

Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky



**Systém pro samoobslužná prodejní místa  
společnosti Do psí misky**

**BAKALÁŘSKÁ PRÁCE**

Studijní program: Aplikovaná informatika

Autor: Erich Pross

Vedoucí bakalářské práce: Ing. et Ing. Stanislav Vojíř, Ph.D.

Praha, květen 2024

## **Poděkování**

Zde bych rád poděkoval mému vedoucímu bakalářské práce Ing. et Ing. Stanislavu Vojířovi, Ph.D, za vstřícný přístup a potřebné rady při psaní této práce. Dále chci poděkovat zadavatelům úkolu, majitelům firmy Do psí misky, Tomášovi a Kateřině Milcovým, za jejich přístup k projektu a za užitečnou zpětnou vazbu při implementaci výsledku práce.

## **Abstrakt**

Bakalářská práce se zabývá návrhem, vývojem a implementací webové aplikace, která umožňuje zákazníkům firmy Do psí misky nakupovat v bezkontaktních samoobslužných prodejnách bez obsluhy či jiného personálu, a poskytuje základní možnosti správy jednotlivých prodejen.

Práce začíná analýzou dostupných řešení pro samoobslužná bezkontaktní prodejní místa na českém i zahraničním trhu, návrhem a analýzou požadavků a potřeb zadavatele, včetně všech funkcí, které by měla aplikace obsahovat. Následuje návrh architektury, včetně databáze, zabezpečení a uživatelského rozhraní. Pro vývoj a implementaci systému byla použita moderní řešení, jako je PHP framework Laravel, který nabízí řadu funkcí pro vývoj webových aplikací.

Výsledná aplikace umožňuje uživatelům vstoupit do prodejny pomocí naskenování QR kódu, následně procházet sortiment pomocí skenování čárových kódů, zobrazit si informace o produktech, které si mohou vložit do svého košíku, a za zboží bezpečně a jednoduše zaplatit pomocí online platby přímo z telefonu.

Po implementaci byla aplikace nasazena na server zadavatele a nyní se čeká na dokončení jednání ohledně platební brány, aby mohla být aplikace zavedena do ostrého provozu.

## **Klíčová slova**

obchod, bezkontaktní samoobslužná prodejna, Laravel, webová aplikace, QR kód, čárový kód, PHP

## **Abstract**

The bachelor's thesis focuses on the design, development, and implementation of a web application that allows customers of Do psí misky to shop in contactless self-service stores without assistance or other personnel, and provides basic store management capabilities.

The work begins with an analysis of available solutions for self-service contactless sales points in the Czech and international markets, followed by the design and analysis of the client's requirements and needs, including all the features the application should have. The design of the architecture follows, including the database, security, and user interface. Modern solutions were used for the development and implementation of the system, such as the PHP framework Laravel, which offers a range of features for web application development.

The resulting application allows users to enter the store by scanning a QR code, then browse the assortment by scanning barcodes, view information about the products, add them to their cart, and securely and easily pay for goods using online payment directly from their phone.

After implementation, the application was deployed on the client's server and is now awaiting the completion of negotiations regarding the payment gateway before the application can be put into full operation.

## **Keywords**

store, contactless self-service store, Laravel, web application, QR code, barcode, PHP

# **Obsah**

Úvod .....	7
1 Bezkontaktní samoobslužný prodej .....	8
2 Definice zadavatele .....	10
2.1 Funkční požadavky.....	12
2.2 Nefunkční požadavky.....	14
2.3 Dostupná řešení .....	15
2.3.1 Srovnání dostupných řešení .....	22
2.4 Využitelné prvky ostatních řešení.....	23
3 Návrh aplikace .....	24
3.1 Use case diagram.....	24
3.2 Mapy procesů.....	26
3.3 Návrh uživatelského rozhraní.....	35
3.3.1 Návrh desktopového rozhraní .....	35
3.3.2 Návrh mobilního rozhraní .....	40
3.4 Technické prvky implementace .....	45
4 Implementace .....	53
4.1 Struktura aplikace .....	53
4.2 Autorizace uživatelů.....	56
4.3 Zobrazení úvodní stránky .....	57
4.4 Přihlášení a registrace skrze lokální účet.....	60
4.5 Přihlášení a registrace skrze účet třetí strany.....	63
4.6 Správa prodejen .....	67
4.7 Správa zboží .....	75
4.8 Správa uživatelů.....	81
4.9 Vstup do prodejny.....	85
4.10 Nákup produktů .....	91
4.11 Úprava produktů v košíku .....	103
4.12 Ukončení nákupu .....	108
5 Nasazení aplikace .....	116
6 Testování aplikace .....	118
6.1 Testování podle předem definovaných scénářů .....	118
6.2 Testování výkonu aplikace.....	122
6.3 Testování bezpečnosti aplikace.....	123

7 Závěr .....	125
8 Použitá literatura.....	126

# Úvod

S rostoucím vývojem technologií a změnami v preferencích zákazníků v oblasti maloobchodu vzniká poptávka po inovativních přístupech k provozování prodejen. Automatizace prodeje a snížení počtu personálu v obchodech se stává stále populárnějším a nabízí potenciál zvýšení efektivity provozu a zároveň lepší zákaznický zážitek. Tento trend lze pozorovat například v nasazení objednávkových terminálů ve fastfoodech či samoobslužných pokladen v obchodních řetězcích.

Historie těchto obchodů sahá do nedávné minulosti a odráží rychlý technologický vývoj v tomto odvětví. Koncept byl silně rozvinut firmou Amazon, která otevřela první autonomní prodejnu v Seattlu již v roce 2016, nejdříve pro zaměstnance firmy Amazon, následně pro ostatní zákazníky v roce 2018 (Cheng, 2019). To však neznamená, že se v tomto odvětví nenachází více hráčů.

Cílem této bakalářské práce je návrh a následná implementace webové aplikace, která dovolí zákazníkům vstup do prodejny produktů. Do psí misky pomocí sdělení stále se měnícího QR kódu, zjišťování informací o produktech pomocí skenování čárových kódů na produktech a jejich následné vložení do online košíku a následná zaplacení, stále ve webové aplikaci.

Je potřeba překonat několik pomyslných překážek ve vývoji – je potřeba analyzovat požadavky zákazníka, jeho cíle a možnosti pro implementaci aplikace, ale také problémy jako je zajištění vstupu zákazníků do budovy, zahájení prodeje samotného zákazníkem, následné skenování kódů, ukončení nákupu, vzhled databáze zboží a prodejů a následná implementace toho všeho.

Je očekáváno, že se bude jednat o webovou aplikaci sepsanou v jazyce PHP s pomocí frameworku Laravel a s použitím prvků JavaScriptu. Od čtenáře této bakalářské práce je očekáváno, že má základní zkušenosti s vývojem webových aplikací a má základní znalost těchto jazyků.

Práce je segmentována do několika částí. Je potřeba zjistit dostupná řešení na trhu (a pokud se nějaké bude podobat kýženému výsledku, čím se inspirovat), následně je potřeba definovat možnosti, cíle a požadavky zákazníka, poté definovat workflow a funkčnost webové aplikace. Následuje implementace, testování a zavedení do testovacího provozu.

# 1 Bezkontaktní samoobslužný prodej

Původ bezkontaktních samoobslužných řešení sahá až do starověku, kdy byly vynalezeny první automaty, konkrétně automat na svěcenou vodu (R., 2014) (Smith, 2015). Od té doby se automaty vyvíjely a rozšiřovaly do dalších oblastí, v moderním podání se jedná např. o automaty na nápoje, nebo třeba i bitcoinové bankomaty, které se v poslední době vlivem růstu cen kryptoměn rozšiřují po celém světě. Specialitou v tomto odvětví jsou například automaty na vyčištění oblečení, rybářské návnady nebo dokonce i automaty automobily. Vývoj technologií umožnil těmto automatům poskytovat širší spektrum produktů a služeb, čímž se staly důležitými v sektoru bezkontaktního prodeje.

Automaty, jako první forma bezkontaktního samoobslužného prodeje, položily potřebné základy pro vývoj bezkontaktní samoobslužných prodejen. Ty představují inovativní trend v maloobchodu, který v posledních letech nabývá na popularitě. Vyznačují se absencí pokladních a tradičních front, což usnadňuje snadný a rychlý nákup bez interakce zákazníků s personálem. Tyto obchody mohou využívat technologie jako jsou automaty, nebo inteligentní kamery a senzory, které snímají pohyb zákazníků a zboží, automaticky přidávají vybrané položky do virtuálního košíku a umožňují platbu přes platební kiosky nebo mobilní aplikace (Podomarof, 2023).

Je potřeba podotknout, že jisté moderní varianty bezkontaktních samoobslužných prodejen jsou již i u nás v ČR několik let, konkrétně v podobě samoobslužných pokladen, které doplňují pokladny klasické u východu z obchodů jako je třeba Lidl, nebo Kaufland. Tato forma samoobslužného obchodu je ale pouze jednou z mnoha, například v Tesku je možnost použít přímo čtečky čárových kódů k nákupu zboží. V jiných případech, jako je třeba Amazon Go, uživatel vstoupí do obchodu s použitím mobilní aplikace a následně je využito strojové učení pro sledování zakoupeného zboží, které následovně zaplatí také pomocí aplikace v telefonu (Picaro, 2024).

Od roku 2019, kdy vlivem pandemie COVID-19 vzrostla potřeba vyřešit minimalizaci kontaktu a zvýšení hygieny v obchodech, se začal samoobslužný maloobchod prudce rozrůstat. Odhaduje se, že pandemie urychlila digitalizaci maloobchodu až o pět let (Alva, 2021). Očekává se, že do roku 2027 trh vzroste na hodnotu \$46 milliard dolarů s ročním růstem až 7 % (PYMNTS, 2021).

Mezi výhodami bezkontaktních samoobslužných prodejen jsou větší efektivita a snížení nákupní doby pro zákazníky. Pro prodejce to většinou znamená větší investici do spuštění prodejny, kterou ale vyvažuje absence zaměstnanců u pokladen – není totiž možnost, jak eliminovat potřebu zaměstnanců pro sledování prodejny a pro doplnění zboží. Obchody také pomocí některých řešení mohou efektivně sledovat stav zásob a analyzovat nákupní chování zákazníků, což umožňuje konstantně vylepšovat zákaznický zážitek a lépe přizpůsobit nabídku.

Mezi problémy lze začlenit obavy o to, zda ztráta osobního kontaktu s personálem nepřináší negativní dopad na zákazníky určitých skupin. Je pravda, že mnozí oceňují rychlosť a efektivitu samoobslužných řešení, jiní mohou postrádat interakci se zaměstnanci při placení. Tato proměnlivost v preferencích zákazníků může pro obchodníka znamenat potřebu dostatečně vyvážit technologickou inovaci a udržení osobní interakce (Podomarof, 2023).

Také je potřeba zvážit dopad využití automatizace na samotný personál prodejen. Pokud se stane, že zaměstnanci budou tuto změnu vnímat jako možné ohrožení jejich pracovní pozice, může být silně ovlivněna jejich pracovní morálka. Druhá strana mince však ukazuje, že toto řešení je velice efektivní v případě nedostatku personálu (Pan Oston, 2023).

Přechod k samoobslužným řešením také přináší výzvy v oblasti bezpečnosti. Absence pokladen s obsluhou jednoznačně zvyšuje riziko krádeží. Využití kamер a senzorů, jako jsou kontrolní brány u východů z obchodních zón, může minimalizovat toto riziko. Nejfektivnější je ale využití personálu, který dohlíží na samoobslužné kiosky a případně náhodně kontroluje nákupy zákazníků. Je však potřeba daný personál naučit rozpoznávat podezřelé chování.

## **2 Definice zadavatele**

Zadavatelem bakalářské práce je rodinná firma Do psí misky, která byla založena v roce 2017. Firma se specializuje na výrobu a následnou distribuci krmiva pro masožravá domácí zvířata, převážně psy a kočky, jejich hlavními produkty je BARF výživa, což je koncept „Biologically Appropriate Raw Foods“, tedy krmivo složené z čerstvých a syrových surovin, což je biologicky vhodná strava pro tato zvířata (Tým Do psí misky).

V nabídce společnosti lze nalézt široký sortiment produktů, ať se jedná již o zmíněný BARF, různé doplňky stravy, sušené maso a přílohy. Krom toho firma poskytuje i poradenské služby týkající se správné výživy pro domácí mazlíčky.

Zadání Tomáše Milce a Kateřiny Milcové, tedy jednatelů a majitelů společnosti Do psí misky bylo vytvoření návrhu a následná implementace systému samoobslužného obchodu v malém měřítku, který by se specifikoval čistě na jejich produkty. Systém bude použit pro případné rozšíření do míst, kde se jejich prodejny nenacházejí.

Zadavatel chce, aby si zákazník vytvořil uživatelský účet, obsahující jeho jméno, příjmení, telefonní číslo a další údaje, který bude využívat v telefonní aplikaci, jenž mu umožní vstup do prodejny, buďto skrze naskenování QR kódu, či skrze zadání vygenerovaného číselného kódu do zámku na dveří. Následně zákazník zahájí nákup, při kterém bude skenovat zboží na prodejně pomocí čárových kódů, opět pomocí mobilního telefonu, přičemž může získávat více informací o zboží přímo v aplikaci, nebo může zboží přidat do virtuálního a reálného košíku a následně zaplatit skrze platební bránu a opět odejít z prodejny.

Při hovorech se zástupci firmy byly specifikované detaily celého projektu. Hned na prvním sezení bylo rozhodnuto, že z důvodu časového a peněžního omezení nebude možné vytvořit a implementovat složitý systém otevírání a zavírání dveří pomocí QR kódu, místo toho se využije změna kódu pro vstup do prodejny při každém doplnění zboží – zadaný kód bude následně nahrán do databáze, ze které se bude uživatelům ukazovat skrze webovou aplikaci.

Je nutné zmínit, že společnost již nyní disponuje webovými stránkami (viz Obrázek 1), které jsou uživatelsky přívětivé a přehledné, jak v desktopovém, tak i v mobilním zobrazení. Implementace webové aplikaci do nich však nebude jednoduchá a v budoucnu je vysoká pravděpodobnost migrace specifických funkcí současných webových stránek do samotné mobilní aplikace.

The screenshot shows the homepage of the website [Do psí misky](https://www.dopsimisky.cz). At the top, there's a navigation bar with links for DOMŮ, AKCE, MASO, Vnitřnosti, KOSTA A CHIRURKÝ, BARK HOTOVÝ, OSTATNÍ, NAŠE PROJEKTY, TRASY ROZVOZU, KONTAKT, PARTNEŘSKÁ SPOLUPRACE, KONTAKTY, BLOG, FACEBOOK, INSTAGRAM, and CZK. There are also social media icons for Facebook and Instagram, and a language switcher.

The main banner features a dog lying next to several bags of dog food. It includes a call-to-action button "Koupit". To the right of the banner are two boxes: "OSOBNÍ ODBĚR" (Personal Collection) and "ROZVOZ" (Delivery). Below the banner are four service icons: "Dovozeme k Vám domů od 119 Kč", "Vysíláme na šíří odběrných míst", "Poštovné", and "Rozvezeme po ČR i SR".

The "Novinky" (New products) section displays eight items:

Produkt	Cena
Dárkový poukaz mix jeky Šedčíka	99 Kč
SUŠENÉ králičí patzny 3 ks	59 Kč
Pangasius filet	109 Kč
Moháčská štítko 0,5 kg	69 Kč
SUŠENÉ kuřecí chipsy	69 Kč
Zdobené Babiččky - Klasické s tvarohem, rýží a zeleninou	109 Kč
Dárkový poukaz 1000,-	1 000 Kč
Kapri	79

Obrázek 1 Printscreen webu Do psí misky s.r.o. ke dni 11.2.2024 (Tým Do psí misky, 2024)

## **2.1 Funkční požadavky**

Pro vývoj aplikace je potřeba si jasně definovat funkční požadavky. Je potřeba zaměřit se na požadavky jak ze strany zákazníků, tak ze strany samotné administrace, která bude sledovat stav prodejen. Při analýze zadání byly rozpoznány tyto funkční požadavky.

### **Registrace a přihlášení uživatelů, správa účtů**

Současný web již obsahuje možnost registrovat a přihlásit se pro uživatele a popřípadě měnit jejich informace, pro aplikaci však bude použit nový systém registrace a přihlášení, který nebude nijak navazovat se systémem současným, a to z důvodu bezpečnosti v prodejně a zamezení krádeží je potřeba získat více informací, jako je fotka uživatele, datum narození a bydliště, popřípadě rodné číslo.

V prodejně budou instalované kamery, podle kterých se v případě zjištění krádeže pomocí uložených přístupů do prodejny může zjistit totožnost uživatele a určit další kroky pro postup ve vyřešení daného problému.

### **Autentizace přístupu do prodejny**

Pro přístup do prodejny bude zapotřebí zadat předem vygenerovaný číselný kód od zámku, který se bude měnit při každém zásobování prodejny. Tím se minimalizuje riziko přístupu neautentizovaných uživatelů. Kód bude zobrazen v aplikaci po naskenování QR kódu u vstupu do prodejny. Tímto způsobem můžeme definovat přístupy do více prodejen, které budou mít rozdílný přístupový kód.

### **Skenování produktu a získání informací o produktu**

Skenování produktů proběhne skrze mobilní aplikaci v reálném čase. Po naskenování daného čárového kódu bude uživateli zobrazena obrazovka s informacemi o daném produktu. Pokud mu bude zboží vyhovovat, může jej přidat jak do svého reálného, tak do virtuálního košíku a následně jej zakoupit.

### **Možnost zaplacení skrze platební bránu**

Pro ukončení produktu uživatel naskenuje QR kód, nebo v okně aplikace klikne na tlačítko ukončení nákupu. Poté bude košík sečten, překontrolován uživatelem a následně bude částka odeslána jako požadavek na platební bránu, prostřednictvím které zákazník nákup zaplatí. Uživateli bude vytvořena elektronická účtenka, kde bude vidět celý jeho nákup.

### **Odchod z prodejny**

Se zadavateli bylo diskutováno o vytvoření kontrolního systému, při kterém by po ukončení nákupu, či jeho přerušení v průběhu uživatelem, byl odemčen východ prodejny. Po přezkoumání bylo zjištěno, že toto provedení by vyžadovalo komplexní řešení, které by splňovalo požární směrnice a bylo by potřeba vše propojit s detektory požáru v objektu, což je velmi složité pro první iteraci systému. Bylo tedy určeno, že z počátku bude odchod uživatelů z prodejny nekontrolovaný. Aplikace však uživatele upozorní na neaktivitu, pokud

z prodejny odešel a je nečinný v aplikaci, popřípadě ukončí nákup za něj, což bude poznamenáno v zápisu o historii nákupů.

### **Správa zásob prodejny a cen zboží**

Zadavatel vyžaduje přesné informace o veškerém předpokládaném stavu zboží na veškerých prodejnách, jak odděleně, tak i celkově. Pro autora to znamená, že bude potřeba vytvořit přehledný online systém pro správu všech produktů, který bude podporovat úpravy všech informací o zboží.

### **Reporty o stavu zboží na skladech**

Zadavatel si vyžádal, zdali by bylo možné u každého zboží nastavit varování, které by bylo přenesené do reportu, který by se zasílal na konci dne na mail zadavatele z důvodu upozornění na potřebné doskladnění zboží na prodejně.

### **Uživatelská administrace**

Zadavatel bude potřebovat možnost upravovat všechny uživatelské profily v případě jakéhokoliv problému z důvodu podpory zákazníků.

### **Přehled o vstupu do prodejen**

Každému uživateli bude přiřazeno unikátní ID, které bude zaznamenáno při načtení QR kódu do aplikace pro vstup do prodejny. Také bude sledováno, jestli započal, nebo dokončil nákup. Toto řešení je vhodné v případě krádeží v obchodě.

## **2.2 Nefunkční požadavky**

Po vytvoření a analýze funkčních požadavků je potřeba sestavit nefunkční požadavky, které jsou s nimi úzce spjaté a vychází z nich.

### **Provedení aplikace**

Řešení bude tvořeno webovou responzivní aplikací, která bude primárně určena pro moderní chytré mobilní telefony z důvodu schopnosti zpracovat požadované údaje jako jsou QR a čárové kódy. Preferovaný jazyk, a také jediný jazyk podporovaný aplikací v první verzi je čeština. Autor aplikace také spolupracuje se zadavatelem i na reálném řešení, jako jsou QR kódy u prodejen a čárové kódy zboží.

### **Bezpečnost a ochrana dat**

Z důvodu bezpečnosti a ochrany dat je zapotřebí minimalizovat riziko úniku těchto dat. To lze vyřešit omezením v kódu samotném, který bude limitovat, jak se k datům dá dostat a jak je lze do databáze vložit. Je potřeba pečlivě vyzkoušet všechny součásti aplikace a zabezpečit server, na kterém bude webová aplikace hostována proti případným útokům.

### **Spolehlivost systému**

Je potřeba zajistit nejvyšší spolehlivost systému, která bude možná a minimalizovat downtime aplikace. Jedná se o řešení, které je možné využívat 24/7 a od toho se i odvíjí potřebná spolehlivost. V případě rozšíření prodejen je potřeba zajistit servery, které budou dostupné v případě selhání jiného serveru.

### **Údržba a rozšiřitelnost**

Aplikace bude celá zdokumentována v kódu pro jednodušší rozšiřitelnost a údržbu s tím, že zadavatel bude v případě zájmu kompletně seznámen s jejím obsahem.

### **Kompatibilita**

Je potřeba zajistit co nejsírší kompatibilitu. Z důvodů rychle vyvíjejících se technologií na straně mobilní telefonů je potřeba zajistit kompatibilitu s telefony které jsou staré minimálně 4 roky. Z důvodu potřeb skenování QR a čárových kódů na straně uživatele a následném zpracování je potřeba výpočetní výkon, který by tyto telefony měli splňovat.

### **Uživatelská přívětivost**

Uživatelské rozhraní aplikace by mělo být jednoduché, a ne příliš komplikované, aby uživatelé s telefony menších rozměrů neměli potíže s použitím aplikace.

Po rozpoznání všech požadavků je potřeba zjistit, jaká dostupná řešení se vyskytují na trhu. Následuje krátké zhodnocení řešení, která se nejvíce podobají tomu, co zadavatel vyžaduje.

## **2.3 Dostupná řešení**

Na trhu jsou dostupná mnohá řešení pro vytvoření samoobslužných prodejen, přičemž tato kapitola se zabývá jejich základním popisem a přiblížením klíčových prvků, které by se mohli využít při vytváření autorova řešení.

### **Pixevia**

První analyzované řešení je vytvořeno firmou Pixevia, která se zaměřuje na automatizovaná prodejní místa. Prodejny fungují na principu kamer, senzorů a chytrých regálů, které snímají, co vše si zákazník koupí.

Vstup do nákupní zóny je zajištěn pomocí vstupních bran, kde se uživatel prokáže buďto kreditní kartou, pomocí virtuální penězenky v telefonu či aplikací k tomu určené. Po přiložení ke čtečce je brána otevřena a uživateli je dovolen vstup do nákupních prostorů. Pomocí kamery je uživatel sledován a jeho trasa je zaznamenávána po celou dobu nákupu (Viessmann) (PIXEVIA).

Umělá inteligence a senzory zajišťují rozpoznání všech akcí, které zákazník provádí, ať už se jedná o otevření chladících boxů, nebo vrácení zboží zpět do regálů. Také se vytváří virtuální nákupní košík, který je definovaný zvlášť pro každého zákazníka. Pro ukončení nákupu zákazník zkонтroluje obsah virtuálního košíku a následně jej zaplatí zvolenou metodou u východu z prodejny (PIXEVIA).

Toto řešení zajišťuje bezproblémový nákup pro zákazníky, a navíc poskytuje majitelům prodejen 3D model celého prostoru, který mapuje kde se zákazníci nachází nejčastěji, jaké používají cesty apod. Řešení navíc poskytuje možnost sledování stavu zboží, upozornění na nízký stav zboží a také předpovídá možnou budoucí poptávku (PIXEVIA).

Pixevia běžně poskytuje řešení pro čerpací stanice, ale i pro večerky, obchody se smíšeným zbožím až po samoobsluhy do velikosti 400 m<sup>2</sup>. V případě jiných požadavků je ale možné s firmou spolupracovat na vlastním řešení. Řešení podporuje implementaci chladících boxů, zboží v bedýnkách (ovoce, zelenina) či produkty 18+. Dle dokumentů firmy má AI přesnost 98.1 % při rozpoznávání zboží (Viessmann) (PIXEVIA).

Na českém trhu je řešení dostupné pomocí spolupráce firmy Viessmann Refrigeration Solutions se společností Pixevia, které lze doplnit o řešení firmy Viessmann pro automatizaci nano skladování pomocí chytrých věží, které automaticky rozpoznávají, o jaké zboží se jedná a v případě zjištěné nedostupnosti na prodejně zboží vyskladní pro doplnění (Viessmann) (Viessmann).



Obrázek 2 Vstup do prodejny založené na technologii Pixevia, zdroj: Pixevia

## **UNIBox**

Řešení, které bylo vytvořeno firmou Loxone Partner – Syreta uni retail technology gmbh a skupinou Unimarkt Gruppe, nese název UNIBox. Jedná se o prodejní místo velikosti drobné večerky, které začalo být nasazeno v Rakousku. Tento typ prodejny nepotřebuje zaměstnance, kromě doplňování zboží, a je plně automatizované (Loxone, 2021).

UNIBox podporuje dva typy nákupu – buďto s platbou skrze mobilní aplikaci, nebo pro zaplacení skrze pokladnu.

V případě, kdy zákazník vlastní chytrý telefon, ve kterém je nainstalována potřebná aplikace, stačí zahájit nákup naskenováním QR kódu u vstupu do budovy. Následně uživatel skenuje čárové kódy na zboží, a v případě zájmu si zboží umístí jak do reálného košíku, tak do toho virtuálního. Pro ukončení nákupu stačí v aplikaci vše zaplatit a uživatel může odejít. Pokud však uživatel nechce platit skrze online platební bránu v mobilním telefonu, může zboží naskenovat na samoobslužné pokladně, která se nachází v prodejně a na ní vše zaplatit (Loxone, 2021) (Unimarkt).

V případě zájmu je toto řešení možné provozovat i v ČR, je ale zapotřebí zakoupit od dodavatele potřebný hardware. Srdce celé prodejny je Loxone Server, který se stará o konektivitu mezi všemi zařízeními v objektu.

Server sám o sobě funguje s CPU frekvencí 400MHz s 64 MB RAM a nominální spotřebou 1.85 W a dle odhadu autora je zařízení založeno na ARM architektuře. Dle podpory konektivity se jedná o počítač podobný například Raspberry Pi s rozvinutou specifickou konektivitou, kterou je Loxone Link a Loxone Tree, které lze využít pro propojení dalších zařízení prodejny, jako jsou chladící boxy, osvětlení, nebo i správce energií, dotykové obrazovky či senzory přítomnosti a pohybu osob a samozřejmě kamery. Pro bezpečnost zákazníků server může být rozšířen i o detektor kouře v prodejně. V případě požáru je automaticky upozorněna obsluha UNIBoxu (Loxone, 2021) (Loxone).

Pro majitele prodejny je dostupná aplikace, která komunikuje s centrálním Miniserverem, který komunikuje s ostatními Miniserversy v prodejnách. Pro majitele prodejny to znamená to, že může v správcovské aplikaci sledovat aktuální stav prodejny – od stavu posuvních dveří až po spotřebu energií či teploty v prodejně (Loxone, 2021).

Vzhledem k podobě řešení UNIBox s cílem bakalářské práce, je potřeba také zvážit cenové řešení UNIBox. Jádro celé prodejny teoreticky vyjde majitele nové prodejny (miniserver, potřebné komponenty pro napojení senzorů) minimálně na 40 000 Kč bez DPH (Loxone, 2023). Tato cena nezahrnuje senzory teploty, energií, pohybu, či chladící boxy, nebo systém pro otevírání dveří. Částka za softwarové řešení nebyla autorem nalezena.



Obrázek 3 Prodejna UNIBox, zdroj: Loxone

## **B.O.S.S. Automatic**

B.O.S.S Automatic je nástavba pokladního a obchodně-skladového systém B.O.S.S. Enterprise firmy PVA Systems. Jedná se převážně o řešení pro rozšíření již zavedených obchodů s obsluhou o samoobslužné pokladny, ale může být použit i pro vytvoření autonomních prodejen, v ČR je například používán obchodním řetězcem COOP. (PVA Systems, 2023)

Samotný B.O.S.S Enterprise slouží jako „*Univerzální řešení problematiky prodeje zboží v centralizované maloobchodní síti a distribučních velkoobchodních skladech, až po poziční paletové skladování*“ (PVA Systems). Ve výsledku se jedná o řešení, které dovoluje prodejci monitorovat, řídit a zajišťovat komunikovat mezi provozy, a to až po pokladní místa. Mezi jeho klíčové vlastnosti spadá centrální databáze, replikace dat a konfigurovatelnost systému, díky čemu je vhodný pro různé oblasti obchodu.

Prodejny fungují buďto v denním, či nočním režimu. Přes den zákazníci platí u pokladen s obsluhou, nebo u samoobslužných pokladen, po konci otevírací doby je prodejna přepnuta do nočního režimu a zaměstnanci ji opouští.

V nočním režimu je pro vstup do prodejny potřeba bankovní identita spárována s aplikací v telefonu, která vygeneruje vstupní QR kód, vstup je ale také možno vyřešit třeba skrze zákaznickou či platební kartu. Zákazník je poté volně vpuštěn do nákupní zóny, ve které je mu umožněno volně nakupovat. Platbu za zboží provede na samoobslužné pokladně po ověření totožnosti naskenováním vygenerovaného QR kódu z aplikace. (PVA Systems, 2023)

Prodejnu je možné doplnit různými moduly, například pro automatizované otevírání dveří, elektronickými cenovkami pro automatické přepisování cen zboží, kamerovým systémem, senzory pro sledování stavu prodejny či systémem pro automatické objednávání zboží. Navíc lze připojit i systém datových skladů a BI pro analýzu obchodních dat či vytváření dashboardů. (PVA Systems, 2023) (PVA Systems)

Řešení je možné použít v prodejnách, které nebyly zamýšleny pro samoobslužný běh, pokud již celý systém prodejen funguje na B.O.S.S. Enterprise. Je také možné si od firmy zajistit pouze softwarovou část, pokud si zákazník přeje využít hardware od jiného dodavatele. (PVA Systems, 2023)

Autora na celém B.O.S.S. řešení nejvíce zaujala možnost propojit již existující prodejnu se samoobslužnou částí, ověření uživatele skrze bankovní identitu a možnost automatické cenotvorby a jejího promítnutí v reálném čase pomocí elektronických cenovek. Z důvodu časového rozložení projektu se autor rozhodl tyto vlastnosti do projektu integrovat až v po prvotním spuštění celého systému.



Obrázek 4 Prodejna značky COOP Jednota se samoobslužným nočním nákupem poháněná systémem B.O.S.S. Automatic, zdroj: PVA Systems

## **Easyout**

Společnost Trigo navrhla systém Easyout v roce 2022. Jedná se o řešení, které se zaměřuje na transformaci existujících obchodů s obsluhou do samoobslužných obchodů s podporou použití aplikace, platebního terminálu nebo samoobslužného kiosku. Systém je například používán v německých obchodech Netto nebo Aldi, či v obchodech Tesco (Trigo, 2023).

Systém se opírá o strojové učení a umělou inteligenci. V prostoru jsou strategicky rozmístěny kamery, které generují a aktualizují virtuální 3D model prostoru a sledují pohyb osob od vstupu, přes nákup, vrácení zboží až po platbu a odchod. V policích jsou umístěny senzory, které sledují vyjmutí, či navrácení zboží, ty následně upozorní kamery na změnu a ty zjistí, jaké zboží bylo navráceno v reálném čase (Trigo, 2022).

Zpracovaná data jsou následně anonymizována a zpracována do podoby, která vyhovuje majitelům obchodu. Je možné sledovat rozhodování zákazníků při nákupu různých typů zboží, sledování tras, které zákazník používá, nebo také může předpovídат spotřebu zboží, přičemž upozorní na případné potřeby zásobování (Trigo, 2022).

Zákazník je do nákupní zóny vpuštěn po naskenování QR kódu z aplikace. Následně je vytvořen jeho virtuální 3D model, který simuluje jeho pohyby v reálném prostoru. Společně se senzory se následně generuje a plní jeho virtuální nákupní košík, který má dle firmy Trigo spolehlivost 99 %. Po dokončení nákupu je možné zaplatit skrze zvolenou metodu dané prodejny z již zmíněných možností a odejít. Pokud uživatel využil aplikaci, je účtenka vložena do ní, pokud ne, je vytisknuta na místě nákupu (Trigo, 2022).



Obrázek 5 Autonomní samoobslužná prodejna Tesco v centru Londýna, zdroj: Trigo

### **2.3.1 Srovnání dostupných řešení**

V současné době existuje na trhu mnoho řešení, které umožňují prodejcům vytvořit vlastní samoobslužná místa. Většina z řešení je dělaná na míru pro dané společnosti a může být problém s licencováním a zavedením řešení do menšího provozu, jako je rodinná společnost Do psí misky.

Srovnání dostupných řešení ukazuje, že požadavkům zadavatele se přibližují řešení B.O.S.S Automatic a UNIBox.

