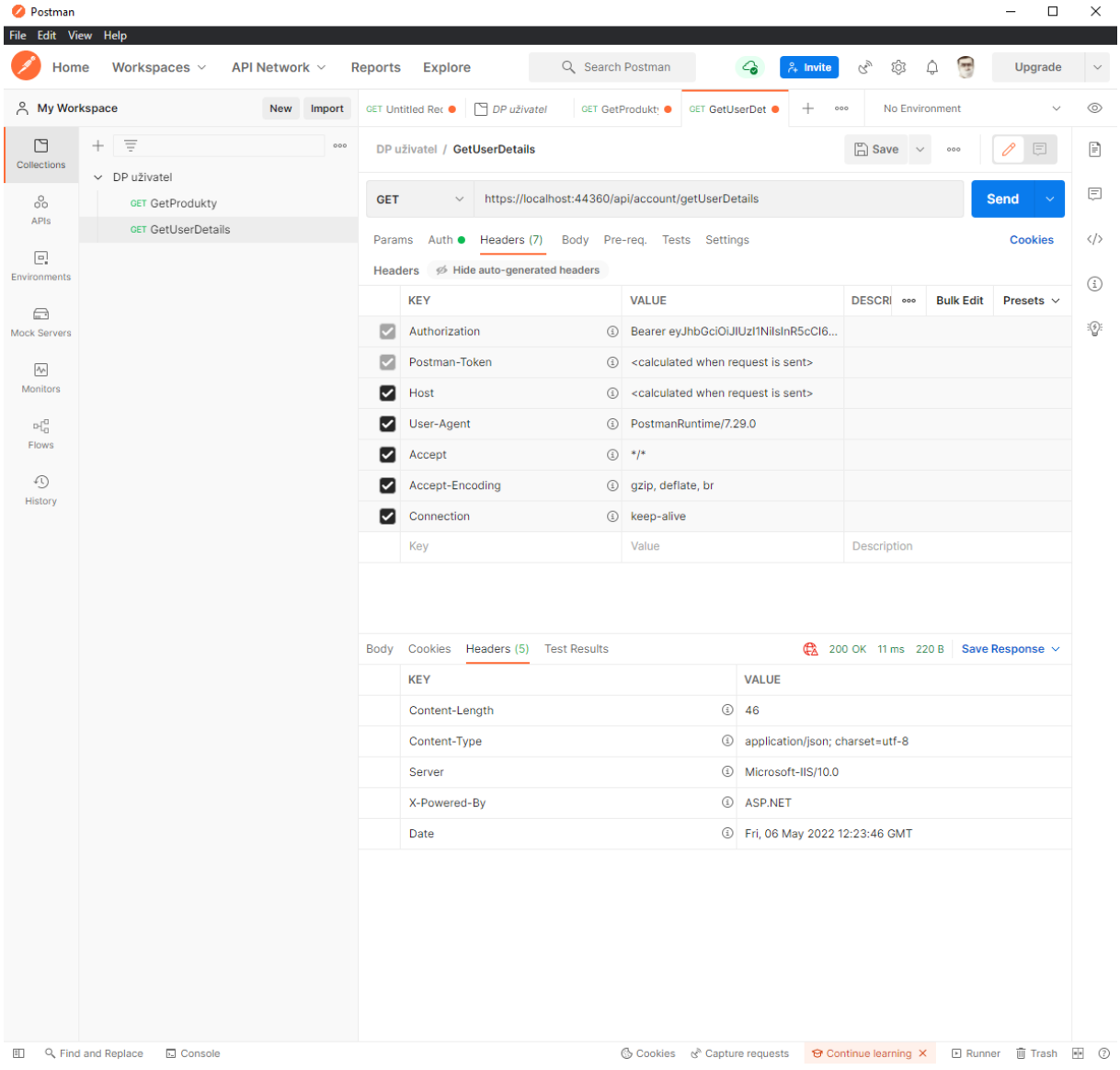


Obrázek 40 – Postman HTTP Get dotaz

Na obrázku 32 je ukázka odeslání jednoduchého HTTP Get dotazu na server. V levé části jsou pod sebou zobrazeny funkce Postmanu (Collections, APIs, Enviroments atd.). Pro vytváření HTTP dotazů je zvolena záložka Collections, ve které je možné si vytvořené HTTP dotazy pojmenovat a uložit pro budoucí opakované použití. V prostřední části jsou zobrazeny veškeré důležité informace a vstupní pole HTTP dotazu. V horní části se nachází vstupní pole pro URL přístupového bodu API a rozbalovací menu pro zvolení dotazovacích metod. Níže je umístěno navigační menu pro vkládání parametrů[32], stanovení autorizační metody, výpis obsahu