Zavedení B.O.S.S Automatic může vyžadovat poměrně vysoké začáteční náklady, včetně kompletního přechodu na systém B.O.S.S Enterprise. Navíc systém nedisponuje čistě možností vytvoření pouze samoobslužného místa a je primárně určeno pro rozšíření existujících obchodů o samoobslužné funkce, kde primárně není využíván mobilní telefon, ale samoobslužná pokladna umístěna v prodejném místě.

Řešení UNIBox, které velikostí prodejních míst a schopnostmi je velice podobné hledanému řešení zadavatele, je potřeba vybavit hardwarem od firmy Loxone, které také vyžaduje poměrně vysoké náklady na zprovoznění.

Po předložení dostupných návrhů zadavateli bylo rozhodnuto, že bude přikročeno k vývoji vlastní aplikace, která bude pohánět samoobslužné bezkontaktní prodejny, které bude na míru dělané pro firmu Do psí misky.

## **2.4 Využitelné prvky ostatních řešení**

V rámci tohoto bakalářského projektu se autor rozhodl nejvíce inspirovat řešením UNIBox, které je samo o sobě nejvíce podobné hledanému řešení zadavatele. Hlavním zaměřením je tedy identifikace a adopce klíčových prvků tohoto modelu s ohledem na omezené časové a finanční prostředky. Hlavním rozdílem je úmyslná absence některých složitějších technologií a funkcí, které jsou součástí UNIBoxu a podobných řešení.

Z ostatních řešení je přejato použití webové aplikace pro registraci uživatelů, jejich vstup do prodejny a proces nakupování. Tento přístup umožňuje budoucí škálovatelnost a možnost implementace prvků, které budou nepoužité, jako jsou pokladní kiosky, složité senzorové systémy nebo využití strojového učení a AI.

Bezpečnost v samotné prodejně bude zajišťována kamerami, které ale nebudou propojeny se systémy tohoto bakalářského projektu. Není tedy potřeba autorem vyřešit zabezpečení a případné zamezení zneužití kamerových systémů.

Klíčovou charakteristikou je důvěra v zákazníky, kteří budou využívat prodejní místa. Je předpokládáno, že skrze využití velkého množství informací o nich (včetně neintegrovaných kamerových záznamů), bude značné množství zákazníků odrazeno od případných krádeží v prodejně. V tomto se autorovo řešení nejvíce rozchází s jinými, které využívají složitější bezpečnostní a monitorovací technologie.

Dále je přejato využití čárových a QR kódů, které budou po prodejně sloužit k mnoha účelům, jako je zobrazení vstupního číselného kódu, nákupu, zobrazení informací o zboží, získávání informací z QR kódů umístěných v prodejně, a ukončení nákupu.

Je třeba zdůraznit, že omezení jsou charakteristická pouze pro prvotní verzi webové aplikace. V případě zájmu ze strany zadavatele bude aplikace dále doplňována o další funkcionalitu a technologie. Cílem je vytvořit flexibilní a škálovatelné řešení, které se bude jednoduše adaptovat na budoucí technologický vývoj a potřeby zákazníků.

# 3 Návrh aplikace

V této kapitole je definován workflow aplikace, vzhled a hrubý návrh celého systému, který byl konzultován se zadavatelem bakalářské práce. Také jsou ukázány mapy procesů, které vytváří funkčnost aplikace. Use case diagram a mapy procesů byly vytvořeny pomocí aplikace Draw.io.

## 3.1 Use case diagram

V rámci návrhu systému byl vypracován use case diagram popisující rozsah interakcí mezi uživateli a systémem. Jsou definovány tři typy uživatelů, a to nový uživatel a registrovaný uživatel a registrovaný uživatel s aktivním účtem. Je také definována administrátorská role, která je rozdělena do rolí A: Aplikace, A: Uživatelů, A: Prodejen a A: Zboží. Pro vytvoření

Nezaregistrovaný (nový) uživatel má omezený přístup v aplikaci, na rozdíl od již registrovaného a přihlášeného uživatele s aktivním účtem. Má přístup k registraci, ale také si může zobrazit katalog dostupného zboží na všech prodejnách, zboží v současné slevě, či informace o ochranně osobních údajů, obchodní podmínky, kontaktní informace obchodníka a návod popisující fungování mobilní části aplikace pro nákup zboží.

Registrovaný zákazník se může přihlásit pomocí svých přihlašovacích údajů, upravit své osobní údaje a aktivovat svůj účet pomocí zaslávaného odkazu v jeho kontaktním mailu. Bez aktivace není uživateli povoleno vstupovat do prodejen, či skenovat zboží.

Registrovaný uživatel s aktivním účtem má přístup k hlavním zákaznickým částem aplikace, může tedy vstoupit do prodejny pomocí naskenování QR kódu a následným zadáním aktuálního bezpečnostního kódu od elektronického zámku, provést nákup zboží v prodejně, zaplatit za zboží a také si může zobrazit historii svých nákupů.

Jedním z problémů, který autor nalezl, je původní návrh zadavatele řešení funkce administrátora. Originálně byla zamýšlena pouze jedna role Administrátor, která byla univerzální pro všechny zaměstnance společnosti. Po konzultaci a popsání problému se zadavatelem bylo rozhodnuto rozdělení této jedné funkce do několika.

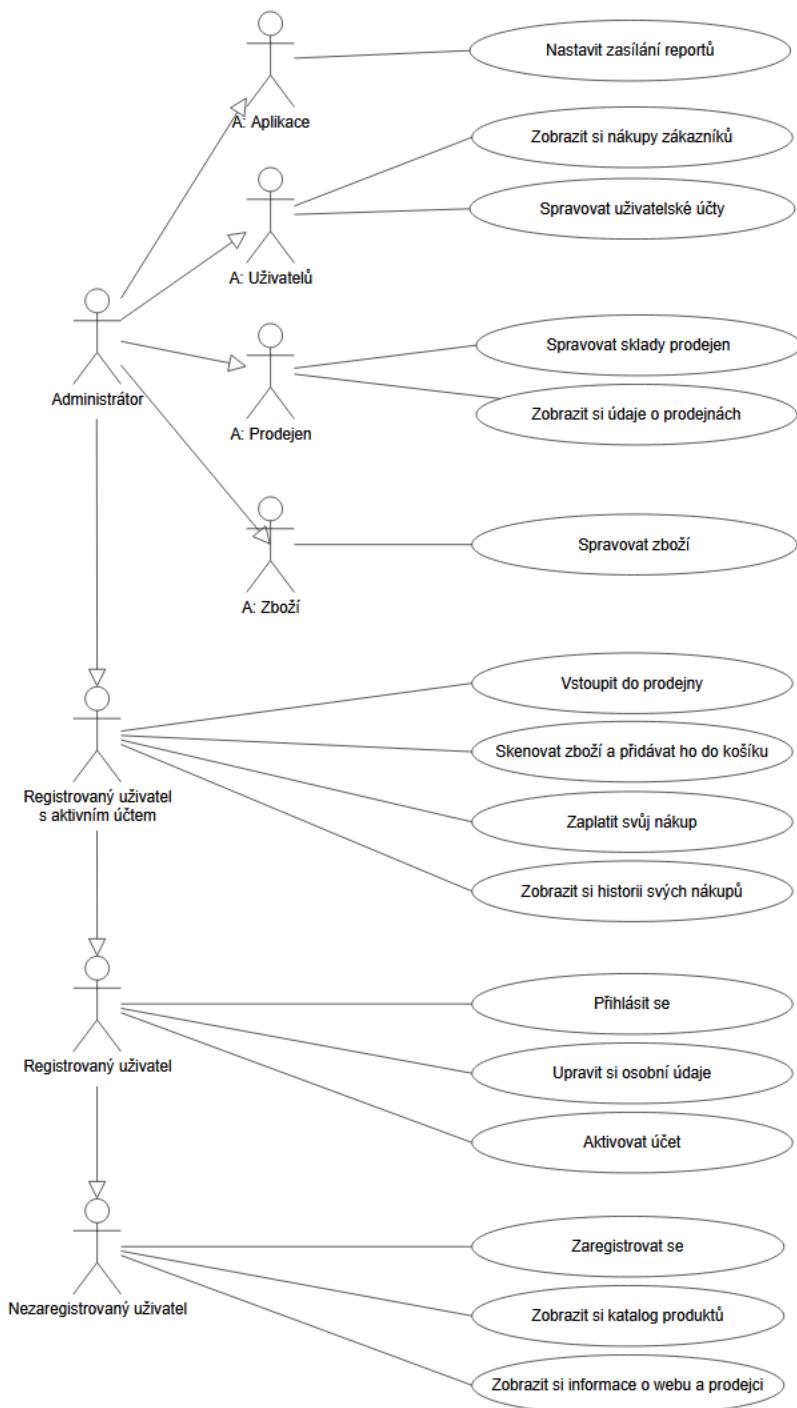
Role A: Aplikace v tuto chvíli bude sloužit čistě pro nastavení zasílání a tvoření reportů o pohybu zboží na skladech, v budoucích verzích je plánováno tyto funkce rozšířit o vytváření grafů prodeje různých typů zboží, sledování výdělku, či sledování úspěšnosti dokončení plateb.

Role A: Uživatelů poskytuje uživateli možnost spravovat účty uživatelů, přidávání poznámek k danému uživateli, či pozastavení uživatelského účtu. Je také možno zobrazit si detailní výpis nákupů každého z uživatelů.

A: Prodejen má možnost zobrazit si, vytvořit, či deaktivovat prodejnu, například při haváriích, či jiných zábranách provozu. Uživatel také může sledovat stavy skladů prodejen,

či přidat/odebrat zboží při jeho doplnění, nebo při inventuře. Také má možnost generovat nové přístupové QR kódy, či měnit přístupové kódy spojené s existujícími QR kódy.

Role A: Zboží může vytvářet nové zboží a vkládat ho do databáze včetně popisků, obrázků a slev. Má také možnost nastavit slevu na dané časové rozmezí, které bude automaticky aktivováno aplikací a případně automaticky deaktivováno. Zboží také může deaktivovat, což zamezuje jeho využití.



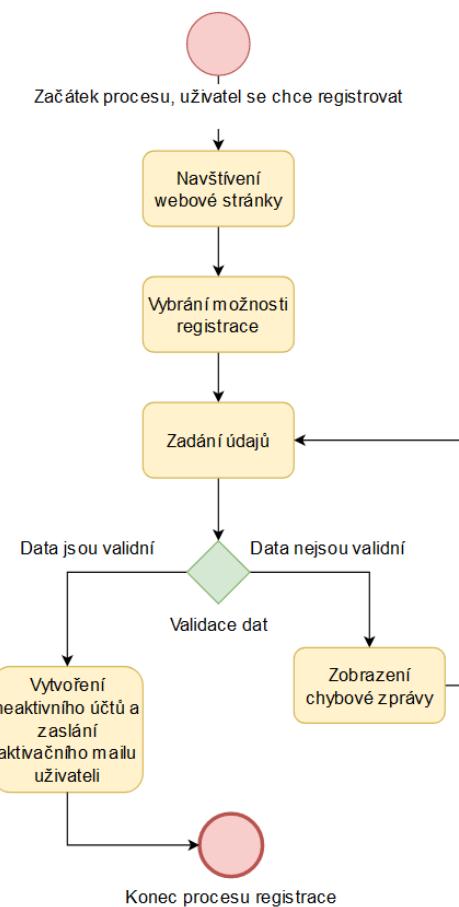
Obrázek 6 Use case diagram

## 3.2 Mapy procesů

V následující podkapitole jsou popsané složitější procesy, které potřebují důkladnější analýzu a vizualizaci. Jednodušší procesy, jako editace existujících produktů, či zobrazení provedených nákupů, v této kapitole nejsou znázorněny z důvodu jejich jednoduchosti.

### Registrace uživatele

Nový uživatel navštíví stránky aplikace. Následně vybere možnost registrace, čímž začíná celý proces registrace (viz Obrázek 7). Uživatel zadává své osobní údaje do formuláře. Po jejich zadání systém provede jejich validaci a pokud jsou všechny informace správné a například mail již není používán jiným účtem, systém vytvoří nový uživatelský účet, který bude označen jako dosud neaktivovaný (nebude možné nakupovat pomocí tohoto účtu). Uživatel musí potvrdit svojí registraci kliknutím na odkaz v mailu, následně je účet aktivován a registrace dokončena. Pokud validace dat neproběhne v pořádku (uživatel již je zaregistrován, některé údaje jsou zadané špatně), je uživateli zobrazena chybová zpráva v registračním formuláři.



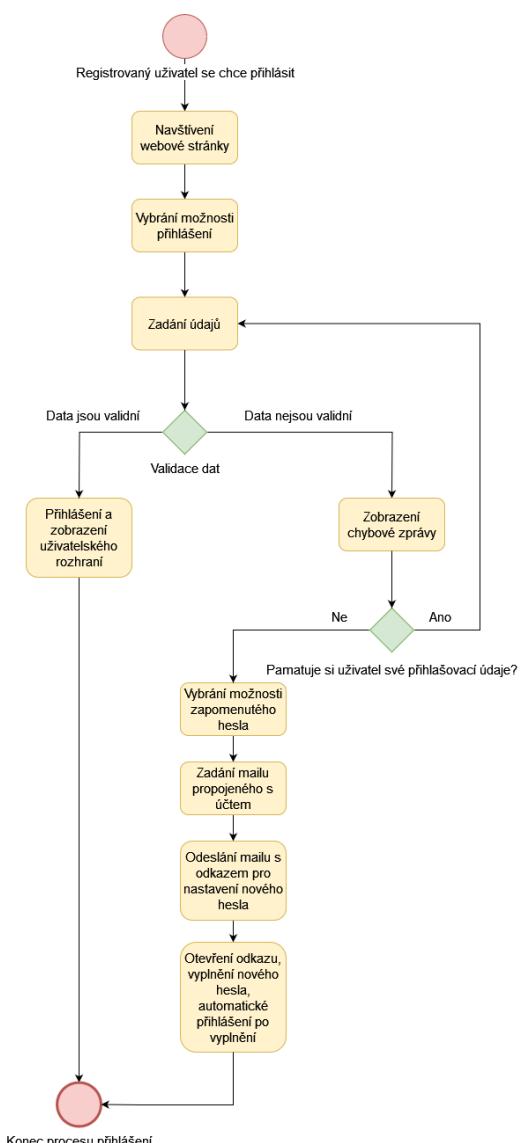
Obrázek 7: Diagram registrace uživatele

## Přihlášení uživatele

Již zaregistrovaný uživatel se může přihlásit do systému. Proces přihlášení (viz Obrázek 8) začíná otevřením webové aplikace, kde uživatel vybere možnost přihlášení. Ve formuláři vyplní své přihlašovací údaje (mail a heslo). Po zadání těchto údajů systém ověří jejich správnost vůči databázi.

Pokud jsou údaje validní a odpovídají existujícímu účtu, systém uživatele přihlásí a umožní mu přístup k jeho uživatelskému rozhraní a pokročilým funkcím aplikace. Pokud data nejsou správná, je uživateli zobrazena chybová zpráva v přihlašovacím formuláři.

Uživatel také může vybrat možnost zapomenutého hesla a zadat svůj mail propojený s aplikací. Pokud je v databázi mail nalezen, uživateli je odeslán mail s odkazem pro obnovení hesla. Po jeho rozkliknutí a zadání nového hesla je uživatel přesměrován na hlavní obrazovku aplikace a automaticky přihlášen.



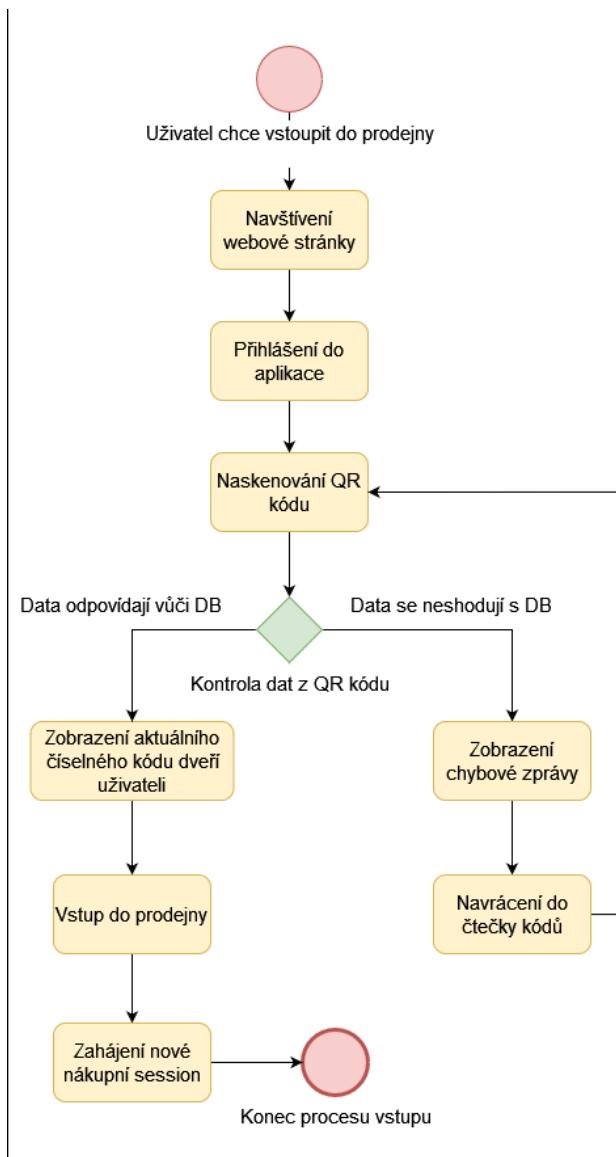
Obrázek 8 Diagram přihlášení uživatele

## Vstup do prodejny

Uživatel, jenž je registrovaný a má aktivní účet a který chce vstoupit do prodejny, musí nejdříve naskenovat QR kód umístěný u vstupu. Tento kód je unikátní pro každou z prodejen a je tvořen z náhodné kombinace znaků, které jsou následně převedeny do formy QR kódu, který slouží k identifikaci a autorizaci vstupu, přičemž systém zaznamenává vstup daného uživatele v databázi do příslušné tabulky.

Po naskenování QR kódu je v databázi ověřen jeho obsah a pokud je obsah stejný vůči záznamu v databázi, je uživateli zobrazen aktuální přístupový kód k digitálnímu zámku dveří nastaven administrátorem aplikace. Také je vytvořena nákupní session a vytvořen košík uživatele pro tento nákup. Uživateli je následně zobrazeno rozhraní určeno pro nákup.

V případě, že obsah QR kódu neodpovídá vůči záznamu v databázi, je uživateli vypsána chybová hláška o odmítnutí přístupu do prodejny.



Obrázek 9 Diagram vstupu do prodejny

## Nákup a zaplacení zboží

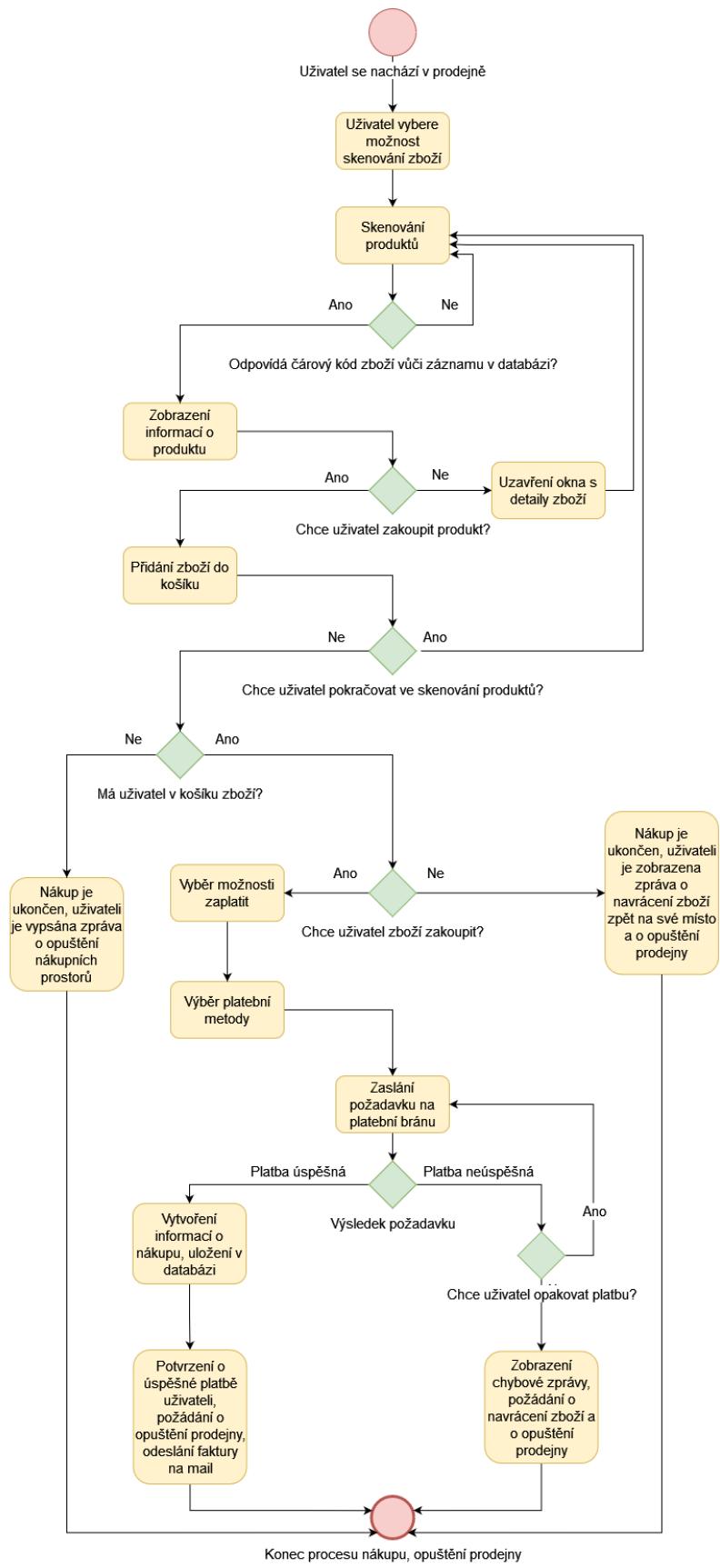
Po vstupu do prodejny je uživateli založena nákupní session a vytvořen nákupní košík pro tuto session. Uživatel následně může vybrat možnost skenování produktů, co otevře okénko s hledáčkem kamery, kde uživatel jednoduše načte čárový kód. Po jeho naskenování je na server poslan požadavek s kódem produktu. Pokud je kód produktu nalezen v databázi, je uživateli zobrazeno okno s údaji o daném produktu, včetně ceny a popisu. Uživatel následně může zboží vložit do svého košíku, či zboží vrátit na své místo a pokračovat v nákupu. Tento proces se opakuje, dokud zákazník nezvolí možnost ukončit nákup.

Pokud uživatel nemá ve svém košíku zboží, je jeho nákup ukončen a následně je požádán o to, aby opustil prostory prodejny.

Pokud má uživatel ve svém košíku zboží, je požádán o jeho kontrolu. Následně může nákup zrušit, či pokračovat k platbě. Pokud nákup zrušil, jeho session je ukončena, je požádán o navrácení zboží zpět na své místo a o následné opuštění prodejny.

Pokud uživatel vybral možnost zaplacení nákupu, je požádán o vybrání platební metody a následně je přesměrován na platební bránu, kde vyplní platební údaje a zašle je ke zpracování. Pokud platební proces:

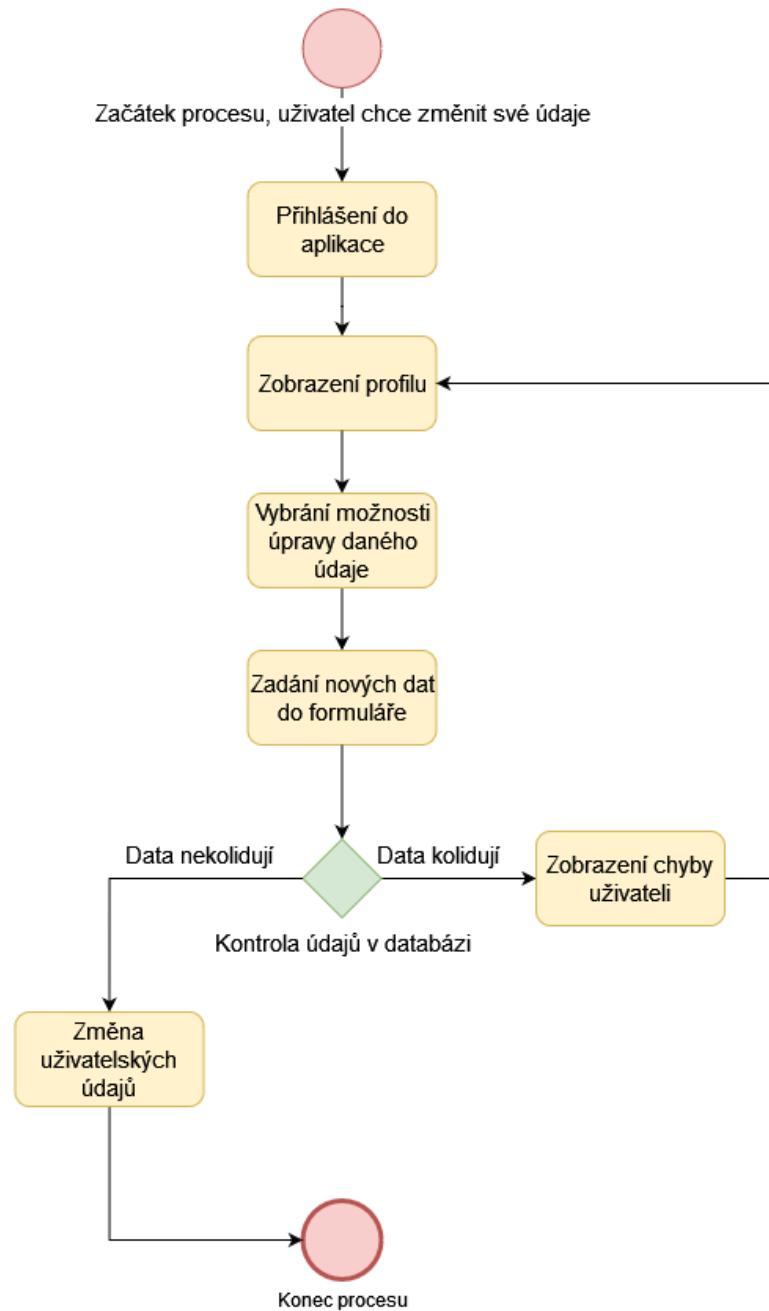
- **Vrací chybu:** Uživatel si může vybrat, zdali chce platbu zkoušit znova. Pokud ano, je proces opakován. Pokud ne, je přesměrován zpět na stránky aplikace, je ukončena jeho nákupní session, je zaznamenán neúspěšný záznam v databázi a je požádán o navrácení zboží a opuštění prodejny.
- **Vrací úspěšné zpracování:** Uživateli je ukončena nákupní session, je vygenerován úspěšný záznam v databázi, je vytvořen záznam o celém jeho nákupu a následně je odeslána faktura potvrzující nákup na jeho mailovou adresu. Následně je zobrazena zpráva pro uživatele o úspěšně dokončeném nákupu a požádání o opuštění prostorů s jeho zakoupeným zbožím.



Obrázek 10 Diagram procesu nákupu a platby

## Úprava osobních údajů

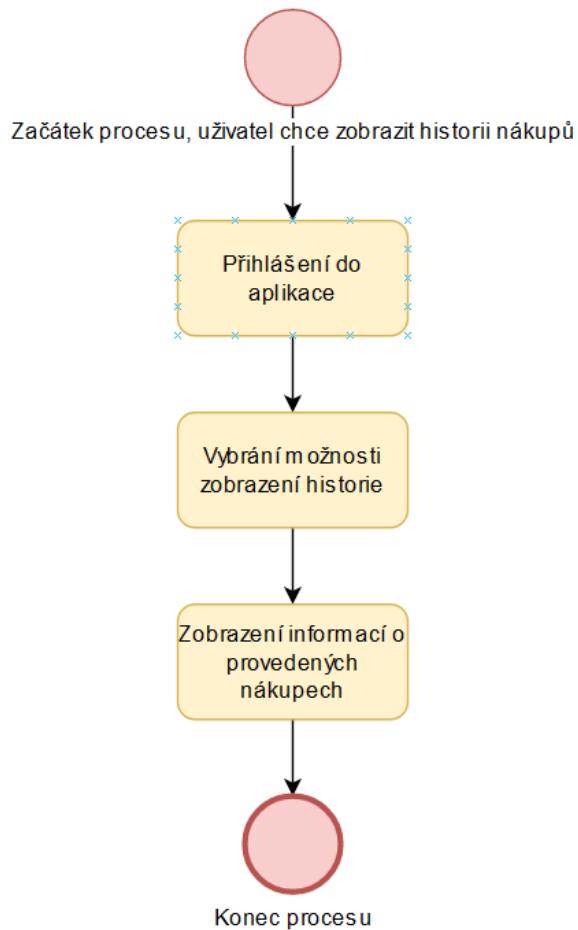
Uživatelé aplikace mají možnost upravit své osobní údaje v nastavení svého účtu. Po přihlášení do aplikace uživatel vybere možnost zobrazení profilu a následně si vybere položku kterou chce a může editovat. Je otevřeno okno, kde zadá nový údaj a následně změny potvrdí. Data jsou zkontovalována systémem, (pokud se jedná např. o telefonní číslo), zda již není účet, který tyto údaje využívá. Pokud data nekolidují, jsou informace ihned změněny a uživatel je o změně informován v okně uživatelského profilu. Pokud data kolidují, je uživateli zobrazena chybová hláška a data zůstávají nezměněna.



Obrázek 11 Diagram změny uživatelských údajů

## Zobrazení historie nákupů

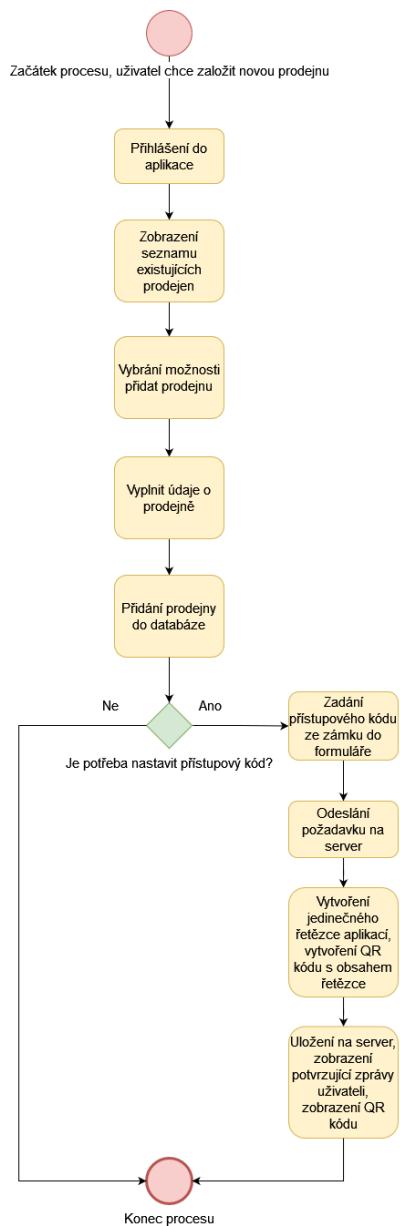
Uživatelé mají možnost zobrazit historii svých nákupů v aplikaci. Po přihlášení a vstupu do nastavení účtu mohou vybrat možnost zobrazení historie nákupů. Systém následně zobrazí seznam všech předchozích nákupů, včetně detailů jako jsou data nákupů, zakoupené položky a celkové částky.



Obrázek 12 Diagram zobrazení historie nákupů

## Vytvoření prodejny, přidání vstupního kódu a generace QR kódu

Administrátor, jenž má roli A: Prodejen má možnost přidat novou prodejnu v seznamu již existujících prodejen. Uživatel vyplní potřebné údaje, jako je adresa prodejny, či její název a následně pošle požadavek na server pro uložení. Po uložení prodejny do databáze má uživatel možnost zadat přístupový kód od prodejny, který bude využíván zákazníky pro přístup do prostorů obchodu. Po zadání kódu je vygenerován náhodný řetězec znaků, který je poslán na server a je využit jako obsah QR kódu, který je serverem vygenerován. Ten je následně uložen na disk serveru a jeho cesta je společně s náhodným řetězcem a přístupovým kódem uložena do databáze. Vygenerovaný QR kód, společně se zadaným přístupovým kódem prodejny je dostupný z administrátorského pohledu a je možné měnit jak již přiřazený QR kód společně s přístupovým kódem, či pouze přístupový kód.



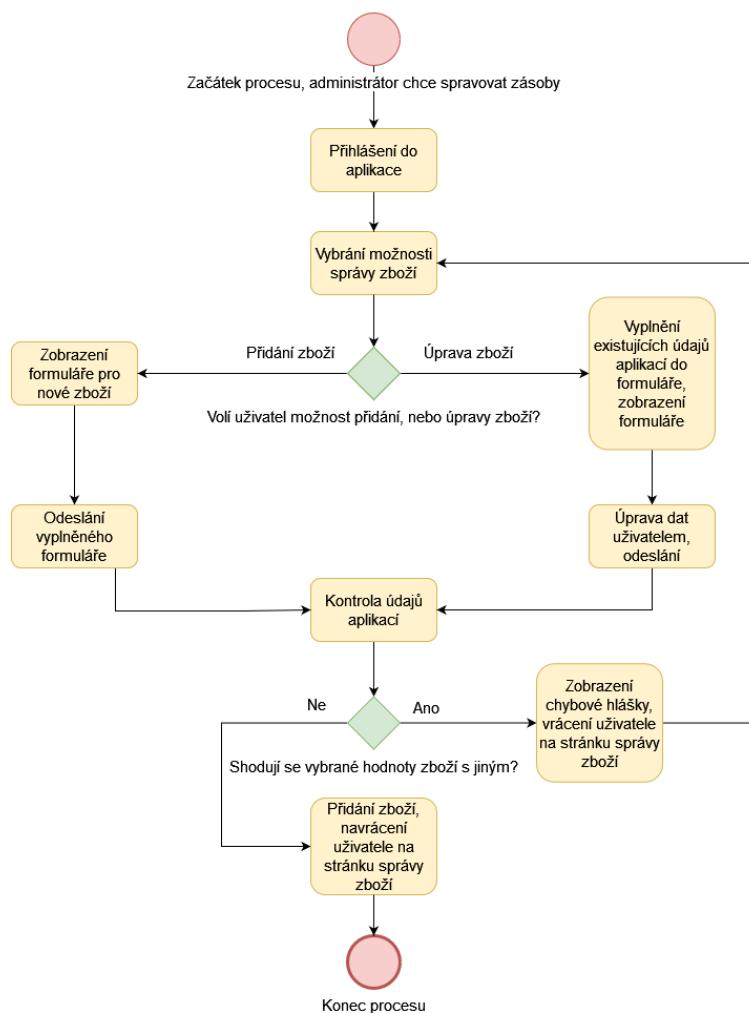
Obrázek 13 Diagram přidání prodejny a příslušného QR kódu

## Přidání nového produktu, úprava existujícího produktu

Administrátor s rolí A: Zboží má možnost přidat nový typ zboží do seznamu zboží již existujícího. Je potřeba vyplnit údaje jako je název zboží, cena, sazba DPH, typ balení, identifikační kód zboží z čárového kódu a jeho popis. Je také možnost přiřadit několik nahraných obrázků, které uživatel nahrál na server před přidáním zboží.

Následně jsou zadané údaje porovnány se záznamy v databázi a pokud se některý údaj shoduje, uživatel je upozorněn na tento problém a vyzván k opravě. V opačném případě je zboží přidáno do seznamu a zpřístupněné k přidání k jakémukoliv prodejnímu místu.

Podobný postup je také využit při úpravě zboží, kdy uživatel vybere produkt, který zamýšlí upravit, jeho data jsou nahrána do formuláře pro úpravu a následně jsou porovnávána s existujícími daty při odeslání.



Obrázek 14 Diagram správy zásob

### 3.3 Návrh uživatelského rozhraní

Uživatelské rozhraní je důležitou součástí všech aplikací. V této kapitole je ukázán postup vývoje uživatelského rozhraní, od prvního návrhu až po finální verzi vytvořenou před zahájením vývoje aplikace, jak desktopového, tak mobilního rozhraní.

Veškeré návrhy uživatelského rozhraní byly vytvořeny skrze aplikaci Figma (Figma, Inc., 2024), jejíž primární účel je vytváření wire-frame návrhů uživatelského rozhraní.

Stránky popsané v desktopovém návrhu nebyly speciálně upraveny pro mobilní rozložení – i přesto je možné je využívat na mobilních telefonech, avšak uživatelský zážitek není takový, jako na desktopu.

#### 3.3.1 Návrh desktopového rozhraní

Následující podkapitola se zaměřuje na přiblížení návrhu stránek, které jsou primárně určené pro rozhraní desktopového prohlížeče. Tyto stránky jsou z velké části primárně určené pro správce aplikace pro jednodušší přístup k obsahu aplikace.

##### Desktop domovská stránka

Desktopové rozhraní aplikace bude obsahovat domovskou stránku, která bude obsahovat základní uživatelské ovládací prvky, jako je lišta s odkazy na ostatní stránky, logo aplikace sloužící jako návratové tlačítko na domovskou stránku a tlačítko profilu, skrze které bude moci uživatel přistupovat do dalších součástí aplikace, jako je správa uživatelů, či zobrazení nákupů.

Stránka bude také obsahovat dva prvky carousel, které budou ukazovat dostupné zboží a případně zboží dostupné ve slevě.

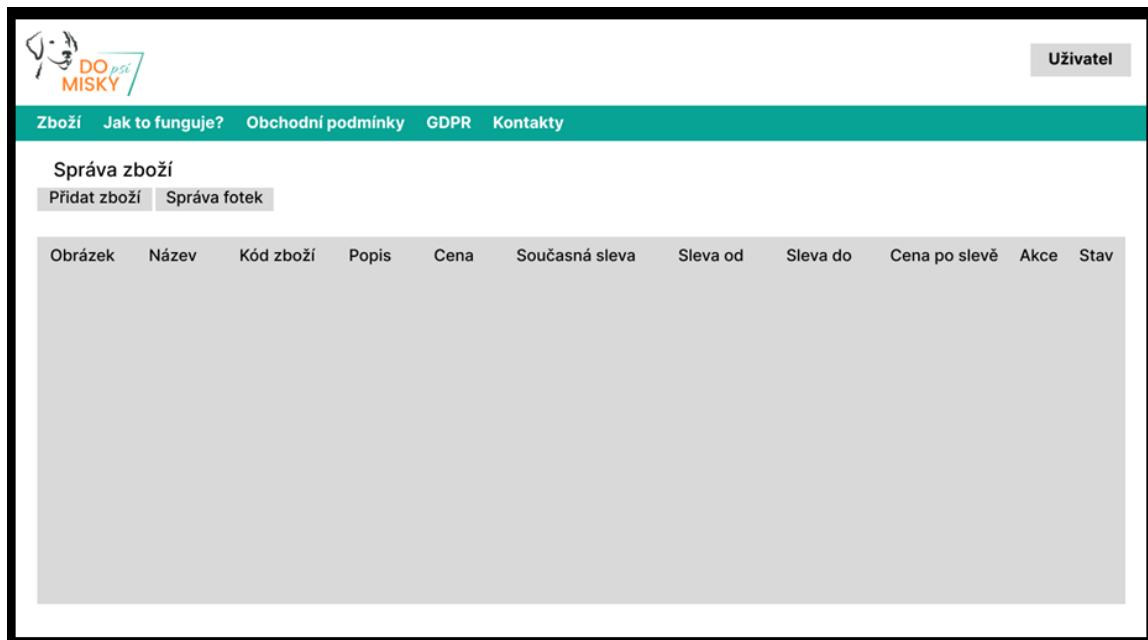


Obrázek 15 Wireframe pro desktop domovskou stránku

## Správa zboží

Správa zboží je primárně určena pro desktopové prostředí. Bude obsahovat veškeré zboží, které bude dostupné pro přidání ve všech prodejnách. Zboží bude ukázáno v tabulce, která obsahuje potřebné údaje o veškerém zboží, jako je název, cena, popis, fotografie produktu, nastavení slevy, či akční tlačítka pro akce, či možnost produkt aktivovat, či zakázat.

Stránka také obsahuje tlačítka pro otevření modálních oken pro nahrání fotografií produktů, či pro správu fotografií (přejmenování, smazání).

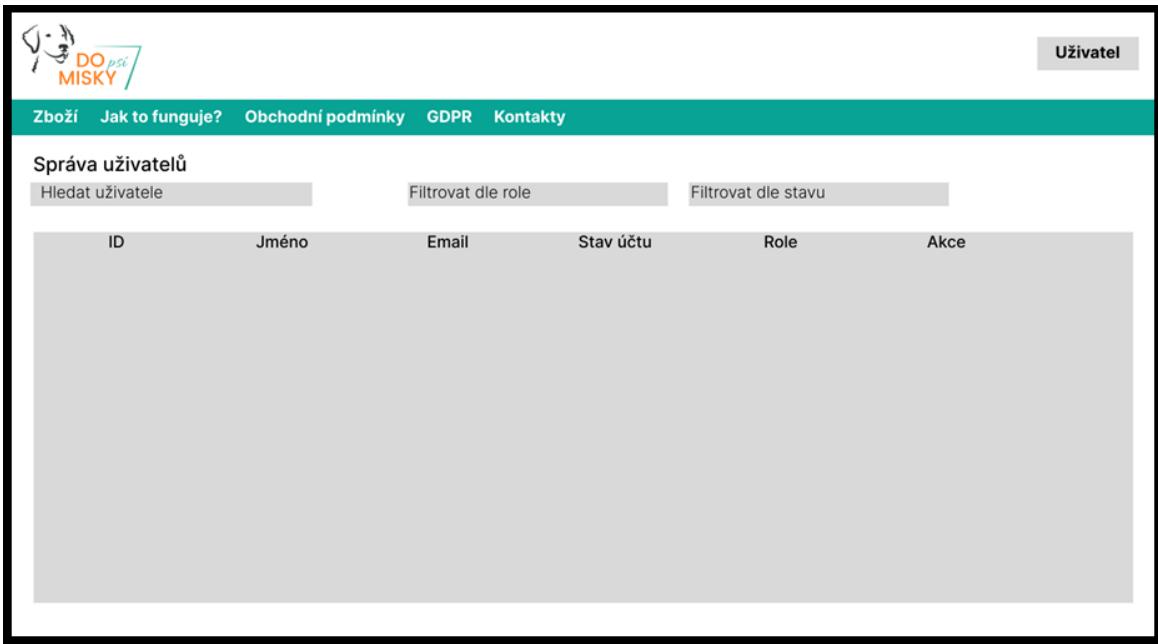


Obrázek 16 Wireframe pro správu zboží

## Správa uživatelů

Správa uživatelů je primárně určena pro desktopové prostředí. Bude obsahovat seznam veškerých uživatelů, kteří jsou zaregistrováni. Administrátor má možnost vidět ID uživatele, jeho jméno, email, stav účtu a jeho role. Administrátor skrze toto prostředí může upravovat role uživatele, či aktivovat/pozastavit/reaktivovat uživatelský účet.

Stránka také obsahuje vyhledávací pole pro hledání uživatelů a filtry pro zúžení vyhledávaných uživatelů skrze role či stavu účtu.

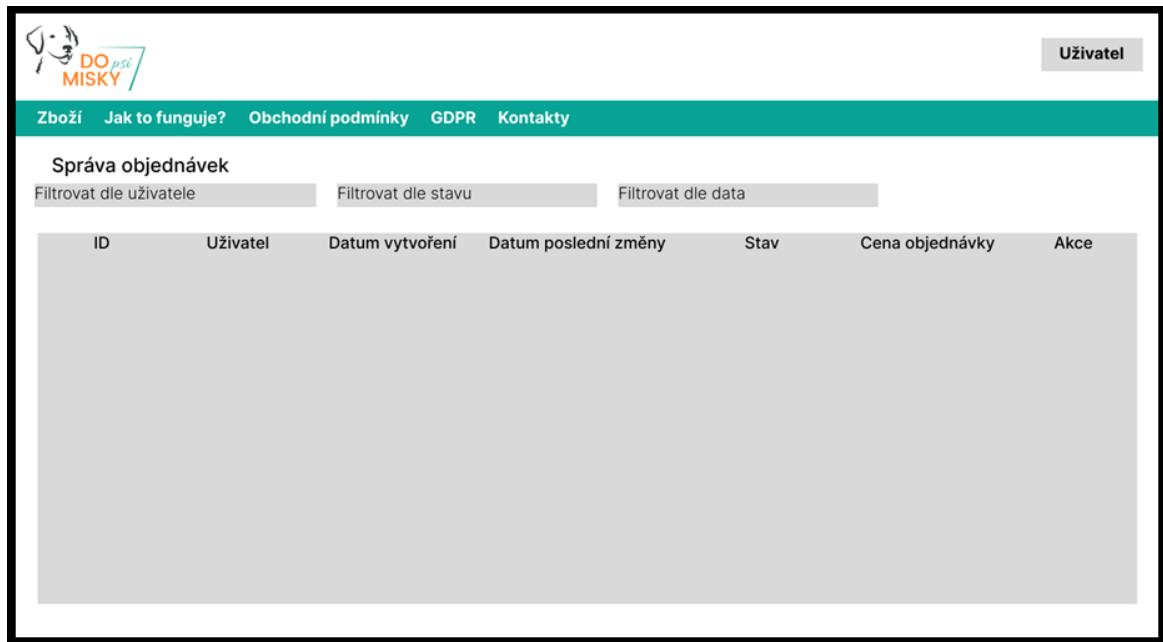


Obrázek 17 Wireframe pro správu uživatelů

## Správa objednávek

Správa objednávek slouží pro zobrazení provedených objednávek odeslaných na platební bránu. Zobrazuje ID objednávky, jméno uživatele který provedl objednávku, datum vytvoření a poslední změny, stav, celkovou výši objednávky a tlačítka akcí.

Uživatel může seznam objednávek filtrovat dle jména uživatele, dle stavu objednávky a dle data vytvoření.

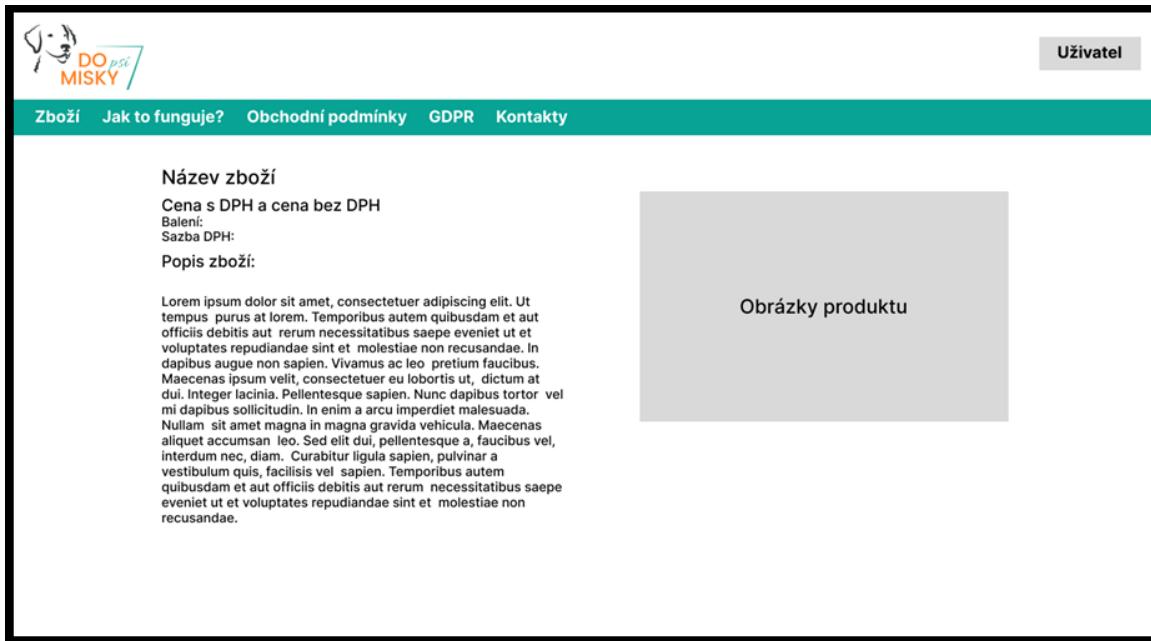


Obrázek 18 Wireframe pro správu objednávek

## Desktopové zobrazení produktu

Zobrazení produktu (Obrázek 19) je na desktopu určenou pouze jako detail produktu při procházení seznamu zboží. V budoucnu bude pravděpodobně využitou v případě rozhodnutí migrace zbytku e-shopu pro autorovi aplikace zadavatelem.

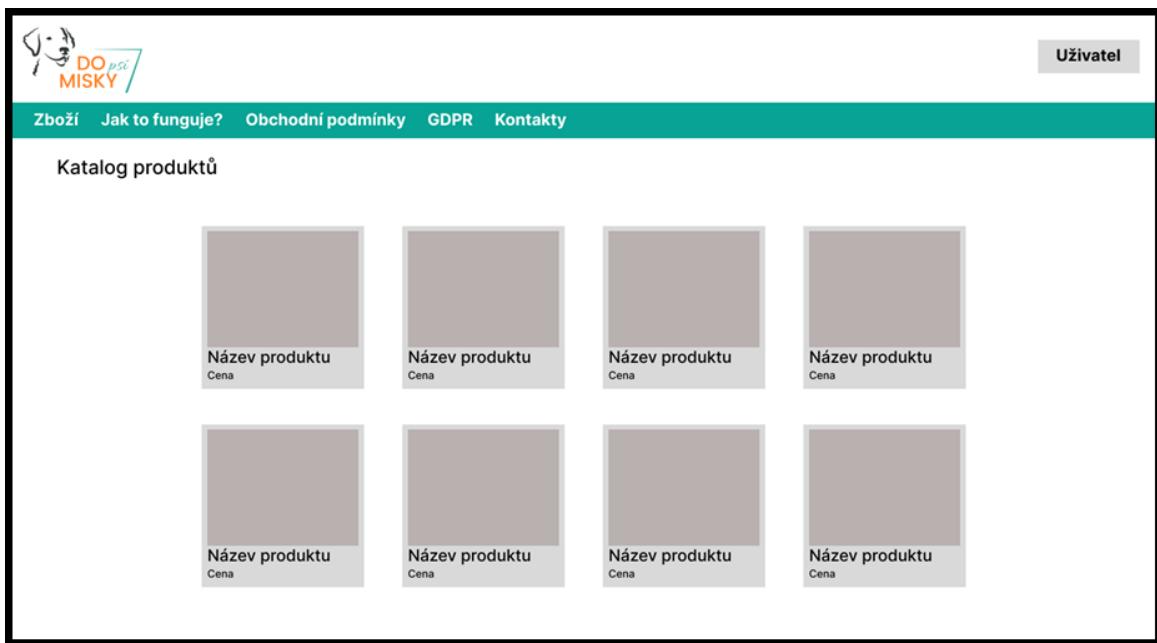
Stránka zobrazuje název zboží, cenu s a bez DPH (v případě slevy je originální cena přeskrtнута a kontrastní barvou nahrazeno cenou po slevě), velikost balení, sazba DPH a popis zboží. Vše je doplněno seznamem obrázků přiřazených ke zboží.



Obrázek 19 Wireframe pro zobrazení detailu zboží

### Desktopové zobrazení katalogu

Katalog (Obrázek 20) slouží k zobrazení veškerých produktů, které jsou v databázi aktivní. Po kliknutí na daný produkt je zobrazena stránka zobrazení produktu (viz Obrázek 19). Stránka se bude dynamicky upravovat velikosti obrazovky zařízení využívané uživatelem pro zobrazení.



Obrázek 20 Wireframe stránky katalogu

### 3.3.2 Návrh mobilního rozhraní

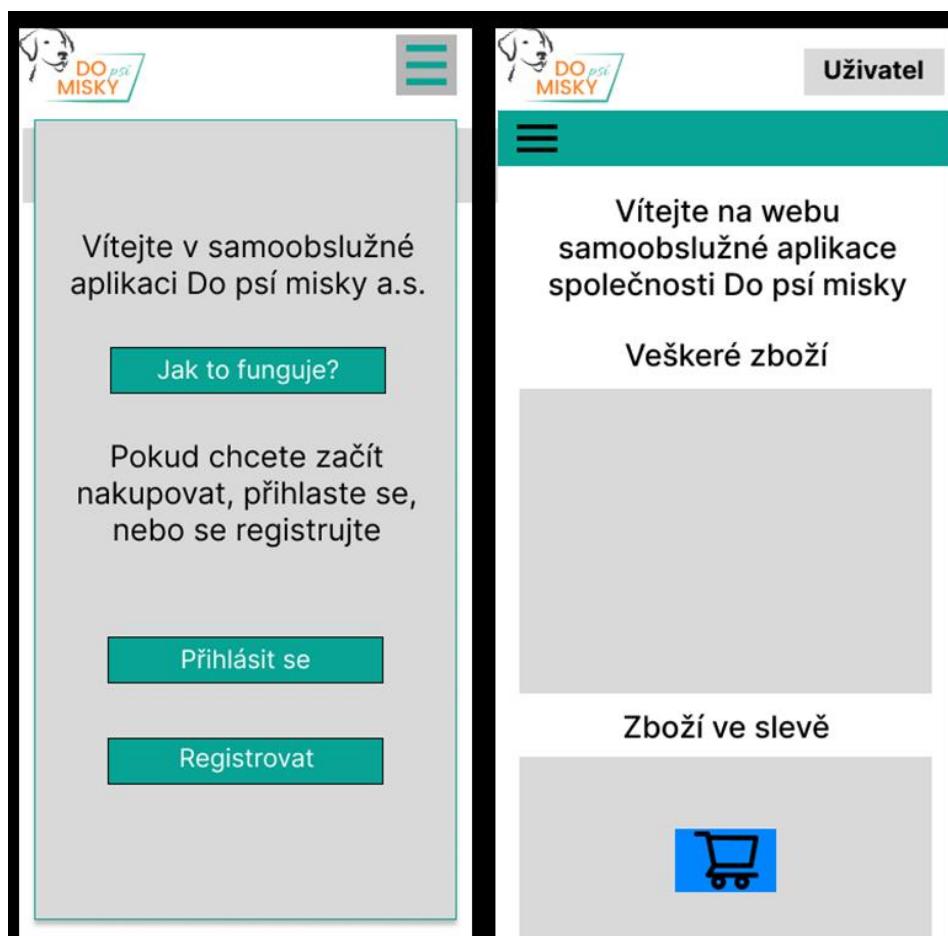
Tato podkapitola se zaměřuje na přiblížení návrhu stránek, které jsou primárně určené pro mobilní rozhraní počítače, zamýšleno pro uživatele nakupující v prodejnách.

Každá stránka, pokud není uvedeno jinak, obsahuje jak původní návrh, tak předělaný návrh po diskusi rozhraní se zadavatelem projektu a vedoucím bakalářské práce. Na obrázku se na levé straně nachází původní návrh a na levé návrh po konzultacích a úpravách.

#### Mobilní domovská stránka

Původní mobilní domovská stránka byla vytvořena kompletně odděleně od desktopového rozhraní. Obsahovala pouze možnosti pro zobrazení FAQ stránky, registračního formuláře, formuláře pro přihlášení a nabídky v horním panelu pro zobrazení informací jako byl uživatelský účet, či kontaktní stránka.

Nové mobilní rozhraní se přibližuje desktopovému návrhu, kde je zobrazen dvakrát prvek carousel, horní menu uživatele a lišta stránek. Pokud je uživatel přihlášen a má aktivní účet, je mu zobrazen prvek s tlačítkem pro zahájení nákupy v podobě vznášejícího se tlačítka s ikonou nákupního košíku.



Obrázek 21 Původní a upravený mobilní design domovské stránky

## Stránka zahájení nákupu

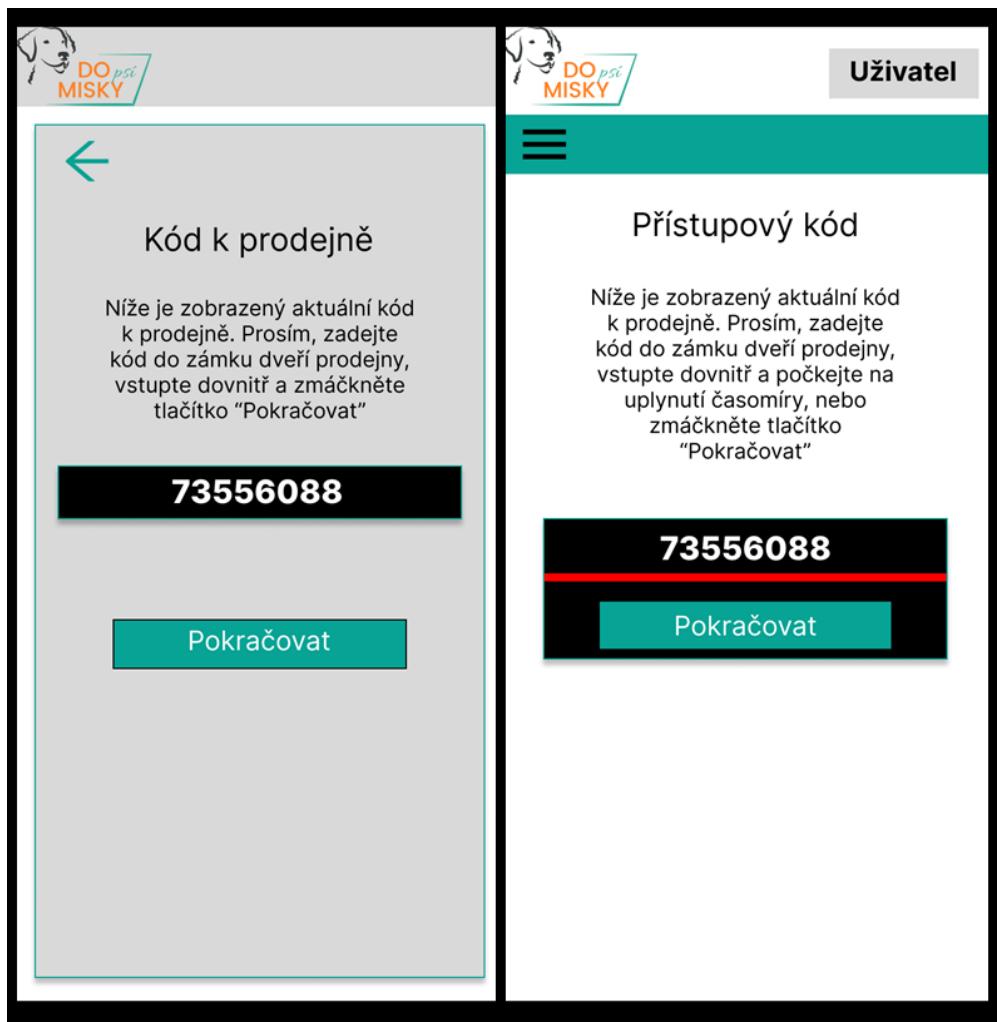
Stránka zahájení nákupu (viz Obrázek 22) umožňuje uživateli zahájit proces nákupu. Disponuje informacemi o nákupním procesu a jednoduchým rozhraním. Oba styly designu se velice podobají, i když nový design se pokouší přiblížit vzhledem zbytku aplikace.



Obrázek 22 Původní a upravený mobilní design stránky zahájení nákupu

## Stránka zobrazení přístupového kódu

Stránka zobrazení přístupového kódu (viz Obrázek 23) po úpravách disponuje časomírou 60 sekund, po které uživateli bude automaticky zobrazena zpráva, zdali vystoupil do prodejny, namísto neomezeného zobrazení přístupového kódu. Uživatel může časomíru přeskočit a rovnou přestoupit ke stránce nákupu.



Obrázek 23 Původní a upravený mobilní design stránky pro zobrazení kódu prodejny

#### Stránka zobrazení košíku/skenování zboží

Stránka zobrazení košíku (viz Obrázek 24) byla původně zamýšlena odděleně od stránky nákupního procesu skenování (viz Obrázek 25), po konzultaci se zadavateli bylo rozhodnuto stránku spojit přímo se stránkou určenou pro skenování zboží, a vytvořit tak jednodušší nákupní proces pro uživatele aplikace.

Po přechodu ze stránky pro naskenování QR kódu a vstupu do prodejny je uživateli ukázán košík se seznamem jeho zboží (viz Obrázek 26), společně s ovládacím prvky pro skenování zboží, či ukončení nákupu.

Po rozkliknutí možnosti pro skenování je uživateli zobrazeno modální okno se čtečkou čárových kódů (viz Obrázek 27). Po úspěšném naskenování je uživateli zobrazen detail produktu v modální okně (viz Obrázek 28), kde si uživatel může vybrat kolik kusů zboží bude přidáno do jeho košíku.



Obrázek 24 Wireframe původního designu košíku



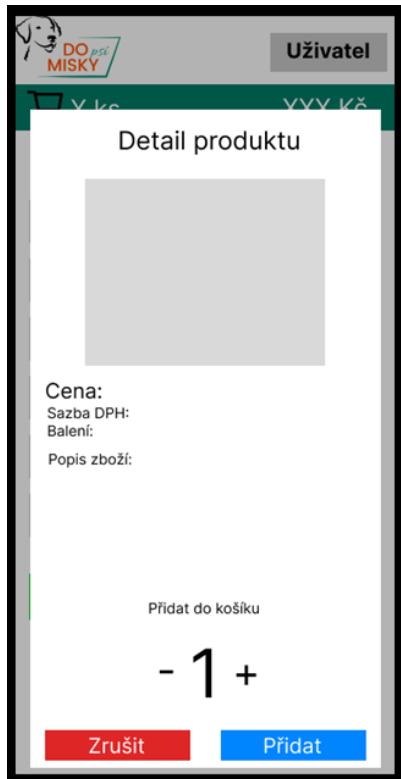
Obrázek 25 Wireframe původního designu stránky pro skenování zboží



Obrázek 26 Wireframe spojeného košíku a ovládacích prvků pro nákup zboží



Obrázek 27 Wireframe nákupního rozhraní s otevřenou čtečkou kódů v modálním okně



Obrázek 28 Wireframe nákupního rozhraní s otevřeným popisem produktu modálním okně

### 3.4 Technické prvky implementace

Následující podkapitola přibližuje uvažované technické prvky aplikace a případně zdůvodňuje výběr daného produktu pro výslednou aplikaci. Je zde popsáno řešení pro ukládání dat, provádění plateb a pohánění webu samotného.

#### Datový model

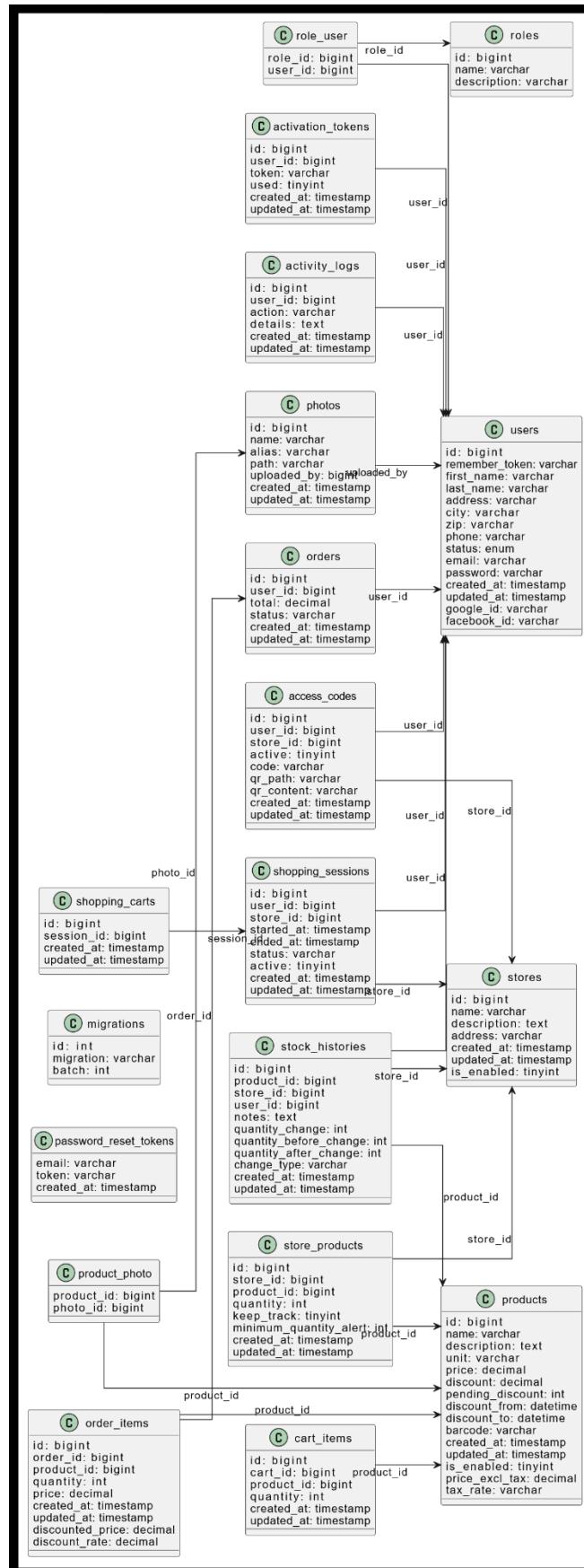
Databáze, která je základem pro webovou aplikaci, byla navržena s cílem zajištění efektivní správy a bezpečného ukládání dat. Tento návrh klade velký důraz na modularitu a rozšiřitelnost, aby bylo možné systém snadno aktualizovat a přizpůsobovat budoucím potřebám bez nutnosti zásadních změn v architektuře. Důležitým aspektem je i optimalizace pro různé typy databázových operací, což zahrnuje vše od běžných transakcí po složitější dotazy pro zpracování a analýzu dat. Struktura databáze je tvořena několika klíčovými tabulkami, které jsou propojeny převážně relacemi typu 1:N, umožňující

efektivní správu vztahů mezi daty, v některých případech však bylo potřeba použít i relaci typu M:N, například při přiřazení rolí danému uživateli.