[32] Parametry se zapisují do URL adresy a jde o hodnoty, které je příjemce schopen číst. Příkladem parametru v URL adrese může být Id produktu.

hlavičky, vložení dat do těla HTTP dotazu a další. Nakonec ve spodní části se nachází výpis HTTP odpovědi ze serveru, kde je možné si prohlédnou obsah těla i hlavičky HTTP dotazu.



Obrázek 41 – Postman HTTP Get dotaz s autorizací

Ne obrázku 33 je zobrazeno odeslání HTTP dotazu na zabezpečený API přístupový bod. HTTP dotaz musí obsahovat validní JWT v hlavičce, aby se provedla odpovídající operace na serveru pro daný API přístupový bod. Pokud v HTTP dotazu nebude JWT, dojde k vrácení odpovědi, která informuje klienta[33] o zamítnutí přístupu. Postman umožňuje volbu mezi několika typy řešení autorizace z nichž jeden je Bearer authentication (též nazývaný jako token

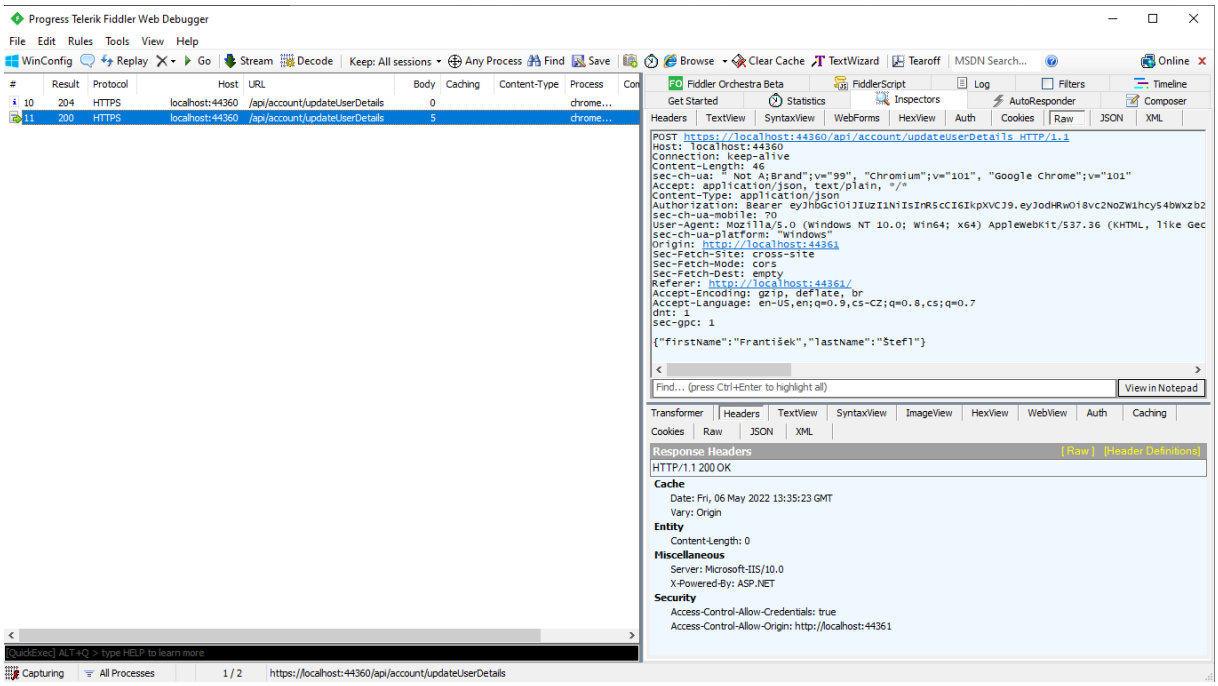
[33] Odesílatel HTTP dotazu, který přistupuje na daný přístupový bod.

authentication). Tento typ autorizace byl použit a lze jej vidět v prvním řádku obsahu hlavičky HTTP dotazu.

Obrázky výše znázorňují použití aplikace Postman pro testování API aplikace a odstraňování chyb. Nejsou zde uvedeny veškeré API přístupové body z důvodu jejich většího počtu. Pro představu a vysvětlení testování za pomoci aplikace Postman stačí ony dva příklady užití.

8.2 Fiddler

Dalším programem, který byl použit pro testování a odstraňování chyb je Fiddler. Jedná se o debugovací nástroj pro logování HTTP dotazů, které je následně možné detailně prohlížet. Fiddler dále nabízí pokročilejší funkce jako jsou: dešifrování HTTPS dotazů, mockování a editace dotazů i odpovědí, vytváření dotazů pro testování REST API a další. Pro účely testování API a odstraňování chyb byly použity základní funkce Fiddleru pro zachytávání HTTP dotazů za účelem detailního zobrazení jejich obsahu. V případě výskytu nějaké chyby byl Fiddler použit k zachytávání HTTP dotazů, které byly následně zkoumány. Typickým příkladem je zkoumání vytvořených HTTP dotazů na klientské straně, protože debugování JavaScriptového kódu v prohlížeči je nešikovné a nefunguje tak dobře, jako debugování kódu na straně serveru za pomoci debugovacích nástrojů obsažených ve Visual Studiu.



Obrázek 42 – Debugovací nástroj Fiddler

V aplikaci Fiddler se v levé části okna vypisují zachycené HTTP dotazy. Fiddler automaticky zachycuje veškeré příchozí i odchozí HTTP dotazy, což může být nežádoucí. Z tohoto důvodu lze ve Fiddleru vytvořit pravidla podle kterých se budou vypisovat pouze odpovídající HTTP dotazy. Zachytávání HTTP dotazů lze i pozastavit. V pravé části okna jsou zobrazeny veškeré informace o vybraném HTTP dotazu.

8.3 Finální testování

Finální testování proběhlo po nasazení aplikace za pomoci prostředí Microsoft Azure Web Sites. Jedná se o platformu od firmy Microsoft pro hostování a spravování webových aplikací. Tato platforma nabízí několik plánů služeb, které jsou naceněny dle velikosti výpočetního výkonu nebo velikosti dostupného uložiště. Plán určený pro testování nasazené aplikace je zcela zdarma a výpočetní výkon a velikost uložiště byl zcela dostačující pro potřeby elektronického obchodu.

Po úspěšném nasazení a zprovoznění elektronického obchodu do testovacího prostředí došlo k otestování aplikace firmou VKS Rynárec. Zaměstnanci firmy byli seznámeni s aplikací, kterou následně během několika dní používali. Cílem tohoto testování bylo zjistit, jak na ně elektronický obchod působí, zdali je přehledný a snadno ovladatelný, jaké jsou nedostatky nebo naopak co přebývá, a nakonec zdali vše funguje dle požadavků.

Výsledkem testování bylo několik drobných problémů, které se týkaly především stylizace textů a chybně uvedeného textu. Konkrétně šlo o problém s výraznou červenou barvu celkové ceny v košíku, červené ikony křížku pro odstranění produktu z košíku a špatná dodací lhůta u detailu produktu. Dle této zpětné vazby byla barva celkové ceny v košíku změněna na šedou, barva červené ikony křížku pro odebrání produktu z košíku na černou a hodnota dodací lhůty upravena na hodnotu požadovanou firmou. Po aplikování oprav byli zaměstnanci firmy spokojeni jak se vzhledem, tak s fungováním webové aplikace.

Po dokončení testování došlo v Azure Web Sites k vytvoření nového plánu, který obsahuje více výpočetního výkonu a většího uložiště pro aplikaci a databázi. Co se volby plánu s prostředky týče, byl zvolen možný nejlevnější pro produkční nasazení. Bylo tak učiněno z důvodu obsáhlosti webové aplikace, velikosti firmy a z očekávané malé návštěvnosti elektronického obchodu. Vybraný plán s prostředky se dá vždy jednoduše změnit, a v případě vysoké zátěže webové aplikace a nedostatečného výpočetního výkonu dojde ke změně plánu s prostředky.