Významné tabulky databáze jsou například:

- Tabulka **users** – je základním prvkem pro správu uživatelů aplikace, obsahuje základní informace o uživatelích, jako jejich jméno, zašifrované heslo, email adresu, či bydliště.
- Tabulka **roles** – ukládá uživatelské role, které jsou dále přiřazovány uživatelům. Pomocí příslušného middleware ve frameworku Laravel jsou role využívány pro omezení přístupu k daným stránkám aplikace.
- Pivot tabulka **role\_user** – slouží jako propojovací tabulka mezi tabulkami users a roles. Pomocí vazby M:N je možno přiřadit několik rolí několika uživatelům, což je využito při rozdělování přístupových práv mezi administrátory
- Tabulka **products** – obsahuje podrobné informace o každém z produktu, jako je název, cena, sazba DPH. Dále obsahuje údaje o naplánovaných slevách a případně aktivních slevách, které jsou automaticky zaváděny, či rušeny aplikací.
- Tabulka **orders** – uchovává objednávky vytvořené uživateli. Každá objednávka má přesně jednoho uživatele, je tedy volena vazba 1:N.
- Tabulka **order\_items** – specifikuje položky dané objednávky. Je propojena cizím klíčem k dané objednávce a danému produktu, navíc doplňuje informace jako byla cena v případě slevy a její výši.
- Tabulka **stock\_histories** – určena pro zaznamenávání změn ve skladových položkách daných prodejen, následně je využita jako zdroj dat pro zasílání reportů o změně stavu zboží.
- Tabulka **access\_codes** – obsahuje přístupové kódy prodejen společně s odkazem na umístění daného QR kódu na serveru. Je zde také ukládán obsah daného QR kódu, který je využit k ověření správnosti kódu naskenovaného uživatelem.
- Tabulka **stores** – eviduje fyzické prodejny vytvořené administrátory aplikace
- Pivot tabulka **store\_products** – propojuje tabulky stores a products, definuje, jaké produkty se nachází v prodejně. Obsahuje také informace o tom, zdali je daný produkt nastaven pro sledování minimální dovolené hodnoty na skladu dané prodejny. Pokud klesne pod minimální hranici, je příslušný administrátor upozorněn periodicky mailem pomocí aplikace.

V následujícím obrázku (Obrázek 29) jsou popsány vazby entit v databázi využívané aplikací.



Obrázek 29 Schéma databáze

## Provádění plateb

Pro provádění plateb v aplikaci bylo zváženo několik dostupných platebních bran, které se současně nacházejí na českém trhu. Jedná se o brány ThePay, Comgate a GoPay.

ThePay se prezentuje jako uživatelsky přívětivé a efektivní řešení, které nabízí téměř 20 platebních metod, včetně off-line platby pomocí naskenování QR kódu, kterou uživatel zadává manuálně ve své bankovní aplikaci. Její silnou stránkou je zákaznická podpora, která je klíčová pro mnohé obchodníky. Poplatky této brány se pohybují od 0,79 % až do 0,99 % dle tarifu, měsíční poplatek je až 139 Kč měsíčně (FastCentrik, 2023) (ThePay, 2024).

Comgate je brána podobná ThePay, s rozdílem podpory až 50 platebních metod, podporou zrychlených bankovních převodů a širokou podporou zahraničních měn. Brána také nabízí možnost odložených plateb. I přes to je brána známá svojí spolehlivostí a rychlostí. Také nabízí širokou flexibilitu při cenotvorbě, kde jsou poplatky za transakce velice příznivé, poplatky z platby kartou se pohybují od 0,69 % až 0,9 %, měsíční poplatky se mohou vyšplhat až na 200 Kč (FastCentrik, 2023) (Comgate, 2024).

Poslední platební bránou je GoPay, která je široce rozšířená po českém trhu, disponuje 55 platebními metodami, platbami na jedno kliknutí a garancí velkého množství dokončených objednávek. Také disponuje možností platit například pomocí Bitcoinu, PayPalu, Apple Pay nebo Google Pay. GoPay přináší vysokou kompatibilitu s e-commerce platformami, bezpečnost a spolehlivost. Obsahuje možnost integrace platební brány inline stylem, což umožňuje uživatelům pohodlně platit přímo na webu, kde nakupují bez zbytečných přechodů na stránky jiné. Dle obratu je transakční poplatek od 0,9 % do 2,2 % + 3 Kč za každou platbu. Poplatky za vedení brány mohou dosáhnout až na 190 Kč měsíčně (FastCentrik, 2023) (GoPay, 2023).

Vlastnost / Platební brána	ThePay	Comgate	GoPay
Platební metody	Téměř 20	Až 50	55, Bitcoin, PayPal
Zákaznická podpora	Vysoká	Vysoká	Vysoká
Transakční poplatky	0,79 % až 0,99 %	0,69 % až 0,9 %	0,9 % až 2,2 % + 3 Kč za platbu
Měsíční poplatek	Až 139 Kč	Až 200 Kč	Až 190 Kč
Spolehlivost a rychlosť	Vysoká	Vysoká	Vysoká
Flexibilita cenotvorby	Omezená	Vysoká	Vysoká
Možnost inline platby	Ne	Ne	Ano

Tabulka 1 Srovnání platebních bran

Po porovnání platebních bran (viz. Tabulka 1) je jasné, že porovnávaná řešení se silně podobají a je rozhodující dostupnost platebních metod a výše transakčních poplatků.

Jako finální platební brána byla vybrána GoPay, jak již z důvodů bezpečnosti, rozšíření, tak hlavně z důvodu možnosti zabudovat platební bránu přímo do webové aplikace pro snížení rizika jakéhokoliv přerušení transakce, které by mohlo nastat. Transakční poplatky jsou sice vyšší, o to má ale více využitelných možností, ze kterých si může zadavatel vybrat a svoji bránu si přizpůsobit svým potřebám. Navíc poskytuje okamžitou zpětnou vazbu o průběhu transakce jak na straně zákazníka, tak správce webu. Platební brána navíc může být dále integrována do aplikace v případě rozvoje a vytvoření nových funkcí, např. vytvoření části fungující jako e-shop.

## **PHP**

PHP je serverový skriptovací jazyk, který je základem mnoha webových aplikací. Je známý pro svou snadnou použitelnost a schopnost rychle vyvíjet dynamický web. PHP je široce používáno díky podpoře široké škály databázových systémů a integraci s HTML (PHP Group, 2023). Ve vyvíjeném projektu byla jako základní programovací platforma zvolena nejnovější stabilní verze PHP, a to PHP 8.3.

Tato verze přináší řadu významných vylepšení, která zahrnují nové funkce, syntaktické změny a zdokonalení typování, což přispívá k lepší čitelnosti a udržitelnosti kódu. Obsahuje také nejnovější bezpečnostní aktualizace a dlouhodobou podporu až do roku 2026 (PHP Group, 2023).

## **Laravel**

Laravel je populární PHP framework známý pro svou jednoduchost, eleganci a výkon. Je založen na architektonickém vzoru MVC (Model-View-Controller), který rozděluje aplikaci do tří hlavních komponent: Model, View a Controller. Tento vzor umožňuje efektivní oddělení uživatelského rozhraní, datového modelu a logiky řízení aplikace (Hanif, 2023).

Modely reprezentují databázové tabulky a obsahují logiku, která je potřeba pro manipulaci s daty v tabulkách. Využívají Eloquent ORM, což je jednoduchý způsob, jak komunikovat s databází – místo příkazů pro přímou práci s databází se styl je využíván Active Record, což znamená, že objekty modelů obsahují vlastní logiku pro manipulaci.

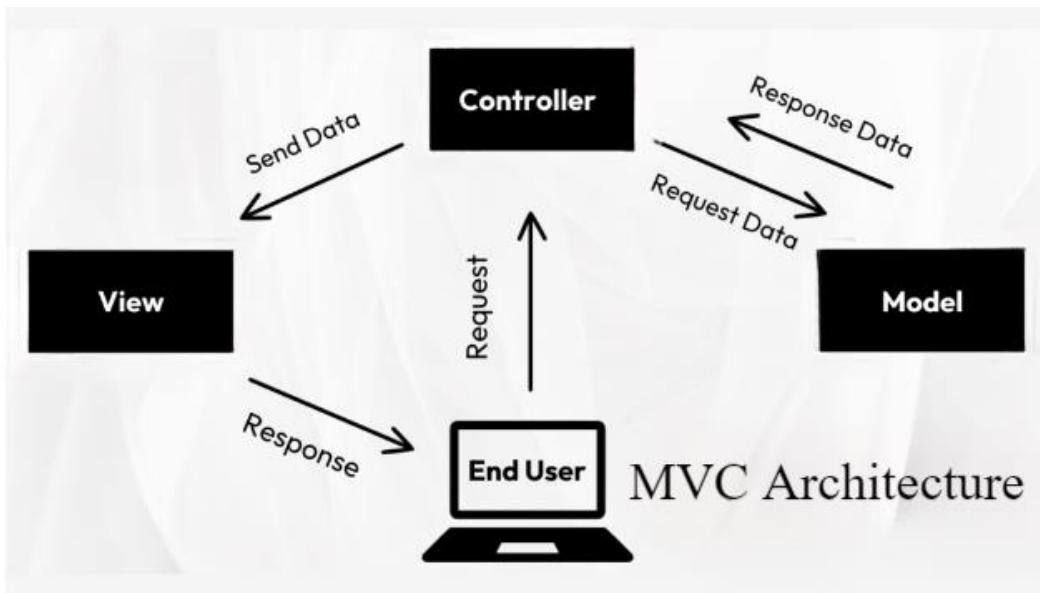
Views v Laravelu představují klíčovou roli při prezentaci dat uživateli. Jsou určeny k vykreslování uživatelského rozhraní na základě získaných dat z modelů a kontrolerů. Laravel používá Blade jako svůj templating engine, který umožňuje používat jednoduché šablony, které mohou obsahovat případné podmínky, smyčky, nebo například nabízí dědění a vkládání šablon do jiných.

Kontrolery fungují jako prostředníky mezi modely a views, zpracovávají vstuupy od uživatelů a následně pracují se zbytky aplikace pro zpracování požadavku a následné zobrazení výsledku. Kontrolery také mohou využívat middleware pro kontrolu přístupu a další zpracování požadavků před jejich dosažením. Data vkládána do kontrolerů také mohou být validována proti vytvořeným šablonám.

Funkčnost těchto tří prvků je také někdy doplněna dalšími součástmi, jako je middleware, který zachytává HTTP požadavky pro zpracování a modifikaci, než dosáhnou kontrolerů.

Často je využíván pro ověření autentizace, autorizace, nebo pro logování. Na jeden požadavek může být použito několik middleware najednou, což usnadňuje řetězení akcí.

V Laravelu také existují routes, které mapují definici mezi HTTP požadavky a jejich přidělení v kontrolerech. Každá route určuje, která HTTP metoda a URI by měla být zpracována kontrolerem. Jsou podporovány všechny běžné HTTP metody. V routes je také možno namapovat příslušný middleware, který bude použit při zpracování požadavku, či přidat jméno routy, které usnadňuje následné využití v šablonách.



Obrázek 30 Schéma MVC architektury (Ibrahim, 2024)

## Typy aplikací v Laravelu

Laravel, jakožto flexibilní a přizpůsobitelný framework, podporuje několik typů architektur, což umožňuje vývojářům volit ideální přístup podle povahy projektu. To rozdelení je možné díky modularitě a rozsáhlým možnostem přizpůsobení. Můžeme tak pozorovat následující typy aplikací:

- **Monolitické aplikace** nabízejí jednoduchou strukturu, vhodnou pro menší týmy a projekty. Využívají kompletní sadu funkcí Laravelu, aby vývojáři měli všechny nástroje v jednom místě.
- **Microservices** umožňují rozdělit aplikaci do menších a nezávislých komponent. Laravel lze využít jako základní stavební kámen v architektuře mikroslužeb díky své modularitě.
- **Serverless** přináší serverless technologie, které snižují náklady a zvyšují flexibilitu škálování. Laravel může běžet v tomto prostředí díky své kompatibilitě a adaptabilitě.
- **REST API** je usnadněno díky podpoře směrování, ověřování a dalším funkcím, které poskytují výkonný nástroj pro integraci s dalšími systémy.

- **Jamstack** představuje moderní způsob vývoje, který odděluje frontend a backend. Laravel zde slouží jako spolehlivý backend, zatímco moderní staticky generované stránky obsluhují frontend.

## Použitá rozšíření a balíčky pro PHP

V projektu byly použity některé užitečné balíčky a rozšíření:

**barryvdh/laravel-dompdf:** Pro generování PDF souborů z HTML<sup>1</sup>.

**gopay/payments-sdk-php:** SDK pro integraci platební brány GoPay<sup>2</sup>.

**laravel/socialite:** Pro autentizaci prostřednictvím OAuth poskytovatelů<sup>3</sup>.

**laravel/tinker:** CMD nástroj pro rychlé testování a manipulaci s daty aplikace<sup>4</sup>.

**simplesoftwareio/simple-qrcode:** Pro generování QR kódů<sup>5</sup>.

Kombinace těchto balíčků spolu s PHP 8.3 a Laravel 10 poskytuje moderní prostředí pro vývoj bezpečných, škálovatelných a udržitelných webových aplikací. Tyto balíčky zjednodušují integraci různých funkcí a umožňují rychlejší vývoj aplikace, či její testování.

## JavaScript a knihovny

V rámci projektu bylo rozhodnuto o použití JavaScriptu, který je zásadním prvkem pro zajištění dynamiky a interaktivnosti uživatelského rozhraní webové aplikace. JavaScript je využíván pro realizaci asynchronních operací, manipulace s Document Object Model (DOM) a integrace různých externích knihoven, které rozšiřují funkčnost aplikace.

Klíčovou technologií pro asynchronní operace jsou AJAX a Fetch API, které umožňují webovým aplikacím asynchronně získávat data ze serveru bez nutnosti obnovovat celou stránku. Tato metoda umožňuje vytvářet plynulé a rychlé uživatelské rozhraní, které efektivně reaguje na uživatelské akce.

---

<sup>1</sup> <https://github.com/barryvdh/laravel-dompdf>

<sup>2</sup> <https://github.com/gopaycommunity/gopay-php-api>

<sup>3</sup> <https://github.com/laravel/socialite>

<sup>4</sup> <https://github.com/laravel/tinker>

<sup>5</sup> <https://github.com/SimpleSoftwareIO/simple-qrcode>

Pro účely skenování QR kódů byla ve webové aplikaci použita knihovna html5-qrcode<sup>6</sup>, která umožňuje snadné skenování QR kódů přímo z kamer mobilních zařízení. Tato knihovna byla zvolena kvůli své rychlosti, spolehlivosti a široké podpoře různých typů zařízení. Pro skenování čárových kódů byla použita knihovna QuaggaJS<sup>7</sup>, která je schopna dekódrovat čárové kódy z obrázků a videa. Její volba byla motivována její všestranností a přesností.

Kromě těchto specifických knihoven bylo rozhodnuto o integraci jQuery<sup>8</sup> a Bootstrapu<sup>9</sup> do projektu. jQuery je v projektu využívána pro zjednodušení manipulace s DOM a pro efektivní zpracování AJAX požadavků. Její použití usnadňuje práci s událostmi, což zvyšuje interaktivitu aplikace. Bootstrap byl zvolen pro stylizaci a vytváření responsivního designu uživatelského rozhraní. Tento framework poskytuje širokou škálu komponent, které zjednodušují vývoj konzistentního a vizuálně přitažlivého rozhraní.

Závěrem, kombinace JavaScriptu, AJAX, Fetch API a těchto knihoven v projektu vytváří silný základ pro rozvoj webové aplikace. Knihovny html5-qrcode a QuaggaJS umožňují skenování kódů, zatímco jQuery a Bootstrap zvyšují efektivitu a estetickou kvalitu uživatelského rozhraní. Tím je zajištěno, že webová aplikace bude moderní, interaktivní a uživatelsky přívětivá.

---

<sup>6</sup> <https://github.com/mebjas/html5-qrcode>

<sup>7</sup> <https://github.com/serratus/quaggaJS>

<sup>8</sup> <https://github.com/jquery/jquery>

<sup>9</sup> <https://github.com/twbs/bootstrap>

# 4 Implementace

Kapitola je zaměřená na popsání implementace webové aplikace, která bude sloužit jako základ pro samoobslužné bezkontaktní prodejny. Nejdříve je definována struktura aplikace, dále jsou podrobněji popsány nejdůležitější funkční části aplikace společně se souvisejícím kódem.

## 4.1 Struktura aplikace

Projekt je strukturovaný dle základního rozdělení Laravel projektů s drobnými změnami v oblasti views. Grafické zobrazení (viz Obrázek 31) znázorňuje strukturu celé aplikace.

Zobrazení stránek aplikace bylo rozděleny na 3 části, a to na **fullViews** (blade šablony využité jako hlavní části zobrazených stránek), do který se následně vkládají **modals** (blade šablony modálních oken) a to vše je doplněno pomocí **js** (Javascript pro zprovoznění obesílání požadavků a aktualizaci zobrazení). V tomto stylu byly složky, které obsahují tyto soubory, strukturované dle následujícího schéma:

- **adminApp** obsahuje části aplikace určené pro správce aplikace
- **adminProducts** obsahuje části aplikace určené pro správce zboží
- **adminStores** obsahuje části aplikace určené pro správce obchodů
- **adminUsers** obsahuje části aplikace určené pro správce uživatelů
- **public** obsahuje části aplikace dostupné pro všechny uživatele, jako je index, nebo zobrazení produktů v katalogu
- **shop** obsahuje části aplikace určené pro nákupní proces

Po pochopení tohoto rozdělení je nyní možné stanovit si celkovou strukturu složek aplikace.

Složka **app** obsahuje:

- **/app/Console**: Složka obsahuje soubor `Kernel.php`, který nastavuje scheduler funkci frameworku Laravel pro plánování opakujících se akcí.
- **/app/Console/Commands**: Složka ukládá soubory příkazů, které lze následně plánovat pomocí Laravel scheduleru. V první verzi aplikace se ve složce nachází příkaz pro odeslání notifikace o nízkém počtu zboží na skladu prodejen `SendLowStockAlert.php`.
- **/app/Http/Controllers**: Složka obsahuje všechny kontrolery vytvořeny autorem pro chod aplikace a kontrolery Laravel pro autentifikaci uživatelů.
- **/app/http/Middleware**: Složka obsahuje middleware použitý Laravelem včetně těch, které vytvořil autor.
- **/app/Listeners**: Obsahuje listener `UpdateUserStatusAfterVerification.php`, který mění stav uživatelského účtu po použití aktivačního odkazu v mailu.
- **/app/Models**: Obsahuje modely potřebné pro komunikaci s databází.

- **/app/Notifications:** Obsahuje vzory notifikací použité pro zasílání informačních mailů uživatelům.
- **/app/Providers:** Ukládá providery, což jsou části kódu sloužící pro registraci jiných komponent po spuštění aplikace, jako je mailer, či cache.

Složka **bootstrap** obsahuje složku **cache** a soubor `app.php`, který spouští instanci aplikace.

Složka **config** obsahuje konfigurační soubory součástí aplikace.

Složka **database** obsahuje části sloužící pro plnění databáze v případě použití migrací (vytvoření, doplnění, nebo odstranění tabulek v databázi), nebo přímo plnění záznamy s využitím factories a seederů.

Složka **node\_modules** obsahuje JavaScriptové knihovny použité Laravelem.

Složka **public:**

- **/public/resources/css:** Obsahuje soubor `app.css`, který nastavuje styl pro celou aplikaci.
- **/public/resources/images:** Uchovává fotografie využívané aplikací, které nelze upravit uživatelem.
- **/public/resources/js:** Obsahuje JavaScriptový kód pro modální okna a views, rozděleny do podsložek ke každému typu view (fullView, modal, nebo layout).
- **/public/storage/products:** Ukládá uživateli vkládané soubory produktů.
- **/public/storage/qr-code:** Slouží pro uložení QR kódů generované aplikací.

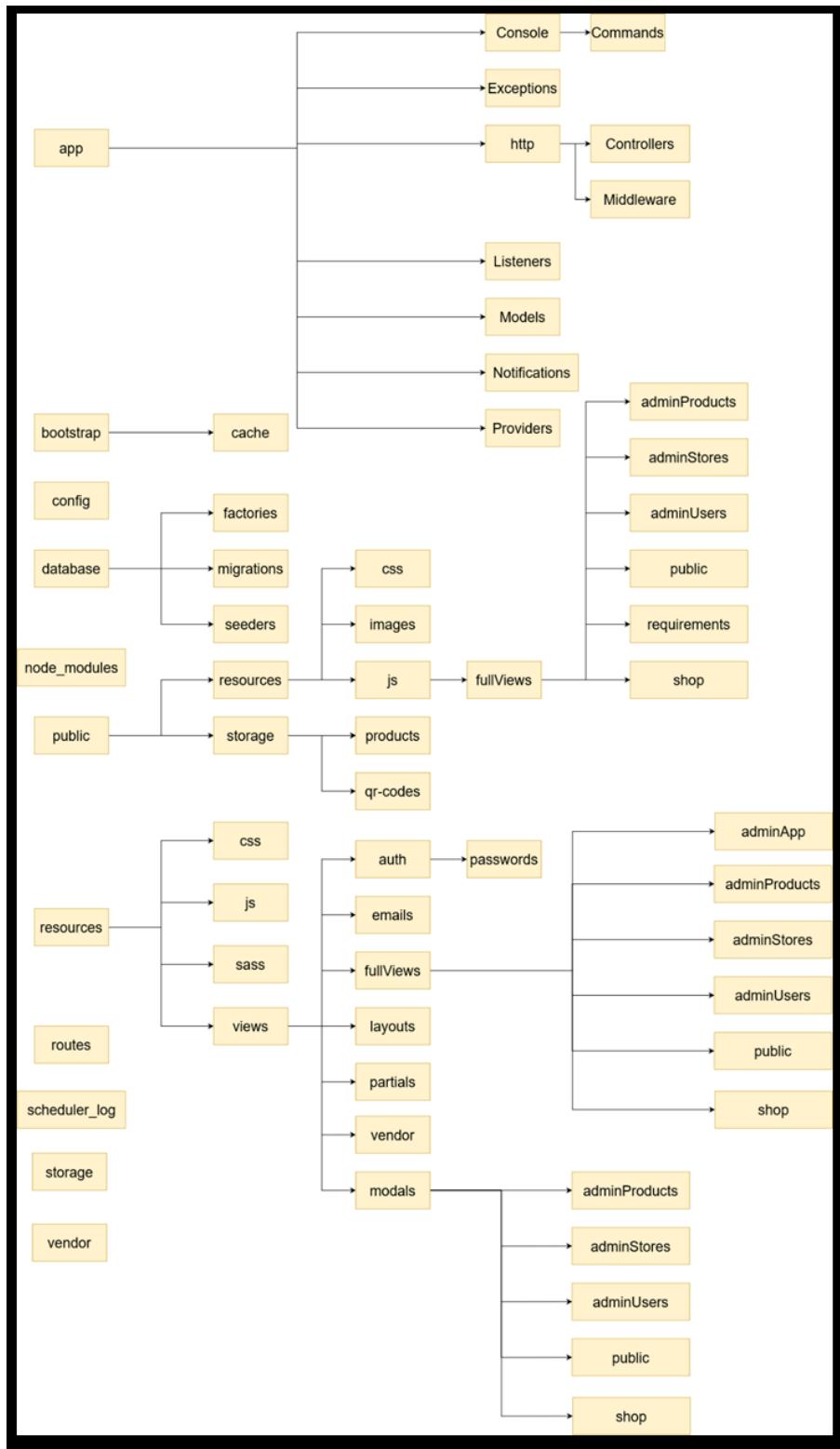
Složka **resources** obsahuje **css, js a sass** podsložky, které jsou využívané Laravelem pro stylování a doplňující funkce.

Složka **/resources/views** obsahuje několik podsložek:

- **/resources/views/auth:** Obsahuje podsložku **passwords**, ve které je uložen `reset.blade.php`, který funguje jako view pro resetování hesla.
- **/resources/views/emails:** Obsahuje šablony pro vytváření PDF souborů zasílané mailem.
- **/resources/views/fullViews:** Obsahuje šablony hlavních stran sloužící jako části aplikace (nákup, index, správa zboží...) dostupná v aplikaci rozděleny do podsložek.
- **/resources/views/layouts:** Obsahuje rozvržení `app.blade.php`, což je základní šablona, která je využívána ostatními views.
- **/resources/views/modals:** Obsahuje šablony pro modální okna, rozděleny do podsložek stejné struktury jako JavaScript a složka fullViews.
- **/resources/views/partials:** Obsahuje `footer.blade.php` a `header.blade.php`, což jsou views sloužící jako univerzální hlavička a patička stránek.
- **/resources/views/vendor/notifications:** Obsahuje `email.blade.php`, upravující výchozí vzhled mailů zasílané pomocí maileru.

Složka **routes** obsahuje soubory definující cesty při procházení stránek aplikace, či při zasílání požadavků. Autorem upravený soubor je pouze `web.php`.

Složky **storage** a **vendor** jsou ve správě frameworku Laravel a uživatelem nebyly nijak upraveny.



Obrázek 31 Složková struktura aplikace

## 4.2 Autorizace uživatelů

Každý uživatel aplikace má po registraci automaticky přiřazen roli „Uživatel“, se kterou má přístup do stejných částí aplikace, jako nepřihlášený uživatel. Pouze po aktivaci účtu skrze mail zaslaný po registraci má možnost přistupovat do dalších částí aplikace, jako je nákup zboží, zobrazení objednávek, či zobrazení a úprava údajů v profilu.

Administrace má práva rozdělena do několik podskupin. V prvním provedení aplikace jsou role rozděleny do „A: Aplikace“, „A: Uživatelů“, „A: Prodejen“ a „A: Zboží“, přičemž každý z administrátorů má navíc roli samotnou roli „Administrátor“. Každá z rolí dává uživateli možnost přistupovat k daným částem aplikace, respektive spravovat upozornění, všechny uživatele, prodejny a zboží.

Přístup k daným částem aplikace je řešeno skrze jednoduchý middleware EnsureUserHasRole (viz Výpis kódu 1), který u každého požadavku kontroluje, zdali má uživatel v databázi přidělenou roli potřebnou k přístupu a také to, jestli byl uživatelův účet aktivován. Pokud ano, je požadavek zpracován. Pokud ne, je uživatel navrácen na uvítací stránku. Definice přístupových rolí k danému požadavku je přímo u každé z routes (viz Výpis kódu 2).

Výpis kódu 1 Třída EnsureUserHasRole fungující jako middleware

```
class EnsureUserHasRole
{
    public function handle($request, Closure $next, ...$roles)
    {
        if (!$user || !$user->roles()->whereIn('name', $roles)->exists() || $user->status
!== 'activated')
        {
            return redirect('/')->with('error', 'Nemáte požadovaná práva pro přístup k
této části aplikace, nebo váš účet není aktivován.');
        }
        return $next($request);
    }
}
```

Výpis kódu 2 Ukázka použití EnsureUserHasRole middleware

```
Route::middleware(['auth', 'role:Uživatel'])->group(function () {
    Route::get('/scan-page', [ScanController::class, 'showScanPage'])->name('scan.page');
    Route::get('/scan-qr', [ScanController::class, 'scanQR'])->name('scan.qr');
});
```

## 4.3 Zobrazení úvodní stránky

Zobrazení indexové stránky je v aplikaci řešeno skrze controller indexController (viz Výpis kódu 3), který při využití routy `Route::get('/', [IndexController::class, 'index'])->name('home');` zobrazuje view index a předává mu informace o produktech z databáze. V případě, že je uživatel přihlášen (zjištěno pomocí funkce `auth()->user()`) a má aktivovaný účet, je předána proměnná `$showScanButton` v hodnotě 1, címž se v šabloně zobrazí tlačítko po zahájení procesu nákupu. Pokud se mezi produkty nachází některé produkty, které mají definovanou slevu, vrací se v proměnné `$discountedProducts`, ostatní produkty se vrací v proměnné `$products`. Následně se vrací vyplněný pohled index s informacemi o produktech a případným aktivním tlačítkem pro skenování produktů.

Výpis kódu 3 Funkce index() v indexController

```
public function index()
{
    $products = Product::all();

    $discountedProducts = Product::whereNotNull('discount')
        ->whereDate('discount_from', '<=' , Carbon::now())
        ->whereDate('discount_to', '>=' , Carbon::now())
        ->get();

    $user = auth()->user();
    $showScanButton = $user && $user->status == 'activated' && $user-
>hasRole('Uživatel');

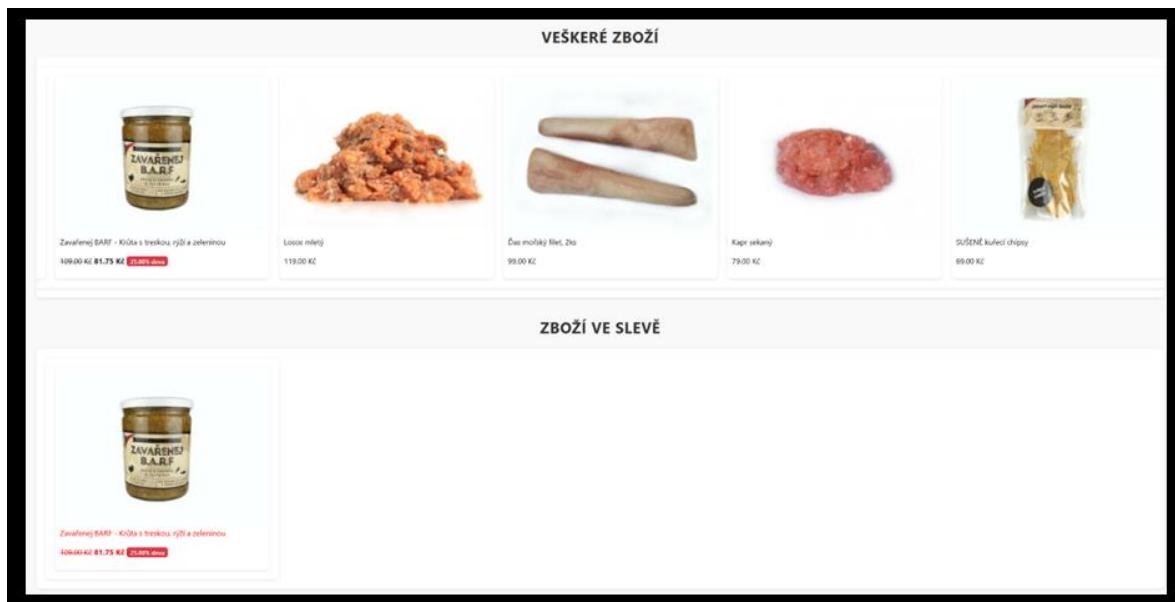
    return view('index', compact('products', 'discountedProducts', 'showScanButton'));
}
```

Samotný pohled index je definován jako blade šablona přebírající informace přímo z proměnných navrácených z indexController. Při zaměření na to, jak jsou data vyobrazována, můžeme rozebrat jednu část z indexové stránky, a to prvek carousel all-products (viz Výpis kódu 4), který v cyklu vypisuje produkty vrácené v proměnné `$products`. Je očekáváno, že do databáze byly veškeré produkty uloženy s potřebnými daty, které musí být vyplněné. Pokud není u produktu definována v pivot tabulce `product_photo` vazba na fotografií zboží, je zobrazen základní obrázek.

Speciálním případem je zboží se slevou, při kterém je původní cena přesktrnuta a je vyobrazena cena po slevě, společně se slevou (viz Obrázek 32). Zboží se slevou je také definováno v prvku carousel discounted-products (viz Výpis kódu 5), který zobrazuje pouze zboží ve slevě. Pokud není nalezeno zboží ve slevě, je jednoduše vypisán odkaz na stránku katalogu zboží.

Výpis kódu 4 Prvek carousel "all-products" z pohledu index

```
<div class="multiple-items all-products">
    @foreach ($products as $product)
        <div>
            <a href="{{ route('products.show', $product->id) }}">
                name }}">
                <p>{{ $product->name }}</p>
                @if ($product->discount > 0)
                    <p>
                        <del>{{ number_format($product->price, 2) }} Kč</del>
                        <strong>{{ number_format($product->price * (1 - $product->discount / 100), 2) }} Kč</strong>
                        <span class="badge bg-danger">{{ $product->discount }}% sleva</span>
                    </p>
                @else
                    <p>{{ number_format($product->price, 2) }} Kč</p>
                @endif
            </a>
        </div>
    @endforeach
</div>
```



Obrázek 32 Vizuální zobrazení prvků carousel „all-products“ a „discounted-products“

Výpis kódu 5 Prvek carousel „discounted-products“ z pohledu index

```
@if ($products->where('discount', '>', 0)->count() > 0)
<h3 class="carousel-heading">Zboží ve slevě</h3>
<div class="multiple-items discounted-products">
    @foreach ($products->where('discount', '>', 0) as $product)
        <div class="discounted">
            <a href="{{ route('products.show', $product->id) }}">
                name }}">
                <p>{{ $product->name }}</p>
                <p>
                    <del>{{ number_format($product->price, 2) }} Kč</del>
                    <strong>{{ number_format($product->price * (1 - $product->discount / 100), 2) }} Kč</strong>
                    <span class="badge bg-danger">{{ $product->discount }}% sleva</span>
                </p>
            </a>
        </div>
    @endforeach
</div>
@else
<div class="no-discount-message-container">
    <div class="no-discount-message">
        <p>V tento moment nejsou žádné produkty ve slevě, můžete si ale projít <a href="/catalog">katalog zboží</a>.</p>
    </div>
</div>
@endif
```

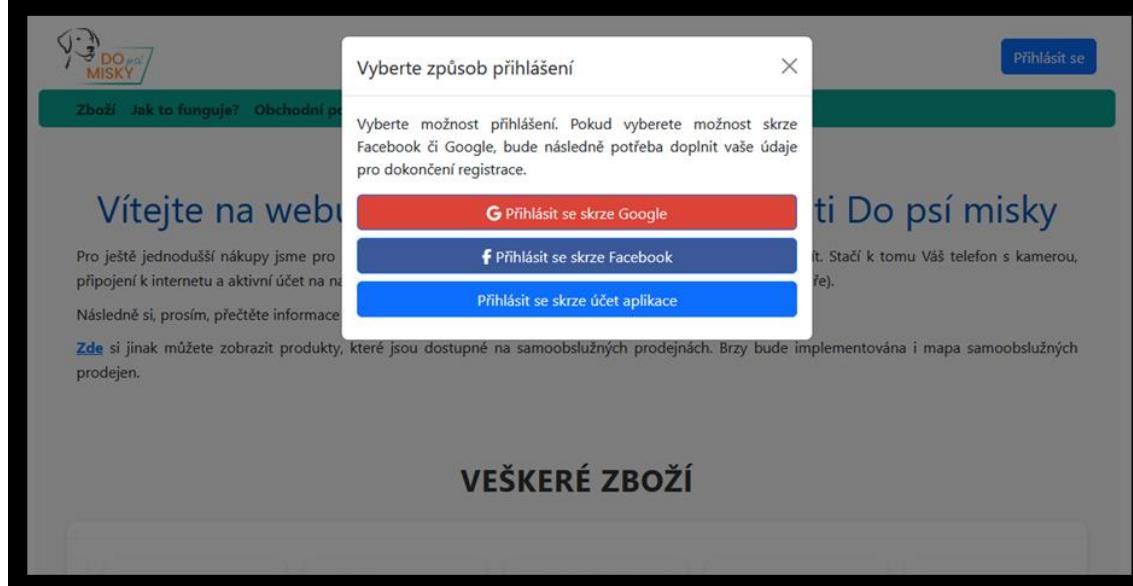
## 4.4 Přihlášení a registrace skrze lokální účet

Přihlášení a registrace v aplikaci probíhá skrze modální okna, která se aktivují po stisknutí tlačítka Přihlásit se v hlavičce stránky. Je zobrazeno modální okno, kde si uživatel může vybrat preferovaný typ přihlášení (viz Obrázek 33).

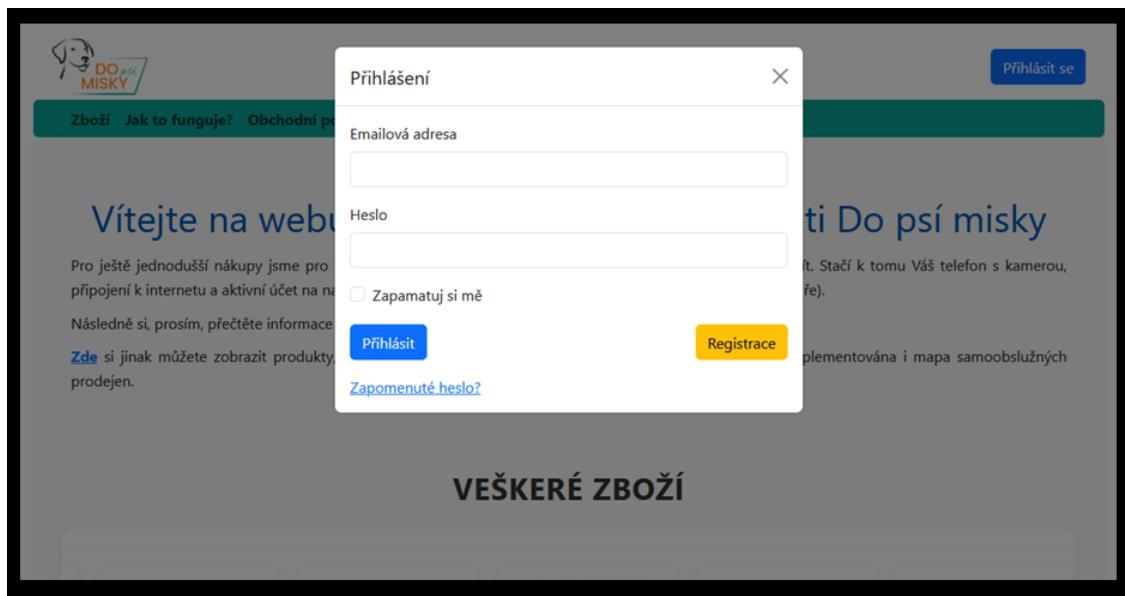
Při vybrání možnosti přihlášení skrze lokální účet je uživateli zobrazeno jednoduché přihlašovací okno (viz Obrázek 34), kde může jednoduše zadat své přihlašovací údaje, případně zaškrtnout možnost trvalého přihlášení. Požadavek je následně zpracován skrze LaravelUI. Pokud vše proběhlo správně, uživateli je zobrazena domovská stránka a je přihlášen.

Pokud uživatel vybere možnost registrace, je zobrazeno modální okno pro registraci (viz Obrázek 35). Po vyplnění údajů je jeho požadavek zpracován pomocí funkce `create` ve třídě `registrationController` (viz Výpis kódu 7).

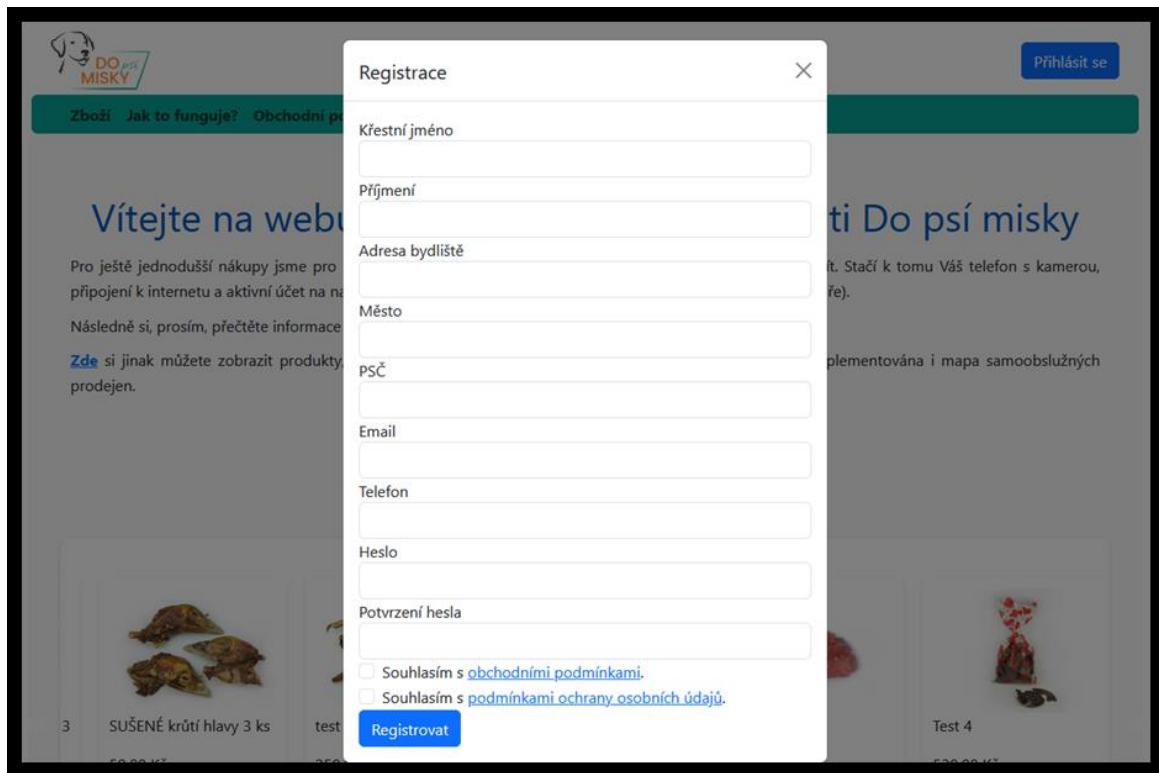
Při zažádání o obnovu hesla je uživateli zobrazeno okno s polem pro zadání mailu (viz Obrázek 36). Po odeslání je zkontovalo, jestli existuje uživatelský účet, který používá zadanou adresu. Pokud ano, je vytvořen token pro reset a ten je zaslán uživateli na mail. Po rozkliknutí odkazu stačí zadat mail, a dvakrát nové heslo (viz Obrázek 37). Po odeslání formuláře je uživateli nastaveno nové heslo a je přihlášen automaticky.



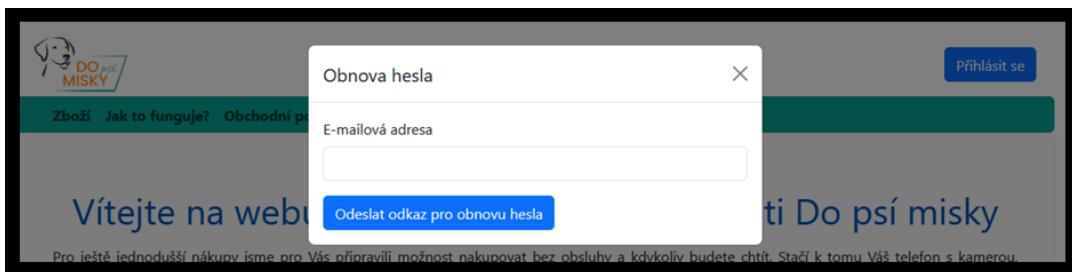
Obrázek 33 Zobrazení modálního okna pro výběr preferovaného typu přihlášení



Obrázek 34 Přihlašovací dialog pro lokální účet



Obrázek 35 Registrační dialog pro lokální účet



Obrázek 36 Modální okno pro obnovu hesla

Obrázek 37 Dialog pro resetování hesla

## 4.5 Přihlášení a registrace skrze účet třetí strany

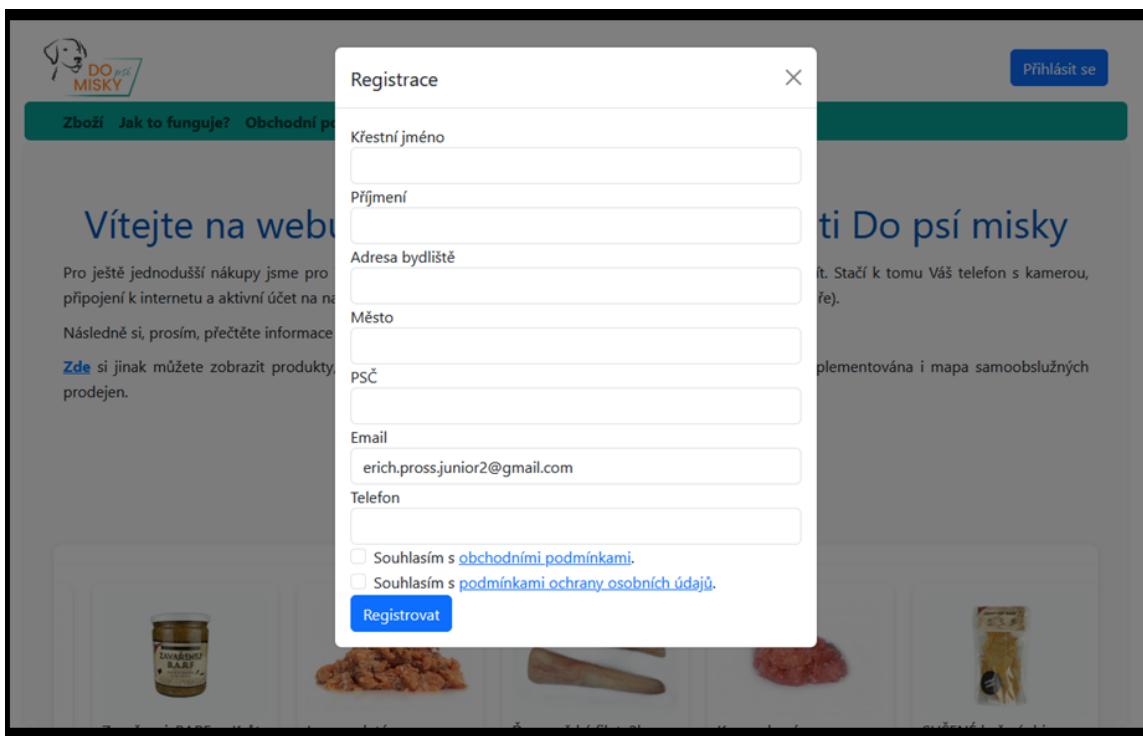
Při výběru přihlášení skrze Google, či Facebook, je uživatel, rozdílně od předešlé kapitoly, přesměrován na přihlašovací bránu dané třetí strany (od teď' jen provider).

Po vybrání účtu je na server vrácen mail uživatele a pomocí funkce `handleProviderCallback` je rozhodnuto, zda již je uživatelský mail uložen v databázi (viz Výpis kódu 6). Pokud ano, a má nastavené id účtu pro Google, či Facebook, je přihlášen. Pokud ano, ale není definováno id u využitého providera, je uživateli dané id přidáno do databáze a je následně přihlášen. Pokud mail není v databázi veden, je uživatel navrácen na indexovou stránku společně s otevřeným modálním oknem pro vyplnění zbývajících údajů (viz Obrázek 38).

Výpis kódu 6 Metoda handleProviderCallback v loginController

```
public function handleProviderCallback($provider)
{
    $socialUser = Socialite::driver($provider)->user();
    $user = User::where('email', $socialUser->getEmail())->first();

    if ($user) {
        if ($provider == 'google') {
            $user->google_id = $socialUser->getId();
        } elseif ($provider == 'facebook') {
            $user->facebook_id = $socialUser->getId();
        }
        $user->save();
        Auth::login($user, true);
        return redirect($this->redirectToPath());
    } else {
        return redirect('/?showRegister=true')->with([
            'email' => $socialUser->getEmail(),
            'name' => $socialUser->getName(),
            $provider . '_id' => $socialUser->getId(),
            'provider' => $provider,
        ]);
    }
}
```



Obrázek 38 Registrační formulář při využití providera

Po odeslání formuláře jsou uživatelská uložena v databázi (viz Výpis kódu 7), následně je uživateli přidána funkce „Uživatel“ a je přihlášen. Také je vygenerován \$activationToken, který je uložen do tabulky activation\_tokens a je zaslán uživateli na zadáný mail pomocí notifikace userActivateNotification ověřovací mail (viz Výpis kódu 8, Obrázek 39).

#### Výpis kódu 7 Funkce create v registerController

```
protected function create(array $data)
{
    $user = User::create([
        'first_name' => $data['first_name'],
        'last_name' => $data['last_name'],
        'address' => $data['address'],
        'city' => $data['city'],
        'zip' => $data['zip'],
        'phone' => $data['phone'],
        'status' => 'unactivated',
        'email' => $data['email'],
        'password' => isset($data['password']) ? bcrypt($data['password']) : null,
        'google_id' => $data['google_id'],
        'facebook_id' => $data['facebook_id'],
    ]);
}
```

```

$user->roles()->attach(1);
$activationToken = ActivationToken::create([
    'user_id' => $user->id,
    'token' => Str::random(60),
    'used' => false
]);
$user->notify(new UserActivateNotification($activationToken->token));
return $user;
}

```

Výpis kódu 8 Notifikace userActivateNotification

```

class UserActivateNotification extends Notification implements ShouldQueue
{
    use Queueable, SerializesModels;

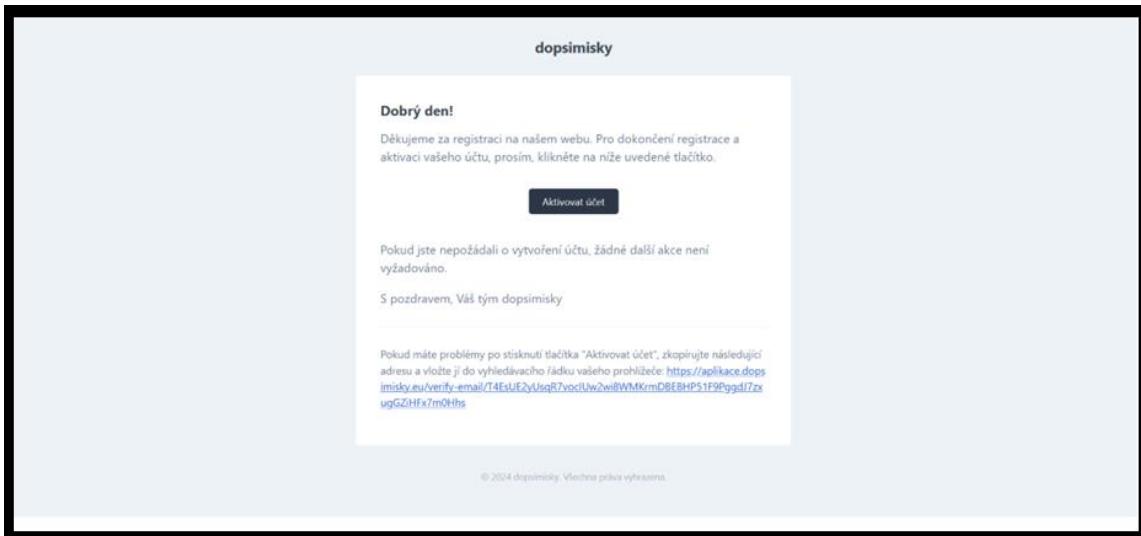
    public $activationToken;

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

    public function via($notifiable)
    {
        return ['mail'];
    }

    public function toMail($notifiable)
    {
        $url = url('/verify-email/' . $this->activationToken);
        return (new MailMessage)
            ->greeting('Dobrý den!')
            ->subject('Aktivace účtu')
            ->line('Děkujeme za registraci na našem webu. Pro dokončení registrace a aktivaci vašeho účtu, prosím, klikněte na níže uvedené tlačítko.')
            ->action('Aktivovat účet', $url)
            ->line('Pokud jste nepožádali o vytvoření účtu, žádné další akce není vyžadováno.')
            ->salutation('S pozdravem, Váš tým dopsimisky');
    }
}

```



Obrázek 39 Vzhled potvrzovacího mailu

Po kliknutí na ověřovací odkaz je zkontrolován ověřovací token vůči databázi a pokud je shodný, uživatelský účet je aktivován přepsáním hodnoty v sloupci `status` na `activated` a aktivační token je označen jako použitý (viz Výpis kódu 9). Uživatel je následně přesměrován na úvodní stránku s potvrzením o aktivaci. Pokud je kód již použit, je uživateli zobrazena domovská stránka s chybou.

#### Výpis kódu 9 Funkce verify v CustomVerificationController

```
public function verify($token)
{
    $activationToken = ActivationToken::where('token', $token)->where('used', false)-
>first();
    if (!$activationToken) {
        return redirect('/')->with('error', 'Neplatný aktivační odkaz nebo již byl
použit.');
    }
    $user = $activationToken->user;
    if ($user && $user->status !== 'activated') {
        $user->status = 'activated';
        $user->save();
        $activationToken->used = true;
        $activationToken->save();
        Auth::login($user);
        return redirect('/?activated=true')->with('success', 'Váš účet byl úspěšně
aktivován.');
    }
}
```

## 4.6 Správa prodejen

Správa prodejen je prováděna skrze view `manageStores` (viz Výpis kódu 10, Obrázek 40), který přebírá informace o prodejnách z kontroleru `storeController` pomocí funkce `index` (viz Výpis kódu 11).

Výpis kódu 10 View `manageStores`

```
<div class="container mt-4 stores">
    <h1>Správa Prodejen</h1>
    <div class="actions d-flex justify-content-between align-items-center">
        <div class="actionButtons ml-3 justify-content-between">
            <button type="button" class="btn btn-success" data-bs-toggle="modal" data-bs-target="#addStoreModal">
                Přidat Prodejnu
            </button>
        </div>
        <div class="form-check">
            <input class="form-check-input" type="checkbox" value="" id="showInactiveStoresCheckbox" checked>
            <label class="form-check-label" for="showInactiveStoresCheckbox">Zobrazit neaktivní prodejny</label>
        </div>
    </div>
    <table class="table table-striped mt-4 full-width-table">
        <thead>
            <tr>
                <th>Název</th>
                <th>Popis</th>
                <th>Adresa</th>
                <th>Akce</th>
                <th>Stav</th>
            </tr>
        </thead>
        <tbody>
            @foreach ($stores as $store)
            <tr class="store-row" data-active="{{ $store->is_enabled ? 'true' : 'false' }}>
                <td>
                    <a href="{{ route('stores.showProducts', $store->id) }}">
                        <strong>{{ $store->name }}</strong></a>
                </td>
                <td>{{ $store->description }}</td>
                <td>{{ $store->address }}</td>
                <td>
                    <button type="button" class="btn btn-sm btn-info" data-bs-toggle="modal" data-bs-target="#editStoreModal{{ $store->id }}>
                        Upravit
                    </button>
                </td>
            </tr>
        </tbody>
    </table>
</div>
```

```

        data-bs-target="#viewQrModal" data-store-id="{{ $store->id
}}">Zobrazit/změnit QR</button>
    <button type="button" class="btn btn-sm btn-primary" data-bs-toggle="modal"
        data-bs-target="#changeAccessCodeModal" data-store-id="{{ $store->id
}}">
        Změnit přístupový kód
    </button>
</td>
<td>
    <button type="button" class="btn btn-sm btn-success btn-enable" data-id="{{
$store->id }}>
        {{ $store->is_enabled ? 'disabled' : '' }}>Povolit</button>
    <button type="button" class="btn btn-sm btn-danger btn-disable" data-id="{{
$store->id }}>
        {{ !$store->is_enabled ? 'disabled' : '' }}>Zakázat</button>
    </td>
</tr>
@endforeach
</tbody>
</table>
</div>
Výpis kódu 11 Funkce index v kontroleru storeController

```

```

public function index()
{
    $stores = Store::all();
    return view('manageStores', compact('stores'));
}

```

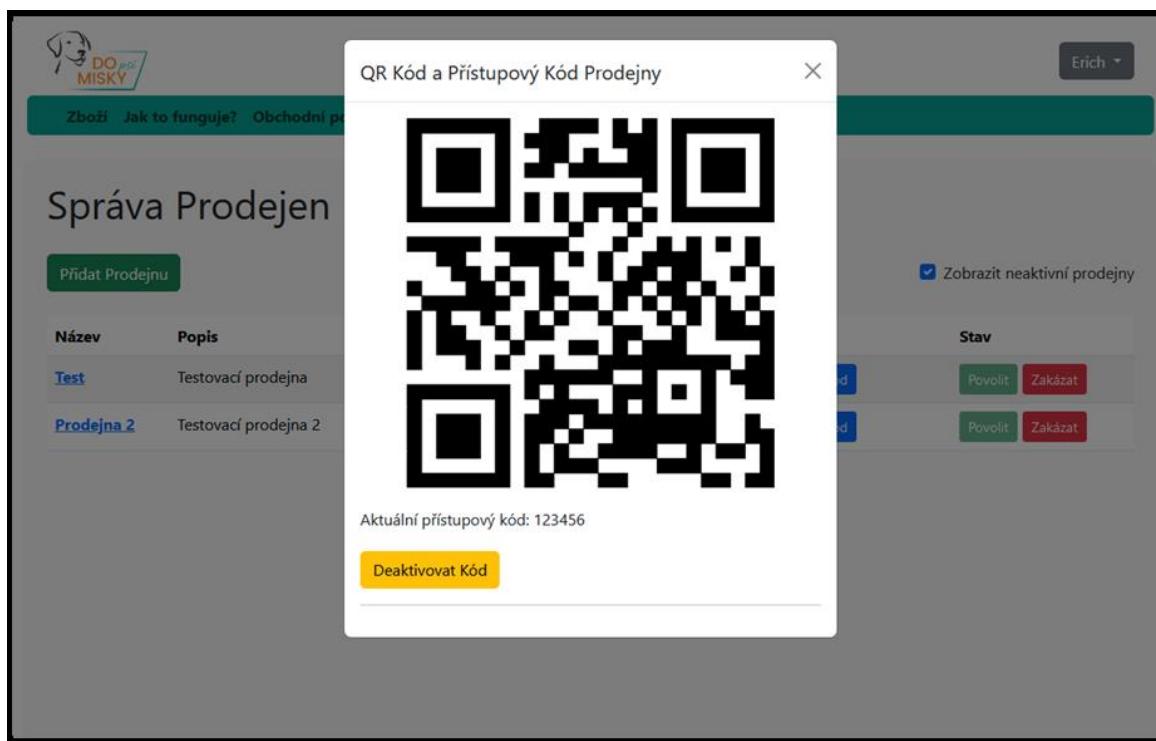
Název	Popis	Adresa	Akce	Stav
<a href="#">Test</a>	Testovací prodejna	Test 123	Zobrazit/změnit QR Změnit přístupový kód	Povolit Zakázat
<a href="#">Prodejna 2</a>	Testovací prodejna 2	Plzeňská 426 Stod	Zobrazit/změnit QR Změnit přístupový kód	Povolit Zakázat

Obrázek 40 Vyobrazení view manageStores

Uživatel z view `manageStores` může provést změnu QR kódu pro přístup do prodejny, změnu pouze přístupového kódu ode dveří prodejny, změnit stav prodejny, nebo spravovat dostupné produkty na prodejně.

V případě vybrání možnosti pro zobrazení, či změny QR kódu, je uživateli zobrazeno modální okno (viz Obrázek 41), který zobrazuje současný QR kód včetně současně nastaveného přístupového kódu a tlačítko pro deaktivaci kódu v případě, že prodejna má daný kód nastavena.

Pokud QR kód nastaven není, je zobrazeno modální okno pro vygenerování nového QR kódu. Po vyplnění nového přístupového kódu od prodejny je vygenerován náhodný řetězec o délce 30 znaků, ze kterého je následně vytvořen QR kód, který je uložen na server a jeho cesta, společně s jeho obsahem a novým přístupovým kódem od zámku prodejny je vložen do databáze (viz Výpis kódu 12).



Obrázek 41 Dialogové okno s QR kódem a současným přístupovým kódem prodejny

Výpis kódu 12 Funkce `createNewCode` pro generování a ukládání nových QR kódů

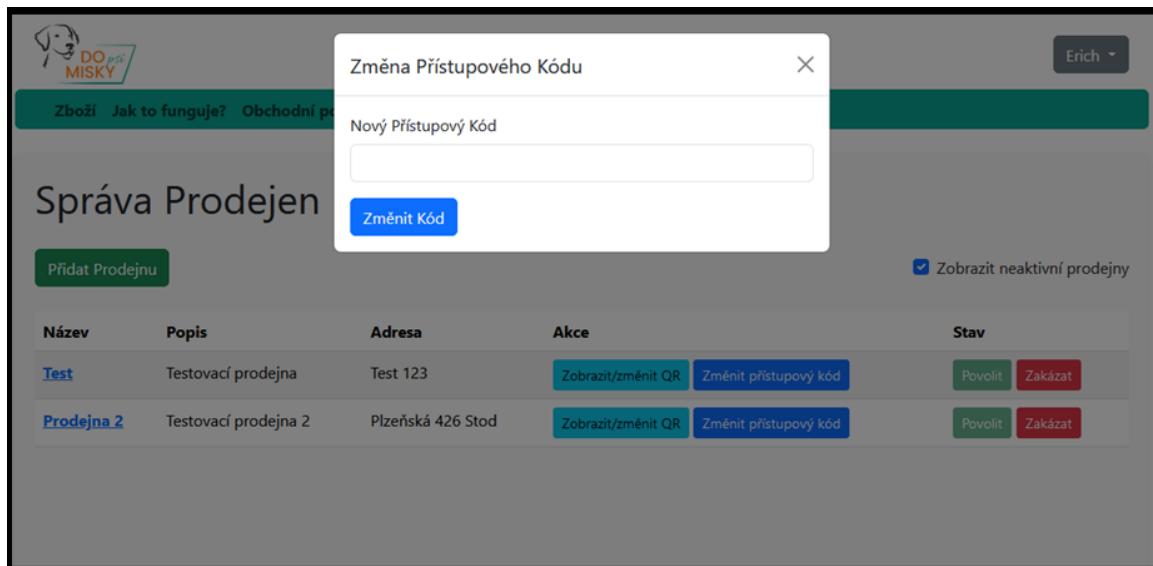
```
public function createNewCode(Request $request, $storeId)
{
    $request->validate([
        'access_code' => 'required|string',
    ]);
    $qrContent = Str::random(30);
    $qrCodeName = 'qr-codes/' . uniqid() . '.png';
```

```

        QrCode::format('png')->size(600)->generate($qrContent, storage_path('app/public/' .
$qrCodeName));
        AccessCode::where('store_id', $storeId)->update(['active' => false]);
        $accessCode = AccessCode::create([
            'store_id' => $storeId,
            'user_id' => auth()->id(),
            'code' => $request->access_code,
            'qr_content' => $qrContent,
            'qr_path' => 'storage/' . $qrCodeName,
            'active' => true,
        ]);
        return response()->json(['success' => 'Nový QR kód byl úspěšně vytvořen a uložen.', 'accessCode' => $accessCode]);
    }
}

```

Po vybrání možnosti pro změnu hesla od zámku prodejny je zobrazeno jednoduché okno, do kterého je potřeba zadat nový kód, který je po odeslání požadavku přepsán v databázi u daného přístupového kódu (viz Obrázek 42). Tato možnost byla přidána po domluvě se zadavatelem jako doplňující funkce.



Obrázek 42 Dialog pro změnu hesla od zámku prodejny

Pokud si uživatel rozklikne název prodejny, je mu zobrazen výpis všech položek, které jsou na skladě dané prodejny a může dané zboží spravovat. Má možnost přidat zboží, upravit počet dostupného zboží na prodejně, aktivovat sledování zboží, či zboží odebrat (viz Obrázek 43). Údaje o prodejně jsou předávány skrze funkci `showProducts` v `storeController` (viz Výpis kódu 13), která následně zobrazí informace o dostupných produktech ve view `store`.

Obrázek 43 Zobrazení dostupných produktů na prodejně

Uživatel má možnost zapnout sledování zboží pomocí funkce `toggleTracking` v `storeController` (viz Výpis kódu 14). Pokud uživatel použije tlačítko pro zapnutí sledování, je mu zobrazeno pole pro zadání sledovaného množství (viz Obrázek 44). Po potvrzení je v pivot tabulce `store_products` nastavena hodnota `keep_track` na 1 a následně nastavena hodnota `minimum_quantity_alert` na požadované množství.