ZÁVĚR

Cílem této diplomové práce bylo vytvoření elektronického obchodu na míru pro firmu VKS Rynárec, s.r.o. se specifickými požadavky, který se bude skládat z administrátorského a uživatelského rozhraní. Tohoto hlavního cíle bylo dosaženo. Výsledná webová aplikace odpovídá většině zadaným požadavkům, obsahuje administrátorské i uživatelské rozhraní a jsou v ní implementovány předem vytyčené funkce jako prohlížení produktů, vkládání produktů do košíku, vytváření objednávek a další.

Bohužel, finální verze elektronického obchodu neobsahuje funkcionalitu pro vkládání akcí. Tato funkce nebyla implementována, neboť firma VKS Rynárec, s.r.o. tuto funkcionalitu zamítla z důvodu jejího nevyužití. Dalším nedostatkem elektronického obchodu je řešení vztahu mezi produktem a kategorií a také unárního vztahu kategorie. Zde je problém v tom, že produkt může mít pouze jednu kategorii, a to vede k vícenásobnému vládání jednoho produktu pokaždé s jinou kategorií. U unárního vztahu kategorie nastává problém v nemožnosti přiřadit kategorii několik potomků a rodičů. Toto vede ke vkládání redundantních kategorií. Příkladem je vložení kategorie s názvem Sypká pro každou nadřazenou kategorii zvlášť, a tedy bude v databázi opakující se kategorie s názvem Sypká.

I přes zmíněné nedostatky elektronického obchodu jsem názoru, že výsledná implementace dopadla úspěšně. Firma je s aplikací spokojena a bezproblémově ji využívá. Firma je také spokojena se celkovým vzhledem a funkcemi elektronického obchodu, a především z pokračující spolupráce s autorem diplomové práce, který bude nadále spravovat a rozšiřovat aplikaci dle požadavků firmy. Dle vyjádření firmy jsou i zákazníci s elektronickým obchodem spokojeni.

Nasazený elektronický obchod budu nadále spravovat a rozšiřovat. Konkrétně je naplánováno změna datového modelu tak, aby bylo umožněno produktu přiřadit několik kategorií oproti stávajícímu řešení, kdy je možné produktu přiřadit pouze jednu kategorii. Dále pak umožnění přidělení kategorii n počtu nadřazených kategorií a podkategorií. Další možné rozšíření je přidání možnosti platby kartou, která je v dnešní době jednou z nejvyužívanějších platebních metod. Co se administrátorského rozhraní týče, zde by se dala přidat funkcionalita generování faktur ve formátu pdf. Pokročilejším rozšířením by bylo napojení elektronického obchodu na ekonomický a informační systém Pohoda, který firma VKS Rynárec, s.r.o. používá.

Vytváření této práce přineslo cenné zkušenosti a ponaučení z nedostatků a chyb. Velmi přínosný byl především celkový průběh vývoje aplikace od prvotní konzultace se zákazníkem, přes návrh aplikace, její implementaci, a nakonec testování a nasazení. Dále díky implementaci elektronického obchodu nabyl autor nových znalostí ve vytváření webových aplikací za použít knihovny React a frameworku Blazor, které bude autor nadále využívat. Nakonec tato diplomová práce může posloužit jako návod či inspirace pro postup navrhování a implementování méně rozměrného elektronického obchodu.

POUŽITÁ LITERATURA

[1] *VKS Pohledští Dvořáci a.s.: Krmné směsi, služby zemědělství* [online]. Havlíčkův Brod, 2022 [cit. 2022-05-16]. Dostupné z: https://www.vkshb.cz/maloobchod.html

[2] *Ryhos, s.r.o.: Ráj zahrádkářů a chovatelů* [online]. 2011 [cit. 2022-05-16]. Dostupné z: https://www.ryhos.cz/e-shop

[3] *Agrotree: Krmiva a pěstitelské potřeby* [online]. 2022 [cit. 2022-05-16]. Dostupné z: https://www.agrotree.cz/

[4] *Krmné směsi Kvidera* [online]. 2017 [cit. 2022-05-16]. Dostupné z: https://www.krmnesmesikvidera.cz/

[5] Welcome to the Visual Studio IDE. In: *Microsoft technical documentation* [online]. New York: Microsoft, 2022 [cit. 2022-05-13]. Dostupné z: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022

[6] PETROUTSOS, Evangelos. *Myslíme v jazyku Visual Basic .NET*. 1st ed. Praha: Grada, 2003. Knihovna programátora (Grada). ISBN 80-247-0371-8.