#### Výpis kódu 13 View store

```
<div class="container mt-4 store">
    <h1>Produkty pro prodejnu: {{ $store->name }}</h1>
    <div class="actions d-flex justify-content-between align-items-center">
        <div class="actionButtons ml-3 justify-content-between">
            <button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#addProductsModal">Přidat
                Zboží</button>
            </div>
        </div>
    @if ($store->products->isEmpty())
        <p>Žádné produkty nejsou přiřazeny k tomuto obchodu.</p>
    @else
        <table class="table table-striped mt-4 full-width-table">
            <thead>
                <tr>
                    <th>Produkt</th>
                    <th>Množství</th>
                    <th>Sledovaný počet zboží</th>
```

```

<th>Změna množství</th>
<th>Hlídání zboží</th>
<th>Akce</th>
</tr>
</thead>
<tbody>
@foreach ($store->products as $product)
<tr>
<td>{{ $product->name }}</td>
<td>{{ $product->pivot->quantity }}</td>
<td>{{ $product->pivot->minimum_quantity_alert ?: '---' }}</td>
<td>
    <!-- Tlačítko pro zobrazení formuláře -->
    <button class="btn btn-sm btn-primary open-update-form"
        data-product-id="{{ $product->id }}">Změnit množství</button>
    <!-- Skrytý formulář pro aktualizaci množství -->
    <div class="update-quantity-form d-none" data-product-id="{{ $product->id
}}">
        <form
            action="{{ route('store.updateProductQuantity', ['storeId' => $store-
>id, 'productId' => $product->id]) }}"
            method="POST">
            @csrf
            <input type="number" name="newQuantity" min="0"
                value="{{ $product->pivot->quantity }}" required>
            <button type="submit" class="btn btn-sm btn-success">OK</button>
            <button type="button"
                class="btn btn-sm btn-secondary cancel-update">Zrušit</button>
        </form>
    </div>
</td>
<td>
    @if ($product->pivot->keep_track)
        <button class="btn btn-warning btn-sm toggle-tracking"
            data-product-id="{{ $product->id }}" data-
action="disable">Vypnout</button>
    @else
        <button class="btn btn-info btn-sm toggle-tracking"
            data-product-id="{{ $product->id }}" data-
action="enable">Zapnout</button>
    @endif
    <div class="tracking-form" data-product-id="{{ $product->id }}"
        style="display: none;">
        <form

```

```

        action="{{ route('store.toggleTracking', ['storeId' => $store->id,
'productId' => $product->id]) }}"
            method="POST" style="display: flex; gap: 10px;">
            @csrf
            <input type="number" name="minimum_quantity_alert" placeholder="Min.
množství"
                min="1">
            <button type="submit" class="btn btn-primary btn-sm">OK</button>
            <button type="button"
                class="btn btn-secondary btn-sm cancel-tracking">Zrušit</button>
        </form>
    </div>
</td>
<td>
    <button class="btn btn-danger btn-sm remove-product"
        data-product-id="{{ $product->id }}>Odebrat</button>
</td>
</tr>
@endforeach
</tbody>
</table>
@endif
<hr>
</div>

```

Výpis kódu 14 Funkce toggleTracking v storeController

```

public function toggleTracking(Request $request, $storeId, $productId)
{
    $store = Store::findOrFail($storeId);
    $enableTracking = $request->input('enable_tracking', false);
    $minimumQuantityAlert = $enableTracking ? $request->input('minimum_quantity_alert', 0)
: 0;
    $store->products()->updateExistingPivot($productId, [
        'keep_track' => $enableTracking,
        'minimum_quantity_alert' => $minimumQuantityAlert,
    ]);
    return response()->json(['success' => 'Sledování produktu bylo aktualizováno.']);
}

```

The screenshot shows a table of products:

Produkt	Množství	Sledovaný počet zboží	Změna množství	Hlídání zboží	Akce
Zavařené BARF - Krůta s třeskou, rýží a zeleninou	666	25	Změnit množství	Vypnout	Odebrat
SUŠENÉ krůtí pařáty 3 ks	48	---	Změnit množství	Min. množství	OK Zrušit Odebrat

Obrázek 44 View store s možností pro zadání sledovaného zboží

Pokud uživatel vybere možnost pro přidání zboží na sklad zvolené prodejny, je zobrazeno modální okno s výpisem zboží (viz Obrázek 45) z tabulky products, které se současně ještě nenachází na skladu (není možné stejně zboží přidat vícekrát). Zboží je přidáno zaškrtnutím checkboxu u zvolené položky, zadáním množství a potvrzením pomocí tlačítka „Přidat zboží“. Stránka je následně obnovena s nově přidanými produkty.

The modal window contains a table:

Vybrat	Produkt	Množství
<input checked="" type="checkbox"/>	Losos mletý	25
<input type="checkbox"/>	Ďas mořský filet, 2ks	
<input checked="" type="checkbox"/>	Kapr sekaný	14
<input type="checkbox"/>	SUŠENÉ kuřecí chipsy	
<input type="checkbox"/>	SUŠENÉ krůtí hlavy 3 ks	
<input type="checkbox"/>	test	
<input type="checkbox"/>	Test 2	
<input type="checkbox"/>	Test 3	
<input type="checkbox"/>	Test 4	

Buttons at the bottom: Zavřít (Close), Přidat Zboží (Add Product).

Obrázek 45 Modální okno pro přidání zboží na sklad prodejny

## 4.7 Správa zboží

Správa zboží je implementována jako view manageProducts, které je velice podobné zobrazení prodejen. Všechno zboží je vypisováno jako tabulka společně se všemi důležitými informacemi o zboží jako je obrázek (nebo obrázky), název, čárový kód, popis, balení, cena, cena po slevě, zobrazení a nastavení slev (viz Obrázek 46).

Obrázek	Název	Kód	Popis	Balení	Cena	Sleva	Sleva od	Sleva do	CPS	Akce	Stav
	Zavářené BARF - Krůta s treskou, rýží a zeleninou	2564325	Krůtí maso, stejně tak jako tr...	375 g	109,00 Kč	-	-	-	109,00 Kč	<button>Upravit</button> <button>Sleva</button> <button>Povol</button> <button>Zákázat</button>	
	Losoš mletý	2564326	Losoš - stejně jako jiné ryby...	cca 1kg	119,00 Kč	-	-	-	119,00 Kč	<button>Upravit</button> <button>Sleva</button> <button>Povol</button> <button>Zákázat</button>	
	Ósas mořský filet_2ks	1549876	Ósas mořský - stejně jako jiné...	2ks; cca 400g - 700g	99,00 Kč	-	-	-	99,00 Kč	<button>Upravit</button> <button>Sleva</button> <button>Povol</button> <button>Zákázat</button>	
	Kapr sekanyj	2658754	Kapr - stejně jako jiné ryby -...	cca 1 kg	79,00 Kč	-	-	-	79,00 Kč	<button>Upravit</button> <button>Sleva</button> <button>Povol</button> <button>Zákázat</button>	
	SUŠENÉ kufeci chipsy	9865354	Hledáte zdravou a chutnou odmě...	100 g	69,00 Kč	-	-	-	69,00 Kč	<button>Upravit</button> <button>Sleva</button> <button>Povol</button> <button>Zákázat</button>	
	SUŠENÉ krůtí palety_3 ks	6521489	Sušené krůtí pařátky jsou velmi...	3 ks	59,00 Kč	-	-	-	59,00 Kč	<button>Upravit</button> <button>Sleva</button> <button>Povol</button> <button>Zákázat</button>	
	SUŠENÉ krůtí hlavy_3 ks	75486932	Sušené krůtí hlavy jsou velmi...	3 ks	59,00 Kč	-	-	-	59,00 Kč	<button>Upravit</button> <button>Sleva</button> <button>Povol</button> <button>Zákázat</button>	
	test	568547521	test	test	250,00 Kč	-	-	-	250,00 Kč	<button>Upravit</button> <button>Sleva</button> <button>Povol</button> <button>Zákázat</button>	

Obrázek 46 View manageProducts

Informace o produktech jsou získané z tabulky products pomocí funkce index v ProductController (viz Výpis kódu 15). Tato funkce vrací informace o všech produktech, tedy i o těch, které jsou případně označeny za neaktivní, což je dále možné filtrovat přímo na stránce.

Je nutné zmínit, že obrázky produktů nejsou navráceny pomocí této funkce, místo toho je využívána vazba na tabulku photos pomocí pivot tabulky product\_photos (viz Výpis kódu 16) přímo v šabloně stránky (viz Výpis kódu 17). Toto řešení s volitelným množstvím obrázků bylo implementované po konzultaci se zadavatelem, přičemž byl vzesen požadavek o možnost přidat k danému zboží více obrázků.

Výpis kódu 15 Funkce index v ProductController

```
public function index()
{
    $products = Product::all();
    return view('manageProducts', compact('products'));
}
```

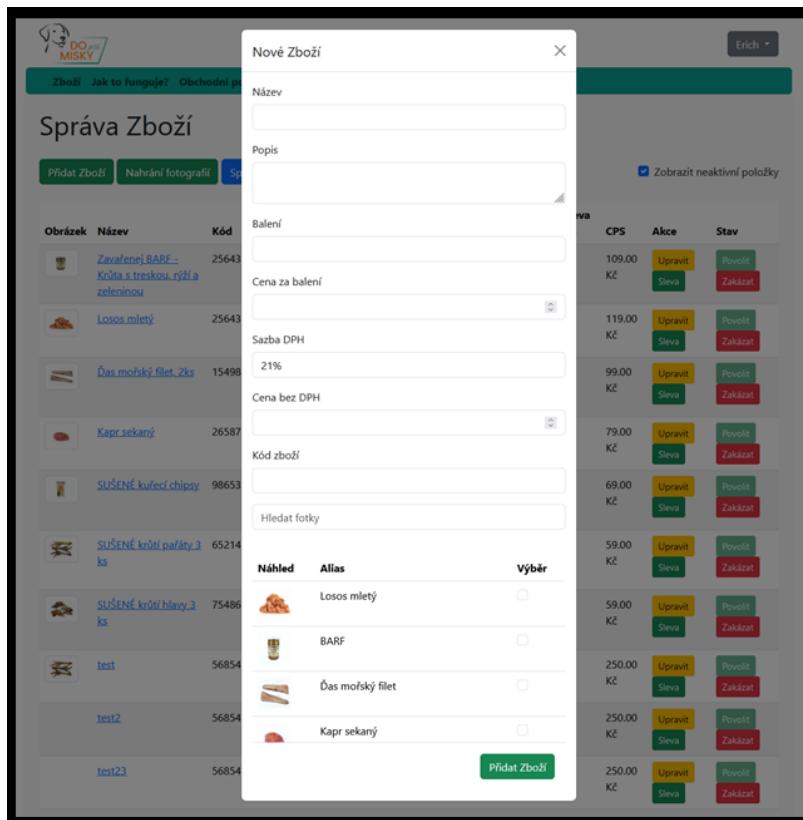
Výpis kódu 16 Funkce photos v modelu Products

```
public function photos()
{
    return $this->belongsToMany(Photo::class, 'product_photo');
```

## Výpis kódu 17 Výpis zboží z view manageProducts

```
@foreach ($products as $product)
<tr class="product-row" data-active="{{ $product->is_enabled ? 'true' : 'false' }}>
    <td>
        @foreach ($product->photos as $photo)
            
        @endforeach
    </td>
    <td><a href="{{ route('products.show', $product) }}>{{ $product->name }}</a></td>
    <td>{{ $product->barcode }}</td>
    ...
</tr>
@endforeach
```

Pro přidání zboží je použito modální okno (viz Obrázek 47), které je potřeba vyplnit všemi informacemi o produktu, kromě obrázků, které lze přidat dodatečně. Je využita funkce `store` v `productController` (viz Výpis kódu 18), která zadané informace zpracuje a vytvoří produkt v databázi. Pokud jsou důležité informace o produktu v databázi již evidovány (kód produktu a jeho název), je na to uživatel upozorněn a přidání zboží neproběhne.



Obrázek 47 Modální okno pro zobrazení zboží

### Výpis kódu 18 Funkce store v productController

```
public function store(Request $request)
{
    $validatedData = $request->validate([
        'name' => 'required|string|unique:products,name|max:255',
        'description' => 'required|string',
        'unit' => 'required|string|max:255',
        'price' => 'required|numeric',
        'tax_rate' => 'required|string',
        'discount' => 'nullable|numeric',
        'photo_ids' => 'nullable|array',
        'photo_ids.*' => 'exists:photos,id',
        'barcode' => 'required|string|unique:products,barcode',
        'price_excl_tax' => 'required',
    ]);
    $product = Product::create([
        'name' => $validatedData['name'],
        'description' => $validatedData['description'],
        'unit' => $validatedData['unit'],
        'price' => $validatedData['price'],
        'tax_rate' => $validatedData['tax_rate'],
        'price_excl_tax' => $validatedData['price_excl_tax'],
        'discount' => $validatedData['discount'],
        'barcode' => $validatedData['barcode']
    ]);

    if ($request->has('photo_ids')) {
        $product->photos()->attach($request->input('photo_ids'));
    }
    if ($product) {
        return response()->json(['message' => 'Zboží bylo úspěšně přidáno.'], 200);
    } else {
        return response()->json(['message' => 'Nepodařilo se přidat zboží.'], 500);
    }
}
```

Nahrání fotografií je provedeno skrze modální okno, do kterého uživatel může vložit libovolné množství fotografií. Ty jsou následně nahrány na server a do databáze je uložen jejich název a cesta (viz Výpis kódu 19). Pokud se mezi nahrávaným obrázky nachází takový, který má stejný název jako obrázek již nahraný na serveru, je přeskočen a nahrávání pokračuje. O této skutečnosti je uživatel obeznámen v okně, které je zobrazeno jako potvrzení procesu nahrávání.

### Výpis kódu 19 Funkce upload v PhotoController

```
public function upload(Request $request)
{
    $request->validate([
        'photos.*' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
    ]);

    $uploadedPhotos = [];
    $skippedPhotos = [];

    if ($request->hasfile('photos')) {
        foreach ($request->file('photos') as $file) {
            $existingPhoto = Photo::where('alias', $file->getClientOriginalName())-
>first();
            if ($existingPhoto) {
                $skippedPhotos[] = $file->getClientOriginalName();
                continue;
            }
            $imageName = time() . '_' . uniqid() . '.' . $file-
>getClientOriginalExtension();
            $path = $file->storeAs('public/products', $imageName); // Uložení souboru
            $url = Storage::url($path);
            $photo = Photo::create([
                'name' => $imageName,
                'alias' => $file->getClientOriginalName(),
                'path' => $url,
                'uploaded_by' => Auth::id(),
            ]);

            $uploadedPhotos[] = $photo;
        }
    }

    return response()->json([
        'message' => 'Proces nahrávání dokončen.',
        'uploaded' => $uploadedPhotos,
        'skipped' => $skippedPhotos,
    ]);
}
```

Úpravy zboží probíhají skrze vizuálně stejné modální okno, jako při přidání nového zboží, přičemž údaje jsou již vyplněny z databáze, funkce uložení změn však není stejná jako pro přidání zboží, je využívaná funkce update v ProductController (viz Výpis kódu 20). V modálním okně uživatel může změnit jakýkoliv z dostupných údajů. Při uložení změn,

v případě totožného názvu, nebo čárového kódu s jiným zboží, je uživateli zobrazena chyba, jinak je zboží aktualizováno. O odeslání a zpracování dat z formuláře se stará JavaScript, který zobrazuje uživateli modální okno s potřebnými informacemi dle výsledku požadavku (viz Výpis kódů 21).

#### Výpis kódů 20 Funkce update v ProductController

```
public function update(Request $request)
{
    $product = Product::find($request->id);
    $request->validate([
        'id' => 'required|exists:products,id',
        'name' => 'required|string|max:255',
        'description' => 'required|string',
        'unit' => 'required|string|max:255',
        'price' => 'required|numeric',
        'discount' => 'nullable|numeric',
        'photo_ids' => 'nullable|array',
        'photo_ids.*' => 'exists:photos,id',
        'barcode' => 'required|string|unique:products,barcode,' . $product->id,
        'pending_discount' => 'nullable|numeric',
        'price_excl_tax' => 'required',
        'tax_rate' => 'required|string',

    ]);
    if ($request->has('photo_ids')) {
        $product->photos()->sync($request->photo_ids);
    } else {
        $product->photos()->detach();
    }
    $product->update([
        'name' => $request->name,
        'description' => $request->description,
        'unit' => $request->unit,
        'price' => $request->price,
        'discount' => $request->discount,
        'barcode' => $request->barcode,
        'price_excl_tax' => $request->price_excl_tax,
        'tax_rate' => $request->tax_rate,
    ]);

    return response()->json(['message' => 'Zboží bylo úspěšně aktualizováno.']);
}
```

#### Výpis kódů 21 JavaScript pro zpracování dat z formuláře a následné odeslání na server

```

$('#editProductForm').on('submit', function (e) {
    e.preventDefault();

    let formData = new FormData(this);

    $('input[name="photo_ids[]":checked]').each(function () {
        formData.append('photo_ids[]', $(this).val());
    });

    $.ajax({
        url: '/products/update',
        type: 'POST',
        data: formData,
        contentType: false,
        processData: false,
        success: function (response) {
            $('#editProductModal').modal('hide');
            showUniversalModal('Úspěch', 'Zboží bylo úspěšně upraveno.', true);
            document.getElementById("editProductForm").reset();
        },
        error: function (xhr, status, errors) {
            let errorMessage = 'Neznámá chyba';

            try {
                const responseJSON = JSON.parse(xhr.responseText);
                if (responseJSON.message && responseJSON.error) {
                    errorMessage = responseJSON.message + ': ' + responseJSON.error;
                } else if (responseJSON.message) {
                    errorMessage = responseJSON.message;
                } else {
                    errorMessage = xhr.responseText;
                }
            } catch (e) {
                errorMessage = xhr.responseText;
            }
            $('#editProductModal').modal('hide');

            showUniversalModal('Chyba', 'Došlo k chybě, zboží nebylo upraveno. \n
Chyba: ' + errorMessage + '\n Status: ' + status, false);
        }
    });
});

```

## 4.8 Správa uživatelů

Správa uživatelů je implementována jako view `users`, který dostává informace o všech uživatelích pomocí funkce `index` v `UserController` (viz Výpis kódu 22). Je zobrazeno jméno a příjmení uživatele, jeho role, stav účtu a je zde přidána možnost pro přidání rolí k danému uživateli, či jeho účet aktivovat, nebo deaktivovat (viz Výpis kódu 23, Obrázek 48). Uživateli je možné filtrovat dle jména, role a stavu účtu.

Role je možné volně přidávat a odebírat, pokud má uživatel přístup k roli A: Uživatelé. Samotná úprava rolí probíhá skrze modální okno (viz Obrázek 49), kde stačí zaškrtnout danou roli a potvrdit výběr. Změny jsou ihned uloženy do pivot tabulky `role_user`.

Výpis kódu 22 Funkce index v UserController

```
public function index(Request $request)
{
    $search = $request->input('search', '');
    $users = User::with('roles')
        ->where('first_name', 'like', '%' . $search . '%')
        ->orWhere('email', 'like', '%' . $search . '%')
        ->paginate(10);

    $roles = Role::all();

    return view('fullViews/adminUsers/users', compact('users', 'search', 'roles'));
}
```

Výpis kódu 23 View users

```
<div class="container overflow-y: auto; user">
    <h1>Uživatelé</h1>
    <div class="filters mb-3 d-flex align-items-end">
        <input type="text" id="searchFilter" class="form-control" placeholder="Vyhledat
uživatele...">

        <select id="roleFilter" class="form-control">
            <option value="">Filtrovat dle role</option>
            @foreach ($roles as $role)
                <option value="{{ $role->id }}>{{ $role->name }}</option>
            @endforeach
        </select>

        <select id="statusFilter" class="form-control">
            <option value="">Filtrovat dle stavu</option>
            <option value="activated">Aktivní</option>
            <option value="unactivated">Neaktivní</option>
        </select>
    </div>
</div>
```

```

        <option value="stopped">Pozastaven</option>
    </select>
</div>
<table class="table" id="usersTable">
    <thead>
        <tr>
            <th>ID</th>
            <th>Jméno</th>
            <th>Email</th>
            <th>Stav účtu</th>
            <th>Role</th>
            <th>Akce</th>
        </tr>
    </thead>
    <tbody>
        @foreach ($users as $user)
        <tr>
            <td>{{ $user->id }}</td>
            <td>{{ $user->first_name }} {{ $user->last_name }}</td>
            <td>{{ $user->email }}</td>
            <td>
                @switch($user->status)
                @case('activated')
                    Aktivní
                @break
                @case('unactivated')
                    Neaktivní
                @break
                @case('stopped')
                    Pozastaven
                @break
                @default
                    Neznámý
                @endswitch
            </td>
            <td>
                @foreach ($user->roles as $role)
                <span class="badge bg-secondary" data-toggle="tooltip" data-placement="top" title="{{ $role->description }}">
                    {{ $role->name }}
                </span>
                @endforeach
            </td>
            <td>

```

```

<form action="{{ route('admin.updateUserStatus', $user->id) }}"
method="POST">
    @csrf
    <button type="submit" class="btn btn-sm {{ $user->status ===
'activated' ? 'btn-secondary' : 'btn-success' }}">
        {{ $user->status === 'activated' ? 'Pozastavit' : 'Aktivovat' }}</button>
</form>
<button class="btn btn-info btn-sm manage-roles" data-user-id="{{
$user->id }}>Spravovat role</button>
</td>
</tr>
@endforeach
</tbody>
</table>
</div>

```

ID	Jméno	Email	Stav účtu	Role	Akce
100	Erich Pross	erich.pross.junior@gmail.com	Aktivní	Uživatel Administrátor A: Prodejen A: Uživatelů A: Zboží A: Aplikace	Pozastavit Spravovat role
101	Tomáš Milec	info@dopsimisky.eu	Aktivní	Uživatel Administrátor A: Prodejen A: Uživatelů A: Zboží A: Aplikace	Pozastavit Spravovat role
114	Tomáš Milec	Tomas_milec@seznam.cz	Aktivní	Uživatel Administrátor A: Prodejen A: Uživatelů A: Zboží A: Aplikace	Pozastavit Spravovat role
118	test test	erich.pross.junior2@gmail.com	Aktivní	Uživatel	Pozastavit Spravovat role

Obrázek 48 Zobrazení správy uživatelů

Správa rolí uživatele

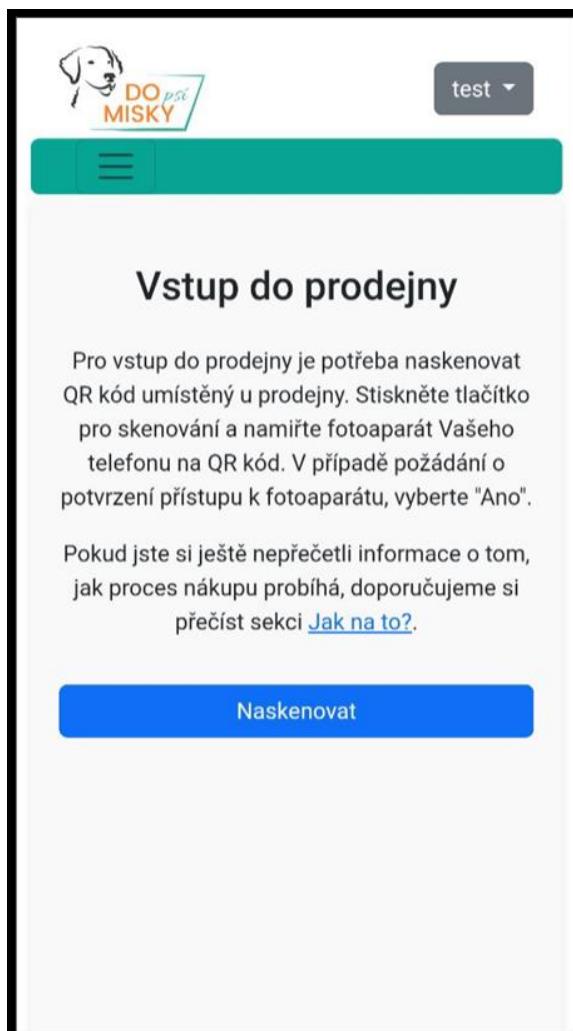
Uživatel  
 Administrátor  
 A: Prodejen  
 A: Uživatelů  
 A: Zboží  
 A: Aplikace  
 Superadmin

ID	Jméno	Email	Akce
100	Erich Pross	erich.pross.junior@gmail.com	<a href="#">Zavřít</a> <a href="#">Uložit změny</a> <a href="#">Pozastavit</a> <a href="#">Spravovat role</a>
101	Tomáš Milec	info@dopsimisky.eu	Aktivní <a href="#">Uživatel</a> <a href="#">Administrátor</a> <a href="#">A: Prodejen</a> <a href="#">A: Uživatelů</a> <a href="#">A: Zboží</a> <a href="#">A: Aplikace</a> <a href="#">Pozastavit</a> <a href="#">Spravovat role</a>
114	Tomáš Milec	Tomas_milec@seznam.cz	Aktivní <a href="#">Uživatel</a> <a href="#">Administrátor</a> <a href="#">A: Prodejen</a> <a href="#">A: Uživatelů</a> <a href="#">A: Zboží</a> <a href="#">A: Aplikace</a> <a href="#">Pozastavit</a> <a href="#">Spravovat role</a>
118	test test	erich.pross.junior2@gmail.com	Aktivní <a href="#">Uživatel</a> <a href="#">Pozastavit</a> <a href="#">Spravovat role</a>

Obrázek 49 Modální okno pro úpravu rolí uživatelů

## 4.9 Vstup do prodejny

Jak již bylo řečeno, vstup do prodejny je prováděno načtením QR kódu umístěného u vstupu do prodejny. Po kliknutí na tlačítko nákupního košíku ve view index je zobrazen view scanQR, informující uživatele o postupu skrze proces skenování (viz Obrázek 50, Výpis kódu 24). Po stisknutí tlačítka „Naskenovat“ je text s tlačítkem schován a následně zobrazena čtečka knihovny html5-qrcode (viz Výpis kódu 25), která skenuje prostředí snímané fotoaparátem uživatele. Pokud je naskenován QR kód čtečkou, je jeho obsah zaslán pomocí Javascriptu (viz Výpis kódu 26) na server, kde je jeho obsah kontrolován funkcí verifyQrCode v ScanController (viz Výpis kódu 27).



Obrázek 50 Grafické zobrazení view scanQR

Výpis kódu 24 View scanQR

```
<div class="container py-5 scanQr">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <h1 class="text-center mb-4">Vstup do prodejny</h1>
            <p class="text-center info-text">
```

Pro vstup do prodejny je potřeba naskenovat QR kód umístěný u prodejny. Stiskněte tlačítko pro skenování a namířte fotoaparát Vašeho telefonu na QR kód. V případě požádání o potvrzení přístupu k fotoaparátu, vyberte "Ano".

```

</p>
<p class="text-center info-text">
    Pokud jste si ještě nepřečetli informace o tom, jak proces nákupu probíhá, doporučujeme si přečíst sekci
        <a href="/how-it-works">Jak na to?</a>.
</p>
<div id="qr-reader" style="width:100%; height: auto;"></div>
<button class="btn btn-primary mt-3 w-100"
id="startScan">Naskenovat</button>
<div id="accessCodeDisplay"
    style="display: none; background-color: black; color: white; padding:
20px; text-align: center; position: relative;">
    <p>Přístupový kód: <span id="accessCode" style="font-size:
2em;"></span></p>
    <div id="timeBar" style="height: 5px; background-color: red; width:
0%;"></div>
    <button class="btn btn-warning mt-3" id="confirmAccess">Zahájit
nákup</button>
</div>
</div>
</div>
```

Výpis kódu 25 Funkce pro zobrazení čtečky QR kódů

```

startScanButton.addEventListener('click', function() {
    startScanButton.style.display = 'none';

    infoText.forEach(function(textElement) {
        textElement.style.display = 'none';
    });

    qrCodeReader.start({
        facingMode: "environment"
    }, {
        fps: 20,
        qrbox: 250
    }, function(qrCodeMessage) {
        qrCodeReader.stop().then(() => {
```

```

        console.log("Skenování zastaveno");
        verifyQrCode(qrCodeMessage);
    }).catch(err => {
        console.error("Chyba při zastavování skenování", err);
    });
}, function(errorMessage) {
    console.log("Chyba při skenování:", errorMessage);
}).catch(err => {
    console.error("Nelze spustit skenování", err);
});
});

```

Výpis kódu 26 Odeslání obsahu QR kódu na server pro kontrolu

```

function verifyQrCode(qrData) {
    fetch('/verify-qr-code', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-
token"]').getAttribute(
                'content'),
        },
        body: JSON.stringify({
            qrContent: qrData
        })
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            displayAccessCode(data.accessCode);
            sessionStorage.setItem('store_id', data.store_id);
        } else {
            alert(data.message);
        }
    })
    .catch(error => console.error('Error:', error));
}

```

Výpis kódu 27 Kontrola QR kódu na serveru, funkce verifyQrCode v ScanController

```

public function verifyQrCode(Request $request)
{
    $qrContent = $request->input('qrContent');

```

```

        $accessCode = AccessCode::where('qr_content', $qrContent)->where('active', true)-
>first();

        if ($accessCode) {
            $this->logActivity('scan_qr', [
                'qr_code' => $accessCode->id ,
                'store_id' => $accessCode->store_id,
                'access_code' => $accessCode->code
            ]);

            return response()->json([
                'success' => true,
                'message' => 'QR kód je platný.',
                'accessCode' => $accessCode->code,
                'store_id' => $accessCode->store_id
            ]);
        }

        return response()->json(['success' => false, 'message' => 'Neplatný QR kód.']);
    }
}

```

Pokud je obsah kódu nalezen v tabulce `access_code`, a kód je označen jako aktivní, je uživateli zobrazen přístupový kód prodejny pomocí funkce `displayAccessCode` (viz Výpis kódu 28), který zobrazí daný kód v prvku `accessCodeDisplay` na stránce se čtečkou a zahájí odpočet pro zakrytí kódu (viz Obrázek 51).

V průběhu odpočtu může uživatel potvrdit vstup do prodejny a přejít k dalšímu kroku nákupu. V případě, že uživatel nechá časovač vyprchat, je kód od zámku prodejny skryt a je zobrazeno modální okno pro potvrzení vstupu (viz Obrázek 52). V případě, že uživatel nevstoupil do prodejny, je možné proces načtení QR kódu provést znova, nebo při problémech s aplikací kontaktovat podporu (viz Obrázek 53).

#### Výpis kódu 28 Funkce `displayAccessCode`

```

function displayAccessCode(code) {
    const accessCodeElement = document.getElementById('accessCode');
    const accessCodeDisplay = document.getElementById('accessCodeDisplay');
    const timeBar = document.getElementById('timeBar');
    const confirmButton = document.getElementById('confirmAccess');
    const denyButton = document.getElementById('denyAccess');
    const totalTime = 10000;
    const scanButton = document.getElementById('startScan');

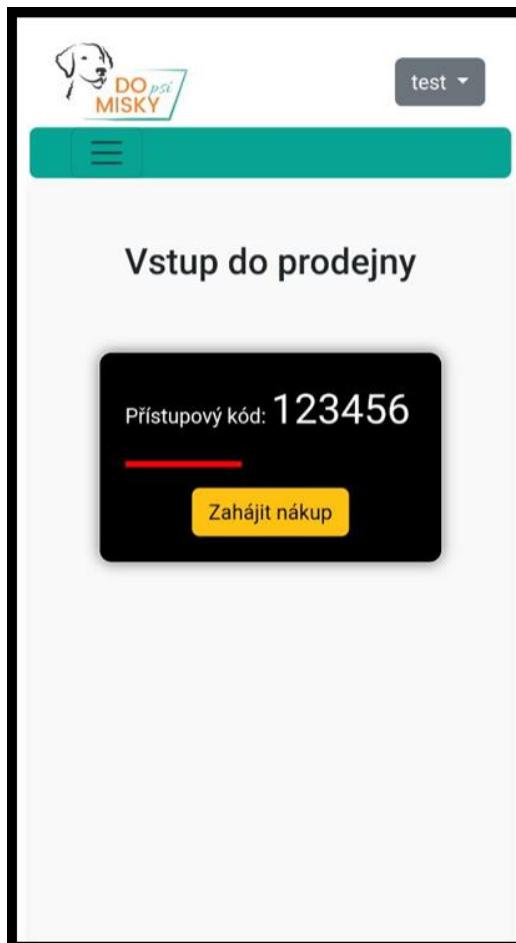
    accessCodeElement.textContent = code;
}

```

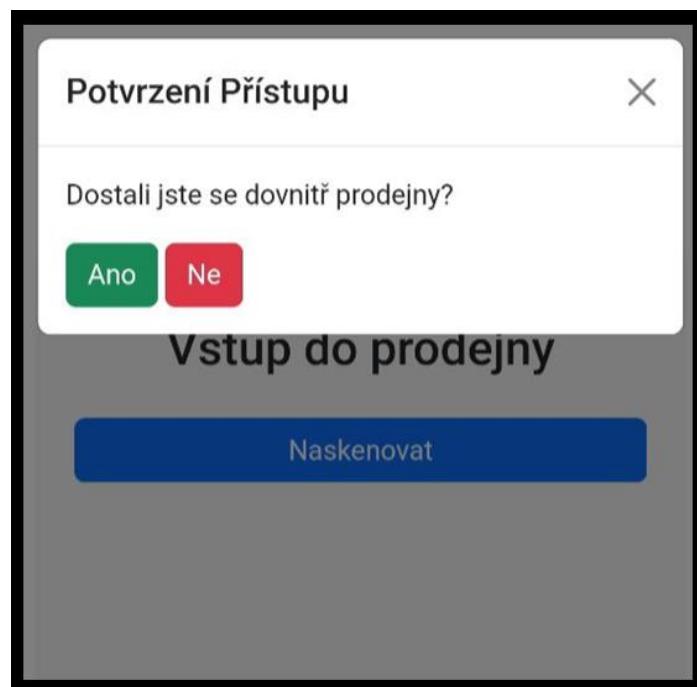
```
accessCodeDisplay.style.display = 'block';
scanButton.style.display = 'none';
let startTime = new Date();

const interval = setInterval(() => {
    const elapsedTime = new Date() - startTime;
    const width = Math.min(100, (elapsedTime / totalTime) * 100);
    timeBar.style.width = width + '%';

    if (elapsedTime >= totalTime) {
        clearInterval(interval);
        timeBar.style.width = '0%';
        accessCodeDisplay.style.display = 'none';
        new
        bootstrap.Modal(document.getElementById('accessConfirmationModal')).show();
        scanButton.style.display = 'block';
    }
}, 1000);
}
```



Obrázek 51 Zobrazení kódu pro vstup do prodejny



Obrázek 52 Modální okno pro potvrzení přístupu



Obrázek 53 Modální okno pro potvrzení přístupu po vybrání možnosti „Ne“

## 4.10 Nákup produktů

Po zahájení nákupu potvrzením vstupu do prodejny je založena nová nákupní session v tabulce `shopping_sessions`, společně s novým uživatelským košíkem v tabulce `session_carts` pomocí funkce `startSession` v `ShoppingController` (viz Výpis kódu 29). Pokud má uživatel již aktivní session ve stejném prodejním místě, jsou využity informace o košíku a ty jsou automaticky načteny a zobrazeny uživateli. Tento postup byl vybrán z důvodu, že mobilní telefony často mohou okno prohlížeče zavřít při přechodu do jiné aplikace a následně se znova pokoušet o načtení po otevření prohlížeče, což by ale přerušilo současný nákup uživatele a byla by potřeba pro vytvoření nové nákupní session, i když má uživatel zboží v košíku a chce nákup dokončit.