[7] STRAUSS, Dirk. *Getting Started with Visual Studio 2019: Learning and Implementing New Features*. 1st ed. New York: Apress, 2019. ISBN 978-1484254486.

[8] Visual Studio - Úvod do vývojového prostředí. In: *ITnetrowk.cz* [online]. Michal Žůrek, 2018 [cit. 2022-05-13]. Dostupné z: https://www.itnetwork.cz/csharp/visual-studio/tutorial-visual-studio-uvod

[9] What is .NET? Introduction and overview. In: *Microsoft technical documentation* [online]. 2022 [cit. 2022-05-13]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/core/introduction

[10] ESPOSITO, Dino. *ASP.NET a ADO.NET: tvorba dynamických webových stránek*. 1st ed. Praha: Grada, 2003. Moderní programování. ISBN 80-247-0474-9.

[11] LANDER, Richard. Introducing .NET 5. In: *.NET Blog* [online]. 2019 [cit. 2022-05-13]. Dostupné z: https://devblogs.microsoft.com/dotnet/introducing-net-5/

[12] The Good and the Bad of .NET Framework Programming. In: *AltexSoft* [online]. 2021 [cit. 2022-05-13]. Dostupné z: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-net-framework-programming/

[13] ROTH, Daniel, Rick ANDERSON a Shaun LUTTIN. Overview to ASP.NET Core. In: *Microsoft technical documentation* [online]. 2022 [cit. 2022-05-13]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0

[14] ASP.NET overview. In: *Microsoft technical documentation* [online]. 2022 [cit. 2022-05-13]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/overview

[15] A tour of the C# language. In: *Microsoft technical documentation* [online]. 2022 [cit. 2022-05-13]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[16] GRIFFITHS, Ian. *Programming C# 8.0: Build Cloud, Web, and Desktop Applications*. 1st ed. Farnham: O'Reilly Media, 2020. ISBN 978-1492056812.

[17] Nullable reference types. In: *Microsoft technical documentation* [online]. 2022 [cit. 2022-05-13]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/nullable-references

[18] LIANG, Mia. Understanding Object-Relational Mapping: Pros, Cons, and Types. In: *AltexSoft* [online]. 2021 [cit. 2022-05-13]. Dostupné z: https://www.altexsoft.com/blog/object-relational-mapping/

[19] JENSEN, Andrew. Identify Object-Relational Mapping (ORM) Tools for .NET. In: *OpenClassrooms* [online]. 2019 [cit. 2022-05-13]. Dostupné z: https://openclassrooms.com/en/courses/5671811-implement-a-relational-database-with-asp-net-core/6588450-identify-object-relational-mapping-orm-tools-for-net

[20] Entity Framework Core. In: *Microsoft technical documentation* [online]. 2021 [cit. 2022-05-13]. Dostupné z: https://docs.microsoft.com/en-us/ef/core/

[21] ASP.NET Core Blazor. In: *Microsoft technical documentation* [online]. 2022 [cit. 2022-05-13]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-6.0

[22] HIMSCHOOT, Peter. *Microsoft Blazor: Building Web Applications in .NET*. 2nd ed. New York: Apress, 2020. ISBN 978-1484259276.

[23] *React: A JavaScript library for building user interfaces* [online]. Copyright © 2022 Meta Platforms, Inc. [cit. 2022-05-13]. Dostupné z: https://reactjs.org/

[24] Dotnet Days CZ: React (Pavel Kříž). In: *YouTube* [online]. 2019 [cit. 2022-05-13]. Dostupné z: https://youtu.be/8fxXJLtUCdw

[25] *JWT: Introduction to JSON Web Tokens* [online]. [cit. 2022-05-13]. Dostupné z: https://jwt.io/introduction

[26] Get Started with JSON Web Tokens: All you wanted to know about JSON Web Tokens but were afraid to ask. In: *Auth0* [online]. Bellevue, c2013–2016 [cit. 2022-05-13]. Dostupné z: https://auth0.com/learn/json-web-tokens/

[27] ROSENCRANCE, Linda a Brein MATTURRO. Remote Procedure Call (RPC). In: *TechTarget* [online]. c2019-2022 [cit. 2022-05-13]. Dostupné z: https://www.techtarget.com/searchapparchitecture/definition/Remote-Procedure-Call-RPC