Výpis kódu 29 Funkce `startSession` v `ShoppingController`

```
public function startSession(Request $request)
{
    if (!Auth::check()) {
        return redirect('/login')->with('error', 'You must be logged in to start shopping.');
    }

    $store_id = $request->input('store_id');
    if (!$store_id) {
        return redirect()->back()->with('error', 'Store ID is required.');
    }

    $user_id = Auth::id();
    $session = ShoppingSession::where('user_id', $user_id)
        ->where('store_id', $store_id)
        ->where('status', 'active')
        ->first();

    if (!$session) {
        $session = new ShoppingSession([
            'user_id' => $user_id,
            'store_id' => $store_id,
            'started_at' => now(),
            'status' => 'active'
        ]);
        $session->save();
    }

    $cart = ShoppingCart::firstOrCreate(
        ['session_id' => $session->id],
        ['session_id' => $session->id]
    );
}
```

```

$cart->load('items.product');

$total = 0;
foreach ($cart->items as $item) {
    $price = $item->product->discount > 0 ? $item->product->price * (1 - $item-
>product->discount / 100) : $item->product->price;
    $total += $item->quantity * $price;
}
$totalQuantity = $cart->items->sum('quantity');

return view('/fullViews/shop/shopping', [
    'sessionId' => $session->id,
    'store_id' => $store_id,
    'cartItems' => $cart->items,
    'total' => $total,
    'totalQuantity' => $totalQuantity,
    'cart' => $cart
]);
}

```

Jako ochrana proti zneužití probíhajících nákupů je vytvořen naplánovaný úkol (viz Výpis kódu 30), který prochází tabulkou shopping\_sessions, a v případě, že datum started\_at je starší než 24 hodin, je daný nákup zrušen přepsání hodnoty active na 0 a je vloženo aktuální datum do hodnoty ended\_at. Toto řešení efektivně zamezuje zneužití nákupních session, pokud uživatel daný nákup ručně nezrušil, nebo nedokončil.

Výpis kódu 30 Naplánovaný úkol pro vymazání probíhajících nákupů starších jednoho dne

```

$schedule->call(function () {
    $inactiveSessions = ShoppingSession::where('active', true)
        ->where('started_at', '<', now()->subDay())
        ->get();
    foreach ($inactiveSessions as $session) {
        $session->update([
            'status' => 'cancelled',
            'ended_at' => now(),
            'active' => false,
        ]);
    }
})->daily()->name('cancel-old-sessions')->withoutOverlapping();
}

```

Nákup probíhá skrze view shopping, který obsahuje výpis nákupního košíku společně s potřebnými ovládacími prvky. Na stránce se nachází výpis o celkové ceně zboží a součet

počtu zboží v košíku na začátku a konci stránky pro jednoduché zobrazení těchto údajů uživateli, ať se nacházejí kdekoli na stránce, k tomu se zde nachází tlačítko pro zahájení skenování zboží a zrušení, či dokončení nákupu (viz Obrázek 54).

Informace se předávají z již zmíněné funkce `startSession` (viz Výpis kódu 29), které jsou následně vyplněny do `view shopping` (viz Výpis kódu 31).

### Výpis kódu 31 View shopping

```
<div class="total-display">
    <div class="cart-info-content">
        <i class="fas fa-shopping-cart"></i>
        <span class="cart-details">
            {{ $totalQuantity ?? 0 }} ks - celkem {{ number_format($total ?? 0, 2, ',', ',') }} Kč
        </span>
    </div>
</div>
<div class="shopping-container mt-2">
    <div class="container w-100">
        <h2>Košík</h2>
        <div id="cartItems">
            @forelse ($cartItems as $item)
                @php
                    $finalPrice = $item->product->discount && $item->product->discount > 0
                        ? $item->product->price * (1 - $item->product->discount / 100)
                        : $item->product->price;
                @endphp
                <div class="cart-item" onclick="openModal('{{ $item->id }})">
                    <span>{{ $item->product->name }} - {{ $item->quantity }} x
                        {{ number_format($finalPrice, 2, ',', ',') }} Kč </span><br>
                    <span>Celkem: {{ number_format($item->quantity * $finalPrice, 2, ',', ',') }} Kč </span>
                    <span id="itemQuantity{{ $item->id }}" hidden>{{ $item->quantity }}</span>
                </div>
            @empty
                <div class="empty-cart-message">Váš košík je prázdný.</div>
            @endforelse
        </div>
        <div class="mb-2 mt-5 w-100">
            <button type="button" class="btn btn-warning w-100" data-bs-toggle="modal"
                data-bs-target="#barcodeScannerModal">
                <i class="fas fa-barcode"></i> Skenovat
            </button>
        </div>
    </div>
</div>
```

```

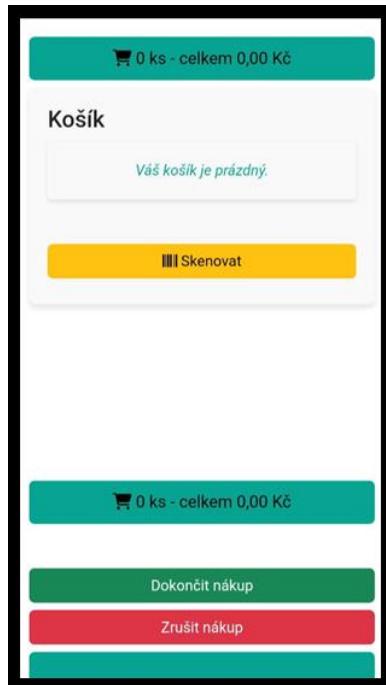
        </button>
    </div>
</div>

</div>
<div class="total-display mb-5">
    <div class="cart-info-content">
        <i class="fas fa-shopping-cart"></i>
        <span class="cart-details">
            {{ $totalQuantity ?? 0 }} ks - celkem {{ number_format($total ?? 0, 2,
',', ' ') }} Kč
        </span>
    </div>
</div>

<div class="mb-2 w-100">
    <button type="button" class="btn btn-success w-100" data-bs-toggle="modal"
        data-bs-target="#completeOrderModal">
        Dokončit nákup
    </button>
</div>
<div class="mb-2 w-100">
    <button type="button" class="btn btn-danger w-100" data-bs-toggle="modal" data-bs-
target="#cancelOrderModal">
        Zrušit nákup
    </button>
</div>

```

Skenování produktů je zahájeno stisknutím tlačítka „Skenovat“ (viz Obrázek 52). Po stisknutí tlačítka se zobrazí modální okno s čtečkou QuaggaJS. Ve chvíli zobrazení modálního okna začíná živý přenos z kamery zařízení a je aktivováno skenování.



Obrázek 54 View shopping

Čtečka (viz Obrázek 55) při detekci kódu zpracovává každý načtený kód, přičemž kontroluje, zda kód nebyl již předtím zpracován. Tato kontrola je implementována pomocí proměnné `lastScannedCode`, která uchovává poslední načtený kód, aby se zabránilo jeho opakovanému zpracování (viz Výpis kódu 32).



Obrázek 55 Čtečka čárových kódů

Výpis kódu 32 Funkce pro skenování čárových kódů

```
document.addEventListener('DOMContentLoaded', function () {
```

```

var scannerActivated = false;
var lastScannedCode = '';

function startScanner() {
    Quagga.init({
        inputStream: {
            name: "Live",
            type: "LiveStream",
            target: document.querySelector('#interactive'),
            constraints: {
                facingMode: "environment"
            },
        },
        decoder: {
            readers: ["code_128_reader"]
        },
    }, function (err) {
        if (err) {
            console.error(err);
            return;
        }
        Quagga.start();
        scannerActivated = true;
    });
}

Quagga.onDetected(function (data) {
    var code = data.codeResult.code;
    if (code !== lastScannedCode) {
        lastScannedCode = code;
        fetchProductInfo(code);
    }
});
}

$('#barcodeScannerModal').on('shown.bs.modal', function () {
    if (!scannerActivated) {
        startScanner();
    }
});

$('#barcodeScannerModal').on('hidden.bs.modal', function () {
    if (scannerActivated) {
        Quagga.stop();
        scannerActivated = false;
        lastScannedCode = '';
    }
});

```

```
        }
    });
});
```

Pokud je nalezen nový kód, funkce `fetchProductInfo` pošle požadavek na backendovou funkci `getProductByBarcode` v `ProductController` (viz Výpis kódu 33). Pokud je nalezen produkt s čárovým kódem, který odpovídá kódu ze čtečky, jsou vráceny informace jako název produktu, cena, sazba DPH, popis, balení a případně i sleva.

Dané informace jsou následně zobrazeny uživateli v modálním okně (viz Obrázek 56) s možností přidat libovolné množství do košíku, volitelné pomocí šipek u čísla znázorňující množství – toto řešení bylo zvoleno po konzultaci s vedoucím této práce a se zadavatelem projektu, jedná se o uživatelsky nejpříjemnější styl zvolení množství produktu.

#### Výpis kódu 33 Funkce `getProductByBarcode` v `ProductController`

```
public function getProductByBarcode(Request $request, $barcode)
{
    $product = Product::with('photos')->where('barcode', $barcode)->first();

    if ($product) {
        $images = $product->photos->map(function ($photo) {
            return ['url' => $photo->path];
        });

        $priceWithDiscount = $product->discount > 0 ?
            $product->price - ($product->price * ($product->discount / 100)) :
            $product->price;
        return response()->json([
            'success' => true,
            'product' => [
                'id' => $product->id,
                'name' => $product->name,
                'description' => $product->description,
                'unit' => $product->unit,
                'price' => number_format($product->price, 2),
                'priceWithDiscount' => number_format($priceWithDiscount, 2),
                'priceExclTax' => number_format($product->price_excl_tax, 2),
                'tax_rate' => $product->tax_rate,
                'discount' => $product->discount,
                'images' => $images,
            ]
        ]);
}
```

```

    }

    return response()->json(['success' => false, 'message' => 'Produkt nenalezen.'],
404);
}

```



Obrázek 56 Modální okno s informacemi o produktu

Po zvolení množství produktu je vložení do košíku potvrzeno tlačítkem „Přidat do košíku“. V tomto okamžiku je pomocí JavaScript funkce `addToCart` (viz Výpis kódu 34) zaslán požadavek na server pro přidání nové položky do košíku. Skript načítá ID produktu a zadané množství z HTML prvků a následně odešle HTTP požadavek typu POST na server s těmito údaji a s ID aktuální nákupní relace.

Na serverové straně funkce `addToCart` nejprve zkонтroluje, zda je uživatel přihlášený. Pokud ano, načte parametry z požadavku a vyhledá nákupní košík podle id nákupní relace. Poté vyhledá produkt podle jeho id a spočítá jeho cenu, přičemž vezme v úvahu případné slevy. Pokud je produkt již v košíku, aktualizuje se jeho množství. Pokud ne, je vytvořena nová položka s příslušnými údaji (viz Výpis kódu 35).

#### Výpis kódu 34 JavaScriptová funkce `addToCart`

```

function addToCart() {
    var productId = document.getElementById('productId').value;
    var quantity = parseInt(document.getElementById('productQuantity').value,

```

```

    10);

if (!productId || quantity <= 0) {
    alert('Prosím, zadejte platné množství.');
    return;
}

var sessionId = document.getElementById('sessionId').value;

fetch('/add-to-cart', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
        'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-
token"]').getAttribute('content')
    },
    body: JSON.stringify({
        sessionId: sessionId,
        productId: productId,
        quantity: quantity
    })
})
.then(response => response.json())
.then(data => {
    if (data.success) {
        refreshCart()
        $('#productInfoModal').modal('hide');
    } else {
        showUniversalModal('Chyba', 'Zboží nebylo nalezeno.' + data.message,
false);
    }
})
.catch(error => {
    console.error('Error:', error);
});
}

```

Výpis kódu 35 Funkce addToCart v ShoppingController

```

public function addToCart(Request $request)
{
    if (!Auth::check()) {
        return response()->json(['success' => false, 'message' => 'You must be logged
in to add items to the cart.']);
    }
}

```

```

}

$productId = $request->input('productId');
$quantity = $request->input('quantity', 1);
$sessionId = $request->input('sessionId');

$cart = ShoppingCart::where('session_id', $sessionId)->first();

if (!$cart) {
    return response()->json(['success' => false, 'message' => 'Shopping cart not found.']);
}

$product = Product::find($productId);
if (!$product) {
    return response()->json(['success' => false, 'message' => 'Product not found.']);
}

$finalPrice = $product->discount > 0 ? $product->price * (1 - ($product->discount / 100)) : $product->price;

$cartItem = $cart->items()->where('product_id', $productId)->first();
if ($cartItem) {
    $cartItem->quantity += $quantity;
    $cartItem->save();
} else {
    $cart->items()->create([
        'product_id' => $productId,
        'quantity' => $quantity,
        'price' => $finalPrice
    ]);
}

$cart->save();

return response()->json([
    'success' => true,
    'message' => 'Product added to cart successfully.',
    'cartItems' => $cart->items->load('product'),
    'total' => $cart->items->sum(function ($item) {
        return $item->quantity * $item->price;
    })
]);
}

```

Pokud ze serveru byla vrácena úspěšná odpověď, je zavolána funkce refreshCart (viz Výpis kódu 36), která aktualizuje zobrazení nákupního košíku s nově přidaným produktem a také aktualizuje celkový výpis množství produktů v košíku a jeho celkovou cenu (viz Obrázek 57).

#### Výpis kódu 36 Funkce refreshCart

```
function refreshCart() {
    var sessionId = document.getElementById('sessionId').value;

    fetch(`/refresh-cart?sessionId=${sessionId}`)
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                const cartItemsDiv = document.getElementById('cartItems');
                cartItemsDiv.innerHTML = '';

                let newTotal = 0;
                let totalQuantity = 0;

                data.cartItems.forEach(item => {
                    const itemElement = document.createElement('div');
                    itemElement.className = 'cart-item';
                    itemElement.setAttribute('onclick', `openEditModal(${item.id})`);

                    const finalPrice = item.discount > 0 ? item.price * (1 - item.discount / 100) : item.price;
                    const totalItemPrice = item.quantity * finalPrice;

                    newTotal += totalItemPrice;
                    totalQuantity += item.quantity;

                    const formatter = new Intl.NumberFormat('cs-CZ', {
                        style: 'currency',
                        currency: 'CZK',
                        minimumFractionDigits: 2,
                    });

                    itemElement.innerHTML =
                        `${item.name} - ${item.quantity} x
${formatter.format(finalPrice)}</span><br>
<span>Celkem: ${formatter.format(totalItemPrice)}</span>
<span id="itemQuantity${item.id}" hidden>${item.quantity}</span>
`;
                    cartItemsDiv.appendChild(itemElement);
                });
            }
        });
}
```

```

});
```

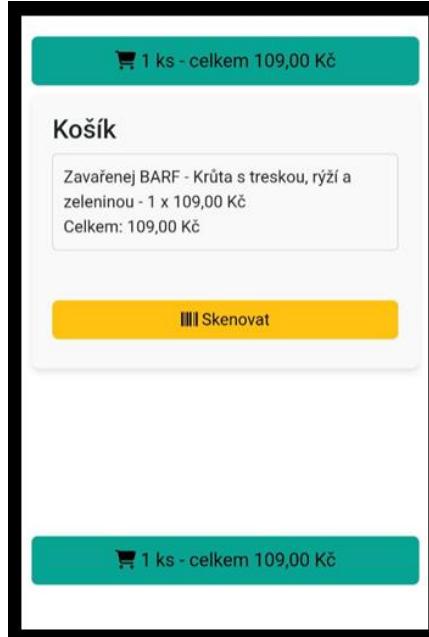
```

const formatter = new Intl.NumberFormat('cs-CZ', {
    style: 'currency',
    currency: 'CZK',
    minimumFractionDigits: 2,
});
```

```

const totalDisplays = document.querySelectorAll('.total-display .cart-
details');
totalDisplays.forEach(display => {
    display.textContent = `${totalQuantity} ks - celkem
${formatter.format(newTotal)} `;
});
} else {
    showUniversalModal('Chyba', 'Chyba při obnově košíku, manuálně obnovte
stránku.', false);
}
})
.catch(error => console.error('Error:', error));
}

```



Obrázek 57 Obnovený košík s přidaným produktem

## 4.11 Úprava produktů v košíku

Úprava daného produktů v košíku je zahájena kliknutím na dlaždici produktu v košíku. Z prvku je načteno id položky v košíku, které je odesláno na server pomocí JavaScriptové funkce `openInfoModal` (viz Výpis kódu 37). Pokud je `id` položky nalezeno pomocí funkce `getCartItemInfo` v `ProductController`, je vrácena odpověď obsahující veškeré informace o produktu, které jsou následně zobrazeny v modálním okně s možností úprav kusů daného zboží v košíku (viz. Obrázek 58).

Výpis kódu 37 JavaScriptová funkce `openInfoModal`

```
function openEditModal(cartItemId) {
    fetch(`/getCartItemInfo/${cartItemId}`)
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                const product = data.product;
                const priceWithDiscount = product.discount > 0
                    ? (product.price - (product.price * (product.discount / 100)))
                    : product.price;
                const originalPrice = product.price;
                const unit = product.unit;

                document.getElementById('productNameCartItem').textContent = product.name;
                document.getElementById('productDescriptionCartItem').innerHTML =
product.description.replace(/\n/g, '<br>');
                document.getElementById('productPriceCartItem').innerHTML =
product.discount > 0
                    ? `Cena: <del>${originalPrice} Kč</del> <strong>${priceWithDiscount} Kč</strong>`
                    : `Cena: ${originalPrice} Kč`;
                document.getElementById('productUnitCartItem').textContent = `Balení: ${unit}`;

                document.getElementById('cartItemId').value = cartItemId;
                document.getElementById('productQuantityUpdate').value = data.quantity;
                document.getElementById('productQuantityDisplayUpdate').textContent =
data.quantity;

                $('#cartItemModal').modal('show');
            } else {
                alert('Informace o produktu nebyly nalezeny.');
            }
        })
        .catch(error => console.error('Error:', error));
}
```

Výpis kódu 38 Funkce getCartItemInfo v ProductController

```
public function getCartItemInfo(Request $request, $cartItemId)
{
    $cartItem = CartItem::with('product.photos')->find($cartItemId);

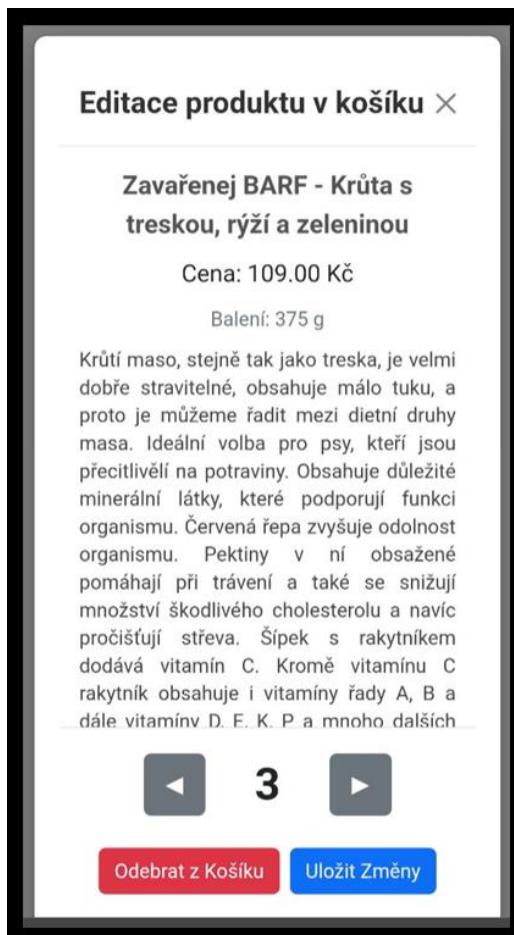
    if ($cartItem && $cartItem->product) {
        $product = $cartItem->product;

        $images = $product->photos->map(function ($photo) {
            return ['url' => $photo->path];
        });

        $priceWithDiscount = $product->discount > 0
            ? $product->price - ($product->price * ($product->discount / 100))
            : $product->price;

        return response()->json([
            'success' => true,
            'product' => [
                'id' => $product->id,
                'name' => $product->name,
                'description' => $product->description,
                'unit' => $product->unit,
                'price' => number_format($product->price, 2),
                'priceWithDiscount' => number_format($priceWithDiscount, 2),
                'priceExclTax' => number_format($product->price_excl_tax, 2),
                'tax_rate' => $product->tax_rate,
                'discount' => $product->discount,
                'images' => $images,
            ],
            'quantity' => $cartItem->quantity
        ]);
    }

    return response()->json(['success' => false, 'message' => 'Položka košíku nebyla nalezena.'], 404);
}
```



Obrázek 58 Modální okno pro editaci množství produktu v košíku

Požadavek o změnu počtu zboží v košíku je odeslán po kliknutí na tlačítko „Uložit změny“. Funkce `updateCartItem` (viz Výpis kódu 39) odesílá požadavek na backendovou funkci `updateCartItem` v `ShoppingController` (viz Výpis kódu 40), která kontroluje zadané množství a podle toho aktualizuje položku nebo ji z košíku odstraňuje. Zboží je z košíku odstraněno i po použití tlačítka „Odebrat z košíku“, přičemž je využito funkce `removeProduct` (viz Výpis kódu 41), která nastavuje počet kusů na nula a odesílá požadavek na frontendovou funkci `updateCartItem`.

Po provedení změn je obnoven košík stejně jako v případě přidání nového zboží do košíku, tedy pomocí funkce `refreshCart` (viz Výpis kódu 36).

#### Výpis kódu 39 JavaScriptová funkce `updateCartItem`

```
function updateCartItem() {
    const cartItemId = $('#cartItemId').val();
    const quantity = $('#productQuantityUpdate').val();

    fetch('/update-cart-item', {
        method: 'POST',
        headers: {
```

```

        'Content-Type': 'application/json',
        'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-
token"]').getAttribute('content')
    },
    body: JSON.stringify({
        cartItemId: cartItemId,
        quantity: quantity
    })
})
.then(response => response.json())
.then(data => {
    if (data.success) {
        $('#cartItemModal').modal('hide');
        refreshCart()
    } else {
        alert('Chyba při aktualizaci produktu: ' + data.message);
    }
});
}

```

Výpis kódu 40 Funkce updateCartItem v ShoppingController

```

public function updateCartItem(Request $request)
{
    if (!Auth::check()) {
        return response()->json(['success' => false, 'message' => 'Musíte být
přihlášeni k aktualizaci položky v košíku.']);
    }

    $cartItemId = $request->input('cartItemId');
    $quantity = $request->input('quantity');

    $cartItem = CartItem::find($cartItemId);

    if (!$cartItem) {
        return response()->json(['success' => false, 'message' => 'Položka nebyla
nalezena.']);
    }

    if ($quantity > 0) {
        $cartItem->quantity = $quantity;
        $cartItem->save();
        return response()->json(['success' => true, 'message' => 'Množství bylo
úspěšně aktualizováno.']);
    } else if ($quantity == 0) {

```

```
    $cartItem->delete();
    return response()->json(['success' => true, 'message' => 'Položka byla úspěšně
odstraněna z košíku.']);
} else {
    return response()->json(['success' => false, 'message' => 'Neplatné
množství.']);
}
}
```

Výpis kódu 41 JavaScriptová funkce removeProduct

```
function removeProduct() {
    document.getElementById('productQuantityUpdate').value = 0;
    document.getElementById('productQuantityDisplayUpdate').textContent = 0;
    updateCartItem();
}
```

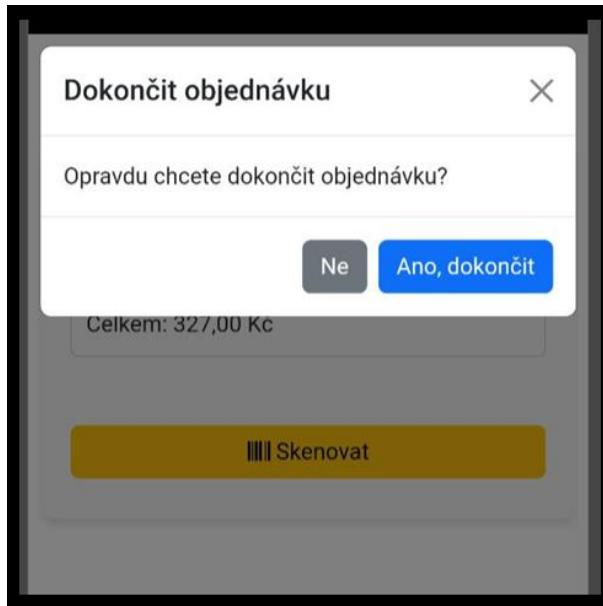
## 4.12 Ukončení nákupu

Před začátkem této podkapitoly je nutné říci, že tato část aplikace nebyla kvůli problémům s nasazením a získáním přístup k platební bráně rádně implementována a je potřeba jí před plnohodnotným nasazením upravit. Podrobněji je problém s nasazením aplikace do provozu popsán v kapitole 5 Nasazení aplikace.

Proces ukončení nákupu začíná v nákupním košíku, kde uživatelé mohou kliknout na tlačítko „Dokončit nákup“. Tlačítko vyvolá zobrazení modálního okna `completeOrderModal` (viz Výpis kódu 42, Obrázek 59), které se ptá uživatele, zda chce skutečně dokončit objednávku. Pokud uživatel potvrdí tlačítkem „Ano, dokončit“, je odeslán požadavek na server pomocí formuláře v modálním okně.

Výpis kódu 42 Modální okno completeOrderModal

```
<div class="modal fade" id="completeOrderModal" tabindex="-1" aria-
labelledby="completeOrderLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="completeOrderLabel">Dokončit objednávku</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
            </div>
            <div class="modal-body">
                Opravdu chcete dokončit objednávku?
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Ne</button>
                <form action="/completeCheckout" method="POST">
                    @csrf
                    <input type="hidden" name="sessionId" value="{{ $sessionId }}>
                    <button type="submit" class="btn btn-primary">Ano, dokončit</button>
                </form>
            </div>
        </div>
    </div>
</div>
```



Obrázek 59 Modální okno completeOrderModal

Na serverové straně funkce `completeCheckout` v `ShoppingController` (viz Výpis kódu 43) zpracovává požadavek a ověřuje platné id nákupní relace. Dále funkce načítá košík s jeho položkami a kontroluje, zda není prázdný. Pokud je košík prázdný, uživatel je přesměrován zpět na úvodní stránku s chybovou zprávou.

Výpis kódu 43 Funkce `completeCheckout` v `ShoppingController`

```
public function completeCheckout(Request $request)
{
    $sessionId = $request->input('sessionId');
    if (!$sessionId) {
        return redirect('/')->with('error', 'Chyba v nákupním procesu. Zkuste to
prosím znova.');
    }
    $session = ShoppingSession::with('store')->where('id', $sessionId)->first();

    $cart = ShoppingCart::with('items.product')->where('session_id', $sessionId)->first();
    if (!$cart || $cart->items->isEmpty()) {
        $request->session()->forget('sessionId');
        return redirect()->route('/')->with('error', 'Nelze dokončit objednávku,
protože váš košík je prázdný.');
    }

    $order = Order::create([
        'user_id' => Auth::id(),
        'total' => $cart->items->sum(function ($item) {
            return $item->quantity * ($item->product->price - ($item->product->price *
($item->product->discount / 100)));
    })
]);
```

```

        });

        'status' => 'pending'
    ]);

    foreach ($cart->items as $cartItem) {
        $discountedPrice = $cartItem->product->price * (1 - ($cartItem->product->discount / 100));
        $order->items()->create([
            'product_id' => $cartItem->product_id,
            'quantity' => $cartItem->quantity,
            'price' => $cartItem->product->price,
            'discounted_price' => $discountedPrice,
            'discount_rate' => $cartItem->product->discount
        ]);
    }

}

$this->sendInvoiceMail($order);

return $this->redirectToPaymentGateway($order, $sessionId);
}

```

V případě, že je košík platný a obsahuje položky, funkce `completeCheckout` vytvoří novou objednávku s celkovou cenou a stavem „pending“. Poté přidá do objednávky každou položku z košíku, včetně vypočítané slevy, pokud je dostupná. Jakmile je objednávka vytvořena, zavolá funkci `sendInvoiceMail` (viz Výpis kódu 44) pro vygenerování a zaslání faktury jako PDF dle předlohy `invoice` (viz Výpis kódu 45) na e-mailovou adresu uživatele.

Výpis kódu 44 Funkce `sendInvoiceMail` v `ShoppingController`

```

protected function sendInvoiceMail(Order $order)
{
    $pdf = PDF::loadView('emails/invoice', compact('order'));

    \Mail::send([], [], function ($message) use ($order, $pdf) {
        $message->to($order->user->email)
            ->subject('Faktura - objednávka #' . $order->id)
            ->attachData($pdf->output(), 'faktura-objednavka-' . $order->id . '.pdf');
    });
}

```

Výpis kódu 45 Šablon pro PDF fakturu

```

<!DOCTYPE html>
<html lang="cs">
<head>

```

```
<meta charset="UTF-8">
<title>Faktura Objednávka #{{ $order->id }}</title>
<style>
    body {
        font-family: 'DejaVu Sans', sans-serif;
        color: #333;
    }
    .header {
        width: 100%;
        text-align: center;
    }
    .section {
        margin-bottom: 20px;
    }
    .left-column {
        float: left;
        width: 50%;
    }
    .right-column {
        float: right;
        width: 50%;
    }
    .clear {
        clear: both;
    }
    table {
        width: 100%;
        border-collapse: collapse;
    }
    th, td {
        border: 1px solid #ddd;
        padding: 8px;
        text-align: left;
    }
    th {
        background-color: #f2f2f2;
    }
    .footer {
        text-align: center;
        margin-top: 30px;
    }
    .signature {
        text-align: right;
        margin-top: 50px;
    }

```

```

</style>
</head>
<body>
    <div class="header">
        <h1>Faktura - daňový doklad</h1>
    </div>

    <div class="section">
        <div class="left-column">
            <h2>Dodavatel</h2>
            <p>Do psí misky s.r.o.</p>
            <p>Láz 266, 262 41, Bohutín</p>
            <p>IČO: 06183816</p>
            <p>DIČ: CZ06183816</p>
        </div>

        <div class="right-column">
            <h2>Odběratel</h2>
            <p>{{ $order->user->first_name }} {{ $order->user->last_name }}</p>
            <p>{{ $order->user->address }}</p>
            <p>{{ $order->user->city }}, {{ $order->user->zip }}</p>
            <p>Tel: {{ $order->user->phone }}</p>
        </div>
        <div class="clear"></div>
    </div>

    <div class="section">
        <p>Objednávka #{{ $order->id }}</p>
        <p>Způsob platby: Platební brána GoPay</p>
    </div>

    <div class="section">
        <table>
            <thead>
                <tr>
                    <th>Produkt</th>
                    <th>Množství</th>
                    <th>Kč/ks bez DPH</th>
                    <th>DPH</th>
                    <th>Kč/ks s DPH</th>
                    <th>Sleva</th>
                    <th>Celkem</th>
                </tr>
            </thead>
            <tbody>

```

```

@foreach ($order->items as $item)
@php
$priceExclTax = $item->product->price / (1 + $item->product->tax_rate
/ 100);
$discountAmount = $item->product->price - $item->discounted_price;
$totalPriceExclTax = $item->quantity * $priceExclTax;
$totalDiscountedPrice = $item->quantity * $item->discounted_price;
@endphp
<tr>
<td>{{ $item->product->name }}</td>
<td>{{ $item->quantity }}</td>
<td>{{ number_format($priceExclTax, 2, ',', ' ') }} Kč</td>
<td>{{ $item->product->tax_rate }}%</td>
<td>{{ number_format($item->product->price, 2, ',', ' ') }} Kč</td>
<td>{{ number_format($item->discount_rate) }}%</td>
<td>{{ number_format($totalDiscountedPrice, 2, ',', ' ') }} Kč</td>
</tr>
@endforeach
</tbody>
</table>
</div>
<div class="signature">
<p>Celkem k úhradě: {{ number_format($order->total, 2, ',', ' ') }} Kč</p>
</div>
</body>
</html>
```

Po odeslání mailu je zavolána funkce pro inicializaci platby skrze platební bránu redirectToPaymentGateway, která předává potřebné údaje jako požadavek na platební bránu. Pokud je požadavek úspěšný, funkce aktualizuje stav nákupní relace na „created\_request“ a uživatele přesměruje na URL platební brány (viz Výpis kódu 46). Pokud požadavek selže, je uživatel přesměrován na domovskou stránku společně s chybovou zprávou. Informace o platební bráne funkce přebírá z .env souboru pomocí klíčů GOPAY\_GOID, GOPAY\_CLIENT\_ID, GOPAY\_CLIENT\_SECRET a GOPAY\_GATEWAY\_URL.