[28] Why gRPC?. In: *GRPC* [online]. 2022 [cit. 2022-05-13]. Dostupné z: https://grpc.io/

[29] NEWTON-KING, James. Introduction to gRPC on .NET. In: *Microsoft technical documentation* [online]. 2022 [cit. 2022-05-13]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/grpc/?view=aspnetcore-6.0

[30] Introduction to gRPC: An introduction to gRPC and protocol buffers. In: *GRPC* [online]. 2021 [cit. 2022-05-13]. Dostupné z: https://grpc.io/docs/what-is-grpc/introduction/

[31] KOĎOUSKOVÁ, Barbora. Co je to API a jaké jsou možnosti jeho využití?. In: *Rascasone* [online]. Praha, 2022 [cit. 2022-05-13]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-api

[32] HANÁK, Drahomír. Stopařův průvodce REST API. In: *ITnetwork.cz* [online]. Praha, 2022 [cit. 2022-05-13]. Dostupné z: https://www.itnetwork.cz/programovani/nezarazene/stoparuv-pruvodce-rest-api/

[33] Spring Framework Overview. In: *Spring* [online]. San Francisco, c2002-2022 [cit. 2022-05-14]. Dostupné z: https://docs.spring.io/spring-framework/docs/current/reference/html/overview.html#spring-introduction

[34] Introduction. In: *Vue* [online]. c2014-2022 [cit. 2022-05-14]. Dostupné z: https://vuejs.org/guide/introduction.html

[35] *Strafelda.cz: PHP* [online]. [cit. 2022-05-14]. Dostupné z: https://www.strafelda.cz/php

[36] WALKER, Alyssa. SOAP Web Services Tutorial: What is SOAP Protocol? EXAMPLE. In: *Guru99* [online]. 2022 [cit. 2022-05-14]. Dostupné z: https://www.guru99.com/soap-simple-object-access-protocol.html

[37] RESELMAN, Bob. An Architect's guide to APIs: SOAP, REST, GraphQL, and gRPC. In: *RedHat* [online]. Releigh, 2020 [cit. 2022-05-14]. Dostupné z: https://www.redhat.com/architect/apis-soap-rest-graphql-grpc

[38] LOISEL, Jérôme. Soap vs Rest (Why comparing them is a nonsense). In: *OctoPerf* [online]. Aubagne, c2014-2022 [cit. 2022-05-14]. Dostupné z: https://octoperf.com/blog/2018/03/26/soap-vs-rest/#comparison-table

[39] ČÁPKA, David. Lekce 2 - UML - Use Case Diagram. In: *ITnetrowk.cz* [online]. Praha, 2018 [cit. 2022-05-14]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram

[40] ŘEPA, Václav, Václav SYNÁČEK, Petr HAMERNÍK, Ondřej DIVIŠ a Ivo KLIMEŠ. *Metodika vývoje informačního systému s pomocí nástroje Power Designer* [online]. Praha, 2006 [cit. 2022-05-14]. Dostupné z: http://www.sybase.panrepa.org/Metodika_vyvoje_IS_06_2006.pdf. Publikace. Vysoká škola ekonomická v Praze.

[41] MIKULÁŠKOVÁ, Petra a Mirek SEDLÁK. *Jak vytvořit úspěšný a výdělečný internetový obchod*. 1st ed. Brno: Computer Press, 2015. ISBN 978-80-251-4383-4.

[42] What is Code-First?. In: *Entity Framework Tutorial* [online]. 2022 [cit. 2022-05-15]. Dostupné z: https://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx

[43] Code-first gRPC services and clients with .NET. In: *Microsoft technical documentation* [online]. 2022 [cit. 2022-05-16]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/grpc/code-first?view=aspnetcore-6.0

[44] ČÁPKA, David. Lekce 1 - Úvod do testování softwaru v C# .NET. In: *ITnetwork.cz* [online]. 2017 [cit. 2022-05-16]. Dostupné z: https://www.itnetwork.cz/csharp/testovani/uvod-do-testovani-softwaru-v-csharp-net

PŘÍLOHY

PŘÍLOHA A – CD

K práci je přiloženo optické medium, které obsahuje zdrojové kódy vytvořené aplikace v rámci této diplomové práce.