Výpis kódu 46 Funkce redirectToPaymentGateway v ShoppingController

```

protected function redirectToPaymentGateway(Order $order, $sessionId)
{
    $gopay = Api::payments([
        'goid' => env('GOPAY_GOID', ''),
        'clientId' => env('GOPAY_CLIENT_ID', ''),
        'clientSecret' => env('GOPAY_CLIENT_SECRET', ''),
```

```

        'gatewayUrl' => env('GOPAY_GATEWAY_URL', 'https://gw.sandbox.gopay.com/api'),
        'scope' => TokenScope::ALL,
        'language' => Language::CZECH,
    ]);

    $response = $gopay->createPayment([
        'amount' => $order->total * 100,
        'currency' => 'CZK',
        'order_number' => $order->id,
        'order_description' => 'Vaše objednávka č. ' . $order->id,
        'items' => $order->items->map(function ($item) {
            return [
                'type' => 'ITEM',
                'name' => $item->product->name,
                'amount' => $item->product->price * $item->quantity * 100,
                'count' => $item->quantity,
            ];
        })->toArray(),
        'callback' => [
            'return_url' => route('order.return'),
            'notification_url' => route('order.notify')
        ]
    ]);

    if ($response->hasSucceed()) {
        ShoppingSession::where('id', $sessionId)->update(['status' =>
'created_request']);
        ShoppingSession::where('id', $sessionId)->update(['active' => FALSE]);
        ShoppingSession::where('id', $sessionId)->update(['ended_at' => now()]);
        return redirect($response->json['gw_url']);
    } else {
        Log::error('GoPay error: ' . $response->statusCode);
        return redirect('/')->with('error', 'Chyba při zahájení procesu platby: ' .
$response->statusCode);
    }
}

```

Pokud uživatel vybral možnost „Zrušit nákup“ a tuto možnost potvrdil v modálním okně `cancelOrderModal` (viz Výpis kódu 47), je předán požadavek funkci `cancelCheckout` v `ProductController` (viz Výpis kódu 48), která současnou relaci ukončí a uživatele vrátí na domovskou stránku společně s informací o ukončení nákupu.

#### Výpis kódu 47 Modální okno cancelOrderModal

```
<div class="modal fade" id="cancelOrderModal" tabindex="-1" aria-labelledby="cancelOrderLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="cancelOrderLabel">Zrušit objednávku</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
            </div>
            <div class="modal-body">
                Opravdu chcete zrušit objednávku?
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Ne</button>
                <form action="{{ route('checkout.cancel') }}" method="POST">
                    @csrf
                    <input type="hidden" name="sessionId" value="{{ $sessionId }}>
                    <button type="submit" class="btn btn-danger">Ano, zrušit</button>
                </form>
            </div>
        </div>
    </div>
</div>
```

#### Výpis kódu 48 Funkce cancelCheckout v ShoppingController

```
public function cancelCheckout(Request $request)
{
    $sessionId = $request->input('sessionId');
    ShoppingSession::where('id', $sessionId)->update(['status' => 'cancelled']);
    ShoppingSession::where('id', $sessionId)->update(['active' => FALSE]);
    ShoppingSession::where('id', $sessionId)->update(['ended_at' => now()]);

    return redirect('/')->with('message', 'Objednávka byla zrušena.');
}
```

# 5 Nasazení aplikace

Nasazení aplikace do testovacího provozu po její implementaci bylo řešeno se zadavatelem skrze několik schůzek a konzultací, které se vedly od začátku práce. Se zadavatele bylo domluveno, že aplikace bude spuštěna na hostingu Wedos, jelikož zde již má vedenou svoji doménu [www.dopsimisky.eu](http://www.dopsimisky.eu).

## Vývojové prostředí

Jako vývojové prostředí byl využit domácí server autora, na kterém testuje své projekty. Pro vývoj aplikace bylo vyhrazeno 4 GB RAM, 128 GB místa na disku a 4 procesorová jádra, jako operační systém byl využit Debian 12.5. Toto prostřední plně vyhovuje nárokům aplikace. Toto prostředí bylo následně využito i pro testovací spuštění aplikace, dokud aplikace nebyla přesunuta na webové prostředí zadavatele. V tuto chvíli domácí server autora funguje jako prostředí pro testování nových funkcí aplikace před nasazením na hosting.

## Hosting

Aplikace je hostována na VPS (Virtual Private Server) společnosti Wedos v základní konfiguraci s 1 GB RAM, 5GB místa na disku a jedním dostupným jádrem procesoru, využitý operační systém je Debian 11, nejvyšší poskytovaná verze společností Wedos pro své servery. Tato konfigurace byla vybrána pro co největší snížení nákladů pro provoz aplikace, dokud nebude plnohodnotně nasazena do provozu, v budoucnu je plánováno zvýšit kapacitu HW dle každodenního zatížení aplikace.

Aplikace je nasazena ve své testovací podobě bez platební brány na adresu aplikace.dopsimisky.eu.

## Platební brána

Dle doporučení autora práce si zadavatel vybral platební bránu GoPay jako jeho řešení pro provádění plateb. Po dokončení procesu kontroly webu, samotného principu provozu aplikace a modelu firmy bylo společností GoPay rozhodnuto, že přístup k platební bráně nebude zadavateli práce přidělen z důvodu rizikovosti celé aplikace.

V tuto chvíli již ze strany zadavatele probíhá proces získání přístupu k platební bráně společnosti Comgate a je odhadováno, že do konce května roku 2024 bude aplikace nasazena do provozu pro vybranou skupinu zákazníků společně s platební bránou Comgate. Pro provedení této změny bude potřeba odstranit implementaci gopay/payments-sdk-php a následně implementovat novou backend logiku.

## Budoucnost aplikace

Před zkušebním zavedením aplikace uzavřené skupině zákazníků bude potřeba upravit některé funkcionality aplikace, jako jsou vzory faktur, či mailů. Také již byl vzneseno několik dalších funkcí, které by zadavatel chtěl v aplikaci mít, jako je možnost podrobného sledování

obratů, nejlépe v grafech, tvořených automaticky dle jeho zadaných parametrů, možnost kontaktovat zákazníky přímo skrze aplikaci a i obráceně, kontaktovat podporu skrze aplikaci v případě potřeby, nebo třeba odchod a vstup do prodejny zjednodušit automatickým odemykáním a zamykáním dveří prodejen, čímž se sníží i riziko odcizení produktů z prodejny.

# 6 Testování aplikace

Tato kapitola prezentuje výsledky testování aplikace nasazené do testovacího provozu na odkazu aplikace.dopsimisky.eu. Testování bylo rozděleno do dvou oblastí, a to testování pomocí předem definovaných scénářů a testování výkonu a zabezpečení aplikace.

## 6.1 Testování podle předem definovaných scénářů

Pro testování aplikace byla vybrána skupina 6 nezávislých testerů. Pro každého z testerů byly jasně definovány testovací scénáře, které byly rozděleny do dvou kategorií – pro standardního uživatele a pro administrátora.

Při testování byla využita zařízení samotných testerů. Následující tabulka (viz Tabulka 2) znázorňuje zařízení, které sada testerů využívala. Lze vidět, že je poskytnuté široké spektrum testovacích zařízení, na který lze aplikaci spolehlivě vyzkoušet.

Tabulka 2 Přehled zařízení testerů

Tester	Mobilní zařízení	OS mobilního zařízení	Webový prohlížeč m.z.	Typ PC	OS počítače	Webový prohlížeč PC
Tester 1	Xiaomi redmi 11 Pro	Android 13	Google Chrome	Přenosný	Windows 10	Google Chrome
Tester 2	Apple iPhone XR	iOS 16	Safari	Přenosný	Windows 10	Google Chrome
Tester 3	Samsung Galaxy A23	Android 12	Google Chrome	Přenosný	Windows 11	Mozilla Firefox
Tester 4	Xiaomi redmi Note 11	Android 13	Mozilla Firefox	Stolní	Debian 12.5	Mozilla Firefox
Tester 5	Xiaomi Mi 8T	Android 11	Google Chrome	Stolní	Windows 7	Google Chrome
Tester 6	Sony Xperia 5 V	Android 13	Google Chrome	Přenosný	Windows 10	Opera GX

## Testovací scénáře

Pro autora práce bylo důležité otestovat zásadní funkcionality aplikace, které jsou nejvíce používány a jsou v současné implementaci dostupné (viz Nasazení aplikace). Následující tabulky U1 až U5 a A1 až A4 popisují jednotlivé scénáře, které ověřují základní funkcionalitu aplikace. Scénáře jsou navrhnuté tak, aby byly co nejjednodušší pro testery, přičemž všichni testeři byly obeznámeni s aplikací a byly jim vysvětleny klíčové body, či části, které jim nebyly jasné.

V případě nalezení chyby v průběhu testování byla chyba autorem zaznamenána pro opravu, to samé platí pro zpětnou vazbu testerů.

Tabulka 3 Testovací scénář U1 – registrace uživatele a aktivace účtu

Název	Registrace uživatele a aktivace účtu
Kroky	<ol style="list-style-type: none"><li>Uživatel si zobrazí domovskou stránku a skrze přihlašovací dialog vybere možnost registrace</li><li>Uživatel vyplní údaje a odešle formulář</li><li>Uživatel aktivuje svůj účet skrze potvrzovací mail</li></ol>
Očekávaný výsledek	Uživatel je zaregistrován a má aktivní účet

Tabulka 4 Testovací scénář U2 – zobrazení přístupového kódu prodejny a zahájení nákupu

Název	Zobrazení přístupového kódu prodejny
Kroky	<ol style="list-style-type: none"><li>Uživatel se přihlásí</li><li>Uživatel zahájí nákup pomocí tlačítka na domovské obrazovce</li><li>Uživatel zvolí možnost skenování</li><li>Uživatel načte dodaný QR kód</li><li>Uživatel potvrdí vstup do prodejny</li></ol>
Očekávaný výsledek	Uživateli je zobrazen aktuální přístupový kód od zámku prodejny, po potvrzení vstupu je zobrazeno nákupní rozhraní. V databázi je vytvořen záznam o nové nákupní relaci se stavem „active“.

Tabulka 5 Testovací scénář U3 – přidání zboží do košíku

Název	Naskenování čárového kódu a přidání zboží do košíku
Kroky	<ol style="list-style-type: none"> <li>1. Uživatel zahájil nákup</li> <li>2. Uživatel naskenuje čárový kód produktu</li> <li>3. Uživatel zadá množství přidaného zboží dle svého uvážení</li> <li>4. Uživatel přidá zboží do košíku</li> </ol>
Očekávaný výsledek	Uživateli je zobrazen aktualizovaný košík se zadáným počtem zboží, v databázi byla do tabulky cart_items přidána nová položka přiřazena nákupnímu košíku uživatele.

Tabulka 6 Testovací scénář U4 – změna počtu zboží v košíku

Název	Odstranění zboží z košíku
Kroky	<ol style="list-style-type: none"> <li>1. Uživatel zvolí zboží v košíku</li> <li>2. Uživatel zadá rozdílné množství zboží, než bylo v košíku</li> <li>3. Uživatel změnu uloží</li> </ol>
Očekávaný výsledek	Uživateli je zobrazen aktualizovaný košík se zadáným počtem zboží

Tabulka 7 Testovací scénář U5 – Zrušení nákupu

Název	Zrušení nákupu
Kroky	<ol style="list-style-type: none"> <li>1. Uživatel vybere možnost "Zrušení nákupu"</li> <li>2. Uživateli potvrdí volbu v modálním okně</li> </ol>
Očekávaný výsledek	Uživatel je přesměrován na úvodní stránku se zprávou o potvrzení zrušení nákupu, v databázi je nákupní relace označena jako „cancelled“

Tabulka 8 Testovací scénář A1 – přidání nového zboží

Název	Přidání nového zboží
Kroky	<ol style="list-style-type: none"> <li>1. Uživatel vybere možnost „Správa zboží“</li> <li>2. Uživatel vybere možnost „Přidat zboží“</li> <li>3. Uživatel vyplní formulář s volitelnými údaji a vybere náhodné fotografie produktu</li> <li>4. Uživatele přidá zboží do seznamu</li> </ol>
Očekávaný výsledek	Je vytvořeno nové zboží, které je následně zobrazeno v zobrazení „Správa Zboží“

Tabulka 9 Testovací scénář A2 – přidání nové prodejny

Název	Přidání nové prodejny
Kroky	1. Uživatel vybere možnost „Správa prodejen“ 2. Uživatel vybere možnost „Přidat prodejnu“ 3. Uživatel vyplní formulář potřebnými údaji 2. Uživatele přidá zboží do seznamu
Očekávaný výsledek	Je vytvořena nová prodejna, která je následně zobrazena v zobrazení „Správa prodejen“

Tabulka 10 Testovací scénář A3 – přidání zboží na sklad prodejny

Název	Přidání nového zboží na sklad prodejny
Kroky	1. Uživatel vybere možnost „Správa prodejen“ 2. Uživatel vybere prodejnu vytvořenou v testovacím scénáři A1 3. Uživatel vybere možnost „Přidat zboží“ 4. Uživatel vybere zboží vytvořené v testovacím scénáři A2 a zadá volitelný počet kusů, odešle požadavek
Očekávaný výsledek	Zboží vytvořené v testovacím scénáři A1 je přidáno do skladu prodejny vytvořené v testovacím scénáři A2 v zadaném množství.

Tabulka 11 Testovací scénář A4 – přidání role libovolnému uživateli

Název	Přidání role libovolnému uživateli v aplikaci
Kroky	1. Uživatel vybere možnost „Správa uživatelů“ 2. Uživatel vybere jiného libovolného uživatele a vybere u jeho záznamu "Spravovat role" 3. Uživatel přidá uživateli libovolnou roli 4. Uživatel potvrdí změny
Očekávaný výsledek	Uživateli je zobrazeno obnovené zobrazení výpisu uživatelů, zvolený uživatel vlastní zvolenou roli

## Výsledky testování

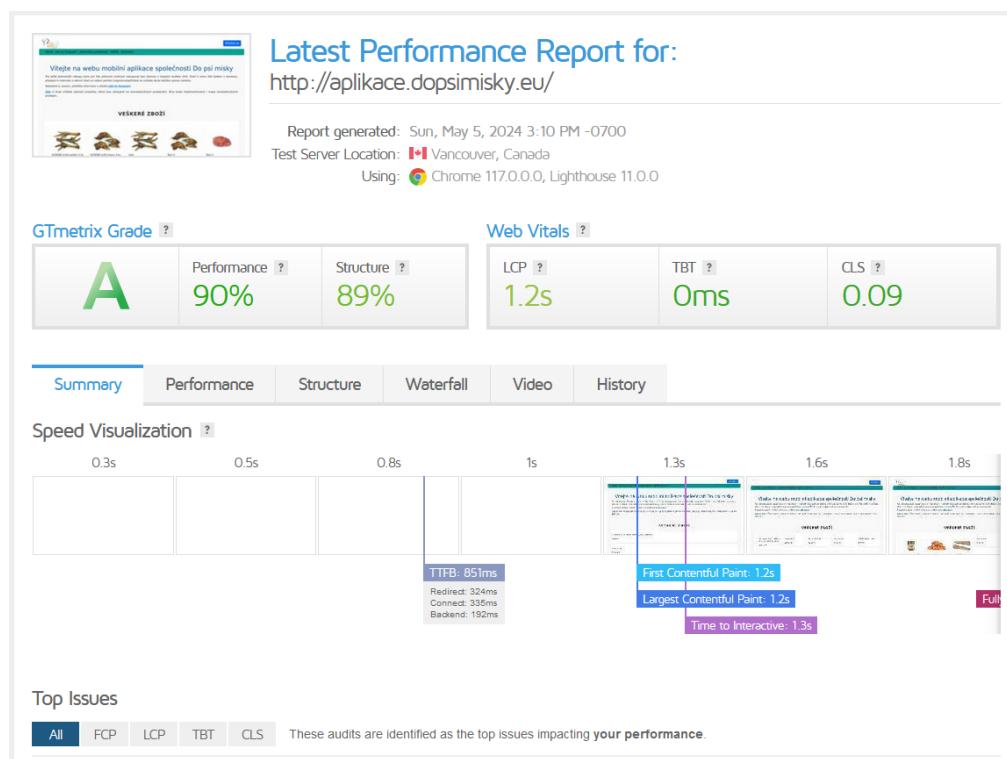
Po celou dobu testování nebyla odhalena chyba, která by způsobila nefunkčnost dané části aplikace. Všechny vstupy byly správně promítnuty jako záznamy do databáze a všechny údaje se správně vypisovaly v daných views. Uživatelé mohli volně nakupovat zboží pomocí aplikace, nebo provádět správu uživatelů, zboží a produktů.

## 6.2 Testování výkonu aplikace

Z hlediska využití aplikace pro nákup zboží je nezbytné, aby koncoví uživatelé zažili co nejplynulejší nákupní zážitek. Pokud je aplikace pomalá a její zpracování požadavků trvá dlouho, je velmi pravděpodobné, že uživatelé přestanou aplikaci používat.

Pro testování výkonu byla využita webová aplikace GTmetrix.com, což je oblíbený nástroj mezi webovými vývojáři, který analyzuje průběh načítání stránky a poskytuje skóre pro parametry jako jsou rychlosť, či optimalizace, ale také dokáže identifikovat problémy, které zbytečně zpomalují načítání a poskytuje návrhy pro zlepšení.

Webová aplikace, testována ze serverů umístěných v Kanadě, získala výkonnostní hodnocení 90 % a hodnocení struktury 89 %, což naznačuje velmi dobrý výkon i při přístupu k aplikaci z takto vzdáleného zařízení (viz Obrázek 60).



Obrázek 60 Výkonnostní testování aplikace (GTmetrix, 2024)

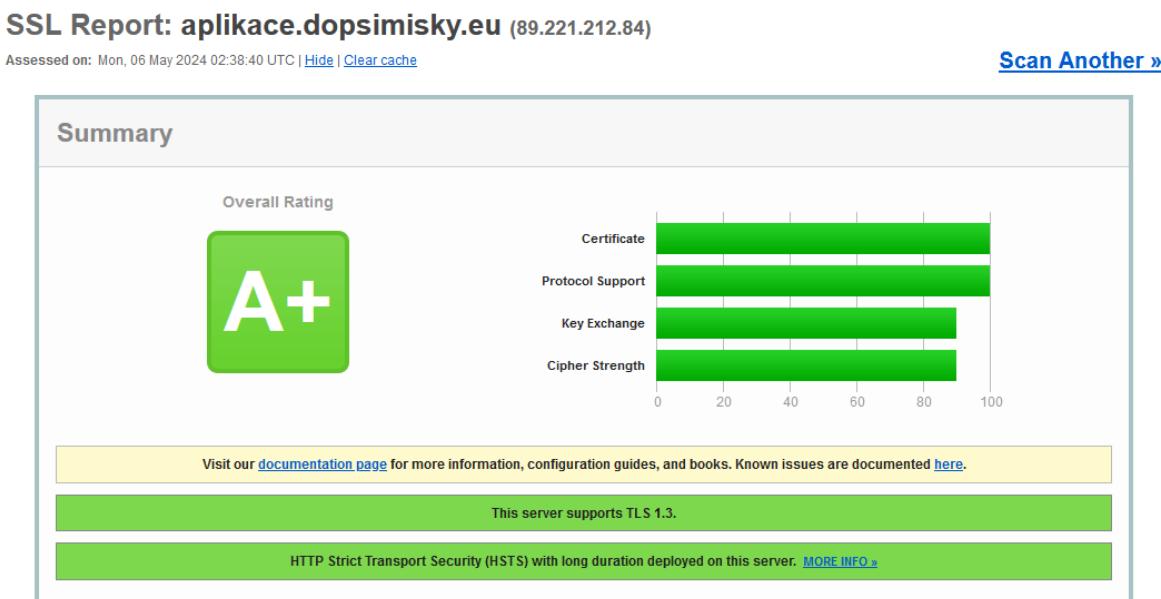
## 6.3 Testování bezpečnosti aplikace

Testování bezpečnosti aplikace probíhalo z pohledu přenosu dat od uživatele k serveru skrze nástroje Qualys SSL Server Test (Qualys, Inc., 2024) a ImmuniWeb (ImmuniWeb, 2024).

Aplikace SSL Server Test je online nástroj pro hodnocení úrovně zabezpečení SSL/TLS připojení na webových serverech. Nástroj provádí sérii testů a poskytuje podrobné informace o konfiguraci certifikátů, šifrovacích algoritmůch a potenciálních zranitelnostech. Celkové hodnocení aplikace zadává v rozmezí A+ až F, přičemž nasazená aplikace v testovacím prostředí získala hodnocení A (viz Obrázek 61).

ImmuniWeb je další online nástroj zaměřený na hodnocení zabezpečení webových aplikací, ale nejen na bezpečnost SSL/TLS, ale také na další aspekty zabezpečení webových aplikací. Analyzuje potenciální zranitelnosti, jako je XSS, SQL injekce, či chyby v konfiguraci serveru. Nabízí celkové hodnocení bezpečnosti webové aplikace a poskytuje podrobné zprávy s doporučeními k nápravě nalezených problémů. Aplikace nasazená v testovacím prostředí získala hodnocení A (viz Obrázek 62).

Na základě výsledků těchto testů je zřejmé, že aplikace splňuje potřebné úrovně zabezpečení, které jsou potřeba v moderních webových aplikacích.



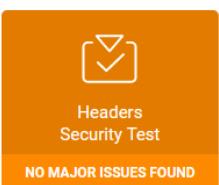
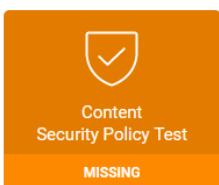
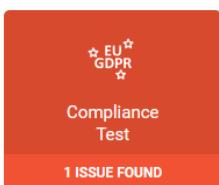
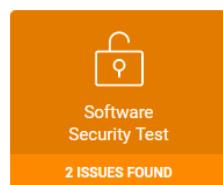
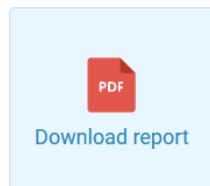
Obrázek 61 Výsledné hodnocení stránky aplikace.dopsimisky.eu pomocí SSL Server Test (Qualys, Inc., 2024)

## Summary of aplikace.dopsimisky.eu [Desktop version] Website Security Test

dopsimisky.eu was tested 8 times during the last 12 months.

### Your final score

Tested on: May 6th, 2024 04:22:44 GMT+2  
Server IP: 89.221.212.84  
Reverse DNS: -  
Location: Havířov   
Client: Desktop version



Obrázek 62 Výsledné hodnocení stránky aplikace.dopsimisky.eu pomocí ImmuniWeb (ImmuniWeb, 2024)

## 7 Závěr

Tato bakalářská práce se zabývala analýzou požadavků zadavatele, srovnáním dostupných řešení samoobslužných prodejen na trhu, návrhem a následnou implementací vlastní webové aplikace, která umožňuje zákazníkům nakupovat pomocí mobilních telefonů v prodejnách společnosti Do psí misky.

V první kapitole byl přiblížen princip a historie samoobslužného prodeje, jeho výhody a nevýhody, stejně jako výrazné zrychlení jeho vývoje od doby pandemie COVID-19.

Ve druhé kapitole byla představena společnost Do psí misky s.r.o., která hledala řešení pro provoz samoobslužných prodejních míst bez potřeby stálého personálu. Po konzultacích byly definovány funkční a nefunkční požadavky na aplikaci a byla provedena analýza dostupných řešení na trhu. V závěru kapitoly byly specifikovány prvky jiných aplikací, které by mohly být do projektu implementovány.

Třetí kapitola se zaměřuje na návrh samotné aplikace, přičemž byl sestaven use case diagram a mapy procesů. Byl také vytvořen návrh vzhledu aplikace v podobě wireframe konceptů, které byly po konzultacích upravovány, až se dosáhlo finálního vzhledu aplikace. Následně byly představeny technické prvky implementace, jako je databáze, provádění plateb a technologie PHP a JavaScript, které byly pro vývoj zvoleny.

Ve čtvrté kapitole je popsána implementace webové aplikace, včetně její architektury, klíčových komponent a funkcí, jako je zahájení nákupu, správa prodejen, uživatelů a zboží.

Pátá kapitola se věnuje nasazení aplikace a překážkám, které se s ním pojí, zejména zamítnutí implementované platební brány společnosti GoPay.

Šestá kapitola je zaměřena na testování aplikace. Bylo provedeno několik typů testů, které zahrnovaly funkční testování podle definovaných scénářů, testování výkonu aplikace a kontrolu bezpečnosti.

Cíl práce byl splněn, implementovaná aplikace naplňuje všechny požadavky zadavatele.

Autor práce je se zadavatelem stále v kontaktu a společně pracují na dalších úpravách aplikace a jejím rozšíření o další funkce. Hlavním cílem je nyní implementovat novou bránu od společnosti Comgate. Odhaduje se, že implementace proběhne do konce května 2024, přičemž bude aplikace spuštěna do provozu pro vybrané zákazníky zadavatele.

Zdrojový kód aplikace a návod na její spuštění jsou dostupné na GitHubu: <https://github.com/DoubleOTism/Bakalarska-prace-DPM>.

## 8 Použitá literatura

**Alva, Mike. 2021.** Impact of Ai on retail industry. *KPMG US*. [Online] 2021. [Citace: 22. 11 2023.] <https://info.kpmg.us/news-perspectives/technology-innovation/thriving-in-an-ai-world/ai-adoption-retail.html>.

**Comgate. 2024.** Platební brána. *Comgate*. [Online] 1. 1 2024. [Citace: 12. 2 2024.] <https://www.comgate.cz/platebni-brana>.

**FastCentrik. 2023.** Velké srovnání platebních bran v roce 2024: Která je pro váš e-shop ta nejlepší? *FastCentrik*. [Online] 5. 1 2023. [Citace: 12. 2 2024.] <https://www.fastcentrik.cz/blog/velke-srovnani-10-platebnich-bran>.

**Figma, Inc. 2024.** *Figma*. [Software] 2024.

**GoPay. 2023.** Poznejte nejpopulárnější platební metody pro váš e-shop! *GoPay*. [Online] 30. 3 2023. [Citace: 12. 2 2024.] <https://www.gopay.com/blog/poznejte-nejpopularnejsi-platebni-metody-pro-vas-e-shop/>.

**Haška, David. 2018.** *Vytvoření realtime webové aplikace pomocí frameworků ReactJS a Laravel*. Praha : Vysoká škola ekonomická v Praze, 2018.

**Cheng, Andria. 2019.** Why Amazon Go May Soon Change The Way We Shop. *Forbes*. [Online] Forbes, 19. Leden 2019. [Citace: 12. říjen 2023.] <https://www.forbes.com/sites/andriacheng/2019/01/13/why-amazon-go-may-soon-change-the-way-we-want-to-shop/>.

**Laravel Holdings Inc.** Installation - Laravel 10.x - The PHP Framework For Web Artisans. *Laravel*. [Online] [Citace: 12. 2 2024.] <https://laravel.com/docs/10.x>.

**Loxone. 2021.** Loxone - Autonomní prodejna díky automatizaci - Loxone blog. *Loxone*. [Online] Loxone, 4. Srpen 2021. [Citace: 21. Říjen 2023.] <https://www.loxone.com/cscz/blog/autonomni-prodejna/>.

—. Miniserver Datasheet. *Loxone*. [Online] [Citace: 21. Říjen 2023.] [https://www.loxone.com/wp-content/uploads/datasheets/Datasheet\\_Miniserver\\_100335.pdf](https://www.loxone.com/wp-content/uploads/datasheets/Datasheet_Miniserver_100335.pdf).

—. 2023. Obchod pro inteligentní elektroinstalace. *Loxone Eshop*. [Online] Loxone, 2023. [Citace: 20. 11 2023.] <https://shop.loxone.com/cscz/>.

**Ltd., JGraph. 2024.** *draw.io*. [Software] 2024.

**Michálek, M. 2017.** *Vzhůru do (responzivního) webdesignu (Verze 1.1.)*. Praha : autor neznámý, 2017.

**Nguyen, Tuan Kiet. 2023.** Webová platforma pro propojení pokladních systémů a platebních bran. Praha : Vysoká škola ekonomická v Praze, 2023.

**Otwell, Taylor.** Laravel Documentation. [Online] Laravel LLC.  
<https://laravel.com/docs/10.x>.

**Pan Oston. 2023.** The social impact of self-service in Retail. *Pan oston*. [Online] 13. 11 2023. [Citace: 22. 11 2023.] <https://panoston.co.uk/news/219/the-social-impact-of-self-service-in-retail>.

**PHP Group. 2023.** PHP 8 Changelog. *PHP*. [Online] 23. 11 2023. [Citace: 12. 2 2024.] <https://www.php.net/ChangeLog-8.php#8.3.0>.

—. What is PHP? *PHP*. [Online] [Citace: 11. 2 2024.] <https://www.php.net/manual/en/intro-whatis.php>.

**Picaro, Elyse Betters. 2024.** Amazon Go stores: How the 'Just walk out' cashierless tech works. *Pocket-lint*. [Online] 28. 1 2024. [Citace: 4. 2 2024.] <https://www.pocket-lint.com/what-is-amazon-go-where-is-it-and-how-does-it-work/>.

**PIXEVIA.** PIXEVIA: autonomous, unmanned, fully automated stores - PIXEVIA. *PIXEVIA*. [Online] PIXEVIA. [Citace: 12. Říjen 2023.] <https://www.pixevia.com/>.

**Podomarof, Piotr. 2023.** Types of Autonomous Stores & How to Choose the Best One for Your Business. *Netguru*. [Online] 5. Září 2023. [Citace: 13. Říjen 2023.] <https://www.netguru.com/blog/types-of-autonomous-stores>.

**PVA Systems. 2023.** Automatické 24/7 prodejny PVA Systems - KONKRÉTNÍ STÁNKA. *PVA Systems - JAK SE TO PLOŠNĚ JMENUJE*. [Online] 2023. [Citace: 21. 10. 2023.] <https://www.pvasystems.cz/automaticke-247-prodejny>.

—. Datové sklady, Business Inteligence PVA Systems. *PVA Systems*. [Online] PVA Systems. [Citace: 21. Říjen 2023.] <https://www.pvasystems.cz/datove-sklady-business-intelligence>.

—. Obchodně skladový a pokladní systém PVA Systems. *PVA Systems*. [Online] PVA Systems. [Citace: 21. Říjen 2023.] <https://www.pvasystems.cz/obchodne-skladovy-a-pokladni-system-2>.

**PYMNTS. 2021.** Deep Dive: How Self-Service Retail Has Advanced And Helped Consumers Amid The Pandemic. *PYMNTS*. [Online] 2021. 5 2021. [Citace: 22. 11 2023.] <https://www.pymnts.com/digital-payments/2021/deep-dive-self-service-retail-advanced-helped-consumers-amid-pandemic/>.

**R., Rajkumar. 2014.** Ancient Coin-Operated Holy Water Dispensing Machine . *Elixir of Knowledge*. [Online] 6. 6 2014. [Citace: 16. 1 2024.] <https://www.elixirofknowledge.com/2014/06/ancient-coin-operated-holy-water.html>.

**Smith, Ernie. 2015.** The History of Vending Machines Goes Back to the 1st Century. *Atlas Obscura*. [Online] 3. 8 2015. [Citace: 6. 1 2024.]

<https://www.atlasobscura.com/articles/the-history-of-vending-machines-goes-back-to-the-1st-century>.

**ThePay. 2024.** S vaším růstem klesá cena. *ThePay*. [Online] 1. 1 2024. [Citace: 12. 2 2024.] <https://www.thepay.eu/cs/cenove-hladiny/>.

**Trigo. 2022.** FAQ | Trigo. *Trigoretail*. [Online] Trigo, 2022. [Citace: 22. Říjen 2023.] <https://www.trigoretail.com/faq/>.

—. **2022.** Seamless Shopping Experience with EasyOut | Trigo. *Trigoretail*. [Online] Trigo, 2022. [Citace: 22. Říjen 2023.] <https://www.trigoretail.com/easyout/>.

—. **2023.** Stories. *Trigo*. [Online] 2023. [Citace: 20. 11 2023.] <https://www.trigoretail.com/stories/>.

**Tým Do psí misky. 2024.** Do psí misky. *Do psí misky*. [Online] 2024. [Citace: 11. 2 2024.] <https://www.dopsimisky.eu/>.

—. O nás. *Do psí misky s.r.o.* [Online] [Citace: 5. 2 2024.] <https://www.dopsimisky.eu/onas/>.

**Unimarkt.** Das ist die UNIBox | Unimarkt. *Unimarkt*. [Online] Unimarkt. [Citace: 21. Říjen 2023.] <https://unimarkt.at/das-ist-die-unibox/>.

**Viessmann.** Automatizované řešení nano-skladování | Viessmann. *Automatizované řešení nano-skladování*. [Online] Viessmann. [Citace: 12. Říjen 2023.] <https://chlazeni.viessmann.cz/cs-cz/nase-produktova-reseni/Reseni-pro-automatizovane-prodejny/Automatizovane-reseni-nano-skladovani>.

—. Autonomní prodejna. *Autonomní prodejna*. [Online] Viessmann. [Citace: 12. Říjen 2023.] <https://chlazeni.viessmann.cz/cs-cz/nase-produktova-reseni/Reseni-pro-automatizovane-prodejny/Autonomni-prodejna>.

Výpis z obchodního rejstříku Do psí misky s.r.o. *Veřejný rejstřík a sbírka listin*. [Online] [Citace: 2024. 2 3.] <https://or.justice.cz/ias/ui/rejstrik-firma.vysledky?subjektId=976993&typ=PLATNY>.